**Analysis of Gene Expression Data** **Spring Semester, 2007**

# Lecture 6: May 31, 2007

*Lecturer: Gideon Dror* *Scribe: Yonit Halperin and David Burstein*[1]

## 6.1 Introduction to Classification

One of the major current applications of microarray technology is using genome-wide expression data in order to classify samples taken from different tissues. "MammaPrint", a microarray that was recently approved by the FDA for diagnosis of breast cancer, is an example for a novel application of such classification.

Classification as a discipline is usually viewed in the context of *Machine Learning* [2], a subfield of *Artificial Intelligence*. Classification is a form of *Supervised Learning*, which is sometimes termed as "learning with a teacher". The algorithm is first presented with an initial *training set* and is expected to extract from it sufficient information in order to successfully handle never-seen inputs.

## 6.2 Problem Definition

For simplicity, this lecture will deal solely with *binary classification* :

**Definition** The problem of binary classification is defined as:

**Input:** a set of $m$ examples $(x^j, y^j)$, $j = 1, 2...m$ (the *learning set*) sampled from some distribution $D$, where $x^j \in R^n$ and $y^j \in \{-1, +1\}$. The $i$-th component of $x^j$, $x^j_i$, is termed *feature i*.

**Output:** a function $f : R^n \rightarrow \{-1, +1\}$ which classifies "well" additional samples $\{x^k\}$ sampled from the same distribution $D$.

In the rest of the scribe $X_i$ will denote the $i$-th *feature* and $x^j_i$ the value of feature $i$ in the $j$-th sample. If $y^j = -1$ the sample will be referred to as a *"negative sample"*, and if $y^j = +1$ it will be referred to as a *"positive sample"*.

## 6.3 Example Applications

**Example** *Classification of tissue samples using gene expression data* - In this case each measured gene comprises a feature and the learning set is composed of vectors containing gene expression measurements for different tissues. The problem can be to classify the tissues as malignant or healthy ($Y$ = malignant/healthy) or to distinguish between different types

[1]Based in part on a scribes by Daniela Raijman and Igor Ulitsky March 2005, Simon Kamenkovich and Erez Greenstein May 2002.

of cancer. An example of the use of classification with gene expression data can be seen in Figure 6.1.

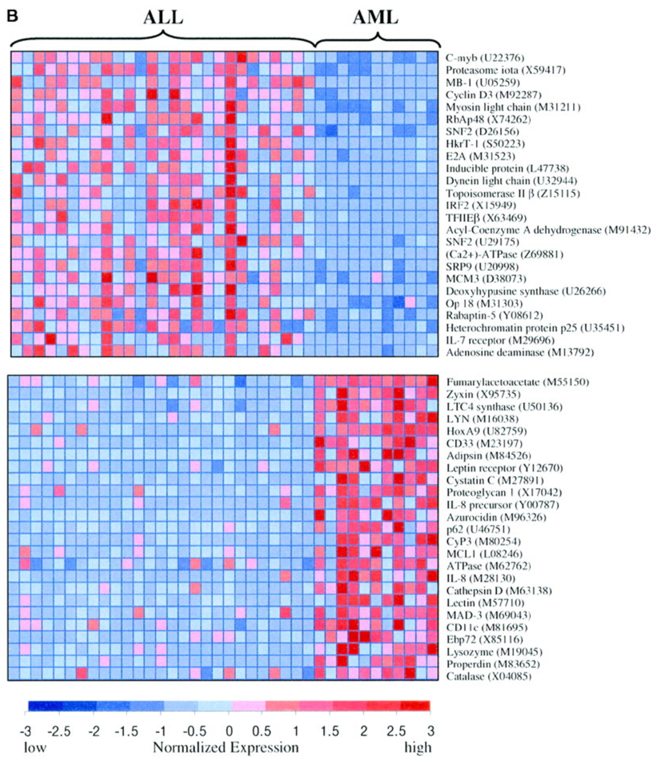

Figure 6.1: Genes distinguishing ALL from AML (two types of leukemia). The 50 genes which give the most information regarding the ALL-AML class are shown. Each row corresponds to a gene, and the columns correspond to expression level in different samples. Expression levels for each gene are normalized across the samples, such that the mean is 0 and the standard deviation is 1. Normalized expression levels greater than the mean are shaded in red, and those below the mean are shaded in blue. The scale indicates standard deviations above or below the mean.

