Binary Classification With Hypergraph Case-Based Reasoning

Alexandre Quemy
IBM Software Lab
Cracow, Poland
Faculty of Computing, Poznań University of Technology
Poznań, Poland
aquemy@pl.ibm.com

ABSTRACT

Binary classification is one of the most common problem in machine learning. It consists in predicting whether a given element is of a particular class. In this paper, a new algorithm for binary classification is proposed using a hypergraph representation. Each element to be classified is partitioned according to its interactions with the training set. For each class, the total support is calculated as a convex combination of the *evidence* strength of the element of the partition. The evidence measure is pre-computed using the hypergraph induced by the training set and iteratively adjusted through a training phase. It does not require structured information, each case being represented by a set of *agnostic information* atoms. Empirical validation demonstrates its high potential on a wide range of well-known datasets and the results are compared to the literature. The time complexity is given and empirically validated. Its capacity to provide good accuracy with few examples is also studied.

1 INTRODUCTION

In many real-life situations, one tries to take a decision based on previous *similar* situations. Each situation is described by a certain amount of information, either collected by an expert according to the relevance of the information, or automatically by some sensors or algorithms. Those situations share similarities on which to make analogies or counter-examples in order to take a new decision. Conversely, in general, if two situations do not share any common characteristic, then they are totally independent, i.e. it is impossible to infer one's outcome from the other one. The purpose of supervised machine learning algorithms is to exploit the available information and interactions between past cases or examples in order to build a model or infer the key rules to take correct decisions.

Due to the large variety of concrete situations that can be reduced to binary classification, it is one of the most studied problems in machine learning. In this paper, we investigate the problem of binary prediction under a supervised setting, i.e. given a history of previous situations labeled with the correct output.

This paper contributes to binary classification with a new algorithm called Hypergraph Case-Based Reasoning (HCBR). The idea is to create a hypergraph where each element of the training set is a hyperedge and vertices are represented by the features describing the elements. The intersections between edges create a partition for each case, and we model the support for each class as a convex combination of the elements of this partition. Each of those elements is valued according to its importance w.r.t. to the set of all the hyperedges it belongs to and their respective labels.

The well-known "no free lunch" theorem states that there is no unique algorithm that can outperform all the others in supervised learning. However, the case description and representation play a preponderant role in the performances but the algorithms are very often constrained by being designed to a specific representation that may not be suitable for a given problem or user data. The proposed method is totally agnostic about the representation[1] and does not require structured representations. For instance, it can work on atomic representations such as Bag-of-Word representations for texts where an atom is a single word (or n-gram). In this case, two elements would not have the same number of elements and it may be hard to properly define the case domain.

The plan of the paper is as follows: in Section 3 the problem of binary classification is formalized in the particular case of a finite countable set of information. The contribution of this paper is divided into two parts: Section 4 defines the HCBR algorithm and Section 5 presents empirical results on 8 datasets from the UC Irvine Machine Learning Repository (UCI)[2] and the LIBSVM[3]. The paper ends in Section 6 with a discussion about the results, future work, and possible improvements.

2 CLASSIFICATION AND HYPERGRAPH

Hypergraphs generalize graphs and can be used to represent higher order relations while graphs are limited to binary relations. A hypergraph is defined by a set of vertices and a collection of hyperedges where each hyperedge is a subset of this set of vertices. Therefore, a graph is a special case of hypergraph for which each hyperedge contains only two vertices. We will formally introduce hypergraphs in Section 4.1. Recently hypergraphs have been used as data representation, and some classification algorithms on hypergraph have been proposed. A vast majority of approaches models the objects to classify as the set of vertices and constructs the hyperedges as the representation of a metric. This conventional approach is known as *neighborhood-based* hypergraph. The metric relies on some assumptions on the data or a specific representation (e.g. Zernike moment and histogram of oriented gradient (HOG) to measure the distance between images in [16]) and for each vertex, a hyperedge is created to represent its k-nearest neighbors [12]. The problem of classification on hypergraph consists in labeling some unlabeled vertices given a training set such that all vertices in a same hyperedge have the same label. As all the vertices are known a priori, the problem is part of transductive learning. To learn the labels, the standard approach is to minimize a cost function based on a hypergraph equivalent of a graph Laplacian [16, 25] with a structural risk:

$$C(x) = x^t \Delta x + \mu ||x - y||^2 \tag{1}$$

[1]With the exception that the current version must work on discrete information.
[2]http://archive.ics.uci.edu/ml/index.php
[3]https://www.csie.ntu.edu.tw/ cjlin/libsvmtools/datasets/binary.html

where Δ is the hypergraph Laplacian, $\mu > 0$ a regularization factor and $||.||$ a norm. The vector y represents the initial labels for all vertices with $y_i = 0$ for unlabeled vertices, a negative (resp. positive) value for label -1 or 1.

On the contrary, the method proposed in this paper models the elements to classify as the hyperedges and the vertices as the different components of those elements. As far as we know, there is no previous work that uses this modeling choice. In addition, it does not require knowing all the elements before building the model: our approach is inductive. If the literature focus on the method to build the hyperedges, the proposed framework has a straightforward hypergraph construction and rather focus on model selection.

3 PROBLEM DEFINITION

Consider a countable finite space of information $\mathbb{F}$ and its σ-algebra $\mathcal{P}(\mathbb{F})$ defined as the powerset of $\mathbb{F}$. We call *case* an element of $\mathcal{P}(\mathbb{F})$. In practice, it is very likely that only a subset of $\mathcal{P}(\mathbb{F})$ may appear (for instance if two features encode two contradictory propositions or if every case has the same number of features). The real class for any case is defined by the unknown measurable mapping:

$$\begin{aligned} J\colon \mathcal{P}(\mathbb{F}) &\longrightarrow \{0,1\} \\ x &\longmapsto J(x) \end{aligned}$$

Given an example set X of n cases, the classification problem consists in minimizing the prediction errors for all the elements of $\mathcal{P}(\mathbb{F})$, that is to say finding a decision mapping $\bar{J}\colon \mathcal{P}(\mathbb{F}) \longrightarrow \{0,1\}$ such that:

$$\forall X \in \mathcal{P}(\mathbb{F})^n,\ \min_{\bar{J}} \sum_{x \in \mathcal{P}(\mathbb{F})} \mathbb{1}_{\{J(x) \neq \bar{J}(x)\}} \tag{2}$$

In other words, for any possible training set, we want to find an estimation of the exact mapping J and thus it is a functional problem. Notice that in this paper we do not consider *uncertainty*: if two situations are described the same in $\mathbb{F}$, then they have the same label.

4 HYPERGRAPH CASE-BASED REASONING

4.1 Representation And Projection

Before describing HCBR, we recall the definition of a hypergraph and induced hypergraph. For more results on hypergraphs, we refer the reader to [3].

Definition 4.1 (Hypergraph). A hypergraph is defined by $H = (V, X)$ with V a set of vertices, X the hyperedges such that $\forall x \in X,\ x \subseteq V$.

A hypergraph can be viewed as a collection of subsets X of a given set of vertices V. It is sometimes convenient to define a hypergraph solely by a collection of sets. In this case, the set of vertices, denoted V_X, is implicitly defined as the union of edges.

Definition 4.2 (Induced Subhypergraph). The subhypergraph $H[A]$ induced by $A \subseteq V$ is defined by $H[A] = (A, X_A)$ with $X_A = \{x \cap A \mid x \cap A \neq \emptyset\}$.

A set of examples X can be seen as a hypergraph $H = (\mathbb{F}, X)$, i.e. such that each example is a hyperedge (Figure 1). In practice, we do not need to know $\mathbb{F}$ as we can always use the implicit hypergraph $H = (\mathbb{F}_X, X)$ where $\mathbb{F}_X$ is the restriction of $\mathbb{F}$ to the features that appear in the sample X.

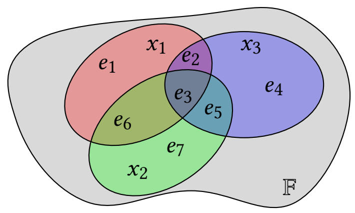

Figure 1: The family $\mathcal{E} = \{e_i\}_i$ forms the partition obtained by the union of the projection of cases and represents how the three cases share information.

Definition 4.3 (Projection defined by a subhypergraph). The projection operator π_H over a hypergraph $H = (V, X)$ for any $A \subseteq V$ is defined by $\pi_H(A) = \{e \subseteq A \mid \exists X' \subseteq X_A,\ e = \bigcap_{x \in X'} x\}$.

