Electronic Journal of Applied Statistical Analysis
EJASA (2012), Electron. J. App. Stat. Anal., Vol. 5, Issue 2, 256 – 270
e-ISSN 2070-5948, DOI 10.1285/i20705948v5n2p256
© 2012 Università del Salento – http://siba-ese.unile.it/index.php/ejasa/index

A FUZZIFIED BRAIN ALGORITHM FOR LEARNING DNF FROM INCOMPLETE DATA

Salvatore Rampone*(1), Ciro Russo(2)

(1)*Dipartimento di Scienze per la Biologia, la Geologia e l'Ambiente, Università degli Studi del Sannio, Benevento, Italy*
(2)*Departamento de Matemática - Instituto de Matemática, Universidade Federal da Bahia, Salvador, Bahia, Brasil*

Received 14 June 2011; Accepted 03 January 2012
Available online 14 October 2012

Abstract: *Aim of this paper is to address the problem of learning Boolean functions from training data with missing values. We present an extension of the BRAIN algorithm, called U-BRAIN (Uncertainty-managing Batch Relevance-based Artificial INtelligence), conceived for learning DNF Boolean formulas from partial truth tables, possibly with uncertain values or missing bits.*
Such an algorithm is obtained from BRAIN by introducing fuzzy sets in order to manage uncertainty. In the case where no missing bits are present, the algorithm reduces to the original BRAIN.

Keywords: *Boolean function, DNF, Learning algorithm, Missing Values*

1. Introduction

In many applications one must deal with data that have been collected incompletely. For example, in censuses and surveys, some participants may not respond to certain questions [19]; in email spam filtering, server information may be unavailable for emails from external sources [6]; in medical studies, measurements on some subjects may be partially lost at certain stages of the treatment [12]; in DNA analysis, gene-expression microarrays may be incomplete due to insufficient resolution, image corruption, or simply dust or scratches on the slide [22]; in sensing applications, a subset of sensors may be absent or fail to operate at certain regions [23].
Traditionally, data are often "completed" by ad hoc editing, such as case deletion and single imputation, where feature vectors with missing values are simply discarded or completed with

* E-mail: rampone@unisannio.it

specific values in the initial stage of analysis, before the main inference. Although analysis procedures designed for complete data become applicable after these edits, shortcomings are clear. For case deletion, discarding information is generally inefficient, especially when data are scarce. Secondly, the remaining complete data may be statistically unrepresentative. More importantly, even if the incomplete-data problem is eliminated by ignoring data with missing features in the training phase, it is still inevitable in the test stage since test data cannot be ignored simply because a portion of features are missing. For single imputation, the main concern is that the uncertainty of the missing features is ignored by imputing fixed values [21].

In this paper the general problem is posed as finding a Boolean function that extends a partially defined one. This type of problem is studied, for example, in learning theory [16, 20], where it is called *consistency problem*. In pattern recognition, a function separating two categories of data T and F is usually called a *discriminant function* (e.g., [13]). Apart from the existence of a solution for this problem (which is a trivial question, if the self-consistency of the data is assumed) it is desirable for such a solution to be presented in a canonical form (Boolean formulas in DNF or CNF, for example) and to be of minimum complexity [3]. It is well-known that such a problem, with a single positive given instance, is equivalent to a set covering problem [17, 5] and, therefore, NP-hard [10]. Nonetheless, it is possible to find a solution of approximately minimum complexity by applying generalized greedy methods (see, for example, [11, 9]).

Here, as a further complication of the problem described above, we assume that training data might not be complete — as previously introduced — and the values of some elements of a given data vector may not be available. Usually a set of data which includes missing bits, is called a *partially defined Boolean function with missing bits* [4].

In order to deal with training data with incompletely observed attributes, we generalize a machine learning algorithm, called *Batch Relevance-based Artificial INtelligence* (BRAIN) algorithm [17], for binary classification rules. This algorithm was originally conceived for recognizing *splice junctions* in human DNA (see also [18, 1]). Splice junctions are points on a DNA sequence at which "superfluous" DNA is removed during the process of protein synthesis in higher organisms [8]. The general method used in the algorithm is related to the STAR technique of Michalski [14], to the candidate-elimination method introduced by Mitchell [15], and to the work of Haussler [9].

Starting from the BRAIN algorithm, we extend it by using fuzzy sets [24], in order to infer a DNF formula that is consistent with a given set of data which may have missing bits.

The paper is structured as follows.

In Section 2 we will formally describe the problem, recall some basics about BRAIN and motivate the use of fuzzy sets.

In Section 3 we shall describe the algorithm U-BRAIN distinguishing — for the sake of clearness — three cases of increasing complexity.

In Section 4 we shall apply the algorithm and discuss its performances on a standard dataset [2]. We shall consider that missing bits in a dataset can appear for different reasons that can be essentially grouped in two cases. Some results are summarized in the tables contained in Appendix A.

Last, in Section 5, we will draw the conclusions.

