# Practical Feature Selection:
from Correlation to Causality

Isabelle Guyon
955 Creston Road, Berkeley, CA 94708, USA
E-mail: isabelle@clopinet.com.

February 22, 2008

Abstract

Feature selection encompasses a wide variety of methods for selecting a restricted number of input variables or "features", which are "relevant" to a problem at hand. In this report, we guide practitioners through the maze of methods, which have recently appeared in the literature, particularly for supervised feature selection. Starting from the simplest methods of feature ranking with correlation coefficients, we branch in various direction and explore various topics, including "conditional relevance", "local relevance", "multivariate selection", and "causal relevance". We make recommendations for assessment methods and stress the importance of matching the complexity of the method employed to the available amount of training data. Software and teaching material associated with this tutorial are available [12].
Keywords: Feature selection, variable selection, correlation, causality, filters, wrappers.

Introduction

Feature selection is a problem pervasive in all domains of application of machine learning and data mining: engineering applications, robotics and pattern recognition (speech, handwriting, face recognition), Internet applications (text categorization), econometrics and marketing applications and medical applications (diagnosis, prognosis, drug discovery). Restricting the input space to a (small) subset of available input variables has obvious economical benefits in terms of data storage, computational requirements, and cost of future data collection. It often also provides better data or model understanding and even better prediction performance. This report is directed to practitioners who seek a brief review of feature selection techniques and practical recommendations. We adopt a rather informal style to avoid introducing a lot of notations. For more formal developments and an extensive bibliography the readers will be referred, in the course of the report, to selected readings from two recently published books on the subject [11, 19]. We begin by reviewing the goals and expectations of feature selection and the cases in which it may not be needed. Then, starting from the simplest feature ranking methods with correlation coefficients, we branch in various directions to introduce refinements, when then are needed. We finish by a discussion of the problem of model selection, emphasizing that many refinements come at the price of increasing the risk of overfitting.

1 Statement of the problem and motivations

In machine learning and data mining, there are many possible statements of the problem of feature selection. We place ourselves in this report in the context of *supervised learning*, in which data samples or "patterns" are recorded as vectors $\mathbf{x}$ of dimension N and a target variable y must be predicted. We will slightly generalize that framework in the course of the report and extend it to non-vectorial data representations and non-scalar targets. We will allude to the case of unsupervised feature selection, but not discuss it in details. In our simplified setting, pairs of training examples or training "patterns" $\{\mathbf{x}_i, y_i\}, i = 1, ...M$ are given in the form of a data matrix $\mathbf{X} = [x_{ij}](i = 1, ...M, j = 1, ...N)$ of dimensions (M, N), with column vectors X_j, and a target matrix $Y = [y_i]$ of dimension (M, 1). With a certain abuse of notation, we will also call Y the random variable (RV) whose realizations in training data are the y_i and we will call X_j the RV for feature j. Until Section 3.4 we will assume that data are i.i.d. (identically and independently distributed).

The primary goal we are pursuing is to make good predictions of the target variable y for new patterns $\mathbf{x}$. We call "predictor" the device making predictions, which may include a learning machine, whose parameters can be adjusted using training data. Prediction performances are evaluated with an objective function (or a "risk functional") on some test data, distinct from the training data. **The feature selection problem consists in making good predictions with as few variables/features as possible**.[1] In most of this report, we bring back the problem of feature selection to that of selecting a set of feature indices $\mathbf{S}$ of dimension N_s, corresponding to given columns of the data matrix, thus implicitly assuming that the same features are useful for making predictions *for any pattern*. In Section 2.5 we will generalize this framework to *local feature selection*, the problem of selecting features, which are most predictive for single patterns.

Feature selection is a multi-objective problem, which has been formalized in different ways, including minimizing a risk functional subject to using a number of selected features N_s lower than a given threshold; or minimizing N_s subject to keeping the risk lower than a given threshold. Such approaches, which include wrappers and embedded methods discussed in Section 3, require adopting efficient search strategies to explore the space of all possible feature subsets. Other approaches rank first the features with an ad hoc criterion and then optimize prediction performances by considering nested subsets of features built from that ordering. Two central questions must be answered before tackling a new feature selection problem:

1. What do we want to achieve with feature selection?
2. Do we really need it?

Reasons for performing feature selection include:

- Improving performance prediction.
- Reducing computational requirements.
- Reducing data storage requirements.
- Reducing the cost of future measurements.
- Improving data or model understanding.

[1]In this report, we will not make a distinction between variable and feature. Such distinction is relevant for algorithms, which build features from the original variables; selecting features is then different from selecting input variables.

The analysis of the results of recently organized benchmarks [10, 13] reveals that feature selection is seldom needed to improve prediction performance. A standard argument in favor of feature selection is that it is needed to "overcome the curse of dimensionality". For cases in which the data matrix is skinny ($N \gg M$), many more features than training examples), it may seem that feature selection is a requirement to avoid to avoid the unfavorable case in which the number of free parameters of the model greatly exceeds the number of training examples, a case known to yield poor "generalization" (poor performance on test data) [24]. But, today's state-of-the-art machine learning algorithms, which have harnessed the problem of overfitting using powerful "regularization" techniques **do not require feature selection as a preprocessing** step to perform well. These include regularized kernel methods (such as Support Vector Machines (SVM), kernel ridge or logistic regression, and Gaussian processes) and ensemble methods (such as boosting and bagging [4]). These algorithms are described in standard textbooks [6, 16] and software is freely available on-line. In fact, in many cases, feature selection is more harmful than useful because the process of feature selection is data hungry and the more training data are used for feature selection, the less are available to perform model parameter estimation. So, improving performance prediction may, after all, not be the main charter of feature selection. Rather, we may seek, for various other reasons, to limit the number of features used, without *significantly* degrading performance.

2 Getting started: Feature ranking and nested subset methods

In this section, we tackle a first goal: reducing the number of features while least degrading performance. We do not necessarily seek to minimize the number of selected features. In particular, we may satisfy ourselves with eliminating "irrelevant" features, and leaving some redundancy among the selected features. Eliminating redundancy will be addressed, to some extent, at the end of this section.

