Data Mining
Classification: Alternative Techniques

Lecture Notes for Chapter 5

Introduction to Data Mining
by
Tan, Steinbach, Kumar

Rule-Based Classifier

- Classify records by using a collection of "if…then…" rules
- Rule: (*Condition*) → *y*
  - where
    - *Condition* is a conjunction of attributes
    - *y* is the class label
  - *LHS*: rule antecedent or condition
  - *RHS*: rule consequent
  - Examples of classification rules:
    - (Blood Type=Warm) ∧ (Lay Eggs=Yes) → Birds
    - (Taxable Income < 50K) ∧ (Refund=Yes) → Evade=No

Rule-based Classifier (Example)

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

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

Application of Rule-Based Classifier

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

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

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

The rule R1 covers a hawk => Bird
The rule R3 covers the grizzly bear => Mammal

Rule Coverage and Accuracy

- Coverage of a rule:
  - Fraction of records that satisfy the antecedent of a rule
- Accuracy of a rule:
  - Fraction of records that satisfy both the antecedent and consequent of a rule

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

(Status=Single) → No

Coverage = 40%, Accuracy = 50%

How does Rule-based Classifier Work?

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

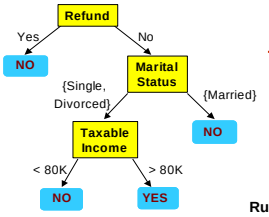
Name	Blood Type	Give Birth	Can Fly	Live in Water	Class
lemur	warm	yes	no	no	?
turtle	cold	no	no	sometimes	?
dogfish shark	cold	yes	no	yes	?

A lemur triggers rule R3, so it is classified as a mammal
A turtle triggers both R4 and R5
A dogfish shark triggers none of the rules

Characteristics of Rule-Based Classifier

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

From Decision Trees To Rules



Classification Rules

(Refund=Yes) ==> No

(Refund=No, Marital Status={Single,Divorced}, Taxable Income<80K) ==> No

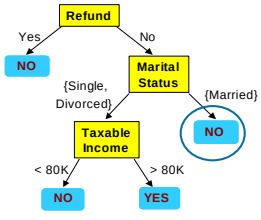
(Refund=No, Marital Status={Single,Divorced}, Taxable Income>80K) ==> Yes

(Refund=No, Marital Status={Married}) ==> No

Rules are mutually exclusive and exhaustive

Rule set contains as much information as the tree

Rules Can Be Simplified



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

Initial Rule: **(Refund=No) ∧ (Status=Married) → No**

Simplified Rule: **(Status=Married) → No**

Effect of Rule Simplification

- Rules are no longer mutually exclusive
  - A record may trigger more than one rule
  - Solution?
    - Ordered rule set
    - Unordered rule set – use voting schemes
- Rules are no longer exhaustive
  - A record may not trigger any rules
  - Solution?
    - Use a default class

Ordered Rule Set

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

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

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

Building Classification Rules

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

Advantages of Rule-Based Classifiers

- As highly expressive as decision trees
- Easy to interpret
- Easy to generate
- Can classify new instances rapidly
- Performance comparable to decision trees

Associative classification

- The classification model is defined by means of association rules

  $(Condition) \rightarrow y$

  - rule body is an itemset
- Model generation
  - Rule selection & sorting
    - based on support, confidence and correlation thresholds
  - Rule pruning
    - Database coverage: the training set is covered by selecting topmost rules according to previous sort

Associative classification

- Strong points
  - interpretable model
  - higher accuracy than decision trees
    - correlation among attributes is considered
  - efficient classification
  - unaffected by missing data
  - good scalability in the training set size
- Weak points
  - rule generation may be slow
    - it depends on support threshold
  - reduced scalability in the number of attributes
    - rule generation may become unfeasible

Bayes theorem

- Let C and X be random variables

  $P(C,X) = P(C|X) P(X)$

  $P(C,X) = P(X|C) P(C)$
- Hence

  $P(C|X) P(X) = P(X|C) P(C)$
- and also

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

Bayesian classification

- Let the class attribute and all data attributes be random variables
  - C = any class label
  - $X = <x_1,\ldots,x_k>$ record to be classified
- Bayesian classification
  - compute $P(C|X)$ for all classes
    - probability that record X belongs to C
  - assign X to the class with *maximal* $P(C|X)$
- Applying Bayes theorem