2. Problem description and formalization

2.1 Learning problem

The problem of finding a Boolean function that extends a partially defined one can be posed as follows: given a pair of disjoint sets:

$$T, F \subseteq \{0,1\}^n$$

and a Boolean map:

$$f : G \to \{0,1\}$$

where $G = T \cup F$, such that $f[T] = \{1\}$ and $f[F] = \{0\}$, establish a Boolean function:

$$f^* : \{0,1\}^n \to \{0,1\}$$

such that:

$$f^*[T] = \{1\}, f^*[F] = \{0\}.$$

The elements of *T* and *F* are usually called, respectively, *positive* and *negative instances*. In [17] the author proposed an algorithm — called BRAIN (Batch Relevance-based Artificial INtelligence) — which infers a DNF Boolean formula in *n* variables of low syntactic complexity and consistent with a given set of positive and negative instances. In other words, given a subset $G \subseteq \{0,1\}^n$ and a function $f : G \to \{0,1\}$, BRAIN yields a function $f^* : \{0,1\}^n \to \{0,1\}$, expressed with a DNF formula of approximately minimum complexity, such that f^* coincide with *f* on *G*.
Instances for which *f* gives the value 1 will be called *positive*, those for which *f* gives 0 will be called *negative*. In order to better distinguish positive and negative instances, we shall denote by:

$$\mathbf{u}_i,\ i = 1,\ldots,p$$

the positive instances and by:

$$\mathbf{v}_j,\ j = 1,\ldots,q$$

the negative ones.
In order to build a formula consistent with the given data, BRAIN compares each given positive instance $\mathbf{u}_i$ with each negative one $\mathbf{v}_j$ and builds a family of sets S_{ij} — which are (crisp) subsets of the set $L = \{x_i, \bar{x}_i \mid i = 1,\ldots,n\}$ of all literals — as:

$$S_{ij} = \{x_k \mid u_{ik} = 1, v_{jk} = 0\} \cup \{\bar{x}_k \mid u_{ik} = 0, v_{jk} = 1\},$$

or, that is the same, defined by their respective characteristic, or membership, functions:

$$\begin{aligned} \chi_{ij}(x_k) &= \begin{cases} 1 \text{ if } u_{ik} = 1 \text{ and } v_{jk} = 0 \\ 0 \text{ otherwise} \end{cases} \\ \chi_{ij}(\bar{x}_k) &= \begin{cases} 1 \text{ if } u_{ik} = 0 \text{ and } v_{jk} = 1 \\ 0 \text{ otherwise} \end{cases} \end{aligned} \qquad \forall k = 1,\dots,n. \tag{1}$$

Each S_{ij} essentially represents a constraint that must be satisfied by the output formula; indeed it contains literals that are simultaneously positive in a positive instance and negative in a negative one. Therefore BRAIN, with a greedy approximation procedure, provides a formula which satisfies such conditions.

2.2 Adding uncertainty

Now we assume, in the same situation, that the given instances may contain "uncertain" values, i.e. for some elements of G we may not know all the coordinates. This may occur as a consequence of errors or combination of data from different sources or, also, as a consequence of the application of a lossy compression algorithm.

Here we formalize such a circumstance by representing G as a subset of $\left\{0,\frac{1}{2},1\right\}^n$, where 1/2 represents the uncertain values.

$$G \subseteq \left\{0,\frac{1}{2},1\right\}^n$$

Our aim, now, is to extend BRAIN in such a way that the new algorithm would infer a function $f^* : \{0,1\}^n \to \{0,1\}$, expressed by a DNF formula of minimum complexity, that is consistent with f in the following sense:

- f^* coincide with f on $G \cap \{0,1\}^n$,
- for any vector $\mathbf{u} = (u_1, \dots, u_n) \in G \cap \left\{0,\frac{1}{2},1\right\}^n$, there exists an element $\mathbf{u}' = (u_1', \dots, u_n') \in \{0,1\}^n$ such that $u_i = u_i'$ for any $u_i \neq 1/2$ and $f^*(\mathbf{u}') = f(\mathbf{u})$.

In what follows, by an *instance* we shall mean an element of $\left\{0,\frac{1}{2},1\right\}^n$ and, if it is in $\{0,1\}^n$, it will be called *certain*.

As we shall see, in our extensions of BRAIN, the sets S_{ij} shall be fuzzy subsets of L.

In writing DNF formulas, throughout the paper we shall often omit the conjunction symbol ∧ and we shall denote by an overlined letter the negation of a variable, in order to make formulas more compact and readable; thus we may write, for instance, $x_1\bar{x}_3 \vee \bar{x}_2 x_4$ instead of $(x_1 \wedge \neg x_3) \vee (\neg x_2 \wedge x_4)$.

