



BITS Pilani
Pilani | Dubai | Goa | Hyderabad

Introduction to Classification

Srikanth Gunturu

In this segment

Introduction to Classification

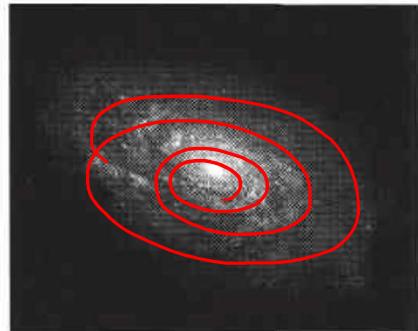
- What is a Classification ?
- Classification – Process and a few terms
- Classification Model – Uses & General Approach



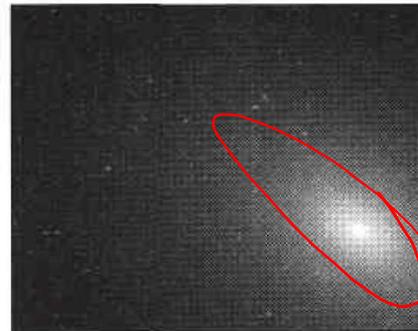
Introduction to Classification

What is Classification

- The task of assigning objects to one of several predefined categories (classes)
- Real world applications of classification
 - Detecting spam email messages based upon the message header and content
 - Categorizing tumours as malignant or benign based upon the results of MRI scan
 - Classifying galaxies based upon their shapes



(a) A spiral galaxy.

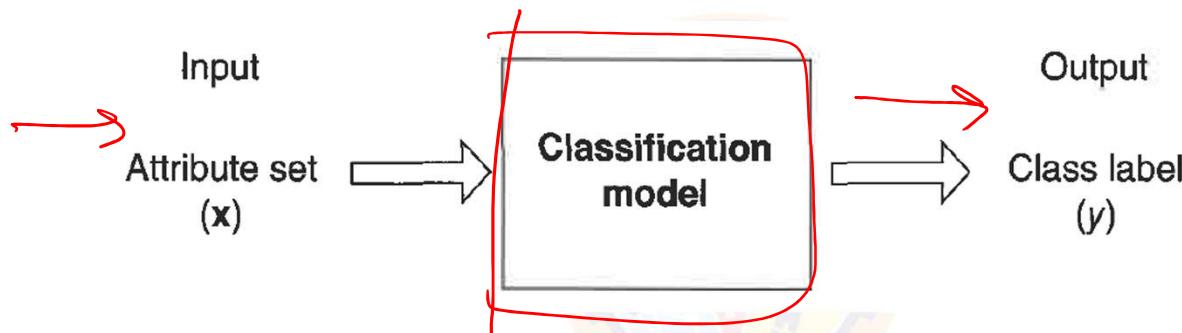


(b) An elliptical galaxy.

Introduction to Classification

Classification – Process and a few terms

- Classification is the task of learning a target function f that maps each attribute set x to one of the predefined class labels y



Two hand-drawn sketches are shown. The top sketch shows a vertical line labeled $y = f(x)$ with a point x marked above it. The bottom sketch shows a more complex, piecewise linear function labeled $y = f(x)$ with a point x marked above it.

- Feature / Attribute – Input attribute that's used in classifying the output
- Feature set – Set of attributes that govern the output class
- Label – Output class that the object is assigned to

Introduction to Classification

Classification Model – Uses

- Descriptive - Serves as an explanatory tool to distinguish between objects of different classes

To - 30

Name	Body Temperature	Skin Cover	Gives Birth	Aquatic Creature	Aerial Creature	Has Legs	Hibernates	Class Label
human	warm-blooded	hair	yes	no	no	yes	no	mammal
python	cold-blooded	scales	no	no	no	no	yes	reptile
salmon	cold-blooded	scales	no	yes	no	no	no	fish
whale	warm-blooded	hair	yes	yes	no	no	no	mammal
frog	cold-blooded	none	no	semi	no	yes	yes	amphibian
komodo dragon	cold-blooded	scales	no	no	no	yes	no	reptile
bat	warm-blooded	hair	yes	no	yes	yes	yes	mammal
pigeon	warm-blooded	feathers	no	no	yes	yes	no	bird
cat	warm-blooded	fur	yes	no	no	yes	no	mammal
leopard	cold-blooded	scales	yes	yes	no	no	no	fish
shark								
turtle	cold-blooded	scales	no	semi	no	yes	no	reptile
penguin	warm-blooded	feathers	no	semi	no	yes	no	bird
porcupine	warm-blooded	quills	yes	no	no	yes	yes	mammal
eel	cold-blooded	scales	no	yes	no	no	no	fish
salamander	cold-blooded	none	no	semi	no	yes	yes	amphibian

36

To

- Predictive – Used to predict the class label of unknown records

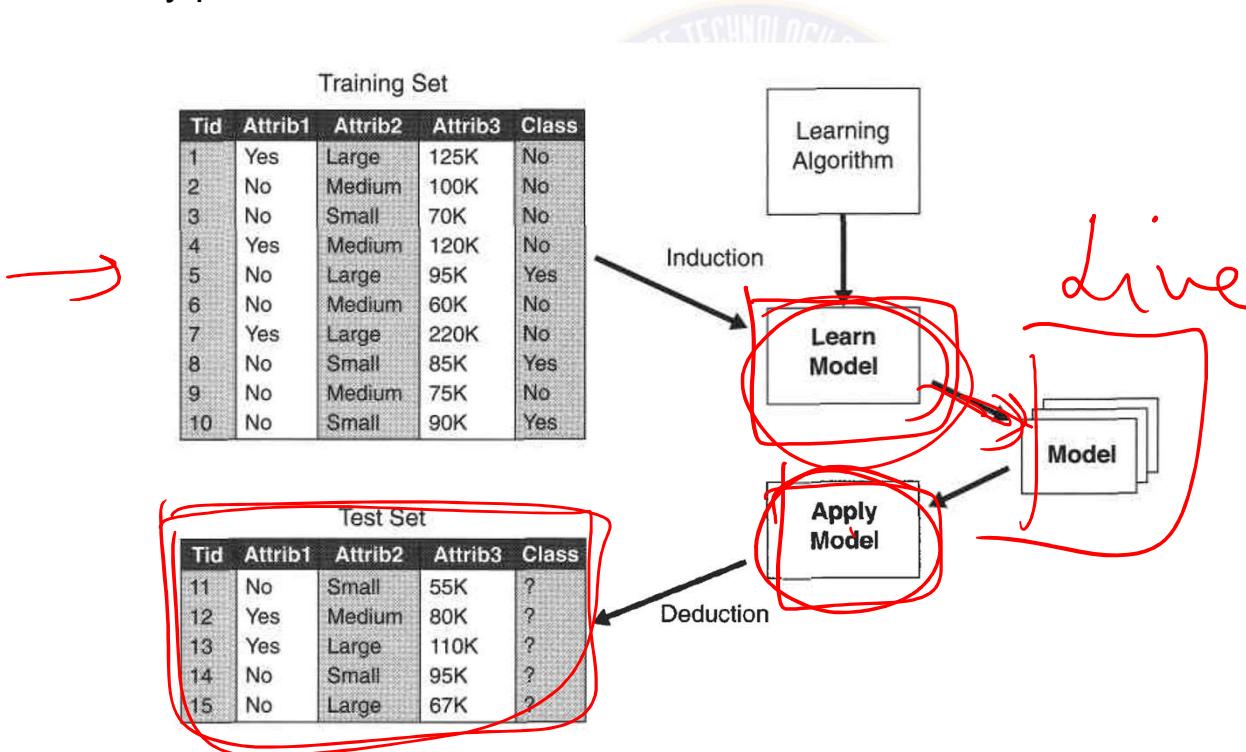
→

Name	Body Temperature	Skin Cover	Gives Birth	Aquatic Creature	Aerial Creature	Has Legs	Hibernates	Class Label
gila monster	cold-blooded	scales	no	no	no	yes	yes	?

Introduction to Classification

Classification Model – General Approach

- Should fit the input data well
- Should correctly predict the class labels of records it has never seen before



In this segment

Types of classification algorithms

- Discriminant functions
- Probabilistic Generative models
- Probabilistic Discriminative models
- Tree based models



Types of classification algorithms

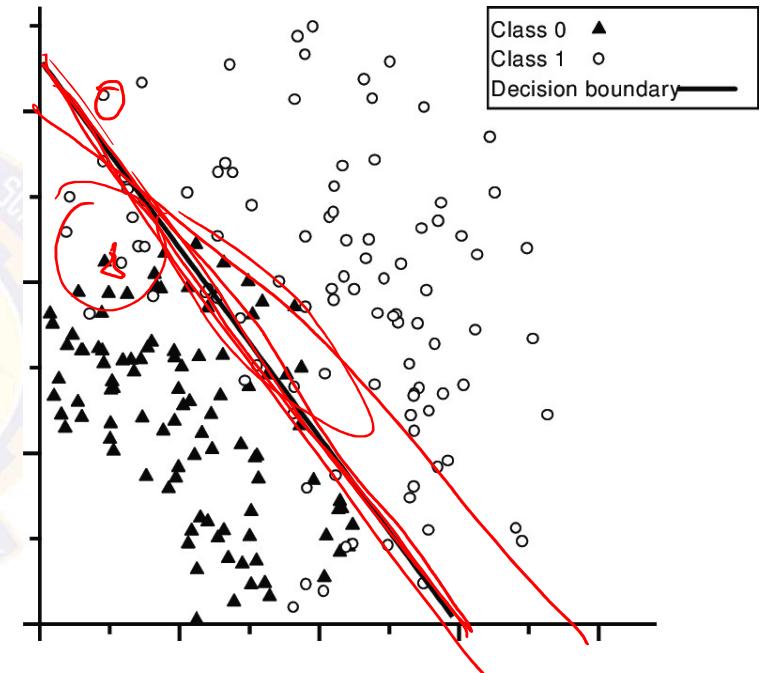
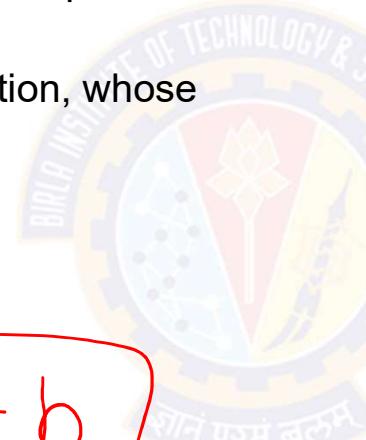
Discriminant functions - Overview

- Both inference and decision are performed together
- Discriminant is a function that maps input vector directly to a class label
- Linear discriminant is a linear function, whose decision surface is a hyperplane
- Examples:
 - Linear Discriminant
 - The Perceptron

$$y = \alpha x_1 + b$$

$$y = \alpha x_1 + b x_2 + c$$

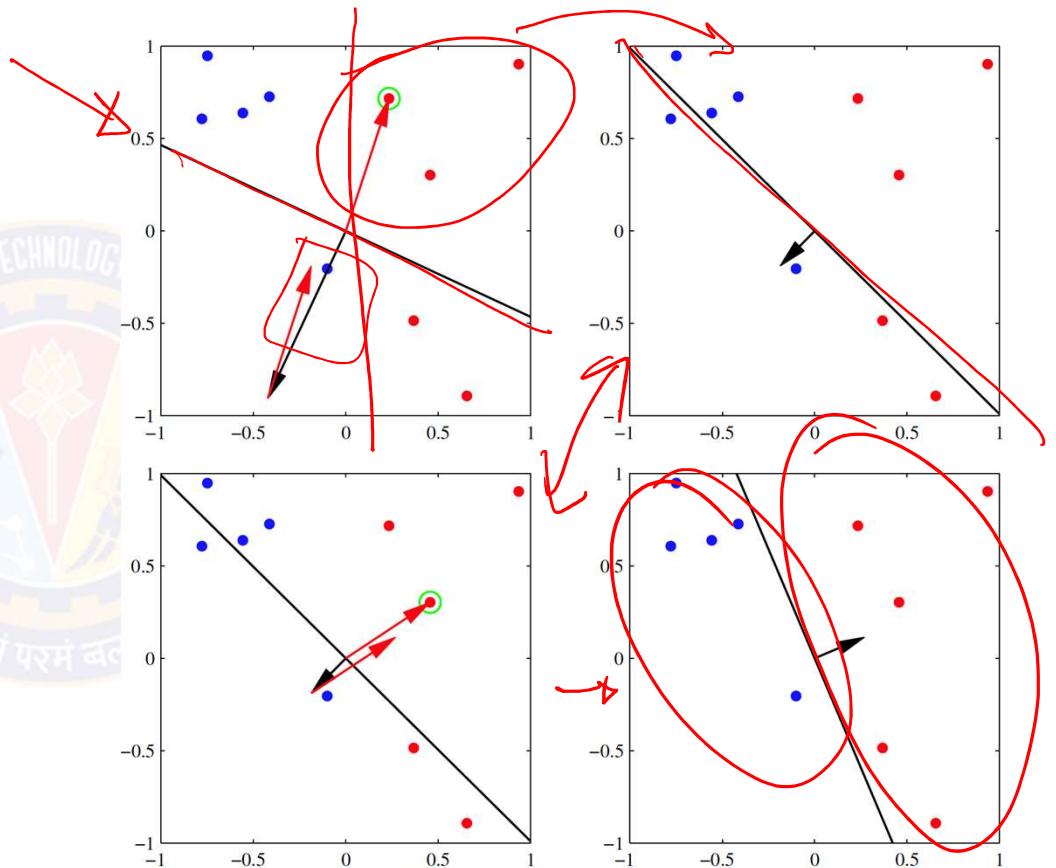
$$y = w_1 x_1 + w_2 x_2 + \dots + w_n x_n + w_0$$



Types of classification algorithms

Discriminant functions – The Perceptron

- Proposed by Frank Rosenblatt in 1962
- Corresponds to a two class model
- Associates zero error with correctly classified patterns
- Tries to minimise the criterion quantity for misclassified pattern
- Perceptron convergence (fig.)

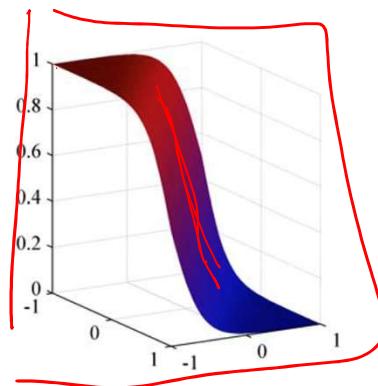
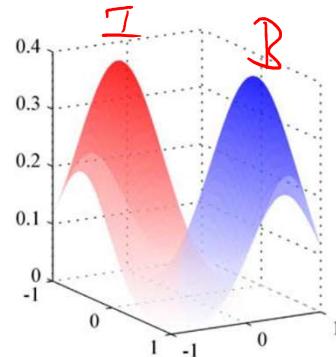
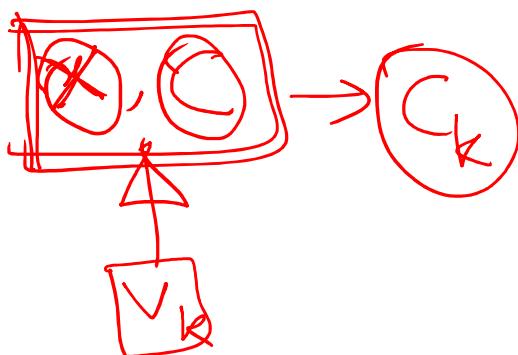


Types of classification algorithms

Probabilistic Generative Models - Overview

- Uses probability theory to generate joint distribution of input and output variables, $P(X,C)$ and then use that distribution to predict the class for a given new input set
- Approach
 - Inference - Model joint distribution of input vector X and class C_k , using probability
 - Decision – Apply the model to given new set of input to determine probability of class C_k
- Two-class – probability
- Examples: Naïve Bayes, Gaussian Mixture Model etc.

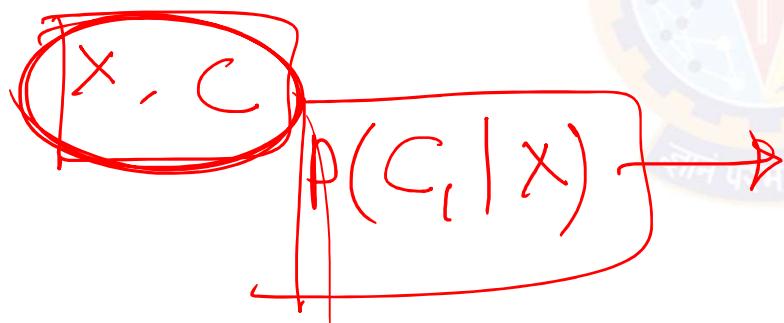
$$p(C_1|x) = \frac{p(x|C_1)p(C_1)}{p(x|C_1)p(C_1) + p(x|C_2)p(C_2)}$$



Types of classification algorithms

Probabilistic Discriminative Models – Overview

- Uses conditional probability of the class variable C_k given the input vector X , i.e. $P(C_k|X)$
- Approach
 - Inference – Use conditional probability to model the probability of class variable given the input set, using training data
 - Decision – Use decision theory to assign each new input set to one of the classes
- Examples – Logistic Regression



Types of classification algorithms

Probabilistic Generative Vs Probabilistic Discriminative Models

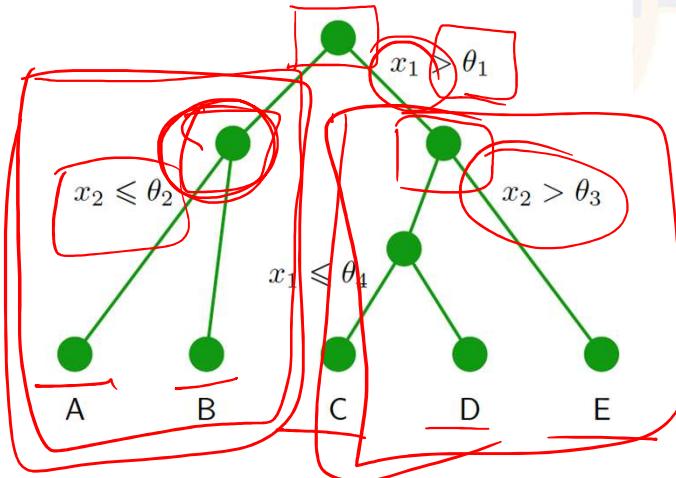
Generative Model	Discriminative Model
Derive joint probability distribution using training data	Derive conditional probability from training data
Requires considerably large training set to generate an decent model	Can work with moderate sized data as well, since the conditional probability is drawn from given input set directly
Models the actual distribution of each class	Models the decision boundary between classes
Use Bayesian theory to assign input to a class	Use maximum likelihood to assign an input to a class

Types of classification algorithms

Tree based Models - Overview

- These models separate input space into multiple partitions based on certain conditions and apply different modelling techniques independently on each sub space (ex: binary tree)
- For a tree with regions R and T number of leaf nodes, optimal prediction is
- Pruning is used for building optimal tree

$$y_{\tau} = \frac{1}{N_{\tau}} \sum_{x_n \in R_{\tau}} t_n$$



In this segment

Classification Algorithms in scope for this course

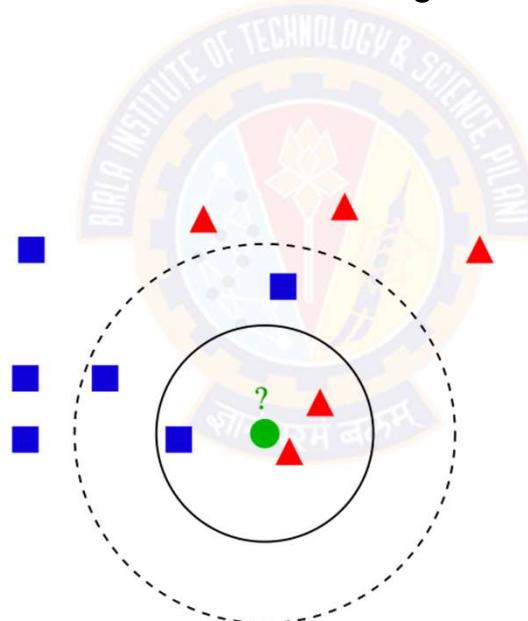
- K-Nearest Neighbor
- Naïve Bayes
- Logistic Regression
- Decision Tree
- Support Vector Machines
- Ensemble Methods



Classification Algorithms in scope for this course

k-Nearest Neighbour Classifier (kNN)

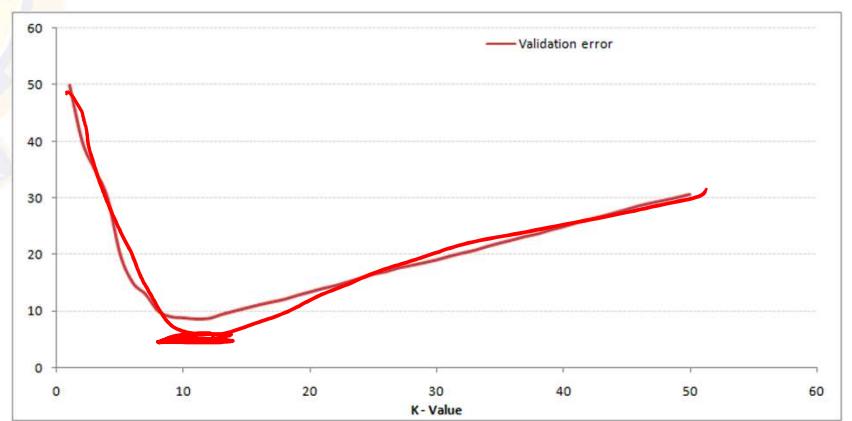
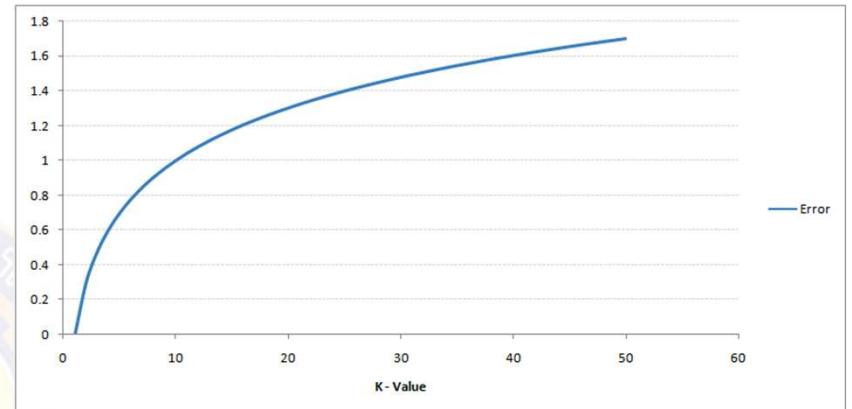
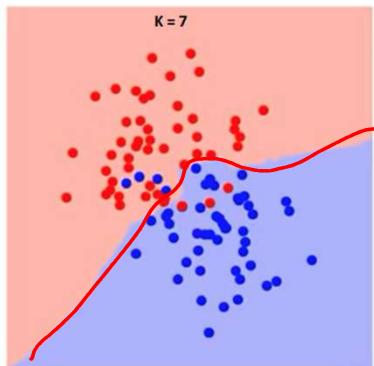
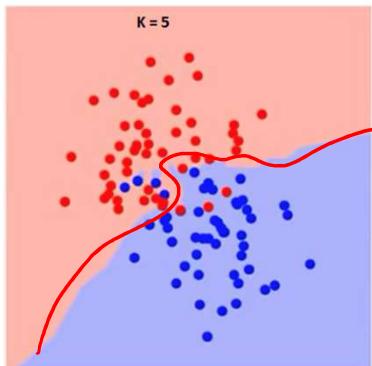
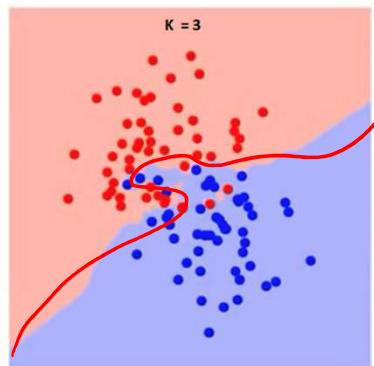
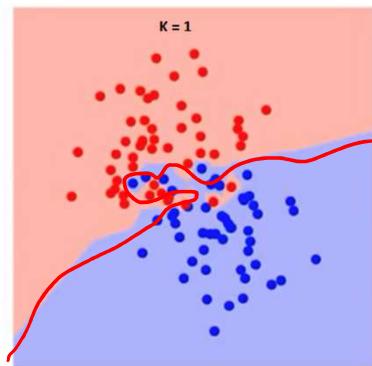
- Instance based classification
- An object is assigned to the class most common among its k nearest neighbours
- How to determine optimal k ?



Classification Algorithms in scope for this course

k-Nearest Neighbour Classifier (kNN)

- As k increases boundaries get smoother, but..



Classification Algorithms in scope for this course

Naïve Bayes Classifier

- Bayes Theorem
- Naïve Bayes Classifier
- Gaussian Naïve Bayes – continuous data
- Multinomial Naïve Bayes – feature vector representing frequency
- Bernoulli Naïve Bayes – Binary valued features

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$
$$P(y|x_1, \dots, x_n) = \frac{P(x_1|y)P(x_2|y)\dots P(x_n|y)P(y)}{P(x_1)P(x_2)\dots P(x_n)}$$

$$P(y|x_1, \dots, x_n) \propto P(y) \prod_{i=1}^n P(x_i|y)$$

$$y = \operatorname{argmax}_y P(y) \prod_{i=1}^n P(x_i|y)$$

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

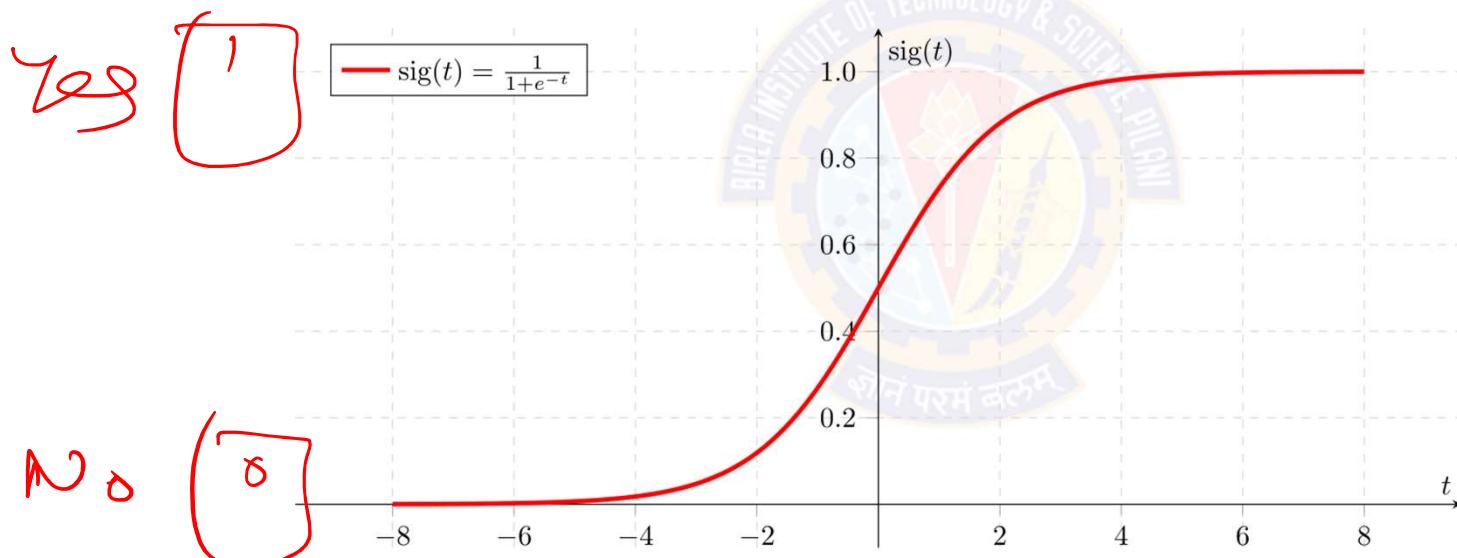
$$p(\mathbf{x} | C_k) = \frac{(\sum_i x_i)!}{\prod_i x_i!} \prod_i p_{ki}^{x_i}$$

$$p(\mathbf{x} | C_k) = \prod_{i=1}^n p_{ki}^{x_i} (1 - p_{ki})^{(1-x_i)}$$

Classification Algorithms in scope for this course

Logistic Regression

- Why “regression” for classification ?
- Logistic regression (logit) minimizes uncertainties, with differentiable decision function



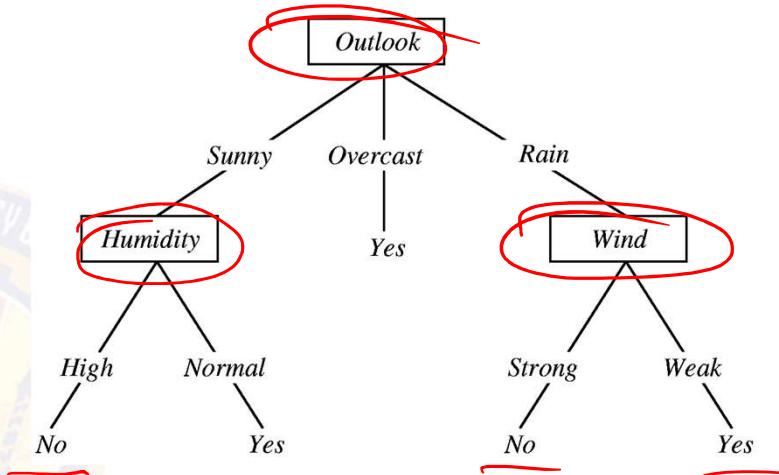
- Posterior probability – log of odds

$$\log \left[\frac{P(Y|X)}{1 - P(Y|X)} \right] = x_0 + x_1\beta_1 + x_2\beta_2 + \dots + x_k\beta_k + \varepsilon = f(x)$$

Classification Algorithms in scope for this course

Decision Tree

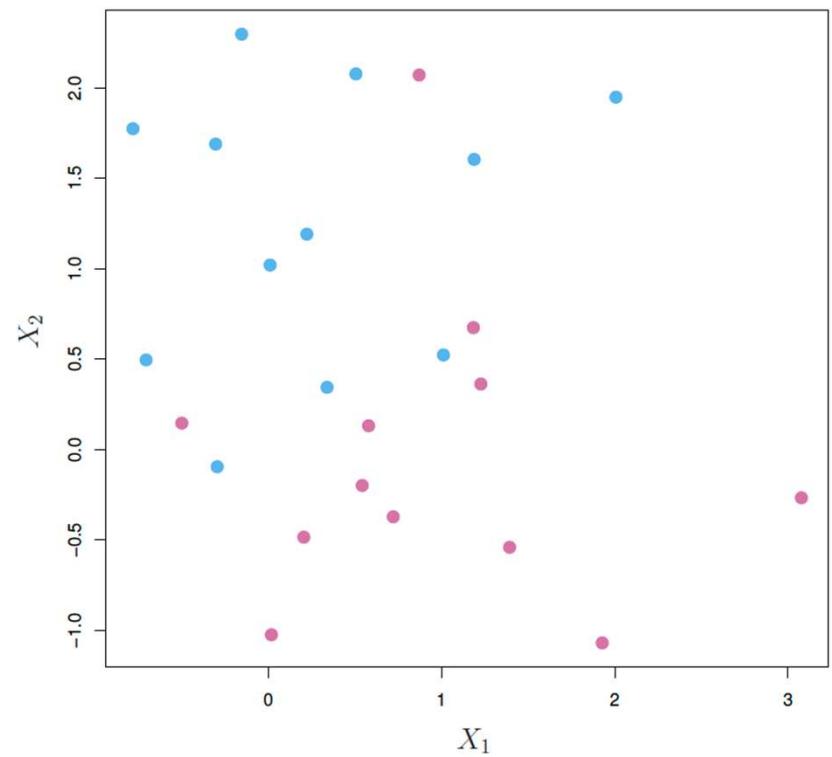
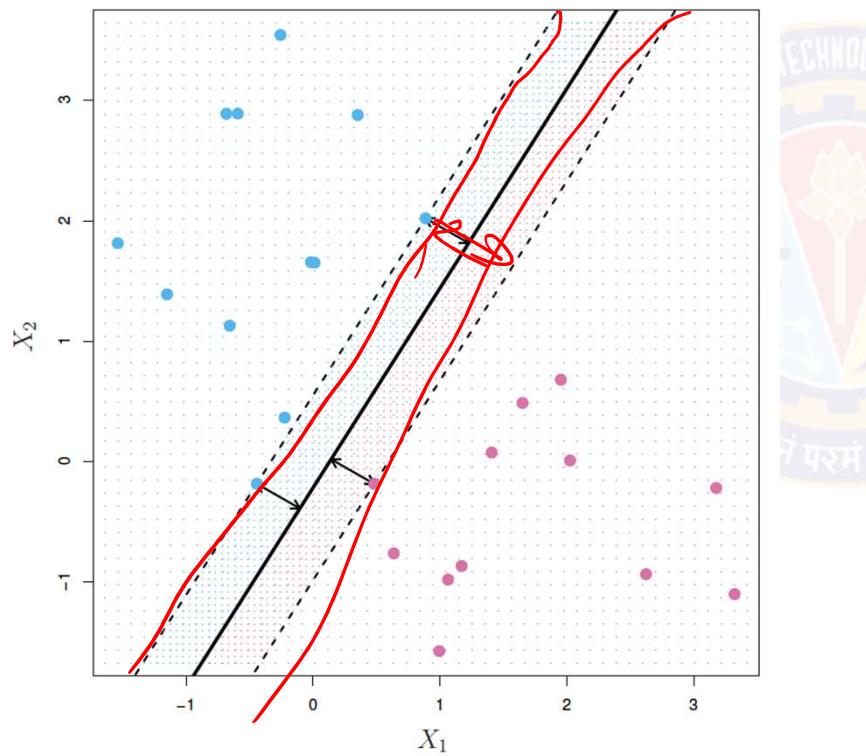
- Leaf represents class label
- Decisions are (internal) nodes/ branches
- Supervised learning for classification / regression
- Can handle categorical and continuous data
- Little preprocessing needed
- Prone to **overfitting**
 - Pruning
 - Early stop of tree building



Classification Algorithms in scope for this course

Support Vector Machines (SVM)

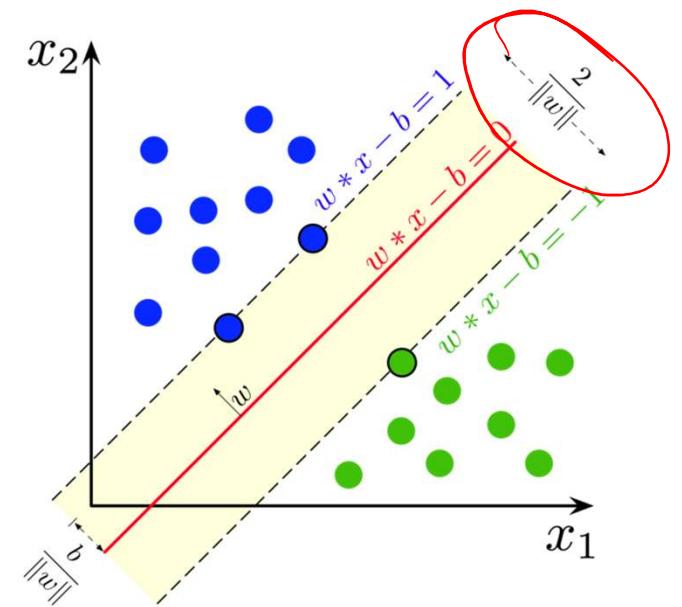
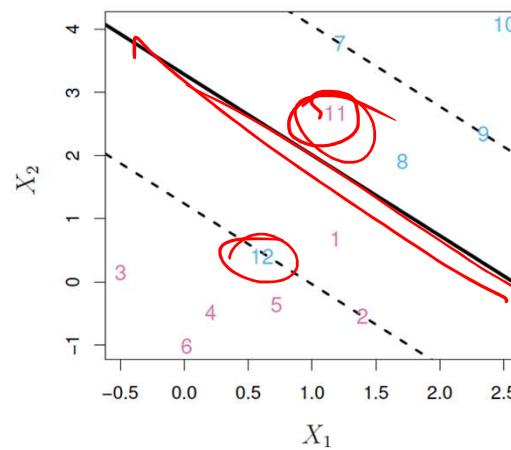
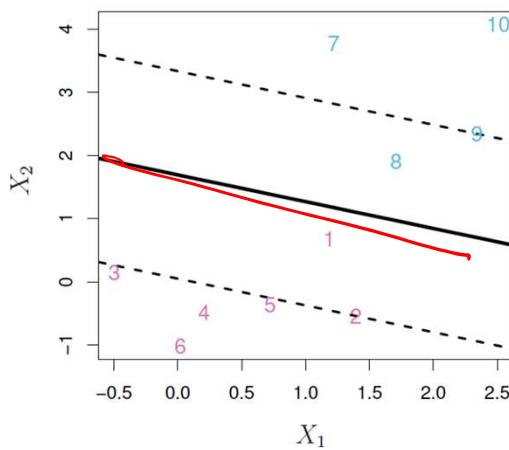
- Maximum margin classifier – does it work always ?



Classification Algorithms in scope for this course

Support Vector Machines (SVM) – Linear classifier

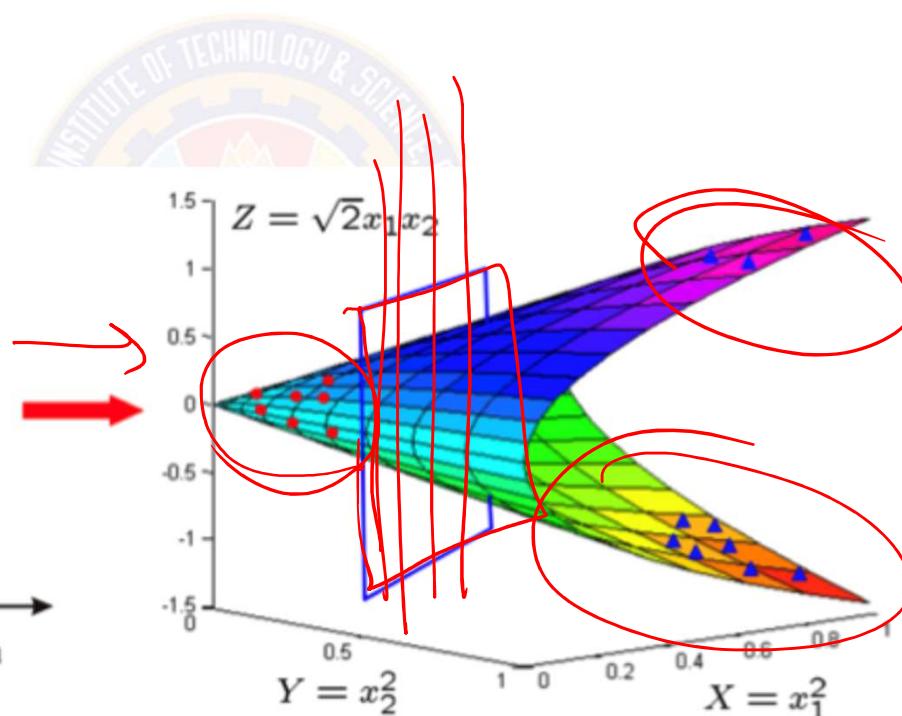
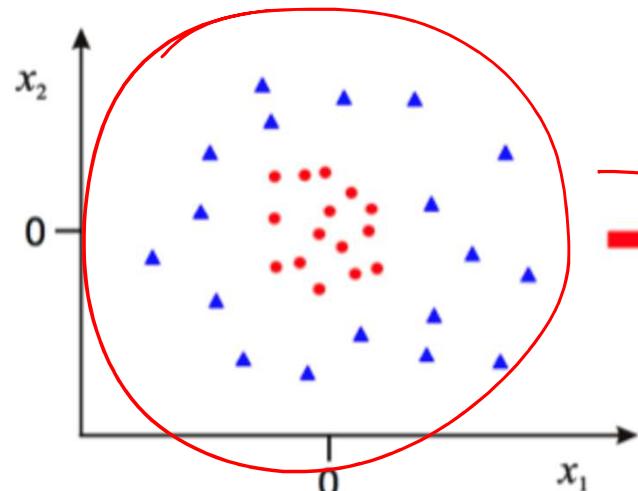
- Support Vector classifier – makes things better but not always
 - Hard-margin
 - Soft-margin



Classification Algorithms in scope for this course

Support Vector Machines (SVM) – Non-linear classifier (kernel trick)

- Transforms the feature space to allow fitment of a linear hyperplane
 - Polynomial
 - Radial basis function (rbf)
 - Hyperbolic Tangent (tanh)



Classification

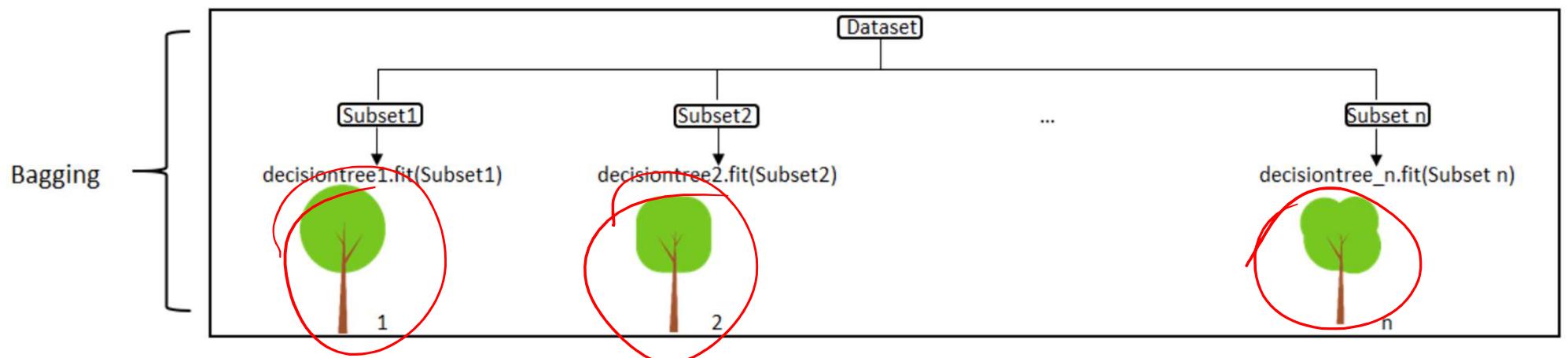
Ensemble Methods

- Bagging (Bootstrap AGGregating)
- Boosting
 - AdaBoost (Adaptive Boosting)
 - XGBoost (eXtreme Gradient Boosting)
- Random Forest



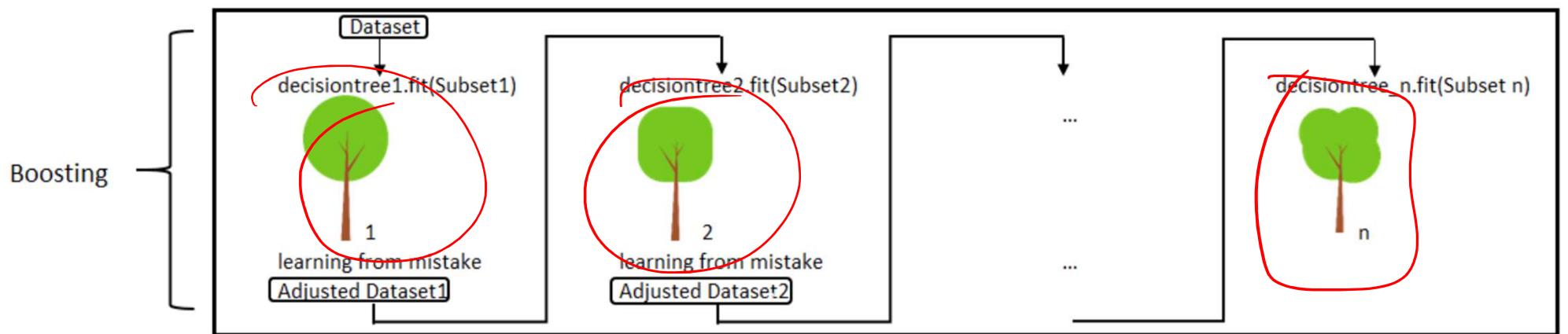
Classification

Bagging



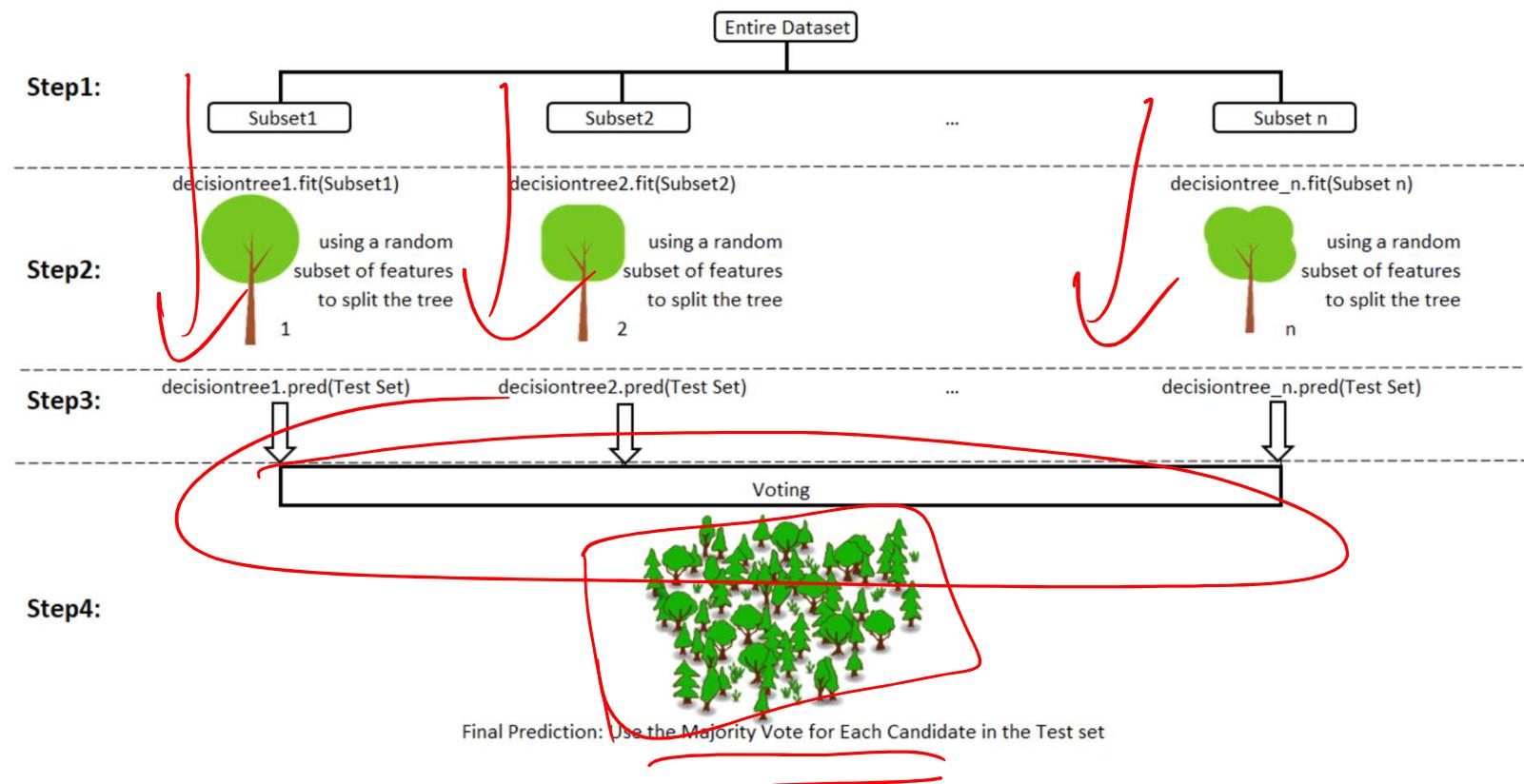
Classification

Boosting



Classification

Random Forest



In this segment

Applications of Classification – Case Study Overview

- Application of Classification
 - Healthcare
 - Finance
 - eCommerce
- Case study of the course - Overview



Applications of Classification – Case Study Overview

Applications of Classification - Healthcare

- Tumour classification – Malignant Benign (image based)
- Risk prediction for Diabetes, Heart disease, Alzheimer's etc.

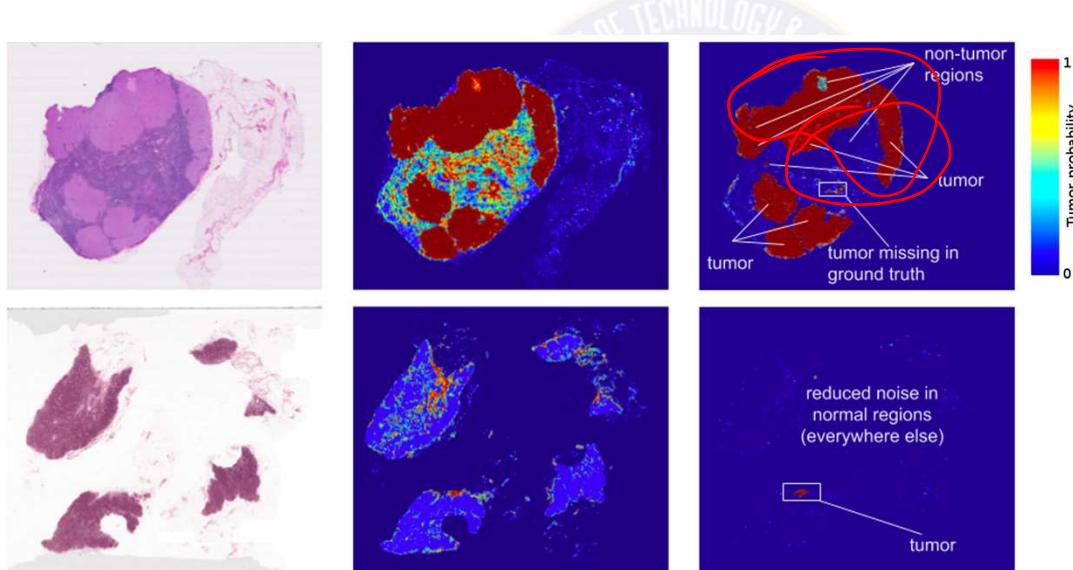


Image courtesy: Google AI blog

Applications of Classification – Case Study Overview

Applications of Classification - Finance

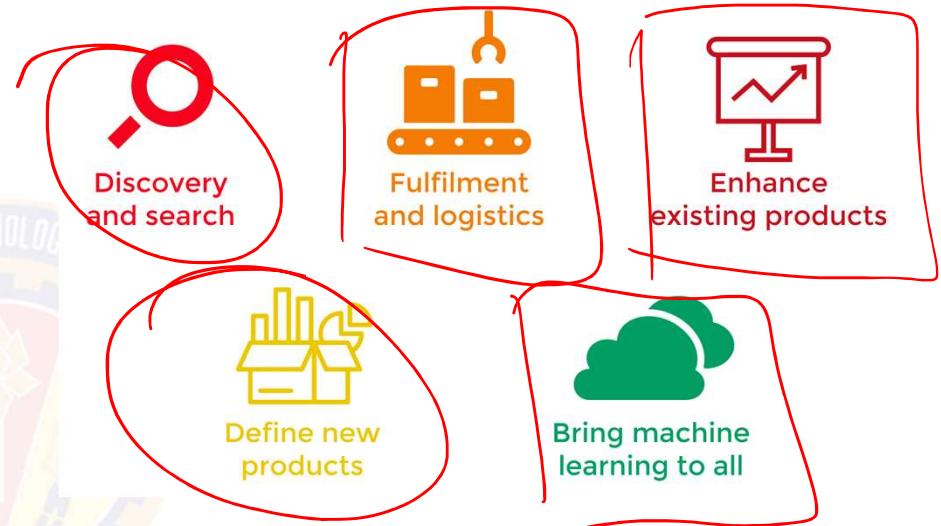
- Determining credit worthiness rating of corporates in lending
- Determine stock worthiness of the portfolio (buy/don't)
- Fraud detection in bulk transactions (stocks/loans etc.)



Applications of Classification – Case Study Overview

Applications of Classification - eCommerce

- Product Recommendations
- Smart search engines – image search
- Warehouse automation
- Sales forecasting
- Inventory forecast
- Dynamic pricing and price bundling



Applications of Classification – Case Study Overview

Case Study 1 (classroom) - Overview

- Diabetes Mellitus prediction in women
- Two-class problem (yes/no classification)
- Data set – 768 patients with health records and family history as parameters
- Input attribute set
 - Pregnancies
 - Glucose
 - Blood Pressure
 - Skin Thickness
 - Insulin
 - BMI
 - Age
- Label
 - Outcome (diabetic or non-diabetic)



Applications of Classification – Case Study Overview

Case Study 2 (exercises) - Overview

- Predict presence of Liver disease in a patient based on vitals
- Two-class problem (yes/no classification)
- Data set – Data 583 patients with health parameters
- Input attribute set
 - Age
 - Gender
 - Bilirubin (total & direct)
 - Alkaline Phosphatase
 - Alamine Aminotransferase
 - Aspartate Aminotransferase
 - Total Proteins
 - Albumin
 - Albumin & Globulin Ratio
- Label
 - Outcome (liver disease present or not)



In this segment

k-Nearest Neighbor Classifier

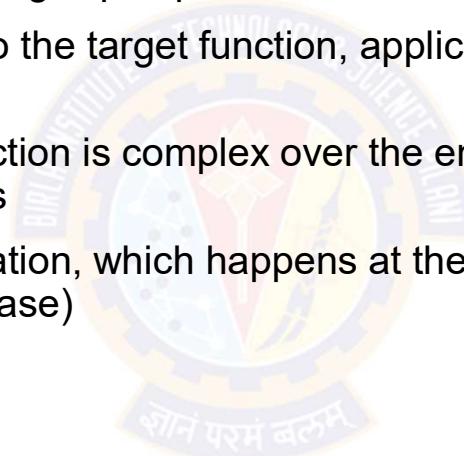
- Nearest Neighbor – Overview
- kNN classifier – Introduction
- kNN classifier – example
- Variations with k



k-Nearest Neighbor Classifier

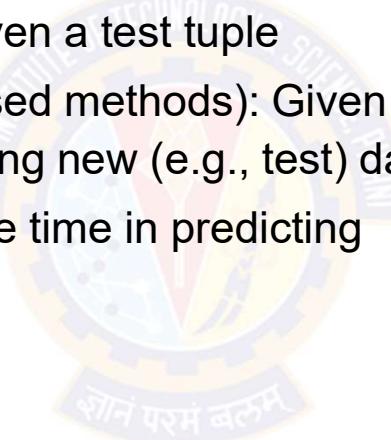
Instance based learning – Nearest Neighbor Classifiers

- Nearest Neighbour classifier is an instance based classifier
- Also called ‘lazy learning’, as learning is postponed until a new instance is encountered
- Constructs a local approximation to the target function, applicable (better suited) in the neighbourhood of new instance
- Suitable in cases where target function is complex over the entire input space, but easily describable in local approximations
- Caveat is the high cost of classification, which happens at the time of processing rather than before hand (there’s no training phase)



Lazy vs. Eager Learning

- Lazy learning (e.g., instance-based learning): Simply stores training data (or only minor processing) and waits until it is given a test tuple
- Eager learning (the above discussed methods): Given a set of training set, constructs a classification model before receiving new (e.g., test) data to classify
- Lazy: less time in training but more time in predicting



k-Nearest Neighbor Classifier

kNN Classifier - Introduction

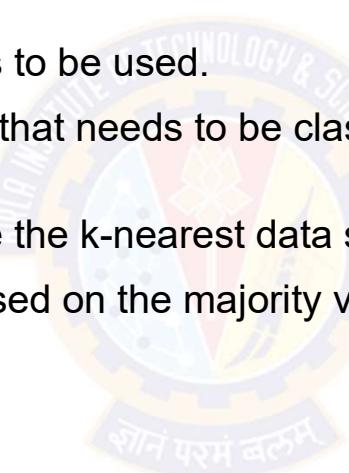
- Considers all instances as members of n-dimensional space
- Nearest neighbours of an instance is determined based on Euclidean distance
- Distance between two n-dimensional instances x_i and x_j is given by:

$$d(x_i, x_j) \equiv \sqrt{\sum_{r=1}^n (a_r(x_i) - a_r(x_j))^2}$$

- For NN classifier, target function can be discrete or continuous

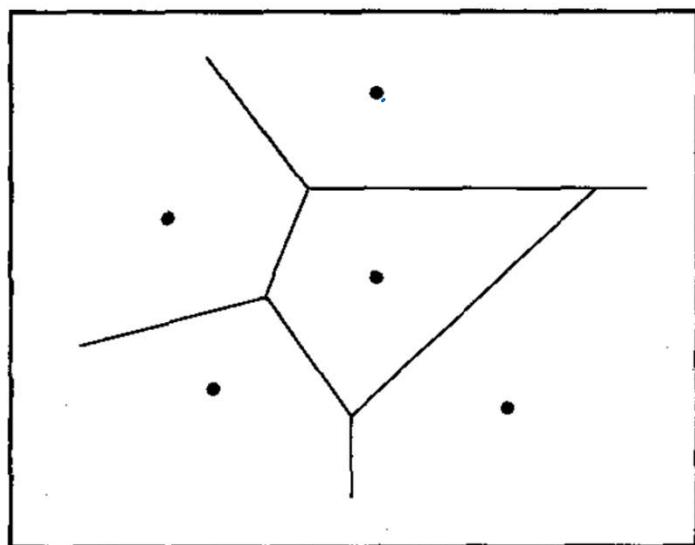
Steps involved in kNN

- Divide the data into training and test data.
- Select a value K.
- Determine which distance function is to be used.
- Choose a sample from the test data that needs to be classified and compute the distance to its n training samples.
- Sort the distances obtained and take the k-nearest data samples.
- Assign the test class to the class based on the majority vote of its k neighbours.

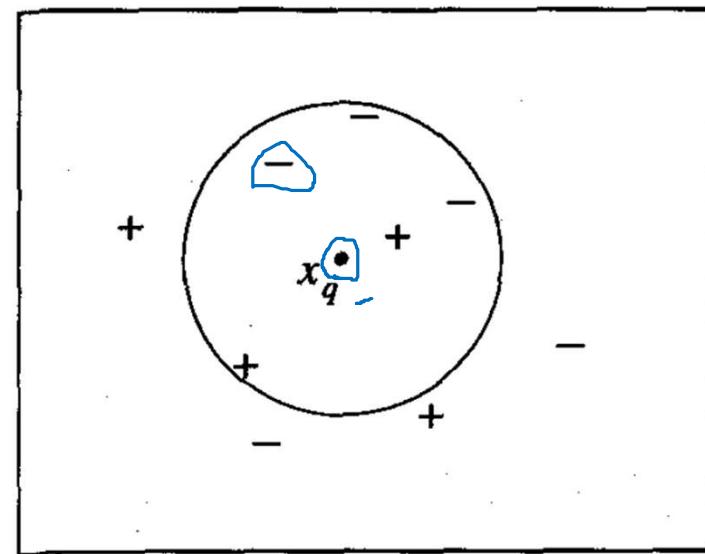


k-Nearest Neighbor Classifier

kNN Classifier – Example(s)



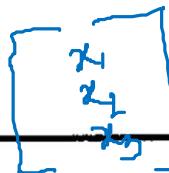
K=1



K=5

k-Nearest Neighbor Classifier

kNN Classifier - Algorithm



Training algorithm:

- For each training example $\langle x, f(x) \rangle$, add the example to the list *training_examples*

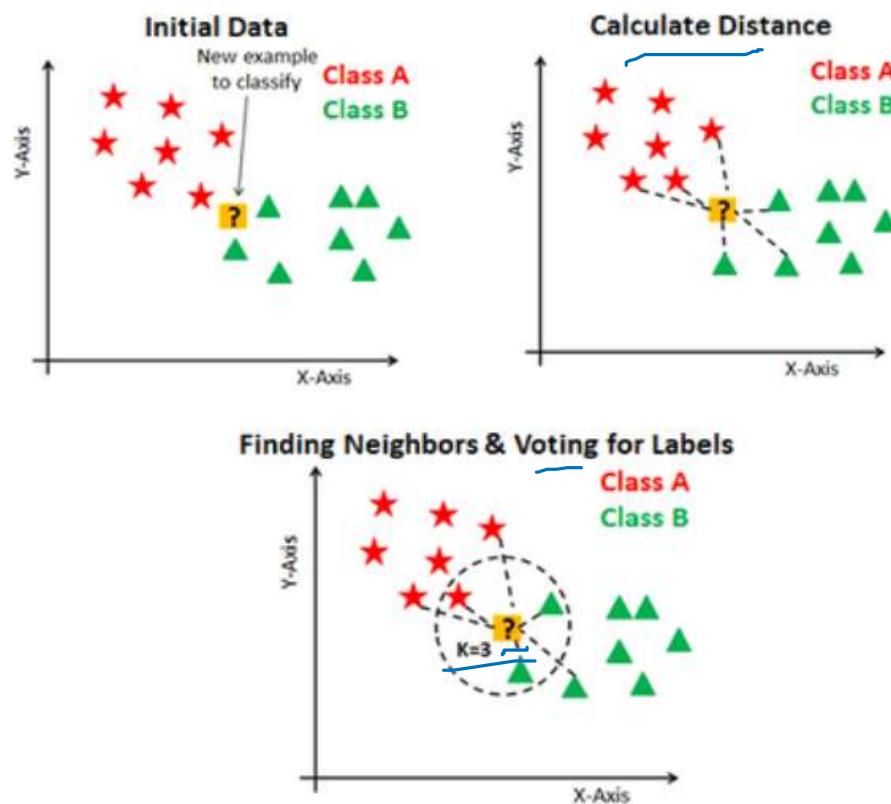
Classification algorithm:

- Given a query instance x_q to be classified,
 - Let $x_1 \dots x_k$ denote the k instances from *training_examples* that are nearest to x_q
 - Return

$$\hat{f}(x_q) \leftarrow \operatorname{argmax}_{v \in V} \sum_{i=1}^k \delta(v, f(x_i))$$

where $\delta(a, b) = 1$ if $a = b$ and where $\delta(a, b) = 0$ otherwise.

