

Data Mining Classification: Basic Concepts, Decision Trees, and Model Evaluation

Lecture Notes for Chapter 4

Data Mining
by
Zhaonian Zou

Outline

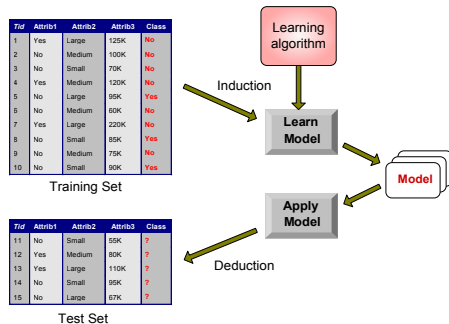
- Basic Concepts
- Decision Tree based Methods
- Model Evaluation
- Nearest Neighbor Methods
- Neural Networks
- Naïve Bayes and Bayesian Belief Networks
- Support Vector Machines
- Ensemble Methods

4.1. Basic Concepts



Classification: Definition

- Given a collection of records (*training set*)
 - Each record contains a set of *attributes*, one of the attributes is the *class*.
- Find a *model* for class attribute as a function of the values of other attributes.
- Goal: previously unseen records should be assigned a class as accurately as possible.
 - A *test set* is used to determine the accuracy of the model. Usually, the given data set is divided into training and test sets, with training set used to build the model and test set used to validate it.

Illustrating Classification Task



Examples of Classification Task

- Predicting tumor cells as benign or malignant
- Classifying credit card transactions as legitimate or fraudulent 
- Classifying secondary structures of protein as alpha-helix, beta-sheet, or random coil 
- Categorizing news stories as finance, weather, entertainment, sports, etc

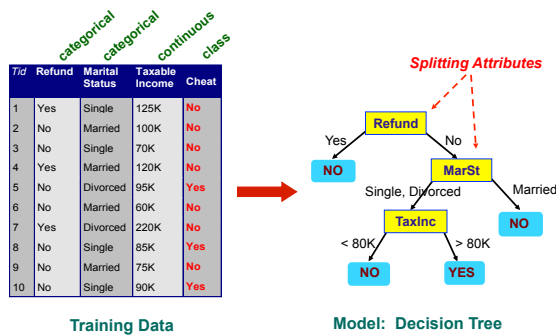
Classification Techniques

- Decision Tree based Methods
- Rule-based Methods
- Memory based Reasoning
- Neural Networks
- Naïve Bayes and Bayesian Belief Networks
- Support Vector Machines

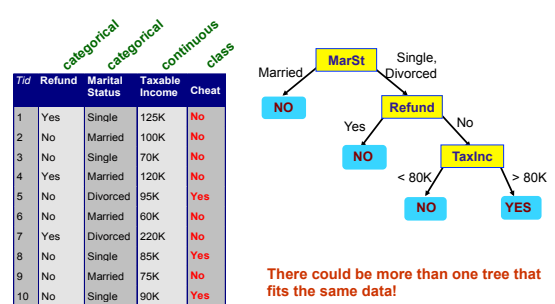
4.2. Decision Tree based Methods

4.2.1. Basic Concepts

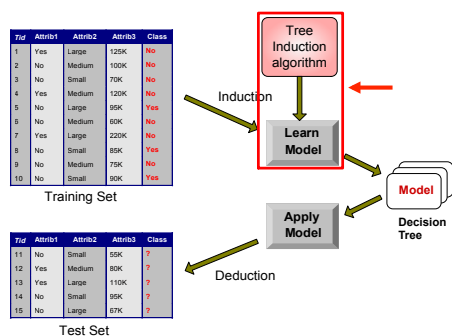
Example of a Decision Tree



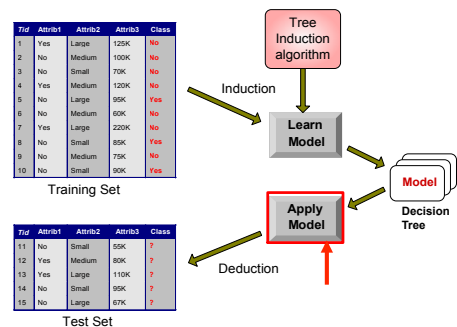
Another Example of Decision Tree



Decision Tree Classification Task



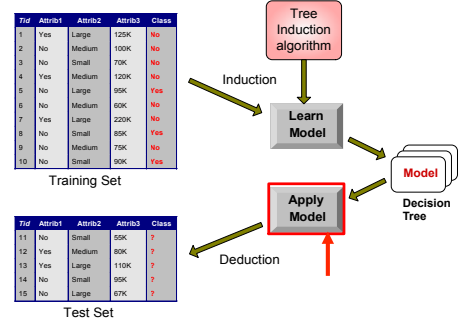
Decision Tree Classification Task



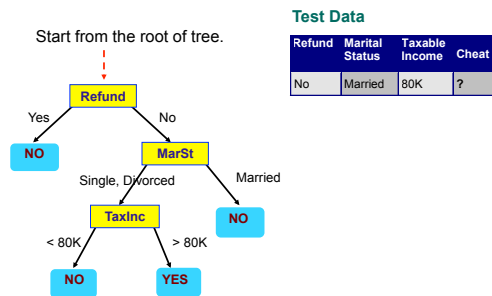
4.2. Decision Tree based Methods

4.2.2. Decision Tree Deduction

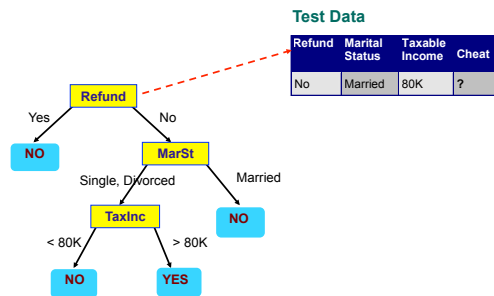
Decision Tree Classification Task



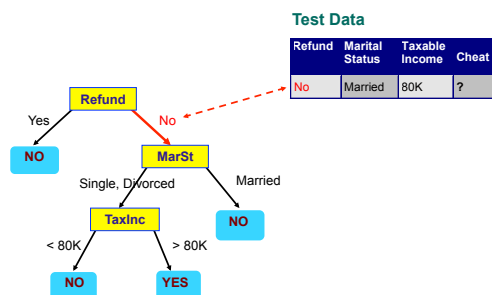
Apply Model to Test Data



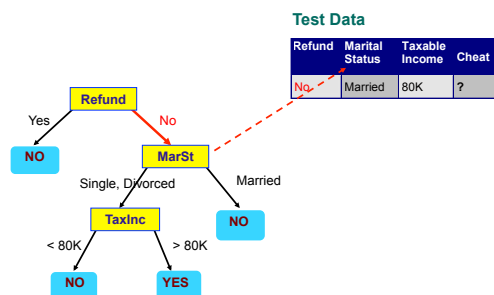
Apply Model to Test Data



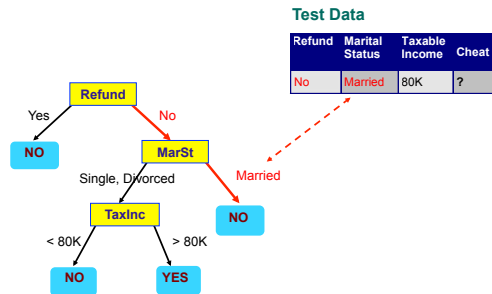
Apply Model to Test Data



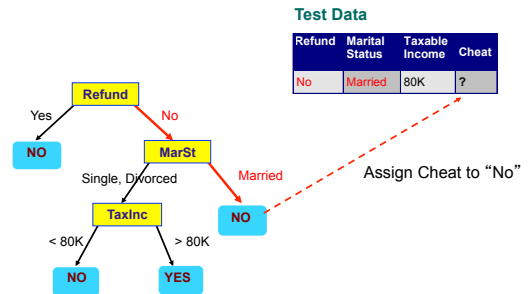
Apply Model to Test Data



Apply Model to Test Data



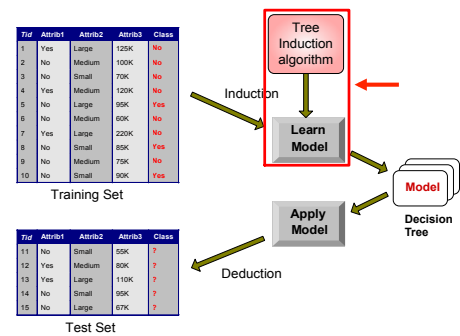
Apply Model to Test Data



4.2. Decision Tree based Methods

4.2.3. Decision Tree Induction

Decision Tree Classification Task

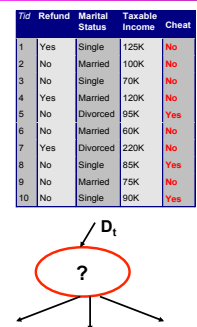


Decision Tree Induction

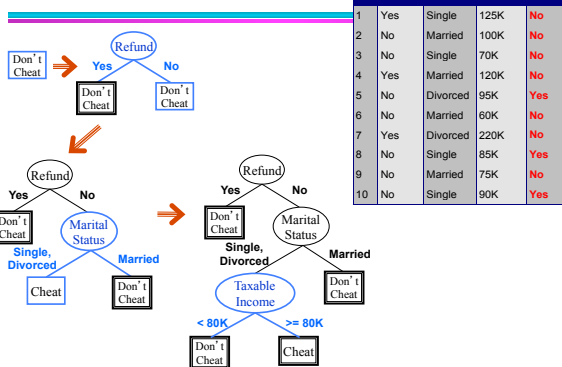
- Many Algorithms:
 - Hunt's Algorithm (one of the earliest)
 - CART
 - ID3, C4.5
 - SLIQ, SPRINT

General Structure of Hunt's Algorithm

- Let D_t be the set of training records that reach a node t
- General Procedure:
 - If D_t contains records that belong to the same class y_t , then t is a leaf node labeled as y_t
 - If D_t is an empty set, then t is a leaf node labeled by the default class, y_d
 - If D_t contains records that belong to more than one class, use an attribute test to split the data into smaller subsets. Recursively apply the procedure to each subset.



