Python for Data Analytics

By
M. Vamsee Krishna Kiran
Assistant Professor, Dept of Computer Science,
Amrita School of Engineering.

Agenda

1. Python for Data analysis
    a. Why Python for Data analysis
    b. Other languages for data analysis
    c. Python Installation and Notebooks
    d. Python concepts needed before jumping to data analysis
2. Python Libraries for Data analysis
3. Understanding Pandas for Dataset loading
4. Building Machine Learning Models
    a. Linear Regression
    b. Logistic Regression
    c. Naive Bayes
    d. K-Means clustering
    e. Decision trees
    f. SVM
5. Conclusion

Why Python for Data Analytics

- Open Source
- Python Has a Healthy, Active and Supportive Community
- Easy to learn
- Python Has Big Data
- Python Has Amazing Libraries

Other languages for Data Analysis

1. Python
2. Java
3. Matlab
4. R
5. Julia
6. Scala etc.

Python + packages Installation

1. Download python from https://www.python.org/downloads/ and install.
2. Pip installer will come with python installation. It will be available in Python installation **folder/scripts**.
3. Navigate to that path in the command prompt and execute the command **pip install numpy**
4. To install matplotlib execute command **pip install matplotlib**.
5. To install using a single command: **python -m pip install --user numpy scipy matplotlib ipython jupyter pandas sympy nose**
6. To install on ubuntu s**udo apt-get install python-numpy python-scipy python-matplotlib ipython ipython-notebook python-pandas python-sympy python-nose**

Installing using Anaconda (Best installation method - single shot)
https://www.anaconda.com/distribution/

Jupyter Notebooks

1. https://notebooks.azure.com/
2. https://cocalc.com/
3. https://colab.research.google.com/
4. https://paiza.cloud/en/jupyter-notebook-online
5. https://jupyter.org/try

Advantages of Jupyter Notebooks:

- Run/re-run individual snippets
- Editing code made simple.
- Share your live code with your peers.
- Runs on cloud.
- Best suitable for interactive research sessions.

Python prerequisites for Data Analytics

- Lists
- Tuples
- Dictionaries
- Sets
- Strings
- Importing libraries
- Iterations and Conditional Statements

Please open the Python Basics Jupyter Notebook for practice

Python Libraries

NUMPY - Numerical Python

NumPy is the fundamental package for scientific computing with Python:

- a powerful N-dimensional array object
- Working on huge matrices and support for Big-data.
- tools for integrating C/C++ and Fortran code
- useful linear algebra, Fourier transform, and random number capabilities

Please use the Numpy-basics Jupyter Notebook for practice.

Pandas for Data Manipulation and Analysis

Python package providing fast, flexible, and expressive data structures designed to make working with structured (tabular, multidimensional, potentially heterogeneous) data.

Primary Data Structures in Pandas:

1. Series (1D)
2. Data frames (2D)

Functionalities:

- Easy handling of missing data (represented as NaN)
- Size mutability
- Intelligent label-based slicing, fancy indexing
- Intuitive merging and joining data sets
- Flexible reshaping and pivoting of data sets
- Automatic and explicit data alignment

Please use the Pandas-basics Jupyter Notebook for practice.

SCIPY - Scientific Python

- SciPy uses NumPy arrays as the basic data structure
- Comes with modules for various commonly used tasks in scientific programming, including **linear** algebra, integration (calculus), ordinary differential equation solving, and signal processing.

Sci-kit Learn (Machine Learning in Python)

- Machine Learning Library for Python
- Supports various Supervised and unsupervised machine learning algorithms like Linear regression, logistic regression, SVM, K-Means clustering, K-NN, Decision Trees, Random forests, Naive Bayes classifier etc.
- Sci-kit learn supports both numpy and scipy packages.

Please look at the cheatsheet of Sci-kit learn for various code snippets.

Matplotlib(Plotting) & Seaborn(Statistical Visualization)

- Matplotlib is a python library used to create 2D graphs
- Supports histogram, bar charts, power spectra, error charts etc
- open source alternative for MatLab

- Seaborn is a Python data visualization library based on matplotlib.
- high-level interface for drawing attractive and informative statistical graphics.

Machine Learning Models

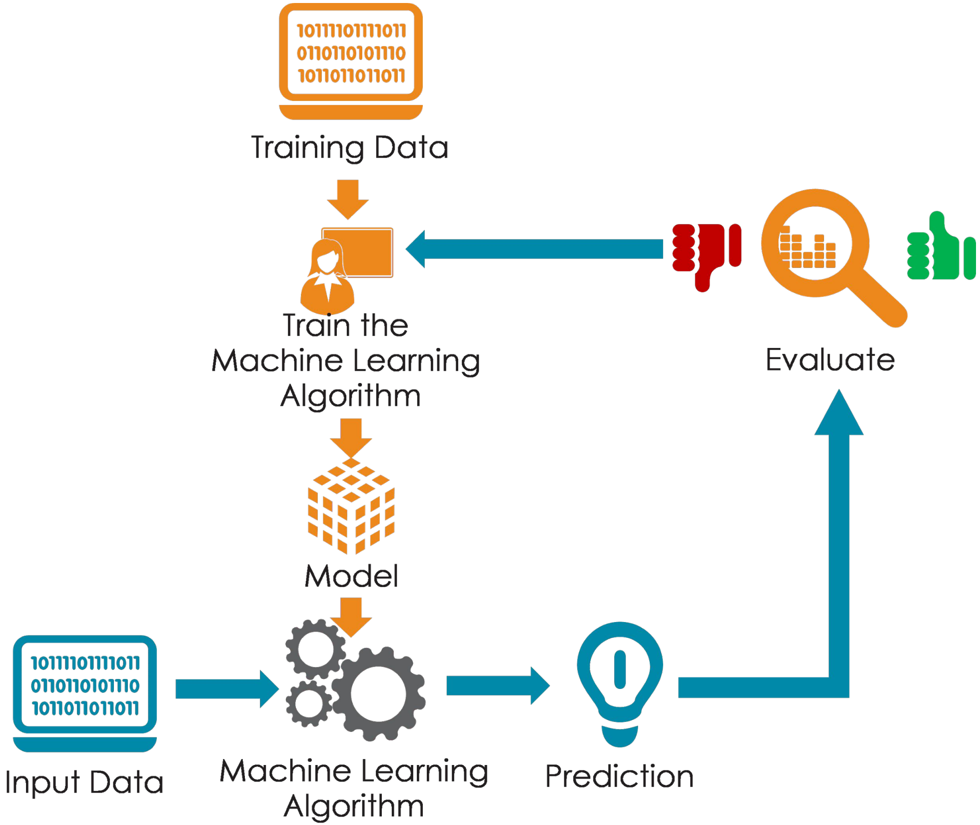
Machine Learning

Please give some real time examples.

Every machine learning algorithm consists of three parts:

1. Representation
2. Evaluation
3. Optimization

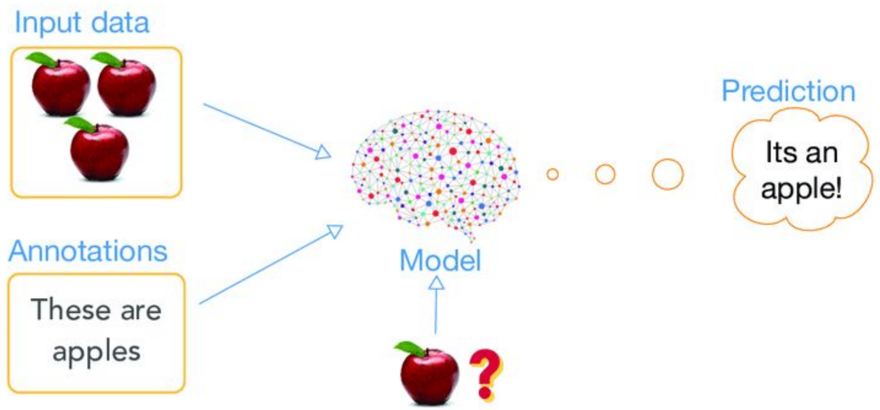
General workflow



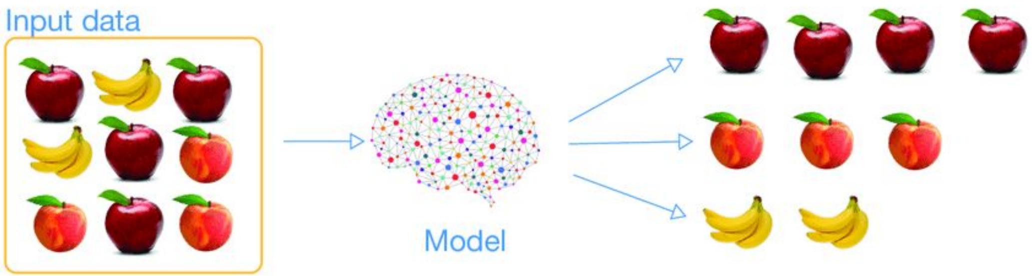
Types of Machine learning algorithms

1. Supervised - with class labels
2. Unsupervised - without class labels
3. Semi supervised - partial data has class labels
4. Reinforcement learning - reward for each action

supervised learning



unsupervised learning



Supervised Learning

Classification Algorithms: discrete output. (0/1, phishing/not-phishing, rain/no-rain)

- Is classification algorithms always predict binary outputs?
- Data will contain features and class labels
- What are features and labels?

Regression Algorithms: continuous output. (rainfall, earthquake intensity, house price, doctor fee)

- Data will contain features and class labels, where labels are numerical values.

Example Algorithms: Linear Regression, Logistic Regression, Naive Bayes, KNN, Decision Tree, SVM, Random Forests etc.

Unsupervised Learning

Clustering Algorithms: (No labels for the data)

- Clusters or groups the data based on some measure usually distance measure like euclidean.

