Journal of Machine Learning Research 3 (2003) 1157-1182 Submitted 11/02; Published 3/03

An Introduction to Variable and Feature Selection

Isabelle Guyon ISABELLE@CLOPINET.COM
Clopinet
955 Creston Road
Berkeley, CA 94708-1501, USA

André Elisseeff ANDRE@TUEBINGEN.MPG.DE
Empirical Inference for Machine Learning and Perception Department
Max Planck Institute for Biological Cybernetics
Spemannstrasse 38
72076 Tübingen, Germany

Editor: Leslie Pack Kaelbling

Abstract

Variable and feature selection have become the focus of much research in areas of application for which datasets with tens or hundreds of thousands of variables are available. These areas include text processing of internet documents, gene expression array analysis, and combinatorial chemistry. The objective of variable selection is three-fold: improving the prediction performance of the predictors, providing faster and more cost-effective predictors, and providing a better understanding of the underlying process that generated the data. The contributions of this special issue cover a wide range of aspects of such problems: providing a better definition of the objective function, feature construction, feature ranking, multivariate feature selection, efficient search methods, and feature validity assessment methods.

Keywords: Variable selection, feature selection, space dimensionality reduction, pattern discovery, filters, wrappers, clustering, information theory, support vector machines, model selection, statistical testing, bioinformatics, computational biology, gene expression, microarray, genomics, proteomics, QSAR, text classification, information retrieval.

1 Introduction

As of 1997, when a special issue on relevance including several papers on variable and feature selection was published (Blum and Langley, 1997, Kohavi and John, 1997), few domains explored used more than 40 features. The situation has changed considerably in the past few years and, in this special issue, most papers explore domains with hundreds to tens of thousands of variables or features:[1] New techniques are proposed to address these challenging tasks involving many irrelevant and redundant variables and often comparably few training examples.

Two examples are typical of the new application domains and serve us as illustration throughout this introduction. One is gene selection from microarray data and the other is text categorization. In the gene selection problem, the variables are gene expression coefficients corresponding to the

1. We call "variable" the "raw" input variables and "features" variables constructed for the input variables. We use without distinction the terms "variable" and "feature" when there is no impact on the selection algorithms, e.g., when features resulting from a pre-processing of input variables are explicitly computed. The distinction is necessary in the case of kernel methods for which features are not explicitly computed (see section 5.3).

abundance of mRNA in a sample (e.g. tissue biopsy), for a number of patients. A typical classification task is to separate healthy patients from cancer patients, based on their gene expression "profile". Usually fewer than 100 examples (patients) are available altogether for training and testing. But, the number of variables in the raw data ranges from 6000 to 60,000. Some initial filtering usually brings the number of variables to a few thousand. Because the abundance of mRNA varies by several orders of magnitude depending on the gene, the variables are usually standardized. In the text classification problem, the documents are represented by a "bag-of-words", that is a vector of dimension the size of the vocabulary containing word frequency counts (proper normalization of the variables also apply). Vocabularies of hundreds of thousands of words are common, but an initial pruning of the most and least frequent words may reduce the effective number of words to 15,000. Large document collections of 5000 to 800,000 documents are available for research. Typical tasks include the automatic sorting of URLs into a web directory and the detection of unsolicited email (spam). For a list of publicly available datasets used in this issue, see Table 1 at the end of the paper.

There are many potential benefits of variable and feature selection: facilitating data visualization and data understanding, reducing the measurement and storage requirements, reducing training and utilization times, defying the curse of dimensionality to improve prediction performance. Some methods put more emphasis on one aspect than another, and this is another point of distinction between this special issue and previous work. The papers in this issue focus mainly on constructing and selecting *subsets of features* that are *useful* to build a good predictor. This contrasts with the problem of finding or ranking all potentially relevant variables. Selecting the most relevant variables is usually suboptimal for building a predictor, particularly if the variables are redundant. Conversely, a subset of useful variables may exclude many redundant, but relevant, variables. For a discussion of relevance *vs.* usefulness and definitions of the various notions of relevance, see the review articles of Kohavi and John (1997) and Blum and Langley (1997).

This introduction surveys the papers presented in this special issue. The depth of treatment of various subjects reflects the proportion of papers covering them: the problem of supervised learning is treated more extensively than that of unsupervised learning; classification problems serve more often as illustration than regression problems, and only vectorial input data is considered. Complexity is progressively introduced throughout the sections: The first section starts by describing *filters* that select variables by ranking them with correlation coefficients (Section 2). Limitations of such approaches are illustrated by a set of constructed examples (Section 3). Subset selection methods are then introduced (Section 4). These include *wrapper methods* that assess subsets of variables according to their usefulness to a given predictor. We show how some embedded methods implement the same idea, but proceed more efficiently by directly optimizing a two-part objective function with a goodness-of-fit term and a penalty for a large number of variables. We then turn to the problem of feature construction, whose goals include increasing the predictor performance and building more compact feature subsets (Section 5). All of the previous steps benefit from reliably assessing the statistical significance of the relevance of features. We briefly review model selection methods and statistical tests used to that effect (Section 6). Finally, we conclude the paper with a discussion section in which we go over more advanced issues (Section 7). Because the organization of our paper does not follow the work flow of building a machine learning application, we summarize the steps that may be taken to solve a feature selection problem in a check list[2]:

2. We caution the reader that this check list is heuristic. The only recommendation that is almost surely valid is to try the simplest things first.

1. **Do you have domain knowledge?** If yes, construct a better set of "ad hoc" features.

2. **Are your features commensurate?** If no, consider normalizing them.

3. **Do you suspect interdependence of features?** If yes, expand your feature set by constructing conjunctive features or products of features, as much as your computer resources allow you (see example of use in Section 4.4).

4. **Do you need to prune the input variables** (e.g. for cost, speed or data understanding reasons)? If no, construct disjunctive features or weighted sums of features (e.g. by clustering or matrix factorization, see Section 5).

5. **Do you need to assess features individually** (e.g. to understand their influence on the system or because their number is so large that you need to do a first filtering)? If yes, use a variable ranking method (Section 2 and Section 7.2); else, do it anyway to get baseline results.

6. **Do you need a predictor?** If no, stop.

7. **Do you suspect your data is "dirty"** (has a few meaningless input patterns and/or noisy outputs or wrong class labels)? If yes, detect the outlier examples using the top ranking variables obtained in step 5 as representation; check and/or discard them.

8. **Do you know what to try first?** If no, use a linear predictor.[3] Use a forward selection method (Section 4.2) with the "probe" method as a stopping criterion (Section 6) or use the ℓ_0-norm embedded method (Section 4.3). For comparison, following the ranking of step 5, construct a sequence of predictors of same nature using increasing subsets of features. Can you match or improve performance with a smaller subset? If yes, try a non-linear predictor with that subset.

9. **Do you have new ideas, time, computational resources, and enough examples?** If yes, compare several feature selection methods, including your new idea, correlation coefficients, backward selection and embedded methods (Section 4). Use linear and non-linear predictors. Select the best approach with model selection (Section 6).

10. **Do you want a stable solution** (to improve performance and/or understanding)? If yes, subsample your data and redo your analysis for several "bootstraps" (Section 7.1).

2 Variable Ranking

Many variable selection algorithms include variable ranking as a principal or auxiliary selection mechanism because of its simplicity, scalability, and good empirical success. Several papers in this issue use variable ranking as a baseline method (see, e.g., Bekkerman et al., 2003, Caruana and de Sa, 2003, Forman, 2003, Weston et al., 2003). Variable ranking is not necessarily used to build predictors. One of its common uses in the microarray analysis domain is to discover a set of drug leads (see, e.g., et al., 1999): A ranking criterion is used to find genes that discriminate between healthy and disease patients; such genes may code for "drugable" proteins, or proteins that may

3. By "linear predictor" we mean linear in the parameters. Feature construction may render the predictor non-linear in the input variables.

themselves be used as drugs. Validating drug leads is a labor intensive problem in biology that is outside of the scope of machine learning, so we focus here on building predictors. We consider in this section ranking criteria defined for individual variables, independently of the context of others. Correlation methods belong to that category. We also limit ourselves to supervised learning criteria. We refer the reader to Section 7.2 for a discussion of other techniques.

2.1 Principle of the Method and Notations

Consider a set of m examples $\{\mathbf{x}_k, y_k\}$ $(k = 1,...m)$ consisting of n input variables $x_{k,i}$ $(i = 1,...n)$ and one output variable y_k. Variable ranking makes use of a scoring function $S(i)$ computed from the values $x_{k,i}$ and y_k, $k = 1,...m$. By convention, we assume that a high score is indicative of a valuable variable and that we sort variables in decreasing order of $S(i)$. To use variable ranking to build predictors, nested subsets incorporating progressively more and more variables of decreasing relevance are defined. We postpone until Section 6 the discussion of selecting an optimum subset size.