For a case x, the operator π takes the subhypergraph $H[x]$ and returns the partition of the features of x defined by the intersection family induced by $H[x]$. Each element of $\pi_H(x)$ is a (sub)set of features. For instance, in Figure 1, $\pi_H(x_1) = \{e_1, e_2, e_3, e_6\}$ and in Figure 2, the projection of x (in yellow) on H is represented by the family $\{e'_i\}_i$. For convenience, for a given $H = (V_X, X)$, we denote by $\mathcal{E}_H = \{e_i\}_{i=1}^m = \{e \in \bigcup_{x \in X} \pi_H(x)\}$ that is to say, the partition of V_X obtained by the intersection of the edges. This partition is unique to a hypergraph.

We call *discretionary features* of x (w.r.t. H) the set (possibly empty) of features that are not in any element of the projection $\pi_H(x)$, denoted D_x. It can be interpreted as the features of x that do not belong to any hyperedge. If a hypergraph induced by a set of examples represents a knowledge base at a given moment, the discretionary features of x are new pieces of information that were never encountered before. For instance, considering the hypergraph composed of x_1 and x_2 as illustrated by Figure 1, the set of discretionary features of x_3 is e_4. In Figure 2, the yellow case x has no discretionary feature: all its features are present at least in one example.

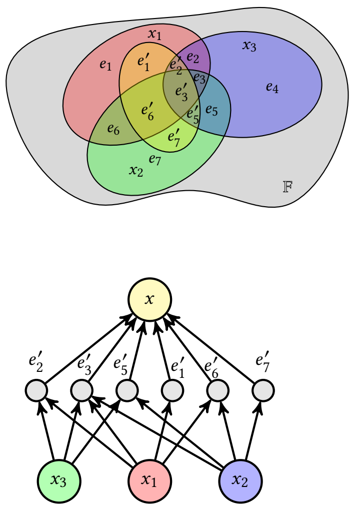

Figure 2: The projection of x on H is represented by the family $\{e'_i\}_i$. Under the projection lies a graph representation of x with the partition elements $\{e_i\}_i$ and their respective connections to the cases $\{x_i\}_i$, in particular, $D_x = \emptyset$.

For any set of features $x \subseteq \mathbb{F}$, we define $d_H(x) = \{x' \in X \mid x \cap x' \neq \emptyset\}$ the set of edges sharing some features with x. In particular, the set d_H can be split into $d_H^{(1)}$ and $d_H^{(0)}$ depending on the label of the case and defined by $d_H^{(l)}(x) = \{x' \in d_H(x) \mid J(x') = l\}$. $|d_H(x)|$ corresponds to the number of cases the case x intersects while $|d_H^{(l)}(x)|$ counts the number of times the class l is used among this set of intersecting cases. Note that if $x \notin X$ and $|d_H(x)| = 0$, then $x = D_x$, i.e. the case x is in relation with no case in X. In the hypergraph literature, $|d(x)|$ is called the *degree* of a hyperedge and its domain of definition is restricted to X.

From now, we consider only the specific hypergraph generated by the set of examples X. For the sake of readability, we remove the subscript H.

4.2 Model Space

In many binary classification approaches such as SVM or logistic regression, the model space consists of the set of hyperplanes of a given inner product space, usually $\mathbb{R}^M$. A hyperplane in $\mathbb{R}^M$ is uniquely defined by $M + 1$ parameters. For an input vector x, its margin $m(w, x)$ is defined by its distance to a hyperplan defined by w and is negative in case x is wrongly classified, positive otherwise. Thus, the problem consists in finding the *best* parametrization to minimize a loss function summed over the training set. For instance, the Perceptron algorithm minimizes the 0-1 loss, SVM the hinge loss, and the Logistic Regression uses the log loss. Those functions are defined by (3).

$$\begin{array}{rcl} L_{01}(w,x) & = & \mathbb{1}_{\{m(w,x)\leq 0\}} \\ L_{\text{hinge}}(w,x) & = & \max(0, 1 - m(w,x)) \\ L_{\log}(w,x) & = & \ln(1 + e^{-mw,x)}) \end{array} \quad (3)$$

In this paper, we relax the three implicit constraints on the input vector space:

(1) we do not assume to have any inner product or metric embedded with the input vector space,
(2) we do not assume the cardinality of the input vectors, such that it is possible to build a model and make predictions on incomplete systems (for instance missing in some rows in a database or Bag-of-Words representation),
(3) we do not assume anything about the concrete *representation* of features while in most classification methods, all the features belongs to the same space, often $\mathbb{R}$.

As a counterpart, HCBR relies on two assumptions: (i) the correct class of an input vector is the result of its features only and (ii) if two input vectors x and x' do not share any feature, they are *independent* i.e. x cannot help to understand the correct class of x' and vice versa. As a result, HCBR produces only *local* models because if a new input vector is independent of all examples, it is impossible to generate a prediction. On the contrary, a hyperplane model is *global* in a sense that it can produce a prediction for any element of $\mathbb{R}^M$. A concrete situation for which such assumptions are natural is a justice trial. Cases are composed of some elements, and the correct label is the result of a reasoning that can possibly use analogies or counter-examples with a set of past cases on top of the legal texts. However, if a judge or lawyer wants to use x to justify the outcome of x', x' must have similarities with x.

Let us formally define the model space. Given the hypergraph $H = (\mathbb{F}, X)$ defined by a training set X and its associated partition $\mathcal{E} = \{e_i\}_i^m$, the relation between an example x and its class is modeled by

$$\begin{cases} s_{w,\mu}(x) & = & \sum_{i=1}^{m} w(e_i, x)\mu(e_i) \\ \sum_{i=1}^{m} w(e_i, x_j) & = & 1 \quad \forall 1 \leq i \leq n \\ \sum_{i=1}^{n} \mu(e_i) & = & 1 \quad \forall 1 \leq i \leq m \end{cases} \quad (4)$$

where $w(e_i, x_j) \geq 0$ models the importance of e_i in x_j and $\mu(e_i)$ the support of e_i for class 1 w.r.t. whole training set. For this reason, we call μ the *intrinsic strength* of e_i. The assumption (ii) implies that if $e_i \cap x$ then $w(e_i, x) = 0$. For readability, we write s in place of $s_{w,\mu}$.

The classification rule consists in selecting the class with the total highest support.

$$\forall x \in \mathcal{P}(\mathbb{F}),\ \bar{J}(x) = \begin{cases} 1 & s(x) > 0 \\ 0 & s(x) \leq 0 \end{cases} \quad \text{(R1)}$$

Our goal is to select w and μ such that the classification rule (R1) provides a good solution to the original problem (2). For this purpose, we proceed in three steps:

(1) Define w and μ to capture as much information as possible from $\mathcal{E}$ for any X (Section 4.3).
(2) Train the model to adjust the intrinsic strength on a specific X using the classification rule (R1) (Section 4.4).
(3) Refine the classification rule (R1) to take into account the local nature of the model (Section 4.4).

A summary and high level view of HCBR is given by Algorithm 1.

Algorithm 1 HCBR (High level view)

```
1: Build $H$ and $\mathcal{E}$ from $X$.
2: Calculate $w$ and $\mu$ on $\mathcal{E}$.
3: Adjust $\mu$ with training algorithm 2 on $X$ using rule (R1)
4: for each $x$ in test set do
5:     Calculate the projection $\pi(x)$.
6:     Calculate the support $s(x)$ using the projection.
7:     Predict using the updated rule (R2).
8: end for
```

While SVM aims at separating the data using a single hyperplane in the original input vector space, HCBR tries to explain the decision for each case by a convex combination expressed in a lower dimensional space $\mathcal{E}$ generated by the data. In addition, the convex combinations depend on each other since the elements of $\mathcal{E}$ are by definition the features resulting in the case intersections. For any case, the decision rule is a combination of the strength of the elements of its projection on the hypergraph.

4.3 Model Selection

In this section, we define how to concretely select and calculate w and μ for a given hypergraph. To ease the notation and for practical reasons, the measure μ is provided w.r.t. the intersection family $\mathcal{E}$ of the hypergraph induced by the training set X. However, μ can be defined over any partition of $\mathbb{F}$.

For x and x' in $\mathcal{P}(\mathbb{F})$, a basic quantitative measure on the importance of x' w.r.t. x can be expressed by $\frac{|x \cap x'|}{|x|}$, i.e. how much x' overlaps with x. This measure is a sort of *potential* for an analogy with x. Potential because it does not account for the individual importance of the features and simply holds the idea that the larger is a subset of features in a case, the higher is the chance it holds important features to decide the outcome.

Let us consider $\mathcal{E}$ and an example $x \in X$.

