Splice Junction Recognition using Machine Learning Techniques

Ana C. Lorena[1], Gustavo E. A. P. A. Batista[1], André C. P. L. F. de Carvalho[1], and Maria C. Monard[1]

[1]Universidade de São Paulo, Instituto de Ciências Matemáticas e de Computação, Av. Trabalhador São-Carlense, 400 - Centro - Cx. Postal 668, São Carlos - São Paulo - Brasil
{aclorena, gbatista, andre, mcmonard}@icmc.sc.usp.br

Abstract. Since the start of the Human Genome Project, a large amount of sequence data has been generated. These data need to be analyzed. One of the main analysis to be carried out is the identification of regions of these sequences that correspond to genes. For such, one can search for particular signals associated with gene expression. Among the searched signals are the splice junctions. This recognition problem can be efficiently accomplished with the use of computational intelligent techniques. Many of the genetic databases, however, are characterized by the presence of high levels of noise, which can deteriorate the learning techniques' performance. This paper investigates the influence of noisy data in the performance of two different learning techniques (Decision Trees and Support Vector Machines), in the splice junction recognition problem. Results indicate that the elimination of noisy patterns from the datasets employed can improve Decision Trees' comprehensiveness and Support Vector Machines' performance.

1 Introduction

The Human Genome Project, whose main goal is the sequencing of all human genetic information (and also the genomes of other selected species), is generating a large amount of sequence data. One of the current issues in Bioinformatics is the recognition of patterns in these data. This work investigates the identification of genes in DNA sequences. This task can be solved by two different approaches: search by signal and search by content [6]. The first approach searches for signals associated with the gene expression process, like promoters and splice junction regions. The second approach looks for general patterns in the sequences that indicate the presence of a coding region.

This work investigates the use of two different Machine Learning (ML) techniques, Decision Trees [12] and Support Vector Machines [13], in the splice junction recognition problem. Since many of the genetic databases are characterized by the presence of high levels of noise [7], the influence of noisy patterns in the learning process is also evaluated. The less reliable patterns (possible noise) were eliminated using the Tomek links heuristic [3, 18].

This paper is organized as follows: Section 2 discusses the splice junction recognition problem. Section 3 describes the pre-processing technique used for noise elimination. Section 4 presents the learning techniques considered and related works. Sections 5 and 6 presents the experiments conducted and the results obtained, respectively. Section 7 concludes this paper.

2 Splice junction recognition

The main process that occurs in all organism's cells is the production of proteins. Proteins are essential components of all living beings, having structural and regulatory functions [10]. The protein coding process from the genetic sequence information is named *gene Expression*.

The gene expression is composed of two stages: *Transcription* and *Translation*. In the Transcription phase, a mRNA (*messenger Ribonucleic Acid*) is synthesized from a DNA (*Deoxyribonucleic Acid*). The protein coding is performed in the Translation stage, using the mRNA sequence as model.

There are differences in the processes described among organisms named eukaryotes and procaryotes. Eukaryotes genes are composed of alternated segments of *exons* and *introns*. Exons correspond to regions that are translated into proteins, and introns to regions that do not code for proteins. Translation in eukaryote organisms has then an additional step, where introns are spliced out from the mRNA molecule. *Splice junctions* are the boundary points where splicing occurs.

The splice junction recognition problem involves identifying if a specified sequence has a splice site or not, and its type (exon-intron or intron-exon). This paper is concerned with the identification of these regions. The final goal of this work is the recognition of genes from DNA sequences. Section 4 presents descriptions of some works devoted to the splice recognition task.

3 Data pre-processing

For evaluating the effect of noisy data in the performance of the learning techniques considered in this work, a pre-processing phase for the elimination of possible noises was applied. Patterns considered less reliable were then eliminated from the dataset employed. The detection of this noisy data was performed using a heuristic called Tomek links [18]. In order to illustrate how this heuristic works, consider the dataset from Fig. 1. The patterns from this dataset can be divided in three groups [3]:

- **Mislabeled samples:** data incorrectly classified. The (-) patterns in the left region of Fig. 1a are examples of mislabeled samples.
- **Borderlines:** patterns too close to the decision border induced for data classification. These examples are unreliable, since even a small quantity of noise can move them to the wrong side of the decision border.

