

ARTS: accurate recognition of transcription starts in human

Sören Sonnenburg¹, Alexander Zien^{2,3} and Gunnar Rätsch^{3,*}

¹Fraunhofer Institute FIRST, Kekuléstr. 7, Berlin, Germany, ²Max Planck Institute for Biological Cybernetics, Spemannstr. 38, Tübingen, Germany and ³Friedrich Miescher Laboratory, Max Planck Society, Spemannstr. 39, Tübingen, Germany

ABSTRACT

We develop new methods for finding transcription start sites (TSS) of RNA Polymerase II binding genes in genomic DNA sequences. Employing Support Vector Machines with advanced sequence kernels, we achieve drastically higher prediction accuracies than state-of-the-art methods.

Motivation: One of the most important features of genomic DNA are the protein-coding genes. While it is of great value to identify those genes and the encoded proteins, it is also crucial to understand how their transcription is regulated. To this end one has to identify the corresponding promoters and the contained transcription factor binding sites. TSS finders can be used to locate potential promoters. They may also be used in combination with other signal and content detectors to resolve entire gene structures.

Results: We have developed a novel kernel based method – called *ARTS* – that accurately recognizes transcription start sites in human. The application of otherwise too computationally expensive Support Vector Machines was made possible due to the use of efficient training and evaluation techniques using *suffix tries*. In a carefully designed experimental study, we compare our TSS finder to state-of-the-art methods from the literature: *McPromoter*, *Eponine* and *FirstEF*. For given false positive rates within a reasonable range, we consistently achieve considerably higher true positive rates. For instance, *ARTS* finds about 35% true positives at a false positive rate of 1/1000, where the other methods find about a half (18%).

Availability: Datasets, model selection results, whole genome predictions, and additional experimental results are available at <http://www.fml.tuebingen.mpg.de/raetsch/projects/arts>

Contact: Gunnar.Raetsch@tuebingen.mpg.de

1 INTRODUCTION

Arguably the most important information about genomic DNA is the location of genes that encode proteins. For further analysis of the genes it is necessary to find their promoters and the contained binding sites of transcription factors, which are responsible for regulating the transcription of the gene.

Transcription start sites are located in the core promoter region and are usually determined by aligning complete mRNA or 5'-end EST sequences (for instance obtained by 5' RACE) against the genome. Note that protein sequences and other ESTs are not sufficient for this task, since they typically start downstream of the TSS. For some species including human, large scale sequencing projects of complete mRNAs have been undertaken, but many low

copy genes still evade being sequenced. In order to identify these genes and their promoter regions, computational TSS finding or better experimental techniques are the only way out.

Moreover, in the vast majority of species the identification of promoters must be accomplished without the support of massive sequencing. One possibility is to exploit homology to well-characterized genes in other species. While this approach can work for common genes, for those genes specific to some species or some family of species it is likely to fail. This leaves a huge demand for accurate *ab initio* TSS prediction algorithms.

Consequently, a fairly large number of TSS finders (TSF) has been developed. Generally TSFs exploit that the features of promoter regions and the TSS are different from features of other genomic DNA. Many different features have been used for the identification: the presence of CpG islands, specific transcription factor binding sites (TFBS), higher density of predicted TFBSs, statistical features of proximal and core promoter regions and homology with orthologous promoters (see Bajic *et al.*, 2004; Werner, 2003) for two recent reviews on mammalian promoter recognition). Methods for recognizing TSSs employed neural networks, discriminant analysis, the Relevance Vector Machine (RVM), interpolated Markov models, and other statistical methods.

In a recent large scale comparison (Bajic *et al.*, 2004;) eight TSFs have been compared. Among the most successful ones were *Eponine* (Down and Hubbard, 2002) (which trains RVMs to recognize a TATA-box motif in a G+C rich domain), *McPromoter* (Ohler *et al.*, 2002) (based on Neural Networks, interpolated Markov models and physical properties of promoter regions) and *FirstEF* (Davuluri *et al.*, 2001) (based on quadratic discriminant analysis of promoters, first exons and the first donor site, using CpG islands). *DragonGSF* (Bajic and Seah, 2003) performs similarly well as the aforementioned TSFs (Bajic *et al.*, 2004). However, it uses additional binding site information based on the TRANSFAC data base (Matys *et al.*, 2006); thus it exploits specific information that is typically not available for unknown promoters. For this reason and also because the program is currently not publicly available, we exclude it from our comparison.¹

One characteristic of TSFs is that they normally rely on the combination of relatively weak features such as physical properties of the DNA or the G+C-content. In none of the above-mentioned approaches the recognition of the actual transcription start site has been seriously considered. In this work we show that by using very recently developed discriminative sequence analysis techniques (Sonnenburg

¹ Further, unlike *DragonGSF* all of the above TSFs could – after retraining – be applied to genomes other than human, where only a few or no TF binding sites are known.

*To whom correspondence should be addressed.



Fig. 1. Given two sequences x_1 and x_2 of equal length, the WD kernel with shift consists of a weighted sum to which each match in the sequences makes a contribution $\gamma_{k,p}$ depending on its length k and relative position p , where long matches at the same position contribute most significantly. The γ 's can be computed from the β 's and δ 's in (2). The spectrum kernel is based on a similar idea, but it only considers substrings of a fixed length and the contributions are independent of the relative positions of the matches to each other.

et al., 2005) – which previously were only tractable on a much smaller scale (Rätsch *et al.*, 2005) – we can drastically improve the performance of TSS recognition.

The remainder of the paper is structured as follows: In Section 2 we discuss the features of the sequences and kernels that we use for learning in order to recognize transcription start sites. In Section 3 we discuss techniques related to the kernels and Support Vector Machine training and evaluation, which were necessary in order to perform the experiments. We discuss the experimental setup and the data generation for a large scale comparison of our method ARTS with other TFSs, and provide experimental results in Sections 4 and 5. We conclude with a discussion and an outlook.

2 BASICS, FEATURES AND KERNELS

Binary classification methods aim at estimating a classification function $f: \mathcal{X} \rightarrow \{\pm 1\}$ using labeled training data from $\mathcal{X} \times \{\pm 1\}$ such that f will correctly classify most unseen examples (test data). In our case, the input space \mathcal{X} will contain sequences $\{A, C, G, T\}^N$ centered at any genomic position, while the labels $+1$ or -1 indicate whether these positions are true TSS or decoy sites, respectively.

2.1 SVMs and Kernels

We use Support Vector Machines (Cortes and Vapnik, 1995) (SVMs) for two reasons. First, they exhibit a very competitive classification performance, since over-fitting is prevented by well-controllable regularization and training is not hampered by any local minima. Second, SVMs can conveniently be adapted to the problem at hand by designing appropriate kernel functions. The kernel function shortcuts mapping points to a feature space \mathcal{F} via an arbitrary function Φ and computes dot products in that space via $k(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$. Particularly advantageous for TSS recognition is the possibility to build complex modular kernel functions by combining several simpler ones. This way of combining different pieces of information has been shown to be very powerful (e.g. [10]).