2.1 Feature ranking

To achieve our initial goal, we advocate the use of simple *univariate feature ranking with correlation coefficients*, sometimes referred to as "filter" methods. Feature selection is then performed by constructing nested subsets of features of increasing size, starting from a subset of one feature (the most correlated to the target), and progressively adding features (less and less correlated with the target). Models are built with every feature subset and evaluated by cross-validation. A standard cross-validation method is 10-fold cross-validation. We recommend using 10 times 10-fold cross-validation to reduce the variance. We also recommend that a separate test set (not used for training or cross-validation) be reserved for a final model testing. The suggested procedure is summarized in Figure 1. At the expense of additional computational burden, the whole procedure can be repeated multiple times (in an outer cross-validation loop) to reduce the variance of performance prediction. Such simple filter methods have proved to work well in practice [7].

In step 4 of the algorithm, a number of state-of-the-art predictive models are suitable, including SVM, kernel ridge regression, boosting and Random Forests. All the models cited perform very well in challenges [10, 13]. The choice may depend on computational considerations.

1. **Reserve test data**: Split the available data into training matrices $\{X, Y\}$ and a test matrices $\{Xt, Yt\}$. A column X_j of X represents the values taken by the j^{th} feature for all the patterns.
2. **Choose criterion**: Choose a ranking statistic $R(X_j, Y)$ (*e.g.* the absolute value of the Pearson correlation coefficient).
3. **Form subsets**: Rank the features: $R(X_{r_1}, Y) \leq R(X_{r_2}, Y) \leq ... \leq R(X_{r_N}, Y)$; and form nested subsets of features: $\mathbf{S}_1 = \{r_1\}, \mathbf{S}_2 = \{r_1, r_2\}, ..., \mathbf{S}_N = \{r_1, r_2, ..., r_N\}$.
4. **Cross-validate**:
    - Split the training data many times (*e.g.* $9/10^{th}$ for training and $1/10^{th}$ for testing), each split being called a "fold".
    - Train models for each feature subset $\mathbf{S}_h$ and each data fold k and test them to obtain the performance $CV(h, k)$.
    - Average $CV(h, k)$ over all folds to obtain $CV(h)$.
    - Select the smallest feature set h^* such that $CV(h)$ is optimal or near optimal (according to a predetermined threshold).
5. **Evaluate on test data**: Restricting the data to the feature subset $\mathbf{S}_{h^*}$, train on a model the entire training set $\{X, Y\}$ and test it on the test set $\{Xt, Yt\}$.

Figure 1: Feature ranking "filter" method.

2.2 Correlation

In step 2 of the algorithm, a ranking criterion must be chosen. The absolute value of the Pearson correlation coefficient is a widely used ranking criterion, which we call PCC:

$$R_{PCC}(X_j, Y) = abs(\frac{(X_j - \overline{X}_j) \cdot (Y - \overline{Y})}{\sqrt{(X_j - \overline{X}_j)^2}\sqrt{(Y - \overline{Y})^2}}) . \tag{1}$$

The PCC is comprised between 0 and 1. It is proportional to the dot product between X_j and Y, after variable standardization (subtracting the mean and dividing by the standard deviation). The PCC is applicable to binary and continuous variables or target values, which makes it very versatile. Categorical variables can be handled by using a complete disjunctive coding (*e.g.* for 4 values [1 0 0 0], [0 1 0 0], [0 0 1 0], [0 0 0 1]).

The PCC is a measure of **linear dependency** between variables. Irrelevant variables (independent of the target) should have a PCC value near zero, but a value of the PCC near zero does not necessarily indicate that the variable is irrelevant: a non-linear dependency may exist, which is not captured by the PCC. It is standard to perform simple data preprocessing to reduce the impact of data non-linearities, *e.g.* taking the logarithm or the square root of the features. Preprocessing is often guided by visual inspection of the data, but it can be subjected to cross-validation. Two simple modifications of the PCC can be made to add robustness against outliers and alleviate the need of preprocessing:

- **Jacknife correlation coefficient**: One large outlying value may result in a spurious high correlation. A standard technique to avoid this problem is to repeat the computation of the correlation coefficient M times, each time removing one value, then averaging the results. The result is called "jacknife correlation coefficient".

- **Spearman correlation coefficient**: Rather than finding an appropriate non-linear scaling of the variables, one may reduce the effect of non-linearities by replacing the variable values by their rank. After this preprocessing, the Pearson correlation coefficient is equivalent to the Spearman correlation coefficient.

Many criteria are closely related to the PCC. For instance the T-statistic, the so-called signal-to-noise-ratio [8], and the F-statistic all give similar results for classification problems, although differences in normalization differentiate when classes are unbalanced. In the following sections, we are branching in several directions to overcome the limitations of feature ranking with the PCC.

2.3 Mutual Information

We began this section by presenting the problem of feature selection as that of removing "irrelevant" variables. Irrelevance of individual variables to the target may be defined in terms of random variable independence:

$$P(X_j, Y) = P(X_j)P(Y)\ , \tag{2}$$

this time denoting by "X_j" and "Y" the random variables distributed like the j^{th} feature and like the target variable. We use in the following the notation $(X_j \perp Y)$ to denote independence and $(X_j \not\perp Y)$ to denote dependence.

The Kullback-Leibler divergence, which measures the discrepancy between $P(X_j, Y)$ and $P(X_j)P(Y)$ can be used as a measure of irrelevance. This is nothing but the well-know mutual information criterion (MIC). For discrete variables, the MIC is defined as:

$$R_{MIC}(X_j, Y) = \sum_{x,y} P(X_j = x, Y = y) \log \frac{P(X_j = x, Y = y)}{P(X_j = x)P(Y = y)}\ . \tag{3}$$

The MIC is a positive quantity, which may be estimated by frequency counts for discrete variables. The advantage of using the MIC rather then the PCC is that dependency is no longer reduced to linear dependency. The drawback is the MIC is harder to estimate than the PCC, particularly for continuous variables. Other non-linear ranking criteria are reviewed in reference [11], Chapter 3.

