ELSEVIER

Expert Systems with Applications 30 (2006) 73–81

Expert Systems with Applications

www.elsevier.com/locate/eswa

Splice site prediction using support vector machines with a Bayes kernel

Ya Zhang [a], Chao-Hsien Chu [a,*], Yixin Chen [b], Hongyuan Zha [c], Xiang Ji [d]

[a] *School of Information Sciences and Technology, 301 K IST Building, Pennsylvania State University, University Park, PA 16802, USA*
[b] *Department of Computer Science, University of New Orleans, New Orleans, LA 70148, USA*
[c] *Department of Computer Science and Engineering, Pennsylvania State University, University Park, PA 16802, USA*
[d] *NEC Laboratories America, Cupertino, CA 95014, USA*

Abstract

One of the most important tasks in correctly annotating genes in higher organisms is to accurately locate the DNA splice sites. Although relatively high accuracy has been achieved by existing methods, most of these prediction methods are computationally extensive. Due to the enormous amount of DNA sequences to be processed, the computational speed is an important issue to consider. In this paper, we present a new machine learning method for predicting DNA splice sites, which first applies a Bayes feature mapping (kernel) to project the data into a new feature space and then uses a linear Support Vector Machine (SVM) as a classifier to recognize the true splice sites. The computation time is linear to the number of sequences tested, while the performance is notably improved compared with the Naive Bayes classifier in terms of classification accuracy, precision, and recall. Our classification results are also comparable to the solution quality obtained by the SVMs with polynomial kernels, while the speed of our proposed method is significantly faster. This is a notable improvement in computational modeling considering the huge amount of DNA sequences to be processed.

Keywords: Splice site prediction; SVM; Support vector machines; Bayes classifier; Machine learning; Splice Site Prediction Using Support Vector Machines with Bayes Kernel

1. Introduction

The advances in sequencing technologies have resulted in a large amount of DNA sequence information and therefore a dramatic increase in the size of genetic and genomic databases. The genome sequence information is produced as sequences of base pairs. However, no real knowledge of how the genome works is revealed unless different regions of the genome and their functions are characterized. Therefore, an important goal in bioinformatics is to accurately annotate the genome sequence information within an acceptable timeframe. Many computational efforts have recently been explored for predicting gene structures (Burge & Karlin, 1997) from DNA sequences and aiding the extensive analysis of the genome sequences, including recognizing translation initiation site of genes (Zien, Ratsch, Mika, Scholkopf, Lengauer & Muller, 2000), discovering transcriptional factor binding sites in promoter sequences (Lim, Sim, Chung, & Park, 2003), and identifying DNA splice sites (Jones & Watkins, 2000; Mache & Levi, 2000; Weber, 2001).

Gene expression in eukaryotes starts with the transcription of DNA sequences into mRNA sequences, followed by the processing of pre-mRNAs to mature mRNAs, and then the translation of mRNAs to proteins. Splicing is one of the primary post-processing steps of pre-mRNAs in eukaryotes. During splicing, the introns, the non-coding regions of genes, are removed from the primary transcripts, and the exons, the coding regions, are joined to form a continuous sequence that specifies a functional polypeptide (See Fig. 1 for illustration). The $5'$ side of the intron is a donor splice site and $3'$ side is an acceptor splice site. As most eukaryotic genes contain introns, many of which interrupt an exon within a codon, an important part of gene prediction in eukaryotes is therefore to predict splice sites.

This paper focuses on the problem of identifying DNA splice sites. Locating splice sites is an interesting problem to address because of the special structure in sequences around splice sites. The residual pairs *GT* and *AG* are often indicative of donor and acceptor splice sites. However, this canonical *GT–AG* rule does not always hold. Thus, it is natural to model the prediction of splice sites as a binary classification problem,

* Corresponding author. : Tel.: +1 814 865 4446; fax: +1 814 865 6426. Current address: Currently on leave at Singapore Management University.

E-mail addresses: yzhang@ist.psu.edu (Y. Zhang), chu@ist.psu.edu (C.-H. Chu), yixin@cs.uno.edu (Y. Chen), zha@cse.psu.edu (H. Zha), xji@sv.nec-labs.com (X. Ji).

0957-4174/$ - see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2005.09.052

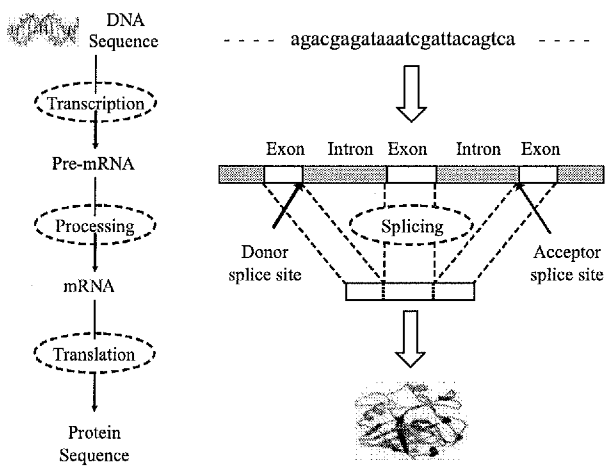

Fig. 1. The process of gene expression.

using DNA sequences with experimentally confirmed splice sites as positive training examples and those DNA sequences with *GT–AG* structure but confirmed *not* to be real splice sites as negative training examples.

Artificial neural networks (Acır & Güzeli, 2004), Bayesian classifiers (Stockwell, 1993), and SVMs (Min & Lee, 2005; Shin, Lee, & Kim, 2005) are important expert systems that have been applied to solve real world problems. Several of these expert systems have been applied to many interesting bioinformatics problems. For example Wang, Kuo, Chen, Hsiao, and Tsai (2005) built a knowledge sharing system for protein families (KSPF) using sequence pattern data mining and knowledge management. In this paper, we focus on the problem of recognizing true splice sites. Table 1 summarizes selected models used in predicting splice sites and their references. Although relatively high accuracy has been achieved with the methods currently available, almost all of the existing methods are computationally very demanding.