**Example** *Detection of spam mail* - In this case the $X$ is some vector representation of e-mail messages (e.g. $X_i$ - the number of times the $i$-th word from some fixed dictionary appears in the message). The problem is to classify the mail into spam and non-spam. The training set in this case is list of messages that have been manually classified.

**Example** *Face Detection* - the problem of deciding whether a given image (represented as a vector of pixels) represents a human face. In this case $X_i$ can be the color intensity of the $i$-th pixel.

**Example** *Signature Recognition* - the classifier can be trained to recognize whether a certain signature belongs to a certain person.

The input to a classification problem can be visualized as in Figure 6.2 as a set of points in the plane, colored with 2 colors representing the two classes that need to be separated.

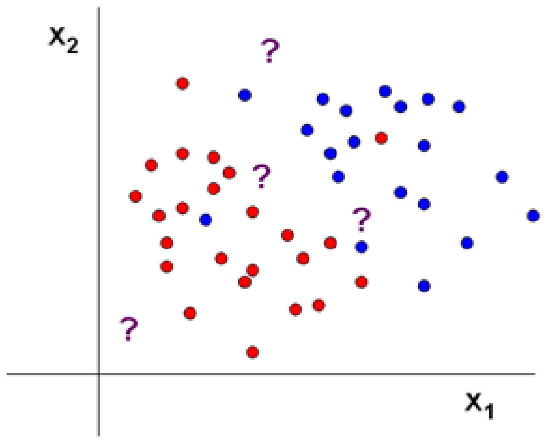

Figure 6.2: Classification in 2D plane. The red points indicate the positive samples, contained in the training set of the classifier and the blue points indicate the negative ones. The question marks stand for the locations of the new samples that need to be classified.

## 6.4 Classification Algorithms

All *classification algorithms* (termed *classifiers*) can be generalized as algorithms that receive a training set and learn a *classification function* of the form $f : R^n \to \{+1, -1\}$. This function is then applied to new inputs and its value represents the class to which the input is classified. Thus, different classification algorithms differ in the form of function they are capable of learning. The common classification algorithms include: Fisher Linear Discriminant (6.4.1), KNN (6.4.2), Decision Trees (6.4.3), Neural networks (6.4.4), Naïve Bayes (6.4.5), SVM (which will be discussed in the next lecture) and Adaboost ([6]).

### Properties of a Classifier

**Training speed** The amount of time the classifier needs in order to learn the classification function given a training set of a certain size.

**Accuracy** The accuracy of a classifier can be evaluated using a *test set* with known class derivation for every item. The classification error can be tested using varying schemes, which will be discussed in 6.7.1.

**Transparency** Some classification functions are very clear (for example a threshold function), making it possible to derive important insights about the problem or the data. On the other hand, some functions are very complex, making any such conclusions infeasible. A common example for a classification function which lacks transparency is the classification performed using *Neural Networks.*

**Hyper-parameters** These are parameters that are not being learned by the algorithm and are part of its input. High number of hyper-parameters might improve the classifier accuracy, but will lower transparency and increase its level of complication.

### 6.4.1 Fisher Linear Discriminant

*Fisher Linear Discriminant* is one of the simplest classification algorithms [3]. The method finds a direction $w$ in the $n$-dimensional space (a vector in $R^n$).

Given a sample that needs to be classified, the Fisher classifier calculates the projection of the sample onto $w$. The idea is to find a direction which, after the projection, will *maximize interclass variability and minimize intraclass variability.* It can be achieved by maximizing the following function:

$$J(w) = \frac{|m_1 - m_2|^2}{{s_1}^2 + {s_2}^2}$$

where $m_1$ and $m_2$ are the mean value of the projected positive and negative samples respectively. $s_1$ and $s_2$ are the standard deviations of the projected samples.

In the simple two-dimensional case, after the points are projected onto the line, the two classes are transformed into two sets of points upon the line. The *interclass variability* in this case is the distance between the class centers, and the *intraclass variability* is the distance of class members from their class centers.

Different criteria can be employed to determine the class for a new sample, for instance:

- Calculating the distance from the point to the means of the projections of the training classes.

