

# Data Mining

Prof. Dr. Nizamettin AYDIN

[naydin@yildiz.edu.tr](mailto:naydin@yildiz.edu.tr)

<http://www3.yildiz.edu.tr/~naydin>

1

# Data Mining

## Classification: Alternative Techniques

- Outline
  - Types of Classifiers
  - Rule-Based Classifier
  - Nearest Neighbor Classifiers
  - Naïve Bayes Classifier
  - Bayesian Networks
  - Logistic Regression
  - Artificial Neural Network (ANN)
  - Deep Learning
  - Support Vector Machine (SVM)
  - Ensemble Methods
  - Class Imbalance Problem
  - Multiclass Problem

2

## Types of Classifiers

- One way to distinguish classifiers is by considering the characteristics of their output
  - Binary versus Multiclass
    - Binary classifiers assign each data instance to one of two possible labels, typically denoted as +1 and -1
    - If there are more than two possible labels available, then the technique is known as a multiclass classifier
  - Deterministic versus Probabilistic
    - A deterministic classifier produces a discrete-valued label to each data instance it classifies
    - A probabilistic classifier assigns a continuous score between 0 and 1 to indicate how likely it is that an instance belongs to a particular class

3

## Types of Classifiers

- Linear versus Nonlinear
  - A linear classifier uses a linear separating hyperplane to discriminate instances from different classes
  - A nonlinear classifier enables the construction of more complex, nonlinear decision surfaces.
- Global versus Local
  - A global classifier fits a single model to the entire data set.
  - A local classifier partitions the input space into smaller regions and fits a distinct model to training instances in each region.
- Generative versus Discriminative
  - Classifiers that learn a generative model of every class in the process of predicting class labels are known as generative classifiers
  - Discriminative classifiers directly predict the class labels without explicitly describing the distribution of every class label

4

## Rule-Based Classifier

5

## Rule-Based Classifier

- Classify records by using a collection of “if...then...” rules
- Rule:  $(Condition) \rightarrow y$ 
  - where
    - Condition is a conjunction of tests on attributes
    - y is the class label
  - Examples of classification rules:
    - $(Blood\ Type=Warm) \wedge (Lay\ Eggs=Yes) \rightarrow Birds$
    - $(Taxable\ Income < 50K) \wedge (Refund=Yes) \rightarrow Evade=No$

6

## Rule-based Classifier (Example)

Name	Blood Type	Give Birth	Can Fly	Live in Water	Class
human	warm	yes	no	no	mammals
python	cold	no	no	no	reptiles
salmon	cold	no	yes	yes	fishes
whale	warm	yes	no	yes	mammals
frog	cold	no	no	sometimes	amphibians
komodo	cold	no	no	no	reptiles
bat	warm	yes	yes	no	mammals
pigeon	warm	yes	yes	no	birds
cat	warm	yes	no	no	mammals
leopard shark	cold	yes	no	yes	fishes
turtle	cold	no	no	sometimes	reptiles
penguin	warm	no	no	no	birds
porcupine	warm	yes	no	no	mammals
eel	cold	no	no	yes	fishes
salamander	cold	no	no	sometimes	amphibians
gila monster	cold	no	no	no	reptiles
platypus	warm	no	no	no	mammals
owl	warm	no	yes	no	birds
dolphin	warm	yes	no	yes	mammals
eagle	warm	no	yes	no	birds

R1: (Give Birth = no)  $\wedge$  (Can Fly = yes)  $\rightarrow$  Birds

R2: (Give Birth = no)  $\wedge$  (Live in Water = yes)  $\rightarrow$  Fishes

R3: (Give Birth = yes)  $\wedge$  (Blood Type = warm)  $\rightarrow$  Mammals

R4: (Give Birth = no)  $\wedge$  (Can Fly = no)  $\rightarrow$  Reptiles

R5: (Live in Water = sometimes)  $\rightarrow$  Amphibians

7

## Application of Rule-Based Classifier

- A rule  $r$  **covers** an instance  $x$  if the attributes of the instance satisfy the condition of the rule

R1: (Give Birth = no)  $\wedge$  (Can Fly = yes)  $\rightarrow$  Birds

R2: (Give Birth = no)  $\wedge$  (Live in Water = yes)  $\rightarrow$  Fishes

R3: (Give Birth = yes)  $\wedge$  (Blood Type = warm)  $\rightarrow$  Mammals

R4: (Give Birth = no)  $\wedge$  (Can Fly = no)  $\rightarrow$  Reptiles

R5: (Live in Water = sometimes)  $\rightarrow$  Amphibians

Name	Blood Type	Give Birth	Can Fly	Live in Water	Class
hawk	warm	no	yes	no	?
grizzly bear	warm	yes	no	no	?

The rule R1 covers a hawk  $\Rightarrow$  Bird

The rule R3 covers the grizzly bear  $\Rightarrow$  Mammal

8

## Rule Coverage and Accuracy

- Coverage of a rule:

– Fraction of records that satisfy the antecedent of a rule

- Accuracy of a rule:

– Fraction of records that satisfy the antecedent that also satisfy the consequent of a rule

Tid	Refund	Marital Status	Taxable Income	Class
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

(Status=Single)  $\rightarrow$  No

Coverage = 40%, Accuracy = 50%

9

## How does Rule-based Classifier Work?

R1: (Give Birth = no)  $\wedge$  (Can Fly = yes)  $\rightarrow$  Birds

R2: (Give Birth = no)  $\wedge$  (Live in Water = yes)  $\rightarrow$  Fishes

R3: (Give Birth = yes)  $\wedge$  (Blood Type = warm)  $\rightarrow$  Mammals

R4: (Give Birth = no)  $\wedge$  (Can Fly = no)  $\rightarrow$  Reptiles

R5: (Live in Water = sometimes)  $\rightarrow$  Amphibians

Name	Blood Type	Give Birth	Can Fly	Live in Water	Class
lemur	warm	yes	no	no	?
turtle	cold	no	no	sometimes	?
dogfish shark	cold	yes	no	yes	?

A lemur triggers rule R3, so it is classified as a mammal

A turtle triggers both R4 and R5

A dogfish shark triggers none of the rules

10

## Characteristics of Rule Sets: Strategy 1

- Mutually exclusive rules
  - Classifier contains mutually exclusive rules if the rules are independent of each other
  - Every record is covered by at most one rule
- Exhaustive rules
  - Classifier has exhaustive coverage if it accounts for every possible combination of attribute values
  - Each record is covered by at least one rule

11

## Characteristics of Rule Sets: Strategy 2

- Rules are not mutually exclusive
  - A record may trigger more than one rule
  - Solution?
    - Ordered rule set
    - Unordered rule set – use voting schemes
- Rules are not exhaustive
  - A record may not trigger any rules
  - Solution?
    - Use a default class

12

## Ordered Rule Set

- Rules are rank ordered according to their priority
  - An ordered rule set is known as a decision list
- When a test record is presented to the classifier
  - It is assigned to the class label of the highest ranked rule it has triggered
  - If none of the rules fired, it is assigned to the default class

R1: (Give Birth = no)  $\wedge$  (Can Fly = yes)  $\rightarrow$  Birds  
 R2: (Give Birth = no)  $\wedge$  (Live in Water = yes)  $\rightarrow$  Fishes  
 R3: (Give Birth = yes)  $\wedge$  (Blood Type = warm)  $\rightarrow$  Mammals  
 R4: (Give Birth = no)  $\wedge$  (Can Fly = no)  $\rightarrow$  Reptiles  
 R5: (Live in Water = sometimes)  $\rightarrow$  Amphibians

Name	Blood Type	Give Birth	Can Fly	Live in Water	Class
turtle	cold	no	no	sometimes	?

13

## Rule Ordering Schemes

- Rule-based ordering
  - Individual rules are ranked based on their quality
- Class-based ordering
  - Rules that belong to the same class appear together

### Rule-based Ordering

(Refund=Yes)  $\Rightarrow$  No  
 (Refund=No, Marital Status=(Single,Divorced), Taxable Income<80K)  $\Rightarrow$  No  
 (Refund=No, Marital Status=(Single,Divorced), Taxable Income>80K)  $\Rightarrow$  Yes  
 (Refund=No, Marital Status=(Married))  $\Rightarrow$  No

### Class-based Ordering

(Refund=Yes)  $\Rightarrow$  No  
 (Refund=No, Marital Status=(Single,Divorced), Taxable Income<80K)  $\Rightarrow$  No  
 (Refund=No, Marital Status=(Married))  $\Rightarrow$  No  
 (Refund=No, Marital Status=(Single,Divorced), Taxable Income>80K)  $\Rightarrow$  Yes

14

## Building Classification Rules

- Direct Method:
  - Extract rules directly from data
  - Examples: RIPPER, CN2, Holte's 1R
- Indirect Method:
  - Extract rules from other classification models (e.g. decision trees, neural networks, etc).
  - Examples: C4.5rules

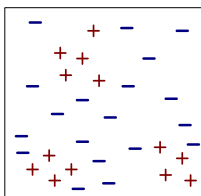
15

## Direct Method: Sequential Covering

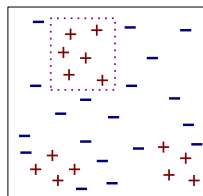
- Start from an empty rule
- Grow a rule using the Learn-One-Rule function
- Remove training records covered by the rule
- Repeat Step (2) and (3) until stopping criterion is met

16

## Example of Sequential Covering



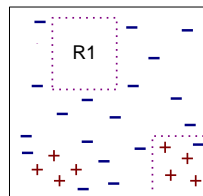
(i) Original Data



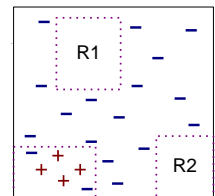
(ii) Step 1

17

## Example of Sequential Covering...



(iii) Step 2

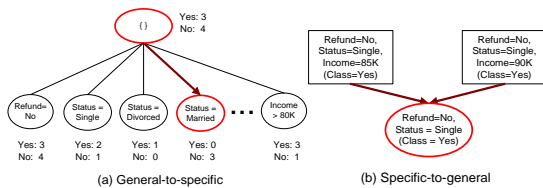


(iv) Step 3

18

## Rule Growing

- Two common strategies



## Rule Evaluation

- Foil's Information Gain

FOIL: First Order Inductive Learner – an early rule-based learning algorithm

- $R_0: {} \Rightarrow \text{class}$  (initial rule)
- $R_1: \{A\} \Rightarrow \text{class}$  (rule after adding conjunct)
- $$\text{Gain}(R_0, R_1) = p_1 \times \left[ \log_2 \left( \frac{p_1}{p_1 + n_1} \right) - \log_2 \left( \frac{p_0}{p_0 + n_0} \right) \right]$$
- $p_0$ : number of positive instances covered by  $R_0$
- $n_0$ : number of negative instances covered by  $R_0$
- $p_1$ : number of positive instances covered by  $R_1$
- $n_1$ : number of negative instances covered by  $R_1$