  $P(C|X) = P(X|C)\cdot P(C) / P(X)$
  - $P(X)$ constant for all C, disregarded for maximum computation
  - $P(C)$ a priori probability of C

    $P(C) = N_c/N$

Bayesian classification

- How to estimate $P(X|C)$, i.e. $P(x_1,\ldots,x_k|C)$?
- Naïve hypothesis

  $P(x_1,\ldots,x_k|C) = P(x_1|C)\, P(x_2|C) \ldots P(x_k|C)$
  - *statistical independence* of attributes $x_1,\ldots,x_k$
  - not always true
    - model quality may be affected
- Computing $P(x_k|C)$
  - for discrete attributes

    $P(x_k|C) = |x_{kC}|/ N_c$
    - where $|x_{kC}|$ is number of instances having value x_k for attribute k and belonging to class C
  - for continuous attributes, use probability distribution
- Bayesian networks
  - allow specifying a subset of dependencies among attributes

Bayesian classification: Example

Outlook	Temperature	Humidity	Windy	Class
sunny	hot	high	false	N
sunny	hot	high	true	N
overcast	hot	high	false	P
rain	mild	high	false	P
rain	cool	normal	false	P
rain	cool	normal	true	N
overcast	cool	normal	true	P
sunny	mild	high	false	N
sunny	cool	normal	false	P
rain	mild	normal	false	P
sunny	mild	normal	true	P
overcast	mild	high	true	P
overcast	hot	normal	false	P
rain	mild	high	true	N

From Han, Kamber,"Data mining; Concepts and Techniques", Morgan Kaufmann 2006

Bayesian classification: Example

outlook	
P(sunny|p) = 2/9	**P(sunny|n) = 3/5**
P(overcast|p) = 4/9	**P(overcast|n) = 0**
P(rain|p) = 3/9	**P(rain|n) = 2/5**
temperature	
P(hot|p) = 2/9	**P(hot|n) = 2/5**
P(mild|p) = 4/9	**P(mild|n) = 2/5**
P(cool|p) = 3/9	**P(cool|n) = 1/5**
humidity	
P(high|p) = 3/9	**P(high|n) = 4/5**
P(normal|p) = 6/9	**P(normal|n) = 2/5**
windy	
P(true|p) = 3/9	**P(true|n) = 3/5**
P(false|p) = 6/9	**P(false|n) = 2/5**

P(p) = 9/14
P(n) = 5/14

From Han, Kamber,"Data mining; Concepts and Techniques", Morgan Kaufmann 2006

Bayesian classification: Example

- Data to be labeled

  X = <rain, hot, high, false>

- For class p

  $P(X|p)\cdot P(p) =$
  $= P(rain|p)\cdot P(hot|p)\cdot P(high|p)\cdot P(false|p)\cdot P(p)$
  $= 3/9\cdot 2/9\cdot 3/9\cdot 6/9\cdot 9/14 = 0.010582$

- For class n

  $P(X|n)\cdot P(n) =$
  $= P(rain|n)\cdot P(hot|n)\cdot P(high|n)\cdot P(false|n)\cdot P(n)$
  $= 2/5\cdot 2/5\cdot 4/5\cdot 2/5\cdot 5/14 = 0.018286$

From Han, Kamber,"Data mining; Concepts and Techniques", Morgan Kaufmann 2006

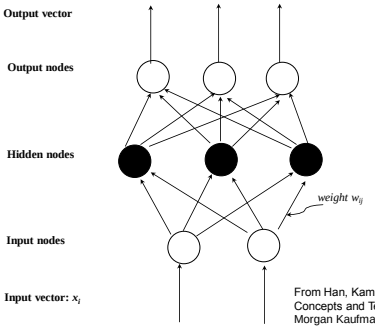
Bayesian classification

- Strong points
  - efficient classification
  - medium model interpretability
  - robust to isolated noise points and irrelevant attributes
  - incremental model update
- Weak points
  - model generation without naïve hypothesis is unfeasible
  - naïve hypothesis may significantly reduce accuracy

