FUZZ-EQ: a data equalizer for boosting the discrimination power of fuzzy classifiers

Mikel Uriz[a,b], Mikel Elkano[a,b], Humberto Bustince[a,b], Mikel Galar[a,b]

[a] *Department of Statistics, Computer Science, and Mathematics, Universidad Pública de Navarra, 31006 Pamplona, Spain*
[b] *Institute of Smart Cities, Universidad Pública de Navarra, 31006 Pamplona, Spain*

Abstract

The definition of linguistic terms is a critical part of the construction of any fuzzy classifier. Fuzzy partitioning methods (FPMs) range from simple uniform partitioning to sophisticated optimization algorithms. In this paper we present FUZZ-EQ, a preprocessing algorithm that facilitates the construction of meaningful fuzzy partitions regardless of the FPM used. The proposed approach is radically different from any existing FPM: instead of adjusting the fuzzy sets to the training data, FUZZ-EQ adjusts the training data to a hypothetical uniform partition before applying any FPM. To do so, the original data distribution is transformed into a uniform distribution by applying the probability integral transform. FUZZ-EQ allows FPMs to provide classifiers with more granularity on high density regions, increasing the overall discrimination capability. Additionally, we describe the procedure to reverse this transformation and recover the interpretability of linguistic terms. To assess the effectiveness of our proposal, we conducted an extensive empirical study consisting of 41 classification tasks and 9 fuzzy classifiers with different FPMs, rule induction algorithms, and rule structures. We also tested the scalability of FUZZ-EQ in Big Data classification problems such as HIGGS, with 11 million examples. Experimental results reveal that FUZZ-EQ significantly boosted the classification performance of those classifiers using the same linguistic terms for all rules, including state-of-the-art classifiers such as FARC-HD or IVTURS.

Keywords: Fuzzy Partitioning, Preprocessing, Fuzzy Rule-Based Classification Systems, Fuzzy Decision Trees, Probability Integral Transform, Quantile Function

1. Introduction

Fuzzy logic has provided machine learning algorithms with the ability to deal with uncertainty and create soft boundaries that enhance classification performance. The usage of linguistic terms also makes fuzzy classifiers a powerful tool for building human-readable models composed of IF-THEN rules. These models allow the system to explain the reasoning behind its predictions and describe how outputs are

Email addresses: mikelxabier.uriz@unavarra.es (Mikel Uriz), mikel.elkano@unavarra.es (Mikel Elkano), bustince@unavarra.es (Humberto Bustince), mikel.galar@unavarra.es (Mikel Galar)

inferred from inputs. In this work, we focus on two well-known types of fuzzy classifiers, i.e., Fuzzy Rule-Based Classification Systems (FRBCSs) and Fuzzy Decision Trees (FDTs), which have been used in several applications [5, 36].

Selecting a suitable strategy for defining linguistic terms is a fundamental aspect for any fuzzy classifier. Every term is defined by a membership function that determines the membership degree of an input value to the fuzzy set associated with the term. In many applications, this function is given by an expert in the field. However, there are situations where no experts are involved and the system has to automatically define the membership functions. In classification tasks, the discrimination capability of classifiers is to a great extent limited by the ability of linguistic terms to represent the data in a meaningful way. For example, if we consider the terms "Low", "Medium", and "High" to represent the variable "Temperature" ranging from -10°C to 40°C and 99% of the observed values range from 13°C to 26°C, there would be no point in uniformly distributing the three terms across the domain. In such a context, an expert would probably consider 13°C and 26°C as "Low" and "High", respectively, instead of assigning the term "Medium" to both values.

To assess the ability of linguistic terms to represent the data, Pedrycz formalized the concept of *data meaningfulness* of fuzzy sets [23]. The author claims that linguistic terms emerge only if there is enough experimental evidence (observed values) to justify their existence. According to this statement, the definition of membership functions should depend on the empirical data distribution. To adjust the position and/or shape of the fuzzy sets, different fuzzy partitioning methods (FPMs) have been proposed [1, 6, 7, 11, 12, 15, 27, 28, 10, 21, 26, 32, 31, 37]. In the context of classification, some FPMs focus on maximizing accuracy by optimizing partition parameters with respect to some criteria [6, 12, 27, 28, 10, 21, 26, 32, 31, 37]. Others prioritize interpretability over accuracy and build fixed uniform partitions with predefined parameters [7, 11, 15]. The combination of both approaches is also rather common among classifiers that focus on the tradeoff between accuracy and interpretability [1, 27, 28]. Besides, FPMs might differ in the stage they are applied. Some of them create the fuzzy partitions before the learning process [7, 11, 15, 10, 21, 26, 32, 31, 37], while others are part of the learning algorithm itself [1, 12, 27, 28, 6].

In this paper, we look at the problem from a different perspective. Instead of adjusting the fuzzy sets to the training data, we attempt to facilitate the work of FPMs by previously adjusting the training data to a hypothetical uniform partition. The proposed FUZZ-EQ algorithm consists in transforming the original data distribution into a uniform distribution by means of the *probability integral transform theorem* [3, 24]. More specifically, FUZZ-EQ equalizes the histogram of each variable so that the fuzzy sets contained in a trivial uniform partition would have the same experimental evidence. This transformation allows FPMs to provide the classifier with more granularity on high density regions, boosting discrimination capability. To the best of our knowledge, this is the first data preprocessing algorithm designed for improving the discrimination capability of fuzzy partitions.

The main contributions of our proposal are the following.

- *Versatility and flexibility.* FUZZ-EQ supports any type of classifier regardless of the FPM used, since the transformation takes place beforehand.
- *No need for any optimization process.* The FPMs that obtain the greatest performance gain are those based on uniform partitions, since FUZZ-EQ implicitly adapts the uniformly distributed fuzzy sets to the training data.
- *Interpretability.* Any data point in the transformed space can be mapped back to the original space, allowing the user to recover the interpretability of fuzzy sets on the original dataset. Furthermore, when the transformation is reversed, we can clearly observe how the shape and position of fuzzy sets are adjusted to the data distribution. This adaptability usually cause membership functions to present nonlinearities that might help partitions better represent the experimental evidence.

To assess the effectiveness of our proposal, we conducted an extensive empirical study consisting of 41 classification tasks available at UCI [18] and KEEL [2] repositories. We included 9 fuzzy classifiers with different FPMs, rule induction algorithms, and rule structures, namely CHI [7], FARC-HD [1], FHGBML [15], FURIA [12], FuzzyID3 [29], GFS-GP [27], IVTURS [28], PDFC [6], and SLAVE [11]. The results revealed that FUZZ-EQ was able to boost classification performance in all cases where fuzzy partitions were common to all rules. Furthermore, in those classifiers where no performance gain was observed (FHGBML, FURIA, and PDFC), the average classification performance was maintained. The scalability of FUZZ-EQ was also tested in 6 Big Data classification problems using CHI-BD [9] as the classifier. The source code of FUZZ-EQ is publicly available at GitHub[1] under the MIT License.

This paper is organized as follows. Section 2 includes the basics of FRBCSs and FDTs and recalls the concept of data meaningfulness of fuzzy sets. In Section 3, we introduce the proposed preprocessing algorithm (FUZZ-EQ). Sections 4 and 5 show the experimental framework and the analysis of the empirical results, respectively. Finally, Section 6 presents concluding remarks.

2. Preliminaries

In this section we briefly describe the basics of fuzzy classifiers and more specifically of Fuzzy Rule-Based Classification Systems (Section 2.1) and Fuzzy Decision Trees (Section 2.2). Additionally, we stress the importance of building meaningful fuzzy partitions for classification tasks (Section 2.3).

A classification problem consists in building a model which predicts the class (category) that a new example (observation) belongs to, based on a training set $\mathbb{X}$ composed of N labeled examples whose class is

[1] `https://github.com/melkano/fuzz-eq`

known. Each example $x = (x_1, \ldots, x_F)$ belongs to a class $y \in \mathbb{C} = \{c_1, c_2, ..., c_M\}$ (M being the number of classes in the problem) and is characterized by a set of F variables (also known as features) where x_i can take any value contained in the set $\mathcal{F}_i$. Thus, the construction of a classifier consists in finding a decision function (boundary) $h : \mathcal{F}_1 \times \ldots \times \mathcal{F}_F \rightarrow \mathbb{C}$ that outputs the class of the examples.

2.1. Fuzzy Rule-Based Classification Systems

Fuzzy Rule-Based Classification Systems (FRBCSs) are popular machine learning algorithms that achieve good trade-offs between classification performance and interpretability by means of a set of IF-THEN fuzzy rules [13]. These rules are used by the fuzzy reasoning method to classify new examples based on the activation strength of the antecedent parts. The antecedent of a rule is given by fuzzy sets representing linguistic terms which are constructed by the fuzzy partitioning method (FPM). In general, the structure of fuzzy rules is the following:

$$\text{Rule } R_j : \text{ If } x_1 \text{ is } A_{j1} \text{ and } \ldots \text{ and } x_F \text{ is } A_{jF} \text{ Then Class} = c_j \text{ with } RW_j \tag{1}$$

where R_j is the label of the j-th rule, $x = (x_1, \ldots, x_F)$ is an F-dimensional pattern vector that represents the example, A_{jf} is a linguistic label modeled by a membership function, c_j is the class label, and RW_j is the rule weight. In some cases rules might contain *don't care* linguistic labels making the classifier to ignore the corresponding attribute value. These labels can simply be removed from the RB, leading to variable rule lengths. Regarding the rule weight, it represents the confidence that the classifier has in the rule and the importance of the rule in the prediction. There exist different methods to compute rule weights, such as the Certainty Factor (CF) or the Penalized Certainty Factor (PCF) [14].

2.2. Fuzzy Decision Trees

A Decision Tree (DT) [25] is a directed acyclic graph where each internal node is a test on an attribute, each branch represents the outcome of the test, and each terminal node (or leaf) contains the final decision (class label). The construction of a DT consists in recursively partitioning the attribute space and selecting the attribute that maximizes the level of homogeneity of the class labels contained in the child nodes with respect to the parent node. For continuous attributes, either brute-force solutions or discretization strategies can be applied. The former test all the possible cut points in the training set, while the latter divide the attribute domain into a discrete set of intervals (also called *bins*).

Fuzzy decision trees (FDTs) [33, 16] make use of fuzzy partitions to characterize continuous attributes instead of considering a discrete set of intervals. As a consequence, a given input value might belong to one or more fuzzy sets with a certain membership degree and activate multiple branches at the same time. Fuzzy partitions allow FDTs to handle smooth transitions between adjacent intervals in continuous attributes, which might lead to more accurate predictions when handling numeric data.

