TIES443

Lecture 10

Classification, Part I: Basic Concepts & Approaches

Mykola Pechenizkiy

Course webpage: *http://www.cs.jyu.fi/~mpechen/TIES443*

November 22, 2006

Department of Mathematical Information Technology
University of Jyväskylä

Topics for today

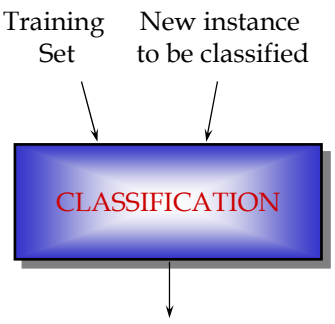
- The task of classification
- Classification techniques
  - Geometrical interpretation
  - Review of some most popular classifiers
  - Classification process: model construction and use
- Improvement of representation space
  - Dimensionality reduction, search for new representation
- Evaluation issues
  - Accuracy vs. utility
- Lecture summary and review what is left for tomorrow

Sources/Acknowledgements

- Beside my own slides, many other slides for this lecture are adopted (with modifications in places) from:
  - Eamonn Keogh's Introduction to Data Mining and Machine Learning: all slides on geometrical interpretation of classifies
  - Tan et al. book Chapter 4 (http://www-users.cs.umn.edu/~kumar/dmbook/dmslides/chap4_basic_classification.ppt) and Chapter 5 (http://www-users.cs.umn.edu/~kumar/dmbook/dmslides/chap5_alternative_classification.ppt)
  - Few slides by Piatetski-Shapiro

The task of classification

J classes, n training observations, p features



Given n training instances (x_i, y_i) where x_i are values of attributes and y is class

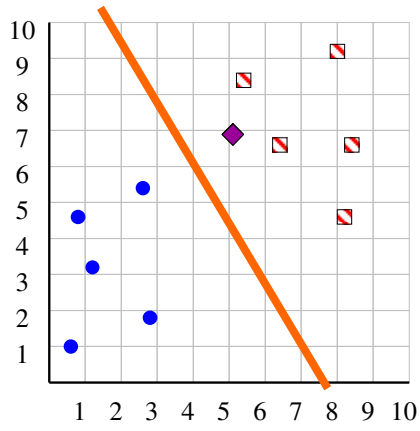
Find a *model* for class attribute as a function of the values of other attributes.

Goal: previously unseen records should be assigned a class as accurately as possible.

Classification Techniques

- Simple Liner Classifier
- Nearest Neighbor Classifier (Memory based reasoning)
- Naïve Bayes and Bayesian Belief Networks
- Support Vector Machines
- Decision Tree based Methods
- Rule-based Methods
- Neural Networks
- … lots of other types of techniques
- Combination of search for better representation space with classification
- Ensemble classification

Simple Linear Classifier

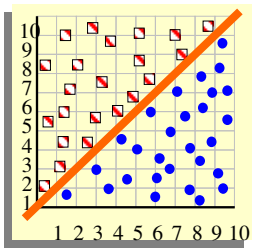


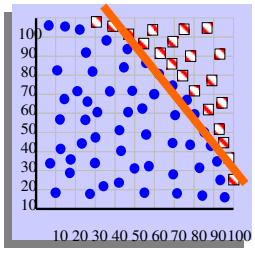
R.A. Fisher
1890-1962

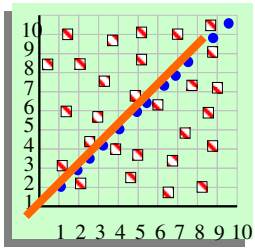
If previously unseen instance above the line
then class is **Red**
else class is **Blue**

What can be solved by the Simple Linear Classifier?

1) Perfect
2) Useless
3) Pretty Good

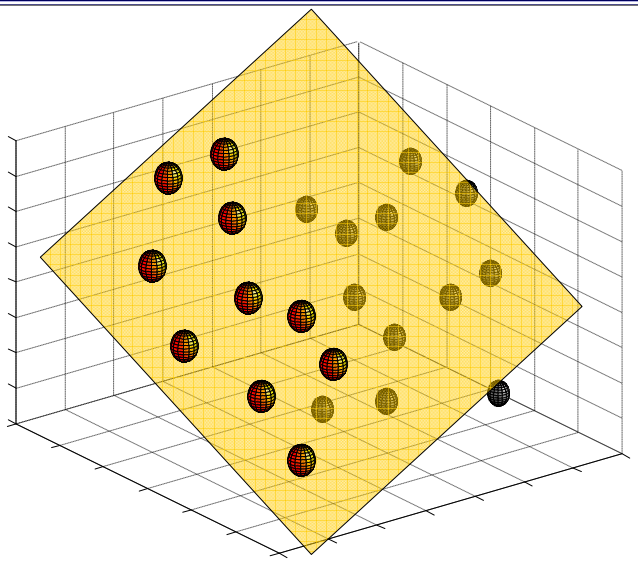




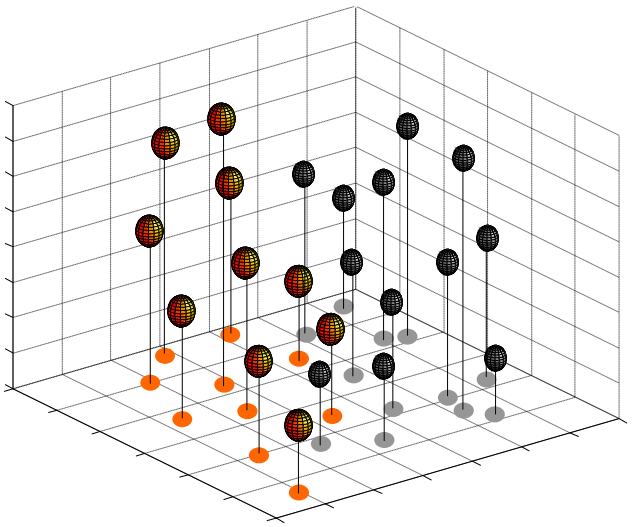


Problems that can be solved by a linear classifier are call **linearly separable**.

And in higher dimensional spaces…

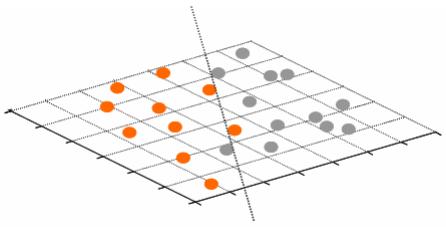


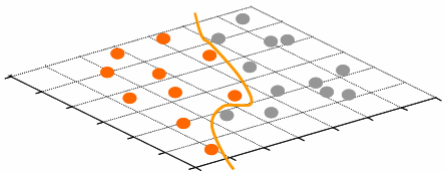
And if we did not have the 3rd dimension in our dataset …



We can no longer get perfe accuracy with the simple linear classifier…

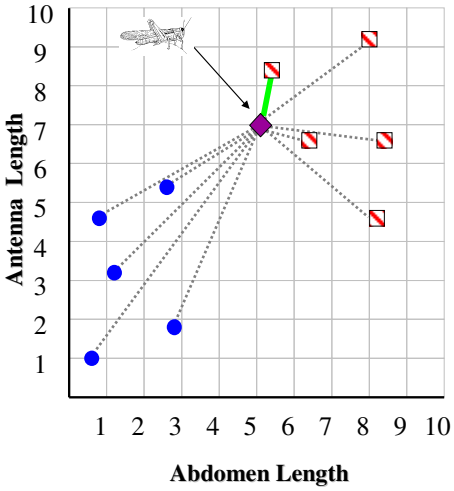
We could try to solve this problem by user a simple *quadratic* classifier or a simple *cubic* classifier..





