Faculty of Sciences

Feature Selection for Classification of Nucleic Acid Sequences

Yvan Saeys

Promotor:
Prof. Dr. Yves Van de Peer

Co-promotor:
Prof. Dr. ir. Dirk Aeyels

September 2004

Dissertation submitted in fulfillment of the requirements for the degree of Doctor (Ph.D.) in Sciences, Computer science
The content of this work is protected by the laws of copyright. The author gives authorisation to consult and to copy part of this text for personal use. Any other use is subjected to copyright regulations, and in particular the obligation to explicitly mention the original source of this text when citing results described in this work.

The author
# Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preface</td>
<td>9</td>
</tr>
<tr>
<td>1 Introduction</td>
<td>11</td>
</tr>
<tr>
<td>2 Supervised learning</td>
<td>15</td>
</tr>
<tr>
<td>2.1 Introduction</td>
<td>15</td>
</tr>
<tr>
<td>2.2 Classification techniques</td>
<td>16</td>
</tr>
<tr>
<td>2.2.1 Bayesian classifiers</td>
<td>18</td>
</tr>
<tr>
<td>2.2.2 Support Vector Machines</td>
<td>20</td>
</tr>
<tr>
<td>2.2.3 Decision trees</td>
<td>24</td>
</tr>
<tr>
<td>2.3 Assessing classification performance</td>
<td>28</td>
</tr>
<tr>
<td>2.4 Summary</td>
<td>32</td>
</tr>
<tr>
<td>3 Feature selection techniques</td>
<td>33</td>
</tr>
<tr>
<td>3.1 Introduction</td>
<td>33</td>
</tr>
<tr>
<td>3.2 Filter approaches</td>
<td>36</td>
</tr>
<tr>
<td>3.3 Wrapper approaches</td>
<td>41</td>
</tr>
<tr>
<td>3.3.1 A new wrapper method: the Constrained Estimation of Distribution Algorithm (CDA)</td>
<td>43</td>
</tr>
<tr>
<td>3.4 Embedded techniques</td>
<td>47</td>
</tr>
<tr>
<td>3.4.1 Weighted Naive Bayes Method</td>
<td>47</td>
</tr>
<tr>
<td>3.4.2 Weighted Linear Support Vector Machine</td>
<td>49</td>
</tr>
<tr>
<td>3.5 Techniques based on signal processing</td>
<td>50</td>
</tr>
<tr>
<td>3.5.1 Fourier transform</td>
<td>50</td>
</tr>
<tr>
<td>3.6 Feature weighting</td>
<td>51</td>
</tr>
<tr>
<td>3.6.1 A new feature weighting approach: EDA-based feature weighting</td>
<td>52</td>
</tr>
<tr>
<td>3.7 Feature selection toolbox</td>
<td>57</td>
</tr>
<tr>
<td>3.7.1 Classifiers</td>
<td>60</td>
</tr>
<tr>
<td>3.7.2 Filters</td>
<td>60</td>
</tr>
<tr>
<td>3.7.3 Wrappers</td>
<td>61</td>
</tr>
<tr>
<td>3.7.4 Embedded methods</td>
<td>61</td>
</tr>
<tr>
<td>3.7.5 Feature weighting methods</td>
<td>61</td>
</tr>
<tr>
<td>3.7.6 Discretization</td>
<td>62</td>
</tr>
</tbody>
</table>
## CONTENTS

3.7.7 Evaluation .................................................. 62
3.7.8 Implementation ............................................ 62
3.8 Summary .......................................................... 63

4 Computational gene prediction ........................................ 65
  4.1 Introduction .................................................. 65
  4.2 Gene Prediction ................................................ 68
    4.2.1 Intrinsic gene prediction ............................... 69
    4.2.2 Complete gene structure prediction ..................... 74
    4.2.3 Feature selection ....................................... 75
  4.3 The Arabidopsis dataset: some statistics ....................... 75
  4.4 Summary ..................................................... 78

5 Feature selection for coding potential prediction .................... 81
  5.1 Introduction .................................................. 81
  5.2 Fourier analysis of coding sequences .......................... 82
  5.3 Combining Fourier features and feature selection ............... 86
  5.4 Extending the feature set: including all frequencies ............ 89
  5.5 A comparison with an Interpolated Markov Model ............... 93
  5.6 Summary ..................................................... 95

6 Feature selection for splice site prediction ........................ 97
  6.1 Introduction .................................................. 97
  6.2 Acceptor prediction: an iterative feature selection approach .. 101
    6.2.1 A simple dataset: position dependent nucleotides ........ 102
    6.2.2 Adding position invariant features ........................ 104
    6.2.3 Adding more complex features: dependencies between
           adjacent nucleotides .................................. 106
    6.2.4 A customized feature set for acceptor prediction .......... 110
    6.2.5 A note on interpreting feature selection results ........... 115
  6.3 Donor prediction: an iterative feature selection approach ....... 117
    6.3.1 A simple dataset: position dependent nucleotides ........ 117
    6.3.2 Adding position invariant features ........................ 118
    6.3.3 Adding position dependent dinucleotides ................... 121
    6.3.4 A customized feature set for donor prediction .............. 121
  6.4 Summary ..................................................... 125

7 Feature selection for TIS prediction ................................ 127
  7.1 Introduction .................................................. 127
  7.2 TIS prediction: an iterative feature selection approach ........ 128
    7.2.1 A simple start: positions dependent features ............. 129
    7.2.2 Adding position invariant features ........................ 132
    7.2.3 An educated guess: including position dependent 3-mers .... 134
    7.2.4 Putting the parts together: a customized feature set for
           TIS prediction ........................................ 135
  7.3 Summary ..................................................... 139
## CONTENTS

8 Conclusions, reflections and future prospects 141
  8.1 Conclusions .......................... 141
  8.2 Reflections and future prospects ............... 143

Nederlandstalige samenvatting 147

Glossary 157

Bibliography 160
When people would have told me four years ago that I would end up doing a PhD at a genetics lab, I most probably would have smiled at them gently, thinking to myself that this would probably not happen in a month of Sundays. However, things can change rapidly - even in science - and after spending six months on developing new genetic algorithms to train digital neural networks, I ended up in a place where genetic algorithms - the real ones - were all around: the department of molecular genetics. Around that time, the bioinformatics group of the department was still a cozy mixture of graduate and PhD students, stuffed together in a small room with about 10 computers, no air-conditioning. Four years later, the number of group members has almost tripled, and the whole department has moved to a new building, bioinformatics now being one of the major research topics in the department and occupying a large room with over thirty spacious desks.

During these years, I had the pleasure to learn and taste from different research disciplines, both in the area of computer science and in biology. I guess the true interdisciplinarity of research in bioinformatics is what attracts me the most in this field, both broadening my scientific background and providing me with enough material to dig into new computational approaches to biological problems. The main thread running through all my research escapades is the combination of mathematical and computational techniques to solve various biological problems. The lion share of my work is bundled into this PhD and concerns new computational approaches for gene prediction, although I have to admit that I also enjoyed working on a few side projects such as a combinatorics of mass spectrometry data, detecting large scale gene duplications and analysing promoter data.

Working as a computer scientist in a biological department opened for me a whole new world of intriguingly complex yet beautiful machinery that nature developed during the course of evolution. Looking at these processes constantly reveals new marvels of how life is organised at the molecular level.

Having arrived at the end of this four-year-lasting and rather intensive journey, I would like to express my gratitude to a number of people who contributed
- each in their own way - to the accomplishment of this work. The start of
my PhD was encouraged by Dirk Frimout, who put me in contact with Marc
Zabeau, at that time head of the department of plant genetics. Both Marc
and his successor, Dirk Inze, acknowledged the importance of expanding their
teams with computer scientists, and I would like to thank both of them, as
well as the Flemish Interuniversity Institute for Biotechnology (VIB) for the
financial support that facilitated my research.

From a scientific point of view, I had the pleasure of being endowed with two
promotors: Yves Van de Peer (my “real” boss) and Dirk Aeyels. I would like
to thank both of them for their respective biological and mathematical super-
vision of my work, and broadening of my vision on several aspects of scientific
research. I also owe a special merci to Pierre Rouze, founding father of the
bioinformatics group, for his advice on several aspects of genome annotation
and for help in biological interpretations of my results.

In everyday life, I appreciate the conversations, excursions and other activi-
ties with all my colleagues of the bioinformatics group. I truly think it is an
exceptional group of people, and I hardly know of any other research group
where the atmosphere is in such a brotherly spirit. A special thanks goes to
my colleague-computer scientist Sven, “partner in crime” for both a number of
research and non-research related adventures, with whom I had a lot of inter-
esting discussions on several aspects of machine learning.

Furthermore I want to thank some other people I had the pleasure to work
with: the colleagues from S9 for providing me with some additional computa-
tional power to run my simulations, and especially Tom Willems who devoted
a lot of his time to the cluster; the people of the Intelligent Systems group
at the university of the Basque Country (special thanks to Iñaki Inza, Endika
Bengoetxea and Pedro Larrañaga); and Bernard De Baets, Hans De Meyer and
Marie-France Sagot for their willingness to be part of my jury.

Saving the best for last, I would like to express my deepest gratitude to a
small group of people, who supported me all the way to the point where I am
now. I would like to thank my parents for all means they gave me to suc-
cessfully accomplish my studies, and their continuous support and faith in my
activities. I am grateful to my best friends; for always being there and provid-
ing me with enough distracting activities, especially these last months. And at
last Catherine, that very special person, always by my side with her love and
care, bedankt schattebol ! We duddits !
Chapter 1

Introduction

During the past decade, technological advances have drastically revolutionized the life sciences, a milestone being the first draft sequence of the human genome, first published in 2001. The massive amounts of genomic (sequence) data that since then have been - and are being - generated resulted in a new field of research, bringing together biology, mathematics and computer science: bioinformatics was born.

A great deal of bioinformatics problems thus relates to analysing biological sequences, where we can distinguish between nucleic acid sequences (forms of DNA and RNA) and amino acid sequences (protein sequences). The latter category can be derived from the first when the exact location of the protein, or the gene that encodes this protein, is known. Therefore, we limit the scope of this work to the analysis of nucleic acid sequences.

When confronted with new sequences, or even with a whole new genome, a first thing to do is to find the genes on this sequence, a task that is termed genome annotation. In this process, the different genes that are encoded in the DNA are documented, and their exact structure is defined. The quality of the annotation defines how well the documented genes resemble the true structure of the genes, and is of major importance for a number of subsequent analyses.

In the early days, gene annotation was only done manually by human experts. However, the increasing pace of newly sequenced genomes that become available raised the need for an automation of this process, which stimulated a new field of research in bioinformatics: gene prediction. In gene prediction, techniques from the field of pattern recognition are used to build an expert system that is able to recognize genes in sequences. In an ideal scenario, such an expert system would be fully equivalent to the biological processes that are involved in recognizing and processing genes, mimicking the biological machinery. In practice however, many of these biological processes are far from being completely understood, and the mathematical models that are used in the expert system are only simplified approximations of the true biological mechanisms.
As a result, improving the quality of automatic genome annotations forms a continuous topic of research in computational gene prediction.

To improve the quality of gene prediction, better submodels are needed, modelling the underlying biological processes in a more realistic way. In order to include as much information as possible, the submodels of the expert system are usually provided with a lot of parameters describing different aspects of the sequence, hoping that the characteristics that are truly important are included in the model. In the context of classification, those descriptive parameters are termed features, and the selection of a good set of features is of key importance in the design of good classification models that will be used afterwards by the expert system.

In this work, we perform a thorough analysis of the techniques that can be used to select good feature sets for the classification of nucleic acid sequences in the framework of gene prediction. We tested both existing methods and new techniques that were developed during our research. For a chosen set of subtasks involved in gene prediction, we perform a detailed analysis, focusing on the specific features that could be beneficial to each subtask. For each task, we investigate the use of feature selection with regard to classification performance and extraction of new knowledge.

The experiments we describe all make use of the plant model species *Arabidopsis thaliana*, yet the techniques we develop are general methods that can be applied to any organism, and even to a broad range of classification problems such as text classification or image processing.

This work is organised as follows. Chapter 2 provides an introduction to supervised learning, acquainting the reader with the essentials of pattern recognition. We introduce the classification techniques that will be used in our experiments, and explain some basic methodologies of building and evaluating classification models.

Chapter 3 further builds on these techniques and discusses the different aspects of feature selection for classification. We discuss the motivations to do feature selection, and describe various approaches to accomplish this. Keeping in mind the applications to gene prediction, we also develop new algorithms for the weighting, ranking and selection of features. In a last part of this chapter we focus on some practical issues when dealing with large datasets described by many features, presenting our new feature selection toolkit.

Chapter 4 opens the door to the biological part of the thesis. The chapter familiarizes the reader with the biological processes of transcription and translation, essential to understand the gene prediction problem. Afterwards, we describe the different subtasks that are involved in gene prediction, and discuss the most common algorithms that are used to tackle these problems.

Chapters 5 through 7 describe three practical applications of feature selection for the most important classification tasks involved in gene prediction. Chapter 5 addresses the problem of coding potential prediction, chapter 6 focuses
on splice site prediction, and chapter 7 deals with the prediction of translation start sites. For each of these classification tasks we perform a comparative evaluation of feature selection techniques, assess classification performance, and investigate whether we can extract new domain knowledge. Finally, chapter 8 concludes this work by summarizing our main achievements and suggesting new research directions that emerged from this work.
Chapter 2

Supervised learning

Learning is not so much an additive process, with new learning simply piling up on top of existing knowledge, as it is an active, dynamic process in which the connections are constantly changing and the structure reformatted

- K. Patricia Cross

2.1 Introduction

During the past decades, advances in genomics have generated a wealth of biological data, increasing the discrepancy between what is observed and what is actually known about life’s organisation at the molecular level. To gain a deeper understanding of the processes underlying the observed data, pattern recognition techniques play an essential role.

The notion of a pattern however, needs to be interpreted in a very broad sense. Essentially, we could define a pattern as everything that is the opposite of chaos. Thus the notion of organisation can be associated with a pattern. The goal of pattern recognition techniques then is to elucidate the organisation of the pattern, resulting in a wide range of subtasks such as recognition, description, classification, and grouping of patterns.

In this work we will focus on one of these tasks, classification, as a way to build models for certain biological processes, where the basic entities consist of nucleic acid sequences. Formally, the problem of classification can be stated as follows: given an input pattern \( x \), assign it to a class \( c \), where a class can be generally defined as a set of similar patterns. When patterns with a known class label are available, the problem is termed supervised classification, but this is usually just abbreviated to classification. When the class labels are not known a priori, the problem is termed unsupervised classification, more gener-
2.2 CLASSIFICATION TECHNIQUES

They are mainly referred to as clustering.

In the current work, the classes for each classification problem are known, hence we will limit our attention to the problem of supervised classification.

For the type of biological data we will deal with, a lot of the underlying mechanisms that would explain the observed behaviour still remain mysterious or unknown. In most cases only some global biological mechanisms are known, yet the details of the system remain hidden. As a result, the means to model these types of processes are limited to approaches that are able to learn the theory automatically from the observed data. Such techniques are often called machine learning approaches, and they provide a higher level mechanism for model building when lower level models (e.g. sets of differential equations) cannot be used due to a lack of theory or knowledge.

The rest of this chapter describes the general setting when using supervised classification techniques to build models. Upon these techniques we will build further when designing feature selection techniques for the classification of nucleic acid sequences.

2.2 Classification techniques

When classifying a pattern \( x \) into a class \( c \) of a set of predefined classes \( \{c_1, c_2, \cdots, c_s\} \) we need some evidence or properties of the pattern to be able to assign a class to it. The characteristics of the pattern are called features, and a pattern can thus be defined by a set of \( n \) features (often also called attributes), represented as an \( n \)-dimensional feature vector \( x = \{x_1, x_2, \cdots, x_n\} \). When the class of a given pattern is known, the pattern is said to be annotated or labelled.

Starting from a set of labelled patterns, supervised learning techniques can then be used to build a model that learns the actual mapping between the patterns and their labels. This is called the principle of pure inductive inference, or learning by induction for short. When learning by induction, given a collection of examples of a decision function \( f \), a supervised classification algorithm returns a function \( h \) that approximates \( f \). The function \( h \) is called the hypothesis or the model. Classification techniques differ from each other in how they represent \( h \) and how they search the space of all possible hypotheses.

Example 2.1

Consider the problem of learning whether to play tennis outdoor or not, depending on the weather conditions. This can be stated as a binary classification problem: (to play) \( \lor \) (not to play), where the features represent characteristics of the weather: \{Outlook, Temperature, Humidity, Wind\}. Each of these features can take a number of values, and a training set of annotated patterns can be defined (see Figure 2.1).
In order to build a classification model, a number of consecutive processing steps need to be performed. This modelling process is illustrated in Figure 2.2.

A first step in the classification process involves obtaining the data, usually bundled into a pre-processing step. This step includes the experimental set-up to acquire the data, and any operation that might be needed to transform the data into a first raw form (e.g. numerical format, image). In the next step, features are extracted from the raw data. However, not all of them might be useful, which is why a subsequent step, feature selection, is often performed to filter out irrelevant information. Using the features selected at this stage, a classification model is then built. Usually, this step involves experimentation with different classification models and model parameters. Finally, some post-processing is done, a typical example being the assessment of the performance of the classifier on an unseen set of patterns, or the combination of different classifiers.

Although depicted here as a linear flow of information, the design process is often repeated several times, each time attempting to improve some part of the process. Additionally, it should be noted that the distinction between some of the consecutive steps in the design process is not always that clearly separable. This is illustrated by the arrows connecting the steps of feature extraction, feature selection and model induction. Indeed, sometimes the selection, or even the construction of features can be incorporated into the classification algorithm (e.g. by choosing an appropriate kernel for a Support Vector Machine, as we will demonstrate later).

In an attempt to select a representative sample of classification models for our
experiments, we selected three classification algorithms, employing totally different strategies to build a hypothesis $h$: a Bayesian classifier (Naive Bayes), an (essentially linear) discriminant function (Support Vector Machine) and a decision tree (C4.5). For a more extensive overview of classification techniques, we refer to [Duda et al., 2001] and [Mitchell, 1997]. The algorithms we selected were chosen because of the following criteria: a) they can handle high-dimensional data (i.e. the number of features used to represent a pattern), b) they are fast, and can thus be applied to large datasets, c) they have been successfully applied in machine learning, and d) they are straightforward to implement or an implementation is available.

### 2.2.1 Bayesian classifiers

A general way to look at patterns stems from probability theory, where features are assumed to have a probability density or mass function (depending on whether they have discrete or continuous values), conditioned on the pattern class. Thus, a pattern $\mathbf{x}$ with associated class $c_i$ can be viewed as an observation drawn randomly from the class-conditional probability function $f(\mathbf{x}|c_i)$. In the Bayesian framework, the joint probability of observing a pattern $\mathbf{x}$ with an associated class $c_i$ can be written as

$$p(\mathbf{x}, c_i) = p(\mathbf{x}|c_i)p(c_i) = p(c_i|\mathbf{x})p(\mathbf{x}) \quad (2.2)$$

From this, Bayes rule can be derived as

$$p(c_i|\mathbf{x}) = \frac{p(\mathbf{x}|c_i)p(c_i)}{p(\mathbf{x})} \quad (2.3)$$
where $p(x) = \sum_{i=1}^{s} p(c_i)p(x|c_i)$ is the probability density function of $x$. The values $p(c_i)$ represent the prior probabilities of observing class $c_i$, and the values $p(x|c_i)$ represent the class-conditional probabilities, describing the distribution of the features over all patterns in each of the classes (likelihood). If the exact distributions of these values are known, these can be directly used to compute the posterior probabilities $p(c_i|x)$. However, in most cases the exact distributions are unknown and need to be estimated from the training data.

To see how we can use these values to define a decision function, we introduce the notion of an expected loss or risk. For a given pattern $x$ the conditional risk of assigning it to class $i$ is denoted as:

$$R_i(x) = \sum_{j=1}^{s} \lambda_{ij} p(c_j|x)$$

(2.4)

where $\lambda_{ij}$ represents a loss function, i.e. a penalty for classifying an example as $c_i$ when it should have been $c_j$ (usually $\lambda_{ij} = 0$ if $i = j$ and $\lambda_{ij} = 1$ if $i \neq j$). An optimal decision rule can then be defined as a rule that minimises the average or overall risk $R$. For a decision rule $\alpha(x)$ that assigns a class to a pattern $x$, the overall risk $R$ is given as

$$R = \int R_{\alpha(x)}(x)p(x)dx$$

(2.5)

where the integral extends over the entire feature space. As a result, to minimise the overall risk, the conditional risks $R_i(x)$ should be minimised, which is done by selecting for each pattern $x$ the class for which $R_i(x)$ is minimal. Minimising this quantity is defined as the “optimal” Bayes decision rule.

For the symmetrical or zero-one loss functions as mentioned above, selecting the class which minimises the conditional risk is equivalent to selecting the class $i$ that maximises the posterior probability $p(c_i|x)$ (maximum a posteriori rule). Thus, a pattern $x$ should be assigned to class $i$ if

$$p(c_i|x) > p(c_j|x), \forall j \neq i$$

(2.6)

In general, application of the Bayes decision rule is strongly dependent on how the class-conditional probabilities $p(x|c_i)$ can be calculated. In the most simple case, all the features that describe a pattern can be regarded as independent (i.e. not correlated) and the probabilities $p(x|c_i)$ can be factorised as

$$p(x|c_i) = p(x_1|c_i)p(x_2|c_i)\cdots p(x_n|c_i) = \prod_{j=1}^{n} p(x_j|c_i)$$

(2.7)

However, in real world classification problems, such a situation hardly occurs, and in theory the probabilities should thus be factorised in the appropriate
2.2. CLASSIFICATION TECHNIQUES

way by including the conditional dependencies. This quickly turns out to be computationally infeasible when the number of features and patterns increases. Moreover, when the dependency relationships among the features used by the classifier are unknown, an exploration of all higher order feature dependencies is infeasible.

A common procedure is then to take the simplest assumption, namely that all features are conditionally independent given the class, i.e. Equation 2.7. This method is called the Naive Bayes method, which often works good in practice, despite its manifest simplicity. This can be formalised in the following decision rule

\[ c = \arg \max_i \prod_{j=1}^n p(x_j|c_i)p(c_i) \]  

(2.8)

The method is robust, fast, and scales well to high-dimensional feature spaces and large datasets. The time complexity of the Naive Bayes method is linear, both in the number of training patterns as in the number of features.

2.2.2 Support Vector Machines

Support Vector Machines belong to a general category of statistical classification methods known as linear discriminant functions. In linear discriminant functions, the hypotheses are represented by functions that are either linear in the components of \( x \), or linear in some given set of functions of \( x \). In the most simple case, the discriminant function can be written as

\[ g(x) = \sum_{i=1}^n w_i x_i + b \]  

(2.9)

and represents an \( n \)-dimensional hyperplane. The resulting classifier can discriminate between two classes, based on the following decision rule: \( f(x) = \text{sign}(\sum_{i=1}^n w_i x_i + b) \) where the pattern is classified as \( c_1 \) if \( f(x) = 1 \), and as \( c_2 \) if \( f(x) = -1 \). The undefined case where \( f(x) = 0 \) is mostly chosen as belonging to either one of the classes (e.g. \( c_1 \) if \( f(x) \geq 0 \)). The case of two classes can be generalised to more classes by defining a set of \( s \) discriminant functions

\[ g_j(x) = \sum_{i=1}^n w_{ij} x_i + b_j, \quad j \in \{1, \cdots, s\} \]

and assigning \( x \) to \( c_j \) if \( g_j(x) > g_k(x) \ \forall k \neq j \). The resulting classifier is called a linear machine.

Evidently, for a given dataset, there might be many hyperplanes that correctly classify the training data. Support Vector Machines (SVMs, [Boser et al., 1992, Vapnik, 1995]) are a special case of linear machines, that aim at choosing the hyperplane that maximises the margin between the hyperplane and the training instances.
2. SUPERVISED LEARNING

![Diagram of hyperplanes and margin]

**Figure 2.3:** For a number of linearly separable patterns, many separating hyperplanes can be constructed (a). The SVM aims at constructing the hyperplane with maximal margin (b).

This is illustrated in Figure 2.3 where part (a) shows the general case: many hyperplanes can separate the instances from $c_1$ (empty circles) from the instances from $c_2$ (filled circles). Part (b) then shows the hyperplane with the largest margin to the training patterns. Here $d_1$ is the shortest distance from the hyperplane to the closest pattern of class $c_1$, likewise is $d_2$ for class $c_2$. The margin of the separating hyperplane is then $d = d_1 + d_2$. The idea of choosing the hyperplane with the largest margin, is that the resulting solution is more likely to classify new test samples correctly, as opposed to some of the separating hyperplanes in part (a).

As the basic SVM algorithm is a two-class classifier, we can consider a set of training patterns $x_i \in \mathbb{R}^n$ with class labels $y_i \in \{-1, +1\}$. Thus, the learning task is to find the weights $w$ and the bias $b$ for the hyperplane that maximises the margin, such that:

\begin{align}
    x_i \cdot w + b &\geq +1 \text{ for } y_i = +1 \quad (2.10) \\
    x_i \cdot w + b &\leq -1 \text{ for } y_i = -1 \quad (2.11) \\
    \Rightarrow y_i(x_i \cdot w + b) &\geq +1 \text{ for } y_i = \pm 1 \quad (2.12)
\end{align}

If we denote by $H$ the decision boundary $w \cdot x = 0$, and by $H_1, H_2$ hyperplanes that are parallel to $H$, then there exist $x_i$ for proper values of $w$ and $b$ such that:

\begin{align}
    H_1 : x_i \cdot w + b &= -1 \quad (2.13) \\
    H_2 : x_i \cdot w + b &= +1 \quad (2.14)
\end{align}
2.2. CLASSIFICATION TECHNIQUES

The margin between $H_1$ and $H_2$ is then $\frac{2}{\|w\|}$, which achieves a maximum when $\frac{\|w\|^2}{2}$ is minimised under the conditions 2.10 and 2.11. Those training patterns for which the equality in Equation 2.12 holds (i.e. are lying on $H_1$ or $H_2$), and whose removal would change the solution found, are called the support vectors. In most cases however, the training samples will not always be as clearly separable as in Figure 2.3 (b). Therefore the notion of a soft margin was introduced [Cortes and Vapnik, 1995]. In this case, slack variables $\xi_i$ are defined and Equations 2.10 and 2.11 are reformulated as:

$$x_i \cdot w + b \geq +1 - \xi_i \text{ for } y_i = +1$$

$$x_i \cdot w + b \leq -1 + \xi_i \text{ for } y_i = -1$$

$$\xi_i \geq 0 \ \forall i$$

The objective function to be minimised changes then from $\frac{\|w\|^2}{2}$ to $\frac{\|w\|^2}{2} + C(\sum \xi_i)^k$ where $C$ is a regularising term to be defined by the user. The higher $C$, the more training errors get penalized. For $k = 1$, this leads to a quadratic optimisation problem where, using the Wolfe dual formulation [Wolfe, 1961], neither the $\xi_i$, nor their Lagrange multipliers appear in the formulation. For a training set of $m$ patterns, this leads to the following formulation [Burges, 1998]:

Maximize $W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j x_i \cdot x_j$

subject to the conditions

$$0 \leq \alpha_i \leq C$$

$$\sum_{i=1}^{m} \alpha_i y_i = 0$$

The solution is then given by

$$w = \sum_{i=1}^{N_s} \alpha_i y_i x_i$$

(2.16)

where $N_s$ is the total number of support vectors and the $\alpha_i$ are the Lagrange multipliers for the constraints 2.12. The fact that in this formulation the training data only appear in the form of dot products $x_i \cdot x_j$ allows the use of so-called kernel functions, enabling the SVM to perform non-linear classification. The trick to do this, is to let the SVM perform its basic, linear discrimination, onto non-linear transformations of the training patterns. As a result, the dot product can be replaced by a kernel function $K(x_i, x_j) = (\Phi(x_i) \cdot \Phi(x_j))$ such that

$$W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j K(x_i, x_j)$$
is maximised. However, the transformation $\Phi$ does not need to be calculated explicitly, as only the result of the dot product is used, allowing a calculation directly in the input space. The kernel function captures the notion of similarity between two input patterns and needs to satisfy the following conditions:

For any set $X$, a function $K : X \times X \to \mathbb{R}$ is a kernel iff

- it is symmetric:
  $$K(x_i, x_j) = K(x_j, x_i)$$
- and it is positive semi-definite:
  $$\gamma^T K \gamma \geq 0 \ \forall \gamma \in \mathbb{R}^m$$

Some examples of kernels are:

1. The linear kernel: $K(x_i, x_j) = x_i \cdot x_j$
2. The polynomial kernel: $K(x_i, x_j) = (x_i \cdot x_j + 1)^p$
3. The Gaussian radial basis function kernel: $K(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)$

An example of the use of a polynomial kernel is given in Figure 2.4. Part (a) represents the input space, i.e. the space of features as they are originally defined by the feature construction and selection process. In this case the input space is $\mathbb{R}^2$. Clearly no optimal separating line can be found that correctly separates both classes. However, when transforming the input features into a six-dimensional space (the feature space) using a polynomial kernel of degree two (this results in the expansion to: 1, $\sqrt{2}x_1$, $\sqrt{2}x_2$, $\sqrt{2}x_1x_2$, $x_1^2$, $x_2^2$), a separating line can be constructed (c). When this line is projected into the original input space, a non-linear decision function is obtained (b).

The idea of using kernel functions is appealing when dealing with more complex objects than just simple real values. This is e.g. the case when one needs to compare DNA sequences or proteins, where similarities are often defined by domain experts, and in this case domain knowledge can be incorporated in SVMs by designing appropriate kernel functions. Conceptually, this is somewhat comparable to the incorporation of domain knowledge in Bayesian classifiers when choosing appropriate priors.

The computational complexity of SVM training algorithms is attracting increasing interest, as applications of SVMs extend to problems of larger and larger size, and new algorithms are being developed to solve the quadratic programming problem efficiently [Laskov, 2002]. One benefit of SVMs is that the complexity of the resulting classifier is characterized by the number of support vectors rather than the dimensionality of the transformed space. As a result, SVMs tend to be less prone to problems of overfitting than other methods.
2.2. CLASSIFICATION TECHNIQUES

Figure 2.4: An example of how non-linear decisions can be made by using an appropriate kernel function. The examples in (a) are not linearly separable in input space. Using a six-dimensional feature space (polynomial kernel of degree 2, (c)), a non-linear decision function can be constructed (b).

The time complexity of the linear SVM is lower order polynomial, usually approximately quadratic or cubic in the number of training patterns [Hush and Scovel, 2000]. When using other, more complex kernel functions, the running time is also heavily influenced by the kernel function (e.g. the number of multiplications for polynomial kernels), each time a dot product needs to be calculated [Joachims, 1998].

2.2.3 Decision trees

In contrast to using probabilities or metric-based methods to classify a pattern, decision trees base their classification of a pattern on a series of subsequent questions/decisions.

In general, decision trees represent a disjunction of conjunctions of constraints on the feature values of patterns. Each path from the tree root to a leaf corresponds to a conjunction of feature tests, and the tree as a whole corresponds to a disjunction of these conjunctions. More specifically, decision trees classify patterns by sorting them down the tree from the root node to some leaf node, which provides the classification of the pattern. Each node in the tree specifies a test of some feature of the pattern, and each branch descending from that node corresponds to one of the possible values for this feature. A pattern is then classified by starting at the root node of the decision tree, testing the feature specified by this node. It then moves down the tree branch corresponding to that value of the selected feature. This process is then repeated at the node on this branch until a leaf node is reached.

An example of a decision tree for the problem of playing tennis is given in Figure 2.5. The construction of a decision tree can be viewed as a recursive splitting of the dataset. At the root node, the full training set is split, and at
2. SUPERVISED LEARNING

Figure 2.5: A decision tree for the example of playing tennis.

Each child node a proper subset of the data is split further. When constructing a tree, a number of issues arise that influence the construction process:

- How many splits should we allow for each node, and which particular split will be made at each node.
- How can we prune a tree if it grows too large.
- Which class label should be assigned to a node that is impure (i.e. does not unambiguously define one class).

In principle, the number of splits at each node can vary throughout the tree, but every decision can be represented by using just binary splits.

