Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics
San Antonio, TX, USA - October 2009

Splice Site Detection in DNA Sequences Using a Fast Classification Algorithm

Jair Cervantes, Xiaoou Li
Department of Computer Science
CINVESTAV
Av. IPN 2508. 07360 México D.F., México
jcanales@cs.cinvestav.mx

Wen Yu
Department of Automatic Control
CINVESTAV
Av. IPN 2508. 07360 México D.F., México
yuw@ctrl.cinvestav.mx

***Abstract*—Support Vector Machines (SVMs) are known to be excellent algorithms for classification problems. The principal disadvantage of SVMs is due to its excessive training time in large data set, such as DNA sequences. This paper presents a novel SVMs classification method which reduces significantly the input data set using Bayesian technique. Using this system, we are able to predict with a high accuracy huge data sets in a reasonable time. The system has been tested successfully on large splice-junction gene sequences (DNA). Experimental results show that the accuracy obtained by the proposed algorithm is comparable (98.2) with other SVMs implementations such as SMO (98.4%), LibSVM (98.4%), and Simple SVM (97.6%). Furthermore the proposed approach is scalable to large data sets with high classification accuracy.**

***Index Terms*—Splice sites detection, DNA, Large data sets, Bayesian classification, SVM.**

I. Introduction

There is a growing need in biological research for data integration methodologies which efficiently examine, drive, exploit and integrate medical or biological information. This need is particularly prominent in the research field of complex processes of mRNA splicing [8]. Computational tools have gotten to be essential in many fields of Bioinformatics. The tasks include classification of sequences, detection of similarities, separation of proteins into sequences of deoxyribonucleic acid (splicing), prediction of molecular structures, discovery of new drugs and therapies among other things. Huge data quantities used in biological experiments have provoked the creation of many databases that contain genomes, sequences of proteins and another types of biological data [5]. Therefore, the basic research in Bioinformatics is oriented to design and implementation of systems that allow solving problems of storage, drive, processing and analysis of huge quantities of DNA.

The problems of gene identification and gene structure prediction have received a wide attention. The classification of gene sequence into regions that code for genetic material and regions that do not is a challenging task in DNA sequence analysis. The recognition of genes is not an easy challenge. It is due to size of DNA sequences and sometimes regions that encode in proteins (exons) can be interrupted by regions that do not encode (introns). These sequences are characterized, however they are not clearly defined by local characteristics at splicing sites. Identifying exons into DNA sequences presents a computational challenge. In some organisms the introns are small regions (30 base pairs) and the splicing sites are fully characterized. However, in some other sequences, including human genome, it is a great problem to localize the correct transition between the regions that encode and the ones that not. Furthermore, the genes in many organisms splice of different way, which complicates considerably the task. On the other hand, splice sites fall into two categories: donor sites of introns and acceptor sites of introns. These sites display certain characteristic patterns, e.g. 99% of donor sites begin with base pairs GT while 99% acceptor sites end with based pairs AG. However, not all locations with base pairs GT or AG are necessarily splice sites. Some occurrences of AG or GT occur outside of a gene or inside an exon. These are called decoys, because they do not indicate the presence of a splice site.

SVM has received considerable attention due to its optimal solution, discriminative power and performance. Lately some SVMs classification algorithms have been used in splice site detection with acceptable accuracies [1] [3] [4] [12] [13]. Cheng et al [1] use SVMs in order to predict mRNA polyadenylation sites [poly(A) sites] the method can help identify genes, define gene boundaries, and elucidate regulatory mechanisms. Damaevicius [3] and Xia [12] use SVMs in order to detect splice-junction (intron-exon or exon-intron) sites in DNA sequences. In [13] the authors use a support vector machine to discover sequence information that could be used to distinguish real exons from pseudo exons. Some authors have been using SVMs for the detection of splicing sites. However, the principal disadvantage of SVMs is due to memory requirements grows with square of input data points, so training complexity of SVMs is highly dependent on the size of a data set [2].

