EXTENDED ABSTRACT

Engineering Support Vector Machine Kernels That Recognize Translation Initiation Sites

A. Zien[†], G. Rätsch[*], S. Mika[*], B. Schölkopf[*], C. Lemmen[†], A. Smola[*], T. Lengauer[†], K.-R. Müller[*]

[†] *GMD.SCAI, Schloss Birlinghoven, 53754 Sankt Augustin, Germany*
{zien, lemmen, lengauer}@gmd.de
[*] *GMD.FIRST, Rudower Chaussee 5, 12489 Berlin, Germany*
{raetsch, mika, bs, smola, klaus}@first.gmd.de

Abstract In order to extract protein sequences from nucleotide sequences, it is an important step to recognize points from which regions encoding proteins start, the so-called translation initiation sites (TIS). This can be modeled as a classification problem. We demonstrate the power of support vector machines (SVMs) for this task, and show how to successfully incorporate biological prior knowledge by engineering an appropriate kernel function.

1 Introduction

Living systems are determined by the proteins they produce according to their genetic information. But only parts of the nucleotide sequences carrying this information encode proteins (CDS, for coding sequence), while other parts (UTR, for untranslated regions) do not. Given a piece of DNA or mRNA sequence, it is most interesting to know whether it contains CDS, and, if so, which protein it encodes.

In principle, both CDS and the encoded protein can be characterized using alignment methods. Programs capable of aligning nucleotide sequences to protein databases include FASTX/FASTY, SearchWise and BLASTX. However, this approach is hampered by two severe problems: First, there are several sources of noise making the task more difficult and error-prone than pure protein alignment: (i) The correct strand and reading frame have to be found. (ii) Additional false hits may result from UTR sequences. (iii) Sequencing errors may disrupt the correct reading frame. Second, alignment based approaches rely on homologous proteins being known. This renders them unapt to find novel genes. Thus, a method to identify CDS in nucleotide sequences is desirable, both in order to ease the task for alignment-based approaches as well as to find new genes.

Since living cells are able to distinguish between CDS and other nucleotide sequence parts without utilizing any homology information, this should in principle also be possible for computer programs. In fact, there are algorithms that identify CDS merely relying on properties intrinsic to nucleotide sequences. The most successful programs include GENSCAN [5] for genomic DNA and ESTScan [6] for ESTs. ESTs are single-read partial sequences derived from mRNA that are particularly error-prone. ESTScan implements a fifth-order hidden Markov model that simultaneously recognizes CDS by typical oligo-nucleotide frequencies and corrects sequencing errors. It does not incorporate a model of translation initiation site (TIS) sequences, that mark the beginning of CDS. GENSCAN employs generalized hidden Markov models to capture the structure of an entire genome. Despite its overall sophistication, GENSCAN uses a relatively crude TIS model: a piece of sequence is assigned a probability for being a TIS, based on the positional relative frequencies of individual nucleotides observed around a true TIS.

There exists a number of more elaborate models of TIS themselves. Salzberg extends the positional probabilities (as used by GENSCAN) to first order Markovian dependencies [13]. This is essentially a proper probabilistic modeling of

positional di-nucleotides, and leads to a significant increase in recognition performance. There also exist methods to explicitly capture correlations between non-adjacent positions near TIS or other signals [1], possibly aiding insight into the structural functioning of these signals. However, since few such correlations can be proved to be significant in TIS sequences, there is little gain for TIS recognition.

All models discussed so far can be called generative, as they can be used to generate potential TIS sequences with approximately the true probability distribution. Applying such models, a sequence is considered a TIS if the probability it is assigned exceeds some threshold. The more closely the true distribution is approximated, the better this approach works. By using so-called discriminative methods, in general a superior distinction between true and pseudo TIS can be achieved. These methods aim at learning to discriminate certain objects from others, without explicitly considering probability distributions.

For example, the program `ATGpr` [12] uses a linear discriminant function that combines several statistical measures derived from the sequence. Each of those features is designed to be distinctive for true versus pseudo TIS. Learning allows to find a (linear) weighted combination of features that achieves a good discrimination.

A radically different approach to learning a discriminating function is taken by Pedersen and Nielsen [11]. They train an artificial neural network (NN) to predict TIS from a fixed-length sequence window around a potential start codon (ATG). The input of the NN consists of a binary encoding of the sequence; no higher-level features are supplied. The intriguing idea is that the NN learns by itself which features derived from the sequence are indicative of a true TIS.

