Data Mining Techniques: Classification and Prediction

Mirek Riedewald

Some slides based on presentations by Han/Kamber, Tan/Steinbach/Kumar, and Andrew Moore

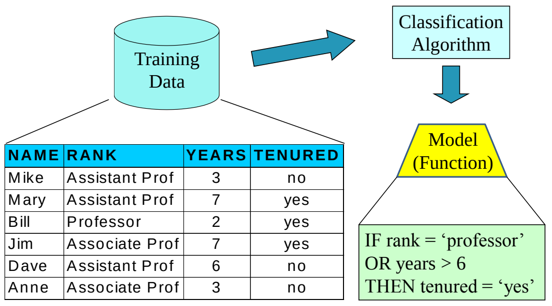
Classification and Prediction Overview

- Introduction
- Decision Trees
- Statistical Decision Theory
- Nearest Neighbor
- Bayesian Classification
- Artificial Neural Networks
- Support Vector Machines (SVMs)
- Prediction
- Accuracy and Error Measures
- Ensemble Methods

Classification vs. Prediction

- Assumption: after data preparation, have single data set where each record has attributes $X_1,\dots,X_n$, and Y.
- Goal: learn a function $f:(X_1,\dots,X_n)\rightarrow Y$, then use this function to predict y for a given input record $(x_1,\dots,x_n)$.
  - Classification: Y is a discrete attribute, called the class label
    - Usually a categorical attribute with small domain
  - Prediction: Y is a continuous attribute
- Called supervised learning, because true labels (Y-values) are known for the initially provided data
- Typical applications: credit approval, target marketing, medical diagnosis, fraud detection

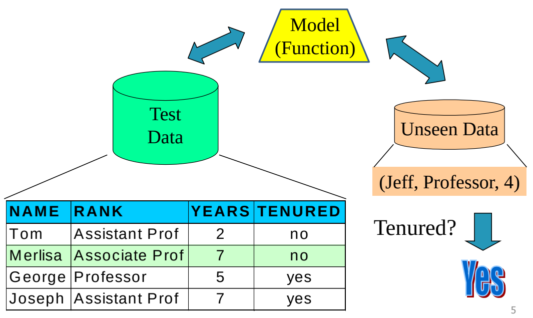
Induction: Model Construction



NAME	RANK	YEARS	TENURED
Mike	Assistant Prof	3	no
Mary	Assistant Prof	7	yes
Bill	Professor	2	yes
Jim	Associate Prof	7	yes
Dave	Assistant Prof	6	no
Anne	Associate Prof	3	no

IF rank = 'professor'
OR years > 6
THEN tenured = 'yes'

Deduction: Using the Model



NAME	RANK	YEARS	TENURED
Tom	Assistant Prof	2	no
Merlisa	Associate Prof	7	no
George	Professor	5	yes
Joseph	Assistant Prof	7	yes

Classification and Prediction Overview

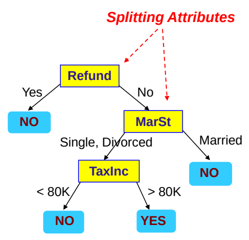
- Introduction
- Decision Trees
- Statistical Decision Theory
- Bayesian Classification
- Artificial Neural Networks
- Support Vector Machines (SVMs)
- Nearest Neighbor
- Prediction
- Accuracy and Error Measures
- Ensemble Methods

Example of a Decision Tree

categorical categorical continuous class

Tid	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

Training Data

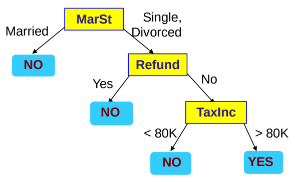


Model: Decision Tree

Another Example of Decision Tree

categorical categorical continuous class

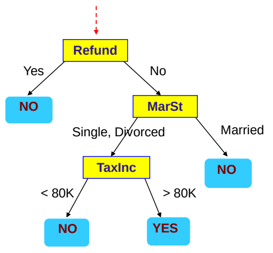
Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



There could be more than one tree that fits the same data!

Apply Model to Test Data

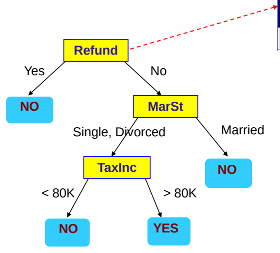
Start from the root of tree.



Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

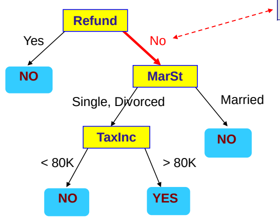
Apply Model to Test Data



Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

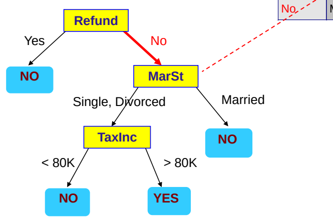
Apply Model to Test Data



Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Apply Model to Test Data



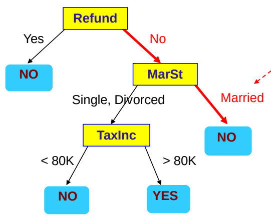
Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Apply Model to Test Data

Test Data

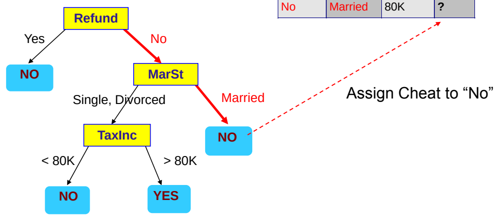
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Apply Model to Test Data

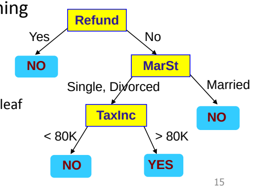
Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

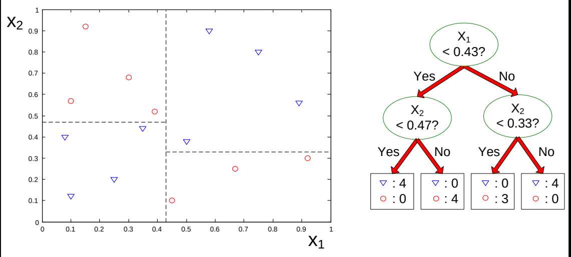


Decision Tree Induction

- Basic greedy algorithm
  - Top-down, recursive divide-and-conquer
  - At start, all the training records are at the root
  - Training records partitioned recursively based on split attributes
  - Split attributes selected based on a heuristic or statistical measure (e.g., information gain)
- Conditions for stopping partitioning
  - Pure node (all records belong to same class)
  - No remaining attributes for further partitioning
    - Majority voting for classifying the leaf
  - No cases left



Decision Boundary



Decision boundary = border between two neighboring regions of different classes.

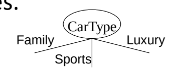
For trees that split on a single attribute at a time, the decision boundary is parallel to the axes.

How to Specify Split Condition?

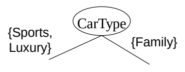
- Depends on attribute types
  - Nominal
  - Ordinal
  - Numeric (continuous)
- Depends on number of ways to split
  - 2-way split
  - Multi-way split

Splitting Nominal Attributes

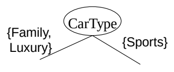
- Multi-way split: use as many partitions as distinct values.



- Binary split: divides values into two subsets; need to find optimal partitioning.

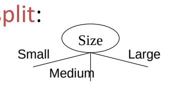


OR

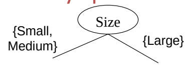


Splitting Ordinal Attributes

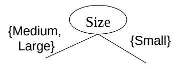
- Multi-way split:



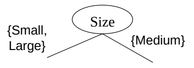
- Binary split:



OR



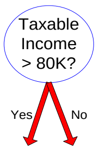
- What about this split?



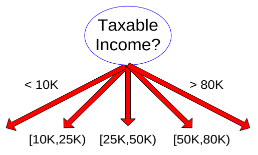
Splitting Continuous Attributes

- Different options
  - Discretization to form an ordinal categorical attribute
    - Static – discretize once at the beginning
    - Dynamic – ranges found by equal interval bucketing, equal frequency bucketing (percentiles), or clustering.
  - Binary Decision: $(A < v)$ or $(A \geq v)$
    - Consider all possible splits, choose best one

Splitting Continuous Attributes



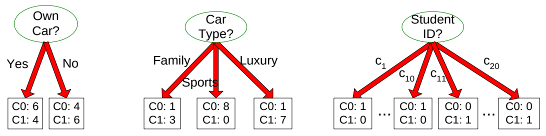
(i) Binary split



(ii) Multi-way split

How to Determine Best Split

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



Which test condition is the best?

How to Determine Best Split

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

C0: 5 C1: 5	C0: 9 C1: 1
Non-homogeneous, High degree of impurity	**Homogeneous, Low degree of impurity**

Attribute Selection Measure: Information Gain

- Select attribute with highest information gain
- p_i = probability that an arbitrary record in D belongs to class C_i, i=1,…,m
- Expected information (entropy) needed to classify a record in D:

$$\text{Info}(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

- Information needed after using attribute A to split D into v partitions $D_1,\ldots, D_v$:

$$\text{Info}_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \text{Info}(D_j)$$

- Information gained by splitting on attribute A:

$$\text{Gain}_A(D) = \text{Info}(D) - \text{Info}_A(D)$$

Example

- Predict if somebody will buy a computer
- Given data set:

Age	Income	Student	Credit_rating	Buys_computer
≤ 30	High	No	Bad	No
≤ 30	High	No	Good	No
31…40	High	No	Bad	Yes
> 40	Medium	No	Bad	Yes
> 40	Low	Yes	Bad	Yes
> 40	Low	Yes	Good	No
31…40	Low	Yes	Good	Yes
≤ 30	Medium	No	Bad	No
≤ 30	Low	Yes	Bad	Yes
> 40	Medium	Yes	Bad	Yes
≤ 30	Medium	Yes	Good	Yes
31…40	Medium	No	Good	Yes
31…40	High	Yes	Bad	Yes
> 40	Medium	No	Good	No

25

Information Gain Example

- Class P: buys_computer = “yes”
- Class N: buys_computer = “no”

$$\text{Info}(D) = I(9,5) = -\frac{9}{14}\log_2\frac{9}{14} - \frac{5}{14}\log_2\frac{5}{14} = 0.940$$

Age	#yes	#no	I(#yes, #no)
≤ 30	2	3	0.971
31…40	4	0	0
>40	3	2	0.971

Age	Income	Student	Credit_rating	Buys_computer
≤ 30	High	No	Bad	No
≤ 30	High	No	Good	No
31…40	High	No	Bad	Yes
> 40	Medium	No	Bad	Yes
> 40	Low	Yes	Bad	Yes
> 40	Low	Yes	Good	No
31…40	Low	Yes	Good	Yes
≤ 30	Medium	No	Bad	No
≤ 30	Low	Yes	Bad	Yes
> 40	Medium	Yes	Bad	Yes
≤ 30	Medium	Yes	Good	Yes
31…40	Medium	No	Good	Yes
31…40	High	Yes	Bad	Yes
> 40	Medium	No	Good	No

$$\text{Info}_{age}(D) = \frac{5}{14}I(2,3) + \frac{4}{14}I(4,0) + \frac{5}{14}I(3,2) = 0.694$$

- $\frac{5}{14}I(2,3)$ means “age ≤ 30” has 5 out of 14 samples, with 2 yes’es and 3 no’s.
  - Similar for the other terms
- Hence
  $$\text{Gain}_{age}(D) = \text{Info}(D) - \text{Info}_{age}(D) = 0.246$$
- Similarly,
  $$\text{Gain}_{income}(D) = 0.029$$
  $$\text{Gain}_{student}(D) = 0.151$$
  $$\text{Gain}_{credit\_rating}(D) = 0.048$$
- Therefore we choose age as the splitting attribute

26

Gain Ratio for Attribute Selection

- Information gain is biased towards attributes with a large number of values
- Use gain ratio to normalize information gain:
  - GainRatio$_A$(D) = Gain$_A$(D) / SplitInfo$_A$(D)

$$\text{SplitInfo}_A(D) = -\sum_{j=1}^{v}\frac{|D_j|}{|D|}\log_2\left(\frac{|D_j|}{|D|}\right)$$

- E.g., $\text{SplitInfo}_{income}(D) = -\frac{4}{14}\log_2\frac{4}{14} - \frac{6}{14}\log_2\frac{6}{14} - \frac{4}{14}\log_2\frac{4}{14} = 0.926$
- GainRatio$_{income}$(D) = 0.029/0.926 = 0.031
- Attribute with maximum gain ratio is selected as splitting attribute

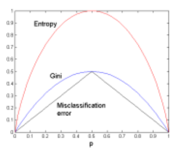
27

Gini Index

- Gini index, gini(D), is defined as $\text{gini}(D) = 1 - \sum_{i=1}^{m} p_i^2$
- If data set D is split on A into v subsets D$_1$,…, D$_v$, the gini index gini$_A$(D) is defined as
  $$\text{gini}_A(D) = \sum_{j=1}^{v}\frac{|D_j|}{|D|}\text{gini}(D_j)$$
- Reduction in Impurity:
  $$\Delta\text{gini}_A(D) = \text{gini}(D) - \text{gini}_A(D)$$
- Attribute that provides smallest gini$_{split}$(D) (= largest reduction in impurity) is chosen to split the node

28

Comparing Attribute Selection Measures



- No clear winner (and there are many more)
  - Information gain:
    - Biased towards multivalued attributes
  - Gain ratio:
    - Tends to prefer unbalanced splits where one partition is much smaller than the others
  - Gini index:
    - Biased towards multivalued attributes
    - Tends to favor tests that result in equal-sized partitions and purity in both partitions

29

Practical Issues of Classification

- Underfitting and overfitting
- Missing values
- Computational cost
- Expressiveness

30

How Good is the Model?

- Training set error: compare prediction of training record with true value
  - Not a good measure for the error on unseen data. (Discussed soon.)
- Test set error: for records that were **not** used for training, compare model prediction and true value
  - Use holdout data from available data set

Training versus Test Set Error

- We'll create a training dataset

Five inputs, all bits, are generated in all 32 possible combinations

Output y = copy of e, except a random 25% of the records have y set to the opposite of e

32 records

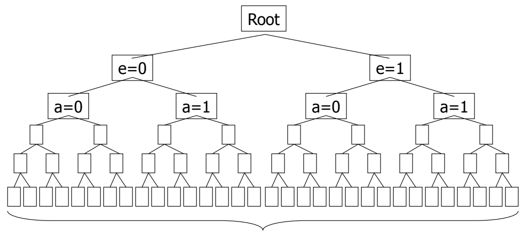
a	b	c	d	e	y
0	0	0	0	0	0
0	0	0	0	1	0
0	0	0	1	0	0
0	0	0	1	1	1
0	0	1	0	0	1
:	:	:	:	:	:
1	1	1	1	1	1

Test Data

- Generate test data using the same method: copy of e, but 25% inverted.
- Some y's that were corrupted in the training set will be uncorrupted in the testing set.
- Some y's that were uncorrupted in the training set will be corrupted in the test set.

a	b	c	d	e	y (training data)	y (test data)
0	0	0	0	0	0	0
0	0	0	0	1	0	1
0	0	0	1	0	0	1
0	0	0	1	1	1	1
0	0	1	0	0	1	1
:	:	:	:	:	:	:
1	1	1	1	1	1	1

Full Tree for The Training Data



25% of these leaf node labels will be corrupted

Each leaf contains exactly one record, hence no error in predicting the training data!

