An Introduction to Feature Extraction

Isabelle Guyon[1] and André Elisseeff[2]

[1] ClopiNet, 955 Creston Rd., Berkeley, CA 94708, USA. isabelle@clopinet.com
[2] IBM Research GmbH, Zürich Research Laboratory, Säumerstrasse 4, CH-8803
Rüschlikon, Switzerland. ael@zurich.ibm.com

This chapter introduces the reader to the various aspects of feature extraction covered in this book. Section 1 reviews definitions and notations and proposes a unified view of the feature extraction problem. Section 2 is an overview of the methods and results presented in the book, emphasizing novel contributions. Section 3 provides the reader with an entry point in the field of feature extraction by showing small revealing examples and describing simple but effective algorithms. Finally, Section 4 introduces a more theoretical formalism and points to directions of research and open problems.

1 Feature Extraction Basics

In this section, we present key notions that will be necessary to understand the first part of the book and we synthesize different notions that will be seen separately later on.

1.1 Predictive modeling

This book is concerned with problems of predictive modeling or supervised machine learning. The latter refers to a branch of computer Science interested in reproducing human learning capabilities with computer programs. The term machine learning was first coined by Samuel in the 50's and was meant to encompass many intelligent activities that could be transferred from human to machine. The term "machine" should be understood in an abstract way: not as a physically instantiated machine but as an automated system that may, for instance, be implemented in software. Since the 50's machine learning research has mostly focused on finding relationships in data and analyzing the processes for extracting such relations, rather than building truly "intelligent systems".

Machine learning problems occur when a task is defined by a series of cases or examples rather than by predefined rules. Such problems are found in a wide variety of application domains, ranging from engineering applications in robotics and pattern recognition (speech, handwriting, face recognition), to Internet applications (text categorization) and medical applications (diagnosis, prognosis, drug discovery). Given a number of "training" examples (also called data points, samples, patterns or observations) associated with desired outcomes, the machine learning process consists of finding the relationship between the patterns and the outcomes using solely the training examples. This shares a lot with human learning where students are given examples of what is correct and what is not and have to infer which rule underlies the decision. To make it concrete, consider the following example: the data points or examples are clinical observations of patient and the outcome is the health status: healthy or suffering from cancer.[3] The goal is to predict the unknown outcome for new "test" examples, e.g. the health status of new patients. The performance on test data is called "generalization". To perform this task, one must build a predictive model or *predictor*, which is typically a function with adjustable parameters called a "learning machine". The training examples are used to select an optimum set of parameters.

We will see along the chapters of this book that enhancing learning machine generalization often motivates feature selection. For that reason, classical learning machines (e.g. Fisher's linear discriminant and nearest neighbors) and state-of-the-art learning machines (e.g. neural networks, tree classifiers, Support Vector Machines (SVM)) are reviewed in Chapter 1. More advanced techniques like ensemble methods are reviewed in Chapter 5. Less conventional neuro-fuzzy approaches are introduced in Chapter 8. Chapter 2 provides guidance on how to assess the performance of learning machines.

But, before any modeling takes place, a data representation must be chosen. This is the object of the following section.

1.2 Feature construction

In this book, data are represented by a fixed number of features which can be binary, categorical or continuous. Feature is synonymous of input variable or attribute.[4] Finding a good data representation is very domain specific and related to available measurements. In our medical diagnosis example, the features may be symptoms, that is, a set of variables categorizing the health status of a patient (e.g. fever, glucose level, etc.).

Human expertise, which is often required to convert "raw" data into a set of useful features, can be complemented by *automatic feature construction* methods. In some approaches, feature construction is integrated in the modeling process. For examples the "hidden units" of artificial neural networks

[3] The outcome, also called target value, may be binary for a 2-class classification problem, categorical for a multi-class problem, ordinal or continuous for regression.

[4] It is sometimes necessary to make the distinction between "raw" input variables and "features" that are variables constructed for the original input variables. We will make it clear when this distinction is necessary.

compute internal representations analogous to constructed features. In other approaches, feature construction is a preprocessing. To describe preprocessing steps, let us introduce some notations. Let $\boldsymbol{x}$ be a pattern vector of dimension n, $\boldsymbol{x} = [x_1, x_2, ...x_n]$. The components x_i of this vector are the original features. We call $\boldsymbol{x}'$ a vector of transformed features of dimension n'. Preprocessing transformations may include:

- *Standardization:* Features can have different scales although they refer to comparable objects. Consider for instance, a pattern $x = [x_1, x_2]$ where x_1 is a width measured in meters and x_2 is a height measured in centimeters. Both can be compared, added or subtracted but it would be unreasonable to do it before appropriate normalization. The following classical centering and scaling of the data is often used: $x'_i = (x_i - \mu_i)/\sigma_i$, where μ_i and σ_i are the mean and the standard deviation of feature x_i over training examples.
- *Normalization:* Consider for example the case where $\boldsymbol{x}$ is an image and the x_i's are the number of pixels with color i, it makes sense to normalize $\boldsymbol{x}$ by dividing it by the total number of counts in order to encode the distribution and remove the dependence on the size of the image. This translates into the formula: $\boldsymbol{x}' = \boldsymbol{x}/\|\boldsymbol{x}\|$.
- *Signal enhancement.* The signal-to-noise ratio may be improved by applying signal or image-processing filters. These operations include baseline or background removal, de-noising, smoothing, or sharpening. The Fourier transform and wavelet transforms are popular methods. We refer to introductory books in digital signal processing (Lyons, 2004), wavelets (Walker, 1999), image processing (R. C. Gonzalez, 1992), and morphological image analysis (Soille, 2004).
- *Extraction of local features:* For sequential, spatial or other structured data, specific techniques like convolutional methods using hand-crafted kernels or syntactic and structural methods are used. These techniques encode problem specific knowledge into the features. They are beyond the scope of this book but it is worth mentioning that they can bring significant improvement.
- *Linear and non-linear space embedding methods:* When the dimensionality of the data is very high, some techniques might be used to project or embed the data into a lower dimensional space while retaining as much information as possible. Classical examples are Principal Component Analysis (PCA) and Multidimensional Scaling (MDS) (Kruskal and Wish, 1978). The coordinates of the data points in the lower dimension space might be used as features or simply as a means of data visualization.
- *Non-linear expansions:* Although dimensionality reduction is often summoned when speaking about complex data, it is sometimes better to increase the dimensionality. This happens when the problem is very complex and first order interactions are not enough to derive good results. This consists for instance in computing products of the original features x_i to create monomials $x_{k_1} x_{k_2} ... x_{k_p}$.

- *Feature discretization.* Some algorithms do no handle well continuous data. It makes sense then to discretize continuous values into a finite discrete set. This step not only facilitates the use of certain algorithms, it may simplify the data description and improve data understanding (Liu and Motoda, 1998).