## Direct Method: RIPPER

- For 2-class problem, choose one of the classes as positive class, and the other as negative class
  - Learn rules for positive class
  - Negative class will be default class
- For multi-class problem
  - Order the classes according to increasing class prevalence (fraction of instances that belong to a particular class)
  - Learn the rule set for smallest class first, treat the rest as negative class
  - Repeat with next smallest class as positive class

21

## Direct Method: RIPPER

- Growing a rule:
  - Start from empty rule
  - Add conjuncts as long as they improve FOIL's information gain
  - Stop when rule no longer covers negative examples
  - Prune the rule immediately using incremental reduced error pruning
  - Measure for pruning:  $v = (p-n)/(p+n)$ 
    - $p$ : number of positive examples covered by the rule in the validation set
    - $n$ : number of negative examples covered by the rule in the validation set
  - Pruning method: delete any final sequence of conditions that maximizes  $v$

22

## Direct Method: RIPPER

- Building a Rule Set:
  - Use sequential covering algorithm
    - Finds the best rule that covers the current set of positive examples
    - Eliminate both positive and negative examples covered by the rule
  - Each time a rule is added to the rule set, compute the new description length
    - Stop adding new rules when the new description length is  $d$  bits longer than the smallest description length obtained so far

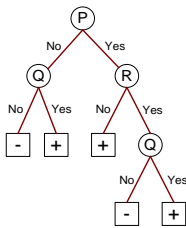
23

## Direct Method: RIPPER

- Optimize the rule set:
  - For each rule  $r$  in the rule set  $R$ 
    - Consider 2 alternative rules:
      - Replacement rule ( $r^*$ ): grow new rule from scratch
      - Revised rule ( $r'$ ): add conjuncts to extend the rule  $r$
    - Compare the rule set for  $r$  against the rule set for  $r^*$  and  $r'$
    - Choose rule set that minimizes MDL principle
  - Repeat rule generation and rule optimization for the remaining positive examples

24

## Indirect Methods



### Rule Set

r1: (P=No,Q=No) ==> -  
 r2: (P=No,Q=Yes) ==> +  
 r3: (P=Yes,R=No) ==> +  
 r4: (P=Yes,R=Yes,Q=No) ==> -  
 r5: (P=Yes,R=Yes,Q=Yes) ==> +

25

## Indirect Method: C4.5rules

- Extract rules from an unpruned decision tree
- For each rule,  $r: A \rightarrow y$ ,
  - consider an alternative rule  $r': A' \rightarrow y$  where  $A'$  is obtained by removing one of the conjuncts in  $A$
  - Compare the pessimistic error rate for  $r$  against all  $r'$ 's
  - Prune if one of the alternative rules has lower pessimistic error rate
  - Repeat until we can no longer improve generalization error

26

## Indirect Method: C4.5rules

- Instead of ordering the rules, order subsets of rules (class ordering)
  - Each subset is a collection of rules with the same rule consequent (class)
  - Compute description length of each subset
    - Description length =  $L(\text{error}) + g L(\text{model})$
    - $g$  is a parameter that takes into account the presence of redundant attributes in a rule set (default value = 0.5)

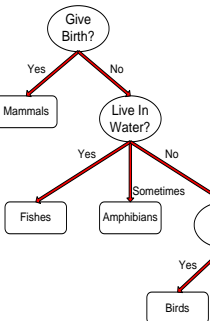
27

## Example

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

28

## C4.5 versus C4.5rules versus RIPPER



### C4.5rules:

(Give Birth=No, Can Fly=Yes) → Birds  
 (Give Birth=No, Live in Water=Yes) → Fishes  
 (Give Birth=Yes) → Mammals  
 (Give Birth=No, Can Fly=No, Live in Water=No) → Reptiles  
 () → Amphibians

### RIPPER:

(Live in Water=Yes) → Fishes  
 (Have Legs=No) → Reptiles  
 (Give Birth=No, Can Fly=No, Live in Water=No) → Reptiles  
 (Can Fly=Yes, Give Birth=No) → Birds  
 () → Mammals

29

## C4.5 versus C4.5rules versus RIPPER

### C4.5 and C4.5rules:

		PREDICTED CLASS				
		Amphibians	Fishes	Reptiles	Birds	Mammals
ACTUAL CLASS	Amphibians	2	0	0	0	0
	Fishes	0	2	0	0	1
	Reptiles	1	0	3	0	0
	Birds	1	0	0	3	0
	Mammals	0	0	1	0	6

### RIPPER:

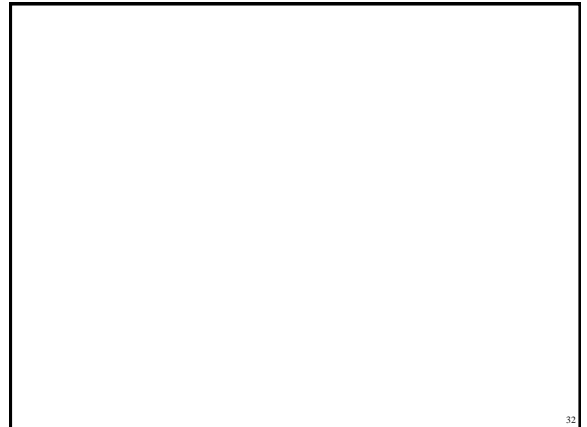
		PREDICTED CLASS				
		Amphibians	Fishes	Reptiles	Birds	Mammals
ACTUAL CLASS	Amphibians	0	0	0	0	2
	Fishes	0	3	0	0	0
	Reptiles	0	0	3	0	1
	Birds	0	0	1	2	1
	Mammals	0	2	1	0	4

30

## Advantages of Rule-Based Classifiers

- Has characteristics quite similar to decision trees
  - As highly expressive as decision trees
  - Easy to interpret (if rules are ordered by class)
  - Performance comparable to decision trees
    - Can handle redundant and irrelevant attributes
    - Variable interaction can cause issues (e.g., X-OR problem)
- Better suited for handling imbalanced classes
- Harder to handle missing values in the test set

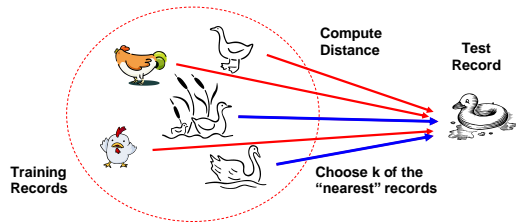
31



32

## Nearest Neighbor Classifiers

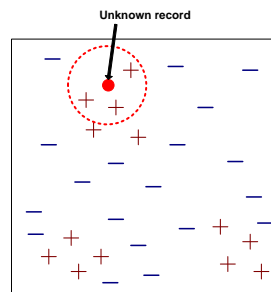
- Basic idea:
  - If it walks like a duck, quacks like a duck, then it's probably a duck



33

## Nearest-Neighbor Classifiers

- Requires the following:
  - A set of labeled records
  - Proximity metric to compute distance/similarity between a pair of records
    - e.g., Euclidean distance
  - The value of  $k$ , the number of nearest neighbors to retrieve
  - A method for using class labels of  $K$  nearest neighbors to determine the class label of unknown record (e.g., by taking majority vote)



34

## How to Determine the class label of a Test Sample?

- Take the majority vote of class labels among the  $k$ -nearest neighbors
- Weight the vote according to distance
  - weight factor,  $w = 1/d^2$

35

## Choice of proximity measure matters

- For documents, cosine is better than correlation or Euclidean

1 1 1 1 1 1 1 1 1 1 1 0	vs	0 0 0 0 0 0 0 0 0 0 0 1
0 1 1 1 1 1 1 1 1 1 1 1		1 0 0 0 0 0 0 0 0 0 0 0

Euclidean distance = 1.4142 for both pairs, but the cosine similarity measure has different values for these pairs.

36

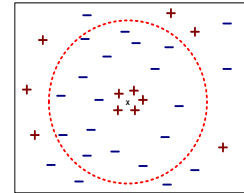
## Nearest Neighbor Classification...

- **Data preprocessing is often required**
  - Attributes may have to be scaled to prevent distance measures from being dominated by one of the attributes
    - Example:
      - height of a person may vary from 1.5m to 1.8m
      - weight of a person may vary from 90lb to 300lb
      - income of a person may vary from \$10K to \$1M
  - Time series are often standardized to have 0 means a standard deviation of 1

37

## Nearest Neighbor Classification...

- Choosing the value of k:
  - If k is too small, sensitive to noise points
  - If k is too large, neighborhood may include points from other classes

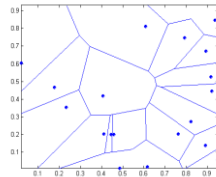


38

## Nearest-neighbor classifiers

- Nearest neighbor classifiers are local classifiers
- They can produce decision boundaries of arbitrary shapes.

1-nn decision boundary is a Voronoi Diagram



39

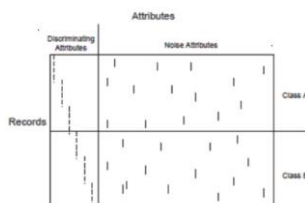
## Nearest Neighbor Classification...

- **How to handle missing values in training and test sets?**
  - Proximity computations normally require the presence of all attributes
  - Some approaches use the subset of attributes present in two instances
    - This may not produce good results since it effectively uses different proximity measures for each pair of instances
    - Thus, proximities are not comparable

40

## K-NN Classifiers... Handling Irrelevant and Redundant Attributes

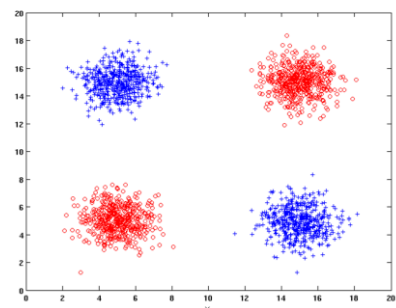
- Irrelevant attributes add noise to the proximity measure
- Redundant attributes bias the proximity measure towards certain attributes



(a) Synthetic data set 1.

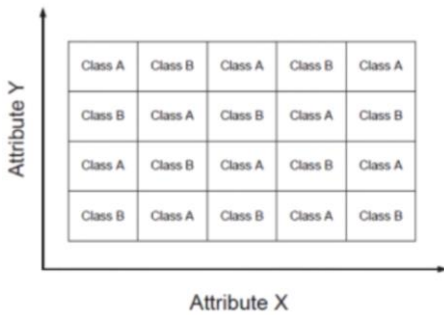
41

## K-NN Classifiers: Handling attributes that are interacting



42

## Handling attributes that are interacting



43

## Improving KNN Efficiency

- Avoid having to compute distance to all objects in the training set
  - Multi-dimensional access methods (k-d trees)
  - Fast approximate similarity search
  - Locality Sensitive Hashing (LSH)
- Condensing
  - Determine a smaller set of objects that give the same performance
- Editing
  - Remove objects to improve efficiency

44

## Bayes Classifier

- A probabilistic framework for solving classification problems
- Conditional Probability:  $P(Y | X) = \frac{P(X, Y)}{P(X)}$   
 $P(X | Y) = \frac{P(X, Y)}{P(Y)}$
- Bayes theorem:  $P(Y | X) = \frac{P(X | Y)P(Y)}{P(X)}$

46

## Using Bayes Theorem for Classification

- Consider each attribute and class label as random variables
- Given a record with attributes  $(X_1, X_2, \dots, X_d)$ , the goal is to predict class Y
  - Specifically, we want to find the value of Y that maximizes  $P(Y | X_1, X_2, \dots, X_d)$
- Can we estimate  $P(Y | X_1, X_2, \dots, X_d)$  directly from data?