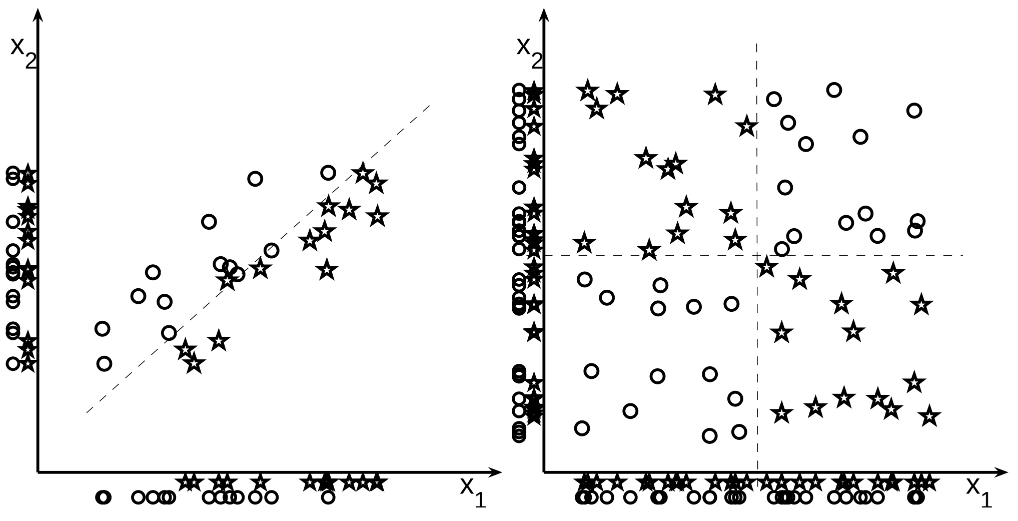
2.4 Conditional relevance

Our first take at feature selection was to rank features according to their individual relevance to the target. Yet, some **individually irrelevant features may become relevant in the context of others**. This is illustrated in Figure 2 by scatter plots of two-dimensional binary classification problems.

It is commonly thought that feature ranking techniques cannot solve problems of relevance "in context". But actually, ranking criteria taking into account the context of other features can be defined, using the notion of **conditional dependency**. For instance, in Figure 2-a, X_2 is independent of Y but it is conditionally dependent, given X_1. Using conditional mutual information has been advocated by several authors (see reference [11], Chapter 6). However, when $N \gg M$ conditioning on more than a few variables is impractical for computational and statistical reasons.
Instead, one may use the Relief criterion [17] a heuristic criterion based on nearest neighbors, which works well in practice for classification problems:

$$R_{Relief}(j) = \frac{M(j)}{H(j)} = \frac{\sum_{i=1}^{M} \sum_{k=1}^{K} |x_{i,j} - x_{M_k(i),j}|}{\sum_{i=1}^{M} \sum_{k=1}^{K} |x_{i,j} - x_{H_k(i),j}|}\ , \tag{4}$$

To evaluate the criterion, first identify in the original feature space, for each example $\mathbf{x}_i$, the K closest examples of the same class $\{\mathbf{x}_{H_k(i)}\}, k = 1...K$ (nearest hits) and the K closest examples of a different class $\{\mathbf{x}_{M_k(i)}\}$ (nearest



(a) $X_2 \perp Y, X_2 \not\perp Y|X_1$ (b) $X_1 \perp Y, X_2 \perp Y, \{X_1, X_2\} \not\perp Y$

Figure 2: **Multivariate dependencies.** (a) Feature X_2 is individually irrelevant to Y ($X_2 \perp Y$), but it becomes relevant in the context of feature X_1 ($X_2 \not\perp Y|X_1$). (b) Two individually irrelevant features ($X_1 \perp Y$ and $X_2 \perp Y$) become relevant when taken jointly ($\{X_1, X_2\} \not\perp Y$).

misses). All features are used to compute the closest examples. Then, in projection on feature j, the sum $M(j)$ of the distances between the examples and their nearest misses is compared to the sum $H(j)$ of distances to their nearest hits. In Equation 4, we use the ratio of $M(j)$ and $H(j)$ to create an criterion independent of feature scale variations. The difference $M(j) - H(j)$ was proposed in the original paper.

The Relief method works for multi-class problems, and it can be extended to continuous targets. One possible extension of Relief to continuous targets is to replace distances by similarities or "kernels" and define a "kernel alignment criterion", inspired by the "kernel alignment" idea [5]:

$$R_{Align}(j) = \sum_i \sum_k \kappa(y_i, y_k) k_j(\mathbf{x}_i, \mathbf{x}_k) K(\mathbf{x}_i, \mathbf{x}_k) \ , \tag{5}$$

where $[\kappa(y_i, y_k)]$ is a (M, M) similarity matrix between target values (*e.g.* the simple outer product YY^T or the correlation matrix), $[K(\mathbf{x}_i, \mathbf{x}_k)]$ is a (M, M) similarity matrix between patterns, and $[k_j(\mathbf{x}_i, \mathbf{x}_k)]$ is a (M, M) similarity matrix between patterns, projected on features j. The kernel alignment criterion is proportional to the square of the Pearson correlation coefficient if $K(\mathbf{x}_i, \mathbf{x}_k) \equiv 1$ and $[\kappa(y_i, y_k)]$ and $[k_j(\mathbf{x}_i, \mathbf{x}_k)]$ are the correlation matrices. For classification, it can be thought of a Parzen windows version of Relief. It is closely related to the Hilbert-Schmidt criterion [9]. Many other extensions of Relief have been proposed, see reference [19], Chapter 9, for a review.

2.5 Local relevance

Another extension of the Relief methodology is to select variables relevant to given patterns, rather than variables relevant to all patterns. This is the concept of “local feature selection”, which is illustrated by the following criterion, in the case of classification problems:

$$R_{Local}(i,j) = \frac{\sum_{k=1}^{K} |x_{i,j} - x_{M_k(i),j}|}{\sum_{k=1}^{K} |x_{i,j} - x_{H_k(i),j}|} \,. \tag{6}$$

Compared to formula 4, we have suppressed the summation over all patterns. A similar extension of formula 5 can be made. Local feature selection can be married with many kernel methods for building predictive models, since kernel methods are based on comparisons of a new example to stored patterns, and are amenable to using a different feature set for each pattern. Other examples of local feature selection methods are found in reference [19], Chapter 11. The concept of local feature selection lends itself to generalizing feature selection to patterns not composed of a fixed number of features, like variable length strings or graphs, since it relies only on pairwise comparisons between patterns. However, despite its attractiveness it remains under-explored, one primary reason being that it is very difficult to overcome the overfitting problem.