- As above, but adding a weighting scheme in order to minimize the bias caused by the relative sizes of the training classes

The advantage of Fisher linear discriminant scheme is that the vector $w$ can be found swiftly using a simple procedure.

### 6.4.2 $k$ Nearest Neighbors

The *$k$ nearest neighbors (KNN)* classification scheme employs a more local method of classification. The method requires no initial processing of the training data, and new samples are classified based on their $k$ nearest neighbors in the training set. The KNN classifier has numerous variants, as the concept of proximity can be defined in various manners, as well as the decision rule of the new sample class based on the neighboring samples. For example, a simple variant of KNN would find the neighbors based on euclidian distance and use the majority rule to set the classification of the new sample. Another variation is giving each neighbor a weight according to its distance from the sample. In KNN $k$ is the only hyper-parameter of the algorithm. The KNN scheme is depicted in Figure 6.3. This kind of classifier is able to learn a relatively complex separation function. A drawback of this method is that in some practical problems, the euclidian distance is inappropriate, and the “correct” distance metric is difficult to define. Another problem is that as the method performs no preprocessing of the sample, the major computational complexity occurs while classifying unseen samples, a stage which should usually be swift.

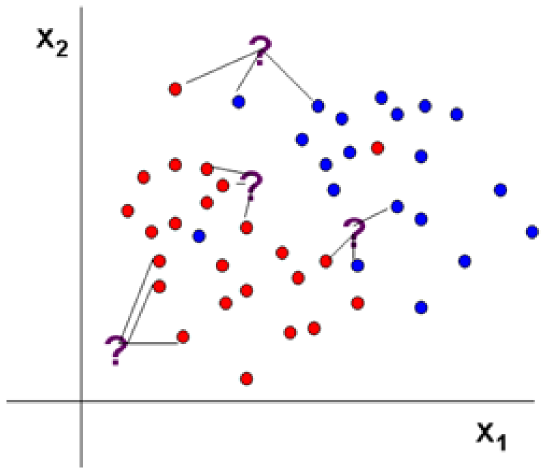

Figure 6.3: Classification using KNN. The class for each new sample (represented by question marks) is set based on its $k$ (3 in this case) closest neighbors.

### 6.4.3 Decision Tree

The *Decision Tree* method constructs a tree representing the classification process. The leaves of the tree represent one of the classes and the internal nodes contain some sort of decision function of the input sample with a boolean value. In the simplest case, this function is a predicate on one of the feathers, e.g. $x_1 > 3$ or $x_5 < 7$. Since the problem of building the most compact tree compatible with training examples is NP-complete, the algorithm applies heuristic methods for the tree construction. One of the simplest heuristics is selecting the most informative feature $X_i$ at every step and constructing an internal node in the tree, discriminating based on this feature. A sample decision tree is presented in Figure 6.4. One of the advantages of the decision tree model is the relative classification speed.In its basic form this is one of the simplest algorithms, as it has no hyper-parameters.

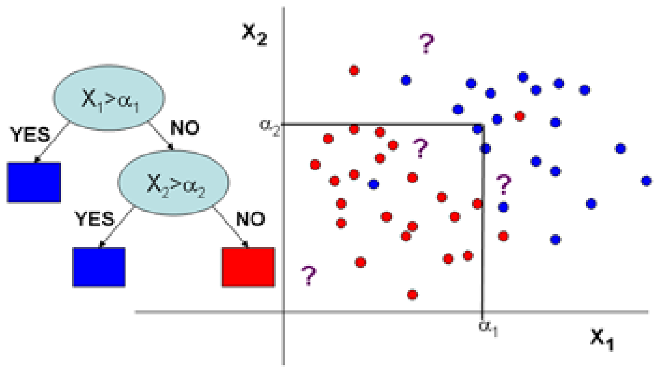

Figure 6.4: Classification using a decision tree. The tree presented in the left half of the figure describes the decision process, which in this case contains two simple predicates on the features $X_1$ and $X_2$. The figure on the right depicts the separation encoded in the tree.

### 6.4.4 Neural Networks

A neural network [1] is a combination of inter-connected networks where the nodes correspond to neurons and the arcs correspond to synaptic connections in the biological metaphor. A neural network represents a function, which is encoded in the weights of the arcs. The hyper-parameter in this case is the structure of the network.

