An Error Tolerant Software Equipment for Human DNA Characterization

Salvatore Rampone, *Member, IEEE*

***Abstract*—We describe a learning algorithm for the prediction of splice site locations in human DNA in the presence of sequence annotation errors in the training data. Experimental results on a common dataset including errors are reported. We also give an efficient implementation. The resulting software package is publicly available.**

***Index Terms*—Algorithm, DNA, efficient implementation, learning, noise, splice.**

I. Introduction

GENE identification and characterization is a fundamental starting point for describing and understanding our structure, function, and development. However, eukaryotic genomes show a complex underlying structure. In such genomes a protein-coding gene, started by a promoter, and ended by a poly-A region, consists of a set of regions called exons usually interrupted by other regions called introns. Introns can interrupt both coding and noncoding (or untranslated) regions. The interruption points, exon-intron (EI) and intron-exon (IE) boundaries, are called donor and acceptor sites, respectively, and in general splice sites.

In fact introns are removed during a process called splicing. The splicing of introns is part of a multistep process of RNA maturation which takes place in the nucleus to generate mature (mRNA) molecules for transport to the cytoplasm. This process involves several factors such as small nuclear ribonucleoprotein particles (snRNPs) and heterogeneous nuclear ribonucleoprotein particles (hnRNPs). This complex assembly is called spliceosome. The existence is known of consensus sequences around splice sites. For example, the short consensus CAG/G, which is preceded by a region of pyrimidine abundance (polypyrimidine tract), is a typical feature of the acceptor sites [1]. However, a lot of similar sequences are not selected as true splice sites (false positives).

Due to the task complexity, the use of computational methods to predict genes and their structures (sets of spliceable exons) has attracted a considerable research attention in recent years. Many approaches focused on prediction of individual functional elements, e.g., promoters, splice sites, coding regions, in isolation or integrating multiple types of information including splice signal sensors, compositional properties of coding and noncoding DNA, and in some cases database homology searching. Examples of such approaches are NetGene [2], GeneID [3], GenMark [4], FGENEH [5], Genie [6], GENESCAN [7], VEIL [8], HMMgene [9], Bayes Networks [10], GeneFinder [11], and many others. However, the use of computational gene prediction tools in genome sequencing projects [12] has led to the suggestion that the computer algorithms used to identify the putative intron-exon junctions give incorrect estimates more frequently than expected. It is now recognized that accurate prediction of eukaryotic gene structure is dependent largely upon the ability to pinpoint exactly the splice site signals in a sequence [13].

In the machine learning approach we use here, the problem is posed as to find a classification rule: given a position in the middle of a window of DNA sequence elements (instance), decide whether this is an "exon → intron" boundary (EI), "intron → exon" boundary (IE), or neither (N). To find such a rule, a learning algorithm receives a set of training DNA windows, each labeled as belonging to a specific class. The algorithm's goal is to produce a classification rule (hypothesis) for correctly assigning new instances to these classes [14]. This process is called inductive inference [15].

The problem addressed in this paper is to define an experimental apparatus for the prediction of splice site locations in human DNA. To this aim we generalize a machine learning algorithm, called *batch relevance-based artificial intelligence* (BRAIN) algorithm [16][17], for binary (two class) classification rules. This algorithm was originally conceived for coding and intron region prediction of uncharacterized genomic DNA. The general methodology used in the algorithm is related to the STAR technique of Michalski [18], to the candidate-elimination method introduced by Mitchell [19], and to the work of Haussler [20].

We extend the algorithm to consider learning problems where there is noise on the data [21][22][23][24]. The modified algorithm, named BRAIN$^{\underline{cube}}$, given a set of DNA training data including sequence/class errors, a common error in the mayor public databases, produces a hypothesis for each kind of splice site. As we formally see in the following, the algorithm respect some constraints (bias): hypotheses are in disjunctive normal form (DNF), their syntactic complexity tends to be minimal, and they fit the given instances according to a noise parameter.

Roughly speaking, we introduce in the BRAIN algorithm an early stopping methodology [25]. Early stopping of training is one of the methods that aim to prevent overtraining due to a too powerful model class, noisy training instances, or a small training set [26]. As matter of fact the BRAIN training corresponds to an iterative reduction of the error function defined with respect to a set of training data. During a typical training session, this error generally decreases as a function of the number of iterations in the algorithm. However, the

Manuscript received November 13, 2003; revised June 24, 2004.

The author is with DSGA and INFM, Facoltà di Scienze MM.FF.NN., Università del Sannio, I-82100 Benevento, Italy (e-mail: rampone@unisannio.it).

Digital Object Identifier 10.1109/TNS.2004.835609

error measured with respect to independent data often shows a decrease at first, followed by an increase as erroneous instances are being considered. Training can be therefore stopped when the training error reaches a fixed error parameter.

Since in the afforded problem the ratio between splicing and nonsplicing instances is less than 0.015, we consider an error tolerance on the splicing instances only.