2.6 Forward selection and backward elimination

Up to now, we have not been preoccupied by the problem of feature redundancy, we have only tried to eliminate irrelevant features. We now examine methods, which eliminate feature redundancy to some extent. The basic principle is to use conditioning on subsets of other variables. Two approaches can be taken:

- **Forward selection**: Starting from an empty subset of variables, add progressively variables, which are relevant, given previously selected variables.
- **Backward elimination**: Starting from all variables, remove progressively variables, which are irrelevant, given the remaining selected variables.

Conditional mutual information has been advocated by several authors (for a review of information-theoretic methods, see reference [11], Chapter 6). However, it is plagued by the difficulty of estimating mutual information. An alternative method, particularly useful in practice, is the **Gram-Schmidt orthogonalization** procedure. It extends the PCC ranking to **forward selection** by adding progressively variables, which correlate to the target, in the space orthogonal to the variables already selected. See Appendix A for a Matlab code implementation of [21]. A typical example of a **backward elimination** procedure is the RFE SVM (Recursive Feature Elimination Support Vector machine) [14]. For the linear SVM, it boils down to training first on all features, eliminating the feature with smallest weight in absolute value, and iterating until no feature remains. A non-linear version extends it to the non-linear SVM. The procedure bears a lot of similarity with the *unsupervised* “gene shaving” method [15], which uses eigenvalues in lieu of weights to perform the elimination procedure. One also easily sees that the Relief criterion and the kernel alignment criterion (Equations 4 and 5) can be extended to do forward selection and backward elimination. It suffices to replace the projection onto a single feature by a projection on a subset of features.

Both forward selection (FS) and backward elimination (BE) algorithms yield nested subsets of features. Thus, they are similar in spirit to feature ranking algorithms and selecting the optimum number of features can be performed by the procedure outlined in Figure 1. The two types of methods have different advantages and disadvantages. BS selects features using the context of all features not eliminated so far and therefore is more capable of finding complementary features than FS. However, for BS a catastrophic performance degradation is often observed for the smallest nested subsets. In contrast, for FS, even the smallest nested subsets are predictive, which is an advantage if one wants to trade *performance* for *number of features*.

3 Moving on: Optimal subset search

Nested subset methods described in the previous section are sub-optimal with respect to finding the most predictive feature subset because they are greedy methods (once a feature has been removed or added, its inclusion in the selected feature set is not questioned again). We examine now strategies in which the entire space of feature subsets is searched for an optimal feature subset. Before undertaking such an endeavor, we must define a criterion of optimality for feature subsets. The notion of conditional variable independence and "Markov blankets" provides us with such a criterion.

3.1 Markov blankets

Following [20], we define a **Markov blanket** of Y as a subset $\mathbf{M}$ of random variables included in the set $\mathbf{X}$ of all variables such that Y is conditionally independent of all other variables given $\mathbf{M}$. In other words, $\mathbf{M}$ in an Markov blanket *iff* :

$$Y \perp (\mathbf{X} - \mathbf{M}) \mid \mathbf{M} \, .$$

A minimal Markov blanket is called a **Markov boundary** (MB) [20]. There are various other Markov blanket/boundary definitions and Markov boundaries are not necessarily unique, but to simplify the discussion, we will talk here about "the MB". The conditional independence between the target variable and the other variables given the MB implies that all the other variables are truly non informative once the MB is given and can therefore be discarded. This makes the MB a good candidate for an optimal feature subset.

Kohavi and John [18] make a distinction between strong and weak relevance. Tsamardinos and collaborators [23] proved that, under certain conditions, the MB is the set of strongly relevant features. Other "relevant" features, in the sense that adding them to a subset of previously selected features changes the posterior probability of Y, are called "weakly relevant".

The MB captures a certain notion of relevance, but it is not universally optimum. Kohavi and John [18] make a distinction between "relevance" and "usefulness". **Not all strongly relevant features (belonging to the MB) are necessarily "useful" with respect to a given objective**.

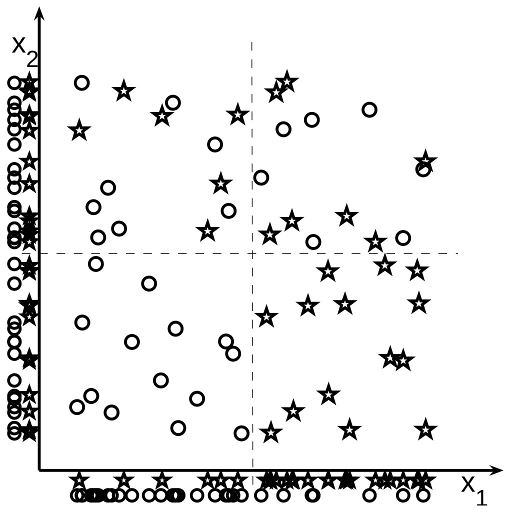


Figure 3: **Objective-dependent relevance.** (a) Feature X_2 is individually irrelevant to Y ($X_2 \perp Y$). But, it is "relevant" in the context of feature X_1 ($X_2 \not\perp Y|X_1$), with respect to the estimation of posterior probabilities: $P(Y|X_1, X_2 = x_2) \neq P(Y|X_1)$ for half of the values of x_2. However, feature X_2 does not contribute to improving classification (the vertical dashed line is the optimum Bayes classification boundary). Hence different objectives yield different feature relevance criteria.

An illustrative example for a two-class classification problem is given in Figure 3. By construction, the examples at the *top* of the figure are labeled randomly, while the examples at the *bottom left* of the figure all belong to one class, and those at the *bottom right* to the other class. Hence, feature X_2 is instrumental in determining the posterior probability distribution of Y. Yet, the optimum Bayes decision boundary is a vertical line, *i.e.* feature X_2 is not helpful to make optimal classification decisions. In this example, if the objective is to minimize classification error, X_2 is not useful, but if the objective is to predict the distribution of Y, it is useful. Even more interestingly, some "useful" features cannot be called "relevant" in any reasonable sense. For instance, for a linear predictor, a feature clamped to a constant value can be "useful" as a means of introducing a bias value, but can hardly be called "relevant".