Id	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

47

## Using Bayes Theorem for Classification

- Approach:
  - compute posterior probability  $P(Y | X_1, X_2, \dots, X_d)$  using the Bayes theorem

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

- Maximum a-posteriori: Choose Y that maximizes  $P(Y | X_1, X_2, \dots, X_d)$
- Equivalent to choosing value of Y that maximizes  $P(X_1, X_2, \dots, X_d | Y) P(Y)$

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

48



## Example Data

Given a Test Record:

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

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
9	No	Married	75K	No
10	No	Single	90K	Yes

- We need to estimate  $P(\text{Evade} = \text{Yes} | X)$  and  $P(\text{Evade} = \text{No} | X)$

In the following we will replace  
Evade = Yes by Yes, and  
Evade = No by No

49

## Example Data

Given a Test Record:

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

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
9	No	Married	75K	No
10	No	Single	90K	Yes

Using Bayes Theorem:

$$P(\text{Yes} | X) = \frac{P(X | \text{Yes})P(\text{Yes})}{P(X)}$$

$$P(\text{No} | X) = \frac{P(X | \text{No})P(\text{No})}{P(X)}$$

- How to estimate  $P(X | \text{Yes})$  and  $P(X | \text{No})$ ?

50

## Conditional Independence

- $X$  and  $Y$  are conditionally independent given  $Z$  if  $P(X|YZ) = P(X|Z)$
- Example: Arm length and reading skills
  - Young child has shorter arm length and limited reading skills, compared to adults
  - If age is fixed, no apparent relationship between arm length and reading skills
  - Arm length and reading skills are conditionally independent given age

51

## 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.

52

## Naïve Bayes on Example Data

Given a Test Record:

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

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
9	No	Married	75K	No
10	No	Single	90K	Yes

$$P(X | \text{Yes}) = P(\text{Refund} = \text{No} | \text{Yes}) \times P(\text{Divorced} | \text{Yes}) \times P(\text{Income} = 120\text{K} | \text{Yes})$$

$$P(X | \text{No}) = P(\text{Refund} = \text{No} | \text{No}) \times P(\text{Divorced} | \text{No}) \times P(\text{Income} = 120\text{K} | \text{No})$$

53

## Estimate Probabilities from Data

- $P(y)$  = fraction of instances of class  $y$ 
  - e.g.,  $P(\text{No}) = 7/10$ ,  $P(\text{Yes}) = 3/10$

- For categorical attributes:

$$P(X_i = c | y) = n_c / n$$

- where  $|X_i = c|$  is number of instances having attribute value  $X_i = c$  and belonging to class  $y$
- Examples:

$$P(\text{Status} = \text{Married} | \text{No}) = 4/7$$

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

54

## Estimate Probabilities from Data

- For continuous attributes:
  - Discretization: Partition the range into bins:**
    - Replace continuous value with bin value
      - Attribute changed from continuous to ordinal
  - 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, use it to estimate the conditional probability  $P(X_i|Y)$

55

## 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	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

- Normal distribution:

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

- One for each  $(X_i, Y_j)$  pair

- For (Income, Class=No):

- If Class=No
  - sample mean = 110
  - sample variance = 2975

$$P(\text{Income} = 120 | \text{No}) = \frac{1}{\sqrt{2\pi(54.54)}} e^{-\frac{(120-110)^2}{2(2975)}} = 0.0072$$

56

## Example of Naïve Bayes Classifier

Given a Test Record:

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

Naïve Bayes Classifier:

$P(\text{Refund} = \text{Yes} | \text{No}) = 3/7$   
 $P(\text{Refund} = \text{No} | \text{No}) = 4/7$   
 $P(\text{Refund} = \text{Yes} | \text{Yes}) = 0$   
 $P(\text{Refund} = \text{No} | \text{Yes}) = 1$   
 $P(\text{Marital Status} = \text{Single} | \text{No}) = 2/7$   
 $P(\text{Marital Status} = \text{Divorced} | \text{No}) = 1/7$   
 $P(\text{Marital Status} = \text{Married} | \text{No}) = 4/7$   
 $P(\text{Marital Status} = \text{Single} | \text{Yes}) = 2/3$   
 $P(\text{Marital Status} = \text{Divorced} | \text{Yes}) = 1/3$   
 $P(\text{Marital Status} = \text{Married} | \text{Yes}) = 0$

For Taxable Income:

If class = No: sample mean = 110  
 sample variance = 2975  
 If class = Yes: sample mean = 90  
 sample variance = 25

- $P(X | \text{No}) = P(\text{Refund} = \text{No} | \text{No})$   
 $\times P(\text{Divorced} | \text{No})$   
 $\times P(\text{Income} = 120K | \text{No})$   
 $= 4/7 \times 1/7 \times 0.0072 = 0.0006$
- $P(X | \text{Yes}) = P(\text{Refund} = \text{No} | \text{Yes})$   
 $\times P(\text{Divorced} | \text{Yes})$   
 $\times P(\text{Income} = 120K | \text{Yes})$   
 $= 1 \times 1/3 \times 1.2 \times 10^{-9} = 4 \times 10^{-10}$

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

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

57

## Naïve Bayes Classifier can make decisions with partial information about attributes in the test record

Even in absence of information about any attributes, we can use Apriori Probabilities of Class Variable:

$P(\text{Yes}) = 3/10$   
 $P(\text{No}) = 7/10$

Naïve Bayes Classifier:

$P(\text{Refund} = \text{Yes} | \text{No}) = 3/7$   
 $P(\text{Refund} = \text{No} | \text{No}) = 4/7$   
 $P(\text{Refund} = \text{Yes} | \text{Yes}) = 0$   
 $P(\text{Refund} = \text{No} | \text{Yes}) = 1$   
 $P(\text{Marital Status} = \text{Single} | \text{No}) = 2/7$   
 $P(\text{Marital Status} = \text{Married} | \text{No}) = 1/7$   
 $P(\text{Marital Status} = \text{Single} | \text{Yes}) = 2/3$   
 $P(\text{Marital Status} = \text{Divorced} | \text{Yes}) = 1/3$   
 $P(\text{Marital Status} = \text{Married} | \text{Yes}) = 0$

For Taxable Income:

If class = No: sample mean = 110  
 sample variance = 2975  
 If class = Yes: sample mean = 90  
 sample variance = 25

If we only know that marital status is Divorced, then:

$P(\text{Yes} | \text{Divorced}) = 1/3 \times 3/10 / P(\text{Divorced})$   
 $P(\text{No} | \text{Divorced}) = 1/7 \times 7/10 / P(\text{Divorced})$

If we also know that Refund = No, then

$P(\text{Yes} | \text{Refund} = \text{No}, \text{Divorced}) = 1 \times 1/3 \times 3/10 / P(\text{Divorced}, \text{Refund} = \text{No})$   
 $P(\text{No} | \text{Refund} = \text{No}, \text{Divorced}) = 4/7 \times 1/7 \times 7/10 / P(\text{Divorced}, \text{Refund} = \text{No})$

If we also know that Taxable Income = 120, then

$P(\text{Yes} | \text{Refund} = \text{No}, \text{Divorced}, \text{Income} = 120) = 1.2 \times 10^{-9} \times 1 \times 1/3 \times 3/10 / P(\text{Divorced}, \text{Refund} = \text{No}, \text{Income} = 120)$   
 $P(\text{No} | \text{Refund} = \text{No}, \text{Divorced}, \text{Income} = 120) = 0.0072 \times 4/7 \times 1/7 \times 7/10 / P(\text{Divorced}, \text{Refund} = \text{No}, \text{Income} = 120)$

58

## Issues with Naïve Bayes Classifier

Given a Test Record:

$X = (\text{Married})$

Naïve Bayes Classifier:

$P(\text{Refund} = \text{Yes} | \text{No}) = 3/7$   
 $P(\text{Refund} = \text{No} | \text{No}) = 4/7$   
 $P(\text{Refund} = \text{Yes} | \text{Yes}) = 0$   
 $P(\text{Refund} = \text{No} | \text{Yes}) = 1$   
 $P(\text{Marital Status} = \text{Single} | \text{No}) = 2/7$   
 $P(\text{Marital Status} = \text{Divorced} | \text{No}) = 1/7$   
 $P(\text{Marital Status} = \text{Married} | \text{No}) = 4/7$   
 $P(\text{Marital Status} = \text{Single} | \text{Yes}) = 2/3$   
 $P(\text{Marital Status} = \text{Divorced} | \text{Yes}) = 1/3$   
 $P(\text{Marital Status} = \text{Married} | \text{Yes}) = 0$

For Taxable Income:

If class = No: sample mean = 110  
 sample variance = 2975  
 If class = Yes: sample mean = 90  
 sample variance = 25

$P(\text{Yes}) = 3/10$

$P(\text{No}) = 7/10$

$P(\text{Yes} | \text{Married}) = 0 \times 3/10 / P(\text{Married})$

$P(\text{No} | \text{Married}) = 4/7 \times 7/10 / P(\text{Married})$

59

## Issues with Naïve Bayes Classifier

Consider the table with Tid = 7 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
6	No	Married	60K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Naïve Bayes Classifier:

$P(\text{Refund} = \text{Yes} | \text{No}) = 2/6$   
 $P(\text{Refund} = \text{No} | \text{No}) = 4/6$   
 $P(\text{Refund} = \text{Yes} | \text{Yes}) = 0$   
 $P(\text{Refund} = \text{No} | \text{Yes}) = 1$   
 $P(\text{Marital Status} = \text{Single} | \text{No}) = 2/6$   
 $P(\text{Marital Status} = \text{Married} | \text{No}) = 0$   
 $P(\text{Marital Status} = \text{Single} | \text{Yes}) = 2/3$   
 $P(\text{Marital Status} = \text{Divorced} | \text{Yes}) = 1/3$   
 $P(\text{Marital Status} = \text{Married} | \text{Yes}) = 0/3$

For Taxable Income:

If class = No: sample mean = 91  
 sample variance = 685  
 If class = Yes: sample mean = 90  
 sample variance = 25

Given  $X = (\text{Refund} = \text{Yes}, \text{Divorced}, 120K)$

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

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

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

60

## 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:

$$\text{original: } P(X_i = c|y) = \frac{n_c}{n}$$

$$\text{Laplace Estimate: } P(X_i = c|y) = \frac{n_c + 1}{n + v}$$

$$\text{m - estimate: } P(X_i = c|y) = \frac{n_c + mp}{n + m}$$

$n$ : number of training instances belonging to class  $y$

$n_c$ : number of instances with  $X_i = c$  and  $Y = y$

$v$ : total number of attribute values that  $X_i$  can take

$p$ : initial estimate of  $(P(X_i = c|y))$  known apriori

$m$ : hyper-parameter for our confidence in  $p$

61

## Example of Naïve Bayes Classifier

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
penquin	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

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

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

62

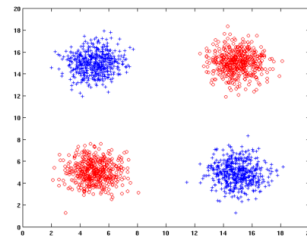
## Naïve Bayes (Summary)

- Robust to isolated noise points
- Handle missing values by ignoring the instance during probability estimate calculations
- Robust to irrelevant attributes
- Redundant and correlated attributes will violate class conditional assumption
  - Use other techniques such as Bayesian Belief Networks (BBN)

63

## Naïve Bayes

- How does Naïve Bayes perform on the following dataset?