Following the classification of Kohavi and John (1997), variable ranking is a *filter* method: it is a preprocessing step, independent of the choice of the predictor. Still, under certain independence or orthogonality assumptions, it may be optimal with respect to a given predictor. For instance, using Fisher's criterion[4] to rank variables in a classification problem where the covariance matrix is diagonal is optimum for Fisher's linear discriminant classifier (Duda et al., 2001). Even when variable ranking is not optimal, it may be preferable to other variable subset selection methods because of its computational and statistical scalability: Computationally, it is efficient since it requires only the computation of n scores and sorting the scores; Statistically, it is robust against overfitting because it introduces bias but it may have considerably less variance (Hastie et al., 2001).[5]

We introduce some additional notation: If the input vector $\mathbf{x}$ can be interpreted as the realization of a random vector drawn from an underlying unknown distribution, we denote by X_i the random variable corresponding to the i^{th} component of $\mathbf{x}$. Similarly, Y will be the random variable of which the outcome y is a realization. We further denote by $\mathbf{x}_i$ the m dimensional vector containing all the realizations of the i^{th} variable for the training examples, and by $\mathbf{y}$ the m dimensional vector containing all the target values.

2.2 Correlation Criteria

Let us consider first the prediction of a continuous outcome y. The Pearson correlation coefficient is defined as:

$$\mathcal{R}(i) = \frac{cov(X_i, Y)}{\sqrt{var(X_i)var(Y)}} , \tag{1}$$

where cov designates the covariance and var the variance. The estimate of $R(i)$ is given by:

$$R(i) = \frac{\sum_{k=1}^{m}(x_{k,i} - \bar{x}_i)(y_k - \bar{y})}{\sqrt{\sum_{k=1}^{m}(x_{k,i} - \bar{x}_i)^2 \sum_{k=1}^{m}(y_k - \bar{y})^2}} , \tag{2}$$

4. The ratio of the between class variance to the within-class variance.
5. The similarity of variable ranking to the ORDERED-FS algorithm (Ng, 1998) indicates that its sample complexity may be logarithmic in the number of irrelevant features, compared to a power law for "wrapper" subset selection methods. This would mean that variable ranking can tolerate a number of irrelevant variables exponential in the number of training examples.

where the bar notation stands for an average over the index k. This coefficient is also the cosine between vectors $\mathbf{x}_i$ and $\mathbf{y}$, after they have been centered (their mean subtracted). Although the $R(i)$ is derived from $\mathcal{R}(i)$ it may be used without assuming that the input values are realizations of a random variable.

In linear regression, the coefficient of determination, which is the square of $R(i)$, represents the fraction of the total variance around the mean value $\bar{y}$ that is explained by the linear relation between $\mathbf{x}_i$ and $\mathbf{y}$. Therefore, using $R(i)^2$ as a variable ranking criterion enforces a ranking according to goodness of linear fit of individual variables.[6]

The use of $R(i)^2$ can be extended to the case of two-class classification, for which each class label is mapped to a given value of y, e.g., ± 1. $R(i)^2$ can then be shown to be closely related to Fisher's criterion (Furey et al., 2000), to the T-test criterion, and other similar criteria (see, e.g., et al., 1999, Tusher et al., 2001, Hastie et al., 2001). As further developed in Section 6, the link to the T-test shows that the score $R(i)$ may be used as a test statistic to assess the significance of a variable.

Correlation criteria such as $R(i)$ can only detect linear dependencies between variable and target. A simple way of lifting this restriction is to make a non-linear fit of the target with single variables and rank according to the goodness of fit. Because of the risk of overfitting, one can alternatively consider using non-linear preprocessing (e.g., squaring, taking the square root, the log, the inverse, etc.) and then using a simple correlation coefficient. Correlation criteria are often used for microarray data analysis, as illustrated in this issue by Weston et al. (2003).

2.3 Single Variable Classifiers

As already mentioned, using $R(i)^2$ as a ranking criterion for *regression* enforces a ranking according to goodness of linear fit of individual variables. One can extend to the *classification* case the idea of selecting variables according to their individual predictive power, using as criterion the performance of a classifier built with a single variable. For example, the value of the variable itself (or its negative, to account for class polarity) can be used as discriminant function. A classifier is obtained by setting a threshold θ on the value of the variable (e.g., at the mid-point between the center of gravity of the two classes).

The predictive power of the variable can be measured in terms of error rate. But, various other criteria can be defined that involve false positive classification rate *fpr* and false negative classification rate *fnr*. The tradeoff between *fpr* and *fnr* is monitored in our simple example by varying the threshold θ. ROC curves that plot "hit" rate *(1-fpr)* as a function of "false alarm" rate *fnr* are instrumental in defining criteria such as: The "Break Even Point" (the hit rate for a threshold value corresponding to *fpr=fnr*) and the "Area Under Curve" (the area under the ROC curve).

In the case where there is a large number of variables that separate the data perfectly, ranking criteria based on classification success rate cannot distinguish between the top ranking variables. One will then prefer to use a correlation coefficient or another statistic like the margin (the distance between the examples of opposite classes that are closest to one another for a given variable).

6. A variant of this idea is to use the mean-squared-error, but, if the variables are not on comparable scales, a comparison between mean-squared-errors is meaningless. Another variant is to use $R(i)$ to rank variables, not $R(i)^2$. Positively correlated variables are then top ranked and negatively correlated variables bottom ranked. With this method, one can choose a subset of variables with a given proportion of positively and negatively correlated variables.

The criteria described in this section extend to the case of binary variables. Forman (2003) presents in this issue an extensive study of such criteria for binary variables with applications in text classification.

2.4 Information Theoretic Ranking Criteria

Several approaches to the variable selection problem using information theoretic criteria have been proposed (as reviewed in this issue by Bekkerman et al., 2003, Dhillon et al., 2003, Forman, 2003, Torkkola, 2003). Many rely on empirical estimates of the mutual information between each variable and the target:

$$I(i) = \int_{x_i} \int_{y} p(x_i, y) \log \frac{p(x_i, y)}{p(x_i)p(y)} dxdy \ , \tag{3}$$

where $p(x_i)$ and $p(y)$ are the probability densities of x_i and y, and $p(x_i, y)$ is the joint density. The criterion $I(i)$ is a measure of dependency between the density of variable x_i and the density of the target y.

The difficulty is that the densities $p(x_i)$, $p(y)$ and $p(x_i, y)$ are all unknown and are hard to estimate from data. The case of discrete or nominal variables is probably easiest because the integral becomes a sum:

$$I(i) = \sum_{x_i} \sum_{y} P(X = x_i, Y = y) \log \frac{P(X = x_i, Y = y)}{P(X = x_i)P(Y = y)} \ . \tag{4}$$

The probabilities are then estimated from frequency counts. For example, in a three-class problem, if a variable takes 4 values, $P(Y = y)$ represents the class prior probabilities (3 frequency counts), $P(X = x_i)$ represents the distribution of the input variable (4 frequency counts), and $P(X = x_i, Y = y)$ is the probability of the joint observations (12 frequency counts). The estimation obviously becomes harder with larger numbers of classes and variable values.

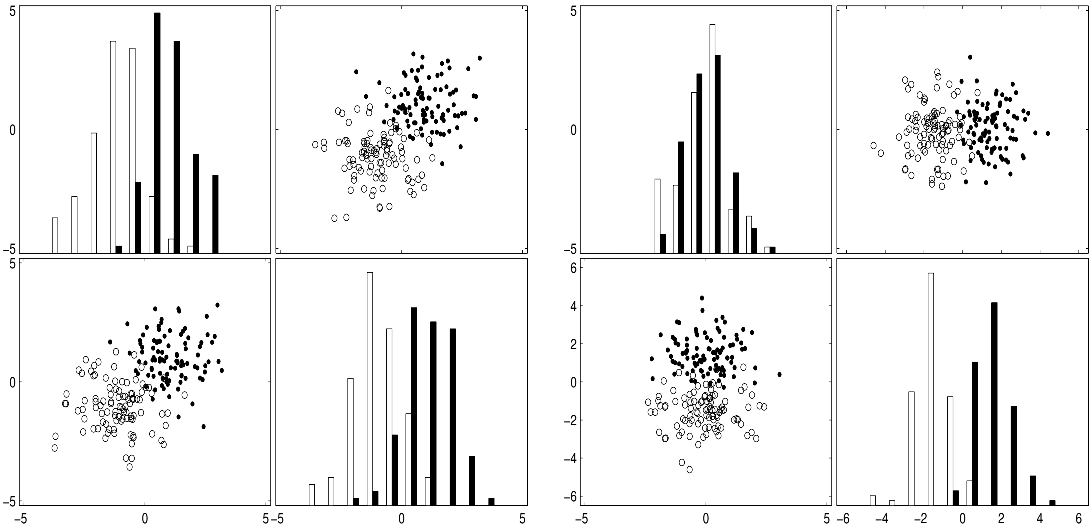
The case of continuous variables (and possibly continuous targets) is the hardest. One can consider discretizing the variables or approximating their densities with a non-parametric method such as Parzen windows (see, e.g., Torkkola, 2003). Using the normal distribution to estimate densities would bring us back to estimating the covariance between X_i and Y, thus giving us a criterion similar to a correlation coefficient.

3 Small but Revealing Examples

We present a series of small examples that outline the usefulness and the limitations of variable ranking techniques and present several situations in which the variable dependencies cannot be ignored.

3.1 Can Presumably Redundant Variables Help Each Other?

One common criticism of variable ranking is that it leads to the selection of a redundant subset. The same performance could possibly be achieved with a smaller subset of complementary variables. Still, one may wonder whether adding presumably redundant variables can result in a performance gain.