Table 1
Machine learning methods for splice sites prediction

Methods/models	Refs.
Statistical methods	
-Logit linear model	Brendel & Kleffe, 1998
-Quadratic discriminant analysis	Zhang & Luo, 2003
-Naïve Bayes classifier	Degroeve, De Baets, Van de Peer, & Rouz, 2002
Decision trees:	
-Maximal dependence decomposition (MDD)	Burge & Karlin, 1997
-MDD with Markov model	Pertea, Lin, & Salzberg, 2001
-C 4.5 induction tree	Patterson, Yasuhara, & Ruzzo, 2002
Artificial neural networks	
-Percepton	Weber, 2001
-Multi-layer Backpropagation	Mache & Levi, 2000; Reese, Eeckman, Kulp, & Haussler, 1997; Sonnenburg, Ratsch, Jagota, & Muller, 2002.
SVMs	
-Linear kernels	Degroeve *et al.*, 2002
-Polynomial kernels	Jones & Watkins, 2000; Patterson *et al.*, 2002; Zien *et al.*, 2000

Consequently, splice site prediction continues to be a major bottleneck in gene annotation.

In this study, we employ a linear SVM, which is computationally less extensive than SVMs with polynomial kernels, to recognize true splice sites. However, the DNA sequence information is given as strings while the SVM classifier can only take numerical inputs. Thus, the very first step is to encode or map the DNA sequences into numbers. A widely used encoding method is sparse encoding, where each letter in the DNA sequence is represented in four bits. But with this encoding method, the sequence data are in general linearly inseparable by SVMs. Instead, a novel mapping/encoding method derived from Bayes' rule is used to project the data into a new feature space where the true splice sites and the false splice sites can then be classified by linear SVMs. An advantage of the Bayes encoding method is that it takes into consideration the natural mutations in the DNA sequences with a probabilistic encoding framework. Experimental results have shown that the performance of our proposed method is comparable to that of SVMs with polynomial kernels in terms of accuracy, precision and recall, while the speed of our method is significantly faster. The computation time is linear to the number of sequences tested, while the performance is notably improved compared to the Naive Bayes classifier in terms of accuracy, precision and recall. Considering the overwhelming amount of DNA sequences that needs to be processed, the increased speed of our method is a very desirable property.

The rest of this paper is organized as follows. In Section 2, we give an introduction to SVMs. In Section 3, the Naive Bayes classifier is explained, and the Bayes feature mapping method is explored. In Section 4, we describe our experiment with splice site prediction and our theoretical analysis of the proposed method. In Section 5, the experiment results are presented. Finally, we give the conclusion in Section 6.

2. Support vector machines

Support Vector Machines (Vapnik, 1998) are powerful pattern recognition techniques that have been successfully applied to many machine learning tasks such as classification (Scholkopf, Burges, & Smola, 1999) and regression (Smola & Scholkopf, 2004). They have outperformed many other machine learning methods such as artificial neural networks and *k*-nearest neighbors and attracted a great deal of attention from the machine learning community because of many needed properties, including good generalization performance, robustness in the presence of noise, ability to deal with high dimensional data, and fast convergence. Classification problems are very common in bioinformatics and many of them involve high-dimensional and noisy data, with which SVMs are known to perform well. Applications of SVMs in bioinformatics include but are not limited to protein structure prediction (Hu, Pan, Harrison, & Tai, 2004), protein/gene function classification (Cai, Han, Ji, Chen, & Chen, 2003), protein subcellular localization prediction (Hua & Sun, 2001), splice site prediction (Degroeve et al., 2002; Jones & Watkins,

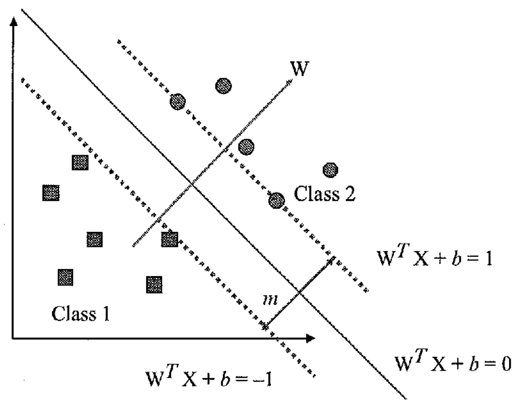

Fig. 2. Illustration of support vector machines.

2000; Patterson et al., 2002; Zien et al., 2000), and microarray data analysis (Brown, Grundy, Lin, Cristianini, Sugnet, Furey, Ares, & Haussler, 2000). Interested readers can refer to Noble (2004) for a comprehensive review.

We only provide a brief introduction of SVMs here. More detailed information can be found in Cristianini and Shawe-Taylor (2000); Burges (1998). Suppose we are given a set of examples X_i ($i=1,\ldots,N$), with corresponding labels $y_i \in \{-1, 1\}$ ($i=1,\ldots,N$), where -1 and 1 stand for the negative and positive classes, respectively. For a linearly separable set of data, there exists a set of separating hyperplanes $(\vec{w}.X)+b=0$, where $\vec{w}$ is a weight vector and b is the bias or threshold. SVMs find a unique separating hyperplane between the two classes in input space that maximizes the margin between the hyperplane and the classes. Fig. 2 is an illustration of how the separating hyperplane partitions the data.

The margin can be expressed as $2/||\vec{w}||^2$. The objective is then to minimize the squared Euclidean norm of $\vec{w}$, $||\vec{w}||^2$. The weight vector is generally expressed in terms of the linear combination of the training patterns: $\vec{w}=\sum_{i=1}^{n}\alpha_i y_i \vec{X}_i$ ($\alpha_i \geq 0$). When $\alpha_i > 0$, the corresponding pattern or support vector contributes to the hyperplane. With the maximal margin hyperplane as the decision hyperplane, the decision function can be written as: $f(X)=\mathrm{sgn}\left(\sum_{i=1}^{n}\alpha_i y_i \vec{X}_i.\vec{X}+b\right)$,where for a number x, $sgn(x)$ is 1 if x is larger than 0 and is -1 otherwise. For linearly separable data, the weight vector $\vec{w}$ can be obtained from the labeled training examples. Then the classification can be directly calculated with the above decision equation.