However, as we will later see, this is probably a bad idea…

Nearest Neighbor Classifier



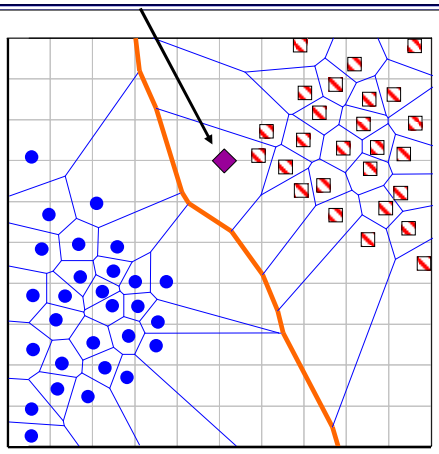
Joe Hodges
1922-2000

Katydids
Grasshoppers

If the **nearest** instance to the previously unseen instance **is a Katydid**
class is **Katydid**
else
class is **Grasshopper**

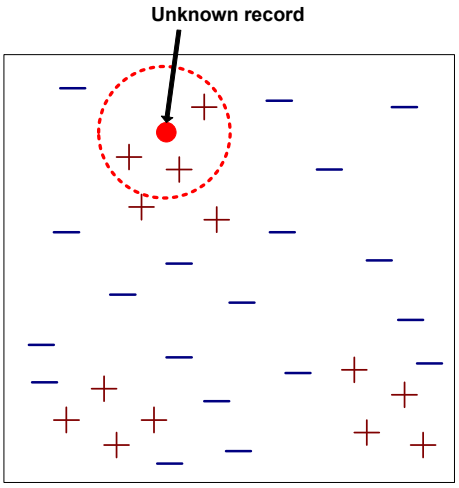
The Nearest Neighbor: a decision surface…

we don't have to construct these surfaces, they are simply the implicit boundaries that divide the space into regions "belonging" to each instance.



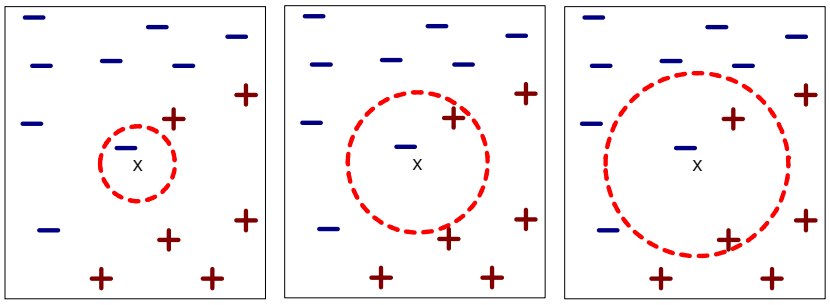
This division of space is called Dirichlet Tessellation (or Voronoi diagram, or Theissen regions).

Nearest-Neighbor Classifiers



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

Definition of Nearest Neighbor



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

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

Nearest Neighbor Classification

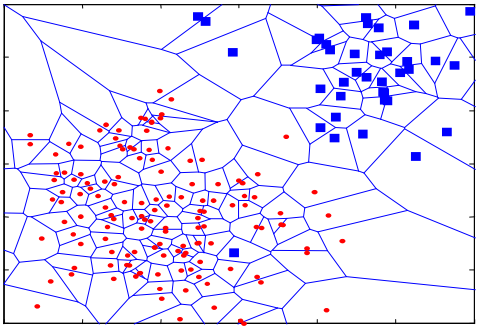
- Compute distance between two points:
  - Euclidean distance

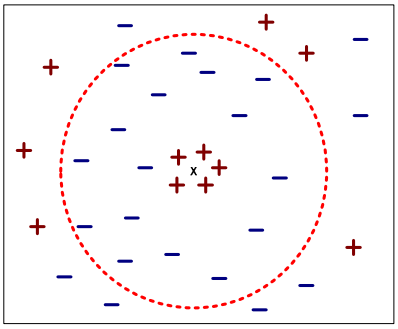
$$d(p,q) = \sqrt{\sum_i (p_i - q_i)^2}$$

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

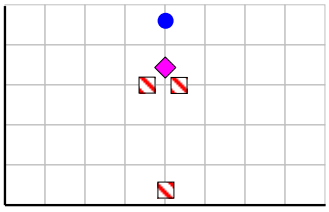
Nearest Neighbor Classification…

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



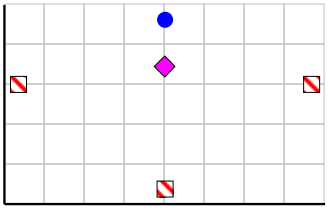


The nearest neighbor is sensitive to the units of measurement



X axis measured in **centimeters**
Y axis measure in dollars

The nearest neighbor to the **pink** unknown instance is **red**.



X axis measured in **millimeters**
Y axis measure in dollars

The nearest neighbor to the **pink** unknown instance is **blue**.

One solution is to normalize the units to pure numbers. Typically the features are Z-normalized to have a mean of zero and a standard deviation of one.
$X = (X - \text{mean}(X))/\text{std}(x)$

Nearest Neighbor: Proc and Cons

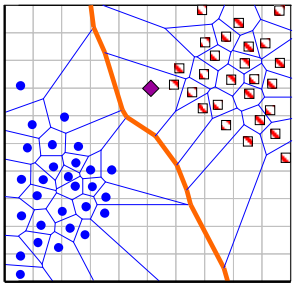
- Advantages:
  - Simple to implement
  - Handles correlated features (Arbitrary class shapes)
  - Defined for any distance measure
  - Handles streaming data trivially
- Disadvantages:
  - Very sensitive to irrelevant features.
  - k-NN classifiers are lazy learners
    - It does not build models explicitly (unlike eager learners such as decision tree induction)
  - Slow classification time for large datasets
  - Works best for real valued datasets

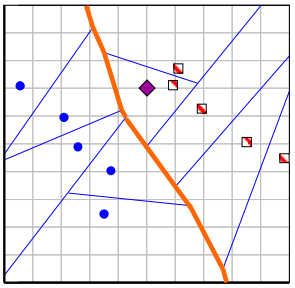
Speeding up nearest neighbor: "throwing away" some data

This is called data editing

Note that this can sometimes improve accuracy!