Thus, even though efficient algorithms to compute the MB have been devised (*e.g.* [23]), it may be preferable to resort to algorithms, which directly optimize the feature subset for given predictors and objective functions. Those will now be reviewed.

3.2 Wrappers

Wrappers are methods, which use any machine learning algorithm as a black box, and search the space of all possible feature subsets to build a predictor with optimum performances. The methodology differs from the step-by-step procedure outlined in Figure 1 only in steps 2 and 3:

2. **Choose criterion**: Choose a search strategy to navigate in the space of 2^N feature subsets.

3. **Form subsets**: Select an ensemble of feature subsets to be investigated or generate new feature subsets iteratively until a stopping criterion is met.

Many search strategies exist, including exhaustive search, beam search, branch and bound, genetic algorithms, simulated annealing, greedy search, to only name a few. For a review, see reference [11], Chapter 4. Extensive search strategies are prone to overfitting for reasons, which will be explained in Section 4.1. Thus, unless a lot of training data are available, the best performing methods are closely related to "greedy search". Greedy search methods include forward selection and backward elimination procedures, which we have already mentioned in Section 2.6. These two approaches have been combined to increase the effectiveness of search, by alternating backward elimination and forward selection, until a stopping criterion is met.

3.3 Embedded methods

Embedded methods refer to a wide variety of techniques in which training data are given to a learning machine, which returns a predictor **and** a subset of features on which it performs predictions. Feature selection is performed in the process of learning, which saves the trouble of a two-step induction process.

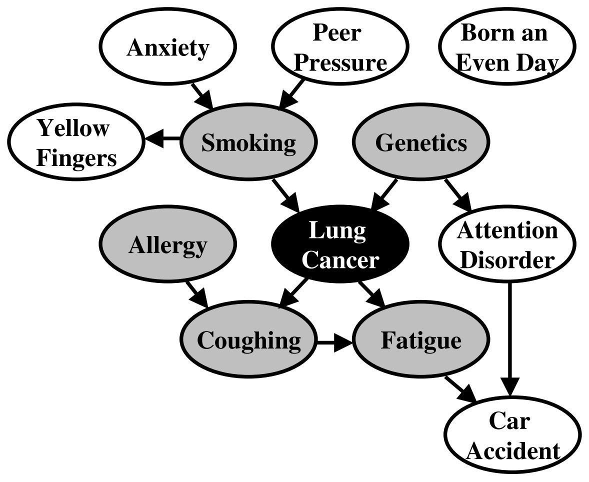
Embedded methods include algorithms, which optimize a regularized risk functional J with respect to two sets of parameters: the parameters of the learning machine α, and parameters $\sigma \in \{0,1\}^N$ indicating which features are selected. Typically, such algorithms alternate two steps until convergence:

$$min_{\alpha} \quad J(\alpha, \sigma) \tag{7}$$

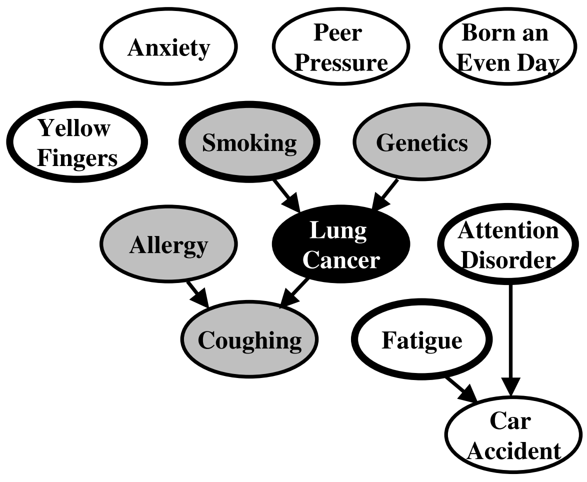
$$min_{\sigma} \quad J(\alpha, \sigma)\ . \tag{8}$$

To facilitate the task of the learning algorithms and allow using parameter search methods like gradient descent, the indicator vector $\sigma \in \{0,1\}^N$ is sometimes replaced by continuous scaling factors $\sigma \in [0,1]^N$ (see reference [11], Chapter 5). A threshold on the scaling factors must be applied when the algorithm terminates selecting features. RFE SVM already mentioned in Section 2.6, may also be considered an embedded approach since it alternates training an SVM and eliminating the feature that least degrades the risk functional (corresponding simply to the feature with smallest weight in absolute value, for the linear SVM). There is a similar method called "multiplicative updates" or zero-norm method, which consists in training a regular SVM, re-scaling its inputs with the absolute values of the weights, and iterating until convergence [26]. This method can be shown to approximately minimize the zero-norm (defined as the number of selected features), under the constraint that training examples are well classified. The regular SVM, which minimizes the two-norm (Euclidean norm) does not have an embedded mechanism of feature selection. But another variant, which minimizes the one-norm (sum of the absolute value of the weights) also has the property that some of the weights are automatically driven to zero, and therefore qualifies as an embedded method [1, 22]. One advantage of these methods is that they converge to an optimum feature subset and do not require performing cross-validation. See Section 4.3 for details.

The above mentioned methods have the flavor of *backward elimination* procedures. Other methods proceed in a *forward selection* manner. For example, tree classifiers iteratively select variables to partition the data such that the data sub-divisions progressively contain a lesser variety of class labels. The RF method, which has been successful in challenges [4, 10, 13], follows this approach. Thus, a variable is selected only if it is "relevant", conditioned on previously selected variables. Data grid models [2] also partition data like trees, but not hierarchically. They provide a piecewise constant approximation to the data distribution. In the course of learning, variables whose optimal discretization consist in a single interval, are deemed uninformative and naturally eliminated.