(a) (b)

Figure 1: **Information gain from presumably redundant variables.** (a) A two class problem with independently and identically distributed (i.i.d.) variables. Each class has a Gaussian distribution with no covariance. (b) The same example after a 45 degree rotation showing that a combination of the two variables yields a separation improvement by a factor $\sqrt{2}$. I.i.d. variables are not truly redundant.

Consider the classification problem of Figure 1. For each class, we drew at random $m = 100$ examples, each of the two variables being drawn independently according to a normal distribution of standard deviation 1. The class centers are placed at coordinates (-1; -1) and (1; 1). Figure 1.a shows the scatter plot in the two-dimensional space of the input variables. We also show on the same figure histograms of the projections of the examples on the axes. To facilitate its reading, the scatter plot is shown twice with an axis exchange. Figure 1.b shows the same scatter plots after a forty five degree rotation. In this representation, the x-axis projection provides a better separation of the two classes: the standard deviation of both classes is the same, but the distance between centers in projection is now $2\sqrt{2}$ instead of 2. Equivalently, if we rescale the x-axis by dividing by $\sqrt{2}$ to obtain a feature that is the average of the two input variables, the distance between centers is still 2, but the within class standard deviation is reduced by a factor $\sqrt{2}$. This is not so surprising, since by averaging n i.i.d. random variables we will obtain a reduction of standard deviation by a factor of $\sqrt{n}$. **Noise reduction and consequently better class separation may be obtained by adding variables that are presumably redundant.** Variables that are independently and identically distributed are not truly redundant.

3.2 How Does Correlation Impact Variable Redundancy?

Another notion of redundancy is correlation. In the previous example, in spite of the fact that the examples are i.i.d. with respect to the class conditional distributions, the variables are correlated because of the separation of the class center positions. One may wonder how variable redundancy is affected by adding within-class variable correlation. In Figure 2, the class centers are positioned

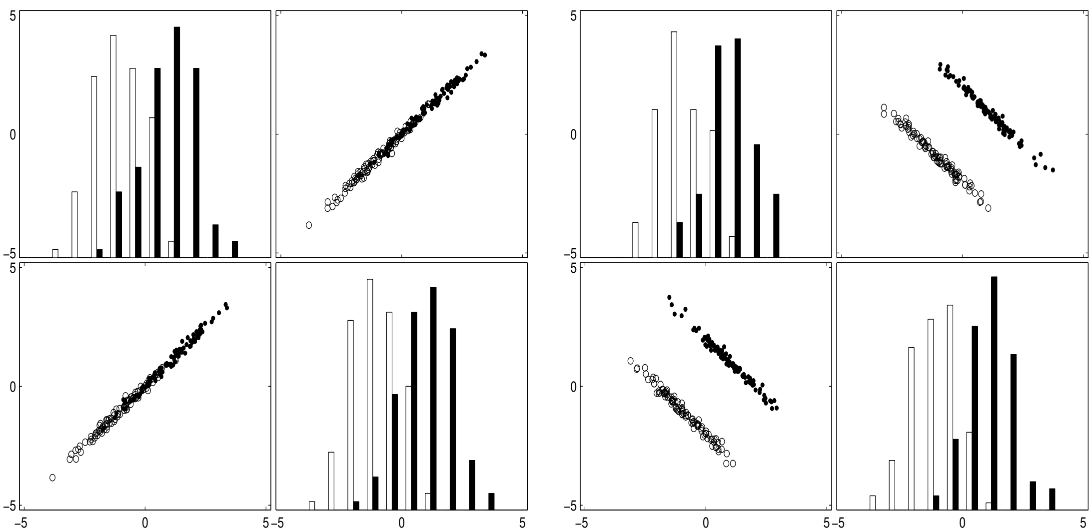


Figure 2: **Intra-class covariance.** In projection on the axes, the distributions of the two variables are the same as in the previous example. (a) The class conditional distributions have a high covariance in the direction of the line of the two class centers. There is no significant gain in separation by using two variables instead of just one. (b) The class conditional distributions have a high covariance in the direction perpendicular to the line of the two class centers. An important separation gain is obtained by using two variables instead of one.

similarly as in the previous example at coordinates (-1; -1) and (1; 1) but we have added some variable co-variance. We consider two cases:

In Figure 2.a, in the direction of the class center line, the standard deviation of the class conditional distributions is $\sqrt{2}$, while in the perpendicular direction it is a small value ($\varepsilon = 1/10$). With this construction, as ε goes to zero, the input variables have the same separation power as in the case of the example of Figure 1, with a standard deviation of the class distributions of one and a distance of the class centers of 2. But the feature constructed as the sum of the input variables has no better separation power: a standard deviation of $\sqrt{2}$ and a class center separation of $2\sqrt{2}$ (a simple scaling that does not change the separation power). Therefore, in the limit of perfect variable correlation (zero variance in the direction perpendicular to the class center line), single variables provide the same separation as the sum of the two variables. **Perfectly correlated variables are truly redundant in the sense that no additional information is gained by adding them.**

In contrast, in the example of Figure 2.b, the first principal direction of the covariance matrices of the class conditional densities is perpendicular to the class center line. In this case, more is gained by adding the two variables than in the example of Figure 1. One notices that in spite of their great complementarity (in the sense that a perfect separation can be achieved in the two-dimensional space spanned by the two variables), the two variables are (anti-)correlated. More anti-correlation is obtained by making the class centers closer and increasing the ratio of the variances of the class conditional distributions. **Very high variable correlation (or anti-correlation) does not mean absence of variable complementarity.**

The examples of Figure 1 and 2 all have variables with the same distribution of examples (in projection on the axis). Therefore, methods that score variables individually and independently of each other are at loss to determine which combination of variables would give best performance.

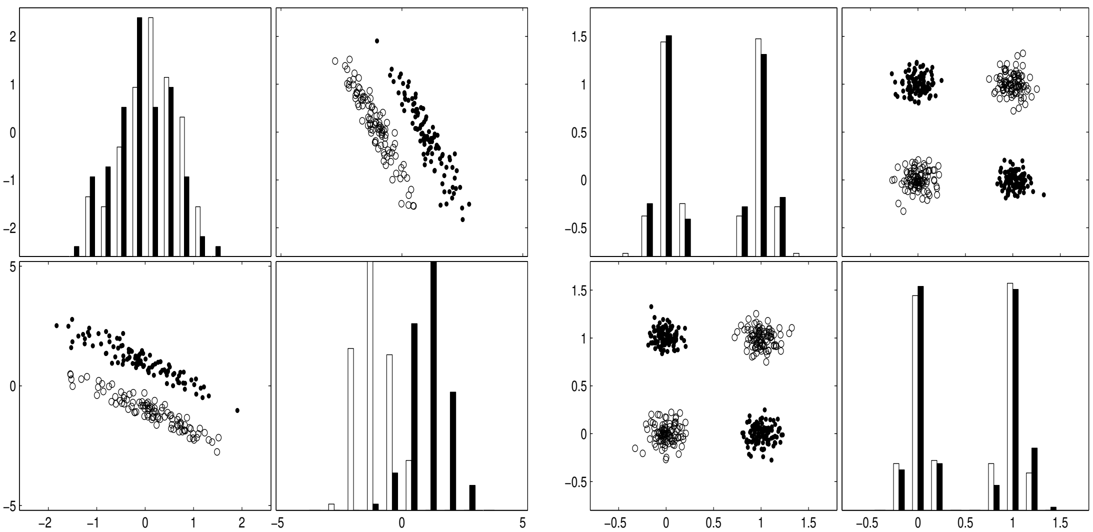
3.3 Can a Variable that is Useless by Itself be Useful with Others?

One concern about multivariate methods is that they are prone to overfitting. The problem is aggravated when the number of variables to select from is large compared to the number of examples. It is tempting to use a variable ranking method to filter out the least promising variables before using a multivariate method. Still one may wonder whether one could potentially lose some valuable variables through that filtering process.

We constructed an example in Figure 3.a. In this example, the two class conditional distributions have identical covariance matrices, and the principal directions are oriented diagonally. The class centers are separated on one axis, but not on the other. By itself one variable is "useless". Still, the two dimensional separation is better than the separation using the "useful" variable alone. Therefore, **a variable that is completely useless by itself can provide a significant performance improvement when taken with others.**

The next question is whether two variables that are useless by themselves can provide a good separation when taken together. We constructed an example of such a case, inspired by the famous XOR problem.[7] In Figure 3.b, we drew examples for two classes using four Gaussians placed on the corners of a square at coordinates (0; 0), (0; 1), (1; 0), and (1; 1). The class labels of these four "clumps" are attributed according to the truth table of the logical XOR function: f(0; 0)=0, f(0; 1)=1, f(1; 0)=1; f(1; 1)=0. We notice that the projections on the axes provide no class separation. Yet, in the two dimensional space the classes can easily be separated (albeit not with a linear decision function).[8] **Two variables that are useless by themselves can be useful together.**

7. The XOR problem is sometimes referred to as the two-bit parity problem and is generalizable to more than two dimensions (n-bit parity problem). A related problem is the chessboard problem in which the two classes pave the space with squares of uniformly distributed examples with alternating class labels. The latter problem is also generalizable to the multi-dimensional case. Similar examples are used in several papers in this issue (Perkins et al., 2003, Stoppiglia et al., 2003).
8. Incidentally, the two variables are also uncorrelated with one another.