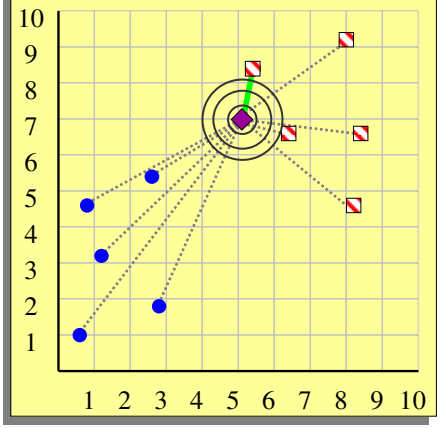
One possible approach. Delete all instances that are surrounded by members of their own class.





The Nearest Neighbor Algorithm: Distance

$$D(Q,C) \equiv \sqrt{\sum_{i=1}^{n}(q_i - c_i)^2}$$



$$D(Q,C) \equiv \sqrt[p]{\sum_{i=1}^{n}(q_i - c_i)^p}$$

Max (p=inf)

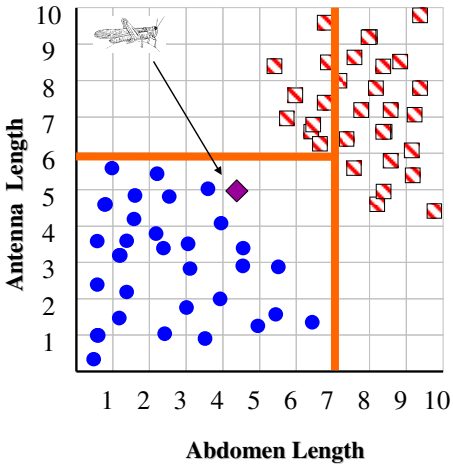
Manhattan (p=1)

Weighted Euclidean

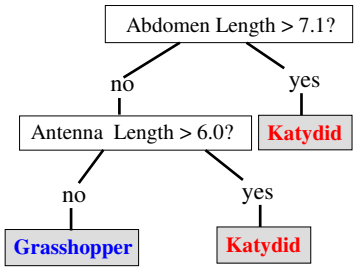
Mahalanob is

Specialized distance measures exist for DNA strings, time series, images, graphs, videos, sets, fingerprints etc…

Decision Tree Classifier



Ross Quinlan

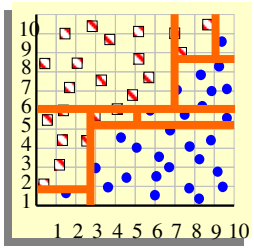


Decision Tree Learning

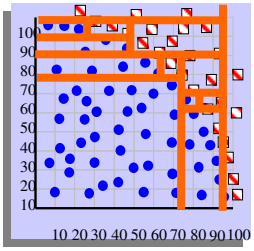
- Basic algorithm (a greedy algorithm)
  - Tree is constructed in a top-down recursive divide-and-conquer manner
  - At start, all the training examples are at the root
  - Attributes are categorical (if continuous-valued, they can be discretized in advance)
  - Examples are partitioned recursively based on selected attributes.
  - Test attributes are selected on the basis of a heuristic or statistical measure (e.g., information gain)
- Conditions for stopping partitioning
  - All samples for a given node belong to the same class
  - There are no remaining attributes for further partitioning – majority voting is employed for classifying the leaf
  - There are no samples left
  - Number of instances in the node is less than predefined value

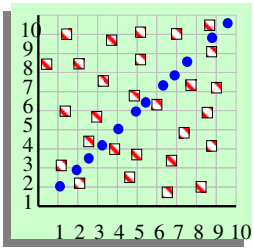
What can be solved by a Decision Tree?

1) Deep Bushy Tree
2) Useless
3) Deep Bushy Tree



The Decision Tree has a hard time with correlated attributes

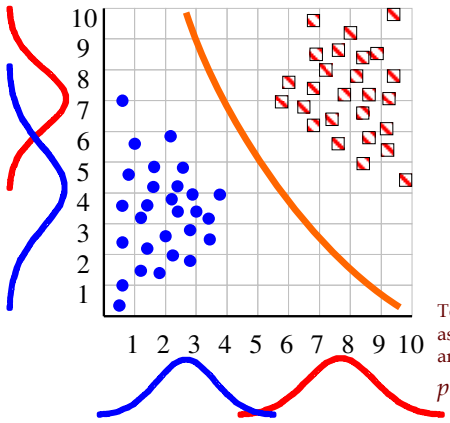




Decision Trees: Proc and Cons

- Advantages:
  - Easy to understand (domain experts love them!)
  - Easy to generate rules
- Disadvantages:
  - May suffer from overfitting.
  - Classifies by rectangular partitioning (so does not handle correlated features very well).
  - Can be quite large – pruning is necessary.
  - Does not handle streaming data easily

Naïve Bayes Classifier



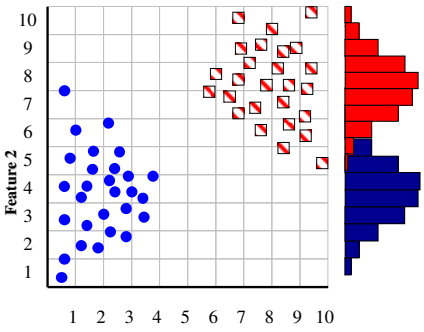
Thomas Bayes
1702 - 1761

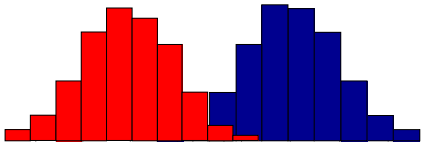
To simplify the task, **naïve Bayesian classifiers** assume attributes have independent distributions, and thereby estimate

$$p(d \mid c_j) = p(d_1 \mid c_j) * p(d_2 \mid c_j) * \ldots * p(d_n \mid c_j)$$

Find out the probability of the previously unseen instance belonging to each class, then simply pick the most probable class.

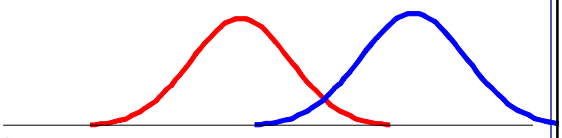
Naïve Bayes Classifier





We can leave the histograms as they are, or we can summarize them with two normal distributions.

Let us us two normal distributions for ease of visualization in the following slides…

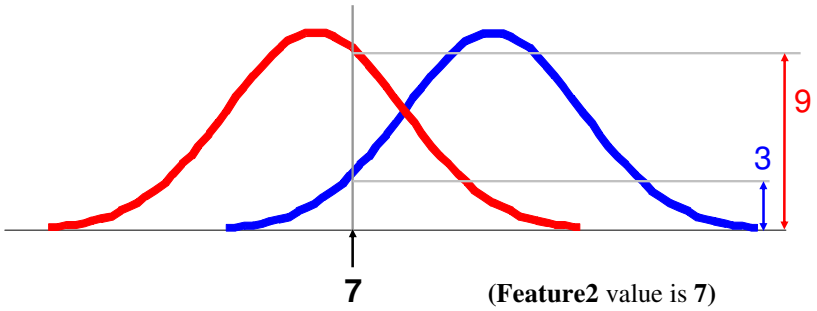


A Formal Way to Discuss The Most *Probable* Classification…

$p(c_j | d)$ = probability of class c_j, given that we have observed d

P(**blue** | **7**) = 3 / (3 + 9) = 0.250

P(**red** | **7**) = 9 / (3 + 9) = 0.750



(**Feature2** value is **7**)

Bayes Classifiers

That was a visual intuition for a simple case of the Bayes classifier, also called:

- Idiot Bayes
- Naïve Bayes
- Simple Bayes

We are about to see some of the mathematical formalisms, and more examples, but keep in mind the basic idea.

