A Class of Edit Kernels for SVMs to Predict Translation Initiation Sites in Eukaryotic mRNAs

Haifeng Li and Tao Jiang
Department of Computer Science and Engineering
University of California, Riverside
{hli, jiang}@cs.ucr.edu

Abstract

The prediction of *translation initiation sites* (TISs) in eukaryotic mRNAs has been a challenging problem in computational molecular biology. In this paper, we present a new algorithm to recognize TISs with a very high accuracy. Our algorithm includes two novel ideas. First, we introduce a class of new sequence-similarity kernels based on string edit, called the *edit kernels*, for use with *support vector machines* (SVMs) in a discriminative approach to predict TISs. The edit kernels are simple and have significant biological and probabilistic interpretations. Although the edit kernels are not positive definite, it is easy to make the kernel matrix positive definite by adjusting the parameters. Second, we convert the region of an input mRNA sequence downstream to a putative TIS into an amino acid sequence before applying SVMs to avoid the high redundancy in the genetic code. The algorithm has been implemented and tested on previously published data. Our experimental results on real mRNA data show that both ideas improve the prediction accuracy greatly and our method performs significantly better than those based on neural networks and SVMs with polynomial kernels or Salzberg kernel.

Keywords: Translation initiation site, support vector machine, edit distance, machine learning, mRNA

1 Introduction

One of the main goals of the Human Genome Project is to provide a complete list of annotated genes to serve as a "periodic table" for biomedical research. The identification of coding regions in uncharacterized eukaryotic DNA sequences is a central problem in gene prediction.

Many algorithms and systems have been developed to automatically find the components of a gene, which include *translation initiation sites* (TISs), exon-intron splice sites, promoters, poly-adenylation signals, and CpG islands.

Although the algorithms (*e.g.* GENSCAN [8]) for finding the internal coding exons of a gene have reached a high degree of sophistication and accuracy, finding translation initiation sites that encode the start of protein translation, still remains a challenge. The codon ATG is the most commonly used start codon. [1] Usually, the initiation of translation starts at the first ATG codon in an mRNA. However, sometimes a downstream ATG is selected due to leaky scanning, reinitiation, and internal initiation of translation (this happens only for some viral mRNAs), *etc.* [25, 26]. According to [24, 45], downstream ATGs are used as start codons in less than 10% of investigated eukaryotic mRNAs. So, it seems that we could easily obtain an accuracy of more than 90% in the prediction of TISs by simply selecting the first ATG, given complete and error-free mRNA sequences. However, it has been reported that, in the GenBank nucleotide data that are annotated as being equivalent to mature mRNAs, almost 40% of the sequences contain upstream ATGs [31]. This problem is enhanced when using unannotated genomic data and when analyzing expressed sequence tags (ESTs), which are single-pass partial sequences derived from cDNAs and are usually error-prone. All these problems make it very difficult to predict TISs accurately.

The recognition of TISs has been extensively studied by using biological approaches, machine learning, and statistical models. In 1987, Kozak found that an ATG codon in a very weak context is not likely to be the start site of translation [22]. The optimal context for initiation of translation in vertebrate mRNA is GCCACCatgG. Within this *consensus* motif, nucleotides in two highly conserved positions exert the strongest effect: a G residue following the ATG codon (position +4) and a purine, preferably A, three nucleotides upstream (position −3). However, such a consensus alone is not sufficient to identify the ATG initiator codon [25, 26]. For example, after an 80S ribosome translates the first *small* open reading frame and reaches a stop codon, the 40S subunit may hold on to the mRNA, resume scanning, and reinitiate at a downstream ATG codon. This procedure is called *reinitiation* [26]. To predict TISs, Kozak developed a weight matrix from an extended collection of data [21].

Statistical methods have also been developed to predict TISs. In 1997, Salzberg developed a *positional conditional probability matrix* that takes into account the dependency between adjacent bases [35]. In 1998, Agarwal and Bafna developed the so called *generalized second-order profiles* that consider dependencies between non-adjacent bases [1]. However, both methods suffer from high rates of false positives.

Since 1997, the machine learning approach has been applied to find TISs. With a neural network, Pedersen and Nielsen achieved a 84.6% accuracy on a collection of 3312 vertebrate

[1] In some special cases, the codon GTG is used.

sequences [31]. Salamov *et al.* used six characteristics to analyze the area around a putative start codon and employed linear discriminant analysis for the final scoring [34]. In 2000, Zien *et al.* used support vector machines (SVMs) to predict TISs and achieved an 88.6% accuracy on Pedersen and Nielsen's data [46]. Recently, Hatzigeorgiou achieved a 94% accuracy on 475 cDNA sequences [16]. Her system includes two modules (both based on neural networks), one sensitive to the conserved motif and the other sensitive to the coding/non-coding potential around the start codon. The program linearly searches the coding ORF and stops once the combination of the two modules predicts a positive score. We observe that it is meaningless to compare the performance between Hatzigeorgiou's approach and Pedersen and Nielsen's approach here since they were tested on the different data. Finding TISs was also addressed indirectly in [11] in terms of finding the first exon of a gene contained in a genomic sequence. In [11], Davuluri *et al.* developed the program FirstEF based on a decision tree consisting of quadratic discriminant functions. Besides TISs, FirstEF can also recognize CpG islands, promoter regions and the first splice-donor sites. Using different models to predict CpG-related and non-CpG-related first exons, FirstEF could predict 86% of the first exons.

In this paper, we present a new algorithm to recognize TISs with very high accuracy. Our algorithm contains two major ideas. First, we introduce a class of new sequence-similarity kernels, called the *edit kernels*, for use with support vector machines (SVMs) in a discriminative approach to predict TISs. Our kernels are based on the *string edit distance* and have natural biological and probabilistic interpretations. Second, we treat the upstream and the downstream regions of a putative TIS in different ways. More precisely, we convert the downstream region of a putative TIS into an amino acid sequence before applying SVMs to avoid the high redundancy in the genetic code. The similarity between amino acids is also considered. The algorithm has been implemented with several variants of the edit kernel and tested on Pedersen and Nielsen's data set (as well as some smaller data sets derived from this data). The experiments demonstrate that our algorithm can achieve an accuracy of 99.90% with 99.92% sensitivity and 99.82% specificity, which are significantly better than those of the previous algorithms.

The rest of the paper is organized as follows. Section 2 gives a brief review of SVMs and introduces the basic edit kernel using string edit distance. Section 3 extends the edit kernel by considering the redundancy in the genetic code and similarity between amino acids, and presents two more sophisticated edit kernels. In Section 4, we describe some experimental results on Pedersen and Nielsen's data and small data sets derived from the data. The performance of the edit kernels, the impact of the choice of the edit cost matrix on the performance and efficiency of the SVMs, and some computational issues are discussed in this section. We also give some intuitive explanation on why the SVMs with the above edit kernels work so well. In Section 5, we introduce a publicly accessible online program, called

TISHunter, for predicting TISs based on our algorithm. Section 6 describes the prediction results on mRNAs from the human genome. Section 7 concludes the paper with some directions of further research.

2 Support Vector Machines and an Edit Kernel

Many methods have been proposed for classification problems in bioinformatics. Roughly, these methods follow two approaches, the *generative* approach and the *discriminative* approach. The generative approach, *e.g.* hidden Markov models, builds a model for the target pattern and then evaluates each candidate sequence to see how well it fits the model. If the fitting score is above some threshold, then the candidate is classified into the pattern. The discriminative approach, *e.g.* neural networks, tries to learn some discriminant function from some samples that have been labeled as positive or negative. After learning, the discriminant functions are employed to decide whether a new sample is positive or not. In this paper, we follow the discriminative approach to recognize TISs. In particular, we will employ SVMs with a new class of sequence-similarity kernels, the edit kernels. In what follows, we first give a brief review of some basic concepts in the theory of SVMs. Then we introduce an edit kernel and discuss its biological and statistical meaning.

