Distributed classification based on distances between probability distributions in feature space

Pablo Montero-Manso[a,*], Laura Morán-Fernández[b], Verónica Bolón-Canedo[b], José A. Vilar[a], Amparo Alonso-Betanzos[b]

[a]*Research Group on Modeling, Optimization and Statistical Inference (MODES), Department of Mathematics, Computer Science Faculty, University of A Coruña*
[b]*Laboratory for Research and Development in Artificial Intelligence (LIDIA), Department of Computer Science, Computer Science Faculty, University of A Coruña*

Abstract

We consider a distributed framework where training and test samples drawn from the same distribution are available, with the training instances spread across disjoint nodes. In this setting, a novel learning algorithm based on combining with different weights the outputs of classifiers trained at each node is proposed. The weights depend on the distributional distance between each node and the test set in the feature space. Two different weighting approaches are introduced, which are referred to as per-Node Weighting (pNW) and per-Instance Weighting (pIW). While pNW assigns the same weight to all test instances at each node, pIW allows distinct weights for test instances differently represented at the node. By construction, our approach is particularly useful to deal with unbalanced nodes. Our methods require no communication between nodes, allowing for data privacy, independence of the kind of trained classifier at each node and maximum training speedup. In fact, our methods do not require retraining of the node's classifiers if available. Although a range of different combination rules are considered to ensemble the single classifiers, theoretical support for the optimality of using the sum rule is provided. Our experiments illustrate all of these properties and show that pIW produces the highest classification accuracies compared

*Corresponding author

Email addresses: p.montero.manso@udc.es (Pablo Montero-Manso), laura.moranf@udc.es (Laura Morán-Fernández), vbolon@udc.es (Verónica Bolón-Canedo), jose.vilarf@udc.es (José A. Vilar), ciamparo@udc.es (Amparo Alonso-Betanzos)

with pNW and the standard unweighted approaches.

Keywords: distributed classification, distributional distances, classifiers combination, imbalanced data set, classification accuracy

---

1. Introduction

Data is growing at an unprecedented pace. With the variety, speed and volume of data flowing through networks and databases, it has become more and more difficult to find patterns that lead to meaningful conclusions. At the same time, organizations need to find ways to obtain some value from all of this data. Unlocking the most value from large, varied sets of information requires new approaches based on machine learning. However, traditional machine learning techniques and, more specifically, data mining algorithms, have been designed to run in a centralized computing environment, where all data could fit in a single machine. But nowadays, in the current scenario, where data size increases beyond capacity, these algorithms do not scale well —memory demands and impracticable runtimes—, damaging performance and efficiency. Thus, distributed learning has become essential.

The motivation for distributed learning is at least twofold. The most obvious reason is the volume of data available nowadays. Data generation, which has been estimated at 2.5 exabytes of data per day [3], comes from everywhere: genomics, astronomy, CERN experiments, transaction records, posts to social media sites (Twitter generates 500 million tweets/day, each about 3 kilobytes including metadata) or digital pictures and videos (YouTube currently has 300 hours of video being uploaded every minute). Second, data is often shared across geographical and organizational boundaries, and it is not economic or legal to gather it in a single location. For example, several datasets concerning business information might be owned by separate organizations that have competitive reasons for keeping the data private. In addition, this data may be physically dispersed over many different geographic locations. However, business organizations may be interested in enhancing their own models by exchanging useful information about the data.

The machine learning community has been essentially focused on the design of distributed or parallel algorithms to deal with massive datasets [17]. Different from the traditional centralized algorithms where a single learner has access to the full dataset, distributed learning algorithms have their foundations in ensemble learning [9]. Ensemble learning consists of a hierarchy

of multiple local learners operating on subsets of the full dataset, and one or more ensemble learners combining the outputs of all the local learners. Thus, the ensemble approach is almost directly applicable to a distributed scenario since a classifier can be trained at each site, using the subset of data stored in it, and then the classifiers can be eventually combined using ensemble strategies. To combine the predictions of a set of classifiers, one of the simplest ways consists of using decision rules [12]. These decision/fixed rules are defined as functions that receive as inputs the outputs of the set of learned classifiers and combine them to produce a unique output. Chan and Stolfo [5] proposed several meta-learning strategies for integrating independently learned classifiers by the same learner in a parallel and distributed computing environment. Breiman [4] presented a procedure to build ensembles of classifiers from small subsets of data, growing a predictor on each subset and then pasting these predictors together. Lazarevic *e*t al. [13] developed a general framework for distributed boosting to integrate efficiently specialized classifiers learned over very large and distributed homogeneous datasets that cannot be merged at a single location using the Mahalanobis distance. Tsoumakas *e*t al. [23] presented a framework for distributed stacking of multiple classifiers. Their method was based on local learning and model stacking using the average probability distribution of the local classifiers' output according to the class as input to the second level classifier. Similar to distributed learning, another common approach for scaling up learning algorithms is parallel machine learning [25], which includes GPU architecture and map reduce techniques.

Data can be distributed either horizontally or vertically. In horizontal partitioning, the dataset is divided into several nodes that have the same features as the original dataset, each containing a subset of the original samples. In vertical partitioning, the original dataset is divided into several nodes that have the same number of instances as the original dataset, each containing a subset of the original set of features. This work is focused on horizontal partitioning, since it constitutes the most suitable and natural approach for most applications. In addition to the common learning scenario assumptions, we assume the availability of a test set large enough to obtain distributional information. In this distributed scenario, class probabilities can be shown to be a weighted average of the individual class probabilities within each node. These weights depend on the marginal probabilities of the instance over each node and over the entire data set. This result motivates the study of the use of distribution distances for improving classification performance.

In this work, two different approaches to approximate these weights are proposed. The first one is based on estimating the distance between feature distributions between each node and the test set, while the second one controls the contribution of each instance of the test set in order to minimize these distributional distances. The resulting learning models exhibit interesting properties including that they work with any local classifier and do not require retraining the classifiers or sharing information between individual nodes. The experimental results on several real and synthetic data sets report benefits in terms of classification accuracy, particularly when the second approach is considered. Besides, we assess a common problem in many real world problems, the "class imbalance change" [10] in classification. In a non-distributed framework, this problem appears when the class balance changes between training and test data sets, due to sample selection bias or non-stationarity of the environment [18]. In our case, unbalancedness happens when the feature distributions differ between nodes. Distributed real-world data sets are usually not symmetric, i.e. the distributions of data for different locations may not be the same. Imagine a group of epidemiologists studying the spread of hepatitis-C in Europe. They are interested in detecting any underlying relation of the emergence of hepatitis-C in Europe with the weather and social patterns. They have access to some large hepatitis-C country-specific databases and an environmental database at EEA (European Environment Agency). These patterns could change considerably from one country to another (e.g. from Denmark to Italy). Besides, analyzing the data from these distributed datasets using a traditional data mining algorithm will require combining the databases at a single location, which is quite impractical, or perhaps not possible due to memory reasons or to privacy issues. It will be shown that our model works particularly well in these scenarios.

The remainder of this paper is organized as follows. Our distributed learning model is introduced and analyzed in detail in Section 2. Specifically, the considered framework and the rationale of the proposal are discussed in Sections 2.1 and 2.2, respectively. A scheme of the methodology is outlined in Section 2.3, and the key issues of the procedure involving the choice of the distributional distance, the weighting criteria and the way of combining the single classifier outputs are discussed in Sections 2.4, 2.5 and 2.6, respectively. The results from an experimental study involving simulated and real data sets are presented in Section 3, which is structured as follows. The experimental design is described in Section 3.1. Different classification algo-

rithms, combination strategies, and partition sizes have been considered in both balanced and unbalanced scenarios. Some results from a pair of specific experiments conducted to motivate our approach are shown in Section 3.2, and the bulk of experimental results are analyzed in Section 3.3. Additional results from our experiments are also provided in Appendix A. Lastly, computational complexity is addressed in Section 4 and some concluding remarks and proposals for future research are given in Section 5.

2. A novel distributed learning model

This section is devoted to formally establish the distributed classification problem and describes in detail the learning model we propose.

2.1. Background

Consider a population Ξ characterized by pairs of measurements $(x, C) \in \mathcal{X} \times \mathcal{C}$, where $\mathcal{X}$ denotes a domain of d-dimensional feature vectors and $\mathcal{C} = \{C_1, \ldots, C_m\}$ is a set of m class labels. Denote by $P(x, C)$ the joint probability function over $\mathcal{X} \times \mathcal{C}$. In a standard classification context, a classifier based on a training sample of n labeled objects $\mathcal{Z} = \{(x_1, C_1), \ldots, (x_n, C_n)\}$ is used to predict the class of unlabeled features. In a horizontally distributed framework, the population Ξ spreads across p disjoint nodes, let us say $\mathcal{P} \equiv \{\mathcal{N}_1, \ldots, \mathcal{N}_P\}$, and the training data set brings together instances from the different nodes, i.e. $\mathcal{Z} = \cup_{i=1}^{p} \mathcal{Z}_i$, with $\mathcal{Z}_i$ formed by n_i instances belonging to $\mathcal{N}_i$, with $\sum_{i=1}^{p} n_i = n$.

In this paper we focus on a distributed environment where a set $\mathcal{T}$ of t unlabeled instances is available and our target is to estimate their labels. The availability of a whole set of unlabeled instances $\mathcal{T}$ enables us to gain knowledge about the underlying distribution of X and to assess how well this distribution is represented at each node. As we will see later, our learning model relies on these distributional distances so that the availability of $\mathcal{T}$ is a basic requirement.