Find out the probability of the previously unseen instance belonging to each class, then simply pick the most probable class.

Bayes Classifiers

- Bayesian classifiers use **Bayes theorem**, which says

$$p(c_j \mid d) = \frac{p(d \mid c_j)\, p(c_j)}{p(d)}$$

- $p(c_j \mid d)$ = probability of instance d being in class c_j, This is what we are trying to compute
- $p(d \mid c_j)$ = probability of generating instance d given class c_j, We can imagine that being in class c_j, causes you to have feature d with some probability
- $p(c_j)$ = probability of occurrence of class c_j, This is just how frequent the class c_j, is in our database
- $p(d)$ = probability of instance d occurring

This can actually be ignored, since it is the same for all classes

Naïve Bayes assumes attributes are independent

$$p(d \mid c_j) = p(d_1 \mid c_j) * p(d_2 \mid c_j) * \ldots * p(d_n \mid c_j)$$

The probability of class c_j generating instance d, equals….

The probability of class c_j generating the observed value for feature 1, multiplied by..

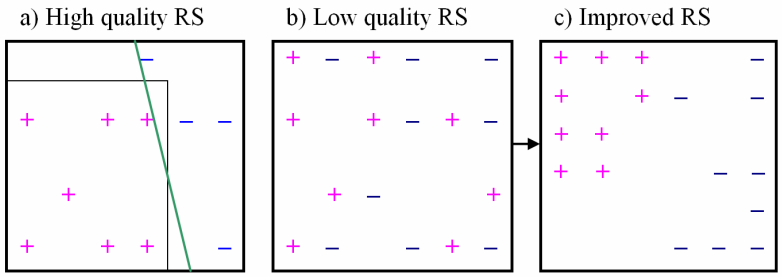
The probability of class c_j generating the observed value for feature 2, multiplied by..

Naïve Bayes: Proc and Cons

- Advantages:
  - Fast to train (single scan). Fast to classify
  - Not sensitive to irrelevant features
  - Handles real and discrete data
  - Handles streaming data well
- Disadvantages:
  - Assumes independence of features

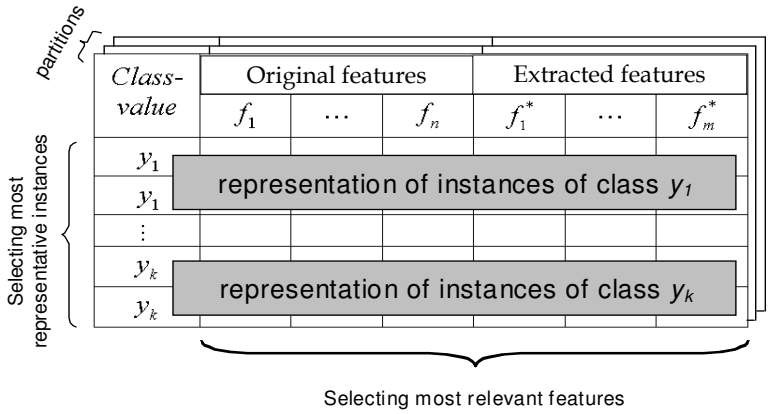
Improvement of Representation Space

- Curse of dimensionality
  - drastic increase in computational complexity and classification error with data having a large number of dimensions
- Indirectly relevant features



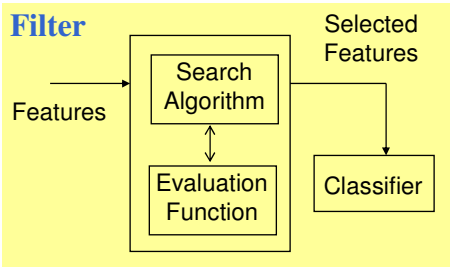
High vs. low quality RS for concept learning

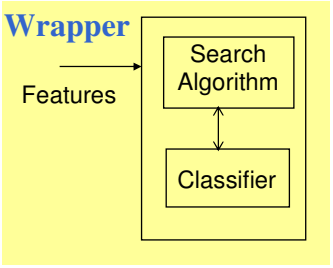
How to construct a good RS for Classification



Feature selection

- 2^n possible feature combinations
  - powerset of all features
- 2 (+ 1) evaluation strategies
  - Filter, Wrapper (+ Embedded)

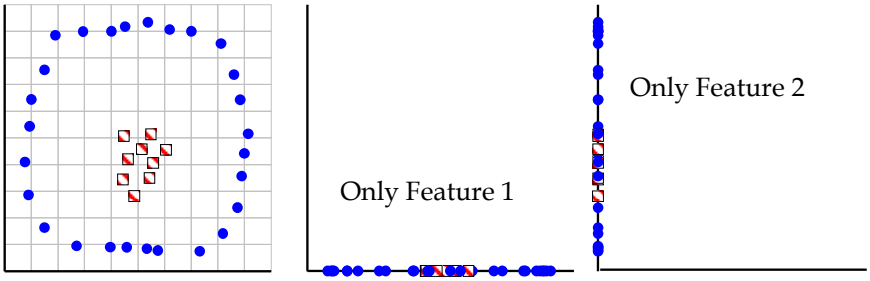




also **Embedded** as in Random Forests

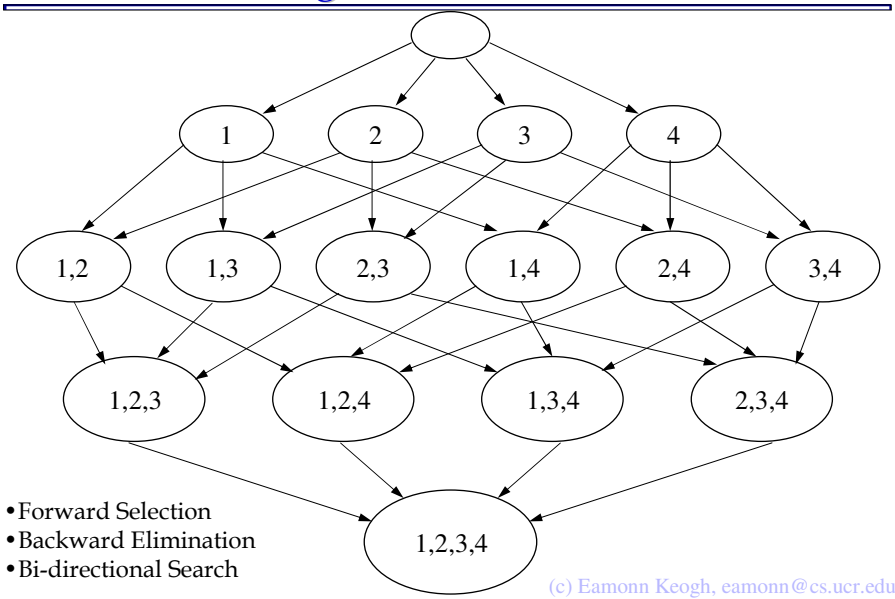
Why searching over feature subsets is hard

Suppose you have the following classification problem, with 100 features, where is happens that Features 1 and 2 (the X and Y below) give perfect classification, but all 98 of the other features are irrelevant…



Using all 100 features will give poor results, but so will using only Feature 1, and so will using Feature 2! Of the 2^{100} –1 possible subsets of the features, only one really works.