Some methods do not alter the space dimensionality (e.g. signal enhancement, normalization, standardization), while others enlarge it (non-linear expansions, feature discretization), reduce it (space embedding methods) or can act in either direction (extraction of local features).

Feature construction is one of the key steps in the data analysis process, largely conditioning the success of any subsequent statistics or machine learning endeavor. In particular, one should beware of not losing information at the feature construction stage. It may be a good idea to add the raw features to the preprocessed data or at least to compare the performances obtained with either representation. We argue that it is always better to err on the side of being too inclusive rather than risking to discard useful information. The medical diagnosis example that we have used before illustrates this point. Many factors might influence the health status of a patient. To the usual clinical variables (temperature, blood pressure, glucose level, weight, height, etc.), one might want to add diet information (low fat, low carbonate, etc.), family history, or even weather conditions. Adding all those features seems reasonable but it comes at a price: it increases the dimensionality of the patterns and thereby immerses the relevant information into a sea of possibly irrelevant, noisy or redundant features. How do we know when a feature is relevant or informative? This is what “feature selection” is about and is the focus of much of this book.

1.3 Feature selection

We are decomposing the problem of feature extraction in two steps: feature construction, briefly reviewed in the previous section, and feature selection, to which we are now directing our attention. Although feature selection is primarily performed to select relevant and informative features, it can have other motivations, including:

1. *general data reduction*, to limit storage requirements and increase algorithm speed;
2. *feature set reduction*, to save resources in the next round of data collection or during utilization;
3. *performance improvement*, to gain in predictive accuracy;
4. *data understanding*, to gain knowledge about the process that generated the data or simply visualize the data

Several chapters in Part I are devoted to feature selection techniques. Chapter 3 reviews *filter* methods. Filters are often identified to feature ranking methods. Such methods provide a complete order of the features using a relevance index. Methods for computing ranking indices include correlation coefficients, which assess the degree of dependence of individual variables with the outcome (or target). A variety of other statistics are used, including classical test statistics (T-test, F-test, Chi-squared, etc.) More generally, methods that select features without optimizing the performance of a predictor are referred to as "filters". Chapter 6 presents information theoretic filters.

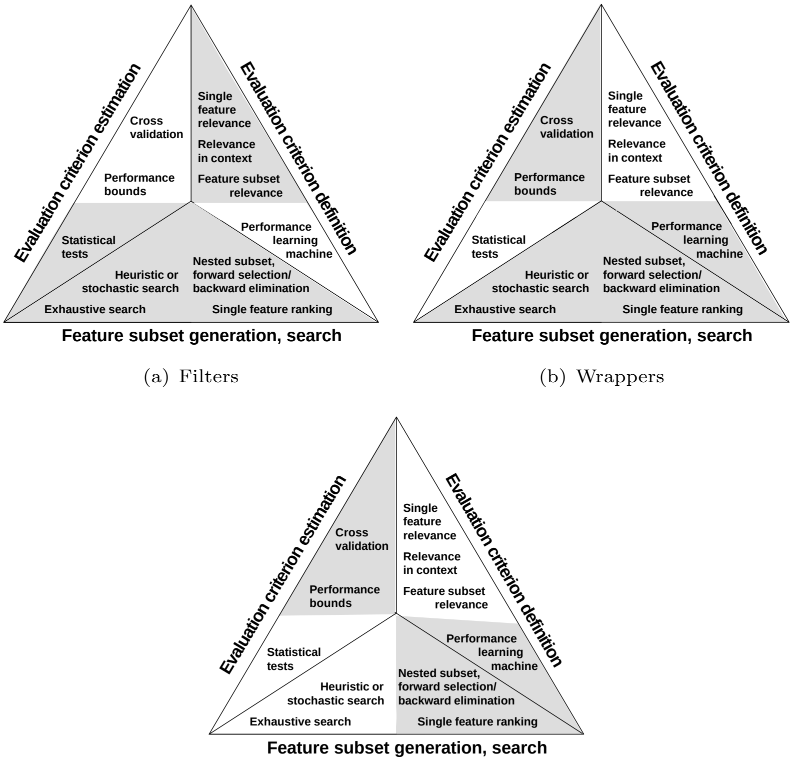
Chapter 4 and Chapter 5 are devoted to *wrappers* and *embedded* methods. Such methods involve the predictor as part of the selection process. Wrappers utilize a learning machine as a "black box" to score subsets of features according to their predictive power. Embedded methods perform feature selection in the process of training and are usually specific to given learning machines. Wrappers and embedded methods may yield very different feature subsets under small perturbations of the dataset. To minimize this effect, Chapter 7 explains how to improve feature set stability by using ensemble methods.

A critical aspect of feature selection is to properly assess the quality of the features selected. Methods from classical statistics and machine learning are reviewed in Chapter 2. In particular, this chapter reviews hypothesis testing, cross-validation, and some aspects of experimental design (how many training examples are needed to solve the feature selection problem.)

A last, it should be noted that it is possible to perform feature construction and feature selection simultaneously, as part of a global optimization problem. Chapter 6 introduces the reader to methods along this line.

1.4 Methodology

The chapters of Part I group topics in a thematic way rather than in a methodological way. In this section, we present a unified view of feature selection that transcends the old cleavage filter/wrapper and is inspired by the views of (Liu and Motoda, 1998).


Evaluation criterion estimation
Cross validation
Performance bounds
Statistical tests
Evaluation criterion definition
Single feature relevance
Relevance in context
Feature subset relevance
Performance learning machine
Heuristic or stochastic search
Nested subset, forward selection/ backward elimination
Exhaustive search
Single feature ranking
Feature subset generation, search

(a) Filters

(b) Wrappers

(c) Embedded methods

Fig. 1. *The three principal approaches of feature selection.* The shades show the components used by the three approaches: filters, wrappers and embedded methods.

There are four aspects of feature extraction:

- feature construction;
- feature subset generation (or search strategy);
- evaluation criterion definition (e.g. relevance index or predictive power);
- evaluation criterion estimation (or assessment method).

The last three aspects are relevant to feature selection and are schematically summarized in Figure 1.

Filters and *wrappers* differ mostly by the *evaluation criterion*. It is usually understood that filters use criteria not involving any learning machine, e.g. a relevance index based on correlation coefficients or test statistics, whereas wrappers use the performance of a learning machine trained using a given feature subset.

Both filter and wrapper methods can make use of *search strategies* to explore the space of all possible feature combinations that is usually too large to be explored exhaustively (see Chapter 4.) Yet filters are sometimes assimilated to feature ranking methods for which feature subset generation is trivial since only single features are evaluated (see Chapter 3). Hybrid methods exist, in which a filter is used to generate a ranked list of features. On the basis of the order thus defined, nested subsets of features are generated and computed by a learning machine, i.e. following a wrapper approach. Another class of *embedded methods* (Chapter 5) incorporate feature subset generation and evaluation in the training algorithm.

