13 Accurate Splice Site Detection for *Caenorhabditis elegans*

Gunnar Rätsch
Sören Sonnenburg

We propose a new system for predicting the splice form of *Caenorhabditis elegans* genes. As a first step we generate a clean set of genes from available exressed sequence tags (EST) and complete complementary (cDNA) sequences. From all such genes we then generate potential acceptor and donor sites as they would be required by any gene finder. This leads to a clean set of true and decoy splice sites. In a second step we use support vector machines (SVMs) with appropriately designed kernels to learn to distinguish between true and decoy sites. Using the newly generated data and the novel kernels we could considerably improve our previous results on the same task.

In the last step we design and test a new *splice finder system* that combines the SVM predictions with additional statistical information about splicing. Using this system we are able to predict the exon-intron structure of a given gene with known translation initiation and stop codon site. The system has been tested successfully on a newly generated set of genes and compared with GenScan. We found that our system predicts the correct splice form for more than 92% of these genes, whereas GenScan only achieves 77.5% accuracy.

13.1 Introduction

Splice sites

Splice sites are locations on DNA at the boundaries of exons (which code for protein parts) and introns (which do not). The more accurately a splice site can be located, the easier and more reliable it becomes to locate the genes on DNA. For this reason, accurate splice site detectors are valuable components of state-of-the-art gene finders (Burge and Karlin, 1997; Reese et al., 1997; Salzberg et al., 1998; Delcher et al., 1999; Pertea et al., 2001). Furthermore, since ever-larger chunks of DNA are to be analyzed by gene finders, the problem of accurate splice site recognition has never been more important.

Support vector machines

SVMs (see, e.g., Vapnik, 1995; Müller et al., 2001; Schölkopf and Smola, 2002), with their strong theoretical roots, are known to be excellent algorithms for solving classification problems. They have been successfully applied to several bioinformatics problems (see, e.g., Jaakkola and Haussler, 1999; Zien et al., 2000; Brown et al., 2000; Tsuda et al., 2002a, and other chapters in this book). In this chapter we apply SVMs to two binary classification problems: the discrimination of donor sites (those at the exon-intron boundary) from decoys for these sites, and the discrimination of acceptor sites (those at the intron-exon boundary) from decoys for these sites. For this study we consider different kernels, in particular the so-called *locality improved* kernel that we proposed in Zien et al. (2000) for translation initiation site (TIS) recognition, the standard polynomial kernel, the SVM-pairwise kernel using alignment scores (Liao and Noble, 2002), the TOP kernel (related to the Fisher kernel; cf. Jaakkola and Haussler, 1999; Tsuda et al., 2002a), and, in addition, a polynomial-like kernel – the *weighted degree kernel.*

Decoys

Although present-day splice site detectors, e.g., based on neural networks or hidden Markov models (HMMs) are reported to perform at a fairly good level (Reese et al., 1997; Rampone, 1998; Cai et al., 2000), several of the reported performance numbers should be interpreted with caution, for a number of reasons. First of all, these results are based on *small* data sets of a limited number (one or two) of organisms. Now that many genomes have been fully sequenced, these results will need to be re-evaluated. Second, often only the single site prediction of acceptor and donor sites is considered, whereas the higher goal is to use it within a gene finder and it is uncertain how good the predictors perform in this setting. Third, issues in generating negative examples (decoys) were, if recognized, not adequately documented. In some cases (e.g., IP-data in Rampone, 1998), the decoy examples were chosen to be so weak that almost any reasonable method would achieve a high accuracy. In some other cases it was neither described how the decoys were chosen nor were the data sets made publicly available. Their reported performance is therefore incomparable with external studies, since the choice of data sets, in particular the decoys, can make a tremendous difference in the measured performance. Moreover, note that the choice of decoys is actually connected with the intended purpose of the system (i.e., use in a gene finding system).

In this study we put particular care into designing an appropriate splice data set for *C. elegans.* First, we derive a clean set of true splice sites from matching complete cDNA and ESTs to the genomic sequence. From them we generate a set of genes. Our decoy examples are chosen to be the sites that a splice finder asks predictions for but are actually not true sites (see details in the appendix). The generation of the data in this way ensures that the resulting classifier performs well in a setting in which a splice finder would actually use it. Previously mentioned approaches do not take this issue into account, since the training set decoys and test set decoys (when used in a splice finder) are generated from completely different probability densities. Moreover, it poses theoretical problems and often leads to a great performance loss in practice.

Splice forms

We train and evaluate the SVM on the newly generated data set. We find great performance improvements in predicting single splice sites compared to our previous study on a related *C. elegans* splice data set (Sonnenburg et al., 2002). However, the single site predictions alone do not suffice to determine the splice form (i.e., the exon-intron structure) of a gene. We therefore *design a splice finder* that uses our donor and acceptor predictions and finds a *consistent segmentation* of a given pre-messenger RNA (pre-mRNA) sequence into exons and introns. We assume here that the start (translation initiation) and end (stop codon) positions on the RNA sequence are known.[1] Our splice finder system uses additional knowledge such as intron and exon length statistics to produce biologically plausible splice forms.

A complete understanding of splice sites not only helps to *correctly* predict the spliced mRNA and thus proteins from DNA but can also be of great help in localizing genes. Actually several other sites, like translation initiation start sites and stop codons, branch sites, promoters and terminators of transcription, and various transcription factor binding sites belonging to the class of *local sites* can help to detect genes (Haussler, 1998). Compared to a gene finder that finds genes and locates their exons, our system only solves the subtask of predicting the coding parts (i.e., locating the exons within a gene) of the sequence when the start and end positions are given. Therefore we do not consider the detection of other local sites. However, the proposed system can quite easily be combined with commonly used techniques for gene finding leading to a full gene finder system.

In this chapter the main focus is on improving the signal sensor for the detection of splice sites. As a second step we use the improved sensor to accurately predict the splice form of a gene, when the start and end position of the coding region is given. In a carefully designed and fair experiment we compare the results of our splice finder with a state-of-the-art gene finder – GenScan (Burge and Karlin, 1997) – and find dramatic performance improvements.

The chapter is organized as follows: In section 13.2 we review some biological background on splicing. In section 13.3 we describe the different kernels and methods we used in this study. In section 13.4 we present our experimental results: (a) We compare the methods on the basic classification task, showing the power of kernel methods. (b) We show interesting relations of the predicted activity of a splice site to the position in the gene. (c) Finally, we compare our best methods with the state-of-the-art method GenScan and find greatly superior performance of our methods in the task of predicting the splice form of a gene. In the appendix to this chapter we describe in detail the data generation process.

[1] We do not consider splicing in the $5'$ and $3'$ untranslated regions (UTRs).