However, in many cases, the data are linearly inseparable. One way to solve this problem is to perform a nonlinear mapping of the input data into a new feature space, usually with a higher dimension, where the input vectors are linearly separable. Unfortunately, it is sometimes difficult to define the mapping function explicitly. Since only the dot product of the two new feature vectors under the new feature space is needed to calculate the decision function, an alternative way to solve the problem is to directly define the dot product under the new feature space with some kernel functions. Two typical kernel functions are:

$$\text{Polynomial kernel}: K(\vec{X}_i,\vec{X}_j)=(\vec{X}_i.\vec{X}_j+1)^d \quad (1)$$

$$\text{Gaussian kernel}: K(\vec{X}_i,\vec{X}_j)=\exp(-r||\vec{X}_i-\vec{X}_j||^2) \quad (2)$$

where d is the order of the polynomial kernel that may be set to any positive integer, $\exp(x)$ is an exponential function of x, and r is a measure of the radius of the Gaussian kernel.

3. The Proposed algorithm-SVMs with Bayes kernel

Fig. 3 depicts a generic framework of the classification process used in this study. The proposed algorithm is a hybrid of SVMs and a Bayes feature mapping (denoted as SVM-B). SVMs are a binary classification method that discriminates one set of data points from another. They only take numerical data as input. However, the DNA sequences are given as strings of nucleotides $\{A, T, C, G\}$. When using computational tools to analyze and classify the sequence data, an important step is encoding the sequences with numbers.

3.1. Sparse encoding

Sparse encoding is a widely used encoding schema which represents each nucleotide with 4 bits: A→1000, C→0100, G→0010, and T→0001 (Jones & Watkins, 2000). Suppose we have a DNA sequence of *AATCGTCAGT*. With the sparse encoding, the sequence is represented as: 1000|1000|0001|0100|0010|0001|0100|1000|0010|0001. Where '|' is a virtual separator used for illustration only and not shown in the actual encoded text.

With the sparse encoding method, the sequence data are in general linearly inseparable by SVMs for the problem of classifying true and false splice sites. Therefore, we need to employ either some mapping method or some kernel to make the data linearly separable. The use of kernel methods such as polynomial kernels usually introduces more computational complexity. On the other hand, the sparse encoding method results in identical distance metric measurements even though the nucleotides may not be biologically identical. When two sequences are compared, they either match or mismatch at a certain position with no intermediate degrees of similarity.

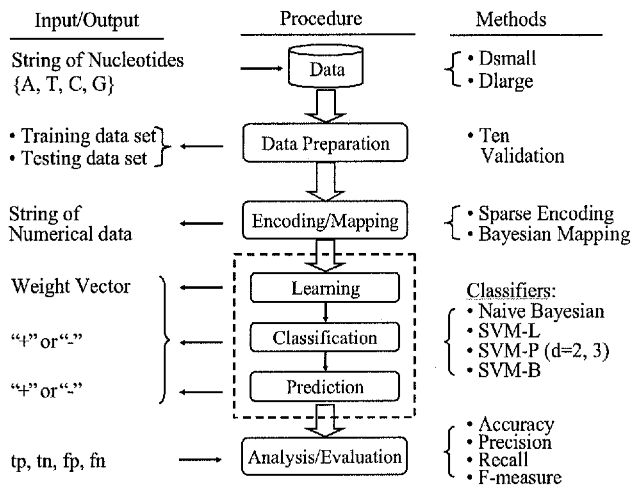

Fig. 3. Sketch of the classification algorithm.

For example, the distance between sequences *ACC* and *ATT* is considered the same as the distance between *ACC* and *AGG*—both have one matched nucleotide and two unmatched. However, in reality, there are always mutations in DNA sequences, with some forms of variation being more tolerable than others. A biologically more intuitive distance metric may assign a higher similarity score to the pair (*ACC* and *AGG*) than to (*ACC* and *ATT*). Therefore, we explore a new way of encoding DNA sequences.

3.2. The Bayes kernel

The Bayes feature mapping method is built from Bayes' rule. Suppose we have a set of examples $E=\{X_1, X_2,\ldots, X_N\}$. Let $X_i=\{a_1,\ldots, a_n\}(X_i \in E)$ denote a DNA sequence, where each a_j ($j=0, 1, \ldots, n$) is a nucleotide. Each $X_i \in E$ falls into one of two categories: c_1 or c_{-1}, where c_1 stands for true splice sites, and c_{-1} for false splice sites. According to Bayes' rule, the posterior probability of the category membership for each DNA sequence can be expressed as:

$$P(c_1|X_i) = \frac{P(X_i|c_1)P(c_1)}{p(X_i)} \tag{3}$$

$$P(c_{-1}|X_i) = \frac{P(X_i|c_{-1})P(c_{-1})}{p(X_i)} \tag{4}$$

Assume that a_j ($j=1, 2, \ldots, n$) are independent of each other. Thus, we get

$$P(X_i|c_1) = \prod_{j=1}^{n} P(a_j|c_1) \tag{5}$$

$$P(X_i|c_{-1}) = \prod_{j=1}^{n} P(a_j|c_{-1}) \tag{6}$$

After a few manipulations of the above equations, Eqs. 3 and 4 can be reformulated as:

$$\log(P(c_1|X_i)) = \sum_{j=1}^{n} \log(P(a_j|c_1)) - \log(P(X_i)) + a \tag{7}$$

$$\log(P(c_{-1}|X_i)) = \sum_{j=1}^{n} \log(P(a_j|c_{-1})) - \log(P(X_i)) + b \tag{8}$$

where $a=\log(P(c_1))$ and $b=\log(P(c_{-1}))$. With the Naive Bayes classifier, the classification decision is to maximize the log likelihood. X_i is assigned to the class c_k ($k=1$ or -1) that would maximize $\log(P(c_k|X_i))$. Thus, the decision function can be expressed as:

$$f(X_i) = \mathrm{sgn}(\log(P(c_1|X_i)) - \log(P(c_{-1}|X_i))) \tag{9}$$

Assuming uniform priority, i.e. $P(c_1)=P(c_{-1})$ and thus $a=b$, we get:

$$f(X_i) = \mathrm{sgn}\left(\sum_{j=1}^{n} \log(P(a_j|c_1))\right) - \sum_{j=1}^{n} \log(P(a_j|c_{-1})) \tag{10}$$