The last item on the list, *criterion estimation*, is covered in Chapter 2. The difficulty to overcome is that a defined criterion (a relevance index or the performance of a learning machine) must be estimated from a *limited amount of training data*. Two strategies are possible: "in-sample" or "out-of-sample". The first one (in-sample) is the "classical statistics" approach. It refers to using all the training data to compute an empirical estimate. That estimate is then tested with a statistical test to assess its significance, or a performance bound is used to give a guaranteed estimate. The second one (out-of-sample) is the "machine learning" approach. It refers to splitting the training data into a training set used to estimate the parameters of a predictive model (learning machine) and a validation set used to estimate the learning machine predictive performance. Averaging the results of multiple splitting (or "cross-validation") is commonly used to decrease the variance of the estimator.

2 What is New in Feature Extraction?

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. We organized in 2001 a first NIPS workshop, the proceedings of which include papers exploring domains with hundreds to tens of thousands of variables or features (Guyon and Elisseeff, 2003). Following this workshop, we organized a feature selection competition, the results of which were presented at a NIPS workshop in 2003. The present book is the outcome of the latter.

Part II of the book describes the methods used by the best ranking participants. Chapter II summarizes the results of the competition. Five datasets were used that were chosen to span a variety of domains (biomarker discovery, drug discovery, handwriting recognition, and text classification) and difficulties (the input variables are continuous or binary, sparse or dense; one dataset has unbalanced classes.) One dataset was artificially constructed to illustrate a particular difficulty: selecting a feature set when no feature is informative individually. We chose datasets that had sufficiently many examples to create a large enough test set and obtain statistically significant results (Guyon, 2003). We introduced a number of random features called *probes* to make the task more difficult and identify the algorithms capable of filtering them out.

The challenge winning methods are described in Chapter 10. The authors use a combination of Bayesian neural networks (Neal, 1996) and Dirichlet diffusion trees (Neal, 2001). Two aspects of their approach were the same for all data sets: (1) reducing the number of features used for classification to no more than a few hundred, either by selecting a subset of features using simple univariate significance tests, or by Principal Component Analysis; (2) applying a classification method based on Bayesian learning, using an Automatic Relevance Determination (ARD) prior that allows the model to determine which of the features are most relevant (MacKay, 1994, Neal, 1996). Bayesian neural network learning with computation by Markov chain Monte Carlo (MCMC) is a well developed technology (Neal, 1996). Dirichlet diffusion trees are a new Bayesian approach to density modeling and hierarchical clustering.

A wide variety of other methods presented in Part II performed nearly as well. For feature selection, filter methods proved quite effective. Four of the top entrants explore successfully the use of Random Forests (RF)[5] as a filter (Chapter 11, Chapter 15, and Chapter 12). Simple correlation coefficients also performed quite well (Chapter 13, Chapter 14, Chapter 20, and Chapter 23), as well as information theoretic ranking criteria (Chapter 22 and Chapter 24). Some of the recently introduced embedded methods using a Support Vector Machine (SVM) or a related kernel method were applied with success (Chapter 12, Chapter 13, Chapter 16, Chapter 18, Chapter 19, and Chapter 21). Among the most innovative methods, Chapter 17 and Chapter 29 present a margin-based feature selection method inspired by the Relief algorithm (Kira and Rendell, 1992).

As far as classifier choices are concerned, the second best entrants (Chapter 11) use the simple regularized least square kernel method as classifier. Many of the other top entrants use regularized kernel methods with various loss functions, including kernel partial least squares (KPLS) (Chapter 21), vanilla Support Vector machines (SVM) (Chapter 12, Chapter 20, Chapter 22, Chapter 23 and Chapter 24), transductive SVM (Chapter 13), Bayesian SVM (Chapter 18), Potential SVM (Chapter 19), and 1-norm SVM (Chapter 16). Two other entrants used neural networks like the winners (Chapter 14 and Chapter 26). Other methods includes Random Forests (RF) (Chapter 15), Naïve Bayes (Chapter 24 and Chapter 25) and simple nearest neighbors (Chapter 17).

Part III of the book devotes several chapters to novel approaches to feature construction. Chapter 27 provides a unifying framework to many methods of linear and non-linear space embedding methods. Chapter 28 proposes a method for constructing orthogonal features for an arbitrary loss. Chapter 31 gives an example of syntactic feature construction: protein sequence motifs.

[5] Random Forests are ensembles of tree classifiers.

3 Getting started

Amidst the forest of methods, the reader who is getting started in the field may be lost. In this section, we introduce basic concepts and briefly describe simple but effective methods. We illustrate with small two-dimensional classification problems (Figure 2) some special cases.

One approach to feature selection is to rank features according to their individual relevance (Section 3.1.) Such feature ranking methods are considered fast and effective, particularly when the number of features is large and the number of available training examples comparatively small (e.g. 10,000 features and 100 examples.) In those cases, methods that attempt to search extensively the space of feature subsets for an optimally predictive can be much slower and prone to "overfitting" (perfect predictions may be achieved on training data, but the predictive power on test data will probably be low.)

However, as we shall see in some other examples (Section 3.2 and 3.3), there are limitations to individual feature ranking, because of the underlying feature independence assumptions made by "univariate" methods:

- features that are not individually relevant may become relevant in the context of others;
- features that are individually relevant may not all be useful because of possible redundancies.

So-called "multivariate" methods take into account feature dependencies. Multivariate methods potentially achieve better results because they do not make simplifying assumptions of variable/feature independence.