In addition, our learning model is constructed under the standard assumption that the training set $\mathcal{Z}$ and the test set $\mathcal{T}$ are independent and identically distributed samples drawn from the population in study, and therefore following the probability model given by $P(x, C)$. Note that this stationarity assumption is the default assumption in many learning scenarios. No distributional or other assumptions are required on the data or how they are distributed across nodes. In fact, data within each node could follow different

distributions since no constraints on the fragmentation scheme are imposed. In particular, our framework encompasses scenarios with unbalanced nodes [19] or with data-driven partitions on the basis of heuristic rules stated to obtain better classification rates [8].

We also impose the restriction that no communication between nodes is required. Intelligent interaction between nodes, e.g. taking advantage of the most informative data at each node, can improve the classification accuracy [7]. Nevertheless, exchanging information between nodes is frequently unfeasible in real problems dealing with distributed data for different reasons such as storage cost, communication cost or private and sensitive data, among others [24].

2.2. An overview of our approach

A common approach in distributed learning [2] consists of building classifiers trained at each node $\mathcal{N}_i$ using $\mathcal{Z}_i$, $i = 1, \ldots, p$, and then combining the classifier outputs by means of a proper ensemble learning strategy [9]. In our approach, we intend to take advantage of the availability of $\mathcal{T}$ to gain insight into the marginal probability distribution of the feature vectors, and using this knowledge to modulate the importance of each individual classifier in the combination rule. Specifically, we wish to estimate the posteriori probability that the j-th instance in $\mathcal{T}$, with observed feature vector x_j, belongs to the class C_k, for $k = 1, \ldots, m$ and $j = 1, \ldots, t$. Under the stationarity assumption and given that $\mathcal{P}$ is a partition of Ξ, we have

$$P\left(C_k \left| x_j \right.\right) P\left(x_j\right) = \sum_{i=1}^{p} P\left(C_k \left| x_j, \mathcal{N}_i \right.\right) P\left(x_j \left| \mathcal{N}_i \right.\right) P\left(\mathcal{N}_i\right), \tag{1}$$

where $P\left(x_j\right)$ denotes the marginal density of x_j, $P\left(x_j \left| \mathcal{N}_i \right.\right)$ the density of x_j conditional on the i-th node, and $P\left(\mathcal{N}_i\right)$ the prior probability of an instance is allocated to $\mathcal{N}_i$.

Let ω_{ji} be the ratio defined by $\omega_{ji} = \dfrac{P\left(x_j \left| \mathcal{N}_i \right.\right)}{P\left(x_j\right)}$, for $i = 1, \ldots, p$. Then, from (1) follows that

$$P\left(C_k \left| x_j \right.\right) = \sum_{i=1}^{p} P\left(C_k \left| x_j, \mathcal{N}_i \right.\right) \omega_{ji} P\left(\mathcal{N}_i\right). \tag{2}$$

Equation (2) establishes that the posteriori probability of the class C_k given an observed feature vector x_j is a weighted average of the posteriori probabilities within each node, with weights depending on the node size and the

ratios ω_{ji}. By definition, ω_{ji} measures how well represented is the observed feature vector x_j in the i-th node. Whether the partition $\mathcal{P}$ has been set evenly and uniformly at random, the feature vectors within each node follow similar distributions and ω_{ji} will take values close to one for all j and i. Otherwise, markedly unbalanced nodes will produce very different ω_{ji}.

The value of $P(C_k|x_j)$ can be directly estimated from (2) as long as the remaining involved probabilities are previously approximated. The posteriori probability within each node, $P(C_k|x_j, \mathcal{N}_i)$, is estimated using the classifier trained at $\mathcal{N}_i$ and whose output consists of a vector of m belief values. The proportion of training data belonging to the i-th node can be taken as an estimate of $P(\mathcal{N}_i)$. For the sake of simplicity and computational efficiency, we will assume nodes of equal size so that the weight of a single prediction is not affected by the nodes' sizes. Lastly, the behavior in probability of the feature vectors over Ξ and over each node $\mathcal{N}_i$ can be modeled with nonparametric kernel densities based on the features forming $\mathcal{T}$ and $\mathcal{Z}_i$, respectively. Nevertheless, this involves several difficulties. First, we could face the "curse of dimensionality" problem since the dimension of the feature space may be arbitrarily large. Moreover, we look for a learning model able to manage different types of features, including mixtures of discrete and continuous variables. But even assuming an affordable dimension and continuous features, $(p+1)$ kernel densities should be obtained, which substantially increases the likelihood of estimation errors. In particular, small errors estimating $P(x_j|\mathcal{N}_i)$ or $P(x_j)$ might produce arbitrarily large or small coefficients ω_{ji}, thus leading to overweight or underweight the predictions in specific nodes.

To overcome these drawbacks, the computation of the $(p+1)$ kernel densities is circumvented by directly estimating the coefficients ω_{ij}. Two different approaches are proposed. In both cases, the aim is to measure the dissimilarity d_i between the feature distributions on the i-th node and the global population, hereafter denoted by $F_{\mathcal{N}_i}$ and $F_{\mathcal{X}}$, respectively. A suitable distance between high-dimensional distributions is considered, and then specific values for d_i, $i = 1, \ldots, p$, are obtained using the empirical distributions based on $\mathcal{Z}_i$ and $\mathcal{T}$. The first proposal consists in taking $\omega_{ji} = K \cdot d_i^{-1}$, for all j and i, and K being a constant. This way, all the instances in $\mathcal{T}$ receive the same weight at each node, which decreases with the distance between $F_{\mathcal{N}_i}$ and $F_{\mathcal{X}}$. The second proposed approach is not simply based on the global distance between empirical distributions. The weights ω_{ji} are determined in order to maximize the matching between $F_{\mathcal{N}_i}(x_j)$ and $F_{\mathcal{X}}(x_j)$, for all $x_j \in \mathcal{T}$. Unlike the prior approach, the test instances receive different weights ω_{ji} at the same

node. The effective computation of the weights is formalized throughout an optimization problem. A detailed description of the two proposed weighting criteria is provided in Section 2.5.

2.3. Outline of the methodology

We propose a distributed learning methodology consisting of the following four stages.

Step 1 Assess the distance between the probability distributions of X on the i-th training node $\mathcal{N}_i$ and the global population Ξ using a suitable statistic to measure dissimilarity between high-dimensional distributions. Denote by d_i the normalized distance obtained for the i-th training node, $i = 1, \ldots, p$.

Step 2 Based on a pre-selected classifier, obtain for each feature vector x_j of the test sample the classifier outputs $\mathbf{Y}_j = \{\mathbf{y}_{j1}, \ldots, \mathbf{y}_{jp}\}$, where $\mathbf{y}_{ji}$ denotes the response generated by the classifier trained at the node $\mathcal{N}_i$, for $i = 1, \ldots, p$ and $j = 1, \ldots, t$.

It is assumed that each classifier output consists of a vector of m membership or belief values, i.e. $\mathbf{y}_{ji} = (y_{ji1}, \ldots, y_{jim})$, where y_{jik} can be interpreted as the amount of confidence or evidence in the assignment of the feature x_j to the k-th class, C_k, for $k = 1, \ldots, m$.

Step 3 Obtain weighted versions of the belief values $\mathbf{Y}_j^{\omega} = \left\{\mathbf{y}_{j1}^{\omega}, \ldots, \mathbf{y}_{jp}^{\omega}\right\}$, with $\mathbf{y}_{ji}^{\omega} = \omega_{ji}\mathbf{y}_{ji}$, where the weights ω_{ji} take into consideration the distributional distances d_i. Two different criteria are proposed to determine how the weights are constructed.

Step 4 Generate a unique decision for classifying the j-th instance in $\mathcal{T}$, with observed feature x_j, by combining the corresponding weighted belief sets $\{\omega_{j1}\mathbf{y}_{j1}, \ldots, \omega_{jp}\mathbf{y}_{jp})$. Following Kittler *et al.* [12], several fixed rules (*decision rules*) involving functions of the elements of $\mathbf{Y}_j^{\omega}$ are considered to produce the required unique output.

The key points of the proposed methodology involve: (i) the choice of the distributional distance d_i, (ii) the weighting criteria on the belief values regarding the distances d_i, and (iii) the selection of a decision rule. Each of these issues is properly discussed below.

2.4. Measuring dissimilarity between high-dimensional distributions

To assess the distance between the probability distributions of X over an arbitrary node $\mathcal{N}_i$ and the population Ξ, we propose to use the so-called *energy statistic* [21, 22]. Consider two independent samples $\mathcal{X}$ and $\mathcal{X}'$ generated from multivariate distributions $F_{\mathcal{X}}$ and $F_{\mathcal{X}'}$, respectively. The energy distance between $\mathcal{X}$ and $\mathcal{X}'$ is defined by

$$E\left(\mathcal{X}, \mathcal{X}'\right) = 2d_{\mathcal{X},\mathcal{X}'} - d_{\mathcal{X},\mathcal{X}} - d_{\mathcal{X}',\mathcal{X}'}, \tag{3}$$

with

$$d_{\mathcal{A},\mathcal{B}} = \frac{1}{rs} \sum_{u=1}^{r} \sum_{v=1}^{s} \left\| a_u - b_v \right\|,$$

where $\|\cdot\|$ denotes the Euclidean norm and $\mathcal{A} \equiv (a_1, \ldots, a_r)$ and $\mathcal{B} \equiv (b_1, \ldots, b_s)$ denote arbitrary data sets.