Example Algorithms:

- K-means clustering where k is the number of centroids.
- Association Analysis
- Apriori Algorithm
- Agglomerative clustering

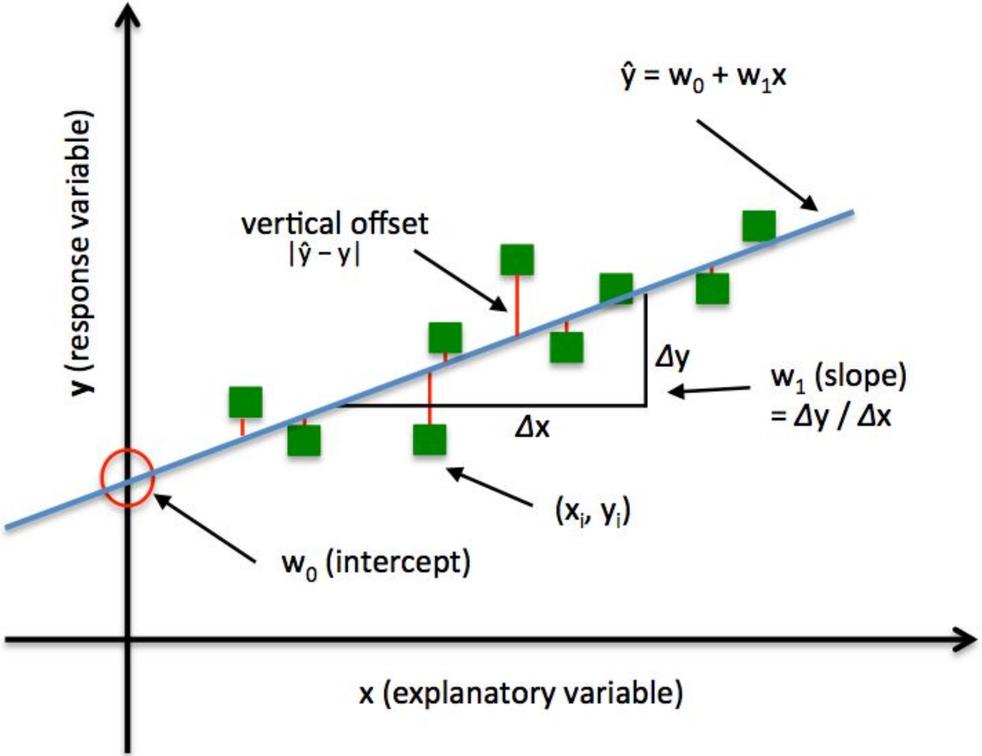
Reinforcement learning

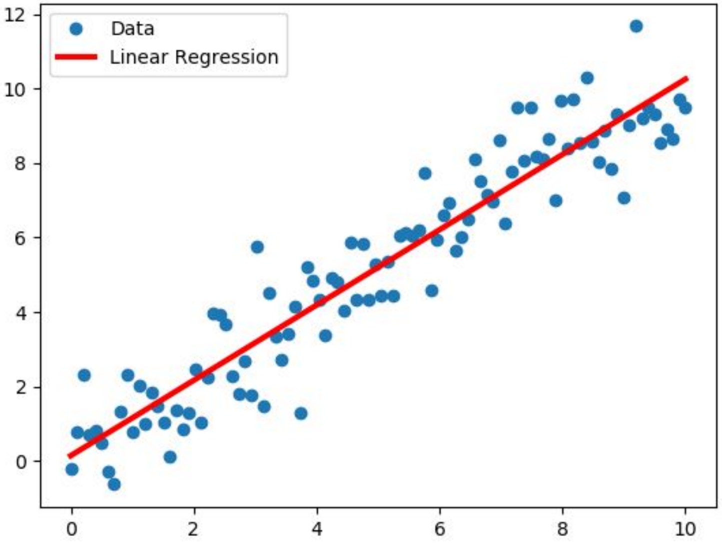
- Class label may be present may not be present.
- Algorithm learns about the environment and performs classification. Gets rewarded if the classification is correct else, penalized.

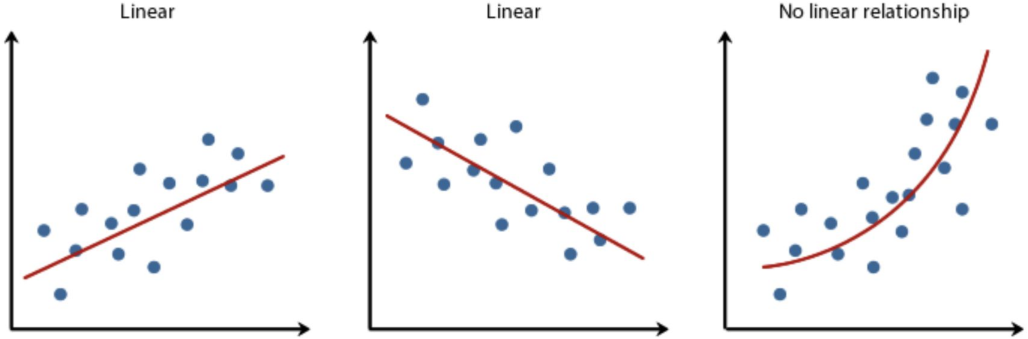
Example algorithms:

- Markov chain process (self driving cars, robot path planning, AlphaGo)
- Q Learning

Linear Regression




Data
Linear Regression
12
10
8
6
4
2
0
0
2
4
6
8
10


Linear
Linear
No linear relationship

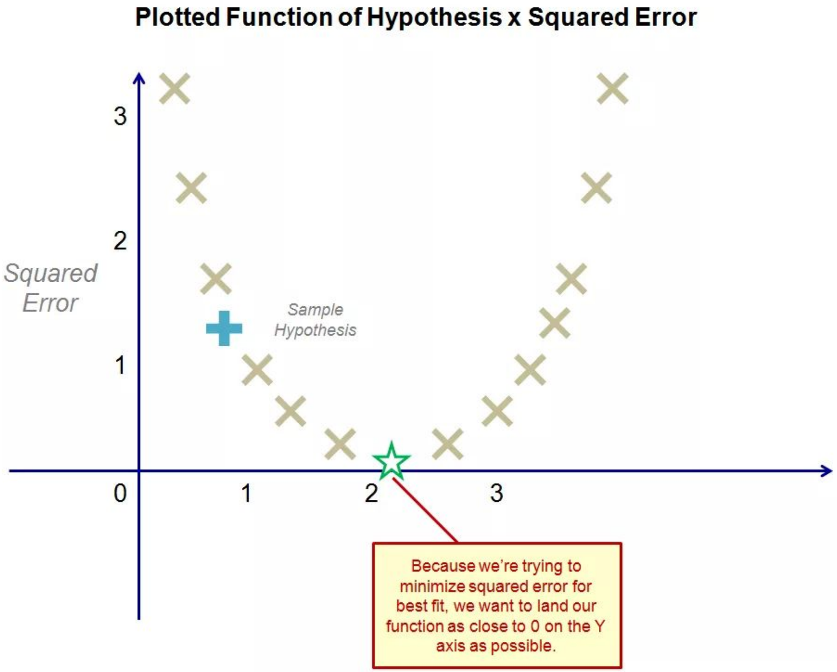
Mean Squared Error

$$MSE = \frac{1}{N} \sum_{i=1}^{N} (f_i - y_i)^2$$

where N is the number of data points,
f_i the value returned by the model and
y_i the actual value for data point i.

Cost function

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$



Overall picture

Hypothesis: $h_\theta(x) = \theta_0 + \theta_1 x$

Parameters: θ_0, θ_1

Cost Function: $J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$

Goal: $\underset{\theta_0, \theta_1}{\text{minimize}}\ J(\theta_0, \theta_1)$

Gradient Descent

Automatic algorithm to update the theta values, helps in convergence i.e reaching global optimum.

Repeat until convergence {

$$\theta_j \leftarrow \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

}

Update rules:

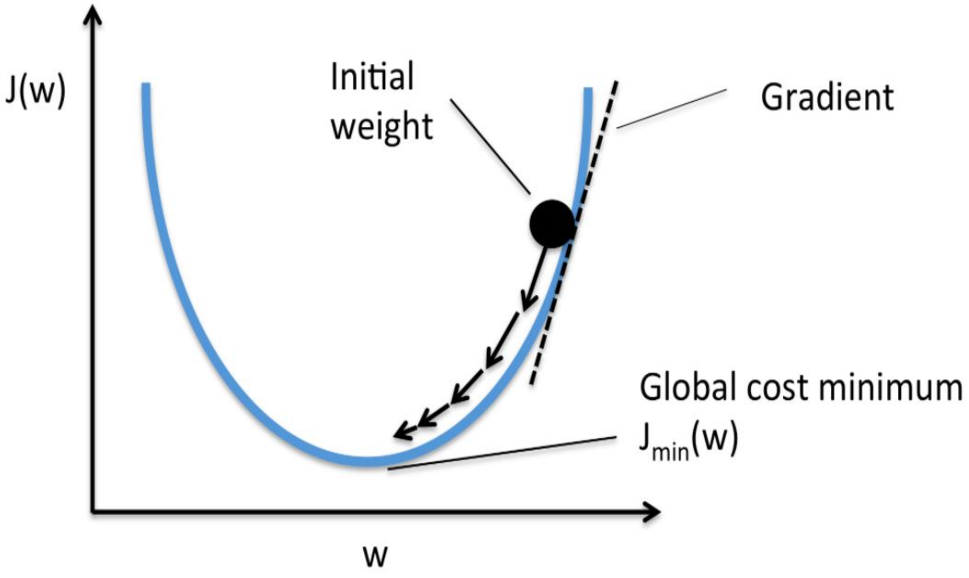
$$\theta_0 := \theta_0 - \alpha \frac{d}{d\theta_0} J(\theta_0, \theta_1)$$

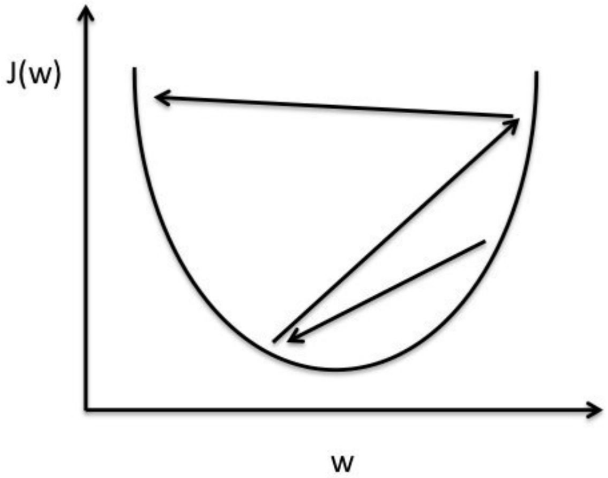
$$\theta_1 := \theta_1 - \alpha \frac{d}{d\theta_1} J(\theta_0, \theta_1)$$

Derivatives:

$$\frac{d}{d\theta_0} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)$$

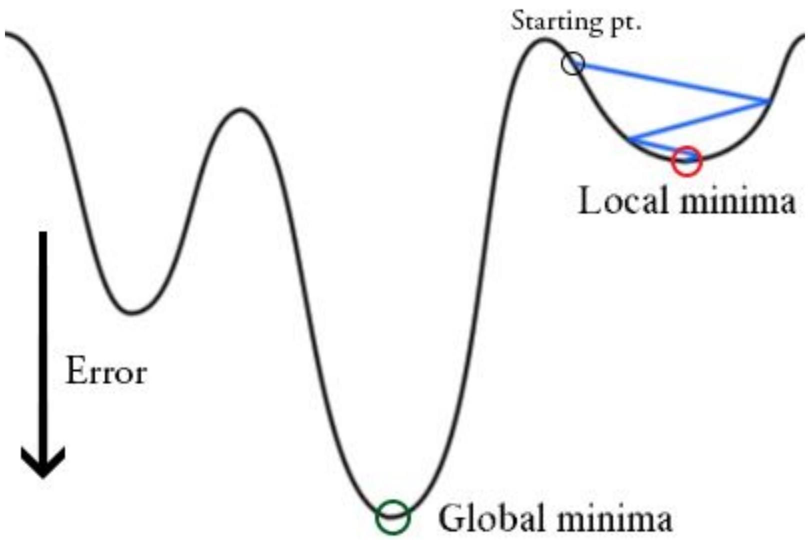
$$\frac{d}{d\theta_1} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right) \cdot x^{(i)}$$





Large learning rate: Overshooting.