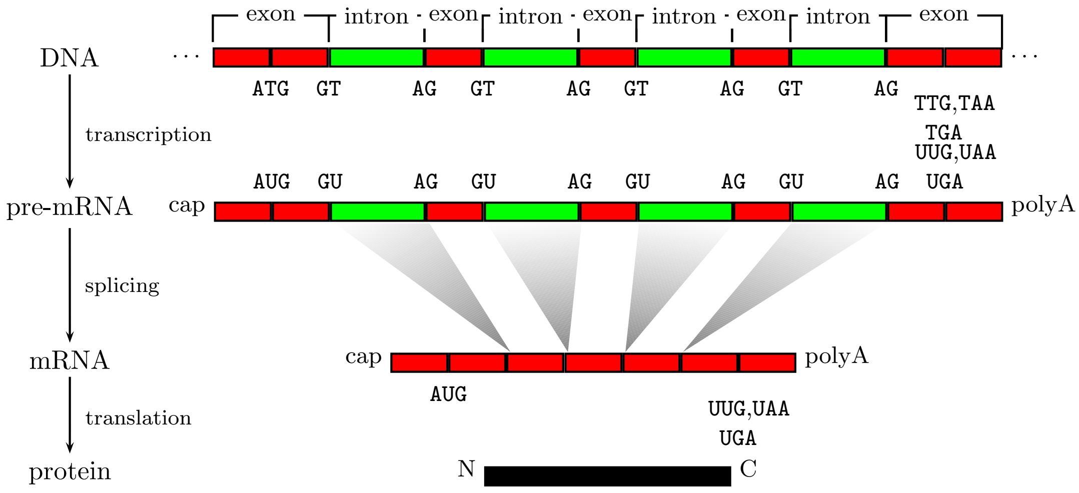


Figure 13.1 The major steps in protein synthesis. See text for details. The idea is from Chuang and Roth (2001) and Lewin (2000)

13.2 Biological Background

Each chromosome consists of coding and intergenic regions. The latter makes up most of the chromosome. But where on the chromosome are the genes located and what is a gene? A *gene* can be defined as a region of DNA that controls a certain characteristic. It corresponds to a sequence used in the production of a specific protein. While the question about the location of genes has not been sufficiently answered yet, a number of stages that are involved in the expression of genes (i.e., the process of synthesizing proteins from genes) have been identified (Lewin, 2000). These steps are carried out sequentially (see figure 13.1):

1. Activation of gene structure
2. Initiation of transcription
3. Elongation of the transcript
4. Post-processing
5. Transport of mRNA to cytoplasm
6. Translation of mRNA

It was discovered that genes are "activated" in a celltype-specific fashion. A gene may be active in cells composing one kind of tissue, but "inactive" in other kinds of tissue. When the precondition – that the gene be active – is fulfilled, it can be transcribed, that is, the process by which a copy of the gene is synthesized and which is encoded on only one strand of the double-stranded DNA (the coding strand for that gene). The copy is not DNA, but single-stranded precursor messenger

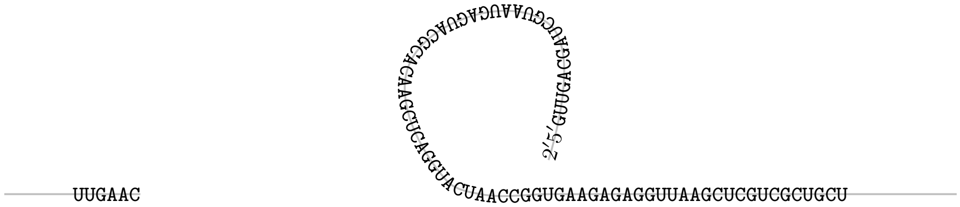

UUGAAC
2′5′
GUUGACGAUCGUAAUGCAUGACGGCACAAGCUCAGGUACUAACCGGUGAAGAGAGGUUAAGCUCGUCGCUGCU

Figure 13.2 Illustration of the splicing process. First a cut is made at the 5′ site right before the GT. Then splicing proceeds through a lariat, in which the 5′ terminus generated at the end of the intron becomes linked by a 5′-2′ bond to a nucleotide within the intron. The target nucleotide is an A in a sequence that is called the branch site (Lewin, 2000). In the second stage, a cut is made at the 3′ site. The free intron is released and both exons are joined at their ends.

ribonucleic acid (pre-mRNA). The chemistry of RNA requires that the role of thymine is taken over by the nucleotide uracil (U). For convenience, we will use uracil and thymine synonymously. The transcription starts when the enzyme RNA polymerase binds to the *promoter*, which is a special region located upstream[2] of the first nucleotide that is transcribed into pre-mRNA. From this *starting point*, RNA polymerase moves *downstream* in the 5′ → 3′ direction, continuously synthesizing pre-mRNA until a terminator sequence is reached. In the post-processing step, the pre-mRNA is transformed into mRNA. One necessary step in the process of obtaining mature mRNA is called *splicing*. The coding sequence of a eukaryotic gene is “interrupted” by noncoding regions called *introns*. A gene starts with an exon and may then be interrupted by an *intron*, followed by another exon, intron and so on until it ends in an exon. In the splicing process, introns are removed (cf. figure 13.2.)

As a result, there are two different splice sites: the exon-intron boundary, referred to as the donor site or 5′ site (of the intron) and the intron-exon boundary, that is the acceptor or 3′ site. Splice sites have quite strong consensus sequences, i.e. almost each position in a small window around the splice site is representative of the most frequently occurring nucleotide when many existing sequences are compared in an alignment. For example, the 5′ site's consensus is $\mathtt{A}_{64}\mathtt{G}_{73}\mathtt{G}_{100}\mathtt{T}_{100}\mathtt{A}_{62}\mathtt{A}_{68}\mathtt{G}_{84}\mathtt{T}_{63}$, while the 3′ site's consensus is $\mathtt{C}_{65}\mathtt{A}_{100}\mathtt{G}_{100}$, where the subscripts denote the frequency of the symbol in percent (Lewin, 2000). The dimers GT and AG can therefore be used to identify potential donor and acceptor sites. Unfortunately the pairs GT...AG occur very frequently; for example, in human DNA (which is $\approx 3 \cdot 10^9$ nucleotides in size), GT occurs about 1 billion times. For some crude estimate of, say, 10^5 genes with ten exons each, only 0.1% of the possible splice sites are *real* splice sites. One can analogously estimate the number of occurrences of AG which leads to a similar result. Therefore it is not enough to look at pairs of GT...AG, evidently there is some intrinsic property around the splice site telling the *spliceosome*, i.e. the large biological splicing apparatus consisting of an array

[2] Upstream means closer to the 5′ end, while downstream means closer to the 3′ end.

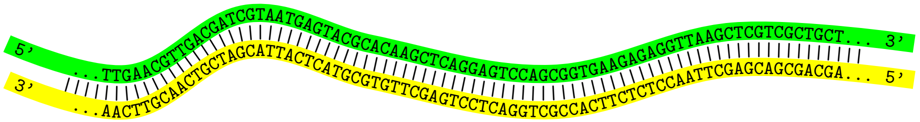

5'
...TTGAACGTTGACGATCGTAATGAGTACGCACAAGCTCAGGAGTCCAGCGGTGAAGAGAGGTTAAGCTCGTCGCTGCT...
3'
3'
...AACTTGCAACTGCTAGCATTACTCATGCGTGTTCGAGTCCTCAGGTCGCCACTTCTCTCCAATTCGAGCAGCGACGA...
5'

Figure 13.3 The two strands of DNA in an ASCII-character chain. As a result of a sequencing project, only one of these sequences is given, since adenine is always connected to thymine, and guanine to cytosine.

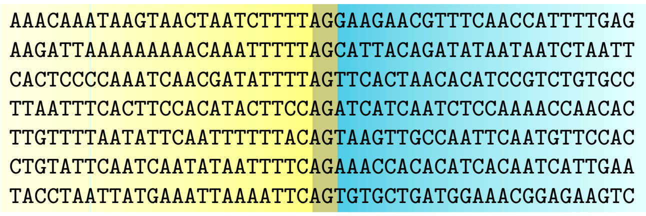
of proteins and ribonucleoproteins, to start the splicing mechanism. Neither is it known what exactly this property is, nor what the details of the reaction involving RNA and the spliceosome are. While the *canonical splice sites* `GT...AG` make up the vast majority of splice sites, other possible combinations, as, for example, `GT...TG`, have been discovered (Burset et al., 2000). We will not take these *noncanonical* sites into account, which would make splice site detection even more difficult.