The most important criterion in decision tree construction is thus the choice of the split that will be made. The underlying idea here is that simple, compact trees with few nodes are preferred over more complex trees (an example of the parsimony principle, in computer science often referred to as Ockham’s Razor). To this end, a procedure is applied that aims at having the subset at each leaf node as homogeneous as possible (impurity criterion). At an intermediate stage during the construction of the tree, a leaf representing an inhomogeneous set is then chosen and replaced by a test node that divides the inhomogeneous set into minimally inhomogeneous subsets, according to the impurity criterion. In a sense, the most informative feature at that particular stage is used to split the data, as it reduces the uncertainty the most. The general algorithm for the basic induction of decision tree ID3 [Quinlan, 1986] is given in Figure 2.6.
2.2. CLASSIFICATION TECHNIQUES

Algorithm ID3

Input: \( F = f_1, \ldots, f_n \): a set of features
\( C = c_1, \ldots, c_s \): the set of classes
\( S = s_1, \ldots, s_m \): the set of training patterns

\[
\text{if } P \text{ is empty}
\quad \text{return a single node with value Failure;}
\]

\[
\text{if } P \text{ only consists of patterns of the same class } c_i
\quad \text{return a single node with label } c_i;
\]

\[
\text{if } F \text{ is empty}
\quad \text{return a single node with as value}
\quad \text{the most frequent class in } P;
\]

Let \( D \) be the feature which minimises the
impurity criterion among the attributes in \( F \);

Let \( \{d_i|i=1,2,\ldots,k\} \) be the possible values of feature \( D \);

Let \( \{S_j|j=1,2,\ldots,k\} \) be the subsets of \( P \) having value \( d_j \)
for feature \( D \);

Return a tree with root labelled \( D \) and arcs labelled
\( d_1, d_2, \ldots, d_k \) going respectively to the trees
\[
\text{ID3} (F\setminus\{D\}, C, S_1),
\text{ID3} (F\setminus\{D\}, C, S_2),
\ldots,
\text{ID3} (F\setminus\{D\}, C, S_k);
\]

Figure 2.6: Algorithm of the ID3 decision tree.

It has to be noted that the tree construction process is a _greedy_ method, as
at each stage the feature that minimises the impurity function is chosen, and
no backtracking is performed to find a possibly better split resulting in a bet-
ter overall tree. As a consequence, the construction method can be seen as a
depth-first search into the space of possible hypotheses (trees), using the im-
puity criterion as a heuristic.

As a consequence, a critical step in the algorithm is the computation of the
impurity, which determines the split to be taken at a node. Several impurity
functions can be chosen, yet the ones mostly used are information gain and
gain ratio. Both types of functions rely on the concept of entropy, which has
to be understood here in the sense of information theory (Shannon-entropy
[Shannon, 1948]). A given subset of patterns \( S \) can be regarded as a distribu-
tion over the class labels, and its entropy can be calculated as

\[
H(S) = - \sum_{i=1}^{s} p(c_i) \log_2 p(c_i) \tag{2.17}
\]
where \( p(c_i) \) denotes the proportion of patterns in \( S \) belonging to class \( c_i \). The information gain \( IG(S, D) \) then represents the expected reduction in entropy (uncertainty) when splitting on feature \( D \), and can be calculated as

\[
IG(S, D) = H(S) - H(S|D) = H(S) - \sum_{j \in V(D)} \frac{|S_j|}{|S|} H(S_j)
\]

(2.18)

where \( V(D) \) denotes the number of possible values for feature \( D \) and \( S_j \) is the subset of \( S \) for which feature \( D \) has value \( j \).

The best feature to be used as a decision criterion is then the one for which \( IG(S, D) \) is maximised (as maximising the information gain minimises the impurity). However, using information gain as a criterion has a drawback, inherent to the entropy, favouring features with a larger number of possible values over features with a smaller number of possible values. To avoid this drawback, the information gain should be scaled by the entropy of \( S \) with respect to the values of feature \( D \), resulting in another criterion called the gain ratio:

\[
\text{GainRatio}(S, D) = \frac{IG(S, D)}{-\sum_{j \in V(D)} \frac{|S_j|}{|S|} \log_2 \frac{|S_j|}{|S|}}
\]

(2.19)

The basic ID3 algorithm has a number of drawbacks, the most important one being the tendency to overfit the training data. This means the classification tree learns the biases of the training set, resulting in bad generalisation performance when confronted with unseen patterns. To draw an analogy with human learning, we can compare overfitting to “learning by heart”. When confronted with an example that was learned, our answer will be perfect, but when having to decide upon a new example we stand a high chance of getting the answer wrong.

Furthermore, the algorithm in its basic form cannot handle continuous-valued features. To cope with these problems, improved versions of ID3, such as C4.5 [Quinlan, 1993] were developed. It has to be noted that all these classification trees are based on the central idea of Classification And Regression Trees (CART), introduced by Breiman in 1984 [Breiman et al., 1984].

In our experiments we will use C4.5 as a decision tree constructor (release 8), for which free source code is available. C4.5 can handle continuous-valued features by using cut-points and introduces a number of measures to avoid overfitting: stopped splitting and pruning. Furthermore, it can handle patterns with missing feature values. The first technique to avoid overfitting is to stop splitting nodes when the number of patterns that would be assigned to the child nodes falls below a certain threshold, or when splitting the data does not yield a statistically significant improvement. The second technique, pruning, aims at simplifying the tree by eliminating subtrees. The pruning technique used in C4.5 is called reduced error pruning (REP), the algorithm for which is given in Figure 2.7.
2.3. ASSESSING CLASSIFICATION PERFORMANCE

Algorithm REP

1. Split the training patterns into a training set and a validation set
2. Build a decision tree on the training set
3. Starting bottom up for each non-leaf node
   a) Evaluate the validation set accuracy of pruning the subtree rooted at the node
   b) Greedily remove the node that most improves validation set accuracy, with its corresponding subtree
   c) Replace the removed node by a leaf with the majority class of the corresponding examples, or by a probability estimate of the class membership, based on instance counts
4. Stop when pruning starts hurting the accuracy on the validation set

Figure 2.7: Algorithm of reduced error pruning (REP).

Pruning the tree then mostly results in a tree that is better able to generalise to unseen patterns. Another advantage is the improved readability and interpretability by a human user, which is an attractive feature of decision trees. This advantage distinguishes them from more “black-box” classification algorithms such as the SVM, and is one of the main reasons for using decision tree classifiers.

For a training set of \( N \) training patterns having \( D \) features, the time complexity of training a decision tree can be approximated as follows. Assuming a binary decision tree, the number of levels in the tree is \( O(\log N) \). At level \( i \) there are \( 2^{i-1} \) nodes and at each node an entropy calculation and a sorting operation needs to be done. Calculating the entropy can be done in \( O(N) \) time for each feature, and sorting can be achieved in \( O(N \log N) \) time. As a result, the complexity at each node can be approximated as \( O(DN \log N) \). At each level, there are a fixed number of nodes, so this is also in \( O(DN \log N) \). Consequently, the total time complexity of the tree is \( O(DN(\log N)^2) \).

2.3 Assessing classification performance

It is known that there is no such thing as the best classification algorithm, when considering a broad scale of classification problems (the no free lunch theorem). It is the type of the problem, prior knowledge, and other information that determines which type of classifier should provide the best performance.

An important aspect thus, when confronted with a specific classification task,
is the evaluation and comparison of the performance of various classifiers and their parameters. To get a more or less reliable result, the trained classifier should be tested on an independent set of patterns. The set of patterns used for training is then termed the *training set* while the set of patterns hold out to test the performance of the trained classifier is termed the *validation set* or the *test set*. A validation on the test set is used to assess the ability of the classifier to generalise to new patterns, as the performance on the training set is often overly optimistic, and sometimes focuses too much onto biases in the training data (overfitting).

A commonly used method to evaluate a classifier is by using *n-fold crossvalidation*. In this procedure, the set of available patterns is partitioned into *n* sets of equal size. For each of the *n* sets, the classifier is trained on the patterns in the remaining parts and tested on the chosen part, and the overall result is averaged over the *n* trials.

The performance of the classifier can be analysed by considering the errors that have been made on the validation set(s). For two-class classification problems, the performance of a classifier can be summarised in four numbers:

- **TP**: the number of true positives (actual positive patterns predicted as positives)
- **TN**: the number of true negatives (actual negative patterns predicted as negatives)
- **FP**: the number of false positives (actual negative patterns predicted as positives)
- **FN**: the number of false negatives (actual positive patterns predicted as negatives)

From these numbers, a series of performance measures have been derived, all aiming at summarising the four numbers into one, to make it easier to compare classifiers:

- **error rate**:
  \[
  \text{error rate} = \frac{\text{FP} + \text{FN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}
  \] (2.20)

- **accuracy**:
  \[
  \text{accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} = 1 - \text{error rate}
  \] (2.21)

- **TP rate/recall/sensitivity**:
  \[
  \text{TP rate/recall/sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FN}}
  \] (2.22)
2.3. ASSESSING CLASSIFICATION PERFORMANCE

- FP rate:
  \[
  \frac{FP}{FP + TN} \tag{2.23}
  \]

- precision/specificity\(^1\):
  \[
  \frac{TP}{TP + FP} \tag{2.24}
  \]

- correlation coefficient:
  \[
  \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TN + FN)(TP + FP)(TN + FP)(FN + TP)}} \tag{2.25}
  \]

- approximate correlation:
  \[
  \frac{1}{2} \left( \frac{TP}{TP + FN} + \frac{TP}{TP + FP} + \frac{TN}{TN + FP} + \frac{TN}{TN - FN} \right) - 1 \tag{2.26}
  \]

These methods have been commonly used in machine learning research, yet have one main drawback: they assume a balanced class distribution (i.e. approximately as many positive as negative training/testing patterns). However, in many real-world datasets, like biological datasets, the class distribution is highly imbalanced. An example of this, as we will see later, is splice site prediction, where the number of pseudo sites (negative patterns) is a number of magnitudes higher than the number of actual sites. Imagine a dataset with 98% negative patterns and a classifier that always outputs the class “negative”. This classifier would then have an accuracy of 0.98, which would be hard to beat by a classifier that tries to capture the real dependencies between the features.

Therefore, new measures have been suggested that try to take into account the class imbalance. One such measure is the F-measure, introduced in [van Rijsbergen, 1979]:

\[
F_\beta = \frac{(\beta^2 + 1) \text{ precision} \cdot \text{ recall}}{\beta^2 \text{ precision + recall}} \tag{2.27}
\]

The \(F_1\)-measure (\(\beta = 1\)) is most often used and represents the harmonic mean of precision and recall. Other values for \(\beta\) can be used to give more weight to either one of these values. A measure that was recently introduced to incorporate content-balancing for imbalanced datasets is the \(q^9\) measure [Zhang, 2002]:

\[
q^9 = \begin{cases} 
\frac{TN - FP}{TP + FN} & \text{if } TP + FN = 0 \\
\frac{TN + FP}{TP - FN} & \text{if } TN + FP = 0 \\
1 - \sqrt{\frac{FN}{(TP + FN)^2}} + \frac{FP}{(TN + FP)^2} & \text{if } TP + FN \neq 0 \text{ and } TN + FP \neq 0
\end{cases} \tag{2.28}
\]

\(^1\)It has to be noted that several definitions of specificity exist. In the other case specificity is defined as \(\frac{TN}{FP + TN}\).
2. SUPERVISED LEARNING

Figure 2.8: Receiver-Operator-Curve (ROC). The left graph shows the result for a sample of measurements at different classification thresholds. The dotted line shows the behaviour of a random guessing classifier. The graph on the right demonstrates how to compare two classifiers. Better classifiers tend more toward the upper left corner of the graph.

Another way to evaluate the performance of a classifier is by analysis of the so-called Receiver-Operator-Curve (ROC), introduced in [Provost and Fawcett, 1997]. On such a curve, the FP-rate is plotted on the x-axis versus the TP-rate (sensitivity or recall) on the y-axis. By varying the decision threshold, several values can be obtained for these rates. This is shown in Figure 2.8 (left), where a curve is fitted through these points (the dotted line represents the behaviour of a random guessing classifier).

Classifiers can then be evaluated by comparing their ROC graphs, where better classifiers are characterised by curves that are more situated toward the top left corner of the graph, as shown in Figure 2.8 (right).

A subsequent step in the modelling process is to determine whether there is a significant difference between the performance of two classifiers. The ability of a statistical test to determine such differences is usually denoted by two types of errors. Type I errors are made when the null hypothesis (the algorithms perform the same) is rejected when no difference exists. If, on the other hand, we accept the null hypothesis when a difference exists, then we incur a type II error. The probability of detecting a difference when a difference exists is called the power of the test, and is equal to $1 - P($type II error$)$. Dietterich reviews five statistical tests [Dietterich, 1998], and concludes that two of them, McNemar’s test and the 5x2 crossvalidation (cv) t test, have low type I error and reasonable power. He proposes to use McNemar’s test if, due to high computational cost, the algorithms can be executed only once. For algorithms that can be executed 10 times, he proposes to use the 5x2 cv t test.

The McNemar test is based on the idea that there is little information in
2.4. SUMMARY

the numbers of instances for which both classifiers get the correct result, or for which both get an incorrect result. Denote by \( n_1 \) the number of patterns that was misclassified by classifier A but not by classifier B. Similarly, let \( n_2 \) denote the number of examples misclassified by B, but not by A. Then the following statistic is approximately distributed as \( \chi^2 \) with one degree of freedom:

\[
\frac{(|n_1 - n_2| - 1)^2}{n_1 + n_2}
\]

In [Alpaydin, 1999], Alpaydin suggests a variant of the 5x2 cv t test, the combined 5x2 cv F test. This test combines multiple statistics to get a more robust test, and has lower type I errors and a higher power than 5x2 cv proper. In this test, five replications of two-fold crossvalidation are performed. In each replication, the dataset is divided into two equal-sized sets. \( p_i^{(j)} \) is the difference between the performance measure of the two classifiers on fold \( j = 1, 2 \) of replication \( i = 1, \cdots , 5 \). The average on replication \( i \) is \( \bar{p}_i = (p_i^{(1)} + p_i^{(2)})/2 \), and the estimated variance is \( s_i^2 = (p_i^{(1)} - \bar{p}_i)^2 + (p_i^{(2)} - \bar{p}_i)^2 \). Then the following statistic is approximately \( F \) distributed with 10 and 5 degrees of freedom:

\[
\frac{\sum_{i=1}^{5} \sum_{j=1}^{2} (p_i^{(j)})^2}{2 \sum_{i=1}^{5} s_i^2}
\]

2.4 Summary

In this chapter we introduced the reader to the concepts of supervised learning, providing the basics of our classification framework. We discussed a representative subset of inductive techniques that will be used later on in our experiments: a Bayesian classifier (Naive Bayes), a linear discriminant technique (the Support Vector Machine) and a decision tree (C4.5). Furthermore, we addressed the question of how different classification models could be compared and evaluated.
Chapter 3

Feature selection techniques

To live content with small means;
to seek elegance rather than luxury,
and refinement rather than fashion;
to be worthy, not respectable, and wealthy,
not rich; to listen to stars and birds,
babes and sages, with open heart;
to study hard; to think quietly, act frankly,
talk gently, await occasions, hurry never; in a word
to let the spiritual, unbidden and unconscious,
grow up through the common–this is my symphony.

- William Henry Channing

3.1 Introduction

The selection of relevant features has become a challenging research topic during the past decades, as datasets arose from fields like text classification, combinatorial chemistry and genetics. Many of these datasets contain hundreds or thousands of features, the majority of which is often redundant or irrelevant, yet is included in the dataset due to the absence of sufficient domain knowledge. As most classifiers were originally not designed to cope with a large amount of irrelevant features, feature selection techniques were developed to fill this lacuna [Jain et al., 2000, Guyon and Elisseeff, 2003].

The objectives of feature selection are:

- to improve the prediction performance of the classifier,
- to provide faster and more cost-effective classifiers,
3.1. INTRODUCTION

Figure 3.1: A situation of feature selection techniques within dimensionality reduction.

- to produce simpler classification models (e.g. a smaller decision tree),
- and to gain insight in the underlying process that generated the data.

The exact definition of the feature selection problem thus depends on its context, yet the most frequent definition is to select that subset of features that results in the best classification performance. The rationale behind this is the fact that irrelevant or redundant features often behave like noise in the data, confusing the classification model and degrading its performance. The removal of such features then results in a restricted subset of features with equal or better classification performance than the full feature set. As a direct consequence, less features are needed to store the data, and classification will be sped up. Additionally, reducing the number of features helps the human expert to focus on a subset of relevant features, providing the ability to get a better insight in the processes described by the data.

The concept of feature selection is very closely related to a number of other techniques, illustrated in Figure 3.1. In its most general form, feature selection is an example of a *dimensionality reduction* technique. Other dimensionality reduction techniques include methods based on projections (e.g. principle components analysis (PCA) and variants thereon) and methods based on compression. Feature selection techniques differ from these techniques in the sense that they do not change the input features. This is important when using feature selection techniques to gain insight in the underlying process.

Within the subset of feature selection techniques, two refinements can be distinguished: feature ranking techniques and feature weighting techniques. These refinements gradually reveal a more detailed picture of the relevance of fea-
3. FEATURE SELECTION TECHNIQUES

tures. Instead of returning a single “best” set of features by the feature selection scheme, feature ranking methods rank the features from most relevant to least relevant. This ranking can then be used to discard features. Feature weighting algorithms even go one step further: they assign a feature relevance weight to each individual feature. These weights can then be sorted to provide a ranking, which can be subsequently used to eliminate features (e.g. by choosing a threshold weight and eliminating all features with a weight below the threshold).

An important question regarding the definition of “relevant subset of features” is the definition of relevance. In [John et al., 1994], the authors review some of the common definitions of relevance and suggest that two degrees of relevance are needed: weak and strong. A feature $X$ is **strongly relevant** if the removal of $X$ alone will result in performance deterioration of an optimal Bayes classifier. A feature $X$ is **weakly relevant** if it is not strongly relevant and there exists a subset of features, $S$, such that the performance of a Bayes classifier on $S$ is worse than the performance on $S \cup \{X\}$. A feature is **irrelevant** if it is not strongly or weakly relevant.

While, for most classification problems, optimal subsets will contain only relevant features, some care has to be taken when linking relevance automatically to optimal feature subsets. In [Kohavi and John, 1997], some examples are given, illustrating the fact that relevance of a feature does not necessarily imply that it is in the optimal feature subset. Similarly, irrelevance does not always imply that a feature should not be in the optimal feature subset. This could happen when classifiers use restricted hypothesis spaces, that cannot utilize all features.

The performance of a classifier depends on the interaction between a number of parameters, such as training set size, number of features, and classifier complexity. The fact that the size of the training set, needed to estimate a function of several variables to a given degree of accuracy, grows exponentially with the number of features is termed the **curse of dimensionality** [Bellman, 1961]. When, for a fixed training set size, the number of features is increased, the number of unknown parameters for the classifier also increases. Correspondingly, the reliability of the parameter estimates decreases, and the performance of the classifier may degrade with an increase in the number of features. This phenomenon is known as the **peaking phenomenon** [Raudys and Jain, 1991]. As a result of these phenomena, one should try to minimise the number of irrelevant features, hence motivating the use of dimensionality reduction techniques.

The feature selection problem can thus be formulated as a search through the space consisting of all possible feature subsets. For a given number of $n$ features, a complete search of this space would require the evaluation of $2^n - 1$ subsets, which quickly gets computationally infeasible when $n$ grows larger. Cover and Van Campenhout have shown that no non-exhaustive sequential
feature selection procedure can be guaranteed to produce the optimal subset in general [Cover and Van Campenhout, 1977]. Therefore, the search for good feature subsets necessarily depends on heuristic algorithms. According to Blum and Langley [Blum and Langley, 1997], this search process can be characterized by four basic properties:

- a starting point in the search space
- an organisation of the search
- an evaluation strategy of the selected subset
- a stopping criterion for halting the search.

Depending on these criteria, feature selection algorithms can be classified into three main classes:

- Filter techniques: features are removed by looking at the intrinsic properties of the data. In most cases a feature relevance score is calculated and low scoring features are removed.

- Wrapper techniques: various subsets of features are generated and evaluated. The evaluation of a specific subset of features is obtained by training and testing a specific classification model.

- Embedded techniques: the feature selection mechanism is built into the classification model, making directly use of the parameters of the induction model to include or reject features.

During the rest of this chapter we will discuss each class of feature selection techniques. We will also discuss a special case of filter techniques: feature selection techniques based on signal processing.

As an extension of feature selection techniques, we will shortly discuss the more general problem of feature weighting. At the end of the chapter we then describe an implementation of a toolkit for feature selection techniques, adapted for large datasets with many features. This toolkit is used for the various feature selection experiments in the following chapters.

### 3.2 Filter approaches

Filter approaches for feature selection are characterised by the fact that they only use the intrinsic properties of the training set to decide which features to keep or to eliminate. As opposed to wrapper methods, they are thus independent of the classification algorithm to be used afterwards (Figure 3.2).

The majority of all filter approaches works as follows. In a first step, a relevance score for each feature is calculated. Subsequently, these scores are sorted and low scoring features are removed (typically some threshold is chosen for
3. FEATURE SELECTION TECHNIQUES

![Diagram](image)

Figure 3.2: The filter approach to feature selection.

the feature scores). The resulting subset of features is then fed as input to a classifier system. We note that this class of techniques essentially produces a feature weighting scheme, which is used afterwards to rank and select features.

To determine the weight (or relevance) of a feature, the following scores are commonly used. In these formulas $X_j$ represents a feature variable, $x_j$ represents an instantiation of this variable (a particular value), $c_i$ denotes class $i$ and $v$ denotes the number of discrete values feature $X_i$ can have:

- **Feature-class entropy**
  \[ H_C(X_i) = - \sum_{j=1}^{v} p(x_j|c_1) \log_2 p(x_j|c_1) + p(x_j|c_2) \log_2 p(x_j|c_2) \]  
  (3.1)

- **Euclidean distance**
  \[ \text{Euc}_C(X_i) = \sqrt{\sum_{j=1}^{v} (p(x_j|c_1) - p(x_j|c_2))^2} \]  
  (3.2)

- **Kullback-Leibler divergence (also called relative entropy)**
  \[ KL_C(X_i) = \sum_{j=1}^{v} p(x_j|c_1) \log_2 \frac{p(x_j|c_1)}{p(x_j)} + \sum_{j=1}^{v} p(x_j|c_2) \log_2 \frac{p(x_j|c_2)}{p(x_j)} \]  
  (3.3)

- **Information gain** (see 2.18, sometimes referred to as mutual information) and gain ratio (see 2.19)

Most filter approaches handle the case of continuous-valued features in the same way, by first discretizing them and then applying one of the above-mentioned measures to calculate the feature weights. A special case where discretization and feature selection is combined in one algorithm is given in

---

1For simplicity, we show the formulas for the basic two-class classification problem.
3.2. FILTER APPROACHES

[Liu and Setiono, 1995]. In this case the $\chi^2$ statistic is used as a feature relevance score.

Due to recent advances in genomics (especially the advent of micro-array data), special score for continuous-valued features were proposed to measure the relation between a feature (in the case of micro-arrays mostly a gene) and the problem class. The following notation is used: $\mu_1$ and $\mu_2$ represent within-class mean feature values (e.g. mean expression levels) for class $c_1$ and $c_2$, and $\sigma_1$ and $\sigma_2$ are the standard deviations of the feature values within $c_1$ and $c_2$.

- signal-to-noise measure (also referred to as P-measure):
  \[
  \text{SNR}_C(X_i) = \frac{|\mu_1 - \mu_2|}{\sigma_1 + \sigma_2} \tag{3.4}
  \]

- $t$-score:
  \[
  t_C(X_i) = \frac{|\mu_1 - \mu_2|}{\sqrt{\frac{n_1\sigma_1^2 + n_2\sigma_2^2}{n_1 + n_2}}} \tag{3.5}
  \]
  where $n_1$ and $n_2$ denote the number of patterns in class $c_1$ and $c_2$ respectively.

The classic filter approaches described above expose one of the major trade-offs in feature selection: the choice either to be able to deal with high-dimensional feature spaces, or to be able to correctly model the feature dependencies. Indeed, the computation of a relevance score for each feature separately enables filter methods to scale to datasets with tens of thousands of features.

However, this ability comes at the price of discarding the underlying structure of the problem, which most likely involves feature dependencies. For this reason, traditional filter techniques might not always find good feature subsets, as it is known that e.g. two features might score badly when considered separately, yet score good when observed together.

To deal with the problem of feature dependencies, filter strategies were designed that were able to include those dependencies. In practice, these techniques make use of feature-feature correlations, which comes at the expense of an $O(n^2)$ calculation to build the two-dimensional matrix of feature correlations. In theory, to capture all feature dependencies, one should also consider higher-order correlations, yet this quickly turns out to be computationally infeasible (sometimes even the $O(n^2)$ operation proves infeasible). As a result, heuristics are used to find good feature subsets, based on the feature-feature and feature-class correlations.

We will now discuss two filter approaches that take into account feature dependencies. The first one is a subset selection algorithm (CFS, [Hall and Smith,
3. FEATURE SELECTION TECHNIQUES

1999), the second one is a feature ranking method based on Markov blankets [Koller and Sahami, 1996].

Correlation-based Feature Selection (CFS)

The central idea of CFS is the following heuristic:

“Good feature subsets contain features that are highly correlated with the class, yet uncorrelated with each other.”

To pour this into a mathematical formulation, a merit function is defined for a feature subset $S$:

$$\text{merit}(S) = \frac{k \cdot \overline{r_{fc}}}{\sqrt{k + k(k - 1) \alpha \overline{r_{ff}}}}$$

where $k$ denotes the number of features in $S$, $\overline{r_{fc}}$ is the average feature-class correlation and $\overline{r_{ff}}$ is the average feature-feature correlation, and $\alpha$ denotes a scaling factor (usually $\alpha = 0.25$) [Hall, personal communication]. In CFS, the feature-class correlations are calculated using the gain ratio (see 2.19), and the symmetrical uncertainty (SU) is used to calculate the feature-feature correlations:

$$\text{SU}(X_i, X_j) = 2 \frac{IG(X_i, X_j)}{H(X_i) + H(X_j)}$$

where

$$IG(X_i, X_j) = H(X_i) - H(X_i | X_j)$$

and

$$H(X_i | X_j) = \sum_{x_j} \sum_{x_i} p(x_j)p(x_i | x_j) \log_2 p(x_i | x_j)$$

The heuristic search algorithm then starts from the empty set of features, and uses a best-first search with a halting criterion of five consecutive fully expanded non-improving subsets. The subset with the highest merit found during the search will be selected.

Markov blanket filter algorithm (KS algorithm)

This algorithm, introduced by Koller and Sahami, eliminates features whose information content is subsumed by some number of the remaining features. The central idea of the algorithm is a Markov blanket.

Let $X$ denote the full set of features and $M$ some set of features that does not contain $X_i$. Then $M$ is a Markov blanket for $X_i$ if $X_i$ is conditionally independent of $X - M - \{X_i\}$ given $M$.

Two sets of variables $A$ and $B$ are conditionally independent given some set of variables $Z$ if, for any assignment of values $a$, $b$, and $z$

$$p(A = a | Z = z, B = b) = p(A = a | Z = z)$$
Algorithm KS (Koller-Sahami)

1. Calculate the cross-entropy of the class distribution given pairs of features
   \[ \gamma_{ij} = KL(p(C|X_i = x_i, X_j = x_j), p(C|X_j = x_j)) \]
   of every pair of features \( X_i \) and \( X_j \).

2. Instantiate \( G \) to \( X \) and iterate the following steps until some pre-specified number of features have been eliminated:
   (a) For each feature \( X_i \in X \), let \( M_i \) be the set of \( K \) features \( X_j \) in \( G\{X_i\} \) for which \( \gamma_{ij} \) is smallest.
   (b) Compute \( \delta_G(X_i|M_i) \) for each \( i \).
   (c) Choose the \( i \) for which this quantity is minimal, and define \( G = G\{X_i\} \).

**Figure 3.3:** Pseudo code for the Markov blanket filter approach of Koller and Sahami.

This means that \( B \) gives no information about \( A \) beyond what is already in \( Z \). An approximate algorithm is then suggested that, starting from the full feature set, iteratively removes the feature with the “best” Markov blanket.

The pseudo code for the algorithm is given in Figure 3.3. In step 2a, for each feature a possible Markov blanket is defined by selecting the \( K \) features \( X_j \) for which the class \( C \) and \( X_i \) are as most conditionally independent as possible given \( X_j \). The parameter \( K \) determines the size of the Markov blanket and exponentially increases running time as \( K \) gets larger. Typical values for \( K \) are \( \{0, 1, 2\} \). In step 2b, the expected cross entropy \( \delta_G(X_i|M_i) \) is used to approximate how close \( M_i \) is to being a Markov blanket for \( X_i \). This quantity is defined as:

\[
\delta_G(X_i|M_i) = \sum_{x_{M_i}, x_i} p(M_i = x_{M_i}, X_i = x_i) \cdot KL_{f_{M_i,f_i}}
\]

with

\[
KL_{f_{M_i,f_i}} = KL(p(C|M = x_M, X_i = x_i), p(C|M = x_M))
\]

where the Kullback-Leibler divergence \( KL \) between two distributions \( \mu \) and \( \sigma \) over a probability space \( \Omega \) is defined as

\[
KL(\mu, \sigma) = \sum_{x \in \Omega} \mu(x) \log \frac{\mu(x)}{\sigma(x)}
\]

In step 2c, the feature for which \( M_i \) most closely resembles a Markov blanket is eliminated and the process is repeated. In the limit, the elimination of features can be iterated until the empty set of features is reached. The KS algorithm can thus be used as a feature ranker.
3. FEATURE SELECTION TECHNIQUES

3.3 Wrapper approaches

The main disadvantage of filter approaches is the fact that they ignore the effect of the selected feature subset on the performance of the classifier to be used afterwards. To resolve this problem, Kohavi and John introduced the concept of wrapper-based feature selection [Kohavi and John, 1997]. In this approach, the feature selection is done using the induction algorithm as a black box. The evaluation of a specific subset of features is obtained by training a classification model, and using either cross-validation on the training set, or a separate training and holdout set to assess the goodness of a feature subset. As a consequence, the wrapper method is tailored to a specific classification model. To search the space of all feature subsets, a search algorithm is then “wrapped” around the classification model (Figure 3.4).

As the space of feature subsets grows exponentially with the number of features, heuristic search methods are used to guide the search for a good subset. In the context of feature selection, these algorithms can be divided in two categories: sequential algorithms and randomized algorithms.

Sequential algorithms [Kittler, 1978] either start with the full or empty feature set, and proceed by greedily adding or removing features. Sequential backward elimination (SBE) starts with the full set of \( n \) features, and considers each of the \( n \) subsets of \( n - 1 \) features by removing each feature once. From these \( n \) subsets, the one giving the highest classification performance (either determined by cross-validation, or on a separate holdout set) is chosen. The process is then repeated for the set of \( n - 1 \) remaining features and so on until some termination criterion is fulfilled.

Similarly sequential forward selection (SFS) can be defined, where the initial state is the empty set of features, and features are greedily added. A number of variants on SBE and SFS have been suggested, such as the plus \( q \) take-away \( r \) algorithm [Ferri et al., 1994] and beam search [Siedlecky and Sklansky, 1988]. In the plus \( q \) take-away \( r \) algorithm, the best \( q \) features are added at each step, and subsequently the \( r \) worst features are eliminated. Beam search is a best-first search, maintaining a fixed-size queue of promising subsets, iteratively

![Figure 3.4: The wrapper approach to feature selection.](image-url)
exploring the best states in the queue. The computational complexity of all these methods is quadratic, as can be easily seen for the basic case (SBE and SFS): the algorithms perform \( n + (n - 1) + (n - 2) + \cdots + 2 + 1 = \sum_{i=1}^{n} \frac{n(n+1)}{2} = O(n^2) \) evaluations.

Randomized search algorithms differ from sequential algorithms in quite some aspects. Instead of starting with the full or empty set of features, these algorithms generally start somewhere in between, by generating a random subset of the features. The search is then proceeded by randomly adapting the initial solution until a stopping criterion is fulfilled. Techniques based on simulated annealing or hill-climbing start from an initial guess and apply random changes until the solution does not change any more, or a fixed number of iterations have been elapsed [Skalak, 1994]. However, these methods are prone to getting stuck in local extrema. As an alternative to this, population-based techniques such as genetic algorithms (GA, [Holland, 1975]) were devised. Instead of working with one solution, these techniques operate on a whole set of possible solutions, termed the population. Using an iterative approach that mimics nature’s evolutionary principle, new generations of the initial population are created using a set of three operators:

- selection: a number of good individuals is chosen and copied from one generation into the next (survival)
- mutation: a certain parameter of a chosen individual is adapted (mutated) and the resulting individual is copied to the next generation
- crossover: the parameters of two individuals are combined, to produce a new individual that is copied to the next generation.

In this way, a GA can be viewed as a stochastic iterative sampling procedure. In the original GA, individuals were represented as a string of bits, yet later on strings representing any kind of parameter type were used. GAs have been particularly useful when faced with complex optimisation problems. Examples of these are non-analytical functions, like the optimisation of the topology of a neural network [Saeys and Van Marck, 2000].