Global vs Local Optimum



Linear Regression with Multi-Variate

Multivariate Linear Regression

Hypothesis: $h_\theta(x) = \theta^T x = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$

Parameters: $\theta_0, \theta_1, \ldots, \theta_n$

Cost function:

$$J(\theta_0, \theta_1, \ldots, \theta_n) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

Gradient descent:

Repeat {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \ldots, \theta_n)$$

} (simultaneously update for every $j = 0, \ldots, n$)

This is very difficult as the theta updates are tedious to calculate

Multi-variate Linear Regression using Normal Equation

Normal equation

$$\Theta = (X^T X)^{-1} X^T y$$

Examples: $m = 4.$

	Size (feet²)	Number of bedrooms	Number of floors	Age of home (years)	Price ($1000)
x_0	x_1	x_2	x_3	x_4	y
1	2104	5	1	45	460
1	1416	3	2	40	232
1	1534	3	2	30	315
1	852	2	1	36	178

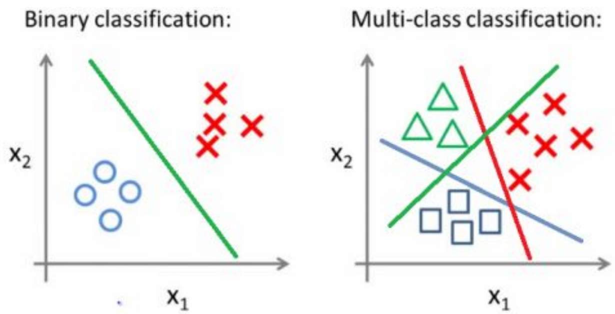
$$X = \begin{bmatrix} 1 & 2104 & 5 & 1 & 45 \\ 1 & 1416 & 3 & 2 & 40 \\ 1 & 1534 & 3 & 2 & 30 \\ 1 & 852 & 2 & 1 & 36 \end{bmatrix} \qquad y = \begin{bmatrix} 460 \\ 232 \\ 315 \\ 178 \end{bmatrix}$$

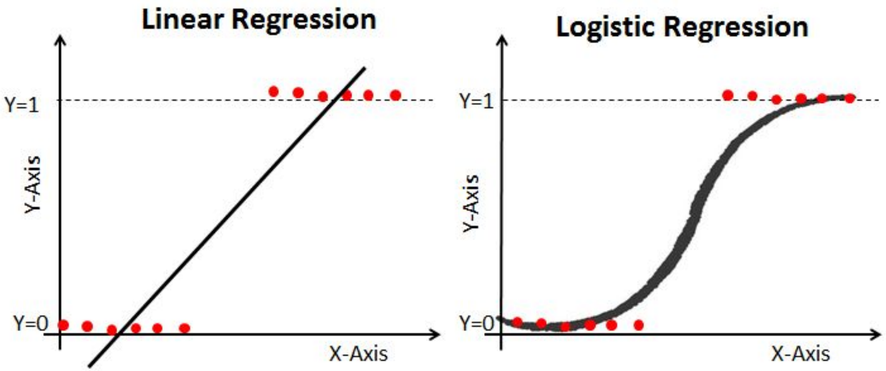
m × (n+1)

m-dimensional vector

$$\boxed{\theta = (X^T X)^{-1} X^T y}$$

Logistic Regression

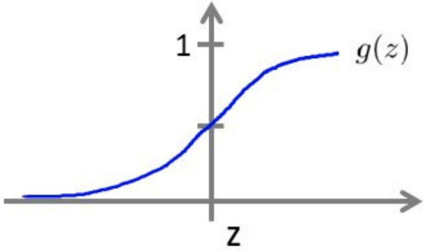




Logistic regression

$$h_\theta(x) = g(\theta^T x)$$

$$g(z) = \frac{1}{1+e^{-z}}$$



Suppose predict "$y = 1$" if $h_\theta(x) \geq 0.5$

predict "$y = 0$" if $h_\theta(x) < 0.5$

Training set: $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \cdots, (x^{(m)}, y^{(m)})\}$

m examples $\quad x \in \begin{bmatrix} x_0 \\ x_1 \\ \cdots \\ x_n \end{bmatrix} \quad x_0 = 1, y \in \{0, 1\}$

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$

How to choose parameters θ ?

Cost Function for Logistic Regression

$$\mathrm{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \mathrm{Cost}(h_\theta(x^{(i)}), y^{(i)})$$

Gradient Descent

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log\left(1 - h_\theta(x^{(i)})\right)\right]$$

Want $\min_\theta J(\theta)$:

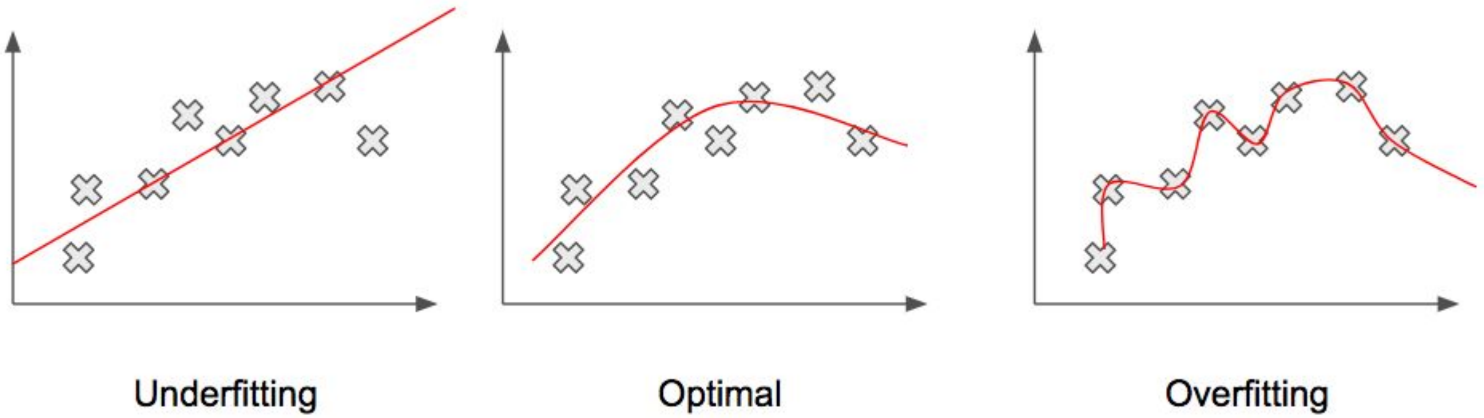
Repeat $\{$

$$\theta_j := \theta_j - \alpha \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

(simultaneously update all θ_j)

$\}$

Overfitting and underfitting



Naive Bayes

Bayes Theorem:

- Let P(c) is probability of a team winning.
- Let P(d) is probability of rain in weather forecast.
- P(c/d) is probability that team will win, given that it will rain.

$$P(c|d) = \frac{P(d|c)P(c)}{P(d)}$$

$$c_{MAP} = \underset{c \in C}{\operatorname{argmax}} P(c \mid d)$$

MAP is "maximum a posteriori" = most likely class

$$= \underset{c \in C}{\operatorname{argmax}} \frac{P(d \mid c)P(c)}{P(d)}$$

Bayes Rule

$$= \underset{c \in C}{\operatorname{argmax}} P(d \mid c)P(c)$$

Dropping the denominator as it is equal to 1, total probability.

$$c_{MAP} = \underset{c \in C}{\operatorname{argmax}} P(d \mid c) P(c)$$

$$= \underset{c \in C}{\operatorname{argmax}} P(x_1, x_2, \ldots, x_n \mid c) P(c)$$

Conditional Independence - Assumption in NB

- Assume the feature probabilities $P(x_i|c_j)$ are independent given the class *c.*

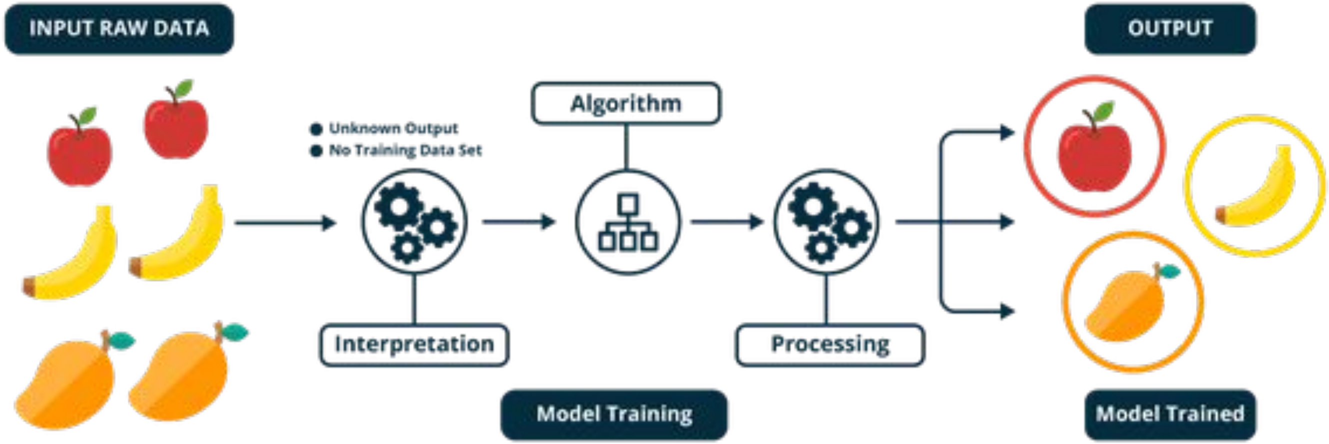
$$P(x_1, x_2, \ldots, x_n \mid c)$$

$$P(x_1, \ldots, x_n \mid c) = P(x_1 \mid c) \bullet P(x_2 \mid c) \bullet P(x_3 \mid c) \bullet \ldots \bullet P(x_n \mid c)$$

$$c_{MAP} = \underset{c \in C}{\operatorname{argmax}}\, P(x_1, x_2, \ldots, x_n \mid c) P(c)$$

$$c_{NB} = \underset{c \in C}{\operatorname{argmax}}\, P(c_j) \prod_{x \in X} P(x \mid c)$$