Of the described methods, only `ATGpr` makes use of the ribosome scanning model [9]. According to this model, the translation starts at the first occurrence of an appropriate signal sequence in the mRNA, and thus sequences further downstream that resemble typical TIS are inactive (pseudo sites). The scanning model can be combined with any TIS recognition method, and is confirmed by the resulting improvements of recognition [2]. Since the model is orthogonal to TIS signal sequence recognition itself, and is limited to complete mRNA sequences, we will not consider it in the following.

In this paper, we show that we can top the performance of established methods for TIS recognition by applying support vector machines (SVMs) [4]. Like NNs, SVMs are a discriminative supervised machine learning technology, i.e. they need training with classified data in order to learn the classification. For the task of TIS recognition, we show that SVMs can be superior to NNs. To achieve this performance gain we use a particularly desirable property of SVMs: the ability to adapt them to the problem at hand by including prior knowledge into the so-called kernel function. Here, we demonstrate how to make use of rather vague biological knowledge. The paper is structured as follows: we first give a brief description of the SVM technique, then present experiments and finally discuss results and potential applications.

2 Methods

2.1 Support Vector Machines

Formally, SVMs like any other classification method aim at estimating a classification function $f : \mathcal{X} \to \{\pm 1\}$ using classified training data from $\mathcal{X} \times \{\pm 1\}$ such that f will correctly classify unseen examples (test data). In our case, $\mathcal{X}$ will contain simple representations of sequence windows, while ± 1 corresponds to true TIS and pseudo sites, respectively.

In order to be successful, two conditions have to be respected. First, the training data must be an unbiased sample from the same source as the test data will be. This concerns the experimental setup. Second, the size of the class of functions that we choose our estimate f from, the so-called capacity of the learning machine, has to be sensibly restricted. If the capacity is too small, complex discriminant functions can not be sufficiently well approximated by any selectable function f – the learning machine is too dumb to learn well. On the other hand, too large a capacity bears the risk of loosing the ability to learn a function that generalizes well to unseen data. The reason lies in the existence of infinitely many functions that are consistent with the training examples, but disagree on unseen (test) examples. Most of those functions per-

fectly memorize the particular examples used for training, but do not reflect general properties of the classification. Picking such a function is called overfitting.

In neural network training, overfitting is avoided by early stopping, regularization or asymptotic model selection [3, 10]. In contrast, the capacity of SVMs is limited according to the statistical theory of learning from small samples [17]. For learning machines implementing linear decision functions this corresponds to finding a large margin separation of the two classes. The margin is the minimal distance of training points to the separation surface (cf. Figure 1). Finding the maximum margin separation can be cast as a convex quadratic programming (QP) problem [4]. The time complexity of solving such a QP scales approximately between quadratic and cubic in the number of training patterns (see [14]), making the SVM technique computationally comparably expensive.

With respect to good generalization, it often is profitable to misclassify some outlying training data points in order to achieve a larger margin between the other training points. See Figure 1 for an example. This 'neglectful' learning strategy also masters inseparable data [16], which is frequent in real-world applications. The trade-off between margin size and number of misclassified training points is controlled by a parameter of the SVM, which therefore can be used to control its capacity. This extension still permits optimization via QP [4].

It is tempting to think that linear functions can be insufficient to solve complex classification tasks. A little thought reveals that this in fact depends on the representation of the data points. Canonical representations, as frequently used to define input space, tend to minimize dimensionality and avoid redundancy. Then, linearity may easily be too restrictive. However, one is free to define (possibly redundant) features that non-linearly derive from any number of input space dimensions. Even for complex problems, well chosen features could ideally be related to the respective classification by rather simple means, e.g. by a linear function (cf. Figure 2).

Any linear learning machine can be extended to functions non-linear in input space $\mathcal{X}$ by