Testing The Tree with The Test Set

	1/4 of the tree nodes are corrupted	3/4 are fine
1/4 of the test set records are corrupted	1/16 of the test set will be correctly predicted for the wrong reasons	3/16 of the test set will be wrongly predicted because the test record is corrupted
3/4 are fine	3/16 of the test predictions will be wrong because the tree node is corrupted	9/16 of the test predictions will be fine

In total, we expect to be wrong on 3/8 of the test set predictions

What's This Example Shown Us?

- Discrepancy between training and test set error
- But more importantly
  - …it indicates that there is something we should do about it if we want to predict well on future data.

Suppose We Had Less Data

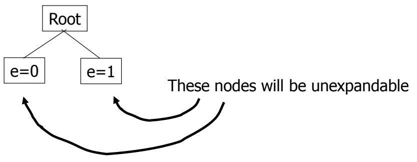
These bits are hidden

Output y = copy of e, except a random 25% of the records have y set to the opposite of e

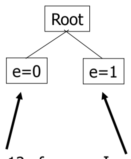
32 records

a	b	c	d	e	y
0	0	0	0	0	0
0	0	0	0	1	0
0	0	0	1	0	0
0	0	0	1	1	1
0	0	1	0	0	1
:	:	:	:	:	:
1	1	1	1	1	1

Tree Learned Without Access to The Irrelevant Bits



Tree Learned Without Access to The Irrelevant Bits



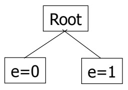
In about 12 of the 16 records in this node the output will be 0

So this will almost certainly predict 0

In about 12 of the 16 records in this node the output will be 1

So this will almost certainly predict 1

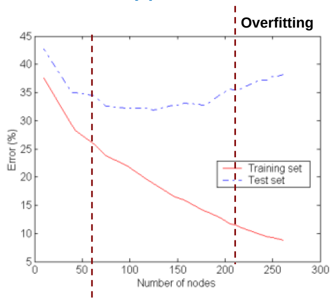
Tree Learned Without Access to The Irrelevant Bits



	almost certainly none of the tree nodes are corrupted	almost certainly all are fine
1/4 of the test set records are corrupted	n/a	1/4 of the test set will be wrongly predicted because the test record is corrupted
3/4 are fine	n/a	3/4 of the test predictions will be fine

In total, we expect to be wrong on only 1/4 of the test set predictions

Typical Observation



Model M overfits the training data if another model M' exists, such that M has smaller error than M' over the training examples, but M' has smaller error than M over the entire distribution of instances.

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

Reasons for Overfitting

- Noise
  - Too closely fitting the training data means the model's predictions reflect the noise as well
- Insufficient training data
  - Not enough data to enable the model to generalize beyond idiosyncrasies of the training records
- Data fragmentation (special problem for trees)
  - Number of instances gets smaller as you traverse down the tree
  - Number of instances at a leaf node could be too small to make any confident decision about class

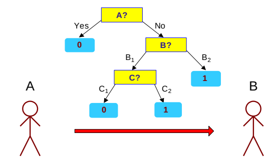
Avoiding Overfitting

- General idea: make the tree smaller
  - Addresses all three reasons for overfitting
- *Pre*pruning: Halt tree construction early
  - Do not split a node if this would result in the goodness measure falling below a threshold
  - Difficult to choose an appropriate threshold, e.g., tree for XOR
- *Post*pruning: Remove branches from a "fully grown" tree
  - Use a set of data different from the training data to decide when to stop pruning
    - Validation data: train tree on training data, prune on validation data, then test on test data

43

Minimum Description Length (MDL)

X	y
X_1	1
X_2	0
X_3	0
X_4	1
…	…
X_n	1

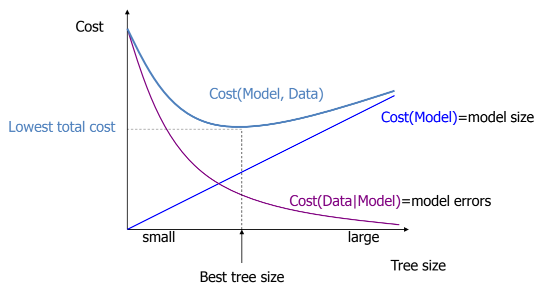


X	y
X_1	?
X_2	?
X_3	?
X_4	?
…	…
X_n	?

- Alternative to using validation data
  - Motivation: data mining is about finding regular patterns in data; regularity can be used to compress the data; method that achieves greatest compression found most regularity and hence is best
- Minimize Cost(Model,Data) = Cost(Model) + Cost(Data|Model)
  - Cost is the number of bits needed for encoding.
    - Cost(Data|Model) encodes the misclassification errors.
    - Cost(Model) uses node encoding plus splitting condition encoding.

44

MDL-Based Pruning Intuition



45

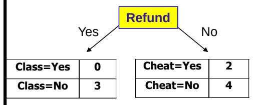
Handling Missing Attribute Values

- Missing values affect decision tree construction in three different ways:
  - How impurity measures are computed
  - How to distribute instance with missing value to child nodes
  - How a test instance with missing value is classified

46

Distribute Instances

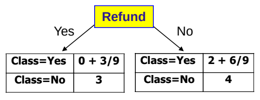
Tid	Refund	Marital Status	Taxable Income	Class
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No



Class=Yes	0
Class=No	3

Cheat=Yes	2
Cheat=No	4

Tid	Refund	Marital Status	Taxable Income	Class
10	?	Single	90K	Yes



Class=Yes	0 + 3/9
Class=No	3

Class=Yes	2 + 6/9
Class=No	4

Probability that Refund=Yes is 3/9

Probability that Refund=No is 6/9

Assign record to the left child with weight = 3/9 and to the right child with weight = 6/9

47

Computing Impurity Measure

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	?	Single	90K	Yes

Before Splitting: Entropy(Parent)
= -0.3 log(0.3)-(0.7)log(0.7) = 0.881

Split on Refund: assume records with missing values are distributed as discussed before

3/9 of record 10 go to Refund=Yes

6/9 of record 10 go to Refund=No

Entropy(Refund=Yes)

= -(1/3 / 10/3)log(1/3 / 10/3)

– (3 / 10/3)log(3 / 10/3) = 0.469

Entropy(Refund=No)
= -(8/3 / 20/3)log(8/3 / 20/3)

– (4 / 20/3)log(4 / 20/3) = 0.971

Entropy(Children)
= 1/3*0.469 + 2/3*0.971 = 0.804

Gain = 0.881 – 0.804 = 0.077

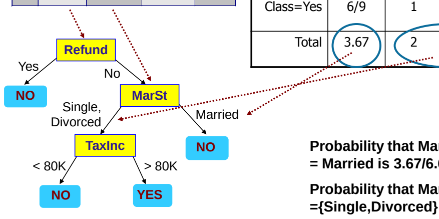
48

Classify Instances

New record:

Tid	Refund	Marital Status	Taxable Income	Class
11	No	?	85K	?

	Married	Single	Divorced	Total
Class=No	3	1	0	4
Class=Yes	6/9	1	1	2.67
Total	3.67	2	1	6.67



Probability that Marital Status = Married is 3.67/6.67

Probability that Marital Status ={Single,Divorced} is 3/6.67

49

Tree Cost Analysis

- Finding an optimal decision tree is NP-complete
  - Optimization goal: minimize expected number of binary tests to uniquely identify any record from a given finite set
- Greedy algorithm
  - O(#attributes * #training_instances * log(#training_instances))
    - At each tree depth, all instances considered
    - Assume tree depth is logarithmic (fairly balanced splits)
    - Need to test each attribute at each node
    - What about binary splits?
      - Sort data once on each attribute, use to avoid re-sorting subsets
      - Incrementally maintain counts for class distribution as different split points are explored
- In practice, trees are considered to be fast both for training (when using the greedy algorithm) and making predictions

50

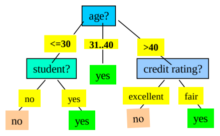
Tree Expressiveness

- Can represent any finite discrete-valued function
  - But it might not do it very efficiently
    - 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 attributes
  - But we can still use a tree for them in practice; it just cannot accurately represent the true function

53

Rule Extraction from a Decision Tree

- One rule is created for each path from the root to a leaf
  - Precondition: conjunction of all split predicates of nodes on path
  - Consequent: class prediction from leaf
- Rules are mutually exclusive and exhaustive
- Example: Rule extraction from buys_computer decision-tree
  - IF age = young AND student = no THEN buys_computer = no
  - IF age = young AND student = yes THEN buys_computer = yes
  - IF age = mid-age THEN buys_computer = yes
  - IF age = old AND credit_rating = excellent THEN buys_computer = yes
  - IF age = young AND credit_rating = fair THEN buys_computer = no



55

Classification in Large Databases

- Scalability: Classify data sets with millions of examples and hundreds of attributes with reasonable speed
- Why use decision trees for data mining?
  - Relatively fast learning speed
  - Can handle all attribute types
  - Convertible to simple and easy to understand classification rules
  - Good classification accuracy, but not as good as newer methods (but tree ensembles are top!)

56

Scalable Tree Induction

- High cost when the training data at a node does not fit in memory
- Solution 1: special I/O-aware algorithm
  - Keep only class list in memory, access attribute values on disk
  - Maintain separate list for each attribute
  - Use count matrix for each attribute
- Solution 2: Sampling
  - Common solution: train tree on a sample that fits in memory
  - More sophisticated versions of this idea exist, e.g., *Rainforest*
    - Build tree on sample, but do this for many bootstrap samples
    - Combine all into a single new tree that is guaranteed to be almost identical to the one trained from entire data set
    - Can be computed with two data scans

57

Tree Conclusions

- Very popular data mining tool
  - Easy to understand
  - Easy to implement
  - Easy to use
    - Little tuning, handles all attribute types and missing values
  - Computationally cheap
- Overfitting problem
- Focused on classification, but easy to extend to prediction (future lecture)

58

Classification and Prediction Overview

- Introduction
- Decision Trees
- Statistical Decision Theory
- Nearest Neighbor
- Bayesian Classification
- Artificial Neural Networks
- Support Vector Machines (SVMs)
- Prediction
- Accuracy and Error Measures
- Ensemble Methods

60

Theoretical Results

- Trees make sense intuitively, but can we get some hard evidence and deeper understanding about their properties?
- Statistical decision theory can give some answers
- Need some probability concepts first

61

Random Variables

- Intuitive version of the definition:
  - Can take on one of possibly many values, each with a certain probability (discrete versus continuous)
  - These probabilities define the probability distribution of the random variable
  - E.g., let X be the outcome of a coin toss, then Pr(X='heads')=0.5 and Pr(X='tails')=0.5; distribution is uniform
- Consider a discrete random variable X with numeric values $x_1,\ldots,x_k$
  - Expectation: $E[X] = \Sigma\, x_i * \Pr(X=x_i)$
  - Variance: $\mathrm{Var}(X) = E[(X - E[X])^2] = E[X^2] - (E[X])^2$

62

Working with Random Variables

- $E[X + Y] = E[X] + E[Y]$
- $\mathrm{Var}(X + Y) = \mathrm{Var}(X) + \mathrm{Var}(Y) + 2\,\mathrm{Cov}(X,Y)$
- For constants a, b
  - $E[aX + b] = a\,E[X] + b$
  - $\mathrm{Var}(aX + b) = \mathrm{Var}(aX) = a^2\,\mathrm{Var}(X)$
- Iterated expectation:
  - $E[X] = E_X[\,E_Y[Y \mid X]\,]$, where $E_Y[Y \mid X] = \Sigma y_i * \Pr(Y=y_i \mid X=x)$ is the expectation of Y for a given value of X, i.e., is a function of X
  - In general for any function f(X,Y): $E_{X,Y}[f(X,Y)] = E_X[\,E_Y[f(X,Y) \mid X]\,]$

63

What is the Optimal Model f(X)?

Let X denote a real - valued random input variable and Y a real - valued random output vaiable

The squared error of trained model *f(X)* is $\mathrm{E}_{X,Y}\left[(Y - f(X))^2\right]$

Which function *f(X)* will minimize the squared error?

Consider the error for a specific value of X and let $\bar{Y} = \mathrm{E}_Y[Y \mid X]$:

$$\mathrm{E}_Y\left[(Y - f(X))^2 \mid X\right] = \mathrm{E}_Y\left[(Y - \bar{Y} + \bar{Y} - f(X))^2 \mid X\right]$$
$$= \mathrm{E}_Y\left[(Y - \bar{Y})^2 \mid X\right] + \mathrm{E}_Y\left[(\bar{Y} - f(X))^2 \mid X\right] + 2\,\mathrm{E}_Y\left[(Y - \bar{Y})(\bar{Y} - f(X)) \mid X\right]$$
$$= \mathrm{E}_Y\left[(Y - \bar{Y})^2 \mid X\right] + \left(\bar{Y} - f(X)\right)^2 + 2(\bar{Y} - f(X))\,\mathrm{E}_Y\left[(Y - \bar{Y}) \mid X\right]$$
$$= \mathrm{E}_Y\left[(Y - \bar{Y})^2 \mid X\right] + \left(\bar{Y} - f(X)\right)^2$$

$(\text{Notice}: \mathrm{E}_Y\left[(Y - \bar{Y}) \mid X\right] = \mathrm{E}_Y\left[Y \mid X\right] - \mathrm{E}_Y\left[\bar{Y} \mid X\right] = \bar{Y} - \bar{Y} = 0)$

64

Optimal Model f(X) (cont.)

The choice of $f(X)$ does not affect $\mathrm{E}_Y\left[(Y-\bar{Y})^2 \mid X\right]$, but $\left(\bar{Y}-f(X)\right)^2$ is minimized for $f(X)=\bar{Y}=\mathrm{E}_Y[Y \mid X]$.

Note that $\mathrm{E}_{X,Y}\left[(Y-f(X))^2\right]=\mathrm{E}_X\left[\mathrm{E}_Y\left[(Y-f(X))^2 \mid X\right]\right]$. Hence

$$\mathrm{E}_{X,Y}\left[(Y-f(X))^2\right]=\mathrm{E}_X\left[\mathrm{E}_Y\left[(Y-\bar{Y})^2 \mid X\right]+\left(\bar{Y}-f(X)\right)^2\right]$$

Hence the squared error is minimzed by choosing $f(X)=\mathrm{E}_Y[Y \mid X]$ for every X.

(Notice that for minimizing absolute error $\mathrm{E}_{X,Y}\left[\,|Y-f(X)|\,\right]$, one can show that the best model is $f(X)=\mathrm{median}(X \mid Y)$.)

65

Implications for Trees

- Best prediction for input X=x is the mean of the Y-values of all records (x(i),y(i)) with x(i)=x
- What about classification?
  - Two classes: encode as 0 and 1, use squared error as before
    - Get f(X) = E[Y| X=x] = 1*Pr(Y=1| X=x) + 0*Pr(Y=0| X=x) = Pr(Y=1| X=x)
  - K classes: can show that for 0-1 loss (error = 0 if correct class, error = 1 if wrong class predicted) the optimal choice is to return the majority class for a given input X=x
    - Called the Bayes classifier
- Problem: How can we estimate E[Y| X=x] or the majority class for X=x from the training data?
  - Often there is just one or no training record for a given X=x
- Solution: approximate it
  - Use Y-values from training records in *neighborhood* around X=x
  - Tree: leaf defines neighborhood in the data space; make sure there are enough records in the leaf to obtain reliable estimate of correct answer

66

Bias-Variance Tradeoff

- Let's take this one step further and see if we can understand overfitting through statistical decision theory
- As before, consider two random variables X and Y
- From a training set D with n records, we want to construct a function f(X) that returns good approximations of Y for future inputs X
  - Make dependence of f on D explicit by writing f(X; D)
- Goal: minimize mean squared error over all X, Y, and D, i.e., $\mathrm{E}_{X,D,Y}$[(Y - f(X; D))^2]

67

Bias-Variance Tradeoff Derivation

$E_{X,D,Y}\left[(Y-f(X;D))^2\right]=E_X E_D E_Y\left[(Y-f(X;D))^2 \mid X,D\right]$. Now consider the inner term:

$E_D E_Y\left[(Y-f(X;D))^2 \mid X,D\right]=E_D\left[E_Y\left[(Y-E[Y \mid X])^2 \mid X,D\right]+\left(f(X;D)-E[Y \mid X]\right)^2\right]$

(Same derivation as before for optimal function f(X).)

