Machine Learning
Bias-Variance Trade Off

Dr. Rizwan Ahmed Khan

Outline

1. Debugging Model
   - Hypothesis Evaluation
2. Generalization Error
   - Expected Label
   - Learned Hypothesis
   - Expected Test Error
   - Expected Classifier
   - Expected Error of A
3. Test Error Decomposition
   - Decomposition of Expected Test Error
4. Model Selection
   - Understanding error
   - Optimum model complexity
   - Dealing with error
5. Tasks

Reference Books

Reference books for this Module:

- Chapter 3: Pattern Recognition, Theodoridis et al., Academic Press, 4^{th} Edition or latest edition.

Reference Books

Reference books for this Module:

- Chapter 3: Pattern Recognition, Theodoridis et al., Academic Press, 4^{th} Edition or latest edition.
- Chapter 3: Pattern Recognition and Machine Learning, Christopher M. Bishop, Springer Books, latest edition.

Reference Books

Reference books for this Module:

- Chapter 3: Pattern Recognition, Theodoridis et al., Academic Press, 4^{th} Edition or latest edition.
- Chapter 3: Pattern Recognition and Machine Learning, Christopher M. Bishop, Springer Books, latest edition.
- Chapter 2: The Elements of Statistical Learning, Hastie et al., Springer Books, 2^{nd} edition.

Reference Books

Reference books for this Module:

- Chapter 3: Pattern Recognition, Theodoridis et al., Academic Press, 4^{th} Edition or latest edition.
- Chapter 3: Pattern Recognition and Machine Learning, Christopher M. Bishop, Springer Books, latest edition.
- Chapter 2: The Elements of Statistical Learning, Hastie et al., Springer Books, 2^{nd} edition.
- Chapter 5: Machine Learning, Tom MITCHELL, McGraw Hill, latest edition.

Section Contents

How to select ML model

- Is it enough to know / understand how different machine learning algorithms work?
- How can we gain insight on whether selected model will perform adequately on unseen data i.e. generalization capabilities?
- What if model training error was within predefined bounds, but it makes unacceptably large error on unseen data?

How to select ML model

What if model training error was within predefined bounds, but it makes unacceptably large error on unseen data?

1. Get more training examples

How to select ML model

What if model training error was within predefined bounds, but it makes unacceptably large error on unseen data?

1. Get more training examples
2. Try smaller sets of features <curse of dimensionality>

How to select ML model

What if model training error was within predefined bounds, but it makes unacceptably large error on unseen data?

1. Get more training examples
2. Try smaller sets of features <curse of dimensionality>
3. Try getting additional features <blessing of dimensionality>

How to select ML model

What if model training error was within predefined bounds, but it makes unacceptably large error on unseen data?

1. Get more training examples
2. Try smaller sets of features <curse of dimensionality>
3. Try getting additional features <blessing of dimensionality>
4. Try adding polynomial features

How to select ML model

What if model training error was within predefined bounds, but it makes unacceptably large error on unseen data?

1. Get more training examples
2. Try smaller sets of features <curse of dimensionality>
3. Try getting additional features <blessing of dimensionality>
4. Try adding polynomial features
5. Try hyper-parameter tuning (tree depth, k in $k - NN$, degree of polynomial in regression, c in SVM etc.)

How to select ML model

What if model training error was within predefined bounds, but it makes unacceptably large error on unseen data?

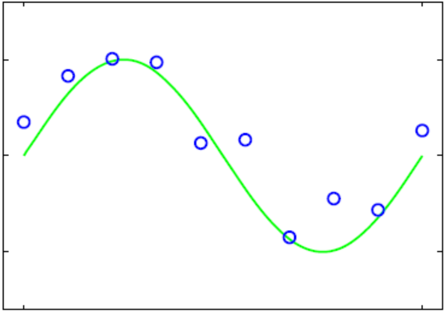
1. Get more training examples
2. Try smaller sets of features <curse of dimensionality>
3. Try getting additional features <blessing of dimensionality>
4. Try adding polynomial features
5. Try hyper-parameter tuning (tree depth, k in $k - NN$, degree of polynomial in regression, c in SVM etc.)

How to evaluate hypothesis

- Usually ML practitioners select any of a/m items randomly (gut feeling) to improve performance of ML algorithm, which most of the time do not give desired results.
- In this module we will tackle this aspect in scientific manner.

Hypothesis Evaluation

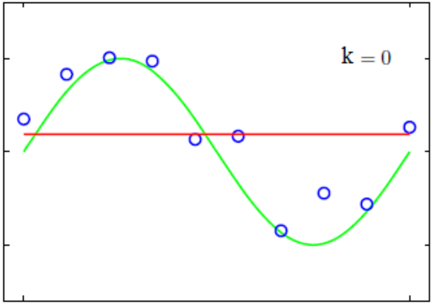
Consider example of "Regression", where we are trying to fits a linear or nonlinear relationship between independent variable x and the dependent variable y.



1. Dataset Dataset (blue points [a]) created by drawing sample from Sinusoidal curve (adding some noise)

Hypothesis Evaluation

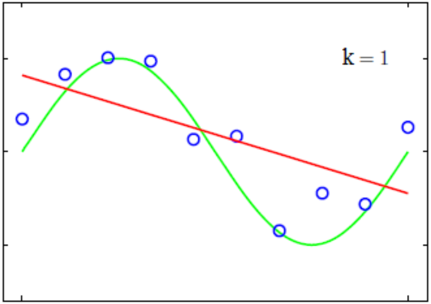
Consider example of "Regression", where we are trying to fits a linear or nonlinear relationship between independent variable x and the dependent variable y.



1. Dataset Dataset (blue points [a]) created by drawing sample from Sinusoidal curve (adding some noise)
2. $k = 0$ Constant (Constant line (average of output values), not a good fit. Under-fitting)

Hypothesis Evaluation

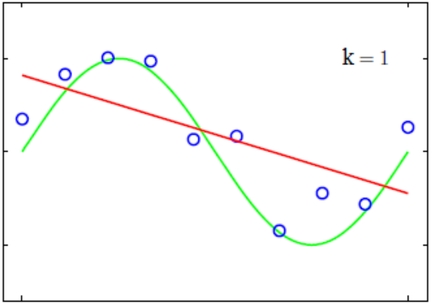
Consider example of "Regression", where we are trying to fits a linear or nonlinear relationship between independent variable x and the dependent variable y.



1. Dataset   Dataset (blue points [a]) created by drawing sample from Sinusoidal curve (adding some noise)
2. k = 0   Constant (Constant line (average of output values), not a good fit. Under-fitting)
3. k = 1   Straight Line (Linear regression, not a good fit. Under-fitting)

Hypothesis Evaluation

Consider example of "Regression", where we are trying to fits a linear or nonlinear relationship between independent variable x and the dependent variable y.



1. Dataset Dataset (blue points [a]) created by drawing sample from Sinusoidal curve (adding some noise)
2. $k = 0$ Constant (Constant line (average of output values), not a good fit. Under-fitting)
3. $k = 1$ Straight Line (Linear regression, not a good fit. Under-fitting)

[a] Images from Bishop's book

Under-fitting

Linear regression is under-fitting the data (high-bias).

Polynomial Regression

- To overcome under-fitting, we need to increase the complexity of the model.

[1] Beware, its different from multi-variate regression. Its not dimension of feature vector.

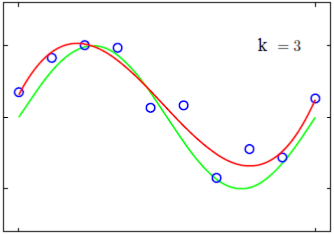
Polynomial Regression

- To overcome under-fitting, we need to increase the complexity of the model.
- To generate a higher order equation, can add powers of the original features as new features. The linear model $h = \theta_0 + \theta_1 x$
can be transformed to $h = \theta_0 + \theta_1 x + \theta_2 x^2 (x - \text{squared})$ *[1]

[1] Beware, its different from multi-variate regression. Its not dimension of feature vector.

Polynomial Regression

- To overcome under-fitting, we need to increase the complexity of the model.
- To generate a higher order equation, can add powers of the original features as new features. The linear model $h = \theta_0 + \theta_1 x$ can be transformed to $h = \theta_0 + \theta_1 x + \theta_2 x^2 (x - \text{squared})$ *[1]

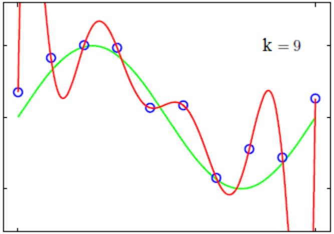


1. k = 2 Parabola
2. k = 3 Cubic (3^{rd} degree Polynomial function, fits nicely)

[1] Beware, its different from multi-variate regression. Its not dimension of feature vector.

Polynomial Regression

- To overcome under-fitting, we need to increase the complexity of the model.
- To generate a higher order equation, can add powers of the original features as new features. The linear model $h = \theta_0 + \theta_1 x$
can be transformed to $h = \theta_0 + \theta_1 x + \theta_2 x^2 (x - \text{squared})$ *[1]

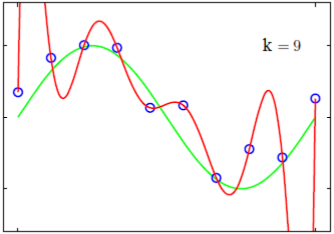


1. k = 2 Parabola
2. k = 3 Cubic (3^{rd} degree Polynomial function, fits nicely)
3. k = 9 9^{th} degree polynomial

[1] Beware, its different from multi-variate regression. Its not dimension of feature vector.

Polynomial Regression

- To overcome under-fitting, we need to increase the complexity of the model.
- To generate a higher order equation, can add powers of the original features as new features. The linear model $h = \theta_0 + \theta_1 x$
can be transformed to $h = \theta_0 + \theta_1 x + \theta_2 x^2 (x - \text{squared})$ *[1]



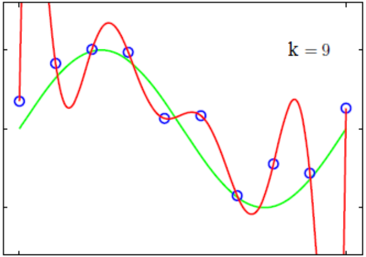
1. k = 2 Parabola
2. k = 3 Cubic (3^{rd} degree Polynomial function, fits nicely)
3. k = 9 9^{th} degree polynomial

General form for Polynomial Regression:

$$h(\theta) = \theta_0 + \theta_1 x + \theta_2 \underbrace{x^2}_{x \cdot x} + \theta_3 \underbrace{x^3}_{x \cdot x \cdot x} + \cdots + \theta_k x^k \quad (1)$$

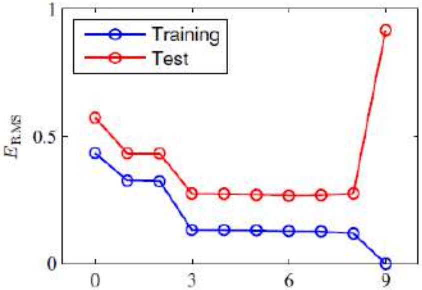
[1] Beware, its different from multi-variate regression. Its not dimension of feature vector.

Polynomial Regression



- 9^{th} degree polynomial fitted curve (shown in red) goes to all the datapoints but otherwise its off by large margin in between points. Ideally fitted curve shape should look like curve shown in green.

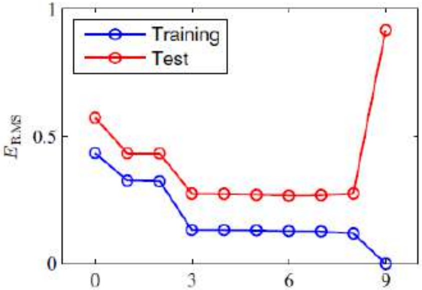
Polynomial Regression



- 9^{th} degree polynomial fitted curve (shown in red) goes to all the datapoints but otherwise its off by large margin in between points. Ideally fitted curve shape should look like curve shown in green.

- It's not surprise to see test error increase exponentially for 9^{th} degree polynomial curve. It shows that despite very low train error, it's generalization capability is very low. It's a perfect example of over-fitting.

Polynomial Regression



- 9^{th} degree polynomial fitted curve (shown in red) goes to all the datapoints but otherwise its off by large margin in between points. Ideally fitted curve shape should look like curve shown in green.

- It's not surprise to see test error increase exponentially for 9^{th} degree polynomial curve. It shows that despite very low train error, it's generalization capability is very low. It's a perfect example of over-fitting.

Over-fitting

9^{th} degree polynomial fitted curve is over-fitting the data (high-variance).

Section Contents

Generalization Error

- Is it enough to know / understand how different machine learning algorithms work?
- How can we gain insight on whether selected model / hypothesis will perform adequately on unseen data i.e. generalization capabilities?
- What if model training error was within predefined bounds, but it makes unacceptably large error on unseen data?
- In this module we will analyze generalization error and decompose it to understand where it comes from.
- Understanding generalization error will give insight to select robust algorithm for a given problem.

Bias-Variance tradeoff

One of the most important topic for machine learning experts.

Formalizing Generalization Error

Setup

- Given dataset $D = \{(\vec{\mathbf{x}}_1, y_1), \ldots, (\vec{\mathbf{x}}_n, y_n)\}$
- D drawn from some distribution $P(X, Y)$ in i.i.d (independent and identically distributed) manner , same supposition for all machine learning algorithms.
- Assume regression setting $y_i \in \mathbb{R}$ (regression setting is easier for derivation)
- Given input x there might not exist a unique label y i.e. if $\vec{\mathbf{x}}$ describes features of a house (e.g. no. of bedrooms, square footage, $\cdots$) and the label y its price, imagine two houses with identical description selling for different prices.

Expected Label : given $\vec{\mathbf{x}} \in \mathbb{R}^d$

Given $\vec{\mathbf{x}} \in \mathbb{R}^d$:

Expected Label : given $\vec{\mathbf{x}} \in \mathbb{R}^d$

Given $\vec{\mathbf{x}} \in \mathbb{R}^d$:

$$\bar{y}(\mathbf{x}) = E_{y|\mathbf{x}}[Y] = \int_y y \Pr(y|\mathbf{x}) \partial y \quad (2)$$

- The **expected label** denotes the label expected to obtain, given a feature vector $\vec{\mathbf{x}}$

Expected Label : given $\vec{\mathbf{x}} \in \mathbb{R}^d$

Given $\vec{\mathbf{x}} \in \mathbb{R}^d$:

$$\bar{y}(\mathbf{x}) = E_{y|\mathbf{x}}[Y] = \int_y y \Pr(y|\mathbf{x})\partial y \tag{2}$$

- The **expected label** denotes the label expected to obtain, given a feature vector $\vec{\mathbf{x}}$

- $\Pr(y|\mathbf{x})$: Probability of y given $\vec{\mathbf{x}}$
- There could be same feature vector $\vec{\mathbf{x}}$ (attributes of a house) but different respective label y (price of a house).

Expected Label : given $\vec{\mathbf{x}} \in \mathbb{R}^d$

Given $\vec{\mathbf{x}} \in \mathbb{R}^d$:

$$\bar{y}(\mathbf{x}) = E_{y|\mathbf{x}}[Y] = \int_y y \Pr(y|\mathbf{x}) \partial y \tag{2}$$

- The **expected label** denotes the label expected to obtain, given a feature vector $\vec{\mathbf{x}}$

- $\Pr(y|\mathbf{x})$: Probability of y given $\vec{\mathbf{x}}$
- There could be same feature vector $\vec{\mathbf{x}}$ (attributes of a house) but different respective label y (price of a house).

- Equation 2 : expected label is average value of infinite many drawn samples (houses) or integrate over all possible y and weight by probability of observing that y given x ($\Pr(y|\mathbf{x})$).

Hypothesis on dataset D

- We draw our training set D, consisting of n inputs, i.i.d. from the distribution P.

- Then call some machine learning algorithm $\mathcal{A}$ on this dataset to learn a hypothesis (aka classifier).

- Formally, we denote this process as:

$$h_D = \mathcal{A}(D) \tag{3}$$

Where $\mathcal{A}$ is machine learning algorithm i.e. Perceptron, DT or SVM etc., D training dataset and h_D is learned hypothesis (a function that takes input $\vec{\mathbf{x}}$ and outputs y).

Expected Test Error : given h_D

- For a given h_D (learned / specific classifier), learned on data set D with algorithm $\mathcal{A}$, we need to compute expected generalization error (error on unseen data points).
- Using sum of squared errors (generally used in regression setting)

$$E_{(\mathbf{x},y)\sim P}\left[\left(h_D(\mathbf{x})-y\right)^2\right] \tag{4}$$

where $E_{(\mathbf{x},y)\sim P}$ is theoretical test error calculated using test point $(\mathbf{x},y)$ drawn from distribution P

Expected Test Error : given h_D

- For a given h_D (learned / specific classifier), learned on data set D with algorithm $\mathcal{A}$, we need to compute expected generalization error (error on unseen data points).
- Using sum of squared errors (generally used in regression setting)

$$E_{(\mathbf{x},y)\sim P}\left[(h_D(\mathbf{x})-y)^2\right] \tag{4}$$

where $E_{(\mathbf{x},y)\sim P}$ is theoretical test error calculated using test point $(\mathbf{x},y)$ drawn from distribution P

$$E_{(\mathbf{x},y)\sim P}\left[(h_D(\mathbf{x})-y)^2\right] = \int_x\int_y (h_D(\mathbf{x})-y)^2 \Pr(\mathbf{x},y)\partial y \partial \mathbf{x} \tag{5}$$

This is what we would like to understand and analyze

Expected Classifier : given $\mathcal{A}$

- Equation 5 is true for a given training set D. However, remember that D itself is drawn from P^n (n samples drawn from P), and is therefore a random variable. Further, h_D is a function of D, and is therefore also a random variable.

- Draw different distribution of D, then you will get slightly different h.

- **Expected Classifier** (given $\mathcal{A}$):

$$\bar{h} = E_{D\sim P^n}\left[\mathcal{A}(D)\right] = \int_D h_D \Pr(D)\partial D \tag{6}$$

Expected Classifier : given $\mathcal{A}$

- Equation 5 is true for a given training set D. However, remember that D itself is drawn from P^n (n samples drawn from P), and is therefore a random variable. Further, h_D is a function of D, and is therefore also a random variable.

- Draw different distribution of D, then you will get slightly different h.

- **Expected Classifier** (given $\mathcal{A}$):

$$\bar{h} = E_{D \sim P^n}[\mathcal{A}(D)] = \int_D h_D \Pr(D) \partial D \tag{6}$$

How to estimate $\bar{h}$?

Make different D (many (infinite many) training sets) by drawing P^n every time and calculate h_D, then average all of them to get $\bar{h}$ (weak law of large numbers).

Expected Test Error : given $\mathcal{A}$

- Earlier we computed expected test error of h (refer Equation 5) or specifically h_D. This is not generalizing well as it only gives expected error for one particular output, but we need to compute how well algorithm do generally.

$$E_{\substack{(\mathbf{x},y)\sim P \\ D\sim P^n}} \left[(h_D(\mathbf{x}) - y)^2 \right] \tag{7}$$

- This is same as Equation 4 but now we integrate over all possible dataset as well.

Explanation

- First draw dataset D from P, then train algorithm to get h_D, then take test point $(\mathbf{x}, y)$ drawn from distribution P and compute the error.
- Do it many times (thousands of time, thousand different h_D and test points $(\mathbf{x}, y)$) to calculate average generalization error of an algorithm $\mathcal{A}$.

Expected Test Error : given $\mathcal{A}$

- Earlier we computed expected test error of h (refer Equation 5) or specifically h_D. This is not generalizing well as it only gives expected error for one particular output, but we need to compute how well algorithm do generally.

$$E_{\substack{(\mathbf{x},y)\sim P \\ D\sim P^n}}\left[(h_D(\mathbf{x}) - y)^2\right] \tag{7}$$

- This is same as Equation 4 but now we integrate over all possible dataset as well.

Explanation

- First draw dataset D from P, then train algorithm to get h_D, then take test point $(\mathbf{x}, y)$ drawn from distribution P and compute the error.
- Do it many times (thousands of time, thousand different h_D and test points $(\mathbf{x}, y)$) to calculate average generalization error of an algorithm $\mathcal{A}$.

$$E_{\substack{(\mathbf{x},y)\sim P \\ D\sim P^n}}\left[(h_D(\mathbf{x}) - y)^2\right] = \int_D \int_{\mathbf{x}} \int_y (h_D(\mathbf{x}) - y)^2 \Pr(\mathbf{x}, y) \Pr(D) \partial\mathbf{x} \partial y \partial D \tag{8}$$

Expected Test Error : given $\mathcal{A}$

- We are interested in exactly this expression, because it evaluates the quality of a machine learning algorithm $\mathcal{A}$ with respect to a data distribution $P(X,Y)$.

- We will pick the algorithm with lowest such error.

$$E_{\substack{(\mathbf{x},y)\sim P \\ D\sim P^n}}\left[(h_D(\mathbf{x})-y)^2\right] = \int_D \int_{\mathbf{x}} \int_y (h_D(\mathbf{x})-y)^2 \, \mathrm{P}(\mathbf{x},y)\mathrm{P}(D)\partial\mathbf{x}\partial y\partial D$$

- Note: x, y are test point drawn from distribution P, and is independent of dataset D. Although D is also dawn from P.

Expected Test Error : given $\mathcal{A}$

- We are interested in exactly this expression, because it evaluates the quality of a machine learning algorithm $\mathcal{A}$ with respect to a data distribution $P(X,Y)$.

- We will pick the algorithm with lowest such error.

$$E_{\substack{(\mathbf{x},y)\sim P \\ D\sim P^n}}\left[(h_D(\mathbf{x})-y)^2\right] = \int_D \int_{\mathbf{x}} \int_y (h_D(\mathbf{x})-y)^2 \,\mathrm{P}(\mathbf{x},y)\mathrm{P}(D)\partial\mathbf{x}\partial y\partial D$$

- Note: x, y are test point drawn from distribution P, and is independent of dataset D. Although D is also dawn from P.

- Next, we will decompose this expression to see from where error creeps in to the system.

Section Contents

Expected Test Error of Algorithm $\mathcal{A}$

- Decomposing this equation, refer Equation 7:

$$E_{\substack{(\mathbf{x},y)\sim P \\ D\sim P^n}}\left[(h_D(\mathbf{x}) - y)^2\right]$$

- **Trick 1**: Add and subtract $\bar{h}(\mathbf{x})$

Expected Test Error of Algorithm $\mathcal{A}$

- Decomposing this equation, refer Equation 7:

$$E_{\substack{(\mathbf{x},y)\sim P \\ D\sim P^n}}\left[\left(h_D(\mathbf{x}) - y\right)^2\right]$$

- **Trick 1**: Add and subtract $\bar{h}(\mathbf{x})$

$$E_{\mathbf{x},y,D}\left[\left[h_D(\mathbf{x}) - y\right]^2\right] = E_{\mathbf{x},y,D}\left[\left[\left(h_D(\mathbf{x}) - \bar{h}(\mathbf{x})\right) + \left(\bar{h}(\mathbf{x}) - y\right)\right]^2\right] \tag{9}$$

Expected Test Error of Algorithm $\mathcal{A}$

- Decomposing this equation, refer Equation 7:

$$E_{\substack{(\mathbf{x},y)\sim P \\ D\sim P^n}}\left[\left(h_D(\mathbf{x}) - y\right)^2\right]$$

- **Trick 1**: Add and subtract $\bar{h}(\mathbf{x})$

$$E_{\mathbf{x},y,D}\left[\left[h_D(\mathbf{x}) - y\right]^2\right] = E_{\mathbf{x},y,D}\left[\left[\left(h_D(\mathbf{x}) - \bar{h}(\mathbf{x})\right) + \left(\bar{h}(\mathbf{x}) - y\right)\right]^2\right] \tag{9}$$

- Its $(a+b)^2 = a^2 + 2ab + b^2$, expand it to analyze each term:

Expected Test Error of Algorithm $\mathcal{A}$

- Decomposing this equation, refer Equation 7:

$$E_{\substack{(\mathbf{x},y)\sim P \\ D\sim P^n}}\left[\left(h_D(\mathbf{x})-y\right)^2\right]$$

- **Trick 1**: Add and subtract $\bar{h}(\mathbf{x})$

$$E_{\mathbf{x},y,D}\left[\left[h_D(\mathbf{x})-y\right]^2\right] = E_{\mathbf{x},y,D}\left[\left[\left(h_D(\mathbf{x})-\bar{h}(\mathbf{x})\right)+\left(\bar{h}(\mathbf{x})-y\right)\right]^2\right] \tag{9}$$

- Its $(a+b)^2 = a^2+2ab+b^2$, expand it to analyze each term:

$$E_{\mathbf{x},D}\left[(\bar{h}_D(\mathbf{x})-\bar{h}(\mathbf{x}))^2\right]+2\;E_{\mathbf{x},y,D}\left[\left(h_D(\mathbf{x})-\bar{h}(\mathbf{x})\right)\left(\bar{h}(\mathbf{x})-y\right)\right]+E_{\mathbf{x},y}\left[\left(\bar{h}(\mathbf{x})-y\right)^2\right] \tag{10}$$

Expected Test Error of Algorithm $\mathcal{A}$

Refer Equation 10:

$$E_{\mathbf{x},D}\left[(\bar{h}_D(\mathbf{x})-\bar{h}(\mathbf{x}))^2\right]+2\;E_{\mathbf{x},y,D}\left[\left(h_D(\mathbf{x})-\bar{h}(\mathbf{x})\right)\left(\bar{h}(\mathbf{x})-y\right)\right]+E_{\mathbf{x},y}\left[\left(\bar{h}(\mathbf{x})-y\right)^2\right]$$

Expected Test Error of Algorithm $\mathcal{A}$

Refer Equation 10:

$$E_{\mathbf{x},D}\left[(\bar{h}_D(\mathbf{x}) - \bar{h}(\mathbf{x}))^2\right] + 2\, E_{\mathbf{x},y,D}\left[\left(h_D(\mathbf{x}) - \bar{h}(\mathbf{x})\right)\left(\bar{h}(\mathbf{x}) - y\right)\right] + E_{\mathbf{x},y}\left[\left(\bar{h}(\mathbf{x}) - y\right)^2\right]$$

- Middle term $(2ab)$ is zero?

Expected Test Error of Algorithm $\mathcal{A}$

Refer Equation 10:

$$E_{\mathbf{x},D}\left[(\bar{h}_D(\mathbf{x}) - \bar{h}(\mathbf{x}))^2\right] + 2\, E_{\mathbf{x},y,D}\left[\left(h_D(\mathbf{x}) - \bar{h}(\mathbf{x})\right)\left(\bar{h}(\mathbf{x}) - y\right)\right] + E_{\mathbf{x},y}\left[\left(\bar{h}(\mathbf{x}) - y\right)^2\right]$$

- Middle term $(2ab)$ is zero?

$$\begin{aligned} E_{\mathbf{x},y,D}\left[\left(h_D(\mathbf{x}) - \bar{h}(\mathbf{x})\right)\left(\bar{h}(\mathbf{x}) - y\right)\right] \\ &= E_{\mathbf{x},y}\left[E_D\left[h_D(\mathbf{x}) - \bar{h}(\mathbf{x})\right]\left(\bar{h}(\mathbf{x}) - y\right)\right] \\ &= E_{\mathbf{x},y}\left[\left(E_D\left[h_D(\mathbf{x})\right] - \bar{h}(\mathbf{x})\right)\left(\bar{h}(\mathbf{x}) - y\right)\right] \\ &= E_{\mathbf{x},y}\left[\left(\bar{h}(\mathbf{x}) - \bar{h}(\mathbf{x})\right)\left(\bar{h}(\mathbf{x}) - y\right)\right] \\ &= E_{\mathbf{x},y}\left[0\right] \\ &= 0 \end{aligned} \tag{11}$$

Expected Test Error of Algorithm $\mathcal{A}$

Refer Equation 10:

$$E_{\mathbf{x},D}\left[(\bar{h}_D(\mathbf{x}) - \bar{h}(\mathbf{x}))^2\right] + 2\,E_{\mathbf{x},y,D}\left[\left(h_D(\mathbf{x}) - \bar{h}(\mathbf{x})\right)\left(\bar{h}(\mathbf{x}) - y\right)\right] + E_{\mathbf{x},y}\left[\left(\bar{h}(\mathbf{x}) - y\right)^2\right]$$

- Middle term $(2ab)$ is zero?

$$\begin{aligned} E_{\mathbf{x},y,D}\left[\left(h_D(\mathbf{x}) - \bar{h}(\mathbf{x})\right)\left(\bar{h}(\mathbf{x}) - y\right)\right] & \\ &= E_{\mathbf{x},y}\left[E_D\left[h_D(\mathbf{x}) - \bar{h}(\mathbf{x})\right]\left(\bar{h}(\mathbf{x}) - y\right)\right] \\ &= E_{\mathbf{x},y}\left[\left(E_D\left[h_D(\mathbf{x})\right] - \bar{h}(\mathbf{x})\right)\left(\bar{h}(\mathbf{x}) - y\right)\right] \\ &= E_{\mathbf{x},y}\left[\left(\bar{h}(\mathbf{x}) - \bar{h}(\mathbf{x})\right)\left(\bar{h}(\mathbf{x}) - y\right)\right] \\ &= E_{\mathbf{x},y}\left[0\right] \\ &= 0 \end{aligned} \tag{11}$$

- Expected value (over D) of $h_D(\mathbf{x})$ is exactly equal to $\bar{h}(\mathbf{x})$
- This is **trick 2** i.e. making term $2ab = 0$.

Expected Test Error of Algorithm $\mathcal{A}$

- Returning to the earlier expression (refer Equation 10), we're left with just two terms corresponding to a^2 and b^2. We just showed expected test of an algorithm $\mathcal{A}$ consists of these two terms (note: point x and dataset D are independent):

$$E_{\mathbf{x},y,D}\left[\left(h_D(\mathbf{x})-y\right)^2\right] = E_{\mathbf{x},D}\left[\left(h_D(\mathbf{x})-\bar{h}(\mathbf{x})\right)^2\right] + E_{\mathbf{x},y}\left[\left(\bar{h}(\mathbf{x})-y\right)^2\right] \tag{12}$$

Expected Test Error of Algorithm $\mathcal{A}$

- Returning to the earlier expression (refer Equation 10), we're left with just two terms corresponding to a^2 and b^2. We just showed expected test of an algorithm $\mathcal{A}$ consists of these two terms (note: point x and dataset D are independent):

$$E_{\mathbf{x},y,D}\left[\left(h_D(\mathbf{x})-y\right)^2\right] = E_{\mathbf{x},D}\left[\left(h_D(\mathbf{x})-\bar{h}(\mathbf{x})\right)^2\right] + E_{\mathbf{x},y}\left[\left(\bar{h}(\mathbf{x})-y\right)^2\right] \tag{12}$$

- What is this term highlighted in red?

Expected Test Error of Algorithm $\mathcal{A}$

- Returning to the earlier expression (refer Equation 10), we're left with just two terms corresponding to a^2 and b^2. We just showed expected test of an algorithm $\mathcal{A}$ consists of these two terms (note: point x and dataset D are independent):

$$E_{\mathbf{x},y,D}\left[\left(h_D(\mathbf{x}) - y\right)^2\right] = E_{\mathbf{x},D}\left[\left(h_D(\mathbf{x}) - \bar{h}(\mathbf{x})\right)^2\right] + E_{\mathbf{x},y}\left[\left(\bar{h}(\mathbf{x}) - y\right)^2\right] \tag{12}$$

- What is this term highlighted in red?

Variance

This is variance and it measures how far a set of numbers is spread out from their mean value (deviation of random variable from mean). $\bar{h}(\mathbf{x})$ is mean function value and $h_D(\mathbf{x})$ is a one of the classifier and when we squared their difference we get variance. So it's variance of prediction / classifier.

Expected Test Error of Algorithm $\mathcal{A}$

- Refer Equation 12, and analyze / decompose second term (highlighted in red) as first term is variance.

$$E_{\mathbf{x},y,D}\left[\left(h_D(\mathbf{x})-y\right)^2\right]=E_{\mathbf{x},D}\left[\left(h_D(\mathbf{x})-\bar{h}(\mathbf{x})\right)^2\right]+E_{\mathbf{x},y}\left[\left(\bar{h}(\mathbf{x})-y\right)^2\right]$$

Expected Test Error of Algorithm $\mathcal{A}$

- Refer Equation 12, and analyze / decompose second term (highlighted in red) as first term is variance.

$$E_{\mathbf{x},y,D}\left[\left(h_D(\mathbf{x})-y\right)^2\right]=E_{\mathbf{x},D}\left[\left(h_D(\mathbf{x})-\bar{h}(\mathbf{x})\right)^2\right]+E_{\mathbf{x},y}\left[\left(\bar{h}(\mathbf{x})-y\right)^2\right]$$

- Now do the same trick again i.e. add and subtract the mean $\bar{y}$ and make term $2ab = 0$.

Expected Test Error of Algorithm $\mathcal{A}$

- Refer Equation 12, and analyze / decompose second term (highlighted in red) as first term is variance.

$$E_{\mathbf{x},y,D}\left[(h_D(\mathbf{x})-y)^2\right] = E_{\mathbf{x},D}\left[\left(h_D(\mathbf{x})-\bar{h}(\mathbf{x})\right)^2\right] + E_{\mathbf{x},y}\left[\left(\bar{h}(\mathbf{x})-y\right)^2\right]$$

- Now do the same trick again i.e. add and subtract the mean $\bar{y}$ and make term $2ab = 0$.

$$E_{\mathbf{x},y}\left[\left(\bar{h}(\mathbf{x})-y\right)^2\right] = E_{\mathbf{x},y}\left[\left(\bar{h}(\mathbf{x})-\bar{y}(\mathbf{x})\right)+\left(\bar{y}(\mathbf{x})-y\right)^2\right] \tag{13}$$

Expected Test Error of Algorithm $\mathcal{A}$

- Refer Equation 12, and analyze / decompose second term (highlighted in red) as first term is variance.

$$E_{\mathbf{x},y,D}\left[\left(h_D(\mathbf{x}) - y\right)^2\right] = E_{\mathbf{x},D}\left[\left(h_D(\mathbf{x}) - \bar{h}(\mathbf{x})\right)^2\right] + E_{\mathbf{x},y}\left[\left(\bar{h}(\mathbf{x}) - y\right)^2\right]$$

- Now do the same trick again i.e. add and subtract the mean $\bar{y}$ and make term $2ab = 0$.

$$E_{\mathbf{x},y}\left[\left(\bar{h}(\mathbf{x}) - y\right)^2\right] = E_{\mathbf{x},y}\left[\left(\bar{h}(\mathbf{x}) - \bar{y}(\mathbf{x})\right) + \left(\bar{y}(\mathbf{x}) - y\right)^2\right] \tag{13}$$

- Its $(a+b)^2 = a^2 + 2ab + b^2$, expand it to analyze each term:

Expected Test Error of Algorithm $\mathcal{A}$

- Its $(a+b)^2 = a^2 + 2ab + b^2$, expand it to analyze each term:

$$\begin{aligned} E_{\mathbf{x},y}\left[\left(\bar{h}(\mathbf{x}) - y\right)^2\right] &= E_{\mathbf{x},y}\left[\left(\bar{h}(\mathbf{x}) - \bar{y}(\mathbf{x})\right) + \left(\bar{y}(\mathbf{x}) - y\right)^2\right] \\ = E_{\mathbf{x},y}\left[\left(\bar{y}(\mathbf{x}) - y\right)^2\right] + E_{\mathbf{x}}\left[\left(\bar{h}(\mathbf{x}) - \bar{y}(\mathbf{x})\right)^2\right] &+ 2\, E_{\mathbf{x},y}\left[\left(\bar{h}(\mathbf{x}) - \bar{y}(\mathbf{x})\right)\left(\bar{y}(\mathbf{x}) - y\right)\right] \end{aligned} \tag{14}$$

Expected Test Error of Algorithm $\mathcal{A}$

- Its $(a+b)^2 = a^2 + 2ab + b^2$, expand it to analyze each term:

$$E_{\mathbf{x},y}\left[\left(\bar{h}(\mathbf{x}) - y\right)^2\right] = E_{\mathbf{x},y}\left[\left(\bar{h}(\mathbf{x}) - \bar{y}(\mathbf{x})\right) + \left(\bar{y}(\mathbf{x}) - y\right)^2\right]$$
$$= E_{\mathbf{x},y}\left[\left(\bar{y}(\mathbf{x}) - y\right)^2\right] + E_{\mathbf{x}}\left[\left(\bar{h}(\mathbf{x}) - \bar{y}(\mathbf{x})\right)^2\right] + 2\,E_{\mathbf{x},y}\left[\left(\bar{h}(\mathbf{x}) - \bar{y}(\mathbf{x})\right)\left(\bar{y}(\mathbf{x}) - y\right)\right] \tag{14}$$

- Before analyzing terms highlighted in red and green, it is argued that last term corresponding to $2ab = 0$ (i.e. expected value of y is $\bar{y}$, thus making the term 0).

Expected Test Error of Algorithm $\mathcal{A}$

- Its $(a+b)^2 = a^2 + 2ab + b^2$, expand it to analyze each term:

$$\begin{aligned} E_{\mathbf{x},y}\left[\left(\bar{h}(\mathbf{x}) - y\right)^2\right] &= E_{\mathbf{x},y}\left[\left(\bar{h}(\mathbf{x}) - \bar{y}(\mathbf{x})\right) + \left(\bar{y}(\mathbf{x}) - y\right)^2\right] \\ = E_{\mathbf{x},y}\left[\left(\bar{y}(\mathbf{x}) - y\right)^2\right] + E_{\mathbf{x}}\left[\left(\bar{h}(\mathbf{x}) - \bar{y}(\mathbf{x})\right)^2\right] &+ 2\, E_{\mathbf{x},y}\left[\left(\bar{h}(\mathbf{x}) - \bar{y}(\mathbf{x})\right)\left(\bar{y}(\mathbf{x}) - y\right)\right] \end{aligned} \tag{14}$$

- Before analyzing terms highlighted in red and green, it is argued that last term corresponding to $2ab = 0$ (i.e. expected value of y is $\bar{y}$, thus making the term 0).

Noise

There is a data point with label y but expected label is $\bar{y}$, it means there is a data point with different label than expected label. Classifier can't do better than $\bar{y}(\mathbf{x})$. So it's a noise. For example, same feature vector but with different labels (noisy data). In regression, same house attributes but one cost 10k$ and the other 1M$.

Expected Test Error of Algorithm $\mathcal{A}$

- Its $(a+b)^2 = a^2 + 2ab + b^2$, expand it to analyze each term:

$$E_{\mathbf{x},y}\left[\left(\bar{h}(\mathbf{x}) - y\right)^2\right] = E_{\mathbf{x},y}\left[\left(\bar{h}(\mathbf{x}) - \bar{y}(\mathbf{x})\right) + \left(\bar{y}(\mathbf{x}) - y\right)^2\right]$$
$$= E_{\mathbf{x},y}\left[\left(\bar{y}(\mathbf{x}) - y\right)^2\right] + E_{\mathbf{x}}\left[\left(\bar{h}(\mathbf{x}) - \bar{y}(\mathbf{x})\right)^2\right] + 2\,E_{\mathbf{x},y}\left[\left(\bar{h}(\mathbf{x}) - \bar{y}(\mathbf{x})\right)\left(\bar{y}(\mathbf{x}) - y\right)\right] \quad (14)$$

- Before analyzing terms highlighted in red and green, it is argued that last term corresponding to $2ab = 0$ (i.e. expected value of y is $\bar{y}$, thus making the term 0).

$Bias^2$

How much error will I get from average classifier (trained from infinite many datasets) and expected label. Here noise is not a issue. This term captures how biased is classifier towards some explanation which is not present in the data. For example data is non linear but I am fitting a line to it. No matter how big is the data, classifier will always make mistakes due to the reason that classifier is biased towards some specific solution. More data will not help. It is bias of the model.

Expected Test Error of Algorithm $\mathcal{A}$

- Plugging back values from Equation 14 to Equation 12 , we get:

$$\underbrace{E_{\mathbf{x},y,D}\left[\left(h_D(\mathbf{x})-y\right)^2\right]}_{\text{Expected Test Error}} = \underbrace{E_{\mathbf{x},D}\left[\left(h_D(\mathbf{x})-\bar{h}(\mathbf{x})\right)^2\right]}_{\text{Variance}} + \underbrace{E_{\mathbf{x},y}\left[\left(\bar{y}(\mathbf{x})-y\right)^2\right]}_{\text{Noise}} + \underbrace{E_{\mathbf{x}}\left[\left(\bar{h}(\mathbf{x})-\bar{y}(\mathbf{x})\right)^2\right]}_{\text{Bias}^2} \quad (15)$$

- In Summary [2] :

[2] Credits: Prof. Kilian, Cornell, USA

Expected Test Error of Algorithm $\mathcal{A}$

- Plugging back values from Equation 14 to Equation 12 , we get:

$$\underbrace{E_{\mathbf{x},y,D}\left[\left(h_D(\mathbf{x})-y\right)^2\right]}_{\text{Expected Test Error}} = \underbrace{E_{\mathbf{x},D}\left[\left(h_D(\mathbf{x})-\bar{h}(\mathbf{x})\right)^2\right]}_{\text{Variance}} + \underbrace{E_{\mathbf{x},y}\left[\left(\bar{y}(\mathbf{x})-y\right)^2\right]}_{\text{Noise}} + \underbrace{E_{\mathbf{x}}\left[\left(\bar{h}(\mathbf{x})-\bar{y}(\mathbf{x})\right)^2\right]}_{\text{Bias}^2} \tag{15}$$

- In Summary [2] :

Variance

Captures how much classifier changes if trained on a different training set. How "over-specialized" is classifier to a particular training set (overfitting)?

[2] Credits: Prof. Kilian, Cornell, USA

Expected Test Error of Algorithm $\mathcal{A}$

- Plugging back values from Equation 14 to Equation 12 , we get:

$$\underbrace{E_{\mathbf{x},y,D}\left[\left(h_D(\mathbf{x}) - y\right)^2\right]}_{\text{Expected Test Error}} = \underbrace{E_{\mathbf{x},D}\left[\left(h_D(\mathbf{x}) - \bar{h}(\mathbf{x})\right)^2\right]}_{\text{Variance}} + \underbrace{E_{\mathbf{x},y}\left[\left(\bar{y}(\mathbf{x}) - y\right)^2\right]}_{\text{Noise}} + \underbrace{E_{\mathbf{x}}\left[\left(\bar{h}(\mathbf{x}) - \bar{y}(\mathbf{x})\right)^2\right]}_{\text{Bias}^2} \tag{15}$$

- In Summary [2] :

Bias

What is the inherent error of the classifier is, even with infinite training data? This is due to your classifier being "biased" to a particular kind of solution (e.g. linear classifier). In other words, bias is inherent to model.

[2] Credits: Prof. Kilian, Cornell, USA

Expected Test Error of Algorithm $\mathcal{A}$

- Plugging back values from Equation 14 to Equation 12 , we get:

$$\underbrace{E_{\mathbf{x},y,D}\left[\left(h_D(\mathbf{x}) - y\right)^2\right]}_{\text{Expected Test Error}} = \underbrace{E_{\mathbf{x},D}\left[\left(h_D(\mathbf{x}) - \bar{h}(\mathbf{x})\right)^2\right]}_{\text{Variance}} + \underbrace{E_{\mathbf{x},y}\left[\left(\bar{y}(\mathbf{x}) - y\right)^2\right]}_{\text{Noise}} + \underbrace{E_{\mathbf{x}}\left[\left(\bar{h}(\mathbf{x}) - \bar{y}(\mathbf{x})\right)^2\right]}_{\text{Bias}^2} \tag{15}$$

- In Summary [2] :

Noise

How big is the data-intrinsic noise? This error measures ambiguity due to data distribution and feature representation.

[2] Credits: Prof. Kilian, Cornell, USA

Section Contents

Bias, Variance & Noise

Machine learning algorithm's generalization error is usually decomposed in:

$$\underbrace{E_{\mathbf{x},y,D}\left[\left(h_D(\mathbf{x})-y\right)^2\right]}_{\text{Expected Test Error}} = \underbrace{E_{\mathbf{x},D}\left[\left(h_D(\mathbf{x})-\bar{h}(\mathbf{x})\right)^2\right]}_{\text{Variance}} + \underbrace{E_{\mathbf{x},y}\left[\left(\bar{y}(\mathbf{x})-y\right)^2\right]}_{\text{Noise}} + \underbrace{E_{\mathbf{x}}\left[\left(\bar{h}(\mathbf{x})-\bar{y}(\mathbf{x})\right)^2\right]}_{\text{Bias}^2} \tag{16}$$

Bias, Variance & Noise

Machine learning algorithm's generalization error is usually decomposed in:

$$\underbrace{E_{\mathbf{x},y,D}\left[\left(h_D(\mathbf{x})-y\right)^2\right]}_{\text{Expected Test Error}} = \underbrace{E_{\mathbf{x},D}\left[\left(h_D(\mathbf{x})-\bar{h}(\mathbf{x})\right)^2\right]}_{\text{Variance}} + \underbrace{E_{\mathbf{x},y}\left[\left(\bar{y}(\mathbf{x})-y\right)^2\right]}_{\text{Noise}} + \underbrace{E_{\mathbf{x}}\left[\left(\bar{h}(\mathbf{x})-\bar{y}(\mathbf{x})\right)^2\right]}_{\text{Bias}^2} \quad (16)$$

Variance (error from sensitivity to small fluctuations in training data)

Captures how much classifier changes if trained on a different training set. How "over-specialized" is classifier to a particular training set (over-fitting)?

Bias, Variance & Noise

Machine learning algorithm's generalization error is usually decomposed in:

$$\underbrace{E_{\mathbf{x},y,D}\left[\left(h_D(\mathbf{x})-y\right)^2\right]}_{\text{Expected Test Error}} = \underbrace{E_{\mathbf{x},D}\left[\left(h_D(\mathbf{x})-\bar{h}(\mathbf{x})\right)^2\right]}_{\text{Variance}} + \underbrace{E_{\mathbf{x},y}\left[\left(\bar{y}(\mathbf{x})-y\right)^2\right]}_{\text{Noise}} + \underbrace{E_{\mathbf{x}}\left[\left(\bar{h}(\mathbf{x})-\bar{y}(\mathbf{x})\right)^2\right]}_{\text{Bias}^2} \quad (16)$$

Bias (error from wrong model assumptions)

What is the inherent error of the classifier is, even with infinite training data? This is due to your classifier being "biased" to a particular kind of solution (e.g. linear classifier). In other words, bias is inherent to model and relates to (under-fitting)

Bias, Variance & Noise

Machine learning algorithm's generalization error is usually decomposed in:

$$\underbrace{E_{\mathbf{x},y,D}\left[\left(h_D(\mathbf{x}) - y\right)^2\right]}_{\text{Expected Test Error}} = \underbrace{E_{\mathbf{x},D}\left[\left(h_D(\mathbf{x}) - \bar{h}(\mathbf{x})\right)^2\right]}_{\text{Variance}} + \underbrace{E_{\mathbf{x},y}\left[\left(\bar{y}(\mathbf{x}) - y\right)^2\right]}_{\text{Noise}} + \underbrace{E_{\mathbf{x}}\left[\left(\bar{h}(\mathbf{x}) - \bar{y}(\mathbf{x})\right)^2\right]}_{\text{Bias}^2} \tag{16}$$

Noise

How big is the data-intrinsic noise? This error measures ambiguity due to data distribution and feature representation.

Bias, Variance & Noise

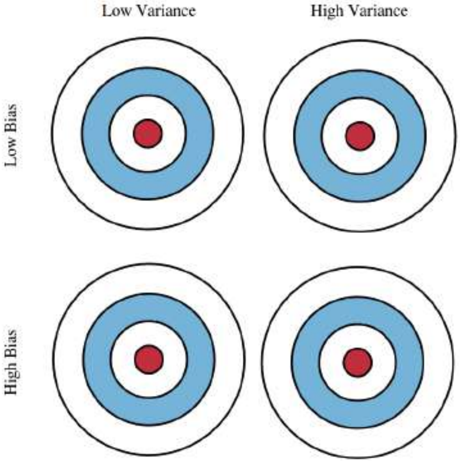
Machine learning algorithm's generalization error is usually decomposed in:

$$\underbrace{E_{\mathbf{x},y,D}\left[\left(h_D(\mathbf{x}) - y\right)^2\right]}_{\text{Expected Test Error}} = \underbrace{E_{\mathbf{x},D}\left[\left(h_D(\mathbf{x}) - \bar{h}(\mathbf{x})\right)^2\right]}_{\text{Variance}} + \underbrace{E_{\mathbf{x},y}\left[\left(\bar{y}(\mathbf{x}) - y\right)^2\right]}_{\text{Noise}} + \underbrace{E_{\mathbf{x}}\left[\left(\bar{h}(\mathbf{x}) - \bar{y}(\mathbf{x})\right)^2\right]}_{\text{Bias}^2} \quad (16)$$

Insight

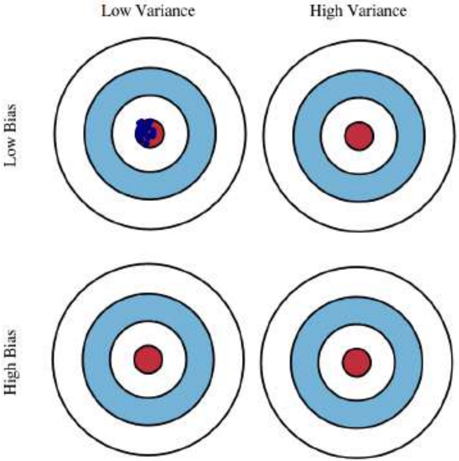
By knowing whether its a bias or variance error or both, will significantly help in improving performance of ML algorithm. The beauty is that terms contributing in the error are quadratic (power of 2) and most probably one term dominants the others. So it is possible to reduce that dominating term without exploding other terms.

Bias- Variance : Graphical Representation



[3] Source: http://scott.fortmann-roe.com/docs/BiasVariance.html

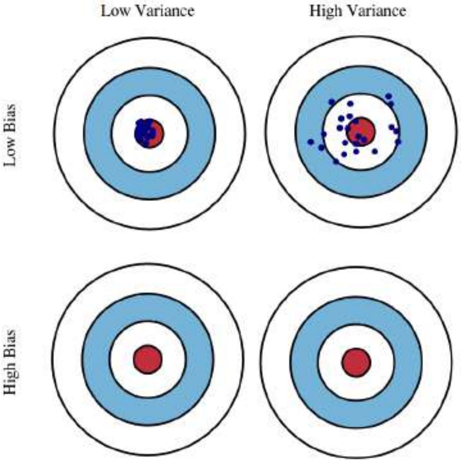
Bias- Variance : Graphical Representation



3

[3] Source: http://scott.fortmann-roe.com/docs/BiasVariance.html

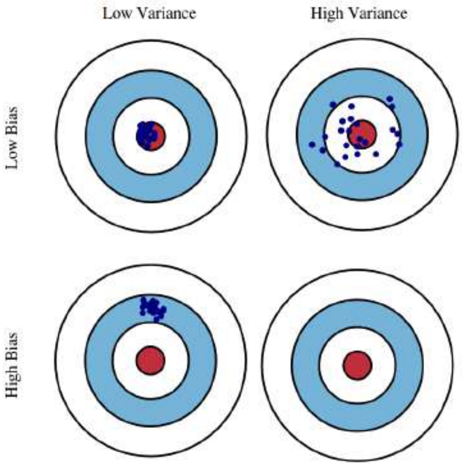
Bias- Variance : Graphical Representation



3

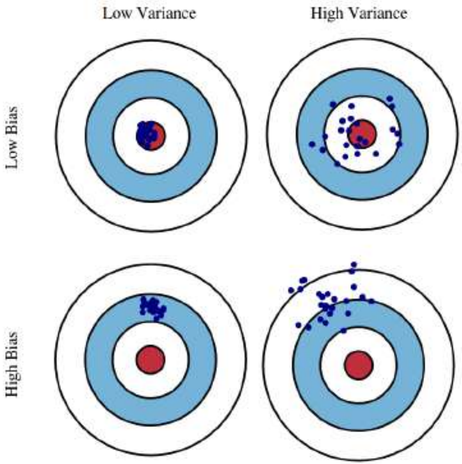
[3]Source: http://scott.fortmann-roe.com/docs/BiasVariance.html

Bias- Variance : Graphical Representation



[3] Source: http://scott.fortmann-roe.com/docs/BiasVariance.html

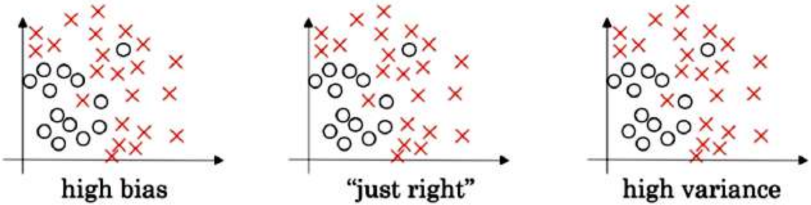
Bias- Variance : Graphical Representation



3

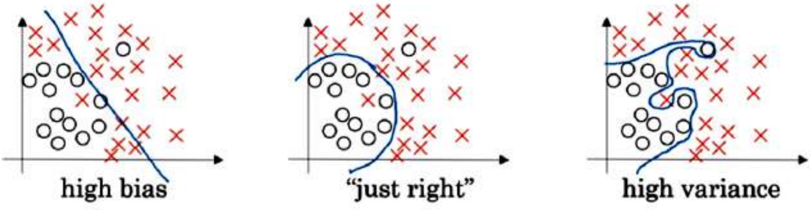
[3] Source: http://scott.fortmann-roe.com/docs/BiasVariance.html

Bias-Variance : Graphical Representation



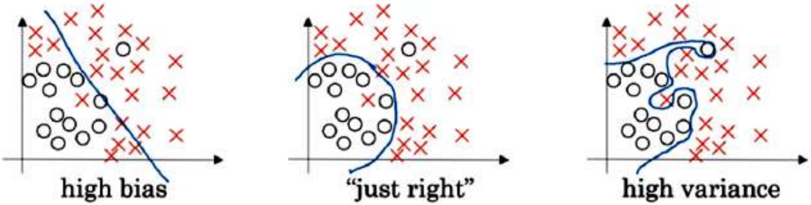
[4]slide from Andrew Ng

Bias-Variance : Graphical Representation



[4] slide from Andrew Ng

Bias-Variance : Graphical Representation

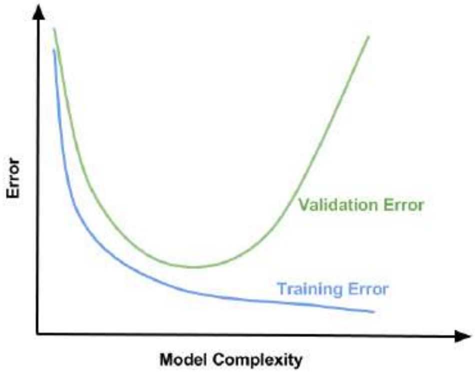


How high bias and high variance looks like?

Note[4]

[4] slide from Andrew Ng

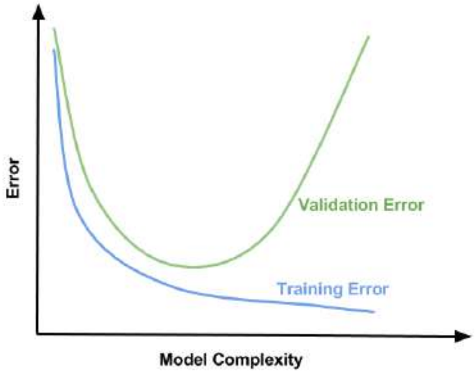
Error type detection



- As model gets complex, training error reduces (it corresponds to overfitting).
- When model is simple, it corresponds to underfitting.

Demo available

Error type detection



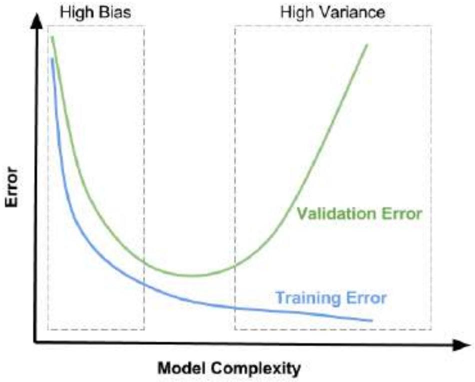
Demo available

- As model gets complex, training error reduces (it corresponds to overfitting).
- When model is simple, it corresponds to underfitting.

Generalization error is high

Is it a bias problem or a variance problem?

Error type detection



Demo available

- As model gets complex, training error reduces (it corresponds to overfitting).
- When model is simple, it corresponds to underfitting.

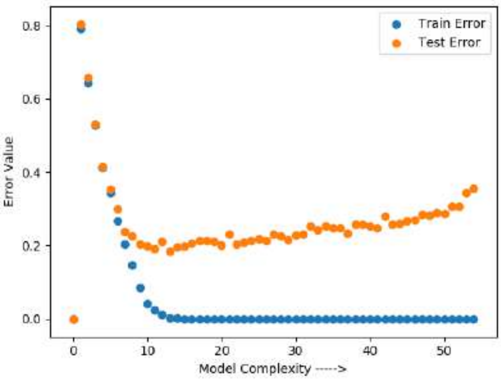
Generalization error is high

Is it a bias problem or a variance problem?

Is it a bias problem or a variance problem

- High bias (under-fit): When training error and validation error, both are high.
- High variance (over-fit): When training error is low, while validation error is very high.

Error type detection



Demo available

- As model gets complex, training error reduces (it corresponds to overfitting).
- When model is simple, it corresponds to underfitting.

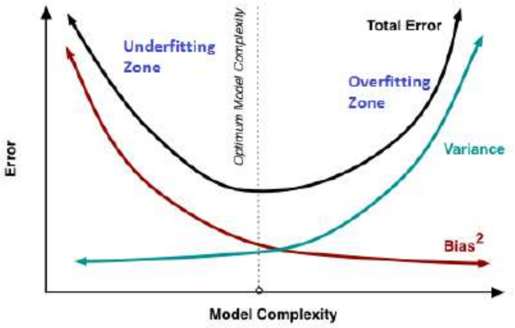
Generalization error is high

Is it a bias problem or a variance problem?

Is it a bias problem or a variance problem

- High bias (under-fit): When training error and validation error, both are high.
- High variance (over-fit): When training error is low, while validation error is very high.

Error type detection : Finding the balance



Bias and variance contributing to total error

- Understanding the illustration:
  1. At its root, dealing with bias and variance is really about dealing with over- and under-fitting.
  2. Bias is reduced and variance is increased in relation to model complexity.
  3. As more and more parameters are added to a model, the complexity of the model rises and variance becomes our primary concern while bias steadily falls.

Model Evaluation

To find that optimum complexity, we can use:

1. Data Partitioning / splitting
2. Early stopping (Stop optimization after M (>= 0) number of gradient steps, even if optimization has not converged yet.)

Model Evaluation : Data Partitioning

- How machine learning trained model generalizes on unseen data is an important aspect. As aim of trained model is to correctly predict new examples. Good training accuracy can be achieved from memorizing trained data.

Model Evaluation : Data Partitioning

- How machine learning trained model generalizes on unseen data is an important aspect. As aim of trained model is to correctly predict new examples. Good training accuracy can be achieved from memorizing trained data.
- The above issue can be handled by evaluating the performance (generalization capability) of a trained model model on unseen data, separated from available dataset. Following are few dataset partitioning techniques:
  - Hold out
  - $k - fold$ Cross validation
  - Bootstrap
  - Leave-one-out cross-validation

Model Evaluation : Data Partitioning

- How machine learning trained model generalizes on unseen data is an important aspect. As aim of trained model is to correctly predict new examples. Good training accuracy can be achieved from memorizing trained data.
- The above issue can be handled by evaluating the performance (generalization capability) of a trained model model on unseen data, separated from available dataset. Following are few dataset partitioning techniques:
  - Hold out
  - $k - fold$ Cross validation
  - Bootstrap
  - Leave-one-out cross-validation

- More training data gives better generalization.
- More test data gives better estimate for the classification error probability.
- Never evaluate performance on training data. The conclusion would be biased.

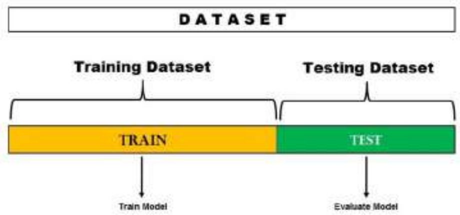
Model Evaluation: Hold out - Data Split

Dataset:

Experience (Yrs)	Salary
1	30k
1.3	33k
1.8	36k
2	45k
3.3	65k
2.2	46k
4	66k
5	70k
6.5	72k
6.2	72k

Hold out cross validation:

- The goal of cross-validation is to define a dataset to test the model, in order to limit problems like overfitting, give an insight on how the model will generalize to an independent dataset.



- Randomly divide data into two: Training and Test set i.e. a hold-out set (30%).

Model Evaluation

- Learn parameter θ from training data (minimizing training error $J(\theta)$).
- Compute test-set error:

For Regression:

$$J(\theta) = \frac{1}{2n} \sum_{i=i}^{n_{test}} (\hat{y}_i - y_i)^2 \tag{17}$$

For Classification:

$$err(i) = \begin{cases} 1, if\ y_i \neq \hat{y}_i \\ 0, otherwsie \end{cases}$$

$$Error_{total} = \frac{1}{n_{test}} \sum_{i=i}^{n_{test}} err(i) \tag{18}$$

Where $\hat{y}_i$ = predicted value on test sample, y_i is actual value and n_{test} is total number of test samples.

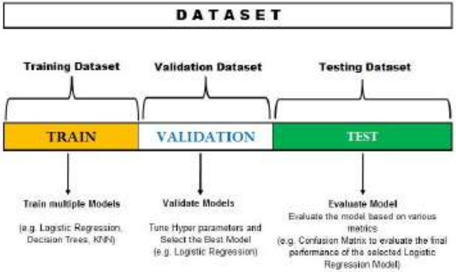
Model Evaluation: Hold out - Data Split

Drawback of test / train split: Error found in the test dataset can highly depend on the observations included in the train and test dataset. Actually, we are fitting learned parameters from train data to test data, by choosing hypothesis that gives best result on test-set. Thus, its an optimistic estimate of generalization error. This method is also not effective for comparing multiple models and tuning hyper-parameters.

Model Evaluation: Hold out - Data Split

Drawback of test / train split: Error found in the test dataset can highly depend on the observations included in the train and test dataset. Actually, we are fitting learned parameters from train data to test data, by choosing hypothesis that gives best result on test-set. Thus, its an optimistic estimate of generalization error. This method is also not effective for comparing multiple models and tuning hyper-parameters.

Improvement: Splitting of dataset into three separate sets i.e. train, (cross) validation & test. Model / hypothesis is selected that has minimum validation error. Estimate of generalization error is then calculated using test-set.



Model Evaluation: Hold out - Data Split

1. Train Data: it is used to initially machine learning model train and make predictions. Model will run on this set of data exhaustively (mostly iteratively). It's the largest part of your overall dataset, comprising around 60-70% of total data used in the project.

Model Evaluation: Hold out - Data Split

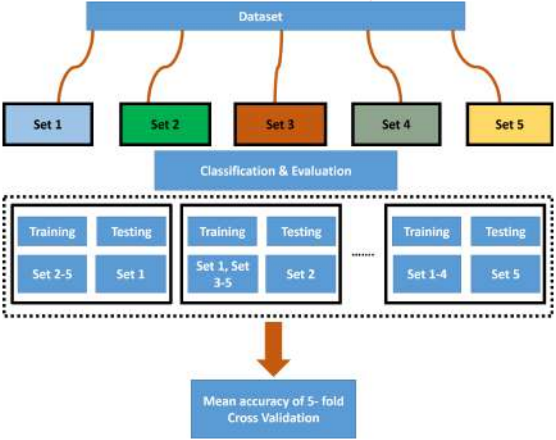
1. Train Data: it is used to initially machine learning model train and make predictions. Model will run on this set of data exhaustively (mostly iteratively). It's the largest part of your overall dataset, comprising around 60-70% of total data used in the project.
2. Validation Data: Trained model uses this data to see whether it can correctly identify relevant new examples. So, used to discover new values that are impacting the process / hyper-parameter tuning. Another common problem often identified during validation is overfitting. Often, after validation, data scientists will often go back to the training data and run through it again, tweaking values and hyper-parameters to make the model more accurate.

Model Evaluation: Hold out - Data Split

1. Train Data: it is used to initially machine learning model train and make predictions. Model will run on this set of data exhaustively (mostly iteratively). It's the largest part of your overall dataset, comprising around 60-70% of total data used in the project.
2. Validation Data: Trained model uses this data to see whether it can correctly identify relevant new examples. So, used to discover new values that are impacting the process / hyper-parameter tuning. Another common problem often identified during validation is overfitting. Often, after validation, data scientists will often go back to the training data and run through it again, tweaking values and hyper-parameters to make the model more accurate.
3. Test Data: comes into play after a lot of improvement and validation. This data is used by the model to make predictions to test whether it will work in the real world.

Model Evaluation: $k - fold$ Cross validation - Data Split

In k-fold cross validation, dataset is divided into k equal subsets. k-1 subsets are used for the training while a single set is retained for testing. The process is repeated k times (k-folds), with each of the k subsets used exactly once for testing. Then, the k estimations (accuracy) from k-folds are averaged to produce final estimated value.



Model Evaluation: Bootstrap - Data Split

- The bootstrap (also called *bagging*[1]) uses sampling with replacement to form the training set.

[1] Proposed in: Breiman, Leo (1996). Bagging predictors. Machine Learning 24 (2): 123–140.

Model Evaluation: Bootstrap - Data Split

- The bootstrap (also called *bagging*[1]) uses sampling with replacement to form the training set.
- Given: the training set T consisting of n entries.

[1] Proposed in: Breiman, Leo (1996). Bagging predictors. Machine Learning 24 (2): 123–140.

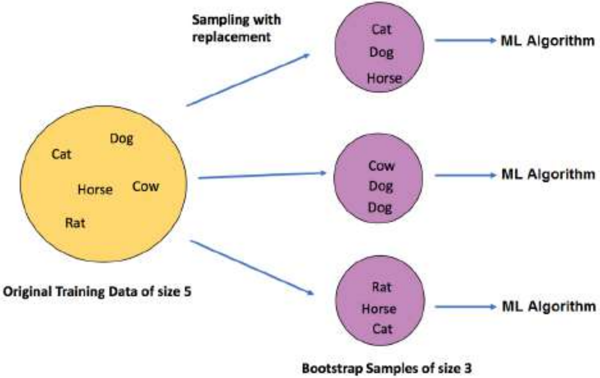
Model Evaluation: Bootstrap - Data Split

- The bootstrap (also called *bagging*[1]) uses sampling with replacement to form the training set.
- Given: the training set T consisting of n entries.
- Bootstrap generates m new datasets T_i each of size $n' < n$ by sampling T uniformly with replacement. The consequence is that some entries can be repeated in T_i.

[1] Proposed in: Breiman, Leo (1996). Bagging predictors. Machine Learning 24 (2): 123–140.

Model Evaluation: Bootstrap - Data Split

- The bootstrap (also called *bagging*[1]) uses sampling with replacement to form the training set.
- Given: the training set T consisting of n entries.
- Bootstrap generates m new datasets T_i each of size $n' < n$ by sampling T uniformly with replacement. The consequence is that some entries can be repeated in T_i.
- The m statistical models (e.g., classifiers, regressors) are learned using the above m bootstrap samples.



[1] Proposed in: Breiman, Leo (1996). Bagging predictors. Machine Learning 24 (2): 123–140.

Model Evaluation: LOOCV - Data Split

- Leave-One-Out Cross-Validation (LOOCV)
  1. Do N experiments. In each experiment, use $N-1$ samples for training, and leave only 1 sample for testing.

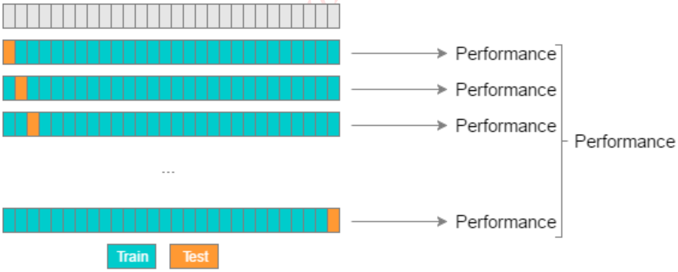
Model Evaluation: LOOCV - Data Split

- Leave-One-Out Cross-Validation (LOOCV)
  1. Do N experiments. In each experiment, use $N-1$ samples for training, and leave only 1 sample for testing.
  2. Compute the testing error $E_i, i = 1, 2, \ldots, N$.

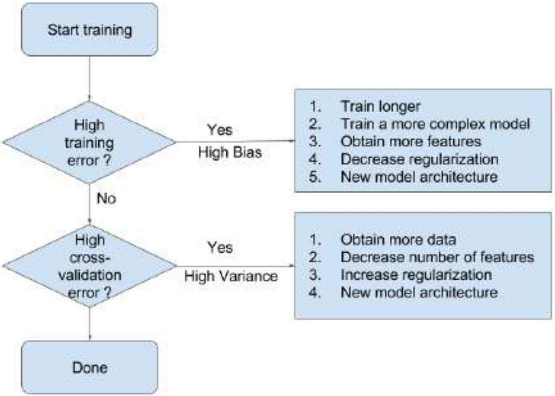
Model Evaluation: LOOCV - Data Split

- Leave-One-Out Cross-Validation (LOOCV)
  1. Do N experiments. In each experiment, use $N-1$ samples for training, and leave only 1 sample for testing.
  2. Compute the testing error $E_i, i = 1, 2, \ldots, N$.
  3. After N experiments, compute the overall estimated error:

$$E = \frac{1}{N}\sum_{i=1}^{N} E_i \tag{19}$$

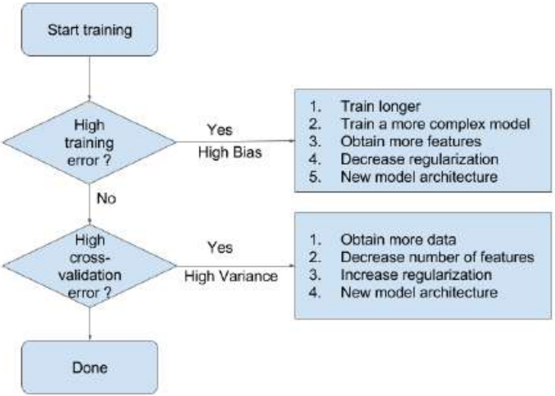


Dealing with Variance and Bias errors



- Keep iterating (image on the left) till low bias and low variance is achieved.
- Bias-Variance Tradeoff : Tool for one can hurt other metric (probably this is not valid for DL). For example training complex model can reduce bias but can increase variance.

Dealing with Variance and Bias errors

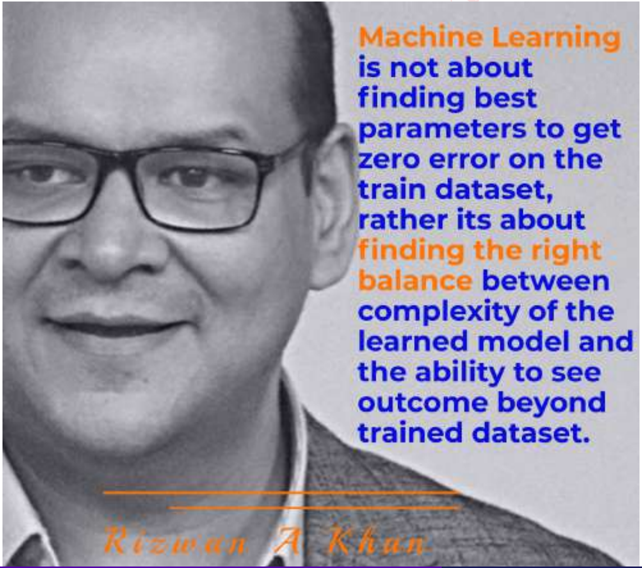


- Keep iterating (image on the left) till low bias and low variance is achieved.
- Bias-Variance Tradeoff : Tool for one can hurt other metric (probably this is not valid for DL). For example training complex model can reduce bias but can increase variance.

Ensemble Learning

Bagging and Boosting are widely used techniques for dealing with high variance and high bias problems respectively.

Dealing with Variance and Bias errors



Section Contents

Tasks

Further Reading

1. Weak law of large numbers.
2. Effect of regularization on bias and variance.

Machine Learning
Unsupervised Learning
Clustering

Dr. Rizwan Ahmed Khan

Outline

Section Contents

Reference Books

Reference books for this Module:

- Chapter 9: Pattern Recognition and Machine Learning, Christopher M. Bishop, Springer Books, latest edition.

Reference Books

Reference books for this Module:

- Chapter 9: Pattern Recognition and Machine Learning, Christopher M. Bishop, Springer Books, latest edition.
- Chapter 14: Principles of Data Mining, Max Bramer, Springer Books.

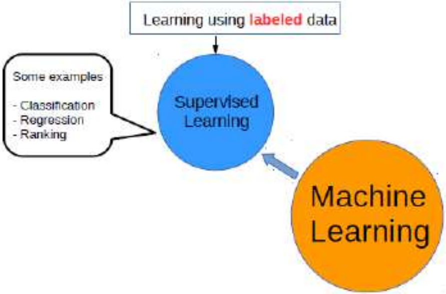
Reference Books

Reference books for this Module:

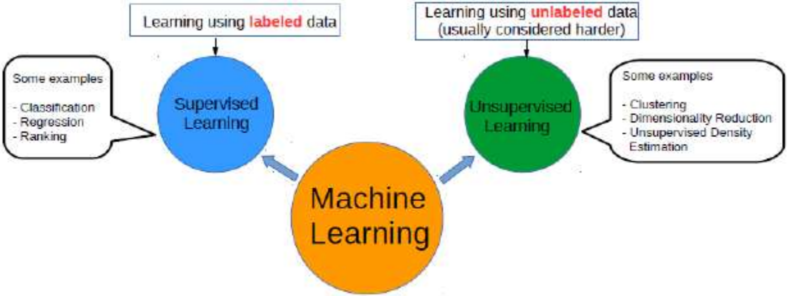
- Chapter 9: Pattern Recognition and Machine Learning, Christopher M. Bishop, Springer Books, latest edition.
- Chapter 14: Principles of Data Mining, Max Bramer, Springer Books.
- Chapter 8: Introduction to Data Mining Kumar et al., 2^{nd} Edition, Pearson Education.

Taxonomy of Machine learning

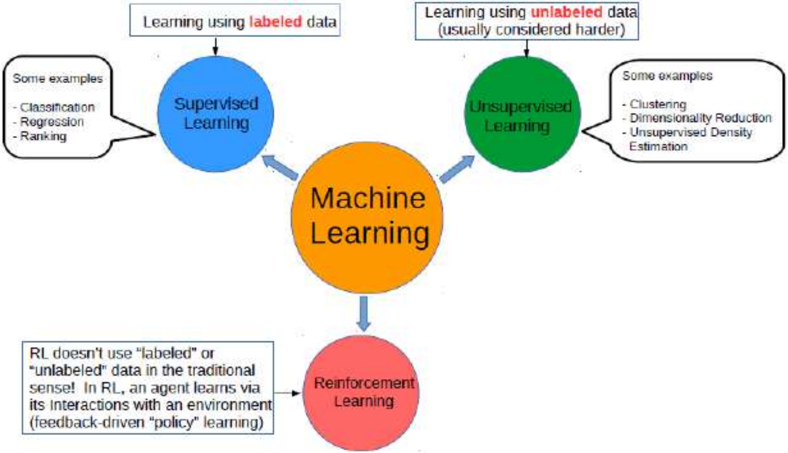
Taxonomy of Machine learning



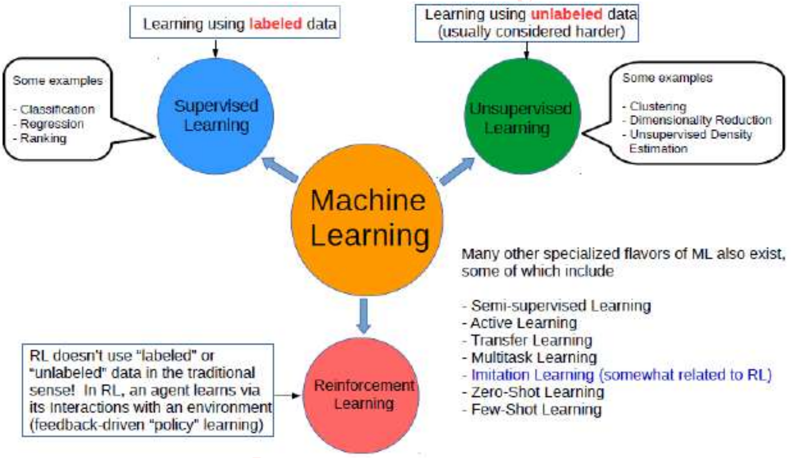
Taxonomy of Machine learning



Taxonomy of Machine learning



Taxonomy of Machine learning



Supervised Learning: Function approximation

Supervised learning is about function approximation

Problem Setting:

- Set of possible instances X
- Unknown target function $f : X \rightarrow Y$
- Set of function hypotheses $H = \{h | h : X \rightarrow Y\}$

Input:

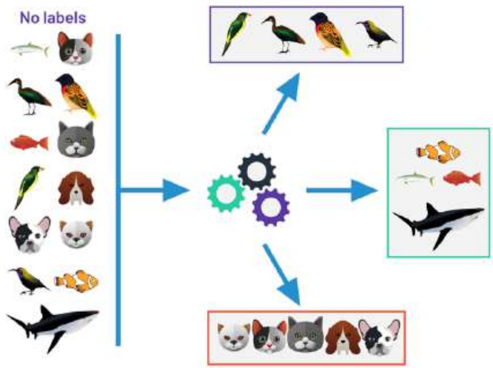
- training examples $\{< x_i, y_i >\}$. For example x is an email and y is either Spam or No Spam.
- We have labeled data in supervised learning.

Output:

- Hypothesis $h \in H$ that best approximates target function f. OR
- a classification "rule" that can determine the class of any object from its attributes values.

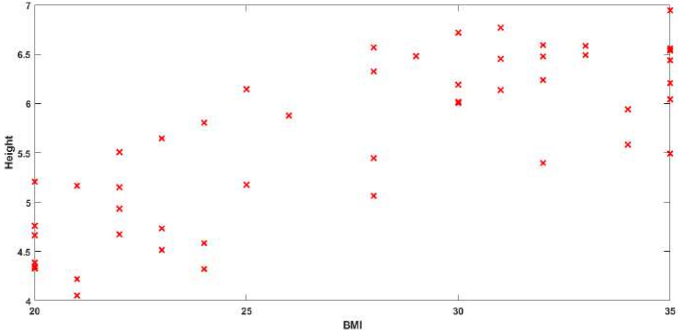
Unsupervised Learning: Deductive Learning

Unsupervised learning is about description, opposed to approximation (supervised learning).



Unsupervised Learning: Deductive Learning

Unsupervised learning is about description, opposed to approximation (supervised learning).



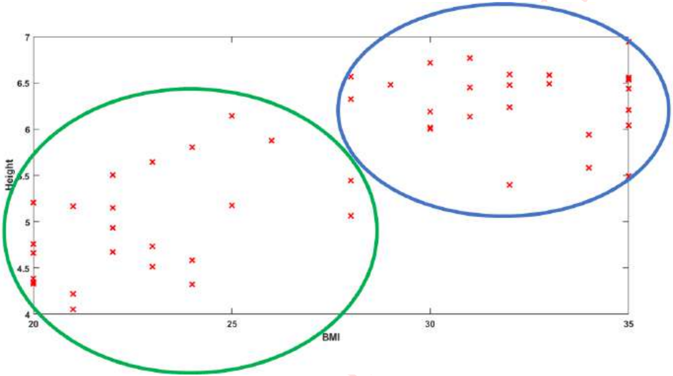
- As data is unlabeled, aim to find structure / pattern in the data.

Input:

- Training examples $\{x_1, x_2, \cdots, x_m\}$.
- Now, data is unlabeled.

Unsupervised Learning: Deductive Learning

Unsupervised learning is about description, opposed to approximation (supervised learning).



- As data is unlabeled, aim to find structure / pattern in the data.

Input:

- Training examples $\{x_1, x_2, \cdots, x_m\}$.
- Now, data is unlabeled.

Unsupervised Learning: Deductive Learning

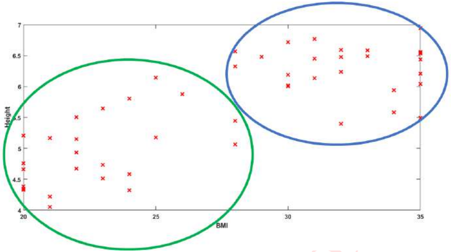
- Unlabeled data / examples

Unsupervised Learning: Deductive Learning

- Unlabeled data / examples
- Derive structure from the data by exploring the relationship b/w input examples

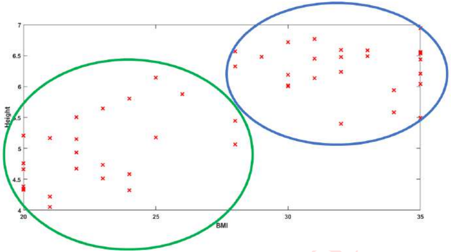
Unsupervised Learning: Deductive Learning

- Unlabeled data / examples
- Derive structure from the data by exploring the relationship b/w input examples



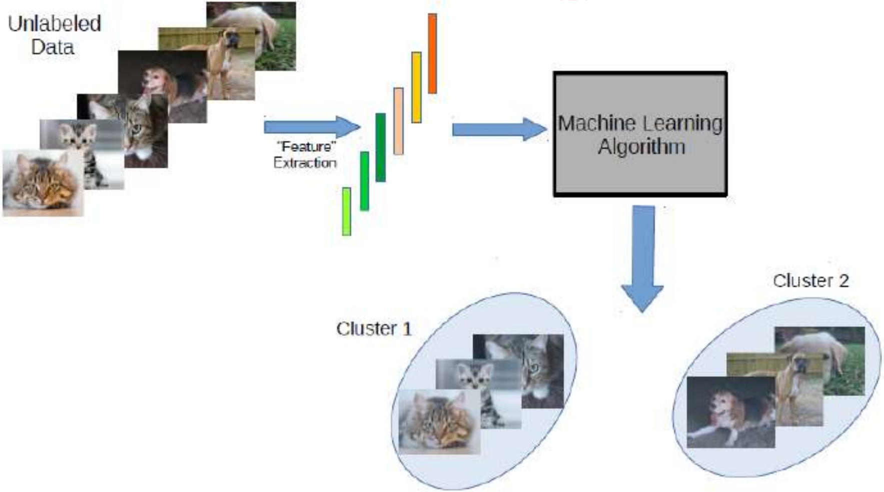
Unsupervised Learning: Deductive Learning

- Unlabeled data / examples
- Derive structure from the data by exploring the relationship b/w input examples

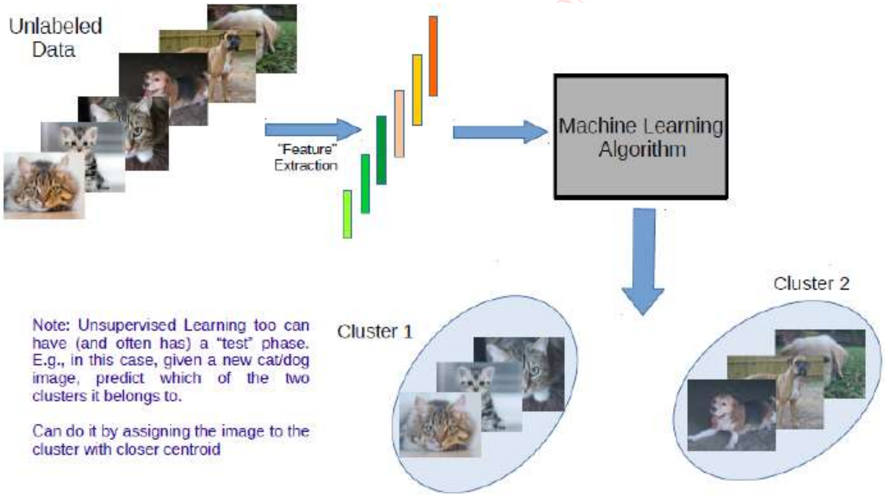


- Unsupervised learning is about description

Unsupervised Learning



Unsupervised Learning



Clustering

- Clustering is one of the most common exploratory data analysis technique.

Clustering

- Clustering is one of the most common exploratory data analysis technique.
- It is used to get an intuition about the structure of the data. And used to:

Clustering

- Clustering is one of the most common exploratory data analysis technique.
- It is used to get an intuition about the structure of the data. And used to:
  - cluster data into meaningful and useful groups i.e. taxonomy of living things.
  - Identify subgroups / clustering in the data. OR
  - finding homogeneous subgroups / clusters (data points in each cluster are as similar as possible) within the data.

Clustering

- Clustering is one of the most common exploratory data analysis technique.
- It is used to get an intuition about the structure of the data. And used to:
  - cluster data into meaningful and useful groups i.e. taxonomy of living things.
  - Identify subgroups / clustering in the data. OR
  - finding homogeneous subgroups / clusters (data points in each cluster are as similar as possible) within the data.
- Types (main) of clustering:
  1. Partitional clustering i.e. K - Means clustering.

Clustering

- Clustering is one of the most common exploratory data analysis technique.
- It is used to get an intuition about the structure of the data. And used to:
  - cluster data into meaningful and useful groups i.e. taxonomy of living things.
  - Identify subgroups / clustering in the data. OR
  - finding homogeneous subgroups / clusters (data points in each cluster are as similar as possible) within the data.
- Types (main) of clustering:
  1. Partitional clustering i.e. K - Means clustering.
  2. Hierarchical clustering i.e. Agglomerative Hierarchical clustering.

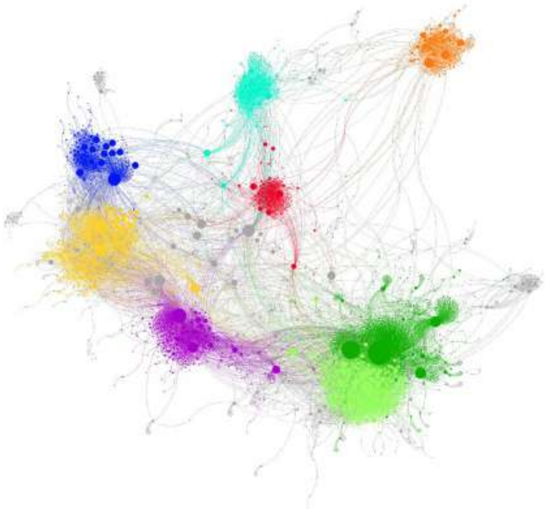
Clustering

- Clustering is one of the most common exploratory data analysis technique.
- It is used to get an intuition about the structure of the data. And used to:
  - cluster data into meaningful and useful groups i.e. taxonomy of living things.
  - Identify subgroups / clustering in the data. OR
  - finding homogeneous subgroups / clusters (data points in each cluster are as similar as possible) within the data.
- Types (main) of clustering:
  1. Partitional clustering i.e. K - Means clustering.
  2. Hierarchical clustering i.e. Agglomerative Hierarchical clustering.
  3. Fuzzy clustering i.e. Fuzzy K-Means.

Clustering Applications

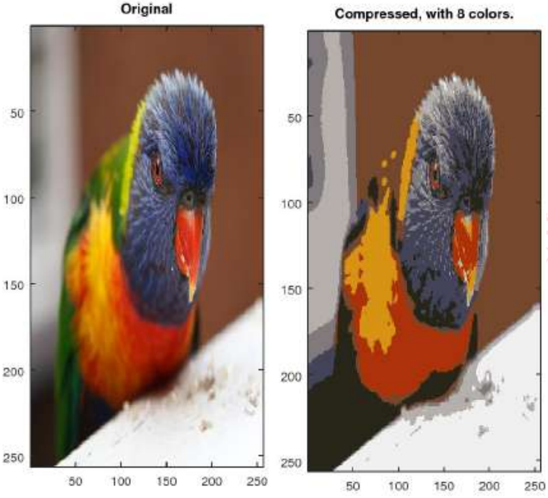
- Market segmentation

Clustering Applications



- Market segmentation
- Social network analysis

Clustering Applications

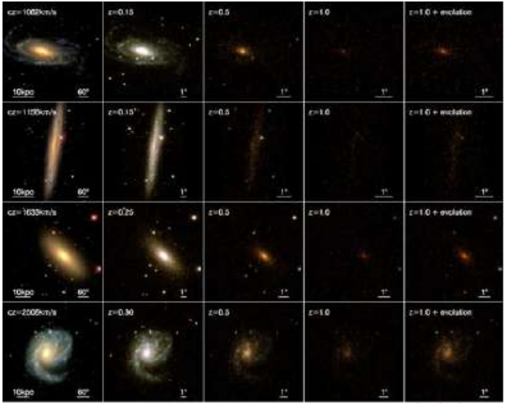


- Market segmentation
- Social network analysis
- Image compression / segmentation

Clustering Applications

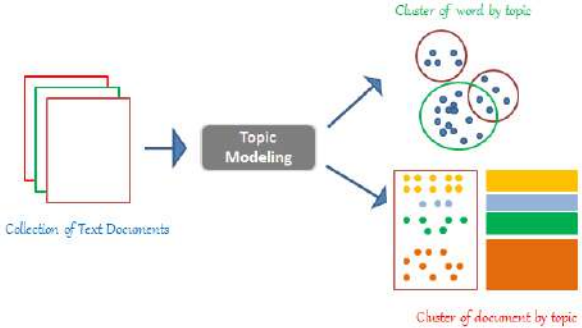
- Market segmentation
- Social network analysis
- Image compression / segmentation
- Organizing computer cluster / data centers

Clustering Applications



- Market segmentation
- Social network analysis
- Image compression / segmentation
- Organizing computer cluster / data centers
- Astronomical data analysis

Clustering Applications



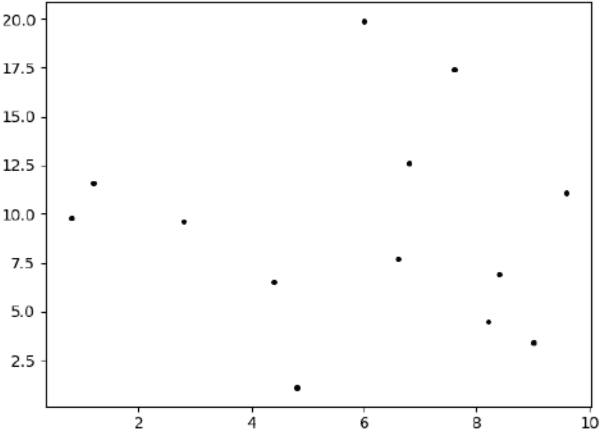
- Market segmentation
- Social network analysis
- Image compression / segmentation
- Organizing computer cluster / data centers
- Astronomical data analysis
- Document clustering

Section Contents

Introduction

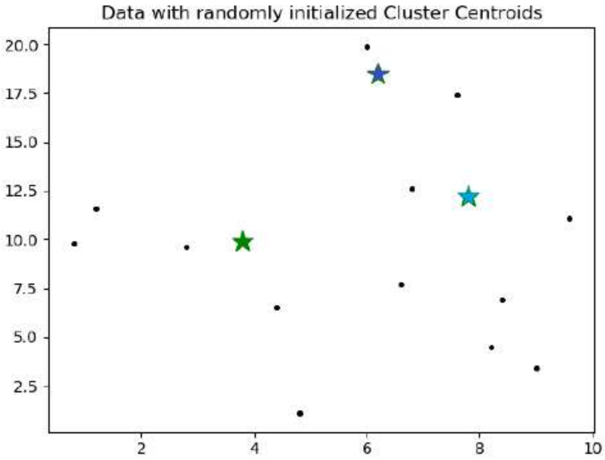
- K-means algorithm is by far the most popular / widely used clustering algorithm.
- K-means algorithm is an iterative algorithm. It tries to:
  - partition the dataset into **K** pre-defined distinct / non-overlapping subgroups (clusters)
  - each data point in a dataset is assigned to only one cluster.

K-Means Pictorial Representation



Input: Unlabeled Dataset
output: Group the data into two clusters (K=3)

K-Means Pictorial Representation

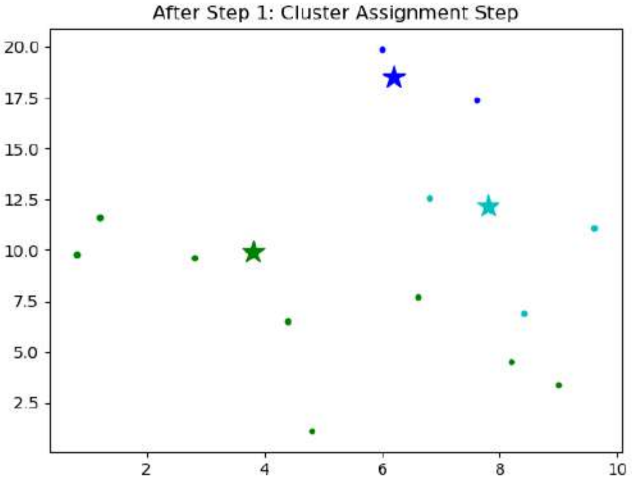


Initialization: Randomly initialize three points (as K=3), called cluster centroids (shown in green).

K-means algorithm is an iterative algorithm. It has two steps.

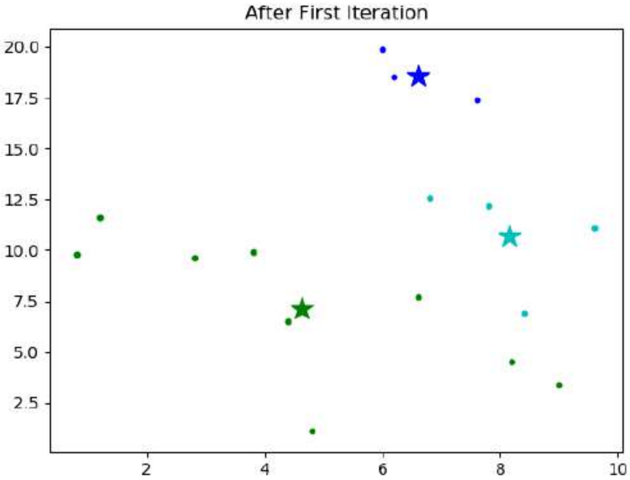
1. Cluster assignment step
2. Move / Update centroid step

K-Means Pictorial Representation



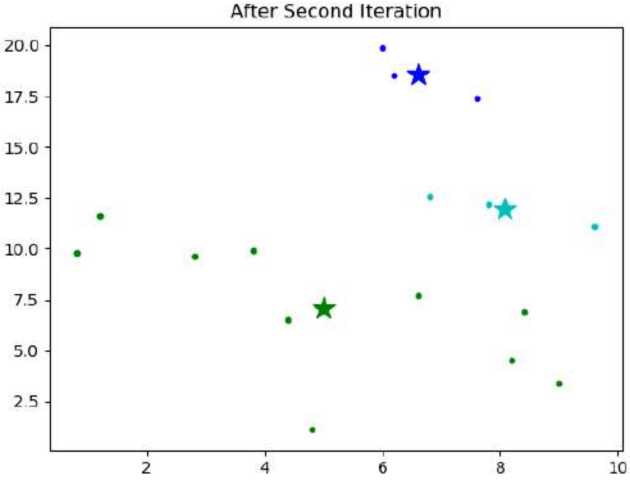
Cluster assignment(Step 1): Algorithm will iterate over all data points and depending on its distance to each cluster centroid, assign data point to closest cluster centroid.

K-Means Pictorial Representation



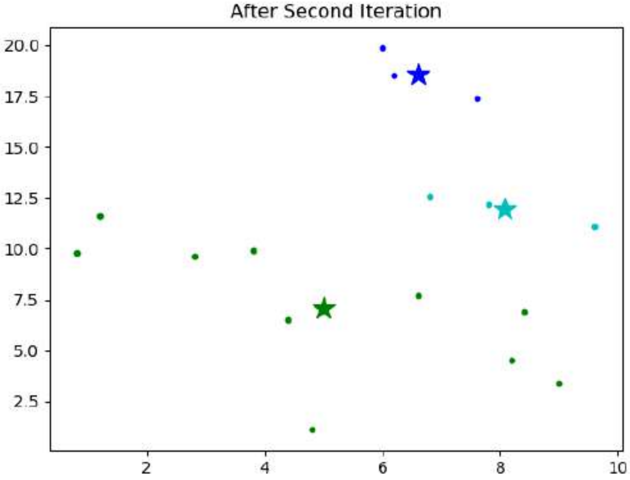
Move centroid (Step 2):
Calculate average of data points assigned to specific cluster and assign that value to cluster centroid (moving centroid to new coordinates).

K-Means Pictorial Representation



After two iterations
This shows result after completion of two iterations.

K-Means Pictorial Representation



Convergence
If you keep iterating nothing will change.

K-Means Algorithm

Input:

- K (number of clusters). Its a parameter.
- Unlabeled training Set

$$\{x_1, x_2, \cdots, x_m\}$$

- where $x_i \in \mathbb{R}^n$
- m datapoints with n dimensions.

K-Means Algorithm

Algorithm 1 K-Means Clustering Algorithm

Input: $x_1, x_2, \cdots, x_m$

Randomly initialize K cluster centroids: $\mu_1, \mu_2, \cdots, \mu_K \in \mathbb{R}^n$

Repeat until convergence[1] {

for $i = 1$ to m **do**
    $c_i \leftarrow$ index of closest cluster centroid to x_i
end for

for $k = 1$ to K **do**
    $\mu_k \leftarrow$ average / mean of points assigned to cluster k
end for }

[1] run until cluster centroids don't change

Decoding K-Means Algorithm

- Steps shown in blue (in previous slide) belongs to cluster assignment step (step 1).

---

for $i = 1$ to m **do**
  $c_i \leftarrow$ index of closest cluster centroid to x_i
end for

- This step is computing distance:

Decoding K-Means Algorithm

- Steps shown in blue (in previous slide) belongs to cluster assignment step (step 1).

for $i = 1$ to m **do**
    $c_i \leftarrow$ index of closest cluster centroid to x_i
end for

- This step is computing distance:

$$c_i \leftarrow \min_k ||x_i - \mu_k||^2$$

- Algorithm will iterate over all data points and depending on its distance to each cluster centroid, assign data point to closest cluster centroid (find value of k that minimizes distance).

Decoding K-Means Algorithm

- Steps shown in red (in previous slide) belongs to move centroid step (step 2).

for $k = 1$ to K **do**
  $\mu_k \leftarrow$ average / mean of points assigned to cluster k
end for

Decoding K-Means Algorithm

- Steps shown in red (in previous slide) belongs to move centroid step (step 2).

for $k = 1$ to K **do**
    $\mu_k \leftarrow$ average / mean of points assigned to cluster k
end for

- Concrete Example:

IF x_1, x_5, x_6, x_{10} are assigned to cluster 2, (from step 1) then

$\implies c_1 = 2, c_5 = 2, c_6 = 2, c_{10} = 2$

$\implies \mu_2 = \frac{1}{4}[x_1 + x_5 + x_6 + x_{10}] \in \mathbb{R}^n$

Decoding K-Means Algorithm

- Steps shown in red (in previous slide) belongs to move centroid step (step 2).

---

for $k = 1$ to K **do**
  $\mu_k \leftarrow$ average / mean of points assigned to cluster k
end for

- Concrete Example:

IF x_1, x_5, x_6, x_{10} are assigned to cluster 2, (from step 1) then

$\implies c_1 = 2, c_5 = 2, c_6 = 2, c_{10} = 2$

$\implies \mu_2 = \frac{1}{4}[x_1 + x_5 + x_6 + x_{10}] \in \mathbb{R}^n$

What if cluster centroid has zero data points assigned to it?

Distance Metrics

Distance metric uses distance function which provides a relationship metric between elements in the dataset.

Minkowski Distance:

$$dist(a,b) = \left(\sum_{i=1}^{n} (a_i - b_i)^p \right)^{\frac{1}{p}} \quad (1)$$

1. if p = 1, Manhattan Distance

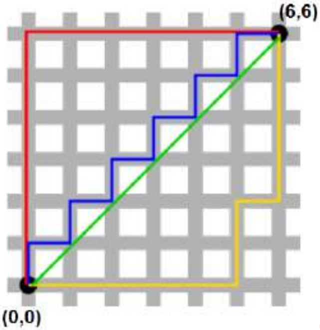
$$dist_{L1}(a,b) = \sum_{i=1}^{n} (\|a_i - b_i\|) \quad (2)$$

2. if p = 2, Euclidean Distance

$$dist_{L2}(a,b) = \sqrt{\sum_{i=1}^{n} (a_i - b_i)^2} \quad (3)$$

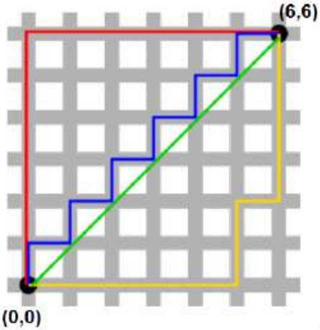
Manhattan or Euclidean Distance

Intuition of distances



Manhattan or Euclidean Distance

Intuition of distances

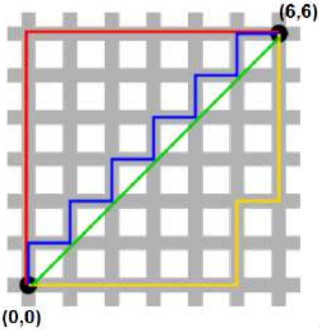


$$dist_{L1}(a,b) = \sum_{i=1}^{n}(\|a_i - b_i\|)$$

$$dist_{L1}(a,b) = (6-0) + (6-0) = 12 \quad (4)$$

Manhattan or Euclidean Distance

Intuition of distances



$$dist_{L1}(a,b) = \sum_{i=1}^{n}(\|a_i - b_i\|)$$

$$dist_{L1}(a,b) = (6-0) + (6-0) = 12 \quad (4)$$

$$dist_{L2}(a,b) = \sqrt{\sum_{i=1}^{n}(a_i - b_i)^2}$$

$$dist_{L2}(a,b) = \sqrt{6^2 + 6^2} = \sqrt{72} \approx 8.49 \quad (5)$$

In Manhattan / taxicab geometry, the red, yellow, and blue paths all have the same shortest path length of 12. In Euclidean geometry, the green line has length $6\sqrt{2} \approx 8.49$ and is the unique shortest path.

Importance?

Why to learn cost function?

1. To better understand algorithm

Importance?

Why to learn cost function?

1. To better understand algorithm
2. It is important to know cost function / objective function to debug algorithm.

Importance?

Why to learn cost function?

1. To better understand algorithm
2. It is important to know cost function / objective function to debug algorithm.
3. To fine tune parameters i.e. k in k-means clusters, and to avoid local minima in order to get better results.

Optimization Function

k-means algorithm optimization objective:

- c_i = index of cluster $(1, 2, \cdots, K)$ to which example x_i is currently assigned.

Optimization Function

k-means algorithm optimization objective:

- c_i = index of cluster $(1, 2, \cdots, K)$ to which example x_i is currently assigned.
- μ_k = cluster centroid k $(\mu_k \in \mathbb{R}^n)$, $k = \{1, 2, \cdots, K\}$.

Optimization Function

k-means algorithm optimization objective:

- c_i = index of cluster $(1, 2, \cdots, K)$ to which example x_i is currently assigned.
- μ_k = cluster centroid k ($\mu_k \in \mathbb{R}^n$), $k = \{1, 2, \cdots, K\}$.
- μ_{c_i} = cluster centroid of cluster to which example x_i has been assigned.

For example: If example x_i is assigned to cluster 5, then
$c_i \leftarrow 5$ and $\mu_{c_i} \leftarrow \mu_5$

Optimization Function

k-means algorithm optimization objective:

- c_i = index of cluster $(1, 2, \cdots, K)$ to which example x_i is currently assigned.
- μ_k = cluster centroid k $(\mu_k \in \mathbb{R}^n)$, $k = \{1, 2, \cdots, K\}$.
- μ_{c_i} = cluster centroid of cluster to which example x_i has been assigned.

  For example: If example x_i is assigned to cluster 5, then
  $c_i \leftarrow 5$ and $\mu_{c_i} \leftarrow \mu_5$

Optimization objective:

$$\min_{c,\mu} J(c_1, c_2, \cdots, c_m; \mu_1, \mu_2, \cdots, \mu_K) = \frac{1}{m} \sum_{i=1}^{m} ||x_i - \mu_{c_i}||^2 \tag{6}$$

Optimization Function

k-means algorithm optimization objective

$$\min_{c_1,\cdots,c_m;\mu_1,\cdots,\mu_K} J(c_1, c_2, \cdots, c_m; \mu_1, \mu_2, \cdots, \mu_K) = \frac{1}{m}\sum_{i=1}^{m} ||x_i - \mu_{c_i}||^2$$

Optimization Function

k-means algorithm optimization objective

$$\min_{c_1,\cdots,c_m;\mu_1,\cdots,\mu_K} J(c_1, c_2, \cdots, c_m; \mu_1, \mu_2, \cdots, \mu_K) = \frac{1}{m}\sum_{i=1}^{m} ||x_i - \mu_{c_i}||^2$$

- Cost function is trying to minimize distance between examples x_i and associated cluster centroids μ_{c_i}.

Optimization Function

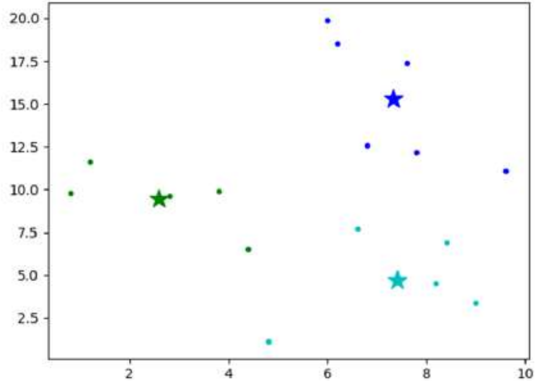
k-means algorithm optimization objective

$$\min_{c_1,\cdots,c_m;\mu_1,\cdots,\mu_K} J(c_1,c_2,\cdots,c_m;\mu_1,\mu_2,\cdots,\mu_K) = \frac{1}{m}\sum_{i=1}^{m} ||x_i - \mu_{c_i}||^2$$

- Cost function is trying to minimize distance between examples x_i and associated cluster centroids μ_{c_i}.
- Or in other words, cost function is finding parameters μ_i and c_i to minimize sum of squared distance between example and cluster centroid to which example is assigned.
- This function is sometimes also called as Distortion or Distortion function of K-Means.

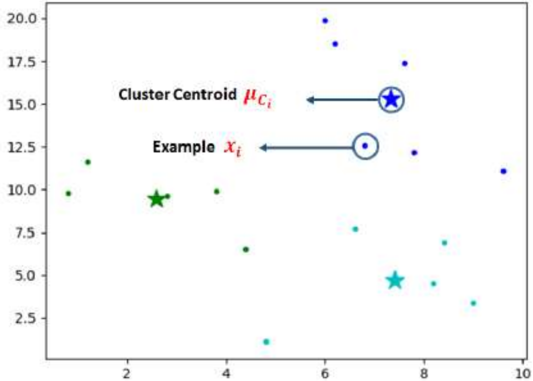
Optimization Function Visualization

$$\min_{c_1,\cdots,c_m;\mu_1,\cdots,\mu_K} J(c_1, c_2, \cdots, c_m; \mu_1, \mu_2, \cdots, \mu_K) = \frac{1}{m}\sum_{i=1}^{m} ||x_i - \mu_{c_i}||^2$$



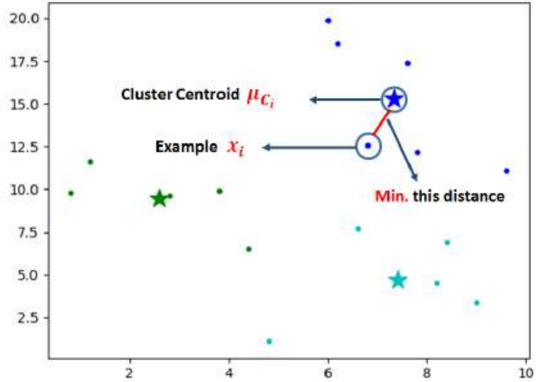
Optimization Function Visualization

$$\min_{c_1,\cdots,c_m;\mu_1,\cdots,\mu_K} J(c_1, c_2, \cdots, c_m; \mu_1, \mu_2, \cdots, \mu_K) = \frac{1}{m}\sum_{i=1}^{m} ||x_i - \mu_{c_i}||^2$$



Optimization Function Visualization

$$\min_{c_1,\cdots,c_m;\mu_1,\cdots,\mu_K} J(c_1, c_2, \cdots, c_m; \mu_1, \mu_2, \cdots, \mu_K) = \frac{1}{m}\sum_{i=1}^{m} ||x_i - \mu_{c_i}||^2$$



Looking at K-Means Algorithm w.r.t J

Input: $x_1, x_2, \cdots, x_m$

Randomly initialize K cluster centroids: $\mu_1, \mu_2, \cdots, \mu_K \in \mathbb{R}^n$

Repeat until convergence {

for $i = 1$ to m **do**

  $c_i \leftarrow$ index of closest cluster centroid to x_i

end for

for $k = 1$ to K **do**

  $\mu_k \leftarrow$ average / mean of points assigned to cluster k

end for

}

Do you see any relation between J and the two loops?

Looking at K-Means Algorithm w.r.t J

Input: $x_1, x_2, \cdots, x_m$

1: Randomly initialize K cluster centroids: $\mu_1, \mu_2, \cdots, \mu_K \in \mathbb{R}^n$

Repeat until convergence {

2: **for** $i = 1$ to m **do**
3: $c_i \leftarrow$ index of closest cluster centroid to x_i
4: **end for**

5: **for** $k = 1$ to K **do**
6: $\mu_k \leftarrow$ average / mean of points assigned to cluster k
7: **end for**

}

Do you see any relation between J and the two loops?

1. Loop 1 (cluster assignment step) is minimizing J w.r.t $c_1, c_2, \cdots, c_m$ while holding $\mu_1, \mu_2, \cdots, \mu_K$ fixed.

Looking at K-Means Algorithm w.r.t J

Input: $x_1, x_2, \cdots, x_m$

Randomly initialize K cluster centroids: $\mu_1, \mu_2, \cdots, \mu_K \in \mathbb{R}^n$

Repeat until convergence {

for $i = 1$ to m **do**
    $c_i \leftarrow$ index of closest cluster centroid to x_i
end for

for $k = 1$ to K **do**
    $\mu_k \leftarrow$ average / mean of points assigned to cluster k
end for

}

Do you see any relation between J and the two loops?

1. Loop 1 (cluster assignment step) is minimizing J w.r.t $c_1, c_2, \cdots, c_m$ while holding $\mu_1, \mu_2, \cdots, \mu_K$ fixed.
2. Loop 2 (move centroid step) is minimizing J w.r.t $\mu_1, \mu_2, \cdots, \mu_K$

Random Initialization

- Up till now we have discussed loops of K-Means algorithm. Any other detail missing?

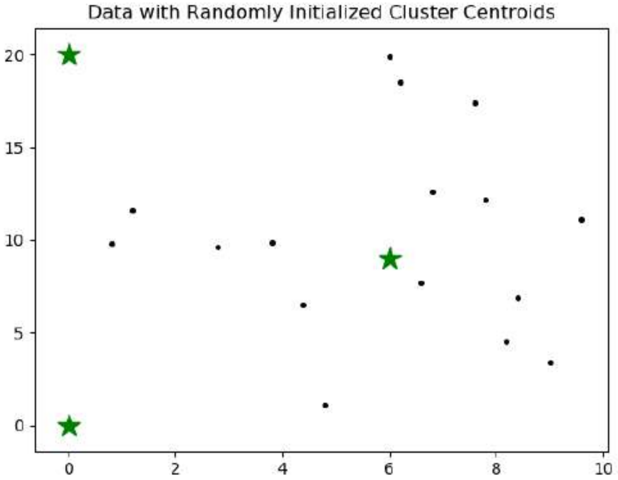
Random Initialization

- Up till now we have discussed loops of K-Means algorithm. Any other detail missing?
- First step of K-Means algorithm is:
  Randomly initialize K cluster centroids: $\mu_1, \mu_2, \cdots, \mu_K \in \mathbb{R}^n$

Random Initialization

- Up till now we have discussed loops of K-Means algorithm. Any other detail missing?
- First step of K-Means algorithm is:
  Randomly initialize K cluster centroids: $\mu_1, \mu_2, \cdots, \mu_K \in \mathbb{R}^n$
- This discussion on random initialization of centroid will also cover discussion on local optima.