Neural networks

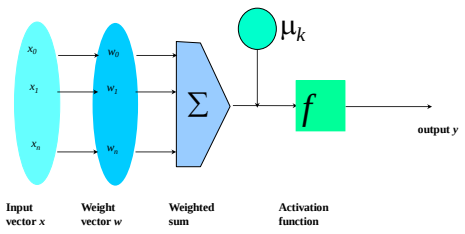
- Inspired to the structure of the human brain
  - Neurons as elaboration units
  - Synapses as connection network

Structure of a neural network



From Han, Kamber,"Data mining; Concepts and Techniques", Morgan Kaufmann 2006

Structure of a neuron



From Han, Kamber,"Data mining; Concepts and Techniques", Morgan Kaufmann 2006

Construction of the Neural Network

- For each node, definition of
  - set of weights
  - offset value

  providing the highest accuracy on the training data
- Iterative approach on training data instances

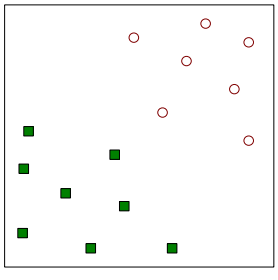
Construction of the Neural Network

- Base algorithm
  - Initially assign random values to weights and offsets
  - Process instances in the training set one at a time
    - For each neuron, compute the result when applying weights, offset and activation function for the instance
    - Forward propagation until the output is computed
    - Compare the computed output with the expected output, and evaluate error
    - Backpropagation of the error, by updating weights and offset for each neuron
  - The process ends when
    - % of accuracy above a given threshold
    - % of parameter variation (error) below a given threshold
    - The maximum number of epochs is reached

Neural Networks

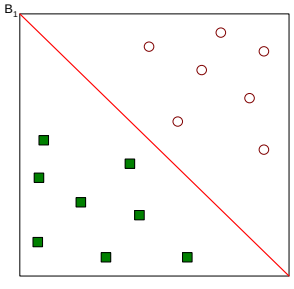
- Strong points
  - High accuracy
  - Robust to noise and outliers
  - Supports both discrete and continuous output
  - Efficient during classification
- Weak points
  - Long training time
    - weakly scalable in training data size
  - Not interpretable model
    - Application domain knowledge cannot be exploited in the model

Support Vector Machines



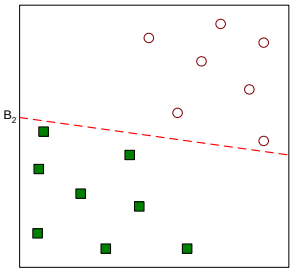
- Find a linear hyperplane (decision boundary) that will separate the data