When applied to feature selection, the solutions in the population represent feature subsets: a 1 indicates the presence of a feature, a 0 the absence of a feature. At the end of the iterative process, the best feature subset encountered during the "evolutionary" search in feature subset space is returned as the final solution. In the literature, GAs have been frequently and successfully used as a wrapper method for feature selection [Vafaie and DeJong, 1993, Kudo and Sklansky, 2000].

The time complexity of a GA depends on a number of parameters. For a GA with a population size \( P \), running for \( I \) iterations and using an elitist approach of \( E \) individuals (the best \( E \) individuals automatically survive to the next generation) the number of model evaluations can be calculated as
3. FEATURE SELECTION TECHNIQUES

\[ P + (I - 1)(P - E) \]. This formula thus shows that the run-time complexity of a GA is, in principle, not dependent on the number of features. In practice however, the population size and the number of iterations do depend on the number of features, as too small a population size will lead to premature convergence and a small amount of iterations will cause the algorithm to stop too early, before convergence has been reached.

Furthermore, some other parameters of the GA need to be tuned carefully. These include the mutation rate, the crossover rate, the number of elitists and the selection scheme.

3.3.1 A new wrapper method: the Constrained Estimation of Distribution Algorithm (CDA)

Although genetic algorithms have been useful for many complex optimisation problems, they have a number of drawbacks. The main critics for standard GAs include the large number of parameters that have to be tuned, the difficult prediction of the movements of the populations in the search space and the fact that there is no mechanism for capturing the relations among the variables of the problem.

To embed the ideas that underlie GAs in a more theoretically sound system, a more probabilistic approach to stochastic iterative sampling has been proposed: the Estimation of Distribution Algorithm (EDA) [Larrañaga and Lozano, 2001]. In this approach, the notion of a population is still preserved, but the iterative sampling process is changed. Instead of using selection, crossover and mutation, a probability distribution modelling the parameter values is explicitly constructed for a chosen subset of good solutions. The new population is then generated by sampling this distribution. Figure 3.5 summarises the steps in the EDA.

The essential steps in the EDA are the estimation of the probability distribution (step 3) and the sampling of this distribution (step 4). Depending on the complexity of the estimation model, various degrees of interactions between the encoded variables are modelled. Afterwards the new population is created by sampling individuals from this model. Just like GAs, EDAs have been applied to a wide range of optimisation problems, including feature selection, reporting good results [Inza et al., 1999, Larrañaga and Lozano, 2001, Saey et al., 2002].

The main action to be taken in an EDA-based evolutionary algorithm, is the construction of a probability distribution, modelling the variables and their dependencies. In general, most EDAs can be represented as probabilistic graphical models [Pearl, 1988]. The structure of the graphical model determines the expressivity of the EDA to model dependencies between variables, and constitutes the major criterion to distinguish subclasses of EDAs.

The most common sub-classification distinguishes between EDAs modelling univariate, bivariate and multivariate dependencies between the variables. A
3.3. WRAPPER APPROACHES

Figure 3.5: The concept of the Estimation of Distribution Algorithm.

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: every bit denotes a feature that can either be included (1) or excluded (0). Figure 3.6 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 \( X_1, \ldots, X_8 \). 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, [Mühlenbein, 1998]) is a very simple model, assuming variables (in this case features) 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, [Pelikan and Mühlenbein, 1999]), pairwise interactions between variables are modelled, and in the case of multiple dependencies, higher order interactions between the variables are modelled. Examples of these are the Bayesian Optimization Algorithm (BOA, [Pelikan et al., 1999]) and the Estimation of Bayesian Networks Algorithm (EBNA, [Etxebarria et al., 1999]).

When applying EDAs to feature selection problems of high dimension, a num-
3. FEATURE SELECTION TECHNIQUES

Figure 3.6: 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, like feature selection. The notation $x_i^j$ denotes the instantiation of variable $i$ with value $j$.

A number of observations can be made:

- many feature subsets result in a similar classification performance,
- many features are irrelevant,
- the algorithm spends most of its time in feature subsets containing about half of the total number of features.

To improve the run-time complexity of EDA based wrapper methods, we developed the Constrained Estimation of Distribution Algorithm (CDA). The CDA drastically narrows down the feature subset search space by setting an upper limit $U$ to the number of features that a subset may contain. In terms of a distribution, it thus creates a bounded version of the distribution (hence the name constrained distribution), by allowing only samples with at most $U$ variables having a value 1 in the distribution. This can be enforced by a post processing step after the new distribution has been sampled. For each sample in the distribution, we check if at most $U$ variables have been set to 1. If not,
3.3. WRAPPER APPROACHES

we randomly set a number of variables to 0, until at most $U$ variables have a value 1.

The CDA has the following advantages:

- A huge reduction in the subset search space. As an example consider a dataset with 400 features. The normal (unconstrained) EDA would then search through a space of $2^{400}$ subsets. On the other hand, a CDA with upper limit $U = 100$ would search through a space of “only” $\sum_{i=0}^{100} \binom{400}{i} \approx 3.3E^{96}$ feature subsets, a reduction by 23 orders of magnitude.

- Faster calculation of the evaluation of a subset. By constraining the size of the subset to a low number of features, it will be faster for a classifier to build a model, and the resulting model will be simpler.

- Scalability to high dimensional datasets. This is especially useful when it is known a priori that not all the features that describe the data are relevant. This is the case for the biological datasets we want to handle.

- Scalability to more complex classifiers, such as SVMs with a higher order polynomial kernel.

When comparing the CDA to sequential wrapper methods for very high dimensional datasets, it is clear that the method will still be able to produce results, whereas starting a SBE or SFS algorithm might already be infeasible (e.g. a dataset with 10,000 features). The same is true when comparing the CDA with the CFS and the Koller-Sahami algorithm, which are also at least quadratic in the number of features.

In [Saeys et al., 2003], we compared the running time and classification performance of the CDA to the SBE for three algorithms: NBM, linear SVM, and ninth order polynomial SVM. While the classification performance of both feature selection methods was comparable, the CDA needed considerably less time. The gain in time when comparing both algorithms increases when the upper limit $U$ decreases, or when the number of features increases. This is shown in Table 3.1.
3. FEATURE SELECTION TECHNIQUES

### Table 3.1: Comparisons of the running times for SBE and CDA

<table>
<thead>
<tr>
<th>Algorithm</th>
<th># Features</th>
<th># Evaluations</th>
<th>Average # Features</th>
<th>Balanced</th>
<th>Unbalanced</th>
</tr>
</thead>
<tbody>
<tr>
<td>NBM</td>
<td>80</td>
<td>76960</td>
<td>275.98</td>
<td>0 h 36 m</td>
<td>0 h 40 m</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>79380</td>
<td>269.48</td>
<td>0 h 37 m</td>
<td>0 h 11 m</td>
</tr>
<tr>
<td>SBE</td>
<td>150</td>
<td>67100</td>
<td>150</td>
<td>0 h 20 m</td>
<td>0 h 21 m</td>
</tr>
<tr>
<td></td>
<td>150</td>
<td>67100</td>
<td>150</td>
<td>0 h 50 m</td>
<td>0 h 11 m</td>
</tr>
<tr>
<td>CDA</td>
<td>80</td>
<td>67100</td>
<td>80</td>
<td>0 h 09 m</td>
<td>0 h 21 m</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>67100</td>
<td>40</td>
<td>0 h 05 m</td>
<td>0 h 11 m</td>
</tr>
<tr>
<td>SVM</td>
<td>150</td>
<td>68875</td>
<td>294.80</td>
<td>2 h 19 m</td>
<td>2 h 38 m</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>79380</td>
<td>269.48</td>
<td>2 h 20 m</td>
<td>2 h 54 m</td>
</tr>
<tr>
<td>CDA</td>
<td>80</td>
<td>67100</td>
<td>80</td>
<td>0 h 17 m</td>
<td>0 h 27 m</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>67100</td>
<td>40</td>
<td>0 h 14 m</td>
<td>0 h 19 m</td>
</tr>
<tr>
<td>SVM</td>
<td>150</td>
<td>13875</td>
<td>296.26</td>
<td>9 h 11 m</td>
<td>62 h 02 m</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>16020</td>
<td>271.03</td>
<td>9 h 48 m</td>
<td>63 h 40 m</td>
</tr>
<tr>
<td>SVM</td>
<td>150</td>
<td>13510</td>
<td>150</td>
<td>4 h 54 m</td>
<td>16 h 48 m</td>
</tr>
<tr>
<td>CDA</td>
<td>80</td>
<td>13510</td>
<td>80</td>
<td>2 h 48 m</td>
<td>9 h 38 m</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>13510</td>
<td>40</td>
<td>1 h 52 m</td>
<td>6 h 16 m</td>
</tr>
</tbody>
</table>

The first two columns indicate the algorithm and the size of the constrained subset. The third and fourth columns show the number of model evaluations that is needed and the average number of features that has to be evaluated. The last two columns show the running time (in hours and minutes) that is needed for a balanced holdout set (1000 positive and 1000 negative examples) and an unbalanced holdout set (263 positive and 20,886 negative examples).

3.4 Embedded techniques

Embedded feature selection techniques represent the third major class of feature selection algorithms. In these algorithms, feature selection is a part of the training process, or uses the parameters of the classification algorithm to select a relevant set of features. Just like wrapper methods, embedded approaches are thus specific to a given learning algorithm.

In this work, we will apply two embedded feature selection techniques: weighted naive Bayes (restricted to two-class classification problems with binary features) and weighted linear Support Vector Machines (WLSVM).

In the case of the decision tree C4.5, we note that feature selection is implicitly built in during tree construction. Two mechanisms account for this: the stop-splitting criterion and pruning. As explained in the previous chapter, splitting is stopped when no significant improvement is made, which can cause the elimination of irrelevant features. The other mechanism, pruning, removes subtrees and can thus also be seen as a form of feature elimination. Nevertheless, it is known that the performance of decision trees can still be improved using filter or wrapper techniques.

3.4.1 Weighted Naive Bayes Method

As explained before (see Equation 2.8), the Naive Bayes method assumes independence between the features. When we restrict ourselves to the basic
two-class classification problem with binary features, the Naive Bayes classifier can be transformed into a linear discriminant function. If we define the following notations:

\[ p_j = p(x_j = 1 | c_1) \quad \text{and} \quad q_j = p(x_j = 1 | c_2) \]

then we can write the class-conditional probabilities as

\[
p(x | c_1) = \prod_{j=1}^{n} p(x_j | c_1) = \prod_{j=1}^{n} p_j^{x_j} (1 - p_j)^{1 - x_j} \quad \text{(3.6)}
\]

\[
p(x | c_2) = \prod_{j=1}^{n} p(x_j | c_2) = \prod_{j=1}^{n} q_j^{x_j} (1 - q_j)^{1 - x_j} \quad \text{(3.7)}
\]

We then choose \( c_1 \) if \( p(c_1 | x) > p(c_2 | x) \) and \( c_2 \) otherwise. This decision can be rewritten as \( g(x) = p(c_1 | x) - p(c_2 | x) \), for which we decide \( c_1 \) if \( g(x) > 0 \) and \( c_2 \) otherwise.

An equivalent decision function is then [Duda et al., 2001]:

\[
g(x) = \ln \frac{p(x | c_1)}{p(x | c_2)} \quad \text{(3.8)}
\]

From Equations 3.6 and 3.7 we can then rewrite \( \frac{p(x | c_1)}{p(x | c_2)} \) as:

\[
\frac{p(x | c_1)}{p(x | c_2)} = \prod_{j=1}^{n} \left( \frac{p_j}{q_j} \right)^{x_j} \left( \frac{1 - p_j}{1 - q_j} \right)^{1-x_j}
\]

Equation 3.8 then becomes:

\[
g(x) = \sum_{j=1}^{n} [x_j \ln \frac{p_j}{q_j} + (1 - x_j) \ln \frac{1 - p_j}{1 - q_j}] + \ln \frac{p(c_1)}{p(c_2)} \quad \text{(3.9)}
\]

which is a linear discriminant function in \( x_i \) that can be written as:

\[
g(x) = \sum_{j=1}^{n} w_j x_j + b \quad \text{(3.10)}
\]

where

\[
w_j = \ln \frac{p_j(1 - p_j)}{q_j(1 - q_j)} \quad \forall \ j = 1, \ldots, n
\]

and

\[
b = \sum_{j=1}^{n} \ln \frac{1 - p_j}{1 - q_j} + \ln \frac{p(c_1)}{p(c_2)}
\]

From the formulation of Naive Bayes as a linear discriminant function, we can immediately derive the weights \( w_j \) for the \( j^{th} \) feature. The value of \( w_j \) determines how important feature \( j \) is in deciding the class, and the absolute values
Algorithm WLSVM

1. Initialise the set of features to the full feature set 
   \[ F = \{1, \ldots, n\} \]
2. Repeat until \( F = \{\} \)
   (a) Train the linear SVM with the features in \( F \)
   (b) Calculate the weight vector of dimension \( t = \text{len}(F) \)
   \[ w = \sum_{j=1}^{N_s} \alpha_j y_j x_j \]
   (c) Sort the \( t \) features according to the ranking: \( \text{score}_i = |w_i| \)
   (d) Remove the feature with the lowest \( \text{score}_i \)
   \[ F = F \setminus \text{argmin}(\text{score}_i) \]

Figure 3.7: Pseudo code for the WLSVM algorithm.

of these weights provide a feature weighting mechanism that can be used to
eliminate features with small (absolute valued) weights.
The advantage of this weighting mechanism in Naive Bayes (WNBM) is that
the probabilities and the features need only be calculated once. During the
feature elimination no retraining is needed, and one only needs to recalculate
\( p(x|c_i) \) taking into account the remaining features.

Two limitations are inherent to the weighted Naive Bayes method: it only
applies to two-class problems, and it can only be applied when all features are
binary. The first problem can be solved by converting a multi-class problem
into a set of two-class problems, as explained in the previous chapter. The
latter problem can be solved using some transformations.
Nominal features can be easily converted into binary format, using sparse vec-
tor encoding. In the case of playing tennis for example, the temperature values
\{hot, medium, cold\} can be converted to \{100,010,001\}, where the three-valued
feature temperature is converted into three binary features. For discrete-valued
features there are two possibilities: either we code them directly into binary
format, or we apply the same sparse vector encoding as in the case of nomi-
nal features. The case of continuous-valued features can be handled by first
discretizing them into discrete features, and then converting these into binary
format.

3.4.2 Weighted Linear Support Vector Machine

As the SVM is also a linear discriminant function, a similar approach as in the
case of WNBM can be followed. From Equation 2.16 we recall that the weight
vector for a linear SVM could be directly calculated from the support vectors
and the \( \alpha_i \) values. Again the weights can thus be used to rank and eliminate
features. However, in the case of SVM, retraining may be needed when a
particular feature has been eliminated. This approach has been tested by
3.5. TECHNIQUES BASED ON SIGNAL PROCESSING

Guyon et al. [Guyon et al., 2002], who found that better results were obtained when retraining the SVM at each iteration. The algorithm for the weighted linear SVM (WLSVM) is shown in Figure 3.7. Similar approaches are discussed in [Brank et al., 2002] and [Weston et al., 2003].

3.5 Techniques based on signal processing

When the training patterns can be represented in the form of signals, techniques based on signal processing can be used to select relevant features. Within the frame of feature selection techniques, we could classify these techniques as filter approaches, as they only take into account the intrinsic properties of the pattern (the signal), and are thus independent of the classification algorithm. Examples of these techniques are the Fourier transform and the wavelet transform. In this work, we will only discuss the Fourier transform.

3.5.1 Fourier transform

The continuous Fourier transform of a signal \( f(t) \) is defined as

\[
F(f) = \int_{-\infty}^{+\infty} f(t) e^{-i2\pi ft} dt
\]  

and its inverse is defined as

\[
f(t) = \int_{-\infty}^{+\infty} F(f) e^{i2\pi ft} df
\]  

However, when applied to real-world signals, computers are mostly used to do the calculations, and the discrete variant of the Fourier transform (DFT) is used. For the DFT, we assume that a finite signal \( f(t_k) = f_0, f_1, \cdots, f_{N-1} \) is periodically repeated to form an infinite signal. The DFT is then defined as

\[
F_n = \sum_{k=0}^{N-1} f_k e^{-i2\pi nk/N}
\]  

and its inverse is defined as

\[
f_k = \frac{1}{N} \sum_{n=0}^{N-1} F_n e^{i2\pi nk/N}
\]  

where the \( F_n \) are called the Fourier transform coefficients, which consist of a real and an imaginary part. Using Euler’s identity

\[
e^{i\theta} = \cos \theta + i \sin \theta
\]

we can rewrite the Fourier coefficients as

\[
F_n = \sum_{k=0}^{N-1} f_k \cos \left( \frac{-i2\pi nk}{N} \right) + \sum_{k=0}^{N-1} f_k i \sin \left( \frac{-i2\pi nk}{N} \right)
\]
or equivalent to this

\[ F_n = \sum_{k=0}^{N-1} f_k \cos \left( \frac{i2\pi nk}{N} \right) - i \sum_{k=0}^{N-1} f_k \sin \left( \frac{i2\pi nk}{N} \right) \] (3.16)

from which we can clearly see how the real and the imaginary part of the signal are constructed. If, for a given Fourier coefficient \( F_n \) we denote the real part as \( \text{Re}(F_n) = F_r^n \) and the imaginary part as \( \text{Imag}(F_n) = F_i^n \) then the magnitude and the phase of \( F_n \) are defined as

\[
\begin{align*}
\text{magnitude} &= \|F_n\| = \sqrt{F_r^n^2 + F_i^n^2} \quad (3.17) \\
\text{phase} &= \arctan \left( \frac{F_i^n}{F_r^n} \right) \quad (3.18)
\end{align*}
\]

Looking at the Fourier transform from a signal processing point of view, the transform maps the signal from its original domain (mostly time) to a frequency spectrum, decomposing the signal into a function of harmonics of different frequencies. As the basic functions \( \sin \) and \( \cos \) are periodic functions, the Fourier transform represents an easy way of detecting periodicities in the signal.

Under certain conditions, the result of the inverse DFT is the exact original signal. To this end, the sampling frequency used to obtain the discrete signal should be at least twice as large as the highest frequency in the signal. Taking a feature selection perspective, we can thus view the Fourier coefficients as weights, representing the importance or contribution of a specific frequency to the overall signal. In this respect we can compare irrelevant features to “noise”, and identify them as those frequencies having a low magnitude. Thus, feature selection can be achieved by setting a threshold \( \delta \) and removing all frequencies with a magnitude lower than \( \delta \). When only those frequencies (i.e. those \( F_n \)) are taken into account when computing the inverse DFT, the result will be a filtered version of the original signal. This technique can then be used to denoise or smooth a signal.

### 3.6 Feature weighting

One of the refinements within feature selection is the assignment of relevance values or weights to the features. Features with higher weights are then considered more relevant than features with lower weights. It has to be noted that in most cases the assignment of a weight only applies to individual features.

A number of previously mentioned techniques directly supports the weighting of features: filter techniques based on a relevance score, weighted Naive Bayes and the Fourier transform.

We will now discuss a new method for feature weighting, based on the idea of Estimation of Distribution Algorithms. This approach has the advantage that it can take into account feature dependencies (e.g. when combined with
3.6. FEATURE WEIGHTING

classifiers that model these dependencies), and is a direct extension of the
EDA-based wrapper approach.

3.6.1 A new feature weighting approach: EDA-based feature weighting

As mentioned earlier in this chapter, the standard approach to using random-
ized wrapper methods such as GAs and EDAs, is to select the best feature
subset encountered during 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 feature selection 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
interesting information, yet remain unanswered using a static analysis.

Feature ranking is a first step toward a dynamical analysis of the feature elim-
nation 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 perfor-
mance for each subset can be calculated, providing a dynamic view.

An even more detailed picture can be obtained using a feature weighting
method. Using feature weights, we can compare the relative importance of
individual features to the classification task.

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 com-
plete view of the selection process can be obtained. A similar methodology can
be applied in the case of most filter methods. As most filter methods calculate
a feature relevance score, we can use these scores directly as feature weights.
These weights can then be sorted, providing a feature ranking.

When using stochastic methods like GA or EDA, a hybrid approach could 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 infeasible, amount 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 developed an EDA-based feature weighting 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 calculating the feature weights.
As a consequence, this technique does not require any additional calculations
and, as all features are modelled in the estimated distribution, it provides a
3. FEATURE SELECTION TECHNIQUES

EDA-R

1. Select $S$ individuals from the final population $D_{\text{final}}$

2. Construct the probability model $P$ from $D_{\text{final}}$, $j \in 1, \cdots, S$
   using an EDA (e.g. UMDA, BMDA, BOA/EBNA)

3. For each variable (feature) $X_i$, calculate the probability $p(x^1_i)$

4. Sort features $X_1, \cdots, X_n$ by their $p(x^1_i)$ probabilities

5. Write out the array of sorted features

\textbf{Figure 3.8: General algorithm to calculate a feature ranking (EDA-R).}

dynamic view of the whole selection process.

\textbf{Deriving feature weights from an EDA}

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 $p(x^1_i)$ can be considered more important than a feature $j$ with a lower value for $p(x^1_j)$. The generalized probabilities $p(x^1_i)$ 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 (EDA-R) consists of the steps presented in Fig. 3.8 [Saeys et al., 2004d].

The most important step in this algorithm is the extraction of the probabilities $p(x^1_i)$ 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 $p(x^1_i)$ 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 [Henrion, 1988], yet the forward procedure of calculating the probabilities results in a unique probability distribution. For the example network of BOA and EBNA in figure 3.6, 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 $X_1, X_3, X_4, X_6, X_5, X_8, X_7, X_2$.

The advantage of using EDA-R as a feature weighting mechanism, compared
3.6. FEATURE WEIGHTING

to other feature weighting methods like filter methods, is that it can directly use the feedback (classification performance) of classifiers that allow 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 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.

**Necessary conditions**

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 should not have 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. 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

$$HD_s(x, y) = \frac{HD(x, y)}{N}$$  (3.19)

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

$$C(P) = \frac{2 \left( \sum_{i=1}^{S-1} \sum_{j=i+1}^{S} HD_s(x_i, x_j) \right)}{S \left( S - 1 \right)}$$  (3.20)

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.

To illustrate the use of $C(P)$ in determining a good feature ranking, we present a case study of a classification problem that will be studied more extensively in chapter 6: acceptor splice site prediction.
**Example 3.21**

The dataset for this problem consists of 400 binary features. 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 formula 3.20, and its evolution for population sizes of 500 and 1000 is shown in Fig. 3.9. 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. 3.10. 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 globally 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
3.6. FEATURE WEIGHTING

Figure 3.10: 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).

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.

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 then 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 will be included in the dataset, hoping that the relevant features are included. As a result, a lot of 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 hardly or not be present.

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. 3.11 shows the results for feature selection on both the training and test set. When comparing the results, it can be observed that the break-point regions for both data sets are very similar.

A final note concerns the automatic tuning of the convergence parameter $C(P)$. As explained before, we used in our experiments a threshold method, stopping
the iterative procedure when $C(P)$ falls below the threshold. A more advanced method can be thought of, stopping the iterative procedure automatically when the “optimal” convergence point has been reached. In order to define such a stopping criterion, we need a measure that defines the goodness of the population at a certain point during the iterative process. Looking again at the graphs in Figure 3.10, a possible measure to define this goodness is the area under the ranking curve.

Using this measure, we can compare the goodness of ranking at each iteration, where better feature rankings are defined as the ones having a bigger area under the ranking curve. In this way, the area under the ranking curve can be tracked during the iterative process, and a simple (greedy) heuristic can be used to decide whether the iterative process should be continued:

“As long as the area under the ranking curve increases, proceed to the next iteration. When the area under the ranking curve is less than the one of the previous iteration, stop the iterative process.”

In theory, this rule thus provides a way for the automatic detection of an “optimal” stopping point in the iterative procedure. However, in reality this rule brings on an additional computational burden: for each iteration, a ranking has to be calculated and evaluated. The evaluation of a ranking requires a lot of additional calculations, as it includes the training and testing of $N$ classifiers, $N$ being the number of features in the dataset.

### 3.7 Feature selection toolbox

As the primary goal of this work is the selection of relevant features for classifying nucleic acid sequences, some criteria have to be taken into account when choosing a good tool to experiment with. Inherent to genomic datasets like the
3.7. FEATURE SELECTION TOOLBOX

ones we experimented with are:

- the large number of features (high dimensionality)
- the large number of training patterns (large dataset sizes).

As a result, efficient methods to deal with such datasets when doing feature selection are needed. Currently, three programming environments to deal with such problems are available: MatLab, MLC++, and Weka. However, none of them really fits our needs:

- MatLab (Matrix Laboratory): although MatLab provides a wide variety of mathematical and engineering tools, the environment is not suited for large-scale experiments. The main reason for this is that the program stores everything in the form of matrices, and MatLab needs contiguous memory blocks to store these matrices. Therefore, MatLab is ideal as a prototyping environment, but for the amount of features and training data we are dealing with, it is inadequate.

- MLC++ (Machine Learning in C++): MLC++ is a library of machine learning algorithms, implemented in C++. This library was written in 1994 by Kohavi and others, yet its last update was in 1997. The library supports a good deal of the early machine learning algorithms and has a built-in wrapper based feature selection mechanism (only sequential selection).
  However, the method is very slow, as the included classifiers are not built-in in the package, but are invoked as standalone programs. As a result, for each feature subset to be evaluated, a datafile needs to be written, the classifier needs to be trained and tested, and the resulting output needs to be read in and parsed again. This makes it very impractical for the high-dimensional problems we are dealing with.

- WEKA (Waikato Environment for Knowledge Analysis): WEKA is a Java implementation of a large series of machine learning algorithms, clustering, pre-processing (including feature selection) and visualisation. WEKA has gained a lot of popularity during the last years, and has replaced MLC++ as main machine learning programming environment. However, the implementation in Java causes quite some overhead, and also has its implications on running time and memory use.

Some other peculiarities of the biological datasets used in this work motivated us to write our own feature selection toolbox. Especially the fact that features for nucleic acid sequences can be sparsely encoded (see later) yields a considerable improvement in memory management.

Figure 3.12 gives a global overview of the functionality of the toolbox. The feature selection techniques implemented in the toolbox are:

- Filter methods:
3. FEATURE SELECTION TECHNIQUES

Figure 3.12: Functional diagram for FeaSt: a Feature Selection Toolbox for large-scale applications.

- Koller-Sahami algorithm,
- future expansions we expect to add (shown in dotted line) are correlation based feature selection (CFS), and entropy based feature selection.

• Wrapper methods:
  - sequential backward elimination (SBE),
  - sequential forward selection (SFS),
3.7. FEATURE SELECTION TOOLBOX

- genetic algorithm (GA),
- Estimation of Distribution Algorithm (UMDA and BMDA),
- Constrained Estimation of Distribution Algorithm (CDA).

- Embedded methods:
  - weighted Naive Bayes method (WNBM),
  - weighted Linear Support Vector Machine (WLSVM).

- Weighting methods:
  - weighted Naive Bayes method (WNBM),
  - EDA-ranking (EDA-R).

3.7.1 Classifiers

The three selected classifiers (NBM, SVM, C4.5) were implemented in the toolbox. The Naive Bayes algorithm was implemented from scratch and can handle binary or discretized data. For now, only a two-class version of the algorithm was implemented.

For the SVM algorithm, we embedded the source code of SVM\textsuperscript{light} [Joachims, 1999] in the toolbox in the form of a library, and wrote an interface such that training data can be directly read in from memory instead of from a file. All functionality of SVM\textsuperscript{light} is thus preserved, including the use of polynomial and gaussian kernel functions. A similar approach was followed for the C4.5 decision tree, where the source code of the latest release (R8) was embedded in the toolbox. Again we made it more time and memory efficient by writing an interface that reads the data directly from memory.

For all these classifiers, IO operations were heavily reduced by loading training and test data only once at the startup of the toolbox. When different subsets of the features are then being generated by the different feature selection mechanisms, the appropriate subset of data can be directly read from memory and passed on to the classifier. This avoids two IO-operations: reading the data from file for every subset to be evaluated, and reading the results from the classification, the first of these operations being the most costly one.

3.7.2 Filters

In the basic version of the toolbox (the version we used in this work) the Koller-Sahami algorithm was chosen as a filter. The main reasons for choosing this algorithm are the fact that it has a sound theoretical basis, and that it is one of the few filter methods that takes into account feature dependencies. For now, the parameter $K$ of this algorithm, which determines the size of the Markov blanket, has been kept fixed to 1, as higher values would require too
3. FEATURE SELECTION TECHNIQUES

much computation time for our high-dimensional datasets. Two other filter methods are considered to be included as well: a baseline entropy method, and the correlation based feature selection (CFS).

3.7.3 Wrappers

Essentially two types of wrapper methods are included in the toolbox: sequential algorithms (SBE and SFS) and stochastic, population based methods (GA, EDA and CDA). Typical to these methods is that they perform a search through feature subset space, thus evaluating many different feature subsets before a step forward in the search is taken. In the case of sequential methods, such a step consists of greedily choosing the best subset of size \( n - 1 \) from \( n \) possible subsets. In the case of the stochastic methods, a step consists of going from one generation (iteration) to the next.

The fact that at each iteration a number of subsets needs to be evaluated independently, makes these algorithms suitable candidates for parallelization. Parallelizing these algorithms provides a linear speedup: using \( p \) processors, the feature selection algorithm can be sped up by a factor \( p \).

All wrappers in the toolbox were parallelized using a master-slave protocol, based on a message-passing interface (MPI). The version of MPI used in this toolbox is a portable platform version (Chameleon version, MPICH) available at http://www-unix.mcs.anl.gov/mpi/mpich.

For the EDA, two estimation algorithms were implemented in the toolbox: UMDA and BMDA. However, due to the high-dimensional feature sets, mostly UMDA was used. This was backed up by the fact that the experiments with BMDA did not lead to better or faster convergence, though required a lot more computing time.

3.7.4 Embedded methods

The embedded methods implemented in the toolbox are the “weighted” versions of the Naive Bayes classifier (WNBM) and the linear SVM (WLSVM). For WNBM, the embedded method can be easily built into the Naive Bayes framework, while for the WLSVM some more work had to be done to adapt the SVM\textsuperscript{light} code. As the WLSVM method needs to be retrained after each elimination of a feature, parallelization is not possible. For WNBM, no retraining is needed as soon as all weights and probabilities are calculated, so this algorithm could be implemented in parallel, but up to now this has not been considered really necessary.

3.7.5 Feature weighting methods

Currently two feature weighting methods are available in the toolbox: WNBM and EDA-R. The WNBM method is equivalent to the embedded version, as
3.7. FEATURE SELECTION TOOLBOX

all weights need only be calculated once. EDA based ranking (EDA-R) is applicable to any classifier, and the feature weights are determined by their frequencies in the final distribution. As it is an extension of the EDA, this method was also parallellized.

3.7.6 Discretization

For datasets with continuous-valued features, or mixed-valued features (e.g. some binary, some discrete, some continuous), some of the techniques in the toolbox are not directly applicable. These are the Koller-Sahami algorithm, which works only on discrete features, and all methods related to the Naive Bayes classifier. To support the use of such datasets, a discretization routine has been added to the toolbox.

A naive way of discretizing a continuous-valued attribute would simply be binning: define a number of bins \( b \) and divide the range of the variable into \( b \) intervals. Feature values within the same interval are then mapped to the same integer (e.g. going from zero to \( b - 1 \)). However, this may be very inefficient when some of the feature values are outliers, and arbitrary boundaries may be introduced, separating potentially related feature values. Therefore, we used a more intelligent discretization strategy, proposed in [Fayyad and Irani, 1993]. This method uses an entropy-based heuristic to recursively split the range of feature values into intervals. For a given range, the best cut point is defined as the one that gives the best information gain when splitting the data on this point. This procedure is then recursively applied until no further significant improvement can be made by defining a cut point.

3.7.7 Evaluation

Several measures of classification performance can be used when doing an evaluation of an algorithm, or when using the wrapper method when searching for a good subset. The measures currently supported are: F-measure, q9 ratio, accuracy and correlation coefficient.

3.7.8 Implementation

The whole toolbox was implemented in C++, and was optimised for large and high-dimensional datasets. As high-dimensional spaces are often sparsely populated, care has to be taken when designing the memory management to deal with such datasets. In the traditional toolboxes, the feature types are often encoded as real values, although the real types of the features are often discrete or binary. As many of the biological datasets that were used in this work were sparsely encoded (i.e. essentially binary), a more efficient means of storing them can be used.

Using unsigned integers and binary operators, features encoded in sparse vector format were directly stored in binary format. Thus, one sparse feature can be
encoded in one bit, whereas another system would probably encode it in a real
format like a double (32 bits), reducing the memory needs by a factor 32.
Similarly, discrete-valued features can be stored in binary format, requiring
less memory. For continuous-valued features, we followed the same strategy as
SVM\textsuperscript{light}, i.e. storing only those features that differ from zero.