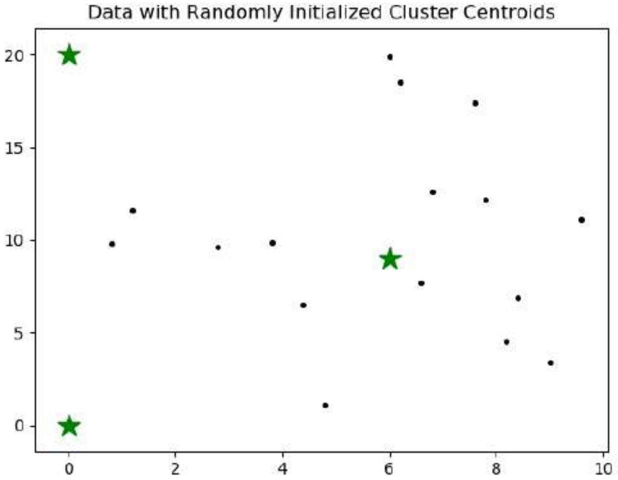
Random Initialization



- $K \ll m$
- What will happen if $K = or \approx m$?
- There are two strategies:

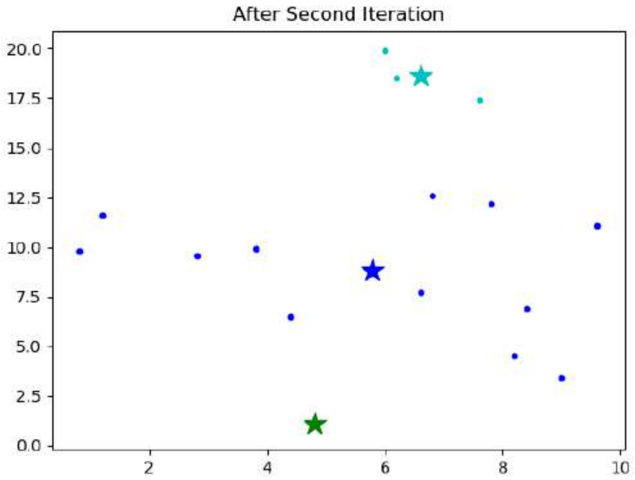
1. Randomly set coordinates of $\mu_1, \mu_2, \cdots, \mu_K$.
2. OR randomly pick K training examples and set $\mu_1, \mu_2, \cdots, \mu_K$ equal to these K examples.

Random Initialization



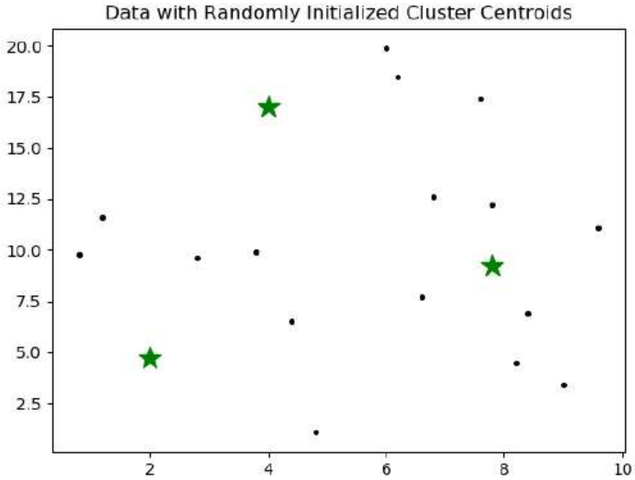
Random Initialization - Bad Initialization
- Stuck at local optima

Random Initialization



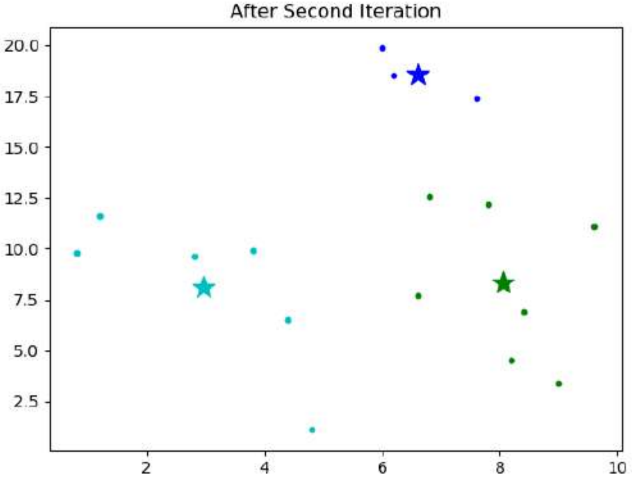
Random Initialization - Bad Initialization
- Stuck at local optima

Random Initialization



Random Initialization - Good Initialization

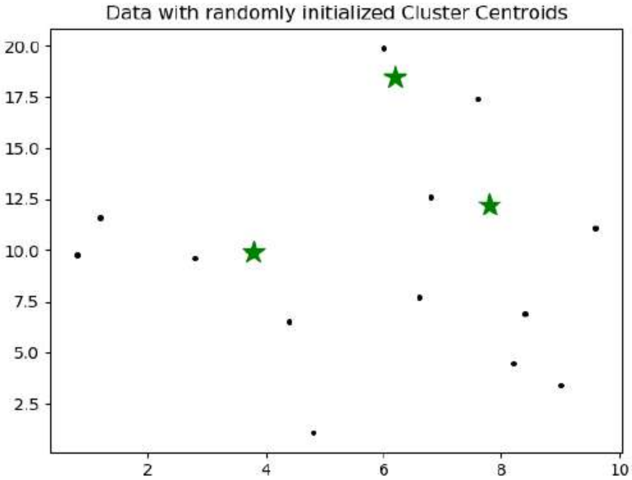
Random Initialization



Random Initialization - Good Initialization

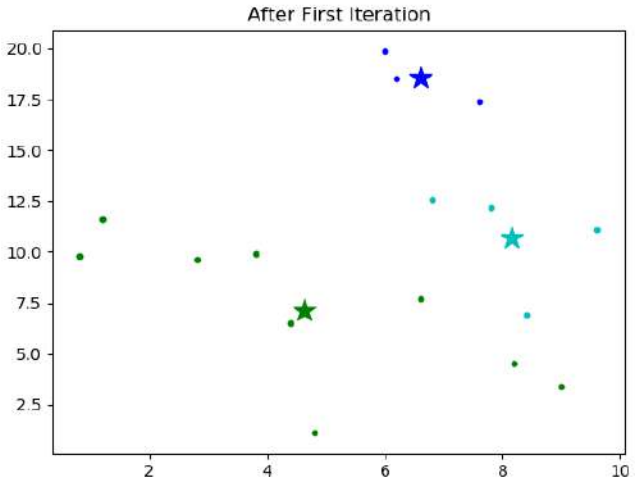
Random Initialization 1

- K-Means can converge on different solutions based on initialization of cluster centroids.



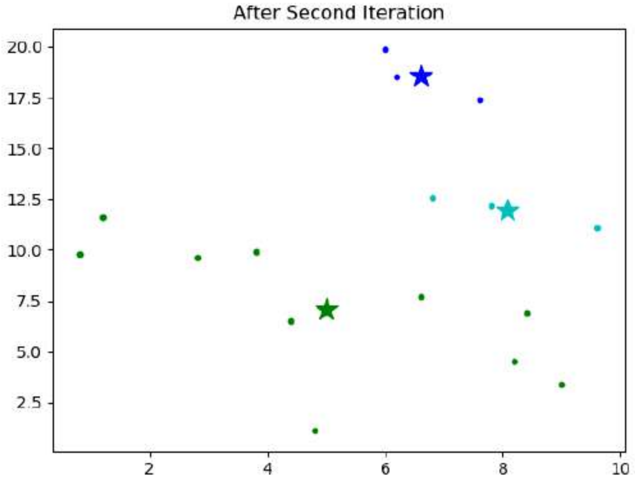
Random Initialization 1

- K-Means can converge on different solutions based on initialization of cluster centroids.



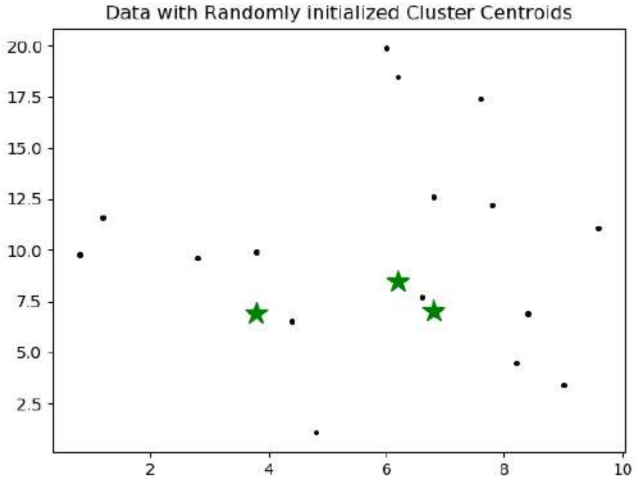
Random Initialization 1

- K-Means can converge on different solutions based on initialization of cluster centroids.



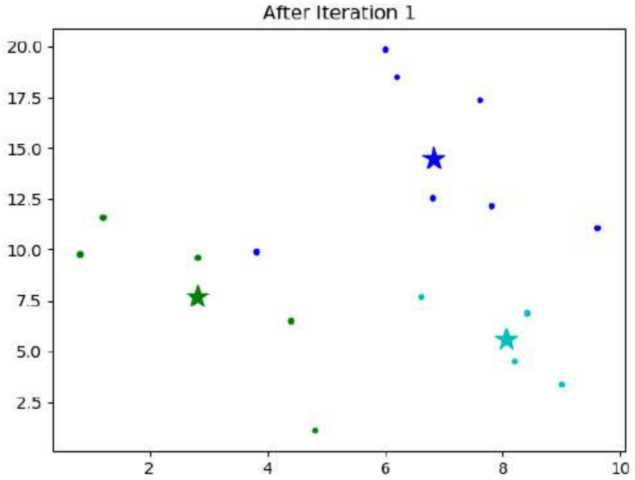
Random Initialization 2

- K-Means can converge on different solutions based on initialization of cluster centroids.



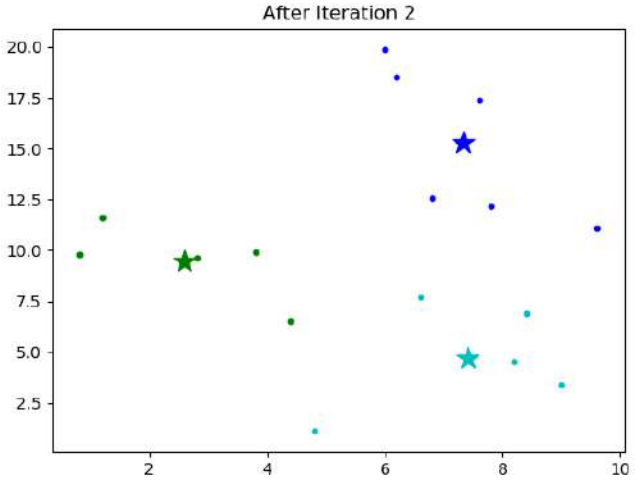
Random Initialization 2

- K-Means can converge on different solutions based on initialization of cluster centroids.

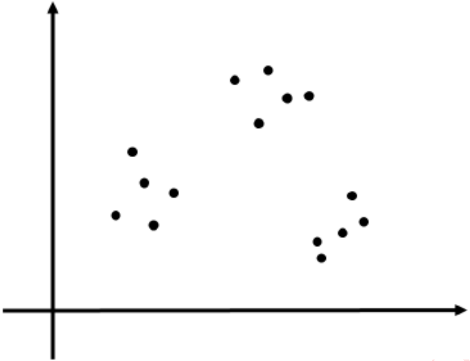


Random Initialization 2

- K-Means can converge on different solutions based on initialization of cluster centroids.

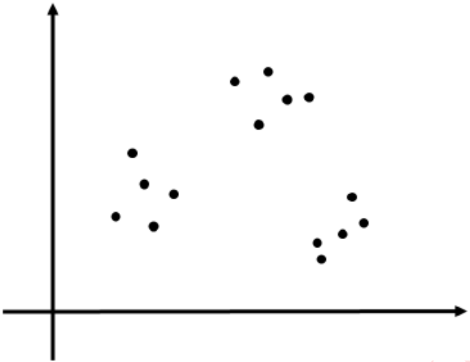


Random Initialization: Local Optima

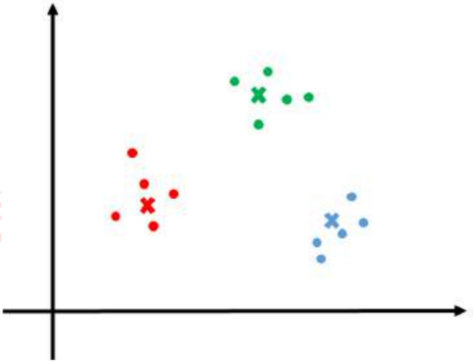


Data points

Random Initialization: Local Optima

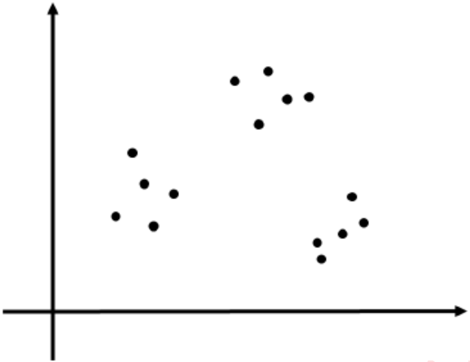


Data points

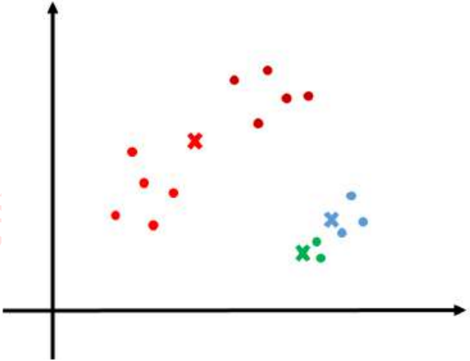


Result after good initialization

Random Initialization: Local Optima

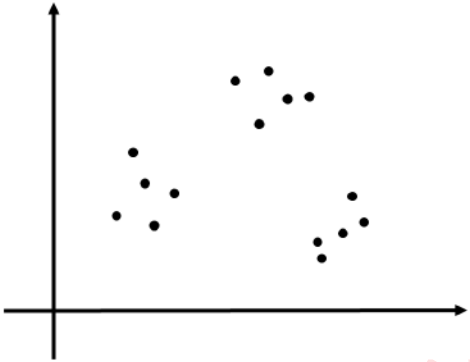


Data points

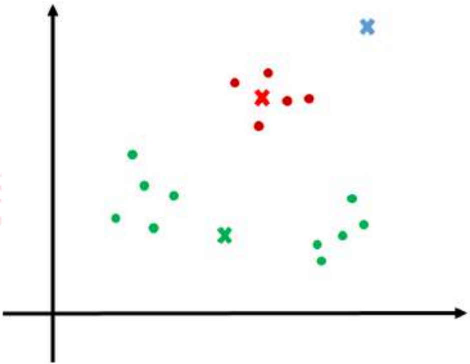


Not so good initialization, stuck in local optima i.e. K-means not doing a good job in minimizing distortion function $J : \min_{c_1,\cdots,c_m;\mu_1,\cdots,\mu_K} J$

Random Initialization: Local Optima



Data points



Not so good initialization, stuck in local optima i.e. K-means not doing a good job in minimizing distortion function $J : \min_{c_1,\cdots,c_m;\mu_1,\cdots,\mu_K} J$

Random Initialization - Avoiding Local Optima

Avoiding Local Optima

Algorithm 2 Random Initialization of K-Means Clustering Algorithm

for $I = 1$ to 100 **do**
  Randomly initialize K-Means.
  Run K-Means. Get $c_1, c_2, \cdots, c_m$ and $\mu_1, \mu_2, \cdots, \mu_K$.
  Computer J (Cost function / distortion Function). Refer Equation 6.
end for

Random Initialization - Avoiding Local Optima

for $I = 1$ to 100 **do**
  Randomly initialize K-Means.
  Run K-Means. Get $c_1, c_2, \cdots, c_m$ and $\mu_1, \mu_2, \cdots, \mu_K$.
  Computer J (Cost function / distortion Function). Refer Equation 6.
end for

Random Initialization - Avoiding Local Optima

for $I = 1$ to 100 **do**

Randomly initialize K-Means.

Run K-Means. Get $c_1, c_2, \cdots, c_m$ and $\mu_1, \mu_2, \cdots, \mu_K$.

Computer J (Cost function / distortion Function). Refer Equation 6.

end for

- After running it 100 times, pick clustering that achieved lowest J (validation of clusters).

$$\min_{c_1,\cdots,c_m;\mu_1,\cdots,\mu_K} J$$

Random Initialization - Avoiding Local Optima

for $I = 1$ to 100 **do**
  Randomly initialize K-Means.
  Run K-Means. Get $c_1, c_2, \cdots, c_m$ and $\mu_1, \mu_2, \cdots, \mu_K$.
  Computer J (Cost function / distortion Function). Refer Equation 6.
end for

- After running it 100 times, pick clustering that achieved lowest J (validation of clusters).

$$\min_{c_1,\cdots,c_m;\mu_1,\cdots,\mu_K} J$$

For larger values of K, even this method might not work!

Choosing number of K

Choosing number of K

- How to choose value for K i.e. number of clusters?

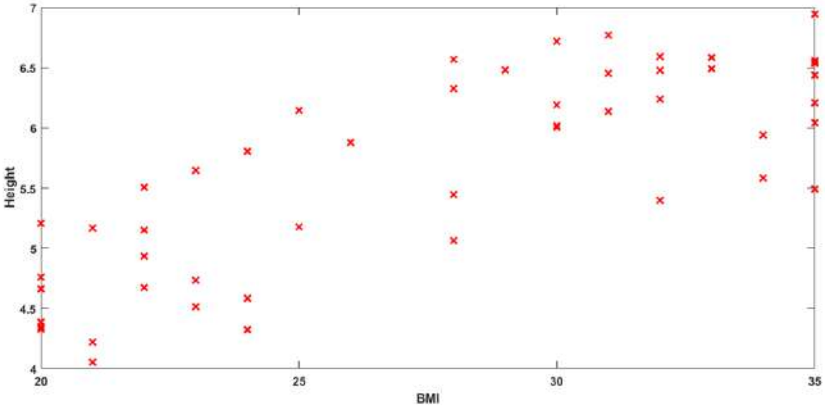
- K can be chosen by visually inspecting the data, to find distinct clusters.

Choosing number of K

Choosing number of K

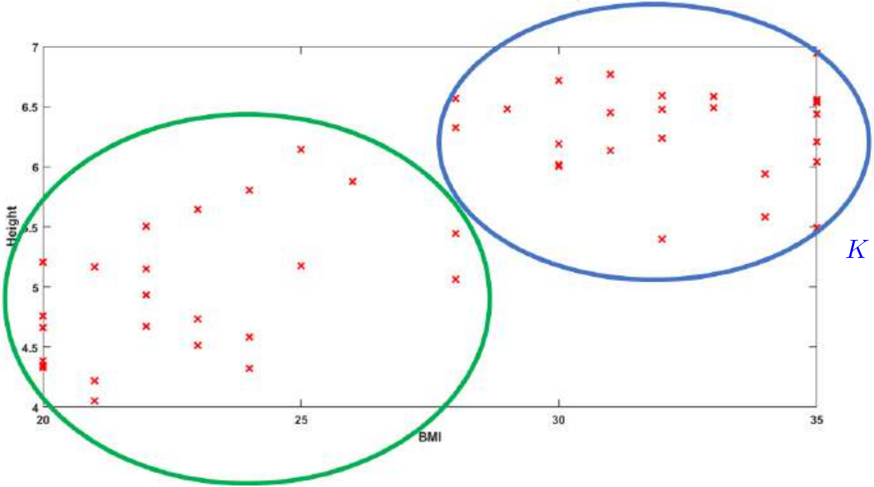
- How to choose value for K i.e. number of clusters?
  - K can be chosen by visually inspecting the data, to find distinct clusters.
  - But sometimes it is impossible!

Choosing number of K



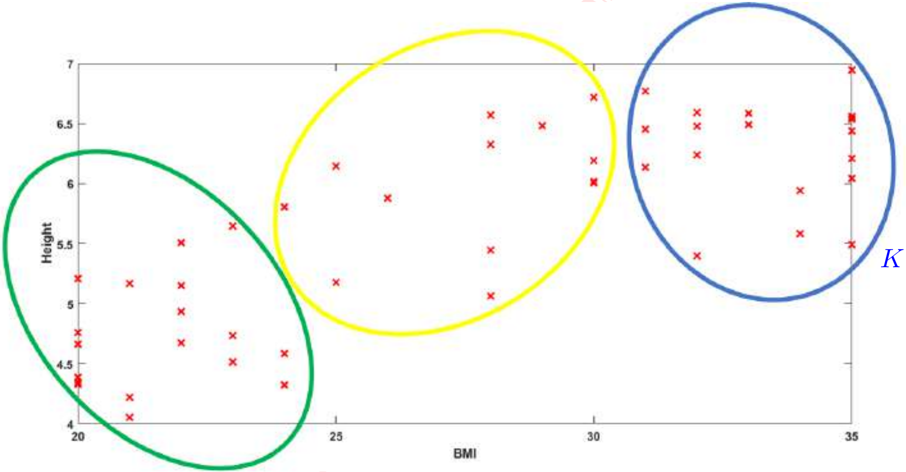
What is correct value of K i.e. number of clusters?

Choosing number of K



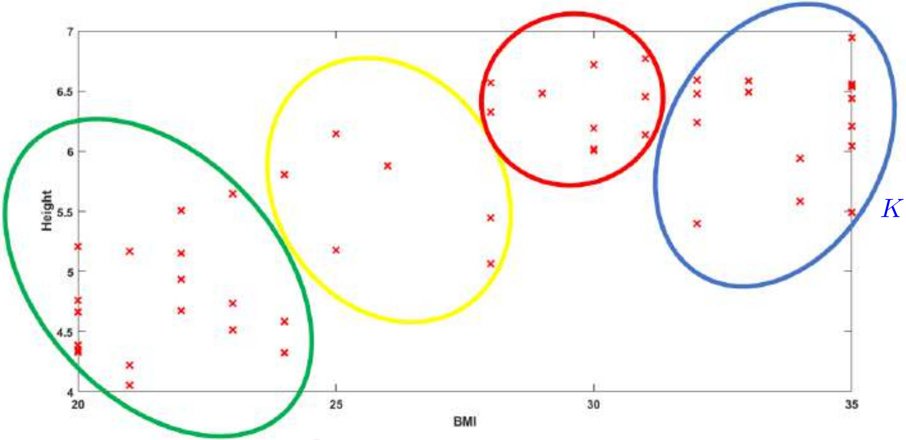
$K = 2$?

Choosing number of K



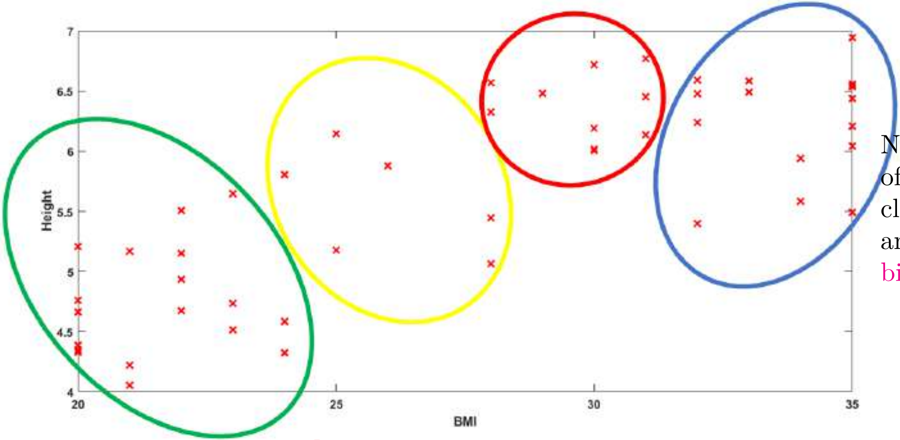
$K = 3$?

Choosing number of K



$K = 4$?

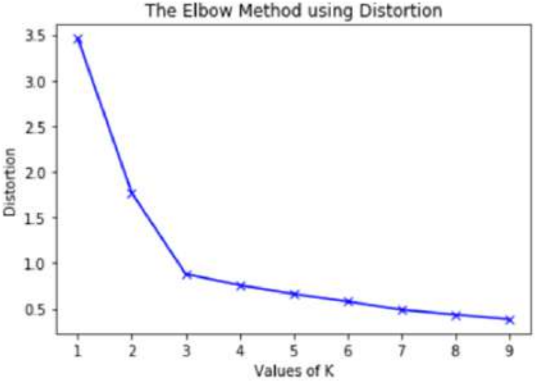
Choosing number of K



Number of clusters are ambiguous

Choosing number of K Algorithmically

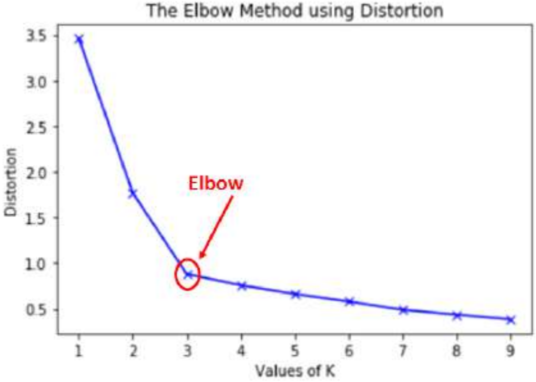
- Elbow method



Number of K can be chosen algorithmically by looking at this graph.

Choosing number of K Algorithmically

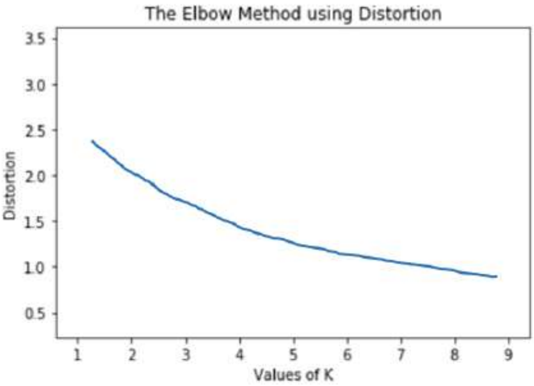
- Elbow method



If we get this kind of graph, then value of K can be chosen where elbow is formed.

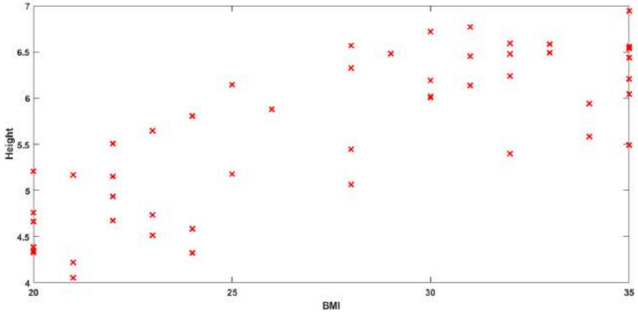
Choosing number of K Algorithmically

- Elbow method



Sometimes, elbow is not formed when plotting distortion with increasing K.

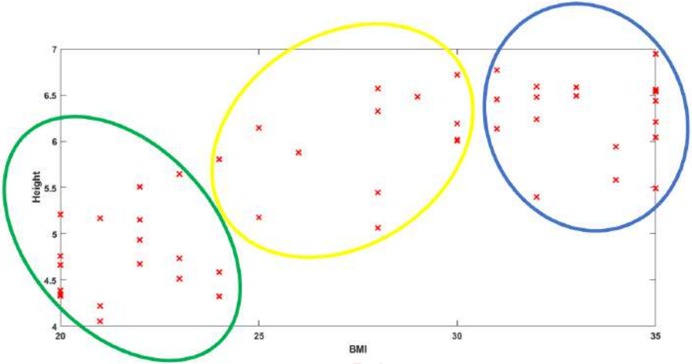
Choosing number of K - Study Problem



Know the problem

- Number of K can be chosen by knowing problem in hand.

Choosing number of K - Study Problem



Know the problem

- Number of K can be chosen by knowing problem in hand.
- For example, if in this problem, if we know that reason for running K-means is to identify “Observe”, “healthy” and “At risk”, then its reasonable to take K=3.

Section Contents

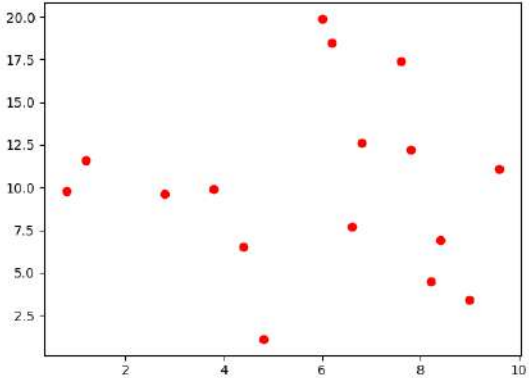
Dataset - Visualization

x	x_1
6.8	12.6
0.8	9.8
1.2	11.6
2.8	9.6
3.8	9.9
4.4	6.5
4.8	1.1
6.0	19.9
6.2	18.5
7.6	17.4
7.8	12.2
6.6	7.7
8.2	4.5
8.4	6.9
9.0	3.4
9.6	11.1

```python
"""
K-means, data taken from book "Principles of Data Mining":
    Chapter 14
@author: rizwan.khan
"""
import numpy as np
import matplotlib.pyplot as plt
#Create Training Set, 2D vector,  Values from book example
x=np.array([
[6.8, 12.6],[0.8, 9.8],
[1.2, 11.6],[2.8, 9.6],
[3.8, 9.9],[4.4, 6.5],
[4.8, 1.1],[6, 19.9],
[6.2, 18.5],[7.6, 17.4],
[7.8, 12.2],[6.6, 7.7],
[8.2, 4.5],[8.4, 6.9],
[9, 3.4],[9.6, 11.1]])
# create color dictionary for printing
colors = {0:'r', 1:'b'}
plt.figure(0)
plt.scatter(x[:, 0], x[:, 1], c='r', cmap=plt.cm.jet)
```

Dataset - Visualization

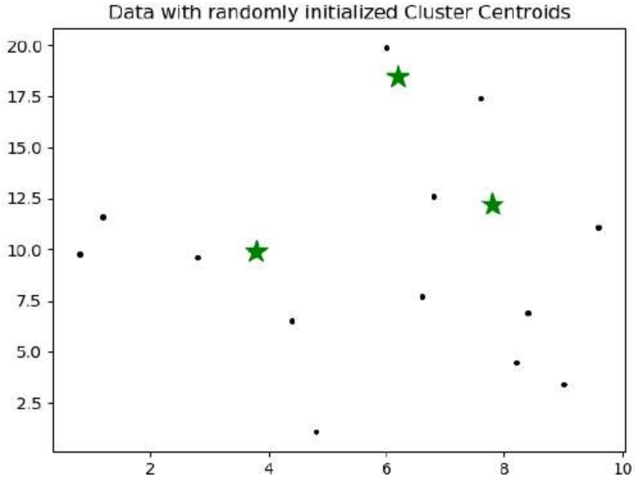
x	x_1
6.8	12.6
0.8	9.8
1.2	11.6
2.8	9.6
3.8	9.9
4.4	6.5
4.8	1.1
6.0	19.9
6.2	18.5
7.6	17.4
7.8	12.2
6.6	7.7
8.2	4.5
8.4	6.9
9.0	3.4
9.6	11.1



Cluster Centroid - Visualization

```
# Euclidean Distance Calculator
def dist(x, y):
    return np.sqrt(np.sum((x-y)**2))

# Number of clusters
K = 3

# X coordinates of random 3 centroids
C_x = np.array([3.8, 7.8, 6.2]) # Book Example Value

# Y coordinates of random 3 centroids
C_y = np.array([9.9, 12.2, 18.5])# Book Example Value

C = np.array(list(zip(C_x, C_y)), dtype=np.float32) # Merging x and y
print(C)  # Cluster Centroids

# Plotting along with the Centroids
plt.figure(1)
plt.scatter(x[:, 0], x[:, 1], c='#050505', s=7)  # s= size
plt.scatter(C_x, C_y, marker='*', s=200, c='g')
```

Cluster Centroid - Visualization



Variables initialization

```
# Cluster Lables(0, 1, 2)
clusters = np.zeros(len(x))

colors = ['g', 'c', 'b', 'y', 'r',' m']

# Variables used inside main loop
distances = np.zeros(K)
cluster = np.zeros(len(x))
count=0
how_many_in_one_cluster = 0

C_new = np.zeros(C.shape)
iteration = 2
```

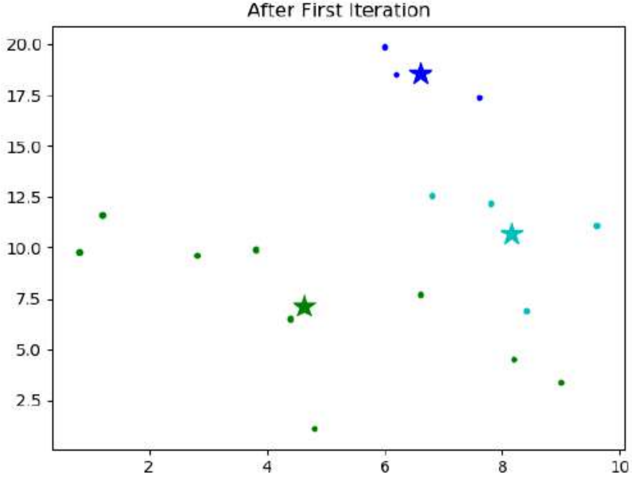
$K-$Means Main Loop

```
while (count < iteration): # or use difference in C and C_new to stop loop
#step 1: Cluster Assignment
    for i in range(len(x)): # loop over points
        for j in range(0, K, 1): # loop over K
            distances[j] = dist(x[i], C[j])
            cluster[i] = np.argmin(distances)

#step 2: Move / update cluster centroid (average values of X)
    C_new = np.zeros(C.shape) # intialize

    for k in range(K):  # Loop over K - clusters
       for i in range(len(x)): # Loop over all data points
           if cluster[i] == k: # if points belongs to specific cluster k
               C_new[k] = C_new[k] + x[i]  # Finding cluster of point with same label
               how_many_in_one_cluster = how_many_in_one_cluster +1 # keeping this values to take mean

       C_new[k] = C_new[k]/ how_many_in_one_cluster     # Average points to find new cluster centroid
       how_many_in_one_cluster = 0
```

$K-$Means Visualization

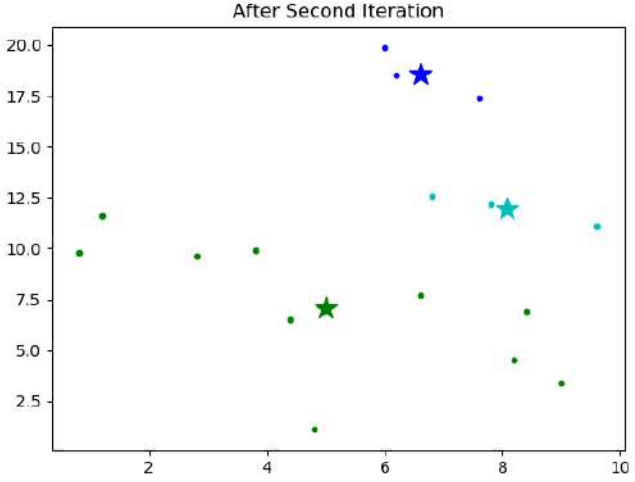
```
# Plotting along with the Centroids
    plt.figure(count+2)

    for i in range(len(x)):
        plt.scatter(x[i, 0], x[i, 1], c=colors[int(cluster[i])], s=10)  # s=size

    #plt.scatter(C_x, C_y, marker='*', s=200, c='r')
    for j in range(K):
        plt.scatter(C_new[j, 0], C_new[j, 1], marker='*', s=200, c=colors[j])

    print('                                                        ')
    print('*******************************************************')
    print('Cluster Centroid After iteration     :  ', count+1)
    print('*******************************************************')
    print(C_new)
    C = C_new  # update cluster centroid
    count=count+1
```

$K-$Means Visualization



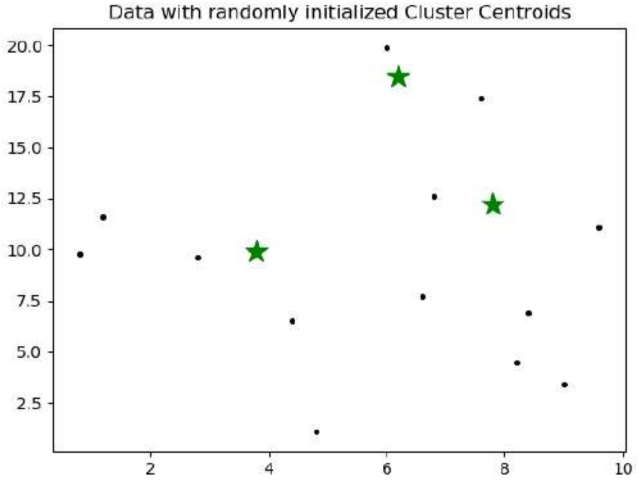

```
*************************************************
Cluster Centroid After iteration    :   1
*************************************************
[[ 4.62222222  7.12222222]
 [ 8.15       10.7       ]
 [ 6.6        18.6       ]]
```

$K-$Means Visualization

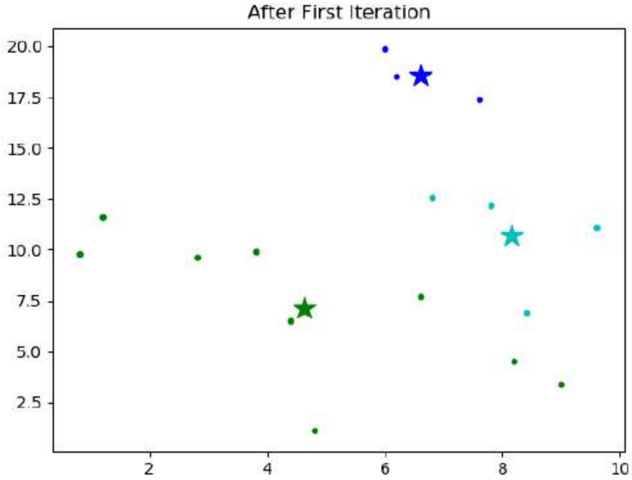



```
********************************************************
Cluster Centroid After iteration     :   2
********************************************************
[[ 5.          7.1       ]
 [ 8.06666667 11.96666667]
 [ 6.6        18.6       ]]
********************************************************
```

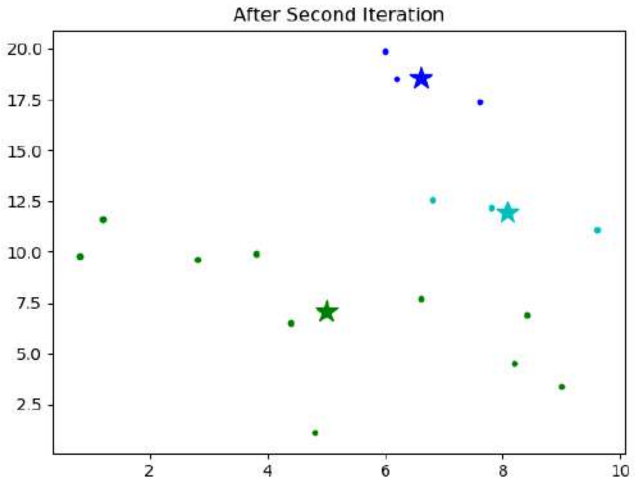
$K-$Means Visualization



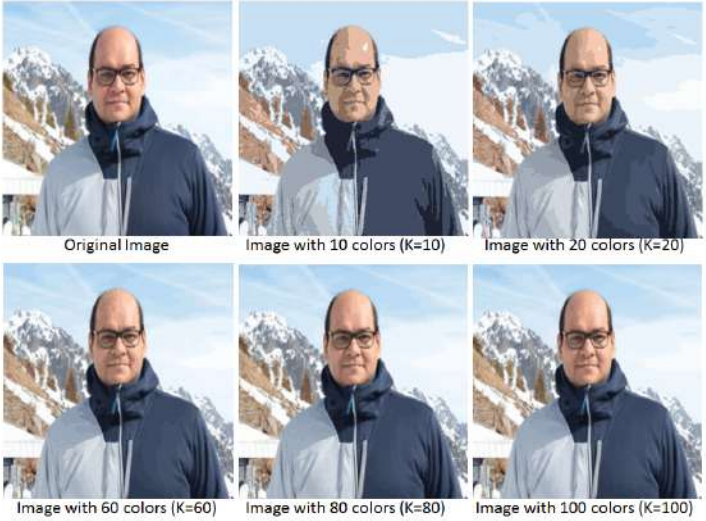
$K-$Means Visualization



K−Means Visualization



$K-$ Means Application on Image Compression



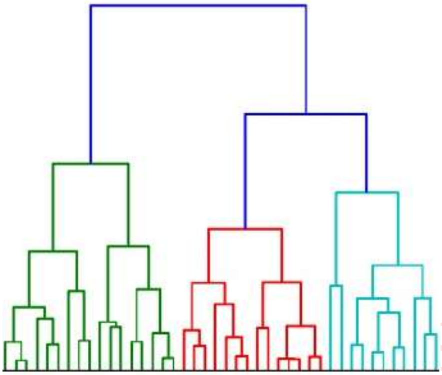
Section Contents

Introduction : Hierarchical Clustering

Introduction:

- Produces set of nested clusters, organized as a Hierarchical Tree. For example, all files and folders on our hard disk are organized in a hierarchy or looking at taxonomy of living things.

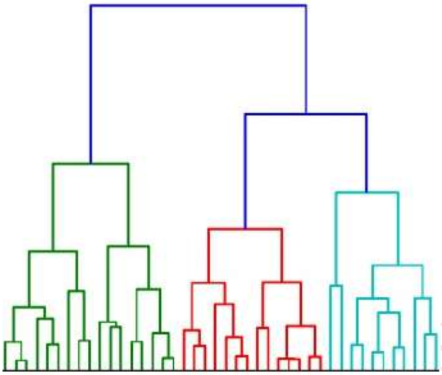
Introduction : Hierarchical Clustering



Introduction:

- Produces set of nested clusters, organized as a Hierarchical Tree. For example, all files and folders on our hard disk are organized in a hierarchy or looking at taxonomy of living things.
- Can be visualized as a dendogram. Dendogram is a tree like structure that records sequence of merges / splits.

Introduction : Hierarchical Clustering



Introduction:

- Produces set of nested clusters, organized as a Hierarchical Tree. For example, all files and folders on our hard disk are organized in a hierarchy or looking at taxonomy of living things.
- Can be visualized as a dendogram. Dendogram is a tree like structure that records sequence of merges / splits.
- Dendograms can reveal more meaningful taxonomies / structure in the data.

Introduction : Hierarchical Clustering

- There are two types of hierarchical clustering:
  1. Agglomerative:
     - Agglomerative is a bottom-up clustering method.
     - Assign each observation to its own cluster i.e. initially each data point is a cluster.
     - Then, compute the similarity (e.g., distance) between each of the clusters and join the two most similar clusters.
     - Proceed until there is only a single cluster left.

Introduction : Hierarchical Clustering

- There are two types of hierarchical clustering:
  1. Agglomerative:
     - Agglomerative is a bottom-up clustering method.
     - Assign each observation to its own cluster i.e. initially each data point is a cluster.
     - Then, compute the similarity (e.g., distance) between each of the clusters and join the two most similar clusters.
     - Proceed until there is only a single cluster left.
  2. Divisive:
     - Divisive clustering is a top-down clustering method.
     - Assign all of the observations to a single cluster and then partition the cluster to two least similar clusters.
     - Proceed recursively on each cluster until convergence / or there is one cluster for each observation.

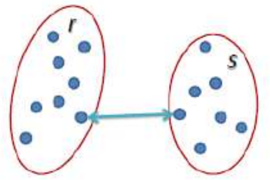
Agglomerative Hierarchical clustering

Algorithm 3 Agglomerative Hierarchical Clustering Algorithm

Input: $x_1, x_2, \cdots, x_m$

1: Each data point be a cluster.
2: **Repeat**
3: Merge the two closest cluster.
4: Update distances matrix.
5: **Until** only a single cluster remains.

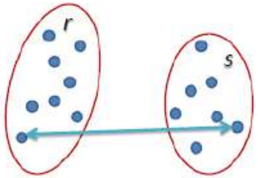
Distances Matrix - Hierarchical clustering



$$L(r,s) = \min(D(x_{ri}, x_{sj}))$$

- Key operation in Agglomerative Hierarchical clustering algorithm is computation of distance between clusters. Different distance definition of distance leads to different algorithms. For Example:
  1. Single Link Clustering (SLC)
     - In single linkage hierarchical clustering, the distance between two clusters is defined as the shortest distance between two points in each cluster.

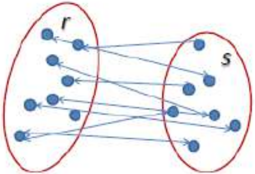
Distances Matrix - Hierarchical clustering



$$L(r,s) = \max(D(x_{ri}, x_{sj}))$$

- Key operation in Agglomerative Hierarchical clustering algorithm is computation of distance between clusters. Different distance definition of distance leads to different algorithms. For Example:
  1. Single Link Clustering (SLC)
     - In single linkage hierarchical clustering, the distance between two clusters is defined as the shortest distance between two points in each cluster.
  2. Complete Link Clustering (CLC)
     - In complete linkage hierarchical clustering, the distance between two clusters is defined as the longest distance between two points in each cluster.

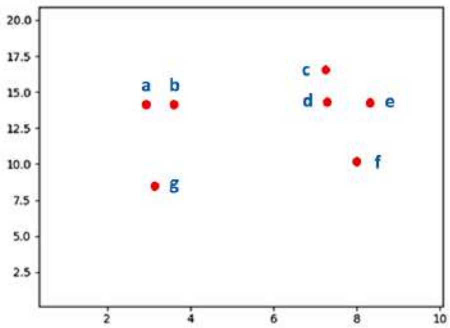
Distances Matrix - Hierarchical clustering



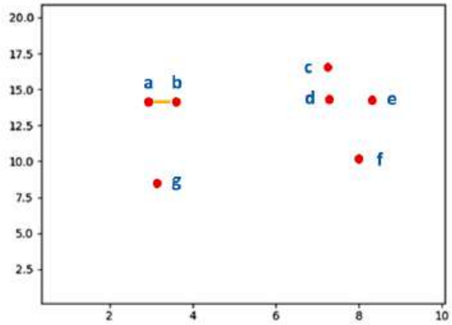
$$L(r,s) = \frac{1}{n_r n_s} \sum_{i=1}^{n_r} \sum_{j=1}^{n_s} D(x_{ri}, x_{sj})$$

- Key operation in Agglomerative Hierarchical clustering algorithm is computation of distance between clusters. Different distance definition of distance leads to different algorithms. For Example:
  1. Single Link Clustering (SLC)
     - In single linkage hierarchical clustering, the distance between two clusters is defined as the shortest distance between two points in each cluster.
  2. Complete Link Clustering (CLC)
     - In complete linkage hierarchical clustering, the distance between two clusters is defined as the longest distance between two points in each cluster.
  3. Average Link Clustering (ALC)
     - In average linkage hierarchical clustering, the distance between two clusters is defined as the average distance between each point in one cluster to every point in the other cluster

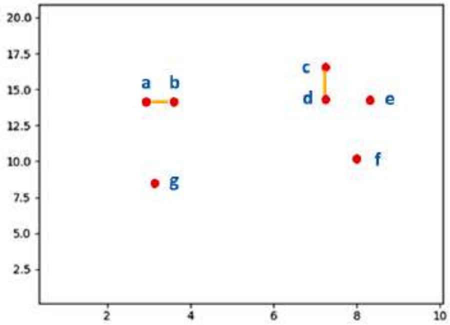
Agglomerative Hierarchical clustering - SLC : Visualization

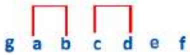


Agglomerative Hierarchical clustering - SLC : Visualization

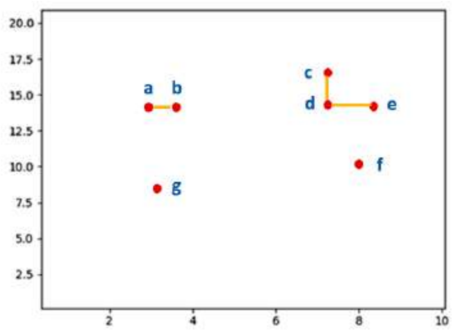


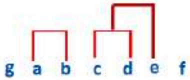
Agglomerative Hierarchical clustering - SLC : Visualization



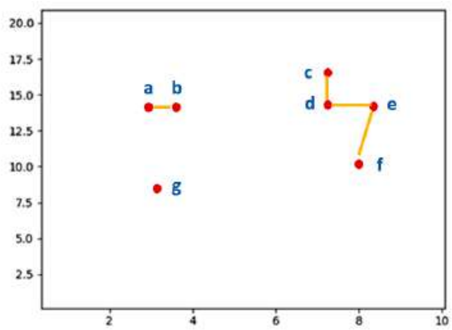


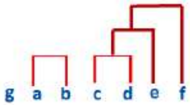
Agglomerative Hierarchical clustering - SLC : Visualization



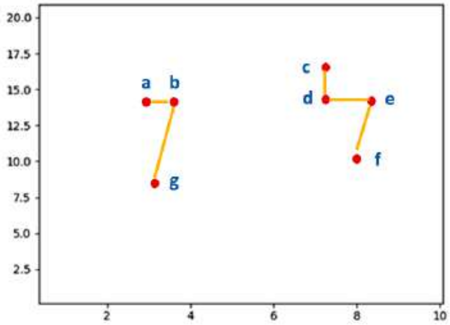


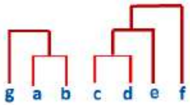
Agglomerative Hierarchical clustering - SLC : Visualization



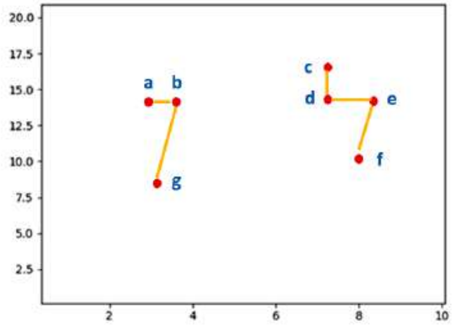


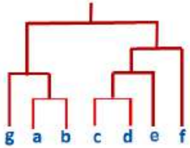
Agglomerative Hierarchical clustering - SLC : Visualization





Agglomerative Hierarchical clustering - SLC : Visualization





Section Contents

Conclusion

- K-means clustering is one of the most popular clustering algorithms.
- K-means is usually the first algorithm practitioners apply when solving clustering tasks as convergence is guaranteed.
- K-means doesn't learn the number of clusters from the data and requires it to be pre-defined, which sometimes is difficult.
- K-means can converge on different clusters based on initial values. It has Computational complexity[2] $\mathcal{O}(n^2)$.
- If there is overlapping between clusters, K-means doesn't have an intrinsic measure for uncertainty.
- Hierarchical clustering is a very useful way of segmentation.
- Hierarchical clustering has an advantage of not having to pre-define the number of clusters.
- Hierarchical clustering doesn't work well for large datasets. Computational complexity $\mathcal{O}(n^3)$.

[2] n = number of datapoints

Further Reading

Further Reading

- Can K-Means algorithm over-fit?

Further Reading

Further Reading

- Can K-Means algorithm over-fit?
- Validity of clusters - Cohesion and Separation (internal measure).

Further Reading

Further Reading

- Can K-Means algorithm over-fit?
- Validity of clusters - Cohesion and Separation (internal measure).
- Validity of clusters - Entropy, Purity etc (external measure).

Further Reading

Further Reading

- Can K-Means algorithm over-fit?
- Validity of clusters - Cohesion and Separation (internal measure).
- Validity of clusters - Entropy, Purity etc (external measure).
- Is K-Means algorithm NP-hard?

Further Reading

Further Reading

- Can K-Means algorithm over-fit?
- Validity of clusters - Cohesion and Separation (internal measure).
- Validity of clusters - Entropy, Purity etc (external measure).
- Is K-Means algorithm NP-hard?
- Different variants of K-Means algorithm:
  - Fuzzy C-Means Clustering
  - K-Means++

Further Reading

Further Reading

- Can K-Means algorithm over-fit?
- Validity of clusters - Cohesion and Separation (internal measure).
- Validity of clusters - Entropy, Purity etc (external measure).
- Is K-Means algorithm NP-hard?
- Different variants of K-Means algorithm:
  - Fuzzy C-Means Clustering
  - K-Means++
- Affect of distance measure used? Is it dependent on type of data?

Machine Learning
Regression

Dr. Rizwan Ahmed Khan

Outline

Section Contents

Reference Books

Reference books for this Module:

- Chapter 1 & 3: Pattern Recognition and Machine Learning, Christopher M. Bishop, Springer Books, latest edition.

Reference Books

Reference books for this Module:

- Chapter 1 & 3: Pattern Recognition and Machine Learning, Christopher M. Bishop, Springer Books, latest edition.
- Chapter 8: Machine Learning, Tom MITCHELL, McGraw Hill, latest edition.

Problem Formalization

Problem formalization

- Set of possible instances X i.e. $\{<\vec{x}_i, y_i>\}$
- Dataset D, given by $D = \{<\vec{x}_i, y_i>, \ldots, <\vec{x}_n, y_n>\} \subseteq X \times Y$
  Where:
  $\vec{x}_i$ is a feature vector ($\mathbb{R}^d$),
  y_i is a label / target variable,
  X is space of all features and
  Y is space of labels.
- Unknown target function $f : X \rightarrow Y$
- Set of function hypotheses $H = \{h | h : X \rightarrow Y\}$

Output:
- Hypothesis $h \in H$ that best approximates target function f.
- Output consists of one or more continuous variables (instead of predefined concepts / classes), the task is called ?

Problem Formalization

Problem formalization

- Set of possible instances X i.e. $\{< \vec{x}_i, y_i >\}$
- Dataset D, given by $D = \{< \vec{x}_i, y_i >, \ldots, < \vec{x}_n, y_n >\} \subseteq X \times Y$
  Where:
  $\vec{x}_i$ is a feature vector ($\mathbb{R}^d$),
  y_i is a label / target variable,
  X is space of all features and
  Y is space of labels.
- Unknown target function $f : X \to Y$
- Set of function hypotheses $H = \{h | h : X \to Y\}$

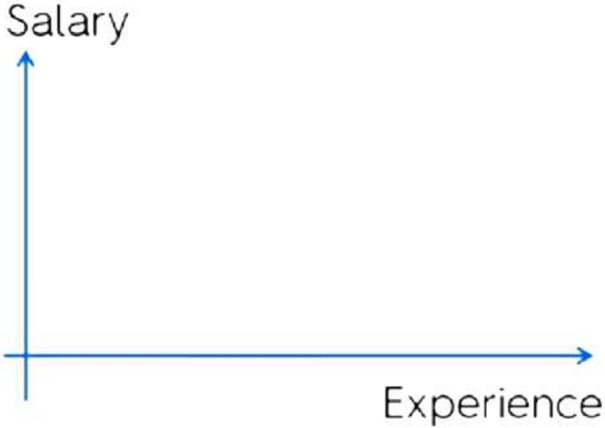
Output:

- Hypothesis $h \in H$ that best approximates target function f.
- Output consists of one or more continuous variables (instead of predefined concepts / classes), the task is called ? REGRESSION.

Section Contents

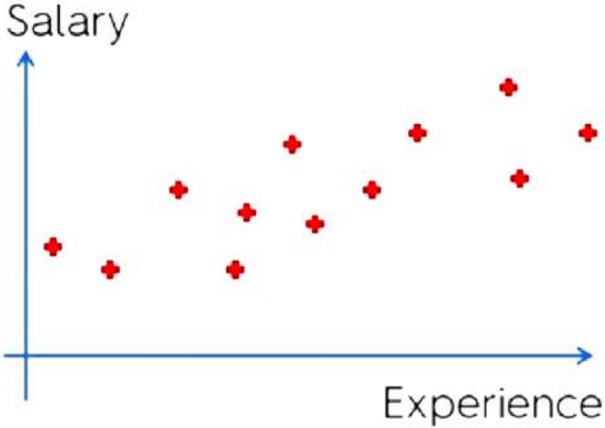
What?

REGRESSION is mapping of continuous inputs to continuous outputs.



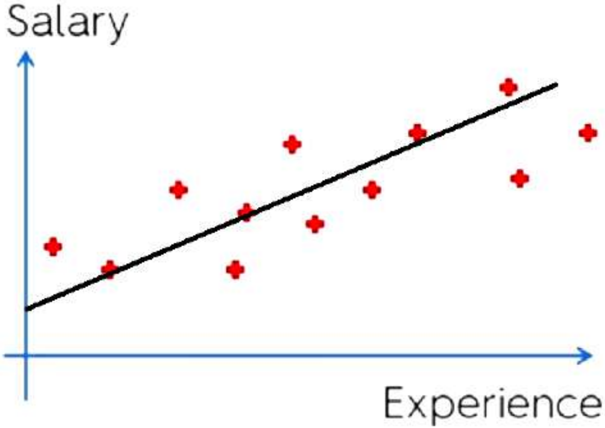
What?

REGRESSION is mapping of continuous inputs to continuous outputs.



What?

REGRESSION is mapping of continuous inputs to continuous outputs.



Why?

Why

Why “regression” is called “regression” in machine learning?

[1] http://www.stat.ucla.edu/~nchristo/statistics100C/history_regression.pdf

Why?

Why

Why “regression” is called “regression” in machine learning?

As per dictionary

“Regression” means to return to a previous and less advanced or worse state

[1] http://www.stat.ucla.edu/~nchristo/statistics100C/history_regression.pdf

Why?

Why

Why “regression” is called “regression” in machine learning?

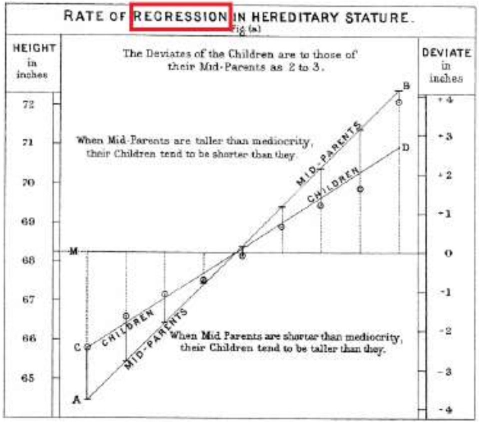
As per dictionary

“Regression” means to return to a previous and less advanced or worse state

Read Paper: Galton, Regression Towards Mediocrity in Hereditary Stature, 1886[1].

[1] `http://www.stat.ucla.edu/~nchristo/statistics100C/history_regression.pdf`

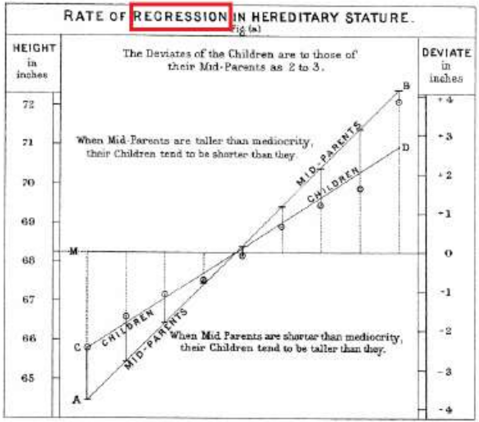
Why?



- According to psychoanalyst Sigmund Freud, regression is a defense mechanism leading to the temporary or long-term reversion of the ego to an earlier stage of development rather than handling unacceptable impulses in a more adaptive way.

[2] http://www.stat.ucla.edu/~nchristo/statistics100C/history_regression.pdf

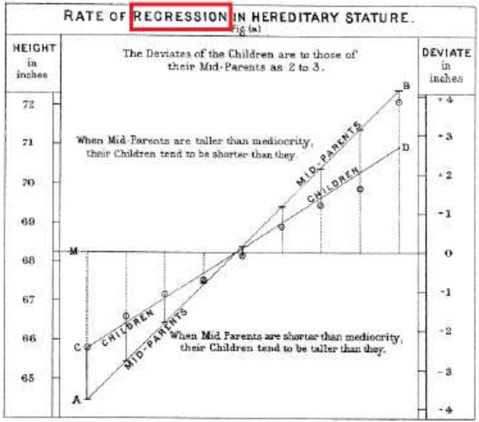
Why?



- According to psychoanalyst Sigmund Freud, regression is a defense mechanism leading to the temporary or long-term reversion of the ego to an earlier stage of development rather than handling unacceptable impulses in a more adaptive way.
- In machine learning the usage of word "regression" is linked to article by "Galton". In this article "Galton" showed that average height of population regresses towards the mean.

[2] http://www.stat.ucla.edu/~nchristo/statistics100C/history_regression.pdf

Why?

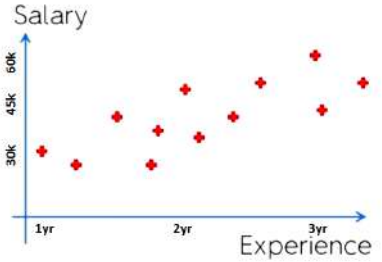


- According to psychoanalyst Sigmund Freud, regression is a defense mechanism leading to the temporary or long-term reversion of the ego to an earlier stage of development rather than handling unacceptable impulses in a more adaptive way.
- In machine learning the usage of word "regression" is linked to article by "Galton". In this article "Galton" showed that average height of population regresses towards the mean.
- So, this relationship b/w height of parent and children is linear with $m < 1$ or $\approx \frac{2}{3}$. From here this term is used in ML as tech. to find mathematical relationship b/w quantities.

Image from article: Galton, Regression Towards Mediocrity in Hereditary Stature, 1886[2].

[2] http://www.stat.ucla.edu/~nchristo/statistics100C/history_regression.pdf

Toy Example



Dataset:

Experience (Yrs)	Salary
1	30k
1.3	33k
1.8	36k
2	45k
3.3	65k
..	..

Problem Setting:

- Set of real-valued instances X
- Unknown target function $f : X \rightarrow Y$

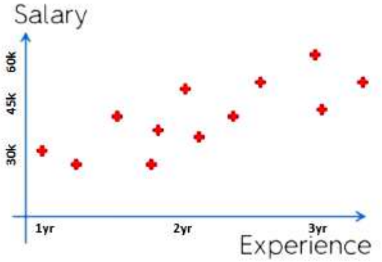
Input:

- "n" training examples $\{< x_i, y_i >\}$. For example x is experience and y is salary of a person.

Output:

- Function $f : X \rightarrow Y$.

Toy Example



Dataset:

Experience (Yrs)	Salary
1	30k
1.3	33k
1.8	36k
2	45k
3.3	65k
..	..

Problem Setting:

- Set of real-valued instances X
- Unknown target function $f : X \rightarrow Y$

Input:

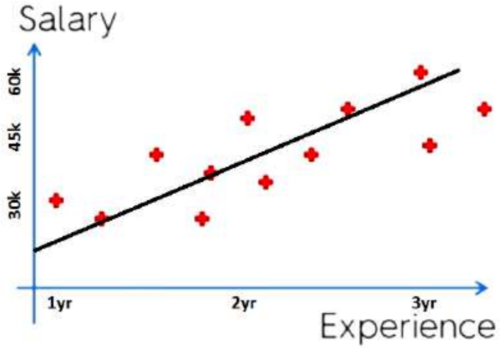
- "n" training examples $\{< x_i, y_i >\}$. For example x is experience and y is salary of a person.

Output:

- Function $f : X \rightarrow Y$.

What would be the salary for a person with experience of 2.5 years?

How?

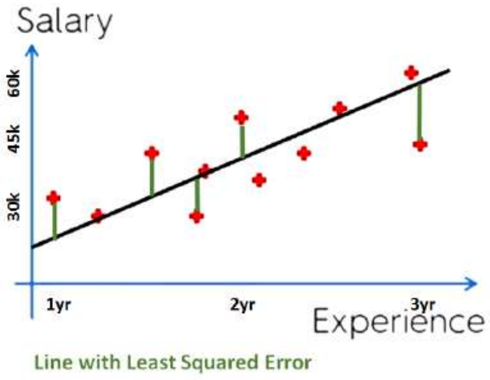


- Model relationship between inputs and outputs using linear function i.e. $y = mx + c$.
- For more complex problem this can be extended to non-linear function as well.
- This example is of linear regression with one variable or univariate linear regression.

How do we find best fit line? $y = mx + c$

1. Calculus
2. Random Search
3. Brute force

How?



- Model relationship between inputs and outputs using linear function i.e. $y = mx + c$.
- For more complex problem this can be extended to non-linear function as well.
- This example is of linear regression with one variable or univariate linear regression.

How do we find best fit line? $y = mx + c$

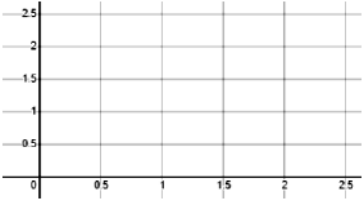
1. Calculus
2. Random Search
3. Brute force

Best line will be the one that minimizes the error between line and the data points.

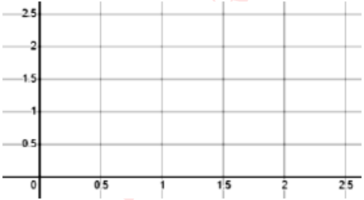
Effect of Parameters

Hypothesis $= mx + c$

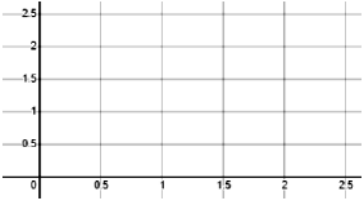
Choices of parameters:



- c = 1.5
- m = 0



- c = 0
- m = 0.5

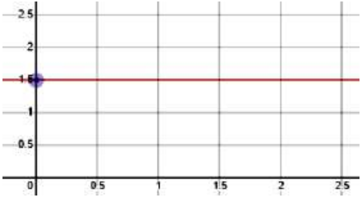


- c = 1
- m = 0.5

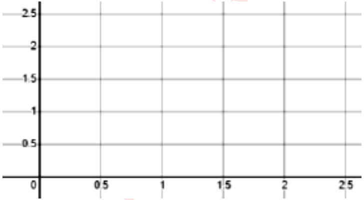
Effect of Parameters

Hypothesis $= mx + c$

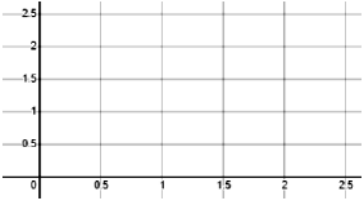
Choices of parameters:



- c = 1.5
- m = 0



- c = 0
- m = 0.5

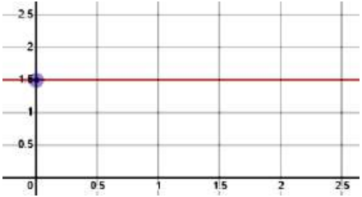


- c = 1
- m = 0.5

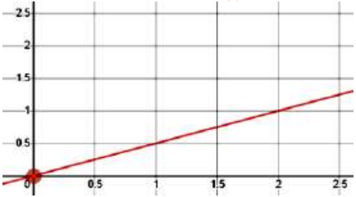
Effect of Parameters

Hypothesis $= mx + c$

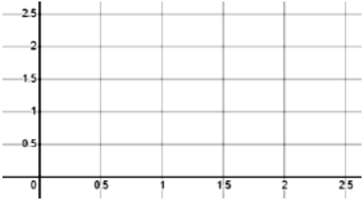
Choices of parameters:



- c = 1.5
- m = 0



- c = 0
- m = 0.5

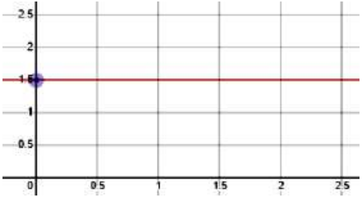


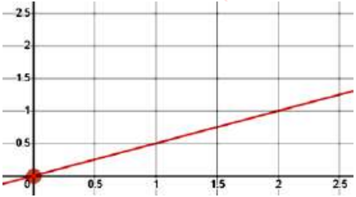
- c = 1
- m = 0.5

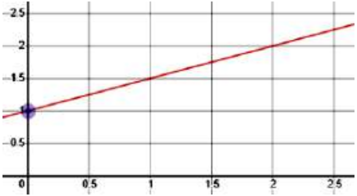
Effect of Parameters

Hypothesis $= mx + c$

Choices of parameters:







- c = 1.5
- m = 0

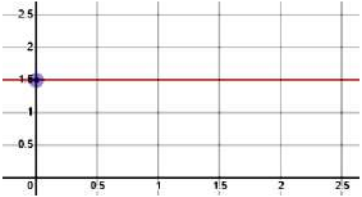
- c = 0
- m = 0.5

- c = 1
- m = 0.5

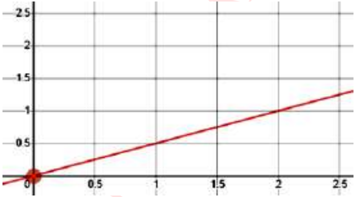
Effect of Parameters

Hypothesis $= mx + c$

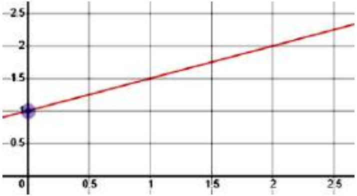
Choices of parameters:



- c = 1.5
- m = 0
- $f(x) = 1.5$



- c = 0
- m = 0.5
- $f(x) = 0.5\,x$



- c = 1
- m = 0.5
- $f(x) = 0.5\,x + 1$

Finding best constant function: Solution 1

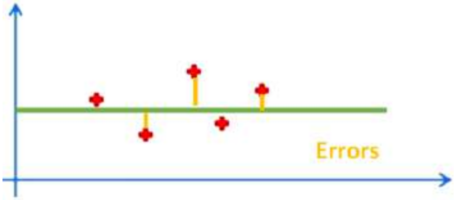
Solution 1:

Trying to find: $f(x) = c$.

So its a constant line without any slope m, and that line gives same output for any input.

Finding best constant function: Solution 1

Solution 1:
Trying to find: $f(x) = c$.
So its a constant line without any slope m, and that line gives same output for any input.

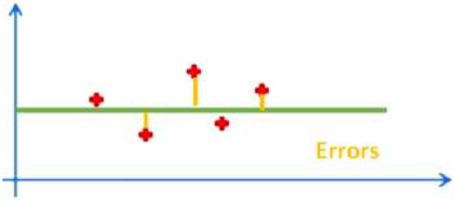


Finding best constant function: Solution 1

Solution 1:

Trying to find: $f(x) = c$.

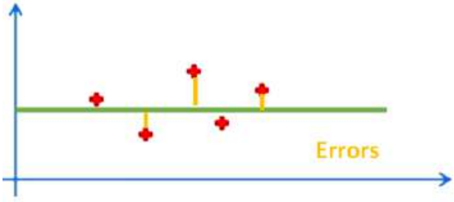
So its a constant line without any slope m, and that line gives same output for any input.



How to do it?

Finding best constant function: Solution 1

Solution 1:
Trying to find: $f(x) = c$.
So its a constant line without any slope m, and that line gives same output for any input.

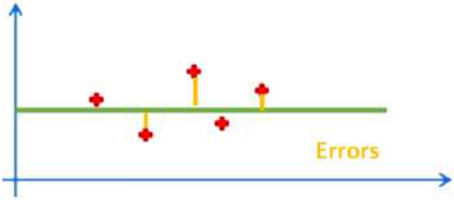


How to do it?
- Best line will be the one that minimizes the error between line and the data points.
- To find $h \in H$ that makes least errors on training data, loss functions are used.

Finding best constant function: Solution 1

Solution 1:
Trying to find: $f(x) = c$.
So its a constant line without any slope m, and that line gives same output for any input.

How to do it?

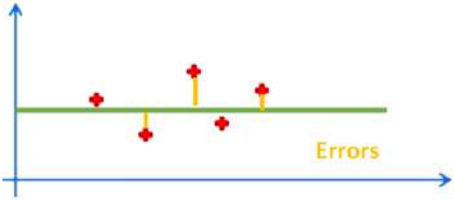


Zero-One Loss

$$\mathcal{L}_{0/1}(h) = \frac{1}{n}\sum_{i=1}^{n} \delta_{h(\mathbf{x}_i)\neq y_i}, \quad \text{where } \delta_{h(\mathbf{x}_i)\neq y_i} = \begin{cases} 1, & \text{if } h(\mathbf{x}_i) \neq y_i \\ 0, & \text{Otherwise} \end{cases} \tag{1}$$

Finding best constant function: Solution 1

Solution 1:
Trying to find: $f(x) = c$.
So its a constant line without any slope m, and that line gives same output for any input.



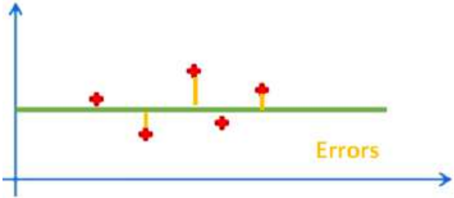
How to do it?

Sum of squared Loss

$$\mathcal{L}_{sq}(h) = \frac{1}{n}\sum_{i=1}^{n}(h(\mathbf{x}_i) - y_i)^2 \tag{2}$$

Finding best constant function: Solution 1

Solution 1:
Trying to find: $f(x) = c$.
So its a constant line without any slope m, and that line gives same output for any input.



How to do it?

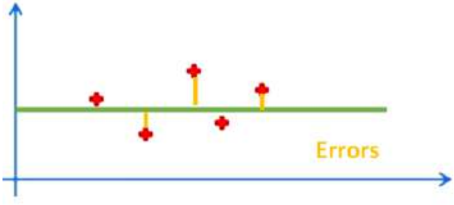
Absolute Loss

$$\mathcal{L}_{abs}(h) = \frac{1}{n}\sum_{i=1}^{n}|h(\mathbf{x}_i) - y_i| \quad (3)$$

Finding best constant function: Solution 1

Solution 1:
Trying to find: $f(x) = c$.
So its a constant line without any slope m, and that line gives same output for any input.

How to do it?



Sum of squared Loss will be used

$$E(c) = \sum_{i=1}^{n} (y_i - c)^2 \quad (4)$$

where, y_i = actual target value, E=error, n = number of samples and c = constant.
Find c that minimizes error.

Finding best constant function: Solution 1

Sum of squared error:

$$E(c) = \sum_{i=1}^{n} (y_i - c)^2 \qquad (5)$$

How to find c that minimizes error?

Finding best constant function: Solution 1

Sum of squared error:

$$E(c) = \sum_{i=1}^{n} (y_i - c)^2 \qquad (5)$$

How to find c that minimizes error?

Fermat's Theorem

If $f(x)$ has a local extremum at $x = a$ and f is differentiable at a, then $f'(a) = 0$.

Finding best constant function: Solution 1

Sum of squared error:

$$E(c) = \sum_{i=1}^{n} (y_i - c)^2 \qquad (5)$$

How to find c that minimizes error?

Fermat's Theorem

If $f(x)$ has a local extremum at $x = a$ and f is differentiable at a, then $f'(a) = 0$.

Take derivative : $\frac{d(E(c))}{dc}$ (How much Error wiggles as a function / changes in c).

Finding best constant function: Solution 1

Sum of squared error:

$$E(c) = \sum_{i=1}^{n} (y_i - c)^2 \qquad (5)$$

How to find c that minimizes error?

Fermat's Theorem

If $f(x)$ has a local extremum at $x = a$ and f is differentiable at a, then $f'(a) = 0$.

Take derivative : $\frac{d(E(c))}{dc}$ (How much Error wiggles as a function / changes in c).

$$\frac{d(E(c))}{dc} = \frac{d}{dc} \sum_{i=1}^{n} (y_i - c)^2$$

$$= \sum_{i=1}^{n} 2(y_i - c)(-1)\text{(set to zero to find min.)}$$

$$-\sum_{i=1}^{n} 2(y_i - c) = 0\text{(Solve for c)}$$

$$\sum_{i=1}^{n} (y_i) = \sum_{i=1}^{n} c \Longrightarrow n.c = \sum_{i=1}^{n} (y_i) \Longrightarrow c = \frac{\sum_{i=1}^{n} (y_i)}{n} \qquad (6)$$

Rem*: $\sum_{i=1}^{n} c = n$ times summation of constant $= n.c$

Finding best constant function: Solution 1

Sum of squared error:

$$E(c) = \sum_{i=1}^{n} (y_i - c)^2 \qquad (5)$$

How to find c that minimizes error?

Fermat's Theorem

If $f(x)$ has a local extremum at $x = a$ and f is differentiable at a, then $f'(a) = 0$.

Take derivative : $\frac{d(E(c))}{dc}$ (How much Error wiggles as a function / changes in c).

$$\frac{d(E(c))}{dc} = \frac{d}{dc}\sum_{i=1}^{n}(y_i - c)^2$$

$$= \sum_{i=1}^{n} 2(y_i - c)(-1) \text{(set to zero to find min.)}$$

$$-\sum_{i=1}^{n} 2(y_i - c) = 0 \text{(Solve for c)}$$

$$\sum_{i=1}^{n}(y_i) = \sum_{i=1}^{n} c \Longrightarrow n.c = \sum_{i=1}^{n}(y_i) \Longrightarrow c = \frac{\sum_{i=1}^{n}(y_i)}{n} \qquad (6)$$

Rem*: $\sum_{i=1}^{n} c = n$ times summation of constant $= n.c$

So, its MEAN value.

Best line that passes through origin: Solution 2

Solution 2:

Trying to find: $f(x) = mx$.
So its a Linear Regression / best line with Zero Intercept or a line that passes through origin:

Best line that passes through origin: Solution 2

Solution 2:

Do it yourself!

Trying to find: $f(x) = mx$.
So its a Linear Regression / best line with Zero Intercept or a line that passes through origin:

Best line that passes through origin: Solution 2

Solution 2:

Do it yourself!

Trying to find: $f(x) = mx$.
So its a Linear Regression / best line with Zero Intercept or a line that passes through origin:

$$\frac{d(E)}{dm} = \frac{d}{dm}\sum_{i=1}^{n}(y_i - mx_i)^2$$

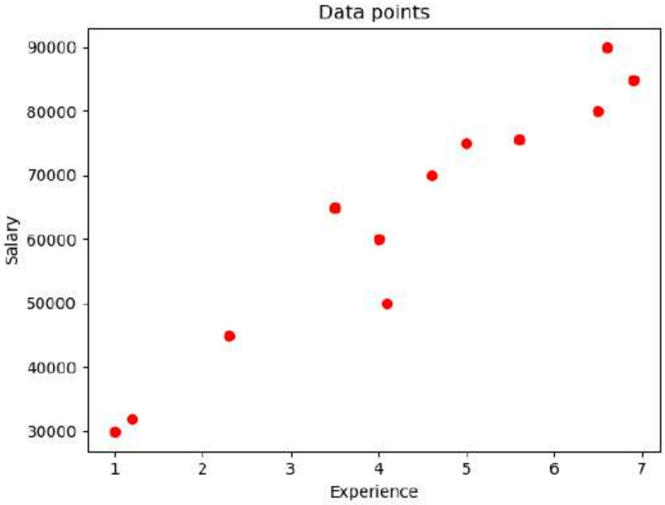
$$= \sum_{i=1}^{n} 2(y_i - mx_i)(-x_i) \text{(set to zero to find min.)}$$

$$-\sum_{i=1}^{n} 2(y_i - mx_i)(x_i) = 0 \text{(Solve for m)}$$

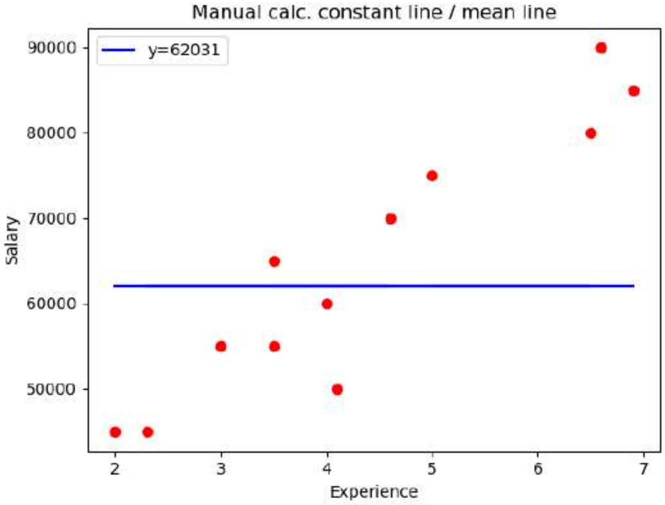
$$\sum_{i=1}^{n}(y_i x_i) = \sum_{i=1}^{n} m{x_i}^2 \implies \sum_{i=1}^{n}(y_i x_i) = m\sum_{i=1}^{n}{x_i}^2$$

$$\implies m = \frac{\sum_{i=1}^{n}(y_i x_i)}{\sum_{i=1}^{n}{x_i}^2} \tag{7}$$

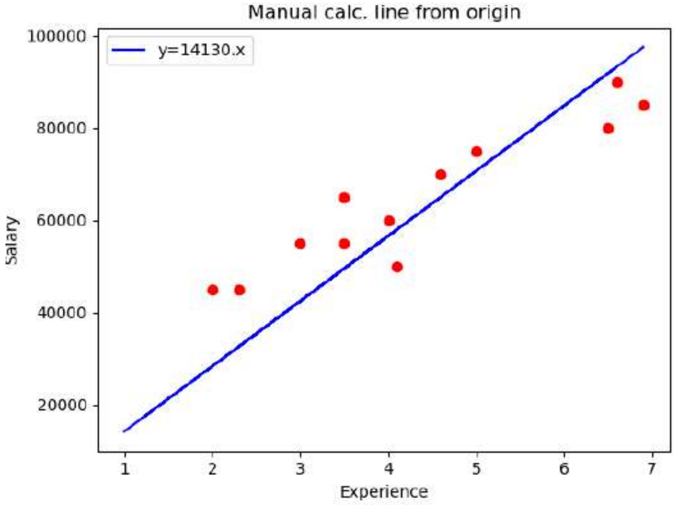
Visualization



Visualization



Visualization



Section Contents

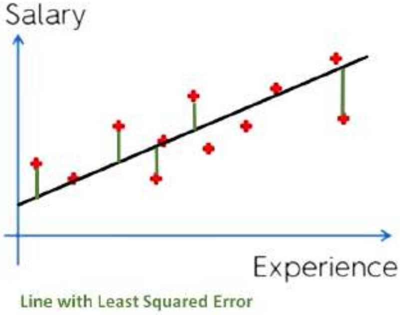
Cost function

Define cost function to find best line for the dataset.

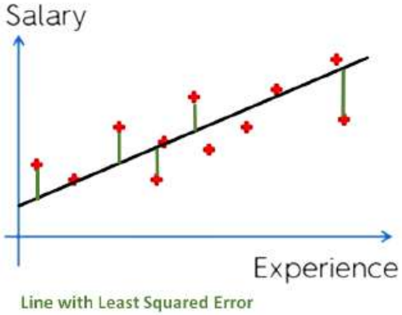
Dataset:

Experience (Yrs)	Salary
1	30k
1.3	33k
1.8	36k
2	45k
3.3	65k
..	..

Hypothesis $= mx + c$
How to choose m and c, which are parameters of the model.

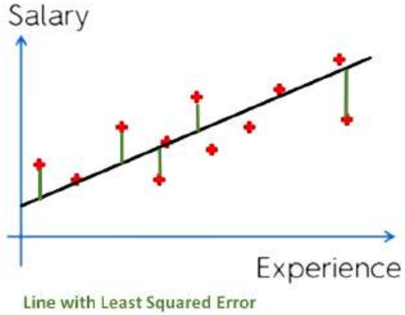


Cost function



Choose m and c such that value of hypothesis ($h = mx + c$) becomes as close as possible to training data $< x_i, y_i >$.

Cost function



Choose m and c such that value of hypothesis ($h = mx + c$) becomes as close as possible to training data $< x_i, y_i >$.

let's formalize this:

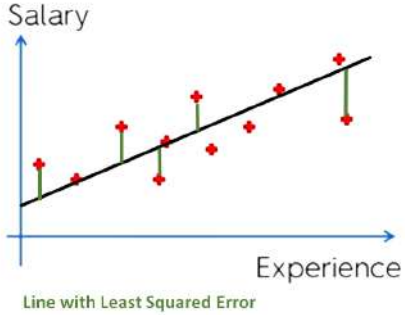
$$\underset{m,c}{\operatorname{argmin}} \sum_{i=1}^{n} (\hat{y}_i - y_i)^2 \quad (8)$$

Where $\hat{y}_i$ = predicted value, y_i is actual value and n is total number of training samples.

$$J(m, c) = \frac{1}{2n} \underset{m,c}{\operatorname{argmin}} \sum_{i=1}^{n} (\hat{y}_i - y_i)^2 \quad (9)$$

where $\hat{y}_i = mx_i + c$ and $J(m, c)$ is cost / loss function.

Cost function



Choose m and c such that value of hypothesis $(h = mx + c)$ becomes as close as possible to training data $< x_i, y_i >$.

let's formalize this:

$$\underset{m,c}{\text{argmin}} \sum_{i=1}^{n} (\hat{y}_i - y_i)^2 \tag{8}$$

Where $\hat{y}_i$ = predicted value, y_i is actual value and n is total number of training samples.

$$J(m, c) = \frac{1}{2n} \underset{m,c}{\text{argmin}} \sum_{i=1}^{n} (\hat{y}_i - y_i)^2 \tag{9}$$

where $\hat{y}_i = mx_i + c$ and $J(m, c)$ is cost / loss function.

- **Aim** to find values of m,c that minimizes cost function $J(m, c)$ (squared error function).

Visualize Cost function in 2D

- Hypothesis:
  $h_x = mx + c$
- Parameters:
  m and c
- Cost function:

$$J(m,c) = \frac{1}{2n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$$

- Goal:

$$\underset{m,c}{\operatorname{argmin}}\, J(m,c)$$

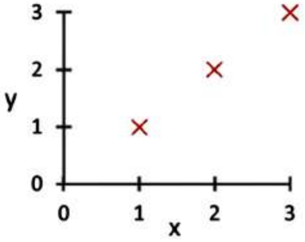
- To visualize cost function J in 2D (one parameter and predicted value), set $c = 0$, so $h_x = mx$. Thus goal becomes

$$\underset{m}{\operatorname{argmin}}\, J(m)$$

- By setting $c = 0$ means we are only considering line from origin with some slope m

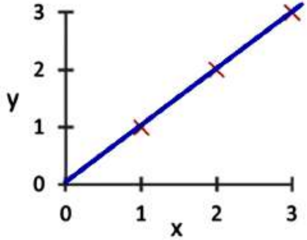
Visualize Cost function in 2D

- Hypothesis function: $h_x = mx$
- Hypothesis is function of x, while cost function is a function of parameter m.



Visualize Cost function in 2D

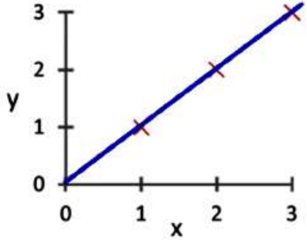
- Hypothesis function: $h_x = mx$
- Hypothesis is function of x, while cost function is a function of parameter m.



for $m = 1$

Visualize Cost function in 2D

- Hypothesis function: $h_x = mx$
- Hypothesis is function of x, while cost function is a function of parameter m.



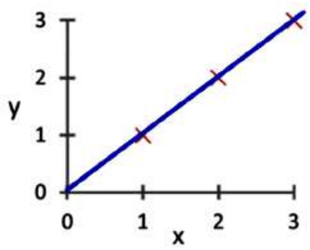
for $m = 1$

- Find $j(m)$ when $m = 1$

$$J(m) = \frac{1}{2n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$$

where: $\hat{y}_i = mx_i$

Visualize Cost function in 2D

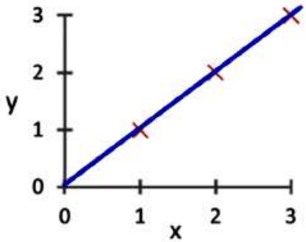


Find $j(m)$ when $m = 1$

$$J(m) = \frac{1}{2n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$$

where: $\hat{y}_i = mx_i$

Visualize Cost function in 2D



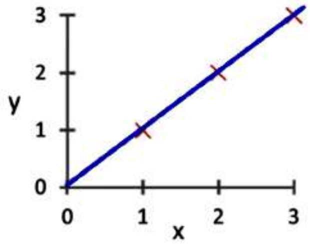
Find $j(m)$ when $m = 1$

$$J(m) = \frac{1}{2n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$$

where: $\hat{y}_i = mx_i$

- **when** $m = 1$;

Visualize Cost function in 2D



Find $j(m)$ when $m = 1$

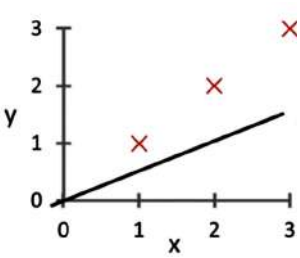
$$J(m) = \frac{1}{2n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$$

where: $\hat{y}_i = mx_i$

- **when** $m = 1$;

$$J(1) = \frac{1}{2n}(0^2 + 0^2 + 0^2) = 0$$

Visualize Cost function in 2D



Find $j(m)$ when $m = 1$

$$J(m) = \frac{1}{2n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$$

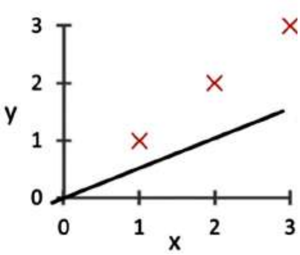
where: $\hat{y}_i = mx_i$

- **when** $m = 1$;

$$J(1) = \frac{1}{2n}(0^2 + 0^2 + 0^2) = 0$$

- **when** $m = 0.5$;

Visualize Cost function in 2D



Find $j(m)$ when $m = 1$

$$J(m) = \frac{1}{2n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$$

where: $\hat{y}_i = mx_i$

- **when** $m = 1$;

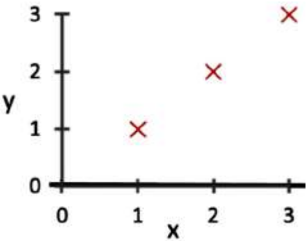
$$J(1) = \frac{1}{2n}(0^2 + 0^2 + 0^2) = 0$$

- **when** $m = 0.5$;

$$J(0.5) =$$

$$J(0.5) = \frac{1}{2n}(0.5^2 + 1^2 + 1.5^2) = 0.583$$

Visualize Cost function in 2D



Find $j(m)$ when $m = 1$

$$J(m) = \frac{1}{2n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$$

where: $\hat{y}_i = mx_i$

- **when** $m = 1$;

$$J(1) = \frac{1}{2n}(0^2 + 0^2 + 0^2) = 0$$

- **when** $m = 0.5$;

$$J(0.5) =$$

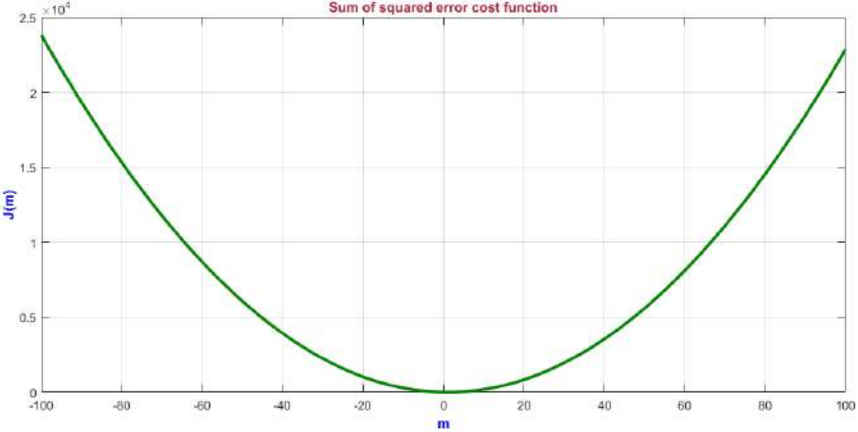
$$J(0.5) = \frac{1}{2n}(0.5^2 + 1^2 + 1.5^2) = 0.583$$

- **when** $m = 0$;

$$J(0) = 2.3$$

Visualize Cost function in 2D

$J(m)$, its a function of parameter m.



Note [3]

[3] Matlab code available

Visualize Cost function in 3D

- Hypothesis:
  $h_x = mx + c$
- Parameters:
  m and c
- Cost function:

$$J(m,c) = \frac{1}{2n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$$

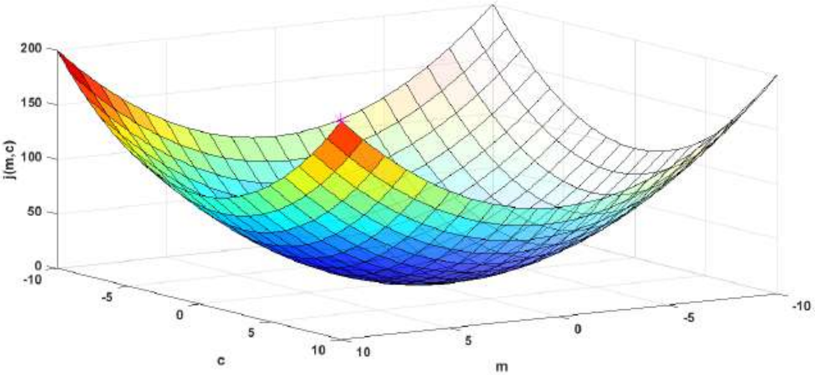
where:
$\hat{y}_i = mx + c$

- Goal:

$$\underset{m,c}{\text{argmin}}\, J(m,c)$$

- It requires 3D plot to visualize cost function J with two parameters (m and c) and predicted value. By keeping both the parameters m and c, we are considering all set of solutions / lines, whether or not they pass from origin (unlike previously).

Visualize Cost function in 3D



- This is the visualization and intuition of cost function, now we need to have an algorithm that automatically finds hypothesis parameters m and c that minimizes $J(m, c)$.

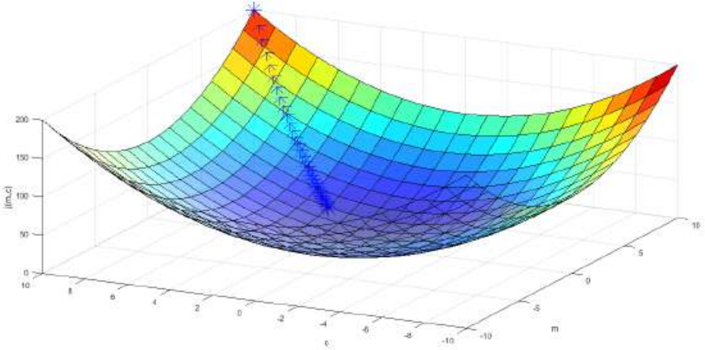
Rem: Cost function for linear regression (SSE) will be a bowl-shaped / convex function. So there is no local minima / optimum, except for one global minima / optimum.

Note [4]

[4] Matlab code available

Gradient Descent (GD) for Minimizing "J"

- Gradient descent is an optimization algorithm (not specific to linear regression) that finds the optimal weights (w_s , i.e. m and c) that reduces prediction error[5] .
- It can optimize weights for any general cost function:
  $\text{argmin}_{w_1,w_2,\ldots,w_n} J(w_1, w_2, \ldots, w_n)$



[5] Matlab code available

Gradient Descent for Minimizing "J"

Algorithm 1 Gradient Descent Algorithm

Input:

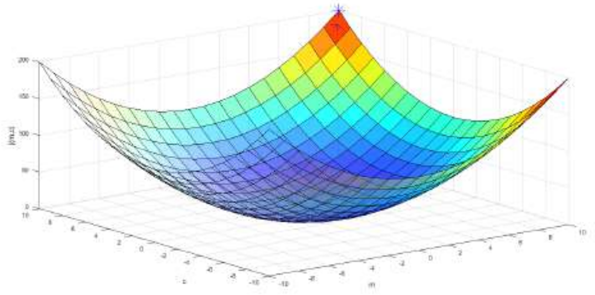
$$J(w_i, w_j)$$

Output:

$$\underset{w_i, w_j}{\operatorname{argmin}} J(w_i, w_j)$$

1: Initialize weights w_s (w_i, w_j), with random values and calculate Error SSE.
2: Calculate gradient i.e. change in SSE when the weights w_s are changed by a very small value from their original randomly initialized value. This helps move the values of w_s in the direction in which SSE is minimized.
3: Adjust weights w_s with the gradients to reach the optimal values where SSE is minimized.
4: Use new weights w_s for prediction and to calculate the new SSE.
5: Repeat steps 2 and 3 till further adjustments to w_s doesn't significantly reduce the Error / convergence.

Gradient Descent in action



repeat until convergence {

$$w_i := w_i - \alpha \frac{\partial}{\partial w_i} J(w_i, w_j) \; \} \quad (10)$$

- $:=$ is "assignment" operator
- α (positive number) is learning rate
- run for w_i & w_j and update weights simultaneously (simultaneous update)

∂ term[6]

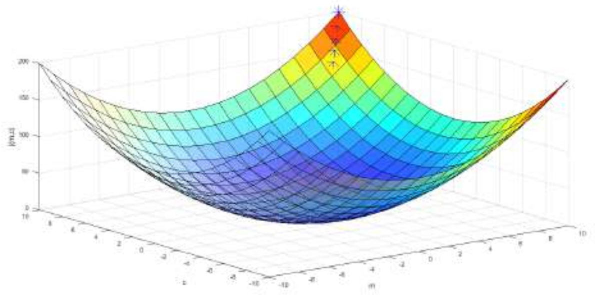
Simultaneous update[7]

---

[6] This slide provides just an intuition of GD algorithm. I will explain this ∂ term in couple of slides

[7] Explained on next slide

Gradient Descent in action



repeat until convergence {

$$w_i := w_i - \alpha \frac{\partial}{\partial w_i} J(w_i, w_j) \; \} \qquad (10)$$

- $:=$ is "assignment" operator
- α (positive number) is learning rate
- run for w_i & w_j and update weights simultaneously (simultaneous update)

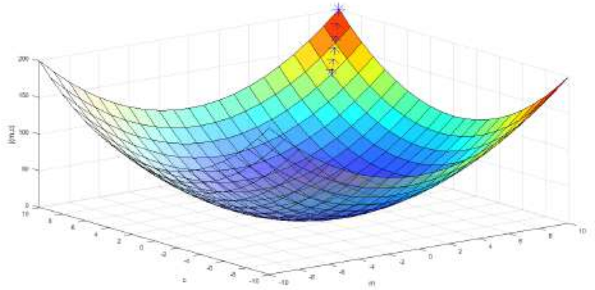
∂ term[6]

Simultaneous update[7]

[6] This slide provides just an intuition of GD algorithm. I will explain this ∂ term in couple of slides

[7] Explained on next slide

Gradient Descent in action



repeat until convergence {

$$w_i := w_i - \alpha \frac{\partial}{\partial w_i} J(w_i, w_j) \; \} \quad (10)$$

- $:=$ is "assignment" operator
- α (positive number) is learning rate
- run for w_i & w_j and update weights simultaneously (simultaneous update)

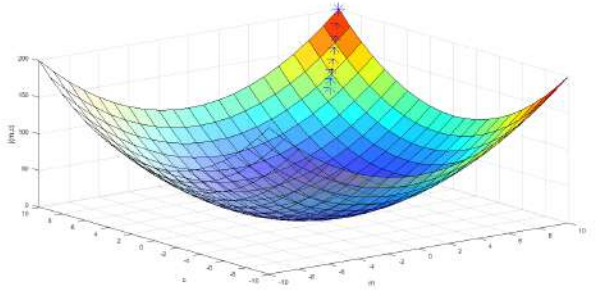
∂ term[6]

Simultaneous update[7]

[6] This slide provides just an intuition of GD algorithm. I will explain this ∂ term in couple of slides

[7] Explained on next slide

Gradient Descent in action



repeat until convergence {

$$w_i := w_i - \alpha \frac{\partial}{\partial w_i} J(w_i, w_j) \; \} \quad (10)$$

- := is "assignment" operator
- α (positive number) is learning rate
- run for w_i & w_j and update weights simultaneously (simultaneous update)

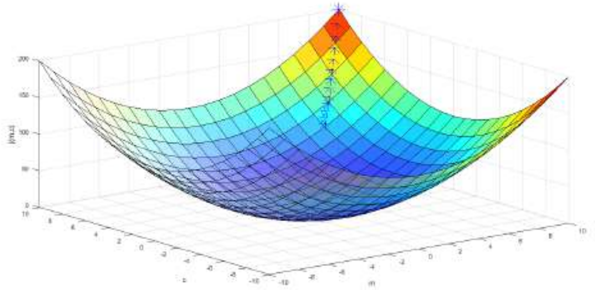
∂ term[6]

Simultaneous update[7]

[6] This slide provides just an intuition of GD algorithm. I will explain this ∂ term in couple of slides

[7] Explained on next slide

Gradient Descent in action



repeat until convergence {

$$w_i := w_i - \alpha \frac{\partial}{\partial w_i} J(w_i, w_j) \; \} \qquad (10)$$

- $:=$ is "assignment" operator
- α (positive number) is learning rate
- run for w_i & w_j and update weights simultaneously (simultaneous update)

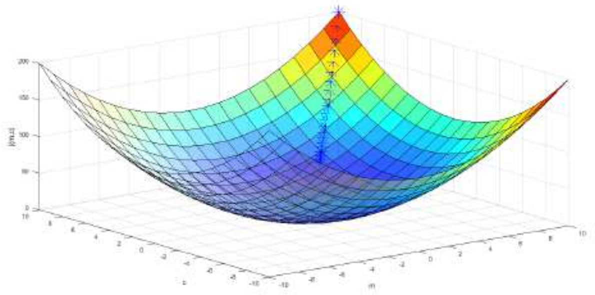
∂ term[6]

Simultaneous update[7]

[6] This slide provides just an intuition of GD algorithm. I will explain this ∂ term in couple of slides

[7] Explained on next slide

Gradient Descent in action



repeat until convergence {

$$w_i := w_i - \alpha \frac{\partial}{\partial w_i} J(w_i, w_j) \; \} \quad (10)$$

- $:=$ is "assignment" operator
- α (positive number) is learning rate
- run for w_i & w_j and update weights simultaneously (simultaneous update)

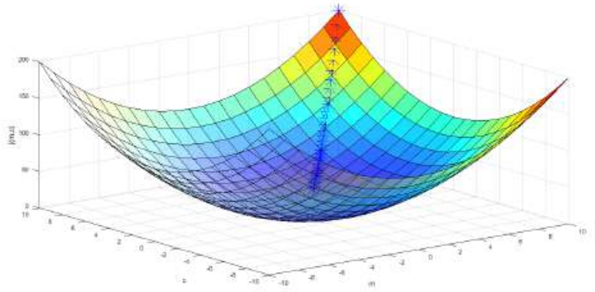
∂ term[6]

Simultaneous update[7]

[6] This slide provides just an intuition of GD algorithm. I will explain this ∂ term in couple of slides

[7] Explained on next slide

Gradient Descent in action



repeat until convergence {

$$w_i := w_i - \alpha \frac{\partial}{\partial w_i} J(w_i, w_j) \; \} \quad (10)$$

- $:=$ is "assignment" operator
- α (positive number) is learning rate
- run for w_i & w_j and update weights simultaneously (simultaneous update)

∂ term[6]

Simultaneous update[7]

[6] This slide provides just an intuition of GD algorithm. I will explain this ∂ term in couple of slides

[7] Explained on next slide

Simultaneous update

$$temp0 := w_i - \alpha \frac{\partial}{\partial w_i} J(w_i, w_j)$$

$$temp1 := w_j - \alpha \frac{\partial}{\partial w_j} J(w_i, w_j)$$

$$w_i := temp0$$

$$w_j := temp1$$

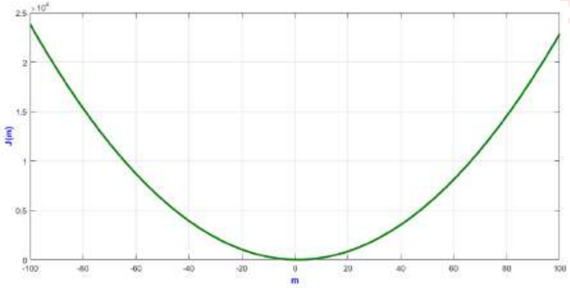
- w_i & w_j to be updated together at the end of iteration, otherwise one weight will be updated earlier and within same iteration updated weight will be used for the calculation of other weight.

Gradient Descent with one parameter (intuition of ∂ term)

repeat until convergence {

$$m := m - \alpha \frac{\partial}{\partial m} J(m) \; \} \tag{11}$$

where α is learning rate.

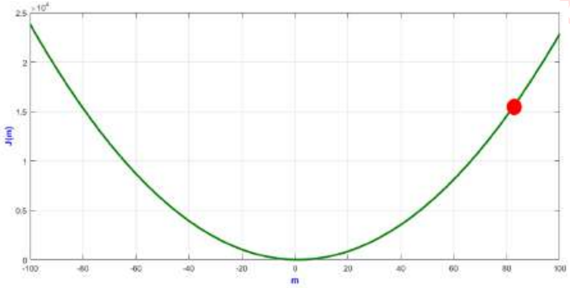


Gradient Descent with one parameter (intuition of ∂ term)

repeat until convergence {

$$m := m - \alpha \frac{\partial}{\partial m} J(m) \; \} \qquad (11)$$

where α is learning rate.



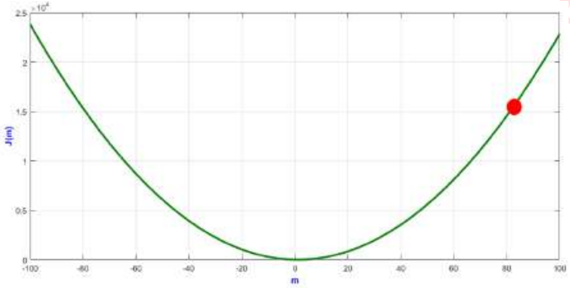
- Suppose we initialize m with an random point.

Gradient Descent with one parameter (intuition of ∂ term)

repeat until convergence {

$$m := m - \alpha \frac{\partial}{\partial m} J(m) \; \} \tag{11}$$

where α is learning rate.



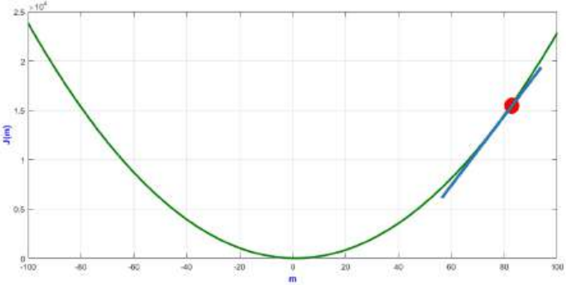
- Suppose we initialize m with an random point.
- We need to find derivative $\frac{\partial}{\partial m} J(m)$, which is a tangent at a given point and provides value of its slope.

Gradient Descent with one parameter (intuition of ∂ term)

repeat until convergence {

$$m := m - \alpha \frac{\partial}{\partial m} J(m) \; \} \tag{11}$$

where α is learning rate.



- Suppose we initialize m with an random point.
- We need to find derivative $\frac{\partial}{\partial m} J(m)$, which is a tangent at a given point and provides value of its slope.
- As slope is $+ve$,

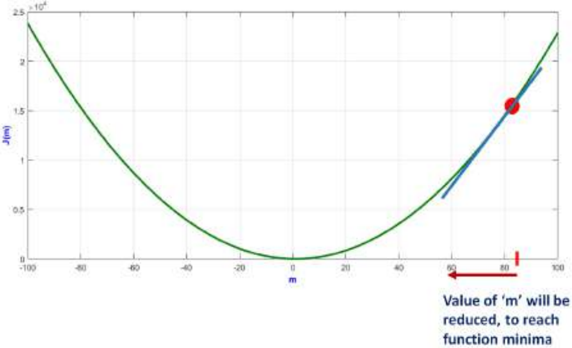
$$m = m - \alpha(+ve\ numb)$$

Gradient Descent with one parameter (intuition of ∂ term)

repeat until convergence {

$$m := m - \alpha \frac{\partial}{\partial m} J(m) \; \} \tag{11}$$

where α is learning rate.



- Suppose we initialize m with an random point.
- We need to find derivative $\frac{\partial}{\partial m} J(m)$, which is a tangent at a given point and provides value of its slope.
- As slope is $+ve$,

$$m = m - \alpha(+ve\ numb)$$

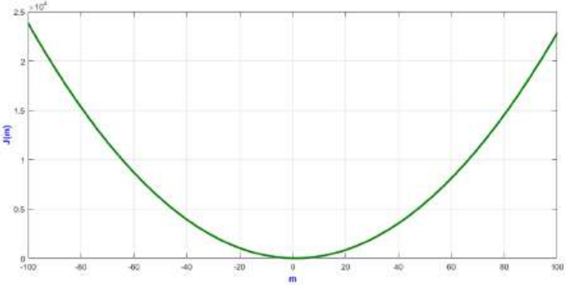
- Finally, updated value of m will be reduced and will move towards function minima.

Gradient Descent with one parameter (intuition of ∂ term)

repeat until convergence {

$$m := m - \alpha \frac{\partial}{\partial m} J(m) \; \} \tag{12}$$

where α is learning rate.

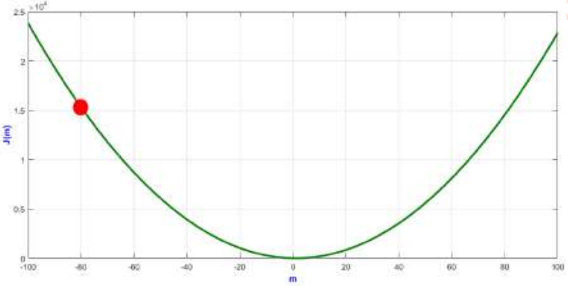


Gradient Descent with one parameter (intuition of ∂ term)

repeat until convergence {

$$m := m - \alpha \frac{\partial}{\partial m} J(m) \; \} \tag{12}$$

where α is learning rate.



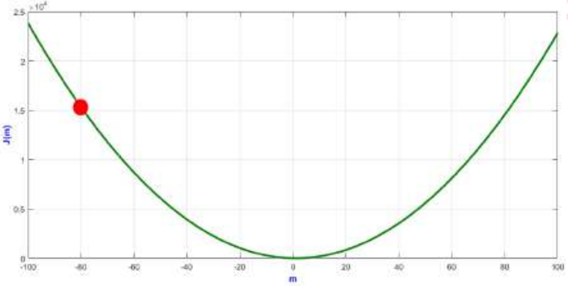
- Suppose we initialize m with an random point.

Gradient Descent with one parameter (intuition of ∂ term)

repeat until convergence {

$$m := m - \alpha \frac{\partial}{\partial m} J(m) \; \} \tag{12}$$

where α is learning rate.



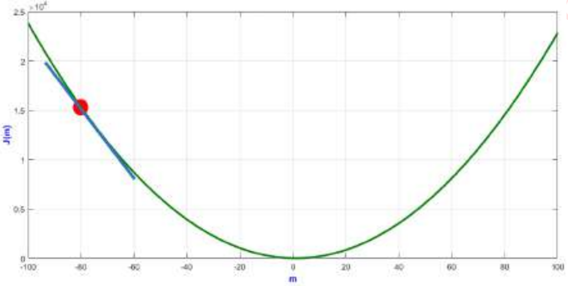
- Suppose we initialize m with an random point.
- We need to find derivative $\frac{\partial}{\partial m} J(m)$, which is a tangent at a given point and provides value of its slope.

Gradient Descent with one parameter (intuition of ∂ term)

repeat until convergence {

$$m := m - \alpha \frac{\partial}{\partial m} J(m) \text{ } \} \tag{12}$$

where α is learning rate.



- Suppose we initialize m with an random point.
- We need to find derivative $\frac{\partial}{\partial m} J(m)$, which is a tangent at a given point and provides value of its slope.
- As slope is $-ve$,

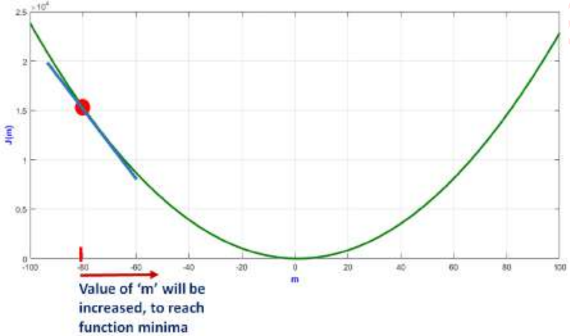
$$m = m - \alpha(-ve\ numb)$$

Gradient Descent with one parameter (intuition of ∂ term)

repeat until convergence {

$$m := m - \alpha \frac{\partial}{\partial m} J(m) \; \} \tag{12}$$

where α is learning rate.



- Suppose we initialize m with an random point.
- We need to find derivative $\frac{\partial}{\partial m} J(m)$, which is a tangent at a given point and provides value of its slope.
- As slope is $-ve$,

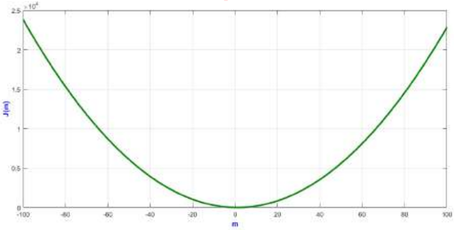
$$m = m - \alpha(-ve\ numb)$$

- Finally, updated value of m, $m + \alpha(numb)$ will be added and will move towards function minima.

Learning Rate α

$$w_i := w_i - \alpha \frac{\partial}{\partial w_i} J(w_i)$$

if α is too small, GD can be slow

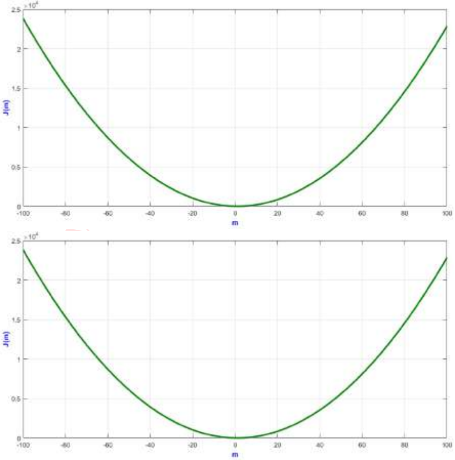


Learning Rate α

$$w_i := w_i - \alpha \frac{\partial}{\partial w_i} J(w_i)$$

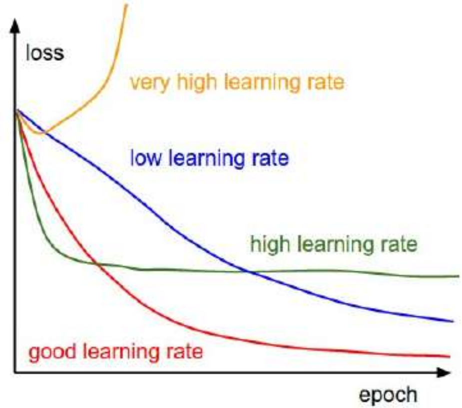
if α is too small, GD can be slow

if α is too large, GD can overshoot the minimum. It may fail to converge.