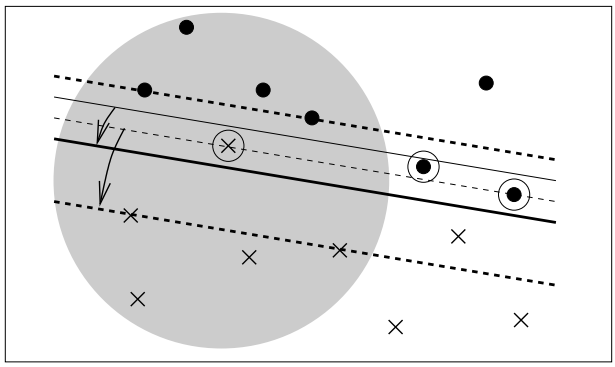

Figure 1: A binary classification toy problem: separate dots from crosses. The shaded region consists of training examples, the other regions of test data (spatial separation for illustration clarity only). The data can be separated with a margin indicated by the slim dotted lines, implicating the slim solid line as decision function. Misclassifying one training point (circled cross) leads to a considerable extension (arrows) of the margin (fat lines) and thereby to the correct classification of two test examples (circled dots).

explicitly transforming the data into a feature space $\mathcal{F}$ using a map $\Phi : \mathcal{X} \to \mathcal{F}$ (see Figure 2). SVMs can do so *implicitly*, thanks to their mathematical niceness: all that SVMs need to know in order to both train and classify are dot products of pairs of data points $\Phi(x), \Phi(y) \in \mathcal{F}$ in feature space. Thus, we only need to supply a so-called kernel function that computes these dot products. This kernel function k implicitly defines the feature space (Mercer's Theorem, e.g. [4]) via

$$k(\mathbf{x}, \mathbf{y}) = (\Phi(\mathbf{x}) \cdot \Phi(\mathbf{y})) \,.$$

Note that neither the SVM nor we need to know Φ, because the mapping is never performed explicitly. Therefore, we can computationally afford very large (e.g. 10^{10}-dimensional) feature spaces. SVMs can still avoid overfitting thanks to the margin maximization mechanism. Simultaneously, they can learn which of the features implied by k are distinctive for the two classes. So, instead of having to design well-suited features by ourselves (which can often be difficult), we can use the SVM to select them from a sufficiently rich feature space. Of course, it well be helpful if the kernel supplies a type of features related to the correct classification. In the next sections, we will show how to boost the process of learning by choosing appropriate kernel functions.

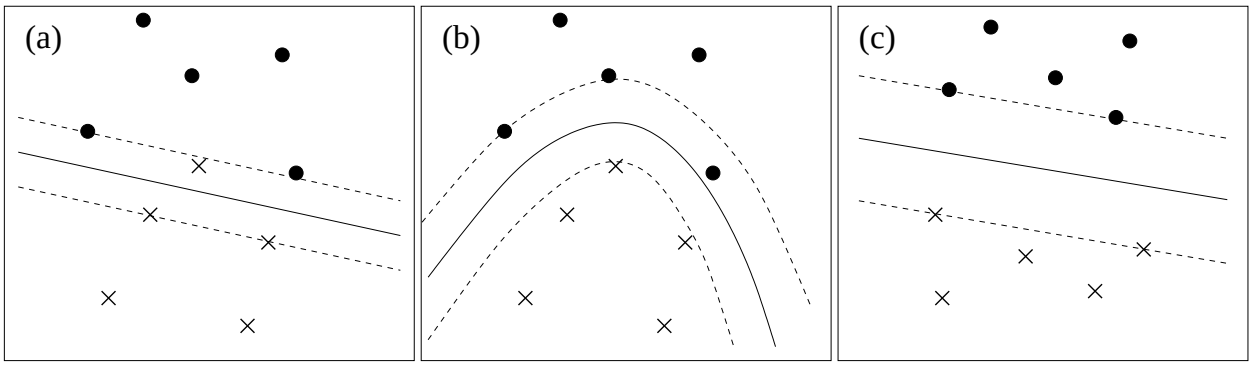

Figure 2: Three different views on the same dot versus cross separation problem. Linear separation of input points (a) does not work well: a reasonably sized margin requires misclassifying one point. A better separation is permitted by non-linear functions in input space (b), which corresponds to a linear function in a feature-space (c). Input space and feature space are related by the kernel function (see main text).

2.2 Data sets

Little experience exists in the application of SVMs to biomolecular problems (to our knowledge, only work by Jaakkola and Haussler [8]). Therefore, we compare the performance of SVMs to that of the most popular alternative general purpose machine learning technology, neural networks (NNs). In order to do so, we use the vertebrate data set provided by Pedersen and Nielsen [11]. We take care to only replace the learning machinery while retaining the setting: the definition of training and test data sets as well as the definition of input space.

The sequence set is built from high quality nuclear genomic sequences of a selected set of vertebrates taken from GenBank. All introns are removed, in analogy to the splicing of mRNA sequences. The set is thoroughly reduced for redundancy, to avoid over-optimistic performance estimates resulting from biased data. This protocol leaves 3312 sequences (see [11]).

