Classification

CPS 296.3: Information Management and Mining

Jun Yang

Duke University

February 17, 2008

† Thanks to contents borrowed from Han (http://www.cs.uiuc.edu/homes/hanj/bk2/slidesindex.html), and Kumar (http://www-users.cs.umn.edu/~kumar/dmbook/)

Announcements

- Reading assignment for next week
  - Chu, Kim, Lin, Yu, Bradski, Ng, and Olukotun. "Map-Reduce for Machine Learning on Multicore." *NIPS* 19, 2007
  - Das, Datar, Garg, Rajaram. "Google News Personalization: Scalable Online Collaborative Filtering." *WWW* 2007
  - Posted on the course Web site
    - See the Web site for reviewing and submission instructions
  - Reviews due by Monday at noon
  - 2 students needed to lead discussion
- Schedule for the remainder of this semester
  - Ideas from only three of you so far
  - I need more!

Classification: definition

- Given a collection of records, each containing a set of attributes, one of which is the class
- Find a model for the class attribute as a function of the values of the other attributes
- Goal: previously unseen records should be assigned a class ("labeled") as accurately as possible
  - Often, we divide given data into two sets
    - Training set is for building the model
    - Test set is for validating the model

➲Supervised learning
  - Compare with clustering, which is unsupervised

Classification: examples

- Classifying credit card transactions as legitimate or fraudulent
- Determine whether an email is spam
- Categorizing news stories as finance, weather, entertainment, sports, etc.
- Predicting tumor cells as benign or malignant

Classification: techniques

- Naïve Bayes and Bayesian Belief Networks
  - Very briefly
- Decision trees
  - More details
- Rule-based
- Case-based
- Neural networks
- Support Vector Machines
- Etc.

Basics

- Conditional probabilities:
  $P(C \mid A) = P(A \cap C) / P(A)$
  $P(A \mid C) = P(A \cap C) / P(C)$
- Bayes' Theorem:
  $P(C \mid A) = P(A \mid C)\, P(C) / P(A)$
  - Example
    - Meningitis causes stiff neck 50% of the time
    - Prior prob. of anyone having meningitis is 1/50,000
    - Prior prob. of anyone having stiff neck is 1/20
    - Given stiff neck, what's the prob. of meningitis?
    - $P(M \mid S) = P(S \mid M)\, P(M) / P(S)$
      $= 0.5 \times 1/50{,}000 / (1/20) = 0.0002$

Bayesian classifiers

- Consider attributes/class as random variables
- Given a record with attribute values (a_1, …, a_n), predict class c
  - I.e., find the label c that maximizes $P(c \mid a_1, …, a_n)$, which also provides a measure of credibility

Approach:

- By Bayes' Theorem: $P(C \mid A_1, …, A_n) = P(A_1, …, A_n \mid C)\, P(C) / P(A_1, …, A_n)$
- Choose value of C that maximizes $P(C \mid a_1, …, a_n)$ ⇔ choose value of C that maximizes
  $P(a_1, …, a_n \mid C)\, P(C)$

Naïve Bayes classifier

- Assume independence among A_i's given C
  - $P(A_1, …, A_n \mid C) = \prod_i P(A_i \mid C)$
- Estimate $P(A_i \mid C)$ from training data
- Estimate $P(C)$ from training data
- New point is labeled c if $P(c) \prod_i P(a_i \mid c)$ is maximal

Estimation from data

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

- Class: $P(c) = N_c / N$
  - P(Cheat=Yes) = 3/10
- Categorical attributes: $P(a \mid c) = N_{a,c} / N_c$
  - P(Status=Married|Cheat=No) = 4 / 7
  - P(Refund=Yes|Cheat=Yes) = 0 / 3 = 0
- Continuous attributes
  - Assume $P(A \mid c)$ follows some distribution (e.g., normal)
  - Use data to estimate its parameters
  - Plug in a to get probability density

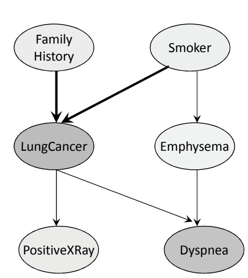
Coping with zeros

- If one conditional probability (density) is zero, then the posterior becomes zero!
  - Easy when training data is sparse
- Original: $P(a \mid c) = N_{a,c} / N_c$
- Laplacian correction: $P(a \mid c) = (N_{a,c} + 1) / (N_c$ + # of possible labels)
- m-estimate: $P(a \mid c) = (N_{a,c} + mp) / (N_c + m)$
  - p: can be regarded as a prior probability
    - Above = p if $N_{a,c} = N_c = 0$
  - m: controls trade-off between prior and observed