- Reading assignment: Problem of vanishing gradients

Learning Rate α



Note: [8]

With low learning rates the improvements will be linear (blue line). With high learning rates they will start to look more exponential. Higher learning rates will decay the loss faster, but they get stuck at worse values of loss (green line). This is because there is too much "energy" in the optimization and the parameters are bouncing around chaotically, unable to settle in a nice spot in the optimization landscape.

[8] Image from Stanford's course on CNN `http://cs231n.stanford.edu/`

Section Contents

Regression with Gradient Descent

Putting together GD and cost function to perform regression.

Linear Regression Model:

- hypothesis: $h = mx + c$
- Cost function:

$$J(m,c) = \frac{1}{2n} \operatorname*{argmin}_{m,c} \sum_{i=1}^{n} (\hat{y}_i - y_i)^2$$

where $\hat{y}_i = mx_i + c$ and $J(m,c)$ is cost / loss function.

Regression with Gradient Descent

Putting together GD and cost function to perform regression.

Linear Regression Model:

- hypothesis: $h = mx + c$
- Cost function:

$$J(m,c) = \frac{1}{2n} \operatorname*{argmin}_{m,c} \sum_{i=1}^{n} (\hat{y}_i - y_i)^2$$

where $\hat{y}_i = mx_i + c$ and $J(m,c)$ is cost / loss function.

Gradient Descent Algorithm:

repeat until convergence {

$$w_i := w_i - \alpha \frac{\partial}{\partial w_i} J(w_1, w_2) \; \}$$

(for i=1 and i=2)

Regression with Gradient Descent

Putting together GD and cost function to perform regression.

Linear Regression Model:

- hypothesis: $h = mx + c$
- Cost function:

$$J(m,c) = \frac{1}{2n} \underset{m,c}{\operatorname{argmin}} \sum_{i=1}^{n} (\hat{y}_i - y_i)^2$$

where $\hat{y}_i = mx_i + c$ and $J(m,c)$ is cost / loss function.

Gradient Descent Algorithm:

repeat until convergence {

$$w_i := w_i - \alpha \frac{\partial}{\partial w_i} J(w_1, w_2) \; \}$$

(for i=1 and i=2)

Linear Regression with GD

Apply gradient descent algorithm to minimize cost function J. $\operatorname{argmin}_{m,c} J(m,c)$

Regression with Gradient Descent

Solving for: $\frac{\partial}{\partial w_i} J(w_1, w_2)$ and (i=1 and i=2):

$$\begin{aligned}\frac{\partial}{\partial w_i} J(w_i, w_j) &= \frac{\partial}{\partial w_i} \frac{1}{2n} \sum_{i=1}^{n} (\hat{y}_i - y_i)^2 \\ &= \frac{\partial}{\partial w_i} \frac{1}{2n} \sum_{i=1}^{n} ((m.x_i + c) - y_i)^2\end{aligned} \tag{13}$$

There are two cases (in our scenario $i = 1 : w_1$ or c and $i = 2 : w_2$ or m):

Regression with Gradient Descent

Solving for: $\frac{\partial}{\partial w_i} J(w_1, w_2)$ and (i=1 and i=2):

$$\begin{aligned}\frac{\partial}{\partial w_i} J(w_i, w_j) &= \frac{\partial}{\partial w_i} \frac{1}{2n} \sum_{i=1}^{n} (\hat{y}_i - y_i)^2 \\ &= \frac{\partial}{\partial w_i} \frac{1}{2n} \sum_{i=1}^{n} ((m.x_i + c) - y_i)^2\end{aligned} \tag{13}$$

There are two cases (in our scenario $i = 1 : w_1$ or c and $i = 2 : w_2$ or m):

1. w_1 = c, $\frac{\partial}{\partial c} J(m, c)$:

$$\frac{\partial}{\partial c} J(m, c) = \frac{1}{n} \sum_{i=1}^{n} (\hat{y}_i - y_i) \tag{14}$$

Regression with Gradient Descent

Solving for: $\frac{\partial}{\partial w_i} J(w_1, w_2)$ and (i=1 and i=2):

$$\begin{aligned} \frac{\partial}{\partial w_i} J(w_i, w_j) &= \frac{\partial}{\partial w_i} \frac{1}{2n} \sum_{i=1}^{n} (\hat{y}_i - y_i)^2 \\ &= \frac{\partial}{\partial w_i} \frac{1}{2n} \sum_{i=1}^{n} ((m.x_i + c) - y_i)^2 \end{aligned} \tag{13}$$

There are two cases (in our scenario $i = 1 : w_1$ or c and $i = 2 : w_2$ or m):

1. w_1 = c, $\frac{\partial}{\partial c} J(m, c)$:

$$\frac{\partial}{\partial c} J(m, c) = \frac{1}{n} \sum_{i=1}^{n} (\hat{y}_i - y_i) \tag{14}$$

2. w_2 = m, $\frac{\partial}{\partial m} J(m, c)$:

$$\frac{\partial}{\partial m} J(m, c) = \frac{1}{n} \sum_{i=1}^{n} (\hat{y}_i - y_i) \cdot x_i \tag{15}$$

Regression with Gradient Descent

Convergence

Plug back equations 14 and 15 into gradient descent algorithm

repeat until convergence {

$$w_i := w_i - \alpha \frac{\partial}{\partial w_i} J(w_i, w_j) \;\}$$

repeat until convergence {

$$c := c - \alpha \frac{1}{n} \sum_{i=1}^{n} (\hat{y}_i - y_i)$$

$$m := m - \alpha \frac{1}{n} \sum_{i=1}^{n} (\hat{y}_i - y_i) \cdot x_i \tag{16}$$

}

Regression (multivariate / multi-variables) with Gradient Descent

Dataset:

Experience (Yrs) x^1	Completed Projects x^2	MOOC x^3	Last Salary x^4	Salary **y**
1	2	2	25k	32k
1.3	2	3	30k	33k
1.8	3	3	40k	43k
2	2	2	41k	49k
3.3	4	2	55k	68k
..	..	..	..	..

where
x^d = feature at d^{th} dimension
x_i = i^{th} training example
x_i^d = feature value at d^{th} dimension for i^{th} training example

Regression (multivariate / multi-variables) with Gradient Descent

Dataset:

Experience (Yrs) x^1	**Completed Projects** x^2	**MOOC** x^3	**Last Salary** x^4	**Salary** **y**
1	2	2	25k	32k
1.3	2	3	30k	33k
1.8	3	3	40k	43k
2	2	2	41k	49k
3.3	4	2	55k	68k
..	..	..	..	..

where
x^d = feature at d^{th} dimension
x_i = i^{th} training example
x_i^d = feature value at d^{th} dimension for i^{th} training example

Previously: hypothesis: $h = c + mx$ or $h = \theta_0 + \theta_1 x^1$

Regression (multivariate / multi-variables) with Gradient Descent

Dataset:

Experience (Yrs) x^1	**Completed Projects** x^2	**MOOC** x^3	**Last Salary** x^4	**Salary** **y**
1	2	2	25k	32k
1.3	2	3	30k	33k
1.8	3	3	40k	43k
2	2	2	41k	49k
3.3	4	2	55k	68k
..	..	..	..	..

where
x^d = feature at d^{th} dimension
x_i = i^{th} training example
x_i^d = feature value at d^{th} dimension for i^{th} training example

Previously: hypothesis: $h = c + mx$ or $h = \theta_0 + \theta_1 x^1$

And Now ?

Regression (multivariate / multi-variables) with Gradient Descent

Hypothesis for multi-variables:

$$h = \theta_0 + \theta_1 x^1 + \theta_2 x^2 + \theta_3 x^3 + \cdots + \theta_d x^d \tag{17}$$

- Parameters of the model: $\theta_0, \theta_1, \cdots, \theta_d$
- Cost function:

$$J(\theta_0, \theta_1, \cdots, \theta_d) = \frac{1}{2n} \underset{\theta_0, \theta_1, \cdots, \theta_d}{\operatorname{argmin}} \sum_{i=1}^{n} (\hat{y}_i - y_i)^2$$

Gradient Descent:

Regression (multivariate / multi-variables) with Gradient Descent

Hypothesis for multi-variables:

$$h = \theta_0 + \theta_1 x^1 + \theta_2 x^2 + \theta_3 x^3 + \cdots + \theta_d x^d \tag{17}$$

- Parameters of the model: $\theta_0, \theta_1, \cdots, \theta_d$
- Cost function:

$$J(\theta_0, \theta_1, \cdots, \theta_d) = \frac{1}{2n} \underset{\theta_0, \theta_1, \cdots, \theta_d}{\text{argmin}} \sum_{i=1}^{n} (\hat{y}_i - y_i)^2$$

Gradient Descent:
repeat until convergence {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1, \cdots, \theta_d)$$

} (simultaneously update for every $j = 0, 1, \cdots, d$)

Regression (multivariate / multi-variables) with Gradient Descent

Previously when $d = 1$

repeat until convergence {

$$\theta_0 := \theta_0 - \alpha \frac{1}{n} \sum_{i=1}^{n} (\hat{y}_i - y_i)$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{n} \sum_{i=1}^{n} (\hat{y}_i - y_i) \cdot x_i^1$$

}

Regression (multivariate / multi-variables) with Gradient Descent

Generally (GD algorithm) for any given d dimensional vector ($d \geq 1$)

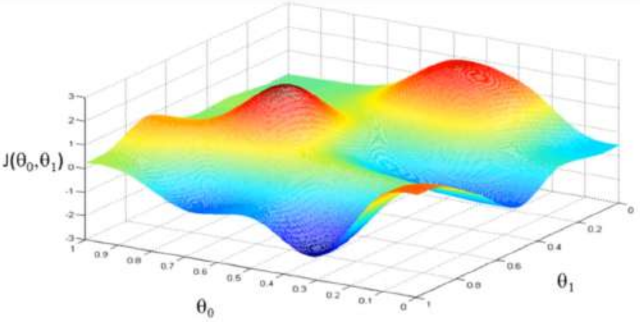
Regression (multivariate / multi-variables) with Gradient Descent

Generally (GD algorithm) for any given d dimensional vector ($d \geq 1$)

$$\text{repeat until convergence } \{ \; \theta_0 := \theta_0 - \alpha \frac{1}{n} \sum_{i=1}^{n} (\hat{y}_i - y_i)$$
$$\theta_1 := \theta_1 - \alpha \frac{1}{n} \sum_{i=1}^{n} (\hat{y}_i - y_i) \cdot x_i^1 \tag{18}$$
$$\theta_2 := \theta_2 - \alpha \frac{1}{n} \sum_{i=1}^{n} (\hat{y}_i - y_i) \cdot x_i^2$$
$$\ldots \}$$

Gradient Descent can stuck in local minima

Gradient Descent algorithm can get stuck in local minima[9].



repeat until convergence {

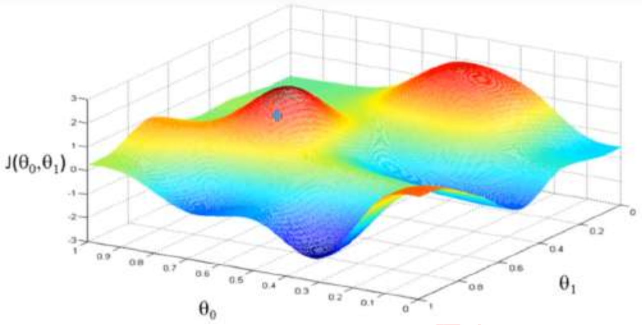
$$w_i := w_i - \alpha \frac{\partial}{\partial w_i} J(w_i, w_j) \}$$

Rem: Cost function for linear regression (SSE) will be a bowl-shaped / convex function. So there is no local minima / optimum, except for one global minima / optimum.

[9] slide from Andrew Ng

Gradient Descent can stuck in local minima

Gradient Descent algorithm can get stuck in local minima[9].



repeat until convergence {

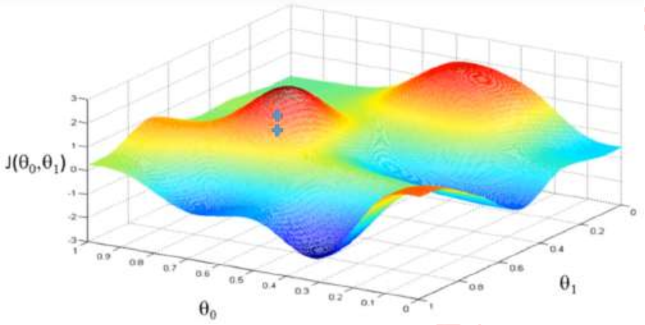
$$w_i := w_i - \alpha \frac{\partial}{\partial w_i} J(w_i, w_j) \;\}$$

Rem: Cost function for linear regression (SSE) will be a bowl-shaped / convex function. So there is no local minima / optimum, except for one global minima / optimum.

[9] slide from Andrew Ng

Gradient Descent can stuck in local minima

Gradient Descent algorithm can get stuck in local minima[9].



repeat until convergence {

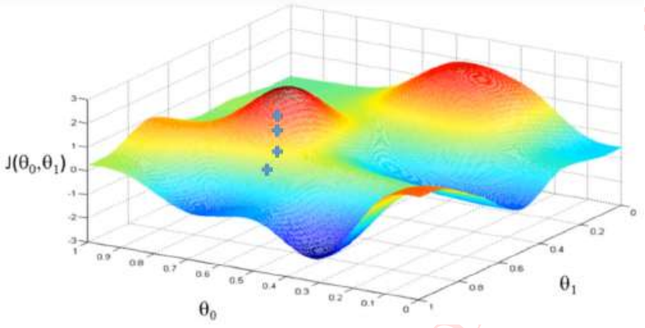
$$w_i := w_i - \alpha \frac{\partial}{\partial w_i} J(w_i, w_j) \; \}$$

Rem: Cost function for linear regression (SSE) will be a bowl-shaped / convex function. So there is no local minima / optimum, except for one global minima / optimum.

[9] slide from Andrew Ng

Gradient Descent can stuck in local minima

Gradient Descent algorithm can get stuck in local minima[9].



repeat until convergence {

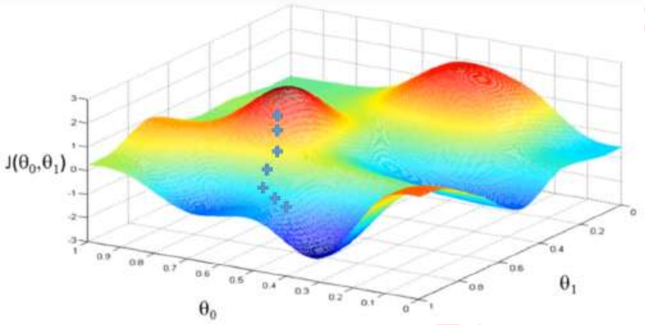
$$w_i := w_i - \alpha \frac{\partial}{\partial w_i} J(w_i, w_j) \ \}$$

Rem: Cost function for linear regression (SSE) will be a bowl-shaped / convex function. So there is no local minima / optimum, except for one global minima / optimum.

[9] slide from Andrew Ng

Gradient Descent can stuck in local minima

Gradient Descent algorithm can get stuck in local minima[9].



repeat until convergence {

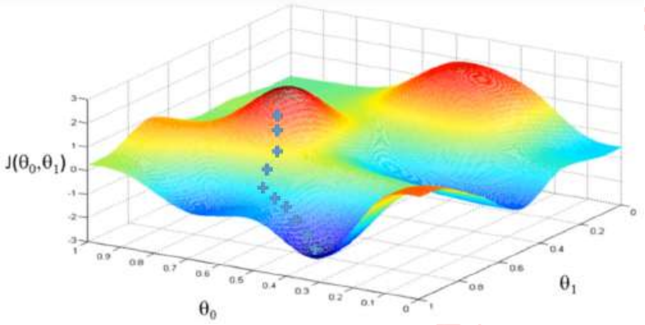
$$w_i := w_i - \alpha \frac{\partial}{\partial w_i} J(w_i, w_j) \; \}$$

Rem: Cost function for linear regression (SSE) will be a bowl-shaped / convex function. So there is no local minima / optimum, except for one global minima / optimum.

[9] slide from Andrew Ng

Gradient Descent can stuck in local minima

Gradient Descent algorithm can get stuck in local minima[9].



$$\text{repeat until convergence } \{$$

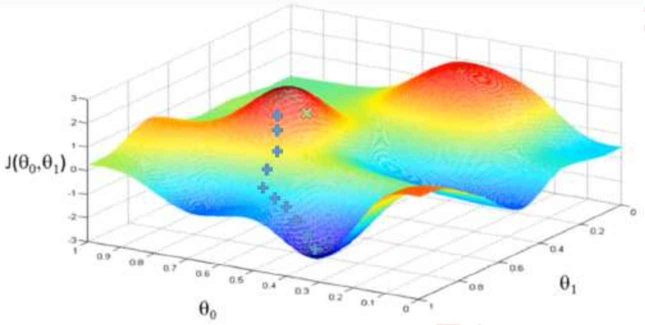
$$w_i := w_i - \alpha \frac{\partial}{\partial w_i} J(w_i, w_j) \; \}$$

Rem: Cost function for linear regression (SSE) will be a bowl-shaped / convex function. So there is no local minima / optimum, except for one global minima / optimum.

[9] slide from Andrew Ng

Gradient Descent can stuck in local minima

Gradient Descent algorithm can get stuck in local minima[9].



$$\text{repeat until convergence } \{$$

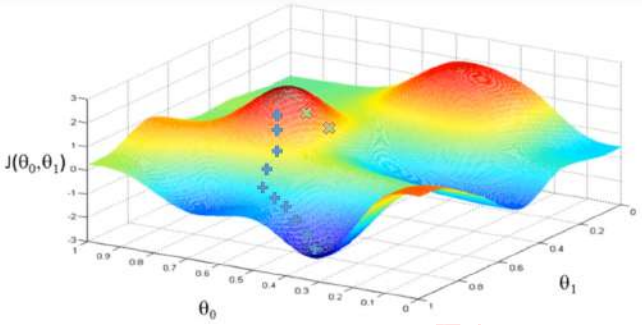
$$w_i := w_i - \alpha \frac{\partial}{\partial w_i} J(w_i, w_j) \ \}$$

Rem: Cost function for linear regression (SSE) will be a bowl-shaped / convex function. So there is no local minima / optimum, except for one global minima / optimum.

[9] slide from Andrew Ng

Gradient Descent can stuck in local minima

Gradient Descent algorithm can get stuck in local minima[9].



repeat until convergence {

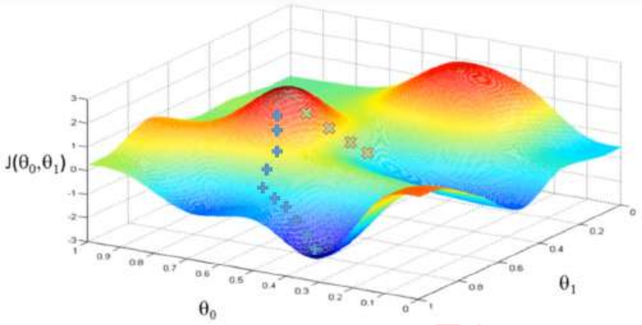
$$w_i := w_i - \alpha \frac{\partial}{\partial w_i} J(w_i, w_j) \;\}$$

Rem: Cost function for linear regression (SSE) will be a bowl-shaped / convex function. So there is no local minima / optimum, except for one global minima / optimum.

[9] slide from Andrew Ng

Gradient Descent can stuck in local minima

Gradient Descent algorithm can get stuck in local minima[9].



repeat until convergence {

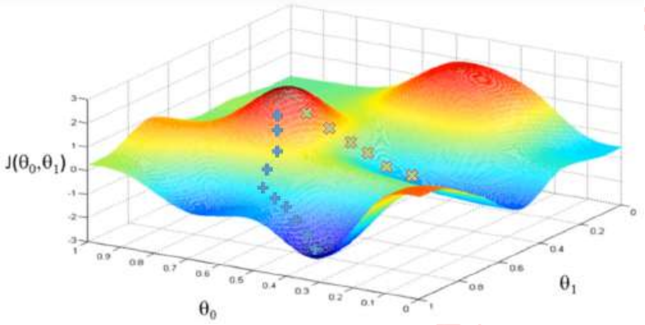
$$w_i := w_i - \alpha \frac{\partial}{\partial w_i} J(w_i, w_j) \; \}$$

Rem: Cost function for linear regression (SSE) will be a bowl-shaped / convex function. So there is no local minima / optimum, except for one global minima / optimum.

[9] slide from Andrew Ng

Gradient Descent can stuck in local minima

Gradient Descent algorithm can get stuck in local minima[9].



repeat until convergence {

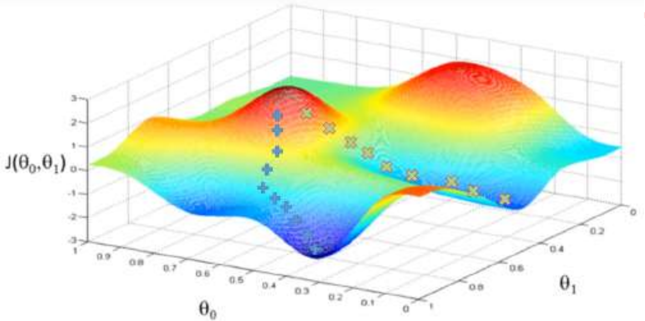
$$w_i := w_i - \alpha \frac{\partial}{\partial w_i} J(w_i, w_j) \; \}$$

Rem: Cost function for linear regression (SSE) will be a bowl-shaped / convex function. So there is no local minima / optimum, except for one global minima / optimum.

[9] slide from Andrew Ng

Gradient Descent can stuck in local minima

Gradient Descent algorithm can get stuck in local minima[9].



repeat until convergence {

$$w_i := w_i - \alpha \frac{\partial}{\partial w_i} J(w_i, w_j) \; \}$$

Rem: Cost function for linear regression (SSE) will be a bowl-shaped / convex function. So there is no local minima / optimum, except for one global minima / optimum.

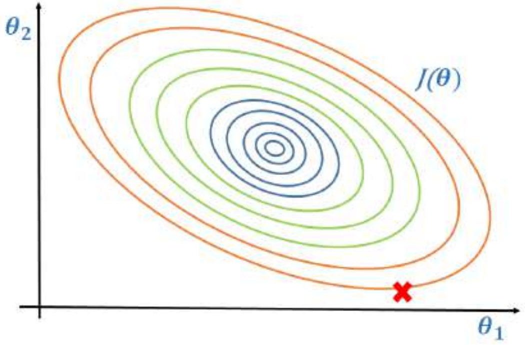
[9] slide from Andrew Ng

Gradient Descent(GD) : Feature Scaling

- GD convergence is effected if features are not scaled.

Gradient Descent(GD) : Feature Scaling

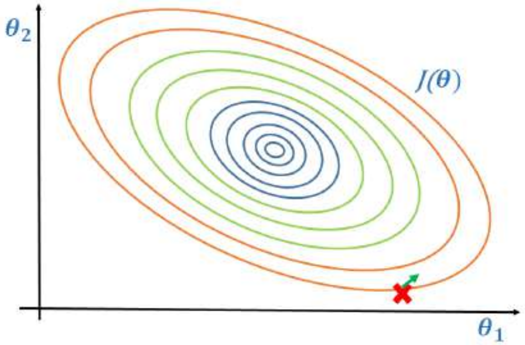
- GD convergence is effected if features are not scaled.



Skewed /elliptical shape (features are not scaled) of contours

Gradient Descent(GD) : Feature Scaling

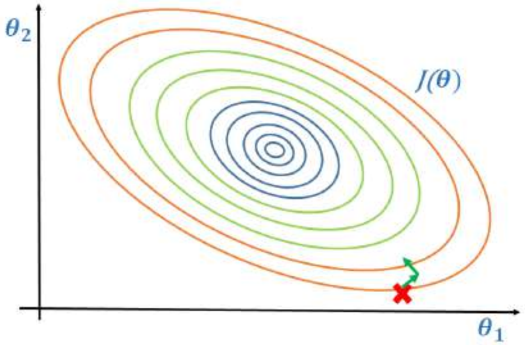
- GD convergence is effected if features are not scaled.



Skewed /elliptical shape (features are not scaled) of contours

Gradient Descent(GD) : Feature Scaling

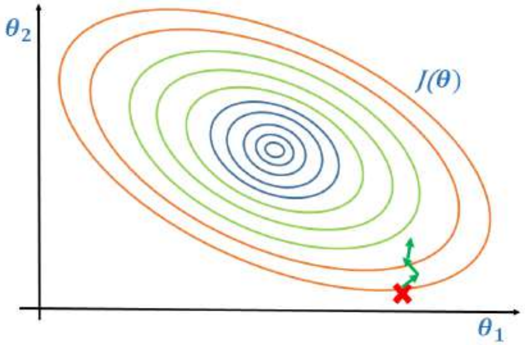
- GD convergence is effected if features are not scaled.



Skewed /elliptical shape (features are not scaled) of contours

Gradient Descent(GD) : Feature Scaling

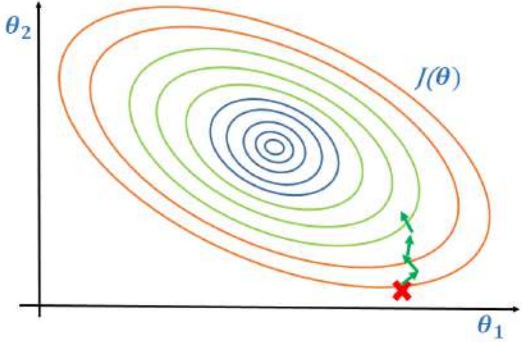
- GD convergence is effected if features are not scaled.



Skewed /elliptical shape (features are not scaled) of contours

Gradient Descent(GD) : Feature Scaling

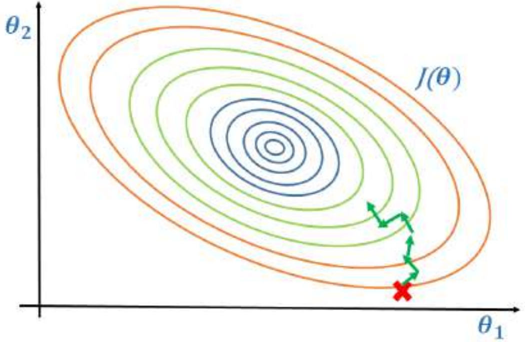
- GD convergence is effected if features are not scaled.



Skewed /elliptical shape (features are not scaled) of contours

Gradient Descent(GD) : Feature Scaling

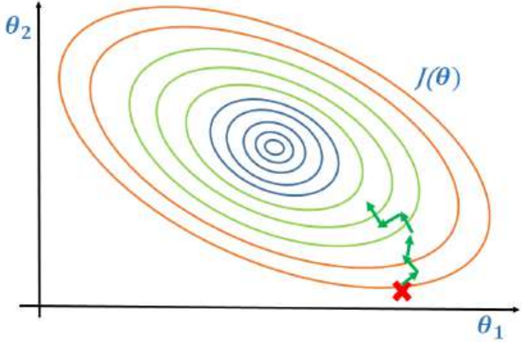
- GD convergence is effected if features are not scaled.

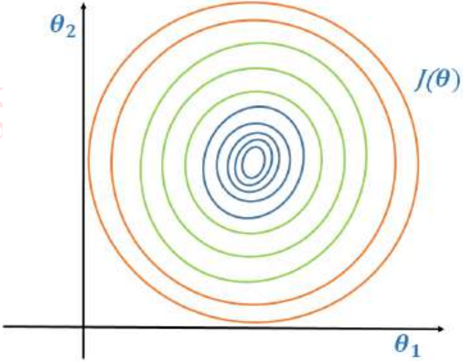


Skewed /elliptical shape (features are not scaled) of contours

Gradient Descent(GD) : Feature Scaling

- GD convergence is effected if features are not scaled.

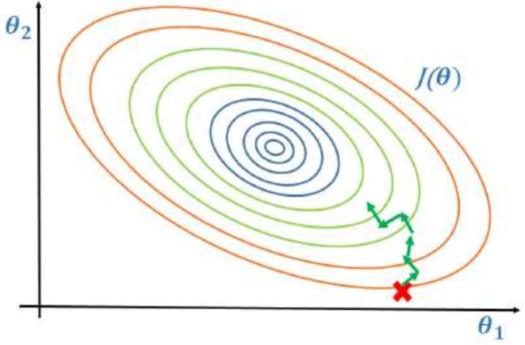




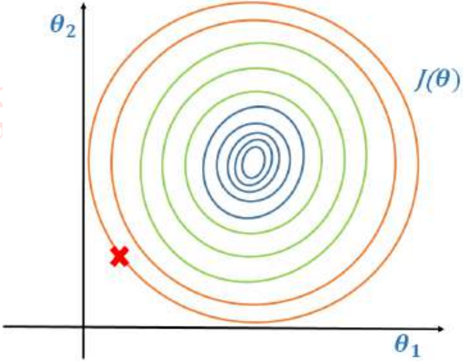
Skewed /elliptical shape (features are not scaled) of contours

Gradient Descent(GD) : Feature Scaling

- GD convergence is effected if features are not scaled.

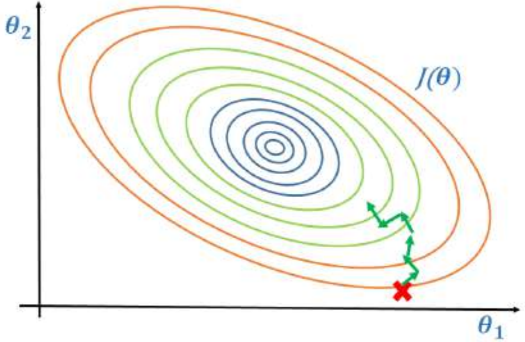


Skewed /elliptical shape (features are not scaled) of contours

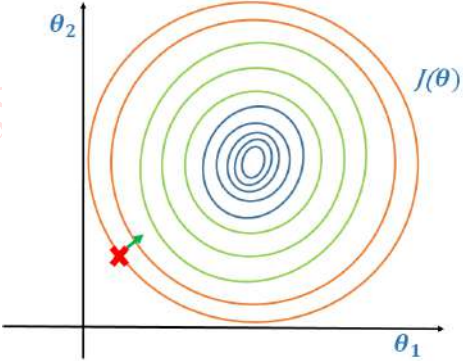


Gradient Descent(GD) : Feature Scaling

- GD convergence is effected if features are not scaled.

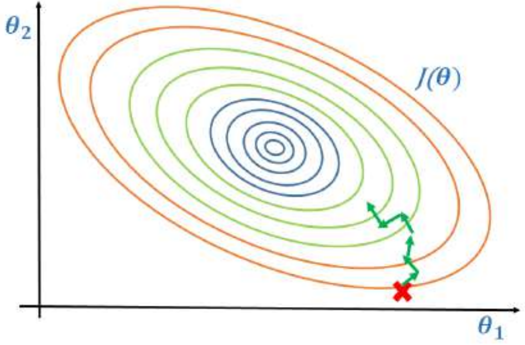


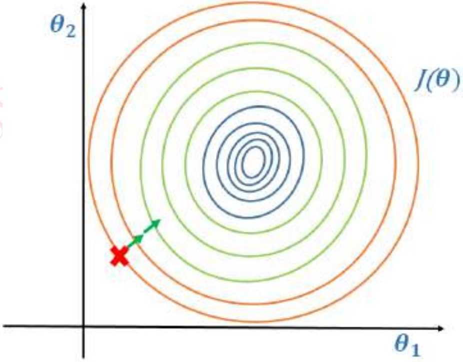
Skewed /elliptical shape (features are not scaled) of contours



Gradient Descent(GD) : Feature Scaling

- GD convergence is effected if features are not scaled.

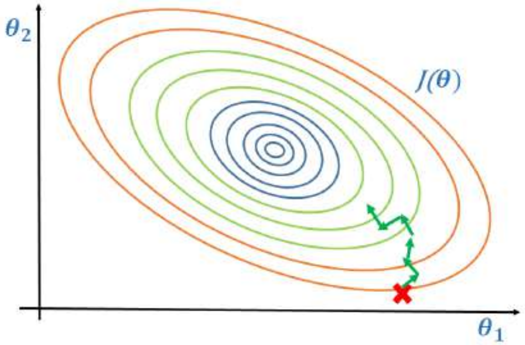




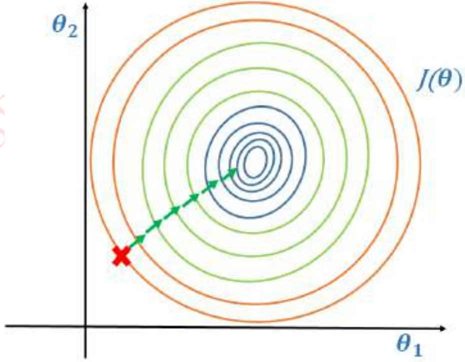
Skewed /elliptical shape (features are not scaled) of contours

Gradient Descent(GD) : Feature Scaling

- GD convergence is effected if features are not scaled.



Skewed /elliptical shape (features are not scaled) of contours



- Better to scale features. You may use Mean Normalization or Standardization scaling method.

Variants of GD

GD algorithm that we have just seen is called Batch Gradient Descent. Most common used GD algorithms are breifly explained below:

1. Batch Gradient Descent is when we sum up over all examples on each iteration when performing the updates to the parameters.

Advantages

1. Fixed learning rate during training.
2. It has straight trajectory towards the minimum and it is guaranteed to converge to global optimum (for convex functions).
3. It has unbiased estimate of gradients.
4. It can benefit from the vectorization

Disadvantages

1. Slow (especially for large datasets), as it goes over all examples.
2. Each step of learning happens after going over all examples (think of outliers in dataset).

Variants of GD

2. Stochastic Gradient Descent (SGD): Instead of going through all examples, Stochastic Gradient Descent (SGD) performs parameters update on each example $< x_i, y_i >$. Therefore, learning happens on every example.

Advantages

1. It is easier to fit into memory (single training sample being processed at a time).
2. It is computationally fast (For larger datasets it can converge faster).
3. Due to frequent updates, the steps taken towards the minima of the loss / cost function have oscillations which can help getting out of local minimums of the loss function.

Disadvantages

1. Due to frequent updates, the steps taken towards the minima are very noisy. This can often lead the gradient descent into sub-optimum directions.
2. It loses the advantage of vectorized operations as it deals with only a single example at a time.

Variants of Gradient Descent

3. Mini-Batch Gradient Descent: This is a mixture of both stochastic and batch gradient descent. The training set is divided into multiple groups called batches. Each batch has a number of training samples in it. For example, assume training set has 100 training examples which is divided into 5 batches with each batch containing 20 training examples.

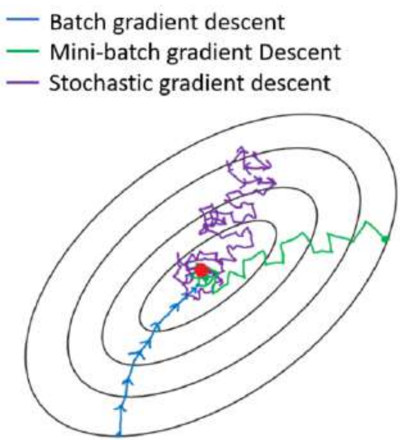
Advantages

1. Easily fits in the memory.
2. It is computationally efficient.
3. Benefit from vectorization.

Disadvantages

1. Due to the noise, the learning steps have more oscillations and requires adding learning-decay to decrease the learning rate as we become closer to the minimum.

Variants of Gradient Descent: Visualization



1. Batch Gradient Descent, slow but unbiased estimate of gradients.
2. Stochastic Gradient Descent (SGD), fast but frequent updates causes noisy steps.
3. Mini-Batch Gradient Descent, computationally efficient but due to the noise the learning steps have more oscillations.

Inductive Bias of Linear Regression

Inductive Bias

The relationship between the attributes x and the output y is linear. The goal is to minimize the sum of squared errors.

Section Contents

Salary prediction: Python

```
#@author: rizwan.khan
import matplotlib.pyplot as plt
import pandas as pd

# Dataset import
dataset=pd.read_csv('data.csv')
X=dataset.iloc[:,:-1].values #data.iloc[:,-1] # last column of data frame
y=dataset.iloc[:,1].values

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size = 1/4)

# import linear regression and fitting it to test data
from sklearn.linear_model import LinearRegression
Regressor=LinearRegression()
Regressor.fit(X_train,y_train)

# predicting trained model on test set
y_pred = Regressor.predict(X_test)
```

Salary prediction: Python

```
# Visualizaing Train set
plt.scatter(X_train,y_train, color = 'red')
plt.plot(X_train, Regressor.predict(X_train), color='blue')
plt.title('Training Set: Exp. Vs Salary')
plt.xlabel('Experience')
plt.ylabel('Salary')
plt.show



# Visualizaing Test set
plt.figure()
plt.scatter(X_test,y_test, color = 'red')
plt.plot(X_train, Regressor.predict(X_train), color='blue')
plt.title('Test Set: Exp. Vs Salary')
plt.xlabel('Experience')
plt.ylabel('Salary')
plt.show
```

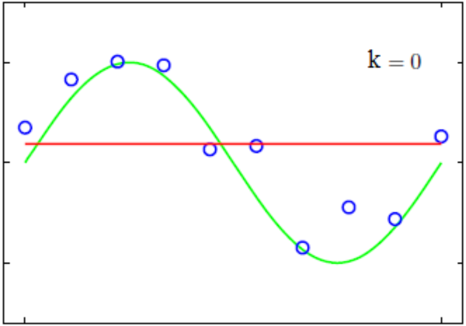
Visualization: Test Set

Visualization: Train Set

Section Contents

Polynomial Regression

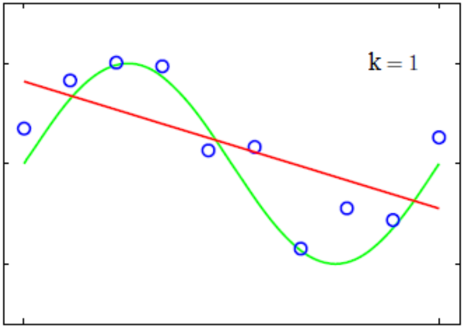
Polynomial regression fits a nonlinear relationship between independent variable x and the dependent variable y.



1. $k = 0$ Constant (Constant line (average of output values), not a good fit. Under-fitting)

Polynomial Regression

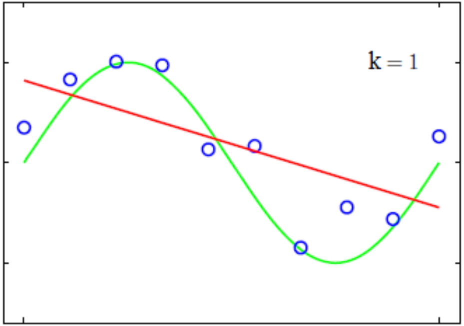
Polynomial regression fits a nonlinear relationship between independent variable x and the dependent variable y.



1. $k = 0$ Constant (Constant line (average of output values), not a good fit. Under-fitting)
2. $k = 1$ Straight Line (Linear regression, not a good fit. Under-fitting)

Polynomial Regression

Polynomial regression fits a nonlinear relationship between independent variable x and the dependent variable y.



1. $\mathrm{k} = 0$ Constant (Constant line (average of output values), not a good fit. Under-fitting)
2. $\mathrm{k} = 1$ Straight Line (Linear regression, not a good fit. Under-fitting)

Under-fitting

Linear regression is under-fitting the data (high-bias).

Polynomial Regression

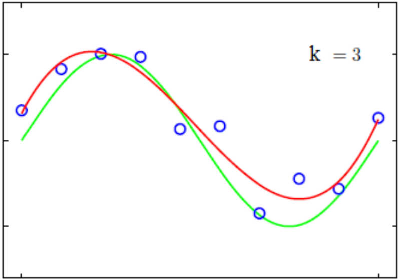
- To overcome under-fitting, we need to increase the complexity of the model.

Polynomial Regression

- To overcome under-fitting, we need to increase the complexity of the model.
- To generate a higher order equation, can add powers of the original features as new features. The linear model $h = \theta_0 + \theta_1 x$
can be transformed to $h = \theta_0 + \theta_1 x + \theta_2 x^2 (x\ -\ \text{squared})$

Polynomial Regression

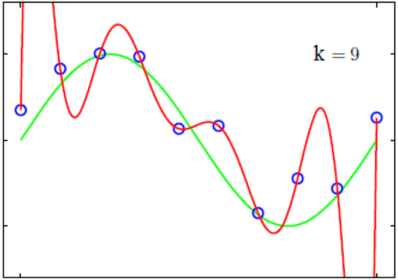
- To overcome under-fitting, we need to increase the complexity of the model.
- To generate a higher order equation, can add powers of the original features as new features. The linear model $h = \theta_0 + \theta_1 x$ can be transformed to $h = \theta_0 + \theta_1 x + \theta_2 x^2 (x - \text{squared})$



1. k = 2 Parabola
2. k = 3 Cubic (Polynomial function, fits nicely)

Polynomial Regression

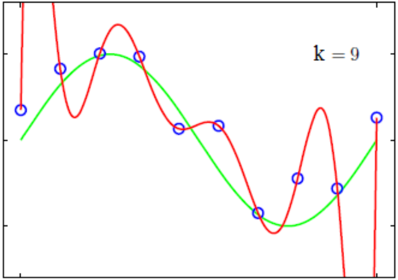
- To overcome under-fitting, we need to increase the complexity of the model.
- To generate a higher order equation, can add powers of the original features as new features. The linear model $h = \theta_0 + \theta_1 x$
can be transformed to $h = \theta_0 + \theta_1 x + \theta_2 x^2 (x - \text{squared})$



1. k = 2 Parabola
2. k = 3 Cubic (Polynomial function, fits nicely)
3. k = 9 9^{th} degree polynomial. Over-fitting

Polynomial Regression

- To overcome under-fitting, we need to increase the complexity of the model.
- To generate a higher order equation, can add powers of the original features as new features. The linear model $h = \theta_0 + \theta_1 x$ can be transformed to $h = \theta_0 + \theta_1 x + \theta_2 x^2 (x - \text{squared})$



1. k = 2 Parabola
2. k = 3 Cubic (Polynomial function, fits nicely)
3. k = 9 9^{th} degree polynomial. Over-fitting

General form for Polynomial Regression:

$$h(\theta) = \theta_0 + \theta_1 x + \theta_2 x^2 + \cdots + \theta_k x^k \tag{19}$$

Solving for Polynomial Regression

Find coefficients w, for cubic regression, number of samples $= n$

Equation: $w_0 + w_1 x + w_2 x^2 + w_3 x^3 \approx y$

Solving for Polynomial Regression

Find coefficients w, for cubic regression, number of samples $= n$

Equation: $w_0 + w_1x + w_2x^2 + w_3x^3 \approx y$

Write this in matrix format:

Solving for Polynomial Regression

Find coefficients w, for cubic regression, number of samples $= n$

Equation: $w_0 + w_1 x + w_2 x^2 + w_3 x^3 \approx y$

Write this in matrix format:

$$\begin{bmatrix} 1 & x_1 & (x_1)^2 & (x_1)^3 \\ 1 & x_2 & (x_2)^2 & (x_2)^3 \\ 1 & x_3 & (x_3)^2 & (x_3)^3 \\ & \vdots & \vdots & \\ 1 & x_n & (x_n)^2 & (x_n)^3 \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \end{bmatrix} \approx \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{bmatrix}$$

where n is number of samples in training data.

Solving for Polynomial Regression

Solve for W:

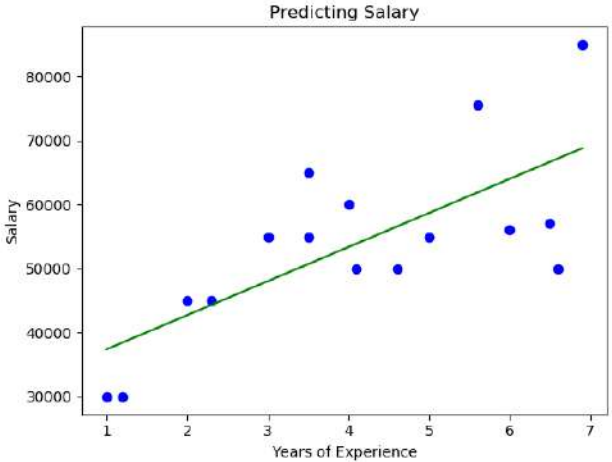
$$\begin{bmatrix} 1 & x_1 & (x_1)^2 & (x_1)^3 \\ 1 & x_2 & (x_2)^2 & (x_2)^3 \\ 1 & x_3 & (x_3)^2 & (x_3)^3 \\ & \vdots & \vdots & \\ 1 & x_n & (x_n)^2 & (x_n)^3 \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \end{bmatrix} \approx \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{bmatrix}$$

$$X\,W \approx Y$$

$$X X^T\,W \approx X^T Y$$

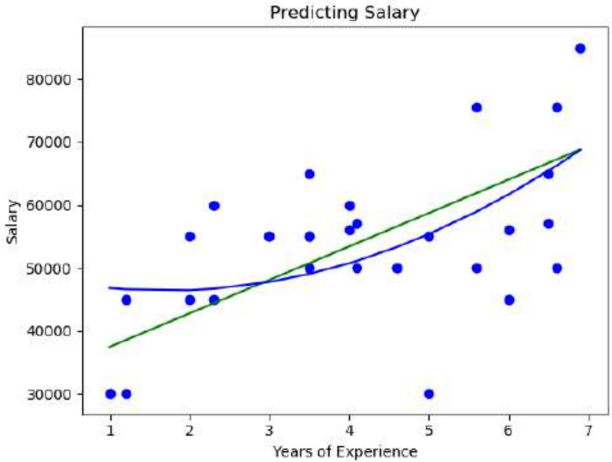
$$W \approx (X^T X)^{-1} X^T Y \text{ (Closed-form solution)}$$

Polynomial Regression Example



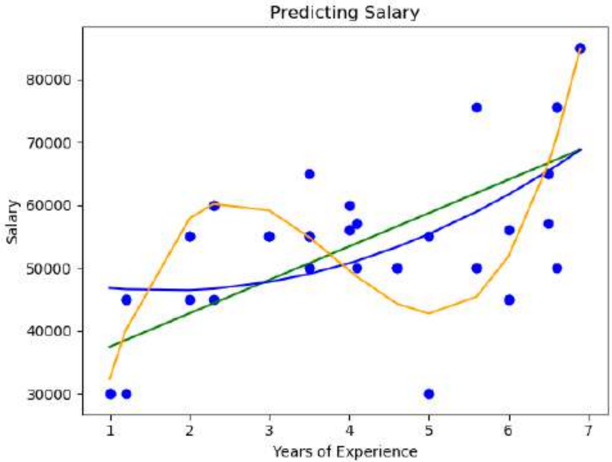
1. Polynomial Degree 1

Polynomial Regression Example



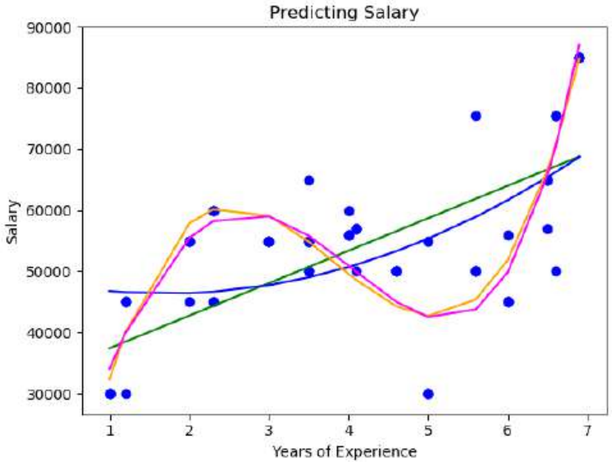
1. Polynomial Degree 1
2. Polynomial Degree 2

Polynomial Regression Example



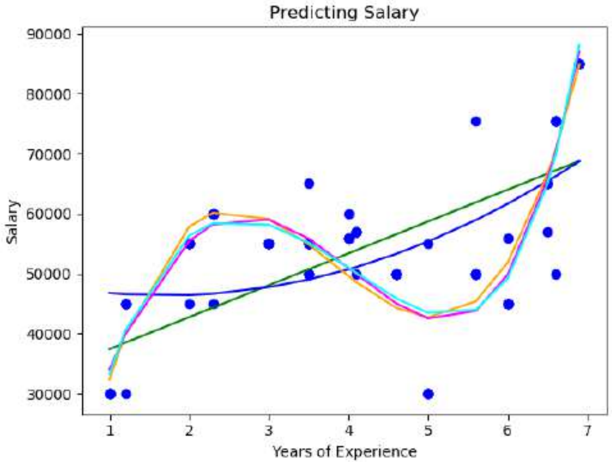
1. Polynomial Degree 1
2. Polynomial Degree 2
3. Polynomial Degree 3

Polynomial Regression Example



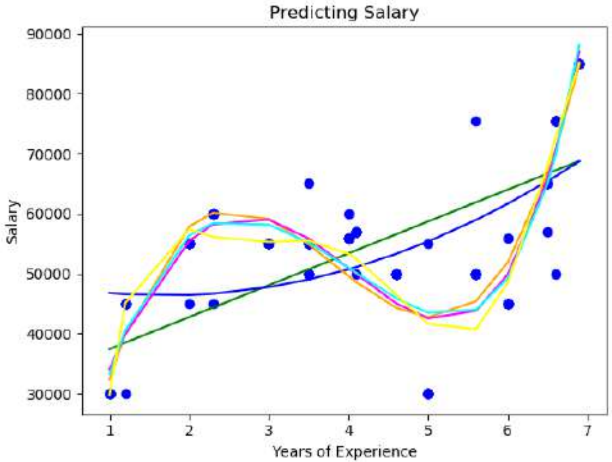
1. Polynomial Degree 1
2. Polynomial Degree 2
3. Polynomial Degree 3
4. Polynomial Degree 4

Polynomial Regression Example



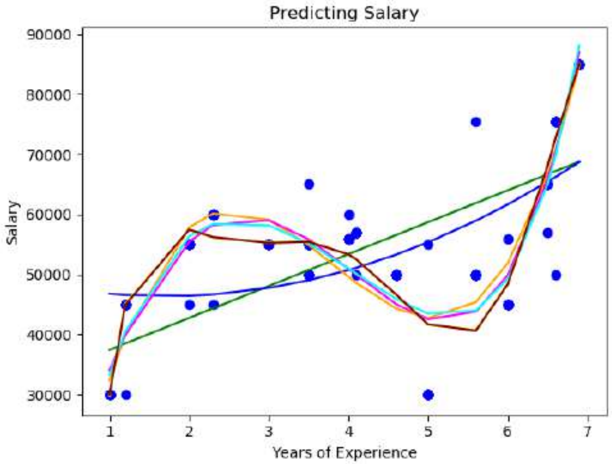
1. Polynomial Degree 1
2. Polynomial Degree 2
3. Polynomial Degree 3
4. Polynomial Degree 4
5. Polynomial Degree 5

Polynomial Regression Example



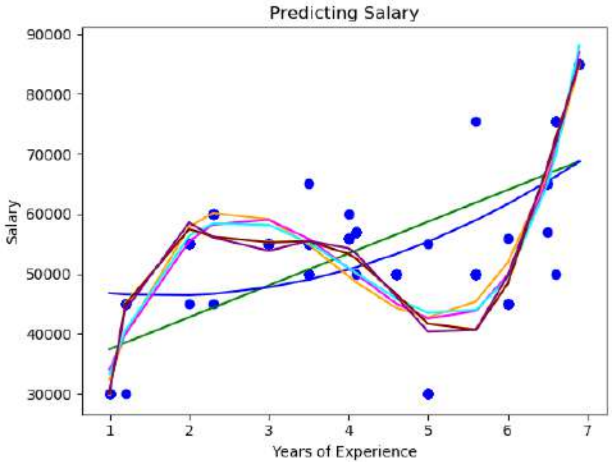
1. Polynomial Degree 1
2. Polynomial Degree 2
3. Polynomial Degree 3
4. Polynomial Degree 4
5. Polynomial Degree 5
6. Polynomial Degree 6

Polynomial Regression Example



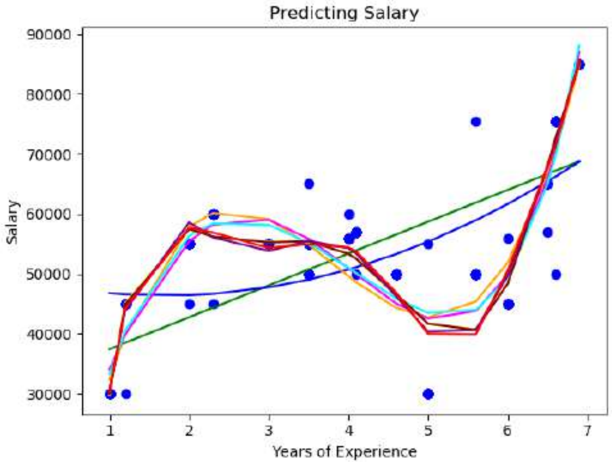
1. Polynomial Degree 1
2. Polynomial Degree 2
3. Polynomial Degree 3
4. Polynomial Degree 4
5. Polynomial Degree 5
6. Polynomial Degree 6
7. Polynomial Degree 7

Polynomial Regression Example



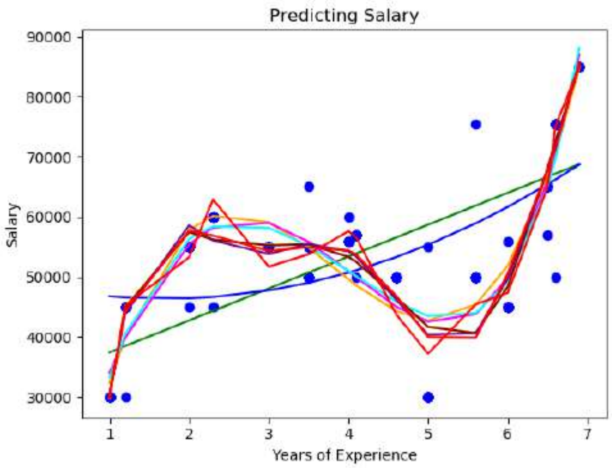
1. Polynomial Degree 1
2. Polynomial Degree 2
3. Polynomial Degree 3
4. Polynomial Degree 4
5. Polynomial Degree 5
6. Polynomial Degree 6
7. Polynomial Degree 7
8. Polynomial Degree 8

Polynomial Regression Example



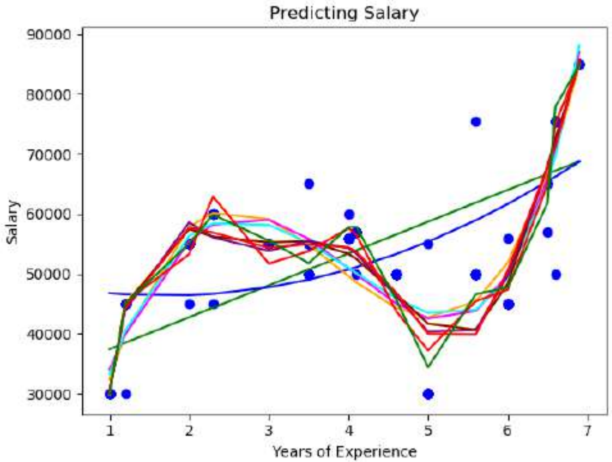
1. Polynomial Degree 1
2. Polynomial Degree 2
3. Polynomial Degree 3
4. Polynomial Degree 4
5. Polynomial Degree 5
6. Polynomial Degree 6
7. Polynomial Degree 7
8. Polynomial Degree 8
9. Polynomial Degree 9

Polynomial Regression Example



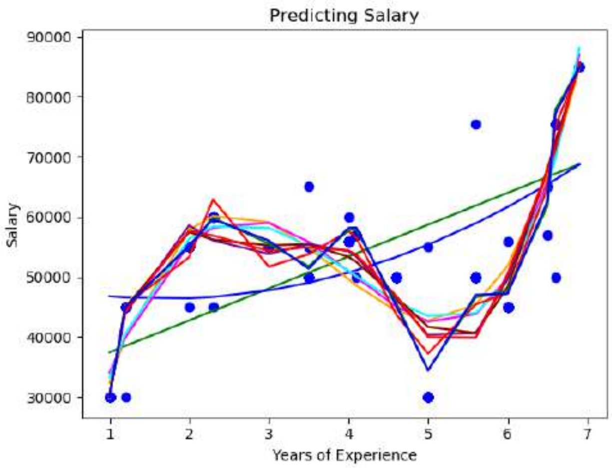
1. Polynomial Degree 1
2. Polynomial Degree 2
3. Polynomial Degree 3
4. Polynomial Degree 4
5. Polynomial Degree 5
6. Polynomial Degree 6
7. Polynomial Degree 7
8. Polynomial Degree 8
9. Polynomial Degree 9
10. Polynomial Degree 10

Polynomial Regression Example



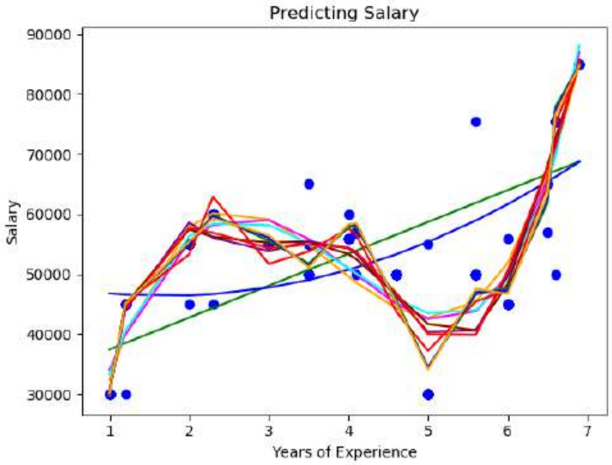
1. Polynomial Degree 1
2. Polynomial Degree 2
3. Polynomial Degree 3
4. Polynomial Degree 4
5. Polynomial Degree 5
6. Polynomial Degree 6
7. Polynomial Degree 7
8. Polynomial Degree 8
9. Polynomial Degree 9
10. Polynomial Degree 10
11. Polynomial Degree 11

Polynomial Regression Example



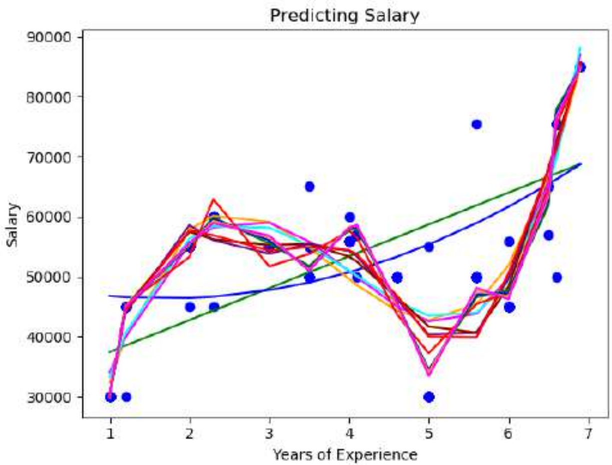
1. Polynomial Degree 1
2. Polynomial Degree 2
3. Polynomial Degree 3
4. Polynomial Degree 4
5. Polynomial Degree 5
6. Polynomial Degree 6
7. Polynomial Degree 7
8. Polynomial Degree 8
9. Polynomial Degree 9
10. Polynomial Degree 10
11. Polynomial Degree 11
12. Polynomial Degree 12

Polynomial Regression Example



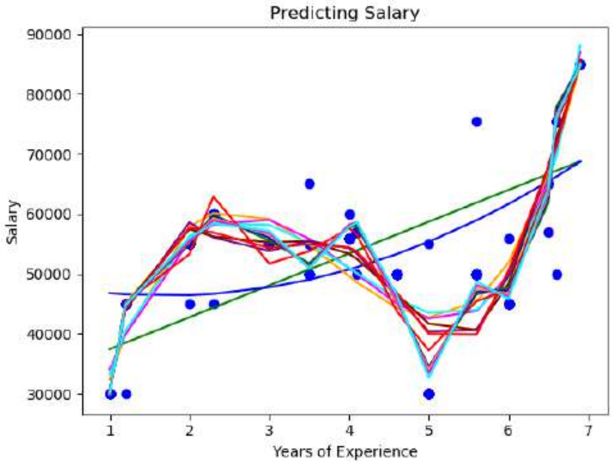
1. Polynomial Degree 1
2. Polynomial Degree 2
3. Polynomial Degree 3
4. Polynomial Degree 4
5. Polynomial Degree 5
6. Polynomial Degree 6
7. Polynomial Degree 7
8. Polynomial Degree 8
9. Polynomial Degree 9
10. Polynomial Degree 10
11. Polynomial Degree 11
12. Polynomial Degree 12
13. Polynomial Degree 13

Polynomial Regression Example



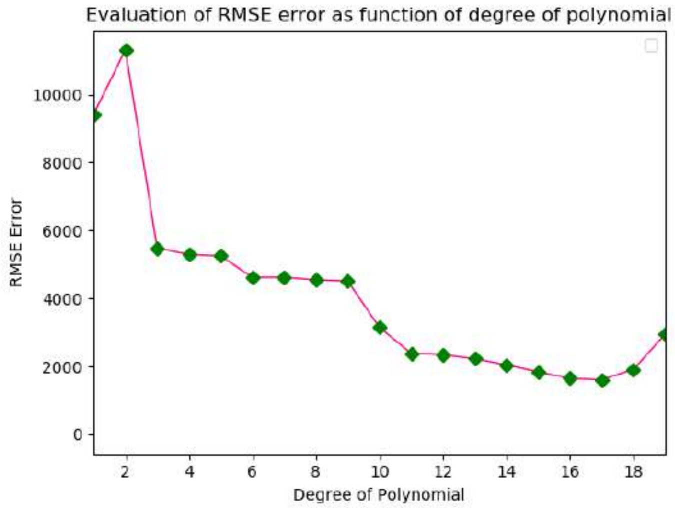
1. Polynomial Degree 1
2. Polynomial Degree 2
3. Polynomial Degree 3
4. Polynomial Degree 4
5. Polynomial Degree 5
6. Polynomial Degree 6
7. Polynomial Degree 7
8. Polynomial Degree 8
9. Polynomial Degree 9
10. Polynomial Degree 10
11. Polynomial Degree 11
12. Polynomial Degree 12
13. Polynomial Degree 13
14. Polynomial Degree 14

Polynomial Regression Example

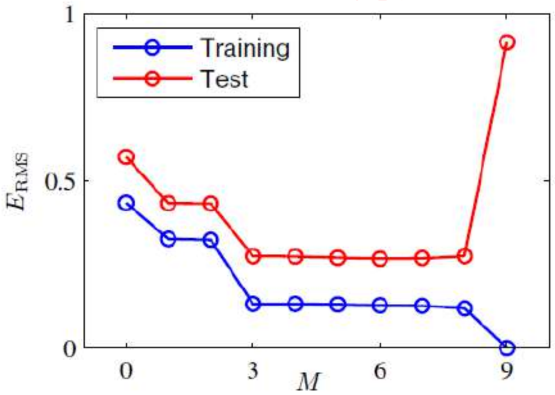


1. Polynomial Degree 1
2. Polynomial Degree 2
3. Polynomial Degree 3
4. Polynomial Degree 4
5. Polynomial Degree 5
6. Polynomial Degree 6
7. Polynomial Degree 7
8. Polynomial Degree 8
9. Polynomial Degree 9
10. Polynomial Degree 10
11. Polynomial Degree 11
12. Polynomial Degree 12
13. Polynomial Degree 13
14. Polynomial Degree 14
15. Polynomial Degree 15

Polynomial Regression Example: RMSE



Best Fit



Either to use Gradient Descent or Normal Equation Method?

Consider: d dimensional feature vector and n training examples.

Gradient Descent

1. Need to choose α
2. Needs many iterations
3. Needs feature scaling
4. Works well for high dimensional feature vector
5. Reasonably efficient for a very large number (millions) of features

Analytical method

1. No need to choose α
2. No need to iterations
3. No Need for feature scaling
4. Slow for high dimensional feature vector. As it need to compute $(X^T X)^{-1}$ which has complexity of $\mathcal{O}(d^3)$
5. Issue of non-invertible or singular matrix
6. Doesn't work well with complex classifiers i.e. logistic regression etc.

Section Contents

Exercise

Implement

1. Do implement GD (without using any library)

Further Reading

1. Cost functions
2. Multivariate Regression
3. Surface plots / Contour plots
4. Variants of Gradient Descent (GD)
5. Regularization: Ridge Regression and LASSO (Least Absolute Shrinkage and Selection Operator) Regression

Machine Learning
Decision Tree

Dr. Rizwan Ahmed Khan

Outline

Section Contents

Reference Books

- Chapter 3: Machine Learning, Tom MITCHELL, McGraw Hill, latest edition.

Reference Books

- Chapter 3: Machine Learning, Tom MITCHELL, McGraw Hill, latest edition.
- Chapter 9: Introduction to Machine Learning, Ethem ALPAYDIN, The MIT Press, latest edition.

Reference Books

- Chapter 3: Machine Learning, Tom MITCHELL, McGraw Hill, latest edition.
- Chapter 9: Introduction to Machine Learning, Ethem ALPAYDIN, The MIT Press, latest edition.
- Microsoft Research Technical Report TR-2011-114: A. Criminisi et al. Decision Forests for Classification, Regression, Density Estimation, Manifold Learning and Semi-Supervised Learning. Microsoft Research 2011.

Problem Formalization / Function approximation

Problem formalization

- Set of possible instances X i.e. $\{< \vec{x}_i, y_i >\}$
- Dataset D, given by $D = \{< \vec{x}_i, y_i >, \ldots, < \vec{x}_n, y_n >\} \subseteq X \times Y$
  Where:
  $\vec{x}_i$ is a feature vector ($\mathbb{R}^d$),
  y_i is a label / target variable,
  X is space of all features and
  Y is space of labels.
- Unknown target function $f : X \rightarrow Y$
- Set of function hypotheses $H = \{h | h : X \rightarrow Y\}$

Output:

- Hypothesis $h \in H$ that best approximates target function f. Or a classification "rule" that can determine the class of any object from its attributes values.
- If training is done correctly $h(\vec{x}_i) \approx y_i$

Introduction

- It is a method of approximating discrete-valued functions, learned function is represented by decision tree.
  - There are some extensions that can handle real-valued functions.

Introduction

- It is a method of approximating discrete-valued functions, learned function is represented by decision tree.
    - There are some extensions that can handle real-valued functions.
- Learned trees can also be represented as sets of if-then rules (advantage as it is human readable).

Introduction

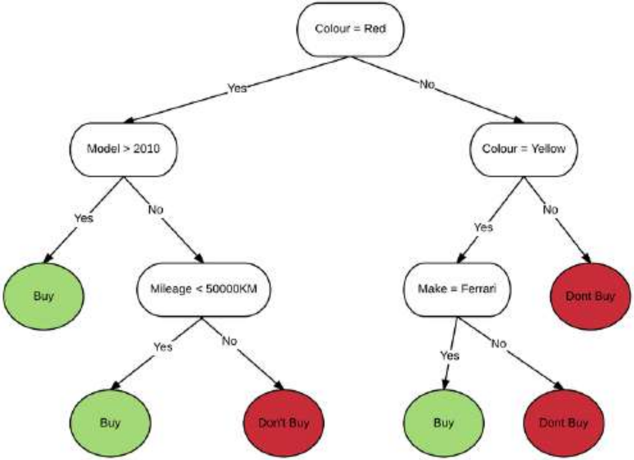
- It is a method of approximating discrete-valued functions, learned function is represented by decision tree.
  - There are some extensions that can handle real-valued functions.
- Learned trees can also be represented as sets of if-then rules (advantage as it is human readable).
- It is simple yet powerful learning algorithm.

Introduction

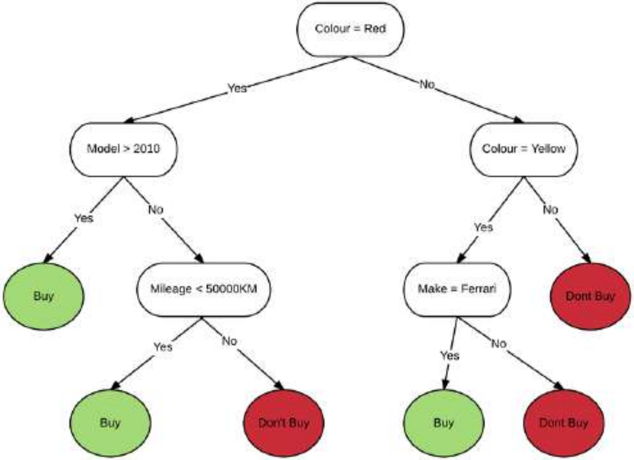
- It is a method of approximating discrete-valued functions, learned function is represented by decision tree.
  - There are some extensions that can handle real-valued functions.
- Learned trees can also be represented as sets of if-then rules (advantage as it is human readable).
- It is simple yet powerful learning algorithm.
- In next Section, we will look how it represents data.

Section Contents

Decision tree representation

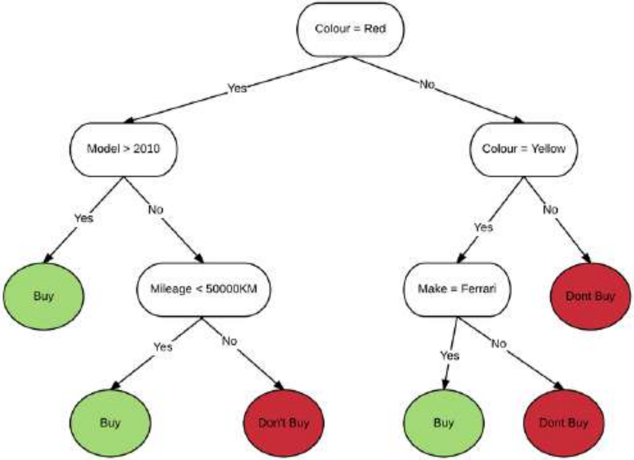


Decision tree representation



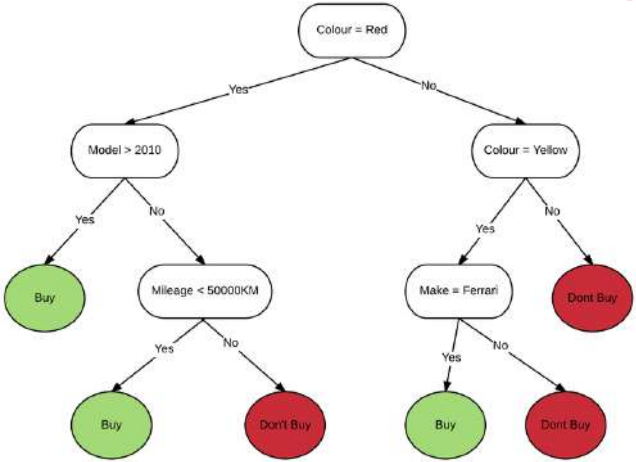
- Can you write if-then rules for this tree?

Decision tree representation



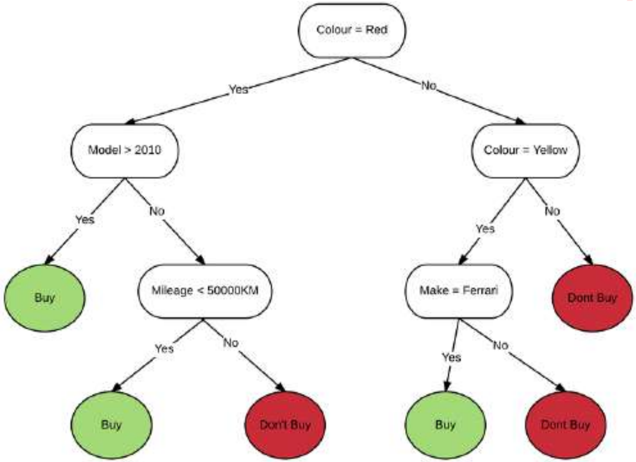
- Can you write if-then rules for this tree?
- Each internal node tests an attribute (discrete-valued).

Decision tree representation



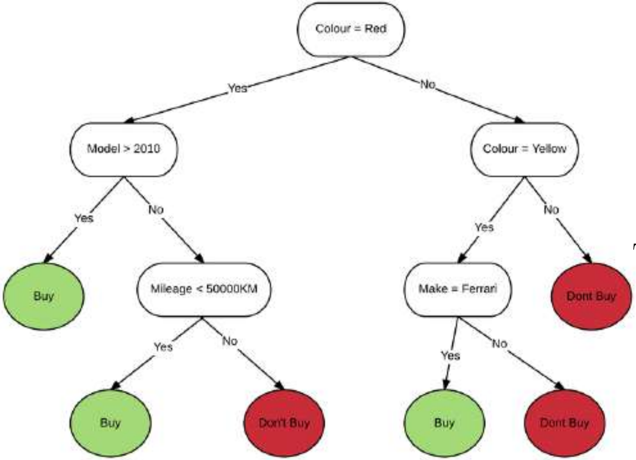
- Can you write if-then rules for this tree?
- Each internal node tests an attribute (discrete-valued).
- Each branch corresponds to an attribute value.

Decision tree representation



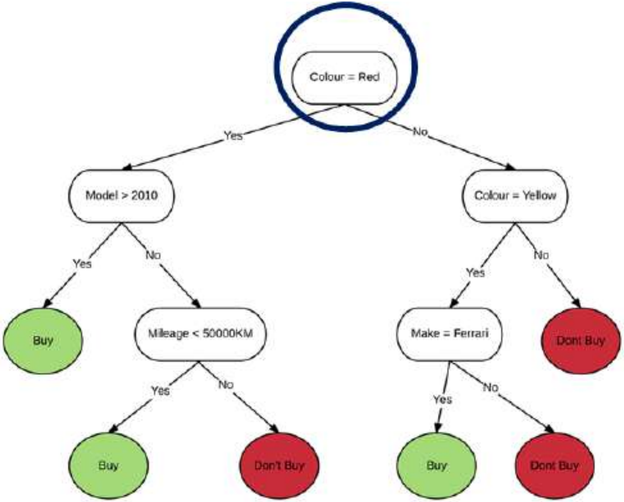
- Can you write if-then rules for this tree?
- Each internal node tests an attribute (discrete-valued).
- Each branch corresponds to an attribute value.
- Each leaf node assigns a classification / label. Predict y or $P(y|x \in leaf)$.

Decision tree representation



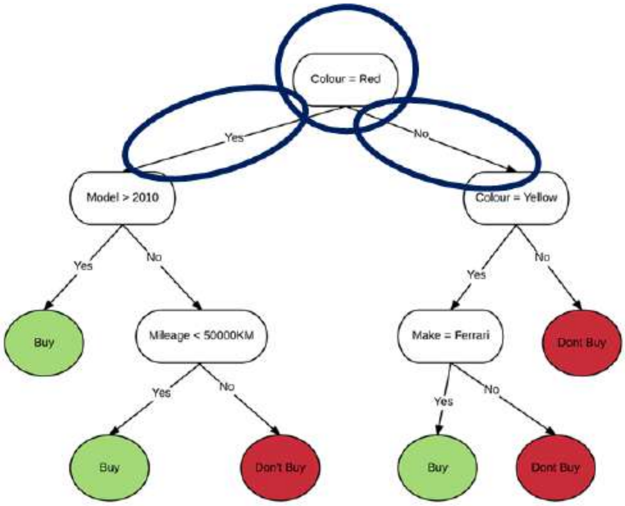
There are different parts of it:

Decision tree representation



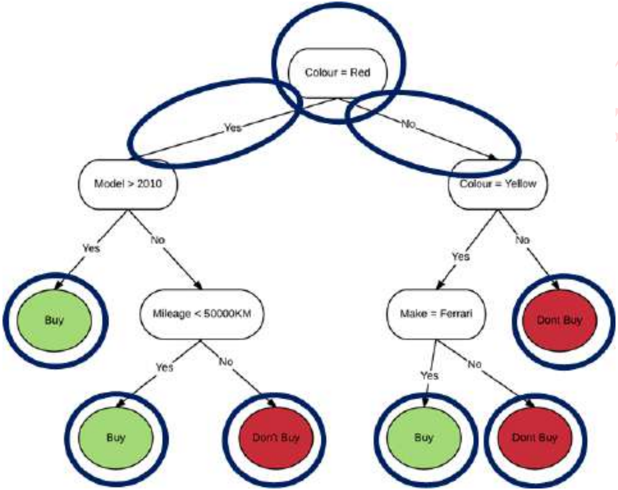
These are nodes, in fact decision nodes. Decision is based on "attribute / feature" value.

Decision tree representation



"Edges" represent path to follow considering decision node attribute value. In summary, nodes represent attributes and edges represent values.

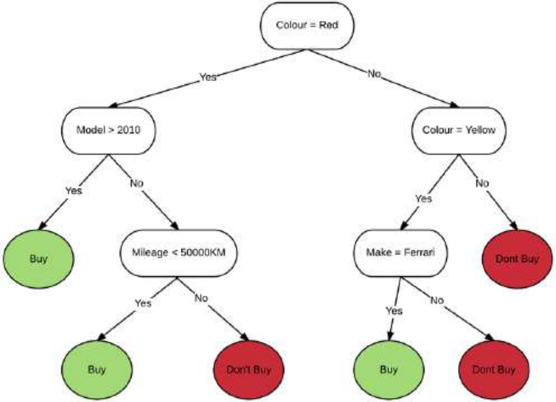
Decision tree representation



"Circles" at the bottom of tree represent decisions. Decision is reached after answering / probing different attribute values.

Decision tree representation

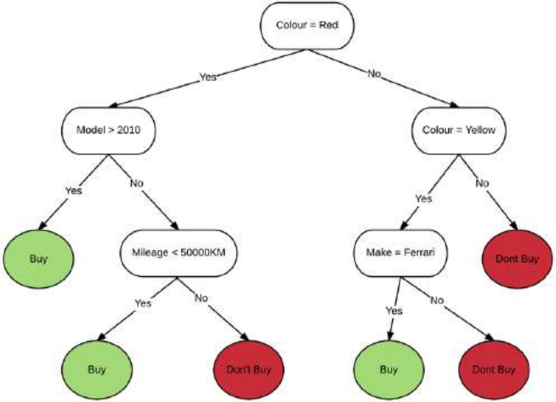
- By asking series of questions, decisions / classification can be made. It's not necessary that all attributes take part in decision making process.



Color	Model	Mileage	Type	Make	Decision
Red	2011	40000	SUV	BMW	Buy
Red	2010	35000	Sports	BMW	Buy
Red	2010	55000	Sedan	Audi	Don't Buy
Yellow	2009	55000	Sedan	Ferrari	Buy
Yellow	2009	55000	SUV	Audi	Don't Buy
Blue	2009	35000	Sports	Audi	Don't Buy
Blue	2011	45000	SUV	BMW	Don't Buy

Decision tree representation

- By asking series of questions, decisions / classification can be made. It's not necessary that all attributes take part in decision making process.

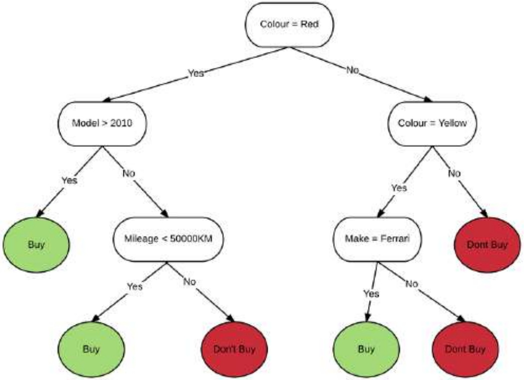


- What will be the output?

Red	2013	60000	Sedan	BMW	??

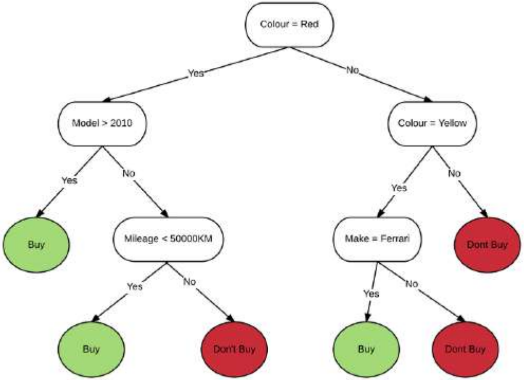
Decision tree representation

In fact, decision tree represents disjunction of conjunctions of constraints on the attribute values of instances / examples. An example is classified by sorting it through the tree from the root to the leaf node.



Decision tree representation

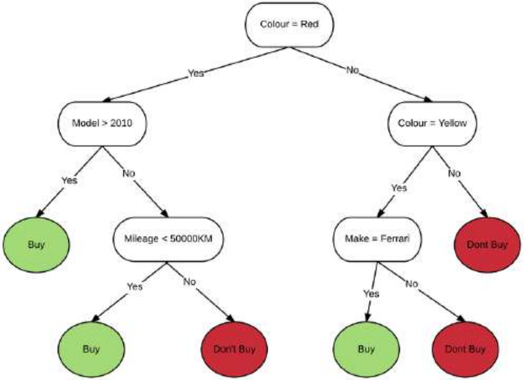
In fact, decision tree represents disjunction of conjunctions of constraints on the attribute values of instances / examples. An example is classified by sorting it through the tree from the root to the leaf node.



Disjunction of Conjunctions :

Decision tree representation

In fact, decision tree represents disjunction of conjunctions of constraints on the attribute values of instances / examples. An example is classified by sorting it through the tree from the root to the leaf node.



Disjunction of Conjunctions :

Disjunction of Conjunctions

(Color=red $\wedge$ Model $>$ 2010)
$\vee$ (Color=red $\wedge$ Model $<$ 2010 $\wedge$ Mileage$<$50000)
$\vee$ (Color=yellow $\wedge$ Make=Ferrari)

Expressiveness of trees: Boolean function

AND $(x \wedge y)$

x	y	xy
0	0	0
0	1	0
1	0	0
1	1	1

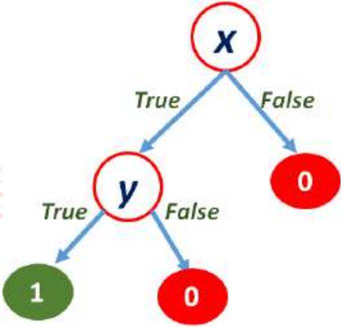
- What will be its decision tree?

Expressiveness of trees: Boolean function

AND $(x \wedge y)$

x	y	xy
0	0	0
0	1	0
1	0	0
1	1	1

- What will be its decision tree?

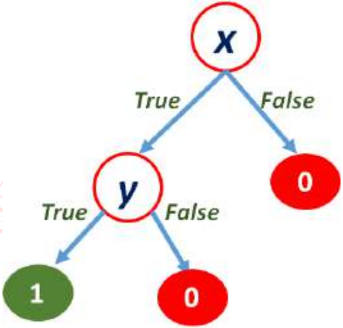


Expressiveness of trees: Boolean function

AND $(x \wedge y)$

x	y	xy
0	0	0
0	1	0
1	0	0
1	1	1

- What will be its decision tree?



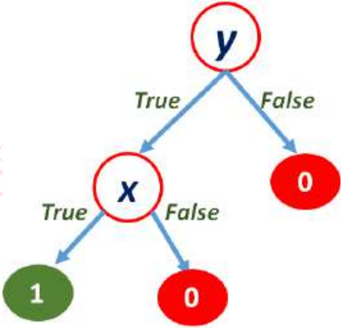
- How about swapping x and y?

Expressiveness of trees: Boolean function

- What will be its decision tree?
- How about swapping x and y?

AND $(x \wedge y)$

x	y	xy
0	0	0
0	1	0
1	0	0
1	1	1

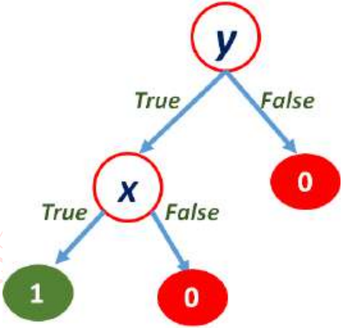


Expressiveness of trees: Boolean function

AND $(x \wedge y)$

x	y	xy
0	0	0
0	1	0
1	0	0
1	1	1

- What will be its decision tree?



By choosing different attribute at the top of the tree, algorithm may find better tree representation (not true in this case), but generally it matters.

Expressiveness of trees: Boolean function

OR $(x \vee y)$

x	y	x+y
0	0	0
0	1	1
1	0	1
1	1	1

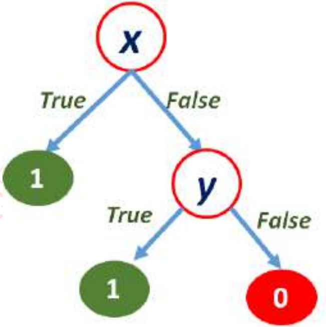
- What will be its decision tree?

Expressiveness of trees: Boolean function

OR $(x \lor y)$

x	y	x+y
0	0	0
0	1	1
1	0	1
1	1	1

- What will be its decision tree?

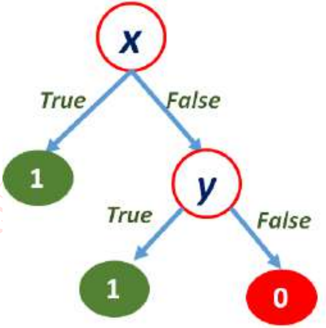


Expressiveness of trees: Boolean function

OR $(x \vee y)$

x	y	x+y
0	0	0
0	1	1
1	0	1
1	1	1

- What will be its decision tree?



- How about swapping x and y?

Expressiveness of trees: Boolean function

OR $(x \vee y)$

x	y	x+y
0	0	0
0	1	1
1	0	1
1	1	1

- What will be its decision tree?
- How about swapping x and y?

By choosing different attribute at the top of the tree, algorithm may find better tree representation (not true in this case), but generally it matters.

Expressiveness of trees: Boolean function

XOR

x	y	x⊕y
0	0	0
0	1	1
1	0	1
1	1	0

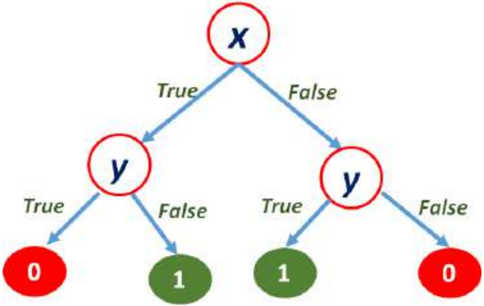
- What will be its decision tree?

Expressiveness of trees: Boolean function

- What will be its decision tree?

XOR

x	y	x⊕y
0	0	0
0	1	1
1	0	1
1	1	0

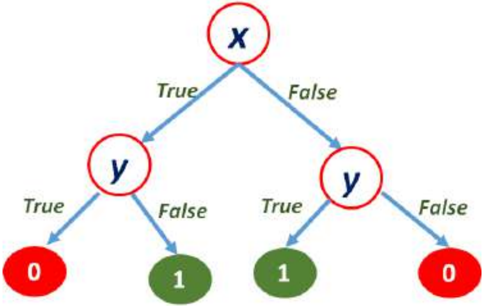


Expressiveness of trees: Boolean function

XOR

x	y	$x \oplus y$
0	0	0
0	1	1
1	0	1
1	1	0

- What will be its decision tree?



- This tree is another representation of full truth table (unlike previous slides, which was compact representation as some branches were not required).
- This "compactness" will matter for inducing tree with many attributes.

Expressiveness of trees

- Consider this problem:

- How many decision trees to be looked at in order to find the right one? (how big is hypotheses space H?)
- Dataset with d dimensional vector / attributes (Boolean).
- Target function Y is also Boolean.

f_1	f_2	f_3	.	.	f_d	**Label**
x_1^1	x_2^1	x_3^1	.	.	x_d^1	+
x_1^2	x_2^2	x_3^2	.	.	x_d^2	-
x_1^3	x_2^3	x_3^3	.	.	x_d^3	+
x_1^4	x_2^4	x_3^4	.	.	x_d^4	-
x_1^5	x_2^5	x_3^5	.	.	x_d^5	-
.	.	.	.	.	.	.

Expressiveness of trees

- Consider this problem:

- How many decision trees to be looked at in order to find the right one? (how big is hypotheses space H?)
- Dataset with d dimensional vector / attributes (Boolean).
- Target function Y is also Boolean.

f_1	f_2	f_3	.	.	f_d	**Label**
T	T	T	.	.	T	+
T	T	T	.	.	F	-
F	T	T	.	.	T	+
T	F	T	.	.	T	-
T	T	F	.	.	T	-
.	.	.	.	.	.	.

Expressiveness of trees

- Consider this problem:

- How many decision trees to be looked at in order to find the right one? (how big is hypotheses space H?)
- Dataset with d dimensional vector / attributes (Boolean).
- Target function Y is also Boolean.

f_1	f_2	f_3	.	.	f_d	**Label**
T	T	T	.	.	T	+
T	T	T	.	.	F	-
F	T	T	.	.	T	+
T	F	T	.	.	T	-
T	T	F	.	.	T	-
.	.	.	.	.	.	.

- How many row are there in the table?

Expressiveness of trees

- Consider this problem:

- How many decision trees to be looked at in order to find the right one? (how big is hypotheses space H?)
- Dataset with d dimensional vector / attributes (Boolean).
- Target function Y is also Boolean.

f_1	f_2	f_3	.	.	f_d	**Label**
T	T	T	.	.	T	+
T	T	T	.	.	F	-
F	T	T	.	.	T	+
T	F	T	.	.	T	-
T	T	F	.	.	T	-
.	.	.	.	.	.	.

- How many row are there in the table?
- There are 2^d possibilities.

Expressiveness of trees

- Consider this problem:

- How many decision trees to be looked at in order to find the right one? (how big is hypotheses space H?)
- Dataset with d dimensional vector / attributes (Boolean).
- Target function Y is also Boolean.

f_1	f_2	f_3	.	.	f_d	**Label**
T	T	T	.	.	T	+
T	T	T	.	.	F	-
F	T	T	.	.	T	+
T	F	T	.	.	T	-
T	T	F	.	.	T	-
.	.	.	.	.	.	.

- How many functions or decision tree possibilities are there in 2^d rows?

Expressiveness of trees

- Consider this problem:

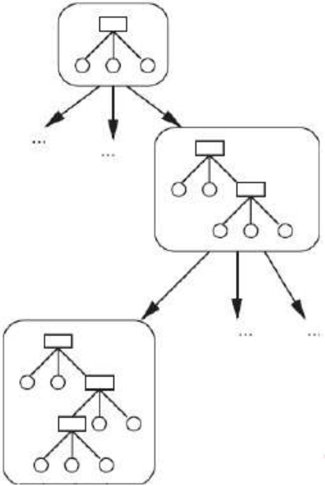
- How many decision trees to be looked at in order to find the right one? (how big is hypotheses space H?)
- Dataset with d dimensional vector / attributes (Boolean).
- Target function Y is also Boolean.

f_1	f_2	f_3	.	.	f_d	**Label**
T	T	T	.	.	T	+
T	T	T	.	.	F	-
F	T	T	.	.	T	+
T	F	T	.	.	T	-
T	T	F	.	.	T	-
.	.	.	.	.	.	.

- How many functions or decision tree possibilities are there in 2^d rows?

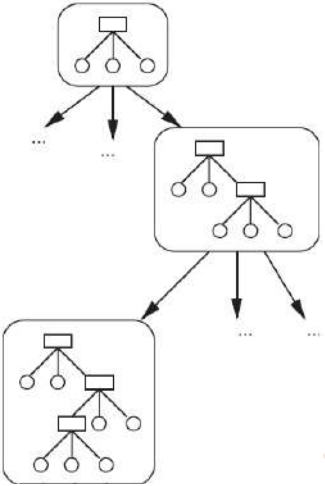
- As there are 2^d rows, output for each row also have two possibilities (either "true" or "false") . Thus, 2^{2^d} possibilities. This is double exponential and gives very big number for very small value of d.

Expressiveness of trees



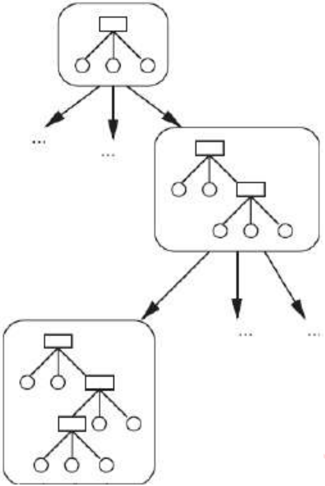
- 2^{2^d} grows very fast.

Expressiveness of trees



- 2^{2^d} grows very fast.
- On the other hand it shows that hypothesis space H is very expressive and there are lots and lots of functions (as we seen on previous slide ("OR" & "AND" function)) that can be represented by decision trees.

Expressiveness of trees



- 2^{2^d} grows very fast.
- On the other hand it shows that hypothesis space H is very expressive and there are lots and lots of functions (as we seen on previous slide ("OR" & "AND" function)) that can be represented by decision trees.
- This also points to the fact that algorithm that selects tree should be robust enough to find the best representation given such huge number of choices.

Decision tree suitability

- Instances are represented by fix set of attributes e.g. attribute "colour" and its values "red" and "yellow".

Different extensions are proposed to basic algorithms which allows handling of real-valued attributes as well.

Decision tree suitability

- Instances are represented by fix set of attributes e.g. attribute “colour” and its values “red” and “yellow”.

Different extensions are proposed to basic algorithms which allows handling of real-valued attributes as well.

- The target function has discrete output values / classes.

Different extensions are proposed which allows handling of real-valued outputs as well but it is less common.

Decision tree suitability

- Instances are represented by fix set of attributes e.g. attribute “colour” and its values “red” and “yellow”.

Different extensions are proposed to basic algorithms which allows handling of real-valued attributes as well.

- The target function has discrete output values / classes.

Different extensions are proposed which allows handling of real-valued outputs as well but it is less common.

- Decision tree learning algorithms (ID3, C4.5) are robust to errors in classifications labels and errors in attribute values.

Decision tree suitability

- Instances are represented by fix set of attributes e.g. attribute “colour” and its values “red” and “yellow”.

Different extensions are proposed to basic algorithms which allows handling of real-valued attributes as well.

- The target function has discrete output values / classes.

Different extensions are proposed which allows handling of real-valued outputs as well but it is less common.

- Decision tree learning algorithms (ID3, C4.5) are robust to errors in classifications labels and errors in attribute values.
- Decision tree learning algorithms are robust to missing attribute values in training data.

Section Contents

ID3 Learning Algorithm

- ID3 (Iterative Dichotomiser 3) is an algorithm invented by Ross Quinlan[1] used to generate a decision tree from a dataset.

[1] Quinlan, J. R. 1986. Induction of Decision Trees. Mach. Learn., pp 81–106

ID3 Learning Algorithm

- ID3 (Iterative Dichotomiser 3) is an algorithm invented by Ross Quinlan[1] used to generate a decision tree from a dataset.
- There are many extensions to it e.g C4.5, CART etc.

[1] Quinlan, J. R. 1986. Induction of Decision Trees. Mach. Learn., pp 81–106

ID3 Learning Algorithm

- ID3 (Iterative Dichotomiser 3) is an algorithm invented by Ross Quinlan[1] used to generate a decision tree from a dataset.
- There are many extensions to it e.g C4.5, CART etc.
- ID3 learns decision tree by constructing them top-down, "which attribute to be tested at the top?"

Which attribute is most discriminative or provides most information to classify ??

[1] Quinlan, J. R. 1986. Induction of Decision Trees. Mach. Learn., pp 81–106

ID3 Learning Algorithm

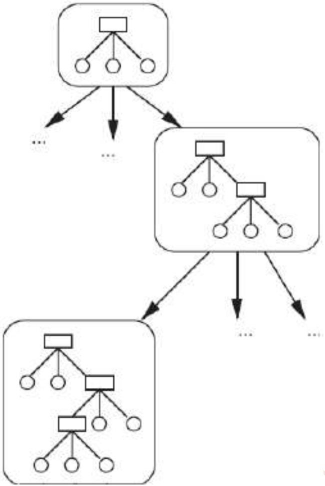
- ID3 (Iterative Dichotomiser 3) is an algorithm invented by Ross Quinlan[1] used to generate a decision tree from a dataset.
- There are many extensions to it e.g C4.5, CART etc.
- ID3 learns decision tree by constructing them top-down, "which attribute to be tested at the top?"

Which attribute is most discriminative or provides most information to classify ??

- Attributes to be evaluated by Statistical test to determine how well specific attribute classifies training data / examples.

[1] Quinlan, J. R. 1986. Induction of Decision Trees. Mach. Learn., pp 81–106

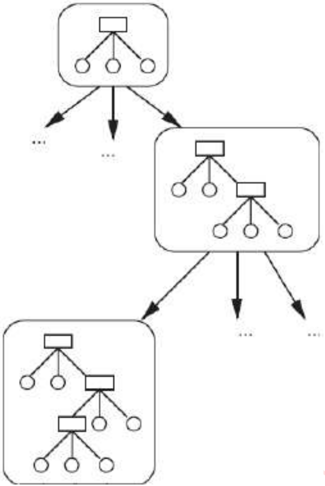
Search for the best hypothesis: ID3



- ID3 performs search through space of decision trees.

[2]Image from Tom's book

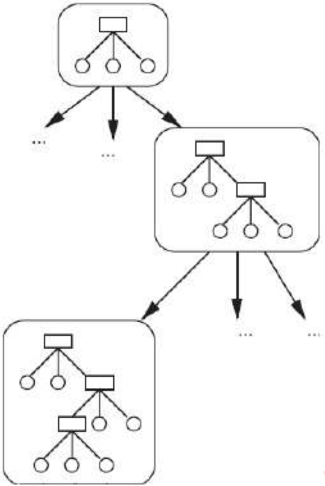
Search for the best hypothesis: ID3



- ID3 performs search through space of decision trees.
- Search to find "best" attributes to test at the top.

[2]Image from Tom's book

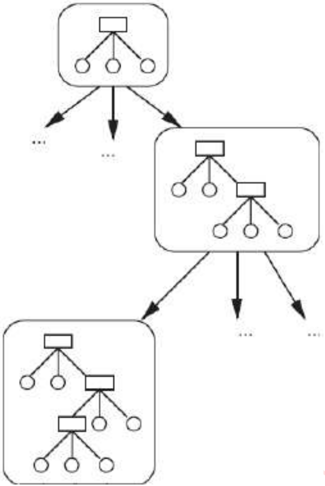
Search for the best hypothesis: ID3



- ID3 performs search through space of decision trees.
- Search to find "best" attributes to test at the top.
- Based on tested attribute examples are sorted, either side of the test attribute.

[2] Image from Tom's book

Search for the best hypothesis: ID3

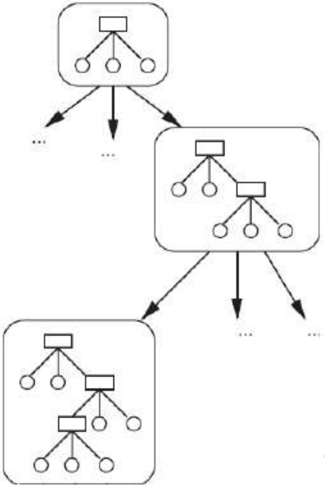


2

- ID3 performs search through space of decision trees.
- Search to find "best" attributes to test at the top.
- Based on tested attribute examples are sorted, either side of the test attribute.
- Feature space is thus recursively divided till the "pure" leaf (uniformly +ve or uniformly -ve) is obtained or "stopping criteria" is met.

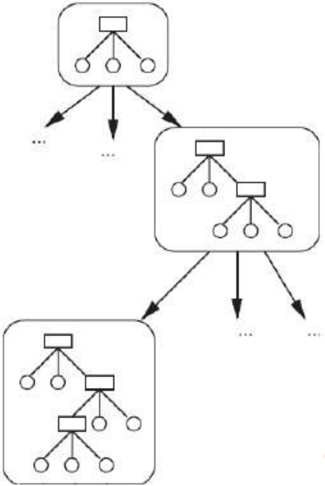
[2] Image from Tom's book

Search for the best hypothesis: ID3



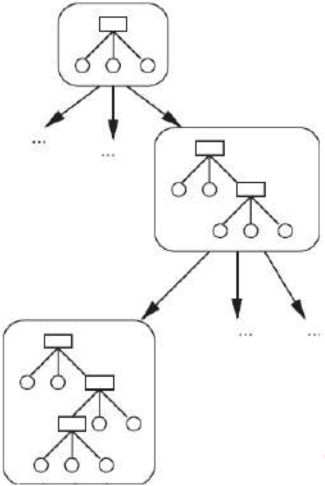
- It is powerful representation. Every discrete valued function can be represented by some decision tree.

Search for the best hypothesis: ID3



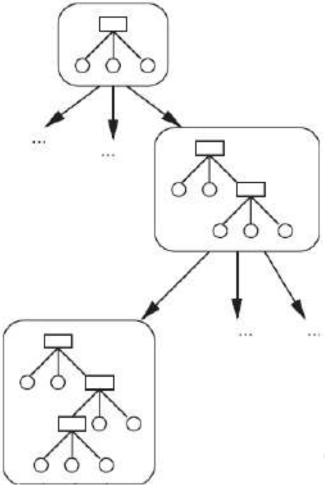
- It is powerful representation. Every discrete valued function can be represented by some decision tree.
- ID3 performs no backtracking. Once attribute is selected at certain level of tree, it never backtracks to reconsider choice (greedy algorithm approach).

Search for the best hypothesis: ID3



- It is powerful representation. Every discrete valued function can be represented by some decision tree.
- ID3 performs no backtracking. Once attribute is selected at certain level of tree, it never backtracks to reconsider choice (greedy algorithm approach).
- ID3 is characterized as searching a space of hypotheses (set of possible decision trees) for one that fits the training examples.

Search for the best hypothesis: ID3



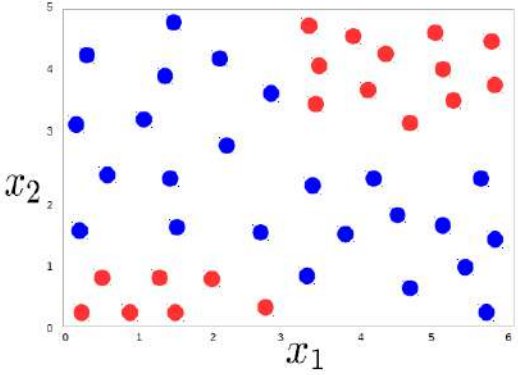
- It is powerful representation. Every discrete valued function can be represented by some decision tree.
- ID3 performs no backtracking. Once attribute is selected at certain level of tree, it never backtracks to reconsider choice (greedy algorithm approach).
- ID3 is characterized as searching a space of hypotheses (set of possible decision trees) for one that fits the training examples.

Bias

Which tree ID3 selects? Discussion on it later

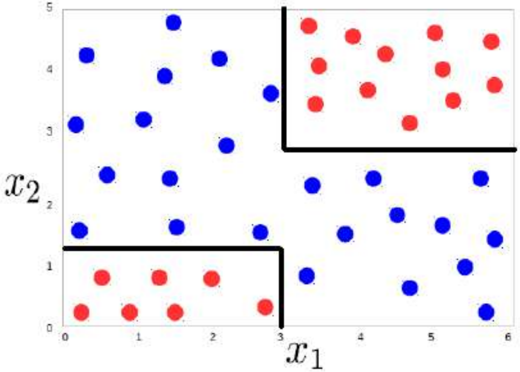
Example: Tree learning algorithm

Consider below presented classification (binary) problem. Assume training data with each instance / example having two attributes / features (attributes x_1, x_2):



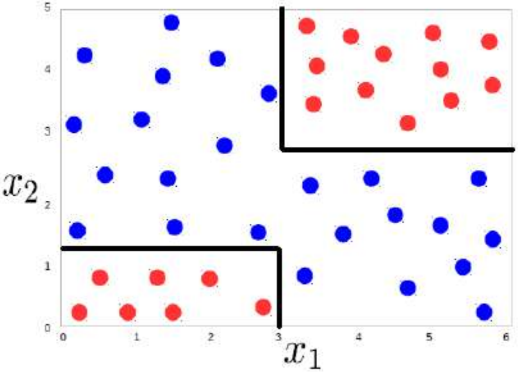
Example: Tree learning algorithm

The “expected” decision boundary given this training data.



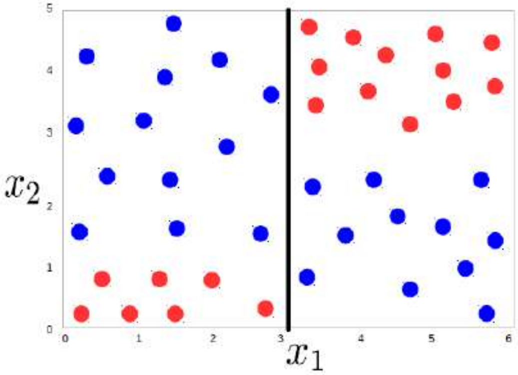
Example: Tree learning algorithm

The "expected" decision boundary given this training data. Learn it!



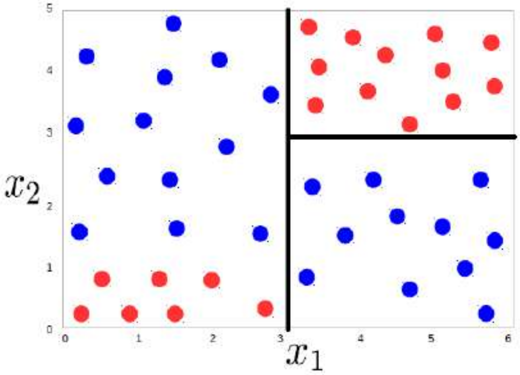
Example: Tree learning algorithm (attribute selection intuition)

Is x_1 (feature 1) greater than 3 ?



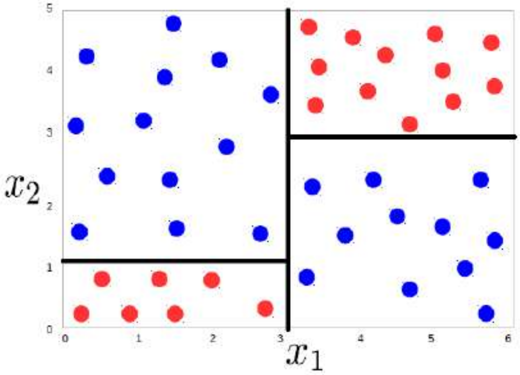
Example: Tree learning algorithm (attribute selection intuition)

Given $x_1 > 3$, is feature 2 (x_2) greater than 3?



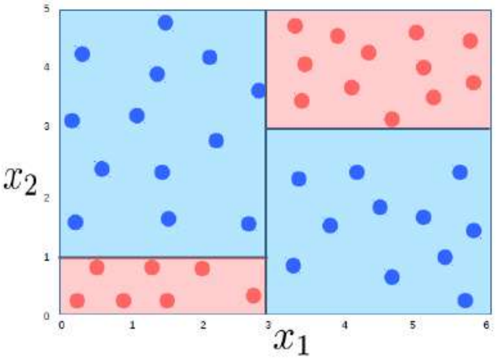
Example: Tree learning algorithm (attribute selection intuition)

Given $x_1 < 3$, is feature 2 (x_2) greater than 1?



Example: Tree learning algorithm (attribute selection intuition)

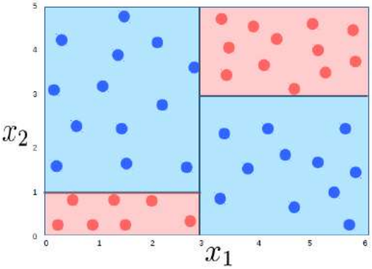
Feature space, learned decision boundary



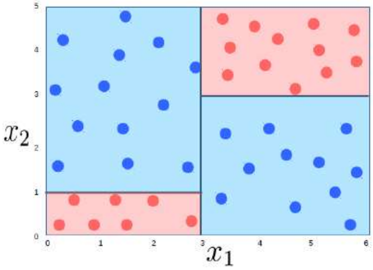
Exercise

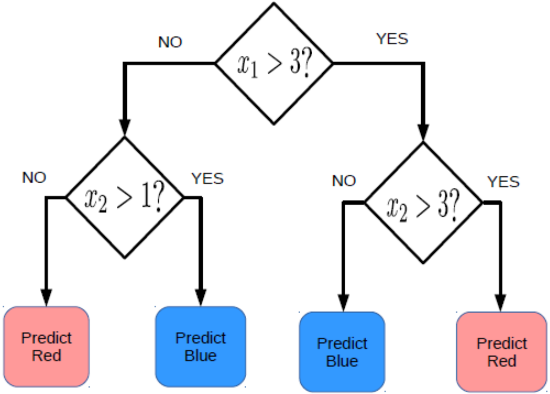
Can you draw corresponding decision tree?

Example solution: Tree learning algorithm

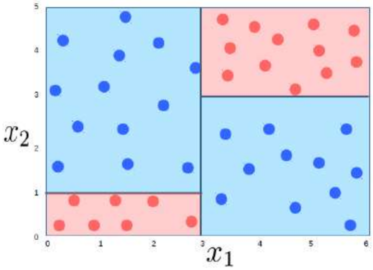


Example solution: Tree learning algorithm

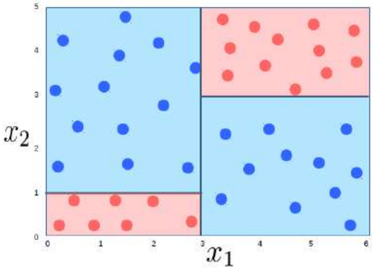


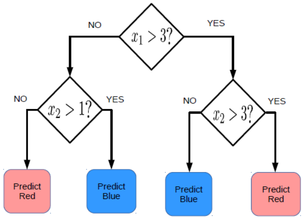


Example solution: Tree learning algorithm



Example solution: Tree learning algorithm





1. These rules perform recursive partitioning of training data into homogenous regions.
   - Homogeneous − > outputs are same / similar for all inputs in that region
2. Given a new test input, we can use the DT to predict its label
3. **A key benefit of DT: Prediction at test time is very fast (just testing a few conditions)**

Section Contents

ID3 Tree Induction / Learning Algorithm

Algorithm 1 ID3 Learning Algorithm

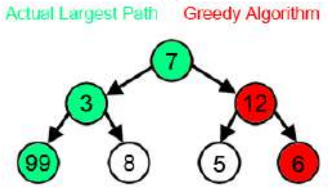
Result: Learned Tree

```
1 initialization node = root
2 while TRUE do
3     - A ← the best attribute for the next node.
       - Assign A as the decision attribute for the node.
       - For each value of A, create new decedent of the node.
       - Sort training examples to leaf nodes.

4     if training examples perfectly classified then
5         break
6     else
7         Iterate over new leaf node (back to line 2)
8     end
9 end
```

Greedy search

- ID3 algorithm forms a greedy search for an acceptable decision tree, in which the algorithm never backtracks to reconsider earlier choices.

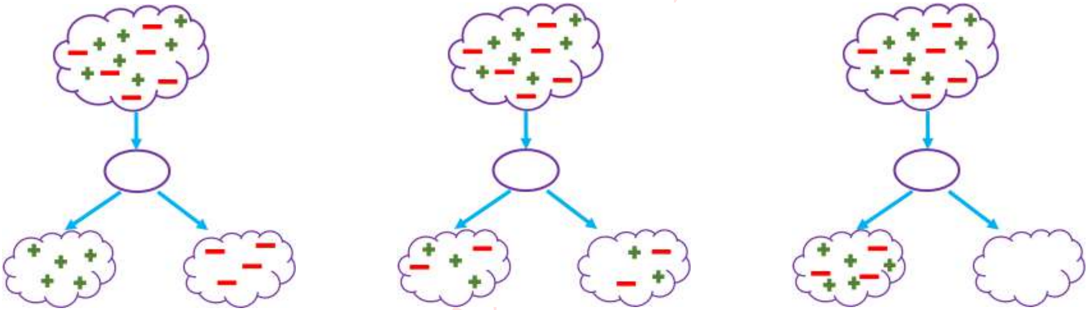
- Greedy is an algorithmic paradigm that builds up a solution piece by piece, always choosing the next piece that offers the most obvious and immediate benefit. So the problems where choosing locally optimal also leads to global solution are best fit for Greedy.



The greedy algorithm fails to solve above presented problem because it makes decisions purely based on what the best answer at the time is: at each step it did choose the largest number.

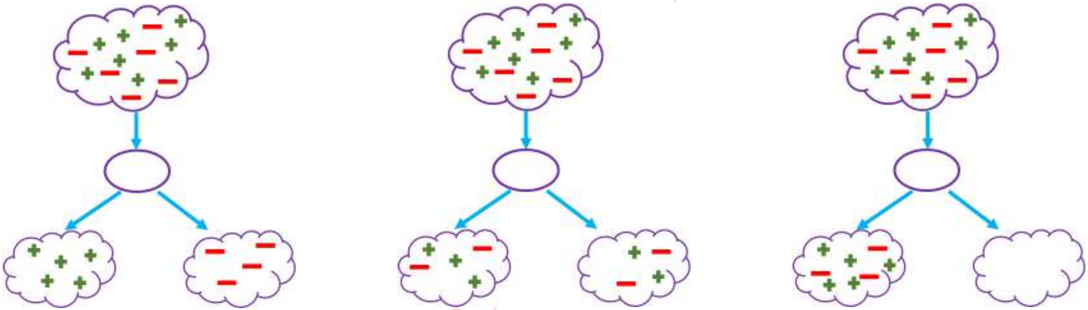
Best Attribute: Quiz

- Which “attribute” is the best?



