BIOINFORMATICS

Vol. 14 no. 8 1998
Pages 676–684

Recognition of splice junctions on DNA sequences by BRAIN learning algorithm

Salvatore Rampone

Dpt of Scienze Fisiche 'E.R.Caianiello', Università di Salerno, Via S. Allende, I-84081 Baronissi (Sa), Italy

Received on January 2, 1998; revised on April 4, 1998; accepted on June 10, 1998

Abstract

Motivation: *The problem addressed in this paper is the prediction of splice site locations in human DNA. The aims of the proposed approach are explicit splicing rule description, high recognition quality, and robust and stable 'one shot' data processing.*

Results: *These results are achieved by means of a new learning algorithm [BRAIN (Batch Relevance-based Artificial INtelligence)], described in the paper, inferring Boolean formulae from examples, and by considering the splicing rules as disjunctive normal form (DNF) formulae. The formula terms are computed in an iterative way, by identifying from the training set a relevance coefficient for each attribute. The classification is then refined by a neural network and combined with a discriminant analysis procedure. This splice site recognition method shows low error rates (0.0002 and 0.0003) and high correlation coefficient measures (0.83 and 0.81) for donor and acceptor sites, respectively; better than other methods.*

Availability: *The BRAIN package (Borland Turbo Pascal for Windows) is available on the EMBL file server. (ftp://ftp.ebi.ac.uk/pub/software/dos under nnbrain$.exe)*

Contact: *rampo@vaxsa.csied.unisa.it*

Introduction

Splice junctions are points on a DNA sequence at which 'superfluous' DNA is removed during the process of protein synthesis in higher organisms (Green, 1986). The problem posed is to recognize, given a sequence of DNA, the boundaries between exons (the parts of the DNA sequence retained after splicing) and introns (the parts of the DNA sequence that are spliced out), i.e. donor and acceptor sites. 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). Since this is the basis of coding and intron region prediction of uncharacterized genomic DNA, many methods and algorithms have been suggested for this task (Staden, 1984; Lapedes *et al*., 1988; Brunak *et al*., 1991; Solovyev *et al*., 1994). At present, the most successful splice site prediction techniques, whose use is established in gene-prediction systems worldwide, are based on back propagation neural networks (Brunak *et al*., 1991) and discriminant analysis (Solovyev *et al*., 1994). However, the best of these prediction methods shows high accuracy [a known bad indicator of predictive performance (Brunak *et al*., 1991)], but low correlation coefficients (0.63 and 0.47) for the splice-junction recognition problem (Solovyev *et al*., 1994). Its successful application for internal exon prediction is due to further steps, incorporating knowledge on intron and exon characteristics.

Our aim is to perfect the first segmentation step, and in this paper we combine the classical discriminant analysis with a new machine learning algorithm, called the BRAIN (Batch Relevance-based Artificial INtelligence) algorithm in the following, for binary (two class) classification rules, to recognize, given a sequence of DNA, the boundaries between exons and introns. In the machine learning approach, a learning algorithm receives a set of training examples, each labeled as belonging to a particular class. The algorithm's goal is to produce a classification rule (hypothesis) for correctly assigning new examples to these classes (Dietterich and Shaving, 1990). This process is called inductive inference (Michalski, 1983). Additional constraints are incorporated in the algorithm, for choosing an appropriate hypothesis. These constraints are called biases (Dietterich and Shaving, 1990). Bias typically takes two forms: restricted hypothesis space bias and preference bias. The former restricts the search of the unknown classification rule in a specific hypothesis space, where the hypotheses are defined in terms of their representations. Logical formulae remains the traditional means of representation both in formal AI systems (McCarthy and Hayes, 1969) and in actual applications, in spite of the difficulty in writing consistent theories, and of the computational problems (Kautz *et al*., 1995). In the BRAIN algorithm, the hypothesis space consists of logical formulae in disjunctive normal form (DNF). The main advantage of this representation with respect to other methods is that we can derive explicit splicing rule description. Furthermore, as evidenced by Solovyev *et al*. (1994), splice junctions that do not contain AG or GT indicate the existence of special mechanisms to recognize them. So the DNF hypothesis space is restricted to GT- or AG-containing