$=E_Y\left[(Y-E[Y \mid X])^2 \mid X\right]+E_D\left[\left(f(X;D)-E[Y \mid X]\right)^2\right]$

(The first term does not depend on D, hence $E_D\left[E_Y\left[(Y-E[Y \mid X])^2 \mid X,D\right]\right]=E_Y\left[(Y-E[Y \mid X])^2 \mid X\right]$)

Consider the second term:

$$\begin{aligned}
E_D\left[\left(f(X;D)-E[Y \mid X]\right)^2\right] &= E_D\left[\left(\left(f(X;D)-E_D[f(X;D)\right)+\left(E_D[f(X;D)]-E[Y \mid X]\right)\right)^2\right]\\
&= E_D\left[\left(f(X;D)-E_D[f(X;D)]\right)^2\right]+E_D\left[\left(E_D[f(X;D)]-E[Y \mid X]\right)^2\right]\\
&\quad +2E_D\left[\left(\left(f(X;D)-E_D[f(X;D)\right)\cdot\left(E_D[f(X;D)]-E[Y \mid X]\right)\right)\right]\\
&= E_D\left[\left(f(X;D)-E_D[f(X;D)]\right)^2\right]+\left(E_D[f(X;D)]-E[Y \mid X]\right)^2\\
&\quad +2E_D\left[f(X;D)-E_D[f(X;D)]\right]\cdot\left(E_D[f(X;D)]-E[Y \mid X]\right)\\
&= E_D\left[\left(f(X;D)-E_D[f(X;D)]\right)^2\right]+\left(E_D[f(X;D)]-E[Y \mid X]\right)^2
\end{aligned}$$

(The third term is zero, because $E_D\left[f(X;D)-E_D[f(X;D)]\right]=E_D[f(X;D)]-E_D[f(X;D)]=0$.)

Overall we therefore obtain:

$$E_{X,D,Y}\left[(Y-f(X;D))^2\right]=E_X\left[\left(E_D[f(X;D)]-E[Y \mid X]\right)^2+E_D\left[\left(f(X;D)-E_D[f(X;D)]\right)^2\right]+E_Y\left[(Y-E[Y \mid X])^2 \mid X\right]\right]$$

68

Bias-Variance Tradeoff and Overfitting

$\left(E_D[f(X;D)]-E[Y \mid X]\right)^2$: **bias**

$E_D\left[\left(f(X;D)-E_D[f(X;D)]\right)^2\right]$: **variance**

$E_Y\left[(Y-E[Y \mid X])^2 \mid X\right]$: **irreducible error** (does not depend on f and is simply the variance of Y given X.)

- Option 1: f(X;D) = E[Y| X,D]
  - Bias: since E_D[E[Y| X,D]] = E[Y| X], bias is zero
  - Variance: (E[Y| X,D]-E_D[E[Y| X,D]])^2 = (E[Y| X,D]-E[Y| X])^2 can be very large since E[Y| X,D] depends heavily on D
  - Might overfit!
- Option 2: f(X;D)=X (or other function independent of D)
  - Variance: (X-E_D[X])^2=(X-X)^2=0
  - Bias: (E_D[X]-E[Y| X])^2=(X-E[Y| X])^2 can be large, because E[Y| X] might be completely different from X
  - Might underfit!
- Find best compromise between fitting training data too closely (option 1) and completely ignoring it (option 2)

69

Implications for Trees

- Bias decreases as tree becomes larger
  - Larger tree can fit training data better
- Variance increases as tree becomes larger
  - Sample variance affects predictions of larger tree more
- Find right tradeoff as discussed earlier
  - Validation data to find best pruned tree
  - MDL principle

70

Classification and Prediction Overview

- Introduction
- Decision Trees
- Statistical Decision Theory
- Nearest Neighbor
- Bayesian Classification
- Artificial Neural Networks
- Support Vector Machines (SVMs)
- Prediction
- Accuracy and Error Measures
- Ensemble Methods

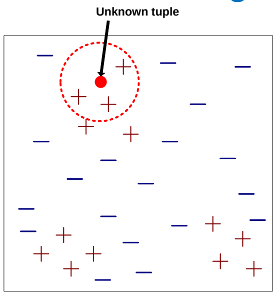
Lazy vs. Eager Learning

- Lazy learning: Simply stores training data (or only minor processing) and waits until it is given a test record
- Eager learning: Given a training set, constructs a classification model before receiving new (test) data to classify
- General trend: Lazy = faster training, slower predictions
- Accuracy: not clear which one is better!
  - Lazy method: typically driven by local decisions
  - Eager method: driven by global and local decisions

Nearest-Neighbor

- Recall our statistical decision theory analysis: Best prediction for input X=x is the mean of the Y-values of all records (x(i),y(i)) with x(i)=x (majority class for classification)
- Problem was to estimate E[Y| X=x] or majority class for X=x from the training data
- Solution was to approximate it
  - Use Y-values from training records in neighborhood around X=x

Nearest-Neighbor Classifiers

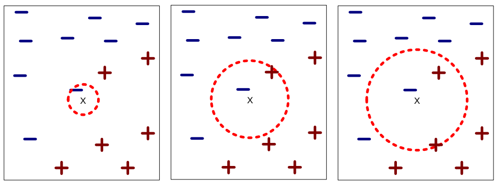


- Requires:
  - Set of stored records
  - Distance metric for pairs of records
    - Common choice: Euclidean

$$d(\mathbf{p},\mathbf{q}) = \sqrt{\sum_i (p_i - q_i)^2}$$

  - Parameter k
    - Number of nearest neighbors to retrieve
- To classify a record:
  - Find its k nearest neighbors
  - Determine output based on (distance-weighted) average of neighbors' output

Definition of Nearest Neighbor

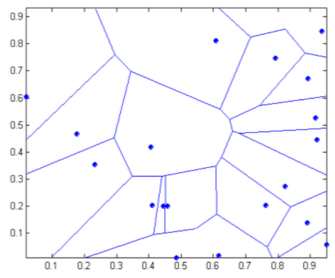


(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

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

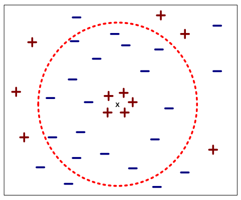
1-Nearest Neighbor

Voronoi Diagram

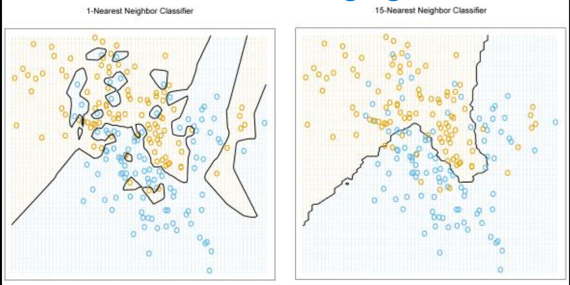


Nearest Neighbor Classification

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



Effect of Changing k



Source: Hastie, Tibshirani, and Friedman. The Elements of Statistical Learning

Explaining the Effect of k

- Recall the bias-variance tradeoff
- Small k, i.e., predictions based on few neighbors
  - High variance, low bias
- Large k, e.g., average over entire data set
  - Low variance, but high bias
- Need to find k that achieves best tradeoff
- Can do that using validation data

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
  - Income difference would dominate record distance

Other Problems

- 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 1 0	vs	1 0 0 0 0 0 0 0 0 0 0 0
0 1 1 1 1 1 1 1 1 1 1 1		0 0 0 0 0 0 0 0 0 0 0 1
d = 1.4142		**d = 1.4142**

  - Solution: Normalize the vectors to unit length
- Irrelevant attributes might dominate distance
  - Solution: eliminate them

Computational Cost

- Brute force: O(#trainingRecords)
  - For each training record, compute distance to test record, keep if among top-k
- Pre-compute Voronoi diagram (expensive), then search spatial index of Voronoi cells: if lucky O(log(#trainingRecords))
- Store training records in multi-dimensional search tree, e.g., R-tree: if lucky O(log(#trainingRecords))
- Bulk-compute predictions for many test records using spatial join between training and test set
  - Same worst-case cost as one-by-one predictions, but usually much faster in practice

Classification and Prediction Overview

- Introduction
- Decision Trees
- Statistical Decision Theory
- Nearest Neighbor
- Bayesian Classification
- Artificial Neural Networks
- Support Vector Machines (SVMs)
- Prediction
- Accuracy and Error Measures
- Ensemble Methods

99

Bayesian Classification

- Performs probabilistic prediction, i.e., predicts class membership probabilities
- Based on Bayes' Theorem
- Incremental training
  - Update probabilities as new training records arrive
  - Can combine prior knowledge with observed data
- Even when Bayesian methods are computationally intractable, they can provide a standard of optimal decision making against which other methods can be measured

100

Bayesian Theorem: Basics

- **X** = random variable for data records ("evidence")
- H = hypothesis that specific record **X**=**x** belongs to class C
- Goal: determine P(H| **X**=**x**)
  - Probability that hypothesis holds given a record **x**
- P(H) = prior probability
  - The initial probability of the hypothesis
  - E.g., person **x** will buy computer, regardless of age, income etc.
- P(**X**=**x**) = probability that data record **x** is observed
- P(**X**=**x**| H) = probability of observing record **x**, given that the hypothesis holds
  - E.g., given that **x** will buy a computer, what is the probability that **x** is in age group 31...40, has medium income, etc.?

101

Bayes' Theorem

- Given data record **x**, the posterior probability of a hypothesis H, P(H| **X**=**x**), follows from Bayes theorem:

$$P(H|\mathbf{X}=\mathbf{x}) = \frac{P(\mathbf{X}=\mathbf{x}|H)P(H)}{P(\mathbf{X}=\mathbf{x})}$$

- Informally: posterior = likelihood * prior / evidence
- Among all candidate hypotheses H, find the maximally probably one, called maximum a posteriori (MAP) hypothesis
- Note: P(**X**=**x**) is the same for all hypotheses
- If all hypotheses are equally probable a priori, we only need to compare P(**X**=**x**| H)
  - Winning hypothesis is called the maximum likelihood (ML) hypothesis
- Practical difficulties: requires initial knowledge of many probabilities and has high computational cost

102

Towards Naïve Bayes Classifier

- Suppose there are m classes $C_1, C_2, \ldots, C_m$
- Classification goal: for record **x**, find class C_i that has the maximum posterior probability P(C_i| **X**=**x**)
- Bayes' theorem:

$$P(C_i|\mathbf{X}=\mathbf{x}) = \frac{P(\mathbf{X}=x|C_i)P(C_i)}{P(\mathbf{X}=\mathbf{x})}$$

- Since P(**X**=**x**) is the same for all classes, only need to find maximum of $P(\mathbf{X}=\mathbf{x}|C_i)P(C_i)$

103

Computing P(**X**=**x**|C_i) and P(C_i)

- Estimate P(C_i) by counting the frequency of class C_i in the training data
- Can we do the same for P(**X**=**x**|C_i)?
  - Need very large set of training data
  - Have $|X_1|*|X_2|*\ldots*|X_d|*m$ different combinations of possible values for X and C_i
  - Need to see every instance **x** many times to obtain reliable estimates
- Solution: decompose into lower-dimensional problems

104

Example: Computing P(**X**=**x**|C_i) and P(C_i)

- P(buys_computer = yes) = 9/14
- P(buys_computer = no) = 5/14
- P(age>40, income=low, student=no, credit_rating=bad| buys_computer=yes) = 0 ?

Age	Income	Student	Credit_rating	Buys_computer
≤ 30	High	No	Bad	No
≤ 30	High	No	Good	No
31…40	High	No	Bad	Yes
> 40	Medium	No	Bad	Yes
> 40	Low	Yes	Bad	Yes
> 40	Low	Yes	Good	No
31...40	Low	Yes	Good	Yes
≤ 30	Medium	No	Bad	No
≤ 30	Low	Yes	Bad	Yes
> 40	Medium	Yes	Bad	Yes
≤ 30	Medium	Yes	Good	Yes
31...40	Medium	No	Good	Yes
31...40	High	Yes	Bad	Yes
> 40	Medium	No	Good	No

Conditional Independence

- X, Y, Z random variables
- X is conditionally independent of Y, given Z, if P(X| Y,Z) = P(X| Z)
  - Equivalent to: P(X,Y| Z) = P(X| Z) * P(Y| Z)
- Example: people with longer arms read better
  - Confounding factor: age
    - Young child has shorter arms and lacks reading skills of adult
  - If age is fixed, observed relationship between arm length and reading skills disappears

Derivation of Naïve Bayes Classifier

- Simplifying assumption: all input attributes conditionally independent, given class

$$P(\mathbf{X}=(x_1,\ldots,x_d)\,|\,C_i)=\prod_{k=1}^{d}P(X_k=x_k\,|\,C_i)=P(X_1=x_1\,|\,C_i)\cdot P(X_2=x_2\,|\,C_i)\cdots P(X_d=x_d\,|\,C_i)$$

- Each P(X_k=x_k| C_i) can be estimated robustly
  - If X_k is categorical attribute
    - P(X_k=x_k| C_i) = #records in C_i that have value x_k for X_k, divided by #records of class C_i in training data set
  - If X_k is continuous, we could discretize it
    - Problem: interval selection
      - Too many intervals: too few training cases per interval
      - Too few intervals: limited choices for decision boundary

Estimating P(X_k=x_k| C_i) for Continuous Attributes without Discretization

- P(X_k=x_k| C_i) computed based on Gaussian distribution with mean μ and standard deviation σ:

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

as

$$\mathrm{P}(X_k=x_k\,|\,C_i)=g(x_k,\mu_{k,C_i},\sigma_{k,C_i})$$

- Estimate $\mu_{k,Ci}$ from sample mean of attribute X_k for all training records of class C_i
- Estimate $\sigma_{k,Ci}$ similarly from sample

Naïve Bayes Example

- Classes:
  - C_1:buys_computer = yes
  - C_2:buys_computer = no
- Data sample **x**
  - age ≤ 30,
  - income = medium,
  - student = yes, and
  - credit_rating = fair

Age	Income	Student	Credit_rating	Buys_computer
≤ 30	High	No	Bad	No
≤ 30	High	No	Good	No
31…40	High	No	Bad	Yes
> 40	Medium	No	Bad	Yes
> 40	Low	Yes	Bad	Yes
> 40	Low	Yes	Good	No
31...40	Low	Yes	Good	Yes
≤ 30	Medium	No	Bad	No
≤ 30	Low	Yes	Bad	Yes
> 40	Medium	Yes	Bad	Yes
≤ 30	Medium	Yes	Good	Yes
31...40	Medium	No	Good	Yes
31...40	High	Yes	Bad	Yes
> 40	Medium	No	Good	No