2.1 Support Vector Machines

Given a set of independent and identically distributed samples (*i.e.* the training data) in the form pairs of patterns x_i and labels y_i,

$$(x_1, y_1), \ldots, (x_\ell, y_\ell) \in \mathcal{X} \times \{\pm 1\}$$

we want to learn a functional dependency $y = f(x; \alpha)$ between x_i and y_i, where α is a parameter from the set Λ. We hope that $f(x; \alpha)$ could make the smallest number of expected errors on the unseen samples drawn from the same distribution.

For a linearly separable data, the SVM is the optimal hyperplane $y = \text{sign}(\langle w, x\rangle + b)$ that maximizes the *margin* $1/\left\|w\right\|^2$ between the classes, which is the minimum distance from positive/negative samples to the separation hyperplane [40,41]. The reason to maximize the margin is that hyperplanes with a larger margin have a smaller capacity (actually a smaller upper bound on the VC-dimension) [40, 41]. In this way, the overfitting problem could be avoided. The optimal hyperplane can be found by maximizing

$$\mathcal{L} = \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{\ell} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle \tag{1}$$

subject to

$$0 \leq \alpha_i \leq C, \quad i = 1, \ldots, \ell \tag{2}$$

$$\sum_{i=1}^{\ell} \alpha_i y_i = 0 \tag{3}$$

where α_i's are the Lagrange multipliers and C is a positive constant. Solving this quadratic programming problem, we get w, b, and thus the optimal hyperplane

$$y = \text{sign}\left(\sum_i \alpha_i y_i \langle x, x_i \rangle - b\right) \tag{4}$$

According to the well-known Karush-Kuhn-Tucker (KKT) necessary conditions [7], the solution w and b to the above quadratic programming problem must satisfy

$$\alpha_i(y_i(\langle w, x_i \rangle + b) - 1) = 0 \tag{5}$$

It means that only the points x_i on the hyperplanes $\langle w, x_i \rangle + b = \pm 1$ have nonzero Lagrange multipliers α_i. Thus, only these points x_i make effects in the optimal hyperplane. Such points are called *support vectors*.

Usually, the input data is not linearly separable. In this case, the input data is first mapped into a high-dimensional feature space $\mathcal{F}$ via a nonlinear function $\Phi(\cdot) : \mathcal{X} \to \mathcal{F}$. SVMs in this high-dimensional space $\mathcal{F}$ can be applied since the data may be linearly separable after the mapping. However, $\mathcal{F}$ could have an arbitrarily large, possibly infinite, dimensionality, which makes it impossible to map points into $\mathcal{F}$ directly. To overcome this obstacle, SVMs can perform the mapping Φ *implicitly*. This is possible because all information that we need supply to an SVM are the dot products $\langle \Phi(x_i), \Phi(x_j) \rangle$ in the feature space $\mathcal{F}$, which can be computed through a *positive definite kernel* $k(\cdot, \cdot)$ in the input data space [2, 4]:

$$k(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle \tag{6}$$

The positive definite kernel (also known as Mercer kernel) is formally defined as follows [5]:

Definition 1 *Let $\mathcal{X}$ be a nonempty set. A function $k(\cdot, \cdot) : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is called a positive definite kernel* [2] *if $k(\cdot, \cdot)$ is symmetric (i.e. $k(x, y) = k(y, x)$ for all $x, y \in \mathcal{X}$) and*

$$\sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j k(x_i, x_j) \geq 0 \tag{7}$$

for all $n \in \mathbb{N}$, $x_1, \ldots, x_n \in \mathcal{X}$ and $c_1, \ldots, c_n \in \mathbb{R}$

[2] The definition can be extended to the more general case of complex-valued kernels. In this paper, we will only need consider real-valued kernels.

For example, polynomial kernel $\langle x, y\rangle^d$ and Gaussian kernel $e^{-\gamma\|x-y\|^2}$ are two well-known positive definite kernels. The matrix $K_{ij} = k(x_i, x_j)$ is called the kernel matrix, which is just the Gram matrix of dot products $K_{ij} = \langle \Phi(x_i), \Phi(x_j)\rangle$ in feature space $\mathcal{F}$.

In practice, the kernel value $k(x, y)$ can also be interpreted as the pairwise similarity between x and y. For example, if x and y are unit-length vectors, the simple dot product $\langle x, y\rangle$ computes the cosine of the angle between x and y. Clearly, two "similar" vectors have a small angle between them and thus a large cosine. Therefore, we can think of the kernel value $k(x, y)$ as a measure of the similarity between x and y in the feature space $\mathcal{F}$. More generally, the similarity represented by $k(x, y)$ could be interpreted in the sense of specific applications and does not necessarily follow the above geometrical interpretation.

2.2 An Edit Kernel

Before introducing our edit kernel, let us review the approach of Zien *et al.* who used SVMs with a polynomial kernel based on Hamming similarity to recognize TISs [46]. Zien *et al.* used a sparse-encoding scheme to represent nucleotides: each nucleotide is encoded by five bits, exactly one of which is set. The position of the set bit indicates whether the nucleotide is A, C, G, T, or N (for unknown). Using this encoding scheme, Zien *et al.* introduced an SVM with the polynomial kernel

$$k(x, y) = \langle x, y\rangle^d$$

Note that, on a sparsely encoded data, the dot product $\langle x, y\rangle$ counts exactly the number of nucleotides that coincide in two sequences represented by x and y, *i.e.* the complement of Hamming distance between x and y. In order to handle local correlations of sequences, Zien *et al.* also introduced two variations of polynomial kernel, called the locality-improved kernel and Salzberg kernel. The locality-improved kernel uses a small sliding window to scan the input sequence and counts matching nucleotides in every window. All these counts are raised to the power of d_1 and then are added up. At last, the sum is taken to the power of d_2. Here, d_1 and d_2 are user-specified parameters. Salzberg kernel is similar to the locality-improved kernel. However, instead of the original nucleotide sequences, Salzberg kernel is applied on the sequences of log odds scores $s_p(x)$, which is defined as

$$s_p(x) = \log \frac{P(x_p \text{ at pos. } p \text{ in } \mathit{TIS} \,|x_{p-1} \text{ at pos. } p-1 \text{ in } \mathit{TIS})}{P(x_p \text{ at pos. } p \text{ in } \mathit{ANY} \,|x_{p-1} \text{ at pos. } p-1 \text{ in } \mathit{ANY})}$$

where x_p is the nucleotide incident at position p in the sequence corresponding to data point x, TIS is the set of training sequences centered around TIS, and ANY is the set of all training sequences.

Clearly, the key in the kernels of Zien *et al.* is the computation of the (complement of) Hamming distance between two sequences though they also consider local positional

dependency. However, Hamming distance is not the best measure of dissimilarity in the comparison of (unaligned) biomolecular sequences. First of all, it requires that the sequences have the same length. Second, in the processes of DNA replication and evolution, the errors like insertions and deletions (*i.e.* indels) of nucleotides are common. In particular, short tandem repeats (STR) are often hotspots for indels [28]. [3] When indels are prevalent, the Hamming distance between two sequences is often an exaggerated over-estimation of the true dissimilarity.

On the other hand, the edit distance (also known as the *Levenshtein metric* [27] and *evolutionary distance* [37]) is a more general and accurate measure of sequence dissimilarities. The (basic, unweighted) edit distance between two sequences denotes the minimum number of edit operations that transform one sequence into the other. Typical edit operations include insertion, deletion and substitution, although other less-frequent operations such as transposition and block moves can also be considered. For simplicity, we only consider the first three operations in this paper. The edit distance between two sequences is a metric and tightly related to the optimal alignment between the two sequences.

