DNA Splice Site Detection: A comparison of specific and general methods

Won Kim[†], PhD and W. John Wilbur[†], MD, PhD

[†]National Center for Biotechnology Information (NCBI)
National Library of Medicine, Bethesda, MD 20894

In an era when whole organism genomes are being routinely sequenced, the problem of gene finding has become a key issue on the road to understanding. For eukaryotic organisms a large part of locating the genes is accomplished by predicting the likely location of splice sites on a DNA strand. This problem of splice site location has been approached using a number of machine learning or statistical methods tailored more or less specifically to the nature of the problem. Recently large margin classifiers and boosting methods have been found to give improvements over more traditional methods in a number of areas. Here we compare large margin classifiers (SVM and CMLS) and boosted decision trees with the three most common models used for splice site detection (WMM, WAM, and MDT). We find that the newer methods compare favorably in all cases and can yield significant improvement in some cases.

INTRODUCTION

Over the past decade as DNA sequence data on many organisms has begun to accumulate, increasing efforts have been made to mine this data for the genes that are responsible for phenotype and in many cases are the key to increased understanding of diseases. Genes are typically recognized in a two-step process. First different signal and content recognizers are crafted that predict the likely location of start sites, splice sites, stop sites, exonic regions, and intronic regions, etc. Then a generalized hidden Markov model is used to find the most likely gene locations among all the possibilities[1, 2]. As the field has developed it has become apparent that for eukaryotic organisms accurate splice site detection is critical to gene finding and that if this task can be successfully accomplished fairly simple methods may use the results for successful gene prediction[3-5]. Thus splice site recognition is a task important in its own right.

A number of different approaches have been tried for the detection of splice sites. Neural networks have been implemented[6-8]. Solovyev and Salamov[9] based linear discriminant functions on a number of different local sequence characteristics around potential splice sites. But much of what has been done has been based on three somewhat more general approaches to sequence classification. First, the position specific weight matrix approach (WMM) proposed by Staden[10] has been used[2, 4, 11]. This model assumes independence of bases along the sequence. The fact that there are dependencies between adjacent bases along the sequence has led to a second model by Zhang and Marr[12], the weight array model (WAM). This model or variants of it have found use in detecting both donor and acceptor sites[2, 11, 13, 14]. Finally, the maximal dependency tree (MDT) has been proposed by Burge and Karlin[2]. This model accounts for dependencies of higher order and not limited to adjacent bases. It also finds use in current systems[2, 3].

Our interest is in the models WMM, WAM, and MDT. The reasons are, first, the models are relatively simple and easily trained. Second, judging by the number of adherents, they have been the most successful approaches to splice site detection. Our goal is to compare these three models with each other and with approaches that have recently emerged in the field of machine learning as having superior performance. Wide margin classifiers in the form of the Support Vector Machine (SVM)[15] and the CMLS algorithm[16] have shown outstanding performance on text classification tasks. Support vector machines have also been used successfully for image recognition, protein homology detection, and gene expression array analysis[17]. Boosting is a technique that has recently been applied to text categorization tasks[18-22]. Here we apply the Adaboost algorithm with decision trees as the weak learner as described[23, 24].

ALGORITHMS

Weight matrix model (WMM) and naïve Bayes'. These two approaches are closely related. The weight matrix model considers for each position i in the sequence and each base s the probability $p_i(s)$ that base s occurs at position i in splice sites. Then a weight is calculated as a log

$$w_i(s) = \log p_i(s) \qquad (1)$$

The score for a given sequence is just the sum of the weights at each position in that sequence. This differs from naïve Bayes' in that naïve Bayes' also uses the probability that a base occurs in non-splice sites. Further Bayes' adds a weight for the absence of a base from a position as well as the weight for the base that is present. For more details on naïve Bayes' the reader may consult refs[22, 25]. We include naïve Bayes' because it is commonly used as a baseline with which to compare other methods.

Weight array model (WAM). The WAM accounts for dependencies between adjacent bases in the sequence. It adds to the weights for the WMM score the additional weights

$$w_i(s,s') = log\left[p_i(s,s') / \left(p_i(s)\, p_{i+1}(s') \right) \right] \quad (2)$$

that account for any interaction between base s at position i and base s' at position $i+1$. See ref[12].

