International Journal of Pattern Recognition and Artificial Intelligence

FAST FEATURE SUBSET SELECTION IN BIOLOGICAL SEQUENCE ANALYSIS

RAINER PUDIMAT and ROLF BACKOFEN
Institut für Informatik, Albert-Ludwigs-Universität, Georges-Köhler-Allee 106, D-79110 Freiburg, Germany
{*rpudimat,backofen*}*@informatik.uni-freiburg.de*

ERNST G. SCHUKAT-TALAMAZZINI
Institut für Informatik, Friedrich-Schiller-Universität, Ernst-Abbe-Platz 3, D-07743 Jena, Germany
schukat@informatik.uni-jena.de

Motivation: Biological research produces a wealth of measured data. Neither it is easy for biologists to postulate hypotheses about the behaviour or structure of the observed entity because the relevant properties measured are not seen in the ocean of measurements. Nor it is easy to design machine learning algorithms to classify or cluster the data items for the same reason. Algorithms for automatically selecting a highly predictive subset of the measured features can help to overcome these difficulties.

Results: We present an efficient feature selection strategy which can be applied to arbitrary feature selection problems. The core technique is a new method for estimating the quality of subsets from previously calculated qualities for smaller subsets by minimising the mean standard error of estimated values with an approach common to support vector machines. This method can be integrated in many feature subset search algorithms. We have applied it with sequential search algorithms and have been able to reduce the number of quality calculations for finding accurate feature subsets by about 70%. We show these improvements by applying our approach to the problem of finding highly predictive feature subsets for transcription factor binding sites.

Keywords: computational biology; transcription factor binding sites; feature selection; combinatorial optimisation; linear predictors; combinatorial regression; kernel methods.

1. Introduction

The investigation of many biological processes and structures produces a wealth of data. In Microarray experiments, expression levels of thousands of genes are measured in parallel at different time points. Similar, high-dimensional data is pro-

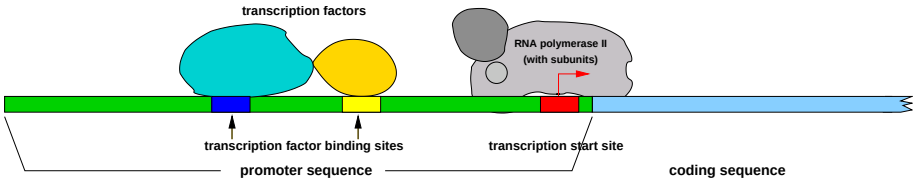

Fig. 1. Typical structure of a gene. The coding sequence is transcribed by the protein complex *RNA polymerase II* which binds at the transcription start site. The region upstream of this site is known as promoter which contains binding sites for several transcription factors. These factors interact with the polymerase complex in order to regulate gene expression.

duced by mass spectrometry in proteomic experiments. There is a vast number of proteins, substances and other metabolites that take part in a metabolic network. Not to mention the huge databases of biological sequences which are growing enormously.

Often, it is not apparent which features (data elements) contribute to an observed behaviour: Which genes are differentially expressed and therefore responsible for some disease? Which set of genes is characteristic for liver cells? What are the functional properties of a particular protein domain? Which features make the difference between a transcription factor binding site and an arbitrary DNA sequence?

Many of these questions are approached by using a *classification system*. The input of such a classification system is a vector of features (i.e. the measured values for these features). For each vector the system outputs a decision to which class the underlying biological object belongs. Beside time efficiency problems which most classification approaches have, when confronted with thousands of different features, the classification performance often increases when irrelevant and redundant features are eliminated from the input. Furthermore, the researcher who is interested in few but highly discriminating features will have it easier to identify simple rules about his researched biological structures or processes if the feature space is reduced to this relevant data. This preprocessing step of selecting a small subset of highly predictive features from a huge set of available features is known as *feature subset selection* (FSS).

In this paper we focus on a popular sequence analysis problem: the prediction of transcription factor binding sites (TFBS). Transcription factors (TF) regulate the gene transcription process by binding at regulatory DNA sequences (i.e. promoter and enhancers) upstream of transcription start site and interacting with subunits of *RNA polymerase II* and coacting transcription factors (see Figure 1). Each TF prefers to bind at its characteristic binding sequence but tolerates variations from this perfect site. The traditional way of modelling TFBS are position specific score matrices (PSSM) [24]. We have shown previously that PSSM do not cover all distinguishing features of these TFBS and that considering additional features in the modelling approach can improve the prediction performance.[18] Among these features are for instance sequence dependent structural parameters or base profiles for

a site's flanking regions. There, the most predicative features of TFBS and interdependencies among these features were modeled in Bayesian network classifiers. We were confronted with the problem of identifying a feature subset which is most accurate for predicting TFBS in promoter sequences from a huge set of possible features. For this purpose we apply feature subset selection algorithms.

It is common to distinguish two main strategies for performing feature subset selection: 1.) *filters* and 2.) *wrappers* [19]. Whereas filters work independent of any classification approach by applying a certain scoring function [30], wrappers evaluate a feature subset by learning a particular classifier in a cross validation scheme and measuring the classification error rate or a related statistic [8]. Filters can be further devided in *ranker methods* which calculate a ranking among all single features from which a promising feature subset can be chosen, and *non-rankers* which simply output a high-scoring feature subset without ranking the features. Furthermore, there are also rankers which rank with respect to a classifier. This last group is named *embedded* methods [10].

Naturally, filters usually run much faster than wrappers, since they do not have to perform cross validation including learning the underlying classifiers for each feature subset to be investigated. This is essentially true for datasets with a large number of samples. However wrappers outperform filters in their predictive performance since their feature subset evaluation is closer to the approached classification task (see result section for a comparison). The predictive performance of filters largely depends on the the applied scoring metric [26].

In this paper we present a third way: an approach which combines the advantages of both strategies. The main idea behind this approach is a new method for estimating objective function values of wrapper methods. The estimation relies on previously calculated objective function values for smaller feature subsets which have been calculated so far during the search process. Whenever the true value of the objective function is calculated, the estimator is updated for this new value, which improves further estimations. The technique relies on the formulation of estimated values as linear combination of previously calculated quality values. This weighted sum can be written as an inner product. Following the kernel trick of support vector machines,[2] the minimisation of the mean standard error of the estimate is reformulated in dual space.

