Identifying splice-junction sequences by hierarchical multiclassifier

Alessandra Lumini, Loris Nanni *

DEIS, IEIIT – CNR, Università di Bologna, Viale Risorgimento 2, 40136 Bologna, Italy

Received 13 September 2004; received in revised form 27 April 2005
Available online 2 May 2006
Communicated by L. Goldfarb

Abstract

Splice sites are locations in DNA which separate protein-coding regions (exons) from noncoding regions (introns). Recently, several works have approached the splice-junction problem by applying methods from the field of machine learning techniques. In this work, we propose a hierarchical multiclassifier (HM) architecture, whose results show a drastically error reduction with respect to the performance of methods proposed in the literature.

Keywords: Hierarchical multiclassifier; Machine learning; Splice-junction

1. Introduction

Splice sites are locations in DNA which separate protein-coding regions (exons) from noncoding regions (introns). Accurate splice site detectors thus form important components of computational gene finders.

Splice junctions (Maisheng and Wang, 1998) are points on a DNA sequence at which ‘superfluous’ DNA is removed during the process of protein creation in higher organism. The problem posed is to recognize, given a sequence of DNA, the boundaries between exons (the part of DNA sequence retained after splicing) and introns (the parts of the DNA sequence that are spliced out).

The computational identification of genes and coding regions in DNA sequences is a major goal for molecular biology. The reliable detection of genes and protein coding regions in DNA sequences is critical for the success of the computational gene discovery (Noordewier et al., 1991). According to the central dogma of molecular biology, DNA codes for the production of messenger RNA (mRNA) during the transcription process. Messenger RNA carries coded information to ribosomes. The ribosomes “read” this information, and use it for protein synthesis during the translation process.

This problem can also be posed as to find a classification rule: given a position in the middle of a window of DNA sequence elements (nucleotides), decide whether this is an ‘intron–exon’ boundary (‘IE’), ‘exon–intron’ boundary (‘EI’), or ‘neither’ (‘N’). Many works (Maisheng and Wang, 1998; Thivierge and Schultz, 2002; Rampone, 1998) show that machine learning techniques perform better than classification based on canonical pattern matching. In (Noordewier et al., 1991) the application of a knowledge based artificial neural network (KBANN) approach is described. KBANN is a hybrid method that combines explanation-based and empirical learning methods.

In (Pertea et al., 2001) a probabilistic model was presented, this method estimate position-specific probabilities of splice sites by computing likelihoods of candidates of signal sequence.

At present, the most successful splice site prediction techniques are (Ho and Rajapakse, 2003; Rampone, 1998). In (Rampone, 1998) the authors present BRAIN, an algorithm that infers Boolean formulae from examples and considers the splicing rules as disjunctive normal form (DNF) formulae. The result is refined by means of a neural...
network and combined with a discriminant analysis procedure. In (Ho and Rajapakse, 2003) the authors present a hybrid approach consisting of two first- and two-second-order Markov chain models at the first stage and of a three-layer neural network at the second stage.

In this work we show that the splice-junction problem can be efficiently solved using a hierarchical multiclassifier (HM) architecture. The proposed method is hierarchical in view of the fact that at each step a classifier is employed which processes some patterns and rejects some others: only the patterns rejected at each step are passed to the next classifier. The system is a multiclassifier since it integrates several data-driven models for the same problem. In the first step of our HM, we adopt a subspace classifier (Oja, 1983), patterns rejected from this step are classified using an edit distance classifier (Levenshtein, 1965), finally remaining patterns are classified using a combination of linear support vector machines (Duda et al., 2001).

There are many ways to build a system of hierarchical multiclassifiers. A typical example is a system where the set of classes is organized in a hierarchical way and different classifiers are trained in order to drive the input pattern towards the most specific classifier to be used for the final classification (Simon et al., 1999). Another example is a system where only a subset of the input features are given as input to a first-level classifier and if the final classification cannot be performed with a sufficient level of confidence the remaining features are used at further classification stages (Rokui and Shimodaira, 1999).

Another approach where a hierarchical system is used in order to improve performance in term of both accuracy and response requires that at each step a classifier with rejection is used to classify patterns with high confidence. Rejected patterns are forwarded to a more complex and usually slower classifier for classification (or further rejection to the next stage). Typically, the system as a whole holds better classification performance with respect to the single classifiers at the cost of a slower response time (Giusti et al., 2002).

