# Feature Ranking Using an EDA-based Wrapper Approach

Yvan Saeys, Sven Degroeve, and Yves Van de Peer

Department of Plant Systems Biology, Ghent University, Flanders Interuniversity Institute for Biotechnology (VIB), Technologiepark 927, B-9052 Ghent, Belgium
{yvan.saeys,sven.degroeve,yves.vandepeer}@psb.ugent.be

**Summary.** Feature subset selection is an important pre-processing step for classification. A more general framework of feature selection is feature ranking. A feature ranking provides an ordered list of the features, sorted according to their relevance. Using such a ranking provides a better overview of the feature elimination process, and allows the human expert to gain more insight into the processes underlying the data. In this chapter, we describe a technique to derive a feature ranking directly from the estimated distribution of an EDA. As an example, we apply the method to the biological problem of acceptor splice site prediction, demonstrating the advantages for knowledge discovery in biological datasets with many features.

## 1 Introduction

Reduction of data dimensionality has become an apparent need in machine learning during the past decades. Examples of large datasets with instances described by many features include problems in image processing, text mining and bioinformatics. To efficiently deal with such data, dimension reduction techniques emerged as a useful pre-processing step in the flow of data analysis. A subset of these techniques is referred to as feature (subset) selection techniques. These techniques differ from other reduction techniques (like projection and compression techniques) in that they do not transform the original input features, but merely select a subset of them.

The reduction of data dimensionality has a number of advantages: attaining good or even better classification performance with a restricted subset of features, faster and more cost-effective predictors, and the ability to get a better insight in the processes described by the data. An overview of feature selection techniques can be found in [10] and [5].

Techniques for feature selection are traditionally divided into two classes: filter approaches and wrapper approaches [12]. Filter approaches usually compute a feature relevance score such as the feature-class entropy, and remove low-scoring features. As such, these methods only look at the intrinsic properties of the dataset, providing a mechanism that is independent of the classification algorithm to be used afterwards. In the wrapper approach, various subsets of features are generated, and

Y. Saeys et al.: *Feature Ranking Using an EDA-based Wrapper Approach*, StudFuzz **192**, 243–258 (2006)
www.springerlink.com

evaluated using a specific classification model. A heuristic search through the space of all subsets is then conducted, using the classification performance of the model as a guidance to find promising subsets. In addition to filter and wrapper approaches, a third class of feature selection methods can be distinguished: embedded feature selection techniques [1]. In embedded methods, the feature selection mechanism is built into the classification model, making direct use of the parameters of the induction model to include or reject features. Examples of these methods are the pruning of decision trees, and recursive feature elimination (RFE) using the weight vector of a linear Support Vector Machine [6].

In this chapter, we will focus on the wrapper approach for feature selection. Wrapper based methods combine a specific classification model with a strategy to search the space of all feature subsets. Commonly used search strategies are sequential forward or backward selection [11], and stochastic iterative sampling methods like genetic algorithms (GA, [13]) or estimation of distribution algorithms (EDA, [14]). The EDA approach to feature selection is shown in Fig. 1. In this case, each individual in the population represents a feature subset, coded as a binary string. Each bit represents a feature, a 1 indicating the presence, a 0 the absence of a particular feature. Individuals are evaluated (step 2, Fig. 1) by training a classification model with the features present in the individual (i.e. the ones having a 1), and afterwards validating it, either by cross-validation on the training set, or by using a separate training and holdout set. The feature subset returned by the algorithm is then the best subset found during the search.

Instead of using the traditional crossover and mutation operators, inherent to GA, an EDA explicitly constructs a model of the selected feature subsets (step 4). Depending on the complexity of the model, univariate, bivariate or multivariate interactions between the encoded features are modelled. In a subsequent step (step 5), the new population is created by sampling feature subsets from this model. The new population can either be completely sampled from the distribution, or can partly consist of sampled subsets and subsets retained from the previous population (elitists). The use of EDAs for feature subset selection was pioneered in [8] and the use of EDAs for FSS in large scale domains, was reported to yield good results [9, 22].