We employ our estimation approach to a prominent representative among wrappers, the *sequential floating forward selection* (SFFS) [17]. Since the estimation is by far faster than the calculation of real objective function values, we are able to retrieve good feature subsets in a comfortable time, compared to the search algorithm without estimator. We demonstrate the performance of this estimation based search algorithms on two datasets of transcription factor binding sites, namely for the TF Sp1 and for AP1 boxes. Our approach finds optimal feature subsets in about a third of the time of the baseline search algorithms.

The organisation of the paper is as follows: Section 2 extends this introduction by showing the importance of FSS in biological sequence analysis on examples of

various fields. In section 3 we introduce the classification system for TFBS which includes a description of the feature space, the employed model for classification as well as the class of used feature selection algorithms. In section 4 we present our new technique for estimating feature subset qualities. Section 5 shows results of this approach, section 6 summarizes this article and gives further conclusions.

2. Importance of FSS in Biological Data Analysis

Due to its central role in improving classifier performance, FSS has recently been in focus of biological and medical research in various fields. Beside sequence related analysis, especially in the case of microarray or proteomic data analysis FSS techniques are of great importance. This kind of data suffers from the curse of dimensionality (thousands of features, few samples). A comparative study[13] for FSS algorithms investigated various feature selection strategies in order to reduce the dimensionality in drug discovery problems. There are several possible tasks for microarray data. One task is the classification of samples with respect to a particular phenotype (e.g. healthy vs. cancer). FSS is used here to rank the features (gene expression levels) with respect to their discriminative power [14,22,6,25,12]. The most discriminative genes are called *marker genes*. For proteins the term *biomarker* is common. Marker genes and biomarkers are crucial for the development of short-time-tests for certain diseases. Another kind of data analysis which is often performed on microarray and proteomic data is *clustering*. There, one is looking for subsets of features (again expression levels) which behave in a similar way over different time points. Results of such an analysis helps to eluciate regulatory networks. For clustering, FSS is used to reduce the influence of noisy genes and avoiding the obscuring evidential genes which are relevant only in conjunction with other features.[27,16]

At the end of this literature survey we like to mention further sequence analysis related FSS. Saeys et al. have employed FSS algorithms to select relevant feature subsets for predicting splice sites in RNA sequences of Arabidopsis thaliana.[20,3] In Ref. 32 similar problems are tackled with different techniques. Another study[31] employed FSS to model and predict translation start sites. More recently, Zhao et al. published a feature selection technique for the classification of protein sequences [33]. Even as an alternative to sequence alignment algorithms concerning gene sequence classification, FSS has been shown to be useful.[1]

3. Feature Subset Search

The classification task here is to predict proper TFBS in long DNA sequences, thus classifying each sequence position into either TFBS or *non*-TFBS. An optimal feature subset $\{F_1, F_2, \ldots, F_d\}$ for modelling TFBS consist of features F_i which all the TFBS's of a factor have in common and which, at the same time, they do not have in common with non-TFBS sequences. For the standard modelling approach, PSSM, these features are simply the nucleotides at the different positions.

In Ref. 18, we have introduced an approach that allows additional features and which outperforms PSSM in sense of prediction performance. The additional feature have been chosen to be from one of the following types:

- **Nucleotide at a certain position:** corresponds to the type of features used by PSSM. Clearly, it is possible to emulate usual PSSM with these features.
- **Structural property feature:** there are structural and chemical properties of DNA, which vary slightly with respect to the nucleotides occurring at a certain position. The mean value for these properties in a given subsequence are used as candidate features. Among the 38 different properties provided there are conformational parameters like *helical twist*, *helical slide* or *minor groove width*[4] and physico-chemical parameters like *free energy change* or *melting temperature*.
- **PSSM predictions for co-acting factor's site** TF often require the presence of a co-acting TF for fullfilling its biological function. A predicted TFBS for this co-acting TF in a closer neighborhood is evidence for a position to be a proper TFBS.
- **start positions of flexible matches of small words** measuring the left- or rightmost match of a small word can help to identify single point deletions in a subgroup of known TFBS.
- **Subsequence nucleotide profiles:** measure the coarse base composition of a given subsequence (e.g. a lower or higher A+T content of the flanking region).

The features found to be characteristic for TFBS are integrated as random variables in a Bayesian network (BN). A BN is a probabilistic graphical model which is an efficient choice of modelling the joint distribution of a set of random variables of diverse domains. Its efficiency its due to the decomposable nature of a BN which means, that it calculates and stores the joint probabilities by only using (conditional) probabilities of single random variables. Each variable is thought to be independent of all its non-parents with respect to the graph structure of the BN. Graph structure and probabilities of a BN for TFBS are learned from a set of aligned TFBS. The learning process comprises the selection of a feature subset, the calculation of the feature values from the training samples, a BN structure learning algorithm[5] and maximum likelihood estimation of the probability distributions for all features given the structure. For the first task, the selection of a predictive feature subset, we have applied sequential search algorithms, which are common in the field of feature subset selection (FSS).[17]

3.1. *Sequential search algorithms*

Traditionally, a feature selection problem is stated as follows: select d features from a set $\mathcal{X}$ of D (with $d < D$) measured features with the performance of the recognition

system as high as possible.[17] Given an objective function $J(\mathcal{Y})$ for evaluating the quality of feature subsets $\mathcal{Y} \subseteq \mathcal{X}$, this task is reduced to the search problem of detecting an optimal feature subset $\mathcal{Y}^*$ based on the objective function:

$$\mathcal{Y}^* = \underset{\mathcal{Y}}{\operatorname{argmax}}\, J(\mathcal{Y}). \tag{1}$$

Since one can expect strong dependencies and redundancies among all candidare features, which make it difficult to judge single features by some filter scoring metric, we have decided to apply a wrapper method. Thus, in our application the quality value $J(\mathcal{Y})$ of feature subset $\mathcal{Y}$ is calculated through a cross validation of a BN model containing the features of $\mathcal{Y}$. Learning in each case a positive (TFBS) model and a negative (background) model on 90% of the data and then counting the false positive rate (FP) for a fixed sensitivity of 90% on the remaining data, we finally use these statistics to calculate the well-known $F_{0.5}$-measure:

$$J(\mathcal{Y}) = F_{0.5} = \frac{2 \cdot r \cdot p}{r + p} \tag{2}$$

with the *recall* $r = \frac{\mathrm{TP}}{\mathrm{TP+FN}}$ being the part of of real TFBS that were considered as matches (which is fixed to 0.9) and the *precision* $p = \frac{\mathrm{TP}}{\mathrm{TP+FP}}$ being the fraction of true positives (TP) among all matches (TP+FP). Notice that due to the cross validation, one run of the objective function comprises the learning and application of 10 BN models.