In this work, we define a four-stage pattern recognition scheme consisting of a hierarchy made up by some classifiers with rejection. It has been proven (Giusti et al., 2002) that the probability of error for a hierarchical system, given a rejection threshold for the classifiers at each levels, can be expressed as the sum of the optimal Bayes error and the error rate of the different classifiers (related to the patterns effectively classified – i.e. not rejected). This result means that in principle, we do not loose the possibility to reach the optimal Bayes error even if all the stand-alone classifiers are not optimal.

The rest of the paper is organized as follows: in Section 2, we review and compare several machine learning methods tested in this work; in Section 3, the new hierarchical multiclassifier is detailed; in Section 4, the results of the experiments are discussed; and finally, in Section 5, we draw some conclusions.

2. Machine learning approaches

In the following a brief description of the feature extraction methodologies, feature transformations, classifiers and ensemble methods combined and tested in this work is given.

2.1. Feature extraction

Feature extraction is a process that extracts a set of new features from the original pattern representation through some functional mapping. In this case a pattern is a DNA sequence of length $m = 60$ over an alphabet of eight symbolic variables: the four kinds of nucleotides (‘A’, ‘G’, ‘T’, ‘C’) and four characters (‘D’, ‘N’, ‘S’, ‘R’) which indicate ambiguity among the first four characters according to the following rules.

- ‘D’ = ‘A’, ‘G’ or ‘T’
- ‘S’ = ‘C’ or ‘G’
- ‘R’ = ‘A’ or ‘G’

In the orthonormal encoding the symbolic variables representing the nucleotides are replaced by eight binary indicator variables: each symbolic variable is then represented by an 8 bits vector with 7 bits set to zero and one bit set to one, so that each vector is orthogonal to all others (see Table 1).

Feature ranking is a process for sorting the features and retaining only a subset of the initial features. In this paper we use a simple feature ranking which sorts the features that maximize the distance between the centroids of the different classes of patterns (Duda et al., 2001). For each feature $f(1, \ldots, n)$ we compute the scalar Mahalanobis distance $d_{M, f} = d_{M}(\mu_{c, f}, S_f)$ between the mean $\mu_{c, f}$ of the training patterns of each class $c$ from a total of $N_c$ classes (in our case $N_c = 3$ : ‘IE’, ‘EI’, ‘N’), and the set of all the

<table>
<thead>
<tr>
<th>Table 1</th>
<th>The orthonormal sparse encoding</th>
</tr>
</thead>
<tbody>
<tr>
<td>Characters</td>
<td>8 bit vector</td>
</tr>
<tr>
<td>‘A’</td>
<td>10000000</td>
</tr>
<tr>
<td>‘G’</td>
<td>01000000</td>
</tr>
<tr>
<td>‘T’</td>
<td>00100000</td>
</tr>
<tr>
<td>‘C’</td>
<td>00010000</td>
</tr>
<tr>
<td>‘D’</td>
<td>00001000</td>
</tr>
<tr>
<td>‘N’</td>
<td>00000100</td>
</tr>
<tr>
<td>‘S’</td>
<td>00000010</td>
</tr>
<tr>
<td>‘R’</td>
<td>00000001</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 2</th>
<th>Codons that must not be contained in class ‘neither’</th>
</tr>
</thead>
<tbody>
<tr>
<td>TGG</td>
<td>GGG</td>
</tr>
</tbody>
</table>
training patterns \( S \). The Mahalanobis distance between a pattern \( x \) (\( \mu_x \) in our case) and a set \( S \) of patterns with mean \( \mu_S \) and covariance matrix \( C_S \) is given by

\[
dM(x, S) = (x - \mu_S)^T C_S^{-1} (x - \mu_S)
\]  

(1)

Features are then ranked according to the following inter-user class separability measure:

\[
\text{DistF}(f) = \sum_{i=1}^{N} \sum_{j=1}^{N} |dM_{i,j} + dM_{j,i}|
\]  

(2)

In this paper, we retain only the features corresponding to the 100 higher values of DistF.

Feature transformation is a process through which a new set of features is created from the original features through some functional mapping. In this work we adopt the Karhunen Loeve (KL) transform (Duda et al., 2001): this transform projects \( n \)-dimensional data onto a lower-dimensional subspace in a way that is optimal in a sum-squared sense. It has known that KL is the best linear transform for dimensionality reduction.

Given a \( n \)-dimensional data points \( x \), the goal of the KL transform is to reduce the dimensionality of the observed vector, by finding \( k \) principal axes, denoted as principal components, which are given by the eigenvectors associated with the \( k \) largest eigenvalues \( \Phi = [\phi_1, \phi_2, \ldots, \phi_k] \) of the covariance matrix \( C_S \) of the training set.