3. Algorithm

3.1 Uncertainty reduction

The first step of our algorithm aims at reducing as much as possible the number of missing bits. Indeed we may have instances in which missing bits can be recovered. Assume, for example, to have a positive instance with a single missing bit and a negative one with no missing bits that coincide on all of their certain values. In this case, since we assume the given instances to be self-consistent, the missing bit in the positive instance must give the only possible difference between the two instances, whence it must be the negation of the corresponding bit in the negative instance.
Therefore, as a first step, we shall update the data as follows.
Let (**u**, **v**) be a pair of instances, one positive and the other one negative, and assume that there exists a unique coordinate $k \in \{1,...,n\}$ such that

- $u_r = v_r \in \{0,1\}$ for all $r \neq k$,
- one of the two k-th components u_k, v_k is certain and the other one is not.

In this case we must have $u_k = \bar{v}_k$, hence we update the instance containing an uncertainty by substituting its k-th component with the negation of the k-th component of the certain instance.
This substitution shall be made whenever possible and the reduction iterated until no more substitutions of this kind are possible.

Example 1. Let $G = \{\mathbf{u}_1, \mathbf{v}_1, \mathbf{v}_2\}$, with:

$\mathbf{u}_1 = (1, 1/2, 0, 0)$, $\mathbf{v}_1 = (1,1,0,0)$, $\mathbf{v}_2 = (1,0,0,1/2)$.

From the comparison of $\mathbf{u}_1$ and $\mathbf{v}_1$ we obtain that the only Boolean value for u_{12} that keeps the set of instances self-consistent is $\bar{v}_{12}$, i.e. 0; so we set $\mathbf{u}_1' = (1,0,0,0)$.
Once we update the set of instances by substituting $\mathbf{u}_1$ with $\mathbf{u}_1'$, we can apply the same argument to $\mathbf{u}_1'$ and $\mathbf{v}_2$ thus obtaining $v'_{24} = 1$and $\mathbf{v}_2' = (1,0,0,1)$. Therefore the new set of instances is $G' = \{\mathbf{u}_1', \mathbf{v}_1', \mathbf{v}_2'\}$.
After the description of the algorithm we will show, in Example 6, the importance of the reduction.

3.2 Repetition deletion

It is possible that the set of instances contains redundant information, i.e. there are some instances that are repeated one or more times, either since the beginning or as a result of the reduction step.
Such redundancy is removed by keeping each certain instance just once and deleting all the repetitions.

3.3 Membership function

We will consider the fuzzy subsets S_{ij} of L defined by the characteristic function χ_{ij}:

$$\chi_{ij}(x_k) = \begin{cases} 1 & \text{if } u_{ik} = 1 \text{ and } v_{jk} = 0 \\ \left(\frac{1}{2}\right)^{p+q} & \text{if } u_{ik} > v_{jk} \text{ and } \frac{1}{2} \in \{u_{ik}, v_{jk}\} \\ \left(\frac{1}{2}\right)^{p+q+1} & \text{if } u_{ik} = v_{jk} = \frac{1}{2} \\ 0 & \text{otherwise} \end{cases}$$

$$\chi_{ij}(\bar{x}_k) = \begin{cases} 1 & \text{if } u_{ik} = 0 \text{ and } v_{jk} = 1 \\ \left(\frac{1}{2}\right)^{p+q} & \text{if } u_{ik} < v_{jk} \text{ and } \frac{1}{2} \in \{u_{ik}, v_{jk}\} \\ \left(\frac{1}{2}\right)^{p+q+1} & \text{if } u_{ik} = v_{jk} = \frac{1}{2} \\ 0 & \text{otherwise} \end{cases} \qquad \forall k = 1,\dots,n. \qquad (2)$$

Such a membership function is thought of in such a way that the certain values have a much prominent role w.r.t. the missing ones in a comparison between two instances. However, as we shall see later on, even uncertain information may gain relevance if the certain one is not sufficient.

3.4 One-to-one

For the sake of clearness, we shall distinguish three situations, where we are given, respectively:

- a positive instance and a negative one,
- a positive instance and several negative ones,
- several positive and negative instances.

First let us find a DNF formula that is consistent with a positive instance $\mathbf{u}_i$ and a negative one $\mathbf{v}_j$. A term t made true by $\mathbf{u}_i$ and false by $\mathbf{v}_j$ must include at least one variable not assigned in the same way in the two instances. By the assumption of self-consistency of our instances, χ_{ij} is not constantly equal to 0. Now, if l is a literal such that $\chi_{ij}(l) = \max \chi_{ij}$, $f^* = l$ is consistent with the given set of instances and of minimum complexity. If there is more than one literal satisfying such a condition, we choose the positive one with the lowest index — if any — or, otherwise, the negative one with the lowest index. By the definition of the membership function (2), the certain differences are necessarily privileged.

Example 2. Let $\mathbf{u}_1 = (1, 1, 0, 1/2, 1)$ and $\mathbf{v}_1 = (1, 0, 0, 1, 0)$. Then:

$$\chi_{11}(l) = \begin{cases} 1 & \text{for } l \in \{x_2, x_5\} \\ 1/4 & \text{for } l = \bar{x}_4 \\ 0 & \text{otherwise} \end{cases} .$$

Therefore $f^* = x_2$.