Best Attribute: Quiz

- Which "attribute" is the best?



- Prefer splits that makes data "less randomized" after the split.

Best Attribute

Question

What is a good quantitative measure to evaluate effectiveness of an attribute for classification task?

Best Attribute

Question

What is a good quantitative measure to evaluate effectiveness of an attribute for classification task?

- Information gain measures how well a given attribute / feature separates the training examples according to their target classification.

Best Attribute

Question

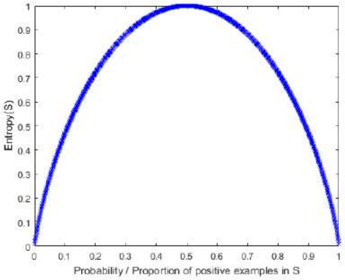
What is a good quantitative measure to evaluate effectiveness of an attribute for classification task?

- Information gain measures how well a given attribute / feature separates the training examples according to their target classification.
- ID3 uses information gain (entropy) to measure to select among the candidate attributes at each step while growing the tree.
- "Best attribute" is the one with lowest entropy or highest information gain.

Statistical measure: Entropy

Given a collection S, containing positive and negative examples of some target concept, the entropy of S relative to this Boolean classification is:

$$Entropy(S) = -p_{\oplus}\ log_2\, p_{\oplus} - p_{\otimes}\ log_2\, p_{\otimes} \qquad (1)$$



where:
- $p_{\oplus}$ is the proportion of positive examples in S.
- $p_{\otimes}$ is the proportion of negative examples in S.
- In all calculations involving entropy we define $0\ log_2\ 0$ to be 0.

  - Entropy[3] is measure of "randomness".

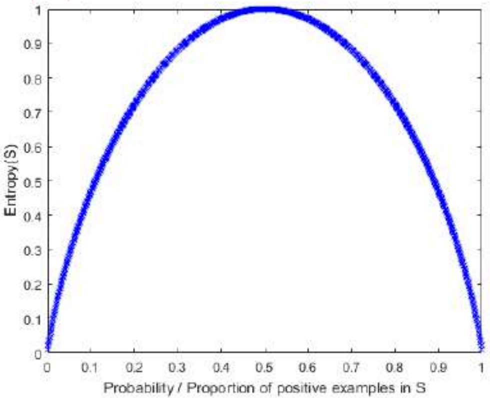
[3] C. E. SHANNON. A Mathematical Theory of Communication. The Bell System Technical Journal, 1948.

Statistical measure: Entropy

If the target attribute can take c different values (classes in our case):

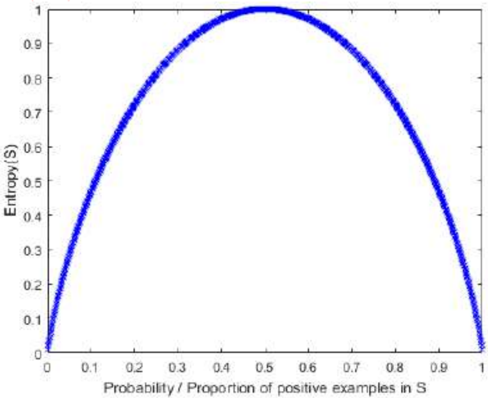
$$Entropy(S) = \sum_{i=1}^{c} -p_i \, log_2 \, p_i \qquad (2)$$

- Entropy is 0 if all members of S belong to the same class.
- The difference in the entropy before and after the split is called Information Gain (IG).



Statistical measure: Entropy

- Entropy characterizes the (im)purity of an arbitrary collection of examples. Entropy is commonly used in **Information theory** to measure amount of information needed to represent an event drawn from a probability distribution for a random variable. OR
- Its a measure of "disorder" or "randomness".
- **Thus, Entropy is 0 if all members of S belong to the same class.**



Statistical measure: Entropy Calculation

Calculation:

$$Entropy(S) = \sum_{i=1}^{c} -p_i \, log_2 \, p_i$$

Colour	Model	Mileage	Buy
Red	2011	10000	Yes
Red	2010	10000	Yes
Red	2010	60000	No
Yellow	2012	40000	Yes
Green	2015	10000	No

Dataset=1

Statistical measure: Entropy Calculation

Colour	Model	Mileage	Buy
Red	2011	10000	Yes
Red	2010	10000	Yes
Red	2010	60000	No
Yellow	2012	40000	Yes
Green	2015	10000	No

Dataset=1

Calculation:

$$Entropy(S) = \sum_{i=1}^{c} -p_i \, log_2 \, p_i$$

- c=2 , buy = Yes or No
- $[p^+, p^-] \Rightarrow [3^+, 2^-]$

Statistical measure: Entropy Calculation

Colour	Model	Mileage	Buy
Red	2011	10000	Yes
Red	2010	10000	Yes
Red	2010	60000	No
Yellow	2012	40000	Yes
Green	2015	10000	No

Dataset=1

Calculation:

$$Entropy(S) = \sum_{i=1}^{c} -p_i \, log_2 \, p_i$$

- c=2 , buy = Yes or No
- $[p^+, p^-] \Rightarrow [3^+, 2^-]$

$$\begin{aligned} Entropy(S) &= -\{\frac{3}{5} \, log_2 \, \frac{3}{5}\} - \{\frac{2}{5} \, log_2 \, \frac{2}{5}\} \\ Entropy(S) &= 0.4422 + 0.5288 \\ Entropy(S) &= 0.9710 \end{aligned} \tag{3}$$

Statistical measure: Entropy Calculation

Calculation:

$$Entropy(S) = \sum_{i=1}^{c} -p_i \, log_2 \, p_i$$

Colour	Model	Mileage	Buy
Red	2011	10000	Yes
Red	2010	10000	Yes
Red	2010	60000	No
Green	2015	10000	No

Dataset=2

Statistical measure: Entropy Calculation

Colour	Model	Mileage	Buy
Red	2011	10000	Yes
Red	2010	10000	Yes
Red	2010	60000	No
Green	2015	10000	No

Dataset=2

Calculation:

$$Entropy(S) = \sum_{i=1}^{c} -p_i \, log_2 \, p_i$$

- c=2 , buy = Yes or No
- $[p^+, p^-] \Rightarrow [2^+, 2^-]$

Statistical measure: Entropy Calculation

Colour	Model	Mileage	Buy
Red	2011	10000	Yes
Red	2010	10000	Yes
Red	2010	60000	No
Green	2015	10000	No

Dataset=2

Calculation:

$$Entropy(S) = \sum_{i=1}^{c} -p_i \, log_2 \, p_i$$

- c=2 , buy = Yes or No
- $[p^+, p^-] \Rightarrow [2^+, 2^-]$

$$\begin{aligned} Entropy(S) &= -\{\frac{2}{4} \, log_2 \, \frac{2}{4}\} - \{\frac{2}{4} \, log_2 \, \frac{2}{4}\} \\ Entropy(S) &= 0.5 + 0.5 \\ Entropy(S) &= 1 \end{aligned} \quad (4)$$

Statistical measure: Entropy Calculation

Calculation:

$$Entropy(S) = \sum_{i=1}^{c} -p_i \, log_2 \, p_i$$

Colour	Model	Mileage	Buy
Red	2011	10000	Yes
Red	2010	10000	Yes
Red	2010	60000	Yes
Yellow	2015	60000	Yes
Yellow	2015	10000	Yes

Dataset=3

Statistical measure: Entropy Calculation

Calculation:

$$Entropy(S) = \sum_{i=1}^{c} -p_i \, log_2 \, p_i$$

Colour	Model	Mileage	Buy
Red	2011	10000	Yes
Red	2010	10000	Yes
Red	2010	60000	Yes
Yellow	2015	60000	Yes
Yellow	2015	10000	Yes

Dataset=3

- c=2 , buy = Yes or No
- $[p^+, p^-] \Rightarrow [5^+, 0^-]$

Statistical measure: Entropy Calculation

Colour	Model	Mileage	Buy
Red	2011	10000	Yes
Red	2010	10000	Yes
Red	2010	60000	Yes
Yellow	2015	60000	Yes
Yellow	2015	10000	Yes

Dataset=3

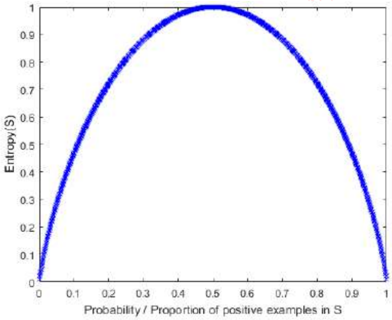
Calculation:

$$Entropy(S) = \sum_{i=1}^{c} -p_i \, log_2 \, p_i$$

- c=2 , buy = Yes or No
- $[p^+, p^-] \Rightarrow [5^+, 0^-]$

$$Entropy(S) = -\{\frac{5}{5} \, log_2 \, \frac{5}{5}\} - 0$$
$$Entropy(S) = 0 \tag{5}$$

Statistical measure: Entropy Calculation



- $[3^+, 2^-]$
- Entropy (S) $= 0.9710$

- $[2^+, 2^-]$
- Entropy (S) $= 1$

- $[5^+, 0^-]$
- Entropy (S) $= 0$

Entropy is measure of disorder or randomness!

Information Gain

- Given entropy as a measure of the impurity in a collection of training examples, we can now define a measure of the effectiveness of an attribute in classifying the training data. Generally, information gain (IG) is used as this measure of effectiveness.
- The difference in the entropy before and after the split is called information gain.

Information Gain

- Given entropy as a measure of the impurity in a collection of training examples, we can now define a measure of the effectiveness of an attribute in classifying the training data. Generally, information gain (IG) is used as this measure of effectiveness.
- The difference in the entropy before and after the split is called information gain.

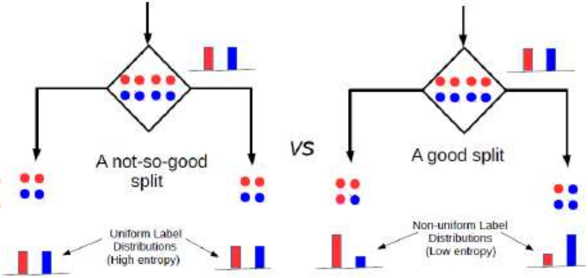
Mathematically:

$$Gain(S, A) = Entropy(S) - \sum_{v \in values(A)} \frac{|S_v|}{|S|} Entropy(S_v) \tag{6}$$

where:
- S set of training data, and
- $values(A)$ is the set of all possible values for attribute A (feature vector dimensions), and
- S_v is the subset of S for which attribute A has value v.

Information Gain

- For DT construction, entropy/IG gives us a criterion to select the best split.



Section Contents

Example Problem statement

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

*[4]

[4] Problem for Tom's book

Example Problem statement

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Problem Setting:

1. Set of possible instances X
   - each instance in X is defined by a feature vector, for example $< Outlook = rain, Humidity = low, Wind = weak, .. >$
   - $x =< x_1, x_2, \ldots, x_n >$
2. Unknown target function $f : X \rightarrow Y$
   - $Y = 1$ if we play tennis on specific day, else 0.
3. Set of function hypotheses $H = \{h|h : X \rightarrow Y\}$
   - each hypothesis h is a decision tree.
   - tree sorts x to leaf, which assigns y.

Example Problem statement

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

- Initially, we have 14 example $[9^+, 5^-]$.
- We need to calculate IG of all attributes to find which attribute is the best.
- List of attributes:

Example Problem statement

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

- Initially, we have 14 example $[9^+, 5^-]$.
- We need to calculate IG of all attributes to find which attribute is the best.
- List of attributes:
  1. Outlook
  2. Temperature
  3. Humidity
  4. Wind

DT Construction using IG Criterion: Outlook

$$Gain(S,A) = Entropy(S) - \sum_{v \in values(A)} \frac{|S_v|}{|S|} Entropy(S_v) \qquad Entropy(S) = \sum_{i=1}^{c} -p_i \log_2 p_i$$

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

- c=2 , play = Yes or No
- $[p^+, p^-] \Rightarrow [9^+, 5^-]$

DT Construction using IG Criterion: Outlook

$$Gain(S,A) = Entropy(S) - \sum_{v \in values(A)} \frac{|S_v|}{|S|} Entropy(S_v) \qquad Entropy(S) = \sum_{i=1}^{c} -p_i \log_2 p_i$$

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

- c=2 , play = Yes or No
- $[p^+, p^-] \Rightarrow [9^+, 5^-]$

$$Entropy(S) = -\{\frac{9}{14} \log_2 \frac{9}{14}\} - \{\frac{5}{14} \log_2 \frac{5}{14}\}$$

$$Entropy(S) = 0.4098 + 0.5305 = 0.9403$$

"Outlook" attribute has three values:

DT Construction using IG Criterion: Outlook

$$Gain(S,A) = Entropy(S) - \sum_{v \in values(A)} \frac{|S_v|}{|S|} Entropy(S_v) \qquad Entropy(S) = \sum_{i=1}^{c} -p_i \log_2 p_i$$

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

- c=2 , play = Yes or No
- $[p^+, p^-] \Rightarrow [9^+, 5^-]$

$$Entropy(S) = -\{\frac{9}{14}\, log_2 \frac{9}{14}\} - \{\frac{5}{14}\, log_2 \frac{5}{14}\}$$

$$Entropy(S) = 0.4098 + 0.5305 = 0.9403$$

“Outlook” attribute has three values:

1. Sunny $[2^+, 3^-]$
2. Overcast $[4^+, 0^-]$
3. Rain $[3^+, 2^-]$

Gain(S, Outlook) = ??

DT Construction using IG Criterion: Outlook

$$Gain(S, A) = Entropy(S) - \sum_{v \in values(A)} \frac{|S_v|}{|S|} Entropy(S_v) \qquad Entropy(S) = \sum_{i=1}^{c} -p_i \log_2 p_i$$

Gain(S, Outlook):

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

In "Outlook" attribute we have three values

a) Sunny $\{2^+, 3^-\}$

b) Overcast $\{4^+, 0^-\}$

c) Rain $\{3^+, 2^-\}$

$$Gain(S, Outlook) = 0.940 - \frac{5}{14} \times \left[-\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5} \right] - \frac{4}{14} \times \left[-\frac{4}{4}\log_2\frac{4}{4} \right] - \frac{5}{14} \times \left[-\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5} \right]$$

$$= 0.940 - 0.346 - 0 - 0.346$$

$$Gain(S, outlook) = 0.248$$

DT Construction using IG Criterion: Temperature

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

2) Gain (S, temperature)

Temperature has ~~two~~ three values

a) Hot $\{2^+, 2^-\}$

b) Mild $\{4^+, 2^-\}$

c) Cool $\{3^+, 1^-\}$

$$Gain(S, Temperature) = Entropy(S) - \sum_{v \in values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

$$= 0.940 - \frac{4}{14} \times \left[-\frac{2}{4}\log_2\frac{2}{4} - \frac{2}{4}\log_2\frac{2}{4}\right]$$

$$- \frac{6}{14} \times \left[-\frac{4}{6}\log_2\frac{4}{6} - \frac{2}{6}\log_2\frac{2}{6}\right]$$

$$- \frac{4}{14} \times \left[-\frac{3}{4}\log_2\frac{3}{4} - \frac{1}{4}\log_2\frac{1}{4}\right]$$

$$\Rightarrow 0.940 - 0.285 - 0.393 - 0.231$$

$$= \boxed{0.031} \rightarrow Gain(S, Temperature)$$

DT Construction using IG Criterion: Humidity

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

3) Gain(S, Humidity)

"Humidity" attribute has two values

a) High {3+, 4-}

b) Normal {6+, 1-}

$$\text{Gain}(S, \text{Humidity}) = 0.940 - \frac{7}{14} \times \left[-\frac{3}{7}\log_2\frac{3}{7} - \frac{4}{7}\log_2\frac{4}{7} \right] - \frac{7}{14} \times \left[-\frac{6}{7}\log_2\frac{6}{7} - \frac{1}{7}\log_2\frac{1}{7} \right]$$

$$= 0.940 - 0.492 - 0.295$$

$$\text{Gain}(S, \text{Humidity}) = 0.153$$

DT Construction using IG Criterion: Wind

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

4) Gain (S, Wind)

a) Weak $\{6^+, 2^-\}$

b) Strong $\{3^+, 3^-\}$

$$\text{Gain}(S, \text{wind}) = 0.940 - \frac{8}{14} \times \left[-\frac{6}{8}\log_2\frac{6}{8} - \frac{2}{8}\log_2\frac{2}{8} \right] - \frac{6}{14} \times \left[-\frac{3}{6}\log_2\frac{3}{6} - \frac{3}{6}\log_2\frac{3}{6} \right]$$

$$\text{Gain}(S, \text{wind}) = 0.048$$

Decision tree: first node decided

$Gain(S, Outlook) : 0.248$
$Gain(S, Temperature) : 0.031$
$Gain(S, Humidity) : 0.153$

$Gain(S, Wind) : 0.048$

Having decided which feature (highest IG) to test at the root, let's grow the tree

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

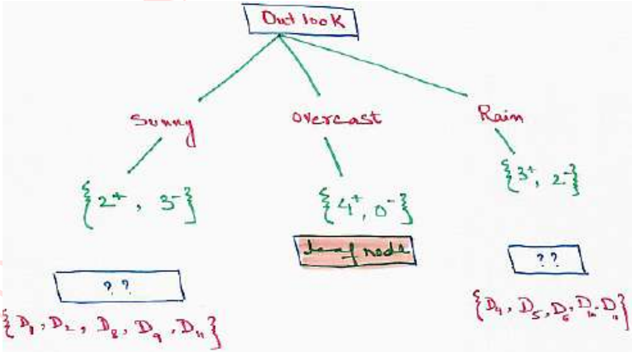
Decision tree: first node decided

$Gain(S, Outlook)$: 0.248
$Gain(S, Temperature)$: 0.031
$Gain(S, Humidity)$: 0.153

$Gain(S, Wind)$: 0.048

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

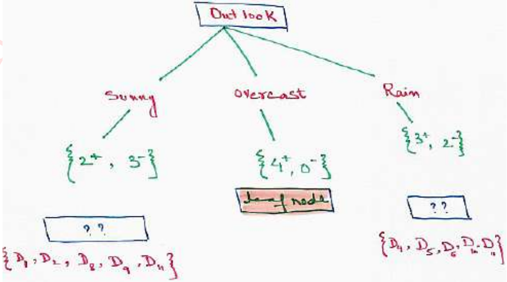
Having decided which feature (highest IG) to test at the root, let's grow the tree



DT Construction using IG Criterion: Second Node

- Iterate - for each child node, select the feature with the highest IG.
- No need to expand the middle node (already pure - all yes).
- If a feature has already been tested along a path earlier, we don't consider it again.

Day	Outlook	Temp.	Humidity	Wind	Play
D_1	Sunny	Hot	High	Weak	No
D_2	Sunny	Hot	High	Strong	No
D_8	Sunny	Mild	High	Weak	No
D_9	Sunny	Cool	Normal	Weak	Yes
D_{11}	Sunny	Mild	Normal	Strong	Yes



DT Construction using IG Criterion: Second Node

$$Gain(S,A) = Entropy(S) - \sum_{v \in values(A)} \frac{|S_v|}{|S|} Entropy(S_v) \qquad Entropy(S) = \sum_{i=1}^{c} -p_i \log_2 p_i$$

- c=2 , play = Yes or No
- $[p^+, p^-] \Rightarrow [2^+, 3^-]$

Day	Outlook	Temp.	Humidity	Wind	Play
D_1	Sunny	Hot	High	Weak	No
D_2	Sunny	Hot	High	Strong	No
D_8	Sunny	Mild	High	Weak	No
D_9	Sunny	Cool	Normal	Weak	Yes
D_{11}	Sunny	Mild	Normal	Strong	Yes

DT Construction using IG Criterion: Second Node

$$Gain(S,A) = Entropy(S) - \sum_{v \in values(A)} \frac{|S_v|}{|S|} Entropy(S_v) \qquad Entropy(S) = \sum_{i=1}^{c} -p_i \, log_2 \, p_i$$

- c=2 , play = Yes or No
- $[p^+, p^-] \Rightarrow [2^+, 3^-]$

Day	Outlook	Temp.	Humidity	Wind	Play
D_1	Sunny	Hot	High	Weak	No
D_2	Sunny	Hot	High	Strong	No
D_8	Sunny	Mild	High	Weak	No
D_9	Sunny	Cool	Normal	Weak	Yes
D_{11}	Sunny	Mild	Normal	Strong	Yes

$$Entropy(S) = -\{\frac{2}{5}\, log_2\, \frac{2}{5}\} - \{\frac{3}{5}\, log_2\, \frac{3}{5}\}$$
$$Entropy(S) = 0.5288 + 0.4422 = 0.9710 \qquad (8)$$

“Temperature” attribute has three values:

DT Construction using IG Criterion: Second Node

$$Gain(S,A) = Entropy(S) - \sum_{v \in values(A)} \frac{|S_v|}{|S|} Entropy(S_v) \qquad Entropy(S) = \sum_{i=1}^{c} -p_i \log_2 p_i$$

- c=2 , play = Yes or No
- $[p^+, p^-] \Rightarrow [2^+, 3^-]$

Day	Outlook	Temp.	Humidity	Wind	Play
D_1	Sunny	Hot	High	Weak	No
D_2	Sunny	Hot	High	Strong	No
D_8	Sunny	Mild	High	Weak	No
D_9	Sunny	Cool	Normal	Weak	Yes
D_{11}	Sunny	Mild	Normal	Strong	Yes

$$Entropy(S) = -\{\frac{2}{5} \log_2 \frac{2}{5}\} - \{\frac{3}{5} \log_2 \frac{3}{5}\} \tag{8}$$
$$Entropy(S) = 0.5288 + 0.4422 = 0.9710$$

"Temperature" attribute has three values:

1. Hot $[0^+, 2^-]$
2. Mild $[1^+, 1^-]$
3. Cold $[1^+, 0^-]$

Gain(S_{Sunny}, Temperature) = ??

DT Construction using IG Criterion: Second Node

$$Gain(S, A) = Entropy(S) - \sum_{v \in values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

$$Entropy(S) = \sum_{i=1}^{c} -p_i \log_2 p_i$$

Gain(S_{Sunny}, **Temperature**) = ??

Day	Outlook	Temp.	Humidity	Wind	Play
D_1	Sunny	Hot	High	Weak	No
D_2	Sunny	Hot	High	Strong	No
D_8	Sunny	Mild	High	Weak	No
D_9	Sunny	Cool	Normal	Weak	Yes
D_{11}	Sunny	Mild	Normal	Strong	Yes

DT Construction using IG Criterion: Second Node

$$Gain(S,A) = Entropy(S) - \sum_{v \in values(A)} \frac{|S_v|}{|S|} Entropy(S_v) \qquad Entropy(S) = \sum_{i=1}^{c} -p_i \log_2 p_i$$

Gain(S_{Sunny}, Temperature) = ??

Day	Outlook	Temp.	Humidity	Wind	Play
D_1	Sunny	Hot	High	Weak	No
D_2	Sunny	Hot	High	Strong	No
D_8	Sunny	Mild	High	Weak	No
D_9	Sunny	Cool	Normal	Weak	Yes
D_{11}	Sunny	Mild	Normal	Strong	Yes

① Gain(S_{Sunny}, Temp)

Hot = $\{0^+, 2^-\}$
Mild = $\{1^+, 1^-\}$
Cool = $\{1^+, 0\}$

$$= 0.971 - \frac{2}{5} \times \left[0 - \frac{2}{2}\log_2 \frac{2}{2}\right] - \frac{2}{5} \times \left[-\frac{1}{2}\log_2 \frac{1}{2} - \frac{1}{2}\log_2 \frac{1}{2}\right] - \frac{1}{5} \times \left[\frac{1}{1}\log_2 1\right]$$

$$= 0.971 - 0 - 0.40 - 0 = \boxed{0.5710}$$

DT Construction using IG Criterion: Second Node

$$Gain(S, A) = Entropy(S) - \sum_{v \in values(A)} \frac{|S_v|}{|S|} Entropy(S_v) \qquad Entropy(S) = \sum_{i=1}^{c} -p_i \log_2 p_i$$

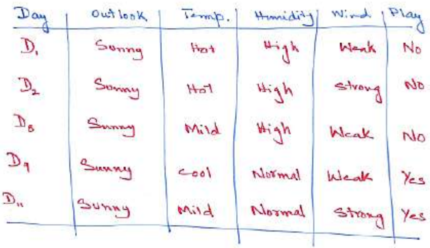
Day	Outlook	Temp.	Humidity	Wind	Play
D_1	Sunny	Hot	High	Weak	No
D_2	Sunny	Hot	High	Strong	No
D_8	Sunny	Mild	High	Weak	No
D_9	Sunny	Cool	Normal	Weak	Yes
D_{11}	Sunny	Mild	Normal	Strong	Yes

Gain(S_{sunny}, Humidity) = ??

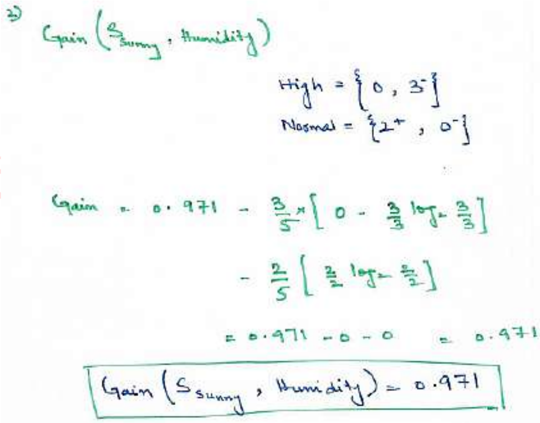
1. High $[0^+, 3^-]$
2. Normal $[2^+, 0^-]$

DT Construction using IG Criterion: Second Node

$$Gain(S, A) = Entropy(S) - \sum_{v \in values(A)} \frac{|S_v|}{|S|} Entropy(S_v) \qquad Entropy(S) = \sum_{i=1}^{c} -p_i \log_2 p_i$$



Day	Outlook	Temp.	Humidity	Wind	Play
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes



Gain(S_{sunny}, Humidity) = ??

1. High $[0^+, 3^-]$
2. Normal $[2^+, 0^-]$

DT Construction using IG Criterion: Second Node

$$Gain(S,A) = Entropy(S) - \sum_{v \in values(A)} \frac{|S_v|}{|S|} Entropy(S_v) \qquad Entropy(S) = \sum_{i=1}^{c} -p_i \log_2 p_i$$

Day	Outlook	Temp.	Humidity	Wind	Play
D_1	Sunny	Hot	High	Weak	No
D_2	Sunny	Hot	High	Strong	No
D_8	Sunny	Mild	High	Weak	No
D_9	Sunny	Cool	Normal	Weak	Yes
D_{11}	Sunny	Mild	Normal	Strong	Yes

Gain(S_{sunny}, Wind) $= ??$

1. Weak $[1^+, 2^-]$
2. Strong $[1^+, 1^-]$

DT Construction using IG Criterion: Second Node

$$Gain(S, A) = Entropy(S) - \sum_{v \in values(A)} \frac{|S_v|}{|S|} Entropy(S_v) \qquad Entropy(S) = \sum_{i=1}^{c} -p_i \log_2 p_i$$

Day	Outlook	Temp.	Humidity	Wind	Play
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes

Gain(S_{sunny}, Wind) $= ??$

1. Weak $[1^+, 2^-]$
2. Strong $[1^+, 1^-]$

3) Gain(S_{sunny}, Wind)

Weak = $\{1^+, 2^-\}$

Strong = $\{1^+, 1^-\}$

$$Gain = 0.971 - \frac{3}{5} \times \left[-\frac{1}{3}\log_2\frac{1}{3} - \frac{2}{3}\log_2\frac{2}{3} \right] - \frac{2}{5} \times \left[-\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2} \right]$$

$$= 0.971 - 0.551 - 0.4 = 0.02$$

Gain(S_{sunny}, Wind) = 0.02

Decision tree: second node decided

Lets grow tree!

Day	Outlook	Temp.	Humidity	Wind	Play
D_1	Sunny	Hot	High	Weak	No
D_2	Sunny	Hot	High	Strong	No
D_8	Sunny	Mild	High	Weak	No
D_9	Sunny	Cool	Normal	Weak	Yes
D_{11}	Sunny	Mild	Normal	Strong	Yes

Gain(S_{Sunny}, Temperature) $= 0.571$

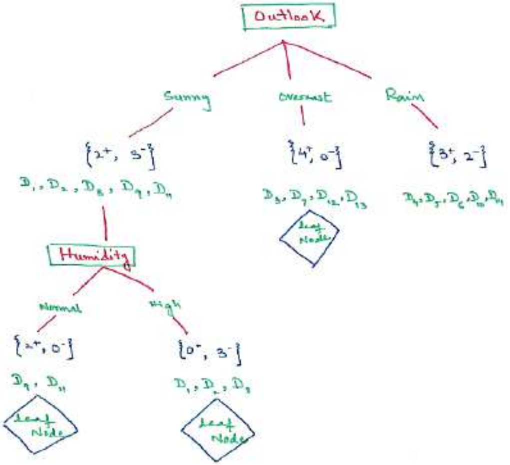
Gain(S_{sunny}, Humidity) $= 0.971$

Gain(S_{sunny}, Wind) $= 0.02$

Decision tree: second node decided

Lets grow tree!

Day	Outlook	Temp.	Humidity	Wind	Play
D_1	Sunny	Hot	High	Weak	No
D_2	Sunny	Hot	High	Strong	No
D_8	Sunny	Mild	High	Weak	No
D_9	Sunny	Cool	Normal	Weak	Yes
D_{11}	Sunny	Mild	Normal	Strong	Yes

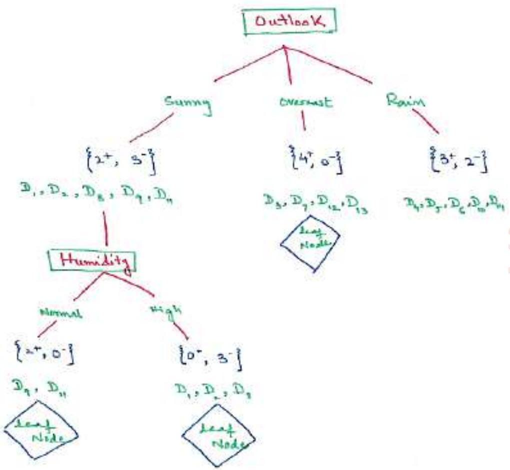


$\text{Gain}(S_{Sunny}, \text{Temperature}) = 0.571$

$\text{Gain}(S_{sunny}, \text{Humidity}) = 0.971$

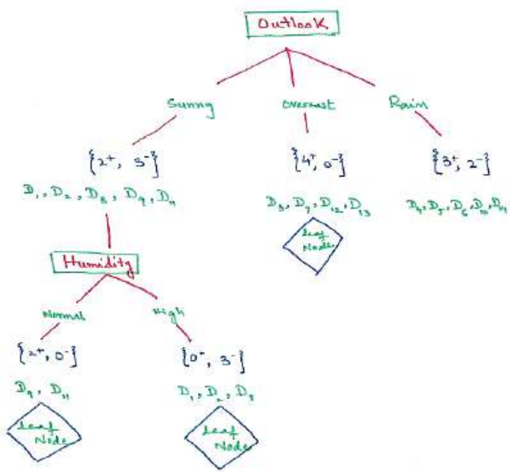
$\text{Gain}(S_{sunny}, \text{Wind}) = 0.02$

DT Construction using IG Criterion: Third Node



What to do next?

DT Construction using IG Criterion: Third Node



What to do next?

Calculate Gain for remaining attributes:

Gain(S_{Rain}, Wind)

Gain(S_{Rain}, Temperature)

Day	Outlook	Temperature	Humidity	Wind	Play
D_4	Rain	Mild	High	Weak	Yes
D_5	Rain	Cool	Normal	Weak	Yes
D_6	Rain	Cool	Normal	Strong	No
D_{10}	Rain	Mild	Normal	Weak	Yes
D_{14}	Rain	Mild	High	Strong	No

Trained classification tree

- c=2 , play = Yes or No
- $[p^+, p^-] \Rightarrow [3^+, 2^-]$

Trained classification tree

- c=2 , play = Yes or No
- $[p^+, p^-] \Rightarrow [3^+, 2^-]$

$$Entropy(S) = -\{\frac{3}{5}\, log_2\, \frac{3}{5}\} - \{\frac{2}{5}\, log_2\, \frac{2}{5}\}$$
$$Entropy(S) = 0.4422 + 0.5288 = 0.9710 \tag{9}$$

Trained classification tree

- c=2 , play = Yes or No
- $[p^+, p^-] \Rightarrow [3^+, 2^-]$

$$Entropy(S) = -\{\frac{3}{5}\, log_2\, \frac{3}{5}\} - \{\frac{2}{5}\, log_2\, \frac{2}{5}\}$$
$$Entropy(S) = 0.4422 + 0.5288 = 0.9710 \tag{9}$$

$$Gain(S_{Rain}, Wind) = 0.971 - \frac{3}{5} \times \{\frac{3}{5}\, log_2\, \frac{3}{5} - 0\} - \frac{2}{5} \times \{0 - \frac{2}{5}\, log_2\, \frac{2}{5}\} \tag{10}$$

Gain(S_{Rain}, Wind)=0.971

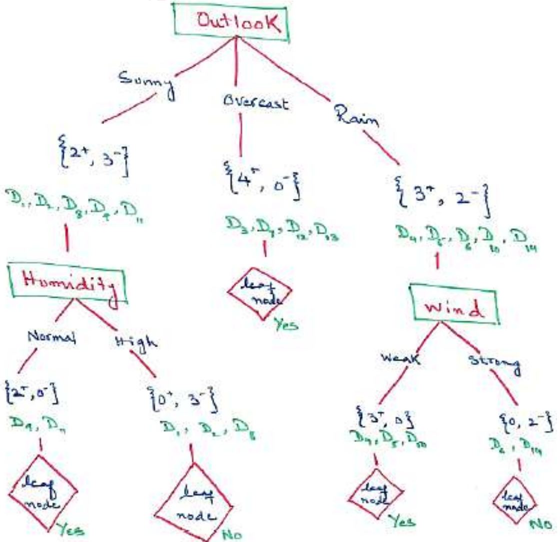
Trained classification tree

- c=2 , play = Yes or No
- $[p^+, p^-] \Rightarrow [3^+, 2^-]$

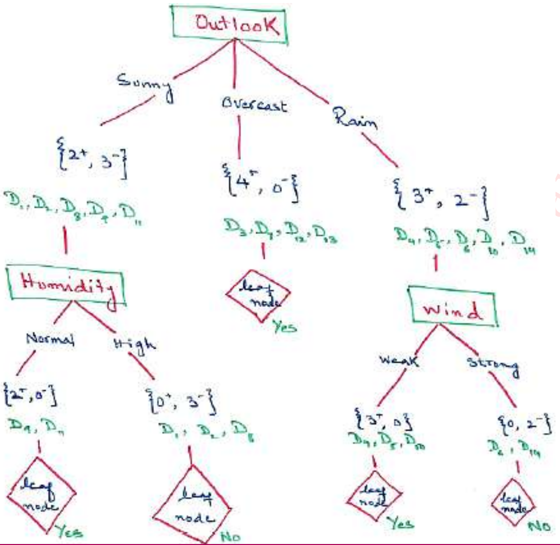
$$Entropy(S) = -\{\frac{3}{5}\, log_2\, \frac{3}{5}\} - \{\frac{2}{5}\, log_2\, \frac{2}{5}\}$$
$$Entropy(S) = 0.4422 + 0.5288 = 0.9710 \quad (9)$$

$$Gain(S_{Rain}, Wind) = 0.971 - \frac{3}{5} \times \{\frac{3}{5}\, log_2\, \frac{3}{5} - 0\} - \frac{2}{5} \times \{0 - \frac{2}{5}\, log_2\, \frac{2}{5}\} \quad (10)$$

Gain(S_{Rain}, Wind)=0.971



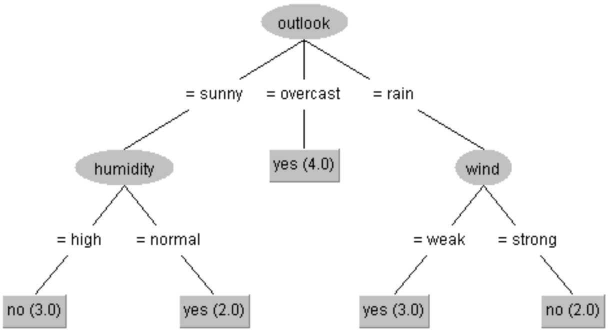
Function Approximation



- We began problem with unknown target function $f : X \rightarrow Y$ that classifies whether to play tennis or not on a given day X.
- Now we have decision tree for $f :< Outlook, Temperature, Humidity, Wind > \rightarrow Y$.

Section Contents

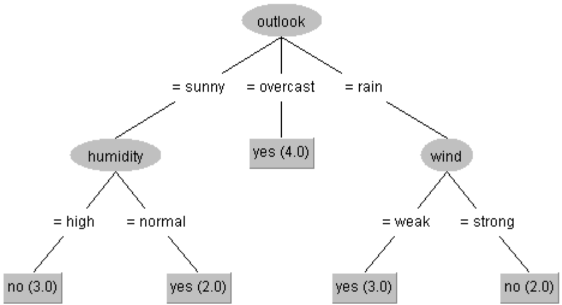
Play Tennis: Weka

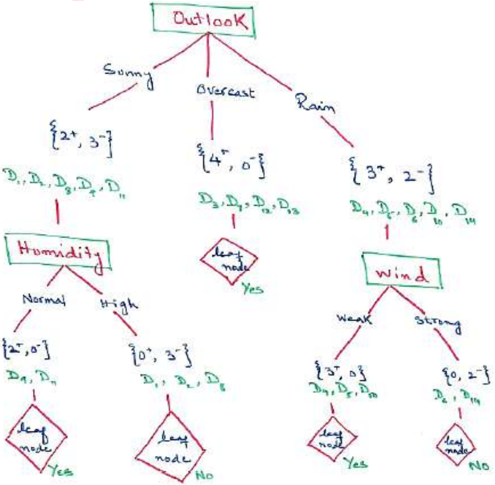


*5

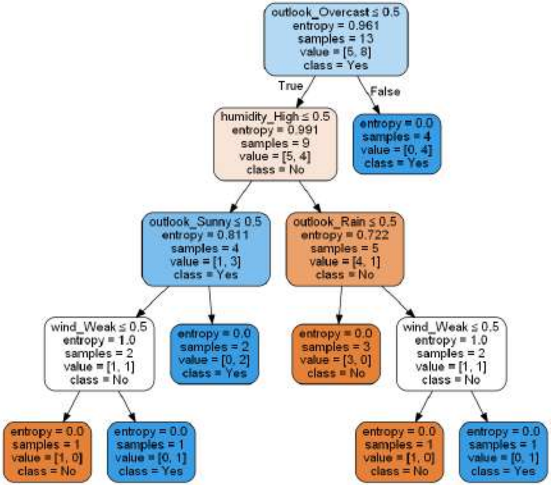
5 ID3 Algorithm

Play Tennis: Weka





Play Tennis: Python



*6

- CART stands for Classification and Regression Trees.
- "scikit-learn" uses an optimized version of the CART algorithm.
- CART constructs binary trees (twoing criteria).
- Unlike ID3, it uses pruning to avoid over-fitting.

[6] Result obtained with CART algorithm

Decision Tree: Python

```python

import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.tree import export_graphviz
from sklearn.preprocessing import OneHotEncoder
from IPython.display import Image
from sklearn.tree import export_graphviz
from pydotplus import graph_from_dot_data

df = pd.read_csv('play_tennis.csv')

# Before we do anything we'll want to split our data into training and test
    sets.
# We'll accomplish this by first splitting the DataFrame into features (X) and
# target (y), then passing X and y to the train_test_split() function to
# split the data so that 70% of it is in the training set, and 30% of
# it is in the testing set.

# loc() function is used to access a group of rows and columns by label(s) or
```

Decision Tree: Python

```
# loc() function is used to access a group of rows and columns by label(s) or
    a boolean array

X = df.loc[:, ['outlook', 'temp', 'humidity', 'wind']]
y = df.loc[:, 'play']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.05,
                                                    random_state = 42)

#Encode categorical data as numbers
#Since all of our data is currently categorical (recall that each column is
#in string format), we need to encode them as numbers. For this,
# we'll use a handy helper object from sklearn's preprocessing module
# called OneHotEncoder.
# One-hot encode the training data and show the resulting
# DataFrame with proper column names
ohe = OneHotEncoder()
ohe.fit(X_train)
X_train_ohe = ohe.transform(X_train).toarray()

# Creating this DataFrame is not necessary its only to show the result of the
    ohe
```

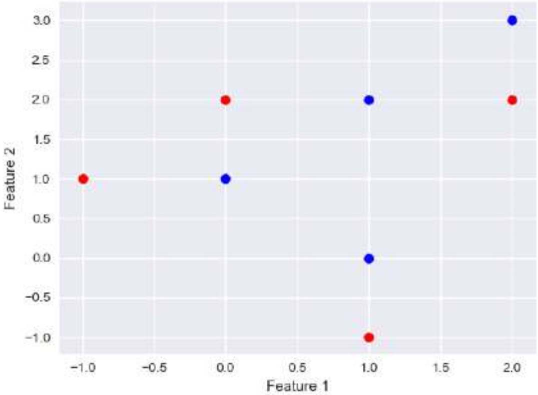
Decision Tree: Python

```
# Creating this DataFrame is not necessary its only to show the result of the
    ohe
ohe_df = pd.DataFrame(X_train_ohe,
                      columns=ohe.get_feature_names(X_train.columns))

# Create the classifier, fit it on the training data and make predictions on
    the test set
clf = DecisionTreeClassifier(criterion='entropy')
clf.fit(X_train_ohe, y_train)
#DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=None
    ,
#                       max_features=None, max_leaf_nodes=None,
#                       min_impurity_decrease=0.0, min_impurity_split=None,
#                       min_samples_leaf=1, min_samples_split=2,
#                       min_weight_fraction_leaf=0.0, presort=False,
#                       random_state=None, splitter='best')
#Plot the decision tree
#You can see what rules the tree learned by plotting this decision tree.
#To do this, you need to use additional packages such as pytdotplus

#Note: If you are run into errors while generating the plot,
# you probably need to install python-graphviz in your machine
```

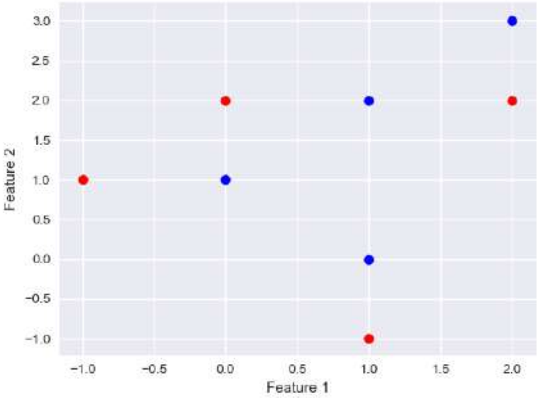
Decision Tree: Python

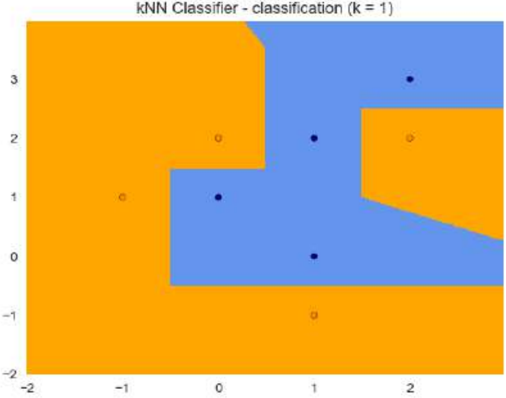
```
# Create DOT data
dot_data = export_graphviz(clf, out_file=None,
                           feature_names=ohe_df.columns,
                           class_names=np.unique(y).astype('str'),
                           filled=True, rounded=True, special_characters=True)

# Draw graph
graph = graph_from_dot_data(dot_data)

# Show graph
Image(graph.create_png())



X_test_ohe = ohe.transform(X_test)
y_preds = clf.predict(X_test_ohe)

print('Accuracy: ', accuracy_score(y_test, y_preds))
```

Ocular Proof

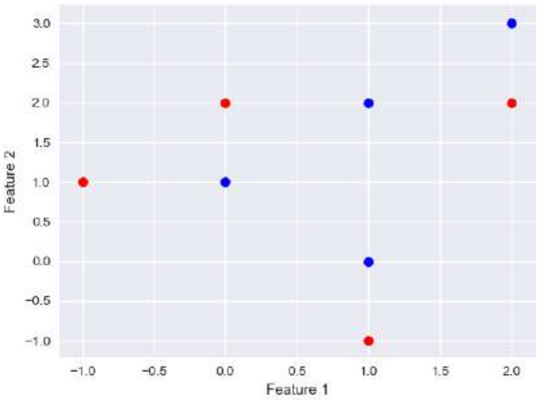


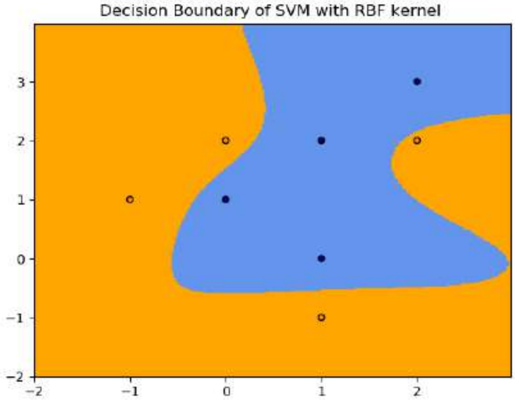
Ocular Proof



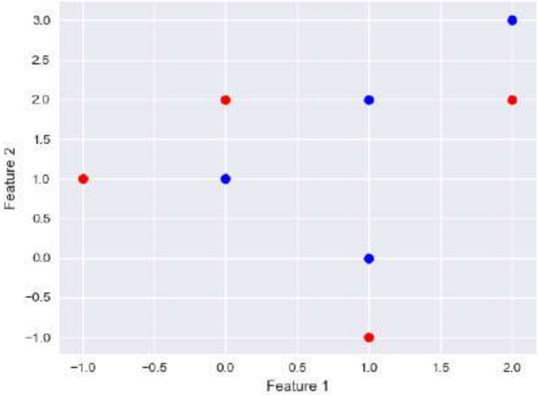


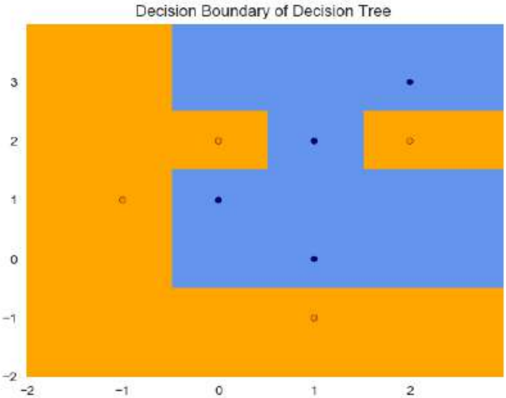
Ocular Proof





Ocular Proof

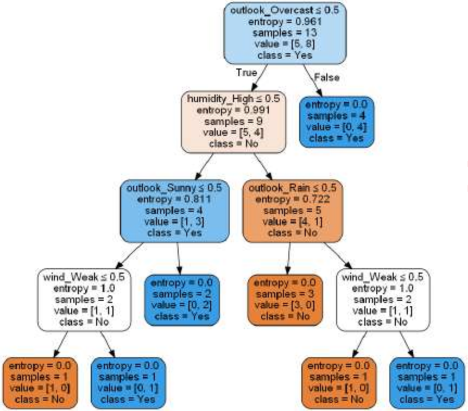




Section Contents

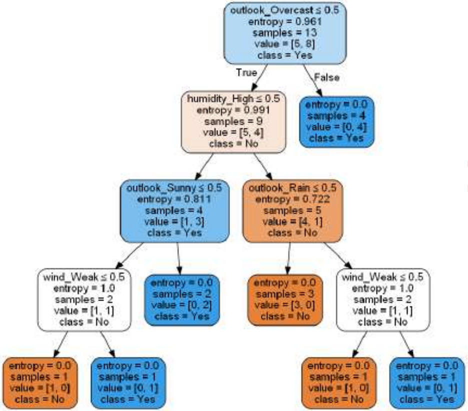
Play Tennis: Python - Changing Statistical Test

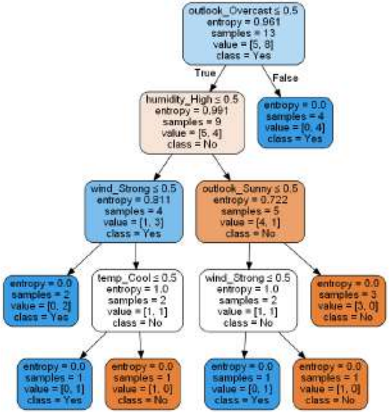
$$Entropy(S) = \sum_{i=1}^{c} -p_i \, log_2 \, p_i$$



Play Tennis: Python - Changing Statistical Test

$$Entropy(S) = \sum_{i=1}^{c} -p_i \, log_2 \, p_i \qquad\qquad Gini(S) = 1 - \sum_{i=1}^{n} (p_i)^2 \tag{11}$$



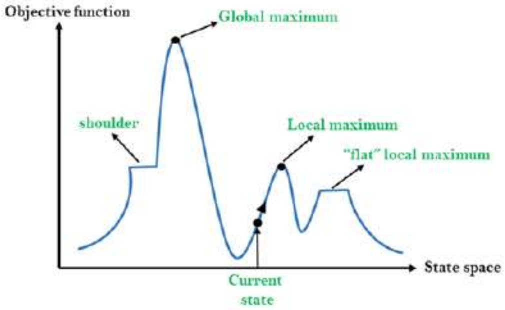


Play Tennis: Python - Changing Statistical Test

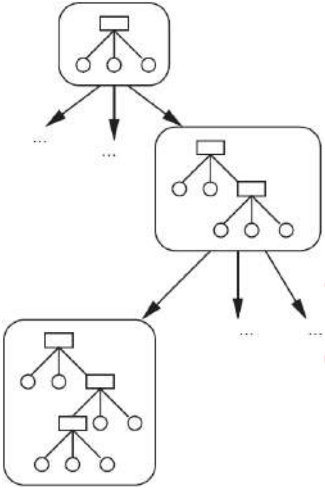
Please read more on different splitting measure / statistical test to understand which one suits which type of datasets and what are benefits and drawbacks for different criteria.

Inductive Bias of Learning Algorithm: ID3

- As with other inductive learning methods, ID3 can be characterized as searching a space of hypotheses (set of possible decision trees) for one that fits the training examples.
- ID3 performs hill-climbing search through hypothesis space.
    1. Hill climbing algorithm is a technique which is used for optimizing the mathematical problems i.e. Traveling Salesman Problem (TSP).
    2. It is also called greedy local search as it only looks to its good immediate neighbor state and not beyond that.
    3. It does not backtrack the search space, as it does not remember the previous states.

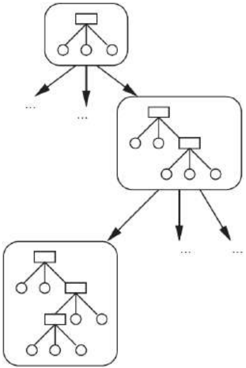


Inductive Bias of Learning Algorithm: ID3



- The evaluation function that guides this hill-climbing search is information gain.
- By looking at the figure, we can get insight into capabilities and limitation of ID3 in terms of search space and search strategy.

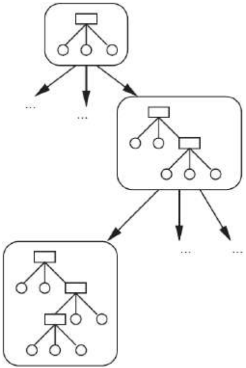
Inductive Bias of Learning Algorithm: ID3



- Every discrete valued function can be represented by some decision tree.
- ID3 performs no backtracking. Once attribute is selected at certain level of tree, it never backtracks to reconsider choice.
- ID3 is characterized as searching a space of hypotheses (set of possible decision trees) for one that fits the training examples.

Which tree ID3 selects?

Inductive Bias of Learning Algorithm: ID3



- Every discrete valued function can be represented by some decision tree.
- ID3 performs no backtracking. Once attribute is selected at certain level of tree, it never backtracks to reconsider choice.
- ID3 is characterized as searching a space of hypotheses (set of possible decision trees) for one that fits the training examples.

Which tree ID3 selects?

It chooses first acceptable tree it encounters in hill climbing (greedy) strategy (placing attribute with highest information gain closest to the root), thus favoring shorter trees.

Inductive Bias of Learning Algorithm: Occam's razor

Occam's Razor

- Is ID3's inductive bias favoring shorter trees a sound basis for generalization?
- Philosophers and Scientists have debated this question for centuries. William of Occam (or William of Ockham, Ockham was the village in the English county of Surrey) was one of the first to discuss this, so this bias often goes by the name of Occam's razor.

Inductive Bias of Learning Algorithm: Occam's razor

Occam's Razor

- Is ID3's inductive bias favoring shorter trees a sound basis for generalization?
- Philosophers and Scientists have debated this question for centuries. William of Occam (or William of Ockham, Ockham was the village in the English county of Surrey) was one of the first to discuss this, so this bias often goes by the name of Occam's razor.

Read more

Are shorter / simpler explanation always correct? Do read more and find issues with Occam's razor.

Stopping Criteria / how deeply to grow tree?

Add Noise / just one example / feature vector:

$< Outlook = Sunny, Temperature = Hot, Humidity = Normal, Wind = Strong, PlayTennis = No >$

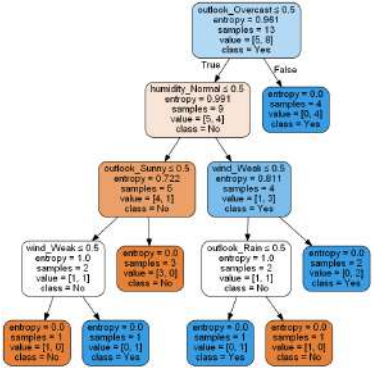
- Right side tree has added another level to cater for one (noise) example.

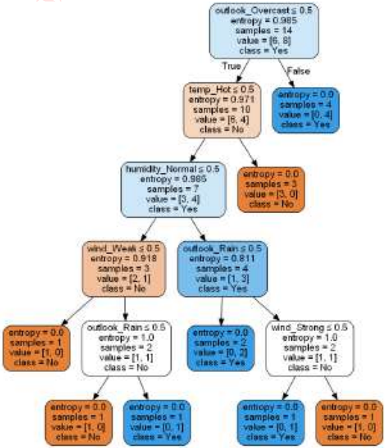
Stopping Criteria / how deeply to grow tree?

Add Noise / just one example / feature vector:

$< Outlook = Sunny, Temperature = Hot, Humidity = Normal, Wind = Strong, PlayTennis = No >$

- Right side tree has added another level to cater for one (noise) example.





Overfitting

ID3 grows deeply enough to perfectly classify all training examples. This leads to problem when there is noise in the data (refer previous slide). This also highlights the problem of overfitting training data.

Overfitting

ID3 grows deeply enough to perfectly classify all training examples. This leads to problem when there is noise in the data (refer previous slide). This also highlights the problem of overfitting training data.

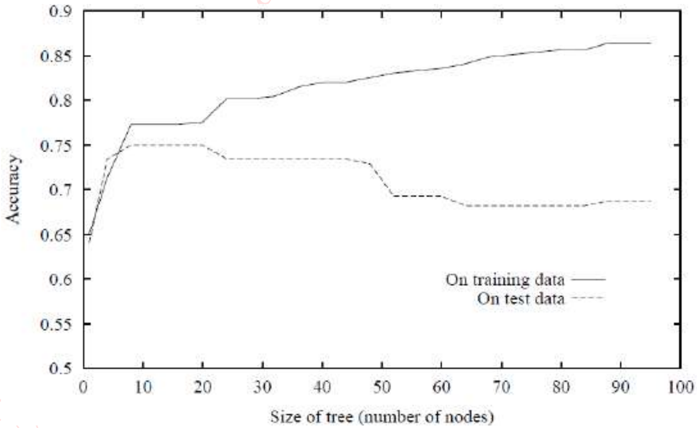
Overfit

Given a hypothesis space H, a hypothesis $h \in H$ is said to overfit the training data if there exists alternative hypothesis $h' \in H$, such that h has smaller error than h' over training examples but h' has smaller error than h over entire distribution of instances.

Overfitting

Overfit

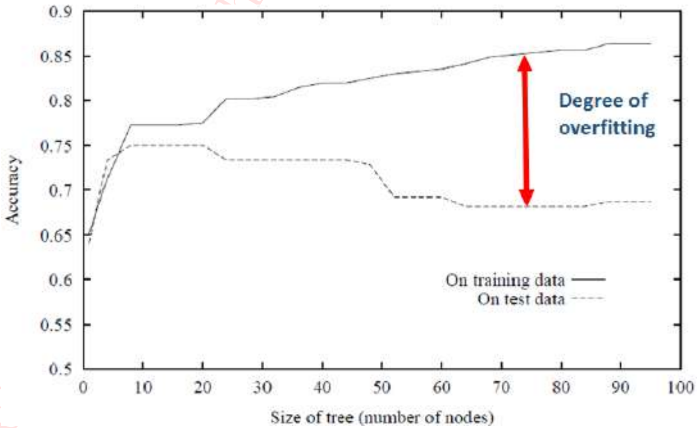
Given a hypothesis space H, a hypothesis $h \in H$ is said to overfit the training data if there exists alternative hypothesis $h' \in H$, such that h has smaller error than h' over training examples but h' has smaller error than h over entire distribution of instances.



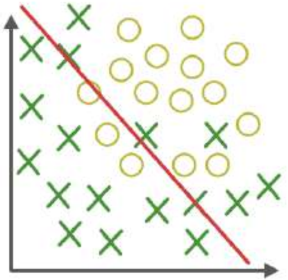
Overfitting

Overfit

Given a hypothesis space H, a hypothesis $h \in H$ is said to overfit the training data if there exists alternative hypothesis $h' \in H$, such that h has smaller error than h' over training examples but h' has smaller error than h over entire distribution of instances.

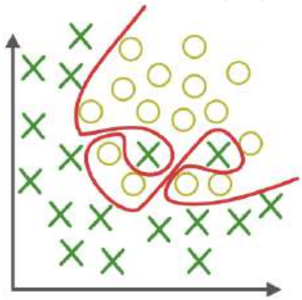


Overfitting Vs Underfitting



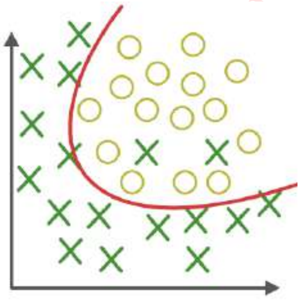
Is this a good fit?

Overfitting Vs Underfitting



Is this a good fit?

Overfitting Vs Underfitting



Is this a good fit?

- Overfitting happens when a model memorizes its training data so well that it is learning noise on top of the signal

How to avoid overfitting in Decision Tree?

Two approaches:

- Stop growing when data split not statistically significant.
- Grow full tree, then post-prune.

How to avoid overfitting in Decision Tree?

Two approaches:

- Stop growing when data split not statistically significant.
- Grow full tree, then post-prune.

Pruning

Pruning reduces the size of decision trees by removing parts of the tree that do not provide statistical significance to classify instances.

Reduced error pruning

- Split data into training and validation set
- Do until further pruning is harmful:
  1. Evaluate impact on validation set of pruning each possible node (plus those below it).
  2. Greedily remove the one that most improves validation set accuracy.
- Produces smallest version of most accurate subtree.

Reduced error pruning

- Split data into training and validation set
- Do until further pruning is harmful:
  1. Evaluate impact on validation set of pruning each possible node (plus those below it).
  2. Greedily remove the one that most improves validation set accuracy.
- Produces smallest version of most accurate subtree.

How to avoid overfitting in Decision Tree?

How to select best tree:

- Measure performance over training data
- Measure performance over separate validation data set

Machine Learning
Support Vector Machines

Dr. Rizwan Ahmed Khan

Outline

Section Contents

Reference Books

Reference books for this Module:

- Chapter 3: Pattern Recognition, S. Theodoridis et al.,Academic Press, 4^{th} or latest edition.

Reference Books

Reference books for this Module:

- Chapter 3: Pattern Recognition, S. Theodoridis et al.,Academic Press, 4^{th} or latest edition.
- Chapter 6 & 7: Pattern Recognition and Machine Learning, Christopher M. Bishop, Springer Books, latest edition.

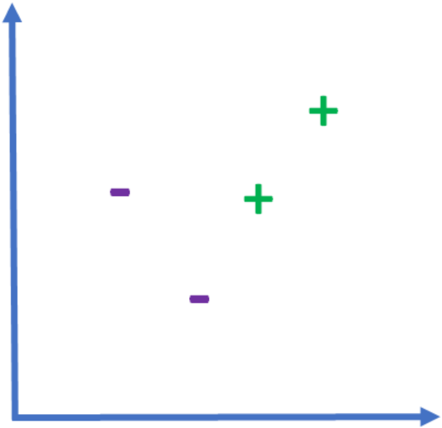
Reference Books

Reference books for this Module:

- Chapter 3: Pattern Recognition, S. Theodoridis et al.,Academic Press, 4^{th} or latest edition.
- Chapter 6 & 7: Pattern Recognition and Machine Learning, Christopher M. Bishop, Springer Books, latest edition.
- Book Support Vector Machines Succinctly, Alexandre Kowalczyk, 2017.

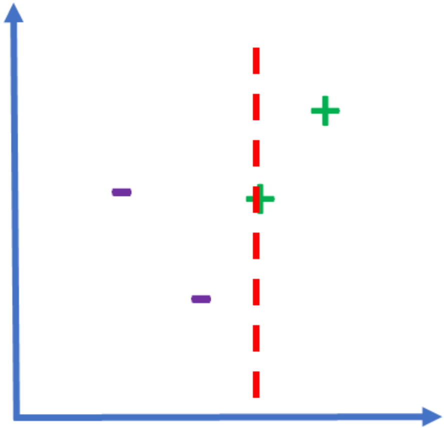
Intuition - Decision Boundary

How would you divide $+ve$ examples from $-ve$ examples?



Intuition - Decision Boundary

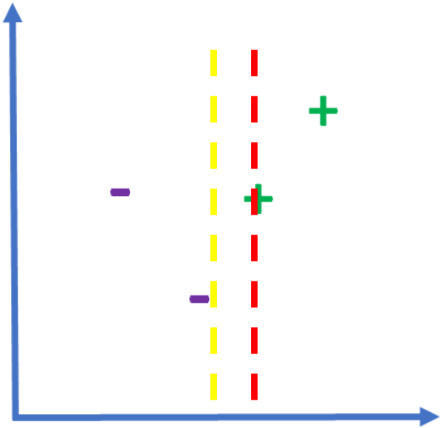
How would you divide $+ve$ examples from $-ve$ examples?



Seems infinite possibilities.

Intuition - Decision Boundary

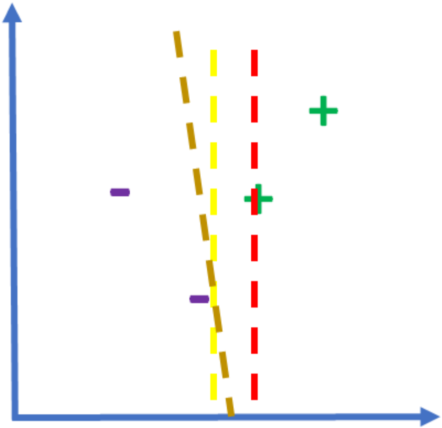
How would you divide $+ve$ examples from $-ve$ examples?



Seems infinite possibilities.

Intuition - Decision Boundary

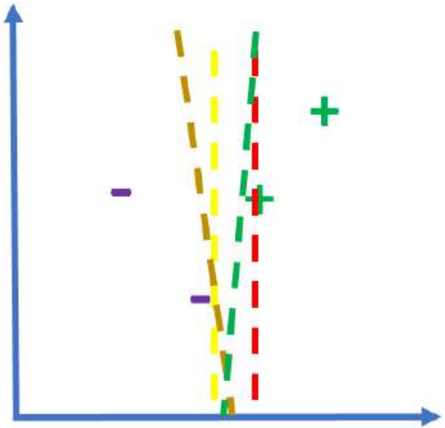
How would you divide $+ve$ examples from $-ve$ examples?



Seems infinite possibilities.

Intuition - Decision Boundary

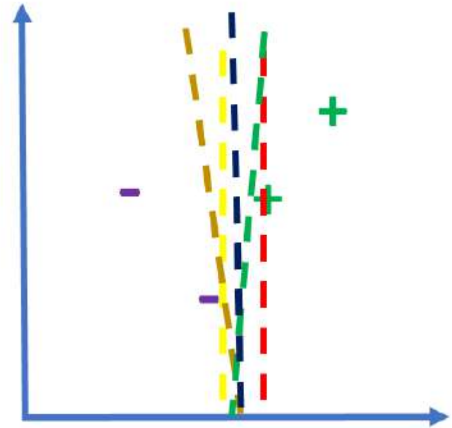
How would you divide $+ve$ examples from $-ve$ examples?



Seems infinite possibilities.

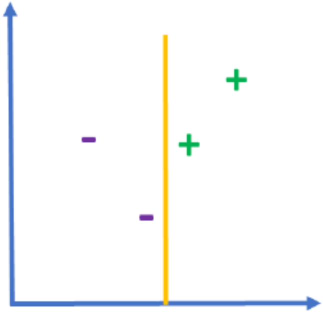
Intuition - Decision Boundary

How would you divide $+ve$ examples from $-ve$ examples?

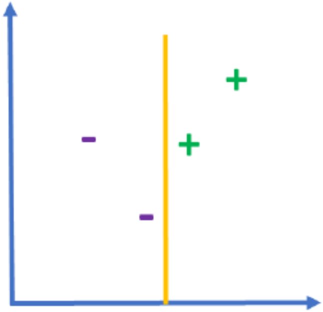


Seems infinite possibilities.

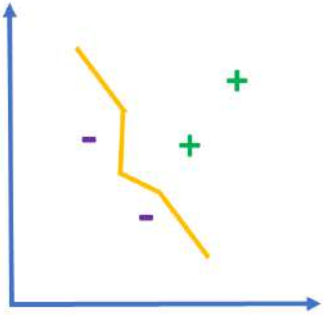
Intuition - Decision Boundary



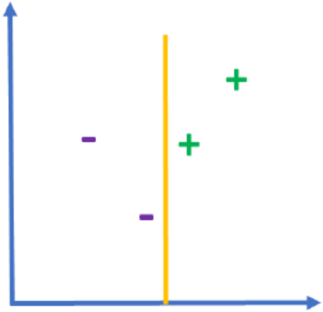
Intuition - Decision Boundary



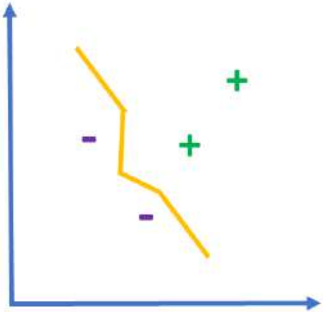
Tree / Perceptron



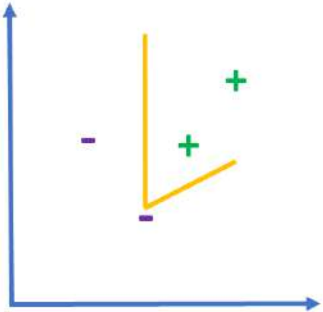
Intuition - Decision Boundary



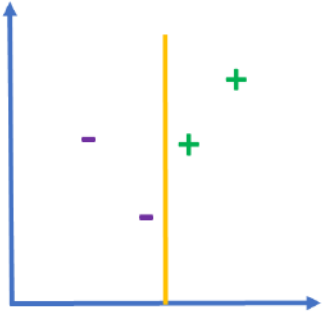
Tree / Perceptron



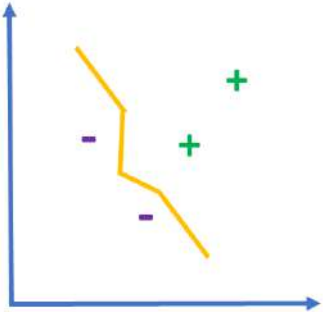
$k-$ Nearest Neighbor



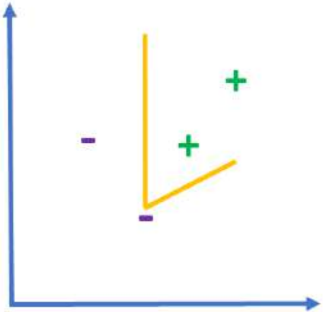
Intuition - Decision Boundary



Tree / Perceptron

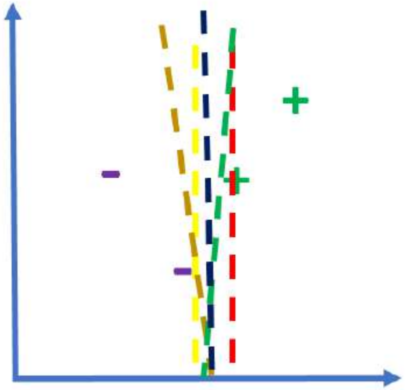


$k-$ Nearest Neighbor



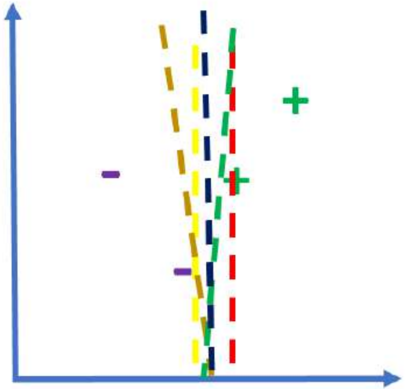
Neural Network

Goal of SVM



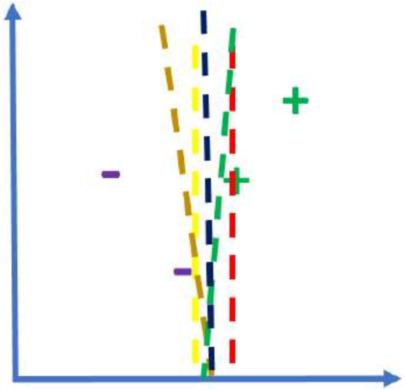
- Goal of SVM is to identify an optimal separating hyperplane which maximizes the margin between different classes of the training data.

Goal of SVM



- Goal of SVM is to identify an optimal separating hyperplane which maximizes the margin between different classes of the training data.
- SVM [a] is completely based on Mathematical Optimization problem.

Goal of SVM



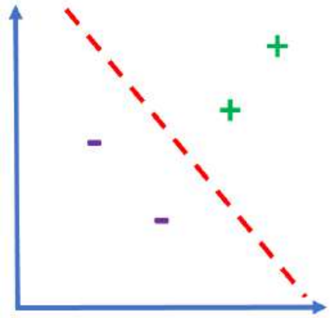
- Goal of SVM is to identify an optimal separating hyperplane which maximizes the margin between different classes of the training data.
- SVM [a] is completely based on Mathematical Optimization problem.
- SVMs are linear classifiers (a line in 2 dimensions, a plane in 3 dimensions, a $n-1$ dimensional hyperplane in n dimensions [b].

[a] Cortes, C., Vapnik, V. Support-vector networks. Machine Learning 20, 273–297 (1995). https://doi.org/10.1007/BF00994018

[b] cs276a SVM Review-Stanford University

Section Contents

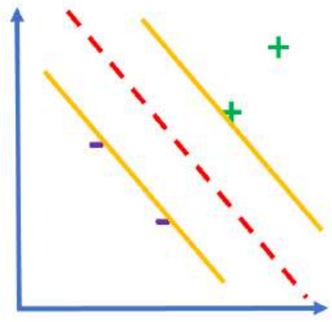
Intuition - Decision Boundary



Widest Street Approach

- Find such a line that maximizes the distance between $+ve$ examples and $-ve$ examples, while deciding decision boundary / surface.

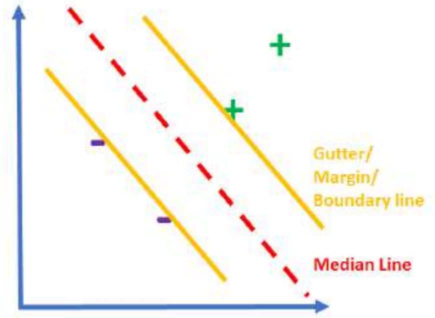
Intuition - Decision Boundary



Widest Street Approach

- Find such a line that maximizes the distance between $+ve$ examples and $-ve$ examples, while deciding decision boundary / surface.
- What would the decision rule?

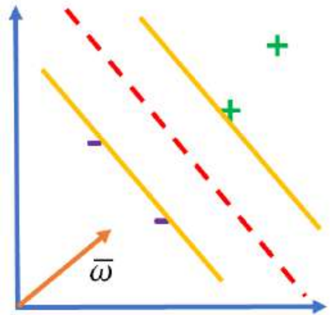
Intuition - Decision Boundary



Widest Street Approach

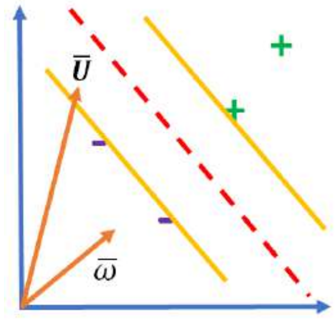
- Find such a line that maximizes the distance between $+ve$ examples and $-ve$ examples, while deciding decision boundary / surface.
- What would the decision rule?

Decision Boundary



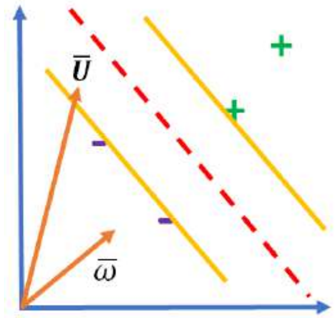
- Consider a vector $\bar{W}$ that is perpendicular to median / or gutter. We don't know anything about its length yet.

Decision Boundary



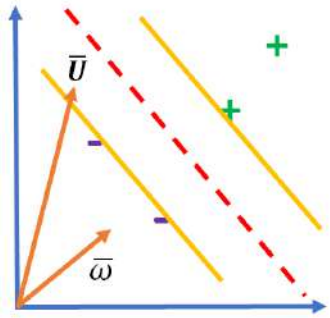
- Consider a vector $\bar{W}$ that is perpendicular to median / or gutter. We don't know anything about its length yet.
- Consider unknown point $\bar{U}$ and a vector points to it.

Decision Boundary



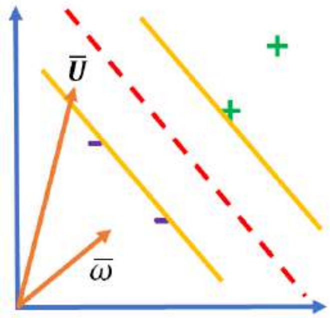
- Consider a vector $\bar{W}$ that is perpendicular to median / or gutter. We don't know anything about its length yet.
- Consider unknown point $\bar{U}$ and a vector points to it.
- We are interested to know whether this unknown is either right side of street or left or we want to know its label.
- What we can do, project that to perpendicular vector. The further we go we can find that its on the right side of the street.

Decision Boundary



- We are interested to know whether this unknown is either right side of street or left or we want to know its label.
- What we can do, project $\bar{U}$ to vector $\bar{W}$ which is perpendicular median line. The further we go, we can find that its on the right side of the street.

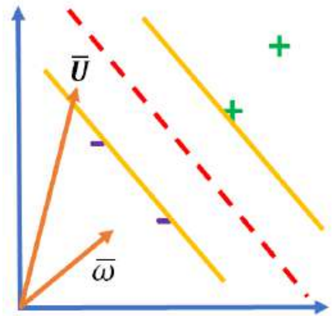
Decision Boundary



- We are interested to know whether this unknown is either right side of street or left or we want to know its label.
- What we can do, project $\bar{U}$ to vector $\bar{W}$ which is perpendicular median line. The further we go, we can find that its on the right side of the street.

$$\bar{W} \cdot \bar{U} \geq C$$

Decision Boundary

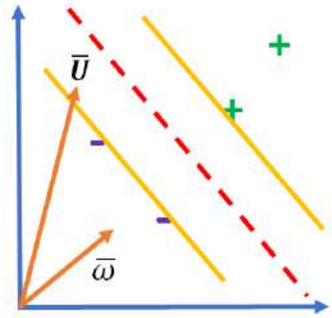


- We are interested to know whether this unknown is either right side of street or left or we want to know its label.
- What we can do, project $\bar{U}$ to vector $\bar{W}$ which is perpendicular median line. The further we go, we can find that its on the right side of the street.

$$\bar{W} \cdot \bar{U} \geq C$$

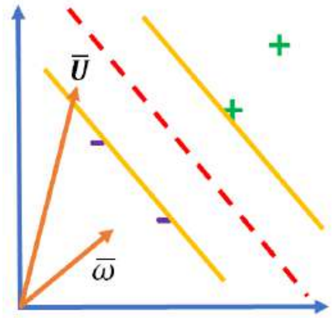
- Dot product is projecting onto $\bar{W}$. The bigger the projection is then it will cross median line and then unknown vector can be labeled as $+ve$ sample.

Decision Boundary



- Then, without loss of generality we can say:

Decision Boundary

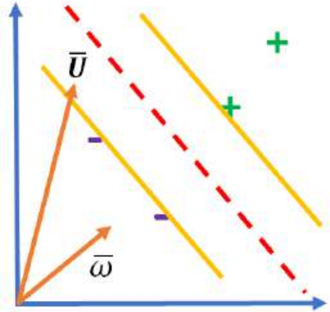


- Then, without loss of generality we can say:

$$\bar{W} \cdot \bar{U} + b \geq 0 \quad \text{THEN } +\text{ve} \tag{1}$$

This is Decision Rule.

Decision Boundary



- Then, without loss of generality we can say:

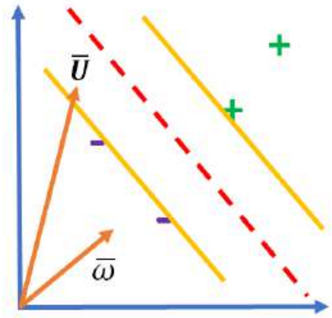
$$\bar{W} \cdot \bar{U} + b \geq 0 \quad \text{THEN } +\text{ve} \tag{1}$$

This is Decision Rule.

- We don't know (yet) what constant b, $C = -b$, to use and what $\bar{W}$ to use either.

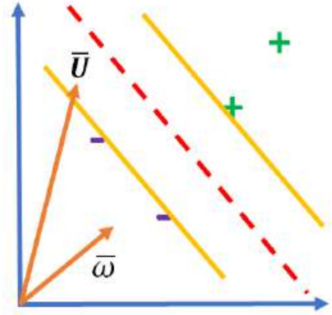
Section Contents

Constraints



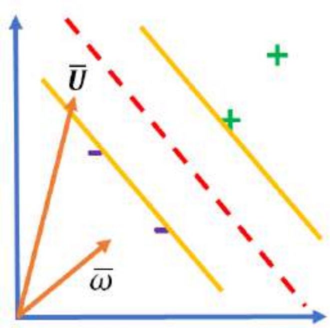
- We just know that $\bar{W}$ needs to be perpendicular to the median line of the street.

Constraints



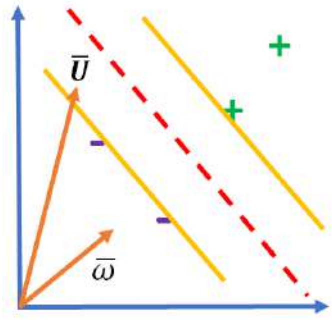
- We just know that $\bar{W}$ needs to be perpendicular to the median line of the street.
- But then there many $\bar{W}_s$ perpendicular to the median line of the street, any length is not fixed yet.
- What should we do?

Constraints



- We just know that $\bar{W}$ needs to be perpendicular to the median line of the street.
- But then there many $\bar{W}_s$ perpendicular to the median line of the street, any length is not fixed yet.
- What should we do?
- So we need to put constraints to find particular $\bar{W}$ and b that maximizes width of the street (separation between $+_s$ and $-_s$).

Constraints



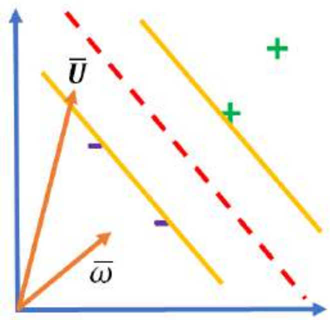
- Putting constraints to calculate $\bar{W}$ and b.

$$\bar{W} \cdot \bar{X}_+ + b \geq 1 \quad \{\text{for +ve samples}\} \tag{2}$$

$$\bar{W} \cdot \bar{X}_- + b \leq -1 \quad \{\text{for -ve samples}\} \tag{3}$$

[1] Did you see similarity with "Perceptron" decision rule?

Constraints



$*1$

- Putting constraints to calculate $\bar{W}$ and b.

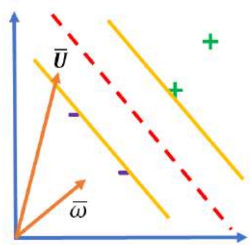
$$\bar{W} \cdot \bar{X_+} + b \geq 1 \quad \{\text{for +ve samples}\} \tag{2}$$

$$\bar{W} \cdot \bar{X_-} + b \leq -1 \quad \{\text{for -ve samples}\} \tag{3}$$

So imposing separation of -1 to +1 for $-ve$ and $+ve$ samples (maximizing margin).

[1] Did you see similarity with "Perceptron" decision rule?

Constraints



- Equation 2 and 3 can be written / combined as:

$$y_i(\bar{W} \cdot \bar{X}_i + b) \geq 1 \qquad (4)$$

where:
- y_i is $+1$ for $+ve$ samples.
- y_i is -1 for $-ve$ samples.

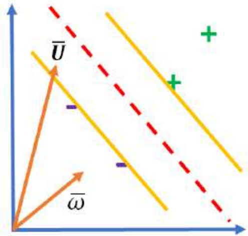
Proof:

1. for $+ve$ samples:

$$\begin{aligned} 1 \times (\bar{W} \cdot \bar{X}_i + b) &\geq 1 \\ \Rightarrow (\bar{W} \cdot \bar{X}_i + b) &\geq 1 \end{aligned} \qquad (5)$$

Same as Eq. 2.

Constraints



- Equation 2 and 3 can be written / combined as:

$$y_i(\bar{W} \cdot \bar{X}_i + b) \geq 1 \qquad (4)$$

where:
- y_i is $+1$ for $+ve$ samples.
- y_i is -1 for $-ve$ samples.

Proof:

1. for $+ve$ samples:

$$\begin{aligned} 1 \times (\bar{W} \cdot \bar{X}_i + b) &\geq 1 \\ \Rightarrow (\bar{W} \cdot \bar{X}_i + b) &\geq 1 \end{aligned} \qquad (5)$$

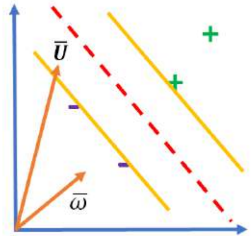
Same as Eq. 2.

2. for $-ve$ samples:

$$\begin{aligned} -1 \times (\bar{W} \cdot \bar{X}_i + b) &\geq 1 \\ \Rightarrow -\bar{W} \cdot \bar{X}_i - b &\geq 1 \\ \Rightarrow \bar{W} \cdot \bar{X}_i + b &\leq -1 \end{aligned} \qquad (6)$$

Same as Eq. 3.

Constraints



Back to equation 4:

$$y_i(\bar{W} \cdot \bar{X}_i + b) \geq 1$$

where:
- y_i is $+1$ for $+ve$ samples.
- y_i is -1 for $-ve$ samples.

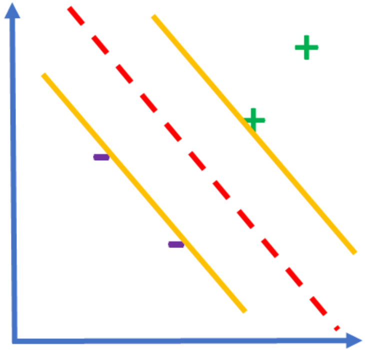
Equation 4 can be written as:

$$\begin{aligned} y_i(\bar{W} \cdot \bar{X}_i + b) &\geq 1 \\ y_i(\bar{W} \cdot \bar{X}_i + b) - 1 &\geq 0 \end{aligned} \tag{7}$$

Additional constraint:

$$y_i(\bar{W} \cdot \bar{X}_i + b) - 1 = 0$$
$$\{\text{for samples } (X_i) \text{ in gutter or at boundary / margin}\} \tag{8}$$

Samples on the boundary / gutter

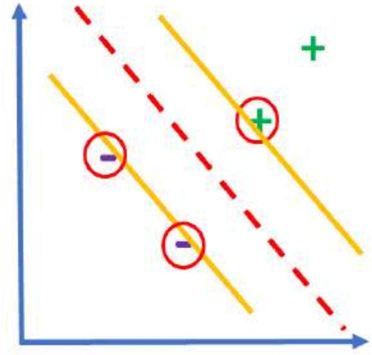


Additional constraint:

$$y_i(\bar{W} \cdot \bar{X}_i + b) - 1 = 0$$

{for samples (X_i) in gutter or at boundary / margin}

Samples on the boundary / gutter



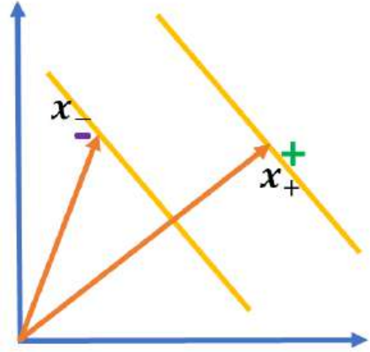
Additional constraint:

$$y_i(\bar{W} \cdot \bar{X}_i + b) - 1 = 0$$

{for samples (X_i) in gutter or at boundary / margin}

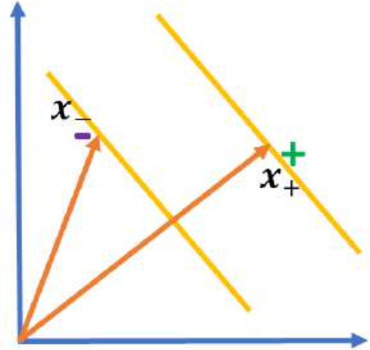
These samples are also called as Support Vectors.

Distance b/w boundary lines / margin / gutters



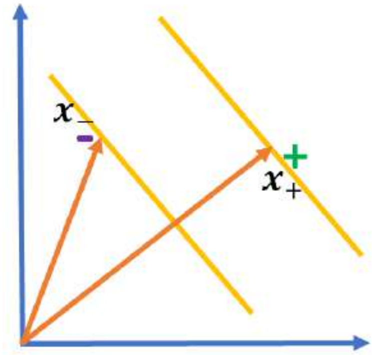
- We are trying to arrange line $\bar{W}$ and b in a such a way that it maximizes width of the street (separation between $+_s$ and $-_s$).

Distance b/w boundary lines / margin / gutters



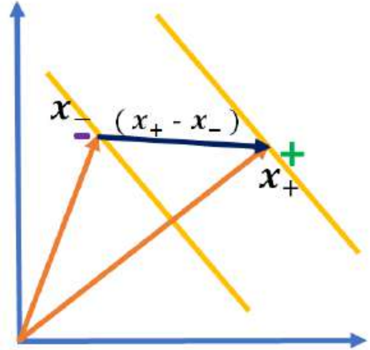
- We are trying to arrange line $\bar{W}$ and b in a such a way that it maximizes width of the street (separation between $+_s$ and $-_s$).
- Boundary lines are parallel to one another, we can pick points on these lines to define width of the street.

Distance b/w boundary lines / margin / gutters



- We are trying to arrange line $\bar{W}$ and b in a such a way that it maximizes width of the street (separation between $+_s$ and $-_s$).
- Boundary lines are parallel to one another, we can pick points on these lines to define width of the street.
- Width of the street is distance b/w the gutters / boundary lines.

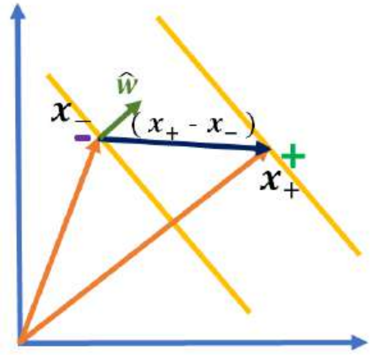
Distance b/w boundary lines / margin / gutters



- We are trying to arrange line $\bar{W}$ and b in a such a way that it maximizes width of the street (separation between $+_s$ and $-_s$).
- Boundary lines are parallel to one another, we can pick points on these lines to define width of the street.
- Width of the street is distance b/w the gutters / boundary lines.
- Difference of two vectors can give us width of the street:

$$(\bar{X}_+ - \bar{X}_-)$$

Distance b/w boundary lines / margin / gutters

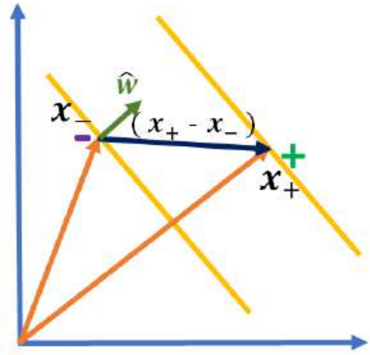


- Width of street:

$$= (\bar{X}_+ - \bar{X}_-) \cdot \frac{\bar{W}}{||W||} \tag{9}$$

- Where $\frac{\bar{W}}{||W||}$ is a unit vector ($\hat{W}$) perpendicular / normal to gutter or boundary line.

Distance b/w boundary lines / margin / gutters



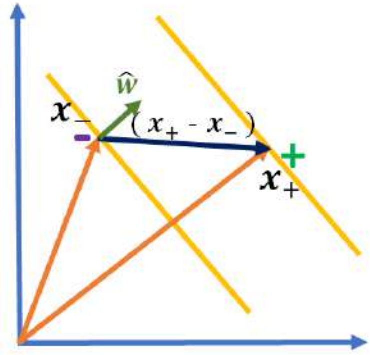
- Width of street:

$$= (\bar{X}_+ - \bar{X}_-) \cdot \frac{\bar{W}}{||W||} \tag{9}$$

- Where $\frac{\bar{W}}{||W||}$ is a unit vector ($\hat{W}$) perpendicular / normal to gutter or boundary line.

- In other words, projection of difference vector on to unit vector ($\hat{W}$) will be width of the street (difference in the direction of $\bar{W}$ vector).

Distance b/w boundary lines / margin / gutters



- Width of street:

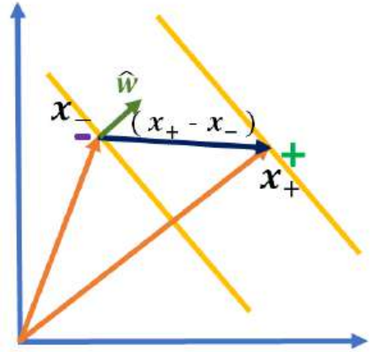
$$= (\bar{X}_+ - \bar{X}_-) \cdot \frac{\bar{W}}{||W||} \tag{9}$$

- Where $\frac{\bar{W}}{||W||}$ is a unit vector ($\hat{W}$) perpendicular / normal to gutter or boundary line.

- In other words, projection of difference vector on to unit vector ($\hat{W}$) will be width of the street (difference in the direction of $\bar{W}$ vector).

- It's a dot product, so its scalar, width of the street.

Distance b/w boundary lines / margin / gutters



Width of street:

$$= (\bar{X}_+ - \bar{X}_-) \cdot \frac{\bar{W}}{||W||}$$

From Equation 8 we know that, samples in a gutter (enforcing constraint) =

$$y_i(\bar{W} \cdot \bar{X}_i + b) - 1 = 0$$

So,

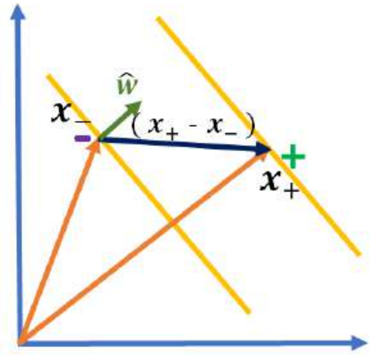
- for $+ve$ sample:

$$\bar{W} \cdot \bar{X}_i = 1 - b \tag{10}$$

- for $-ve$ sample:

$$\begin{aligned} -\bar{W} \cdot \bar{X}_i - b - 1 = 0 \\ -\bar{W} \cdot \bar{X}_i = 1 + b \end{aligned} \tag{11}$$

Distance b/w boundary lines / margin / gutters



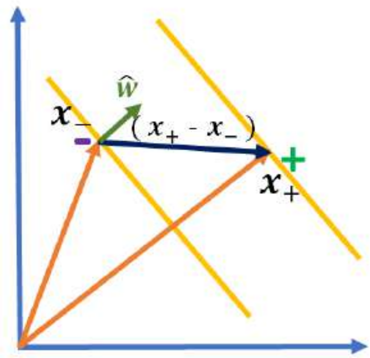
Width of street:

$$(\bar{X_+} - \bar{X_-}) \cdot \frac{\bar{W}}{||W||}$$
$$\bar{W} \cdot \bar{X_+} - \bar{W} \cdot \bar{X_-} \cdot \frac{1}{||W||} \tag{12}$$

Putting back values from Equations 10 and 11 into Equation 12.

$$\text{Width} = \frac{2}{||W||} \tag{13}$$

Distance b/w gutter



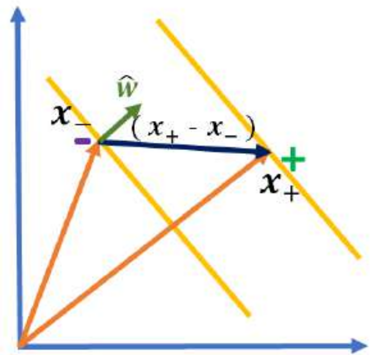
Width of street:

$$\text{Width} = \frac{2}{||W||}$$

- SVM tries to maximize this width, to have maximum possible separation between samples of different classes.

$$\text{Width} = \text{Max}\frac{2}{||W||} \implies \text{Min}||W|| \implies \text{Min}\frac{1}{2}||W||^2$$

Distance b/w gutter



Width of street:

$$\text{Width} = \frac{2}{||W||}$$

- SVM tries to maximize this width, to have maximum possible separation between samples of different classes.

$$\text{Width} = \text{Max}\frac{2}{||W||} \implies \text{Min}||W|| \implies \text{Min}\frac{1}{2}||W||^2$$

$$\text{Width} = \text{Min}\frac{1}{2}||W||^2 \qquad (14)$$

Summary

Stages in the development:

1. Decision Rule:

$$\bar{W} \cdot \bar{U} + b \geq 0 \quad \text{THEN +ve}$$

2. Projection of W on U and put a constraint then it should be $\geq +1$ for $+ve$ samples and ≤ -1 for $-ve$ samples.

$$\bar{W} \cdot \bar{X}_{+} + b \geq 1 \quad \{\text{for +ve samples}\}$$

$$\bar{W} \cdot \bar{X}_{-} + b \leq -1 \quad \{\text{for -ve samples}\}$$

3. Additional constraint for samples in gutter

$$y_i(\bar{W} \cdot \bar{X}_i + b) - 1 = 0$$

4. Then we discovered we wish to maximize / minimize this expression:

$$\text{Width} = \text{Min}\frac{1}{2}||W||^2$$

What's next?

- We have now transformed the problem into a form that can be efficiently solved.

$$\text{Width} = \text{Min}\frac{1}{2}||W||^2$$

- The above is an optimization problem with a convex quadratic objective and some constraints. Its solution gives us the optimal margin classifier.

What's next?

- We have now transformed the problem into a form that can be efficiently solved.

$$\text{Width} = \text{Min}\frac{1}{2}||W||^2$$

- The above is an optimization problem with a convex quadratic objective and some constraints. Its solution gives us the optimal margin classifier.
- How can we go forward (QUIZ):

What's next?

- We have now transformed the problem into a form that can be efficiently solved.

$$\text{Width} = \text{Min}\frac{1}{2}||W||^2$$

- The above is an optimization problem with a convex quadratic objective and some constraints. Its solution gives us the optimal margin classifier.
- How can we go forward (QUIZ):
  1. Laplace

What's next?

- We have now transformed the problem into a form that can be efficiently solved.

$$\text{Width} = \text{Min}\frac{1}{2}||W||^2$$

- The above is an optimization problem with a convex quadratic objective and some constraints. Its solution gives us the optimal margin classifier.
- How can we go forward (QUIZ):
  1. Laplace
  2. Legendre

What's next?

- We have now transformed the problem into a form that can be efficiently solved.

$$\text{Width} = \text{Min}\frac{1}{2}||W||^2$$

- The above is an optimization problem with a convex quadratic objective and some constraints. Its solution gives us the optimal margin classifier.
- How can we go forward (QUIZ):
  1. Laplace
  2. Legendre
  3. Lagrange

Section Contents

Lagrange Multiplier

- Lagrange multipliers provides a way for finding extremum of a function subject to equality constraints i.e., subject to the condition that one or more equations have to be satisfied exactly by the chosen values of the variables.
- The great advantage of this method is that it allows the optimization to be solved without explicit parameterization in terms of the constraints.
- Method can be summarized as follows: in order to find the stationary points of a function $f(x)$ subjected to the equality constraint $g(x) = 0$, form the Lagrangian function:

$$L(x, \lambda) = f(x) - \lambda g(x) \tag{15}$$

where λ = Lagrange multiplier

[2]

[2] Refer Section 8 to see disscussion on intuition of Lagrange multiplier

Lagrange Multiplier

- Taking Equation 15 and writing function that we are trying to find extremum.

$$L = \frac{1}{2}||W||^2 - \sum \alpha_i (\text{write down constraints})$$

Lagrange Multiplier

- Taking Equation 15 and writing function that we are trying to find extremum.

$$L = \frac{1}{2}||W||^2 - \sum \alpha_i (\text{write down constraints})$$

- Constraint is given in Equation 8

$$L = \frac{1}{2}||W||^2 - \sum \alpha_i \left[y_i(\bar{W} \cdot \bar{X}_i + b) - 1 \right] \tag{16}$$

where α_i = Lagrange multiplier
α_i will be non-zero for vectors connected with samples in gutter, otherwise it will be zero.

Find Extremum

- What needs to be done to find extremum of Equation 16 ?

$$L = \frac{1}{2}||W||^2 - \sum \alpha_i \left[y_i(\bar{W} \cdot \bar{X}_i + b) - 1 \right]$$

Find Extremum

- What needs to be done to find extremum of Equation 16 ?

$$L = \frac{1}{2}||W||^2 - \sum \alpha_i \left[y_i(\bar{W} \cdot \bar{X}_i + b) - 1\right]$$

- Take derivative and set it equal to zero.

Find Extremum

- What needs to be done to find extremum of Equation 16 ?

$$L = \frac{1}{2}||W||^2 - \sum \alpha_i \left[y_i(\bar{W} \cdot \bar{X}_i + b) - 1\right]$$

- Take derivative and set it equal to zero.

- **Take derivative w.r.t. “W”:**

Find Extremum

- What needs to be done to find extremum of Equation 16 ?

$$L = \frac{1}{2}||W||^2 - \sum \alpha_i \left[y_i(\bar{W} \cdot \bar{X}_i + b) - 1\right]$$

- Take derivative and set it equal to zero.

- **Take derivative w.r.t. "W":**

$$\frac{\partial L}{\partial W} = \bar{W} - \sum_i \alpha_i y_i \bar{X}_i = 0 \Longrightarrow W = \sum_i \alpha_i y_i \bar{X}_i \qquad (17)$$

Where α_i is a scalar, y_i is +1 or -1 and X_i is sample vector. **What this equation signifies?**

Find Extremum

- What needs to be done to find extremum of Equation 16 ?

$$L = \frac{1}{2}||W||^2 - \sum \alpha_i \left[y_i(\bar{W} \cdot \bar{X}_i + b) - 1 \right]$$

- Take derivative and set it equal to zero.

- **Take derivative w.r.t. "W":**

$$\frac{\partial L}{\partial W} = \bar{W} - \sum_i \alpha_i y_i \bar{X}_i = 0 \Longrightarrow W = \sum_i \alpha_i y_i \bar{X}_i \quad (17)$$

Where α_i is a scalar, y_i is +1 or -1 and X_i is sample vector. **What this equation signifies?**

What this equation signifies

Decision vector W is linear sum of **some** samples. Some, in the sense that α_i will be non-zero for few vectors (connected with samples in gutter).

Find Extremum

- Back to Equation 16. Any other variable that may vary?

$$L = \frac{1}{2}||W||^2 - \sum \alpha_i \left[y_i(\bar{W} \cdot \bar{X}_i + b) - 1 \right]$$

Find Extremum

- Back to Equation 16. Any other variable that may vary?

$$L = \frac{1}{2}||W||^2 - \sum \alpha_i \left[y_i(\bar{W} \cdot \bar{X}_i + b) - 1 \right]$$

- **Take derivative w.r.t. "b":**

Find Extremum

- Back to Equation 16. Any other variable that may vary?

$$L = \frac{1}{2}||W||^2 - \sum \alpha_i \left[y_i(\bar{W} \cdot \bar{X}_i + b) - 1 \right]$$

- **Take derivative w.r.t. "b":**

$$\frac{\partial L}{\partial b} = -\sum_i \alpha_i y_i = 0 \Longrightarrow \sum_i \alpha_i y_i = 0 \tag{18}$$

Find Extremum

Plug back value of W from Equation 17 to Equation 16.

$$W = \sum_i \alpha_i y_i \bar{X}_i$$

$$L = \frac{1}{2}||W||^2 - \sum_i \alpha_i \left[y_i(\bar{W} \cdot \bar{X}_i + b) - 1 \right]$$

Find Extremum

Plug back value of W from Equation 17 to Equation 16.

$$W = \sum_i \alpha_i y_i \bar{X}_i$$

$$L = \frac{1}{2}||W||^2 - \sum_i \alpha_i \left[y_i(\bar{W} \cdot \bar{X}_i + b) - 1 \right]$$

$$\begin{aligned} L = &\frac{1}{2}(\sum_i \alpha_i y_i \bar{X}_i) \cdot (\sum_j \alpha_j y_j \bar{X}_j) \\ &-(\sum_i \alpha_i y_i \bar{X}_i) \cdot (\sum_j \alpha_j y_j \bar{X}_j) \\ &-\sum_i \alpha_i y_i b + \sum_i \alpha_i \end{aligned} \tag{19}$$

Find Extremum

Term shown in red in Equation 19 $= 0$, refer Equation 18

Now arrange and re-write Lagrangian Equation 19

Find Extremum

Term shown in red in Equation 19 $= 0$, refer Equation 18

Now arrange and re-write Lagrangian Equation 19

$$L = \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j \bar{X}_i \cdot \bar{X}_j \tag{20}$$

Find Extremum

Term shown in red in Equation 19 = 0 , refer Equation 18

Now arrange and re-write Lagrangian Equation 19

$$L = \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j \bar{X}_i \cdot \bar{X}_j \tag{20}$$

What this equation signifies

This optimization / finding extremum of a function depends only on dot products of pairs of samples (dual problem).

Decision Rule

Recall Equation 1, related to decision rule:

$$\bar{W} \cdot \bar{U} + b \geq 0 \quad \text{THEN } +\text{ve}$$

Now, we can update this Equation, with derived value of $\bar{W}$, refer Equation 17:

Decision Rule

Recall Equation 1, related to decision rule:

$$\bar{W} \cdot \bar{U} + b \geq 0 \quad \text{THEN } +\text{ve}$$

Now, we can update this Equation, with derived value of $\bar{W}$, refer Equation 17:

$$\sum_i \alpha_i y_i \bar{X}_i \cdot \bar{U} + b \geq 0 \quad \text{THEN } +\text{ve} \tag{21}$$

Decision Rule

Recall Equation 1, related to decision rule:

$$\bar{W} \cdot \bar{U} + b \geq 0 \quad \text{THEN } +\text{ve}$$

Now, we can update this Equation, with derived value of $\bar{W}$, refer Equation 17:

$$\sum_i \alpha_i y_i \bar{X}_i \cdot \bar{U} + b \geq 0 \quad \text{THEN } +\text{ve} \tag{21}$$

What this equation signifies?

Decision rule depends again only on dot product of unknown vector and sample vector.

Summary

Summary

- Recall Equation 13, Width $= \text{Max} \frac{2}{||W||}$, while satisfying constraints, given by: classify training examples correctly $y_i(\bar{W} \cdot \bar{X}_i + b) - 1 \geq 0, \forall i$.

Summary

Summary

- Recall Equation 13, Width = $\text{Max}\frac{2}{||W||}$, while satisfying constraints, given by: classify training examples correctly $y_i(\bar{W} \cdot \bar{X}_i + b) - 1 \geq 0, \forall i$.
- Then, we transformed above equation into equivalent problem (which is easier to solve), Equation 14, Width = $\text{Min}\frac{1}{2}||W||^2$.
- Why its easier?

Summary

Summary

- Recall Equation 13, Width $= \text{Max}\frac{2}{||W||}$, while satisfying constraints, given by: classify training examples correctly $y_i(\bar{W} \cdot \bar{X}_i + b) - 1 \geq 0, \forall i$.
- Then, we transformed above equation into equivalent problem (which is easier to solve), Equation 14, Width $= \text{Min}\frac{1}{2}||W||^2$.
- Why its easier?
- Actually, this is easier to solve as when we have optimization problem in the form given above while satisfying constraints, is called quadratic programming (QP) problem. QP is well known field and it's solution is easier to find.

Summary

Summary

- Recall Equation 13, Width $= \text{Max}\frac{2}{||W||}$, while satisfying constraints, given by: classify training examples correctly $y_i(\bar{W} \cdot \bar{X}_i + b) - 1 \geq 0, \forall i$.
- Then, we transformed above equation into equivalent problem (which is easier to solve), Equation 14, Width $= \text{Min}\frac{1}{2}||W||^2$.
- Why its easier?
- Actually, this is easier to solve as when we have optimization problem in the form given above while satisfying constraints, is called quadratic programming (QP) problem. QP is well known field and it's solution is easier to find.
- Optimization problems of this form have convex function and thus unique solution is always guaranteed.

Summary

Summary

- Recall Equation 13, Width $= \text{Max}\frac{2}{||W||}$, while satisfying constraints, given by: classify training examples correctly $y_i(\bar{W} \cdot \bar{X}_i + b) - 1 \geq 0, \forall i$.
- Then, we transformed above equation into equivalent problem (which is easier to solve), Equation 14, Width $= \text{Min}\frac{1}{2}||W||^2$.
- Why its easier?
- Actually, this is easier to solve as when we have optimization problem in the form given above while satisfying constraints, is called quadratic programming (QP) problem. QP is well known field and it's solution is easier to find.
- Optimization problems of this form have convex function and thus unique solution is always guaranteed.
- Quadratic programming problem form: $L = \sum_i \alpha_i - \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j \bar{X}_i \cdot \bar{X}_j$

Summary

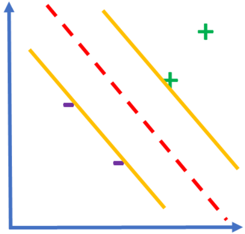
Summary

- Quadratic programming problem form: $L = \sum_i \alpha_i - \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j \bar{X}_i \cdot \bar{X}_j$
  While, $W = \sum_i \alpha_i y_i \bar{X}_i$ and
  $\sum_i \alpha_i y_i = 0$

Summary

Summary

- Quadratic programming problem form: $L = \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j \bar{X}_i \cdot \bar{X}_j$
  While, $W = \sum_i \alpha_i y_i \bar{X}_i$ and
  $\sum_i \alpha_i y_i = 0$

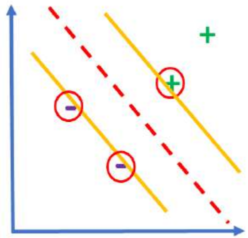


It turns out most of α_i are zeros, which implies that only few vectors (with non-zero α_i) matters in finding solution / decision boundary while most of vectors do not. Thus, building a **machine** with few **support vectors** (with non-zero α_i).

Summary

Summary

- Quadratic programming problem form: $L = \sum_i \alpha_i - \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j \bar{X}_i \cdot \bar{X}_j$
  While, $W = \sum_i \alpha_i y_i \bar{X}_i$ and
  $\sum_i \alpha_i y_i = 0$



It turns out most of α_i are zeros, which implies that only few vectors (with non-zero α_i) matters in finding solution / decision boundary while most of vectors do not. Thus, building a **machine** with few **support vectors** (with non-zero α_i).

Summary

Summary

- Quadratic programming problem form: $L = \sum_i \alpha_i - \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j \bar{X}_i \cdot \bar{X}_j$
  While, $W = \sum_i \alpha_i y_i \bar{X}_i$ and
  $\sum_i \alpha_i y_i = 0$
- This optimization / finding extremum of a function depends only on dot products of pairs of samples , $(\bar{X}_i{}^T \cdot \bar{X}_j)$. Note[a].
- Analyzing : $\bar{X}_i{}^T \cdot \bar{X}_j$, What it actually means?

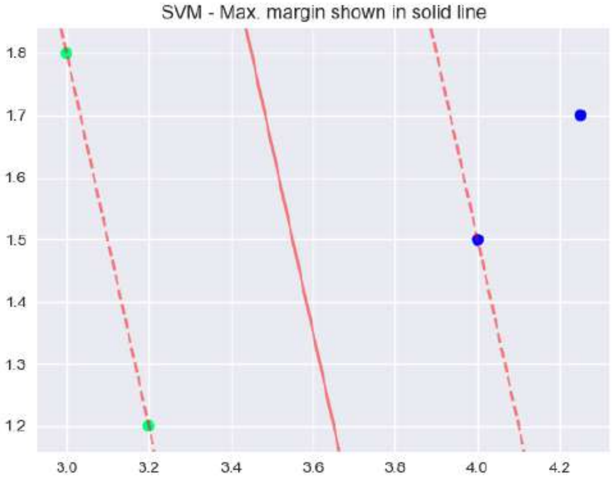
Summary

Summary

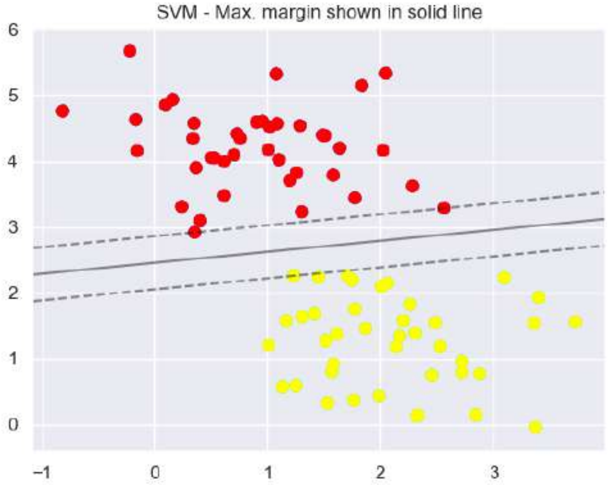
- Quadratic programming problem form: $L = \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j \bar{X}_i \cdot \bar{X}_j$
  While, $W = \sum_i \alpha_i y_i \bar{X}_i$ and
  $\sum_i \alpha_i y_i = 0$
- This optimization / finding extremum of a function depends only on dot products of pairs of samples , $(\bar{X}_i{}^T \cdot \bar{X}_j)$. Note[a].
- Analyzing : $\bar{X}_i{}^T \cdot \bar{X}_j$, What it actually means?
  1. Its a dot product, projection of one on another.
  2. It is a **measure of similarity** between two non-zero vectors. If vectors are orthogonal value will be zero, and if vector in opposite direction value will be $-ve$.

[a] Transpose is used to make matrix dimensions compatible

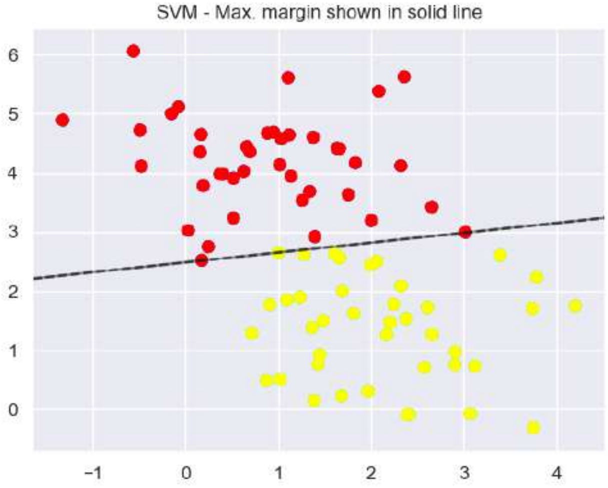
Ocular proof



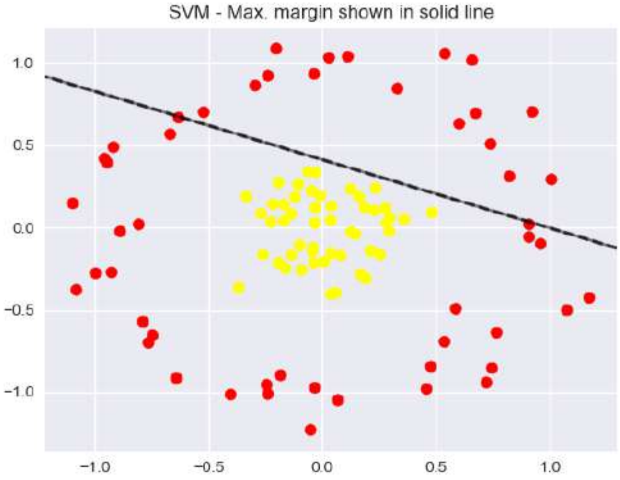
Ocular proof



Ocular proof



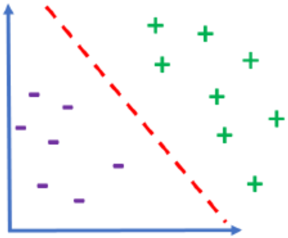
Ocular proof



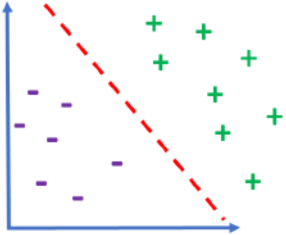
What to do here? Data is not linearly separable!