This paper presents a novel Donnor/Acceptor classification model using SVMs and Bayesian classification. According to architecture of SVMs, the training data that are near boundaries are important data points and the data far away from the hyperplane do not contribute on SVM training. Therefore, the training time can be reduced if the data far from the hyperplane are deleted. As a first step we obtain an initial support vectors (SV) data sets from a small dataset, in a second step we train a Bayesian classifier with SV and non-SV obtained in the

978-1-4244-2794-9/09/$25.00 ©2009 IEEE

previous step, and reduce the original input data set obtaining the most important data points (SVs candidate), finally we use the SVs candidate obtained in the previous step and use a second stage of SVMs. The rest of the paper is organized as following: Section II reviews some preliminaries. Section III focuses on explaining the proposed SVM classification algorithm. Section IV shows experimental results. Conclusion is given in Section V.

II. Preliminaries

In this section we highlight some preliminaries regarding to SVM, Bayesian classification and DNA classification.

A. Support Vector Machines

As first method we use SVMs, considering binary classification, The generated optimal classification function can be written as

$$y_i = sign\left(\sum_{i=1}^{n} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}_j) + b\right) \tag{1}$$

where $\mathbf{x} = [x_1, x_2, \ldots, x_l]$ is the input data, α_i and y_i are Lagrange multipliers. A new object x can be classified using (1). The vector $\mathbf{x}_i$ is shown only in the way of inner product. There is a Lagrangian multiplier α for each training point. When the maximum margin of the hyperplane is found, only the closed points to the hyperplane satisfy $\alpha > 0$. These points are called support vectors SV, the other points satisfy $\alpha = 0$, so the solution vector is sparse. Where b is determined by Kuhn-Tucker conditions:

$$\begin{array}{l} \frac{\partial L}{\partial w} = 0, w = \sum_{i=1}^{n} \alpha_i y_i \varphi(x_i) \\ \frac{\partial L}{\partial b} = 0, \sum_{i=1}^{n} \alpha_i y_i = 0 \\ \frac{\partial L}{\partial \xi_i} = 0, \alpha_i - c \geq 0 \\ \alpha_i \left\{ y_i \left[w^T \varphi(x_i) + b \right] - 1 + \xi_i \right\} = 0 \end{array} \tag{2}$$

B. Bayesian classification

As second method we use Bayesian classification, a Bayesian classifier is based on the principle of Bayes decision theory which provides a fundamental methodology for solve pattern classification problems when the probability distribution of the pattern is known. Bayesian classifier uses probabilistic approach to assign the class to a sample. Let C denote a class from the set of m classes $(C_1, C_2, C_2, ..., C_m)$ and X_k is a sample described by a vector of k features i.e. $X_k = [x_1, x_2, ..., x_k]$. Bayesian classifier computes the posterior conditional probability $p(Ci \mid Xk)$ using Bayes rule:

$$p(C_i \mid X_k) = \frac{p(X_k \mid C_i) p(C_i)}{p(X_k)}, i = 1, 2, ..., m. \tag{3}$$

In Eq. 3 $p(Ci \mid Xk)$, $p(C_i)$ and $p(X_k)$ are calculated using training data. According to Bayesian theory, for a given observation X_k, one predicts a class for which the posterior probability i s maximum:

$$f(X_k) = \arg\max_i p(C_i \mid X_k)$$

C. DNA Encoding

There are several techniques to encode by features vectors the DNA sequences [5][7]. Each technique is based on the most important features to be shown. Sparse encoding is a widely used encoding schema which represents each nucleotide with 4 bits: $A \rightarrow 1000, C \rightarrow 0100, G \rightarrow 0010$ and $T \rightarrow 0001$ [6]. Suppose we have a DNA sequence of ACTTCGTAAGAT. With the sparse encoding, the sequence is represented as: 1000 | 0100 | 0001 | 0001 | 0100 | 0010 | 0001 | 1000 | 1000 | 0010 | 1000 | 0001. where | is a virtual separator used for illustrate the example.

Some researchers argument the disadvantages of this method that "the sequence of data is lineally not separable in the case of classifying false and true sites splices". However, it is possible to solve this problem by using some kernels. Although using kernels increments the computational complexity, our proposed method is not affected because the computational complexity of the proposed algorithm can be greatly reduced for using a small part of training data set.