(a) (b)

Figure 3: **A variable useless by itself can be useful together with others.** (a) One variable has completely overlapping class conditional densities. Still, using it jointly with the other variable improves class separability compared to using the other variable alone. (b) XOR-like or chessboard-like problems. The classes consist of disjoint clumps such that in projection on the axes the class conditional densities overlap perfectly. Therefore, individual variables have no separation power. Still, taken together, the variables provide good class separability .

4 Variable Subset Selection

In the previous section, we presented examples that illustrate the usefulness of selecting subsets of variables that together have good predictive power, as opposed to ranking variables according to their individual predictive power. We now turn to this problem and outline the main directions that have been taken to tackle it. They essentially divide into wrappers, filters, and embedded methods. **Wrappers** utilize the learning machine of interest as a black box to score subsets of variable according to their predictive power. **Filters** select subsets of variables as a pre-processing step, independently of the chosen predictor. **Embedded** methods perform variable selection in the process of training and are usually specific to given learning machines.

4.1 Wrappers and Embedded Methods

The wrapper methodology, recently popularized by Kohavi and John (1997), offers a simple and powerful way to address the problem of variable selection, regardless of the chosen learning machine. In fact, the learning machine is considered a perfect black box and the method lends itself to the use of off-the-shelf machine learning software packages. In its most general formulation, the wrapper methodology consists in using the prediction performance of a given learning machine to assess the relative usefulness of subsets of variables. In practice, one needs to define: (i) how to search the space of all possible variable subsets; (ii) how to assess the prediction performance of a learning machine to guide the search and halt it; and (iii) which predictor to use. An exhaustive search can conceivably be performed, if the number of variables is not too large. But, the problem is known to be NP-hard (Amaldi and Kann, 1998) and the search becomes quickly computationally intractable. A wide range of search strategies can be used, including best-first, branch-and-bound, simulated annealing, genetic algorithms (see Kohavi and John, 1997, for a review). Performance assessments are usually done using a validation set or by cross-validation (see Section 6). As illustrated in this special issue, popular predictors include decision trees, naïve Bayes, least-square linear predictors, and support vector machines.

Wrappers are often criticized because they seem to be a "brute force" method requiring massive amounts of computation, but it is not necessarily so. Efficient search strategies may be devised. Using such strategies does not necessarily mean sacrificing prediction performance. In fact, it appears to be the converse in some cases: coarse search strategies may alleviate the problem of overfitting, as illustrated for instance in this issue by the work of Reunanen (2003). Greedy search strategies seem to be particularly computationally advantageous and robust against overfitting. They come in two flavors: *forward selection* and *backward elimination*. In forward selection, variables are progressively incorporated into larger and larger subsets, whereas in backward elimination one starts with the set of all variables and progressively eliminates the least promising ones.[9] Both methods yield *nested subsets* of variables.

By using the learning machine as a black box, wrappers are remarkably universal and simple. But embedded methods that incorporate variable selection as part of the training process may be more efficient in several respects: they make better use of the available data by not needing to split the training data into a training and validation set; they reach a solution faster by avoiding retraining a predictor from scratch for every variable subset investigated. Embedded methods are not new: decision trees such as CART, for instance, have a built-in mechanism to perform variable selection (Breiman et al., 1984). The next two sections are devoted to two families of embedded methods illustrated by algorithms published in this issue.

4.2 Nested Subset Methods

Some embedded methods guide their search by estimating changes in the objective function value incurred by making moves in variable subset space. Combined with greedy search strategies (backward elimination or forward selection) they yield nested subsets of variables.[10]

Let us call s the number of variables selected at a given algorithm step and $J(s)$ the value of the objective function of the trained learning machine using such a variable subset. Predicting the change in the objective function is obtained by:

1. **Finite difference calculation:** The difference between $J(s)$ and $J(s+1)$ or $J(s-1)$ is computed for the variables that are candidates for addition or removal.

2. **Quadratic approximation of the cost function:** This method was originally proposed to prune weights in neural networks (LeCun et al., 1990). It can be used for backward elimination of variables, via the pruning of the input variable weights w_i. A second order Taylor expansion of J is made. At the optimum of J, the first-order term can be neglected, yield-

9. The name greedy comes from the fact that one never revisits former decisions to include (or exclude) variables in light of new decisions.
10. The algorithms presented in this section and in the following generally benefit from variable normalization, except if they have an internal normalization mechanism like the Gram-Schmidt orthogonalization procedure .

ing for variable i to the variation $DJ_i = (1/2)\frac{\partial^2 J}{\partial w_i^2}(Dw_i)^2$. The change in weight $Dw_i = w_i$ corresponds to removing variable i.

3. **Sensitivity of the objective function calculation:** The absolute value or the square of the derivative of J with respect to x_i (or with respect to w_i) is used.

Some training algorithms lend themselves to using finite differences (method 1) because exact differences can be computed efficiently, without retraining new models for each candidate variable. Such is the case for the linear least-square model: The Gram-Schmidt orthogonolization procedure permits the performance of forward variable selection by adding at each step the variable that most decreases the mean-squared-error. Two papers in this issue are devoted to this technique (Stoppiglia et al., 2003, Rivals and Personnaz, 2003). For other algorithms like kernel methods, approximations of the difference can be computed efficiently. Kernel methods are learning machines of the form $f(\mathbf{x}) = \sum_{k=1}^{m} \alpha_k K(\mathbf{x}, \mathbf{x}_k)$, where K is the kernel function, which measures the similarity between $\mathbf{x}$ and $\mathbf{x}_k$ (Schoelkopf and Smola, 2002). The variation in $J(s)$ is computed by keeping the α_k values constant. This procedure originally proposed for SVMs (Guyon et al., 2002) is used in this issue as a baseline method (Rakotomamonjy, 2003, Weston et al., 2003).

The "optimum brain damage" (OBD) procedure (method 2) is mentioned in this issue in the paper of Rivals and Personnaz (2003). The case of linear predictors $f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b$ is particularly simple. The authors of the OBD algorithm advocate using DJ_i instead of the magnitude of the weights $|w_i|$ as pruning criterion. However, for linear predictors trained with an objective function J that is quadratic in w_i these two criteria are equivalent. This is the case, for instance, for the linear least square model using $J = \sum_{k=1}^{m} (\mathbf{w} \cdot \mathbf{x_k} + b - y_k)^2$ and for the linear SVM or optimum margin classifier, which minimizes $J = (1/2)||w||^2$, under constraints (Vapnik, 1982). Interestingly, for linear SVMs the finite difference method (method 1) and the sensitivity method (method 3) also boil down to selecting the variable with smallest $|w_i|$ for elimination at each step (Rakotomamonjy, 2003).

The sensitivity of the objective function to changes in w_i (method 3) is used to devise a forward selection procedure in one paper presented in this issue (Perkins et al., 2003). Applications of this procedure to a linear model with a cross-entropy objective function are presented. In the formulation proposed, the criterion is the absolute value of $\frac{\partial J}{\partial w_i} = \sum_{k=1}^{m} \frac{\partial J}{\partial \rho_k} \frac{\partial \rho_k}{\partial w_i}$, where $\rho_k = y_k f(\mathbf{x_k})$. In the case of the linear model $f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b$, the criterion has a simple geometrical interpretation: it is the the dot product between the gradient of the objective function with respect to the margin values and the vector $[\frac{\partial \rho_k}{\partial w_i} = x_{k,i} y_k]_{k=1...m}$. For the cross-entropy loss function, we have: $\frac{\partial J}{\partial \rho_k} = \frac{1}{1+e^{\rho_k}}$.

An interesting variant of the sensitivity analysis method is obtained by replacing the objective function by the *leave-one-out* cross-validation error. For some learning machines and some objective functions, approximate or exact analytical formulas of the leave-one-out error are known. In this issue, the case of the linear least-square model (Rivals and Personnaz, 2003) and SVMs (Rakotomamonjy, 2003) are treated. Approximations for non-linear least-squares have also been computed elsewhere (Monari and Dreyfus, 2000). The proposal of Rakotomamonjy (2003) is to train non-linear SVMs (Boser et al., 1992, Vapnik, 1998) with a regular training procedure and select features with backward elimination like in RFE (Guyon et al., 2002). The variable ranking criterion however is not computed using the sensitivity of the objective function J, but that of a leave-one-out bound.

4.3 Direct Objective Optimization

A lot of progress has been made in this issue to formalize the objective function of variable selection and find algorithms to optimize it. Generally, the objective function consists of two terms that compete with each other: (1) the **goodness-of-fit** (to be maximized), and (2) the **number of variables** (to be minimized). This approach bears similarity with two-part objective functions consisting of a goodness-of-fit term and a regularization term, particularly when the effect of the regularization term is to "shrink" parameter space. This correspondence is formally established in the paper of Weston et al. (2003) for the particular case of classification with linear predictors $f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b$, in the SVM framework (Boser et al., 1992, Vapnik, 1998). Shrinking regularizers of the type $||\mathbf{w}||_p^p = (\sum_{i=1}^n w_i^p)^{1/p}$ (ℓ_p-norm) are used. In the limit as $p \to 0$, the ℓ_p-norm is just the number of weights, i.e., the number of variables. Weston et al. proceed with showing that the ℓ_0-norm formulation of SVMs can be solved approximately with a simple modification of the vanilla SVM algorithm:

1. Train a regular linear SVM (using ℓ_1-norm or ℓ_2-norm regularization).

2. Re-scale the input variables by multiplying them by the absolute values of the components of the weight vector $\mathbf{w}$ obtained.