For a test example \mathbf{x} the classification function generated by an SVM can be written as

$$f(\mathbf{x}) = \text{sign} \left(\sum_{i=1}^N \alpha_i y_i k(\mathbf{x}_i, \mathbf{x}) + b \right), \quad (1)$$

where $y_i \in \{\pm 1\}$ is the label of training example \mathbf{x}_i ($i = 1, \dots, N$). The coefficients α_i and the bias b are the results of SVM training. Please note that (1) is equivalent to $f(\mathbf{x}) = \mathbf{w} \cdot \Phi(\mathbf{x}) + b$, where $\mathbf{w} = \sum_{i=1}^N \alpha_i y_i \Phi(\mathbf{x}_i) \in \mathcal{F}$. The SVM constructs a maximum margin linear classifier in the Φ -space. The computation of SVMs only depends on the inner products of training examples; therefore it is usually sufficient to specify the kernel function $k(\mathbf{x}, \mathbf{x}') = \Phi(\mathbf{x}) \cdot \Phi(\mathbf{x}')$ that computes the inner products in feature space.

2.2 Features for TSS recognition

As most other TSFs our method combines several features, thereby utilizing prior knowledge about the structure of transcription start sites. We put, however, particular care in analyzing the actual transcription start site. We have considered the following:

- The TSS is only determined up to a small number of base pairs. Further, nearby binding sites may also not be positionally fixed. In order to model the actual TSS site, we thus need a set of features that are approximately localized and allow for limited flexibility. We have recently proposed a kernel — the extended Weighted Degree kernel *with shifts* (WD_S) — for the identification of alternatively spliced exons [16] which is also well suited for this task:

$$k(\mathbf{x}, \mathbf{x}') = \sum_{k=1}^K \beta_k \sum_{l=1}^{L-k+1} \sum_{\substack{s=0 \\ s+l \leq L}}^S \delta_s \mu_{k,l,s,\mathbf{x},\mathbf{x}'},$$

$$\mu_{k,l,s,\mathbf{x},\mathbf{x}'} = \mathbf{I}(\mathbf{u}_{k,l+s}(\mathbf{x}) = \mathbf{u}_{k,l}(\mathbf{x}')) + \mathbf{I}(\mathbf{u}_{k,l}(\mathbf{x}) = \mathbf{u}_{k,l+s}(\mathbf{x}')), \quad (2)$$

where $\beta_k = 2(K - k + 1)/(K(K + 1))$, $\delta_s = 1/(2(s + 1))$ and $\mathbf{u}_{k,l}(\mathbf{x})$ is the subsequence of \mathbf{x} of length k that starts at position l . The idea is to count the matches between two sequences \mathbf{x} and \mathbf{x}' between the words $\mathbf{u}_{k,i}(\mathbf{x})$ and $\mathbf{u}_{k,i}(\mathbf{x}')$ where $\mathbf{u}_{k,i}(\mathbf{x}) = x_i x_{i+1} \dots x_{i+k-1}$ for all i and $1 \leq k \leq K$. The parameter k denotes the length of the words to be compared, and S is the maximum distance by which a sequence is shifted. See Figure 1 and [16] for details.

- Upstream of the TSS lies the promoter, which contains transcription factor binding sites. Comparing different promoters, it was noted that the order of TFBS can differ quite drastically. Thus, we use the so-called spectrum kernel [11] on a few hundred bps upstream of the TSS. The spectrum kernel is typically used to recognize regions in which certain k -mers are over- or under-represented (“content sensors”):

$$k(\mathbf{x}, \mathbf{x}') = \sum_{\sigma \in \Sigma^k} \#\{\sigma \text{ appears in } \mathbf{x}\} \cdot \#\{\sigma \text{ appears in } \mathbf{x}'\},$$

where $\#\{\sigma \text{ appears in } \mathbf{x}\}$ is the number of times a k -mer σ appears as a substring in \mathbf{x} . Since it does not preserve the information where the subsequences are located, it may not be appropriate for modeling localized signal sequences such as the actual transcription start site.

- Downstream of the TSS follows the 5' UTR, and further downstream introns and coding regions. Since these sequences may significantly differ in oligo-nucleotide composition from intergenic or other regions, we use a second spectrum kernel for the downstream region.

Table 1. Parameters of the combined kernels and the SVM for TSS recognition. The ranges are specified according to our prior knowledge or intuition. A parameter value of 0 marked with * means that the sub-kernel is excluded from the combined kernel

Parameter	Set of values	Init. guess	Opt. value	Explanation
TSS signal (weighted degree with shift):				
• r-start	{-100, -90, ..., -10}	-50	-70	start of considered sequence region
• r-end	{+10, +20, ..., +100}	+50	+70	end of considered sequence region
• order	{0*, 2, ..., 24}	10	24	length of substrings compared
• shift	{4, 8, ..., 48}	20	32	positional shift (base pairs)
Promoter (spectrum):				
• r-start	{-1000, -900, ..., -100} ∪ {-150}	-600	-600	start of considered sequence region
• r-end	{-200, -150, ..., +200}	0	0	end of considered sequence region
• order	{0*, 1, ..., 6}	3	4	length of substrings considered
1 st exon (spectrum):				
• r-start	{-100, -50, ..., +300}	+100	0	start of considered sequence region
• r-end	{+100, +200, ..., +1000}	+600	+900	end of considered sequence region
• order	{0*, 1, ..., 6}	3	4	length of substrings considered
angles (linear):				
• r-start	{-1000, -900, ..., -200}	-600	-600	start of considered sequence region
• r-end	{-600, -500, ..., +200}	-100	-100	end of considered sequence region
• smoothing	{0*, 10, ..., 100}	50	70	width of smoothing window
Energies (linear):				
• r-start	{-1000, -900, ..., -200}	-600	-	start of considered sequence region
• r-end	{-600, -500, ..., +200}	-100	-	end of considered sequence region
• smoothing	{0*, 10, ..., 100}	50	0*	width of smoothing window
SVM: • C	{2 ^{-2.5} , 2 ⁻² , ..., 2 ^{+2.5} }	2 ⁰	2 ¹	regularization constant

- The 3D structure of the DNA near the TSS must allow the transcription factors to bind to the promoter region and the transcription to be started. To implement this insight, we apply two linear kernels to the sequence of twisting angles and stacking energies. Both properties are assigned based on dinucleotides as done by the *emboss* program *btwisted*.² The fourth and fifth kernel are then computed as the inner product $k(\mathbf{x}, \mathbf{x}') = \mathbf{x} \cdot \mathbf{x}'$, where \mathbf{x} is derived from a sequence of DNA twisting angles and stacking energies, respectively, by smoothing with a sliding window and using only every 20th of the resulting values.