A map between the original space and the reduced eigenspace is performed by means of the operator of projection, where the projection \( z \) of a vector \( x \in \mathbb{R}^n \) into the reduced space \( \mathbb{R}^k \) is

\[
z = \Phi^T (x - \mu_S)
\]  

(3)

where \( \mu_S \) is the mean of the training set \( S \).

2.2. Classification

A classifier is a component that uses the feature vector provided by the feature extraction or transformation to assign a pattern to a class. In our hierarchical multiclassifier the following classifiers are used:

- Linear support vector machines.
- SubSpace classifier (Oja, 1983).
- Edit distance classifier (EDC).

The goal of this two-class classifier (LSVM) is to establish the equation of a hyperplane that divides the training set leaving all the points of the same class on the same side, while maximizing the distance between the two classes and the hyperplane. We briefly recall the basic notions of the theory of LSVMs (Cristianini and Shawe-Taylor, 2000). Each pattern \( x_i \) belongs to either of two classes and thus is given a label \( y_i \). The goal is to establish the equation of a hyperplane that divides the training set \( S \), leaving all the points of the same class on the same side, while maximizing the distance between the two classes and the hyperplane.

The set \( S \) is linearly separable if there exists \( w \) such that:

\[
y_i(w \cdot x_i + b) \geq 1 \quad i = 1, 2, \ldots, \#S
\]  

(4)

Given a linearly separable set \( S \), the optimal separating hyperplane is the separating hyperplane for which the distance of the closest point of \( S \) is maximum. Since the distance of the closest point equals \(|w|/\|w\|\), the optimal separating hyperplane can be regarded as the solution of the problem:

\[
\text{Minimize} \quad \frac{1}{2} w \cdot w \quad \text{subject to:} \quad y_i(w \cdot x_i + b) \geq 1 \\
i = 1, 2, \ldots, \#S
\]  

(5)

It is important to note that the parameter \( b \) enters in the constraints but not in the function to be minimized. The quantity \(|w|/\|w\|\), the lower bound of the minimum distance between points of different classes, is named “margin”.

3. Hierarchical multiclassifier

We develop a four-phases hierarchical structure for the classification of patterns, where only the patterns rejected at one step are evaluated by the classifier of the next step.
To determine an order for the classifiers we adopt the following greedy rule: we rank the classifiers by their precision in the classification of the patterns with a high confidence value. The first classifier is a subspace classifier due to its better performance for patterns classified with high confidence value with respect to LSVM (see the Section 4.3). As to the edit distance classifier combined with a syntactic rule employed in the second phase of our hierarchical multiclassifier, it is useful mainly to classify a limited number of patterns with a very low error rate (see Table 6). In phase 3 we found the two most likely classes for each pattern, and we discriminate between those three couples of class by ad hoc LSVMs coupled with some rules for splice-junction determination (Thivierge and Schultz, 2002). Considering that a LSVM classifier gives a higher performance than a Subspace classifier on patterns classified with low confidence, we adopt this classifier in the last phase where no rejection is allowed.

The algorithm for the definition of the structure is detailed in Fig. 2:

- First phase is composed by: **Subspace classifier**. We classify only the patterns whose similarity is larger than a fixed threshold $t_1$ (we use $t_1 = 0.125$ in our experiments), while reject the others. The confidence value adopted in our classification approach is given by the difference between the two highest similarity values given by our classifier.

- Second phase is composed by: **Edit distance and rule**. For these particular problems the class ‘neither’ is shift invariant, the other classes are not. Shift invariance means that the category remains unchanged if we shift left or right the pattern of one position. For instance, the sequences “CGT...AAT” belong to class ‘neither’, which means that all its subsequences belong to the same category. The other categories, however, are not shift invariant because we believe the splice sites to occur at one specificity site and not in nearby sites.

- Third phase is composed by: **LSVM 2-Most**. We trained a LSVM classifier (using the 1-vs-1 approach (Duda et al., 2001) to solve this multi-class problem), using “orthonormal encoding” as feature extractor and KL as feature transformation (reducing the feature vector to 50 dimensions), and storing, for each pattern, the 2-most likely classes (obtained using the confidence value of the LSVM trained in this step). Then we discriminate between the two most likely classes of each pattern by
  - **LSVM1** (when 2-most likely classes are ‘EI’ and ‘N’).
  - **LSVM 2-most**.

The edit distance classifier gives good performance for patterns belonging to the shift invariant class, while it is not reliable when assigns a pattern to the shift variant class. For example, if a new pattern is near (with respect the edit distance) to a pattern of the shift invariant class we can reasonable assume that it belongs to the same class, on the contrary if the new pattern is near to a pattern of the shift variant class, we cannot make any assumption with high degree of certainly. Starting from this consideration, we design a classifier that assign to the class ‘neither’ each pattern classified as ‘neither’ by EDC, while reject the others.