Eq. (10) can be reformulated as

$$f(x) = \mathrm{sgn}(\vec{w}.\vec{p}) \tag{11}$$

where $\vec{w}=\{w_1, w_2, \ldots, w_{2n}\}$ is the weight vector, and $\vec{p}=\{p_1, p_2, \ldots, p_{2n}\}$ is the posterior probability vector.

To estimate the probability for a set of DNA sequences, we need to first derive the positional profile of the data set. The positional profile of an alignment of DNA sequences with length l is defined as a $[4\times l]$ matrix $(p_{N,i})$, where $p_{N,i}$ is the frequency of nucleotide N in the ith position of the alignment. Once the positional profile is known for the DNA sequences, the p_i value can then be obtained by looking up the positional profile. See example 1 for a detailed illustration.

Example 1. : The Estimation of P_i values

Assume that we have the following set of DNA sequences to be analyzed:

TTCTTTTAGG
ACTTACTCTT
CATCCGTAAT
AAATGACTAT

Since the length of each DNA sequence is 10, the positional profile is a 4×10 matrix **P**. To arrive at the profile, we first consider the observed frequency of nucleotide A at position 1, which is $2/4=0.5$. Thus, we obtain $P_{A,1}=0.5$. Similarly, the entry for $P_{T,1}=1/4=0.25$. The remaining entries for $P_{N,I}$ are calculated in the same manner. Table 2 shows the positional profile for this set of DNA sequences. Suppose the DNA sequence that we would like to encode is: ATTCGACAAC. For each nucleotide at each position, we look at Table 2 to obtain its corresponding Pi value. For instance, P_1 is $P_{A,1}$, which is 0.50. P_2 is $P_{T,2}$, which is 0.25. Therefore, the encoding for that DNA sequence is: 0.50, 0.25, 0.50, 0.25, 0.25, 0.25, 0.25, 0.50, 0.50, 0.00.

In the Naive Bayes classifier, we have $w_i=1$ for $i \in\{1, \ldots, n\}$, $w_i=-1$ for $i\in\{n+1, \ldots, 2n\}$, $p_i=P(x_i \mid c_1)$ for $i\in\{1, \ldots, n\}$, and $p_i=P(x_{i-n}|c_{-1})$ for $i\in\{n+1, \ldots, 2n\}$. Suppose we have the positional profiles generated from the positive examples and negative examples as shown in Table 3 (a) and (b), respectively.

Assume that we want to encode the following DNA sequence: ATTCGACAAC. We first look up the corresponding $P(N,i)$ values from the positive positional profile and then the negative profile. Table 4 shows these values. The Bayes feature mapping considers both positive and negative encodings. Thus, we obtain the encoded code as a vector of length 20: [0.5, 0.25, 0.50, 0.25,

Table 2
Example of position profile

Nucleotide (n)	Position (i)									
	1	2	3	4	5	6	7	8	9	10
A	0.50	0.50	0.25	0.00	0.25	0.25	0.00	0.50	0.50	0.00
C	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.00	0.00
G	0.00	0.00	0.00	0.00	0.25	0.25	0.00	0.00	0.25	0.25
T	0.25	0.25	0.50	0.75	0.25	0.25	0.75	0.25	0.25	0.75

Table 3
Positional profile for example 2

Nucleotide(n)	Position (i)									
	1	2	3	4	5	6	7	8	9	10
(a) Positive positional profile										
A	0.50	0.50	0.25	0.00	0.25	0.25	0.00	0.50	0.50	0.00
C	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.00	0.00
G	0.00	0.00	0.00	0.00	0.25	0.25	0.00	0.00	0.25	0.25
T	0.25	0.25	0.50	0.75	0.25	0.25	0.75	0.25	0.25	0.75
(b) Negative positional profile										
A	0.25	0.50	0.50	0.00	0.25	0.25	0.50	0.25	0.50	0.00
C	0.25	0.00	0.25	0.25	0.25	0.50	0.25	0.25	0.25	0.50
G	0.25	0.25	0.00	0.25	0.50	0.25	0.25	0.50	0.00	0.50
T	0.25	0.25	0.25	0.50	0.00	0.00	0.00	0.00	0.25	0.00

Table 4
Encoding using bayes feature mapping

Position (i)	1	2	3	4	5	6	7	8	9	10
Nucleotide	A	T	T	C	G	A	C	A	A	C
P_i (positive)[a]	0.50	0.25	0.50	0.25	0.25	0.25	0.25	0.50	0.50	0.00
P_i (negative)[b]	0.25	0.25	0.25	0.25	0.50	0.25	0.25	0.25	0.50	0.50

[a] This is the $P(a_i|c_1)$ value in Eq. (10).
[b] This is the $P(a_i|c_{-1})$ value in Eq. (10).

0.25, 0.25, 0.25, 0.5, 0.5, 0.0, 0.25, 0.25, 0.25, 0.25, 0.50, 0.25, 0.25, 0.25, 0.5, 0.5]. This vector, representing features of the DNA sequence, is then input to the SVM classifier.