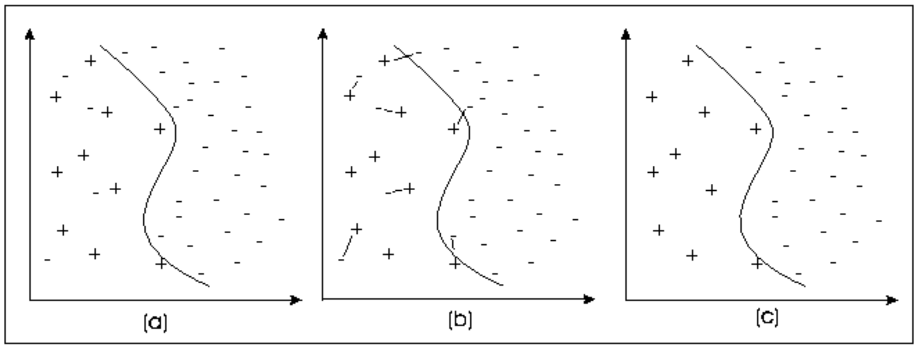

Fig. 1. Applying Tomek links to a dataset. Original data set (a), Tomek links identified (b), and Tomek links removed (c) [3].

– **Safe samples:** remaining patterns. These samples should compose the learning dataset.

The Tomek links heuristic allows the identification of mislabeled and borderline samples. Given two examples $\mathbf{x}$ and $\mathbf{y}$ from distinct classes, be $d(\mathbf{x}, \mathbf{y})$ the distance between these instances. A pair $(\mathbf{x}, \mathbf{y})$ is considered a Tomek link if there is not a case $\mathbf{z}$, such that $d(\mathbf{x}, \mathbf{z}) < d(\mathbf{x}, \mathbf{y})$ and $d(\mathbf{y}, \mathbf{z}) < d(\mathbf{y}, \mathbf{x})$ — Fig. 1. The computation of the distances d was performed with the *Value Difference Metric* (VDM) [3, 17].

4 Learning techniques

There are several supervised Machine Learning (ML) algorithms able of extracting concepts from data samples. Given a set of known examples, the learning algorithm induces a classifier C that should be able to predict the class of any pattern from the same domain where the learning process occurred. The class represents the item one wishes to make previsions about [2].

Among the ML works in splice junction recognition one can mention [14] and [19], in which propositional rules of the biological domain are used to initialize an Artificial Neural Network (ANN [8]). ANNs have achieved good performance in this task. The Statlog Project [11] also reports the use of various ML techniques for splice site identification. Another approach based on ANNs was proposed by Rampone [16], in which Boolean formulaes inferred from data were refined by an ANN.

This work investigates the use of two ML techniques following different approaches: Decision Trees (DTs) [12], a symbolic learning technique, and Support Vector Machines (SVMs) [13], a main representant of statistical learning.

DTs organize the information in a structure composed of nodes and ramifications [2]. The nodes represent tests applied to data or represent classes when the node is a leaf. The ramifications are possible results of the tests. The main advantage of DTs is the comprehensiveness of the induced rules in the classification process.

SVMs are learning techniques based on the Statistical Learning Theory, proposed by Vapnik and Chervonenkis [20]. They map the input data to an abstract space of high dimension, where the examples can be efficiently separated by an hyperplane. The SVM incorporates this concept with the use of functions named *Kernels*. These functions allow the access to complex spaces in a simplified and computational efficient way. The optimal hyperplane in this space is defined as the one that maximizes the separation margin between data belonging to different classes. The main advantages of SVMs are their precision and their robustness with high dimensional data. However, unlike DTs, SVMs classifiers are not directly interpretable.

5 Experiments

The splice junction dataset used in this work is composed of known primate DNA sequences, collected from Genbank 64.1 by Noordewier et al. [14], with their correspondent classes. This dataset is available in the UCI benchmark database [4]. The possible classifications for the DNA sequences are: EI, when the sequence has an *exon-intron* border, IE, for the *intron-exon* border, or N, if the sequence does not have a splice region. Table 1 summarizes this dataset, showing the total number of instances (♯ Instances), the number of continuous and nominal features present (♯ Features), the approximate class distribution (Class %), majority error (ME) and if there are missing values (MV).