Naïve Bayesian Computation

- Compute P(C_i) for each class:
  - P(buys_computer = “yes”) = 9/14 = 0.643
  - P(buys_computer = “no”) = 5/14= 0.357
- Compute P(X_k=x_k| C_i) for each class
  - P(age = “≤ 30” | buys_computer = “yes”) = 2/9 = 0.222
  - P(age = “≤ 30” | buys_computer = “no”) = 3/5 = 0.6
  - P(income = “medium” | buys_computer = “yes”) = 4/9 = 0.444
  - P(income = “medium” | buys_computer = “no”) = 2/5 = 0.4
  - P(student = “yes” | buys_computer = “yes) = 6/9 = 0.667
  - P(student = “yes” | buys_computer = “no”) = 1/5 = 0.2
  - P(credit_rating = “fair” | buys_computer = “yes”) = 6/9 = 0.667
  - P(credit_rating = “fair” | buys_computer = “no”) = 2/5 = 0.4
- Compute P(**X**=**x**| C_i) using the Naive Bayes assumption
  - P(≤30, medium, yes, fair |buys_computer = “yes”) = 0.222 * 0.444 * 0.667 * 0.667 = 0.044
  - P(≤30, medium, yes, fair | buys_computer = “no”) = 0.6 * 0.4 * 0.2 * 0.4 = 0.019
- Compute final result P(**X**=**x**| C_i) * P(C_i)
  - P(**X**=**x** | buys_computer = “yes”) * P(buys_computer = “yes”) = 0.028
  - P(**X**=**x** | buys_computer = “no”) * P(buys_computer = “no”) = 0.007
- Therefore we predict buys_computer = “yes” for input **x** = (age = “≤30”, income = “medium”, student = “yes”, credit_rating = “fair”)

Zero-Probability Problem

- Naïve Bayesian prediction requires each conditional probability to be non-zero (why?)

$$P(\mathbf{X}=(x_1,\ldots,x_d)\,|\,C_i)=\prod_{k=1}^{d}P(X_k=x_k\,|\,C_i)=P(X_1=x_1\,|\,C_i)\cdot P(X_2=x_2\,|\,C_i)\cdots P(X_d=x_d\,|\,C_i)$$

- Example: 1000 records for buys_computer=yes with income=low (0), income= medium (990), and income = high (10)
  - For input with income=low, conditional probability is zero
- Use Laplacian correction (or Laplace estimator) by adding 1 dummy record to each income level
    - Prob(income = low) = 1/1003
    - Prob(income = medium) = 991/1003
    - Prob(income = high) = 11/1003
  - "Corrected" probability estimates close to their "uncorrected" counterparts, but none is zero

111

Naïve Bayesian Classifier: Comments

- Easy to implement
- Good results obtained in many cases
  - Robust to isolated noise points
  - Handles missing values by ignoring the instance during probability estimate calculations
  - Robust to irrelevant attributes
- Disadvantages
  - Assumption: class conditional independence, therefore loss of accuracy
  - Practically, dependencies exist among variables
- How to deal with these dependencies?

112

Probabilities

- Summary of elementary probability facts we have used already and/or will need soon
- Let X be a random variable as usual
- Let A be some predicate over its possible values
  - A is true for some values of X, false for others
  - E.g., X is outcome of throw of a die, A could be "value is greater than 4"
- P(A) is the fraction of possible worlds in which A is true
  - P(die value is greater than 4) = 2 / 6 = 1/3

113

Axioms

- $0 \le P(A) \le 1$
- P(True) = 1
- P(False) = 0
- $P(A \vee B) = P(A) + P(B) - P(A \wedge B)$

114

Theorems from the Axioms

- $0 \le P(A) \le 1$, P(True) = 1, P(False) = 0
- $P(A \vee B) = P(A) + P(B) - P(A \wedge B)$

- From these we can prove:
  - P(not A) = P(~A) = 1 - P(A)
  - $P(A) = P(A \wedge B) + P(A \wedge \sim B)$

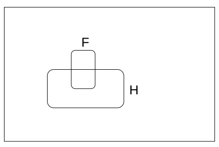
115

Conditional Probability

- P(A|B) = Fraction of worlds in which B is true that also have A true

H = "Have a headache"
F = "Coming down with Flu"

P(H) = 1/10
P(F) = 1/40
P(H|F) = 1/2



"Headaches are rare and flu is rarer, but if you're coming down with flu there's a 50-50 chance you'll have a headache."

116

Definition of Conditional Probability

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

Corollary: the Chain Rule

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

117

Multivalued Random Variables

- Suppose X can take on more than 2 values
- X is a random variable with arity k if it can take on exactly one value out of $\{v_1, v_2, \ldots, v_k\}$
- Thus

$$P(X = v_i \wedge X = v_j) = 0 \text{ if } i \neq j$$

$$P(X = v_1 \vee X = v_2 \vee \ldots \vee X = v_k) = 1$$

118

Easy Fact about Multivalued Random Variables

- Using the axioms of probability
  - $0 \leq P(A) \leq 1$, P(True) = 1, P(False) = 0
  - $P(A \vee B) = P(A) + P(B) - P(A \wedge B)$
- And assuming that X obeys

$$P(X = v_i \wedge X = v_j) = 0 \text{ if } i \neq j$$

$$P(X = v_1 \vee X = v_2 \vee \ldots \vee X = v_k) = 1$$

- We can prove that

$$P(X = v_1 \vee X = v_2 \vee \ldots \vee X = v_i) = \sum_{j=1}^{i} P(X = v_j)$$

- And therefore: $\sum_{j=1}^{k} P(X = v_j) = 1$

119

Useful Easy-to-Prove Facts

$$P(A|B) + P(\sim A|B) = 1$$

$$\sum_{j=1}^{k} P(X = v_j | B) = 1$$

120

The Joint Distribution

Example: Boolean variables A, B, C

Recipe for making a joint distribution of d variables:

121

The Joint Distribution

Example: Boolean variables A, B, C

Recipe for making a joint distribution of d variables:

1. Make a truth table listing all combinations of values of your variables (has 2^d rows for d Boolean variables).

A	B	C
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

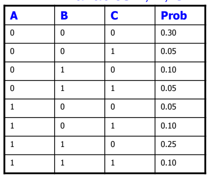
122

The Joint Distribution

Example: Boolean variables A, B, C

Recipe for making a joint distribution of d variables:

1. Make a truth table listing all combinations of values of your variables (has 2^d rows for d Boolean variables).
2. For each combination of values, say how probable it is.



A	B	C	Prob
0	0	0	0.30
0	0	1	0.05
0	1	0	0.10
0	1	1	0.05
1	0	0	0.05
1	0	1	0.10
1	1	0	0.25
1	1	1	0.10

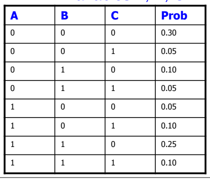
123

The Joint Distribution

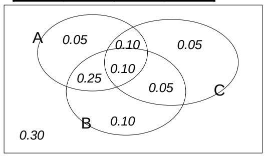
Example: Boolean variables A, B, C

Recipe for making a joint distribution of d variables:

1. Make a truth table listing all combinations of values of your variables (has 2^d rows for d Boolean variables).
2. For each combination of values, say how probable it is.
3. If you subscribe to the axioms of probability, those numbers must sum to 1.

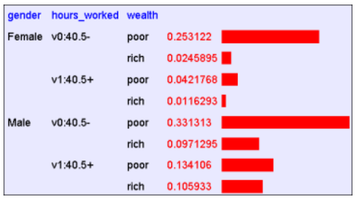


A	B	C	Prob
0	0	0	0.30
0	0	1	0.05
0	1	0	0.10
0	1	1	0.05
1	0	0	0.05
1	0	1	0.10
1	1	0	0.25
1	1	1	0.10



124

Using the Joint Dist.



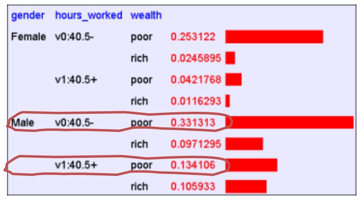
gender	hours_worked	wealth		
Female	v0:40.5-	poor	0.253122	🟥
		rich	0.0245895	🟥
	v1:40.5+	poor	0.0421768	🟥
		rich	0.0116293	🟥
Male	v0:40.5-	poor	0.331313	🟥
		rich	0.0971295	🟥
	v1:40.5+	poor	0.134106	🟥
		rich	0.105933	🟥

Once you have the JD you can ask for the probability of any logical expression involving your attribute

$$P(E) = \sum_{\text{rows matching } E} P(\text{row})$$

125

Using the Joint Dist.



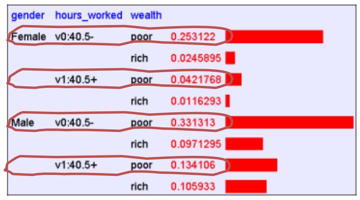
gender	hours_worked	wealth		
Female	v0:40.5-	poor	0.253122	🟥
		rich	0.0245895	🟥
	v1:40.5+	poor	0.0421768	🟥
		rich	0.0116293	🟥
Male	v0:40.5-	poor	0.331313	🟥
		rich	0.0971295	🟥
	v1:40.5+	poor	0.134106	🟥
		rich	0.105933	🟥

P(Poor $\wedge$ Male) = 0.4654

$$P(E) = \sum_{\text{rows matching } E} P(\text{row})$$

126

Using the Joint Dist.



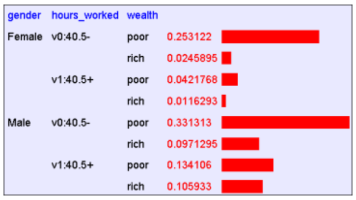
gender	hours_worked	wealth		
Female	v0:40.5-	poor	0.253122	🟥
		rich	0.0245895	🟥
	v1:40.5+	poor	0.0421768	🟥
		rich	0.0116293	🟥
Male	v0:40.5-	poor	0.331313	🟥
		rich	0.0971295	🟥
	v1:40.5+	poor	0.134106	🟥
		rich	0.105933	🟥

P(Poor) = 0.7604

$$P(E) = \sum_{\text{rows matching } E} P(\text{row})$$

127

Inference with the Joint Dist.

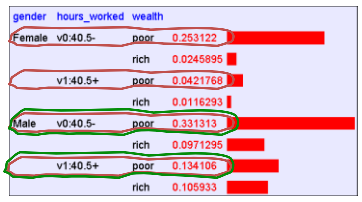


gender	hours_worked	wealth		
Female	v0:40.5-	poor	0.253122	🟥
		rich	0.0245895	🟥
	v1:40.5+	poor	0.0421768	🟥
		rich	0.0116293	🟥
Male	v0:40.5-	poor	0.331313	🟥
		rich	0.0971295	🟥
	v1:40.5+	poor	0.134106	🟥
		rich	0.105933	🟥

$$P(E_1 \mid E_2) = \frac{P(E_1 \wedge E_2)}{P(E_2)} = \frac{\sum_{\text{rows matching } E_1 \text{ and } E_2} P(\text{row})}{\sum_{\text{rows matching } E_2} P(\text{row})}$$

128

Inference with the Joint Dist.



gender	hours_worked	wealth	
Female	v0:40.5-	poor	0.253122
		rich	0.0245895
	v1:40.5+	poor	0.0421768
		rich	0.0116293
Male	v0:40.5-	poor	0.331313
		rich	0.0971295
	v1:40.5+	poor	0.134106
		rich	0.105933

$$P(E_1 \mid E_2) = \frac{P(E_1 \wedge E_2)}{P(E_2)} = \frac{\sum_{\text{rows matching } E_1 \text{ and } E_2} P(\text{row})}{\sum_{\text{rows matching } E_2} P(\text{row})}$$

P(Male | Poor) = 0.4654 / 0.7604 = 0.612

Joint Distributions

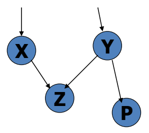
- Good news: Once you have a joint distribution, you can answer important questions that involve uncertainty.
- Bad news: Impossible to create joint distribution for more than about ten attributes because there are so many numbers needed when you build it.

What Would Help?

- Full independence
  - P(gender=g ∧ hours_worked=h ∧ wealth=w) = P(gender=g) * P(hours_worked=h) * P(wealth=w)
  - Can reconstruct full joint distribution from a few marginals
- Full conditional independence given class value
  - Naïve Bayes
- What about something between Naïve Bayes and general joint distribution?

Bayesian Belief Networks

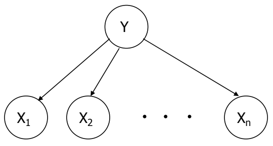
- Subset of the variables conditionally independent
- Graphical model of causal relationships
  - Represents dependency among the variables
  - Gives a specification of joint probability distribution



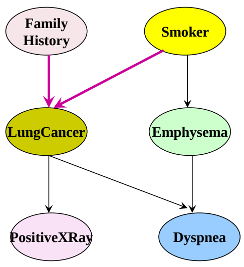
- ❑ Nodes: random variables
- ❑ Links: dependency
- ❑ X and Y are the parents of Z, and Y is the parent of P
- ❑ Given Y, Z and P are independent
- ❑ Has no loops or cycles

Bayesian Network Properties

- Each variable is conditionally independent of its non-descendents in the graph, given its parents
- Naïve Bayes as a Bayesian network:



Bayesian Belief Network Example



Bayesian Belief Networks

Conditional probability table (**CPT**) for variable LungCancer:

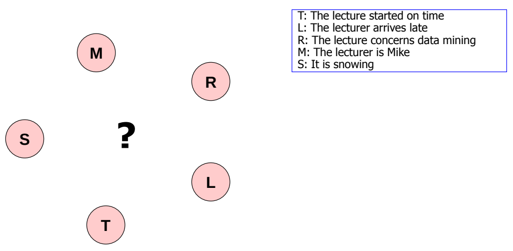
	(FH, S)	(FH, ~S)	(~FH, S)	(~FH, ~S)
LC	0.8	0.5	0.7	0.1
~LC	0.2	0.5	0.3	0.9

CPT shows the conditional probability for each possible combination of its parents

Easy to compute joint distribution for all attributes $X_1, \ldots, X_d$, from CPT:

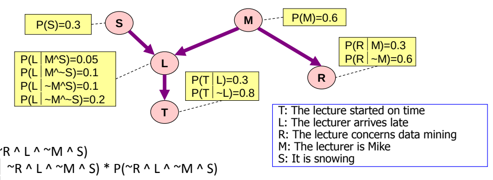
$$P(\mathbf{X} = (x_1, \ldots, x_d)) = \prod_{i=1}^{d} P(X_i = x_i \mid \text{parents}(X_i))$$

Creating a Bayes Network


T: The lecture started on time
L: The lecturer arrives late
R: The lecture concerns data mining
M: The lecturer is Mike
S: It is snowing
M
R
S
?
L
T

135

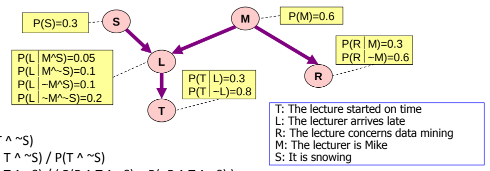
Computing with Bayes Net



P(T ^ ~R ^ L ^ ~M ^ S)
= P(T | ~R ^ L ^ ~M ^ S) * P(~R ^ L ^ ~M ^ S)
= P(T | L) * P(~R ^ L ^ ~M ^ S)
= P(T | L) * P(~R | L ^ ~M ^ S) * P(L^~M^S)
= P(T | L) * P(~R | ~M) * P(L ^ ~M ^ S)
= P(T | L) * P(~R | ~M) * P(L | ~M ^ S) * P(~M ^ S)
= P(T | L) * P(~R | ~M) * P(L | ~M ^ S) * P(~M | S) * P(S)
= P(T | L) * P(~R | ~M) * P(L | ~M ^ S) * P(~M) * P(S)

136

Computing with Bayes Net



P(R | T ^ ~S)
= P(R ^ T ^ ~S) / P(T ^ ~S)
= P(R ^ T ^ ~S) / (P(R ^ T ^ ~S) + P(~R ^ T ^ ~S))

P(R ^ T ^ ~S): Compute as P(L ^ M ^ R ^ T ^ ~S) + P(~L ^ M ^ R ^ T ^ ~S)
+ P(L ^ ~M ^ R ^ T ^ ~S) + P(~L ^ ~M ^ R ^ T ^ ~S)
Compute P(~R ^ T ^ ~S) similarly

Any problem here? Yes, possibly many terms to be computed...