Hunt's Algorithm



Tree Induction

- Greedy strategy.
 - Split the records based on an attribute test that optimizes certain criterion.
- Issues
 - Determine how to split the records
 - ◆ How to specify the attribute test condition?
 - ◆ How to determine the best split?
 - Determine when to stop splitting

Tree Induction

- Greedy strategy.
 - Split the records based on an attribute test that optimizes certain criterion.
- Issues
 - Determine how to split the records
 - ◆ How to specify the attribute test condition?
 - ◆ How to determine the best split?
 - Determine when to stop splitting

4.2. Decision Tree based Methods

4.2.3. Decision Tree Induction

Issue 1: Specify Test Condition

How to Specify Test Condition?

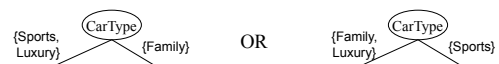
- Depends on attribute types
 - Nominal
 - Ordinal
 - Continuous
- Depends on number of ways to split
 - 2-way split
 - Multi-way split

Splitting Based on Nominal Attributes

- **Multi-way split:** Use as many partitions as distinct values.

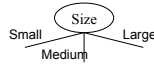


- **Binary split:** Divides values into two subsets. Need to find optimal partitioning.

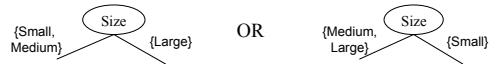


Splitting Based on Ordinal Attributes

- **Multi-way split:** Use as many partitions as distinct values.



- **Binary split:** Divides values into two subsets. Need to find optimal partitioning.



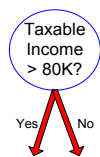
- What about this split?



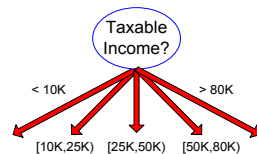
Splitting Based on Continuous Attributes

- Different ways of handling
 - **Discretization** to form an ordinal categorical attribute
 - ♦ Static – discretize once at the beginning
 - ♦ Dynamic – ranges can be found by equal interval bucketing, equal frequency bucketing (percentiles), or clustering.
 - **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

4.2. Decision Tree based Methods

4.2.3. Decision Tree Induction

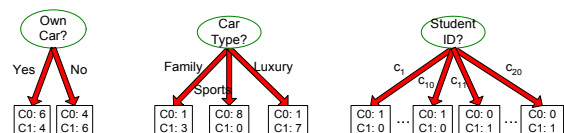
Issue 2: Determine the Best Split

Tree Induction

- Greedy strategy.
 - Split the records based on an attribute test that optimizes certain criterion.
- Issues
 - Determine how to split the records
 - ♦ How to specify the attribute test condition?
 - ♦ How to determine the best split?
 - Determine when to stop splitting

How to determine the Best Split

Before Splitting: 10 records of class 0,
10 records of class 1



Which test condition is the best?

How to determine the Best Split

- Greedy approach:
 - Nodes with **homogeneous** class distribution are preferred
- Need a measure of node impurity:

C0: 5
C1: 5

Non-homogeneous,
High degree of impurity

C0: 9
C1: 1

Homogeneous,
Low degree of impurity

Measures of Node Impurity

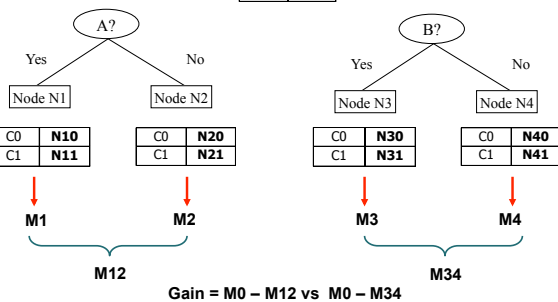
- Gini Index
- Entropy
- Misclassification error

How to Find the Best Split

Before Splitting:

C0	N00
C1	N01

→ M0



Measures of Node Impurity

- Gini Index
- Entropy
- Misclassification error

Measure of Impurity: GINI

- Gini Index for a given node t :

$$GINI(t) = 1 - \sum_j [p(j|t)]^2$$

(NOTE: $p(j|t)$ is the relative frequency of class j at node t).

- Maximum ($1 - 1/n_c$) when records are equally distributed among all classes, implying least interesting information
- Minimum (0.0) when all records belong to one class, implying most interesting information

C1	0
C2	6

Gini=0.000

C1	1
C2	5

Gini=0.278

C1	2
C2	4

Gini=0.444

C1	3
C2	3

Gini=0.500

Examples for computing GINI

$$GINI(t) = 1 - \sum_j [p(j|t)]^2$$

C1	0
C2	6

$P(C1) = 0/6 = 0$ $P(C2) = 6/6 = 1$
 $Gini = 1 - P(C1)^2 - P(C2)^2 = 1 - 0 - 1 = 0$

C1	1
C2	5

$P(C1) = 1/6$ $P(C2) = 5/6$
 $Gini = 1 - (1/6)^2 - (5/6)^2 = 0.278$

C1	2
C2	4

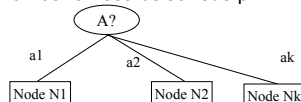
$P(C1) = 2/6$ $P(C2) = 4/6$
 $Gini = 1 - (2/6)^2 - (4/6)^2 = 0.444$

Splitting Based on GINI

- Used in CART, SLIQ, SPRINT.
- When a node p is split into k partitions (children), the quality of split is computed as,

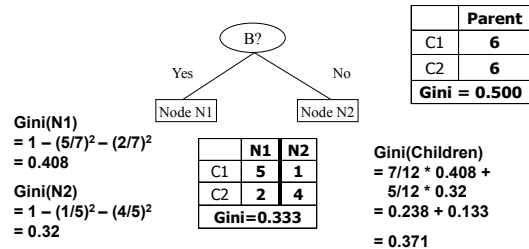
$$GINI_{split} = \sum_{i=1}^k \frac{n_i}{n} GINI(i)$$

where, n_i = number of records at child i ,
 n = number of records at node p .



Binary Attributes: Computing GINI Index

- Splits into two partitions
- Effect of Weighing partitions:
 - Larger and Purer Partitions are sought for.



Categorical Attributes: Computing Gini Index

- For each distinct value, gather counts for each class in the dataset
- Use the count matrix to make decisions

Multi-way split			
	CarType		
	Family	Sports	Luxury
C1	1	2	1
C2	4	1	1
Gini	0.393		

Two-way split (find best partition of values)			
	CarType {Sports, Luxury}		
	{Sports, Luxury}	{Family}	
C1	3	1	
C2	2	4	
Gini	0.400		

	CarType {Sports, Luxury}		
	{Sports}	{Family, Luxury}	
C1	2	2	
C2	1	5	
Gini	0.419		

Continuous Attributes: Computing Gini Index

- Use Binary Decisions based on one value
- Several Choices for the splitting value
 - Number of possible splitting values = Number of distinct values
- Each splitting value has a count matrix associated with it
 - Class counts in each of the partitions, $A < v$ and $A \geq v$
- Simple method to choose best v
 - For each v , scan the database to gather count matrix and compute its Gini index
 - Computationally Inefficient! Repetition of work.

Id	Refund	Marital Status	Taxable Income	Cheat
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

Diagram showing a decision node: Taxable Income > 80K? with 'Yes' and 'No' branches.

Continuous Attributes: Computing Gini Index...