The techniques described in the previous Section were applied in the generation of binary classifiers[1], because the Tomek links heuristic is simpler to perform when dealing with two classes. The splice junction recognition problem was then divided in the following manner: a classifier was induced to recognize sequences that have splice junctions (IE+EI) from the ones that do not have (N). If the presence of a splice junction is verified, a second classifier distinguishes if it is of the IE or EI type. For simplicity of reference, the IE+EI vs N subproblem will be referenced as “splice detection” (SD), and the IE vs EI one will be named “splice type identification” (STI).

Table 1. Dataset summary description

♯ Instances	♯ Features (nom., cont.)	Class	%	ME	MV
3190	60(60, 0)	IE	25%	50%	no
		EI	25%		
		N	50%		

[1] Classifiers involving only two classes.

The dataset was divided in 10 disjoint sets of approximately equal size, according to the 10-fold cross validation method [12]. Nine of these sets are used in the learning technique's training, and the resulting classifier is tested on the remaining set. This process is repeated ten times, making ten training and testing cycles. The total error is averaged over the errors obtained in each cycle. Since the original problem was divided in two (SD and STI), this procedure was performed for the total dataset (SD case) and for a subset with only the IE and EI examples (STI subproblem).

The pre-processing procedure was then applied to all training sets generated, eliminating instances considered mislabeld and borderlines. Approximately 6% of the instances were eliminated from the SD training sets, and approximately 5% were removed from the STI training sets. For simplicity, from now one we shall refer to the cleaned training datasets as "pre-processed datasets", and as "original datasets" the ones that did not pass by this pre-processing phase.

These datasets were then used in the generation of the classifiers, produced by DTs and SVMs. The DT induction was performed with the use of the C4.5 algorithm [15] and the SVMs with the assistance of the SVMTorch II tool [5].

Unlike DTs, SVMs require data to be in a numerical format. So, the splice junction dataset features had to be coded in a continuous format. The following coding was used: A = (1 0 0 0), C = (0 1 0 0), G = (0 0 1 0) and T = (0 0 0 1). This coding scheme ensures equidistance between all the possible feature values. Thus, for SVMs, each sample has 240 attributes (features).

6 Results

Decision Trees. An important propriety of DTs is their comprehensiveness, which in general is better for smaller trees. For the original dataset of the SD subproblem, the mean size of the induced trees was 230.6 ± 8.0 nodes. For the SD pre-processed training data, this size was of 209.0 ± 14.3 nodes, showing approximately 10% of reduction, or some comprehensiveness improvement. For the original STI training set, the mean DTs size was of 88.2 ± 9.6 nodes. After pre-processing, this size reduced to 81.8 ± 8.8 nodes (approximately 10% of reduction), also denoting a comprehensiveness gain. These measures refer to the trees induced after the pruning process[2]. Before pruning the mean size of the SD trees were of 690.6 ± 26.0 and 591.4 ± 25.0 nodes, respectively (15% of reduction with pre-processing). In the STI case, the mean sizes were of 295.4 ± 17.8 and 197.0 ± 16.9 nodes (33% of reduction). These facts show that the Tomek links heuristic really managed to clean the data from noisy examples.

Table 2 shows the overall performance of all induced trees. This table presents, for each experiment (ST and STI), the total misclassification mean error (Error), as well as for each class (IE+EI, N, IE and EI). It can be observed that errors on both datasets (on each experiment) are similar. However, comprehensiveness always improves, although this improvement is only statistically significant for DTs before the pruning process.

[2] Pruning is a technique that minimizes the noise influence in the classifier induction.