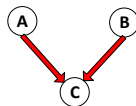


Conditional independence of attributes is violated

64

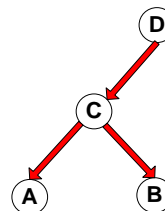
## Bayesian Belief Networks

- Provides graphical representation of probabilistic relationships among a set of random variables
- Consists of:
  - A directed acyclic graph (dag)
    - ◆ Node corresponds to a variable
    - ◆ Arc corresponds to dependence relationship between a pair of variables
  - A probability table associating each node to its immediate parent



65

## Conditional Independence



D is parent of C

A is child of C

B is descendant of D

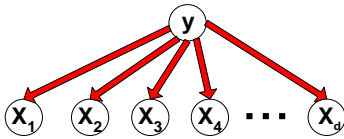
D is ancestor of A

- A node in a Bayesian network is conditionally independent of all of its nondescendants, if its parents are known

66

## Conditional Independence

- Naïve Bayes assumption:



67

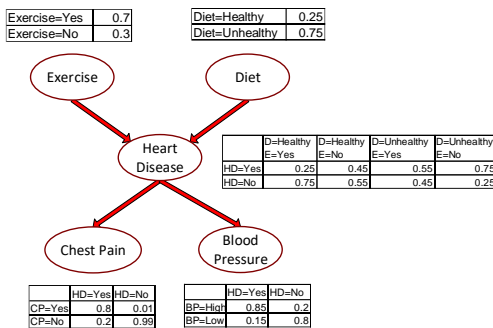
## Probability Tables

- If  $X$  does not have any parents, table contains prior probability  $P(X)$
- If  $X$  has only one parent ( $Y$ ), table contains conditional probability  $P(X|Y)$
- If  $X$  has multiple parents ( $Y_1, Y_2, \dots, Y_k$ ), table contains conditional probability  $P(X|Y_1, Y_2, \dots, Y_k)$



68

## Example of Bayesian Belief Network



69

## Example of Inferencing using BBN

- Given:  $X = (E=No, D=Yes, CP=Yes, BP=High)$   
 - Compute  $P(HD|E,D,CP,BP)?$

- $P(HD=Yes | E=No, D=Yes) = 0.55$   
 $P(CP=Yes | HD=Yes) = 0.8$   
 $P(BP=High | HD=Yes) = 0.85$   
 -  $P(HD=Yes | E=No, D=Yes, CP=Yes, BP=High)$   
 $\propto 0.55 \times 0.8 \times 0.85 = 0.374$
- $P(HD=No | E=No, D=Yes) = 0.45$   
 $P(CP=Yes | HD=No) = 0.01$   
 $P(BP=High | HD=No) = 0.2$   
 -  $P(HD=No | E=No, D=Yes, CP=Yes, BP=High)$   
 $\propto 0.45 \times 0.01 \times 0.2 = 0.0009$

**Classify X as Yes**

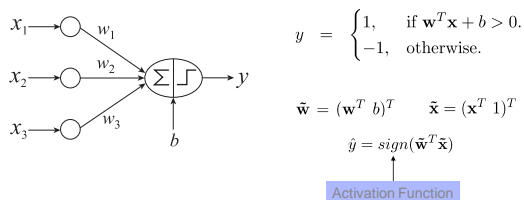
70

## Artificial Neural Networks (ANN)

- Basic Idea:** A complex non-linear function can be learned as a composition of simple processing units
- ANN is a collection of simple processing units (nodes) that are connected by directed links (edges)
  - Every node receives signals from incoming edges, performs computations, and transmits signals to outgoing edges
  - Analogous to *human brain* where nodes are neurons and signals are electrical impulses
  - Weight of an edge determines the strength of connection between the nodes
- Simplest ANN: **Perceptron** (single neuron)

72

## Basic Architecture of Perceptron

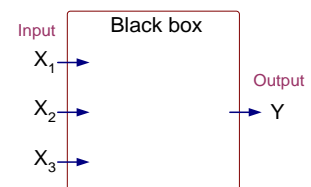


- Learns linear decision boundaries
- Related to logistic regression (activation function is sign instead of sigmoid)

73

## Perceptron Example

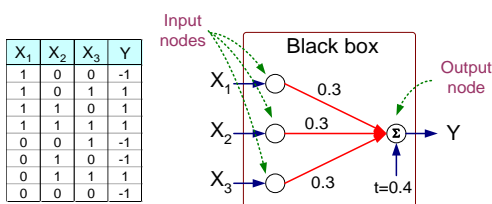
$X_1$	$X_2$	$X_3$	$Y$
1	0	0	-1
1	0	1	1
1	1	0	1
1	1	1	1
0	0	1	-1
0	1	0	-1
0	1	1	1
0	0	0	-1



Output Y is 1 if at least two of the three inputs are equal to 1.

74

## Perceptron Example



$$Y = \text{sign}(0.3X_1 + 0.3X_2 + 0.3X_3 - 0.4)$$

$$\text{where } \text{sign}(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ -1 & \text{if } x < 0 \end{cases}$$

75

## Perceptron Learning Rule

- Initialize the weights ( $w_0, w_1, \dots, w_d$ )
- Repeat

– For each training example ( $x_i, y_i$ )

- Compute  $\hat{y}_i$
- Update the weights:

$$w_j^{(k+1)} = w_j^{(k)} + \lambda(y_i - \hat{y}_i^{(k)})x_{ij}$$

- Until stopping condition is met
- k: iteration number;  $\lambda$ : learning rate

76

## Perceptron Learning Rule

- Weight update formula:

$$w_j^{(k+1)} = w_j^{(k)} + \lambda(y_i - \hat{y}_i^{(k)})x_{ij}$$

- Intuition:

– Update weight based on error:  $e = (y_i - \hat{y}_i)$

- If  $y = \hat{y}$ ,  $e=0$ : no update needed
- If  $y > \hat{y}$ ,  $e=2$ : weight must be increased (assuming  $x_{ij}$  is positive) so that  $\hat{y}$  will increase
- If  $y < \hat{y}$ ,  $e=-2$ : weight must be decreased (assuming  $x_{ij}$  is positive) so that  $\hat{y}$  will decrease

77

## Example of Perceptron Learning

$$\lambda = 0.1$$

$X_1$	$X_2$	$X_3$	$Y$
1	0	0	-1
1	0	1	1
1	1	0	1
1	1	1	1
0	0	1	-1
0	1	0	-1
0	1	1	1
0	0	0	-1

	$w_0$	$w_1$	$w_2$	$w_3$
0	0	0	0	0
1	-0.2	-0.2	0	0
2	0	0	0	0.2
3	0	0	0	0.2
4	0	0	0	0.2
5	-0.2	0	0	0
6	-0.2	0	0	0
7	0	0	0.2	0.2
8	-0.2	0	0.2	0.2

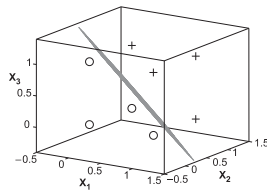
Weight updates over first epoch

Epoch	$w_0$	$w_1$	$w_2$	$w_3$
0	0	0	0	0
1	-0.2	0	0.2	0.2
2	-0.2	0	0.4	0.2
3	-0.4	0	0.4	0.2
4	-0.4	0.2	0.4	0.4
5	-0.6	0.2	0.4	0.2
6	-0.6	0.4	0.4	0.2

Weight updates over all epochs

## Perceptron Learning

- Since  $y$  is a linear combination of input variables, decision boundary is linear



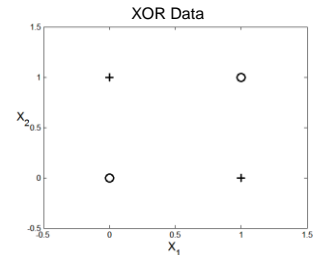
79

## Nonlinearly Separable Data

For nonlinearly separable problems, perceptron learning algorithm will fail because no linear hyperplane can separate the data perfectly

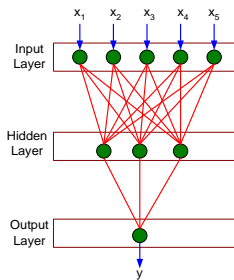
$$y = x_1 \oplus x_2$$

$x_1$	$x_2$	$y$
0	0	-1
1	0	1
0	1	1
1	1	-1



80

## Multi-layer Neural Network

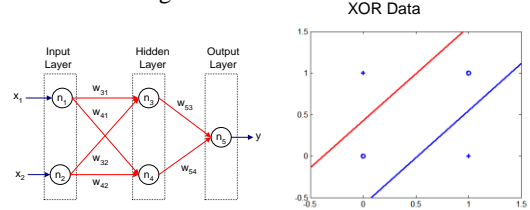


- More than one *hidden layer* of computing nodes
- Every node in a hidden layer operates on activations from preceding layer and transmits activations forward to nodes of next layer
- Also referred to as “feedforward neural networks”

81

## Multi-layer Neural Network

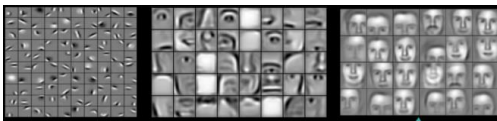
- Multi-layer neural networks with at least one hidden layer can solve any type of classification task involving nonlinear decision surfaces



82

## Why Multiple Hidden Layers?

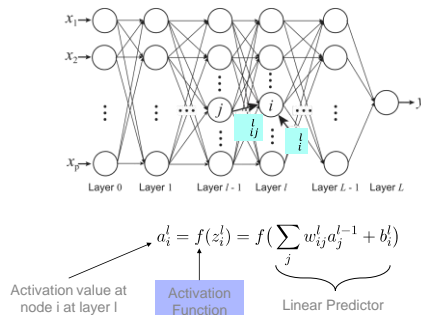
- Activations at hidden layers can be viewed as features extracted as functions of inputs
- Every hidden layer represents a level of abstraction
  - Complex features are compositions of simpler features



- Number of layers is known as **depth** of ANN
  - Deeper networks express complex hierarchy of features

83

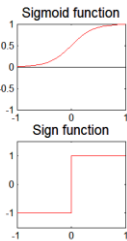
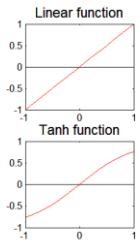
## Multi-Layer Network Architecture



84

## Activation Functions

$$a_i^l = f(z_i^l) = f\left(\sum_j w_{ij}^l a_j^{l-1} + b_i^l\right)$$



$$a_i^l = \sigma(z_i^l) = \frac{1}{1 + e^{-z_i^l}}$$

$$\frac{\partial a_i^l}{\partial z_i^l} = \frac{\partial \sigma(z_i^l)}{\partial z_i^l} = a_i^l(1 - a_i^l)$$