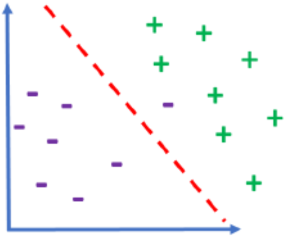
Section Contents

Non-linearly separable

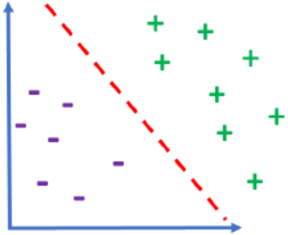


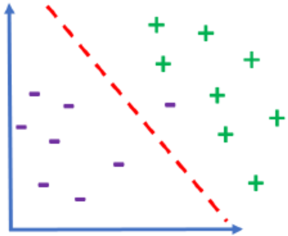
Non-linearly separable





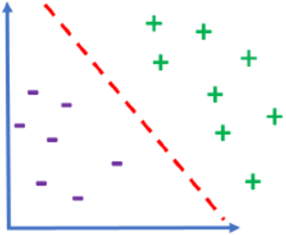
Non-linearly separable

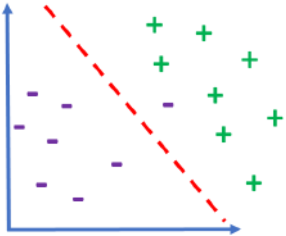




- What to do, since SVM find linear classification boundary? SVMs are linear classifiers (a line in 2 dimensions, a plane in 3 dimensions, a $n-1$ dimensional hyperplane in n dimensions.
- Some probabilities:

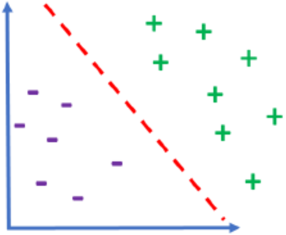
Non-linearly separable

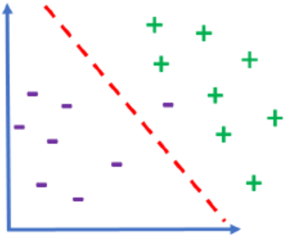




- What to do, since SVM find linear classification boundary? SVMs are linear classifiers (a line in 2 dimensions, a plane in 3 dimensions, a $n-1$ dimensional hyperplane in n dimensions.
- Some probabilities:
  1. Probably its a outlier!

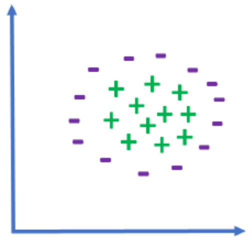
Non-linearly separable



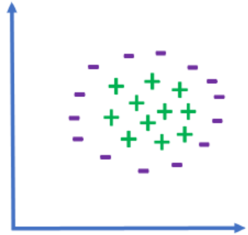


- What to do, since SVM find linear classification boundary? SVMs are linear classifiers (a line in 2 dimensions, a plane in 3 dimensions, a $n-1$ dimensional hyperplane in n dimensions.
- Some probabilities:
  1. Probably its a outlier!
  2. or find linear decision boundary while minimizing some cost function that penalizes for training error.

Non-linearly separable

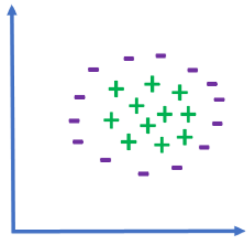


Non-linearly separable



- In previous example, one example seemed outlier and solution was proposed, but in this case it is impossible to come up with linear classifier.

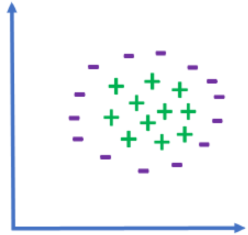
Non-linearly separable



- In previous example, one example seemed outlier and solution was proposed, but in this case it is impossible to come up with linear classifier.

- As it seems impossible to use SVM / linear classifier, should we just remove SVM from machine learning toolkit?

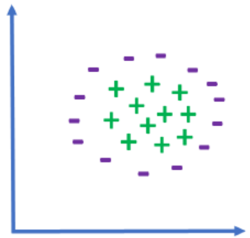
Non-linearly separable



- In previous example, one example seemed outlier and solution was proposed, but in this case it is impossible to come up with linear classifier.

- As it seems impossible to use SVM / linear classifier, should we just remove SVM from machine learning toolkit?
- No, there is a little trick that can be done to transform data (change the data point without changing the data point) in a such away that it become linearly separable.

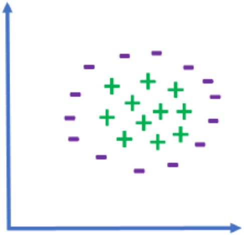
Non-linearly separable



- In previous example, one example seemed outlier and solution was proposed, but in this case it is impossible to come up with linear classifier.

- As it seems impossible to use SVM / linear classifier, should we just remove SVM from machine learning toolkit?
- No, there is a little trick that can be done to transform data (change the data point without changing the data point) in a such away that it become linearly separable.
- Define function Φ that will take data point and change its dimension (in our example there are two dimension).

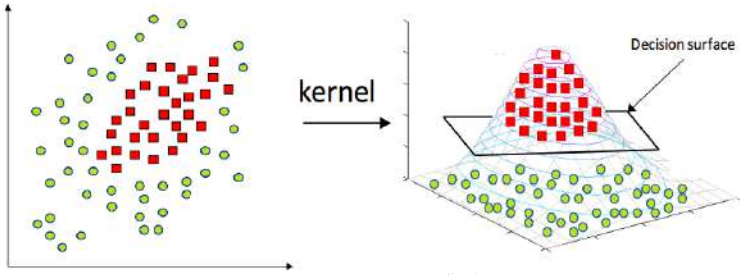
Non-linearly separable



- In previous example, one example seemed outlier and solution was proposed, but in this case it is impossible to come up with linear classifier.

- As it seems impossible to use SVM / linear classifier, should we just remove SVM from machine learning toolkit?
- No, there is a little trick that can be done to transform data (change the data point without changing the data point) in a such away that it become linearly separable.
- Define function Φ that will take data point and change its dimension (in our example there are two dimension).
- For example, we can transform data from our example in three dimensions using:
  $\Phi(x) =< x_1^2, x_2^2, \sqrt{2}x_1x_2 >$
  where: x_1 and x_2 are dimension of same vector

Intuition

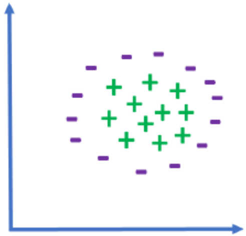


Video showing XOR data from 2D to 3D. [3]

- There is a little trick that can be done to transform data (change the data point without changing the data point) in a such away that it become linearly separable.
- Define function Φ that will take data point and change its dimension.

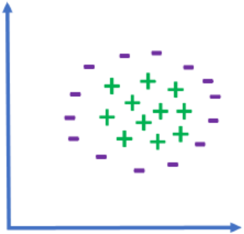
[3] https://youtu.be/5KIYu3zKvqo

Non-linearly separable



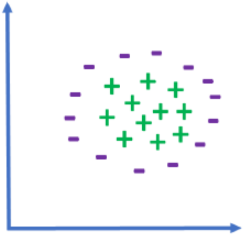
- $\Phi(x) =< x_1^2, x_2^2, \sqrt{2}x_1x_2 >$
- There isn't any new information!

Non-linearly separable



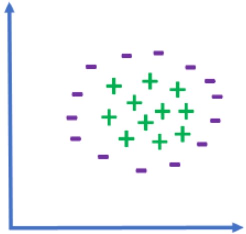
- $\Phi(x) =< x_1^2, x_2^2, \sqrt{2}x_1x_2 >$
- There isn't any new information!
- Reminder: Quadratic programming problem form: $L = \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i\alpha_j y_i y_j \bar{X}_i \cdot \bar{X}_j$

Non-linearly separable



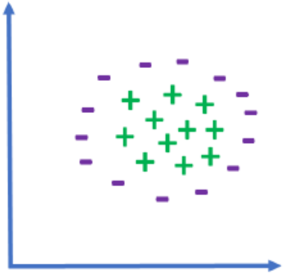
- $\Phi(x) =< x_1^2, x_2^2, \sqrt{2}x_1x_2 >$
- There isn't any new information!
- Reminder: Quadratic programming problem form: $L = \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j \bar{X}_i \cdot \bar{X}_j$
- Reminder: This optimization / finding extremum of a function depends only on dot products of pairs of samples, given by $\bar{X}_i^T \cdot \bar{X}_j$.

Non-linearly separable



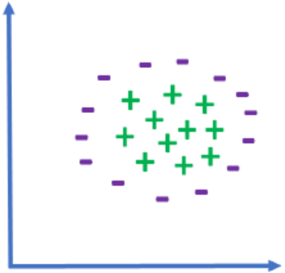
- $\Phi(x) =< x_1^2, x_2^2, \sqrt{2}x_1x_2 >$
- There isn't any new information!
- Reminder: Quadratic programming problem form: $L = \sum_i \alpha_i - \frac{1}{2}\sum_i \sum_j \alpha_i\alpha_j y_i y_j \bar{X_i} \cdot \bar{X_j}$
- Reminder: This optimization / finding extremum of a function depends only on dot products of pairs of samples, given by $\bar{X_i}^T \cdot \bar{X_j}$.
- What will be $\Phi(\bar{X_i})^T \cdot \Phi(\bar{X_j})$ for a given $\Phi(x)$

Non-linearly separable



- **QUIZ:** What will be $\Phi(\bar{X}_i)^T \cdot \Phi(\bar{X}_j)$ for a given $\Phi(x)$?

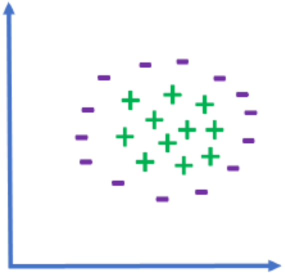
Non-linearly separable



- **QUIZ:** What will be $\Phi(\bar{X}_i)^T \cdot \Phi(\bar{X}_j)$ for a given $\Phi(x)$?

- Consider X_i as x and X_j as y, for the ease of notations:
  $\Phi(x)^T \Phi(y) =$

Non-linearly separable

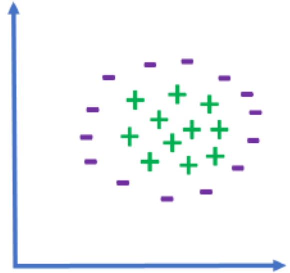


- **QUIZ:** What will be $\Phi(\bar{X}_i)^T \cdot \Phi(\bar{X}_j)$ for a given $\Phi(x)$?

- Consider X_i as x and X_j as y, for the ease of notations: $\Phi(x)^T\Phi(y) =$

$$\begin{aligned} < x_1^2, x_2^2, \sqrt{2}x_1x_2 >^T \cdot < y_1^2, y_2^2, \sqrt{2}y_1y_2 > \\ x_1^2y_1^2 + 2x_1x_2y_1y_2 + x_2^2y_2^2 \\ (x_1y_1 + x_2y_2)^2 \end{aligned} \tag{22}$$

Non-linearly separable



- **QUIZ:** What will be $\Phi(\bar{X}_i)^T \cdot \Phi(\bar{X}_j)$ for a given $\Phi(x)$?

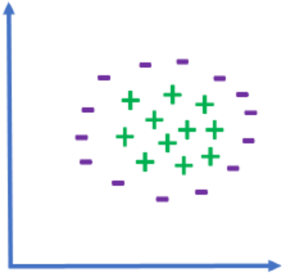
- Consider X_i as x and X_j as y, for the ease of notations: $\Phi(x)^T\Phi(y) =$

$$< x_1^2, x_2^2, \sqrt{2}x_1x_2 >^T \cdot < y_1^2, y_2^2, \sqrt{2}y_1y_2 >$$
$$x_1^2y_1^2 + 2x_1x_2y_1y_2 + x_2^2y_2^2 \qquad (22)$$
$$(x_1y_1 + x_2y_2)^2$$

- or this is equal to:

$$(x^Ty)^2 \qquad (23)$$

Non-linearly separable



- **QUIZ:** What will be $\Phi(\bar{X}_i)^T \cdot \Phi(\bar{X}_j)$ for a given $\Phi(x)$?

- Consider X_i as x and X_j as y, for the ease of notations: $\Phi(x)^T \Phi(y) =$

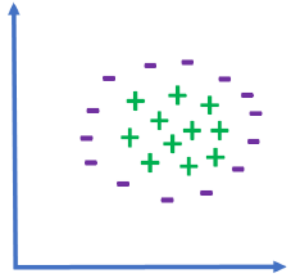
$$< x_1^2, x_2^2, \sqrt{2}x_1x_2 >^T \cdot < y_1^2, y_2^2, \sqrt{2}y_1y_2 >$$
$$x_1^2y_1^2 + 2x_1x_2y_1y_2 + x_2^2y_2^2 \quad (22)$$
$$(x_1y_1 + x_2y_2)^2$$

- or this is equal to:

$$(x^Ty)^2 \quad (23)$$

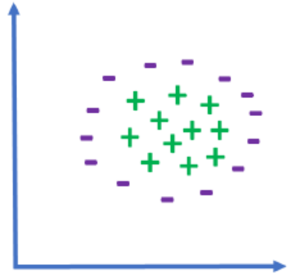
- So, dot product becomes square of last dot product.

Kernel Trick



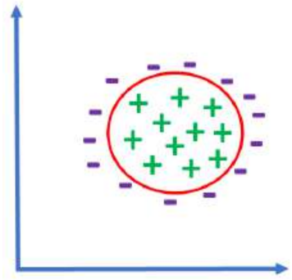
- Refer Equation 23, its particular form of equation of circle in matrix form.

Kernel Trick



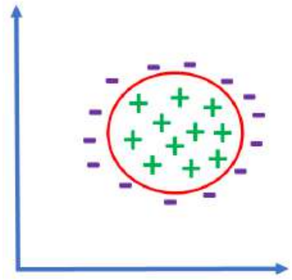
- Refer Equation 23, its particular form of equation of circle in matrix form.
- Somehow, the notion of similarity $(\bar{X_i}^T \cdot \bar{X_j})$ is transformed to notion of circle where some points remain inside the circle while others don't.

Kernel Trick



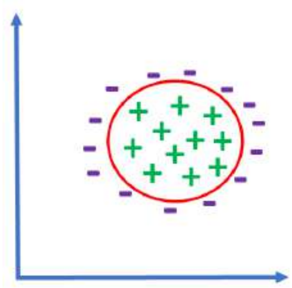
- Refer Equation 23, its particular form of equation of circle in matrix form.
- Somehow, the notion of similarity $(\bar{X}_i^T \cdot \bar{X}_j)$ is transformed to notion of circle where some points remain inside the circle while others don't.
- So, data got transformed from 2D to 3D (without any additional information) in such a way that now it can be separated by hyperplane.

Kernel Trick



- Refer Equation 23, its particular form of equation of circle in matrix form.
- Somehow, the notion of similarity $(\bar{X}_i{}^T \cdot \bar{X}_j)$ is transformed to notion of circle where some points remain inside the circle while others don't.
- So, data got transformed from 2D to 3D (without any additional information) in such a way that now it can be separated by hyperplane.
- This little trick of projecting data into higher dimension space to make it separable is called Kernel trick.

Kernel Trick



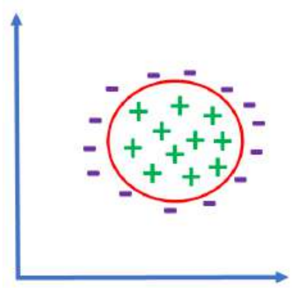
- Coming back to this equation:
  Quadratic programming problem form:

$$L = \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j \bar{X}_i \cdot \bar{X}_j$$

and Equation 23:

$$(x^T y)^2$$

Kernel Trick



- Coming back to this equation:
  Quadratic programming problem form:

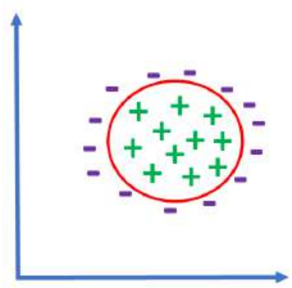
$$L = \sum_i \alpha_i - \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j \bar{X}_i \cdot \bar{X}_j$$

and Equation 23:

$$(x^T y)^2$$

- This signifies that data points don't need to be transformed separately, but rather its just a dot product squared! **So we actually never used** Φ.

Kernel Trick



- Coming back to this equation:
  Quadratic programming problem form:

$$L = \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j \bar{X}_i \cdot \bar{X}_j$$

and Equation 23:

$$(x^T y)^2$$

- This signifies that data points don't need to be transformed separately, but rather its just a dot product squared! **So we actually never used Φ.**
- **That's the beauty of mathematics and SVM.**

Kernel Trick

- $$L = \sum_i \alpha_i - \frac{1}{2}\sum_i\sum_j \alpha_i\alpha_j y_i y_j \bar{X}_i \cdot \bar{X}_j$$
- This equation can now be written as:

$$L = \sum_i \alpha_i - \frac{1}{2}\sum_i\sum_j \alpha_i\alpha_j y_i y_j K(\bar{X}_i \cdot \bar{X}_j) \tag{24}$$

  Where K is Kernel function, that takes X_i and X_j and returns scalar, the inner product (generalization of the dot product) between two points in a suitable feature space.
- Kernel functions allow to inject domain knowledge into classifier.

Kernel Trick

- Gaussian Kernel

$$K(\mathbf{x}_i, \mathbf{x}_j) = e^{\frac{-\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}} \tag{25}$$

- Polynomial

$$K(\mathbf{x}_i, \mathbf{x}_j) = ((\mathbf{x}_i)^T \mathbf{x}_j + c)^p \tag{26}$$

- Neural-net inspired!

$$K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\kappa \mathbf{x}_i \mathbf{x}_j - \delta)^p \tag{27}$$

- Radial Basis

$$K(\mathbf{x}_i, \mathbf{x}_j) = e^{(\gamma||\mathbf{x}_i - \mathbf{x}_j||^2)} \tag{28}$$

Section Contents

Python for SVM

```
from __future__ import division, print_function
import numpy as np
from sklearn import datasets, svm
#from sklearn.cross_validation import train_test_split
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt

from sklearn.tree import DecisionTreeClassifier
from sklearn. ensemble import RandomForestClassifier, BaggingClassifier,
    AdaBoostClassifier, VotingClassifier

iris = datasets.load_iris()
X = iris.data[:,:2]  # First two features, can take last two using [:,2:]
y = iris.target

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,
    random_state=42)
```

Python for SVM

```
def evaluate_on_test_data(model=None):
    predictions = model.predict(X_test)
    correct_classifications = 0
    for i in range(len(y_test)):
        if predictions[i] == y_test[i]:
            correct_classifications += 1
    accuracy = 100*correct_classifications/len(y_test) #Accuracy as a percentage
    return accuracy

kernels = ('linear','poly','rbf')
accuracies = []
for index, kernel in enumerate(kernels):
    model = svm.SVC(kernel=kernel)
    model.fit(X_train, y_train)
    acc = evaluate_on_test_data(model)
    accuracies.append(acc)
    print("{} % Test accuracy obtained with kernel = {}".format(acc, kernel))
```

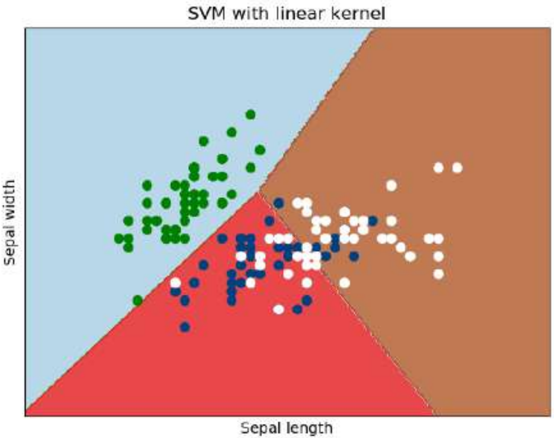
Python for SVM

```
#Train SVMs with different kernels
svc = svm.SVC(kernel='linear').fit(X_train, y_train)
rbf_svc = svm.SVC(kernel='rbf', gamma=0.7).fit(X_train, y_train)
poly_svc = svm.SVC(kernel='poly', degree=9).fit(X_train, y_train)



#Create a mesh to plot in
h = .02  # step size in the mesh
x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, h),
                     np.arange(y_min, y_max, h))

#Define title for the plots
titles = ['SVM with linear kernel',
          'SVM with RBF kernel',
          'SVM with polynomial (degree 9) kernel']
```

Python for SVM

```python
for i, clf in enumerate((svc, rbf_svc, poly_svc)):
    # Plot the decision boundary. For that, we will assign a color to each
    # point in the mesh [x_min, m_max]x[y_min, y_max].
    plt.figure(i)

    Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])
    # Put the result into a color plot
    Z = Z.reshape(xx.shape)
    plt.contourf(xx, yy, Z, cmap=plt.cm.Paired, alpha=0.8)

    # Plot also the training points
    plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.ocean)
    plt.xlabel('Sepal length')
    plt.ylabel('Sepal width')
    plt.xlim(xx.min(), xx.max())
    plt.ylim(yy.min(), yy.max())
    plt.xticks(())
    plt.yticks(())
    plt.title(titles[i])
plt.show()
```

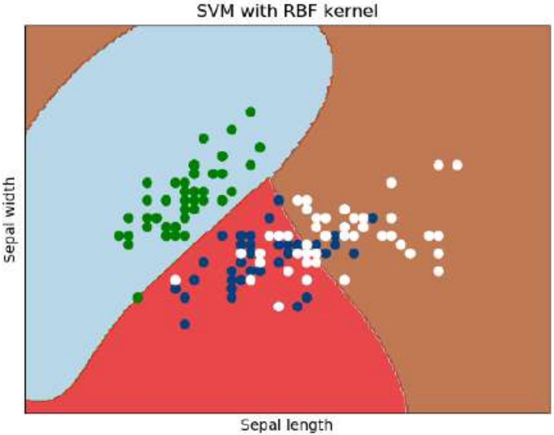
SVM Visulaization



SVM with Linear Kernel (No transformation)

$$K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i{}^T \mathbf{x}_j + c$$

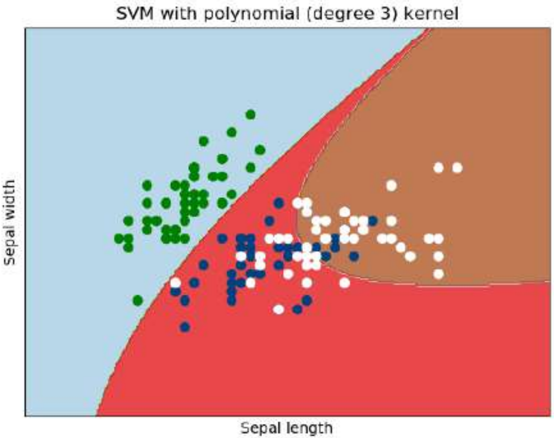
SVM Visulaization



SVM with RBF

$$K(\mathbf{x}_i, \mathbf{x}_j) = e^{\frac{-\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}}$$

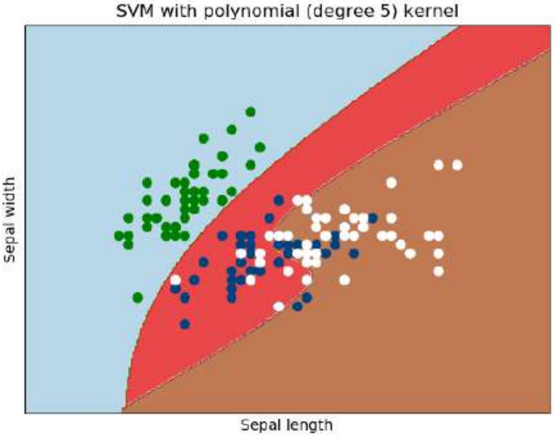
SVM Visulaization



SVM with Polynomial Kernel (Degree 3)

$$K(\mathbf{x}_i, \mathbf{x}_j) = ((\mathbf{x}_i)^T \mathbf{x}_j + c)^p$$

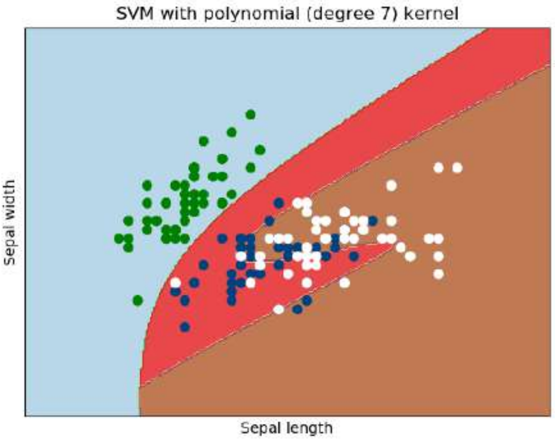
SVM Visulaization



SVM with Polynomial Kernel (Degree 5)

$$K(\mathbf{x}_i, \mathbf{x}_j) = ((\mathbf{x}_i)^T \mathbf{x}_j + c)^p$$

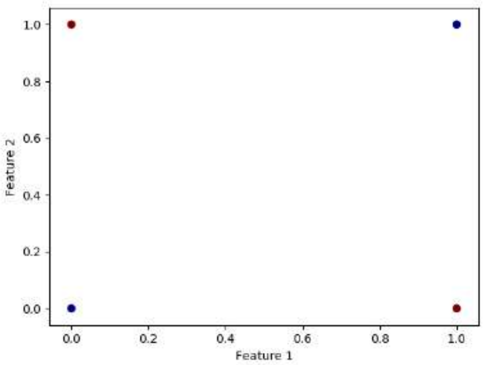
SVM Visulaization



SVM with Polynomial Kernel (Degree 7)

$$K(\mathbf{x}_i, \mathbf{x}_j) = ((\mathbf{x}_i)^T \mathbf{x}_j + c)^p$$

XOR problem

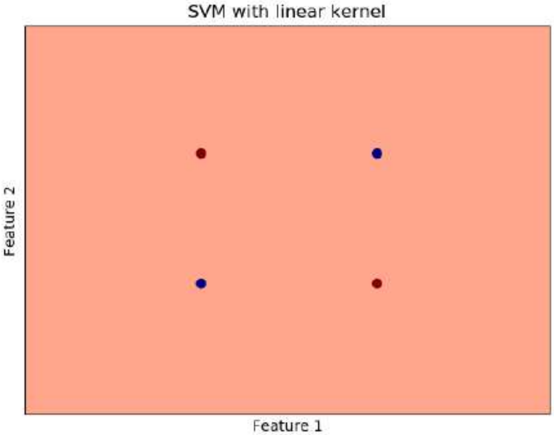


Video [4]

- XOR data.
- Problem is non-linearly separable in the given feature space.

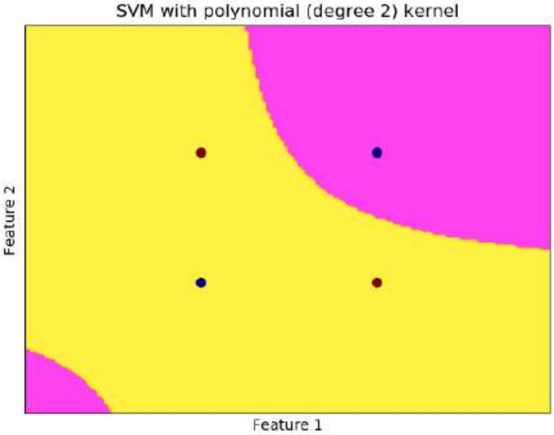
[4] https://youtu.be/5KIYu3zKvqo

XOR problem



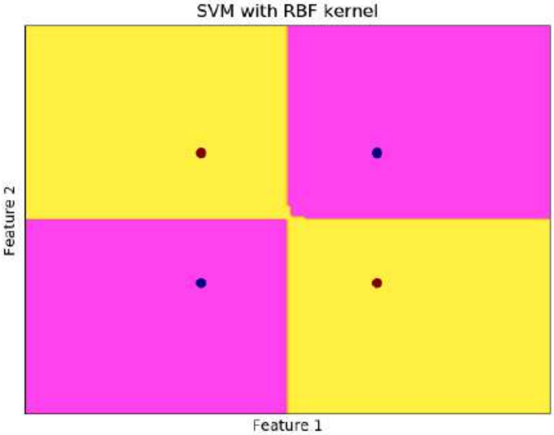
SVM with Linear Kernel (No transformation)

XOR problem



SVM with Polynomial Kernel

XOR problem



SVM with RBF Kernel

Conclusion

- Powerful theoretical grounds.
- Global, unique solution (convex optimization function).
- Performance depends on choice of kernel and parameters.
- Training is memory-intensive.
- Complexity dependent on the number of support vectors.

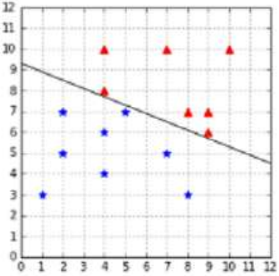
Video [5]

[5] https://youtu.be/FxLIsbnp_5c

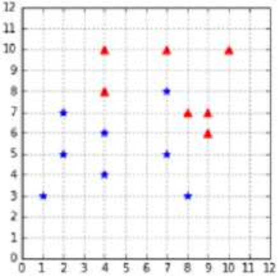
Section Contents

Need for soft Margin SVM

- Real-life data is often noisy, thus linear separability is an issue.
- Even when the data is linearly separable, the outlier can be closer to the other examples than most of the examples of its class, thus reducing the margin, or it can be among the other examples and break linear separability.



Outlier reducing the margin



Outlier breaks linear separability *6

- In this case, there is no solution to the optimization problem solved earlier.

[6] Image taken from SVM Succinctly

Slack variable

- In 1995, Vapnik and Cortes introduced a modified version of the original SVM that allows the classifier to make some mistakes.
- The goal is now not to make zero classification mistakes, but to make as few mistakes as possible.
- Constraints of the optimization problem was modified, so the constraint (refer Eq 4)

$$y_i(\bar{W} \cdot \bar{X}_i + b) \geq 1$$

becomes

$$y_i(\bar{W} \cdot \bar{X}_i + b) \geq 1 - \xi_i \tag{29}$$

where: ξ= slack variable and $\forall_i \geq 0$.

Slack variable

- In 1995, Vapnik and Cortes introduced a modified version of the original SVM that allows the classifier to make some mistakes.
- The goal is now not to make zero classification mistakes, but to make as few mistakes as possible.
- Constraints of the optimization problem was modified, so the constraint (refer Eq 4)

$$y_i(\bar{W} \cdot \bar{X}_i + b) \geq 1$$

becomes

$$y_i(\bar{W} \cdot \bar{X}_i + b) \geq 1 - \xi_i \tag{29}$$

where: ξ= slack variable and $\forall_i \geq 0$.

- ξ is subtracted from 1, in order to make it possible to satisfy constraint.

Slack variable

- The problem is that we could choose a huge value of ξ for every example, and all the constraints will be satisfied.
- To avoid this, we need to modify the objective function (refer Eq. 14 for objective function of hard margin SVM) to penalize the choice of a big ξ:

$$\underset{W,b,\xi}{\operatorname{argmin}} \frac{1}{2}||W||^2 + C\sum_{i=1}^{n}\xi_i \tag{30}$$

subject to

$$y_i(\bar{W}\cdot\bar{X}_i + b) \geq 1 - \xi_i$$

and $\xi_i \geq 0 \, \forall_i$

Slack variable

- Consider:

$$\underset{W,b,\xi}{\text{argmin}} \frac{1}{2}||W||^2 + C\sum_{i=1}^{n} \xi_i$$

The slack variable ξ_i allows the input xi to be closer to the hyperplane (or even be on the wrong side), but there is a penalty in the objective function for such "slack".

- How to select hyper-parameter C?

Slack variable

- Consider:

$$\underset{W,b,\xi}{\operatorname{argmin}} \frac{1}{2}||W||^2 + C\sum_{i=1}^{n}\xi_i$$

  The slack variable ξ_i allows the input xi to be closer to the hyperplane (or even be on the wrong side), but there is a penalty in the objective function for such "slack".

- How to select hyper-parameter C?

Value of C (hyper-parameter tuning)

1. If C is very large (penalty is higher), the SVM becomes very strict and tries to classify all data points correctly.

Slack variable

- Consider:

$$\underset{W,b,\xi}{\text{argmin}}\ \frac{1}{2}||W||^2 + C\sum_{i=1}^{n}\xi_i$$

  The slack variable ξ_i allows the input xi to be closer to the hyperplane (or even be on the wrong side), but there is a penalty in the objective function for such “slack”.

- How to select hyper-parameter C?

Value of C (hyper-parameter tuning)

1. If C is very large (penalty is higher), the SVM becomes very strict and tries to classify all data points correctly.
2. If C is very small, the SVM becomes very loose and may “sacrifice” some points to obtain a simpler solution.

Slack variable

- Consider:

$$\underset{W,b,\xi}{\operatorname{argmin}} \frac{1}{2}||W||^2 + C\sum_{i=1}^{n}\xi_i$$

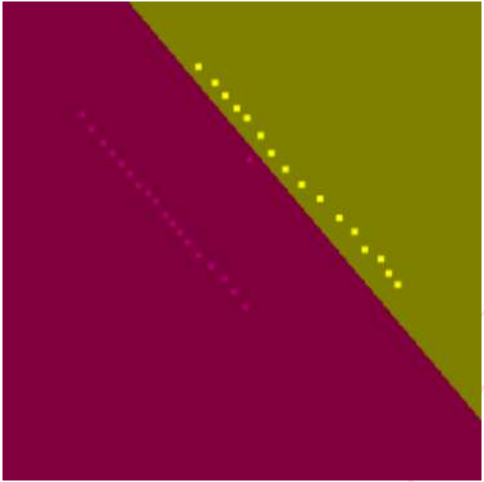
The slack variable ξ_i allows the input xi to be closer to the hyperplane (or even be on the wrong side), but there is a penalty in the objective function for such “slack”.

- How to select hyper-parameter C?

Value of C (hyper-parameter tuning)

1. If C is very large (penalty is higher), the SVM becomes very strict and tries to classify all data points correctly.
2. If C is very small, the SVM becomes very loose and may “sacrifice” some points to obtain a simpler solution.
3. Usually telescopic / grid search is applied to find best C for the given dataset.

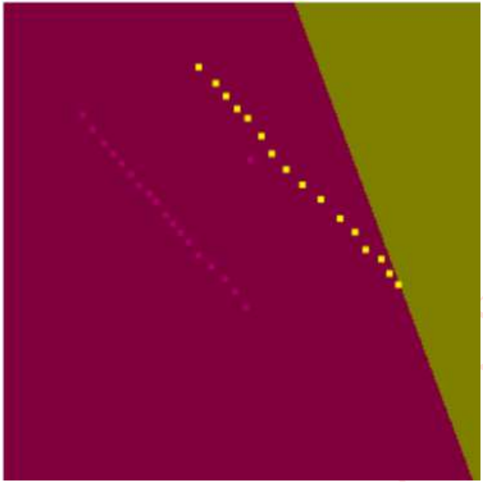
Ocular Proof



This image corresponds to large value of C.

[7]Demo images from LIBSVM website: https://www.csie.ntu.edu.tw/~cjlin/libsvm/

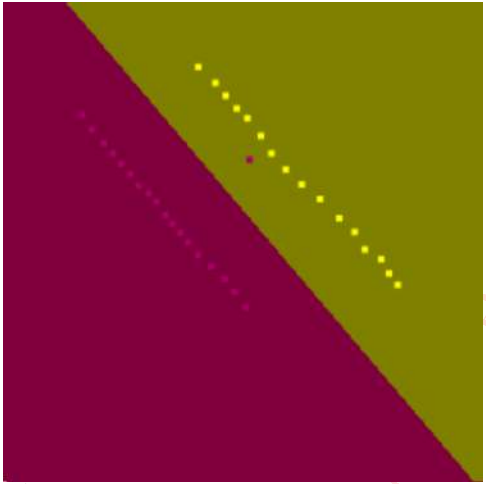
Ocular Proof



This image corresponds to small value of C, over-simplification of solution.

[7] Demo images from LIBSVM website: `https://www.csie.ntu.edu.tw/~cjlin/libsvm/`

Ocular Proof


*7

This image corresponds to appropriate value of C given dataset (maximizing margin, sacrificing few (one data point) to obtain wider margin / better generalization).

[7] Demo images from LIBSVM website: `https://www.csie.ntu.edu.tw/~cjlin/libsvm/`

Section Contents

Optimizing function with constraint

- Maximize $f(x,y) = x^2y$ on the set $x^2 + y^2 = 1$.

Optimizing function with constraint

- Maximize $f(x, y) = x^2 y$ on the set $x^2 + y^2 = 1$.

- Here we are trying to
  - Optimize multi-variable function $f(x, y) = x^2 y$
  - with the constraint $g(x, y)$ that $x^2 + y^2 = 1$ (unit circle)

Optimizing function with constraint

- Maximize $f(x, y) = x^2y$ on the set $x^2 + y^2 = 1$.

- Here we are trying to
  - Optimize multi-variable function $f(x, y) = x^2y$
  - with the constraint $g(x, y)$ that $x^2 + y^2 = 1$ (unit circle)

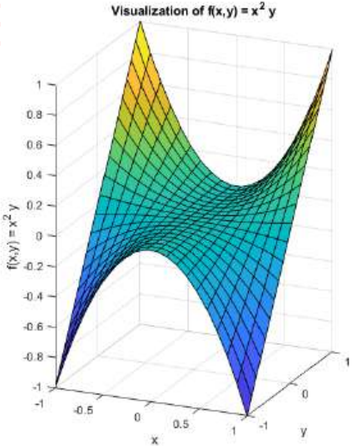
Visualization

Optimizing function with constraint

- Maximize $f(x,y) = x^2y$ on the set $x^2 + y^2 = 1$.

- Here we are trying to
  - Optimize multi-variable function $f(x,y) = x^2y$
  - with the constraint $g(x,y)$ that $x^2 + y^2 = 1$ (unit circle)

Visualization



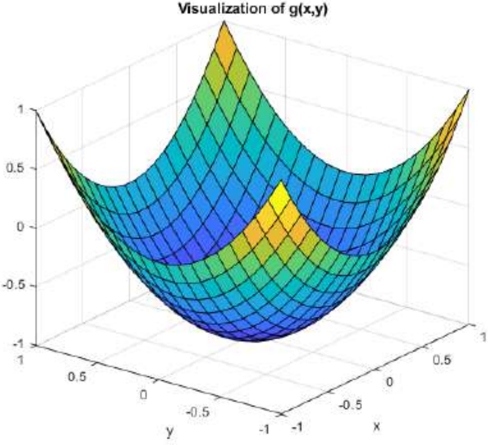
The function is in 3-D. To analytically examine the problem , we can use contour plot.

Optimizing function with constraint

- Maximize $f(x, y) = x^2y$ on the set $x^2 + y^2 = 1$.

- Here we are trying to
  - Optimize multi-variable function $f(x, y) = x^2y$
  - with the constraint $g(x, y)$ that $x^2 + y^2 = 1$ (unit circle)

Visualization

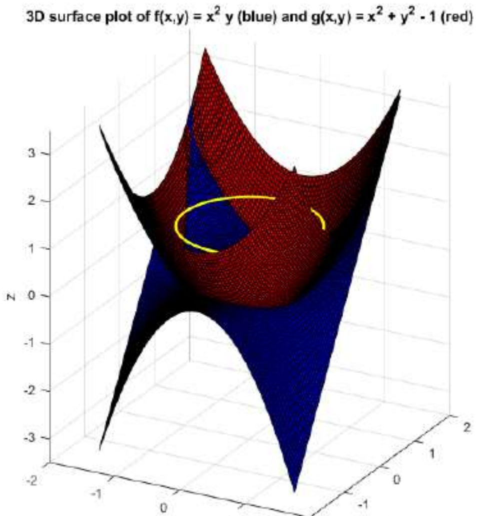


Optimizing function with constraint

- Maximize $f(x, y) = x^2 y$ on the set $x^2 + y^2 = 1$.

- Here we are trying to
  - Optimize multi-variable function $f(x, y) = x^2 y$
  - with the constraint $g(x, y)$ that $x^2 + y^2 = 1$ (unit circle)

Visualization

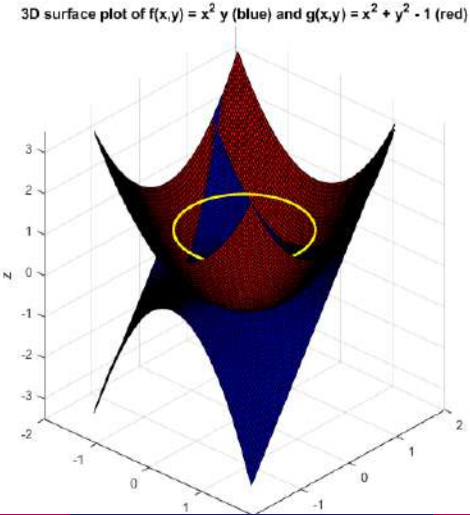


Optimizing function with constraint

- Maximize $f(x, y) = x^2y$ on the set $x^2 + y^2 = 1$.

- Here we are trying to
  - Optimize multi-variable function $f(x, y) = x^2y$
  - with the constraint $g(x, y)$ that $x^2 + y^2 = 1$ (unit circle)

Visualization

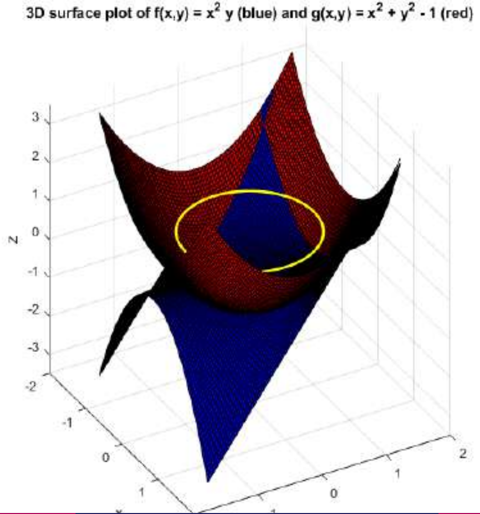


Optimizing function with constraint

Visualization

- Maximize $f(x, y) = x^2y$ on the set $x^2 + y^2 = 1$.

- Here we are trying to
  - Optimize multi-variable function $f(x, y) = x^2y$
  - with the constraint $g(x, y)$ that $x^2 + y^2 = 1$ (unit circle)



Optimizing function with constraint

- Maximize $f(x, y) = x^2y$ on the set $x^2 + y^2 = 1$.

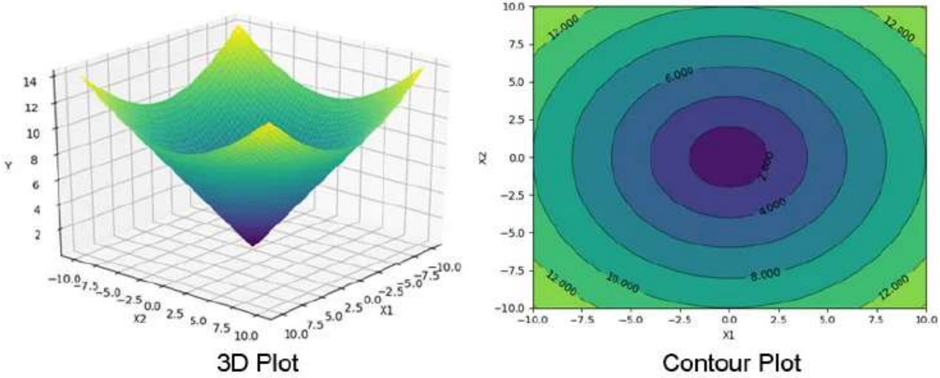
- Here we are trying to
  - Optimize multi-variable function $f(x, y) = x^2y$
  - with the constraint $g(x, y)$ that $x^2 + y^2 = 1$ (unit circle)

Visualization

Contour plot

A contour plot is a graphical technique for representing a 3-D surface by plotting constant z slices, called contours, on a 2-D format. That is, lines are drawn for all possible pairs of (x, y) that produce same/constant output (z).

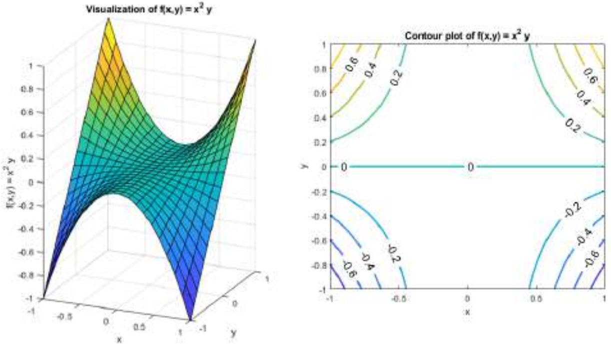
Contour Plot



Contour plot

A contour plot is a graphical technique for representing a 3-D surface by plotting constant z slices, called contours, on a 2-D format. That is, lines are drawn for all possible pairs of (x, y) that produce same/constant output (z).

Contour Plot



3D plot of function, along with corresponding contour plot of the problem in hand.

Contour plot

A contour plot is a graphical technique for representing a 3-D surface by plotting constant z slices, called contours, on a 2-D format. That is, lines are drawn for all possible pairs of (x, y) that produce same/constant output (z).

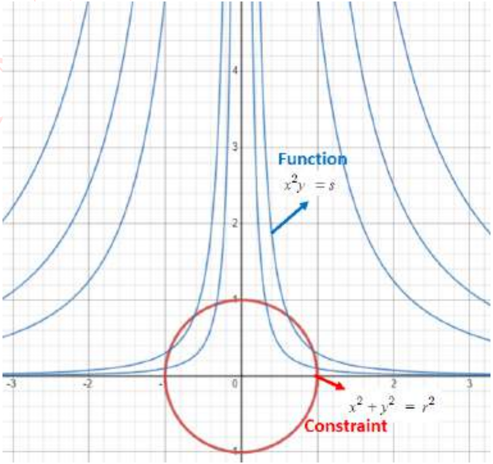
Optimizing function with constraint

- Maximize $f(x, y) = x^2y$ on the set $x^2 + y^2 = 1$.

- Here we are trying to
  - Optimize multi-variable function $f(x, y) = x^2y$
  - with the constraint that $x^2 + y^2 = 1$ (unit circle)

Visualization

Function with constraint visualization



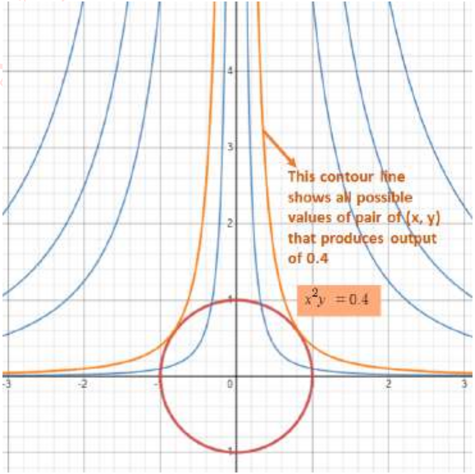
Optimizing function with constraint

- Maximize $f(x, y) = x^2y$ on the set $x^2 + y^2 = 1$.

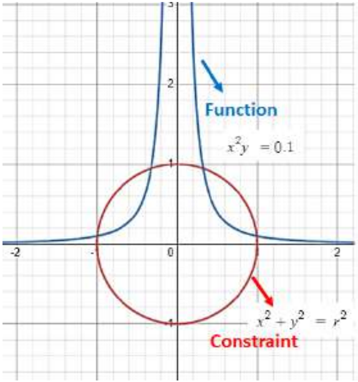
- Here we are trying to
  - Optimize multi-variable function $f(x, y) = x^2y$
  - with the constraint that $x^2 + y^2 = 1$ (unit circle)

Visualization

Function with constraint visualization

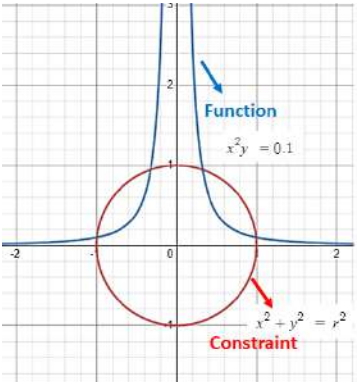


Optimizing function with constraint

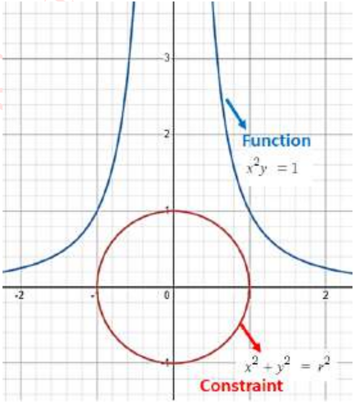


Function intersects with the constraint. This means that this pair of (x, y) satisfies constraint (four possible pair of (x, y) values), but visually we can observe that they are maximum values.

Optimizing function with constraint



Function intersects with the constraint. This means that this pair of (x, y) satisfies constraint (four possible pair of (x, y) values), but visually we can observe that they are maximum values.



Function never intersects with the constraint. This means that this pair of (x, y) is off the constraint. It also shows that, as we are max. x^2y subject to constraint, we can never go as high as these values of (x, y).

Optimizing function with constraint

- Here we are trying to

- Optimize multi-variable function $f(x, y) = x^2y$
- with the constraint that $x^2 + y^2 = 1$ (unit circle)

Optimizing function with constraint

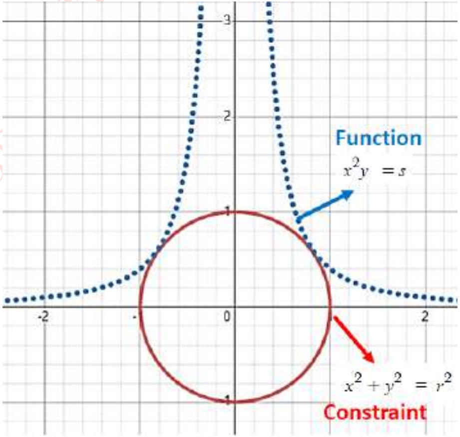
- Here we are trying to
  - Optimize multi-variable function $f(x, y) = x^2 y$
  - with the constraint that $x^2 + y^2 = 1$ (unit circle)

Objective

To maximize function $f(x, y) = x^2 y$ while satisfying constraint $x^2 + y^2 = 1$, is to find maximum value of pair of (x, y) or value of constant s to the point that afterwards its off the constraint.
This will only happen when the two functions ($f(x, y)$ & $g(x, y)$) are **tangent**.

Optimizing function with constraint

- Here we are trying to
  - Optimize multi-variable function $f(x, y) = x^2 y$
  - with the constraint that $x^2 + y^2 = 1$ (unit circle)

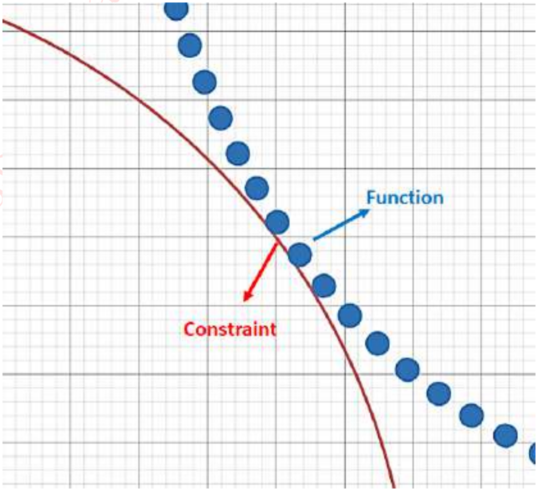


Objective

To maximize function $f(x, y) = x^2 y$ while satisfying constraint $x^2 + y^2 = 1$, is to find maximum value of pair of (x, y) or value of constant s to the point that afterwards its off the constraint.
This will only happen when the two functions ($f(x, y)$ & $g(x, y)$) are **tangent**.

Optimizing function with constraint

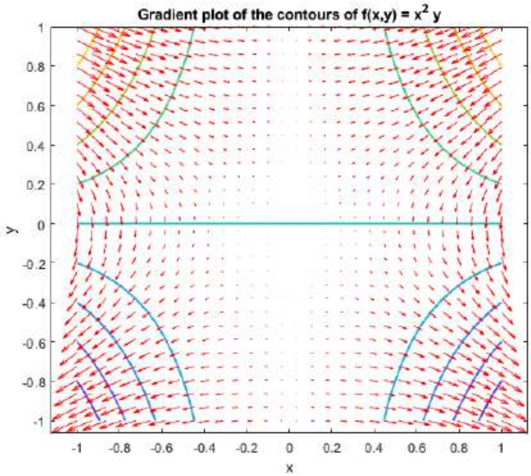
- Here we are trying to
  - Optimize multi-variable function $f(x, y) = x^2 y$
  - with the constraint that $x^2 + y^2 = 1$ (unit circle)



Objective

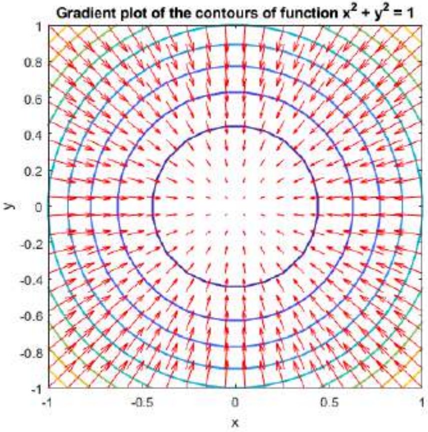
To maximize function $f(x, y) = x^2 y$ while satisfying constraint $x^2 + y^2 = 1$, is to find maximum value of pair of (x, y) or value of constant s to the point that afterwards its off the constraint.
This will only happen when the two functions ($f(x, y)$ & $g(x, y)$) are **tangent**.

Optimizing function with constraint



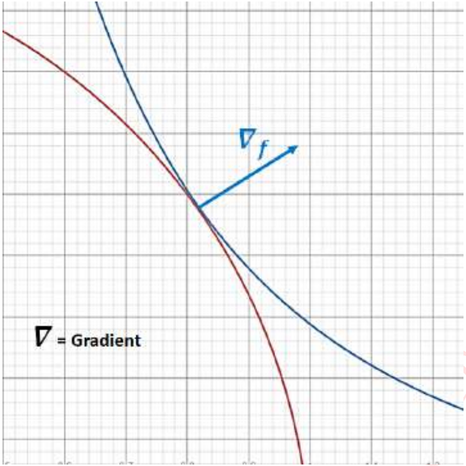
The gradient of f or g evaluated at a point (x_0, y_0) always gives a vector perpendicular to the contour line passing through that point (as there is no change in value along contour line).

Optimizing function with constraint



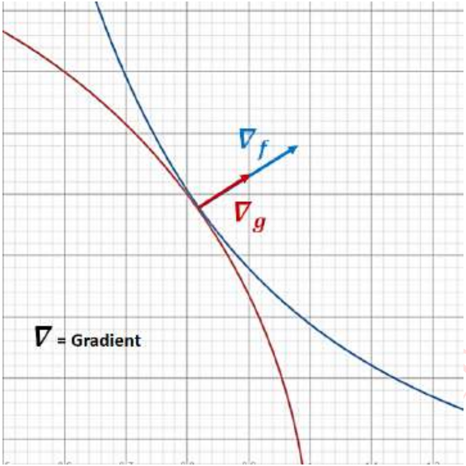
The gradient of f or g evaluated at a point (x_0, y_0) always gives a vector perpendicular to the contour line passing through that point (as there is no change in value along contour line).

Optimizing function with constraint



- The gradient of f evaluated at a point (x_0, y_0) always gives a vector perpendicular to the contour line passing through that point (as there is no change in value along contour line).
- When the contour lines of two functions f and g are tangent, their gradient vectors are parallel.

Optimizing function with constraint

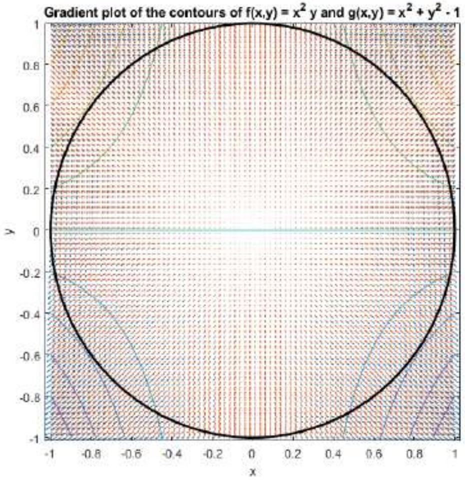


- The fact that contour lines are tangent tells us nothing about the magnitude of each of these gradient vectors, but that's okay. When two vectors point in the same direction, it means we can multiply one by some constant to get the other.
- Since this tangency means their gradient vectors align:

$$\nabla f(x,y) = \lambda \nabla g(x,y)$$

λ = Lagrange multiplier
$f(x,y)$ = Function
$g(x,y)$ = Constraint

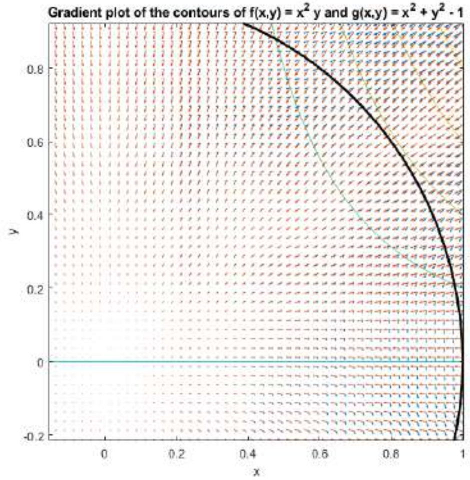
Ocular Proof



- Ocular Proof: When the contour lines of two functions f and g are tangent, their gradient vectors are parallel.
- Since this tangency means their gradient vectors align:

$$\nabla f(x,y) = \lambda \nabla g(x,y)$$

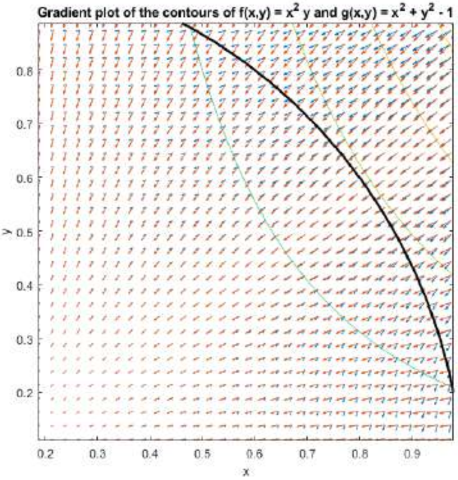
Ocular Proof



- Ocular Proof: When the contour lines of two functions f and g are tangent, their gradient vectors are parallel.
- Since this tangency means their gradient vectors align:

$$\bigtriangledown f(x,y) = \lambda \bigtriangledown g(x,y)$$

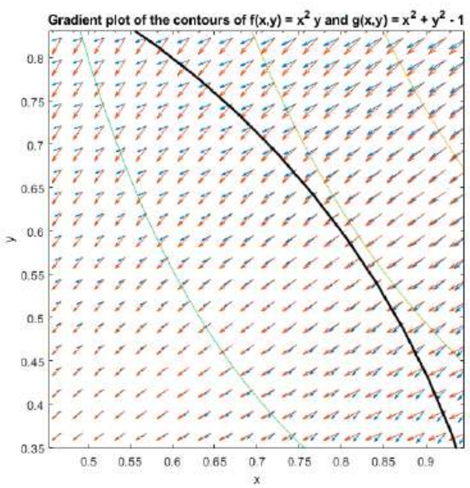
Ocular Proof



- Ocular Proof: When the contour lines of two functions f and g are tangent, their gradient vectors are parallel.
- Since this tangency means their gradient vectors align:

$$\bigtriangledown f(x,y) = \lambda \bigtriangledown g(x,y)$$

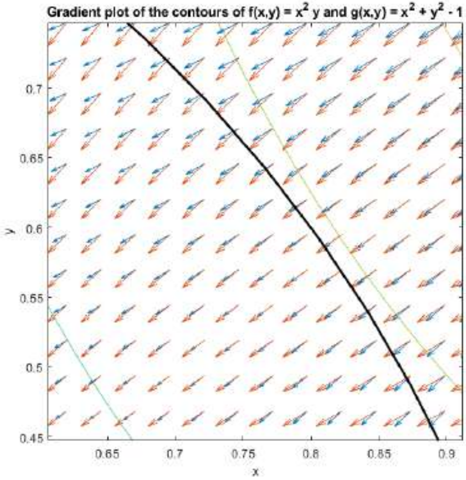
Ocular Proof



- Ocular Proof: When the contour lines of two functions f and g are tangent, their gradient vectors are parallel.
- Since this tangency means their gradient vectors align:

$$\bigtriangledown f(x,y) = \lambda \bigtriangledown g(x,y)$$

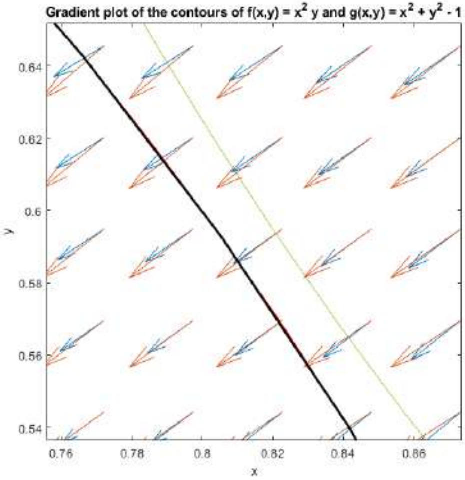
Ocular Proof



- Ocular Proof: When the contour lines of two functions f and g are tangent, their gradient vectors are parallel.
- Since this tangency means their gradient vectors align:

$$\bigtriangledown f(x, y) = \lambda \bigtriangledown g(x, y)$$

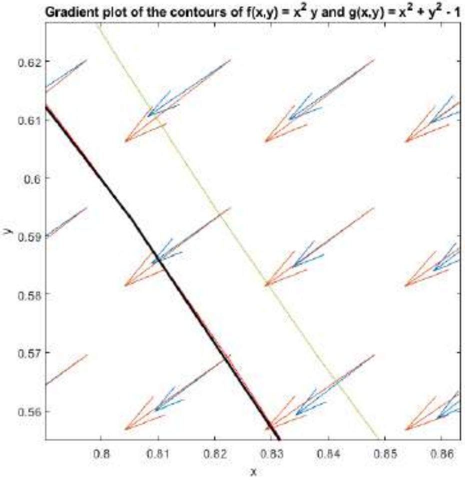
Ocular Proof



- Ocular Proof: When the contour lines of two functions f and g are tangent, their gradient vectors are parallel.
- Since this tangency means their gradient vectors align:

$$\bigtriangledown f(x,y) = \lambda \bigtriangledown g(x,y)$$

Ocular Proof



- Ocular Proof: When the contour lines of two functions f and g are tangent, their gradient vectors are parallel.
- Since this tangency means their gradient vectors align:

$$\nabla f(x,y) = \lambda \nabla g(x,y)$$

Optimizing function with constraint: Example Solution

$$
\begin{aligned}
L &= f(x, y) - \lambda g(x, y) \\
&= x^2 y - \lambda[x^2 + y^2]
\end{aligned}
$$

To find max. take derivative. First partial derivative w.r.t "x":

$$
\begin{aligned}
\frac{\partial L}{\partial x} &= 2xy - \lambda 2x \\
y &= \lambda
\end{aligned} \tag{31}
$$

Partial derivative w.r.t "y":

$$
\begin{aligned}
\frac{\partial L}{\partial y} &= x^2 - \lambda 2y \\
x^2 &= \lambda 2y \\
x^2 &= 2\lambda^2 (\text{as } y = \lambda) \\
x &= \pm\sqrt{2}\lambda
\end{aligned} \tag{32}
$$

Optimizing function with constraint: Example Solution

Putting back values of "x" and "y" found from equations 31 and 32 in the constraint equation:

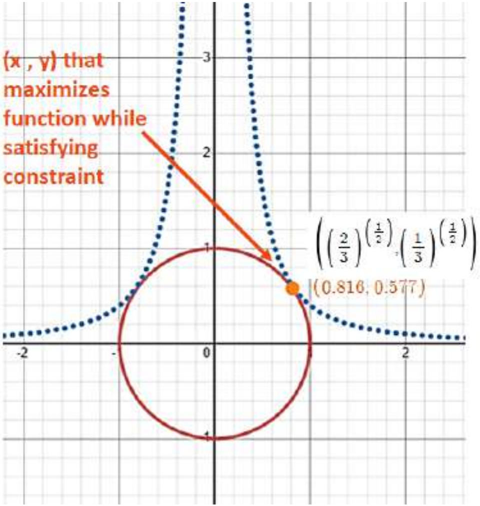
$$\begin{aligned} x^2 + y^2 &= 1 \\ [\sqrt{2}\lambda]^2 + \lambda^2 &= 1 \\ 3\lambda^2 &= 1 \\ \lambda &= \pm\sqrt{\frac{1}{3}} \end{aligned} \tag{33}$$

Put back value of λ in equations 31 and 32 to find value of x and y:

$$y = \pm\sqrt{\frac{1}{3}} \tag{34}$$

$$x = \pm\sqrt{2}\sqrt{\frac{1}{3}} = \pm\sqrt{\frac{2}{3}} \tag{35}$$

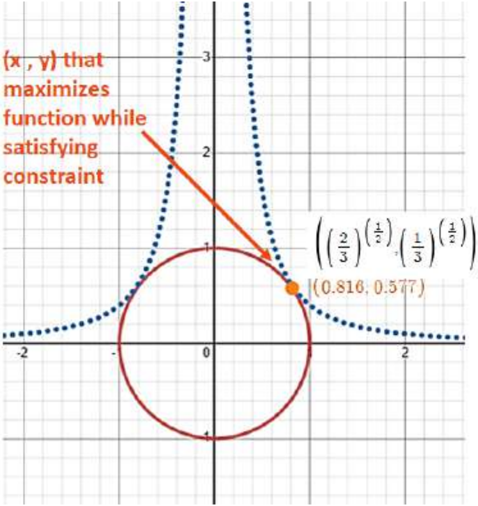
Optimizing function with constraint: Example Solution



- From equations 34 and 35 , we know values of x and y. They make four possible pairs of (x, y):

1. $(\sqrt{\frac{2}{3}}, \sqrt{\frac{1}{3}})$
2. $(-\sqrt{\frac{2}{3}}, \sqrt{\frac{1}{3}})$
3. $(\sqrt{\frac{2}{3}}, -\sqrt{\frac{1}{3}})$
4. $(-\sqrt{\frac{2}{3}}, -\sqrt{\frac{1}{3}})$

Optimizing function with constraint: Example Solution



- From equations 34 and 35 , we know values of x and y. They make four possible pairs of (x, y):

1. $(\sqrt{\frac{2}{3}}, \sqrt{\frac{1}{3}})$
2. $(-\sqrt{\frac{2}{3}}, \sqrt{\frac{1}{3}})$
3. $(\sqrt{\frac{2}{3}}, -\sqrt{\frac{1}{3}})$
4. $(-\sqrt{\frac{2}{3}}, -\sqrt{\frac{1}{3}})$

- Last two point make function (x^2y) negative (will not max. func.) and first two points gives same output and that is the maximum value function can achieve while satisfying constraint (refer image on the left).

Perceptron
A linear Classifier

Dr. Rizwan Ahmed Khan

Outline

Reference Books

Reference books for this lecture:

- Chapter 4: Machine Learning, Tom MITCHELL, McGraw Hill, latest edition.

Reference Books

Reference books for this lecture:

- Chapter 4: Machine Learning, Tom MITCHELL, McGraw Hill, latest edition.
- Chapter 5: Pattern Classification, R. DUDA et al., Wiley Interscience, latest edition.

Reference Books

Reference books for this lecture:

- Chapter 4: Machine Learning, Tom MITCHELL, McGraw Hill, latest edition.
- Chapter 5: Pattern Classification, R. DUDA et al., Wiley Interscience, latest edition.
- Chapter 3: Pattern Recognition, S. Theodoridis et al.,Academic Press, 4^{th} or latest edition.

Section Contents

Issues with K-Nearest Neighbors

- Although k-nearest neighbor is a strong classifier and can achieve good results if the number of training samples (n) are very large, but one issue that restricts to use it (for practical reason) is:

Issues with K-Nearest Neighbors

- Although k-nearest neighbor is a strong classifier and can achieve good results if the number of training samples (n) are very large, but one issue that restricts to use it (for practical reason) is:

What is computational complexity of K-Nearest Neighbors

1. Compare query data / test data to all training examples.
2. Training Complexity : $\mathcal{O}(1)$

Issues with K-Nearest Neighbors

- Although k-nearest neighbor is a strong classifier and can achieve good results if the number of training samples (n) are very large, but one issue that restricts to use it (for practical reason) is:

What is computational complexity of K-Nearest Neighbors

1. Compare query data / test data to all training examples.
2. Training Complexity : $\mathcal{O}(1)$
3. Test Complexity : $\mathcal{O}(nd)$, where n = number of training instances and d = dimensions of training data. It's linear time algorithm and that is not good!
4. Result: K-Nearest Neighbors is **slow**.

- For practical application, test time is more important that train time.

Section Contents

Historical Context

- The first artificial neural network (ANN) was invented in 1958 by psychologist Frank Rosenblatt, called Perceptron.
- It was intended to model how the human brain processed visual data and learned to recognize objects.
- Press Conference in 1958: “the embryo of an electronic computer that [the US Navy (funding agency)] expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence”.

Historical Context

- The first artificial neural network (ANN) was invented in 1958 by psychologist Frank Rosenblatt, called Perceptron.
- It was intended to model how the human brain processed visual data and learned to recognize objects.
- Press Conference in 1958: "the embryo of an electronic computer that [the US Navy (funding agency)] expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence".
- In 1969 it was proved that Perceptron could not be trained for non-linearly separable data (i.e. XOR problem). This lead to field of neural network research to stagnate for many years (almost quarter of a century – A.I winter).

Historical Context

1958 New York Times...

NEW NAVY DEVICE LEARNS BY DOING

Psychologist Shows Embryo of Computer Designed to Read and Grow Wiser

WASHINGTON, July 7 (UPI) —The Navy revealed the embryo of an electronic computer today that it expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence.

The embryo—the Weather Bureau's $2,000,000 "704" computer—learned to differentiate between right and left after fifty attempts in the Navy's demonstration for newsmen.

The service said it would use this principle to build the first of its Perceptron thinking machines that will be able to read and write. It is expected to be finished in about a year at a cost of $100,000.

Dr. Frank Rosenblatt, designer of the Perceptron, conducted the demonstration. He said the machine would be the first device to think as the human brain. As do human beings, Perceptron will make mistakes at first, but will grow wiser as it gains experience, he said.

Dr. Rosenblatt, a research psychologist at the Cornell Aeronautical Laboratory, Buffalo, said Perceptrons might be fired to the planets as mechanical space explorers.

Without Human Controls

The Navy said the perceptron would be the first non-living mechanism "capable of receiving, recognizing and identifying its surroundings without any human training or control."

The "brain" is designed to remember images and information it has perceived itself. Ordinary computers remember only what is fed into them on punch cards or magnetic tape.

Later Perceptrons will be able to recognize people and call out their names and instantly translate speech in one language to speech or writing in another language, it was predicted.

Mr. Rosenblatt said in principle it would be possible to build brains that could reproduce themselves on an assembly line and which would be conscious of their existence.

In today's demonstration, the "704" was fed two cards, one with squares marked on the left side and the other with squares on the right side.

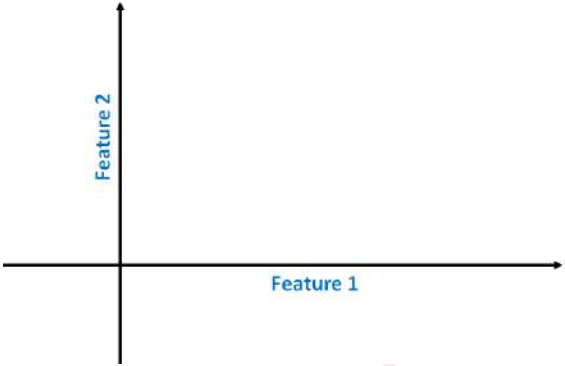
Learns by Doing

In the first fifty trials, the machine made no distinction between them. It then started registering a "Q" for the left squares and "O" for the right squares.

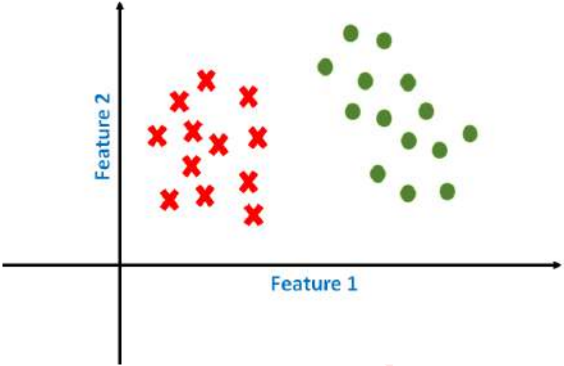
Dr. Rosenblatt said he could explain why the machine learned only in highly technical terms. But he said the computer had undergone a "self-induced change in the wiring diagram."

The first Perceptron will have about 1,000 electronic "association cells" receiving electrical impulses from an eye-like scanning device with 400 photo-cells. The human brain has 10,000,000,000 responsive cells, including 100,000,000 connections with the eyes.

Assumption



Assumption

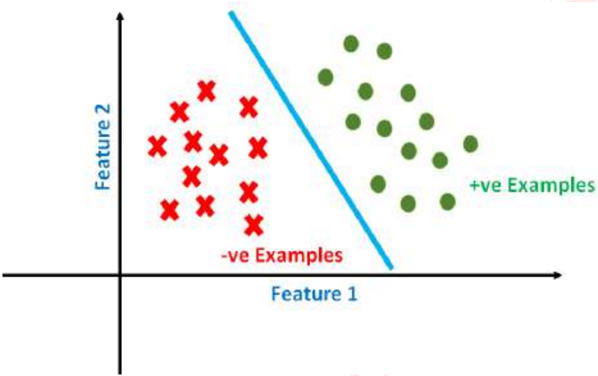


Assumptions or Bias:

- Binary classification

$$y_i \in \{-1, +1\}$$

Assumption



Assumptions or Bias:

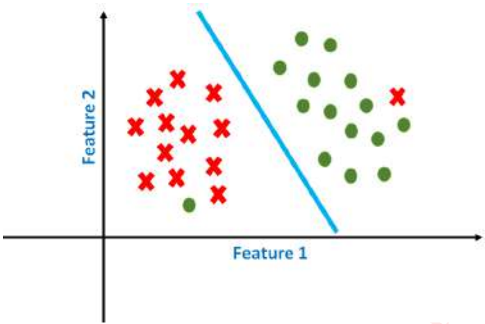
- Binary classification

$$y_i \in \{-1, +1\}$$

- There must be a hyperplane that linearly separates the data (one class from the other).

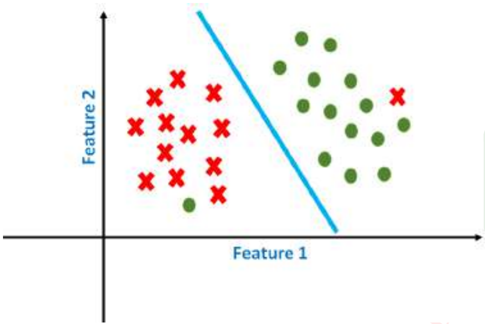
- All data points from one class lie on one side of hyperplane.

Assumption



- What will happen in this case? Now data is not linearly separable.

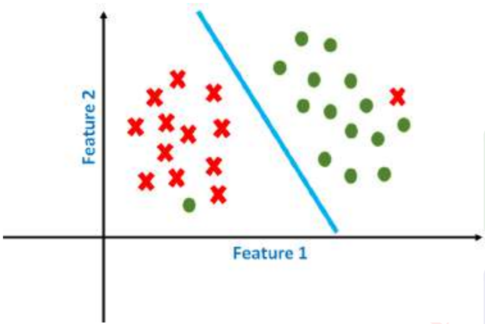
Assumption



- What will happen in this case? Now data is not linearly separable.
- In high dimensional space data points tend to be far away from each other (difficult to visualize).

In low dimensional spaces linear separability doesn't hold for long but in high dimensional space it almost holds i.e. (kernel trick).

Assumption



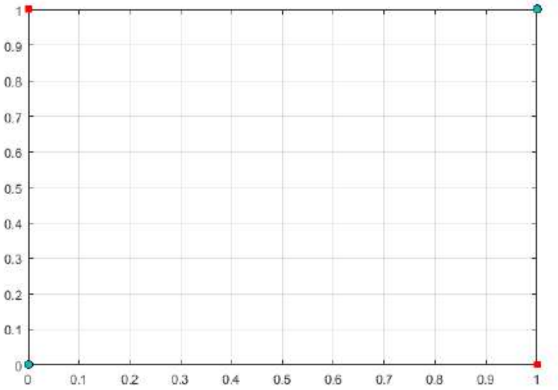
- What will happen in this case? Now data is not linearly separable.
- In high dimensional space data points tend to be far away from each other (difficult to visualize).

In low dimensional spaces linear separability doesn't hold for long but in high dimensional space it almost holds i.e. (kernel trick).

In essence Perceptron is opposite of k-NN as k-NN works better in low dimensional spaces (rem: curse of dimensionality) while Perceptron assumption holds in high dimensional spaces.

Assumption : Data in higher dimensional space

XOR Problem



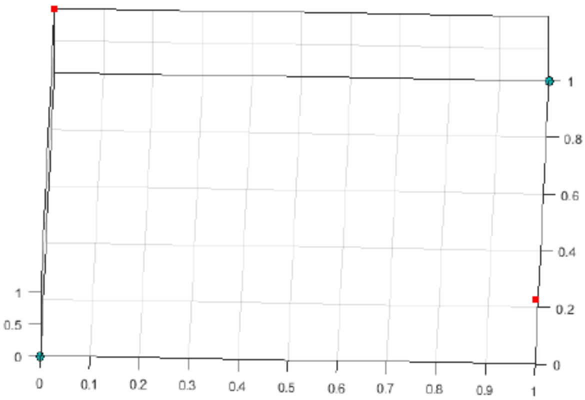
Inputs		Output
0	0	0
0	1	1
1	0	1
1	1	0

- XOR in 2D is not linearly separable but in 3D it is.
- In low dimensional spaces linear separability doesn't hold for long but in high dimensional space it almost holds i.e. (kernel trick).

XOR in 3D: `https://www.youtube.com/watch?v=5KIYu3zKvqo`

Assumption : Data in higher dimensional space

XOR Problem



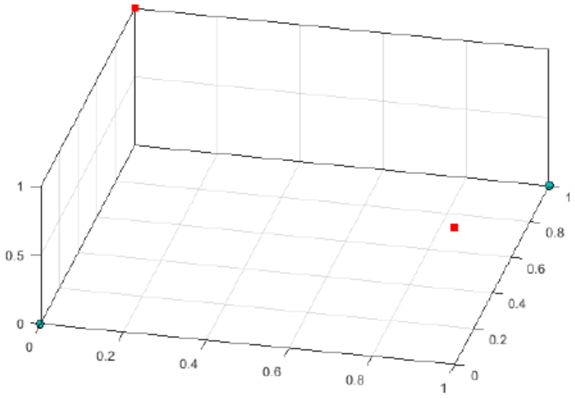
XOR in 3D: `https://www.youtube.com/watch?v=5KIYu3zKvqo`

Inputs		**Output**
0	0	0
0	1	1
1	0	1
1	1	0

- XOR in 2D is not linearly separable but in 3D it is.
- In low dimensional spaces linear separability doesn't hold for long but in high dimensional space it almost holds i.e. (kernel trick).

Assumption : Data in higher dimensional space

XOR Problem



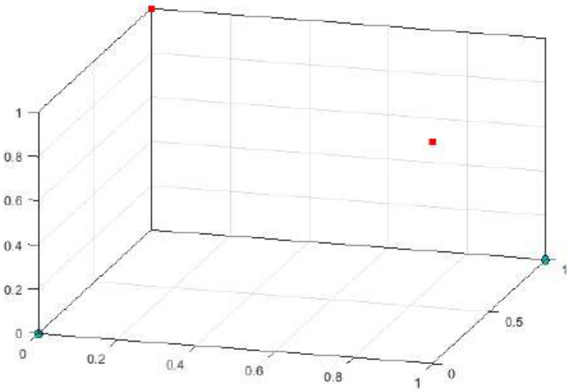
XOR in 3D: `https://www.youtube.com/watch?v=5KIYu3zKvqo`

Inputs		Output
0	0	0
0	1	1
1	0	1
1	1	0

- XOR in 2D is not linearly separable but in 3D it is.
- In low dimensional spaces linear separability doesn't hold for long but in high dimensional space it almost holds i.e. (kernel trick).

Assumption : Data in higher dimensional space

XOR Problem



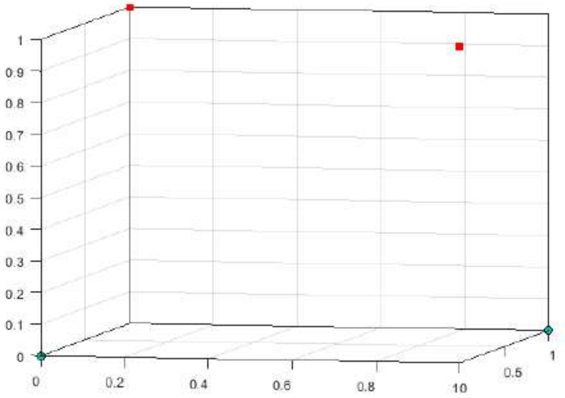
XOR in 3D: https://www.youtube.com/watch?v=5KIYu3zKvqo

Inputs		Output
0	0	0
0	1	1
1	0	1
1	1	0

- XOR in 2D is not linearly separable but in 3D it is.
- In low dimensional spaces linear separability doesn't hold for long but in high dimensional space it almost holds i.e. (kernel trick).

Assumption : Data in higher dimensional space

XOR Problem

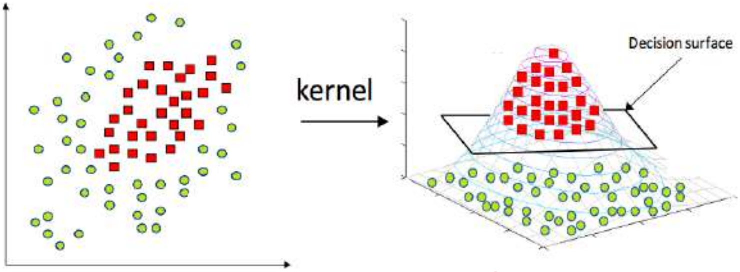


XOR in 3D: `https://www.youtube.com/watch?v=5KIYu3zKvqo`

Inputs		Output
0	0	0
0	1	1
1	0	1
1	1	0

- XOR in 2D is not linearly separable but in 3D it is.
- In low dimensional spaces linear separability doesn't hold for long but in high dimensional space it almost holds i.e. (kernel trick).

Mapping data in higher dimensional space: Kernel function



Video showing XOR data from 2D to 3D. [1]

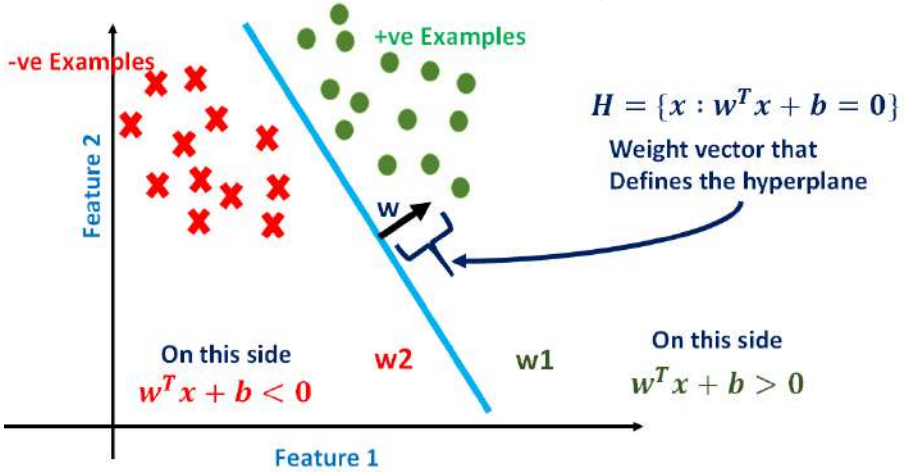
Kernel Trick [2]

- There is a little trick that can be done to transform data (change the data point without changing the data point) in a such away that it become linearly separable.
- Define function Φ that will take data point and change its dimension.

[1] https://youtu.be/5KIYu3zKvqo

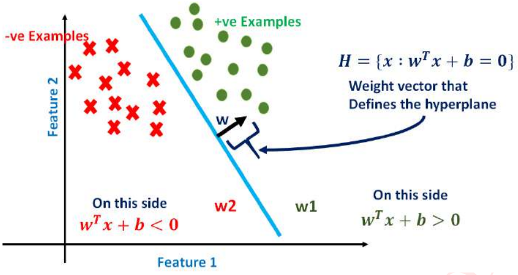
[2] Later in the course during lecture on SVM

Classifier Visualization : Defining hyperplane



- In case of difficulty in understanding equation of a hyperplane, refer Section 7.

Classifier Calculus

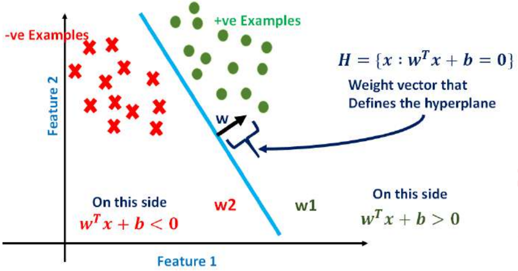


- Assuming that hyperplane exists that linearly separates data according to labels, Perceptron algorithm tries to find it.
- Mathematically hyperplane can be given by:

$$\mathcal{H} = \{x : (\bar{\mathbf{w}}^\top \bar{\mathbf{x}} + b) = 0\}$$

  where: b is the bias term (without the bias term, the hyperplane that $\mathbf{w}$ defines would always have to go through the origin).
- Learning a perceptron involves choosing values for weights $\mathbf{w}$.

Classifier Calculus



- What to do at test time? (unknown sample $\mathbf{x}_i$)

$$h(x_i) = \text{sign}(\mathbf{w}^\top \mathbf{x}_i + b)$$

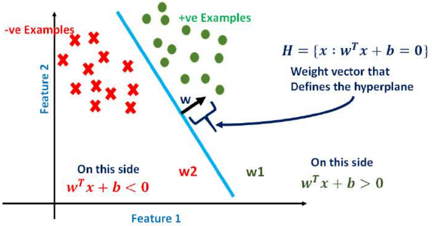
OR

$$\mathbf{w}^\top \mathbf{x} + b > 0 \quad \forall \text{ x in class1, +ve Examples}$$

$$\mathbf{w}^\top \mathbf{x} + b < 0 \quad \forall \text{ x in class2, -ve Examples}$$

- This means test time speed is constant. It's very fast.

Classifier Calculus



- Dealing with b separately is difficult (difficult for mathematical proofs and for programming), thus this term can be merged with weight vector w. Under this convention:

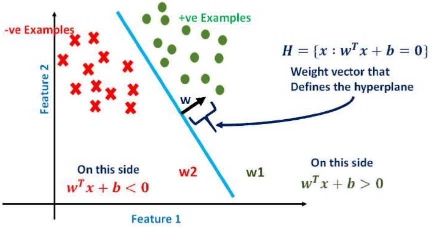
$$\mathbf{x}_i \quad \text{becomes} \quad \begin{bmatrix} \mathbf{x}_i \\ 1 \end{bmatrix}$$

$$\mathbf{w} \quad \text{becomes} \quad \begin{bmatrix} \mathbf{w} \\ b \end{bmatrix}$$

- We can verify:

$$\begin{bmatrix} \mathbf{x}_i \\ 1 \end{bmatrix} \begin{bmatrix} \mathbf{w} \\ b \end{bmatrix}^\top = \mathbf{w}^\top \mathbf{x}_i + b$$

Classifier Calculus



$\mathbf{x}_i$ becomes $\begin{bmatrix} \mathbf{x}_i \\ 1 \end{bmatrix}$

$\mathbf{w}$ becomes $\begin{bmatrix} \mathbf{w} \\ b \end{bmatrix}$

Now we can say:

$$\mathcal{H} = \{x : (\bar{\mathbf{w}}^\top \mathbf{x}) = 0\}$$

Rem: We absorbed b with w, in essence b is offset and w is orientation of hyperplane.

Section Contents

Perceptron Learning Algorithm

Algorithm 1 Perceptron Learning Algorithm

Result: Learned Hyperplane / Decision Boundary

initialization $\vec{w} = 0$

while *TRUE* **do**

    missClassification $= 0$

    for $(x_i, y_i) \in D$ **do**

        if $y_i(\vec{w}^\top \vec{x_i}) \leq 0$ **then**

            $\vec{w} \leftarrow \vec{w} + y\vec{x}$

            $missClassification \leftarrow missClassification + 1$

        end

    end

    if *missClassification = 0* **then**

        break

    end

end

Perceptron Learning Algorithm

- In algorithm, what this statement specifies?

$$y_i(\vec{w}^\top \vec{x_i}) \leq 0 \quad (1)$$

- Remember: We are dealing binary classification $y_i \in \{-1, +1\}$
  And

$$\vec{w}^\top \mathbf{x} > 0 \quad \forall \text{ +ve Examples} \quad (2)$$

$$\vec{w}^\top \mathbf{x} < 0 \quad \forall \text{ -ve Examples} \quad (3)$$

Perceptron Learning Algorithm

- In algorithm, what this statement specifies?

$$y_i(\vec{w}^\top \vec{x_i}) \leq 0 \tag{1}$$

- Remember: We are dealing binary classification $y_i \in \{-1, +1\}$
  And

$$\vec{w}^\top \mathbf{x} > 0 \quad \forall \text{ +ve Examples} \tag{2}$$

$$\vec{w}^\top \mathbf{x} < 0 \quad \forall \text{ -ve Examples} \tag{3}$$

By combining Equations 2 and 3, we can write:

$$y_i(\vec{w}^\top \vec{x_i}) \geq 0 \tag{4}$$

Perceptron Learning Algorithm

- In algorithm, what this statement specifies?

$$y_i(\vec{w}^\top \vec{x_i}) \leq 0 \tag{1}$$

- Remember: We are dealing binary classification $y_i \in \{-1, +1\}$
  And

$$\vec{w}^\top \mathbf{x} > 0 \quad \forall \text{ +ve Examples} \tag{2}$$

$$\vec{w}^\top \mathbf{x} < 0 \quad \forall \text{ -ve Examples} \tag{3}$$

By combining Equations 2 and 3, we can write:

$$y_i(\vec{w}^\top \vec{x_i}) \geq 0 \tag{4}$$

Proof:

1. $y_i(\vec{w}^\top \vec{x_i}) \geq 0$, $y_i = +1$ for +ve samples
   $+1(\vec{w}^\top \vec{x_i}) \geq 0 \implies (\vec{w}^\top \vec{x_i}) \geq 0$
   same as Equation 2

Perceptron Learning Algorithm

- In algorithm, what this statement specifies?

$$y_i(\vec{w}^\top \vec{x_i}) \leq 0 \qquad (1)$$

- Remember: We are dealing binary classification $y_i \in \{-1, +1\}$
  And

$$\vec{w}^\top \mathbf{x} > 0 \quad \forall \text{ +ve Examples} \qquad (2)$$

$$\vec{w}^\top \mathbf{x} < 0 \quad \forall \text{ -ve Examples} \qquad (3)$$

By combining Equations 2 and 3, we can write:

$$y_i(\vec{w}^\top \vec{x_i}) \geq 0 \qquad (4)$$

Proof:

1. $y_i(\vec{w}^\top \vec{x_i}) \geq 0$, $y_i = +1$ for +ve samples
   $+1(\vec{w}^\top \vec{x_i}) \geq 0 \implies (\vec{w}^\top \vec{x_i}) \geq 0$
   same as Equation 2
2. $y_i(\vec{w}^\top \vec{x_i}) \geq 0$, y_i = -1 for -ve samples
   $-1(\vec{w}^\top \vec{x_i}) \geq 0 \implies (\vec{w}^\top \vec{x_i}) \leq 0$
   same as Equation 3

Perceptron Learning Algorithm

- Again, in perceptron learning algorithm, what this statement (refer Equation 1) specifies?

$$y_i(\vec{w}^\top \vec{x_i}) \leq 0$$

Perceptron Learning Algorithm

- Again, in perceptron learning algorithm, what this statement (refer Equation 1) specifies?

$$y_i(\vec{w}^\top \vec{x_i}) \leq 0$$

This shows a misclassification!

Perceptron Learning Algorithm

- Again, in perceptron learning algorithm, what this statement (refer Equation 1) specifies?

$$y_i(\vec{w}^\top \vec{x_i}) \leq 0$$

This shows a misclassification!

-

$$\vec{w} \leftarrow \vec{w} + y\vec{x}$$

This is weight update rule.

Perceptron Learning Algorithm

- Again, in perceptron learning algorithm, what this statement (refer Equation 1) specifies?

$$y_i(\vec{w}^\top \vec{x_i}) \leq 0$$

This shows a misclassification!

- $$\vec{w} \leftarrow \vec{w} + y\vec{x}$$

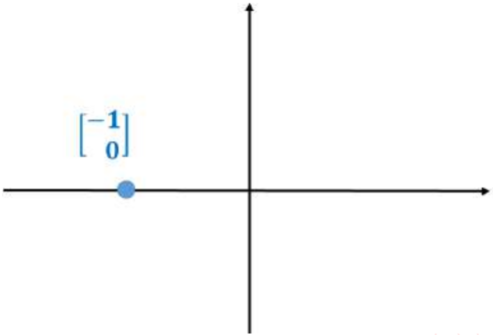
This is weight update rule.

1. if misclassified sample is from $+1$ class then add in $\vec{w}$ amount proportional to $\vec{x}$
2. if misclassified sample is from -1 class then subtract in $\vec{w}$ amount proportional to $\vec{x}$

- The algorithm belongs to a more general algorithmic family known as reward and punishment schemes.

Example : Perceptron Learning Algorithm

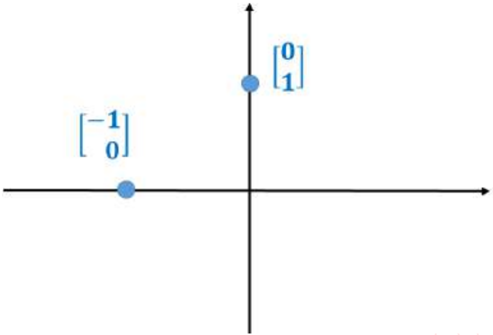
- Design a linear classifier using the perceptron algorithm



- Consider four data points (first two points belong to class $w1$, while other two belongs to class $w2$):
  1. $\begin{bmatrix} -1 \\ 0 \end{bmatrix}$

Example : Perceptron Learning Algorithm

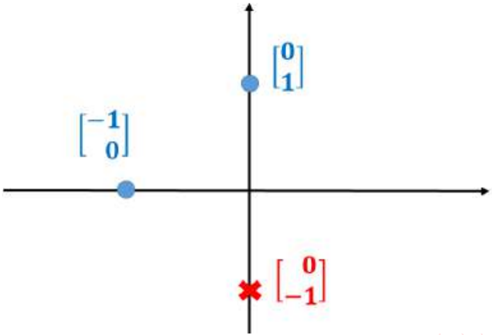
- Design a linear classifier using the perceptron algorithm



- Consider four data points (first two points belong to class $w1$, while other two belongs to class $w2$):
  1. $\begin{bmatrix}-1\\0\end{bmatrix}$
  2. $\begin{bmatrix}0\\1\end{bmatrix}$

Example : Perceptron Learning Algorithm

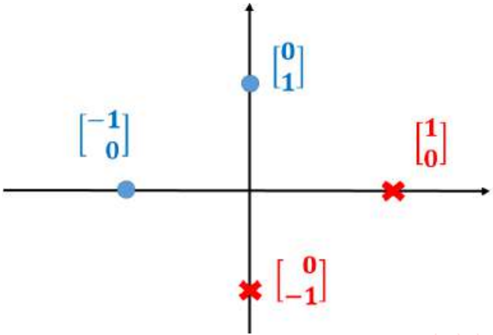
- Design a linear classifier using the perceptron algorithm



- Consider four data points (first two points belong to class $w1$, while other two belongs to class $w2$):
  1. $\begin{bmatrix}-1\\0\end{bmatrix}$
  2. $\begin{bmatrix}0\\1\end{bmatrix}$
  3. $\begin{bmatrix}0\\-1\end{bmatrix}$

Example : Perceptron Learning Algorithm

- Design a linear classifier using the perceptron algorithm



- Consider four data points (first two points belong to class $w1$, while other two belongs to class $w2$):
  1. $\begin{bmatrix}-1\\0\end{bmatrix}$
  2. $\begin{bmatrix}0\\1\end{bmatrix}$
  3. $\begin{bmatrix}0\\-1\end{bmatrix}$
  4. $\begin{bmatrix}1\\0\end{bmatrix}$
- Consider initial weight vector is chosen as $w(0) = \begin{bmatrix}0\\0\\0\end{bmatrix}$ in extended 3D space i.e. merged w and b.

Example : Perceptron Learning Algorithm

1. Consider first data point $\begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$, find $\vec{w}^T\vec{x}$

Example : Perceptron Learning Algorithm

1. Consider first data point $\begin{bmatrix}-1\\0\\1\end{bmatrix}$, find $\vec{w}^T\vec{x}$

$= \begin{bmatrix}0 & 0 & 0\end{bmatrix} \begin{bmatrix}-1\\0\\1\end{bmatrix} = 0$ (Miss-classification, result should be > 0 for $w1$ samples)

Example : Perceptron Learning Algorithm

1. Consider first data point $\begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$, find $\vec{w}^T\vec{x}$

$= [0\ 0\ 0] \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} = 0$ (Miss-classification, result should be > 0 for $w1$ samples)

- update rule, $w: \vec{w} \leftarrow \vec{w} + y\vec{x}$

Example : Perceptron Learning Algorithm

1. Consider first data point $\begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$, find $\vec{w}^T\vec{x}$

$$= \begin{bmatrix} 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} = 0 \text{ (Miss-classification, result should be } > 0 \text{ for } w1 \text{ samples)}$$

- update rule, w : $\vec{w} \leftarrow \vec{w} + y\vec{x}$

$$w\vec{(1)} \leftarrow \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + (1) \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} \text{ (This is updated } w\text{)}$$

Example : Perceptron Learning Algorithm

1. Consider first data point $\begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$, find $\vec{w}^T\vec{x}$

$$= \begin{bmatrix} 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} = 0 \text{ (Miss-classification, result should be } > 0 \text{ for } w1 \text{ samples)}$$

- update rule, w : $\vec{w} \leftarrow \vec{w} + y\vec{x}$

$$\vec{w(1)} \leftarrow \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + (1) \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} \text{ (This is updated } w\text{)}$$

2. Consider second data point $\begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}$, find $\vec{w}^T\vec{x}$

Example : Perceptron Learning Algorithm

1. Consider first data point $\begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$, find $\vec{w}^T\vec{x}$

$= \begin{bmatrix} 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} = 0$ (Miss-classification, result should be > 0 for $w1$ samples)

- update rule, w : $\vec{w} \leftarrow \vec{w} + y\vec{x}$

$\vec{w(1)} \leftarrow \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + (1) \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$ (This is updated w)

2. Consider second data point $\begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}$, find $\vec{w}^T\vec{x}$

$= \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} = 1 > 0$ (Correct as $\vec{w}^T\vec{x} > 0$ for $w1$ samples, no update in $\vec{w(1)}$ required, $\vec{w(2)} = \vec{w(1)}$)

Example : Perceptron Learning Algorithm

3. Consider third data point $\begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix}$, find $\vec{w}^T\vec{x}$

Example : Perceptron Learning Algorithm

3. Consider third data point $\begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix}$, find $\vec{w}^T\vec{x}$

$$\begin{bmatrix} -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix} = 1 > 0$$ (Miss-classification, result should be < 0 for $w2$ samples)

Example : Perceptron Learning Algorithm

3. Consider third data point $\begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix}$, find $\vec{w}^T\vec{x}$

$[-1\ 0\ 1]\begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix} = 1 > 0$ (Miss-classification, result should be < 0 for $w2$ samples)

- update rule, w : $\vec{w} \leftarrow \vec{w} + y\vec{x}$

$$\vec{w(3)} \leftarrow \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} - 1\begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix} = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} - \begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix} = \begin{bmatrix} -1 \\ 1 \\ 0 \end{bmatrix}$$ (This is updated $\vec{w(3)}$)

Example : Perceptron Learning Algorithm

3. Consider third data point $\begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix}$, find $\vec{w}^T\vec{x}$

$$[-1\ 0\ 1]\begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix} = 1 > 0 \text{ (Miss-classification, result should be } < 0 \text{ for } w2 \text{ samples)}$$

- update rule, $w:\ \vec{w} \leftarrow \vec{w} + y\vec{x}$

$$w(\vec{3}) \leftarrow \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} - 1\begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix} = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} - \begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix} = \begin{bmatrix} -1 \\ 1 \\ 0 \end{bmatrix} \text{ (This is updated } w(\vec{3}))$$

4. Consider fourth data point $\begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$, find $\vec{w}^T\vec{x}$

Example : Perceptron Learning Algorithm

3. Consider third data point $\begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix}$, find $\vec{w}^T \vec{x}$

$$[-1\ 0\ 1] \begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix} = 1 > 0$$ (Miss-classification, result should be < 0 for $w2$ samples)

- update rule, w : $\vec{w} \leftarrow \vec{w} + y\vec{x}$

$$w(\vec{3}) \leftarrow \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} - 1 \begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix} = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} - \begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix} = \begin{bmatrix} -1 \\ 1 \\ 0 \end{bmatrix}$$ (This is updated $w(\vec{3})$)

4. Consider fourth data point $\begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$, find $\vec{w}^T \vec{x}$

$$[-1\ 1\ 0] \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} = \text{-1} < 0$$

Example : Perceptron Learning Algorithm

3. Consider third data point $\begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix}$, find $\vec{w}^T\vec{x}$

$$[-1\ 0\ 1] \begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix} = 1 > 0$$ (Miss-classification, result should be < 0 for $w2$ samples)

- update rule, w : $\vec{w} \leftarrow \vec{w} + y\vec{x}$

$$w(\vec{3}) \leftarrow \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} - 1 \begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix} = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} - \begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix} = \begin{bmatrix} -1 \\ 1 \\ 0 \end{bmatrix}$$ (This is updated $w(\vec{3})$)

4. Consider fourth data point $\begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$, find $\vec{w}^T\vec{x}$

$$[-1\ 1\ 0] \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} = \text{-1} < 0$$

(Correct as $\vec{w}^T\vec{x} < 0$ for $w2$ samples, no update in $w(\vec{3})$ required, $w(\vec{4}) = w(\vec{3})$)

Example : Perceptron Learning Algorithm

5. One loop on dataset is completed in which misclassification were encountered, now again go through dataset (loop will only stop if there is no misclassification). Consider $\begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$, find $\vec{w}^T\vec{x}$

Example : Perceptron Learning Algorithm

5. One loop on dataset is completed in which misclassification were encountered, now again go through dataset (loop will only stop if there is no misclassification). Consider $\begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$, find $\vec{w}^T\vec{x}$

$$\begin{bmatrix} -1 & 1 & 0 \end{bmatrix} \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} = 1 > 0 \quad (\text{Correct as } \vec{w}^T\vec{x} > 0 \text{ for } w1 \text{ samples}, \ w(\vec{5}) = w(\vec{4}))$$

6. Consider second data point $\begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}$, find $\vec{w}^T\vec{x}$

Example : Perceptron Learning Algorithm

5. One loop on dataset is completed in which misclassification were encountered, now again go through dataset (loop will only stop if there is no misclassification). Consider $\begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$, find $\vec{w}^T\vec{x}$

$$\begin{bmatrix} -1 & 1 & 0 \end{bmatrix} \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} = 1 > 0 \quad (\text{Correct as } \vec{w}^T\vec{x} > 0 \text{ for } w1 \text{ samples}, \; w(\vec{5}) = w(\vec{4}))$$

6. Consider second data point $\begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}$, find $\vec{w}^T\vec{x}$

$$\begin{bmatrix} -1 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} = 1 > 0 \; (\text{Correct as } \vec{w}^T\vec{x} > 0 \text{ for } w1 \text{ samples}, \; w(\vec{6}) = w(\vec{5}))$$

Example : Perceptron Learning Algorithm

7. Consider third data point $\begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix}$, find $\vec{w}^T\vec{x}$

Example : Perceptron Learning Algorithm

7. Consider third data point $\begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix}$, find $\vec{w}^T\vec{x}$

$$\begin{bmatrix} -1 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix} = \text{-1} < 0$$

(Correct as $\vec{w}^T\vec{x} < 0$ for $w2$ samples, $w(\vec{7}) = w(\vec{6})$)

Example : Perceptron Learning Algorithm

7. Consider third data point $\begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix}$, find $\vec{w}^T\vec{x}$

$$\begin{bmatrix} -1 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix} = \text{-}1 < 0$$

(Correct as $\vec{w}^T\vec{x} < 0$ for $w2$ samples, $w(\vec{7}) = w(\vec{6})$)

- Since for four consecutive steps no correction is needed, all points are correctly classified and the algorithm terminates. Final weight vector $w = \begin{bmatrix} -1 & 1 & 0 \end{bmatrix}^T$.

Example : Perceptron Learning Algorithm

7. Consider third data point $\begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix}$, find $\vec{w}^T\vec{x}$

$$\begin{bmatrix} -1 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix} = \text{-1} < 0$$

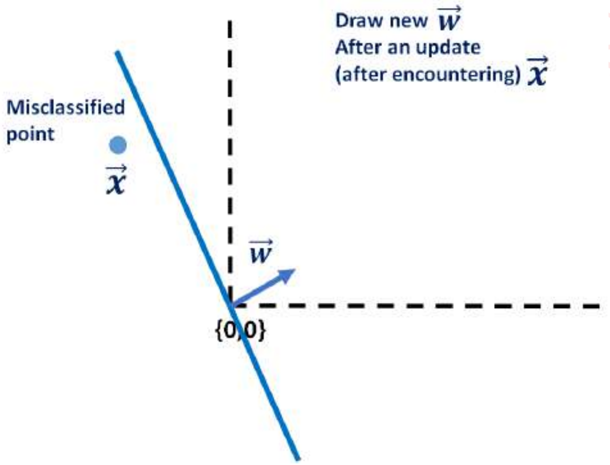
(Correct as $\vec{w}^T\vec{x} < 0$ for $w2$ samples, $\vec{w(7)} = \vec{w(6)}$)

- Since for four consecutive steps no correction is needed, all points are correctly classified and the algorithm terminates. Final weight vector $w = \begin{bmatrix} -1 & 1 & 0 \end{bmatrix}^T$.

- That is the resulting linear classifier that correctly separates all data points. This line has slope = 1 and intercept = 0, how?

Section Contents

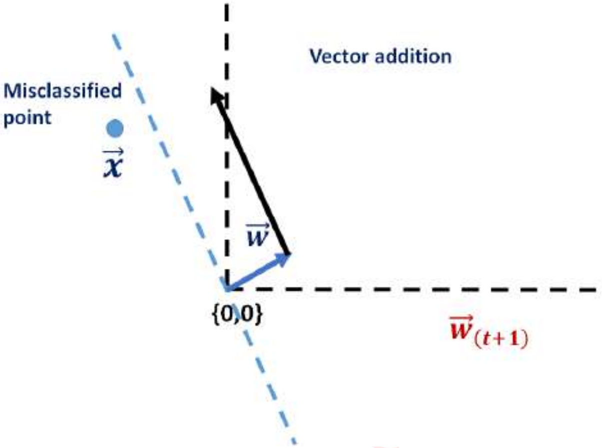
w update visualization



- Draw new $\vec{w}$ after encountering $\vec{x} \in w_+$, which is misclassified point.

- update rule, w : $\vec{w} \leftarrow \vec{w} + y\vec{x}$

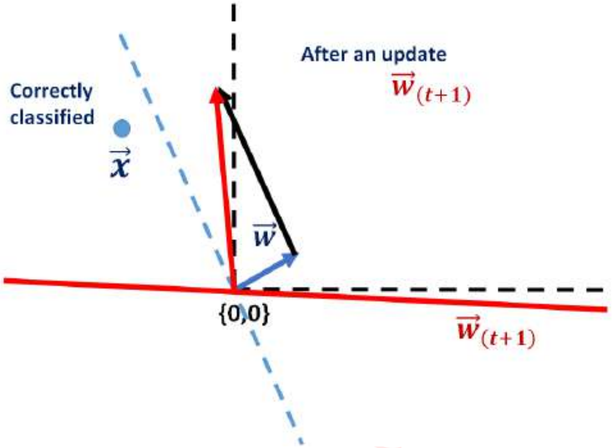
w update visualization



- Draw new $\vec{w}$ after encountering $\vec{x} \in w_+$, which is misclassified point.

- update rule, w : $\vec{w} \leftarrow \vec{w} + y\vec{x}$

w update visualization

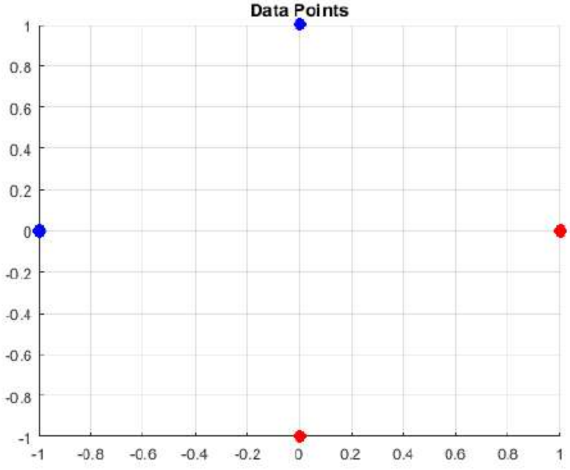


- Draw new $\vec{w}$ after encountering $\vec{x} \in w_+$, which is misclassified point.

- update rule, w : $\vec{w} \leftarrow \vec{w} + y\vec{x}$

- In our example after an update $\vec{x}$ gets correctly classified but there is no guarantee that after one update data point will be correctly classified.