Table 2. DTs' performance

	SD experiment			STI experiment		
Dataset	Error	IE+EI Error	N Error	Error	IE Error	EI Error
Original	4.4 ± 0.6	1.9 ± 0.6	6.7 ± 0.9	4.3 ± 1.5	5.3 ± 2.3	3.3 ± 2.8
Pre-processed	4.5 ± 0.9	1.9 ± 1.0	6.8 ± 1.0	4.5 ± 1.6	5.3 ± 2.4	3.7 ± 2.4

Support Vector Machines. In the SVM experiments, several types of Kernel functions were tested. The functions considered were: Polynomial of different degrees (1 to 5), Gaussians with varying standard deviation values (0.01, 0.1, 1, 10, 50, 100) and a Sigmoid. Training was performed until a training error rate inferior to 0.01 was reached. Besides the Kernel parameters mentioned, other parameters were set to the default values of SVMTorch II [5].

In the SD experiments, the best results for the original dataset were achieved by the Gaussian Kernel with a standard deviation of 5, and with a Polynomial Kernel of third degree for the pre-processed dataset. In the STI case, the best Kernel for both datasets was the Polynomial of fifth degree.

For SVMs an important measure is the final number of *support vectors* (SVs) of the generated model. The SVs are the most representative training data for the SVM classificatory task. They are the patterns closest to the optimal hyperplane. The equation of this hyperplane is defined using these samples. Thus, the presence of noisy patterns can influence the determination of this hyperplane equation. In the original SD experiment, the final number of SVs was 1696.4 ± 8.3, against 1529.4 ± 16.5 for the pre-processed data (10% of reduction). In the STI experiment, the results were of 1175.3 ± 6.1 and 1109.9 ± 7.4 SVs (6% of reduction), respectively. It was noticed too that the training time reduced with data pre-processing (approximately 7% in the SD experiment and 15% in the STI case).

Similarly to Table 2, Table 3 shows the overall performance of the SVMs classifiers during test. It can be verified that the overall performance of the SVMs classifiers was maintained with data pre-processing (while gains were achieved with reductions in the training time and number of SVs). For the SD experiment, in particular, the results were better with data pre-processing, at a 95% confidence level.

Table 3. SVMs' performance

	SD experiment			STI experiment		
Dataset	Error	IE+EI Error	N Error	Error	IE Error	EI Error
Original	3.6 ± 0.9	1.3 ± 0.8	2.2 ± 0.6	1.9 ± 1.1	1.7 ± 1.1	2.1 ± 2.3
Pre-processed	2.9 ± 0.8	1.0 ± 0.8	2.0 ± 0.4	2.1 ± 1.2	1.2 ± 1.2	0.9 ± 0.5

Performance comparison. Using the t-test to paired data [9] over the misclassification error of the different classifiers induced, a performance comparison was performed among the leaning techniques considered. Through this test, it was verified that SVMs outperforms DTs in all cases, with a 95% confidence level. This is due to their ability to deal with high dimensional patterns, as those in the splice junction dataset. DTs, like other symbolic ML techniques, are not well suited for this kind of data.

The UCI benchmark [4] also provides results from previous works for the same data (ex. [14, 19]). However, these works did not divide the problem in two binary subproblems, difficulting the comparison with the results presented in this paper. The unique comparable result is N Error, in which [19] obtained an average error rate of 4.6 %. The SVMs classifiers modelled in this work achieved better results (average error of 2.0 %). It should be pointed out, however, that [19] used only 1000 examples randomly chosen from the complete dataset, while in the present work all 3190 examples were employed.

7 Conclusions

This paper presented a study of the influence of noisy data in the performance of two different learning techniques: Decision Trees and Support Vector Machines. The application problem was the recognition of splice junctions in DNA sequences. Many genetic databases are characterized by the presence of noise, justifying the choice of this application domain. For this evaluation, a pre-processing for the elimination of possible noisy examples was applied to the datasets employed. The application of this phase lead to improvements in the learning methods performance.

In the DTs case, gains were mainly noticed on the comprehensiveness of the induced tree. For SVMs, significant reductions were obtained on training time and on in the final number of support vectors, patterns that determine the optimal hyperplane equation for data separation. This suggests that the optimal hyperplane is no more oriented by unreliable cases.

The cost of the pre-processing phase should also be considered. For n examples having m features, this phase has $O(m \cdot n^2)$ complexity. Thus, it is a costly process. It should be observed, however, that this phase is applied only once, independent of the number of ML techniques employed afterwards. In spite of the elimination of noisy patterns by the Tomek links heuristic, some of the examples removed may be relevant to the classifier induction process. To avoid this situation, the tuning of this algorithm must be carefully performed.