Definition 4.4 (Intrinsic strength w.r.t. x). The intrinsic strength of $e \in \mathcal{E}$ w.r.t. $x \in X$ is defined by

$$\forall l \in \{0,1\},\ S^{(l)}(e,x) = \frac{|d^{(l)}(e)| \frac{|x \cap e|}{|x|}}{\sum_{e_j \in \mathcal{E}} |d^{(l)}(e_j)| \frac{|x \cap e_j|}{|x|}} = \frac{|d^{(l)}(e)||x \cap e|}{\sum_{e_j \in \mathcal{E}} |d^{(l)}(e_j)||x \cap e_j|} \tag{5}$$

In particular, for a given $x \in X$, $S^{(l)}(e_i,x) = 0$ if e_i is not part of the projection of x on H. Also, $\sum_{e \in \mathcal{E}} S^{(l)}(e,x) = 1$ which can be interpreted as how much e accounts for the set of labels l the element x holds in its projection. The more e_i belongs to many cases with the same class l and the higher $S^{(l)}(e_i,x)$ is. Conversely, for a fixed number of cases, the more e_i describes x, the higher $S^{(l)}(e_i,x)$ is. As $\forall e_i \in \mathcal{E},\ |d(e_i)| > 0$, either we have $S^{(1)}(e_i,x) \neq 0$ or $S^{(0)}(e_i,x) \neq 0$. We have $S^{(l)}(e_i,x) = 0$ only when all the cases in which e_i results of are labeled with the opposite class $\bar{l}$. For $S^{(l)}(e_i,x) = 1$, it needs both the unanimity of labels for the cases in which e_i belongs to and that $e_i = x$. The relation $e_i = x$ implies that x does not share any feature with any other examples or that x is included into another example.

Definition 4.5 (Intrinsic strength w.r.t. a hypergraph H). The instrinsic strength of $e \in \mathcal{E}$ w.r.t. $H = (\mathbb{F}_X, X)$ is defined by

$$\forall l \in \{1,0\},\ S^{(l)}(e) = \frac{|e|}{\sum_{e' \in \mathcal{E}} |e'|} \sum_{x \in d^{(l)}(e)} S^{(l)}(e,x) = \frac{|e|}{|\mathbb{F}_X|} \sum_{x \in d^{(l)}(e)} S^{(l)}(e,x) \tag{6}$$

The measure $S^{(l)}(e)$ is simply the sum of the relevance for all the examples e belongs too. The more e belongs to several cases, the more information it has to support a class or another. As $\mathcal{E}$ represents the sets of features that appear all the time together, we favor the larger $e \in \mathcal{E}$ as they hold more information to explain a decision. The normalized version is defined by:

$$\forall l \in \{1,0\},\ \mu^{(l)}(e) = \frac{S^{(l)}(e)}{\sum_{e' \in \mathcal{E}} S^{(l)}(e')} \tag{7}$$

Finally, the measure μ is simply defined by the difference between the strength of both classes:

$$\mu(e) = \mu^{(1)}(e) - \mu^{(0)}(e) \tag{8}$$

For any case x, the projection on the hypergraph $\pi(x)$ intersects with some elements of $\mathcal{E}$. We use this intersection to define w by

$$w(e,x) = \frac{|x \cap e|}{|x \setminus D_x|} \tag{9}$$

, i.e. we modulate the strength of an element of the partition by its importance in x w.r.t. set inclusion. In a sense, the projection on the hypergraph provides all the possible analogies with the training set, either seen as a support for a particular outcome l or as a counter-example. The intrinsic strength of each element of this partition is a measure of "how much it can be considered as a counter-example or an analogy" taking into account simultaneously all the analogies between the examples that are in relation with the considered case. It favors the importance of the relation (the more features it shares, the stronger is the analogy) but also the quality of the relation (the more examples sharing the same class the stronger the analogy).

Example: Consider the hypergraph in Figure 1 made of x_1, x_2 and x_3 arbitrarily labeled with resp. 1, 0 and 1. Arbitrarily, we assume the cardinal of the elements of $\mathcal{E}$ to be $\#e = (2,1,2,3,1,2,3)$ such that the cardinal of cases are $\#x = (7,8,7)$ and $|\mathbb{F}_X| = 14$. The values of $|d^{(l)}(e)|$ can be summarized by the vectors $\#d^{(0)} = (0,0,1,0,1,1,1)$ and $\#d^{(1)} = (1,2,2,1,1,1,0)$. Let us calculate $S^{(0)}(e_3)$:

$$S^{(1)}(e_3,x_1) = \frac{2 \times 2}{2 \times 1 + 1 \times 2 + 2 \times 2 + 1 \times 2} = \frac{4}{10}$$
$$S^{(1)}(e_3,x_2) = \frac{2 \times 2}{2 \times 2 + 1 \times 1 + 1 \times 2 + 0 \times 3} = \frac{4}{7}$$
$$S^{(1)}(e_3,x_3) = \frac{2 \times 2}{2 \times 1 + 2 \times 2 + 3 \times 1 + 1 \times 1} = \frac{4}{10}$$

which we interpret as e_3 being responsible for $\frac{4}{10}$ of the support toward class 1 in x_1 and x_3, while $\frac{4}{7}$ for x_2. This lead to

$$S^{(1)}(e_3) = \frac{2}{14}\left[\frac{4}{10} + \frac{4}{7} + \frac{4}{10}\right] \simeq 0.1959$$

Similarly, we calculate the support for each e and both labels. We summarize this into the following vectors:

$$S^{(1)} \simeq (0.0286, 0.0286, 0.1959, 0.0643, 0.0173, 0.0694, 0.0000)$$
$$S^{(0)} \simeq (0.0000, 0.0000, 0.2024, 0.0000, 0.0327, 0.1071, 0.0000)$$

After normalization, we obtain the intrinsic strength:

$$\mu \simeq (0.0707, 0.0707, 0.0060, 0.1591, -0.0345, -0.0818, -0.1901)^T$$

Let us evaluate the model on the three examples:

$$s(x_1) = \frac{2}{7}\mu(e_1) + \frac{1}{7}\mu(e_2) + \frac{2}{7}\mu(e_3) + \frac{2}{7}\mu(e_6) \simeq 0.0086$$
$$s(x_2) = \frac{2}{8}\mu(e_3) + \frac{1}{8}\mu(e_5) + \frac{2}{8}\mu(e_6) + \frac{3}{8}\mu(e_7) \simeq -0.0946$$
$$s(x_3) = \frac{1}{7}\mu(e_2) + \frac{2}{7}\mu(e_3) + \frac{3}{7}\mu(e_4) + \frac{1}{7}\mu(e_5) \simeq 0.0751$$

As a result, x_1 and x_3 are labeled 1 and x_2 is labeled 0. All three cases are correctly labeled. The highest support is given for case x_2 and x_3 while the support for x_1 is one order of magnitude lower then for x_3. This is because the discretionary features of x_3 are larger while the intersection with x_2 is lower than for x_1 ($\frac{3}{8}$ of x_3 against $\frac{4}{7}$ of x_1).

Consider a new case x as described on Figure 2. Its support is given by $s(x) = \sum_{e \in \pi(x)} w(e,x)\mu(e)$ with $\pi(x) = \{e_1, e_2, e_3, e_5, e_6, e_7\}$. It is impossible for x to be classified as 1 because the highest support would be for a maximal intersection with e_1, e_2, e_3 and minimal for e_5, e_6 and e_7 such that $s(x) = \frac{1}{8}(2\mu(e_1) + \mu(e_2) + 2\mu(e_3) + \mu(e_5) + \mu(e_6) + \mu(e_7)) \simeq -0.0103 < 0$. It can be explained by the fact that the support for 1 is provided by a larger set of features (11 features versus 8). On top of that, the intersections between positive cases (e_2 and e_3) are too small (1 for e_2 compared to e.g. 3 for e_7) or include also negative cases (e_3).

4.4 Training and Decision

In this section, we give an algorithm to adjust the intrinsic strength $\mu(e_i)$ in order to minimize a hinge loss and we update the classification rule to take into account the fact the model cannot provide predictions over $\mathbb{F}$ entirely.

Once the model is built, it can be evaluated on the training set. Analogously to SVM, we define the margin as the distance to the correct class, i.e. $m(w, \mu, x) = s_{w,\mu}(x)|J(x) - \bar{J}(x)|$. To improve the pertinence of the strength of the elements of $\mathcal{E}$, we use the iterative algorithm described by Algorithm 2 to minimize the hinge loss over the training set X. When and only when a classification for a case x is wrong, it modifies each element of its projection by lowering its strength for the wrong class and increasing it for the proper class. The margin is split between the element of the projection w.r.t. their respective weight in x i.e. $w(e, x)$. If a case x is wrongly classified, it is due to the cases intersecting with it. Indeed, if x was not intersecting with any other example, its projection would be itself, and its support toward the wrong class would be 0 and positive for the real class. In other words, x would be correctly classified. Thus, the idea is not to directly bring the support of x to the correct class but to gradually adjust the weights such that the neighbors are modified enough for x to be correctly classified. In particular, it is sensitive to the order in which the cases are considered: a modification in the strength of any $e \in \mathcal{E}$ impacts all cases in which it appears and potentially changes the predicted class for those cases. In addition, there is no guarantee of convergence. Future work will focus on the optimal order or a modification schema such that the algorithm converges. Investigating other minimizing functions is also considered.