## 2 EDA-based Feature Ranking

### 2.1 Feature Ranking

As mentioned in the introduction, the standard approach to using EDA for feature subset selection (FSS), is to select the best feature subset encountered in the iterative process as the final solution. However, selecting the single best subset of features provides a rather static view of the whole elimination process. When using FSS to gain more insight in the underlying processes, the human expert has no idea of the context of the specific subset. Questions about how much and which features can still be eliminated before the classification performance drastically drops down provide

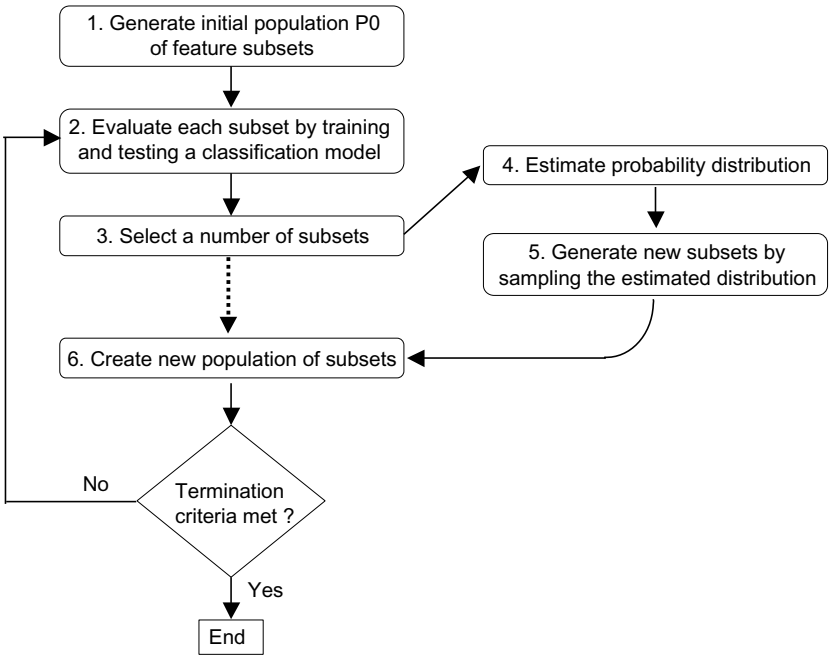



**Fig. 1.** The general scheme of the EDA approach to wrapper based feature selection

interesting information, yet remain unanswered using a static analysis. Feature ranking is a first step towards a dynamical analysis of the feature elimination process. The result of a feature ranking is an ordering of the features, sorted from the least relevant to the most relevant. Starting from the full/empty feature set, features can then be removed/added and the classification performance for each subset can be calculated, providing a dynamic view.

Traditional sequential wrapper algorithms such as sequential forward/backward search inherently provide a feature ranking. These algorithms either start from the full or empty feature set, and greedily add or discard one feature at the time. If this process is iterated until all features are added or removed, a complete view of the selection process can be obtained. A similar methodology can be applied in the case of most filter methods, where the feature relevance scores can be sorted and provide a feature ranking.

When using stochastic methods like GA or EDA, a hybrid approach can be used to yield a dynamical view of the selection process. The solution found by the evolutionary algorithm is then used as the starting point for a sequential forward or backward wrapper method. However, such practice may result in a large, sometimes unfeasible, number of additional calculations, depending on the number of features selected, or the range of the dynamic view.

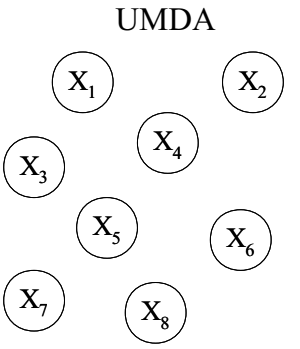
Instead of combining an evolutionary method with a sequential method into a hybrid, we present an EDA-based technique that directly results in a feature ranking. Instead of using a single best solution, we use the estimated probability distribution as a basis for the feature ranking. As a consequence, this technique does not require any additional calculations and, as all features are modelled in the estimated distribution, it provides a dynamic view of the whole selection process.

### 2.2 Deriving a Ranking from an EDA

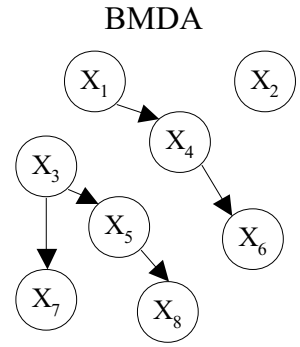
The main action to be taken in an EDA-based evolutionary algorithm is the construction of the probability distribution that models the variables and their dependencies. In general, most EDAs can be represented graphically as probabilistic graphical models [19]. The structure of the graphical model determines the expressive power of the EDA to model dependencies between variables, and constitutes the major criterion to distinguish subclasses of EDAs. The most common subclassification distinguishes between EDAs modelling univariate, bivariate and multivariate dependencies between the variables. A second aspect of the probabilistic graphical model is a set of generalized probability distributions, associated with the variables. Depending on the domain of the variables, these distributions can be either discrete or continuous. In the case of feature subset selection, all variables are discrete and binary. Fig. 2 shows a few examples of probabilistic graphical models for the three major classes of EDAs in the case of a feature selection problem with eight features $(X_1, \cdots, X_8)$. The notation $p(x_i^j)$ denotes the probability of feature $i$ having value $j$. As features are either present or absent, $j$ can only be 0 or 1.
The Univariate Marginal Distribution Algorithm (UMDA [18]) is a very simple model, assuming variables are independent. This is reflected in the structure of the graphical model, as no arcs between different variables are present, and the probability distributions do not contain conditional probabilities. In the Bivariate Marginal Distribution Algorithm (BMDA [21]), pairwise interactions between variables are modelled, and in the case of multiple dependencies, higher order interactions between the variables are modelled. Examples of these include the Bayesian Optimization Algorithm (BOA [20]) and the Estimation of Bayesian Networks Algorithm (EBNA [4]).