A simple neural network with one layer and a single output neuron is termed *Perceptron* and it is capable of distinguishing between classes which can be separated by a straight line (hyperplane, as shown in Figure 6.5). In this aspect the perceptron is somewhat similar to the Fisher linear discriminant classifier. More complex neural networks with multiple layers and multiple output neurons are theoretically capable of separation using any continuous surface. However, the neural network model suffers from several drawbacks:
(1) The function constructed by the neural network lacks transparency, making it almost impossible to deduce conclusions regarding the data. In other words, the neural network is a "black box", which performs well in some situations. (2) The iterative algorithm employed for learning the classification function may converge slowly or not at all for some problems. (3) As any gradient-based iterative optimization search algorithm, the learning of the neural networks is susceptible to local minima. (4) Neural networks tend to be influenced by noise, and are prone to overfitting.

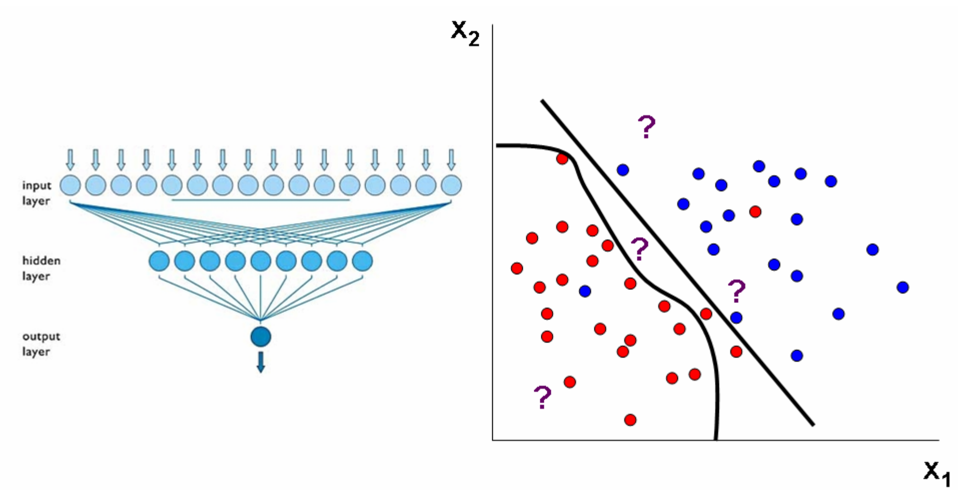

Figure 6.5: Separation by neural networks. The straight line depicts the separation achieved by a simple *Perceptron* and the curve the separation by a multi-layered network (left), which is in theory able to learn any separating function.

### 6.4.5 Naïve Bayes

The *Naïve Bayes* classifier is based on the concepts of *Bayesian decision theory*, which is closely related to *hypothesis testing*. The two classes are treated as the two hypotheses: $A$ is "this sample belongs to class A" and $B$ is "this sample belongs to class B". In Bayes theory, the decision between the two hypotheses is based on the *log-likelihood ratio*: $LR(x^j) = \log \frac{L_A(x^j)}{L_B(x^j)}$. If $LR(Data) > \log \frac{1-\lambda}{\lambda}$, $x^j$ is classified to class A. $\lambda$ in this case is the *prior* probability that $x^j$ belongs to class A. In this case it is assumed that the costs of deciding $A$ when $B$ is correct and vise versa are identical. For a detailed explanation of the naïve Bayes method see [4].

## 6.5 Dimensionality Reduction

Up to this point in the lecture we have dealt solely with $n$-dimensional data, $n$ being the original dimension of the input data. A reduction of the data dimension can provide with several important advantages for both learning and classification.

### 6.5.1 Dealing with Overfitting

One of the major problems encountered by all classifying schemes is *overfitting* of the data. The data in the learning set can be viewed as containing general information characterizing the population, along with information specific to the sampled training set. An ideal classifier is supposed to work only on the general characteristics. This is usually termed as performing *generalization*. If the classifier adheres strongly to signals specific to the learning set it is said to *overfit* it. For example, in the decision tree classifying scheme, a large tree containing multiple complex functions with a single training sample at each "leaf", will probably perform superbly on the training set, but poorly on new samples.