Many search algorithms when applied to feature subset selection (e.g. Branch and Bound) require the objective function $J(\mathcal{Y})$ to be monotonically increasing in the sense that it holds:[29]

$$\mathcal{Y}_t \subseteq \mathcal{Y}_s \Longrightarrow J(\mathcal{Y}_t) \leq J(\mathcal{Y}_s). \tag{3}$$

It is important to mention that our objective function is not monotonically increasing which is partly a result of modelling dependencies of (potentially) redundant features in our BN models. Thus we discuss search strategies which do not have this requirement. Examples can be found in the class of sequential search algorithms.

3.2. *Sequential forward selection*

The simplest sequential algorithms are the *sequential forward selection* (SFS).[28] SFS starts with an initially empty feature subset and successively adds that single feature to the current which best improves the quality of the extended subset compared to the quality of the current subset. A schematically illustration of its way to traverse the search space is shown in Figure 3a.) The performance of SFS suffers from nesting problems, especially in our setting (BNs modelling dependencies among features and lots of redundant features) the search process will often follow dead end paths into local optima.[34] This is also true for the generalised version (GSFS) which successively adds small subsets of a given size g. In order to overcome the nesting problems, several algorithms were proposed which allow the

deletion of previously added features. The Plus l – Take Away r (PTA, see Ref. 23) alternates between adding l single features to the current set and deleting r features from it. The possibility of cancelling previously added suboptimal features helps to lower the risk of nesting problems. However, the fixed parameters l and r are too rigid to omit it completely.

3.3. *Sequential forward floating selection*

The *sequential forward floating selection* which is used in this work dynamically adds and deletes features from the current feature set.[17] To get more concrete, at each search step, exactly one feature is added and afterward as many features are deleted as it improves the quality measure. The time complexity of SFFS is $\mathcal{O}(2^D)$ [11]. The pseudocode of SFFS is given in Figure 2. The running time of SFFS

SFFS
Input: $\mathcal{X}$ feature set
Output: $\mathcal{Y}_k$ selected feature subset
Pseudocode:

(1) $\mathcal{Y}_k = \emptyset$, $k = 0$
(2) **Step 1:** (inclusion)
  (a) **IF** $k = d$ **THEN GOTO** Stop
  (b) $x^+ = \text{argmax}_{x \in \mathcal{X} \setminus \mathcal{Y}_k} J(\mathcal{Y}_k \cup \{x\})$
  (c) $\mathcal{Y}_{k+1} = \mathcal{Y}_k \cup \{x^+\}$, $k = k + 1$
(3) **Step 2** (conditional exclusion)
  (a) $x^- = \text{argmax}_{x \in \mathcal{Y}_k} J(\mathcal{Y}_k \setminus \{x\})$
  (b) **IF** $J(\mathcal{Y}_k \setminus \{x^-\}) > J(\mathcal{Y}_{k-1})$ **THEN**
    i. $\mathcal{Y}_{k-1} = \mathcal{Y}_k \setminus \{x^-\}$
    ii. $k = k - 1$
    iii. **GOTO** Step 2
    ELSE GOTO Step 1
(4) **RETURN** $\text{argmax}_{\mathcal{Y}_i : i \in \{1,\ldots,d\}} J(\mathcal{Y}_i)$

Fig. 2. Pseudocode of the SFFS algorithm.

(and any sequential algorithm) is largely determined by the number of performed evaluations $J(\mathcal{Y})$ which itself is a quite time-consuming procedure. To improve the performance of such algorithms we have developed a method for finding good estimations $\hat{J}(\mathcal{Y})$ of real objective function values. This approach is introduced in the following section.

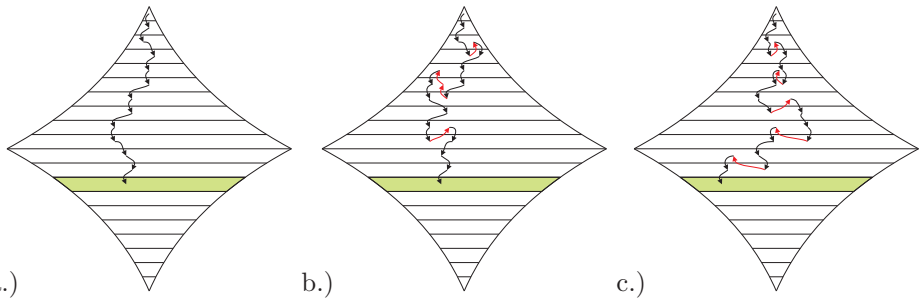

Fig. 3. Illustration of the diverse strategies for traversing the search space. The ith layer of each diamond represents the set of all feature subsets with i features. The upper peak (first layer) represents the single empty subset, whereas the lower peak (last layer) the single subset $\mathcal{Y} = \mathcal{X}$. a.) SFS: can only add features to the current subset, b.) SFFS only deletes features in cases when this improves the current feature subset c.) PTA (Plus $l = 3$, Take Away $r = 1$)

3.4. *Lazy-SFFS*

Lazy-SFFS is a variant of SFFS in which the majority of huge number the $J(\cdot)$-calls in the inclusion step are substituted by our quality estimation function $\hat{J}(\,\cdot\,)$. Before the first round of the SFFS search, the estimator is initialised with the real objective function values for single features $x \in \mathcal{X}$. After that, the selection of a new feature to add is divided into three substeps:

(1) for each candidate feature $x \notin \mathcal{Y}$ the estimated quality $\hat{J}(\mathcal{Y} \cup \{x\})$ is calculated.
(2) Only a small fraction (20% in our study) of candidates scoring best w.r.t. the estimate $\hat{J}(\cdot)$ will be evaluated using the exact objective function $J(\cdot)$.
(3) The $J(\cdot)$ values calculated in step (2) are now used to update the estimator $\hat{J}(\cdot)$.