To derive a feature ranking from a probability distribution, some sort of importance or relevance score for each feature needs to be calculated. Evidently, a feature $i$ having a higher value for $p(x_i^1)$ could be considered more important than a feature $j$ with a lower value for $p(x_j^1)$. The generalized probabilities $p(x_i^1)$ can thus be considered as feature relevance scores, and a list of features sorted by these probabilities returns a feature ranking. The general algorithm to calculate such a ranking consists of the steps presented in Fig. 3.

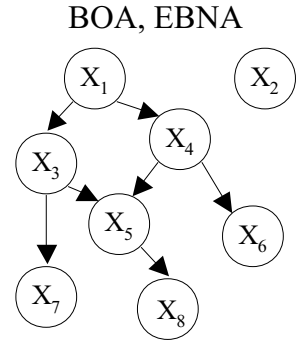
The most important step in this algorithm is the extraction of the probabilities $p(x_i^1)$ from the model. For models with univariate dependencies like the UMDA, the extraction of these probabilities is trivial, as they can be directly inferred from the model. For higher order EDAs like BMDA, BOA and EBNA, the probabilities $p(x_i^1)$




| $X_i$ | $p(x_i^j)$ j=0,1 |
|---|---|
| $X_1$ | $p(x_1^j)$ |
| $X_2$ | $p(x_2^j)$ |
| $X_3$ | $p(x_3^j)$ |
| $X_4$ | $p(x_4^j)$ |
| $X_5$ | $p(x_5^j)$ |
| $X_6$ | $p(x_6^j)$ |
| $X_7$ | $p(x_7^j)$ |
| $X_8$ | $p(x_8^j)$ |




| $X_i$ | $p(x_i^j)$ j,k=0,1 |
|---|---|
| $X_1$ | $p(x_1^j)$ |
| $X_2$ | $p(x_2^j)$ |
| $X_3$ | $p(x_3^j)$ |
| $X_4$ | $p(x_4^j \mid x_1^k)$ |
| $X_5$ | $p(x_5^j \mid x_3^k)$ |
| $X_6$ | $p(x_6^j \mid x_4^k)$ |
| $X_7$ | $p(x_7^j \mid x_3^k)$ |
| $X_8$ | $p(x_8^j \mid x_5^k)$ |




| $X_i$ | $p(x_i^j)$ j,k,l=0,1 |
|---|---|
| $X_1$ | $p(x_1^j)$ |
| $X_2$ | $p(x_2^j)$ |
| $X_3$ | $p(x_3^j)$ |
| $X_4$ | $p(x_4^j \mid x_1^k)$ |
| $X_5$ | $p(x_5^j \mid x_3^k, x_4^l)$ |
| $X_6$ | $p(x_6^j \mid x_4^k)$ |
| $X_7$ | $p(x_7^j \mid x_3^k)$ |
| $X_8$ | $p(x_8^j \mid x_5^k)$ |

**Fig. 2.** Some examples of probabilistic graphical models for EDAs with varying complexity: univariate dependencies (UMDA), bivariate dependencies (BMDA) and multiple dependencies (BOA, EBNA). The probability distributions are illustrated for a problem with discrete, binary variables, e.g. FSS. The notation $x_i^j$ denotes the instantiation of variable $i$ with value $j$

EDA-R

1. Select $S$ individuals from the final population $D_{\texttt{final}}$
2. Construct the probability model $P$ from $D^{S_j}_{\texttt{final}}, j \in 1, \cdots, S$ using an EDA (e.g. UMDA, BMDA, BOA/EBNA)
3. For each variable (feature) $X_i$, calculate the probability $p(x_i^1)$
4. Sort features $X_1, \cdots, X_n$ by their $p(x_i^1)$ probabilities
5. Write out the array of sorted features

**Fig. 3.** General algorithm to calculate a feature ranking (EDA-R)

need to be calculated in a forward manner, as they may involve conditional probabilities. To enable this, an ancestral ordering of the nodes in the graphical model is needed. The probabilities of nodes without ancestors are calculated first. Afterwards, probabilities for nodes depending on these ancestors can be calculated, followed by the probabilities of their descendants. This process is repeated in a forward manner, until all probabilities are calculated. It has to be noted that an ancestral ordering is not unique [7], yet the forward procedure of calculating the probabilities results in a unique probability distribution. For the example network of BOA and EBNA in Fig. 2, a possible ancestral ordering of the nodes is $X_1$,$X_2$,$X_3$,$X_4$,$X_5$,$X_6$,$X_7$,$X_8$. Another possible ordering would be for example $X_1$,$X_3$,$X_4$,$X_6$,$X_5$,$X_8$,$X_7$,$X_2$.