Overall, any complex separating function is vulnerable to overfitting, as can be seen in Figure 6.6. Reducing the dimensionality of the data can usually help overcome the maladies of overfitting by allowing the classifier to focus on the important features.

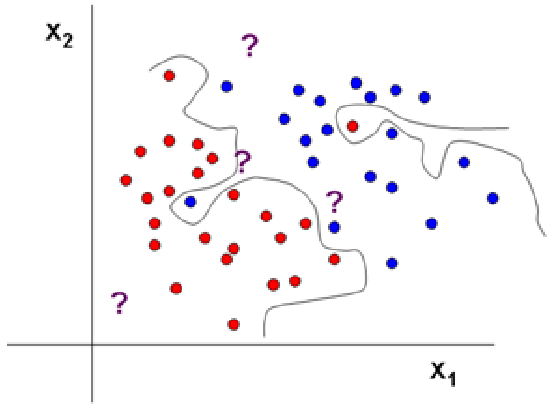

Figure 6.6: A classification function which overfits the training set.

### 6.5.2 Merits of the Dimensionality Reduction

- As described above, overfitting is reduced and generalization is improved .

- A well-performed dimensionality reduction can improve the performance of the classifier, by removing features that do not contribute to the classification, and may circumvent it with misleading noise.
- Several classification algorithms suffer from difficulties when dealing with high-dimensional data.
- In all classification schemes, a high dimensionality of the data causes greater time/memory consumption in the learning and classification phases.
- The use of fewer dimensions improves the clarity of the classification, allowing a better understanding of the meaningful signals found in the data.
- In the context of gene expression data, it is significantly easier, cheaper and more accurate to deal with expression measures from a small number of genes (e.g. 50) instead of a whole-genome survey (including up to 60,000 probes).

### 6.5.3 Approaches to Dimensionality Reduction

#### Feature Construction

In the *Feature Construction* approach, $n$ features of the input are transformed into $l$ other features using some linear/non-linear transformation (e.g. rotation). For example, in the application to face recognition problem, the $n$ pixels extracted from the image can be reduced to a set of distances between points with distinctive colors. A common method in feature construction is PCA - *Principal Component Analysis* [3], an analytical method which finds a linear transformation that chooses a new coordinate system for the data set. In the new coordinates, the greatest variance by any projection of the data set comes to lie on the first axis (termed the first principal component), the second greatest variance on the second axis, and so on. Other linear methods are ICA (independent component analysis, )[8] and Fisher linear discriminant. Examples for non-linear methods are non-linear component analysis, Kernel PCA [7] and LLE (Local linear embedding)[5].

#### Feature Selection

In *Feature Selection*, given a training set of $n$ dimensional samples, we're interested in selecting $l$ features, which maximize some trait of interest. This can of course be the performance of the classification process, but can also be some other trait, for instance detecting the important features (e.g., genes) in the training set.

An exhaustive search among the possible sets of selected features is infeasible in almost all practical cases, so heuristics are commonly employed. The integration of those with the classifier can be divided into three cases, which are depicted in Figure 6.7:

**Filter** The features are selected in a separate process before the classifier is trained. This method will be elaborated in 6.5.4

**Wrapper** In an iterative manner, the learning process alternates between selecting features and learning a classification function, improving the feature selection process based on the feedback from the classifier. Various optimization techniques can be used here, such as *hill climbing*. Two possible variations of hill climbing are: (a) Forward selection: keep adding features one at a time until no further improvement can be achieved; (b) Backward selection: start with the full set of predictors and keep removing features one at a time, until no further improvement can be achieved.

**Embedded** The selection process is embedded in the classifier. The difference from the Wrapper scheme is that in this case the two processes cannot be separated into two iteration phases. For example, the learning process of the decision trees includes an implicit selection of the features that appear in the node functions.