- For efficient computation: for each attribute,
 - Sort the attribute on values
 - Linearly scan these values, each time updating the count matrix and computing gini index
 - Choose the split position that has the least gini index

		Cheat	No	No	No	Yes	Yes	Yes	No	No	No	No
		Taxable Income										
Sorted Values	→	60	70	75	85	90	95	100	120	125	172	230
Split Positions	→	55	65	72	80	87	92	97	110	122	172	230
		<=	<	<=	<	<=	<	<=	<	<=	<	<=
Yes		0	3	0	3	0	3	1	2	2	1	3
No		0	7	1	6	2	5	3	4	3	5	2
Gini		0.420	0.400	0.375	0.343	0.417	0.400	0.300	0.343	0.375	0.400	0.420

Measures of Node Impurity

- Gini Index
- Entropy
- Misclassification error

Measure of Impurity: Entropy

- Entropy at a given node t:

$$Entropy(t) = -\sum_j p(j|t) \log p(j|t)$$

(NOTE: $p(j|t)$ is the relative frequency of class j at node t).

- Measures homogeneity of a node.
 - Maximum ($\log n_c$) when records are equally distributed among all classes implying least information
 - Minimum (0.0) when all records belong to one class, implying most information
- Entropy based computations are similar to the GINI index computations

Examples for computing Entropy

$$Entropy(t) = -\sum_j p(j|t) \log_2 p(j|t)$$

C1	0
C2	6

$$P(C1) = 0/6 = 0 \quad P(C2) = 6/6 = 1$$

$$Entropy = -0 \log_2 0 - 1 \log_2 1 = -0 - 0 = 0$$

C1	1
C2	5

$$P(C1) = 1/6 \quad P(C2) = 5/6$$

$$Entropy = -(1/6) \log_2 (1/6) - (5/6) \log_2 (5/6) = 0.65$$

C1	2
C2	4

$$P(C1) = 2/6 \quad P(C2) = 4/6$$

$$Entropy = -(2/6) \log_2 (2/6) - (4/6) \log_2 (4/6) = 0.92$$

Splitting Based on Entropy

- Information Gain:

$$GAIN_{split} = Entropy(p) - \left(\sum_{i=1}^k \frac{n_i}{n} Entropy(i) \right)$$

Parent Node, p is split into k partitions;
 n_i is number of records in partition i

- Measures Reduction in Entropy achieved because of the split. Choose the split that achieves most reduction (maximizes GAIN)
- Used in ID3 and C4.5
- Disadvantage: Tends to prefer splits that result in large number of partitions, each being small but pure.

Splitting Based on Entropy

- Gain Ratio:

$$GainRatio_{split} = \frac{GAIN_{split}}{SplitINFO} \quad SplitINFO = -\sum_{i=1}^k \frac{n_i}{n} \log \frac{n_i}{n}$$

Parent Node, p is split into k partitions
 n_i is the number of records in partition i

- Adjusts Information Gain by the entropy of the partitioning (SplitINFO). Higher entropy partitioning (large number of small partitions) is penalized!
- Used in C4.5
- Designed to overcome the disadvantage of Information Gain

Measures of Node Impurity

- Gini Index
- Entropy
- Misclassification error

Measure of Impurity: Classification Error

- Classification error at a node t :

$$Error(t) = 1 - \max_i P(i|t)$$

- Measures misclassification error made by a node.
 - Maximum ($1 - 1/n_c$) when records are equally distributed among all classes, implying least interesting information
 - Minimum (0.0) when all records belong to one class, implying most interesting information

Examples for Computing Error

$$Error(t) = 1 - \max_i P(i | t)$$

C1	0
C2	6

$$P(C1) = 0/6 = 0 \quad P(C2) = 6/6 = 1$$

$$Error = 1 - \max(0, 1) = 1 - 1 = 0$$

C1	1
C2	5

$$P(C1) = 1/6 \quad P(C2) = 5/6$$

$$Error = 1 - \max(1/6, 5/6) = 1 - 5/6 = 1/6$$

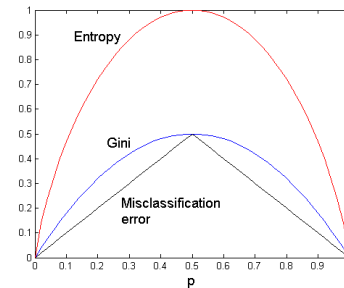
C1	2
C2	4

$$P(C1) = 2/6 \quad P(C2) = 4/6$$

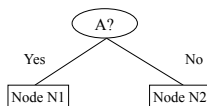
$$Error = 1 - \max(2/6, 4/6) = 1 - 4/6 = 1/3$$

Comparison among Splitting Criteria

For a 2-class problem:



Misclassification Error vs Gini



	Parent
C1	7
C2	3
Gini	0.42
Error	0.3

$$Gini(N1)$$

$$= 1 - (3/3)^2 - (0/3)^2$$

$$= 0$$

$$Gini(N2)$$

$$= 1 - (4/7)^2 - (3/7)^2$$

$$= 0.489$$

$$Error(N1) = 0$$

$$Error(N2) = 3/7$$

	N1	N2
C1	3	4
C2	0	3
Gini	0.361	
Error	0.3	

$$Gini(Children)$$

$$= 3/10 * 0$$

$$+ 7/10 * 0.489$$

$$= 0.342$$

$$Error(Children)$$

$$= 3/10 * 0 + 7/10 * 3/7 = 0.3$$

Gini improves !!

4.2. Decision Tree based Methods

4.2.3. Decision Tree Induction

Issue 3: Stopping Criteria

Tree Induction

- Greedy strategy.
 - Split the records based on an attribute test that optimizes certain criterion.
- Issues
 - Determine how to split the records
 - ◆ How to specify the attribute test condition?
 - ◆ How to determine the best split?
 - Determine when to stop splitting

Stopping Criteria for Tree Induction

- Stop expanding a node when all the records belong to the same class
- Stop expanding a node when all the records have the same attribute values
- Early termination (to be discussed later)

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

4.2. Decision Tree based Methods

4.2.4. Model Overfitting

Classification Errors

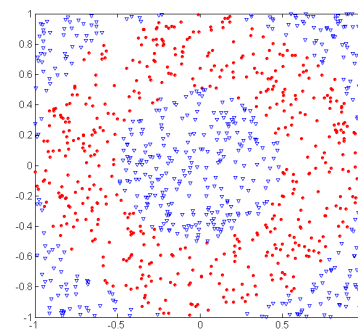
ACTUAL CLASS	PREDICTED CLASS	
	Class=Yes	Class=No
	Class=Yes	Class=No
Class=Yes	a (TP)	b (FN)
Class=No	c (FP)	d (TN)

Most widely-used metric:

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

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

Underfitting and Overfitting (Example)

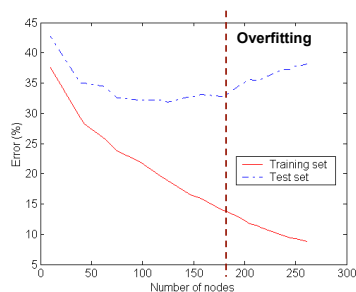


500 circular and 500 triangular data points.

Circular points:
 $0.5 \leq \sqrt{x_1^2 + x_2^2} \leq 1$

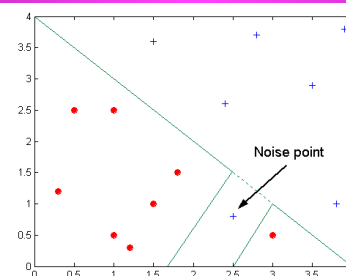
Triangular points:
 $\sqrt{x_1^2 + x_2^2} < 0.5$ or
 $\sqrt{x_1^2 + x_2^2} > 1$

Underfitting and Overfitting



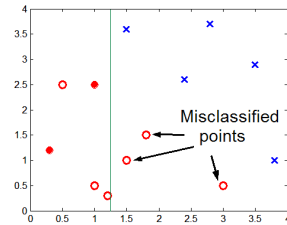
Underfitting: when model is too simple, both training and test errors are large

Overfitting due to Noise



Decision boundary is distorted by noise point

Overfitting due to Insufficient Examples



Lack of data points in the lower half of the diagram makes it difficult to predict correctly the class labels of that region

- Insufficient number of training records in the region causes the decision tree to predict the test examples using other training records that are irrelevant to the classification task

Notes on Overfitting

- Overfitting results in decision trees that are more complex than necessary
- Training error no longer provides a good estimate of how well the tree will perform on previously unseen records
- Need new ways for estimating errors

4.2. Decision Tree based Methods

4.2.4. Model Overfitting

Generalization Errors

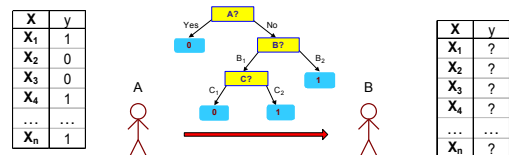
Estimating Generalization Errors

- **Training errors:** error on training ($\sum e(t)$)
- **Generalization errors:** error on testing ($\sum e'(t)$)
- Methods for estimating generalization errors:
 - **Optimistic approach:** $e'(t) = e(t)$
 - **Pessimistic approach:**
 - ♦ For each leaf node: $e'(t) = (e(t) + 0.5)$
 - ♦ Total errors: $e'(T) = e(T) + N \times 0.5$ (N: number of leaf nodes)
 - ♦ For a tree with 30 leaf nodes and 10 errors on training (out of 1000 instances):
 - Training error = $10/1000 = 1\%$
 - Generalization error = $(10 + 30 \times 0.5)/1000 = 2.5\%$
 - **Reduced error pruning (REP):**
 - ♦ uses validation data set to estimate generalization error

Occam's Razor

- Given two models of similar generalization errors, one should prefer the simpler model over the more complex model
- For complex models, there is a greater chance that it was fitted accidentally by errors in data
- Therefore, one should include model complexity when evaluating a model

Minimum Description Length (MDL)



- **Cost(Model, Data) = Cost(Data|Model) + Cost(Model)**
 - Cost is the number of bits needed for encoding.
 - Search for the least costly model.
- Cost(Data|Model) encodes the misclassification errors.
- Cost(Model) uses node encoding (number of children) plus splitting condition encoding.