Bayesian Belief Networks

- The independence assumption of Naïve Bayes was too strong
- ➲BBN: a graphical model that allows some dependencies (and conditional independencies) to be captured
  - Gives a specification of joint probability distribution from the graph structure and conditional probability tables (CPTs)

BBN example



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

Shows the conditional probability for each possible combination of its parents

Probability of a particular combination of values (x_1, …, x_n):

$P(x_1, …, x_n) = \prod_i P(x_i \mid \text{parents}(x_i))$

Training BBNs

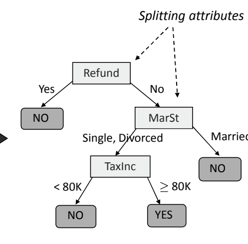
- Network structure is known, and all variables are observable: just compute the CPTs
- Structure known, some variables hidden: find CPTs that best model the data
- Structure unknown, all variables observable: need to search through the space of possible structures
- Structure unknown, some variables hidden: hard

Decision tree: example

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



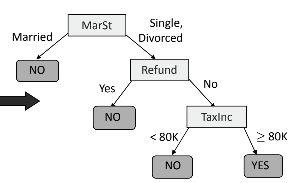
Model: Decision Tree

Another example

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



There can be many trees that “fit” the same data!

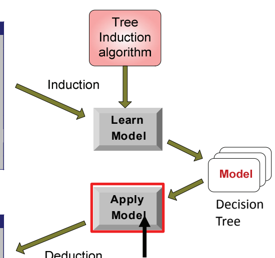
Decision tree classification

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

Training Set

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

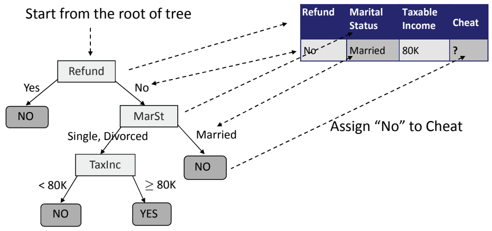
Test Set



Applying model to test data

Start from the root of tree

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



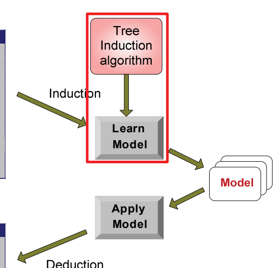
Tree induction

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

Training Set

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

Test Set



19

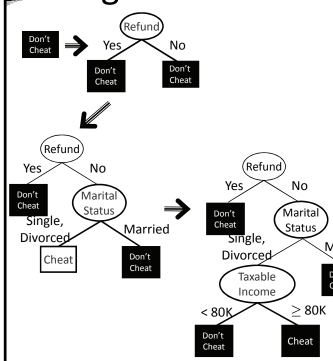
A top-down algorithm

- Initially, all training records are at the root
- Let D_t be the set of training records at a node t
- If records in D_t belong to more than one class, pick an attribute test to split D_t into subsets, each as a child of t, and recursively apply the procedure
  - However, if there are no more attributes for partitioning, make t a leaf labeled by majority voting
    - The most popular class in D_t
- If all records in D_t belong to the same class y_t, make t a leaf labeled y_t

➲What if D_t is empty?

20

Algorithm in action



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

21

A closer look

- Greedy strategy: split records based on an attribute test that optimizes some criterion now

➲What exactly is an "attribute test"?

➲How do you defined the "best" split?

22

Attribute test

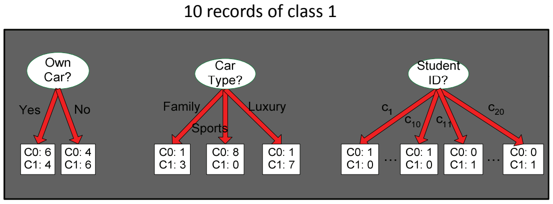
- Depends on attribute types (nominal, ordinal, continuous) and on fan-out (2-way vs. multi-way)
  - Multi-way: as many partitions as # of distinct values
  - 2-way: divide values into two subsets; need to find optimal partitioning
- A continuous attribute can be discretized into an ordinal attribute
  - Equal-interval bucketing, equal-frequency bucketing, clustering, or consider all possible discretizations and find the best
    - E.g., ($A < v$) or ($A \geq v$), considering all possible v's

23

What is the best split?