(a) Unmanipulated



(b) Manipulated

Figure 4: **Manipulations.** The gray shaded nodes represents the Markov boundary of the target variable: "Lung Cancer". In the manipulated graph (b), the four manipulated nodes ("Yellow Fingers", "Smoking", "Attention Disorder", and "Fatigue") are emphasized. As a result of being manipulated, they are disconnected from their original causes. The Markov boundary in graph (b) no longer includes the "Fatigue" node.

3.4 Causal feature selection

Thus far, we have always assumed that data are i.i.d. and, in particular, that the training and test sets are distributed similarly. This is often NOT the case in practice. We consider in this section a particular case of distribution shift, which results from "manipulations" of the data by an "external agent". This gives rise to the distinction between features, which are "causes" of the target, and features, which are "consequences" of the target.

Motivation. In the case of i.i.d. data, feature selection is not concerned with causality: causes and consequences of the target are both predictive. For example, to predict lung cancer, smoking (a possible cause) may be as predictive as coughing (a possible consequence). However, **acting** on a cause may result in a change in the target while acting on a consequence will not. Oftentimes experiments are required to determine causal relationships. But many experiments are impractical, costly or unethical. For example, preventing people from smoking may be costly, forcing them to smoke would be unethical. For this reason, it is important to develop algorithms capable of "causal feature selection" to select variables, which truly influence the target when actions are performed. For instance, we would like to know, before enforcing a new policy preventing to smoke in public place, whether this will be beneficial to public health.

Markov blankets revisited. Bayesian networks are often used to uncover causal relationships. Informally, a Bayesian network (BN) is a graph whose nodes are random variables and vertices represent conditional probabilities. The Markov condition states that a node is independent of non-descendants given its parents. Hence, the data of $P(X|parents(X))$ is sufficient to calculate all conditional variable dependencies. Causal Bayesian networks are Bayesian networks in which arcs are interpreted as causal relationships. In a wide range of practical cases, the Markov boundary (MB) (see Section 3.1) coincides with the set of parents (direct causes), children (direct effects), and spouses (direct causes of direct effects). We show an example in Figure 4-a.

If test data are drawn from the same distribution as training data, the MB does not change and its optimality with respect to making predictions remains the same. However, if test data are manipulated (Figure 4-a) *i.e.* some nodes are set to given values by an external agent and therefore disconnected from their original causes, the MB shrinks to a subset of the original MB. With only the knowledge of which features are going to be manipulated and without actually seeing any test data, if the edges of the network are properly oriented to indicate causal relationships, we can infer which nodes of the MB will no longer be predictive features. Note that, in this framework, the target is NOT manipulated. Hence, no matter which manipulations are performed to the variables, the direct causes of Y will always be predictive. In particular, if **all** variables (other than Y) are manipulated, **only** the direct causes of Y are predictive.

There exist efficient algorithms to determine the local causal neighborhood of the target, see reference [19], Chapter 4, for a review. It should be noted though that several causal graphs may be consistent with the data distribution and therefore not all causal relationships can be determined from "observational data" alone. One must resort to performing some experiments, if such ambiguous causal relationships are to be resolved.

4 Assessment methods

In this section, we discuss methods of model selection and performance prediction.

4.1 Cross-validation

In previous sections, we have advocated cross-validation (see Figure 1). But we have not made recommendations for the value of K in K-fold cross-validation beyond mentioning that most practitioners choose the one-size-fit-all value $K = 10$. This section sheds light on this problem.

For simplicity of the discussion, we consider a single split of the training data into m training examples and p validation examples, $m + p = M$ (for a total of M training examples). This could eventually be repeated K times in a cross-validation loop. As before, we pursue the goal of obtaining best "generalization" performance, *i.e.* performance on "future" data, represented by the separate test set, which we reserved and do not touch during training. Feature selection is cast into a two-level inference problem: the training set is used to adjust the parameters of the predictive model, whereas the validation set is used to select the feature set. We are interested in studying the statistical complexity of the **second level of inference** (feature selection). That level consists in selecting among a **discrete number of predictive models**, corresponding to all the feature subsets under investigation. Because we select among a discrete number of models, the following type of bound applies, for classification problem [24]:

$$E_{gene} \leq E_{valid} + f(\frac{logN_s}{p}) \; , \tag{9}$$

where E_{gene} is the generalization error, N_s is the number of selected feature, p is the number of validation examples, E_{valid} the validation set error, and $f(.)$ is a non-decreasing function of $logN_s/p$, which also depends on the confidence with which the bound applies. Importantly, $logN_s$ is a measure of complexity of the feature selection problem since the ratio $logN_s/p$ governs generalization performances. In that respect, N_s should be made as small as possible and p as large as possible.
Unfortunately increasing p means reducing the number of training examples m and reducing N_s means reducing our chances of finding the optimal feature subset, both of which might result in increasing E_{valid}. So this might not have the expected beneficial effect on E_{gene}.

Table 1 gives an order of magnitude of the complexity of a few feature selection methods. **A rule-of-thumb to avoid overfitting at the second level of inference is that $logN_s$ must be commensurate to p.** It easily understood why nested subset methods and greedy wrappers performing forward selection or backward elimination run much less at risk of overfitting than exhaustive search wrappers: they require only $p = O(logN)$ examples instead of

Table 1: Complexity of feature selection methods.

Method	**Number of subsets tried:** N_s	**Complexity:** $O(\log N_s)$
Feature ranking and nested subset methods	N	$\log N$
Greedy wrappers	$N(N+1)/2$	$\log N$
Exhaustive search wrappers	2^N	N

$p = O(N)$ examples. This is why they have been widely deployed in applications such as genomics and proteomics, text processing, and imaging where the number of features is in the tens of thousands, while the number of training examples remains orders of magnitude smaller. Wrapper methods, which explore an number of subsets exponential in the number of features, are only suitable if the number of training examples exceeds the number of features.

We examine in the next sections alternatives to two-level of inference methods, which save the burden of splitting the training data, and are computationally less expensive than cross-validation.