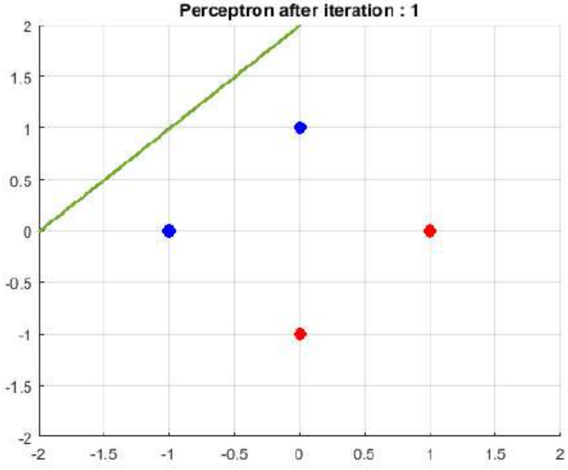
Demo 1: Perceptron Learning Algorithm



*3

[3] Matlab demo available

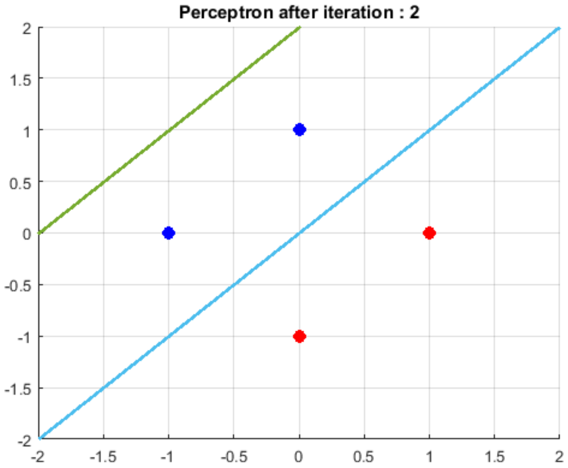
Demo 1: Perceptron Learning Algorithm



*3

[3] Matlab demo available

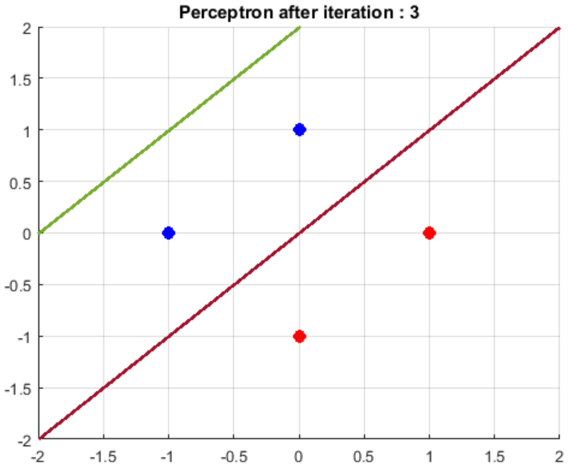
Demo 1: Perceptron Learning Algorithm



*3

[3] Matlab demo available

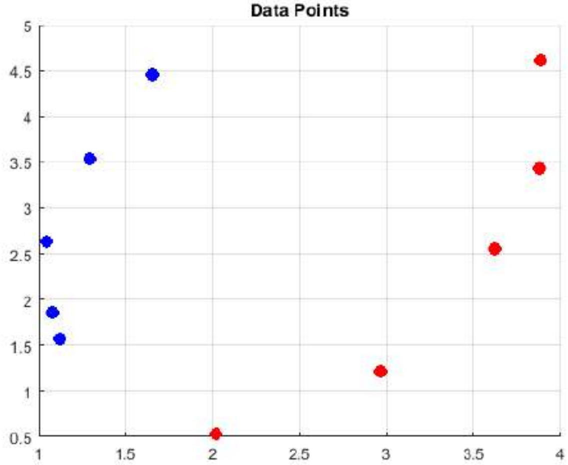
Demo 1: Perceptron Learning Algorithm



*3

[3] Matlab demo available

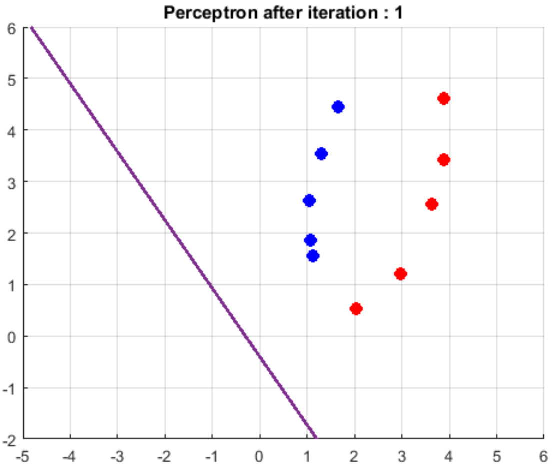
Demo 2: Perceptron Learning Algorithm



*4

[4]Matlab demo available

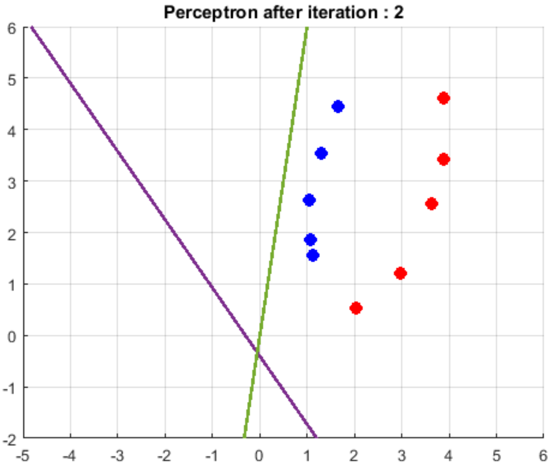
Demo 2: Perceptron Learning Algorithm



*4

[4] Matlab demo available

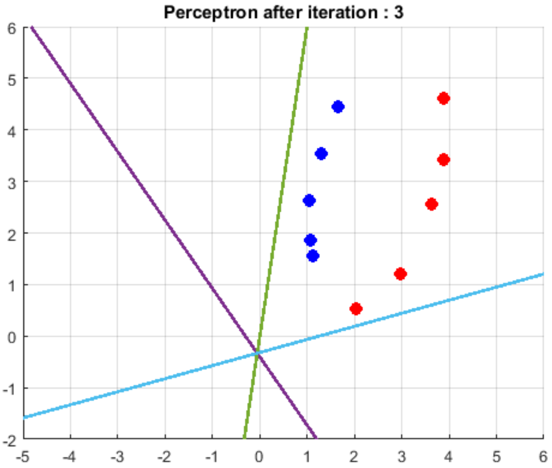
Demo 2: Perceptron Learning Algorithm



*4

[4]Matlab demo available

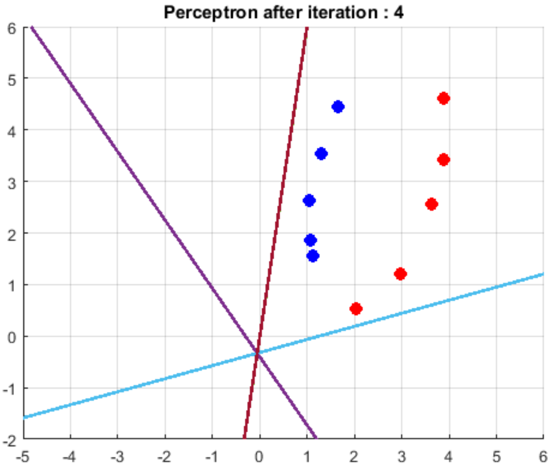
Demo 2: Perceptron Learning Algorithm



*4

[4]Matlab demo available

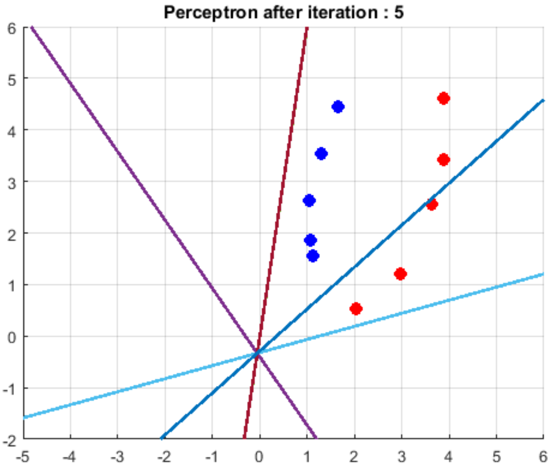
Demo 2: Perceptron Learning Algorithm



*4

[4]Matlab demo available

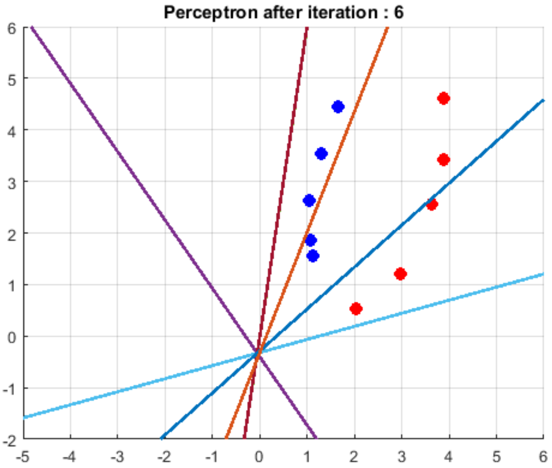
Demo 2: Perceptron Learning Algorithm



*4

[4] Matlab demo available

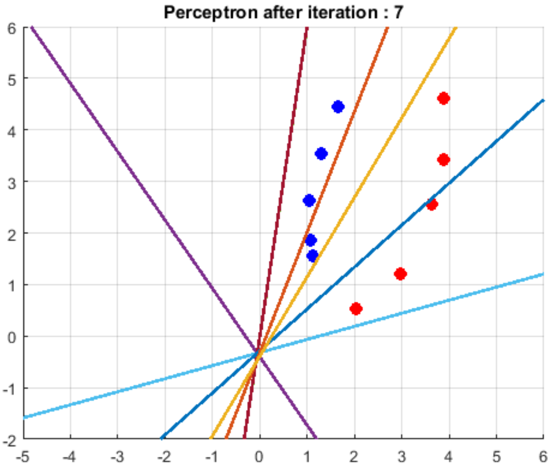
Demo 2: Perceptron Learning Algorithm



*4

[4] Matlab demo available

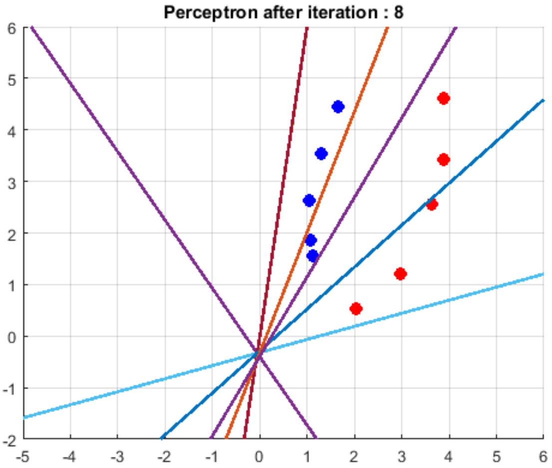
Demo 2: Perceptron Learning Algorithm



*4

[4] Matlab demo available

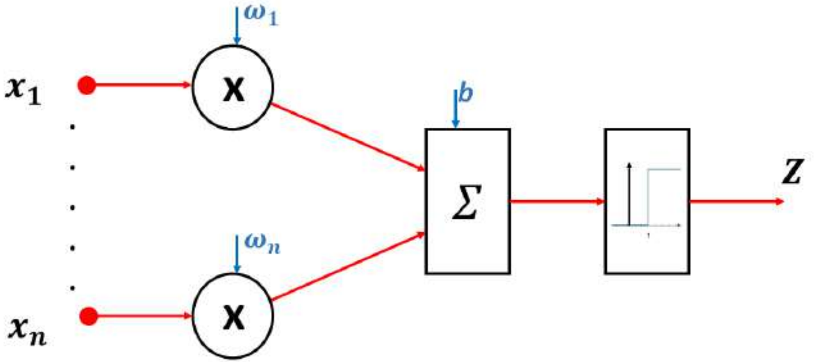
Demo 2: Perceptron Learning Algorithm



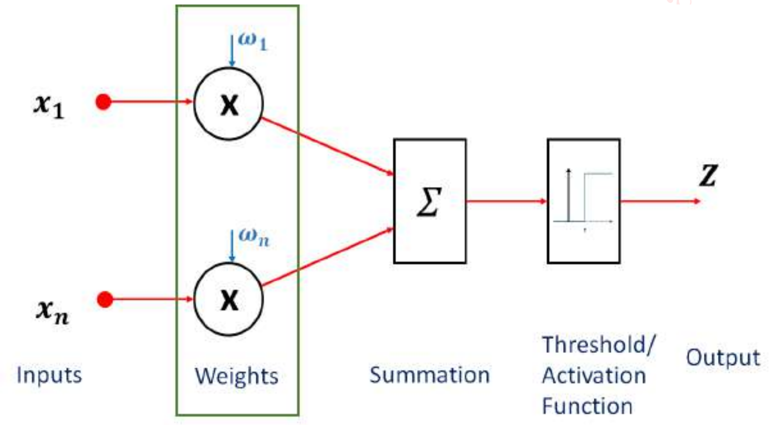
*[4]

[4]Matlab demo available

Perceptron or Artificial Neuron



Artificial Neuron

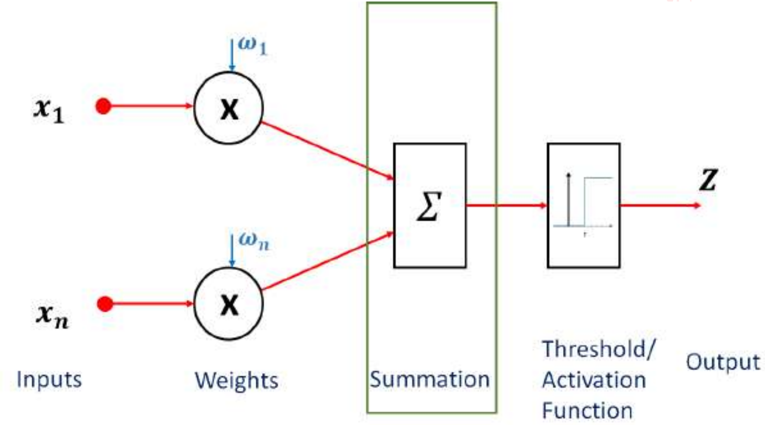


Step 1

Modeling synaptic connection.

$$x_i \times w_i$$

Artificial Neuron

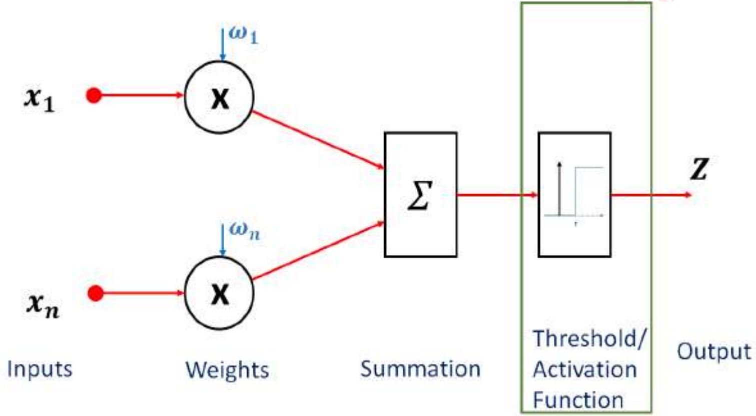


Step 2

Modeling collection of inputs

$$\sum_i x_i w_i$$

Artificial Neuron



Step 3

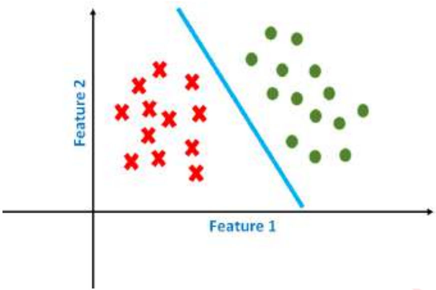
Decision, whether collective input is more than threshold to fire neuron

$$f(x) = \begin{cases} 1, if x \geq T \\ 0, otherwise \end{cases}$$

Section Contents

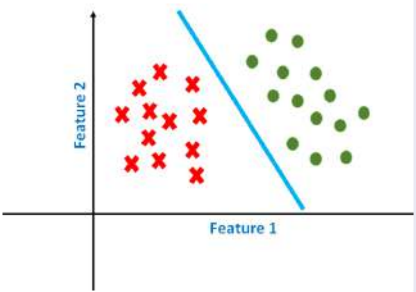
Perceptron Algorithm

- First algorithm with a strong formal guarantee of convergence.
  1. If the data is linearly separable, it will find a separating hyperplane in a finite number of updates.
  2. If the data is not linearly separable, it will loop forever.



Perceptron Algorithm

- First algorithm with a strong formal guarantee of convergence.
  1. If the data is linearly separable, it will find a separating hyperplane in a finite number of updates.
  2. If the data is not linearly separable, it will loop forever.



Perceptron Algorithm

- If $\exists \mathbf{w}$ such that $y_i(\mathbf{w}^\top \mathbf{x}) > 0 \ \forall (\mathbf{x}_i, y_i) \in D$, then Perceptron will find that $\mathbf{w}$ in finite number of steps.
  - **Condition to satisfy:**

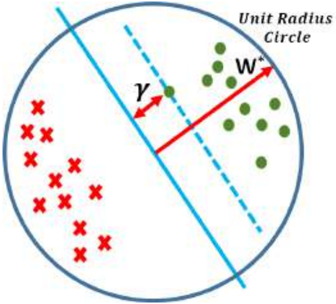
$$y(\mathbf{w}^\top \mathbf{x}) > 0 \begin{cases} y & = +1 : \ \mathbf{w}^\top \mathbf{x} > 0 \\ y & = -1 : \ \mathbf{w}^\top \mathbf{x} < 0 \end{cases} \quad (5)$$

  - **Update rule:**

$$\vec{w} \leftarrow \vec{w} + y\vec{x} \begin{cases} y & = +1 : \ \vec{w} \leftarrow \vec{w} + \vec{x} \\ y & = -1 : \ \vec{w} \leftarrow \vec{w} - \vec{x} \end{cases} \quad (6)$$

Perceptron Convergence : Setup

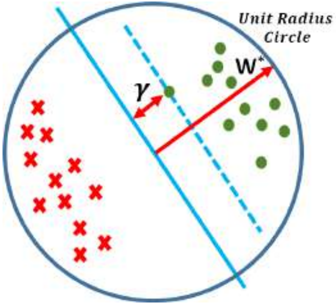
1. If $\exists \mathbf{w}^*$ such that $y_i(\mathbf{w}*^\top \mathbf{x}) > 0 \ \forall(\mathbf{x}_i, y_i) \in D$
2. Rescale each data point and the $\mathbf{w}^*$ such that:
   $||\mathbf{w}^*|| = 1$ and $||\mathbf{x}_i|| \leq 1 \ \ \forall \mathbf{x}_i \in D$
   - To get $||\mathbf{x}_i|| \leq 1$, divide all $\mathbf{x}$ by norm of max of $\mathbf{x}$.
3. Let us define the Margin (it's a constant) (the distance from the hyperplane to the closest data point) γ of the hyperplane $\mathbf{w}^*$ as $\gamma = \min_{(\mathbf{x}_i, y_i) \in D} |\mathbf{w}*^\top \mathbf{x}_i|$



Note:
$\mathbf{w}^*$ is one of the hyperplane that separates data and point no. 2 elaborates on how data is scaled to be confined in unit radius circle. This helps in proof of convergence.

Perceptron Convergence : Setup

1. If $\exists \mathbf{w}^*$ such that $y_i(\mathbf{w}*^\top \mathbf{x}) > 0 \ \forall(\mathbf{x}_i, y_i) \in D$
2. Rescale each data point and the $\mathbf{w}^*$ such that:
   $||\mathbf{w}^*|| = 1$ and $||\mathbf{x}_i|| \leq 1 \ \ \forall \mathbf{x}_i \in D$
   - To get $||\mathbf{x}_i|| \leq 1$, divide all $\mathbf{x}$ by norm of max of $\mathbf{x}$.
3. Let us define the Margin (it's a constant) (the distance from the hyperplane to the closest data point) γ of the hyperplane $\mathbf{w}^*$ as $\gamma = \min_{(\mathbf{x}_i, y_i) \in D} |\mathbf{w}*^\top \mathbf{x}_i|$



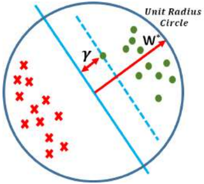
Note:
$\mathbf{w}^*$ is one of the hyperplane that separates data and point no. 2 elaborates on how data is scaled to be confined in unit radius circle. This helps in proof of convergence.

Theorem

If all of the above holds, then the Perceptron algorithm makes at most $1/\gamma^2$ mistakes before it converges.

Perceptron Convergence : Setup

- $\mathbf{w}$ is initial hyperplane that we have (let's say all zeros)
- $\mathbf{w}^*$ is a separating hyperplane that we want to obtain.
- Keeping previous definition, consider the effect of an update $(\mathbf{w} + y\mathbf{x})$ on the two terms:
  1. $\mathbf{w}^\top \mathbf{w}^*$
  2. $\mathbf{w}^\top \mathbf{w}$,



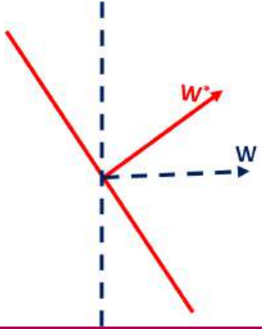
Why these two terms?

1. First Term $(\mathbf{w}^\top \mathbf{w}^*)$: Calculates how closer $\mathbf{w}$ is getting to $\mathbf{w}^*$, inner product.
2. Second term $(\mathbf{w}^\top \mathbf{w})$: This is required in order to understand that increase in first term is not due to scaling (first term can grow even if hyperplanes are not getting close but getting scaled i.e. scaled by 2) but these hyperplanes are actually getting closer i.e. $\mathbf{w}$ is tilting towards $\mathbf{w}^*$. So it is required that this term should not grow fast.

Perceptron Convergence : Two Terms

Why these two terms?

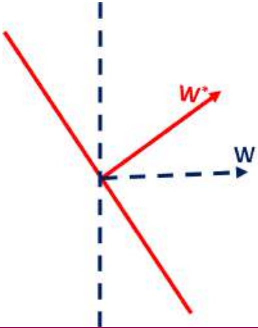
1. First Term ($\mathbf{w}^\top \mathbf{w}^*$): Calculates how closer $\mathbf{w}$ is getting to $\mathbf{w}^*$, inner product.
2. Second term ($\mathbf{w}^\top \mathbf{w}$): This is required in order to understand that increase in first term is not due to scaling (first term can grow even if hyperplanes are not getting close but getting scaled i.e. scaled by 2) but these hyperplanes are actually getting closer i.e. $\mathbf{w}$ is tilting towards $\mathbf{w}^*$. So it is required that this term should not grow fast.

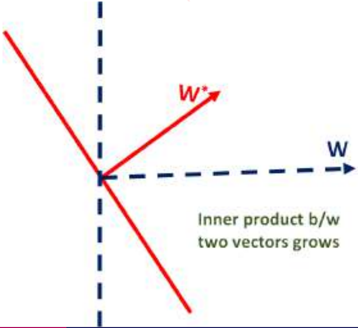


Perceptron Convergence : Two Terms

Why these two terms?

1. First Term ($\mathbf{w}^\top \mathbf{w}^*$): Calculates how closer $\mathbf{w}$ is getting to $\mathbf{w}^*$, inner product.
2. Second term ($\mathbf{w}^\top \mathbf{w}$): This is required in order to understand that increase in first term is not due to scaling (first term can grow even if hyperplanes are not getting close but getting scaled i.e. scaled by 2) but these hyperplanes are actually getting closer i.e. $\mathbf{w}$ is tilting towards $\mathbf{w}^*$. So it is required that this term should not grow fast.

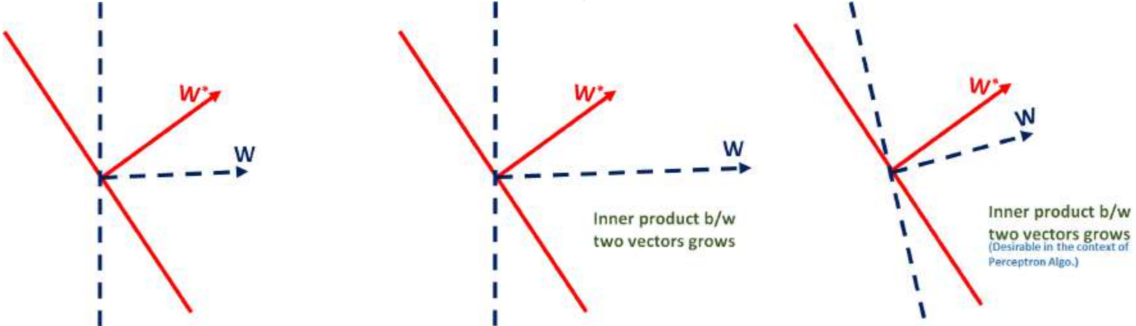




Perceptron Convergence : Two Terms

Why these two terms?

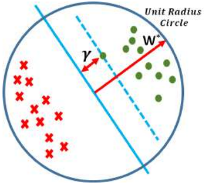
1. First Term ($\mathbf{w}^\top \mathbf{w}^*$): Calculates how closer $\mathbf{w}$ is getting to $\mathbf{w}^*$, inner product.
2. Second term ($\mathbf{w}^\top \mathbf{w}$): This is required in order to understand that increase in first term is not due to scaling (first term can grow even if hyperplanes are not getting close but getting scaled i.e. scaled by 2) but these hyperplanes are actually getting closer i.e. $\mathbf{w}$ is tilting towards $\mathbf{w}^*$. So it is required that this term should not grow fast.



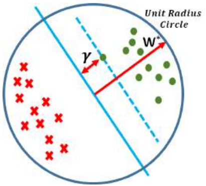
Perceptron Convergence : First Term

Two facts, in case $\mathbf{w}$ gets updated

1. $y(\mathbf{x}^\top \mathbf{w}) \leq 0$: This holds because $\mathbf{x}$ is misclassified by $\mathbf{w}$ - otherwise update wouldn't happen.
2. $y(\mathbf{x}^\top \mathbf{w}^*) > 0$: This holds because $\mathbf{w}^*$ is a separating hyper-plane and classifies all points correctly.



Perceptron Convergence : First Term



Two facts, in case w gets updated

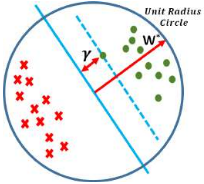
1. $y(\mathbf{x}^\top \mathbf{w}) \leq 0$: This holds because $\mathbf{x}$ is misclassified by $\mathbf{w}$ - otherwise update wouldn't happen.
2. $y(\mathbf{x}^\top \mathbf{w}^*) > 0$: This holds because $\mathbf{w}^*$ is a separating hyper-plane and classifies all points correctly.

- How this update ($\vec{w} \leftarrow \vec{w} + y\vec{x}$) **effects (first term)**, which is $\mathbf{w}^\top \mathbf{w}^*$:

Perceptron Convergence : First Term

Two facts, in case $\mathbf{w}$ gets updated

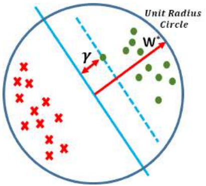
1. $y(\mathbf{x}^\top \mathbf{w}) \leq 0$: This holds because $\mathbf{x}$ is misclassified by $\mathbf{w}$ - otherwise update wouldn't happen.
2. $y(\mathbf{x}^\top \mathbf{w}^*) > 0$: This holds because $\mathbf{w}^*$ is a separating hyper-plane and classifies all points correctly.



- How this update ($\vec{w} \leftarrow \vec{w} + y\vec{x}$) **effects (first term)**, which is $\mathbf{w}^\top \mathbf{w}^*$:
  -

$$(\mathbf{w} + y\mathbf{x})^\top \mathbf{w}^* = \mathbf{w}^\top \mathbf{w}^* + \underbrace{y(\mathbf{x}^\top \mathbf{w}^*)}_{>0 \text{ or } \geq \gamma} \underbrace{\geq \mathbf{w}^\top \mathbf{w}^* + \gamma}_{Resultant} \tag{7}$$

Perceptron Convergence : First Term



Two facts, in case $\mathbf{w}$ gets updated

1. $y(\mathbf{x}^\top \mathbf{w}) \leq 0$: This holds because $\mathbf{x}$ is misclassified by $\mathbf{w}$ - otherwise update wouldn't happen.
2. $y(\mathbf{x}^\top \mathbf{w}^*) > 0$: This holds because $\mathbf{w}^*$ is a separating hyper-plane and classifies all points correctly.

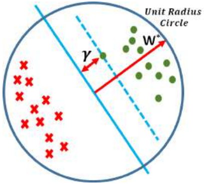
- How this update ($\vec{w} \leftarrow \vec{w} + y\vec{x}$) **effects (first term)**, which is $\mathbf{w}^\top \mathbf{w}^*$:
  - $$(\mathbf{w} + y\mathbf{x})^\top \mathbf{w}^* = \mathbf{w}^\top \mathbf{w}^* + \underbrace{y(\mathbf{x}^\top \mathbf{w}^*)}_{>0 \text{ or } \geq\gamma} \underbrace{\geq \mathbf{w}^\top \mathbf{w}^* + \gamma}_{Resultant} \quad (7)$$

    the distance from the hyperplane defined by $\mathbf{w}^*$ to $\mathbf{x}$ must be at least γ
    or
    $y(\mathbf{x}^\top \mathbf{w}^*) = |\mathbf{x}^\top \mathbf{w}^*| \geq \gamma$

Perceptron Convergence : First Term

Two facts, in case $\mathbf{w}$ gets updated

1. $y(\mathbf{x}^\top \mathbf{w}) \leq 0$: This holds because $\mathbf{x}$ is misclassified by $\mathbf{w}$ - otherwise update wouldn't happen.
2. $y(\mathbf{x}^\top \mathbf{w}^*) > 0$: This holds because $\mathbf{w}^*$ is a separating hyper-plane and classifies all points correctly.



- How this update ($\vec{w} \leftarrow \vec{w} + y\vec{x}$) **effects (first term)**, which is $\mathbf{w}^\top \mathbf{w}^*$:
  -

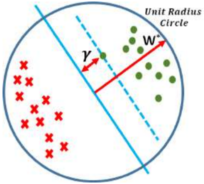
$$(\mathbf{w} + y\mathbf{x})^\top \mathbf{w}^* = \mathbf{w}^\top \mathbf{w}^* + \underbrace{y(\mathbf{x}^\top \mathbf{w}^*)}_{>0 \text{ or } \geq\gamma} \underbrace{\geq \mathbf{w}^\top \mathbf{w}^* + \gamma}_{Resultant} \quad (7)$$

the distance from the hyperplane defined by $\mathbf{w}^*$ to $\mathbf{x}$ must be at least γ
or
$y(\mathbf{x}^\top \mathbf{w}^*) = |\mathbf{x}^\top \mathbf{w}^*| \geq \gamma$

Conclusion-1

This means that for each update, $\mathbf{w}^\top \mathbf{w}^*$ grows by at least γ i.e. $\mathbf{w}^\top \mathbf{w}^* + \gamma$.

Perceptron Convergence : Second Term



Two facts, in case w gets updated

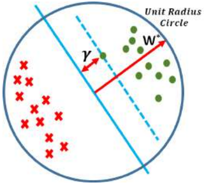
1. $y(\mathbf{x}^\top \mathbf{w}) \leq 0$: This holds because **x** is misclassified by **w** - otherwise update wouldn't happen.
2. $y(\mathbf{x}^\top \mathbf{w}^*) > 0$: This holds because $\mathbf{w}^*$ is a separating hyper-plane and classifies all points correctly.

- How this update ($\vec{w} \leftarrow \vec{w} + y\vec{x}$) **effects (second term)**, which is $\mathbf{w}^\top \mathbf{w}$:

Perceptron Convergence : Second Term

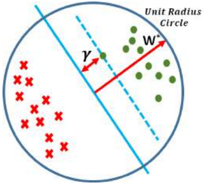
Two facts, in case $\mathbf{w}$ gets updated

1. $y(\mathbf{x}^\top \mathbf{w}) \leq 0$: This holds because $\mathbf{x}$ is misclassified by $\mathbf{w}$ - otherwise update wouldn't happen.
2. $y(\mathbf{x}^\top \mathbf{w}^*) > 0$: This holds because $\mathbf{w}^*$ is a separating hyper-plane and classifies all points correctly.



- How this update ($\vec{w} \leftarrow \vec{w} + y\vec{x}$) **effects (second term)**, which is $\mathbf{w}^\top \mathbf{w}$:
  - $$(\mathbf{w} + y\mathbf{x})^\top (\mathbf{w} + y\mathbf{x}) = \mathbf{w}^\top \mathbf{w} + \underbrace{2y(\mathbf{w}^\top \mathbf{x})}_{<0} + \underbrace{y^2}_{=1} \underbrace{(\mathbf{x}^\top \mathbf{x})}_{\leq 1} \underbrace{\leq \mathbf{w}^\top \mathbf{w} + 1}_{Resultant} \quad (8)$$

Perceptron Convergence : Second Term



Two facts, in case $\mathbf{w}$ gets updated

1. $y(\mathbf{x}^\top \mathbf{w}) \leq 0$: This holds because $\mathbf{x}$ is misclassified by $\mathbf{w}$ - otherwise update wouldn't happen.
2. $y(\mathbf{x}^\top \mathbf{w}^*) > 0$: This holds because $\mathbf{w}^*$ is a separating hyper-plane and classifies all points correctly.

- How this update ($\vec{w} \leftarrow \vec{w} + y\vec{x}$) **effects (second term)**, which is $\mathbf{w}^\top \mathbf{w}$:
  - $$(\mathbf{w} + y\mathbf{x})^\top (\mathbf{w} + y\mathbf{x}) = \mathbf{w}^\top \mathbf{w} + \underbrace{2y(\mathbf{w}^\top \mathbf{x})}_{<0} + \underbrace{y^2}_{=1} \underbrace{(\mathbf{x}^\top \mathbf{x})}_{\leq 1} \underbrace{\leq \mathbf{w}^\top \mathbf{w} + 1}_{Resultant} \tag{8}$$
  - The inequality follows from the fact that:
    - $2y(\mathbf{w}^\top \mathbf{x}) < 0$ as we had to make an update, meaning $\mathbf{x}$ was misclassified.
    - $0 \leq y^2(\mathbf{x}^\top \mathbf{x}) \leq 1$ as $y^2 = 1$ and $\mathbf{x}^\top \mathbf{x} \leq 1$ (because $\|\mathbf{x}\| \leq 1$, data was scaled to have max. norm of 1)

Conclusion-2

This means that for each update, $\mathbf{w}^\top \mathbf{w}$ grows by at most 1, i.e. $\mathbf{w}^\top \mathbf{w} + 1$.

Perceptron Convergence : Final Step

After M updates, the following two inequalities must hold:

1. $\mathbf{w}^\top \mathbf{w}^* \geq M\gamma$ as $\mathbf{w}^\top \mathbf{w}^*$ grows by at least γ, so after M updates it must be at least $M\gamma$
2. $\mathbf{w}^\top \mathbf{w} \leq M$ as $\mathbf{w}^\top \mathbf{w}$ grows by at most 1

$$M\gamma \leq \mathbf{w}^\top \mathbf{w}^* = \underbrace{|\mathbf{w}^\top \mathbf{w}^*|}_{Abs.Val.}$$

[5] Cauchy-Schwarz inequality: For two vectors, their inner products is less than equal to product of their norms

Perceptron Convergence : Final Step

After M updates, the following two inequalities must hold:

1. $\mathbf{w}^\top \mathbf{w}^* \geq M\gamma$ as $\mathbf{w}^\top \mathbf{w}^*$ grows by at least γ, so after M updates it must be at least $M\gamma$
2. $\mathbf{w}^\top \mathbf{w} \leq M$ as $\mathbf{w}^\top \mathbf{w}$ grows by at most 1

$$M\gamma \leq \mathbf{w}^\top \mathbf{w}^* = \underbrace{|\mathbf{w}^\top \mathbf{w}^*|}_{Abs.Val.}$$

$$\leq \underbrace{||\mathbf{w}^\top||.||\mathbf{w}^*||}_{\text{Cauchy-Schwarz inequality}} = ||\mathbf{w}^\top|| \quad \text{as } ||\mathbf{w}^*|| = 1 \text{ (Data scaled)}$$

[5] Cauchy-Schwarz inequality: For two vectors, their inner products is less than equal to product of their norms

Perceptron Convergence : Final Step

After M updates, the following two inequalities must hold:

1. $\mathbf{w}^\top \mathbf{w}^* \geq M\gamma$ as $\mathbf{w}^\top \mathbf{w}^*$ grows by at least γ, so after M updates it must be at least $M\gamma$
2. $\mathbf{w}^\top \mathbf{w} \leq M$ as $\mathbf{w}^\top \mathbf{w}$ grows by at most 1

$$M\gamma \leq \mathbf{w}^\top \mathbf{w}^* = \underbrace{|\mathbf{w}^\top \mathbf{w}^*|}_{Abs.Val.}$$

$$\leq \underbrace{||\mathbf{w}^\top||.||\mathbf{w}^*||}_{\text{Cauchy-Schwarz inequality}} = ||\mathbf{w}^\top|| \text{ as } ||\mathbf{w}^*|| = 1 \text{ (Data scaled)}$$

$$= \underbrace{\sqrt{\mathbf{w}^\top \mathbf{w}}}_{\text{Definition of norm}}$$

[5] Cauchy-Schwarz inequality: For two vectors, their inner products is less than equal to product of their norms

Perceptron Convergence : Final Step

After M updates, the following two inequalities must hold:

1. $\mathbf{w}^\top \mathbf{w}^* \geq M\gamma$ as $\mathbf{w}^\top \mathbf{w}^*$ grows by at least γ, so after M updates it must be at least $M\gamma$
2. $\mathbf{w}^\top \mathbf{w} \leq M$ as $\mathbf{w}^\top \mathbf{w}$ grows by at most 1

$$M\gamma \leq \mathbf{w}^\top \mathbf{w}^* = \underbrace{|\mathbf{w}^\top \mathbf{w}^*|}_{Abs.Val.}$$

$$\leq \underbrace{||\mathbf{w}^\top||.||\mathbf{w}^*||}_{\text{Cauchy-Schwarz inequality}} = ||\mathbf{w}^\top|| \text{ as } ||\mathbf{w}^*|| = 1 \text{ (Data scaled)}$$

$$= \underbrace{\sqrt{\mathbf{w}^\top \mathbf{w}}}_{\text{Definition of norm}}$$

- What do we know about $\mathbf{w}^\top \mathbf{w}$?

[5] Cauchy-Schwarz inequality: For two vectors, their inner products is less than equal to product of their norms

Perceptron Convergence : Final Step

After M updates, the following two inequalities must hold:

1. $\mathbf{w}^\top \mathbf{w}^* \geq M\gamma$ as $\mathbf{w}^\top \mathbf{w}^*$ grows by at least γ, so after M updates it must be at least $M\gamma$
2. $\mathbf{w}^\top \mathbf{w} \leq M$ as $\mathbf{w}^\top \mathbf{w}$ grows by at most 1

$$M\gamma \leq \mathbf{w}^\top \mathbf{w}^* = \underbrace{|\mathbf{w}^\top \mathbf{w}^*|}_{Abs.Val.}$$

$$\leq \underbrace{||\mathbf{w}^\top||.||\mathbf{w}^*||}_{\text{Cauchy-Schwarz inequality}} = ||\mathbf{w}^\top|| \quad \text{as } ||\mathbf{w}^*|| = 1 \text{ (Data scaled)}$$

$$= \underbrace{\sqrt{\mathbf{w}^\top \mathbf{w}}}_{\text{Definition of norm}}$$

- What do we know about $\mathbf{w}^\top \mathbf{w}$?
- $\mathbf{w}^\top \mathbf{w}$ grows by at most 1 Conc-2.

[5] Cauchy-Schwarz inequality: For two vectors, their inner products is less than equal to product of their norms

Perceptron Convergence : Final Step

After M updates, the following two inequalities must hold:

1. $\mathbf{w}^\top\mathbf{w}^* \geq M\gamma$ as $\mathbf{w}^\top\mathbf{w}^*$ grows by at least γ, so after M updates it must be at least $M\gamma$
2. $\mathbf{w}^\top\mathbf{w} \leq M$ as $\mathbf{w}^\top\mathbf{w}$ grows by at most 1

$$M\gamma \leq \mathbf{w}^\top\mathbf{w}^* = \underbrace{|\mathbf{w}^\top\mathbf{w}^*|}_{Abs.Val.}$$

$$\leq \underbrace{||\mathbf{w}^\top||.||\mathbf{w}^*||}_{\text{Cauchy-Schwarz inequality}} = ||\mathbf{w}^\top|| \text{ as } ||\mathbf{w}^*|| = 1 \text{ (Data scaled)}$$

$$= \underbrace{\sqrt{\mathbf{w}^\top\mathbf{w}}}_{\text{Definition of norm}}$$

- What do we know about $\mathbf{w}^\top\mathbf{w}$?
- $\mathbf{w}^\top\mathbf{w}$ grows by at most 1 Conc-2.
- So, after **M** updates:

$= \sqrt{\mathbf{w}^\top\mathbf{w}} \leq \sqrt{M}$

[5] Cauchy-Schwarz inequality: For two vectors, their inner products is less than equal to product of their norms

Perceptron Convergence : Final Step

After M updates, the following two inequalities must hold:

1. $\mathbf{w}^\top\mathbf{w}^* \geq M\gamma$ as $\mathbf{w}^\top\mathbf{w}^*$ grows by at least γ, so after M updates it must be at least $M\gamma$
2. $\mathbf{w}^\top\mathbf{w} \leq M$ as $\mathbf{w}^\top\mathbf{w}$ grows by at most 1

$$M\gamma \leq \mathbf{w}^\top\mathbf{w}^* = \underbrace{|\mathbf{w}^\top\mathbf{w}^*|}_{Abs.Val.}$$

$$\leq \underbrace{||\mathbf{w}^\top||.||\mathbf{w}^*||}_{\text{Cauchy-Schwarz inequality}} = ||\mathbf{w}^\top|| \text{ as } ||\mathbf{w}^*|| = 1 \text{ (Data scaled)}$$

$$= \underbrace{\sqrt{\mathbf{w}^\top\mathbf{w}}}_{\text{Definition of norm}}$$

[5]

- What do we know about $\mathbf{w}^\top\mathbf{w}$?
- $\mathbf{w}^\top\mathbf{w}$ grows by at most 1 Conc-2.
- So, after **M** updates:

$= \sqrt{\mathbf{w}^\top\mathbf{w}} \leq \sqrt{M}$

Interesting find

$M\gamma \leq \sqrt{M}$

[5] Cauchy-Schwarz inequality: For two vectors, their inner products is less than equal to product of their norms

Perceptron Convergence : Final Step

We proved

$$M\gamma \leq \sqrt{M}$$

Perceptron Convergence : Final Step

We proved

$M\gamma \leq \sqrt{M}$

- Solve for **M**:

Perceptron Convergence : Final Step

We proved

$M\gamma \leq \sqrt{M}$

- Solve for **M**:

$$M\gamma \leq \sqrt{M} \tag{9}$$

$$M^2\gamma^2 \leq M \tag{10}$$

$$M \leq \frac{1}{\gamma^2} \tag{11}$$

- This proof made Frank Rosenblatt famous. Such a strong result!

Perceptron Convergence : Final Step

We proved

$M\gamma \leq \sqrt{M}$

- Solve for **M**:

$$M\gamma \leq \sqrt{M} \tag{9}$$

$$M^2\gamma^2 \leq M \tag{10}$$

$$M \leq \frac{1}{\gamma^2} \tag{11}$$

- This proof made Frank Rosenblatt famous. Such a strong result!

Perceptron Algorithm Convergence

$M \leq \frac{1}{\gamma^2}$: This means number of updates **M** is bounded from above by a constant. So algorithm wouldn't make more mistakes than constant $\frac{1}{\gamma^2}$ (smallest distance between data point $\mathbf{x}$ and $\mathbf{w}^*$) before finding a linear separating hyperplane.

Section Contents

Interesting Facts

Frank Rosenblatt
1928–1969

Rosenblatt's perceptron played an important role in the history of machine learning. Initially, Rosenblatt simulated the perceptron on an IBM 704 computer at Cornell in 1957, but by the early 1960s he had built special-purpose hardware that provided a direct, parallel implementation of perceptron learning. Many of his ideas were encapsulated in "Principles of Neuro-dynamics: Perceptrons and the Theory of Brain Mechanisms" published in 1962. Rosenblatt's work was criticized by Marvin Minksy, whose objections were published in the book "Perceptrons", co-authored with Seymour Papert. This book was widely misinterpreted at the time as showing that neural networks were fatally flawed and could only learn solutions for linearly separable problems. In fact, it only proved such limitations in the case of single-layer networks such as the perceptron and merely conjectured (incorrectly) that they applied to more general network models. Unfortunately, however, this book contributed to the substantial decline in research funding for neural computing, a situation that was not reversed until the mid-1980s. Today, there are many hundreds, if not thousands, of applications of neural networks in widespread use, with examples in areas such as handwriting recognition and information retrieval being used routinely by millions of people.

*[6]

[6] Image from Pattern Recognition and Machine Learning Book by Christopher Bishop

Interesting Facts

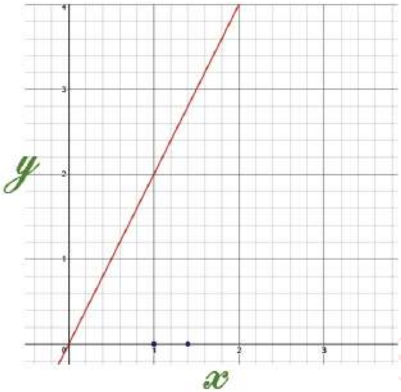
Figure 4.8 Illustration of the Mark 1 perceptron hardware. The photograph on the left shows how the inputs were obtained using a simple camera system in which an input scene, in this case a printed character, was illuminated by powerful lights, and an image focussed onto a 20×20 array of cadmium sulphide photocells, giving a primitive 400 pixel image. The perceptron also had a patch board, shown in the middle photograph, which allowed different configurations of input features to be tried. Often these were wired up at random to demonstrate the ability of the perceptron to learn without the need for precise wiring, in contrast to a modern digital computer. The photograph on the right shows one of the racks of adaptive weights. Each weight was implemented using a rotary variable resistor, also called a potentiometer, driven by an electric motor thereby allowing the value of the weight to be adjusted automatically by the learning algorithm.

*[7]

[7] Image from Pattern Recognition and Machine Learning Book by Christopher Bishop

Section Contents

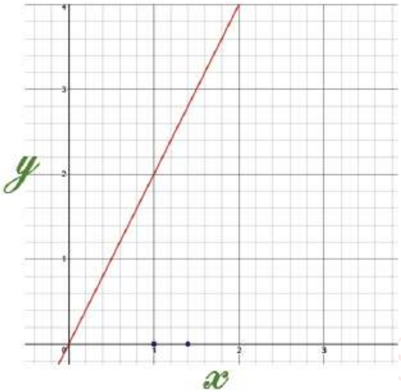
Equation of a line



Equation of a line:

$$y = mx + c$$

Equation of a line

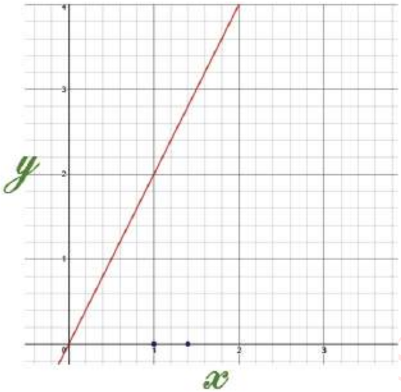


Equation of a line:

$$y = mx + c$$

$$m = \frac{rise}{run}$$

Equation of a line



Equation of a line:

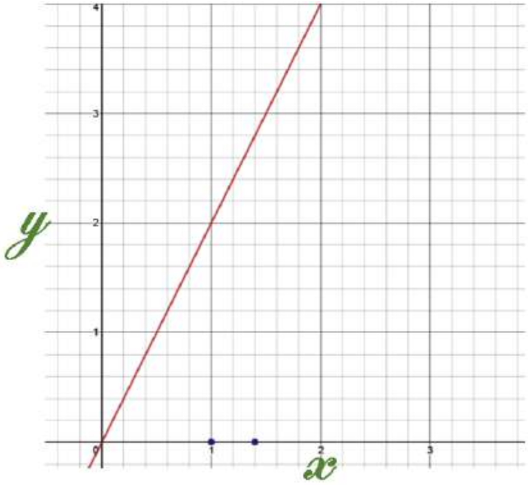
$$y = mx + c$$

$m = \frac{rise}{run}$

- m = slope
- c = y-intercept

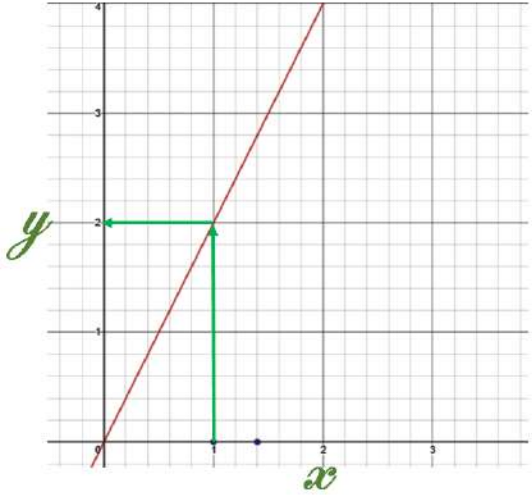
Equation of a line: Slope

Derivative / Slope Recap



Equation of a line: Slope

Derivative / Slope Recap



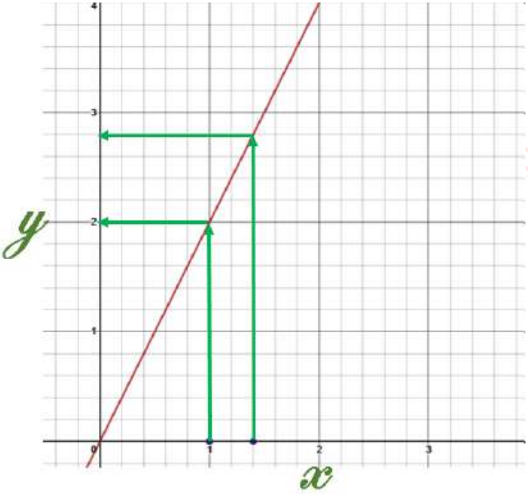
- Consider

$$f(x) = 2(x) \ or \ y = 2x$$

- if $x = 1$ then $f(x) = 2$

Equation of a line: Slope

Derivative / Slope Recap



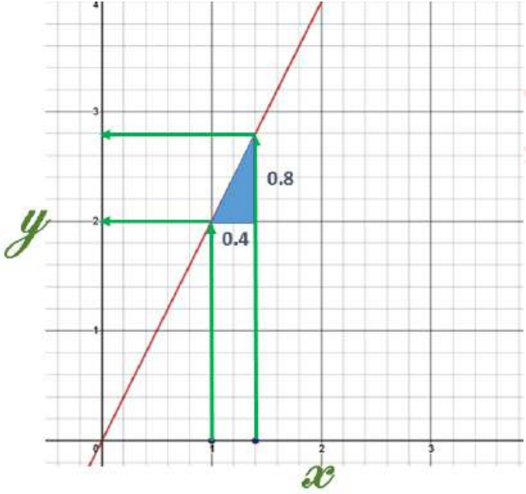
- Consider

$$f(x) = 2(x) \ \ or \ \ y = 2x$$

- if $x = 1$ then $f(x) = 2$
- if $x = 1.4$ then $f(x) = 2.8$

Equation of a line: Slope

Derivative / Slope Recap



- Consider

$$f(x) = 2(x) \;\; or \;\; y = 2x$$

- if $x = 1$ then $f(x) = 2$
- if $x = 1.4$ then $f(x) = 2.8$
- Slope $(\frac{dy}{dx})$ of $f(x)$ is 2.
  $\frac{dy}{dx} = \frac{height}{width}$
  $\frac{0.8}{0.4} = 2$

Equation of a line: General Form (2D)

$$ax + by + c = 0 \tag{12}$$

Equation of a line: General Form (2D)

$$ax + by + c = 0 \tag{12}$$

This equation (ref Equation 12) is same as slope form of a line $y = mx + c$

Equation of a line: General Form (2D)

$$ax + by + c = 0 \tag{12}$$

This equation (ref Equation 12) is same as slope form of a line $y = mx + c$

$$ax + by + c = 0 \tag{13}$$

$$y = \underbrace{-\frac{c}{b}}_{c \text{ or, } y-intercept} - \underbrace{\frac{a}{b}}_{m \text{ or, } slope} x \tag{14}$$

Equation of a line: General Form (2D)

$$ax + by + c = 0 \quad (12)$$

This equation (ref Equation 12) is same as slope form of a line $y = mx + c$

$$ax + by + c = 0 \quad (13)$$

$$y = \underbrace{-\frac{c}{b}}_{c\ or,\ y-intercept} - \underbrace{\frac{a}{b}}_{m\ or,\ slope} x \quad (14)$$

- If axis are x_1 and x_2, then $ax + by + c = 0$ can be written as:

$$ax_1 + bx_2 + c = 0 \quad (15)$$

Equation of a line: General Form (2D)

- Get rid of a and b as well, since we may need to write equation in n dimensions and then in this case we will run out of alphabets. Thus, Equation 15 can be written as:

$$w_1x_1 + w_2x_2 + w_0 = 0 \tag{16}$$

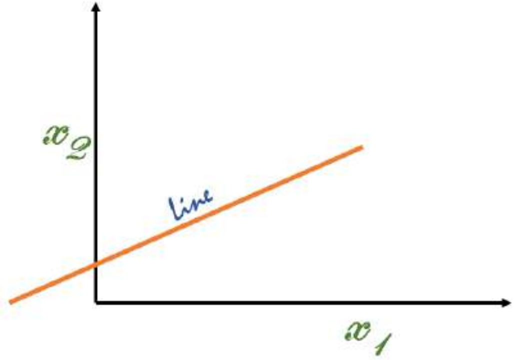
- What about in $3D$?

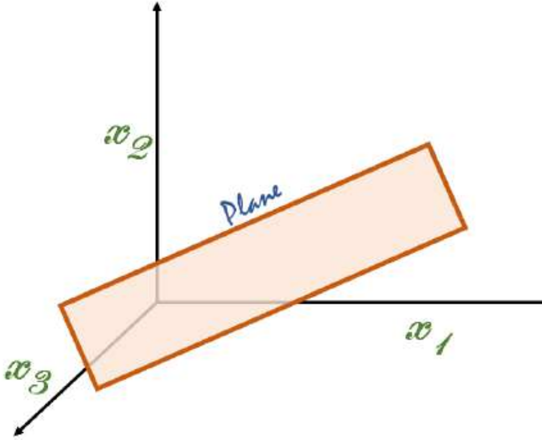
Plane in $3D$

- Equivalent of a line in $2D$ is a plane in $3D$.
- Idea is same. Line separates data in $2D$ surface, while plane separates data in $3D$ volume.

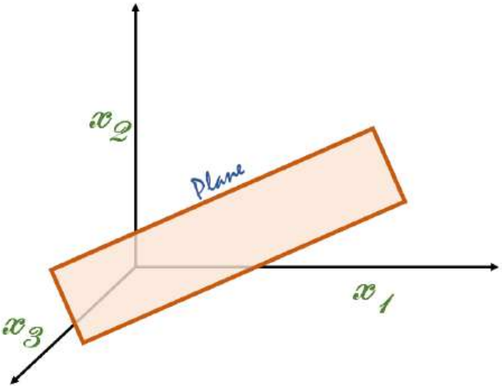
Plane in $3D$

- Equivalent of a line in $2D$ is a plane in $3D$.
- Idea is same. Line separates data in $2D$ surface, while plane separates data in $3D$ volume.





Plane in $3D$



- Extending Equation 16 to write equation of a plane in $3D$:

$$w_1x_1 + w_2x_2 + w_3x_3 + w_0 = 0 \tag{17}$$

- What about plane in nD?

Plane in nD

- Plane in n dimensions is called hyperplane.
- Equation of a plane nD can be formulated easily from Equations 16 and 17.

$$w_0 + w_1x_1 + w_2x_2 + w_3x_3 + \cdots + w_nx_n = 0 \tag{18}$$

Plane in nD

- Plane in n dimensions is called hyperplane.
- Equation of a plane nD can be formulated easily from Equations 16 and 17.

$$w_0 + w_1x_1 + w_2x_2 + w_3x_3 + \cdots + w_nx_n = 0 \tag{18}$$

- Is there a more concise way to write this equation?

Plane in nD

- Plane in n dimensions is called hyperplane.
- Equation of a plane nD can be formulated easily from Equations 16 and 17.

$$w_0 + w_1x_1 + w_2x_2 + w_3x_3 + \cdots + w_nx_n = 0 \tag{18}$$

- Is there a more concise way to write this equation?

$$w_0 + \sum_{i=1}^{n} w_ix_i = 0 \tag{19}$$

- Above form is summation form / notation of an equation. Is there a vector form to write this equation?

Vector notation of a plane in nD

- Vector notation of a plane in nD

$$w_0 + \underbrace{[w_1, w_2, w_3, \cdots, w_n]}_{w\ vector} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}}_{x\ vector} = 0 \tag{20}$$

- This equation, Equation 20 is exactly same as Equation 19.

Vector notation of a plane in nD

- Vector notation of a plane in nD

$$w_0 + \underbrace{[w_1, w_2, w_3, \cdots, w_n]}_{w\ vector} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}}_{x\ vector} = 0 \tag{20}$$

- This equation, Equation 20 is exactly same as Equation 19.
- Vector w has dimensions of $1 \times n$ ($w_{1\times n}$)

Vector notation of a plane in nD

- Vector notation of a plane in nD

$$w_0 + \underbrace{[w_1, w_2, w_3, \cdots, w_n]}_{w\ vector} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}}_{x\ vector} = 0 \tag{20}$$

- This equation, Equation 20 is exactly same as Equation 19.
- Vector w has dimensions of $1 \times n$ ($w_{1\times n}$)

- Vector x has dimensions of $n \times 1$ ($x_{n\times 1}$)

Vector notation of a plane in nD

- Vector notation of a plane in nD

$$w_0 + \underbrace{[w_1, w_2, w_3, \cdots, w_n]}_{w\ vector} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}}_{x\ vector} = 0 \tag{20}$$

- This equation, Equation 20 is exactly same as Equation 19.
- Vector w has dimensions of $1 \times n$ ($w_{1\times n}$)

- Vector x has dimensions of $n \times 1$ ($x_{n\times 1}$)

- Multiplication of vector w & vector x will give scalar or 1×1 matrix (multiplication of a row vector with a column vector).

Vector notation of a plane in nD

- In ML literature, as a standard, vector are written as column vector i.e. $\begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ \vdots \\ w_n \end{bmatrix}$

Taking Equation 20, and using standard notation, we can write:

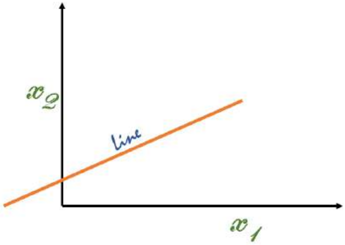
$$w_0 + \bar{w}^\top \bar{x} = 0 \tag{21}$$

- This is standard form of hyperplane equation!

Hyperplane equation with reference to line equation

- Equation of plane in 2D :

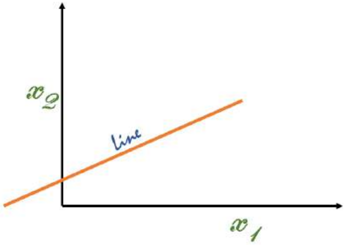
$$w_1x_1 + w_2x_2 + w_0 = 0$$



Hyperplane equation with reference to line equation

- Equation of plane in 2D :

$$w_1x_1 + w_2x_2 + w_0 = 0$$



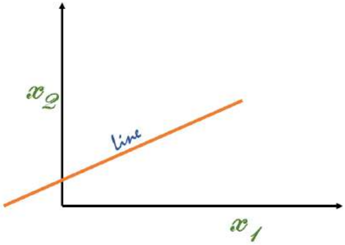
- Rearrange:

$$x_2 = -\frac{w_0}{w_2} - \frac{w_1}{w_2}x_1$$

Hyperplane equation with reference to line equation

- Equation of plane in 2D :

$$w_1x_1 + w_2x_2 + w_0 = 0$$



- Rearrange:

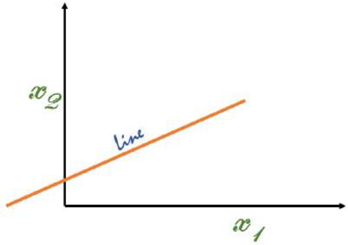
$$x_2 = -\frac{w_0}{w_2} - \frac{w_1}{w_2}x_1$$

- Can you find correspondence of this equation with $y = mx + c$

Hyperplane equation with reference to line equation

- Equation of plane in 2D :

$$w_1x_1 + w_2x_2 + w_0 = 0$$



- Rearrange:

$$x_2 = -\frac{w_0}{w_2} - \frac{w_1}{w_2}x_1$$

- Can you find correspondence of this equation with $y = mx + c$

$$\underbrace{x_2}_{y} = \underbrace{-\frac{w_0}{w_2}}_{c} - \underbrace{\frac{w_1}{w_2}}_{m} x_1 \tag{22}$$

Hyperplane passing through origin

As we have seen:

$$x_2 = -\frac{w_0}{w_2} - \frac{w_1}{w_2}x_1$$

- If this line passes through origin then $c = 0$ or $w_0 = 0$. Then Equation 16 will become:

$$w_1x_1 + w_2x_2 = 0 \tag{23}$$

- In 3D (Plane)

Hyperplane passing through origin

As we have seen:

$$x_2 = -\frac{w_0}{w_2} - \frac{w_1}{w_2}x_1$$

- If this line passes through origin then $c = 0$ or $w_0 = 0$. Then Equation 16 will become:

$$w_1x_1 + w_2x_2 = 0 \tag{23}$$

- In 3D (Plane)

$$w_1x_1 + w_2x_2 + w_3x_3 = 0 \tag{24}$$

- In nD (Hyperplane)

Hyperplane passing through origin

As we have seen:

$$x_2 = -\frac{w_0}{w_2} - \frac{w_1}{w_2} x_1$$

- If this line passes through origin then $c = 0$ or $w_0 = 0$. Then Equation 16 will become:

$$w_1 x_1 + w_2 x_2 = 0 \tag{23}$$

- In 3D (Plane)

$$w_1 x_1 + w_2 x_2 + w_3 x_3 = 0 \tag{24}$$

- In nD (Hyperplane)

$$w_1 x_1 + w_2 x_2 + w_3 x_3 + \cdots + w_n x_n = 0 \tag{25}$$

Hyperplane passing through origin

- Vector form of equation of hyperplane passing through origin:

$$\bar{w}^\top \bar{x} = 0 \tag{26}$$

- Vector form of equation of hyperplane not passing through origin:

$$w_0 + \bar{w}^\top \bar{x} = 0 \tag{27}$$

Geometric interpretation of Hyperplane

- Consider hyperplane that passes through origin, so Equation would be $\bar{w}^\top \bar{x} = 0$, where

$$w = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ \vdots \\ w_n \end{bmatrix} \; and \; x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}$$

$$w \cdot x = w^\top x = ||w|| \; ||x|| \; cos\theta_{w,x} \tag{28}$$

Geometric interpretation of Hyperplane

- Consider hyperplane that passes through origin, so Equation would be $\bar{w}^\top \bar{x} = 0$, where

$$w = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ \vdots \\ w_n \end{bmatrix} \; and \; x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}$$

$$w \cdot x = w^\top x = ||w|| \, ||x|| \, cos\theta_{w,x} \tag{28}$$

- According to definition of hyperplane passing through origin $\bar{w}^\top \bar{x} = 0$. This will only be true if vector w and x are orthogonal i.e ($cos(90) = 0$).

Geometric interpretation of Hyperplane

$$||w||\ ||x||\ cos\theta_{w,x} = 0$$

- As w and x vectors are orthogonal

- Usually vector w is taken as vector perpendicular ($\perp$) to the hyperplane as well, for all data points / vector x lie on the plane.

Geometric interpretation of Hyperplane

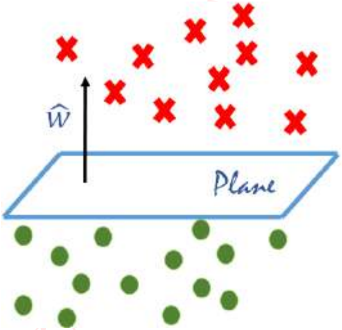
$$||w||\ ||x||\ cos\theta_{w,x} = 0$$

- As w and x vectors are orthogonal

- Usually vector w is taken as vector perpendicular ($\perp$) to the hyperplane as well, for all data points / vector x lie on the plane.
- Often hyperplane is defined by a unit vector $\hat{w} = \frac{w}{||w||}$ (e.g. $w \perp hyperplane$).

Geometric interpretation of Hyperplane

- Often hyperplane is defined by a unit vector $\hat{w} = \frac{w}{||w||}$ (e.g. $w \perp hyperplane$).



Machine Learning
Instance-Based Learning & Nearest Neighbor Classifier

Dr. Rizwan Ahmed Khan

Outline

Reference Books

Reference books for this Module:

- Chapter 8: Machine Learning, Tom MITCHELL, McGraw Hill, latest edition.

Reference Books

Reference books for this Module:

- Chapter 8: Machine Learning, Tom MITCHELL, McGraw Hill, latest edition.
- Chapter 2 & 5: Pattern Recognition, Konstantinos Koutroumbas and Sergios Theodoridi, Academic Press, 4th or latest edition

Section Contents

Abstraction: 1-D

If we live in one dimensional world:

Abstraction: 1-D

If we live in one dimensional world:

Abstraction: 1-D

If we live in one dimensional world:

Abstraction: 1-D

If we live in one dimensional world:

Abstraction: 1-D

If we live in one dimensional world:

Abstraction: 1-D

If we live in one dimensional world:

What would you say?

Abstraction: 1-D

- Previous slide presented points with associated labels i.e. 1 and 6.

Abstraction: 1-D

- Previous slide presented points with associated labels i.e. 1 and 6.
- When we are presented with point with unknown label i.e. test point, based on training points we were quick to answer.

Abstraction: 1-D

- Previous slide presented points with associated labels i.e. 1 and 6.
- When we are presented with point with unknown label i.e. test point, based on training points we were quick to answer.

- We were quick to understand underlying pattern in the data.

Abstraction: 1-D

- Previous slide presented points with associated labels i.e. 1 and 6.
- When we are presented with point with unknown label i.e. test point, based on training points we were quick to answer.

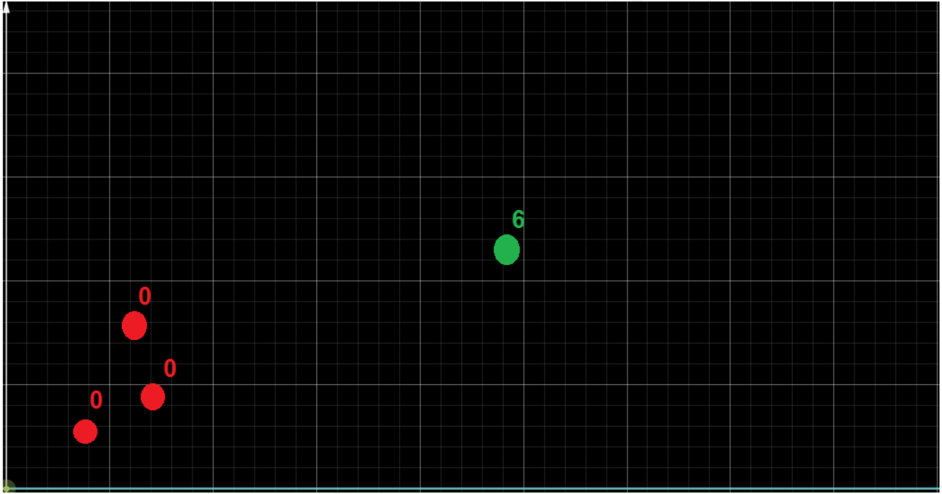
- We were quick to understand underlying pattern in the data.
- Without much of information we figured out that points are grouping together i.e. minimum distance

Abstraction: 2-D

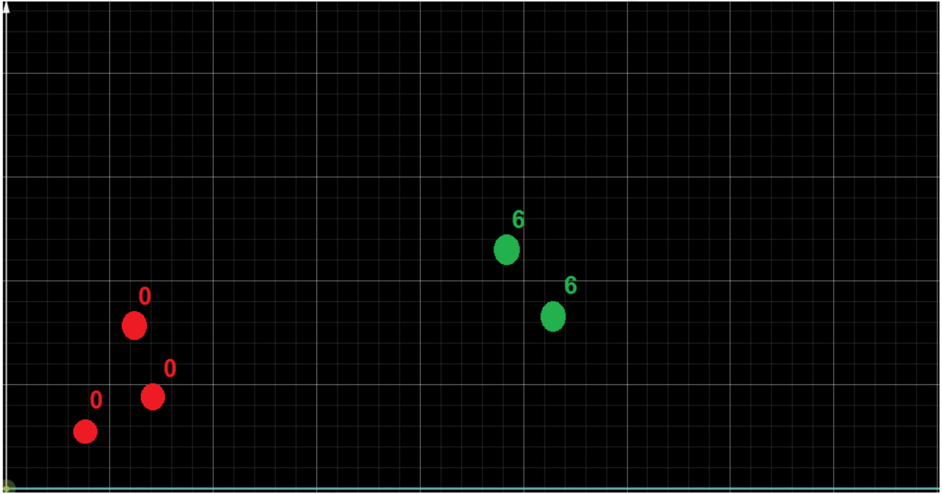
Abstraction: 2-D

Abstraction: 2-D

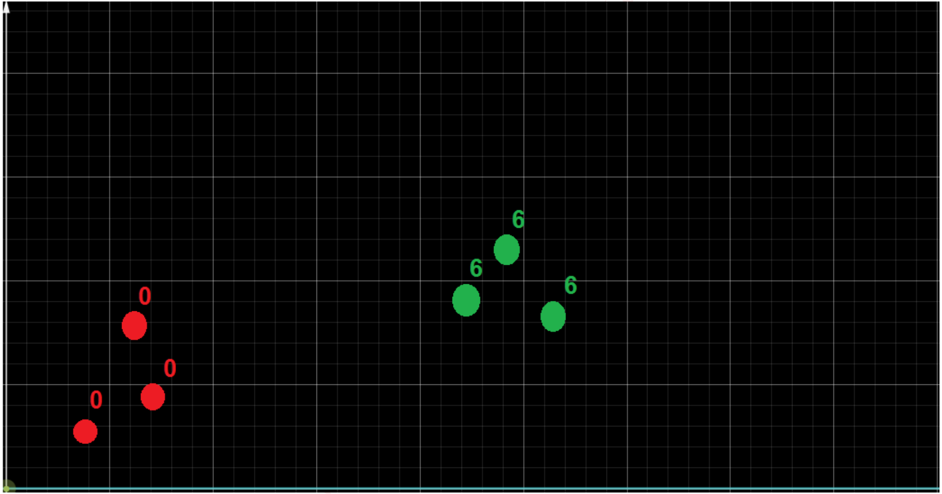
Abstraction: 2-D



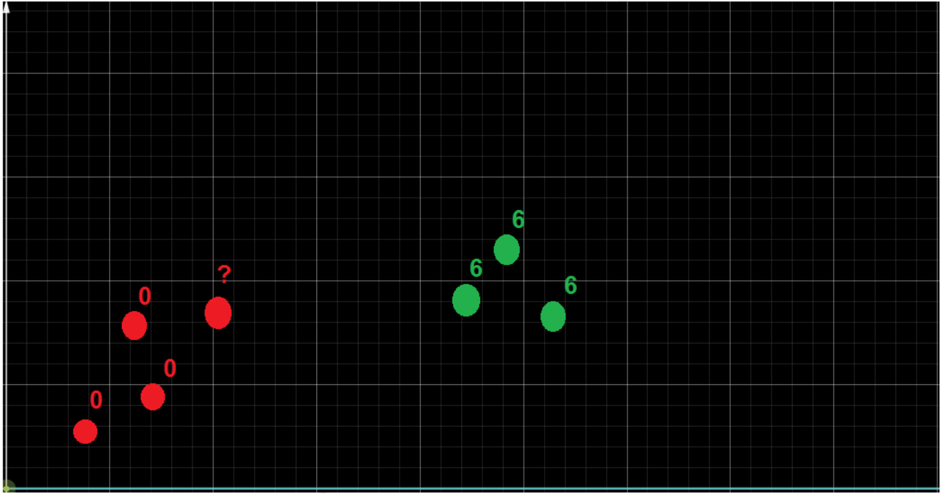
Abstraction: 2-D



Abstraction: 2-D



Abstraction: 2-D



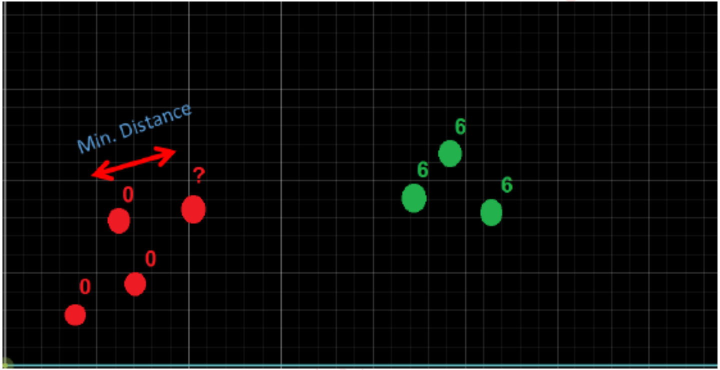
What would you say?

Abstraction: 2-D

- Again, in 2-D points with associated labels i.e. 1 and 6 were presented. When we are presented with point with unknown label i.e. test point, based on training points we were quick to answer.

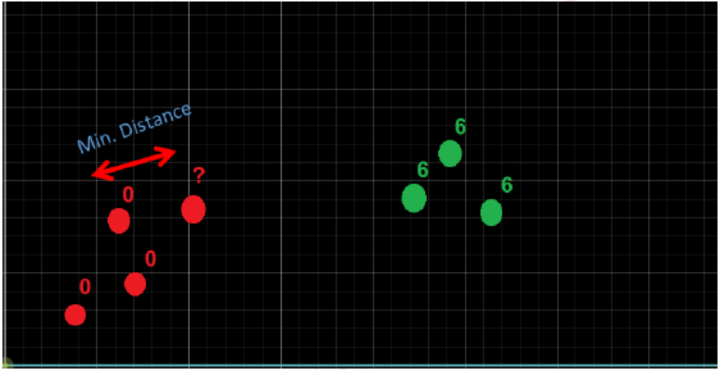
Abstraction: 2-D

- Again, in 2-D points with associated labels i.e. 1 and 6 were presented. When we are presented with point with unknown label i.e. test point, based on training points we were quick to answer.



Abstraction: 2-D

- Again, in 2-D points with associated labels i.e. 1 and 6 were presented. When we are presented with point with unknown label i.e. test point, based on training points we were quick to answer.



- Without much of information we figured out that points are grouping together (understand underlying pattern) i.e. minimum distance.

Abstraction: 2-D

- Again, in 2-D points with associated labels i.e. 1 and 6 were presented. When we are presented with point with unknown label i.e. test point, based on training points we were quick to answer.
- Without much of information we figured out that points are grouping together (understand underlying pattern) i.e. minimum distance.

Section Contents

K-NN Algorithm [1]

- Assumption: Similar Inputs have similar outputs.
- Classification rule: For a test input x, assign the most common label amongst its k most similar training inputs.
- Formal definition of k-NN:
  - Test point : x
  - Denote the set of the k nearest neighbors of x as S_x.
  - Formally S_x is defined as $S_{\mathbf{x}} \subseteq D$ (dataset) s.t. $|S_{\mathbf{x}}| = k$ and $\forall(\mathbf{x}', y') \in D \backslash S_{\mathbf{x}}$

$$\text{dist}(\mathbf{x}, \mathbf{x}') \geq \max_{(\mathbf{x}'', y'') \in S_{\mathbf{x}}} \text{dist}(\mathbf{x}, \mathbf{x}'') \quad (1)$$

That is every point in D but not in S_x is at least as far away from x as the farthest point in S_x.

[1] Cover, Thomas and Hart, Peter. Nearest neighbor pattern classification. Information Theory, IEEE Transactions on, 1967, 13(1): 21-27

Distance Metrics

Distance metric learning is a research field, but most commonly used are Minkowski Distance. Distance metric uses distance function which provides a relationship metric between elements in the dataset.

Minkowski Distance:

$$dist(a, b) = \left(\sum_{i=1}^{n} (a_i - b_i)^p \right)^{\frac{1}{p}} \tag{2}$$

Distance Metrics

Distance metric learning is a research field, but most commonly used are Minkowski Distance. Distance metric uses distance function which provides a relationship metric between elements in the dataset.

Minkowski Distance:

$$dist(a,b) = \left(\sum_{i=1}^{n} (a_i - b_i)^p \right)^{\frac{1}{p}} \tag{2}$$

1. if p = 1, Manhattan Distance

$$dist_{L1}(a,b) = \sum_{i=1}^{n} (\|a_i - b_i\|) \tag{3}$$

Distance Metrics

Distance metric learning is a research field, but most commonly used are Minkowski Distance. Distance metric uses distance function which provides a relationship metric between elements in the dataset.

Minkowski Distance:

$$dist(a,b) = \left(\sum_{i=1}^{n}(a_i - b_i)^p\right)^{\frac{1}{p}} \tag{2}$$

1. if p = 1, Manhattan Distance

$$dist_{L1}(a,b) = \sum_{i=1}^{n}(\|a_i - b_i\|) \tag{3}$$

2. if p = 2, Euclidean Distance

$$dist_{L2}(a,b) = \sqrt{\sum_{i=1}^{n}(a_i - b_i)^2} \tag{4}$$

Distance Metrics

Distance metric learning is a research field, but most commonly used are Minkowski Distance. Distance metric uses distance function which provides a relationship metric between elements in the dataset.

Minkowski Distance:

$$dist(a,b) = \left(\sum_{i=1}^{n} (a_i - b_i)^p \right)^{\frac{1}{p}} \quad (2)$$

1. if p = 1, Manhattan Distance

$$dist_{L1}(a,b) = \sum_{i=1}^{n} (\|a_i - b_i\|) \quad (3)$$

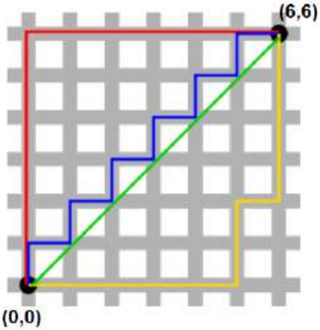
2. if p = 2, Euclidean Distance

$$dist_{L2}(a,b) = \sqrt{\sum_{i=1}^{n} (a_i - b_i)^2} \quad (4)$$

3. if p = ∞, Chebychev Distance / Max difference

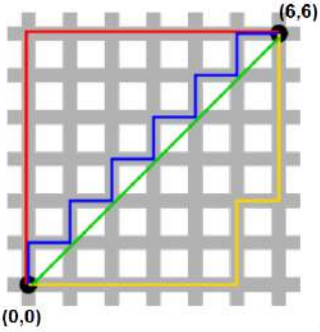
Manhattan or Euclidean Distance

Intuition of distances



Manhattan or Euclidean Distance

Intuition of distances

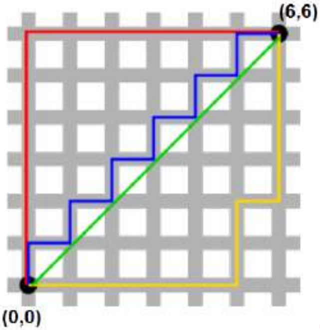


$$dist_{L1}(a,b) = \sum_{i=1}^{n}(\|a_i - b_i\|)$$

$$dist_{L1}(a,b) = (6-0) + (6-0) = 12 \quad (5)$$

Manhattan or Euclidean Distance

Intuition of distances



$$dist_{L1}(a,b) = \sum_{i=1}^{n}(\|a_i - b_i\|)$$

$$dist_{L1}(a,b) = (6-0) + (6-0) = 12 \quad (5)$$

$$dist_{L2}(a,b) = \sqrt{\sum_{i=1}^{n}(a_i - b_i)^2}$$

$$dist_{L2}(a,b) = \sqrt{6^2 + 6^2} = \sqrt{72} \approx 8.49 \quad (6)$$

In Manhattan / taxicab geometry, the red, yellow, and blue paths all have the same shortest path length of 12. In Euclidean geometry, the green line has length $6\sqrt{2} \approx 8.49$ and is the unique shortest path.

Toy Problem statement

$K - NN$: Toy dataset

We are given a training dataset with $n = 6$ observations of $d = 2$ dimensions.

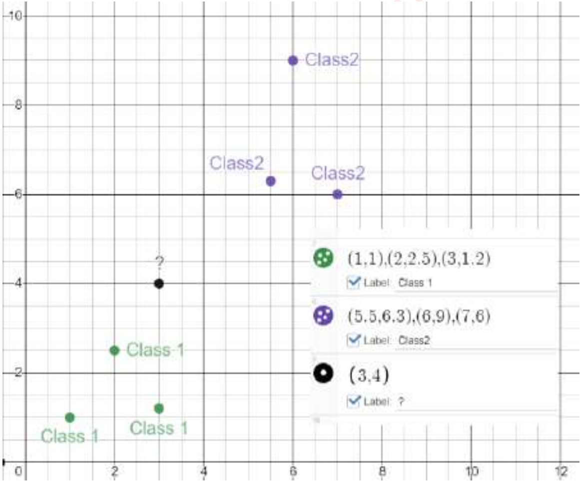
Table 1: Toy dataset

x_1	x_2	Label
1	1	class 1
2	2.5	class 1
3	1.2	class 1
5.5	6.3	class 2
6	9	class 2
7	6	class 2

Predict output class / label for query data point $x_q = [3, 4]^T$ for $K = 1$.

Toy Problem

Visualization of toy problem[2]

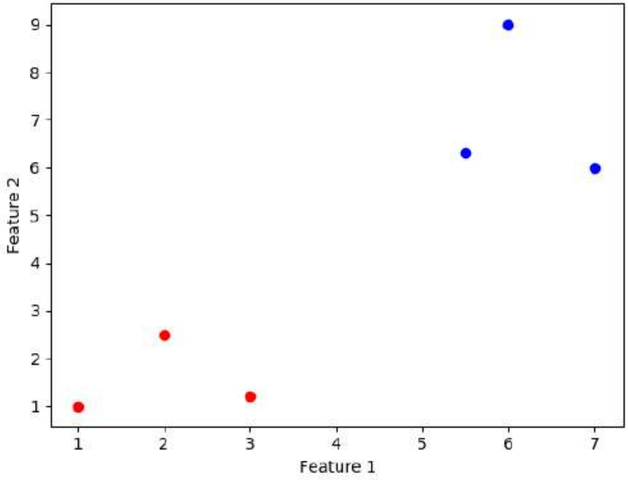


[1] https://www.desmos.com/

Toy Problem: Python

```python
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

#Create Training Set, 2D vector
x_train=np.array([[1,1], [2,2.5], [3,1.2], [5.5, 6.3], [6,9], [7,6]])
y_train=(1,1,1,2,2,2)


# create color dictionary for printing
colors = {1:'r', 2:'b'}

fig, ax = plt.subplots()
# plot each data-point
for i in range(len(x_train)):
    ax.scatter(x_train[i,0], x_train[i,1],color=colors[y_train[i]])

ax.set_xlabel('Feature 1')
ax.set_ylabel('Feature 2')
```

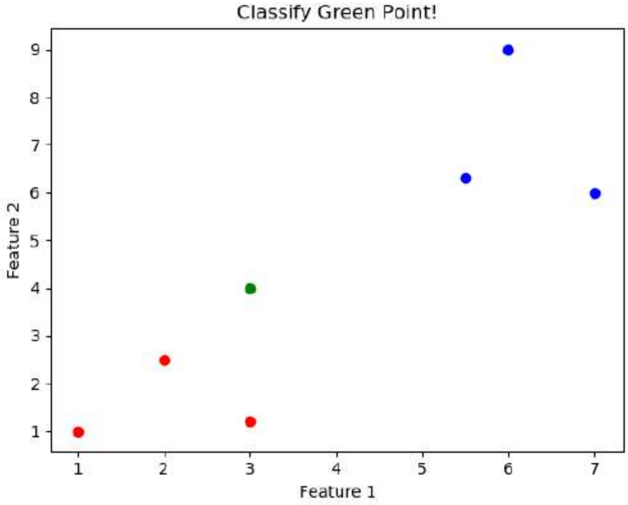
Toy Problem: Python Visualization



Toy Problem: Python

```

ax.set_xlabel('Feature 1')
ax.set_ylabel('Feature 2')


# Create Test point
x_test=np.array([3,4])
y_test=(1)


#plot again train + test data
#plt.figure()
fig, ax1 = plt.subplots()
for i in range(len(x_train)):
    ax1.scatter(x_train[i,0], x_train[i,1],color=colors[y_train[i]])
ax1.scatter(x_test[0],x_test[1],color='g')
ax1.set_xlabel('Feature 1')
ax1.set_ylabel('Feature 2')
ax1.set_title('Classify Green Point!')
```

Toy Problem: Python Visualization



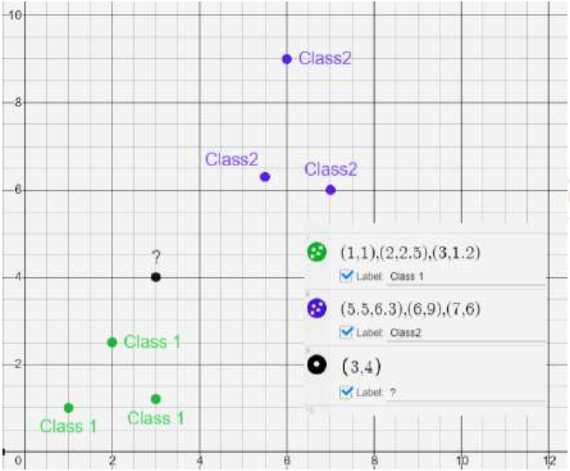
Toy Problem: Python

```
"""
Intuition : It seems new point (GREEN) is nearer to red points but How
mathematically we can prove that new point  is near to red point?
Step 1: Find Distance to all points in training
Step 2: Find point with minimum distance in training set
Step 3: Assign label of nearest point to test point

STEP 1
"""
def dist(x, y):
    return np.sqrt(np.sum((x-y)**2))

distance=np.zeros(len(x_train))
for i in range(len(x_train)):
    distance[i]=dist(x_train[i],x_test)
print(distance)

#Step 2: Find point with minimum distance in training set
min_index = np.argmin(distance)
#Step 3: Assign label of nearest point to test point
print('New point is classified in Class : ',y_train[min_index])
```

Toy Problem: Python Visualization


```
In [21]: print(distance)
    ...: print('New point is classified in Class : ',y_train[min_index])
    ...:
[3.60555128 1.80277564 2.8        3.39705755 5.83095189 4.47213595]
New point is classified in Class :  1
```

Verify Result: Euclidean Distance



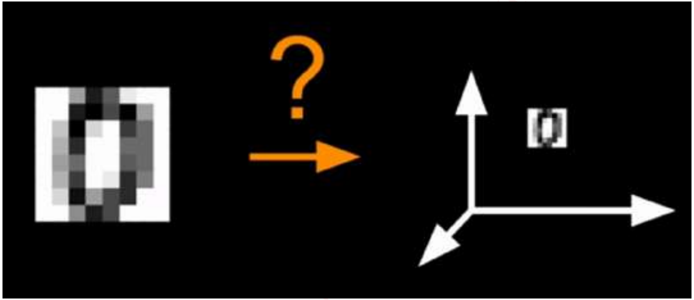

```
In [21]: print(distance)
    ...: print('New point is classified in Class : ',y_train[min_index])
    ...:
[3.60555128 1.80277564 2.8        3.39705755 5.83095189 4.47213595]
New point is classified in Class :  1
```

Move forward

- This is intuitive, easy to understand.
- Now the question is, can we map an image, audio, document to a point in feature space? As we have already seen the method to classify unknown point on feature space i.e. **Nearest Neighbor Classifier**.

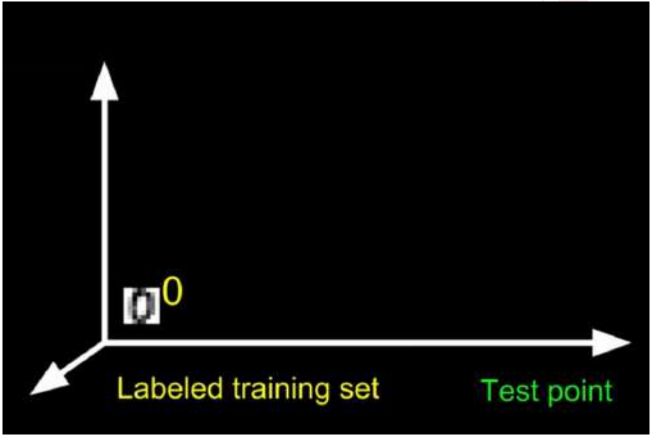
Move forward

- This is intuitive, easy to understand.
- Now the question is, can we map an image, audio, document to a point in feature space? As we have already seen the method to classify unknown point on feature space i.e. **Nearest Neighbor Classifier**.



Move forward

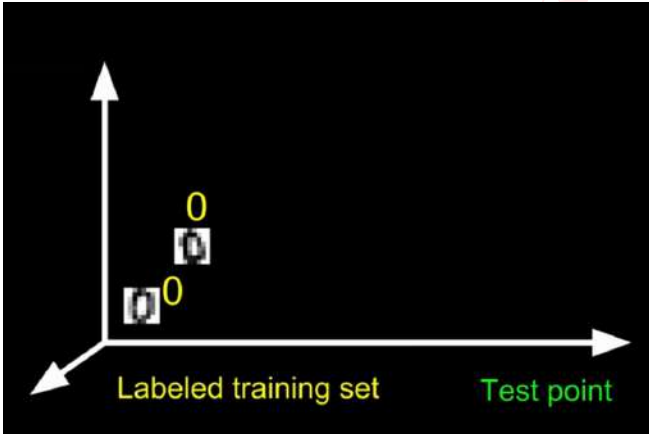
- If we can represent image in a space[3]like we did with toy example than its easy to classify it.



[2]Example images from CS50 - Harvard University.

Move forward

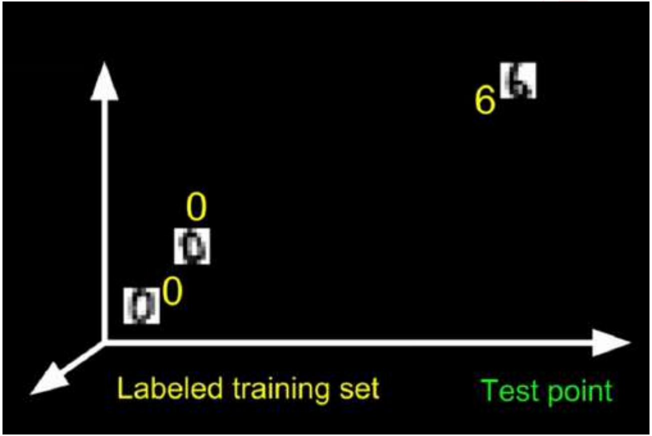
- If we can represent image in a space[3] like we did with toy example than its easy to classify it.



[2] Example images from CS50 - Harvard University.

Move forward

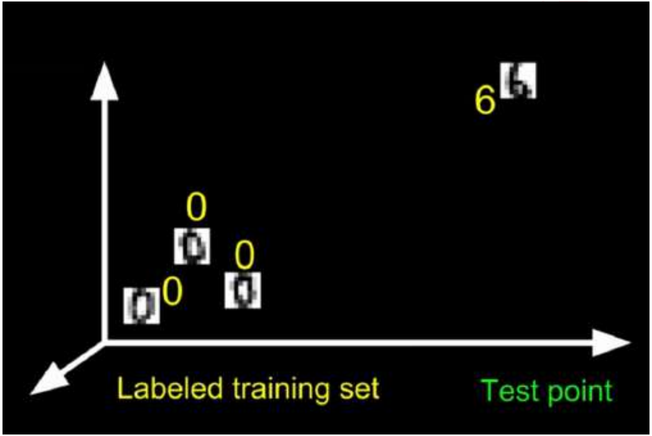
- If we can represent image in a space[3] like we did with toy example than its easy to classify it.



[2] Example images from CS50 - Harvard University.

Move forward

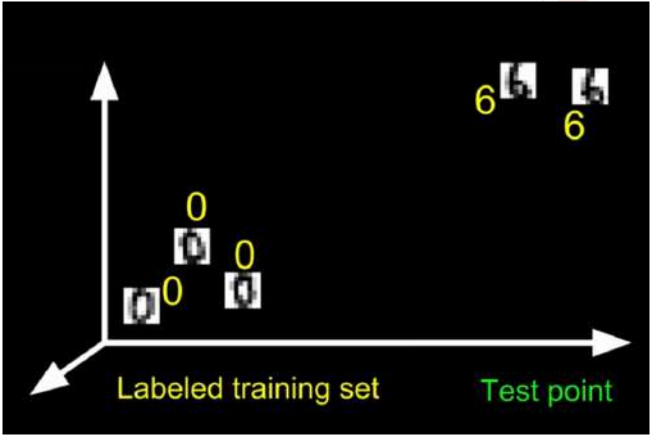
- If we can represent image in a space[3] like we did with toy example than its easy to classify it.



[2]Example images from CS50 - Harvard University.

Move forward

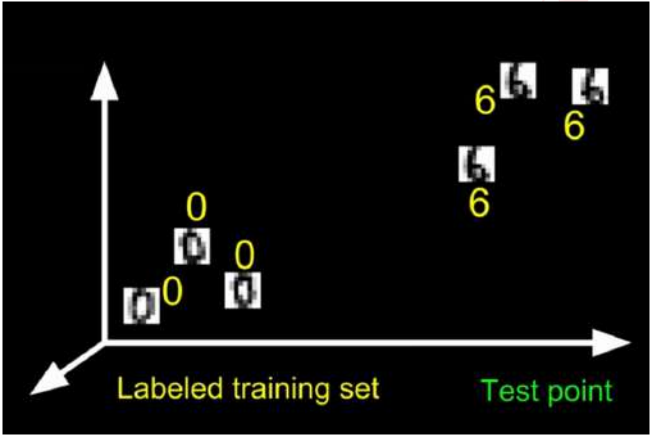
- If we can represent image in a space[3] like we did with toy example than its easy to classify it.



[2] Example images from CS50 - Harvard University.

Move forward

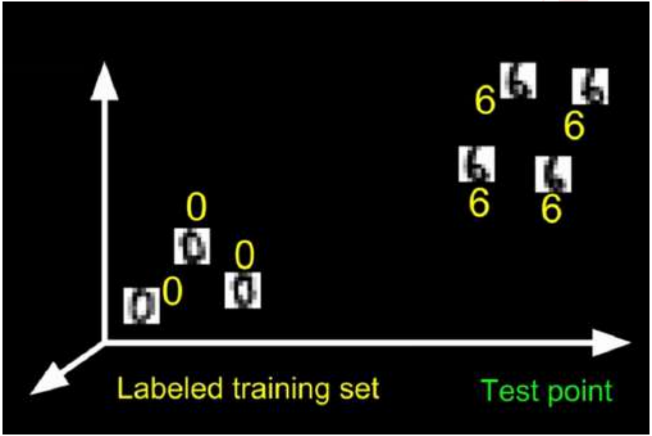
- If we can represent image in a space[3]like we did with toy example than its easy to classify it.



[2]Example images from CS50 - Harvard University.

Move forward

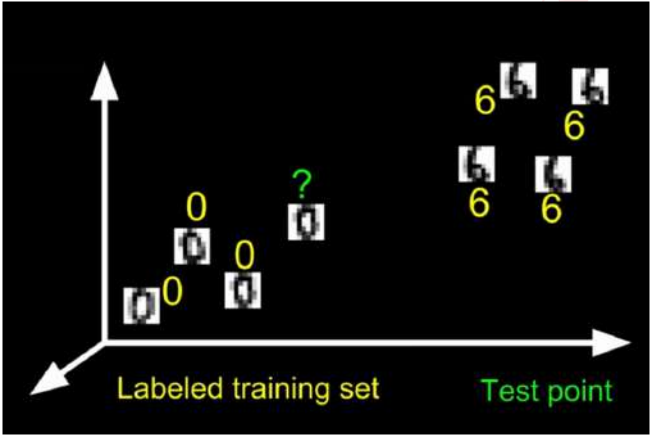
- If we can represent image in a space[3]like we did with toy example than its easy to classify it.



[2]Example images from CS50 - Harvard University.

Move forward

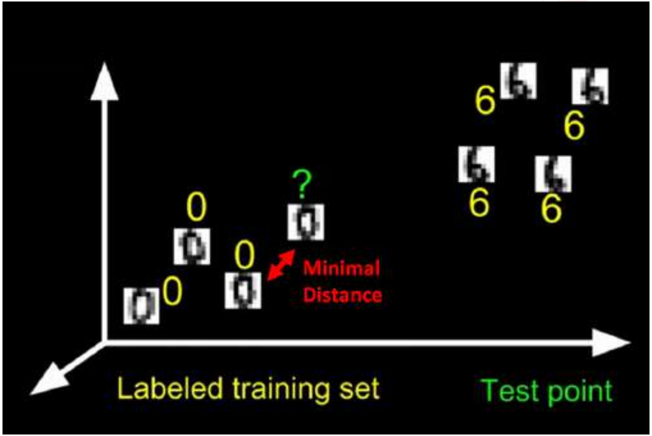
- If we can represent image in a space[3]like we did with toy example than its easy to classify it.



[2]Example images from CS50 - Harvard University.

Move forward

- If we can represent image in a space[3] like we did with toy example than its easy to classify it.

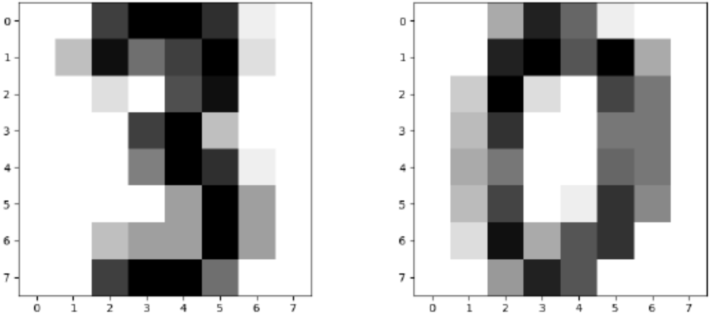


[2] Example images from CS50 - Harvard University.

Section Contents

Concrete Example: Digits Dataset

This dataset is made up of 1797, 8 x 8 images. Each image, like the one shown below, is of a hand-written digit[4].



[3] https://archive.ics.uci.edu/ml/datasets/Pen-Based+Recognition+of+Handwritten+Digits

Concrete Example: Digits Dataset

This dataset is made up of 1797, 8 x 8 images. Each image, like the one shown below, is of a hand-written digit[4].

[3] https://archive.ics.uci.edu/ml/datasets/Pen-Based+Recognition+of+Handwritten+Digits

Image Representation in Feature Space

How can we represent image in feature space?

Image Representation in Feature Space

How can we represent image in feature space?

- We can represent it with any dimension 1D, 2D, 3D , $\cdots$ nD

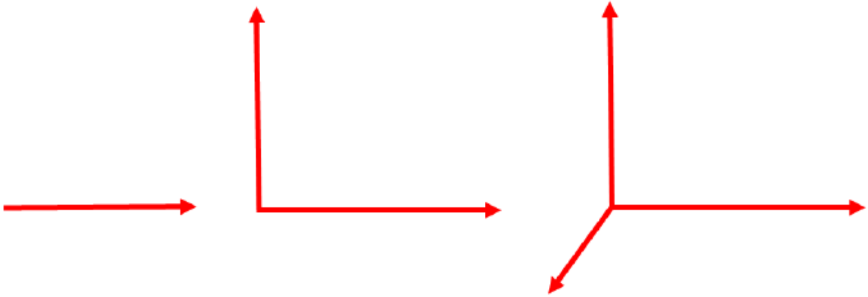
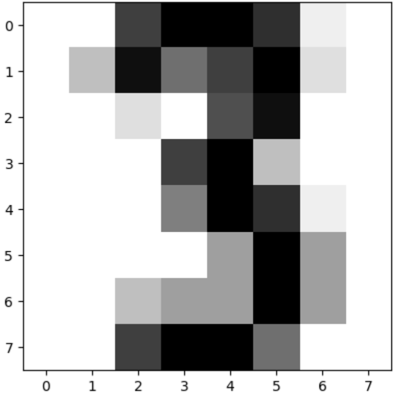


Image Representation in Feature Space

- The dataset we are working with has 8 x 8 pixels with max. pixel value of 16.
- This image could be thought of point in 64-D space.




```
[ 0.   0. 10. 14.  8.  1.  0.  0.
  0.   2. 16. 14.  6.  1.  0.  0.
  0.   0. 15. 15.  8. 15.  0.  0.
  0.   0.  5. 16. 16. 10.  0.  0.
  0.   0. 12. 15. 15. 12.  0.  0.
  0.   4. 16.  6.  4. 16.  6.  0.
  0.   8. 16. 10.  8. 16.  8.  0.
  0.   1.  8. 12. 14. 12.  1.  0.]
```

Image Representation in Feature Space

- This image could be thought of point in 64-D space.

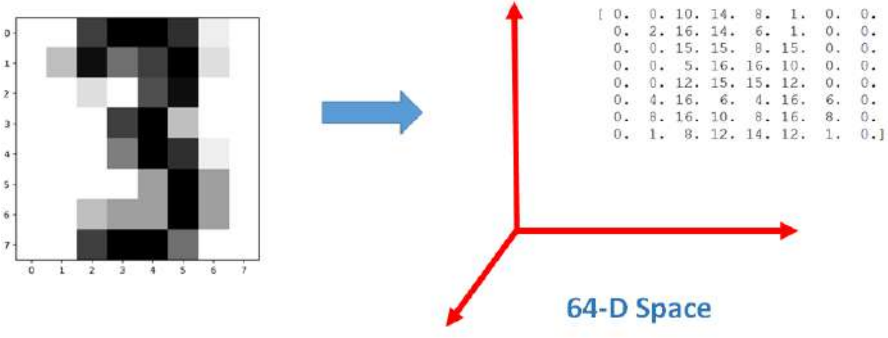


Image Representation in Feature Space

If we can represent image with a point in 64-D space, then we need to find distance of test example to training set and can assign label of nearest train example! We have a classifier!

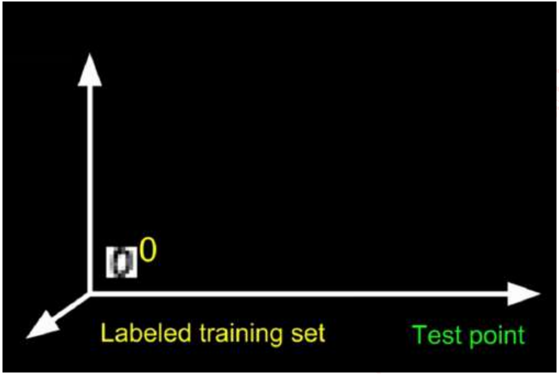


Image Representation in Feature Space

If we can represent image with a point in 64-D space, then we need to find distance of test example to training set and can assign label of nearest train example! We have a classifier!

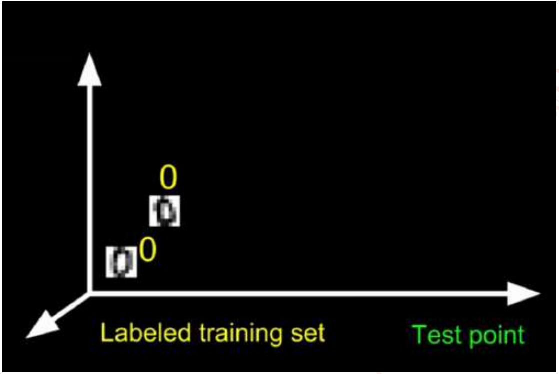


Image Representation in Feature Space

If we can represent image with a point in 64-D space, then we need to find distance of test example to training set and can assign label of nearest train example! We have a classifier!

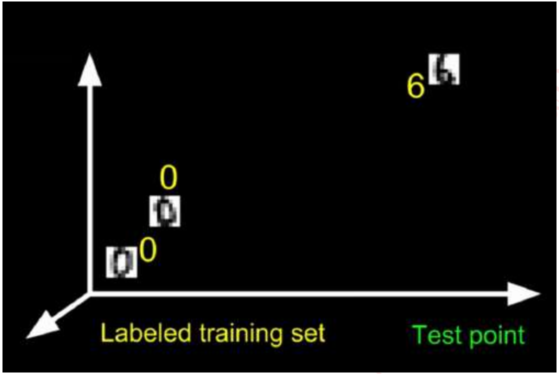


Image Representation in Feature Space

If we can represent image with a point in 64-D space, then we need to find distance of test example to training set and can assign label of nearest train example! We have a classifier!

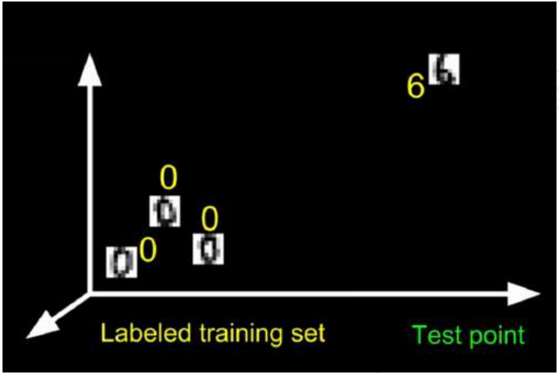


Image Representation in Feature Space

If we can represent image with a point in 64-D space, then we need to find distance of test example to training set and can assign label of nearest train example! We have a classifier!

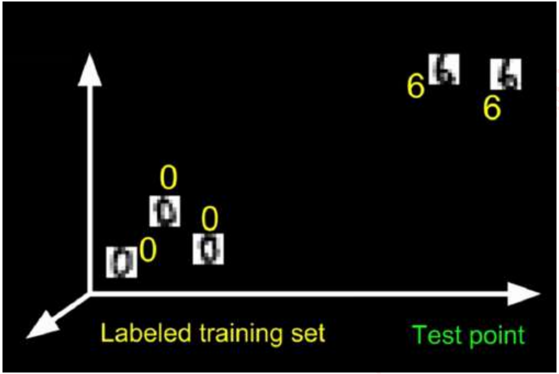


Image Representation in Feature Space

If we can represent image with a point in 64-D space, then we need to find distance of test example to training set and can assign label of nearest train example! We have a classifier!

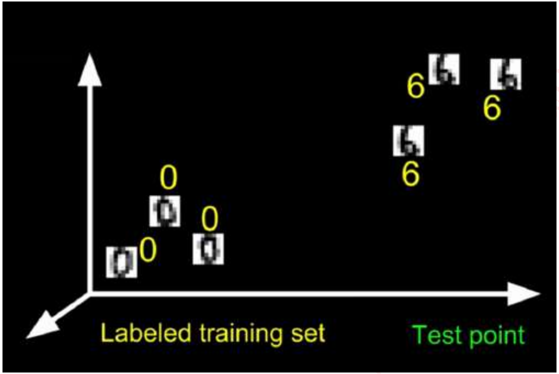


Image Representation in Feature Space

If we can represent image with a point in 64-D space, then we need to find distance of test example to training set and can assign label of nearest train example! We have a classifier!

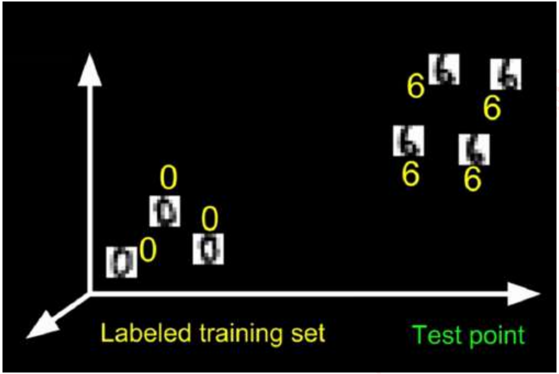


Image Representation in Feature Space

If we can represent image with a point in 64-D space, then we need to find distance of test example to training set and can assign label of nearest train example! We have a classifier!

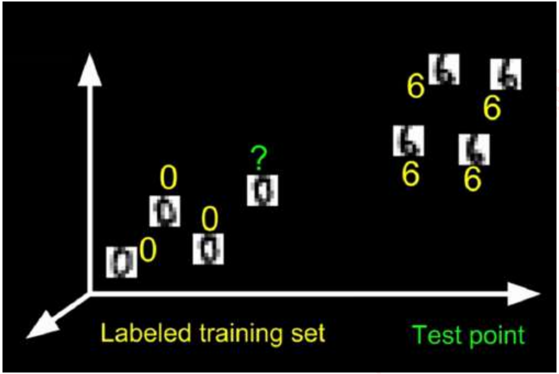


Image Representation in Feature Space

If we can represent image with a point in 64-D space, then we need to find distance of test example to training set and can assign label of nearest train example! We have a classifier!

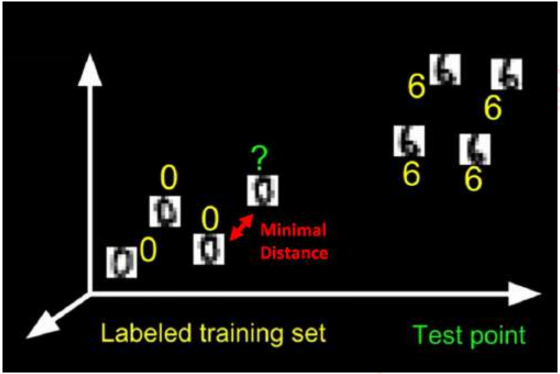


Image Distance or Similarity Measure

Distance metric uses distance function which provides a relationship metric between elements in the dataset.

Minkowski Distance:

$$dist(a,b) = \left(\sum_{i=1}^{n}(a_i - b_i)^p\right)^{\frac{1}{p}} \tag{7}$$

1. if p = 1, Manhattan Distance

$$dist_{L1}(a,b) = \sum_{i=1}^{n}(\|a_i - b_i\|) \tag{8}$$

2. if p = 2, Euclidean Distance

$$dist_{L2}(a,b) = \sqrt{\sum_{i=1}^{n}(a_i - b_i)^2} \tag{9}$$

3. if p = ∞, Chebychev Distance

Image Distance or Similarity Measure

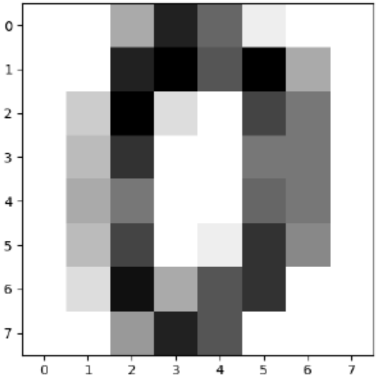
Image Distance or Similarity Measure

Section Contents

Nearest Neighbor Classifier: Python

```python
import numpy as np
import matplotlib.pyplot as plt

from sklearn import datasets
digits = datasets.load_digits()

"""
The dataset contains 1797 images. Two array:
digits.images
digits.target
"""
print(digits.images[0])
print(digits.target[0]) # label of image

# What is this number?
plt.figure()
plt.imshow(digits.images[0], cmap = plt.cm.gray_r, interpolation='nearest')
plt.show()
```

Visualization

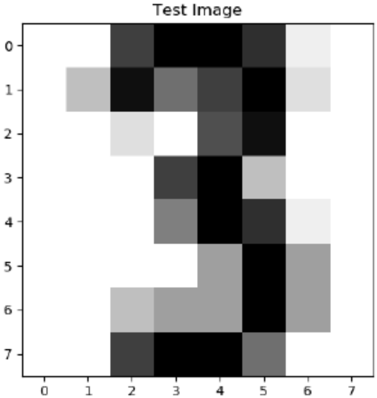


Nearest Neighbor Classifier: Python

```
#Creating training set by selecting first 10 images

#x_train = digits.images[0:10]
x_train = digits.data[0:10] # it is reshaped images in one row
y_train = digits.target[0:10]

#x_test = digits.images[345]
x_test = digits.data[345]

# To visulaize test image
plt.figure()
plt.imshow(digits.images[345], cmap = plt.cm.gray_r, interpolation='nearest')
plt.title('Test Image')
plt.show()
####
```

Training Set

Visualization



Nearest Neighbor Classifier: Python

```
"""
Step 1: Find Distance to all points in training
Step 2: Find point with minimum distance in training set
Step 3: Assign label of nearest point to test point
STEP 1
"""
def dist(x, y):
    return np.sqrt(np.sum((x-y)**2))

#Step 2: Find point with minimum distance in training set

distance=np.zeros(len(x_train))
for i in range(len(x_train)):
    distance[i]=dist(x_train[i],x_test)

min_index = np.argmin(distance)

#Step 3: Assign label of nearest point to test point
print('New point is classified in Class : ',y_train[min_index])
```

Visualization

```
    ...: print('New point is classified in Class : ',y_train[min_index])
New point is classified in Class :  3
```

Calculating Error on 100 Test Images

- Up till now we have trained model on 10 images and tested it on only one image.
- How about running / testing same model on last 100 (test) images.

Nearest Neighbor Classifier: Python

```python
"""
So far, it seems a good classifier with correct result.
How about running it for 100 test images to see how accurate it is?
"""
num=len(x_train)
no_errors=0
distance=np.zeros(num)

for j in range(1697, 1797):    # taking last 100 images as test
    x_test=digits.data[j]

    for i in range(num):     # Cal. dist. from selested test examp to all train
     examp.
        distance[i]=dist(x_train[i],x_test)

    min_index=np.argmin(distance) # labeling test example.

    if y_train[min_index] != digits.target[j]:
        no_errors +=1

print('Total error : ', no_errors)
```

Visualization

```
In [30]: print('Total error : ', no_errors)
Total error :  37
```

Improvement?

Improvement?

- How to improve accuracy?
- For 100 test examples, 37 examples are misclassified.
- Any idea?

Improvement?

Idea-1

We have used only 10 training examples to train. With only 10 samples model will not be able to capture all the variations of writings present in database. To cater different variation we need to add more training samples!

Improvement?

Idea-1

We have used only 10 training examples to train. With only 10 samples model will not be able to capture all the variations of writings present in database. To cater different variation we need to add more training samples!

Idea-2

It is possible that add more neighbors! By adding more neighbors, final label of test sample can be verified by majority voting.

Improvement?

Idea-1

We have used only 10 training examples to train. With only 10 samples model will not be able to capture all the variations of writings present in database. To cater different variation we need to add more training samples!

Idea-2

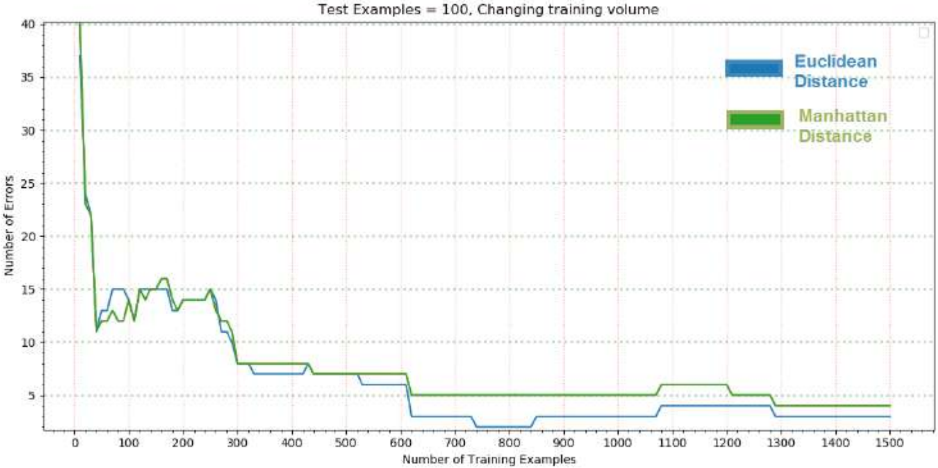
It is possible that add more neighbors! By adding more neighbors, final label of test sample can be verified by majority voting.