From these sequences, the data set for TIS recognition is built as follows. For each potential start codon (the nucleotide sequence ATG) on the forward strand, one data point is generated. This leads to 13503 data points, of which 3312 (24.5%) correspond to true TIS and the other 10191 (75.5%) correspond to pseudo sites. Each data point is represented by a sequence window of 200 nucleotides centered around the respective ATG triplet. Pedersen and Nielsen divide the data into six parts of nearly equal size (≈ 2200 points) and fraction of true TIS. Each part is in turn reserved for testing the classification learned from the other five parts.

2.3 Engineering the kernel function

Given the data sets, we have to choose a kernel function k for training. A standard kernel function is the simple polynomial kernel $k(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y} + \kappa)^d$. This kind of kernel takes two parameters: the degree d and an additional constant κ. Here, we use homogenous ($\kappa = 0$) polynomials of first to fifth degree ($d = 1, \ldots, 5$). Degree one corresponds to a linear separation in input space. The input space is defined by a sparse bit-encoding scheme as used by Pedersen and Nielsen (personal communication): each nucleotide is coded by five bits, exactly one of which is set. The position of the set bit indicates whether the nucleotide is A, C, G or T, or if it is unknown. Thus, the dot product $\mathbf{x} \cdot \mathbf{y}$ simply counts the number of nucleotides that coincide in the two sequences represented by $\mathbf{x}$ and $\mathbf{y}$. If the degree d is set to two, the feature space reflects all pairwise correlations of the nucleotide frequencies at any two sequence positions. A degree of three would correspond to all correlations of (possibly scattered) triplets, and so on. With this simple kernel function we already achieve results competitive to those of the NN devised by Pedersen and Nielsen (see Table 1).

We design an improved kernel function by incorporating basic biological knowledge. We make use of only one observation: While certain local correlations are typical for TIS, dependencies between distant positions are of minor importance or do not even exist. We want the feature space to reflect this fact. Thus, we modify the kernel utilizing a technique is described in [15]: At each sequence position, we compare

algorithm	overall	true TIS	pseudo sites	specificity	sensitivity
SVM, simple polynomial	13.2%	30.1%	7.8%	74.6%	69.9%
SVM, locality-improved kernel	11.9%	30.1%	5.9%	79.5%	69.9%
SVM, codon-improved kernel	12.3%	29.8%	6.4%	78.2%	70.2%
neural network	15.4%	17.6%	14.8%	64.5%	82.4%
positional preference scores	12.3%	24.9%	8.4%	74.4%	75.1%

Table 1: Comparison of classification errors (first three columns: on all, on positive and on negative data points). All results (except for preliminary codon-improved figures) are averages over the six data partitions. SVMs are trained on 8000 data points, leaving the remaining training data ($\approx$ 3300 points) for the determination of suitable parameter values. The NN results are those achieved by Pedersen and Nielsen ([11], personal communication). Here, model selection seems to have involved test data, which might lead to slightly overoptimistic figures. Positional preference scores are calculated analogously to Salzberg [13], but extended to the 200 nucleotides around the ATG triplet also supplied to the other methods. All values shown correspond to the optimal overall performance, though the true vs. pseudo TIS (or equivalently sensitivity vs. specificity) trade-off can be controlled by varying the classification function threshold.

the two sequences locally, within a small window of length $2l+1$ around that position. Again, we count matching nucleotides, this time multiplied with weights w increasing from the borders to the center of the window. The resulting weighted counts are taken to the d_1^{th} power. d_1 reflects the order of local correlations (within the window) we expect to be of importance.

$$\text{win}_p(\mathbf{x},\mathbf{y}) = \left(\sum_{j=-l}^{+l} w_j \, \text{match}_{p+j}(\mathbf{x},\mathbf{y})\right)^{d_1}$$

Here, $\text{match}_{p+j}(\mathbf{x},\mathbf{y})$ is one for matching nucleotides at position $p+j$ and zero otherwise. The window scores computed with win_p are added up over the whole length of the sequence. Correlations between up to d_2 windows are taken into account by applying potentiation with d_2 to the resulting sum.

$$k(\mathbf{x},\mathbf{y}) = \left(\sum_{p=1}^{l} \text{win}_p(\mathbf{x},\mathbf{y})\right)^{d_2}$$

We call this kernel locality-improved. In Table 2 its TIS recognition performance is compared to that of the polynomial kernel for differently sized training sets.