Maximal Dependency Tree (MDT). The MDT begins by computing a consensus sequence for splice sites based on the training data and the positions to be considered. Let C_i be the consensus indicator function so that $C_i(sequ)$ is 1 or 0 depending on whether $sequ$ has the consensus base at position i or not. For any other position j let $X_j(sequ)$ equal the base that occurs in $sequ$ at position j. Define $\chi^2(C_i, X_j)$ to be the χ^2 statistic for C_i versus X_j and for any position i define

$$S_i = \sum_{j \neq i} \chi^2(C_i, X_j). \quad (3)$$

Then perform the first split at that position i with maximum value S_i. In the split send all sequences with the consensus base to one child node and all the other sequences to the other child node (a binary tree). At each child node carry on the process of finding the sums in (3) and splitting at the position with maximal sum. Do this recursively on child nodes until one of three conditions is satisfied: i) in this branch all available positions have already been used for splitting, ii) no significant dependencies are found between remaining positions, iii) at least one child node would have fewer than 175 sequences, which would be too few to train a WMM model. Once splitting is completed a WMM model is trained at each leaf node for scoring. See ref[2].

Support Vector Machine (SVM) and CMLS. Define the loss function

$$h(z) = \begin{cases} |1-z|, & z < 1 \\ 0, & 1 \leq z \end{cases} \quad (4)$$

Assume we are given training data $\{(\vec{x}_i, y_i)\}_i$ where y_i is 1 or -1 depending on whether the data point $\vec{x}_i$ is classified in the + or the – class. For SVM one seeks that vector $\vec{w}$ that minimizes

$$\sum_i h(y_i \vec{x}_i \cdot \vec{w} - 1) + \lambda \|\vec{w}\|^2. \quad (5)$$

We solve this problem using Platt's sequential minimal optimization algorithm[26, 27]. We have found $\lambda = 5$ to work well ($C = 0.1$ in the usual treatment[26, 28]). For the CMLS algorithm one seeks that vector $\vec{w}$ that minimizes

$$\sum_i \left[h(y_i \vec{x}_i \cdot \vec{w} - 1) \right]^2 + \lambda \|\vec{w}\|^2. \quad (6)$$

The method of solution is a sophisticated gradient descent as given by Zhang and Oles[16]. We have found $\lambda = 10^{-5}$ to work well for our data.

Boosted decision trees. We use an improved version of Adaboost[23] and boost binary decision trees where the splitting criterion is designed to minimize the error limit computed in Adaboost. We have examined trees of depths 1-5 and have found depth 3 to give the best results and that is what we report here. There is no set limit to the number of rounds of boosting to use in learning. However, in examining many rounds of boosting one generally sees improvement early and then a more or less stable performance with some oscillation, which appears due to noise. We apply an adaptation of the "one standard error" rule[29]. We calculate the best performance in the first 100 iterations and then calculate the standard deviation of the score over the next 1000 iterations. We then take the model that results from the fewest iterations and produces a score within this standard deviation of the best score in the first 100 iterations.

TESTING METHODOLOGY

Test Sequences. In order to study splice site recognition, a large set of true splice sites and a large set of non-splice sites were needed. These were constructed from data associated with Genbank human contig build 26.0. First, 14,047 human RefSeqs (mRNAs) were aligned with all human contigs and their complements (both strands). Those alignments that involved at least 1 kb or at least 50% of the bases for a RefSeq were considered valid. Any RefSeq that produced a unique valid alignment with 100% coverage of the sequence was marked as valid. From the unique alignments of such valid RefSeqs the splice sites defined by adjacent exons with a 100% match to the contig were extracted to form a set of donor sites and a set of acceptor sites. Additional selection was done to obtain a set of non-splice sites. Non-splice sites were randomly sampled uniformly over the length of contigs with the exclusion of any sites already identified in RefSeq alignments as possible true splice sites. Possible true splice sites were identified by a valid alignment in which adjacent exons achieved at least 95% identity with the contig. Following this selection process all sites were screened and any containing 'N' in the sequence were discarded. The result is a set of 35,136 donor sites, 35,097 acceptor sites, and 767,800 non-splice sites. The alignment program used to define these sets for testing purposes "genome_alignments.c" was written by Richa Agarwala (richa@helix.nih.gov) at NCBI.

Test Sets. All splice and non-splice sites consist of 50 bases, 25 bases before the proposed splice numbered negative and 25 bases after the proposed splice numbered positive. From this data we constructed

four test sets:

DN1: Donor set 1 consisting of all donor sequences and all random sequences cut down to [-3,6] or nine bases along the sequence.

DN2: Donor set 2 consisting of all donor sequences and all random sequences [-25,25] (no reduction in sequence length).