The hypotheses produced by the learning algorithm are used in a recognition package. Experimental results on a common dataset including errors show that BRAIN$^{\underline{cube}}$ outperforms its previous version, in particular in the complexity of the produced hypothesis.

The algorithm, while polynomial, needs to be optimized for large scale real world applications. We define an efficient implementation, indicated as FastBRAIN. The resulting software equipment, publicly available, is easily applicable to large scale problems.

This paper is organized as follows. In Section II we describe the learning environment, and, in Section III, we introduce the noise tolerant algorithm. Sections IV and V are devoted to the adopted optimization procedures. Section VI shows experimental results. Some examples and details are reported in Appendixes I and II.

II. Learning Environment

A *DNA sequence* can be viewed as a string over an alphabet of four symbolic variables (the four kinds of nucleotides A, G, T, C). Here the symbolic variables representing the nucleotides are replaced by four binary indicator variables

$$A \to 1000; \quad C \to 0100; \quad G \to 0010; \quad T \to 0001. \tag{1}$$

In our treatment the DNA sequence is then divided in observation windows (instances) of n binary attributes, corresponding to $n/4$ nucleotides.

Since the hypotheses we will treat are Boolean formulas, we need some basic definitions. A *Boolean classification rule* (briefly *function*) f_c on n *variables*

$$x_1, x_2, \ldots, x_n \tag{2}$$

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

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

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

$$\mathbf{w}_i^+ = w_{i,1}^+, w_{i,2}^+, \ldots, w_{i,n}^+; \quad w_{i,k}^+ \in \{0,1\} \tag{4}$$

if $f_c(\mathbf{w}_i^+) = 1$, i.e., if $\mathbf{w}_i^+$ belongs to the class c. So p positive instances are indicated as $\mathbf{w}_1^+, \mathbf{w}_2^+, \ldots, \mathbf{w}_p^+$.

An instance is *negative,* and it is indicated as the vector

$$\mathbf{w}_j^- = w_{j,1}^-, w_{j,2}^-, \ldots, w_{j,n}^-; \quad w_{j,k}^- \in \{0,1\} \tag{5}$$

if $f_c(\mathbf{w}_j^-) = 0$, i.e., if $\mathbf{w}_j^-$ does not belong to the class c. So q negative instances are indicated as $\mathbf{w}_1^-, \mathbf{w}_2^-, \ldots, \mathbf{w}_q^-$.

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 $x_1, x_2, \ldots, x_n$, 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.

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

$$f_c = m_1 + m_2 + \cdots + m_L. \tag{6}$$

By the previous definitions our problem is: given a window of n binary values, in the following called *instance* $\mathbf{w}$, decide if the middle point is a splice site, i.e., if the instance belongs to a certain class c [$1 =$ exon-intron junction (EI), $2 =$ intron-exon junction (IE), $3 =$ no junction (N)] by finding a Boolean classification rule

$$f_c : \{0,1\}^n \to \{0,1\} \tag{7}$$

such that $f_c(\mathbf{w}) = 1$ if $\mathbf{w}$ belongs to the class c, and $f_c(\mathbf{w}) = 0$ otherwise.

III. The Error Tolerant Algorithm

Our goal is to find a classification rule, which captures the underlying systematic aspects of the data without fitting the noise on the data.

A known learning algorithm (BRAIN), given a set of labeled training data, returns a DNF function which is consistent with all the training instances (provided that we have no contradictory instances), and of approximately minimum size. To this aim the algorithm builds iteratively the function terms by means of a probability distribution over the literals derived from the training data. This probability distribution is built as follows.

Let us consider to have p positive instances and q negative ones

$$\mathbf{w}_1^+, \mathbf{w}_2^+, \ldots, \mathbf{w}_p^+, \mathbf{w}_1^-, \mathbf{w}_2^-, \ldots, \mathbf{w}_q^-. \tag{8}$$

Let us consider the sets

$$S_{i,j} = \{x_k \,|\, w^+_{i,k} = 1, w^-_{j,k} = 0\} \cup \{\bar{x}_k \,|\, w^+_{i,k} = 0, w^-_{j,k} = 1\} \tag{9}$$

that we collect as

$$S_i = \{S_{i,1}, S_{i,2}, \ldots, S_{i,q}\}. \tag{10}$$

The relative frequencies of a the variables in the $S_{i,j}$ sets are computed

$$R(v_k) = \frac{1}{pq} \sum_{i=1}^{p} \sum_{j=1}^{q} R_{i,j}(v_k) \tag{11}$$

where

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

and where $I_{S_{i,j}}(v_k)$ is the characteristic function of the $S_{i,j}$ set, that is 1 if $v_k \in S_{i,j}$ and is 0 otherwise. The $R(v_k)$ coefficients, called relevances, form our probability distribution.

The BRAIN algorithm uses the relevance as a choice criteria to build DNF function terms. The choice aims at the same time both to cover greatest number of positive instances, and to select the less possible number of variables. An algorithm scheme is reported in Appendix I.[1]

By several reasons [27], annotation errors may be present in the training data. In some cases the error is not significative, since the instance middle point corresponds in effect to the indicated kind of junction; in the general case, an annotation error may lead to a misclassified training instance.