Example 2. : Illustration of Bayes feature mapping

The Naive Bayes classifier is guaranteed to be optimal only when the attributes are independent in the given class. However, assumption of independence may not always hold. Thus, the estimation of the distribution of X_i may not be accurate. In addition, the Naive Bayes classifier assumes each position is equally important, which might not be true in the case of splice site prediction. Some positions may be essential while others may be trivial. The idea here is to use the Bayes mapping to project the data into a new feature space and then use linear SVM to determine the optimal weight vectors. We expect this to improve the classification accuracy gained by the Naive Bayes classifier while maintaining the simplicity in computation. A numerical illustration of the prediction process using the proposed SVM-B is given in Fig. 4.

4. Experimental design

We test the relative performance of the combined Bayes Mapping and SVMs method (denoted as SVM-B) in recognizing true splice sites, with a series of 10-fold cross validation experiments. We compare the performance of the proposed SVM-B method with Naive Bayes classifier, SVMs with linear kernel (SVM-L) and SVMs with polynomial kernels (SVM-P) with the d-value equal to 2 or 3. As a benchmark, we use the traditional sparse encoding method (Jones & Watkins, 2000) for these SVMs methods.

We report the results in terms of *accuracy*, *precision*, *recall* and *F-measure*, the common measures used in data mining research (Witten & Frank, 2000). These measures are defined as follows:

$$\text{Accuracy} = (tp + tn)/(tp + tn + fp + fn) \tag{12}$$

$$\text{Precision} = tp/(tp + fp) \tag{13}$$

$$\text{Recall} = tp/(tp + fn) \tag{14}$$

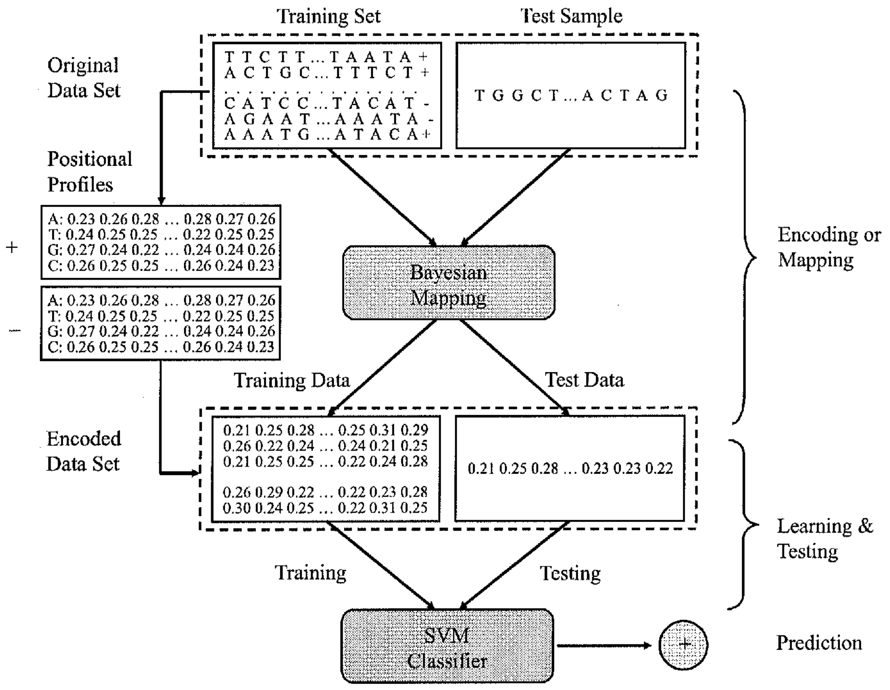
Training Set
Test Sample
Original Data Set
T T C T T ... T A A T A +
A C T G C ... T T T C T +
C A T C C ... T A C A T -
A G A A T ... A A A T A -
A A A T G ... A T A C A +
T G G C T ... A C T A G
Positional Profiles
+
A: 0.23 0.26 0.28 ... 0.28 0.27 0.26
T: 0.24 0.25 0.25 ... 0.22 0.25 0.25
G: 0.27 0.24 0.22 ... 0.24 0.24 0.26
C: 0.26 0.25 0.25 ... 0.26 0.24 0.23
−
A: 0.23 0.26 0.28 ... 0.28 0.27 0.26
T: 0.24 0.25 0.25 ... 0.22 0.25 0.25
G: 0.27 0.24 0.22 ... 0.24 0.24 0.26
C: 0.26 0.25 0.25 ... 0.26 0.24 0.23
Bayesian Mapping
Encoding or Mapping
Training Data
Test Data
Encoded Data Set
0.21 0.25 0.28 ... 0.25 0.31 0.29
0.26 0.22 0.24 ... 0.24 0.21 0.25
0.21 0.25 0.25 ... 0.22 0.24 0.28
0.26 0.29 0.22 ... 0.22 0.23 0.28
0.30 0.24 0.25 ... 0.22 0.31 0.25
0.21 0.25 0.28 ... 0.23 0.23 0.22
Learning & Testing
Training
Testing
SVM Classifier
+
Prediction

Fig. 4. Numerical illustration of the algorithm.

Table 5
Illustration used in measures

Real	Predict	
	True	False
True	*Tp*	*Fn*
False	*Fp*	*tn*

$$F-\text{measure} = (2\times\text{Precision}\times\text{Recall})/(\text{Precision}+\text{Recall}) \quad (15)$$

where *tp* is the number of sequences with real splice sites which are predicted to be true (true positives), *tn* is the number of sequences without real splice sites which are predicted to be false (true negatives), *fp* is the number of sequences without real splice sites which are predicted to be true (false positives), *fn* is the number of sequences with real splice sites which are predicted to be false (false negatives). They are illustrated in Table 5.

Accuracy represents the percentage of correct predictions. Precision measures the ratio of the number of correctly predicted splice sites to the total number of predicted splice sites. Recall is the ratio of the number of correctly predicted splice sites to the total number of real splice sites.