K-Means Clustering



K-Means Algorithm

1. Initialization
   a. Randomly choose K data points as initial centroids
   b. Each centroid defines one cluster
2. Cluster Assignment
   a. All the data points are assigned to one of the clusters based on euclidean distance
3. Move the centroid
   a. Calculate the new cluster center by taking the average of the data points under the cluster
   b. Repeat step 2 and 3 until convergence

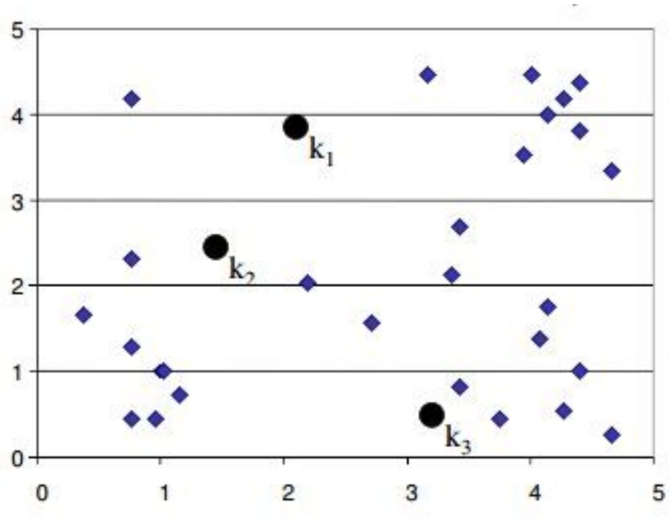
Objective function

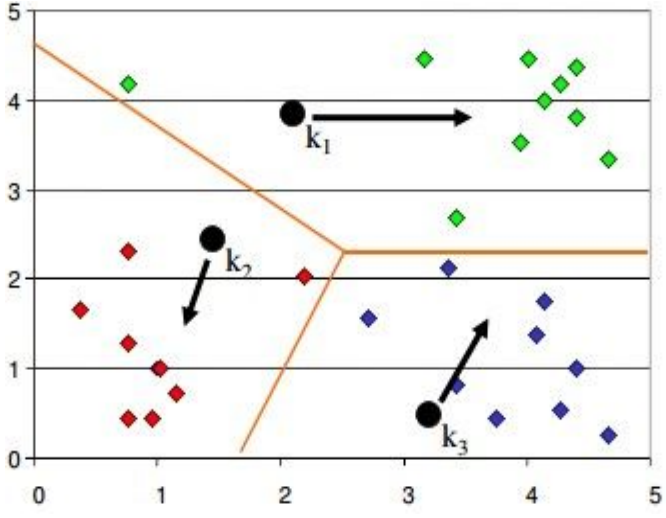
$$\underset{c_i \in C}{\arg\min}\; dist(c_i, x)^2$$

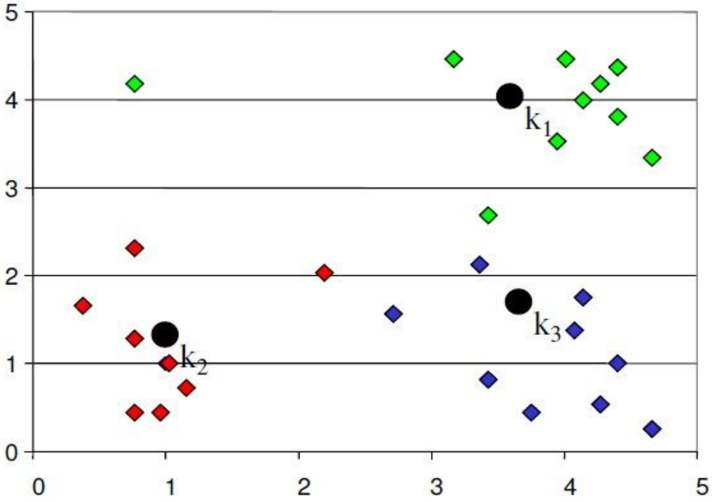
$$c_i = \frac{1}{|S_i|}\sum_{x_i \in S_i} x_i$$

$$d(\mathbf{p}, \mathbf{q}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2}.$$

Iterating cluster centroids



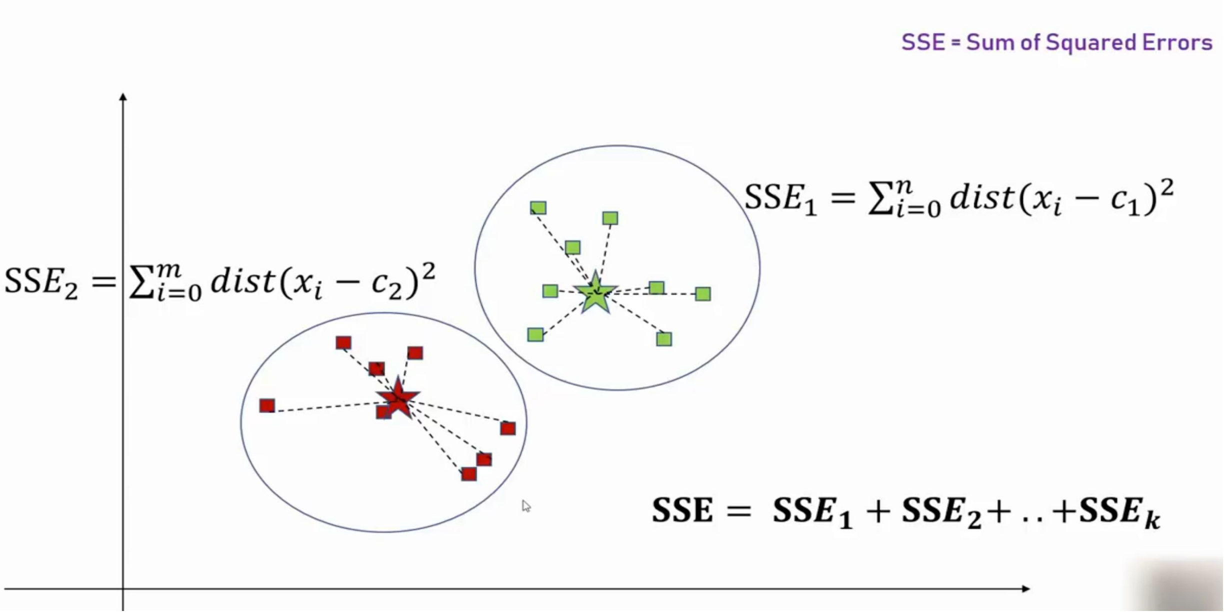


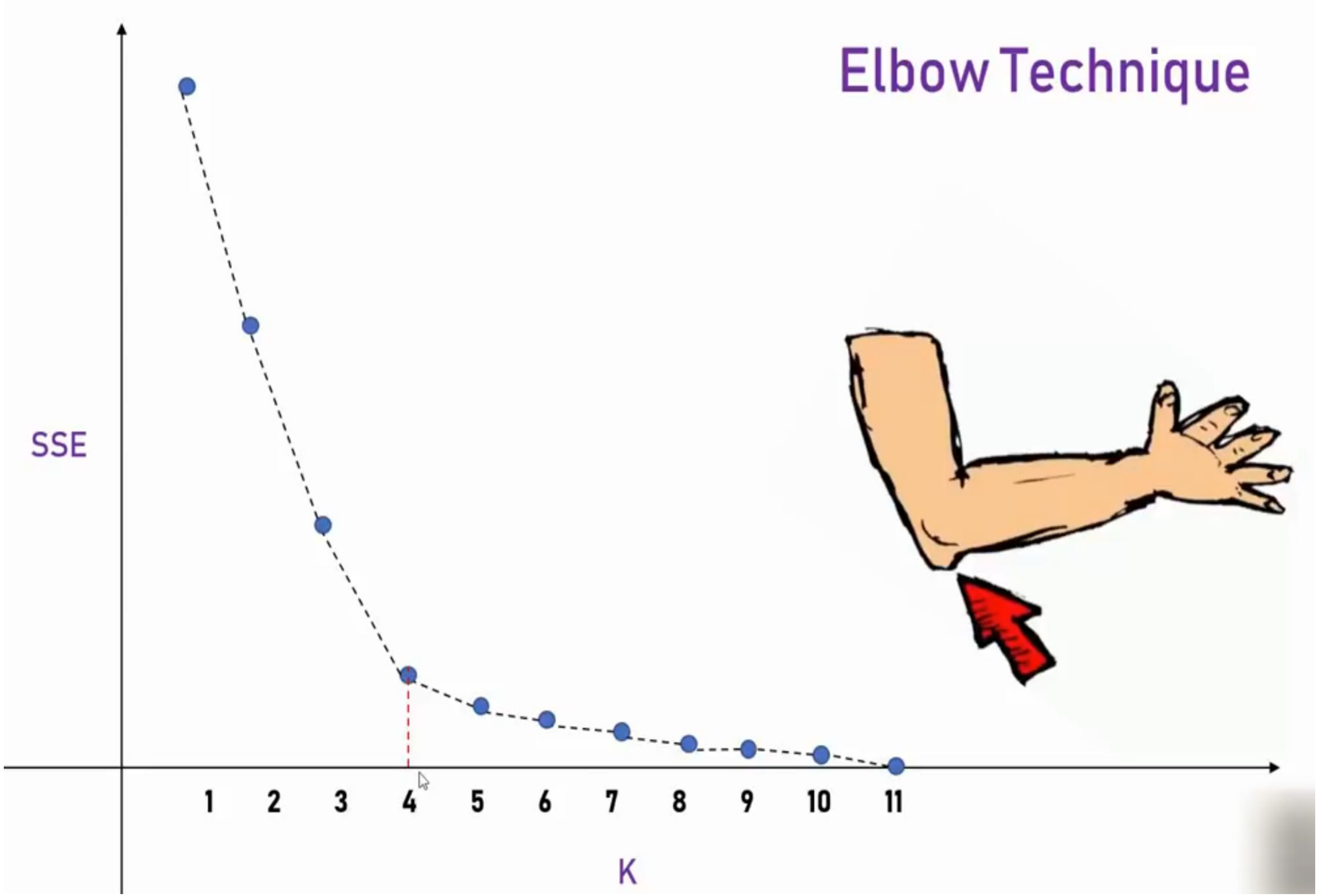


5
4
3
2
1
0
0
1
2
3
4
5
k1
k2
k3

How do you decide the value of K?