Under mild regularity conditions on the generating patterns, Székely and Rizzo [21] established the consistency of the statistic (3) to check the equality of the generating distributions $F_{\mathcal{X}}$ and $F_{\mathcal{X}'}$. Hence $E\left(\mathcal{X}, \mathcal{X}'\right)$ can be seen as a measure of the distance between $F_{\mathcal{X}}$ and $F_{\mathcal{X}'}$ in such a way that the larger value of the statistic, the more distant are the distributions. By construction, $E\left(\mathcal{X}, \mathcal{X}'\right)$ is based on comparing averages of interpoint distances evaluated within and between samples, which means to move the multidimensional problem to dimension one. Thus, the energy distance is particularly attractive to be applied in arbitrarily high dimension. It is also worthy to remark that different types of interpoint distances could be used to construct $E(\cdot, \cdot)$, thus providing versatility to deal with features taking nominal, categorical, continuous and also mixed values. Also, the good analytical properties of the energy distance will allow us to formalize in the next section a suitable optimization problem designed to provide useful weights for the belief values. Supported by these nice properties, we decided to evaluate the distributional distance between each $\mathcal{N}_i$ and Ξ by means of the energy distance between the i-th training sample and the test sample, i.e. by $d_i = E\left(\mathcal{Z}_i, \mathcal{T}\right)$, for $i = 1, \ldots, p$.

2.5. Weighting the belief values generated by the single classifiers

From Step 2 of the proposed methodology, the outputs of the single classifiers $\mathbf{Y}_j = \{\mathbf{y}_{j1}, \ldots, \mathbf{y}_{jp}\}$ are available for each feature vector x_j of the test sample. As mentioned, we assume that $\mathbf{y}_{ji}$ is a vector of levels of belief in the

assignment of x_j to each of the classes. Working with belief levels enables us to analyze the performance of a range of efficient classifier combination rules [12] in Step 4 of the proposed methodology. Step 3 consists in correcting these belief values by introducing the distributional distances d_i. Two different criteria are proposed.

One approach consists in assigning weights in inverse proportion to the energy distance for the corresponding node, i.e. $\omega_{ji} = K \cdot d_i^{-1}$ for all j, where $K = \left(\sum_{i=1}^{p} d_i^{-1}\right)^{-1}$, is a normalizing constant used to make the sum of weights equal to one. This way, the belief values generated from each local classifier receive a common weight for all instances in the test set, resulting $\mathbf{y}_j^{\omega} = K\, d_i^{-1}\mathbf{y}_j$, for all $j = 1, \ldots, t$. Hereafter this weighting approach will be referred to **per-Node Weighting** and denoted by **pNW**.

In order to provide a finer grain approach where the belief degrees per instance in the test set receive different weights, an alternative weighting approach is proposed. The aim is to assign weights in order to minimize the energy distance between each training sample and the test set. The procedure can be understood as if, for each node $\mathcal{N}_i$, a weighted resampling scheme of the test set is carried out to overweight belief values associated to instances better represented at the node than in the test sample. Features x_j allocated in low probability zones in the test set but belonging to high probability zones in a specific node will receive high weights, and conversely instances with low probability in the test set but high probability in the node will be downweighted (see Figure 1). By assigning high weights to instances with low representation in the test set but well-represented at the node, we ensure an efficient use of the training samples. Notice that equation (2) in Section 2.2 leads to theoretical weights $\omega_{ji} = P\left(x_j \,|\mathcal{N}_i\right) / P\left(x_j\right)$, thus accounting for the rationale of this approach. Unlike the per-node belief approach, under this new weighting criterion each node produces a weight for each instance in the test set. For this reason, this weighting approach will be referred as **per-Instance Weighting** and denoted by **pIW**.

According to the definition of the energy distance in (3), the per-instance weights for the set of test instances at the i-th node, $\omega_i = (\omega_{1i}, \ldots, \omega_{ti})$, are obtained by minimizing the objective function $E\left(\omega_i\right)$ given by

$$E\left(\omega_i\right) = 2D_{\mathcal{Z}_i,\mathcal{T}}\omega_i^T - D_{\mathcal{Z}_i,\mathcal{Z}_i} - \omega_i D_{\mathcal{T},\mathcal{T}}\omega_i^T, \tag{4}$$

where $D_{\mathcal{A},\mathcal{B}}$ is the matrix whose (u, v)-element is $D_{\mathcal{A},\mathcal{B}}(u, v) = \|a_u - b_v\|$, for arbitrary data sets $\mathcal{A} \equiv (a_1, \ldots, a_r)$ and $\mathcal{B} \equiv (b_1, \ldots, b_s)$.

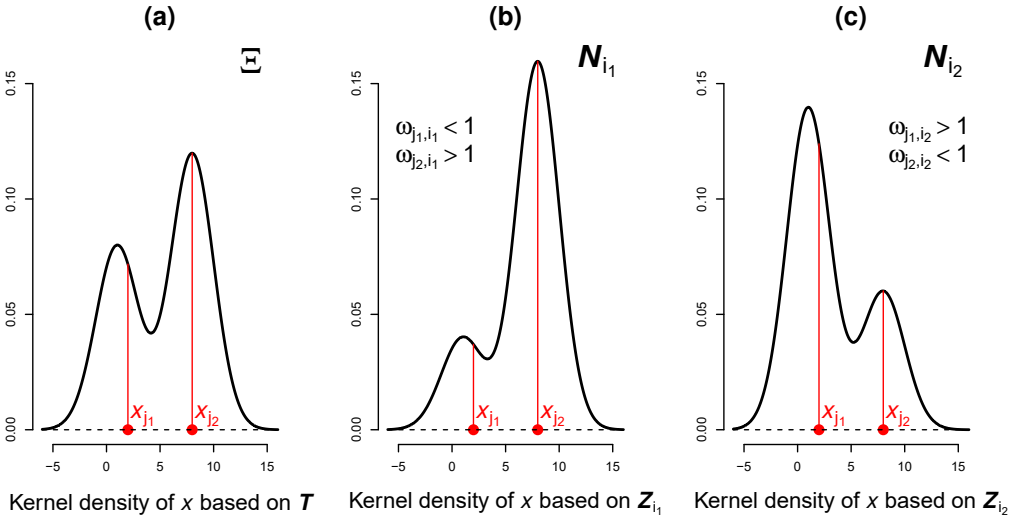

Figure 1: Graphical illustration of the per-Instance Weighting (pIW) criterion.

In practice, the minimization of $E(\omega_i)$ is posed by means of the optimization problem:

$$\begin{aligned} \underset{\omega_i}{\text{minimize}} \quad & \frac{1}{t} D_{\mathcal{N}_i,\mathcal{T}} \omega_i^T - \omega_i D_{\mathcal{T},\mathcal{T}} \omega_i^T \\ \text{subject to} \quad & \sum_{i=1}^{p} \omega_i = 1,\ \omega_i \succeq 0. \end{aligned}$$

2.6. Combining the belief values generated by the single classifiers

Last step in the proposed methodology consists in combining the weighted outputs of the single classifiers trained at each of the nodes, namely the vectors of belief degrees $\mathbf{y}_{ji}^{\omega} = \omega_{ji}\mathbf{y}_{ji} = (\omega_{ji}y_{ji1}, \ldots, \omega_{ji}y_{jim})$, whose k-th element $y_{jik}^{\omega} = \omega_{ji}y_{jik}$ provides an estimate of the posteriori probability $P(C_k\,|x_j, \mathcal{N}_i)$, for $k = 1, \ldots, m$. Having available continuous outputs in form of belief values allows us to consider different functions of these values, so-called *decision rules* [17], to get a unique output. Kittler *et* al. [12] argue that the decision rules provide a useful approach to circumvent the complex problem of inferring the posteriori probability function

$$P(x_j \text{ is assigned to the class } C_k\,|\mathbf{y}_{j1}, \ldots, \mathbf{y}_{jp}),$$

which would allow us to determine the most likely class using the Bayesian theory. Some alternative classifier combination approaches include tech-

niques such as Stacked Generalization, Meta-Learning, Knowledge Probing and Effective Stacking [17]. Nevertheless, these methods work training a new classifier based on single outputs produced by each node, which requires access to a common training set or sharing of private training information among nodes, thus limiting their applicability and violating the condition of no communication between nodes stated in Section 2.1. Supported by these arguments, we propose to use some of the most popular decision rules to generate the final assignment.

Following Kittler *et* al. [12], where a common theoretical framework for different decision rules is provided, we have considered in our experiments the set of rules presented below. In all cases, we assume that the belief values have been normalized so that $P\left(C_k\left|x_j, \mathcal{N}_i\right.\right) = y_{jik}^{\omega} / \sum_{l=1}^{m} y_{jil}^{\omega}$, for all j and i.

- *Product rule.* The instance with observed feature vector x_j is assigned to the class C_k if

$$\prod_{i=1}^{p} y_{jik}^{\omega} = \max_{1 \leq l \leq m} \prod_{i=1}^{p} y_{jil}^{\omega}.$$

  Note that, under this rule, a class with a zero or very small belief value from only one node will receive a zero or very small combined belief degree, even if the rest of nodes provide high belief degrees to the mentioned class. Hence, this rule will exhibit a bad performance if for example a class is not represented in a particular node.

- *Sum rule.* The instance with observed feature x_j is assigned to the class C_k if

$$\sum_{i=1}^{p} y_{jik}^{\omega} = \max_{1 \leq l \leq m} \sum_{i=1}^{p} y_{jil}^{\omega}.$$

  The theoretical support for the sum rule lies on assuming that the posteriori probabilities do not deviate greatly from the prior probabilities [12], that is $P\left(C_k\left|x_j, \mathcal{N}_i\right.\right) = P\left(C_k\right) + \varepsilon_{jk}$, with ε_{jk} taking very small values for all k and j. Kittler *et* al. [12] have shown that the sum rule is less sensitive to the estimate errors than the product rule.

- *Max rule.* The instance with observed feature x_j is assigned to the class C_k if

$$\max_{1 \leq i \leq p} y_{jik}^{\omega} = \max_{1 \leq l \leq m} \max_{1 \leq i \leq p} y_{jil}^{\omega}.$$

The class obtaining the highest belief degree over all the nodes is selected as combined output. It can be shown that this rule approximates the sum rule under the assumption of equal prior probabilities for the classes.