137

Inference with Bayesian Networks

- Want to compute P(C_i| **X**=**x**)
  - Assume the output attribute Y node's parents are all input attribute nodes and all these input values are given
  - Then we have P(C_i| **X**=**x**) = P(C_i| parents(Y)), i.e., we can read it directly from CPT
- What if values are given only for a subset of attributes?
  - Can still compute it from the Bayesian network
  - But: exact inference of probabilities in general for an arbitrary Bayesian network is NP-hard
  - Solutions: probabilistic inference, trade precision for efficiency

138

Training Bayesian Networks

- Several scenarios:
  - Given both the network structure and all variables are observable: learn only the CPTs
  - Network structure known, some hidden variables: gradient descent (greedy hill-climbing) method, analogous to neural network learning
  - Network structure unknown, all variables observable: search through the model space to reconstruct network topology
  - Unknown structure, all hidden variables: No good algorithms known for this purpose
- Ref.: D. Heckerman: Bayesian networks for data mining

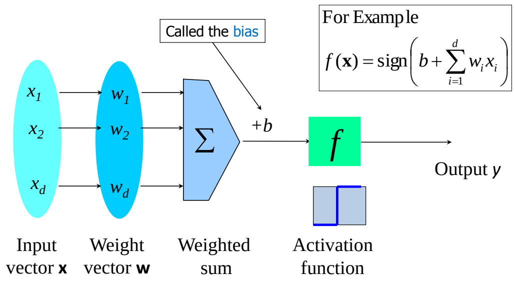
139

Classification and Prediction Overview

- Introduction
- Decision Trees
- Statistical Decision Theory
- Nearest Neighbor
- Bayesian Classification
- Artificial Neural Networks
- Support Vector Machines (SVMs)
- Prediction
- Accuracy and Error Measures
- Ensemble Methods

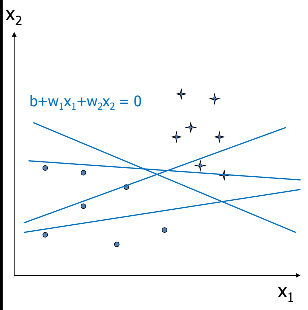
141

Basic Building Block: Perceptron



142

Perceptron Decision Hyperplane



Input:	$\{(x_1, x_2, y), \ldots\}$
Output:	classification function f(**x**)
	f(**x**) > 0: return +1
	f(**x**) ≤ 0: return = -1

Decision hyperplane: b+**w**·**x** = 0

Note: b+**w**·**x** > 0, if and only if

$$\sum_{i=1}^{d} w_i x_i > -b$$

b represents a threshold for when the perceptron "fires".

143

Representing Boolean Functions

- AND with two-input perceptron
  - $b=-0.8$, $w_1=w_2=0.5$
- OR with two-input perceptron
  - $b=-0.3$, $w_1=w_2=0.5$
- m-of-n function: true if at least m out of n inputs are true
  - All input weights 0.5, threshold weight b is set according to m, n
- Can also represent NAND, NOR
- What about XOR?

144

Perceptron Training Rule

- Goal: correct +1/-1 output for each training record
- Start with random weights, select constant η (learning rate)
- For each training record (**x**, y)
  - Let f_{old}(**x**) be the output of the current perceptron for **x**
  - Set b:= b + Δb, where Δb = η(y - f_{old}(**x**))
  - For all i, set $w_i := w_i + \Delta w_i$, where $\Delta w_i = \eta(y - f_{old}(\mathbf{x}))x_i$
- Keep iterating over training records until all are correctly classified
- Converges to correct decision boundary, if the classes are linearly separable and a small enough η is used
  - Why?

145

Gradient Descent

- If training records are not linearly separable, find best fit approximation.
  - Gradient descent to search the space of possible weight vectors
  - Basis for Backpropagation algorithm
- Consider un-thresholded perceptron (no sign function applied), i.e., u(**x**) = b + **w**·**x**
- Measure training error by squared error

$$E(b,\mathbf{w}) = \frac{1}{2}\sum_{(\mathbf{x},y)\in D}(y - u(\mathbf{x}))^2$$

  - D = training data

146

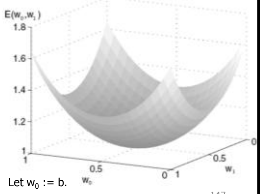
Gradient Descent Rule

- Find weight vector that minimizes E(b,**w**) by altering it in direction of steepest descent
  - Set (b,**w**) := (b,**w**) + Δ(b,**w**), where Δ(b,**w**) = -η ∇E(b,**w**)
    - -∇E(b,**w**)=[∂E/∂b, ∂E/∂w_1,…, ∂E/∂w_n] is the gradient, hence

$$b := b - \eta\frac{\partial E}{\partial b} = b - \eta\left(-\sum_{(\mathbf{x},y)\in D}(y - u(\mathbf{x}))\right)$$

$$w_i := w_i - \eta\frac{\partial E}{\partial w_i} = w_i - \eta\sum_{(\mathbf{x},y)\in D}(y - u(\mathbf{x}))(-x_i)$$

- Start with random weights, iterate until convergence
  - Will converge to global minimum if η is small enough



Let w_0 := b.

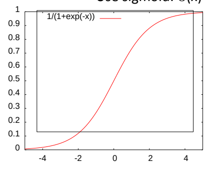
147

Gradient Descent Summary

- Epoch updating (aka batch mode)
  - Do until satisfied with model
    - Compute gradient over entire training set
    - Update all weights based on gradient
- Case updating (aka incremental mode, stochastic gradient descent)
  - Do until satisfied with model
    - For each training record
      - Compute gradient for this single training record
      - Update all weights based on gradient
- Case updating can approximate epoch updating arbitrarily close if η is small enough
- Perceptron training rule and case updating might seem identical
  - Difference: error computation on thresholded vs. unthresholded output

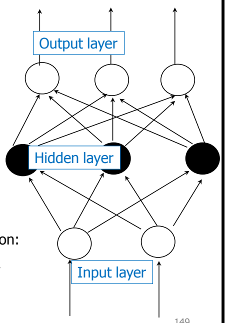
Multilayer Feedforward Networks

- Use another perceptron to combine output of lower layer
  - What about linear units only? Can only construct linear functions!
  - Need nonlinear component
    - sign function: not differentiable (gradient descent!)
    - Use sigmoid: $\sigma(x)=1/(1+e^{-x})$

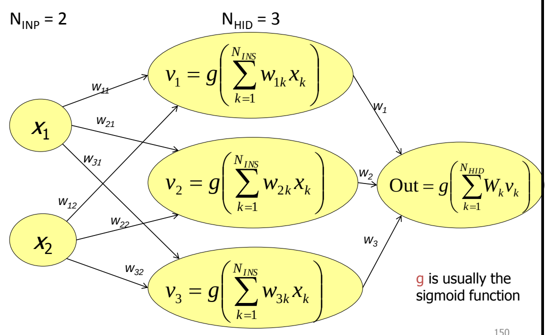


Perceptron function:

$$y=\frac{1}{1+e^{-b-\mathbf{w}\cdot\mathbf{x}}}$$



1-Hidden Layer Net Example


$N_{INP} = 2$
$N_{HID} = 3$
x_1
x_2
w_{11}
w_{21}
w_{31}
w_{12}
w_{22}
w_{32}
$v_1 = g\left(\sum_{k=1}^{N_{INS}} w_{1k}x_k\right)$
$v_2 = g\left(\sum_{k=1}^{N_{INS}} w_{2k}x_k\right)$
$v_3 = g\left(\sum_{k=1}^{N_{INS}} w_{3k}x_k\right)$
W_1
W_2
W_3
$\mathrm{Out} = g\left(\sum_{k=1}^{N_{HID}} W_k v_k\right)$

g is usually the sigmoid function

Making Predictions

- Inputs: all input data attributes
  - Record fed simultaneously into the units of the input layer
  - Then weighted and fed simultaneously to a hidden layer
    - Number of hidden layers is arbitrary, although usually only one
- Weighted outputs of the last hidden layer are the input to the units in the output layer, which emits the network's prediction
- The network is feed-forward
  - None of the weights cycles back to an input unit or to an output unit of a previous layer
- Statistical point of view: neural networks perform nonlinear regression

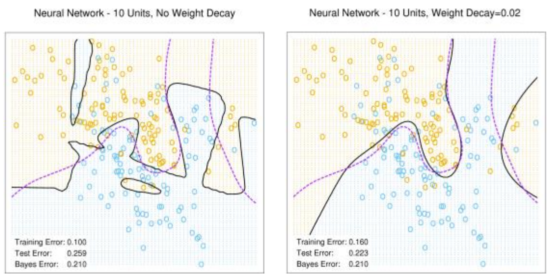
Backpropagation Algorithm

- We discussed gradient descent to find the best weights for a *single* perceptron using simple un-thresholded function
  - If sigmoid (or other differentiable) function is applied to weighted sum, use *complete function* for gradient descent
- Multiple perceptrons: optimize over all weights of all perceptrons
  - Problems: huge search space, local minima
- Backpropagation
  - Initialize all weights with small random values
  - Iterate many times
    - Compute gradient, starting at output and working back
      - Error of hidden unit h: how do we get the true output value? Use weighted sum of errors of each unit influenced by h.
    - Update all weights in the network

Overfitting

- When do we stop updating the weights?
  - Might overfit to training data
- Overfitting tends to happen in later iterations
  - Weights initially small random values
  - Weights all similar => smooth decision surface
  - Surface complexity increases as weights diverge
- Preventing overfitting
  - Weight decay: decrease each weight by small factor during each iteration, or
  - Use validation data to decide when to stop iterating

Neural Network Decision Boundary



Source: Hastie, Tibshirani, and Friedman. The Elements of Statistical Learning

154

Backpropagation Remarks

- Computational cost
  - Each interation costs O(|D|*|**w**|), with |D| training records and |**w**| weights
  - Number of iterations can be exponential in n, the number of inputs (in practice often tens of thousands)
- Local minima can trap the gradient descent algorithm
  - Convergence guaranteed to *local* minimum, not *global*
- Backpropagation highly effective in practice
  - Many variants to deal with local minima issue
  - E.g., case updating might avoid local minimum

155

Defining a Network

1. Decide network topology
   - # input units, # hidden layers, # units in each hidden layer, # output units
2. Normalize input values for each attribute to [0.0, 1.0]
   - Transform nominal and ordinal attributes: one input unit *per domain value*, each initialized to 0
   - Why not map the attribute to a single input with domain [0.0, 1.0]?
3. Output for classification task with >2 classes: one output unit per class
4. Choose learning rate η
   - Too small: can take days instead of minutes to converge
   - Too large: diverges (MSE gets larger while the weights increase and usually oscillate)
   - Heuristic: set it to 1 / (#training iterations)
5. If model accuracy is unacceptable, re-train with different network topology, different set of initial weights, or different learning rate
   - Might need a lot of trial-and-error

156

Representational Power

- Boolean functions
  - Each can be represented by a 2-layer network
  - Number of hidden units can grow exponentially with number of inputs
    - Create hidden unit for each input record
    - Set its weights to activate only for that input
    - Implement output unit as OR gate that only activates for desired output patterns
- Continuous functions
  - Every bounded continuous function can be approximated arbitrarily close by a 2-layer network
- Any function can be approximated arbitrarily close by a 3-layer network

157

Neural Network as a Classifier

- Weaknesses
  - Long training time
  - Many non-trivial parameters, e.g., network topology
  - Poor interpretability: What is the meaning behind learned weights and hidden units?
    - Note: hidden units are alternative representation of input values, capturing their relevant features
- Strengths
  - High tolerance to noisy data
  - Well-suited for continuous-valued inputs and outputs
  - Successful on a wide array of real-world data
  - Techniques exist for extraction of rules from neural networks

158

Classification and Prediction Overview

- Introduction
- Decision Trees
- Statistical Decision Theory
- Nearest Neighbor
- Bayesian Classification
- Artificial Neural Networks
- Support Vector Machines (SVMs)
- Prediction
- Accuracy and Error Measures
- Ensemble Methods

160

SVM—Support Vector Machines

- Newer and very popular classification method
- Uses a nonlinear mapping to transform the original training data into a higher dimension
- Searches for the optimal separating hyperplane (i.e., “decision boundary”) in the new dimension
- SVM finds this hyperplane using support vectors (“essential” training records) and margins (defined by the support vectors)

161

SVM—History and Applications

- Vapnik and colleagues (1992)
  - Groundwork from Vapnik & Chervonenkis’ statistical learning theory in 1960s
- Training can be slow but accuracy is high
  - Ability to model complex nonlinear decision boundaries (margin maximization)
- Used both for classification and prediction
- Applications: handwritten digit recognition, object recognition, speaker identification, benchmarking time-series prediction tests

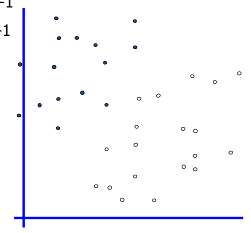
162

Linear Classifiers

• denotes +1

◦ denotes -1

$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}\cdot\mathbf{x} + b)$



How would you classify this data?

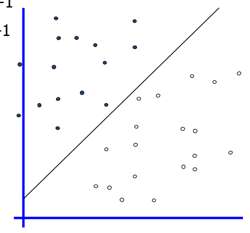
163

Linear Classifiers

• denotes +1

◦ denotes -1

$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}\cdot\mathbf{x} + b)$



How would you classify this data?

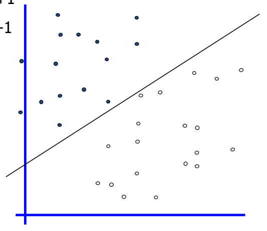
164

Linear Classifiers

• denotes +1

◦ denotes -1

$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}\cdot\mathbf{x} + b)$



How would you classify this data?

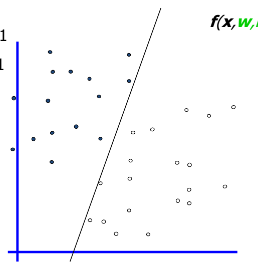
165

Linear Classifiers

• denotes +1

◦ denotes -1

$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}\cdot\mathbf{x} + b)$



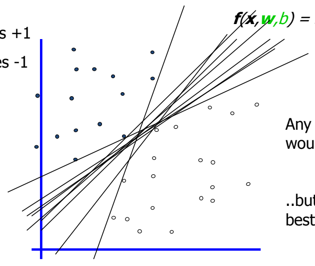
How would you classify this data?

166

Linear Classifiers

● denotes +1

◦ denotes -1



f(**x**,**w**,*b*) = *sign*(**w**·**x** + *b*)

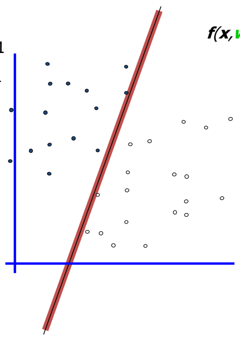
Any of these would be fine..

..but which is best?