In principle, one could perhaps find the TIS of an mRNA sequence by aligning the sequence with some mRNA sequences with known TISs and checking if an ATG codon is the TIS by comparing the edit distance with some threshold. However, such a naive approach (referred to as *template matching*) usually has a large error rate, especially for low-quality sequences (*e.g.* ESTs). Nevertheless, we observe that the above alignment approach uses a weak classifier (template matching) with a good dissimilarity measure (edit distance). On the other hand, Zien *et al.* use a sophisticated classifier (SVM) with a weak similarity measure (the complement of Hamming distance). This observation suggests that we might be able to recognize TISs more accurately by combining the merits of both approaches.

In order to incorporate the edit distance into SVMs, we define the edit kernel to measure the similarity between two sequences:

$$k(x, y) = e^{-\gamma \cdot edit(x,y)} \tag{8}$$

where $edit(x, y)$ is the edit distance between x and y, and γ is a positive real value to scale the kernel value for numerical stability. In fact, γ will play a very important role in making the kernel matrix positive definite, which we will show later. The edit kernel also has a natural probabilistic interpretation. Recall that the edit distance between two (biomolecular) sequences is the minimum number of the edit operations that transform one sequence into the other. Equivalently, we can think of such an edit process as a sequence of (independent) evolutionary events. However, these evolutionary events occur in nature

[3] This is important because the context of a TIS includes the $5'$ UTR, which usually contains STRs, for example CpG islands.

with different probabilities. Let $P(b|a)$ denote the probability of nucleotide (or amino acid, as discussed later) a mutating into nucleotide (or amino acid) b, where a or b (but not both) could be a space (thus denoting an indel). Although identity substitutions of the form $a \to a$ are not used explicitly in string edit, to satisfy $edit(x, x) = 0$, let us assume that $P(a|a) = 1$. This assumption does not generally hold in molecular evolution, But it is usually not a serious problem in practice. Thus, the *cost* of an edit operation $a \to b$ [4] could be interpreted as the negative logarithm of the mutation (*i.e.* substitution or indel) probability $P(b|a)$ and the edit distance between two sequences could be interpreted as the summation [5] of the negative logarithms of the mutation probabilities:

$$edit(x, y) = -\sum_i \log P(y_i|x_i) \tag{9}$$

where $x_i \to y_i$ is the ith mutation in an optimal edit from x into y. We call (9) the *log probability* model. [6] Under this interpretation, the edit kernel defined in (8) is just the probability, raised to the power of γ, of sequence x mutating into y:

$$k(x, y) = \left(\prod_i P(y_i|x_i) \right)^\gamma \tag{10}$$

If two sequences are similar, the mutation probability $\prod_i P(x_i|y_i)$ will be large and thus the kernel value $k(x, y)$ will be large. Otherwise, the probability and the kernel value will be small. Therefore, such an edit kernel measures the similarity between two sequences in the sense of both evolution and probability. Note that, so far we have implicitly assumed that $P(a|b) = P(b|a)$, which may not always be true, especially for amino acids. When this assumption does not hold, we may interpret the edit distance as

$$edit(x, y) = -\frac{1}{2} \left(\sum_i \log P(x_i|y_i) + \sum_i \log P(y_i|x_i) \right) \tag{11}$$

which is the average of the negative log probability of transforming x into y and that of transforming y into x to keep the symmetry property.

To use the edit kernel in support vector machines, we would hope that it is positive definite. Unfortunately, it has been shown that the edit kernel is not positive definite [9, 10].

[4] The cost is restricted to 1 or 0 in this basic string edit model. However, it will be relaxed to arbitrary nonnegative numbers in the next section.

[5] The additive cost scheme corresponds to the assumption that mutations at different sites occur independently. The assumption appears to be a reasonable approximation of the evolution of mRNAs, which are linear and unstructured. The cost scheme can be easily extended to accommodate more general probabilistic models such as affine gap costs.

[6] A similar model is the *log-odds ratio model* [3, 13]. However, we think that the log probability interpretation is more direct and it perhaps deals with indels better.

However, we can still use the edit kernel in support vector machines according to the following theorem [36].

Theorem 2 *Suppose the data $x_1, \ldots, x_\ell$ and the kernel $k(\cdot, \cdot)$ are such that the matrix*

$$K_{ij} = k(x_i, x_j) \tag{12}$$

is positive. Then it is possible to construct a map Φ into a feature space $\mathcal{F}$ such that

$$k(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle \tag{13}$$

Conversely, for a map Φ into some feature space $\mathcal{F}$, the matrix $K_{ij} = \langle \Phi(x_i), \Phi(x_j) \rangle$ is positive.

This theorem implies that, even though the kernel $k(\cdot, \cdot)$ is not positive definite, we can still use $k(\cdot, \cdot)$ in support vector machines or other algorithms that require $k(\cdot, \cdot)$ to correspond to a dot product in some space if the kernel matrix K is positive for the given training data. In fact, Theorem 2 does not even require $x_1, \ldots, x_\ell$ to belong to a vector space. This is very useful for biological sequences analysis because it is hard to come up with a sensible vector representation of biomolecular sequences. Thus, we can use the edit kernel in SVMs if we could make the kernel matrix positive definite. This can be achieved by adjusting the parameter γ. It is well known that a symmetric diagonally dominant real matrix with nonnegative diagonal entries is positive definite. If we choose a large enough γ, we can make the kernel matrix diagonally dominant and thus positive definite. In fact, this requirement could be relaxed in practice. We found in our experiments that a reasonably large γ could already make the kernel matrix positive definite but not diagonally dominant.

Although the edit kernel in (8) has a good biological interpretation, it cannot be applied directly to predict TISs. First, the start codon ATG may be at different positions in different sequences. The above simple edit/alignment approach does not pay attention to the position of a putative start codon and the content of its neighbors (*e.g.* similarity to the consensus sequence GCCACCatgG). Second, the nucleotides in the $5'$ UTR upstream of a start codon and the ones in the downstream region (*i.e.* nucleotides encoding amino acids) usually follow different evolutionary processes because of the selective pressure exerted on protein-coding regions. Therefore, we should compute the edit distance for (putative) $5'$ UTRs and downstream regions separately. In other words, we should employ the edit kernel as follows:

$$k(x, y) = e^{-\gamma_1 \cdot edit(x', y') - \gamma_2 \cdot edit(x'', y'')} \tag{14}$$

where $\gamma_1, \gamma_2 > 0$; $x = x'x'', y = y'y''$ are two (mRNA) sequences (with postulated TISs); x', y' and x'', y'' are the putative $5'$ UTRs and downstream coding regions of x and y, respectively. We call the kernel in (14) *edit kernel I*. We can still easily make the kernel matrix of edit kernel I positive definite by adjusting γ_1 and γ_2 because the product of two positive definite matrices is still positive definite.

3 Redundancy of Codons and More General Edit Costs

We can improve edit kernel I by considering the following simple biological fact. The genetic code is highly *redundant* because there are 61 valid codons (including the ATG start codon) and only 20 amino acids exist [14]. Moreover, the redundancy is not random. In fact, certain features of the redundancy are quite regular. For example, pairs of codons of forms XYC and XYT always code for the same amino acid and pairs of forms XYG and XYA usually code for the same amino acid. Since the region of an mRNA downstream of the TIS codes for amino acids, it makes sense to consider such a downstream region as an amino acid sequence rather than a sequence of nucleotides when computing the edit distance $edit(x'', y'')$. So, we define a new kernel, *edit kernel II*, that has the same form as edit kernel I but with a different domain. The domain of edit kernel I is $\mathcal{X} \times \mathcal{X}$, where $\mathcal{X} = \{\text{all nucleotide sequences}\}$. However, the domain of edit kernel II is $\mathcal{X}' \times \mathcal{X}'$, $\mathcal{X}' = \{\text{all sequences of nucleotides and amino acids}\}$.