Conceptually, the idea of EDA-based feature ranking is based on a balance between two characteristics of the population. On the one hand, the population should consist of medium to good quality solutions, implying that already some sort of convergence has been accomplished. On the other hand, the population should still preserve some diversity, implying that it has not fully converged yet (e.g. in the ultimate case of convergence all individuals in the populations are the same). Thus, we seek a measure to define how long the iterative process should be continued, resulting in a population that has already converged, but not too much.

To quantify this idea of "convergence" of a population we need a measure of how similar/diverse it is. In the case of feature selection, the individuals are represented by bitstrings, and we can use the Hamming distance as a measure of distance between two individuals [15]. The Hamming distance $HD(x, y)$ between two bitstrings of length $N$ is the number of bits in which the two strings differ. This number thus varies between 0 and $N$. To normalize this number we calculate the scaled hamming distance as

$$HD_s(x, y) = \frac{HD(x, y)}{N} \tag{1}$$

The convergence of a population can then be calculated as the average scaled hamming distance between all pairs of individuals. For a population $P$ of size $S$ the convergence is calculated as

$$C(P) = \frac{2\left(\sum_{i=1}^{S-1}\sum_{j=i+1}^{S} HD_s(\mathbf{x}_i, \mathbf{x}_j)\right)}{S\,(S-1)} \tag{2}$$

where $\mathbf{x}_i$ denotes the $i\text{th}$ individual in the population. The parameter $C(P)$ can then be monitored during the stochastic iterative sampling process, which can be stopped when $C(P)$ falls below an a priori specified threshold. The calculation of this threshold can be done using a sub-sample of the data.

A more advanced, yet computationally more expensive way of tuning the parameter $C(P)$ can be thought of, stopping the iterative procedure automatically when the "optimal" convergence point has been reached. This can be done by calculating at each iteration the feature ranking curve, and tracking the area under this curve. A simple greedy heuristic can then be used to halt the iterative process when the area under the ranking curve of the current iteration is less then the area under the ranking curve of the previous iteration.

### 2.3 Deriving a Feature Weighting Scheme from an EDA

In the previous section, we described how a feature ranking could be derived from the generalized probabilities $p(x_i^1)$. However, these probabilities could also be directly used as feature relevance scores, or feature weights. In this way, we can construct a wrapper based feature weighting mechanism. The derivation of feature weights conveys important additional information, that can be used to gain new insights in the processes that generated the data (knowledge discovery). We will elaborate further on that aspect in the second part of this chapter, where we will discuss the application of the method to a biological classification problem.

The advantage of using EDA-R as a feature weighting mechanism, compared to other feature weighting methods like filter methods, is that it can directly use the feedback (classification performance) of classifiers that allows modelling of higher order dependencies, whereas most filter methods only determine the relevance of each feature by itself. As a direct extension of using EDA-R to rank individual features, it can be easily seen that the method can be generalized to subsets of $k$ features (e.g. weighting of all pairs, triples, of features). Thus the method can be easily extended to *feature subset weighting*. Another possibility for future extensions of this method is by incorporating a more sophisticated weighting scheme, and also taking into account the fitness of the selected individuals when assigning the feature weights.

## 3 A Real-World Application: Acceptor Splice Site Prediction

Recent advances in genomics have generated large amounts of biological sequence data. An important problem in bioinformatics is to analyse these sequences and predict the location and structure of genes, often referred to as *gene prediction*. Because the problem of correctly predicting genes is quite complex [16], gene prediction systems have a modular structure, combining the outputs of several components that are specialized in recognizing specific structural elements of a gene. An example of such structural elements are the so-called splice sites. These sites are the boundaries between coding and non coding regions in the genomes of higher organisms

(eukaryotes), and are of key importance in identifying the correct gene structure. In this chapter we will focus on acceptor splice site prediction, which is the transition from a non coding region (intron) to a coding region (exon). Acceptor splice sites are characterized by the fact that they have a conserved AG subsequence at the intron border side. As a result, acceptor prediction can be formally stated as a two-class classification task: given an AG subsequence, predict whether it is a true acceptor splice site or not. In this chapter we will focus on the prediction of acceptor splice sites in the plant model species *Arabidopsis thaliana*.