Before splitting:

10 records of class 0,
10 records of class 1



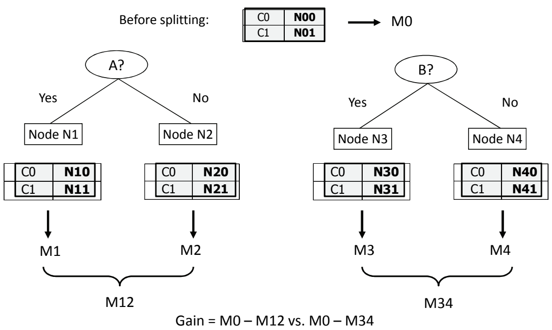
24

Measure of node impurity

- Intuition: nodes with homogeneous class distributions are preferred
- Need a measure of impurity:

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

Comparing splits



Measure of impurity: GINI

- GINI index for a given node t:
  GINI(t) = 1 – $\sum_j [p(j \mid t)]^2$
  - $p(j \mid t)$ is the relative freq. of class j at node t
- Maximum = $(1 - 1/n_c)$, when records are equally distributed among all classes—least interesting
- Minimum = 0, when all records belong to one class—most interesting

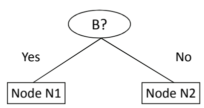
C1	0	C1	1	C1	2	C1	3
C2	6	C2	5	C2	4	C2	3
Gini=0.000		Gini=0.278		Gini=0.444		Gini=0.500	

P(C1) = 1/(1+5) = 1/6, P(C2) = 5/(1+5) = 5/6
GINI = 1 – P(C1)² – P(C2)² = 1 – (1/6)² – (5/6)² ≈ 0.278

GINI for a split

- When a node t is split into k partitions (children) t_1, …, t_k, GINI of the split is given by:
  GINI($\{t_1, \ldots, t_k\}$) = $\sum_i (n_i/n)$ GINI(t_i),
  - n_i = # of records at t_i; n = # of records at t
- Example (binary attribute)

	Parent
C1	6
C2	6
Gini = 0.500	



	N1	N2
C1	5	1
C2	2	4
Gini=0.347		

GINI(N1)
= 1 – (5/7)² – (2/7)²
= 0.408

GINI(N2)
= 1 – (1/5)² – (4/5)²
= 0.320

GINI(Children)
= 7/12×0.408 + 5/12×0.320
= 0.347

Comparing splits with GINI

	CarType		
	Family	Sports	Luxury
C1	1	2	1
C2	4	1	1
Gini	0.393		

	CarType	
	{Sports, Luxury}	{Family}
C1	3	1
C2	2	4
Gini	0.400	

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

And compare with GINI for splits on other attributes…

- To compare, need to compute, for each attribute, and for each possible partitioning
  - Total # of records for every (partition, class)
    - I.e., the "count matrix"

Scaling up GINI computation

- Suppose we want to find the best binary attribute test for a continuous attribute A, with m values in D_t
  - I.e., optimal v for ($A < v$) vs. ($A \geq v$)
- How many possibilities are there?
  - $m - 1$
- How do we do it fast, when D_t doesn't fit in memory?
  - If ($m \times$ # of classes) counters can fit in memory, easy
  - Otherwise
    - Sort D_t by A to get total # by A (and by class)
    - In one pass, with v increasing
      - Accumulate total # for ($A < v$, class)
        - Total # for ($A \geq v$, class) can be found by subtraction
      - Compute GINI and remember the smallest

Alternative measure: entropy

- Entropy at a given node t:
  Entropy(t) = $-\sum_j p(j \mid t) \log p(j \mid t)$
  - Again, $p(j \mid t)$ = relative freq. of class j at node t
- Maximum = $\log n_c$, when records are equally distributed among all classes—least information
- Minimum = 0, when all records belong to one class—most information

C1	0	C1	1	C1	2
C2	6	C2	5	C2	4
Ent. =0.000		Ent. =0.65		Ent. =0.92	

P(C1) = 1/(1+5) = 1/6, P(C2) = 5/(1+5) = 5/6
Entropy = – (1/6) log(1/6) – (5/6) log(5/6) ≈ 0.65