splice sites. Most of the preference biases attempt to minimize some measure of the hypothesis syntactic complexity. This follows Occam's razor: the simpler of two competing hypotheses should always be preferred. A theoretical explanation of why Occam's razor is successful has been presented by Blumer *et al*. (1987). As preference bias, the BRAIN algorithm tries to minimize the syntactic complexity of the hypothesis by means of a preference value called relevance. The performance of the learning algorithm is measured in terms of the hypothesis classification accuracy on previously unseen sequences, and of the computational effort required to produce the hypothesis. While theoretical analysis can provide bounds on the number of examples needed to guarantee a given level of confidence in the accuracy of learning, these bounds are based on worst-case arguments, and hence they tend not to be useful in practice. On the other hand, experimental methods based on the training set/test set methodology can provide good assessments of the accuracy of learned classification rules (Weiss and Kapouless, 1989). This technique has been widely employed in actual applications (Brunak *et al*., 1991; Solovyev *et al*., 1994), and this is the technique we apply here. The computational effort the BRAIN algorithm requires to produce the hypothesis is bounded by a polynomial in the length of the DNA window, and in the number of examples. To improve the classification accuracy, we also used a neural network, a feed-forward one (Bishop, 1996), on the functions produced by BRAIN. The method is then combined with a classical discriminant analysis procedure. We show that the resulting achieved error rates and related correlation coefficients are 0.0002 and 0.83 for donor, and 0.0003 and 0.81 for acceptor splice sites. Although application of inductive inference to the space of binary sequences of fixed length has undergone a tremendous explosion of interest [Dietterich and Shaving (1990) is an excellent collection of papers on the matter], our algorithm appears to be novel. The general methodology used in developing the learning algorithm is related to the STAR technique of Michalski (1983), to the candidate-elimination method introduced by Mitchell (1982), and to the work of Haussler (1988). About its application, past usage of the same DNA datasets in machine learning can be found in Noordewier *et al*. (1991), Towell *et al*. (1991), Brunak *et al*. (1991), and Towell and Shavlik, (1992), and in the StatLog project (http://www.ncc.up.pt/liacc/ML/statlog). Our results compare favorably with the biological literature (Brunak *et al*., 1991; Solovyev *et al*., 1994) and further show explicit description of splicing rules.

Systems and methods

The data

Two different sets of data are used in this study. The former referring material is the Irvine Primate splice-junction database, called IPData in the following. While this dataset is very out of date, its past use in the StatLog project allows comparisons with many machine learning algorithms, including both back propagation and discriminant analysis. There are 767 donor splice sites and 765 acceptor splice sites, where eight donor and four acceptor sites do not have the GT and AG conserved dinucleotide in flanking positions. As carried out by Brunak *et al*. (1991), in the experiments the symbolic variables representing the nucleotides (A,G,T,C) are replaced by four binary indicator variables:

$$A \rightarrow 1000;\ C \rightarrow 0100;\ G \rightarrow 0010;\ T \rightarrow 0001 \qquad (1)$$

The DNA sequence is then divided into windows of 60 nucleotides, corresponding to 240 binary attributes. Each window is labeled to belong to one of the three categories EI, IE and N. Categories EI and IE include every 'split-gene' for primates in Genbank 64.1, and non-splice (N) examples are taken from sequences known not to include a splicing site. According to the StatLog usage, there are 3186 examples, divided into training (2000) and test (1186) examples. Each class is indicated by a number:

$$EI \rightarrow 1;\ IE \rightarrow 2;\ N \rightarrow 3 \qquad (2)$$

Table 1 shows the example distribution over the classes. The latter dataset, called BruData in the following, is constituted by 93 of the 95 entries of Genbank 62.0 used by Brunak *et al*. (1991) in the NetGene assessment. [One sequence (HUMALBGC) was not included in the set provided by Brunak, and one sequence (HUMACCYBA) includes many unspecified bp (X). A known dataset annotation error (HUMAK1, 10517) has been corrected.] There are 630 donor splice sites and 630 acceptor splice sites, where one donor and one acceptor sites do not have the GT and AG conserved dinucleotide in flanking positions. As in the original study, the dataset is divided into two parts: a training set including 63 of all sequences (362 829 bp) and a test set containing the remaining ones (190 987 bp). Table 2 shows the example distribution over the classes.

Table 1. IPData example distribution

Class	Train	Test
EI(1)	464 (23.20%)	303 (25.55%)
IE(2)	485 (24.25%)	280 (23.61%)
N(3)	1051 (52.55%)	603 (50.84%)
Sum	2000	1186

Table 2. BruData example distribution

Class	Train	Test
EI(1)	312 (0.09%)	118 (0.06%)
IE(2)	312 (0.09%)	118 (0.06%)
N(3)	362 205 (99.82%)	190 751 (99.88%)
Sum	362 829	190 987

Problem description

So our problem is: given a window of n binary values, in the following called instance **w**, decide whether this belongs to class 1, 2 or 3. The problem can be further subdivided to find three Boolean classification rules

$$f_c : \{0,1\}^n \rightarrow \{0,1\} \tag{3}$$

such that f_c (**w**) = 1 if **w** belongs to the class c, and f_c (**w**) = 0 otherwise, for c = 1, …3. While f_3 may be considered redundant, it will be used in the classification refinement, as we will see in Results.

Formulae and properties

Since our hypothesis space consists of Boolean formulae, for the sake of clearness, we report some basic definitions. A Boolean classification rule (briefly, function) f_c on n variables

$$x_1, x_2, \ldots, x_n \tag{4}$$

each one taking only the values 1 or 0, is a mapping from the n-dimensional instance space $\{0, 1\}^n$ to $\{0, 1\}$

$$f_c : \{0,1\}^n \rightarrow \{0,1\} \tag{5}$$

An instance can be viewed as a value assignment to the variables in f_c. An instance is positive, and it is indicated as the vector

$$\boldsymbol{w}_i^+ = w_{i,1}^+, w_{i,2}^+, \ldots, w_{i,n}^+; \quad w_{i,k}^+ \in \{0,1\} \tag{6}$$

