**Machine learning methods:**

Machine learning algorithms are organized into taxonomy, based on the desired outcome of the algorithm.

Common algorithm types include:

* **Supervised learning** - Generates a function that maps inputs to desired outputs. For example, in a [classification](http://en.wikipedia.org/wiki/Statistical_classification) problem, the learner approximates a function mapping a vector into classes by looking at input-output examples of the function.
* **Unsupervised learning** - Models a set of inputs: like clustering
* **Semi-supervised learning** - Combines both labeled and unlabeled examples to generate an appropriate function or classifier.
* **Reinforcement learning** - Learns how to act given an observation of the world. Every action has some impact in the environment, and the environment provides feedback in the form of rewards that guides the learning algorithm.
* **Transduction** - Tries to predict new outputs based on training inputs, training outputs, and test inputs.
* **Learning to learn** - Learns its own inductive bias based on previous experience.
* **Pareto-based multi-objective learning** - a Pareto-based approach to learning that results in a set of learning models, which typically tradeoff between performance and complexity

Some of the machine learning methods categorized by their algorithm type are:

**Supervised learning**

* Artificial neural network
  + Backpropagation
* Bayesian statistics
  + Naive Bayes classifier
  + Bayesian network
  + Bayesian Knowledge base
* Case-based reasoning
* Decision trees
* Inductive logic programming
* Gaussian process regression
* Learning Automata
* Minimum message length (decision trees, decision graphs, etc.)
* Lazy learning
* Instance-based learning
  + Nearest Neighbor Algorithm
* Probably approximately correct learning (PAC) learning
* Support vector machines
* Regression analysis
* Information Fuzzy Networks (IFN)

**Statistical classification**

* Linear classifiers
  + Fisher's linear discriminant
  + Naive Bayes classifier
  + Perceptron
  + Support vector machines
* Quadratic classifiers
* k-nearest neighbor
* Boosting
* Decision trees
  + C4.5
  + Random forests
* Neural networks
* Bayesian networks
* Hidden Markov models

**Unsupervised learning**

* Artificial neural network
* Data clustering
* Expectation-maximization algorithm
* Self-organizing map
* Radial basis function network
* Generative topographic map
* Information bottleneck method

**Supervised learning – parametric and non-parametric approaches:** is a machine learning technique for deducing a function from training data. The training data consist of pairs of input objects (typically vectors), and desired outputs. The output of the function can be a continuous value (called regression), or can predict a class label of the input object (called classification). The task of the supervised learner is to predict the value of the function for any valid input object after having seen a number of training examples (i.e. pairs of input and target output). To achieve this, the learner has to generalize from the presented data to unseen situations in a "reasonable" way (see inductive bias).

**Parametric approaches:**

**Naïve Bayes:** The naive Bayes classifier applies to learning tasks where each instance x is described by a conjunction of attribute values and where the target function f ( x ) can take on any value from some finite set V. A set of training examples of the target function is provided, and a new instance is presented, described by the tuple of attribute values (a1, a2...an). The learner is asked to predict the target value, or classification, for this new instance. The Bayesian approach to classifying the new instance is to assign the most probable target value, VMAP given the attribute values (a1, a2 ...an) that describe the instance.

****

The naive Bayes classifier is based on the simplifying assumption that the attribute values are conditionally independent given the target value. In other words, the assumption is that given the target value of the instance, the probability of observing the conjunction **a1, a2...an**is just the product of the probabilities for the individual attributes: **P(a1, a2**...**an|vj)** = **P(ai|vj).**

****

To summarize, the naive Bayes learning method involves a learning step in which the various P(vj) and P(ai|vj) terms are estimated, based on their frequencies over the training data. The set of these estimates corresponds to the learned hypothesis.