Figure 4 shows the pseudocode of the inclusion step. The exclusion step is not changed since it usually requires few target function calls compared to the inclusion step. Next, we describe the mathematical foundation of the estimator $\hat{J}(\cdot)$.

The described way of substituting real quality function values by quality estimations works analogously for other sequential feature selection algorithms. At any stage of a sequential search, candidate features are first evaluated using the estimation. For a given percentage (20% in our example) of candidates with best estimation values the real quality value is calculated after that. The search proceeds with the best candidate with respect to the real quality value.

4. Predicting Values of the Objective Function $J(\cdot)$

The challenge is to build a good approximation $\hat{J} : \{0,1\}^{\mathcal{X}} \to [0,1]$ of the true (exact) quality function $J(\cdot)$. The parametrised function $\hat{J}(\cdot)$ will be estimated based on a training sequence $\mathcal{X}_1, \ldots, \mathcal{X}_T \subseteq \mathcal{X}$ of feature subsets, along with their true target values $z_t = J(\mathcal{X}_t)$, $1 \leq t \leq T$. The respective adaptation pairs are

Lazy-SFFS

Input: $\mathcal{X}$ feature set
Output: $\mathcal{Y}_k$ selected feature subset
Pseudocode:

(1) $\mathcal{Y}_k = \emptyset$, $k = 0$

(2) **Step 1:** (inclusion)

(a) **IF** $k = d$ **THEN GOTO** Stop
(b) $\forall x \in \mathcal{X} \setminus \mathcal{Y}_k : \hat{j}_x := \hat{J}(\mathcal{Y}_k \cup \{x\})$
(c) $C := \{x \in \mathcal{X} \setminus \mathcal{Y}_k : \hat{j}_x \text{ is among the } t\% \text{ best}\}$
(d) $\forall x \in C : j_x := J(\mathcal{Y}_k \cup \{x\})$
(e) $\forall x \in C : update_{\hat{J}}(j_x)$
(f) $x^+ = \text{argmax}_{x \in C}\, j_x$
(g) $\mathcal{Y}_{k+1} = \mathcal{Y}_k \cup \{x^+\}$, $k = k + 1$

(3) **Step 2** (conditional exclusion)

$\vdots$

(4) **RETURN** $\text{argmax}_{\mathcal{Y}_i : i \in \{1,\ldots,d\}}\, J(\mathcal{Y}_i)$

Fig. 4. Pseudocode of the part of Lazy-SFFS algorithm which differs from the original SFFS. t is the percentage of estimations for which the real objective function values are calculated. $update(\cdot)$ means that the estimator is updated for a new real objective function value.

resulting from all the exact evaluations of the objective function $J(\cdot)$ during the search process so far.

Prediction of a value $J(\mathcal{Y})$ should access information about which features f_i, which feature pairs $\{f_i, f_j\}$, and which triples $\{f_i, f_j, f_k\}$ and so on are included in $\mathcal{Y}$, respectively. Thus, we assume a certain system $\mathfrak{B} \subseteq \{0,1\}^{\mathcal{X}}$ of feature tuples, or subsets, relevant for prediction. We model the impact of the properties $\mathcal{S} \subseteq \mathcal{Y}$ on target value $J(\mathcal{Y})$ by a linear combination

$$\hat{J}(\mathcal{Y}) \;=\; \hat{J}(\mathcal{Y}|\boldsymbol{w}) \;=\; \sum_{\mathcal{S} \in \mathfrak{B}} w_{\mathcal{S}} \cdot \delta_{\mathcal{S} \subseteq \mathcal{Y}}, \tag{4}$$

where $\delta_{\mathcal{S} \subseteq \mathcal{Y}}$ is the characteristic function.

A particularly simple example of an expansion base $\mathfrak{B}$ is the set of all feature singletons $\{f_i\}$, $i = 1, \ldots, D$. The resulting predictor assumes all individual features of a subset $\mathcal{Y}$ contributing linearly and independently to the overall success rate $J(\mathcal{Y})$. When adding pairs, triples, or even larger feature subsets to the ansatz, pairwise dependences and higher order interactions will be accounted for in the model, too. In practice, combinatorially complete bases $\mathfrak{B}_{=k} = \{\mathcal{S} \subseteq \mathcal{X} \mid \text{card}\,(\mathcal{S}) = k\}$ of moderate order ($k = 3, 4, 5$) and their cumulative counterparts $\mathfrak{B}_{\leq k} = \{\mathcal{S} \subseteq \mathcal{X} \mid \text{card}\,(\mathcal{S}) \leq k\}$ are employed. A similar representation, termed ANOVA decomposition kernels, has been introduced before for a continuous-variable support vector regression task. [21]

The predictor model can obviously be written in inner product form

$$\hat{J}(\mathcal{Y}|\boldsymbol{w}) \;=\; \boldsymbol{w}^\top \boldsymbol{\phi}(\mathcal{Y}) \;, \quad \phi_{\mathcal{S}}(\mathcal{Y}) = \begin{cases} 1 & \mathcal{S} \subseteq \mathcal{Y} \\ 0 & \mathcal{S} \not\subseteq \mathcal{Y} \end{cases} , \tag{5}$$

where $\boldsymbol{\phi}(\mathcal{Y})$ denotes the array of dimension $b = \mathrm{card}\,(\mathfrak{B})$ with components $\phi_{\mathcal{S}}(\mathcal{Y})$, $\mathcal{S} \in \mathfrak{B}$.

The mean squared error (MSE) estimate $\hat{\boldsymbol{w}}$, minimising the average distortion

$$\varepsilon(\boldsymbol{w}) \;=\; \frac{1}{T} \cdot \sum_{t=1}^{T} \left(z_t - \boldsymbol{w}^\top \boldsymbol{\phi}(\mathcal{X}_t) \right)^2 \tag{6}$$

of the predictor w.r.t. the data is easily obtained as the solution

$$\hat{\boldsymbol{w}} \;=\; \left(\boldsymbol{X}^\top \boldsymbol{X} \right)^{-1} \boldsymbol{X}^\top \boldsymbol{z} \tag{7}$$

of the Gaussian normal equations $\boldsymbol{X}^\top \boldsymbol{X} \boldsymbol{w} = \boldsymbol{X}^\top \boldsymbol{z}$. Target vector $\boldsymbol{z} \in \mathbb{R}^T$ and data matrix $\boldsymbol{X} \in \mathbb{R}^{T \times b}$ are understood to attain the value z_t or the binary vector $\boldsymbol{\phi}(\mathcal{X}_t)$ in its t^{th} row, respectively. See Ref. 9 for details about MSE estimates, in particular, about regularisation in case the moment matrix $\boldsymbol{X}^\top \boldsymbol{X}$ has less than full rank.