Generally we follow ELBOW method to determine the best K value.




Elbow Technique
SSE
1
2
3
4
5
6
7
8
9
10
11
K

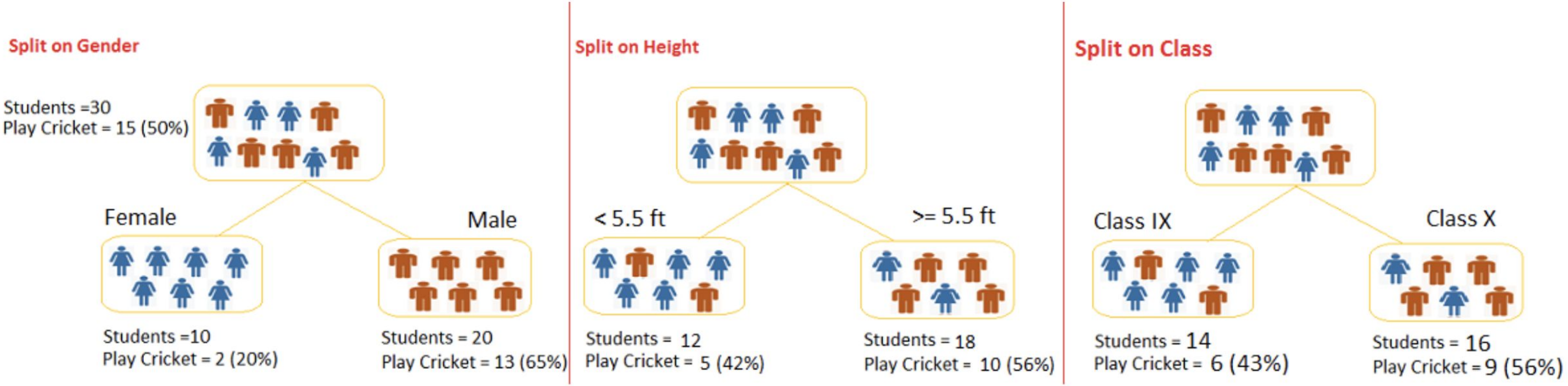
Decision trees

- Best suitable for multi class classification.
- Suitable for both categorical data and continuous data.
- Splits the population into homogeneous sets (pure sets). To do that, it uses parameters like gini index, entropy etc.
- **Types:** Classification trees (works well with categorical data), Regression trees (works with continuous data).

Example:

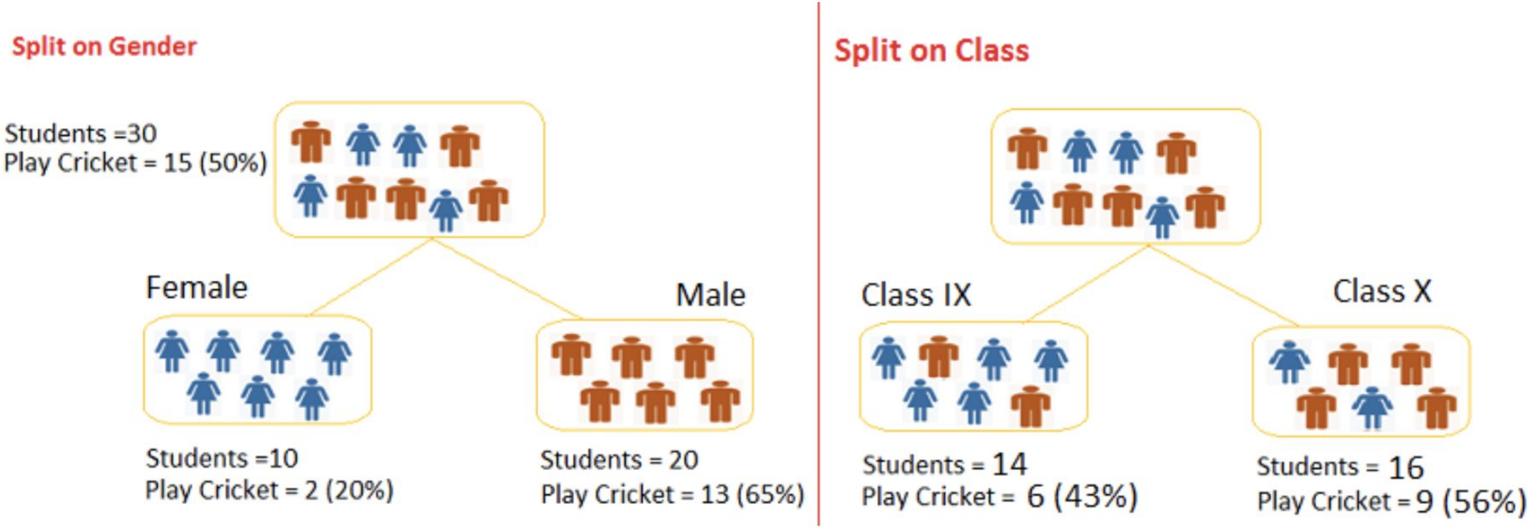
A class of 30 students including boys and girls, create a model to predict who will play cricket during leisure period?

Where to split? Split on gender, height or class?



Gini Index

- Gini says, if we select two items from a population at random then they must be of same class and probability for this is 1 if population is pure.
- Performs only binary splits
- Works with both classification and regression trees.
- Higher the value of Gini, higher the homogeneity in data.



Split on Gender:

1. Calculate, Gini for sub-node Female = (0.2)*(0.2)+(0.8)*(0.8)=0.68
2. Gini for sub-node Male = (0.65)*(0.65)+(0.35)*(0.35)=0.55
3. Calculate weighted Gini for Split Gender = (10/30)*0.68+(20/30)*0.55 = **0.59**

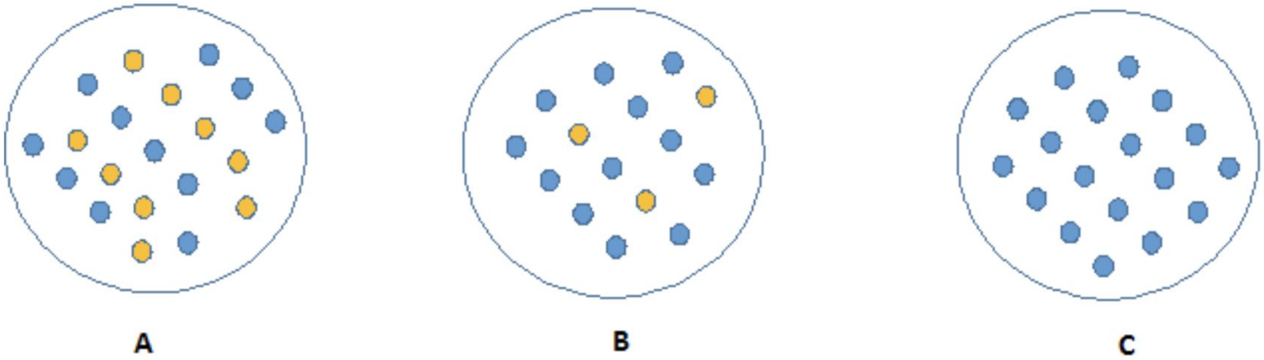
Similar for Split on Class:

1. Gini for sub-node Class IX = (0.43)*(0.43)+(0.57)*(0.57)=0.51
2. Gini for sub-node Class X = (0.56)*(0.56)+(0.44)*(0.44)=0.51
3. Calculate weighted Gini for Split Class = (14/30)*0.51+(16/30)*0.51 = **0.51**

Entropy and Information Gain

Information gain = 1 - entropy.

A = more impure; B = impure; C= pure



$$Entropy = \sum_{i=1}^{n} -p(c_i)log_2(p(c_i))$$

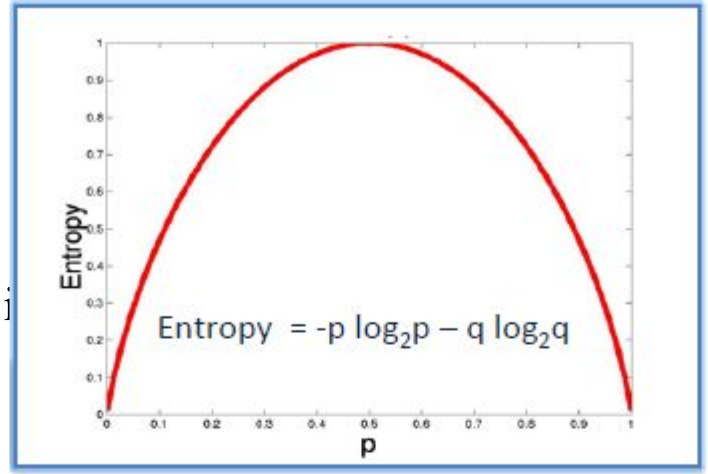
where $p(c_i)$ is the probability/percentage of class c_i in a node.

Here p and q are the probability of success and failure.

Steps:

1. Calculate entropy of parent node
2. Calculate entropy of each individual node of split and

   calculate weighted average of all sub-nodes available i



Entropy = $-0.5 \log_2 0.5 - 0.5 \log_2 0.5 = 1$

1. Entropy for parent node = -(15/30) log2 (15/30) – (15/30) log2 (15/30) = **1**. Here 1 shows that it is a impure node.
2. Entropy for Female node = -(2/10) log2 (2/10) – (8/10) log2 (8/10) = 0.72 and for male node, -(13/20) log2 (13/20) – (7/20) log2 (7/20) = **0.93**
3. Entropy for split Gender = Weighted entropy of sub-nodes = (10/30)*0.72 + (20/30)*0.93 = **0.86**
4. Entropy for Class IX node, -(6/14) log2 (6/14) – (8/14) log2 (8/14) = 0.99 and for Class X node, -(9/16) log2 (9/16) – (7/16) log2 (7/16) = 0.99.
5. Entropy for split Class = (14/30)*0.99 + (16/30)*0.99 = **0.99**

Since entropy is less for the gender, we split on that.