In order to improve the classification accuracy, we use 20 additional features with the sparse encoding schema. The first 16 components define the nucleotide pairs into a DNA sequence, which are defined as $\beta = \{(x_{AA}), (x_{AC}), (x_{AG}), (x_{AT}), \ldots, (x_{TA}), (x_{TC}), (x_{TG}), (x_{TT})\}$. When some nucleotide pair is in the sequence, it is marked with 1 and an absence of this pair is marked with 0. The DNA sequence, ACTTCGTAAGAT can be encoding by this schema as: 1 1 1 1 0 0 1 1 1 0 0 1 1 1 0 1. The final 4 components correspond to the abundance of each nucleotide in the sequence. Each sequence is represented by a vector with 4 features which represent the relative presence of each nucleotide. The sequence ACTTCGTAAGAT can be encoding by this schema as: $\gamma = \{f_A, f_C, f_G, f_T\} = \{0.33, 0.16, 0.16, 0.33\}$.

The proposed encoding schema allows to obtain completely the nucleotides of each sequence, encoding the pairs show the importance of some pairs in the sequence, and obtain the abundance of each nucleotide in the sequence. The previous DNA sequence can be encoding by the complete schema as: 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1 |1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1 | 0.33, 0.16, 0.16, 0.33. Where | is a virtual separator which objective is just illustrate the three techniques used. With the proposed encoding schema SVM can use the features and discriminate between the classes.

III. Fast SVM via Bayesian classification

SVM is a powerful technique for binary classification and regression. SVMs are based in the idea of finding an optimal classification hyperplane that divide two classes. When the data points are not linearly separable the SVM maps the data points to a highly-dimensional feature space where a separating hyper-plane can be found. This mapping implicitly transforms the input space into another high dimensional feature space by the kernel trick. However, to find the classification hyperplane is computationally very expensive. Training a SVM is usually

posed as a quadratic programming (QP) problem and to solve this problem is computationally expensive because it has a quadratic complexity, consequently SVMs needs enormous quantities of time and memory. To train a classic SVM with one million of records would be take years. The Figure 1 shows the training time with checkerboard data set using two fast SVM implementations.

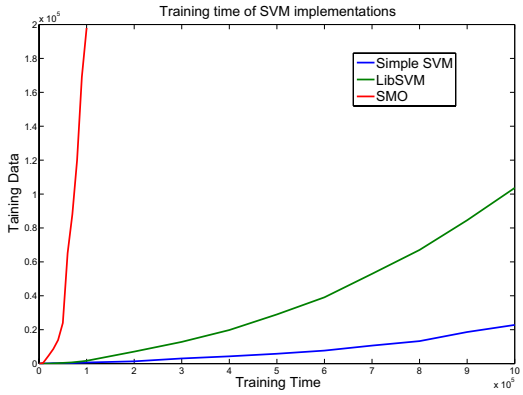

Fig. 1. Training time of SVM implementations

Many applications require the processing of huge data sets, however the training time is a serious problem in the processing of such data sets. Many experiments have shown that SVs constitute only a small fraction of the training data, the SVs contain the necessary features to obtain a discriminative hyperplane. Consequently, training data that are non-SVs and they not contribute to improve the accuracy or to find the hyperplane. A simple form to reduce the training time is to eliminate those data far away from hyperplane, while data close to the hyperplane are obtained and kept. Nevertheless, is not possible to know the hyperplane before to use SVM on a data set. Is clear that, we can use SVM on a very small data set and to use the obtained hyperplane in order to eliminate data points that are far from hyperplane and to obtain those data points close to the hyperplane from the total data set. From the previous analysis, we can do a modification to the SVM algorithm. The Fig 2 shows the proposed approach in six steps: 1) Random selection of a small data set 2) The first stage of SVM on the small data set 3) Getting SVs and Non SVs 4) Obtaining SVs from the original data set by Bayesian classification 5) Reduction of the original input data set 6) Second stage of SVM classification.

Before applying above approach on Intron/Exon classification, we need to encode DNA data. The encoding step recovers the most important features into DNA sequence.

A. Choice a representative data set

We use a random selection algorithm in order to obtain a small data set of DNA sequences from the original data set. Given a input data set (X, Y), where $X = \{x_1, x_2, ..., x_n\}$ is the total data set of encoded DNA sequences, x_i is a vector