3.8 Summary

In this chapter, we introduced feature selection techniques for classification.
We discussed the motivations to apply feature selection and gave an overview
of the different techniques that can be used. Furthermore we developed two
new methods. The first of these; the Constrained Estimation of Distribution
Algorithm (CDA) combined the efficiency of a wrapper technique with a search
directed toward small feature subsets. This resulted in a technique that per-
formed as good as, and sometimes better than the traditional approaches, while
heavily reducing the running time needed. A second new technique (EDA-
based feature ranking) focused more on the use of feature selection techniques
to gain insight in the processes that generated the data. We showed that this
method naturally extends the use of population-based search methods toward
obtaining feature weights and a feature ranking. In a last part of this chapter
we discussed our new software tool for feature selection in large scale datasets
(FeaST).
Chapter 4

Computational gene prediction

Thus, from the war of nature, from famine and death, the most exalted object which we are capable of conceiving, namely, the production of the higher animals, directly follows. There is grandeur in this view of life, with its several powers, having been originally breathed into a few forms or into one; and that, whilst this planet has gone cycling on according to the fixed law of gravity, from so simple a beginning endless forms most beautiful and most wonderful have been, and are being evolved.

- Charles R. Darwin

4.1 Introduction

Gene structure prediction is an important, yet complex task in bioinformatics. A reliable gene prediction forms the basis of many applications in functional, structural and comparative genomics. To understand the nature of this task, a “gradient descent” into the molecular mechanisms behind transcription and translation seems appropriate.

The central actors during the life of an organism are proteins. Proteins determine how an organism is structured, as well as how its behaviour is defined. The genetic constituents that code for proteins are genes. When a certain protein is needed, the gene that codes for that protein will be expressed and the appropriate protein will be synthesized.

This brings us to the central dogma of molecular genetics, characterizing the synthesis of proteins as a two-step decoding process:
4.1. INTRODUCTION

Figure 4.1: Transcription and translation in eukaryotes.

- DNA codes for RNA = transcription
- RNA codes for proteins = translation
- DNA is the informational molecule which specifies the structure of the proteins using the RNA as intermediate
- RNA transmits the genetic information in the DNA from the nucleus to the cytoplasm, where it is translated into a protein.

The mechanisms behind gene expression depend on the complexity of the organism being studied. Two types of organisms can be distinguished: prokaryotes (lower organisms where the cells only contain cytoplasm) and eukaryotes (higher organisms where the cells contain a nucleus). In this work, we focus our attention to the case of eukaryotes.

To know when and where to be expressed, each gene is preceded by a promoter region. The promoter region can be seen as a gene’s identity card. When the appropriate proteins bind to this region, the start sign is given to produce the protein. This process is illustrated in Figure 4.1.

When the start of the gene (promoter region) is recognized, a copy of the rest of the gene is made. This step is called the transcription, and produces a single stranded copy of the gene: the pre-messenger RNA (pre-mRNA). In eukaryotes, this transcript contains both non-coding and coding regions. The non-coding intervening regions are termed introns, while the expressed (coding) regions are called exons. After transcription, the introns have to be spliced out precisely, producing the mature mRNA. The mRNA will then be transported from the nucleus to the cell’s cytoplasm, where it is translated into a protein. To this end, a special marker on the mRNA is looked for: the start-codon. From this position on, the mRNA is translated until another marker (stop-codon) is encountered.
4. COMPUTATIONAL GENE PREDICTION

Figure 4.2: Translation of the mRNA sequence into a protein.

<table>
<thead>
<tr>
<th>5</th>
<th>U</th>
<th>C</th>
<th>A</th>
<th>G</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>U</td>
<td>Phe</td>
<td>Ser</td>
<td>Tyr</td>
<td>Cys</td>
<td>U</td>
</tr>
<tr>
<td></td>
<td>Phe</td>
<td>Ser</td>
<td>Tyr</td>
<td>Cys</td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>Leu</td>
<td>Ser</td>
<td>stop</td>
<td>stop</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>Leu</td>
<td>Ser</td>
<td>stop</td>
<td>stop</td>
<td>G</td>
</tr>
<tr>
<td>C</td>
<td>Leu</td>
<td>Pro</td>
<td>His</td>
<td>Arg</td>
<td>U</td>
</tr>
<tr>
<td></td>
<td>Leu</td>
<td>Pro</td>
<td>His</td>
<td>Arg</td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>Leu</td>
<td>Pro</td>
<td>Gin</td>
<td>Arg</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>Leu</td>
<td>Pro</td>
<td>Gin</td>
<td>Arg</td>
<td>G</td>
</tr>
<tr>
<td>A</td>
<td>Ile</td>
<td>Thr</td>
<td>Asn</td>
<td>Ser</td>
<td>U</td>
</tr>
<tr>
<td></td>
<td>Ile</td>
<td>Thr</td>
<td>Asn</td>
<td>Ser</td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>Ile</td>
<td>Thr</td>
<td>Lys</td>
<td>Arg</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>Met</td>
<td>Thr</td>
<td>Lys</td>
<td>Arg</td>
<td>G</td>
</tr>
<tr>
<td>G</td>
<td>Val</td>
<td>Ala</td>
<td>Asp</td>
<td>Gly</td>
<td>U</td>
</tr>
<tr>
<td></td>
<td>Val</td>
<td>Ala</td>
<td>Asp</td>
<td>Gly</td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>Val</td>
<td>Ala</td>
<td>Glu</td>
<td>Gly</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>Val</td>
<td>Ala</td>
<td>Glu</td>
<td>Gly</td>
<td>G</td>
</tr>
</tbody>
</table>

Figure 4.3: The codon table: translation from triplets into amino acids.
4.2 Gene Prediction

The computational gene prediction problem is the problem of the automatic annotation of the location, structure, and functional class of protein-coding
4. COMPUTATIONAL GENE PREDICTION

genes [Fickett, 1996]. A correct annotation forms the basis of many subsequent modelling steps, and thus should be done with great care. Driven by the explosion of genome data, computational approaches to identify genes have thus proliferated.

Current methods of gene prediction mostly focus onto the problem of identifying the coding and non-coding parts of the sequence, often leaving the detection of the promoter region aside. Two approaches can be distinguished: a) the prediction of genes by aligning them with genes already known (extrinsic prediction), and b) the use of classification models based on features extracted from the sequence (intrinsic prediction). While finding genes using the extrinsic approach is reasonably fast and accurate, it only facilitates the detection of about 50% of the genes [Matthé et al., 2002]. The reason for this is that a homology-based search will not find the genes that do not resemble those in the databases. A second explanation is that it remains difficult to find genes that are rarely expressed (e.g. if they are very specific to some tissue), or if they are expressed at a low level. Although the databases have grown fast, those genes are still under-represented in the databases. For the remainder of the genes, intrinsic prediction methods need to be used. In this work, we will only consider intrinsic gene prediction methods, as these are the methods we can combine with feature selection techniques.

The prediction of the complete structure of a gene can be described as a two-step process. In a first step, the various structural elements are predicted: boundaries between introns and exons (these are the so called splice sites), branch point, start- and stop-codon, and very rarely promoter elements such as transcription factor binding sites and TATA-box. In a second step, all of these predictions are combined into an overall, consistent gene structure. Consistency implies that genes should begin with a start codon and end in a stop codon, do not have an intermediate stop codon, have no overlapping exons, are reading-frame compatible, and have an acceptable length (usually compared to the length distribution of genes in the training set).

4.2.1 Intrinsic gene prediction

Whereas extrinsic gene prediction methods use homology information to detect gene structures similar to those in databases, intrinsic methods use pattern recognition techniques to predict functional elements in genes. The information sources or features that are used by these methods are generally grouped into two classes, according to the locality of the information they represent: content sensors and signal sensors.

Content sensors can be viewed as global measures that try to capture a general tendency of the sequence. These measures are used to define if a sequence is a coding region (exon), a non coding region (intron) or an intergenic region. Signal sensors can be viewed as more local features, that try to detect the pres-
4.2. GENE PREDICTION

ence of the functional sites specific to a gene, such as splice sites, start-codon and stop-codon. Although both types of information are essentially different, many of the techniques to detect them are very similar [Degroeve et al., 2004].

Weight Matrix method

The most simple generative method to classify sequences according to their type is the Naive Bayes classifier, which - in gene prediction literature - is often referred to as the Weight Matrix Method (WMM, [Staden, 1984]). To classify a sequence $S = s_1 s_2 \ldots s_n$ into one of the classes $\{c_1, c_2\}$ the posterior probabilities $p(c_1|S)$ and $p(c_2|S)$ are calculated and the log-odds ratio is used as a prediction for $S$:

$$\text{pred}(S) = \ln \frac{p(c_1|S)}{p(c_2|S)}$$

In a WMM, the probabilities $P(S|c)$ needed to calculate $P(c|S)$ (see 2.8) are simplified as

$$P(S|c) = \prod_{i=1}^{n} p(s_i|c)$$

where $p(s_i|c)$ denotes the relative frequency of the nucleotide $s_i$ at position $i$. WMMs are mostly used for predicting signal elements. In Genscan [Burge and Karlin, 1997] they are used to model promoter elements (TATA-box and Cap) and start codons.

Weight Array method

A natural extension of the WMM is to include information that captures dependencies between adjacent nucleotides, motivating the use of Markov Chains. In a Markov Chain, a collection of random variables has the property that, given the present, the future is conditionally independent of the past. When applied to DNA sequences, where the element of time is transformed to the notion of position in the sequence, this is equivalent to saying that the probability of observing $s_i$ is only dependent on the nucleotide $s_{i-1}$ immediately preceding it. Thus, the probabilities $P(S|c)$ are modelled as:

$$P(S|c) = P(s_1|c) \prod_{i=2}^{n} P(s_i|s_{i-1}, c)$$

where $P(s_i|s_{i-1}, c)$ is estimated from the conditional frequency of observing $s_i$ at position $i$ given the occurrence of $s_{i-1}$ at position $i - 1$. In gene prediction, this model was first described by Zhang and Marr [Zhang and Marr, 1993] and is commonly referred to as the Weight Array Method (WAM). WAMs are used to detect acceptor sites in Genscan, and splice sites in GeneID [Guigo et al., 1992].

Naturally, the method can be used to include even more dependencies. In the case where a position $i$ is dependent on the $k$ previous positions, the Markov
property of independence of the past can be relaxed, and the Markov Chain is called a Markov Chain of order \( k \), or \( k \)th order Markov model. For a \( k \)th order Markov model, the probabilities \( p(S|c) \) are modelled as:

\[
p(S|c) = p(s_1|c) \cdots p(s_k|s_1, \ldots, s_{k-1}, c) \prod_{i=k+1}^{n} p(s_i|s_{i-k}, \ldots, s_{i-1}, c)
\]

When these Markov models are used as a signal sensor, the transition probabilities between two states (in the case of sequences each position is a state) do not vary over time, as mostly a fixed-length context is used to score a site. In this case, the Markov model is termed a homogeneous Markov model. In the Glimmer gene prediction program such a second order Markov model is used to detect splice sites [Salzberg et al., 1998].

However, when used as a content sensor, a variation on these Markov models, called three-periodic Markov models are often used. This means that, for each of the three possible reading frames, a separate Markov model is constructed, which allows the immediate detection of the reading frames in coding regions. The model then consists of three states, corresponding to each of the three possible positions in a codon, with transition probabilities from state 1 to 2, 2 to 3 and 3 to 1, hence it is periodic. In this case, the transition probabilities between states may vary over time, and the model is termed a non homogeneous Markov model. 5th order Markov models are used in Genscan and GeneMark [Borodovsky and McIninch, 1993], where the homogeneous version is used to model introns (non-coding sequences) and the three-periodic, non-homogeneous models are used to model exons (coding sequences).

If higher order dependencies exist, then modelling these should result in better prediction methods, as they model the underlying sequence in a more realistic way. However, to model DNA, the number of parameters of the model (the \( k \)-mer probabilities) increases exponentially with the order \( k \) of the Markov model: \( O(4^{k+1}) \). As a result, the number of training instances needed to reliably estimate each model parameter also grows exponentially with \( k \).

To cope with the problem of reliably estimating model parameters, a number of refinements to the basic \( k \)th order Markov model have been proposed. A direct extension of the WAM is proposed by Burge in Genscan, using a windowed WAM model (WWAM). In this method, the probabilities \( p(s_i|s_{i-k}, \ldots, s_{i-1}, c) \) for position \( i \) are averaged over the positions \( i-\delta, i-(\delta-1), \ldots, i, i+1, \ldots, i+\delta \), typically for small values of \( \delta \). As a result, the data available for estimating a parameter increases by a factor \( 2\delta + 1 \). This model is used in Genscan to detect branch point regions.

**Interpolated Markov model**

A more common way to account for a limited number of training examples is the Interpolated Markov Model (IMM). This method was first described in the Glimmer gene predictor and combines probabilities from contexts of
4.2. GENE PREDICTION

varying lengths to make predictions. It then uses only those contexts for which sufficient data is available. For a given order $k$, this is done by estimating the probabilities as weighted linear combinations of the parameters $p(s_i|c), p(s_i|s_{i-1},c), \ldots, p(s_i|s_{i-k}, s_{i-1}, c)$. The weights of the linear combination are computed as a combination of two parameters: the $\chi^2$ significance and the frequency of occurrence. As a result, higher order interactions will only be modelled if there is enough training data and if including them provides a significant difference compared to using only the lower order combinations. This improvement allows the use of higher order Markov models compared to the WAMs, which makes IMMs a good technique to model coding or non-coding sequences. In GlimmerM [Salzberg et al., 1999] and EuGene [Schiex et al., 2001] a three-periodic $8^{th}$ order IMM is used to model exons, and normal IMMs are used to model introns and intergenic regions (EuGene).

Maximal dependence decomposition

Although the use of Markov Chains to model gene sequences is biologically motivated, a limitation of these approaches is that they only consider dependencies between adjacent nucleotides. However, nucleotide studies by Burge [Burge and Karlin, 1997] and Vignal [Vignal et al., 1999] have shown that a large number of dependencies exist between non-adjacent nucleotides as well. To model these dependencies, the Maximal Dependence Decomposition (MDD) method was developed by Burge et. al. This method is a probabilistic binary decision tree, based on the $\chi^2$ statistic. At each node in the tree the position with the highest dependence on the remaining positions is placed. Based on the value of this position, the dataset is split into the set of sequences that have the consensus nucleotide at this position, and those who have not. The process is then recursively repeated until no further subdivision is possible, no significant dependencies are detected, or the number of sequences remaining in the subset has become too small. Finally, separate WMM models are derived for each leave node in the tree.

Classification of a sequence is then obtained by following the appropriate path in the tree, until a leaf node is reached. The WMM associated with that leaf is then applied to make a prediction. The MDD is used in Genscan to predict donor sites, and is used in combination with a second order Markov model in GeneSplicer [Pertea et al., 2001] to detect splice sites.

Interpolated context model

A similar approach is explored in the Glimmer2 system [Delcher et al., 1999]. Their method, the Interpolated Context Model (ICM) is a generalisation of the IMM, that considers dependencies between nucleotides (not necessarily adjacent) in a local context $C = b_1 b_2 \ldots b_k$ of length $k$ and nucleotide $b_{k+1}$. For all $k + 1$-mers in the dataset, the mutual information (MI) is then calculated between each position $i = 1 \ldots k$ and position $k + 1$. The position $i$ with the
maximal MI is then chosen as the root split of the tree: depending on the nucleotide at position $i$ the training set of $k+1$-mers is split into four subsets.

For each of these subsets the same procedure can be applied recursively, until the tree reaches a pre-determined depth, or the size of the set of windows becomes too small to reliably estimate probabilities. Each leaf of the tree is labelled with the probability distribution of the nucleotide at position $k+1$, given the nucleotide values along the path from the root to that leaf. In the example of Figure 4.5, the leaf shaded gray would be labelled with the distribution

$$p(X_{k+1} = x | X_7 = A, X_{10} = A, X_4 = T)$$

Thus, the ICM generalises the notion of the IMM.

**Discriminative models**

The methods discussed so far all belong to the class of generative classification models, as these are the methods that are used in most gene prediction systems. A few other methods use discriminative classification techniques to predict functional sites and coding potential. Multi-Layered Perceptrons (MLP) were used to identify splice sites in NetPlantGene [Hebsgaard et al., 1996], NetGene2 [Tolstrup et al., 1994] and Genie [Reese et al., 1997]. The Linear Discriminant Analysis (LDA) method is used to detect splice sites in the collection of predictors in Gene-Finder [Solovyev and Salomov, 1997, Solovyev et al., 1995], while Quadratic Discriminant Analysis (QDA) is used in MZEF [Zhang, 1997]. More recent methods also use Support Vector Machines, e.g. to predict splice sites [Sonnenburg, 2002, Degroeve et al., 2002, Saeys et al., 2003].
4.2. GENE PREDICTION

When the different functional elements of a gene have been detected, an integrative approach is needed to come up with a consistent structure of a gene. Efficiently finding such a structure provides quite a challenge from a computational point of view, as the number of potential gene models grows exponentially with the number of predicted exons.

The problem of finding the most likely parsing for an observed sequence is very similar to the problems encountered in speech processing. As a result, methods based on Hidden Markov Models (HMM) are frequently used to find gene structures, as these models have a long standing tradition in speech processing [Rabiner, 1989]. An overview of the application of HMMs to gene prediction can be found in [Krogh, 1998]. In most cases the different content and signal sensors are just embedded into a HMM framework, like in GeneMark.hmm [Lukashin and Borodovsky, 1998]. An example of the state space of a HMM for gene prediction in Genscan is shown in Figure 4.6. In this model, each circle or diamond represents a functional unit (state) of a gene or a genomic region: N (intergenic region), P (promoter), F and T (5' extending from the start of transcription up to the translation initiation signal, 3' extending from just after the stop codon to the polyadenylation signal), \( E_{\text{sgn}} \) (single exon gene, from start to stop codon), \( E_{\text{init}} \) (initial exon, from start codon to donor splice

**Figure 4.6:** State space for the HMM used in Genscan [Burge and Karlin, 1997].

4.2.2 Complete gene structure prediction

When the different functional elements of a gene have been detected, an integrative approach is needed to come up with a consistent structure of a gene. Efficiently finding such a structure provides quite a challenge from a computational point of view, as the number of potential gene models grows exponentially with the number of predicted exons.

The problem of finding the most likely parsing for an observed sequence is very similar to the problems encountered in speech processing. As a result, methods based on Hidden Markov Models (HMM) are frequently used to find gene structures, as these models have a long standing tradition in speech processing [Rabiner, 1989]. An overview of the application of HMMs to gene prediction can be found in [Krogh, 1998]. In most cases the different content and signal sensors are just embedded into a HMM framework, like in GeneMark.hmm [Lukashin and Borodovsky, 1998]. An example of the state space of a HMM for gene prediction in Genscan is shown in Figure 4.6. In this model, each circle or diamond represents a functional unit (state) of a gene or a genomic region: N (intergenic region), P (promoter), F and T (5' extending from the start of transcription up to the translation initiation signal, 3' extending from just after the stop codon to the polyadenylation signal), \( E_{\text{sgn}} \) (single exon gene, from start to stop codon), \( E_{\text{init}} \) (initial exon, from start codon to donor splice

74
4. COMPUTATIONAL GENE PREDICTION

site) and $E_{\text{term}}$ (terminal exon, from acceptor splice site to stop codon), $E_k$ ($0 \leq k \leq 2$, phase $k$ internal exon, from acceptor splice site to donor splice site), $I_k$ ($0 \leq k \leq 2$, phase $k$ internal intron, from just after donor splice site to just before branch point/acceptor splice site), and $A$ (polyadenylation signal).

In most of the gene predictors, including Genscan, the employed model is a variation on the standard HMM, called a generalised HMM (GHMM). A GHMM is a HMM where the states are arbitrary sub-models (e.g. neural networks, higher order Markov models, etc.). To find the path that maximises the likelihood given the sequence, an optimisation technique called Dynamic Programming (DP, [Bellman, 1957]) is used. Different algorithms, such as the “forward” and “backward” algorithms [Rabiner, 1989] or the Viterbi algorithm [Viterbi, 1967] can then be used to solve the problem efficiently. A more detailed overview of gene prediction methods can be found in [Fickett, 1996, Mathé et al., 2002, Zhang, 2002].

4.2.3 Feature selection

As large amounts of annotated genomic sequences become available, opportunities are opened to allow the use of supervised learning methods to automate the task of sequence annotation. Apart from using these methods for mere classification of unknown sequences, they can be used to identify patterns in the data at hand, and hence provide useful domain knowledge.

To be able to build a model for a specific biological task (e.g. splice site prediction), supervised learning methods need information sources or features from which the models can be computed. However, as it is not yet completely known how many and which features are relevant for the different subtasks of gene prediction, the initial set of features is chosen large enough, increasing the probability that the relevant information is in the set.

The motivation for combining gene structure prediction with feature selection techniques consists of two parts. The first part concerns the purely computational aspects: a reduced set of features, allowing a better and faster classification of the data. As increasingly more genomes have been sequenced, these aspects are of key importance for high-throughput genome annotation projects. The second part focuses more on the gain of insight into the biological processes related to transcription and translation.

4.3 The Arabidopsis dataset: some statistics

For our experiments, we used the data available for the plant model species Arabidopsis thaliana. The Arabidopsis thaliana data set was generated by aligning mRNAs (with SIM4 [Florea et al., 1998]) obtained from the public EMBL database (June 5th 2000), with the BAC-sequences that were used for the Arabidopsis chromosome assembly. For future evaluation purposes, we excluded
4.3. **THE ARABIDOPSIS DATASET: SOME STATISTICS**

![Intron length distribution](image1)

![Exon length distributions](image2)

**Figure 4.7:** Distributions of the length for introns (upper part) and exons (lower part) in *Arabidopsis thaliana.*
4. COMPUTATIONAL GENE PREDICTION

all genes that where in AraSet [Pavy et al., 1999]. Redundant genes were ex-
cluded by applying algorithm 2 of [Hobohm et al., 1992]. This method counts
the neighbours (two genes are neighbours when they show more than 80% iden-
tity at the nucleotide level) of every gene, discards the gene with the largest
number of neighbours and repeats this process until no genes with neighbours
remain. Of the 1812 genes obtained from EMBL (Aubourg et al., unpublished),
1495 genes were kept after removing redundant ones. Furthermore, the genes
were carefully checked for frameshifts and start and stop codons, resulting in
a high-quality dataset for genome annotation. We refer to this dataset as the
PlantGene dataset.

To get a quick view of the length distributions of introns and exons, a statistical
analysis of these regions was conducted. Introns and exons were partitioned
in six categories: intron/exon in 5’ UTR, initial intron/exon (if more than one
intron/exon), single intron/exon (the gene consists of only one intron/exon),
internal intron/exon, terminating intron/exon and introns/exons in 3’ UTR.
Figure 4.7 shows the distributions for introns (upper part) and exons (lower
part). For introns, it is observed that the distribution for introns in 5’ UTR
is more or less uniform between 80 and 600 nucleotides, differing considerably
from the other types of intron distributions, where a clear majority of the in-
trons has a length between 80 and 120 base pairs. For exons, the distribution
of single exon genes differs most from the other types of exons. A more detailed
statistical study of the Arabidopsis genome, including statistics on codon usage
and composition of the different genomic regions can be found in [Mathe, 2000].

In addition to such a more content based analysis, simple techniques can be
used to quickly investigate the local context surrounding functional sites. The
sequence logo [Schneider and Stephens, 1990] is an easy way to visualise bi-
ological sequences. From an alignment of sequences, a logo is created that
displays the frequencies of bases at each position as the relative heights of let-
ters. The degree of sequence conservation is displayed as the total height of
a stack of letters, measured in bits of information (entropy). Subtle frequen-
cies are not lost in the final product as they would be in a consensus sequence.
The vertical scale is in bits, with a maximum of 2 bits possible at each position.

Sequence logos are created for a number of aligned functional sites: Transcrip-
tion Start Sites (TSS), Translation Initiation Sites (TIS), donor splice sites
and acceptor splice sites. These are shown in Figure 4.8. The figures clearly
show the conserved sequences around the sites for TIS, donor splice site and
acceptor splice site. In these cases, one part of the context represents coding
information, the other part is non coding. Clearly, more conservation is present
in the non coding part of the local context. In the case of TSS, a clear signal
containing a lot of A and T nucleotides is observed around 30 base pairs up-
stream (left) of the translation start site. This signal is named the TATA-box
and is an essential element of the core promoter.
4.4. SUMMARY

In the next chapters, we will study the different subtasks involved in gene prediction in more detail, taking a feature selection tour to analyse the processes in greater detail. We start with the content problem of distinguishing coding sequences from non-coding sequences. Then we move on to the prediction of signal sensors; the major contribution consists in the analysis of feature selection techniques for splice site prediction. Furthermore, we will also discuss the application of FSS techniques to the detection of translation initiation sites. For each of these problems, the PlantGene dataset will be used to investigate the results for *Arabidopsis*, although all methods are generally applicable to other species.

4.4 Summary

In this chapter, we introduced the gene prediction problem. The chapter started by introducing the reader to the biological processes involved in gene transcription and translation. We then proceeded by a more algorithmical exposition of the various techniques that are nowadays commonly used in gene prediction, focusing on intrinsic gene prediction techniques. We also motivated the use of feature selection techniques in the context of the gene prediction problem. We concluded the chapter by discussing the data set that will be used during our experiments, providing a glimpse of what is awaiting in the following chapters.
4. COMPUTATIONAL GENE PREDICTION

![A. thaliana Transcription start site](image)

![A. thaliana Translation initiation site](image)

![A. thaliana Donor site](image)

![A. thaliana Acceptor site](image)

**Figure 4.8:** Sequence logos for functional sites in *Arabidopsis thaliana*: transcription start sites, translation initiation start sites, donor and acceptor splice sites.
Chapter 5

Feature selection for coding potential prediction

I know of scarcely anything so apt to impress the imagination as the wonderful form of cosmic order expressed by the “Law of Frequency of Error”. The law would have been personified by the Greeks and deified, if they had known of it. It reigns with serenity and in complete self-effacement, amidst the wildest confusion. The huger the mob, and the greater the apparent anarchy, the more perfect is its sway. It is the supreme law of Unreason. Whenever a large sample of chaotic elements are taken in hand and marshaled in the order of their magnitude, an unsuspected and most beautiful form of regularity proves to have been latent all along.

- Francis Galton

5.1 Introduction

An important problem in the context of gene prediction, is the distinction between sequences that code for proteins and non coding sequences. The tendency of a sequence to code for a protein is called its coding potential, and the prediction of coding potential is an essential component in every gene predictor. On the one hand it is used to score potential exons, while on the other hand it is used as an additional measure to predict functional sites. An example of the latter application is the fact that splice sites occur at the boundaries between introns and exons, and thus the coding potential at both sides of the splice site should differ. This enables the classification model to separate them from pseudo sites that occur within coding or non coding regions, where the coding potential on both sides is similar.
5.2. FOURIER ANALYSIS OF CODING SEQUENCES

To be able to distinguish coding from non-coding sequences, there should be an essential difference that allows us to discriminate between them. This difference is the underlying codon structure of coding sequences, inherent to the fact that nucleotides are translated into amino acids in triplets.

A first bias already exists within the translation from nucleotide triplets into amino acids (see Figure 4.3), as twenty amino acids are built from 64 triplets, and not all of the amino acids are equally represented.

Besides the unequal distribution of amino acids in terms of codons, another bias affects the characteristics of coding regions: codon bias. This bias results from the fact that codons are not equally used throughout the coding parts of a genome. Furthermore, the codon bias may even be different for genes of the same genome [Mathe et al., 1999].

To capture these peculiarities from coding sequences, a large number of protein coding measures have been developed. These were reviewed by Fickett and Tung in 1992 [Fickett and Tung, 1992], where the authors suggest to use only a few of them, like in-phase hexamer counts and the Fourier measure. The use of hexamer counts is tightly related to the use of fifth-order Markov models, as explained in the previous chapter. However, in this chapter we explore the other approach: the use of Fourier coefficients to distinguish coding from non-coding sequences. The Fourier measure differs from the $k$-mer count approach in that it measures the global periodical characteristics of the sequence, whereas the $k$-mer approach only looks for dependencies in a local window of $k$ nucleotides.

In the rest of this chapter, we discuss the application of the Fourier transform to nucleic acid sequences, and in particular its use in identifying coding sequences. Furthermore we explore the use of feature selection in combination with these Fourier features.

5.2 Fourier analysis of coding sequences

A well-known characteristic when applying Fourier analysis to DNA coding sequences is the observation of a peak at frequency $1/3$ in the Fourier spectrum [Silverman and Linsker, 1986, Voss, 1992]. The peak at this frequency is a direct result from the fact that coding sequences consist of codons, combined with the fact that the codons are not equally used. As a result, this peak is a recognition of the boundary between codons, rather than a recognition of an exact repeat of a triplet. The latter would not only lead to a peak at frequency$^{1}$ $1/3$, but also to peaks at its harmonics which are integer multiples of the frequency $1/3$.

The most common way to apply Fourier analysis to DNA sequences is to de-

$^{1}$To be notationally correct, we should clarify that the unity in this frequency is one in three nucleotides. However, we will just use the abbreviation $1/3$ during the rest of the text.
compose them first into four binary indicator sequences [Silverman and Linsker, 1986, Tiwari et al., 1997], apply the Fourier transform to each of these sequences, and then sum the Fourier coefficients. Following the notation of Voss, a binary indicator sequence is obtained by using a projection operator $U_\alpha$ which selects the elements of the sequence that are equal to the symbol $\alpha$, namely $U_\alpha(x_j) = 1$ if $x_j = \alpha$ and 0 otherwise. Using the operators $U_A$, $U_T$, $U_C$ and $U_G$ then results in four binary sequences.

Example 5.1
5.2. FOURIER ANALYSIS OF CODING SEQUENCES

For each of the indicator sequences, we can then calculate the magnitudes of the Fourier coefficients $\|F^\alpha_n\|$ for each $\alpha \in \{A,T,C,G\}$, and the sum of these magnitudes represents a global measure of periodicity for the given sequence:

$$\sum_{\alpha} \|F^\alpha_n\|$$

To investigate the effect of the sequence length on the Fourier spectrum, we compared the spectra of biased sequences to unbiased sequences for various lengths. This was done as follows. First we constructed a codon usage matrix, making use of all the full-length coding sequences (CDS) in the PlantGene dataset. Using this matrix, sequences with the PlantGene codon bias were generated. On the other hand, unbiased (random) sequences were generated. Figure 5.1 shows the results of this comparison. For each graph, the x-axis denotes the sequence length $n$, $n = 1 \ldots \frac{N}{2}$, where $N$ is the total sequence length, and the frequency can be calculated as $\frac{n}{N}$. The y-axis represents the sum of the magnitudes over all nucleotides. For each type of sequence (coding and non coding), the spectra were averaged over 100 sequences, and the magnitudes of the Fourier coefficients were plotted. The peak at frequency $1/3$ is apparent in the coding sequences while absent in the non coding sequences.

In a feature selection perspective, the magnitude at frequency $1/3$ is highly discriminative between coding and non coding sequences, which can be simply discovered by calculating the difference in peak height between coding and non coding spectra.
Similar experiments can be done for functional sites that characterize transitions between coding and non-coding sequences. In the gene prediction framework, the most important ones are the translation initiation site (TIS, start-codon) and the splice sites (donor and acceptor sites). The case of TIS is shown in Figure 5.2, where a symmetrical context of 120 bases around the site was used to calculate the spectra. The spectra were averaged over 500 start sites, showing a clear peak at frequency $1/3 = 40/120$ for the downstream (coding) part of the context. No such peak is apparent in the upstream part of the site, which is known to be the 5' untranslated region (UTR).

Similar observations can be made for splice sites. Figure 5.3 compares different context lengths for the case of acceptor prediction. While for length 120, only a peak at $1/3$ is observed downstream of the splice site, clearly a similar signal emerges when going to longer sequence lengths (150, 180). This can be explained by the fact that the upstream part of the acceptor site is located in an intron, and the majority of the introns is shorter than 120 nucleotides (see

Figure 5.3: Fourier spectra upstream (left, non-coding) and downstream (right, coding) of the acceptor site.
5.3. COMBINING FOURIER FEATURES AND FEATURE SELECTION

Figure 4.7). As a consequence, enlarging the context to 150 or 180 nucleotides will include on average more coding sequences from the exon that precedes the intron. Similar observations can be made for the case of donor sites.

In the next section, we will use the Fourier coefficients as a measure to discriminate between coding and non-coding sequences. Furthermore, we will combine the use of different features based on the Fourier transform with feature selection techniques.

5.3 Combining Fourier features and feature selection

Fourier-based features are used in a number of approaches to discriminate between coding and non-coding sequences. In [Tiwari et al., 1997] the signal-to-noise ratio (SNR) of the peak at frequency $1/3$ is used as a measure for gene prediction. A similar path is explored in [Anastassiou, 2000] where the Fourier coefficients of the indicator sequences at frequency $1/3$ are taken together in a linear combination to define a decision threshold between coding and non-coding sequences.