AC1: Acceptor set 1 consisting of all acceptor sequences and all random sequences cut down to [-20,3] or 23 bases along the sequence.

AC2: Acceptor set 2 consisting of all acceptor sequences and all random sequences [-25,25].

Sets DN1 and AC1 are defined to correspond to the sequence segments used by Burge and Karlin[2]. Sets DN2 and AC2 use more of the sequence and allow us to assess the potential utility of longer sequences in identifying splice sites.

For learning purposes one must identify the features to be used. We mostly take the standard approach where a feature is the occurrence of a base at a particular position in the sequence being considered. Thus for AC2 each sequence possesses fifty features. For this example there are 200 possible features coming from the standard four bases "ACGU" and fifty different locations along the sequence. However, the occurrence of other letters such as 'Y' and 'R' somewhat increases the number of possible features. Only for the case of DN1 we have also considered a pairs model in which features are pairs of positions with the bases at those positions. In this case the number of features is much greater.

In all cases there is no local weighting and a feature is either present or absent.

EVALUATION

Our results are mostly given as lists of precisions at the eleven recall values (0,10%, 20%, ...,90%,100%) and 11-point average precisions. These are calculated in the standard way[30]. All test sets have been randomly divided into thirds and all results are produced by three-way cross validation.

RESULTS

We have studied DN1 most intensively largely because it is the data type for which the MDT learning model was proposed. We applied all learning models to it in standard form and also Bayes, SVM, and CMLS to it with pairs of bases as features. Results for single base features are shown in tables 1 & 2.

A comparison of these results suggests that WMM and Bayes are equivalent in performance. They are both outperformed by WAM, MDT, SVM, and CMLS, which again are approximately equivalent.

Table 1. General machine learning techniques applied to DN1. Precisions at given recall levels.

Recall	Bayes (single)	SVM (single)	CMLS (single)	Boost iter=83
0.0	0.997	0.997	0.997	0.999
0.1	0.990	0.989	0.989	0.990
0.2	0.987	0.985	0.986	0.987
0.3	0.982	0.983	0.983	0.985
0.4	0.978	0.980	0.981	0.981
0.5	0.971	0.976	0.975	0.979
0.6	0.961	0.970	0.971	0.973
0.7	0.948	0.962	0.962	0.968
0.8	0.928	0.949	0.948	0.959
0.9	0.902	0.921	0.922	0.939
1.0	0.045	0.045	0.044	0.047
Avg	0.881	0.887	0.887	0.891

Table 2. Models historically proposed for splice site detection applied to DN1.

Recall	WMM	WAM	MDT
0.0	0.997	0.997	0.994
0.1	0.990	0.990	0.989
0.2	0.987	0.987	0.987
0.3	0.983	0.983	0.983
0.4	0.979	0.980	0.978
0.5	0.973	0.975	0.973
0.6	0.963	0.968	0.967
0.7	0.949	0.960	0.961
0.8	0.929	0.949	0.950
0.9	0.897	0.914	0.926
1.0	0.045	0.048	0.045
Avg.	0.881	0.886	0.887

Boosted trees at depth three outperform all of these methods. We looked at the pairs model and found 11-point averages for Bayes 0.884, SVM 0.890, and CMLS 0.892. Thus all three improve and the latter two methods give a result approximately as effective as boosted trees with single base features. Because the pairs formulation is expensive to create and to train on, we did not consider it further.

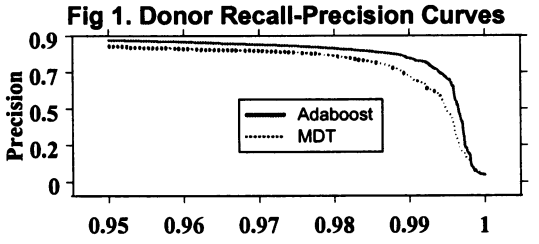
We processed DN2, AC1, and AC2 with the standard features and all the learning methods. The results are shown in Table 3 along with the results for DN1 for comparison. We also performed statistical tests between the best (Boost) and the next best (CMLS) methods for all four test sets using the bootstrap shift precision test[31]. All of the differences are statistically significant at the 5% level.