The *Arabidopsis thaliana* data set was generated from sequences that were retrieved from the EMBL database, and contained only experimentally validated genes (i.e. no genes that resulted from a prediction). Redundant genes were excluded, and splice site datasets were constructed from 1495 genes. More details on how these datasets were generated can be found in [2].

Because in real sequences, the number of true acceptor sites is largely outnumbered by the number of false acceptor sites, we chose to enforce a *class imbalance* in our datasets for feature selection. We constructed a dataset of 6000 positive instances and 36,000 negative instances. To obtain stable solutions for feature selection, a 10-fold validation of this dataset was used to test all feature selection methods. This was done by doing 5 replications of a two-fold cross-validation, maintaining the same class imbalance of 1 positive versus 6 negative instances in every partition. For the EDA-based wrapper approach, the internal evaluation of classification performance was obtained by doing a 5-fold cross-validation on the training set.

As the EDA-R method is a wrapper approach, it is specific to a fixed classification model. In our experiments, we used the Naive Bayes method (NBM [3]). This classification method follows the Bayes optimal decision rule, combining it with the assumption that the probability of the features given the class, is the product of the probabilities of the individual features (conditional independence assumption). The advantages of using NBM in the context of feature selection are its abilities to cope with high-dimensional data, its robustness and its speed. The latter aspect is of particular importance when using population based methods like EDA or GA , because for every individual, a classification model has to be trained, and tested by cross-validation. As a measure of classification performance, we used the F-measure [17] due to its ability to deal well with imbalanced datasets.

### 3.1 Feature Ranking for Acceptor Prediction

We start from the knowledge that the discrimination between true and false acceptor sites is determined by the part of the sequence where the site is located, more precisely the *local context* around the acceptor site. Therefore, the nucleotides A,T,C and G occurring on either side of the acceptor constitute a basic feature set. A local context of 100 nucleotides (50 to the left, 50 to the right) around the acceptor sites was chosen, having at each position one of the four nucleotides $\{$A,T,C,G$\}$. These features were extracted for the positive and negative instances, resulting in a dataset of 100 4-valued features, which were converted into binary format using

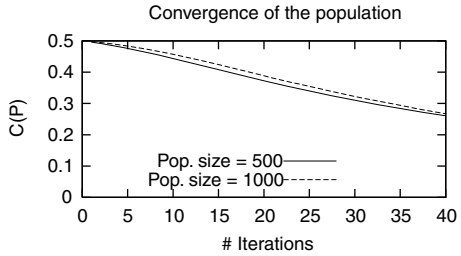



**Fig. 4.** Evolution of the population convergence as a function of the number of iterations

sparse vector encoding (A = 1000, T = 0100, C = 0010, G = 0001). This results in a dataset described by 400 binary features.

To this dataset, we applied the EDA-R feature ranking method. For different population sizes, ranging from 100 to 1000 individuals, we ran the experiments for 40 iterations. For each iteration $i$, we monitored the value of $C(P)$ and derived a feature ranking $F_i$. Afterwards, we compared the evaluations for each feature ranking.

The convergence of the population was calculated at each iteration using equation 2, and its evolution for population sizes of 500 and 1000 is shown in Fig. 4. The x-axis shows the number of iterations, while the y-axis shows the convergence value $C(P)$ of the population. At the beginning of the iterative process, the initial population consists of randomly generated feature subsets, where, for every feature, $p(x_i^1) = p(x_i^0) = 0.5$. As a result, feature subsets will have, on average, half of the features in common, and $C(P)$ will be approximately equal to 0.5. When the iterative process would be repeated ad infinitum, all individuals in the population would converge to the same individual, resulting in $C(P) = 0$. The figure shows that for 40 iterations, convergence will be roughly half way between 0.5 and 0.

As mentioned earlier, the ideal value of $C(P)$ is achieved when the population has already converged, yet not too much. To explore the effect of the number of iterations (and thus $C(P)$) on the feature ranking, we compared the evaluation of the feature ranking during the course of evolution. For a particular iteration number, we derived a feature ranking from the population at that time. This was done by starting with the full feature set, and iteratively eliminating the least relevant feature, according to the feature ranking. The results for a few iterations (iteration 1, 20 and 40) are shown in Fig. 5.

The left part of the figure shows the results for a population of 500 individuals, the right part for a population of 1000 individuals. The results after the first iteration are shown as a baseline result. As soon as the first iterations have passed, the feature ranking improves quickly, until at some point a good feature ranking is obtained (iteration 20). If the iterative process is then continued, populations that are too specific are obtained (iteration 40), characterized by the fact that classification performance drops down earlier when smaller feature sets are evaluated. Furthermore it can be observed that the results for a population size of 1000 individuals are only marginally better than the results using a population of 500 individuals. Gradually worse results are obtained when populations smaller than 500 individuals are used.