Searching over Feature Subsets



- Forward Selection
- Backward Elimination
- Bi-directional Search

Feature Extraction

100% Variance covered 87%

	age	sex	resting blood pressure	maximum heart rate achieved	3 = normal 7 = reversable defect	PC1	PC2	PC3	3 = normal 7 = reversable defect
1.	70	1	130	109	3		.490	-.852	3
2.	67	0	115	160				-.293	7
3.								-.918	7
4.	64	1	128	163			.513	-.840	7
5.	74	0	120	121			-.293	-.215	3
6.								-.872	7
7.								-.964	6
8.	59	1	110	142		-.095	.455	-.820	7
9.								-1.095	7
10.								-.527	7
11.	59	1	135	161	7	-.114	.288	-1.040	7
12.	53	1	142	111	7	-.272	.440	-.974	7
13.	44	1	140	180	3	.166	.205	-1.153	3
14.	61	1	134	145	3	-.208	.348	-.989	3
15.	57	0	128	159	3	-.162	-.455	-.400	3
16.	71	0	112	125	3	-.448	-.274	-.176	3
17.									
18.									
19.	64	1	110	144	3	-.159	.446	-.815	3
20.	40	1	140	178	7	.215	.213	-1.156	7

-0.7·Age+0.1·Sex-0.43·RestBP+0.57·MaxHeartRate

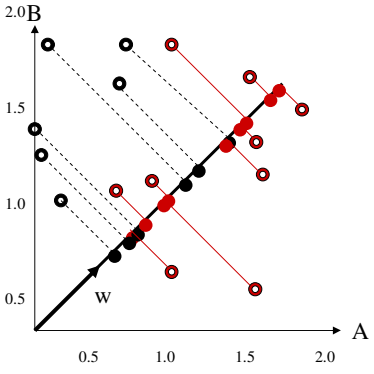
-0.01·Age+0.78·Sex-0.42·RestBP-0.47·MaxHeartRate

0.1·Age-0.6·Sex-0.73·RestBP-0.33·MaxHeartRate

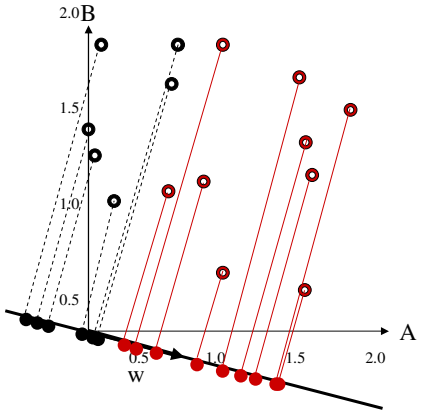
60% <= classification accuracy => 67%

PCA vs. Linear Discriminant Analysis

PCA

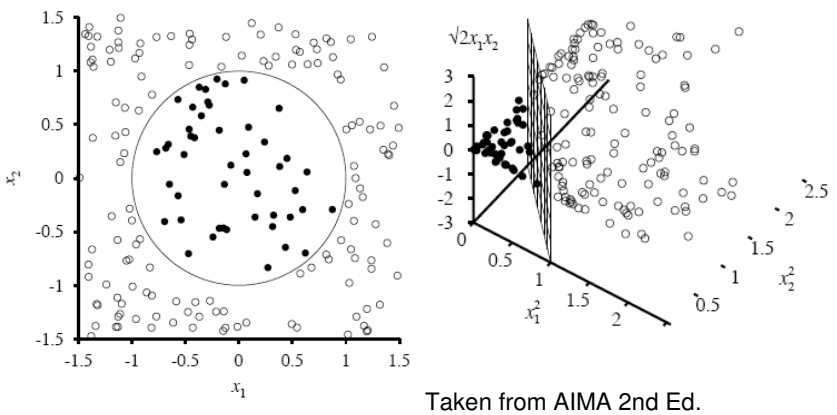


LDA: discovers a discriminating projection



Expanding the dimensionality: SVMs

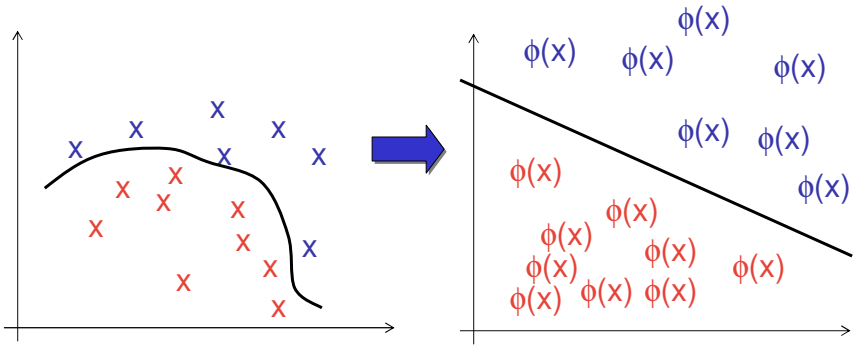
- Dot-product in higher dimension space where patterns are linearly separable.



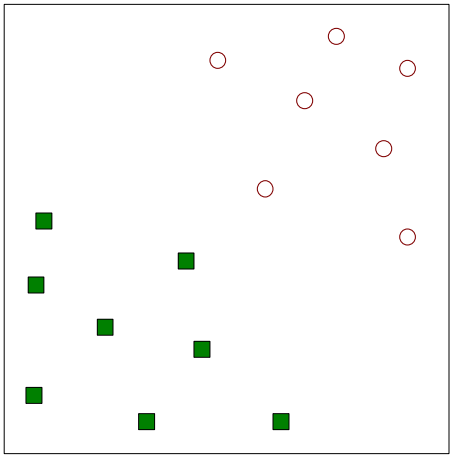
Taken from AIMA 2nd Ed.

Kernels & Linear Separability

Kernel projects data into a higher dimension space where classes are linearly separable.