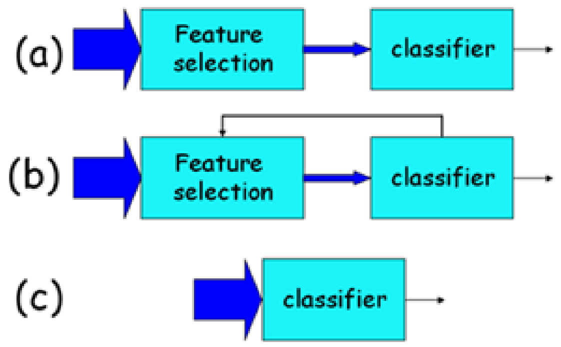

Figure 6.7: The three basic options of introducing feature selection to a classifier. (a) Filter (b) Wrapper (c) Embedded.

As can be seen from the definition, while the classifier is ignored in the filter method, the wrapper and embedding techniques can accommodate themselves to maximize the performance of a specific classifier.

### 6.5.4 Features Filtering

Even though the filtering approach does not take into consideration properties specific to the classifier, it can accomplish surprisingly good results. The filtering is usually based on extracting features $X_i$, which are more *informative* regarding the association of the samples to class $Y$. This notion of information can be captured by several means:

- *Pearson correlation* $\rho(X_i) = \frac{cov(X_i,Y)}{\sigma(X_i)\cdot\sigma(Y)}$ : Using this measure, features that are highly correlative to the class will be extracted. An example of such a feature can be seen in Figure 6.8.
- $\chi^2$ : This measure will extract features whose *distribution* is similar to that of $Y$.
- *Fisher criterion* $F(X_i) = \frac{\mu^+_{X_i}-\mu^-_{X_i}}{\sqrt{{\sigma^+_{X_i}}^2+{\sigma^-_{X_i}}^2}} > C$ : $\mu^+_{X_i}, \mu^-_{X_i}$ are the mean values of the $i$-th feature in the positive and negative samples, respectively, and $\sigma^+_{X_i}, \sigma^-_{X_i}$ are the standard

deviations of the $i$-th feature in the positive and negative samples. This measure prefers features with distinctively different distributions between the two target classes.

- *Golub criterion* $F(X_i) = \frac{\mu^+_{X_i} - \mu^-_{X_i}}{\|\sigma^+_{X_i}{}^2 + \sigma^-_{X_i}{}^2\|} > C$ : Similar to Fisher criterion.

- *Mutual information* $I(X_i, Y) = \sum P(X_i, Y) \log \frac{P(X_i,Y)}{P(X_i)P(Y)}$ : A quantity derived from *Information Theory* that measures the mutual dependence of the two random variables.

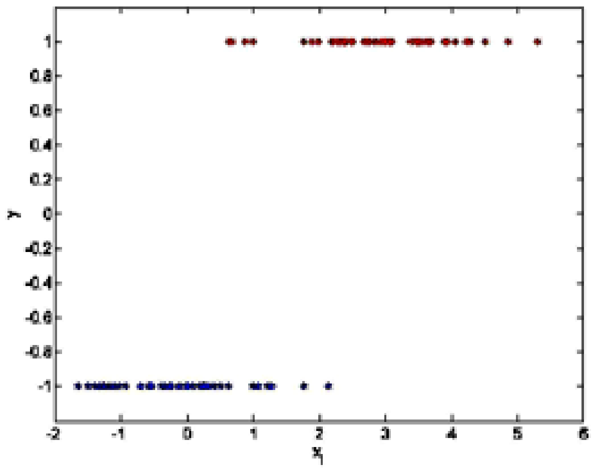

Figure 6.8: An example of a classification problem where the feature $X_i$ will be selected using *Pearson correlation*, as the feature is highly correlated with $Y$.

For all the measures described above, the most prominent $l$ features are extracted from the original $n$ features.

The filtering method suffers from a few drawbacks:

- The filtering performs well when the features are independent, since each feature is considered separately. When the dependencies between features and the targets are informative, filtering performance might decrease . A classic example of an ill performance of filtering is the XOR problem (see Figure 6.9)

- As mentioned above, the filtering disregards the classifier.

Nevertheless, the filtering method is very fast, thus allowing multiple filtering schemes to be efficiently tested in order to select the one giving the best performance results. For many practical purposes, filtering performs well, as can be seen in Figure 6.10.