3. Iterate the first 2 steps until convergence.

The method is reminiscent of backward elimination procedures based on the smallest $|w_i|$. Variable normalization is important for such a method to work properly.

Weston et al. note that, although their algorithm only approximately minimizes the ℓ_0-norm, in practice it may generalize better than an algorithm that really did minimize the ℓ_0-norm, because the latter would not provide sufficient regularization (a lot of variance remains because the optimization problem has multiple solutions). The need for additional regularization is also stressed in the paper of Perkins et al. (2003). The authors use a three-part objective function that includes goodness-of-fit, a regularization term (ℓ_1-norm or ℓ_2-norm), and a penalty for large numbers of variables (ℓ_0-norm). The authors propose a computationally efficient forward selection method to optimize such objective.

Another paper in the issue, by Bi et al. (2003), uses ℓ_1-norm SVMs, without iterative multiplicative updates. The authors find that, for their application, the ℓ_1-norm minimization suffices to drive enough weights to zero. This approach was also taken in the context of least-square regression by other authors (Tibshirani, 1994). The number of variables can be further reduced by backward elimination.

To our knowledge, no algorithm has been proposed to directly minimize the number of variables for non-linear predictors. Instead, several authors have substituted for the problem of variable selection that of variable scaling (Jebara and Jaakkola, 2000, Weston et al., 2000, Grandvalet and Canu, 2002). The variable scaling factors are "hyper-parameters" adjusted by model selection. The scaling factors obtained are used to assess variable relevance. A variant of the method consists of adjusting the scaling factors by gradient descent on a bound of the leave-one-out error (Weston et al., 2000). This method is used as baseline method in the paper of Weston et al. (2003) in this issue.

4.4 Filters for Subset Selection

Several justifications for the use of filters for subset selection have been put forward in this special issue and elsewhere. It is argued that, compared to wrappers, filters are faster. Still, recently proposed efficient embedded methods are competitive in that respect. Another argument is that some filters (e.g. those based on mutual information criteria) provide a generic selection of variables, not tuned for/by a given learning machine. Another compelling justification is that filtering can be used as a preprocessing step to reduce space dimensionality and overcome overfitting.

In that respect, it seems reasonable to use a wrapper (or embedded method) with a *linear* predictor as a filter and then train a more complex *non-linear* predictor on the resulting variables. An example of this approach is found in the paper of Bi et al. (2003): a linear ℓ_1-norm SVM is used for variable selection, but a non-linear ℓ_1-norm SVM is used for prediction. The complexity of linear filters can be ramped up by adding to the selection process products of input variables (monomials of a polynomial) and retaining the variables that are part of any selected monomial. Another predictor, e.g., a neural network, is eventually substituted to the polynomial to perform predictions using the selected variables (Rivals and Personnaz, 2003, Stoppiglia et al., 2003). In some cases however, one may on the contrary want to reduce the complexity of linear filters to overcome overfitting problems. When the number of examples is small compared to the number of variables (in the case of microarray data for instance) one may need to resort to selecting variables with correlation coefficients (see Section 2.2).

Information theoretic filtering methods such as Markov blanket[11] algorithms (Koller and Sahami, 1996) constitute another broad family. The justification for classification problems is that the measure of mutual information does not rely on any prediction process, but provides a bound on the error rate using any prediction scheme for the given distribution. We do not have any illustration of such methods in this issue for the problem of variable subset selection. We refer the interested reader to Koller and Sahami (1996) and references therein. However, the use of mutual information criteria for individual variable ranking was covered in Section 2 and application to feature construction and selection are illustrated in Section 5.

5 Feature Construction and Space Dimensionality Reduction

In some applications, reducing the dimensionality of the data by selecting a subset of the original variables may be advantageous for reasons including the expense of making, storing and processing measurements. If these considerations are not of concern, other means of space dimensionality reduction should also be considered.

The art of machine learning starts with the design of appropriate data representations. Better performance is often achieved using features derived from the original input. Building a feature representation is an opportunity to incorporate domain knowledge into the data and can be very application specific. Nonetheless, there are a number of generic feature construction methods, including: clustering; basic linear transforms of the input variables (PCA/SVD, LDA); more sophisticated linear transforms like spectral transforms (Fourier, Hadamard), wavelet transforms or convolutions of kernels; and applying simple functions to subsets of variables, like products to create monomials.

11. The Markov blanket of a given variable x_i is a set of variables not including x_i that render x_i "unnecessary". Once a Markov blanket is found, x_i can safely be eliminated. Furthermore, in a backward elimination procedure, it will remain unnecessary at later stages.

Two distinct goals may be pursued for feature construction: achieving best reconstruction of the data or being most efficient for making predictions. The first problem is an unsupervised learning problem. It is closely related to that of data compression and a lot of algorithms are used across both fields. The second problem is supervised. Are there reasons to select features in an unsupervised manner when the problem is supervised? Yes, possibly several: Some problems, e.g., in text processing applications, come with more unlabelled data than labelled data. Also, unsupervised feature selection is less prone to overfitting.

In this issue, four papers address the problem of feature construction. All of them take an information theoretic approach to the problem. Two of them illustrate the use of clustering to construct features (Bekkerman et al., 2003, Dhillon et al., 2003), one provides a new matrix factorization algorithm (Globerson and Tishby, 2003), and one provides a supervised means of learning features from a variety of models (Torkkola, 2003). In addition, two papers whose main focus is directed to variable selection also address the selection of monomials of a polynomial model and the hidden units of a neural network (Rivals and Personnaz, 2003, Stoppiglia et al., 2003), and one paper addresses the implicit feature selection in non-linear kernel methods for polynomial kernels (Weston et al., 2003).

5.1 Clustering

Clustering has long been used for feature construction. The idea is to replace a group of "similar" variables by a cluster centroid, which becomes a feature. The most popular algorithms include K-means and hierarchical clustering. For a review, see, e.g., the textbook of Duda et al. (2001).

Clustering is usually associated with the idea of unsupervised learning. It can be useful to introduce some supervision in the clustering procedure to obtain more discriminant features. This is the idea of distributional clustering (Pereira et al., 1993), which is developed in two papers of this issue. Distributional clustering is rooted in the information bottleneck (IB) theory of Tishby et al. (1999). If we call $\tilde{X}$ the random variable representing the constructed features, the IB method seeks to minimize the mutual information $I(X,\tilde{X})$, while preserving the mutual information $I(\tilde{X},Y)$. A global objective function is built by introducing a Lagrange multiplier β: $J = I(X,\tilde{X}) - \beta I(\tilde{X},Y)$. So, the method searches for the solution that achieves the largest possible compression, while retaining the essential information about the target.

Text processing applications are usual targets for such techniques. Patterns are full documents and variables come from a bag-of-words representation: Each variable is associated to a word and is proportional to the fraction of documents in which that word appears. In application to feature construction, clustering methods group words, not documents. In text categorization tasks, the supervision comes from the knowledge of document categories. It is introduced by replacing variable vectors containing document frequency counts by shorter variable vectors containing document category frequency counts, i.e., the words are represented as distributions over document categories.

The simplest implementation of this idea is presented in the paper of Dhillon et al. (2003) in this issue. It uses K-means clustering on variables represented by a vector of document category frequency counts. The (non-symmetric) similarity measure used is the Kullback-Leibler divergence $K(\mathbf{x}_j,\tilde{\mathbf{x}}_i) = exp(-\beta \sum_k x_{k,j} ln(x_{k,j}/\tilde{x}_{k,i}))$. In the sum, the index k runs over document categories. A more elaborate approach is taken by Bekkerman et al. (2003) who use a "soft" version of K-means (allowing words to belong to several clusters) and who progressively divide clusters by varying the Lagrange multiplier β monitoring the tradeoff between $I(X,\tilde{X})$ and $I(\tilde{X},Y)$. In this way, documents are represented as a distribution over word centroids. Both methods perform well. Bekkerman et al. mention that few words end up belonging to several clusters, hinting that "hard" cluster assignment may be sufficient.

5.2 Matrix Factorization

Another widely used method of feature construction is singular value decomposition (SVD). The goal of SVD is to form a set of features that are linear combinations of the original variables, which provide the best possible reconstruction of the original data in the least square sense (Duda et al., 2001). It is an unsupervised method of feature construction. In this issue, the paper of Globerson and Tishby (2003) presents an information theoretic unsupervised feature construction method: sufficient dimensionality reduction (SDR). The most informative features are extracted by solving an optimization problem that monitors the tradeoff between data reconstruction and data compression, similar to the information bottleneck of Tishby et al. (1999); the features are found as Lagrange multipliers of the objective optimized. Non-negative matrices P of dimension (m, n) representing the joint distribution of two random variables (for instance the co-occurrence of words in documents) are considered. The features are extracted by information theoretic I-projections, yielding a reconstructed matrix of special exponential form $\tilde{P} = (1/Z)\exp(\Phi\Psi)$. For a set of d features, Φ is a $(m, d+2)$ matrix whose $(d+1)^{th}$ column is ones and Ψ is a $(d+2, n)$ matrix whose $(d+2)^{th}$ column is ones, and Z is a normalization coefficient. Similarly to SVD, the solution shows the symmetry of the problem with respect to patterns and variables.