if f_c ($\boldsymbol{w}_i^+$) = 1, i.e. if $\boldsymbol{w}_i^+$ belongs to the class c. So p positive instances are indicated as $\boldsymbol{w}_1^+$, $\boldsymbol{w}_2^+$, …, $\boldsymbol{w}_p^+$. An instance is negative, and it is indicated as the vector

$$\boldsymbol{w}_j^- = w_{j,1}^-, w_{j,2}^-, \ldots, w_{j,n}^-; \quad w_{j,k}^- \in \{0,1\} \tag{7}$$

if f_c ($\boldsymbol{w}_j^-$) = 0, i.e. if $\boldsymbol{w}_j^-$ does not belong to the class c. So q negative instances are indicated as $\boldsymbol{w}_1^-$, $\boldsymbol{w}_2^-$, …, $\boldsymbol{w}_q^-$. A Boolean function is consistent with a set of instances if, and only if, it matches every positive instance and no negative instance in the set. A variable is in true form, and it is indicated as the literal x_k, if it assumes value 1 when it is assigned to 1, and 0 otherwise. On the contrary, a variable is in negated form, and it is indicated as the literal $\bar{x}_k$, if it assumes value 0 when x_k is assigned to 1, and 1 otherwise. A term m is the conjunction of any subset of the n variables x_1, x_2, …, x_n, each one in true or negated form. A term is made true by an instance if each of its variables is 1 by assigning the corresponding instance values. In the same way, a term is not made true by an instance if at least one of its variables is 0 by assigning the corresponding instance values.

Example 1: The term $m = x_1\bar{x}_2x_4$ is made true by the instances $\boldsymbol{w}_1^+ = \{1, 0, 0, 1\}$ and $\boldsymbol{w}_2^+ = \{1, 0, 1, 1\}$, and it is not made true by all the other instances $\boldsymbol{w}_1^-$, $\boldsymbol{w}_2^-$, …, $w_{2^4-2}^-$ of the four-dimensional Boolean space $\{0, 1\}^4$.

A Boolean function is expressed as a DNF formula if it is expressed as a disjunction (or) of terms. Given an instance, it assumes value 1 if at least one term of the function is made true by the assignment, and it assumes value 0 otherwise. An L-term DNF function can be represented as:

$$f_c = m_1 + m_2 + \ldots + m_L \tag{8}$$

Example 2: Let us consider the DNF function $f_c = x_1\bar{x}_2x_4 + x_2\bar{x}_3x_4$. It assumes value 1 on the positive instances $\boldsymbol{w}_1^+ = \{1, 0, 0, 1\}$, $\boldsymbol{w}_2^+ = \{1, 0, 1, 1\}$, $\boldsymbol{w}_3^+ = \{0, 1, 0, 1\}$ and $\boldsymbol{w}_4^+ = \{1, 1, 0, 1\}$, and it is 0 on all the other instances of the four-dimensional Boolean space $\{0, 1\}^4$.

Algorithm

Our aim is to infer a consistent DNF classification rule of minimum syntactic complexity from a set of instances. Here, the minimum syntactic complexity means the minimum number of terms, each one with the minimum number of variables.

One-to-one

Let us start finding a DNF formula that is consistent both with a positive instance and a negative one. Given a couple of instances ($\boldsymbol{w}_i^+$, $\boldsymbol{w}_j^-$), a term m made true by $\boldsymbol{w}_i^+$ and not made true by $\boldsymbol{w}_j^-$ must (we assume the instances to be self-consistent, i.e. an instance cannot be positive and negative at the same time) include at least one variable not assigned in the same way in $\boldsymbol{w}_i^+$ and $\boldsymbol{w}_j^-$. So given a couple ($\boldsymbol{w}_i^+$, $\boldsymbol{w}_j^-$), let us consider the set:

$$S_{i,j} = \left\{x_k | w_{i,k}^+ = 1, w_{j,k}^- = 0\right\} \bigcup \left\{\bar{x}_k | w_{i,k}^+ = 0, w_{j,k}^- = 1\right\}$$

It can be immediately seen that, given the set of instances $\{\boldsymbol{w}_i^+, \boldsymbol{w}_j^-\}$, then the formula

$$f_c = v_k; \qquad v_k \in S_{i,j} \tag{9}$$

where v_k is the kth variable in true or negated form, is consistent with the set and of minimum size.

Example 3: Given the instances $\boldsymbol{w}_1^+ = \{1, 0, 0, 1\}$ and $\boldsymbol{w}_1^- = \{1, 0, 1, 1\}$, we have $S_{1,1} = \{\bar{x}_3\}$. So the function $f_c = \bar{x}_3$ is consistent with the set and of minimum size.