In such a noisy environment, the structural risk minimization principle [28] tells us that a function which makes a few errors on the training set might give better generalization than a larger function (with more features and more conjunctions) which makes zero empirical error.

To include this flexibility into the BRAIN algorithm, we limit the hypothesis consistency by early stopping the learning algorithm when there remains a few more training instances to be covered.

Let us formalize this step. Let p and q be the number of positive and negative instance given. A parameter ε_p will specify the error tolerance on the positive instances. A similar ε_q will account of the errors allowed on the negative instance set.

Now let us suppose that, during the algorithm iterations, there are p_r positive instances to be covered. If $(p_r/p) \leq \varepsilon_p$ such instances are discarded.

It is worth noting that the introduction of the noise parameter ε_p does not alter the BRAIN produced DNF terms, but just their number. The reduced terms will account for at least $(1 - \varepsilon_p)p$ training positive instances.

[1]In the reported scheme, the original genomic bias has been removed.

In the same way let us suppose that there are q_r negative instances to be considered. The adding procedure halts when $(q_r/q) \leq \varepsilon_q$.

The resulting procedure can be sketched as follows:

BRAIN$^{\mathbf{C}_{ube}}$ Algorithm

Step 1: *Input:* $n =$ variable number,
$G = \{\mathbf{w}^+_1, \mathbf{w}^+_2, \ldots, \mathbf{w}^+_p, \mathbf{w}^-_1, \mathbf{w}^-_2, \ldots, \mathbf{w}^-_q\}$ the set of training instances.
$\varepsilon_p, \varepsilon_q =$ noise parameters
$p =$ positive instance number
$q =$ negative instance number
Initialization: Set $f_c = \emptyset; p_r = p$.

Step 2: While $(p_r/p) > \varepsilon$ do
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 $m = \emptyset$. $q_r = q$.
2.3 *Build the term :* While $(q_r/q) > \varepsilon$ do
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 and update p_r. Erase the $S_{i,j}$ sets including v_k and update q_r.
2.4 *Add the term:* $f_c \leftarrow f_c + m$.
2.5 *Update instances:* Erase from G the positive instances satisfying m.

Step 3: *Output:* f_c

IV. Difference Set Management

In the described procedure we evidence the S_{ij} management as a main computational drawback. In fact it involves

$$O(pqn) \tag{13}$$

steps. In real world cases, as p, q, and n grow, such a computation appears to be quite expensive.

However, by definition (9), S_{ij}s can be derived from the generating $\mathbf{w}^+_i, \mathbf{w}^-_j$.

In fact, initially S_{ij}s are derived from the given positive and negative instances (8). When a new literal v_k is selected by the relevance, we perform two steps (2.3.3):

1) Erase the S_i sets not including v_k and
2) Erase the $S_{i,j}$ sets including v_k.

Theorem 1: The $S_{i,j}$ update step 1) can be done by erasing $\mathbf{w}^+_i$ having 0 in position k if v_k is in true form, or $\mathbf{w}^+_i$ having 1 in position k if v_k is in negated form, i.e., the positive instances whose indexes belong to

$$I = \{i \,|\, v_k(w^+_{i,k}) = 0\}. \tag{14}$$

Proof: S_i sets not including v_k are obtained by comparing to the negative instances a $\mathbf{w}^+_i$ that does not make true v_k, i.e.,

- a $\mathbf{w}^+_i$ not having 1 in position k if v_k is in true form; or
- a $\mathbf{w}^+_i$ not having 0 in position k if v_k is in negated form.

As a consequence, if we erase such an instance $\mathbf{w}_i^+$ from the initial set (8), the corresponding S_i set can not be generated. In other words the elimination step can be performed over the $\mathbf{w}_i^+$s.

In the same way, we have Theorem 2.

Theorem 2: The update step 2) can done by erasing $\mathbf{w}_j^-$s having 0 in position k if v_k is in true form, or $\mathbf{w}_j^-$ having 1 in position k if v_k is in negated form, i.e., the negative instances whose indexes belong to

$$J = \{j \mid v_k(w_{j,k}^-) = 0\}. \tag{15}$$

Proof: $S_{i,j}$ sets including v_k are made by comparing positive instances to a $\mathbf{w}_j^-$ that do not make v_k true, i.e., to

- a $\mathbf{w}_j^-$ not having 1 in position k if v_k is in true form; or
- a $\mathbf{w}_j^-$ not having 0 in position k if v_k is in negated form.

Then the potential of S_{ij} can be erased by erasing the corresponding $\mathbf{w}_j^-$.

In this way we can substitute to the pq sets S_{ij} at most $p+q$ instances $\mathbf{w}_i^+, \mathbf{w}_j^-$.

A simple example is reported in Appendix II.