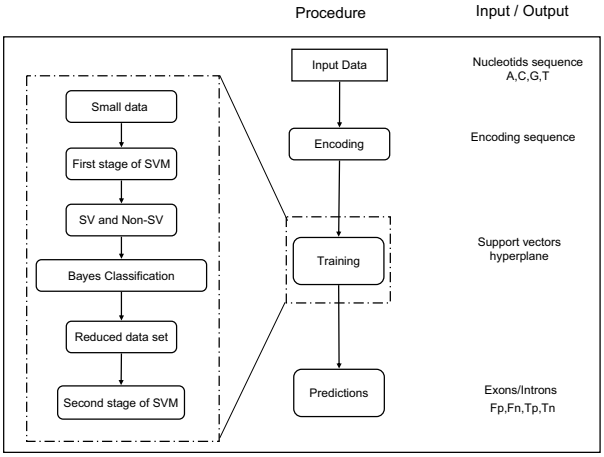

Fig. 2. The SVM algorithm proposed

that represent the encode sequence of p dimension, i.e., $x_i = (x_{i_1}...x_{ip})^T$, $Y = \{y_1, y_2, \ldots, y_n\}$ is the data set with labels, in this case the label $y_i \in \{-1, 1\}$. Our objective is obtain a small data set $C = \{x_i, y_i\}_{i=1}^{l}$, $l \ll n$ from X . After of random selection, the data set obtained is C. Where, the reduced data set C can be divided into two subsets, i.e.,

$$\begin{aligned} C^+ &= \{\cup C_i \mid y = +1\} \\ C^- &= \{\cup C_i \mid y = -1\} \end{aligned} \tag{4}$$

B. First stage of SVM

SVs are the most important data points in all the input data set and constitute a small fraction of the training examples. If most non-SVs can be removed the SVM training can be accelerated drastically. However, we need identify the features of SVs and non-SVs in order to recover all SVs and remove data points that are non-SVs. After of random selection, the algorithm obtains a small data set. Only these data points will be used as training data in the first stage of SVM classification, the hyperplane obtained is just a sketch of the original hyperplane. However, this hyperplane is very important in order to obtain the features of SVs and non-SVs and reduced the original input dataset. After of first classification stage the hyperplane obtained is given by:

$$\sum_{k \in V_1} y_k \alpha_{1,k}^* K(x_k, x) + b_1^* = 0 \tag{5}$$

where V_1 are the support vectors, i.e., the DNA sequences that contain the most important features in the first stage of SVM classification.

C. SVs and non-SVs

The previous classification hyperplane is not representative enough, however, it gives us a reference about data distribution. From previous step, we obtain data points that are support vectors and non-support vectors and their features that define them. The support vectors constitute the most important data points into the selected representative data set, we mark them

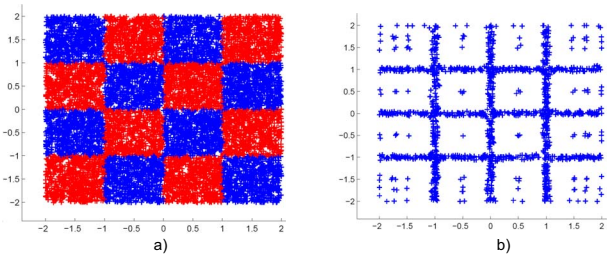

Fig. 3. Checkerboard dataset and support vectors

an extra label E^{+1}, The data points that are not support vectors are far from hyperplane and are not important, we mark them an extra label E^{-1}.

Formally the small data set is given by $C = \{x_i, y_i\}_{i=1}^{l}$ where $l \ll n$, The support vector in the first stage are given by

$$\{(x_1^*, y_1), (x_2^*, y_2), \ldots, (x_k^*, y_k)\}, 1, 2, \ldots, k \in V_1 \quad (6)$$

and the non support vectors is given by

$$\{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\}, 1, 2, \ldots, m \notin V_1 \quad (7)$$

where V_1 represent the support vectors in the first stage, is clear that $\{x_i, y_i\}_{i=1}^{l} = \{x_k, y_k\} \cup \{x_m, y_m\}$, so, to the data set $\{x_i, y_i\}_{i=1}^{k} \in V_1$ its new class is $\{x_i, y_i\}_{i=1}^{k} \in E^{+1}$ and $\{x_i, y_i\}_{i=1}^{m} \notin V_1$ its new class is E^{-1}. The Figure 3 a) shows the input data set and the Figure 3 b) shows SVs dataset which is obtained after on first stage of SVM.