- *Min rule.* The instance with observed feature x_j is assigned to the class C_k if

$$\min_{1\leq i\leq p} y_{jik}^{\omega} = \max_{1\leq l\leq m} \min_{i=1}^{p} y_{jil}^{\omega}.$$

Assuming as before that classes are a priori equiprobable, the min rule approximates the product rule.

- *Majority vote rule.* The instance with observed feature x_j is assigned to the class C_k if

$$\sum_{i=1}^{p} \Delta_{jik} = \max_{1\leq l\leq m} \sum_{i=1}^{p} \Delta_{jil},$$

where $\Delta_{jil} = 1$ if $y_{jil}^{\omega} = \max_{1\leq u\leq m} y_{jiu}^{\omega}$ and $\Delta_{jil} = 0$ otherwise. Therefore, the combined output consists in selecting the class receiving the largest number of votes from the single classifiers. Under the equiprobability assumption for the prior probabilities, this rule matches the sum rule when the belief values are discretized by using the Δ_{jil} values.

2.7. Some remarks

Some remarks concerning the proposed methodology are highlighted below.

Remark 1. The estimated distributional distance d_i between a particular node $\mathcal{N}_i$ and Ξ could be small (large) even though a few test instances are bad (well) represented at $\mathcal{N}_i$. In any case, all the classifier outputs obtained at $\mathcal{N}_i$ will receive the same weight when the pNW criterion is used. This is an unsuitable consequence of taking weights based on the global distance d_i such as pNW does. On the contrary, the pIW criterion checks the point-to-point distribution matching, thus being sensitive to local deviations. Note that if a feature vector x_j is badly represented at a specific node, then it must be well represented at another node because $\mathcal{P}$ is a partition of the feature domain. In sum, the pIW criterion is expected to outperform the pNW one, and the improvement would be more substantial with unbalanced nodes. In our experimental evaluation in Section 3.3, both weighting criteria

are examined and compared with a standard approach without weighting the single belief values (an unweighted approach denoted by **UW**). A scheme of the three distributed approaches is shown in Figure 2.

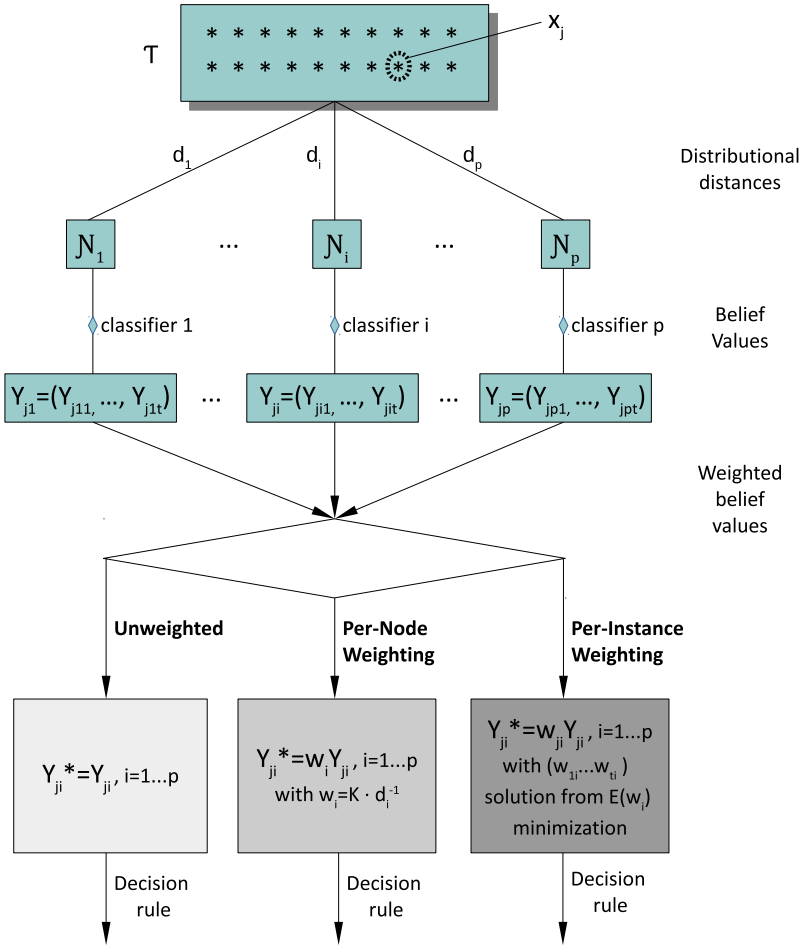

Figure 2: Distributed approaches schemes.

Remark 2. In a non-distributed classification context with different distributions for the training and test sets (*sample selection bias* problem), Huang *et* al. [11] proposed to use the unlabeled data to reweight the training data in such a way that the means of the training and test features in a reproducing kernel Hilbert space are close. Although in a different context, this is a similar idea to the pIW approach and it is worthy to emphasize the

main differences. In our work, the reweighting process is applied to the test data because the nodes cannot be retrained in our distributed scenario. On the other hand, a key assumption in [11] is that the conditional probability of $C|x$ is the same for the training and test populations so that the bias is only exhibited by the feature distributions. In our framework, stationarity is assumed and therefore the bias can only be present between nodes. Nevertheless, it is not necessary to require that the conditional probabilities of $C|x$ remain unchanged across the nodes, which would be a very restrictive constraint.

Remark 3. As already mentioned, Kittler *e*t al.[12] pointed out some nice properties of the sum rule to combine the single classifier outputs. Beyond these properties, equation (2) provides theoretical support to use this criterion since the posteriori probabilities are expressed as a weighted sum of the single classifier outputs within each node.

Remark 4. The proposed learning model is not restricted to the use of a particular classification model at each node. The unique requirement is that the classifier outcome consists of a vector of belief values or posteriori probabilities of the classes for a given feature vector. Thus, artificial neural network, logistic regression, support vector machines, Bayesian classifiers, and Random Forest could be used among others.

3. Experiments

An empirical study addressed to motivate and evaluate the performance of the proposed learning models has been carried out. A description of the experimental procedure and an overview and discussion of the main results are presented in this section.

3.1. Experimental setup

The main characteristics of the experiments are detailed below.

Classifiers. To study the interaction between the distributed learning models and the classifier type, five classification algorithms are considered, namely Random Forest (RF), a support vector machine with RBF Kernel (SVM), the Fisher's linear discriminant (LDA), the classifier based on multinomial logistic regression (Mult), and the XGBoost (eXtreme Gradient Boosting) algorithm (XGB), a fast implementation of the gradient boosting using decision trees. All of them were executed by using different R packages, `randomForest`

[14] for RF, `e1071` [16] for SVM, `xgboost`[6] for XGB, and `MASS` and `nnet` [26] for LDA and Mult, respectively. The default parameters are taken in all cases since our concern is not to determine the most efficient inputs but comparing the models under homogeneous conditions. All classifiers provide the options to output belief values in addition to classes

Data sets. Seven data sets are used to analyze the coupling between the proposed method and the underlying classification problem. Five databases (Spambase, KDD Cup 99, Connect-4, Covertype, and Higgs) contain real data and are available from the UCI Machine Learning Repository [15]. The other two databases (Simul-C2 and Simul-C8) consist of synthetic data generated from simulated classification scenarios. The main characteristics of these data sets are summarized in Table 1, including the total number of instances, the dimension d of the feature space, and the number m of classes forming $\mathcal{C}$.

Table 1: Data sets characteristics

Dataset	# Instances	# Features	# Classes
Spambase	4,601	57	2
KDD Cup 99	825,050	41	5
Connect-4	67,557	42	3
Covertype	581,012	54	7
Higgs	100,000	28	2
Simul-C2		5	2
Simul-C8		3	8

To get a quick understanding on the nature of these data sets, a very brief description of each one is provided below.

- Spambase. Data set based on the properties of diverse "spam" concept. It includes 4,601 instances corresponding to e-mail messages, 1,813 of which are spam. From the original e-mail messages, 57 attributes were computed, most of them indicating whether a particular word or character frequently occurred in the e-mail.

- KDD Cup 99. Benchmark data set in the intrusion detection field, which contains 5 million instances featured by 41 attributes and 39

types of distinct attacks, grouped into four classes of attack (DoS, Probe, R2L and U2R) and one class of non-attack (normal pattern) [1]. In our study, a smaller subset with 494,021 instances is used as training sample (10% of the original training set). For the test set, we used a subset of 331,029 patterns including new attacks that are not present in the training set. Around 20% of the two datasets are normal patterns (no attacks). The percentages of class labels for the training and test sets are shown in Table 2. As can be seen, the percentage of attacks in both data sets is very high, overcoming 80%, where most of the attacks belong to type DoS. Furthermore, it is a very unbalanced data set, with some classes (such as U2R and R2L) formed by very few instances. Due to these characteristics, KDD Cup 99 becomes a real challenge for the classification task.

Table 2: Distribution (in percentage) of normal activities and kinds of attacks in KDD Cup 99 data set.

Type	Training set	Test set
Normal	19.69	19.48
DoS	79.24	73.90
Probe	0.83	1.34
R2L	0.23	5.21
U2R	0.01	0.07

- Connect-4. This data set contains all legal 8-ply positions in the game of Connect-4 in which neither player has won yet, and where the next move is not forced. The dataset contains 67,557 instances represented by 42 attributes indicating whether a particular board position is occupied by the first player, the second player or it is black. The outcome class indicates if the attributes lead to a win, loss, or draw for the first player.

- Covertype. It contains the forest cover type for 30 x 30 meter cells obtained from US Forest Service (USFS) Region 2 Resource Information System (RIS). The database has 581,012 instances. The feature vectors are measurements of 54 cartographic variables used to predict the forest cover type (seven types are available).