Larger values of accuracy, precision, and recall represent good classification performance.

We formulate two categories of hypotheses to statistically test the difference between SVM-B and benchmarked methods—Naive Bayes classifier, SVM-L, SVM-P ($d=2$) and SVM-P ($d=3$)—in terms of solution quality (prediction accuracy, precision, recall), and computational efficiency (CPU times).

Hypothesis Ha: There is no difference in solution quality obtained by SVM-B and benchmarked methods.
Hypothesis Hb: There is no difference in computational efficiency between SVM-B and benchmarked methods.

Two data sets, *Dsmall* and *Dlarge*, obtained from the open literature (Weber, 2001) are used for the experiments. They contain 1000 and 10,000 nucleotide sequences of splice site data, respectively. All the sequences are 50 bases long, and for each sequence the *GT–AG* structure occurs in the middle. Both data sets contain examples of true splice sites as well as false splice sites. Each data set is randomly split into ten subsets of equal size. Each time, one subset is used for testing, and the rest are combined and used for training. First, the positional profiles are estimated for the true splice sites from the positive examples in the training set and for false splice sites from negative examples. We assume that the sequences in the training set are representative of all DNA sequences. Therefore, entries of the positional profiles for the true and false splice sites can be approximated with the observed frequency of occurrence of given nucleotides at given positions in the positive and negative examples of the training set, respectively. This is to estimate the posterior probability $P(x_{ik}|c_j)$. ($i\in\{1,\ldots,n\}$. $k\in\{A,T,C,G\}$,. $j\in\{1,-1\}$) from the training set, where c_1 means the true splice sites, c_{-1} means the false splice sites, and x_i represents the ith nucleotide. Then, the logarithms of each posterior probability $\log(P(x_{ik}|c_j))$ are input to a linear SVM. The open source software SVMlight, which can be downloaded from http://www.support-vector.net, is used for the SVM learning and classification (Joachims, 1999). Parameter C of the SVM classifiers is empirically set to be 150 based on our experiments (result not shown).

We also perform a theoretical analysis in order to provide some theoretical evidence to support the proposal that our method is a better and simpler classifier for recognizing true splice sites. We use the Vapnik-Chervonenkis (VC) dimension (Vapnik, 1995; Blumer, Ehrenfeucht, Haussler, & Warmuth, 1989) as a benchmark. The VC-dimension of a classifier is defined as the largest number of vectors that can be separated in all possible ways. The VC-dimension of a classifier often involves the dimensionality of the feature space. Given some machine f, let h be its VC dimension; then h is a measure of f's power. Vapnik showed that with probability $1-\eta$,

$$\text{TestErr}(\alpha)\le\text{TrainErr}(\alpha)+\sqrt{\frac{h(\log(2R/h)+1)-\log(\eta/4)}{R}} \quad (16)$$

where η ranges from 0 to 1, TrainErr(α) and TestErr(α) are the training error and testing error, respectively, and R is the number of the data points in the training set. This gives us a way to estimate the error on future data based only on the training error and the VC-dimension of the classifier.

Table 6
Computational results for a small data set (*Dsmall*)

Methods	Statistics	Performance measures				
		Accuracy	Precision	Recall	*F*-measure	CPU time[a]
Naïve Bayes	Average	87.8	89.0	86.9	89.2	0.05
	SD	3.7	3.3	5.1	–	–
SVM-L$^+$	Average	86.6	88.2	85.1	86.6	2.9
	SD	2.6	2.5	6.6	–	–
SVM-P$^+$ ($d=2$)	Average	88.9	89.3	88.8	89.0	3.5
	SD	2.7	3.6	3.9	–	–
SVM-P$^+$ ($d=3$)	Average	89.8	90.9	88.8	89.8	4.9
	SD	3.2	3.6	4.6	–	–
SVM-B$^+$	Average	89.2	90.9	87.8	89.3	0.8
	SD	3.4	3.9	5.3	–	–

[a] Unit in seconds; $+C=150$.

Table 7
Paired-t tests for a small data set (*Dsmall*)

Methods	Statistics	SVM-B			
		Accuracy	Precision	Recall	CPU time
Naïve	t-value	1.98	3.79	−0.27	43.19
Bayes	P-value	0.04	0.002	0.40	0.000
SVM-L	t-value	4.12	4.07	2.21	−83.82
	P-value	0.001	0.001	0.03	0.000
SVM-P ($d=2$)	t-value	1.00	−0.34	1.09	−60.57
	P-value	0.170	0.37	0.15	0.000
SVM- P($d=3$)	t-value	−0.13	−2.03	1.52	−113.0
	P-value	0.450	0.04	0.08	0.000

5. Results and analyses

5.1. Experimental results

The results of the experiments are shown in Tables 6–9. Tables 6 and 7 summarize the computational results and paired-t test for a small data set (*Dsmall*), where the accuracy, precision, recall and CPU times were averaged from the ten-fold cross validation experiments. Their standard deviations were also computed. Based on the average precision and recall, we then computed the overall F-measure.

As can be seen from the computational results, in terms of accuracy and F-measure, the proposed SVM-B method outperformed Naive Bayes classifier and SVM classifier with linear kernel and polynomial kernel of $d=2$ for a small data set (*Dsmall*). The results of SVM classifier with polynomial kernel of $d=3$ were slightly better. The paired-t test indicated that the proposed SVM-B method is significantly better than Naïve Bayes classifier in terms of prediction precision and SVM-L (with sparse encoding) in terms of prediction accuracy and precision. The differences in other results, however, were not statistically significant. In terms of computational efficiency, as we can see in the tables, the proposed method was significantly faster than the conventional SVMs with both linear kernels and polynomial kernels. However, it was slower than the Naive Bayes classifier.