The dataset documentation indicates that much better performance is generally observed if attributes closest to the junction are used. In our case, this means that we use, in this step, only feature between $@-5$ a $@+5$. Where $@-5$ indicates 5 symbols before the potential splice point. A possible method to reduce the error of this step is to reject the patterns that satisfy this rule:

“Patterns classified as ‘N’ must not contain the codons (sequences of three bases) shown in Table 2 in position $@-3 @-2 @-1 @+1$”

From a statistical study on the training set, we noted that the codons listed above are rarely contained in the ‘neither’ class.

- **LSVM 2-Most**.

We train a LSVM classifier (using the 1-vs-1 approach (Duda et al., 2001) to solve this multi-class problem), using “orthonormal encoding” as feature extractor and KL as feature transformation (reducing the feature vector to 50 dimensions), and storing, for each pattern, the 2-most likely classes (obtained using the confidence value of the LSVM trained in this step). Then we discriminate between the two most classes of each pattern by
  - **LSVM1** (when 2-most likely classes are ‘EI’ and ‘N’).

We train a LSVM classifier on the feature vectors...
reduced to 50 dimensions by KL. We classify only the patterns whose similarity is larger than a fixed threshold $t_2$ ($t_2 = 1$ in our experiments).

- **LSVM2** (when 2-most likely classes are ‘EI’ and ‘IE’). We train a LSVM classifier as in the step LSVM1. In this step we classify only the patterns whose similarity is larger than a fixed threshold $t_2$. Moreover, the following rules partially matching with those shown in (Thivierge and Schultz, 2002) are applied:
  - “Patterns classified as ‘IE’ but with a similarity lower than half of the threshold $t_2$ are classified only if between $@ - 16$ and $@ - 5$ there are more than 6 ‘C’ or ‘T’”
  - “All the patterns containing the codons ‘TAA’ or ‘TAG’ in position $@ + 1 @ + 2 @ + 3$ are classified as ‘IE’.”

- **LSVM3** (when 2-most likely classes are ‘IE’ and ‘N’). We train a LSVM classifier as in the step LSVM1. In this step we classify only the patterns whose similarity is larger than a fixed threshold $t_2$. Moreover, the following rules partially matching with those shown in (Thivierge and Schultz, 2002) are applied:
  - “Patterns classified as ‘IE’ but with a similarity lower than half of the previous threshold $t_2$ are classified only if between $@ - 16$ and $@ - 5$ there are at least 5 ‘C’”
  - “Patterns classified as ‘N’ but with a similarity lower than the prefixed threshold $t_2 = 0.35$ in our experiments are classified only if between $@ - 16$ and $@ - 5$ there are less than 3 ‘C’”
  - “All patterns containing the codons ‘TAA’ or ‘TAG’ in position $@ + 1 @ + 2 @ + 3$ are classified as ‘IE’”.

- Fourth phase is composed by **LSVM**. The patterns, rejected by the previous steps, are finally classified by a LSVM classifier trained on the feature vectors reduced to 50 dimensions by KL.

4. Experimental results

This method has been tested on the “IP data set”. “IP data set” is a benchmark set of human splice site data from the UCI Machine Learning Repository (Blake and Merz, 1998). It contains 765 acceptor sites, 767 donor sites and 1654 decoys. The task is a donor/acceptor classification, given a position in the middle of a window of 60 DNA letters as input. There are no missing attribute values. Each instance is of one of 3 classes (‘EI’, ‘IE’ or ‘N’). Each of the 60 fields is filled by one of the eight characters listed in Section 2.

On this dataset, we performed 10 tests, each time randomly resampling learning, and test sets (containing respectively 2000, and 1186 patterns) as in (Rampone, 1998), but maintaining the distribution of the patterns in the three classes. The results reported refer to the average classification accuracy achieved throughout the 10 experiments.

We use as indicators of the predictive performance the error rate and the correlation coefficient (CC), a measure that takes the relationship between correctly predicted positives and negatives as well as false positives and negatives:

$$CC_j = \frac{P_j N_j - P_N N_j}{\sqrt{(N_j + N_N)(N_j + P_j)(P_j + N_j)(P_j + P_N)}}$$

where $P_j$ and $N_j$ are the correctly predicted splicing and non-splicing sites for class $j$, respectively, and $P_N$ and $N_N$ are similarly the incorrectly predicted sites and $j \in \{‘IE’, ‘EI’\}$ (the class ‘N’ is not considered for this indicator).