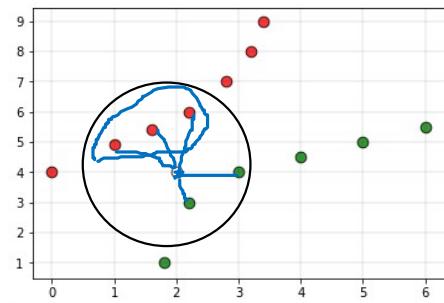
k-Nearest Neighbor Classifier



Source: DataCamp

Challenges in k-NN

- Performance of a classifier largely depends on the of the hyperparameter k
 - Choosing smaller values for K, noise can have a higher influence on the result.
 - Larger values of k are computationally expensive
- Assigning the class labels can be tricky. For example, in the below case, for (k=5) the point is closer to 'green' classification, but gets classified as 'red' due to higher red votes/majority voting to 'red'



k-Nearest Neighbor Classifier- Distance weighed

- This refinement adds a "weight" factor to kNN algorithm
- Weight would be based on the distance (ex: inverse square of distance from query instance), which gives more weightage to the closer neighbours
- Modified function would be:

$$\hat{f}(x_q) \leftarrow \operatorname{argmax}_{v \in V} \sum_{i=1}^k w_i \delta(v, f(x_i))$$

where $w_i \equiv \frac{1}{d(x_q, x_i)^2}$

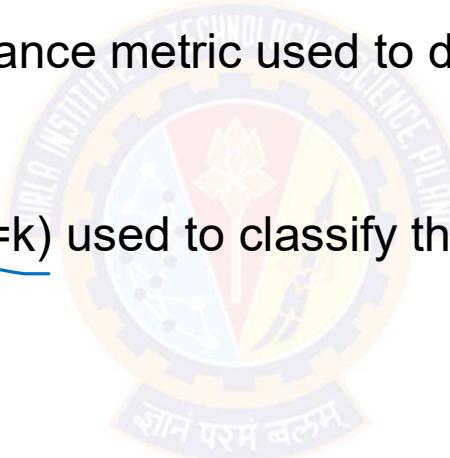
$\cancel{x_q, x_i}$

- This method is robust to noisy training data and effective for large set of training data
- kNN is the well-known “curse of dimensionality”
 - Can be overcome by weighing each attribute differently while calculating distances or eliminating irrelevant attributes

Factors that affect kNN Performance

Performance of the K-NN algorithm is influenced by the following factors:

- The distance function or distance metric used to determine the nearest neighbours.
- The number of neighbours (=k) used to classify the new example.



PROS and CONS of kNN

Pros

- The K-NN algorithm is very easy to implement.
- Nearly optimal in the large sample limit.
- Uses local information, which can yield highly adaptive behaviour.

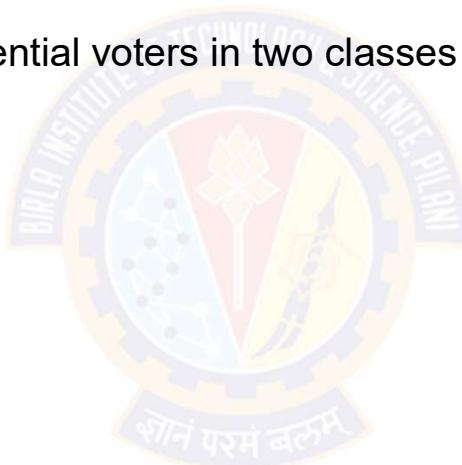


Cons

- Large storage requirements.
- Computationally intensive recall.
- Highly susceptible to the curse of dimensionality.
- Consider all attributes of instances to retrieve similar training examples from memory

Real world applications

- Finance — financial institutes will predict the credit rating of customers.
- Healthcare
- Political Science — classifying potential voters in two classes will vote or won't vote.
- Handwriting detection.
- Image Recognition.



In this segment

Measures of prediction accuracies of classifiers

- Metrics
 - Precision
 - Recall
 - F1
- ROC Curve – AUC of ROC
- Generating ROC Curve



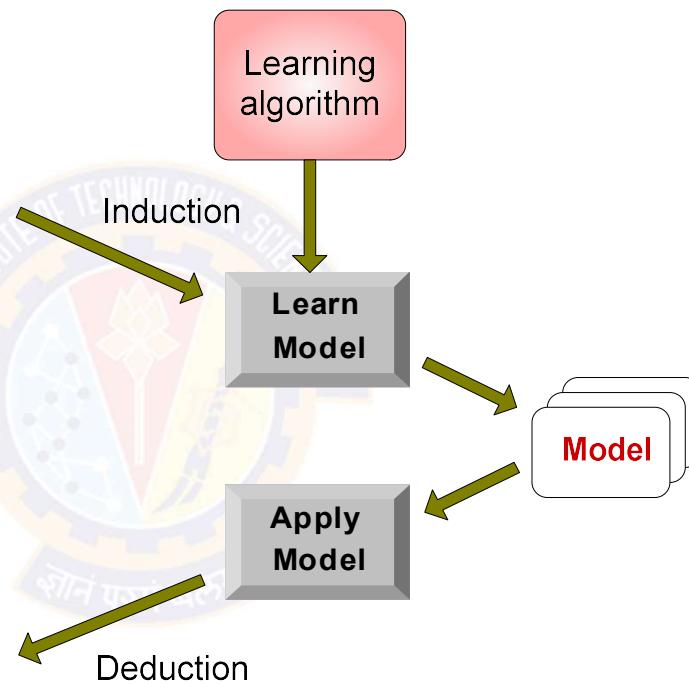
Classification

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set

Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set



Confusion Matrix

- **True Positive (TP):** It refers to the number of predictions where the classifier correctly predicts the positive class as positive.
- **True Negative (TN):** It refers to the number of predictions where the classifier correctly predicts the negative class as negative.
- **False Positive (FP):** It refers to the number of predictions where the classifier incorrectly predicts the negative class as positive.
- **False Negative (FN):** It refers to the number of predictions where the classifier incorrectly predicts the positive class as negative.

Predicted class ->	C_1	$\neg C_1$
Actual class ↓		
C_1	True Positives (TP)	False Negatives (FN)
$\neg C_1$	False Positives (FP)	True Negatives (TN)

Classifier Evaluation Metrics: Accuracy, Error Rate,

- **Classifier Accuracy**, or recognition rate: percentage of test set tuples that are correctly classified

$$\text{Accuracy} = (\text{TP} + \text{TN})/\text{All}$$

most effective when the class distribution is relatively balanced

- **Classification Error/ Misclassification rate:** $1 - \text{accuracy}$, or
 $= (\text{FP} + \text{FN})/\text{All}$

A\P	C	-C	
C	TP	FN	P
-C	FP	TN	N
	P'	N'	All

Accuracy

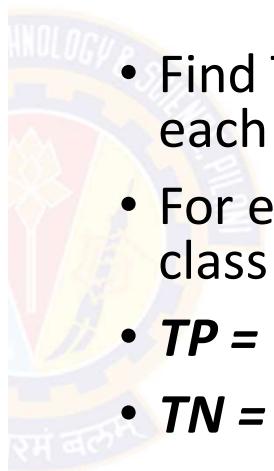
- Most widely-used metric:

		PREDICTED CLASS	
		Class=Yes	Class>No
ACTUAL CLASS	Class=Yes	a (TP)	b (FN)
	Class>No	c (FP)	d (TN)

$$\text{Accuracy} = \frac{a + d}{a + b + c + d} = \frac{TP + TN}{TP + TN + FP + FN}$$

Multiclass

		True Class		
		Apple	Orange	Mango
Predicted Class	Apple	7	8	9
	Orange	1	2	3
	Mango	3	2	1



- Find TP, TN, FP and FN for each individual class.
- For example, if we take class Apple,
- $TP = 7$
- $TN = (2+3+2+1) = 8$
- $FP = (8+9) = 17$
- $FN = (1+3) = 4$

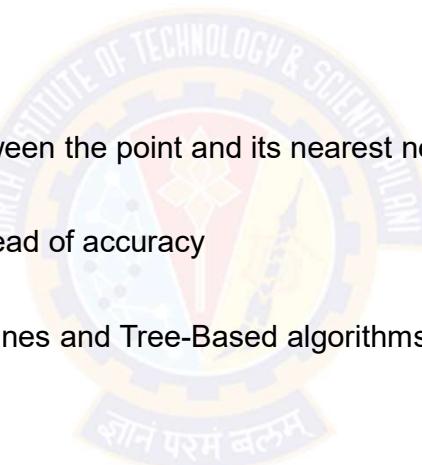
Class Imbalance Problem

- Find needle in haystack
- Lots of classification problems where the classes are skewed (more records from one class than another)
 - Credit card fraud
 - Intrusion detection
 - Defective products in manufacturing assembly line



Approaches to solve Class imbalance problem

- Up-sample minority class
 - randomly duplicating observations from a minority class
- Down-sample majority class
 - removing random observations.
- Generate Synthetic Samples
 - new samples based on the distances between the point and its nearest neighbors
- Change the performance metric
 - Use Recall, Precision or ROC curves instead of accuracy
- Try different algorithms
 - Some algorithms as Support Vector Machines and Tree-Based algorithms are better to work with imbalanced classes.



Problem with Accuracy

- Consider a 2-class problem
 - Number of Class NO examples = 990
 - Number of Class YES examples = 10
- If a model predicts everything to be class NO, accuracy is $990/1000 = 99\%$
 - This is misleading because the model does not detect any class YES example
 - Detecting the rare class is usually more interesting (e.g., frauds, intrusions, defects, etc)

Alternative Measures

		PREDICTED CLASS	
		Class=Yes	Class>No
ACTUAL CLASS	Class=Yes	a (TP)	b (FN)
	Class>No	c (FP)	d (TN)

$$\text{Precision (p)} = \frac{a}{a + c} \quad precision = \frac{TP}{TP + FP}$$

$$\text{Recall (r)} = \frac{a}{a + b} \quad recall = \frac{TP}{TP + FN}$$

$$\text{F - measure (F)} = \frac{2rp}{r + p} = \frac{2a}{2a + b + c}$$

Alternative Measures

		PREDICTED CLASS	
ACTUAL CLASS		Class=Yes	Class>No
	Class=Yes	10	0
	Class>No	10	980

$$\text{Precision (p)} = \frac{10}{10+10} = 0.5$$

$$\text{Recall (r)} = \frac{10}{10+0} = 1$$

$$\text{F - measure (F)} = \frac{2 * 1 * 0.5}{1 + 0.5} = 0.62$$

$$\text{Accuracy} = \frac{990}{1000} = 0.99$$

		PREDICTED CLASS	
ACTUAL CLASS		Class=Yes	Class>No
	Class=Yes	1	9
	Class>No	0	990

$$\text{Precision (p)} = \frac{1}{1+0} = 1$$

$$\text{Recall (r)} = \frac{1}{1+9} = 0.1$$

$$\text{F - measure (F)} = \frac{2 * 0.1 * 1}{1 + 0.1} = 0.18$$

$$\text{Accuracy} = \frac{991}{1000} = 0.991$$

Example

Given below is a confusion matrix for medical data where the class values are yes and no for a class label attribute, cancer. Calculate precision and recall of the classifier.

Classes	yes	no	Total	Recognition (%)
yes	90	210	300	30.00
no	140	9560	9700	98.56
Total	230	9770	10,000	96.40

Confusion matrix for the classes *cancer = yes* and *cancer = no*.

Reference: *Data Mining: concepts and techniques, Han kamber, 3rd edition*

Measures of prediction accuracies of classifiers

Metrics - Basics

- True positive rate (TPR) or **sensitivity** is defined as the fraction of positive examples predicted correctly by the model

$$TPR = TP / (TP + FN)$$

- True negative rate (TNR) or **specificity** is defined as the fraction of negative examples predicted correctly by the model

$$TNR = TN / (TN + FP)$$

- False positive rate (FPR) is the fraction of negative examples predicted as a positive class

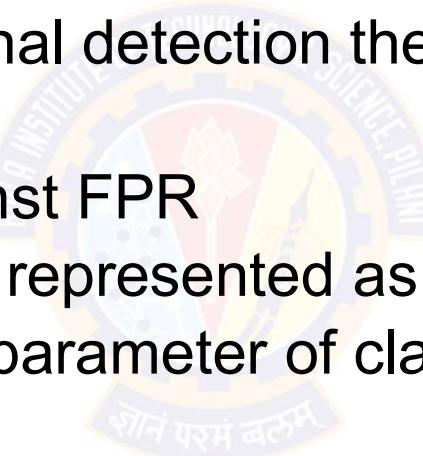
$$FPR = FP / (TN + FP)$$

- False negative rate (FNR) is the fraction of positive examples predicted as a negative class

$$FNR = FN / (TP + FN)$$

ROC (Receiver Operating Characteristic)

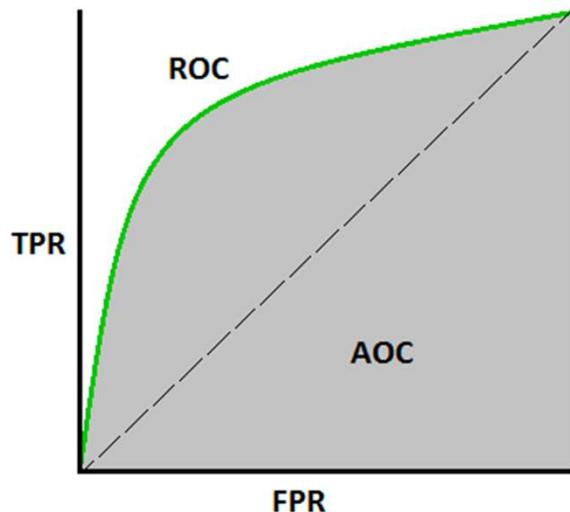
- A graphical approach for displaying trade-off between detection rate and false alarm rate
- Developed in 1950s for signal detection theory to analyze noisy signals
- ROC curve plots TPR against FPR
 - Performance of a model represented as a point in an ROC curve
 - Changing the threshold parameter of classifier changes the location of the point



ROC

Receiver Operating Characteristic (ROC) Curve

- ROC curve plots TPR and FPR, to graphically represent their trade-off
- Area Under Curve (AUC) of ROC – evaluates model performance on average
 - AUC of ROC = 1, for a perfect model
 - AUC of ROC = 0.5, if the model is random
- For model comparison, AUC of ROC should be larger for the model to be superior or better performing



$$TPR = TP/(TP + FN)$$

$$FPR = FP/(TN + FP)$$

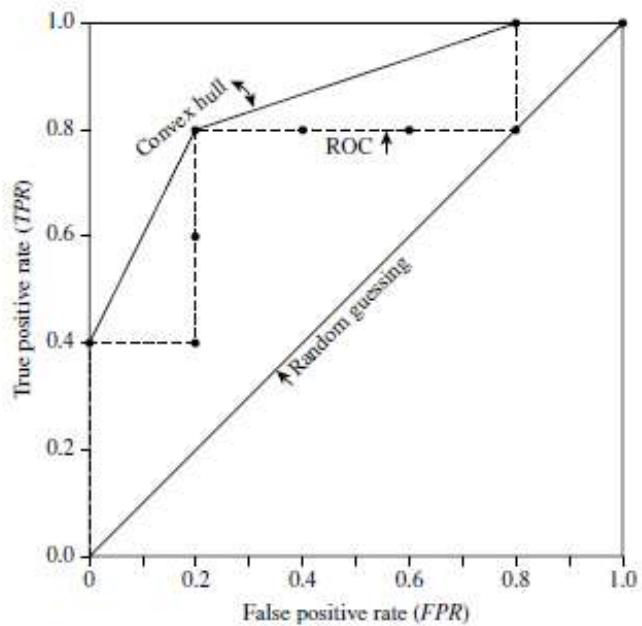
Example

- The table below shows the probability value (column 3) returned by a probabilistic classifier for each of the 10 tuples in a test set, sorted by decreasing probability order. The corresponding ROC is given on right hand side.

$$TPR = TP/(TP + FN)$$

$$FPR = FP/(TN + FP)$$

Tuple #	Class	Prob.	TP	FP	TN	FN	TPR	FPR
1	P	0.90	1	0	5	4	0.2	0
2	P	0.80	2	0	5	3	0.4	0
3	N	0.70	2	1	4	3	0.4	0.2
4	P	0.60	3	1	4	2	0.6	0.2
5	P	0.55	4	1	4	1	0.8	0.2
6	N	0.54	4	2	3	1	0.8	0.4
7	N	0.53	4	3	2	1	0.8	0.6
8	N	0.51	4	4	1	1	0.8	0.8
9	P	0.50	5	4	0	1	1.0	0.8
10	N	0.40	5	5	0	0	1.0	1.0

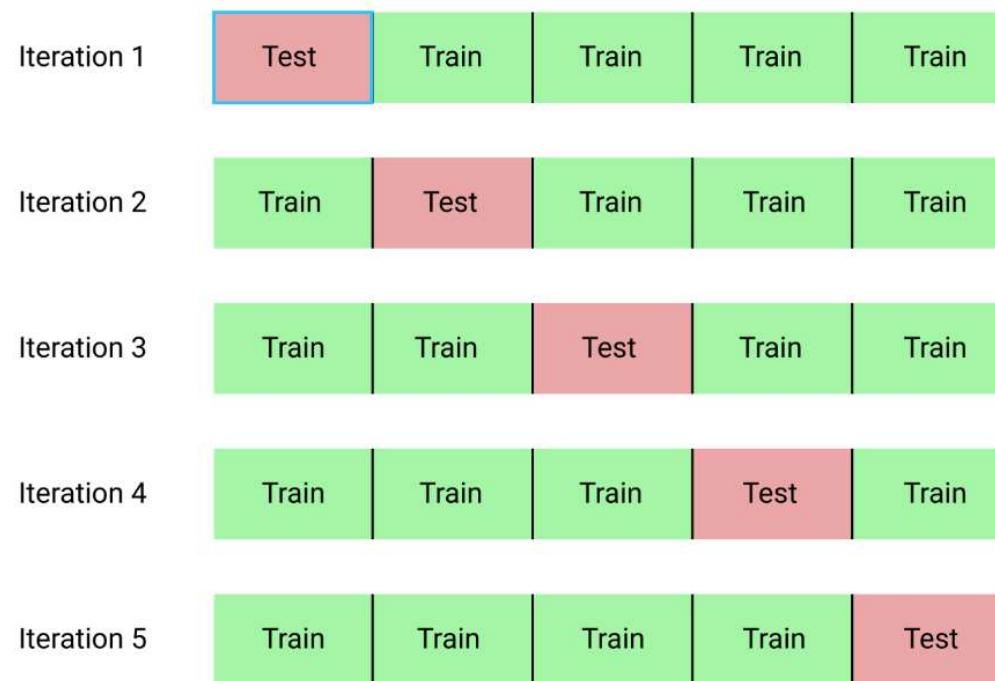


Evaluating Classifier Accuracy

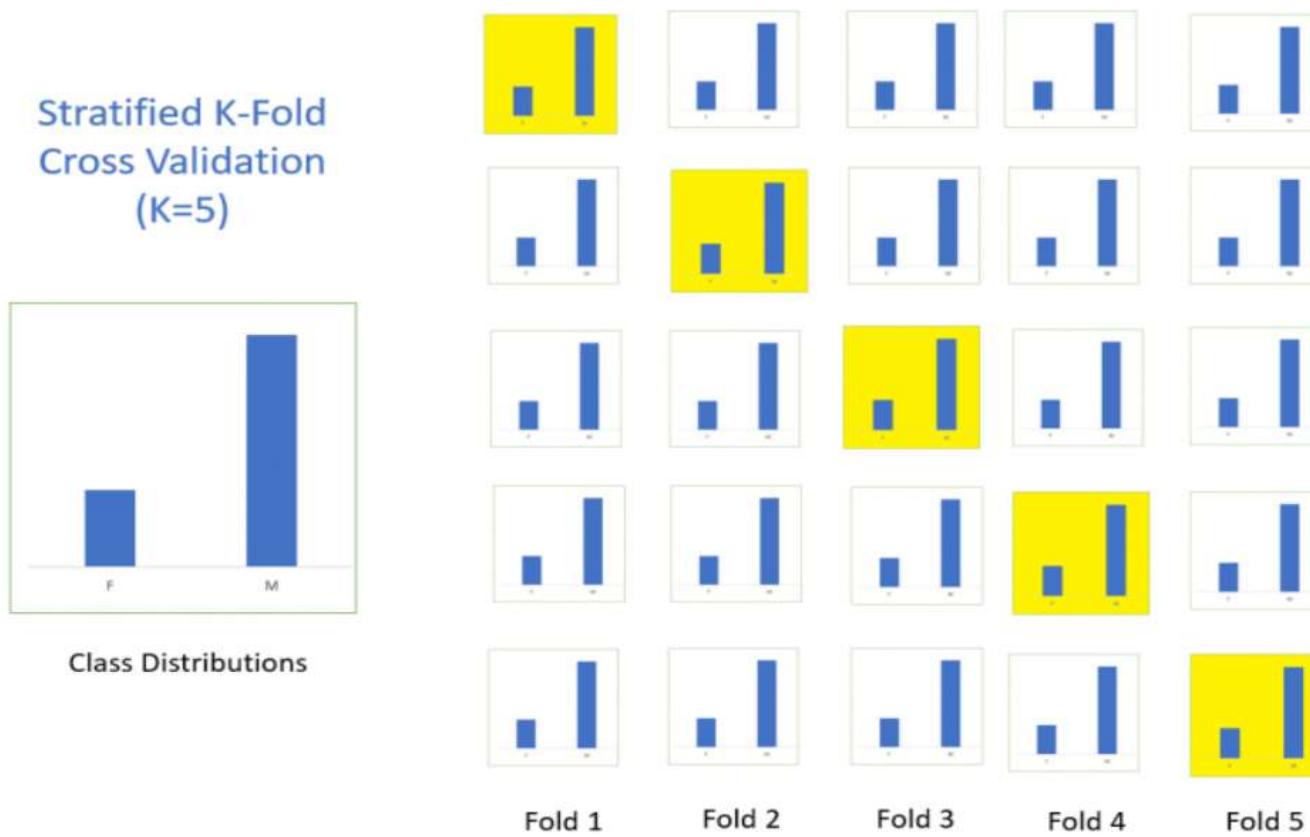
- **Holdout method**
 - Given data is randomly partitioned into two independent sets
 - Training set (e.g., 2/3) for model construction
 - Test set (e.g., 1/3) for accuracy estimation
 - Random sampling: a variation of holdout
 - Repeat holdout k times, accuracy = avg. of the accuracies obtained
- **Cross-validation (k -fold, where $k = 10$ is most popular)**
 - Randomly partition the data into k *mutually exclusive* subsets, each approximately equal size
 - At i -th iteration, use D_i as test set and others as training set
 - The Accuracy of the model is the average of the accuracy of each fold.

Cross Validation

k-Fold Cross Validation:



Stratified Cross Validation



In this segment

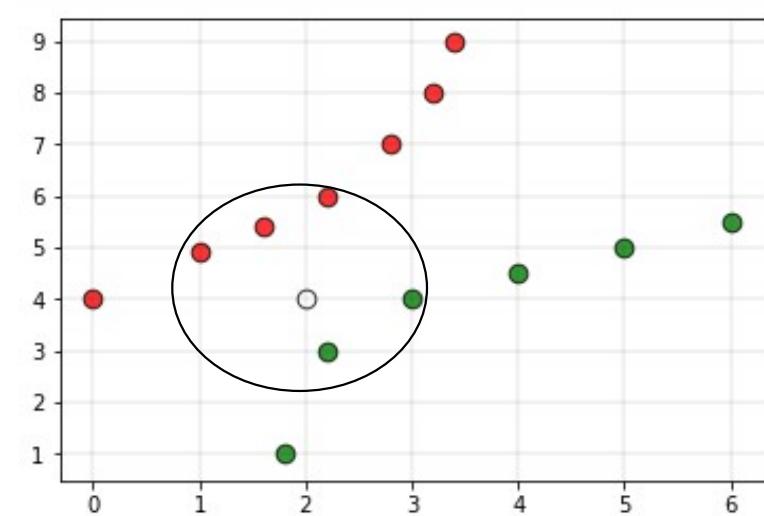
Finding optimal k for kNN classifiers

- The problem of finding K
- Elbow method – Optimal K



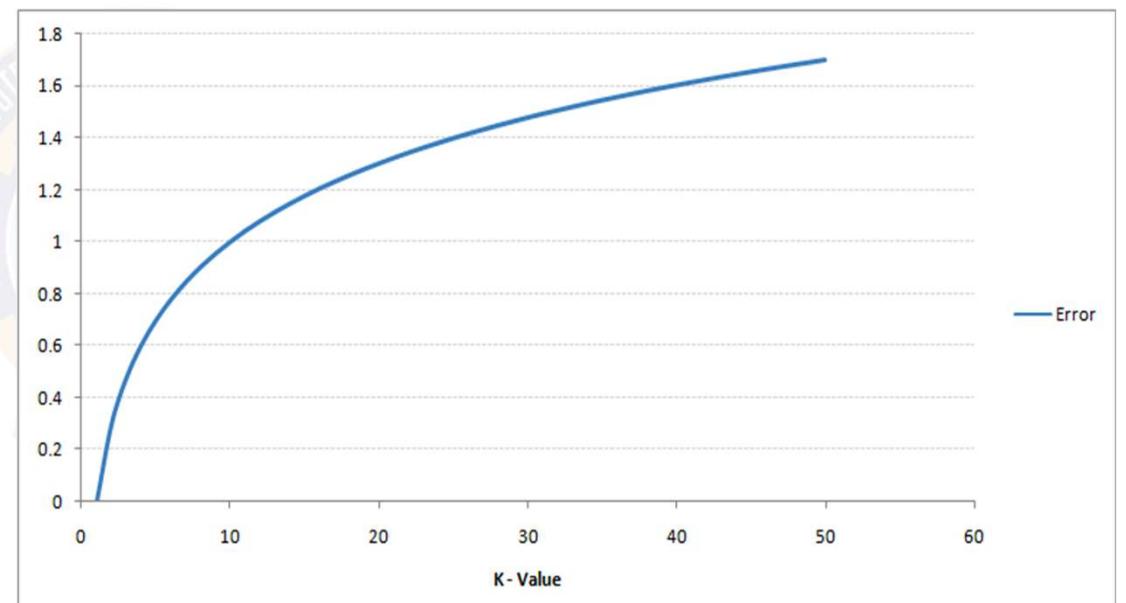
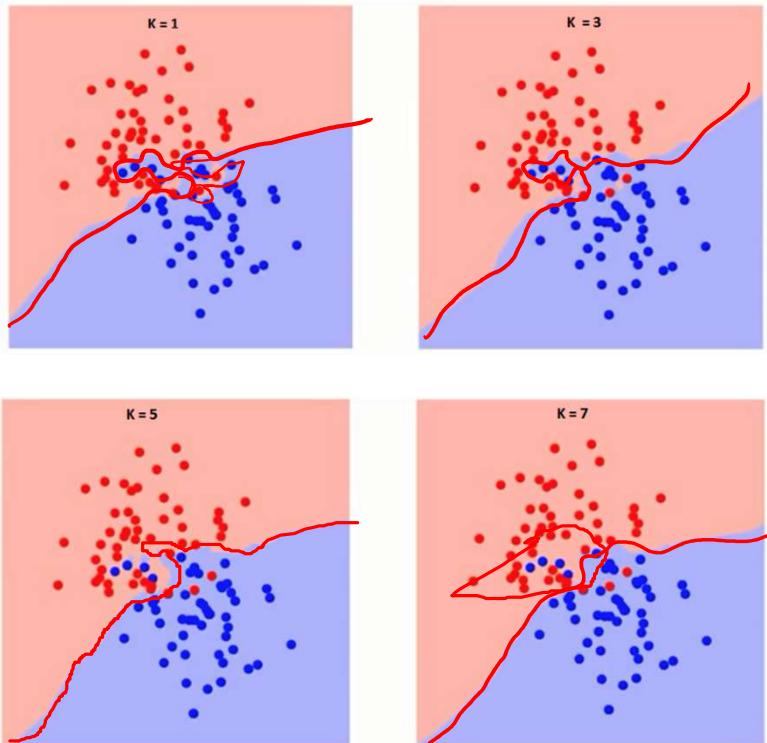
Various issues that affect the performance of kNN

- Performance of a classifier largely depends on the of the hyperparameter k
 - Choosing smaller values for K, noise can have a higher influence on the result.
 - Larger values of k are computationally expensive
- Assigning the class labels can be tricky. For example, in the below case, for (k=5) the point is closer to 'green' classification, but gets classified as 'red' due to higher red votes/majority voting to 'red'



Finding optimal k for kNN classifiers

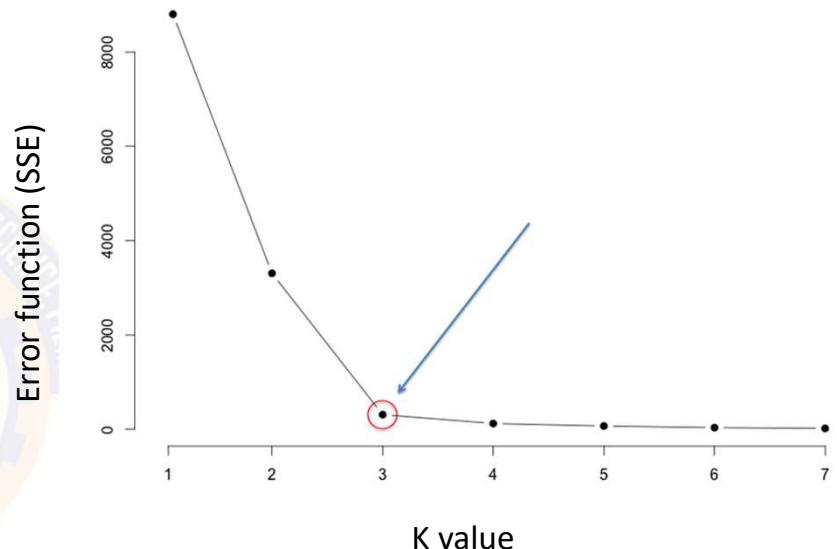
What is the ideal value of 'K'?



Finding optimal k for kNN classifiers

Finding K - Elbow method

- Compute sum of squares error (SSE) or any other error function for varying values of K (1 to a reasonable X) and plot against K
- In the plot, the elbow (see pic) gives the value of K beyond which the error function plot almost flattens
- As K approaches the total number of instances in the set, error function drops down to '0', but beyond optimal K, the model becomes too generic



In this segment

Probability Foundations

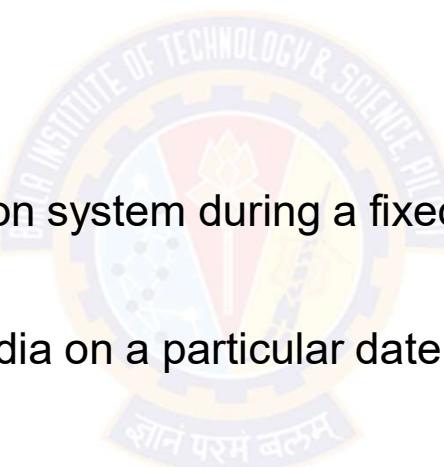
- Probability - Terminology
- Discrete Random Variable
- Continuous Random Variable
- Conditional Independence



Probability basics

Random Experiment

- A random experiment is an experiment or a process for which the outcome cannot be predicted with certainty
- Example:
 - Tossing a coin
 - number of calls to a communication system during a fixed length interval of time
 - Rolling a die
 - Number of dishwashers sold in India on a particular date



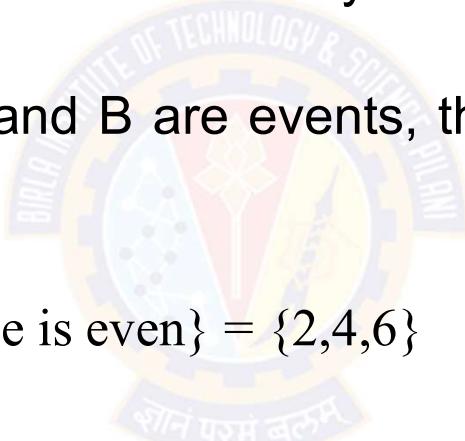
Sample Space:

- The sample space of a random experiment is the set of all possible outcomes. Denoted by omega Ω or sample space S
- Example: Random Experiment: Toss a fair coin once.
Sample Space: $\Omega = \{\text{Head, Tail}\}$

Probability basics

Events

- Events are the subsets of the sample space
 - Simple Events: If an event consists of only one outcome, it is a simple event
- Union and Intersection: If A and B are events, then $A \cup B$ and $A \cap B$ are also events.
- Examples:
 - $A = \{\text{the outcome that the dice is even}\} = \{2, 4, 6\}$
 - $B = \{\text{at least 2 tails}\}$



Outcome: A result of a random experiment.

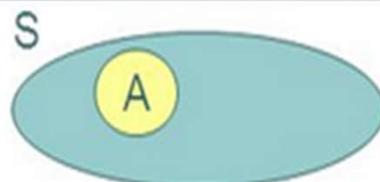
Sample Space: The set of all possible outcomes.

Event: A subset of the sample space.

Probability Theory

Let A be an event, then we denote

$P(A)$ the probability for A



It always hold that $0 \leq P(A) \leq 1$ $P(\emptyset) = 0$ $P(S) = 1$

Consider an experiment which has N **equally likely** outcomes, and let exactly n of these events correspond to the event A . Then

$$P(A) = \frac{n}{N} = \frac{\text{\# successful outcomes}}{\text{\# possible outcomes}}$$

Example:

Rolling a dice

$P(\text{even number})$

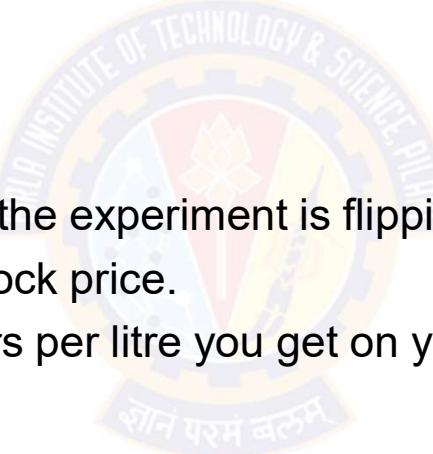
$$= \frac{3}{6} = \frac{1}{2}$$

Random Variable

- A **random variable**, usually written X , is a **variable** whose possible values are numerical outcomes of a **random** phenomenon or experiment.

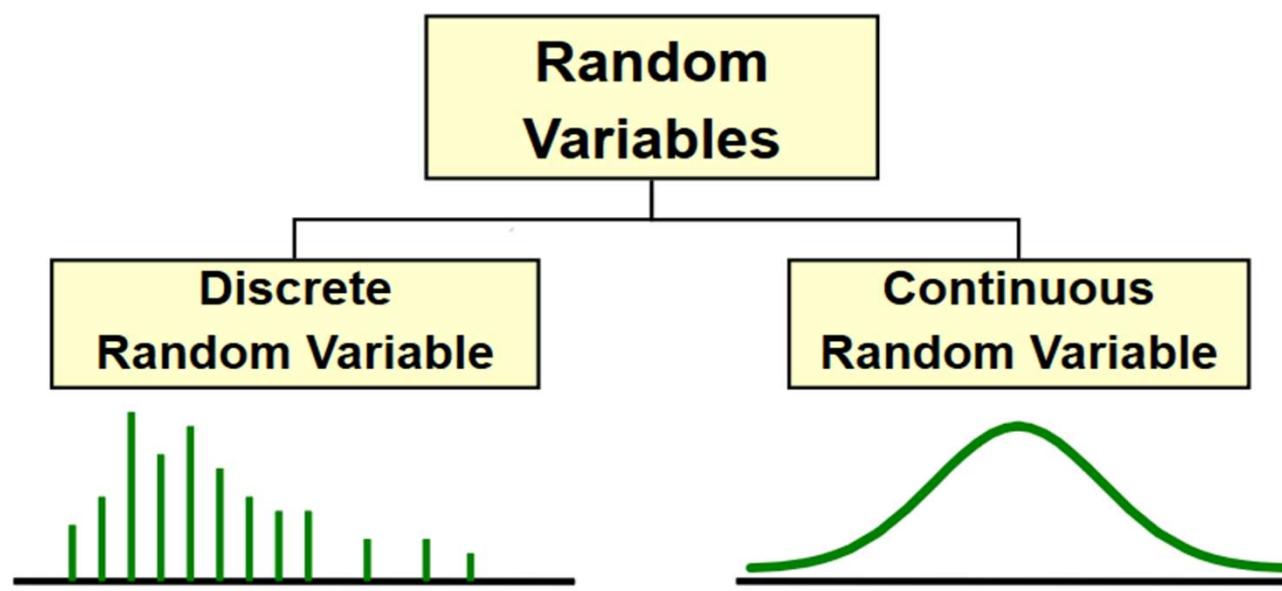
Examples

- ✓ X = number of heads when the experiment is flipping a coin 20 times.
- ✓ C = the daily change in a stock price.
- ✓ R = the number of kilometers per litre you get on your car during a family vacation.



Random Variable

Represents a possible numerical value from a random event



Random Variable

Discrete Random Variable

- one that takes on a ***countable*** number of values
- usually count data [Number of]
- list **all** possible outcomes without missing any of them

Example:

✓ X = sum of values on the roll of two dice:
 X has to be either 2, 3, 4, ..., or 12.

✓ Y = number of students in AIML:
 Y has to be 60,65,70

Random Variable

Continuous Random Variable

- Variable that takes on an uncountable number of values
- Usually measurement data [time, weight, distance, etc.]
- You can never list all possible outcomes even if you had an infinite amount of time

Example:

$X =$ time it takes you to drive home from work place: $X > 0$,
might be 30.1 minutes measured to the nearest tenth but in
reality the actual time is 30.100001..... minutes?)

Probability Theory

Notation

- A **random variable X** represents outcomes or states of the world.
- We will write $p(x)$ to mean $\text{Probability}(X = x)$
- **Sample space:** the space of all possible outcomes (may be discrete, continuous, or mixed)
- $p(x)$ is the **probability mass (density) function**
 - Assigns a number to each point in sample space
 - Non-negative, sums (integrates) to 1
 - Intuitively: how often does x occur, how much do we believe in x .

Probability Densities

Continuous Probability Distribution

Let X be a continuous rv. Then a *probability distribution or probability density function (pdf)* of X is a function $f(x)$ such that for any two numbers a and b ,

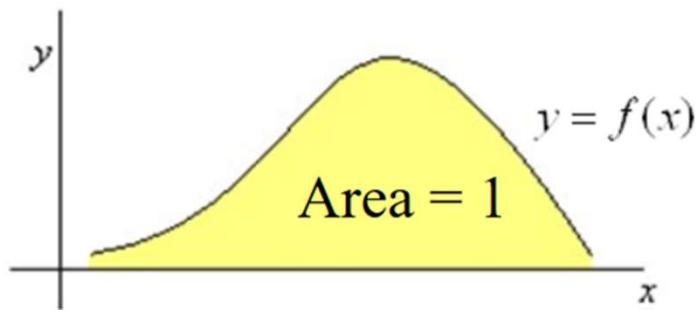
$$P(a \leq X \leq b) = \int_a^b f(x) dx$$

The graph of f is the *density curve*.

Probability Densities

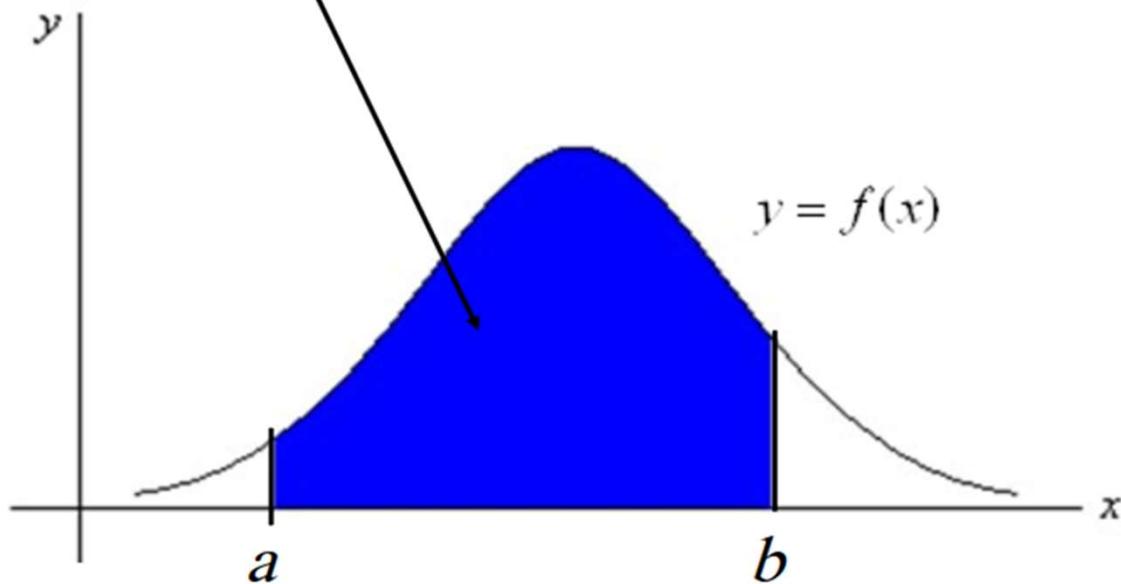
For $f(x)$ to be a pdf

1. $f(x) > 0$ for all values of x .
2. The area of the region between the graph of f and the x – axis is equal to 1.



Probability Densities

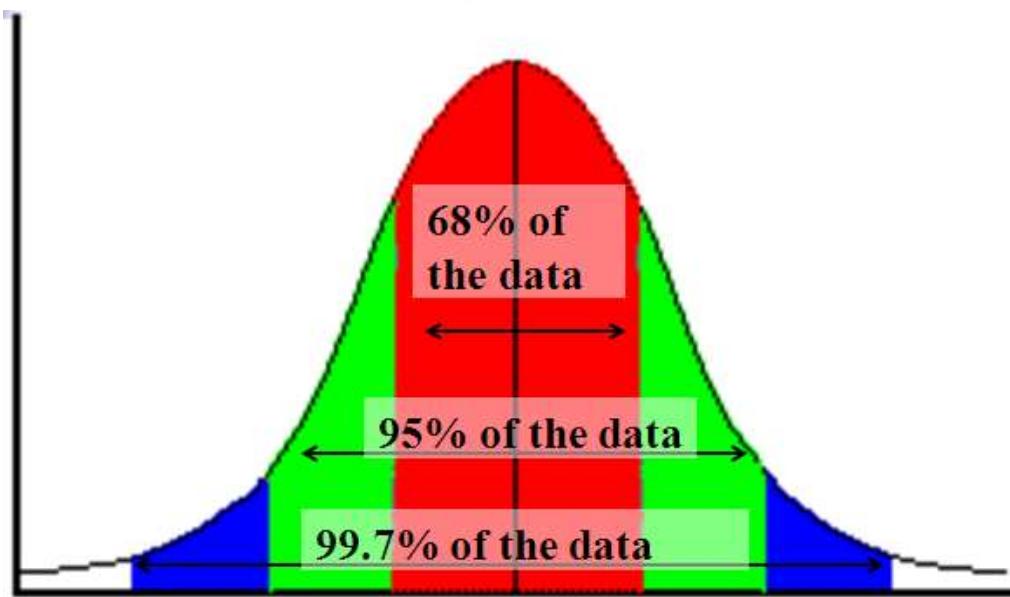
$P(a \leq X \leq b)$ is given by the area of the shaded region.



Gaussian Distribution

Empirical Rule:

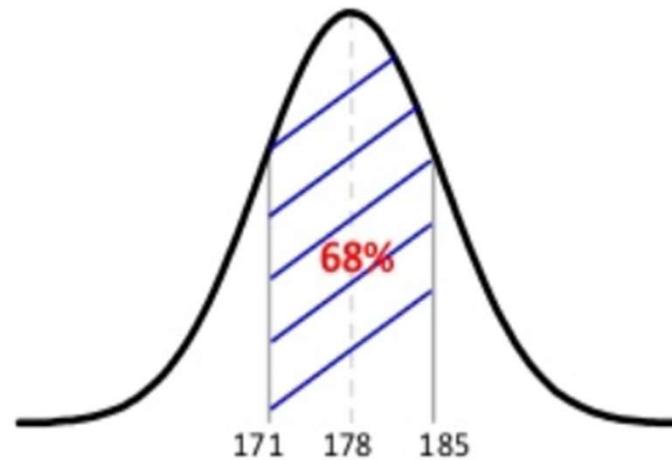
- For any normally distributed data:
 - 68% of the data fall within **1** standard deviation of the mean.
 - 95% of the data fall within **2** standard deviations of the mean.
 - 99.7% of the data fall within **3** standard deviations of the mean.



Gaussian Distribution

- Suppose that the heights of a sample men are normally distributed.
- The mean height is **178** cm and a standard deviation is **7** cm.

- **We can generalize that:**
 - **68%** of population are between **171** cm and **185** cm.
 - This might be a generalization, but it's true if the data is normally distributed.



Mean, Variance & Standard Deviation

- ✓ The mean of a discrete random variable is the **weighted average** of all of its values. The weights are the probabilities.
- ✓ This parameter is also called the expected value of X and is represented by $E(X)$.

$$E(X) = \mu = \sum_{\text{all } x} xP(x)$$

- ✓ The variance is

$$V(X) = \sigma^2 = \sum_{\text{all } x} (x - \mu)^2 P(x)$$

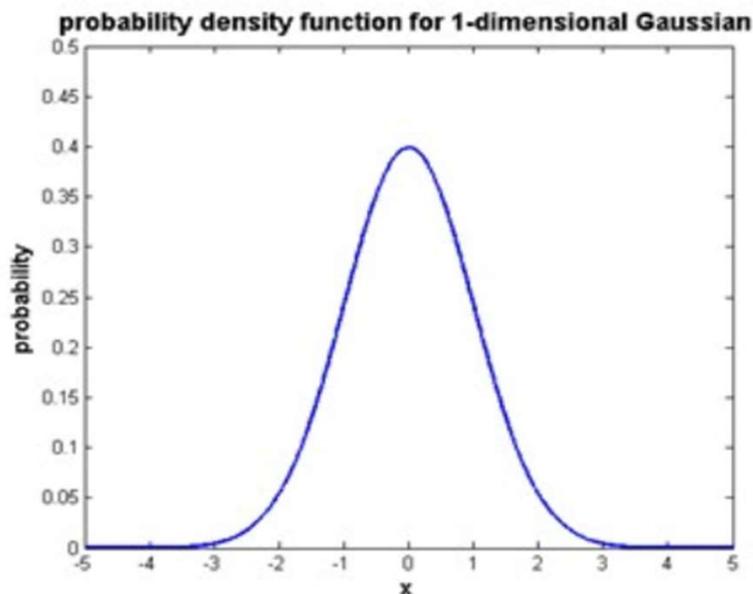
- ✓ The standard deviation is

$$\sigma = \sqrt{\sigma^2}$$

Gaussian Distribution

In one dimension

$$N(x | \mu, \sigma^2) = \frac{1}{(2\pi\sigma^2)^{1/2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$



Gaussian Distribution

In one dimension

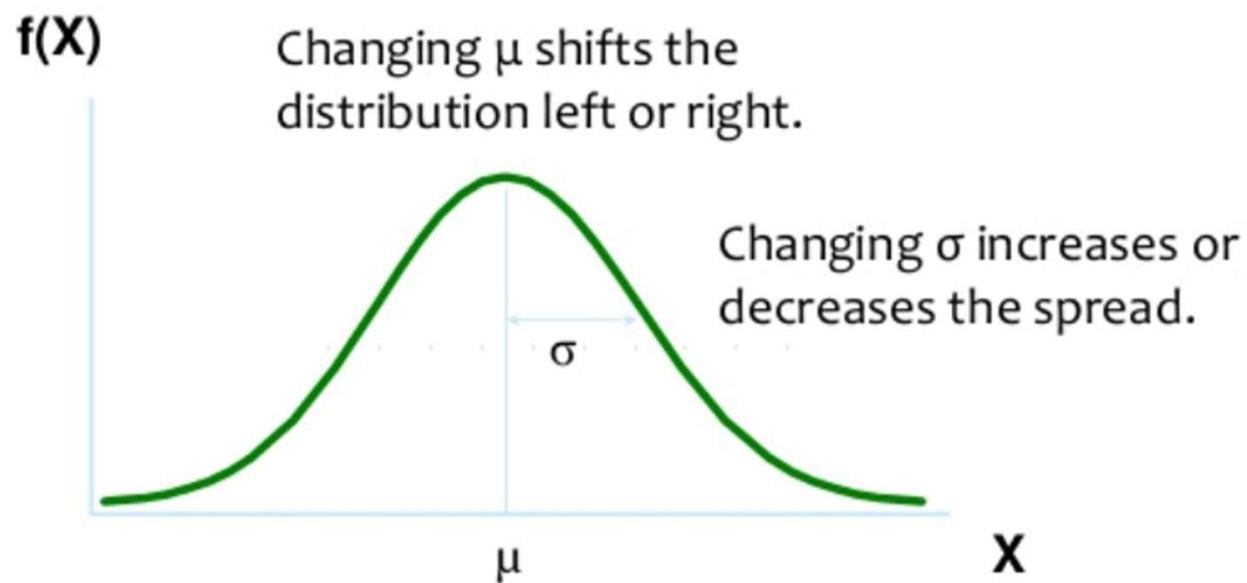
$$N(x | \mu, \sigma^2) = \frac{1}{(2\pi\sigma^2)^{1/2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Normalizing constant:
insures that distribution
integrates to 1

Causes pdf to decrease as
distance from center
increases

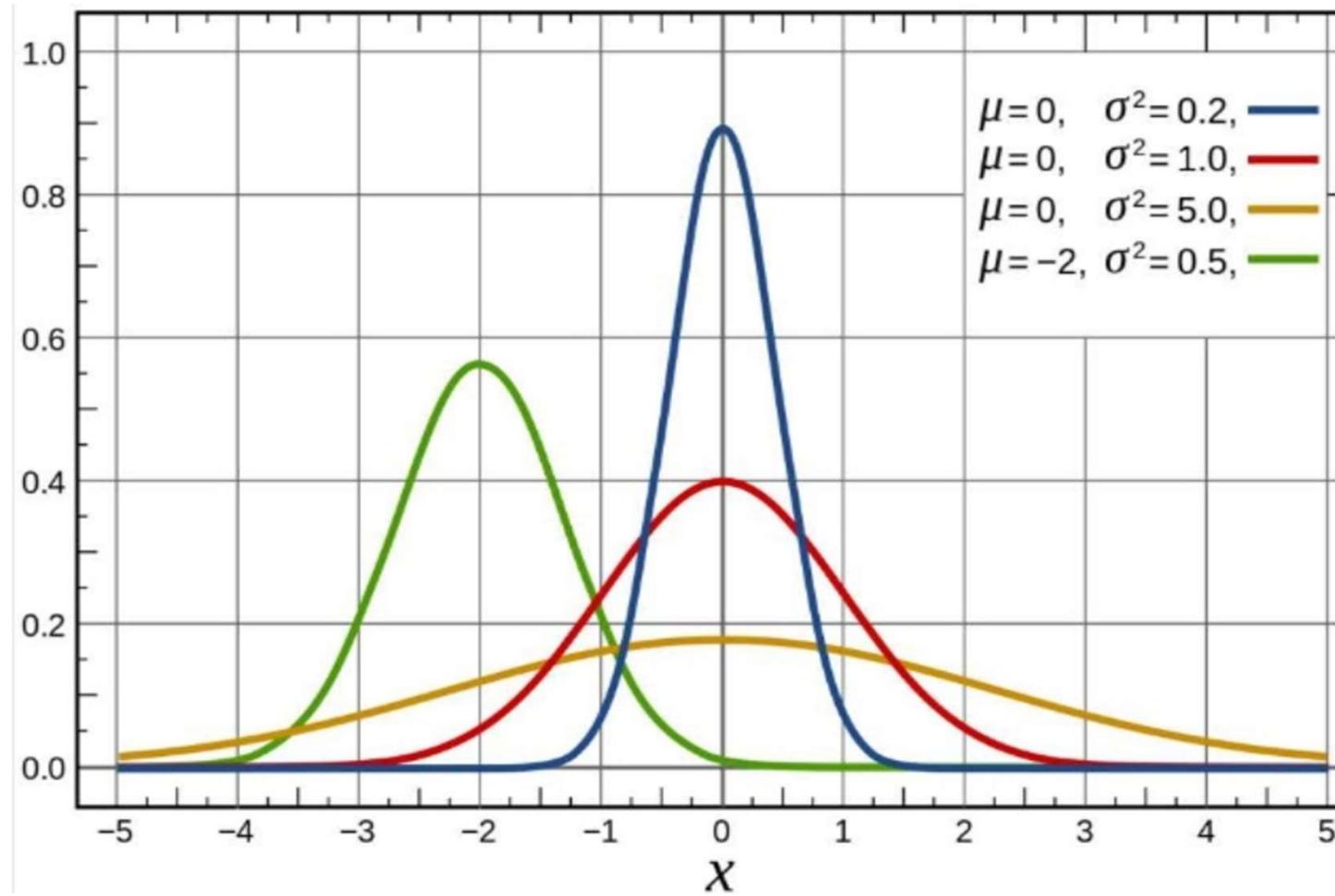
Controls width of curve

Gaussian Distribution



The normal curve is not a single curve but a family of curves, each of which is determined by its mean and standard deviation.

Gaussian Distribution

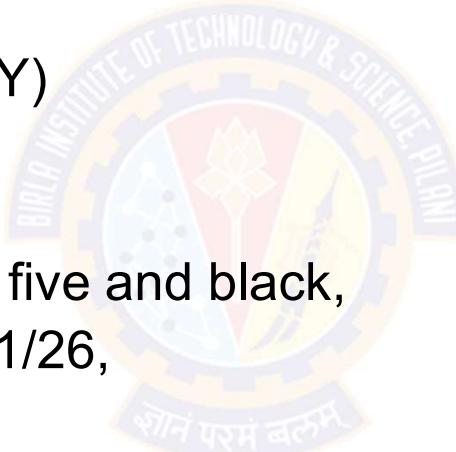


Joint Probability

- **Joint probability** is the probability of two events happening together. The two events are usually designated *event A* and *event B*. In probability terminology, it can be written as:
- $P(X \text{ and } Y)$ or $P(A \cap B)$ or $P(X, Y)$

Example:

- The probability that a card is a five and black,
 $p(\text{five and black}) = 2/52 = 1/26,$



(There are two black fives in a deck of 52 cards, the five of spades and the five of clubs)

Independence

- Two events A and B are independent if and only if $P(\underline{A \cap B}) = P(\underline{A})P(\underline{B})$.

$$\begin{aligned} P(A|B) &= \frac{P(A \cap B)}{P(B)} \\ &= \frac{P(A)P(B)}{P(B)} \\ &= P(A). \end{aligned}$$

- If two events A and B are independent and $P(B) \neq 0$, then $P(\underline{A|B}) = P(\underline{A})$.
- In general, for n events $\underline{A_1, A_2, \dots, A_n}$ to be independent we must have

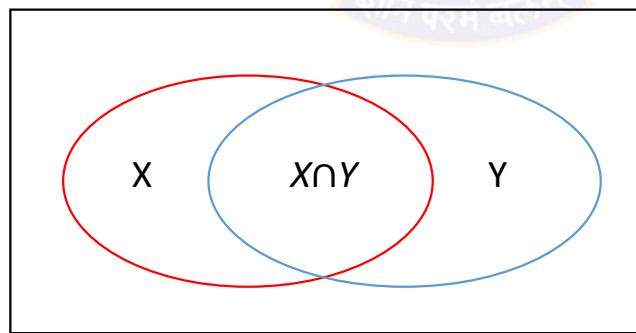
$$P(A_i \cap A_j) = P(A_i)P(A_j), \text{ for all distinct } i, j \in \{1, 2, \dots, n\};$$

$$P(A_i \cap A_j \cap A_k) = \underline{P(A_i)P(A_j)P(A_k)}, \text{ for all distinct } i, j, k \in \{1, 2, \dots, n\};$$

Conditional Probability

- Two random variables X and Y are said to be independent if their occurrence is not dependent on each other
- In that case, probability of both events occurring $P(X \cap Y) = P(X).P(Y)$
 - Ex: Outcome of 2 coin tosses
- If random variable X is dependent on Y, then the probability of X given Y occurs is termed as ‘conditional probability’ $P(X | Y)$, which can be expressed as the fraction of times X and Y occurs over the occurrence of Y, i.e.

$$P(X | Y) = P(X \cap Y) / P(Y)$$



Bayes Theorem

- Conditional probability - given random variables X and Y, probability of X given Y can be expressed as:

$$P(X | Y) = P(X \cap Y) / P(Y)$$

- The same can be written for Y given X as:

$$P(Y | X) = P(Y \cap X) / P(X)$$

- Since $P(X \cap Y) = P(Y \cap X)$, solving both equations gives:

$$P(X \cap Y) = P(X | Y) P(Y) = P(Y | X) P(X)$$

- We can rewrite conditional probability of X given Y as:

$$P(X | Y) = P(Y | X) P(X) / P(Y)$$

- **This is also known as Bayes theorem**

- In plain English, this can be written as

$$\text{posterior} = \frac{\text{likelihood} \times \text{prior}}{\text{evidence}}$$

Bayes Theorem Example 1

if patient has meningitis, then very often a stiff neck is observed

$$P(S|M) = 0.8 \text{ (can be easily determined by counting)}$$

observation: 'I have a stiff neck! Do I have meningitis?' (is it reasonable to be afraid?)

$$P(M|S) = ?$$

we need to know: $P(M) = 0.0001$ (one of 10000 people has meningitis)

and $P(S) = 0.1$ (one out of 10 people has a stiff neck).

then:

$$P(M|S) = \frac{P(S|M)P(M)}{P(S)} = \frac{0.8 \times 0.0001}{0.1} = 0.0008$$

Keep cool. Not very likely

Bayes Theorem Example 2

- Consider two boxes of fruit – red and blue, each containing apples (green color) and oranges
- Assume the probabilities of selecting red box ($B=r$) and blue box($B=b$) at random is:

$$p(B = r) = 4/10 \quad p(B = b) = 6/10$$

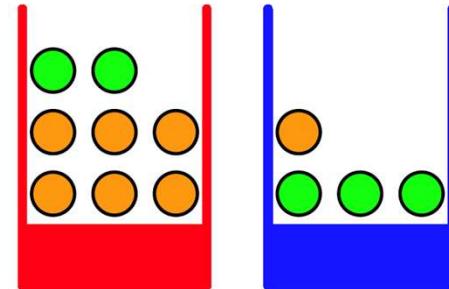
- If we select a box at random and then select a fruit at random, the conditional probabilities for fruit selection thus become:

$$p(F = a | B = r) = 2/8=1/4$$

$$p(F = o | B = r) = 6/8=3/4$$

$$p(F = a | B = b) = 3/4$$

$$p(F = o | B = b) = 1/4$$



- Lets find out probability of picking an apple in general, using sum and products rule of probability

$$\begin{aligned} p(F = a) &= p(F = a | B = r) . p(B = r) + p(F = a | B = b) . p(B = b) \\ &= 1/4 \times 4/10 + 3/4 \times 6/10 = 11/20 \end{aligned}$$

- Since $p(F = a) + p(F = o) = 1$, Probability of choosing an orange becomes:

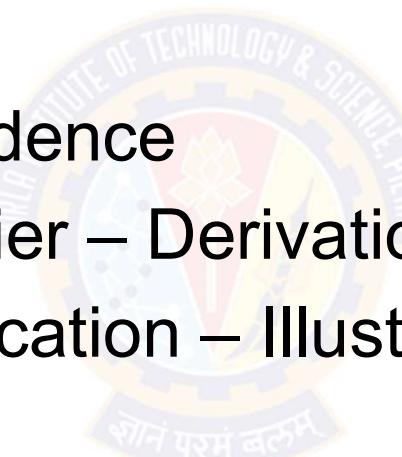
$$p(F = o) = 1 - p(F = a) = 1 - 11/20 = 9/20$$

- Now, if the chosen fruit is orange, using Bayes theorem, the probability that it came from red box can be determined as:

$$p(B = r | F = o) = p(F = o | B = r).p(B = r) / p(F = o) = 3/4 \times 4/10 / (9/20) = 2/3$$

In this segment

- Bayes' Theorem
- Conditional Independence
- Naïve Bayes Classifier – Derivation
- Naïve Bayes Classification – Illustrative Examples



Naïve Bayes Classifier

- Bayesian learning algorithms calculate explicit probabilities for each class.
- Naive Bayes classifier, are among the most practical approaches to certain types of learning problems
- For example: Problem of learning to classify text documents such as electronic news articles. For such learning tasks, the naive Bayes classifier is among the most effective algorithms.

Independence

- Two events A and B are independent if and only if $P(A \cap B) = P(A)P(B)$.
- If two events A and B are independent then $P(A|B) = P(A)$

$$\begin{aligned} P(A|B) &= \frac{P(A \cap B)}{P(B)} \\ &= \frac{P(A)P(B)}{P(B)} \\ &= P(A). \end{aligned}$$

For example, the probability that a fair coin shows "heads" after being flipped is $1/2$. What if we knew the day was Tuesday? Does this change the probability of getting "heads"? The probability of getting "heads," given that it's a Tuesday, is still $1/2$.

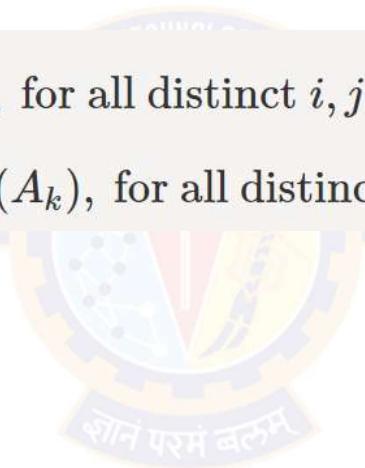
So the result of a coin flip and the day being Tuesday are independent events; knowing it was a Tuesday didn't change the probability of getting "heads."