Here, we extend this approach by defining a set of 19 features, derived from the Fourier coefficients at frequency $1/3$. These are:

- The four real parts of the Fourier coefficient for each of the indicator sequences: $A_{\text{real}}, T_{\text{real}}, C_{\text{real}}$ and $G_{\text{real}}$
- The four imaginary parts of the Fourier coefficient for each of the indicator sequences: $A_{\text{imag}}, T_{\text{imag}}, C_{\text{imag}}$ and $G_{\text{imag}}$
- The four magnitudes (absolute values) of the Fourier coefficients: $A_{\text{abs}}, T_{\text{abs}}, C_{\text{abs}}$ and $G_{\text{abs}}$
- The global magnitude: $\text{Sum}_{\text{abs}} = \sum \alpha_{\text{abs}}$
- The four phases for each indicator sequence: $A_{\text{phase}}, T_{\text{phase}}, C_{\text{phase}}$ and $G_{\text{phase}}$
- The sum of the phases: $\text{Sum}_{\text{phase}} = \sum \alpha_{\text{phase}}$
- The signal to noise ratio of the peak at frequency $1/3$:

$$\text{SNR} = \frac{\text{Sum}_{\text{abs}}}{\overline{S}}$$

where $\overline{S}$ denotes the average of the total spectrum:

$$\overline{S} = \frac{1}{N}(1 + \frac{1}{N} - \sum \rho_{\alpha}^2)$$
5. FEATURE SELECTION FOR CODING POTENTIAL PREDICTION

with \( N \) the sequence length and \( \rho_\alpha \) the frequency of occurrence of nucleotide \( \alpha \) [Tiwari et al., 1997].

In our experimental set-up, we focused on detecting coding potential for short sequences: datasets were constructed for sequences with lengths of 120 and 60 nucleotides. As explained in the previous chapter, models to detect coding regions often are split up in three different parts, one for each reading frame.

In our experiments, different datasets for each reading frame were constructed: positive examples were extracted in the appropriate reading frame from complete coding sequences (CDS), and negative examples were extracted from introns and intergenic regions. Sequences were converted to the same reading direction, resulting in three datasets, one for each reading frame. During the rest of this chapter, we will discuss the results for reading frame 1. Similar results can be obtained for the other reading frames.

For each of the sequence lengths (60 and 120 nucleotides), a dataset was constructed containing 8000 positive and 8000 negative examples. The dataset was split into a training and test set, each containing 4000 positives and 4000 negatives. A two-fold cross validation of the dataset was replicated five times, resulting in ten (5x2) possible train-test combinations.

For each of these datasets, the 19 Fourier features as described earlier were extracted. Different combinations of feature selection techniques and classification algorithms were then evaluated on these datasets. The Naive Bayes Method (NBM) and the C4.5 decision tree were combined with EDA based ranking (EDA-R) and the Koller-Sahami algorithm (KS). For these methods, the datasets were discretized using the method of Fayyad and Irani [Fayyad and Irani, 1993]. The internal evaluation of subsets in the EDA-R technique was obtained by doing a 5-fold cross validation of the training set. For the linear Support Vector Machine, the embedded method WLSVM was used. In this case, the real-valued datasets were not discretized, but were scaled to the interval \([0,1]\). The C parameter of the linear SVM was tuned to 0.8 for sequences of length 60, and to 0.7 for sequences of length 120.

Figure 5.4 shows the results of the feature selection. For each of both sequence lengths (60, 120) the average classification performance (F-measure, y-axis) is plotted versus the number of features (x-axis). At the origin, the results represent the evaluation on the full feature set (0 features eliminated). At the other end of the graph, only one feature remains. In between, features are iteratively discarded.

A first observation that can be made is that, in general, better results are obtained for the longer sequences. This result is not surprising, as the probability of detecting the codon bias increases with sequence length (see also Figure 5.1). In terms of Fourier coefficients, this will result in a better detection of the peak at frequency 1/3.
5.3. COMBINING FOURIER FEATURES AND FEATURE SELECTION

Figure 5.4: Evaluation of the feature selection methods for Fourier features of frequency 1/3. At the x-axis, the number of features eliminated thus far is shown. At the y-axis the average classification performance (F-measure) is shown.

A second observation is that more than half of the features can be eliminated before the classification performance drastically drops down. Using only five features (14 features eliminated), the results are still comparable to the results...
5. FEATURE SELECTION FOR CODING POTENTIAL PREDICTION

<table>
<thead>
<tr>
<th>Rank</th>
<th>WLSVM</th>
<th>EDA-R NBM</th>
<th>EDA-R C4.5</th>
<th>KS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SNR</td>
<td></td>
<td></td>
<td>SNR</td>
</tr>
<tr>
<td>2</td>
<td>Sum_abs</td>
<td>T_real</td>
<td></td>
<td>SNR</td>
</tr>
<tr>
<td>3</td>
<td>G_real</td>
<td></td>
<td>T_real</td>
<td>G_phase</td>
</tr>
<tr>
<td>4</td>
<td>T_real</td>
<td></td>
<td>G_real</td>
<td>Sum_abs</td>
</tr>
<tr>
<td>5</td>
<td>T_abs</td>
<td></td>
<td>G_phase</td>
<td>T_real</td>
</tr>
</tbody>
</table>

Table 5.1: Ranking of the top 5 features for each of the feature selection schemes for a sequence length of 60 nucleotides.

<table>
<thead>
<tr>
<th>Rank</th>
<th>WLSVM</th>
<th>EDA-R NBM</th>
<th>EDA-R C4.5</th>
<th>KS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SNR</td>
<td>SNR</td>
<td></td>
<td>SNR</td>
</tr>
<tr>
<td>2</td>
<td>Sum_abs</td>
<td>T_phase</td>
<td>Sum_phase</td>
<td>G_phase</td>
</tr>
<tr>
<td>3</td>
<td>G_real</td>
<td>G_real</td>
<td>Sum_abs</td>
<td>G_real</td>
</tr>
<tr>
<td>4</td>
<td>T_real</td>
<td>T_real</td>
<td>G_real</td>
<td>T_real</td>
</tr>
<tr>
<td>5</td>
<td>T_abs</td>
<td>A_phase</td>
<td>A_real</td>
<td>Sum_abs</td>
</tr>
</tbody>
</table>

Table 5.2: Ranking of the top 5 features for each of the feature selection schemes for a sequence length of 120 nucleotides.

Table 5.1 and 5.2 show the top 5 features selected by the different feature selection algorithms, respectively for length 60 and 120. Clearly the signal-to-noise ratio (SNR) of the peak at frequency 1/3 is the most discriminative feature. However, combining this feature with a number of other features significantly enhances classification, as was demonstrated by the feature selection graphs.

Averaging the results over all feature selection methods, the following four features are found as most relevant: SNR, G\_real, T\_real, and Sum\_abs. Apparently, the frequencies for the indicator sequences of the nucleotides G and T are thus more informative than the frequencies of the nucleotides A and C, together with the total magnitude of the peak at frequency 1/3 and the SNR.

5.4 Extending the feature set: including all frequencies

Although the peak at frequency 1/3 is a clear discriminator between coding and non-coding sequences, there might be other useful information in a Fourier spectrum apart from the “visible” information regarding frequency 1/3. In an attempt to investigate whether other periodic information might be extracted by the classification method, we extended our feature set to include the infor-
5.4. EXTENDING THE FEATURE SET: INCLUDING ALL FREQUENCIES

Figure 5.5: Evaluation of the feature selection methods for Fourier features of all frequencies. At the x-axis, the number of features eliminated thus far is shown. At the y-axis the average classification performance (F-measure) is shown.

Information for all Fourier coefficients $F_n$, $n = 0 \ldots \frac{N}{2}$ where $N$ is the sequence length.
5. FEATURE SELECTION FOR CODING POTENTIAL PREDICTION

For the dataset with sequences of length 60 this results in a set of 589 (19x31) features. For sequences of length 120, datasets with 1159 (19x61) features were created. We then applied the same feature selection techniques as in the previous section. For the linear SVM, the C parameter was tuned to 0.05 for sequences of length 60, and to 0.01 for sequences of length 120. The results of the feature selection are shown in Figure 5.5.

A general conclusion that can be drawn, is that all methods benefit from the extension of the feature set. For sequences of length 60, the NBM benefits the most: an increase in F-measure from 0.76 to 0.92. For the linear SVM and C4.5, the increase amounts to about 10%. For sequences of length 120, the gain in F-measure is about 10% for all classifiers.

Furthermore, the significant increase in the number of features has some consequences on the feature selection algorithms. When comparing the Koller-Sahami method to EDA-R for Naive Bayes and C4.5, significantly better results are obtained using EDA-R. For sequences of length 120, the NBM even outperforms the SVM.

Another interesting result is the fact that C4.5 profits most from feature selection, achieving the best results with very small feature sets. This results from the fact that C4.5 is known to overfit the data on the training set, and feature selection can thus help to remove this bias.

A last, yet remarkable fact to note is the strange behaviour of WLSVM on sequences of length 120 in the range between 250 and 650 eliminated features. As these curves are averages over the 10 crossvalidation folds, a careful analysis of each of these curves showed that this behaviour was found in 4 out of the 10 folds. The other 6 curves were smooth like the ones for all other classifiers. The fact that for some of the folds worse results were obtained for some of the feature sets in the range between 250 and 650 eliminated features is probably related to the way the C parameter of the SVM is tuned. This parameter is tuned beforehand on the full feature set using a crossvalidation procedure. However, for some feature subsets, this parameter may not be optimal, and can lead to worse results.

A solution to this would be to retune the C parameter after the removal of a feature, but for bigger datasets like the ones we are dealing with, this quickly turns out to be computationally infeasible because tuning of the C-parameter requires, for every feature removal, the retraining and testing of the classifier for a whole range of values for C.

Looking at the feature selection graphs, it is clear that the majority of the features can be eliminated, without degrading the classification performance. This points to the existence of a small subset of relevant features. To find out which features these are, we take a look at the features selected by one of the best scoring techniques: EDA-R. This technique has the advantage that an implicit feature weighting can be calculated from the final distribution (see section 3.6.1).
5.4. EXTENDING THE FEATURE SET: INCLUDING ALL FREQUENCIES

An appropriate way to visualise these weights is by color coding them onto a “heat” map, where blue denotes the least important, and red denotes the most important features. This is shown in Figure 5.6. The figure shows the result of the feature weighting using EDA-R combined with the Naive Bayes classifier. The upper part of the figure shows the features for sequences of length 60, the lower part the ones of length 120. Each row represents one of

Figure 5.6: Visualisation of the feature weights resulting from an EDA-R feature selection with the Naive Bayes classifier. More important features tend toward red, less important ones to blue.
the 19 selected Fourier features, each column represents the coefficients \( k \) (the frequency can then be calculated as \( \frac{k}{N} \)).

Clearly, the gain in classification performance stems almost entirely from the inclusion of the features pertaining to the coefficient 0. These coefficients have a special meaning, and represent the composition of the sequence (i.e. the number of A’s, T’s, C’s and G’s). Clearly such compositional information is essential for detecting coding potential, and boosts the classification performance.

Other interesting patterns that can be observed are the importance of the feature \( C_{\text{abs}} \), not only at frequency 1/3, but also at neighbouring frequencies. Furthermore, the phase features at frequency 1/3 seem to be very important, especially for the nucleotides T and G. The inclusion of Fourier phase features has not been exploited by techniques for coding potential prediction up to now. Our results clearly show that they can contribute significantly to a better classification.

As such, the application of feature selection techniques nicely illustrates how biologically important knowledge can be derived from a classification model.

5.5 A comparison with an Interpolated Markov Model

Interpolated Markov Models (IMM, [Salzberg et al., 1998]) are state-of-the-art models for coding potential prediction. At current, up to 8th order dependencies between adjacent nucleotides are taken into account when discriminating between coding and non-coding regions. IMMs are incorporated in the microbial gene-finder Glimmer [Salzberg et al., 1999], and its eukaryotic counterpart GlimmerM [Pertea and Salzberg, 2002], as well as in the gene predictor EuGène [Schiex et al., 2001].

While the IMM is known to perform well for coding potential prediction for long sequences (at least several hundreds of nucleotides), little is known about its performance on shorter sequences. In this work, we compare the results obtained with the Fourier features to the results of an IMM. The IMM was trained on the same datasets (with lengths 60 and 120), and the same 5x2 cross validation was applied. For the IMM, we used the source code, publicly available at the TIGR website (The Institute for Genomic Research, http://www.tigr.org).

For the classification methods using the Fourier features, we compared the two best classifiers (NBM and linear SVM) to the IMM. For these models, we used three types of datasets:

1. A simple dataset containing only the 19 Fourier features of frequency 1/3,

2. a complex dataset containing the Fourier features of all frequencies,
5.5. A COMPARISON WITH AN INTERPOLATED MARKOV MODEL

Figure 5.7: ROC curves comparing the classification performance of the classifiers. Results for sequences of length 60 are shown in the top part, results for sequences of length 120 in the bottom part.

3. a reduced dataset, containing the 38 features, pertaining to the frequencies 0 and 1/3, inspired by the feature selection outcome.
5. FEATURE SELECTION FOR CODING POTENTIAL PREDICTION

For dataset 3, the C-parameter of the SVM was tuned to 5. For each of the classification algorithms, ROC curves were calculated by varying the decision threshold, the results of which are shown in Figure 5.7.

For each of the sequence lengths (60,120) the ROC plots show the true positive rate (TP-rate) versus the false positive rate (FP-rate). Better classifiers are situated more toward the top left corner. On each graph the classifiers are sorted in the legend, according to their performance. The best classifier is listed on top, the worst one on the bottom.

In general, NBM and LSVM using only the 19 features of frequency 1/3 perform worst. The reduced set of 38 features is shown to perform comparable or better than the set containing all frequencies. Furthermore, the reduced set of 38 features clearly outperforms the IMM. While the difference is relatively small for sequences of length 120, the distinction becomes more apparent when going to shorter sequences of length 60. This points to the fact that using the Fourier features might be more advantageous than using an IMM to detect coding potential for short sequences.

To explain the superiority of the Fourier features for short sequences, the global character of the Fourier transform is of great importance. While the IMM only looks at the sequence by way of a short, moving window of 9 nucleotides, the Fourier transform looks at the occurrence of a particular frequency in the whole sequence. Doing this, it takes into account dependencies between nucleotides at equidistant positions, depending on the particular frequency.

5.6 Summary

In this chapter we investigated the use of features, derived from the Fourier transform, for the problem of coding potential prediction. We started out with a simple set of 19 features, belonging to the frequency 1/3, this frequency being a clear indication of coding potential in the Fourier spectrum.

We extended this approach to the inclusion of Fourier features for all frequencies, investigating if any useful information could be derived from these. Using feature selection techniques, we showed that the major increase in classification performance resulted from the inclusion of the Fourier features of frequency 0. While this cannot be considered a “true” frequency, it proved that an important contribution to coding potential prediction could be made by including compositional information.

To conclude, we performed a comparative analysis of the different classifiers based on the Fourier features. We also compared the results to a state-of-the-art coding potential predictor, the Interpolated Markov Model. Our results show that using a selected subset of relevant Fourier features, better predictions could be made than using the IMM. While the difference was somewhat small for sequences of length 120, it increased when going to shorter sequences.
5.6. SUMMARY

of length 60, proving the value of Fourier features for coding potential prediction for short sequences.
Chapter 6

Feature selection for splice site prediction

We have good news and bad news.
The good news is that by the end of this decade, we may know most, if not all, of the transcription factors active in many cell types and how they interact to initiate or repress transcription.

The bad news is that many of us will have to learn physical chemistry to understand these data.

- Scott Gilbert, 1997

6.1 Introduction

The DNA sequences of most genes are coding for messenger RNA (mRNA) themselves encoding proteins. While in lower prokaryotes the mRNA is a mere copy of a fragment of the DNA, in eukaryotes the DNA contains non-coding segments in genes (introns) which should be precisely spliced out to produce the mRNA. The splice sites we refer to here are the border sides of such introns. The splice site in the upstream part of the intron is called the donor site, the other site is termed the acceptor site.

Donors are characterized by a conserved “GU” dinucleotide in the intron part, acceptors have a characteristic “AG” dinucleotide in the intron part. However, as these dinucleotides occur very frequently throughout the genome, only a very small percentage of them will be true splice sites. As a consequence, the local context around the putative acceptor site has to be taken into account when deciding if a site is either a true or a pseudo splice site.
Thus, splice site prediction can be divided into two subtasks: prediction of donor sites and prediction of acceptor sites. Each of these subtasks can be formally stated as a two-class classification task: \{donor site, non-donor site\} and \{acceptor site, non-acceptor site\}.

In biology, two main forms of splicing exist: self-splicing in RNA’s and spliceosomal splicing [Gesteland and Atkins, 1993]. Within the context of gene prediction we turn our attention to spliceosomal splicing. For this type of splicing, the splicing process is accomplished by a large ribonucleoprotein complex, the spliceosome, which binds to the pre-mRNA and removes the introns. Introns are excised in a two-step cleavage-ligation reaction, where the first step involves cleavage at the 5’ splice site (donor site) with formation of an intron lariat at an adenosine nucleotide (the branch point), usually 18-40 nucleotides upstream of the 3’ splice site (acceptor site). In a second phase, following cleavage at the 3’ splice site, the exons are ligated, and the intron is released as a lariat, which is then debranched and degraded. This process is illustrated in Figure 6.1.

The excision of intron sequences from the pre-mRNA by the spliceosome is a dynamic yet orderly process that involves many small nuclear ribonucleoprotein particles (snRNPs) and numerous other proteins. To be able to build a model for splicing, a more detailed look at the processes involved in splicing is needed.

The most abundant form of spliceosomal splicing is the case where donors are characterized by a conserved GU sequence and acceptors by a conserved AG
sequence in the intron. These processes are illustrated in Figure 6.2. Splicing begins with the U1 snRNP being recruited to the donor site, where the 5' end of U1 snRNA interacts with the donor site through base pairing (step 1). Around the same time, the U2 snRNP binds to a region defined as the branch point region, upstream of the acceptor site (step 2). Subsequently the U4/U6 and U5 snRNPs bind (step 3), and then U1 and U4 are displaced (step 4). In the next step (5) U6 switches to bind to U2, bringing the donor and acceptor site into close proximity and forming a lasso-structure (lariat). U6 then acts
6.1. INTRODUCTION

Although some characteristics of splicing seem well known, other still remain mysterious. An example of this is the selection of the acceptor site. Following the scanning model defined in [Smith et al., 1989], the spliceosome scans from the branch point and usually selects the first AG dinucleotide downstream of the branch point. However, other experiments suggest that the acceptor site may be recognized early in the splicing process, and then aids the identification of the internal branch point sequence [Liu et al., 1995].

Another concern, pointing to the importance of splicing in the overall process of gene expression and its regulation, is the phenomenon of alternative splicing. In alternative or differential splicing, a single pre-mRNA sequence can be spliced in various ways, producing different proteins (Figure 6.3). It has been shown that alternative exons possess weaker splice sites than the normal exons [Stamm et al., 1994], suggesting the use of additional signals in the sequence that might enhance their recognition (exonic splicing enhancers or ESEs, [Fairbrother et al., 2002]). Further it is becoming clear that splicing of a particular intron depends on a fine balance in the “strength” of the multiple signals involved in splice site selection.

Whereas alternative splicing in plants is rather exceptional, the most common estimate nowadays indicates that 60% of all human genes have alternatively spliced variants, and this number may increase as the size of EST databases grow. The reason for this is that differentiation in animals often requires alternative splicing to produce cell-type specific mRNAs from a single transcript. The selection of alternative splice sites then produces proteins with different activities. An extreme example of (combinatorial) alternative splicing is the axonic guidance gene Dscam in Drosophila melanogaster which can be

Figure 6.3: Alternative splicing.
alternatively spliced - in theory - into over 38,000 different mRNA isoforms [Graveley, 2001].

The methods that we developed during this work support the detection of alternative transcripts, as every potential donor and acceptor site will be analysed by the classification method. As a result, alternate forms of donors and acceptors can be detected by the system. It is then the task of the global gene prediction system to decide what to do with this information. Either this information is discarded by the prediction system, e.g. when only one transcript can be predicted, or alternative sites can be taken into account to predict more than one transcript.

The rest of this chapter discusses each of the splice site prediction problems: acceptor prediction and donor prediction. For each of the problems, we investigate how feature selection techniques can be used to improve the model, and how we can gain more insight into the data using them.

6.2 Acceptor prediction: an iterative feature selection approach

As already discussed in the introduction, many biological processes are still far from being understood. This greatly influences the design of the features that will be used by a classification method that tries to model this process. In this section we describe an iterative feature construction and feature selection approach, resulting in increasingly more complex features and datasets, the design of which is guided by the feature selection techniques [Saeys et al., 2004c]. 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 of the acceptor site. Therefore, the nucleotides A,T,C and G occurring on either side of the acceptor constitute a basic feature set. Figure 6.4 exemplifies the use of a local context for the case of an acceptor site.

![Figure 6.4: Local context for feature extraction for acceptor sites.](image-url)
6.2. ACCEPTOR PREDICTION: AN ITERATIVE FEATURE SELECTION APPROACH

6.2.1 A simple dataset: position dependent nucleotides

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 sparse vector encoding (A=1000, T=0100, C=0010, G=0001). This results in a dataset described by 400 binary features. For this dataset, the filter method by Koller and Sahami (KS) and EDA based feature ranking (EDA-R) constitute the two basic feature selection techniques that were combined with all classifiers (NBM, C4.5 and SVM). In addition, the two embedded methods WNBM and WLSVM were included in the experiments. For EDA-R the distribution size and the number of iterations were tuned to 500 and 20 respectively. For the SVM, the C-parameter was tuned to 0.05, using the full feature set.

The dataset consisted of 6000 positive and 36,000 negative examples. From this set, training and test sets were compiled with equal class imbalance (1 positive versus 6 negative examples). This was done by five resamplings of the dataset, each time splitting it up in a training and test set containing 3000 positive and 18,000 negative examples. As a result, five replications of a two-fold crossvalidation were obtained.

A comparative evaluation of the results of the experiments is shown in Table 6.1. The classifier/selection approach combinations are tabulated row wise, and the results on different feature subsets are shown in the columns. Apart from the results using the full feature set (100%), the results using only 50%, 25%, 10% and 5% of the features are shown. The numbers represent the average F-measure over the 10 cross-validation folds, and the standard deviation. For each of the reduced feature sets, the result was compared to the results on the full feature set, using the combined 5x2 cv F test. Statistically significant improvements at confidence levels of 0.9 and 0.99 were denoted respectively by $^*$ and $^{**}$. Statistically equivalent results compared to the full feature set were denoted by $^\dagger$ and statistically worse results were not marked.

On a global scale, the NBM and C4.5 benefit most from feature selection. For NBM, the best results are obtained with EDA-R, while for C4.5 the KS filter method scores better. Note that the embedded method WNBM performs considerably worse than the other methods, especially when many features are eliminated. For the linear SVM, no significant gain in classification performance could be obtained. This can be explained by the fact that the SVM already implicitly uses a feature weighting scheme. For the SVM, the embedded method WLSVM achieves the best results overall, followed by EDA-R and KS.

For knowledge discovery, the only method in our experiments that is able to derive feature weights is the EDA-R method. As the SVM method scores best
## 6. FEATURE SELECTION FOR SPLICE SITE PREDICTION

### Simple dataset: 400 features

<table>
<thead>
<tr>
<th>Method</th>
<th>100%</th>
<th>50%</th>
<th>25%</th>
<th>10%</th>
<th>5%</th>
</tr>
</thead>
<tbody>
<tr>
<td>KS NBM</td>
<td>80.87 ± 0.31</td>
<td>80.85 ± 0.37</td>
<td>78.77 ± 0.45</td>
<td>74.67 ± 0.79</td>
<td>72.14 ± 0.70</td>
</tr>
<tr>
<td>EDA-R NBM</td>
<td>80.87 ± 0.31</td>
<td>82.32 ± 0.32</td>
<td>80.65 ± 0.37</td>
<td>76.70 ± 0.73</td>
<td>69.49 ± 2.46</td>
</tr>
<tr>
<td>WNBM</td>
<td>80.87 ± 0.31</td>
<td>80.80 ± 0.42</td>
<td>76.84 ± 0.41</td>
<td>67.52 ± 0.52</td>
<td>60.39 ± 1.73</td>
</tr>
<tr>
<td>KS C4.5</td>
<td>55.25 ± 0.61</td>
<td>57.52 ± 0.53</td>
<td>59.52 ± 0.75</td>
<td>60.97 ± 0.88</td>
<td>62.16 ± 0.62</td>
</tr>
<tr>
<td>EDA-R C4.5</td>
<td>55.25 ± 0.61</td>
<td>56.67 ± 0.58</td>
<td>57.27 ± 0.46</td>
<td>59.32 ± 0.84</td>
<td>61.58 ± 0.98</td>
</tr>
<tr>
<td>KS LSVM</td>
<td>84.45 ± 0.30</td>
<td>82.75 ± 0.28</td>
<td>80.66 ± 0.49</td>
<td>75.05 ± 0.42</td>
<td>71.00 ± 0.49</td>
</tr>
<tr>
<td>EDA-R LSVM</td>
<td>84.45 ± 0.30</td>
<td>84.17 ± 0.38</td>
<td>81.62 ± 0.46</td>
<td>76.32 ± 0.67</td>
<td>68.73 ± 2.09</td>
</tr>
<tr>
<td>WLSVM</td>
<td>84.45 ± 0.30</td>
<td>84.00 ± 0.40</td>
<td>81.87 ± 0.39</td>
<td>76.73 ± 0.43</td>
<td>71.23 ± 0.40</td>
</tr>
</tbody>
</table>

**Table 6.1:** Acceptor prediction: F test comparisons for the dataset of 400 features.

### Extended dataset: 528 features

<table>
<thead>
<tr>
<th>Method</th>
<th>100%</th>
<th>50%</th>
<th>25%</th>
<th>10%</th>
<th>5%</th>
</tr>
</thead>
<tbody>
<tr>
<td>KS NBM</td>
<td>78.21 ± 0.50</td>
<td>78.40 ± 0.50</td>
<td>77.96 ± 0.64</td>
<td>77.26 ± 0.46</td>
<td>74.21 ± 0.58</td>
</tr>
<tr>
<td>EDA-R NBM</td>
<td>78.21 ± 0.50</td>
<td>84.48 ± 0.30</td>
<td>83.52 ± 0.36</td>
<td>80.79 ± 0.57</td>
<td>75.93 ± 0.87</td>
</tr>
<tr>
<td>WNBM</td>
<td>78.21 ± 0.50</td>
<td>77.17 ± 0.51</td>
<td>77.85 ± 0.37</td>
<td>74.06 ± 0.32</td>
<td>67.96 ± 0.68</td>
</tr>
<tr>
<td>KS C4.5</td>
<td>61.86 ± 0.56</td>
<td>62.85 ± 0.73</td>
<td>65.27 ± 0.71</td>
<td>67.36 ± 0.65</td>
<td>68.25 ± 0.69</td>
</tr>
<tr>
<td>EDA-R C4.5</td>
<td>61.86 ± 0.56</td>
<td>62.87 ± 1.03</td>
<td>63.68 ± 0.73</td>
<td>65.35 ± 0.71</td>
<td>67.23 ± 0.92</td>
</tr>
<tr>
<td>KS LSVM</td>
<td>87.52 ± 0.49</td>
<td>87.15 ± 0.32</td>
<td>86.03 ± 0.41</td>
<td>82.05 ± 0.46</td>
<td>77.03 ± 0.80</td>
</tr>
<tr>
<td>EDA-R LSVM</td>
<td>87.52 ± 0.49</td>
<td>86.72 ± 0.54</td>
<td>85.64 ± 0.50</td>
<td>82.34 ± 0.43</td>
<td>77.02 ± 1.10</td>
</tr>
<tr>
<td>WLSVM</td>
<td>87.52 ± 0.49</td>
<td>87.20 ± 0.49</td>
<td>86.40 ± 0.50</td>
<td>84.35 ± 0.48</td>
<td>78.34 ± 0.92</td>
</tr>
</tbody>
</table>

**Table 6.2:** Acceptor prediction: F test comparisons for the dataset of 528 features.
on this dataset, we can thus use the derived weights of the EDA-R LSVM combination to visualize the relevance of the features. This is shown in Figure 6.5. For each of the nucleotides (rows), the nucleotide positions (columns) are shown for both parts of the local context, the acceptor site being in the middle. Several patterns can be observed. The nucleotides bordering the acceptor site are of primary importance, representing binding information. Furthermore the nucleotides T in the left part of the context are highly important, representing the well-known pyrimidine-stretch (an excess of nucleotides T and C). A last pattern that can be observed is the three-base periodicity in the right part of the context, especially for nucleotides T and G, capturing the fact that this part of the sequence is the coding part (exon), and that nucleotides in this part are organized in triplets.

6.2.2 Adding position invariant features

Position dependent features are not the best solutions when trying to capture information like coding potential and composition in the sequence. Therefore, a second dataset was constructed as an extension of the dataset in the previous experiment. In addition to the position dependent nucleotide features, we also added a number of position invariant features. These features capture the occurrence of 3-mers (words of length 3) in the sequence flanking the acceptor. An example of such a feature is the occurrence of the word “GAG” in the left part of the context. This results in another 128 binary features (64 for each part of the context), a 1 decoding the presence, a 0 the absence of the specific word in the context. Together with the position dependent features, this yields a dataset consisting of 528 binary features. The same parameters for EDA-R and SVM were used as with the previous dataset.

The results of the feature selection experiments on this dataset are shown in Table 6.2. Comparing the results for NBM to the previous dataset, the classification performance on the full feature set is lower than on the first dataset. However, using feature selection, better classification results than on the first dataset can be obtained. Again, the best results for NBM were obtained with the EDA-R wrapper method. Using only 10% of the features, this method still obtains significantly better results than using the full feature set. The KS filter method performs second best, WNBM performs worst.

For C4.5, adding position invariant features boosts the classification performance with about 6%. Using feature selection techniques even improves significantly on the results using the full feature set. Similar to the previous dataset, C4.5 achieves the best results using only 5% of the data, improving classification performance by about 6% compared to using the full feature set.

For the SVM, a significant gain of 3% in classification performance was obtained by adding the position invariant features. Similar to the previous dataset, the performance could not be improved using feature selection methods, and the embedded method WLSVM obtains the best results.
Figure 6.5: EDA based feature weighting for acceptor prediction (400 features). The EDA-R approach was combined with the linear SVM.

Figure 6.6: EDA based feature weighting for acceptor prediction (528 features). The EDA-R approach was combined with the linear SVM.
6.2. ACCEPTOR PREDICTION: AN ITERATIVE FEATURE SELECTION APPROACH

The visualization of the feature weights, obtained by the EDA-R LSVM approach, is shown in Figure 6.6. While the same patterns as in the previous dataset can be observed, it is clear that some information is translated from position dependent features into position invariant features. An example of this is the pyrimidine stretch, which is somewhat shortened, together with the fact that T-rich 3-mers in the left part of the context show up as very important. Another example is the fact that the 3-base periodicity on the coding side is less pronounced, yet some 3-mers are shown to be highly relevant. The results from the feature weighting, combined with the improved classification results explain that indeed position invariant features contribute to the prediction of acceptor sites.

6.2.3 Adding more complex features: dependencies between adjacent nucleotides

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, like C4.5 or SVMs with polynomial kernels. However, these methods have the disadvantage of being quite slow to train, rendering the feature selection process more computationally intensive. Here, we describe another approach to deal with nucleotide dependencies, having the advantage that linear (and thus fast) classification models can be used. We do this by constructing more complex features, capturing the nucleotide dependencies at the feature level. Another important advantage then is that the combination with feature selection techniques allows us to select those dependencies that are of primary importance, and visualize them.

To this end, we created complex features that capture dependencies between two adjacent nucleotides. These features are represented as position dependent 2-mers (words of length 2). At each position \(i\) of the local context, these features represent the word appearing at position \(i\) and \(i + 1\). This results in a set of 1568 binary features \(49 \times 16 \times 2\). Together with the position dependent nucleotides and the position invariant features, this results in a dataset described by 2096 features. For this dataset, the C-parameter of the SVM was tuned to 0.005.

The results of the feature selection experiments for this dataset are shown in Table 6.3. Compared to the results on the previous datasets, similar trends can be observed. The NBM classifier performs worse than with dataset 1 on the full feature set, but outperforms the results on dataset 1 and 2, when EDA-R is used with only 50% of the features.

For C4.5, the best results are again obtained using only 5% of the features, resulting in an increase of about 7% compared to using the full feature set. EDA-R and KS perform comparably on this dataset, yet compared to dataset 1 and 2, no increase in classification performance could be obtained.
### Table 6.3: Acceptance prediction: F test comparisons for the dataset of 2096 features.