Algorithm 2 Model training

```
Input:
 - $X$: training set
 - $y$: correct labels for $X$
 - $k$: number of training iterations
 - $\mu^{(1)}, \mu^{(0)}$: weights calculated with (4.5)
Output:
 - Modified vectors $\mu^{(1)}, \mu^{(0)}$
 1: for $k$ iterations do
 2:     for $x_i \in X$ do
 3:         $\bar{y}_i \leftarrow \bar{J}(x_i)$
 4:         if $\bar{y}_i \neq y_i$ then
 5:             for $e \in \pi(x_i)$ do
 6:                 $\mu^{(y_i)}(e) \leftarrow \mu^{(y_i)}(x_i) + w(e, x_i)|\mu(e)|$
 7:                 $\mu^{(\bar{y}_i)}(e) \leftarrow \mu^{(\bar{y}_i)}(x_i) - w(e, x_i)|\mu(e)|$
 8:             end for
 9:         end if
10:     end for
11: end for
```

The measure μ is defined as the difference of support for both classes. Thus, by linearity we can rewrite

$$\begin{aligned} s(x) &= \sum_{i=1}^{m} w(e_i, x)\mu^{(1)}(e_i) - \sum_{i=1}^{m} w(e_i, x)\mu^{(0)}(e_i) \\ &= s^{(1)}(x) - s^{(0)}(x) \end{aligned} \tag{10}$$

This form is convenient because we can control how much evidence we need to support a specific class using the following constraints and a familly η of four hyperparameters:

$$s^{(0)}(x) > \max(\frac{\bar{\eta}_0}{1 - \bar{\eta}_0} s^{(1)}(x), \eta_0) \geq 0 \tag{C_0}$$

$$s^{(1)}(x) > \max(\frac{\bar{\eta}_1}{1 - \bar{\eta}_1} s^{(0)}(x), \eta_1) \geq 0 \tag{C_1}$$

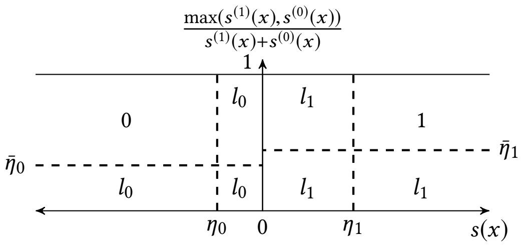

Figure 3: Representation of the updated decision rule (12).

with $\eta_0, \eta_1 \in \mathbb{R}^+$ and $\bar{\eta}_0, \bar{\eta}_1 \in [0, 1]$. The constraints on η_0 and η_1 defines a minimal amount of support respectively toward class 0 and 1 while $\bar{\eta}_0$ and $\bar{\eta}_1$ requires the support toward a class to be significantly higher than the support for the other class. As μ is normalized over the partition $\mathcal{E}$, the value of η_0 and η_1 must be set w.r.t. the hypergraph. On the contrary, $\bar{\eta}_1$ and $\bar{\eta}_0$ can be set independently of the hypergraph.

Those constraints may be used to design a decision rule for new cases depending on the application or the dataset. The most generic decision rule is as follows:

$$\tilde{J}(x) = \begin{cases} 1 & s(x) > 0 \quad \text{and } C_1 \\ 0 & s(x) \leq 0 \quad \text{and } C_0 \\ l_1 & s(x) > 0 \quad \text{and not } C_1 \\ l_0 & s(x) \leq 0 \quad \text{and not } C_0 \end{cases} \tag{12}$$

where l_1, l_0 are two labels. A representation is given by Figure 3. Those hyperparameters are intended to model the "burden of proof". For instance, in a trial, one is assumed innocent until proven guilty which implies the support for the class "guilty" must be *beyond a reasonable doubt* (where the term reasonable is defined by the jurisprudence of the applicable country). In case $\eta_0 = \eta_1 = \bar{\eta}_0 = \bar{\eta}_1$ (and $l_0 = 0$ and $l_1 = 1$), then the decision rule is equivalent to the original one defined by (R1). In case x has too many discretionary features, this classification rule is likely to be irrelevant. Indeed, the intersection between x and $\mathbb{F}_X$ is to small to hold enough information and make strong analogies with x. To overcome this drawback, $\mathcal{P}(\mathbb{F})$ is split into two subsets:

- $\mathcal{F}_1 = \{x \in \mathcal{P}(\mathbb{F}) \mid |x \cap \mathbb{F}_X| \geq \delta\}, \ \forall \delta \in \mathbb{N}$
- $\mathcal{F}_2 = \mathcal{P}(\mathbb{F}) \setminus \mathcal{F}_1$

$\mathcal{F}_1$ corresponds to the elements such that they share some features with the examples. An alternative may be considered by using $\mathcal{F}_1 = \{x \in \mathcal{P}(\mathbb{F}) \mid \frac{D_x}{|x|} \leq \delta\}, \ \forall \delta \in [0, 1]$. In this case, $\mathcal{F}_1$ contains the elements for which we have enough information provided by the examples. From our preliminary tests, the choice depends on the dataset structure.

Finally, the decision rule for new cases is built as follows:

$$\bar{J}(x) = \begin{cases} \tilde{J}(x) & \text{if } x \in \mathcal{F}_1 \\ o_x & \text{if } x \in \mathcal{F}_2 \end{cases} \tag{R2}$$

where o_x is one draw from a random variable that has Bernoulli law with parameter $p = \frac{|\{x \in X \mid J(x)=1\}|}{|X|}$, i.e. the prevalence of 1 in X. This assumes that X is correctly representing $\mathcal{P}(\mathbb{F})$ (or that the prevalence does not change in time for sequential problems in which the new cases are generated by an unknown random measure). The rational behind is that if for a case x, it is not possible to exploit the model built on the hypergraph, then we can still consider that J acts as a Bernoulli random variable and use a maximum likelihood estimation for p. In a sense, it is extending the *local* model to the entire input vector space $\mathcal{P}(\mathbb{F})$.

4.5 Complexity

Model Building: Given $X \in \mathcal{P}(\mathbb{F})^n$, constructing $\mathcal{E}_H$ can be done in $O(\sum_{x \in X} |x|)$ by using a Partition Refinement data structure [18]. Given $x \in X$, calculating the family $\{S(e,x)\}_{e \in \mathcal{E}_H}$ can be done in $O(|x|)$ by asking for each feature of x the e it belongs to and maintaining the size of each e during the construction of $\mathcal{E}_H$. Thus, calculating $\{S(e,x)\}_{e \in \mathcal{E}_H}$ for all $x \in X$ can be done in $O(\sum_{x \in X} |x|)$. On m-uniform hypergraphs (when all cases are described with m features) with n hyperedges, it becomes $O(mn)$.

Calculating $\{S(e)\}_{e \in \mathcal{E}_H}$ and μ can be done in $O(|\mathcal{E}_H|)$ because it requires to iterate over $\mathcal{E}_H$. An obvious upper bound on $|\mathcal{E}_H|$ is $|\mathbb{F}_X|$ i.e. the number of vertices in the hypergraph. The worst-case cardinal of $\mathcal{E}_H$ is when each $x \in X$ intersects with all the others and none of them is strictly a subset of any other. Thus, $|\mathcal{E}_H| \leq \min(2^n - 1, |\mathbb{F}_X|)$.

Learning Phase: For each wrongly classified $x \in X$, the training requires at most $O(|x|)$ steps (maximal cardinal for $\pi(x)$). The worst-case scenario is when the model wrongly classifies every $x \in X$. Thus, the learning phase worst-case complexity is $O(k \sum_{x \in X} |x|)$ and on m-uniform hypergraphs it becomes $O(kmn)$.

Model Query: For a case $x \in \mathcal{P}(\mathbb{F})$, the projection can be done in $O(|x|)$. Calculating the classification rule also requires at most $O(|x|)$ (maximal cardinal for $\pi(x)$).

5 EXPERIMENTS

The experiments are broken down into two parts: the comparison with literature results in terms of classification performance, and intrinsic performance (computation time and influence of parameters).