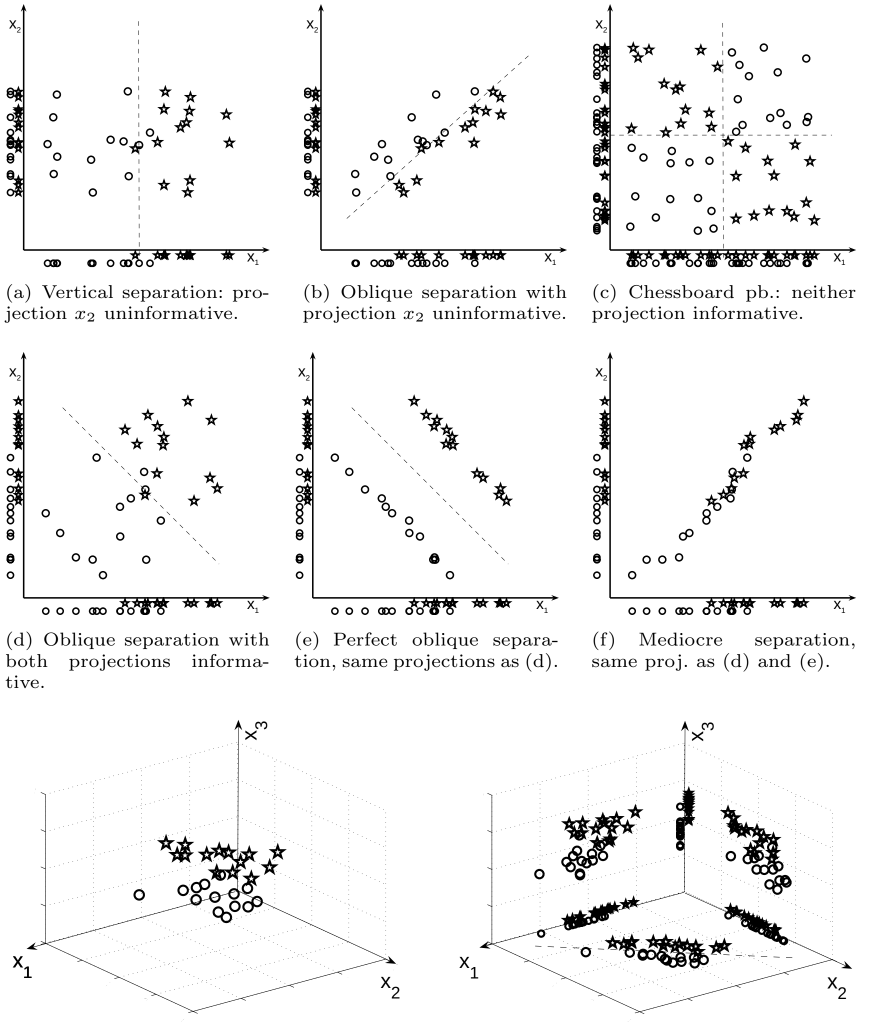
3.1 Individual relevance ranking

Figure 2-a shows a situation in which one feature (x_1) is relevant individually and the other (x_2) does not help providing a better class separation. For such situations individual feature ranking works well: the feature that provides a good class separation by itself will rank high and will therefore be chosen.

The **Pearson correlation coefficient** is a classical relevance index used for individual feature ranking. We denote by $\boldsymbol{x}_j$ the m dimensional vector containing all the values of the j^{th} feature for all the training examples, and by $\boldsymbol{y}$ the m dimensional vector containing all the target values. The Pearson correlation coefficient is defined as:

$$C(j) = \frac{|\sum_{i=1}^{m}(x_{i,j} - \bar{x}_j)(y_i - \bar{y})|}{\sqrt{\sum_{i=1}^{m}(x_{i,j} - \bar{x}_j)^2 \sum_{i=1}^{m}(y_i - \bar{y})^2}} \ , \tag{1}$$

where the bar notation stands for an average over the index i. This coefficient is also the absolute value of the cosine between vectors $\boldsymbol{x}_i$ and $\boldsymbol{y}$, after they have been centered (their mean subtracted). The Pearson correlation coefficient may be used for regression and binary classification problems. For



(a) Vertical separation: projection x_2 uninformative.

(b) Oblique separation with projection x_2 uninformative.

(c) Chessboard pb.: neither projection informative.

(d) Oblique separation with both projections informative.

(e) Perfect oblique separation, same projections as (d).

(f) Mediocre separation, same proj. as (d) and (e).

(g) Three-d example: Best single feature = x_3; best pair = $\{x_1, x_2\}$.

(h) Projections of Figure (g).

Fig. 2. *Small classification examples.* One class is represented by circles and the other by stars. The horizontal axis represents one feature and the vertical axis the other. In the last example we have a third feature. We represent each class by circles or stars. We show the projections of the classes on the axes as superimposed circles and stars.

multi-class problems, one can use instead the closely related Fisher coefficient. The Pearson correlation coefficient is also closely related to the T-test statistic, and the Naïve Bayes ranking index. See Chapter 3 for details and for other examples of ranking criteria.

Rotations in feature space often simplify feature selection. Figure 2-a is obtained from Figure 2-d by a 45 degree rotation. One notices that to achieve the same separation, two features are needed in Figure 2-d, while only one is needed in Figure 2-a. Rotation is a simple linear transformation. Several preprocessing methods such as principal component analysis (PCA) perform such linear transformations, which permit reducing the space dimensionality and exhibit better features.

The notion of **relevance is related to the objective** being pursued. A feature that is irrelevant for classification may be relevant for predicting the class conditional probabilities. Such is the case of feature x_2 in Figure 2-a. The examples of the two classes are drawn from overlapping Gaussian distributions whose class centers are aligned with axis x_1. Thus, $P(y|\mathbf{x})$ is not independent of x_2, but the error rate of the optimum Bayes classifier is the same whether feature x_2 is kept or discarded. This points to the fact that density estimation is a harder problem than classification and usually requires more features.

3.2 Relevant features that are individually irrelevant

In what follows, we justify the use of multivariate methods, which make use of the predictive power of features considered jointly rather than independently.

A helpful feature may be irrelevant by itself. One justification of multivariate methods is that features that are individually irrelevant may become relevant when used in combination. Figure 2-b gives an example of a linear separation in which an individually irrelevant feature helps getting a better separation when used with another feature.[6] This case occurs in real world examples: feature x_1 might represent a measurement in an image that is randomly offset by a local background change; feature x_2 might be measuring such local offset, which by itself is not informative. Hence, feature x_2 might be completely uncorrelated to the target and yet improve the separability of feature x_1, if subtracted from it.

Two individually irrelevant features may become relevant when used in combination. The case of Figure 2-c, known as the "chessboard problem", illustrates this situation.[7] In the feature selection challenge (see Part II), we proposed a problem that generalizes this case in a higher dimension space: The Madelon dataset is built from clusters placed on the vertices of a hypercube in five dimensions and labeled at random.

The **Relief method** is a classical example of multivariate filter. Most multivariate methods rank subsets of features rather than individual features.

[6] It is worth noting that the x_2 projection is the same in Figures 2-a and 2-b.

[7] This is a small 2x2 chessboard. This problem is analogous to the famous XOR problem, itself a particular case of the parity problem.

Still, there exist multivariate relevance criteria to rank individual features according to their relevance *in the context of others*. To illustrate this concept, we give as example a ranking index for classification problems derived from the Relief algorithm (Kira and Rendell, 1992):

$$C(j) = \frac{\sum_{i=1}^{m} \sum_{k=1}^{K} |x_{i,j} - x_{M_k(i),j}|}{\sum_{i=1}^{m} \sum_{k=1}^{K} |x_{i,j} - x_{H_k(i),j}|} , \tag{2}$$

Notations will be explained shortly. The Relief algorithm uses an approach based on the K-nearest-neighbor algorithm. To evaluate the index, we first identify in the original feature space, for each example $\boldsymbol{x}_i$, the K closest examples of the same class $\{\boldsymbol{x}_{H_k(i)}\}, k = 1...K$ (nearest hits) and the K closest examples of a different class $\{\boldsymbol{x}_{M_k(i)}\}$ (nearest misses.)[8] Then, in projection on feature j, the sum of the distances between the examples and their nearest misses is compared to the sum of distances to their nearest hits. In Equation 2, we use the ratio of these two quantities to create an index independent of feature scale variations. The Relief method works for multi-class problems.