One-to-many

Now let us consider having one positive instance and q negative ones:

$$\boldsymbol{w}_i^+, \boldsymbol{w}_1^-, \boldsymbol{w}_2^-, \ldots \boldsymbol{w}_q^- \tag{10}$$

In this case, different variables can disagree on different pairs $(\boldsymbol{w}_1^+, \boldsymbol{w}_j^-)$. Then a term, to be consistent with the given instances, needs to include at least one variable for each set

$$S_{i,j} = \left\{x_k | w_{i,k}^+ = 1, w_{j,k}^- = 0\right\} \bigcup \left\{\bar{x}_k | w_{i,k}^+ = 0, w_{j,k}^- = 1\right\} \quad j = 1, \ldots, q \tag{11}$$

Furthermore, as preference bias, we require a term of the smallest total size. This can be viewed as a set covering problem (Cormen *et al*., 1990), and while the problem is NP-hard (Cormen *et al*., 1990), an approximate solution can be found by a generalized greedy method (Johnson, 1974), as applied in machine learning by Haussler (1988). In our case, given the $S_{i,j}$ sets, it consists of choosing the variable that is in the most of sets, and to erase such sets, until there are no more sets. Obviously, this greedy choice can be improved by taking into account the size of the $S_{i,j}$ sets. [This was also applied by Johnson (1974) in the MAX-SAT approximation algorithm improvement.] For example, if there is only a variable in an $S_{i,j}$ set, that variable will be surely included in the term. We say that variable has a great relevance. Let us formalize this concept. For each $S_{i,j}$ set, we call the relevance of v_k in the $S_{i,j}$ set the value:

$$R_{i,j}(v_k) = \frac{I_{S_{i,j}}(v_k)}{\#(S_{i,j})} \tag{12}$$

where $I_{Si,j}(v_k)$ is the characteristic function of the S_{ij} set, i.e. it is 1 if $v_k \nabla S_{i,j}$ and 0 otherwise, then we call relevance of a variable v_k in the q $S_{i,j}$ sets the value:

$$R_i\ (v_k) = \frac{1}{q} \sum_{j=1}^{q} R_{i,j}\ (v_k) \tag{13}$$

So, given the $S_{i,j}$ sets, we repeatedly compute the relevances, choosing the variable with the greatest one, and erasing the sets including that variable, until there are no more sets.

Example 4: Given the instances $\boldsymbol{w}_1^+ = (1, 0, 0, 1)$, $\boldsymbol{w}_1^- = \{1, 0, 1, 1\}$ and $\boldsymbol{w}_2^- = \{1, 1, 0, 0\}$, we have $S_{1,1} = \{\bar{x}_3\}$, $S_{1,2} = \{\bar{x}_2, x_4\}$. The initial non-zero relevances are $R_1(\bar{x}_2) = ¼$, $R_1(\bar{x}_3) = ½$, $R_1(x_4) = ¼$. The variable $\bar{x}_3$ has the maximum relevance, and, once we have selected it, we erase the set $S_{1,1}$ only, leaving $S_{1,2} = \{\bar{x}_2, x_4)$. The new non-zero relevances are $R_1(\bar{x}_2) = ½$, $R_1(x_4) = ½$. We can choose any of the two variables in $S_{1,2}$, and then all the $S_{i,j}$ sets are erased. So a possible output is $\bar{x}_2\bar{x}_3$.

Many-to-many

Now let us consider having p positive instances and q negative ones:

$$w_1^+, w_2^+, \ldots w_p^+, w_1^-, w_2^-, \ldots w_q^- \tag{14}$$

Not necessarily the formula contains just a term. In the general case, a consistent DNF formula will be a disjunction of a set of terms. Each positive instance satisfies at least one term and none of the terms is satisfied by any negative instance. As in the one-to-many case, for each couple $(\boldsymbol{w}_i^+, \boldsymbol{w}_j^-)$ a term made true by $\boldsymbol{w}_i^+$ and not made true by any $\boldsymbol{w}_j^-$ must include at least a variable not assigned in the same way in $\boldsymbol{w}_i^+$ and $\boldsymbol{w}_j^-$ for each couple $\boldsymbol{w}_i^+, \boldsymbol{w}_j^-$. Then, for each $\boldsymbol{w}_i^+$, there is a term that contains at least a variable in each $S_{i,j}$ set. While the problem of finding the minimal consistent DNF formula is NP-hard (Kearns *et al*., 1987), an approximate solution can also be found by a generalized greedy method as applied by Haussler (1988). Given the instances, it consists of choosing the term that is made true by the greatest number of positive instances and it is not made true by all the negative ones, and erasing such positive instances, until there are no more positive instances. Unfortunately, such a method is impractical because the term selection step has high cost. It can be argued that it is possible to use an approximate solution for the term selection, as we will do in this paper, but a general performance evaluation is actually an open problem. What we do is to extend the relevance evaluation to take into account the relative frequency of a variable in the $S_{i,j}$ sets, for $i = 1, \ldots p$ and build the function terms in the iterative way of a one-to-many case. For each couple $(\boldsymbol{w}_i^+, \boldsymbol{w}_j^-)$, let us consider the sets:

$$S_{i,j} = \left\{x_k | w_{i,k}^+ = 1, w_{j,k}^- = 0\right\} \bigcup \left\{\bar{x}_k | w_{i,k}^+ = 0, w_{j,k}^- = 1\right\} \tag{15}$$

that we collect as

$$S_i = \left\{S_{i,1}, S_{i,2}, \ldots, S_{i,q}\right\} \tag{16}$$

and the relevances

$$R_i\ (v_k) = \frac{1}{q} \sum_{j=1}^{q} R_{i,j}\ (v_k) \tag{17}$$

To take into account the relative frequencies, we extend the relevance to all the $S_{i,j}$s, so having:

$$R(v_k) = \frac{1}{p} \sum_{i=1}^{p} R_i\ (v_k) = \frac{1}{pq} \sum_{i=1}^{p} \sum_{j=1}^{q} R_{i,j}\ (v_k) \tag{18}$$

By using this extended relevance, the greedy approximation algorithm is applied at the same time both to cover the greatest number of positive instances, and to select the least possible number of variables.