4.2. Decision Tree based Methods

4.2.4. Model Overfitting

Decision Tree Pruning

How to Address Overfitting

● Pre-Pruning (Early Stopping Rule)

- Stop the algorithm before it becomes a fully-grown tree
- Typical 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:
 - ◆ 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 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
- Can use MDL for post-pruning

Example of Post-Pruning

Class = Yes	20
Class = No	10
Error	= 10/30

Training Error (Before splitting) = 10/30

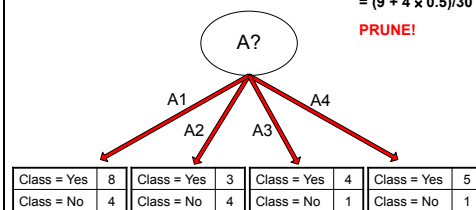
Pessimistic error = $(10 + 0.5)/30 = 10.5/30$

Training Error (After splitting) = 9/30

Pessimistic error (After splitting)

= $(9 + 4 \times 0.5)/30 = 11/30$

PRUNE!



4.2. Decision Tree based Methods

4.2.5. Other Issues

Other Issues

- Data Fragmentation
- Search Strategy
- Expressiveness
- Tree Replication

Data Fragmentation

- Number of instances gets smaller as you traverse down the tree
- Number of instances at the leaf nodes could be too small to make any statistically significant decision

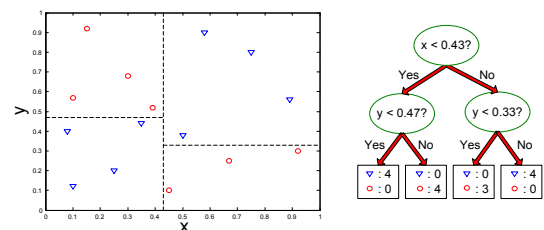
Search Strategy

- Finding an optimal decision tree is NP-hard
- The algorithm presented so far uses a greedy, top-down, recursive partitioning strategy to induce a reasonable solution
- Other strategies?
 - Bottom-up
 - Bi-directional

Expressiveness

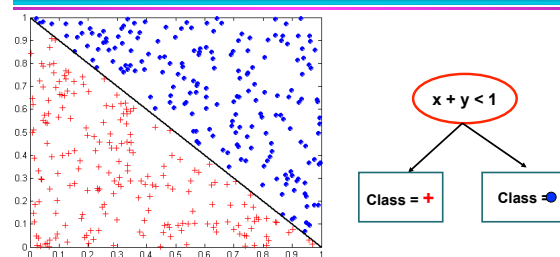
- Decision tree provides expressive representation for learning discrete-valued function
 - But they do not generalize well to certain types of Boolean functions
 - ♦ Example: parity function:
 - Class = 1 if there is an even number of Boolean attributes with truth value = True
 - Class = 0 if there is an odd number of Boolean attributes with truth value = True
 - ♦ For accurate modeling, must have a complete tree
- Not expressive enough for modeling continuous variables
 - Particularly when test condition involves only a single attribute at-a-time

Decision Boundary



- Border line between two neighboring regions of different classes is known as decision boundary
- Decision boundary is parallel to axes because test condition involves a single attribute at-a-time

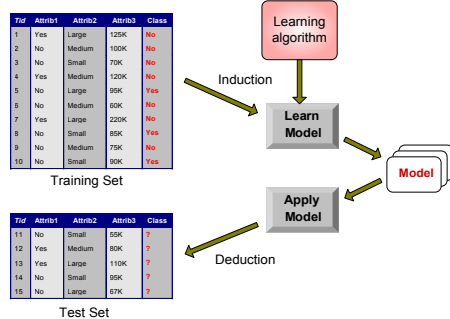
Oblique Decision Trees



- Test condition may involve multiple attributes
- More expressive representation
- Finding optimal test condition is computationally expensive

4.3. Model Evaluation

Illustrating Classification Task



Model Evaluation

- Metrics for Performance Evaluation
 - How to evaluate the performance of a model?
- Methods for Performance Evaluation
 - How to obtain reliable estimates?
- Methods for Model Comparison
 - How to compare the relative performance among competing models?

4.3. Model Evaluation

4.3.1. Performance Metrics

Model Evaluation

- Metrics for Performance Evaluation
 - How to evaluate the performance of a model?
- Methods for Performance Evaluation
 - How to obtain reliable estimates?
- Methods for Model Comparison
 - How to compare the relative performance among competing models?

Metrics for Performance Evaluation

- Focus on the predictive capability of a model
 - Rather than how fast it takes to classify or build models, scalability, etc.
- Confusion Matrix:

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

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

Metrics for Performance Evaluation...

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

- True Positive Rate (TPR) = $TP / (TP + FN)$
- False Negative Rate (FNR) = $FN / (TP + FN)$
- True Negative Rate (TNR) = $TN / (FP + TN)$
- False Positive Rate (FPR) = $FP / (FP + TN)$

Metrics for Performance Evaluation...

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

- Most widely-used metric:

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

Limitation of Accuracy

- Consider a 2-class problem
 - Number of Class 0 examples = 9990
 - Number of Class 1 examples = 10
- If model predicts everything to be class 0, accuracy is $9990/10000 = 99.9\%$
 - Accuracy is misleading because model does not detect any class 1 example

Cost Matrix

	PREDICTED CLASS		
ACTUAL CLASS	C(i j)	Class=Yes	Class=No
	Class=Yes	C(Yes Yes)	C(No Yes)
	Class=No	C(Yes No)	C(No No)

$C(i|j)$: Cost of misclassifying class j example as class i

Computing Cost of Classification

Cost Matrix	PREDICTED CLASS		
ACTUAL CLASS	C(i j)	+	-
	+	-1	100
	-	1	0

Model M ₁	PREDICTED CLASS		
ACTUAL CLASS		+	-
	+	150	40
	-	60	250

Accuracy = 80%
Cost = 3910

Model M_2	PREDICTED CLASS		
ACTUAL CLASS		+	-
	+	250	45
	-	5	200

Accuracy = 90%
Cost = 4255

Cost vs Accuracy

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

Accuracy is proportional to cost if
1. $C(\text{Yes}|\text{No}) = C(\text{No}|\text{Yes}) = q$
2. $C(\text{Yes}|\text{Yes}) = C(\text{No}|\text{No}) = p$

$$N = a + b + c + d$$

$$\text{Accuracy} = (a + d)/N$$

Cost	PREDICTED CLASS	
	Class=Yes	Class=No
	Class=Yes	Class=No
	p	q
	q	p

$$\begin{aligned} \text{Cost} &= p(a + d) + q(b + c) \\ &= p(a + d) + q(N - a - d) \\ &= qN - (q - p)(a + d) \\ &= N[q - (q - p) \times \text{Accuracy}] \end{aligned}$$

Cost-Sensitive Measures

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

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

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

- Precision is biased towards $C(\text{Yes}|\text{Yes})$ & $C(\text{Yes}|\text{No})$
- Recall is biased towards $C(\text{Yes}|\text{Yes})$ & $C(\text{No}|\text{Yes})$
- F-measure is biased towards all except $C(\text{No}|\text{No})$

$$\text{Weighted Accuracy} = \frac{w_1 a + w_2 d}{w_1 a + w_1 b + w_2 c + w_2 d}$$

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

4.3. Model Evaluation

4.3.2. Evaluation Methods

Model Evaluation

- Metrics for Performance Evaluation
 - How to evaluate the performance of a model?
- Methods for Performance Evaluation
 - How to obtain reliable estimates?
- Methods for Model Comparison
 - How to compare the relative performance among competing models?