3.3 Redundant features

Another justification of multivariate methods is that they take into account feature redundancy and yield more compact subsets of features. Detecting redundancy cannot be done by analyzing only the feature projections, as univariate methods do. This point is illustrated in the following examples.

Noise reduction can be achieved with features having identical projected distributions. In Figure 2-d, the two features look similar if we compare their projected distributions. Yet they are not completely redundant: the two-dimensional distribution shows a better class separation than the one achievable with either feature. In this example the data points of the two classes are generated from Gaussian distributions with equal variance σ^2. In projection on either feature, the distance d between the two classes is identical. The signal to noise ratio of each individual feature is therefore d/σ. In projection on the first diagonal, the distance between the two classes is $d\sqrt{2}$, hence the signal-to-noise ratio is improved by $\sqrt{2}$. Adding n features having such class conditional independence would result in an improvement of the signal-to-noise ratio by $\sqrt{n}$.

Correlation does NOT imply redundancy. Figures 2-e and Figure 2-f show even more striking examples in which the feature projections are the same as in Figure 2-d. It is usually thought that feature correlation (or anti-correlation) means feature redundancy. In Figure 2-f, the features are correlated and indeed redundant: the class separation is not significantly improved by using two features rather than one. But in Figure 2-e, despite that two

[8] All features are used to compute the closest examples.

features have similar projections and are anti-correlated, they are not redundant at all: a perfect separation is achieved using the two features while each individual feature provides a poor separation.

3.4 Forward and backward procedures

Having recognized the necessity of selecting features in the context of other features and eliminating redundancy, we are left with a wide variety of algorithms to choose from. Among wrapper and embedded methods (Chapter 4 and Chapter 5), greedy methods (forward selection or backward elimination) are the most popular. In a forward selection method one starts with an empty set and progressively add features yielding to the improvement of a performance index. In a backward elimination procedure one starts with all the features and progressively eliminate the least useful ones. Both procedures are reasonably fast and robust against overfitting. Both procedures provide nested feature subsets. However, as we shall see, they may lead to different subsets and, depending on the application and the objectives, one approach may be preferred over the other one. We illustrate each type of procedure with examples of algorithms.

Forward or backward? In Figures 2-g and h, we show an example in three dimensions illustrating differences of the forward and backward selection processes. In this example, a forward selection method would choose first x_3 and then one of the two other features, yielding to one of the orderings x_3, x_1, x_2 or x_3, x_2, x_1. A backward selection method would eliminate x_3 first and then one of the two other features, yielding to one of the orderings x_1, x_2, x_3 or x_2, x_1, x_3. Indeed, on Figure 2-h, we see that the front projection in features x_1 and x_2 gives a figure similar to Figure 2-e. The last feature x_3 separates well by itself, better than x_1 or x_2 taken individually. But, combined with either x_1 or x_2, it does not provide as good a separation as the pair $\{x_1, x_2\}$. Hence, the forward selection ordering yields a better choice if we end up selecting a single feature (the top ranking x_3), but the backward selection method will give better results if we end up selecting two features (the top ranking x_1 and x_2). Backward elimination procedures may yield better performances but at the expense of possibly larger feature sets. However if the feature set is reduced too much, the performance may degrade abruptly. In our previous example, choosing the top ranking feature by backward selection would be much worse than choosing x_3 as given by the forward approach.

Forward selection algorithm examples are now provided. The *Gram-Schmidt orthogonalization* procedure is a simple example of *forward selection* method (see Chapter 2 for details and references.) The first selected feature has largest cosine with the target. For centered features, this is equivalent to selecting first the feature most correlated to the target (Equation 1.) The subsequent features are selected iteratively as follows:

- the remaining features and the target are projected on the null space of the features already selected;

- the feature having largest cosine with the target in that projection is added to the selected features.

The procedure selects the features that incrementally decrease most the least-square error of a linear predictor. One can stop the procedure using a statistical test or by cross-validation (Chapter 2.) This procedure has the advantage to be described in few lines of codes and it performs well in practice. We give a Matlab implementation of the algorithm in Appendix A. It is worth noting the similarity with the Partial Least Square (PLS) method (see e.g. (Hastie et al., 2000)): both methods involve iteratively the computation of the correlation of the (projected) input features with the target, followed by a new projection on the null space of the features selected; the difference is that, in Gram-Schmidt, original input features are selected while in PLS the features selected are constructed as a weighted sum of the original features, the weights being given by the correlation to the target.

Another more advanced example of forward selection method is "Random Forests" or RF. Ensembles of decision trees (like Random Forests (Breiman, 2001)) select features in the process of building a classification or regression tree. A free RF software package is available from `http://www.stat.berkeley.edu/users/breiman/RandomForests/` and a Matlab interface from `http://sunsite.univie.ac.at/statlib/matlab/RandomForest.zip`.

Backward elimination algorithm examples are now provided. The *recursive feature elimination Support Vector Machine (RFE-SVM)* is a simple example of *backward elimination* method (see Chapter 5 for details and references.) For the linear SVM having decision function $f(\boldsymbol{x}) = \boldsymbol{w} \cdot \boldsymbol{x} + b$, the method boils down to simply iteratively removing the feature x_i with the smallest weight in absolute value $|w_i|$ and retraining the model.[9] At the expense of some sub-optimality, the method can be sped up by removing several features at a time at each iteration. The method can also be extended to the non-linear SVM (Chapter 5.) SVMs are described in Chapter 1 and numerous free software packages are available (see `http://www.kernel-machines.org/`) which makes this approach rather simple in terms of implementation.

RFE is a weight pruning method according to the smallest change in objective function. It follows the same paradigm than the Optimal Brain Damage procedure (OBD), which is used to prune weights in neural networks and can be used for feature selection. OBD also bears resemblance with the Automatic Relevance Determination (ARD) Bayesian method used by the winners of the competition (see Chapter 7 and Chapter 10 for details.)

3.5 Recapitulation

Table 1 summarizes the methods mentioned in this section. We recommend to try the methods in order of increasing statistical complexity:

[9] RFE usually works best on standardized features, see Section 1.2.