- HIGGS. The Higgs boson data set was generated using Monte Carlo simulations of physics events. The feature vectors include: 21 attributes with kinematic properties measured by the particle detectors in the accelerator, and 7 attributes with high-level features derived by physicists from the first 21 attributes. The target is to determine whether or not an event corresponds to the Higgs boson. In our study, a subset of 100,000 instances of the original database is considered.

- SIMUL-C2. Consider a square grid of size 3 in dimension 5 and, centered at each grid node, a 5-dimensional Gaussian distribution with uncorrelated components of equal variance 0.05^2. Each Gaussian is assigned to one of $m = 2$ possible classes at random. In this scenario, an identical number of data are drawn out from each Gaussian to form our first synthetic data set. Figure 3 provides an intuition on the structure of Simul-C2 in dimension 2. This scenario lets us have an exact knowledge of the complexity of the classification task in order to derive some insight into the results.

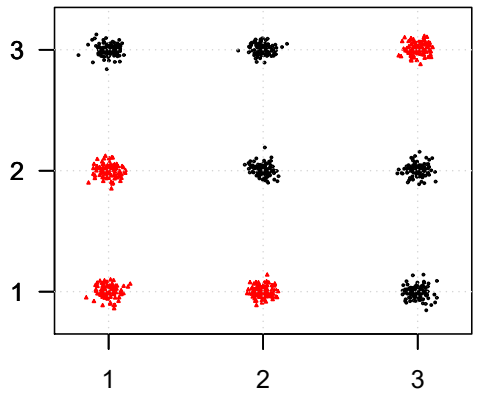

Figure 3: Plot of a simulated trial from a 2-dimensional version of Simul-C2 scenario. Color identifies the class.

- SIMUL-C8. Synthetic data set generated in a similar way as Simul-C2,

but now with 3-dimensional Gaussian distributions randomly assigned to $m = 8$ classes.

Sample size. At each experimental trial, the sizes of both the training sample $\mathcal{Z}$ and the test sample $\mathcal{T}$ are fixed to 500, i.e. $n = t = 500$. The training sample is then equidistributed between the nodes so that $n_i = 500/p$, for all $i = 1, \ldots, p$.

Balancedness. Since no constraints on the fragmentation scheme are imposed, it is interesting to check the behavior of our learning model with balanced and unbalanced nodes, i.e. nodes exhibiting similar or different distributions, respectively. The balanced scenarios are recreated by allocating instances to each node at random and without replacement while maintaining the class proportions. The unbalanced scenarios are set up as follows. First, one node with the same class proportions as the entire training set is formed. For the rest of nodes, the class proportions are perturbed by multiplying each one of them by a random number uniformly generated between 0.3 and 1.7, and then normalizing. In consecutive nodes, the overall class proportions are updated on the basis of the number of remaining training instances, and sampling without replacement is always carried out.

Partition size. To assess the classification accuracy as data fragmentation increases, the training set was randomly split into 2, 4, 7, 11 and 15 nodes. The unique randomization restrictions are imposed by the class proportions at each node, which depend on whether a balanced or unbalanced scenario is considered.

Decision Rules. The belief values generated by the classifiers at each node are combined according to the five decision rules enumerated in Section 2.6, namely the Product, Sum, Max, Min and Majority rules.

3.2. Some motivating experiments

By construction, the proposed learning models take into account the distances between the probability distributions of the features in the population and within each node. The heuristic is that, in general, smaller distributional distances between training and test sets tend to produce better classification results. Indeed, the key issue is how these distances should be jointly used to attain this improvement. Beyond this issue, a pair of motivating experiments designed to provide empirical support for this heuristic have been carried out.

The first experiment consisted in checking for the existence of negative correlation between distributional distance and classification accuracy. In the second one, a distributed scenario is considered, and then the proportion of times that the node with the smallest distributional distance produces the best classification accuracy is measured.

For these specific experiments, the Spambase data set is used and the within-node distributions are generated according to the unbalancing approach described in Section 3.1. The number of nodes is set to $p = 5$ and a training sample of size $n_i = 200$ is used at each node to train a Random Forest classifier.

Considering test samples with the same size, $t = 200$, the first experiment consisted in measuring the distributional distances between training and test samples using the energy distance d_i introduced in Section 2.4, and simultaneously computing the proportion of test data correctly classified at each node. This process was performed for a large number of trials, and the outputs are plotted in Figure 4. A clear negative correlation between distributional distance and classification accuracy is observed, thus supporting the argument that less distant nodes tend to produce better classification accuracy.

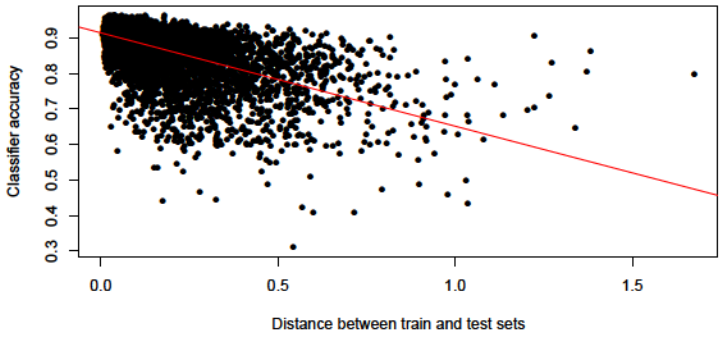

Figure 4: Correlation between distributional distance and classifier accuracy.

In the second experiment, the test sample size is not constant at all trials, taking values moving from 200 to 3200. Notice that the distributional distance becomes more accurately approximated as test sample size increases, and therefore the classification accuracy should be also higher. In a five node

distributed scenario, the expected proportion of times that a node picked at random produces the best classifier is 0.2. Table 3 shows the proportion of times that the node with the smallest distributional distance produced the highest classification accuracy in our experiment, denoted by $p_{\min(d_i)}$. It is observed that the node with the smallest distributional distance becomes the best one in an increasing proportion with the test sample size, always above the baseline proportion 0.2, until it is approximately doubled.

Table 3: Proportion of times that the smallest distributional distance leads to the best classifier node ($p_{\min(d_i)}$) against the test sample size (t).

t	200	1200	2200	3200
$p_{\min(d_i)}$	0.28	0.35	0.37	0.37

In sum, these first experiments empirically illustrate the interest in distributed learning models regarding distributional distances between training nodes and test samples. We propose models taking into consideration this principle, but in addition, they take advantage of combining efficiently the classifiers produced by each node instead of simply selecting one of them.

3.3. Results

The accuracy of the two weighting criteria (pNW and pIW) described in Section 2.5 was checked on all the combinations of parameters involved in our experimental setup, namely classifiers, data sets, partition sizes, decision rules, and balanced and unbalanced scenarios (Section 3.1). For comparison purposes, accuracy results based on a standard unweighted distributed model (UW) and a non-distributed model (ND) were also obtained. The ND model uses the entire data set $\mathcal{Z}$ to train a unique classifier. Hence, ND is expected to achieve the highest classification accuracy, and its results can be taken as an upper reference level. Besides the classification accuracy, values of precision and recall were also evaluated. Since very similar conclusions are obtained, for the sake of clarity in the presentation, the results based on precision and recall are provided in Appendix A.

For each combination of parameters, the experiment was replicated $N = 300$ times and average classification accuracy values were obtained for each learning model. In order to examine how the learning models interact with the different parameters, the average results were aggregated in different

ways. For example, Table 4 shows the average accuracy attained with each classifier. It is observed that the weighted models interact better with SVM, LDA and Mult than with Random Forest and XGBoost in the unbalanced setting (see Figure 5). In particular, the most significant improvement rates due to the pIW model in the unbalanced setup are observed for Mult and SVM. In the latter case, this may be connected with the fact that SVM with Gaussian kernel and energy distance are based on Euclidean inter-point distances.

Table 4: Average classification accuracy values conditional on classifier type. Rows under the last sub-table (MEAN) show the averages over all trials, including balanced and unbalanced scenarios. The lack of results for ND in the first two sub-tables is due to the ND model assumes non-distributed data, i.e. no partitions (balanced or unbalanced) are considered.

	Classifier				
Model	RF	SVM	XGB	LDA	Mult
BALANCED					
pNW	0.7087	0.5852	0.6923	0.5768	0.6084
pIW	0.7173	0.5908	0.7016	0.5826	0.6168
UW	0.7131	0.5881	0.6964	0.5769	0.6066
UNBALANCED					
pNW	0.6935	0.5804	0.6843	0.5727	0.6035
pIW	0.7059	0.5921	0.6977	0.5863	0.6186
UW	0.6996	0.5784	0.6909	0.5749	0.6022
MEAN					
pNW	0.7011	0.5828	0.6883	0.5747	0.6060
pIW	0.7116	0.5915	0.6997	0.5845	0.6177
UW	0.7063	0.5832	0.6937	0.5759	0.6044
ND	0.7566	0.6719	0.7398	0.6315	0.6423

The average results aggregated by decision rule are reported in Table 5 and graphically represented using bar charts in Figure 6. Regardless of whether the partitioning is balanced or not, the SUM rule produces the best average results with the three distributed models. This result is consistent with the experimental findings in Kittler *et al.* [12] and with our theoretical

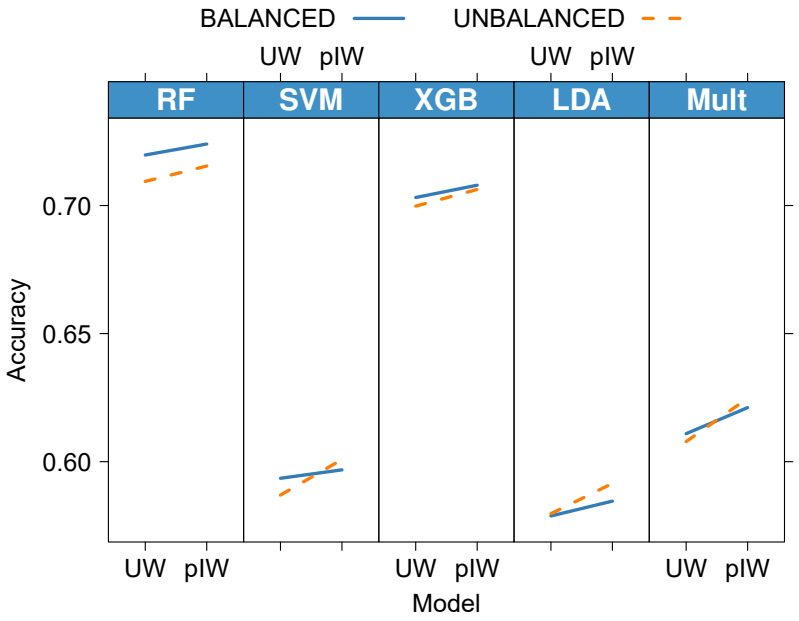

Figure 5: Accuracy-based interaction plot to check the joint effect of classifier, learning model and scenario.

arguments in Section 2 (Remark 3 in Section 2.7).