GT/AG bias

Finally, for class 1 and 2, we limit ourselves to the data having the GT and AG conserved dinucleotide in flanking positions. This is done by testing the presence of:

$$BIAS_1 = w_{i,123}, w_{i,128} = 1,1 \quad (GT) \qquad (19)$$

$$BIAS_2 = w_{i,113}, w_{i,119} = 1,1 \quad (GT) \qquad (20)$$

for class 1 and 2 instances, respectively. This constraint is then introduced in the function terms by adding:

$$mBIAS_1 = x_{123}x_{128} \qquad (21)$$

$$mBIAS_2 = x_{113}x_{119} \qquad (22)$$

Dummy values of $BIAS_c$ and $mBIAS_c$, are introduced for $c = 3$.

The remainder of the procedure is the classical greedy approximation. This procedure can be sketched as follows:

BRAIN Algorithm

Step 1: *Input*: n = variable number, c = class, $G = \{w_1^+, w_2^+ ... w_p^+, w_1^-, w_2^-, ... w_q^-\}$ the set of training instances. *Initialization*: Set $f_c := \phi$. *AG-GT Bias*: Erase from G the set of instances not including $BIAS_c$.

Step 2: While there are positive instances in G

2.1 $S_{i,j}$*-sets*: Build from G the sets $S_{i,j}$ and collect them in S_is.

2.2 *Start a new term*: Set $m = \phi$.

2.3 *Build the term*: While there are $S_{i,j}$ sets

2.3.1 *Relevances*: Compute the relevances $R(v_k)$.

2.3.2 *Add variable*: Select the variable v_k such that $R(v_k)$ is maximum. $m \leftarrow m \cup \{v_k\}$

2.3.3 *Update sets*: Erase the S_i sets not including v_k. Erase the $S_{i,j}$ sets including v_k.

2.4 *Add the term*: $f_c \leftarrow f_c + \{m \cup mBIAS_c\}$.

2.5 *Update instances*: Erase from G the positive instances satisfying m.

Step 3: *Output*: f_c

The resulting f_c is the inferred formula. It is easy to see that the algorithm has polynomial time complexity, upper bounded by $O(n^2t^3)$, where n is the variable number on which f_c is defined and t is the number of examples seen.

Implementation

The BRAIN algorithm has been implemented both in C and Pascal. The experiments in C were carried out on a SUN/OS SPARK II workstation. The Pascal version runs on Intel-based PCs.

Results

Evaluation measures

We will use as indicators of the predictive performance the error number, i.e. the number of patterns in the test set erroneously classified, the error rate, i.e. the ratio between the error number and the test set size, and the correlation coefficient, a measure that takes the relationship between correctly predicted positives and negatives as well as false positives and negatives:

$$C_c = \frac{P_c N_c - \overline{P}_c \overline{N}_c}{\sqrt{(N_c + \overline{N}_c)(N_c + \overline{P}_c)(P_c + \overline{N}_c)(P_c + \overline{P}_c)}} \qquad (23)$$

where P_c and N_c are the correctly predicted splicing and not-splicing sites for class c, respectively, and $\overline{P}_c$ and $\overline{N}_c$ are similarly the incorrectly predicted sites. In the experiments on real sequences, we also report the accuracy, i.e. the percentage of correctly predicted splice sites.

Base results

We applied the BRAIN algorithm on the IPData training set, building a formula f_c for each class. For each class, the training set consists of 2000 instances, and the test set of 1186 (see Table 3). The number of patterns in the test set erroneously classified, the error rate and the correlation coefficient are reported, for each class, in Table 4.

Table 3. Training and test sets

Class	Training (2000) Positive + (Negative)	Test (1186) Positive + (Negative)
EI(1)	464 + (485 + 1051)	303 + (280 + 603)
IE(2)	485 + (464 + 1051)	280 + (303 + 603)
N(3)	1051 + (464 + 485)	603 + (303 + 280)

Table 4. Base results

Class	Error number	Error rate	Correlation coefficient
EI(1)	41/1186	0.034	0.91
IE(2)	59/1186	0.049	0.86
N(3)	85/1186	0.071	0.86