What is known about splicing?

- The splicing process takes place in the *nucleus*.
- Exons are relatively short, 100 to 200 nucleotides (*nt*) while introns are often longer than 1 knt.
- An average mammalian gene has 7 to 8 exons spread over ≈ 16 knt.
- There are no reading frames in introns.
- Splice sites are generic: They do not have a specificity for individual RNA precursors, and individual precursors do not convey specific information (such as secondary structure) that is needed for splicing.
- The apparatus for splicing is not tissue-specific: RNA can usually be spliced properly by any cell, whether or not it is usually synthesized in that cell.
- Experiments show that any $5'$ splice site can in principle be connected to any $3'$ splice site, that is, only *local* information is relevant in the splicing process.
- In *higher* eukaryotes, 18 to 40 nt upstream of the $3'$ site, lies the branch site. To this site the `GU` from the $5'$ site connects to an `A` of the branch site.

Thus, the sequences needed for splicing are the short consensus sequences at the $5'$ and $3'$ splice sites and at the branch site. In higher eukaryotes mutations or deletions of the branch site result in a *proximate* $3'$ *site* to be taken. The branch site therefore identifies the $3'$ site to be used as the target for connection to the $5'$ site (Lewin, 2000), but its removal does not prevent splicing.

After pre-mRNA has been spliced to form mRNA, the splicing product is transported from the nucleus to the cytoplasm. There, in the step of translation, mRNA is read as *codons*, i.e. as triplets of nucleotides. Hence, there are three different *reading frames*, that is, ways of reading triplets of RNA (one for each of the possible start positions: $0, 1,$ or 2). Sixty-one of the $4^3 = 64$ possible codons code for 20 amino acids, while the remaining three (`UAG,UAA,UGA`) are termination codons,



```
AAACAAATAAGTAACTAATCTTTTAGGAAGAACGTTTCAACCATTTTGAG
AAGATTAAAAAAAAAACAAATTTTTAGCATTACAGATATAATAATCTAATT
CACTCCCCAAATCAACGATATTTTAGTTCACTAACACATCCGTCTGTGCC
TTAATTTCACTTCCACATACTTCCAGATCATCAATCTCCAAAACCAACAC
TTGTTTTAATATTCAATTTTTTACAGTAAGTTGCCAATTCAATGTTCCAC
CTGTATTCAATCAATATAATTTTCAGAAACCACACATCACAATCATTGAA
TACCTAATTATGAAATTAAAATTCAGTGTGCTGATGGAAACGGAGAAGTC
```

Figure 13.4 Illustration of how acceptor site examples are constructed. Windows of fixed length are taken around the splice site, while AG is aligned to be at the same position in all examples. The left part, including the AG, is intronic, while the rest is exonic.

which mark the end of a gene. The translation begins almost always at the start codon AUG, called the *translation initiation site* (TIS). However, only the minority of the codon AUG, which represents the amino acid methionine, really signals the translation initiation. SVMs have been successfully used to model this site (Zien et al., 2000). When a stop codon is reached, the translation is terminated and the sequence of amino acids – the result of the translation process – forms a long polypeptide, the *protein*.

13.2.1 Data

To gain some understanding about what the data look like, we give some examples:

A fully sequenced genome consists of all the chromosomes found in a cell of the respective organism. Each sequenced chromosome is a sequence of the characters A,C,G,T, like that in figure 13.3.

When dealing with splice sites, the examples are aligned such that AG appears at the same position in all examples, while each example consists of a fixed number of nucleotides around the site (figure 13.4).

In this chapter we consider the problem of distinguishing true splice sites from decoys. Most learning algorithms need positive as well as negative examples for learning. While it is relatively simple to extract positive examples from, for example, cDNA matches (cf. subsection 1.1.1), it is less obvious how to determine negative examples. We do this as follows: from matching cDNA to the genomic sequence we generate a set of “virtual genes” and simulate a run of a splice finder for the whole gene. Our negative examples are chosen to be the sites that any splice finder would ask predictions for, but are not true sites. A detailed explanation of how the splice data set was generated is given in the appendix. All data sets used in this study together with more information are publicly available from http://ida.first.fhg.de/splice.

13.3 Methods

In the following we describe three methods for different subtasks of splice site recognition. The *first task* is to classify a given sequence whether it be a donor or not and whether it be an acceptor site or not (two two-class problems). This is done by training Markov models (MMs) or SVMs on the training data and tuning their hyperparameters on the validation data. The *second task* is to predict the splice form for a given sequence. In this step we use the scores provided by the single site detectors (first task) for every appearing `AG` and `GT` dimer. The challenge is to find a splice form that consistently combines all predictions. It turns out that it is beneficial to combine the single site scores with available statistical information about the sequences and certain rules about the structure of the resulting splice product (e.g., open reading frames). In the second step it is assumed that the number of exons to be found is given in advance. The *third task* is to determine the numbers of exons – using the results of the first two stages for different number of exons. For a complete prediction of a splice form from a given sequence, one first computes the single site scores for each potential splice site, then builds the probabilistic model and computes the optimal splice form for various numbers of exons, and finally selects one of the splice forms (i.e., exon number) which is the result of the prediction.

13.3.1 Identifying a Single Splice Site

Machine learning classification methods aim at estimating a classification function $g : \mathcal{X} \to \{\pm 1\}$ using labeled training data from $\mathcal{X} \times \{\pm 1\}$ such that g will correctly classify unseen examples (test data). In our case, input space $\mathcal{X}$ will contain simple representations of sequences $\{A, C, G, T\}^N$, while ±1 corresponds to true splice and decoy sites, respectively. We will use the posterior log-odds of a simple probabilistic model and SVMs using different kernels as classifiers.

13.3.1.1 Posterior Log-Odds

The posterior log-odds of a probabilistic model with parameters $\boldsymbol{\theta}$ are defined by

$$f(\mathbf{x}) := \log\left(P(y=+1|\mathbf{x},\boldsymbol{\theta})\right) - \log\left(P(y=-1|\mathbf{x},\boldsymbol{\theta})\right) \tag{13.1}$$

$$= \log\big(P(\mathbf{x}|\boldsymbol{\theta}^+)\big) - \log\big(P(\mathbf{x}|\boldsymbol{\theta}^-)\big) + b, \tag{13.2}$$