D. Train the Bayes decision function and recover important data

In order to obtain all data points close to the first decision hyperplane, in this stage we train a bayes classifier with SVs and non-SVs, the new distribution class is expressed by:

$$(\mathbf{x}_{a1}, \mathbf{y}_{a1}), (\mathbf{x}_{a2}, \mathbf{y}_{a2}), \ldots, (\mathbf{x}_{an}, \mathbf{y}_{an}) \quad (8)$$

$i.e.$ $X = \{x_{ai}, y_{ai}\}_{i=1}^{q}$ where $q = k + m$, $x_{ai} \in R^d$ and $y_{ai} \in (+1, -1)$. Here the support vectors in the previous stage are expressed by $y_{ai} \in (+1)$ and the non support vectors are expressed as $y_{ai} \in (-1)$. Where $y_{ai} \in (+1)$ constitute the DNA sequences that are exons and contain the most important features and $y_{ai} \in (-1)$ constitute the DNA sequences that are introns or decoys.

To find a decision function that separate SVs from non-SVs we train a Bayes classifier, from it we obtain a decision function which separate the two class in the new distribution (data points close from hyperplane and data points far from hyperplane). Finally the decision function is obtained by the Bayesian classifier.

The decision function obtained with the Bayesian classifier recover all data points close from support vectors. The Figure 4 shows the SVs in blue and non-SVs in red obtained from first stage and Figure 4 b) shows the reduced dataset, the reduced dataset can be expressed as

$$\{x_{pi}, y_{pi}\} \cup (\{x_i, y_i\}_{i=1}^{k} \in V_1), \quad (9)$$

where x_{pi} are the data close from the decision hyperplane obtained in the first stage, which have a positive label, and V_1 are the support vector from the first stage.

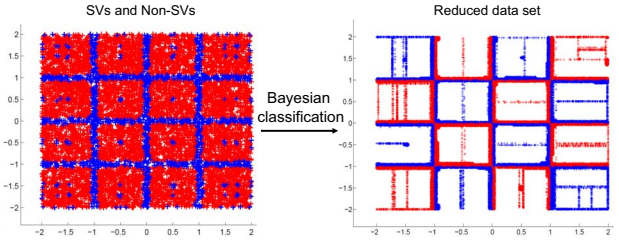

Fig. 4. SVs and reduced data set

E. The second stage SVM

From the recovery data points we can obtain a new training data set and use SVM again in order to obtain a final classification hyperplane, the final classification hyperplane can be written as:

$$\sum_{k \in V_2} y_k \alpha_{2,k}^* K(x_k, x) + b_2^* = 0 \quad (10)$$

where V_2 are the DNA sequences with most important features in the input data set. In general, the hyperplane (10) is very similar to (5). However the accuracy obtained with final hyperplane is better than the accuracy with the first hyperplane.

The recovery data are the most important data from optimization point of view. Therefore, the hyperplane (10) is an approximation of hyperplane obtained with SVM using SMO with the original data set.

IV. Experimental results

In this section, we show and analyze the experimental results obtained. First we define the methods and datasets used and finally we analyze the results obtained.

A. Methods

1) Model selection and evaluation: In our experiments we select the RBF function kernel (Radial Base Function), which is expressed as

$$K(x_i - x_j) = \exp(-\gamma \|x_i - x_j\|^2), \gamma > 0 \quad (11)$$

where γ is a parameter used with SVM. In order to obtain the best parameter we use a mesh search method, controlling the tradeoff between training error and complexity.

2) Performance measures: The performance of proposed SVM to calculate the proportion of noncoding nucleotides that have been correctly predicted as noncoding is evaluated as

$$S_n^{false} = \frac{T_N}{T_N + F_P} \quad (12)$$

S_n is the proportion of candidate sites in the testing data set that have been correctly predicted and it is expressed as

$$S_n = \frac{N_c}{N_t} \quad (13)$$

S_n^{true} is the proportion of coding nucleotides that have been correctly predicted as coding, i.e.,

$$S_n^{true} = \frac{T_p}{T_p + F_N} \tag{14}$$

where T_P is the number os sequences with real splice sites which are predicted to be true (true positives), T_N is the number of sequences without real splice sites which are predicted to be false (true negatives), F_P is the number of sequences without real splice sites which are predicted to be true (false positives) F_N is the number of sequences with real splice sites which are predicted to be false (false negatives), N_c is the number of exons that have been correctly predicted in the testing data set, and N_t is the total number of exons sites in the testing data set.