For the first part, we measured the confusion matrix obtained over all the runs but also after each prediction. From this confusion matrix, we calculated the standard performance indicators: accuracy, recall, specificity, precision, negative prediction value, F_1-score and Matthews correlation coefficient. Denoting by TP the number of true positives, TN the true negatives, FP the false positives and FN the false negative, the F_1-score and Matthews correlation coefficient are defined by:

$$F_1 = \frac{2TP}{2TP + FP + FN}$$

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$$

F_1- score and Matthews correlation coefficients respectively belongs to $[0,1]$ and $[-1,1]$ and the closer to 1, the better it is. Both takes into account false positive and false negatives. In particular, Matthews correlation coefficient is very adapted for binary classification on unbalanced dataset (with a prevalence far from 0.5). However, as many studies do not report them, we will base our comparison with literature results on the accuracy defined by:

$$ACC = \frac{TP + TN}{TP + TN + FP + FN}$$

For the second part, we studied the computation time depending on the number of cases and the size of each case in order to validate the worst-case complexity given in Section 4.5. We also studied the influence of the training set size on the accuracy.

The integrality of the data used for the experiments, as well as the scripts to transform them and analyze the results are available in the HCBR Github repository[4] and the whole experimental campaign starting from the raw data can be reproduced in "one click".

5.1 Classification performance

To validate the approach, 8 datasets for binary classification have been used. They are available either from the UCI Machine Learning Repository[5] or provided with the LIBSVM[6] : `adult`, `audiology`, `breasts`, `heart`, `mushrooms`, `phishing`, `skin` and `splice`. For each dataset, the original features `(name=value)` are converted into a unique identifier and the union of all such identifiers constitute the information set $\mathbb{F}$ considered by the algorithm. Notice that there is no need to remove the rows with empty values. The dataset `audiology` initially contains several classes corresponding to several ear abnormalities. They are grouped to obtain two classes (normal ear and abnormal ear). able 1 describes each dataset. The minimal, maximal and average size give information about the case sizes (notice some cases are missing values for `adult`, `heart` and `mushrooms` datasets). The unique features are the number of `(name=value)` in the original dataset. In addition, two datasets have at least one real-valued attribute as indicated by the column "Real" in Table 1.

The validation was made using a 10-fold cross-validation: each dataset has been split into 10 equal sized samples, 90% has been used as training set and the remaining 10% to test the classification. Each subsample is used once as testing set and the final metrics are calculated as the average of the 10 runs. The training set was not rebalanced and kept the original prevalence. The of training steps k was adjusted with a manual trial and errors approach, the familly η set to $\eta_0 = \eta_1 = \bar{\eta}_0 = \bar{\eta}_1 = 0$ and δ was set to 1. Further work will focus on automatic parameter tuning. The average confusion matrix obtained for each dataset

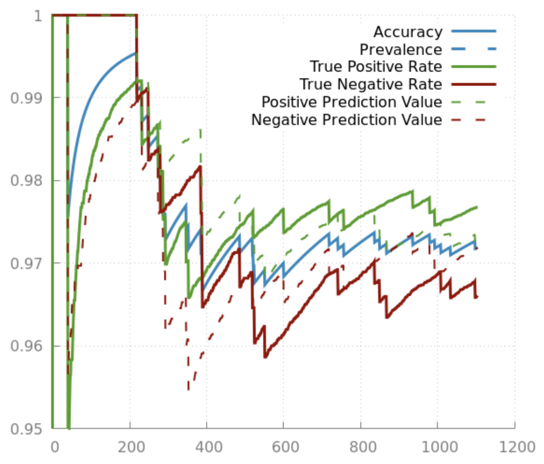

Figure 4: Evolution of the confusion matrix during the prediction phase for `phishing` dataset.

is showed in Table 2. The performance indicators are reported in Table 3. The proposed algorithm performs very well on a wide range of datasets as reported by tables 2 and 3, in particular

[4]https://github.com/aquemy/HCBR
[5]https://archive.ics.uci.edu/ml/index.php
[6]https://www.csie.ntu.edu.tw/ cjlin/libsvmtools/datasets/binary.html

Table 1: Datasets description.

	Cases	Total Features	Unique	Min. Size	Max. Size	Average Size	Real
adult	32561	418913	118	10	13	12.87	No
audiology	200	13624	376	70	70	70	No
breasts	699	5512	80	8	8	8	No
heart	270	3165	344	12	13	12.99	Yes
mushrooms	8124	162374	106	20	20	20	No
phishing	11055	319787	808	29	29	29	No
skin	245057	734403	768	3	3	3	Yes
splice	3175	190263	237	60	60	60	No

when they contain strong predictors as it is the case for `mushroom`. The accuracy is contained in a range from 0.8206 (`adult`) to 1 (`mushrooms`) while the F_1-score is bounded by 0.8653 (`heart`) and 1 (`mushrooms`). On `adult`, the accuracy is only 6% higher than the prevalence. A model consisting in returning 1 for any point would be only 6% worse. This relatively poor performance in learning the underlying decision mapping is better reflected by the Matthews correlation coefficient of 0.51.

The false positives and false negatives are equilibrated for each dataset, despite a huge variation in the prevalence (between 20% and 64%, cf. Table 2) which may be a desirable property depending on applications. In addition, the results were obtained with non-balanced training sets which are known to be a problem for many machine learning algorithms. Moreover, the performance indicators remain stable during the whole prediction phase as shown on Figure 4 for the dataset `phishing` (similar result is observed for all datasets).

For a given case, the support is a metric of confidence in the prediction as illustrated in Figure 5. In general, the wrongly classified cases have a smaller difference between the evidence for each class which confirm the interest in the hyperparameters η and $\bar{\eta}$ used in (R2).

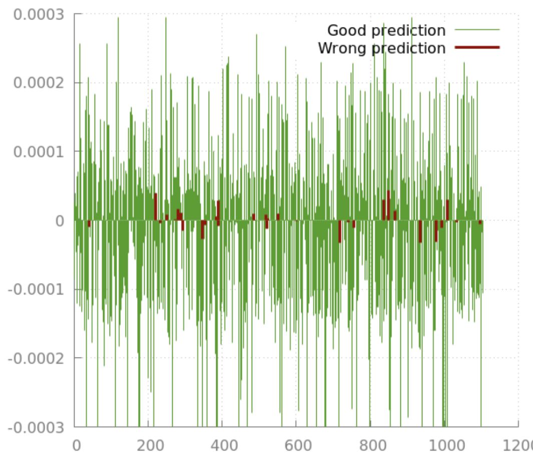

Figure 5: Difference between the weight assigned to both classes for each decision on `phishing` (average). Similar results are observed for all datasets.

To compare the results of the proposed method, we explored the best results from the literature for each dataset. The comparison with `audiology` is not relevant due to the transformation into a two-class problem. The results are summarized in Table 4. In [9], 5 rule-based classification techniques dedicated to medical databases are compared and achieve at best 95.85% and 82.96% accuracy resp. on `breast`, and `heart` datasets. Comparing bayesian approaches, [13] demonstrated 97.35% (`breast`) and 83.00% (`heart`) accuracy. A 5 layers neural network with fuzzy inference rules achieved 87.78% on `heart` [21] while a k-NN algorithm reached 99.96% on `mushrooms` [8]. The best alternative among 6 rules-based classification methods achieved 95.84% on `breast` and 100.00% on `mushroom` [11]. Using 80% of `phishing` as training set, an adaptative neural network achieved an average accuracy of 93.76% (among 6 alternatives) with the best run at 94.90% [23]. Still on `phishing`, [22] proposes to combine several classifiers and reaches 97.75% accuracy for the best hybrid model (and demonstrates 97.58% for Random Forest classifier). On `adult`, the comparison of several classifiers (naive bayes, decision tree, ...) demonstrated at most 86.25% accuracy [14] while a Support Vector Machine approach reached 85.35% [15]. On `splice`, a method using Fuzzy Decision Trees [4] reaches 94.10% accuracy and a neural network combined to boosting [5] 97.54%. On `breast`, Support Vector Machine approaches reached resp. 96.87%, 98.53%, 99.51% accuracy [1, 7, 19], 99.26% and 97.36% for neural network based techniques [17, 24], 98.1% for a bayesian network method [10], or 94.74% using Decision Trees [20]. On `skin`, [5] reports 98.94% accuracy against 99.68% for Decision Tree based method [6]. The best result, as far as we know, is 99.92%, obtained by a Generalized Linear Model [2] (with 80% training set).

The accuracy is slightly lower than the best results from the literature (`adult` 82.06% against 86.25%, `breast` 96.96% against 99.51%, `heart` 85.77% against 87.78%, `phishing` 96.05% against 97.75%, `splice` 94.43% against 97.54%, `skin` 98.65% against 99.92%). We explain this by at least two factors. First, the best methods on a given dataset are often dedicated to this dataset with *ad-hoc* or engineered parts which is not the case of HCBR. Secondly, the hyperparameter familly η have not been tuned but as we will show in next section, they might have a decisive impact on performance. HCBR performed better than Bayes classifier in two thirds of cases. Bayes classifier performs better on `breast` by $\tilde{1}$% which represents less than one case wrongly classified. Similar results are observed with Decision Trees. However, the 1% difference on `skin` represents an average of 7 cases misclassified in comparison in favor of Bayes. It performs better than Rule-based approaches (or gives similar results on `mushrooms` with an accuracy of 1) in the four considered references on three different datasets. Notice that combined with Neural Network, Rule-based achieves the state-of-art result on `heart` dataset. Except for `phishing`, Neural Network returns better results (0.46 more cases with correct classification in average for `breast`, 71 for skin and almost 10 for `splice`). Last, SVM gives better results in all three cases, but appear only as best results in two datasets.