In the experimental study, we interchangeably refer to antecedents/internal nodes and rules/leaves because a leaf can be represented by a set of IF-THEN rules whose antecedents correspond to all the internal nodes forming each of the paths to the leaf. In fact, the procedure used by FDTs to classify examples is usually the same as that of FRBCSs.

2.3. Data meaningfulness of fuzzy sets

The construction of membership functions is essential for the discrimination capability of fuzzy classifiers. To illustrate their role, Fig. 1 shows a fuzzy partition $\mathbb{A} = \{A_1, A_2, A_3, A_4, A_5\}$ composed of 5 fuzzy sets defined for the variable *FAsym* of the *MAGIC Gamma Telescope* dataset[2]. If we consider a trivial uniform partition without applying any optimization process, it is clear that in this case the majority of data points lie on the support of the middle fuzzy set. As a result, some linguistic labels are not represented by any data point at all and their contribution to decision making will be negligible or nonexistent.

The idea of *data meaningfulness* of fuzzy sets was formalized by Pedrycz in [23]. The author used the concept of probability of *fuzzy events* introduced by Zadeh [34] to construct *data-justifiable* fuzzy partitions. The proposed algorithm assumes that linguistic terms emerge only if there is experimental evidence (data) that justifies their existence. To assess whether a fuzzy set A_i is experimentally meaningful for a set of observed values $\mathbb{X} = \{x_1, x_2, \ldots, x_N\}$, the cumulative probability of A_i is computed as:

$$P(A_i) = \frac{1}{N}\sum_{j=1}^{N} \mu_{A_i}(x_j) = \sum_{x \in supp(A_i)} \mu_{A_i}(x)p(x), \tag{2}$$

where $p(x)$ is the probability mass function (PMF) of $\mathbb{X}$ and $supp(A_i)$ is the support of A_i defined as $supp(A_i) = \{x \in \mathbb{X} \mid \mu_{A_i}(x) > 0\}$. For a continuous random variable $\mathbb{X}$, this probability is also determined by integrating over the support of the fuzzy set, but in this case replacing the PMF with the corresponding probability density function (PDF):

$$P(A_i) = \int_x \mu_{A_i}(x)p(x)dx. \tag{3}$$

In order for fuzzy sets to be equally meaningful and carry the same amount of experimental evidence, all of them must have the same cumulative probability:

$$P(A_1) = P(A_2) = \ldots = P(A_L) = \frac{1}{L}. \tag{4}$$

The algorithm proposed by Pedrycz [23] iteratively optimizes the arguments of membership functions to fulfill Eq. (4). Besides Pedrycz, a few other authors proposed considering the histogram or the PMF/PDF of the data to construct data-meaningful fuzzy partitions [21, 26, 32, 31]. All of them attempt to optimize the arguments of either trapezoidal or triangular membership functions to adjust fuzzy sets to the PMF/PDF

[2] `https://archive.ics.uci.edu/ml/datasets/magic+gamma+telescope`

of the data. In this paper, we propose a different approach to avoid any optimization process: instead of adjusting the fuzzy sets to the data distribution, we transform the original data distribution to obtain meaningful partitions without the need for adjusting the fuzzy sets.

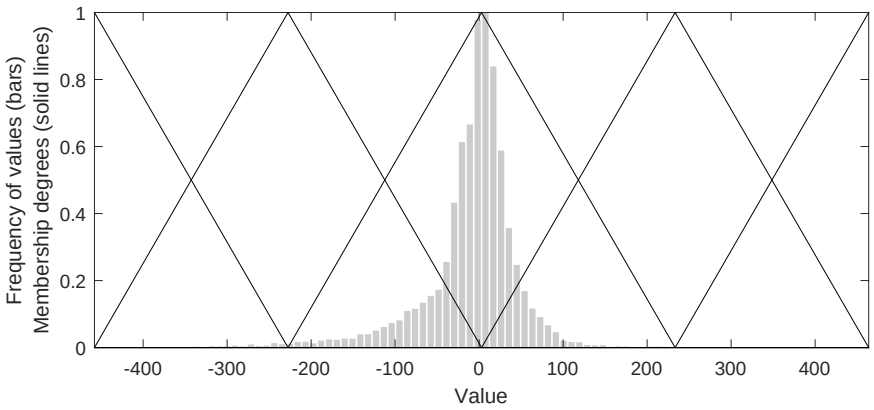


Figure 1: Uniform fuzzy partition for *FAsym* on *mag*.

3. FUZZ-EQ: equalizing data to construct data-meaningful fuzzy partitions that boost classification performance

In this work, we present a preprocessing algorithm for boosting the discrimination capability of fuzzy classifiers. The core idea of the proposed algorithm (called FUZZ-EQ) is to maximize the data meaningfulness of a hypothetical uniform partition, which would be built by simply distributing a number of equally-shaped triangles across the domain. We focus on improving naive uniform partitions because we aim to design a preprocessing algorithm that implicitly adjusts the position and shape of the fuzzy sets even when the subsequent FPM does not involve any optimization process [7, 11] or start from a uniform partition [1, 28]. For this reason, assuming the uniform distribution is the ideal case for a uniform partitioning in terms of data meaningfulness, FUZZ-EQ converts the original data distribution into a uniform distribution by means of the *probability integral transform theorem* [3, 24]. After this transformation, all the fuzzy sets are equally meaningful and have the same experimental evidence, providing the classifier with more granularity on high density regions.

The main differences between the existing FPMs mentioned in Section 2.3 and our method are the following.

- The contribution of this work is not a new FPM, but a preprocessing algorithm that boosts the data meaningfulness of fuzzy partitions built by existing FPMs. The proposed algorithm runs right before the corresponding fuzzy classifier (either for training or test) and does not modify any of its components at all (including the FPM). This means that FUZZ-EQ automatically improves fuzzy partitions in a transparent way without the need for any adaptation, which is clearly one of its main benefits.

- The final shape of the fuzzy sets not only depends on the parameters of the membership functions, but also on the data distribution. In this way, even if the FPM optimizes only linear functions, the resulting fuzzy sets in the original feature space might present nonlinearities that allow the partition to create a more accurate representation of the data.

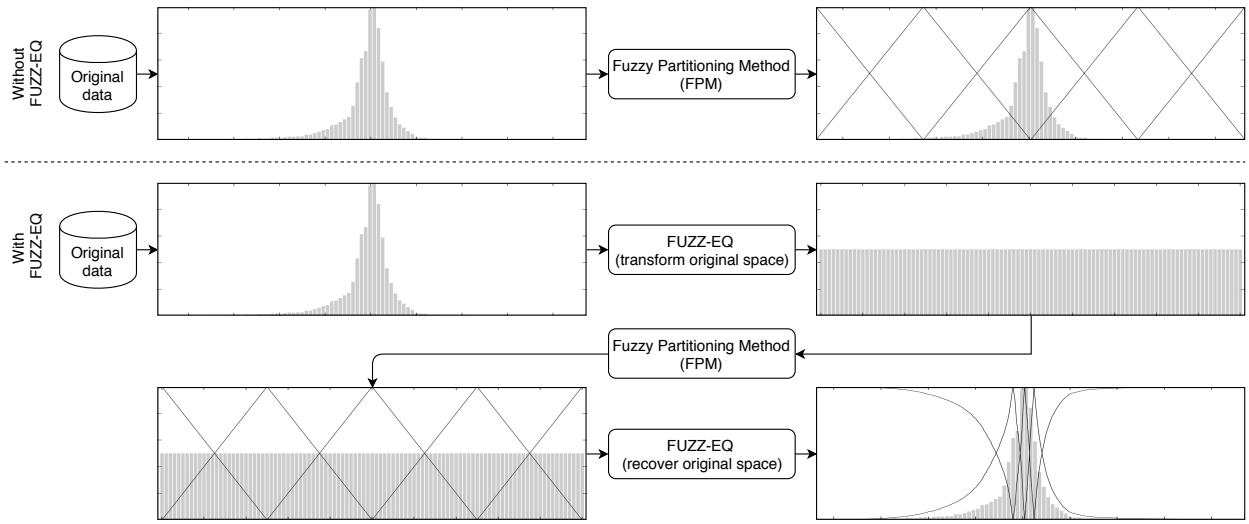


Figure 2: Illustrative example of the difference between applying and not applying FUZZ-EQ.

FUZZ-EQ comprises two stages:

1. *Transformation stage.* Converts the original distribution of the training data into a uniform distribution that help FPMs construct data-meaningful fuzzy partitions. The FPM and the corresponding classifier can run directly on the transformed dataset without any modification.

2. *Translation stage* (optional). If the interpretability of the constructed fuzzy partitions and models needs to be recovered, FUZZ-EQ provides an optional *translator* that brings any fuzzy set or data point back to the original space.

Fig. 2 illustrates the whole pipeline with and without FUZZ-EQ. The source code is publicly available at GitHub[3] under the MIT License.

3.1. Transformation stage

The purpose of this transformation is to equalize the histogram of each variable so that a trivial uniform partitioning would yield a partition composed of equally meaningful fuzzy sets (Fig. 3), according to Eq. (4). To this end, FUZZ-EQ applies the *probability integral transform theorem* [3, 24]. This theorem implies that any dataset can be transformed into a new dataset where all the variables follow a uniform distribution,

[3] `https://github.com/melkano/fuzz-eq`

regardless of the original distribution. Since this transformation cannot be applied directly when the original distribution (and consequently the exact CDF) is unknown, we propose computing the q-quantiles of the training set to obtain an approximate CDF for each variable. To this end, all the values are sorted to compute all the quantiles. High values of q lead to a more accurate approximation of the CDF but it might be too computationally expensive in large datasets. For this reason, we set a maximum number of quantiles q_{max} to be computed (in this paper $q_{max} = 1000$). When the number of examples (N) is greater than q_{max}, the CDF of a certain value is linearly interpolated on the interval $[Q_{i-1}, Q_i]$, Q_i being the first quantile greater than the value. If the value is smaller than the first quantile (Q_1) or greater than the last quantile (Q_q), the CDF is 0 or 1, respectively. Of course, the transformation of the testing set is performed by interpolating the CDF using the quantiles extracted from the training set.

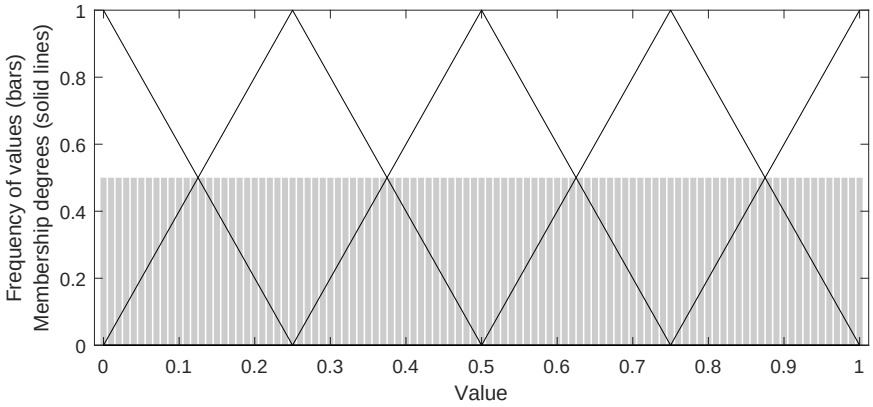


Figure 3: A trivial uniform fuzzy partitioning for a uniformly distributed variable. In this case, all fuzzy sets have the same cumulative probability (they are equally meaningful).