**Support Vector Machines:** *Support Vector Machines* (SVMs) are motivated by many of the same considerations, but rely on preprocessing the data to represent patterns in a high dimension — typically much higher than the original feature space. With an appropriate nonlinear mapping () to a sufficiently high dimension, data from two categories can always be separated by a hyperplane (Problem 27). Here we assume each pattern **x***k* has been transformed to **y***k* = (**x***k*); we return to the choice of () below. For each of the *n* patterns, *k* = 1*,* 2*, ..., n*, we let *zk* = 1, according to whether pattern *k* is in *ω*1 or *ω*2. A linear discriminant in an augmented **y** space is



where both the weight vector and the transformed pattern vector are augmented (by *a*0 = *w*0 and *y*0 = 1, respectively). Thus a separating hyperplane insures



The goal in training a Support Vector Machine is to find the separating hyperplane with the largest margin; we expect that the larger the margin, the better generalization of the classifier. As illustrated in Fig. 5.2 the distance from any hyperplane to a (transformed) pattern **y** is |g(**y**)|/||**a**||, and assuming that a positive margin *b* exists:

  
the goal is to find the weight vector a that maximizes b. Of course, the solution vector can be scaled arbitrarily and still preserve the hyperplane, and thus to insure uniqueness we impose the constraint b ||a|| = 1;

The *support vectors* are the (transformed) training patterns represents an equality — that is, the support vectors are (equally) close to the hyperplane vector. The support vectors are the training samples that define the optimal separating hyperplane and are the most difficult patterns to classify. Informally speaking, they are the patterns most informative for the classification task. If *Ns* denotes the total number of support vectors, then for *n* training patterns the expected value of the generalization error rate is bounded, according to:



where the expectation is over all training sets of size *n* drawn from the (stationary) distributions describing the categories. This bound is independent of the dimensionality of the space of transformed vectors, determined by ().

**Non-parametric methods:**

**K nearest neighbor:** The most basic instance-based method is the k-NEAREST NEIGHBORING algorithm. This algorithm assumes all instances correspond to points in the n-dimensional space Rn. The nearest neighbors of an instance are defined in terms of the standard Euclidean distance. More precisely, let an arbitrary instance ***x*** be described by the feature vector







**Fisher’s linear discriminant:** We can reduce the dimensionality from *d* dimensions to one dimension if we merely project the *d*-dimensional data onto a line. Of course, even if the samples formed well-separated, compact clusters in *d*-space, projection onto an arbitrary line will usually produce a confused mixture of samples from all of the classes, and thus poor recognition performance. However, by moving the line around, we might be able to find an orientation for which the projected samples are well separated. This is exactly the goal of classical discriminant analysis.

Suppose that we have a set of *n d*-dimensional samples **x**1*, ...,* **x***n*, *n*1 in the subset *D*1 labelled *ω*1 and *n*2 in the subset *D*2 labelled *ω*2. If we form a linear combination of the components of **x**, we obtain the scalar dot product:



and a corresponding set of *n* samples *y*1*, ..., yn* divided into the subsets *Y*1 and *Y*2. Geometrically, if||**w**||= 1, each *yi* is the projection of the corresponding **x***i* onto a line in the direction of **w**. Actually, the magnitude of **w** is of no real significance, since it merely scales *y*. The direction of **w** is important, however. If we imagine that the samples labelled *ω*1 fall more or less into one cluster while those labelled *ω*2 fall in another, we want the projections falling onto the line to be well separated, not thoroughly intermingled. A measure of the separation between the projected points is the difference of the sample means. If **m***i* is the *d*-dimensional sample mean given by:



then the sample mean for the projected points is given by



The distance between the projected means is:



and that we can make this difference as large as we wish merely by scaling **w**. Rather than forming sample variances, we define the *scatter* for projected samples labelled *ωi* by:



Thus, (1*/n*)(˜*s*21 + ˜*s*22) is an estimate of the variance of the pooled data, and ˜*s*21 + ˜*s*22 is called the total *within-class scatter* of the projected samples.