Independence of events

In general, for n events A₁,A₂,…,A_n to be independent we must have

$$P(A_i \cap A_j) = P(A_i)P(A_j), \text{ for all distinct } i, j \in \{1, 2, \dots, n\};$$

$$P(A_i \cap A_j \cap A_k) = P(A_i)P(A_j)P(A_k), \text{ for all distinct } i, j, k \in \{1, 2, \dots, n\};$$



Conditional Probability

- Conditional Probability is a measure of the probability of an event given that another event has already occurred.
- Let the event A assumed to have occurred.
- If the event of interest is B and is dependent on occurrence of A
- This probability is written $P(B|A)$ notation for the *probability of B given A*.
- The joint probability of event A and B occurring is:

$$P(A \cap B) = P(A) \times P(B|A)$$

Conditional Probability Example

- **Event A** is drawing a King first, and **Event B** is drawing a King second.
- For the first card the chance of drawing a King is 4 out of 52 (there are 4 Kings in a deck of 52 cards):

$$P(A) = 4/52$$

- But after removing a King from the deck the probability of the 2nd card drawn is **less** likely to be a King (only 3 of the 51 cards left are Kings):

$$P(B|A) = 3/51$$

- And so:

$$\mathbf{P(A \text{ and } B)} = \mathbf{P(A) \times P(B|A)} = (4/52) \times (3/51) = 12/2652 = 1/221$$

- So the chance of getting 2 Kings is 1 in 221, or about 0.5%

Law of Total Probability

If B_1, B_2, B_3, \dots is a partition of the sample space S , then for any event A we have

$$P(A) = \sum_i P(A \cap B_i) = \sum_i P(A|B_i)P(B_i).$$

Example: A person has undertaken a mining job. The probabilities of completion of job on time with and without rain are 0.42 and 0.90 respectively. If the probability that it will rain is 0.45, then determine the probability that the mining job will be completed on time.

Solution:

Let A be the event that the mining job will be completed on time and B be the event that it rains.

$$P(B) = 0.45,$$

$$P(\text{no rain}) = P(B') = 1 - P(B) = 1 - 0.45 = 0.55$$

$$P(A|B) = 0.42, \text{ and } P(A|B') = 0.90$$

Since, events B and B' form partitions of the sample space S , by total probability theorem, we have

$$P(A) = P(B) P(A|B) + P(B') P(A|B')$$

$$= 0.45 \times 0.42 + 0.55 \times 0.9 = 0.189 + 0.495 = 0.684$$

So, the probability that the job will be completed on time is 0.684

Bayes' theorem

Using Conditional Probability

$$P(A|B)P(B) = P(A, B) = P(B|A)P(A)$$

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Bayes' Rule

- For any two events A and B , where $P(A) \neq 0$, we have

$$P(B|A) = \frac{P(A|B)P(B)}{P(A)}.$$

- If B_1, B_2, B_3, \dots form a partition of the sample space S , and A is any event with $P(A) \neq 0$, we have

$$P(B_j|A) = \frac{P(A|B_j)P(B_j)}{\sum_i P(A|B_i)P(B_i)}.$$

Bayes' theorem example

Given:

- A doctor knows that meningitis causes stiff neck 50% of the time
- Prior probability of any patient having meningitis is 1/50,000
- Prior probability of any patient having stiff neck is 1/20

If a patient has stiff neck, what's the probability he/she has meningitis?

$$P(M | S) = \frac{P(S | M)P(M)}{P(S)} = \frac{0.5 \times 1/50000}{1/20} = 0.0002$$

Conditional independence

Definition: X is conditionally independent of Y given Z , if the probability distribution governing X is independent of the value of Y , given the value of Z

$$(\forall i, j, k) P(X = x_i | Y = y_j, Z = z_k) = P(X = x_i | Z_k)$$

$$P(X|Y, Z) = P(X|Z)$$

Example:

$$P(\text{Thunder}|\text{Rain, Lightning}) = P(\text{Thunder}|\text{Lightning})$$

Bayesian Classifiers

Let the attributes X_1, X_2, \dots, X_n and class labels Y_1, Y_2, \dots, Y_m be random variables

- Given a record with attributes (X_1, X_2, \dots, X_n)
 - Goal is to predict class $Y=Y_k$
 - We want to find the value of Y that maximizes $P(Y| X_1, X_2, \dots, X_n)$
- Can we estimate $P(Y| X_1, X_2, \dots, X_n)$ directly from data?

Bayesian Classifiers

- Compute the posterior probability $P(Y| X_1, X_2, \dots, X_n)$ for all values of Y using the Bayes theorem

$$P(Y| X_1, X_2, \dots, X_n) = \frac{P(X_1, X_2, \dots, X_n | Y)P(Y)}{P(X_1, X_2, \dots, X_n)}$$

- Choose value of C that maximizes $P(Y| X_1, X_2, \dots, X_n)$
- Equivalent to choosing value of Y that maximizes $P(X_1, X_2, \dots, X_n | Y)P(Y)$

How to estimate $P(X_1, X_2, \dots, X_n | Y)$?

Chain Rule for Conditional Probability

$$P(A \cap B) = P(A)P(B|A) = P(B)P(A|B)$$

Now we can extend this formula to three or more events:

$$\begin{aligned} P(A \cap B \cap C) &= P(A \cap (B \cap C)) = P(A)P(B \cap C|A) \\ &\quad P(B \cap C) = P(B)P(C|B). \end{aligned}$$

Conditioning both sides on A , we obtain

$$P(B \cap C|A) = P(B|A)P(C|A, B)$$

Combining Equation 1.6 and 1.7 we obtain the following chain rule:

Chain rule for conditional probability:

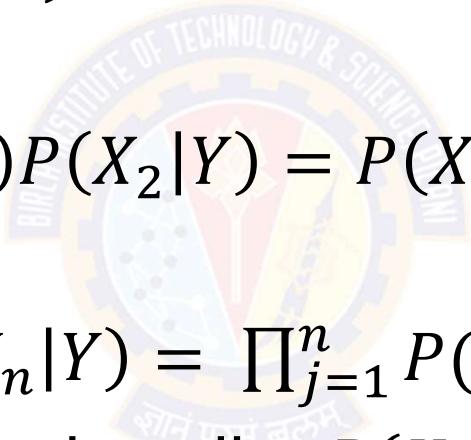
$$P(A_1 \cap A_2 \cap \cdots \cap A_n) = P(A_1)P(A_2|A_1)P(A_3|A_2, A_1) \cdots P(A_n|A_{n-1}A_{n-2} \cdots A_1)$$

https://www.probabilitycourse.com/chapter1/1_4_0_conditional_probability.php

Applying conditional independence

Naïve Bayes assumes X_i are conditionally independent given Y
e.g., $P(X_1|X_2, Y) = P(X_1|Y)$

$$P(X_1, X_2|Y) = P(X_1|X_2, Y)P(X_2|Y) = P(X_1|Y)P(X_2|Y)$$


$$\text{General form: } P(X_1, \dots, X_n|Y) = \prod_{j=1}^n P(X_j|Y)$$

How many parameters to describe $P(X_1, \dots, X_n|Y)$?

Naïve Bayes Independence assumption

Assumption:

$$P(X_1, \dots, X_n | Y) = \prod_{j=1}^n P(X_j | Y)$$

i.e., X_i and X_j are conditionally independent given Y for $i \neq j$

Naïve Bayes Classifier

- Bayes rule:

$$P(Y = y_k | X_1, \dots, X_n) = \frac{P(Y = y_k)P(X_1, \dots, X_n | Y = y_k)}{\sum_j P(Y = y_j)P(X_1, \dots, X_n | Y = y_j)}$$

- Assume conditional independence among X_i 's:

$$P(Y = y_k | X_1, \dots, X_n) = \frac{P(Y = y_k)\prod_i P(X_i | Y = y_k)}{\sum_j P(Y = y_j)\prod_i P(X_i | Y = y_j)}$$

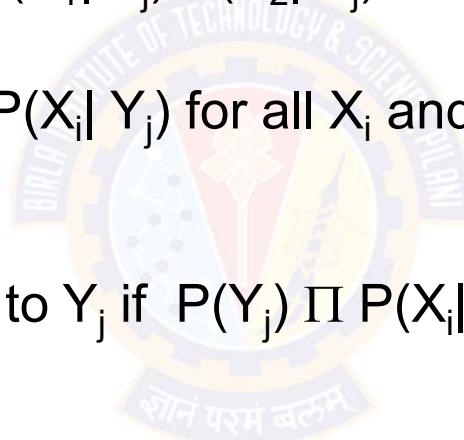
- Pick the most probable (MAP) Y

$$\hat{Y} \leftarrow \operatorname{argmax}_{y_k} P(Y = y_k)\prod_i P(X_i | Y = y_k)$$

↑
Prior Probability ↑
 MLE

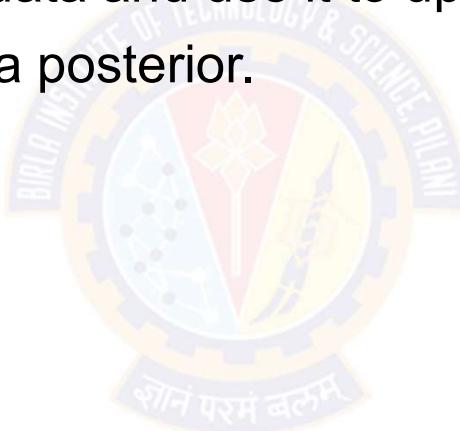
Naïve Bayes Classifier

- Assume independence among attributes X_i when class is given:
 - $P(X_1, X_2, \dots, X_d | Y_j) = P(X_1 | Y_j) P(X_2 | Y_j) \dots P(X_d | Y_j)$
 - Now we can estimate $P(X_i | Y_j)$ for all X_i and Y_j combinations from the training data
 - New point is classified to Y_j if $P(Y_j) \prod P(X_i | Y_j)$ is maximal.



Naïve Bayes Classifier

- Start with some belief about the system, called a prior.
- Then we obtain some data and use it to update our belief.
- The outcome is called a posterior.



Naïve Bayes Classifier – Example 1

Name	Give Birth	Can Fly	Live in Water	Have Legs	Class
human	yes	no	no	yes	mammals
python	no	no	no	no	non-mammals
salmon	no	no	yes	no	non-mammals
whale	yes	no	yes	no	mammals
frog	no	no	sometimes	yes	non-mammals
komodo	no	no	no	yes	non-mammals
bat	yes	yes	no	yes	mammals
pigeon	no	yes	no	yes	non-mammals
cat	yes	no	no	yes	mammals
leopard shark	yes	no	yes	no	non-mammals
turtle	no	no	sometimes	yes	non-mammals
penguin	no	no	sometimes	yes	non-mammals
porcupine	yes	no	no	yes	mammals
eel	no	no	yes	no	non-mammals
salamander	no	no	sometimes	yes	non-mammals
gila monster	no	no	no	yes	non-mammals
platypus	no	no	no	yes	mammals
owl	no	yes	no	yes	non-mammals
dolphin	yes	no	yes	no	mammals
eagle	no	yes	no	yes	non-mammals

Give Birth	Can Fly	Live in Water	Have Legs	Class
yes	no	yes	no	?

A: attributes

M: mammals

N: non-mammals

$$P(A|M) = \frac{6}{7} \times \frac{6}{7} \times \frac{2}{7} \times \frac{2}{7} = 0.06$$

$$P(A|N) = \frac{1}{13} \times \frac{10}{13} \times \frac{3}{13} \times \frac{4}{13} = 0.0042$$

$$P(A|M)P(M) = 0.06 \times \frac{7}{20} = 0.021$$

$$P(A|N)P(N) = 0.004 \times \frac{13}{20} = 0.0027$$

$$P(A|M)P(M) > P(A|N)P(N)$$

=> Mammals

Naïve Bayes Classifier – Example 2

- Use case is to predict the chances of playing tennis given a few parameters
 - Weather outlook
 - Temperature
 - Humidity
 - Wind
- Data is given for 14 days

Day	<i>Outlook</i>	<i>Temperature</i>	<i>Humidity</i>	<i>Wind</i>	<i>PlayTennis</i>
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

Naïve Bayes Classifier – Example 2

- New input set to be classified is –
 $\{Outlook = \text{sunny}, Temperature = \text{cool}, Humidity} = \text{high}, Wind = \text{strong}\}$
- Output parameter ($PlayTennis$) has 2 classes – Yes, No
- Step-1 Calculate conditional probabilities of each input parameter given each of the classes (likelihood)

$$P(Outlook=\text{sunny} | PlayTennis=yes) = 2 / 9$$

$$P(Outlook=\text{sunny} | PlayTennis=no) = 3 / 5$$

$$P(Temp=\text{cool} | PlayTennis=yes) = 3 / 9$$

$$P(Temp=\text{cool} | PlayTennis=no) = 1 / 5$$

$$P(Humidity=\text{high} | PlayTennis=yes) = 3 / 9$$

$$P(Humidity=\text{high} | PlayTennis=no) = 4 / 5$$

$$P(Wind=\text{strong} | PlayTennis=yes) = 3 / 9$$

$$P(Wind=\text{strong} | PlayTennis=no) = 3 / 5$$

- Step-2 Calculate total probability of each of the classes (prior)

$$P(PlayTennis=yes) = 9/14$$

$$P(PlayTennis=no) = 5/14$$

Naïve Bayes Classifier – Example 2

- Step-3 Calculate probability for each class given the input conditions in new input set (posterior)

$$P(\text{PlayTennis=yes} \mid \text{Outlook=sunny, Temp=cool, Humidity=high, Wind=strong})$$

$$= P(\text{yes}). P(\text{sunny} \mid \text{yes}). P(\text{cool} \mid \text{yes}). P(\text{high} \mid \text{yes}). P(\text{strong} \mid \text{yes})$$

$$= 9/14 * 2/9 * 3/9 * 3/9 * 3/9 = 0.00529$$

$$P(\text{PlayTennis=no} \mid \text{Outlook=sunny, Temp=cool, Humidity=high, Wind=strong})$$

$$= P(\text{no}). P(\text{sunny} \mid \text{no}). P(\text{cool} \mid \text{no}). P(\text{high} \mid \text{no}). P(\text{strong} \mid \text{no})$$

$$= 5/14 * 3/5 * 1/5 * 4/5 * 3/5 = 0.02057$$

Using Naïve Bayes classifier, $\text{argmax}(P) = \text{no}$, which mean the classifier assign $\text{PlayTennis} = \text{No}$

Discrete and Continuous value attributes

To compute, $P(x_k|C_i)$

- A_k is categorical:

the number of tuples of class C_i in D having the value x_k for A_k

$$P(x_k|C_i) = \frac{\text{the number of tuples of class } C_i \text{ in } D.}{\text{the number of tuples of class } C_i \text{ in } D.}$$

- A_k is continuous:

A continuous-valued attribute is typically assumed to have a Gaussian distribution with a mean μ and standard deviation σ

$$g(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}},$$

$$P(x_k|C_i) = g(x_k, \mu_{C_i}, \sigma_{C_i}).$$

Estimate Probabilities from Continuous Data

Tid	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

- Normal distribution:
- One for each (A_i, c_i) pair



$$P(A_i | c_j) = \frac{1}{\sqrt{2\pi\sigma_{ij}^2}} e^{-\frac{(A_i - \mu_{ij})^2}{2\sigma_{ij}^2}}$$

- For (Income, Class=No):
 - If Class=No
 - sample mean = 110
 - sample variance = 3179.16

$$P(\text{Income} = 120 | \text{No}) = \frac{1}{\sqrt{2\pi(56.38)}} e^{-\frac{(120-110)^2}{2(3179)}} = 0.0069$$

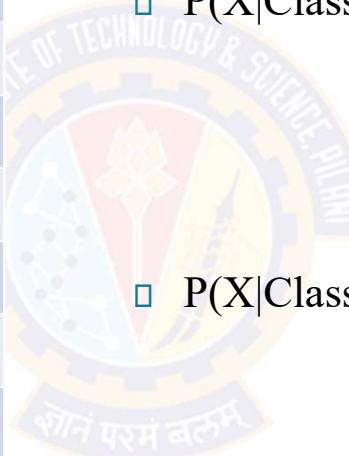
Example of Naïve Bayes Classifier

Tid	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Given a Test Record:

$$X = (\text{Refund} = \text{No}, \text{Married}, \text{Income} = 120\text{K})$$

- $P(X|\text{Class}=\text{No}) = P(\text{Refund}=\text{No}|\text{Class}=\text{No}) \times P(\text{Married}|\text{Class}=\text{No}) \times P(\text{Income}=120\text{K}|\text{Class}=\text{No}) = 4/7 \times 4/7 \times 0.0069 = 0.0023$



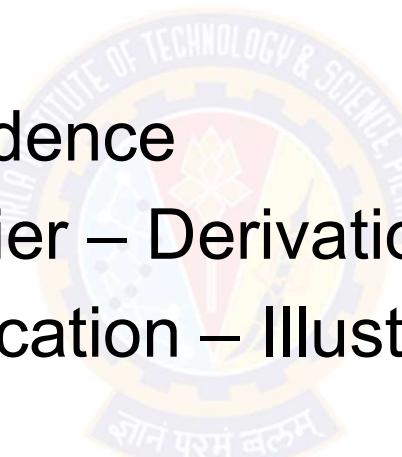
- $P(X|\text{Class}=\text{Yes}) = P(\text{Refund}=\text{No}|\text{Class}=\text{Yes}) \times P(\text{Married}|\text{Class}=\text{Yes}) \times P(\text{Income}=120\text{K}|\text{Class}=\text{Yes}) = 1 \times 0 \times 1.2 \times 10^{-9} = 0$

Since $P(X|\text{No})P(\text{No}) > P(X|\text{Yes})P(\text{Yes})$

Therefore $P(\text{No}|X) > P(\text{Yes}|X)$
 $\Rightarrow \text{Class} = \text{No}$

In this segment

- Bayes' Theorem
- Conditional Independence
- Naïve Bayes Classifier – Derivation
- Naïve Bayes Classification – Illustrative Examples



Naïve Bayes Classifier

- Bayesian learning algorithms calculate explicit probabilities for each class.
- Naive Bayes classifier, are among the most practical approaches to certain types of learning problems
- For example: Problem of learning to classify text documents such as electronic news articles. For such learning tasks, the naive Bayes classifier is among the most effective algorithms.

Independence

- Two events A and B are independent if and only if $P(A \cap B) = P(A)P(B)$.
- If two events A and B are independent then $P(A|B) = P(A)$

$$\begin{aligned} P(A|B) &= \frac{P(A \cap B)}{P(B)} \\ &= \frac{P(A)P(B)}{P(B)} \\ &= P(A). \end{aligned}$$

For example, the probability that a fair coin shows "heads" after being flipped is $1/2$. What if we knew the day was Tuesday? Does this change the probability of getting "heads"? The probability of getting "heads," given that it's a Tuesday, is still $1/2$.

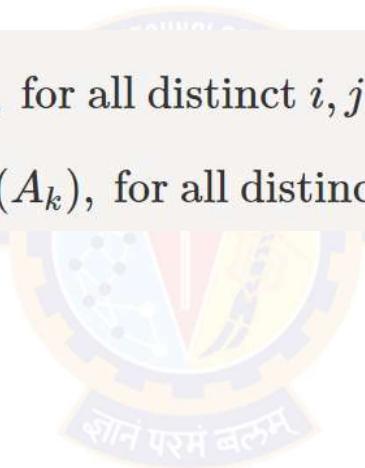
So the result of a coin flip and the day being Tuesday are independent events; knowing it was a Tuesday didn't change the probability of getting "heads."

Independence of events

In general, for n events A₁,A₂,…,A_n to be independent we must have

$$P(A_i \cap A_j) = P(A_i)P(A_j), \text{ for all distinct } i, j \in \{1, 2, \dots, n\};$$

$$P(A_i \cap A_j \cap A_k) = P(A_i)P(A_j)P(A_k), \text{ for all distinct } i, j, k \in \{1, 2, \dots, n\};$$



Conditional Probability

- Conditional Probability is a measure of the probability of an event given that another event has already occurred.
- Let the event A assumed to have occurred.
- If the event of interest is B and is dependent on occurrence of A
- This probability is written $P(B|A)$ notation for the *probability of B given A*.
- The joint probability of event A and B occurring is:

$$P(A \cap B) = P(A) \times P(B|A)$$

Conditional Probability Example

- **Event A** is drawing a King first, and **Event B** is drawing a King second.
- For the first card the chance of drawing a King is 4 out of 52 (there are 4 Kings in a deck of 52 cards):

$$P(A) = 4/52$$

- But after removing a King from the deck the probability of the 2nd card drawn is **less** likely to be a King (only 3 of the 51 cards left are Kings):

$$P(B|A) = 3/51$$

- And so:

$$\mathbf{P(A \text{ and } B) = P(A) \times P(B|A) = (4/52) \times (3/51) = 12/2652 = 1/221}$$

- So the chance of getting 2 Kings is 1 in 221, or about 0.5%

Law of Total Probability

If B_1, B_2, B_3, \dots is a partition of the sample space S , then for any event A we have

$$P(A) = \sum_i P(A \cap B_i) = \sum_i P(A|B_i)P(B_i).$$

Example: A person has undertaken a mining job. The probabilities of completion of job on time with and without rain are 0.42 and 0.90 respectively. If the probability that it will rain is 0.45, then determine the probability that the mining job will be completed on time.

Solution:

Let A be the event that the mining job will be completed on time and B be the event that it rains.

$$P(B) = 0.45,$$

$$P(\text{no rain}) = P(B') = 1 - P(B) = 1 - 0.45 = 0.55$$

$$P(A|B) = 0.42, \text{ and } P(A|B') = 0.90$$

Since, events B and B' form partitions of the sample space S , by total probability theorem, we have

$$\begin{aligned} P(A) &= P(B) P(A|B) + P(B') P(A|B') \\ &= 0.45 \times 0.42 + 0.55 \times 0.9 = 0.189 + 0.495 = 0.684 \end{aligned}$$

So, the probability that the job will be completed on time is 0.684

Bayes' theorem

Using Conditional Probability

$$P(A|B)P(B) = P(A, B) = P(B|A)P(A)$$

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Bayes' Rule

- For any two events A and B , where $P(A) \neq 0$, we have

$$P(B|A) = \frac{P(A|B)P(B)}{P(A)}.$$

- If B_1, B_2, B_3, \dots form a partition of the sample space S , and A is any event with $P(A) \neq 0$, we have

$$P(B_j|A) = \frac{P(A|B_j)P(B_j)}{\sum_i P(A|B_i)P(B_i)}.$$

Bayes' theorem example

Given:

- A doctor knows that meningitis causes stiff neck 50% of the time
- Prior probability of any patient having meningitis is 1/50,000
- Prior probability of any patient having stiff neck is 1/20

If a patient has stiff neck, what's the probability he/she has meningitis?

$$P(M | S) = \frac{P(S | M)P(M)}{P(S)} = \frac{0.5 \times 1/50000}{1/20} = 0.0002$$

Conditional independence

Definition: X is conditionally independent of Y given Z , if the probability distribution governing X is independent of the value of Y , given the value of Z

$$(\forall i, j, k) P(X = x_i | Y = y_j, Z = z_k) = P(X = x_i | Z_k)$$

$$P(X|Y, Z) = P(X|Z)$$

Example:

$$P(\text{Thunder}|\text{Rain, Lightning}) = P(\text{Thunder}|\text{Lightning})$$

Bayesian Classifiers

Let the attributes X_1, X_2, \dots, X_n and class labels Y_1, Y_2, \dots, Y_m be random variables

- Given a record with attributes (X_1, X_2, \dots, X_n)
 - Goal is to predict class $Y=Y_k$
 - We want to find the value of Y that maximizes $P(Y| X_1, X_2, \dots, X_n)$
- Can we estimate $P(Y| X_1, X_2, \dots, X_n)$ directly from data?

Bayesian Classifiers

- Compute the posterior probability $P(Y| X_1, X_2, \dots, X_n)$ for all values of Y using the Bayes theorem

$$P(Y| X_1, X_2, \dots, X_n) = \frac{P(X_1, X_2, \dots, X_n | Y)P(Y)}{P(X_1, X_2, \dots, X_n)}$$

- Choose value of C that maximizes $P(Y| X_1, X_2, \dots, X_n)$
- Equivalent to choosing value of Y that maximizes $P(X_1, X_2, \dots, X_n | Y)P(Y)$

How to estimate $P(X_1, X_2, \dots, X_n | Y)$?

Chain Rule for Conditional Probability

$$P(A \cap B) = P(A)P(B|A) = P(B)P(A|B)$$

Now we can extend this formula to three or more events:

$$\begin{aligned} P(A \cap B \cap C) &= P(A \cap (B \cap C)) = P(A)P(B \cap C|A) \\ &\quad P(B \cap C) = P(B)P(C|B). \end{aligned}$$

Conditioning both sides on A , we obtain

$$P(B \cap C|A) = P(B|A)P(C|A, B)$$

Combining Equation 1.6 and 1.7 we obtain the following chain rule:

Chain rule for conditional probability:

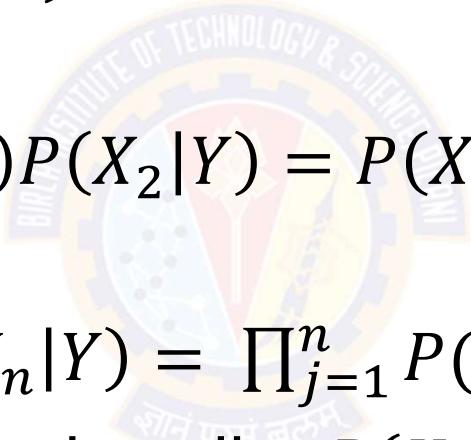
$$P(A_1 \cap A_2 \cap \cdots \cap A_n) = P(A_1)P(A_2|A_1)P(A_3|A_2, A_1) \cdots P(A_n|A_{n-1}A_{n-2} \cdots A_1)$$

https://www.probabilitycourse.com/chapter1/1_4_0_conditional_probability.php

Applying conditional independence

Naïve Bayes assumes X_i are conditionally independent given Y
e.g., $P(X_1|X_2, Y) = P(X_1|Y)$

$$P(X_1, X_2|Y) = P(X_1|X_2, Y)P(X_2|Y) = P(X_1|Y)P(X_2|Y)$$


$$\text{General form: } P(X_1, \dots, X_n|Y) = \prod_{j=1}^n P(X_j|Y)$$

How many parameters to describe $P(X_1, \dots, X_n|Y)$?

Naïve Bayes Independence assumption

Assumption:

$$P(X_1, \dots, X_n | Y) = \prod_{j=1}^n P(X_j | Y)$$

i.e., X_i and X_j are conditionally independent given Y for $i \neq j$

Naïve Bayes Classifier

- Bayes rule:

$$P(Y = y_k | X_1, \dots, X_n) = \frac{P(Y = y_k) P(X_1, \dots, X_n | Y = y_k)}{\sum_j P(Y = y_j) P(X_1, \dots, X_n | Y = y_j)}$$

- Assume conditional independence among X_j 's:

$$P(Y = y_k | X_1, \dots, X_n) = \frac{P(Y = y_k) \Pi_i P(X_i | Y = y_k)}{\sum_j P(Y = y_j) \Pi_i P(X_i | Y = y_j)}$$

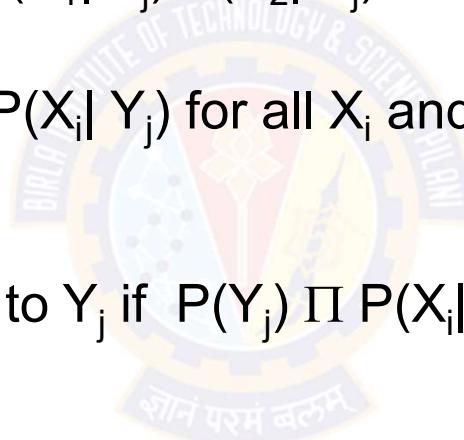
- Pick the most probable (MAP) Y

$$\hat{Y} \leftarrow \operatorname*{argmax}_{y_k} P(Y = y_k) \Pi_i P(X_i | Y = y_k)$$


 Prior Probability MLE

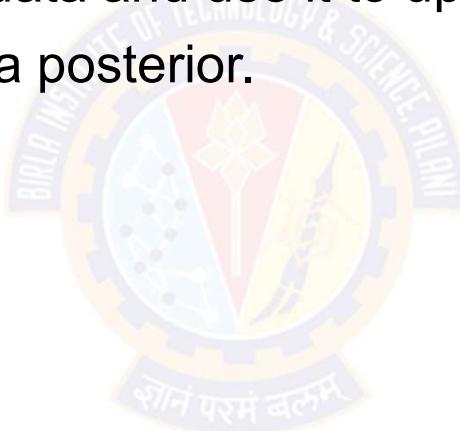
Naïve Bayes Classifier

- Assume independence among attributes X_i when class is given:
 - $P(X_1, X_2, \dots, X_d | Y_j) = P(X_1 | Y_j) P(X_2 | Y_j) \dots P(X_d | Y_j)$
 - Now we can estimate $P(X_i | Y_j)$ for all X_i and Y_j combinations from the training data
 - New point is classified to Y_j if $P(Y_j) \prod P(X_i | Y_j)$ is maximal.



Naïve Bayes Classifier

- Start with some belief about the system, called a prior.
- Then we obtain some data and use it to update our belief.
- The outcome is called a posterior.



Naïve Bayes Classifier – Example 1

Name	Give Birth	Can Fly	Live in Water	Have Legs	Class
human	yes	no	no	yes	mammals
python	no	no	no	no	non-mammals
salmon	no	no	yes	no	non-mammals
whale	yes	no	yes	no	mammals
frog	no	no	sometimes	yes	non-mammals
komodo	no	no	no	yes	non-mammals
bat	yes	yes	no	yes	mammals
pigeon	no	yes	no	yes	non-mammals
cat	yes	no	no	yes	mammals
leopard shark	yes	no	yes	no	non-mammals
turtle	no	no	sometimes	yes	non-mammals
penguin	no	no	sometimes	yes	non-mammals
porcupine	yes	no	no	yes	mammals
eel	no	no	yes	no	non-mammals
salamander	no	no	sometimes	yes	non-mammals
gila monster	no	no	no	yes	non-mammals
platypus	no	no	no	yes	mammals
owl	no	yes	no	yes	non-mammals
dolphin	yes	no	yes	no	mammals
eagle	no	yes	no	yes	non-mammals

Give Birth	Can Fly	Live in Water	Have Legs	Class
yes	no	yes	no	?

A: attributes

M: mammals

N: non-mammals

$$P(A|M) = \frac{6}{7} \times \frac{6}{7} \times \frac{2}{7} \times \frac{2}{7} = 0.06$$

$$P(A|N) = \frac{1}{13} \times \frac{10}{13} \times \frac{3}{13} \times \frac{4}{13} = 0.0042$$

$$P(A|M)P(M) = 0.06 \times \frac{7}{20} = 0.021$$

$$P(A|N)P(N) = 0.004 \times \frac{13}{20} = 0.0027$$

$$P(A|M)P(M) > P(A|N)P(N)$$

=> Mammals

Naïve Bayes Classifier – Example 2

- Use case is to predict the chances of playing tennis given a few parameters
 - Weather outlook
 - Temperature
 - Humidity
 - Wind
- Data is given for 14 days

Day	<i>Outlook</i>	<i>Temperature</i>	<i>Humidity</i>	<i>Wind</i>	<i>PlayTennis</i>
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

Naïve Bayes Classifier – Example 2

- New input set to be classified is –
 $\{Outlook = \text{sunny}, Temperature = \text{cool}, Humidity} = \text{high}, Wind = \text{strong}\}$
- Output parameter ($PlayTennis$) has 2 classes – Yes, No
- Step-1 Calculate conditional probabilities of each input parameter given each of the classes (likelihood)

$$P(Outlook=\text{sunny} | PlayTennis=yes) = 2 / 9$$

$$P(Outlook=\text{sunny} | PlayTennis=no) = 3 / 5$$

$$P(Temp=\text{cool} | PlayTennis=yes) = 3 / 9$$

$$P(Temp=\text{cool} | PlayTennis=no) = 1 / 5$$

$$P(Humidity=\text{high} | PlayTennis=yes) = 3 / 9$$

$$P(Humidity=\text{high} | PlayTennis=no) = 4 / 5$$

$$P(Wind=\text{strong} | PlayTennis=yes) = 3 / 9$$

$$P(Wind=\text{strong} | PlayTennis=no) = 3 / 5$$

- Step-2 Calculate total probability of each of the classes (prior)

$$P(PlayTennis=yes) = 9/14$$

$$P(PlayTennis=no) = 5/14$$

Naïve Bayes Classifier – Example 2

- Step-3 Calculate probability for each class given the input conditions in new input set (posterior)

$$P(\text{PlayTennis=yes} \mid \text{Outlook=sunny, Temp=cool, Humidity=high, Wind=strong})$$

$$= P(\text{yes}). P(\text{sunny} \mid \text{yes}). P(\text{cool} \mid \text{yes}). P(\text{high} \mid \text{yes}). P(\text{strong} \mid \text{yes})$$

$$= 9/14 * 2/9 * 3/9 * 3/9 * 3/9 = 0.00529$$

$$P(\text{PlayTennis=no} \mid \text{Outlook=sunny, Temp=cool, Humidity=high, Wind=strong})$$

$$= P(\text{no}). P(\text{sunny} \mid \text{no}). P(\text{cool} \mid \text{no}). P(\text{high} \mid \text{no}). P(\text{strong} \mid \text{no})$$

$$= 5/14 * 3/5 * 1/5 * 4/5 * 3/5 = 0.02057$$

Using Naïve Bayes classifier, $\text{argmax}(P) = \text{no}$, which mean the classifier assign $\text{PlayTennis} = \text{No}$

Discrete and Continuous value attributes

To compute, $P(x_k|C_i)$

- A_k is categorical:

the number of tuples of class C_i in D having the value x_k for A_k

$$P(x_k|C_i) = \frac{\text{the number of tuples of class } C_i \text{ in } D.}{\text{the number of tuples of class } C_i \text{ in } D.}$$

- A_k is continuous:

A continuous-valued attribute is typically assumed to have a Gaussian distribution with a mean μ and standard deviation σ

$$g(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}},$$

$$P(x_k|C_i) = g(x_k, \mu_{C_i}, \sigma_{C_i}).$$

Estimate Probabilities from Continuous Data

Tid	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

- Normal distribution:
- One for each (A_i, c_i) pair



$$P(A_i | c_j) = \frac{1}{\sqrt{2\pi\sigma_{ij}^2}} e^{-\frac{(A_i - \mu_{ij})^2}{2\sigma_{ij}^2}}$$

- For (Income, Class=No):
 - If Class=No
 - sample mean = 110
 - sample variance = 3179.16

$$P(\text{Income} = 120 | \text{No}) = \frac{1}{\sqrt{2\pi(56.38)}} e^{-\frac{(120-110)^2}{2(3179)}} = 0.0069$$

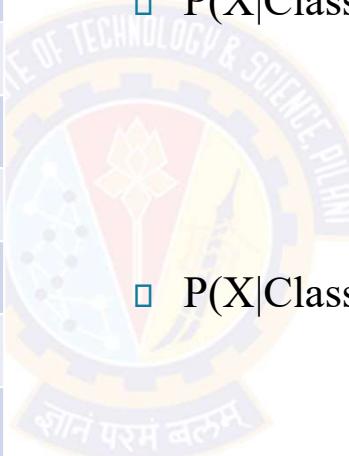
Example of Naïve Bayes Classifier

Tid	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Given a Test Record:

$$X = (\text{Refund} = \text{No}, \text{Married}, \text{Income} = 120\text{K})$$

- $P(X|\text{Class}=\text{No}) = P(\text{Refund}=\text{No}|\text{Class}=\text{No}) \times P(\text{Married}|\text{Class}=\text{No}) \times P(\text{Income}=120\text{K}|\text{Class}=\text{No}) = 4/7 \times 4/7 \times 0.0069 = 0.0023$



- $P(X|\text{Class}=\text{Yes}) = P(\text{Refund}=\text{No}|\text{Class}=\text{Yes}) \times P(\text{Married}|\text{Class}=\text{Yes}) \times P(\text{Income}=120\text{K}|\text{Class}=\text{Yes}) = 1 \times 0 \times 1.2 \times 10^{-9} = 0$

Since $P(X|\text{No})P(\text{No}) > P(X|\text{Yes})P(\text{Yes})$

Therefore $P(\text{No}|X) > P(\text{Yes}|X)$
=> Class = No

In this segment

Naïve Bayes – Advantages, Interpretability

- Issues with Naïve Bayes Classifier
- Laplace Smoothing
- Naïve Bayes Classifier : Generative model
- Advantages of Naïve Bayes Classifier



Issues with Naïve Bayes Classifier

Consider the table with Tid = 6 deleted

Tid	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Given X = (Refund = Yes, Divorced, 120K)

$$P(X | \text{No}) = 2/6 \times 0 \times 0.0083 = 0$$

$$P(X | \text{Yes}) = 0 \times 1/3 \times 1.2 \times 10^{-9} = 0$$

Naïve Bayes Classifier:

$$P(\text{Refund} = \text{Yes} | \text{No}) = 2/6$$

$$P(\text{Refund} = \text{No} | \text{No}) = 4/6$$

$$P(\text{Refund} = \text{Yes} | \text{Yes}) = 0$$

$$P(\text{Refund} = \text{No} | \text{Yes}) = 1$$

$$P(\text{Marital Status} = \text{Single} | \text{No}) = 2/6$$

$$P(\text{Marital Status} = \text{Divorced} | \text{No}) = 0$$

$$P(\text{Marital Status} = \text{Married} | \text{No}) = 4/6$$

$$P(\text{Marital Status} = \text{Single} | \text{Yes}) = 2/3$$

$$P(\text{Marital Status} = \text{Divorced} | \text{Yes}) = 1/3$$

$$P(\text{Marital Status} = \text{Married} | \text{Yes}) = 0/3$$

For Taxable Income:

If class = No: sample mean = 91
sample variance = 685

If class = Yes: sample mean = 90
sample variance = 25

**Naïve Bayes will not be able to
classify X as Yes or No!**

Issues with Naïve Bayes Classifier

- If one of the conditional probabilities is zero, then the entire expression becomes zero
- Need to use other estimates of conditional probabilities than simple fractions
- Probability estimation:

$$\text{Original: } P(A_i | C) = \frac{N_{ic}}{N_c}$$

$$\text{Laplace: } P(A_i | C) = \frac{N_{ic} + 1}{N_c + T}$$

$$\text{m - estimate: } P(A_i | C) = \frac{N_{ic} + mp}{N_c + m}$$

T: number of unique words across all documents

p: prior probability of the class

m: parameter

N_c : number of instances in the class

N_{ic} : number of instances having attribute value A_i in class c

Case Study – Text Classification

- Naïve Bayes is commonly used for **text classification**
- For a document with k terms $d = (t_1, \dots, t_k)$

Fraction of documents in c

$$P(c|d) = P(c)P(d|c) = P(c) \prod_{t_i \in d} P(t_i|c)$$

- $P(t_i|c)$ = Fraction of terms from **all documents** in c that are t_i .
- Number of times t_i appears in some document in c
- $$P(t_i|c) = \frac{N_{ic} + 1}{N_c + T}$$
- Laplace Smoothing
- Total number of terms in all documents in c
- Number of unique words (vocabulary size)
- Easy to implement and works relatively well

Case Study – Text Classification

Text	Tag
“A great game”	Sports
“The election was over”	Not sports
“Very clean match”	Sports
“A clean but forgettable game”	Sports
“It was a close election”	Not sports

Which tag does the sentence *A very close game* belong to?
 $P(\text{sports} \mid \text{A very close game})$

Bag of words i.e. word frequencies without considering order

Using Bayes Theorem:

$$P(\text{sports} \mid \text{A very close game})$$

$$= P(\text{A very close game} \mid \text{sports}) P(\text{sports})$$

$$\frac{P(\text{A very close game})}{P(\text{A very close game})}$$

We assume that every word in a sentence is **independent** of the other ones

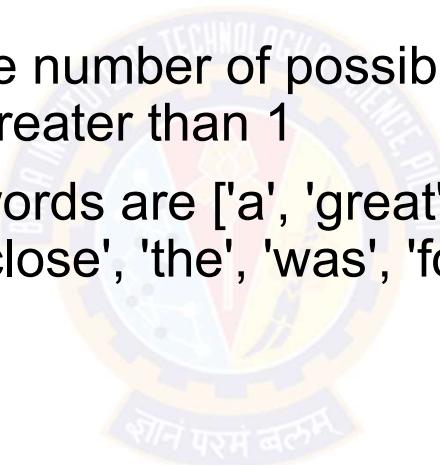
$$P(\text{a very close game}) = P(a) \times P(\text{very}) \times P(\text{close}) \times P(\text{game})$$

$$P(\text{a very close game} \mid \text{Sports}) = P(a \mid \text{Sports}) \times P(\text{very} \mid \text{Sports}) \times P(\text{close} \mid \text{Sports}) \times P(\text{game} \mid \text{Sports})$$

“close” doesn’t appear in sentences of sports tag, So $P(\text{close} \mid \text{sports}) = 0$, which makes product 0

Laplace smoothing

- Laplace smoothing: we add 1 or in general constant k to every count so it's never zero.
- To balance this, we add the number of possible words to the divisor, so the division will never be greater than 1
- In our case, the possible words are ['a', 'great', 'very', 'over', 'it', 'but', 'game', 'election', 'clean', 'close', 'the', 'was', 'forgettable', 'match'].



Apply Laplace Smoothing

Word	P(word Sports)	P(word Not Sports)
a	2+1 / 11+14	1+1 / 9+14
very	1+1 / 11+14	0+1 / 9+14
close	0+1 / 11+14	1+1 / 9+14
game	2+1 / 11+14	0+1 / 9+14

$$\begin{aligned} & P(a|Sports) \times P(very|Sports) \times P(close|Sports) \times P(game|Sports) \times \\ & P(Sports) \\ & = 2.76 \times 10^{-5} \\ & = 0.0000276 \end{aligned}$$

$$\begin{aligned} & P(a|Not\ Sports) \times P(very|Not\ Sports) \times P(close|Not\ Sports) \times \\ & P(game|Not\ Sports) \times P(Not\ Sports) \\ & = 0.572 \times 10^{-5} \\ & = 0.00000572 \end{aligned}$$

Naïve Bayes Classifier: Generative Model

Naïve Bayes Classifier is a Generative Model

- Naïve Bayes classifier uses likelihood and prior probability to calculate conditional probability of the class
- Likelihood is based on joint probability, which is the core principle of probabilistic generative model
- Naïve Bayes simplifies the calculation of likelihood by the assumption of conditional independence among input parameters
- Each parameter's likelihood is determined using joint probability of the input parameter and the output label

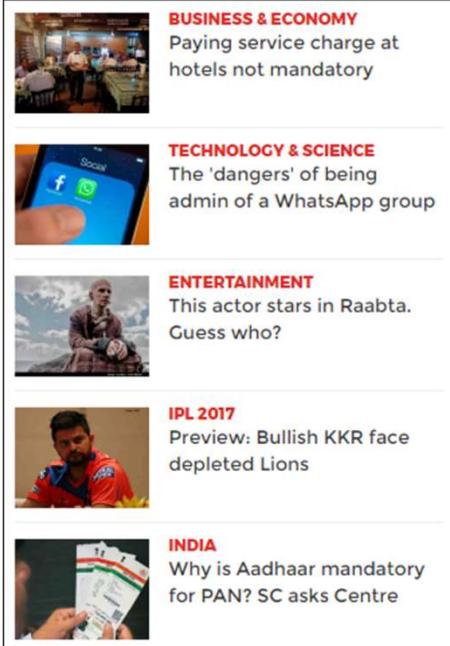
Naïve Bayes – Advantages

Advantages of Naïve Bayes Classifier

- Algorithm is simple to implement and fast
- If conditional independence holds, it will converge quickly than other methods
- Even in cases where conditional independence doesn't hold, its results are quite acceptable
- Needs less training data (due to conditional independence assumption)
- Highly scalable, scales linearly with the number of predictors and data points
- Can be used for both binary and multi-class classification problems
- Handles continuous and discrete data
- Not sensitive to irrelevant features
- Doesn't overfit the data due to small model size
- Handles missing values well (we can drop the entire feature and still not impact the result if the missing values are significant in population)

Naïve Bayes Classifier Applications

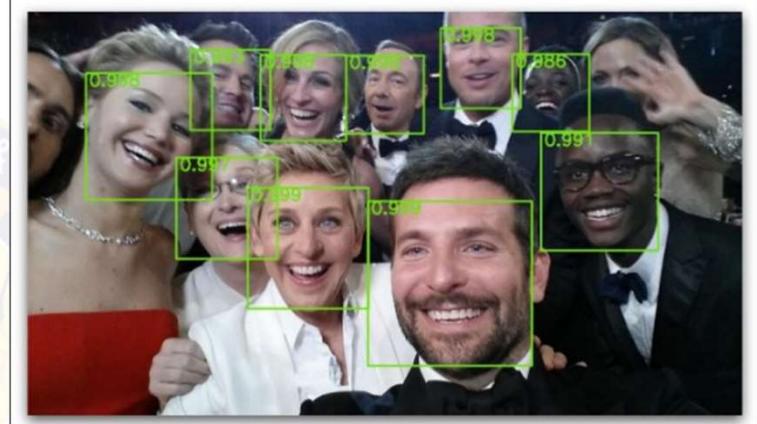
Categorizing News



Email Spam Detection



Face Recognition



Sentiment Analysis



In this segment

Fundamentals – Sigmoid Function

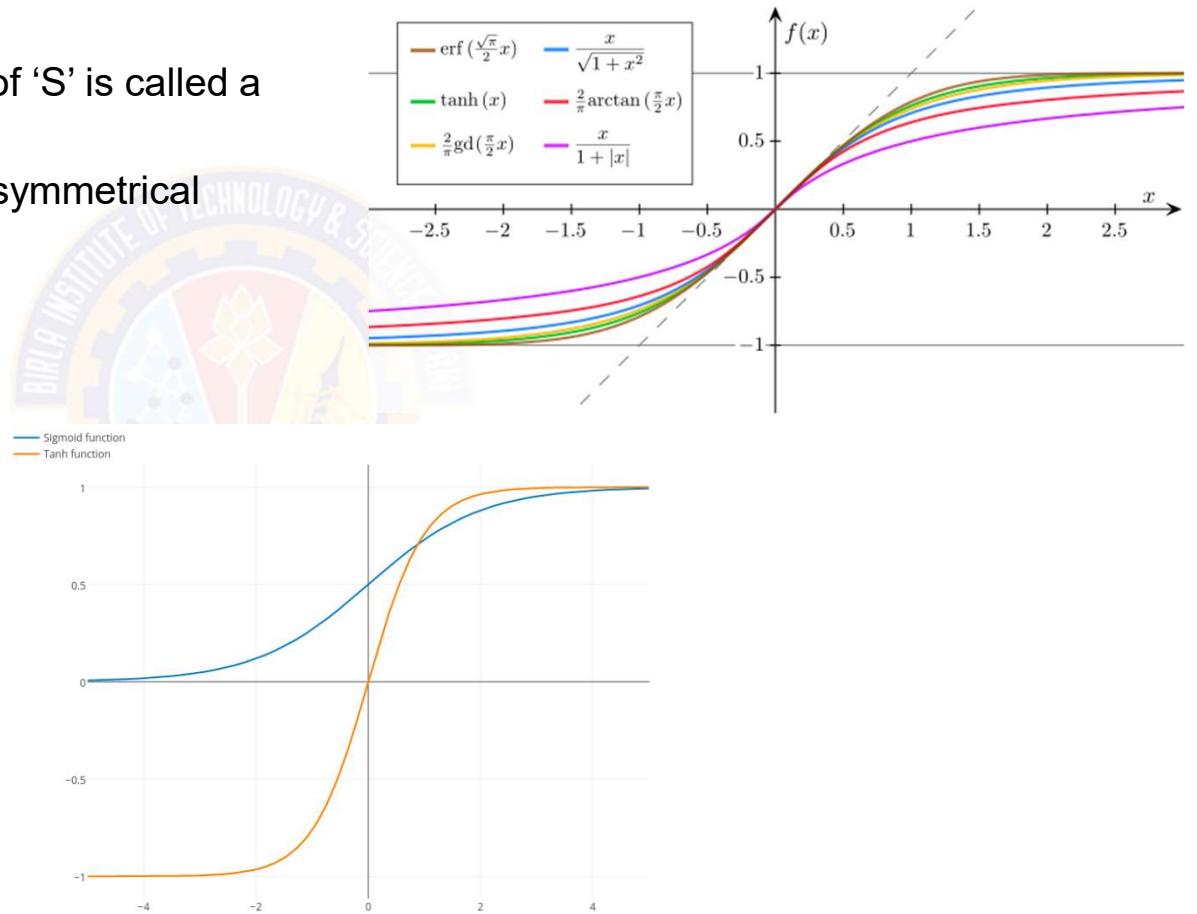
- Sigmoid Function – Overview
- Sigmoid Function – Significance
- Sigmoid Function – Derivative



Fundamentals – Sigmoid Function

Sigmoid Function – Overview

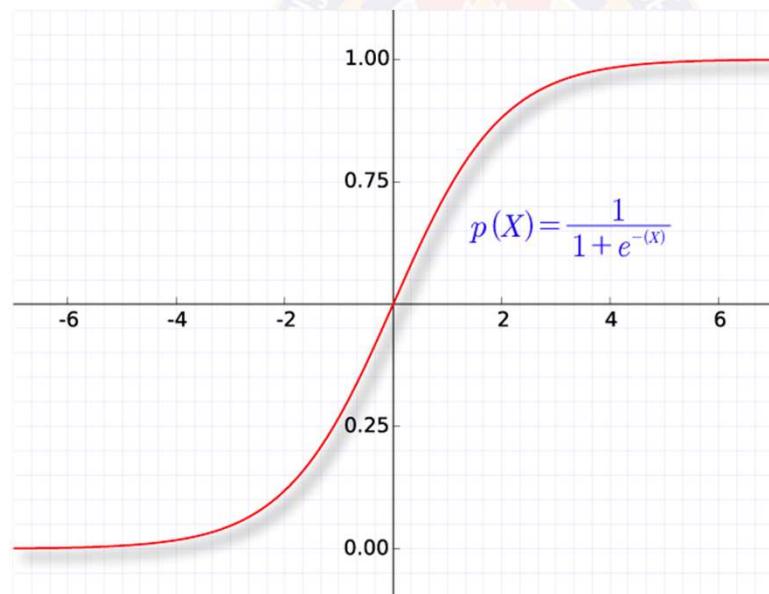
- Any function that takes the shape of 'S' is called a sigmoid function
- Maps input horizon to a bounded, symmetrical output range
- Examples:
 - Logistic Function
 - Hyperbolic Tangent
 - Arctangent function
 - Gudermannian function
 - Error function
 - Generalized Logistic function
 - ...



Fundamentals – Sigmoid Function

Sigmoid Function – Significance

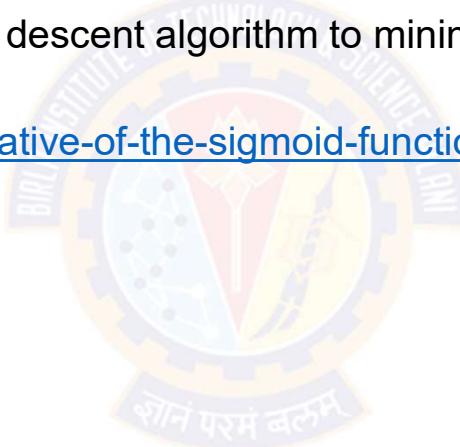
- Known as ‘squashing function’
- Logistic function (a variant of sigmoid) bounds the output between 0 and 1
- Conditional probability of a random variable can be expressed as a sigmoid (logistic) function



Derivative of Sigmoid Function

- Maximum likelihood to determine the parameters of the logistic regression model.
- To do this, we shall make use of the derivative of the logistic sigmoid function
- Use any algorithm like the gradient descent algorithm to minimize cost function by using derivative

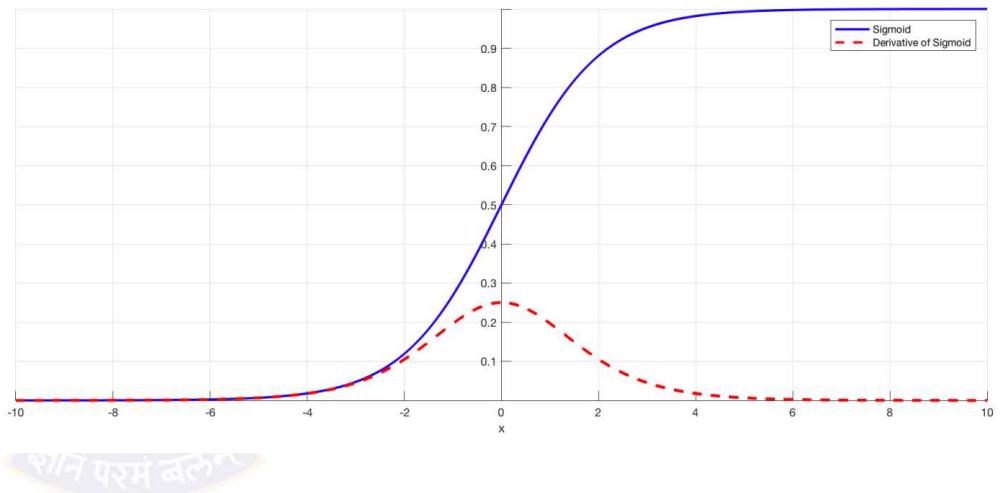
<https://towardsdatascience.com/derivative-of-the-sigmoid-function-536880cf918e>



Fundamentals – Sigmoid Function

Derivative of Sigmoid Function

$$\begin{aligned}\sigma(x) &= \frac{1}{1 + e^{-x}}. \\ \frac{d}{dx} \sigma(x) &= \frac{d}{dx} \left[\frac{1}{1 + e^{-x}} \right] \\ &= \frac{d}{dx} (1 + e^{-x})^{-1} \\ &= -(1 + e^{-x})^{-2} (-e^{-x}) \\ &= \frac{e^{-x}}{(1 + e^{-x})^2} \\ &= \frac{1}{1 + e^{-x}} \cdot \frac{e^{-x}}{1 + e^{-x}} \\ &= \frac{1}{1 + e^{-x}} \cdot \frac{(1 + e^{-x}) - 1}{1 + e^{-x}} \\ &= \frac{1}{1 + e^{-x}} \cdot \left(\frac{1 + e^{-x}}{1 + e^{-x}} - \frac{1}{1 + e^{-x}} \right) \\ &= \frac{1}{1 + e^{-x}} \cdot \left(1 - \frac{1}{1 + e^{-x}} \right) \\ &= \sigma(x) \cdot (1 - \sigma(x))\end{aligned}$$



Derivative of sigmoid function is significant in the sense that it represents a bell shaped probability distribution function of the data

Fundamentals – Sigmoid Function

Logistic Function for Classification - Overview

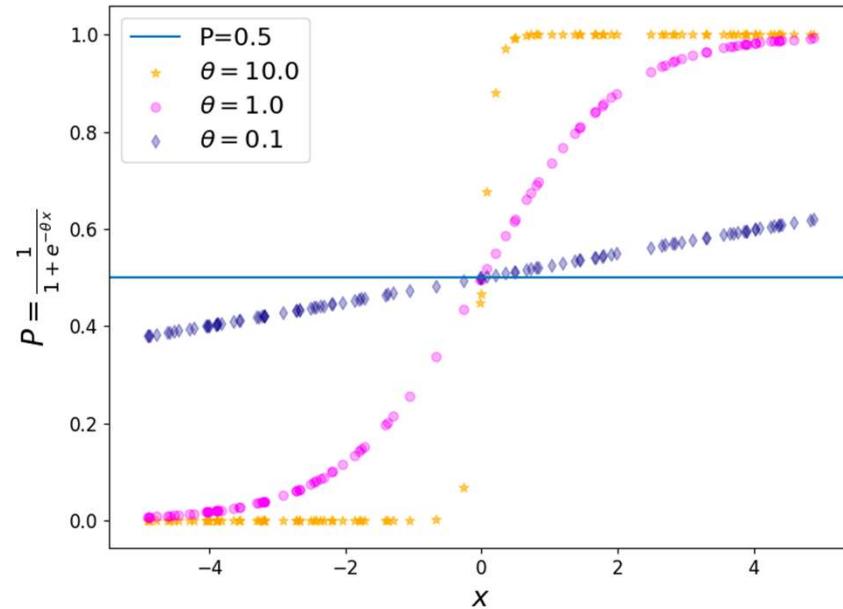
$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

$$P(Y = 1 | \mathbf{X} = \mathbf{x}) = \sigma(z) \text{ where } z = \theta_0 + \sum_{i=1}^m \theta_i x_i$$

$$\theta^T \mathbf{x} = \sum_{i=1}^m \theta_i x_i = \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_m x_m$$

$$P(Y = 1 | \mathbf{X} = \mathbf{x}) = \sigma(\theta^T \mathbf{x})$$

$$P(Y = 0 | \mathbf{X} = \mathbf{x}) = 1 - \sigma(\theta^T \mathbf{x})$$



In this segment

Foundations - Maximum Likelihood Estimation

- Maximum Likelihood Estimation – Overview
- Maximum Likelihood Estimation – Example



MLE and MAP

- Both Maximum Likelihood Estimation (MLE) and Maximum A Posterior (MAP) are used to estimate parameters for a distribution.
- MLE is a special case of Max-A-Posteriori discussed in Naïve Bayes

Recap:

- Bayes rule:

$$P(Y = y_k | X_1, \dots, X_n) = \frac{P(Y = y_k)P(X_1, \dots, X_n | Y = y_k)}{\sum_j P(Y = y_j)P(X_1, \dots, X_n | Y = y_j)}$$

- Assume conditional independence among X_i 's:

$$P(Y = y_k | X_1, \dots, X_n) = \frac{P(Y = y_k)\prod_i P(X_i | Y = y_k)}{\sum_j P(Y = y_j)\prod_i P(X_i | Y = y_j)}$$

- Pick the most probable (MAP) Y

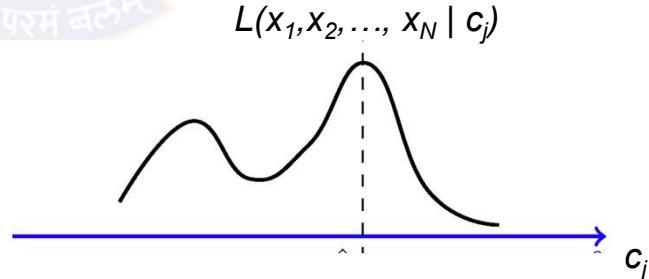
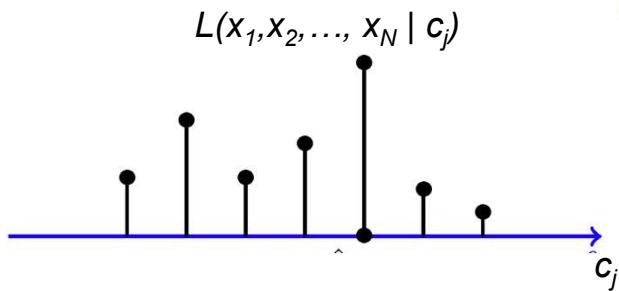
$$\hat{Y} \leftarrow \operatorname{argmax}_{y_k} P(Y = y_k)\prod_i P(X_i | Y = y_k)$$

MLE is a special case of MAP, when prior follows a uniform distribution.