85

## Learning Multi-layer Neural Network

- Can we apply perceptron learning rule to each node, including hidden nodes?
  - Perceptron learning rule computes error term  $e = y - \hat{y}$  and updates weights accordingly
    - Problem: how to determine the true value of  $y$  for hidden nodes?
  - Approximate error in hidden nodes by error in the output nodes
    - Problem:
      - Not clear how adjustment in the hidden nodes affect overall error
      - No guarantee of convergence to optimal solution

86

## Gradient Descent

- Loss Function to measure errors across all training points

$$E(\mathbf{w}, \mathbf{b}) = \sum_{k=1}^n \text{Loss}(y_k, \hat{y}_k) \quad \text{Squared Loss:} \quad \text{Loss}(y_k, \hat{y}_k) = (y_k - \hat{y}_k)^2$$

- Gradient descent: Update parameters in the direction of “maximum descent” in the loss function across all points

$$w_{ij}^l \leftarrow w_{ij}^l - \lambda \frac{\partial E}{\partial w_{ij}^l}, \quad \lambda: \text{learning rate}$$

$$b_i^l \leftarrow b_i^l - \lambda \frac{\partial E}{\partial b_i^l}$$

- Stochastic gradient descent (SGD), update the weight for every instance (minibatch SGD: update over min-batches of instances)

87

## Computing Gradients

$$\frac{\partial E}{\partial w_{ij}^l} = \sum_{k=1}^n \frac{\partial \text{Loss}(y_k, \hat{y}_k)}{\partial w_{ij}^l}, \quad \hat{y} = a^L$$

$$a_i^l = f(z_i^l) = f\left(\sum_j w_{ij}^l a_j^{l-1} + b_i^l\right)$$

- Using chain rule of differentiation (on a single instance):

$$\frac{\partial \text{Loss}}{\partial w_{ij}^l} = \frac{\partial \text{Loss}}{\partial a_i^l} \times \frac{\partial a_i^l}{\partial z_i^l} \times \frac{\partial z_i^l}{\partial w_{ij}^l}$$

- For sigmoid activation function:

$$\frac{\partial \text{Loss}}{\partial w_{ij}^l} = \delta_i^l \times a_i^l(1 - a_i^l) \times a_j^{l-1}$$

$$\text{where } \delta_i^l = \frac{\partial \text{Loss}}{\partial a_i^l}$$

- How can we compute  $\delta_i^l$  for every layer?

88

## Backpropagation Algorithm

- At output layer  $L$ :

$$\delta^L = \frac{\partial \text{Loss}}{\partial a^L} = \frac{\partial (y - a^L)^2}{\partial a^L} = 2(a^L - y)$$

- At a hidden layer  $l$  (using chain rule):

$$\delta_j^l = \sum_i (\delta_i^{l+1} \times a_i^{l+1}(1 - a_i^{l+1}) \times w_{ij}^{l+1})$$

- Gradients at layer  $l$  can be computed using gradients at layer  $l+1$
- Start from layer  $L$  and “backpropagate” gradients to all previous layers
- Use gradient descent to update weights at every epoch
- For next epoch, use updated weights to compute loss fn. and its gradient
- Iterate until convergence (loss does not change)

89

## Design Issues in ANN

- Number of nodes in input layer
  - One input node per **binary/continuous** attribute
  - $k$  or  $\log_2 k$  nodes for each **categorical** attribute with  $k$  values
- Number of nodes in output layer
  - One output for **binary** class problem
  - $k$  or  $\log_2 k$  nodes for **k-class** problem
- Number of hidden layers and nodes per layer
- Initial weights and biases
- Learning rate, max. number of epochs, mini-batch size for mini-batch SGD, ...

90

## Characteristics of ANN

- Multilayer ANN are universal approximators but could suffer from overfitting if the network is too large
  - Naturally represents a hierarchy of features at multiple levels of abstractions
- Gradient descent may converge to local minimum
- Model building is compute intensive, but testing is fast
- Can handle redundant and irrelevant attributes because weights are automatically learnt for all attributes
- Sensitive to noise in training data
  - This issue can be addressed by incorporating model complexity in the loss function
- Difficult to handle missing attributes

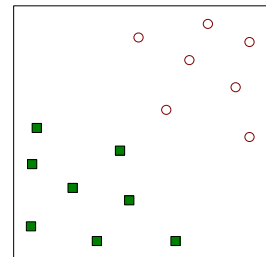
91

## Deep Learning Trends

- Training **deep** neural networks (more than 5-10 layers) could only be possible in recent times with:
  - Faster computing resources (GPU)
  - Larger labeled training sets
- Algorithmic Improvements in Deep Learning
  - Responsive activation functions (e.g., ReLU)
  - Regularization (e.g., Dropout)
  - Supervised pre-training
  - Unsupervised pre-training (auto-encoders)
- Specialized ANN Architectures:
  - Convolutional Neural Networks (for image data)
  - Recurrent Neural Networks (for sequence data)
  - Residual Networks (with skip connections)
- Generative Models: Generative Adversarial Networks

92

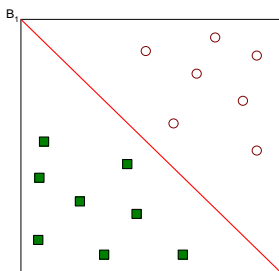
## Support Vector Machines



- Find a linear hyperplane (decision boundary) that will separate the data

94

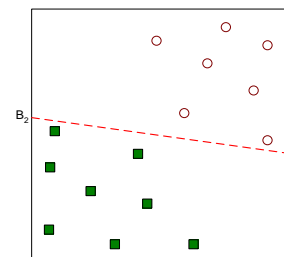
## Support Vector Machines



- One Possible Solution

95

## Support Vector Machines

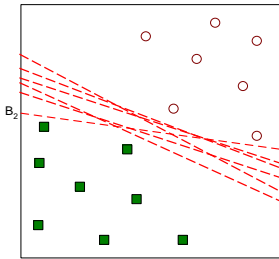


- Another possible solution

96



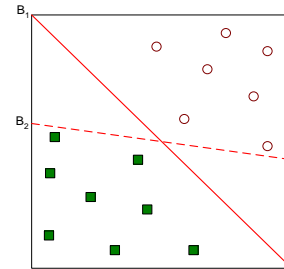
## Support Vector Machines



- Other possible solutions

97

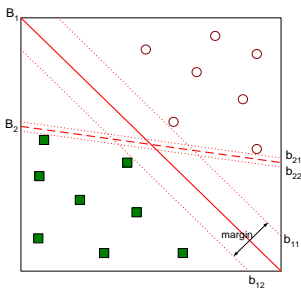
## Support Vector Machines



- Which one is better? B1 or B2?
- How do you define better?

98

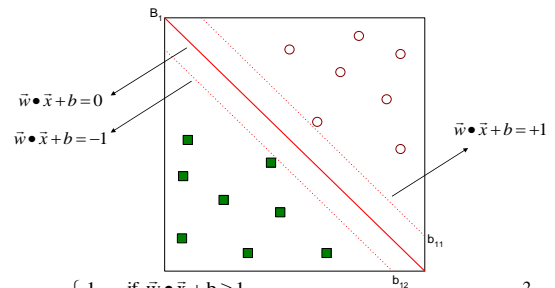
## Support Vector Machines



- Find hyperplane **maximizes** the margin => B1 is better than B2

99

## Support Vector Machines



$$f(\vec{x}) = \begin{cases} 1 & \text{if } \vec{w} \bullet \vec{x} + b \geq 1 \\ -1 & \text{if } \vec{w} \bullet \vec{x} + b \leq -1 \end{cases}$$

$$\text{Margin} = \frac{2}{\|\vec{w}\|}$$

100

## Linear SVM

- Linear model:

$$f(\vec{x}) = \begin{cases} 1 & \text{if } \vec{w} \bullet \vec{x} + b \geq 1 \\ -1 & \text{if } \vec{w} \bullet \vec{x} + b \leq -1 \end{cases}$$

- Learning the model is equivalent to determining the values of  $\vec{w}$  and  $b$ 
  - How to find  $\vec{w}$  and  $b$  from training data?

101

## Learning Linear SVM

- Objective is to maximize:  $\text{Margin} = \frac{2}{\|\vec{w}\|}$

– Which is equivalent to minimizing:

– Subject to the following constraints:

$$y_i = \begin{cases} 1 & \text{if } \vec{w} \bullet \vec{x}_i + b \geq 1 \\ -1 & \text{if } \vec{w} \bullet \vec{x}_i + b \leq -1 \end{cases}$$

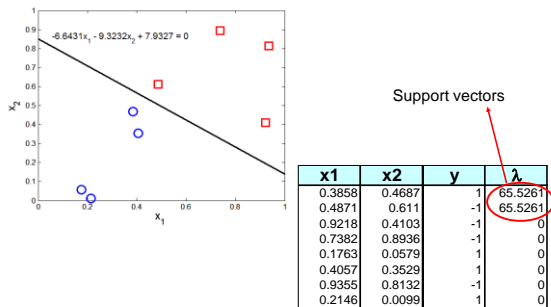
or

$$y_i(\vec{w} \bullet \vec{x}_i + b) \geq 1, \quad i = 1, 2, \dots, N$$

- This is a constrained optimization problem
  - Solve it using Lagrange multiplier method

102

## Example of Linear SVM



103

## Learning Linear SVM

- Decision boundary depends only on support vectors

– If you have data set with same support vectors, decision boundary will not change

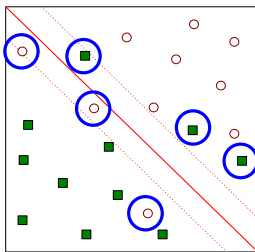
– How to classify using SVM once  $\mathbf{w}$  and  $b$  are found?  
Given a test record,  $\mathbf{x}_i$

$$f(\vec{x}_i) = \begin{cases} 1 & \text{if } \vec{w} \bullet \vec{x}_i + b \geq 1 \\ -1 & \text{if } \vec{w} \bullet \vec{x}_i + b \leq -1 \end{cases}$$

104

## Support Vector Machines

- What if the problem is not linearly separable?



105

## Support Vector Machines

- What if the problem is not linearly separable?

– Introduce slack variables

- Need to minimize:

$$L(w) = \frac{\|\vec{w}\|^2}{2} + C \left( \sum_{i=1}^N \xi_i^k \right)$$

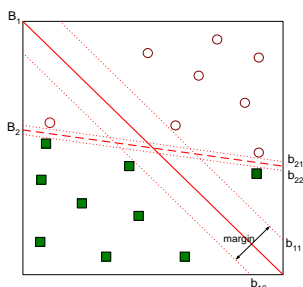
- Subject to:

$$y_i = \begin{cases} 1 & \text{if } \vec{w} \bullet \vec{x}_i + b \geq 1 - \xi_i \\ -1 & \text{if } \vec{w} \bullet \vec{x}_i + b \leq -1 + \xi_i \end{cases}$$

- If  $k$  is 1 or 2, this leads to similar objective function as linear SVM but with different constraints (see textbook)

106

## Support Vector Machines

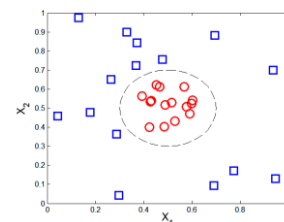


- Find the hyperplane that optimizes both factors

107

## Nonlinear Support Vector Machines

- What if decision boundary is not linear?