5.3 Supervised Feature Selection

We review three approaches for selecting features in cases where features should be distinguished from variables because both appear simultaneously in the same system:

Nested subset methods. A number of learning machines extract features as part of the learning process. These include neural networks whose internal nodes are feature extractors. Thus, node pruning techniques such as OBD LeCun et al. (1990) are feature selection algorithms. Gram-Schmidt orthogonalization is presented in this issue as an alternative to OBD (Stoppiglia et al., 2003).

Filters. Torkkola (2003) proposes a filter method for constructing features using a mutual information criterion. The author maximizes $I(\phi, \mathbf{y})$ for m dimensional feature vectors ϕ and target vectors $\mathbf{y}$.[12] Modelling the feature density function with Parzen windows allows him to compute derivatives $\partial I/\partial\phi_i$ that are transform independent. Combining them with the transform-dependent derivatives $\partial\phi_i/\partial\mathbf{w}$, he devises a gradient descent algorithm to optimize the parameters $\mathbf{w}$ of the transform (that need not be linear):

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \eta\frac{\partial I}{\partial \mathbf{w}} = \mathbf{w}_t + \eta\frac{\partial I}{\partial \phi_i}\frac{\partial \phi_i}{\partial \mathbf{w}} \,. \tag{5}$$

Direct objective optimization. Kernel methods possess an implicit feature space revealed by the kernel expansion: $k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x}).\phi(\mathbf{x}')$, where $\phi(\mathbf{x})$ is a feature vector of possibly infinite dimension. Selecting these implicit features may improve generalization, but does not change the

12. In fact, the author uses a quadratic measure of divergence instead of the usual mutual information.

running time or help interpreting the prediction function. In this issue, Weston et al. (2003) propose a method for selecting implicit kernel features in the case of the polynomial kernel, using their framework of minimization of the ℓ_0-norm.

6 Validation Methods

We group in this section all the issues related to out-of-sample performance prediction (generalization prediction) and model selection. These are involved in various aspects of variable and feature selection: to determine the number of variables that are "significant", to guide and halt the search for good variable subsets, to choose hyperparameters, and to evaluate the final performance of the system.

One should first distinguish the problem of model selection from that of evaluating the final performance of the predictor. For that last purpose, it is important to set aside an independent test set. The remaining data is used both for training and performing model selection. Additional experimental sophistication can be added by repeating the entire experiment for several drawings of the test set.[13]

To perform model selection (including variable/feature selection and hyperparameter optimization), the data not used for testing may be further split between fixed training and validation sets, or various methods of cross-validation can be used. The problem is then brought back to that of estimating the significance of differences in validation errors. For a fixed validation set, statistical tests can be used, but their validity is doubtful for cross-validation because independence assumptions are violated. For a discussion of these issues, see for instance the work of Dietterich (1998) and Nadeau and Bengio (2001). If there are sufficiently many examples, it may not be necessary to split the training data: Comparisons of training errors with statistical tests can be used (see Rivals and Personnaz, 2003, in this issue). Cross-validation can be extended to time-series data and, while i.i.d. assumptions do not hold anymore, it is still possible to estimate generalization error confidence intervals (see Bengio and Chapados, 2003, in this issue).

Choosing what fraction of the data should be used for training and for validation is an open problem. Many authors resort to using the leave-one-out cross-validation procedure, even though it is known to be a high variance estimator of generalization error (Vapnik, 1982) and to give overly optimistic results, particularly when data are not properly independently and identically sampled from the "true" distribution. The leave-one-out procedure consists of removing one example from the training set, constructing the predictor on the basis only of the remaining training data, then testing on the removed example. In this fashion one tests all examples of the training data and averages the results. As previously mentioned, there exist exact or approximate formulas of the leave-one-out error for a number of learning machines (Monari and Dreyfus, 2000, Rivals and Personnaz, 2003, Rakotomamonjy, 2003).

Leave-one-out formulas can be viewed as corrected values of the training error. Many other types of penalization of the training error have been proposed in the literature (see, e.g., Vapnik, 1998, Hastie et al., 2001). Recently, a new family of such methods called "metric-based methods" have been proposed (Schuurmans, 1997). The paper of Bengio and Chapados (2003) in this issue

13. In the limit, the test set can have only one example and leave-out-out can be carried out as an "outer loop", outside the feature/variable selection process, to estimate the final performance of the predictor. This computationally expensive procedure is used in cases where data is extremely scarce.

illustrates their application to variable selection. The authors make use of unlabelled data, which are readily available in the application considered, time series prediction with a horizon. Consider two models f_A and f_B trained with nested subsets of variables $A \subset B$. We call $d(f_A, f_B)$ the discrepancy of the two models. The criterion involves the ratio $d_U(f_A, f_B)/d_T(f_A, f_B)$, where $d_U(f_A, f_B)$ is computed with unlabelled data and $d_T(f_A, f_B)$ is computed with training data. A ratio significantly larger than one sheds doubt on the usefulness of the variables in subset B that are not in A.

For variable ranking or nested subset ranking methods (Sections 2 and 4.2), another statistical approach can be taken. The idea is to introduce a probe in the data that is a random variable. Roughly speaking, variables that have a relevance smaller or equal to that of the probe should be discarded. Bi et al. (2003) consider a very simple implementation of that idea: they introduce in their data three additional "fake variables" drawn randomly from a Gaussian distribution and submit them to their variable selection process with the other "true variables". Subsequently, they discard all the variables that are less relevant than one of the three fake variables (according to their weight magnitude criterion). Stoppiglia et al. (2003) propose a more sophisticated method for the Gram-Schmidt forward selection method. For a Gaussian distributed probe, they provide an analytical formula to compute the rank of the probe associated with a given risk of accepting an irrelevant variable. A non-parametric variant of the probe method consists in creating "fake variables" by randomly shuffling real variable vectors. In a forward selection process, the introduction of fake variables does not disturb the selection because fake variables can be discarded when they are encountered. At a given step in the forward selection process, let us call f_t the fraction of true variables selected so far (among all true variables) and f_f the fraction of fake variables encountered (among all fake variables). As a halting criterion one can place a threshold on the ratio f_f/f_t, which is an upper bound on the fraction of falsely relevant variables in the subset selected so far. The latter method has been used for variable ranking (Tusher et al., 2001). Its parametric version for Gaussian distributions using the T statistic as ranking criterion is nothing but the T-test.

7 Advanced Topics and Open Problems

7.1 Variance of Variable Subset Selection

Many methods of variable subset selection are sensitive to small perturbations of the experimental conditions. If the data has redundant variables, different subsets of variables with identical predictive power may be obtained according to initial conditions of the algorithm, removal or addition of a few variables or training examples, or addition of noise. For some applications, one might want to purposely generate alternative subsets that can be presented to a subsequent stage of processing. Still one might find this variance undesirable because (i) variance is often the symptom of a "bad" model that does not generalize well; (ii) results are not reproducible; and (iii) one subset fails to capture the "whole picture".

One method to "stabilize" variable selection explored in this issue is to use several "bootstraps" (Bi et al., 2003). The variable selection process is repeated with sub-samples of the training data. The union of the subsets of variables selected in the various bootstraps is taken as the final "stable" subset. This joint subset may be at least as predictive as the best bootstrap subset. Analyzing the behavior of the variables across the various bootstraps also provides further insight, as described in the paper. In particular, an index of relevance of individual variables can be created considering how frequently they appear in the bootstraps.

Related ideas have been described elsewhere in the context of Bayesian variable selection (Jebara and Jaakkola, 2000, Ng and Jordan, 2001, Vehtari and Lampinen, 2002). A distribution over a population of models using various variable subsets is estimated. Variables are then ranked according to the marginal distribution, reflecting how often they appear in important subsets (i.e., associated with the most probable models).

7.2 Variable Ranking in the Context of Others

In Section 2, we limited ourselves to presenting variable ranking methods using a criterion computed from single variables, ignoring the context of others. In Section 4.2, we introduced nested subset methods that provide a useful ranking of subsets, not of individual variables: some variables may have a low rank because they are redundant and yet be highly relevant. Bootstrap and Bayesian methods presented in Section 7.1, may be instrumental in producing a good variable ranking incorporating the context of others.

The relief algorithm uses another approach based on the nearest-neighbor algorithm (Kira and Rendell, 1992). For each example, the closest example of the same class (nearest hit) and the closest example of a different class (nearest miss) are selected. The score $S(i)$ of the i^{th} variable is computed as the average over all examples of magnitude of the difference between the distance to the nearest hit and the distance to the nearest miss, in projection on the i^{th} variable.