4.1. *Dualisation of the problem*

Since for typical expansions $\hat{J}(\cdot)$ the cardinality of the system $\mathfrak{B}$ is extraordinarily large — the maximum cardinality $b = 2^D$ occurs if the power set $\{0,1\}^{\mathcal{X}}$ itself is chosen for $\mathfrak{B}$ in (4) — the computations in (5), (7) are far beyond practical feasibility. As a consequence, the MSE minimisation problem has to be reformulated in dual space, a procedure that has been coined 'kernel trick' in the SVM literature.[2]

First we observe, using elementary transformations along with the singular value decomposition $\boldsymbol{V}\boldsymbol{D}\boldsymbol{U}^\top$ of training data matrix $\boldsymbol{X}$, that any weight vector with minimum MSE can be represented as a linear combination

$$\boldsymbol{w} \;=\; \boldsymbol{X}^\top \boldsymbol{\beta} \;=\; \sum_{t=1}^{T} \beta_t \cdot \boldsymbol{\phi}(\mathcal{X}_t) \;, \qquad \boldsymbol{\beta} \in \mathbb{R}^T \tag{8}$$

in a T-dimensional subspace of $\mathbb{R}^b$ spanned by the training data. After substitution, the MSE solution becomes

$$\hat{\boldsymbol{w}} \;=\; \boldsymbol{X}^\top \hat{\boldsymbol{\beta}} \;, \qquad \hat{\boldsymbol{\beta}} \;=\; \boldsymbol{G}^{-1} \boldsymbol{z} \;=\; \left(\boldsymbol{X}\boldsymbol{X}^\top \right)^{-1} \boldsymbol{z} \tag{9}$$

and the prediction translates into

$$\hat{J}(\mathcal{Y}|\boldsymbol{\beta}) \;=\; \boldsymbol{\beta}^\top \boldsymbol{X} \boldsymbol{\phi}(\mathcal{Y}) \;=\; \sum_{t=1}^{T} \beta_t \cdot (\boldsymbol{\phi}(\mathcal{X}_t))^\top \boldsymbol{\phi}(\mathcal{Y}) \;. \tag{10}$$

The $(T \times T)$-matrix $\boldsymbol{G}$ is called the Gram matrix of the data, and its entries

$$G_{st} \;=\; (\boldsymbol{\phi}(\mathcal{X}_s))^\top \boldsymbol{\phi}(\mathcal{X}_t) \tag{11}$$

are inner products in feature subset indicator space.

4.2. *Computation of the kernel function*

The dual equations for minimisation (9) and prediction (10) are of polynomial complexity ($\mathrm{O}(T^3)$ or $\mathrm{O}(T^1)$, resp.) w.r.t. the adaptation set size T. Then again, note that the number of samples which form the empirical basis of the actual feature selection task does obviously not exert any influence on the predictor's computational cost.

However, to obtain an efficient computation, the underlying kernel operator

$$\mathsf{K}(\mathcal{A},\mathcal{B}) \stackrel{\mathrm{def}}{=} (\boldsymbol{\phi}(\mathcal{A}))^{\top}\boldsymbol{\phi}(\mathcal{B}) \;=\; \sum_{\mathcal{S}\in\mathfrak{B}} \underbrace{\delta_{\mathcal{S}\subseteq\mathcal{A}}\cdot\delta_{\mathcal{S}\subseteq\mathcal{B}}}_{\delta_{\mathcal{S}\subseteq\mathcal{A}\cap\mathcal{B}}} \tag{12}$$

must be evaluable with reasonable effort. Counting the number of base sets $\mathcal{S}\in\mathfrak{B}$ which are a subset of $\mathcal{A}\cap\mathcal{B}$ is an easy task as long as regularly structured expansion bases are considered like the combinatorial k-order systems $\mathfrak{B}_{=k}$ or $\mathfrak{B}_{\leq k}$, resp. Solely depending on the size

$$\nu \;=\; \mathrm{card}\,(\mathcal{A}\cap\mathcal{B})$$

of the intersection, the requested inner product is obtained by the following formulae which are easily verified using elementary combinatorics:

$$\mathsf{K}(A,B) \;=\; \begin{cases} \nu & \mathfrak{B}=\mathfrak{B}_{=1} \\ \binom{\nu}{k} \text{ if } \nu\geq k \text{ else } 0 & \mathfrak{B}=\mathfrak{B}_{=k} \\ \sum_{j=0}^{\min(\nu,k)}\binom{\nu}{j} & \mathfrak{B}=\mathfrak{B}_{\leq k} \\ 2^{\nu} & \mathfrak{B}=\mathfrak{B}_{\leq D} \end{cases} \tag{13}$$

If the counts in Eq. (13) are tabulated, the computation of $\mathsf{K}(A,B)$ involves costs essentially proportional to the smaller of the cardinalities of subsets A and B.

4.3. *Incremental adaptation of predictor $\hat{J}(\cdot)$*

On top level of the FSS search procedure, a dynamically evolving sequence $(\mathcal{X}_t)_{t\in\mathbb{N}}$ of (locally) most promising candidate feature subsets is successively evaluated. Let $\boldsymbol{G}$, $\boldsymbol{G}^{-1}$ denote the Gram matrix at time T and its inverse, respectively; then, at time $(T+1)$, as soon as target score $z_{T+1}=J(\mathcal{X}_{T+1})$ is made available, the predictor has to be provided with the $(T+1)$-dimensional updates

$$\tilde{\boldsymbol{G}} \;=\; \begin{pmatrix} \boldsymbol{G} & \boldsymbol{g} \\ \boldsymbol{g}^{\top} & \gamma \end{pmatrix} \quad\text{and}\quad \tilde{\boldsymbol{G}}^{-1} \;=\; \begin{pmatrix} \boldsymbol{H} & \boldsymbol{h} \\ \boldsymbol{h}^{\top} & \eta \end{pmatrix} \tag{14}$$