Information gain of a split

- When a node t is split into k partitions (children) $t_1, \ldots, t_k$, the quality of the split is given by: Gain($\{t_1, \ldots, t_k\}$) = Entropy(t) – $\sum_i (n_i/n)$Entropy(t_i),
  - n_i = # of records at t_i; n = # of records at t
- Disadvantage: tends to prefer splits that result in a large # of partitions, each being small but pure

A fix—gain ratio

- GainRatio($\{t_1, \ldots, t_k\}$) = Gain($\{t_1, \ldots, t_k\}$) / SplitInfo($\{t_1, \ldots, t_k\}$)
- SplitInfo($\{t_1, \ldots, t_k\}$) = – $\sum_j (n_j/n) \log (n_j/n)$
  - I.e., entropy of the partitioning itself

➲Higher-entropy partitioning (large number of small partitions) is thus penalized

More on scalability: SPRINT

SPRINT [Shafer, Agrawal, Mehta, VLDB '96]

- Start with attribute lists, one for each attribute
  - (value, rid, label), sorted by value
  - Supports efficient evaluation of splitting criteria
- Say we split using attribute A
- For every other attribute B
  - Partition B's attribute list among children, conceptually as a hash join between A and B's attribute lists on rid
    - Care is taken to ensure that the ordering within B's attribute list is preserved

RainForest

[Gehrke, Ramakrishnan, Ganti, VLDB '98]

- For SPRINT, many cost terms are linear in $|D_t|$
- But the only info we need is a node's AVC-group
  - Collection of AVC-sets, one for each attribute
  - AVC-set for an attribute captures the class distribution by this attribute
    - (value, label, # of records in D_t with value and label)
    - Size = (# of distinct values) × (# of distinct labels)
    - ➲Perhaps not that big
    - ➲A node's AVC-group is no larger in size than parent's AVC-group except the splitting attribute's AVC-set

RainForest: RF-Write

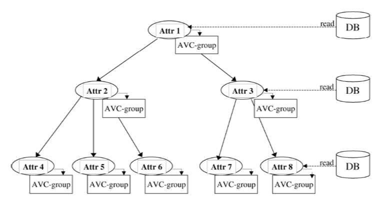
- Scan db and create AVC-group in memory
- Use AVC-group to decide on splitting
- Scan db again and create one partition per child
  - No need to output the splitting attribute
- Now, continue recursively for each child

➲The root's AVC-group (the largest) must fit in memory

➲2 full db reads and 1 full db write for each level of the decision tree

RainForest: RF-Read

- Start just like RF-Write, but instead of writing sub-partitions, read the db, construct AVC-groups for children in memory
- Continue recursively, breadth first



➲1 full db scan for each level

RainForest: RF-Read (cont'd)

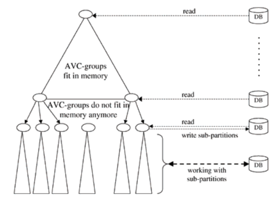
- What if there is not enough memory to keep AVC-groups on a level?
  - Process a subset of these groups (that fit in memory) at a time
  - Will need a new db scan for each such subset

➲Not good since # of groups per level grows exponentially as tree grows deeper

RainForest: RF-Hybrid

Combination of RF-Read and RF-Write

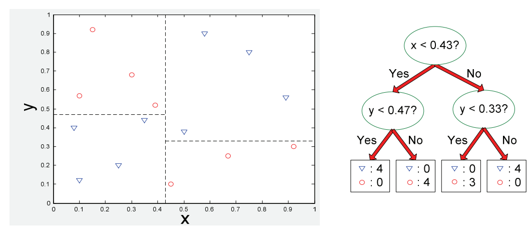
- Start as RF-Read until there is not enough memory to hold all AVC-groups in a level
- At this point, switch to RF-Write
- Continue recursively
  - RF-Read* RF-Write
    RF-Read* RF-Write…



Expressiveness

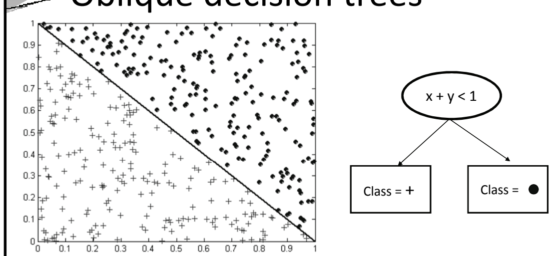
- Decision tree provides expressive representation for learning discrete-valued function
  - But they do not generalize well to certain types of functions
    - Example: parity function
      - Class = 0 if there are even number of Boolean attributes with true values, or 1 otherwise
- Decision tree also is not expressive enough for continuous variables, particularly because test condition involves one attribute at a time