Idea-3

Changing distance measure? e.g. Mahalanobis distance, Bhattacharyya distance, etc.

Results: Idea 1 and Idea 3

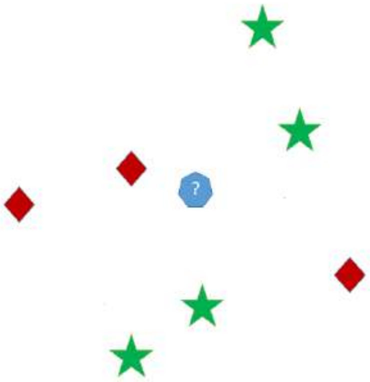
- Adding more training samples (samples added with step size of 10)
- Distance Measure Comparison



Section Contents

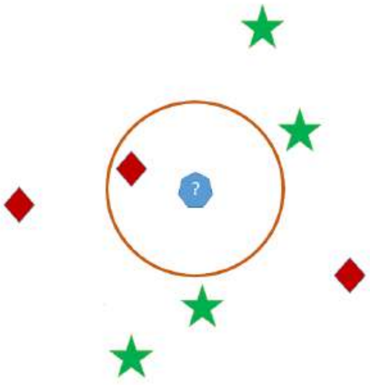
K-Nearest Neighbors

- Taking Idea-2 forward: to label query / test data we have looked only 1-neighbor. For most problems one neighbor can lead to misclassification i.e. noise in data, inter class variations in data point are less.



K-Nearest Neighbors

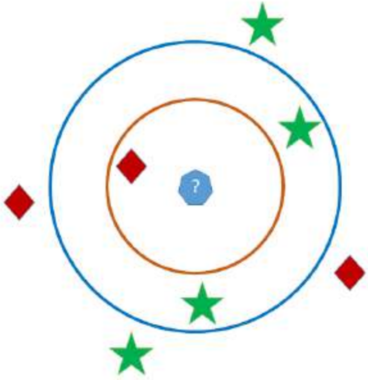
- Taking Idea-2 forward: to label query / test data we have looked only 1-neighbor. For most problems one neighbor can lead to misclassification i.e. noise in data, inter class variations in data point are less.



1. if k=1, label = diamond

K-Nearest Neighbors

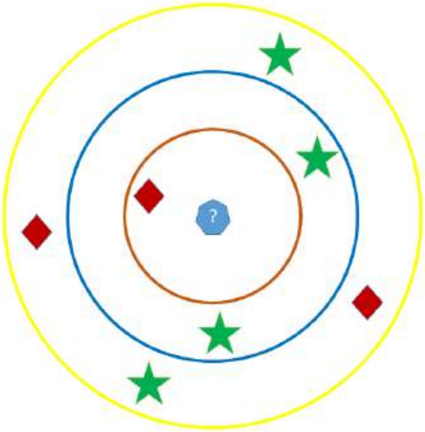
- Taking Idea-2 forward: to label query / test data we have looked only 1-neighbor. For most problems one neighbor can lead to misclassification i.e. noise in data, inter class variations in data point are less.



1. if k=1, label = diamond
2. if k=3, label = star

K-Nearest Neighbors

- Taking Idea-2 forward: to label query / test data we have looked only 1-neighbor. For most problems one neighbor can lead to misclassification i.e. noise in data, inter class variations in data point are less.



1. if k=1, label = diamond
2. if k=3, label = star
3. if k=7, label = star

K-Nearest Neighbors (k-NN): Pseudo-code

Algorithm 1 K-Nearest Neighbors (k-NN)

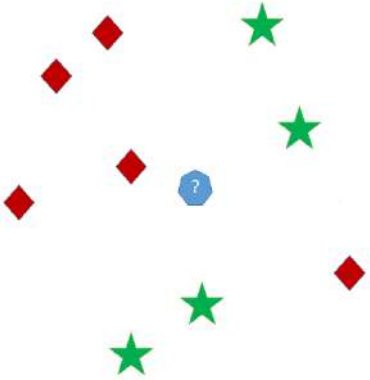
Out of the N training vectors, identify the k nearest neighbors, regardless of class label. Caution: k is chosen to be odd for a two class problem, and in general not to be a multiple of the number of classes M.

Out of these k samples, identify the number of vectors k_i , that belong to class w_i, $i = 1, 2, \ldots, m$. $\sum_i k_i = k$.

Assign x to the class w_i with the maximum number k_i of samples.

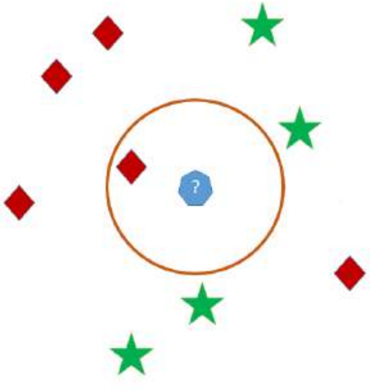
Number of Neighbors: K

- Taking Idea-2 forward: add K neighbors. It's a parameter that has to be learned for problem in hand!



Number of Neighbors: K

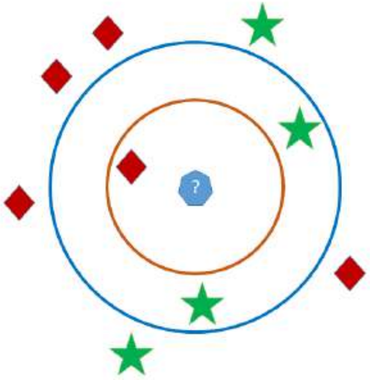
- Taking Idea-2 forward: add K neighbors. It's a parameter that has to be learned for problem in hand!



1. if k=1, label = diamond

Number of Neighbors: K

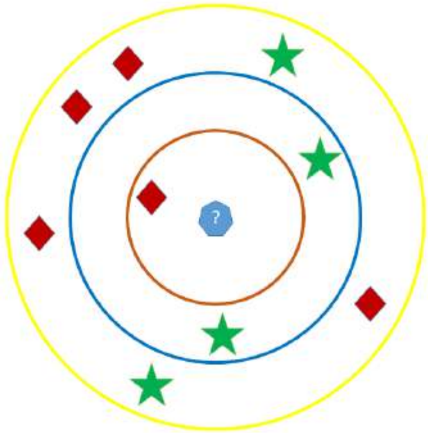
- Taking Idea-2 forward: add K neighbors. It's a parameter that has to be learned for problem in hand!



1. if k=1, label = diamond
2. if k=3, label = star

Number of Neighbors: K

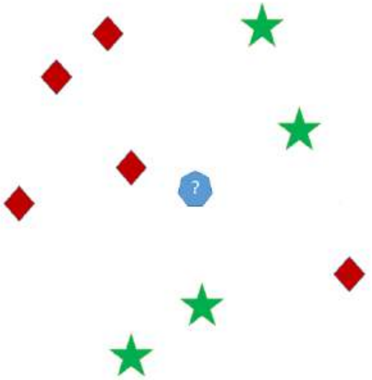
- Taking Idea-2 forward: add K neighbors. It's a parameter that has to be learned for problem in hand!



1. if k=1, label = diamond
2. if k=3, label = star
3. if k=7, label = diamond

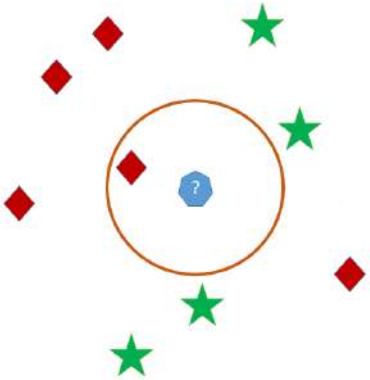
Number of Neighbors: K

- Taking Idea-2 forward: add K neighbors - improvement!



Number of Neighbors: K

- Taking Idea-2 forward: add K neighbors - improvement!



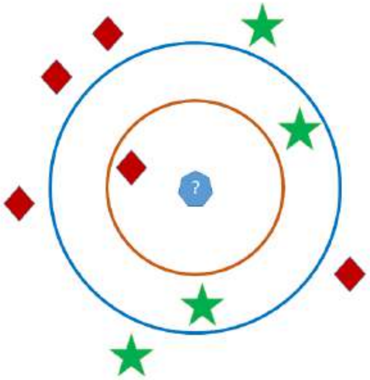
1. Distance-weighted nearest neighbor algorithm: One obvious refinement, to weight the contribution of each of the k neighbors according to their distance to the query / test point.
2. Giving greater weight to closer neighbors.

$$weight_i = \frac{1}{dist(train_i, test)^2} \quad (10)$$

3. Robust to noise

Number of Neighbors: K

- Taking Idea-2 forward: add K neighbors - improvement!



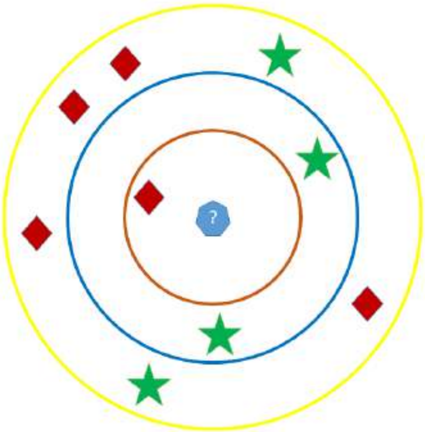
1. Distance-weighted nearest neighbor algorithm: One obvious refinement, to weight the contribution of each of the k neighbors according to their distance to the query / test point.
2. Giving greater weight to closer neighbors.

$$weight_i = \frac{1}{dist(train_i, test)^2} \quad (10)$$

3. Robust to noise

Number of Neighbors: K

- Taking Idea-2 forward: add K neighbors - improvement!



1. Distance-weighted nearest neighbor algorithm: One obvious refinement, to weight the contribution of each of the k neighbors according to their distance to the query / test point.
2. Giving greater weight to closer neighbors.

$$weight_i = \frac{1}{dist(train_i, test)^2} \quad (10)$$

3. Robust to noise

Practical issue with K-Nearest Neighbors

What is computational complexity of K-Nearest Neighbors

Practical issue with K-Nearest Neighbors

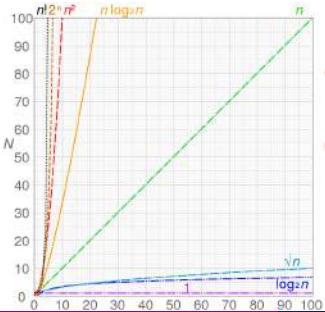
What is computational complexity of K-Nearest Neighbors

1. Compare query data / test data to all training examples.
2. Training Complexity : $\mathcal{O}(1)$

Practical issue with K-Nearest Neighbors

What is computational complexity of K-Nearest Neighbors

1. Compare query data / test data to all training examples.
2. Training Complexity : $\mathcal{O}(1)$
3. Test Complexity : $\mathcal{O}(nd)$, where n = number of training instances and d = dimensions of training data. It's linear time algorithm and that is not good!
4. Result: K-Nearest Neighbors is **slow**.



Suggestions

Any suggestions to make it fast i.e. to reduce its complexity!

Practical issue with K-Nearest Neighbors

1. Curse of dimensionality (more on this later)
   - Reduce d by removing irrelevant features (feature selection).

Practical issue with K-Nearest Neighbors

1. Curse of dimensionality (more on this later)
   - Reduce d by removing irrelevant features (feature selection).
2. Reduce "n"
   - Don't compare all n.

Practical issue with K-Nearest Neighbors

1. Curse of dimensionality (more on this later)
   - Reduce d by removing irrelevant features (feature selection).
2. Reduce "n"
   - Don't compare all n.
   - Quickly (ideally in constant time or in log time) identify potential m nearest neighbors, $| \, m << n$. Thus, complexity will reduce to $\mathcal{O}(md)$.

Practical issue with K-Nearest Neighbors

1. Curse of dimensionality (more on this later)
   - Reduce d by removing irrelevant features (feature selection).
2. Reduce "n"
   - Don't compare all n.
   - Quickly (ideally in constant time or in log time) identify potential m nearest neighbors, $\mid m << n$. Thus, complexity will reduce to $\mathcal{O}(md)$.
   - **KD-tree**: Its a data structure and can find m nearest neighbors in $\mathcal{O}(log_2 n)$. Works well for low dimensional data but it can miss neighbors.

Practical issue with K-Nearest Neighbors

1. Curse of dimensionality (more on this later)
   - Reduce d by removing irrelevant features (feature selection).
2. Reduce "n"
   - Don't compare all n.
   - Quickly (ideally in constant time or in log time) identify potential m nearest neighbors, $\mid m << n$. Thus, complexity will reduce to $\mathcal{O}(md)$.
   - **KD-tree**: Its a data structure and can find m nearest neighbors in $\mathcal{O}(log_2 n)$. Works well for low dimensional data but it can miss neighbors.
   - **Inverted list**: data structure for storing a mapping from data.

Practical issue with K-Nearest Neighbors

1. Curse of dimensionality (more on this later)
   - Reduce d by removing irrelevant features (feature selection).
2. Reduce "n"
   - Don't compare all n.
   - Quickly (ideally in constant time or in log time) identify potential m nearest neighbors, $\mid m << n$. Thus, complexity will reduce to $\mathcal{O}(md)$.
   - **KD-tree**: Its a data structure and can find m nearest neighbors in $\mathcal{O}(log_2 n)$. Works well for low dimensional data but it can miss neighbors.
   - **Inverted list**: data structure for storing a mapping from data.
   - **Locality-sensitive hashing**: hashes similar input items into the same "bucket" with high probability. Its a way to reduce the dimensionality while preserving relative distances between items. It can also miss neighbors.

Inductive Bias

Inductive Bias

What is the inductive bias of k-NN classifier?

Inductive Bias

Inductive Bias

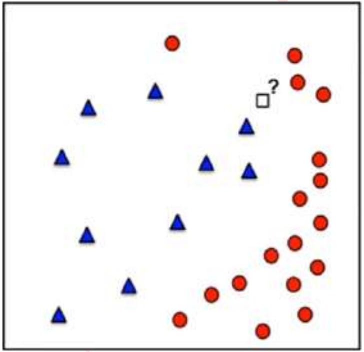
What is the inductive bias of k-NN classifier?

Inductive Bias

For k-NN classifier inductive bias corresponds to an assumption that the classification of an test instance, will be most similar to the classification of other instances that are nearby in (Euclidean) space.

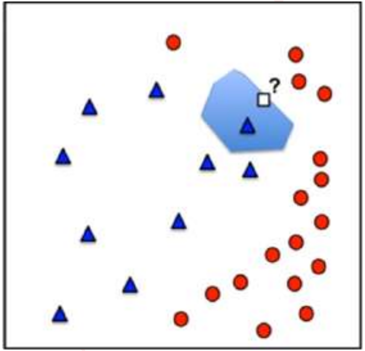
Decision Boundary

Voronoi Cells / Diagram / Tessellation



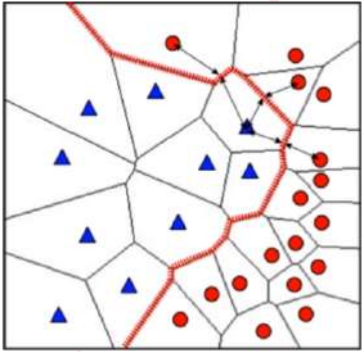
Decision Boundary

Voronoi Cells / Diagram / Tessellation

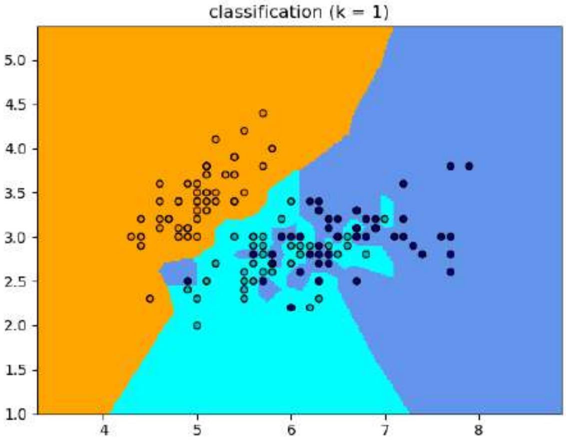


Decision Boundary

Voronoi Cells / Diagram / Tessellation



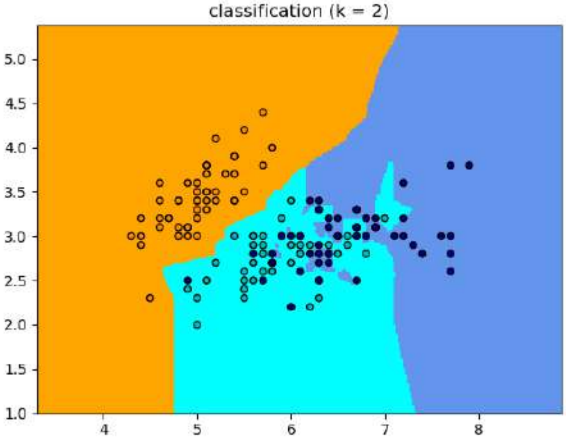
Decision Boundary



A small value of k could lead to overfitting as well as a big value of k can lead to underfitting. Overfitting imply that the model is well on the training data but has poor performance when new data is coming i.e. high variance.

*5

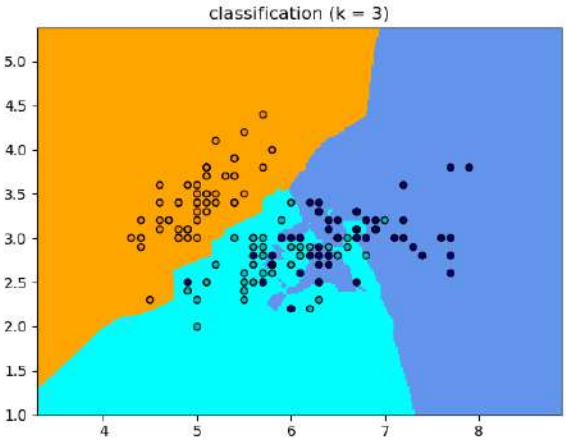
Decision Boundary



A small value of k could lead to overfitting as well as a big value of k can lead to underfitting. Overfitting imply that the model is well on the training data but has poor performance when new data is coming i.e. high variance.

*5

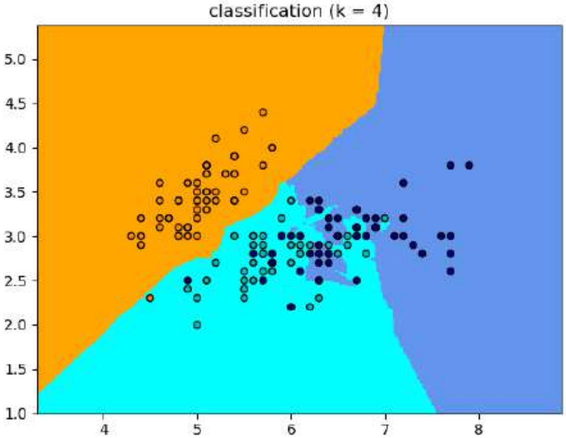
Decision Boundary



A small value of k could lead to overfitting as well as a big value of k can lead to underfitting. Overfitting imply that the model is well on the training data but has poor performance when new data is coming i.e. high variance.

*5

Decision Boundary



A small value of k could lead to overfitting as well as a big value of k can lead to underfitting. Overfitting imply that the model is well on the training data but has poor performance when new data is coming i.e. high variance.

*5

Decision Boundary

Decision Boundary : Exercise

Consider following $2D$ dataset:

1. $+ve$: $(-1, 3), (-2, 2), (1, 1)$
2. $-ve$: $(2, 1), (-1, 2), (-1, 0)$

Draw decision boundary for $1 - NN$ classifier with Euclidean distance.

Decision Boundary

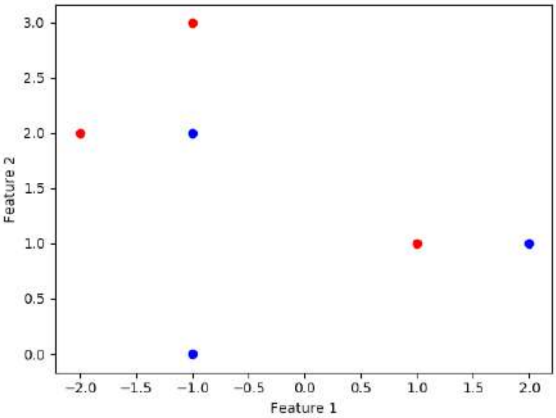
$+ve$ examples are shown in color red, while $-ve$ examples are shown in color blue.

Decision Boundary : Exercise

Consider following $2D$ dataset:

1. $+ve$: $(-1, 3), (-2, 2), (1, 1)$
2. $-ve$: $(2, 1), (-1, 2), (-1, 0)$

Draw decision boundary for $1 - NN$ classifier with Euclidean distance.



Decision Boundary

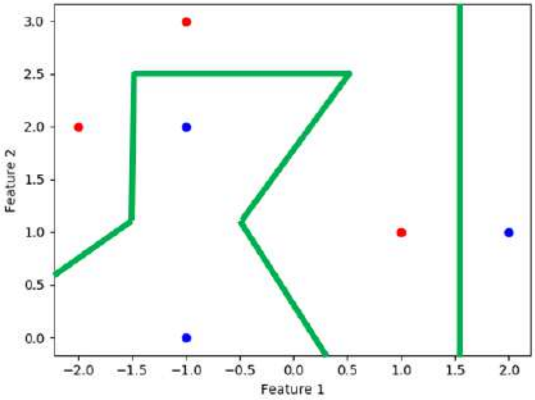
$+ve$ examples are shown in color red, while $-ve$ examples are shown in color blue.

Decision Boundary : Exercise

Consider following $2D$ dataset:

1. $+ve$: $(-1,3), (-2,2), (1,1)$
2. $-ve$: $(2,1), (-1,2), (-1,0)$

Draw decision boundary for $1-NN$ classifier with Euclidean distance.



Instance-based Learning

- Instance-based learning methods (Lazy learners) simply store the training examples (lazy learner vs Eager learner).
- Generalizing beyond given examples is postponed until a new instance gets classified.
- A key advantage: instead of estimating the target function once for the entire instance space, it learns target function for each new instance to be classified.
- Instance-based learning includes:
  1. k-Nearest Neighbor (Instances represented as points in a Euclidean space)
  2. Locally weighted regression methods (Constructs local approximation)
  3. Case-based reasoning methods (Uses symbolic representations and knowledge-based inference)

Instance-based Learning

- Disadvantages:
  1. Cost of classifying new instances is high (nearly all computation takes place at classification time rather than when the training examples are first encountered (eager learner approach)).
  2. All attributes of the instances are considered when attempting to retrieve similar training examples from memory.

Section Contents

Curse of Dimensionality

Curse of Dimensionality

- One practical issue in applying k-NN classifier is that the distance between instances is calculated based on all attributes / features of the instance / example.

Curse of Dimensionality

Curse of Dimensionality

- One practical issue in applying k-NN classifier is that the distance between instances is calculated based on all attributes / features of the instance / example.
- This is in contrast to many other ML algorithms i.e. Decision Tree where learning selects only a subset of the instance attributes when forming the hypothesis.

Curse of Dimensionality

Curse of Dimensionality

- One practical issue in applying k-NN classifier is that the distance between instances is calculated based on all attributes / features of the instance / example.
- This is in contrast to many other ML algorithms i.e. Decision Tree where learning selects only a subset of the instance attributes when forming the hypothesis.
- Consider applying k-NN classifier to a problem that has 20 features, but only 2 attributes are relevant or inter-class variability depends only on 2 features. In this case k-NN distance function can give misleading results.

Curse of Dimensionality

Curse of Dimensionality

- One practical issue in applying k-NN classifier is that the distance between instances is calculated based on all attributes / features of the instance / example.
- This is in contrast to many other ML algorithms i.e. Decision Tree where learning selects only a subset of the instance attributes when forming the hypothesis.
- Consider applying k-NN classifier to a problem that has 20 features, but only 2 attributes are relevant or inter-class variability depends only on 2 features. In this case k-NN distance function can give misleading results.
- This difficulty, which arises when many irrelevant attributes are present, is sometimes referred to as the curse of dimensionality.

Curse of Dimensionality

Formally, curse of dimensionality

As the number of features or dimensions grows, the amount of data need to be generalized accurately grows exponentially.

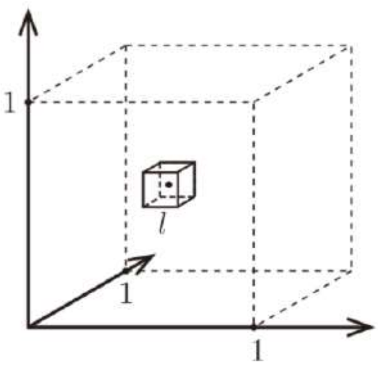
- The $K-$NN classifier makes the assumption that similar points share similar labels.

Curse of Dimensionality

Formally, curse of dimensionality

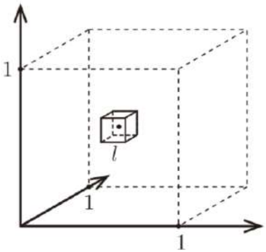
As the number of features or dimensions grows, the amount of data need to be generalized accurately grows exponentially.

- The $K-$NN classifier makes the assumption that similar points share similar labels.
- Unfortunately, in high dimensional spaces, points that are drawn from a probability distribution, tend to never be close together, Example –>:
- How big this little box has to be to encapsulate all $K-$ nearest neighbors of a test point?



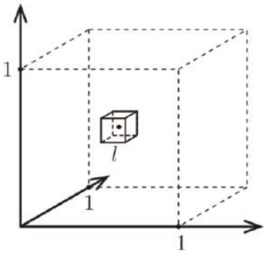
Curse of Dimensionality

- Formally, this is unit cube of d dimensions i.e. $\mathbb{R}^d$. All training data is sampled uniformly within this cube.

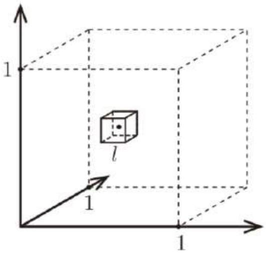


Curse of Dimensionality

- Formally, this is unit cube of d dimensions i.e. $\mathbb{R}^d$. All training data is sampled uniformly within this cube.
- Considering the $k = 10$ nearest neighbors and $n = 1000$.



Curse of Dimensionality

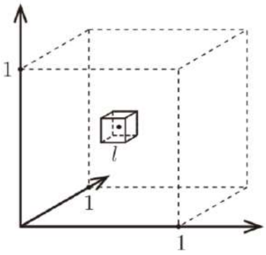


- Formally, this is unit cube of d dimensions i.e. $\mathbb{R}^d$. All training data is sampled uniformly within this cube.
- Considering the $k = 10$ nearest neighbors and $n = 1000$.
- Then: volume of box =
  - $\ell^d \approx \frac{k}{n}$ (as the box contains k points out of n). This says, roughly volume is same as the ratio of the points, because of uniform distribution.

$$\ell^d \approx \frac{k}{n} \implies \ell \approx \left(\frac{k}{n}\right)^{1/d}$$

- How large is ℓ?

Curse of Dimensionality



- Formally, this is unit cube of d dimensions i.e. $\mathbb{R}^d$. All training data is sampled uniformly within this cube.
- Considering the $k = 10$ nearest neighbors and $n = 1000$.
- Then: volume of box =
  - $\ell^d \approx \frac{k}{n}$ (as the box contains k points out of n). This says, roughly volume is same as the ratio of the points, because of uniform distribution.

$$\ell^d \approx \frac{k}{n} \implies \ell \approx \left(\frac{k}{n}\right)^{1/d}$$

- How large is ℓ?

d	ℓ
2	0.1
10	0.63
100	0.955
1000	0.9954

Curse of Dimensionality

Problems identified

- So as d dimension increases almost the entire space is needed to find the $10 - NN$. What does it mean?

Curse of Dimensionality

Problems identified

- So as d dimension increases almost the entire space is needed to find the $10 - NN$. What does it mean?
- 10 points are at the edge of smaller cube and that edge of cube is almost touching outer cube that that has remain 9990 points.

Curse of Dimensionality

Problems identified

- So as d dimension increases almost the entire space is needed to find the $10 - NN$. What does it mean?
- 10 points are at the edge of smaller cube and that edge of cube is almost touching outer cube that that has remain 9990 points.
- This breaks down the k-NN assumptions, because the k-NN are not particularly closer (and therefore more similar) than any other data points in the training set.
- So the distance between two randomly drawn data points increases drastically with their dimensionality. Neighbors are not close! All the points whether they are in neighbors or not are roughly at the same distance.

Curse of Dimensionality

Should we remove k-NN from toolkit?

- Not all is lost. Data may lie in low dimensional subspace or on sub-manifolds. Example: natural images (digits, faces (they are not uniformly distributed)). Here, the true dimensionality of the data can be much lower than its ambient space.

Curse of Dimensionality

Should we remove k-NN from toolkit?

- Not all is lost. Data may lie in low dimensional subspace or on sub-manifolds. Example: natural images (digits, faces (they are not uniformly distributed)). Here, the true dimensionality of the data can be much lower than its ambient space.
- k-NN would work if data has low intrinsic dimensionality.

Curse of Dimensionality

Should we remove k-NN from toolkit?

- Not all is lost. Data may lie in low dimensional subspace or on sub-manifolds. Example: natural images (digits, faces (they are not uniformly distributed)). Here, the true dimensionality of the data can be much lower than its ambient space.
- k-NN would work if data has low intrinsic dimensionality.
- Ref Figure [a]: a manifold is a topological space that locally resembles Euclidean space near each point, but globally it is not. For k-NN it works as only nearby points are considered.
- Human faces are a typical example of an intrinsically low dimensional data set. Although an image of a face may require 10M pixels, a person may be able to describe this person with less than 50 attributes / features (e.g. male/female, blond/dark hair, ...) along which faces vary.

[a] Image courtesy Dr. Kilian Weinberger

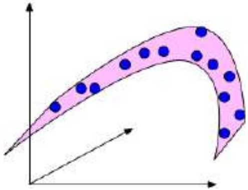


Figure 1: An example of a data set in 3D that is drawn from an underlying 2D manifold. The blue points are confined to the pink surface area, which is embedded in a 3D ambient space.

Dimensionality Reduction

- Generally, when the number of features are very large (relative to the number of observations in your dataset (not completely true, as in the case of k-NN)), algorithms struggle to train effective models.
- Preprocessing of dataset is always recommended before applying machine learning algorithm.
- One of the main step of preprocessing is dimensionality reduction. Approaches for dimensionality reduction can be divided into feature selection and feature extraction.

Dimensionality Reduction

- Generally, when the number of features are very large (relative to the number of observations in your dataset (not completely true, as in the case of k-NN)), algorithms struggle to train effective models.
- Preprocessing of dataset is always recommended before applying machine learning algorithm.
- One of the main step of preprocessing is dimensionality reduction. Approaches for dimensionality reduction can be divided into feature selection and feature extraction.
  1. Feature selection : try to find a subset of the input variables/ features. The three strategies are:
     1. the filter strategy (e.g. information gain)
     2. the wrapper strategy (e.g. search guided by accuracy),
     3. and the embedded strategy

Dimensionality Reduction

- Generally, when the number of features are very large (relative to the number of observations in your dataset (not completely true, as in the case of k-NN)), algorithms struggle to train effective models.
- Preprocessing of dataset is always recommended before applying machine learning algorithm.
- One of the main step of preprocessing is dimensionality reduction. Approaches for dimensionality reduction can be divided into feature selection and feature extraction.
    1. Feature selection : try to find a subset of the input variables/ features. The three strategies are:
        1. the filter strategy (e.g. information gain)
        2. the wrapper strategy (e.g. search guided by accuracy),
        3. and the embedded strategy
    2. Feature extraction / projection / transformation : Feature projection (also called Feature extraction) transforms the data from the high-dimensional space to a space of fewer dimensions. Mostly used technique for feature extraction, principal component analysis (PCA), performs a linear mapping of the data to a lower-dimensional space in such a way that the variance of the data in the low-dimensional representation is maximized.

Feature Selection Algorithm

Problem:

n features $\rightarrow m$ features : $m \leq n$

How hard is this problem?

1. Linear
2. Polynomial
3. Exponential

Feature Selection Algorithm

Problem:

n features $\rightarrow$ m features : $m \leq n$

How hard is this problem?

1. Linear
2. Polynomial
3. Exponential

Solution:

- It's like choosing a subset of n features that gives best score. $f(n) \rightarrow$ Score

Feature Selection Algorithm

Problem:

n features $\rightarrow$ m features : $m \leq n$

How hard is this problem?

1. Linear
2. Polynomial
3. Exponential

Solution:

- It's like choosing a subset of n features that gives best score. $f(n) \rightarrow$ Score
- Intuitively, need to create all possible subsets of n features and to try which one is best.

Feature Selection Algorithm

Problem:

n features $\rightarrow$ m features : $m \leq n$

How hard is this problem?

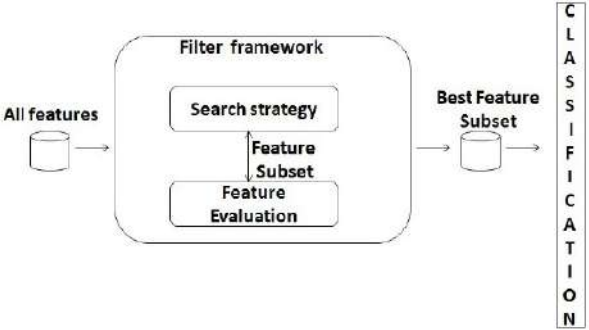
1. Linear
2. Polynomial
3. Exponential

Solution:

- It's like choosing a subset of n features that gives best score. $f(n) \rightarrow$ Score
- Intuitively, need to create all possible subsets of n features and to try which one is best.
- Exponential number of subsets i.e.

$$\sum_{0 \leq m \leq n} \binom{n}{m} = 2^n$$

Feature Selection Algorithm: Filtering



Filtering based approach

- Feature selection algorithm doesn't take feed back from final classification / learning algorithm to score selected feature subset.
- Selection criterion is independent from classification / learning criterion.

Feature Selection Algorithm: Filtering

Advantage:

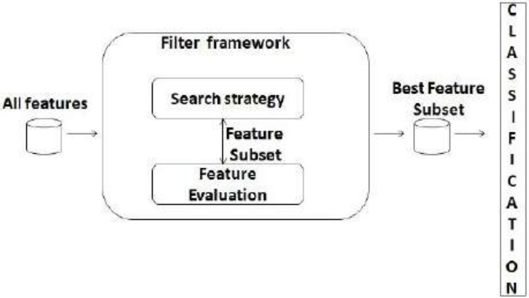
1. Fast

Disadvantage:

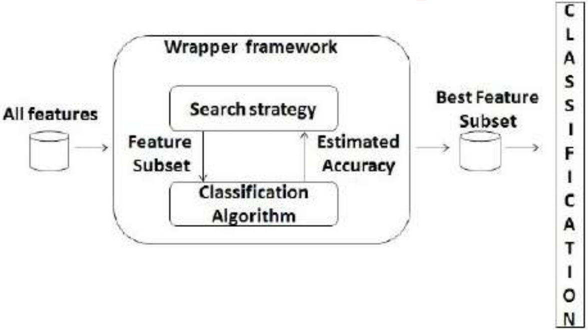
1. There isn't any feedback from learning algorithm.
2. Features are scored in isolation.

Examples:

1. Decision trees (e.g. using inductive bias of DT to learn features than using k-NN to classify)
2. Statistical tests:
   1. Pearson Correlation
   2. Chi-square
   3. Gini Index



Feature Selection Algorithm: Wrapper



Wrapper based approach

- Feature selection algorithm, after selecting subset of features gets feedback from final classification / learning algorithm to score selected feature subset.

Feature Selection Algorithm: Wrapper

Advantage:

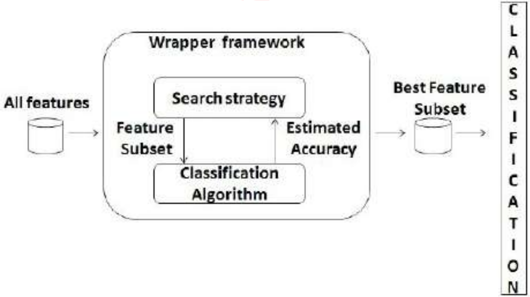
1. With feed back from learning algorithm, feature selection is optimal.
2. Takes into account learning bias of final learning algorithm.

Disadvantage:

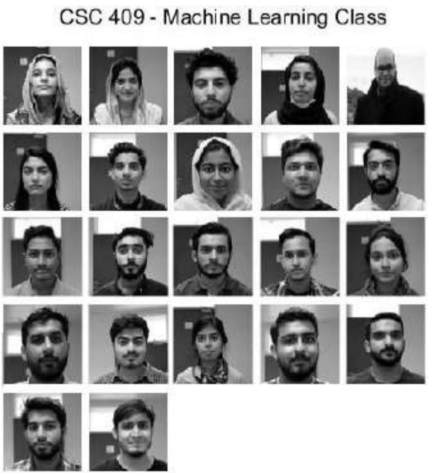
1. Very slow

Examples:

1. Recursive Feature Elimination
2. Genetic Algorithm
3. Forward Search Algorithm
4. Backward Search Algorithm (consider football team, remove player who is not performing)



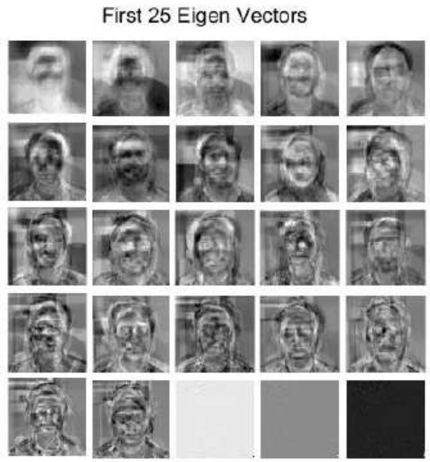
Principal Component Analysis (PCA)



1. Construct the covariance matrix from d-dimensional dataset D.
2. Decompose the covariance matrix into its Eigenvectors and Eigenvalues.

*6

Principal Component Analysis (PCA)

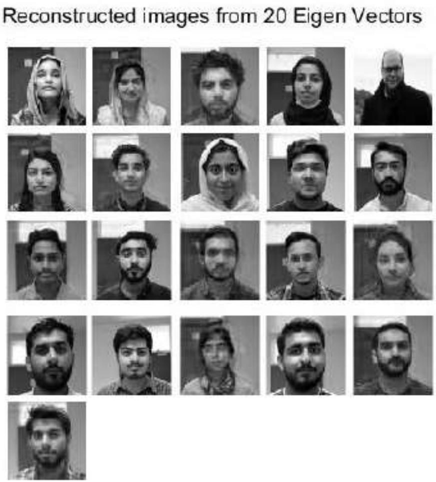


*6

3. Sort the eigenvalues by decreasing order to rank the corresponding eigenvectors.
4. Select k eigenvectors which correspond to the k largest eigenvalues, where k is the dimensionality of the new feature subspace ($k \leq d$).

[6] Matlab demo

Principal Component Analysis (PCA)



*6

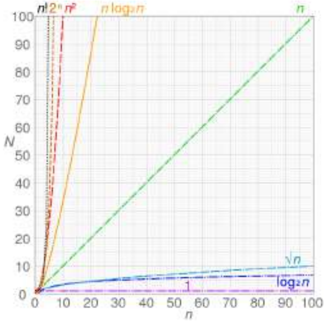
5. Construct a projection matrix W from the "top" k eigenvectors.
6. Transform the d-dimensional input dataset D using the projection matrix W to obtain the new k-dimensional feature subspace.

[6] Matlab demo

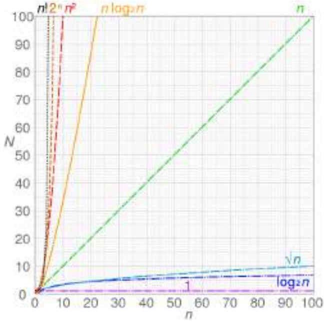
Section Contents

Practical issue with K-Nearest Neighbors

Practical issue with K-Nearest Neighbors

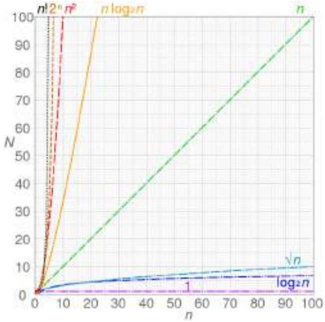


Practical issue with K-Nearest Neighbors



What is computational complexity of K-Nearest Neighbors

Practical issue with K-Nearest Neighbors



Suggestions

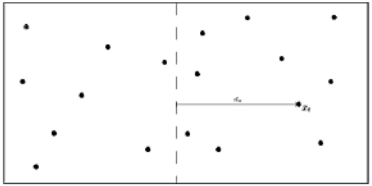
We can reduce neighborhood search complexity using appropriate data structure.

What is computational complexity of K-Nearest Neighbors

1. Compare query data / test data to all training examples.
2. Training Complexity : $\mathcal{O}(1)$
3. Test Complexity : $\mathcal{O}(nd)$, where n = number of training instances and d = dimensions of training data. It's linear time algorithm and that is not good!
4. Result: K-Nearest Neighbors is **slow**.

KD-tree data structure for kNN search

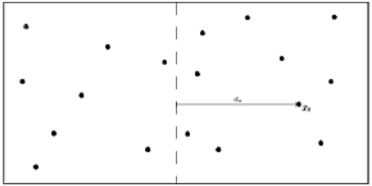
- Consider one neighbor case.
- Claim: Just look for the nearest neighbor in the partition in which test / query point lies. Proof?



2x speedup

KD-tree data structure for kNN search

- Consider one neighbor case.
- Claim: Just look for the nearest neighbor in the partition in which test / query point lies. Proof?



2x speedup

- Identify which side the test/query point lies in, e.g. the right side.
- Find the NN x_{NN}^{R} of x_t in the same side. The R denotes that nearest neighbor is also on the right side.
- Compute the distance between x_t and the dividing "wall". Denote this as d_w. IF

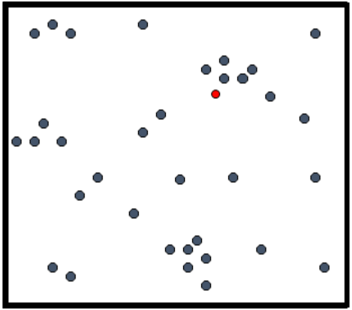
$$d_w > d(x_t, x_{\mathrm{NN}}^{R})$$

we got 2x speedup.

- Simply, if the distance to the partition is larger than the distance to closest neighbor, it means none of the data points inside that partition can be closer.

Space division by KD-tree data structure

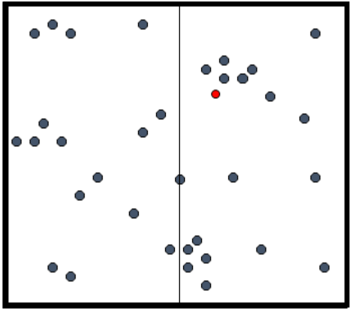
- We can split feature space again to gain more speedup (like previous example)



- The general idea of KD-trees is to partition the feature space.
- Only one-dimensional (axis aligned) splits. Instead of splitting in the middle, choose the split "carefully" (many variations).
- By using KD-tree lots of data points immediately gets discarded from search space as their partition is further away than k closest neighbors.

Space division by KD-tree data structure

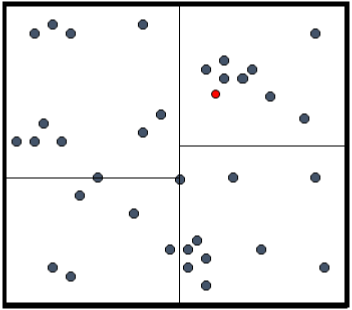
- We can split feature space again to gain more speedup (like previous example)



- The general idea of KD-trees is to partition the feature space.
- Only one-dimensional (axis aligned) splits. Instead of splitting in the middle, choose the split "carefully" (many variations).
- By using KD-tree lots of data points immediately gets discarded from search space as their partition is further away than k closest neighbors.

Space division by KD-tree data structure

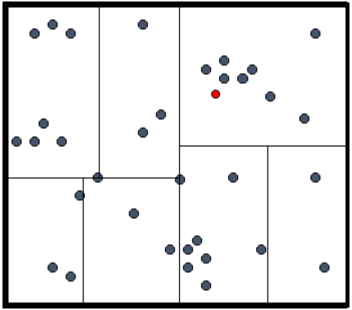
- We can split feature space again to gain more speedup (like previous example)



- The general idea of KD-trees is to partition the feature space.
- Only one-dimensional (axis aligned) splits. Instead of splitting in the middle, choose the split "carefully" (many variations).
- By using KD-tree lots of data points immediately gets discarded from search space as their partition is further away than k closest neighbors.

Space division by KD-tree data structure

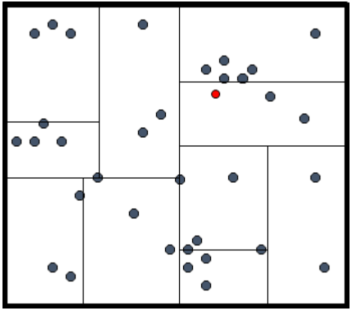
- We can split feature space again to gain more speedup (like previous example)



- The general idea of KD-trees is to partition the feature space.
- Only one-dimensional (axis aligned) splits. Instead of splitting in the middle, choose the split "carefully" (many variations).
- By using KD-tree lots of data points immediately gets discarded from search space as their partition is further away than k closest neighbors.

Space division by KD-tree data structure

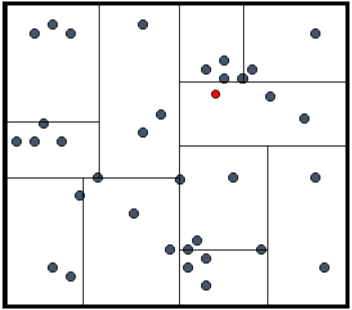
- We can split feature space again to gain more speedup (like previous example)



- The general idea of KD-trees is to partition the feature space.
- Only one-dimensional (axis aligned) splits. Instead of splitting in the middle, choose the split "carefully" (many variations).
- By using KD-tree lots of data points immediately gets discarded from search space as their partition is further away than k closest neighbors.

Space division by KD-tree data structure

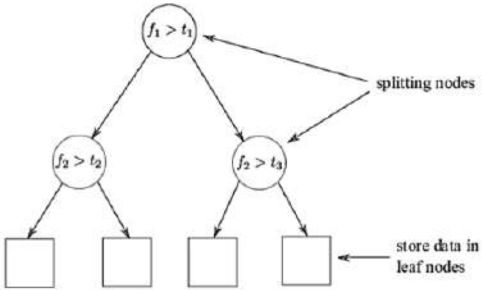
- We can split feature space again to gain more speedup (like previous example)



- The general idea of KD-trees is to partition the feature space.
- Only one-dimensional (axis aligned) splits. Instead of splitting in the middle, choose the split "carefully" (many variations).
- By using KD-tree lots of data points immediately gets discarded from search space as their partition is further away than k closest neighbors.

Space division by KD-tree data structure

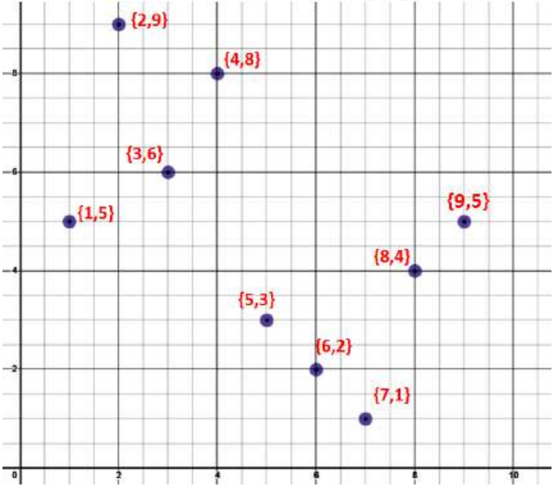
- We can split feature space again to gain more speedup (like previous example)



- The general idea of KD-trees is to partition the feature space.
- Only one-dimensional (axis aligned) splits. Instead of splitting in the middle, choose the split “carefully” (many variations).
- By using KD-tree lots of data points immediately gets discarded from search space as their partition is further away than k closest neighbors.

KD-tree data structure

Example Dataset



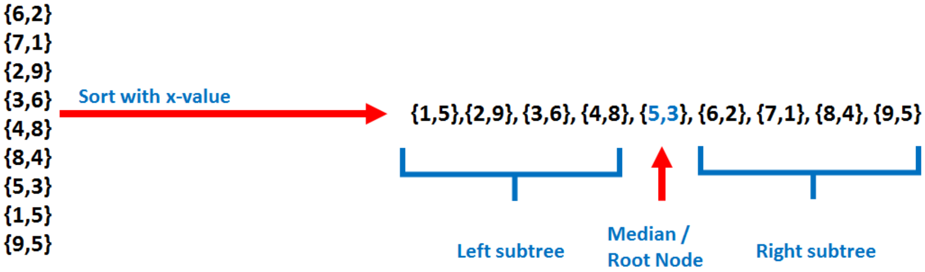
KD-tree data structure

{6,2}
{7,1}
{2,9}
{3,6}
{4,8}
{8,4}
{5,3}
{1,5}
{9,5}

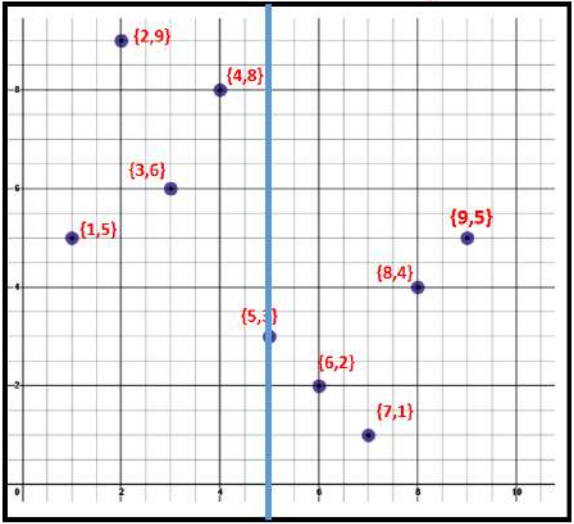
Sort with x-value →

{1,5},{2,9}, {3,6}, {4,8}, {5,3}, {6,2}, {7,1}, {8,4}, {9,5}

KD-tree data structure



KD-tree data structure



KD-tree data structure

{1,5},{2,9}, {3,6}, {4,8}, {5,3}, {6,2}, {7,1}, {8,4}, {9,5}

Left subtree

Median / Root Node

Right subtree

Sort subtrees using y-axis

{1,5},{3,6}, {4,8}, {2,9}, {5,3}, {7,1}, {6,2}, {8,4}, {9,5}

KD-tree data structure

{1,5},{2,9}, {3,6}, {4,8}, {5,3}, {6,2}, {7,1}, {8,4}, {9,5}

Left subtree | Median / Root Node | Right subtree

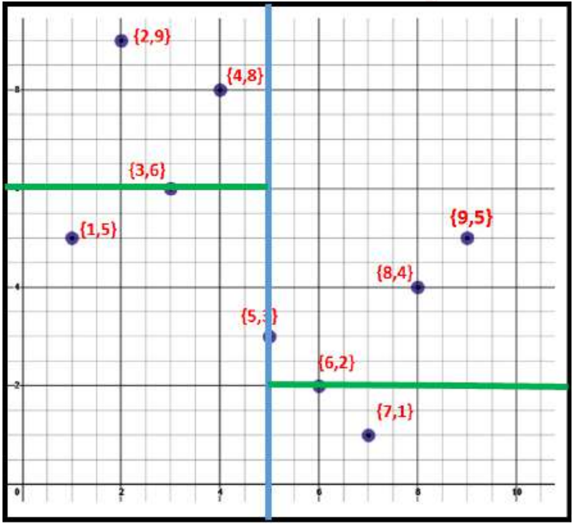
Sort subtrees using y-axis

{1,5},{3,6}, {4,8}, {2,9}, {5,3}, {7,1}, {6,2}, {8,4}, {9,5}

Find root node

{1,5},{3,6}, {4,8}, {2,9}, {5,3}, {7,1}, {6,2}, {8,4}, {9,5}

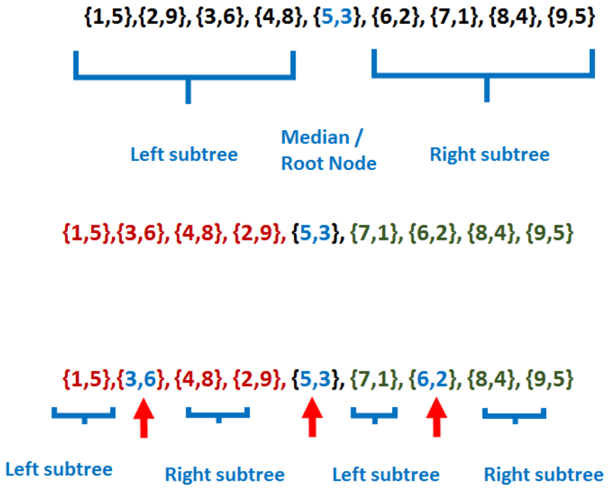
KD-tree data structure



KD-tree data structure

Sort subtrees using y-axis

Find root node



KD-tree data structure

{1,5},{3,6}, {4,8}, {2,9}, {5,3}, {7,1}, {6,2}, {8,4}, {9,5}

Left subtree Right subtree Left subtree Right subtree

Sort with x-value

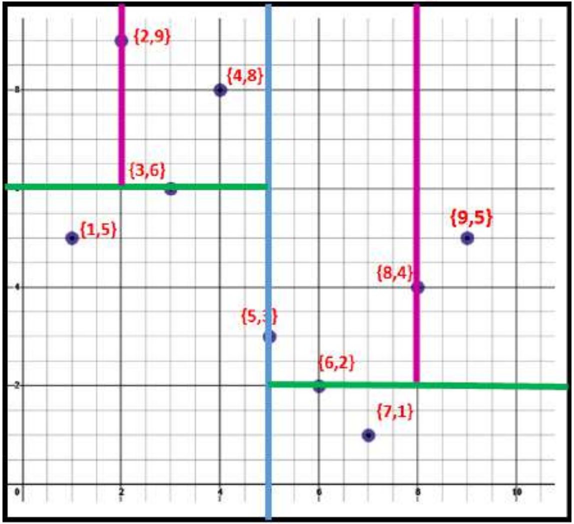
{1,5},{3,6}, {2,9}, {4,8}, {5,3}, {7,1}, {6,2}, {8,4}, {9,5}

Find root node

{1,5},{3,6}, {2,9}, {4,8}, {5,3}, {7,1}, {6,2}, {8,4}, {9,5}

No more elements to sort as only one element in each half is left!

KD-tree data structure

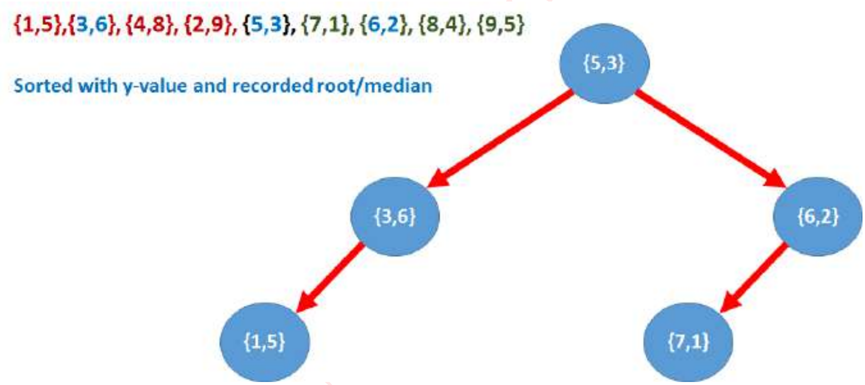


KD-tree data structure

{1,5},{2,9}, {3,6}, {4,8}, {5,3}, {6,2}, {7,1}, {8,4}, {9,5}

Sorted with x-value and recorded root/median

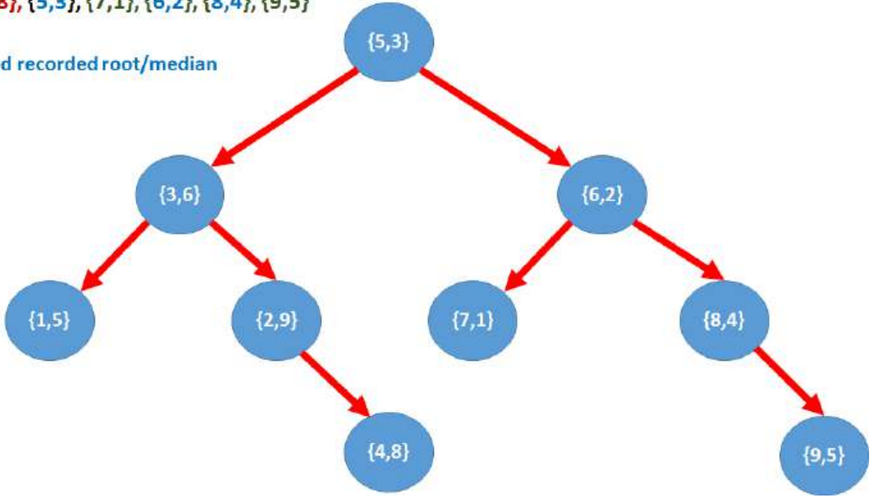
KD-tree data structure



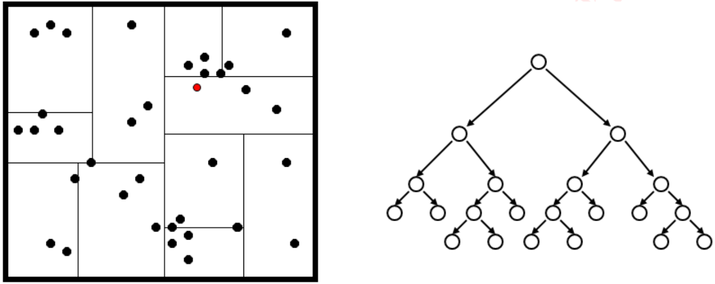
KD-tree data structure

{1,5},{3,6}, {2,9}, {4,8}, {5,3}, {7,1}, {6,2}, {8,4}, {9,5}

Sorted with x-value and recorded root/median

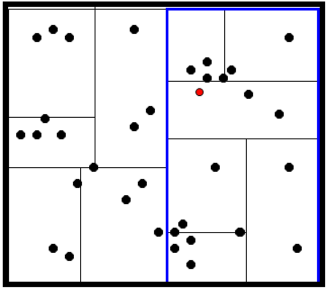


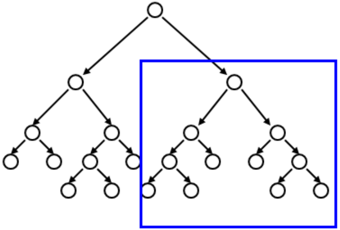
KD-tree data structure for kNN search



We traverse the tree looking for the nearest neighbor of the query point.

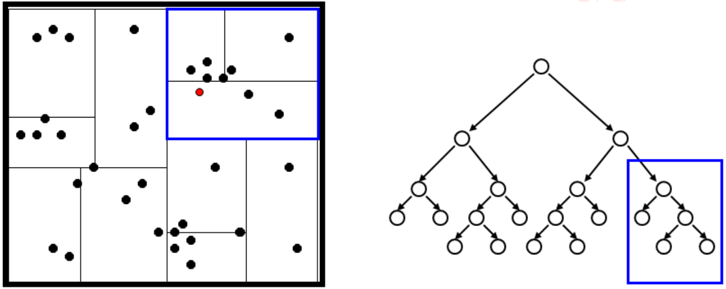
KD-tree data structure for kNN search





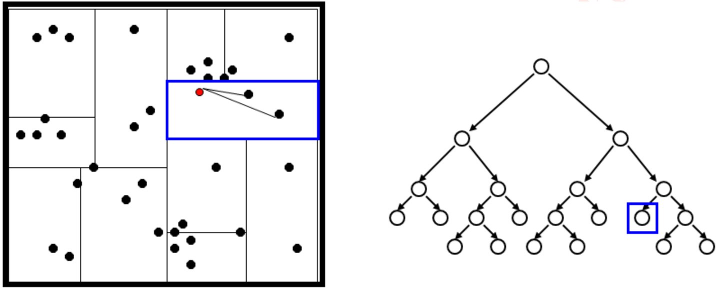
Examine nearby points first: Explore the branch of the tree that is closest to the query point first.

KD-tree data structure for kNN search



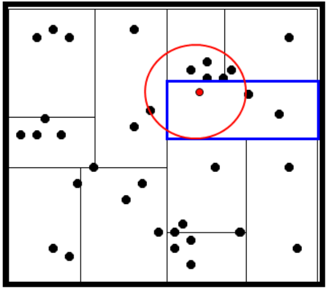
<u>Examine nearby points first</u>: Explore the branch of the tree that is closest to the query point first.

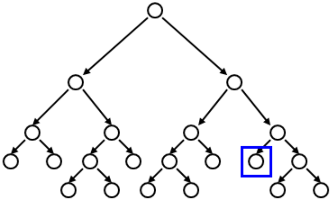
KD-tree data structure for kNN search



When we reach a leaf node: compute the distance to each point in the node.

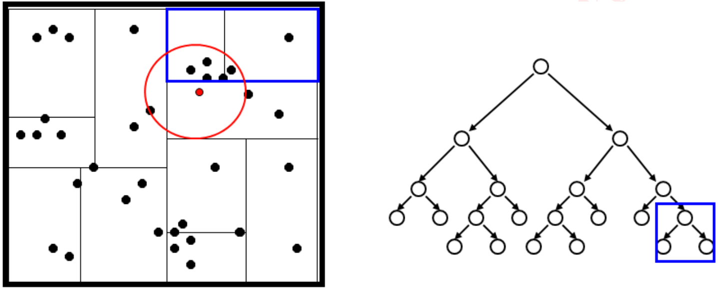
KD-tree data structure for kNN search





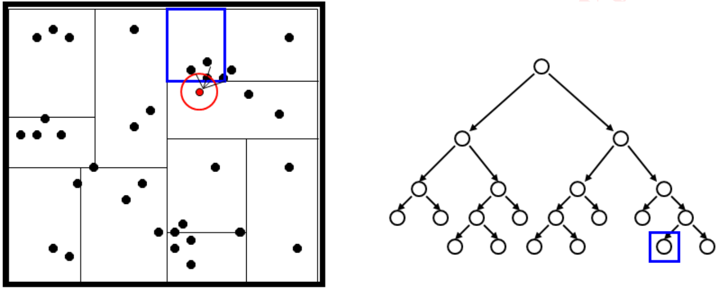
When we reach a leaf node: compute the distance to each point in the node.

KD-tree data structure for kNN search



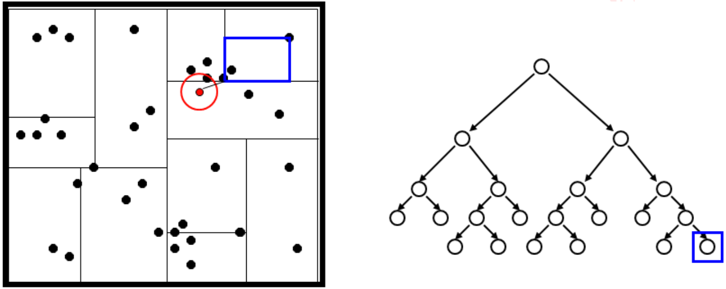
Then we can backtrack and try the other branch at each node visited.

KD-tree data structure for kNN search



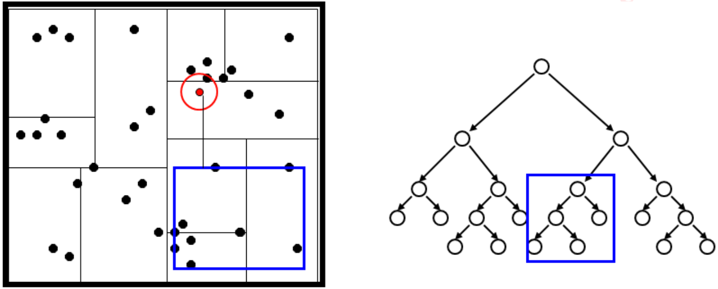
Each time a new closest node is found, we can update the distance bounds.

KD-tree data structure for kNN search



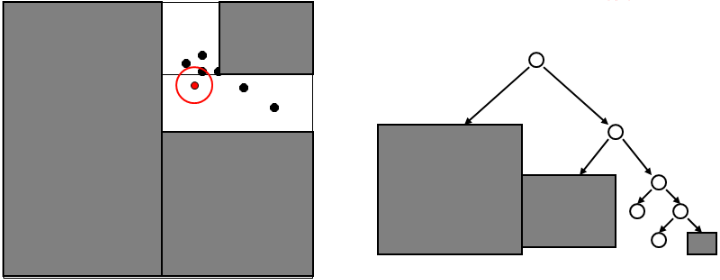
Using the distance bounds and the bounds of the data below each node, we can prune parts of the tree that could NOT include the nearest neighbor.

KD-tree data structure for kNN search



Using the distance bounds and the bounds of the data below each node, we can prune parts of the tree that could NOT include the nearest neighbor.

KD-tree data structure for kNN search



Using the distance bounds and the bounds of the data below each node, we can prune parts of the tree that could NOT include the nearest neighbor.

KD-tree data structure summary

Pros

- Exact
- Easy to build
- Popular in Computer Graphics i.e. meshes, polygons , used to find which points are close in 3D surfaces.

KD-tree data structure summary

Pros

- Exact
- Easy to build
- Popular in Computer Graphics i.e. meshes, polygons , used to find which points are close in 3D surfaces.

Cons

- Curse of dimensionality makes KD-Trees ineffective for higher number of dimensions (almost all data points on the edges far away). Will not work if data is confined to manifold which is present is high dimensional ambient space. In such cases ball trees will be useful.
- All splits are axis aligned.

KD-tree data structure summary

Pros

- Exact
- Easy to build
- Popular in Computer Graphics i.e. meshes, polygons , used to find which points are close in 3D surfaces.

Cons

- Curse of dimensionality makes KD-Trees ineffective for higher number of dimensions (almost all data points on the edges far away). Will not work if data is confined to manifold which is present is high dimensional ambient space. In such cases ball trees will be useful.
- All splits are axis aligned.

Approximation: Limit search to m leafs only

Section Contents

Exercise

Question

1. Repeat experiment with Digits dataset by varying values of k and find its optimal value.
2. What is the error bound of k-NN classifier. What happen when number of samples $n \to \infty$?

Further Reading

1. Effect of K on decision boundary i.e. $k = 1$ or $k = 3$ or $k = 7$ etc.
2. Feature transformation / reduction : Singular Value Decomposition (SVD), Principal Component Analysis (PCA)
3. Feature selection techniques i.e. statistical test and GA
4. Locally weighted regression
5. Ball-Trees

Machine Learning
Problem Setup, Understanding Dataset, Preprocessing & Model Evaluation

Dr. Rizwan Ahmed Khan

Outline

Section Contents

Reference Books

Reference books for this Module:

- Chapter 1 & 5: Pattern Recognition, Theodoridis et al., Academic Press, 4^{th} Edition or latest edition.

Reference Books

Reference books for this Module:

- Chapter 1 & 5: Pattern Recognition, Theodoridis et al., Academic Press, 4^{th} Edition or latest edition.
- Chapter 1: Combining Pattern Classifiers: Methods and Algorithms, Ludmila I. Kuncheva, Wiley-Interscience.

Reference Books

Reference books for this Module:

- Chapter 1 & 5: Pattern Recognition, Theodoridis et al., Academic Press, 4^{th} Edition or latest edition.
- Chapter 1: Combining Pattern Classifiers: Methods and Algorithms, Ludmila I. Kuncheva, Wiley-Interscience.
- Chapter 2: Data Mining , Practical Machine Learning Tools & Techniques, Witten and Franck, Elsevier Books, 2^{nd} Edition.

Reference Books

Reference books for this Module:

- Chapter 1 & 5: Pattern Recognition, Theodoridis et al., Academic Press, 4^{th} Edition or latest edition.
- Chapter 1: Combining Pattern Classifiers: Methods and Algorithms, Ludmila I. Kuncheva, Wiley-Interscience.
- Chapter 2: Data Mining , Practical Machine Learning Tools & Techniques, Witten and Franck, Elsevier Books, 2^{nd} Edition.
- Chapter 2: Data Mining & Analysis : Fundamental Concepts & Algorithms, Zaki and Meira, Cambridge University Press 2014.

Section Contents

Basic Terminology

- Algorithm: An Algorithm is a set of rules that a machine follows to achieve a particular goal. An algorithm can be considered as a recipe that defines the inputs, the output and all the steps needed to get from the inputs to the output.

for $j = 1$ to N **do**
  detect color ($image_N$)
  lots of code
end for

Basic Terminology

- Algorithm: An Algorithm is a set of rules that a machine follows to achieve a particular goal. An algorithm can be considered as a recipe that defines the inputs, the output and all the steps needed to get from the inputs to the output.

```
for j = 1 to N do
    detect color (image_N)
    lots of code
end for
```

- Learner: A Learner or Machine Learning Algorithm is the program used to learn a machine learning model from data. Another name is "inducer" (e.g. "tree inducer: is a program which builds the decision tree from data").

Basic Terminology

- Classification: Classification is the process of predicting the class of given data points. Classes are sometimes called as targets / labels or categories.

Basic Terminology

- Classification: Classification is the process of predicting the class of given data points. Classes are sometimes called as targets / labels or categories.
- Target function: The target function $f : X \to Y$ is the function f that we want to model. It maps data points to targets / labels.

Basic Terminology

- Classification: Classification is the process of predicting the class of given data points. Classes are sometimes called as targets / labels or categories.
- Target function: The target function $f : X \to Y$ is the function f that we want to model. It maps data points to targets / labels.
- Machine Learning Model / Classifier / Hypothesis: A Machine Learning Model is the learned program / function that maps inputs to outputs / predictions. For example: decision tree is a classifier or this can be a set of weights for a linear model or for a neural network.

Problem Setting:

- Set of possible instances X i.e. $\{< x_i, y_i >\}$
- Unknown target function $f : X \to Y$
- Set of function hypotheses $H = \{h | h : X \to Y\}$

Problem Formalization

Problem formalization

- Set of possible instances X i.e. $\{< \vec{x}_i, y_i >\}$
- Dataset D, given by $D = \{< \vec{x}_i, y_i >, \ldots, < \vec{x}_n, y_n >\} \subseteq X \times Y$
  Where:
  $\vec{x}_i$ is a feature vector ($\mathbb{R}^d$),
  y_i is a label / target variable,
  X is space of all features and
  Y is space of labels.
- Unknown target function $f : X \rightarrow Y$
- Set of function hypotheses $H = \{h | h : X \rightarrow Y\}$

Output:

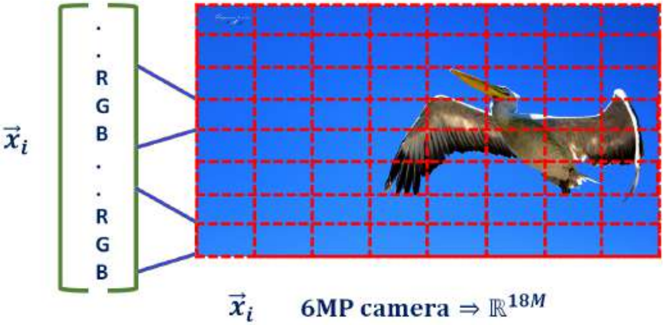
- Hypothesis $h \in H$ that best approximates target function f. Or a classification "rule" that can determine the class of any object from its attributes values.
- If training is done correctly $h(\vec{x}_i) \approx y_i$

Problem Formalization

Examples of Label Space Y

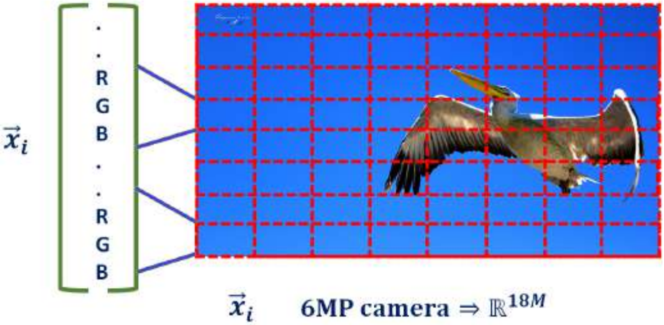
- Binary classification
  Y={0,1}
  Y={-1,+1}
- Multi-class classification
  Y=$\{1, 2, \cdots, K\}$
  where $(K > 2)$
- Regression
  Y=$\mathbb{R}$

Example of Feature vector : image features



- This is actually not a good representation and before deep learning / CNN, raw pixel values were not used for learning concepts (feature extraction).

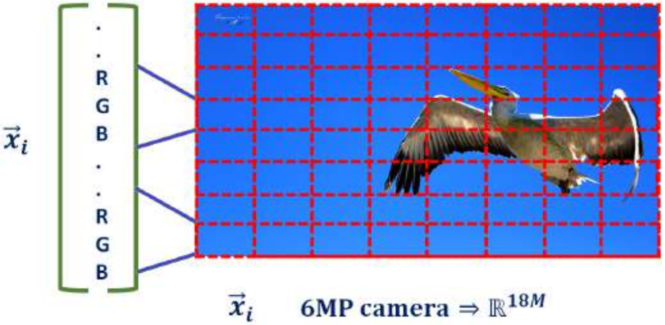
Example of Feature vector : image features



- This is actually not a good representation and before deep learning / CNN, raw pixel values were not used for learning concepts (feature extraction).

- Word document : Sparse representation!

Example of Feature vector : image features



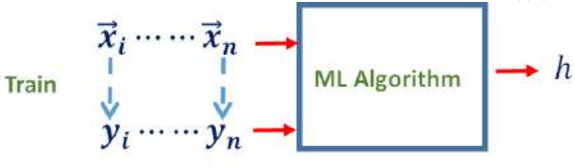
- This is actually not a good representation and before deep learning / CNN, raw pixel values were not used for learning concepts (feature extraction).

- Word document : Sparse representation!

Some common / traditional image feature extractors:

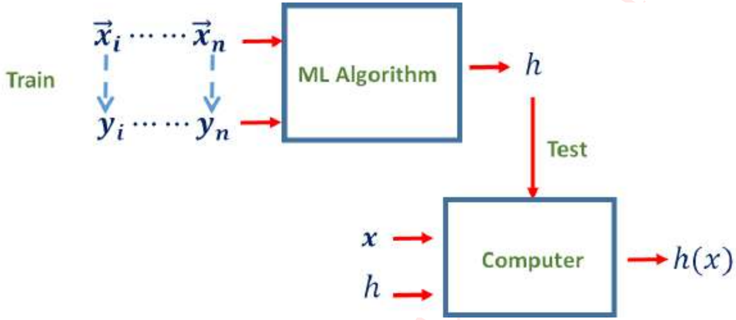
- Scale-Invariant Feature Transform (SIFT)
- Speeded-Up Robust Features (SURF)
- Local Binary Pattern (LBP)
- GIST extractor
- Histogram of Oriented Gradients (HoG)
- ....

Test / Train setup



- Aim is that algo. should learn to map $\vec{x}_i \rightarrow y_i$
- If training is done correctly $h(\vec{x}_i) \approx y_i$

Test / Train setup



- For test, take $\vec{x}$ whose label is unknown.
- Then computer passes that $\vec{x}$ to h to make prediction on unknown data.

Important

It will only work if train and test data are drawn from the same distribution.

Hypothesis

- Hypothesis $h \in H$ that best approximates target function f.
- Before we can find a function h from infinite many possibilities H , we must specify what type of function it is that we are looking for. It could be:
  1. Decision Tree
  2. Nearest Neighbor
  3. SVM
  4. ANN
  5. Bayesian classifier
  6. ..
- There is NO best algorithm. It all depends on the problem and on the data.

Hypothesis class selection

- How to select $h \in H$?

Hypothesis class selection

- How to select $h \in H$?

Some random ideas:

Hypothesis class selection

- How to select $h \in H$?

Some random ideas:

1. Pick $h \in H$ randomly (and hope it works!).

Hypothesis class selection

- How to select $h \in H$?

Some random ideas:

1. Pick $h \in H$ randomly (and hope it works!).
   - Problem: Space H (the set of functions that can possibly be learned) is very large and it is very unlikely that randomly picked function would work.
   - Any corner case, where it might work?

Hypothesis class selection

- How to select $h \in H$?

Some random ideas:

1. Pick $h \in H$ randomly (and hope it works!).
   - Problem: Space H (the set of functions that can possibly be learned) is very large and it is very unlikely that randomly picked function would work.
   - Any corner case, where it might work?
   - It may work, only if, H is restricted enough (set of function that will work)

Hypothesis class selection

- How to select $h \in H$?

Some random ideas:

1. Pick $h \in H$ randomly (and hope it works!).
   - Problem: Space H (the set of functions that can possibly be learned) is very large and it is very unlikely that randomly picked function would work.
   - Any corner case, where it might work?
   - It may work, only if, H is restricted enough (set of function that will work)
2. Traverse all the h in hypothesis class H and chose the one that works best i.e. least error.

Hypothesis class selection

- How to select $h \in H$?

Some random ideas:

1. Pick $h \in H$ randomly (and hope it works!).
   - Problem: Space H (the set of functions that can possibly be learned) is very large and it is very unlikely that randomly picked function would work.
   - Any corner case, where it might work?
   - It may work, only if, H is restricted enough (set of function that will work)
2. Traverse all the h in hypothesis class H and chose the one that works best i.e. least error.
   - Problem: Again space H is very large.

Hypothesis class selection

- How to select $h \in H$?

Some random ideas:

1. Pick $h \in H$ randomly (and hope it works!).
   - Problem: Space H (the set of functions that can possibly be learned) is very large and it is very unlikely that randomly picked function would work.
   - Any corner case, where it might work?
   - It may work, only if, H is restricted enough (set of function that will work)
2. Traverse all the h in hypothesis class H and chose the one that works best i.e. least error.
   - Problem: Again space H is very large.

$h \in H$

Essentially, we try to find a function h within the hypothesis class that makes the fewest mistakes within training data.

Objective of Machine Learning

- The purpose of machine learning is to discover patterns in the data and then make predictions on test set based on experience / data. Thus, selected function h within the hypothesis class H, should minimize error on unseen future examples (prediction). But before making prediction, function h is selected based on lowest error on the training set.
- To find $h \in H$ that makes least errors on training data loss functions are used.
- The higher the loss, the worse it is - a loss of zero means it makes no errors.

Loss Functions or Objective Functions

1. Zero-One Loss: The simplest loss function is the zero-one loss. It literally counts how many mistakes an hypothesis function h makes on the training set.

Loss Functions or Objective Functions

1. Zero-One Loss: The simplest loss function is the zero-one loss. It literally counts how many mistakes an hypothesis function h makes on the training set.

$$\mathcal{L}_{0/1}(h) = \frac{1}{n}\sum_{i=1}^{n}\delta_{h(\mathbf{x}_i)\neq y_i}, \text{ where } \delta_{h(\mathbf{x}_i)\neq y_i} = \begin{cases} 1, & \text{if } h(\mathbf{x}_i) \neq y_i \\ 0, & \text{Otherwise} \end{cases} \quad (1)$$

Loss Functions or Objective Functions

1. Zero-One Loss: The simplest loss function is the zero-one loss. It literally counts how many mistakes an hypothesis function h makes on the training set.

$$\mathcal{L}_{0/1}(h) = \frac{1}{n}\sum_{i=1}^{n} \delta_{h(\mathbf{x}_i)\neq y_i}, \text{ where } \delta_{h(\mathbf{x}_i)\neq y_i} = \begin{cases} 1, & \text{if } h(\mathbf{x}_i) \neq y_i \\ 0, & \text{Otherwise} \end{cases} \quad (1)$$

- Is this loss function fine for regression settings?

Loss Functions or Objective Functions

1. Zero-One Loss: The simplest loss function is the zero-one loss. It literally counts how many mistakes an hypothesis function h makes on the training set.

$$\mathcal{L}_{0/1}(h) = \frac{1}{n}\sum_{i=1}^{n}\delta_{h(\mathbf{x}_i)\neq y_i}, \text{ where } \delta_{h(\mathbf{x}_i)\neq y_i} = \begin{cases} 1, & \text{if } h(\mathbf{x}_i) \neq y_i \\ 0, & \text{Otherwise} \end{cases} \quad (1)$$

- Is this loss function fine for regression settings?

2. Squared loss: The squared loss function is typically used in regression settings.

Loss Functions or Objective Functions

1. Zero-One Loss: The simplest loss function is the zero-one loss. It literally counts how many mistakes an hypothesis function h makes on the training set.

$$\mathcal{L}_{0/1}(h) = \frac{1}{n}\sum_{i=1}^{n}\delta_{h(\mathbf{x}_i)\neq y_i}, \text{ where } \delta_{h(\mathbf{x}_i)\neq y_i} = \begin{cases} 1, & \text{if } h(\mathbf{x}_i) \neq y_i \\ 0, & \text{Otherwise} \end{cases} \quad (1)$$

- Is this loss function fine for regression settings?

2. Squared loss: The squared loss function is typically used in regression settings.

$$\mathcal{L}_{sq}(h) = \frac{1}{n}\sum_{i=1}^{n}(h(\mathbf{x}_i) - y_i)^2 \quad (2)$$

Loss Functions or Objective Functions

1. Zero-One Loss: The simplest loss function is the zero-one loss. It literally counts how many mistakes an hypothesis function h makes on the training set.

$$\mathcal{L}_{0/1}(h) = \frac{1}{n}\sum_{i=1}^{n}\delta_{h(\mathbf{x}_i)\neq y_i}, \text{ where } \delta_{h(\mathbf{x}_i)\neq y_i} = \begin{cases} 1, & \text{if } h(\mathbf{x}_i) \neq y_i \\ 0, & \text{Otherwise} \end{cases} \tag{1}$$

- Is this loss function fine for regression settings?

2. Squared loss: The squared loss function is typically used in regression settings.

$$\mathcal{L}_{sq}(h) = \frac{1}{n}\sum_{i=1}^{n}(h(\mathbf{x}_i) - y_i)^2 \tag{2}$$

- The loss suffered grows quadratically with the absolute mis-predicted amount. This property encourages no predictions to be really far off (or the penalty would be so large that a different hypothesis function is likely better suited). Penalty of one example off by 10 is much higher than penalty of ten examples off by 1.

Loss Functions or Objective Functions

3. Absolute loss: is also typically used in regression settings. Loss grows linearly (as opposed to squared loss) with mis-predictions, thus it is more suitable for noisy data.

$$\mathcal{L}_{abs}(h) = \frac{1}{n}\sum_{i=1}^{n}|h(\mathbf{x}_i) - y_i| \quad (3)$$

Hypothesis class selection

- How to select $h \in H$?

Some random ideas:

Hypothesis class selection

- How to select $h \in H$?

Some random ideas:

3. If you find a function $h(\cdot)$ (i.e. memorizer*) with low loss on your data D, how do you know whether it will still get examples right that are not in D?

   memorizer*

$$h(x) = \begin{cases} y_i, & \text{if } \exists(\mathbf{x}_i, y_i) \in D, \text{ s.t., } \mathbf{x} = \mathbf{x}_i, \\ 0, & \text{other wise} \end{cases}$$

Hypothesis class selection

- How to select $h \in H$?

Some random ideas:

3. If you find a function $h(\cdot)$ (i.e. memorizer*) with low loss on your data D, how do you know whether it will still get examples right that are not in D?

   memorizer*

$$h(x) = \begin{cases} y_i, & \text{if } \exists(\mathbf{x}_i, y_i) \in D, \text{ s.t., } \mathbf{x} = \mathbf{x}_i, \\ 0, & \text{other wise} \end{cases}$$

It has ZERO training error.

Hypothesis class selection

- How to select $h \in H$?

Some random ideas:

3. If you find a function $h(\cdot)$ (i.e. memorizer*) with low loss on your data D, how do you know whether it will still get examples right that are not in D?

memorizer*

$$h(x) = \begin{cases} y_i, & \text{if } \exists(\mathbf{x}_i, y_i) \in D, \text{ s.t., } \mathbf{x} = \mathbf{x}_i, \\ 0, & \text{other wise} \end{cases}$$

What is the issue with this algorithm?

Hypothesis class selection

- How to select $h \in H$?

Some random ideas:

3. If you find a function $h(\cdot)$ (i.e. memorizer*) with low loss on your data D, how do you know whether it will still get examples right that are not in D?
   memorizer*

$$h(x) = \begin{cases} y_i, & \text{if } \exists(\mathbf{x}_i, y_i) \in D, \text{ s.t., } \mathbf{x} = \mathbf{x}_i, \\ 0, & \text{other wise} \end{cases}$$

It will perform horribly with samples not in D, i.e., there's the over-fitting issue with this function.

Hypothesis class selection : Generalization

- We have the metric to measure loss on training set. How about Generalization?

Hypothesis class selection : Generalization

- We have the metric to measure loss on training set. How about Generalization?
- What actually is required?

Hypothesis class selection : Generalization

- We have the metric to measure loss on training set. How about Generalization?
- What actually is required?
- We want min error for
  $\forall(\vec{x}, y) \sim P$ Where $(\vec{x}, y)$ are new data points drawn from distribution P and P is not known , although distribution D is drawn from P.

Or Minimize Expected Loss

Hypothesis class selection : Generalization

- We have the metric to measure loss on training set. How about Generalization?
- What actually is required?
- We want min error for
  $\forall(\vec{x}, y) \sim P$ Where $(\vec{x}, y)$ are new data points drawn from distribution P and P is not known , although distribution D is drawn from P.

Or Minimize Expected Loss

$$E\left[\mathcal{L}(h_i(\vec{x}, y))\right]_{(\vec{x},y)\sim P} \tag{4}$$

Hypothesis class selection : Generalization

- We have the metric to measure loss on training set. How about Generalization?
- What actually is required?
- We want min error for $\forall(\vec{x}, y) \sim P$ Where $(\vec{x}, y)$ are new data points drawn from distribution P and P is not known , although distribution D is drawn from P.

Or Minimize Expected Loss

$$E\left[\mathcal{L}(h_i(\vec{x}, y))\right]_{(\vec{x},y)\sim P} \tag{4}$$

Generalization

- We don't have distribution P so we can't compute loss for it. Good thing is we can approximate it.
- In ML usually dataset is divided in three parts train, validation and test to measure generalization capabilities (more on this later in the lecture).

Summary

1. We train our classifier by minimizing the training loss:

$$\text{Learning: } h^*(\cdot) = \text{argmin}_{h(\cdot)\in\mathcal{H}} \frac{1}{|D_{\text{TR}}|} \sum_{(\mathbf{x},y)\in D_{\text{TR}}} \ell(\mathbf{x}, y|h(\cdot))$$

where $\mathcal{H}$ is the set of all possible classifiers $h(.)$. In other words, we are trying to find a hypothesis h which would have performed well on the training data.

Summary

1. We train our classifier by minimizing the training loss:

$$\text{Learning: } h^*(\cdot) = \text{argmin}_{h(\cdot)\in\mathcal{H}} \frac{1}{|D_{\text{TR}}|} \sum_{(\mathbf{x},y)\in D_{\text{TR}}} \ell(\mathbf{x}, y|h(\cdot))$$

   where $\mathcal{H}$ is the set of all possible classifiers $h(.)$. In other words, we are trying to find a hypothesis h which would have performed well on the training data.

2. We evaluate our classifier on the test data to calculate testing loss:

$$\text{Evaluation: } \epsilon_{\text{TE}} = \frac{1}{|D_{TE}|} \sum_{(\mathbf{x},y)\in D_{\text{TE}}} \ell(\mathbf{x}, y|h^*(\cdot))$$

Summary

1. We train our classifier by minimizing the training loss:

$$\text{Learning: } h^*(\cdot) = \text{argmin}_{h(\cdot)\in\mathcal{H}} \frac{1}{|D_{\text{TR}}|} \sum_{(\mathbf{x},y)\in D_{\text{TR}}} \ell(\mathbf{x}, y|h(\cdot))$$

   where $\mathcal{H}$ is the set of all possible classifiers $h(.)$. In other words, we are trying to find a hypothesis h which would have performed well on the training data.

2. We evaluate our classifier on the test data to calculate testing loss:

$$\text{Evaluation: } \epsilon_{\text{TE}} = \frac{1}{|D_{TE}|} \sum_{(\mathbf{x},y)\in D_{\text{TE}}} \ell(\mathbf{x}, y|h^*(\cdot))$$

3. If the samples are drawn independent and identically distributed from the distribution $\mathcal{P}$, then the testing loss is an unbiased estimator of the true generalization loss:

$$\text{Generalization: } \epsilon = \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{P}}[\ell(\mathbf{x}, y|h^*(\cdot))]$$

Summary

Quiz

Why does $\epsilon_{\text{TE}} \to \epsilon$ as $|D_{\text{TE}}| \to +\infty$?
or
Why Test error ϵ_{TE} becomes same as generalization error ϵ when test set is really large $n \to +\infty$.

Summary

Quiz

Why does $\epsilon_{\text{TE}} \rightarrow \epsilon$ as $|D_{\text{TE}}| \rightarrow +\infty$?
or
Why Test error ϵ_{TE} becomes same as generalization error ϵ when test set is really large $n \rightarrow +\infty$.

Read

Weak law of large numbers : the empirical average of data drawn from a distribution converges to its mean

Section Contents

Understanding Dataset

- A Dataset / Training set / database: is set of data containing features and the target to predict.

Understanding Dataset

- A Dataset / Training set / database: is set of data containing features and the target to predict.
- An Instance: is a row in the dataset. Other names for instance are: (data) point, example, observation. An instance consists of the feature values.

Understanding Dataset

- A Dataset / Training set / database: is set of data containing features and the target to predict.
- An Instance: is a row in the dataset. Other names for instance are: (data) point, example, observation. An instance consists of the feature values.
- The Features / Attributes: are the inputs used for prediction or classification. A feature is a column in the dataset. A feature is an individual measurable property or characteristic of a phenomenon being observed. Choosing informative, discriminating and independent features is a crucial step for effective algorithms in Machine Learning.

Toy Dataset

Training Data / Features extracted from real data

Weight	**Texture**	**Class**
150g	Bumpy	Orange
170g	Bumpy	Orange
140g	Smooth	Apple
130g	Smooth	Apple
..	..	..
..	..	..

Toy Dataset

Training Data / Features extracted from real data

Weight	**Texture**	**Class**
150g	Bumpy	Orange
170g	Bumpy	Orange
140g	Smooth	Apple
130g	Smooth	Apple
..	..	..
..	..	..

1. Each row in training data is an example (Feature extractor algorithm).

Toy Dataset

Training Data / Features extracted from real data

Weight	Texture	Class
150g	Bumpy	Orange
170g	Bumpy	Orange
140g	Smooth	Apple
130g	Smooth	Apple
..	..	..
..	..	..

1. Each row in training data is an example (Feature extractor algorithm).
2. Last column is class / label / target.

Understanding Dataset

Weight	Texture	Class
150g	Bumpy	Orange
170g	Bumpy	Orange
140g	Smooth	Apple
130g	Smooth	Apple
..	..	..
..	..	..

Toy Dataset.

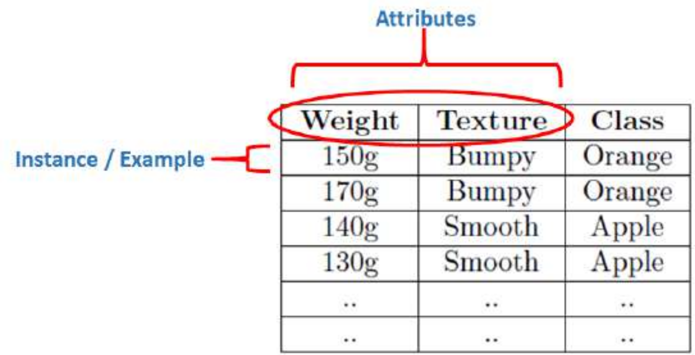
Understanding Dataset

Instance / Example →

Weight	Texture	Class
150g	Bumpy	Orange
170g	Bumpy	Orange
140g	Smooth	Apple
130g	Smooth	Apple
..	..	..
..	..	..

An Instance is a row in the dataset. It is also called as Obervation , Example.

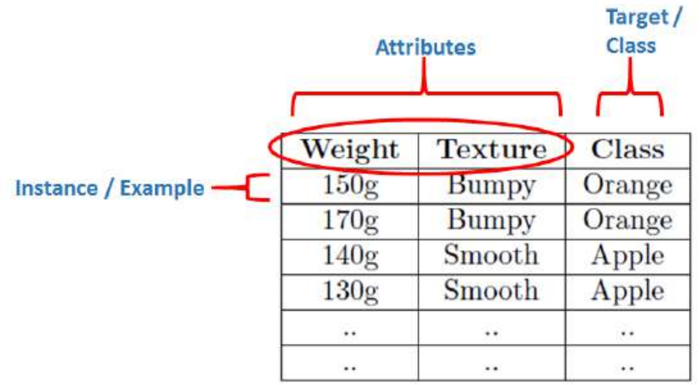
Understanding Dataset



Weight	Texture	Class
150g	Bumpy	Orange
170g	Bumpy	Orange
140g	Smooth	Apple
130g	Smooth	Apple
..	..	..
..	..	..

The Features / Attributes: are the inputs used for prediction or classification.

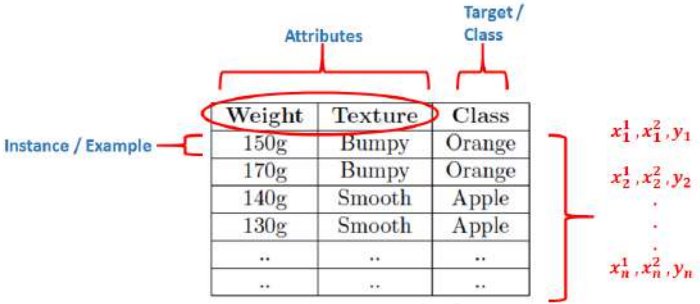
Understanding Dataset



Weight	Texture	Class
150g	Bumpy	Orange
170g	Bumpy	Orange
140g	Smooth	Apple
130g	Smooth	Apple
..	..	..
..	..	..

Target / class is the information the machine learns to predict.

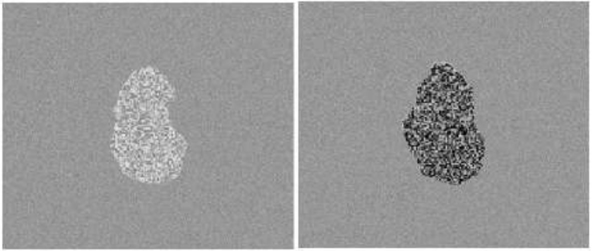
Understanding Dataset



Complete dataset consists of features and class variables.
One instance is represented as $< {x_1}^1, {x_1}^2, \cdots, {x_1}^n, y_1 >$ or $< \vec{x}_1, y_1 >$ where $\vec{x}_1 \in \mathbb{R}^n$

From Image to Data Point

- To understand all previously described terms, have a look at this example[1] from Medical Image Classification.



- Two images, each having a distinct region inside it.
  - First image from a benign lesion
  - Second image from malignant one (cancer)

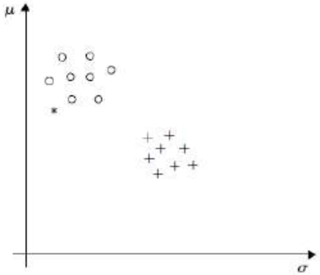
[1] Image from Theodoridis book

From Image to Data Point

- The first step is to identify the measurable quantities or features that make these two regions distinct from each other (problem of feature identification / engineering).

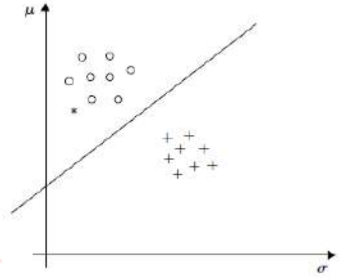
From Image to Data Point

- The first step is to identify the measurable quantities or features that make these two regions distinct from each other (problem of feature identification / engineering).
- Figure below shows a plot of the mean value of the intensity in each region of interest versus the corresponding standard deviation around this mean.
- Each point corresponds to a different image from the available database.



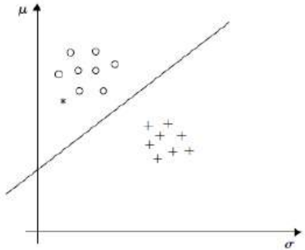
From Image to Data Point

- The first step is to identify the measurable quantities or features that make these two regions distinct from each other (problem of feature identification / engineering).
- Figure below shows a plot of the mean value of the intensity in each region of interest versus the corresponding standard deviation around this mean.
- Each point corresponds to a different image from the available database.



From Image to Data Point

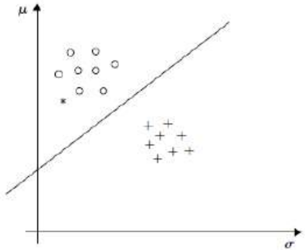
- Assume that we are given a new image (shown as *). Algorithm will again calculate same features i.e. mean intensity and standard deviation in the region of interest to classify new data point.



- The measurements used for the classification, the mean value and the standard deviation in this case, are known as features.

From Image to Data Point

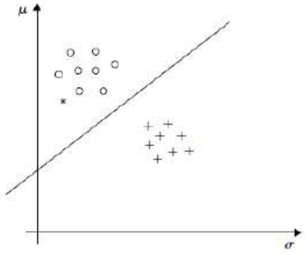
- Assume that we are given a new image (shown as *). Algorithm will again calculate same features i.e. mean intensity and standard deviation in the region of interest to classify new data point.



- The measurements used for the classification, the mean value and the standard deviation in this case, are known as features.
- Feature is an individual measurable property or characteristic of a phenomenon being observed [a].

From Image to Data Point

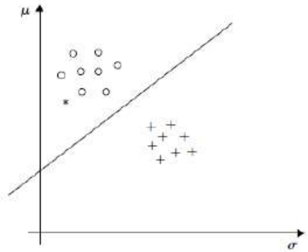
- Assume that we are given a new image (shown as *). Algorithm will again calculate same features i.e. mean intensity and standard deviation in the region of interest to classify new data point.



- The measurements used for the classification, the mean value and the standard deviation in this case, are known as features.
- Feature is an individual measurable property or characteristic of a phenomenon being observed [a].
- Generally, n features are used to describe one observation : $< {x_i}^1, {x_i}^2, \cdots, {x_i}^n > \in \mathbb{R}^n$. This is also called feature vector.

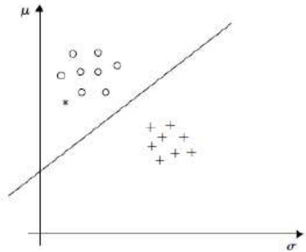
[a] Bishop, Christopher (2006). Pattern recognition and machine learning

From Image to Data Point



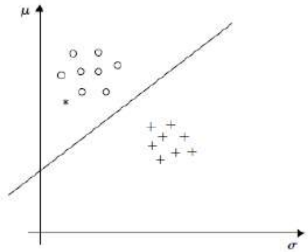
- Each of the feature vectors ($< {x_i}^1, {x_i}^2, \cdots, {x_i}^n > \in \mathbb{R}^n$) identifies uniquely a single pattern (object / observation / example).

From Image to Data Point



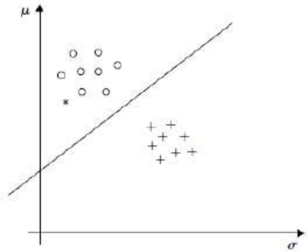
- Each of the feature vectors ($< {x_i}^1, {x_i}^2, \cdots, {x_i}^n > \in \mathbb{R}^n$) identifies uniquely a single pattern (object / observation / example).
- The straight line in Figure is known as the decision line, and it constitutes the classifier whose role is to divide the feature space into regions that correspond to either class A or class B.

From Image to Data Point



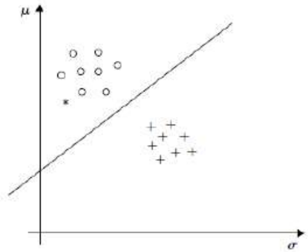
- Each of the feature vectors ($< {x_i}^1, {x_i}^2, \cdots, {x_i}^n > \in \mathbb{R}^n$) identifies uniquely a single pattern (object / observation / example).
- The straight line in Figure is known as the decision line, and it constitutes the classifier whose role is to divide the feature space into regions that correspond to either class A or class B.
- If decision line fails to correctly classify example, a misclassification has occurred.

From Image to Data Point



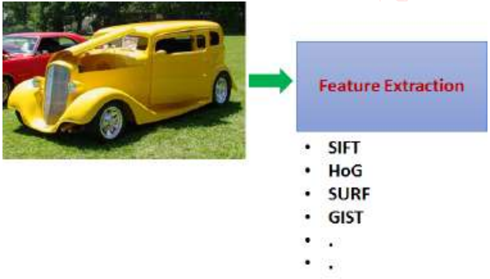
- Each of the feature vectors $(< {x_i}^1, {x_i}^2, \cdots, {x_i}^n > \in \mathbb{R}^n)$ identifies uniquely a single pattern (object / observation / example).
- The straight line in Figure is known as the decision line, and it constitutes the classifier whose role is to divide the feature space into regions that correspond to either class A or class B.
- If decision line fails to correctly classify example, a misclassification has occurred.
- The feature vectors whose true class $< y_i >$ is known (supervised learning) and which are used for the design / training of the classifier are known as training feature vectors in broad sense.

From Image to Data Point



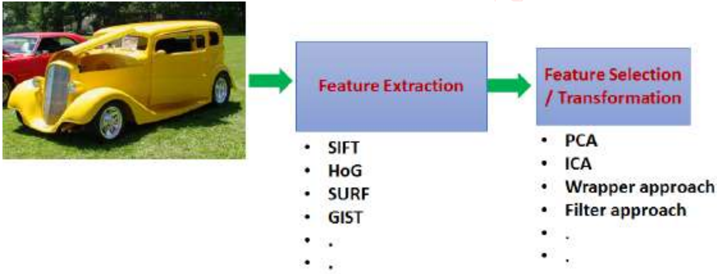
- Each of the feature vectors ($< {x_i}^1, {x_i}^2, \cdots, {x_i}^n > \in \mathbb{R}^n$) identifies uniquely a single pattern (object / observation / example).
- The straight line in Figure is known as the decision line, and it constitutes the classifier whose role is to divide the feature space into regions that correspond to either class A or class B.
- If decision line fails to correctly classify example, a misclassification has occurred.
- The feature vectors whose true class $< y_i >$ is known (supervised learning) and which are used for the design / training of the classifier are known as training feature vectors in broad sense.
- Training feature vectors are further divided into train, validation and test set.

Basic Questions: Classification Task



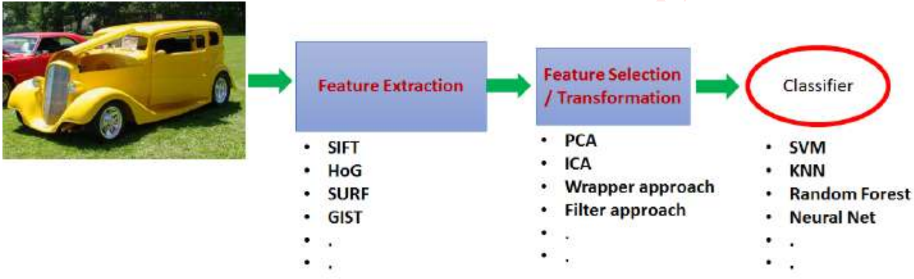
1. How are the features generated? It is not trivial to know which feature will have discriminative ability. It is problem dependent, and it concerns the feature generation / extraction / engineering stage of the design of a classification system. In image above few feature extraction algorithms are given, each of which transform image data to n-dimensional feature vector.

Basic Questions: Classification Task



1. How are the features generated? It is not trivial to know which feature will have discriminative ability. It is problem dependent, and it concerns the feature generation / extraction / engineering stage of the design of a classification system. In image above few feature extraction algorithms are given, each of which transform image data to n-dimensional feature vector.
2. What are the best n number of features to use? This is also a very important task and it concerns the feature transformation / selection / preprocessing stage of the classification system.

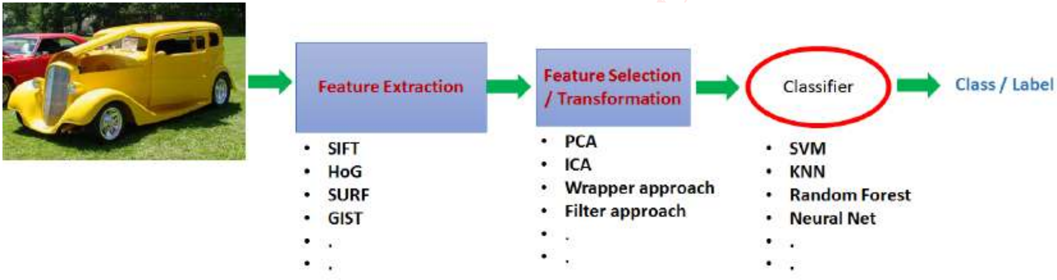
Basic Questions: Classification Task



3. How does one design the classifier? Is linear classifier a good choice (like the one in previous example). These questions concern the classifier design stage.

[2] Question 2-4 will be discussed

Basic Questions: Classification Task



3. How does one design the classifier? Is linear classifier a good choice (like the one in previous example). These questions concern the classifier design stage.
4. Finally, how can one assess the performance of the designed classifier? That is, what is the classification error rate? This is the task of the system / model evaluation stage.

Note [2]

[2] Question 2-4 will be discussed

Features quality

Feature is an individual measurable property or characteristic of a phenomenon being observed.

Fundamental question

What are good features?

Features quality

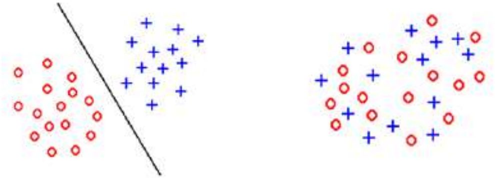
Feature is an individual measurable property or characteristic of a phenomenon being observed.

Fundamental question

What are good features?

Good feature

Good features makes it easy for classifier to decide (learn) between two different classes / concepts / labels OR good features enhances inter class variations while minimize intra class varaition.



"Good" features "Bad" features

Feature Types

Mainly feature variable can have two distinct types:

Weight	Texture	Class
150g	Bumpy	Orange
170g	Bumpy	Orange
140g	Smooth	Apple
130g	Smooth	Apple
..	..	..
..	..	..

1. Numerical variable / feature :
   Numerical data is a type of data that is expressed in terms of numbers rather than natural language descriptions

Feature Types

Mainly feature variable can have two distinct types:

Weight	Texture	Class
150g	Bumpy	Orange
170g	Bumpy	Orange
140g	Smooth	Apple
130g	Smooth	Apple
..	..	..
..	..	..

1. Numerical variable / feature :
   Numerical data is a type of data that is expressed in terms of numbers rather than natural language descriptions
2. Categorical variable / feature :
   Categorical data is a type of data that can be stored into groups or categories

Feature Types

1. Numerical variable / feature
   1. Continuous: Observations can take any value between a certain set of real numbers.

Feature Types

1. Numerical variable / feature
   1. Continuous: Observations can take any value between a certain set of real numbers.
   2. Discrete: Observations can take a value based on a count from a set of distinct whole values. A discrete variable cannot take the value of a fraction between one value and the next closest value.

Feature Types

1. Numerical variable / feature
   1. Continuous: Observations can take any value between a certain set of real numbers.
   2. Discrete: Observations can take a value based on a count from a set of distinct whole values. A discrete variable cannot take the value of a fraction between one value and the next closest value.
2. Categorical variable / feature
   1. Ordinal: Observations can take a value that can be logically ordered or ranked. The categories associated with ordinal variables can be ranked higher or lower than another, but do not necessarily establish a numeric difference between each category e.g. short, tall.

Feature Types

1. Numerical variable / feature
   1. Continuous: Observations can take any value between a certain set of real numbers.
   2. Discrete: Observations can take a value based on a count from a set of distinct whole values. A discrete variable cannot take the value of a fraction between one value and the next closest value.
2. Categorical variable / feature
   1. Ordinal: Observations can take a value that can be logically ordered or ranked. The categories associated with ordinal variables can be ranked higher or lower than another, but do not necessarily establish a numeric difference between each category e.g. short, tall.
   2. Nominal: Observations can take a value that is not able to be organized in a logical sequence e.g. the name or colour of an object. A nominal variable may be numerical in form, but the numerical values have no mathematical interpretation. E.g. label 10 people as numbers $1, 2, 3, ..., 10$, but any arithmetic with such values, e.g. $1 + 2 = 3$ would be meaningless.

Section Contents

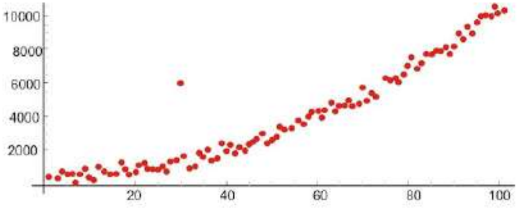
What is Preprocessing?

Preprocessing

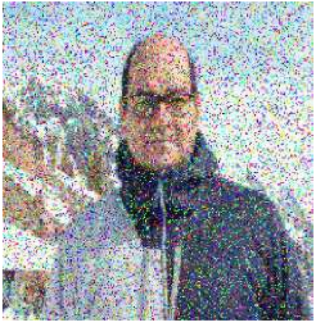
- Pre-processing refers to the transformations applied to our data before feeding it to the algorithm. It converts the raw data into a clean data set (improved interpretability), suitable for machine learning.
- Data preprocessing is an integral step in Machine Learning as the quality of data and the useful information that can be derived from it directly affects the ability of model to learn.

Why Preprocess Data?

- It helps in removing redundant information / Outliers (a point that lies very far from the mean of the corresponding random variable).



Why Preprocess Data?



- It helps in removing redundant information / Outliers (a point that lies very far from the mean of the corresponding random variable).
- Noise removal to improve performance. Data may come from some “sensors”e.g. physical devices, instruments, software programs such as web crawlers, manual surveys, etc which are prone to malfunction. Secondly, there could be human error in recoding data as well.

Why Preprocess Data?

0	2	5.0	3.0	6.0	NaN
1	9	NaN	9.0	0.0	7.0
2	19	17.0	NaN	9.0	NaN
3	7	10.0	3.0	6.0	4.0
4	2	8.0	10.0	NaN	3.0

- It helps in removing redundant information / Outliers (a point that lies very far from the mean of the corresponding random variable).
- Noise removal to improve performance. Data may come from some "sensors"e.g. physical devices, instruments, software programs such as web crawlers, manual surveys, etc which are prone to malfunction. Secondly, there could be human error in recoding data as well.
- Some specified machine learning algorithm needs information in a specified format, for example:
  - Random Forest algorithm does not support null values.
  - Principal Component Analysis (PCA) algorithm requires data to have zero mean and unit variance.

Data Preprocessing Techniques

1. Data Scaling / Data Normalization : This technique transforming feature / data so that it fits within a specific scale, like 0–100 or 0–1. For example, standardization transforms attributes to a standard Gaussian distribution with a mean of 0 and a standard deviation of 1 (requirement for PCA).

Data Preprocessing Techniques

1. Data Scaling / Data Normalization : This technique transforming feature / data so that it fits within a specific scale, like 0–100 or 0–1. For example, standardization transforms attributes to a standard Gaussian distribution with a mean of 0 and a standard deviation of 1 (requirement for PCA).
2. Outlier Removal : Points with values very different from the mean value produce large errors during training and may have disastrous effects.

Data Preprocessing Techniques

1. Data Scaling / Data Normalization : This technique transforming feature / data so that it fits within a specific scale, like 0–100 or 0–1. For example, standardization transforms attributes to a standard Gaussian distribution with a mean of 0 and a standard deviation of 1 (requirement for PCA).
2. Outlier Removal : Points with values very different from the mean value produce large errors during training and may have disastrous effects.
3. Missing Data / Null value handling : Two ways to handle
   1. Discard feature vectors with missing values, provided large data sets and these values are rare.
   2. "Complete" the missing values by (a) zeros or (b) mean (c) defining customized function

Completing the missing values in a set of data is also known as imputation.

Feature / Data Scaling

Feature / Data Scaling - Motivation

1. Range of values of attributes / features varies widely. Thus, features with large values may have a larger influence in the cost function than features with small values, although this does not necessarily reflect their respective significance in the design of the classifier.

Feature / Data Scaling

Feature / Data Scaling - Motivation

1. Range of values of attributes / features varies widely. Thus, features with large values may have a larger influence in the cost function than features with small values, although this does not necessarily reflect their respective significance in the design of the classifier.
   - For example, many classifiers and clustering algorithm (i.e. K- nearest neighbor, K-Means) calculate the distance between two points by the Euclidean distance. Without scaling one feature (with broad range of values) will dominate this calculation.

Feature / Data Scaling

Feature / Data Scaling - Motivation

1. Range of values of attributes / features varies widely. Thus, features with large values may have a larger influence in the cost function than features with small values, although this does not necessarily reflect their respective significance in the design of the classifier.
    - For example, many classifiers and clustering algorithm (i.e. K- nearest neighbor, K-Means) calculate the distance between two points by the Euclidean distance. Without scaling one feature (with broad range of values) will dominate this calculation.
2. Secondly, scaling is applied as some algorithm i.e. gradient descent, converges much faster with feature scaling than without it.

Feature / Data Scaling

Feature / Data Scaling - Motivation

1. Range of values of attributes / features varies widely. Thus, features with large values may have a larger influence in the cost function than features with small values, although this does not necessarily reflect their respective significance in the design of the classifier.
   - For example, many classifiers and clustering algorithm (i.e. K- nearest neighbor, K-Means) calculate the distance between two points by the Euclidean distance. Without scaling one feature (with broad range of values) will dominate this calculation.
2. Secondly, scaling is applied as some algorithm i.e. gradient descent, converges much faster with feature scaling than without it.
3. In Principle Component Analysis (PCA), without scaling results will be biased towards feature that has higher range (components that maximize the variance).