3.1 Edit Costs Based on Mutation Probabilities

In an attempt to further improve the above edit kernels, we generalize the cost model of edit operations. Edit kernels I and II use the unit cost model, *i.e.* the cost of editing a into b is 1 if $a \neq b$ or 0 otherwise. The model's predominant virtue is its simplicity. In general, more sophisticated cost models should be used. For example, substitutions between two purines (or pyrimidines), *i.e.* transitions, are more frequent than those between a purine and a pyrimidine, *i.e.* transversions, and thus should cost less according to the connection between edit distance and mutation probabilities. Similarly, replacing an amino acid with a biochemically similar one should cost less than replacements using amino acids with totally different properties. We call the kernel defined on a (weighted) edit distance using some general edit cost matrices for nucleotides and amino acids *edit kernel III*.

A general edit cost matrix can be defined for nucleotides based on some fixed transversion/transition ratio. The most widely used (similarity) score matrices for amino acids are PAM [12] and BLOSUM [17] matrices. PAM matrices are based on the Dayhoff model of evolutionary rates. Using an alternative approach, BLOSUM matrices were derived from about 2000 blocks of aligned sequence segments characterizing more than 500 groups of related proteins. Although PAM and BLOSUM matrices are popular and are good for sequence alignment, neither can be applied directly in the edit kernel because they are generated following the log-odds ratio model rather than the log probability model.

To obtain a cost matrix for edit kernel III, we propose the following algorithm based on the log probability model:

1. *Raise the 1-PAM matrix to the power of p and denote it M;*

2. $M \leftarrow -\log M$;

3. *Calculate the average value* m *of the diagonal of* M;

4. $M \leftarrow M - m$;

5. *Set the diagonal elements of* M *to zero;*

6. *If there are some negative elements in* M*, set them to zero;*

where 1-PAM is a substitution matrix that describes the probability of a substitution in the unit time, *i.e.* the period during which 1% of a nucleotide (or amino acid) sequence is expected to change. The entries of 1-PAM are of the form $P(b|a, t = 1)$ denoting the probability of a mutating into b in the unit time. The algorithm works as follows. By raising 1-PAM to the power of p, *e.g.* $p = 120$ or 250, we obtain the matrix M with entries $P(b|a, t = p)$, which is the probability of a mutating into b in p time units. Then we take logarithm on M and change the sign to obtain a cost matrix that is suitable for sequences with an average divergence of p time units. In order to make it satisfy the condition $edit(x, x) = 0$, we subtract the average m of the diagonal from M. Such a shift makes the average of the diagonal of the cost matrix zero. Now we can set the diagonal to zero. [7] Usually, there may be a very small number of off-diagonal entries slightly less than zero after the shift. We also set them to zero to make the matrix nonnegative. We call the cost matrix calculated by this algorithm a *Substitution Cost Matrix* (SCM). Note that, an SCM may be asymmetric. When using such an asymmetric cost matrix, we need modify the edit distance definition by taking the average of $edit(x, y)$ and $edit(y, x)$ in the kernel as discussed in the previous section.

To obtain the cost matrices for nucleotides, we use the 1-PAM matrix from [30], which is based on Kimura's two-parameter (K2P) model of nucleotide substitutions [20]. Specifically, the probability of a transition for each nucleotide is 0.006 and that of a transversion is 0.002. The cost matrix with $p = 250$ is listed in Table 1. Note that, the matrix is symmetric.

Since the 1-PAM matrix for amino acids is not symmetric [12], a resulting SCM for amino acids may not be symmetric. We may use such a matrix directly and modify the edit distance definition in the kernel to make it symmetric as mentioned above. Alternatively, we may make the resulting cost matrix symmetric by computing $M \leftarrow (M + M^T)/2$ either (i) immediately before step 6 or (ii) immediately before step 2. The former option seems more logical, but it might result in an edit distance similar to the above modified edit

[7] One may attempt to set the diagonal entries $-\log P(a|a, t = p)$ to zero directly without the shift. However, $P(a|a, t = p)$ is usually small when p is large and thus $-\log P(a|a, t = p)$ is not close to zero, So, setting the diagonal to zero directly might introduce significant unfair bias against non-identity substitutions and indels.

Table 1: The SCM250 cost matrix for nucleotides.

	A	C	G	T
A	0.0000	0.3009	0.0626	0.3009
C	0.3009	0.0000	0.3009	0.0626
G	0.0626	0.3009	0.0000	0.3009
T	0.3009	0.0626	0.3009	0.0000

distance. Hence, we will only consider the latter option as an alternative, and call the cost matrix resulted from this (alternative) approach an *Approximate Substitution Cost Matrix* (ASCM). An SCM cost matrix for amino acids with $p = 250$ is listed in Table 2.

Finally, we need define costs for indels. Since there are few rigorous treatments of indels, we define indel costs based on empirical experience. By some preliminary experiments, we have found that SVMs usually perform better if we set the indel cost for $5'$ UTRs small and that for the downstream region relatively large (the actual values will be given in the next section). This is consistent with the fact that the indels are expected to be more frequent in $5'$ UTRs than in downstream regions that code amino acids.

4 Experiments

To evaluate the performance of our algorithm for predicting TISs, we test all three edit kernels on Pedersen and Nielsen's original data set [31] and some small data sets derived from the data. The experimental results, as described below, show that our methods perform significantly better than the Salzberg method, neural networks, and the SVM with Salzberg kernel. [8] In particular, the sensitivity and specificity of our methods are much higher than those of the previous methods. Just like accuracy, high sensitivity and specificity are both key desirable properties in a practical prediction application. In the following, we will describe Pedersen and Nielsen's data set, our experimental results, and some efficiency issues in the implementation of our algorithm.

4.1 Data

The test data consists of a collection of 3312 sequences from vertebrates, which were originally extracted from GenBank by Pedersen and Nielsen [31]. To mimic mRNAs, all sequences

[8] Unfortunately, we were not able to obtain the program and data of Hatzigeorgiou and compare our method with her method in [16]. The method of Davuluri *et al.* [11] requires the presence of the first introns and thus cannot be applied to Pedersen and Nielsen's data.

Table 2: The SCM250 cost matrix for amino acids.