Decision boundary



- Decision boundaries are parallel to axes because each test condition involves only one attribute

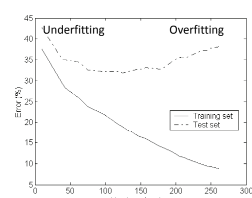
Oblique decision trees



- Allow one test condition to involve multiple attributes
  - More expressive
  - More expensive to find the optimal test condition

Underfitting and overfitting

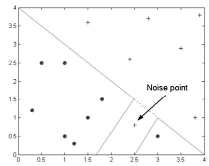
- Underfitting
  - Model too simple
  - High training and test errors



- Overfitting
  - Model too complex
    ≈ memorizing all training data!
  - Low training error, but poor generalization

Possible causes of overfitting

- Decision boundary distorted by noise



- Too little data, too many model parameters
  - With enough # of model parameters, you are bound to find some setting that happens to work

Rethinking errors

- Re-substitution errors (on training data): $\sum e(t)$
  - $e(t)$ = # of errors in leaf t
  - Does not provide a good estimate of how well the tree will perform on previously unseen records!
- Generalization errors (on unseen data): $\sum e'(t)$
  - Optimistic estimation: $e'(t) = e(t)$
  - Pessimistic estimation: $e'(t) = e(t) + \Omega(t)$
    - Per-leaf penalty $\Omega(t)$ accounts for model complexity
    - E.g., say $\Omega(t) = 0.5$; for a tree with 30 leaves and 10 errors on 1000 training records, training error % = 10/1000 = 1%, while generalization error % = $(10 + 0.5 \times 30)/1000$ = 2.5%
  - Use validation data (a subset of training data reserved for validation) to estimate

Occam's razor

- Given two models of similar generalization errors, we should prefer the simpler model over the more complex ones
  - Complex models tend to be more specific/brittle
- Therefore, include model complexity in evaluation
- Think of it as compressing the class labels
  - Cost = Cost(model) + Cost(data|model)
    - Measure cost by # of bits needed for encoding
    - Cost(data|model) is for encoding misclassification errors

➲Minimum Description Length (MDL) Principle: search for the model that offers the lowest overall cost

Pre-prune to avoid overfitting

- Pre-pruning (early stopping): stop the algorithm before the tree becomes fully grown
- Typical stopping conditions for a node
  - Stop if all instances belong to the same class
  - Stop if all non-class attribute values are the same
- More restrictive conditions
  - Stop if # of instances is too small
  - Stop if class distribution of instances are independent of the available features (e.g., using χ^2 test)
  - Stop if splitting the current node does not reduce impurity measures (e.g., GINI)

Post-prune to avoid overfitting

- Post-pruning
  - Grow decision tree to its entirety
  - Trim the nodes in a bottom-up fashion
  - If generalization error improves after trimming (i.e., replacing a subtree by a leaf), do so
    - Or use MDL
  - Label leaf nodes by majority voting

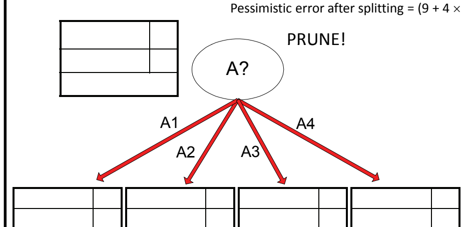
Example of post-pruning

Training error before splitting = 10/30

Pessimistic error before splitting = (10 + 0.5)/30 = 10.5/30

Training error after splitting = 9/30

Pessimistic error after splitting = $(9 + 4 \times 0.5)/30$ = 11/30



Model evaluation

- What are the right metrics for evaluating the "performance" of a model?
- How do we obtain reliable estimates of model performance?

Confusion matrix

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

	PREDICTED CLASS		
ACTUAL CLASS			

TP (true positive)
FN (false negative)
FP (false positive)
TN (true negative)

Accuracy

- Most widely used metric:
  Accuracy = (TP + TN) / (TP + TN + FP + FN)

Limitation?