Table 2: Average confusion matrix obtained with a 10-fold cross-validation.

	TP	FN	FP	TP	Prevalence
adult	2182.40	295.30	288.50	488.80	0.7586
audiology	12.70	0.10	0.00	6.20	0.6048
breast	23.00	1.40	0.70	43.90	0.3338
heart	12.40	1.80	1.90	9.90	0.5107
mushrooms	390.60	0.00	0.00	420.40	0.4804
phishing	595.40	23.80	19.80	465.00	0.5562
skin	4886.30	132.40	199.40	19286.90	0.2075
splice	155.70	9.10	8.50	142.70	0.5164

5.2 Intrinsic performance

In this section, we evaluate the capacity of HCBR to build an efficient model (w.r.t. the accuracy measure) with few examples. We also study the influence of the two main parameters (number of examples and size of the examples) on the computation time.

Training set size: To evaluate HCBR, we used a standard 90-10 dataset split in the previous section. Additionally, we studied the accuracy depending on the split from 1% to 99%. For each split value, we performed 100 runs with random splits and averaged the accuracy. As shown in Figure 6, HCBR reaches its maximal accuracy with about 15% of the dataset as examples. The dataset `adult` shows a constant results from 1% while for `audiology` more than 40% is required to achieve over 90% accuracy. Despite the datasets having different sizes (e.g. `skin` is 350 larger than `breast`), the trajectories look the same. Further work should focus on determining the optimal training set size depending on the size of cases and the underlying measure to generate them in $\mathcal{P}(\mathbb{F})$ (as it influences the induced hypergraph, it influences the strength measure and thus the decisions). Additional experiments need to be performed to fairly compare those results to the existing algorithms. However, it is commonly accepted that non-linear classifiers requires very large training sets.

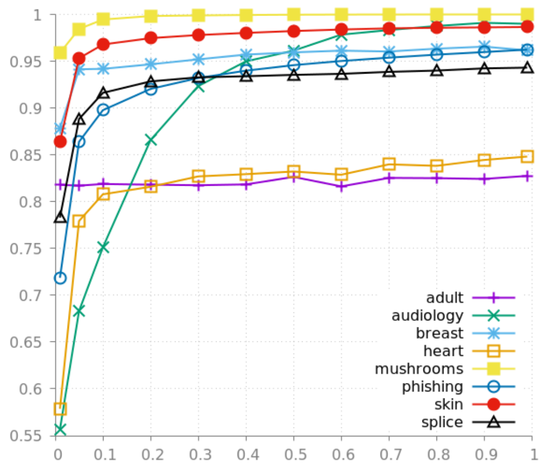

Figure 6: Accuracy depending on the percentage of the dataset used as training set.

Computation Time: We generated a casebase of n cases of size m such that case i is described by $\{i, ..., i + m\}$ i.e., each case is partitioned into m elements (one discretionary feature). This is the worst-case scenario in terms of the size of $\mathcal{E}$ if $m < n$ because the family grows exponentially in function of m or n. We split the computation time into constructing the hypergraph (and determining the intersection family) and calculating the strength of the partition. The results are illustrated in Figure 7. By increasing n with a fixed m, the partition grows exponentially and thus, it is expected to have an exponential curve for the strength computation. On the contrary, building the hypergraph can be done in linear time when m fixed. When n is fixed and m increases, constructing the hypergraph is still doable in linear time as expected. Interestingly, calculating the strength has two phases: if $m \leq n$, increasing m exponentially increases the time (because $\mathcal{E}$ exponentially increases) but if $m > n$, increasing m cannot results in an exponential growth in the computation time (because $\mathcal{E}$ grows linearly).

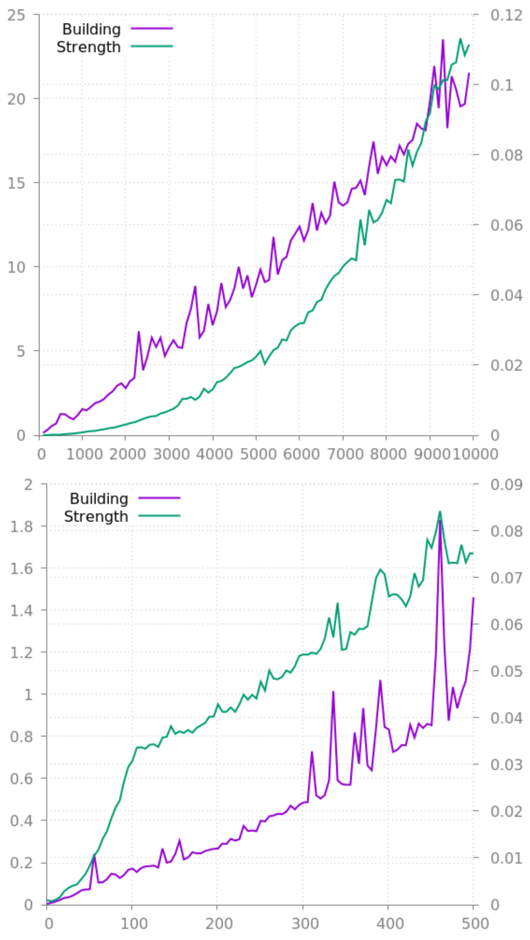

Figure 7: At the top, computation time to build the model (hypergraph construction + strength calculation) depending on n ($m = 10$), and at the bottom, depending on m ($n = 100$). The case i is described by $\{i, ..., i + m\}$ such that each case is partitionned into m elements (one discretionary feature). Right scale for bulding and left scale for strength.

Hyperparameters η: We used a 90-10 split and set $\eta_0 = \eta_1$ to ease the visualization. Instead of using the decision function defined by (12), we did not produce a prediction if the constraints C_1 or C_0 were not respected. It can be viewed as creating a third class

Table 3: Average performances obtained with a 10-fold cross-validation.

	Accuracy (standard dev.)	Recall	Specificity	Precision	Neg. Pred. Value	F_1 score	Matthews corr. coef.
`adult`	0.8206 (0.0094)	0.8832	0.6233	0.8808	0.6290	0.8820	0.5081
`audiology`	0.9947 (0.0166)	1.0000	0.9875	0.9917	1.0000	0.9957	0.9896
`breasts`	0.9696 (0.0345)	0.9691	0.9676	0.9479	0.9844	0.9575	0.9344
`heart`	0.8577 (0.0943)	0.8695	0.8437	0.8699	0.8531	0.8653	0.7178
`mushrooms`	1.0000 (0.0000)	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
`phishing`	0.9605 (0.0081)	0.9680	0.9514	0.9615	0.9590	0.9647	0.9199
`skin`	0.9865 (0.0069)	0.9608	0.9932	0.9736	0.9898	0.9672	0.9587
`splice`	0.9443 (0.0124)	0.9478	0.9398	0.9450	0.9441	0.9463	0.8884

Table 4: Previous literature results measured as the highest accuracy obtained by the authors.

Dataset	Ref.	Type	Accuracy
`adult`	[14]	Many classifiers	86.25%
	[15]	SVM	85.35%
		HCBR	**82.06%**
`breast`	[1]	SVM	99.51%
	[17]	Neural Network	99.26%
	[19]	SVM	98.53%
	[10]	Bayes	98.1%
	[24]	Neural Network	97.36%
	[13]	Bayes	97.35%
		HCBR	**96.96%**
	[7]	SVM	96.87%
	[9]	Rule-based	95.85%
	[11]	Rule-based	95.84%
	[20]	Decision Tree	94.74%
`heart`	[21]	Neural Network + Rule-based	87.78%
		HCBR	**85.77%**
	[13]	Bayes	83.00%
	[9]	Rule-based	82.96%
`mushrooms`	[11]	Rule-Based	100.00%
		HCBR	**100.00%**
	[8]	k-NN	99.96%
`phishing`	[22]	Ensemble	97.75%
	[22]	Random-Forest	97.58%
		HCBR	**96.05%**
	[23]	Neural Network	94.90%
`skin`	[2]	Generalized Linear Model	99.92%
	[6]	Decision Tree	99.68%
	[5]	Neural Network + Boosting	98.94%
		HCBR	**98.65%**
`splice`	[5]	Neural Network + Boosting	97.54%
		HCBR	**94.43%**
	[4]	(fuzzy) Decision Tree	94.10%