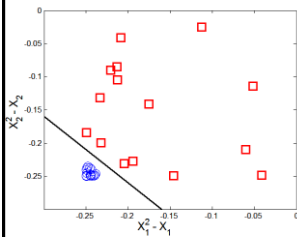


$$g(x_1, x_2) = \begin{cases} 1 & \text{if } \sqrt{(x_1 - 0.5)^2 + (x_2 - 0.5)^2} > 0.2 \\ -1 & \text{otherwise} \end{cases}$$

108

## Nonlinear Support Vector Machines

- Transform data into higher dimensional space



$$x_1^2 - x_1 + x_2^2 - x_2 = -0.46.$$

$$\Phi : (x_1, x_2) \longrightarrow (x_1^2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, 1).$$

$$w_4x_1^2 + w_3x_2^2 + w_2\sqrt{2}x_1 + w_1\sqrt{2}x_2 + w_0 = 0.$$

Decision boundary:

$$\vec{w} \bullet \Phi(\vec{x}) + b = 0$$

109

## Learning Nonlinear SVM

- Optimization problem:

$$\min_w \frac{\|w\|^2}{2} \quad \text{subject to} \quad y_i(w \cdot \Phi(x_i) + b) \geq 1, \quad \forall \{x_i, y_i\}$$

- Which leads to the same set of equations (but involve  $\Phi(x)$  instead of  $x$ )

$$L_D = \sum_{i=1}^n \lambda_i - \frac{1}{2} \sum_{i,j} \lambda_i \lambda_j y_i y_j \Phi(x_i) \cdot \Phi(x_j) \quad w = \sum_i \lambda_i y_i \Phi(x_i)$$

$$\lambda_i \{y_i (\sum_j \lambda_j y_j \Phi(x_j) \cdot \Phi(x_i) + b) - 1\} = 0,$$

$$f(z) = \text{sign}(w \cdot \Phi(z) + b) = \text{sign}(\sum_{i=1}^n \lambda_i y_i \Phi(x_i) \cdot \Phi(z) + b).$$

110

## Learning NonLinear SVM

- Issues:
  - What type of mapping function  $\Phi$  should be used?
  - How to do the computation in high dimensional space?
    - Most computations involve dot product  $\Phi(x_i) \bullet \Phi(x_j)$
    - Curse of dimensionality?

111

## Learning Nonlinear SVM

- Kernel Trick:
  - $\Phi(x_i) \bullet \Phi(x_j) = K(x_i, x_j)$
  - $K(x_i, x_j)$  is a kernel function (expressed in terms of the coordinates in the original space)
    - Examples:

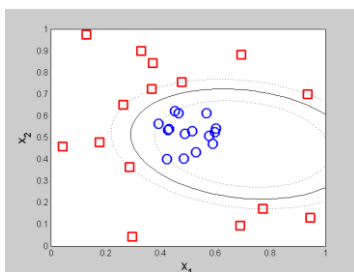
$$K(x, y) = (x \cdot y + 1)^p$$

$$K(x, y) = e^{-\|x - y\|^2 / (2\sigma^2)}$$

$$K(x, y) = \tanh(kx \cdot y - \delta)$$

112

## Example of Nonlinear SVM



SVM with polynomial degree 2 kernel

113

## Learning Nonlinear SVM

- Advantages of using kernel:
  - Don't have to know the mapping function  $\Phi$
  - Computing dot product  $\Phi(x_i) \bullet \Phi(x_j)$  in the original space avoids curse of dimensionality
- Not all functions can be kernels
  - Must make sure there is a corresponding  $\Phi$  in some high-dimensional space
  - Mercer's theorem (see textbook)

114

## Characteristics of SVM

- The learning problem is formulated as a convex optimization problem
  - Efficient algorithms are available to find the global minima
  - Many of the other methods use greedy approaches and find locally optimal solutions
  - High computational complexity for building the model
- Robust to noise
- Overfitting is handled by maximizing the margin of the decision boundary.
- SVM can handle irrelevant and redundant attributes better than many other techniques
- The user needs to provide the type of kernel function and cost function
- Difficult to handle missing values
- What about categorical variables?

115

## Ensemble Methods

- Construct a set of base classifiers learned from the training data
- Predict class label of test records by combining the predictions made by multiple classifiers (e.g., by taking majority vote)

117

## Example: Why Do Ensemble Methods Work?

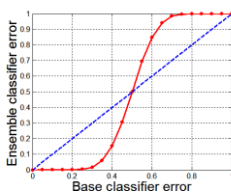
- Suppose there are 25 base classifiers
  - Each classifier has error rate,  $\epsilon = 0.35$
  - Majority vote of classifiers used for classification
  - If all classifiers are identical:
    - ♦ Error rate of ensemble =  $\epsilon$  (0.35)
  - If all classifiers are independent (errors are uncorrelated):
    - ♦ Error rate of ensemble = probability of having more than half of base classifiers being wrong

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

118

## Necessary Conditions for Ensemble Methods

- Ensemble Methods work better than a single base classifier if:
  1. All base classifiers are independent of each other
  2. All base classifiers perform better than random guessing (error rate < 0.5 for binary classification)



Classification error for an ensemble of 25 base classifiers, assuming their errors are uncorrelated.

119

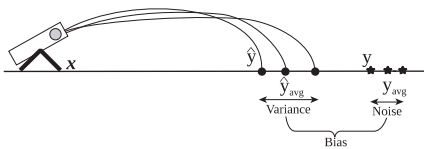
## Rationale for Ensemble Learning

- Ensemble Methods work best with **unstable base classifiers**
  - Classifiers that are sensitive to minor perturbations in training set, due to *high model complexity*
  - Examples: Unpruned decision trees, ANNs, ...

120

## Bias-Variance Decomposition

- Analogous problem of reaching a target  $y$  by firing projectiles from  $x$  (regression problem)

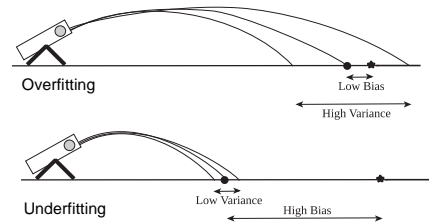


- For classification, the generalization error of model  $m$  can be given by:

$$gen.error(m) = c_1 + bias(m) + c_2 \times variance(m)$$

121

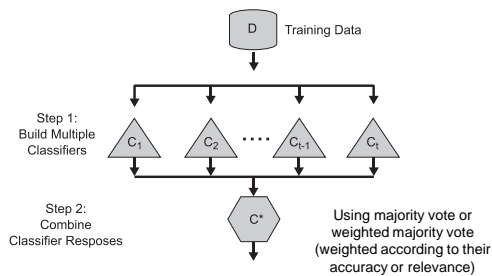
## Bias-Variance Trade-off and Overfitting



- Ensemble methods try to reduce the variance of complex models (with low bias) by *aggregating* responses of multiple base classifiers

122

## General Approach of Ensemble Learning



123

## Constructing Ensemble Classifiers

- By manipulating training set
  - Example: bagging, boosting, random forests
- By manipulating input features
  - Example: random forests
- By manipulating class labels
  - Example: error-correcting output coding
- By manipulating learning algorithm
  - Example: injecting randomness in the initial weights of ANN

124

## Bagging (Bootstrap AGGREGATING)

- Bootstrap sampling: sampling with replacement

Original Data	1	2	3	4	5	6	7	8	9	10
Bagging (Round 1)	7	8	10	8	2	5	10	10	5	9
Bagging (Round 2)	1	4	9	1	2	3	2	7	3	2
Bagging (Round 3)	1	8	5	10	5	5	9	6	3	7

- Build classifier on each bootstrap sample
- Probability of a training instance being selected in a bootstrap sample is:
  - $1 - (1 - 1/n)^n$  ( $n$ : number of training instances)
  - $\sim 0.632$  when  $n$  is large

125

## Bagging Algorithm

### Algorithm 4.5 Bagging algorithm.

- Let  $k$  be the number of bootstrap samples.
- for  $i = 1$  to  $k$  do
- Create a bootstrap sample of size  $N$ ,  $D_i$ .
- Train a base classifier  $C_i$  on the bootstrap sample  $D_i$ .
- end for
- $C^*(x) = \arg\max_y \sum_i \delta(C_i(x) = y)$ .

$\{\delta(\cdot) = 1$  if its argument is true and 0 otherwise. $\}$

126

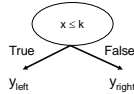
## Bagging Example

- 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 of size 1)
  - Decision rule:  $x \leq k$  versus  $x > k$
  - Split point  $k$  is chosen based on entropy



127

## 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

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

128

## 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

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

129

## 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

$x \leq 0.95 \Rightarrow y = 1$   
 $x > 0.95 \Rightarrow y = 1$

130

## Bagging Example

- Summary of Trained Decision Stumps:

Round	Split Point	Left Class	Right Class
1	0.35	1	-1
2	0.7	1	1
3	0.35	1	-1
4	0.3	1	-1
5	0.35	1	-1
6	0.75	-1	1
7	0.75	-1	1
8	0.75	-1	1
9	0.75	-1	1
10	0.05	1	1

131

## Bagging Example

- Use majority vote (sign of sum of predictions) 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
Predicted Class	Sign	1	1	1	-1	-1	-1	-1	1	1

- Bagging can also increase the complexity (representation capacity) of simple classifiers such as decision stumps

132

## Boosting

- An iterative procedure to adaptively change distribution of training data by focusing more on previously misclassified records
  - Initially, all  $N$  records are assigned equal weights (for being selected for training)
  - Unlike bagging, weights may change at the end of each boosting round

133

## Boosting

- Records that are wrongly classified will have their weights increased in the next round
- Records that are classified correctly will have their weights decreased in the next round

Original Data	1	2	3	4	5	6	7	8	9	10
Boosting (Round 1)	7	3	2	8	7	9	4	10	6	3
Boosting (Round 2)	5	4	9	4	2	5	1	7	4	2
Boosting (Round 3)	4	4	8	10	4	5	4	6	3	4

- Example 4 is hard to classify
- Its weight is increased, therefore it is more likely to be chosen again in subsequent rounds

134

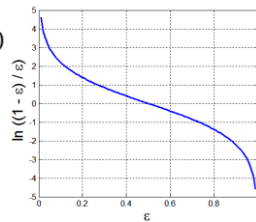
## AdaBoost

- Base classifiers:  $C_1, C_2, \dots, C_T$
- Error rate of a base classifier:

$$\epsilon_i = \frac{1}{N} \sum_{j=1}^N w_j^{(i)} \delta(C_i(x_j) \neq y_j)$$

- Importance of a classifier:

$$\alpha_i = \frac{1}{2} \ln \left( \frac{1 - \epsilon_i}{\epsilon_i} \right)$$



135

## AdaBoost Algorithm

- Weight update:

$$w_j^{(i+1)} = \frac{w_j^{(i)}}{Z_i} \times \begin{cases} e^{-\alpha_i} & \text{if } C_i(x_j) = y_j \\ e^{\alpha_i} & \text{if } C_i(x_j) \neq y_j \end{cases}$$