B. Datasets

1) Example 1: The First dataset used for this study is Primate splice-junction gene sequences (DNA) with associated imperfect domain theory taken from Genbank 64.1 (ftp site: *genbank.bio.net*). The DNA data set contains 3190 DNA sequences with 62 descriptors for each sequence, 767 exon/intron boundaries (referred to as EI sites), 768 intron/exon boundaries (referred to as IE sites) and 1655 neither. We use 80% of the input data to train the SVM and 20% to test. The SVM was trained and evaluated 20 times, the experimental results are shown in the Table I. The Table I shows the experimental results obtained with the proposed approach with the best accuracy obtained, worst accuracy obtained, the average (Av) and the standard deviation (SD) of the results obtained. However, in this case the dataset is very small the training time is almost the same with some SVM implementations like Simple-SVM, Libsvm, Sequential Minimal Optimization (SMO), but when the trainig data set is large the training time grows exponentially.

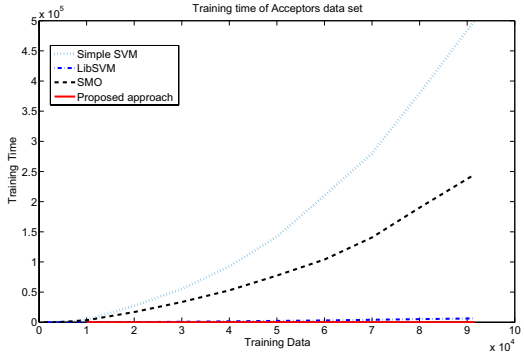

Fig. 5. Training time of SVM Implementations with Acceptors data set

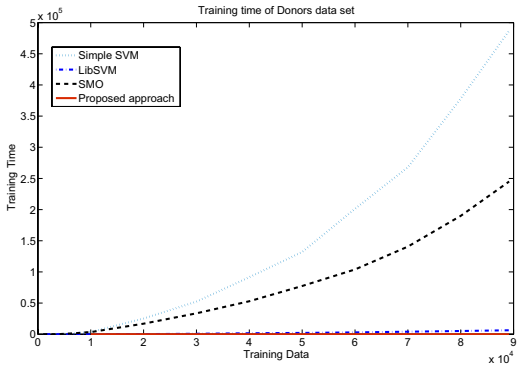

Fig. 6. Training time of SVM Implementations with Donors data set

Table I

Genbank 64.1 data set				
	Acc	S_n^{true}	S_n^{false}	S_n
Best	99.14	0.982	0.972	0.987
Worst	97.36	0.943	0.938	0.97
Av	98.02	0.979	0.98	0.98
SD	28.25	0.2729	0.256	0.242

Acc.-accuracy, Best.- the best result obtained Worst.-the best result obtained, Av.- Average SD.- standard deviation.

Furthermore, the comparison with other experimental results show that the accuracy obtained by proposed approach is comparable (99.14%) with other implementations such as decision tree ID3 (89.5%), Backpropagation algorithm (91.20%), and plausible Neural Networks (96.67%).

2) Example 2: The second example is acceptor/donor dataset which was obtained from http://www2.fml.tuebingen.mpg.de/raetsch/projects/. The dataset contains 91546 training data points and 75905 (2132 true sites) testing data points for acceptors and 89163 training data points and 73784 (2132 true sites) testing data points for donors. In this example we show the difference of training time between the proposed approach and other fast SVM implementations.

Table II

Acceptor data set				
	Proposed App		LIBSVM	
#	t	Acc	t	Acc
50000	8	95.9	2279	96.3
91546	72	98.2	6371	98.4

traininig data, t training time in seconds Acc accuracy.

We use the RBF function kernel. In order to select the parameters, we tried with different values of C and γ and use a grid search. For Acceptors data set, we found $\gamma = 0.027$, for Donors data set, we found $\gamma = 0.45$.

The Figure 5, shows the training time of some SVM implementations, in the Figure 1 the training of Simple SVM is very fast, because the training time is very small with 1 million of data points. However, each data point has just two features. In this case, to train the entire dataset (91546 data points) where each data point has 244 features the training time is very expensive. The Figure 5 shows that the time is about 500,000 seconds or 138 hours. SMO is a fast SVM implementation, however in this case, to train the entire data set take us about 245,000 seconds or 68 hours. The most competitive of these SVM implementations is LIBSVM, because the training time is very small (6371 seconds)in comparison with Simple SVM and SMO. However, the training time of proposed approach is very small (72 seconds), the Table II shows the training time and accuracy obtained with the proposed approach and Libsvm.