Tables 8 and 9 summarize the computational results and paired-t test for a large data set (*Dlarge*). As can be seen from Table 8, when the *Dlarge* data set was used, the proposed SVM-B method outperformed all the other methods. The paired-t tests indicated that the proposed SVM-B method is significantly better than the Naive Bayes classifier in terms of prediction precision and recall. Also, the proposed method was significantly better than SVM-P ($d=2$) in terms of prediction accuracy and SVM-P ($d=3$) in term of prediction recall. The differences in other results, however, were not statistically significant. In terms of computational efficiency, as can be seen from the tables, our proposed method was significantly faster than the conventional SVMs with both linear kernels and polynomial kernels, but slower than the Naive Bayes classifier. The differences in computational times were all statistically significant.

5.2. Theoretical analysis

Intuitively speaking, a smaller VC-dimension should correspond to a simpler model. Under identical training errors, a smaller VC-dimension corresponded to a smaller upper bound on the test error. Table 10 lists the upper bounds of VC-dimensions for the SVM classifiers compared here (estimated by the SVM training program). As we can see here, when the Bayes kernel was used, the upper bound of VC-dimension was much smaller than that of the linear kernel and those of the polynomial kernels in conjunction with sparse encoding. This suggests that in general SVM with Bayes kernel generalizes better than SVMs with linear and polynomial kernels.

5.3. Discussion

In summary, the computational results and theoretical analysis confirmed that the proposed Bayes feature-mapping scheme works well with the linear SVMs. Its performance was fairly robust especially when applied to a large data size problem. This is a very appealing feature, as most real world applications involve analyzing huge volumes of genome sequence data. Moreover, the standard deviations of the computational results were all relatively small, indicating that the ten-fold experiment was reliable and appropriate for splice site prediction. This is a very desirable property considering the overwhelming amount of DNA sequences that needs to be processed.

Table 8
Computational results for a large data set (*Dlarge*)

Methods	Statistics	Performance measures				
		Accuracy	Precision	Recall	F-measure	CPU time[a]
Naïve Bayes	Average	91.1	90.8	91.5	91.1	0.48
	SD	1.0	1.6	1.4	–	–
SVM-L$^+$	Average	91.0	91.5	90.3	90.9	46.2
	SD	1.3	1.4	2.0	–	–
SVM-P$^+$ ($d=2$)	Average	89.2	89.0	89.4	89.2	328.2
	SD	0.7	0.8	1.3	–	–
SVM-P$^+$($d=3$)	Average	90.7	91.0	90.5	90.7	491.3
	SD	0.9	1.0	1.3	–	–
SVM-B$^+$	Average	91.4	92.0	90.6	91.3	10.8
	SD	0.9	1.3	1.4	–	–

[a] Unit in seconds; $+C=150$.

Table 9
Paired-*t* tests for a large data set (*Dlarge*)

Methods	Statistics	SVM-B			
		Accuracy	Precision	Recall	CPU time
Naïve	*t*-value	2.42	7.29	−4.75	532.73
Bayes	*P*-value	0.02	0.000	0.000	0.000
SVM-L	*t*-value	2.30	2.01	−0.39	−228.27
	P-value	0.02	0.04	0.35	0.000
SVM-P	*t*-value	3.19	2.04	2.60	−752.36
($d=2$)	*P*-value	0.005	0.04	0.014	0.000
SVM-P	*t*-value	−0.26	−2.79	4.08	−707.37
($d=3$)	*P*-value	0.40	0.01	0.001	0.000

Table 10
Upper bound of VC-dimension given by the SVM classifiers

Algorithm	Average upper bound of VC dimension[a]	
	Dsmall	Dlarge
SVM-B	82	136
SVM-L	173	937
SVM-P ($d=2$)	295	1203
SVM ($d=3$)	396	1810

[a] The result is averages of the upper bounds obtained with 10-fold cross validation.

6. Conclusion

Predicting splice sites is an important part of gene structure prediction. During the past years, several emerging machine learning methods such as artificial neural networks, perceptron, and support vector machines have been employed to approach the problem with sufficiently high accuracy in recognizing true splice sites. However, almost all of the existing methods are computationally extensive; therefore, splice site prediction remains a major bottleneck in gene annotation.

In this paper, we presented a novel idea of constructing a mapping method from Bayes' rule. This mapping method was then integrated with SVM classifier and applied to the problem of splice site prediction in DNA sequences. Experiments on two data sets with ten-fold cross validation demonstrated that our method outperforms the benchmark methods: Naive Bayes classifier, SVM classifiers with linear kernel and polynomial kernel ($d=2$ and $d=3$) in terms of accuracy, precision, recall, and F-measure.

The results confirmed that the proposed SVM-B method enhances the solution quality of Naive Bayes classifier for DNA splice site prediction. Furthermore, when the speed of computation was taken into consideration, the method was as quick as the Naive Bayes classifier and performed much faster than SVM with non-linear kernel methods. Solution quality, computational speed, user-friendliness, flexibility, and simplicity are some of the key but conflicting factors in selecting and implementing new technology. The common industry practice is to trade off simplicity and speed of performance for solution quality. Our study showed that by carefully selecting the proper encoding method, the linear SVMs can perform as well as complicated SVMs with polynomial kernels for splice site prediction, while maintaining computational efficiency. Therefore, with the proposed method there is no need for us to trade reduced accuracy for improved efficiency in SVMs applications.

Bayes classifier is a simple generative learning method and SVM classifier represents a type of discriminative learning method (Ng & Jordan, 2001). This proposed method represents an effort in integrating the generative learning methods into discriminative learning. With the success of the proposed method, more complex generative learning methods, such as Hidden Markov Model (HMM) may be integrated into SVM classifiers in a similar fashion. Therefore, future research can introduce some other model building techniques such as HMM or improved Bayes' rules to improve the results of splice site prediction in DNA sequencing.

References

Acır, N., & Güzeli, C. (2004). Automatic recognition of sleep spindles in EEG by using artificial neural networks. *Expert Systems with Applications*, *27*(3), 451–458.