4.2 Statistical tests

Statistical tests are often used to assess the significance of individual features for feature ranking methods, in the following way. The ranking criterion R is thought of as a statistic, which is distributed in a certain way for irrelevant features. **If for a given feature j, $R(j)$ significantly departs from the expected value of R for irrelevant features, that feature is called "relevant"**. The "null hypothesis" to be tested is that the feature is irrelevant. This method is easily applied to ranking criteria, which are already tabulated statistics, like the Pearson correlation coefficient, the T-statistic, the F-statistic, the G-statistic, etc. Other ranking criteria like the Relief criterion or the "kernel alignment" criterion (Section 2.4), are not tabulated statistics. For these, we can emulate the distribution of irrelevant features with artificially generated "probe" variables, which are added to the original data matrix. Typically, one uses permutations of the values of the columns of the data matrix as probes and the pvalue of the test for a given (real) feature j is approximated by:

$$pval(j) = \frac{N_{sp}}{N_p} \ ,$$

where N_{sp} is the number of selected probes, *i.e.* the number of probes having a value of R greater than $R(j)$, and N_p is the total number of probes. Obviously, the larger the number of probes, the better the approximation. With this method, one is left with the choice of setting a threshold on the pvalue to select the most significant features: small pvalues shed doubt on the validity of the null hypothesis that the feature is irrelevant, *i.e.* relevant features should have small pvalues, which translates in few "probes" having a higher rank. The choice of the threshold of significance must take into account the problem of multiple testing, because we generally select simultaneously multiple features. This can be handled with the simple Bonferroni correction, which consist in multiplying the pvalue by N. However, this correction is generally too conservative and one prefers setting a threshold on the "false discovery rate", the *fraction of features falsely called significant among the selected features*, which is estimated by:

$$FDR(j) = pvalue(j)\frac{N}{N_s} \ ,$$

where N is the total number of features and N_s the number of selected features. See reference [11] Chapter 2 for details.

Statistical tests may also be used to assess the significance of features for multivariate methods, which select an optimal feature subset rather than ranking individual features. The model built from the set of selected features is taken as reference or "null model". A test is performed on the significance of the difference between the null model and a model built with one feature removed. Closed form tabulated statistics can sometimes be derived. The example of the Gram-Schmidt method is treated in reference [11] Chapter 2.

We do not generally recommend using statistical tests for selecting an optimum number of features in lieu of using cross-validation because there is usually little correlation between the prediction performance of the predictor and the fraction of false positive. Rather, the FDR is a useful complementary feature subset quality control indicator.

4.3 Penalty-based methods

Statistical tests are a disguised way of using the "training error" for model selection. In that respect, it is not surprising that they should not perform as well as cross-validation in most cases. There is another alternative to cross-validation, which lumps the two levels of inference into one: penalty-based methods. The general idea is the following: The "training error" is overly optimistic to evaluate model performance (since it has been minimized over the parameters of the model); hence, to become useful, it must be augmented by a penalty term (see *e.g.* [24]). Two types of penalty-based methods are presently most popular: Structural Risk Minimization (SRM) and Bayesian methods. Even though they are based on different theoretical arguments, they often end up with similar penalty terms, which penalize more complex models.

A classical example is that of weight "shrinkage". For a linear predictor $f(\mathbf{x}) = \mathbf{w}\mathbf{x} + b$, the SRM method advocates building a structure to rank models in order of increasing complexity, based on the Euclidean norm (or 2-norm) of the weight vector $\|\mathbf{w}\|_2$. In the Bayesian framework, this is equivalent to choosing a prior on the weights centered on $\|\mathbf{w}\|_2 = \mathbf{0}$. The resulting penalty term is proportional to $\|\mathbf{w}\|_2^2$. Depending on the choice of the norm and of the loss function, various methods are obtained, including ridge regression and Support Vector Machines (SVM) for the 2-norm, lasso and 1-norm SVM for the 1-norm, and 0-norm SVM for the 0-norm. See reference [11] Chapter 1 for other examples and more details. As previously mentioned in Section 3.3, the 2-norm methods do not perform feature selection while the 1-norm and 0-norm methods do. These last 2 types of methods converge to an optimal feature subset in the process of training. Some authors have reported that penalty-based methods may overfit in some sense by selecting too few features. Cross-validation may be used as a halting criterion to overcome this problem [26]. But then of course the benefit of penalty-based method is partially lost.

4.4 Ensemble methods and stability

A discussion of model selection would not be complete without mentioning that ensemble and Bayesian methods, which pool a large number of solutions and "vote" among them, circumvent the problem of choosing a single one and often yield better results. A typical "frequentist" approach is to use "bagging" to obtain feature subset diversity by resampling the training data set and/or the feature set before feature selection. Such is the underlying methodology of Random Forests (RF) [4]. A more popular approach among Bayesians is to use Markov Chain Monte Carlo methods, both searching for useful feature subsets and attributing them a "weight" (the posterior probability of the feature subset) [25]. Generally, two approaches can be taken:

1. Using a committee of predictors, each based on a different feature subset (the voting weights are based on the expected individual prediction performance of the committee members, as determined *e.g.* by cross-validation).

2. Pooling the feature sets, retaining only the features, which are most often selected among the best feature sets encountered. A single predictor is then built from the pooled set.

The performance improvement obtained by ensemble methods is generally attributed to a reduction in "variance" or "instability". Formal relations between generalization performance and stability have been established [3]. Since multivariate methods often yield instable results, a lot of effort has been concentrating on stabilizing multivariate methods. While both approached (1) and (2) mentioned above are applicable to univariate methods, approach (2) does not guarantee that the feature set obtained will be optimal for multivariate methods since there is not necessarily a complementarity of features in a pooled set. If one seeks a stable complementary subset of features, one can resort to computing a feature subset "centroid", *e.g.* the feature subset closest on average to all selected feature subsets, according to some edit distance.

5 Conclusion

Feature selection is a multi-faceted problem, which has evolved over the past few years from a collection of mostly heuristic methods to a theoretically grounded methodology. In this report, we took a few snapshots of the kaleidoscope of existing methods to guide practitioners towards the best solution to their particular problem. Our final recommendation is to always start with the simplest methods (*e.g.* univariate feature ranking with correlation coefficients) and add complexity only as needed. Many promising avenues remain under-explored, such as "local" or "causal" feature selection, or even "dynamic" feature selection, in which features are selected on-the-fly, like in the game of 20 questions [2].

Acknowledgements

We are grateful to Gavin Brown for making corrections and suggestions.