Foundations - Maximum Likelihood Estimation

Maximum Likelihood Estimation - Overview

- For a case of N-input parameters $X = \{x_1, x_2, \dots, x_N\}$ and output variable of K classes $C = \{c_1, c_2, \dots, c_K\}$, likelihood is $P(x_1, x_2, \dots, x_N | c_j)$
- If the input parameters are discrete, the likelihood function is defined as the probability mass function of likelihood,
i.e. $L(x_1, x_2, \dots, x_N | c_j) = P(x_1, x_2, \dots, x_N | c_j)$
- For continuous random variables, the same becomes probability density function
i.e. $L(x_1, x_2, \dots, x_N | c_j) = f(x_1, x_2, \dots, x_N | c_j)$
- Maximum likelihood estimate (MLE) of C, shown by as C_{ML} is a value of c_j that maximizes the likelihood function $L(x_1, x_2, \dots, x_N | c_j)$
i.e. $C_{ML} = \operatorname{argmax}_{c_j} L(x_1, x_2, \dots, x_N | c_j)$



Foundations - Maximum Likelihood Estimation

Maximum Likelihood Estimation - Example

A bag has 3 balls (red or blue). If 4 random pick and replacement events results in the following results – blue, red, blue, blue; what is the number of blue balls that gives max probability for the observed data ?

$$P_{X_i}(x) = \begin{cases} \frac{\theta}{3} & \text{for } x = 1 \\ 1 - \frac{\theta}{3} & \text{for } x = 0 \end{cases}$$

$$\begin{aligned} P_{X_1 X_2 X_3 X_4}(1, 0, 1, 1) &= \frac{\theta}{3} \cdot \left(1 - \frac{\theta}{3}\right) \cdot \frac{\theta}{3} \cdot \frac{\theta}{3} \\ &= \left(\frac{\theta}{3}\right)^3 \left(1 - \frac{\theta}{3}\right). \end{aligned}$$

θ	$P_{X_1 X_2 X_3 X_4}(1, 0, 1, 1; \theta)$
0	0
1	0.0247
2	0.0988
3	0

In this segment

Logistic Regression - Cross entropy error function

- Logistic Regression Function
- Cross entropy Error function
- Optimal Solution



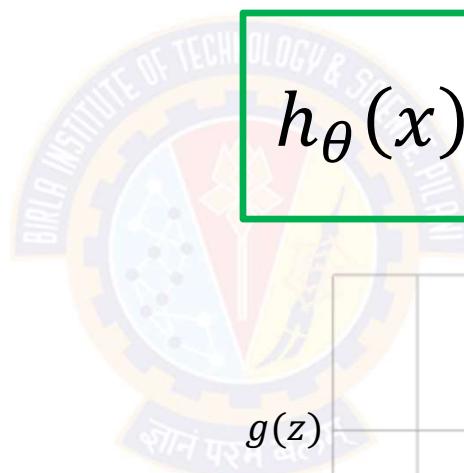
Logistic Regression Function

- Want $0 \leq h_\theta(x) \leq 1$

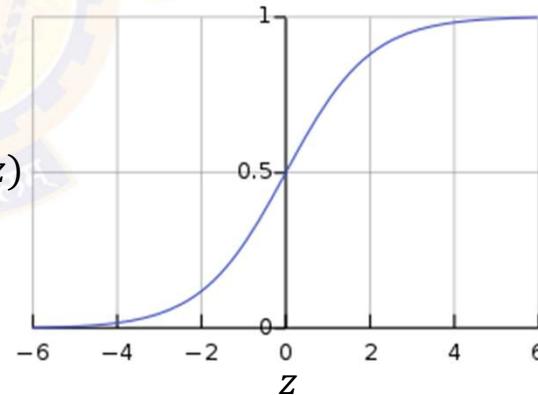
- $h_\theta(x) = g(\theta^\top x)$,

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

- Sigmoid function
- Logistic function



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



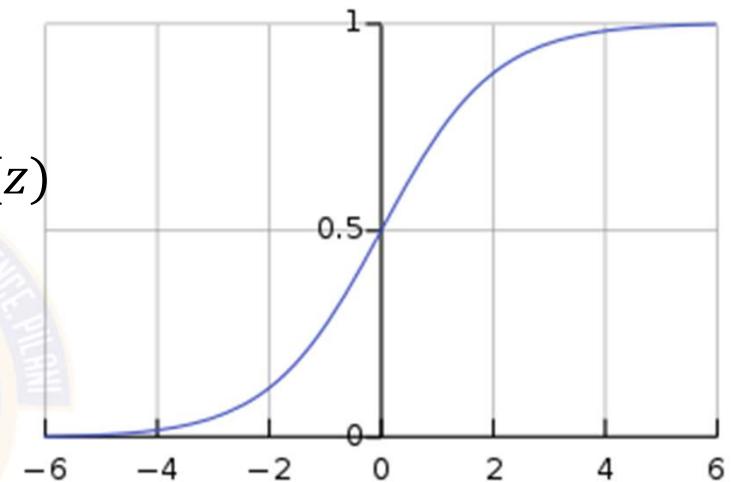
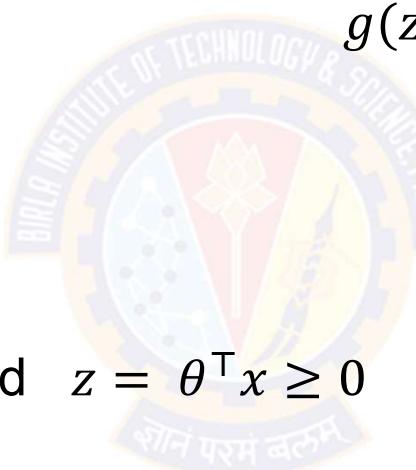
Slide credit: Andrew Ng

Logistic regression Classifier

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

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

$$z = \theta^T x$$



Predict “y = 1” if $h_{\theta}(x) \geq 0.5$ and $z = \theta^T x \geq 0$

Predict “y = 0” if $h_{\theta}(x) < 0.5$ and $z = \theta^T x < 0$

Decision boundary

- $h_{\theta}(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$

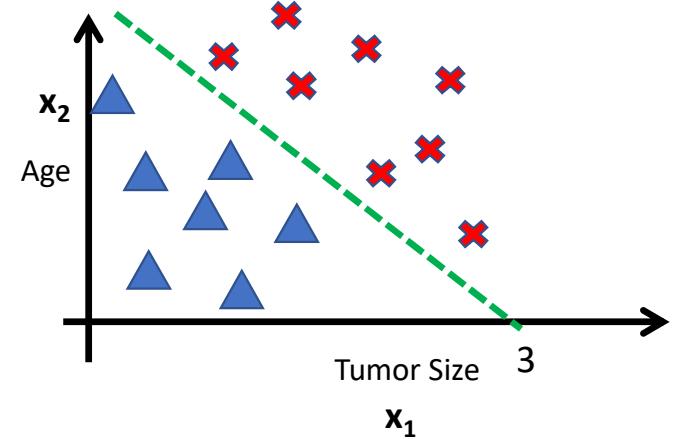
E.g., $\theta_0 = -3, \theta_1 = 1, \theta_2 = 1$

Predict " $y = 1$ " if $-3 + x_1 + x_2 \geq 0$
i.e $x_1 + x_2 \geq 3$

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

$$z = \theta^T x$$

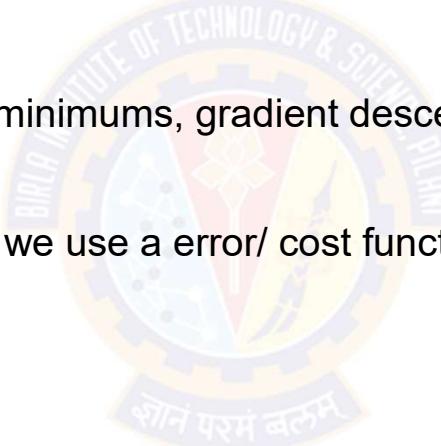
$$g(z) = \frac{1}{1 + e^{-z}}$$



Slide credit: Andrew Ng

Logistic Regression Error Function

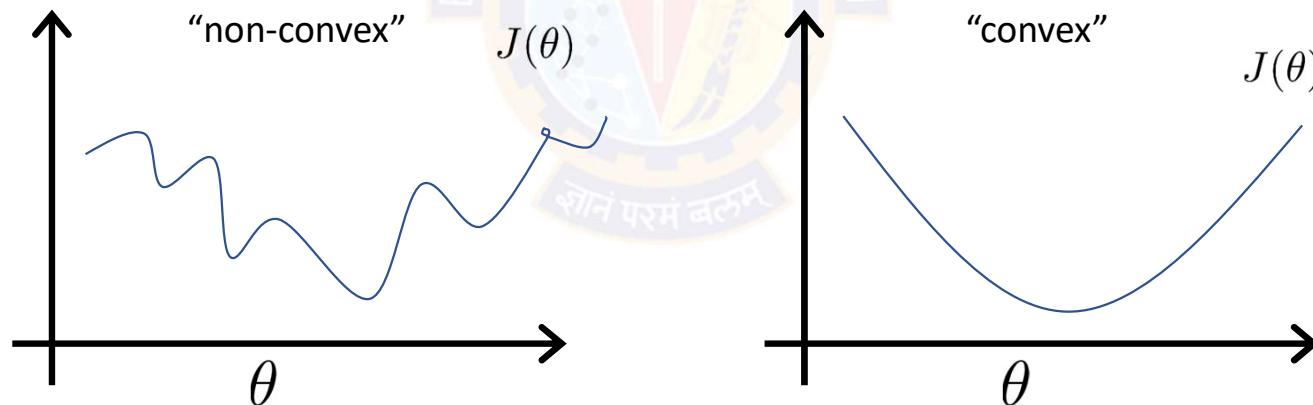
- Our prediction function is non-linear (due to sigmoid transform)
- Squaring this prediction as we do in MSE results in a non-convex function with many local minima.
- If our cost function has many local minimums, gradient descent may not find the optimal global minimum.
- So instead of Mean Squared Error, we use a error/ cost function called [Cross-Entropy](#), also known as Log Loss.



MSE Cost Function

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

$$\text{Cost}(h_\theta(x^{(i)}), y^{(i)}) = \frac{1}{2} (h_\theta(x^{(i)}) - y^{(i)})^2$$



Entropy

- It rained heavily in Shillong yesterday
- There was a heavy rainfall in Rajasthan last night.
- The amount of information conveyed by a message is inversely proportional to its probability of occurrence. That is

$$I_k \propto -\frac{1}{p_k}$$

Entropy

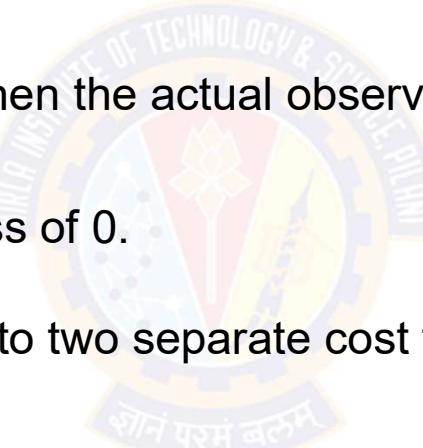
- The mathematical operator satisfies above properties is the logarithmic operator. Therefore,

$$I_k = \log_r \frac{1}{p_k} \text{ units}$$



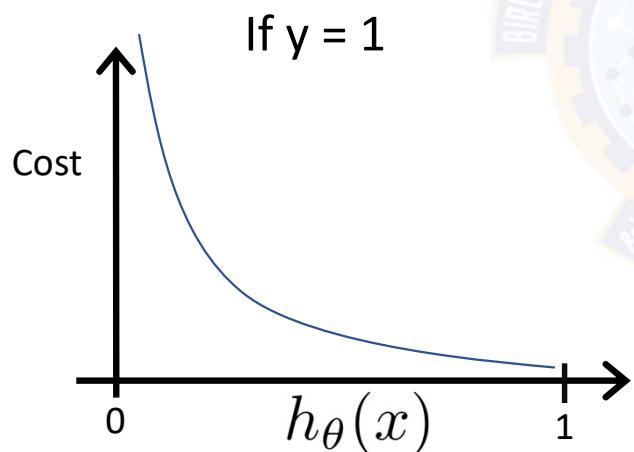
Cross Entropy

- Cross-entropy loss, or log loss, measures the performance of a classification model whose output is a probability value between 0 and 1.
- Cross-entropy loss increases as the predicted probability diverges from the actual label.
- So predicting a probability of .012 when the actual observation label is 1 would be bad and result in a high loss value.
- A perfect model would have a log loss of 0.
- Cross-entropy loss can be divided into two separate cost functions:
one for $y=1$ and
one for $y=0$.



Logistic Regression Cost Function (cross entropy)

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

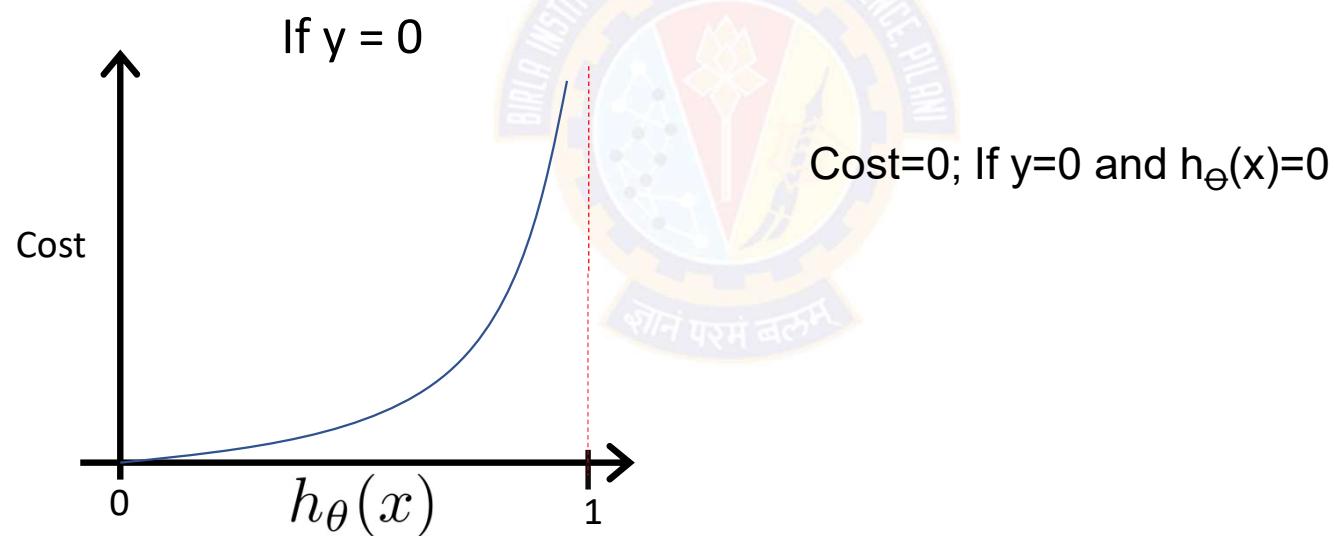


BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE
Cost = 0 if $y = 1, h_\theta(x) = 1$
But as $h_\theta(x) \rightarrow 0$
 $Cost \rightarrow \infty$

Captures intuition that if $h_\theta(x) = 0$,
(predict $P(y = 1|x; \theta) = 0$), but $y = 1$,
we'll penalize learning algorithm by a very
large cost.

Logistic Regression Cost Function (cross entropy)

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



Logistic Regression Cost Function (cross entropy)

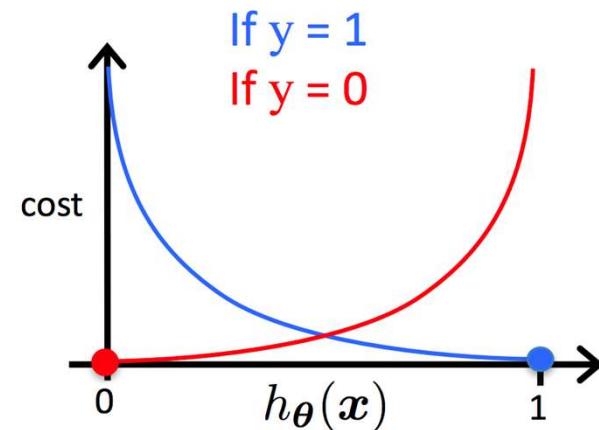
$$\begin{aligned} J(\theta) &= \frac{1}{m} \sum_{i=1}^m \text{Cost}(h_\theta(x^{(i)}), y^{(i)}) \\ &= -\frac{1}{m} \left[\sum_{i=1}^m y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_\theta(x^{(i)})) \right] \end{aligned}$$

To fit parameters θ : [Apply Gradient Descent Algorithm](#)

$$\min_{\theta} J(\theta)$$

To make a prediction given new x

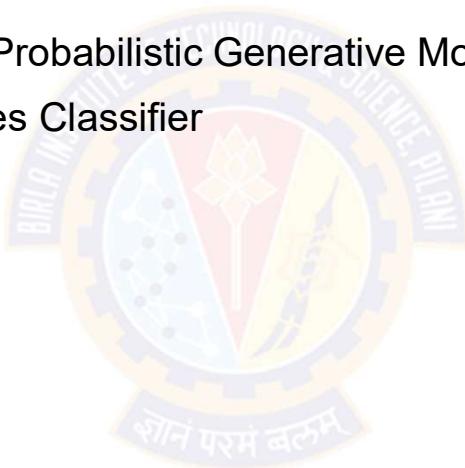
$$\text{Output } h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$$



In this segment

Logistic Regression - Probabilistic Discriminative

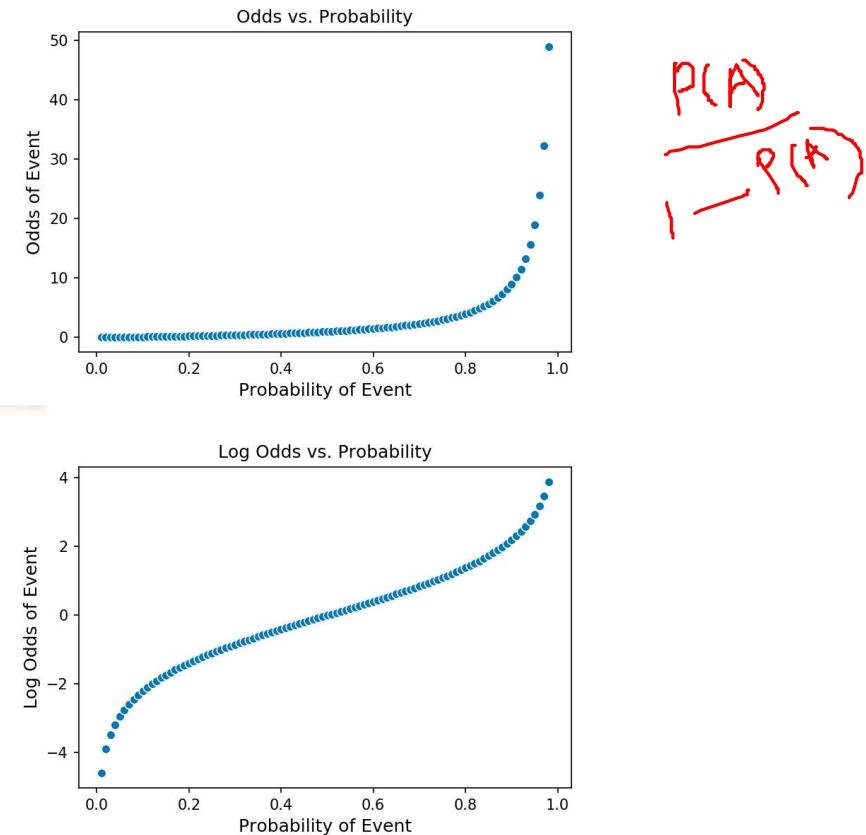
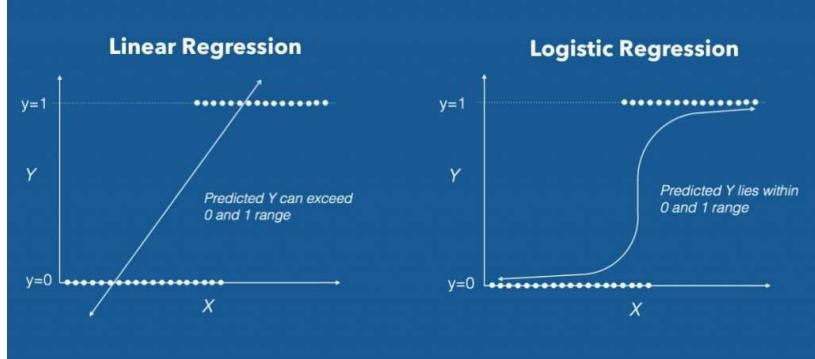
- Logistic Regression - Log of Odds
- Logistic Regression – Illustration
- Probabilistic discriminative versus Probabilistic Generative Models
- Logistic Regression v/s Naïve Bayes Classifier



Logistic Regression is Probabilistic Discriminative

Logistic Regression – where it works

- Linear discriminants have limitations on some data sets which are not linearly separable
- Logistic Regression can be used to approximate such datasets, using log of odds



Logistic Regression is Probabilistic Discriminative

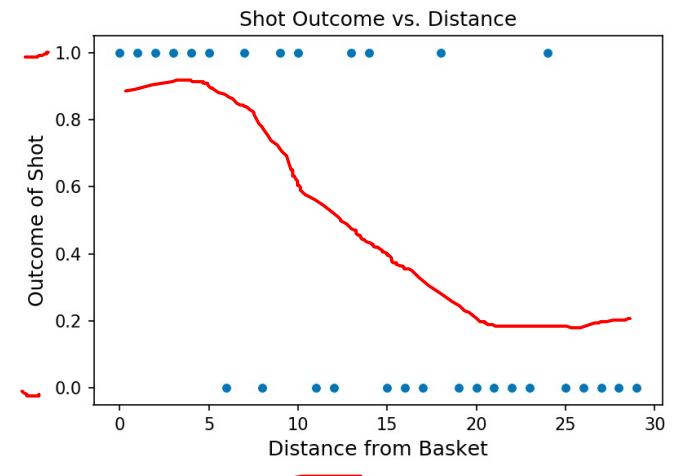
Logistic Regression – Log of Odds

- Log of Odds – $O = \frac{p(X)}{1-p(X)}$
- Use log of odds z such that – $Z_i = \ln\left(\frac{P_i}{1-P_i}\right) = \alpha + \beta_1 x_1 + \dots + \beta_n x_n$
- Solving for P_i , we get $P_i = E(y=1|x_i) = \frac{e^z}{1+e^z} = \frac{e^{\alpha+\beta_i x_i}}{1+e^{\alpha+\beta_i x_i}}$
- Which is sigmoid on z (log of odds)
- Coefficients are determined using Maximum likelihood estimator, while keeping cost function(cross entropy error function) minimum

Logistic Regression is Probabilistic Discriminative

Logistic Regression – Illustration

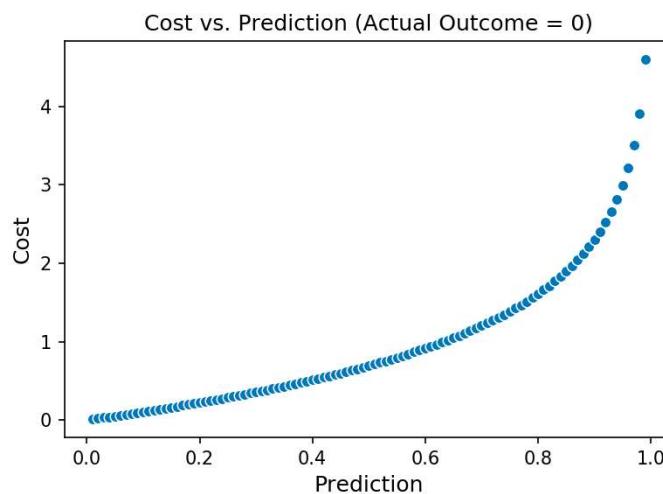
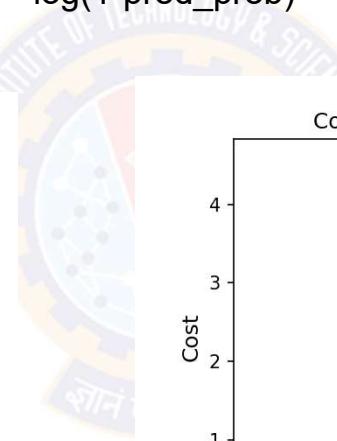
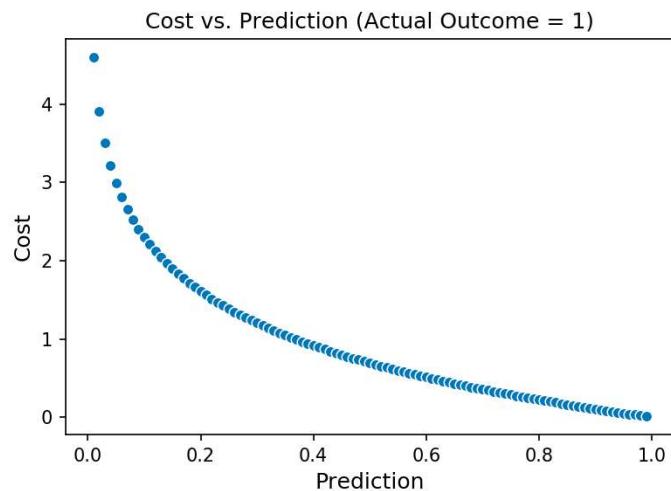
- Basket Ball – calculate the probability of shooting ball into a basket based on distance from basket
- Input parameter (X) – distance from basket
- Outcome (Y) – Yes or No
- Data distribution is not linearly separable
- The parameter for sigmoid, $Z = B_0 + B_1 * X$
 - i.e. $Z = B_0 + B_1 * \text{distance_from_basket}$
 - Also $Z = \log(\text{odds_of_making_shot})$
- Probability of making shot (Y) = $1 / [1 + e^{-Z}]$



Logistic Regression is Probabilistic Discriminative

Logistic Regression – Illustration

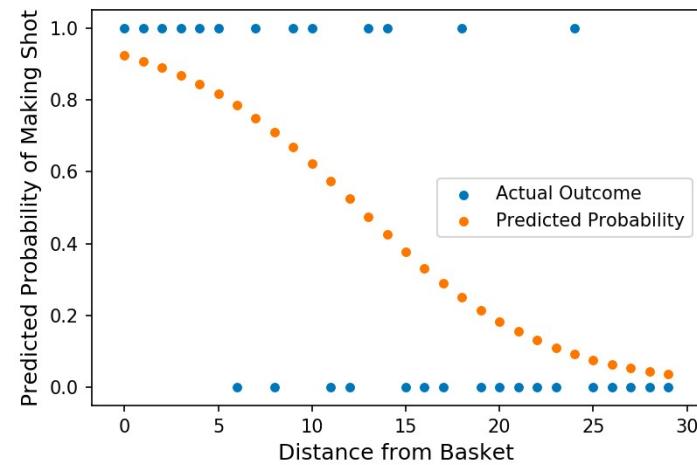
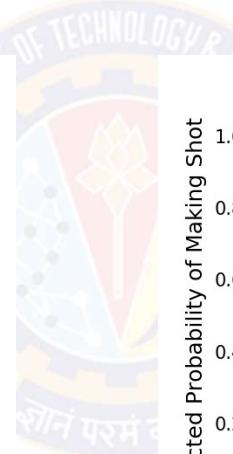
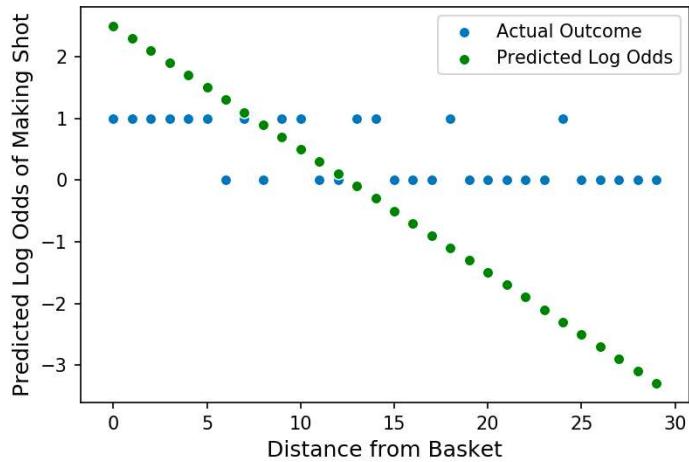
- Cost function calculation
 - If Actual Outcome = 1, then Cost = $-\log(\text{pred_prob})$ $y=1$
 - Else if Actual Outcome = 0, then Cost = $-\log(1-\text{pred_prob})$
- Compute total cost



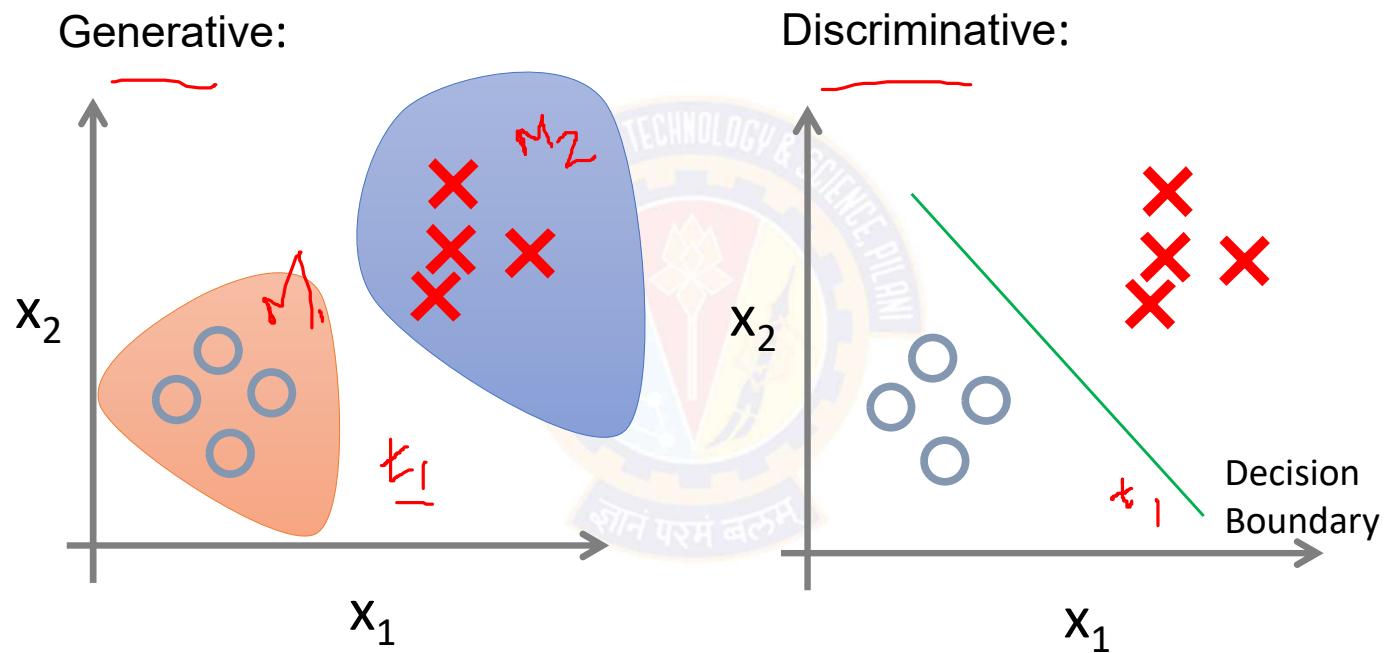
Logistic Regression is Probabilistic Discriminative

Logistic Regression – Illustration

- Using optimization techniques (linear regression), find B_0 and B_1 that minimize total cost
- Calculate Z in each case and penalize based on cost function



Probabilistic Generative v/s Discriminative Model

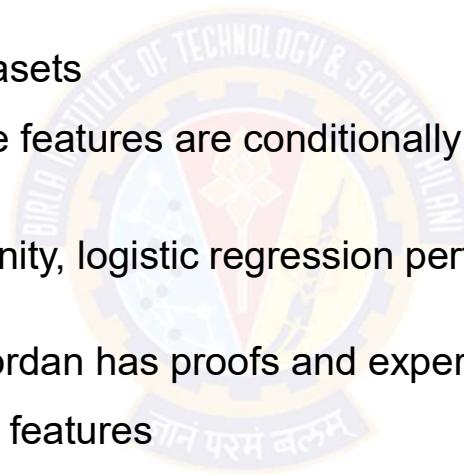


Probabilistic Generative v/s Discriminative Model

Generative	Discriminative
Ex: Naïve Bayes <u>Ex: Naïve Bayes</u>	Ex: Logistic Regression <u>Ex: Logistic Regression</u>
Estimate $P(Y)$ and $P(X Y)$ <u>Estimate $P(Y)$</u> <u>and $P(X Y)$</u>	Finds class label directly $P(Y X)$ <u>Finds class label directly</u> <u>$P(Y X)$</u>
Prediction $\hat{y} = \text{argmax}_y P(Y = y)P(X = x Y = y)$ <u>$\hat{y} = \text{argmax}_y$</u>	Prediction $\hat{y} = P(Y = y X = x)$

Naïve Bayes versus Logistic Regression

- Naïve Bayes are Generative Models which Logistic Regression are Discriminative Models
- Naïve Bayes easy to construct
- Naïve Bayes better on smaller datasets
- Naive Bayes also assumes that the features are conditionally independent. Real data sets are never perfectly independent
- When the training size reaches infinity, logistic regression performs better than the generative model Naive Bayes.
 - Optional reading by Ng and Jordan has proofs and experiments
- Logistic regression allows arbitrary features



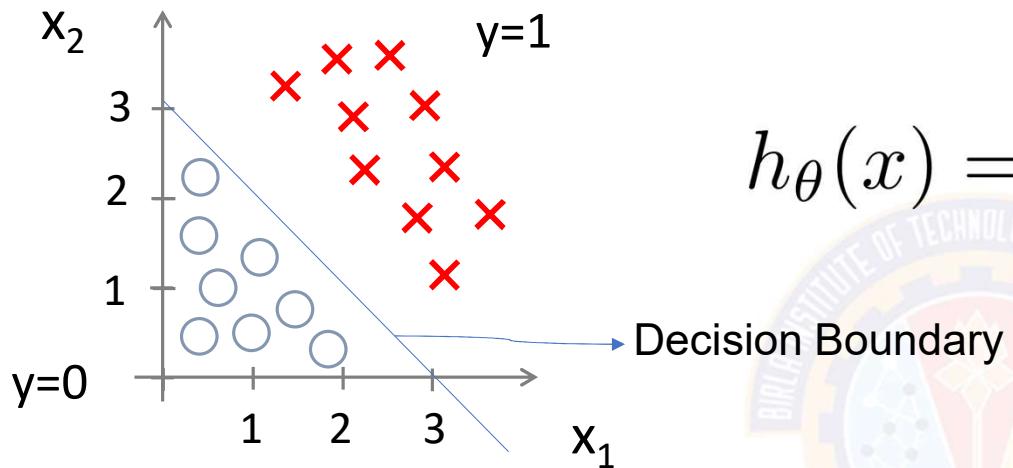
In this segment

Logistic Regression – Miscellaneous Topics

- Logistic Regression – Decision Boundary
- Overfitting - Countermeasures
- Interpretability of Logistic Regression



Decision Boundary



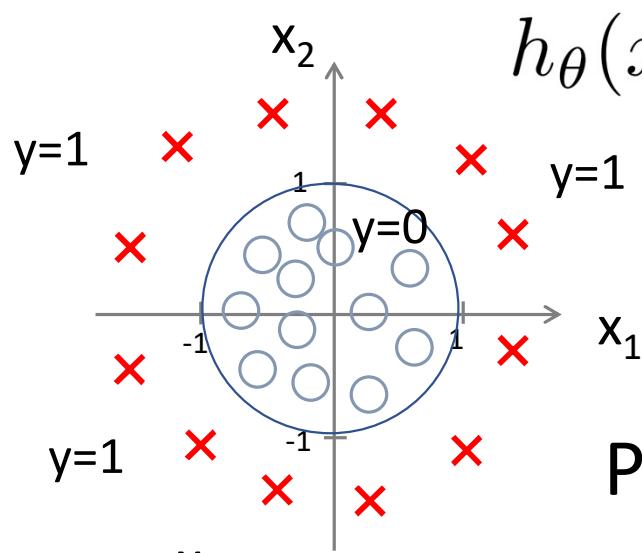
$$h_{\theta}(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$$

$$\theta = \begin{bmatrix} -3 \\ 1 \\ 1 \end{bmatrix}$$

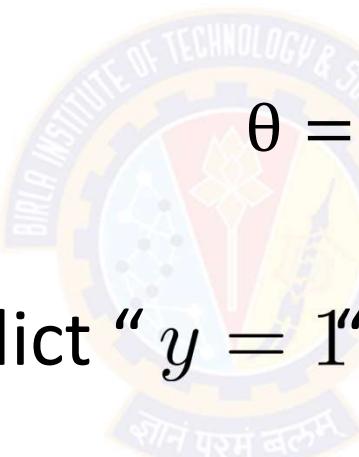
Predict " $y = 1$ " if $-3 + x_1 + x_2 \geq 0$

Decision Boundary

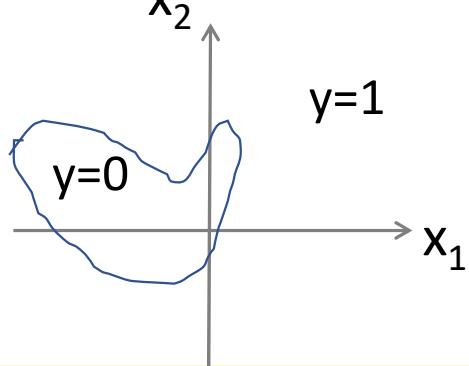
Non-linear decision boundaries



$$h_{\theta}(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2)$$


$$\theta = \begin{bmatrix} -1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Predict " $y = 1$ " if $-1 + x_1^2 + x_2^2 \geq 0$

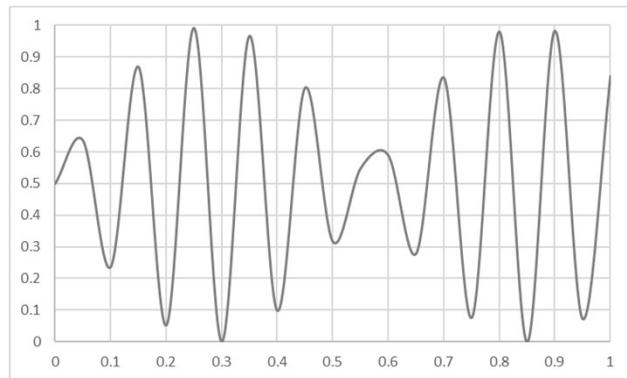


$$h_{\theta}(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_1^2 x_2 + \theta_5 x_1^2 x_2^2 + \theta_6 x_1^3 x_2 + \dots)$$

Overfitting vs Underfitting

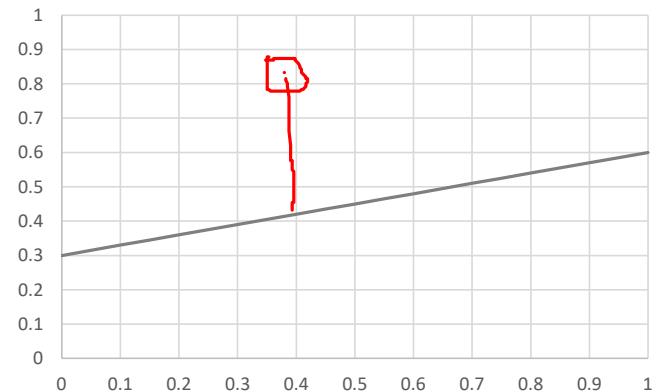
Overfitting

- Fitting the data too well
 - Features are noisy / uncorrelated to concept
 - Modeling process very sensitive (powerful)



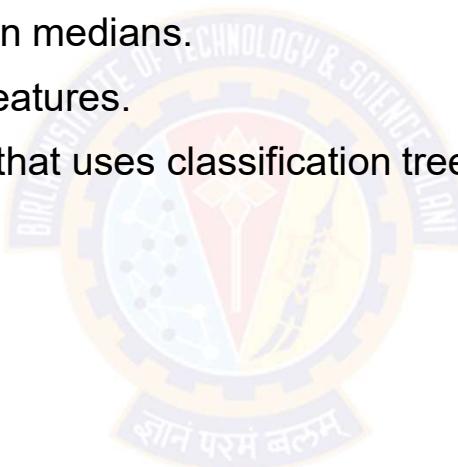
Underfitting

- Learning too little of the true concept
 - Features don't capture concept
 - Too much bias in model



Handling Missing Values

- Replace missing values with column averages (i.e. replace missing values in feature 1 with the average for feature 1).
- Replace missing values with column medians.
- Remove records that are missing features.
- Use a machine learning technique that uses classification trees, e.g. Decision tree



Interpretability of Logistic Regression

$h_\theta(x)$ = estimated probability that $y = 1$ on input x

Example: If $x = \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = \begin{bmatrix} 1 \\ \text{tumorSize} \end{bmatrix}$

and $h_\theta(x) = 0.7$

Tell patient that 70% chance of tumor being malignant

Probability that $y = 1$, given x , parameterized by θ

$$P(y = 0|x; \theta) + P(y = 1|x; \theta) = 1$$

$$P(y = 0|x; \theta) = 1 - P(y = 1|x; \theta)$$

Interpretability of Logistic Regression

- Outcome in logistic regression is a probability between 0 and 1.
- Big advantage over models that can only provide the final classification. Knowing that an instance has a 99% probability for a class compared to 51% makes a big difference.

$$\frac{P(y=1)}{1 - P(y=1)} = odds = \exp(\beta_0 + \beta_1 x_1 + \dots + \beta_p x_p)$$

- Adding 1 to one of the feature values

$$\frac{odds_{x_j+1}}{odds} = \frac{\exp(\beta_0 + \beta_1 x_1 + \dots + \beta_j(x_j + 1) + \dots + \beta_p x_p)}{\exp(\beta_0 + \beta_1 x_1 + \dots + \beta_j x_j + \dots + \beta_p x_p)}$$

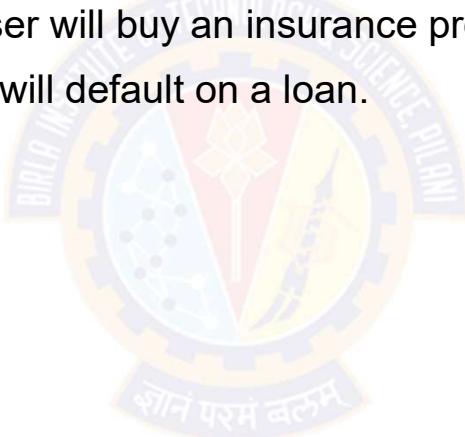
- Using $\frac{\exp(a)}{\exp(b)} = \exp(a - b)$

$$\frac{odds_{x_j+1}}{odds} = \exp(\beta_j(x_j + 1) - \beta_j x_j) = \exp(\beta_j)$$

- Change in x_j by one unit increases the log odds ratio by the value of the corresponding weight.

Logistic Regression Applications

- **Credit Card Fraud** : Predicting if a given credit card transaction is fraud or not
- **Health** : Predicting if a given mass of tissue is benign or malignant
- **Marketing** : Predicting if a given user will buy an insurance product or not
- **Banking** : Predicting if a customer will default on a loan.



References

- <http://www.cs.cmu.edu/~tom/NewChapters.html>
 - <http://ai.stanford.edu/~ang/papers/nips01-discriminativegenerative.pdf>
 - https://medium.com/@sangha_deb/naive-bayes-vs-logistic-regression-a319b07a5d4c
 - <https://www.youtube.com/watch?v=-la3q9d7AKQ>
- Interpretability
- <https://christophm.github.io/interpretable-ml-book/logistic.html>

In this segment

Decision Tree - Introduction

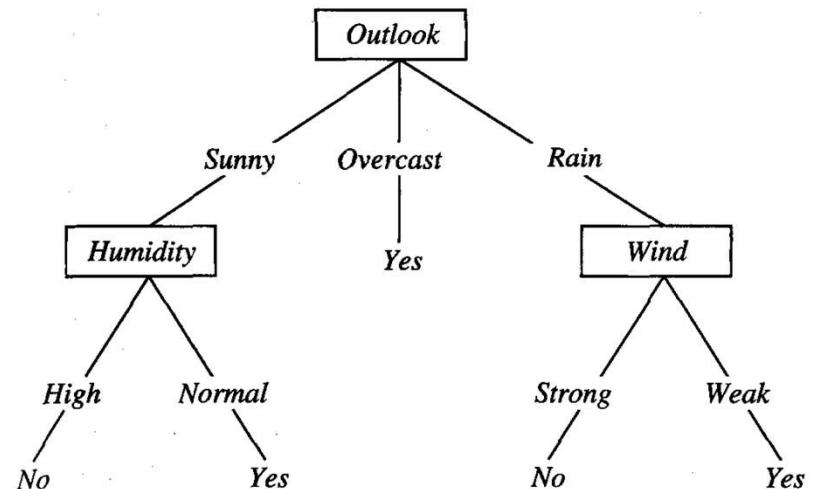
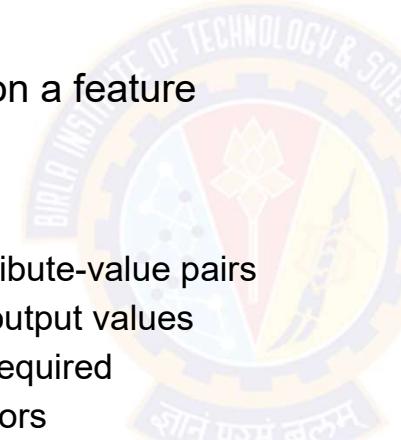
- Decision Tree Representation
- Entropy
- Information Gain



Decision Tree - Introduction

Decision Tree - Representation

- One of the most widely used methods for inductive inference
- Classifies instances by sorting them down the tree from the root to some leaf node
- Non-Leaf Node – test (condition) on a feature
- Leaf node – Class label
- Applicable when:
 - Instances are represented by attribute-value pairs
 - The target function has discrete output values
 - Disjunctive descriptions may be required
 - The training data may contain errors
 - The training data may contain missing attribute values
- Algorithms used for building Decision Tree:
 - ID3
 - ASSISTANT
 - C4.5



Decision Tree - Introduction

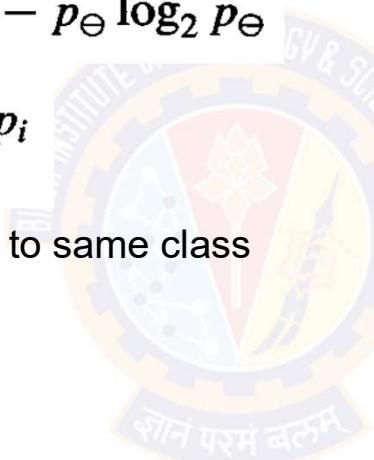
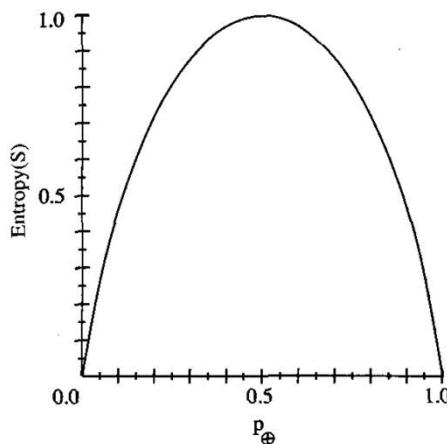
Entropy

- Characterizes the (im)purity of an arbitrary collection of datapoints

$$Entropy(S) \equiv -p_{\oplus} \log_2 p_{\oplus} - p_{\ominus} \log_2 p_{\ominus}$$

$$Entropy(S) \equiv \sum_{i=1}^c -p_i \log_2 p_i$$

- Entropy is '0' if all members belong to same class



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

$$\begin{aligned}
 Entropy([9+, 5-]) &= -(9/14) \log_2(9/14) - (5/14) \log_2(5/14) \\
 &= 0.940
 \end{aligned}$$

Decision Tree - Introduction

Information Gain

- Measure of the effectiveness of an attribute in classifying the training data
- Defined as the expected reduction in entropy caused by partitioning the examples according to the attribute

$$Gain(S, A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

where $Values(A)$ is the set of all possible values for attribute A , and S_v , is the subset of S for which attribute A has value v



Decision Tree - Introduction

Information Gain - Example

- If we take 'Wind' (possible values – *Strong, Weak*) as the attribute to sort the dataset by, then the information gain will be calculated as:

$$Values(Wind) = Weak, Strong$$

$$S = [9+, 5-]$$

$$S_{Weak} \leftarrow [6+, 2-]$$

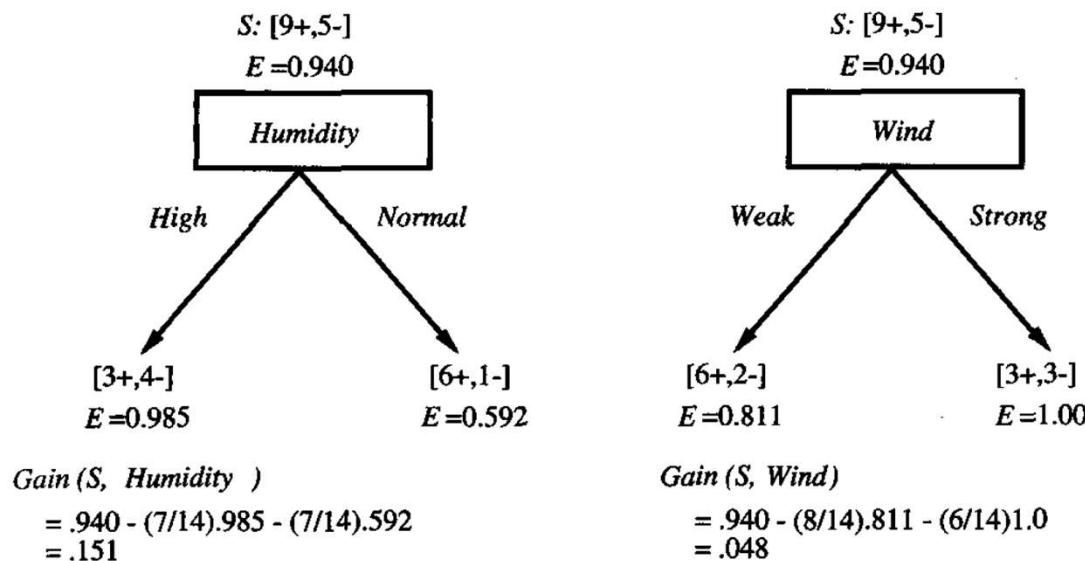
$$S_{Strong} \leftarrow [3+, 3-]$$

$$\begin{aligned} Gain(S, Wind) &= Entropy(S) - \sum_{v \in \{Weak, Strong\}} \frac{|S_v|}{|S|} Entropy(S_v) \\ &= Entropy(S) - (8/14)Entropy(S_{Weak}) \\ &\quad - (6/14)Entropy(S_{Strong}) \\ &= 0.940 - (8/14)0.811 - (6/14)1.00 \\ &= 0.048 \end{aligned}$$

Decision Tree - Introduction

Information Gain - Significance

- Information Gain is the basis for ID3 Algorithm to choose the attributes for growing the tree
- Ex: If we compare attributes *Wind* and *Humidity*, using Information Gain



- Clearly Humidity is better among these two.

In this segment

ID3 Algorithm for Decision Tree Learning

- ID3 Algorithm for Decision Tree
- Illustration
- Search in Hypothesis Space



ID3 Algorithm for Decision Tree Learning

ID3 Algorithm

ID3(Examples, Target_attribute, Attributes)

Examples are the training examples. *Target_attribute* is the attribute whose value is to be predicted by the tree. *Attributes* is a list of other attributes that may be tested by the learned decision tree. Returns a decision tree that correctly classifies the given *Examples*.

- Create a *Root* node for the tree
 - If all *Examples* are positive, Return the single-node tree *Root*, with label = +
 - If all *Examples* are negative, Return the single-node tree *Root*, with label = -
 - If *Attributes* is empty, Return the single-node tree *Root*, with label = most common value of *Target_attribute* in *Examples*
 - Otherwise Begin
 - $A \leftarrow$ the attribute from *Attributes* that best* classifies *Examples*
 - The decision attribute for *Root* $\leftarrow A$
 - For each possible value, v_i , of *A*,
 - Add a new tree branch below *Root*, corresponding to the test $A = v_i$
 - Let $Examples_{v_i}$ be the subset of *Examples* that have value v_i for *A*
 - If $Examples_{v_i}$ is empty
 - Then below this new branch add a leaf node with label = most common value of *Target_attribute* in *Examples*
 - Else below this new branch add the subtree $ID3(Examples_{v_i}, Target_attribute, Attributes - \{A\})$
 - End
 - Return *Root*
-

* The best attribute is the one with highest *information gain*.

ID3 Algorithm for Decision Tree Learning

ID3 Algorithm - Illustration

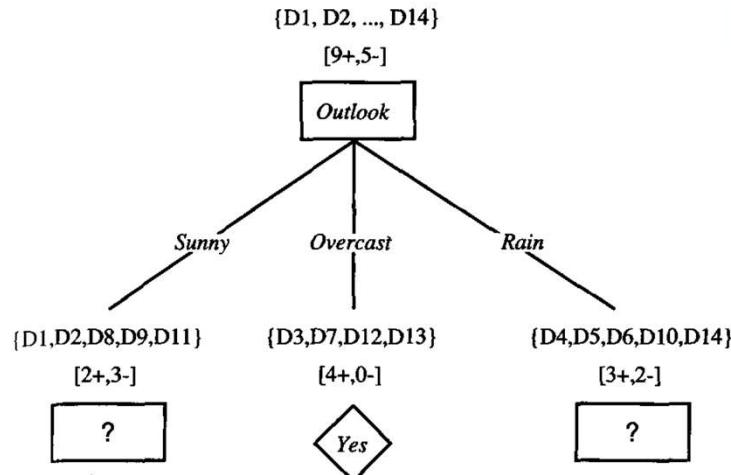
- Step-1: Root node
 - Outlook*

$$Gain(S, \text{Outlook}) = 0.246$$

$$Gain(S, \text{Humidity}) = 0.151$$

$$Gain(S, \text{Wind}) = 0.048$$

$$Gain(S, \text{Temperature}) = 0.029$$



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

ID3 Algorithm for Decision Tree Learning

ID3 Algorithm - Illustration

- Step-2: Attribute for the sub-tree Outlook=Sunny
 - Humidity

$$S_{\text{sunny}} = \{D1, D2, D8, D9, D11\}$$

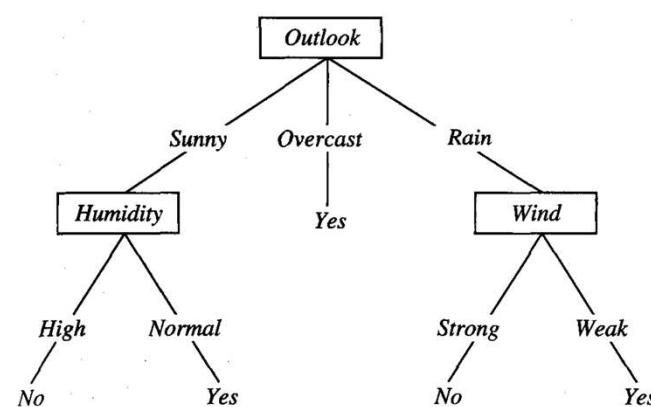
$$\text{Gain}(S_{\text{sunny}}, \text{Humidity}) = .970 - (3/5) 0.0 - (2/5) 0.0 = .970$$

$$\text{Gain}(S_{\text{sunny}}, \text{Temperature}) = .970 - (2/5) 0.0 - (2/5) 1.0 - (1/5) 0.0 = .570$$

$$\text{Gain}(S_{\text{sunny}}, \text{Wind}) = .970 - (2/5) 1.0 - (3/5) .918 = .019$$

- Continuing this process for each new leaf node...

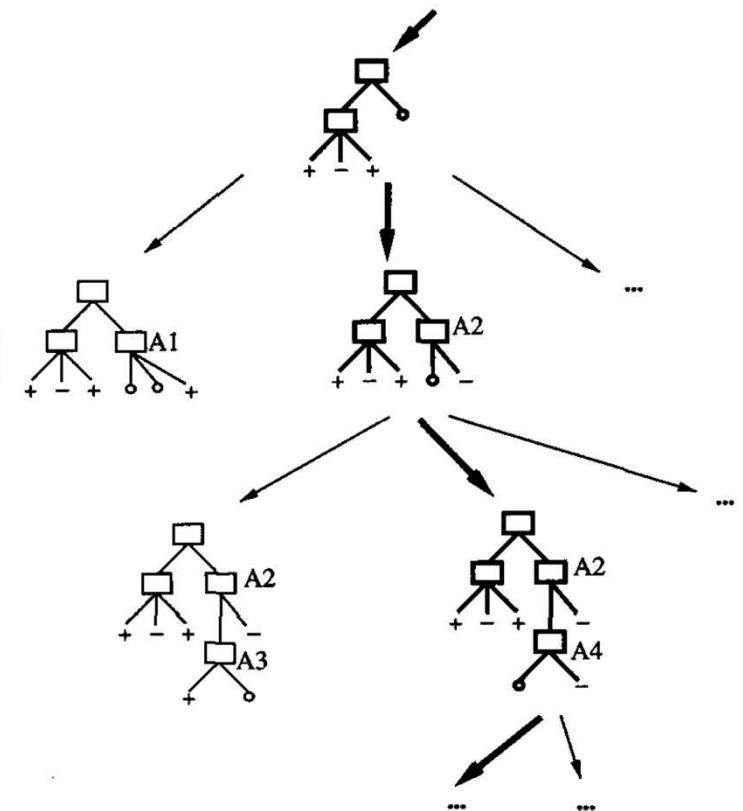
Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	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



ID3 Algorithm for Decision Tree Learning

Search in Hypothesis Space – ID3

- A simple-to complex, hill-climbing search is conducted in the Hypothesis space, using Information Gain as the guiding factor
- ID3's hypothesis space of all decision trees is a complete space of finite discrete-valued functions, relative to the available attributes
- Maintains only a single current hypothesis as it searches through the space of decision trees
- ID3 in its pure form performs no backtracking in its search
 - Susceptible to converging to locally optimal solutions that are not globally optimal
- Uses all training examples at each step in the search to make statistically based decisions regarding how to refine its current hypothesis



In this segment

Decision Tree – Miscellaneous Topics

- Prefer Short Vs Long Hypothesis - Occam's Razor
- Avoiding Overfitting in Decision Tree
- Reduced Error Pruning
- Rule Post Pruning
- Alternate Measures for Attribute Selection



Decision Tree – Miscellaneous Topics

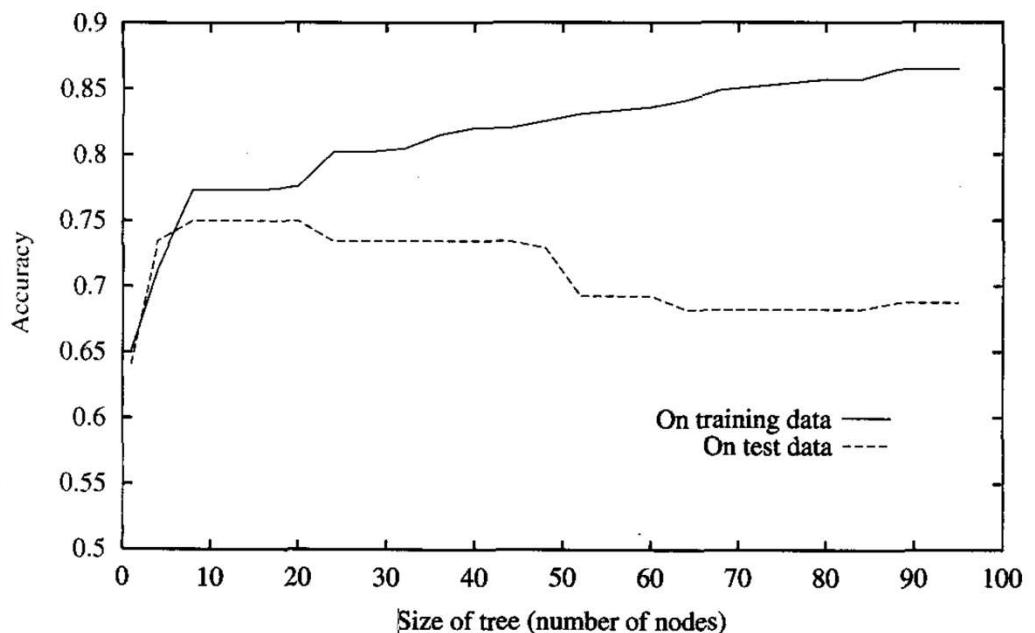
Prefer Short Vs Long Hypothesis – Occam's Razor

- Approximate inductive bias of ID3: Shorter trees are preferred over larger trees
- Occam's razor: Prefer the simplest hypothesis that fits the data (William of Occam, circa 1320)
- Argument in favor - There are fewer short hypotheses than long ones
- Issue - Easy to produce long hypothesis that fit training data and can be generalised
- Issue – Size of hypothesis is determined by the internal representation used internally by the learner
- On the contrary, evolution will create internal representations that make the learning algorithm's inductive bias a self-fulfilling prophecy, simply because it can alter the representation easier than it can alter the learning algorithm

Decision Tree – Miscellaneous Topics

Overfitting in Decision Tree

- Overfitting – A hypothesis overfits the training examples if some other hypothesis that fits the training examples less well actually performs better over the entire distribution of instances
- Random noise can lead to overfitting
 - Ex: (Outlook = Sunny, Temperature = Hot, Humidity = Normal, Wind = Strong, PlayTennis = No)
- Even with noise-free data, smaller set of examples with leaf nodes can cause overfitting



Decision Tree – Miscellaneous Topics

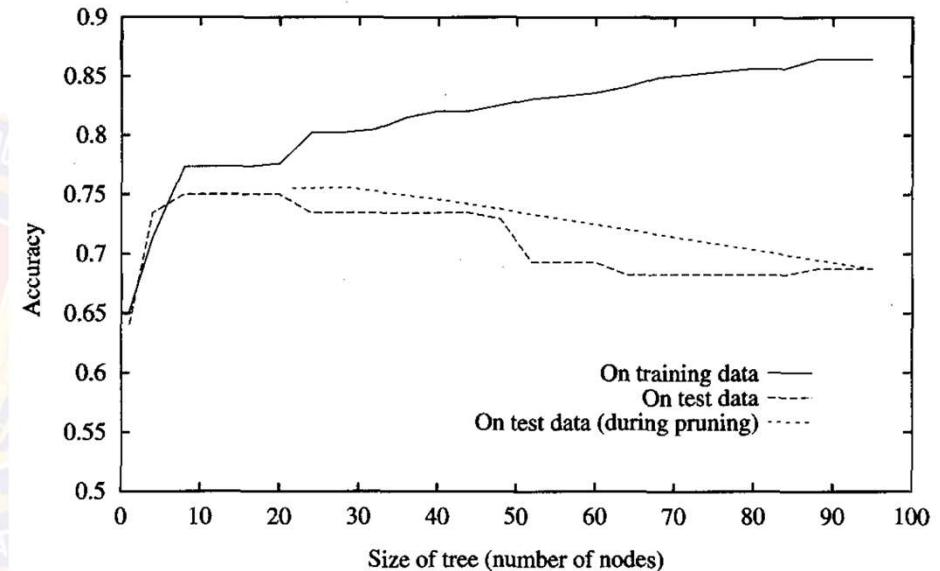
Avoiding Overfitting in Decision Tree

- **Stop growing** the tree earlier, before it reaches the point where it perfectly classifies the training data
- Allow the tree to overfit the data, and then **post-prune** the tree
- Determining correct tree size (for both stop growing and post-pruning)
 - Use a separate set of examples, distinct from the training examples, to evaluate the utility of post-pruning nodes from the tree – **training and validation set** approach
 - Use all the available data for training, but apply a statistical test to estimate whether expanding (or pruning) a particular node is likely to produce an improvement beyond the training set
 - Use an explicit measure of the complexity for encoding the training examples and the decision tree, halting growth of the tree when this encoding size is minimized

Decision Tree – Miscellaneous Topics

Reduced Error Pruning

- Consider each of the decision nodes in the tree to be candidates for pruning
 - Remove the subtree rooted at the node
 - Make it a leaf node
 - Assign the most common classification of the training examples affiliated with that node
- Removal is done only if the resulting pruned tree performs no worse over the validation set
- Nullifies coincidental irregularities in assigning leaf nodes
- Pruning continues until further pruning is harmful



Decision Tree – Miscellaneous Topics

Rule Post Pruning

- Infer the decision tree from the training set, growing the tree until the training data is fit as well as possible and allowing overfitting to occur
- Convert the learned tree into an equivalent set of rules by creating one rule for each path from the root node to a leaf node
- Prune (generalize) each rule by removing any preconditions that result in improving its estimated accuracy
- Sort the pruned rules by their estimated accuracy, and consider them in this sequence when classifying subsequent instances
- Benefits:
 - Allows distinguishing among the different contexts in which a decision node is used
 - Removes the distinction between attribute tests that occur near the root of the tree and those that occur near the leaves
 - Improves readability as rules are often easier for to understand

Decision Tree – Miscellaneous Topics

Alternate Measures for Attribute Selection

- Often, Information Gain comes with natural bias towards attributes with many values
- Alternate measure – Gain Ratio
- We start with split information, which tells how broadly and uniformly the attribute splits the data

$$SplitInformation(S, A) \equiv - \sum_{i=1}^c \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|}$$

- Gain Ratio is defined as – $GainRatio(S, A) \equiv \frac{Gain(S, A)}{SplitInformation(S, A)}$



BITS Pilani
Pilani | Dubai | Goa | Hyderabad

Optimization Foundations - Basics

Dr. Chetana Gavankar

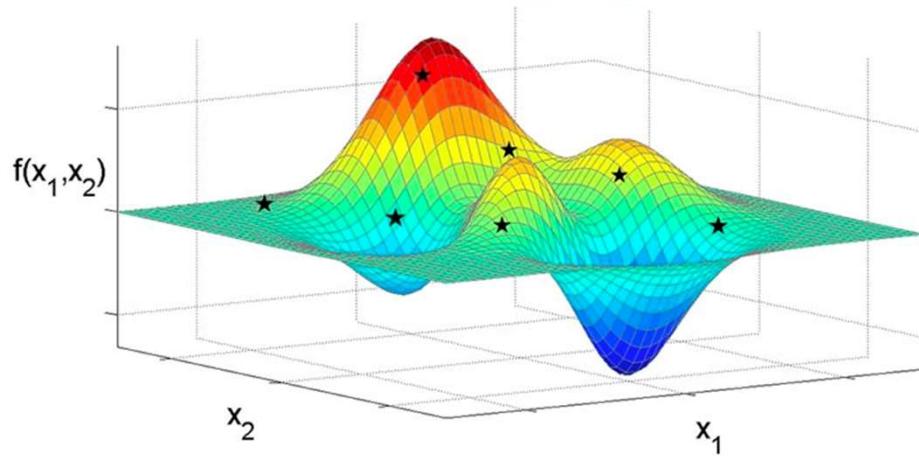
In this segment

- Constrained Optimization
- Unconstrained Optimization



Optimization Foundations - Basics

- Optimization is a methodology for finding the maximum or minimum value of a function
- Ex: In minimizing cost functions, error functions etc.
- For a function $f(x)$, it aims to find the value of x that gives max or min of $f(x)$, with x bound to some constraints or without any constraints



Optimization Problem

- Optimization problem is typically written:

Minimize $f(x)$

subject to

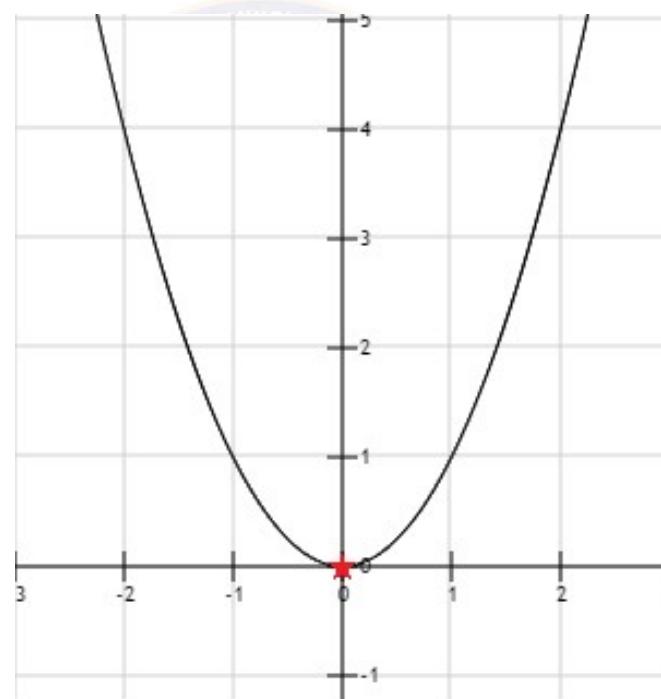
$$g_i(x) = 0, \quad i=1, \dots, p$$

$$h_i(x) \leq 0, \quad i=1, \dots, m$$

- $f(x)$ is called the objective function
- By changing x (the optimization variable) we wish to find a value x^* for which $f(x)$ is at its minimum.
- p functions of g_i define equality constraints and
- m functions of h_i define inequality constraints.
- The value we find MUST respect these constraints!