The combined kernel is simply the sum of all sub-kernels, which is equivalent to appending the feature vectors in feature space. The sub-kernels can be expected to be of different importance for the overall performance; thus, it may seem appropriate to use a weighted sum. Experiments to verify this (not shown) indicated that a uniform weighting performs just as well as reducing the weights for the less important sub-kernels. An explanation for this may be that the SVM is able to learn relative weights itself. The only requirement is that the (weighted) function values of the sub-kernels are on a comparable scale; otherwise, those on a low scale are effectively switched off.

Note that we normalized all kernels with the exception of the linear kernels such that the vectors $\Phi(\mathbf{x})$ in feature space have unit length. This can be done efficiently by redefining the kernel as follows:

$$\tilde{k}(\mathbf{x}, \mathbf{x}') = \frac{k(\mathbf{x}, \mathbf{x}')}{\sqrt{k(\mathbf{x}, \mathbf{x})k(\mathbf{x}', \mathbf{x}')}}. \quad (3)$$

² <http://www.hgmp.mrc.ac.uk/Software/EMBOSS/Apps/btwisted.html>

This normalization solves convergence problems of SVM optimizers and balances the importance of the kernels among each other. In total we combine five kernels which each have several parameters as listed in Table 1.

3 EFFICIENCY CONSIDERATIONS

Our model is complex in that it consists of several sophisticated kernels applied to rather long stretches of DNA. Furthermore, we have to train it on as many examples as possible in order to attain a high prediction accuracy.³ Even with highly optimized general purpose SVM packages like *LibSVM* or *SVM^{light}*, training and tuning our model with tens of thousands of points is intractable. The main reason is that the kernel computation, in particular of the WD kernel with shift, is very expensive (single kernel computation $\mathcal{O}(KLS)$). In addition, many kernel elements need to be computed several times when the kernel cache is not large enough, which is quite likely with $\gg 10,000$ examples. However, fast training is possible without kernel caching, if the SVM output for any training point \mathbf{x}_i , i.e. $\mathbf{w} \cdot \Phi(\mathbf{x}_i)$, can be computed efficiently. In the following subsection, we show how algorithms can be modified to take advantage of fast computations of $\mathbf{w} \cdot \Phi(\mathbf{x}_i)$ during training and testing. In the second subsection we show how this can be accomplished for the different kernels that we use.

³For instance on a splice site recognition task we were able to reduce the error rate by 20% when doubling the amount of training data – over a wide range of training set sizes (Sonnenburg et al., 2006).

Algorithm 1 Outline of the decomposition algorithm that exploits the fast computations of linear combinations of kernels (e.g. by suffix tries).

```

 $f_i = 0, \alpha_i = 0$  for  $i = 1, \dots, N$ 
for  $t = 1, 2, \dots$  do
  Check optimality conditions and stop if optimal
  select  $Q$  variables  $i_1, \dots, i_Q$  based on  $\mathbf{f}$  and  $\alpha$ 
   $\alpha^{old} = \alpha$ 
  solve SVM dual w.r.t. the selected variables and update  $\alpha$ 
  generate data structures to prepare efficient computation of
     $g(\mathbf{x}) = \sum_{q=1}^Q (\alpha_{i_q} - \alpha_{i_q}^{old}) y_{i_q} k(\mathbf{x}_{i_q}, \mathbf{x})$ 
  update  $f_i = f_i + g(\mathbf{x}_i)$  for all  $i = 1, \dots, N$ 
end for

```

3.1 Faster SVM training and evaluation

As it is not feasible to use standard optimization toolboxes for solving large scale SVM training problems, decomposition (also called *chunking* in the machine learning literature) techniques are used in practice. Most decomposition algorithms work by first selecting a working set $W \subseteq \{1, \dots, N\}$ with (the indices of) Q variables of the N training points based on the current solution. Then the corresponding reduced problem is solved with respect to the working set variables. These two steps are repeated until some optimality conditions are satisfied [see e.g. Joachims (1998)].

Efficient Updates in Decomposition Algorithms For selecting the working set and checking the termination criteria in each iteration, the vector \mathbf{f} with $f_i = \sum_{j=1}^N \alpha_j y_j k(\mathbf{x}_i, \mathbf{x}_j)$, $i = 1, \dots, N$, is needed. To avoid computation of \mathbf{f} in every iteration one typically starts with $\mathbf{f} = 0$ and computes updates of \mathbf{f} on the changed variables:

$$f_i \leftarrow f_i^{old} + \sum_{j \in W} (\alpha_j - \alpha_j^{old}) y_j k(x_i, x_j), \quad \forall i = 1, \dots, N,$$

where $Q = |W|$ is the size of the working set. One typically uses kernel-caching to reduce the computational effort of this operation, which is, however, is not sufficient in the case of large training sets. Fortunately, for all kernels considered in this work we can efficiently compute linear combinations of kernel elements, i.e. $\mathbf{w} \cdot \Phi(\mathbf{x})$, where \mathbf{w} is of the form $\sum_i \alpha_i y_i \Phi(\mathbf{x}_i)$. Algorithm 1 implements a simple idea how to use this to speedup SVM training.

For all considered kernels, the operation $\mathbf{w} \cdot \Phi(\mathbf{x})$ is almost as cheap as computing the dot product of two $\Phi(\mathbf{x})$'s. Hence, Algorithm 1 leads to a speedup of up to factor Q . Note that creating the data structure for Q examples (e.g. the below-mentioned suffix tries) can be expensive, however, it is a fixed cost per iteration. If the number of examples is large enough, then the speedup of the evaluation leads to a great advantage.

Efficient Evaluation In the application we have in mind we need to compute predictions for every position in the human genome ($\approx 7 \cdot 10^9$). For kernel methods that generate several thousands of support vectors, each of which is of length one thousand, this would mean more than 10^{16} floating point operations. This is too much even for modern computer cluster systems. By using the same idea as in training we can efficiently compute the SVM prediction $f(\mathbf{x}) = \sum_{i=1}^N \alpha_i y_i k(\mathbf{x}_i, \mathbf{x})$ for new sequences. This leads to a speedup of M , where M is the number of Support Vectors (with $\alpha_i \neq 0$), and makes the genome-wide computation of promoter predictions

feasible — with still ≈ 350 h computing time for the entire human genome.