Markov model

where b is the bias. We use Markov chains

$$P(\mathbf{x}|\boldsymbol{\theta}^\pm) = P(x_1,\ldots,x_N|\boldsymbol{\theta}^\pm) = P(x_1,\ldots,x_\omega|\boldsymbol{\theta}^\pm)\prod_{i=\omega+1}^{N} P(x_i|x_{i-1},\ldots,x_{i-\omega},\boldsymbol{\theta}^\pm) \tag{13.3}$$

as, for instance, described in Durbin et al. (1998). Each factor in this product has to be estimated in model training, that is, one counts how often each symbol appears at each position in the training data conditioned on every possible $x_{i-1}, \dots, x_{i-\omega}$. Then, for given model parameters $\boldsymbol{\theta}$ we have

$$P(\mathbf{x}|\boldsymbol{\theta}^{\pm}) = \theta_0^{\pm}(x_1, \dots, x_\omega) \prod_{i=\omega+1}^{N} \theta_i^{\pm}(x_i, \dots, x_{i-\omega}) \tag{13.4}$$

where $\theta_0^{\pm}$ is an estimate for $P(x_1, \dots, x_\omega)$ and $\theta_i(x_i, \dots, x_{i-\omega})$ an estimate for $P(x_i|x_{i-1}, \dots, x_{i-\omega})$. As the alphabet has four letters, each model has $(N - \omega + 1) \cdot 4^{\omega+1}$ parameters and the maximum likelihood estimate is given by

$$\theta_0(s_1, \dots, s_\omega) = \frac{1}{m + \tau} \left(\sum_{k=1}^{m} \mathbf{I}(s_1 = x_1^k \wedge \cdots \wedge s_\omega = x_\omega^k) + \tau \right)$$

$$\theta_i(s_i, \dots, s_{i-\omega}) = \frac{\sum_{k=1}^{m} \mathbf{I}(s_i = x_i^k \wedge \cdots \wedge s_{i-\omega} = x_{i-\omega}^k) + \tau}{\sum_{k=1}^{m} \mathbf{I}(s_i = x_{i-1}^k \wedge \cdots \wedge s_{i-\omega} = x_{i-\omega}^k) + 4\tau},$$

where $\mathbf{I}(\cdot)$ is the indicator function, k enumerates over the number of observed sequences m, and τ the commonly used pseudo-count (a model parameter; cf. Durbin et al., 1998). The bias b is tuned by minimizing the number of misclassifications on the validation set. Finally, $g(\mathbf{x}) = \text{sign}(f(\mathbf{x}))$ defines the classifier we obtain from the posterior log-odds.

13.3.1.2 SVM and Kernels for Splice Site Detection

As the second method we use SVMs as described in chapter 2. The generated classification function can be written as

$$g(\mathbf{x}) = \text{sign}\left(\sum_{i=1}^{m} y_i \alpha_i k(\mathbf{x}_i, \mathbf{x}) + b \right), \tag{13.5}$$

where $y_i \in \{-1, +1\}$ $(i = 1, \dots, m)$ is the label of example $\mathbf{x}_i$. The α_i's are Lagrange multipliers and b is the usual bias which are the results of SVM training. The kernel k is the *key ingredient* for learning with SVMs. It implicitly defines the feature space and the mapping Φ via

$$k(\mathbf{x}, \mathbf{x}') = \langle \Phi(\mathbf{x}), \Phi(\mathbf{x}') \rangle. \tag{13.6}$$

In the following paragraphs we describe the kernels which are used in this study.

Polynomial kernel

As the well-known (homogeneous) *Polynomial kernel* of degree d

$$k(\mathbf{x}, \mathbf{x}') = \langle \mathbf{x}, \mathbf{x}' \rangle^d.$$

is originally defined on real-valued inputs, it cannot directly be applied to discrete data, like DNA. However, a commonly used trick is to map the alphabet A,C,G,T into a binary representation: $\mathbf{x} \in \{A, C, G, T\}^N$ is represented as

$$\begin{aligned}\tilde{\mathbf{x}} = (&\mathbf{I}(x_1 = A), \mathbf{I}(x_1 = C), \mathbf{I}(x_1 = G), \mathbf{I}(x_1 = T),\\ &\mathbf{I}(x_2 = A), \mathbf{I}(x_2 = C), \mathbf{I}(x_2 = G), \mathbf{I}(x_2 = T),\\ &\ldots,\\ &\mathbf{I}(x_N = A), \mathbf{I}(x_N = C), \mathbf{I}(x_N = G), \mathbf{I}(x_N = T))^\top .\end{aligned}$$

Now we can apply the standard polynomial kernel $k(\tilde{\mathbf{x}}, \tilde{\mathbf{x}}') = \langle \tilde{\mathbf{x}}, \tilde{\mathbf{x}}' \rangle^d$. This kernel takes all correlations of matches $\mathbf{I}(\tilde{x}_i = \tilde{x}'_i)$ up to order d into account. The features used for learning are position-dependent. They carry *local and global* information about the sequence, as, for instance, any position is combined with any other position to form a feature in the kernel feature space (generated by raising the scalar product to the power of d; cf. Müller et al., 2001). Note that this kernel can be computed in an efficient manner directly on the input space by

$$k(\mathbf{x}, \mathbf{x}') = \left(\sum_{i=1}^{N} \mathbf{I}(x_i = x'_i) \right)^d . \qquad (13.7)$$

Locality improved kernel

The so-called locality improved (LI) kernel has been proved useful in the context of TIS recognition (Zien et al., 2000). It essentially works like the polynomial kernel but only considers local correlations within a small window. It is obtained by comparing the two sequences locally within a window of length $2l + 1$ around a sequence position, where one counts matching nucleotides. The resulting fraction of hits is taken to the dth power, where d reflects the order of local correlations (within the window) that we expect to be of importance:

$$\mathrm{win}_p(\mathbf{x}, \mathbf{x}') = \left(\frac{1}{2l+1} \sum_{j=-l}^{+l} \mathbf{I}(x_{p+j} = x'_{p+j}) \right)^d , \qquad (13.8)$$

where $p = l+1, \ldots, N-l$. These window scores are then summed up over the length of the sequence using a weighting w_p which linearly decreases to both ends of the sequence, that is, $w_p = \left\{ \begin{smallmatrix} p-l & p \le N/2 \\ N-p-l+1 & p > N/2 \end{smallmatrix} \right.$. Then we have the following kernel:

$$k(\mathbf{x}, \mathbf{x}') = \sum_{p=l+1}^{N-l} w_p \mathrm{win}_p(\mathbf{x}, \mathbf{x}'). \qquad (13.9)$$

The weighting allows one to emphasize regions of the sequence which are believed to be of higher importance (in our case the center, which is the location of the site). Note that the definition of the LI kernel by Zien et al. (2000) is slightly different from ours. Previously the weighting was inside the window which was not very effective. Moreover, the proposed version of the kernel can be computed $2l + 1$ times faster than the original one.

Weighted degree kernel

In a similar approach one counts the matches between two sequences $\mathbf{x}$ and $\mathbf{x}'$ between the words $\boldsymbol{u}_{\omega,i}(\mathbf{x})$ and $\boldsymbol{u}_{\omega,i}(\mathbf{x}')$ where $\boldsymbol{u}_{\omega,i}(\mathbf{x}) = x_i x_{i+1} \ldots x_{i+\omega-1}$ for all i and $1 \leq \omega \leq d$. The parameter ω denotes the order (length of the word) to be compared. The weighted degree kernel is defined as

$$k(\mathbf{x}, \mathbf{x}') = \sum_{\omega=1}^{d} w_\omega \sum_{i=1}^{N-d} \mathbf{I}(\boldsymbol{u}_{\omega,i}(\mathbf{x}) = \boldsymbol{u}_{\omega,i}(\mathbf{x}')) \tag{13.10}$$

where we chose the weighting to be $w_k = d-\omega+1$, that is, higher-order matches get lower weights. This kernel emphasizes position-dependent information and decreases the influence of higher-order matches. It can be computed very efficiently without even extracting and enumerating all words from the sequences.[3] Note that this kernel is similar to the spectrum kernel as proposed by Leslie et al. (2002), with the main difference that the weighted degree kernel uses position-specific information.