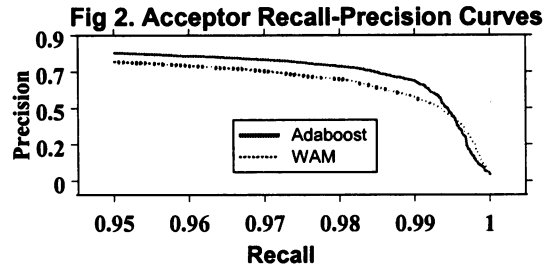
Table 3. 11-point average precisions for the various test sets and splice site detection algorithms.

	DN1	DN2	AC1	AC2
Bayes	0.881	0.878	0.864	0.867
SVM	0.887	0.893	0.869	0.878
CMLS	0.887	0.893	0.869	0.878
Boost	0.891	0.896	0.876	0.884
WMM	0.881	0.886	0.858	0.860
WAM	0.886	0.870	0.866	0.859
MDT	0.887	0.876	0.826	0.819

Several observations are pertinent here. First, including more sequence data does not seem to help but rather hurts performance of the WAM and MDT models. On the other hand the general learners are almost always able to improve their performance with more data. Second, boosting always gives the best result for each test set. In particular the best donor detection is with boosting on DN2 and the best acceptor detection is with boosting on AC2.

From a practical point of view one may expect to use any method of splice site detection to limit the number of gene models that need be considered in the process of gene identification. For this purpose one would not only like splice site detection to be sensitive, but also specific out to very high recall levels. To show the potential advantage of the general models for this purpose we have compared the best historical model result with the best general learner result for both donor and acceptor identification. These are contained in Figures 1 and 2. One sees an evident advantage in the Adaboost algorithm here.





DISCUSSION

One obvious question raised by the data in Table 3 is why WAM, and MDT degrade when the sequence segment considered is longer. While we cannot give a definitive answer to this question, these models were originally designed to give good performance in detecting donor sites based on short segments of DNA surrounding those sites [-4,+7] for WAM and [-3,+6] for MDT. Information more distant from a donor site is generally believed to be the result of a coding propensity or lack thereof and has been modeled by a different method involving higher order Markov processes[3, 4]. What we have observed here is consistent with this approach. It seems that SVM, CMLS, and the tree boosting approach do not have the limitations that are evident for WAM and MDT.

A second observation is that SVM and CMLS are almost the same in performance. One might have guessed that this would be the case based on the close relationship between the solutions they find (see (5) and (6)). In our experience CMLS usually has a slight edge (evident if more decimal places are included in the results). This was also noted on text classification tasks[16]. The really significant advantage of CMLS is speed. What took us days to do with SVM we were able to do in as many hours with CMLS.

Finally one cannot ignore the outstanding performance of Adaboost applied to decision trees. Boosted decision trees have given some of the best results reported in text classification[32, 33]. Decision trees allow one to take account of dependencies between features and boosting allows one to combine the results of multiple trees in an efficient manner to improve results. While the method is greedy in which trees it considers it has the advantage of avoiding the need to make all possible trees or look for all possible dependencies between features. While we used a splitting criterion proposed by Schapire and Singer[23] we also tried the CART and C4.5 splitting criteria and found essentially no difference in performance. Thus it is the boosting and not the particular splitting criterion that is producing the good performance.

Future work. In future work we hope to experiment with the general learning algorithms on even longer human sequence segments. It may be possible to push performance even higher and make screening for splice sites more efficient by this approach. In the end we hope that our results will lead to an enhancement of practical systems for gene identification. The general learning methods may also provide some advantage in dealing with the large number of different organisms with their own peculiarities that are responsible for more and more sequence data. One may hope to make use of this data without having to build specific models to fit each such organism.

Acknowledgements. The authors would like to thank Wratko Hlavina for producing the test sequence files and Richa Agarwala for the knowledge and programming that made this possible.