<table>
<thead>
<tr>
<th>Method</th>
<th>100%</th>
<th>50%</th>
<th>25%</th>
<th>10%</th>
<th>5%</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>KS NBM</strong></td>
<td>79.21 ± 0.33</td>
<td>79.46 ± 0.30</td>
<td>79.08 ± 0.39</td>
<td>79.07 ± 0.57</td>
<td>78.03 ± 0.97</td>
</tr>
<tr>
<td>EDA-R NBM</td>
<td>79.21 ± 0.33</td>
<td>85.29 ± 0.36</td>
<td>83.81 ± 0.69</td>
<td>79.90 ± 0.62</td>
<td>76.51 ± 0.99</td>
</tr>
<tr>
<td><strong>WNBM</strong></td>
<td>79.21 ± 0.33</td>
<td>79.90 ± 0.44</td>
<td>79.52 ± 0.34</td>
<td>77.36 ± 0.50</td>
<td>75.61 ± 0.58</td>
</tr>
<tr>
<td><strong>KS C4.5</strong></td>
<td>61.03 ± 1.29</td>
<td>62.90 ± 1.12</td>
<td>64.50 ± 1.24</td>
<td>66.61 ± 0.62</td>
<td>68.21 ± 1.28</td>
</tr>
<tr>
<td>EDA-R C4.5</td>
<td>61.03 ± 1.29</td>
<td>63.74 ± 0.73</td>
<td>65.12 ± 0.64</td>
<td>66.55 ± 0.47</td>
<td>66.96 ± 0.82</td>
</tr>
<tr>
<td><strong>KS LSVM</strong></td>
<td>88.24 ± 0.51</td>
<td>87.56 ± 0.41</td>
<td>85.62 ± 0.64</td>
<td>83.10 ± 0.49</td>
<td>79.88 ± 1.06</td>
</tr>
<tr>
<td>EDA-R LSVM</td>
<td>88.24 ± 0.51</td>
<td>87.90 ± 0.36</td>
<td>86.66 ± 0.43</td>
<td>84.07 ± 0.74</td>
<td>81.73 ± 0.56</td>
</tr>
<tr>
<td><strong>WLSVM</strong></td>
<td>88.24 ± 0.51</td>
<td>88.22 ± 0.44</td>
<td>88.08 ± 0.35</td>
<td>87.10 ± 0.32</td>
<td>85.86 ± 0.34</td>
</tr>
</tbody>
</table>
6.2. ACCEPTOR PREDICTION: AN ITERATIVE FEATURE SELECTION APPROACH

For the SVM, an increase in classification performance is noted, compared to dataset 1 and 2. Again, the result on the full feature set cannot be improved using feature selection methods, although the same classification performance can be reached using only 25% of the features (WLSVM).

Visualizing the weights derived from the EDA-R LSVM combination (Figure 6.7) reveals some remarkable, new patterns. In addition to the previous patterns, three new patterns, related to the inclusion of dependencies between adjacent nucleotides, can be observed. Firstly, it is observed that nucleotide dependencies immediately neighbouring the acceptor site are of great importance. Furthermore, two patterns related to the dinucleotides AG and TG emerge in the left part of the context.

Remark that the only result that can be drawn from this visualization is the fact that these are “important” features for classification. The method does not reveal whether e.g. AG occurs more often or less often at these positions in true acceptor sites than in false sites. To find out this information, inspection of the classification model or the datasets is needed. In the case of the AG-feature, an analysis of our dataset shows that there is a strong selection against the occurrence of AG dinucleotides in the left part of the context for true acceptors. This can be explained by the acceptor scanning mechanism (see introduction of this chapter). In this process, a protein binds to the branch point and then scans the sequence until an AG is encountered (usually the first AG encountered is the splice site). As a result, our feature selection method discovers this “AG-scanning” as very important in the distinction between true and false acceptors, as false acceptors will usually have more AG dinucleotides in the upstream part of the sequence. The second pattern (TG) was identified as being more abundant in true acceptor sites than in false sites, and is probably related to the T-abundance of the pyrimidine stretch.

Comparing the results of all feature selection combinations on the three datasets reveals some general trends for these acceptor datasets. For the NBM classifier, classification performance could be significantly improved upon using feature selection, especially using the EDA-R wrapper method, which achieves the best results when half of the features have been eliminated. Using feature selection on the most complex dataset achieves an F-measure of 85%, which is about 5% better than using all features of the simplest dataset. Overall, the EDA-R method gives the best results for NBM, followed by the KS filter method, and the embedded method WNBM.

For C4.5, applying feature selection techniques seems indispensable to achieve good classification performance. Moreover, the best results were achieved when only 5% of the features were used, showing that for our datasets, the methods works best with only a very small subset of features.

For the linear SVM, classification performance could not be improved upon using feature selection. At least equivalent results could be obtained using fea-
Figure 6.7: EDA based feature weighting for acceptor prediction (2096 features). The EDA-R approach was combined with the linear SVM.
6.2. ACCEPTOR PREDICTION: AN ITERATIVE FEATURE SELECTION APPROACH

Figure 6.8: Construction of the branch point features.

We start from the knowledge that little is known about the branch point motif, except the fact that it should contain the nucleotide A, and that the U2 snRNP binds to this part of the sequence. Given an upstream region of a potential acceptor site, we can thus consider every A as potentially belonging to the branch point. To select the A that most likely forms part of the branch point, we compare a local context of the A to the part of U2 that binds to the pre-mRNA. From the “fit” of the context to the U2 base-pairing part, we calculate a matching score. The A having the highest score is then considered the “putative” branch point. Besides this score, a number of other features can be derived from the putative branch point:

- the distance from the A to the acceptor site,
- the number of AG-dinucleotides appearing between the putative branch point and the acceptor,
- the percentage of nucleotides T appearing between the putative branch point and the acceptor.

6.2.4 A customized feature set for acceptor prediction

Using the knowledge derived from the previous experiments, we can now build a customized set of features for acceptor prediction. For this purpose, we created a number of new, important “branch point” features that are inspired by the AG-scanning feature. Figure 6.8 shows how these features are constructed. We start from the knowledge that little is known about the branch point motif, except the fact that it should contain the nucleotide A, and that the U2 snRNP binds to this part of the sequence. Given an upstream region of a potential acceptor site, we can thus consider every A as potentially belonging to the branch point. To select the A that most likely forms part of the branch point, we compare a local context of the A to the part of U2 that binds to the pre-mRNA. From the “fit” of the context to the U2 base-pairing part, we calculate a matching score. The A having the highest score is then considered the “putative” branch point. Besides this score, a number of other features can be derived from the putative branch point:

- the distance from the A to the acceptor site,
- the number of AG-dinucleotides appearing between the putative branch point and the acceptor,
- the percentage of nucleotides T appearing between the putative branch point and the acceptor.
and the percentage of nucleotides T or C (pyrimidines) appearing between the putative branch point and the acceptor.

Simulations of these new features showed the benefit of an additional parameter: the skip-parameter. Instead of directly counting the features mentioned above, first a number of nucleotides are skipped. Only then the counting of distance, number of AG dinucleotides and percentages T and C/T is started. The biological motivation behind the skip-parameter is the fact that U2 binds to the branch point, thereby probably blocking some of the sequence downstream of the branch point. Our simulations for acceptor prediction showed that the value 7 for the skip-parameter gave the best results.

Another addition to the feature set is motivated by the fact that the upstream part of the acceptor is intron (non-coding), while the downstream part is exon (coding). As a consequence, the Fourier features derived in the previous chapter can be added to capture this information. Furthermore, some other features like dinucleotide frequencies and AG/TG counts in the upstream part of the acceptor were added. This results in the following customized set of 875 features for acceptor prediction:

- position dependent nucleotides in a local context of [-5,+5] around the acceptor site (40 binary features)
- position dependent dinucleotides in a local context of [-5,+5] around the acceptor site (128 binary features)
- position dependent trinucleotides in a local context of [-5,+5] around the acceptor site (384 binary features)
- position invariant features in a local context of [-50,+50] around the acceptor site (128 binary features)
- AG/TG counts in windows varying from position -1 to position -50 upstream of the acceptor site (98 real-valued features)
- branch point features extracted from the window [-60,-10] (5 real-valued features)
- dinucleotide percentage features in the upstream part of the context (16 real-valued features)
- Fourier features for frequencies 0 and 1/3 extracted from [-60,-1] and [1,60] (76 real-valued features)

As this dataset contains some redundant and irrelevant features, we again performed feature selection to see how much smaller the set of features could be made without degrading classification performance. The idea behind this is to see if, using more complex features that are tailored to the classification problem at hand, we can achieve very good classification performance with a “core”
### 6.2. ACCEPTOR PREDICTION: AN ITERATIVE FEATURE SELECTION APPROACH

Customized dataset: 875 features

<table>
<thead>
<tr>
<th>Method</th>
<th>100%</th>
<th>50%</th>
<th>25%</th>
<th>10%</th>
<th>5%</th>
</tr>
</thead>
<tbody>
<tr>
<td>KS NBM</td>
<td>64.21 ± 0.32</td>
<td>62.08 ± 0.49</td>
<td>70.05 ± 2.36</td>
<td>80.67 ± 0.78</td>
<td>80.27 ± 0.55</td>
</tr>
<tr>
<td>EDA-R NBM</td>
<td>64.21 ± 0.32</td>
<td>83.58 ± 0.73</td>
<td>81.86 ± 1.11</td>
<td>78.50 ± 1.69</td>
<td>75.21 ± 3.80</td>
</tr>
<tr>
<td>KS C4.5</td>
<td>68.99 ± 0.78</td>
<td>69.23 ± 0.56</td>
<td>71.05 ± 0.63</td>
<td>72.37 ± 1.50</td>
<td>69.48 ± 0.97</td>
</tr>
<tr>
<td>EDA-R C4.5</td>
<td>68.99 ± 0.78</td>
<td>75.69 ± 0.66</td>
<td>76.80 ± 0.82</td>
<td>77.57 ± 0.50</td>
<td>78.61 ± 0.84</td>
</tr>
<tr>
<td>EDA-R LSVM</td>
<td>89.92 ± 0.39</td>
<td>90.03 ± 0.42</td>
<td>89.80 ± 0.62</td>
<td>89.03 ± 0.61</td>
<td>87.09 ± 1.13</td>
</tr>
<tr>
<td>WLSVM</td>
<td>89.92 ± 0.39</td>
<td>90.12 ± 0.39</td>
<td>90.10 ± 0.41</td>
<td>89.92 ± 0.55</td>
<td>88.60 ± 0.50</td>
</tr>
</tbody>
</table>

*Table 6.4: Acceptor prediction: F test comparisons for the customized dataset of 875 features.*
set of features. The results of the feature selection are shown in Table 6.4. In
general, it can be observed that for almost all combinations, similar or better
results than using the full feature set can be obtained using only 10% of the
features. The Naive Bayes classifier benefits the most from feature selection,
with an increase in F-measure of about 20% using only 50% of the features. For
C4.5, the best results are obtained with 5% of the features (EDA-R), improv-
ing the result on the full feature set by 10%. For the linear SVM a small, but
significant improvement could be obtained with 50% of the features (WLSVM).

Comparing the results of feature selection on this customized feature set to
the original results on the dataset containing the simplest features, some sub-
stantial improvements can be observed. For NBM, the initial result of 80.87%
could be improved to 83.58%, using 437 features. However, it should be noted
that a slightly better result (85.29%) could be obtained using 1048 features of
the dataset with 2096 features. For C4.5, the initial result of 55.25% could
be improved to 78.61% using only 44 features, an increase by more than 23%.
Similarly, the construction of a customized set of features was beneficial to the
linear SVM. The initial result of 84.45% could be improved to 90.12% using
437 features, a gain in F-measure of almost 6%.

For the best scoring method (WLSVM), the 10% best features are shown in
Figure 6.9. Features are grouped according to their type, and the remaining
features are shown in black. The following patterns can be observed:

- Only a small number of position dependent features are essential. These
  are the nucleotides at position 1 upstream, and 1 downstream, and pair
  wise interactions between position 2 and 1 upstream, and 1-2 and 2-3
downstream. Interactions between three adjacent nucleotides are not
  essential.

- Features capturing coding potential are not of major importance in ac-
  ceptor prediction. This can be observed by the low number of Fourier
  features and position invariant features that are selected.

- Features related to the branch point and AG-scanning are essential, espe-
  cially the distance between the (putative) branch point and the acceptor,
  and the number of intervening AG dinucleotides, a feature which is also
  captured by the AG counts.

- T-richness in the upstream part of the acceptor is essential. This charac-
  teristic is captured by the T-% branch point feature, the TG counts, and
  some of the dinucleotide percentage features in the upstream part of the
  context.

Our results show that significant improvements in classification performance
can be obtained using a careful tuning of the feature set. Furthermore, the
results of feature selection can be used to guide this construction process, giving
at the same time more insight in the underlying biological process.
6.2. ACCEPTOR PREDICTION: AN ITERATIVE FEATURE SELECTION APPROACH

Figure 6.9: A core set of features (shown in black) for acceptor prediction. The selected features are the best 10% features obtained using the WLSVM feature selection method.
6. FEATURE SELECTION FOR SPLICE SITE PREDICTION

6.2.5 A note on interpreting feature selection results

An important aspect to consider, when comparing different feature selection techniques and different classifiers, is how much the algorithms differ in selecting relevant features. We already showed that, for a given classifier, different feature selection techniques could behave very differently. As an example, consider the Naive Bayes classifier on the simplest dataset. When comparing the results for the 10% best features, it can be observed that feature selection using EDA-R achieves an F-measure that is almost 10% better than the result obtained using WNBM. As a result, the features that will be selected by EDA-R will be superior to the ones selected by WNBM in explaining which information is really relevant. Care should thus be taken when trying to deduce “knowledge” from selection techniques with inferior results.

A second aspect to consider is the comparison of feature selection techniques for different classifiers. Due to the different bias and variance aspects of each algorithm, different features might be used to construct the decision function, and features that might be important for one algorithm may be useless for another one. As a result, interpretations of the results of different algorithms should be done with great care.

To show the importance of being careful in drawing conclusions from feature selection results, we will compare the results for NBM and LSVM when using the EDA-R feature selection method. We start by comparing the results on the simplest dataset (400 features). For LSVM, we already showed the results in Figure 6.5. The results in the case of NBM are shown in Figure 6.10. In general, similar trends like the importance of the local context, poly-pyrimidine stretch and periodicities in the coding part of the context can be observed. However, also a number of differences exist. An important difference is the dark blue color of the nucleotides A in the upstream part of the context. This means the Naive Bayes method benefits from not including these features. Another difference are the nucleotides T at position -1 and -2, which are less important in the case of NBM than in the case of SVM.

A second example, illustrating the difference between the two classifiers even better concerns the dataset of 2096 features. For this dataset, the results of the feature weighting for the combination of LSVM with EDA-R were given in Figure 6.7. The results for EDA-R combined with NBM are shown in Figure 6.11. Some remarkable differences, all related to the poly-pyrimidine stretch can be noticed. Whereas in the dataset of 400 features the poly-pyrimidine stretch showed up as a very important feature, it has completely disappeared in the dataset of 2096 features, whereas in the case of SVM the pyrimidine stretch is retained. In the case of NBM, the features related to T-richness (both position dependent as position independent features) are mostly colored dark blue, again indicating that the algorithm benefits from discarding these features. On the other hand, the AG-scanning feature appears very important, similar to the result of the SVM.
Figure 6.10: EDA based feature weighting for acceptor prediction (400 features). The EDA-R approach was combined with the Naive Bayes classifier.

Figure 6.11: EDA based feature weighting for acceptor prediction (2096 features). The EDA-R approach was combined with the Naive Bayes classifier.
6. FEATURE SELECTION FOR SPLICE SITE PREDICTION

When interpreting the results for SVM and NBM independently, one could thus infer two contradictory conclusions: a) the poly-pyrimidine stretch is very important, and b) the poly-pyrimidine stretch is totally irrelevant. To explain these contradictions, a deeper inspection of both the classifiers as the biological problem is needed. In the case of acceptor prediction, the explanation boils down to the notion that the poly-pyrimidine stretch and AG-scanning are highly correlated features, and that NBM and SVM deal differently with correlated features. For NBM, it is known that significantly better results can be obtained when correlated features are discarded. As NBM performs better without correlated features, only the “best” of these features are used, which appear to be the AG-scanning features. This explains why most of the features related to the poly-pyrimidine stretch are discarded. SVM has its own way of dealing with correlated features, as it already implicitly performs a feature weighting. Nevertheless, it can be observed that the SVM is better able to use the information in correlated features, as both AG-scanning and T-richness appear to be important. Furthermore, the SVM achieves better classification performance than the NBM.

These results indicate that a comparative evaluation of classifiers and feature selection techniques provides a more robust way of extracting knowledge from the data. As a rule of thumb, we could say that generally important features can be identified by looking for common patterns when comparing different algorithms and feature selection techniques. Features that only appear using certain classifiers or selection techniques should be investigated in more detail, using as much domain knowledge as possible.

6.3 Donor prediction: an iterative feature selection approach

Similar to the case of acceptor prediction, we used an iterative feature selection approach to detect important features for donor prediction. The same combinations of feature selection algorithms and classifiers were used as in the case of acceptor prediction, and similar training and test set sizes were used. We start with a simple set of position dependent nucleotide features that were extracted from the local context of the donor site.

6.3.1 A simple dataset: position dependent nucleotides

Just like the analysis of acceptor prediction, we extracted nucleotide features from a local context of 100 nucleotides around the splice site (50 positions upstream, 50 positions downstream). These nucleotide features were again converted into binary format using sparse vector encoding, yielding a dataset of 400 binary features. For this dataset, the parameter C of the linear SVM was tuned to 0.05 and the population size and number of iterations of the EDA-R approach were respectively tuned to 500 and 20.
6.3. DONOR PREDICTION: AN ITERATIVE FEATURE SELECTION APPROACH

The results of the feature selection process are shown in Table 6.5. In most cases, the classification performance could not be improved using feature selection, except in the case of C4.5, where the best results are obtained using only 5% of the features. This results in a gain of 7% compared to using the full feature set. For NBM and LSVM, feature selection does not improve the classification performance, but results that were equivalent to using the full feature set could be obtained using only half of the features.

A visualization of the feature weights for the combination of LSVM and EDA-R is shown in Figure 6.12. A number of clear patterns emerge from this picture. Similar to acceptor prediction, the nucleotides neighbouring the donor site are of key importance, and periodicity can be observed in the coding part of the context, especially for the nucleotides T and G. Furthermore, the intron side reveals two clear patterns: the importance of nucleotides T all over the intron part, and the importance of nucleotides G immediately downstream of the donor site. The importance of nucleotides T is a known motif in plants, as plant introns tend to be T-rich [Lorkovic et al., 2000]. The importance of nucleotides G is a new feature, and a statistical analysis of our dataset reveals that this nucleotide is strongly under-represented in this part of the context for true donors, in contrast to the pseudo donors in the dataset. An exception occurs at position +3, where the nucleotide G is over-represented in true donors, as part of the donor consensus sequence (see also Figure 4.8).

6.3.2 Adding position invariant features

In an attempt to capture more compositional sequence information, we added position invariant trinucleotides to the feature set. These features were constructed in the same way as in the case of acceptor prediction. Table 6.6 shows the results of the feature selection experiments. It can be observed that significantly better results can be obtained using feature selection. For NBM, the best result was obtained using the EDA-R approach, using half of the features, while C4.5 again performed best using only 5% of the features (KS). For the linear SVM no improvements could be made, although 50% of the features could be eliminated without hurting the classification performance (KS). Comparing these results to the ones of the simplest dataset, it can be observed that only a slight improvement in classification performance could be obtained.

Figure 6.13 shows the visualization of the feature weights for this extended dataset. While the nucleotides neighbouring the donor site are still marked as highly relevant, it can be observed that some of the information from the other patterns is replaced by position invariant features. On the coding side, the periodicity pattern is less pronounced, and some triplets show up as highly relevant. On the intron side, the T-stretch is still visible in the position dependent features, but T-rich triplets (especially TTA, TTT, TTC and TCT) now also capture the T-richness. A slightly different phenomenon occurs for the stretch of G-features. Although some G-rich triplets are now shown to be
### Simple dataset: 400 features

<table>
<thead>
<tr>
<th>Method</th>
<th>100%</th>
<th>50%</th>
<th>25%</th>
<th>10%</th>
<th>5%</th>
</tr>
</thead>
<tbody>
<tr>
<td>KS NBM</td>
<td>85.71 ± 0.32</td>
<td>85.47 ± 0.40</td>
<td>83.97 ± 0.31</td>
<td>79.78 ± 0.26</td>
<td>74.26 ± 0.42</td>
</tr>
<tr>
<td>EDA-R NBM</td>
<td>85.71 ± 0.32</td>
<td>85.84 ± 0.26</td>
<td>83.58 ± 0.29</td>
<td>78.95 ± 0.72</td>
<td>72.39 ± 2.075</td>
</tr>
<tr>
<td>WNBM</td>
<td>85.71 ± 0.32</td>
<td>81.31 ± 0.78</td>
<td>75.69 ± 0.34</td>
<td>67.70 ± 0.55</td>
<td>55.20 ± 1.74</td>
</tr>
<tr>
<td>KS C4.5</td>
<td>68.81 ± 0.58</td>
<td>69.73 ± 0.73</td>
<td>71.73 ± 0.78</td>
<td>73.16 ± 0.65</td>
<td>75.03 ± 0.97</td>
</tr>
<tr>
<td>EDA-R C4.5</td>
<td>68.81 ± 0.58</td>
<td>69.46 ± 0.58</td>
<td>70.18 ± 0.87</td>
<td>71.12 ± 0.98</td>
<td>74.01 ± 0.66</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>100%</th>
<th>50%</th>
<th>25%</th>
<th>10%</th>
<th>5%</th>
</tr>
</thead>
<tbody>
<tr>
<td>KS LSVM</td>
<td>87.66 ± 0.35</td>
<td>87.13 ± 0.22</td>
<td>85.26 ± 0.32</td>
<td>80.29 ± 0.38</td>
<td>74.21 ± 0.64</td>
</tr>
<tr>
<td>EDA-R LSVM</td>
<td>87.66 ± 0.35</td>
<td>86.69 ± 0.29</td>
<td>84.41 ± 0.30</td>
<td>80.46 ± 0.34</td>
<td>77.08 ± 0.59</td>
</tr>
<tr>
<td>WLSVM</td>
<td>87.66 ± 0.35</td>
<td>87.40 ± 0.30</td>
<td>85.09 ± 0.36</td>
<td>81.24 ± 0.22</td>
<td>76.44 ± 0.45</td>
</tr>
</tbody>
</table>

**Table 6.5:** Donor prediction: F test comparisons for the dataset of 400 features.

### Extended dataset: 528 features

<table>
<thead>
<tr>
<th>Method</th>
<th>100%</th>
<th>50%</th>
<th>25%</th>
<th>10%</th>
<th>5%</th>
</tr>
</thead>
<tbody>
<tr>
<td>KS NBM</td>
<td>78.06 ± 0.53</td>
<td>78.23 ± 0.69</td>
<td>78.01 ± 0.55</td>
<td>80.41 ± 0.47</td>
<td>78.78 ± 0.54</td>
</tr>
<tr>
<td>EDA-R NBM</td>
<td>78.06 ± 0.53</td>
<td>86.41 ± 0.32</td>
<td>83.88 ± 0.90</td>
<td>79.85 ± 0.74</td>
<td>75.98 ± 0.58</td>
</tr>
<tr>
<td>WNBM</td>
<td>78.06 ± 0.53</td>
<td>72.38 ± 0.75</td>
<td>72.57 ± 0.72</td>
<td>69.28 ± 0.68</td>
<td>65.58 ± 2.24</td>
</tr>
<tr>
<td>KS C4.5</td>
<td>71.39 ± 0.73</td>
<td>72.16 ± 0.95</td>
<td>73.83 ± 0.68</td>
<td>75.85 ± 0.33</td>
<td>77.45 ± 0.61</td>
</tr>
<tr>
<td>EDA-R C4.5</td>
<td>71.39 ± 0.73</td>
<td>71.62 ± 0.74</td>
<td>72.59 ± 0.67</td>
<td>73.63 ± 0.85</td>
<td>75.52 ± 0.81</td>
</tr>
<tr>
<td>KS LSVM</td>
<td>89.58 ± 0.27</td>
<td>89.43 ± 0.32</td>
<td>88.25 ± 0.26</td>
<td>84.59 ± 0.35</td>
<td>80.22 ± 0.67</td>
</tr>
<tr>
<td>EDA-R LSVM</td>
<td>89.58 ± 0.26</td>
<td>88.59 ± 0.38</td>
<td>87.12 ± 0.33</td>
<td>84.55 ± 0.59</td>
<td>81.41 ± 0.71</td>
</tr>
<tr>
<td>WLSVM</td>
<td>89.56 ± 0.27</td>
<td>89.31 ± 0.24</td>
<td>88.19 ± 0.30</td>
<td>85.66 ± 0.39</td>
<td>82.57 ± 0.37</td>
</tr>
</tbody>
</table>

**Table 6.6:** Donor prediction: F test comparisons for the dataset of 528 features.
6.3. DONOR PREDICTION: AN ITERATIVE FEATURE SELECTION APPROACH

Figure 6.12: EDA based feature weighting for donor prediction (400 features). The EDA-R approach was combined with the linear SVM.

Figure 6.13: EDA based feature weighting for donor prediction (528 features). The EDA-R approach was combined with the linear SVM.
highly relevant, it can be observed that most of the information stays in the position dependent features ranging from position 1 to position 16 downstream of the donor site. While the T-richness thus seems to be a more general compositional feature of the intron, the inhibition of nucleotides G turns out to be specific for the sequence immediately downstream of the donor site.

6.3.3 Adding position dependent dinucleotides

To investigate whether any dependencies between adjacent nucleotides exist, which may point to an underlying scanning mechanism, we again added the set of position dependent dinucleotides. Together with the position dependent nucleotides and the position invariant features this results in a set of 2096 binary features. The feature selection results for this dataset are shown in Table 6.7. The results show that the inclusion of these features does not really improve the classification accuracy. The only algorithm for which the results are marginally improved is the linear SVM. For NBM the results are similar to the results on the previous dataset, and for C4.5 the results are even slightly worse than on the extended dataset (528 features).

To investigate the features that were selected as relevant for the SVM, the feature weights of the combination of SVM with EDA-R are shown in Figure 6.14. From this figure we can learn that the only important interactions between pairs of adjacent features occur in the immediate neighbourhood of the donor site. No scanning mechanism as in the case of acceptor prediction appears to be involved.

6.3.4 A customized feature set for donor prediction

Guided by our observations on the previously constructed datasets we constructed a customized set of features for donor prediction. Similarly to acceptor prediction, we hoped to identify a small set of core features that characterizes the classification of donor sites. This resulted in the following customized set of 816 features for donor prediction:

- position dependent nucleotides in a local context of [-5,+20] around the donor site (100 binary features)
- position dependent dinucleotides in a local context of [-5,+5] around the donor site (128 binary features)
- position dependent trinucleotides in a local context of [-5,+5] around the donor site (384 binary features)
- position invariant trinucleotides in a local context of [-50,+50] around the donor site (128 binary features)
- Fourier features for frequencies 0 and 1/3 extracted from [-60,-1] and [1,60] (76 real-valued features)
### Complex dataset: 2096 features

<table>
<thead>
<tr>
<th>Method</th>
<th>100%</th>
<th>50%</th>
<th>25%</th>
<th>10%</th>
<th>5%</th>
</tr>
</thead>
<tbody>
<tr>
<td>KS NBM</td>
<td>78.95 ± 0.55</td>
<td>78.72 ± 0.77</td>
<td>79.85 ± 0.56</td>
<td>81.67 ± 0.56</td>
<td>80.49 ± 0.54</td>
</tr>
<tr>
<td>EDA-R NBM</td>
<td>78.95 ± 0.55</td>
<td>86.30 ± 0.44</td>
<td>84.31 ± 0.78</td>
<td>79.67 ± 0.56</td>
<td>77.68 ± 0.83</td>
</tr>
<tr>
<td>WNBM</td>
<td>78.95 ± 0.55</td>
<td>78.84 ± 0.54</td>
<td>77.28 ± 0.60</td>
<td>78.47 ± 1.06</td>
<td>79.13 ± 0.56</td>
</tr>
<tr>
<td>KS C4.5</td>
<td>68.23 ± 0.51</td>
<td>69.40 ± 0.78</td>
<td>71.06 ± 0.43</td>
<td>73.93 ± 0.54</td>
<td>75.68 ± 0.52</td>
</tr>
<tr>
<td>EDA-R C4.5</td>
<td>68.23 ± 0.51</td>
<td>68.81 ± 0.65</td>
<td>70.14 ± 0.45</td>
<td>71.72 ± 0.63</td>
<td>72.56 ± 0.78</td>
</tr>
<tr>
<td>KS LSVM</td>
<td>90.10 ± 0.20</td>
<td>90.27 ± 0.30</td>
<td>89.01 ± 0.34</td>
<td>85.49 ± 0.46</td>
<td>82.48 ± 0.44</td>
</tr>
<tr>
<td>EDA-R LSVM</td>
<td>90.24 ± 0.25</td>
<td>89.98 ± 0.19</td>
<td>89.10 ± 0.24</td>
<td>86.90 ± 0.27</td>
<td>84.92 ± 0.51</td>
</tr>
<tr>
<td>WLSVM</td>
<td>90.11 ± 0.23</td>
<td>90.41 ± 0.21</td>
<td>90.31 ± 0.23</td>
<td>90.00 ± 0.26</td>
<td>88.62 ± 0.35</td>
</tr>
</tbody>
</table>

### Customized dataset: 816 features

<table>
<thead>
<tr>
<th>Method</th>
<th>100%</th>
<th>50%</th>
<th>25%</th>
<th>10%</th>
<th>5%</th>
</tr>
</thead>
<tbody>
<tr>
<td>KS NBM</td>
<td>85.09 ± 0.39</td>
<td>85.75 ± 0.60</td>
<td>85.33 ± 0.57</td>
<td>85.33 ± 0.59</td>
<td>84.71 ± 0.53</td>
</tr>
<tr>
<td>EDA-R NBM</td>
<td>85.09 ± 0.39</td>
<td>88.66 ± 0.25</td>
<td>88.16 ± 0.61</td>
<td>86.52 ± 0.67</td>
<td>84.48 ± 1.79</td>
</tr>
<tr>
<td>KS C4.5</td>
<td>72.63 ± 1.13</td>
<td>73.36 ± 0.55</td>
<td>74.92 ± 0.56</td>
<td>77.17 ± 0.43</td>
<td>78.59 ± 0.69</td>
</tr>
<tr>
<td>EDA-R C4.5</td>
<td>72.63 ± 1.13</td>
<td>77.66 ± 0.58</td>
<td>78.83 ± 0.59</td>
<td>79.61 ± 0.57</td>
<td>81.23 ± 0.92</td>
</tr>
<tr>
<td>KS LSVM</td>
<td>91.47 ± 0.28</td>
<td>91.22 ± 0.23</td>
<td>90.83 ± 0.35</td>
<td>88.93 ± 0.93</td>
<td>86.37 ± 1.74</td>
</tr>
<tr>
<td>EDA-R LSVM</td>
<td>91.47 ± 0.28</td>
<td>91.54 ± 0.31</td>
<td>91.40 ± 0.31</td>
<td>90.71 ± 0.39</td>
<td>89.77 ± 0.29</td>
</tr>
</tbody>
</table>

Table 6.7: Donor prediction: F test comparisons for the dataset of 2096 features.

Table 6.8: Donor prediction: F test comparisons for the customized dataset of 816 features.
Figure 6.14: EDA based feature weighting for donor prediction (2096 features). The EDA-R approach was combined with the linear SVM.
6.3. DONOR PREDICTION: AN ITERATIVE FEATURE SELECTION APPROACH

Figure 6.15: A core set of features (shown in black) for donor prediction. The selected features are the best 25% features obtained using the WLSVM feature selection method.
To this customized set of features we again applied feature selection. The results of these experiments are shown in Table 6.8. A general trend that can be observed is the fact that 75% of the features can be eliminated without sacrificing classification performance. For NBM, the best results (88.66%) are obtained with EDA-R using 50% of the features. Comparing this result to the initial results on the simplest dataset (85.71), a gain of about 3% is obtained. For C4.5 the best results are obtained with EDA-R using only 5% of the features (81.23%). Comparing this to the result on the simplest dataset (68.81), a significant gain of 13% can be noted. For the SVM, the initial result of 87.66% was improved to 91.54%, an increase by 4%.