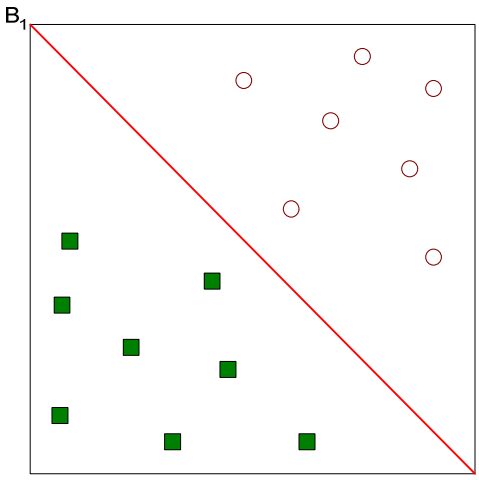


Support Vector Machines



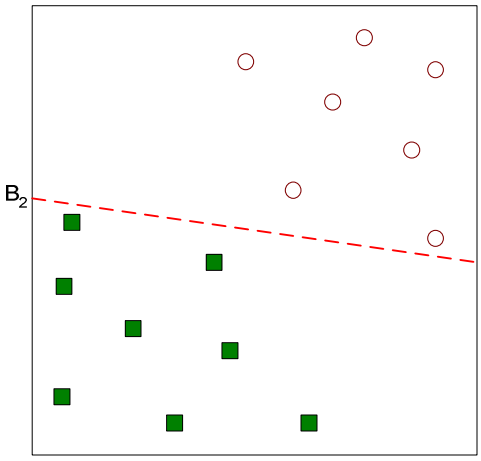
- Find a linear hyperplane (decision boundary) that will separate the data

Support Vector Machines



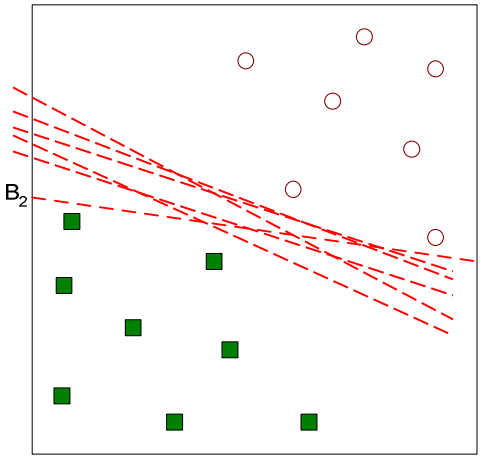
- One Possible Solution

Support Vector Machines



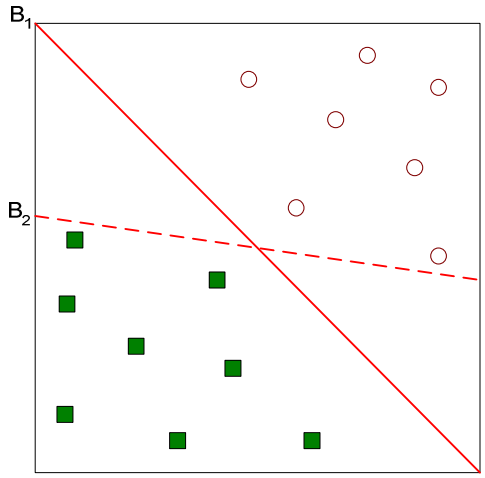
- Another possible solution

Support Vector Machines



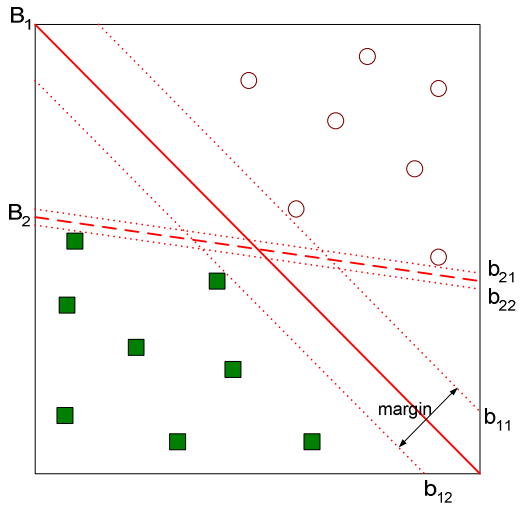
- Other possible solutions

Support Vector Machines



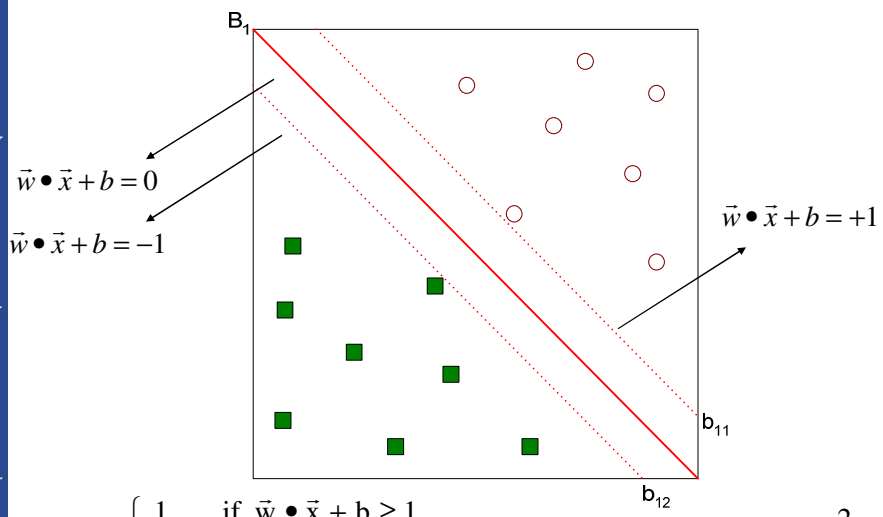
- Which one is better? B1 or B2?
- How do you define better?

Support Vector Machines



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

Support Vector Machines



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

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

Support Vector Machines

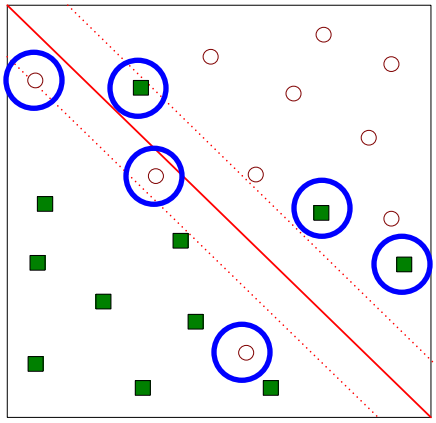
- We want to maximize: $\text{Margin} = \frac{2}{\| \vec{w} \|^2}$
  - Which is equivalent to minimizing: $L(w) = \frac{\| \vec{w} \|^2}{2}$
  - But subjected to the following constraints:

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

  - This is a constrained optimization problem
    - Numerical approaches to solve it (e.g., quadratic programming)

Support Vector Machines

- What if the problem is not linearly separable?