	Ala	Arg	Asn	Asp	Cys	Gln	Glu	Gly	His	Ile
Ala	0.0000	1.0288	0.6270	0.6013	1.1485	0.7606	0.5934	0.3688	0.9903	0.7640
Arg	1.8122	0.0000	1.4515	1.7407	2.2667	1.1464	1.6854	2.1189	1.0766	1.8287
Asn	1.3744	1.4431	0.0000	0.9558	2.2558	1.2619	1.1007	1.3277	1.0902	1.8317
Asp	1.1885	1.5994	0.8086	0.0000	2.4359	0.9167	0.5009	1.1236	1.1471	1.8011
Cys	2.1520	2.3747	2.4974	2.8536	0.0000	2.8506	2.8806	2.6636	2.4044	2.2314
Gln	1.6104	1.2289	1.3249	1.1284	2.7457	0.0000	0.9326	1.8024	0.8330	1.9517
Glu	1.1337	1.4899	0.9001	0.4475	2.4284	0.6629	0.0000	1.1732	1.0959	1.6573
Gly	0.3518	1.2276	0.5681	0.5141	1.4189	0.9065	0.6062	0.0000	1.1413	1.2020
His	1.9552	1.2860	1.2679	1.4856	2.4181	0.9739	1.5091	2.1935	0.0000	2.2691
Ile	1.6842	2.0071	1.9466	2.0938	1.9942	2.0105	2.0369	2.1497	2.1313	0.0000
Leu	1.1537	1.3987	1.3301	1.6096	2.0744	1.0983	1.4796	1.5850	1.1605	0.1232
Lys	1.0046	0.0000	0.5183	0.7172	1.9786	0.5806	0.7472	1.1485	0.7751	1.1524
Met	2.7194	2.5547	2.8212	3.0657	3.6201	2.6505	2.9833	3.1669	2.9342	1.9184
Phe	2.1427	2.3371	2.1122	2.6012	2.3679	2.4438	2.5744	2.3206	1.8045	1.2066
Pro	0.9213	1.2422	1.2908	1.3707	1.8339	1.1129	1.2793	1.2794	1.2333	1.6099
Ser	0.6327	0.9573	0.7302	0.8164	0.9170	0.9984	0.8803	0.6450	1.0993	1.2013
Thr	0.7754	1.2408	0.9567	1.0868	1.5666	1.2364	1.1424	1.0474	1.3877	1.0372
Trp	4.1465	2.3507	4.0147	4.5089	4.4423	4.1136	4.5235	4.4084	3.9310	4.2786
Tyr	2.4368	2.8054	2.1614	2.6514	1.6218	2.6273	2.6000	2.8303	1.7231	1.9500
Val	0.9590	1.5266	1.3838	1.4980	1.4263	1.4188	1.4346	1.3315	1.5029	0.1265

	Leu	Lys	Met	Phe	Pro	Ser	Thr	Trp	Tyr	Val
Ala	1.0608	0.9189	0.9299	1.4089	0.3837	0.4102	0.3822	2.0051	1.5063	0.6157
Arg	2.0535	0.6364	1.4917	2.2782	1.4798	1.5275	1.6300	0.9498	2.4702	1.9885
Asn	2.1256	1.1884	1.8476	2.1482	1.4958	1.2568	1.3134	2.3480	1.9161	1.8199
Asp	2.1670	1.2707	1.9026	2.4924	1.4231	1.1918	1.2914	2.8151	2.3072	1.7417
Cys	3.0225	2.8388	2.8337	2.6512	2.1350	1.6894	2.0654	3.4106	1.6243	2.0090
Gln	1.8786	1.3129	1.7412	2.4912	1.4180	1.6237	1.6749	2.6009	2.4778	1.9543
Glu	1.9149	1.2394	1.7237	2.3930	1.2870	1.2097	1.2900	2.8204	2.2546	1.6099
Gly	1.5027	1.0430	1.3131	1.7617	0.7339	0.3931	0.6308	2.2513	1.9164	0.9339
His	2.2386	1.6246	2.1971	1.9660	1.6870	1.8340	1.9248	2.2129	1.6833	2.2617
Ile	1.0006	1.9119	1.0465	1.3243	2.0545	1.8714	1.5128	2.7572	1.8350	0.6836
Leu	0.0000	1.2478	0.0000	0.2937	1.2701	1.3498	1.0543	1.1239	0.9640	0.2597
Lys	1.4263	0.0000	0.6409	1.8984	0.9669	0.7738	0.7315	1.5634	1.8331	1.2834
Met	1.5751	2.2624	0.0000	2.4703	2.9574	2.7928	2.5171	3.4405	3.1083	2.0059
Phe	1.0519	2.5536	1.3868	0.0000	2.5790	2.0818	2.0377	1.3594	0.0000	1.7456
Pro	1.6878	1.4042	1.6342	2.0554	0.0000	0.9767	1.1066	2.4848	2.3167	1.4815
Ser	1.5323	0.9043	1.2614	1.6077	0.6480	0.0000	0.5726	1.4826	1.6151	1.1126
Thr	1.4431	1.0372	1.2016	1.7432	0.9501	0.7444	0.0000	2.2463	1.7763	0.9951
Trp	4.4659	3.6745	4.2634	2.5395	4.1474	3.2582	3.9810	0.0000	2.5088	4.6172
Tyr	1.9193	2.9312	2.2830	0.1207	2.8663	2.3555	2.3499	1.7131	0.0000	2.2774
Val	0.5623	1.4650	0.5669	1.2317	1.2693	1.2151	0.9088	2.4074	1.5771	0.0000

Table 3: Comparison of six-fold cross validation classification accuracies among different TISs prediction methods. The neural network results were obtained by Pedersen and Nielsen [31]. The results of the Salzberg method and the SVM with Salzberg kernel were obtained by Zien *et al.* [46]. The parameters were chosen by the five-fold cross validations on a small data set.

Method	Parameters	Accuracy	Specificity	Sensitivity	Correlation
Neural Network		84.6%	64.5%	82.4%	62.7%
Salzberg method		86.2%	73.7%	68.1%	61.9%
SVM Salzberg kernel	$d_1 = 3$ $l = 1$	88.6%	76.0%	78.4%	69.6%
SVM edit kernel I	$C = 4$ $\gamma_1 = 0.00195$ $\gamma_2 = 0.00391$	93.2%	94.5%	89.3%	82.2%
SVM edit kernel II	$C = 16$ $\gamma_1 = 0.00017$ $\gamma_2 = 0.00049$	96.5%	96.0%	98.0%	91.0%

were "spliced" by removing possible introns and joining the remaining exon parts. Besides, only sequences containing at least 10 nucleotides upstream and at least 150 nucleotides downstream of their respective start codons were selected. A very thorough reduction of redundancy was performed to avoid over-estimating the prediction accuracy.

Because all the sequences contain TISs (*i.e.* they are all positive samples), we generate our test samples (both positive and negative) as follows [31, 46]. For every potential start codon ATG in some sequence, a new sequence of length at most 210 nucleotides is extracted. Each of these sequences contains at most 30 nucleotides upstream and at most 180 nucleotides downstream (relative to the A in the putative start codon ATG). This leads to 13503 training and testing sequences, of which 3312 are positive and the rest are negative samples.

Note that in the experiments of Pedersen and Nielsen [31] and Zien *et al.* [46], they used a 200 nucleotide window approach to generate samples. Each window contains a potential TIS at the center of the window. For an ATG codon near the (left or right) end of an mRNA sequence, each missing position in the test sequence (window) is filled with an 'N' (unknown). However, here we do not need fill in 'N's for missing positions because our kernels are based on string edit, which does not require the sequences have the same length.

4.2 Experimental Results

The experimental results are shown in Tables 3 and 4. All the results are based on the six-fold cross validation method, as done in [31] and [46]. Namely, the data is divided into

Table 4: Comparison of six-fold cross validation classification accuracies of the SVM with edit kernel III using different cost matrices. The parameters were chosen by five-fold cross validations on a small data set. In the indel penalty column, the first number is the penalty for nucleotides and the second is the penalty for amino acids.

Cost Matrix	Parameters	Indel Penalty	Accuracy	Specificity	Sensitivity	Correlation
SCM120	$C = 32$	1.00	99.64%	99.73%	99.37%	99.02%
	$\gamma_1 = 0.00195$	9.00				
	$\gamma_2 = 0.00195$					
SCM250	$C = 16$	0.35	99.84%	99.88%	99.73%	99.58%
	$\gamma_1 = 0.00195$	8.00				
	$\gamma_2 = 0.00781$					
ASCM120	$C = 8$	1.00	99.67%	99.69%	99.61%	99.10%
	$\gamma_1 = 0.00195$	9.00				
	$\gamma_2 = 0.00781$					
ASCM250	$C = 8$	0.35	99.90%	99.92%	99.82%	99.72%
	$\gamma_1 = 0.00195$	7.00				
	$\gamma_2 = 0.00781$					
PAM250	$C = 8$	0.35	97.75%	98.17%	96.44%	93.97%
	$\gamma_1 = 0.00195$	13.9				
	$\gamma_2 = 0.00195$					

six parts of approximately equal sizes and each part is in turn reserved for testing the SVM learned on the other five parts. The average of the six prediction results are reported. The parameters C, γ_1, γ_2 are chosen optimally by cross validation experiments on a small data set of size 500. In the tables, accuracy is defined as $(TP+TN)/N$, specificity is $TN/(TN+FP)$, [9] sensitivity is $TP/(TP + FN)$, and correlation is Mathews correlation coefficient defined as:

$$Corr. = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

where N, TP, TN, FP, and FN denote the numbers of the test samples, true positives, true negatives, false positives, and false negatives, respectively.