Unconstrained Optimization

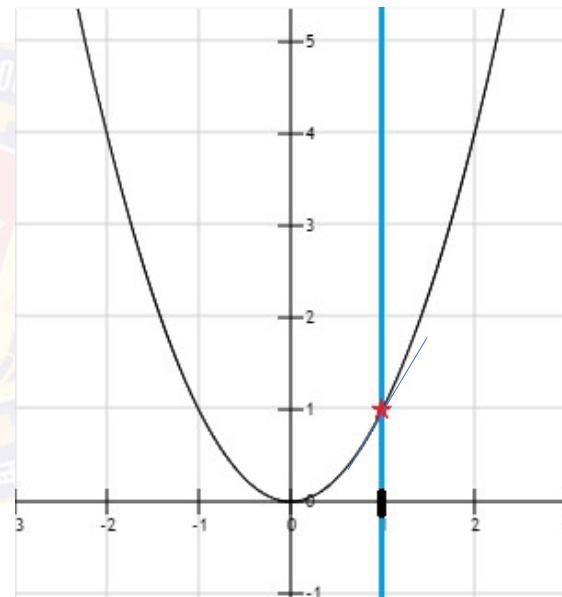
- Minimize x^2



Constrained Optimization -Equality Constraint

Minimize x^2

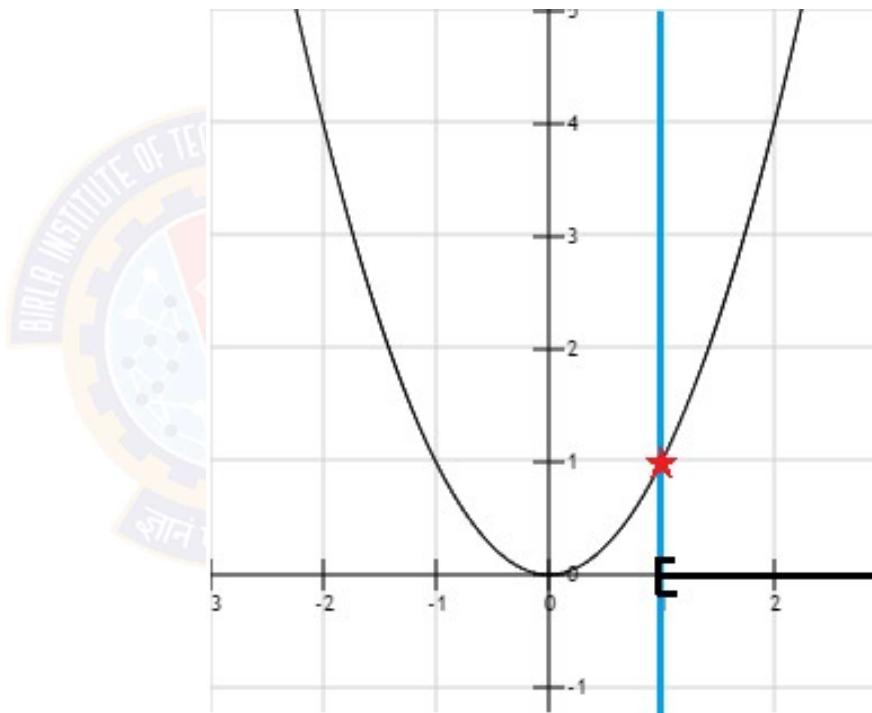
Subject to $x = 1$



Constrained Optimization -Inequality Constraint

Minimize x^2

Subject to $x \geq 1$



Constrained optimization

- We can also have mix equality and inequality constraints together.
- Only restriction is that if we use contradictory constraints, we can end up with a problem which does not have a feasible set

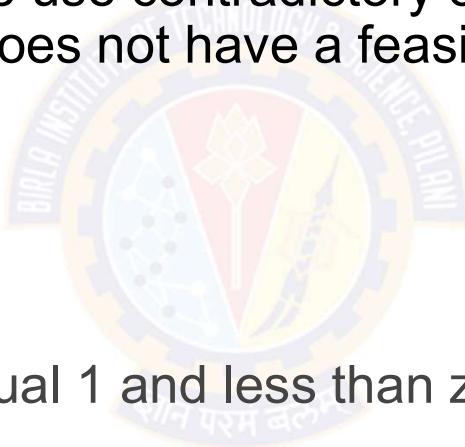
Minimize x^2

Subject to

$$x = 1$$

$$x < 0$$

Impossible for x to be equal 1 and less than zero at the same



Constrained optimization

- A solution is an assignment of values to variables.
- A feasible solution is an assignment of values to variables such that all the constraints are satisfied.
- The objective function value of a solution is obtained by evaluating the objective function at the given solution.
- An optimal solution (assuming minimization) is one whose objective function value is less than or equal to that of all other feasible solutions.

Unconstrained optimization

- For a univariate function $f(x)$ with first and second level derivatives, the aim is to locate x^* which maximizes / minimizes $f(x)$ without imposing any constraints
- The solution x^* , called stationary point, is found using first derivative of $f(x)$:

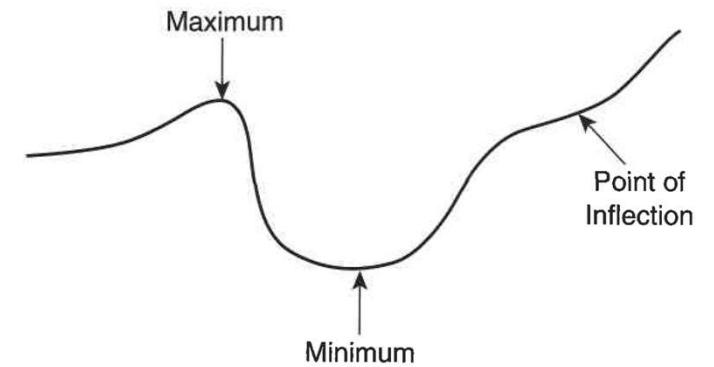
$$\frac{df}{dx} \Big|_{x=x^*} = 0$$

- $f(x^*)$ will take maximum or minimum value based on the second order derivative on $f(x)$ given as:

x^* is a maximum stationary point if $\frac{d^2f}{dx^2} < 0$ at $x = x^*$

x^* is a minimum stationary point if $\frac{d^2f}{dx^2} > 0$ at $x = x^*$.

x^* is a point of inflection when $\frac{d^2f}{dx^2} = 0$ at $x = x^*$



$$\mathbf{H}(\mathbf{x}) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1 \partial x_1} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_d} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_d} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_d \partial x_1} & \frac{\partial^2 f}{\partial x_d \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_d \partial x_d} \end{bmatrix}$$

- For a multi-variate function $f(x_1, x_2, \dots, x_d)$, stationary point $\mathbf{x}^* = [x_1^*, x_2^*, \dots, x_d^*]$ is found using the condition:

$$\frac{\partial f}{\partial x_i} \Big|_{x_i=x_i^*} = 0, \quad \forall i = 1, 2, \dots, d$$

Hessian matrix

- "**Hessian matrix**" of a multivariable function $f(x, y, z, \dots)$, organizes all second partial derivatives into a matrix:

$$\mathbf{H}f = \begin{bmatrix} \frac{\partial^2 f}{\partial \textcolor{teal}{x}^2} & \frac{\partial^2 f}{\partial \textcolor{teal}{x} \partial \textcolor{red}{y}} & \frac{\partial^2 f}{\partial \textcolor{teal}{x} \partial \textcolor{green}{z}} & \cdots \\ \frac{\partial^2 f}{\partial \textcolor{red}{y} \partial \textcolor{teal}{x}} & \frac{\partial^2 f}{\partial \textcolor{red}{y}^2} & \frac{\partial^2 f}{\partial \textcolor{red}{y} \partial \textcolor{green}{z}} & \cdots \\ \frac{\partial^2 f}{\partial \textcolor{green}{z} \partial \textcolor{teal}{x}} & \frac{\partial^2 f}{\partial \textcolor{green}{z} \partial \textcolor{red}{y}} & \frac{\partial^2 f}{\partial \textcolor{green}{z}^2} & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

Hessian matrix

$$f(x,y) = x^3 - 2xy - y^6$$

First order derivative

$$f_x(x,y) = \frac{\partial}{\partial x} (x^3 - 2xy - y^6) = 3x^2 - 2y$$

$$f_y(x,y) = \frac{\partial}{\partial y} (x^3 - 2xy - y^6) = -2x - 6y^5$$

Second order derivative

$$f_{xx}(x,y) = \frac{\partial}{\partial x} (3x^2 - 2y) = 6x$$

$$f_{xy}(x,y) = \frac{\partial}{\partial y} (3x^2 - 2y) = -2$$

$$f_{yx}(x,y) = \frac{\partial}{\partial x} (-2x - 6y^5) = -2$$

$$f_{yy}(x,y) = \frac{\partial}{\partial y} (-2x - 6y^5) = -30y^4$$

Hessian matrix

$$f(x,y) = x^3 - 2xy - y^6$$

The Hessian matrix in this case is a 2×2 matrix with these functions as entries:

$$\mathbf{H}f(x,y) = \begin{bmatrix} f_{xx}(x,y) & f_{yx}(x,y) \\ f_{xy}(x,y) & f_{yy}(x,y) \end{bmatrix} = \begin{bmatrix} 6x & -2 \\ -2 & -30y^4 \end{bmatrix}$$



BITS Pilani
Pilani | Dubai | Goa | Hyderabad

Primal, Dual, Quadratic Programming

Dr. Chetana Gavankar

In this segment

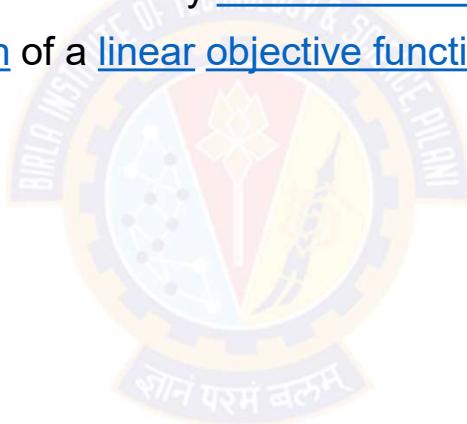
Primal, Dual, Quadratic Programming

- Linear Programming
 - Duality Theory
 - Motivation for duality
 - Duality Principle
 - Dual Problem
 - Strong Duality
 - Rules for Primal to Dual Problem conversion
- Non-linear Programming
 - Quadratic Programming



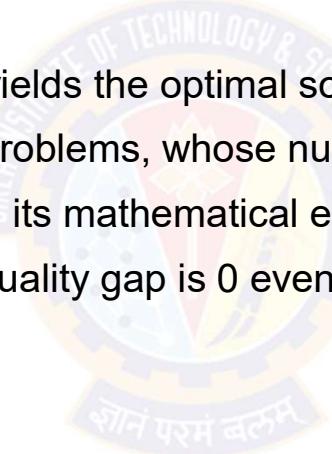
Linear programming(LP)

- LP, also called **linear optimization**
- Method to achieve the best outcome (such as maximum profit or lowest cost) in a mathematical model whose requirements are represented by linear relationships.
- It is a technique for the optimization of a linear objective function, subject to linear equality and linear inequality constraints.



Duality Theory

- Every LP problem (called the ‘Primal’) has associated with another problem called the ‘Dual’.
- The ‘Dual’ problem is an LP defined directly and systematically from the original (or Primal) LP model.
- The optimal solution of one problem yields the optimal solution to the other.
- Duality ease the calculations for the problems, whose number of variables is large.
- Conversion from primal to dual due to its mathematical elegance
- The dual is not only convex, but the duality gap is 0 even if the primal problem is not convex!



Motivation for duality

Try and get an upper estimate of Z without solving

Maximize $10X_1 + 9X_2$

Subject to

$$3X_1 + 3X_2 \leq 21 \quad \times \quad 4$$

$$4X_1 + 3X_2 \leq 24$$

$$X_1, X_2 \geq 0$$

Constraint 1 $\times 10$ gives $30X_1 + 30X_2 \leq 210$; Upper estimate = 210

Constraint 1 $\times 4$ gives $12X_1 + 12X_2 \leq 84$; Upper estimate = 84

Constraint 2 $\times 3$ gives $12X_1 + 9X_2 \leq 72$; Upper estimate = 72

Constraint 1 $\times 3$ + constraint 2 $\times 1$ gives $13X_1 + 12X_2 \leq 87$; Upper estimate = 87

Motivation for duality

Maximize $10X_1 + 9X_2$

Subject to

$$3X_1 + 3X_2 \leq 21$$

$$4X_1 + 3X_2 \leq 24$$

$$X_1, X_2 \geq 0$$

If I multiply the first constraint by a and the second constraint by b ($a, b \geq 0$) such that $3a + 4b \geq 10$ and $3a + 3b \geq 9$, then $21a + 24b$ is an upper estimate of Z .

We want to minimize the upper estimate and we solve another LP

Minimize $21a + 24b$

Subject to $3a + 4b \geq 10$;
 $3a + 3b \geq 9$, $a, b \geq 0$.

Replace a, b by Y_1 and Y_2

Duality or Duality principle

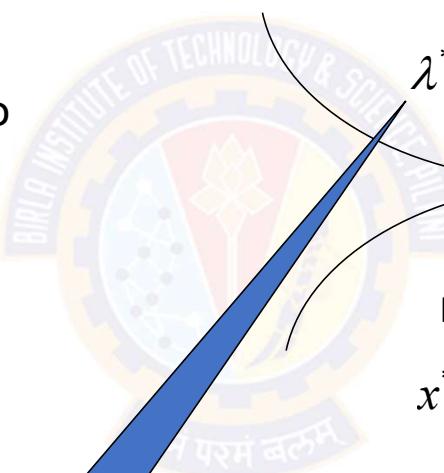
- **Duality** means that optimization problems may be viewed from two perspectives,
 - **Primal problem or the Dual problem**
 - **Solution to the dual problem provides a upper bound to the solution of the primal (maximization) problem.**

$$\begin{aligned} & \text{Max } f(x) \\ & \text{s.t. } g(x) \leq 0 \end{aligned}$$

- **Strong Duality Theorem:** If strong duality theorem holds, then the primal and dual optimal objective values are equal.
- Solving the dual problem is simpler than solving the primal problem.

Dual Problem

- Using dual problem
 - Constrained optimization → unconstrained optimization
- Need to change **maximization** to **minimization**
- strong duality when the original optimization problem is convex/concave



Dual Problem

$$\lambda^* = \arg \min_{\lambda} l(\lambda)$$

Primal Problem

$$x^* = \arg \max_x f(x)$$

subject to $g(x) = c$

$x^* = \lambda^*$
When
convex /
concave

Strong Duality

Optimality Criterion theorem

If the primal and dual have feasible solution with the same value of the objective function, then both are optimal to the primal and dual respectively.

$$\begin{aligned} & \text{Maximize } 10X_1 + 9X_2 \\ & \text{Subject to} \\ & 3X_1 + 3X_2 \leq 21 \\ & 4X_1 + 3X_2 \leq 24 \\ & X_1, X_2 \geq 0 \end{aligned}$$

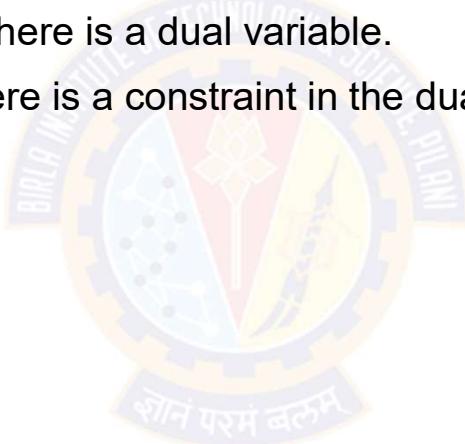
$$\begin{aligned} & \text{Minimize } 21Y_1 + 24Y_2 \\ & \text{Subject to} \\ & 3Y_1 + 4Y_2 \geq 10 \\ & 3Y_1 + 3Y_2 \geq 9 \\ & Y_1, Y_2 \geq 0 \end{aligned}$$

$$\begin{aligned} (0,0) Z &= 0 \\ (1,1) Z &= 19 \\ (3, 1) Z &= 39 \\ (6, 0) Z &= 60 \\ (2, 5) Z &= 65 \\ (3, 4) Z &= 66 \quad \checkmark \end{aligned}$$

$$\begin{aligned} (10,10) W &= 450 \\ (6, 5) W &= 246 \\ (5, 4) W &= 201 \\ (3, 3) W &= 135 \\ (2, 2) W &= 90 \\ (2, 1) W &= 66 \quad \checkmark \end{aligned}$$

Rules for converting Primal to Dual

- If the Primal is to maximize, the dual is to minimize.
- If the Primal is to minimize, the dual is to maximize.
- For every constraint in the primal, there is a dual variable.
- For every variable in the primal, there is a constraint in the dual.



Primal to Dual conversion table

Primal	Dual
Max	Min
Variables	Constraints
Constraints	Variables
RHS	Objective Function
Objective Function	RHS
A	A^T



Primal

$$\text{Min. } Z = 10x_1 + 15x_2$$

Subject to constraints:

$$5x_1 + 7x_2 \geq 80$$

$$6x_1 + 11x_2 \geq 100$$

$$x_1, x_2 \geq 0$$

Dual

$$\text{Max. } Z' = 80y_1 + 100y_2$$

Subject to constraints:

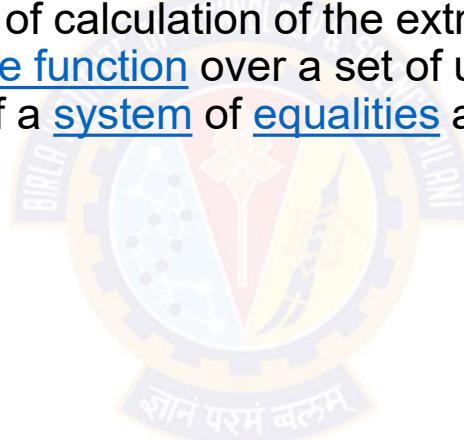
$$5y_1 + 6y_2 \leq 10$$

$$7y_1 + 11y_2 \leq 15$$

$$y_1, y_2 \geq 0$$

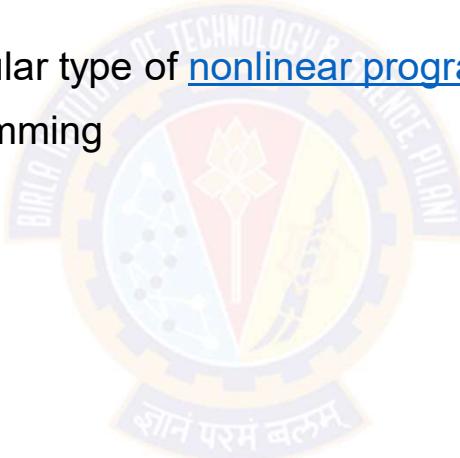
Non-linear Programming

- Process of solving an optimization problem where some of the constraints or the objective function are nonlinear. $x^3 - 2x + 1 = 0$
- An optimization problem is one of calculation of the extrema (maxima, minima or stationary points) of an objective function over a set of unknown real variables and conditional to the satisfaction of a system of equalities and inequalities



Quadratic Programming

- Problem of optimizing (minimizing or maximizing) a quadratic function (one or more variables in which the highest-degree term is of the second degree) of several variables subject to linear constraints on these variables.
- Quadratic programming is a particular type of nonlinear programming.
- Simplest form of non-linear programming





BITS Pilani
Pilani | Dubai | Goa | Hyderabad

Optimization Foundations - Basics

Dr. Chetana Gavankar

In this segment

Optimization Foundations - Basics

- Lagrange Multiplier
- KKT conditions



Lagrange Multipliers

How do we find the solution to an optimization problem with constraints?

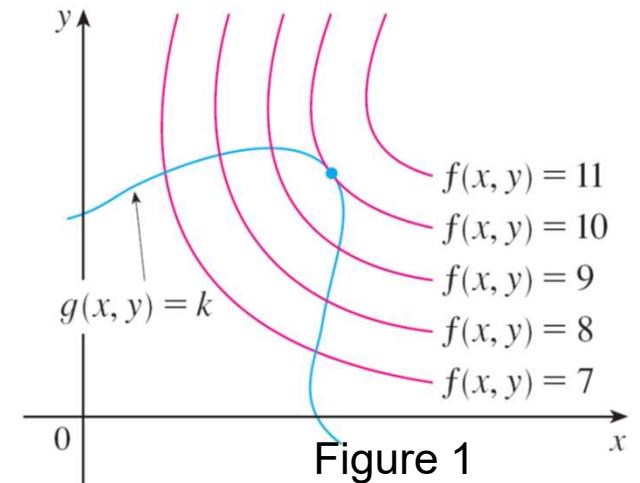
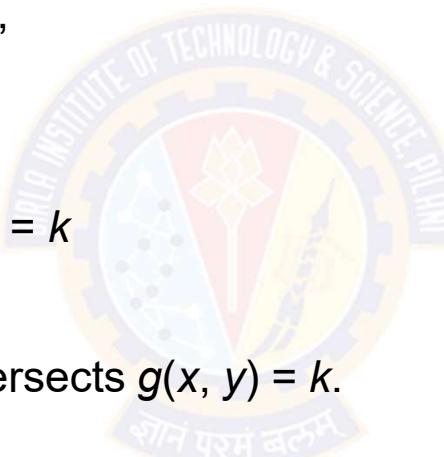
- Constrained maximization (minimization) problem is rewritten as a Lagrange function whose optimal point is a saddle point, i.e. a global maximum (minimum)
- *Lagrange function use Lagrange multipliers as a strategy for finding the local maxima and minima of a function subject to equality constraints*
- Lagrange multipliers **only works with equality constraints**

Lagrange Multipliers

- Figure 1 shows this curve together with several level curves of f .

- These have the equations $f(x, y) = c$, where $c = 7, 8, 9, 10, 11$.

- To maximize $f(x, y)$ subject to $g(x, y) = k$ is to find the largest value of c such that the level curve $f(x, y) = c$ intersects $g(x, y) = k$.



- It appears from Figure 1 that this happens when these curves just touch each other, that is, when they have a common tangent line. (Otherwise, the value of c could be increased further.)

Example of Lagrange Multiplier

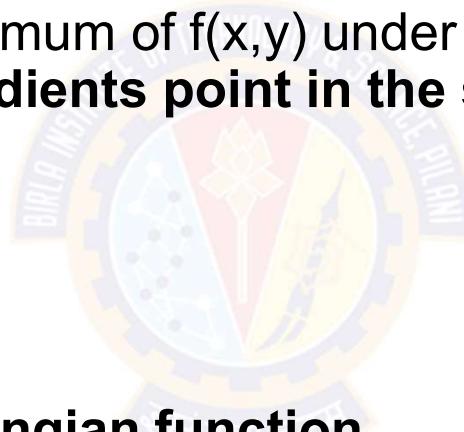
Minimize $f(x,y) = x^2 + y^2$

Subject to $g(x, y) = x + y - 1 = 0$

Lagrange found that Minimum of $f(x,y)$ under constraint $g(x, y)$ is obtained **when their gradients point in the same direction.**

Mathematically,

$$\nabla f(x,y) = \lambda \nabla g(x,y)$$

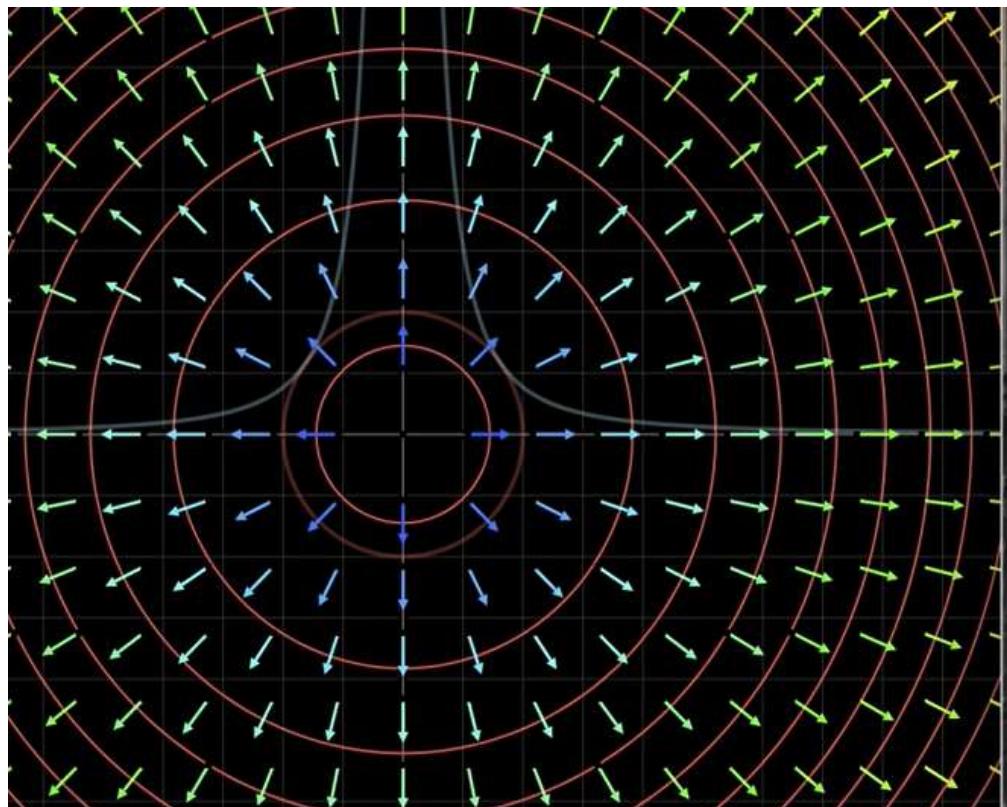


We introduce the Lagrangian function

$$L(x,y,\lambda) = f(x,y) - \lambda g(x,y)$$



Example of Lagrange Multiplier



Maximize $f(x, y) = x^2y$
on the set $x^2 + y^2 = 1$

$f(x_1, y_1) = c$ Lagrange
 $\nabla f(x_m, y_m) = \lambda \nabla g(x_m, y_m)$
 $g(x, y) = x^2 + y^2$

[Constrained optimization introduction \(video\) | Khan Academy](#)

Example of Lagrange Multiplier

- Introduce a Lagrange multiplier λ for constraint
- Construct the Lagrangian

$$\max_{x,y} (xy); \quad \text{subject to } x + y = 6$$

$$L(x, y) = xy - \lambda(x + y - 6)$$

- Stationary points

$$\begin{aligned}\frac{\partial L(x, y)}{\partial \lambda} &= x + y - 6 = 0 \\ \left. \begin{aligned}\frac{\partial L(x, y)}{\partial x} &= y - \lambda = 0 \\ \frac{\partial L(x, y)}{\partial y} &= x - \lambda = 0\end{aligned} \right\} &\Rightarrow x = y = \lambda \\ \Rightarrow x = y &= 3\end{aligned}$$

x and y values remain same even if you take $+\lambda$ or $-\lambda$ for equality constraint

$$\begin{aligned}2x &= 6 \\ x &= y = 3 \\ \lambda &= 3\end{aligned}$$

Karush–Kuhn–Tucker (KKT) theorem

- KKT approach to nonlinear programming (quadratic) generalizes the method of [Lagrange multipliers](#), which allows only equality constraints.
- KKT allows inequality constraints



Karush-Kuhn-Tucker (KKT) conditions

- Start with

$\min f(x)$ subject to

$$g_i(x) = 0 \text{ and } h_j(x) \geq 0 \text{ for all } i, j$$

- Make the Lagrangian function

$$\mathcal{L} = f(x) - \sum_i \lambda_i g_i(x) - \sum_j \mu_j h_j(x)$$

- Take gradient and set to 0 – but other conditions also.

KKT conditions

- Make the Lagrangian function for minimization:

$$\mathcal{L} = f(x) - \sum_i \lambda_i g_i(x) - \sum_j \mu_j h_j(x)$$

- Necessary conditions to have a minimum are

$$\nabla_x \mathcal{L}(x^*, \lambda^*, \mu^*) = 0$$

$$g_i(x^*) = 0 \text{ for all } i$$

$$h_j(x^*) \geq 0 \text{ for all } j$$

$$\mu_j \geq 0 \text{ for all } j$$

$$\mu_j^* h_j(x^*) = 0 \text{ for all } j$$



BITS Pilani
Pilani | Dubai | Goa | Hyderabad

Maximum Margin Classifier

Dr. Chetana Gavankar

In this segment

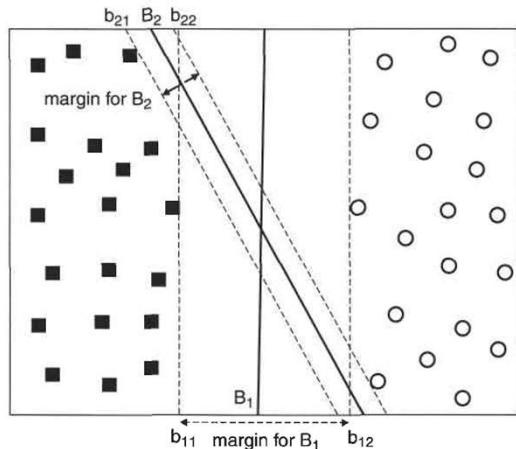
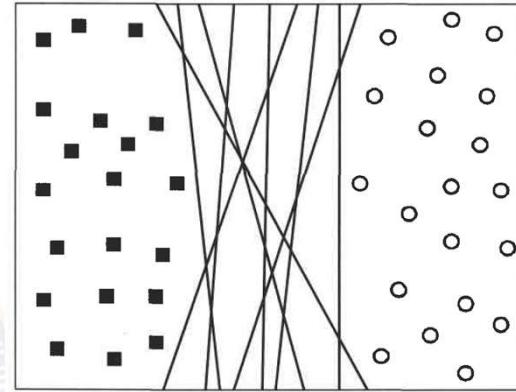
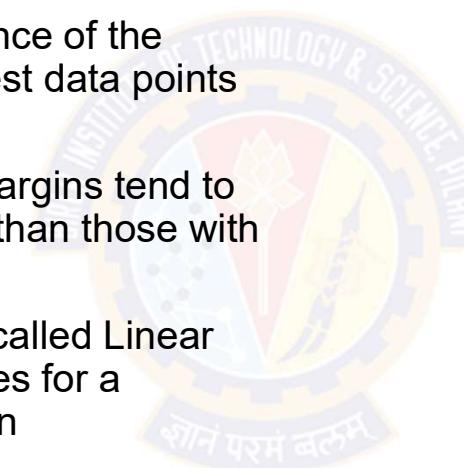
Understanding the spirit and significance of maximum margin classifier



Maximum Margin Classifier

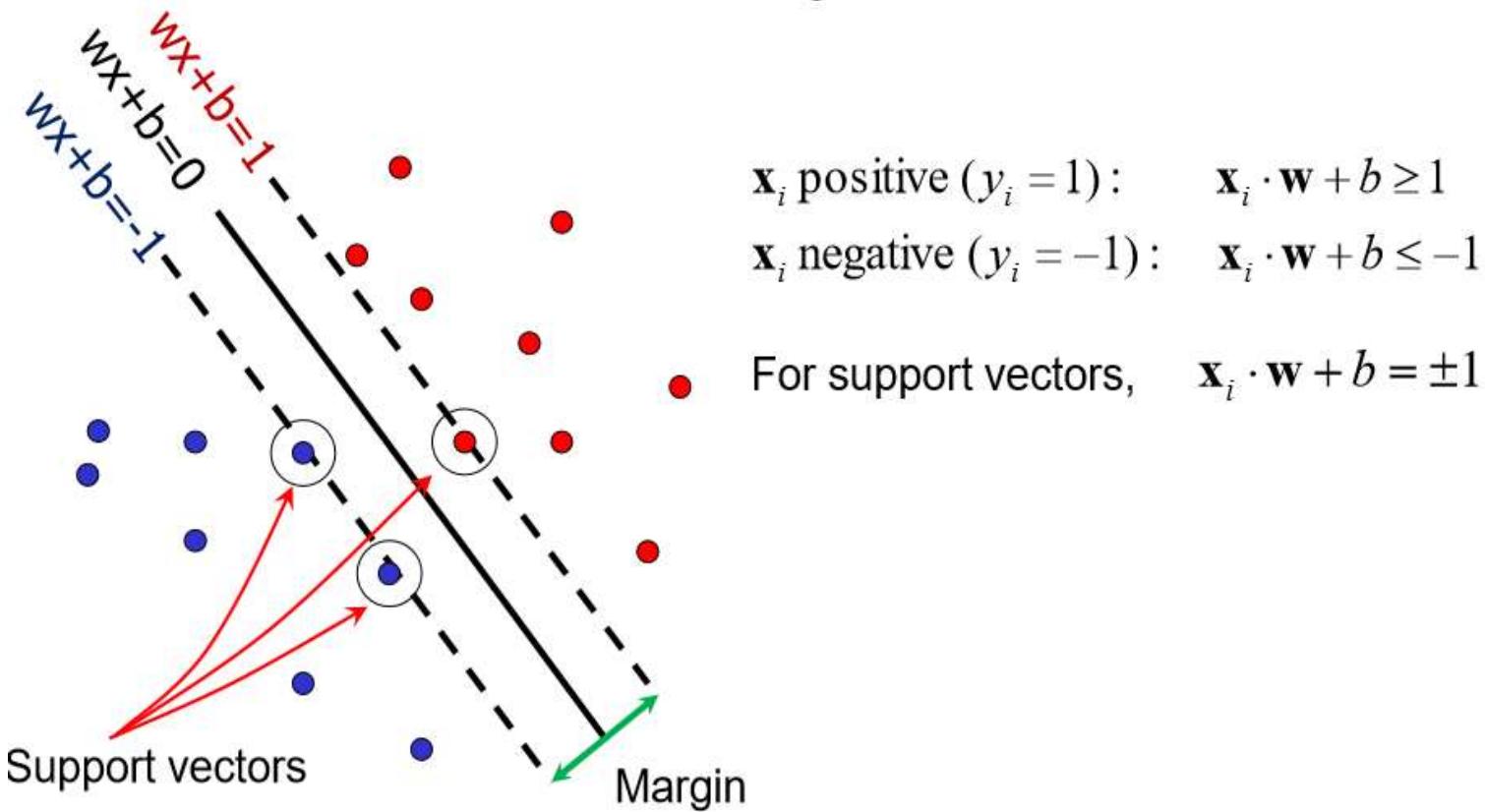
Overview

- In (linear) classifiers, decision boundary is the hyperplane that separates the classes
- Margin for a classifier is the distance of the decision boundary from the nearest data points on either side
- Decision boundaries with large margins tend to have better generalization errors than those with small margins.
- Maximum margin classifier (also called Linear SVM) is the classifier that searches for a hyperplane with the largest margin



Maximum Margin Classifier

- Want line that maximizes the margin.

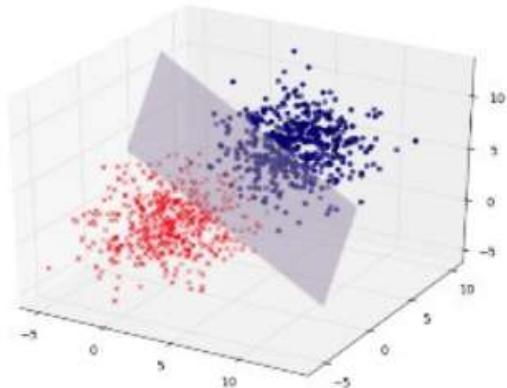


Example

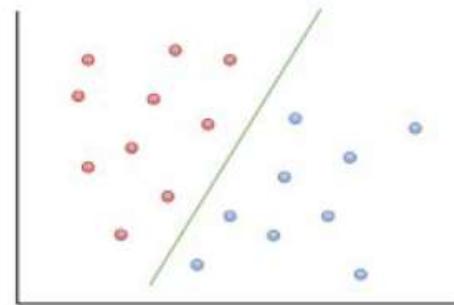
$$\mathbf{w}^T \mathbf{x} = 0$$

$$y = ax + b$$

Hyperplane



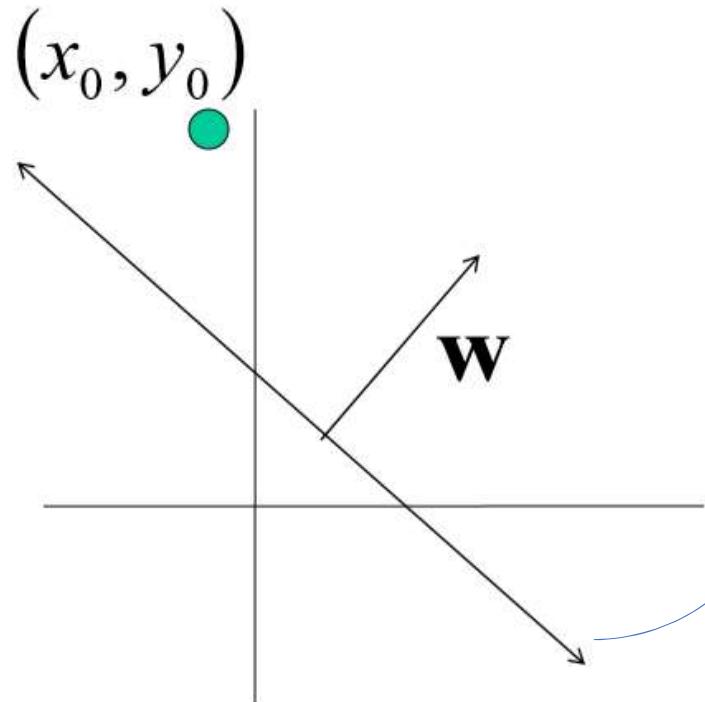
Line



Support Vectors

- Geometric description of SVM is that the max-margin hyperplane is completely determined by those points that lie nearest to it.
- Points that lie on this margin are the support vectors.
- The points of our data set which if removed, would alter the position of the dividing hyperplane

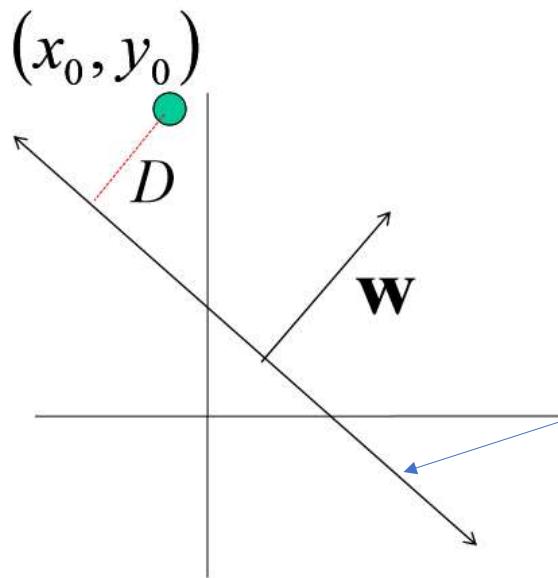
Line with 2 features: R2



Let $\mathbf{w} = \begin{bmatrix} a \\ c \end{bmatrix}$ $\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix}$

$$\begin{array}{c} ax + cy + b = 0 \\ \Updownarrow \\ \mathbf{w} \cdot \mathbf{x} + b = 0 \end{array}$$

Line with 2 features: R2



Let $\mathbf{w} = \begin{bmatrix} a \\ c \end{bmatrix}$ $\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix}$

$$ax + cy + b = 0$$

$$\mathbf{w} \cdot \mathbf{x} + b = 0$$

$$D = \frac{|ax_0 + cy_0 + b|}{\sqrt{a^2 + c^2}} = \frac{|\mathbf{w}^\top \mathbf{x} + b|}{\|\mathbf{w}\|}$$

} distance from
point to line

Weight vector is perpendicular to the hyperplane

Consider the points x_a and x_b , which lie on the decision boundary.

This gives us two equations:

$$w^T x_a + b = 0$$

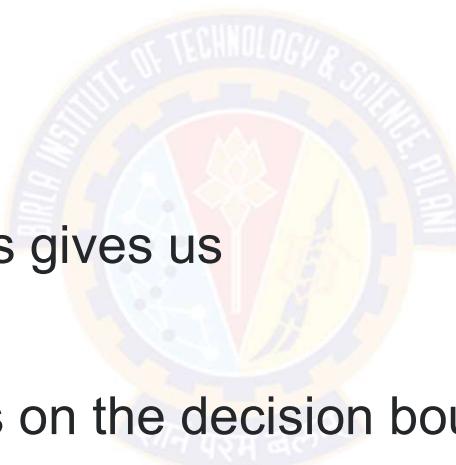
$$w^T x_b + b = 0$$

Subtracting these two equations gives us

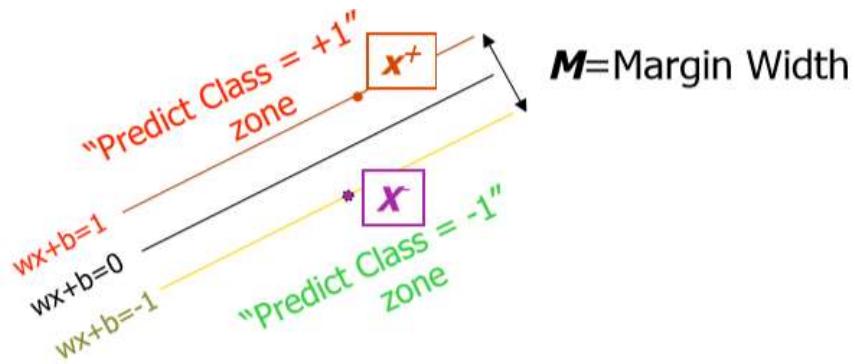
$$w^T.(x_a - x_b) = 0$$

Note that the vector $x_a - x_b$ lies on the decision boundary, and it is directed from x_b to x_a .

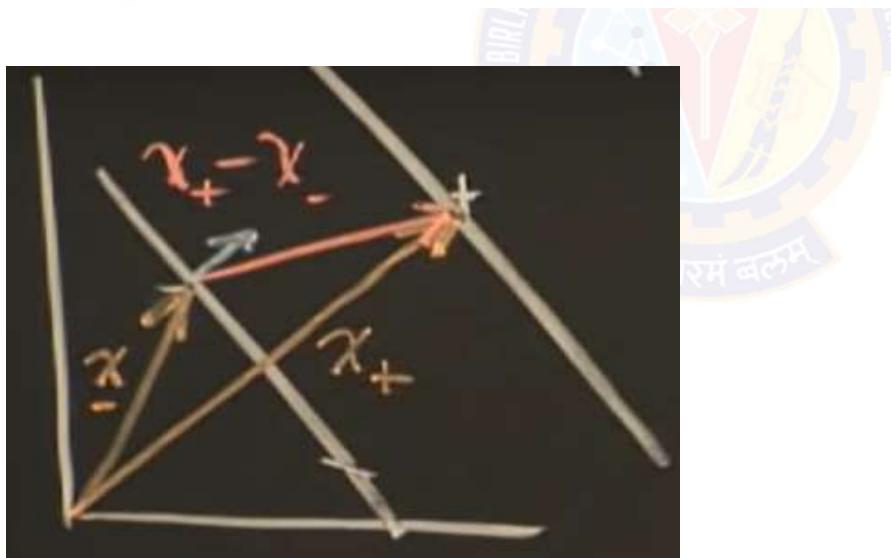
Since the dot product $w^T.(x_a - x_b)$ is zero, w^T must be orthogonal to $x_a - x_b$ and in turn, to the decision boundary.



Linear SVM Mathematically



$$\begin{aligned} w \cdot x^+ + b &= +1 \\ w \cdot x^- + b &= -1 \end{aligned}$$



Margin width

$$= x^+ - x^- \cdot \frac{w}{||w||}$$

$$= \frac{w \cdot x^+ - w \cdot x^-}{||w||}$$

$$= (1-b) - (-1-b) / ||w||$$

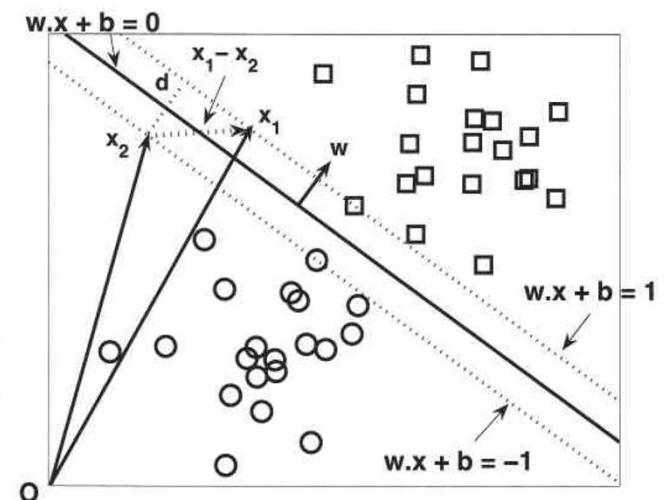
$$= \frac{2}{||w||}$$

Maximum Margin Classifier

Summary

- If the decision boundary is given by: $w \cdot x + b = 0$
- It can be proven that the direction of w is perpendicular to the boundary
- Margin of the classifier is the distance between the points on either side nearest to the boundary.
- Margin is calculated as : $d = \frac{2}{\|w\|}$
- For maximum margin classifier, we need to maximize the margin, by minimizing the objective function

$$f(w) = \frac{\|w\|^2}{2}$$



$$w \cdot x_i + b \geq 1 \text{ if } y_i = 1,$$

$$w \cdot x_i + b \leq -1 \text{ if } y_i = -1.$$



BITS Pilani
Pilani | Dubai | Goa | Hyderabad

SVM - Optimization

Dr. Chetana Gavankar

In this segment

SVM Optimization

- Converting the constrained optimization problem into unconstrained using Lagrange multipliers
- Dual of the optimization problem
- Appreciation of sparse kernel machine and support vectors in the solution of the optimization problem



Solving the Optimization Problem

1. Maximize margin $2/\|\mathbf{w}\|$
2. Correctly classify all training data points:

$$\mathbf{x}_i \text{ positive } (y_i = 1) : \quad \mathbf{x}_i \cdot \mathbf{w} + b \geq 1$$

$$\mathbf{x}_i \text{ negative } (y_i = -1) : \quad \mathbf{x}_i \cdot \mathbf{w} + b \leq -1$$

Quadratic optimization problem:

Find \mathbf{w} and b such that

$\Phi(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2$ is minimized;

and for all $\{(\mathbf{x}_i, y_i)\}$: $y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1$

$$y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1$$

$$+1(\mathbf{w}^T \mathbf{x}_i + b) \geq 1$$

$$-1(\mathbf{w}^T \mathbf{x}_i + b) \leq 1$$

same as $(\mathbf{w}^T \mathbf{x}_i + b) \geq 1$

SVM - Non-overlapping

Optimization problem in Non-overlapping class

- For non-overlapping classes, the learning of SVM is defined as the constrained optimization problem

$$\min_{\mathbf{w}} \frac{\|\mathbf{w}\|^2}{2}$$

$$\text{subject to } y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1, \quad i = 1, 2, \dots, N.$$

- Here the objective function is minimised to maximize the margin
- Objective function is quadratic and the constraints are linear in the parameters w and b

Constrained into unconstrained optimization problem

- The solution involves constructing a *unconstrained problem* where a *Lagrange multiplier* α_i is associated with every constraint in the primary problem:

$$L(w, b, \alpha_i) = \frac{1}{2} \|w\|^2 - \sum \alpha_i [y_i (w^T x_i + b) - 1]$$

- Taking partial derivative with respect to w , $\frac{\partial L}{\partial w} = 0$
 - $w - \sum \alpha_i y_i x_i = 0$
 - $w = \sum \alpha_i y_i x_i$
- Taking partial derivative with respect to b , $\frac{\partial L}{\partial b} = 0$
 - $-\sum \alpha_i y_i = 0$
 - $\sum \alpha_i y_i = 0$

Lagrange Multipliers

$$L(w, b, \alpha_i) = \frac{1}{2} \|w\|^2 - \sum \alpha_i [y_i (w^T x_i + b) - 1]$$

- Expanding above equation:

$$L(w, b, \alpha_i) = \frac{1}{2} w^T w - \sum \alpha_i y_i w^T x_i + \sum \alpha_i y_i b + \sum \alpha_i$$

- Substituting $w = \sum \alpha_i y_i x_i$ and $\sum \alpha_i y_i = 0$ in above equation

$$L(w, b, \alpha_i) = \frac{1}{2} (\sum_i \alpha_i y_i x_i) (\sum_j \alpha_j y_j x_j) - (\sum_i \alpha_i y_i x_i) (\sum_j \alpha_j y_j x_j) + \sum \alpha_i$$

$$L(w, b, \alpha_i) = \sum \alpha_i - \frac{1}{2} (\sum_i \alpha_i y_i x_i) (\sum_j \alpha_j y_j x_j)$$

$$L(w, b, \alpha_i) = \sum \alpha_i - \frac{1}{2} (\sum_i \sum_j \alpha_i \alpha_j y_i y_j x_i \cdot x_j)$$

Dual of the optimization problem

Find \mathbf{w} and b such that

$\Phi(\mathbf{w}) = \frac{1}{2} \mathbf{w}^T \mathbf{w} = \frac{1}{2} \|\mathbf{w}\|^2$ is minimized;
and for all $\{(\mathbf{x}_i, y_i)\}$: $y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1$

$$L(w, b, \alpha_i) = \frac{1}{2} \|\mathbf{w}\|^2 - \sum \alpha_i [y_i (\mathbf{w}^T \mathbf{x}_i + b) - 1]$$

Find $\alpha_1 \dots \alpha_N$ such that

$Q(\alpha) = \sum \alpha_i - \frac{1}{2} \left(\sum_i \sum_j \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \right)$ is maximized and

- (1) $\sum \alpha_i y_i = 0$
- (2) $\alpha_i \geq 0$ for all α_i

KKT conditions

- Make the Lagrangian function for minimization:

$$\mathcal{L} = f(x) - \sum_i \lambda_i g_i(x) - \sum_j \mu_j h_j(x)$$

- Necessary conditions to have a minimum are

$$\nabla_x \mathcal{L}(x^*, \lambda^*, \mu^*) = 0$$

$$g_i(x^*) = 0 \text{ for all } i$$

$$h_j(x^*) \geq 0 \text{ for all } j$$

$$\mu_j \geq 0 \text{ for all } j$$

$$\mu_j^* h_j(x^*) = 0 \text{ for all } j$$

Appreciation of sparse kernel and support vectors

Using KKT conditions :

$$\alpha_i [y_i (\mathbf{w}^T \mathbf{x}_i + b) - 1] = 0$$

For this condition to be satisfied

EITHER

$$\alpha_i = 0 \text{ and } y_i (\mathbf{w}^T \mathbf{x}_i + b) - 1 > 0$$

OR

$$y_i (\mathbf{w}^T \mathbf{x}_i + b) - 1 = 0 \text{ and } \alpha_i > 0$$

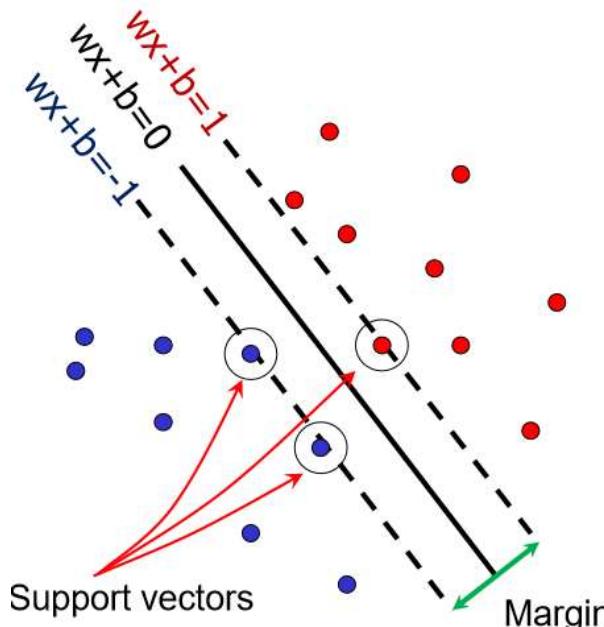
For support vectors:

$$y_i (\mathbf{w}^T \mathbf{x}_i + b) - 1 = 0$$

For all points other than support vectors:

$$\alpha_i = 0$$

- Want line that maximizes the margin.



$$\mathbf{x}_i \text{ positive } (y_i = 1) : \quad \mathbf{x}_i \cdot \mathbf{w} + b \geq 1$$

$$\mathbf{x}_i \text{ negative } (y_i = -1) : \quad \mathbf{x}_i \cdot \mathbf{w} + b \leq -1$$

$$\text{For support vectors, } \mathbf{x}_i \cdot \mathbf{w} + b = \pm 1$$

$$L(\mathbf{w}, b, \alpha_i) = \frac{1}{2} \|\mathbf{w}\|^2 - \sum \alpha_i [y_i (\mathbf{w}^T \mathbf{x}_i + b) - 1]$$

Solving the Optimization Problem

- Solution: $\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$

Learned
weight

Support
vector

Solving the Optimization Problem

- Solution: $\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$
 $b = y_i - \mathbf{w} \cdot \mathbf{x}_i$ (for any support vector)
- Classification function:

$$\begin{aligned}f(x) &= \text{sign}(\mathbf{w} \cdot \mathbf{x} + b) \\&= \text{sign}\left(\sum_i \alpha_i y_i \mathbf{x}_i \cdot \mathbf{x} + b\right)\end{aligned}$$

If $f(x) < 0$, classify as negative, otherwise classify as positive.