Blumer, A., Ehrenfeucht, A., Haussler, D., & Warmuth, M. K. (1989). Learnability and the Vapnik-Chervonenkis dimension. *Journal of the Association for Computing Machinery*, *36*(4), 929–965.

Brendel, V., & Kleffe, J. (1998). Prediction of locally optimal splice sites in plant pre-mRNA with applications to gene identification in *Arabidopsis thaliana* genomic DNA. *Nucleic Acids Research*, *26*, 4748–4757.

Brown, M. P. S., Grundy, W. N., Lin, D., Cristianini, N., Sugnet, C., Furcy, T. S., et al. (2000). Knowledge-based analysis of microarray gene expression data using support vector machines. *Proceedings of the National Academy of Science USA*, *97*(1), 262–267.

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

Burges, C. (1998). A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, *2*(2), 1–47.

Cai, C. Z., Han, L. Y., Ji, Z. L., Chen, X., & Chen, Y. Z. (2003). SVM-Prot: Web-based support vector machine software for functional classification of a protein from its primary sequence. *Nucleic Acids Research*, *31*(13), 3692–3697.

Cristianini, N., & Shawe-Taylor, J. (2000). *An introduction to support vector machines and other kernel based learning methods*. Cambridge, UK: Cambridge University Press.

Degroeve, S., De Baets, B., Van de Peer, Y., & Rouz, P. (2002). Feature subset selection for splice site prediction. *Bioinformatics*, *18*, S75–S83.

Hu, H., Pan, Y., Harrison, R., & Tai, P. C. (2004). Improved protein secondary structure prediction using support vector machine with a new encoding scheme and an advanced tertiary classifier. *IEEE Transactions on Nanobioscience*, *3*(4), 265–271.

Hua, S., & Sun, Z. (2001). Support vector machine approach for protein subcellular localization prediction. *Bioinformatics*, *17*(8), 721–728.

Joachims, T. (1999). Making large-scale SVM Learning practical. In B. Schölkopf, C. Burges, & A. Smola (Eds.), *Advances in kernel methods—Support vector learning*. Cambridge, MA: MIT-Press.

Jones, D., & Watkins, C. (2000). *Comparing kernels using synthetic DNA and genomic data*. Technical report. Department of Computer Science, University of London, UK.

Lim, M. E., Sim, J. S., Chung, M. G., & Park, S. H. (2003). Prediction of transcription factor binding sites with suffix arrays. *Genome Informatics*, *14*, 400–401.

Mache, N., & Levi, P. (2000). *Parallel neural network training and cross validation on a Cray T3E system and application to splice site prediction in human DNA*. Technical report. Stuffgart, Germany: Institute of Parallel and Distributed High Performance Systems.

Min, J. H., & Lee, Y. C. (2005). Bankruptcy prediction using support vector machine with optimal choice of kernel function parameters. *Expert Systems with Applications*, *28*(4), 603–614.

Ng, A., & Jordan, M. (2001). *On discrminiative vs. generative classifiers: A comparison of logistic regression and Naive Bayes Neural Information Processing Systems—NIPS* (pp. 841–848) Canada.

Noble, W. S. (2004). Support vector machine applications in computational biology. In B. Schoelkopf, K. Tsuda, & J. P. Vert (Eds.), *Kernel methods in computational biology* (pp. 71–92). Cambridge, MA: MIT Press.

Patterson, D. J., Yasuhara, K., & Ruzzo, W. L. (2002). Pre-mRNA secondary structure prediction aids splice site prediction. *Pacific Symposium on Biocomputing*, 223–234.

Pertea, M., Lin, X., & Salzberg, S. L. (2001). GeneSplicer: A new computational method for splice site prediction. *Nucleic Acids Research*, *29*(5), 1185–1190.

Reese, M. G., Eeckman, F. H., Kulp, D., & Haussler, D. (1997). Improved splice site detection in genie. *Journal of Computational Biology*, *4*(3), 311–323.

Scholkopf, B., Burges, C., & Smola, A. (1999). *Advances in kernel methods: Support vector learning*. Cambridge, MA: The MIT Press.

Shin, K. S., Lee, T. S., & Kim, H. J. (2005). An application of support vector machines in bankruptcy prediction model. *Expert Systems with Applications*, *28*(1), 127–135.

Smola, A., & Scholkopf, B. (2004). A tutorial on support vector regression. *Statistics and Computing*, *14*, 199–222.

Sonnenburg, S., Ratsch, G., Jagota, A., & Muller, K. R. (2002). New methods for splice site recognition. *Proceedings of the international conference on artificial neural networks. Lecture notes in computer science.* (pp. 329–336) *Vol. 2415*.

Stockwell, D. R. B. (1993). LBS: Bayesian learning system for rapid expert system development. *Expert Systems with Applications*, *6*(2), 137–147.

Vapnik, V. (1995). *The nature of statistical learning theory*. New York: Springer.

Vapnik, V. N. (1998). *Statistical learning theory. Adaptive and learning systems for signal processing communications and control*. New York, NY: Wiley.

Wang, H. C., Kuo, H. C., Chen, H. H., Hsiao, Y. Y., & Tsai, W. C. (2005). KSPF: Using gene sequence patterns and data mining for biological knowledge management. *Expert Systems with Applications*, *28*(3), 537–545.

Weber, R. (2001). DNA splice site prediction with kernels and voting. *Proceedings of international conference on mathematical and engineering techniques in medicine and biological sciences, Nevada*.

Witten, I. H., & Frank, E. (2000). *Data mining: Practical machine learning tools and techniques with JAVA implementations*. San Francisco, CA: Morgan Kaufmann.

Zhang, L., & Luo, L. (2003). Splice site prediction with quadratic discriminant analysis using diversity measure. *Nucleic Acids Research*, *31*(21), 6214–6220.

Zien, A., Ratsch, G., Mika, S., Scholkopf, B., Lengauer, T., & Muller, K. (2000). Engineering support vector machine kernels that recognize translation initiation sites. *Bioinformatics*, *16*(9), 799–807.