V. Relevance Computation

Obviously, the $S_{i,j}$ avoiding advantage may be outperformed by the relevance (11) computation. So let us consider (12). In this equation $\#(S_{i,j})$ is the number of differences between $\mathbf{w}_i^+$ and $\mathbf{w}_j^-$, i.e., the Hamming distance $d_H(\mathbf{w}_i^+, \mathbf{w}_j^-)$. Furthermore, by definition (9), $I_{S_{i,j}}(v_k)$ in (12) can be viewed as

$$\begin{aligned} I_{S_{i,j}}(x_k) &= w_{i,k}^+(1 - w_{j,k}^-) \\ I_{S_{i,j}}(\bar{x}_k) &= (1 - w_{i,k}^+)w_{j,k}^- \end{aligned} \tag{16}$$

and (11) becomes

$$\begin{aligned} R(x_k) &= \frac{1}{pq}\sum_{i=1}^{p}\sum_{j=1}^{q}\frac{w_{i,k}^+(1 - w_{j,k}^-)}{d_H(\mathbf{w}_i^+, \mathbf{w}_j^-)} \\ R(\bar{x}_k) &= \frac{1}{pq}\sum_{i=1}^{p}\sum_{j=1}^{q}\frac{(1 - w_{i,k}^+)w_{j,k}^-}{d_H(\mathbf{w}_i^+, \mathbf{w}_j^-)}. \end{aligned} \tag{17}$$

We observe that the quantity $(1/pq)$ is a just a scaling factor, and it can be omitted, and that $d_H(\mathbf{w}_i^+, \mathbf{w}_j^-)$ can be computed once and for all as $d_{i,j} \forall i, j$. In this way we value the relevances by

$$\begin{aligned} \overline{R}(x_k) &= \sum_{i=1}^{p}\sum_{j=1}^{q}\frac{w_{i,k}^+(1 - w_{j,k}^-)}{d_{i,j}} \\ \overline{R}(\bar{x}_k) &= \sum_{i=1}^{p}\sum_{j=1}^{q}\frac{(1 - w_{i,k}^+)w_{j,k}^-}{d_{i,j}}. \end{aligned} \tag{18}$$

These quantities can be computed just once for each term. In fact, by using (18), it is easy to see that, when we update $S_{i,j}$s, as in Section IV, the corresponding relevance update is for step 1)

$$\begin{aligned} \bar{R}'(x_k) &= \bar{R}(x_k) - w_{i,k}^+\sum_{j=1}^{q}\frac{(1 - w_{j,k}^-)}{d_{i,j}} \\ \bar{R}'(\bar{x}_k) &= R(\bar{x}_k) - (1 - w_{i,k}^+)\sum_{j=1}^{q}\frac{w_{j,k}^-}{d_{i,j}} \end{aligned} \tag{19}$$

and for step 2)

$$\begin{aligned} \bar{R}'(x_k) &= R(x_k) - (1 - w_{j,k}^-)\sum_{i=1, i\notin I}^{p}\frac{w_{i,k}^+}{d_{i,j}} \\ \bar{R}'(\bar{x}_k) &= R(\bar{x}_k) - w_{j,k}^-\sum_{i=1, i\notin I}^{p}\frac{(1 - w_{i,k}^+)}{d_{i,j}}. \end{aligned} \tag{20}$$

The considerations of the last two sections lead to the following scheme:

FastBRAIN Algorithm

Step 1: *Input:* $n =$ variable number,
$G = \{\mathbf{w}_1^+, \mathbf{w}_2^+, \ldots, \mathbf{w}_p^+, \mathbf{w}_1^-, \mathbf{w}_2^-, \ldots, \mathbf{w}_q^-\}$ the set of training instances.
$\varepsilon_p, \varepsilon_q =$ noise parameters
$p =$ positive instance number
$q =$ negative instance number
Initialization: Set $f_c = \emptyset; p_r = p;$
$d_{i,j} \leftarrow d_H(\mathbf{w}_i^+, \mathbf{w}_j^-) \quad \forall i, j.$

Step 2: While $(p_r/p) > \varepsilon$ do
2.1 $S \leftarrow G$
2.2 *Relevances:* Compute the relevances $\bar{R}(v_k)$ by (18).
2.3 *Start a new term:* Set $m = \emptyset$. $q_r = q$.
2.4 *Build the term:* While $(q_r/q) > \varepsilon$ do
2.4.1 *Add variable:* Select the variable v_k such that $\bar{R}(v_k)$ is maximum. $m \leftarrow m \cup \{v_k\}$
2.4.2 *Update S and Relevances:* Let I the set of indexes (14). Erase from S the instances $\mathbf{w}_i^+$ and update relevances by (19), $\forall i \in I$; and update p_r.
Let J the set of indexes (15). Erase from S the instances $\mathbf{w}_j^-$ and update relevances by (20), $\forall j \in J$; and update q_r
2.5 *Add the term:* $f_c \leftarrow f_c + m$.
2.6 *Update instances:* Erase from G the positive instances satisfying m.