For the best scoring method (WLSVM), the 25% best scoring features are shown in Figure 6.15. The following features are shown to be essential for donor prediction:

- position dependent features, ranging from position 3 upstream to position 4 downstream,
- position dependent nucleotides G in the range [1,20] downstream of the donor site,
- T-rich and G-rich position invariant features in the downstream part of the context,
- features capturing coding potential (especially the Fourier features).

6.4 Summary

In this chapter we investigated the application of feature selection techniques to the biological problem of splice site prediction. For each of the two types of splice sites (donor sites and acceptor sites) we followed an iterative procedure of feature construction and feature selection steps, thereby increasing the number and complexity of the features describing the sequences around the splice site. In a final step, we constructed for each of the types of splice sites a customized set of features from which we derived a minimal set of features that characterizes the classification problem.

A general trend that can be observed is that donor prediction seems somewhat an easier classification problem than acceptor prediction, as better classification results were obtained for donor prediction with all classifiers. This can be explained by the biological processes underlying donor and acceptor recognition. While the recognition of donor sites is almost entirely defined by binding mechanisms, a more complex machinery is involved in recognizing the acceptor site, including recognition of the branch point and scanning toward the acceptor.

The application of feature selection techniques led to the identification of both
6.4. SUMMARY

a number of known and new insights for splice site prediction. For acceptor prediction, we discovered a new biologically motivated feature: AG-scanning [Saeys et al., 2004a]. For donor prediction the selection against the occurrence of nucleotides G immediately downstream of the donor site turned out to be essential for building a good donor site predictor.

Another interesting result is that the addition of features that capture the coding potential on both sides of the splice site contributes in a different way to donor and acceptor prediction. While for donor prediction, including such features proved to be essential, they did not contribute much for acceptor prediction, where the major contribution comes from features related to AG-scanning.
Chapter 7

Feature selection for TIS prediction

The boughs of no two trees ever have
the same arrangement.
Nature always produces individuals;
she never produces classes.
- Lydia Maria Child

7.1 Introduction

When the pre-mRNA is processed by the spliceosome, introns are removed and
the mature mRNA is formed (see also Figure 4.1). The next step in the synthe-
sis from gene to protein is the actual translation of the mRNA (the code) into
the protein. In this step, the start codon is looked for, and from this position
on, the mRNA is translated codon per codon into amino acids. These amino
acids are synthesized and concatenated by the ribosome.
According to the scanning model of Kozak [Kozak, 1996], the ribosome binds
to the 5’ end of the mRNA and then scans the sequence until it finds the first
ATG (start codon) that is in an optimal nucleotide context. From this start
codon, translation is then initiated.

In most cases, the translation initiation site (TIS) will be the first ATG when
scanning the sequence from the 5’ end. However, with natural mRNAs, three
escape mechanisms allow access to ATG codons which, although not first, are
still close to the 5’ end of the mRNA: leaky scanning, reinitiation and internal
initiation of translation [Kozak, 1999]. As a result, classification techniques can
be used to extract sequence information that allows to discriminate between
real TIS and pseudo TIS.
7.2 TIS PREDICTION: AN ITERATIVE FEATURE SELECTION APPROACH

A number of machine learning methods to predict translation initiation sites have been described in the literature. The first method described to model translation initiation sites was a weight matrix, described by Kozak [Kozak, 1987]. The basic line of further research was set by Pedersen and Nielsen [Pedersen and Nielsen, 1997] who used a neural network approach, also taking into account nucleotides from a larger local context around the TIS. Around the same time, Salzberg used a conditional probability (CP) matrix (a method equivalent to a first order Markov model) to model TIS, acceptors and donors [Salzberg, 1997]. These ideas were further combined in [Zien et al., 2000], who combined the use of SVMs with specially developed kernels based on Salzberg’s CP matrices.

The method achieving the best results up to now was proposed by Hatzigeorgiou [Hatzigeorgiou, 2002] and is based on a multi-step integrated neural network. The use of feature generation and feature selection for TIS prediction was explored by Zeng et al. [Zeng et al., 2002]. In this paper, the authors combined the use of correlation based feature selection (CFS) with a wide range of classifiers and combinations of classifiers. Their results are comparable to the results of Hatzigeorgiou, but are achieved with only a small subset of features.

Up to now, few methods for identifying translation initiation sites focus on plants. Of the methods described above, all focus on vertebrate genomes. The only method that was tested on plants was the neural network approach of Pedersen and Nielsen, which they also applied to Arabidopsis.

Similar to the application of splice site prediction, we here describe an iterative feature selection approach to model the translation initiation sites in the PlantGene dataset.

7.2 TIS prediction: an iterative feature selection approach

From the PlantGene dataset, 1284 true translation initiation sites having a local context of at least 120 nucleotides upstream and downstream were selected. The negative instances were selected from introns, exons and intergenic regions. A class imbalance of one positive versus six negative examples was enforced, resulting in a dataset with 1284 positive and 7710 negative examples. From this dataset, five pairs of training and test sets were sampled (with the same class imbalance), similar to the experiments in coding potential prediction and splice site prediction.

Like the previous experiments on splice site prediction, we adopted an iterative feature selection approach, starting with a simple set of position dependent nucleotides, followed by a gradual expansion of the feature set.
7. FEATURE SELECTION FOR TIS PREDICTION

A simple set of position dependent nucleotides was extracted from the local context of the translation initiation site. Nucleotides were extracted from a window of 50 bases upstream, and 50 bases downstream of the site, and were converted to binary features using sparse vector encoding. This results in a dataset of 400 binary features. The feature selection graphs are shown in Figure 7.1. In general, more than half of the features can be eliminated before classification performance drastically drops down. On this dataset, the Naive Bayes classifier outperforms the SVM, regardless of the feature selection technique. The C4.5 decision tree consistently performs worse than NBM and SVM, yet benefits the most from feature selection. A statistical analysis of the different feature subsets is shown in Table 7.1. Compared to the full feature set, the only significant increase in classification performance is achieved by the C4.5 decision tree using only 5% of the features.

To gain more insight into the problem, finding out which features are the most relevant, the EDA based feature weighting approach was used. Color coding the feature weights is shown in Figure 7.2, where an EDA-R approach was combined with the Naive Bayes classifier. From this figure, a number of patterns
7.2. TIS PREDICTION: AN ITERATIVE FEATURE SELECTION APPROACH

**Simple dataset : 400 features**

<table>
<thead>
<tr>
<th>Method</th>
<th>All</th>
<th>50%</th>
<th>25%</th>
<th>10%</th>
<th>5%</th>
</tr>
</thead>
<tbody>
<tr>
<td>KS NBM</td>
<td>61.62 ± 1.26</td>
<td>61.78 ± 1.11</td>
<td>60.60 ± 1.47</td>
<td>55.43 ± 1.63</td>
<td>50.64 ± 1.41</td>
</tr>
<tr>
<td>EDA-R NBM</td>
<td>61.62 ± 1.26</td>
<td>60.58 ± 0.83</td>
<td>58.43 ± 1.06</td>
<td>54.05 ± 1.18</td>
<td>49.37 ± 1.39</td>
</tr>
<tr>
<td>WNBM</td>
<td>61.62 ± 1.26</td>
<td>61.84 ± 0.97</td>
<td>60.55 ± 1.28</td>
<td>53.29 ± 2.30</td>
<td>44.30 ± 1.09</td>
</tr>
<tr>
<td>KS C4.5</td>
<td>36.32 ± 2.25</td>
<td>37.17 ± 2.07</td>
<td>38.25 ± 1.29</td>
<td>39.94 ± 1.43</td>
<td>40.69 ± 1.67</td>
</tr>
<tr>
<td>EDA-R C4.5</td>
<td>36.32 ± 2.25</td>
<td>36.51 ± 2.05</td>
<td>36.25 ± 2.25</td>
<td>36.49 ± 1.67</td>
<td>37.53 ± 1.88</td>
</tr>
<tr>
<td>WNBM</td>
<td>57.24 ± 0.72</td>
<td>56.80 ± 0.71</td>
<td>54.90 ± 1.16</td>
<td>44.67 ± 1.04</td>
<td>26.18 ± 11.45</td>
</tr>
<tr>
<td>KS LSVM</td>
<td>63.21 ± 0.76</td>
<td>63.76 ± 0.90</td>
<td>64.02 ± 1.03</td>
<td>63.71 ± 1.40</td>
<td>62.19 ± 1.55</td>
</tr>
<tr>
<td>EDA-R LSVM</td>
<td>63.21 ± 0.76</td>
<td>68.91 ± 0.84</td>
<td>68.01 ± 1.13</td>
<td>65.33 ± 1.51</td>
<td>62.45 ± 2.15</td>
</tr>
<tr>
<td>WLSVM</td>
<td>63.21 ± 0.76</td>
<td>64.49 ± 0.67</td>
<td>63.95 ± 0.93</td>
<td>62.15 ± 1.16</td>
<td>58.35 ± 1.22</td>
</tr>
<tr>
<td>KS C4.5</td>
<td>45.69 ± 1.77</td>
<td>46.67 ± 1.31</td>
<td>48.70 ± 1.72</td>
<td>50.09 ± 1.12</td>
<td>51.41 ± 1.46</td>
</tr>
<tr>
<td>EDA-R C4.5</td>
<td>45.69 ± 1.77</td>
<td>46.53 ± 2.13</td>
<td>46.62 ± 1.82</td>
<td>47.33 ± 2.04</td>
<td>48.98 ± 1.61</td>
</tr>
<tr>
<td>KS LSVM</td>
<td>66.77 ± 1.21</td>
<td>68.35 ± 1.09</td>
<td>67.18 ± 0.84</td>
<td>64.98 ± 1.01</td>
<td>59.66 ± 0.83</td>
</tr>
<tr>
<td>EDA-R LSVM</td>
<td>66.77 ± 1.21</td>
<td>66.67 ± 1.23</td>
<td>66.88 ± 1.14</td>
<td>63.04 ± 1.41</td>
<td>58.85 ± 1.96</td>
</tr>
<tr>
<td>WLSVM</td>
<td>66.77 ± 1.21</td>
<td>67.73 ± 1.24</td>
<td>66.90 ± 0.99</td>
<td>65.32 ± 1.09</td>
<td>62.13 ± 1.38</td>
</tr>
</tbody>
</table>

Table 7.1: TIS prediction: F test comparisons for the dataset of 400 features.

**Extended dataset : 528 features**

<table>
<thead>
<tr>
<th>Method</th>
<th>All</th>
<th>50%</th>
<th>25%</th>
<th>10%</th>
<th>5%</th>
</tr>
</thead>
<tbody>
<tr>
<td>KS NBM</td>
<td>63.21 ± 0.76</td>
<td>63.76 ± 0.90</td>
<td>64.02 ± 1.03</td>
<td>63.71 ± 1.40</td>
<td>62.19 ± 1.55</td>
</tr>
<tr>
<td>EDA-R NBM</td>
<td>63.21 ± 0.76</td>
<td>68.91 ± 0.84</td>
<td>68.01 ± 1.13</td>
<td>65.33 ± 1.51</td>
<td>62.45 ± 2.15</td>
</tr>
<tr>
<td>WNBM</td>
<td>63.21 ± 0.76</td>
<td>64.49 ± 0.67</td>
<td>63.95 ± 0.93</td>
<td>62.15 ± 1.16</td>
<td>58.35 ± 1.22</td>
</tr>
<tr>
<td>KS C4.5</td>
<td>45.69 ± 1.77</td>
<td>46.67 ± 1.31</td>
<td>48.70 ± 1.72</td>
<td>50.09 ± 1.12</td>
<td>51.41 ± 1.46</td>
</tr>
<tr>
<td>EDA-R C4.5</td>
<td>45.69 ± 1.77</td>
<td>46.53 ± 2.13</td>
<td>46.62 ± 1.82</td>
<td>47.33 ± 2.04</td>
<td>48.98 ± 1.61</td>
</tr>
<tr>
<td>KS LSVM</td>
<td>66.77 ± 1.21</td>
<td>68.35 ± 1.09</td>
<td>67.18 ± 0.84</td>
<td>64.98 ± 1.01</td>
<td>59.66 ± 0.83</td>
</tr>
<tr>
<td>EDA-R LSVM</td>
<td>66.77 ± 1.21</td>
<td>66.67 ± 1.23</td>
<td>66.88 ± 1.14</td>
<td>63.04 ± 1.41</td>
<td>58.85 ± 1.96</td>
</tr>
<tr>
<td>WLSVM</td>
<td>66.77 ± 1.21</td>
<td>67.73 ± 1.24</td>
<td>66.90 ± 0.99</td>
<td>65.32 ± 1.09</td>
<td>62.13 ± 1.38</td>
</tr>
</tbody>
</table>

Table 7.2: TIS prediction: F test comparisons for the dataset of 528 features.
**Figure 7.2:** EDA based feature weighting for TIS prediction (400 features). The EDA-R approach was combined with the Naive Bayes classifier.

**Figure 7.3:** EDA based feature weighting for TIS prediction (528 features). The EDA-R approach was combined with the Naive Bayes classifier.
7.2. **TIS PREDICTION: AN ITERATIVE FEATURE SELECTION APPROACH**

emerge:

- Similar to splice site prediction, the local context around the site captures the most important information. The nucleotide A in the upstream part of the context is of high importance, as could also be derived from the sequence logo (see Figure 4.8).

- Periodicity in the downstream part of the context. Just like in the case of acceptor prediction, the downstream part of the context represents a coding region. Whereas in acceptor prediction, a clear bias toward periodicity for the nucleotides T and G in the same phase could be observed, for TIS prediction the nucleotides A and C, and T and G seem to appear pairwise in the same phase. Another difference in the periodicity pattern with acceptor prediction is the fact that the pattern for TIS does not start immediately after the site, but only emerges starting from about 20 nucleotides downstream of the site.

- A last pattern that can be observed, is the importance of the nucleotide G in the upstream part of the context.

### 7.2.2 Adding position invariant features

To capture the importance of compositional features, an extended feature set was built, adding to the position dependent features also a number of position invariant features. For now, we only include position invariant 3-mers. Analogous to the case of splice site prediction, this results in a second dataset, consisting of 528 features. Figure 7.4 shows the feature selection graphs using this extended feature set. Comparing the results on the full feature set, SVM and C4.5 benefit the most from the inclusion of position invariant features. Both algorithms gain about 10% in classification performance (F-measure), whereas NBM only increases about 2%. However, when NBM is combined with the EDA-R feature selection method, its performance can be increased by 5%, thereby outperforming even the SVM, which had a better performance on the full feature set.

A statistical analysis of the different feature subsets is shown in Table 7.2. Compared to the simple dataset of 400 features, more significant improvements can be made using feature selection techniques. Again, the algorithm profiting most from feature selection is the C4.5 decision tree, achieving its best performance with only 5% of the features, increasing its performance with 6%, compared to using the whole feature set.

Again, the EDA based feature weighting can then be used to investigate which of these new features are the most relevant ones. A color coding of the feature weights is shown in Figure 7.3. Comparing the results to the feature weighting of the dataset using only position dependent features provides some useful insights:
The importance of G nucleotides in the upstream part of the context has completely disappeared from the position dependent features. On the other hand, position invariant features having a G-rich composition are shown to be highly relevant. This indicates that the position dependent features in the small dataset were trying to capture some form of information that was rather position invariant.

A closer look at the top scoring position invariant features in the upstream part of the context identifies the following triplets as most relevant: GCC, GGA, GCC, ATG, TAC, TCC, CAG, GAC and GCT. The majority of these triplets contains the nucleotides G and C, pointing to the importance of GC in the upstream part of the acceptor. These features probably try to catch some compositional information of the UTR, which precedes the translation initiation site.

The periodicity in the downstream part of the context is less pronounced than in the case of the simple dataset, showing that some of the coding information is now captured by the position invariant features belonging to the downstream part of the context.
7.2. TIS PREDICTION: AN ITERATIVE FEATURE SELECTION APPROACH

7.2.3 An educated guess: including position dependent 3-mers

Remembering the biological mechanisms involved in the detection of acceptor splice sites, it is worthwhile noticing that a similar “scanning” mechanism occurs in the detection of translation initiation sites. In the case of acceptor sites the spliceosome scans for an acceptor site, starting from the branch point. In a similar fashion, the ribosome binds upstream of the TIS and then selects an appropriate ATG as the start codon.

Just like the emergence of the AG-feature in acceptors, we could thus suspect an ATG-feature in the case of TIS. To verify this hypothesis, a large feature set was constructed containing:

- the position dependent nucleotides in a context of 50 bases upstream and downstream of the TIS (400 binary features),
- the position dependent dinucleotides (2-mers) in a context of 50 bases upstream and downstream of the TIS (1568 binary features),
- the position dependent trinucleotides (3-mers) in a context of 50 bases upstream and downstream of the TIS (6144 binary features).

This results in a dataset totaling 8112 binary features. To this dataset, we applied the EDA based feature ranking and weighting method, combined with the Naive Bayes classifier (distribution of 500 samples, 20 iterations). From these results, four clear patterns show up as important:

1. Dependencies between nucleotides, adjacent to the translation initiation site, especially the AA and AAA features immediately preceding TIS, and GC and GC\{T,C,G\} immediately following the TIS.

2. The ATG trinucleotide in the upstream part of the TIS is highly important, confirming our hypothesis about the scanning mechanism.

3. The in-frame TAA triplet in the downstream part of the context. As TAA is a stop codon, real TIS will not have an in-frame stop codon within 50 nucleotides, as this would lead to a very short gene, which is highly improbable.

4. The in-frame TGA triplet in the downstream part of the context. Similarly to TAA, this is also a stop codon.

Including these features significantly enhances classification performance for NBM: an average F-measure of 70 ± 0.864 was reached, compared to 61% on the dataset with 400 features, and 63% on the dataset with 528 features. However, many of the included position dependent dimers and triplets turned out to be irrelevant, paving the way towards a reduced set of well-chosen features.
7.2.4 Putting the parts together: a customized feature set for TIS prediction

To obtain a reduced set of features, suitable for TIS prediction, we combined the results of the previous experiments, adding some complex, tailored features. This results in the following set of 762 features:

- position dependent nucleotides, in a local context of 5 positions upstream and 5 positions downstream of the TIS (40 binary features)
- position dependent dinucleotides, also in a context of [-5,5] around the TIS (128 binary features)
- position dependent triplets, in a context of [-5,5] around the TIS (384 binary features)
- position invariant triplets, in a context of [-50,50] around the TIS (128 binary features)
- start and stop codon features: the start codon features count the number of ATG triplets in the upstream part of the context (both in-frame as global), the stop codon features count the number of TAA and TGA triplets in the downstream part of the context (both in-frame as global, 6 real-valued features)
- Fourier coding features for frequencies 0 and 1/3, separated for both parts of the context (76 real-valued features)

Figure 7.5 shows the results of the feature selection graphs for the reduced set of features. The C parameter of the SVM was again tuned to this dataset (C=0.05). Compared to the dataset consisting of both position dependent and position invariant features (528 binary features), a remarkable gain in performance (about 9% in F-measure) can be observed. Table 7.3 shows the statistical analysis of different feature subset sizes for this dataset. For all combinations of classification algorithms and feature selection techniques, it can be observed that roughly 75% of the features can be eliminated without degrading the classification performance. This means that from the 762 features in the customized feature set, about 200 features are sufficient to form a “core” set of features for TIS prediction.

Again, it can be observed that NBM and C4.5 benefit the most from feature selection. Using EDA-R, an increase by 3% in F-measure can be obtained using only half of the features, while the performance of C4.5 can be improved with 5% using only 5% of the features. Also remark that NBM combined with EDA-R outperforms the combination of EDA-R and LSVM, and performs almost as good as WLSVM.

When comparing the results of feature selection on the customized dataset to the initial results on the simple dataset, some remarkable improvements
### 7.2. TIS PREDICTION: AN ITERATIVE FEATURE SELECTION APPROACH

<table>
<thead>
<tr>
<th>Method</th>
<th>All</th>
<th>50%</th>
<th>25%</th>
<th>10%</th>
<th>5%</th>
</tr>
</thead>
<tbody>
<tr>
<td>KS NBM</td>
<td>72.44 ± 1.18</td>
<td>71.78 ± 1.53</td>
<td>70.96 ± 1.52</td>
<td>68.82 ± 1.61</td>
<td>66.82 ± 1.75</td>
</tr>
<tr>
<td>KS C4.5</td>
<td>62.06 ± 2.80</td>
<td>54.82 ± 1.36</td>
<td>57.16 ± 1.30</td>
<td>57.34 ± 1.42</td>
<td></td>
</tr>
<tr>
<td>L SVM</td>
<td>75.06 ± 1.37</td>
<td>74.68 ± 1.71</td>
<td>74.01 ± 1.54</td>
<td>71.60 ± 2.25</td>
<td></td>
</tr>
</tbody>
</table>

Table 7.3: TIS prediction: F test comparisons for the customized dataset.
in classification performance can be noticed. For Naive Bayes, the initial F-measure of 61.62% using 400 features could be improved to about 75% using EDA-R, a gain of 13%. This gain in performance could be obtained using 381 “tailored” features. In addition, a slightly worse result of 74.4% could be obtained using only 190 of these features.

For the C4.5 decision tree, the gain in classification performance is even higher. Starting with an F-measure of 36.32% on the simple dataset, a boost of 22% could be obtained using only 38 of the 762 customized features, yielding an F-measure of more than 58%, using the EDA-R combination with C4.5.

A similar observation can be made in the case of the linear SVM, where the initial F-measure of 57% could be improved to 75%, a gain in F-measure of 18%. However, in this case it should be noted that this gain in performance only results from the iterative feature selection/construction process, and not by doing an additional feature selection on the set of customized features, as opposed to the other classifiers. Although feature selection on the final set of customized features does not result in better classification performance for the SVM, it is important to note that equivalent results can be obtained using only 190 of the 762 features.
7.2. TIS PREDICTION: AN ITERATIVE FEATURE SELECTION APPROACH

Figure 7.6: A core set of features (shown in black) for TIS prediction. The selected features are the best 25% features obtained using the WLSVM feature selection method.
To investigate which features are essential for TIS prediction, we plotted the 25% best features obtained with the best scoring combination (WLSVM). These features are grouped by type and are shown in Figure 7.6. Features that belong to the 25% of best scoring features are colored black, the other features are left blank. From this a number of conclusions can be drawn:

- in the position dependent nucleotides, position 3 upstream and positions 1 to 4 downstream seem most important,
- in all position dependent features, A-rich features show up as important in the upstream part immediately preceding the translation initiation site,
- coding information is essential, as is indicated by the large number of selected Fourier features, combined with some of the position invariant features,
- start-codon and stop-codon features are essential for a correct prediction of the TIS.

### 7.3 Summary

In this chapter we investigated the use of feature selection techniques for the problem of predicting translation initiation sites (TIS, start-codons). Using an iterative approach of feature construction and feature selection we showed how classification performance could be gradually improved. We started with a simple set of position dependent nucleotide features, which was extended using position-invariant 3-mers, and position dependent dinucleotide and trinucleotide features. This led to the identification of both a number of known and new features, such as the presence of ATG before the real start-codon, in-frame stop-codons downstream of the start codon, and compositional information related to the presence of UTR upstream of the start-codon. Combining all feature selection efforts, we constructed a customized set of features for TIS prediction, making use of our previously developed Fourier-features and also some newly constructed features for recognition of start- and stop-codons. From this set, a core set of features that are essential for TIS prediction could be derived.
Chapter 8

Conclusions, reflections and future prospects

Facilius per partes in cognitionem totius adducimur.
- Seneca

8.1 Conclusions

The selection of relevant features for classification of nucleic acid sequences is a challenging topic of research. The large amounts of available data contrast with the limited amount of knowledge about important biological mechanisms, such as those involved in transcription and translation. From a scientific point of view, dealing with such problems demands some creativity and promotes the interaction between different research disciplines.

In chapter 1, we started by describing the motivations for this work, introducing the main themes upon which our research is inspired. The remaining chapters can be roughly divided in two parts: chapters 2 and 3 are more mathematical/algorithmical, while chapters 4 to 7 are practical applications in the domain of gene prediction. Chapter 2 introduced supervised learning and classification. In this chapter we explained the classification techniques that were used throughout the work: the Naive Bayes classifier, the Support Vector Machine and the C4.5 decision tree. Furthermore we introduced some basic concepts about model selection and model evaluation.

In chapter 3 we reached the heart of the matter: feature selection techniques. In this chapter, we motivated the use of feature selection, and we gave a comprehensive overview of the various existing techniques, discussing their strengths and weaknesses. This led to the development of a number of new techniques for feature selection, and a new tool for feature selection for large-scale datasets.
8.1. CONCLUSIONS

(FeaST). A first new technique was the Constrained Estimation of Distribution Algorithm (CDA), which combined the efficiency of a wrapper technique with a search directed toward small feature subsets. We showed that this technique performed as good as, and sometimes better than the traditional approaches, while heavily reducing the running time needed. A second new technique (EDA-based feature ranking) focused more on the use of feature selection techniques to gain insight in the processes that generated the data. We showed that this method naturally extends the use of population-based search methods toward obtaining feature weights and a feature ranking. This weighting method proved to be a useful tool for the classification problems, addressed in the second part of the work.

Chapter 4 opened the door to the second, more applied part of our work. In this part, we turned our attention to the problem of gene prediction, focusing on a well-chosen subset of classification problems that are part of the job. The chapter introduces the reader to the concept of gene prediction, starting with an explanation of the biological processes involved in gene transcription and translation. The remainder of the chapter describes the techniques that are nowadays commonly used in gene prediction, preparing the reader for the next chapters. In these chapters, we focused on three subtasks of gene prediction, taking a feature selection tour to analyse the processes in greater detail.

The first of these tasks deals with the problem of coding potential prediction, forming the subject of chapter 5. In this chapter, we combined the use of the Fourier transform with supervised learning. Whereas the traditional techniques for doing this only take into account the signal-to-noise ratio of the peak at frequency 1/3, our analysis showed that better classification results could be obtained using a combination of features based on the Fourier transform. A second research topic in this chapter was the extension toward including all frequencies in the spectrum. Combining these features with feature selection revealed - somewhat surprisingly - that only two frequencies are of key importance: frequency 0 and 1/3. The frequency 0 has a clear biological meaning and represents the compositional information in the sequence. Our results showed that adding such information significantly enhances the correct identification of coding regions. A last research topic in this chapter was the comparison of our technique with a state-of-the-art model for coding potential prediction, the Interpolated Markov Model (IMM). Using a ROC-analysis we showed that our method significantly outperformed the IMM when dealing with short sequences.

A second problem we tackled was the identification of splice sites, the main theme of chapter 6. For both donor and acceptor splice site prediction we showed how feature selection could both improve classification accuracy and human understandability, using our newly developed feature selection schemes. An important milestone in this chapter was the identification of a new, biologically motivated feature for acceptor prediction: AG-scanning. Furthermore we used the results obtained in the previous chapter as additional features for
splice site prediction, heavily reducing the number of features needed to describe donor and acceptor prediction. This led to a core set of features for donor and acceptor prediction. An interesting result was that the addition of features that capture the coding potential on both sides of the splice site contributes in a different way to donor and acceptor prediction. While for donor prediction, including such features proves to be essential, these features do not contribute much for acceptor prediction, where the major contribution comes from features related to AG-scanning.

The final trip of our feature selection journey concerned the detection of translation initiation sites (TIS, chapter 7). In a fashion, similar to the prediction of splice sites, we set up an iterative framework of feature construction and selection. This led to the identification of both a number of known and new features, such as the presence of ATG before the real start codon, in-frame stop codons downstream of the start codon, and compositional information related to the presence of UTR upstream of the start codon. Combining all feature selection efforts, this led to the identification of a small subset of core features for TIS prediction, significantly outperforming the basic, position dependent nucleotide features.

To summarize, this work studied the effect of feature selection techniques for classifying nucleic acid sequences. The scientific contribution of this study comprises several aspects. The first aspect concerns the algorithmical part of the work, where we made advances in the field of pattern recognition, developing new methods for feature selection, feature ranking and feature weighting. A second aspect of the work is the application of these techniques to the task of gene prediction. For a chosen subset of classification problems within the gene prediction task, we showed how feature selection techniques can be used to improve classification performance and to extract new knowledge about the biological processes we are modelling. Furthermore, we obtained some more general insights in the application of feature selection techniques. We showed that different techniques might select different features, and that care has to be taken when interpreting the results obtained by feature selection. A robust way of obtaining reliable domain knowledge is to compare different feature selection techniques and different classifiers, and to identify common patterns in the features that are selected as relevant.

### 8.2 Reflections and future prospects

Where is feature selection for gene prediction heading toward? The answer to this question is concealed by a number of peculiarities of the combination of feature selection and biology [Saey et al., 2004b]:

- the difficulty of appropriately approximating the “true” biological features by (a set of) experimental features,
8.2. REFLECTIONS AND FUTURE PROSPECTS

The RNA secondary structure clearly demonstrates potential dependencies between distant parts of the sequence.

- the existence of many correlations between parts of the sequence,
- the diversity of the various subtasks involved in gene prediction,
- the modularity of the gene prediction problem.

Biology is not an exact science. Although already a number of important mechanisms regarding transcription and translation have been discovered, many things still remain unknown. A problem related to the uncertainty about the underlying biological processes is how to map biological features to experimental features. By biological features, we here mean the true, but largely unknown features of molecules and proteins responsible for the underlying process. Based on biological knowledge and statistical analyses of data, we can construct approximations of the biological features, and use them to train classification models, hence the term experimental features.

An additional problem in the quest for appropriate experimental features is the fact that many correlations exist between parts of the genomic sequence. In our work, we made a first attempt to capture these dependencies at the feature level. Examples of these are the Fourier-based features, position dependent dinucleotides and trinucleotides, and position invariant 3-mers. An interesting way to go would be the extension of these features to more distant correlations, e.g. extended binding correlations between non-adjacent nucleotides, and correlations related to the structural properties of DNA and RNA. An example of such structural properties is the secondary structure of RNA. The secondary structure describes the sequence using structural elements like loops and stems. This is illustrated in Figure 8.1 where a structure with three loops and two stems is shown. From this figure it can be easily seen how dependencies between distant parts of the sequence may exist.

As explained earlier, gene structure prediction consists of a number of subtasks. Each subtask models a certain underlying, biological process, and these
processes are very different from each other, turning the subtasks into very diverse problems. As a consequence, each of the subtasks will have its own type of features, and features that might be useful for one problem, may turn out useless for another one. Therefore, each of these sub problems has to be tackled individually, extracting potentially useful features and subsequently applying feature selection techniques, tuning the feature set to the problem at hand.

The modularity of the gene prediction problem suggests another useful line of further research. At the level of the complete gene structure, it may well be that there exist some dependencies between features of different subtasks. Such higher level correlations would add a new layer of complexity to the feature selection process.

Related to this, we can also consider feature selection evaluated at the gene level, because it might be that features that make up a good stand-alone splice site predictor, are not necessarily those features that improve gene prediction. Although it is likely that they do, looking at a higher level might eliminate even more features.

All these topics suggest useful directions for further research on feature selection within the gene prediction setting. More general extensions are the combination of different classifiers for each subtask (ensemble learning), and the construction of more specific models such as different types of splice sites (UTR, first exon).
8.2. REFLECTIONS AND FUTURE PROSPECTS
Nederlandstalige samenvatting

Inleiding

Recente ontwikkelingen in de biotechnologie zorgden voor een ware explosie aan genetische informatie. Een logische stap die volgt op de generatie van de data is de analyse en interpretatie ervan. Om zulke grote hoeveelheden genomische data te interpreteren schieten de klassieke biologische methoden echter tekort. Bovendien noopt de overvloed aan data ons tot het automatiseren van bepaalde stappen in het interpretatieproces, waardoor men een beroep dient te doen op wiskundige technieken om deze data te analyseren (Hoofdstuk 1).

Classificatiemethoden vormen een belangrijk onderdeel van deze wiskundige technieken (Hoofdstuk 2). Deze methoden kunnen aangewend worden om biologische sequenties van elkaar te onderscheiden met betrekking tot hun functie of structuur. Uitgaande van een aantal beschrijvende kenmerken van de sequentie, zal een classificatiemethode dan een wiskundig model opstellen, dat kan gebruikt worden om de sequenties van elkaar te onderscheiden. De classificatiemodellen die we tijdens dit werk gebruikten waren de Naive Bayes Methode (NBM), de beslissingsboom C4.5 en de lineaire Support Vector Machine (SVM).

Een voorbeeld van zo’n classificatie is de opdeling van DNA sequenties, enerzijds in sequenties die genen bevatten en anderzijds sequenties die geen genen bevatten. Aangezien de relatie tussen de sequentiedata en het gedrag van de onderliggende biologische processen echter nog niet éénduidig bepaald is, zijn ook de beschrijvende kenmerken die cruciaal zijn voor een goede classificatie niet altijd geweten. Daarom worden de classificatiemethoden vaak overstelpt met vele beschrijvende kenmerken van genomische sequenties, in de hoop dat de kenmerken die relevant zijn voor de verklaring van het geobserveerde gedrag aanwezig zijn.