References

[1] J. Bi, K. Bennett, M. Embrechts, C. Breneman, and M. Song. Dimensionality reduction via sparse support vector machines. *JMLR*, 3:1229–1243, 2003.

[2] M. Boullé. Report on preliminary experiments with data grid models in the agnostic learning vs. prior knowledge challenge. In *IEEE/INNS conference IJCNN 2007*, Orlando, Florida, August 12-17 2007.

[3] O. Bousquet and A. Elisseeff. Stability and generalization. *JMLR*, 2, 2002.

[4] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[5] N. Cristianini, J. Shawe-Taylor, A. Elisseeff, and J. S. Kandola. On kernel-target alignment. In *Advances in Neural Information Processing Systems 14*, pages 367–373. MIT press, 2002.

[6] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. John Wiley and Sons, USA, 2001.

[7] I. Guyon et al. Competitive baseline methods set new standards for the NIPS 2003 feature selection benchmark. *Pattern Recogn. Lett.*, 28(12):1438–1444, 2007.

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

[2] In the game of 20 questions, you can ask 20 questions, which can be answered by "yes" or "no" to guess the identity of an object. Each question can be thought of as a feature. If the feature set is restricted and the goal of the game is replaced by guessing an attribute, which cannot be queried, the game of 20 questions is a dynamic feature selection problem

[9] A. Gretton, O. Bousquet, A. J. Smola, and B. Schölkopf. Measuring statistical dependence with hilbert-schmidt norms. In *Algorithmic Learning Theory*, pages 63–77. Springer, 2005.

[10] I. Guyon, S. Gunn, A. Ben-Hur, and G. Dror. Result analysis of the NIPS 2003 feature selection challenge. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 545–552. MIT Press, Cambridge, MA, 2005.

[11] I. Guyon, S. Gunn, M. Nikravesh, and L. Zadeh, Editors. *Feature Extraction, Foundations and Applications*. Studies in Fuzziness and Soft Computing. Physica-Verlag, Springer, 2006.

[12] I. Guyon and A. Saffari. The challenge learning object package (CLOP): Matlab(r) machine learning and feature selection methods, 2007, `http://clopinet.com/CLOP/`.

[13] I. Guyon, A. Saffari, G. Dror, and J. Buhmann. Performance prediction challenge. In *IEEE/INNS conference IJCNN 2006*, Vancouver, Canada, July 16-21 2006.

[14] 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.

[15] T. Hastie, R. Tibshirani, A. Eisen, R. Levy, L. Staudt, D. Chan, and P. Brown. Gene shaving as a method for identifying distinct sets of genes with similar expression patterns. *Genome Biology*, 1:1–21, 2000.

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

[17] K. Kira and L. Rendell. A practical approach to feature selection. In D. Sleeman and P. Edwards, editors, *International Conference on Machine Learning*, pages 249–256. Morgan Kaufmann, July 1992.

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

[19] H. Liu and H. Motoda, Editors. *Computational Methods of Feature Selection (Chapman & Hall/Crc Data Mining and Knowledge Discovery Series)*. Chapman & Hall/CRC, 2007.

[20] J. Pearl. *Probabilistic reasoning in intelligent systems*. Morgan Fauffman, 1988.

[21] H. Stoppiglia, G. Dreyfus, R. Dubois, and Y. Oussar. Ranking a random feature for variable and feature selection. *JMLR*, 3:1399–1414, 2003.

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

[23] I. Tsamardinos, C.F. Aliferis, and A. Statnikov. Algorithms for large scale Markov blanket discovery. In *16th International Florida Artificial Intelligence Research Society (FLAIRS) Conference*, pages 376–380, St. Augustine, Florida, USA, May 12-14 2003. AAAI Press.

[24] V. Vapnik. *Statistical Learning Theory*. John Wiley and Sons, N.Y., 1998.

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

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

A Forward selection with Gram-Schmidt orthogonalization

```
function idx = gram_schmidt(X, Y, featnum)
%idx = gram_schmidt(X, Y, featnum)
% Feature selection by Gram Schmidt orthogonalization.
% X          -- Data matrix (m, n), m patterns, n features.
% Y          -- Target vector (m,1).
% featnum    -- Number of features selected.
% idx        -- Ordered indices of the features (best first.)

[m, N]=size(X);
if nargin<3 | isempty(featnum), featnum=min(m,N); end
idx=zeros(1,featnum); w=zeros(1,featnum);
rss=zeros(1,featnum);       % Residual sum of squares
colid=1:N;                  % Original feature numbering
n=N;
for k=1:featnum % Main loop over features
    fprintf('\nTraining on feature set size: %d\n', N-n+1);
    % Normalize
    %(subtract the mean to get a correlation rather that a cos)
    XN=sqrt(sum(X.^2));     % Norms of the feature vectors
    XN(XN==0)=eps;
    X_norma = X./repmat(XN, m,1); % Normalized feature matrix
    % Project onto Y
    y_proj = sum(repmat(Y, 1, n).*X_norma);
    ay_proj=abs(y_proj);
    % Find the direction of maximum projection
    [maxval, maxidx] = max(ay_proj); % Dir. of max. proj.
    idx(k)=colid(maxidx);            % Index of that feature
    % Update the model
    w(k)=y_proj(maxidx)/XN(maxidx);  % Weight of that feature
    Y_proj = w(k)*X(:,maxidx); % Proj. Y on dir. X(:,maxidx)
    Y_residual = Y - Y_proj;         % New residual
    rss(k) = sum(Y_residual.^2);     % Residual error model
    % Compute the residual X vectors
    X_proj = sum(repmat(X_norma(:,maxidx),1,n).*X);
    X_residual = X-repmat(X_proj, m, 1).* ...
                        repmat(X_norma(:,maxidx), 1, n);
    % Change the matrix to iterate
    Y=Y_residual;
    X=X_residual(:, [1:maxidx-1,maxidx+1:n]);
    colid=colid([1:maxidx-1,maxidx+1:n]);
    n=n-1;
    fprintf('Training mse: %5.2f\n', rss(k)/m);
    fprintf('Features selected:\nidx=[');
    fprintf('%d ',idx(1:k));
    fprintf(']\n');
end
```