As possible future work, the Tomek links heuristic will be applied to other datasets from the Molecular Biology field. Further experiments should also be performed to adjust the ML algorithms parameters.

Finally, it can be stated that the splice junction dataset used in the experiments conducted does not have a high level of noise. This may be due to previous pre-processing applied to this data, since it has been used in several works (ex. [14, 19]). Despite this fact, the results obtained are encouraging.

Acknoledgements

The authors would like to thank CNPq and Fapesp, Brazilian research support agencies, for the financial support provided.

References

1. Baldi, P., Brunak, S.: Bioinformatics - The Machine Learning Approach. The MIT Press (1998)
2. Baranauskas, J. A., Monard, M. C.: Reviewing some Machine Learning Concepts and Methods. Technical Report 102, Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, Brazil, `ftp://ftp.icmc.sc.usp.br/pub/BIBLIOTECA/rel_tec/RT_102.ps.zip` (2000)
3. Batista, G., Carvalho, A., Monard, M. C.: Applying one-sided selection to unbalanced datasets. Mexican International Conference on Artificial Intelligence (MICAI) Lecture Notes in Artificial Intelligence, Vol. 1793. Springer-Verlag (2000) 315–325
4. Blake, C. L., Merz, C. J.: UCI repository of machine learning databases. `http://www.ics.uci.edu/~mlearn/` (1998)
5. Collobert, R., Bengio, S.: SVMTorch: Support vector machines for large scale regression problems. Journal of Machine Learning Research, Vol. 1 (2001) 143–160
6. Cravem, M. W., Shavlik, J. W.: Machine Learning Approaches to Gene Recognition. IEEE Expert, Vol. 9, No. 2. IEEE Computer Society Press (1994) 2–10
7. Cristianini, N.: Support vector and kernel methods for bioinformatics. Pacific Symposium on Biocomputing Tutorial, Kaua'I Marriott, Kaua'I, `http://www.support-vector.net/PSB2002.pdf` (2002)
8. Haykin, S.: Neural Networks - a compreensive foundation. Prentice Hall (1999)
9. Johnson, R. A.: Miller and Freund's Probability and Statistics for engineers. Prentice Hall (2000)
10. Lewis, R.: Human Genetics - Concepts and Applications. McGraw Hill (2001)
11. Michie, D., Spiegelhalter, D. J., Taylor, C. C.: Machine Learning, Neural and Statistical Classification. Ellis Horwood (1994)
12. Mitchell, T.: Machine Learning. McGraw Hill (1997)
13. Müller, K. R., Mika, S., Rätsch, G., Tsuda, K., Schölkopf, B.: An introduction to kernel-based learning algorithms. IEEE Transactions on Neural Networks, Vol. 12, No. 2. IEEE Computer Society Press (2001) 181–201
14. Noordewier, M. O., Towell, G. G., Shavlik, J. W.: Training Knowledge-Based Neural Networks to Recognize Genes in DNA Sequences. Advances in Neural Information Processing Systems, Vol. 3, Morgan Kaufmann (1991) 530–536
15. Quilan, J. R.: C4.5 Programs for Machine Learning. Morgan Kaufmann, CA (1988)
16. Rampone, S.: Recognition of splice junctions on DNA sequences by BRAIN learning algorithm. Bioinformatics, Vol. 14, No. 8, Oxford University Press (1998) 676–684
17. Stanfill, C., Waltz, D.: Toward Memory-Based Reasoning. Communications of the ACM, Vol. 29, No. 12 (1986) 1213–1228
18. Tomek, I.: Two Modifications of CNN. IEEE Transactions on Systems Man and Communications, SMC-6. IEEE Computer Society Press (1976) 769–772
19. Towell, G. G.: Symbolic Knowledge and Neural Networks: Insertion, Refinement, and Extraction. PhD Thesis, University of Wisconsin - Madison (1991)
20. Vapnik, V. N., Chervonenkis, A.: On the Uniform Convergence of Relative Frequencies of Events to their Probabilities. Theory of Probability and Its Applications, No. 16 (1968) 262–280