Classifier Margin

● denotes +1

◦ denotes -1



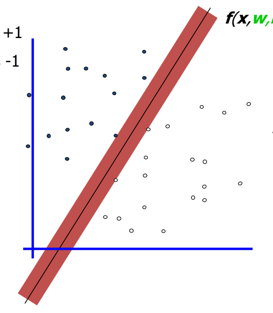
f(**x**,**w**,*b*) = *sign*(**w**·**x** + *b*)

Define the margin of a linear classifier as the width that the boundary could be increased by before hitting a data record.

Maximum Margin

● denotes +1

◦ denotes -1



f(**x**,**w**,*b*) = *sign*(**w**·**x** + *b*)

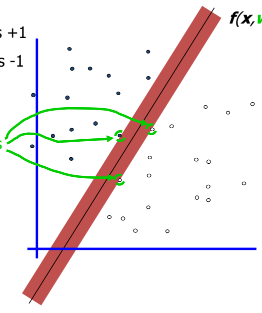
Find the maximum margin linear classifier.

This is the simplest kind of SVM, called linear SVM or LSVM.

Maximum Margin

● denotes +1

◦ denotes -1



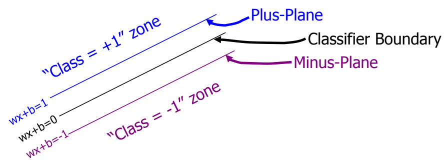
f(**x**,**w**,*b*) = *sign*(**w**·**x** + *b*)

Support Vectors are those datapoints that the margin pushes up against

Why Maximum Margin?

- If we made a small error in the location of the boundary, this gives us the least chance of causing a misclassification.
- Model is immune to removal of any non-support-vector data records.
- There is some theory (using VC dimension) that is related to (but not the same as) the proposition that this is a good thing.
- Empirically it works very well.

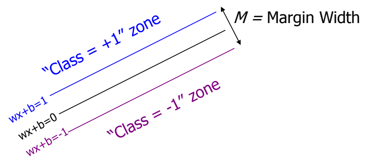
Specifying a Line and Margin



- Plus-plane = { **x** : **w**·**x** + b = +1 }
- Minus-plane = { **x** : **w**·**x** + b = -1 }

Classify as	+1	if	**w** ·**x** + b ≥ 1
	-1	if	**w**·**x** + b ≤ -1
	what	if	-1 < **w**·**x** + b < 1 ?

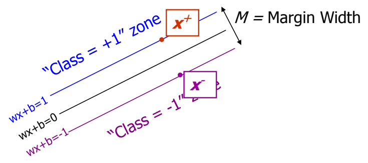
Computing Margin Width



- Plus-plane = { **x** : **w**·**x** + b = +1 }
- Minus-plane = { **x** : **w**·**x** + b = -1 }
- Goal: compute M in terms of **w** and b
  - Note: vector **w** is perpendicular to plus-plane
    - Consider two vectors **u** and **v** on plus-plane and show that **w**·(**u**-**v**)=0
    - Hence it is also perpendicular to the minus-plane

173

Computing Margin Width



- Choose arbitrary point $\mathbf{x}^-$ on minus-plane
- Let $\mathbf{x}^+$ be the point in plus-plane closest to $\mathbf{x}^-$
- Since vector **w** is perpendicular to these planes, it holds that $\mathbf{x}^+ = \mathbf{x}^- + \lambda\mathbf{w}$, for some value of λ

174

Putting It All Together

- We have so far:
  - $\mathbf{w}\cdot\mathbf{x}^+ + b = +1$ and $\mathbf{w}\cdot\mathbf{x}^- + b = -1$
  - $\mathbf{x}^+ = \mathbf{x}^- + \lambda\mathbf{w}$
  - $|\mathbf{x}^+ - \mathbf{x}^-| = M$
- Derivation:
  - $\mathbf{w}\cdot(\mathbf{x}^- + \lambda\mathbf{w}) + b = +1$, hence $\mathbf{w}\cdot\mathbf{x}^- + b + \mathbf{w}\lambda\mathbf{w} = 1$
  - This implies $\lambda\mathbf{w}\cdot\mathbf{w} = 2$, i.e., $\lambda = 2 / \mathbf{w}\cdot\mathbf{w}$
  - Since $M = |\mathbf{x}^+ - \mathbf{x}^-| = |\lambda\mathbf{w}| = \lambda\,|\mathbf{w}| = \lambda(\mathbf{w}\cdot\mathbf{w})^{0.5}$
  - We obtain $M = 2\,(\mathbf{w}\cdot\mathbf{w})^{0.5} / \mathbf{w}\cdot\mathbf{w} = 2 / (\mathbf{w}\cdot\mathbf{w})^{0.5}$

175

Finding the Maximum Margin

- How do we find **w** and b such that the margin is maximized and *all training records are in the correct zone for their class*?
- Solution: Quadratic Programming (QP)
- QP is a well-studied class of optimization algorithms to maximize a quadratic function of some real-valued variables subject to linear constraints.
  - There exist algorithms for finding such constrained quadratic optima efficiently and reliably.

176

Quadratic Programming

Find $\arg\max_{\mathbf{u}} \; c + \mathbf{d}^T\mathbf{u} + \dfrac{\mathbf{u}^T R\mathbf{u}}{2}$ ⟵ Quadratic criterion

Subject to

$$\left.\begin{array}{c} a_{11}u_1 + a_{12}u_2 + \ldots + a_{1m}u_m \le b_1 \\ a_{21}u_1 + a_{22}u_2 + \ldots + a_{2m}u_m \le b_2 \\ \vdots \\ a_{n1}u_1 + a_{n2}u_2 + \ldots + a_{nm}u_m \le b_n \end{array}\right\}$$

n additional linear inequality constraints

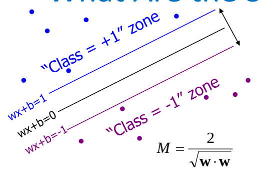
And subject to

$$\left.\begin{array}{c} a_{(n+1)1}u_1 + a_{(n+1)2}u_2 + \ldots + a_{(n+1)m}u_m = b_{(n+1)} \\ a_{(n+2)1}u_1 + a_{(n+2)2}u_2 + \ldots + a_{(n+2)m}u_m = b_{(n+2)} \\ \vdots \\ a_{(n+e)1}u_1 + a_{(n+e)2}u_2 + \ldots + a_{(n+e)m}u_m = b_{(n+e)} \end{array}\right\}$$

e additional linear equality constraints

177

What Are the SVM Constraints?

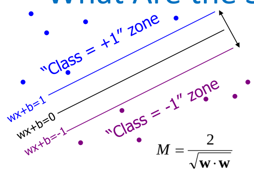


$$M = \frac{2}{\sqrt{\mathbf{w}\cdot\mathbf{w}}}$$

- What is the quadratic optimization criterion?
- Consider n training records (**x**(k), y(k)), where y(k) = +/- 1
- How many constraints will we have?
- What should they be?

178

What Are the SVM Constraints?



- What is the quadratic optimization criterion?
  - Minimize **w**·**w**
- Consider n training records (**x**(k), y(k)), where y(k) = +/- 1
- How many constraints will we have? n.
- What should they be?

For each 1 ≤ k ≤ n:

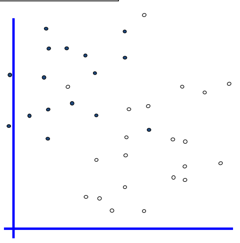
w·**x**(k) + b ≥ 1, if y(k)=1

w·**x**(k) + b ≤ -1, if y(k)=-1

179

Problem: Classes Not Linearly Separable

• denotes +1
◦ denotes -1

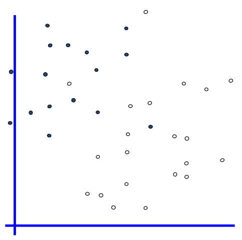


- Inequalities for training records are not satisfiable by any **w** and b

180

Solution 1?

• denotes +1
◦ denotes -1

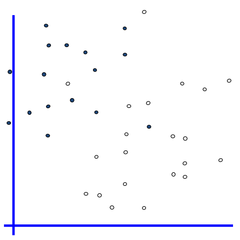


- Find minimum **w**·**w**, while also minimizing number of training set errors
  - Not a well-defined optimization problem (cannot optimize two things at the same time)

181

Solution 2?

• denotes +1
◦ denotes -1

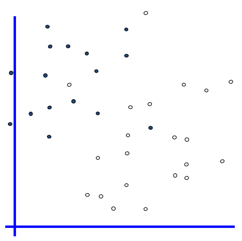


- Minimize **w**·**w** + C·(#trainSetErrors)
  - C is a tradeoff parameter
- Problems:
  - Cannot be expressed as QP, hence finding solution might be slow
  - Does not distinguish between disastrous errors and near misses

182

Solution 3

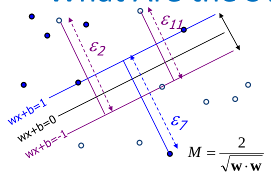
• denotes +1
◦ denotes -1



- Minimize **w**·**w** + C·(distance of error records to their correct place)
- This works!
- But still need to do something about the unsatisfiable set of inequalities

183

What Are the SVM Constraints?



- What is the quadratic optimization criterion?
  - Minimize

$$\frac{1}{2}\mathbf{w}\cdot\mathbf{w} + C\sum_{k=1}^{n}\varepsilon_k$$

- Consider n training records (**x**(k), y(k)), where y(k) = +/- 1
- How many constraints will we have? n.
- What should they be?

For each 1 ≤ k ≤ n:

w·**x**(k)+b ≥ 1 - ε_k, if y(k)=1

w·**x**(k)+b ≤ -1+ε_k, if y(k)=-1

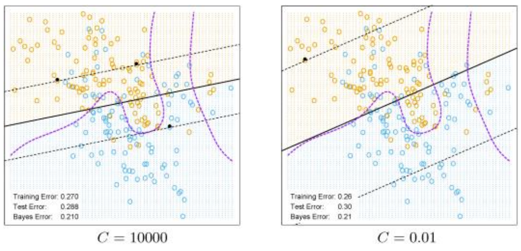
$\varepsilon_k \geq 0$

184

Facts About the New Problem Formulation

- Original QP formulation had d+1 variables
  - $w_1, w_2, \ldots, w_d$ and b
- New QP formulation has d+1+n variables
  - $w_1, w_2, \ldots, w_d$ and b
  - $\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n$
- C is a new parameter that needs to be set for the SVM
  - Controls tradeoff between paying attention to margin size versus misclassifications

Effect of Parameter C



Source: Hastie, Tibshirani, and Friedman. The Elements of Statistical Learning

An Equivalent QP (The "Dual")

Maximize $$\sum_{k=1}^{n}\alpha_k - \frac{1}{2}\sum_{k=1}^{n}\sum_{l=1}^{n}\alpha_k\alpha_l \cdot y(k)\cdot y(l)\cdot \mathbf{x}(k)\cdot\mathbf{x}(l)$$

Subject to these constraints: $$\forall k: 0 \le \alpha_k \le C \qquad \sum_{k=1}^{n}\alpha_k y(k) = 0$$

Then define:

$$\mathbf{w} = \sum_{k=1}^{n}\alpha_k \cdot y(k)\cdot\mathbf{x}(k)$$

$$b = \operatorname*{AVG}_{k:0<\alpha_k<C}\left\{\frac{1}{y(k)} - \mathbf{x}(k)\cdot\mathbf{w}\right\}$$

Then classify with:

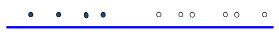
*f(**x**,**w**,b) = sign(**w**·**x** + b)*

Important Facts

- Dual formulation of QP can be optimized more quickly, but result is equivalent
- Data records with $\alpha_k > 0$ are the support vectors
  - Those with $0 < \alpha_k < C$ lie on the plus- or minus-plane
  - Those with $\alpha_k = C$ are on the wrong side of the classifier boundary (have $\varepsilon_k > 0$)
- Computation for **w** and b only depends on those records with $\alpha_k > 0$, i.e., the support vectors
- Alternative QP has another major advantage, as we will see now...

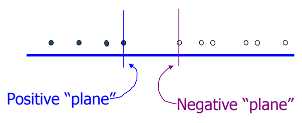
Easy To Separate

What would SVMs do with this data?



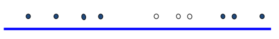
Easy To Separate

Not a big surprise



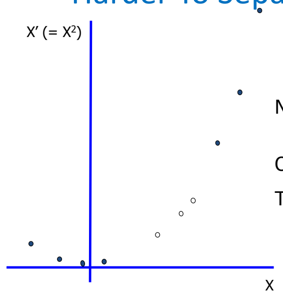
Harder To Separate

What can be done about this?



191

Harder To Separate



Non-linear basis functions:

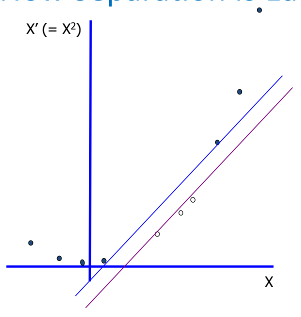
Original data: (X, Y)

Transformed: (X, X², Y)

Think of X² as a new attribute, e.g., X′

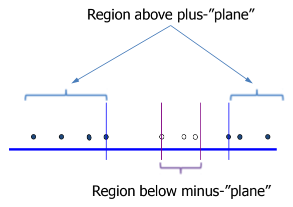
192

Now Separation Is Easy Again



193

Corresponding "Planes" in Original Space



194

Common SVM Basis Functions

- Polynomial of attributes $X_1,\ldots, X_d$ of certain max degree, e.g., $X_2+X_1X_3+X_4^2$
- Radial basis function
  - Symmetric around center, i.e., KernelFunction(|**X** - **c**| / kernelWidth)
- Sigmoid function of **X**, e.g., hyperbolic tangent
- Let $\Phi(\mathbf{x})$ be the transformed input record
  - Previous example: $\Phi((x)) = (x, x^2)$

195

Quadratic Basis Functions

$$\Phi(\mathbf{x}) = \left(\begin{array}{c} 1 \\ \sqrt{2}x_1 \\ \sqrt{2}x_2 \\ \vdots \\ \sqrt{2}x_d \\ x_1^2 \\ x_2^2 \\ \vdots \\ x_d^2 \\ \sqrt{2}x_1x_2 \\ \sqrt{2}x_1x_3 \\ \vdots \\ \sqrt{2}x_1x_d \\ \sqrt{2}x_2x_3 \\ \vdots \\ \sqrt{2}x_1x_d \\ \vdots \\ \sqrt{2}x_{d-1}x_d \end{array}\right)$$

Constant Term

Linear Terms

Pure Quadratic Terms

Quadratic Cross-Terms

Number of terms (assuming d input attributes):

(d+2)-choose-2

= (d+2)(d+1)/2

≈ $d^2/2$

Why did we choose this specific transformation?