Step 3: *Output:* f_c

VI. Results

A. Evaluation Measures

We will use several indicators of the predictive performances. The *error number* is the number of patterns in the test set erroneously classified. The *error rate* is the ratio between the error

TABLE I
IPDATA NUCLEOTIDE DISTRIBUTION

Nucleotide	Neither	EI	IE
A	24.984%	22.153%	20.577%
G	25.653%	31.415%	22.383%
T	24.273%	21.771%	26.445%
C	25.077%	24.561%	30.588%
D	0.001%	–	0.002%
N	0.010%	0.010%	–
S	–	–	0.002%
R	–	–	0.002%

TABLE II
IPDATA INSTANCE 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

number and the test set size. The *correlation coefficient* is a measure that takes the relation between correctly predicted positives and negatives as well as false positives and negatives

$$C_c = \frac{P_c N_c - \overline{P}_c \overline{N}_c}{\sqrt{(N_c + \overline{N}_c)(N_c + \overline{P}_c)(P_c + \overline{N}_c)(P_c + \overline{P}_c)}} \quad (21)$$

where P_c and N_c are the correctly predicted spicing and not splicing sites for class c, respectively, and $\bar{P}_c$ and $\bar{N}_c$ are similarly the incorrectly predicted sites.

As indicator of generalization capabilities we will also value the *number of function terms*.

Finally, we will consider the FastBRAIN execution time and the *ratio* between execution times of FastBRAIN and BRAIN implementations as a measure of relative speedup.

B. Data Set

The referring material is the Irvine Primate spice-junction dataset extracted from Genbank 64.1 (IPData). While this dataset is very out of date, its past usage in the StatLog project allows comparisons with many machine learning algorithms. An extensive comparison of the BRAIN algorithm against several others on these data has been performed in previous works [16]. The results evidenced the algorithm low error rates and high correlation measures, better than other methods.

There are 767 donor splice sites and 765 acceptor splice sites, where eight donor and four acceptor sites do not have GT and AG conserved dinucleotide in flanking positions. The complete list is available online [29].

Table I shows the IPData nucleotide distribution. For further details we refer to the dataset documentation [30]–[32].

As made by Brunak *et al.* [2], in the experiments the symbolic variables representing the nucleotides (A, G, T, C) are replaced by four binary indicator variables, as in (1).

The DNA sequence is then divided in 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 "splitgene" for primates in Genbank 64.1, and nonsplice (N) instances are taken from sequences known not to include a splicing site.

TABLE III
BASE RESULTS

ε_p	Class	Error number	Error rate	Correlation	Terms
0.000	EI(1)	41/1186	0.034	0.91	13
	IE(2)	59/1186	0.049	0.86	18
	N(3)	85/1186	0.071	0.86	21
0.001	EI(1)	41/1186	0.034	0.91	13
	IE(2)	59/1186	0.049	0.86	18
	N(3)	84/1186	0.071	0.86	21
0.010	EI(1)	38/1186	0.032	0.91	11
	IE(2)	69/1186	0.058	0.81	16
	N(3)	70/1186	0.039	0.88	10
0.100	EI(1)	79/1186	0.066	0.82	3
	IE(2)	186/1186	0.156	0.51	2
	N(3)	199/1186	0.168	0.70	2

TABLE IV
RESTRICTED WINDOW RESULTS

ε_p	Class	Error number	Error rate	Correlation	Terms
0.000	EI(1)	40/1186	0.034	0.91	16
	IE(2)	52/1186	0.044	0.88	24
	N(3)	58/1186	0.049	0.90	26
0.001	EI(1)	40/1186	0.034	0.91	16
	IE(2)	52/1186	0.044	0.88	24
	N(3)	53/1186	0.045	0.91	24
0.010	EI(1)	42/1186	0.035	0.91	10
	IE(2)	50/1186	0.042	0.88	18
	N(3)	53/1186	0.045	0.91	10
0.100	EI(1)	92/1186	0.078	0.79	3
	IE(2)	167/1186	0.141	0.58	3
	N(3)	196/1186	0.165	0.71	2

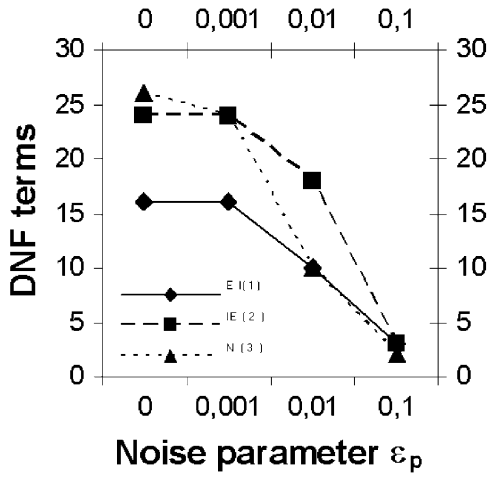

Fig. 1. DNF terms as function of the noise parameter in the restricted window case.

According to the StatLog usage, there are 3186 instances, divided in training (2000) and test (1186) instances.

Each class is indicated by a number

$$EI \to 1; \quad IE \to 2; \quad N \to 3. \quad (22)$$

Table II shows the instance distribution over the classes.