Besides approximating the original CDF, FUZZ-EQ must deal with another issue: the probability integral transform theorem only applies to continuous random variables. The discrete nature of digital data might cause a concentrated mass distribution of the training data, which would result in a transformed distribution that moves away from the uniform distribution [20]. In other words, when there is a high proportion of duplicates, the uniformity of quantiles might not be hold. This will negatively affect the data meaningfulness of the fuzzy sets and will reduce the discrimination capability of the partition.

To deal with this situation, we could assess whether the transformed distribution is sufficiently uniform based on a given statistical distance. If the distance was greater than a certain threshold, we would omit the transformation and work directly on the original space. Since we attempt to boost classification performance, we can get rid of thresholds by introducing entropy-based measures. In the context of classification tasks, entropy measures the homogeneity of class labels in a set of examples. We say that a given set has maximum entropy when all classes are equally likely to be represented in the set. On the other hand, the minimum entropy is reached when the examples of a set always belong to the same class. Therefore, if the transformation results in the examples of different classes being divided into different fuzzy sets, the

discrimination capability of the partition will be probably increased. Based on this assumption, FUZZ-EQ will finally transform the variable only if the entropy decreases after transforming it. To implement this procedure, we use the following concepts:

- Probability of the α-cut of a fuzzy set A_i: instead of considering the probability of the fuzzy set (Eq. (2)), we compute the probability of the α-cut of A_i as follows:

$$P(A_i^\alpha) = \sum_{x \in A_i^\alpha} p(x), \tag{5}$$

  where $A_i^\alpha = \{x \in \mathbb{X} \mid \mu_{A_i}(x) \geq \alpha\}$ and $p(x)$ is the probability mass function of $\mathbb{X}$. This probability determines the weight or importance of the fuzzy set in the computation of the partition entropy.

- Frequency of a class c_m in the α-cut of a fuzzy set A_i: represents the probability that an example of the class c_m belongs to the α-cut of A_i.

$$f\left(c_m, A_i^\alpha\right) = \frac{\mid \{x^\alpha \in A_i^\alpha \mid class(x^\alpha) = c_m\} \mid}{\mid A_i^\alpha \mid}, \tag{6}$$

  where $class(x^\alpha)$ is the class label of the example x^α.

- Entropy of the α-cut of a fuzzy set A_i: although the entropy of fuzzy sets is usually measured considering the sum of membership degrees, we use crisp cardinalities to be more restrictive when assessing the improvement obtained after transforming the data.

$$H^\alpha(A_i^\alpha) = \sum_{m=1}^{M} -f\left(c_m, A_i^\alpha\right) \cdot log_2\left(f\left(c_m, A_i^\alpha\right)\right). \tag{7}$$

- Entropy of a fuzzy partition $\mathbb{A}$: corresponds to the weighted average of the entropy of the α-cut of all the fuzzy sets belonging to the partition.

$$H(\mathbb{A}) = \sum_{i=1}^{L} P(A_i^\alpha) \cdot H^\alpha(A_i^\alpha). \tag{8}$$

This procedure allows FUZZ-EQ to selectively transform each of the variables depending on the decrease of entropy observed when building a trivial uniform partition in the transformed space. For the experimental study the number of fuzzy sets was set to 3 with $\alpha = 0.5$. Fig. 4 shows the pseudo-code of this stage.

3.2. Translation stage

Although any classifier can directly work with the data transformed by FUZZ-EQ and automatically obtain enhanced classification performance, fuzzy sets need to be brought back to the original space to recover their interpretability. To this end, FUZZ-EQ provides an optional *translator* tool that applies the *inverse cumulative distribution function* or *quantile function* [22] to reverse the transformation performed

Function transform_training_set ($\mathbb{X}$)

Input: Original training set $\mathbb{X}$.

Output: Transformed training set $\mathbb{X}^{tran}$.

Begin

 # Set the number of quantiles (q) based on the number of training examples (N)

if $N > q_{max}$ **then**

  $q \leftarrow q_{max}$

else

  $q \leftarrow N$

end if

 # Transform each variable separately ($\mathbb{X}_f$ contains the value of the variable f for each example)

$\mathbb{X}^{tran} \leftarrow \{\}$

for each $f \in \mathcal{F}$ **do**

  # Compute the q-quantiles

  $\mathbb{X}_f^{sorted} \leftarrow sort\ (\mathbb{X}_f)$

  $quantiles \leftarrow get\_quantiles\ (\mathbb{X}_f^{sorted})$

  # Transform the value of the variable f for each example

  $\mathbb{X}_f^{tran} \leftarrow \{\}$

  for each $x_f \in \mathbb{X}_f$ **do**

    $quantileIdx \leftarrow get\_quantile\_index\ (x_f)$

    if $quantileIdx = 1$ **then**

      $y \leftarrow 1/q$

    else if $quantileIdx = q$ **then**

      $y \leftarrow 1$

    else

      $x1 \leftarrow quantiles\ (quantileIdx - 1)$

      $y1 \leftarrow quantileIdx\ /\ q$

      $x2 \leftarrow quantiles\ (quantileIdx)$

      $y2 \leftarrow (quantileIdx + 1)\ /\ q$

      $m \leftarrow (y2 - y1)\ /\ (x2 - x1)$

      $y \leftarrow m\ (x_f - x1) + y1$

    end if

    $\mathbb{X}_f^{tran} \leftarrow \mathbb{X}_f^{tran}\ \cup\ y$

  end for

  # Compute the entropy of a uniform partition in the original ($\mathbb{A}$) and the transformed ($\mathbb{A}'$) spaces

  $$H(\mathbb{A}) \leftarrow \sum_{i=1}^{L} \left(P(A_i^{\alpha}) \cdot \sum_{m=1}^{M} -f\left(c_m, A_i^{\alpha}\right) \cdot log_2\left(f\left(c_m, A_i^{\alpha}\right)\right) \right)$$

  $$H(\mathbb{A}') \leftarrow \sum_{i=1}^{L} \left(P(A_i'^{\alpha}) \cdot \sum_{m=1}^{M} -f\left(c_m, A_i'^{\alpha}\right) \cdot log_2\left(f\left(c_m, A_i'^{\alpha}\right)\right) \right)$$

  # If the entropy has not decreased, do not transform the values of the variable f

  if $H(\mathbb{A}') >= H(\mathbb{A})$ **then**

    $\mathbb{X}_f^{tran} \leftarrow \mathbb{X}_f$

  end if

  $\mathbb{X}^{tran} \leftarrow concatenate\ (\mathbb{X}^{tran},\ \mathbb{X}_f^{tran})$

end for

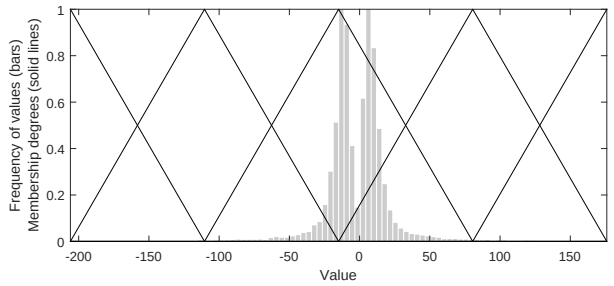
RETURN $\mathbb{X}^{tran}$

End

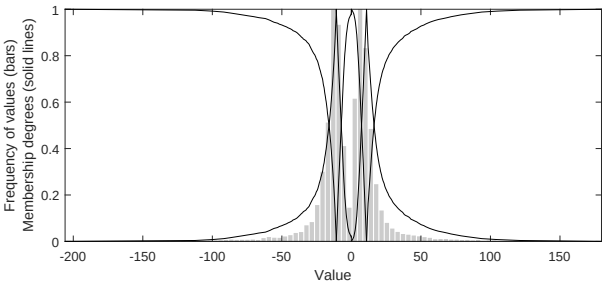
Figure 4: Pseudo-code of the proposed FUZZ-EQ preprocessing algorithm (transformation stage).

in the previous stage. To bring a given point back from the transformed space, the corresponding value is linearly interpolated between the two closest quantiles by computing the inverse of the linear function used to compute the CDF. It is worth noting that FUZZ-EQ supports any family of membership functions.

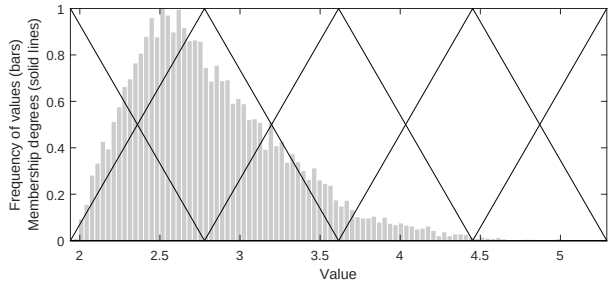
Fig. 5 shows an illustrative example of how fuzzy sets are distributed when applying a trivial uniform partitioning with and without FUZZ-EQ. Solid lines and bar plots represent the membership functions of the fuzzy sets and the original distribution of the variables, respectively.



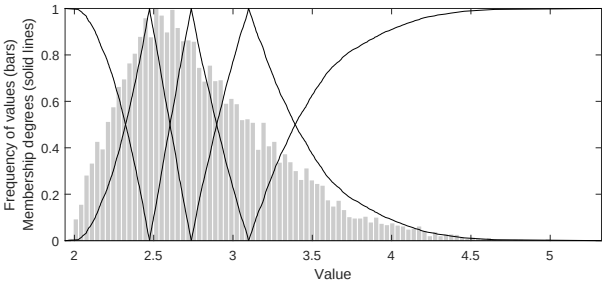
(a) Uniform partition for the variable *FM3Trans* (without FUZZ-EQ)



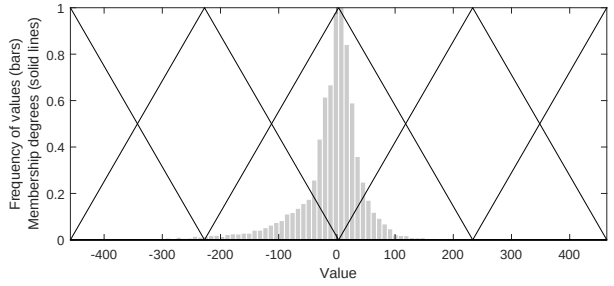
(b) Uniform partition for the variable *FM3Trans* (with FUZZ-EQ)



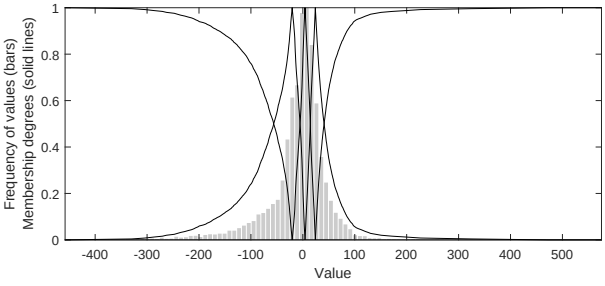
(c) Uniform partition for the variable *FSize* (without FUZZ-EQ)



(d) Uniform partition for the variable *FSize* (with FUZZ-EQ)



(e) Uniform partition for the variable *FAsym* (without FUZZ-EQ)



(f) Uniform partition for the variable *FAsym* (with FUZZ-EQ)

Figure 5: Fuzzy partitions built for *FM3Trans*, *FSize*, and *FAsym* on the *mag* dataset with and without FUZZ-EQ.