Feature selection	Matching classifier	Computational complexity	Comments
Pearson (Eq. 1)	Naïve bayes	nm	Feature ranking filter. Linear univariate. Makes independence assumptions between features. Low computational and statistical complexity.
Relief (Eq. 2)	Nearest neighbors	nm^2	Feature ranking filter. Non-linear multivariate. Statistical complexity monitored by the number of neighbors.
Gram-Schmidt (Sec. 3.4)	linear RLSQ	fnm	Forward selection, stopped at f features. Linear multivariate. The statistical complexity of RLSQ monitored by the regularization parameter or "ridge".
RFE-SVM (Sec. 3.4)	SVM	$\max(n,m)m^2$	Backward elimination. Multivariate, linear or non-linear. Statistical complexity monitored by kernel choice and "soft-margin" constraints.
OBD/ARD	Neural Nets	$min(n,m)nmh$	Backward elimination. Non-linear multivariate. Statistical complexity monitored by the number h of hidden units and the regularization parameter or"weight decay".
RF	RF	$t\sqrt{n}\ m \log m$	Ensemble of t tree classifiers, each preforming forward selection. Non-linear multivariate.

Table 1. *Frequently used feature selection methods.* We use the abbreviations: RLSQ=regularized least square; RFE=recursive feature elimination; SVM=support vector machine; OBD=optimum brain damage; ARD=automatic relevance determination; RF=random forest. We call m the number of training examples and n the number of features. The computational complexity main vary a lot depending on the implementation and should be taken with caution.

1. **Univariate methods** making independence assumptions between variables. Feature selection: Ranking with the Pearson correlation coefficient. Classifier: Naïve Bayes.
2. **Linear multivariate methods.** Feature selection: Gram-Schmidt forward selection or RFE with linear SVM. Predictors: Linear SVM or linear regularized least-square model (RLSQ.)
3. **Non-linear multivariate methods.** Feature selection: Relief, RFE, OBD or ARD combined with non-linear models. Predictors: Nearest neighbors, non-linear SVM or RLSQ, neural network, RF.

Computational complexity is also sometimes of consideration. We have added to Table 1 some orders of magnitude of the computational complexity of the feature selection process. This does not include the assessment part determining the optimum number of features to be selected. Justifications of our estimates are provided in Appendix B.

4 Advanced topics and open problems

This book presents the status of a rapidly evolving field. The applications are driving this effort: bioinformatics, cheminformatics, text processing, speech processing, and machine vision provide machine learning problems in very high dimensional spaces, but often comparably few examples (hundreds). It may be surprising that there is still a profusion of feature selection methods and that no consensus seems to be emerging. The first reason is that there are several statements of the feature selection problem. Other reasons include that some methods are specialized to particular cases (e.g. binary inputs or outputs), some methods are computationally inefficient so they can be used only for small numbers of features, some methods are prone to "overfitting" so they can be used only for large numbers of training examples.

The fact that simple methods often work well is encouraging for practitioners. However, this should not hide the complexity of the problems and the challenges ahead of us to improve on the present techniques and consolidate the theory. Inventing a new algorithm is a good way to be acquainted with the problems. But there exist already so many algorithms that it is difficult to improve significantly over the state of the art without proceeding in a principled way. This section proposes some formal mathematical statements of the problems on which new theories can be built.

Let us first introduce some notations. A pattern is a feature vector $\boldsymbol{x} = [x_1, x_2, ...x_n]$, which is an instance of a random vector $\boldsymbol{X} = [X_1, X_2, ...X_n]$. For each assignment of values, we have a probability $P(\boldsymbol{X} = \boldsymbol{x})$. We assume that the values are discrete for notational simplicity. The target is a random variable Y taking values y. The dependency between $\boldsymbol{X}$ and Y is governed by the distribution $P(\boldsymbol{X} = \boldsymbol{x}, Y = y) = P(Y = y|\boldsymbol{X} = x)P(\boldsymbol{X} = x)$. When we write $P(\boldsymbol{X}, Y) = P(Y|\boldsymbol{X})P(\boldsymbol{X})$, we mean that the equality hold true for all the values taken by the random variables. Let $\boldsymbol{V}$ be some subset of $\boldsymbol{X}$. Let $\boldsymbol{X}^{-i}$ be the subset of $\boldsymbol{X}$ excluding x_i and $\boldsymbol{V}^{-i}$ be some subset of $\boldsymbol{X}^{-i}$.

4.1 Relevant features

We start with the notion of relevant feature. We first define irrelevance as a consequence of random variable independence and then define relevance by contrast. First we assume the knowledge of the data distributions, which in reality are unknown. We then discuss what can be done in the finite sample case.

Definition 1 (Surely irrelevant feature). *A feature X_i is surely irrelevant iff for all subset of features $\boldsymbol{V}^{-i}$ including $\boldsymbol{X}^{-i}$,*

$$P(X_i, Y|\boldsymbol{V}^{-i}) = P(X_i|\boldsymbol{V}^{-i})P(Y|\boldsymbol{V}^{-i}).$$

Since we care little about cases that occur with zero or small probability it seems natural to measure irrelevance in probability e.g. with the Kullback-Leibler divergence between $P(X_i, Y|\boldsymbol{V}^{-i})$ and $P(X_i|\boldsymbol{V}^{-i})P(Y|\boldsymbol{V}^{-i})$:

$$MI(X_i, Y|\boldsymbol{V}^{-i}) = \sum_{\{X_i,Y\}} P(X_i, Y|\boldsymbol{V}^{-i}) \log \frac{P(X_i, Y|\boldsymbol{V}^{-i})}{P(X_i|\boldsymbol{V}^{-i})P(Y|\boldsymbol{V}^{-i})}$$

The sum runs over all possible values of the random variables X_i and Y. We note that the expression obtained is the conditional mutual information. It is therefore a function of $n-1$ variables.[10] In order to derive a score that summarizes how relevant feature X_i is, we average over all the values of $\boldsymbol{V}^{-i}$:

$$EMI(X_i, Y) = \sum_{\boldsymbol{V}^{-i}} P(\boldsymbol{V}^{-i}) MI(X_i, Y|\boldsymbol{V}^{-i})$$

We define then:

Definition 2 (Approximately irrelevant feature). *A feature X_i is approximately irrelevant, with level of approximation $\epsilon > 0$ or ϵ-relevant, iff, for all subset of features $\boldsymbol{V}^{-i}$ including $\boldsymbol{X}^{-i}$,*

$$EMI(X_i, Y) \leq \epsilon.$$

When $\epsilon = 0$, the feature will be called **almost surely irrelevant**.

With that statement, conditional mutual information comes as a natural relevance ranking index and we may define relevance by contrast to irrelevance. The practical use of our definitions to perform feature selection is

[10] Recall that n is the total number of features.

computationally expensive since it requires considering all subset of features $\boldsymbol{V}^{-i}$ and summing over all the values of $\boldsymbol{V}^{-i}$. However if we assume that features X_i and X_j for all $i \neq j$, are independent, the average conditional mutual information is the same as the mutual information between X_i and Y:

$$EMI(X_i, Y) = MI(X_i, Y)$$

This motivates the following definition:

Definition 3 (Individually irrelevant feature). *A feature X_i is individually irrelevant iff for some relevance threshold epsilon* ≥ 0

$$MI(X_i, Y) \leq \epsilon.$$

The derivation of this definition justifies of the use of mutual information as a feature ranking index (see Chapter 6.)