Where  $Z_i$  is the normalization factor

- If any intermediate rounds produce error rate higher than 50%, the weights are reverted back to  $1/n$  and the resampling procedure is repeated
- Classification:  $\hat{y} = \arg \max_y \sum_{i=1}^T \alpha_i \delta(C_i(x) = y)$

136

## AdaBoost Algorithm

### Algorithm 4.6 AdaBoost algorithm.

- $\mathbf{w} = \{w_j = 1/N \mid j = 1, 2, \dots, N\}$ . {Initialize the weights for all  $N$  examples.}
- Let  $k$  be the number of boosting rounds.
- for**  $i = 1$  to  $k$  **do**
- Create training set  $D_i$  by sampling (with replacement) from  $D$  according to  $\mathbf{w}$ .
- Train a base classifier  $C_i$  on  $D_i$ .
- Apply  $C_i$  to all examples in the original training set,  $D$ .
- $\epsilon_i = \frac{1}{N} [\sum_j w_j \delta(C_i(x_j) \neq y_j)]$  {Calculate the weighted error.}
- if**  $\epsilon_i > 0.5$  **then**
- $\mathbf{w} = \{w_j = 1/N \mid j = 1, 2, \dots, N\}$ . {Reset the weights for all  $N$  examples.}
- end if**
- $\alpha_i = \frac{1}{2} \ln \frac{1 - \epsilon_i}{\epsilon_i}$ .
- Update the weight of each example according to Equation 4.103.
- end for**
- $C^*(\mathbf{x}) = \arg \max_y \sum_{j=1}^T \alpha_j \delta(C_j(\mathbf{x}) = y)$ .

137

## AdaBoost 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 rule:  $x \leq k$  versus  $x > k$
  - Split point  $k$  is chosen based on entropy



138

## AdaBoost Example

- Training sets for the first 3 boosting rounds:

Boosting Round 1:

x	0.1	0.4	0.5	0.6	0.6	0.7	0.7	0.7	0.8	1
y	1	-1	-1	-1	-1	-1	-1	-1	1	1

Boosting Round 2:

x	0.1	0.1	0.2	0.2	0.2	0.2	0.3	0.3	0.3	0.3
y	1	1	1	1	1	1	1	1	1	1

Boosting Round 3:

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

- Summary:

Round	Split Point	Left Class	Right Class	alpha
1	0.75	-1	1	1.738
2	0.05	1	1	2.7784
3	0.3	1	-1	4.1195

139

## AdaBoost Example

- Weights

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	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
2	0.311	0.311	0.311	0.01	0.01	0.01	0.01	0.01	0.01	0.01
3	0.029	0.029	0.029	0.228	0.228	0.228	0.228	0.009	0.009	0.009

- Classification

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
Sum	5.16	5.16	5.16	-3.08	-3.08	-3.08	-3.08	0.397	0.397	0.397
Predicted Class	1	1	1	-1	-1	-1	-1	1	1	1

140

## Random Forest Algorithm

- Construct an ensemble of decision trees by manipulating training set as well as features
  - Use bootstrap sample to train every decision tree (similar to Bagging)
  - Use the following tree induction algorithm:
    - At every internal node of decision tree, randomly sample  $p$  attributes for selecting split criterion
    - Repeat this procedure until all leaves are pure (unpruned tree)

141

## Characteristics of Random Forest

- Base classifiers are unpruned trees and hence are *unstable classifiers*
- Base classifiers are *decorrelated* (due to randomization in training set as well as features)
- Random forests reduce variance of unstable classifiers without negatively impacting the bias
- Selection of hyper-parameter  $p$ 
  - Small value ensures lack of correlation
  - High value promotes strong base classifiers
  - Common default choices:  $\sqrt{d}$ ,  $\log_2(d + 1)$

142

## Gradient Boosting

- Constructs a series of models
  - Models can be any predictive model that has a differentiable loss function
  - Commonly, trees are the chosen model
    - XGboost (extreme gradient boosting) is a popular package because of its impressive performance
- Boosting can be viewed as optimizing the loss function by iterative functional gradient descent.
- Implementations of various boosted algorithms are available in Python, R, Matlab, and more.

143

144



## Class Imbalance Problem

- 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
  - COVID-19 test results on a random sample
- **Key Challenge:**
  - Evaluation measures such as accuracy are not well-suited for imbalanced class

145

## Confusion Matrix

- Confusion Matrix:

	PREDICTED CLASS		
ACTUAL CLASS		Class=Yes	Class=No
	Class=Yes	a	b
	Class=No	c	d

a: TP (true positive)  
 b: FN (false negative)  
 c: FP (false positive)  
 d: TN (true negative)

146

## Accuracy

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

- Most widely-used metric:

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

147

## 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 this trivial model does not detect any class YES example
  - Detecting the rare class is usually more interesting (e.g., frauds, intrusions, defects, etc)

	PREDICTED CLASS		
ACTUAL CLASS		Class=Yes	Class=No
	Class=Yes	0	10
	Class=No	0	990

148

## Which model is better?

A

	PREDICTED		
ACTUAL		Class=Yes	Class=No
	Class=Yes	0	10
	Class=No	0	990

Accuracy: 99%

B

	PREDICTED		
ACTUAL		Class=Yes	Class=No
	Class=Yes	10	0
	Class=No	500	490

Accuracy: 50%

149

## Which model is better?

A

	PREDICTED		
ACTUAL		Class=Yes	Class=No
	Class=Yes	5	5
	Class=No	0	990

B

	PREDICTED		
ACTUAL		Class=Yes	Class=No
	Class=Yes	10	0
	Class=No	500	490

150

## Alternative Measures

ACTUAL CLASS	PREDICTED CLASS	
	Class=Yes	Class=No
Class=Yes	a	b
Class=No	c	d

$$\text{Precision (p)} = \frac{a}{a+c}$$

$$\text{Recall (r)} = \frac{a}{a+b}$$

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

151

## Alternative Measures

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

$$\text{Precision (p)} = \frac{10}{10+0} = 0.5$$

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

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

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

152

## Alternative Measures

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

$$\text{Precision (p)} = \frac{10}{10+0} = 0.5$$

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

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

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

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

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

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

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

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

153

## Which of these classifiers is better?

A

ACTUAL CLASS	PREDICTED CLASS	
	Class=Yes	Class=No
Class=Yes	40	10
Class=No	10	40

$$\text{Precision (p)} = 0.8$$

$$\text{Recall (r)} = 0.8$$

$$\text{F-measure (F)} = 0.8$$

$$\text{Accuracy} = 0.8$$

B

ACTUAL CLASS	PREDICTED CLASS	
	Class=Yes	Class=No
Class=Yes	40	10
Class=No	1000	4000

$$\text{Precision (p)} \approx 0.04$$

$$\text{Recall (r)} = 0.8$$

$$\text{F-measure (F)} \approx 0.08$$

$$\text{Accuracy} \approx 0.8$$

154

## Measures of Classification Performance

ACTUAL CLASS	PREDICTED CLASS	
	Yes	No
	Yes	No
Yes	TP	FN
No	FP	TN

$\alpha$  is the probability that we reject the null hypothesis when it is true. This is a Type I error or a false positive (FP).

$\beta$  is the probability that we accept the null hypothesis when it is false. This is a Type II error or a false negative (FN).

$$\text{Accuracy} = \frac{TP + TN}{TP + FN + FP + TN}$$

$$\text{ErrorRate} = 1 - \text{accuracy}$$

$$\text{Precision} = \text{Positive Predictive Value} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \text{Sensitivity} = \text{TP Rate} = \frac{TP}{TP + FN}$$

$$\text{Specificity} = \text{TN Rate} = \frac{TN}{TN + FP}$$

$$\text{FP Rate} = \alpha = \frac{FP}{TN + FP} = 1 - \text{specificity}$$

$$\text{FN Rate} = \beta = \frac{FN}{FN + TP} = 1 - \text{sensitivity}$$

$$\text{Power} = \text{sensitivity} = 1 - \beta$$

155

## Alternative Measures

A

ACTUAL CLASS	PREDICTED CLASS	
	Class=Yes	Class=No
Class=Yes	40	10
Class=No	10	40

$$\text{Precision (p)} = 0.8$$

$$\text{TPR} = \text{Recall (r)} = 0.8$$

$$\text{FPR} = 0.2$$

$$\text{F-measure (F)} = 0.8$$

$$\text{Accuracy} = 0.8$$

$$\frac{\text{TPR}}{\text{FPR}} = 4$$

B

ACTUAL CLASS	PREDICTED CLASS	
	Class=Yes	Class=No
Class=Yes	40	10
Class=No	1000	4000

$$\text{Precision (p)} = 0.038$$

$$\text{TPR} = \text{Recall (r)} = 0.8$$

$$\text{FPR} = 0.2$$

$$\text{F-measure (F)} = 0.07$$

$$\text{Accuracy} = 0.8$$

$$\frac{\text{TPR}}{\text{FPR}} = 4$$

156

# Which of these classifiers is better?

A	PREDICTED CLASS		
ACTUAL CLASS		Class=Yes	Class=No
	Class=Yes	10	40
	Class=No	10	40

Precision (p) = 0.5

TPR = Recall (r) = 0.2

FPR = 0.2

F - measure = 0.28

B	PREDICTED CLASS		
ACTUAL CLASS		Class=Yes	Class=No
	Class=Yes	25	25
	Class=No	25	25

Precision (p) = 0.5

TPR = Recall (r) = 0.5

FPR = 0.5

F - measure = 0.5

C	PREDICTED CLASS		
ACTUAL CLASS		Class=Yes	Class=No
	Class=Yes	40	10
	Class=No	40	10

Precision (p) = 0.5

TPR = Recall (r) = 0.8

FPR = 0.8

F - measure = 0.61

157

A	PREDICTED CLASS		
ACTUAL CLASS		Class=Yes	Class=No
	Class=Yes	10	40
	Class=No	10	40

Precision (p) = 0.5  
TPR = Recall (r) = 0.2  
FPR = 0.2  
F-measure = 0.28

B	PREDICTED CLASS		
ACTUAL CLASS		Class=Yes	Class=No
	Class=Yes	25	25
	Class=No	25	25

Precision (p) = 0.5  
 TPR = Recall (r) = 0.5  
 FPR = 0.5  
 F-measure = 0.5

C	PREDICTED CLASS		
ACTUAL CLASS		Class=Yes	Class=No
	Class=Yes	40	10
	Class=No	40	10

Precision (p) = 0.5  
TPR = Recall (r) = 0.8  
FPR = 0.8  
F-measure = 0.61

15

## 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

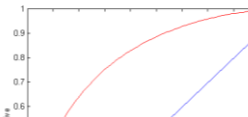
- A graphical approach for displaying trade-off between detection rate and false alarm rate
- Developed in 1950s for signal detection theory to analyze noisy signals
- ROC curve plots TPR against FPR
  - Performance of a model represented as a point in an ROC curve

158

# ROC Curve

(TPR,FPR):

- (0,0): declare everything to be negative class
- (1,1): declare everything to be positive class
- (1,0): ideal
- Diagonal line:
  - Random guessing
  - Below diagonal line:
    - prediction is opposite of the true class



The image shows a Receiver Operating Characteristic (ROC) Curve. The x-axis is labeled 'False Positive' and the y-axis is labeled 'True Positive', both ranging from 0 to 1. A diagonal blue line represents random guessing. A red curve is plotted above the diagonal line, indicating a classifier's performance that is better than random. The red curve starts at (0,0) and ends at (1,1), staying consistently above the diagonal line.