3.3. Computational complexity

Based on Fig. 4, one can estimate the computational complexity of FUZZ-EQ:

- Iterate over the features: $O(F)$
  - Sort the values of the corresponding feature: $O(N \cdot log(N))$
  - Get the quantiles: $O(N)$
  - Transform the values: $O(N)$
  - Compute the entropy of each fuzzy set: $O(L \cdot M)$

This results in $O(F \cdot (N \cdot log(N) + 2N + L \cdot M))$, which can be simplified to $O(F \cdot N \cdot log(N))$, assuming that L (number of fuzzy sets) and M (number of classes) are much smaller than F (number of features) or N (number of examples). Therefore, the complexity of FUZZ-EQ will be given by the corresponding sorting algorithm ($O(N \cdot log(N))$) when $N >> F$ and by the number of iterations ($O(F)$) when $F >> N$. It is worth noting that both complexity factors can be reduced by parallelizing either the feature transformation or the sorting process. In fact, in Section 5.2 we propose a distributed version of FUZZ-EQ callled FUZZ-EQ-Spark which is able to deal with Big Data problems.

4. Experimental framework

This section introduces the framework used to conduct the experimental study presented in Section 5. In particular, we show a brief description of the datasets (Section 4.1) and methods (Section 4.2) considered in the study and describe the performance measures and statistical tests used to assess the effectiveness of our proposal.

4.1. Datasets

The experimental study consists of 41 classification tasks available at UCI [18] and KEEL [2] repositories. Table 1 provides the description of the datasets showing the number of examples (#Examples), real (R)/integer(I)/categorical(C)/total(T) features (#Features), and classes (#Classes). The performance of all methods was assessed with a *5-fold stratified cross-validation scheme*, where each dataset is randomly split into five equal-sized partitions of data and the model is trained with a combination of four of them (80%) and tested with the remaining partition. Therefore, the result of each dataset was computed as the average of the five partitions.

4.2. Methods, performance metrics, and statistical tests

We included 9 fuzzy classifiers from the KEEL repository with different FPMs, rule induction algorithms, and rule structures. As for the parameters used in the experiments, we set their values based on the recommendations from the authors. Next, we include a brief description of each algorithm, with special attention to the FPM used.

Table 1: Description of the datasets.

ID	Dataset	#Examples	#Features (R/I/C)/T	#Classes
app	appendicitis	106	(7/0/0)/7	2
bal	balance	625	(4/0/0)/4	3
ban	banana	5300	(2/0/0)/2	2
bup	bupa	345	(1/5/0)/6	2
con	contraceptive	1473	(0/9/0)/9	3
eco	ecoli	336	(7/0/0)/7	8
gla	glass	214	(9/0/0)/9	7
hab	haberman	306	(0/3/0)/3	2
hay	hayes-roth	160	(0/4/0)/4	3
hea	heart	270	(1/12/0)/13	2
ion	ionosphere	351	(32/1/0)/33	2
iri	iris	150	(4/0/0)/4	3
led	led7digit	500	(7/0/0)/7	10
let	letter	20000	(0/16/0)/16	26
mag	magic	19020	(10/0/0)/10	2
mon	monk-2	432	(0/6/0)/6	2
new	newthyroid	215	(4/1/0)/5	3
opt	optdigits	5620	(0/64/0)/64	10
pag	page-blocks	5472	(4/6/0)/10	5
pen	penbased	10992	(0/16/0)/16	10
pho	phoneme	5404	(5/0/0)/5	2
pim	pima	768	(8/0/0)/8	2
rin	ring	7400	(20/0/0)/20	2
sah	saheart	462	(5/3/1)/9	2
sat	satimage	6435	(0/36/0)/36	7
seg	segment	2310	(19/0/0)/19	7
son	sonar	208	(60/0/0)/60	2
spa	spambase	4597	(57/0/0)/57	2
spe	spectfheart	267	(0/44/0)/44	2
tae	tae	151	(0/5/0)/5	3
tex	texture	5500	(40/0/0)/40	11
thy	thyroid	7200	(6/15/0)/21	3
tit	titanic	2201	(3/0/0)/3	2
two	twonorm	7400	(20/0/0)/20	2
veh	vehicle	846	(0/18/0)/18	4
vow	vowel	990	(10/3/0)/13	11
wdb	wdbc	569	(30/0/0)/30	2
win	wine	178	(13/0/0)/13	3
wew	winequality-red	1599	(11/0/0)/11	11
wwh	winequality-white	4898	(11/0/0)/11	11
yea	yeast	1484	(8/0/0)/8	10

- *CHI* [7] is the simplest classifier included in the study. To build the rule base, CHI creates a new fuzzy rule for each training example using the conjunction of the fuzzy sets with the highest membership degrees. As for the FPM, CHI uses a fixed uniform partitioning.

- *FARC-HD* [1] is a fuzzy association rule-based classifier that creates fuzzy rules from the frequent itemsets extracted by the Apriori algorithm. The rule base is later optimized with an evolutionary algorithm that simultaneously performs rule selection and lateral tuning of linguistic labels. This is a clear example of an FPM which initially creates a trivial uniform partition that is later optimized during the learning process of the classifier.

- *FHGBML* [15] is a hybrid algorithm of Pittsburgh-style and Michigan-style fuzzy genetics-based machine learning approaches. FHGBML simultaneously uses four fuzzy partitions for each feature, each having 2, 3, 4, and 5 fuzzy sets, respectively. As a result, each antecedent can be associated with one of these 14 fuzzy sets plus a "don't care" label.

- *FURIA* [12] fuzzifies the rules extracted by RIPPER [8] and optimizes the rule base by means of a greedy algorithm. During the learning process, FURIA replaces the interval representing each antecedent with a trapezoidal membership function which is later optimized to improve the coverage degree of the rule. Therefore, the FPM used by FURIA generates an independent fuzzy partition for each rule.

- *FuzzyID3* [29] is an extension of the ID3 decision tree which uses fuzzy sets to select the test attribute based on the fuzzy information gain. Although the fuzzy sets used in the original paper were previously defined by experts without any FPM, in this study we used a trivial uniform partition.

- *GFS-GP* [27] uses genetic programming with tree-shaped genotypes where rules are represented as single chains in a context-free grammar by means of their parse tree or by a pair, composed by a syntactic tree and a chain of parameters. In this case, the FPM is included in an evolutionary learning process that simultaneously generates the rules and optimizes the fuzzy sets.

- *IVTURS* [28] extends the FARC-HD algorithm by introducing interval-valued fuzzy sets (IVFSs) to model the uncertainty and ignorance that might exist when defining membership functions. In IVFSs, the membership degree of an element to the set is an interval instead of a single number. The amplitude of such an interval can be considered as the ignorance related to the assignment of a single value to the membership function.

- *PDFC* [6] learns a Support Vector Machine to extract the support vectors along with their corresponding Lagrange multipliers. Each support vector is converted into a fuzzy rule, where fuzzy partitions are given by adding the values of the support vector to some reference membership functions and the

rule weight is determined by the Lagrange multiplier. Although the reference membership functions are uniform and common to all rules, fuzzy partitions are later optimized based on the corresponding support vector, and thus they are rule-specific.

- *SLAVE* [11] uses an iterative approach to extract the best fuzzy rule for each class at each iteration by means of a genetic algorithm. The original method uses a trivial uniform partition as CHI and FuzzyID3.

The way in which all these classifiers construct fuzzy partitions is key to understanding the variation in the effectiveness of FUZZ-EQ when applied with different classifiers. For this reason, Table 2 shows a summary of the FPM used by each classifier.

Regarding performance metrics, we decided to use the average accuracy rate per class (Acc_{Class}) and the geometric mean (GM) [4] because they consider the discrimination capability for each class, and thus they are suitable for the many multi-class problems considered in the experimental study. These metrics are defined as follows:

$$Acc_{Class} = \frac{\sum_{m=1}^{M} TPR_m}{M}; \qquad GM = \sqrt[M]{\prod_{m=1}^{M} TPR_m}, \tag{9}$$

where TPR_m is the true positive rate of class c_m (proportion of correctly classified examples belonging to class c_m), N_m is the number of examples from class c_m, and N is the total number of examples.

To find statistical evidence in the improvement obtained when applying FUZZ-EQ, we used the Wilcoxon signed-ranks test [30] to perform pairwise comparisons of the classification performance of each method with and without FUZZ-EQ.

Table 2: Description of the fuzzy partitions used by each method.

Algorithm	**#FS**	**Trainable**	**Shared**
CHI	3	No (fixed uniform partitions)	✓
FARC-HD	5	Position	✓
FHGBML	14	No (4 fixed uniform partitions per variable)	✓
FURIA	-	#Fuzzy sets, position, and shape	✗
FuzzyID3	5	No (fixed uniform partitions)	✓
GFS-GP	3	Position and shape	✓
IVTURS	5	Position and shape (interval-valued)	✓
PDFC	3	Position	✗
SLAVE	5	No (fixed uniform partitions)	✓

#FS: number of fuzzy sets per variable

Trainable: shows which property of the fuzzy sets is optimized

Shared: shows whether fuzzy partitions are shared by all rules

5. Experimental study

In this study, we assessed the effectiveness of our proposal in terms of classification performance gain by comparing the Acc_{Class} and GM achieved by each classifier when working with and without FUZZ-EQ. Additionally, we measured the effect of FUZZ-EQ on the rule base complexity and reported the additional execution time required to apply the proposed transformation on each dataset. Finally, we analyzed the scalability of FUZZ-EQ when dealing with Big Data.