Methods for Performance Evaluation

- How to obtain a reliable estimate of performance?
- Performance of a model may depend on other factors besides the learning algorithm:
 - Class distribution
 - Cost of misclassification
 - Size of training and test sets

Methods of Estimation

- Holdout
 - Reserve 2/3 for training and 1/3 for testing
- Random subsampling
 - Repeated holdout
- Cross validation
 - Partition data into k disjoint subsets
 - k-fold: train on k-1 partitions, test on the remaining one
 - Leave-one-out: $k=n$
- Stratified sampling
 - oversampling vs undersampling
- Bootstrap
 - Sampling with replacement

4.3. Model Evaluation

4.3.3. Model Comparison

Model Evaluation

- Metrics for Performance Evaluation
 - How to evaluate the performance of a model?
- Methods for Performance Evaluation
 - How to obtain reliable estimates?
- Methods for Model Comparison
 - How to compare the relative performance among competing models?

4.3. Model Evaluation

4.3.3. Model Comparison

ROC

ROC (Receiver Operating Characteristic)

- Developed in 1950s for signal detection theory to analyze noisy signals
 - Characterize the trade-off between positive hits and false alarms
- ROC curve plots TPR (on the y-axis) against FPR (on the x-axis)
- Performance of each classifier represented as a point on the ROC curve
 - changing the threshold of algorithm, sample distribution or cost matrix changes the location of the point

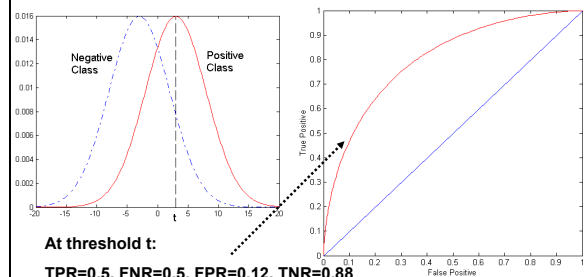
Metrics for Performance Evaluation...

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

- True Positive Rate (TPR) = $TP / (TP + FN)$
- False Negative Rate (FNR) = $FN / (TP + FN)$
- True Negative Rate (TNR) = $TN / (FP + TN)$
- False Positive Rate (FPR) = $FP / (FP + TN)$

ROC Curve

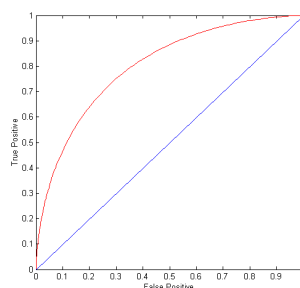
- 1-dimensional data set containing 2 classes (positive and negative)
- any points located at $x > t$ is classified as positive



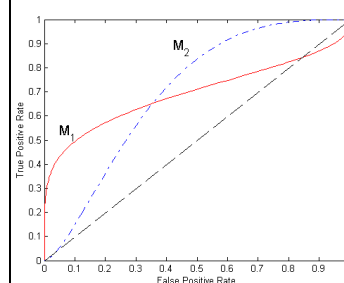
ROC Curve

(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



Using ROC for Model Comparison



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

How to Construct an ROC curve

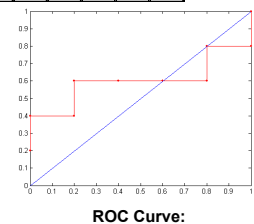
Instance	P(+ A)	True Class
1	0.95	+
2	0.93	+
3	0.87	-
4	0.85	-
5	0.85	-
6	0.85	+
7	0.76	-
8	0.53	+
9	0.43	-
10	0.25	+

- Use classifier that produces posterior probability for each test instance P(+|A)
- Sort the instances according to P(+|A) in decreasing order
- Apply threshold at each unique value of P(+|A)
- Count the number of TP, FP, TN, FN at each threshold
- TP rate, TPR = TP/(TP+FN)
- FP rate, FPR = FP/(FP + TN)

How to construct an ROC curve

Class	+	-	+	-	-	-	+	-	+	+	-
Threshold >=	0.25	0.43	0.53	0.76	0.85	0.85	0.85	0.87	0.93	0.95	1.00
TP	5	4	4	3	3	3	3	2	2	1	0
FP	5	5	4	4	3	2	1	1	0	0	0
TN	0	0	1	1	2	3	4	4	5	5	5
FN	0	1	1	2	2	2	2	3	3	4	5
TPR	1	0.8	0.8	0.6	0.6	0.6	0.6	0.4	0.4	0.2	0
FPR	1	1	0.8	0.8	0.6	0.4	0.2	0.2	0	0	0

Instance	P(+ A)	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	+



4.3. Model Evaluation

4.3.3. Model Comparison

Test of Significance

Test of Significance

- Given two models:
 - Model M1: accuracy = 85%, tested on 30 instances
 - Model M2: accuracy = 75%, tested on 5000 instances
- Can we say M1 is better than M2?
 - How much confidence can we place on accuracy of M1 and M2?
 - Can the difference in performance measure be explained as a result of random fluctuations in the test set?

Confidence Interval for Accuracy

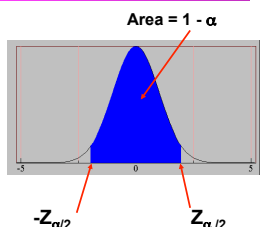
- Prediction can be regarded as a Bernoulli trial
 - A Bernoulli trial has 2 possible outcomes
 - Possible outcomes for prediction: correct or wrong
 - Collection of Bernoulli trials has a Binomial distribution:
 - ♦ $x \sim \text{Bin}(N, p)$ x : number of correct predictions
 - ♦ e.g: Toss a fair coin 50 times, how many heads would turn up?
Expected number of heads = $N \times p = 50 \times 0.5 = 25$
- Given x (# of correct predictions) or equivalently, $\text{acc} = x/N$, and N (# of test instances),

Can we predict p (true accuracy of model)?

Confidence Interval for Accuracy

- For large test sets ($N > 30$),
 - acc has a normal distribution with mean p and variance $p(1-p)/N$

$$P(-Z_{\alpha/2} < \frac{\text{acc} - p}{\sqrt{p(1-p)/N}} < Z_{\alpha/2}) = 1 - \alpha$$



- Confidence Interval for p :

$$p = \frac{2 \times N \times \text{acc} + Z_{\alpha/2}^2 \pm \sqrt{Z_{\alpha/2}^2 + 4 \times N \times \text{acc} - 4 \times N \times \text{acc}^2}}{2(N + Z_{\alpha/2}^2)}$$

Confidence Interval for Accuracy

- Consider a model that produces an accuracy of 80% when evaluated on 100 test instances:
 - N=100, acc = 0.8
 - Let $1-\alpha = 0.95$ (95% confidence)
 - From probability table, $Z_{\alpha/2}=1.96$

N	50	100	500	1000	5000
p(lower)	0.670	0.711	0.763	0.774	0.789
p(upper)	0.888	0.866	0.833	0.824	0.811

$1-\alpha$	Z
0.99	2.58
0.98	2.33
0.95	1.96
0.90	1.65

Comparing Performance of 2 Models

- Given two models, say M1 and M2, which is better?
 - M1 is tested on D1 (size=n1), found error rate = e_1
 - M2 is tested on D2 (size=n2), found error rate = e_2
 - Assume D1 and D2 are independent
 - If n1 and n2 are sufficiently large, then

$$e_1 \sim N(\mu_1, \sigma_1)$$

$$e_2 \sim N(\mu_2, \sigma_2)$$

- Approximate: $\hat{\sigma}_i = \frac{e_i(1-e_i)}{n_i}$

Comparing Performance of 2 Models

- To test if performance difference is statistically significant: $d = e_1 - e_2$
 - $d \sim N(d_t, \sigma_d)$ where d_t is the true difference
 - Since D1 and D2 are independent, their variance adds up:

$$\begin{aligned}\sigma_d^2 &= \sigma_1^2 + \sigma_2^2 \approx \hat{\sigma}_1^2 + \hat{\sigma}_2^2 \\ &= \frac{e_1(1-e_1)}{n_1} + \frac{e_2(1-e_2)}{n_2}\end{aligned}$$

- At $(1-\alpha)$ confidence level, $d_t = d \pm Z_{\alpha/2} \hat{\sigma}_d$

An Illustrative Example

- Given: M1: $n_1 = 30$, $e_1 = 0.15$
M2: $n_2 = 5000$, $e_2 = 0.25$
- $d = |e_2 - e_1| = 0.1$ (2-sided test)

$$\hat{\sigma}_d = \frac{0.15(1-0.15)}{30} + \frac{0.25(1-0.25)}{5000} = 0.0043$$

- At 95% confidence level, $Z_{\alpha/2}=1.96$

$$d_t = 0.100 \pm 1.96 \times \sqrt{0.0043} = 0.100 \pm 0.128$$

=> Interval contains 0 => difference may not be statistically significant

Comparing Performance of 2 Algorithms

- Each learning algorithm may produce k models:
 - L1 may produce M11, M12, ..., M1k
 - L2 may produce M21, M22, ..., M2k
- If models are generated on the same test sets D1, D2, ..., Dk (e.g., via cross-validation)