ε_p	Rule	Splice										Site									
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0.000/0.001	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				
0.010	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								
0.100	EI_1				c							G	T	c	A	G					
	EI_2						t		c		t	G	T			G					
	EI_3		ct									G	T	A	t						

Fig. 2. EI DNF terms as function of the noise parameter in the restricted window case.

C. Base Results

Since the ratio between positive and negative instances is less than 0.015 in DNA sequences, we consider an error tolerance on the positive instances only. This means $\varepsilon_q = 0$.

We applied the FastBRAIN algorithm on the IPData, varying the ε_p parameter in the range [0.000, 0.100], and building a formula f_c for each class. For each class, the training set consists of 2000 instances, and the test set of 1186. The number of patterns in the test set erroneously classified, the error rate, and the correlation coefficient are reported, for each class, in Table III. In the table we also report the number of function terms. It is easy to see the main advantage of the introduction of the noise parameter is the dramatic reduction of function terms. The error rate and the correlation are preserved or improved until ε_p approaches 0.100. They slowly degrade as ε_p further grows. In the considered case, the best choice appears to be a noise parameter in the order of 0.010.

D. Restricted Window

Better performances are generally observed if attributes closest to the junction are used. In our case this means to restrict the window to 20 nucleotides by using binary attributes 81–160 only. We achieved the results reported in Table IV

Fig. 1 shows the dramatic reduction of the function terms as the noise parameter grows. Also in this case the best tradeoff between correlation and function terms appears to be $\varepsilon_p = 0.010$.

Figs. 2 and 3 report the EI and IE DNF terms as function of the noise parameter. For $\varepsilon_p = 0.010$, ten terms account for the first 458 of 464 positive EI instances (99%) (In the figures 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 EI_3 if nucleotide 2 is not C nor T, nucleotides 11, 12, 13 are G, T, A, respectively, and nucleotide 14 is not T.).

E. Speedup

The FastBRAIN running time has been computed on a common Intel Celeron based PC. The results are reported in Table V. The times are in the order of few seconds, and decreasing as the error parameter grows.

The FastBRAIN running time is then compared to the BRAIN one, for each class. To make the test meaningful, the error parameter has been set to zero. The results, reported in Table VI, show a mean ratio of 0.0720. In other words the FastBRAIN implementation is about 14 times faster than the BRAIN one.

VII. Conclusion

The problem addressed in this paper is to recognize, given a sequence of DNA, the boundaries between exons and introns. This is due by means of a learning algorithm, described in this paper, inferring Boolean formulas from noisy instances, and by considering the splicing rules as DNF formulas. The formula terms are computed in an iterative way, by identifying from the training set a relevance coefficient for each attribute. As in previous papers [16], the result can be further refined by means of a neural network and combined to a discriminant analysis method.

The proposed approach introduces a noise tolerance by means of an early stopping methodology. It also maintains the low error rates and high correlation measures, the explicit splicing rules description as a DNF formula, and the polynomial computational complexity of the original algorithm.

This also involves a dramatic reduction of the function size, an empirical indicator of learning time reduction and of better generalization properties on problems of greater size.

ε_p	Rule	Splice										Site									
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0.000/0.001	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	
0.010	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
0.100	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	

Fig. 3. IE DNF terms as function of the noise parameter in the restricted window case.

TABLE V
FASTBRAIN EXECUTION TIMES ON IPDATA

Junction	ε_p	time (s)
EI(1)	0.000	1.172
	0.001	1.170
	0.010	1.161
	0.100	1.092
IE(2)	0.000	5.538
	0.001	5.488
	0.010	5.497
	0.100	5.177
NN(3)	0.000	23.504
	0.001	23.293
	0.010	22.612
	0.100	18.697

Finally, the given implementation, called FastBRAIN, publicly available [33], shows a very good speed up of the tested benchmark, and makes the algorithm easily applicable to large scale real world problems. Preliminary results on newly obtained genome data are in agreement with the results obtained by using the IPData.

TABLE VI
RATIO BETWEEN FASTBRAIN AND BRAIN EXECUTION TIMES

Class	Ratio
EI(1)	0,0526
IE(2)	0,0857
N(3)	0,0777
Mean	0,0720

APPENDIX I
BRAIN ALGORITHM

BRAIN Algorithm

Step 1: *Input:* $n =$ variable number,

$G = \{\mathbf{w}_1^+, \mathbf{w}_2^+, \ldots, \mathbf{w}_p^+, \mathbf{w}_1^-, \mathbf{w}_2^-, \ldots, \mathbf{w}_q^-\}$

the set of training instances.

$p =$ positive instance number

$q =$ negative instance number

Initialization: Set $f_c = \emptyset$.

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 $m = \emptyset$.

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:* $f_c \leftarrow f_c + m$.

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.