7.3 Unsupervised Variable Selection

Sometimes, no target y is provided, but one still would want to select a set of most significant variables with respect to a defined criterion. Obviously, there are as many criteria as problems can be stated. Still, a number of variable ranking criteria are useful across applications, including *saliency*, *entropy*, *smoothness*, *density* and *reliability*. A variable is salient if it has a high variance or a large range, compared to others. A variable has a high entropy if the distribution of examples is uniform. In a time series, a variable is smooth if on average its local curvature is moderate. A variable is in a high-density region if it is highly correlated with many other variables. A variable is reliable if the measurement error bars computed by repeating measurements are small compared to the variability of the variable values (as quantified, e.g., by an ANOVA statistic).

Several authors have also attempted to perform variable or feature selection for clustering applications (see, e.g., Xing and Karp, 2001, Ben-Hur and Guyon, 2003, and references therein).

7.4 Forward *vs.* Backward Selection

It is often argued that forward selection is computationally more efficient than backward elimination to generate nested subsets of variables. However, the defenders of backward elimination argue that weaker subsets are found by forward selection because the importance of variables is not assessed in the context of other variables not included yet. We illustrate this latter argument by the example of Figure 4. In that example, one variable separates the two classes better by itself than either of the two other ones taken alone and will therefore be selected first by forward selection. At the next step, when it is complemented by either of the two other variables, the resulting class separation in two dimensions will not be as good as the one obtained jointly by the two variables that were discarded at the first step. A backward selection method may outsmart forward selection by eliminating at the first step the variable that by itself provides the best separation to retain the two variables that

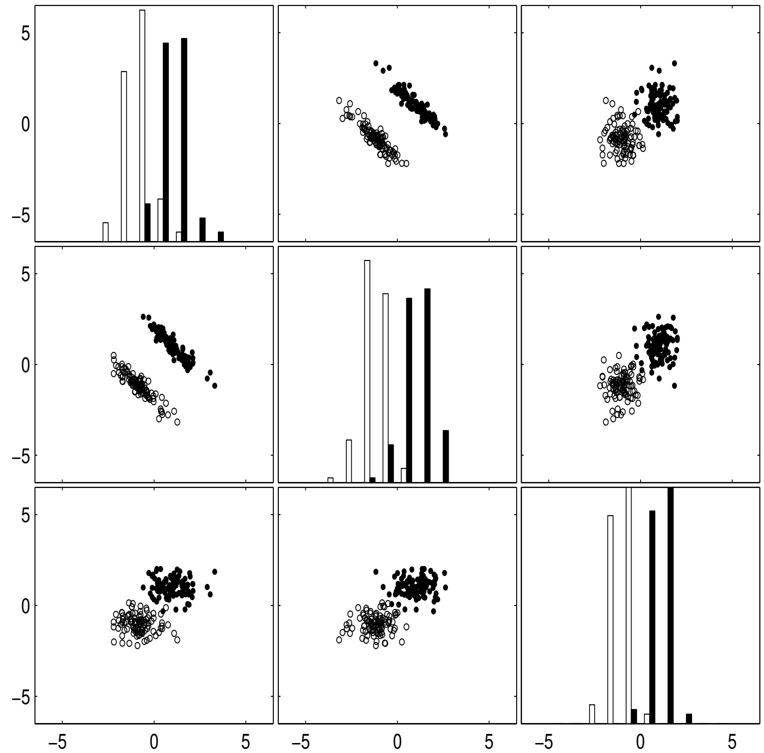


Figure 4: **Forward or backward selection?** Of the three variables of this example, the third one separates the two classes best by itself (bottom right histogram). It is therefore the best candidate in a forward selection process. Still, the two other variables are better taken together than any subset of two including it. A backward selection method may perform better in this case.

together perform best. Still, if for some reason we need to get down to a single variable, backward elimination will have gotten rid of the variable that works best on its own.

7.5 The Multi-class Problem

Some variable selection methods treat the multi-class case directly rather than decomposing it into several two-class problems: All the methods based on mutual information criteria extend naturally to the multi-class case (see in this issue Bekkerman et al., 2003, Dhillon et al., 2003, Torkkola, 2003). Multi-class variable ranking criteria include Fisher's criterion (the ratio of the between class variance to the within-class variance). It is closely related to the F statistic used in the ANOVA test, which is one way of implementing the probe method (Section 6) for the multi-class case. Wrappers or embedded methods depend upon the capability of the classifier used to handle the multi-class case. Examples of such classifiers include linear discriminant analysis (LDA), a multi-class version of Fisher's linear discriminant (Duda et al., 2001), and multi-class SVMs (see, e.g., Weston et al., 2003).

One may wonder whether it is advantageous to use multi-class methods for variable selection. On one hand, contrary to what is generally admitted for classification, the multi-class setting is in some sense easier for variable selection than the two-class case. This is because the larger the number of classes, the less likely a "random" set of features provide a good separation. To illustrate this point, consider a simple example where all features are drawn independently from the same distribution P and the first of them is the target y. Assume that all these features correspond to rolling a die with Q faces n times (n is the number of samples). The probability that one fixed feature (except the first one) is exactly y is then $(1/Q)^n$. Therefore, finding the feature that corresponds to the target y when it is embedded in a sea of noisy features is easier when Q is large. On the other hand, Forman (2003) points out in this issue that in the case of uneven distributions across classes, multi-class methods may over-represent abundant or easily separable classes. A possible alternative is to mix ranked lists of several two-class problems. Weston et al. (2003) propose one such mixing strategy.

7.6 Selection of Examples

The dual problems of feature selection/construction are those of pattern selection/construction. The symmetry of the two problems is made explicit in the paper of Globerson and Tishby (2003) in this issue. Likewise, both Stoppiglia et al. (2003) and Weston et al. (2003) point out that their algorithm also applies to the selection of examples in kernel methods. Others have already pointed out the similarity and complementarity of the two problems (Blum and Langley, 1997). In particular, mislabeled examples may induce the choice of wrong variables. Conversely, if the labeling is highly reliable, selecting wrong variables associated with a confounding factor may be avoided by focusing on informative patterns that are close to the decision boundary (Guyon et al., 2002).

7.7 Inverse Problems

Most of the special issue concentrates on the problem of finding a (small) subset of variables useful to build a good predictor. In some applications, particularly in bioinformatics, this is not necessarily the only goal of variable selection. In diagnosis problems, for instance, it is important to identify the factors that triggered a particular disease or unravel the chain of events from the causes to the symptoms. But reverse engineering the system that produced the data is a more challenging task than building a predictor. The readers interested in these issues can consult the literature on gene networks in the conference proceedings of the pacific symposium on biocomputing (PSB) or intelligent systems for molecular biology conference (ISMB) and the causality inference literature (see, e.g., Pearl, 2000). At the heart of this problem is the distinction between correlation and causality. Observational data such as the data available to machine learning researchers allow us only to observe correlations. For example, observations can be made about correlations between expression profiles of given genes or between profiles and symptoms, but a leap of faith is made when deciding which gene activated which other one and in turn triggered the symptom.

In this issue, the paper of Caruana and de Sa (2003) presents interesting ideas about using variables discarded by variable selection as additional outputs of a neural network. They show improved performance on synthetic and real data. Their analysis supports the idea that some variables are more efficiently used as outputs than as inputs. This could be a step toward distinguishing causes from consequences.

Data set	Description	patterns	variables	classes	References
Linear[a,b]	Artificial linear	10-1200	100-240	reg-2	SWBe
Multi-cluster[c]	Artificial non-linear	1000-1300	100-500	2	PS
QSAR[d]	Chemistry	30-300	500-700	reg	Bt
UCI[e]	ML repository	8-60	500-16000	2-30	ReBnToPC
LVQ-PAK[f]	Phoneme data	1900	20	20	T
Raetch bench.[g]	UCI/Delve/Statlog	200-7000	8-20	2	Ra
Microarray[a]	Cancer classif.	6-100	2000-4000	2	WRa
Microarray[a]	Gene classification	200	80	5	W
Aston Univ[h]	Pipeline transport	1000	12	3	T
NIPS 2000[i]	Unlabeled data	200-400	5-800	reg	Ri
20 Newsgroup[j,o]	News postings	20000	300-15000	2-20	GBkD
Text filtering[k]	TREC/OSHUMED	200-2500	3000-30000	6-17	F
IR datasets [l]	MED/CRAN/CISI	1000	5000	30-225	G
Reuters-21578[m,o]	newswire docs.	21578	300-15000	114	BkF
Open Dir. Proj.[n]	Web directory	5000	14500	50	D

Table 1: **Publicly available data sets used in the special issue.** Approximate numbers or ranges of patterns, variables, and classes effectively used are provided. The "classes" column indicates "reg" for regression problems, or the number of queries for Information Retrieval (IR) problems. For artificial data sets, the fraction of variables that are relevant ranges from 2 to 10. The initial of the first author are provided as reference: Bk=Bekkerman, Bn=Bengio, Bt=Bennett, C=Caruana, D=Dhillon, F=Forman, G=Globerson, P=Perkins, Re=Reunanen, Ra=Rakotomamonjy, Ri=Rivals, S=Stoppiglia, T=Torkkola, W=Weston. Please also check the JMLR web site for later additions and preprocessed data.