- (0,0): declare everything to be negative class
- (1,1): declare everything to be positive class
- (1,0): ideal

- 

159

# ROC (Receiver Operating Characteristic)

- To draw ROC curve, classifier must produce continuous-valued output
  - Outputs are used to rank test records, from the most likely positive class record to the least likely positive class record
  - By using different thresholds on this value, we can create different variations of the classifier with TPR/FPR tradeoffs
- Many classifiers produce only discrete outputs (i.e., predicted class)
  - How to get continuous-valued outputs?
    - Decision trees, rule-based classifiers, neural networks, Bayesian classifiers, k-nearest neighbors, SVM

160

- To draw ROC curve, classifier must produce continuous-valued output
  - Outputs are used to rank test records, from the most likely positive class record to the least likely positive class record
  - By using different thresholds on this value, we can create different variations of the classifier with TPR/FPR tradeoffs
- Many classifiers produce only discrete outputs (i.e., predicted class)
  - How to get continuous-valued outputs?
    - Decision trees, rule-based classifiers, neural networks, Bayesian classifiers, k-nearest neighbors, SVM

160

# Example: Decision Trees

Decision Tree

Continuous-valued outputs

161

```

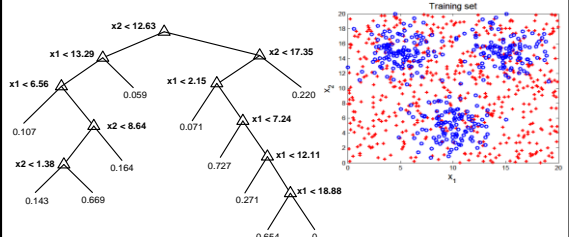
graph TD
    Root(( )) -- "x2 < 12.83" --> L1(( ))
    Root -- "x2 < 12.83" --> R1(( ))
    L1 -- "x1 < 13.29" --> L1L(( ))
    L1 -- "x1 < 13.29" --> L1R(( ))
    R1 -- "x2 < 17.35" --> R1L(( ))
    R1 -- "x2 < 17.35" --> R1R(( ))
    L1L -- "x1 < 6.56" --> L1LL(( ))
    L1L -- "x1 < 6.56" --> L1LR(( ))
    L1R -- "x2 < 8.64" --> L1RL(( ))
    L1R -- "x2 < 8.64" --> L1RR(( ))
    R1L -- "x1 < 2.15" --> R1LL(( ))
    R1L -- "x1 < 2.15" --> R1LR(( ))
    R1R -- "x1 < 7.24" --> R1RL(( ))
    R1R -- "x1 < 7.24" --> R1RR(( ))
    L1LL --> L1LLL[0.107]
    L1LR --> L1LRL[0.059]
    L1RL --> L1RLL[0.164]
    L1RR --> L1RRR[0.069]
    R1LL --> R1LLL[0.071]
    R1LR --> R1LRL[0.220]
    R1RL --> R1RLL[0.271]
    R1RR --> R1RRR[0.064]
    L1LLL --> L1LLL1[0.143]
    L1LLL --> L1LLL2[0.138]
    L1LRL --> L1LRL1[0.107]
    L1LRL --> L1LRL2[0.056]
    L1RLL --> L1RLL1[0.143]
    L1RLL --> L1RLL2[0.138]
    L1RRR --> L1RRR1[0.143]
    L1RRR --> L1RRR2[0.138]
    R1LLL --> R1LLL1[0.143]
    R1LLL --> R1LLL2[0.138]
    R1LRL --> R1LRL1[0.143]
    R1LRL --> R1LRL2[0.138]
    R1RLL --> R1RLL1[0.143]
    R1RLL --> R1RLL2[0.138]
    R1RRR --> R1RRR1[0.143]
    R1RRR --> R1RRR2[0.138]
  
```

16

# ROC Curve Example

		Predicted Class	
		Class o	Class +
Actual Class	Class o	645	209
	Class +	298	948

		Predicted Class	
		Class o	Class +
Actual Class	Class o	181	673
	Class +	78	1168

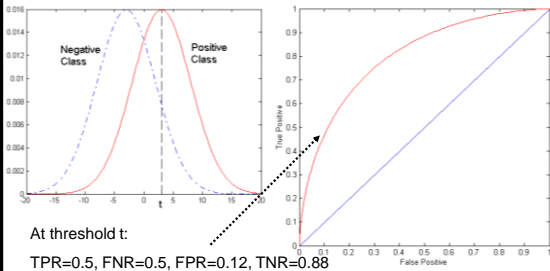


$\alpha = 0.3$		Predicted Class	
		Class o	Class +
Actual Class	Class o	645	209
	Class +	298	948

$\alpha = 0.7$		Predicted Class	
		Class o	Class +
Actual Class	Class o	181	673
	Class +	78	1168

## ROC Curve Example

- 1-dimensional data set containing 2 classes (positive and negative)
- Any points located at  $x > t$  is classified as positive



163

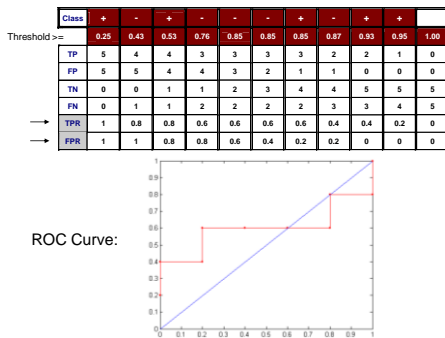
## How to Construct an ROC curve

Instance	Score	True Class
1	0.95	+
2	0.93	+
3	0.87	-
4	0.85	-
5	0.85	-
6	0.85	+
7	0.76	-
8	0.53	+
9	0.43	-
10	0.25	+

- Use a classifier that produces a continuous-valued score for each instance
  - The more likely it is for the instance to be in the + class, the higher the score
- Sort the instances in decreasing order according to the score
- Apply a threshold at each unique value of the score
- Count the number of TP, FP, TN, FN at each threshold
  - $TPR = TP/(TP+FN)$
  - $FPR = FP/(FP+TN)$

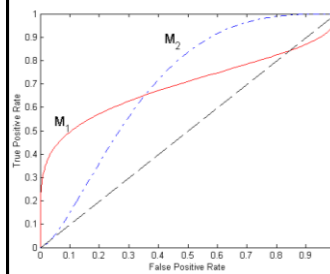
164

## How to construct an ROC curve



165

## Using ROC for Model Comparison



- No model consistently outperforms the other
  - $M_1$  is better for small FPR
  - $M_2$  is better for large FPR
- Area Under the ROC curve (AUC)
  - Ideal:
    - Area = 1
  - Random guess:
    - Area = 0.5

166

## Dealing with Imbalanced Classes - Summary

- Many measures exist, but none of them may be ideal in all situations
  - Random classifiers can have high value for many of these measures
  - TPR/FPR provides important information but may not be sufficient by itself in many practical scenarios
  - Given two classifiers, sometimes you can tell that one of them is strictly better than the other
    - $C_1$  is strictly better than  $C_2$  if  $C_1$  has strictly better TPR and FPR relative to  $C_2$  (or same TPR and better FPR, and vice versa)
  - Even if  $C_1$  is strictly better than  $C_2$ ,  $C_1$ 's F-value can be worse than  $C_2$ 's if they are evaluated on data sets with different imbalances
  - Classifier  $C_1$  can be better or worse than  $C_2$  depending on the scenario at hand (class imbalance, importance of TP vs FP, cost/time tradeoffs)

167

## Which Classifier is better?

T1	PREDICTED CLASS	
	Class=Yes	Class=No
	50	50

Precision (p) = 0.98  
 $TPR = Recall (r) = 0.5$   
 $FPR = 0.01$   
 $TPR/FPR = 50$   
 $F - \text{measure} = 0.66$

T2	PREDICTED CLASS	
	Class=Yes	Class=No
	99	1

Precision (p) = 0.9  
 $TPR = Recall (r) = 0.99$   
 $FPR = 0.1$   
 $TPR/FPR = 9.9$   
 $F - \text{measure} = 0.94$

T3	PREDICTED CLASS	
	Class=Yes	Class=No
	99	1

Precision (p) = 0.99  
 $TPR = Recall (r) = 0.99$   
 $FPR = 0.01$   
 $TPR/FPR = 99$   
 $F - \text{measure} = 0.99$

168

## Which Classifier is better? Medium Skew case

T1	PREDICTED CLASS		
		Class=Yes	Class=No
	Class=Yes	50	50
	Class=No	10	990

Precision (p) = 0.83  
 TPR = Recall (r) = 0.5  
 FPR = 0.01  
 TPR/FPR = 50  
 F-measure = 0.62

T2	PREDICTED CLASS		
		Class=Yes	Class=No
	Class=Yes	99	1
	Class=No	100	900

Precision (p) = 0.5  
 TPR = Recall (r) = 0.99  
 FPR = 0.1  
 TPR/FPR = 9.9  
 F-measure = 0.66

T3	PREDICTED CLASS		
		Class=Yes	Class=No
	Class=Yes	99	1
	Class=No	10	990

Precision (p) = 0.9  
 TPR = Recall (r) = 0.99  
 FPR = 0.01  
 TPR/FPR = 99  
 F-measure = 0.94

169

## Which Classifier is better? High Skew case

T1	PREDICTED CLASS		
		Class=Yes	Class=No
	Class=Yes	50	50
	Class=No	100	9900

Precision (p) = 0.3  
 TPR = Recall (r) = 0.5  
 FPR = 0.01  
 TPR/FPR = 50  
 F-measure = 0.375

T2	PREDICTED CLASS		
		Class=Yes	Class=No
	Class=Yes	99	1
	Class=No	1000	9000

Precision (p) = 0.09  
 TPR = Recall (r) = 0.99  
 FPR = 0.1  
 TPR/FPR = 9.9  
 F-measure = 0.165

T3	PREDICTED CLASS		
		Class=Yes	Class=No
	Class=Yes	99	1
	Class=No	100	9900

Precision (p) = 0.5  
 TPR = Recall (r) = 0.99  
 FPR = 0.01  
 TPR/FPR = 99  
 F-measure = 0.66

170

### Building Classifiers with Imbalanced Training Set

- Modify the distribution of training data so that rare class is well-represented in training set
  - Undersample the majority class
  - Oversample the rare class

171