TOP kernel from Markov models

Similarly to the well-known Fisher Kernel (Jaakkola and Haussler, 1999), the main idea of the TOP kernel (cf. Tsuda et al., 2002a) is to incorporate prior knowledge via a given probabilistic model. It is derived from the tangent vectors of posterior log-odds (TOP) and especially designed for classification. It was shown to outperform the Fisher kernel on protein family classification tasks (Tsuda et al., 2002a). It is defined as $k(\mathbf{x}, \mathbf{x}') = \langle \boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}), \boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}') \rangle$, where

$$\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}) := (v(\mathbf{x}, \boldsymbol{\theta}), \partial_{\theta_1} v(\mathbf{x}, \boldsymbol{\theta}), \ldots, \partial_{\theta_p} v(\mathbf{x}, \boldsymbol{\theta}))^\top \tag{13.11}$$

and

$$v(\mathbf{x}, \boldsymbol{\theta}) = \log(P(y = +1|\mathbf{x}, \boldsymbol{\theta})) - \log(P(y = -1|\mathbf{x}, \boldsymbol{\theta})). \tag{13.12}$$

For the MM described in subsection 13.3 this kernel is particularly simple to compute. Then we have $\partial_{\theta_{i,j}} v(\mathbf{x}, \boldsymbol{\theta}) = \mathbf{I}(x_i = j)/\theta_{i,j}$ and hence the kernel is computed as

$$\begin{aligned} k(\mathbf{x}, \mathbf{x}') = {} & v(\mathbf{x}, \boldsymbol{\theta}) v(\mathbf{x}', \boldsymbol{\theta}) \\ & + \mathbf{I}(x_1 = x_1' \wedge \ldots, \wedge x_\omega = x_\omega')/(\theta_0^+(x_1, \ldots, x_\omega))^2 \\ & + \mathbf{I}(x_1 = x_1' \wedge \ldots, \wedge x_\omega = x_\omega')/(\theta_0^-(x_1, \ldots, x_\omega))^2 \\ & + \sum_{i=\omega+1}^{N} \mathbf{I}(x_i = x_i' \wedge \ldots, \wedge x_{i-\omega} = x_{i-\omega}')/(\theta_i^+(x_i, \ldots, x_{i-\omega}))^2 \\ & + \sum_{i=\omega+1}^{N} \mathbf{I}(x_i = x_i' \wedge \ldots \wedge x_{i-\omega} = x_{i-\omega}')/(\theta_i^-(x_i, \ldots, x_{i-\omega}))^2. \end{aligned} \tag{13.13}$$

SVM-pairwise

Moreover, we use the approach of Liao and Noble (2002) in which Smith-Waterman alignment scores (Smith and Waterman, 1981) make up the SVM's

[3] An implementation can be downloaded from the mentioned website.

feature space.[4] Here, we use a randomly chosen subset of 250 positive and 250 negative examples and compute the alignment scores of a sequence to all 500 reference sequences. Computing the alignment to *all* training points, as done in Liao and Noble (2002), was not feasible for our problem. This procedure generates a 500-dimensional vector describing each sequence which we used as the input to a linear kernel, that is, $k(\mathbf{x}, \mathbf{x}') = \langle \mathbf{x}, \mathbf{x}' \rangle$.

Spectrum kernel

Additionally, we considered the spectrum kernel (Leslie et al., 2002). However, since it does not contain the information where the subsequences are located, it did not seem appropriate to use it for our problem. Moreover, it performed very poorly in some preliminary tests on our data sets (not shown).

Normalization

Finally, a remark regarding the normalization of the kernels is in order. In the case of the TOP kernel and SVM-pairwise, we first normalized each feature to have zero mean and standard deviation 1. Moreover, we normalized all kernels except the LI kernel such that the vectors in feature space $\Phi(\mathbf{x})$ have length 1. This can be done efficiently by redefining the kernel as follows:

$$\tilde{k}(\mathbf{x}, \mathbf{x}') = \frac{k(\mathbf{x}, \mathbf{x}')}{\sqrt{k(\mathbf{x}, \mathbf{x}) k(\mathbf{x}', \mathbf{x}')}}. \tag{13.14}$$

In particular, the latter normalization solved many convergence problems of the SVM optimizer. In the case of the LI kernel the normalization was not necessary, since it never led to any convergence problems without normalization.

13.3.2 Predicting the Splice Form

Using the techniques of the previous section we can identify single splice sites. In some cases, however, the classifier will predict false positives or miss a site. In these cases it is beneficial to incorporate more knowledge about the splice product. In particular, that after every donor has to follow an acceptor site and that an exon has to have an open reading frame. The input to the second stage of the system as described in this section will be a list of single sites together with the SVM or MM scores. From this list we generate a consistent splice form for a particular gene.

At this stage we assume that the number E of exons of a gene is known and we have already generated a list of potential exons (containing an open reading frame) from a virtual gene as described in the appendix. Then the task is to predict the E exons that constitute the coding region of a (virtual) gene.

We first define a probabilistic model (HMM) of a gene and then – by using the Viterbi algorithm – find the most likely path to predict which of the potential exons are used. The model uses the following components:

1. *Probability density estimates over the lengths of introns and exons.*
On the training and validation data we counted how often which length of exons

[4] Thanks to A. Zien for providing the code.

and introns appear. We used a log-scaled histogram with 100 bins and with pseudo-count 1 to estimate the probabilities $p_e^l(\mathbf{s})$ and $p_i^l(\mathbf{s})$ that a sequence $\mathbf{s}$ of a certain length is an exon or an intron, respectively.

2. *Probabilities for frame shifts in splicing.*
Here, we counted how often a frame shift of zero, one, or two nucleotides appears on the basis of the length of exons in training and validation data (pseudo-count one). We denote this probability by $p^{fs}(\mathbf{s})$ for a sequence $\mathbf{s}$.

3. *Probability density estimates for the GC content of introns and exons.*
We estimated the average GC content of exons ($\mu_e = 40.31\%$) and introns ($\mu_i = 35.9\%$). We assume a simple Gaussian model for the GC content and use the following probabilities for a sequence $\mathbf{s}$: $p^{gc}(\mathbf{s}) \sim \exp(|gc(\mathbf{s}) - \mu|^2/\sigma^2)$, where $gc(\mathbf{s})$ denotes the GC content of $\mathbf{s}$, μ can either be μ_e or μ_i, and σ is the standard deviation which we used for both models ($\sigma = 5.56\%$, as estimated from the exon sequences).

4. *Scores derived from the SVM predictions.*
SVMs do not output posterior probabilities. The usual way to deal with this problem (Platt, 2001) is to transform the output of the SVM with the sigmoid function $\theta(\cdot)$ to obtain a probability-like score. In our model we used $\hat{p}^{svm}(\mathbf{s}) = \frac{1}{2} + \theta(af(\mathbf{s}) + b))/2$, where $f(\cdot)$ is the SVM prediction and the constants $a = \frac{3}{4}$ and $b \approx -\frac{3}{4}$ were found to be optimal on the validation set (depending on the kernel).

5. *Rule about the length of the translated region.*
We designed the HMM in a way such that only such exons are predicted whose lengths add up to a number that is a multiple of 3.