3.2. Fast string kernel computations

All considered kernels correspond to a feature space \mathcal{F} that can be very high dimensional. For instance in the case of the WD kernel on DNA sequences of length 100 with $K = 20$, the corresponding feature space is 10^{14} dimensional (one feature per position and possible k -mer, $1 \leq k \leq K$). However, most dimensions in the feature space are not used since only a few of the many different k -mers actually appear in the sequences. An appropriate choice of the data representation is therefore crucial for fast algorithms. If the data can be efficiently represented as sparse vectors in the feature space \mathcal{F} , one achieves significant speedups in SVM training and testing.

Spectrum kernels with explicit feature maps If the dimensionality of the feature space is small enough, then one can store the whole vector $\mathbf{v} \in \mathcal{F}$ in memory and perform direct operations on its elements. This is true for our linear kernels (4 and 5) and also the spectrum kernel for relatively short K -mers (e.g. $K = 6$ leads to a 4096 dimensional space). For the latter case one may first preprocess the sequences \mathbf{x} into a sparse vector $\Phi(\mathbf{x})$ and later perform computations with mixed sparse and full vectors, which can be implemented very efficiently. This approach has exponentially growing memory demands ($\mathcal{O}(|\Sigma|^K)$), but is very fast and best suited for instance for the spectrum kernel on DNA sequences with $K \leq 14$ and on protein sequences with $K \leq 6$.

WD kernels with suffix tries The difference between the WD kernel (without shifts) and the spectrum kernel is (a) the position dependence and (b) the consideration of K -mers vs. $1, \dots, K$ -mers, i.e. also including subsequences of the K -mers. If one would use a weighted sum of spectrum kernels for all degrees $\leq K$ at every position of the sequence, then it is equivalent to the WD kernel (Sonnenburg *et al.*, 2005). So in principle we could apply the idea used for the Spectrum kernel to speedup the WD kernel as well. However, when using long K -mers (e.g. $K = 20$) the memory demand becomes intractable and the sparse weight vectors need to be stored and operated with more efficiently. In (Sonnenburg *et al.*, 2006) we have suggested to use *suffix tries*, i.e. trees that store weights not only at the leaves but also at internal nodes. The idea is to use one trie of degree four ($|\Sigma|$) and depth K per position in the sequence. A node in the trie at depth k is addressed by a k -mer and stores its associated value. See Figure 2 for illustration. Note that we can easily add several sequences to the trie and the worst-case cost for performing a lookup operation is $\mathcal{O}(K)$. This is the key to the speedup of SVM training and evaluation.

Please note that the tries for the WD kernel *with shifts* can be analogously constructed. Now a string has to be found several times. Either we store it in several neighboring trees (with decaying weights) or we store it only once and query the neighboring trees during lookup operations. So far the latter version was used, which turned out to require too much computing time. The first option, however, requires rather large tries as each string is stored in several tries. This particularly matters during testing when the trie needs to store all subsequences of support vectors and one tree can grow to more than 200Mb. One therefore cannot build all trees at once, but only sequentially. For considerations of how to extend this approach to mismatching k -mers see (Sonnenburg *et al.*, 2005).

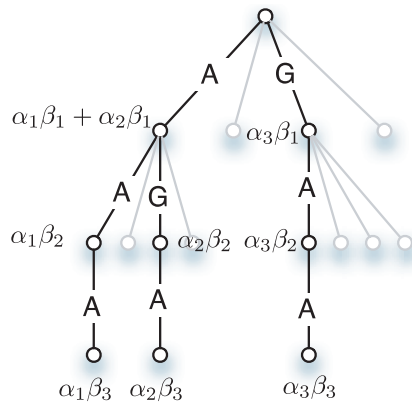


Fig. 2. Three sequences AAA, AGA, GAA being added to the trie. The figure displays the resulting weights at the nodes, where the β 's correspond to a weighting over the the depth of the trie and the α 's are the sequence weights.

Implementations The implementation of the kernel functions, their efficient computation during SVM training (including Multiple Kernel Learning (Sonnenburg *et al.*, 2006)) and evaluation is implemented in C++ and will be made available as part of a kernel learning toolbox called *Shogun* interfacing to R, Octave, Matlab and Python (see <http://www.fml.mpg.de/raetsch/projects/shogun>).

4 TRAINING AND TUNING THE MODEL

For training a TSF and selecting its model parameters (“model selection”) it is crucial to use proper training and testing data. In Section 4.1, we elaborate on the generation of suitable datasets, including all relevant steps of data pre-processing. We also explain in necessary detail how we perform the model selection (Section 4.3).

4.1 Datasets

Both for training our TSS finder and for assessing its accuracy we need known TSSs as well as known non-TSSs.

To generate TSS data for training, we use dbTSS (Suzuki *et al.*, 2002) version 4 (“dbTSSv4”), which is based on the human genome sequence and annotation version 16 (“hg16”). It contains transcription start sites of 12763 RefSeq genes (Kim *et al.*, 2005). First we extract RefSeq identifiers from dbTSSv4 and then obtain the corresponding mRNA sequences using NCBI nucleotide batch retrieval.⁴ Next, we align these mRNAs to the hg16 genome using BLAT [9].⁵ From dbTSS we extracted putative TSS positions (Field: Position of TSS) which we compared with the best alignment of the mRNA. We discard all positions that do not pass all of the following checks: 1. Chromosome and strand of the TSS position and of the best BLAT hit match. 2. The TSS position is within 100 base pairs from the gene start as found by the BLAT alignment. 3. No already processed putative TSS is within 100bp of the current one. This procedure leaves us with 8508 genes, each annotated with

⁴ <http://ncbi.nih.gov/entrez/batchentrez.cgi?db=Nucleotide>

⁵ We used the options `-tileSize=16 -minScore=100 -minMatch=4 -minIdentity=98 -t=dna -q=rna`.

gene start and end. To generate positive training data, we extract windows of size $[-1000, +1000]$ around the TSS.

To discriminatively train a classifier one also needs to generate “negative” data. However there is no single natural way of doing this: since there are further yet unknown TSS hidden in the rest of the genome, it is dangerous to sample negative points randomly from it. So we choose to proceed similarly to Bajic *et al.*, (2004) by extracting “negative” points (again, windows of size $[-1000, +1000]$) from the interior of the gene. More precisely, we draw 10 negatives at random from locations between 100 bp downstream of the TSS and the end of the gene.⁶ We finally obtain 8508 positive and 85042 negative examples, of which we will use 50% for training a TSS classifier and 50% for validating it. The final evaluation is done on a differently generated test data set (cf. Section 5.2).