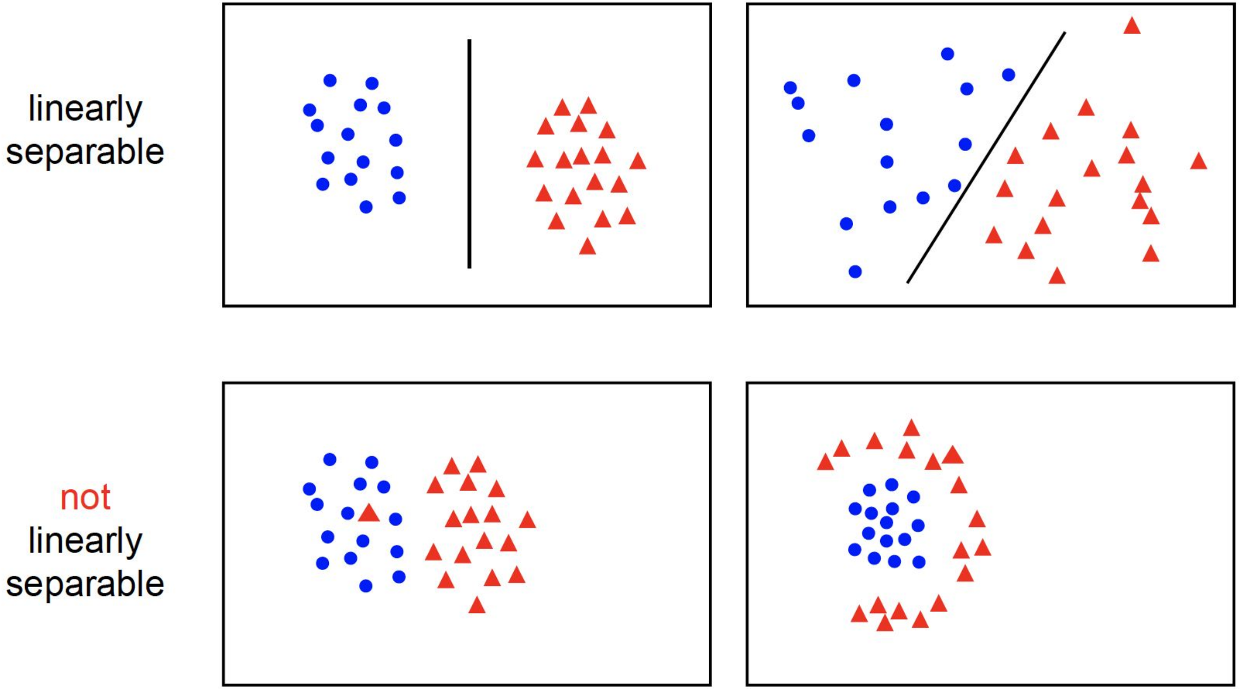
Entropy less for gender indicates that, information gain is more for gender attribute.

Disadvantages

Decision trees are prone to overfitting, specially when the data is highly numerical.

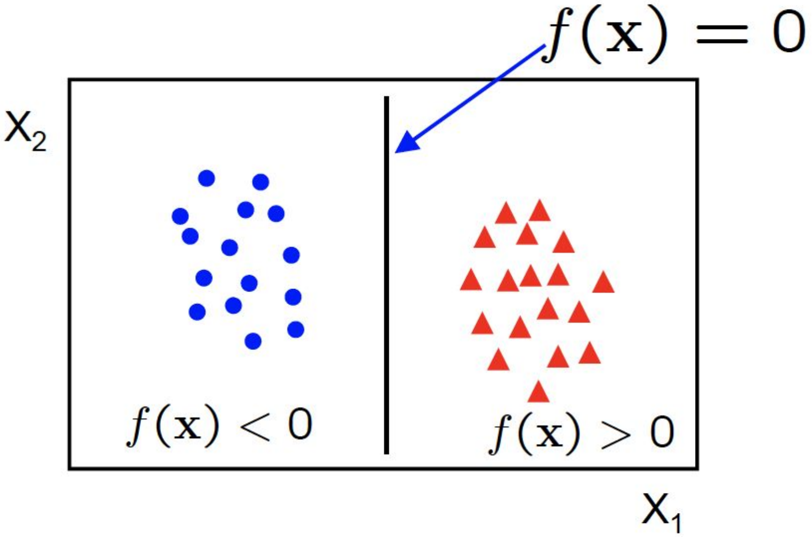
Solution is to prune the trees or go for some ensemble techniques like Random forests.

Support Vector Machines (SVM)



A linear classifier has the form

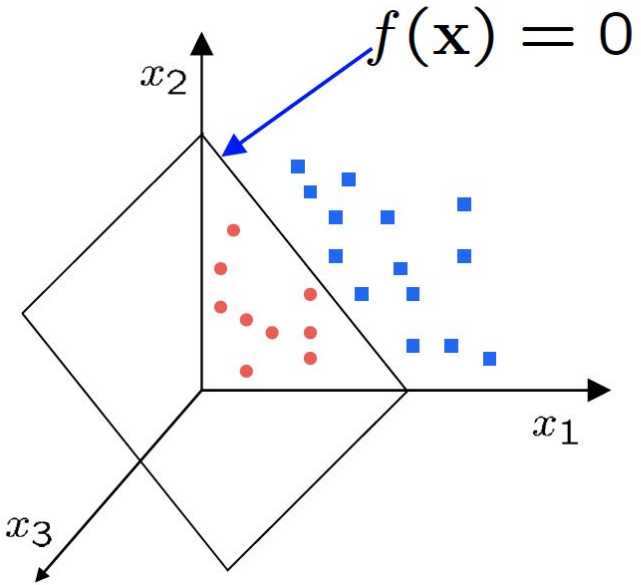
$$f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$$



- in 2D the discriminant is a line
- $\mathbf{w}$ is the normal to the line, and b the bias
- $\mathbf{w}$ is known as the weight vector