Rule										Splice	Site									
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
EI_1				c							G	T	c	A	G					
EI_2						t		c		t	G	T			G					
EI_3		ct									G	T	A	t						
EI_4								C			G	T			G	T				
EI_5			t				C			G	G	T			G					
EI_6										a	G	T	A	A						
EI_7	t	t	g						A		G	T	G						C	
EI_8						G	t				G	T	A							
EI_9	t	t						g	A		G	T				G				
EI_{10}	G						A		A		G	T								
EI_{11}						A		C			G	T			T					
EI_{12}		c	G								G	T	G					C		
EI_{13}			A		t			t		G	G	T								
EI_{14}			G				T				G	T			G					
EI_{15}			T			T	C	A			G	T								
EI_{16}	G			A							G	T				A				
IE_1				C				C	A	G		t								g
IE_2			g	T	g			C	A	G			a		g					
IE_3	T						t	C	A	G					g				g	
IE_4	T				T				A	G					g				t	
IE_5	C		g	g				C	A	G		t								
IE_6			c	C				C	A	G			a	a	g					
IE_7			T					T	A	G		t								
IE_8					g		c	C	A	G	cg									
IE_9		T							A	G		a			g		g			G
IE_{10}							t	C	A	G			ag	a	gT					
IE_{11}	T					T			A	G		t			c					
IE_{12}					T			C	A	G			a				g	t		
IE_{13}				g				T	A	G				g						
IE_{14}	C	t	C			T		C	A	G										
IE_{15}				T				C	A	G				A		Gt			G	
IE_{16}					C		c	C	A	G		t			c					
IE_{17}			T				t	C	A	G		t								
IE_{18}	C	T							A	G			g							C
IE_{19}	a			T		C			A	G			t							G
IE_{20}		C	a			C	A		A	G									t	
IE_{21}				T		T	G	C	A	G									G	
IE_{22}		c		g	g		A		A	G										C
IE_{23}								T	A	G					G		T			
IE_{24}				C				C	A	G								G	cg	

Fig. 1. Splicing rules derived by BRAIN in the restricted window case.

Table 5. Restricted window results

Class	Error number	Error rate	Correlation coefficient
EI(1)	37/1186	0.031	0.91
IE(2)	51/1186	0.043	0.88
N(3)	59/1186	0.049	0.90

Restricted window

The dataset documentation indicates that much better performance is generally observed if attributes closest to the junction are used. In our case, this means restricting the window to 20 nucleotides by using binary attributes 81–160 only. Following this hint, we achieved the results reported in Table 5. Figure 1 reports the corresponding splicing rules. (In the figure, lowercase letters mean variables in negated form, i.e. there is not a nucleotide represented by the corresponding uppercase letter in that position. For example, given a sequence of 20 nucleotides, there is an EI junction in the middle according to rule 3 if nucleotide 2 is not C or T, nucleotides 11, 12, 13 are G, T, A, respectively, and nucleotide 14 is not T.) Further experimented restrictions (10–15 nucleotides) decrease the recognition performance.

Cross-validation

The results achieved by using a '10-fold cross-validation' methodology (Weiss and Kapouless, 1989) on 1000 examples randomly selected from the complete set of 3190 are reported in Table 6. We should point out that there is very little variance between the results of each experiment, i.e. the algorithm has very stable behavior. In the same table, there are the error rates produced by various machine learning algorithms on the same dataset and in the same experimental conditions (all experiments, except BRAIN, carried out at the University of Wisconsin) (Noordewier *et al.*, 1991; Towell *et al.*, 1991; Towell and Shavlik 1992).

Table 6. Error rates produced by BRAIN and by various ML algorithms by using a '10-fold cross-validation' methodology

System	EI(1)	IE(2)	Neither(3)
BRAIN	0.050	0.040	0.040
KBANN	0.076	0.085	0.046
BackProp	0.057	0.107	0.053
PEBLS	0.082	0.075	0.069
Perceptron	0.163	0.174	0.040
ID3	0.106	0.140	0.088
COBWEB	0.150	0.095	0.118
Near. Neighbor	0.116	0.091	0.311

Table 7. Neural network base results

Class	Error number	Error rate	Correlation coefficient
EI(1)	38/1186	0.032	0.91
IE(2)	59/1186	0.049	0.86

Neural network hybrid approach

To refine the classification accuracy, we used the output of the three functions as input to a single-layer feed-forward neural network with just two neurons (Bishop, 1996). Each neuron has sigmoidal activation. The net input are the four binary values $f_1(\mathbf{w})$, $f_2(\mathbf{w})$, $f_3(\mathbf{w})$, 1, where $\mathbf{w}$ is the example to be classified, $f_1(\mathbf{w})$, …, $f_3(\mathbf{w})$ are the DNF functions computed by BRAIN for each class, and 1 is a bias coefficient. The two outputs are labeled as the classes 1 and 2, and the greatest one over a fixed threshold indicates the classification. If the outputs are both under the threshold, we recognize a not-splicing site.

The net training has been by means of a delta rule (Bishop, 1996). Since the functions error is zero on the training set, the net training is made on the half of the test set, using the rest as test. Then a new random 2000 + 1186 set is selected, and, after BRAIN has been completed, the net is applied to the output. The network main rule is to solve ambiguous cases, and in fact it reduces the number of false donor and acceptor sites. The neural network hybrid approach results on base and restricted window data are reported in Tables 7 and 8, respectively. The overall error rate, given by the global number of misclassified patterns over the data set size, is reported in Table 9. In the same table, there are the corresponding values of some StatLog Project experimented systems. For the StatLog systems description, we refer to the StatLog documentation.