Example 3. Let $\mathbf{u}_1 = (1, 1/2, 0)$ and $\mathbf{v}_1 = (1/2, 0, 0)$. Then

$$\chi_{11}(l) = \begin{cases} 1/4 & \text{for } l \in \{x_1, x_2\} \\ 0 & \text{otherwise} \end{cases}.$$

Therefore $f^* = x_1$.
It is worth noticing that the selection of x_1 in the last example implies a decision on the negative instance, which is implicitly assumed to be $v_1' = (0, 0, 0)$.

3.5 One-to-many

Now assume we have a positive instance $\mathbf{u}_i$ and q negative ones $\mathbf{v}_{1, \dots,} \mathbf{v}_q$. In this case, the problem is equivalent to a set covering one and we follow the BRAIN algorithm or, in a different version, the set covering approximation improvement presented in [1]. Recalling that, for a fuzzy subset F of a given set S, the fuzzy cardinality is defined as:

$$\#(F) = \sum_{x \in S} \chi_F(x), \tag{3}$$

we define the *relevance* of a literal l_k in S_{ij} as:

$$R_{ij}(l_k) = \frac{\chi_{ij}(l_k)}{\#(S_{ij})}, \tag{4}$$

where χ_{ij} is the membership function of S_{ij}. Then we set:

$$R_i(l_k) = \frac{1}{q} \sum_{j=1}^{q} R_{ij}(l_k). \tag{5}$$

So, given the S_{ij} sets, we proceed as follows.

1. Compute the relevance of each literal (the relevance of the opposite of the previously selected literals is always set to zero).
2. Choose the literal l with the highest relevance.
3. Erase the sets containing l and the occurrences of $\bar{l}$ inside the sets where they appear.
4. Repeat steps 1 to 3 until there are no more sets.

The resulting formula will be the conjunction of the chosen literals $l_{i_1}, \dots, l_{i_k}$ and is obviously consistent with $\mathbf{u}_1$. On the other hand, for each $j \le q$, $\mathbf{v}_j$ either contains a certain value which is in contradiction with one of such literals, or contains an uncertain value that is assumed, by the algorithm, contradicting a literal of f^*. Then the $\mathbf{v}_j$'s are indeed negative instances of f^*.

Example 4. Let $\mathbf{u}_1 = (1, 0, 0)$, $\mathbf{v}_1 = (0, 1, 1)$, $\mathbf{v}_2 = (1, 0, 1)$, and $\mathbf{v}_3 = (1, 1/2, 1)$. The membership functions defining the S_{1j} sets are

$$\chi_{11}(l)=\begin{cases}1 & \text{for } l = x_1, \bar{x}_2, \bar{x}_3 \\ 0 & \text{otherwise}\end{cases},$$

$$\chi_{12}(l)=\begin{cases}1 & \text{for } l = \bar{x}_3 \\ 0 & \text{otherwise}\end{cases}, \text{and}$$

$$\chi_{13}(l)=\begin{cases}1 & \text{for } l = \bar{x}_3 \\ 1/32 & \text{for } l = \bar{x}_2 \\ 0 & \text{otherwise}\end{cases}.$$

The maximum resulting relevance is $R_1(\bar{x}_3)$ and $\chi_{1j}(\bar{x}_3) > 0$ for all j; it follows $f^* = \bar{x}_3$.
Example 5. Let $\mathbf{u}_1 = (1, 0, 1/2, 1)$, $\mathbf{v}_1 = (0, 1, 1, 1)$, and $\mathbf{v}_2 = (1, 0, 1, 0)$. The S_{1j}'s are defined by:

$$\chi_{11}(l)=\begin{cases}1 & \text{for } l \in \{\mathrm{x}_1, \bar{x}2\} \\ 1/8 & \text{for } l = \bar{x}_3 \\ 0 & \text{otherwise}\end{cases}, \text{and}$$

$$\chi_{12}(l)=\begin{cases}1 & \text{for } l = x_4 \\ 1/8 & \text{for } l = \bar{x}_3 \\ 0 & \text{otherwise}\end{cases},$$

hence $\max R_1 = R_1(x_4)$. Once we choose x_4, we erase S_{12}, which is the only set where the membership function of x_4 is not 0. Then we choose x_1 in S_{11} and the resulting formula is: $f^* = x_4 \wedge x_1$.

3.6 Many-to-many

Let us now consider the case of p positive instances and q negative ones:

$u_1, ..., u_p, v_1, ..., v_q$.