- Notice that it relies on an *inner product* between the test point \mathbf{x} and the support vectors \mathbf{x}_i
- (Solving the optimization problem also involves computing the inner products $\mathbf{x}_i \cdot \mathbf{x}_j$ between all pairs of training points)

SVM optimization example

Let 2 points for classification be $x_1=(2,1)$ and $x_2=(1,2)$
With class labels $y_1=-1$ and $y_2=1$

$$\begin{aligned}\max_{\alpha} L_D &= \sum_{i=1}^L \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle \\&= \alpha_1 + \alpha_2 - \frac{1}{2} \left(\alpha_1 \alpha_1 \cdot 1 \cdot 1 \cdot \langle \begin{pmatrix} 2 \\ 1 \end{pmatrix}, \begin{pmatrix} 2 \\ 1 \end{pmatrix} \rangle + \right. \\&\quad + 2 \cdot \alpha_1 \alpha_2 \cdot 1 \cdot (-1) \cdot \langle \begin{pmatrix} 1 \\ 2 \end{pmatrix}, \begin{pmatrix} 2 \\ 1 \end{pmatrix} \rangle + \\&\quad \left. + \alpha_2 \alpha_2 \cdot (-1) \cdot (-1) \cdot \langle \begin{pmatrix} 1 \\ 2 \end{pmatrix}, \begin{pmatrix} 1 \\ 2 \end{pmatrix} \rangle \right) \\&= \alpha_1 + \alpha_2 - \frac{1}{2} (5\alpha_1^2 - 8\alpha_1\alpha_2 + 5\alpha_2^2)\end{aligned}$$

subject to $\alpha_1 y_1 + \alpha_2 y_2 = 0$
 $\alpha_1 \geq 0, \alpha_2 \geq 0$

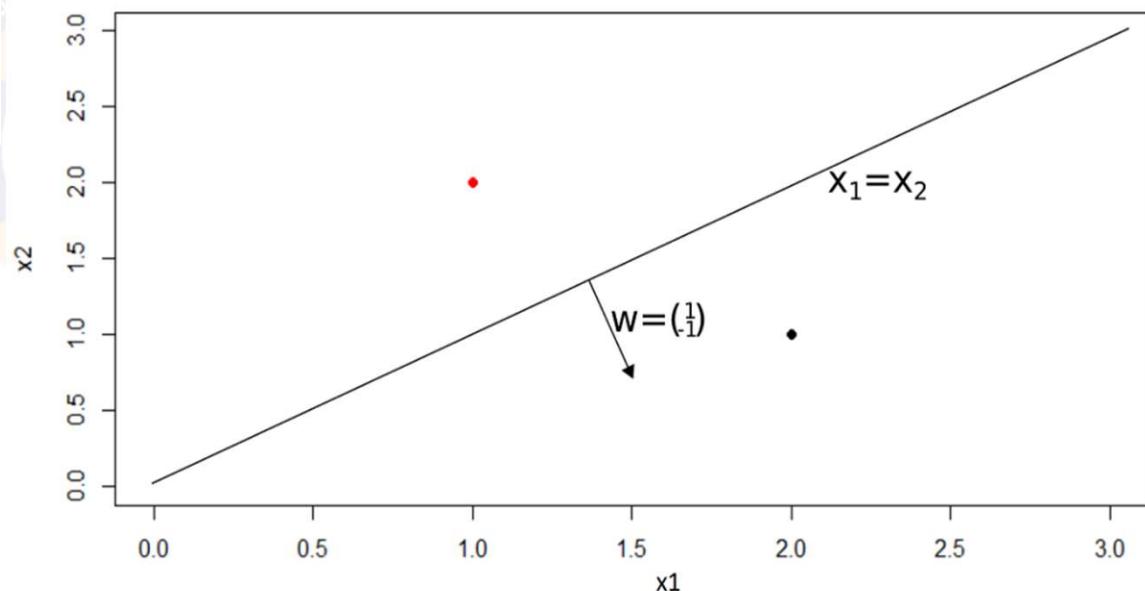
Solve Quadratic Programming Problem using wolfram or any library,
we get $\alpha_1=1$ and $\alpha_2=1$

SVM optimization example

Solve to get w:

$$\begin{aligned} w &= \sum_{i=1}^L \alpha_i y_i x_i = 1 \cdot 1 \cdot \begin{pmatrix} 2 \\ 1 \end{pmatrix} + 1 \cdot (-1) \cdot \begin{pmatrix} 1 \\ 2 \end{pmatrix} \\ &= \begin{pmatrix} 2 \\ 1 \end{pmatrix} - \begin{pmatrix} 1 \\ 2 \end{pmatrix} \\ &= \begin{pmatrix} 1 \\ -1 \end{pmatrix} \end{aligned}$$

Hyperplane can be represented by
 $wx+b=0$
 $\Rightarrow x_1-x_2=0$
 $\Rightarrow x_1=x_2$





BITS Pilani
Pilani | Dubai | Goa | Hyderabad

Soft Margin Classification

Dr. Chetana Gavankar

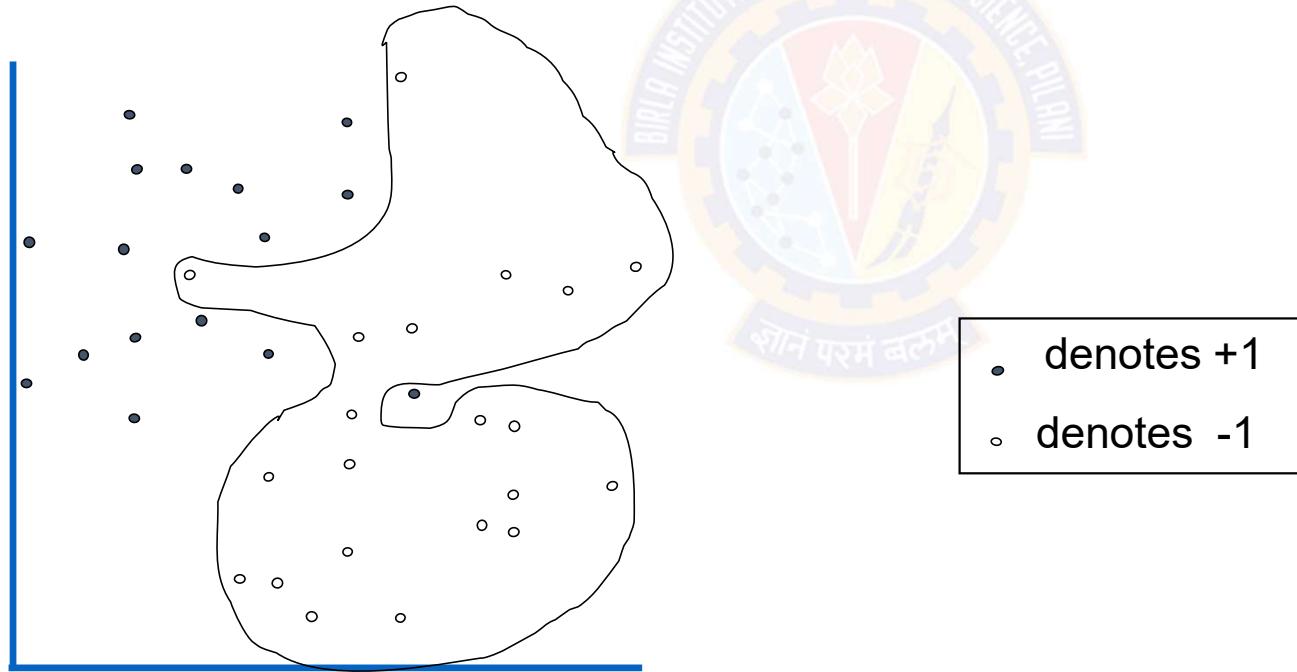
In this segment

- Dataset with noise
- Soft Margin Classification
- C parameter to avoid overfitting
- Effect of Margin versus Misclassification cost



Dataset with noise

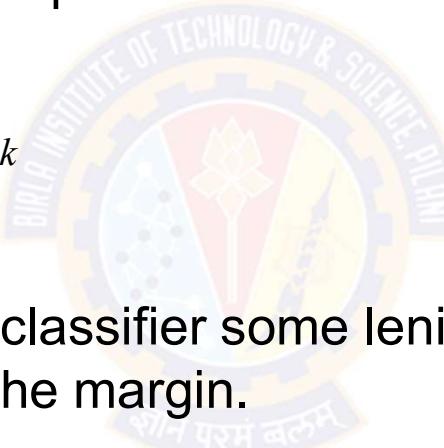
- Hard Margin: So far we require all data points be classified correctly
 - No training error
- What if the training set is noisy?



Soft Margin Classification

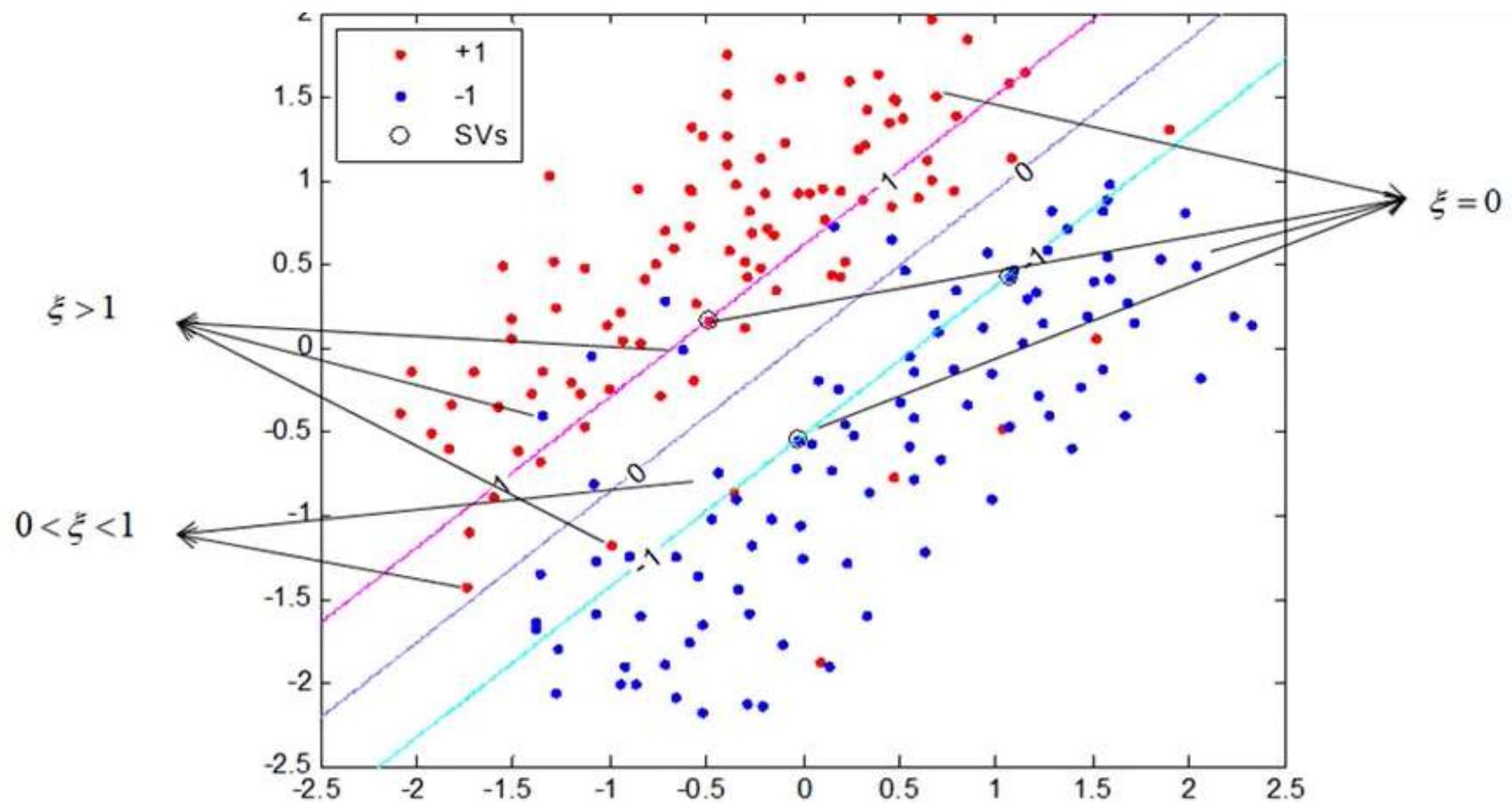
- Slack variables ξ_i can be added to allow misclassification of difficult or noisy examples.
- What should our quadratic optimization criterion be?

$$\text{Minimize } \frac{1}{2} \mathbf{w} \cdot \mathbf{w} + C \sum_{k=1}^R \xi_k$$



- Slack variable as giving the classifier some leniency when it comes to moving around points near the margin.
- Parameter C can be viewed as a way to control overfitting.
- When C is large, larger slacks penalize the objective function of SVM's more than when C is small.

Soft Margin Classification



Soft Margin Classification

$$\min_{\mathbf{w}} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^N \xi_i$$

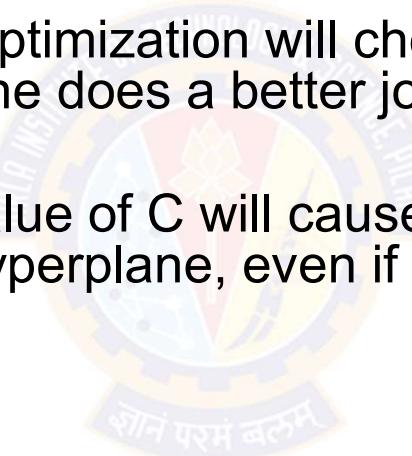
The \mathbf{w} that minimizes... Maximize margin Minimize misclassification

Misclassification cost # data samples Slack variable

subject to $y_i \mathbf{w}^T \mathbf{x}_i \geq 1 - \xi_i,$
 $\xi_i \geq 0, \quad \forall i = 1, \dots, N$

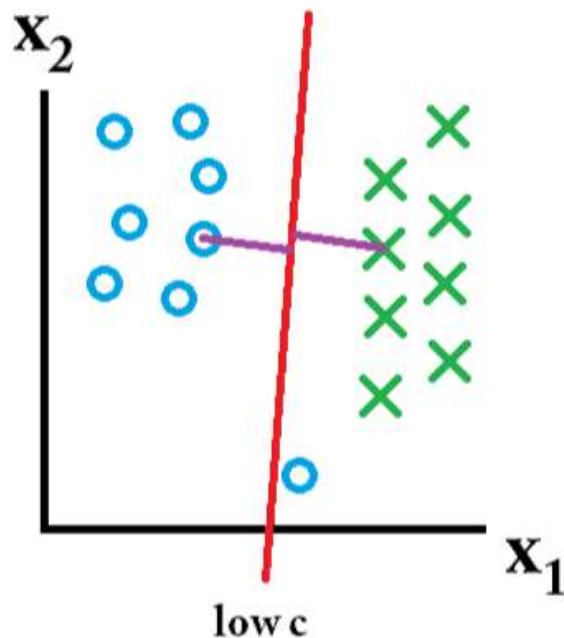
Value of C parameter

- C parameter tells the SVM optimization how much you want to avoid misclassifying each training example.
- For large values of C, the optimization will choose a smaller-margin hyperplane if that hyperplane does a better job of getting all the training points classified correctly.
- Conversely, a very small value of C will cause the optimizer to look for a larger-margin separating hyperplane, even if that hyperplane misclassifies more points.

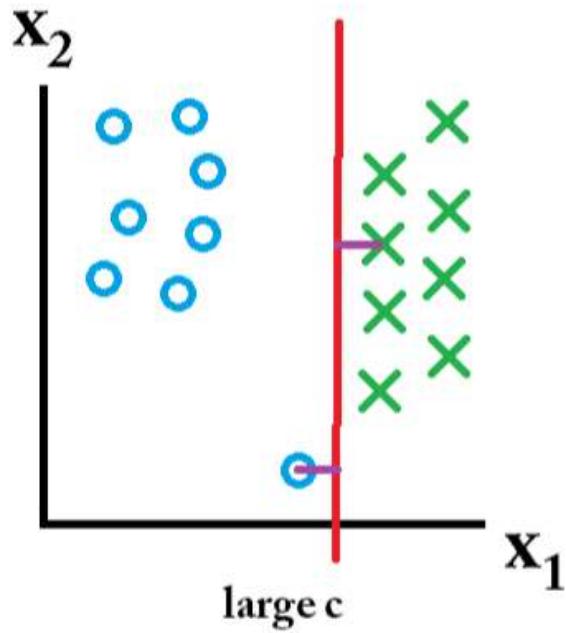


Effect of Margin size v/s misclassification cost

Training set



Misclassification ok, want large margin



overfitting
Misclassification not ok



BITS Pilani
Pilani | Dubai | Goa | Hyderabad

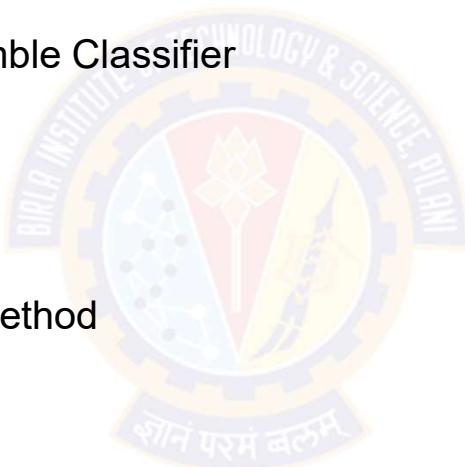
Ensemble Methods - Introduction

Dr. Chetana Gavankar

In this segment

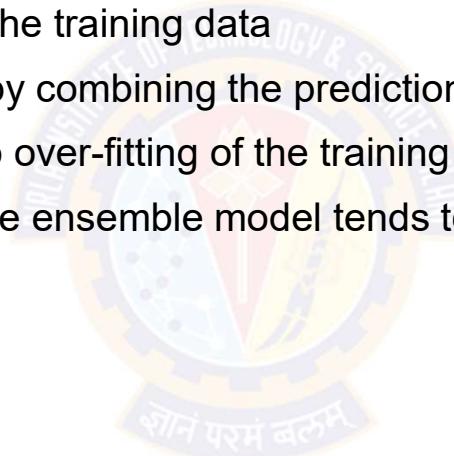
Ensemble Methods - Introduction

- Ensemble Methods
- Rationale for Ensemble Methods
- Methods for constructing an Ensemble Classifier
 - By training set
 - By input features
 - By class labels
 - By learning algorithm
- General procedure for ensemble method



Ensemble Methods

- **Ensemble methods** use multiple learning algorithms to obtain better [predictive performance](#) than could be obtained from any of the constituent learning algorithms alone
- Construct a set of classifiers from the training data
- Predict class label of test records by combining the predictions made by multiple classifiers
- Tend to reduce problems related to over-fitting of the training data.
- By combining individual models, the ensemble model tends to be more flexible (less bias) and less data-sensitive(less variance).



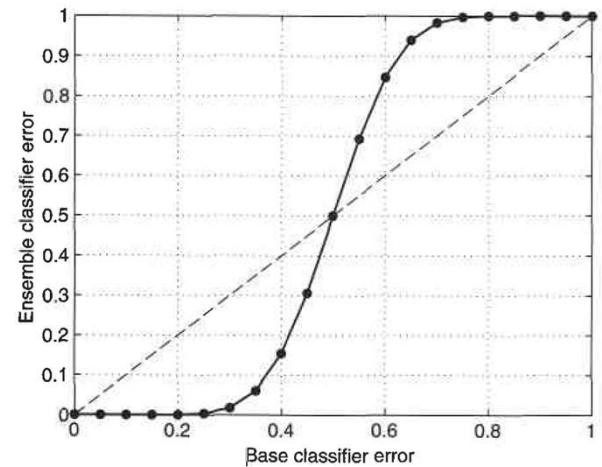
When does Ensemble work?

- Ensemble classifier performs better than the base classifiers when each classifier error is smaller than 0.5
- Necessary conditions for an ensemble classifier to perform better than a single classifier:
 - Base classifiers should be independent of each other
 - Base classifiers should do better than a classifier that performs random guessing

Ensemble Methods - Introduction

Rationale for Ensemble Method - Example

- Consider an ensemble of twenty-five binary classifiers, each of which has an error rate of $e : 0.35$
- If the base classifiers are identical, then the ensemble will misclassify the same examples predicted incorrectly by the base classifiers.
- Thus, the error rate of the ensemble remains 0.35
- If the base classifiers are independent-i.e., their errors are uncorrelated-then the ensemble makes a wrong prediction only if more than half of the base classifiers predict incorrectly
- In this case, the error rate of the ensemble classifier is:

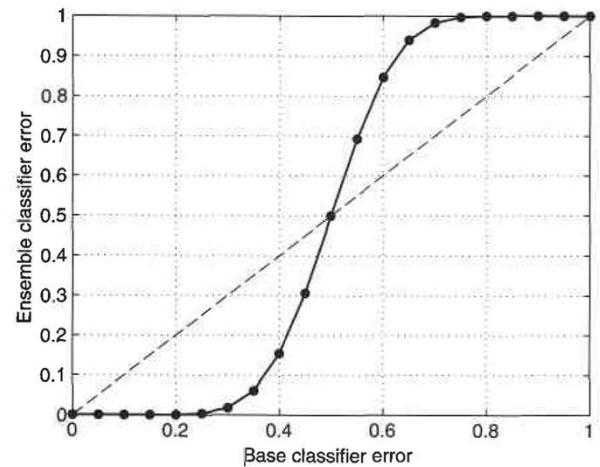


$$e_{\text{ensemble}} = \sum_{i=13}^{25} \binom{25}{i} \epsilon^i (1-\epsilon)^{25-i} = 0.06$$

Ensemble Methods - Introduction

Rationale for Ensemble Method – Example (contd.)

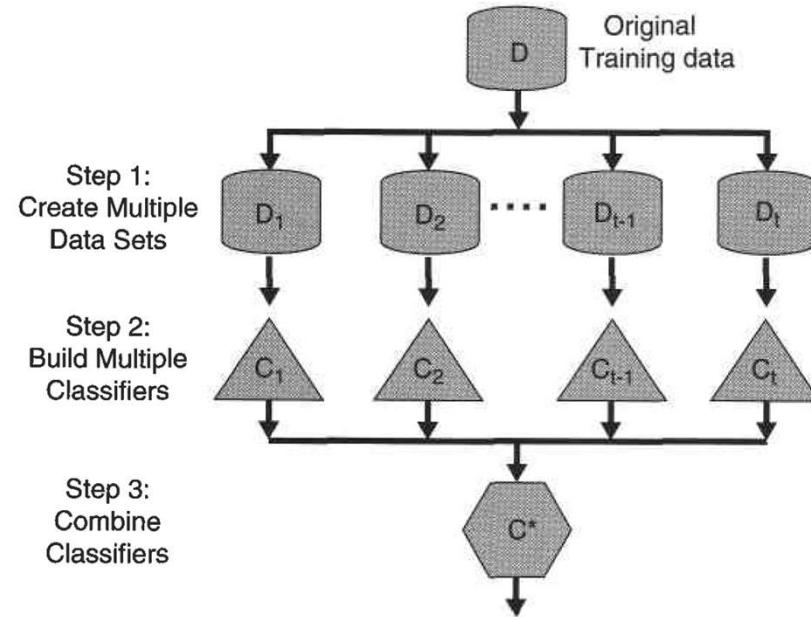
- The diagonal line represents the case in which the base classifiers are identical
- The solid line represents the case in which the base classifiers are independent
- Ensemble classifier performs worse than the base classifiers when e is larger than 0.5
- Necessary conditions for an ensemble classifier to perform better than a single classifier:
 - Base classifiers should be independent of each other
 - Base classifiers should do better than a classifier that performs random guessing
- Nevertheless, improvements in classification accuracies have been observed in ensemble methods in which the base classifiers are slightly correlated



Ensemble Methods - Introduction

Constructing an Ensemble Classifier - Methods

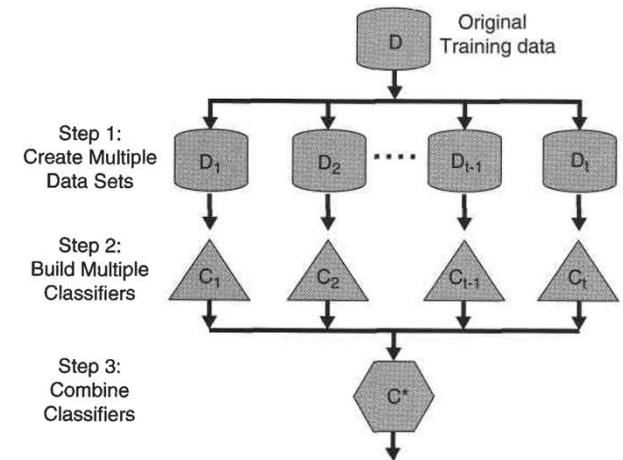
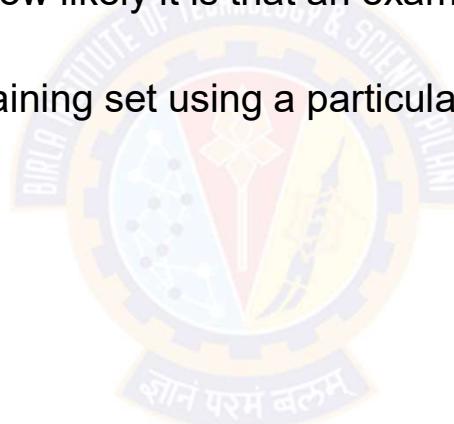
- By manipulating the training set
- By manipulating the input features
- By manipulating the class labels
- By manipulating the learning algorithm



Ensemble Methods - Introduction

Constructing an Ensemble Classifier - by training set

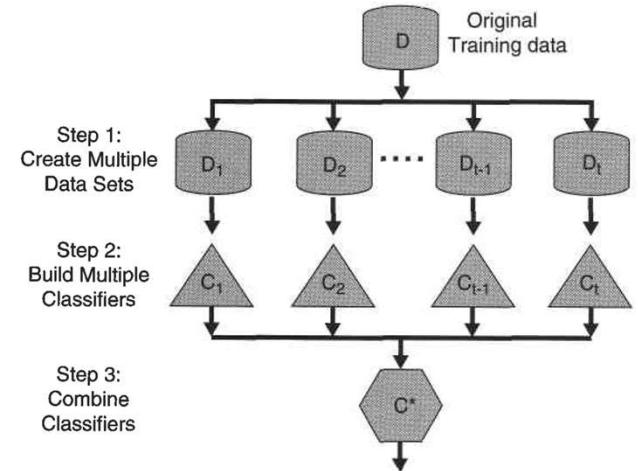
- Multiple training sets are created by resampling the original data according to some sampling distribution
- Sampling distribution determines how likely it is that an example will be selected for training
- Classifier is then built from each training set using a particular learning algorithm.
- Ex: Bagging and Boosting



Ensemble Methods - Introduction

Constructing an Ensemble Classifier - by input features

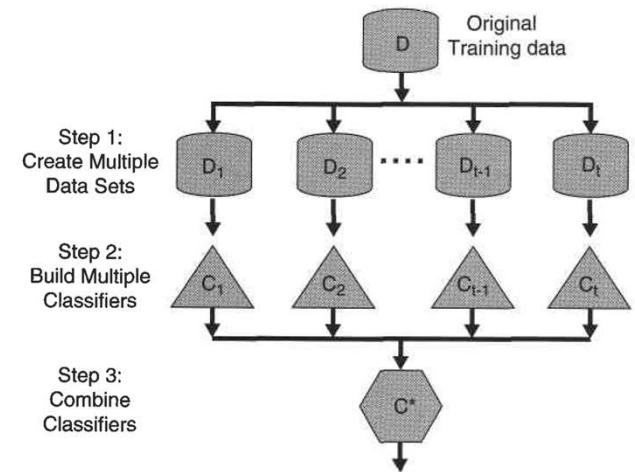
- A subset of input features is chosen to form each training set
- The subset can be either chosen randomly or based on the recommendation of domain experts
- Works well with data sets that contain highly redundant features
- Ex: Random forest



Ensemble Methods - Introduction

Constructing an Ensemble Classifier - by class labels

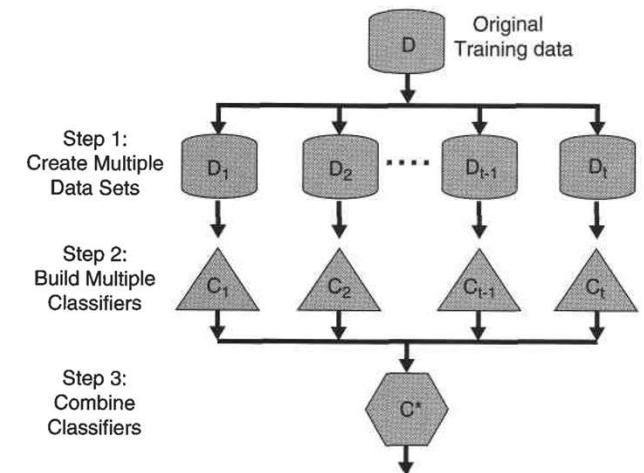
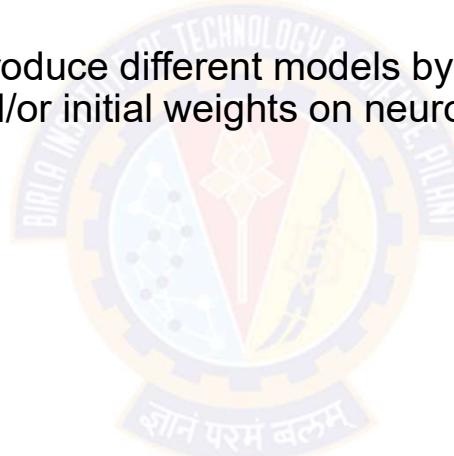
- Used when the number of classes is sufficiently large
- Training Process
 - The training data is transformed into a binary class problem by randomly partitioning the class labels into two disjoint subsets, A_0 and A_1
 - Training examples whose class label belongs to the subset A_0 are assigned to class 0 and so on
 - The relabelled examples are then used to train a base classifier
 - By repeating the class-relabelling and model-building steps multiple times, an ensemble of base classifiers is obtained
- Testing Process
 - When a test example is presented, each base classifier C_i is used to predict its class label
 - If the test example is predicted as class 0, then all the classes that belong to A_0 will receive a vote and so on.
 - The votes are tallied and the class that receives the highest vote is assigned to the test example
- Ex: Error-correcting output coding method



Ensemble Methods - Introduction

Constructing an Ensemble Classifier - by learning algorithm

- Learning algorithms can be manipulated in such a way that applying the same algorithm several times on same training data will result in different models
- Ex: Artificial Neural Network can produce different models by changing the network topology and/or initial weights on neurons



Ensemble Methods - Introduction

General procedure for ensemble method

-
- 1: Let D denote the original training data, k denote the number of base classifiers, and T be the test data.
 - 2: **for** $i = 1$ to k **do**
 - 3: Create training set, D_i from D .
 - 4: Build a base classifier C_i from D_i .
 - 5: **end for**
 - 6: **for** each test record $x \in T$ **do**
 - 7: $C^*(x) = \text{Vote}(C_1(\mathbf{x}), C_2(\mathbf{x}), \dots, C_k(\mathbf{x}))$
 - 8: **end for**
-





BITS Pilani
Pilani | Dubai | Goa | Hyderabad

Bagging-Random Forest Algorithm

Dr. Chetana Gavankar

In this segment

- Bagging
- Random Forest Algorithm

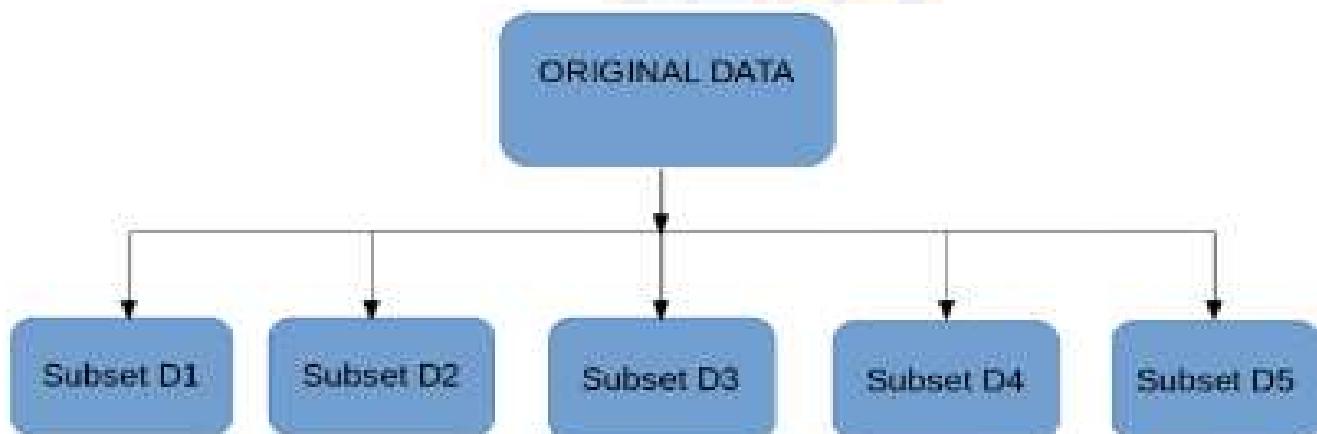


Bagging (Bootstrap Aggregating)

- Technique uses these subsets (bags) to get a fair idea of the distribution (complete set).
- The size of subsets created for bagging may be less than the original set.
- Bootstrapping is a sampling technique in which we create subsets of observations from the original dataset, **with replacement**.
- When you sample with replacement, items are independent. One item does not affect the outcome of the other. You have $1/7$ chance of choosing the first item and a $1/7$ chance of choosing the second item.
- If the two items are **dependent**, or linked to each other. When you choose the first item, you have a $1/7$ probability of picking a item. Assuming you don't replace the item, you only have six items to pick from. That gives you a $1/6$ chance of choosing a second item.

Bagging

- Multiple subsets are created from the original dataset, selecting observations with replacement.
- A base model (weak model) is created on each of these subsets.
- The models run in parallel and are independent of each other.
- The final predictions are determined by combining the predictions from all the models.



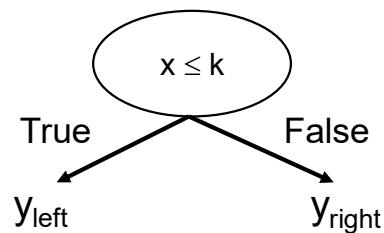
Bagging Example

- Consider 1-dimensional data set:

Original Data:

x	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
y	1	1	1	-1	-1	-1	-1	1	1	1

- Classifier is a decision stump
- Decision tree with one internal node (the root) which is immediately connected to the terminal nodes (its leaves). Decision stump makes a prediction based on the value of just a single input feature. Sometimes they are also called **1-rules**
 - Decision rule: $x \leq k$ versus $x > k$
 - Split point k is chosen based on entropy



Bagging Example

Bagging Round 1:

x	0.1	0.2	0.2	0.3	0.4	0.4	0.5	0.6	0.9	0.9
y	1	1	1	1	-1	-1	-1	-1	1	1

$x \leq 0.35 \rightarrow y = 1$
 $x > 0.35 \rightarrow y = -1$

Bagging Round 2:

x	0.1	0.2	0.3	0.4	0.5	0.5	0.9	1	1	1
y	1	1	1	-1	-1	-1	1	1	1	1

$x \leq 0.7 \rightarrow y = -1$
 $x > 0.7 \rightarrow y = 1$

Bagging Round 3:

x	0.1	0.2	0.3	0.4	0.4	0.5	0.7	0.7	0.8	0.9
y	1	1	1	-1	-1	-1	-1	-1	1	1

$x \leq 0.35 \rightarrow y = 1$
 $x > 0.35 \rightarrow y = -1$

Bagging Round 4:

x	0.1	0.1	0.2	0.4	0.4	0.5	0.5	0.7	0.8	0.9
y	1	1	1	-1	-1	-1	-1	-1	1	1

$x \leq 0.3 \rightarrow y = 1$
 $x > 0.3 \rightarrow y = -1$

Bagging Round 5:

x	0.1	0.1	0.2	0.5	0.6	0.6	0.6	1	1	1
y	1	1	1	-1	-1	-1	-1	1	1	1

$x \leq 0.35 \rightarrow y = 1$
 $x > 0.35 \rightarrow y = -1$

Bagging Example

Bagging Round 6:

x	0.2	0.4	0.5	0.6	0.7	0.7	0.7	0.8	0.9	1
y	1	-1	-1	-1	-1	-1	-1	1	1	1

$x \leq 0.75 \rightarrow y = -1$
 $x > 0.75 \rightarrow y = 1$

Bagging Round 7:

x	0.1	0.4	0.4	0.6	0.7	0.8	0.9	0.9	0.9	1
y	1	-1	-1	-1	-1	1	1	1	1	1

$x \leq 0.75 \rightarrow y = -1$
 $x > 0.75 \rightarrow y = 1$

Bagging Round 8:

x	0.1	0.2	0.5	0.5	0.5	0.7	0.7	0.8	0.9	1
y	1	1	-1	-1	-1	-1	-1	1	1	1

$x \leq 0.75 \rightarrow y = -1$
 $x > 0.75 \rightarrow y = 1$

Bagging Round 9:

x	0.1	0.3	0.4	0.4	0.6	0.7	0.7	0.8	1	1
y	1	1	-1	-1	-1	-1	-1	1	1	1

$x \leq 0.75 \rightarrow y = -1$
 $x > 0.75 \rightarrow y = 1$

Bagging Round 10:

x	0.1	0.1	0.1	0.1	0.3	0.3	0.8	0.8	0.9	0.9
y	1	1	1	1	1	1	1	1	1	1

$x \leq 0.05 \rightarrow y = 1$
 $x > 0.05 \rightarrow y = 1$

Bagging Example

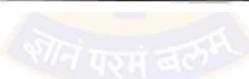
- Assume test set is the same as the original data
- Use majority vote to determine class of ensemble classifier

Round	x=0.1	x=0.2	x=0.3	x=0.4	x=0.5	x=0.6	x=0.7	x=0.8	x=0.9	x=1.0
1	1	1	1	-1	-1	-1	-1	-1	-1	-1
2	1	1	1	1	1	1	1	1	1	1
3	1	1	1	-1	-1	-1	-1	-1	-1	-1
4	1	1	1	-1	-1	-1	-1	-1	-1	-1
5	1	1	1	-1	-1	-1	-1	-1	-1	-1
6	-1	-1	-1	-1	-1	-1	-1	1	1	1
7	-1	-1	-1	-1	-1	-1	-1	1	1	1
8	-1	-1	-1	-1	-1	-1	-1	1	1	1
9	-1	-1	-1	-1	-1	-1	-1	1	1	1
10	1	1	1	1	1	1	1	1	1	1
Sum	2	2	2	-6	-6	-6	-6	2	2	2
Sign	1	1	1	-1	-1	-1	-1	1	1	1

Bagging

Bagging - Algorithm

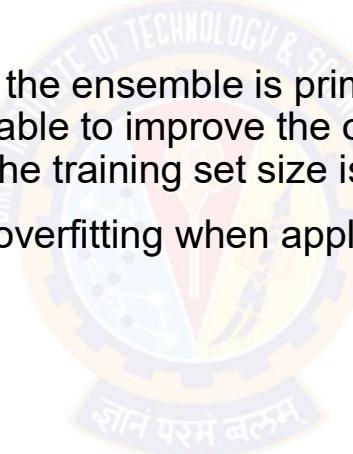
-
- 1: Let k be the number of bootstrap samples.
 - 2: **for** $i = 1$ to k **do**
 - 3: Create a bootstrap sample of size N , D_i .
 - 4: Train a base classifier C_i on the bootstrap sample D_i .
 - 5: **end for**
 - 6: $C^*(x) = \operatorname{argmax}_y \sum_i \delta(C_i(x) = y).$
 $\{\delta(\cdot) = 1 \text{ if its argument is true and 0 otherwise}\}.$
-



Bagging

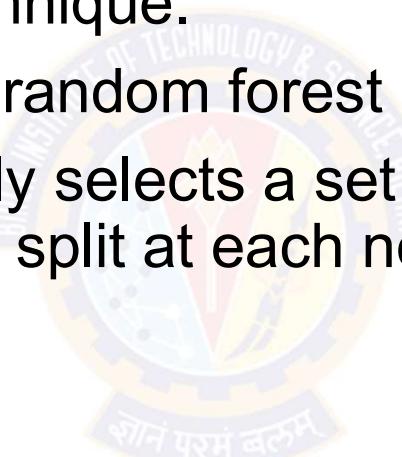
Bagging – Salient features

- Improves generalization error by reducing the variance of the base classifier
- For unstable base classifier, bagging helps to reduce the errors associated with random fluctuations in the training data
- For stable base classifier, the error of the ensemble is primarily caused by bias in the base classifier, hence bagging may not be able to improve the overall performance or may even degrade the base classifier's performance as the training set size is effectively 37% less
- Bagging is less susceptible to model overfitting when applied to noisy data, since sample selection is random



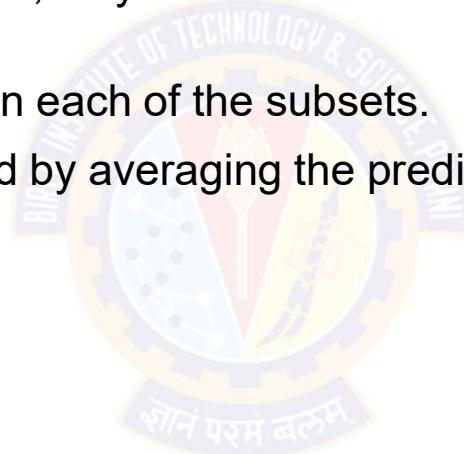
Random Forest

- Random Forest is ensemble machine learning algorithm that follows the bagging technique.
- The base estimators in random forest are decision trees.
- Random forest randomly selects a set of features which are used to decide the best split at each node of the decision tree.



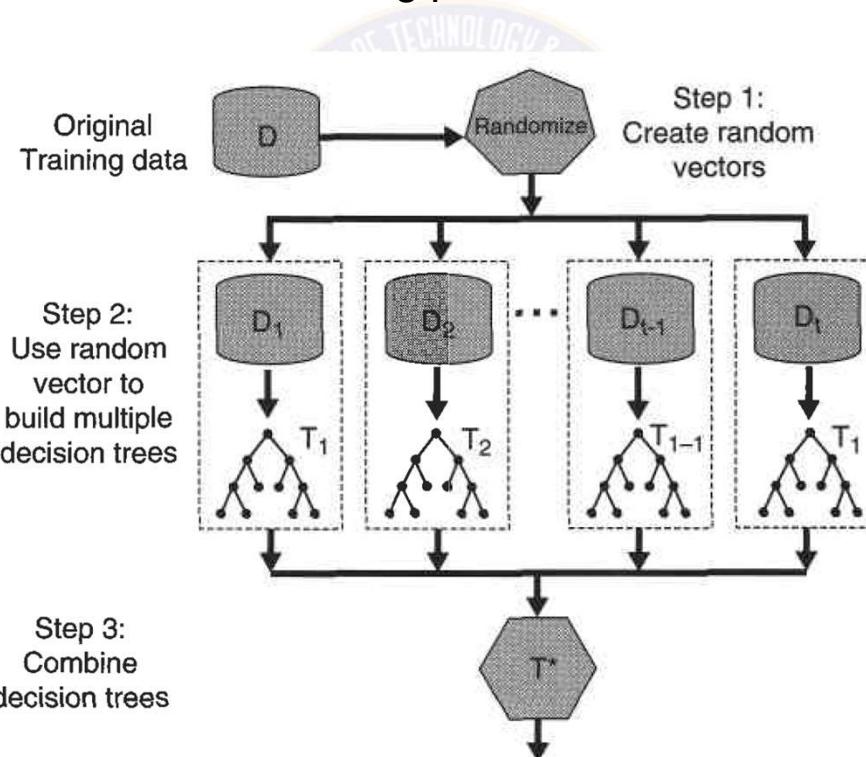
Random Forest

- Random subsets are created from the original dataset (bootstrapping).
- At each node in the decision tree, only a random set of features are considered to decide the best split.
- A decision tree model is fitted on each of the subsets.
- The final prediction is calculated by averaging the predictions from all decision trees.



Random Forest

- Combines the predictions made by multiple decision trees
- Each tree is generated based on the values of an independent set of random vectors
- Randomness is injected into the model-building process



Random Input selection

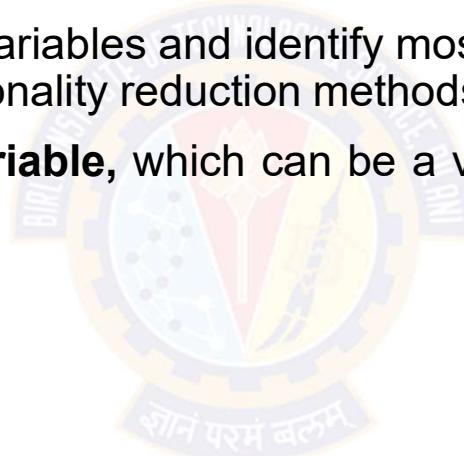
- Randomly select F input features to split at each node of the decision tree and then fully grow the tree without pruning
- This helps reduce the bias present in the resulting tree
- The predictions are combined using a majority voting scheme
- To increase randomness, bagging can also be used to generate bootstrap samples
- The strength and correlation of random forests may depend on the size of F features
 - If F is sufficiently small, then the trees tend to become less correlated
- The strength of the tree classifier tends to improve with a larger number of F
- Optimal number of $F = \log_2 d + 1$ (where d is number of input features)
- Reduces the runtime of the algorithm

Random Combinations

- Used when number of features d is small
- Increases the feature space is to create linear combinations of the input features
- Specifically, at each node, a new feature is generated by randomly selecting L of the input features
- The input features are linearly combined
- At each node, F of such randomly combined new features are generated
- The best of them is subsequently selected to split the node

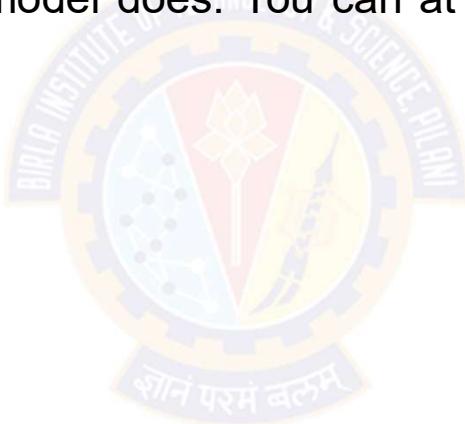
Advantages of Random Forest

- Algorithm can solve both type of problems i.e. classification and regression
- Handles large data set with higher dimensionality.
- It can handle thousands of input variables and identify most significant variables so it is considered as one of the dimensionality reduction methods.
- Model outputs **Importance of variable**, which can be a very handy feature (on some random data set).



Disadvantages of Random Forest

- May over-fit data sets that are particularly noisy.
- Random Forest can feel like a black box approach for statistical modelers – you have very little control on what the model does. You can at best – try different parameters and random seeds!





BITS Pilani
Pilani | Dubai | Goa | Hyderabad

eXtreme Gradient Boosting (XGBoost)

Dr. Chetana Gavankar

In this segment

eXtreme Gradient Boosting (XGBoost)

- Gradient Boosting
- XGBoost



Gradient Boosting

- In Gradient Boosting, "shortcomings" are identified by gradients.
- Recall that, in Adaboost, "shortcomings" are identified by high-weight data points. Both high-weight data points and gradients tell us how to improve our model.



eXtreme Gradient Boosting (XGBoost)

Gradient Boosting

- Gradient boosting was developed by Friedman as a generalization of AdaBoost
- It is a numerical optimization problem where the objective is to minimize the loss of the model by adding weak learners using a gradient descent like procedure



Gradient Boosting

- Gradient Boosting for Different Problems Difficulty: regression ==> classification ==> ranking



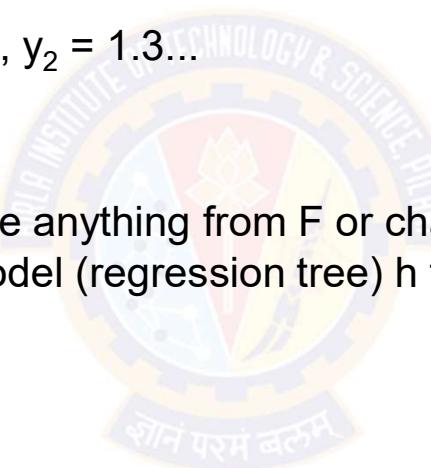
Gradient Boosting

- You are given $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$, and the task is to fit a model $F(x)$ to minimize loss
- There are some mistakes:

$F(x_1) = 0.8$, $y_1 = 0.9$, and $F(x_2) = 1.4$, $y_2 = 1.3\dots$

How can you improve this model?

- Rules:
 - You are not allowed to remove anything from F or change any parameter in F .
 - You can add an additional model (regression tree) h to F , so the new prediction will be $F(x) + h(x)$.



Gradient Boosting

- You wish to improve the model such that

- $F(x_1) + h(x_1) = y_1$
- $F(x_2) + h(x_2) = y_2 \dots$
- $F(x_n) + h(x_n) = y_n$

Or, equivalently, you wish

$$h(x_1) = y_1 - F(x_1)$$

$$h(x_2) = y_2 - F(x_2) \dots$$

$$h(x_n) = y_n - F(x_n)$$

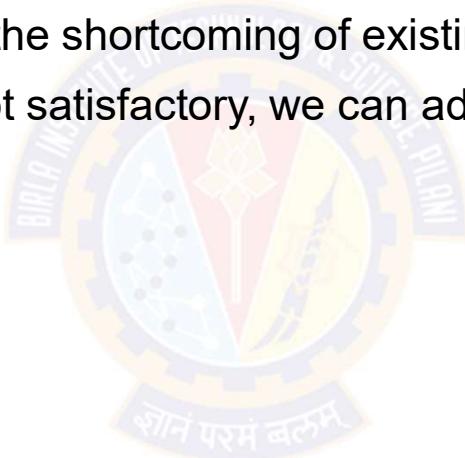
Fit a tree h to data

$$(x_1, y_1 - F(x_1)), (x_2, y_2 - F(x_2)), \dots, (x_n, y_n - F(x_n))$$



Gradient Boosting

- Simple solution: $y_i - F(x_i)$ are called residuals. These are the parts that existing model F cannot do well.
- The role of h is to compensate the shortcoming of existing model F .
- If the new model $F + h$ is still not satisfactory, we can add another tree...



eXtreme Gradient Boosting (XGBoost)

XGBoost

- XGBoost is an implementation of gradient boosted decision trees designed for speed and performance, created by Tianqi Chen, now with contributions from developer community
- Model features
 - Gradient Boosting
 - Stochastic Gradient Boosting
 - Regularized Gradient Boosting



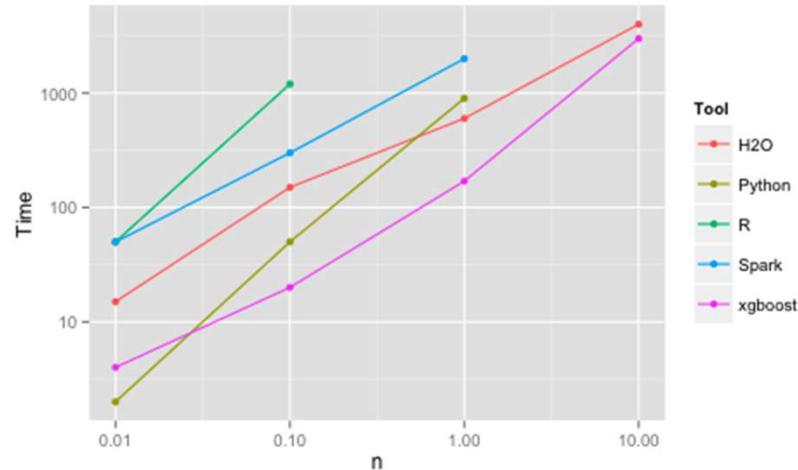
Why XGBoost so popular?

- **Speed** : faster than other ensemble classifiers.
- **Core algorithm is parallelizable**: harness the power of multi-core computers and networks of computers enabling to train on very large datasets
- **Consistently outperforms other algorithm methods** : It has shown better performance on a variety of machine learning benchmark datasets.
- **Wide variety of tuning parameters** : cross-validation, regularization, missing values, tree parameters, etc
- XGBoost (Extreme Gradient Boosting) uses the gradient boosting (GBM) framework at its core.

eXtreme Gradient Boosting (XGBoost)

XGBoost – Why use ?

- XGBoost Execution Speed



- XGBoost Model Performance

- XGBoost dominates structured or tabular datasets on classification and regression predictive modelling problems.
- The evidence is that it is the go-to algorithm for competition winners on the Kaggle competitive data science platform.

eXtreme Gradient Boosting (XGBoost)

Sources

- <https://machinelearningmastery.com/gentle-introduction-gradient-boosting-algorithm-machine-learning/>
- <https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/>
- <https://towardsdatascience.com/a-beginners-guide-to-xgboost-87f5d4c30ed7>
- https://en.wikipedia.org/wiki/Gradient_boosting



BITS Pilani

Pilani|Duba|Goa|Hyderabad

K-Nearest Neighbor Classification

Swarna Chaudhary
Assistant Professor
Swarna.Chaudhary@pilani.bits-pilani.ac.in



- *The slides presented here are obtained from the authors of the books and from various other contributors. I hereby acknowledge all the contributors for their material and inputs.*
- *I have added and modified a few slides to suit the requirements of the course.*



In this segment

- kNN classifier – Introduction
- Model Evaluation Measures
- Finding optimal k
- Python Implementation



Lazy vs. Eager Learning

Lazy vs. eager learning

- Lazy learning (e.g., instance-based learning): Simply stores training data (or only minor processing) and waits until it is given a test tuple
- Eager learning (the above discussed methods): Given a set of training set, constructs a classification model before receiving new (e.g., test) data to classify

Lazy: less time in training but more time in predicting



k-Nearest Neighbor Classifier

- Nearest Neighbour classifier is an instance based classifier, in contrast to generative models
- Also called ‘lazy learning’, as learning is postponed until a new instance is encountered
- Constructs a local approximation to the target function, applicable (better suited) in the neighbourhood of new instance
- Suitable in cases where target function is complex over the entire input space, but easily describable in local approximations
- Caveat is the high cost of classification, which happens at the time of processing rather than before hand (there's no training phase)



k-NN Classifier - Introduction

- Considers all instances as members of n-dimensional space
- Nearest neighbours of an instance is determined based on distance measures.
- For NN classifier, target function can be discrete or continuous



Distance Measures: Minkowski Distance

Minkowski distance: A popular distance measure

$$d(i, j) = \sqrt[h]{|x_{i1} - x_{j1}|^h + |x_{i2} - x_{j2}|^h + \cdots + |x_{ip} - x_{jp}|^h}$$

where $i = (x_{i1}, x_{i2}, \dots, x_{ip})$ and $j = (x_{j1}, x_{j2}, \dots, x_{jp})$ are two p -dimensional data objects, and h is the order (the distance so defined is also called L- h norm)

Properties

- $d(i, j) > 0$ if $i \neq j$, and $d(i, i) = 0$ (Positive definiteness)
- $d(i, j) = d(j, i)$ (Symmetry)
- $d(i, j) \leq d(i, k) + d(k, j)$ (Triangle Inequality)

A distance that satisfies these properties is a **metric**



Distance Measures : Special Cases of Minkowski

- $h = 1$: Manhattan (city block, L₁ norm) distance
 - E.g., the Hamming distance: the number of bits that are different between two binary vectors

$$d(i, j) = |x_{i1} - x_{j1}| + |x_{i2} - x_{j2}| + \dots + |x_{ip} - x_{jp}|$$

- $h = 2$: (L₂ norm) Euclidean distance

$$d(i, j) = \sqrt{(|x_{i1} - x_{j1}|^2 + |x_{i2} - x_{j2}|^2 + \dots + |x_{ip} - x_{jp}|^2)}$$

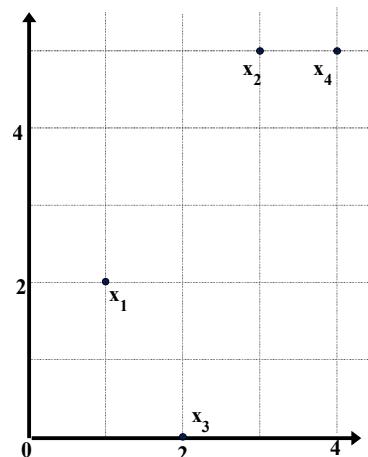
- $h \rightarrow \infty$. “supremum” (L_{max} norm, L _{∞} norm) distance.
 - This is the maximum difference between any component (attribute) of the vectors

$$d(i, j) = \lim_{h \rightarrow \infty} \left(\sum_{f=1}^p |x_{if} - x_{jf}|^h \right)^{\frac{1}{h}} = \max_f |x_{if} - x_{jf}|$$

Example: Minkowski Distance

(Dissimilarity Matrices)

point	attribute 1	attribute 2
x1	1	2
x2	3	5
x3	2	0
x4	4	5



Manhattan (L_1)

L	x1	x2	x3	x4
x1	0			
x2	5	0		
x3	3	6	0	
x4	6	1	7	0

Euclidean (L_2)

L2	x1	x2	x3	x4
x1	0			
x2	3.61	0		
x3	2.24	5.1	0	
x4	4.24	1	5.39	0

Supremum

L_∞	x1	x2	x3	x4
x1	0			
x2	3	0		
x3	2	5	0	
x4	3	1	5	0



Standardizing Numeric Data

- Z-score:
 - X: raw score to be standardized, μ : mean of the population, σ : standard deviation
 - the distance between the raw score and the population mean in units of the standard deviation
 - negative when the raw score is below the mean, “+” when above

Where

$$z = \frac{x - \mu}{\sigma}$$

Ordinal Variables

- An ordinal variable can be discrete or continuous
- Order is important, e.g., rank
- Can be treated like interval-scaled
 - replace x_{if} by their rank
 - map the range of each variable onto $[0, 1]$ by replacing i -th object in the f -th variable by
- compute the dissimilarity using methods for interval-scaled variables

Attributes of Mixed Type: Gower distance

- A database may contain all attribute types
 - Nominal, symmetric binary, asymmetric binary, numeric, ordinal

$$d(i, j) = \frac{\sum_{c=1}^n \omega_c \delta_{ij}^{(c)} d_{ij}^{(c)}}{\sum_{c=1}^n \omega_c \delta_{ij}^{(c)}}$$

$d(i, j)$ = dissimilarity between row i and row j

c = the c th column

n = number of columns in the dataset

ω_c = weight of c th column = $\frac{1}{\text{#rows in dataset}}$

$\delta_{ij}^{(c)}$ =
$$\begin{cases} 0 & \text{if column } c \text{ is missing in row } i \text{ or } j \\ 0 & \text{if column } c \text{ is asymmetric binary and both} \\ & \text{values in row } i \text{ and } j \text{ are 0} \\ 1 & \text{otherwise} \end{cases}$$

$d_{ij}^{(c)}$ (categorical) =
$$\begin{cases} 0 & \text{if } i \text{ and } j \text{ are equal in column } c \\ 1 & \text{otherwise} \end{cases}$$

$d_{ij}^{(c)}$ (continuous/ordinal) =
$$\frac{|(\text{row } i \text{ in column } c) - (\text{row } j \text{ in column } c)|}{\max(\text{column } c) - \min(\text{column } c)}$$

<https://healthcare.ai/clustering-non-continuous-variables/>

Example

Based on the information given in the table below, find most similar and most dissimilar persons among them. Apply min-max normalization on income to obtain [0,1] range. Consider profession and mother tongue as nominal. Consider native place as ordinal variable with ranking order of Village, Small Town, Suburban, Metropolitan. Each attribute.

Name	Income	Profession	Mother tongue	Native Place
Ram	70000	Doctor	Bengali	Village
Balram	50000	Data Scientist	Hindi	Small Town
Bharat	60000	Carpenter	Hindi	Suburban
Kishan	80000	Doctor	Bhojpuri	Metropolitan

Solution

After normalizing income and quantifying native place, we get

Name	Income	Profession	Mother tongue	Native Place
Ram	0.67	Doctor	Bengali	1
Balram	0	Data Scientist	Hindi	2
Bharat	0.33	Carpenter	Hindi	3
Kishan	1	Doctor	Bhojpuri	4

$$d(\text{Ram}, \text{Balram}) = 0.67 + 1 + 1 + (2-1)/(4-1) = 3 \quad d(\text{Ram}, \text{Bharat}) = 0.33 + 1 + 1 + (3-1)/(4-1) = 3$$

$$d(\text{Ram}, \text{Kishan}) = 0.33 + 0 + 1 + (4-1)/(4-1) = 2.33 \quad d(\text{Balram}, \text{Bharat}) = 0.33 + 1 + 0 + (3-2)/(4-1) = 1.67$$

$$d(\text{Balram}, \text{Kishan}) = 1 + 1 + 1 + (4-2)/(4-1) = 3.67 \quad d(\text{Bharat}, \text{Kishan}) = 0.67 + 1 + 1 + (4-3)/(4-1) = 3$$

Most similar – Balram and Bharat; Most dissimilar – Balram and Kishan



k-Nearest Neighbor Classifier

kNN Classifier - Algorithm

Training algorithm:

- For each training example $\langle x, f(x) \rangle$, add the example to the list *training_examples*

Classification algorithm:

- Given a query instance x_q to be classified,
 - Let $x_1 \dots x_k$ denote the k instances from *training_examples* that are nearest to x_q
 - Return

$$\hat{f}(x_q) \leftarrow \operatorname{argmax}_{v \in V} \sum_{i=1}^k \delta(v, f(x_i))$$

where $\delta(a, b) = 1$ if $a = b$ and where $\delta(a, b) = 0$ otherwise.

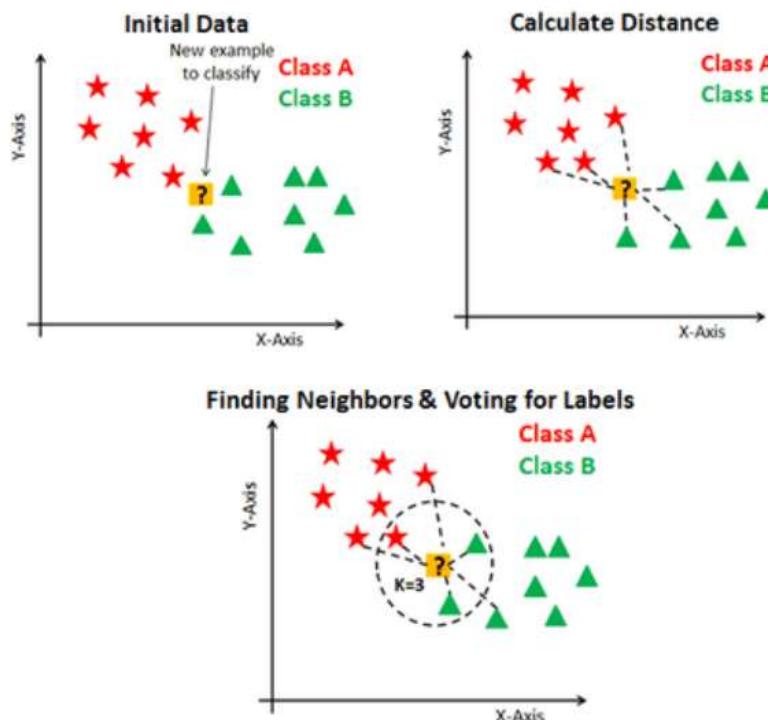
* It can be used for Regression as well.



Steps involved in kNN

- Divide the data into training and test data.
- Select a value K.
- Determine which distance function is to be used.
- Choose a sample from the test data that needs to be classified and compute the distance to its n training samples.
- Sort the distances obtained and take the k-nearest data samples.
- Assign the test class to the class based on the majority vote of its k neighbors.

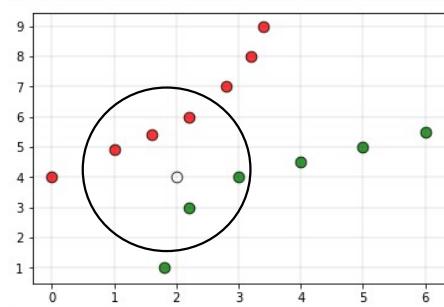
k-Nearest Neighbor Classifier



Source: DataCamp

Challenges in k-NN

- Performance of a classifier largely depends on the of the hyperparameter k
 - Choosing smaller values for K, noise can have a higher influence on the result.
 - Larger values of k are computationally expensive
- Assigning the class labels can be tricky. For example, in the below case, for (k=5) the point is closer to ‘green’ classification, but gets classified as ‘red’ due to higher red votes/majority voting to ‘red’





kNN Classifier - Distance weighted

- This refinement adds a "weight" factor to kNN algorithm
- Weight would be based on the distance (ex: inverse square of distance from query instance), which gives more weightage to the closer neighbours
- Modified function would be:
- where

$$\hat{f}(x_q) \leftarrow \operatorname{argmax}_{v \in V} \sum_{i=1}^k w_i \delta(v, f(x_i)) \quad w_i \equiv \frac{1}{d(x_q, x_i)^2}$$

- This method is robust to noisy training data and effective for large set of training data
- Con for kNN is the well-known “curse of dimensionality”
 - Can be overcomed by weighing each attribute differently while calculating distances or eliminating irrelevant attributes



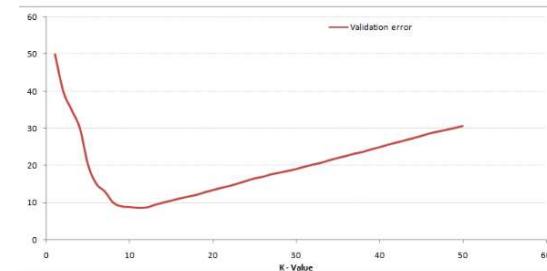
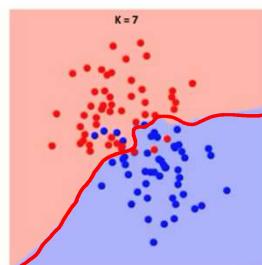
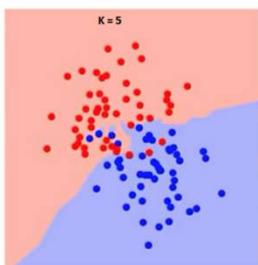
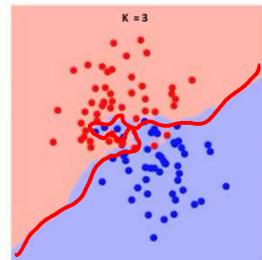
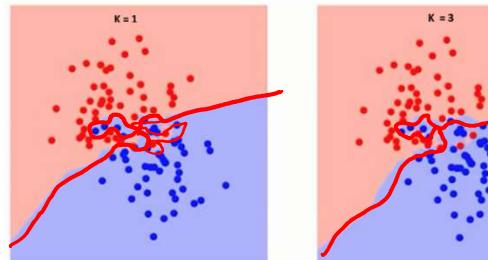
Finding optimal k for kNN classifiers

- The problem of finding K
- Optimal K



Finding optimal k for kNN classifiers

What is the ideal value of 'K'?