where $\gamma=\mathsf{K}(\mathcal{X}_{T+1},\mathcal{X}_{T+1})$ and $\boldsymbol{g}\in\mathbb{R}^T$ has components $g_t=\mathsf{K}(\mathcal{X}_t,\mathcal{X}_{T+1})$. The well-known rules

$$\begin{aligned} \boldsymbol{H}^{-1} &= \boldsymbol{G}-\frac{1}{\gamma}\cdot\boldsymbol{g}\boldsymbol{g}^{\top} \\ \eta^{-1} &= \gamma-\boldsymbol{g}^{\top}\boldsymbol{G}^{-1}\boldsymbol{g} \\ \boldsymbol{h} &= -\eta\cdot\boldsymbol{G}^{-1}\boldsymbol{g} \end{aligned}$$

of block matrix algebra relate the desired components η, $\boldsymbol{h}$, $\boldsymbol{H}$ to the updated table $\tilde{\boldsymbol{G}}$ of inner products. In particular, we can solve for the northwest block in $\mathrm{O}(T^2)$ time by making use of a (rank one) special case

$$\boldsymbol{H} \;=\; \boldsymbol{G}^{-1} + \frac{1}{\gamma - \boldsymbol{g}^\top \boldsymbol{G}^{-1} \boldsymbol{g}} \cdot (\boldsymbol{G}^{-1}\boldsymbol{g})(\boldsymbol{G}^{-1}\boldsymbol{g})^\top \tag{15}$$

of the Sherman-Morrison-Woodbury formula.[7]

Taking advantage of this incremental algorithm and using our efficient kernel formula (13), the costs for a prediction step and the costs for an adaptation step scale linearly and quadratically, resp., with the size T of the predictor's training set. It is worth mentioning, however, that the application we have in mind is to speed up a FSS wrapper procedure, and so calculating values of the black-box function $J(\cdot)$ is usually a very time-consuming process. In other words, the scenario at hand will never allow us to produce exact J-values at a scale that renders training or running the predictor a dominant factor in the total expenditure.

5. Results

We validated our new search algorithm, Lazy-SFFS, on two datasets. The first one contains 108 Sp1 binding sites taken from TRANSFAC[15] (the positive samples) and 4577 randomly chosen sequences (the negative samples). The second dataset contains 77 AP1 boxes (the positives) and again 4577 negatives. AP1 box is the name for a binding site for dimerised transcription factors of the JUN/FOS family, either for homodimers of two JUN factors or for heterodimers of a JUN and a FOS protein. The sequences of AP1 boxes are less conserved than the Sp1 boxes.

Since the sequence context of a TFBS could be extended theoretically to thousands of basepairs and some feature types have continuous parameters (e.g. the threshold of a structural parameter feature), one could produce an infinite set of feature candidates. Since SFFS does not work on infinite dataset, we had to preselect features from the infinite set $\mathcal{X}$ of all possible features. The preselection was done by more-or-less iterating over the free parameters of each feature type. In the end, we kept 802 features for the Sp1 dataset and 495 features for the AP1 dataset.

Table 1. How many evaluations ($\#J(\mathcal{Y})$) had to be calculated to find the feature subset with a given quality (max $J(\mathcal{Y})$)?

a.)

algorithm	$\#J(\mathcal{Y})$	$\lvert Y\rvert$
SFFS	14,535	10
Lazy-SFFS	3,996	16

b.)

algorithm	$\#J(\mathcal{Y})$	$\lvert Y\rvert$
SFFS	9,983	10
Lazy-SFFS	2,345	11

Note: a.) Sp1 data set: the size of the feature subset found by Lazy-SFFS is higher than that found by SFFS. (see text for discussion). However, Lazy-SFFS reduced the number of evaluations more 70% compared to baseline SFFS. b.) AP1 data set: Our *lazy* approach therefore calculates more than 75% less evaluations than the standard SFFS.

We were interested in the number of evaluations $J(\mathcal{Y})$ which are required to find a feature subset of a given quality measured by $J(\mathcal{Y})$ which is the $F_{0.5}$-measure of a cross validation with BNs modelling a certain feature subset $\mathcal{Y}$). To validate our quality estimation method we compared the common SFFS algorithm with our Lazy-SFFS.

The latter algorithm includes a predictor $\hat{J}(\cdot)$ equipped with the kernel function (Eq. 13) according to the expansion base $\mathfrak{B}_{\leq 4}$. The choice of kernel type (cumulative) and kernel order ($k = 4$) was based on the results of preceding Lazy-SFFS runs using non-TFBS data sets. On both datasets the common SFFS needs approximately three times the evaluations of the lazy variant (see Figure 6). The number of evaluations $J(\mathcal{Y})$ which were performed by both search algorithms at the time when they first reached a meaningful quality value (0.93 for Sp1, 0.82 for AP1,) are shown in Tables 5a.) and b.). Furthermore, it shows the sizes of the resulting feature subsets. In the case of Sp1 the feature subset found by SFFS is rather smaller than that returned by Lazy-SFFS. Naturally , when using estimations instead of real objective function values, an interesting point is how the estimates fit to the real values. The correlation plot in Figure 5 shows that our estimations show an impressive predictive power on the whole range of quality values except in the middle range where the prediction seems to underestimate the real values a bit. In spite of high correlations between estimation and real quality values, the intersection of final feature subsets picked by the original SFFS and the Lazy-SFFS is small for both datasets. For the Sp1 dataset, the final feature subsets have three features in common, for the AP1 dataset, there are two features in common. This is not surprising, since the candidate feature set $\mathcal{X}$ were created by iterating through the free-parameter-space of each type of features. Thus, there are redundant features, which lead to very close results but influence the further traversal through the search space.

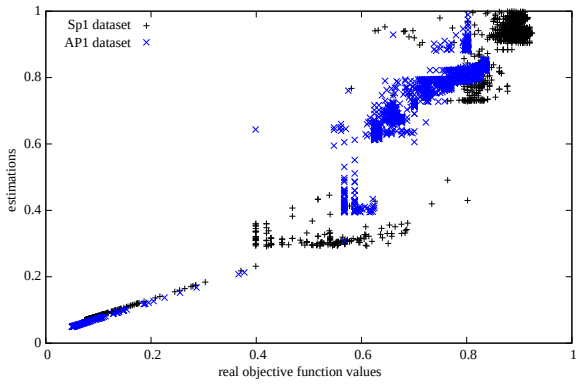

Fig. 5. Dot plot of real objective function values $J(\mathcal{Y})$ for feature subsets vs. the corresponding estimations $\hat{J}(\mathcal{Y})$.