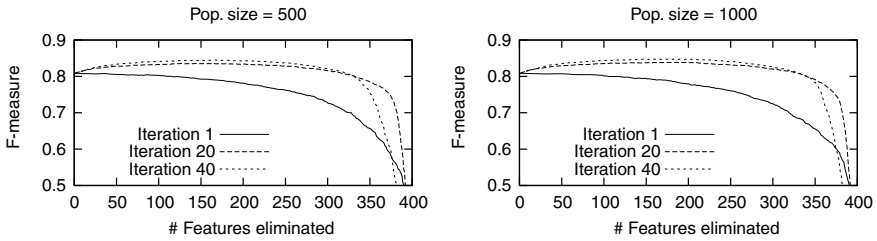



**Fig. 5.** Evaluation of a feature ranking for a number of iterations (1, 20 and 40). The left part shows the results for a population size of 500 individuals, the right part for a population of 1000 individuals. The origin represents the full feature set. The x-axis represents the number of features that have been eliminated thus far, while the y-axis shows the classification performance on the test set (F-measure)

It can be observed that, e.g. for the case of 20 iterations, many features can be eliminated before the classification performance drops down, showing that many irrelevant or redundant features are present in the dataset. The advantage of a feature ranking is the identification of a "break-point" region. This is the part of the graph where the classification performance drastically drops down, indicating the removal of strongly relevant features. It should be noted that the observation of a break-point region is strongly dependent on the dataset. For many biological processes, it is not completely known which features are relevant for the classification task at hand. Therefore, many potentially useful features are included in the dataset, hoping that the relevant features are included as well. As a result, many irrelevant or redundant features will be present, and a clear break-point can be observed. For other datasets with little or no redundant features, this phenomenon will not be observed.

Strictly speaking, the identification of the break-point region should be considered as a part of the training process. Therefore, the identification of this region should be done on the training set, and only thereafter the test set can be used for evaluation. Fig. 6 shows the results for feature selection on both the training and test set. When comparing the results, it can observed that the break-point regions for both data sets are very similar.

### 3.2 Feature Weighting for Knowledge Discovery in Acceptor Prediction

An important advantage of feature selection techniques is their ability to distinguish between relevant and irrelevant features, providing new insights in complex datasets.

It is known that correlations exist between nucleotides in the vicinity of splice sites. To detect these dependencies, higher-order (i.e. non-linear) classification methods can be used. When combining higher-order classification methods with EDA-based feature selection, this would require the use of higher-order estimation models. An example of such a combination could be a second order polynomial Support Vector Machine , in combination with the BMDA. However, using such higher order

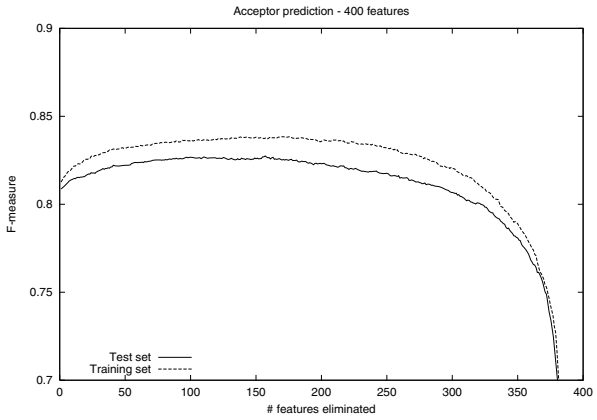



**Fig. 6.** Evaluation of the feature ranking obtained with a population size of 500 individuals after 20 iterations. Both the evaluation of the ranking on the training and on the test set are shown

classification algorithms and estimation models would make the EDA-R method very slow.

To circumvent the use of higher order models, yet still be able to extract correlations between nucleotides, we applied the following trick. We constructed an additional set of features that captures the nucleotide correlations already at the feature level. This has the advantage that linear models can still be used, while at the same time considering nucleotide dependencies. Another important advantage is that the combination with feature selection techniques allows us to select those dependencies that are of primary importance, and visualize them.

In addition to the simple nucleotide features used in the previous section, we added two layers of more complex features. The first layer captures the idea of compositional sequence information. These type of features extract sequence information that is position invariant. In our experiments, we included position invariant features of length 3, capturing the occurrence of subsequences of length 3 in the sequence neighbouring the splice site. An example of such a feature would be the occurrence of the subsequence "TCA" in the sequence to the left of the acceptor site. For the sequence on either side of the acceptor, including these features results in an additional set of 128 binary features, a 1 indicating the presence, a 0 the absence of the specific subsequence. The second layer of complexity comprises features that capture dependencies between adjacent nucleotides. To this end, we included all position dependent dinucleotides (subwords of length 2) in our analysis, resulting in an additional set of 1568 features. Summing up all features eventually results in a dataset described by 2096 features.