Table 8. Neural network restricted window results

Class	Error number	Error rate	Correlation coefficient
EI(1)	31/1186	0.026	0.93
IE(2)	54/1186	0.043	0.87

Table 9. Overall error rates on IPData

System	Overall error rate
NN BRAIN	0.021
Radial	0.025
Dipol92	0.022
Alloc80	0.061
QuaDisc	0.022
Discrim	0.043
LogDisc	0.028
Bayes	0.058
Castle	0.065
IndCart	0.052
C4.5	0.053
Cart	0.079
BackProp	0.042
BayTree	0.036
Cn2	0.037
Ac2	0.037
NewId	0.037
Smart	0.064
Cal5	0.114
Itrule	0.132
KNN	0.054
Kohonen	0.191
Default	0.481

BRAIN/discriminant combination

The BRAIN algorithm is then combined with a discriminant analysis methodology. The idea behind this process is quite simple. A discriminant analysis technique, based on a strong domain knowledge (Mural *et al*., 1990; Solovyev and Laurence, 1993; Solovyev *et al*., 1994), easily splices out most of the not-splicing sites (method details are reported in the Appendix). Nevertheless, its decision boundaries are hyperplanar (Bishop, 1996), and it can misclassify borderline examples, unless the border is moved, so reducing the error rate while decreasing correlation. On the other hand, DNF functions are sums of hypercubes, and while they can easily treat non-linearities, they may partially fill the instance space, especially when the ratio between positive and nega-

tive examples is <0.015, as in DNA sequences. However, since the two systems divide the instance space differently, they tend to fail on different instances. So we arrange them in a cascade. The first system is applied on the data, so obtaining a number of misclassified negative patterns (false positives) comparable to the positive instances. Then, the BRAIN functions are applied to reduce the false positives.

BruData results

We train the discriminant analysis method on the BruData training set, while the NN BRAIN functions are derived from the IPData. Attention has been paid to not overlapping the 2000 IPData training instances with the BruData test set material. After the systems training has been completed, the discriminant method is applied on the BruData test set. We apply an observation window of 110 bp, for each nucleotide, from the start +55 to the end –55 of each gene, and each window is coded as (1) and labeled as (2). After the analysis has been completed, the residual instances (the windows corresponding to splice sites correctly classified, and to false positives, i.e. all the instances where the discriminant analysis answers EI or IE) are reduced to 20/60 bp and given as input to the trained NN BRAIN. The results obtained in the restricted window case are reported in Table 10. In Table 11, we report the overall results of the combined method.

Table 10. BruData test set residual instances results

Class	Accuracy	Error rate	Correlation coefficient
EI(1)	98%	0.203	0.63
IE(2)	97%	0.199	0.64

Table 11. BruData test set combined method results

Class	Accuracy	Error rate	Correlation coefficient
EI(1)	96%	0.0002	0.83
IE(2)	92%	0.0003	0.81

Discussion

The problem addressed in this paper is to recognize, given a sequence of DNA, the boundaries between exons and introns. This is done by means of the BRAIN algorithm, inferring Boolean formulae from examples, and by considering the splicing rules as DNF formulae. The formula terms are computed in an iterative way, by identifying from the training set a relevance coefficient for each attribute. The result is refined by means of a neural network and combined with a discriminant analysis method. The major advantages of the proposed approach are the low error rates and high correlation measures, better than other 'stand-alone' methods, the explicit splicing rules description as a DNF formula, a polynomial computational complexity, and robust and stable 'one shot' learning. Furthermore, from a theoretical point of view, the computational learning approach (Dietterich and Shaving, 1990) shows that the hypotheses acquired from examples are approximately correct with a degree of probability that grows with the size of the training example. Similar results have been achieved in the field of pattern recognition under a variety of problem settings (Saitta and Bergadano, 1993). This means that the results obtained using the very limited IPData can be further improved by a larger and more precise set of data. As carried out by other authors, for exon prediction it is possible to combine the splicing method with the characteristics describing the 5-intron region, coding region and 3-intron region, and it will be the subject of a future study.

Acknowledgements

The author is grateful to S.Brunak for kindly providing his original data, to V.Colantuoni and G.Paolella for useful discussions, to E.Rampone for bibliographic assistance, and to the referees for very helpful comments on the first version of this paper.

References

Bishop,C.M. (1996) *Neural Networks for Pattern Recognition*. Clarendon Press, Oxford.

Blumer,A., Ehrenfeucht,A. and Haussler,D. (1987) Occam's razor. *Inf. Proc. Lett.*, **24**, 377–380.

Brunak,S., Engelbrecht,J. and Knudsen,S. (1991) Prediction of human mRNA donor and acceptor sites from the DNA sequence. *J. Mol. Biol.*, **220**, 49–65.

Cormen,T.H., Leiserson,C.H. and Rivest,R.L. (1990) *Introduction to Algorithms*. MIT Press, Cambridge.

Dietterich,T.G. and Shaving,J.W. (eds) (1990) *Readings in Machine Learning*. Morgan Kaufmann, San Mateo.

Green,M.R. (1986) Pre-mRNA splicing. *Annu. Rev. Genet.*, **20**, 671–708.