Support Vector Machines

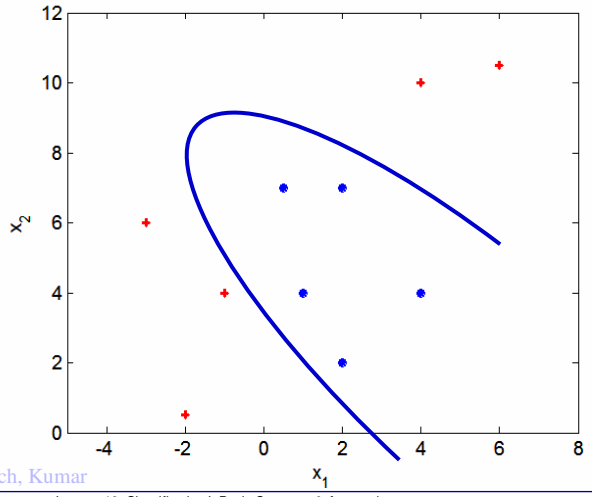
- What if the problem is not linearly separable?
  - Introduce slack variables
    - Need to minimize:
    - Subject to:

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

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

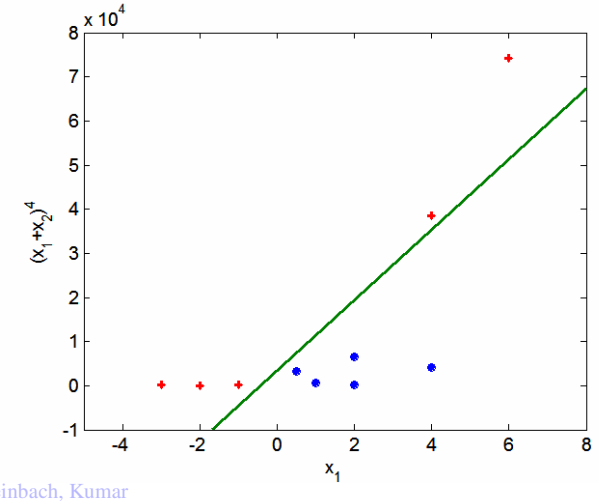
Nonlinear Support Vector Machines

- What if decision boundary is not linear?



Nonlinear Support Vector Machines

- Transform data into higher dimensional space



Classification – A Two-Step Process

- Model construction: describing a set of predetermined classes
  - Each tuple/sample is assumed to belong to a predefined class, as determined by the class label attribute
  - The set of tuples used for model construction: training set
  - The model is represented as classification rules, decision trees, or mathematical formulae
- Model usage: for classifying future or unknown objects
  - Estimate accuracy of the model
    - The known label of test sample is compared with the classified result from the model
    - Accuracy rate is the percentage of test set samples that are correctly classified by the model
    - Test set is independent of training set, otherwise over-fitting will occur

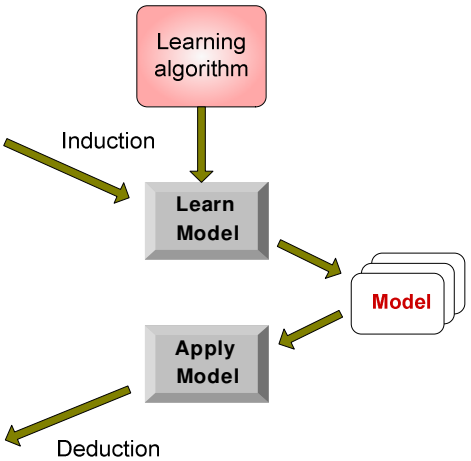
Classification Process: Model Induction & Use

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



Evaluating Classification Methods

- Predictive accuracy
  - We will focus on this issue during the lecture "Evaluation"
- Speed and scalability
  - time to construct the model
  - time to use the model
- Robustness
  - handling noise and missing values
- Relevance to current context
  - concept drift in data streams
- Scalability
  - efficiency in disk-resident databases
- Interpretability:
  - understanding and insight provided by the model
- Goodness of rules
  - decision tree size
  - compactness of classification rules

Utility-based Classification

- Cost-sensitive classification
  - Unbalanced datasets
  - Misclassification cost
  - Attribute/feature measuring cost
  - Data acquisition cost
  - Active learning -> Cost of asking an Oracle for a class label
  - Time complexity and associated costs
- Cost to develop and use a classification system
  - Implementation, testing, deploying, supporting
  - Use - impact factors, individual vs. organizational utility
    - Usability, transparency of classifier, interpretability of results
    - Increase/decrease of responsibility, satisfaction, employee/company image

Summary of Classification

We have seen some popular classification techniques: Simple linear classifier, Nearest neighbor, Decision tree, Naïve Bayes, and SVM.

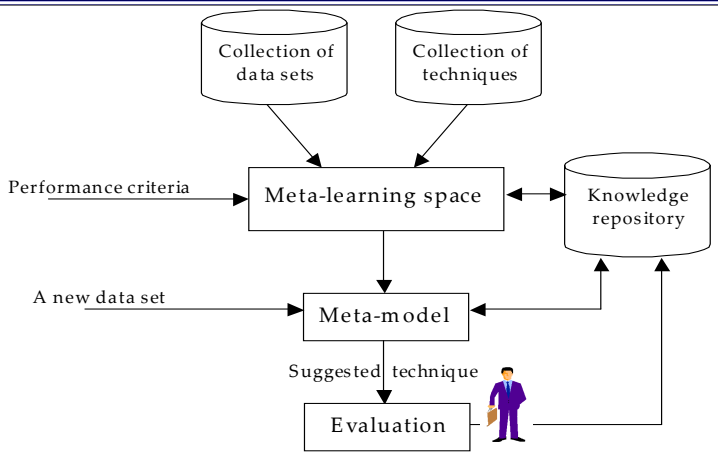
There are many other techniques:

Neural Networks, Genetic algorithms, Markov Models, Case Based Reasoning …

Ensemble learning - we will consider them tomorrow!

In general, there is no one best classifier for all problems. You have to consider what you hope to achieve, and the data itself… and decide about your DM strategy …

Meta-Learning



This is an extremely hard task to predict,
which technique will show the best performance on a given dataset

Summary

- What is classification
- Classifier induction and classifier application
- Geometrical interpretation of different classifiers
- Nearest Neighbour
- Naïve Bayes
- Decision Tree
- Linear Classifier
- Representation space improvement
- Utility-issues

What else did you get from this lecture?