4.1. Literature comparison

In Tables 3–5 we compare the classification error rate, the correlation coefficient and the variance of the performance obtained by our hierarchical method (HM), by the systems proposed in (Thivierge and Schultz, 2002) (KBANN, MLP, ID3, NN) and (Rampone, 1998) (NNBRAIN, BRAIN) and by some stand-alone classifiers (used in our HM): a subspace classifier (SUBSPACE); a Linear SVM classifier (RANKSVM) trained using only the best 100 features corresponding to the higher values of DistF (see Section 2.1) reduced to a vector of 50 dimensions by KL; and another Linear SVM classifier (SVM) trained using orthonormal encoding as feature extractor (without the feature ranking) and KL as feature transformation (reducing the feature vector to 50 dimensions). This last

<table>
<thead>
<tr>
<th>Table 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error rates produced by HM and by various machine learning algorithms</td>
</tr>
<tr>
<td>System</td>
</tr>
<tr>
<td>HM</td>
</tr>
<tr>
<td>SUBSPACE</td>
</tr>
<tr>
<td>RANKSVM</td>
</tr>
<tr>
<td>SVM</td>
</tr>
<tr>
<td>NN BRAIN</td>
</tr>
<tr>
<td>BRAIN</td>
</tr>
<tr>
<td>KBANN</td>
</tr>
<tr>
<td>MLP</td>
</tr>
<tr>
<td>ID3</td>
</tr>
<tr>
<td>NN</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correlation coefficient produced by HM, NN-BRAIN and the stand-alone classifiers</td>
</tr>
<tr>
<td>System</td>
</tr>
<tr>
<td>HM</td>
</tr>
<tr>
<td>SUBSPACE</td>
</tr>
<tr>
<td>RANKSVM</td>
</tr>
<tr>
<td>SVM</td>
</tr>
<tr>
<td>NN BRAIN</td>
</tr>
</tbody>
</table>
classifier has been tested in order to prove that feature ranking is a major step for improving performance. These results show that our hierarchical multiclassifier outperforms all the methods proposed in the literature by Thiéвre and Schultz (2002) and Rampone (1998) and the considered stand-alone classifiers.

In Table 6 the classification performances of each step of the new hierarchical classifier are summarized: the local error rate is evaluated considering only the patterns effectively classified at each step, while the global error rate is the cumulative error obtained considering all the patterns classified till that step; analogously with “local classified” (LC) we mean the percentage of the whole patterns classified at each step, while “global classified” (GC) is the cumulative percentage of classified patterns at each step. It is interesting to note that the 15% of patterns can be considered “difficult”, since they contribute to generate the higher part of the total error rate.

The introduced method allows a light improvement (as shown in Tables 3 and 6) in comparison to the best “stand-alone” method: RANKSVM. The greater advantage in the use of our HM is a lower error rate and a higher correlation coefficient, with the same rejection rate, with respect to RANKSVM (as shown in Table 7). That is if we classify the 34.35% of patterns with higher confidence the error rate is 0.9% for RANKSVM while is 0.44% for HM.

4.2. Tests varying the dimensionality of the training set and of the validation set

In Tables 8 and 9 the classification performances of each step of our classifier are reported for different sizes of the training and test sets. These results show that both the global error rate and the percentage of pattern classified in each phase (LC) for the HM classifier are similar using different sizes for the training and the test sets. Moreover this test confirms that the size of the training set does not affect the conclusion about the advantage of our HM classifier versus the RANKSVM: a lower error rate with the same rejection value.

All parameters have been set using a validation set (200 patterns for class) extracted from the training set. The size of the validation set has been chosen by experimental considerations: a lower number of patterns per class would cause overfitting, while a larger number would reduce too much the size of the training set (fixed to 2000 in these experiments). In Table 10 the global error rate of HM for different sizes of the validation set is reported.

4.3. Rejection curve for SUBSPACE and RANKSVM

In Fig. 3 the rejection curves (global error rate) of two methods used in our multiclassifier (SUBSPACE and RANKSVM) are reported. It is possible to note the better
performance of the Subspace classifier with a higher rejection rate: this result justifies our choice of using Subspace before RankSVM in HM.

5. Conclusions

The problem addressed in this paper is to recognize, given a sequence of DNA letters, splice-junction sites. In this work, we proposed a hierarchical multiclassifier. We show by experimental results that our method is able to improve classification accuracy and correlation coefficient with respect to other state-of-art methods. Moreover our tests show that the size of the training set does not affect the conclusion about the advantage of our HM classifier versus other stand-alone classifiers: a lower error rate with the same rejection value.

References