- For each set: compute $d_j = e_{1j} - e_{2j}$
- d_j has mean d_t and variance σ_t
- Estimate:

$$\hat{\sigma}_t^2 = \frac{\sum_{j=1}^k (d_j - \bar{d})^2}{k(k-1)}$$

$$d_t = d \pm t_{1-\alpha, k-1} \hat{\sigma}_t$$

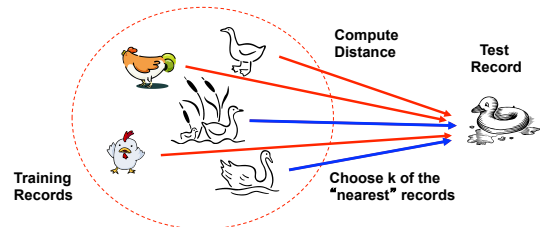
4.4. Nearest-Neighbor Classifiers

Instance Based Classifiers

- Examples:
 - Rote-learner
 - ◆ Memorizes entire training data and performs classification only if attributes of record match one of the training examples exactly
 - Nearest neighbor
 - ◆ Uses k “closest” points (nearest neighbors) for performing classification

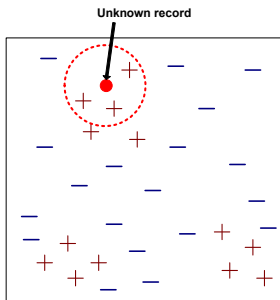
Nearest Neighbor Classifiers

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

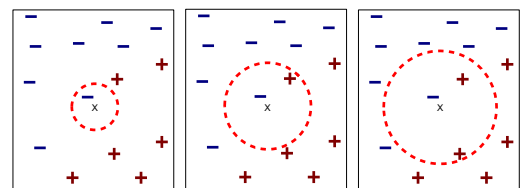


Nearest-Neighbor Classifiers

- Requires three things
 - The set of stored records
 - Distance Metric to compute distance between records
 - The value of k , the number of nearest neighbors to retrieve
- To classify an unknown record:
 - Compute distance to other training records
 - Identify k nearest neighbors
 - Use class labels of nearest neighbors to determine the class label of unknown record (e.g., by taking majority vote)



Definition of Nearest Neighbor



(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

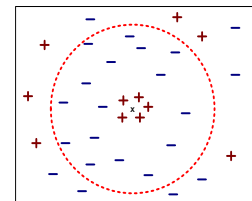
K-nearest neighbors of a record x are data points that have the k smallest distance to x

Nearest Neighbor Classification

- Compute distance between two points:
 - Euclidean distance
$$d(p, q) = \sqrt{\sum_i (p_i - q_i)^2}$$
- Determine the class from nearest neighbor list
 - take the majority vote of class labels among the k -nearest neighbors
 - Weigh the vote according to distance
 - ◆ weight factor, $w = 1/d^2$

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



Nearest Neighbor Classification...

- Scaling issues
 - 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

Nearest Neighbor Classification...

- Problem with Euclidean measure:
 - High dimensional data
 - ◆ curse of dimensionality
 - Can produce counter-intuitive results

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

d = 1.4142 d = 1.4142

- ◆ Solution: Normalize the vectors to unit length

Dealing with Curse of Dimensionality

- Multidimensional Index Structures
 - kd-trees, R-trees, quad trees
- Unfortunately, for high-dimensional data, there is little that can be done to avoid searching a large portion of the data.
- Two ways to deal with the curse of dimensionality
 - VA files
 - Dimensionality reduction

Nearest neighbor Classification...

- k-NN classifiers are lazy learners
 - It does not build models explicitly
 - Unlike eager learners such as decision tree induction and rule-based systems
 - Classifying unknown records are relatively expensive

4.5 Bayes Classifiers

4.5.1. Foundations

Bayes Classifier

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

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

Example of Bayes Theorem

- 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?

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

Bayesian Classifiers

- Consider each attribute and class label as random variables
- Given a record with attributes (A_1, A_2, \dots, A_n)
 - Goal is to predict class C
 - Specifically, we want to find the value of C that maximizes $P(C | A_1, A_2, \dots, A_n)$
- Can we estimate $P(C | A_1, A_2, \dots, A_n)$ directly from data?

Bayesian Classifiers

- Approach:
 - compute the posterior probability $P(C | A_1, A_2, \dots, A_n)$ for all values of C using the Bayes theorem
- $$P(C | A_1, A_2, \dots, A_n) = \frac{P(A_1, A_2, \dots, A_n | C)P(C)}{P(A_1, A_2, \dots, A_n)}$$
- Choose value of C that maximizes $P(C | A_1, A_2, \dots, A_n)$
 - Equivalent to choosing value of C that maximizes $P(A_1, A_2, \dots, A_n | C) P(C)$
- How to estimate $P(A_1, A_2, \dots, A_n | C)$?

4.5. Bayes Classifiers

4.5.2. Naïve Bayes Classifiers

Naïve Bayes Classifier

- Assume independence among attributes A_i when class is given:
 - $P(A_1, A_2, \dots, A_n | C_j) = P(A_1 | C_j) P(A_2 | C_j) \dots P(A_n | C_j)$
 - Can estimate $P(A_i | C_j)$ for all A_i and C_j .
 - New point is classified to C_j if $P(C_j) \prod P(A_i | C_j)$ is maximal.

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

- Class: $P(C) = N_c / N$
 - e.g., $P(\text{No}) = 7/10$, $P(\text{Yes}) = 3/10$

- For discrete attributes:

$$P(A_i | C_k) = |A_{ik}| / N_{C_k}$$

- 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
 - violates independence assumption
 - Two-way split:** ($A < v$) or ($A > v$)
 - choose only one of the two splits as new attribute
 - 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 $P(A_i|c)$

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

- Normal distribution:

$$P(A_i | c_j) = \frac{1}{\sqrt{2\pi\sigma_j^2}} e^{-\frac{(A_i - \mu_j)^2}{2\sigma_j^2}}$$

- One for each (A_i, c_i) pair

- For (Income, Class=No):

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

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

Example of Naïve Bayes Classifier

Given a Test Record:

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

naive Bayes Classifier:

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

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

- $P(X|\text{Class}=\text{No}) = P(\text{Refund}=\text{No}|\text{Class}=\text{No}) \times P(\text{Married}|\text{Class}=\text{No}) \times P(\text{Income}=120K|\text{Class}=\text{No}) = 4/7 \times 4/7 \times 0.0072 = 0.0024$
- $P(X|\text{Class}=\text{Yes}) = P(\text{Refund}=\text{No}|\text{Class}=\text{Yes}) \times P(\text{Married}|\text{Class}=\text{Yes}) \times P(\text{Income}=120K|\text{Class}=\text{Yes}) = 1 \times 0 \times 1.2 \times 10^{-9} = 0$

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

Naïve Bayes Classifier

- If one of the conditional probability is zero, then the entire expression becomes zero
- Probability estimation:

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

c: number of classes

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

p: prior probability

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

m: parameter

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
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

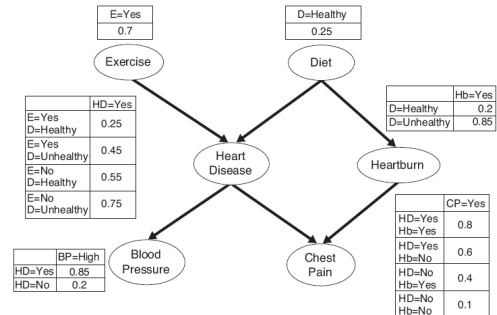
Naïve Bayes (Summary)

- Robust to isolated noise points
- Handle missing values by ignoring the instance during probability estimate calculations
- Robust to irrelevant attributes
- Independence assumption may not hold for some attributes
 - Use other techniques such as Bayesian Belief Networks (BBN)

4.5. Bayes Classifiers

4.5.2. Bayes Networks

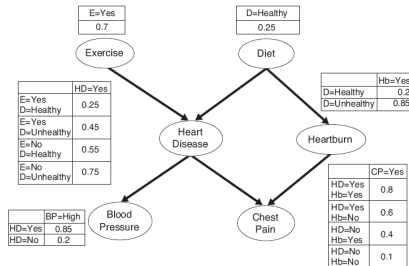
Bayes Networks



Bayes Networks

Conditional Independence

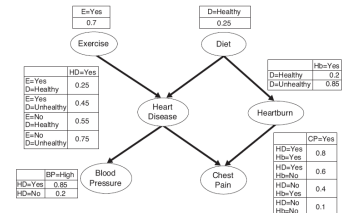
- A node is conditional independent of its non-descendants, if its parents are known.