To this dataset we applied the EDA-R feature ranking method, deriving the feature weights from the probabilities $p(x_i^1)$ of the population at iteration 20 (for computational reasons we used the same EDA setting as in the previous experiment). A nice way of visualizing the feature weights is by color coding them using a heat

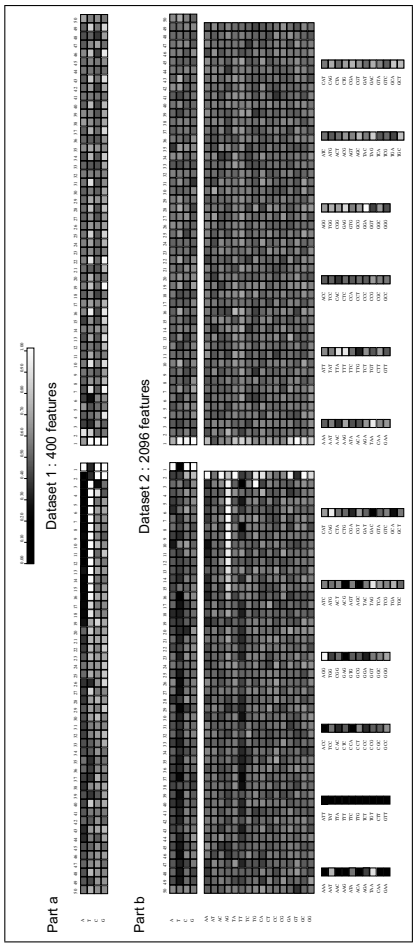



**Fig. 7.** Color coding of the feature weights using a gradient from black (the feature should be left out) to white (the feature should be included). Part 'a' shows the result for the simplest dataset (400 features), part 'b' shows the result for the complex dataset (2096 features). Features are grouped by their position relative to the acceptor site, which is denoted by the blank space in the middle. For each part of the context the position invariant features of length 3 are grouped according to their composition (A-rich, T-rich, C-rich, G-rich, equally distributed)

map, where a gradient ranging from black (the feature should be left out) to white (the feature should be included) shows the feature weights. This is shown in Fig. 7 where we graphically show the feature weights for both the simple dataset (400 features) and the extended dataset (2096 features). In this figure, features are shown row wise, while the columns indicate positions around the splice site (the gap in the middle). Part 'a' shows the results for the simplest dataset: every row represents one of the nucleotides A,T,C and G. Part 'b' shows the results for the complex dataset, with

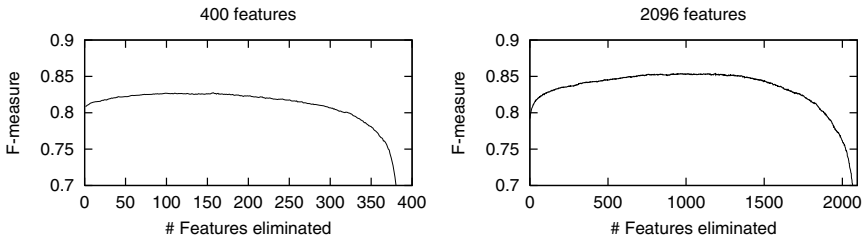



**Fig. 8.** Evaluation of the classification performance on both datasets, when features are iteratively discarded. The left part of the figure shows the result for the most simple dataset (400 features), while the right part shows the result for an extended version of this dataset, using also position invariant features and position dependent dinucleotide features (2096 features). The x-axis shows the number of features that have been eliminated, the y-axis shows the classification performance (F-measure)

the position invariant features (middle part) and the position dependent dinucleotides (lower part).

Several patterns can be observed. In both datasets, the nucleotides immediately surrounding the acceptor splice site are of key importance. Another pattern can be explained by looking at the right side of the context. In the simplest dataset, a clear periodical pattern is visible for the nucleotides T and G, capturing the fact that the right side of the context is a coding region (exon). In this region, nucleotides are organised in codons (triplets). However, as this is a general characteristic of the sequence, it is observed that the position invariant features in the second dataset better seem to grasp this characteristic.