Table 5 and Figure 6 also allow to compare the average accuracy attained with the different models. Except for the MIN and PROD rules, the highest accuracy values are obtained with the per-Instance Weighting. Nevertheless, the MIN rule fairly produces the worst results and no differences between weighting approaches are observed with the PROD rule. Therefore, it is concluded that the per-Instance Weighting approach fairly leads to the best results.

Overall, pNW performs worse than UW on average. This behavior is somewhat surprising in the light of the results showed in the motivating experiments of Section 3.2. We guess that this may be caused by the joint effect of two circumstances, namely the global character of the per-Node weights (see Remark 1 in Section 2.7) and the noise increase generated by the variability of these weights (Figure 4 illustrates this variability).

The average accuracies aggregated by partition size are shown in Figure 7. Significant degradation of accuracy with the number of nodes is evident for all combinations of decision rule and learning model in balanced and un-

Table 5: Average classification accuracy values conditional on the decision rules.

	Decision rule					
Model	MAJ	MAX	MIN	PROD	SUM	Mean
BALANCED						
pNW	0.6442	0.6252	0.6184	0.6347	0.6491	0.6343
pIW	0.6582	0.6461	0.6066	0.6345	0.6639	0.6418
UW	0.6456	0.6283	0.6205	0.6347	0.6519	0.6362
Mean	0.6493	0.6332	0.6152	0.6346	0.6549	0.6374
UNBALANCED						
pNW	0.6407	0.6191	0.5977	0.6276	0.6493	0.6269
pIW	0.6652	0.6513	0.5823	0.6275	0.6743	0.6401
UW	0.6454	0.6189	0.6008	0.6277	0.6531	0.6292
Mean	0.6504	0.6297	0.5936	0.6276	0.6589	0.6321
ND	—	—	—	—	—	0.6884

balanced scenarios. For all the models, the MIN rule degrades faster with fragmentation, although it is very competitive with UW and pNW for a small number of nodes. As the best-performing pIW approach is considered, the MIN rule is substantial and uniformly the worst decision rule. SUM, MAJ and MAX exhibit similar performance, with SUM having a slight edge. The good behavior of pIW deserves particular attention. Note that, except for the MIN rule, the pIW approach always produces the highest percentages of correct classification for all the levels of fragmentation regardless of the used rule.

All our results allow to conclude that the favorable effects of the weighting approaches are more important in unbalanced scenarios. In particular, the amount of accuracy improvement produced by the per-Instance Weighting model is clearly stronger when unbalancing.

Figure 8 shows separately the average results for each data set and provides additional insight into the behavior of the proposed models. Concerning the KDD Cup 99 data set, it is noticeable the good performance showed by the per-Node Weighting approach in unbalanced scenarios. In fact, pNW and pIW behave very similarly and fairly outperform the Unweighted model. Since KKD Cup 99 exhibits concept drift, this result illustrates that our dis-

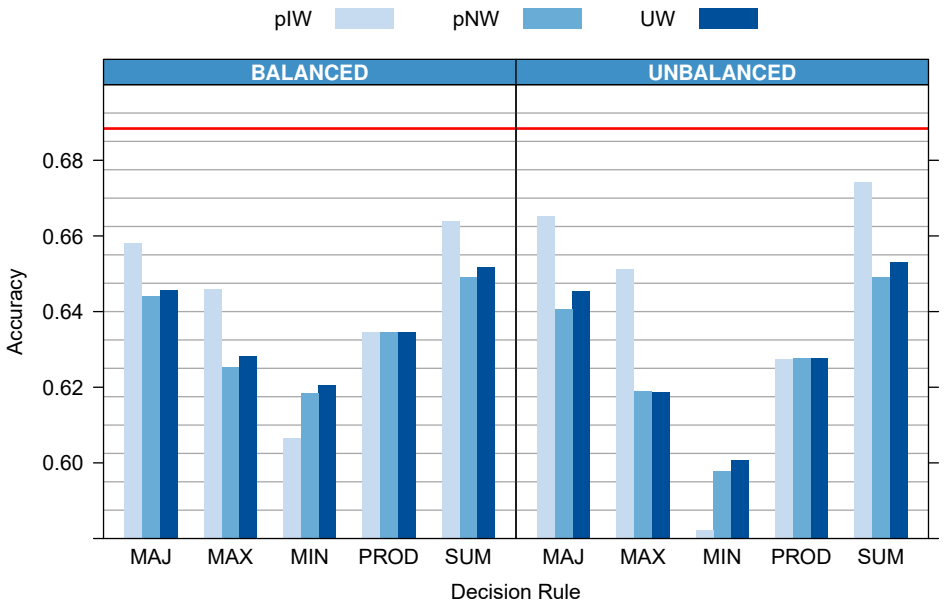

Figure 6: Average classification accuracy values aggregated by decision rules. The horizontal red line indicates the average accuracy for the ND model.

tributed approaches work well for various types of differences in distribution between nodes and population, no matter how these differences occur. In other words, the effectiveness of our proposal is not restricted to the case of distributional differences caused by a non-uniform partition of the data.

The pIW model reports lower accuracy with Spambase, while it performs slightly better than the other distributed models in Connect-4, Covertype and Higgs, the most complex scenarios in terms of classification. In these cases, the non-distributed model only reaches an accuracy around 0.6, and the distributed approaches are reasonably close to this proportion for all partition sizes. An atypical behavior is observed for Connect-4 since the classification accuracy does not monotonically decrease with the number of partitions. So, in this particular case, it looks more likely that the local scenarios generated by splitting the data lead to an easier classification task.

In addition to knowing the exact underlying distributions, the analysis of simulated data is free of limitations to generate nodes maintaining the required regularity and provides insight into the level of difficulty. The results for our simulated data sets, Simul-C2 and Simul-C8, are particularly inter-

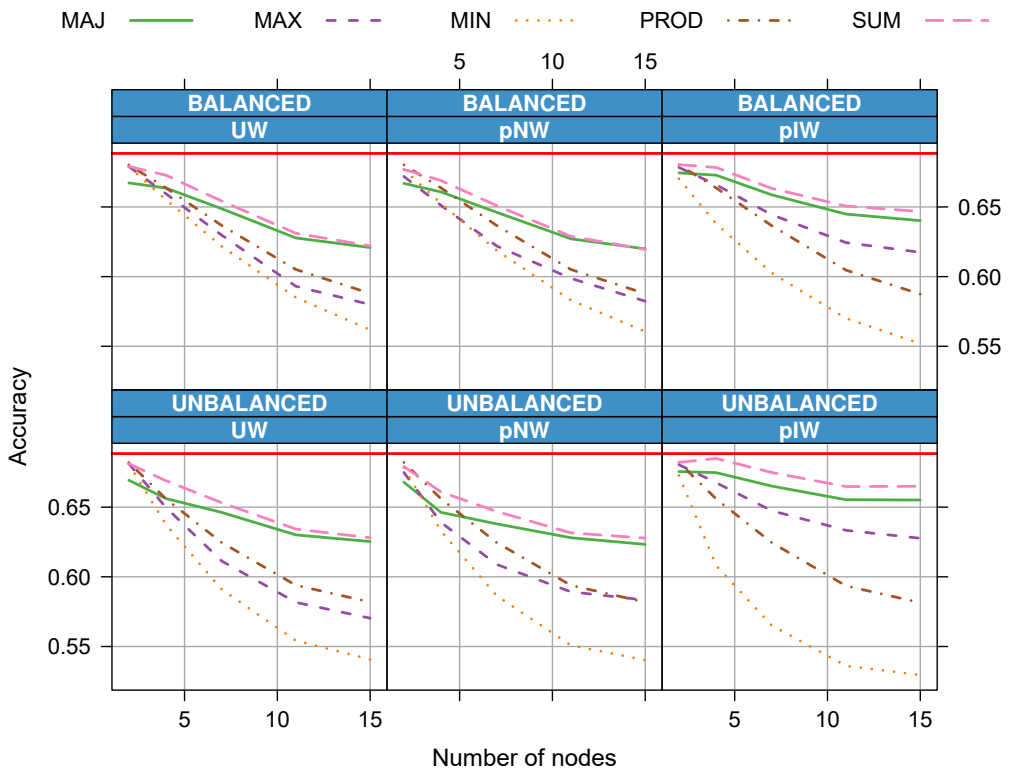

Figure 7: Average accuracy as function of the partition size. The horizontal red lines indicate the average accuracy for the ND model.

esting. In both cases, pIW clearly draws the best results, and the differences are more substantial in the unbalanced setting. In the scenario with only two classes, degradation with the number of nodes is almost prevented in the balanced setting, and the results end up surpassing the non-distributed approach in Simul-C2. In the non-distributed approach, a linear classifier will have bad performance in this scenario, since the classification frontier is non linear. When distributed, different local models work better. In Simul-C8, degradation of pIW with the fragmentation is more marked, but still less severe than in the case of pNW and UW.