a. http://www.kyb.tuebingen.mpg.de/bs/people/weston/l0 (ℓ_0 not 10)
b. http://www.clopinet.com/isabelle/Projects/NIPS2001/Artificial.zip
c. http://nis-www.lanl.gov/∼simes/data/jmlr02/
d. http://www.rpi.edu/∼bij2/featsele.html
e. http://www.ics.uci.edu/∼mlearn/MLRepository.html
f. http://www.cis.hut.fi/research/software.shtml
g. http://ida.first.gmd.de/∼raetsch/data/benchmarks.htm
h. http://www.nerg.aston.ac.uk/GTM/3PhaseData.html
i. http://q.cis.uoguelph.ca/ skremer/NIPS2000/
j. http://kdd.ics.uci.edu/databases/20newsgroups/20newsgroups.html
k. http://trec.nist.gov/data.html (Filtering Track Collection)
l. http://www.cs.utk.edu/∼lsi/
m. http://www.daviddlewis.com/resources/testcollections/reuters21578/
n. http://dmoz.org/ and http://www.cs.utexas.edu/users/manyam/dmoz.txt
o. http://www.cs.technion.ac.il/∼ronb/thesis.html

8 Conclusion

The recent developments in variable and feature selection have addressed the problem from the pragmatic point of view of improving the performance of predictors. They have met the challenge of operating on input spaces of several thousand variables. Sophisticated wrapper or embedded methods improve predictor performance compared to simpler variable ranking methods like correlation methods, but the improvements are not always significant: domains with large numbers of input variables suffer from the curse of dimensionality and multivariate methods may overfit the data. For some domains, applying first a method of automatic feature construction yields improved performance and a more compact set of features. The methods proposed in this special issue have been tested on a wide variety of data sets (see Table 1), which limits the possibility of making comparisons across papers. Further work includes the organization of a benchmark. The approaches are very diverse and motivated by various theoretical arguments, but a unifying theoretical framework is lacking. Because of these shortcomings, it is important when starting with a new problem to have a few baseline performance values. To that end, we recommend using a linear predictor of your choice (e.g. a linear SVM) and select variables in two alternate ways: (1) with a variable ranking method using a correlation coefficient or mutual information; (2) with a nested subset selection method performing forward or backward selection or with multiplicative updates. Further down the road, connections need to be made between the problems of variable and feature selection and those of experimental design and active learning, in an effort to move away from observational data toward experimental data, and to address problems of causality inference.

References

E. Amaldi and V. Kann. On the approximation of minimizing non zero variables or unsatisfied relations in linear systems. *Theoretical Computer Science*, 209:237–260, 1998.

R. Bekkerman, R. El-Yaniv, N. Tishby, and Y. Winter. Distributional word clusters vs. words for text categorization. *JMLR*, 3:1183–1208 (this issue), 2003.

A. Ben-Hur and I. Guyon. Detecting stable clusters using principal component analysis. In M.J. Brownstein and A. Kohodursky, editors, *Methods In Molecular Biology*, pages 159–182. Humana Press, 2003.

Y. Bengio and N. Chapados. Extensions to metric-based model selection. *JMLR*, 3:1209–1227 (this issue), 2003.

J. Bi, K. Bennett, M. Embrechts, C. Breneman, and M. Song. Dimensionality reduction via sparse support vector machines. *JMLR*, 3:1229–1243 (this issue), 2003.

A. Blum and P. Langley. Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 97(1-2):245–271, December 1997.

B. Boser, I. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. In *Fifth Annual Workshop on Computational Learning Theory*, pages 144–152, Pittsburgh, 1992. ACM.

L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth and Brooks, 1984.

R. Caruana and V. de Sa. Benefitting from the variables that variable selection discards. *JMLR*, 3: 1245–1264 (this issue), 2003.

I. Dhillon, S. Mallela, and R. Kumar. A divisive information-theoretic feature clustering algorithm for text classification. *JMLR*, 3:1265–1287 (this issue), 2003.

T. G. Dietterich. Approximate statistical test for comparing supervised classification learning algorithms. *Neural Computation*, 10(7):1895–1924, 1998.

R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. John Wiley & Sons, USA, 2nd edition, 2001.

T. R. Golub et al. Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. *Science*, 286:531–537, 1999.

G. Forman. An extensive empirical study of feature selection metrics for text classification. *JMLR*, 3:1289–1306 (this issue), 2003.

T. Furey, N. Cristianini, Duffy, Bednarski N., Schummer D., M., and D. Haussler. Support vector machine classification and validation of cancer tissue samples using microarray expression data. *Bioinformatics*, 16:906–914, 2000.

A. Globerson and N. Tishby. Sufficient dimensionality reduction. *JMLR*, 3:1307–1331 (this issue), 2003.

Y. Grandvalet and S. Canu. Adaptive scaling for feature selection in SVMs. In *NIPS 15*, 2002.

I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46(1-3):389–422, 2002.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer series in statistics. Springer, New York, 2001.

T. Jebara and T. Jaakkola. Feature selection and dualities in maximum entropy discrimination. In *16th Annual Conference on Uncertainty in Artificial Intelligence*, 2000.

K. Kira and L. Rendell. A practical approach to feature selection. In D. Sleeman and P. Edwards, editors, *International Conference on Machine Learning*, pages 368–377, Aberdeen, July 1992. Morgan Kaufmann.

R. Kohavi and G. John. Wrappers for feature selection. *Artificial Intelligence*, 97(1-2):273–324, December 1997.

D. Koller and M. Sahami. Toward optimal feature selection. In *13th International Conference on Machine Learning*, pages 284–292, July 1996.

Y. LeCun, J. Denker, S. Solla, R. E. Howard, and L. D. Jackel. Optimal brain damage. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems II*, San Mateo, CA, 1990. Morgan Kaufmann.

G. Monari and G. Dreyfus. Withdrawing an example from the training set: an analytic estimation of its effect on a nonlinear parameterized model. *Neurocomputing Letters*, 35:195–201, 2000.

C. Nadeau and Y. Bengio. Inference for the generalization error. *Machine Learning (to appear)*, 2001.

A. Y. Ng. On feature selection: learning with exponentially many irrelevant features as training examples. In *15th International Conference on Machine Learning*, pages 404–412. Morgan Kaufmann, San Francisco, CA, 1998.

A. Y. Ng and M. Jordan. Convergence rates of the voting Gibbs classifier, with application to Bayesian feature selection. In *18th International Conference on Machine Learning*, 2001.

J. Pearl. *Causality*. Cambridge University Press, 2000.

F. Pereira, N. Tishby, and L. Lee. Distributional clustering of English words. In *Proc. Meeting of the Association for Computational Linguistics*, pages 183–190, 1993.

S. Perkins, K. Lacker, and J. Theiler. Grafting: Fast incremental feature selection by gradient descent in function space. *JMLR*, 3:1333–1356 (this issue), 2003.

A. Rakotomamonjy. Variable selection using SVM-based criteria. *JMLR*, 3:1357–1370 (this issue), 2003.

J. Reunanen. Overfitting in making comparisons between variable selection methods. *JMLR*, 3: 1371–1382 (this issue), 2003.

I. Rivals and L. Personnaz. MLPs (mono-layer polynomials and multi-layer perceptrons) for nonlinear modeling. *JMLR*, 3:1383–1398 (this issue), 2003.

B. Schoelkopf and A. Smola. *Learning with Kernels*. MIT Press, Cambridge MA, 2002.

D. Schuurmans. A new metric-based approach to model selection. In *9th Innovative Applications of Artificial Intelligence Conference*, pages 552–558, 1997.

H. Stoppiglia, G. Dreyfus, R. Dubois, and Y. Oussar. Ranking a random feature for variable and feature selection. *JMLR*, 3:1399–1414 (this issue), 2003.

R. Tibshirani. Regression selection and shrinkage via the lasso. Technical report, Stanford University, Palo Alto, CA, June 1994.

N. Tishby, F. C. Pereira, and W. Bialek. The information bottleneck method. In *Proc. of the 37th Annual Allerton Conference on Communication, Control and Computing*, pages 368–377, 1999.

K. Torkkola. Feature extraction by non-parametric mutual information maximization. *JMLR*, 3: 1415–1438 (this issue), 2003.

V. G. Tusher, R. Tibshirani, and G. Chu. Significance analysis of microarrays applied to the ionizing radiation response. *PNAS*, 98:5116–5121, April 2001.

V. Vapnik. *Estimation of dependencies based on empirical data*. Springer series in statistics. Springer, 1982.

V. Vapnik. *Statistical Learning Theory*. John Wiley & Sons, N.Y., 1998.

A. Vehtari and J. Lampinen. Bayesian input variable selection using posterior probabilities and expected utilities. Report B31, 2002.

J. Weston, A. Elisseff, B. Schoelkopf, and M. Tipping. Use of the zero norm with linear models and kernel methods. *JMLR*, 3:1439–1461 (this issue), 2003.

J. Weston, S. Mukherjee, O. Chapelle, M. Pontil, T. Poggio, and V. Vapnik. Feature selection for SVMs. In *NIPS 13*, 2000.

E.P. Xing and R.M. Karp. Cliff: Clustering of high-dimensional microarray data via iterative feature filtering using normalized cuts. In *9th International Conference on Intelligence Systems for Molecular Biology*, 2001.