Support Vector Machines



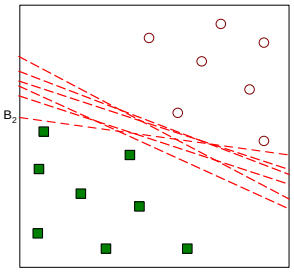
- One Possible Solution

Support Vector Machines



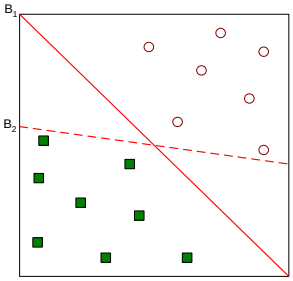
- Another possible solution

Support Vector Machines



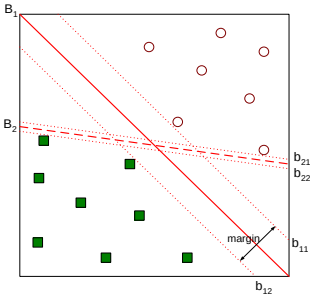
- Other possible solutions

Support Vector Machines



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

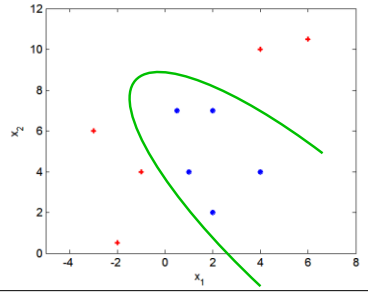
Support Vector Machines



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

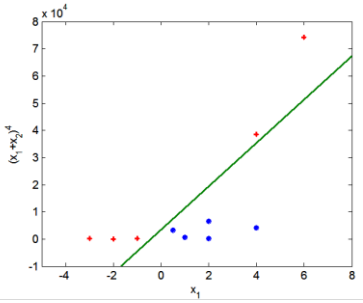
Nonlinear Support Vector Machines

- What if decision boundary is not linear?



Nonlinear Support Vector Machines

- Transform data into higher dimensional space



Instance-Based Classifiers

Set of Stored Cases

Atr1	.....	AtrN	Class
			A
			B
			B
			C
			A
			C
			B

- **Store the training records**
- **Use training records to predict the class label of unseen cases**

Unseen Case

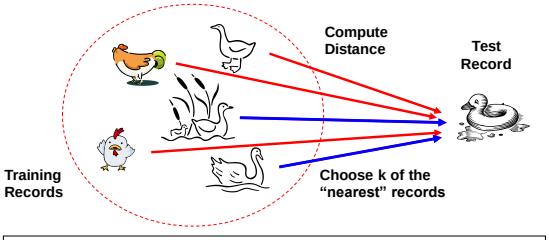
Atr1	.....	AtrN

Instance Based Classifiers

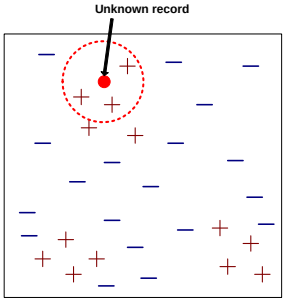
- Examples:
  - Rote-learner
    - Memorizes entire training data and performs classification only if attributes of record match one of the training examples exactly
  - Nearest neighbor
    - Uses k "closest" points (nearest neighbors) for performing classification

Nearest Neighbor Classifiers

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

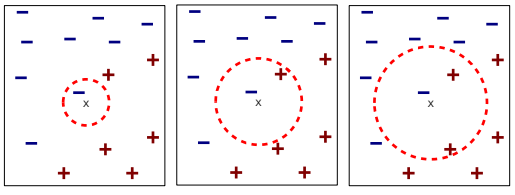


Nearest-Neighbor Classifiers



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

Definition of Nearest Neighbor

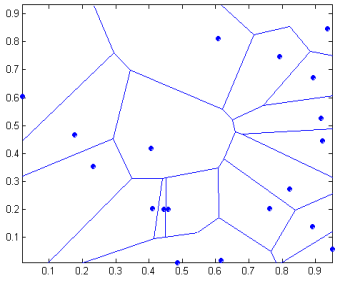


(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

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

1 nearest-neighbor

Voronoi Diagram



Nearest Neighbor Classification

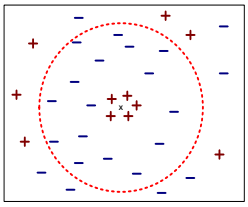
- Compute distance between two points:
  - Euclidean distance

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

- Determine the class from nearest neighbor list
  - take the majority vote of class labels among the k-nearest neighbors
  - Weigh the vote according to distance
    - weight factor, $w = 1/d^2$

Nearest Neighbor Classification…

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



Nearest Neighbor Classification…

- Scaling issues
  - Attribute domain should be normalized to prevent distance measures from being dominated by one of the attributes
  - Example:
    - height of a person may vary from 1.5m to 1.8m
    - weight of a person may vary from 90lb to 300lb
    - income of a person may vary from $10K to $1M

Nearest Neighbor Classification…

- Problem with distance measures
  - High dimensional data
    - curse of dimensionality
  - Can produce counter-intuitive results

1 1 1 1 1 1 1 1 1 1 1 0		1 0 0 0 0 0 0 0 0 0 0 0	
0 1 1 1 1 1 1 1 1 1 1 1	vs	0 0 0 0 0 0 0 0 0 0 0 1	
d = 1.4142		d = 1.4142	