On the other hand, the Figure 6 shows the training time obtained with Donor data set, is clear that the training time is very similar as training time of Acceptor data set.

Table III

	Acceptor data set			Donor data set		
	Error	S_n^{true}	S_n^{false}	Error	S_n^{true}	S_n^{false}
SMO	1.6	0.84	0.98	1.8	0.82	0.98
SSVM	2.4	0.72	0.97	2.6	0.71	0.97
Libsvm	1.6	0.84	0.98	1.8	0.82	0.98
PA	1.8	0.837	0.98	1.9	0.81	0.98

PA.- proposed approach.

The Table III shows the results of S_n^{true}, S_n^{false} and the general error. In the Table SSVM is the Simple SVM implementation, and PA is the proposed approach. The S_n^{true} obtained with the proposed approach is better than the obtained with GenScan which was reported in [10].

V. Conclusion

In this paper, we presented a new SVM classification approach for large data sets. The algorithm uses a bayesian classifier in order to eliminate unnecessary training data from the input data set, therefore the disadvantage of SVM training with large data sets with nonlinear kernels can be overcome. We train, tune, and compare the proposed approach with other fast SVM implementations, experimental results on biological data sets show that the training time of large data sets is significatively reduced. Furthermore, the proposed approach has the ability of predict with a good accuracy whether a site is a true splice site or a decoy.

References

[1] Yiming Cheng, Robert M. Miura and Bin Tian *Prediction of mRNA polyadenylation sites by support vector machine*, Bioinformatics, Vol. 22 no. 19, pp 2320-2325, 2006.

[2] Christianini, N. and Shawe-Taylor, J. (2000). An introduction to support vector machines and other kernel-based learning methods. Cambridge University Press.

[3] Robertas Damaevicius *Splice Site Recognition in DNA Sequences Using K-mer Frequency Based Mapping for Support Vector Machine with Power Series Kernel*, International Conference on Complex, Intelligent and Software Intensive Systems, pp 687-692, 2008.

[4] Gideon Dror, Rotem Sorek and Ron Shamir *Accurate identification of alternatively spliced exons using support vector machine*, Bioinformatics, Vol. 21 no. 7, pp 897-901, 2005.

[5] Fickett, J. W. *Finding genes by computer: the state of the art*, Trends Genet, Vol. 12 no.8 pp 316320, 1996.

[6] Jones, D. and Watkins, C. *Comparing kernels using synthetic dna and genomic data*,. Technical report, University of London, UK, 2000.

[7] Liew, A. W.-C., Wu, Y., and Yan, H. *Selection of statistical features based on mutual information for classification of human coding and non-coding dna sequences*. Bioinformatics technologies, In ICPR 04: Proceedings of the Pattern Recognition, 17th International Conference on (ICPR04) Volume 3, pages 766769,Washington, DC, USA, 2004.

[8] Liew., A., Yan, H., and Yang, M. *Data mining for Bioinformatics*. Bioinformatics technologies, Springer, 2005.

[9] Naenna, T., Bress, R., and Embrechts, M. (2003). DNA classifications with self-organizing maps (SOMS). In Proceedings of the 2003 IEEE International Workshop on Soft Computing in Industrial Applications, pp 151154, Washington, DC, USA. IEEE Computer Society.

[10] Gunnar Rtsch and Sren Sonnenburg. Accurate Splice Site Detection for Caenorhabditis elegans.,in Kernel Methods in Computational Biology B. Schlkopf, K. Tsuda and J.-P. Vert Editors, MIT press.

[11] Vapnik, V. N. (1998). Statistical Learning Theory. Springer-Verlag New York, Inc., New York, NY, USA.

[12] Jing Xia, Doina Caragea and Susan Brown *Exploring Alternative Splicing Features Using Support Vector Machines*, Proceedings of the 2008 IEEE International Conference on Bioinformatics and Biomedicine, pp 231-238, 2008.

[13] Xiang H-F. Zhang, Katherine A. Heller, Ilana Hefter, Christina S. Leslie and Lawrence A. Chasin *Sequence Information for the Splicing of Human Pre-mRNA Identified by Support Vector Machine Classification*, Genome Research, Vol.13, pp. 2637-2650, 2003.