REFERENCES

1. Kulp D, Haussler D, Reese MG, Eeckman FH. A generalized hidden Markov model for the recognition of human genes in DNA. ISMB'96, 1996.
2. Burge C, Karlin S. Prediction of complete gene structures in human genomic DNA. J Mol Biol 1997;268(1):78-94.
3. Pertea M, Lin X, Salzberg SL. GeneSplicer: a new computational method for splice site prediction. Nucleic Acids Research 2001;29(5):1185-1190.
4. Parra G, Blanco E, Guigo R. GeneID in Drosophila. Genome Res 2000;10(4):511-5.
5. Guigo R. Computational gene identification: an open problem. Comput Chem 1997;21(4):215-22.
6. Reese MG, Eeckman FH, Kulp D, Haussler D. Improved splice site detection in Genie. J Comput Biol 1997;4(3):311-23.
7. Uberbacher EC, Mural RJ. Locating protein-coding regions in human DNA sequences by a multiple sensor-neural network approach. Proc Natl Acad Sci U S A 1991;88(24):11261-5.
8. Brunak S, Engelbrecht J, Knudsen S. Prediction of human mRNA donor and acceptor sites from the DNA sequence. J Mol Biol 1991;220(1):49-65.
9. Solovyev V, Salamov A. The Gene-Finder computer tools for analysis of human and model organisms genome sequences. Proc Int Conf Intell Syst Mol Biol 1997;5:294-302.
10. Staden R. Computer methods to locate signals in nucleic acid sequences. Nucleic Acids Research 1984;12(1):505-519.
11. Salzberg S, Chen X, Henderson J, Fasman K. Finding genes in DNA using decision trees and dynamic programming. Proc Int Conf Intell Syst Mol Biol 1996;4:201-10.
12. Zhang MQ, Marr TG. A weight array method for splicing signal analysis. Comput Appl Biosci 1993;9(5):499-509.
13. Zhang MQ. Identification of protein coding regions in the human genome by quadratic discriminant analysis. Proc Natl Acad Sci U S A 1997;94(2):565-8.
14. Salzberg S, Delcher AL, Fasman KH, Henderson J. A decision tree system for finding genes in DNA. J Comput Biol 1998;5(4):667-80.
15. Boser BE, Guyon IM, Vapnik VN. A training algorithm for optimal margin classifiers. In: Haussler D, ed. Annual ACM Workshop on Computational Learning Theory: ACM Press, 1992:144-152.
16. Zhang T, Oles FJ. Text categorization based on regularized linear classification methods. Information Retrieval 2001;4(1):5-31.
17. Cristianini N, Shawe-Taylor J. Support Vector Machines. Cambridge: Cambridge University Press, 2000.
18. Freund Y, Schapire RE. A decision-theoretic generalization of on-line learning and an application to boosting. Murray Hill, New Jersey: AT & T Research, 1995:
19. Freund Y, Schapire RE. Experiments with a New Boosting Algorithm. Machine Learning: Proceedings of the Thirteenth International Conference, 1996.
20. Freund Y, Schapire RE. A decision-theoretic generalization of on-line learning and an application to boosting. Journal or Computer and System Sciences 1997;55(1):119-139.
21. Elkan C. Boosting and naive Bayesian learning. La Jolla, California: University of California, San Diego, 1997:
22. Wilbur WJ. Boosting Naive Bayesian Learning on a Large Subset of MEDLINE. American Medical Informatics 2000 Annual Symposium. Los Angeles, CA: American Medical Informatics Association, 2000:918-922.
23. Schapire RE, Singer Y. Improved boosting algorithms using confidence-rated predictions. Machine Learning 1999;37(3):297-336.
24. Carreras X, Marquez L. Boosting trees for anti-spam email filtering. RANLP2001. Tzigov Chark, Bulgaria, 2001.
25. Mitchell TM. Machine Learning. Boston: WCB/McGraw-Hill, 1997.
26. Platt JC. Fast training of support vector machines using sequential minimal optimitzation. In: Scholkopf B, Burges CJC, Smola AJ, eds. Advances in Kernal Methods. Cambridge, Massachusetts: The MIT Press, 1999:185-208.
27. Platt J. How to implement SVMs. IEEE Intelligent Systems 1998(July/August):26-28.
28. Burges CJC. A tutorial on support vector machines for pattern recognition. Bell Laboratories, Lucent Technologies, 1999:
29. Hastie T, Tibshirani R, Friedman J. The Elements of Statistical Learning. New York: Springer, 2001.
30. Witten IH, Moffat A, Bell TC. Managing Gigabytes. (Second ed.) San Francisco: Morgan-Kaufmann Publishers, Inc., 1999.
31. Wilbur WJ. Nonparametric significance tests of retrieval performance comparisons. Journal of Information Science 1994;20(4):270-284.
32. Dumais S, Platt J, Heckerman D, Sahami M. Inductive learning algorithms and representations for text categorization. In: Gardarin G, French J, Pissinou N, Makki K, Bouganim L, eds. Proceedings of the 1998 ACM CIKM International Conference on Information and Knowledge Management. Bethesda, MD: ACM Press, 1998:148-155.
33. Apte C, Damerau F, Weiss S. Text mining with decision rules and decision trees. Proceedings of the Conference on Automated Learning and Discovery. CMU, 1998.