Bayes Network Inference

Inferring $P(HD = Yes)$

$$\begin{aligned}
 P(HD=Yes) &= P(HD|E,D)P(E)P(D) \\
 &+ P(HD|E,\sim D)P(E)P(\sim D) \\
 &+ P(HD|\sim E,D)P(\sim E)P(D) \\
 &+ P(HD|\sim E,\sim D)P(\sim E)P(\sim D) \\
 &= 0.49
 \end{aligned}$$

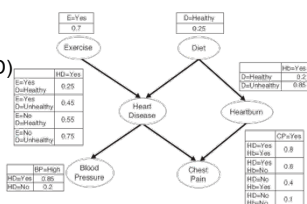


Bayes Network Inference

Inferring $P(HD = Yes | BP = High)$

$$\begin{aligned}
 P(HD=Yes|BP=High) &= \frac{P(BP=High|HD=Yes)P(HD=Yes)}{P(BP=High)} \\
 &= \frac{0.85 \cdot 0.49}{0.85 \cdot 0.49 + 0.2 \cdot 0.51} \\
 &= 0.5185
 \end{aligned}$$

$$\begin{aligned}
 P(BP=High) &= P(BP=High|HD)P(HD) \\
 &+ P(BP=High|\sim HD)P(\sim HD) \\
 &= 0.85 \cdot 0.49 + 0.2 \cdot 0.51 \\
 &= 0.5185
 \end{aligned}$$

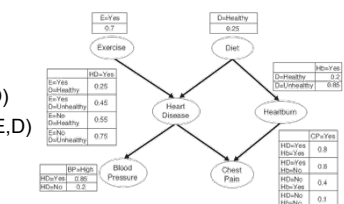


Bayes Network Inference

Inferring $P(HD=Yes|BP=High, E=Yes, D=Healthy)$

$$\begin{aligned}
 P(HD=Yes|BP=High, E=Yes, D=Healthy) &= \frac{P(BP=High|HD, E, D)P(HD|E, D)P(E)P(D)}{P(BP|E, D)P(E)P(D)} \\
 &= \frac{P(BP|HD)P(HD|E, D)}{P(BP|E, D)}
 \end{aligned}$$

$$\begin{aligned}
 P(BP|E, D) &= P(BP|HD)P(HD|E, D) \\
 &+ P(BP|\sim HD)P(\sim HD|E, D)
 \end{aligned}$$

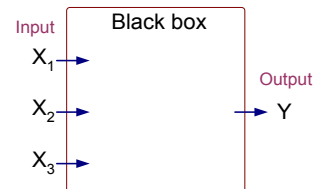


4.6. Artificial Neural Networks

4.6.1. Perceptrons

Perceptrons

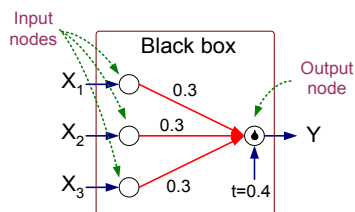
X_1	X_2	X_3	Y
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1
0	0	1	0
0	1	0	0
0	1	1	1
0	0	0	0



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

Perceptrons

X_1	X_2	X_3	Y
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1
0	0	1	0
0	1	0	0
0	1	1	1
0	0	0	0



$$Y = I(0.3X_1 + 0.3X_2 + 0.3X_3 - 0.4 > 0)$$

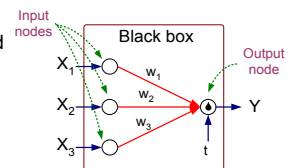
$$\text{where } I(z) = \begin{cases} 1 & \text{if } z \text{ is true} \\ 0 & \text{otherwise} \end{cases}$$

Perceptron Model

- Model is an assembly of inter-connected nodes and weighted links

- Output node sums up each of its input value according to the weights of its links

- Compare output node against some threshold t

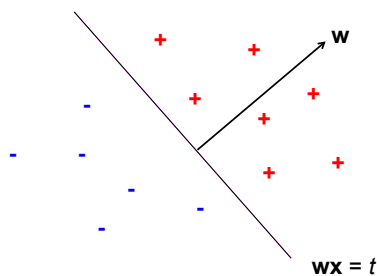


Perceptron Model

$$Y = I\left(\sum_i w_i X_i - t\right) \quad \text{or} \\ Y = \text{sign}\left(\sum_i w_i X_i - t\right)$$

Perceptrons: Another Perspective

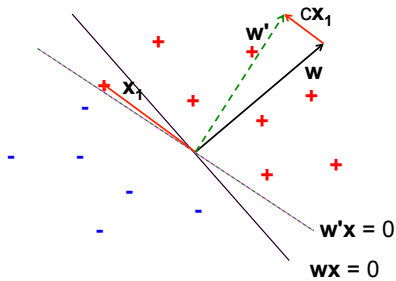
- A perceptron is a linear binary classifier.



Training a Perceptron

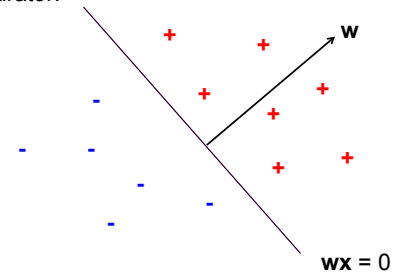
- Assume $t = 0$
- Initialize $\mathbf{w} = \mathbf{0}$
- Pick a learning-rate parameter c
- Repeat until convergence
 - Get next training example (\mathbf{x}, y)
 - Let $y' = \mathbf{w}\mathbf{x}$
 - If $\text{sign}(y') = \text{sign}(y)$, then do nothing; else replace \mathbf{w} by $\mathbf{w} + c\mathbf{y}\mathbf{x}$

Training a Perceptron: Illustration



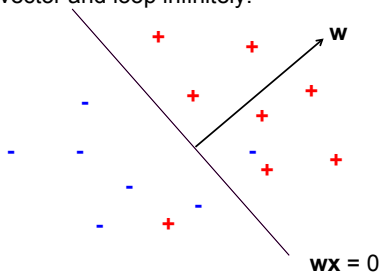
Convergence of Perceptrons

- If the data is linearly separable, then converge to a separator.



Convergence of Perceptrons

- If the data is not linearly separable, then repeat a weight vector and loop infinitely.



Training a Perceptron: An Example

x	x1	x2	x3	x4	x5	y
r1	1	1	0	1	1	+1
r2	0	0	1	1	0	-1
r3	0	1	1	0	0	+1
r4	1	0	0	1	0	-1
r5	1	0	1	0	1	+1
r6	1	0	1	1	0	-1

$c = 0.5$

x	wx	w1	w2	w3	w4	w5
		0	0	0	0	0
r1	0	1/2	1/2	0	1/2	1/2
r2	1/2	1/2	1/2	-1/2	0	1/2
r3	0	1/2	1	0	0	1/2
r4	1/2	0	1	0	-1/2	1/2
r5	1/2	0	1	0	-1/2	1/2
r6	-1/2	0	1	0	-1/2	1/2

Training a Perceptron

- Allow t to vary
- Let $\mathbf{w}' = [\mathbf{w}, t]$
- Let $\mathbf{x}' = [\mathbf{x}, -1]$
- $\mathbf{w}'\mathbf{x}' = t$ is equivalent to $\mathbf{w}\mathbf{x} - t = 0$

Training a Perceptron: An Example

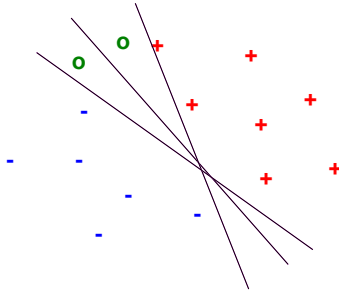
x	x1	x2	x3	x4	x5	t	y
r1	1	1	0	1	1	-1	+1
r2	0	0	1	1	0	-1	-1
r3	0	1	1	0	0	-1	+1
r4	1	0	0	1	0	-1	-1
r5	1	0	1	0	1	-1	+1
r6	1	0	1	1	0	-1	-1

$c = 0.5$

x	wx	w1	w2	w3	w4	w5	t
		0	0	0	0	0	0
r1	0	1/2	1/2	0	1/2	1/2	-1/2
r2	1	1/2	1/2	-1/2	0	1/2	0
r3	0	1/2	1	0	0	1/2	1/2
r4	-1/2	1/2	1	0	0	1/2	1/2
r5	1/2	1/2	1	0	0	1/2	1/2
r6	0	0	1/2	-1/2	-1/2	0	0

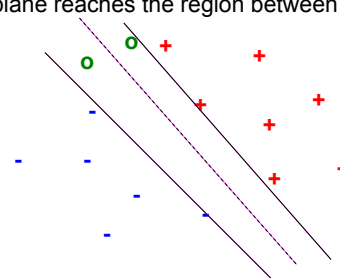
Problems with Perceptrons

- Not all hyperplanes are equally good.



Problems with Perceptrons

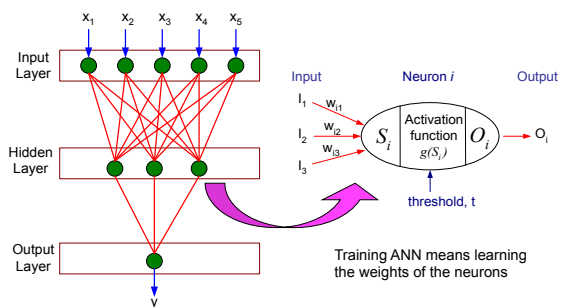
- Perceptrons converge as soon as the separating hyperplane reaches the region between classes.