APPENDIX II
EXAMPLE 1

Let the given instances be

$$\mathbf{w}_1^+ = \{1,0,0,1\}; \quad \mathbf{w}_2^+ = \{1,0,1,0\};$$
$$\mathbf{w}_1^- = \{1,0,1,1\}; \quad \mathbf{w}_2^- = \{1,1,0,0\}.$$

We set

$$S = \{\mathbf{w}_1^+, \mathbf{w}_2^+, \mathbf{w}_1^-, \mathbf{w}_2^-\}$$

corresponding to

$$S_1 = \{S_{1,1} = \{\bar{x}_3\}; S_{1,2} = \{\bar{x}_2, x_4\}\},$$
$$S_2 = \{S_{2,1} = \{\bar{x}_4\}; S_{2,2} = \{\bar{x}_2, x_3\}\}.$$

By computing the relevances, we have the nonzero values

$$R(x_3) = 1/8; R(x_4) = 1/8;$$
$$R(\bar{x}_2) = 1/4; \quad R(\bar{x}_3) = 1/4; \quad R(\bar{x}_4) = 1/4.$$

$\bar{x}_2$ is selected and $S_{1,2}$ e $S_{2,2}$ are erased, or, equivalently, we exclude $\mathbf{w}_2^-$. Then

$$S = \{\mathbf{w}_1^+, \mathbf{w}_2^+, \mathbf{w}_1^-\}$$

corresponding to

$$S_1 = \{S_{1,1} = \{\bar{x}_3\}\}, \quad S_2 = \{S_{2,1} = \{\bar{x}_4\}\}.$$

Now the nonzero relevances are

$$R(\bar{x}_3) = 1/2; \quad R(\bar{x}_4) = 1/2.$$

$\bar{x}_3$ is selected and $S_{1,1}$ is erased, or, equivalently, we exclude $\mathbf{w}_1^- = \{1,0,1,1\}$, and S_2 is erased or, equivalently, we exclude $\mathbf{w}_2^+ = \{1,0,1,0\}$. Then there are no more $S_{i,j}$ sets, or, equivalently, there are no more negatives in S. The term $m = \bar{x}_2\bar{x}_3$ is added to the function.

In the next step we have

$$S = \{\mathbf{w}_2^+, \mathbf{w}_1^-, \mathbf{w}_2^-\}$$

corresponding to

$$S_2 = \{S_{2,1} = \{\bar{x}_4\}; S_{2,2} = \{\bar{x}_2, x_3\}\}.$$

The nonzero relevances are

$$R(x_3) = 1/4; \quad R(\bar{x}_2) = 1/4; \quad R(\bar{x}_4) = 1/2.$$

$\bar{x}_4$ is selected and $S_{2,1}$ is erased or, equivalently, we exclude $\mathbf{w}_1^-$. Then

$$S = \{\mathbf{w}_2^+, \mathbf{w}_2^-\}$$

i.e.,

$$S_2 = \{S_{2,2} = \{\bar{x}_2, x_3\}\}.$$

The remaining nonzero relevances are

$$R(x_3) = 1/2; \quad R(\bar{x}_2) = 1/2.$$

x_3 is selected and $S_{2,2}$ is erased, or, equivalently, we exclude $\mathbf{w}_2^-$. There are no more constraints: the term $m = \bar{x}_4 x_3$ is added to the function.

The process ends with a resulting

$$f = \bar{x}_2\bar{x}_3 + \bar{x}_4 x_3.$$

ACKNOWLEDGMENT

The author is grateful to Prof. A. Aloisio (University of Sannio, Italy) for the useful discussions, to V. Morfino (CTC Computer Technology Consultancy, Italy) for his help in prototyping the BRAIN$^{\underline{\text{cube}}}$ on-line version, and to the referees for their useful comments.

REFERENCES

[1] P. Senapathy, M. B. Shapiro, and N. L. Harris, "Splice junctions, branch point sites, and exons: Sequence statistics, identification, and applications to genome project," *Meth. Enzymol.*, vol. 183, pp. 252–278, 1990.

[2] S. Brunak, J. Engelbrecht, and S. Knudsen, "Prediction of the human mRNA donor and acceptor sites from the DNA Sequence," *J. Mol. Biol.*, vol. 220, pp. 49–65, 1991.

[3] R. Guigo, S. Knudsen, N. Drake, and T. Smith, "Prediction of gene structure," *J. Mol. Biol.*, vol. 226, pp. 141–157, 1992.

[4] M. Borodovsky and J. McIninch, "Recognition of genes in DNA sequence with ambiguities," *Biosystems*, vol. 30, pp. 161–171, 1993.

[5] V. V. Solovyev, A. A. Salamov, and C. B. Lawrence, "Predicting internal exons by oligonucleotide composition and discriminant analysis of spliceable open reading frames," *Nucleic Acids Res.*, vol. 22, pp. 5156–5163, 1994.

[6] D. Kulp, D. Haussler, M. G. Reese, and F. H. Eeckman, "A generalized hidden Markov model for the recognition of human genes in DNA," in *Proc. Int. Conf. Intell. Syst. Mol. Biol.*, 1996, pp. 134–142.

[7] C. Burge and S. Karlin, "Prediction of complete gene structure in human genomic DNA," *J. Mol. Biol.*, vol. 268, pp. 78–94, 1997.