5.1. Performance

Tables 3 and 4 show the Acc_{Class} and GM on the original and transformed datasets for each classifier. According to these results, transforming the data with FUZZ-EQ resulted in an overall performance gain, except for FHGBML, FURIA, and PDFC, where the transformation had little effect in general. As shown in Table 2, FURIA and PDFC are the only classifiers included in this study that build rule-specific fuzzy partitions. The fuzzy sets built by these methods are specifically optimized for each rule based on its discrimination capability. This leaves any previous process with little room for improvement in terms of classification performance. As for FHGBML, each rule can use any of the four fuzzy partitions for each feature, which allows the classifier to dynamically adapt the granularity in each case. This adaptability minimizes the benefits of adjusting the granularity of each region based on its probability mass, which is what FUZZ-EQ does. Nevertheless, the three classifiers which did not obtain any performance gain (FHGBML, FURIA, PDFC) were not negatively affected by FUZZ-EQ.

To assess whether the classification performance gain derived from the usage of FUZZ-EQ is statistically significant, we used the Wilcoxon's test to perform a pairwise comparison for each classifier (Table 5). In all cases, we compared the Acc_{Class} and GM when running the corresponding classifier with and without FUZZ-EQ. As expected, FHGBML, FURIA, and PDFC did not show any statistical difference. The classifiers that obtained the most significant performance gain were CHI, FARC-HD, IVTURS, and SLAVE, where Acc_{Class} and GM were both increased with $\alpha = 0.05$. As for GFS-GP, we found evidence of improvement with $\alpha = 0.05$ for the GM and $\alpha = 0.1$ for the Acc_{Class}. Finally, FuzzyID3 benefited from FUZZ-EQ in a significant way in terms of Acc_{Class} with $\alpha = 0.1$, but no difference was observed in GM.

Regarding the effect of FUZZ-EQ on the rule base complexity, Table 6 reports the average rule length and number of rules built by each classifier with and without FUZZ-EQ. We include a supplementary document with the results for each dataset. Overall, the classification performance gain derived from FUZZ-EQ was accompanied by an increase in the number of rules, while the average rule length was maintained. Although rule induction algorithms can greatly differ from each other, the fact that FUZZ-EQ increases the granularity of those fuzzy sets covering high probability mass regions causes rules to be more specific, needing more rules to correctly cover training examples. Interestingly, this increase in the number of rules bring along an

increase in the final performance, which shows the usefulness of FUZZY-EQ to make linguistic labels more meaningful.

Finally, we measured the time required for transforming each of the datasets with FUZZ-EQ. According to Table 7, the transformation runs on the order of milliseconds on an Intel Xeon E5-2620v2 processor with 6 physical/12 virtual cores at 2.1 GHz.

Table 3: Accuracy rate per class (Acc_{Class}) comparison between the original (ORG) and transformed (TRA) datasets.

	CHI		**FARC-HD**		**FHGBML**		**FURIA**		**FuzzyID3**		**GFS-GP**		**IVTURS**		**PDFC**		**SLAVE**	
Dataset	**ORG**	**TRA**	**ORG**	**TRA**	**ORG**	**TRA**	**ORG**	**TRA**	**ORG**	**TRA**	**ORG**	**TRA**	**ORG**	**TRA**	**ORG**	**TRA**	**ORG**	**TRA**
app	**80.56**	73.56	67.38	**72.97**	**63.45**	58.75	**74.97**	**74.97**	**71.65**	69.24	68.56	**72.47**	**76.06**	69.88	74.88	**76.06**	68.56	**72.24**
bal	**65.39**	65.16	63.08	**64.12**	45.15	**63.90**	**59.96**	**59.96**	47.92	**56.60**	**56.61**	56.14	62.28	**63.19**	81.17	**82.07**	47.34	**49.55**
ban	56.05	**69.22**	85.48	**85.68**	**82.46**	79.96	**88.67**	**88.67**	81.36	**87.29**	69.90	**72.67**	80.82	**85.34**	**90.22**	90.20	**77.45**	75.89
bup	51.57	**67.23**	**67.46**	66.99	56.98	**57.54**	**67.03**	**67.03**	62.34	**62.55**	55.98	**64.38**	64.59	**69.75**	70.91	**71.10**	53.52	**65.39**
con	42.59	**46.88**	49.61	**49.72**	27.80	**39.09**	**51.08**	**51.08**	44.88	**45.52**	45.69	**48.02**	49.76	**51.01**	**50.82**	50.26	40.15	**47.20**
eco	60.47	**68.60**	**73.53**	70.11	41.67	**46.25**	61.33	**61.80**	**71.05**	61.19	37.32	**52.32**	69.10	**70.27**	72.97	**74.56**	66.77	**68.52**
gla	40.65	**53.51**	56.48	**57.52**	33.46	**43.13**	**68.50**	66.61	16.52	**26.60**	**38.18**	35.10	**61.95**	57.43	66.55	**66.90**	53.03	**57.56**
hab	51.17	**57.41**	53.75	**57.87**	50.25	**50.64**	**61.35**	**61.35**	55.78	**57.81**	55.71	**56.30**	58.88	**62.01**	**57.20**	53.47	**57.54**	56.29
hay	61.19	**64.37**	77.13	**88.16**	52.34	**56.87**	**84.75**	**84.75**	**75.73**	**75.73**	69.80	**72.63**	82.09	**84.04**	**82.78**	**82.78**	**85.90**	85.33
hea	**68.42**	65.08	81.67	**82.33**	48.61	**56.92**	**82.25**	**82.25**	**78.42**	74.08	70.42	**76.33**	82.17	**84.58**	78.08	**81.08**	75.25	**77.08**
ion	**56.49**	54.99	**88.11**	87.94	**35.35**	34.30	**87.91**	87.69	85.49	**87.11**	66.35	**69.72**	**91.38**	88.27	**93.35**	93.11	79.35	**85.75**
iri	**93.33**	92.00	**94.67**	**94.67**	**84.67**	83.33	**93.33**	**93.33**	**96.00**	**96.00**	90.00	**93.33**	**95.33**	**95.33**	**94.67**	**94.67**	**96.67**	96.00
led	**61.71**	**61.71**	**70.03**	69.52	33.84	**40.24**	**72.14**	**72.14**	**67.81**	**67.81**	48.92	**49.05**	**70.28**	69.66	**68.38**	68.31	65.72	**66.26**
let	33.63	**76.78**	57.98	**68.67**	**11.90**	10.43	**91.03**	90.99	80.79	**89.42**	18.26	**26.24**	48.20	**63.50**	97.22	**97.41**	39.86	**75.04**
mag	68.41	**71.49**	**80.72**	80.40	**66.83**	56.81	**80.64**	**80.64**	79.62	**81.61**	73.14	**74.33**	**79.84**	78.78	**85.10**	85.05	75.19	**75.93**
mon	55.77	**81.42**	**100.00**	**100.00**	**97.37**	97.12	**100.00**	**100.00**	**100.00**	**100.00**	97.37	**98.02**	**99.78**	98.38	**100.00**	**100.00**	**100.00**	**100.00**
new	66.67	**89.75**	92.70	**94.16**	69.48	**93.81**	89.17	**89.40**	90.89	**94.32**	75.21	**87.78**	**93.43**	92.25	**97.33**	95.05	85.02	**88.73**
opt	**50.52**	29.66	93.38	**93.76**	**10.60**	10.37	**95.49**	**95.49**	**87.33**	86.78	**43.42**	40.23	94.37	**94.67**	**99.04**	98.95	82.35	**84.92**
pag	33.02	**66.64**	59.39	**72.16**	35.50	**45.29**	**83.72**	83.66	48.24	**65.18**	**51.56**	33.62	56.85	**73.40**	85.49	**87.99**	43.10	**55.95**
pen	**97.87**	97.31	96.14	**96.17**	**36.12**	33.01	**98.23**	**98.23**	**97.72**	97.47	**55.35**	52.08	**95.47**	95.06	99.61	**99.63**	97.30	**97.31**
pho	52.97	**72.24**	79.04	**80.63**	**69.51**	68.55	81.54	**81.57**	75.86	**82.34**	**73.41**	70.27	77.52	**79.12**	85.68	**87.58**	69.69	**75.55**
pim	64.02	**70.68**	72.42	**72.99**	**65.79**	65.71	**70.84**	70.74	**71.85**	67.10	67.12	**67.46**	71.85	**71.93**	**70.06**	70.05	**72.79**	70.75
rin	54.98	**61.51**	**93.72**	89.84	**63.90**	56.74	94.21	**94.25**	**91.04**	85.15	71.05	**75.06**	**90.04**	89.15	**97.19**	96.32	**91.23**	90.95
sah	**65.68**	61.62	62.66	**64.64**	**49.29**	49.18	64.35	**64.52**	61.34	**62.91**	**60.79**	60.75	**67.37**	65.21	60.92	**61.07**	61.18	**64.76**
sat	46.45	**83.30**	76.50	**84.00**	**55.65**	45.78	**86.20**	**86.20**	49.12	**60.40**	54.30	**67.08**	68.00	**81.37**	89.97	**91.34**	31.79	**67.69**
seg	86.23	**91.52**	93.33	**94.98**	**58.67**	49.89	**97.32**	97.27	92.12	**95.71**	67.14	**70.00**	90.74	**93.16**	**98.48**	98.44	89.52	**94.55**
son	**76.81**	59.91	**84.11**	78.53	26.33	**29.85**	**77.69**	**77.69**	**78.19**	69.10	66.58	**68.98**	**81.70**	80.98	**90.07**	87.19	**72.07**	69.75
spa	64.96	**77.27**	90.28	**92.21**	**51.45**	39.24	**93.20**	93.09	90.47	**91.30**	68.49	**80.76**	90.48	**92.42**	94.38	**95.00**	68.69	**89.42**
spe	56.31	**58.00**	61.84	**64.10**	**32.95**	31.76	**59.62**	**59.62**	53.49	**61.77**	53.55	**53.79**	63.89	**70.47**	**69.63**	68.32	**50.63**	50.41
tae	**56.15**	54.35	55.71	**57.78**	**46.62**	37.98	**45.91**	**45.91**	49.94	**51.88**	**58.17**	51.97	57.03	**57.18**	**62.23**	61.70	**54.29**	49.61
tex	73.82	**86.04**	93.18	**93.75**	**47.88**	34.23	**96.38**	96.36	94.35	**97.78**	54.65	**63.84**	86.42	**90.82**	**99.67**	99.58	76.02	**88.47**
thy	36.93	**57.36**	59.17	**90.23**	46.21	**52.95**	**98.47**	98.38	46.17	**67.67**	**53.23**	36.96	59.33	**89.71**	95.57	**96.20**	35.53	**70.68**
tit	**72.35**	**72.35**	**69.35**	**69.35**	**69.35**	**69.35**	**68.44**	**68.44**	**69.35**	**69.35**	69.33	**71.60**	**69.35**	**69.35**	**69.35**	**69.35**	**69.35**	69.19
two	90.55	**93.04**	**95.11**	93.89	64.54	**69.83**	**93.62**	93.58	**94.69**	89.22	78.25	**78.51**	**95.91**	94.50	**97.39**	97.36	92.43	**93.04**
veh	61.83	**64.55**	69.60	**70.45**	41.02	**42.25**	**74.56**	**74.56**	70.30	**70.78**	51.61	**53.75**	68.41	**70.82**	**80.25**	79.88	60.19	**72.67**
vow	53.23	**77.27**	74.75	**81.41**	19.07	**19.26**	81.72	**81.92**	92.32	**92.93**	26.36	**38.38**	64.44	**75.05**	98.59	**98.79**	71.62	**87.68**
wdb	**92.11**	91.58	94.68	**96.00**	**59.62**	57.35	**94.96**	**94.96**	94.30	**95.63**	**92.70**	89.83	94.44	**96.14**	97.07	**97.21**	91.84	**94.17**
win	94.59	**95.51**	**95.32**	93.54	**72.05**	69.02	**93.83**	**93.83**	**97.06**	92.10	**90.57**	89.50	95.32	**97.34**	**98.52**	98.08	93.58	**94.48**
wre	22.86	**27.86**	26.44	**27.43**	20.13	**20.49**	29.40	**29.55**	0.48	**3.78**	21.75	**22.88**	23.68	**27.20**	36.46	**37.21**	22.86	**28.68**
wwh	20.15	**27.26**	**21.43**	21.36	17.67	**18.07**	25.44	**25.46**	5.53	**6.73**	**17.35**	16.74	**21.56**	20.70	**39.55**	39.44	17.45	**23.48**
yea	34.54	**45.63**	**51.32**	46.13	**22.83**	21.26	**49.15**	**49.15**	**52.43**	41.53	**35.47**	15.48	38.20	**38.63**	51.82	**52.17**	**46.04**	39.56
AVG.	60.32	**67.84**	73.87	**76.00**	48.40	**48.94**	**77.28**	77.25	70.00	**71.65**	59.01	**60.35**	72.88	**75.56**	81.19	**81.24**	66.56	**72.35**