Dit werk behandelt de methoden die kunnen gebruikt worden om deze relevanterekenmerken te ontdekken in nucleïnezuursequenties (DNA,RNA). De bijdragen
Kenmerkselectie voor classificatie

van dit werk situeren zich op twee gebieden: enerzijds de meer algoritmische kant, met de ontwikkeling van nieuwe methoden voor kenmerkselectie, en anderzijds de toepassing van deze methoden op de verschillende classificatieproblemen die opduiken bij het automatisch herkennen van genen (genpredictie). Voorts toont dit werk ook aan dat voorzichtig dient omgesprongen te worden met de resultaten die bekomen worden d.m.v. kenmerkselectietechnieken. Verschillende selectietechnieken selecteren immers niet noodzakelijk dezelfde kenmerken, wat de interpretatie van de resultaten bemoeilijkt. Het vergelijken van verschillende selectietechnieken in combinatie met verschillende classificatiemodellen is dus een noodzaak om tot betrouwbare resultaten te komen.

Kenmerkselectie voor classificatie

In vele onderzoeksdomeinen worden gegevens gegenereerd die door een groot aantal kenmerken beschreven worden. Enkele voorbeelden hiervan zijn domeinen zoals beeldverwerking, tekstclassificatie en de biologische en biomedische wetenschappen. Om efficiënt met deze gegevens om te kunnen gaan worden technieken gebruikt die het aantal kenmerken (de dimensionaliteit van de data) reduceren. Technieken om de dimensionaliteit van data te reduceren kunnen ruwweg in drie klassen onderverdeeld worden: kenmerkselectietechnieken, projectietechnieken (zoals principal component analysis), en comprimeringstechnieken (Hoofdstuk 3). Kenmerkselectietechnieken verschillen van de andere klassen in de manier waarop ze de data reduceren. Daar waar bij projectie- of comprimeringstechnieken de originele kenmerken niet behouden blijven, maar vervangen worden door nieuwe kenmerken (bijvoorbeeld lineaire combinaties van de originele kenmerken) worden bij kenmerkselectie de originele kenmerken niet veranderd. De reductie van de dimensionaliteit gebeurt enkel door het selecteren van een deelverzameling van de originele kenmerken.

Het selecteren van een (kleine) verzameling relevante kenmerken heeft een aantal voordelen. De voornaamste hiervan zijn:

1. de mogelijkheid om, door het verwijderen van overtollige of gecorreleerde kenmerken, een betere classificatie te bekomen met een beperkte verzameling kenmerken,

2. een betere kost-kwaliteit verhouding: een beperkte verzameling van kenmerken geeft aanleiding tot snellere classificatiemodellen (betere tijdscomplexiteit) en tot een vermindering van de benodigde opslagruimte (betere geheugencomplexiteit),

3. het verwerven van inzicht in de processen die door de gegevens beschreven worden.

Technieken voor kenmerkselectie kunnen in drie klassen worden onderverdeeld: filtertechnieken (filter methods), verpakkingsotechnieken (wrapper methods) en ingebedde technieken (embedded methods).
Filtertechnieken worden gekenmerkt door het feit dat ze onafhankelijk zijn van een bepaald classificatiemodel. Om een onderscheid tussen relevante en irrelevante kenmerken te maken baseren deze methoden zich enkel op de intrinsieke karakteristieken van de beschikbare gegevens. De meeste filtermethoden gaan als volgt te werk. Voor elk kenmerk wordt een relevantiemaat (score) berekend die aanduidt in hoeverre het beschouwde kenmerk bijdraagt tot een goede classificatie (onderscheidend vermogen van het kenmerk). Vervolgens worden al deze waarden gesorteerd, en worden kenmerken met een lage score verwijderd. Dit kan bijvoorbeeld gebeuren door een bepaalde drempelwaarde voor de score op voorhand vast te leggen, of door een limiet te stellen op het aantal te behouden kenmerken. Eens de deelverzameling van relevante kenmerken bepaald is, kan in een volgende stap een classificatiemodel naar keuze getraind worden, gebruik makend van de geselecteerde kenmerken. Voordelen van filtertechnieken zijn dat ze onafhankelijk zijn van een classificatiemodel, en dat de scores zeer snel berekend kunnen worden, waardoor zeer grote hoeveelheden gegevens kunnen geanalyseerd worden. Een belangrijk nadeel van deze methoden is echter dat de scores in de meeste gevallen slechts voor elk kenmerk afzonderlijk berekend worden, zodat geen rekening gehouden wordt met eventuele interacties tussen kenmerken.

Een alternatieve manier om een deelverzameling van relevante kenmerken te ontdekken is het gebruik van de zogenaamde verpakkingstechnieken (wrapper methods). In tegenstelling tot filtertechnieken zijn verpakkingstechnieken specifiek voor een bepaald classificatiemodel. Het idee hierachter is dat op deze manier de interactie tussen een specifiek model en een specifieke gegevensverzameling beter kan bemeten worden, zodat betere resultaten bekomen kunnen worden. Daartoe wordt het classificatiemodel “verpakt” in een zoekstrategie. Het doel van deze zoekstrategie is de ruimte, bestaande uit alle mogelijke deelverzamelingen van kenmerken te doorzoeken, met het doel een optimale deelverzameling te vinden, die specifiek is voor het gekozen classificatiemodel en de gegevensverzameling. Omdat de omvang van deze zoekruimte exponentieel toeneemt met het aantal kenmerken worden veelal heuristische methoden gebruikt. Veelgebruikte methoden hierbij zijn traditionele sequentiële methoden zoals voorwaartse of achterwaartse selectie (diepte-eerst) en meta-heuristische technieken zoals genetische algoritmen. Het zoekalgoritme zal dan de evaluatie van een deelverzameling van kenmerken door het classificatiemodel gebruiken als een performantiemaat om tot een optimale deelverzameling te komen. Het voordeel van verpakkingstechnieken is het potentieel om betere oplossingen (d.w.z. relevantere deelverzamelingen van kenmerken) te bekomen dan filtertechnieken, mede door het feit dat eventuele interacties tussen kenmerken nu wel in beschouwing worden genomen, omdat gebruik wordt gemaakt van een classificatiemodel. Een nadeel echter van deze methode is het feit dat, voor iedere deelverzameling van kenmerken die door de zoekstrategie gegenereerd wordt, een classificatiemodel dient getraind en getest te worden, wat een hoge
Kenmerkselectie voor classificatie

uitvoeringskost met zich meebrengt.

Een laatste klasse van kenmerkselectietechnieken zijn de ingebedde technieken (embedded methods). In tegenstelling tot de vorige twee methoden is het principe van kenmerkselectie bij deze methoden geïncorporeerd in het classificatiemodel. Dit gebeurt door de parameters van het classificatiemodel meteen te gebruiken om te beslissen welke kenmerken al dan niet behouden dienen te blijven. Enkele voorbeelden van zulke methoden zijn het snoeien van beslissingsbomen en het gebruiken van de gewichten van een lineaire discriminantfunctie om kenmerken met een laag gewicht te elimineren.

In het kader van dit werk werden twee nieuwe technieken voor kenmerkselectie ontworpen. Beide technieken vinden hun oorsprong in de aard van de biologische classificatieproblemen die tijdens dit werk behandeld werden. Deze problemen worden enerzijds gekenmerkt door een zeer groot aantal beschrijvende kenmerken (gaande van enkele honderden tot enkele duizenden), waarvan geweten is dat een groot aantal waarschijnlijk irrelevant of redundant zijn. Het is echter niet geweten welke van deze kenmerken relevant zijn en welke niet. Enerzijds worden we aldus geconfronteerd met een groot aantal kenmerken, wat het gebruik van filter methoden rechtvaardigt. Anderzijds is er echter ook de wetenschap dat waarschijnlijk betere kenmerken kunnen gevonden worden met inpakkingsmethoden of ingebedde technieken.

Een eerste nieuwe techniek die in het raam van dit onderzoek ontwikkeld werd was het beperkte distributieschattingsalgoritme (Constrained Estimation of Distribution Algorithm, CDA). Dit algoritme combineert de efficiëntie van inpakkingsmethoden gebaseerd op Estimation of Distribution Algorithms (EDA, een veralgemening van genetische algoritmen) met de wetenschap dat slechts een beperkte verzameling van kenmerken relevant is. Daartoe werd een beperking ingevoerd op de maximale grootte van een verzameling geselecteerde kenmerken. Het invoeren van deze beperking zorgt niet alleen voor een grote reductie van de te doorzoeken ruimte, maar geeft ook aanleiding tot een aanzienlijke tijdwinst, vermits de tijd die nodig is om een classificatiemodel te trainen voornamelijk afhangt van het aantal beschrijvende kenmerken van de instanties.

Een tweede techniek die ontwikkeld werd spitst zich meer toe op het vermogen van kenmerkselectietechnieken om inzicht te verwerven in de (biologische) processen die door de gegevens beschreven worden. Hierbij dient voornamelijk opgemerkt te worden dat de selectie van één enkele deelverzameling van kenmerken eerder een statisch beeld geeft van het hele selectieproces. Een belangrijke stap om meer inzicht te verwerven in het gemodelleerde probleem is een dynamische analyse van de context van deze deelverzameling. Belangrijke vragen zoals welke van deze kenmerken nog verwijderd kunnen worden zonder dat de performantie al te erg beïnvloed wordt blijven immers onbeantwoord in dit statisch beeld, doch bevatten zeer bruikbare informatie.
De uitbreiding naar een meer dynamisch beeld van het selectieproces kan in twee stappen verdiept worden. Een eerste stap is het ordenen van de kenmerken van minst relevant tot meest relevant. Een tweede stap, die nog meer informatie geeft, is het toekennen van gewichten aan de kenmerken.

In de context van een meer dynamische analyse van het selectieproces werd een nieuwe inpakmethode beschreven die toelaat zowel een ordening als een weging van de kenmerken te berekenen. Wederom maakten we gebruik van EDA om tot een ordening en weging te komen. De nieuwe techniek bouwt voort op het feit dat EDA een populatie-gebaseerde methode is. Dit wil zeggen dat we niet slechts één oplossing tegelijkertijd beschouwen, maar een verzameling van oplossingen. In een iteratief proces van selectie en modellering wordt deze verzameling oplossingen dan geleidelijk verbeterd, totdat convergentie optreedt. We toonden aan dat een partiële convergentie van de populatie ideaal was om tot een goede ordening van kenmerken te komen. Voorts legden we ook de basis om gewichten af te leiden uit de probabiliteiten die tijdens het EDA-proces bekomen werden. Een mogelijke uitbreiding die we daarbij suggereerden was het uitbreiden van de wegingsmethode van individuele kenmerken naar deelverzamelingen van kenmerken (feature subset weighting).

Deze wegingsmethode vormde de basis om nieuwe patronen te ontdekken voor verscheidene biologische classificatieproblemen.

In het kader van dit werk werd ook een nieuw softwarespakket voor kenmerkselectie ontwikkeld (FeaST). Dit pakket werd speciaal ontworpen om efficiënt om te gaan met grote hoeveelheden data met veel beschrijvende kenmerken. Alle nieuwe methoden voor kenmerkselectie werden in dit pakket geïmplementeerd, alsook een representatief aantal reeds bestaande technieken.

Classificatie van nucleïnezuursequenties

Een groot deel van de biologische data die in de afgelopen jaren gegenereerd werd bereikt ons in de vorm van nucleïnezuursequenties: het dubbelstrengige DNA en het enkelstrengige RNA. Omdat het DNA in elk organism de drager is van alle genetische informatie spitst veel onderzoek zich toe op het ontwikkelen van de mechanismen die te maken hebben met het coderen en vertalen van deze informatie.

Een belangrijk onderdeel van dit onderzoek is de zoektocht naar de exacte locatie en structuur van de genen op het DNA. Het is immers geweten dat bij vele organismen een groot deel van het DNA niet meer actief is, en als evolutionaire ballast uit het verleden kan beschouwd worden. Slechts enkele percenten van de totale DNA-sequenti bevatten in die gevallen genen, en het zoeken naar deze breinaalden in een hooiberg is dan ook een uitdagend patroonherkenningsprobleem.

De automatische herkenning van genen in DNA is tijdens de laatste decennia uitgegroeid tot een belangrijk onderzoeks domein binnen de bioinformatica.
Kenmerkselectie voor herkenning van coderende gebieden

(Hoofdstuk 4). Een verklaring hiervoor is te vinden in het feit dat de herkenning van genen de basis vormt voor een hele waaier aan verdere analyses, zoals vergelijkende studies tussen organismen, evolutie, en interactie tussen verschillende genen (genetische netwerken). Bovendien leert het bouwen van modellen voor genpredictie ons ook heel wat over de onderliggende biologische mechanismen die te maken hebben met transcriptie en translatie.

Het herkennen van genen is een modulair probleem, dat kan opgedeeld worden in een aantal deeltaken, waarvan de resultaten uiteindelijk dienen gecombineerd te worden om tot een consistente genstructuur te komen. Deze deeltaken kunnen ruwweg in twee klassen verdeeld worden: enerzijds het detecteren van globale sequentiekenmerken zoals coderende gebieden of promotorgebieden, en anderzijds het ontdekken van lokale signalen in het DNA (functionele sites). In dit werk ging onze aandacht naar de belangrijkste van deze deeltaken: het detecteren van coderende gebieden, het detecteren van splice sites (overgangen tussen coderende en niet-coderende gebieden), en het vinden van het begin van het coderende gebied van een gen (start-codon, translation initiation site).

Kenmerkselectie voor herkenning van coderende gebieden

Een eerste voorwerp van onze studie vormde de toepassing van kenmerkselectietechnieken op het herkennen van coderende gebieden (Hoofdstuk 5). In dit deel van het onderzoek maakten we gebruik van kenmerken die afgeleid werden uit de Fourier-transformatie van het DNA signaal. Een eerste onderzoekspunt hierbij vormde de uitbreiding van het aantal kenmerken die gebruikt kunnen worden om de sequentie te beschrijven. Daar waar de traditionele technieken op basis van de Fourier-transformatie enkel de signaal-ruisverhouding van de frequentie $\frac{1}{3}$ beschouwen, toonde onze analyse aan dat classificatiemodellen die beschikken over een meer gedetailleerde verzameling van kenmerken (19 kenmerken in plaats van 1) in staat zijn om betere resultaten te bekomen. Op deze 19 kenmerken pasten we vervolgens kenmerkselectie toe, en kwamen tot de conclusie dat in alle gevallen een deelverzameling van 5 kenmerken volstond om een classificatie te bekomen die even goed was als zouden we de volledige verzameling van 19 kenmerken gebruiken. Het meest belangrijke kenmerk bleek steeds de signaal-ruisverhouding van de frequentie $\frac{1}{3}$ te zijn, wat de aanpak van de traditionele technieken bevestigt.

In een tweede onderzoekspunt stelden we ons de vraag of een uitbreiding naar
alle frequenties zou resulteren in een betere classificatie. Daartoe werd voor elke frequentie het spectrum van de 19 kenmerken berekend, wat voor sequenties van lengte 60 en 120 resulteerde in een verzameling van 589 respectievelijk 1159 kenmerken. Voor elk van deze sequenties werden verschillende combinaties van classificatiemodellen en kenmerkselectietechnieken geëvalueerd. Opnieuw kwamen we tot de conclusie dat het overgrote deel van deze kenmerken kon geëlimineerd worden, zonder aan onderscheidend vermogen in te binden. Een nader onderzoek van de relevante kenmerken bracht aan het licht dat slechts twee frequenties van belang zijn: de eerder vermelde frequentie 1/3 die de organisatie in triplassen van coderende gebieden vat en de frequentie 0. De - op het eerste zicht verrassende - aanwezigheid van de frequentie 0 heeft echter een duidelijke biologische betekenis. Deze frequentie stelt immers de compositionele informatie van de sequentie voor (o.a. het aantal A’s, T’s, C’s en G’s). Ons onderzoek toonde aan dat het toevoegen van zulke informatie een belangrijke bijdrage levert tot het correct herkennen van coderende gebieden.

Een laatste punt in het onderzoek naar de classificatie van coderende gebieden was een vergelijkende studie tussen een geavanceerd classificatiemodel (een 8ste-orde Interpolated Markov Model, zoals geïmplementeerd in het genpredictieprogramma Glimmer) en de combinatie van NBM, SVM en C4.5 met de geoptimaliseerde verzameling kenmerken op basis van de Fourier transformatie. Aangezien het herkennen van coderende gebieden moeilijker wordt naarmate de sequenties die onderzocht worden korter worden, spitsten we onze aandacht toe op het herkennen van korte coderende sequenties, bestaande uit respectievelijk 60 en 120 baseparen. Een ROC-analyse toonde aan dat zowel voor sequenties van lengte 60 als lengte 120 betere resultaten konden bekomen worden met de Fourier-kenmerken wanneer zowel de frequenties 0 als 1/3 in rekening werden gebracht. Deze analyse toonde ook duidelijk aan dat de methode op basis van Fourier-kenmerken een potentiële goede kandidaat vormt als deelcomponent voor de herkenning van korte coderende sequenties in een genpredictiesysteem. Suggesties voor verder onderzoek zijn het identificeren van coderende gebieden in ESTs (expressed sequence tags) en het detecteren van het reading frame in coderende gebieden.

Kenmerkselectie voor classificatie van splice sites

Een tweede probleem waar we het gebruik van kenmerkselectietechnieken op onderzochten was het identificeren van de correcte overgangen tussen introns en exons, de zogenaamde splice sites (Hoofdstuk 6). Voor elk van de twee types splice sites (donor sites en acceptor sites) volgden we een iteratief proces van kenmerkselectie en kenmerkconstructie, wat ons toeliet om zowel betere classificatieresultaten te bekomen als meer inzicht te verwerven in de kenmerken die van belang zijn bij splicing.

Als eerste onderzochten we het effect van kenmerkselectie op acceptorherkenning. Hierbij vertrokken we van een beperkte verzameling kenmerken die de
Kenmerkselectie voor classificatie van splice sites

aan- of afwezigheid van nucleotiden in de locale context rond de acceptor site voorstellen. Deze verzameling van kenmerken werd in een eerste fase uitgebreid met positie-invariante kenmerken (woorden van lengte 3), en vervolgens ook met positie-afhankelijke dinucleotiden. De toepassing van kenmerkselectietechnieken leerde dat een groot deel van deze kenmerken kon geëlimineerd worden, en leidde tot een basisverzameling van relevante kenmerken: de voor-naamste hiervan zijn de naburige nucleotiden rond de acceptor site, de codon bias in het coderende gedeelte (exon), en het pyrimidine-rijk gebied tussen het branch point en de acceptor. Voorts brachten deze technieken ook een nieuw, biologisch relevant kenmerk aan het licht: AG-scanning. Dit kenmerk was tot op heden nog niet beschreven in de literatuur over acceptordetectie, en toont het belang aan van het feit dat in echte acceptors gewoonlijk slechts weinig AG-dinucleotiden voorkomen in het gebied tussen het branch point en de acceptor. Voorts toonden we ook aan dat de classificatiemodellen NBM en C4.5 aanzienlijk verbeterd konden worden door het gebruik van kenmerkselectie. In een laatste fase combineerden we een deelverzameling van de bovenvermelde kenmerken met enkele nieuwe, op maat van acceptors gesneden kenmerken: AG- en TG-scanning, branch point-kenmerken, dinucleotide percentages en de Fourier-kenmerken die het al dan niet coderend karakter van de context rond de splice site modelleren. Dit leidde uiteindelijk tot een verzameling van 875 kenmerken, waarvan er d.m.v. kenmerkselectie een 90-tal kon gedeclareerd worden als basisverzameling van relevante kenmerken voor acceptor predictie. We toonden ook aan dat met deze verzameling van basiskenmerken een aanzienlijke winst in classificatieperformantie kon bekomen worden.

Een zelfde aanpak werd gevolgd in het geval van de detectie van donor splice sites. Ook hier werd duidelijk dat betere resultaten bekomen konden worden door het toepassen van kenmerkselectie. Een opvallend nieuw kenmerk dat uit onze analyse naar voren kwam was een duidelijke afwezigheid van de nucleotide G in het introngebied dat de donor site flankeert. Ook in dit geval werd een verzameling kenmerken voor donor sites op maat gemaakt, waarvan het grootste deel door kenmerkselectie geëlimineerd kon worden, wat wederom resulteerde in een basisverzameling van kenmerken, dit maal specifiek voor donorpredictie.

Een nadeel van het gebruik van lineaire modellen (NBM, lineaire SVM) voor splice site predictie is het feit dat correlaties tussen nucleotiden niet gemedeleerd kunnen worden. Het combineren van hogere-orde classificatiemodellen heeft echter als nadeel dat het kenmerkselectieproces veel langer zal duren, wat in het geval van grote gegevensverzamelingen kan leiden tot onpraktisch lange berekeningen. Om het gebruik van lineaire modellen toch te kunnen combineren met het modelleren van interacties tussen nucleotiden werd de volgende kunstgreep toegepast. In plaats van de detectie van correlaties tussen nucleotiden aan het classificatiemodel over te laten werden deze correlaties meteen op het niveau
van de kenmerken geïncorporeerd. Voorbeelden hiervan waren de correlaties tussen naburige nucleotiden zoals de positie-afhankelijke dinucleotiden of trimucleotiden. Het exploreren van andere vormen van correlaties tussen nucleotiden vormt zeker en vast een boeiend thema voor verder onderzoek, denken we maar aan het in rekening brengen van interacties die te maken hebben met het modelleren van de secundaire structuur van RNA.

Kenmerkselectie voor herkenning van start-codons

Een laatste halte in onze zoektocht naar relevante kenmerken in het kader van gendetector was het herkennen van start-codons (translation initiation sites, Hoofdstuk 7). Ook in dit geval volgden we een iteratieve procedure bestaande uit opeenvolgende stappen van kenmerkselectie en -constructie. Een berekende experiment was het toevoegen van kenmerken die correlaties tussen drie opeenvolgende nucleotiden modelleren. Het toepassen van kenmerkselectie op deze kenmerken toonde aan dat de aanwezigheid van een start-codon (ATG) in de regio voorafgaand aan het echte start-codon (UTR) een zeer bepalende factor was in het correct classificeren van start-codons. Voorts bleek ook dat het voorkomen van in-frame stop-codons (vooral de twee stop-codons TAA en TGA) een belangrijke discriminerende functie heeft.

Een tweede bevinding die uit onze analyse naar voor kwam was het feit dat de kenmerkselectie karakteristieken van het onvertaalde gebied (UTR) voorafgaand aan het start-codon ontdekte. Als voornaamste kenmerk konden we hieruit afleiden dat GC-rijkheid een belangrijke rol speelt bij het onderscheiden van start-codons.

Ook voor dit probleem werd een beperkte verzameling kenmerken op maat gemaakt. Dit gaf aanleiding tot een verzameling van 762 kenmerken. Het toepassen van kenmerkselectie op deze verzameling toonde aan dat deze kon gereduceerd worden tot ongeveer 190 kenmerken, zonder dat de performantie daalde. Aldus werd een basisverzameling van relevante kenmerken voor het herkennen van start-codons geconstrueerd.

Conclusie

In dit werk voerden we een grondige studie uit van kenmerkselectiemethoden, toegepast op de verschillende classificatieproblemen die deel uitmaken van het automatisch herkennen van genen in nucleïnezuursequenties. De wetenschappelijke bijdrage van dit werk situeert zich op twee gebieden. Enerzijds was er het meer algoritmische luik, waarin we nieuwe technieken voor het selecteren, sorteren en wegen van kenmerken ontwikkelden. Anderzijds was er de meer toegepaste kant, waarbij we ons concentreerden op drie classificatieproblemen die centraal staan in de automatische annotatie van genen in nucleïnezuursequenties: het herkennen van coderende gebieden, splice sites en start-codons. Voor elk van deze classificatieproblemen werd een raamwerk voor kenmerkselectie ontwikkeld, waarbij we zowel oog hadden voor het bouwen van betere
modellen als voor het ontdekken van relevante kenmerken. Onze resultaten toonden aan dat het toepassen van kenmerkselectie in de meeste gevallen leidde tot performantere modellen, waarbij in sommige gevallen zelfs betere resultaten bekomen werden dan met de huidige, geavanceerde technieken die gebruikt worden in genherkenning. Voorts gaf de combinatie van complexere kenmerken en kenmerkselectie ook aanleiding tot de ontdekking van nieuwe inzichten. Enkele voorbeelden hiervan zijn het belang van compositionaliteit van de herkenning van coderende gebieden, het ontdekken van scanning mechanismen (AG-scanning in het geval van acceptorherkenning en ATG-scanning in het geval van detectie van het start-codon) en UTR-compositie (start-codon).

Dit werk legde ook de basis voor verder onderzoek naar relevante kenmerken in nucleïnezuursequenties. Hierbij denken we vooral aan het toevoegen van complexere kenmerken die inherent reeds bepaalde interacties tussen gedeelten van de sequentie modelleren. Om efficiënt om te gaan met dit groeiend aantal beschrijvende kenmerken werd in het kader van dit werk ook een nieuw softwarepakket ontwikkeld (FeaST). Dit pakket integreert verschillende classificatiemodellen en kenmerkselectiemethoden, en werd speciaal ontworpen om grote datasets te analyseren.
Glossary

5′/3′
Nucleotides are connected by a phosphodiester bond between the 5′ carbon of the ribose sugar moiety of one nucleotide to the 3′ carbon of the sugar moiety of another nucleotide. Typically, polynucleotides (DNA, RNA) will contain a 5′ phosphate and 3′ hydroxyl terminal group. The common representation of polynucleotides is a sequence with the 5′ end at the left and the 3′ end at the right.

amino acid
Biochemical building blocks that make up proteins. Twenty amino acids are encoded by the standard genetic code.

CDS
CoDing Sequence: part of the sequence that codes for proteins.

discriminative learning
Discriminative classifiers model the posterior probability $p(c|x)$ directly: they learn the mapping between instance and class, whereas generative classification handles a more general problem.

DNA
DeoxyriboNucleic Acid: the nucleic acid which carries genetic instructions for the biological development of all cellular forms of life. During reproduction, it is replicated and transmitted to offspring. In eukaryotes, most of the DNA is found in the chromosomes, which are located in the cell nucleus. In most of its forms DNA is double stranded, forming hydrogen bonds between the bases A and T on one hand, and C and G on the other hand.

downstream
Toward the 3′ end of the sequence.

eukaryotes
Organisms where each cell contains a nucleus, e.g. plants and animals.

exons
Regions of a transcribed gene that are not spliced out and which are
retained in the final mRNA molecule as part of the protein-coding region.

**Generative learning**
Generative classifiers induce a model of the joint probability $p(x,c)$ from which we can calculate $p(c|x)$ for all $c$ in $C$. Classification is then performed by choosing the most likely class $c$.

**Introns**
Regions within a gene that do not encode part of the protein that the gene produces, and are spliced out of the pre-mRNA. Introns are not junk DNA.

**Junk DNA**
Those portions of the DNA for which no function has been identified or intuited.

**K-mer**
A word (subsequence) of length $k$.

**Nucleotide**
An organic molecule consisting of a nitrogenous heterocyclic base (a purine or a pyrimidine), a pentose sugar (deoxyribose in DNA or ribose in RNA), and a phosphate or polyphosphate group. There are four kinds of nucleotides, which are commonly referred to by the identity of their bases. These are adenine (A), thymine (T), cytosine (C), and guanine (G). In RNA, uracil (U) is the counterpart of the base thymine (T), found in DNA.

**Pre-mRNA**
Preliminary messenger RNA: a single strand of RNA, synthesized from the DNA in the nucleus of a cell during the process of transcription. The pre-mRNA is further processed by the spliceosome to produce the mature messenger RNA (mRNA) which is afterwards translated into a protein.

**Prokaryotes**
Lower organisms like bacteria and Archaeabacteria (Archaea). These organisms do not have a nucleus, and the DNA is more or less distributed throughout the cell.

**Ribosome**
An organelle that translates mRNA into a protein. Ribosomes are found in the cytosol (the internal fluid) of all cells.

**RNA**
RiboNucleic Acid: nucleic acid similar to DNA, except for the presence of an additional hydroxyl group, and the presence of the base uracil (U) instead of thymine (T). In most forms, RNA is single stranded.
RNA transmits genetic information from DNA (via transcription) into proteins (by translation).

**splice site**
Either end of an intron. The 5’ end of the intron is called the donor splice site, the 3’ end is the acceptor splice site.

**spliceosome**
A large ribonucleoprotein complex that binds to the pre-mRNA, removes introns and ligates the surrounding exons.

**transcription**
The first process during the expression of a gene. In this phase, DNA is copied to RNA by an enzyme (RNA polymerase).

**translation**
The process during which the mature mRNA is converted into a protein by the ribosome.

**upstream**
Toward the 5’ end of the sequence.
Bibliography


composition for support vector machines. Technical report, Los Alamos
National Laboratory, Los Alamos, NM 87545.

Feature subset selection by bayesian networks based optimization. Artificial
Intelligence, 27:143–164.

recognition: A review. IEEE Transactions on Pattern Analysis and Machine
Intelligence, 22(1):4–37.

chine learning practical. In B. Schölkopf, C. Burges, A. S., editor, Advances

features and the subset selection problem. In Proc. 11th Intl. Conf. on
Machine Learning, pages 121–129.

chapter Feature set search algorithms, pages 41–60. Sijthoff and Noordhoff,
Alphen aan den Rijn, Netherlands.


292, Bari, Italy.


from studies on translation. Mammalian Genome, 7:563–754.


chapter An introduction to hidden Markov models for biological sequences,

algorithms that select features for pattern classifiers. Pattern Recognition,
33:25–41.


Index

alternative splicing, 100

classification, 15
  Bayesian classifiers, 18
  likelihood, 19
  maximum a posteriori rule, 19
  priors, 19
Decision trees, 24
  impurity criterion, 25
  pruning, 27
  stopped splitting, 27
Linear Discriminant Functions, 20

classifiers
  C4.5, 27
  ID3, 25
  Naive Bayes, 20
  Support Vector Machines, 20
  kernel function, 22
  soft margin, 22
  support vector, 22

codon, 68
CP matrix, 128
crossvalidation, 29
curse of dimensionality, 35
dimensionality reduction techniques, 34
discretization, 62

entropy, 26

estimation of distribution algorithm (EDA), 43
  BMDA, 44
  BOA, 44
  constrained EDA (CDA), 43
  EBNA, 44
  UMDA, 44
eukaryotes, 66
evaluation measures, 28
  accuracy, 29
  approximate correlation, 30
  correlation coefficient, 30
  error rate, 29
  F-measure, 30
  FP rate, 29
  precision, 30
  q9 measure, 30
  recall, 29
  Receiver Operator Curve (ROC), 31
  sensitivity, 29
  specificity, 30
  TP rate, 29
exon, 66

FeaST: a feature selection toolbox, 57

feature relevance scores
  relative entropy, 37
  Euclidean distance, 37
  feature-class entropy, 37
  gain ratio, 27, 37
  information gain, 27, 37
  Kullback-Leibler divergence, 37
  mutual information, 37
  P-measure, 38
  SNR, 38, 86
  t-score, 38

feature selection, 33
  embedded methods, 47
    weighted linear SVM, 49
    weighted Naive Bayes, 47
INDEX

filter methods, 36
  Correlation-based feature selection, 39
Fourier transform, 50
Koller-Sahami algorithm, 39, 40
Markov blanket filter, 39
signal processing, 50
wrapper methods, 41
  beam search, 41
genetic algorithms, 42
  plus q take-away r, 41
randomized methods, 42
SBE, 41
sequential methods, 41
SFS, 41
feature weighting, 51
Fourier transform, 50
  Discrete Fourier transform (DFT), 50
gain ratio, 27
gene prediction, 68
  content sensors, 69
  extrinsic prediction, 69
ICM, 73
IMM, 72
intrinsic prediction, 69
MDD, 72
signal sensors, 69
WAM, 70
WMM, 70
WWAM, 71
gene prediction systems
  EuGène, 72
  GeneMark.hmm, 74
  GeneSplicer, 72
  Genie, 73
  Genscan, 74
  Glimmer, 71
  Glimmer2, 72
  MZEF, 73
  NetGene2, 73
  NetPlantGene, 73
genetic algorithm, 42
heat map, 92
information gain, 27
intron, 66
linear machine, 20
Markov models, 71
  GHMM, 75
  HMM, 74
  Markov blanket, 39
peaking phenomenon, 35
PlantGene dataset, 77
pre-mRNA, 66
prokaryotes, 66
pruning, 27
sequence logo, 77
splice sites, 69
splicing
  acceptor site, 97
  branch point, 98
  donor site, 97
  self-splicing, 98
  spliceosomal splicing, 98
start-codon, 66
statistical tests
  combined 5x2 cv F test, 32
  McNemar test, 31
stop-codon, 66
symmetrical uncertainty, 39
transcription, 66
translation, 66
untranslated region (UTR), 68

171