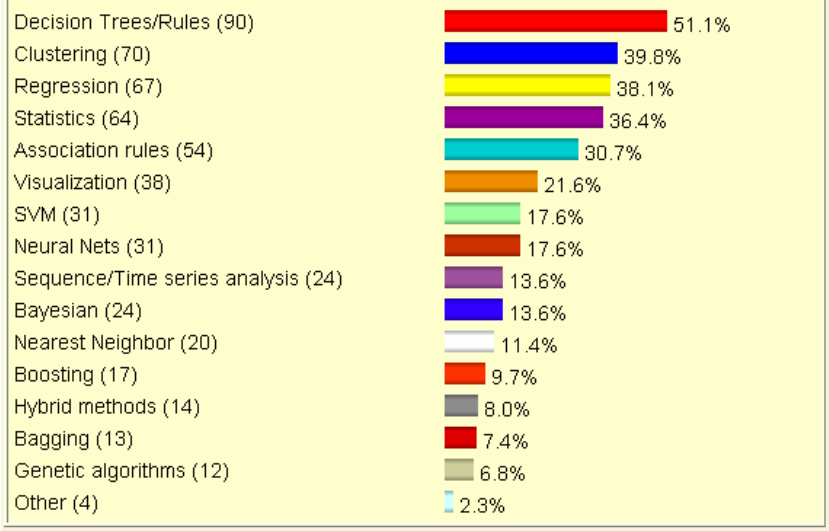
Additional Slides

WEKA – about 50 classifiers …

- ADTree, AODE, BayesNet, ComplementNaiveBayes, ConjunctiveRule, DecisionStump, DecisionTable, HyperPipes, IB1, IBk, Id3, J48, JRip, KStar, LBR, LeastMedSq, LinearRegression, LMT, Logistic, LogisticBase, M5Base, MultilayerPerceptron, MultipleClassifiersCombiner, NaiveBayes, NaiveBayesMultinomial, NaiveBayesSimple, NBTree, NNge, OneR, PaceRegression, PART, PreConstructedLinearModel, Prism, RandomForest, RandomizableClassifier, RandomTree, RBFNetwork, REPTree, Ridor, RuleNode, SimpleLinearRegression, SimpleLogistic, SingleClassifierEnhancer, SMO, SMOreg, VFI, VotedPerceptron, Winnow, ZeroR

KDnuggets : Polls : Data Mining Methods (Apr 2006)

Data mining/ analytic methods you used frequently in the last year: [176 voters]



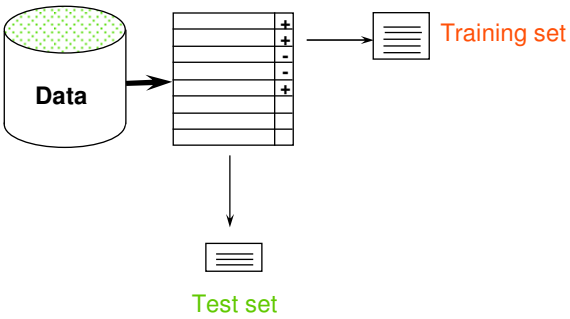
KDnuggets : Polls : Data Mining Techniques (Feb 2005)

Data mining/analytic techniques you use frequently: [784 votes total]

Decision Trees/Rules (107)	14%
Clustering (101)	13%
Regression (90)	11%
Statistics (80)	10%
Visualization (63)	8%
Neural Nets (61)	8%
Association rules (54)	7%
Nearest Neighbor (34)	4%
SVM (Support vector machine) (31)	4%
Bayesian (30)	4%
Sequence/Time series analysis (26)	3%
Boosting (25)	3%
Hybrid methods (23)	3%
Bagging (20)	3%
Genetic algorithms (19)	2%
Other (20)	3%

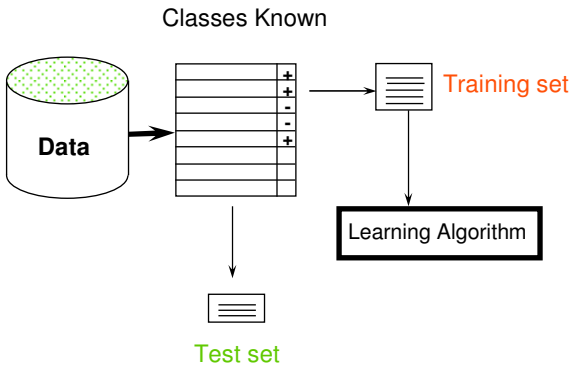
Hold-out estimate - step 1:

THE PAST
Classes Known



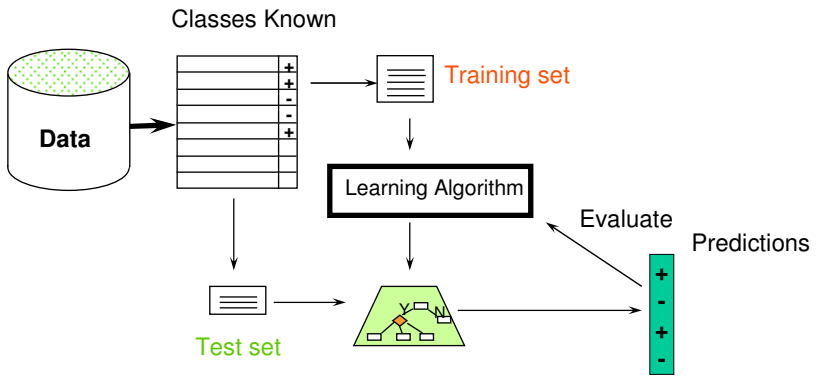
Split data into train and test sets

Hold-out estimate - step 2:



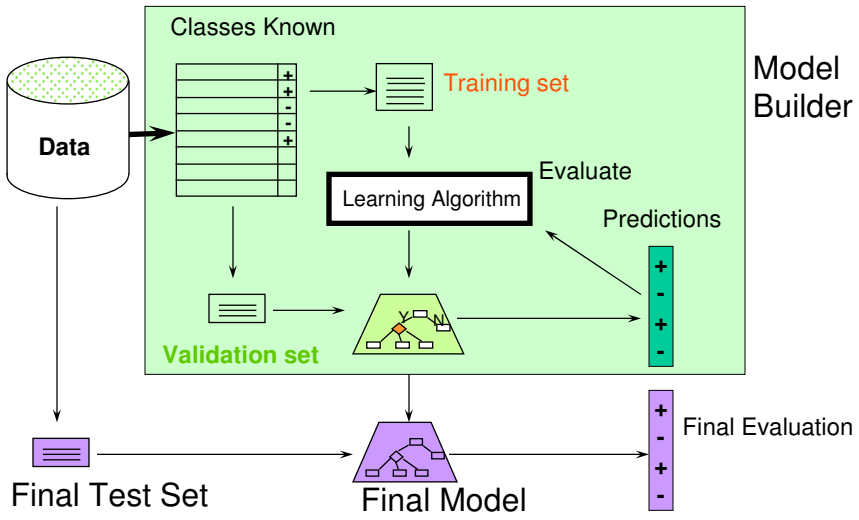
Build a model on a training set

Hold-out estimate - step 3:



Evaluate on test set (re-train?)

Train/ Validation/ Test split

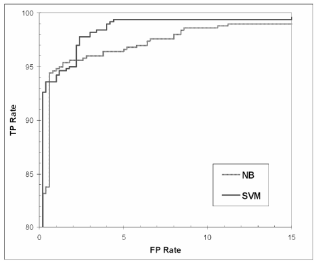


Accuracy, Confusion matrix, ROC curves

$$\text{Accuracy} = \frac{\text{Number of correct classifications}}{\text{Number of instances in our dataset}}$$

A **confusion matrix** gives us additional useful information, especially for unbalanced data and different misclassification costs…

A **ROC curve** analysis also gives us an additional insight on the behavior of classifier(s)



True label is...

Classified as a…

	Cat	Dog	Pig
Cat	100	0	0
Dog	9	90	1
Pig	45	45	10