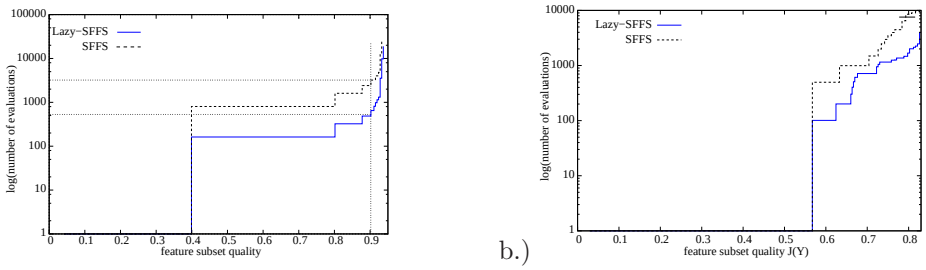

Fig. 6. Number of evaluations in dependence of an approached feature subset quality. The y-axis is in logscale. a.) Sp1 dataset. Two points are marked to illustrate the improvement of Lazy-SFFS compared to SFFS. To reach a model quality of 0.9, SFFS evaluates approximately 3200 feature subsets. Lazy-SFFS only needs 650. Following the lower horizontal dashed line reveals that spending 650 evaluations achieves a quality of 0.9 for Lazy-SFFS and a quality less than 0.4 for the baseline SFFS, respectively. b.) AP1 dataset

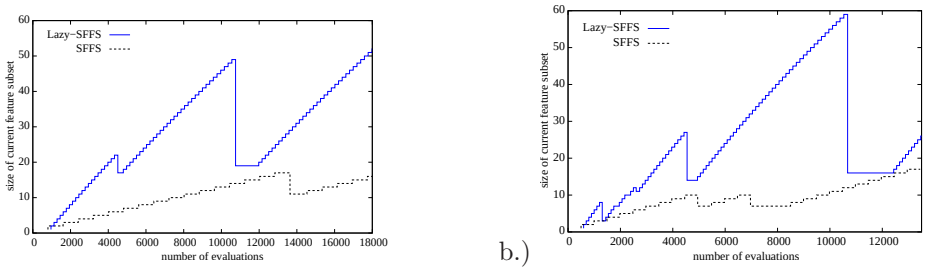

Fig. 7. Size of currently examined feature subsets in relation to the number of quality function computations so far. a.) Sp1 data set, b.) AP1 data set. The Lazy-SFFS reaches deeper regions of the search space for a fixed number of evaluations. After having found the (putative) optimum, the search strategy (SFFS) always adds new features without having the possibility of deleting some of them because deletion would not improve the quality. This behavior is given by the SFFS strategy and can be ommitted by defining a suitable stop criterion in real-world applications.

In order to provide a comparison of concrete running time between both algorithms, we measured the time an algorithm takes for adding the next best feature when applied to the Sp1 dataset. The original SFFS which evaluates each candidate feature by a complete cross validation, needs 344 seconds for this task, averaged over 10 rounds. The Lazy-SFFS calculates only 20% of true quality values, but updating the estimation unit with these true qualities needs 138 seconds. Of this time, only 3.3 seconds are consumed for estimating the qualities of all candidate feature subsets.

A common working hypothesis for developing wrapper methods is that in general wrappers achieve better predictions of feature subset qualities than filters. To test this argument for our datasets, we applied two filter methods and evaluated the resulting feature subsets by cross validation. The first filter method is a ranker which

sorts the features with respect to the normalized transinformation of single features and the class labels [30]. The second filter is the *FCBF* algorithm, a correlation based filter published in [30], which accounts for redundancy among features. Since the original SFFS algorithm achieved the best result for feature subsets of size 10 in both datasets, we chose the 10 best features predicted by both filters and calculated qualities $J(\mathcal{Y})$. In all cases the feature subsets resulting from filter methods range well below what is achievable with wrapper methods with respect to the cross-validated $F_{0.5}$-measure. See Table 2 for results of a ranker and FCBF as well as for the wrappers SFFS and Lazy-SFFS which constitute a lower bound for an optimal feature set the SFFS feature subsets with respect to the cross-validated $F_{0.5}$-measure (see Table 2).

dataset	ranker	FCBF	SFFS	Lazy-SFFS
Sp1	0.8435	0.8584	0.9282	0.9282
AP1	0.3409	0.6736	0.7804	0.08082

Table 2. Comparison of quality values $J(\mathcal{Y})$ of feature subsets of size 10 found by the two filter methods and the first subsets of size 10 evaluated by SFFS and Lazy-SFFS. For both datasets, the wrapper finds a feature subset of higher quality. Note that SFFS and Lazy-SFFS differ significantly in search efficiency rather than $F_{0.5}$ performance. Note further, that the filter methods and the wrapper methods are in the same sense optimistic since the same part of data which is used for feature selection is also used for evaluation.

6. Conclusions

Feature subset selection is an central task for designing recognition systems and is applied in diverse fields of computational biology. In this paper we have faced the problem of selecting predicative properties of transcription factor binding sites for the purpose of predicting unknown sites in promoter sequences. We have developed a quality estimation technique which reduces the number of feature subset evaluations by more than 70% compared to the baseline SFFS algorithm. For example, for the Sp1 dataset, spending 650 evaluations achieves a quality of 0.9 for Lazy-SFFS and a quality less than 0.4 for the baseline SFFS, respectively (see 6). The estimator is refined incrementally while searching, using a kernel function for minimising the mean standard error. This is a common technique in the field of support vector machines. The reduction of the time-consuming cross validations of each feature subset allows us to search deeper regions of the search space in a suitable time. Naturally, the approach can be applied very easily to arbitrary feature selection problems. Especially microarray and mass spectrography data are of special interest.

Acknowledgments

This work was supported by the German Ministry of Education and Research under grant number $0312704K$.