- Consider a 2-class problem
  - Number of class-0 examples: 9990
  - Number of class-1 examples: 10
- Model predicts everything to be class 0
  - Accuracy = 9990/10000 = 99.9%
    - Misleading because model does not detect any class-1 examples

Cost matrix

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

C(i|j): cost of classifying class j example as class i

Computing cost

Cost Matrix	PREDICTED CLASS		
ACTUAL CLASS	C(i|j)	+	–
	+		
	–		

Model M_1	PREDICTED CLASS		
ACTUAL CLASS		+	–
	+		
	–		

Accuracy = 80%
Cost = 3910

Model M_2	PREDICTED CLASS		
ACTUAL CLASS		+	–
	+		
	–		

Accuracy = 90%
Cost = 4255

Cost vs. accuracy

- Accuracy is linearly related to cost if
  - C(Yes|Yes) = C(No|No) = a
  - C(Yes|No) = C(No|Yes) = b
- Accuracy = (TP+TN)/(TP+TN+FP+FN)
- Cost = a(TP+TN) + b(FP+FN)
  = a(TP+TN) + b(TP+TN+FP+FN)–b(TP+TN)
  = (a – b)(TP+TN) + b(TP+TN+FP+FN)
  = (TP+TN+FP+FN) ((a – b) × Accuracy + b)

55

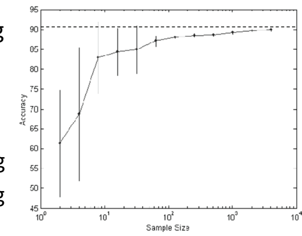
Cost-sensitive measures

- Precision (p) = TP / (TP+FP)
- Recall (r) = TP / (TP+FN)
- F-measure (F) = $2rp/(r+p)$ = 2TP / (2TP + FP + FN)
  - Harmonic mean of r and p
- Weighted accuracy = (w_{TP}TP + w_{TN}TN)/(w_{TP}TP + w_{TN}TN + w_{FP}FP + w_{FN}FN)
  - Precision: w_{TP} = w_{FP} = 1; others 0
  - Recall: w_{TP} = w_{FN} = 1; others 0
  - F-measure: w_{TP} = 2; w_{FP} = w_{FN} = 1; others 0
  - Accuracy: all 1

56

Learning curve

- Shows how accuracy changes with varying sample size
- Requires a sampling schedule
  - Arithmetic sampling
  - Geometric sampling



57

ROC

Receiver Operating Characteristic

- Developed in the 1950's for signal detection theory to characterize the trade-off between positive hits and false alarms
- Plot TP rate (sensitivity), TP/(TP+FN), on y-axis
- Plot FP rate (1 – specificity), FP/(TN+FP), on x-axis
- Performance of a classifier $\rightarrow$ point on ROC curve
  - A decision tree $\rightarrow$ one point
  - A Bayesian classifier, which produces a probability $\rightarrow$ one point for each threshold setting $\rightarrow$ curve

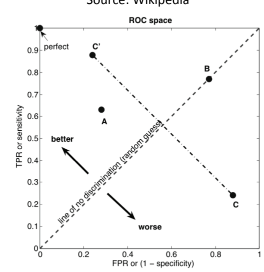
58

ROC space

Source: Wikipedia

(TPR, FPR)

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



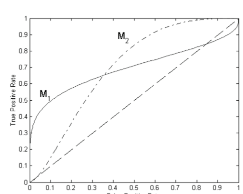
Diagonal line: random guessing

Below diagonal line: prediction opposite of true class

59

Using ROC to compare models

- Neither model consistently outperforms the other
  - M_1 is better for small FPR
  - M_2 is better for large TPR



- AUC: area under curve
  - Ideal: AUC = 1
  - Random guess: AUC = 0.5

60

Methods of estimation – 1/2

- Holdout
  - Reserve a fraction of labeled data for training (e.g., 2/3) and the rest for testing
- Random subsampling
  - Repeated holdout; take average
- Cross validation
  - Partition data into k disjoint subsets
  - k-fold: train on k – 1 partitions, test on the last; sum up # of errors
    - Leave-one-out: k = # of records

Methods of estimation – 2/2

- Bootstrap
  - To get a bootstrap sample, sample N times from N labeled records, with replacement
    - When N is large, $\approx 1 - e^{-1}$ = 63.2% will be selected
    - Use sample to train, and remaining records to test
  - .632 bootstrap
    - Use many bootstrap samples, take the average accuracy, weigh it by 63.2%, and add (1 – 63.2%) of the re-substitution accuracy (using all labeled records to train/test)