The *Fisher linear discriminant* employs that linear function **w***t***x** for which the criterion function:



is maximum (and independent of ||**w**||). While the **w** maximizing *J*(*·*) leads to the best separation between the two projected sets (in the sense just described

**Unsupervised learning (self-learning) – minimax, kMeans:**

In [machine learning](http://en.wikipedia.org/wiki/Machine_learning), **unsupervised learning** is a class of problems in which one seeks to determine how the data are organized. It is distinguished from [supervised learning](http://en.wikipedia.org/wiki/Supervised_learning) (and [reinforcement learning](http://en.wikipedia.org/wiki/Reinforcement_learning)) in that the learner is given only unlabeled examples.

Unsupervised learning is closely related to the problem of [density estimation](http://en.wikipedia.org/wiki/Density_estimation) in [statistics](http://en.wikipedia.org/wiki/Statistics). However unsupervised learning also encompasses many other techniques that seek to summarize and explain key features of the data.

One form of unsupervised learning is [clustering](http://en.wikipedia.org/wiki/Data_clustering). Another example is blind source separation based on [Independent Component Analysis](http://en.wikipedia.org/wiki/Independent_Component_Analysis) (ICA).

Among [neural network](http://en.wikipedia.org/wiki/Neural_network) models, the [Self-Organizing Map](http://en.wikipedia.org/wiki/Self-Organizing_Map) (SOM) and [Adaptive resonance theory (ART)](http://en.wikipedia.org/wiki/Adaptive_resonance_theory) are commonly used unsupervised learning algorithms.

**Minimax**

Sometimes we must design our classifier to perform well over a *range* of prior probabilities. A reasonable approach is then to design our classifier so that the *worst* overall risk for any value of the priors is as small as possible — that is, minimize the maximum possible overall risk.

*R*1 denote that (as yet unknown) region in feature space where the classifier decides *ω*1 and likewise for *R*2 and *ω*2, and then write our overall risk:





This equation shows that once the decision boundary is set (i.e., *R*1 and *R*2 determined), the overall risk is linear in *P*(*ω*1). If we can find a boundary such that the constant of proportionality is 0, then the risk is independent of priors. This is the *minimax solution*, and the *minimax risk*, *Rmm*



Briefly stated, we search for the prior for whom the Bayes risk is *maximum*, the corresponding decision boundary gives the minimax solution. The value of the minimax risk, *Rmm*, is hence equal to the worst Bayes risk.

**K-means**

Is a method of [cluster analysis](http://en.wikipedia.org/wiki/Cluster_analysis) which aims to [partition](http://en.wikipedia.org/wiki/Partition_of_a_set) *n* observations into *k* clusters in which each observation belongs to the cluster with the nearest [mean](http://en.wikipedia.org/wiki/Mean). It is similar to the [expectation-maximization algorithm](http://en.wikipedia.org/wiki/Expectation-maximization_algorithm) for mixtures of [Gaussians](http://en.wikipedia.org/wiki/Gaussian_distribution) in that they both attempt to find the centers of natural clusters in the data.

Given a set of observations (**x**1, **x**2, …, **x***n*), where each observation is a *d*-dimensional real vector, then *k*-means clustering aims to partition the *n* observations into *k* sets (*k* < *n*) **S**={*S*1, *S*2, …, *Sk*} so as to minimize the within-cluster sum of squares (WCSS):



where ***μ****i* is the mean of *Si*.

The most common algorithm uses an iterative refinement technique. Due to its ubiquity it is often called the ***k*-means algorithm**; it is also referred to as [**Lloyd's algorithm**](http://en.wikipedia.org/wiki/Lloyd%27s_algorithm), particularly in the computer science community.

Given an initial set of *k* means **m**1(1),…,**m***k*(1), which may be specified randomly or by some heuristic, the algorithm proceeds by alternating between two steps: [[7]](http://en.wikipedia.org/wiki/K-means_clustering#cite_note-6)

**Assignment step**: Assign each observation to the cluster with the closest mean (i.e. partition the observations according to the [Voronoi diagram](http://en.wikipedia.org/wiki/Voronoi_diagram) generated by the means).



**Update step**: Calculate the new means to be the centroid of the observations in the cluster.