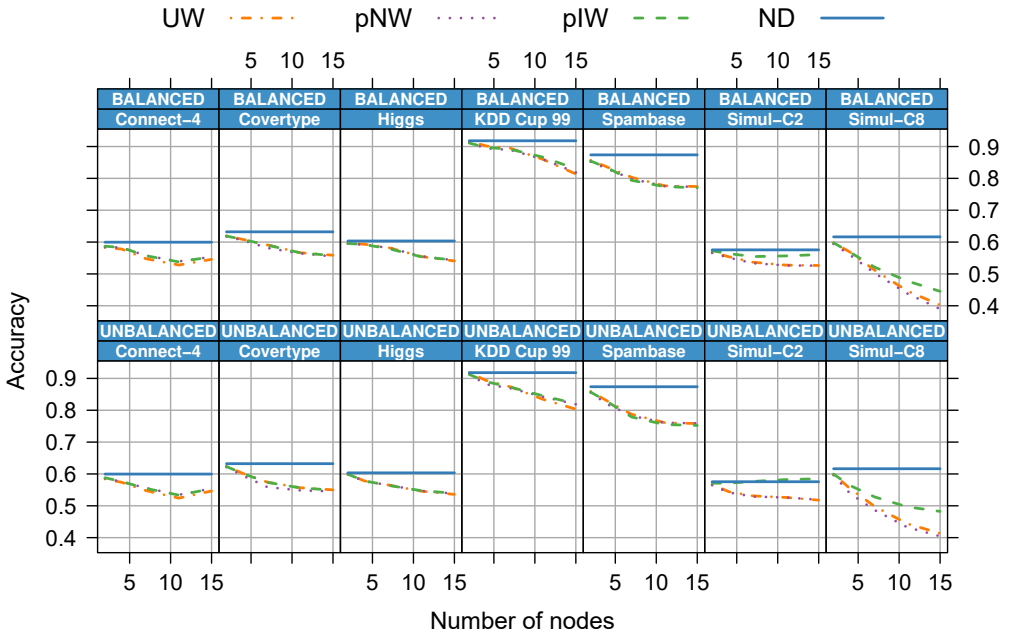

Figure 8: Average accuracy as function of the partition size for each data set. The horizontal blue lines indicate the average accuracy for the ND model.

4. Computational Cost

In addition to the cost of training each classifier, our approach requires calculating the distributional distance between each node and the test set, and the pIW model requires each node to calculate the weights that minimize this distance.

The cost of training the classifiers depends on the chosen method, but the improvement is equivalent to reducing the training sample by the number of nodes, e.g. if a given classifier trains in $\mathcal{O}(n^3)$ then the complexity will be reduced to $\mathcal{O}(n^3/p^3)$. Similarly, calculating the energy statistic has computational complexity $\mathcal{O}((n/p)^2 + t^2 + (n/p)t)$, given n the training size, p the number of nodes that the training set is fragmented into and t the test size. Even though it can be considered a constant factor, we introduce the number of nodes p to highlight the strong computational benefits of this distributed approach.

Finding the individual weights for the pIW approach has the complexity of solving the related quadratic programming problem, and is the usually

dominating complexity: Experimentally, this quadratic programming complexity is between $\mathcal{O}(t^2)$ and $\mathcal{O}(t^3)$. The influence of the training test size on the quadratic programming complexity is linear: $\mathcal{O}(n/p)$.

Assuming the underlying classifiers test in constant time, the complexity of classifying a given test set goes from $\mathcal{O}(t)$ to a worst case of $\mathcal{O}(t^3 + n)$ for the pIW version.

5. Conclusions and future work

In a distributed classification framework, we have proposed two weighted approaches that combine local classifiers trained at each node to improve overall classification accuracy. The two approaches assume the availability of a test set and are based on the distance between the distributions of the feature vectors of each node and the test set. The first approach, per-Node Weighting, assigns the same weight at each node to all test instances, while the per-Instance Weighting approach achieves finer granularity by allowing distinct weights for each test instance at each node.

Under the general assumption that both the test set and the entire training set are i.i.d. samples from the population in study, we have motivated the proposed weighting criteria and provided theoretical support for the optimality of combining the classifier outcomes using a weighted sum. Our framework makes no assumptions about the structure or distribution of the data across the nodes. In fact, by construction, our classification models are particularly useful to deal with heterogeneity of data among the nodes, which usually happens in real-world distributed data sets. In addition, our technique requires no communication between nodes, preserving data privacy, allowing combination of different classifier models and maximizing computational efficiency.

Our experimental study involving synthetic and real data sets has illustrated the good performance of the proposed models compared to standard classifier combination rules. Overall, the per-Instance Weighting approach achieves the best results. As expected, the improvement is more substantial when treating with unbalanced nodes, under all tested classifiers an partition sizes. Our experiments also illustrate that the sum rule outperforms other alternative decision rules. The per-Node Weighting approach does not achieve improvement over the standard approaches but on the most extreme cases when the individual nodes training sets differ the most.

There are several topics related to our approach to be considered further. It is interesting to check by the usefulness of our approach to select one or a small subset of nodes to perform the classification and evaluating whether a significant degradation in accuracy is observed. Other future research direction involves the study of a linear-time approximation of the pIW approach.

Acknowledgments

This research has been supported by Spanish Ministerio de Economía y Competitividad (grants TIN2015-65069-C2-1-R, MTM2014-52876-R and MTM2017-82724-R), and by the Consellería de Industria of the Xunta de Galicia (projects GRC2014/035, Grupos de Referencia Competitiva ED431D-R2016/045 and ED431C-2016-015, and Centro Singular de Investigación de Galicia ED431G/01), all of them through the European Regional Development Fund, ERDF.

References

[1] KDD Cup 99 dataset. [Online; accessed April-2017].

[2] Ron Bekkerman, Mikhail Bilenko, and John Langford. Scaling up machine learning: Parallel and distributed approaches. In *Proceedings of the 17th ACM SIGKDD International Conference Tutorials*, KDD '11 Tutorials, pages 4:1–4:1, New York, NY, USA, 2011. ACM.

[3] Gema Bello-Orgaz, Jason J Jung, and David Camacho. Social big data: Recent achievements and new challenges. *Information Fusion*, 28:45–59, 2016.

[4] Leo Breiman. Pasting small votes for classification in large databases and on-line. *Machine learning*, 36(1):85–103, 1999.

[5] Phillip K Chan, Salvatore J Stolfo, et al. Toward parallel and distributed learning by meta-learning. In *AAAI workshop in Knowledge Discovery in Databases*, pages 227–240, 1993.

[6] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794. ACM, 2016.

[7] Hal Daumé, Jeff M Phillips, Avishek Saha, and Suresh Venkatasubramanian. Efficient protocols for distributed classification and optimization. In *International Conference on Algorithmic Learning Theory*, pages 154–168. Springer, 2012.

[8] Travis Dick, Mu Li, Venkata Krishna Pillutla, Colin White, Maria-Florina Balcan, and Alexander J. Smola. Data driven resource allocation for distributed learning. *CoRR*, abs/1512.04848, 2015.

[9] Thomas G Dietterich. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pages 1–15. Springer, 2000.

[10] Marthinus Christoffel Du Plessis and Masashi Sugiyama. Semi-supervised learning of class balance under class-prior change by distribution matching. *Neural Networks*, 50:110–119, 2014.

[11] Jiayuan Huang, Alexander J. Smola, Arthur Gretton, Karsten M. Borgwardt, and Bernhard Scholkopf. Correcting sample selection bias by unlabeled data. In *Proceedings of the 19th International Conference on Neural Information Processing Systems*, NIPS'06, pages 601–608, Cambridge, MA, USA, 2006. MIT Press.

[12] Josef Kittler, Mohamad Hatef, Robert PW Duin, and Jiri Matas. On combining classifiers. *IEEE transactions on pattern analysis and machine intelligence*, 20(3):226–239, 1998.

[13] Aleksandar Lazarevic and Zoran Obradovic. The distributed boosting algorithm. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 311–316. ACM, 2001.

[14] Andy Liaw and Matthew Wiener. Classification and regression by randomforest. *R News*, 2(3):18–22, 2002.

[15] M. Lichman. UCI machine learning repository, 2013. [Online; accessed April-2017].

[16] David Meyer, Evgenia Dimitriadou, Kurt Hornik, Andreas Weingessel, and Friedrich Leisch. *e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien*, 2015. R package version 1.6-7.

[17] Diego Peteiro-Barral and Bertha Guijarro-Berdiñas. A survey of methods for distributed machine learning. *Progress in Artificial Intelligence*, 2(1):1–11, 2013.

[18] Joaquin Quionero-Candela, Masashi Sugiyama, Anton Schwaighofer, and Neil D Lawrence. *Dataset shift in machine learning*. The MIT Press, 2009.

[19] Miguel Ángel Rodríguez, Alberto Fernández, Antonio Peregrín, and Francisco Herrera. A review of distributed data models for learning. *Hybrid Artificial Intelligent Systems*, page 88, 2017.

[20] Marina Sokolova and Guy Lapalme. A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, 45(4):427–437, 2009.

[21] Gábor J Székely and Maria L Rizzo. Testing for equal distributions in high dimension. *InterStat*, 5:1–6, 2004.

[22] Gábor J Székely and Maria L Rizzo. Energy statistics: A class of statistics based on distances. *Journal of statistical planning and inference*, 143(8):1249–1272, 2013.

[23] Grigorios Tsoumakas and Ioannis Vlahavas. Effective stacking of distributed classifiers. In *Proceedings of the 15th European conference on artificial intelligence*, pages 340–344. IOS Press, 2002.