For a given splicing S of a gene into exons $\mathbf{e}_1, \ldots, \mathbf{e}_E$ and introns $\mathbf{i}_1, \ldots, \mathbf{i}_{E-1}$ we can compute the *score* of S, $\hat{p}(S)$, as follows:

$$p^{gc}(\mathbf{e}_1)p^{fs}(\mathbf{e}_1)p_e^l(\mathbf{e}_1)\prod_{j=2}^{E}\hat{p}_{don}^{svm}(\mathbf{ei}_j)\ p_i^{gc}(\mathbf{i}_j)p_i^l(\mathbf{i}_j)\ \hat{p}_{acc}^{svm}(\mathbf{ie}_j)\ p_e^{gc}(\mathbf{e}_j)p^{fs}(\mathbf{e}_j)p_e^l(\mathbf{e}_j), \tag{13.15}$$

where $\mathbf{ei}_j$ is the donor site sequence at the boundary of the $(j-1)$th exon and $(j-1)$th intron and $\mathbf{ie}_j$ is the sequence on the border of the $(j-1)$th intron and the jth exon. Using standard techniques (Viterbi algorithm; cf. Durbin et al., 1998) we can now find the most likely sequence of exons and introns which forms our prediction. Note that the model above is not normalized in a probabilistic sense, as the sum over all paths do not sum to 1. Moreover, since the score tends to decrease exponentially with the number of exons it cannot directly be used for predicting the number of exons.

All parameters and probabilities have been estimated using the training and validation set, but not the test set.

13.3.3 Predicting the Number of Exons

One of the most important things to know about the gene is the number of exons. If this prediction is wrong, all subsequent steps have to fail. There are several ways to estimate the number of exons. The common approach is to choose the most likely number of exons for a probabilistic model like the above, this may require some additional normalization terms in (13.15). In this chapter we follow a different approach, using the SVM predictions only.

Our prediction works as follows: We start by assuming that the gene has E exons (starting with two) and compute the optimal splicing as described in subsection 13.3.2. Then we sum up the SVM scores for all donor and acceptor sites, that is, $s_E = \sum_{j=2}^{E} \left(f_{don}(\mathbf{ei}_j) + f_{acc}(\mathbf{ie}_j)\right)$. This score is compared with the score s_{E+1} for $E+1$ exons. If $s_E \geq s_{E+1} + \alpha$, then one predicts E exons. Otherwise, one increases E and repeats the loop. Here α is a parameter of the algorithm and controls whether one tends to predict too few or too many exons.

The rule is indeed very simple, but it works surprisingly well – as we will see later. It might have a biochemical interpretation, in that the splicing process stops in the first local minimum of some energy – assuming the SVM outputs are related to the energy.

13.4 Results and Discussion

In this section we discuss the experimental results obtained with our methods on the left-out set of genes (cf. appendix). First we only consider the accuracy of identifying a single donor and acceptor site. For the best methods we then compare our splice predictions with the ones of GenScan (Burge and Karlin, 1997).

13.4.1 Accuracy of Single Donor and Acceptor Predictions

Setup and Model Selection We start with generating true splice and decoy examples for training, validation, and test genes as described in the appendix. Here, we choose a window of ±30 nt around the site with the consensus `AG` or `GT` dimer centered. A similar window length has been used in previous studies (cf. Sonnenburg, 2002; Sonnenburg et al., 2002). To be able to apply the SVM, we have to find the complexity parameter C, controlling the tradeoff between training error and complexity, and the kernel parameters. For instance, in the case of the LI kernel this is the degree d and window size l. To select these hyper parameters, we train the SVM on the training set and evaluate it on the validation set for different settings of C, d, and l. We tried $C = [0.25, 0.5, 0.75, 1, 2, 5, 10, 20]$, $d = 1, \ldots, 5$ and $l = 1, \ldots, 6$. For acceptor predictions we found $C = 0.75$, $d = 4$, and $l = 3$ as optimal parameters. For donor predictions $C = 1$, $d = 3$, and $l = 2$ are optimal. We omit the details on the model selection for the other methods, but they are

Table 13.1 Classification error and ROC score for Markov models, TOP kernel, SVM-pairwise, polynomial kernel, locality improved kernel, and weighted degree kernel. Note that the data sets are quite unbalanced: the constant classifier $g(\mathbf{x}) = -1$ would achieve an error of about 2.89%. Hence, the classification errors shown should be taken with care. Also, they depend heavily on the choice of the bias.

	Test Error		ROC Score	
	Donor	Acceptor	Donor	Acceptor
Markov	1.85%	1.54%	98.23%	98.88%
TOP	1.82%	1.66%	98.32%	98.70%
Pairwise	2.17%	1.94%	97.60%	98.00%
Polynomial	1.91%	1.53%	98.31%	98.95%
Locality	1.81%	1.44%	98.48%	99.08%
Weighted degree	1.79%	1.45%	98.47%	99.05%

available from the website `http://ida.first.fhg.de/splice` together with the data.

Generalization Performance For the selected hyper parameters we measure the performance on the test set. For acceptor predictions the test set contained 75,905 examples (2132 true sites). For donor predictions we have 73,784 test sequences (2132 true sites). In table 13.1 we summarize the results of our comparison. We show the classification error on the test set and also the receiver operating characteristic (ROC) score (the area under the ROC curve). We have a few observations:

- The TOP kernel slightly improves the simple probabilistic model for donor sites, but is (according to the ROC scores) not as good as, for instance, the SVM with LI or weighted degree kernel.
- The simple polynomial kernel performs surprisingly well, given that the feature space contains the correlation of all positions up to order 4 (in our case). It is only slightly worse than the specialized kernels such as the LI or weighted degree kernel.
- While the test errors for the MM are considerably greater than the ones for the polynomial-like kernels, the ROC score is very similar. For this reason we chose to include the Markov model in the GenScan comparison.
- The SVM-pairwise method did not perform well. This might be due to the small set of reference sequences. We tried to double the number but could not measure significant differences in performance.

We can conclude that the LI and the weighted degree kernel are best suited for the task of identifying single splice sites. The latter and the MM are chosen for the comparison with GenScan in section 13.4.3.

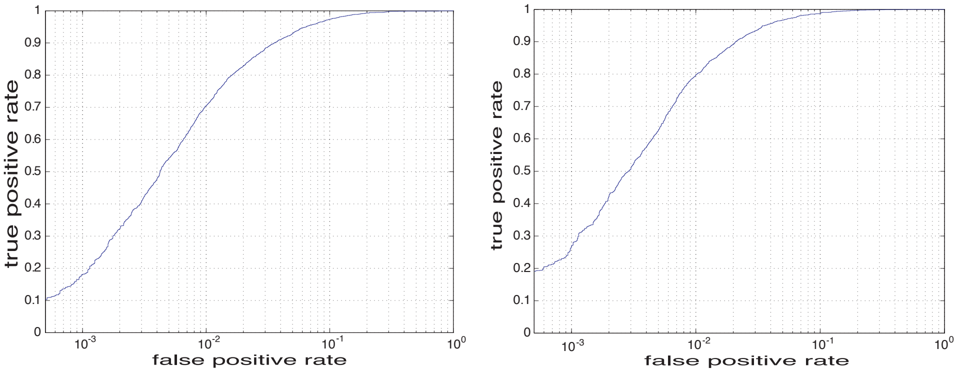


Figure 13.5 ROC curves for donor (*left*) and acceptor (*right*) predictions on the test set. The false-positive rate is plotted in log-scale.

In figure 13.5 we show ROC curves for acceptor and donor predictions of the SVM using the LI kernel. In Sonnenburg et al. (2002) and Sonnenburg (2002), we obtained considerably worse results for prediction on *C. elegans* using SVMs with the same kernel (e.g., only 97.3% accuracy for acceptor sites). In this case, the data were generated differently and the observed improvement suggests that the current data set is much cleaner and thus allows more accurate predictions. Also, in our previous study we have shown that our method is superior to a state-of-the-art method such as NN-BRAIN (Rampone, 1998). In future work we will compare our method with other prediction methods such as NetGene (Hebsgaard et al., 1996) and GeneSplicer (Pertea et al., 2001).