Using the SVM with edit kernel I, our algorithm performed reasonably well on the data. The accuracy (93.2%) is much better than that (88.6%) of the SVM with Salzberg kernel. In particular, the specificity (94.5%) and the sensitivity (89.3%) are significantly better than those (76.0% and 78.4%, respectively) of the SVM with Salzberg kernel. With edit kernel II, the accuracy is improved to 96.5%. The specificity is slightly improved to 96.0%, while the sensitivity is significantly improved to 98.0%.

[9] Zien *et al.* used a different definition, $TP/(TP + FP)$, which is usually called *precision* [15].

Table 5: Comparison of the accuracies among the edit kernels on small data sets of size 500, 1000, and 2000. Edit kernel III employs the SCM250 cost matrix. The accuracy is estimated by six-fold cross validations.

	500	1000	2000
Edit kernel I	86.0%	86.4%	89.5%
Edit kernel II	92.0%	92.9%	95.1%
Edit kernel III	98.6%	99.3%	99.4%

To evaluate edit kernel III, we used the cost matrices SCM120, ASCM120, SCM250, and ASCM250, *i.e.* the SCM and ASCM with $p = 120$ and $p = 250$, respectively. Considering the popularity of PAM250, we also tested edit kernel III with PAM250. Of course, the PAM250 matrix cannot be used directly in the edit kernel because the edit distance based on PAM250 does not meet the requirements of metric. Thus, we modify PAM250 as follows. We first convert PAM250 into a cost matrix. Then we shift PAM250 up by subtracting the average of the diagonal from every element in the matrix. Finally, we set the diagonal elements and the two off-diagonal negative elements to zero. The comparative results of edit kernel III with these different cost matrices are shown in Table 4, which are in general much better than those in Table 3. Again, the parameters and gap penalties were chosen by experiments on a small data set. The SVMs with matrices SCM120, SCM250, ASCM120, and ASCM250 all performed equally well, with only some minor differences. The accuracy is amazingly improved to 99.9% with the cost matrix ASCM250. On the other hand, the accuracy of the SVM with PAM250 is 97.7%, which is only slightly better than that of the SVM with edit kernel II employing the unit cost. This could perhaps be an evidence suggesting that the log probability model is more suitable for the edit distance in edit kernel III than the log-odds ratio model. On the other hand, the results in Table 4 also indicate that the SVM with edit kernel III is not very sensitive to the actual cost matrix used as long as it is based on the log probability model (although it might require different parameters for different matrices).

In fact, the SVM with edit kernel III is better than it appears in Table 4, which is based on cross validations on a large data set with 13503 samples. On small data sets, the SVM with edit kernel III can still achieve a very high accuracy, as shown in Table 5. In contrast, the accuracies of the SVMs with edit kernels I and II drop notably on the small data sets. For example, on a set of 500 samples, the SVM with edit kernel III based on SCM250 can achieve an accuracy of 98.6%, while the SVMs with edit kernels I and II could only achieve 86.0% and 92.0%, respectively.

Besides accuracy, it is also important to compare the SVMs with different edit kernels in

Table 6: Comparison of the average numbers of support vectors among the edit kernels.

Kernel	Average # of SVs
Edit kernel I	2312
Edit kernel II	2316
Edit kernel III, SCM120	319
Edit kernel III, SCM250	230
Edit kernel III, ASCM120	507
Edit kernel III, ASCM250	293
Edit kernel III, PAM250	821

terms of the number of support vectors. It is known that the set of support vectors provides a "compressed" version of the training data containing all the information necessary to solve a given classification task [36]. The average number of support vectors of the SVMs with edit kernels I, II and III on the large set of 13503 samples are listed in Table 6. These numbers are again based on six-fold cross validations. The SVMs with edit kernel I and II have about 2300 support vectors, which is pretty small compared with the number of training samples. Moreover, the SVM with edit kernel III based on SCM250 uses only 230 support vectors. In contrast, the SVM based on SCM120 has 319 support vectors. This indicates that SCM250 (*i.e.* the substitution cost matrices over 250 evolutionary time units) is perhaps more suitable than SCM120 for Pedersen and Nielsen's data set of vertebrate mRNAs. Besides, when ASCM250 and ASCM120 are used, the SVMs require 293 and 507 support vectors, respectively. A reason that slightly more support vectors are required here is probably because some information is lost when we take the average on the probabilities in an ASCM. The fact that the SVM with PAM250 requires 821 support vectors suggests again that the log-odds ratio model may not be as suitable as the log probability model for the edit kernel.

A reason that the edit kernel III uses a small number of support vectors may be that evolutionary information is already incorporated into the kernel so that the SVM is able to do prediction with fewer support vectors. It is also known that the expectation of the number of learned support vectors from a training set of size ℓ, divided by $\ell - 1$, is an upper bound on the expected probability of test error [42]. Thus, the very small number of support vectors required by edit kernel III with SCM250 is an assurance of the SVM's good performance.

Table 7: Comparison of the average numbers of iterations among the edit kernels. Edit kernel III employs the SCM250 cost matrix.

	Average # of iterations
Edit kernel I	105065
Edit kernel II	20208
Edit kernel III	4466

4.3 Computational Issues

The time of training an SVM depends on both the number of iterations and the time complexity of the kernel function. If the *sequential minimal optimization* (SMO) is employed to train the SVM, the number of iterations ranges somewhere between linear and quadratic in the training set size [32], depending on the actual data and kernel function. Table 7 lists the average numbers of the iterations used in training the SVMs with different edit kernels in the above six-fold cross validation experiment on the large data set. For all three edit kernels, the numbers of iterations are (roughly) proportional to the size of training data, although the SVM with edit kernel I seems to require significantly more iterations.

In the prediction/testing phase, the speed of an SVM depends on both the number of support vectors and the time complexity of the kernel function. According to the indicator function (4), the fewer support vectors, the faster the SVM predicts. Since the SVM with edit kernel III requires the fewest support vectors as shown in Table 6, they are the fastest in the prediction phase as well.

In both training and testing phases, the time complexity of the kernel function plays an important role in the speed of an SVM. Many kernel functions, *e.g.* the polynomial kernels, can be computed in $O(n)$ time, where n is the length of input vectors (or sequences). However, our edit kernels have time complexity $O(n^2)$ based on dynamic programming. To improve the time complexity, one may attempt to use some fast algorithm to compute the edit distance, such as Ukkonen's algorithm [39]. Ukkonen's algorithm runs in $O(s * n)$ time for instances of length n and edit distance s. Ukkonen's algorithm, however, works only for the unit cost model. Although Ukkonen's algorithm has been improved by Wu *et al.* [43, 44] and Berghel and Roach [6], the improved algorithms still depends on the unit cost model. Therefore, Ukkonen's algorithm and its improvements are not suitable for edit kernel III. Finding a fast (approximate) algorithm for edit kernel III is an interesting future research topic. For example, we may limit the number of indels, say k, so that we need only compute the k-diagonal elements in the table of dynamic programming. Such a band-based algorithm is particularly suitable for computing edit distances on coding regions since indels are usually

not frequent in these regions.