Factors that affect kNN Performance

Performance of the K-NN algorithm is influenced by the following factors :

- The distance function or distance metric used to determine the nearest neighbors.
- The number of neighbors (=k) used to classify the new example.



PROS and CONS

Pros

- The K-NN algorithm is very easy to implement.
- Nearly optimal in the large sample limit.
- Uses local information, which can yield highly adaptive behavior.

Cons

- Large storage requirements.
- Computationally intensive recall.
- Highly susceptible to the curse of dimensionality.
 - Consider all attributes of instances to retrieve similar training examples from memory



Model Evaluation Measures

- Metrics
 - Precision
 - Recall
 - F1
- ROC Curve – AUC of ROC
- Generating ROC Curve



Classification—A Two-Step Process

Model construction: describing a set of predetermined classes

- Each tuple/sample is assumed to belong to a predefined class, as determined by the class label attribute
- The set of tuples used for model construction is training set
- The model is represented as classification rules, decision trees, or mathematical formulae

Model usage: for classifying future or unknown objects

- Estimate accuracy of the model
 - The known label of test sample is compared with the classified result from the model
 - Accuracy rate is the percentage of test set samples that are correctly classified by the model
 - Test set is independent of training set, otherwise over-fitting will occur
- If the accuracy is acceptable, use the model to classify data tuples whose class labels are not known



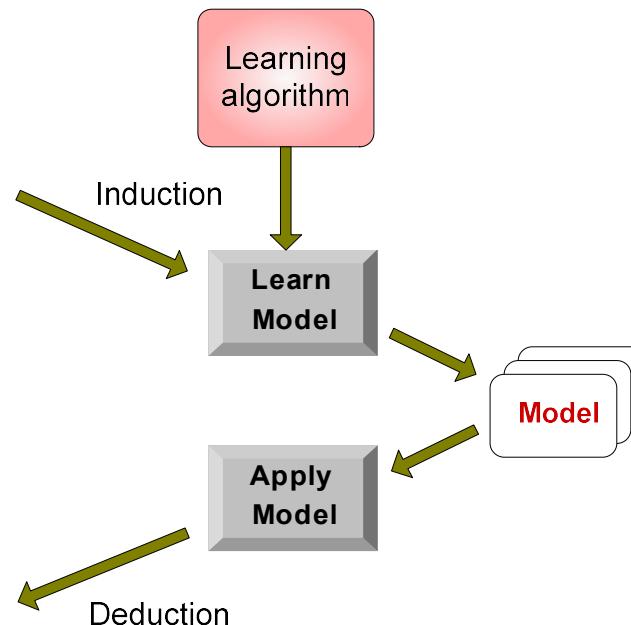
Illustrating Classification Task

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set

Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set





Classifier Evaluation Metrics: Confusion Matrix

Confusion Matrix:

Given m classes, an entry, $CM_{i,j}$ in a **confusion matrix** indicates # of tuples in class i that were labeled by the classifier as class j

May have extra rows/columns to provide totals

- **True Positive (TP):** It refers to the number of predictions where the classifier correctly predicts the positive class as positive.
- **True Negative (TN):** It refers to the number of predictions where the classifier correctly predicts the negative class as negative.
- **False Positive (FP):** It refers to the number of predictions where the classifier incorrectly predicts the negative class as positive.
- **False Negative (FN):** It refers to the number of predictions where the classifier incorrectly predicts the positive class as negative.

Predicted class ->	C_1	$\neg C_1$
Actual class ↓		
C_1	True Positives (TP)	False Negatives (FN)
$\neg C_1$	False Positives (FP)	True Negatives (TN)



Classifier Evaluation Metrics: Accuracy, Error Rate,

Classifier Accuracy, or recognition rate: percentage of test set tuples that are correctly classified

$$\text{Accuracy} = (\text{TP} + \text{TN})/\text{All}$$

most effective when the class distribution is relatively balanced

Classification Error/ Misclassification rate: $1 - \text{accuracy}$, or

$$= (\text{FP} + \text{FN})/\text{All}$$

A\P	C	$\neg C$	
C	TP	FN	P
$\neg C$	FP	TN	N
	P'	N'	All



Example

Given below is a confusion matrix for medical data where the class values are yes and no for a class label attribute, cancer. Calculate the accuracy of the classifier.

Classes	yes	no	Total	Recognition (%)
yes	90	210	300	30.00
no	140	9560	9700	98.56
Total	230	9770	10,000	96.40

Confusion matrix for the classes *cancer = yes* and *cancer = no*.



Classifier Evaluation Metrics: Example

Actual Class\Predicted class	cancer = yes	cancer = no	Total	Recognition(%)
cancer = yes	90	210	300	30.00 (<i>sensitivity</i>)
cancer = no	140	9560	9700	98.56 (<i>specificity</i>)
Total	230	9770	10000	96.40 (<i>accuracy</i>)



Class Imbalance Problem

- The main class of interest is rare.
- the data set distribution reflects a significant majority of the negative class and a minority positive class.
- For example,
 - fraud detection applications, the class of interest (or positive class) is “*fraud*,”
 - medical tests, there may be a rare class, such as “*cancer*”
- Accuracy might not be a good option for measuring performance in case of class imbalance problem

Approaches to solve Class imbalance problem

- Generate Synthetic Samples
 - new samples based on the distances between the point and its nearest neighbors
 - E.g. **Synthetic Minority Oversampling Technique**, or **SMOTE class in sklearn**
- Change the performance metric
 - Use Recall, Precision or ROC curves instead of accuracy
- Try different algorithms
 - Some algorithms as Support Vector Machines and Tree-Based algorithms may work better with imbalanced classes.



Model Evaluation Measures

- True positive rate (TPR) or **sensitivity** is defined as the fraction of positive examples predicted correctly by the model

$$TPR = TP / (TP + FN)$$

- True negative rate (TNR) or **specificity** is defined as the fraction of negative examples predicted correctly by the model

$$TNR = TN / (TN + FP)$$

- False positive rate (FPR) is the fraction of negative examples predicted as a positive class

$$FPR = FP / (TN + FP)$$

- False negative rate (FNR) is the fraction of positive examples predicted as a negative class

$$FNR = FN / (TP + FN)$$

A \ P	C	$\neg C$	
C	TP	FN	P
$\neg C$	FP	TN	N
	P'	N'	All



Classifier Evaluation Metrics: Precision and Recall, and F-measures

Precision: exactness – what % of tuples that the classifier labeled as positive are actually positive

$$precision = \frac{TP}{TP + FP}$$

Recall/Sensitivity: completeness – what % of positive tuples did the classifier label as positive?

Perfect score is 1.0

$$recall = \frac{TP}{TP + FN}$$

Inverse relationship between precision & recall

F measure (F_1 or F-score): harmonic mean of precision and recall,

$$F = \frac{2 \times precision \times recall}{precision + recall}$$

High value of F_1 measure ensures that both precision and recall are high

F_β : weighted measure of precision and recall

- assigns β times as much weight to recall as to precision

$$F_\beta = \frac{(1 + \beta^2) \times precision \times recall}{\beta^2 \times precision + recall}$$



Classifier Evaluation Metrics: Example

$Precision = 90/230 = 39.13\%$

$Recall = 90/300 = 30.00\%$

Actual Class\Predicted class	cancer = yes	cancer = no	Total	Recognition(%)
cancer = yes	90	210	300	30.00 (<i>sensitivity</i>)
cancer = no	140	9560	9700	98.56 (<i>specificity</i>)
Total	230	9770	10000	96.40 (<i>accuracy</i>)



Example: Contingency for Multi-Class Classifier

		True Class		
		Apple	Orange	Mango
Predicted Class	Apple	7	8	9
	Orange	1	2	3
	Mango	3	2	1

Class	Precision	Recall	F1-score
Apple	0.29	0.64	0.40
Orange	0.33	0.17	0.22
Mango	0.17	0.08	0.11

Precision, Recall and F1-score for Each Class

357



Evaluating Classifier Accuracy: Holdout & Cross-Validation Methods

Holdout method

- Given data is randomly partitioned into two independent sets
 - Training set (e.g., 2/3) for model construction
 - Test set (e.g., 1/3) for accuracy estimation
- Random sampling: a variation of holdout
 - Repeat holdout k times, accuracy = avg. of the accuracies obtained

Cross-validation (k -fold, where $k = 10$ is most popular)

- Randomly partition the data into k *mutually exclusive* subsets, each approximately equal size
- At i -th iteration, use D_i as test set and others as training set
- ***Stratified cross-validation***: folds are stratified so that class dist. in each fold is approx. the same as that in the initial data
- The Accuracy of the model is the average of the accuracy of each fold.



Cross Validation

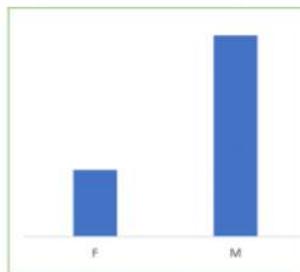
k-Fold Cross Validation:



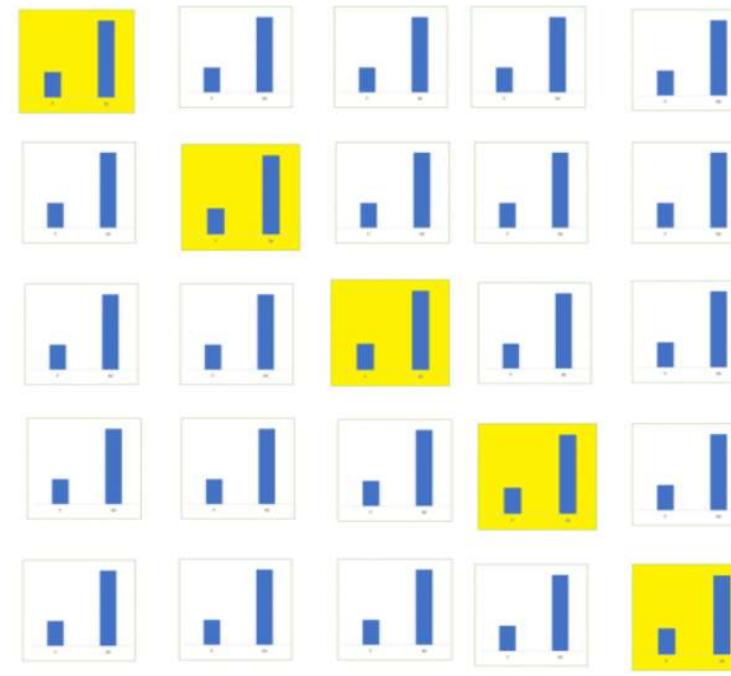


Stratified Cross Validation

Stratified K-Fold
Cross Validation
(K=5)



Class Distributions



Fold 1

Fold 2

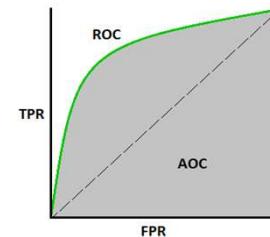
Fold 3

Fold 4

Fold 5

Receiver Operating Characteristic (ROC) Curve

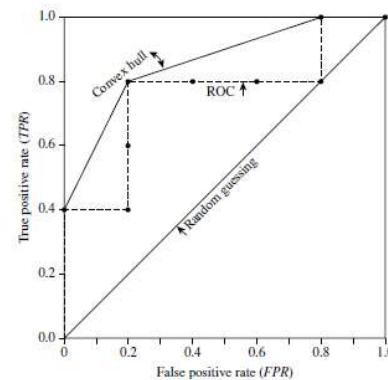
- ROC curve plots TPR and FPR, to graphically represent their trade-off
- AUC represents degree or measure of separability. It tells how much model is capable of distinguishing between classes.
 - Higher the AUC, better the model is at predicting
- Area Under Curve (AUC) of ROC – evaluates model performance on average
 - AUC of ROC = 1, for a perfect model
 - AUC of ROC = 0.5, if the model is random
- For model comparison, AUC of ROC should be larger for the model to be superior or better performing



Example

The table below shows the probability value (column 3) returned by a probabilistic classifier for each of the 10 tuples in a test set, sorted by decreasing probability order. The corresponding ROC is given on right hand side.

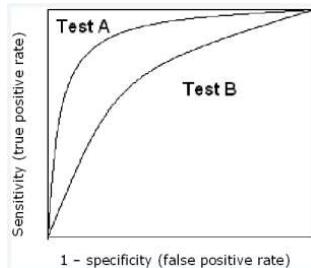
Tuple #	Class	Prob.	TP	FP	TN	FN	TPR	FPR
1	P	0.90	1	0	5	4	0.2	0
2	P	0.80	2	0	5	3	0.4	0
3	N	0.70	2	1	4	3	0.4	0.2
4	P	0.60	3	1	4	2	0.6	0.2
5	P	0.55	4	1	4	1	0.8	0.2
6	N	0.54	4	2	3	1	0.8	0.4
7	N	0.53	4	3	2	1	0.8	0.6
8	N	0.51	4	4	1	1	0.8	0.8
9	P	0.50	5	4	0	1	1.0	0.8
10	N	0.40	5	5	0	0	1.0	1.0





Usage

- Threshold selection
- Performance assessment
- Classifier comparison





Practise Problems

- **Tune KNN.** Try different values of k to see if you can improve the performance of the algorithm.
- **Use weighted kNN** and check the performance
- **Regression.** Apply it to a regression predictive modeling problem (e.g. predict a numerical value)
- **More Distance Measures.** Implement other distance measures such as Hamming distance, Manhattan distance and Minkowski distance.
- **Data Preparation.** Distance measures are strongly affected by the scale of the input data. Experiment with standardization and other data preparation methods in order to improve results.
- **Dealing with categorical attributes**



Naïve Bayesian Classifier



BITS Pilani

Pilani|Duba|Goa|Hyderabad

Swarna Chaudhary
Assistant Professor
swarna.chaudhary@pilani.bits-pilani.ac.in



- *The slides presented here are obtained from the authors of the books and from various other contributors. I hereby acknowledge all the contributors for their material and inputs.*
- *I have added and modified a few slides to suit the requirements of the course.*



In this segment

- Probability Basics
- Bayes Theorem
- Naïve Bayesian Classifier
- An illustrative Example
- Naïve Bayes Classifier is a generative model
- Advantages of Naïve Bayes Classifier and when to use Naïve Bayes Classifier?
- Interpretability of Naïve Bayes Classifier
- Python Implementation



Probability basics

Random Experiment

- A random experiment is an experiment or a process for which the outcome cannot be predicted with certainty
- Example:
 - Tossing a coin
 - number of calls to a communication system during a fixed length interval of time
 - Rolling a die

Sample Space:

- The sample space of a random experiment is the set of all possible outcomes.
Denoted by omega Ω/S
- Example:

Random Experiment: Toss a fair coin once.

Sample Space: $\Omega = \{\text{Head, Tail}\}$



Probability basics...contd

Events

- Events are the subsets of the sample space
 - Simple Events: If an event consists of only one outcome, it is a simple event
- Union and Intersection: If A and B are events, then $A \cup B$ and $A \cap B$ are also events.
- Examples:
 - $A = \{\text{the outcome that the dice is even}\} = \{2, 4, 6\}$
 - $B = \{\text{at least 2 tails}\}$

Outcome: A result of a random experiment.

Sample Space: The set of all possible outcomes.

Event: A subset of the sample space.

369

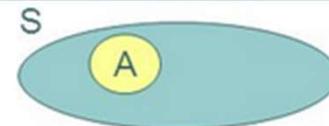


Probability

Let A be an event, then we denote

$P(A)$ the probability for A

It always hold that $0 \leq P(A) \leq 1$ $P(\emptyset) = 0$ $P(S) = 1$



Consider an experiment which has N **equally likely** outcomes, and let exactly n of these events correspond to the event A . Then

$$P(A) = \frac{n}{N} = \frac{\text{\# successful outcomes}}{\text{\# possible outcomes}}$$

Example:
Rolling a dice

$P(\text{even number})$

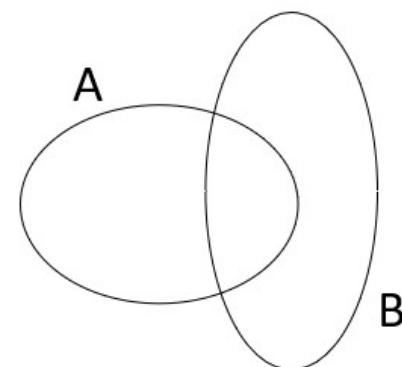
$$= \frac{3}{6} = \frac{1}{2}$$

Question: You roll a fair die. What is the probability of $E=\{1,5\}$?



Probability -Axioms

- Probability of event A is denoted by $P(A)$, $P(A) \in [0,1]$
 - Probability measure P is thus a real-valued set function defined on the set of events \mathcal{J} , $P: \mathcal{J} \rightarrow [0,1]$
- Properties:
 - (i) $0 \leq P(A) \leq 1$
 - (ii) $P(\emptyset) = 0$
 - (iii) $P(\Omega) = 1$
 - (iv) $P(A^c) = 1 - P(A)$
 - (v) $P(A \cup B) = P(A) + P(B) - P(A \cap B)$
 - (vi) $A \cap B = \emptyset \Rightarrow P(A \cup B) = P(A) + P(B)$
 - (vii) $\{B_i\}$ is a partition of $A \Rightarrow P(A) = \sum_i P(B_i)$
 - (viii) $A \subset B \Rightarrow P(A) \leq P(B)$





Random Variables

- A **random variable**, is a variable whose possible values are numerical outcomes of a **random experiment**.
 - Random variable X is a function from the sample space to the real numbers.
 - Denoted by Capital letter

I toss a coin five times. This is a random experiment and the sample space can be written as

$$S = \{TTTTT, TTTTH, \dots, HHHHH\}.$$

Note that here the sample space S has $2^5 = 32$ elements. Suppose that in this experiment, we are interested in the number of heads. We can define a random variable X whose value is the number of observed heads. The value of X will be one of 0, 1, 2, 3, 4 or 5 depending on the outcome of the random experiment.



Random Variables

- 1. **Discrete Random Variable** is one which may take on only a countable number of distinct values such as 0,1,2,3,4
 - X = sum of values on the roll of two dice,
 - X has to be either 2, 3, 4, ..., or 12.
 - X be the number of heads that result from the toss of 2 coins.
 - Here X can take values 0,1, or 2. X is a discrete random variable.

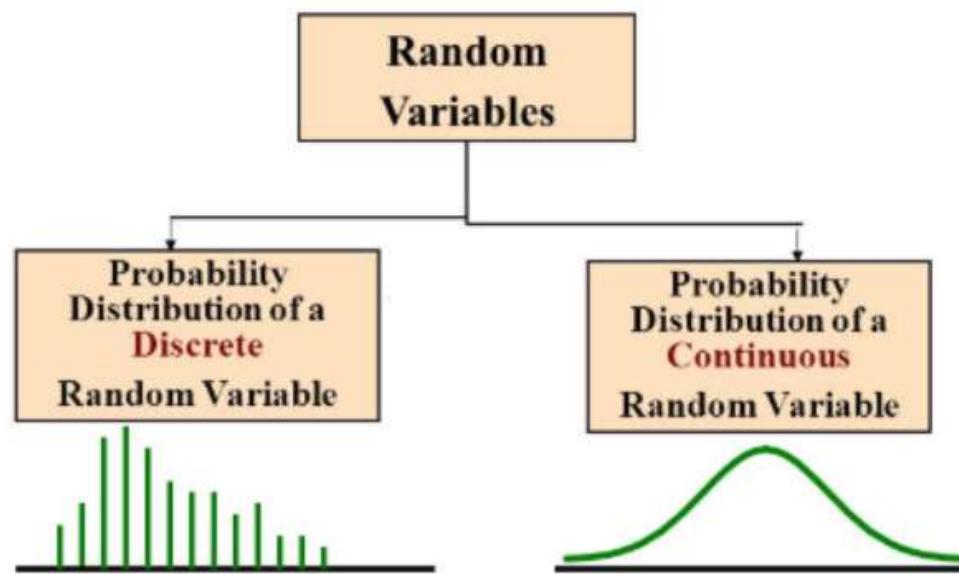
No of Heads(X)	Probability $P(X)$
0	0.25
1	0.5
2	0.25

Probabilities distribution of random variable X

- 2. **Continuous Random Variable** Continuous Data can take any value within a range.

- Example, List all the real numbers between $[0,1]$

Random Variables





Probability Distribution of Discrete Random Variable

- **Discrete Probability Distribution:** The mathematical definition of a discrete probability function, $p(x)$, is a function that satisfies the following properties. This is referred as **Probability Mass Function**.

Let X be a discrete random variable with range $R_X = \{x_1, x_2, x_3, \dots\}$ (finite or countably infinite). The function

$$P_X(x_k) = P(X = x_k), \text{ for } k = 1, 2, 3, \dots,$$

is called the *probability mass function (PMF)* of X .

- Probability mass function (pmf). $P(X=x_i)$
- Simple facts about pmf

1. The probability that x can take a specific value is $p(x)$, i.e.

$$P[X=x]=p(x)$$

2. $p(x)$ is non-negative for all real x .

3. The sum of $p(x)$ over all possible values of x is 1, i.e.

$$\sum_x P(x) = 1$$



Example

I toss a fair coin twice, and let X be defined as the number of heads I observe. Find the range of X , R_X , as well as its probability mass function P_X .

Solution

Here, our sample space is given by

$$S = \{HH, HT, TH, TT\}.$$

The number of heads will be 0, 1 or 2. Thus

$$R_X = \{0, 1, 2\}.$$

Since this is a finite (and thus a countable) set, the random variable X is a discrete random variable. Next, we need to find PMF of X . The PMF is defined as

$$P_X(k) = P(X = k) \text{ for } k = 0, 1, 2.$$

We have

$$P_X(0) = P(X = 0) = P(TT) = \frac{1}{4},$$

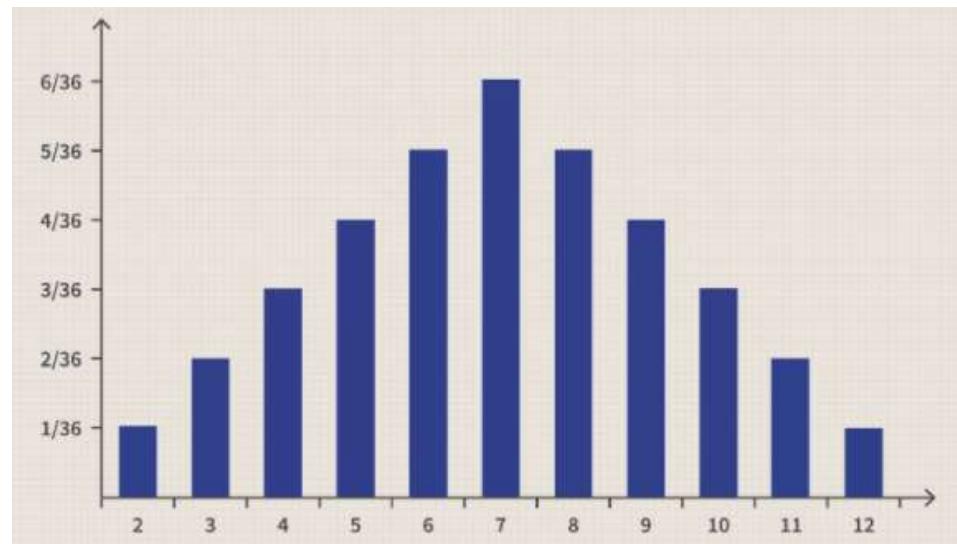
$$P_X(1) = P(X = 1) = P(\{HT, TH\}) = \frac{1}{4} + \frac{1}{4} = \frac{1}{2},$$

$$P_X(2) = P(X = 2) = P(HH) = \frac{1}{4}.$$



Probability Distribution - Example

- Sum of 2 dice





Probability of Continuous Random Variable

- **Continuous Probability Distribution:** The mathematical definition of a continuous probability function, $f(x)$, is a function that satisfies the following properties. This is referred as **Probability Density Function**.
- Simple facts about pdf
 1. The probability that x is in between two points a and b is $P(x)$
$$P[a \leq x \leq b] = \int_a^b f(x)dx$$
 2. $f(x)$ is non-negative for all real x .
 3. The sum of $p(x)$ over all possible values of x is 1, i.e.
$$\int_{-\infty}^{\infty} f(x)dx = 1$$

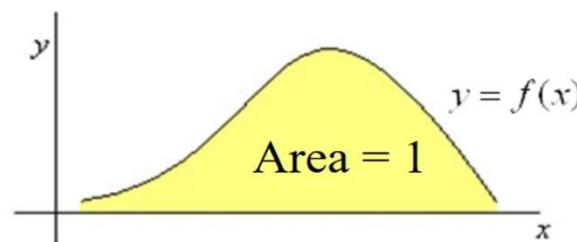


Probability Density Function

Probability Density Function

For $f(x)$ to be a pdf

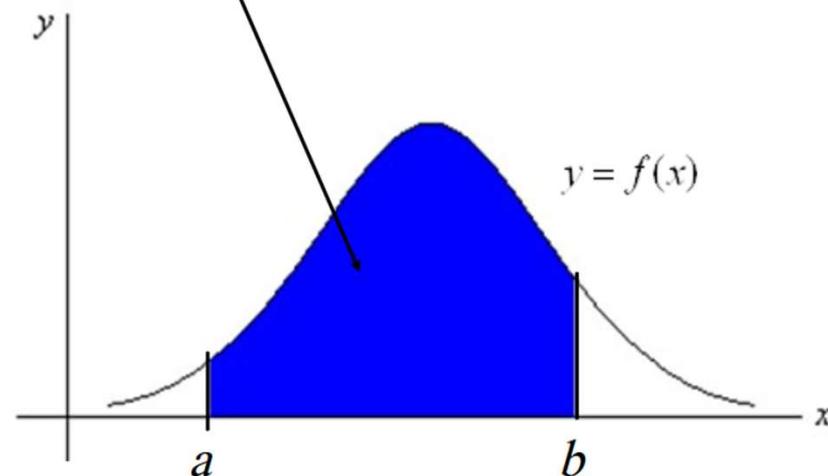
1. $f(x) > 0$ for all values of x .
2. The area of the region between the graph of f and the x -axis is equal to 1.



Probability Density Function

Probability Density Function

$P(a \leq X \leq b)$ is given by the area of the shaded region.



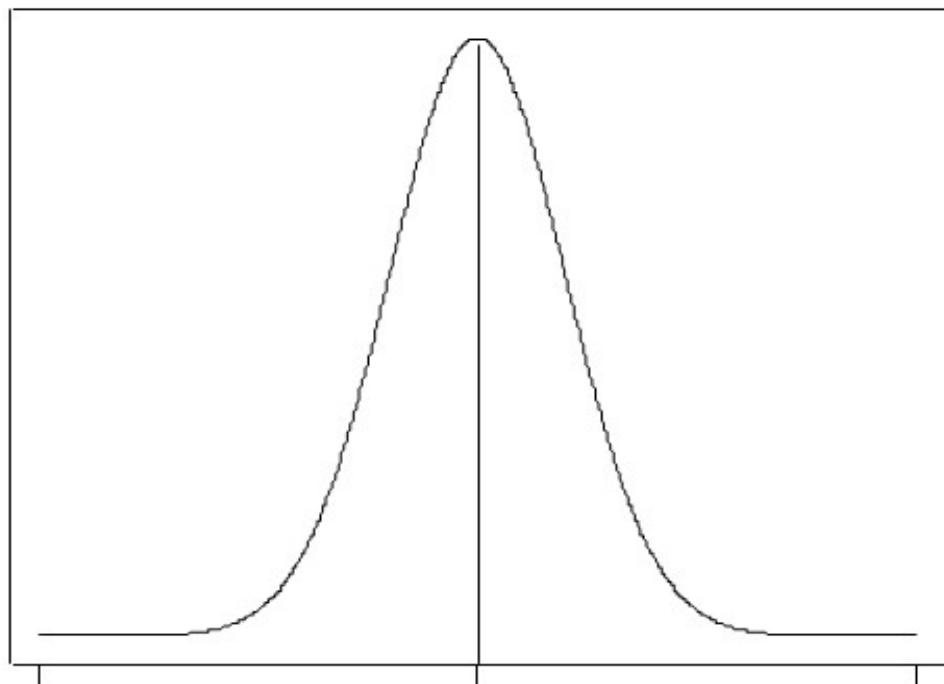


Gaussian Distribution

- The normal curve is *bell-shaped* and has a single peak at the exact center of the distribution.
- The arithmetic mean, median, and mode of the distribution are equal and located at the peak.
- Half the area under the curve is above the peak, and the other half is below it.
- The normal distribution is symmetrical about its mean.
- The normal distribution is asymptotic - the curve gets closer and closer to the x-axis but never actually touches it.
- Unimodal-a probability distribution is said to be normal if the mean, median, and mode coincide at a single point



Gaussian Distribution



Mean, median, and mode are equal

Normal curve is symmetrical

Theoretically, curve extends to infinity

MATHEMATICAL FUNCTION (pdf)

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \cdot e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2}$$

Note constants:

$\pi=3.14159$

$e=2.71828$

This is a bell shaped curve with different centers and spreads depending on μ and σ



Gaussian Distribution- Parameter

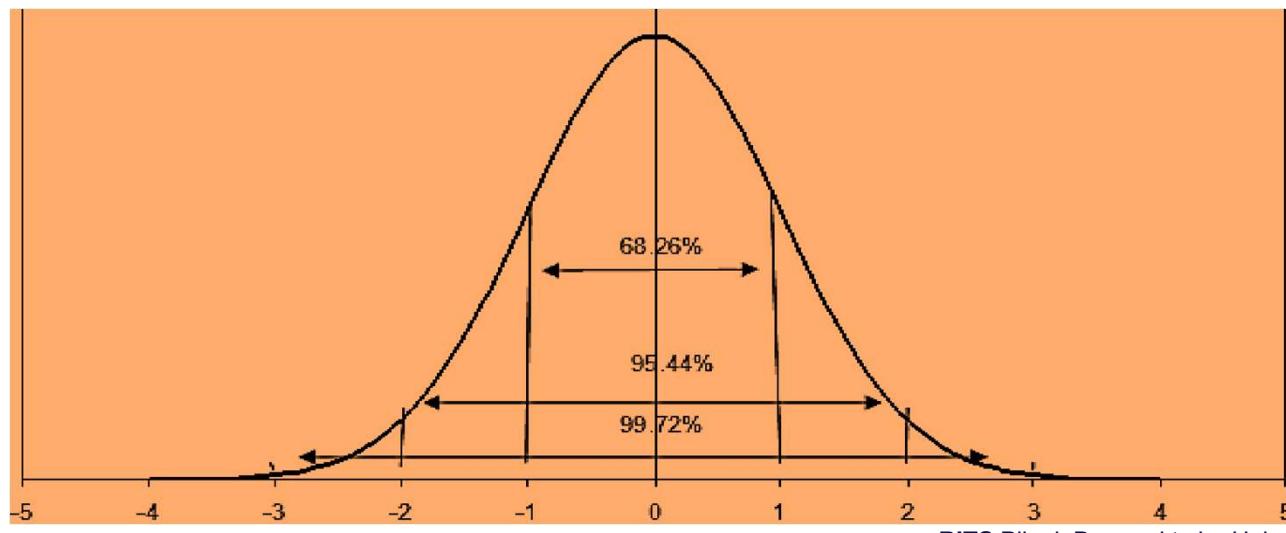
- The normal distribution can be completely specified by two parameters:

1. Mean
2. Standard deviation

Properties of normal curve

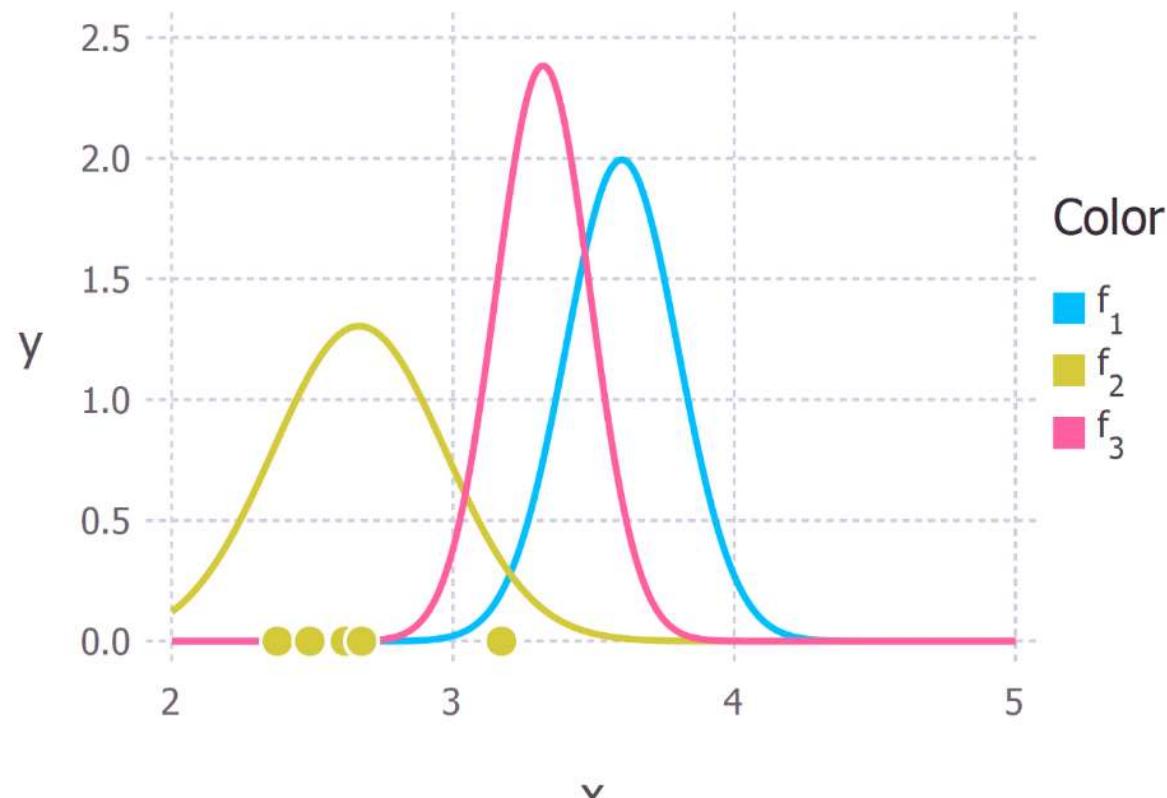
No matter what the value of μ and σ are, area under normal curve remain in certain fixed proportions within a specified number of standard deviation on either side of μ . For example the interval

- $\mu \pm \sigma$ will always contain 68.26%
- $\mu \pm 2\sigma$ will always contain 95.44%
- $\mu \pm 3\sigma$ will always contain 99.73%





Gaussian Distribution





Joint Probability

- **Joint probability** is the probability of two events happening together. The two events are usually designated *event A* and *event B*. In probability terminology, it can be written as:
- $P(X \text{ and } Y)$ or $P(A \cap B)$ or $P(X, Y)$
- **Example:** The probability that a card is a five and black, $p(\text{five and black}) = 2/52 = 1/26$, (There are two black fives in a deck of 52 cards, the five of spades and the five of clubs)



Conditional Probability

- Conditional Probability is a measure of the probability of an event given that (by assumption, presumption, assertion or evidence) another event has already occurred. If the event of interest is A and the event B is known or assumed to have occurred.
- This probability is written $P(A|B)$, notation for the *probability of A given B*.

$$P(A | B) = \frac{P(A \cap B)}{P(B)}$$

- Example: Let a pair of dice is thrown. If the sum of the numbers on the dice is 7, Find the probability that at least one of the dice shows 2.



Independence

- Two events A and B are independent if and only if $P(A \cap B) = P(A)P(B)$.
- if two events A and B are independent and $P(B) \neq 0$, then $P(A|B) = P(A)$.

$$\begin{aligned} P(A|B) &= \frac{P(A \cap B)}{P(B)} \\ &= \frac{P(A)P(B)}{P(B)} \\ &= P(A). \end{aligned}$$

- In general, for n events A_1, A_2, \dots, A_n to be independent we must have

$P(A_i \cap A_j) = P(A_i)P(A_j)$, for all distinct $i, j \in \{1, 2, \dots, n\}$;

$P(A_i \cap A_j \cap A_k) = P(A_i)P(A_j)P(A_k)$, for all distinct $i, j, k \in \{1, 2, \dots, n\}$;

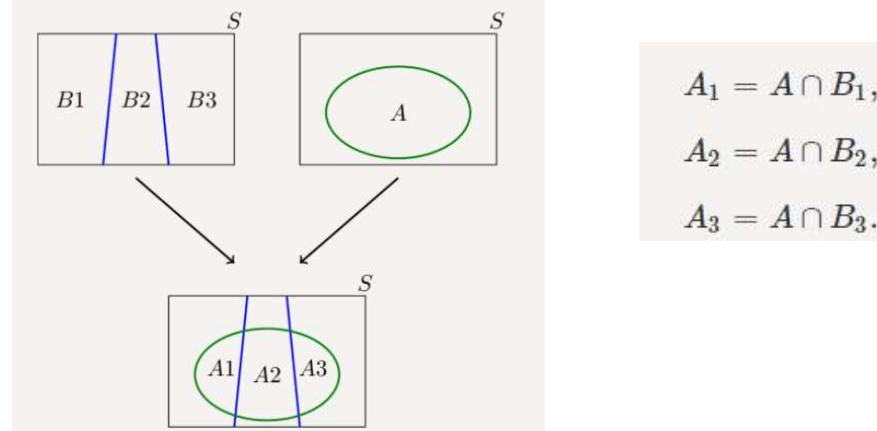
- **Example:** If a dice is thrown twice, What is the probability that the first throw results in a number greater than 4 and the second throw results in a number less than 3.

Law of Total Probability

If B_1, B_2, B_3, \dots is a partition of the sample space S , then for any event A we have

$$P(A) = \sum_i P(A \cap B_i) = \sum_i P(A|B_i)P(B_i).$$

$$P(A) = P(A_1) + P(A_2) + P(A_3).$$





Law of Total Probability

I have three bags that each contain 100 marbles:

- Bag 1 has 75 red and 25 blue marbles;
- Bag 2 has 60 red and 40 blue marbles;
- Bag 3 has 45 red and 55 blue marbles.

I choose one of the bags at random and then pick a marble from the chosen bag, also at random. What is the probability that the chosen marble is red?

Solution

Let R be the event that the chosen marble is red. Let B_i be the event that I choose Bag i . We already know that

$$P(R|B_1) = 0.75,$$

$$P(R|B_2) = 0.60,$$

$$P(R|B_3) = 0.45$$

We choose our partition as B_1, B_2, B_3 . Note that this is a valid partition because, firstly, the B_i 's are disjoint (only one of them can happen), and secondly, because their union is the entire sample space as one the bags will be chosen for sure, i.e., $P(B_1 \cup B_2 \cup B_3) = 1$. Using the law of total probability, we can write

$$\begin{aligned} P(R) &= P(R|B_1)P(B_1) + P(R|B_2)P(B_2) + P(R|B_3)P(B_3) \\ &= (0.75)\frac{1}{3} + (0.60)\frac{1}{3} + (0.45)\frac{1}{3} \\ &= 0.60 \end{aligned}$$



Bayes Rule

From the definition of conditional distribution

$$P(A|B)P(B) = P(A, B) = P(B|A)P(A)$$

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Bayes' Rule

- For any two events A and B , where $P(A) \neq 0$, we have

$$P(B|A) = \frac{P(A|B)P(B)}{P(A)}.$$

- If B_1, B_2, B_3, \dots form a partition of the sample space S , and A is any event with $P(A) \neq 0$, we have

$$P(B_j|A) = \frac{P(A|B_j)P(B_j)}{\sum_i P(A|B_i)P(B_i)}.$$



Example of Bayes Theorem

Suppose the fraction of undergraduate students who play sport is 15% and the fraction of graduate students who play sports is 23%. If one-fifth of the college students are graduate students and the rest are undergraduates, what is the probability that a student who plays a sport is a graduate student?



Example of Bayes Theorem

Tid	Home Owner	Marital Status	Annual Income	Defaulted Borrower	class
1	Yes	Single	125K	No	binary
2	No	Married	100K	No	categorical
3	No	Single	70K	No	continuous
4	Yes	Married	120K	No	binary
5	No	Divorced	95K	Yes	categorical
6	No	Married	60K	No	continuous
7	Yes	Divorced	220K	No	binary
8	No	Single	85K	Yes	categorical
9	No	Married	75K	No	continuous
10	No	Single	90K	Yes	binary

Training set for predicting the loan default problem.

Classify the record:

X = (Home Owner = No, Marital Status = "Married", Annual Income = \$120k)



Bayesian Classifiers

Let the attributes X_1, X_2, \dots, X_n and class labels Y_1, Y_2, \dots, Y_m be random variables

Given a record with attributes (X_1, X_2, \dots, X_n)

- Goal is to predict class $Y=Y_k$
- We want to find the value of Y that maximizes $P(Y| X_1, X_2, \dots, X_n)$

Can we estimate $P(Y| X_1, X_2, \dots, X_n)$ directly from data?



Bayesian Classifiers

- Compute the posterior probability $P(Y| X_1, X_2, \dots, X_n)$ for all values of Y using the Bayes theorem

$$P(Y| X_1, X_2, \dots, X_n) = \frac{P(X_1, X_2, \dots, X_n | Y)P(Y)}{P(X_1, X_2, \dots, X_n)}$$

- Choose value of Class that maximizes $P(Y| X_1, X_2, \dots, X_n)$
- Equivalent to choosing value of Y that maximizes $P(X_1, X_2, \dots, X_n | Y)P(Y)$

How to estimate $P(X_1, X_2, \dots, X_n | Y)$?



Chain Rule for Conditional Probability

$$P(A \cap B) = P(A)P(B|A) = P(B)P(A|B)$$

Now we can extend this formula to three or more events:

$$P(A \cap B \cap C) = P(A \cap (B \cap C)) = P(A)P(B \cap C|A)$$

$$P(B \cap C) = P(B)P(C|B).$$

Conditioning both sides on A , we obtain

$$P(B \cap C|A) = P(B|A)P(C|A, B)$$

Combining Equation 1.6 and 1.7 we obtain the following chain rule:

$$P(A \cap B \cap C) = P(A)P(B|A)P(C|A, B).$$

Chain rule for conditional probability:

$$P(A_1 \cap A_2 \cap \cdots \cap A_n) = P(A_1)P(A_2|A_1)P(A_3|A_2, A_1) \cdots P(A_n|A_{n-1}A_{n-2} \cdots A_1)$$

https://www.probabilitycourse.com/chapter1/1_4_0_conditional_probability.php



Conditional independence

Two events A and B are **conditionally independent** given an event C with $P(C) > 0$

$$P(A \cap B|C) = P(A|C)P(B|C)$$

Recall that from the definition of conditional probability,

$$P(A|B) = \frac{P(A \cap B)}{P(B)},$$

if $P(B) > 0$. By conditioning on C , we obtain

$$P(A|B, C) = \frac{P(A \cap B|C)}{P(B|C)}$$

if $P(B|C), P(C) \neq 0$. If A and B are conditionally independent given C , we obtain

$$\begin{aligned} P(A|B, C) &= \frac{P(A \cap B|C)}{P(B|C)} \\ &= \frac{P(A|C)P(B|C)}{P(B|C)} \\ &= P(A|C). \end{aligned}$$

Thus, if A and B are conditionally independent given C , then

$$P(A|B, C) = P(A|C)$$





Applying conditional independence

Naïve Bayes assumes X_i are conditionally independent given Y

$$\text{e.g., } P(X_1|X_2, Y) = P(X_1|Y)$$

$$P(X_1, X_2|Y) = P(X_1|X_2, Y)P(X_2|Y) = P(X_1|Y)P(X_2|Y)$$

$$\text{General form: } P(X_1, \dots, X_n|Y) = \prod_{j=1}^n P(X_j|Y)$$

How many parameters to describe $P(X_1, \dots, X_n|Y)$?



Naïve Bayes Independence assumption

Assumption:

$$P(X_1, \dots, X_n | Y) = \prod_{j=1}^n P(X_j | Y)$$

i.e., X_i and X_j are conditionally independent given Y for $i \neq j$



Naïve Bayes Classifier

Assume independence among attributes X_i when class is given:

- $P(X_1, X_2, \dots, X_d | Y_j) = P(X_1 | Y_j) P(X_2 | Y_j) \dots P(X_d | Y_j)$
- Now we can estimate $P(X_i | Y_j)$ for all X_i and Y_j combinations from the training data
- New point is classified to Y_j if $P(Y_j) \prod P(X_i | Y_j)$ is maximal.

Slide adopted from “Introduction to Data mining” Vipin Kumar



Class Conditional Probabilities

To compute, $P(x_k|C_i)$

- A_k is categorical:

the number of tuples of class C_i in D having the value x_k for A_k

$$P(x_k|C_i) = \frac{\text{the number of tuples of class } C_i \text{ in } D.}{\text{the number of tuples of class } C_i \text{ in } D.}$$

- A_k is continuous:

A continuous-valued attribute is typically assumed to have a Gaussian distribution with a mean μ and standard deviation σ

$$g(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}},$$

$$P(x_k|C_i) = g(x_k, \mu_{C_i}, \sigma_{C_i}).$$

How to Estimate Probabilities from Data?

Given a Test Record:

$$X = (\text{Refund} = \text{No}, \text{Married}, \text{Income} = 120\text{K})$$

Tid	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125	No
2	No	Married	100	No
3	No	Single	70	No
4	Yes	Married	120	No
5	No	Divorced	95	Yes
6	No	Married	60	No
7	Yes	Divorced	220	No

- Class: $P(C) = N_c/N$
 - e.g., $P(\text{No}) = 7/10$, $P(\text{Yes}) = 3/10$
- For discrete attributes:
$$P(A_i | C_k) = |A_{ik}| / N_{ck}$$
- where $|A_{ik}|$ is number of instances having attribute A_i and belongs to class C_k
- Examples:
 $P(\text{Status}=\text{Married}|\text{No}) = 4/7$
 $P(\text{Refund}=\text{Yes}|\text{Yes})=0$

How to Estimate Probabilities from Data?

- For continuous attributes:
 - Discretize the range into bins
 - one ordinal attribute per bin
 - Too many bins – training records are too few to provide reliable probability for each interval
 - Too few bins – some intervals may aggregate from different classes & we may miss the correct decision boundary
 - Probability density estimation:
 - Assume attribute follows a normal distribution
 - Use data to estimate parameters of distribution (e.g., mean and standard deviation)
 - Once probability distribution is known, can use it to estimate the conditional probability $P(A_i|C_k)$

How to Estimate Probabilities from Data?

Tid	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	110K	No
8	No	Single	85K	Yes
9	No	Married	75K	No

- Normal distribution:

$$P(A_i | c_j) = \frac{1}{\sqrt{2\pi\sigma_{ij}^2}} e^{-\frac{(A_i - \mu_{ij})^2}{2\sigma_{ij}^2}}$$

- One for each (A_i, c_i) pair
- For (Income, Class=No):
 - If Class=No
 - sample mean = 110
 - sample variance = 2975

$$P(\text{Income} = 120 | \text{No}) = \frac{1}{\sqrt{2\pi}(54.54)} e^{-\frac{(120-110)^2}{2(2975)}} = 0.0072$$

Example of Naïve Bayes Classifier

Given a Test Record:

$$X = (\text{Refund} = \text{No}, \text{Married}, \text{Income} = 120\text{K})$$

Tid	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes

$$\begin{aligned}\square P(X|\text{Class}=\text{No}) &= P(\text{Refund}=\text{No}|\text{Class}=\text{No}) \\ &\quad \times P(\text{Married}|\text{Class}=\text{No}) \\ &\quad \times P(\text{Income}=120\text{K}|\text{Class}=\text{No}) \\ &= 4/7 \times 4/7 \times 0.0069 = 0.0023\end{aligned}$$

$$\begin{aligned}\square P(X|\text{Class}=\text{Yes}) &= P(\text{Refund}=\text{No}|\text{Class}=\text{Yes}) \\ &\quad \times P(\text{Married}|\text{Class}=\text{Yes}) \\ &\quad \times P(\text{Income}=120\text{K}|\text{Class}=\text{Yes}) \\ &= 1 \times 0 \times 1.2 \times 10^{-9} = 0\end{aligned}$$

Since $P(X|\text{No})P(\text{No}) > P(X|\text{Yes})P(\text{Yes})$

Therefore $P(\text{No}|X) > P(\text{Yes}|X)$
 $\Rightarrow \text{Class} = \text{No}$

Naive Bayes Classifier

- If one of the conditional probability is zero, then the entire expression becomes zero
- Probability estimation is done with Laplacian correction:

$$\text{Original: } P(A_i | C) = \frac{N_{ic}}{N_c}$$

$$\text{Laplace: } P(A_i | C) = \frac{N_{ic} + 1}{N_c + c}$$

C: No. of distinct
values of an
attribute

Example of Naïve Bayes Classifier

Given a Test Record:

$X = (\text{Refund} = \text{Yes}, \text{Status} = \text{Single}, \text{Income} = 80K)$

naive Bayes Classifier:

$$\begin{aligned} P(\text{Refund}=\text{Yes}|\text{No}) &= 4/9 \\ P(\text{Refund}=\text{No}|\text{No}) &= 5/9 \\ P(\text{Refund}=\text{Yes}|\text{Yes}) &= 1/5 \\ P(\text{Refund}=\text{No}|\text{Yes}) &= 4/5 \end{aligned}$$

$$\begin{aligned} P(\text{Marital Status}=\text{Single}|\text{No}) &= 3/10 \\ P(\text{Marital Status}=\text{Divorced}|\text{No}) &= 2/10 \\ P(\text{Marital Status}=\text{Married}|\text{No}) &= 5/10 \\ P(\text{Marital Status}=\text{Single}|\text{Yes}) &= 3/6 \\ P(\text{Marital Status}=\text{Divorced}|\text{Yes}) &= 2/6 \\ P(\text{Marital Status}=\text{Married}|\text{Yes}) &= 1/6 \end{aligned}$$

For taxable income:

If class=No: sample mean=110

sample variance=2975

If class=Yes: sample mean=90

sample variance=25

With Laplace Smoothing

- $P(X|\text{Class}=\text{No}) = P(\text{Refund}=\text{No}|\text{Class}=\text{No}) \times P(\text{Married}|\text{Class}=\text{No}) \times P(\text{Income}=120K|\text{Class}=\text{No})$
 $= 4/9 \times 3/10 \times 0.0062 = 0.00082$
 - $P(X|\text{Class}=\text{Yes}) = P(\text{Refund}=\text{No}|\text{Class}=\text{Yes}) \times P(\text{Married}|\text{Class}=\text{Yes}) \times P(\text{Income}=120K|\text{Class}=\text{Yes})$
 $= 1/5 \times 3/6 \times 0.01 = 0.001$
 - $P(\text{No}) = 0.7, P(\text{Yes}) = 0.3$
 - $P(X|\text{No})P(\text{No}) = 0.0005$
 - $P(X|\text{Yes})P(\text{Yes}) = 0.0003$
- $\Rightarrow \text{Class} = \text{No}$

Tid	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Example

- You are working in google.com where people disclose their salary information. As part of profile creation, the firm captures various important details about its members. Some of the samples are shown in the below table. Using this sample dataset, with the help of Bayesian classification technique, classify the following tuple either as “High”, or “Middle” or “Low” income bracket member.
- {SrNo = “7”, Age = “Above_40”, Gender = “M”, Occupation = “Software Engineer”}

Sr No	Age	Gender	Occupation	Income Bracket
1	Above_40	F	Software Engineer	High
2	Below_30	M	Marketing Executive	Middle
3	Between_31_to_40	M	Unemployed	Low
4	Below_30	M	Data Scientist	High
5	Between_31_to_40	F	Software Engineer	High
6	Below_30	F	Unemployed	Low



Naïve Bayes Generative Model

- Naïve Bayes classifier uses likelihood and prior probability to calculate conditional probability of the class
- Likelihood is based on joint probability, which is the core principle of probabilistic generative model
- Naïve Bayes simplifies the calculation of likelihood by the assumption of conditional independence among input parameters
- Each parameter's likelihood is determined using joint probability of the input parameter and the output label



Naïve Bayes – When to use

- When the training data is small
- When the features are conditionally independent (mostly)
- When we have a few missing data
- When we have large number of features with minimal data set
 - Ex: Text classification



Gaussian Naïve Bayes Algorithm

- A Gaussian Naive Bayes algorithm is a special type of NB algorithm.
- It's specifically used when the features have continuous values.
- It's also assumed that all the features are following a Gaussian distribution i.e, normal distribution.



MultinomialNB

<https://www.sicara.ai/blog/2018-02-28-naive-bayes-classification-sklearn>



Extra Reading

- Text Classification in NLP using Naïve Bayes
- <https://medium.com/@theflyingmantis/text-classification-in-nlp-naive-bayes-a606bf419f8c>
- More on Probability
- https://www.probabilitycourse.com/chapter1/1_0_0_introduction.php



References

- <https://towardsdatascience.com/probability-concepts-explained-bayesian-inference-for-parameter-estimation-90e8930e5348>
- <https://towardsdatascience.com/probability-concepts-explained-introduction-a7c0316de465>
- <https://medium.com/@theflyingmantis/text-classification-in-nlp-naive-bayes-a606bf419f8c>
- <https://www.youtube.com/watch?v=5Pck0Cqw-zc>
- <https://towardsdatascience.com/basic-probability-theory-and-statistics-3105ab637213>
- <https://nlp.stanford.edu/IR-book/html/htmledition/properties-of-naive-bayes-1.html>



Logistic Regression



BITS Pilani
Pilani Campus

Swarna Chaudhary
Assistant Professor
Swarna.Chaudhary@pilani.bits-pilani.ac.in



Text Book(s)

- | | |
|----|--|
| T1 | Christopher Bishop: Pattern Recognition and Machine Learning, Springer International Edition |
| T2 | Tom M. Mitchell: Machine Learning, The McGraw-Hill Companies, Inc.. |

These slides are prepared by the instructor, with grateful acknowledgement of Prof. Andrew Ng, Prof. Tom Mitchell and many others who made their course materials freely available online.



Topics to be covered

- Significance of Sigmoid function and finding its derivative
- Cross entropy error function for logistic regression and its optimal solution
- Decision boundary of logistic regression
- Overfitting of logistic regression and counter measures

Linear Regression versus logistic regression

- **Linear regression** is used for predicting the continuous dependent variable using a given set of independent features whereas **Logistic Regression** is used to predict the categorical.
- **Linear regression** is used to solve **regression** problems whereas **logistic regression** is used to solve classification problems

Linear Regression versus Logistic Regression

Classification requires discrete values:

$$y = 0 \text{ or } 1$$

For linear Regression output values:

$h_{\theta}(x)$ can be much > 1 or much < 0

Logistic Regression: $0 \leq h_{\theta}(x) \leq 1$

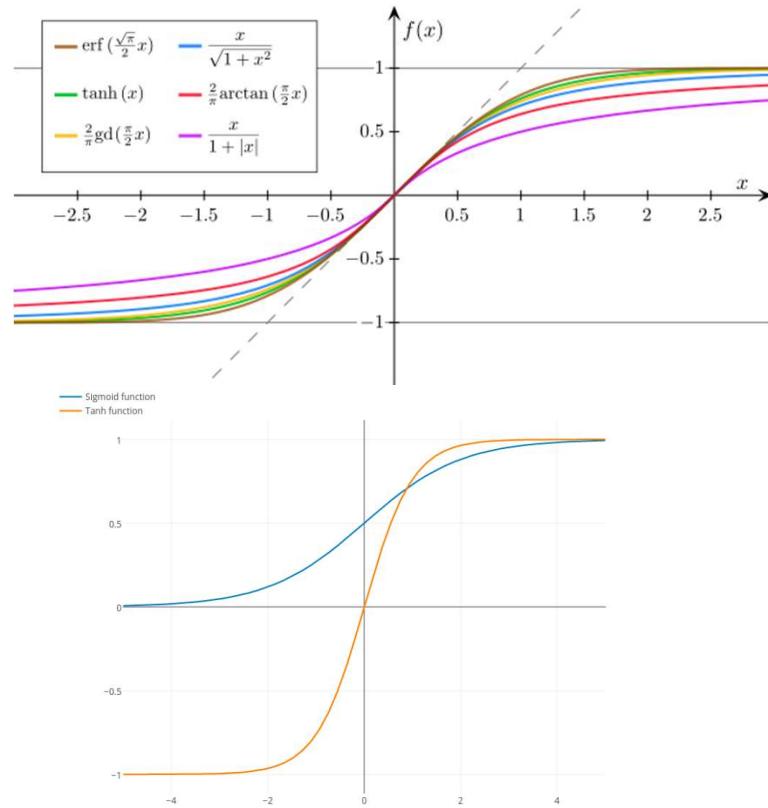
Sigmoid/Logistic Function

- Sigmoid/logistic function takes a real value as input and outputs another value between 0 and 1
- That framework is called logistic regression
 - Logistic: A special mathematical sigmoid function it uses
 - Regression: Combines a weight vector with observations to create an answer

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

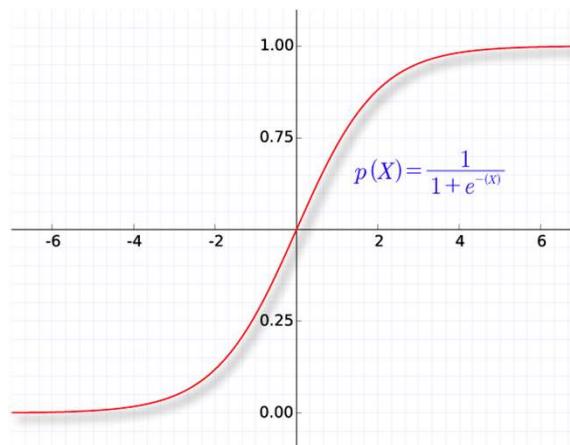
Sigmoid Function

- Any function that takes the shape of ‘S’ is called a sigmoid function
- Maps input horizon to a bounded, symmetrical output range
- Examples:
 - Logistic Function
 - Hyperbolic Tangent



Sigmoid Function

- Logistic function (a variant of sigmoid) bounds the output between 0 and 1
- Conditional probability of a random variable can be expressed as a sigmoid (logistic) function



Logistic Regression

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

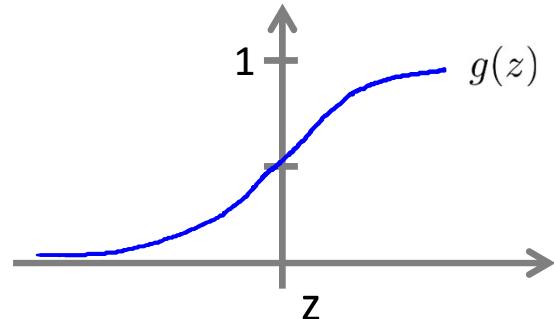
$$g(z) = \frac{1}{1+e^{-z}}$$

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

$$\theta^T x \geq 0$$

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

$$\theta^T x < 0$$



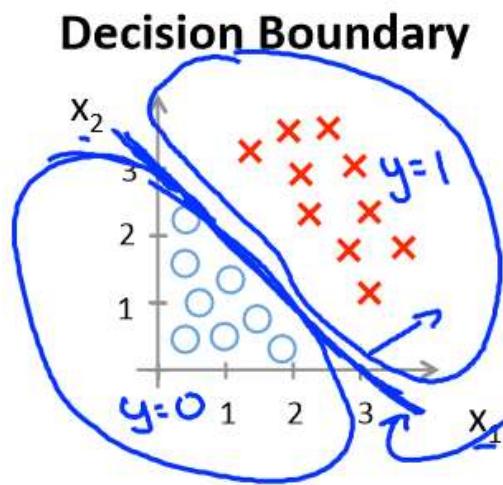
$$g(z) \geq 0.5$$

when $z \geq 0$

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

$$g(z) < 0.5$$

when $z < 0$



Predict " $y = 1$ " if $\underline{-3 + x_1 + x_2 \geq 0}$

$$\theta = \begin{bmatrix} -3 \\ 1 \\ 1 \end{bmatrix} \leftarrow$$

$$h_{\theta}(x) = g(\underline{\theta_0} + \underline{\theta_1 x_1} + \underline{\theta_2 x_2})$$

Decision boundary

$$x_1, x_2$$

$$h_{\theta}(x) = 0.5$$

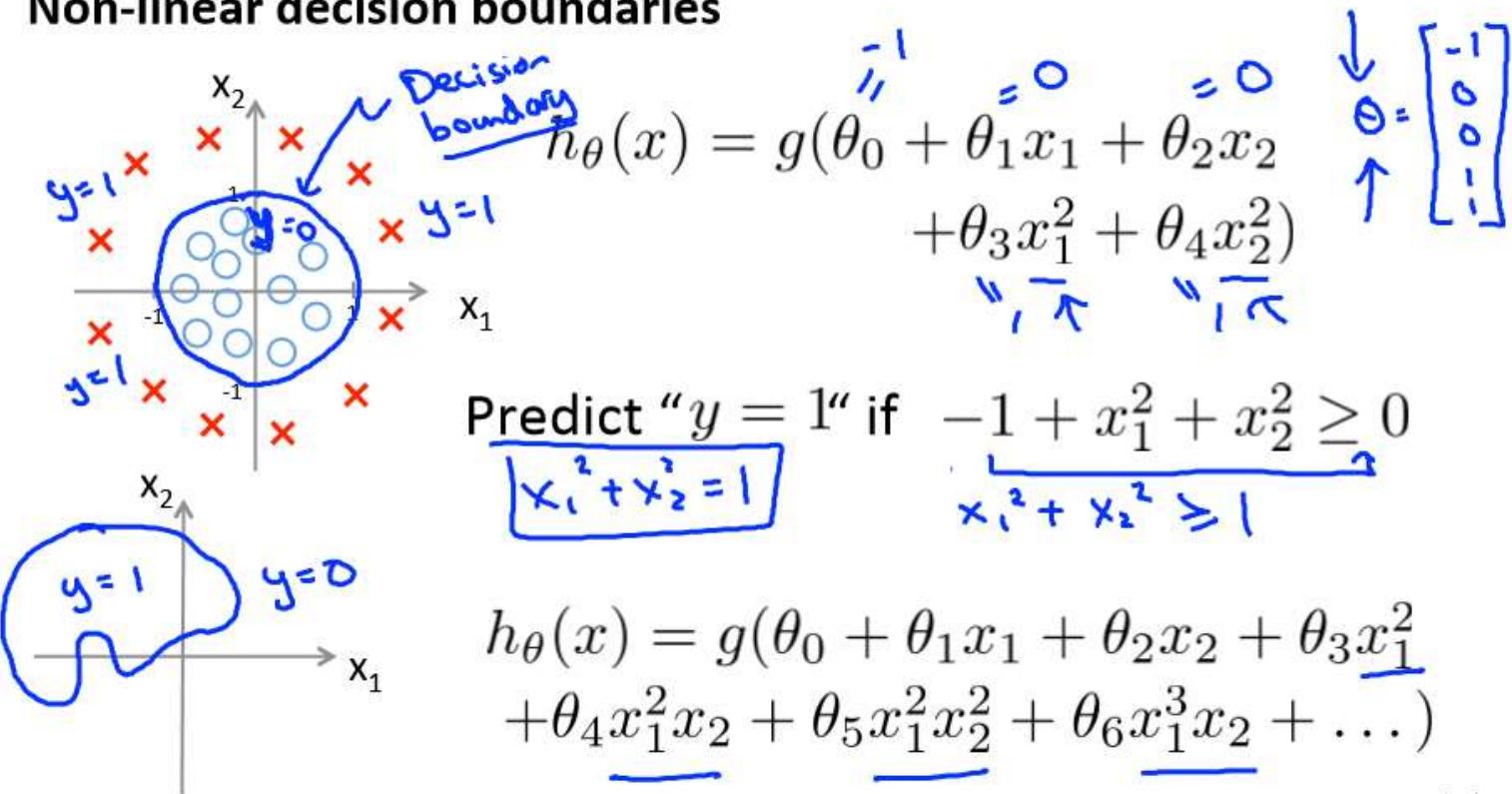
$$x_1 + x_2 = 3$$

$$x_1 + x_2 \geq 3$$

$$x_1 + x_2 < 3$$

$$y = 0$$

Non-linear decision boundaries



Andrew Ng

Learning model parameters

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

- m examples

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

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

- How to choose parameters (feature weights) θ ?
- To learn parameters that make predicted value as close to the true value.
 - We need a cost function and objective to minimize cost function to find optimal values of parameters

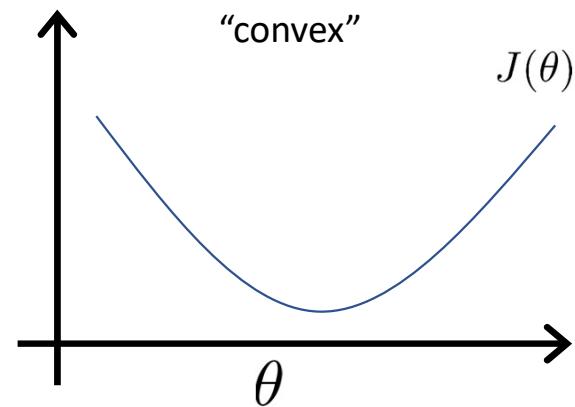
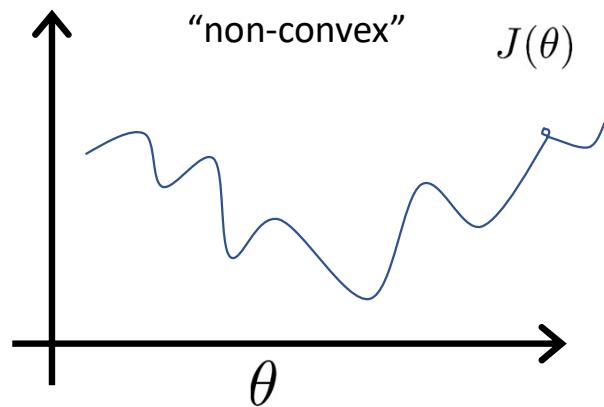
Error (Cost) Function

- Our prediction function is non-linear (due to sigmoid transform)
- Squaring this prediction as we do in MSE results in a non-convex function with many local minima.
- If our cost function has many local minimums, gradient descent may not find the optimal global minimum.
- So instead of Mean Squared Error, we use a error/ cost function called [Cross-Entropy](#), also known as Log Loss.