Our kernel function poses the problem of how to set a number of parameters (in addition to the general parameter for SVM capacity control described in section 2.1). Since we consider long distance correlations unimportant we set d_2 to one. For each of the remaining parameters, we select a small number of values in an appropriate range. SVMs are trained with all combinations of these values, while excluding a part of the training set. This part is then used to measure the performance of the trained SVM and to select the corresponding parameters. With respect to window size ($2l+1$), we consider nucleotide composition (two and more), interactions between neighboring amino acids (six) and the assumed length of the ribosomal binding site (up to 14). For the degree of local correlations (d_1), we consider values up to five. Table 2 shows that the optimal parameterization of the kernel depends on the training set size. The more data available to the SVM, the more complex features it can reliably learn. The table also shows that the performance improvements over the polynomial kernel are very substantial for small numbers of training vectors, but decrease for larger training sets. Again, this is consistent with statistical learning theory. It is known that SVMs (as well as other learning algorithms) are asymptotically optimal with whatever kernel one chooses, i.e. they will perform well when supplied with enough training data (e.g. [17]).

In an attempt to further improve performance we try to incorporate another piece of knowledge into the kernel, that again is rather diffuse: the codon-structure of coding sequence. By definition the difference between a true TIS from pseudo sites is that downstream of a TIS there is CDS (which shows codon structure), while upstream there is not. CDS and UTR show statistically different compositions. It is likely that the SVM exploits this difference for classification. We could hope to improve the kernel by reflecting the fact that CDS shifted by

SVM kernel function	data points used for training			
	400		1000	
simple polynomial	18.1%	d=2	16.0%	d=2
locality-improved	17.9%	d_1=3,l=2	15.9%	d_1=4,l=3
codon-improved	18.4%	d_1=1,l=2	15.6%	d_1=1,l=2

Table 2: Comparison kernel functions using differently sized subsets of the training data set, averaged over the six partitions. The percentages denote the overall classification errors that are achieved using the indicated supposedly optimal parameter settings. Note that the windows consist of $2l + 1$ nucleotides.

three nucleotides still looks like CDS. Therefore, we further modify the locality-improved kernel function to account for this translation-invariance. In addition to counting matching nucleotides on corresponding positions, we also count matches that are shifted by three positions. We call this kernel codon-improved.

Tables 2 and 1 suggest that this modification actually decreases performance. On the other hand, a similar modification to the simple polynomial kernel leads to a significant increase of recognition accuracy (data not shown). We therefore conclude that the process of learning some relevant features (e.g. subtile local correlations) is distorted by the modification. In contrast to the simple polynomial kernel, the locality-improved kernel seems to be rich enough to easily learn translation-invariance by itself, wherever this proves advantageous.

Nevertheless, both engineered kernel functions clearly outperform the NN as devised by Pedersen and Nielsen by reducing the overall number of misclassifications by about 20% (see Table 1). The SVM also beats the performance of positional conditional probabilities, which work surprisingly well when applied to larger windows than suggestest by Salzberg.

3 Discussion

First, we will briefly discuss applications of our TIS recognition method. This leads to potential paths of improvement. Finally, we suggest promising fields of application of techniques similar to those presented above.

TIS recognition can be used to improve reliability and accuracy of amino acid prediction from nucleotide sequences. The two main fields of application are ESTs and genomic sequences. For EST data, the program ESTScan aims at identifying CDS as accurately as possible. In a slight misuse of ESTScan, we investigate how well it finds the correct TIS within the set of 3312 mRNA-like sequences described in section 2.2. Results are shown in Table 3. On average, the program misses the true TIS position by 41.6 nucleotides. Both this figure and the table indicate that ESTScan could profit from a TIS recognition module. For genomic sequences and programs like GENSCAN, a similar situation could be expected.

However, caution would be necessary in order to use our method within a rigorous probabilistic framework like those of GENSCAN or ESTScan. The SVM (as well as Pedersen and Nielsen's NN) seems to exploit the different oligo-nucleotide preferences of CDS in comparison to UTRs. These preferences are already incorporated in GENSCAN and ESTScan, leading to probability distribution dependencies that must be taken into account. In order to avoid these dependencies, it would be easiest to restrict the sequence window presented to the SVM to the ribosome binding site. This would most probably lead to a decrease of classification performance. In addition, it would be desirable that the TIS recognition method computes probability values for potential TIS to be true TIS. Meanwhile, it should be useful to heuristically combine our TIS recognition with GENSCAN or ESTScan output. We plan to devote work to this area.