Feature Scaling Methods

1. Min-max normalization: This equation scales features to the range in [0, 1].

$$x^{scaled} = \frac{x - min(x)}{max(x) - min(x)} \tag{5}$$

where x is an original value, x^{scaled} is the normalized value.

Feature Scaling Methods

1. Min-max normalization: This equation scales features to the range in [0, 1].

$$x^{scaled} = \frac{x - min(x)}{max(x) - min(x)} \tag{5}$$

where x is an original value, x^{scaled} is the normalized value.

2. Mean normalization :

$$x^{scaled} = \frac{x - \bar{x}}{max(x) - min(x)} \tag{6}$$

where $\bar{x}$ = distribution average /mean

Feature Scaling Methods

1. Min-max normalization: This equation scales features to the range in [0, 1].

$$x^{scaled} = \frac{x - min(x)}{max(x) - min(x)} \tag{5}$$

where x is an original value, x^{scaled} is the normalized value.

2. Mean normalization :

$$x^{scaled} = \frac{x - \bar{x}}{max(x) - min(x)} \tag{6}$$

where $\bar{x}$ = distribution average /mean

3. Standardization: Feature standardization makes the values of each feature in the data have zero-mean and unit-variance. This method is widely used for normalization.

$$x^{scaled} = \frac{x - \bar{x}}{\sigma} \tag{7}$$

where $\bar{x}$ = distribution average /mean and σ is standard deviation.

Feature Scaling - Example

- Consider this data presented below [3]. This data needs to be scaled as values of attributes / features are varying widely.

$\mathbf{x}_i$	Age (X_1)	Income (X_2)
$\mathbf{x}_1$	12	300
$\mathbf{x}_2$	14	500
$\mathbf{x}_3$	18	1000
$\mathbf{x}_4$	23	2000
$\mathbf{x}_5$	27	3500
$\mathbf{x}_6$	28	4000
$\mathbf{x}_7$	34	4300
$\mathbf{x}_8$	37	6000
$\mathbf{x}_9$	39	2500
$\mathbf{x}_{10}$	40	2700

- Calculate:
  Scale feature using Equation 5 :
  $x^{scaled} = \frac{x - min(x)}{max(x) - min(x)}$

[3] Data from Data mining book by Zaki & Meira

Feature Scaling - Example

x_i	Age (X_1)	Income (X_2)
x_1	12	300
x_2	14	500
x_3	18	1000
x_4	23	2000
x_5	27	3500
x_6	28	4000
x_7	34	4300
x_8	37	6000
x_9	39	2500
x_{10}	40	2700

Calculate:

Scale feature using Equation 5 :

$$x^{scaled} = \frac{x - min(x)}{max(x) - min(x)}$$

Feature Scaling - Example

x_i	Age (X_1)	Income (X_2)
x_1	12	300
x_2	14	500
x_3	18	1000
x_4	23	2000
x_5	27	3500
x_6	28	4000
x_7	34	4300
x_8	37	6000
x_9	39	2500
x_{10}	40	2700

Scaled values are ...

Feature Scaling - Example

$\mathbf{x}_i$	Age (X_1)	Income (X_2)
$\mathbf{x}_1$	12	300
$\mathbf{x}_2$	14	500
$\mathbf{x}_3$	18	1000
$\mathbf{x}_4$	23	2000
$\mathbf{x}_5$	27	3500
$\mathbf{x}_6$	28	4000
$\mathbf{x}_7$	34	4300
$\mathbf{x}_8$	37	6000
$\mathbf{x}_9$	39	2500
$\mathbf{x}_{10}$	40	2700

X_normalized - NumPy array

	0	1
0	0	0
1	0.0714286	0.0350877
2	0.214286	0.122807
3	0.392857	0.298246
4	0.535714	0.561404
5	0.571429	0.649123
6	0.785714	0.701754
7	0.892857	1
8	0.964286	0.385965
9	1	0.421053

Feature Scaling - Python

```
#@author: rizwan.khan
import numpy as np
from sklearn import preprocessing
from sklearn.preprocessing import StandardScaler

#Create Training Set, 2D vector,  Values from Zaki's book example
X=np.array([[12  , 300], [14  , 500], [18  , 1000], [23  , 2000], [27  ,
    3500],
[28  , 4000],[34  , 4300],[37  , 6000],[39  , 2500],[40  , 2700]])
# First Method: Range Normalization  (xi-min(xi))/(max(xi)-min(xi))

max_x1=np.max(X [:,0])
max_x2=np.max(X [:,1])
min_x1=np.min(X [:,0])
min_x2=np.min(X [:,1])

x1_tran=(X[:,0]-min_x1)/(max_x1-min_x1)
x2_tran=(X[:,1]-min_x2)/(max_x2-min_x2)

X_normalized =np.r_[x1_tran[None,:],x2_tran[None,:]]
X_normalized = np.transpose (X_normalized)
```

Feature Scaling - Example

x_i	Age (X_1)	Income (X_2)
x_1	12	300
x_2	14	500
x_3	18	1000
x_4	23	2000
x_5	27	3500
x_6	28	4000
x_7	34	4300
x_8	37	6000
x_9	39	2500
x_{10}	40	2700

Scale feature using Equation 7 :
$x^{scaled} = \frac{x-\bar{x}}{\sigma}$

Feature Scaling - Example

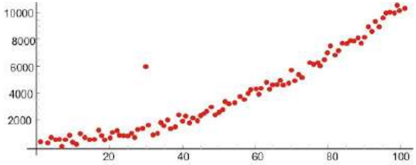
x_i	Age (X_1)	Income (X_2)
x_1	12	300
x_2	14	500
x_3	18	1000
x_4	23	2000
x_5	27	3500
x_6	28	4000
x_7	34	4300
x_8	37	6000
x_9	39	2500
x_{10}	40	2700

X_s - NumPy array

	0	1
0	-1.55654	-1.37879
1	-1.35173	-1.26292
2	-0.942117	-0.973263
3	-0.430097	-0.39394
4	-0.0204808	0.475045
5	0.0819232	0.764707
6	0.696347	0.938504
7	1.00356	1.92335
8	1.20837	-0.104278
9	1.31077	0.0115865

Detecting Outliers

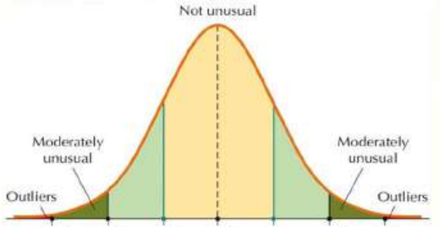
- Outliers are data points with values very different from the mean value. Thus they may produce large errors during training and can have disastrous effects. For example AdaBoost increase the weights of misclassified example, thus outliers can have more weights as they tend to be often misclassified.
- linear & logistic regression are easily impacted by the outliers in the training data, so does $K-NN$, if K is small.



Detecting Outliers

- Outliers are data points with values very different from the mean value. Thus they may produce large errors during training and can have disastrous effects. For example AdaBoost increase the weights of misclassified example, thus outliers can have more weights as they tend to be often misclassified.
- linear & logistic regression are easily impacted by the outliers in the training data, so does $K - NN$, if K is small.

- Common methods for detecting outliers :
  - Z-Score: Z-Score is calculated using Equation 7. The data points which are way too far from zero mean can be outliers.



Detecting Outliers

- Outliers are data points with values very different from the mean value. Thus they may produce large errors during training and can have disastrous effects. For example AdaBoost increase the weights of misclassified example, thus outliers can have more weights as they tend to be often misclassified.
- linear & logistic regression are easily impacted by the outliers in the training data, so does $K - NN$, if K is small.
- Common methods for detecting outliers :
  - Z-Score: Z-Score is calculated using Equation 7. The data points which are way too far from zero mean can be outliers.
  - Box-Plot : This is quickest and easiest way to identify outliers is by visualizing them using plots.

Dealing with Outliers

- If the number of outliers is very small and dataset is large enough, outliers are usually discarded.

Dealing with Outliers

- If the number of outliers is very small and dataset is large enough, outliers are usually discarded.
- In some applications where dataset is small, dropping data is a harsh step and should be avoided.
- Few techniques to deal with outliers, if they are not dropped:
  - Winsorizing : setting the extreme values of an attribute to some specified value. For example, for a 90% Winsorization, the bottom 5% of values are set equal to the minimum value in the 5th percentile, while the upper 5% of values are set equal to the maximum value in the 95th percentile.

Dealing with Outliers

- If the number of outliers is very small and dataset is large enough, outliers are usually discarded.
- In some applications where dataset is small, dropping data is a harsh step and should be avoided.
- Few techniques to deal with outliers, if they are not dropped:
  - Winsorizing : setting the extreme values of an attribute to some specified value. For example, for a 90% Winsorization, the bottom 5% of values are set equal to the minimum value in the 5th percentile, while the upper 5% of values are set equal to the maximum value in the 95th percentile.
  - Log-Scale Transformation : This method is often used to reduce the variability of data including outlying observation.

Dealing with Outliers

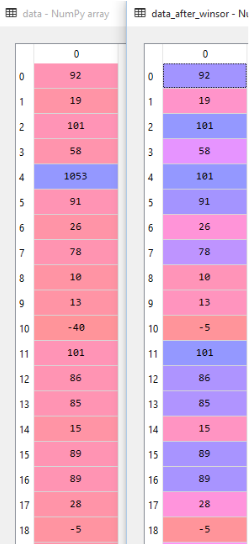
- If the number of outliers is very small and dataset is large enough, outliers are usually discarded.
- In some applications where dataset is small, dropping data is a harsh step and should be avoided.
- Few techniques to deal with outliers, if they are not dropped:
  - Winsorizing : setting the extreme values of an attribute to some specified value. For example, for a 90% Winsorization, the bottom 5% of values are set equal to the minimum value in the 5th percentile, while the upper 5% of values are set equal to the maximum value in the 95th percentile.
  - Log-Scale Transformation : This method is often used to reduce the variability of data including outlying observation.
  - Adopt cost functions that are not very sensitive in the presence of outliers.

Winsorization: Python

```
"""
@author: rizwan.khan
"""

import scipy.stats
import numpy as np
a = np.array([92, 19, 101, 58, 1053, 91, 26, 78,
    10, 13, -40, 101, 86, 85, 15, 89, 89, 28, -5,
    41])

print(a)
print(scipy.stats.mstats.winsorize(a, limits
    =[0.05, 0.05]))
```

Winsorization: Python

```python
"""
@author: rizwan.khan
"""

import scipy.stats
import numpy as np
a = np.array([92, 19, 101, 58, 1053, 91, 26, 78,
    10, 13, -40, 101, 86, 85, 15, 89, 89, 28, -5,
    41])

print(a)
print(scipy.stats.mstats.winsorize(a, limits
    =[0.05, 0.05]))
```



	data - NumPy array (0)	data_after_winsor - Nu (0)
0	92	92
1	19	19
2	101	101
3	58	58
4	1053	101
5	91	91
6	26	26
7	78	78
8	10	10
9	13	13
10	-40	-5
11	101	101
12	86	86
13	85	85
14	15	15
15	89	89
16	89	89
17	28	28
18	-5	-5

Section Contents

Workflow: Supervised Learning Algorithm

Workflow of a supervised learning algorithm for classification:

1. Data preprocessing and feature extraction (not required in DL)

Workflow: Supervised Learning Algorithm

Workflow of a supervised learning algorithm for classification:

1. Data preprocessing and feature extraction (not required in DL)
2. Training phase : $\{< x_i, y_i >\} \rightarrow algorithm$

Workflow: Supervised Learning Algorithm

Workflow of a supervised learning algorithm for classification:

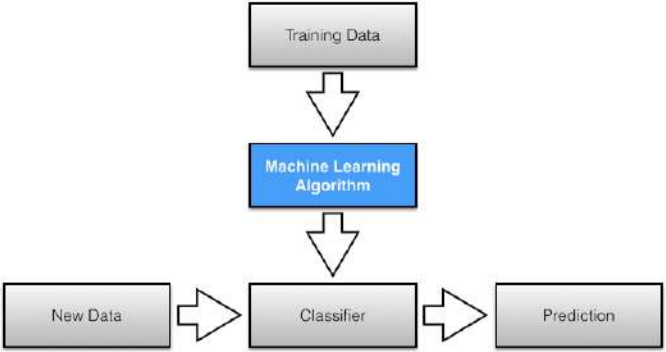
1. Data preprocessing and feature extraction (not required in DL)
2. Training phase : $\{< x_i, y_i >\} \rightarrow algorithm$
3. Evaluation phase : provides feedback to improve model accuracy.

Workflow: Supervised Learning Algorithm

Workflow of a supervised learning algorithm for classification:

1. Data preprocessing and feature extraction (not required in DL)
2. Training phase : $\{< x_i, y_i >\} \rightarrow algorithm$
3. Evaluation phase : provides feedback to improve model accuracy.

The training process is repeated until a desired accuracy level is achieved

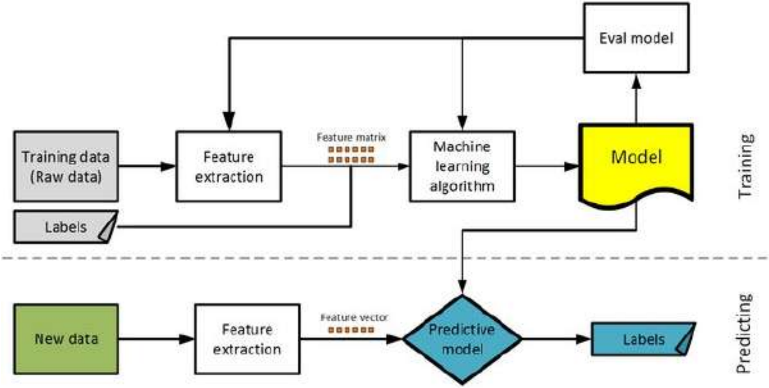


Workflow: Supervised Learning Algorithm

Workflow of a supervised learning algorithm for classification:

1. Data preprocessing and feature extraction (not required in DL)
2. Training phase : $\{< x_i, y_i >\} \rightarrow algorithm$
3. Evaluation phase : provides feedback to improve model accuracy.

The training process is repeated until a desired accuracy level is achieved



Model Evaluation

- Classifiers (both supervised and unsupervised) are learned / trained on a finite training set.

Model Evaluation

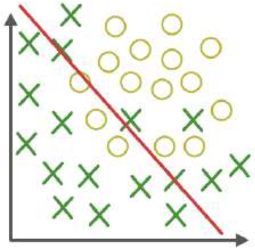
- Classifiers (both supervised and unsupervised) are learned / trained on a finite training set.
- A learned classifier has to be tested on a different test set to gauge its performance.

Model Evaluation

- Classifiers (both supervised and unsupervised) are learned / trained on a finite training set.
- A learned classifier has to be tested on a different test set to gauge its performance.
- The experimental performance on the test data is a proxy for the performance on unseen data. It checks the classifier's generalization ability.

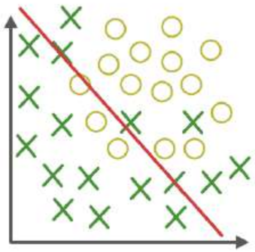
Model Evaluation

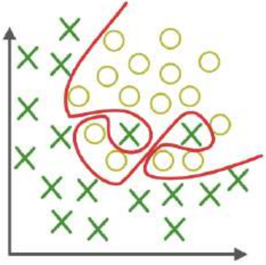
- Classifiers (both supervised and unsupervised) are learned / trained on a finite training set.
- A learned classifier has to be tested on a different test set to gauge its performance.
- The experimental performance on the test data is a proxy for the performance on unseen data. It checks the classifier's generalization ability.



Model Evaluation

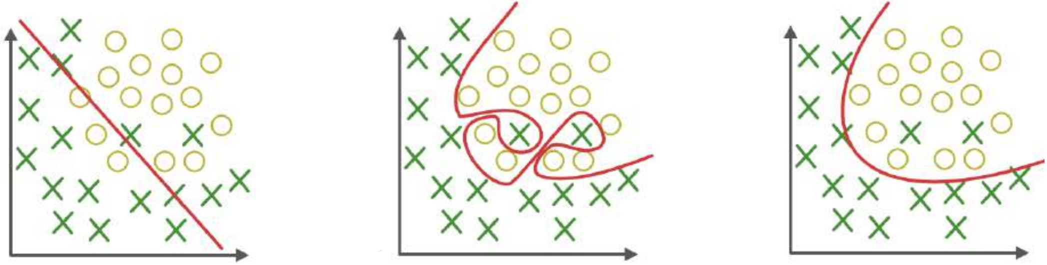
- Classifiers (both supervised and unsupervised) are learned / trained on a finite training set.
- A learned classifier has to be tested on a different test set to gauge its performance.
- The experimental performance on the test data is a proxy for the performance on unseen data. It checks the classifier's generalization ability.





Model Evaluation

- Classifiers (both supervised and unsupervised) are learned / trained on a finite training set.
- A learned classifier has to be tested on a different test set to gauge its performance.
- The experimental performance on the test data is a proxy for the performance on unseen data. It checks the classifier's generalization ability.



Learning the training data too precisely usually leads to poor classification results on new data.

Model Evaluation

- How machine learning trained model generalizes on unseen data is an important aspect. As aim of trained model is to correctly predict new examples. Good training accuracy can be achieved from memorizing trained data.

Model Evaluation

- How machine learning trained model generalizes on unseen data is an important aspect. As aim of trained model is to correctly predict new examples. Good training accuracy can be achieved from memorizing trained data.
- The above issue can be handled by evaluating the performance (generalization capability) of a trained model model on unseen data, separated from available dataset. Following are few dataset partitioning techniques:
  - Hold out
  - $k - fold$ Cross validation
  - Bootstrap
  - Leave-one-out cross-validation

Model Evaluation

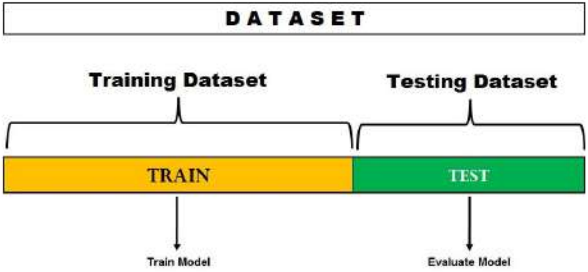
- How machine learning trained model generalizes on unseen data is an important aspect. As aim of trained model is to correctly predict new examples. Good training accuracy can be achieved from memorizing trained data.
- The above issue can be handled by evaluating the performance (generalization capability) of a trained model model on unseen data, separated from available dataset. Following are few dataset partitioning techniques:
  - Hold out
  - $k - fold$ Cross validation
  - Bootstrap
  - Leave-one-out cross-validation

- More training data gives better generalization.
- More test data gives better estimate for the classification error probability.
- Never evaluate performance on training data. The conclusion would be biased.

Hold Out Cross Validation

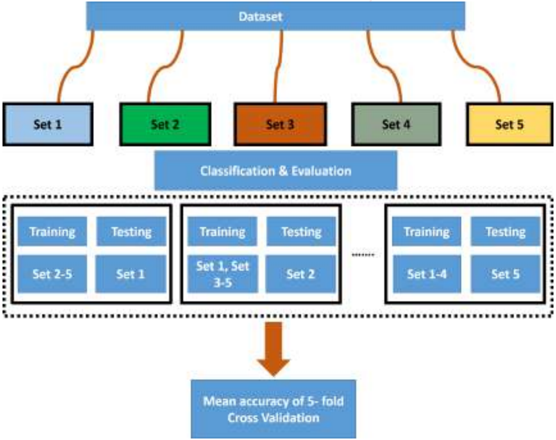
Hold out cross validation

- Given data is randomly partitioned into two independent sets i.e. training set and the testing set.
- The function approximator / classifier fits a function using the training set only. Then learned model is used to predict the output values for the data in the testing set.
- It is now becoming a common practice to use three instead of two data sets: one for training, one for validation, and one for testing.. More on this later.



$k - fold$ Cross Validation

In *k*-fold cross validation, dataset is divided into k equal subsets. k-1 subsets are used for the training while a single set is retained for testing. The process is repeated k times (k-folds), with each of the k subsets used exactly once for testing. Then, the k estimations (accuracy) from k-folds are averaged to produce final estimated value.



Bootstrap

- The bootstrap (also called *bagging*[1]) uses sampling with replacement to form the training set.

[1]Proposed in: Breiman, Leo (1996). Bagging predictors. Machine Learning 24 (2): 123–140.

Bootstrap

- The bootstrap (also called *bagging*[1]) uses sampling with replacement to form the training set.
- Given: the training set T consisting of n entries.

[1] Proposed in: Breiman, Leo (1996). Bagging predictors. Machine Learning 24 (2): 123–140.

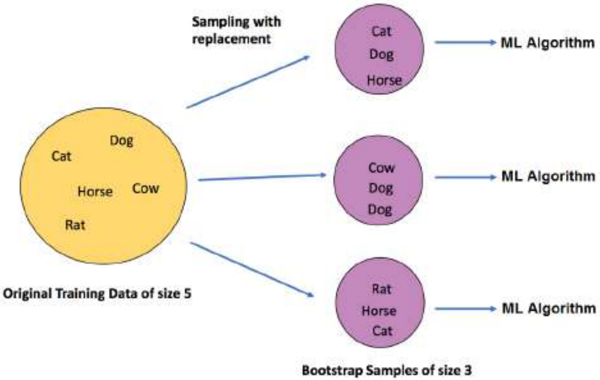
Bootstrap

- The bootstrap (also called *bagging*[1]) uses sampling with replacement to form the training set.
- Given: the training set T consisting of n entries.
- Bootstrap generates m new datasets T_i each of size $n' < n$ by sampling T uniformly with replacement. The consequence is that some entries can be repeated in T_i.

[1] Proposed in: Breiman, Leo (1996). Bagging predictors. Machine Learning 24 (2): 123–140.

Bootstrap

- The bootstrap (also called *bagging*[1]) uses sampling with replacement to form the training set.
- Given: the training set T consisting of n entries.
- Bootstrap generates m new datasets T_i each of size $n' < n$ by sampling T uniformly with replacement. The consequence is that some entries can be repeated in T_i.
- The m statistical models (e.g., classifiers, regressors) are learned using the above m bootstrap samples.



[1]Proposed in: Breiman, Leo (1996). Bagging predictors. Machine Learning 24 (2): 123–140.

Leave-One-Out Cross-Validation

1. Do N experiments. In each experiment, use $N - 1$ samples for training, and leave only 1 sample for testing.

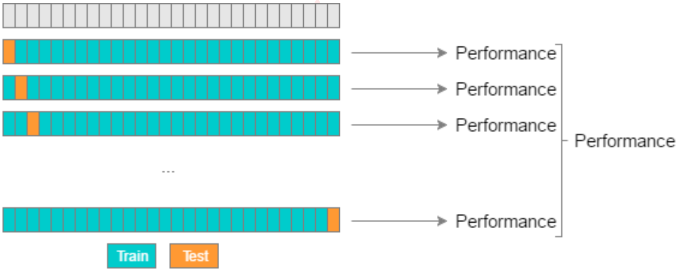
Leave-One-Out Cross-Validation

1. Do N experiments. In each experiment, use $N-1$ samples for training, and leave only 1 sample for testing.
2. Compute the testing error $E_i, i = 1, 2, \ldots, N$.

Leave-One-Out Cross-Validation

1. Do N experiments. In each experiment, use $N-1$ samples for training, and leave only 1 sample for testing.
2. Compute the testing error $E_i, i = 1, 2, \ldots, N$.
3. After N experiments, compute the overall estimated error:

$$E = \frac{1}{N}\sum_{i=1}^{N} E_i \tag{8}$$



Error Estimation

- Performance evaluation metrics explain the performance of a model on unseen data and provides feedback. Thus allowing to make continuous improvements till desired accuracy is achieved.

Error Estimation

- Performance evaluation metrics explain the performance of a model on unseen data and provides feedback. Thus allowing to make continuous improvements till desired accuracy is achieved.
- The choice of evaluation metrics / error estimation depends on a problem in hand (such as classification, regression, clustering, topic modeling, among others) and final goal.

Error Estimation

- Performance evaluation metrics explain the performance of a model on unseen data and provides feedback. Thus allowing to make continuous improvements till desired accuracy is achieved.
- The choice of evaluation metrics / error estimation depends on a problem in hand (such as classification, regression, clustering, topic modeling, among others) and final goal.
- Most used classification performance evaluation metrics:
  - Classification accuracy
  - Confusion matrix
  - Precision
  - Recall
  - F-Measure
  - Receiver Operating Characteristic (ROC) Area Under Curve (AUC)
  - Logarithmic loss

Classification Accuracy

- The most simple way to calculate the accuracy of any classification machine learning model is:

$$Accuracy = \frac{N_{cc}}{D_{ts}}$$

where N_{cc} = Number of examples correctly classified and D_{ts} = total examples in testing data set.

Classification Accuracy

- The most simple way to calculate the accuracy of any classification machine learning model is:

$$Accuracy = \frac{N_{cc}}{D_{ts}}$$

where N_{cc} = Number of examples correctly classified and D_{ts} = total examples in testing data set.

- N_{cc} in actual is sum of true positives T_p and true negative T_n.

Classification Accuracy

- The most simple way to calculate the accuracy of any classification machine learning model is:

$$Accuracy = \frac{N_{cc}}{D_{ts}}$$

where N_{cc} = Number of examples correctly classified and D_{ts} = total examples in testing data set.

- N_{cc} in actual is sum of true positives T_p and true negative T_n.

Any drawbacks?

Classification Accuracy

- The most simple way to calculate the accuracy of any classification machine learning model is:

$$Accuracy = \frac{N_{cc}}{D_{ts}}$$

where N_{cc} = Number of examples correctly classified and D_{ts} = total examples in testing data set.

- N_{cc} in actual is sum of true positives T_p and true negative T_n.

Any drawbacks?

Consider that in D_{ts}, 98% samples belongs to class A (class imbalance problem). According to this method, model can achieve 98% accuracy by simply predicting every training sample to class A.

Classification Accuracy

- The real problem arises, when the cost of misclassification of the minor class samples are very high. For example If we deal with a rare but fatal disease, the cost of failing to diagnose (False negative, F_n) the disease of a sick person is much higher than the cost of sending a healthy person to more tests (False positive, F_p).

Classification Accuracy

- The real problem arises, when the cost of misclassification of the minor class samples are very high. For example If we deal with a rare but fatal disease, the cost of failing to diagnose (False negative, F_n) the disease of a sick person is much higher than the cost of sending a healthy person to more tests (False positive, F_p).
- Let's understand:
  1. True positives T_p: Classifier predicted disease and the person actually has the disease.

Classification Accuracy

- The real problem arises, when the cost of misclassification of the minor class samples are very high. For example If we deal with a rare but fatal disease, the cost of failing to diagnose (False negative, F_n) the disease of a sick person is much higher than the cost of sending a healthy person to more tests (False positive, F_p).
- Let's understand:
  1. True positives T_p: Classifier predicted disease and the person actually has the disease.
  2. True negative T_n: Classifier predicted no disease and the person actually is healthy.

Classification Accuracy

- The real problem arises, when the cost of misclassification of the minor class samples are very high. For example If we deal with a rare but fatal disease, the cost of failing to diagnose (False negative, F_n) the disease of a sick person is much higher than the cost of sending a healthy person to more tests (False positive, F_p).
- Let's understand:
  1. True positives T_p: Classifier predicted disease and the person actually has the disease.
  2. True negative T_n: Classifier predicted no disease and the person actually is healthy.
  3. False positive F_p: Classifier predicted disease but the person actually is healthy, also known as a "Type I error".

Classification Accuracy

- The real problem arises, when the cost of misclassification of the minor class samples are very high. For example If we deal with a rare but fatal disease, the cost of failing to diagnose (False negative, F_n) the disease of a sick person is much higher than the cost of sending a healthy person to more tests (False positive, F_p).
- Let's understand:
  1. True positives T_p: Classifier predicted disease and the person actually has the disease.
  2. True negative T_n: Classifier predicted no disease and the person actually is healthy.
  3. False positive F_p: Classifier predicted disease but the person actually is healthy, also known as a "Type I error".
  4. False negative F_n: Classifier predicted no disease and the person actually has the disease, also known as a "Type II error".

Confusion Matrix

- To overcome problem shown above, we have a diagnostic / visualization tool, called Confusion Matrix.

Confusion Matrix

- To overcome problem shown above, we have a diagnostic / visualization tool, called Confusion Matrix.
- It contains information about actual and predicted classifications. For Example:

Confusion Matrix

- To overcome problem shown above, we have a diagnostic / visualization tool, called Confusion Matrix.
- It contains information about actual and predicted classifications. For Example:

		Predicted Class	
		Disease	No Disease
Actual Class	Disease	**45**	20
	No Disease	5	**30**

Please label these quantities as T_p, T_n, F_p & F_n

Confusion Matrix

- To overcome problem shown above, we have a diagnostic / visualization tool, called Confusion Matrix.
- It contains information about actual and predicted classifications. For Example:

		Predicted Class	
		Disease	No Disease
Actual Class	Disease	45 T_p	20 F_n
	No Disease	5 F_p	30 T_n

Please label these quantities as T_p, T_n, F_p & F_n

Confusion Matrix

- To find out how the errors are distributed across the classes we construct a confusion matrix using the testing data set D_{ts}. The entry a_{ij} (off-diagonal) of such a matrix denotes the number of elements from D_{ts} whose true class is w_i, and which are assigned by classifier to class other than w_i.

[4] https://tel.archives-ouvertes.fr/tel-01166539/

Confusion Matrix

- To find out how the errors are distributed across the classes we construct a confusion matrix using the testing data set D_{ts}. The entry a_{ij} (off-diagonal) of such a matrix denotes the number of elements from D_{ts} whose true class is w_i, and which are assigned by classifier to class other than w_i.

	Sadness	Happiness	Surprise	Anger	Disgust	Fear
Sadness	**68.1**	8.8	6.3	4.3	3.5	9
Happiness	10.8	**70.8**	16.4	0	2	0
Surprise	9	10.8	**70.1**	4	1.7	4.4
Anger	0	10.5	0	**62.1**	15.1	12.3
Disgust	10.3	15.5	8.4	0	**63.3**	2.5
Fear	3	2.6	3.3	10.1	20.7	**60.3**

Confusion matrix (in multiclass problem, define one class as $+ve$ and rest of other as $-ve$) from my PhD research[4]

- The additional information that the confusion matrix provides is where the misclassifications have occurred.

[4] https://tel.archives-ouvertes.fr/tel-01166539/

Confusion Matrix

- To find out how the errors are distributed across the classes we construct a confusion matrix using the testing data set D_{ts}. The entry a_{ij} (off-diagonal) of such a matrix denotes the number of elements from D_{ts} whose true class is w_i, and which are assigned by classifier to class other than w_i.

	Sadness	Happiness	Surprise	Anger	Disgust	Fear
Sadness	**68.1**	8.8	6.3	4.3	3.5	9
Happiness	10.8	**70.8**	16.4	0	2	0
Surprise	9	10.8	**70.1**	4	1.7	4.4
Anger	0	10.5	0	**62.1**	15.1	12.3
Disgust	10.3	15.5	8.4	0	**63.3**	2.5
Fear	3	2.6	3.3	10.1	20.7	**60.3**

Confusion matrix (in multiclass problem, define one class as $+ve$ and rest of other as $-ve$) from my PhD research[4]

- The additional information that the confusion matrix provides is where the misclassifications have occurred.
- Information provided can help to focus on classes that are difficult to classify, or classes that are more similar / confusing than others.

[4] https://tel.archives-ouvertes.fr/tel-01166539/

Sensitivity, Specificity & Precision

Actual Class \ Predicted Class	Spam	Non-Spam
Spam	TP=45	FN=20
Non-Spam	FP=5	TN=30

- Sensitivity / Recall calculates the ratio of positive class correctly detected. This metric gives how good the model is to recognize a positive class. $\frac{T_p}{T_p+F_n}$

Sensitivity, Specificity & Precision

Actual Class \ Predicted Class	Spam	Non-Spam
Spam	TP=45	FN=20
Non-Spam	FP=5	TN=30

- Sensitivity / Recall calculates the ratio of positive class correctly detected. This metric gives how good the model is to recognize a positive class. $\frac{T_p}{T_p+F_n}$
- Specificity is characterized as the ratio of actual negatives, which model predicted as a negative class or true negative. $\frac{F_p}{F_p+T_n}$

Sensitivity, Specificity & Precision

		Predicted Class: Spam	Predicted Class: Non-Spam
Actual Class	Spam	TP=45	FN=20
Actual Class	Non-Spam	FP=5	TN=30

- Sensitivity / Recall calculates the ratio of positive class correctly detected. This metric gives how good the model is to recognize a positive class. $\frac{T_p}{T_p+F_n}$
- Specificity is characterized as the ratio of actual negatives, which model predicted as a negative class or true negative. $\frac{F_p}{F_p+T_n}$
- Precision is ratio of total number of correctly classified positive examples and the total number of predicted positive examples. $\frac{T_p}{T_p+F_p}$

Sensitivity, Specificity & Precision

- Precision and recall both indicate accuracy but there is subtle difference between the two. Precision means the percentage of results which are relevant. Recall refers to the percentage of total relevant results correctly classified by algorithm.

Sensitivity, Specificity & Precision

- Precision and recall both indicate accuracy but there is subtle difference between the two. Precision means the percentage of results which are relevant. Recall refers to the percentage of total relevant results correctly classified by algorithm.

Trade-off

- To increase recall algorithm needs to keep generating results which are not accurate, hence lowering precision.

Sensitivity, Specificity & Precision

- Precision and recall both indicate accuracy but there is subtle difference between the two. Precision means the percentage of results which are relevant. Recall refers to the percentage of total relevant results correctly classified by algorithm.

Trade-off

- To increase recall algorithm needs to keep generating results which are not accurate, hence lowering precision.
- Thus, it is not possible to maximize both these metrics at the same time.

Sensitivity, Specificity & Precision

- Precision and recall both indicate accuracy but there is subtle difference between the two. Precision means the percentage of results which are relevant. Recall refers to the percentage of total relevant results correctly classified by algorithm.

Trade-off

- To increase recall algorithm needs to keep generating results which are not accurate, hence lowering precision.
- Thus, it is not possible to maximize both these metrics at the same time.
- For simplicity, there is another metric available, called F-score:

$$F_\alpha = (1+\alpha^2)\frac{\text{precision} \times recall}{\alpha^2 \times precision + recall} \tag{9}$$

F-score is a harmonic mean of precision and recall.

Sensitivity, Specificity & Precision

- Precision and recall both indicate accuracy but there is subtle difference between the two. Precision means the percentage of results which are relevant. Recall refers to the percentage of total relevant results correctly classified by algorithm.

Trade-off

- To increase recall algorithm needs to keep generating results which are not accurate, hence lowering precision.
- Thus, it is not possible to maximize both these metrics at the same time.
- For simplicity, there is another metric available, called F-score:

$$F_\alpha = (1 + \alpha^2)\frac{\text{precision} \times recall}{\alpha^2 \times precision + recall} \tag{9}$$

F-score is a harmonic mean of precision and recall.

- when $\alpha = 1$ (F1 - score)

$$F_1 = 2 \times \frac{\text{precision} \times recall}{precision + recall} \tag{10}$$

Sensitivity, Specificity & Precision: Quiz

In the following machine learning application domain, which metric would be more useful?

1. Cancer Detection

   1. Precision
   2. Recall
   3. Accuracy
   4. Specificity

2. Spam Email Identification

   1. Precision
   2. Recall
   3. Accuracy
   4. Specificity

3. Object Detection (balanced dataset)

   1. Precision
   2. Recall
   3. Accuracy
   4. Specificity

Sensitivity, Specificity & Precision: Quiz

In the following machine learning application domain, which metric would be more useful?

1. Cancer Detection
    1. Precision
    2. Recall
    3. Accuracy
    4. Specificity

    Recall
    Cost of failing to diagnose the disease of a sick person F_n is much higher than the cost of sending a healthy person to more tests F_p.

2. Spam Email Identification
    1. Precision
    2. Recall
    3. Accuracy
    4. Specificity

3. Object Detection (balanced dataset)
    1. Precision
    2. Recall
    3. Accuracy
    4. Specificity

Sensitivity, Specificity & Precision: Quiz

In the following machine learning application domain, which metric would be more useful?

1. Cancer Detection

1. Precision
2. Recall
3. Accuracy
4. Specificity

Recall

Cost of failing to diagnose the disease of a sick person F_n is much higher than the cost of sending a healthy person to more tests F_p.

2. Spam Email Identification

1. Precision
2. Recall
3. Accuracy
4. Specificity

Precision

It is important that legitimate email should not be classified as Spam, so cost of F_p is high.

3. Object Detection (balanced dataset)

1. Precision
2. Recall
3. Accuracy
4. Specificity

Sensitivity, Specificity & Precision: Quiz

In the following machine learning application domain, which metric would be more useful?

1. Cancer Detection

1. Precision
2. Recall
3. Accuracy
4. Specificity

Recall

Cost of failing to diagnose the disease of a sick person F_n is much higher than the cost of sending a healthy person to more tests F_p.

2. Spam Email Identification

1. Precision
2. Recall
3. Accuracy
4. Specificity

Precision

It is important that legitimate email should not be classified as Spam, so cost of F_p is high.

3. Object Detection (balanced dataset)

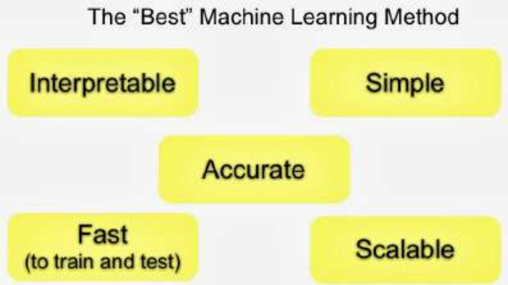
1. Precision
2. Recall
3. Accuracy
4. Specificity

Accuracy

Accuracy is a good measure when the target variable classes in the data are nearly balanced.

$$Accuracy = \frac{T_p+T_n}{T_p+F_p+T_n+F_n}$$

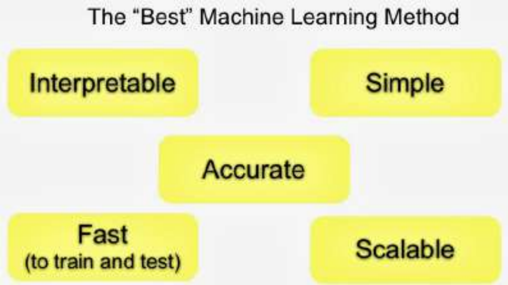
It's not only about numbers



http://radar.oreilly.com/2013/09/gaining-access-to-the-best-machine-learning-methods.html

1. Accuracy or Interpretability?
   Interpretability is critical if a model has to be explained for transparency.

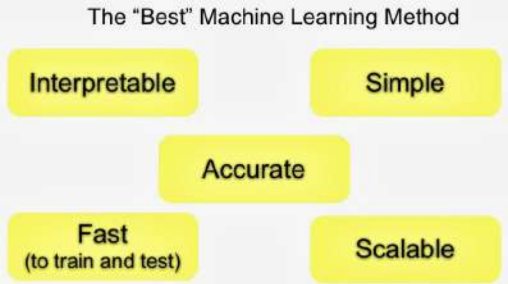
It's not only about numbers



http://radar.oreilly.com/2013/09/gaining-access-to-the-best-machine-learning-methods.html

1. Accuracy or Interpretability?
   Interpretability is critical if a model has to be explained for transparency.
2. Complex or Simple?
   Simplicity is important for practical reasons: it is impossible to tune model if model has "too many knobs to tune" .

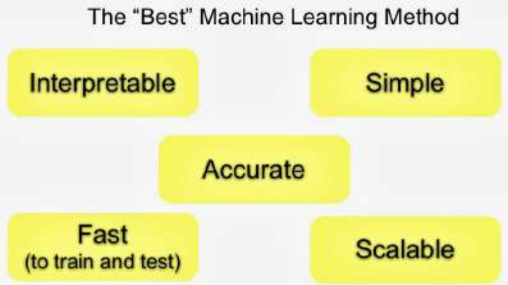
It's not only about numbers



http://radar.oreilly.com/2013/09/gaining-access-to-the-best-machine-learning-methods.html

1. Accuracy or Interpretability?
   Interpretability is critical if a model has to be explained for transparency.
2. Complex or Simple?
   Simplicity is important for practical reasons: it is impossible to tune model if model has "too many knobs to tune" .
3. Scalability?
   Either model needs to be scalable in terms of size of data or parameters.

It's not only about numbers



http://radar.oreilly.com/2013/09/gaining-access-to-the-best-machine-learning-methods.html

1. Accuracy or Interpretability?
   Interpretability is critical if a model has to be explained for transparency.
2. Complex or Simple?
   Simplicity is important for practical reasons: it is impossible to tune model if model has "too many knobs to tune" .
3. Scalability?
   Either model needs to be scalable in terms of size of data or parameters.
4. Fast prototyping?
   Either production needs to deliver fast or can R & D be initiated?

It's not only about numbers

Why Netflix Never Implemented The Algorithm That Won The Netflix $1 Million Challenge

Innovation

from the *times-change* dept
Fri, Apr 13th 2012 12:07am — **Mike Masnick**

You probably recall all the excitement that went around when a group **finally won** the big Netflix $1 million prize in 2009, improving Netflix's recommendation algorithm by 10%. But what you might *not* know, is that **Netflix never implemented that solution itself.** Netflix recently put up a blog post **discussing some of the details of its recommendation system**, which (as an aside) explains why the winning entry never was used. First, they note that they *did* make use of an earlier bit of code that came out of the contest:

> *A year into the competition, the Korbell team won the first Progress Prize with an 8.43% improvement. They reported more than 2000 hours of work in order to come up with the final combination of 107 algorithms that gave them this prize. And, they gave us the source code. We looked at the two underlying algorithms with the best performance in the ensemble: Matrix Factorization (which the community generally called SVD, Singular Value Decomposition) and Restricted Boltzmann Machines (RBM). SVD by itself provided a 0.8914 RMSE (root mean squared error), while RBM alone provided a competitive but slightly worse 0.8990 RMSE. A linear blend of these two reduced the error to 0.88. To put these algorithms to use, we had to work to overcome some limitations, for instance that they were built to handle 100 million ratings, instead of the more than 5 billion that we have, and that they were not built to adapt as members added more ratings. But once we overcame those challenges, we put the two algorithms into production, where they are still used as part of our recommendation engine.*

Section Contents

Further Reading

Further Reading

- Dense Vs. Sparse representation of feature vector.
- Data Preprocessing techniques in Machine Learning:
  - Handling Categorical Variables
  - One-Hot Encoding
  - Outlier handling - Cook's distance

Further Reading

Further Reading

- Dense Vs. Sparse representation of feature vector.
- Data Preprocessing techniques in Machine Learning:
  - Handling Categorical Variables
  - One-Hot Encoding
  - Outlier handling - Cook's distance
- Considering recent trend of having large datasets, which dataset partitioning technique is suitable?

Further Reading

Further Reading

- Dense Vs. Sparse representation of feature vector.
- Data Preprocessing techniques in Machine Learning:
  - Handling Categorical Variables
  - One-Hot Encoding
  - Outlier handling - Cook's distance
- Considering recent trend of having large datasets, which dataset partitioning technique is suitable?
- Dealing with Imbalance dataset.
- Article reading: The use of the area under the ROC curve in the evaluation of machine learning algorithms (`https://www.sciencedirect.com/science/article/abs/pii/S0031320396001422`)

Introduction to Machine Learning

Dr. Rizwan Ahmed Khan

Outline

1. Big Picture
   - Context
   - Demystifying AI
   - AI waves
2. Machine Learning
   - Intuition
   - What?
   - Why?
3. Taxonomy
   - Introduction
   - Supervised Learning
   - Unsupervised Learning
   - Reinforcement Learning
4. Workflow
   - Features
   - Python code
5. Examples

Reference Books

Reference books for this lecture:

- Chapter 1: Machine Learning, Tom MITCHELL, McGraw Hill, latest edition.

Reference Books

Reference books for this lecture:

- Chapter 1: Machine Learning, Tom MITCHELL, McGraw Hill, latest edition.
- Chapter 1: Pattern Classification, R. DUDA et al., Wiley Interscience, latest edition.

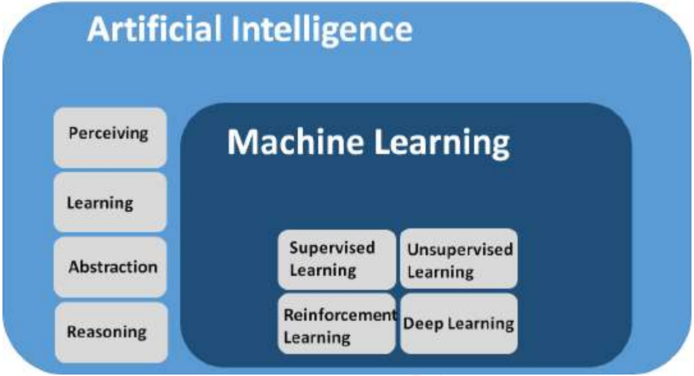
Reference Books

Reference books for this lecture:

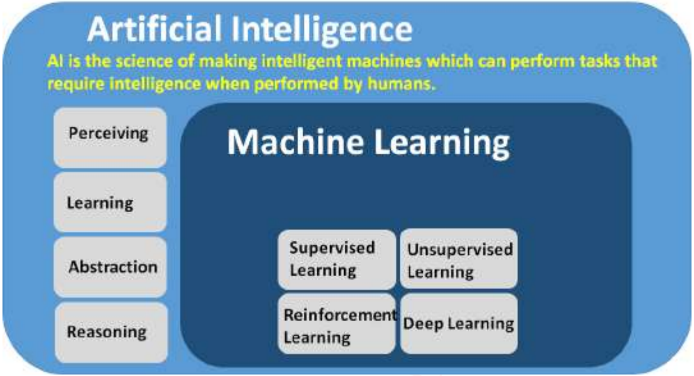
- Chapter 1: Machine Learning, Tom MITCHELL, McGraw Hill, latest edition.
- Chapter 1: Pattern Classification, R. DUDA et al., Wiley Interscience, latest edition.
- Chapter 1: Pattern Recognition, S. Theodoridis et al.,Academic Press, 4^{th} or latest edition.

Section Contents

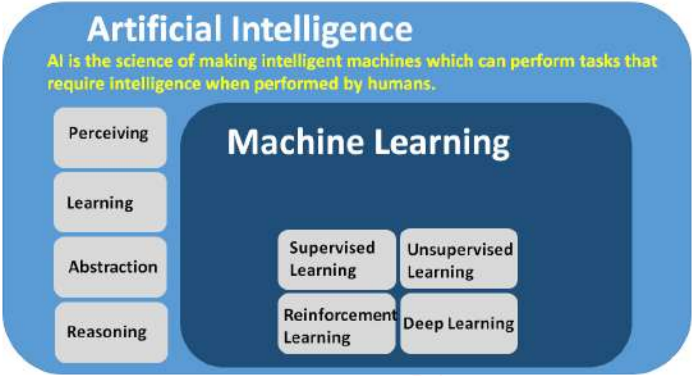
Context


Artificial Intelligence
Perceiving
Learning
Abstraction
Reasoning
Machine Learning
Supervised Learning
Unsupervised Learning
Reinforcement Learning
Deep Learning

Context


Artificial Intelligence
AI is the science of making intelligent machines which can perform tasks that require intelligence when performed by humans.
Perceiving
Learning
Abstraction
Reasoning
Machine Learning
Supervised Learning
Unsupervised Learning
Reinforcement Learning
Deep Learning

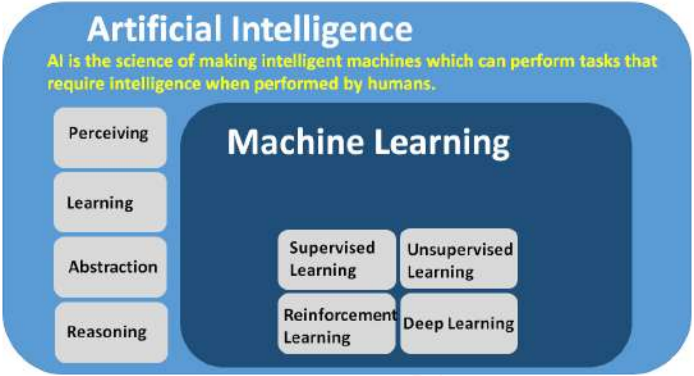
Context



Perception / Representation

In AI, perception is a process to interpret, acquire, select, and then organize the sensory information from the physical world to make actions like humans.

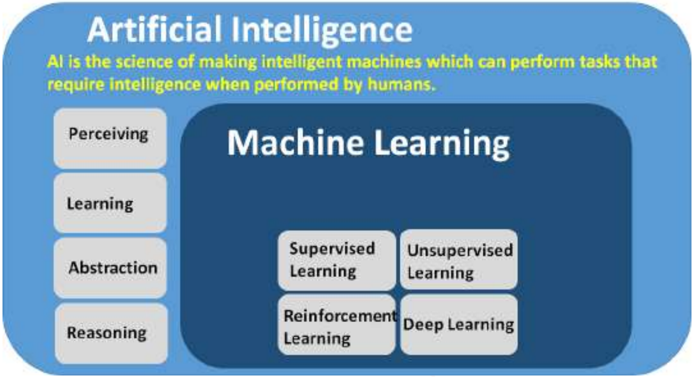
Context



Learning

Learning is the ability of a system to improve its behavior based on experience.

Context



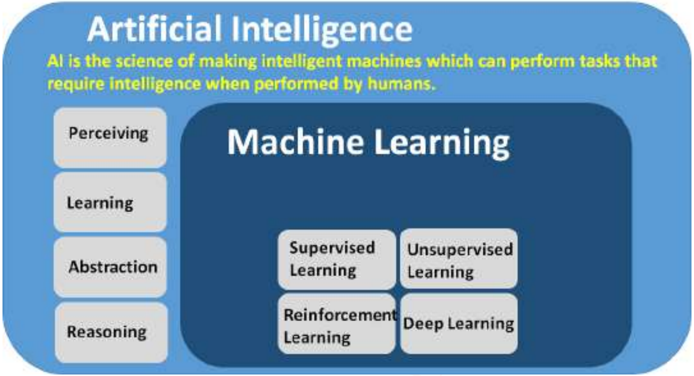
Reasoning

Reasoning is a way to infer facts from existing data. It is a general process of thinking rationally, to find valid conclusions.

Machine Learning Vs Machine Reasoning: one is about finding patterns, while the other is about understanding relationships (tackle new problems with a deductive and inductive reasoning approach) [a]

[a] From Machine Learning to Machine Reasoning, L Bottou 2011

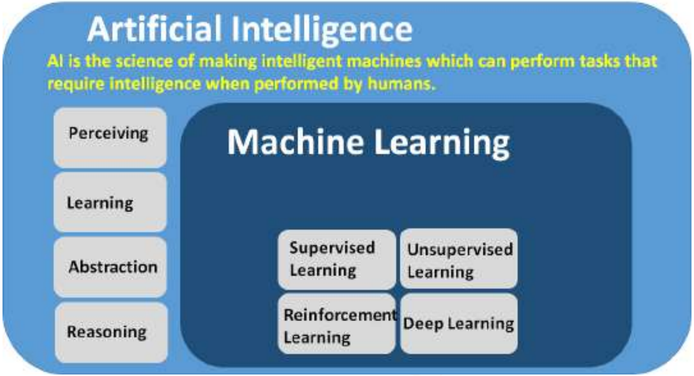
Context



Abstraction

Abstraction is a fundamental mechanism underlying both human and artificial perception, representation of knowledge, reasoning and learning. It aims at taking knowledge that is discovered at certain level and applying it up at another level.

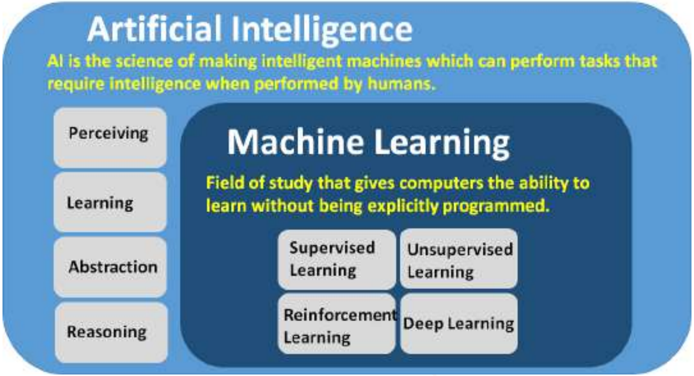
Context



Bottleneck

"The most important problem for AI today is abstraction and reasoning" — Francois Chollet-IBM

Context



AI / ML

- Write AI for fund-raising (Science fiction feel)
- Write Machine Learning for Hiring (Engineering sensibility)

AI Hype

- There is lot of hype about AI, that it will exceed the capabilities of human beings or will displace humanity.

AI Hype

- There is lot of hype about AI, that it will exceed the capabilities of human beings or will displace humanity.
- In this lecture and course I will try to demystify the hype and will discuss where technology currently is and where it is head. What it can do or can't do.

AI Hype

- There is lot of hype about AI, that it will exceed the capabilities of human beings or will displace humanity.
- In this lecture and course I will try to demystify the hype and will discuss where technology currently is and where it is head. What it can do or can't do.
- When people or media refer to term AI, usually they refer to General AI or AGI.

AI Hype

- There is lot of hype about AI, that it will exceed the capabilities of human beings or will displace humanity.
- In this lecture and course I will try to demystify the hype and will discuss where technology currently is and where it is head. What it can do or can't do.
- When people or media refer to term AI, usually they refer to General AI or AGI.

Artificial General Intelligence (AGI)

Hypothetical intelligence of a machine that has the capacity to understand or learn any intellectual task that a human being can. Full autonomy, topic of science fiction (at the moment).

AI Hype

- There is lot of hype about AI, that it will exceed the capabilities of human beings or will displace humanity.
- In this lecture and course I will try to demystify the hype and will discuss where technology currently is and where it is head. What it can do or can't do.
- When people or media refer to term AI, usually they refer to General AI or AGI.

Artificial General Intelligence (AGI)

Hypothetical intelligence of a machine that has the capacity to understand or learn any intellectual task that a human being can. Full autonomy, topic of science fiction (at the moment).

Artificial Narrow Intelligence (ANI)

ANI is focused on one narrow task. Every sort of machine intelligence that surrounds us today is Narrow AI.

- Google Assistant
- Google Translate
- Siri
- Recommender systems, etc.

AI : Why Now?

Note: [1]

[1] Image inspiration: MIT-Mathematics of Big Data and Machine Learning

AI : Why Now?

Note: [1]

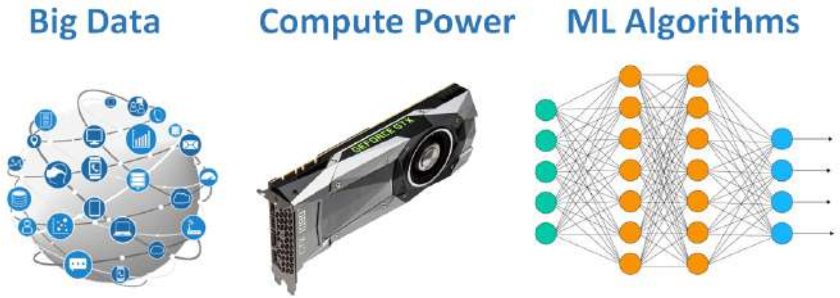
[1] Image inspiration: MIT-Mathematics of Big Data and Machine Learning

AI : Why Now?

Note: [1]

[1] Image inspiration: MIT-Mathematics of Big Data and Machine Learning

AI : Why Now?

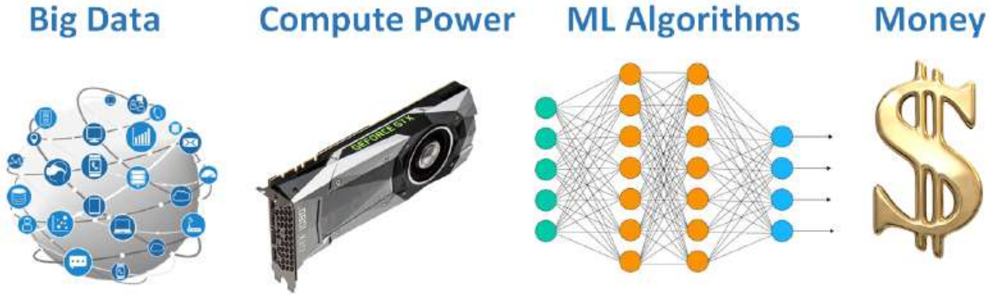


Convergence of big data, compute power, advancements in machine learning algorithms and investment (big) helped in widespread AI development / deployment.

Note: [1]

[1] Image inspiration: MIT-Mathematics of Big Data and Machine Learning

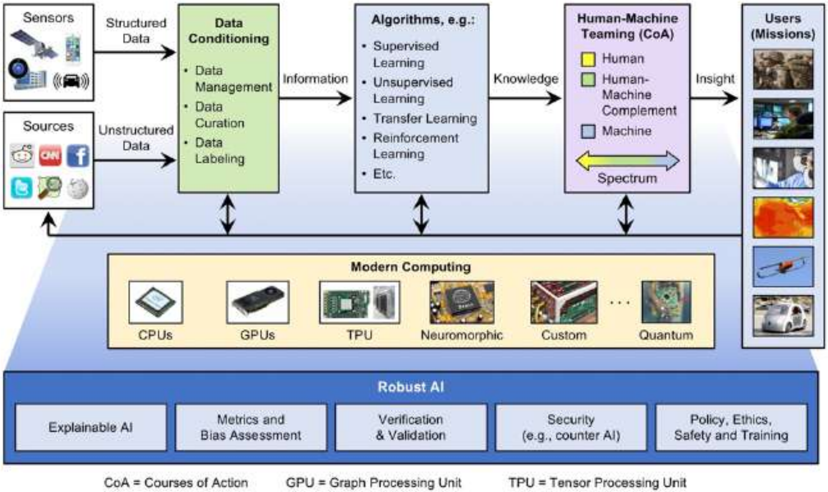
AI : Why Now?



Note: [1]

[1] Image inspiration: MIT-Mathematics of Big Data and Machine Learning

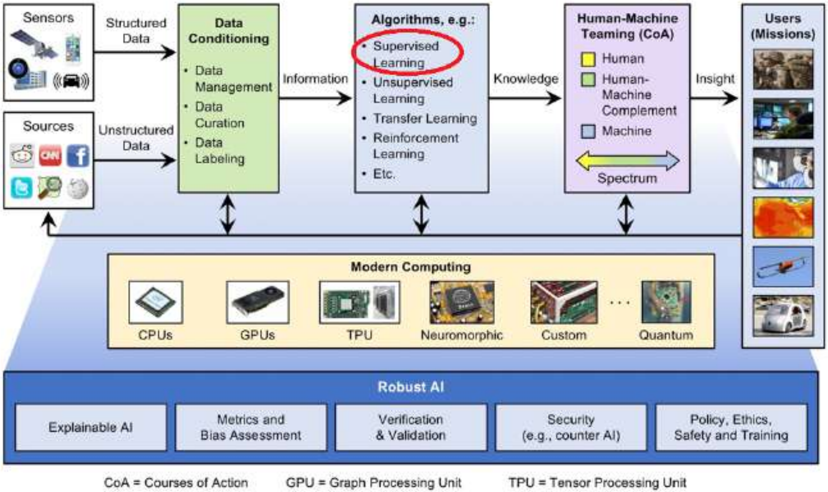
AI System Architecture : End-to-End Pipeline



- Data Conditioning, relates to pre-processing steps
- Algorithms: Life beyond NN or DNN

[2] Image courtesy: MIT-Mathematics of Big Data and Machine Learning

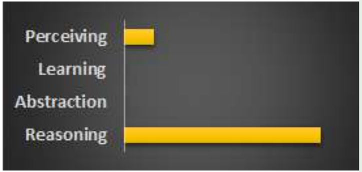
AI System Architecture : End-to-End Pipeline



- Data Conditioning, relates to pre-processing steps
- Algorithms: Life beyond NN or DNN
- Supervised Learning: This course

[2] Image courtesy: MIT-Mathematics of Big Data and Machine Learning

First AI wave : Reasoning

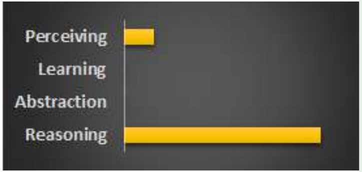


Handcrafted knowledge / Reasoning based systems

- Experts took knowledge (a particular domain) and characterize it in rule that fit in the computers. Good at explainability of AI (XAI).

[3] Waves adapted from John Launchbury-DARPA

First AI wave : Reasoning

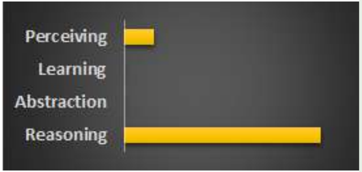


Handcrafted knowledge / Reasoning based systems

- Experts took knowledge (a particular domain) and characterize it in rule that fit in the computers. Good at explainability of AI (XAI).
- Huge data not required. No learning. Operate in narrow domain. No perception (doesn't sense natural world).

[3] Waves adapted from John Launchbury-DARPA

First AI wave : Reasoning



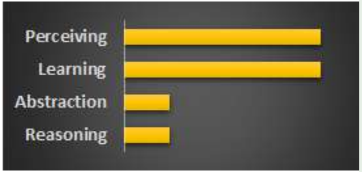
Handcrafted knowledge / Reasoning based systems

- Experts took knowledge (a particular domain) and characterize it in rule that fit in the computers. Good at explainability of AI (XAI).
- Huge data not required. No learning. Operate in narrow domain. No perception (doesn't sense natural world).
- Example: Expert System. Reasoning through knowledge, represented mainly as if–then rules.
  - MYCIN: diagnosis of infectious diseases.
  - CaDet: identification of cancer.
  - IBM's Deep Blue: Defeated chess champion in 1997.
- Enables reasoning over narrowly defined problems but with no learning and abstraction (handling uncertainty) capabilities. Still valid today (for some applications).

∗3

[3] Waves adapted from John Launchbury-DARPA

Second AI wave : Learning



Statistical / Machine Learning

- Enabled by learning algorithms and lots of data. Algorithm itself learns rules / patterns from the data to make prediction on unseen data.
- Good to perceive natural world, e.g. identify person, object, sound etc.
- They are not capable to contextualize / abstract information and provide limited reasoning power (black box).
- Most of recent success is based on research and advancements in ML algorithms. Examples of ML based tools (more on this later):
  - SIRI / Google Assistant
  - Autonomous cars
  - Spam filters
  - Medical diagnosis

Challenges with second AI wave

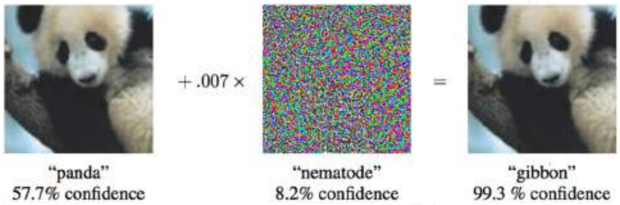
While ML / neural networks achieve statistically impressive results across large sample sizes, they are "individually unreliable" and often make mistakes humans would never make.

Robustness

ML algo results are only as good as data it is trained on. Neural networks fed inaccurate or incomplete data will simply produce the wrong results.

Challenges with second AI wave

While ML / neural networks achieve statistically impressive results across large sample sizes, they are "individually unreliable" and often make mistakes humans would never make.



Object Recognition

No robustness against noise

Challenges with second AI wave

While ML / neural networks achieve statistically impressive results across large sample sizes, they are "individually unreliable" and often make mistakes humans would never make.

Face recognition

With colorful glasses system failed

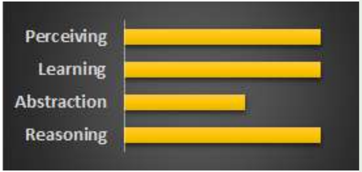
Challenges with second AI wave

While ML / neural networks achieve statistically impressive results across large sample sizes, they are "individually unreliable" and often make mistakes humans would never make.

Microsoft's Tay-Tweets

Microsoft took it down just after 24 hours. This chat-bot got offensive messages and learned the pattern (skewed training data).

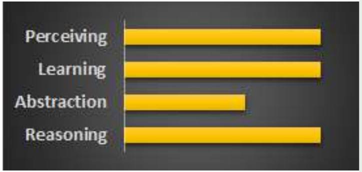
Third (future) AI wave : Context



Contextual Adaptation

- To remove bottlenecks of techniques of second wave of AI. Research is at beginning stages.
- Third wave: systems construct explanatory models that allow them to characterize real-world phenomena.
  - **Example:** Third Wave AI will not only recognize the “cat”, but will be able to explain why it’s a cat and how it arrived at that conclusion (i.e. has a fur, two ears and a tail etc.) — a giant leap from today’s “black box” systems. Third wave -> (XAI).

Third (future) AI wave : Context



Contextual Adaptation

- To remove bottlenecks of techniques of second wave of AI. Research is at beginning stages.
- Third wave: systems construct explanatory models that allow them to characterize real-world phenomena.
  - **Example:** Third Wave AI will not only recognize the "cat", but will be able to explain why it's a cat and how it arrived at that conclusion (i.e. has a fur, two ears and a tail etc.) — a giant leap from today's "black box" systems. Third wave -> (XAI).
- It does not take much imagination to envision the tremendous possibilities of Third Wave AI. Some under development products:
  - Pandai
  - Aigo

Section Contents

Intuition

Have you ever thought?

- How easily we recognize face, color, shape or handwritten characters.
- How children learn to balance or develop preference to some taste.

[4] Cecilia Heyes, New thinking: the evolution of human cognition, Philosophical Transactions of the Royal Society 2012.

Intuition

Have you ever thought?

- How easily we recognize face, color, shape or handwritten characters.
- How children learn to balance or develop preference to some taste.

- Human's cognitive abilities have transformed every aspect of our lives.
- Human mind is a set of cognitive gadgets, specialized to learn. [4]

[4] Cecilia Heyes, New thinking: the evolution of human cognition, Philosophical Transactions of the Royal Society 2012.

Intuition


How children learn?
• No explicit features identification given.
• They learn from experience.
• Eyes take image every 200 ms (Saccade and fixation, 5 pictures / second) (300 pictures / minute).
• Enormous amount of data given as input (ages - >).

Intuition

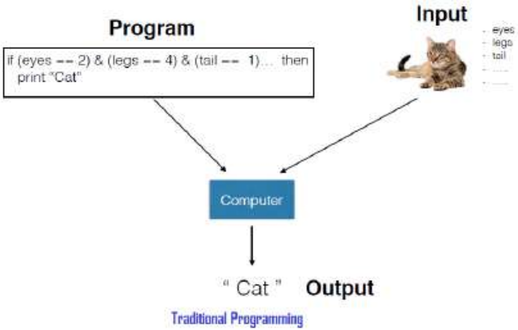
Humans learn from experience!

How Machine Learning is different from Traditional Programming?

Activity

Write a program (pseudo-code) to identify "cat" in an image

How Machine Learning is different from Traditional Programming?



Traditional programing

for $j = 1$ to N **do**
  detect color ($image_N$)
  lots of code
end for

for $j = 1$ to N **do**
  detect shape ($image_N$)
  lots of code
end for

for $j = 1$ to N **do**
  detect fur ($image_N$)
  lots of code
end for

.

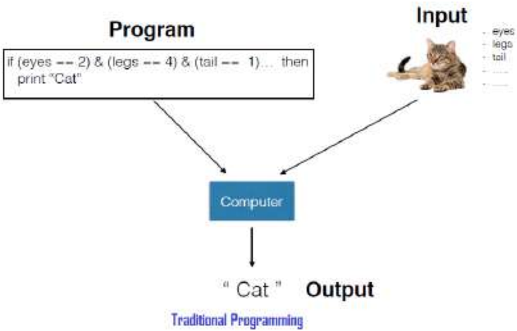
.

Is this enough to recognize cat?

Note[5]

[5] courtesy: Prof. Fei-Fei Li (Stanford)

How Machine Learning is different from Traditional Programming?

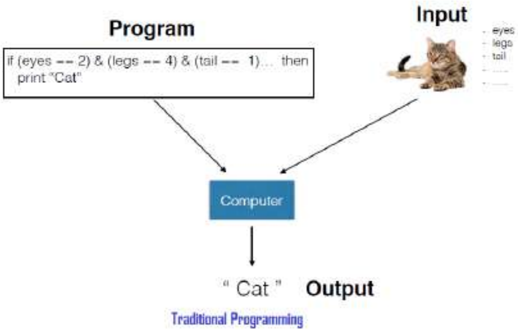


Can we manually write an algorithm (hard code) that caters all the variations?

Note[5]

[5] courtesy: Prof. Fei-Fei Li (Stanford)

How Machine Learning is different from Traditional Programming?

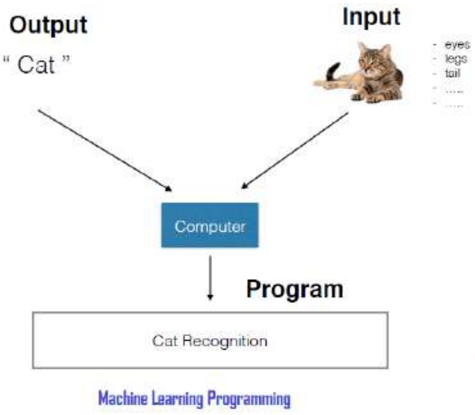


Note[5]

Can we manually write an algorithm (hard code) that caters all the variations?

[5] courtesy: Prof. Fei-Fei Li (Stanford)

How Machine Learning is different from Traditional Programming?



Note[5]

- Machine learning algorithms are algorithms that learn models from data / experience.
- No need to formulate explicit rules.
- Algorithm performance gets better with experience / data.

[5] courtesy: Prof. Fei-Fei Li (Stanford)

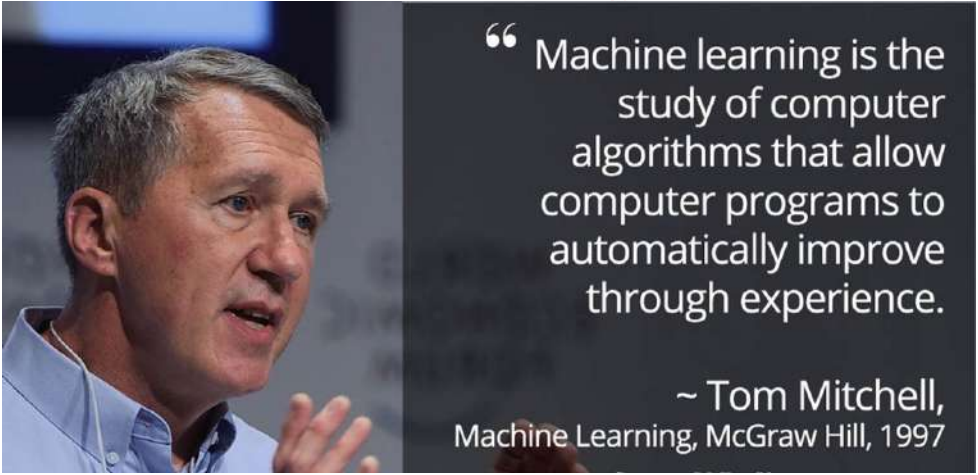
What is Machine learning?

Machine Learning

Field of study that gives computers the ability to learn without being explicitly programmed.

Arthur Samuel, 1959

What is Machine learning?



What is Machine learning?

ML in a Nutshell

A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.

Example:

- Task T : Recognize human face
- Performance measure P : Accuracy of prediction
- Experience E : Dataset of human faces

What is Machine learning?

ML in a Nutshell

A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.

Example:
- Task T : Recognizing hand-written words
- Performance measure P : Percentage of words correctly classified
- Experience E : Database of human-labeled images of handwritten words

What is Machine learning?

ML in a Nutshell

A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.

Example:

- Task T : Categorize email messages as spam or legitimate
- Performance measure P : Percentage of email messages correctly classified
- Experience E : Database of emails, some with human-given labels

What is Machine learning?

ML in a Nutshell

A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.

Example:

- Task T : Categorize X-ray image having lung disease
- Performance measure P : Percentage of X-ray images correctly classified
- Experience E : Database of X-ray image with domain expert labels

What is Machine learning?

- Machine Learning algorithms ingest data and learn a model (hypothesis).
- The learned model can be used to:
  1. Detect pattern / trends / structures etc. from the data
  2. Make predictions on unseen / new data

Why Machine Learning?

Machine Learning is used when:

- Humans can't explain their expertise:
  - speech recognition

Why Machine Learning?

Machine Learning is used when:

- Humans can't explain their expertise:
  - speech recognition
  - visual recognition

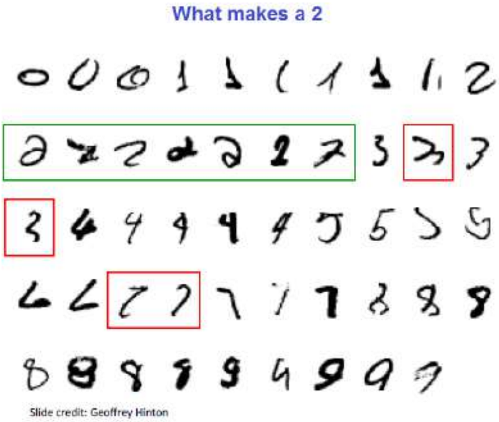
Why Machine Learning?

Detected faces

Machine Learning is used when:

- Humans can't explain their expertise:
  - speech recognition
  - visual recognition
  - face detection, expressions recognition

Why Machine Learning?



Machine Learning is used when:

- Humans can't explain their expertise:
  - speech recognition
  - visual recognition
  - face detection, expressions recognition
  - hand writing recognition (pattern recog.) etc

Why Machine Learning?

Machine Learning is used when:

- Humans can't explain their expertise:
  - speech recognition
  - visual recognition
  - face detection, expressions recognition
  - hand writing recognition (pattern recog.) etc
- Models must be customized (personalized medicine, personalized recommendations, home assistant)

Why Machine Learning?

Machine Learning is used when:

- Humans can't explain their expertise:
  - speech recognition
  - visual recognition
  - face detection, expressions recognition
  - hand writing recognition (pattern recog.) etc
- Models must be customized (personalized medicine, personalized recommendations, home assistant)
- Models are based on huge amounts of data
  - genomics
  - stock prices

Why Machine Learning?

Machine Learning is used when:

- Humans can't explain their expertise:
  - speech recognition
  - visual recognition
  - face detection, expressions recognition
  - hand writing recognition (pattern recog.) etc
- Models must be customized (personalized medicine, personalized recommendations, home assistant)
- Models are based on huge amounts of data
  - genomics
  - stock prices
  - self driving cars etc.

Why Machine Learning?

Machine Learning is used when:

- Humans can't explain their expertise:
    - speech recognition
    - visual recognition
    - face detection, expressions recognition
    - hand writing recognition (pattern recog.) etc
- Models must be customized (personalized medicine, personalized recommendations, home assistant)
- Models are based on huge amounts of data
    - genomics
    - stock prices
    - self driving cars etc.
- Human expertise does not exist (Mars navigating)

Why Machine Learning?

Machine Learning is used when:

- Humans can't explain their expertise:
    - speech recognition
    - visual recognition
    - face detection, expressions recognition
    - hand writing recognition (pattern recog.) etc
- Models must be customized (personalized medicine, personalized recommendations, home assistant)
- Models are based on huge amounts of data
    - genomics
    - stock prices
    - self driving cars etc.
- Human expertise does not exist (Mars navigating)
- Human capabilities needs to be augmented (medical diagnosis)

Why ML is growing?

ML Niche

Why ML is growing?

Why ML is growing?

ML Niche

Why ML is growing?

1. ML is preferred approach to:

Why ML is growing?

ML Niche

Why ML is growing?

1. ML is preferred approach to:

- Medical imaging

Why ML is growing?

ML Niche

Why ML is growing?

1. ML is preferred approach to:

- Medical imaging
- Speech recognition

Why ML is growing?

ML Niche

Why ML is growing?

1. ML is preferred approach to:

- Medical imaging
- Speech recognition
- Robotics

Why ML is growing?

ML Niche

Why ML is growing?

1. ML is preferred approach to:

- Medical imaging
- Speech recognition
- Robotics
- Computer vision:

Why ML is growing?

ML Niche

Why ML is growing?

1. ML is preferred approach to:

- Medical imaging
- Speech recognition
- Robotics
- Computer vision:
  1. Person identification
  2. Activity recognition
  3. Object detection
  4. Autonomus driving
  5. ....
- ...

2. ML is preferred approach to all of the above problems and:

- Improved ML algorithms

Why ML is growing?

ML Niche

Why ML is growing?

1. ML is preferred approach to:

- Medical imaging
- Speech recognition
- Robotics
- Computer vision:
  1. Person identification
  2. Activity recognition
  3. Object detection
  4. Autonomus driving
  5. ....
- ...

2. ML is preferred approach to all of the above problems and:

- Improved ML algorithms
- Availability of large volumes of datasets

Why ML is growing?

ML Niche

Why ML is growing?

1. ML is preferred approach to:

- Medical imaging
- Speech recognition
- Robotics
- Computer vision:
  1. Person identification
  2. Activity recognition
  3. Object detection
  4. Autonomus driving
  5. ....
- ...

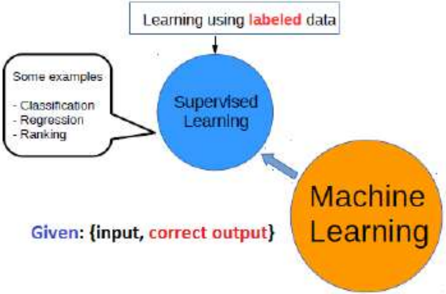
2. ML is preferred approach to all of the above problems and:

- Improved ML algorithms
- Availability of large volumes of datasets
- Self customizing software i.e. Speech recognition or Spam filter

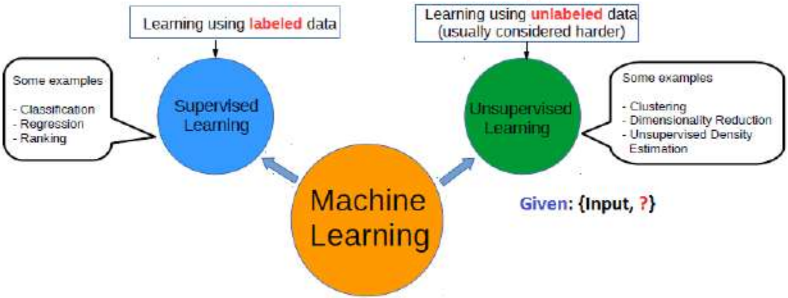
Section Contents

Taxonomy of Machine learning

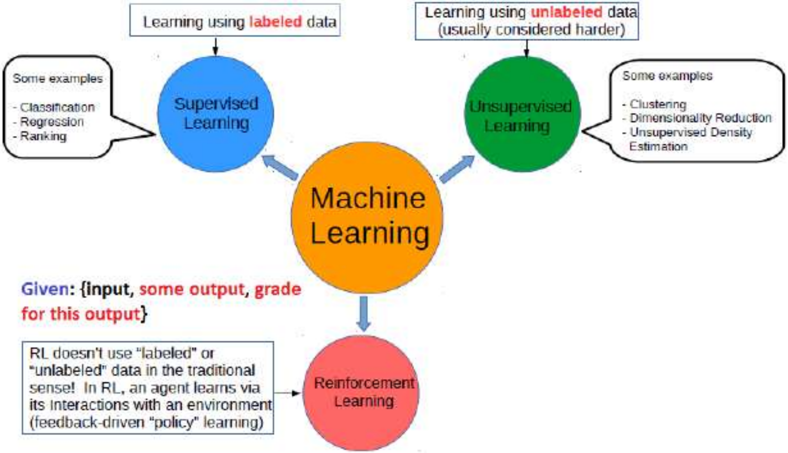
Taxonomy of Machine learning



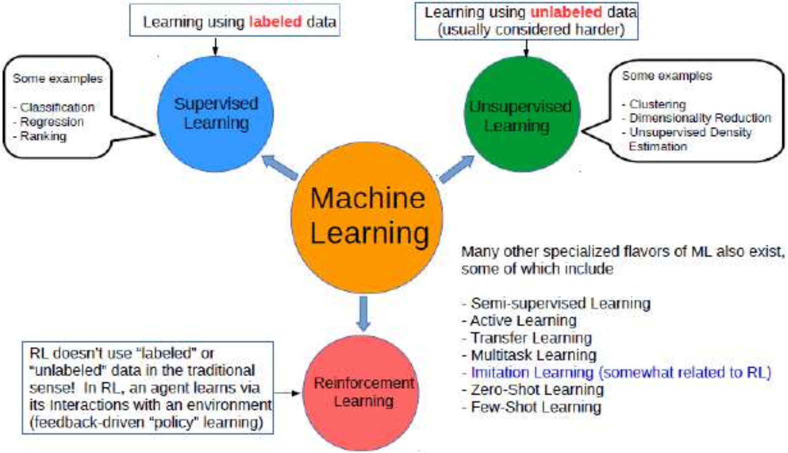
Taxonomy of Machine learning



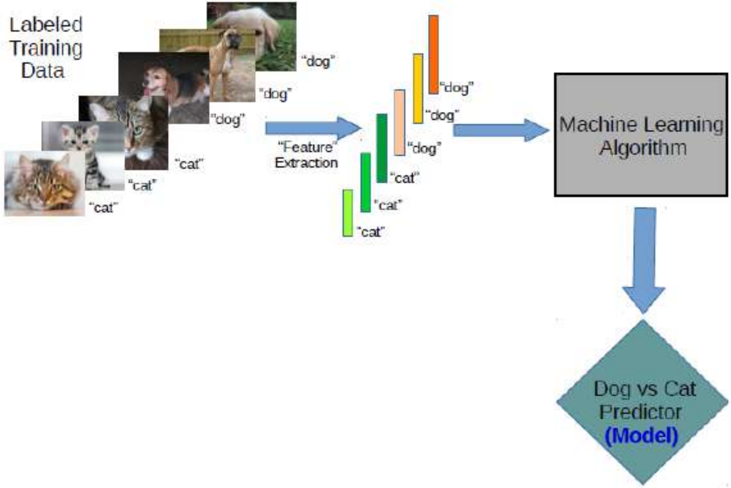
Taxonomy of Machine learning



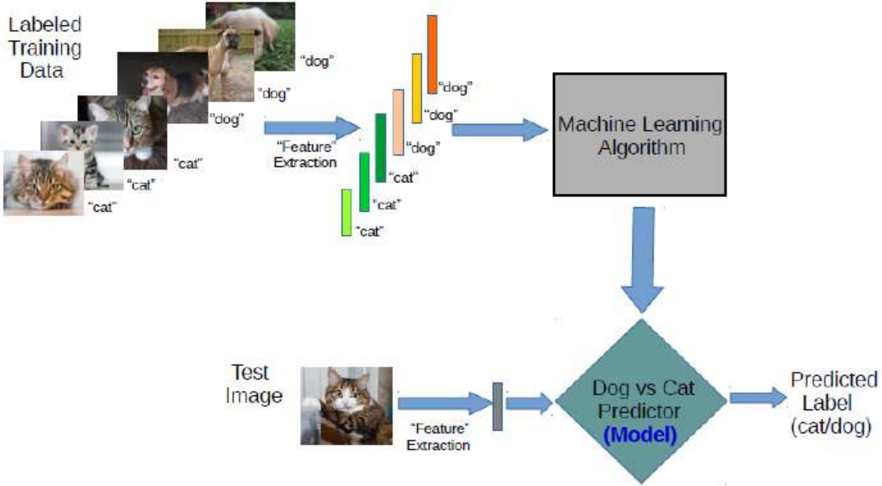
Taxonomy of Machine learning



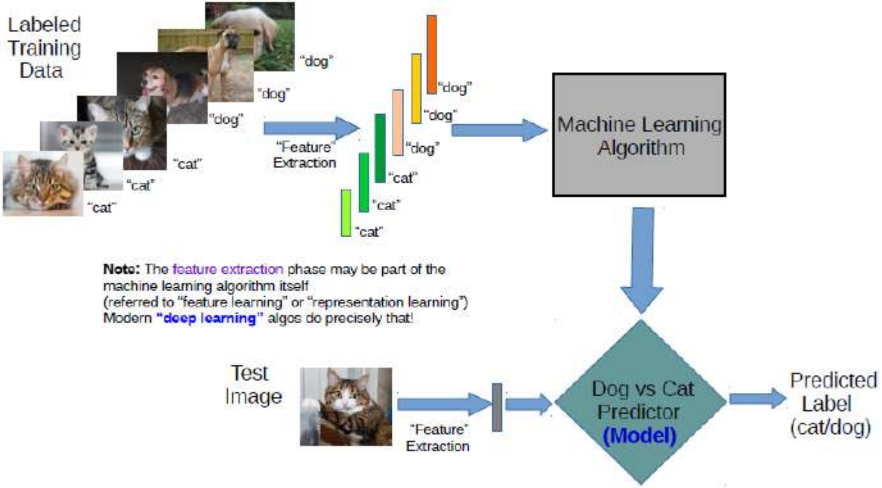
Supervised Learning Workflow for classification



Supervised Learning Workflow for classification



Supervised Learning Workflow for classification



Function approximation

Supervised learning is about function approximation

Problem Setting:

- Set of possible instances X
- Unknown target function $f : X \to Y$
- Set of function hypotheses $H = \{h | h : X \to Y\}$

Input:

- training examples $\{< x_i, y_i >\}$. For example x is an email and y is either Spam or No Spam.

Output:

- Hypothesis $h \in H$ that best approximates target function f. OR
- a classification "rule" that can determine the class of any object from its attributes values.

Inductive Learning

Example

Input	1	2	3	4	5	6	7
Output	1	4	9	16	25	36	??

Inductive Learning

Example

Input	1	2	3	4	5	6	7
Output	1	4	9	16	25	36	??

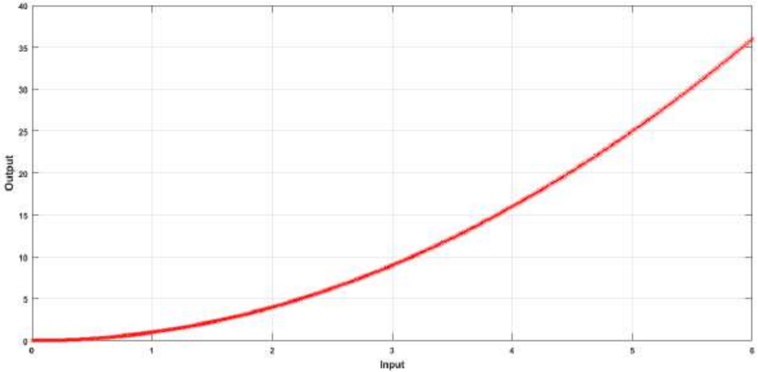
- $f : X^2 \rightarrow Y$ OR
- $f : input^2 \rightarrow output$

Inductive Learning

Example

Input	1	2	3	4	5	6	7
Output	1	4	9	16	25	36	??

- $f : X^2 \rightarrow Y$ OR
- $f : input^2 \rightarrow output$

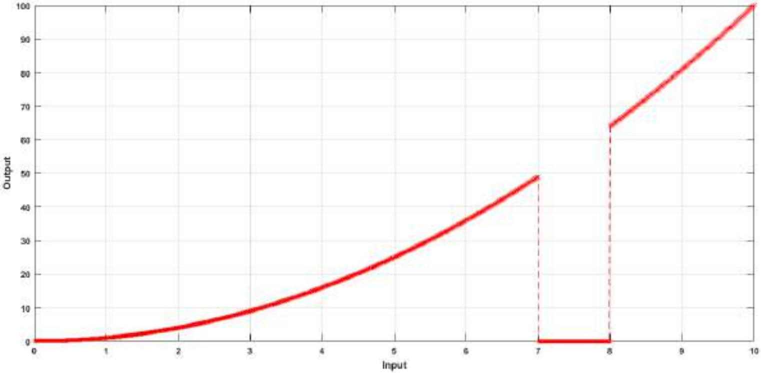


Inductive Learning

Example

Input	1	2	3	4	5	6	7
Output	1	4	9	16	25	36	??

- $f : X^2 \rightarrow Y$ OR
- $f : input^2 \rightarrow output$

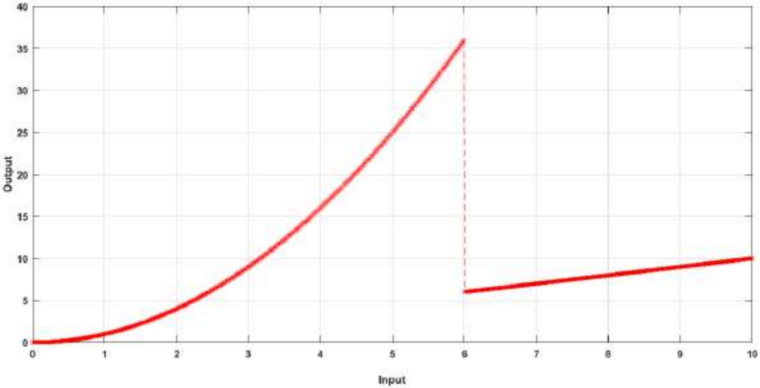


Inductive Learning

Example

Input	1	2	3	4	5	6	7
Output	1	4	9	16	25	36	??

- $f : X^2 \rightarrow Y$ OR
- $f : input^2 \rightarrow output$



Inductive Learning

Example

Input	1	2	3	4	5	6	7
Output	1	4	9	16	25	36	??

Inductive Learning

Example

Input	1	2	3	4	5	6	7
Output	1	4	9	16	25	36	??

- $f : X^2 \rightarrow Y$ OR
- $f : input^2 \rightarrow output$

Inductive Learning

Example

Input	1	2	3	4	5	6	7
Output	1	4	9	16	25	36	??

- $f : X^2 \rightarrow Y$ OR
- $f : input^2 \rightarrow output$

Guarantee

What if function is not well behaved? What if everything squared up to 6?

Inductive Learning

Example

Input	1	2	3	4	5	6	7
Output	1	4	9	16	25	36	??

- $f : X^2 \rightarrow Y$ OR
- $f : input^2 \rightarrow output$

Guarantee

What if function is not well behaved? What if everything squared up to 6?

Fundamental assumption:

- Function is well behaved and consistent with the data
- Generalize (induction)

Inductive Learning

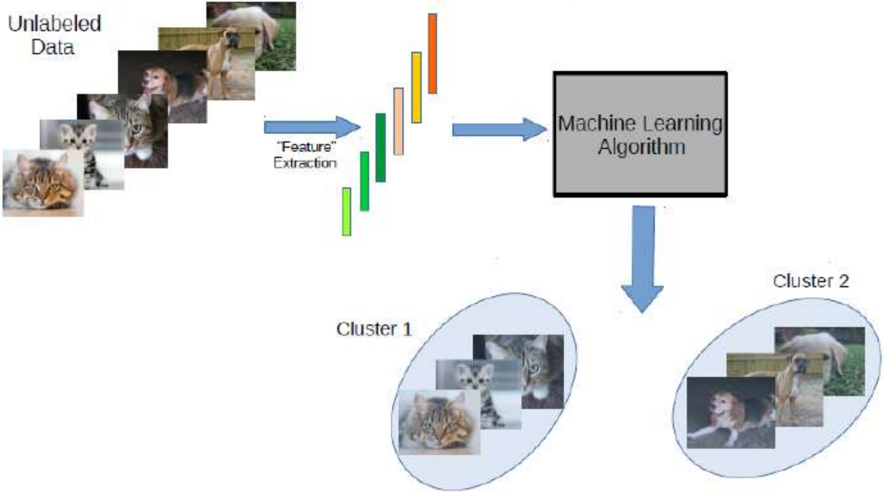
- $Specifics \rightarrow generality$
- $Examples/observed\,instances \rightarrow general\,rules$

Inductive Learning

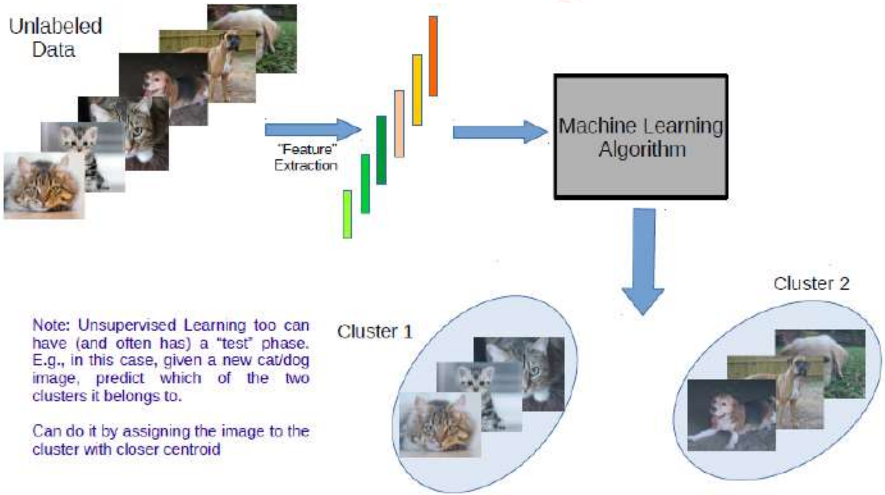
- $Specifics \rightarrow generality$
- $Examples/observed\, instances \rightarrow general\, rules$

Supervised learning is about function approximation or induction of approximate function.

Unsupervised Learning Workflow for clustering

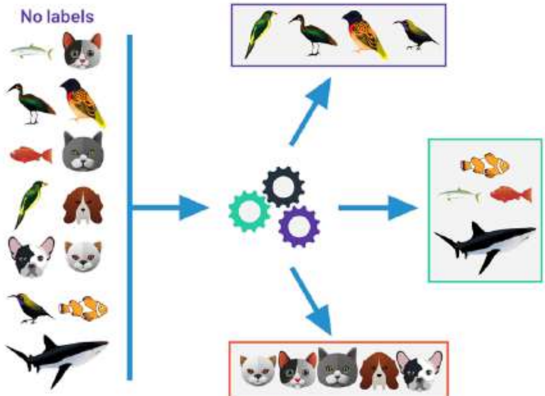


Unsupervised Learning Workflow for clustering



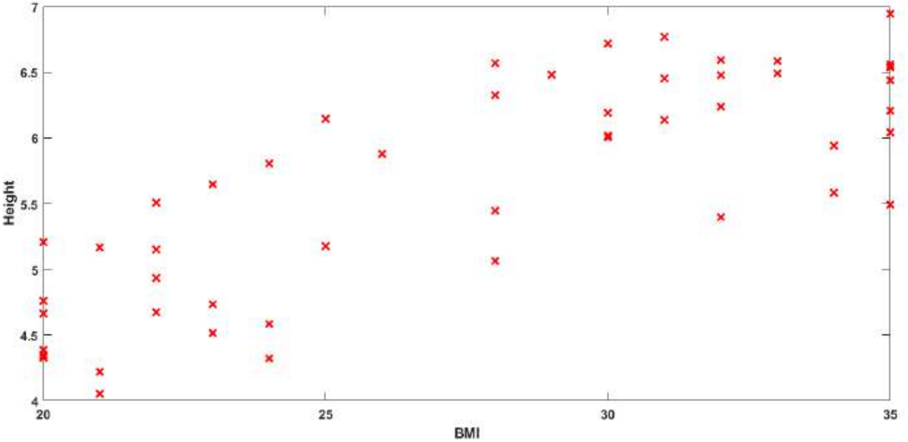
Deductive Learning

Unsupervised learning is about description, opposed to approximation (supervised learning).



Deductive Learning

Unsupervised learning is about description, opposed to approximation (supervised learning).



Deductive Learning

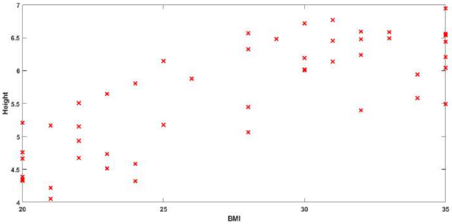
- Unlabeled data / examples

Deductive Learning

- Unlabeled data / examples
- Derive structure from the data by looking at relationship b/w input examples

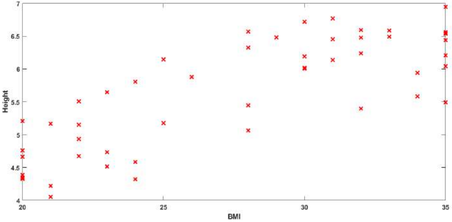
Deductive Learning

- Unlabeled data / examples
- Derive structure from the data by looking at relationship b/w input examples



Deductive Learning

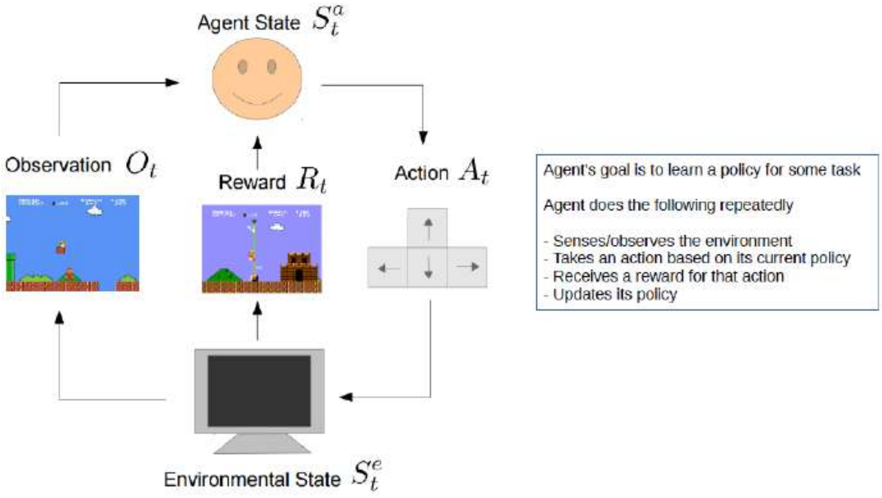
- Unlabeled data / examples
- Derive structure from the data by looking at relationship b/w input examples



- Unsupervised learning is about description

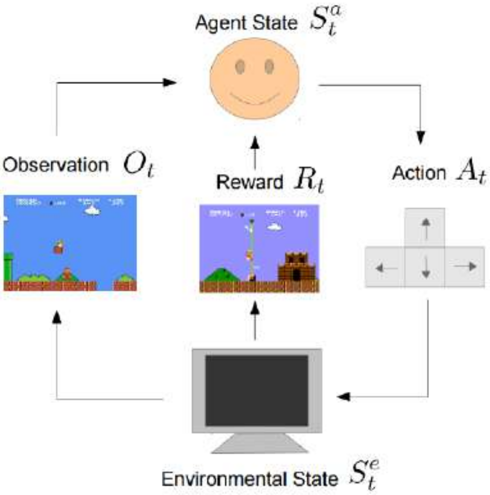
Reinforcement Learning Workflow

- Learning from delayed reward



Reinforcement Learning Workflow

- Learning from delayed reward



Agent's goal is to learn a policy for some task

Agent does the following repeatedly

- Senses/observes the environment
- Takes an action based on its current policy
- Receives a reward for that action
- Updates its policy

There is supervision, not explicit (as in Supervised Learning) but rather implicit (feedback based)

Section Contents

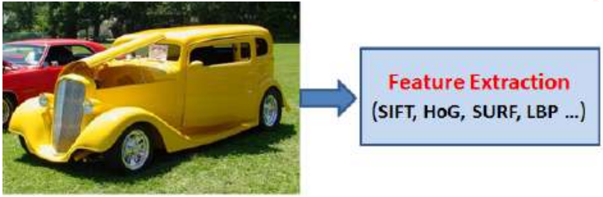
Features or attributes

Traditional Workflow for classification

[1]Bishop, Christopher (2006). Pattern recognition and machine learning

Features or attributes

Traditional Workflow for classification



Feature is an individual measurable property or characteristic of a phenomenon being observed[1].

[1] Bishop, Christopher (2006). Pattern recognition and machine learning

Features or attributes

Traditional Workflow for classification

Feature is an individual measurable property or characteristic of a phenomenon being observed[1].

[1] Bishop, Christopher (2006). Pattern recognition and machine learning

Features or attributes

Traditional Workflow for classification

Feature is an individual measurable property or characteristic of a phenomenon being observed[1].

[1]Bishop, Christopher (2006). Pattern recognition and machine learning

Features or attributes

Traditional Workflow for classification

Feature is an individual measurable property or characteristic of a phenomenon being observed[1].

[1]Bishop, Christopher (2006). Pattern recognition and machine learning

Features: Toy Example

Toy Example

What features can differentiate between Apple and Oranges, consider different color variations.

Features: Toy Example

Toy Example

What features can differentiate between Apple and Oranges, consider different color variations.

Features: Toy Example

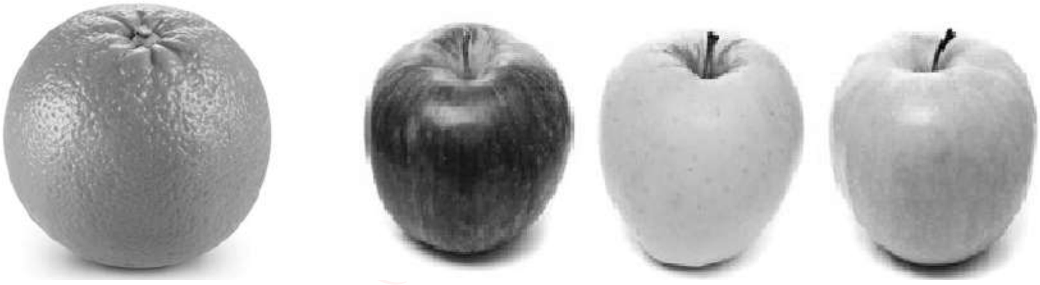
Toy Example

What features can differentiate between Apple and Oranges, consider different color variations.

Features: Toy Example

Toy Example

What features can differentiate between Apple and Oranges, consider different color variations.



Features quality

Feature is an individual measurable property or characteristic of a phenomenon being observed.

Fundamental question

What are good features?

Features quality

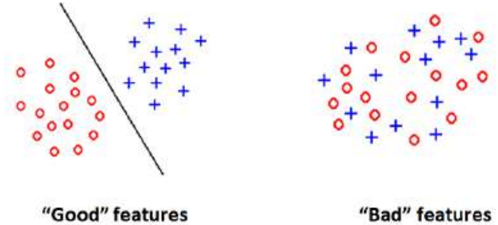
Feature is an individual measurable property or characteristic of a phenomenon being observed.

Fundamental question

What are good features?

Good feature

Good features makes it easy for classifier to decide (learn) between two different classes / concepts / labels OR good features enhances inter class variations while minimize intra class varaition.



Features: Toy Example

Coming back to toy example. What are good features (individual measurable property or characteristic) to learn concept of “Apple” and “Orange” ?

In (supervised)Machine Learning algorithm (more on this):

- Input is set of features and label / class.
- Output is set of rules or pattern related to specific class. Simply output is trained Classifier or Decision Surface
- Classifier is function $f : X \rightarrow Y$

First ML code: Toy Example

Remember this!

First ML code: Toy Example

Remember this!

Steps:

1. Collect training data (features extraction)

First ML code: Toy Example

Remember this!

Steps:

1. Collect training data (features extraction)
2. Train classifier

First ML code: Toy Example

Remember this!

Steps:

1. Collect training data (features extraction)
2. Train classifier
3. Predict new data

First ML code: Toy Example

Training Data / Features extracted from real data

Weight	**Texture**	**Class**
150g	Bumpy	Orange
170g	Bumpy	Orange
140g	Smooth	Apple
130g	Smooth	Apple
..	..	..
..	..	..

First ML code: Toy Example

Training Data / Features extracted from real data

Weight	Texture	Class
150g	Bumpy	Orange
170g	Bumpy	Orange
140g	Smooth	Apple
130g	Smooth	Apple
..	..	..
..	..	..

1. Each row in training data is an example (Feature extractor Algorithm)
2. Last column is class / label
3. Train classifier (ML Algorithm) - More data, better classifier training!
4. Predict new data

First ML code: Toy Example

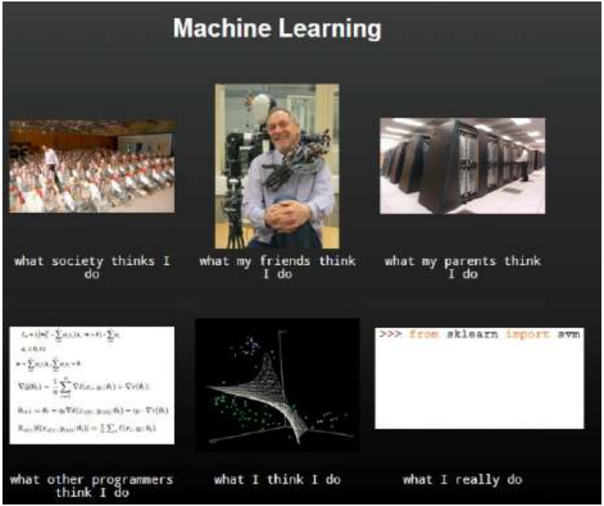
```
from sklearn import tree
#features=[[140, "smooth" ],[130, "smooth"],[150,"bumpy" ], [170, "bumpy" ]]
#labels=["apple","apple","orange","orange"]

#sklearn uses real-valued features

features=[[140, 1 ],[130, 1],[150,0 ], [170, 0 ]]
labels=[0,0,1,1]

#Train Classifier - Decision Tree
clf = tree.DecisionTreeClassifier()
clf=clf.fit(features,labels) #Classifier is trained on our data

#Predict
print (clf.predict([[140,0]]))
```

Expectation from ML Specialist?

Previous example has six lines of code!

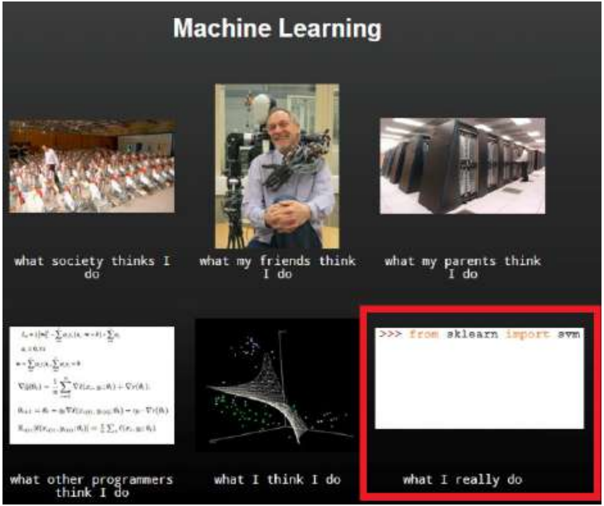
Expectation from ML Specialist?

Previous example has six lines of code!



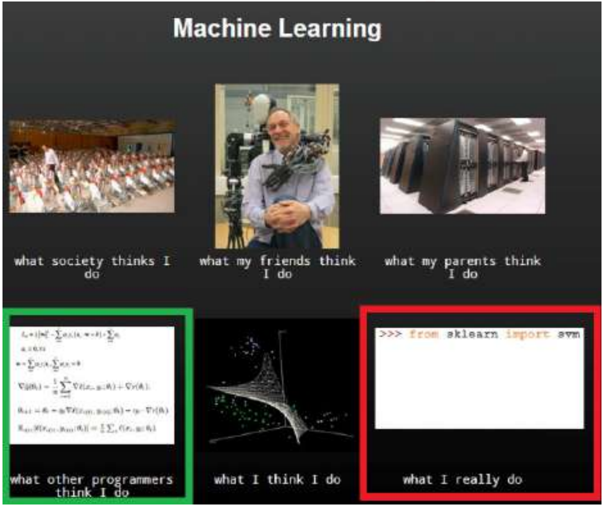
Expectation from ML Specialist?

Previous example has six lines of code!



Expectation from ML Specialist?

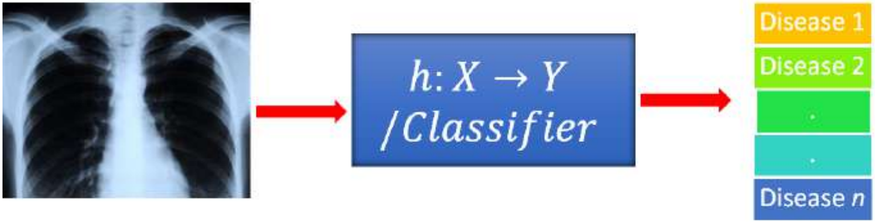
Previous example has six lines of code!



Section Contents

Examples

- In practice it is almost always too hard to estimate the function, so we are looking for very good approximations of the function.



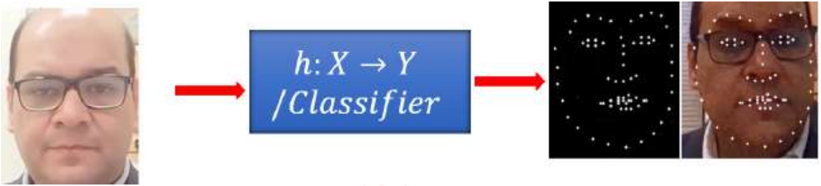
- Some practical examples of (supervised learning) are:

Disease diagnosis

- The X are the properties of the patient.
- The $f(X)$ is the disease they suffer from.

Examples

- In practice it is almost always too hard to estimate the function, so we are looking for very good approximations of the function.



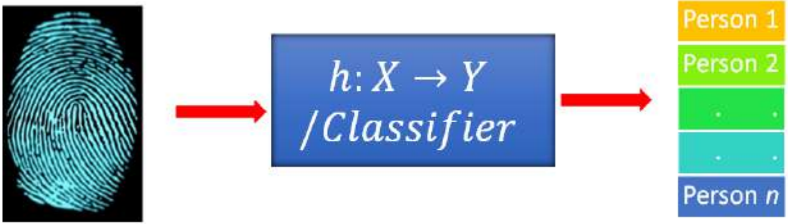
- Some practical examples of (supervised learning) are:

Person identification

- The X are images of face.
- The $f(X)$ is the identified person.

Examples

- In practice it is almost always too hard to estimate the function, so we are looking for very good approximations of the function.



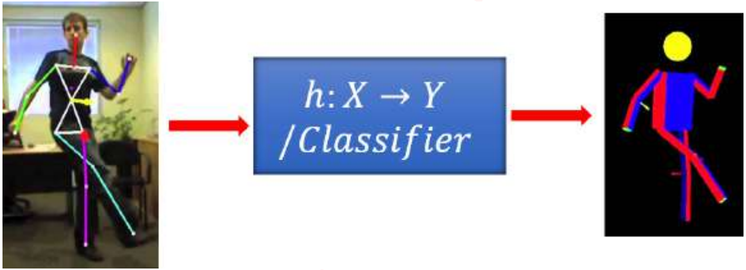
- Some practical examples of (supervised learning) are:

Person identification / Biometric

- The X are finger.
- The $f(X)$ is the identified person.

Examples

- In practice it is almost always too hard to estimate the function, so we are looking for very good approximations of the function.



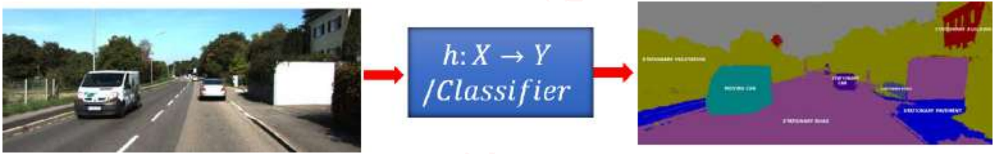
- Some practical examples of (supervised learning) are:

Posture Analysis

- The X are images with different postures.
- The $f(X)$ is the recognized posture / activity.

Examples

- In practice it is almost always too hard to estimate the function, so we are looking for very good approximations of the function.



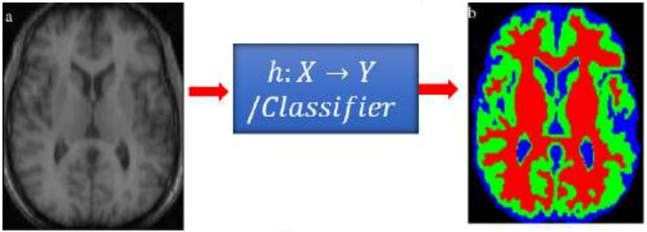
- Some practical examples of (supervised learning) are:

Semantic Scene Analysis

- The X are images.
- The $f(X)$ is the recognized label for each pixel.

Examples

- In practice it is almost always too hard to estimate the function, so we are looking for very good approximations of the function.



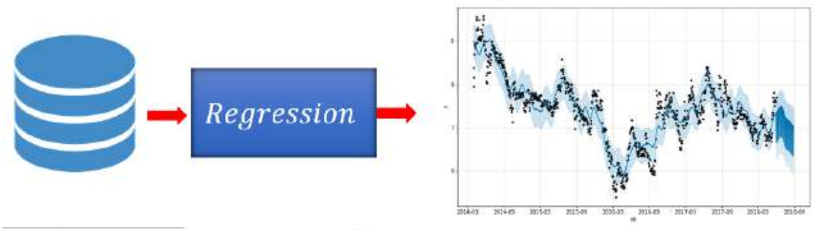
- Some practical examples of (supervised learning) are:

Medical Image segmentation

- The X are images coming from different modalities.
- The $f(X)$ is the segmented images with clear boundaries.

Examples

- In practice it is almost always too hard to estimate the function, so we are looking for very good approximations of the function.



- Some practical examples of (supervised learning) are:
- **This is Regression**. i.e. real-valued output.

Stock Price Prediction

- The X data recorded for t time.
- The $f(X)$ is prediction.

Golden Words