[8] J. Henderson, S. Salzberg, and K. H. Fasman, "Finding genes in DNA with a hidden Markov model," *J. Comput. Biol.*, vol. 4, pp. 127–141, 1997.
[9] A. Krogh, "An introduction to hidden Markov models for biological sequences," in *Computational Methods in Molecular Biology*, S. L. Salzberg, D. B. Searls, and S. Kasif, Eds. Amsterdam, Elsevier, the Netherlands, 1998, pp. 45–63.
[10] D. Cai, A. Delcher, B. Kao, and S. Kasif, "Modeling splice sites with Bayes networks," *Bioinformatics*, vol. 16, pp. 152–158, 2000.
[11] M. Pertea, X. Lin, and S. L. Salzberg, "GeneSplicer: A new computational method for splice site prediction," *Nucleic Acids Res.*, vol. 29, pp. 1185–1190, 2001.
[12] J. C. Venter *et al.*, "The sequence of the human genome," *Science*, vol. 291, pp. 1304–1351, 2001. Erratum in June 5; vol. 292, pp. 1838, 2001.
[13] T. A. Thanaraj, "Positional characterization of false positives from computational prediction of human splice sites," *Nucleic Acids Res.*, vol. 28, pp. 744–754, 2000.
[14] C. B. Burge and S. Karlin, "Finding the genes in genomic DNA," *Curr. Opin. Struct. Biol.*, vol. 8, pp. 346–354, 1998.
[15] T. G. Dietterich and J. W. Shaving, Eds., *Readings in Machine Learning*. San Mateo, CA: Morgan Kaufmann, 1990.
[16] S. Rampone, "Recognition of splice-junctions on DNA sequences by BRAIN learning algorithm," *Bioinformatics*, vol. 14, pp. 676–684, 1998.
[17] ——, "Splice-junction recognition on gene sequences (DNA) by BRAIN learning algorithm," in *Proc. 1998 IEEE-INNS-ENNS International Joint Conf. Neural Networks (IJCNN'98-WCCI'98)*, 1998, pp. 774–779.
[18] R. S. Michalski, "A theory and methodology of inductive learning," *Artif. Intell.*, vol. 20, pp. 111–116, 1983.
[19] T. M. Mitchell, "Generalization as search," *Artif. Intell.*, vol. 18, pp. 203–226, 1982.
[20] D. Haussler, "Quantifying inductive bias: AI learning algorithms and Valiant's learning framework," *Artif. Intell.*, vol. 36, pp. 177–222, 1988.
[21] S. Rampone, "Linear codes interpolation from noisy patterns by means of a vector quantization process," *Comput. Math. Applicat.*, vol. 30, pp. 91–106, 1995.
[22] M. Carozza and S. Rampone, "An incremental multivariate regression method for function approximation from noisy data," *Pattern Recognit.*, vol. 34, pp. 179–186, 2001.
[23] ——, "Function approximation from noisy data by an incremental RBF network," *Pattern Recognit.*, vol. 32, pp. 2081–2083, 1999.
[24] ——, "Toward an incremental SVM for regression," in *Proc. IEEE-INNS-ENNS Int. Joint Conf. Neural Networks (IJCNN'2000) Neural Computing: New Challenges and Perspectives for the New Millennium*, 2000, pp. 405–410.
[25] C. M. Bishop, *Neural Networks for Pattern Recognition*. Oxford, U.K.: Clarendon, 1996.
[26] Z. Cataltepe, Y. S. Abu-Mostafa, and M. Magdon-Ismail, "No free lunch for early stopping," *Neural Comput.*, vol. 11, pp. 995–1009, 1999.
[27] P. Pollastro and S. Rampone, "HS3D, a dataset of homo sapiens splice regions, and its extraction procedure from a major public database," *Int. J. Modern Phys. C*, vol. 13, pp. 1105–1117, 2002.
[28] V. N. Vapnik, *Statistical Learning Theory*. New York: Wiley, 1998.
[29] G. G. Towell, M. O. Noordewier, and J. W. Shavlik. (1992) Primate splice-junction gene sequences (DNA) with associated imperfect domain theory [Online]. Available: http://www.datalab.uci.edu/data/mldb-sgi/data/
[30] M. O. Noordewier, G. G. Towell, and J. W. Shavlik, "Training knowledge-based neural networks to recognize genes in DNA sequences," in *Advances in Neural Information Processing Systems*. Denver, CO: Morgan Kaufmann, 1991, vol. 3, pp. 530–536.
[31] G. G. Towell, J. W. Shavlik, and M. W. Craven, "Constructive induction in knowledge-based neural networks," in *Proc. Eighth International Machine Learning Workshop*. Evanston, IL, 1991, pp. 213–217.
[32] G. G. Towell and J. W. Shavlik, "Interpretation of artificial neural networks: Mapping knowledge-based neural networks into rules," in *Advances in Neural Information Processing Systems*. Denver, CO: Morgan Kaufmann, 1992, vol. 4, pp. 977–984.
[33] S. Rampone, "BRAIN on Line,", http://brain.sci.unisannio.it, 2003.