4.2 Performance measures

We use two established measures of performance as guidance for model selection and, later on, for evaluating our success. The sensitivity (or recall) is defined as the fraction of correctly classified positive examples among the total number of positive examples, i.e. it equals the true positive rate $TPR = TP/(TP + FN)$. Analogously, the fraction $FPR = FP/(TN + FP)$ of negative examples wrongly classified positive is called the false positive rate. Plotting FPR against TPR results in the Receiver Operator Characteristic Curve (ROC) (Metz, 1978; Fawcett, 2003). Plotting the true positive rate against the positive predictive value (also precision) $PPV = TP/(FP + TP)$, i.e. the fraction of correct positive predictions among all positively predicted examples, one obtains the Precision Recall Curve (PRC) (see e.g. [4]). For both graphs, the area under the curve is a useful single-number performance measure, which we refer to as *auROC* (for area under ROC) and *auPRC*, respectively.

4.3 Model selection

As seen before (Table 1), there are many (in fact, 17) parameters that need to be set to reasonable values in order for our approach to work well. We treat this as a model selection problem: each parameter setting corresponds to a set of assumptions, i.e. a model, on what distinguishes the surroundings of TSS from other genomic loci. We want to select the closest approximation (within the framework defined by the kernel function) to reality, which can be identified by having the best predictive power. Thus we train the SVM with different parameter settings and assess the resulting prediction performance on a separate validation set.

While model selection is often done by trying all points on a regular grid in the space of parameters, this is computationally infeasible for more than a few parameters. Therefore, we resort to iterated independent axis-parallel searches. First, we specify a start point in parameter space based on prior knowledge and intuition. Then, in each round candidate points are generated by changing any single parameter to any value from a pre-defined small set; this is done for every parameter independently. Finally, the new parameter setting is assembled by choosing for each parameter the value that performed best while leaving the other parameter values unchanged.

We choose the model that yields the highest *auROC* on the validation set. It achieves 93.99% *auROC* and 78.20% *auPRC*

⁶ If a gene is too short, fewer or even no negative examples are extracted from that particular gene.

Table 2. Results obtained by removing sub-kernels. The energies kernel is already turned off by the model selection

Subkernel	Area under ROC	Area under PRC
w/o TSS signal	90.75%	70.72%
w/o promoter	93.33%	74.94%
w/o 1 st exon	92.76%	74.94%
w/o angles	93.99%	78.26%
Complete	93.99%	78.20%

Table 3. Results obtained when only a single specific sub-kernel is used. The actual TSS signal discriminates strongest, but also the 1st exon carries much discriminative information

Subkernel	Area under ROC	Area under PRC
TSS signal	91.42%	69.38%
Promoter	86.55%	55.33%
1 st exon	88.54%	64.29%
angles	45.31%	7.86%

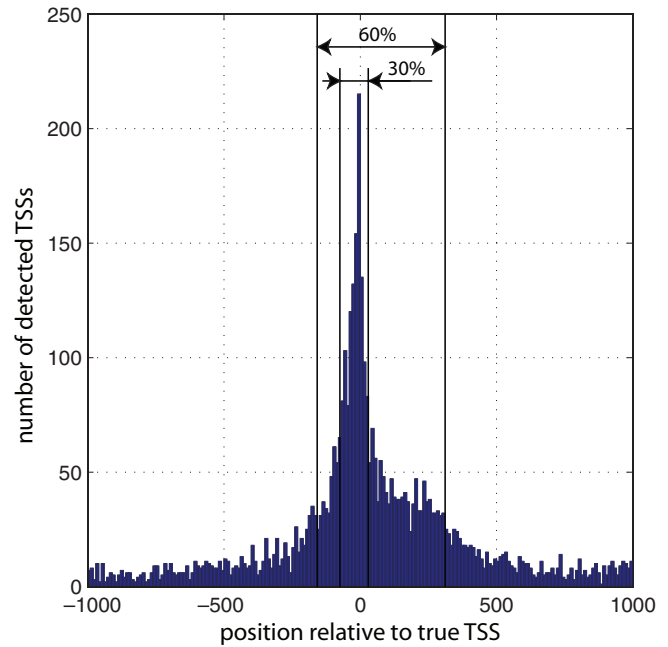
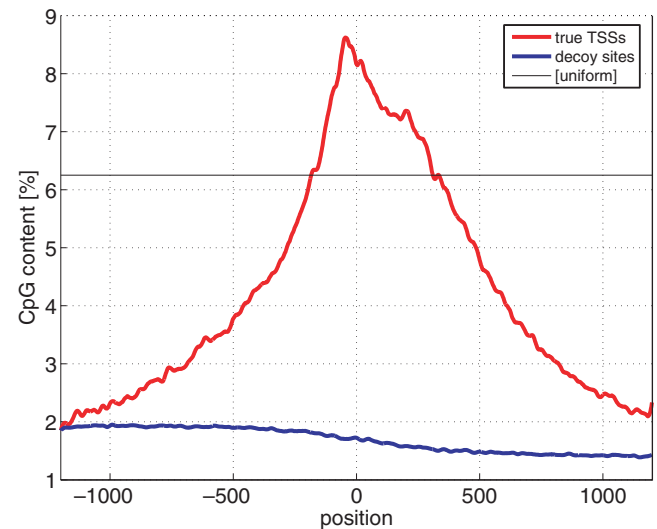
(99.93% and 99.91% on the training data, respectively). The selected parameter settings are shown in the second but last column of Table 1.

4.4 Importance of the kernels

In addition to optimizing the parameter settings of all sub-kernels, we investigate whether and how much each sub-kernel contributes to the overall classification. To do so, we remove each sub-kernel and retrain the remaining model (with all other parameters kept fixed at the selected values). The accuracies obtained on the validation set are shown in Table 2. Removing the WD_S kernel, which models the signal at $[-70, +70]$ around the TSS, decreases the performance of the classifier most, although it still performs rather well (auROC > 90%). The 1st exon kernel, which models the 4-mer nucleotide frequency in the range $[0, +900]$ downstream, appears to be of second most importance in our kernel ensemble. Removing the linear kernels, which take into account the binding energies and the twisting of the DNA, has almost no effect on the result.

A different view on the contribution of the individual kernels can be obtained by retraining single-kernel SVMs. The respective results are displayed in Table 3. Again the WD_S kernel contributes most, followed by the two spectrum kernels modeling the first exon and the promoter. The DNA twistedness angle-measure performs even worse than at random, probably because SVM's regularization parameter C was not properly tuned for the single kernel case.

For illustration we analyze in Figure 3 how the TSS signal predictions are localized relative to the true transcription start sites. We consider a window of ± 1000 around a true TSS and record the location of the maximal TSS signal prediction (TSS signal kernel only). Figure 3 displays a histogram of the recorded positions on our validation set. We observe an expected strong concentration near the true TSS. We also observe that the distribution is skewed – a possible explanation for this is offered by Figure 4:

**Fig. 3.** Localization of ARTS's TSS signal predictions: Shown is a histogram over the location with maximal score in a window ± 1000 around true TSSs. In 60% of the cases the predicted TSSs is within $[-150, +270]$ bp of the true TSS (30% within $[-70, +40]$).**Fig. 4.** Average positional frequency of CpG dinucleotides around true TSS and in decoy sequences (smoothed by convolution with a triangle of 39 bps length).

the predictor might be misled by the distribution of CpG islands, which is skewed in a similar manner.