Table 4: Geometric mean (GM) comparison between the original (ORG) and transformed (TRA) datasets.

	CHI		**FARC-HD**		**FHGBML**		**FURIA**		**FuzzyID3**		**GFS-GP**		**IVTURS**		**PDFC**		**SLAVE**	
Dataset	**ORG**	**TRA**	**ORG**	**TRA**	**ORG**	**TRA**	**ORG**	**TRA**	**ORG**	**TRA**	**ORG**	**TRA**	**ORG**	**TRA**	**ORG**	**TRA**	**ORG**	**TRA**
app	**.7898**	.6941	.6115	**.6863**	**.4009**	.3593	**.7147**	**.7147**	**.6646**	.5593	.6196	**.6742**	**.7315**	.6435	.7216	**.7315**	.6196	**.6596**
bal	**.0000**	**.0000**	**.0000**	**.0000**	**.0000**	**.0000**	**.0000**	**.0000**	**.0000**	**.0000**	**.0000**	**.0000**	**.0000**	**.0000**	.7547	**.7696**	**.0000**	**.0000**
ban	.3750	**.6329**	.8514	**.8534**	**.8186**	.6837	**.8852**	**.8852**	.8056	**.8700**	.6493	**.7220**	.8041	**.8522**	**.9012**	.9010	**.7726**	.7461
bup	.1745	**.6464**	.6534	**.6557**	.3894	**.4716**	**.6549**	**.6549**	.5819	**.6168**	.4730	**.5953**	.6364	**.6884**	.7042	**.7082**	.3172	**.6306**
con	.3787	**.4447**	**.4763**	.4712	.0000	**.1618**	**.4825**	**.4825**	**.4423**	.4389	.3384	**.4307**	.4623	**.4845**	**.5001**	.4956	.3862	**.4547**
eco	.3633	**.4516**	**.4800**	.3383	**.0000**	**.0000**	**.1305**	**.1305**	**.4693**	.1541	**.0000**	**.0000**	.2795	**.4830**	**.4856**	.4851	.1379	**.3329**
gla	**.0000**	**.0000**	.0000	**.1135**	**.0000**	**.0000**	**.1346**	.0000	**.0000**	**.0000**	**.0000**	**.0000**	.0000	**.2397**	**.3828**	.2826	**.0000**	**.0000**
hab	.1474	**.4297**	.4016	**.4818**	.2709	**.2865**	**.4874**	**.4874**	.4059	**.5066**	.3949	**.4099**	.4700	**.5601**	**.4761**	.4389	**.4579**	.4298
hay	.4463	**.5001**	.7573	**.8747**	.0000	**.1491**	**.8404**	**.8404**	**.7426**	**.7426**	.6846	**.7046**	.8126	**.8357**	**.8183**	**.8183**	**.8493**	.8404
hea	**.6325**	.5981	.8093	**.8210**	.0000	**.1474**	**.8172**	**.8172**	**.7813**	.7372	.6953	**.7573**	.8172	**.8435**	.7795	**.8099**	.7498	**.7682**
ion	**.4086**	.3585	**.8792**	.8755	**.0000**	**.0000**	**.8760**	.8740	.8481	**.8674**	.5619	**.6411**	**.9130**	.8784	**.9313**	.9285	.7717	**.8471**
iri	**.9270**	.9107	.9432	**.9440**	**.5655**	.5516	**.9301**	**.9301**	**.9575**	**.9575**	.8952	**.9286**	**.9514**	**.9514**	**.9442**	**.9442**	**.9657**	.9588
led	**.4525**	**.4525**	**.1739**	.1722	**.0000**	**.0000**	**.7051**	**.7051**	**.5044**	**.5044**	**.0000**	**.0000**	**.1700**	.1690	**.5154**	.5145	.6378	**.6447**
let	.0000	**.7466**	.0000	**.6595**	**.0000**	**.0000**	**.9094**	.9090	.7981	**.8925**	**.0000**	**.0000**	.0000	**.5807**	.9720	**.9739**	.0000	**.4517**
mag	.6231	**.6667**	**.7977**	.7951	**.4555**	.1575	**.7958**	**.7958**	.7955	**.8129**	.7163	**.7315**	**.7891**	.7782	.8464	**.8469**	.7296	**.7395**
mon	.4410	**.8065**	**1.000**	**1.000**	**.9732**	.9708	**1.000**	**1.000**	**1.000**	**1.000**	.9732	**.9799**	**.9978**	.9837	**1.000**	**1.000**	**1.000**	**1.000**
new	.6175	**.8914**	.9235	**.9395**	.3204	**.9335**	.8817	**.8833**	.9036	**.9416**	.7100	**.8710**	**.9307**	.9180	**.9719**	.9470	.8355	**.8768**
opt	**.4555**	.2053	.9328	**.9367**	**.0000**	**.0000**	**.9545**	**.9545**	**.8715**	.8653	.0000	**.0527**	.9429	**.9459**	**.9903**	.9895	.6592	**.8442**
pag	.0350	**.6122**	.4101	**.4347**	**.0000**	**.0000**	**.8257**	.8251	.2130	**.6031**	**.0000**	**.0000**	.2085	**.5934**	.8454	**.8708**	.1374	**.3684**
pen	**.9785**	.9729	.9610	**.9613**	**.0000**	**.0000**	**.9822**	**.9822**	**.9770**	.9746	**.2107**	.0705	**.9542**	.9496	.9961	**.9963**	.9724	**.9727**
pho	.2636	**.6979**	.7856	**.8009**	**.5866**	.5760	.8098	**.8101**	.7513	**.8181**	**.7234**	.6709	.7721	**.7847**	.8541	**.8739**	.6707	**.7432**
pim	.5554	**.6910**	.7158	**.7192**	.5395	**.5437**	**.6874**	.6867	**.7026**	.6544	.6361	**.6460**	**.7062**	.7031	.6949	**.6964**	**.7127**	.6876
rin	.3129	**.4917**	**.9365**	.8974	.1788	**.2728**	.9414	**.9418**	**.9056**	.8485	.6977	**.7460**	**.8996**	.8904	**.9716**	.9628	**.9114**	.9086
sah	**.6056**	.5844	.5865	**.6212**	.2085	**.2230**	.5954	**.5965**	.5806	**.6142**	**.5519**	.5424	**.6488**	.6349	.5833	**.5870**	.5275	**.6061**
sat	.0000	**.8153**	.0000	**.8235**	**.0000**	**.0000**	.8478	**.8480**	**.0000**	**.0000**	.1137	**.3546**	.0000	**.7920**	.8934	**.9084**	**.0000**	**.0000**
seg	.8448	**.9092**	.9304	**.9481**	**.0000**	**.0000**	**.9728**	.9723	.9112	**.9558**	.1292	**.6009**	.9018	**.9288**	**.9847**	.9843	.8845	**.9433**
son	**.7439**	.4368	**.8396**	.7809	**.0000**	**.0000**	**.7737**	**.7737**	**.7792**	.6869	.6602	**.6854**	**.8151**	.8077	**.8972**	.8689	**.7116**	.6951
spa	.5620	**.7437**	.9017	**.9217**	**.0000**	**.0000**	**.9314**	.9302	.9042	**.9127**	.6601	**.8041**	.9039	**.9239**	.9435	**.9499**	.5799	**.8923**
spe	.4636	**.4679**	.5452	**.5819**	**.0000**	**.0000**	**.4950**	**.4950**	.3124	**.5604**	.1973	**.2845**	.5761	**.6795**	**.6560**	.6148	.0843	**.1185**
tae	**.4855**	.4608	.5308	**.5613**	**.1590**	.0986	**.2356**	**.2356**	.4526	**.4935**	**.5631**	.4979	.5216	**.5500**	**.6119**	.6000	**.5026**	.4578
tex	.6382	**.8490**	.9300	**.9355**	**.0000**	**.0000**	**.9633**	.9631	.9397	**.9776**	.0626	**.4707**	.8547	**.9037**	**.9967**	.9958	.2848	**.5459**
thy	.0372	**.3381**	.0000	**.8984**	**.0000**	**.0000**	**.9847**	.9838	.0000	**.6232**	**.0392**	.0000	.0000	**.8935**	.9545	**.9613**	.0000	**.4936**
tit	**.6903**	**.6903**	**.6305**	**.6305**	**.6305**	**.6305**	**.6170**	**.6170**	**.6305**	**.6305**	.6304	**.6755**	**.6305**	**.6305**	**.6305**	**.6305**	**.6305**	.6281
two	.9048	**.9304**	**.9511**	.9389	.1799	**.3516**	**.9362**	.9358	**.9469**	.8921	.7784	**.7810**	**.9590**	.9450	**.9739**	.9736	.9243	**.9304**
veh	.5723	**.5878**	.6403	**.6642**	**.0000**	**.0000**	**.6999**	**.6999**	**.6687**	.6634	.4115	**.4519**	.6348	**.6628**	**.7799**	.7753	.2484	**.6877**
vow	.0000	**.7472**	.6808	**.8015**	**.0000**	**.0000**	.8088	**.8109**	.9197	**.9270**	**.0000**	**.0000**	.5830	**.7334**	.9855	**.9876**	.6509	**.8636**
wdb	**.9205**	.9135	.9460	**.9597**	**.0000**	**.0000**	**.9490**	**.9490**	.9414	**.9556**	**.9244**	.8956	.9435	**.9610**	.9702	**.9718**	.9155	**.9415**
win	.9438	**.9537**	**.9521**	.9337	**.1895**	.1678	**.9379**	**.9379**	**.9698**	.9162	**.8990**	.8919	.9522	**.9724**	**.9849**	.9801	.9341	**.9429**
wre	**.0000**	**.0000**	**.0000**	**.0000**	**.0000**	**.0000**	**.0000**	**.0000**	**.0000**	**.0000**	**.0000**	**.0000**	**.0000**	**.0000**	**.0000**	**.0000**	**.0000**	**.0000**
wwh	**.0000**	**.0000**	**.0000**	**.0000**	**.0000**	**.0000**	**.0000**	**.0000**	**.0000**	**.0000**	**.0000**	**.0000**	**.0000**	**.0000**	**.0000**	**.0000**	**.0000**	**.0000**
yea	**.0000**	**.0000**	**.0000**	**.0000**	**.0000**	**.0000**	**.0000**	**.0000**	**.0000**	**.0000**	**.0000**	**.0000**	**.0000**	**.0000**	**.0000**	**.0000**	**.0000**	**.0000**
AVG.	.4339	**.5690**	.5991	**.6691**	.1675	**.1887**	**.6877**	.6844	.6117	**.6384**	.4049	**.4529**	.5896	**.6775**	**.7538**	.7506	.5164	**.6013**