4.4 Discussion

The above experimental results show that the SVMs with the edit kernels have a very high accuracy in predicting TISs that is unmatched by any previous technique. Why do these algorithms work so well? Below, we try to give some insights from several perspectives.

First of all, our algorithms are based on SVMs. SVMs have a solid foundation in statistical learning theory and have proven to be more general and powerful than many other learning techniques in applications. More precisely, the capacity control capability makes SVMs free of the overfitting problem [40, 41]. SVMs can also been interpreted in the framework of regularization theory [38], which is a general approach to handle ill-posed problems. The small number of support vectors used in an SVM also has natural interpretations in the context of algorithmic complexity and *minimum description length* (MDL) principle [40].

Although SVMs are general and powerful, they are not a silver bullet. The performance of an SVM largely depends on the choice of its kernel. In general, the more prior knowledge is incorporated into the kernel function, the better the performance of the SVM. This is why the SVM with Salzberg kernel that incorporates local dependency in the sequences performed better than the SVM with a plain polynomial kernel based on Hamming distance [46]. Our edit kernel I is based on the edit distance, which is more general than the Hamming distance and measures the dissimilarity between sequences more accurately. With such a simple improvement on the kernel, our SVM achieved a notably higher accuracy than the SVM with Salzberg kernel. Therefore, it is natural to expect that the SVMs with edit kernels II and III, which take into account more prior knowledge including codon redundancy and evolutionary costs between nucleotides and amino acids, would perform even better than the SVM with edit kernel I.

5 An Online Program: TISHunter

Based on the SVM with edit kernel III, we have developed an online program *TISHunter* [10] for the prediction of TISs in mRNA sequences. TISHunter uses SCM250 because it has resulted in the high accuracy, the smallest number of support vectors, and the fastest training time in the tests done so far. TISHunter works on a "per sequence" basis, *i.e.* it linearly scans each input mRNA sequence and predicts a score for every potential start codon ATG. The ATG codon with the largest predicted value is output as the putative TIS in the sequence if

[10] The server is publicly accessible at http://bioinfo.ucr.edu/∼hli.

Table 8: Six-fold evaluation of TISHunter with the SCM250 cost matrix on Pedersen and Nielsen's data set of 3312 sequences. The results of ESTScan, the Salzberg method and the SVM with Salzberg kernel were obtained by Zien *et al.* [46].

Method	Sample #	Error #
ESTScan, closest ATG	2350	729
Salzberg method	3312	1095
SVM, Salzberg kernel	3312	530
TISHunter	3312	13

the score is positive. [11]

To evaluate TISHunter, we again divide Pedersen and Nielsen's original set of 3312 sequences into six parts, each of which has 552 sequences. For each part, we train TISHunter on the other five parts and test it on the missing part. The test results and comparisons with those of the previous methods are summarized in Table 8. Note that such a comparison is not fair for ESTScan since it was not trained on these sequences. Also, Zien *et al.* took advantage of the fact that input every input mRNA sequence has a TIS, and simply chose the ATG codon with the largest predicted value as TIS even if the value is negative.

It is amazing that TISHunter made only 13 incorrect predictions totally, while the other programs made at least 40 times more errors. It turns out that these 13 errors are all false negatives (*i.e.* no TISs were predicted in the involved sequences) because none of the candidate start codons in the sequences received a positive prediction score. If we take advantage of the fact that these mRNA sequences all have TISs, we could modify TISHunter to simply output the ATG codon with the largest prediction score as the putative TIS in each sequence. Interestingly, the modified program would make no errors on this data set. The reason that the modified program could perform perfectly here is that, when working on a "per ATG codon" basis, our algorithm usually gives a true start codon a larger score than those of the true negatives in the same mRNA sequence. Thus, even if a true start codon is incorrectly predicted as a negative, its prediction score could still be larger than those of the true negatives in the same sequence and will thus be output as a putative TIS by the modified program.

[11] Note that, TISHunter does not assume that the input sequence must have a TIS.

6 Prediction on the Human mRNAs

We downloaded all human mRNAs with the status code REVIEWED from NCBI Reference Sequence (RefSeq) database [33]. These sequences have been reviewed by NCBI staffs or their collaborators and we may assume that they are of high quality. The dataset contains 8824 sequences. After deleting the sequences whose upstream region to the left of TIS is less than 10 nucleotides or downstream region is less than 150 nucleotides, we kept 8225 sequences as the experimental dataset. These 8225 sequences have an average length of 2844 and contain 417880 potential ATG start codons. Of the 8225 sequences, there are only 4400 (53.5%) sequences that use the first ATG as TIS. It is very different from Kozak's claim that downstream ATGs are used as start codons in less than 10% of her investigated eukaryotic mRNAs [24].

Because the signal-to-noise ratio (8225/409655 = 2%) is very low, we use a different experimental setting from what we used on Pedersen and Nielsen's data. For each ATG before (and including) the true TIS, we generate a data point that contains at most 30 nucleotides upstream and 270 nucleotides downstream. This length is determined based on our experiments on small size data. Note that the downstream region is longer than that used on Pedersen and Nielsen's eukaryotic mRNAs. With this setting, we generate 20877 data points, of which 8825 are positive. Based on a three-fold cross validation, the support vector machine achieves 96.7% accuracy with edit kernel III and SCM250 cost matrix. The specificity, sensitivity, and correlation are 95.7%, 98.0%, and 93.1%, respectively. The average number of support vectors are 4961, which are 23.8% of the training data. In the experiment, the parameter C, γ_1, and γ_2 are set to 32, 0.0625, and 0.015625, respectively.

We also run TISHunter on the human mRNAs. TISHunter works on a per sequence basis and linearly scans the input mRNA sequence. The first ATG that gets a positive score is reported as the putative TIS. We use this setting because the training data contains only the ATGs before (and including) TISs. TISHunter can predict 92.9% of the 8225 true TISs correctly. Among the erroneously predicted ATGs, we find the following results. In the 584 incorrect predictions, 486 (83.2%) are in the upstream of the true TISs, 98 (16.8%) are in the downstream, 365 (62.5%) are in the reading frames, and 219 (37.5%) are out of the reading frames.

Interestingly, we find that, out of the 486 false predictions in the upstream of TISs, 173 (35.6%) contain a stop codon between the wrongly predicted position and the true start codon. We think that many of these false TISs may contribute to the procedure reinitiation. As we mentioned before, reinitiation happens when an 80S ribosome translates the first small open reading frame (upORF) and reaches a stop codon. In the study of human immunodeficiency virus type 1 mRNAs, Luukkonen *et al.* found that downstream translation initiation is inhibited in 50% of the cases by an upORF of 84 nucleotides and should be

entirely abrogated by an ORF longer than 165 nucleotides (predicted by extrapolation) [29]. In our 173 cases, 51 (29.5%) and 74 (42.8%) of the false TISs have stop codons in the 84 and 165 nucleotide downstream regions, respectively. Besides, reinitiation in eukaryotes is most efficient when the upORF terminates at some distance before the start of the next cistron [23]. The reason is that the 40S ribosomal subunit requires time (distance) to re-acquire Met-tRNA$_i$·eIF-2, without which the downstream ATG codon cannot be recognized [18]. According to the study of Luukkonen *et al.*, an intercistronic distance shorter than 37 nucleotides appears to negatively affect initiation frequency at downstream ATGs [29]. In the 74 false TISs that have stop codons in the 165 nucleotides downstream region, 53 (71.6%) have the intercistronic distance larger than 37 nucleotides. Further analysis on these 53 false TISs is in progress.