Haussler,D. (1988) Quantifying inductive bias: AI learning algorithms and Valiant's learning framework. *Artif. Intell.*, **36**, 177–222.

Johnsen,D.S. (1974) Approximation algorithms for combinatorial problems. *J. Comput. Syst. Sci.*, **9**, 256–278.

Kautz,H., Kearns,M. and Selman,B. (1995) Horn approximations of empirical data. *Artif. Intell.*, **74**, 129–145.

Kearns,M., Li,M., Pitt,L. and Valiant,L. (1987) On the learnability of Boolean formulae. In *Proceedings of the 9th Annual ACM Symposium on Theory of Computing*, pp. 285–295.

Lapedes,A., Barnes,C., Burks,C., Farber,R. and Sirotkin,K. (1988) Application of neural networks and other machine learning algorithm to DNA sequence analysis. *Proc. Santa Fe Inst.*, **7**, 157–182.

McCarthy,J. and Hayes,P.J. (1969) Some philosophical problems from the standpoint of artificial intelligence. *Machine Intell.*, **4**.

Michalski,R.S. (1983) A theory and methodology of inductive learning. *Artif. Intell.*, **20**, 111–116.

Mitchell,T.M. (1982) Generalization as search. *Artif. Intell.*, **18**, 203–226.
Mural,R.J., Mann,R.C. and Uberbacher,E.C. (1990) *Proceedings of the First International Conference of ESHG*. World Scientific, London, pp. 164–172.
Noordewier,M.O., Towell,G.G. and Shavlik,J.W. (1991) Training knowledge-based neural networks to recognize genes in DNA sequences. In *Advances in Neural Information Processing Systems.* Morgan Kaufman, Vol. 3.
Saitta,L. and Bergadano,F. (1993) Pattern recognition and Valiant's learning framework. *IEEE Trans. PAMI*, **15**, 145–155.
Solovyev,V.V. and Laurence,C. (1993) *Proceedings of the First International Conference on ISMB.* NLH HIH, Bethesda, 371–379.
Solovyev,V.V., Salamov,A.A. and Lawrence,C.B. (1994) Predicting internal exons by oligonucleotide composition and discriminant analysis of spliceable open reading frames. *Nucleic Acids Res*., **22**, 5156–5163.
Staden,R. (1984) Computer methods to locate signals in nucleic acids sequences. *Nucleic Acids Res*., **12**, 505–519.
Towell,G.G. and Shavlik,J.W. (1992) Interpretation of artificial neural networks: mapping knowledge-based neural networks into rules. In *Advances in Neural Information Processing Systems.* Morgan Kaufmann, Vol. 4.
Towell,G.G., Shavlik,J.W. and Craven,M.W. (1991) Constructive induction in knowledge-based neural networks. In *Proceedings of the 8th International Machine Learning Work.* Morgan Kaufmann.
Weiss,S. and Kapouless,I. (1989) An empirical comparison of pattern recognition, neural nets, and machine learning classification methods. In *Proceedings of the 11th IJCAI.* Morgan Kaufmann, Detroit. pp. 781–787.

Appendix: Discriminant analysis procedure

Feature extraction

The method takes some characteristics from each position (positions are reported by assuming an observation window of 110 bp) in a window around an AG/GT dinucleotide. Let $F_{j,tp}$ be the sample frequency of a specific triplet *tp* in the *j*th position in the positive instances, and let $\overline{F}_{j,tp}$ be the corresponding frequency in the negative instances. The preference of the triplet *tp* in position *j* is given by:

$$Pref_{j,tp} = \sigma\left(\frac{F_{j,tp}}{F_{j,tp} + \overline{F}_{j,tp}} - \alpha\right) \tag{24}$$

where $\alpha > 0.5$ is a significance threshold value and $\sigma(x)$ is x if $x > 0$ and 0 otherwise. The mean of this value for positive (M_k^+), and negative (M_k^-) instances is computed on different instance regions *k*:

1. *Donor sites* (26,51), (52,63), (64,107);
2. *Acceptor sites* (6,20), (21,47), (48,60), (61,85);

Then the number of G/GG/GGG in (63,107) and the number of T/C in (21,47) for donors and acceptors, respectively, is valued. Finally, in a way similar to the triplets, the preference in the octanucleotide composition, in (4,55) and (58,110) for donors, and (56,109) and (1,53) for acceptors, is computed.

Decision rule

An instance $\mathbf{w}_i$ is classified in the *c* class according to the following rule:

$$\sum_k \alpha_k \, z_k - s_c > 0 \tag{25}$$

where α_k and s_c are derived from the pooled covariance matrix, and

$$z_k = \sum_{regionk} tp(w_{i,j}, w_{i,j+1}, \ldots, w_{i,j+11}) Pref_{j,tp} \tag{26}$$

for triplets preferences

$$z_k = \sum_{regionk} B \tag{27}$$

for $B \in \{G/GG/GGG, C/T\}$

$$z_k = \sum_{regionk} oct(w_{i,j}, w_{i,j+1}, \ldots w_{i,j+31}) Pref_{j,oct} \tag{28}$$

for octanucleotide preferences.