Table 5: Wilcoxon's test: original (+) vs. transformed (-) datasets.

Accuracy rate per class (Acc_{Class})

Algorithm	$\mathbf{R^+}$	$\mathbf{R^-}$	**Hypothesis**	**p-value**
CHI	136.0	644.0	Rejected for (-) with ($\alpha = 0.05$)	0.0004
FARC-HD	184.0	557.0	Rejected for (-) with ($\alpha = 0.05$)	0.0068
FHGBML	421.0	399.0	Not rejected	0.8825
FURIA	119.5	111.5	Not rejected	0.8894
FuzzyID3	225.0	441.0	Rejected for (-) with ($\alpha = 0.1$)	0.0897
GFS-GP	285.0	576.0	Rejected for (-) with ($\alpha = 0.1$)	0.0594
IVTURS	199.0	581.0	Rejected for (-) with ($\alpha = 0.05$)	0.0077
PDFC	310.0	393.0	Not rejected	0.5313
SLAVE	132.0	688.0	Rejected for (-) with ($\alpha = 0.05$)	0.0002

Geometric mean (GM)

Algorithm	$\mathbf{R^+}$	$\mathbf{R^-}$	**Hypothesis**	**p-value**
CHI	114.0	481.0	Rejected for (-) with ($\alpha = 0.05$)	0.0017
FARC-HD	141.0	489.0	Rejected for (-) with ($\alpha = 0.05$)	0.0044
FHGBML	61.0	110.0	Not rejected	0.2860
FURIA	110.0	61.0	Not rejected	0.2860
FuzzyID3	176.0	289.0	Not rejected	0.2452
GFS-GP	91.0	405.0	Rejected for (-) with ($\alpha = 0.05$)	0.0021
IVTURS	116.0	514.0	Rejected for (-) with ($\alpha = 0.05$)	0.0011
PDFC	307.0	288.0	Not rejected	0.8710
SLAVE	88.0	507.0	Rejected for (-) with ($\alpha = 0.05$)	0.0003

Table 6: Average rule length (RL) and number of rules (#Rules) built on the original (ORG) and transformed (TRA) datasets.

	#Rules		**RL**	
Method	**ORG**	**TRA**	**ORG**	**TRA**
CHI	369.93	1090.69	16.73	16.73
FARC-HD	35.60	51.63	2.25	2.22
FHGBML	7.01	7.83	16.73	16.73
FURIA	42.45	42.45	2.87	3.21
FuzzyID3	14015.11	1654.72	8.56	5.78
GFS-GP	5.00	5.00	180.17	177.84
IVTURS	38.68	58.11	2.27	2.24
PDFC	1687.99	1952.20	16.76	16.76
SLAVE	45.91	73.56	29.79	28.21

Table 7: Runtime(ms) of FUZZ-EQ on an Intel Xeon E5-2620v2 processor.

Dataset	Time (ms)	Dataset	Time (ms)	Dataset	Time (ms)
app	68.5	mag	400.0	spe	236.3
bal	77.3	mon	75.1	tae	112.8
ban	201.8	new	124.1	tex	713.6
bup	136.9	opt	1,330.3	thy	571.8
con	176.0	pag	336.7	tit	146.2
eco	129.2	pen	466.5	two	235.5
gla	128.5	pho	227.0	veh	199.1
hab	119.4	pim	162.5	vow	107.5
hay	65.2	rin	228.8	wdb	199.2
hea	132.8	sah	146.1	win	122.8
ion	187.9	sat	967.7	wre	221.9
iri	114.4	seg	246.9	wwh	320.5
led	74.9	son	172.9	yea	208.5
let	792.1	spa	1,287.4		

5.2. Scalability

In order to test the scalability of our proposal in Big Data, we implemented a Spark[4] version of FUZZ-EQ callled FUZZ-EQ-Spark. The parallelization of computation primarily relies on the *sortBy*[5] function provided by the *Resilient Distributed Dataset* (RDD) [35] API of Spark. This function sorts the values of a variable in a distributed fashion and is used by FUZZ-EQ-Spark to compute the quantiles, which is the task that determines the time complexity of FUZZ-EQ. Each variable is preprocessed sequentially, and thus we measured the scalability of the algorithm with respect to the number of examples rather than features.

We considered 6 Big Data classification problems available at UCI [18] and OpenML[6] repositories (Table 8) and applied the 5-fold cross-validation scheme described in the previous section. In this case, we ran the only open-source fuzzy classifier for Big Data that does not apply any optimization process to adjust the fuzzy partitions, i.e., CHI-BD [9]. Other Big Data Chi-based methods such as Chi-FRBCS-BigData [19] were excluded from the study because CHI-BD reported better results. The scalability was measured in terms of three well-known metrics to evaluate distributed systems, i.e., speedup, sizeup, and scaleup [17].

- *Speedup*: the data size is kept constant and the number of cores is increased. An ideal distributed algorithm should feature linear speedup, that is, a system with m cores must provide a speedup of m. However, in practice a linear speedup is difficult to obtain due to communication and synchronization overhead.

$$Speedup(m) = \frac{\text{runtime on 1 core}}{\text{runtime on } m \text{ cores}} \tag{10}$$

[4] `https://spark.apache.org`

[5] `https://spark.apache.org/docs/2.0.2/api/scala/#org.apache.spark.rdd.RDD`

[6] `https://www.openml.org/search?type=data`

- *Sizeup*: the number of cores is kept constant and the data size is increased. Sizeup measures how much longer it will take to process an m-times larger dataset. A linear increase in execution time represents the ideal case.

$$Sizeup(data, m) = \frac{\text{runtime for processing } m \cdot data}{\text{runtime for processing } data} \tag{11}$$

- *Scaleup*: the ability of a system to run an m-times greater job with m-times larger system is measured, whose ideal value should be 1 (runtime of the baseline system).

$$Scaleup(data, m) = \frac{\text{runtime for processing } data \text{ on 1 core}}{\text{runtime for processing } m \cdot data \text{ on } m \text{ cores}} \tag{12}$$

We ran the algorithms in a Hadoop[7] cluster composed of 6 slave nodes and a master node connected via 1Gb/s Ethernet LAN network. Half of the slave nodes have 2 Intel Xeon E5-2620 v3 processors at 2.4 GHz (3.2 GHz with Turbo Boost) with 12 virtual cores in each one (where 6 of them are physical). The other half are equipped with 2 Intel Xeon E5-2620 v2 processors at 2.1 GHz with the same number of cores as the previous ones. The master node is composed of an Intel Xeon E5-2609 processor with 4 physical cores at 2.4 GHz. All slave nodes are equipped with 64 GB of RAM memory, while the master works with 32 GB of RAM memory. With respect to the storage specifications, all nodes use Hard Disk Drives featuring a read/write performance of 128 MB/s.

Table 9 shows the runtime of FUZZ-EQ-Spark on several reduced versions of HIGGS with different number of single-core executors. More specifically, we took 4 executors and 10% of HIGGS as the baseline case ($m = 1$) and we gradually doubled both the number of executors and the data size (maintaining the original class distribution), until 32 executors and 80% of HIGGS. This way, for each number of executors (4, 8, 16, 32) we run the algorithm using 10%, 20%, 40%, and 80% of data. These runtimes were used to compute the speedup, sizeup, and scaleup (Fig. 6).

According to the results, FUZZ-EQ-Spark provides near-linear speedup and sizeup, leading to scaleup values closed to 1 (ideal case). Regarding the benefits of using FUZZ-EQ in Big Data classification problems, Table 10 shows that CHI-BD significantly improved its discrimination power on COVERTYPE and SUSY and got slightly worse results on BNG Australian. In the case of HIGGS, the fine tuning of fuzzy partitions led to a rule number explosion due to the inherent learning nature of CHI-BD, which is explained in detailed in [9]. These results suggest that FUZZ-EQ will probably be more beneficial in those cases were the learning algorithm (e.g. CHI-BD) exhibits low classification performance.

6. Concluding remarks

In this paper we have proposed a preprocessing algorithm (called FUZZ-EQ) for helping fuzzy partitioning methods (FPMs) adjust the fuzzy sets to the distribution of training data and boost the classification

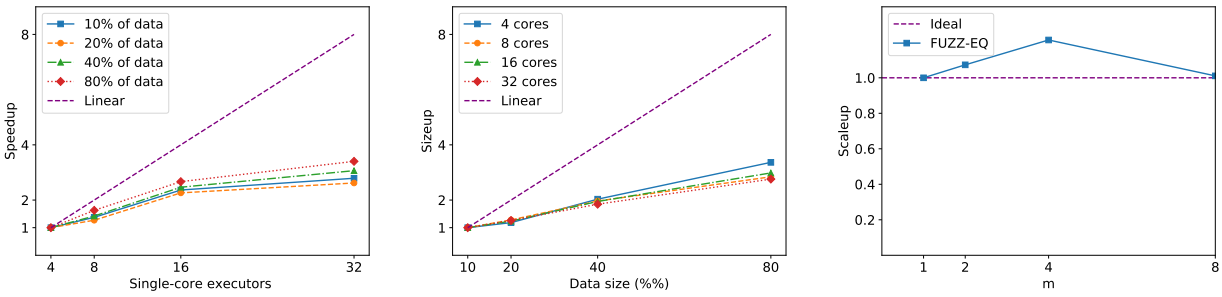
[7] `http://hadoop.apache.org`

Table 8: Description of the *Big Data* datasets.