In conclusion it seems to be the WD_S kernel that models the region around the TSS best. The relatively large shift of 32 found by the model selection suggests the existence of motifs located around the TSS at highly variable positions. Neither *epoinine* nor *FirstEF* model this regions explicitly. Thus, the WD_S kernel seems to be one of the reasons for ARTS' superior accuracy.

5 RESULTS AND DISCUSSION

We compare the performance of *ARTS*, our proposed TSS finding method, to that of *McPromoter* (Ohler *et al.*, 2002), *Eponine* (Down and Hubbard, 2002) and *FirstEF* (Davuluri *et al.*, 2001), which are among the best previously reported methods (Bajic *et al.*, 2004). The evaluation protocol highly affects a TSF comparison; we thus give a detailed explanation (Section 5.1) of the criteria we use.

5.1 Setup

As POL II binds to a rather vague region, there seems to be no single true TSS location, but rather regions of roughly $[-20, +20]$ bp constituting potential TSSs. For that reason one has to use evaluation criteria different from the ones used in standard two-class-classification. Bajic *et al.* (2004) suggest to cluster predicted TSS locations that have at most 1000bp distance to the neighboring locations. As evaluation criterion for each gene, they score a true positive if a prediction is located within ± 2000 bp of the true TSS (otherwise, a false negative is counted); false positives and true negatives are counted from the TSS position $+2001$ to the end of the gene. However, each TSF is tuned to obtain a maximum true positive rate at a *different* false positive rate. Hence, this criterion suffers from the fact that it remains unclear how to compare results when the sensitivity and positive predictive value are both different (cf. Table 2 in Bajic *et al.* (2004)).

To alleviate this problem and allow for direct comparison via Receiver Operator Characteristic and Precision Recall Curves (ROC and PRC) we propose a different evaluation criterion. We compute whole genome point-wise predictions, which are then converted into *non-overlapping* fixed length chunks (e.g. of size 50 or 500). Within each chunk the maximum TSF output is taken. One can think of this chunking⁷ process as a ‘‘lens’’, allowing us to look at the genome at a lower resolution. Obviously, for a chunk size of 1 this is the same as a point-wise TSS prediction. As ‘‘lenses’’ we use chunk sizes of 50 and 500. A chunk is labeled as +1 if it falls within the range ± 20 bp of an annotated TSS; chunks downstream of this range until the end of the gene are labeled -1 .

Note that according to the above scheme some TSS will label two chunks as positive. This, however, does not constitute a problem, as it is a rare event if the chunk size is large. Furthermore, it is not unlikely that a TSF predicts both chunks as positive, as the maximum of the scores within each chunk is taken. We also considered an alternative criterion, in which only the chunk in which the maximum TSFs output is larger is labeled as positive, whereas the other chunk is removed from the evaluation. As a downside, this introduces a labeling that is dependent on the TSF output (i.e. there is no ground truth labeling over all TSFs), and leads to only small variations (auROC/auPPV increased/decreased by $\leq 3.5\%$ for chunk size 50 and $\leq 1\%$ for *all* TSFs for chunk size 500). Chunks obtain negative labels if they were not positively labeled and fall within the range gene start $+20$ bp to gene end and are excluded from evaluation otherwise.

This way the labeling of the genome stays the same for all TSFs. Considering *all* TSSs in dbTSSv5 we obtain labelings for chunk size 50 (500) with 28,366 (16,892) positives and 16,593,892 (1,658,483) negatives where TSS fall into two chunks in 15,223

⁷ Not to be confused with the ‘‘chunking’’ (decomposition) algorithms used for SVM training.

Table 4. Evaluation of the Transcriptions Start Finder at a chunk size resolution of 50 on dbTSSv5 excluding dbTSSv4 using the area under the Receiver Operator Characteristic Curve and the area under the Recall Precision Curve (larger values are better). For details see text

dbTSSv5-dbTSSv4 evaluation on chunk size 50		
TSF	Area under ROC	Area under PRC
<i>Eponine</i>	88.48%	11.79%
<i>McPromoter</i>	92.55%	6.32%
<i>FirstEF</i>	71.29%	6.54%
<i>ARTS</i>	92.77%	26.18%

Table 5. Evaluation of the Transcriptions Start Finder at a chunk size resolution of 500 on dbTSSv5 excluding dbTSSv4 using the area under the Receiver Operator Characteristic Curve and the area under the Recall Precision Curve (larger values are better). For details see text

dbTSSv5-dbTSSv4 evaluation on chunk size 500		
TSF	Area under ROC	Area under PRC
<i>Eponine</i>	91.51%	40.80%
<i>McPromoter</i>	93.59%	24.23%
<i>FirstEF</i>	90.25%	40.89%
<i>ARTS</i>	93.44%	57.19%

(1,499) cases, covering in total 829,694,600 bp ($\approx 12\%$) of the human genome.⁸ In summary, the chunking allows for a controlled amount of positional deviations in the predictions. Unlike the clustering of predictions, it does not complicate the evaluation or hamper the comparability of TSF.

5.2 Test dataset

To allow for a fair comparison of promoter detectors, one needs to create a proper test set such that no promoter detector has seen the examples in training. We decide to take all ‘‘new’’ genes from dbTSSv5 (Yamashita *et al.*, 2006) (which is based on hg17) for which a representative TSS was identified (i.e., the field ‘‘The selected representative TSS’’ is not empty). From dbTSSv5 we remove all genes that already appear in dbTSSv4 according to the RefSeq NM identifier. To take care of cases in which IDs changed over time or are not unique, we also remove all genes from dbTSSv5 for which mRNAs overlap by more than 30%. This leads to a total of 1,024 TSS to be used in a comparative evaluation. The comparison is done on this test set using chunk sizes 50 and 500 as resolutions (cf. Section 5.1), which results in 1,588 (943) positives and 1,087,664 (108,783) negatives. In 816 (67) cases the TSS fall into two chunks.

5.3 TSF Performance evaluation

ROC curves are an established criterion for comparing classifiers. While they are meaningful on balanced datasets, they lose explanatory value when highly skewed datasets are compared. Exactly

⁸ Here we used the dbTSS field *Position(s) of 5' end(s) of NM_(or known transcript)* as the field *Selected representative TSS* is often empty.