13.4.2 Splice Site Activity Variations for Different Intron Ranks

There seems to be evidence that the removal of introns occurs more or less sequentially in a certain order. In Lewin (2000) we found an example with seven introns where the fifth and sixth introns are removed first, then the fourth and seventh introns, followed by the first and second, and at last, by the third intron. It is conjectured that the third intron is often removed last, while the fifth or sixth intron is usually removed first. This was explained by the fact that after removing an intron the conformation of the mRNA is changed, other sites become available for splicing, and the conformation changes determine the order of intron removal.

Here we show that the order in which the introns are removed might rather be explained by the "activity" of the splice sites, in particular of the donor sites. In order to show this, we use the previously described SVMs for donor and acceptor sites and compute the median of all donor and acceptor scores of introns that appear at a certain position within the gene (figure 13.6). The rank of an intron is determined before assembling the virtual genes and hence is based on the true location within a gene. Since some introns might be missing, the rank might be estimated as too low (in particular for larger ranks). We chose the median to obtain a robust estimate.

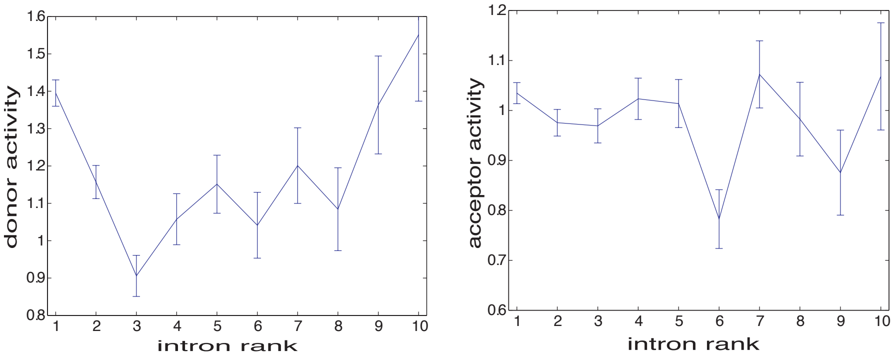


Figure 13.6 Median of the SVM predictions for donor (*left*) and acceptor (right) sites of introns with different ranks within a gene. The error bar indicates a confidence interval for the estimate.

We observe that the third donor site has indeed the lowest "activity" and one may need the most energy[5] to remove the third intron and hence the removal is slowest. Furthermore, we can quite accurately reproduce the previously described order of intron removal, when considering only the first seven intron positions: first, the seventh and second, then the fifth, fourth, and sixth, and at last the third intron. The removal of the second intron seems to be out of order. Additionally, the first intron's donor sites seem to have a particularly high score. However, according to Lewin (2000), it is not removed first. We conjecture that there are interactions with other processes, such as adding the cap structure, that delay the removal of the first intron (cf. Proudfoot et al., 2002). The same argument might apply to a lesser extent to the second intron. The order of the remaining introns matches reasonably well the order described in Lewin (2000). This suggests that also the biochemical activity of a site seems to determine the splicing order and not only the tertiary structure of the mRNA.

The activities of the acceptor sites stay reasonably constant for different ranks, except for two very weak positions: the sixth and the ninth introns' acceptors. The deviation is quite significant, but we have not yet found a biochemical explanation for it. Although one could conclude from these results that the acceptor sites only play a minor role in determining the splicing order, additional results on a new data set (not shown) suggest that the acceptor can also be important in determining the splicing order (cf. Rätsch and Sonnenburg, 2004).

[5] Again, we assume that the SVM output is related to the biochemical activity of a site related to the needed energy of the biochemical reaction.

Table 13.2 Accuracies of GenScan and our methods: first using our exon predictions ($\hat{E}$) and second, for completeness, assuming knowledge of the correct number of exons (E). The line "all correct" states the fraction of genes for which all splice sites have been correctly predicted. The line "splice correct" denotes the fraction of correctly spliced genes given that the number of exons was predicted correctly. The line "exon—no. correct" states the fraction of genes for which the number of exons was correctly determined. Note that the top line is the product of the following two lines.

	GenScan	Markov model		LI Kernel		WD Kernel	
		$\hat{E}$	E	$\hat{E}$	E	$\hat{E}$	E
all correct	77.5%	90.0%	n/a	92.4%	n/a	**92.7%**	n/a
splice correct	87.7%	97.4%	96.1%	98.2%	97.9%	**98.7%**	**98.3%**
exon—no. correct	88.3%	92.4%	n/a	**94.0%**	n/a	93.9%	n/a

13.4.3 Comparison with GenScan

In a last experiment we compared our splice form prediction method as described in subsection 13.3.2 with a state-of-the-art method – GenScan. Since GenScan is designed to solve the harder task of finding complete genes, including promoters, initial and terminal exons, and so on, we needed to take special care that the experimental setup allowed a fair comparison. The idea is to run GenScan on all virtual genes in the test set and determine for which genes it found the correct start of the initial exon and the end of the terminal exon. All other genes were not used in this comparison. This procedure is necessary, since the start/end position is assumed to be available to our algorithm.

We used GenScan with the *Arabidopsis* genome setting, which seems a reasonable choice for *C. elegans*.[6] The original test set contains 1686 virtual genes of which GenScan finds the correct start and end position for 889 sequences. We say splice form is correctly predicted, if *all* exons and introns are predicted correctly. The prediction accuracy (i.e. the fraction of correctly predicted splice forms) of GenScan on this restricted set of genes is given in table 13.2.

To use our splice site prediction we first have to determine the number of exons as described in subsection 13.3.3. This method has the hyperparameter α, which we find by maximizing the prediction accuracy on the validation set. We find $\alpha = 1.7$ to be optimal for the LI and weighted degree-kernel. Given the number of exons, we employ the method described in subsection 13.3.2 using the tuned SVMs as in subsection 13.4.1. The results are given in table 13.2 (on the same reduced set of genes).

We found that our simple method for predicting the number of exons is quite accurate (93.9%) – given its simplicity, which corresponds to an error rate of less

[6] All other available choices perform much worse and an optimized *C. elegans* setting is not available.

than half of that of GenScan. The achieved accuracy suggests that mainly the biochemical activity of the splice sites (estimated with the SVM score) determine whether an intron is removed. Only to a lesser extend does it depend on the tertiary structure (which would be very hard to predict from the short sequences given to the SVM) or other properties of the sequence.

The overall prediction accuracy of the best method is 92.7%, while GenScan performs much worse (77.5%). The improvement is mainly due to better splice prediction when given the correct number of exons (cf. table 13.2). For this part the improved predictions using SVMs enriched with some statistical information is crucial. Our statistical modeling is still quite basic and covers only a few of many statistical aspects. We conjecture that our predictions will be improved when using a more refined statistical model.

13.5 Conclusion

In this work we designed a new splice finder system that accurately predicts the splice form for *C. elegans* genes. A main achievement is a newly generated set of EST and complete cDNA confirmed virtual genes. Only by using this set of genes were we able to train and tune a support vector classifier which very accurately predicts whether a site is a true splice site or a decoy. We tested several kernels for SVMs and found that the LI kernel and the weighted degree kernel are best suited for the single site prediction. Enriched with statistical information such as intron and exon length statistics and a simple heuristic for predicting the number of exons, our splice finder system was able to translate the high accuracy on the single site prediction into a considerably improved accuracy on the splice form prediction: Our system makes only a third of the number of mistakes of GenScan (92.7% vs. 77.5% accuracy).