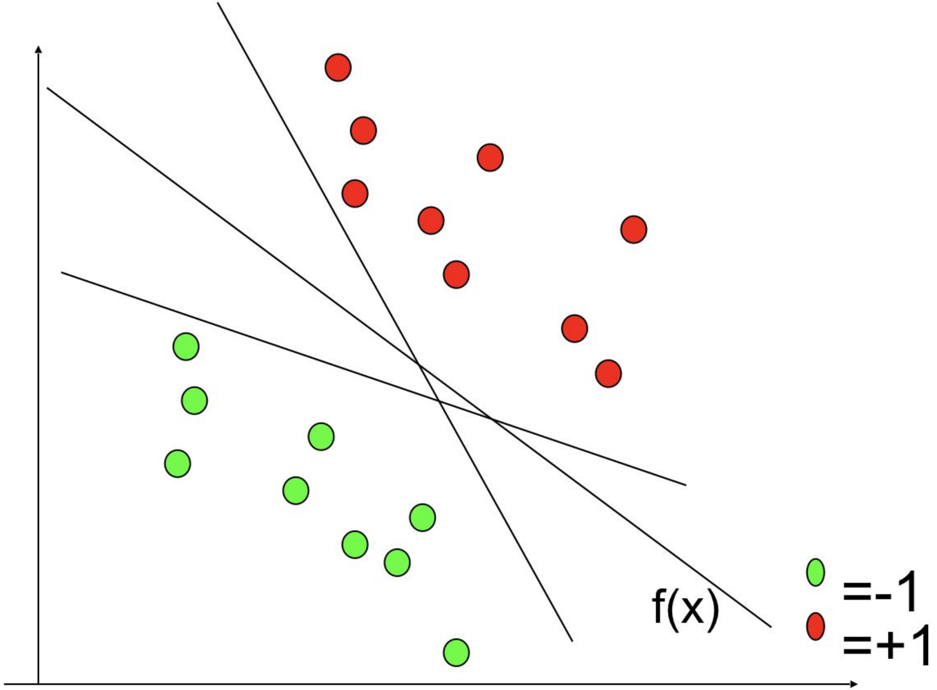
A linear classifier has the form

$$f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$$

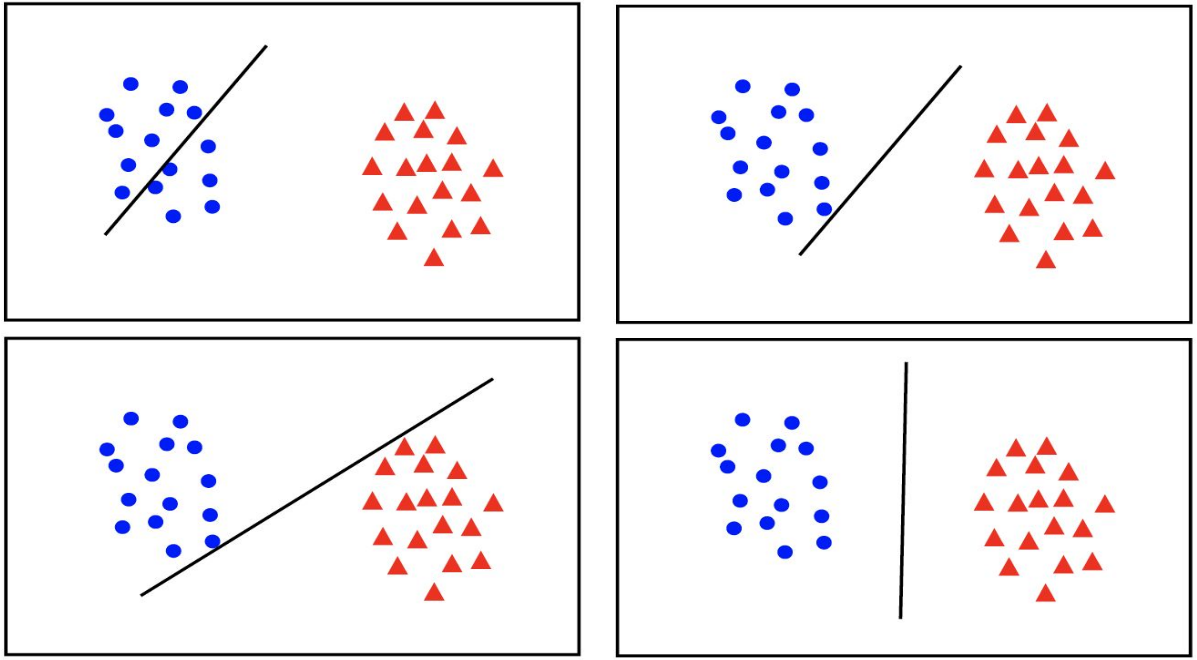


- in 3D the discriminant is a plane, and in nD it is a hyperplane

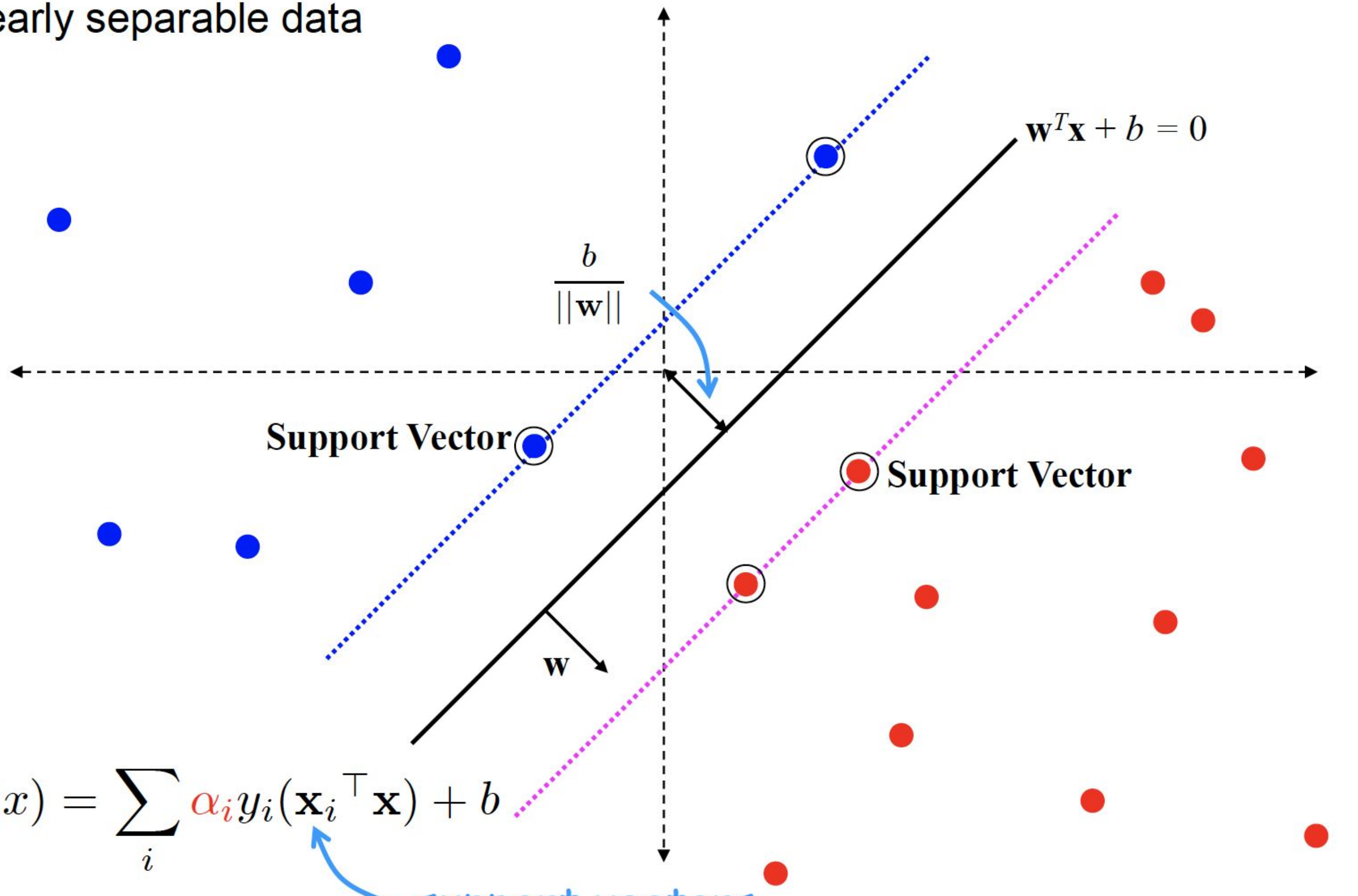
Fitting the correct hypothesis



What is the best w?

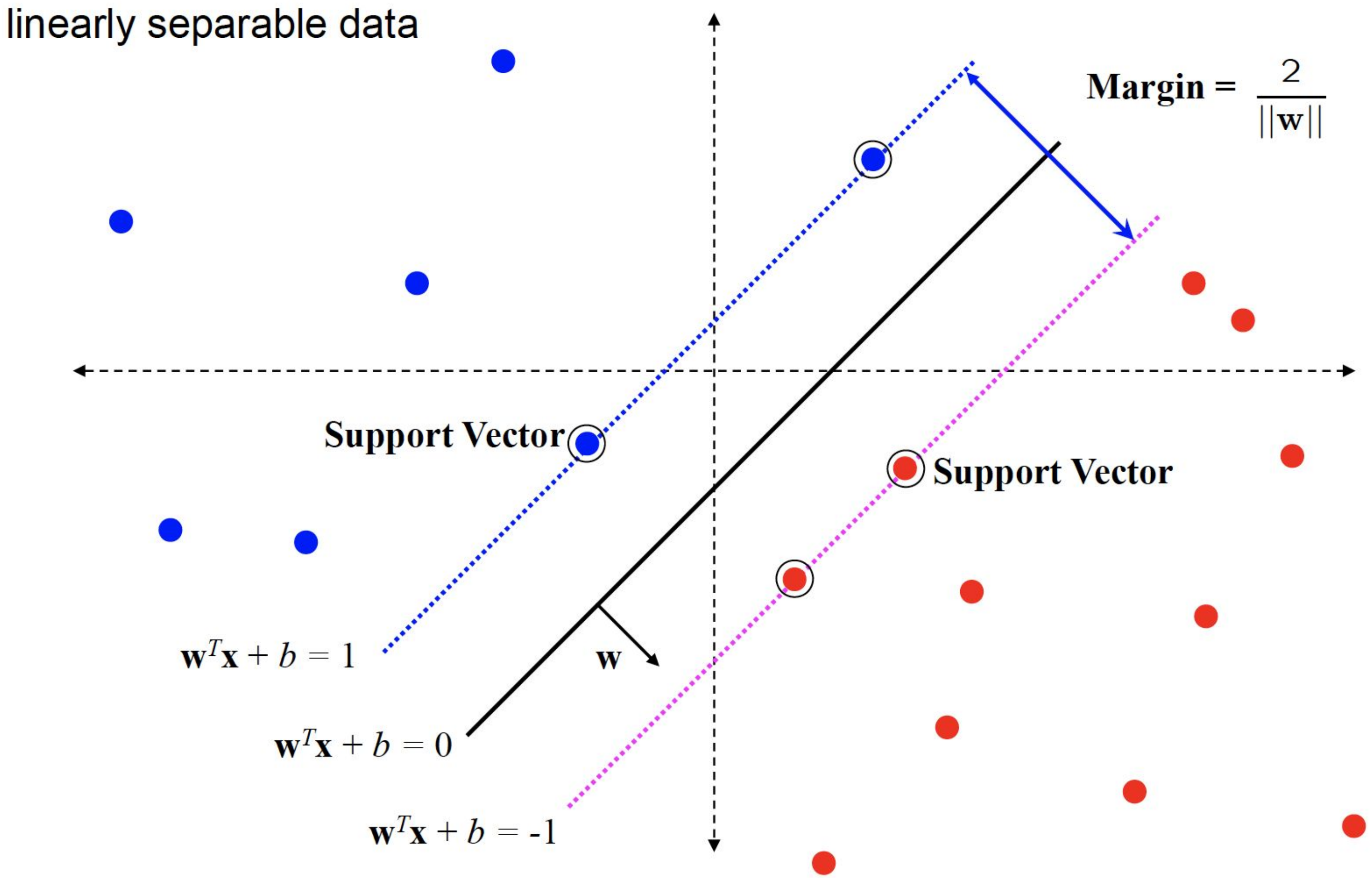


• maximum margin solution: most stable under perturbations of the inputs linearly separable data

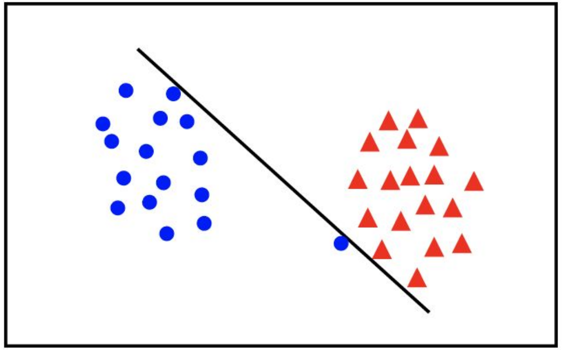


$$f(x) = \sum_i \alpha_i y_i (\mathbf{x}_i{}^\top \mathbf{x}) + b$$

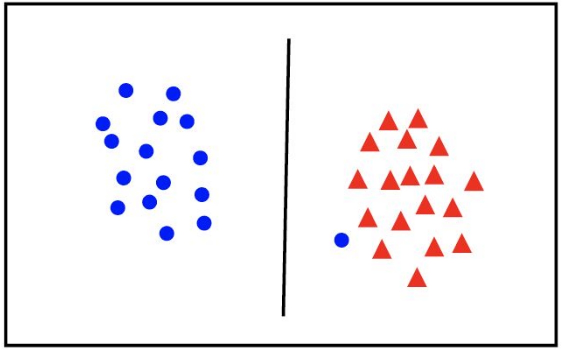
support vectors



linearly separable data
Margin = $\frac{2}{||\mathbf{w}||}$
Support Vector
Support Vector
$\mathbf{w}^T\mathbf{x} + b = 1$
$\mathbf{w}$
$\mathbf{w}^T\mathbf{x} + b = 0$
$\mathbf{w}^T\mathbf{x} + b = -1$



• the points can be linearly separated but there is a very narrow margin



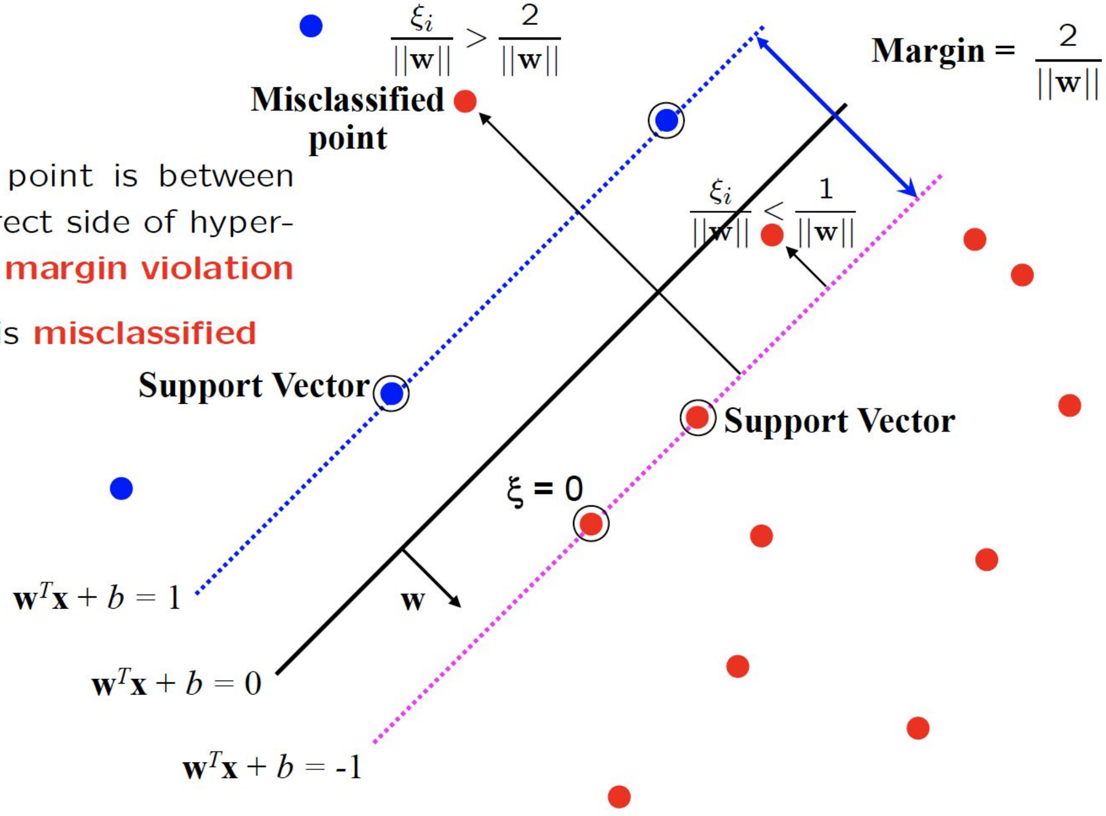
• but possibly the large margin solution is better, even though one constraint is violated