## 6.6 Model Regularization

One of the methods for avoiding overfitting is regularization, enforcing the simplicity of the model used by the classifier. In decision trees classifiers, regularization can be applied

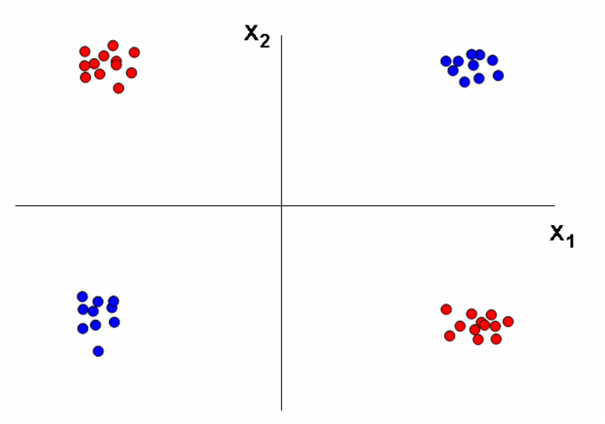

Figure 6.9: The XOR problem, using only $x_1$ or $x_2$, will result in poor separation of the samples, while by using both features a good separation can be achieved.

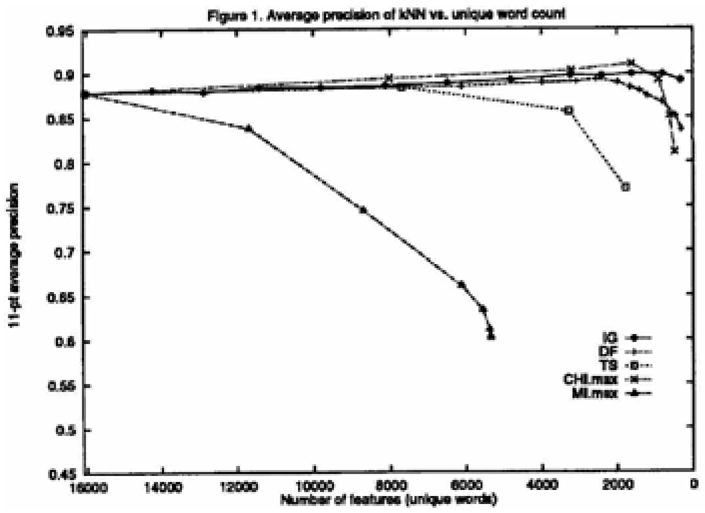

Figure 6.10: Results of applying the filtering feature selection on a dataset of e-mail messages classified to spam/non-spam. The classifier in this case is KNN and the precision of the classifier is plotted against the number of features used (in this case every unique word is a feature). The graphs represent different measures for filtering: **CHI.max** - $\chi^2$ measure, **MI.max** - Mutual information. As it can be seen, the optimal filtering is performed using $\chi^2$ and selecting about 2000 features.

by *pruning* - limiting the depth of the tree, and *ensembling* - building several trees and averaging their outputs. In order to avoid a too complex function in neural networks, arcs with a weight lower then some threshold are deleted, and the total sum of the weight squares is limited.

## 6.7 Performance Assessment

A crucial part of building a classifier lies in evaluating its performance. This evaluation is very important, as the classifier usually employs several parameters whose optimal values can be found using the assessment process. Also, in most problems several classifiers are available and the optimal choice is selected through trial and evaluation. In the context of feature selection, the number of features selected is usually determined using performance assessment.

### 6.7.1 Measures of Performance

There are several measures to quantify the performance of a classifier:

**Error rate** The most naïve measure is simply the fraction of samples that were misclassified. This measure might not reflect the performance reliably. For example, in a population with 99% healthy people, a classifier that classifies every sample to "healthy", will have an error rate of 0.01.

**Balanced error rate** $\frac{1}{2}(\frac{FP}{P} + \frac{FN}{N})$, where $P, N$ are the total number of samples that were classified as positive or negative respectively. $FP, FN$, are the number of samples that were misclassified as positive or negative respectively (false positives and false negatives). In the previous example the balanced error rate is 0.5.

**Area under ROC curve** The ROC (receiver operating characteristic) curve is a graphical plot of the fraction of false positives vs. the fraction of true positives encountered during the classifier testing (see Figure 6.11). It can be used for classifiers that assign for each sample a score representing the classification confidence rather than a binary output. The ROC curve can used for calibrating a threshold that will discriminate positive from negative samples. Note that a better classifier has a more convex ROC curve. The area under the ROC curve represents the probability of a random positive sample to receive a better score than a random negative sample. The expected area for a random classifier is 0.5.