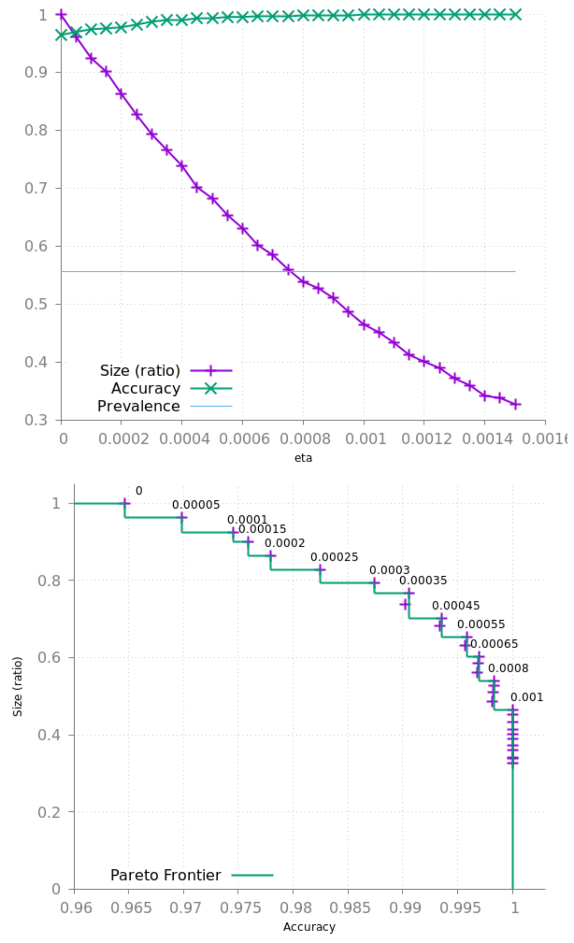

Figure 8: Influence of η on `phishing` dataset.

unknown for which we consider we cannot produce a decision. We measured the accuracy and the test set size ratio for which a prediction has been produced for different values of $\eta := \eta_0 = \eta_1$. If $\bar{J}$ correctly approximates J, increasing η should increase the accuracy while the test set ratio should remain high. Additionally, we plot the test set ratio in function of the accuracy and calculate the Pareto frontier[7] which represents the best compromises accuracy/ratio. The closer the points are to $(1, 1)$ the better it is. A Pareto frontier consisting of $(1, 1)$ represents the perfect model (e.g. reached on `mushroom` dataset). Figures 8, 9, 10 and 11 provides the result for the best and worst two datasets. Figure 12 shows all of the four Pareto frontiers. As expected, the results are better on `phishing` and `breast`. On `phishing`, `breast` and `heart`, the accuracy globally increases with η while on `heart` the accuracy slightly decreases indicating poor influence of the hyperparameters and model.

Notice that for certain values of η it is possible to reach 100% accuracy with `heart` while it is not with `breast`. Also, for high values of η, we observe a fall in accuracy for `breast`. We suspect those two phenomena to appear because we used the same value for η_0 and η_1.

More work is required to fully study the influence of the hyperparameters $(\eta_0, \eta_1, \bar{\eta}_0, \bar{\eta}_1)$ and how to select l_0 and l_1. We believe it is the key to improve the overall performance, and it is possible to derivate the best values from the training set. Also, a comparison with binary classification methods that provide a prediction confidence metric is necessary.

[7]Points such that improving one component would degrades the other one.

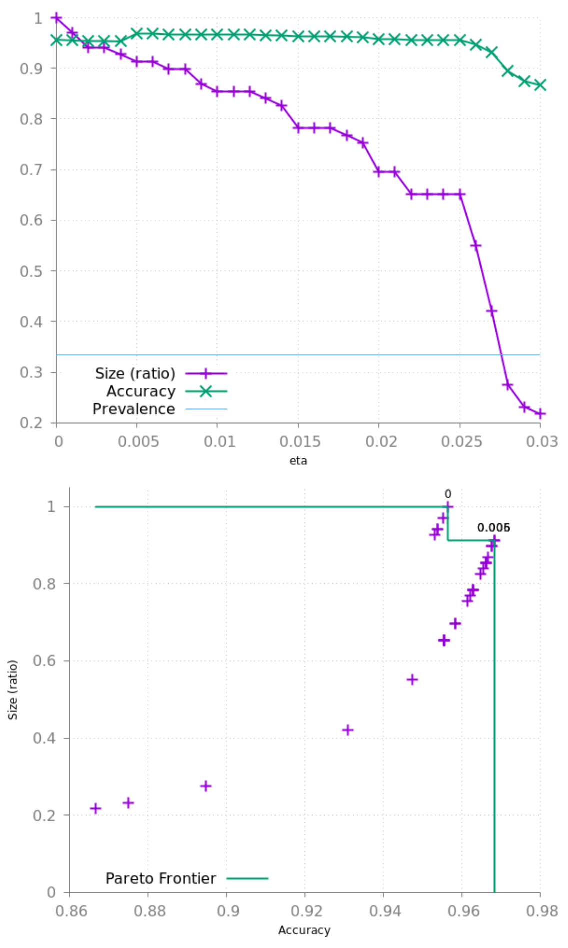

Figure 9: Influence of η on `breast` dataset.

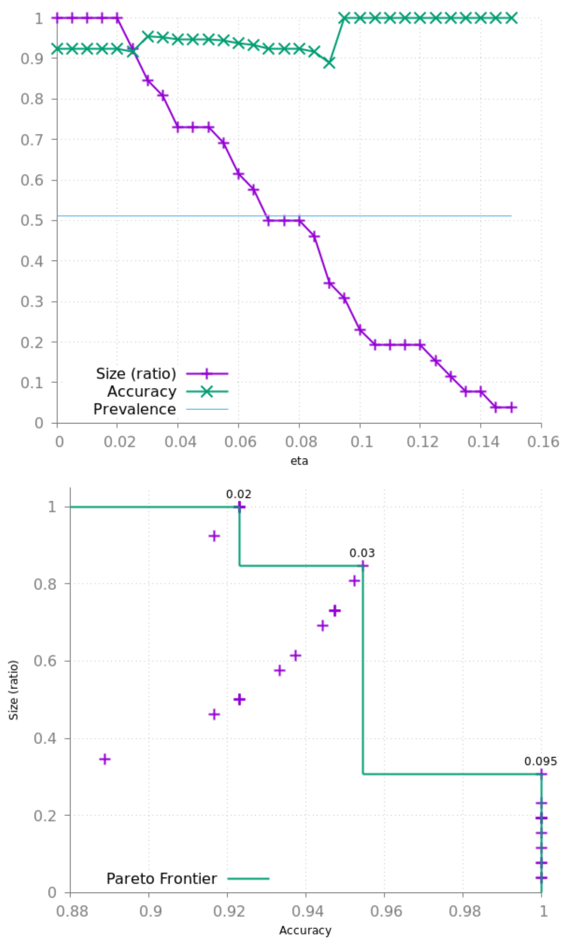

Figure 10: Influence of η on `heart` dataset.

6 CONCLUSION

This paper presented HCBR, a method for binary classification using a hypergraph representation of information and building a convex combination out of the induced partition to determine the support for each class. The general framework introduced by (4) is instantiated in Section 4.3 where the support is determined using all the interactions between the hyperedges. Beyond this specific implementation, one can imagine different model selection methods to be used, e.g. using some assumptions on the data.

However, being totally agnostic on the data representation is convenient compared to many methods such as SVM. It allows combining information from multiple sources by simply *stacking* the information. It does not require transforming the data to fit the algorithm, often by designing specific ad-hoc metrics.

HCBR has been tested on 8 well-known datasets and demonstrated similar accuracies when compared to the best results from the literature. The algorithm has shown a strong stability in terms of accuracy, true positive and negative rates during the whole prediction phase. We showed that the difference of class support is a good confidence indicator for the prediction. We demonstrated the capacity to obtain a good accuracy using very few examples from the dataset (10% to 15% of the training set) without balanced classes. This last property is very important for robustness as in practice, the dataset are rarely balanced. Finally, we empirically validated the exponential worst-case complexity.

This proof of concept raises many questions and offer many improvement axes. First of all, it seems relatively easy to extend the method to several classes but it needs an additional empirical validation. As most of the computational effort is on calculating the class support, adding more classes will linearly increase the computation time and thus, working on a faster algorithm or an approximation of the main measure should be investigated. The solution may come from exploring the feature selection capacity of HCBR. Indeed, by the hypergraph construction, it may be possible to remove from the partition some elements that do not participate enough (e.g. not being in enough cases at the same time), reducing the computation time. Additionally, we plan to investigate how to generate an explanation about each prediction and one about the decision function J itself, using not only the convex combination and the strength of the partition elements, but also the link between cases in a similar way a lawyer may use past cases to make analogies or counter-examples. We also work on an online version of HCBR where the hypergraph is constructed case after case, including forgetting some old past cases (which would allow handling non-stationary environment). It seems also possible not only to add new examples dynamically, but also some vertices (i.e. adding some pertinent information to some cases) without generating the whole model from scratch.