In general there is a trade off between the margin and the number of mistakes on the training data

Slack Variables

$$\xi_i \geq 0$$

- for $0 < \xi \leq 1$ point is between margin and correct side of hyper-plane. This is a **margin violation**
- for $\xi > 1$ point is **misclassified**



Soft Margin and Hard Margin

The optimization problem becomes

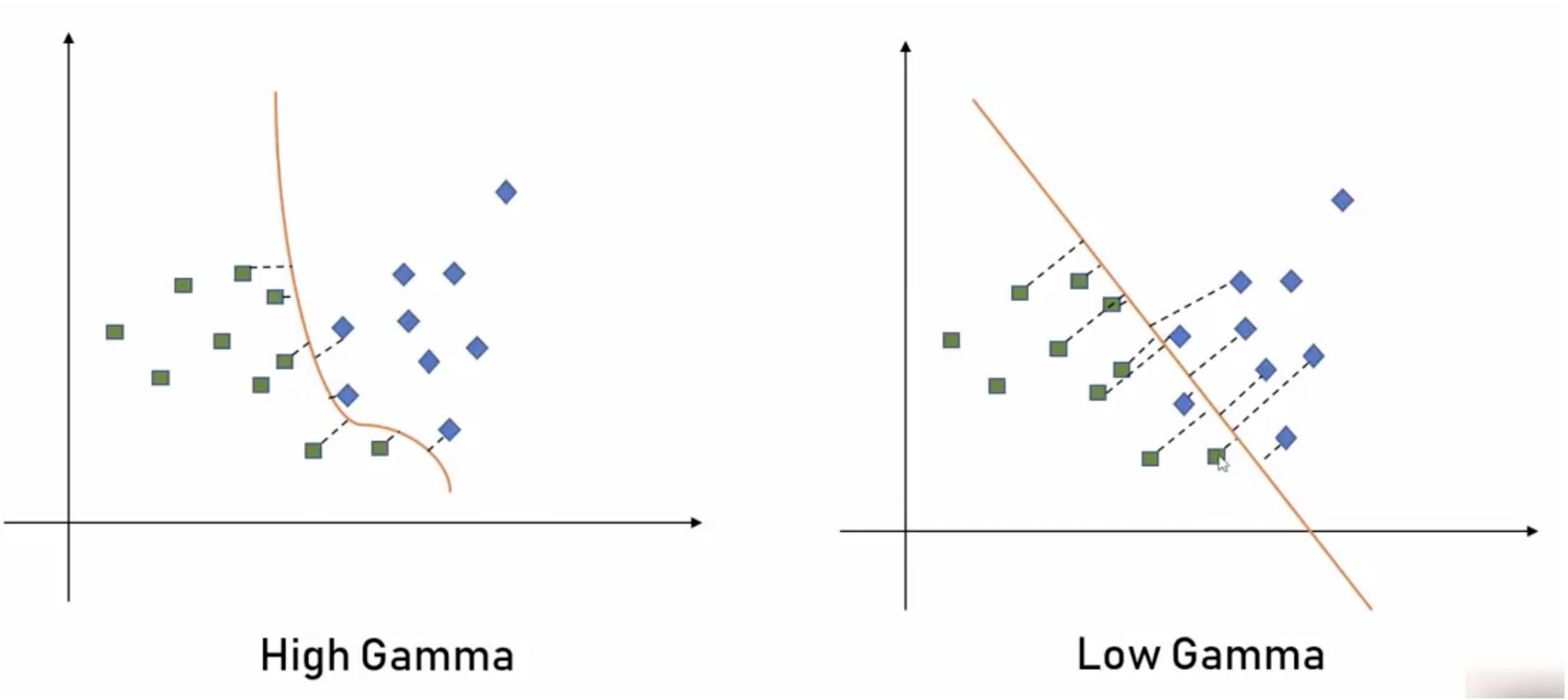
$$\min_{\mathbf{w}\in\mathbb{R}^d, \xi_i \in \mathbb{R}^+} ||\mathbf{w}||^2 + C \sum_i^N \xi_i$$

subject to

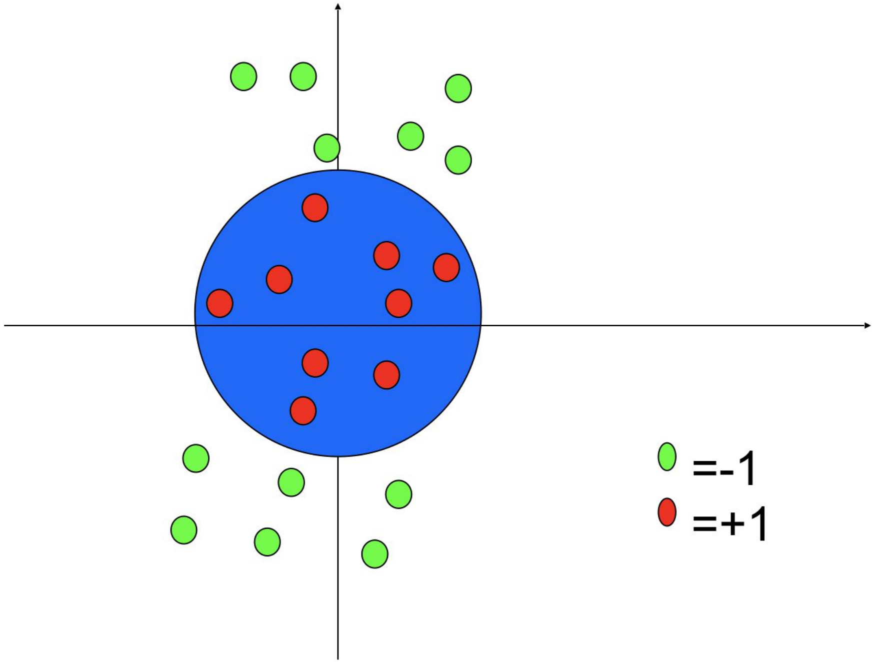
$$y_i \left(\mathbf{w}^\top \mathbf{x}_i + b \right) \geq 1 - \xi_i \text{ for } i = 1 \ldots N$$

- Every constraint can be satisfied if ξ_i is sufficiently large
- C is a regularization parameter:
  - small C allows constraints to be easily ignored $\rightarrow$ large margin
  - large C makes constraints hard to ignore $\rightarrow$ narrow margin
  - $C = \infty$ enforces all constraints: hard margin

Other parameter is Gamma



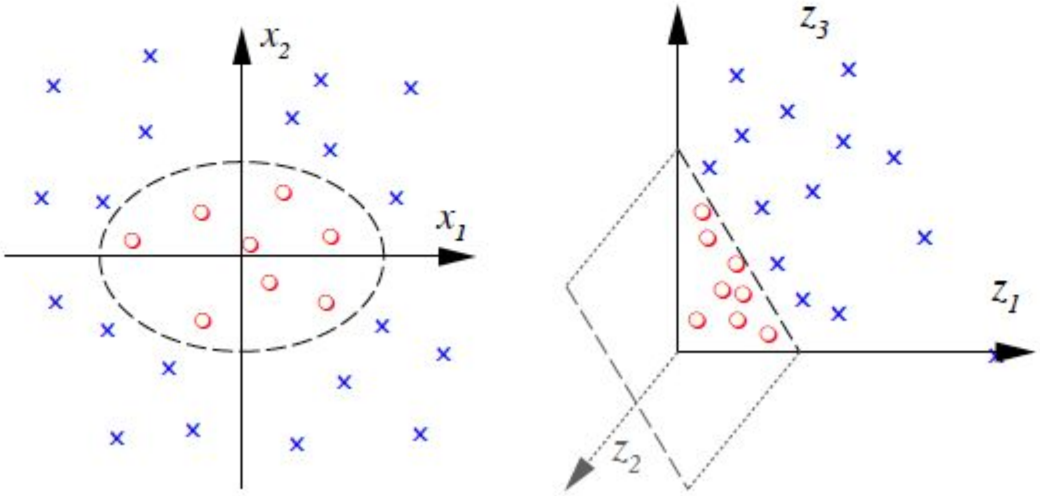
Kernel trick



What if the decision function is not a linear?

Kernel Trick

$$\Phi : R^2 \rightarrow R^3$$
$$(x_1, x_2) \mapsto (z_1, z_2, z_3) := (x_1^2, \sqrt(2)x_1x_2, x_2^2)$$



Kernels

1. Linear
2. Polynomial
3. Gaussian
4. Sigmoid

Advantages

- Can be used for both classification and regression
- Performs better on data with noise and outliers.
- Best for most of the datasets, with medium to large number of samples.

Drawbacks:

- Its’ almost a blackbox when it deals with high dimensional data.
- Its slow for large datasets (large number of features/ instances)
- Memory intensive

References

1. Machine Learning course by Dr. Andrew NG, Coursera.
2. Machine learning topics from Towards Data Science, available at http://towardsdatascience.com/
3. Machine learning topics from Analytics Vidhya, available at https://www.analyticsvidhya.com/
4. Saikat Dutt et. al., Machine learning book by Pearson Publications.

Thank You :)

Feel free to contact:

Vamsee Krishna Kiran M

mk_vamseekrishna@cb.amrita.edu

mvk.kiran@gmail.com