196

Dual QP With Basis Functions

Maximize $\sum_{k=1}^{n} \alpha_k - \frac{1}{2} \sum_{k=1}^{n} \sum_{l=1}^{n} \alpha_k \alpha_l \cdot y(k) \cdot y(l) \cdot \mathbf{\Phi}(\mathbf{x}(k)) \cdot \mathbf{\Phi}(\mathbf{x}(l))$

Subject to these constraints: $\forall k : 0 \le \alpha_k \le C$ $\quad \sum_{k=1}^{n} \alpha_k y(k) = 0$

Then define:

$$\mathbf{w} = \sum_{k=1}^{n} \alpha_k \cdot y(k) \cdot \mathbf{\Phi}(\mathbf{x}(k))$$

$$b = \underset{k:0<\alpha_k<C}{\mathrm{AVG}} \left\{ \frac{1}{y(k)} - \mathbf{\Phi}(\mathbf{x}(k)) \cdot \mathbf{w} \right\}$$

Then classify with:

f(**x**,**w**,b) = sign(**w**·Φ(**x**) + b)

197

Computation Challenge

- Input vector **x** has d components (its d attribute values)
- The transformed input vector Φ(**x**) has $d^2/2$ components
- Hence computing Φ(**x**(k))·Φ(**x**(l)) now costs order $d^2/2$ instead of order d operations (additions, multiplications)
- ...or is there a better way to do this?
  - Take advantage of properties of certain transformations

198

Quadratic Dot Products

$$\mathbf{\Phi}(\mathbf{a}) \cdot \mathbf{\Phi}(\mathbf{b}) = \begin{pmatrix} 1 \\ \sqrt{2}a_1 \\ \sqrt{2}a_2 \\ : \\ \sqrt{2}a_d \\ a_1^2 \\ a_2^2 \\ : \\ a_d^2 \\ \sqrt{2}a_1a_2 \\ \sqrt{2}a_1a_3 \\ : \\ \sqrt{2}a_1a_d \\ \sqrt{2}a_2a_3 \\ : \\ \sqrt{2}a_1a_d \\ : \\ \sqrt{2}a_{d-1}a_d \end{pmatrix} \bullet \begin{pmatrix} 1 \\ \sqrt{2}b_1 \\ \sqrt{2}b_2 \\ : \\ \sqrt{2}b_d \\ b_1^2 \\ b_2^2 \\ : \\ b_d^2 \\ \sqrt{2}b_1b_2 \\ \sqrt{2}b_1b_3 \\ : \\ \sqrt{2}b_1b_d \\ \sqrt{2}b_2b_3 \\ : \\ \sqrt{2}b_1b_d \\ : \\ \sqrt{2}b_{d-1}b_d \end{pmatrix}$$

$$1 + \sum_{i=1}^{m} 2a_ib_i + \sum_{i=1}^{m} a_i^2 b_i^2 + \sum_{i=1}^{m} \sum_{j=i+1}^{m} 2a_ia_jb_ib_j$$

199

Quadratic Dot Products

$$\mathbf{\Phi}(\mathbf{a}) \cdot \mathbf{\Phi}(\mathbf{b}) = 1 + 2\sum_{i=1}^{d} a_ib_i + \sum_{i=1}^{d} a_i^2 b_i^2 + \sum_{i=1}^{d} \sum_{j=i+1}^{d} 2a_ia_jb_ib_j$$

Now consider another function of **a** and **b**:

$$(\mathbf{a} \cdot \mathbf{b} + 1)^2$$

$$= (\mathbf{a} \cdot \mathbf{b})^2 + 2\mathbf{a} \cdot \mathbf{b} + 1$$

$$= \left(\sum_{i=1}^{d} a_ib_i \right)^2 + 2\sum_{i=1}^{d} a_ib_i + 1$$

$$= \sum_{i=1}^{d} \sum_{j=1}^{d} a_ib_ia_jb_j + 2\sum_{i=1}^{d} a_ib_i + 1$$

$$= \sum_{i=1}^{d} (a_ib_i)^2 + 2\sum_{i=1}^{d} \sum_{j=i+1}^{d} a_ib_ia_jb_j + 2\sum_{i=1}^{d} a_ib_i + 1$$

200

Quadratic Dot Products

- The results of Φ(**a**)·Φ(**b**) and of $(\mathbf{a}\cdot\mathbf{b}+1)^2$ are identical
- Computing Φ(**a**)·Φ(**b**) costs about $d^2/2$, while computing $(\mathbf{a}\cdot\mathbf{b}+1)^2$ costs only about d+2 operations
- This means that we can work in the high-dimensional space ($d^2/2$ dimensions) where the training records are more easily separable, but pay about the same cost as working in the original space (d dimensions)
- Savings are even greater when dealing with higher-degree polynomials, i.e., degree q>2, that can be computed as $(\mathbf{a}\cdot\mathbf{b}+1)^q$

201

Any Other Computation Problems?

$$\mathbf{w} = \sum_{k=1}^{n} \alpha_k \cdot y(k) \cdot \mathbf{\Phi}(\mathbf{x}(k)) \qquad b = \underset{k:0<\alpha_k<C}{\mathrm{AVG}} \left\{ \frac{1}{y(k)} - \mathbf{\Phi}(\mathbf{x}(k)) \cdot \mathbf{w} \right\}$$

- What about computing w?
  - Finally need **f**(**x**,**w**,b) = sign(**w**·Φ(**x**) + b):

$$\mathbf{w} \cdot \mathbf{\Phi}(\mathbf{x}) = \sum_{k=1}^{n} \alpha_k \cdot y(k) \cdot \mathbf{\Phi}(\mathbf{x}(k)) \cdot \mathbf{\Phi}(\mathbf{x})$$

  - Can be computed using the same trick as before
- Can apply the same trick again to b, because

$$\mathbf{\Phi}(\mathbf{x}(k)) \cdot \mathbf{w} = \sum_{j=1}^{n} \alpha_j \cdot y(j) \cdot \mathbf{\Phi}(\mathbf{x}(k)) \cdot \mathbf{\Phi}(\mathbf{x}(j))$$

202

SVM Kernel Functions

- For which transformations, called kernels, does the same trick work?
- Polynomial: K(**a**,**b**)=(**a** · **b** +1)^q
- Radial-Basis-style (RBF):

$$K(\mathbf{a},\mathbf{b}) = \exp\left(-\frac{(\mathbf{a}-\mathbf{b})^2}{2\sigma^2}\right)$$

σ, κ and δ are magic parameters that must be chosen by a model selection method.

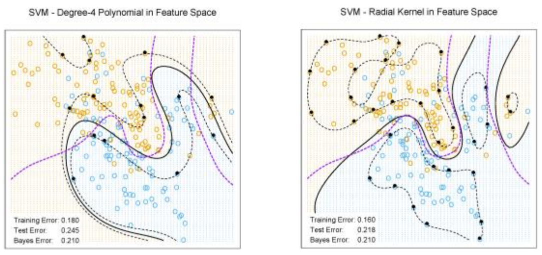
  – Neural-net-style sigmoidal:

$$K(\mathbf{a},\mathbf{b}) = \tanh(\kappa \cdot \mathbf{a} \cdot \mathbf{b} - \delta)$$

Overfitting

- With the right kernel function, computation in high dimensional transformed space is no problem
- But what about overfitting? There are so many parameters...
- Usually not a problem, due to maximum margin approach
  - Only the support vectors determine the model, hence SVM complexity depends on number of support vectors, not dimensions (still, in higher dimensions there might be more support vectors)
  - Minimizing **w**·**w** discourages extremely large weights, which smoothes the function (recall weight decay for neural networks!)

Different Kernels



Source: Hastie, Tibshirani, and Friedman. The Elements of Statistical Learning

Multi-Class Classification

- SVMs can only handle two-class outputs (i.e. a categorical output variable with arity 2).
- What can be done?
- Answer: with output arity N, learn N SVM's
  - SVM 1 learns "Output==1" vs "Output != 1"
  - SVM 2 learns "Output==2" vs "Output != 2"
  - :
  - SVM N learns "Output==N" vs "Output != N"
- To predict the output for a new input, just predict with each SVM and find out which one puts the prediction the furthest into the positive region.

Why Is SVM Effective on High Dimensional Data?

- Complexity of trained classifier is characterized by the number of support vectors, not dimensionality of the data
- If all other training records are removed and training is repeated, the same separating hyperplane would be found
- The number of support vectors can be used to compute an upper bound on the expected error rate of the SVM, which is independent of data dimensionality
- Thus, an SVM with a small number of support vectors can have good generalization, even when the dimensionality of the data is high

SVM vs. Neural Network

- SVM
  - Relatively new concept
  - Deterministic algorithm
  - Nice Generalization properties
  - Hard to train – learned in batch mode using quadratic programming techniques
  - Using kernels can learn very complex functions
- Neural Network
  - Relatively old
  - Nondeterministic algorithm
  - Generalizes well but doesn't have strong mathematical foundation
  - Can easily be learned in incremental fashion
  - To learn complex functions—use multilayer perceptron (not that trivial)

Classification and Prediction Overview

- Introduction
- Decision Trees
- Statistical Decision Theory
- Nearest Neighbor
- Bayesian Classification
- Artificial Neural Networks
- Support Vector Machines (SVMs)
- Prediction
- Accuracy and Error Measures
- Ensemble Methods

210

What Is Prediction?

- Essentially the same as classification, but output is continuous, not discrete
  - Construct a model
  - Use model to predict continuous output value for a given input
- Major method for prediction: regression
  - Many variants of regression analysis in statistics literature; not covered in this class
- Neural network and k-NN can do regression "out-of-the-box"
- SVMs for regression exist
- What about trees?

211

Regression Trees and Model Trees

- Regression tree: proposed in CART system (Breiman et al. 1984)
  - CART: Classification And Regression Trees
  - Each leaf stores a continuous-valued prediction
    - Average output value for the training records that reach the leaf
- Model tree: proposed by Quinlan (1992)
  - Each leaf holds a regression model—a multivariate linear equation
- Training: like for classification trees, but uses *variance* instead of purity measure for selecting split predicates

212

Classification and Prediction Overview

- Introduction
- Decision Trees
- Statistical Decision Theory
- Nearest Neighbor
- Bayesian Classification
- Artificial Neural Networks
- Support Vector Machines (SVMs)
- Prediction
- Accuracy and Error Measures
- Ensemble Methods

213

Classifier Accuracy Measures

		Predicted class		total
		buy_computer = yes	buy_computer = no	
True class	buy_computer = yes	6954	46	7000
	buy_computer = no	412	2588	3000
total		7366	2634	10000

- Accuracy of a classifier M, acc(M): percentage of test records that are correctly classified by M
  - Error rate (misclassification rate) of M = 1 – acc(M)
  - Given m classes, CM[i,j], an entry in a confusion matrix, indicates # of records in class i that are labeled by the classifier as class j

	C_1	C_2
C_1	True positive	False negative
C_2	False positive	True negative

214

Precision and Recall

- Precision: measure of exactness
  - t-pos / (t-pos + f-pos)
- Recall: measure of completeness
  - t-pos / (t-pos + f-neg)
- F-measure: combination of precision and recall
  - 2 * precision * recall / (precision + recall)
- Note: Accuracy = (t-pos + t-neg) / (t-pos + t-neg + f-pos + f-neg)

215

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
- Always predicting the majority class defines the baseline
  - A good classifier should do better than baseline

Cost-Sensitive Measures: Cost Matrix

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

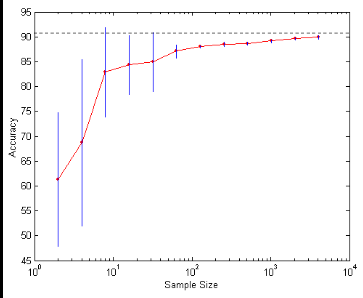
Prediction Error Measures

- Continuous output: it matters how far off the prediction is from the true value
- Loss function: distance between y and predicted value y’
  - Absolute error: $| y – y’|$
  - Squared error: $(y – y’)^2$
- Test error (generalization error): average loss over the test set
- Mean absolute error: $\frac{1}{n}\sum_{i=1}^{n}|y(i)-y'(i)|$ Mean squared error: $\frac{1}{n}\sum_{i=1}^{n}\left(y(i)-y'(i)\right)^2$
- Relative absolute error: $\frac{\sum_{i=1}^{n}|y(i)-y'(i)|}{\sum_{i=1}^{n}|y(i)-\bar{y}|}$ Relative squared error: $\frac{\sum_{i=1}^{n}(y(i)-y'(i))^2}{\sum_{i=1}^{n}(y(i)-\bar{y})^2}$
- Squared-error exaggerates the presence of outliers

Evaluating a Classifier or Predictor

- Holdout method
  - The given data set is randomly partitioned into two sets
    - Training set (e.g., 2/3) for model construction
    - Test set (e.g., 1/3) for accuracy estimation
  - Can repeat holdout multiple times
    - Accuracy = avg. of the accuracies obtained
- Cross-validation (k-fold, where k = 10 is most popular)
  - Randomly partition data into k mutually exclusive subsets, each approximately equal size
  - In i-th iteration, use D_i as test set and others as training set
  - Leave-one-out: k folds where k = # of records
    - Expensive, often results in high variance of performance metric

Learning Curve



- Accuracy versus sample size
- Effect of small sample size:
  - Bias in estimate
  - Variance of estimate
- Helps determine how much training data is needed
  - Still need to have enough test and validation data to be representative of distribution

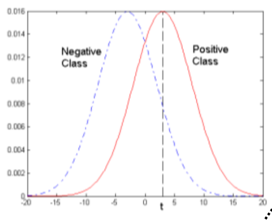
ROC (Receiver Operating Characteristic)

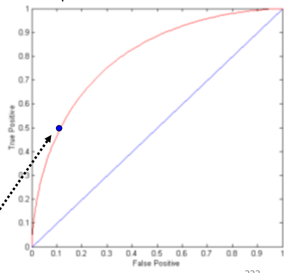
- Developed in 1950s for signal detection theory to analyze noisy signals
  - Characterizes trade-off between positive hits and false alarms
- ROC curve plots T-Pos rate (y-axis) against F-Pos rate (x-axis)
- Performance of each classifier is represented as a point on the ROC curve
  - Changing the threshold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

222

ROC Curve

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





At threshold t:

TPR=0.5, FPR=0.12

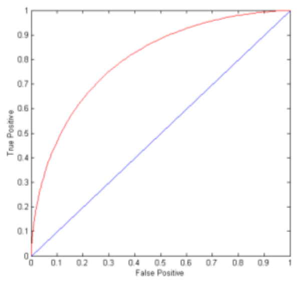
223

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



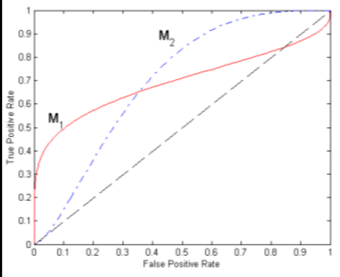
224

Diagonal Line for Random Guessing

- Classify a record as positive with fixed probability p, irrespective of attribute values
- Consider test set with *a* positive and *b* negative records
- True positives: p*a, hence true positive rate = (p*a)/a = p
- False positives: p*b, hence false positive rate = (p*b)/b = p
- For every value $0 \le p \le 1$, we get point (p,p) on ROC curve

225

Using ROC for Model Comparison



- Neither model consistently outperforms the other
  - M1 better for small FPR
  - M2 better for large FPR
- Area under the ROC curve
  - Ideal: area = 1
  - Random guess: area = 0.5

226

How to Construct an ROC curve

record	P(+|**x**)	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	+

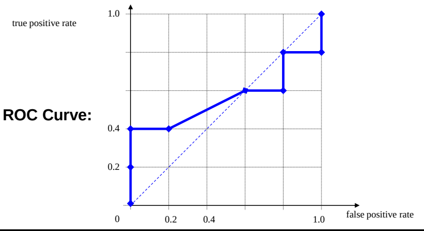
- Use classifier that produces posterior probability P(+|**x**) for each test record **x**
- Sort records according to P(+|**x**) in decreasing order
- Apply threshold at each unique value of P(+|**x**)
  - Count number of TP, FP, TN, FN at each threshold
  - TP rate, TPR = TP/(TP+FN)
  - FP rate, FPR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

227

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			2	2	1	0
	FP	5	5	4	4	3			1	0	0	0
	TN	0	0	1	1	2			4	5	5	5
	FN	0	1	1	2	2			3	3	4	5
$\longrightarrow$	TPR	1	0.8	0.8	0.6	0.6			0.4	0.4	0.2	0
$\longrightarrow$	FPR	1	1	0.8	0.8	0.6			0.2	0	0	0

ROC Curve:



228

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 accuracy be explained as a result of random fluctuations in the test set?