The **finite sample case** is now discussed. In practical cases, we do not have access to the probability distributions $P(X)$ and $P(Y|X)$, but we have training examples drawn from these distributions. We define a new notion of probable approximate irrelevance. At the same time, we replace in our definition the criteria $EMI(X_i, Y)$ or $MI(X_i, Y)$ by a generic non-negative index $C(i)$ whose expected value is zero for irrelevant features. We write our index as $C(i, m)$ to emphasize that it is an empirical index computed from m training examples.

Definition 4 (Probably approximately irrelevant feature). *A feature i is probably approximately irrelevant with respect to an index C estimated with m examples, with level of approximation $\epsilon \geq 0$ and risk $\delta \geq 0$ iff*

$$\mathrm{P}(C(i, m) > \epsilon(\delta, m)) \leq \delta.$$

Clearly, for relevant features, we do not know the probability distribution of $C(i, m)$ across different drawings of the training set of size m, so it does not seem that we have progressed very much. However, we may be able to make some assumptions about the distribution of C for irrelevant features. Following the paradigm of hypothesis testing, we call the distribution of C for irrelevant features the "null" distribution. For a given candidate feature i, the null hypothesis is that this feature is irrelevant. We will reject this null hypothesis if $C(i, m)$ departs significantly from zero. Using the "null" distribution and a chosen risk δ, we can compute the significance threshold $\epsilon(\delta, m)$. This method of assessing the statistical significance of feature relevance is further developed in Chapter 2.

Discussion. Many definitions of relevance have been provided in the literature. Kohavi and John (Kohavi and John, 1997) make a distinction between strongly and weakly relevant features. We recall below those definitions:

> A feature X_i is strongly relevant iff there exists some values x_i, y and $\boldsymbol{v}_i$ with $P(X_i = x_i, \boldsymbol{X}^{-i} = \boldsymbol{v}_i) > 0$ such that: $P(Y = y|X_i = x_i, \boldsymbol{X}^{-i} = \boldsymbol{v}_i) \neq P(Y = y|\boldsymbol{X}^{-i} = \boldsymbol{v}_i)$. A feature X_i is weakly relevant iff it is not strongly relevant and if there exists a subset of features $\boldsymbol{V}^{-i}$ for which there exists for values x_i, y and $\boldsymbol{v}_i$ with $P(X_i = x_i, \boldsymbol{V}_i = \boldsymbol{v}_i) > 0$ such that: $P(Y = y|X_i = x_i, \boldsymbol{V}_i = \boldsymbol{v}_i) \neq P(Y = y|\boldsymbol{V}_i = \boldsymbol{v}_i)$.

Our asymptotic definitions of relevance are similarly based on conditioning. Kohavi and John's introduction of strong and weak relevance seems to have been guided by the need to account for redundancy: the strongly relevant feature that is needed on its own and cannot be removed, and the weakly relevant feature that is redundant with other relevant features and can therefore be omitted if similar features are retained. Our approach separates the notion of redundancy from that of relevance: A feature is relevant if it contains some information about the target. Since our definition of relevance is less specific, we introduce in Section 4.2 the notion of *sufficient feature subset*, a concept to extract a minimum subset of relevant feature and therefore to rule out redundancy when required.

4.2 Sufficient feature subset

In the previous section, we have provided formal definitions for the notion of feature relevance. As outlined in section 3.3, relevant features may be redundant. Hence, a ranking of features in order of relevance does not allow us to extract a minimum subset of features that are sufficient to make optimal predictions. In this section, we propose some formal definitions of feature subset sufficiency. We introduce the additional notation $\bar{\boldsymbol{V}}$ for the subset that complements a set of feature $\boldsymbol{V}$ in $\boldsymbol{X}$: $\boldsymbol{X} = [\boldsymbol{V}, \bar{\boldsymbol{V}}]$.

Definition 5 (Surely sufficient feature subset). *A subset $\boldsymbol{V}$ of features is surely sufficient iff, for all assignments of values to its complementary subset $\bar{\boldsymbol{V}}$,*

$$P(Y|\boldsymbol{V}) = P(Y|\boldsymbol{X}).$$

As in the case of the definition of feature relevance, since we care little about cases that occur with zero or small probability, it seems natural to measure sufficiency in probability. We define a new quantity:

$$DMI(\boldsymbol{V}) = \sum_{\{\boldsymbol{v}, \bar{\boldsymbol{v}}, y\}} P(\boldsymbol{X} = [\boldsymbol{v}, \bar{\boldsymbol{v}}], Y = y) \log \frac{P(Y = y|\boldsymbol{X} = [\boldsymbol{v}, \bar{\boldsymbol{v}}])}{P(Y = y|\boldsymbol{V} = \boldsymbol{v})}.$$

This quantity, introduced in (Koller and Sahami, 1996), is the expected value over $P(\boldsymbol{X})$ of the Kullback-Leibler divergence between $P(Y|\boldsymbol{X})$ and $P(Y|\boldsymbol{V})$. It can be verified that:

$$DMI(\boldsymbol{V}) = MI(\boldsymbol{X}, Y) - MI(\boldsymbol{V}, Y).$$

Definition 6 (Approximately sufficient feature subset). *A subset $\boldsymbol{V}$ of features is approximately sufficient, with level of approximation $\epsilon \geq 0$, or* **ϵ-sufficient**, *iff,*

$$DMI(\boldsymbol{V}) \leq \epsilon.$$

If $\epsilon = 0$ the subset $\boldsymbol{V}$ will be called **almost surely sufficient**.

Definition 7 (Minimal approximately sufficient feature subset). *A subset $\boldsymbol{V}$ of features is minimal approximately sufficient , with level of approximation $\epsilon \geq 0$ iff it is ϵ-sufficient and there does not exist other ϵ-sufficient subsets of smaller size.*

From our definition, it follows that a minimal approximately sufficient feature subset is a solution (probably not unique) to the optimization problem:

$$\min_{\boldsymbol{V}} \|\boldsymbol{V}\|_0 \quad \text{such that} \quad DMI(\boldsymbol{V}) \leq \epsilon,$$

where $\|\boldsymbol{V}\|_0$ denotes the number of features selected. Such optimization problem can be transform via the use of a Lagrange multiplier $\lambda > 0$ into:

$$\min_{\boldsymbol{V}} \|\boldsymbol{V}\|_0 + \lambda \; DMI(\boldsymbol{V}).$$

Noting that $MI(\boldsymbol{X}, Y)$ is constant, this is equivalent to:

$$\min_{\boldsymbol{V}} \|\boldsymbol{V}\|_0 - \lambda \; MI(\boldsymbol{V}, Y).$$

We recover the feature selection problem stated in Chapter 6: find the smallest possible feature subset that maximizes the mutual information between the feature subset and the target.