A consistent DNF formula will be a disjunction of a set of conjunction (or product) terms, i.e. of a set of conjunctions of literals. Each positive instance shall satisfy at least one product term and none of the terms shall be satisfied by any of the negative instances. So, for each pair ($\mathbf{u}_i$, $\mathbf{v}_j$), there exists a term that is satisfied by $\mathbf{u}_i$ and, like all the other terms, is not satisfied by $\mathbf{v}_j$; this implies that such a term must contain a variable that is not assigned in the same way in $\mathbf{u}_i$ and $\mathbf{v}_j$, which means that it contains a literal of S_{ij}. Then, for all i = 1, ..., p and j = 1, ..., q, there exists a term in f^* containing a literal in S_{ij}.
In this case we have the pq sets S_{ij} that we collect, for our convenience, as:

$$S_i = \{S_{i1}, ..., S_{iq}\}, \quad i = 1, ..., p,$$

with the associated relevance functions R_i defined by (5) and the total relevance defined by:

$$R(l_k) = \frac{1}{p}\sum_{i=1}^{p} R_i(l_k) = \frac{1}{pq}\sum_{i=1}^{p}\sum_{j=1}^{q} R_{ij}(l_k). \tag{6}$$

The U-BRAIN algorithm uses the relevance as a greedy criterion to build DNF function terms. Starting from an empty term, it selects the most relevant variable and add it to the term, erasing all the satisfied constraints S_{ij} and all the incompatible S_i sets, until at least a set S_i of constraints is satisfied (and therefore empty). This greedy choice aims at the same time both at covering the greatest number of positive instances and at selecting the less possible number of variables. The process is iterated until there are no more constraints.

3.7 Negative Instances Updating

Each time a term is produced, the implicit choices over the uncertain components of the negative instances, if any, must be made clear in order to avoid contradiction with the terms to be generated in the following. So for each negative instance $\mathbf{v}_j$, if a particular choice of the uncertain values can satisfy the last generated term m, i.e. there exists an element $\mathbf{v}' = (v_1', ..., v_n') \in \{0,1\}^n$ such that $v_{j_i} = v_i'$ for any $v_{j_i} \neq 1/2$ and $m(\mathbf{v}') = 1$, then the uncertain element of lowest index of $\mathbf{v}_j$ is set to a certain value contradicting the truthfulness of the term. Once the negative instances have been updated, the algorithm checks the self-consistency of the new set of instances and, in case a consistency issue arises, it stops. The whole U-BRAIN algorithm can be formally depicted as follows.

U-BRAIN algorithm

```
1. Input:
     ▪ the number of variables: $n$,
     ▪ the set of training instances: $G = \{\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_p, \mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_q\}$.
   Initialization: set $f^* = \phi$.
2. While there are positive instances in G
     2.0 Uncertainty reduction, repetition deletion.
     2.1 $S_{ij}$ sets: Build from $G$ the $S_{ij}$ sets and collect them in $S_i = \{S_{ij}\}_{j=1}^{q}$ for all $i = 1, ..., p$.
     2.2 Start a new term: Set $m = \phi$.
     2.3 Build the term: While there are $S_{ij}$ sets
          2.3.1 Relevances: Compute the relevance $R(l_k)$, for $l_k \in \{x_k, \bar{x}_k\}$, $k = 1,...,n$;
          2.3.2 Add variable: Select $l_k$ such that $R(l_k)$ is equal to max $R$, $m \leftarrow m \wedge l_k$;
          2.3.3 Update sets: Erase the $\bar{l}_k$ occurrences, if any, in the $S_i$ sets where also $l_k$
                appears, erase the $S_i$ sets in which $l_k$ does not occur, erase the $S_{ij}$ sets where $l_k$
                appears.[1]
     2.4 Add the term: $f^* \leftarrow f^* \vee m$.
     2.5 Update positive instances: Erase from $G$ all the positive instances satisfying $m$.
     2.6 Update negative instances: Update uncertain values, for all $\mathbf{v}_j \in G$.
3. Output: $f^*$.
```

[1] By a literal that *appears* or *occurs* in an S_{ij} we mean that its value under the ij-th membership function is > 0; a literal appears in S_i if it appears in S_{ij} for some j.

It is worth noticing that, if the given set of instances is contained in $\{0,1\}^n$, then the algorithm proposed coincides with the original BRAIN.
As we anticipated in subsection 3.1, the following example shows the importance of the reduction step.

Example 6. Let $\mathbf{u}_1 = (1,0,0)$, $\mathbf{u}_2 = (1/2,1,0)$ be positive instances, and $\mathbf{v}_1 = (1,1/2,0)$ a negative instance.
If we apply the reduction, first $\mathbf{v}_1$ shall be substituted by $\mathbf{v}_1' = (1,1,0)$ and then $\mathbf{u}_2$ will be substituted by $\mathbf{u}_2' = (0,1,0)$. Then the algorithm will give us the formula $\bar{x}_1 \vee \bar{x}_2$ which is consistent with the given set of instances.
If we do not apply the reduction, the algorithm will give back the formula $\bar{x}_2 \vee x_2$ which is a tautology and, therefore, cannot be consistent with our training set since the latter contains a negative instance.

4. Performance evaluations

We shall distinguish two cases, which essentially depend on the reason of uncertainty.
As we anticipated in Section 1, missing bits may have different origin which, after all, can be taken back to two main classes. More precisely, we may have uncertain values due to the following circumstances:

1. *random uncertainties*, due e.g. to the presence of some noise or errors in data storage, transmission and/or retrieving,
2. *trustworthy uncertainties*, occurring, for instance, when missing bits have not been provided because some of the sources recognize them as not relevant.