Nearest neighbor Classification

- k-NN classifiers are lazy learners
  - The model is not built explicitly
  - Unlike eager learners such as decision tree induction and rule-based systems
  - Classifying unknown records is relatively expensive

Model Evaluation

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

Metrics for Performance Evaluation

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

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

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

Metrics for Performance Evaluation

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

- Most widely-used metric:

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

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
- Not appropriate for
  - unbalanced class label distribution
  - different class relevance

Class specific measures

- Evaluated separately for each class

$$\text{Recall} = \frac{\text{Number of objects correctly assigned to C}}{\text{Number of objects belonging to C}}$$

$$\text{Precision} = \frac{\text{Number of objects correctly assigned to C}}{\text{Number of objects assigned to C}}$$

- On the contingency table (binary classification problem), for the positive class

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

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

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

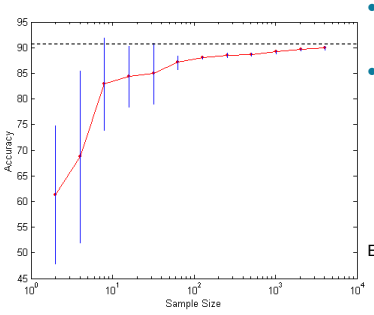
Model Evaluation

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

Methods for Performance Evaluation

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

Learning Curve



- Learning curve shows how accuracy changes with varying sample size
- Requires a sampling schedule for creating learning curve:
  - Arithmetic sampling (Langley, et al)
  - Geometric sampling (Provost et al)

Effect of small sample size:

- Bias in the estimate
- Variance of estimate

Methods of Estimation

- Partitioning labeled data in
  - training set for model building
  - test set for model evaluation
- Several partitioning techniques
  - holdout
  - cross validation
- Stratified sampling to generate partitions
  - without replacement
- Bootstrap
  - Sampling with replacement

Holdout

- Holdout
  - reserve 2/3 for training and 1/3 for testing
  - appropriate for large datasets
  - may be repeated several times
    - repeated holdout

Cross validation

- Cross validation
  - partition data into k disjoint subsets (i.e., folds)
  - k-fold: train on k-1 partitions, test on the remaining one
    - repeat for all folds
  - reliable accuracy estimation, not appropriate for very large datasets
- Leave-one-out
  - cross validation for k=n
  - only appropriate for very small datasets

Model Evaluation

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

ROC (Receiver Operating Characteristic)

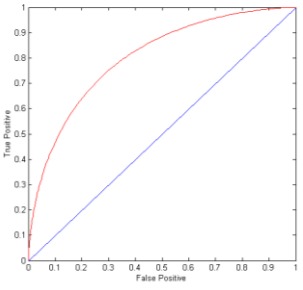
- Developed in 1950s for signal detection theory to analyze noisy signals
  - Characterize the trade-off between positive hits and false alarms
- ROC curve plots TPR (on the y-axis) against FPR (on the x-axis)
- Performance of each classifier represented as a point on the ROC curve
  - changing the threshold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

ROC Curve

(TPR,FPR):

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

- Diagonal line:
  - Random guessing
  - Below diagonal line:
    - prediction is opposite of the true class



How to construct a ROC curve

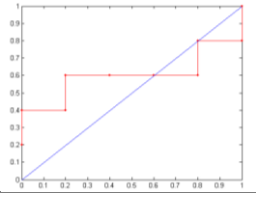
Instance	P(+|A)	True Class
1	0.95	+
2	0.93	+
3	0.87	-
4	0.85	-
5	0.85	-
6	0.85	+
7	0.76	-
8	0.53	+
9	0.43	-
10	0.25	+

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

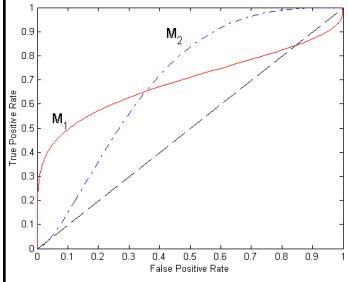
How to construct a ROC curve

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

ROC Curve:



Using ROC for Model Comparison



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