References

1. N. A. Chuzhanova, A. J. Jones, and S. Margetts. Feature selection for genetic sequence classification. *Bioinformatics*, 14(2):139–43, 1998.
2. N. Cristianini and J. Shawe-Taylor. *Support Vector Machines and other Kernel-Based Learning Methods*. Cambridge University Press, 2000.
3. Sven Degroeve, Bernard De Baets, Yves Van de Peer, and Pierre Rouze. Feature subset selection for splice site prediction. *Bioinformatics*, 18 Suppl 2:S75–83, 2002.
4. M. A. El Hassan and C. R. Calladine. Conformational characteristics of dna: Empirical classifications and a hypothesis for the conformational behaviour of dinucleotide steps. *Phil. Trans. R. Soc. Lond. A*, 355:43–100, 1997.
5. Nir Friedman, Dan Geiger, and Moises Goldszmidt. Bayesian network classifiers. *Machine Learning*, 29(2-3):131–163, 1997.
6. Cesare Furlanello, Maria Serafini, Stefano Merler, and Giuseppe Jurman. Entropy-based gene ranking without selection bias for the predictive classification of microarray data. *BMC Bioinformatics*, 4:54, 2003.
7. G.H. Golub and C.F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, 1996.
8. Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003.
9. T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer, 2001.
10. Lawrence B. Holder, Ingrid Russell, Zdravko Markov, Anthony G. Pipe, and Brian Carse. Current and future trends in feature selection and extraction for classification problems. *IJPRAI*, 19(2):133–142, 2005.
11. M. Kudo and J. Sklansky. Comparison of algorithms that select features for pattern classifiers. *Pattern Recognition*, 33(1):25–41, 2000.
12. Yi Li, Colin Campbell, and Michael Tipping. Bayesian automatic relevance determination algorithms for classifying gene expression data. *Bioinformatics*, 18(10):1332–9, 2002.
13. Ying Liu. A comparative study on feature selection methods for drug discovery. *J Chem Inf Comput Sci*, 44(5):1823–8, 2004.
14. Yong Mao, Xiao-Bo Zhou, Dao-Ying Pi, You-Xian Sun, and Stephen T. C. Wong. Parameters selection in gene selection using Gaussian kernel support vector machines by genetic algorithm. *J Zhejiang Univ Sci B*, 6(10):961–73, 2005.
15. V. Matys, E. Fricke, R. Geffers, E. Gossling, M. Haubrock, R. Hehl, K. Hornischer, D. Karas, A. E. Kel, O. V. Kel-Margoulis, D-U Kloos, S. Land, B. Lewicki-Potapov, H. Michael, R. Munch, I. Reuter, S. Rotert, H. Saxel, M. Scheer, S. Thiele, and E. Wingender. TRANSFAC: transcriptional regulation, from patterns to profiles. *Nucleic Acids Research*, 31(1):374–8, 2003.
16. G. J. McLachlan, R. W. Bean, and D. Peel. A mixture model-based approach to the clustering of microarray expression data. *Bioinformatics*, 18(3):413–22, 2002.
17. P. Pudil, J. Novovicova, and J. Kittler. Floating search methods in feature-selection. *Pattern Recognition Letters*, 15(11):1119–1125, 1994.
18. Rainer Pudimat, E.G. Schukat-Talamazzini, and Rolf Backofen. A multiple-feature

framework for modelling and predicting transcription factor binding sites. *Bioinformatics*, 21(14):3082–8, 2005.
19. Lior Rokach, Barak Chizi, and Oded Maimon. A methodology for improving the performance of non-ranker feature selection filters. *IJPRAI*, 21(5):809–830, 2007.
20. Y. Saeys, S. Degroeve, D. Aeyels, Y. Van De Peer, and P. Rouze. Fast feature selection using a simple estimation of distribution algorithm: a case study on splice site prediction. *Bioinformatics*, 19 Suppl 2:II179–II188, 2003.
21. G. Saunders, A. Gammerman, and V. Vovk. Ridge regression learning algorithm in dual variables. In *Proc. 15th International Conf. on Machine Learning*, pages 515–521. Morgan Kaufmann, San Francisco, CA, 1998.
22. Chao Sima, Ulisses Braga-Neto, and Edward R. Dougherty. Superior feature-set ranking for small samples using bolstered error estimation. *Bioinformatics*, 21(7):1046–54, 2005.
23. S. D. Stearns. On selecting features for pattern classifiers. In *Third International Conference on Pattern Recognition*, pages 71–75, 1976.
24. G. D. Stormo. Dna binding sites: representation and discovery. *Bioinformatics*, 16(1):16–23, 2000.
25. Minghe Sun and Momiao Xiong. A mathematical programming approach for gene selection and tissue classification. *Bioinformatics*, 19(10):1243–51, 2003.
26. Gulden Uchyigit and Keith Clark. A new feature selection method for text classification. *IJPRAI*, 21(2):423–438, 2007.
27. Sudhir Varma and Richard Simon. Iterative class discovery and feature selection using Minimal Spanning Trees. *BMC Bioinformatics*, 5:126, 2004.
28. A. W. Whitney. A direct method of nonparametric measurement selection. *IEEE Transactions on Computers*, 20(6):1100–1103, 1971.
29. B. Yu and B. Yuan. A more efficient branch and bound algorithm for feature selection. *Pattern Recognition*, 26(6):883–889, 1993.
30. Lei Yu and Huan Liu. Feature selection for high-dimensional data: A fast correlation-based filter solution. In *ICML*, pages 856–863, 2003.
31. Fanfan Zeng, Roland H. C. Yap, and Limsoon Wong. Using feature generation and feature selection for accurate prediction of translation initiation sites. *Genome Inform Ser Workshop Genome Inform*, 13:192–200, 2002.
32. Lirong Zhang and Liaofu Luo. Splice site prediction with quadratic discriminant analysis using diversity measure. *Nucleic Acids Research*, 31(21):6214–20, 2003.
33. Xing-Ming Zhao, Ji-Xiang Du, Hong-Qiang Wang, Yunping Zhu, and Yixue Li. A new technique for selecting features from protein sequences. *IJPRAI*, 20(2):271–283, 2006.
34. D. Zongker and A. Jain. Algorithms for feature selection: An evaluation. In *Proceedings of the 13th International Conference on Pattern Recognition*, pages 18–22, Vienna, Austria, 1996.