7 Conclusion

Automating the process of sequence annotations is an important part of the post-sequencing genomics research. In this paper, we show that a powerful machine learning technique, SVMs, can effectively find TISs if we carefully incorporate the biological knowledge into the kernel function. The method proposed in this paper can be extended to the recognition of other biological signals, *e.g.* binding sites of regulatory proteins, because one can incorporate motif information into edit kernels. We hope that further analysis on the learned support vectors will elucidate the real mechanisms involved in a translation initiation process (*e.g.* how much does the ribosome care about a downstream region).

8 Acknowledgement

We would like to thank A.G. Pedersen and H. Nielsen very much for sharing their data, A. Zien for helpful information, D. Gunopulo for some valuable comments, and N. Cristianini, J.-P. Vert, K. Tsuda and G. Lanckriet for pointing out an error in the early version of this paper. This work was partially supported by NSF grants CCR-9988353, ACI-0085910, and CCR-0309902, and National Key Project for Basic Research (973) grant 2002CB512801.

References

[1] P. K. Agarwal and V. Bafna. Detecting non-adjoining correlations with signals in DNA. In *Proceedings of the 2nd Annual International Conference on Research in Computational Molecular Biology*, pages 1–7, 1998.

[2] M. Aizerman, E.Braverman, and L.Rozonoer. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25:821–837, 1964.

[3] S. F. Altschul. Amino acid substitution matrices from an information theoretic perspective. *Journal of Molecular Biology*, 219(3):555–565, 1991.

[4] N. Aronszajn. Theory of reproducing kernels. *Transactions of the American Mathematical Society*, 68(3):337–404, 1950.

[5] C. Berg, J. P. R. Christensen, and P. Ressel. *Harmonic Analysis on Semigroups*. Springer-Verlag, 1984.

[6] H. Berghel and D. Roach. An extension of Ukkonen's enhanced dynamic programming ASM algorithm. *ACM Transactions on Information Systems*, 14(1):94–106, 1996.

[7] D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, Belmont, MA, 2nd edition, 1999.

[8] C. Burge and S. Karlin. Prediction of complete gene structures in human genomic DNA. *Journal of Molecular Biology*, 268(1):78–94, 1997.

[9] C. Cortes, P. Haffner, and M. Mohri. Rational kernels. In *Advances in Neural Information Processing Systems 15*, pages 41–56, 2002.

[10] C. Cortes, P. Haffner, and M. Mohri. Positive definite rational kernels. In *Proceedings of 16th Annual Conference on Computational Learning Theory*, pages 41–56, 2003.

[11] R. Davuluri, I. Grosse, and M. Q. Zhang. Computational identification of promoters and first exons in the human genome. *Nature Genetics*, 29(4):412–417, 2001.

[12] M. O. Dayhoff, R. M. Schwartz, and B. C. Orcutt. A model of evolutionary change in proteins. In M. O. Dayhoff, editor, *Atlas of Protein Sequence and Structure*, volume 5, pages 345–352, 1978.

[13] R. Durbin, S. R. Eddy, A. Krogh, and G. Mitchison. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, 1998.

[14] D. Freifelder. *Molecular Biology*. Jones and Bartlett, Boston, MA, 2nd edition, 1987.

[15] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers, San Francisco, 2000.

[16] A. G. Hatzigeorgiou. Translation initiation start prediction in human cDNAs with high accuracy. *Bioinformatics*, 18(2):343–350, 2002.

[17] S. Henikoff and J. G. Henikoff. Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Sciences of USA*, 89(22):10915–10919, 1992.

[18] A. G. Hinnebusch. Translational regulation of yeast GCN4. *Journal of Biological Chemistry*, 272(35):21661–21664, 1997.

[19] C. Iseli, C. V. Jongeneel, and P. Bucher. ESTScan: A program for detecting, evaluating, and reconstructing potential coding regions in EST sequences. In *Proceedings of the 7th International Conference on Intelligent Systems for Molecular Biology*, pages 138–148, Menlo Park, CA, 1999. AAAI Press.

[20] M. Kimura. A simple method for estimating evolutionary rates of base substitutions through comparative studies of nucleotide sequences. *Journal of Molecular Evolution*, 16(2):111–120, 1980.

[21] M. Kozak. An analysis of 5′-noncoding sequences from 699 vertebrate messenger RNAs. *Nucleic Acids Research*, 15(20):8125–8148, 1987.

[22] M. Kozak. At least six nucleotides preceding the AUG initiator codon enhance translation in mammalian cells. *Journal of Molecular Biology*, 196(4):947–950, 1987.

[23] M. Kozak. Effects of intercistronic length on the efficiency of reinitiation by eucaryotic ribosomes. *Molecular and Cellular Biology*, 7(10):3438–3445, 1987.

[24] M. Kozak. The scanning model for translation: An update. *Journal of Cell Biology*, 108:229–241, 1989.

[25] M. Kozak. Interpreting cDNA sequences: Some insights from studies on translation. *Mamalian Genome*, 7(8):563–574, 1996.

[26] M. Kozak. Initiation of translation in prokaryotes and eukaryotes. *Gene*, 234(2):187–208, 1999.

[27] V. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Daklady*, 10:707–710, 1966.

[28] W.-H. Li and D. Graur. *Fundamentals of Molecular Evolution*. Sinauer Associates, Sunderland, MA, 1991.

[29] B. G. M. Luukkonen, W. Tan, and S. Schwartz. Efficiency of reinitiation of translation on human immunodeficiency virus type 1 mRNAs is determined by the length of the upstream open reading frame and by intercistronic distance. *Journal of Virology*, 69(7):4086–4094, 1995.

[30] D. W. Mount. *Bioinformatics: Sequence and Genome Analysis*. Cold Spring Harbor Laboratory Press, Cold Spring Harbor, N.Y., 2001.

[31] A. G. Pedersen and H. Nielsen. Neural network prediction of translation initiation sites in eukaryotes: Perspectives for EST and genome analysis. In *Proceedings of the 5th International Conference on Intelligent Systems for Molecular Biology*, pages 226–233, 1997.

[32] J. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods — Support Vector Learning*, pages 185–208, Cambridge, MA, 1999. MIT Press.

[33] K. D. Pruitt and D. R. Maglott. Refseq and locuslink: Ncbi gene-centered resources. *Nucleic Acids Research*, 29(1):137–140, 2001.

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

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

[36] B. Schölkopf. *Support Vector Learning*. R. Oldenbourg Verlag, Munich, 1997.

[37] P. H. Sellers. On the theory and computation of evolutionary distances. *SIAM Journal on Applied Mathematics*, 26(4):787–793, 1974.

[38] A. N. Tikhonov. Solution of incorrectly formulated problems and the regularization method. *Soviet Mathematics Doklady*, 4:1035–1038, 1963.

[39] E. Ukkonen. Algorithms for approximate string matching. *Information and Control*, 64:100–118, 1985.

[40] V. N. Vapnik. *The Nature of Statistical Learning Thoery*. Springer-Verlag, New York, 1995.

[41] V. N. Vapnik. *Statistical Learning Theory*. John Wiley & Sons, New York, 1998.

[42] V. N. Vapnik and A. Chervonenkis. *Theory of Pattern Recognition*. Nauka, Moscow, 1974.

[43] S. Wu and U. Manber. Fast text searching allowing errors. *CACM*, 35(10):83–91, 1992.

[44] S. Wu, U. Manber, and G. Myers. An O(NP) sequence comparison algorithm. *Information Processing Letters*, 35:317–323, 1990.

[45] H. Yoon and T. F. Donahue. Control of translation initiation in *saccharomyces cerevisiae*. *Molecular Microbiology*, 6(11):1413–1419, 1992.

[46] A. Zien, G. Rätsch, S. Mika, B. Schölkopf, T. Lengauer, and K.-R. Müller. Engineering support vector machine kernels that recognize translation initiation sites. *Bioinformatics*, 16(9):799–807, 2000.