Empirically, further experiments should focus on more unstructured datasets (for instance for text classification). As stated

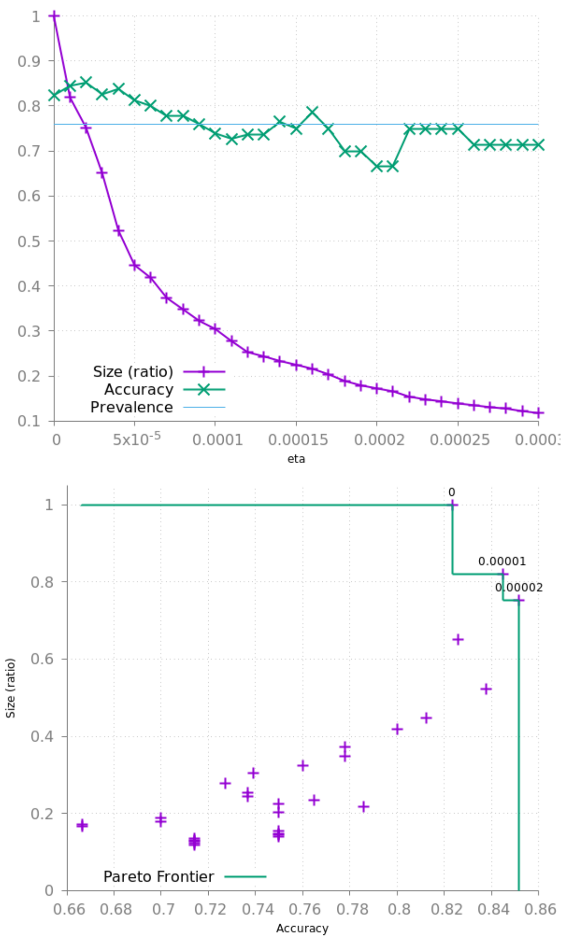

Figure 11: Influence of η on adult dataset.

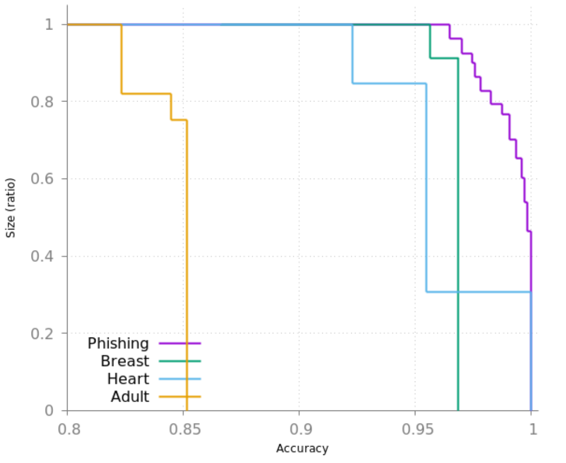

Figure 12: Pareto Frontiers comparison.

previously, strategies for hyperparameter tunning are also a priority. Last but not least, we would like to answer some questions: is it possible to find a method to adjust the strength measure such that the accuracy on the training set is 1? Can we provide some quality bounds depending on the initial hypergraph and thus the dataset? How to handle continuous values?

ACKNOWLEDGMENT

The author warmly thanks Pr. Robert Wrembel, Poznan University of Technology, and Dr. Jean-François Puget, IBM Analytics, for their useful suggestions and advice to improve this paper.

REFERENCES

[1] M. F. Akay. 2009. Support vector machines combined with feature selection for breast cancer diagnosis. *Expert systems with applications* 36, 2 (2009), 3240–3247.

[2] Mesa A. Dinh N.-T. Basterrech, S. 2015. Generalized Linear Models Applied for Skin Identification in Image Processing. In *Intelligent Data Analysis and Applications*. Springer, 97–107.

[3] C. Berge. 1984. *Hypergraphs: combinatorics of finite sets*. Vol. 45. Elsevier.

[4] Sharma G. Dhall-A. Chaudhury S. Bhatt, R. B. 2009. Efficient Skin Region Segmentation Using Low Complexity Fuzzy Decision Tree Model. In *2009 Annual IEEE India Conference*. 1–4.

[5] F. Ö. Çatak. 2017. Classification with boosting of extreme learning machine over arbitrarily partitioned data. *Soft Computing* 21, 9 (2017), 2269–2281.

[6] Ribeiro M. X. Cazzolato, M. T. 2013. A statistical decision tree algorithm for medical data stream mining. In *Proceedings of the 26th IEEE International Symposium on Computer-Based Medical Systems*. 389–392.

[7] Yang B. Liu-J. Liu D.-Y. Chen, H.-L. 2011. A support vector machine classifier with rough set-based feature selection for breast cancer diagnosis. *Expert Systems with Applications* 38, 7 (2011), 9014 – 9022.

[8] S. Das. 2001. Filters, Wrappers and a Boosting-Based Hybrid for Feature Selection. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML '01)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 74–81.

[9] Saha S. Datta, R. P. 2016. Applying rule-based classification techniques to medical databases: an empirical study. *International Journal of Business Intelligence and Systems Engineering* 1, 1 (2016), 32–48.

[10] Jafari S. Fallahi, A. 2011. An expert system for detection of breast cancer using data preprocessing and bayesian network. *International Journal of Advanced Science and Technology* 34 (2011), 65–70.

[11] Issa G. Ishtaiwi A. Hadi, W. 2017. ACPRISM: Associative classification based on PRISM algorithm. *Information Sciences* 417, Supplement C (2017), 287 – 300.

[12] Liu Q. Zhang S.-Metaxas D. N. Huang, Y. 2010. Image retrieval via probabilistic hypergraph ranking. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 3376–3383.

[13] L. Jiang. 2012. Learning Instance Weighted Naive Bayes from Labeled and Unlabeled Data. *J. Intell. Inf. Syst.* 38, 1 (2012), 257–268.

[14] Lu Y. Peng Y.-Shi Y. Kou, G. 2012. Evaluation of classification algorithms using MCDM and rank correlation. *International Journal of Information Technology & Decision Making* 11, 01 (2012), 197–225.

[15] Mangasarian O.L. Lee, Y.-J. 2001. SSVM: A Smooth Support Vector Machine for Classification. *Computational Optimization and Applications* 20, 1 (2001), 5–22.

[16] Xibin Zhao-Hai Wan Ming Gu Jiaguang Sun Lifan Su, Yue Gao. 2017. Vertex-Weighted Hypergraph Learning for Multi-View Object Classification. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*. 2779–2785.

[17] Quintanilla-DomÃŋnguez J.-Andina D. Marcano-CedeÃśo, A. 2011. WBCD breast cancer database classification applying artificial metaplasticity neural network. *Expert Systems with Applications* 38, 8 (2011), 9573 – 9579.

[18] Tarjan R. E. Paige, R. 1987. Three Partition Refinement Algorithms. *SIAM J. Comput.* 16, 6 (1987), 973–989.

[19] Güneş S. Polat, K. 2007. Breast cancer diagnosis using least square support vector machine. *Digital Signal Processing* 17, 4 (2007), 694 – 701.

[20] J. R. Quinlan. 1996. Improved use of continuous attributes in C4. 5. *Journal of artificial intelligence research* 4 (1996), 77–90.

[21] A. M. Sagir and S. Sathasivam. 2017. A Hybridised Intelligent Technique for the Diagnosis of Medical Diseases. *Pertanika Journal of Science & Technology* 25, 2 (2017).

[22] Asghar-S. Zafar A. Gillani S. Tahir, M. A. U. H. 2016. A Hybrid Model to Detect Phishing-Sites Using Supervised Learning Algorithms. In *2016 International Conference on Computational Science and Computational Intelligence (CSCI)*. 1126–1133.

[23] Mohammad R. M. McCluskey L. Thabtah, F. 2016. A dynamic self-structuring neural network model to combat phishing. In *2016 International Joint Conference on Neural Networks (IJCNN)*. 4221–4226.

[24] E. D. Übeyli. 2007. Implementing automated diagnostic systems for breast cancer detection. *Expert Systems with Applications* 33, 4 (2007), 1054–1062.

[25] Denny Zhou, Jiayuan Huang, and Bernhard Schölkopf. 2007. Learning with Hypergraphs: Clustering, Classification, and Embedding. In *Advances in Neural Information Processing Systems 19*, B. Schölkopf, J. C. Platt, and T. Hoffman (Eds.). MIT Press, 1601–1608.