These two aspects of incomplete information appear to naturally correspond to fuzzy logic and probability respectively [7].
Now, since our aim is to treat both these situations, the best approach seems to require both fuzzy logic and probability in some sense. So, in the definition of the χ_{ij} sets, we privileged the logical viewpoint assuming that the missing bit is a vague information that must be used only when no better one is present. On the other hand, the possibility that missing bits come from randomly distributed errors is taken care of at a subsequent stage, namely when we define the *relevance* of a literal (subsections 3.5 and 3.6). In fact the total relevance defined by (6) is indeed a probability distribution and its definition by means of the fuzzy cardinality (3) gives some weight back to missing bits when necessary.
So the algorithm has been tested using the standard "Zoo" dataset in [2], with the introduction of random and trustworthy missing bits alternatively.
The dataset contains one hundred one animals divided in seven types (mammal, bird, reptile, fish, amphibian, insect, invertebrate) and described by means of fifteen Boolean attributes (hair, feathers, eggs, milk, airborne, aquatic, predator, toothed, backbone, breathes, venomous, fins, tail, domestic, cat-size) and a numeric one (legs = 0, 2, 4, 5, 6, or 8) that we made into five further Boolean values (legs = 2, legs = 4, legs = 5, legs = 6, legs = 8), with legs = 0 represented by assigning the value 0 to all of them.
The tests have been performed by considering the animals of a given type as positive instances and all the others as negative ones, and then computing f^* by means of U-BRAIN. Eventually,

missing bits have been randomly distributed inside the data, up to 50% of the total number of bits; analogously, the trustworthy uncertainties have been distributed among the data, up to 50% of the total number of bits.

We use two indicators of the performances. The *Average Error Number AEN* is the average number of instances in the dataset erroneously classified. The *error rate R* is the ratio between the error number and the dataset size.

In Table 1 such indicators are reported while in the Appendix (Tables 2 to 8) we present some results of such performances; each "E" column indicates the number of errors introduced by the algorithm. More precisely, it shows the number of instances that the resulting formula changes from positive to negative or vice-versa.

Table 1. Overall Average Error Number and Error Rate (Zoo Dataset) varying the missing bit percentage in the Random and Trustworthy cases.

% Missing bits	Random uncertainties		Trustworthy uncertainties	
	AEN	R	AEN	R
10	0.36	0.003	0.57	0.006
20	0.93	0.009	0.28	0.003
30	1.00	0.010	0.28	0.003
40	1.50	0.015	0	0
50	5.43	0.054	0	0

In the case of random missing bits, the average error percentage is below 1/10 of the percentage of missing bits, showing a highly reliable behaviour. To what extent the case of trustworthy uncertainties, as Table 1 shows, the results obtained by U-BRAIN show that the number of errors decreases as the number of missing bits increases. This circumstance suggests the possibility of applying U-BRAIN for the reconstruction of highly compressed data, which is a motivation for future works.

5. Concluding remarks

The problem addressed in this paper is to find, given a partially defined Boolean function with missing bits, a Boolean formula in disjunctive normal form, of approximately minimum complexity, which is consistent with the given data.

The solutions proposed is a learning algorithm — obtained as an extension of the BRAIN algorithm by means of the introduction of fuzzy sets — inferring Boolean formulas from incomplete instances. The conjunctive terms of the formula are computed in an iterative way by identifying, from the given data, a family of sets of conditions that must be satisfied by all the positive instances and violated by all the negative ones; such conditions allow the computation of a relevance coefficient for each attribute (literal).

The proposed approach introduces the possibility of managing uncertain values by considering the aforementioned sets of conditions as fuzzy sets, whose characteristic functions play a significant role in the relevance coefficients.

The new algorithm appears to have low error rates and maintains the polynomial computational complexity of the original BRAIN algorithm.

References

[1]. Aloisio A., Izzo V., Rampone S. (2007). VLSI implementation of greedy-based distributed routing schemes for ad hoc networks. *Soft Computing*, **11**(9), 865–872.

[2]. Asuncion A., Newman D.J (2007). UCI Machine Learning Repository, University of California, School of Information and Computer Science, Irvine, CA. http://www.ics.uci.edu/~mlearn/MLRepository.html.

[3]. Blumer A., Ehrenfeucht A., Haussler D.,Warmuth M.K. (1987). Occam's Razor. *Information Processing Letters*, 24, 377–380.

[4]. Boros E., Ibaraki T., Makino K. (1999). Logical analysis of binary data with missing bits. *Artificial Intelligence*, 107, 219–263.

[5]. Cormen T.H., Leiserson C.H., Rivest R.L. (1990). *Introduction to algorithms.* Cambridge: MIT Press.

[6]. Dick U., Haider P., Scheffer T. (2008). Learning from Incomplete Data with Infinite Imputations, in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland.

[7]. Dubois D., Prade H. (1993). Fuzzy sets and probability: misunderstandings, bridges and gaps, in *Second IEEE International Conference on Fuzzy Systems*.