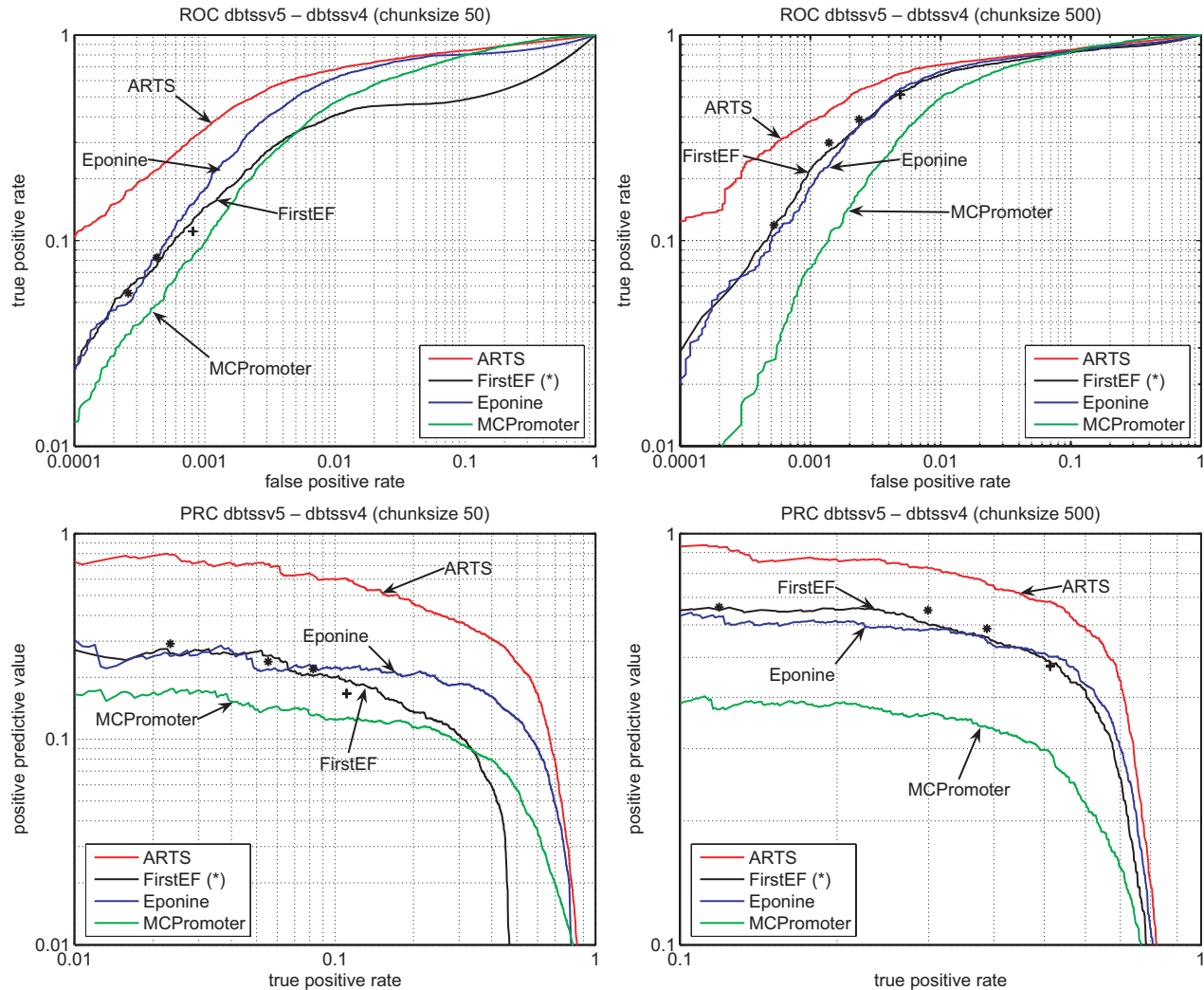


Fig. 5. Performance evaluation of the *ARTS*, *FirstEF*, *Eponine* and *McPromoter* TSS predictors. Evaluation was done on all genes whose TSS was found to be newly annotated in dbTSSv5 (i.e. genes whose TSS was not already in dbTSSv4). Receiver Operator Characteristic and Precision Recall Curves on decreased output resolution were computed (taking the maximum output within non-overlapping chunks of size 50 (left column) and 500 (right column) for more details see text). Windows were marked positive if a known TSS lies in a range of ± 20 bp and negative otherwise. Please note that the 'bumps' in the upper right corner in the *FirstEF*/*Eponine* plots for low window sizes are artifacts, caused by the method not giving predictions for every position. However the interesting area is in the left (lower false positive rate).

this is the case for TSS prediction, where we have about 7 billion loci of which, even for optimistic guesses, less than 3 million bps, i.e. only 0.05%, belong to true TSS. Let us consider a TSF that correctly classifies 100% of the true TSS sites (TPR) while wrongly classifying 1% of the non TSS loci (FPR). The area under the ROC would score at least 99% suggesting a particularly good classifier. However if only 0.05% of the negative examples (which in absolute values is 300 million) achieve a higher score than all of the positive examples (3 million), the area under the Precision Recall Curve will be less than 1%. For a reliably useful measure of prediction accuracy, we thus resort to the auPRC.

As the performance evaluation via ROC/PRC curve needs (genome-wide) real valued outputs for each TSF, we set the TSF's thresholds to the lowest acceptable values. *Eponine* is run with the options `-threshold 0.5`. As *McPromoter* provides the outputs for every tenth base pair we can use the unfiltered raw values

directly. *FirstEF* does not provide a single score as output, but probability scores for the promoter, exon, and donor. By default, a prediction is made if each probability equals or is larger than a pre-defined threshold (promoter: 0.4, exon: 0.5, donor: 0.4), which yields just a single point in the ROC and PRC space. We therefore set all thresholds to 0.001 and later use the product of the scores as a single output.⁹

Next we chunk the output, as described above in Section 5.1, and perform evaluation on all genes whose TSS was found to be newly annotated in dbTSSv5 (cf. Section 5.2). Tables 4 and 5 display the results for the performance measures area under the ROC and PRC curve for *ARTS*, *FirstEF*, *McPromoter*, and *Eponine*. Table 4 shows

⁹ As a validation we run *FirstEF* with the default settings and a variety of other thresholds. The obtained TPR/FPR and TPR/PPV values fit the curves produced using the single score extremely well (cf. Figure 5 below).

results for chunk size 50 and Table 5 for chunk size 500, corresponding to different levels of positional accuracy or resolution. In both cases our proposed TSS finder, *ARTS*, clearly outperforms the other methods in terms of both auROC and auPRC. This is also seen in Figure 5, which supplies detailed information on the true positive rates (top) and the positive predictive values (bottom) for a range of relevant true positive rates.