We remark that the quantity $\|\boldsymbol{V}\|_0$ is discrete and therefore difficult to optimize. It has been suggested (Tishby et al., 1999) to replace it by $MI(\boldsymbol{X}, \boldsymbol{V})$. As noted in section 3.1, the prediction of posterior probabilities is a harder problem than classification or regression. Hence, we might want to replace the problem of maximizing mutual information by that of minimizing a given risk functional, e.g. the classification error rate. The formulation of the "zero-norm" feature selection method follows this line of thoughts (see Chapter 5.)

4.3 Variance of feature subset selection

If the data have redundant features, different subsets of features can be equally efficient. 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 developed in Chapter 7 is to use ensemble methods. The feature selection process may be repeated e.g.

with sub-samples of the training data. The union of the subsets of features selected may be taken as the final "stable" subset. An index of relevance of individual features can be created considering how frequently they appear in the selected subsets.

This approach has shown great promises but the following limitation is worth mentioning: when one feature that is highly relevant by itself is complemented by many alternative features having weak individual relevance, the highly relevant feature will easily emerge from the procedure while the weak features will be difficult to differentiate from irrelevant features. This may be detrimental to performances.

4.4 Suggested problems

Before closing this chapter, we would like to describe some research directions that we believe deserve attention.

More theoretically grounded algorithms. A lot of popular algorithms are not principled and it is difficult to understand what problem they seek to solve and how optimally they solve it. It is important to start with a clean mathematical statement of the problem addressed (see Sections **??** and 4.2 for preliminary guidelines.) It should be made clear how optimally the chosen approach addresses the problem stated. Finally, the eventual approximations made by the algorithm to solve the optimization problem stated should be explained. An interesting topic of research would be to "retrofit" successful heuristic algorithms in a theoretical framework.

Better estimation of the computational burden. Computational considerations are fairly well understood. But, even though the ever increasing speed of computers lessens the importance of algorithmic efficiency, it remains essential to estimate the computational burden of algorithms for feature selection problems. The computational time is essentially driven by the search strategy and the evaluation criterion. Several feature selection methods require examining a very large number of subsets of features, and possibly, all subsets of features, i.e. 2^n subsets. Greedy methods are usually more parsimonious and visit only of the order of n or n^2 subsets. The evaluation criterion may also be expensive as it may involve training a classifier or comparing every pairs of examples or features. Additionally, the evaluation criterion may involve one or several nested cross-validation loops. Finally, ensemble methods offer performance increases at the expense of additional computations.

Better performance assessment of feature selection. The other important question to be addressed is of statistical nature: some methods require more training examples than others to select relevant features and/or obtain good predictive performances. The danger of "overfitting" is to find features that "explain well" the training data, but have no real relevance or no predictive power. Making theoretical predictions on the number of examples needed to "solve" the feature selection problem is essential both to select an appropriate feature selection method and to plan for future data acquisition. Initial results to tackle this problem are found e.g. in (Almuallim and Dietterich, 1991) and (Ng, 1998).

The sagacious reader will have noticed that we did not treat the finite sample case in Section 4.2 for "sufficient feature subsets". There is still a lack of adequate formalism. We argue that in the finite sample case, feature subsets that are NOT sufficient may yield better performance than sufficient subsets (even if they are minimal and contain no irrelevant feature) because further reducing the space dimensionality may help reducing the risk of overfitting. In line with the "wrapper" methodology (Kohavi and John, 1997), it might be necessary to introduce a notion of "efficient feature subset": a subset providing best expected value of the risk when the learning machine is trained with a finite number m of examples. One central issue is to devise performance bounds characterizing efficient feature subsets.

Other challenges. Although we have made an effort in this introduction and in the book to cover a large number of topics related to feature extraction, we have not exhausted all of them. We briefly list some other topics of interest.

- *Unsupervised variable selection.* Several authors have attempted to perform feature selection for clustering applications (see, e.g., Xing and Karp, 2001, Ben-Hur and Guyon, 2003, and references therein). For supervised learning tasks, one may want to pre-filter a set of most significant variables with respect to a criterion which does not make use of y to lessen the problem of overfitting.
- *Selection of examples.* The dual problems of feature selection/construction are those of example selection/construction. Mislabeled examples may induce the choice of wrong variables, so it may be preferable to perform jointly the selection of variables and examples.
- *Reverse engineering the system.* Our introduction focuses on the problem of constructing and selecting features useful to build a good predictor. Unraveling the causal dependencies between variables and reverse engineering the system that produced the data is a far more challenging task (see, e.g., Pearl, 2000) that goes beyond the scope of this book.

5 Conclusion

We have presented in this introductions many aspects of the problem of feature extraction. This book covers a wide variety of topics and provides access to stimulating problems, particularly via the feature selection challenge, which is the object of Part II of the book. Simple but effective solutions have been presented as a starting point. The reader is now invited to study the other chapters to discover more advanced solutions. We have indicated a number of open problems to challenge the reader to contribute to this rapidly evolving field.

Acknowlegments

We are grateful to Eugene Tuv for providing us information on the computational complexity of RF.

References

H. Almuallim and T. G. Dietterich. Learning with many irrelevant features. In *Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI-91)*, volume 2, pages 547–552, Anaheim, California, 1991. AAAI Press.

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.

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

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

I. Guyon. Design of experiments of the NIPS 2003 variable selection benchmark. *http://www.nipsfsc.ecs.soton.ac.uk/papers/NIPS2003-Datasets.pdf*, 2003.

I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *JMLR*, 3:1157–1182, March 2003.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning, Data Mining, Inference and Prediction*. Springer Verlag, 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 249–256, 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.

J. Kruskal and M. Wish. *Multidimensional Scaling*. Sage Publications, 1978.

H. Liu and H. Motoda. *Feature Extraction, Construction and Selection: A Data Mining Perspective*. Kluwer Academic, 1998.

R. G. Lyons. *Understanding Digital Signal Processing*. Prentice Hall, 2004.

D. J. C. MacKay. Bayesian non-linear modeling for the energy prediction competition. *ASHRAE Transactions*, 100:1053–1062, 1994.

R. M. Neal. Defining priors for distributions using dirichlet diffusion trees. Technical Report 0104, Dept. of Statistics,University of Toronto, 2001.

R. M. Neal. *Bayesian Learning for Neural Networks*. Number 118 in Lecture Notes in Statistics. Springer-Verlag, New York, 1996.

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.

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

R. E. Woods R. C. Gonzalez. *Digital Image Processing*. Prentice Hall, 1992.

P. Soille. *Morphological Image Analysis*. Springer-Verlag, 2004.

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.

J. S. Walker. *A primer on wavelets and their scientific applications*. Chapman and Hall/CRC, 1999.

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.