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

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

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

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

[12]. Ibrahim J. (1990). Incomplete data in generalized linear models. Journal of the American Statistical Association, 85, 765–769.

[13]. Mangasarian O.L., Setiono R., Wolberg W.H. (1990). Pattern recognition via linear programming: Theory and applications to medical diagnosis, in *Large-Scale Numerical Optimization*, T.E. Coleman and Y. Li eds., SIAM, Philadelphia, PA, 22–30.

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

[15]. Mitchell T.M. (1982). Generalization as search. *Artif. Intell.*, 18, 203–226.

[16]. Pitt L., Valiant L.G. (1988). Computational limitations on learning from examples. *J. ACM*, 35, 965–984.

[17]. Rampone S. (1998), Recognition of spline-junctions on DNA sequences by BRAIN learning algorithm. *Bioinformatics Journal*, **14**(8), 676–684.

[18]. Rampone S. (2004). An Error Tolerant Software Equipment For Human DNA Characterization. *IEEE Transactions on Nuclear Science*, **51**(5), 2018–2026.

[19]. Rubin D. B. (1987). *Multiple Imputation for Nonresponse in Surveys.* John Wiley and Sons Inc.

[20]. Valiant L.G. (1984). A theory of the learnable. *Comm. ACM*, 27, 1134–1142, 1984.

[21]. Wang C., Liao X., Carin L., Dunson D. B. (2010). Classification with Incomplete Data Using Dirichlet Process Priors. *JMLR*, 11(Dec), 3269–3311.

[22]. Wang X., Li A., Jiang Z., Feng H. (2006). Missing value estimation for DNA microarray gene expression data by support vector regression imputation and orthogonal coding scheme. *BMC Bioinformatics*, 7, 32.

[23]. Williams D., Carin L. (2005). Analytical kernel matrix completion with incomplete multi-view data, in: *Proceedings of the International Conference on Machine Learning (ICML), Workshop on Learning with Multiple Views*, 80–86.

[24]. Zadeh L.A. (1965). Fuzzy sets. *Information and Control*, **8**(3), 338–353.

Appendix A. Test tables

Table 2. Type 1 – Mammal.

$f = x_4$						
	Random (1)		Random (2)		Trustworthy	
% Missing bits	f^*	E	f^*	E	f^*	E
10	x_4	0	x_4	0	x_4	0
20	x_4	0	x_4	0	x_4	0
30	x_4	0	x_4	0	x_4	0
40	x_4	0	x_4	0	x_4	0
50	$\bar{x}_3\bar{x}_{11} \vee x_{16}x_{19}$	1	$x_1x_8 \vee \bar{x}_3x_8$	1	x_4	0

Table 3. Type 2 – Bird.

$f = x_2$						
	Random (1)		Random (2)		Trustworthy	
% Missing bits	f^*	E	f^*	E	f^*	E
10	x_2	0	x_2	0	x_2	0
20	x_2	0	x_2	0	x_2	0
30	x_2	0	x_2	0	x_2	0
40	$x_{17}\bar{x}_8$	0	$x_{17}\bar{x}_1$	0	x_2	0
50	$x_{17}\bar{x}_4$	0	x_2	0	x_2	0

Table 4. Type 3 – Reptile.

$f = \bar{x}_6\bar{x}_1x_8 \vee x_{11}\bar{x}_3x_6 \vee x_{16}\bar{x}_8\bar{x}_6$						
	Random (1)		Random (2)		Trustworthy	
% Missing bits	f^*	E	f^*	E	f^*	E
10	$\bar{x}_1x_8\bar{x}_{12}x_{18}\bar{x}_6 \vee x_{11}x_8x_{18} \vee x_{16}\bar{x}_8\bar{x}_1$	2	$x_{11}x_{18}\bar{x}_{20} \vee x_{16}\bar{x}_1\bar{x}_6$	2	$x_{15}x_8 \vee x_{16}\bar{x}_8\bar{x}_6$	4
20	$x_{11}x_8\bar{x}_{16} \vee x_{16}\bar{x}_1x_{20}$	4	$\bar{x}_1x_8\bar{x}_{12}x_7 \vee x_{16}\bar{x}_8x_9$	4	$\bar{x}_1\bar{x}_6x_8 \vee \bar{x}_1x_{15}x_{16} \vee x_{11}\bar{x}_3x_6$	1
30	$\bar{x}_1x_8\bar{x}_{12} \vee x_{20}\bar{x}_8\bar{x}_7$	7	$\bar{x}_1\bar{x}_6x_9\bar{x}_2 \vee x_{11}x_{18}\bar{x}_{12}$	1	$\bar{x}_1\bar{x}_6x_8 \vee \bar{x}_1x_{15}x_{16} \vee x_{11}\bar{x}_3x_6$	1
40	$\bar{x}_1x_8x_{11} \vee x_{16}x_3\bar{x}_6$	3	$\bar{x}_1x_{16} \vee \bar{x}_4x_8x_{17}$	8	$\bar{x}_6\bar{x}_1x_8 \vee x_{11}\bar{x}_3x_6 \vee x_{16}\bar{x}_8\bar{x}_6$	0
50	$\bar{x}_4x_8x_7 \vee \bar{x}_8x_{20}$	20	$\bar{x}_5x_3x_{10} \vee x_{11}\bar{x}_3$	14	$\bar{x}_1\bar{x}_6x_8 \vee x_{11}\bar{x}_3x_6 \vee x_{16}\bar{x}_8\bar{x}_6$	0

Table 5. Type 4 – Fish.