MSE Cost Function

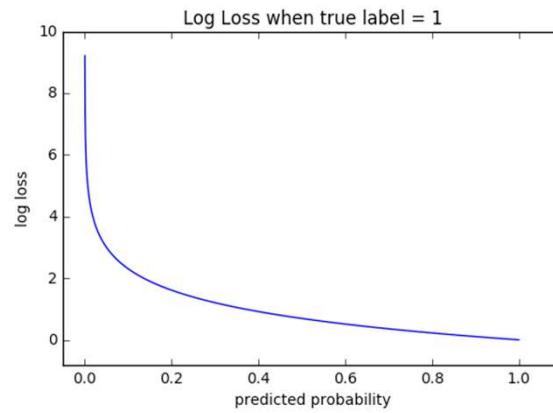
Linear regression: $J(\theta) = \frac{1}{m} \sum_{i=1}^m \frac{1}{2} (h_\theta(x^{(i)}) - y^{(i)})^2$

$$\text{Cost}(h_\theta(x^{(i)}), y^{(i)}) = \frac{1}{2} (h_\theta(x^{(i)}) - y^{(i)})^2$$



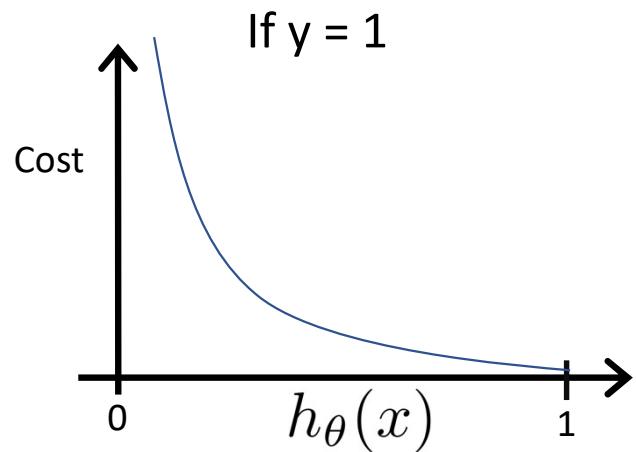
Cross Entropy

- Cross-entropy loss, or log loss, measures the performance of a classification model whose output is a probability value between 0 and 1.
- Cross-entropy loss increases as the predicted probability diverges from the actual label.
- So predicting a probability of .012 when the actual observation label is 1 would be bad and result in a high loss value.
- A perfect model would have a log loss of 0.
- Cross-entropy loss can be divided into two separate cost functions:
one for $y=1$ and
one for $y=0$.



Logistic regression cost function (cross entropy)

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

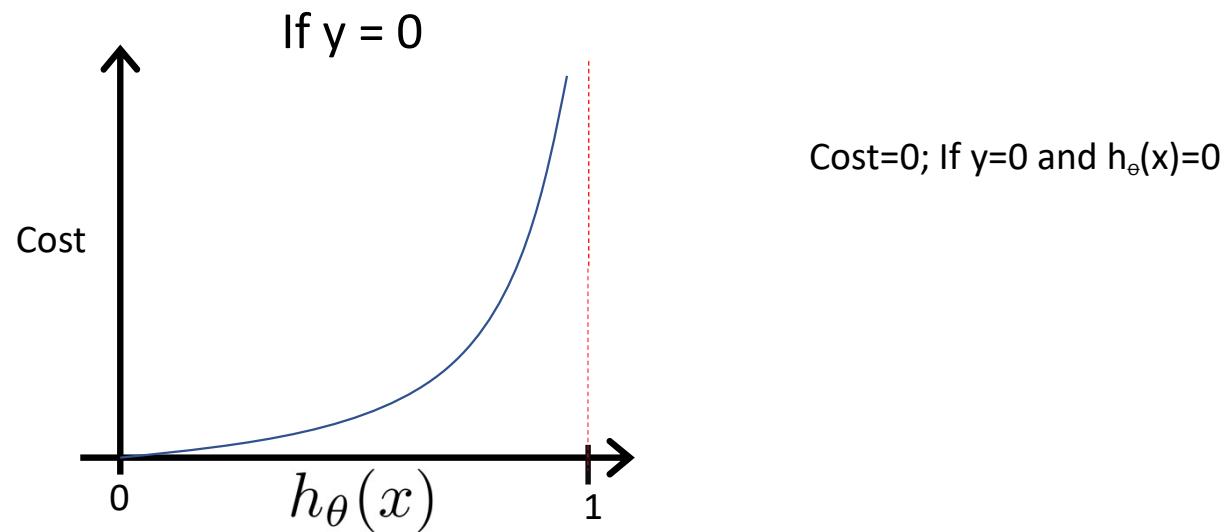


Cost = 0 if $y = 1, h_\theta(x) = 1$
But as $h_\theta(x) \rightarrow 0$
 $Cost \rightarrow \infty$

Captures intuition that if $h_\theta(x) = 0$,
(predict $P(y = 1|x; \theta) = 0$), but $y = 1$,
we'll penalize learning algorithm by a very
large cost.

Logistic regression cost function

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



Cost function

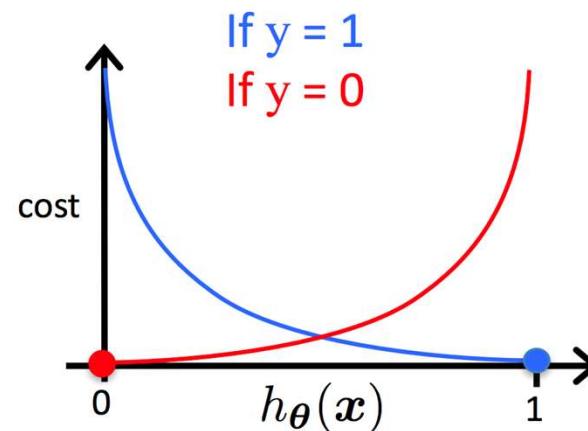
$$\begin{aligned} J(\theta) &= \frac{1}{m} \sum_{i=1}^m \text{Cost}(h_\theta(x^{(i)}), y^{(i)}) \\ &= -\frac{1}{m} \left[\sum_{i=1}^m y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_\theta(x^{(i)})) \right] \end{aligned}$$

To fit parameters : θ [Apply Gradient Descent Algorithm](#)

$$\min_{\theta} J(\theta)$$

To make a prediction given new : x

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



Gradient Descent Algorithm

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

Goal: $\min_{\theta} J(\theta)$

Repeat

{

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

}

$$\frac{\partial}{\partial \theta_j} J(\theta) = \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

Gradient Descent Algorithm

Linear Regression

Repeat {

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

$$h_\theta(x) = \theta^\top x$$

}

Logistic Regression

Repeat {

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

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

}

$$h_\theta(x) = \frac{1}{1 + e^{-(\theta_0 + \theta_1 CGPA + \theta_2 IQ)}}$$

Slide credit: Andrew Ng

Derivative of sigmoid function

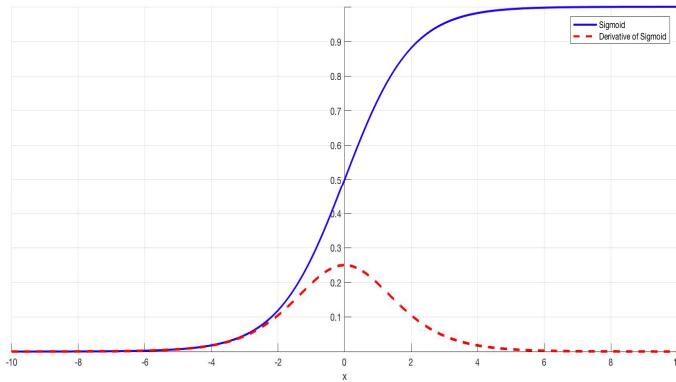
- Maximum likelihood to determine the parameters of the logistic regression model.
- To do this, we shall make use of the derivative of the logistic sigmoid function
- Use any algorithm like the gradient descent algorithm to minimize cost function by using derivative

<https://towardsdatascience.com/derivative-of-the-sigmoid-function-536880cf918e>

Sigmoid Function

$$\sigma(x) = \frac{1}{1 + e^{-x}}.$$

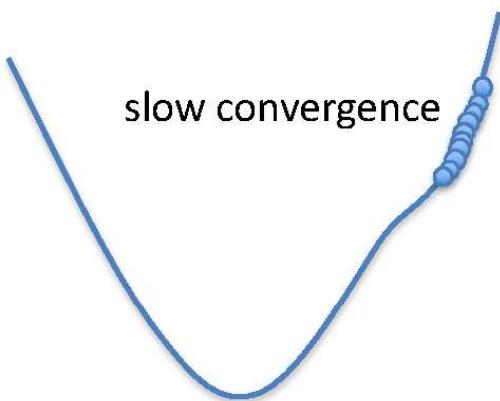
$$\begin{aligned}\frac{d}{dx} \sigma(x) &= \frac{d}{dx} \left[\frac{1}{1 + e^{-x}} \right] \\&= \frac{d}{dx} (1 + e^{-x})^{-1} \\&= -(1 + e^{-x})^{-2} (-e^{-x}) \\&= \frac{e^{-x}}{(1 + e^{-x})^2} \\&= \frac{1}{1 + e^{-x}} \cdot \frac{e^{-x}}{1 + e^{-x}} \\&= \frac{1}{1 + e^{-x}} \cdot \frac{(1 + e^{-x}) - 1}{1 + e^{-x}} \\&= \frac{1}{1 + e^{-x}} \cdot \left(\frac{1 + e^{-x}}{1 + e^{-x}} - \frac{1}{1 + e^{-x}} \right) \\&= \frac{1}{1 + e^{-x}} \cdot \left(1 - \frac{1}{1 + e^{-x}} \right) \\&= \sigma(x) \cdot (1 - \sigma(x))\end{aligned}$$



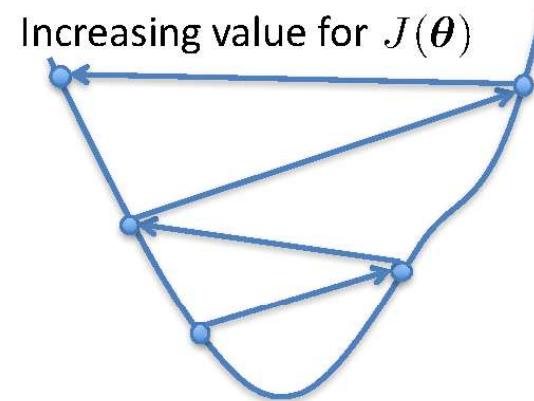
Derivative of sigmoid function is significant in the sense that it represents a bell shaped probability distribution function of the data

Choosing α

α too small



α too large



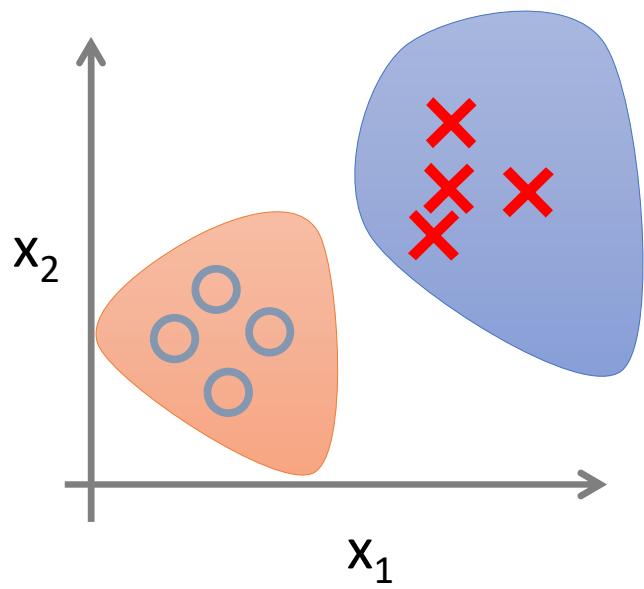
- May overshoot the minimum
- May fail to converge
- May even diverge

To see if gradient descent is working, print out $J(\theta)$ each iteration

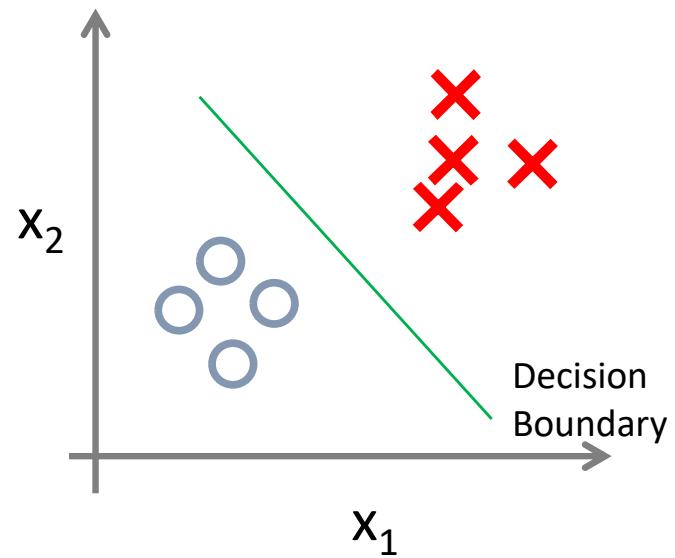
- The value should decrease at each iteration
- If it doesn't, adjust α

Probabilistic Generative Model versus Probabilistic Discriminative Model

Generative:



Discriminative:



Probabilistic Generative Model versus Probabilistic Discriminative Model

Generative	Discriminative
Ex: Naïve Bayes	Ex: Logistic Regression
Estimate $P(Y)$ and $P(X Y)$	Finds class label directly $P(Y X)$
Prediction $\hat{y} = \text{argmax}_y P(Y = y)P(X = x Y = y)$	Prediction $\hat{y} = P(Y = y X = x)$

Naïve Bayes versus Logistic Regression

- Naïve Bayes are Generative Models which Logistic Regression are Discriminative Models
- Naïve Bayes easy to construct
- Naïve Bayes better on smaller datasets
- Naive Bayes also assumes that the features are conditionally independent. Real data sets are never perfectly independent
- When the training size reaches infinity, logistic regression performs better than the generative model Naive Bayes.
 - Optional reading by Ng and Jordan has proofs and experiments
- Logistic regression allows arbitrary features

Logistic regression (Classification)

- **Model**

$$h_{\theta}(x) = P(Y = 1|X_1, X_2, \dots, X_n) = \frac{1}{1+e^{-\theta^T x}}$$

- **Cost function**

$$J(\theta) = \frac{1}{m} \sum_{i=1}^m \text{Cost}(h_{\theta}(x^{(i)}), y^{(i)}) \quad \text{Cost}(h_{\theta}(x), y) = \begin{cases} -\log(h_{\theta}(x)) & \text{if } y = 1 \\ -\log(1 - h_{\theta}(x)) & \text{if } y = 0 \end{cases}$$

- **Learning**

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

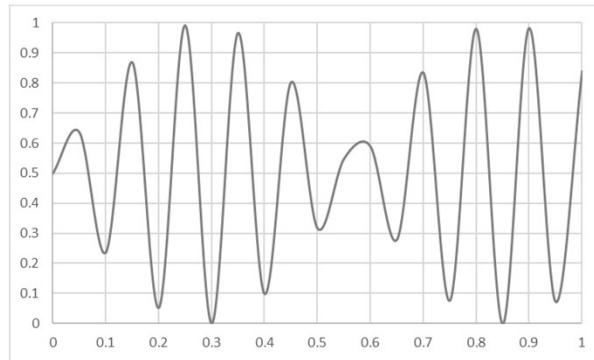
- **Inference**

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

Overfitting vs Underfitting

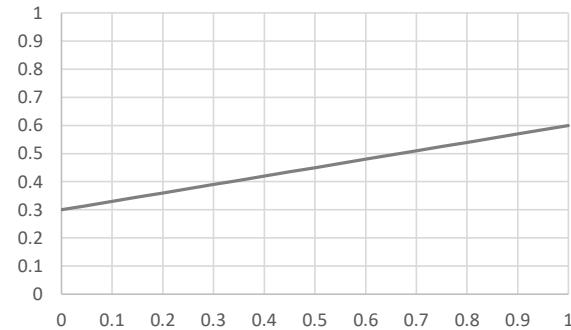
Overfitting

- Fitting the data too well
 - Features are noisy / uncorrelated to concept



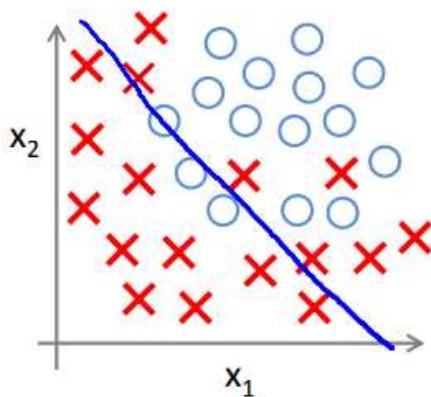
Underfitting

- Learning too little of the true concept
 - Features don't capture concept
 - Too much bias in model



Example

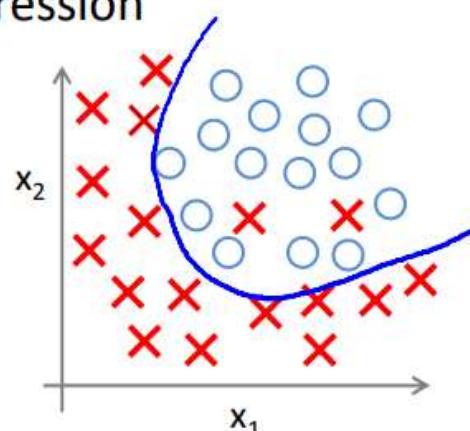
Example: Logistic regression



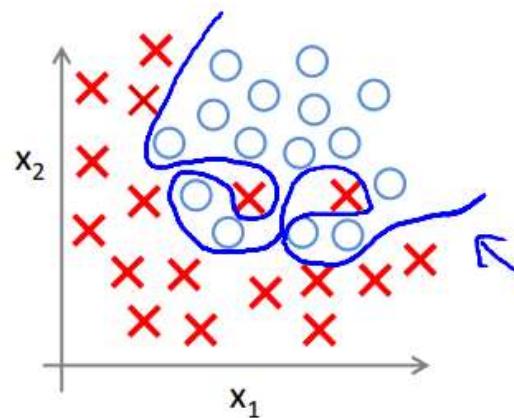
$$h_\theta(x) = g(\underline{\theta_0 + \theta_1 x_1 + \theta_2 x_2})$$

(g = sigmoid function)

↖ "Under-fit"



$$\begin{aligned} &g(\underline{\theta_0 + \theta_1 x_1 + \theta_2 x_2}) \\ &+ \theta_3 \underline{x_1^2} + \theta_4 \underline{x_2^2} \\ &+ \theta_5 \underline{x_1 x_2}) \end{aligned}$$



$$\begin{aligned} &g(\underline{\theta_0 + \theta_1 x_1 + \theta_2 x_1^2}) \\ &+ \theta_3 \underline{x_1^2 x_2} + \theta_4 \underline{x_1^2 x_2^2} \\ &+ \theta_5 \underline{x_1^2 x_2^3} + \theta_6 \underline{x_1^3 x_2} + \dots) \end{aligned}$$

↖ "Over-fit"

Overfitting

- There are two main options to address the issue of overfitting:
- 1) Reduce the number of features:
 - Manually select which features to keep.
 - Use a model selection algorithm
- 2) Regularization
 - Keep all the features, but reduce the magnitude of parameters θ
 - Regularization works well when we have a lot of features and each of which contribute a bit to predicting “y”

Ways to Control Overfitting

- Regularization

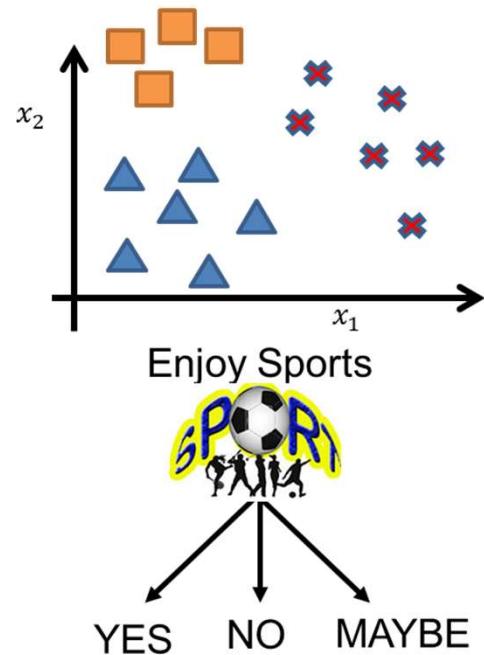
$$Loss(S) = \sum_i^n Loss(\hat{y}_i, y_i) + \alpha \sum_j^{\# Weights} |\theta_j|$$

Types of Classification

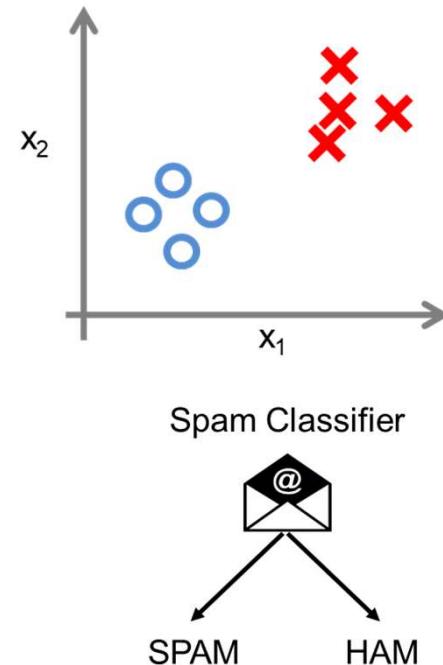
Output Labels

- Target Concept

Multi Class

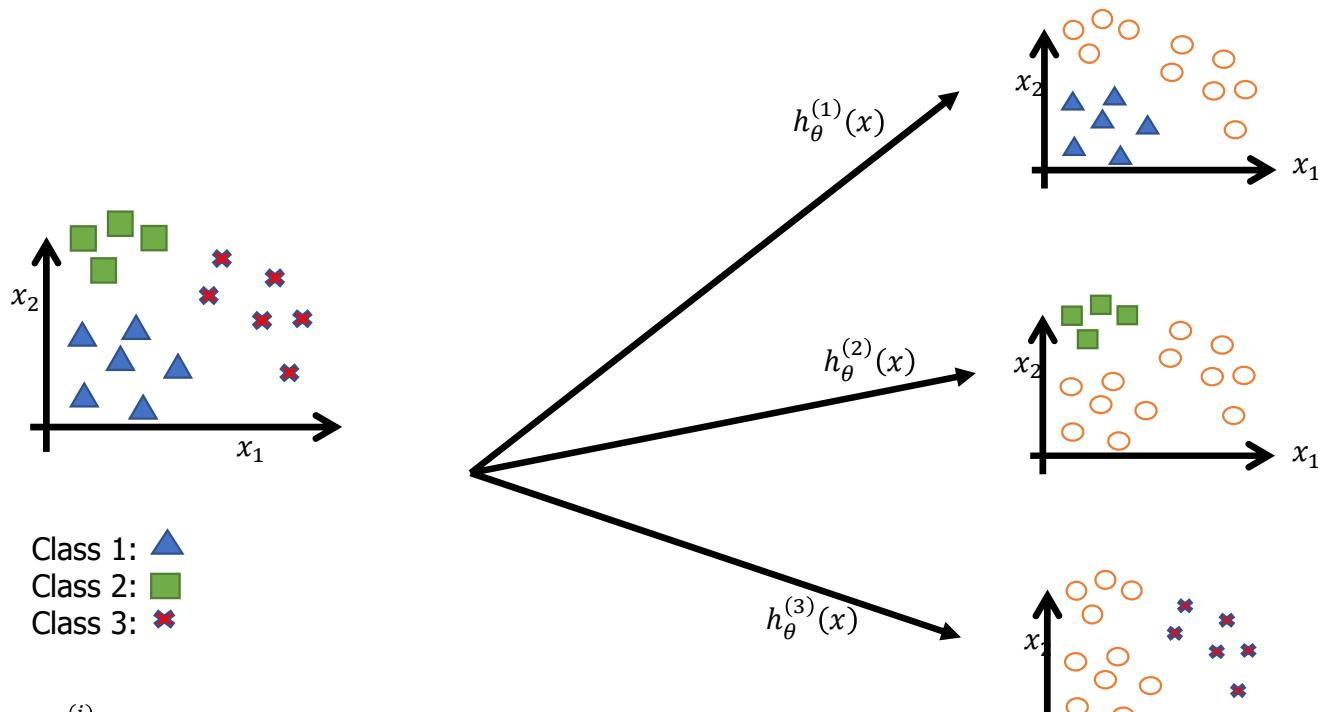


Binary



Prediction – Multi class Classification

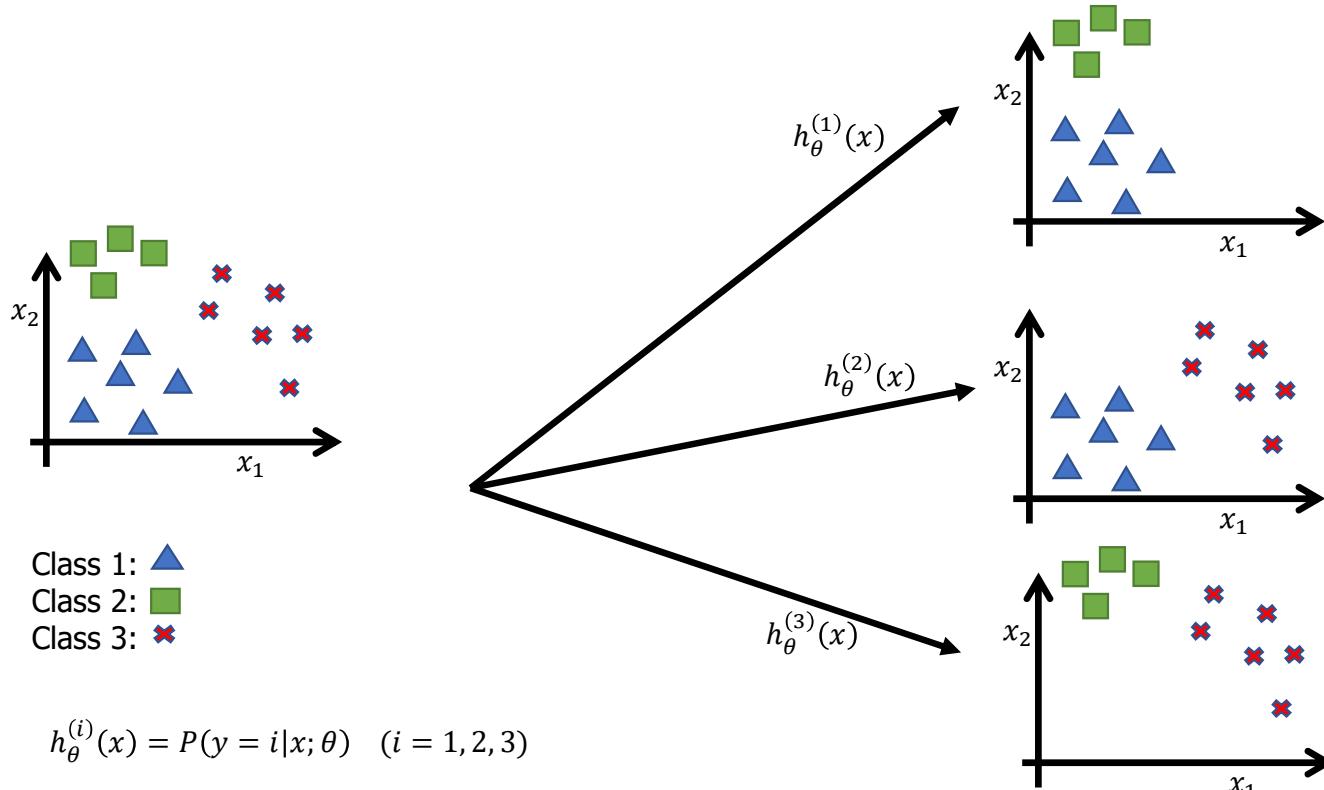
One Vs All Strategy(one-vs-rest)



Note: Scikit-Learn detects when you try to use a binary classification algorithm for a multi-class classification task, and it automatically runs OvA (except for SVM classifiers for which it uses OvO)

Prediction – Multi class Classification

One Vs One Strategy



$N \times (N - 1) / 2$ classifiers

References

- <http://www.cs.cmu.edu/~tom/NewChapters.html>
- <http://ai.stanford.edu/~ang/papers/nips01-discriminativegenerative.pdf>
- https://medium.com/@sangha_deb/naive-bayes-vs-logistic-regression-a319b07a5d4c
- <https://www.youtube.com/watch?v=-la3q9d7AKQ>

Interpretability

- <https://christophm.github.io/interpretable-ml-book/logistic.html>
- Regularization
- <https://www.youtube.com/watch?v=HFkbEOMgq8A&list=PL9jmvZt4yYPp4bCHzOp2gJE0E7OV6MLjW&index=3>
- Logit
- https://www.youtube.com/watch?v=NmjT1_nClzg



Classification: Decision Tree



BITS Pilani
Pilani Campus

Swarna Chaudhary

swarna.chaudhary@pilani.bits-pilani.ac.in



- *The slides presented here are obtained from the authors of the books and from various other contributors. I hereby acknowledge all the contributors for their material and inputs.*
- *I have added and modified a few slides to suit the requirements of the course.*



Topics to be covered

Module 5 : Decision Tree

1. Decision Tree Representation
2. Entropy and Information Gain for an attribute
3. ID3 Algorithm for decision tree learning
4. Alternative measures for selecting attributes
5. Prefer short hypothesis to longer ones, Occam's razor
6. Overfitting in Decision Tree
7. Reduced Error Pruning and Rule post pruning
8. Interpretability of Decision Trees



BITS Pilani
Pilani Campus



Decision Tree

Decision trees

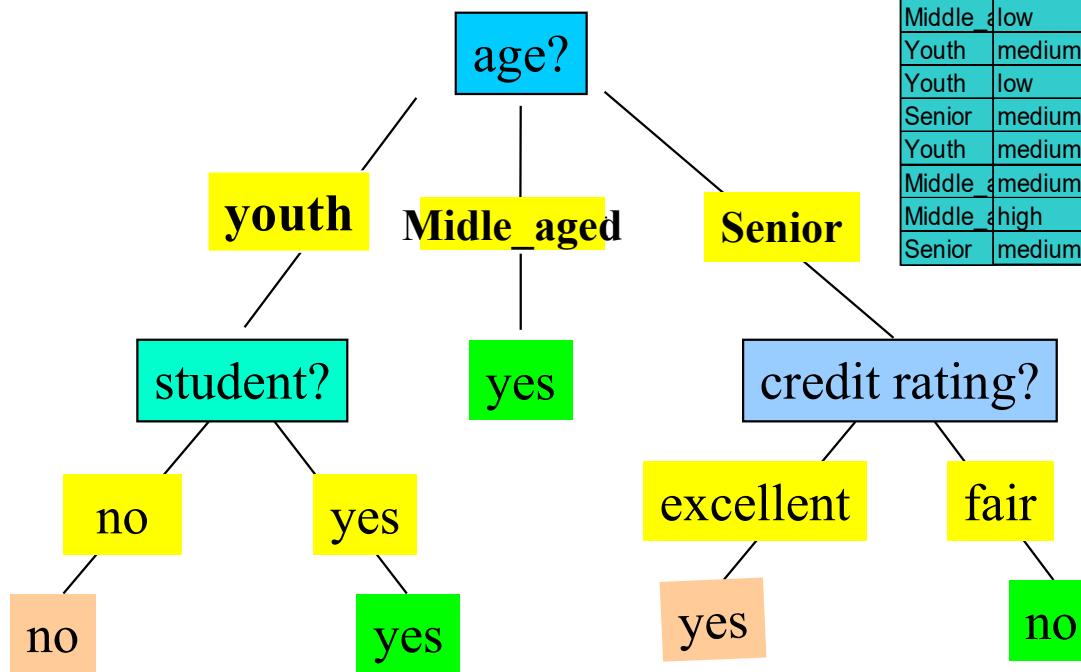
- *Decision Trees* is one of the most widely used and practical methods of *inductive inference*
- Features
 - Method for approximating *discrete-valued* functions
 - Learned functions are represented as *decision trees* (or *if-then-else* rules)
 - Interpretable Not black box
 - Humans can understand decisions
 - Decision tree good at handling noisy or missing system (low information gain)
 - Fast and Compact
 - Greedy (disadvantages)
 - Overfitting avoided by pruning

Decision Tree Induction

- Decision tree induction is the learning of decision trees from class-labeled training tuples.
- Decision tree is a flowchart-like tree structure,
 - internal node (denoted by rectangles) denotes a test on an attribute,
 - each branch represents an outcome of the test, and
 - each leaf node (or terminal node, denoted by ovals) holds a class label
- Used for classification.
- Easily converted to classification rules.
- Does not require any domain knowledge.
- Decision tree algorithms: ID3(Iterative Dichotomiser), C4.5 (successor of ID3), CART(classification and regression trees).
 - Adopt greedy approach
 - Based on top- down recursive divide and conquer approach.

Decision Tree Induction: An Example

- Training data set: Buys_computer
- Resulting tree:



age	income	student	credit_rating	buys_computer
Youth	high	no	fair	no
Youth	high	no	excellent	no
Middle	high	no	fair	yes
Senior	medium	no	fair	yes
Senior	low	yes	fair	yes
Senior	low	yes	excellent	no
Middle	low	yes	excellent	yes
Youth	medium	no	fair	no
Youth	low	yes	fair	yes
Senior	medium	yes	fair	yes
Youth	medium	yes	excellent	yes
Middle	medium	no	excellent	yes
Middle	high	yes	fair	yes
Senior	medium	no	excellent	no

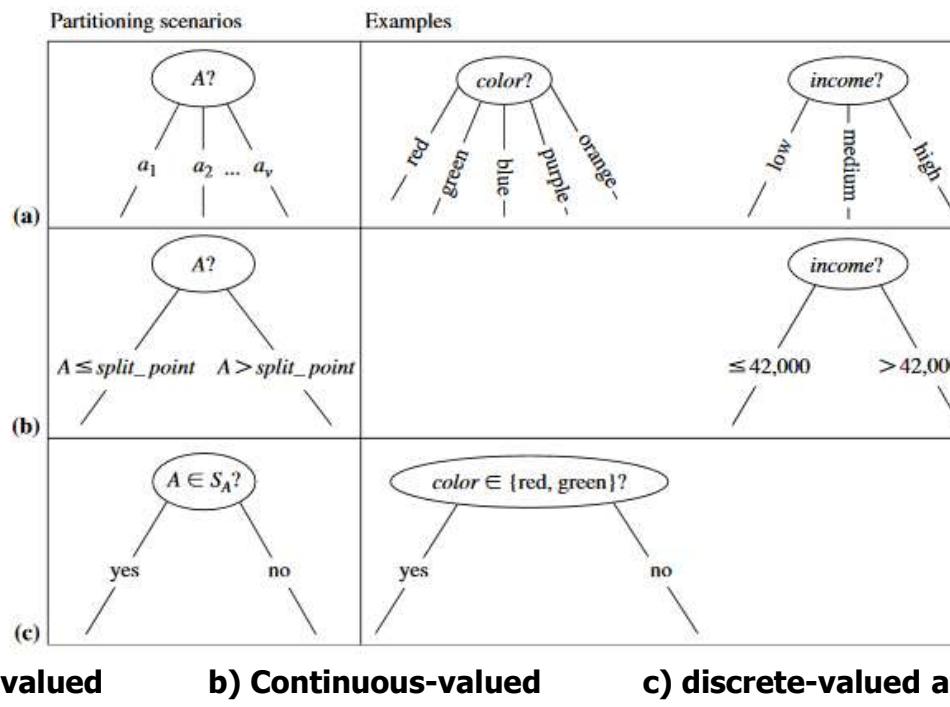
Algorithm for Decision Tree Induction

- Basic algorithm (a greedy algorithm)
 - At start, all the training examples are at the root
 - Attributes are categorical (if continuous-valued, they are discretized in advance)
 - It uses “**Attribute_selection_method**” to determine the splitting criteria.
 - Attributes are selected on the basis of a heuristic or statistical measure (e.g., [information gain](#))
- Conditions for stopping partitioning
 - All samples for a given node belong to the same class
 - There are no remaining attributes for further partitioning – [majority voting](#) is employed for classifying the leaf
 - There are no samples left
- Complexity is $O(n \times |D| \times \log|D|)$, where n is the number of attributes describing the tuples in D, $|D|$ is the number of training tuples.

Splitting Criteria

- Splitting criterion tells us which attribute to test at node N by determining the “best” way to separate or partition the tuples in D into individual classes
- Also, tells us which branches to grow from node N with respect to the outcomes of the chosen test.
- Splitting criterion indicates the splitting attribute and may also indicate either a split-point or a splitting subset
- Partition is “**Pure**” i.e. all the tuples in it belong to the same class.
- Splitting attribute **A** can be:
 - **A is discrete valued:** outcome correspond to the known values of A.
 - **A is continuous-valued:** the test at node N has two possible outcomes, corresponding⁶¹ to the conditions $A \leq \text{split point}$ and $A > \text{split point}$.

Splitting criteria...contd



Decision-tree Algorithm

Algorithm: `Generate_decision_tree`. Generate a decision tree from the training tuples of data partition, D .

Input:

- Data partition, D , which is a set of training tuples and their associated class labels;
- $attribute_list$, the set of candidate attributes;
- $Attribute_selection_method$, a procedure to determine the splitting criterion that “best” partitions the data tuples into individual classes. This criterion consists of a $splitting_attribute$ and, possibly, either a $split_point$ or $splitting_subset$.

Output: A decision tree.

Method:

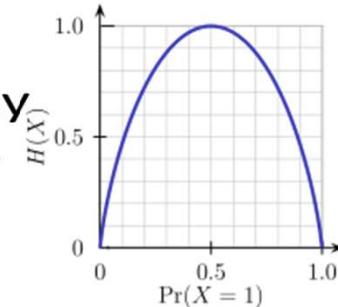
- (1) create a node N ;
- (2) if tuples in D are all of the same class, C , then
 - (3) return N as a leaf node labeled with the class C ;
 - (4) if $attribute_list$ is empty then
 - (5) return N as a leaf node labeled with the majority class in D ; // majority voting
 - (6) apply $Attribute_selection_method(D, attribute_list)$ to find the “best” $splitting_criterion$;
 - (7) label node N with $splitting_criterion$;
 - (8) if $splitting_attribute$ is discrete-valued and
 - multiway splits allowed then // not restricted to binary trees
 - (9) $attribute_list \leftarrow attribute_list - splitting_attribute$; // remove $splitting_attribute$
 - (10) for each outcome j of $splitting_criterion$
 - // partition the tuples and grow subtrees for each partition
 - (11) let D_j be the set of data tuples in D satisfying outcome j ; // a partition
 - (12) if D_j is empty then
 - (13) attach a leaf labeled with the majority class in D to node N ;
 - (14) else attach the node returned by `Generate_decision_tree($D_j, attribute_list$)` to node N ;
 - (15) endfor

Attribute Selection Measures

- Also, known as Splitting rules
- Measure is a heuristic for selecting the splitting criterion that “best” separates a given data partition D, of class-labeled training tuples into individual classes.
- Partition should be pure (i.e., all the tuples that fall into a given partition would belong to the same class).
- Provides ranking to each attribute of training tuple, and the attribute having “best” score is chosen as the splitting attribute.

Brief Review of Entropy

- **Entropy (Information Theory)**
 - A measure of uncertainty associated with a random variable
 - Calculation: For a discrete random variable Y taking m distinct values $\{y_1, \dots, y_m\}$,
 - $H(Y) = -\sum_{i=1}^m p_i \log(p_i)$, where $p_i = P(Y = y_i)$
 - Interpretation:
 - Higher entropy => higher uncertainty
 - Lower entropy => lower uncertainty
- **Conditional Entropy**
 - $H(Y|X) = \sum_x p(x)H(Y|X = x)$



m = 2

Attribute Selection Measure: Information Gain (ID3/C4.5)

- Select the attribute with the highest information gain
- This attribute minimizes the expected number of tests needed to classify a given tuple.
- Let p_i be the probability that a tuple in D belongs to class C_i , estimated by $|C_{i,D}|/|D|$, m is the number of distinct classes, v is the number of distinct values in an attribute.
- **Expected information** (entropy) needed to classify a tuple in D:

$$Info(D) = - \sum_{i=1}^m p_i \log_2(p_i)$$

- **Information** needed (after using A to split D into v partitions) to classify D

$$Info_A(D) = \sum_{j=1}^v \frac{|D_j|}{|D|} \times Info(D_j)$$

- The smaller the expected information required, greater the purity of the partitions.

Attribute Selection Measure: Information Gain (ID3/C4.5)

- **Information gained** by branching on attribute A

$$Gain(A) = Info(D) - Info_A(D)$$

- Defined as the difference between the original information requirement (i.e. based on just the proportion of the classes) and the new requirement (i.e. obtained after partitioning of attribute A).
- Attribute with highest information gain $Gain(A)$, is chosen as the splitting attribute at node N.

Attribute Selection: Information Gain

■ Class P: $\text{buys_computer} = \text{"yes"}$

■ Class N: $\text{buys_computer} = \text{"no"}$

$$\text{Info}(D) = I(9,5) = -\frac{9}{14} \log_2(\frac{9}{14}) - \frac{5}{14} \log_2(\frac{5}{14}) = 0.940$$

age	p_i	n_i	$I(p_i, n_i)$
Youth	2	3	0.971
Middle	4	0	0
Senior	3	2	0.971

$$\begin{aligned} \text{Info}_{age}(D) &= \frac{5}{14} I(2,3) + \frac{4}{14} I(4,0) \\ &\quad + \frac{5}{14} I(3,2) = 0.694 \end{aligned}$$

$\frac{5}{14} I(2,3)$ means "age ≤ 30 " has 5 out of 14 samples, with 2 yes's and 3 no's.

age	income	student	credit_rating	buys_computer
Youth	high	no	fair	no
Youth	high	no	excellent	no
Middle	high	no	fair	yes
Senior	medium	no	fair	yes
Senior	low	yes	fair	yes
Senior	low	yes	excellent	no
Middle	low	yes	excellent	yes
Youth	medium	no	fair	no
Youth	low	yes	fair	yes
Senior	medium	yes	fair	yes
Youth	medium	yes	excellent	yes
Middle	medium	no	excellent	yes
Middle	high	yes	fair	yes
Senior	medium	no	excellent	no

$$\begin{aligned} \text{Info}_{age}(D) &= (5/14)X(-2/5\log(2/5) - 3/5\log(3/5)) \\ &\quad + (4/4)X(-4/4)\log(4/4) \\ &\quad + 5/14((-3/5)X\log(3/5) - (2/5)\log(2/5)) = 0.694 \end{aligned}$$

$$\text{Gain}(age) = \text{Info}(D) - \text{Info}_{age}(D) = 0.246$$

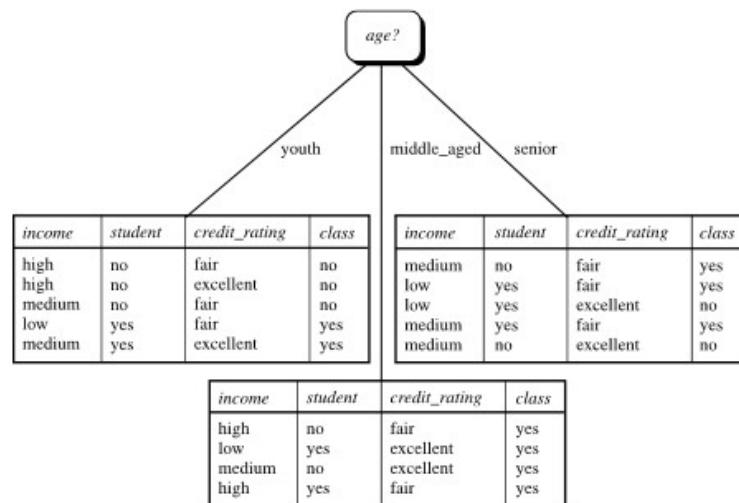
Similarly,

$$\text{Gain } (\text{income }) = 0.029$$

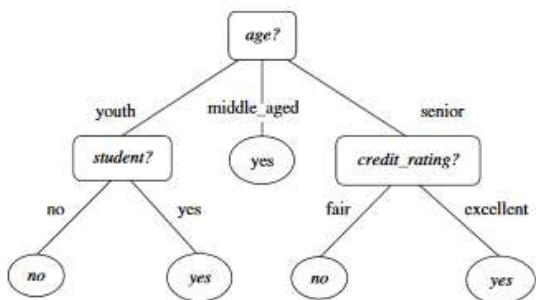
$$\text{Gain } (\text{student }) = 0.151$$

$$\text{Gain } (\text{credit_rating }) = 0.048$$

Attribute Selection: Information Gain



Decision Tree



Decision tree for the concept "***buys_computer***"

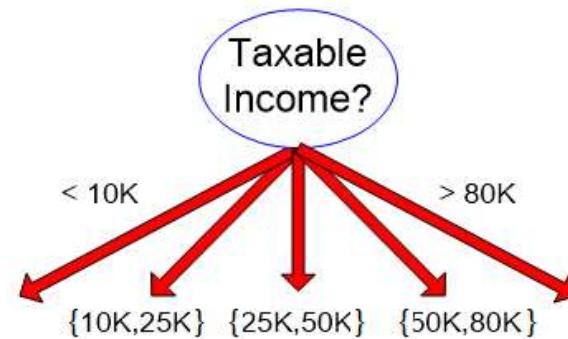
Splitting Based on Continuous Attributes

- Different ways of handling
 - **Discretization** to form an ordinal categorical attribute
 - **Binary Decision:** $(A < v)$ or $(A \geq v)$
 - consider all possible splits and finds the best cut
 - can be more compute intensive

Splitting Based on Continuous Attributes



(i) Binary split



(ii) Multi-way split

Computing Information-Gain for Continuous-Valued Attributes

- Let attribute A be a continuous-valued attribute
- Must determine the *best split point* for A
 - Sort the value A in increasing order
 - Typically, the midpoint between each pair of adjacent values is considered as a possible *split point*
 - $(a_i + a_{i+1})/2$ is the midpoint between the values of a_i and a_{i+1}
 - The point with the *minimum expected information requirement* for A is selected as the split-point for A
- Split:
 - D1 is the set of tuples in D satisfying $A \leq$ split-point, and D2 is the set of tuples in D satisfying $A >$ split-point

Question?

- Compute the information gain for every possible split for the given continuous valued attribute?
- Sort the values and find the midpoint between each pair of adjacent values

Class	+	-	+	-	+	-	-
Sorted A2	1	3	4	5	6	7	8
Split_Point		2	3.5	4.5	5.5	6.5	7.5

A1	A2	Class
T	1	+
T	6	+
T	5	-
F	4	+
F	7	-
F	3	-
F	8	-
T	7	+
F	5	-

- Calculate Info(D)
- Calculate the entropy for each split_point for “ \leq ” and “ $>$ ”
- Find Gain for each split_point

Gain Ratio for Attribute Selection (C4.5)

- Information gain measure is biased towards attributes with a large number of distinct values. Eg. Product_ID (unique for every tuple), resulting in large number of partitions as $\text{Info}_{\text{product_ID}}(D) = 0$, Such partitioning is useless.
- C4.5 (a successor of ID3) uses gain ratio to overcome the problem.
- It applies normalization to information gain using a “split information”

$$\text{SplitInfo}_A(D) = - \sum_{j=1}^v \frac{|D_j|}{|D|} \times \log_2 \left(\frac{|D_j|}{|D|} \right)$$

- $\text{GainRatio}(A) = \text{Gain}(A)/\text{SplitInfo}_A(D)$
- Ex. Gain Ratio of “income” on the given data set.

$$\text{SplitInfo}_{\text{income}}(D) = -\frac{4}{14} \times \log_2 \left(\frac{4}{14} \right) - \frac{6}{14} \times \log_2 \left(\frac{6}{14} \right) - \frac{4}{14} \times \log_2 \left(\frac{4}{14} \right) = 1.557$$

- $\text{gain_ratio}(\text{income}) = 0.029/1.557 = 0.019$
- The attribute with the maximum gain ratio is selected as the splitting attribute

Gini Index (CART, IBM IntelligentMiner)

- If a data set D contains examples from n classes, gini index, $gini(D)$ is defined as

$$gini(D) = 1 - \sum_{j=1}^n p_j^2$$

where p_j is the probability that a tuple in D belongs to class j is $|C_{j,D}|/|D|$

- It considers binary-split for each attribute
- If a data set D is split on A (such that $A \in S_A$) into two subsets D_1 and D_2 , the gini index $gini(D)$ is defined as

$$gini_A(D) = \frac{|D_1|}{|D|} gini(D_1) + \frac{|D_2|}{|D|} gini(D_2)$$

- Reduction in Impurity:

$$\Delta gini(A) = gini(D) - gini_A(D)$$

- The attribute provides the minimum $gini_{split}(D)$ (or the largest reduction in impurity) is chosen to split the node (*need to enumerate all the possible splitting points for each attribute*)

Computation of Gini Index

- Ex. D has 9 tuples in buys_computer = “yes” and 5 in “no”

$$gini(D) = 1 - \left(\frac{9}{14}\right)^2 - \left(\frac{5}{14}\right)^2 = 0.459$$

- Suppose the attribute income partitions D into 10 in D_1 : {low, medium} and 4 in D_2

$$gini_{income \in \{low, medium\}}(D) = \left(\frac{10}{14}\right)Gini(D_1) + \left(\frac{4}{14}\right)Gini(D_2)$$

$$= \frac{10}{14} \left(1 - \left(\frac{7}{10}\right)^2 - \left(\frac{3}{10}\right)^2\right) + \frac{4}{14} \left(1 - \left(\frac{2}{4}\right)^2 - \left(\frac{2}{4}\right)^2\right)$$

$$= 0.443$$

$$= Gini_{income \in \{high\}}(D).$$

$Gini_{\{low, high\}}$ is 0.458; $Gini_{\{medium, high\}}$ is 0.450. Thus, split on the {low, medium} (and {high}) since it has the lowest Gini index

- May need other tools, e.g., clustering, to get the possible split values

age	income	student	credit rating	buys_computer
Youth	high	no	fair	no
Youth	high	no	excellent	no
Middle	high	no	fair	yes
Senior	medium	no	fair	yes
Senior	low	yes	fair	yes
Senior	low	yes	excellent	no
Middle	low	yes	excellent	yes
Youth	medium	no	fair	no
Youth	low	yes	fair	yes
Senior	medium	yes	fair	yes
Youth	medium	yes	excellent	yes
Middle	medium	no	excellent	yes
Middle	high	yes	fair	yes
Senior	medium	no	excellent	no

Comparing Attribute Selection Measures

- The three measures, in general, return good results but
 - **Information gain:**
 - biased towards multivalued attributes
 - **Gain ratio:**
 - tends to prefer unbalanced splits in which one partition is much smaller than the others
 - **Gini index:**
 - biased to multivalued attributes
 - has difficulty when # of classes is large

Decision Tree Based Classification

- Advantages:
 - Inexpensive to construct
 - Extremely fast at classifying unknown records
 - Easy to interpret for small-sized trees
 - Accuracy is comparable to other classification techniques for many simple data sets

Inductive bias in decision tree learning

- Inductive bias is the assumptions made by the model to learn the target function and to generalize beyond training data.
- What is the inductive bias of DT learning?
 1. Shorter trees are preferred over longer trees

This is the bias exhibited by a simple breadth first algorithm generating all DT's & selecting the shorter one
 2. Prefer trees that place high information gain attributes close to the root

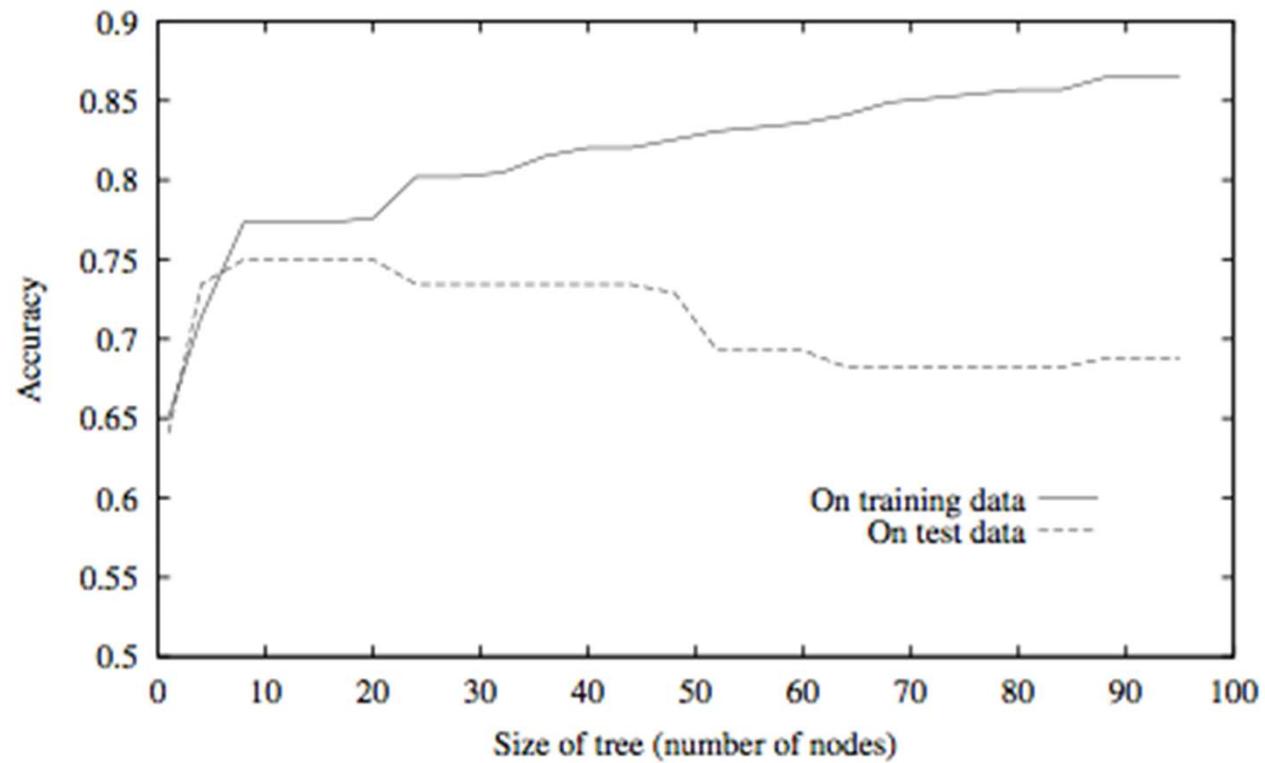
Prefer shorter hypotheses: Occam's razor

- Why prefer shorter hypotheses?
- Arguments in favor:
 - There are fewer short hypotheses than long ones
 - If a short hypothesis fits data unlikely to be a coincidence
 - Elegance and aesthetics
- Arguments against:
 - Not every short hypothesis is a reasonable one.
- Occam's razor says that when presented with competing hypotheses that make the same predictions, one should select the solution which is simple"

Issues in decision trees learning

- Overfitting
 - Building trees that “adapt too much” to the training examples may lead to “overfitting”.
 - May therefore fail to fit additional data or predict future observations reliably

Overfitting in decision tree learning



How to Address Overfitting

- Pre-Pruning (Early Stopping Rule)
 - Stop the algorithm before it becomes a fully-grown tree
 - General stopping conditions for a node:
 - Stop if all instances belong to the same class
 - Stop if all the attribute values are the same
 - More restrictive conditions (for pre-pruning) :
 - Stop if number of instances is less than some user-specified threshold
 - Stop if class distribution of instances are independent of the available features (e.g., using χ^2 test)
 - Stop if expanding the current node does not improve impurity measures (e.g., Gini or information gain).

How to Address Overfitting...

- Post-pruning
 - Grow decision tree to its entirety
 - Trim the nodes of the decision tree in a bottom-up fashion
 - If generalization error(i.e. expected error of the model on previously unseen records) improves after trimming, replace sub-tree by a leaf node.
 - Class label of leaf node is determined from majority class of instances in the sub-tree

Good References

Decision Tree

- https://www.youtube.com/watch?v=eKD5gxPPeY0&list=PLBv09BD7ez_4temBw7vLA19p3tdQH6FYO&index=1

Overfitting

- https://www.youtube.com/watch?time_continue=1&v=t56Nid85Thg
- <https://www.youtube.com/watch?v=y6SpA2Wuyt8>
- Decision tree for regression
- https://www.saedsayad.com/decision_tree_reg.htm

Textbook/Reference Books

- Tom M. Mitchell: Machine Learning, The McGraw-Hill Companies, Inc..
- Tan P. N., Steinbach M & Kumar V. “Introduction to Data Mining” Pearson Education
- Data Mining: Concepts and Techniques, Third Edition by Jiawei Han, Micheline Kamber and Jian Pei Morgan Kaufmann Publishers