Another important pattern are the nucleotides T at the left side of the acceptor for the simplest dataset. In the complex dataset, this pattern has completely disappeared, and is replaced by a stretch of AG dinucleotides that now seems to be most important. This is a nice example of the combination of a wrapper based method with NBM. It is known that the classification performance of NBM can be improved by discarding correlated features. Without going into much detail, we here mention that the presence of a poly-pyrimidine stretch (an excess of nucleotides C and T) to the left of the acceptor is correlated with the absence of AG dinucleotides in this part of the sequence. Apparently, NBM chooses the absence of AG dinucleotides as being more informative than the importance of T, and thus discards the T features to the left of the acceptor. The benefit of not including these features is even more apparent when looking at the position invariant features in part ‘b’, where all T-rich subsequences of length 3 are colored dark. This indicates that NBM strongly benefits from not including these features.

To verify that the features selected for the complex dataset are indeed better at describing the acceptor prediction problem, we compared the classification performance for both datasets (Fig. 8). This figure shows that better classification performance can be obtained using the more complex features.

## 4 Conclusions

In this chapter, we introduced two extensions of EDA-based feature selection: EDA-based feature ranking and EDA-based feature weighting. Using the EDA framework, these extensions can be naturally derived from the estimated distribution, and are immediately available when using simple EDA algorithms like the UMDA. We illustrated both techniques on a biological classification problem: the prediction of acceptor splice sites, an important subtask of gene prediction. Using the combination of more complex features and feature selection, we were able to extract a new, important feature for acceptor prediction: the inhibition of dinucleotides AG immediately upstream the acceptor site. Using feature selection allows us in this way to gain more insight in the computational modelling of this particular biological classification problem.

## References

1. A.I. Blum and P. Langley. Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 97:245–271, 1997.
2. S. Degroeve, B. De Baets, Y. Van de Peer, and P. Rouzé. Feature subset selection for splice site prediction. *Bioinformatics*, 18(2):75–83, 2002.
3. R. Duda and P. Hart. *Pattern Classification and Scene Analysis*. John Wiley and Sons, New York, 1973.
4. R. Etxeberria and P. Larrañaga. Global optimization with Bayesian networks. In *Proceedings of the Second Symposium on Artificial Intelligence. Special Session on Distributions and Evolutionary Optimization*, pp. 332–339, 1999.
5. I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003.
6. I. Guyon, J. Weston, S. Barnhill, and V.N. Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46(1-3):389–422, 2000.
7. M. Henrion. Propagating uncertainty in Bayesian networks by probabilistic logic sampling. In J. F. Lemmer and L. N. Kanal, editors, *Uncertainty in Artificial Intelligence*, volume 2, pp. 149–163. North-Holland, Amsterdam, 1988.
8. I. Inza, P. Larrañaga, R. Etxeberria, and B. Sierra. Feature subset selection by Bayesian network-based optimization. *Artificial Intelligence*, 123(1-2):157–184, 2000.
9. I. Inza, P. Larrañaga, and B. Sierra. Feature Subset Selection by Estimation of Distribution Algorithms. In P. Larrañaga and J. A. Lozano, editors, *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*, pp. 269–294. Kluwer Academic Publishers, 2001.
10. A. K. Jain, R. W. Duin, and J. Mao. Statistical Pattern Recognition. A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):4–37, 2000.
11. J. Kittler. Feature set search algorithms. In C.H. Chen, editor, *Pattern Recognition and Signal Processing*, pp. 41–60. Sithoff and Noordhoff, 1978.
12. R. Kohavi and G. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1-2):273–324, 1997.
13. M. Kudo and J. Sklansky. Comparison of algorithms that select features for pattern classifiers. *Pattern Recognition*, 33:25–41, 2000.

14. P. Larrañaga and J. A. Lozano. *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*. Kluwer Academic Press, 2001.
15. S.J. Louis and G.J.E. Rawlins. Predicting convergence time for genetic algorithms. Technical Report TR370, Indiana University, 1993.
16. C. Mathé, M.F. Sagot, T. Schiex, and P. Rouzé. Current methods of gene prediction, their strengths and weaknesses. *Nucleic Acids Research*, 30:4103–4117, 2002.
17. D. Mladenić and M. Grobelnik. Feature selection on hierarchy of web documents. *Decision Support Systems*, 35:45–87, 2003.
18. H. Mühlenbein. The equation for response to selection and its use for prediction. *Evolutionary Computation*, 5:303–346, 1998.
19. J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, 1988.
20. M. Pelikan, D.E. Goldberg, and E. Cantú-Paz. BOA: the Bayesian optimization algorithm. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 525–532, 1999.
21. M. Pelikan and H. Mühlenbein. The bivariate marginal distribution algorithm. In *Advances in Soft Computing-Engineering Design and Manufacturing*, pp. 521–535, 1999.
22. Y. Saeys, S. Degroeve, D. Aeyels, Y. Van de Peer, and P. Rouzé. Fast feature selection using a simple estimation of distribution algorithm: a case study on splice site prediction. *Bioinformatics*, 19(2):179–188, 2003.