229

Confidence Interval for Accuracy

- Classification can be regarded as a Bernoulli trial
  - A Bernoulli trial has 2 possible outcomes, "correct" or "wrong" for classification
  - Collection of Bernoulli trials has a Binomial distribution
    - Probability of getting c correct predictions if model accuracy is p (=probability to get a single prediction right):

$$\binom{n}{c} p^c (1-p)^{n-c}$$

- Given c, or equivalently, ACC = c / n and n (#test records), can we predict p, the true accuracy of the model?

230

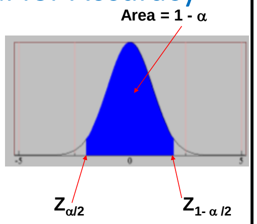
Confidence Interval for Accuracy

- Binomial distribution for X="number of correctly classified test records out of n"
  - E(X)=pn, Var(X)=p(1-p)n
- Accuracy = X / n
  - E(ACC) = p, Var(ACC) = p(1-p) / n
- For large test sets (n>30), Binomial distribution is closely approximated by normal distribution with same mean and variance
  - ACC has a normal distribution with mean=p, variance=p(1-p)/n

$$P\left(Z_{\alpha/2} < \frac{\text{ACC} - p}{\sqrt{p(1-p)/n}} < Z_{1-\alpha/2} \right) = 1 - \alpha$$

- Confidence Interval for p:

$$p = \frac{2n \cdot \text{ACC} + Z^2_{\alpha/2} \pm \sqrt{Z^2_{\alpha/2} + 4n \cdot \text{ACC} - 4n \cdot \text{ACC}^2}}{2(n + Z^2_{\alpha/2})}$$



231

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-α = 0.95 (95% confidence)
  - From probability table, $Z_{\alpha/2}$ = 1.96

1-α	Z
0.99	2.58
0.98	2.33
0.95	1.96
0.90	1.65

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

$$p = \frac{2n \cdot \text{ACC} + Z^2_{\alpha/2} \pm \sqrt{Z^2_{\alpha/2} + 4n \cdot \text{ACC} - 4n \cdot \text{ACC}^2}}{2(n + Z^2_{\alpha/2})}$$

232

Comparing Performance of Two Models

- Given two models M1 and M2, which is better?
  - M1 is tested on D_1 (size=n_1), found error rate = e_1
  - M2 is tested on D_2 (size=n_2), found error rate = e_2
  - Assume D_1 and D_2 are independent
  - If n_1 and n_2 are sufficiently large, then

$$\text{err}_1 \sim N(\mu_1, \sigma_1)$$
$$\text{err}_2 \sim N(\mu_2, \sigma_2)$$

  - Estimate: $\hat{\mu}_i = e_i$ and $\hat{\sigma}_i^2 = \dfrac{e_i(1-e_i)}{n_i}$

233

Testing Significance of Accuracy Difference

- Consider random variable d = err$_1$– err$_2$
  - Since err$_1$, err$_2$ are normally distributed, so is their difference
  - Hence d ~ N (d_t, σ_t) where d_t is the true difference
- Estimator for d_t:
  - E[d] = E[err$_1$-err$_2$] = E[err$_1$] – E[err$_2$] $\approx$ e$_1$ - e$_2$
  - Since D$_1$ and D$_2$ are independent, variance adds up:

$$\hat{\sigma}_t^2 = \hat{\sigma}_1^2 + \hat{\sigma}_2^2 = \frac{e_1(1-e_1)}{n_1} + \frac{e_2(1-e_2)}{n_2}$$

  - At (1-α) confidence level, $d_t = \mathrm{E}[d] \pm Z_{\alpha/2}\hat{\sigma}_t$

234

An Illustrative Example

- Given: M1: n_1 = 30, e_1 = 0.15
  M2: n_2 = 5000, e_2 = 0.25
- E[d] = |e$_1$ – e$_2$| = 0.1
- 2-sided test: d_t = 0 versus $d_t \neq 0$

$$\hat{\sigma}_t^2 = \frac{0.15(1-0.15)}{30} + \frac{0.25(1-0.25)}{5000} = 0.0043$$

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

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

- Interval contains zero, hence difference may not be statistically significant
- But: may reject null hypothesis ($d_t \neq 0$) at lower confidence level

235

Significance Test for K-Fold Cross-Validation

- Each learning algorithm produces k models:
  - L1 produces M11 , M12, …, M1k
  - L2 produces M21 , M22, …, M2k
- Both models are tested on the same test sets D$_1$, D$_2$,…, D$_k$
  - For each test set, compute $d_j = e_{1,j} – e_{2,j}$
  - For large enough k, d_j is normally distributed with mean d_t and variance σ_t
  - Estimate:

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

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

t-distribution: get t coefficient $t_{1-\alpha,k-1}$ from table by looking up confidence level (1-α) and degrees of freedom (k-1)

236

Classification and Prediction Overview

- Introduction
- Decision Trees
- Statistical Decision Theory
- Nearest Neighbor
- Bayesian Classification
- Artificial Neural Networks
- Support Vector Machines (SVMs)
- Prediction
- Accuracy and Error Measures
- Ensemble Methods

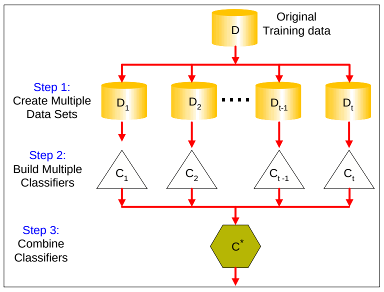
237

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

238

General Idea


D
Original Training data
Step 1: Create Multiple Data Sets
D$_1$
D$_2$
D$_{t-1}$
D$_t$
Step 2: Build Multiple Classifiers
C$_1$
C$_2$
C$_{t-1}$
C$_t$
Step 3: Combine Classifiers
C*

239

Why Does It Work?

- Consider 2-class problem
- Suppose there are 25 base classifiers
  - Each classifier has error rate ε = 0.35
  - Assume the classifiers are independent
- Return majority vote of the 25 classifiers
  - Probability that the ensemble classifier makes a wrong prediction:

$$\sum_{i=13}^{25} \binom{25}{i} \varepsilon^i (1-\varepsilon)^{25-i} = 0.06$$

240

Base Classifier vs. Ensemble Error

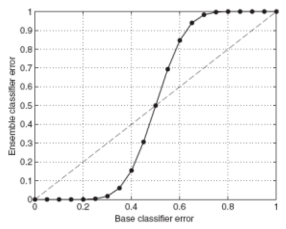


Figure 5.30. Comparison between errors of base classifiers and errors of the ensemble classifier.

241

Model Averaging and Bias-Variance Tradeoff

- Single model: lowering bias will usually increase variance
  - "Smoother" model has lower variance but might not model function well enough
- Ensembles can overcome this problem
  1. Let models overfit
     - Low bias, high variance
  2. Take care of the variance problem by averaging many of these models
- This is the basic idea behind bagging

242

Bagging: Bootstrap Aggregation

- Given training set with n records, sample n records randomly 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

- Train classifier for each bootstrap sample
- Note: each training record has probability $1 - (1 - 1/n)^n$ of being selected at least once in a sample of size n

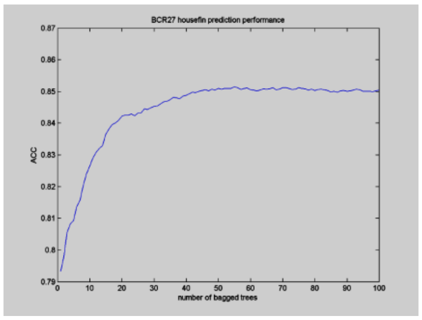
243

Bagged Trees

- Create k trees from training data
  - Bootstrap sample, grow large trees
- Design goal: independent models, high variability between models
- Ensemble prediction = average of individual tree predictions (or majority vote)
- Works the same way for other classifiers

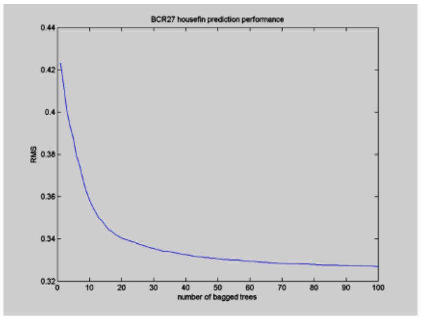
244

Typical Result

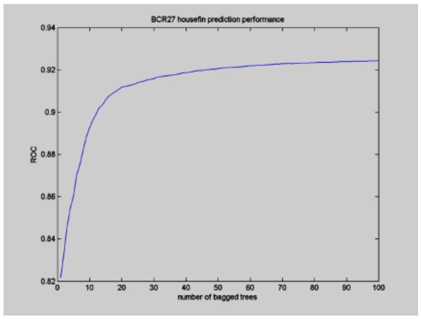


245

Typical Result



Typical Result

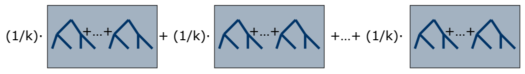


Bagging Challenges

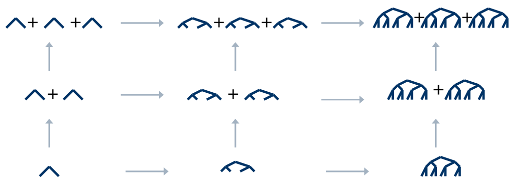
- Ideal case: all models independent of each other
- Train on independent data samples
  - Problem: limited amount of training data
    - Training set needs to be representative of data distribution
  - Bootstrap sampling allows creation of many “almost” independent training sets
- Diversify models, because similar sample might result in similar tree
  - Random Forest: limit choice of split attributes to small random subset of attributes (new selection of subset for each node) when training tree
  - Use different model types in same ensemble: tree, ANN, SVM, regression models

Additive Grove

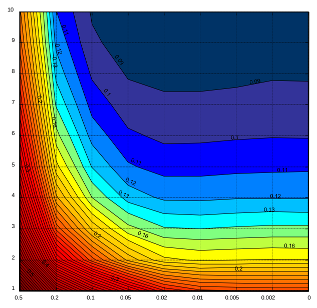
- Ensemble technique for predicting continuous output
- Instead of individual trees, train additive models
  - Prediction of single Grove model = sum of tree predictions
- Prediction of ensemble = average of individual Grove predictions
- Combines large trees and additive models
  - Challenge: how to train the additive models without having the first trees fit the training data too well
    - Next tree is trained on residuals of previously trained trees in same Grove model
    - If previously trained trees capture training data too well, next tree is mostly trained on noise



Training Groves

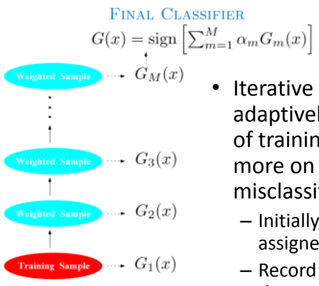


Typical Grove Performance



- Root mean squared error
  - Lower is better
- Horizontal axis: tree size
  - Fraction of training data when to stop splitting
- Vertical axis: number of trees in each single Grove model
- 100 bagging iterations

Boosting



- Iterative procedure to adaptively change distribution of training data by focusing more on previously misclassified records
  - Initially, all n records are assigned equal weights
  - Record weights may change at the end of each 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

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

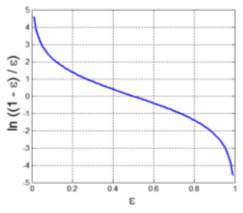
Example: AdaBoost

- Base classifiers: $C_1, C_2, \ldots, C_T$
- Error rate (n training records, w_j are weights that sum to 1):

$$\varepsilon_i = \sum_{j=1}^{n} w_j \delta\left(C_i(x_j) \neq y_j\right)$$

- Importance of a classifier:

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



AdaBoost Details

- Weight update:

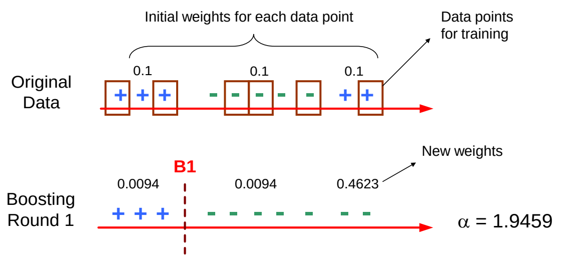
$$w_j^{(i+1)} = \frac{w_j^{(i)}}{Z_i} \cdot \begin{cases} \frac{\varepsilon_i}{1-\varepsilon_i} & \text{if } C_i(x_j) = y_j \\ 1 & \text{if } C_i(x_j) \neq y_j \end{cases}$$

where Z_i is the normalization factor

- Weights initialized to 1/n
- Z_i ensures that weights add to 1
- If any intermediate rounds produce error rate higher than 50%, the weights are reverted back to 1/n and the resampling procedure is repeated
- Final classification:

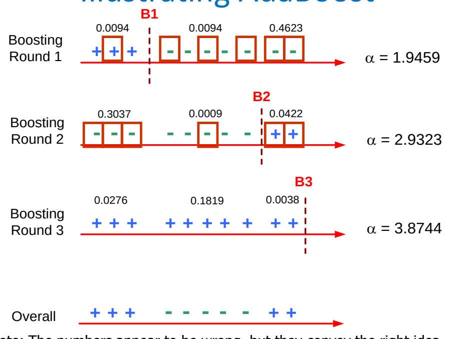
$$C^*(x) = \arg\max_{y} \sum_{i=1}^{T} \alpha_i \delta\left(C_i(x) = y\right)$$

Illustrating AdaBoost



Note: The numbers appear to be wrong, but they convey the right idea…

Illustrating AdaBoost



Note: The numbers appear to be wrong, but they convey the right idea…

Bagging vs. Boosting

- Analogy
  - Bagging: diagnosis based on multiple doctors' majority vote
  - Boosting: weighted vote, based on doctors' previous diagnosis accuracy
- Sampling procedure
  - Bagging: records have same weight; easy to train in parallel
  - Boosting: weights record higher if model predicts it wrong; inherently sequential process
- Overfitting
  - Bagging robust against overfitting
  - Boosting susceptible to overfitting: make sure individual models do not overfit
- Accuracy usually significantly better than a single classifier
  - Best boosted model often better than best bagged model
- Additive Grove
  - Combines strengths of bagging and boosting (additive models)
  - Shown empirically to make better predictions on many data sets
  - Training more tricky, especially when data is very noisy

258

Classification/Prediction Summary

- Forms of data analysis that can be used to train models from data and then make predictions for new records
- Effective and scalable methods have been developed for decision tree induction, Naive Bayesian classification, Bayesian networks, rule-based classifiers, Backpropagation, Support Vector Machines (SVM), nearest neighbor classifiers, and many other classification methods
- Regression models are popular for prediction. Regression trees, model trees, and ANNs are also used for prediction.

259

Classification/Prediction Summary

- K-fold cross-validation is a popular method for accuracy estimation, but determining accuracy on large test set is equally accepted
  - If test sets are large enough, a significance test for finding the best model is not necessary
- Area under ROC curve and many other common performance measures exist
- Ensemble methods like bagging and boosting can be used to increase overall accuracy by learning and combining a series of individual models
  - Often state-of-the-art in prediction quality, but expensive to train, store, use
- No single method is superior over all others for all data sets
  - Issues such as accuracy, training and prediction time, robustness, interpretability, and scalability must be considered and can involve trade-offs

260