Dataset	**#Examples**	**#Features** (R/I/C)/T	**#Classes**
BNG Australian	1,000,000	(8/6/0)/14	2
COVERTYPE	581,012	(10/0/44)/54	7
HIGGS	11,000,000	(28/0/0)/28	2
SUSY	5,000,000	(18/0/0)/18	2

Table 9: Runtime (s) of FUZZ-EQ-Spark on HIGGS.

Data size	**4 executors**	**8 executors**	**16 executors**	**32 executors**
10%	279	203	118	100
20%	330	260	146	126
40%	567	399	230	185
80%	939	577	352	276



(a) Speedup (b) Sizeup (c) Scaleup

Figure 6: Scalability of FUZZ-EQ-Spark on HIGGS.

Table 10: Classification performance of CHI-BD on original and transformed *Big Data* datasets.

Dataset	**Accuracy rate per class** (Acc_{Class})		**Geometric mean** (GM)	
	Original	**Transformed**	**Original**	**Transformed**
BNG	**85.02**	82.87	**.8483**	.8264
COVERTYPE	66.73	**79.44**	.6419	**.7847**
HIGGS*	58.48	-	.5847	-
SUSY	62.42	**76.65**	.5524	**.7634**

*The rule number explosion made CHI-BD run out of memory.

performance of fuzzy classifiers. FUZZ-EQ consists in transforming the original data distribution into a uniform distribution by means of the probability integral transform theorem. Specifically, the histogram of each variable is equalized so that the fuzzy sets contained in a hypothetical uniform partition would carry the same amount of experimental evidence. The proposed transformation improves the representation of training data by implicitly adapting the granularity of fuzzy sets depending on the probability mass.

The experimental results revealed that FUZZ-EQ boosted the classification performance of those classifiers that share the same linguistic terms among their rules. Even in those cases where membership functions were rule-specific (i.e., each rule uses its own linguistic terms), FUZZ-EQ did not negatively affect discrimination capability and was able to maintain classification performance. As for interpretability, we include a tool for recovering the interpretability of linguistic terms by mapping each data point in the transformed space back to the original space. The nonlinearities which might appear in membership functions when applying this tool helps FPMs create a more accurate representation of the data.

Beyond fuzzy approaches, future work might include a non-fuzzy version of FUZZ-EQ to help algorithms discretize continuous domains into a set of bins or intervals, which is a common practice in many popular machine learning methods such as decision trees. This would also allow us to empirically analyze how fuzzy classifiers perform with respect to non-fuzzy alternatives.

Acknowledgment

This work has been supported by the Spanish Ministry of Science and Technology under the project TIN2016-77356-P and the Public University of Navarre under the project PJUPNA13.

References

[1] Alcalá-Fdez, J., Alcalá, R., Herrera, F., 2011a. A Fuzzy Association Rule-Based Classification Model for High-Dimensional Problems With Genetic Rule Selection and Lateral Tuning. IEEE Transactions on Fuzzy Systems 19, 857–872.

[2] Alcalá-Fdez, J., Fernández, A., Luengo, J., Derrac, J., García, S., 2011b. KEEL Data-Mining Software Tool: Data Set Repository, Integration of Algorithms and Experimental Analysis Framework. Multiple-Valued Logic and Soft Computing 17, 255–287.

[3] Angus, J.E., 1994. The Probability Integral Transform and Related Results. SIAM Review 36, 652–654. arXiv:https://doi.org/10.1137/1036146.

[4] Barandela, R., Sánchez, J., García, V., Rangel, E., 2003. Strategies for learning in class imbalance problems. Pattern Recognition 36, 849–851.

[5] Chatterjee, K., De, A., Chan, F.T., 2019. Real time traffic delay optimization using shadowed type-2 fuzzy rule base. Applied Soft Computing 74, 226 – 241.

[6] Chen, Y., Wang, J., 2003. Support Vector Learning for Fuzzy Rule-Based Classification Systems. IEEE Transactions on Fuzzy Systems 11, 716–728.

[7] Chi, Z., Yan, H., Pham, T., 1996. Fuzzy algorithms with applications to image processing and pattern recognition. World Scientific .

[8] Cohen, W.W., 1995. Fast effective rule induction.

[9] Elkano, M., Galar, M., Sanz, J., Bustince, H., 2017. CHI-BD: A Fuzzy Rule-Based Classification System for Big Data classification problems. Fuzzy Sets and Systems In Press.

[10] Fayyad, U., Irani, K., 1993. Multi-Interval Discretization of Continuous-Valued Attributes for Classification Learning, in: 13th International Joint Conference on Uncertainly in Artificial Intelligence(IJCAI93), pp. 1022–1029.

[11] González, A., Perez, R., 2001. Selection of relevant features in a fuzzy genetic learning algorithm. IEEE Transactions on Systems and Man and and Cybernetics and Part B: Cybernetics 31, 417–425.

[12] Hühn, J., Hüllermeier, E., 2009. FURIA: an algorithm for unordered fuzzy rule induction. Data Mining and Knowledge Discovery 19, 293–319.

[13] Ishibuchi, H., Nakashima, T., Nii, M., 2004. Classification and modeling with linguistic information granules: Advanced approaches to linguistic Data Mining. Springer-Verlag.

[14] Ishibuchi, H., Yamamoto, T., 2005. Rule Weight Specification in Fuzzy Rule-Based Classification Systems. IEEE Transactions on Fuzzy Systems 13, 428–435.

[15] Ishibuchi, H., Yamamoto, T., Nakashima, T., 2005. Hybridization of Fuzzy GBML Approaches for Pattern Classification Problems. IEEE Transactions on Systems and Man and Cybernetics - Part B: Cybernetics 35, 359–365.

[16] Janikow, C., 1998. Fuzzy decision trees: Issues and methods. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 28, 1–14.

[17] Jogalekar, P., Woodside, M., 2000. Evaluating the Scalability of Distributed Systems. IEEE Transactions on Parallel and Distributed Systems 11, 589–603.

[18] Lichman, M., 2013. UCI Machine Learning Repository. URL: http://archive.ics.uci.edu/ml.

[19] López, V., del Río, S., Benítez, M., Herrera, F., 2015. Cost-sensitive linguistic fuzzy rule based classification systems under the MapReduce framework for imbalanced big data. Fuzzy Sets and Systems 258, 5–38.

[20] Muelas, D., Gordo, M., García-Dorado, J.L., de Vergara, J.E.L., 2015. Dictyogram: A statistical approach for the definition and visualization of network flow categories, in: 2015 11th International Conference on Network and Service Management (CNSM), pp. 219–227.

[21] Murata, T., Ishibuchi, H., Gen, M., 1998. Adjusting fuzzy partitions by genetic algorithms and histograms for pattern

classification problems, in: 1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98TH8360), pp. 9–14.

[22] Nair, N.U., Sankaran, P.G., Balakrishnan, N., 2013. Quantile-Based Reliability Analysis. Springer New York, New York, NY. chapter Quantile Functions. pp. 1–28.

[23] Pedrycz, W., 2001. Fuzzy equalization in the construction of fuzzy sets. Fuzzy Sets and Systems 119, 329 – 335.

[24] Quesenberry, C.P., 2004. Probability Integral Transformations. John Wiley & Sons, Inc.

[25] Quinlan, J.R., 1993. C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers Inc.

[26] Ramani, B.L., Poosapati, P., 2017. Adaptive Lion Fuzzy System to Generate the Classification Rules using Membership Functions based on Uniform Distribution. International Journal of Applied Engineering Research 12, 14421–14433.

[27] Sánchez, L., Couso, I., Corrales, J., 2001. Combining GP Operators With SA Search To Evolve Fuzzy Rule Based Classifiers. Information Sciences 136, 175–192.

[28] Sanz, J., Fernández, A., Bustince, H., Herrera, F., 2013. IVTURS: A Linguistic Fuzzy Rule-Based Classification System Based On a New Interval-Valued Fuzzy Reasoning Method With Tuning and Rule Selection. IEEE Transactions on Fuzzy Systems 21, 399–411.

[29] Umanol, M., Okamoto, H., Hatono, I., Tamura, H., Kawachi, F., Umedzu, S., Kinoshita, J., 1994. Fuzzy decision trees by fuzzy ID3 algorithm and its application to diagnosis systems, in: Proceedings of 1994 IEEE 3rd International Fuzzy Systems Conference, pp. 2113–2118.

[30] Wilcoxon, F., 1945. Individual comparisons by ranking methods. Biometrics 1, 80–83.

[31] Yadav, D.K., Yadav, H.B., 2015a. Developing Membership Functions and Fuzzy Rules from Numerical Data for Decision Making, in: 2015 Conference of the International Fuzzy Systems Association and the European Society for Fuzzy Logic and Technology (IFSA-EUSFLAT-15), Atlantis Press. doi:https://doi.org/10.2991/ifsa-eusflat-15.2015.79.

[32] Yadav, H., Yadav, D., 2015b. A histogram technique to construct membership function of numeric data. International Journal of Applied Engineering Research 10, 2292–2297.

[33] Yuan, Y., Shaw, M., 1995. Induction of fuzzy decision trees. Fuzzy Sets and Systems 69, 125–139.

[34] Zadeh, L., 1968. Probability measures of Fuzzy events. Journal of Mathematical Analysis and Applications 23, 421 – 427.

[35] Zaharia, M., Chowdhury, M., Das, T., Dave, A., Ma, J., McCauley, M., Franklin, M.J., Shenker, S., Stoica, I., 2012. Resilient Distributed Datasets: A Fault-tolerant Abstraction for In-memory Cluster Computing, in: Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation, pp. 2–2.

[36] Zarandi, M.F., Soltanzadeh, S., Mohammadi, A., Castillo, O., 2019. Designing a general type-2 fuzzy expert system for diagnosis of depression. Applied Soft Computing 80, 329 – 341.

[37] Zeinalkhani, M., Eftekhari, M., 2014. Fuzzy partitioning of continuous attributes through discretization methods to construct fuzzy decision tree classifiers. Information Sciences 278, 715 – 735.