An interesting observation is that, judging by the auROC, *McPromoter* constitutes the second best performing TSF, while, on the other hand, it performs worst in the auPRC evaluation. An explanation can be found when looking at the ROC and PRC in Figure 5 where the left column displays ROC/PRC for chunk size 50 and the right for chunk size 500. Analyzing the ROC figures, we observe that *McPromoter* outperforms *FirstEF* and *Eponine* for false positive rates starting around 10% – a region contributing most to the auROC (note that both axes are on log scale). All of the three aforementioned promoter detectors perform similarly well. At a reasonable false positive level of 0.1% the TSFs perform as follows (chunk size 50): *ARTS* 34.7%, *Eponine* 17.9%, *FirstEF* 14.5%, and *McPromoter* 9.8%. Also note that the ROC curves for both chunk sizes are very similar, as the ROC curves are independent of class ratios between the negative and the positive class.¹⁰ On the other hand class ratios affect the PRC quite significantly: For instance, at a true positive rate of 50%, *ARTS* achieves a PPV of 23.5% for chunk size 50 and 68.3% for chunk size 500. *ARTS*'s ROC and PRC, however, constantly remains well above its competitors.

6 CONCLUSION

We have developed a novel and *accurate* transcription start finder, called *ARTS*, for the human genome. It is based on Support Vector Machines that previously were computationally too expensive to solve this task. It has therefore been an important part of our work to develop more efficient SVM training and evaluation algorithms using sophisticated string kernels. In a carefully designed experimental study we compared *ARTS* to other state-of-the-art transcription start finders that are used to annotate TSSs in the human genome. We show that *ARTS* by far outperforms all other methods: it achieves true positive rates that are twice as large as those of established methods. In the future, we plan to train and evaluate the *ARTS* system on other genomes and make the system as well as its predictions publicly available. It would be interesting to see how much of *ARTS*' higher accuracy can be translated into improved *ab initio* gene predictions.

ACKNOWLEDGEMENTS

The authors gratefully acknowledge partial support from the PASCAL Network of Excellence (EU #506778), DFG grants JA 379/13-2 and MU 987/2-1. We thank Uwe Ohler, Vladimir B. Bajic, Michael Zhang, Ramana Davuluri, Ivo Grosse, Koji Tsuda,

and Klaus Robert Müller for helpful and motivating discussions. We thank P. Philips, F. de Bona, C. Dieterich, and G. Zeller for proof-reading the manuscript.

REFERENCES

- V.B. Bajic and S.H. Seah. Dragon gene start finder: an advanced system for finding approximate locations of the start of gene transcriptional units. *Nucleic Acids Research*, 31:3560–3563, 2003.
- V. B. Bajic, S. L. Tan, Y. Suzuki, and S. Sugano. Promoter prediction analysis on the whole human genome. *Nat Biotechnol*, 22(11):1467–73, November 2004.
- C. Cortes and V. N. Vapnik. Support vector networks. *Machine Learning*, 20:273–297, 1995.
- J. Davis and M. Goadrich. The relationship between precision-recall and roc curves. Technical report #1551, University of Wisconsin Madison, January 2006.
- R. V. Davuluri, I. Grosse, and M. Q. Zhang. Computational identification of promoters and first exons in the human genome. *Nat Genet*, 29(4):412–417, December 2001.
- T. A. Down and T. J. P. Hubbard. Computational detection and location of transcription start sites in mammalian genomic DNA. *Genome Res*, 12:458–461, 2002.
- T. Fawcett. Roc graphs: Notes and practical considerations for data mining researchers. Technical report hpl-2003-4, HP Laboratories, Palo Alto, CA, USA, January 2003.
- T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In Claire Nédellec and Céline Rouveirol, editors, *Proceedings of the European Conference on Machine Learning*, pages 137–142, Berlin, 1998. Springer.
- W. J. Kent. BLAT—the BLAST-like alignment tool. *Genome Res*, 12(4):656–664, April 2002.
- G. R. Lanckriet, T. De Bie, N. Cristianini, M. I. Jordan, and W. S. Noble. A statistical framework for genomic data fusion. *Bioinformatics*, 20(16):2626–2635, 2004.
- C. Leslie, E. Eskin, and W. S. Noble. The spectrum kernel: A string kernel for SVM protein classification. In Russ B. Altman, A. Keith Dunker, Lawrence Hunter, Kevin Lauerdale, and Teri E. Klein, editors, *Proceedings of the Pacific Symposium on Biocomputing (PSB)*, pages 566–575. World Scientific, 2002.
- V. Matys, O.V. Kel-Margoulis, E. Fricke, I. Liebich, S. Land, A. Barre-Dirrie, I. Reuter, D. Chekmenev, M. Krull, K. Hornischer, N. Voss, P. Stegmaier, B. Lewicki-Potapov, H. Saxel, A.E. Kel, and E. Wingender. TRANSFAC and its module TRANSCompel: Transcriptional gene regulation in eukaryotes. *Nucleic Acids Res.*, 34:D108–110, 2006.
- C. E. Metz. Basic principles of ROC analysis. *Seminars in Nuclear Medicine*, VIII(4), October 1978.
- U. Ohler, G. C. Liao, H. Niemann, and G. M. Rubin. Computational analysis of core promoters in the drosophila genome. *Genome Biol*, 3(12):RESEARCH0087, 2002.
- Kim D. Pruitt, Tatiana Tatusova, and Donna R. Maglott. NCBI Reference Sequence (RefSeq): a curated non-redundant sequence database of genomes, transcripts and proteins. *Nucl. Acids Res.*, 33(S1):D501–504, 2005.
- G. Rättsch, S. Sonnenburg, and B. Schölkopf. RASE: Recognition of alternatively spliced exons in *C. elegans*. *Bioinformatics*, 21(Suppl. 1):i369–i377, June 2005.
- S. Sonnenburg, G. Rättsch, S. Schäfer, and B. Schölkopf. Large scale multiple kernel learning. *Journal of Machine Learning Research*, 2006. Accepted.
- S. Sonnenburg, G. Rättsch, and B. Schölkopf. Large scale genomic sequence SVM classifiers. In S. Dzeroski, editor, *Proceedings of the 22nd International Conference on Machine Learning, ICML*. ACM Press, 2005.
- Y. Suzuki, R. Yamashita, K. Nakai, and S. Sugano. dbTSS: Database of human transcriptional start sites and full-length cDNAs. *Nucleic Acids Res*, 30(1):328–331, January 2002.
- T. Werner. The state of the art of mammalian promoter recognition. *Brief Bioinform*, 4(1):22–30, March 2003.
- R. Yamashita, Y. Suzuki, H. Wakaguri, K. Tsuritani, K. Nakai, and S. Sugano. dbTSS: Database of human transcription start sites, progress report 2006. *Nucleic Acids Res*, 34:D86–89, January 2006. Database issue.

¹⁰ They vary very slightly as for smaller chunk sizes TSS more often fall into two chunks.