$f = x_{12}x_3$						
	Random (1)		Random (2)		Trustworthy	
% Missing bits	f^*	E	f^*	E	f^*	E
10	$\bar{x}_{10}x_8x_3$	0	$x_{12}x_3$	0	$x_{12}x_3$	0
20	$x_{12}x_3$	0	$\bar{x}_{10}x_{12}$	0	$x_{12}x_3$	0
30	$\bar{x}_{10}x_{18}$	1	$x_{12}\bar{x}_4$	0	$x_{12}x_3$	0
40	$\bar{x}_{10}x_{18}$	1	$x_{12}\bar{x}_4$	0	$x_{12}x_3$	0
50	$\bar{x}_{10}x_9$	1	$x_{12}\bar{x}_{20} \vee x_{12}\bar{x}_5$	5	$x_{12}x_3$	0

Table 6. Type 5 – Amphibian.

$f = x_{16}x_6x_8x_3$						
	Random (1)		Random (2)		Trustworthy	
% Missing bits	f^*	E	f^*	E	f^*	E
10	$x_{16}x_6x_8$	1	$x_{16}x_6\bar{x}_{20}$	1	$x_6x_{16}x_8x_3$	0
20	$x_6x_{10}\bar{x}_4\bar{x}_{17}$	0	$\bar{x}_{18}x_9x_3 \vee x_6x_{16}x_{10}$	2	$x_6x_{16}x_8x_3$	0
30	$x_{16}x_{16}x_9$	2	$x_{16}\bar{x}_1x_8$	1	$x_6x_{16}x_8x_3$	0
40	$x_{16}x_{16}x_8$	1	$x_{16}x_3x_8$	1	$x_6x_{16}x_8x_3$	0
50	$\bar{x}_{18}x_9$	7	$x_8\bar{x}_1x_{18} \vee x_{16}\bar{x}_1x_8$	19	$x_6x_{16}x_8x_3$	0

Table 7. Type 6 – Insect.

$f = x_{14}x_{10}$						
	Random (1)		Random (2)		Trustworthy	
% Missing bits	f^*	E	f^*	E	f^*	E
10	$x_{14}\bar{x}_6$	0	$x_{14}\bar{x}_6$	0	$x_{14}x_{10}$	0
20	$x_{14}\bar{x}_7$	0	$\bar{x}_9x_{14}x_{10}$	0	$x_{14}x_{10}$	0
30	$x_{14}x_{10}$	1	$x_{14}x_{10}$	0	$x_{14}x_{10}$	0
40	$\bar{x}_8\bar{x}_2\bar{x}_7 \vee x_5\bar{x}_2\bar{x}_9$	1	$\bar{x}_9\bar{x}_7$	0	$x_{14}x_{10}$	0
50	$\bar{x}_9x_{10}x_{14}$	1	$x_{14}\bar{x}_7$	5	$x_{14}x_{10}$	0

Table 8. Type 7 – Invertebrate.

$f = \bar{x}_9\bar{x}_{14} \vee \bar{x}_{10}x_{14}$						
	Random (1)		Random (2)		Trustworthy	
% Missing bits	f^*	E	f^*	E	f^*	E
10	$\bar{x}_9\bar{x}_{10} \vee \bar{x}_9\bar{x}_{14}$	0	$\bar{x}_9\bar{x}_{14} \vee \bar{x}_9x_6$	0	$\bar{x}_9\bar{x}_{14} \vee \bar{x}_{10}x_{14}$	0
20	$\bar{x}_9\bar{x}_{14} \vee x_{14}\bar{x}_{10}$	0	$\bar{x}_9\bar{x}_5$	2	$\bar{x}_9\bar{x}_{14} \vee x_{14}x_6$	1
30	$\bar{x}_9x_6 \vee \bar{x}_9\bar{x}_1x_7 \vee \bar{x}_9\bar{x}_1\bar{x}_5$	2	$\bar{x}_9\bar{x}_{14} \vee x_{14}\bar{x}_{10}$	0	$\bar{x}_9\bar{x}_{14} \vee x_{14}\bar{x}_{10}$	0
40	$\bar{x}_9\bar{x}_{10} \vee \bar{x}_9\bar{x}_{14}$	0	$\bar{x}_9\bar{x}_{14} \vee x_{14}x_7$	1	$\bar{x}_9\bar{x}_{14} \vee x_{14}\bar{x}_{10}$	0
50	$\bar{x}_9x_7$	3	$\bar{x}_9\bar{x}_1$	4	$\bar{x}_9\bar{x}_{14} \vee x_{14}\bar{x}_{10}$	0