4.6. Artificial Neural Networks

4.6.2. ANN

General Structure of ANN

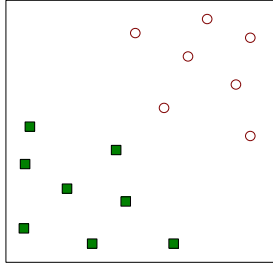


Algorithm for learning ANN

- Initialize the weights (w_0, w_1, \dots, w_k)
- Adjust the weights in such a way that the output of ANN is consistent with class labels of training examples
 - Objective function: $E = \sum_i [Y_i - f(w_i, X_i)]^2$
 - Find the weights w_i 's that minimize the above objective function
 - e.g., backpropagation algorithm (see lecture notes)
 - <http://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/>

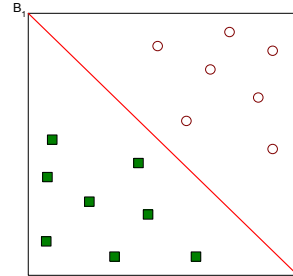
4.7. Support Vector Machines

Support Vector Machines



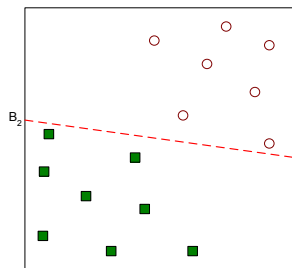
- Find a linear hyperplane (decision boundary) that will separate the data

Support Vector Machines



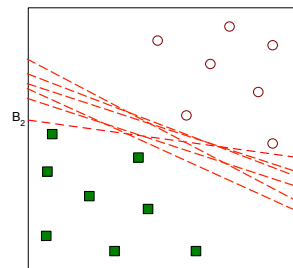
- One Possible Solution

Support Vector Machines



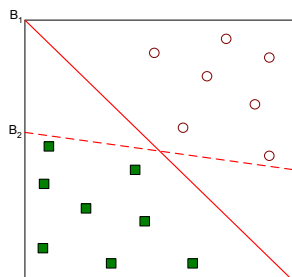
- Another possible solution

Support Vector Machines



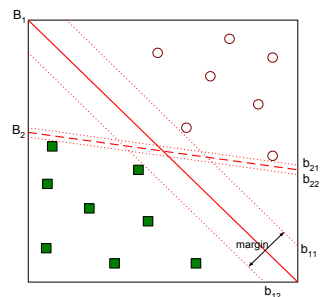
- Other possible solutions

Support Vector Machines



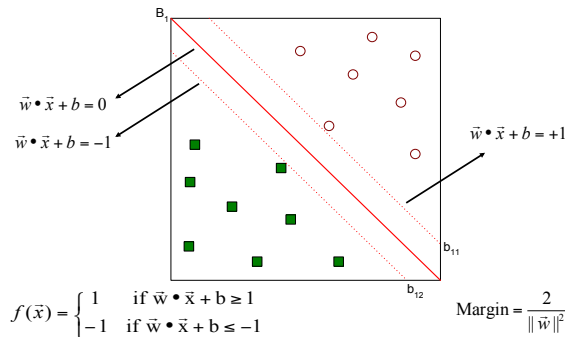
- Which one is better? B1 or B2?
- How do you define better?

Support Vector Machines



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

Support Vector Machines



Support Vector Machines

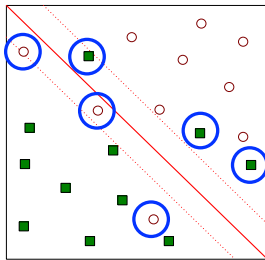
- We want to maximize: $\text{Margin} = \frac{2}{\|\tilde{w}\|^2}$
 - Which is equivalent to minimizing: $L(w) = \frac{\|\tilde{w}\|^2}{2}$
 - But subjected to the following constraints:

$$\tilde{w} \cdot \tilde{x}_i + b \geq 1 \quad \text{if } y_i = 1$$

$$\tilde{w} \cdot \tilde{x}_i + b \leq -1 \quad \text{if } y_i = -1$$
- ♦ This is a constrained optimization problem
 - Numerical approaches to solve it (e.g., quadratic programming)

Support Vector Machines

- What if the problem is not linearly separable?



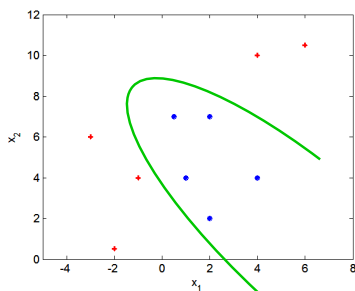
Support Vector Machines

- What if the problem is not linearly separable?
 - Introduce slack variables
 - ♦ Need to minimize: $L(w) = \frac{\|\tilde{w}\|^2}{2} + C \left(\sum_{i=1}^N \xi_i^k \right)$
 - ♦ Subject to:

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

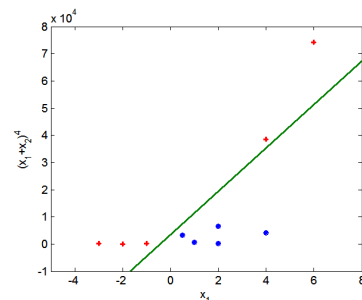
Nonlinear Support Vector Machines

- What if decision boundary is not linear?



Nonlinear Support Vector Machines

- Transform data into higher dimensional space

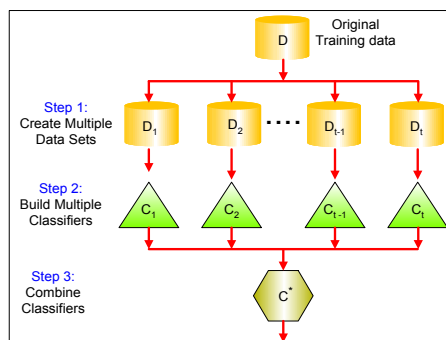


4.8 Ensemble Methods

Ensemble Methods

- Construct a set of classifiers from the training data
- Predict class label of previously unseen records by aggregating predictions made by multiple classifiers

General Idea



Why does it work?

- Suppose there are 25 base classifiers
 - Each classifier has error rate, $\epsilon = 0.35$
 - Assume classifiers are independent
 - Probability that the ensemble classifier makes a wrong prediction:

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

Examples of Ensemble Methods

- How to generate an ensemble of classifiers?
 - Bagging
 - Boosting

Bagging

- 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
- Each sample has probability $(1 - 1/n)^n$ of being selected

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
 - Unlike bagging, weights may change at the end of boosting round

Boosting

- Records that are wrongly classified will have their weights increased
- Records that are classified correctly will have their weights decreased

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

Example: AdaBoost

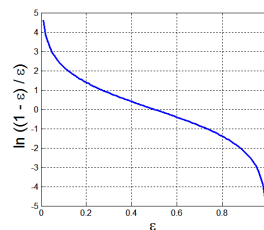
- Base classifiers: C_1, C_2, \dots, C_T

- Error rate:

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

- Importance of a classifier:

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



Example: AdaBoost

- Weight update:

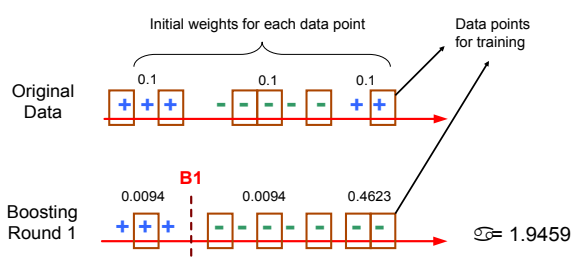
$$w_i^{(j+1)} = \frac{w_i^{(j)}}{Z_j} \begin{cases} \exp^{-\alpha_j} & \text{if } C_j(x_i) = y_i \\ \exp^{\alpha_j} & \text{if } C_j(x_i) \neq y_i \end{cases}$$

where Z_j is the normalization factor

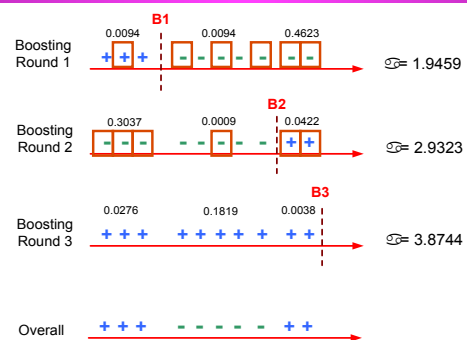
- If any intermediate rounds produce error rate higher than 50%, the weights are reverted back to $1/n$ and the resampling procedure is repeated
- Classification:

$$C^*(x) = \arg \max_y \sum_{j=1}^T \alpha_j \delta(C_j(x) = y)$$

Illustrating AdaBoost



Illustrating AdaBoost



THANKS

Finally, I will learn a classifier to
classify each of you into one of the
classes A+, A, B+, B, C+, C, and F.