There are far too many interesting classification tasks in bioinformatics too be covered here, so we restrict ourselves to two of the most popular problems. First, we could imagine that the excellent protein classification performance of the Fisher kernel method developed by Jaakkola and Haussler [7] could still be improved by considering local amino acid correlations in a manner similar to our locality-improved kernel. Second, we believe that SVMs will prove successful for exploiting the information gathered with

ATG selection	overall	true TIS	pseudo sites	specificity	sensitivity
left	39.0%	93.7%	21.1%	8.9%	6.3%
right	18.1%	51.3%	7.3%	68.7%	48.7%
closest	21.4%	57.9%	9.4%	59.3%	42.1%

Table 3: Application of ESTScan for TIS recognition on the original set of 3312 sequences (cf. section 2.2). For each predicted CDS, an ATG triplet near the supposed start point of the CDS is selected as predicted TIS. Evaluation is shown for three different selection strategies. (Column labels are as in Table 1.)

DNA chips. Here, kernel functions could be engineered that reflect the structure of expression data as collection of unrelated time series. These are fields of further future work.

In summary, we have compared the performance of important methods for sequence classification on a bio-molecular problem of practical relevance. We show that SVMs are competitive to other, more frequently used machine learning methods and offer the unique advantage of an easy way to include prior knowledge to improve performance.

Acknowledgments This work was supported by the BMBF (TargId, 0311615), by the DFG (JA 379/5-2,7-1), and by the EC STORM project 25387. We thank Pedersen and Nielsen for providing their data sets and sharing unpublished data. We thank M. Schwan for help with the experiments.

References

[1] P. Agarwal and V. Bafna. Detecting non-adjoining correlations within signals in DNA. In *RECOMB'98*, 1998.

[2] P. Agarwal and V. Bafna. The Ribosome Scanning Model for Translation Initiation: Implications for Gene Prediction and Full-Length cDNA Detection. In *ISMB'98*, pages 2–7, 1998.

[3] C. M. Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press, 1995.

[4] B. Boser, I. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In *COLT*, pages 144–152, 1992.

[5] C. Burge and S. Karlin. Prediction of Complete Gene Structures in Human Genomic DNA. *JMB*, 268(1):78–94, 1997.

[6] C. Iseli, C. V. Jongeneel, and P. Bucher. ESTScan: a program for detecting, evaluating, and reconstructing potential coding regions in EST sequences. In *ISMB'99*, pages 138–147, 1999.

[7] T. Jaakkola, M. Diekhans, and D. Haussler. A discriminative framework for detecting remote protein homologies, 1998.

[8] T. S. Jaakkola and D. Haussler. Exploiting Generative Models in Discriminative Classifiers. In *NIPS*, 1998.

[9] M. Kozak. Interpreting cDNA sequences: some insights from studies on translation. *Mammalian Genome*, 7:563–574, 1996.

[10] J. Orr and K.-R. Müller, editors. *Neural Networks: Tricks of the Trade.* Springer, 1998.

[11] A. G. Pedersen and H. Nielsen. Neural Network Prediction of Translation Initiation Sites in Eukaryotes: Perspectives for EST and Genome analysis. In *ISMB'97*, pages 226–233, 1997.

[12] A. A. Salamov, T. Nishikawa, and M. B. Swindells. Assessing protein coding region integrity in cDNA sequencing projects. *Bioinformatics*, 14(5):384–390, 1998.

[13] S. L. Salzberg. A method for identifying splice sites and translational start sites in eukaryotic mRNA. *CABIOS*, 13(4):365–376, 1997.

[14] B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors. *Advances in Kernel Methods – Support Vector Learning.* MIT Press, 1999.

[15] B. Schölkopf, P. Simard, A. Smola, and V. Vapnik. Prior knowledge in support vector kernels. In *Advances in Neural Information Processing Systems 10*, pages 640–646. MIT Press, 1998.

[16] B. Schölkopf, A. Smola, R. Williamson, and P. Bartlett. New support vector algorithms. Technical report, NeuroColt2, 1998. to appear in Neural Computation.

[17] V. N. Vapnik. *The nature of statistical learning theory.* Springer-Verlag, 1995.