Note, however, that in this work we only considered virtual genes generated from ESTs and complete cDNA, i.e., they tend to be too short (some exons and introns at both ends may be missing). Moreover, we considered only the subset of genes for which GenScan detected the correct position of TIS and stop codon. This introduces an additional bias toward genes that are easy to predict.[7]

It is our goal to further improve our results. A better statistical modeling (e.g., normalization of the model) may lead to a better exon number prediction. Additionally, a cleaner data set will likely lead to accuracy improvements in the single site prediction. Moreover, we have not considered the prediction of the TIS and stop codon positions, for which a particularly designed support vector classifier might be beneficial when used within a gene finder. Other directions to consider are alternative splicing and noncanonical splice sites.

[7] Recently we performed experiments on a newly generated set of complete genes on which GenScan and our method performes less accurately (our method performed much better thann GenScan though).

Acknowledgments

We particularly thank A. Pannek for great discussions and for proofreading the manuscript. Moreover, we thank S. Heymann, K. Tsuda, A. Zien, A. Zahler, C. Sugnet, K.-R. Müller, A. Smola, M. Warmuth, C. Leslie, and E. Eskin for stimulating discussions. This work was partially funded by DFG under contract JA 379/9-2, JA 379/7-2, MU 987/1-1, and NSF grant CCR-9821087 and supported by an award under the Merit Allocation Scheme of the National Facility of the Australian Partnership for Advanced Computing. Part of this work was done while G. R. was at the Australian National University in Canberra.

Appendix: Data Generation

Data generation proceeds in two steps: First, we generate a list of virtual genes from complete cDNA matches to the genomic sequence, and second, we derive true and decoy donor and acceptor splice examples from the virtual genes with known splice sites.

Generating Virtual Genes from EST and Complete cDNA Sequences

To generate a clean splice data set we started with the genomic sequence (*C. elegans* Sequencing Consortium, 1998) and the set of known EST and complete cDNA sequences (Benson et al., 1999) of *C. elegans*, which we downloaded from the Intronerator website (Kent and Zahler, 2000). We employed NCBI Blast (Altschul et al., 1990) to match all cDNA sequences to the genomic DNA (with default parameters) and obtained a list of matches, of which we only used the matches without gaps on the cDNA and with at least 80% identity. Using these matches we generated lists of introns, internal exons, and potential initial and terminal exons for each cDNA sequence as follows:

- **Introns** An intron is defined by a large gap on the DNA in the match of cDNA to DNA. We only considered gaps of length at most 500 knt. If cDNA matches to different positions on DNA, then we use the one leading to the largest fraction of used cDNA and the smallest gap on the DNA (preferring smaller introns). The nonmatching DNA sequence is assumed to be an intron and added to the intron list for the current cDNA sequence, if the consensus sequence `GT...AG` was found. (We only consider so-called canonical splice sites, i.e., sites with the consensus `GT` or `AG` at the splice site.)
- **Internal Exons** A match between cDNA and DNA is considered to be an internal exon if there are two adjacent introns in the intron list of the cDNA sequence.
- **Initial/terminal exons** If only one intron at the boundary of a cDNA match was found, then it is assumed to be an initial/terminal exon. The length of the exon is given by the cDNA match, but not all of it is part of the coding region. Hence,

we check for open reading frames and fix the length of the exon by cutting it at the position of the appearing stop codon (`TAA`, `TAG`, or `TGA`). However, the exact start/end position cannot be derived from the cDNA matches. We included this exon in the list of initial/terminal exons if it did not overlap with any internal exon on the same strand.

In the next step we removed redundancies by removing exact duplicates within the intron and internal exon lists. Furthermore we removed initial exons with identical ends and terminal exons with equal start positions. This left 24,191 introns and 41,245 exons (35% initial, 30% internal, 35% terminal). The number of introns derived by (Kent and Zahler, 2000) using different techniques is slightly larger.

For this study we were interested in generating a clean splice data set and therefore wanted to exclude the effects of alternative splicing. Thus, we removed introns and exons of all genes for which we could find an intron overlapping with an exon or an exon within an intron. We found evidence for alternative splicing for 2560 introns on 1228 genes (out of 9247 total). Finally, we ended up with 19,544 introns and 35,464 exons in 8019 remaining genes.

By considering all exons and introns of one gene, we can determine the positions of the introns relative to each other and compute ranks for each intron within the gene. Furthermore, we can check whether introns or exons are missing. Each contiguous sequence of maximal length and at least two exons (with matching introns) we call a *virtual gene,* of which we found 4413.

Unfortunately, it happens rather often that different cDNA sequences of one gene are labeled with different gene identifiers, matched to the same location on the DNA. Hence one will generate introns labeled with different gene identifiers in the first step, which are then omitted. Therefore, we may end up with contiguous introns/exons with different gene identifiers. This poses a problem for the above-described method and we miss a considerable amount of (virtual) genes. It will be possible, in the future, to find all matching sequences, independent of the labeling.

Each virtual gene may or may not start with an initial exon and end with a terminal exon. To properly define the start and end of the virtual gene, which will be needed later for comparison with GenScan, we remove the first half of the first and the last half of the last exon and append fixed sequences, which have been derived from gene *Y48G1C_55.c* on chromosome 1 of *C. elegans* as follows. We generated two sequences: (1) 571 nt upstream of the TIS (including the promoter) to 51 nt downstream and (2) 51 nt of the last exon to 783 nt downstream from the stop codon. This gene and these positions have been chosen such that GenScan finds the correct TIS and stop codon as often as possible for our set of genes.

Finally, we have a set of 4413 virtual genes with defined start and end positions, and with all splice sites known. We randomly chose 2153 for training, 574 for validation (tuning), and 1686 for testing.

Generating Donor and Acceptor Sites

From the sets of virtual genes with known splice sites we can straightforwardly derive positive training sequences for donor and acceptor sites. It is more difficult to come up with an appropriate set of negative sequences (decoys), which is needed for training a supervised learning algorithm. In Pertea et al. (2001) decoy sites were chosen randomly. In Reese et al. (1997) a window of ±40 nt around true sites was used, from which a list of sequences containing the consensus dimer at the correct position was generated. However, the chosen window size is quite arbitrary and also has a great effect on the performance (cf. Sonnenburg, 2002). There are two additional problems: (1) One may generate decoy examples that can logically not be a boundary of an exon (since there is, e.g., no open reading frame) and (2) for long exons or introns one misses the majority of potential splice sites in the interior of the sequence and will therefore not be very successful in applying the splice detector in a gene finder.

Arguably, the best way to generate the decoy sequences for use in a gene finder would be to run the first pass of a gene finder. The result is a list of potential exons, which all have the property to start after the `AG` dimer (followed by an open reading frame; in our case at least 12 nt in length) and to end before the `GT` dimer. From this list one then generates the list of potential donor and acceptor sites, excluding the true sites and using the remaining sites as decoy examples. This way we obtain about 180,000 donor and 195,000 acceptor decoy examples and 8150 true donor/acceptor examples. Note that the above-described method does not require another gene finder, but only a list of potential exons with acceptor and donor sites that any other gene finder would also consider to find the splice form.