### 6.7.2 Test Set Estimation

A naïve and wrong approach to performance assessment is to use the same data set for training the classifier and assessing its performance. This approach holds a grave problem, as it introduces a downward bias. After a complex enough learning procedure many classifiers are capable to perfectly classify the training data, yet perform poorly on new data (a symptom of overfitting). In order to overcome this, the classifier is trained using some part of the learning set, termed *training set*, and its performance is evaluated using an independent *test set*. The construction of those two sets can be performed using two main approaches:

- The initial learning set is divided into two distinct groups - $L_1$ (training set) and $L_2$ (test set). In order for this method to perform well, $L_1$ and $L_2$ must be approximately independently distributed. The main drawback of this method is the fact that the effective size of the learning set, the number of samples used for training, is reduced,

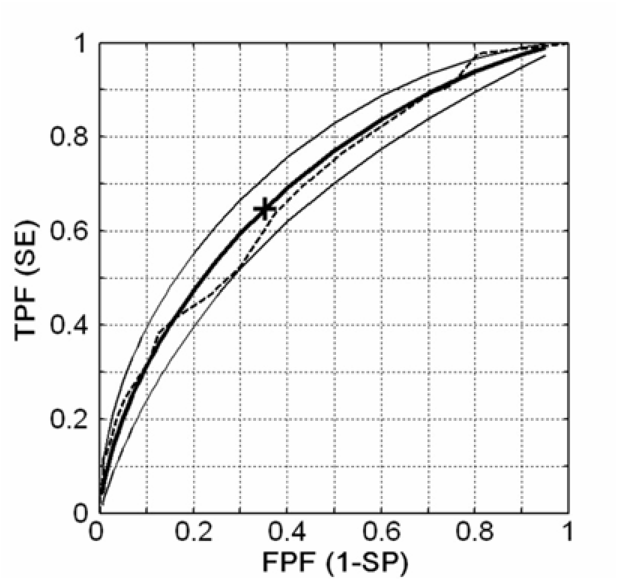

Figure 6.11: A ROC curve, plotting the fraction of false positives vs. the fraction of true positives

thus harming the training process. In addition only a small portion of the data is used as a test set, thus the performance assessment is less reliable.

- The method of *m-fold cross validation* randomly divides the learning set into $m$ distinct portions of nearly the same size. The learning is composed of $m$ iterations, where in each iteration one portion from the learning set is put aside, and the training is based on the rest of the samples. The performance is then evaluated based on the samples set aside. After $m$ iterations the average performance is calculated. A special case of the $m$-fold cross validation occurs when $m$ equals the number of samples - it is called *Leave-one-out cross validation (LOOCV).*

When feature selection is incorporated in the classification (Embedded scheme), the feature selection must be performed only by using the learning set in order to avoid a downward bias.

# Bibliography

[1] Anders Krogh J.Hertz and Richard G. Palmer. *Introduction to the theory of neural computation*. Perseus, 1991.

[2] Tom Mitchell. *Pattern Classification and Scene Analysis*. McGraw Hill, 1997.

[3] D.F. Morrison. *Multivariate Statistical Methods*. McGraw-Hill, 1990.

[4] Richard O.Duda and Peter E.Hart. *Pattern Classification and Scene Analysis*. John Wiley and Sons Inc., 1973.

[5] L.K. Saul and S.T. Roweis. An introduction to locally linear embedding. *Report at AT&T Labs-Research*, 2000.

[6] Yoav Freund Robert E. Schapire. A short introduction to boosting, http://www.site.uottawa.ca/ stan/csi5387/boost-tut-ppr.pdf. *Journal of Japanese Society for Artificial Intelligence*, 5(14):771–780, October 1999.

[7] B. Schölkopf, A. Smola, and K.R. Müller. Kernel principal component analysis. *Advances in Kernel Methods-Support Vector Learning*, pages 327–352, 1999.

[8] Richard Everson Stephen Roberts. *Independent Component Analysis: Principles and Practice*. Cambridge University Press, 2001.