[24] Grigorios Tsoumakas and Ioannis Vlahavas. Distributed data mining. *Encyclopedia of Data Warehousing and Mining*, 2009.

[25] Sujatha R Upadhyaya. Parallel approaches to machine learning—A comprehensive survey. *Journal of Parallel and Distributed Computing*, 73(3):284–292, 2013.

[26] W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S*. Springer, New York, fourth edition, 2002. ISBN 0-387-95457-0.

Appendix A.

The same numerical analysis performed in Section 3.3 with the accuracy values has been carried out for recall and precision values, two alternative performance measures. The attained results are shown in this Appendix. In the case of binary classification, recall measures the effectiveness of a classifier to identify correctly classified positive instances (sensitivity), while precision evaluates the class agreement of the instance labels with the positive labels given by the classifier. One possible way to extend these concepts to the multi-class classification task is to obtain the averages of these measures calculated over all the classes $\{C_1, \ldots, C_m\}$. This generalization approach is known by *macro-averaging* [20]. This way, we have

$$\begin{aligned} Precision &= \frac{1}{m}\sum_{i=1}^{m}\frac{tp_i}{tp_i + fp_i}, \\ Recall &= \frac{1}{m}\sum_{i=1}^{m}\frac{tp_i}{tp_i + fn_i}, \end{aligned}$$

with m the number of classes in the dataset, and tp_i denoting the number of true positive for C_i, and fp_i and fn_i the false positive and false negative counts, respectively.

The results attained for these alternative criteria are displayed below, using the same scheme of tables and figures as in Section 3.3 for accuracy. It can be seen that very similar results are also obtained, thus supporting the main conclusions of our work.

Table A.6: Average recall values conditional on classifier type.

	Classifier				
Model	RF	SVM	XGB	LDA	Mult
BALANCED					
pNW	0.7154	0.5912	0.6991	0.5788	0.6128
pIW	0.7240	0.5968	0.7079	0.5846	0.6211
UW	0.7197	0.5935	0.7031	0.5788	0.6109
UNBALANCED					
pNW	0.7029	0.5890	0.6926	0.5777	0.6093
pIW	0.7154	0.6008	0.7063	0.5915	0.6247
UW	0.7094	0.5869	0.6998	0.5797	0.6078
MEAN					
pNW	0.7091	0.5901	0.6959	0.5783	0.6110
pIW	0.7197	0.5988	0.7071	0.5880	0.6229
UW	0.7146	0.5902	0.7015	0.5792	0.6094
ND	0.7676	0.6794	0.7477	0.6418	0.6515

Table A.7: Average recall values conditional on the decision rules.

	Decision rule					
Model	MAJ	MAX	MIN	PROD	SUM	Mean
BALANCED						
pNW	0.6505	0.6291	0.6237	0.6401	0.6539	0.6395
pIW	0.6633	0.6506	0.6119	0.6399	0.6687	0.6469
UW	0.6507	0.6323	0.6262	0.6402	0.6567	0.6412
Mean	0.6548	0.6374	0.6206	0.6401	0.6598	0.6425
UNBALANCED						
pNW	0.6493	0.6245	0.6060	0.6353	0.6564	0.6343
pIW	0.6733	0.6582	0.5895	0.6351	0.6826	0.6478
UW	0.6527	0.6257	0.6086	0.6353	0.6613	0.6367
Mean	0.6584	0.6362	0.6014	0.6353	0.6668	0.6396
ND	—	—	—	—	—	0.6976

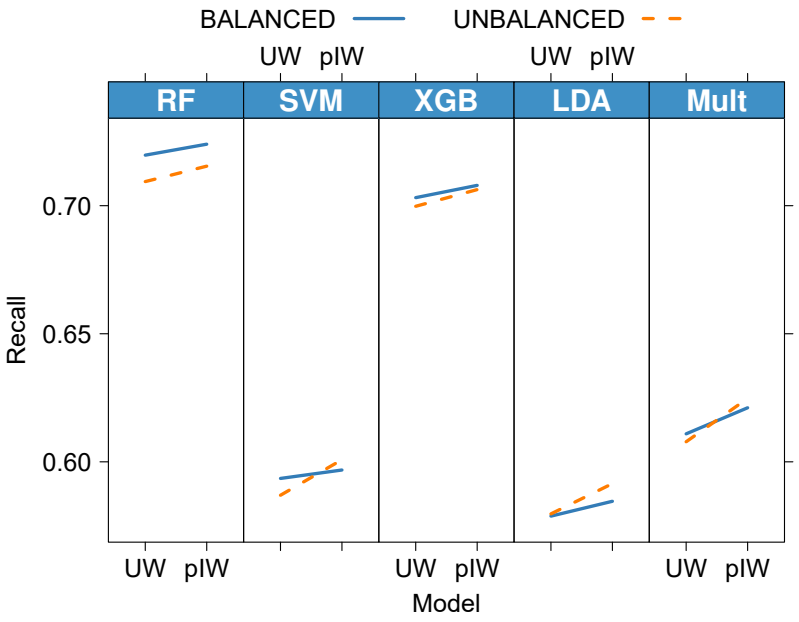

Figure A.9: Recall-based interaction plot to check the joint effect of classifier, learning model and scenario.

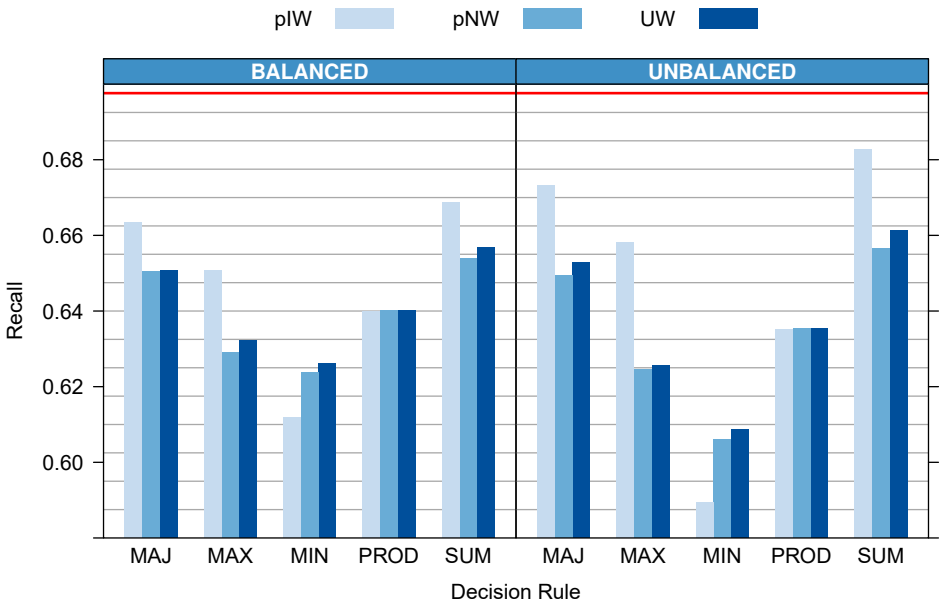

Figure A.10: Average recall values aggregated by decision rules. The horizontal red line indicates the average recall for the ND model.

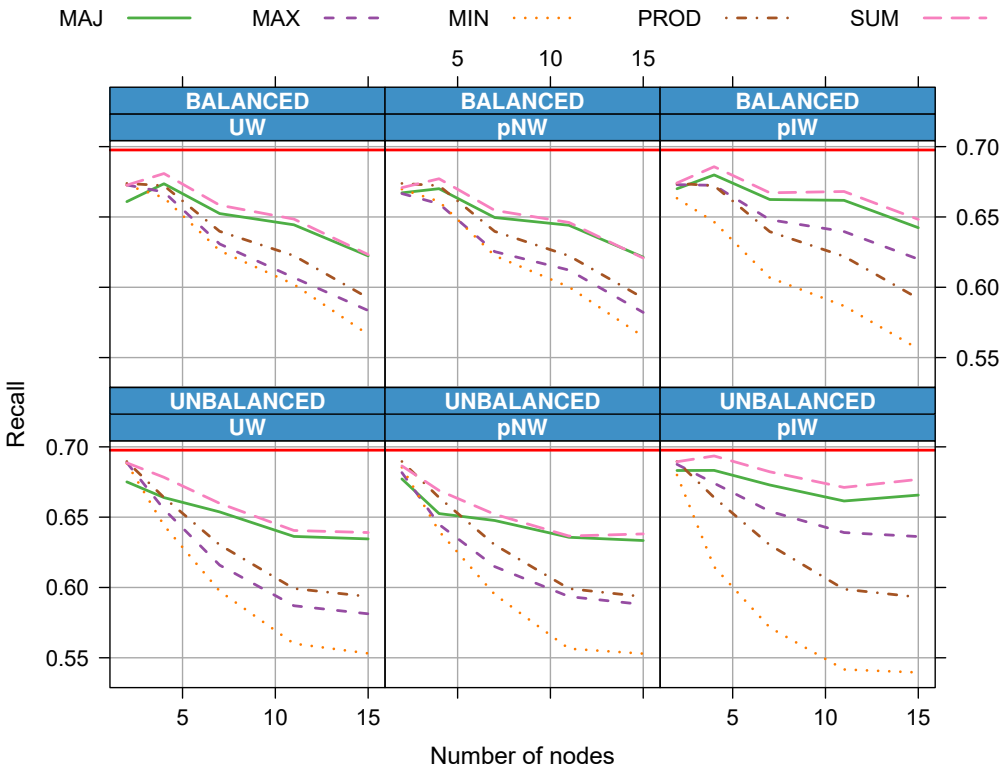

Figure A.11: Average recall as function of the partition size. The horizontal red lines indicate the average recall for the ND model.

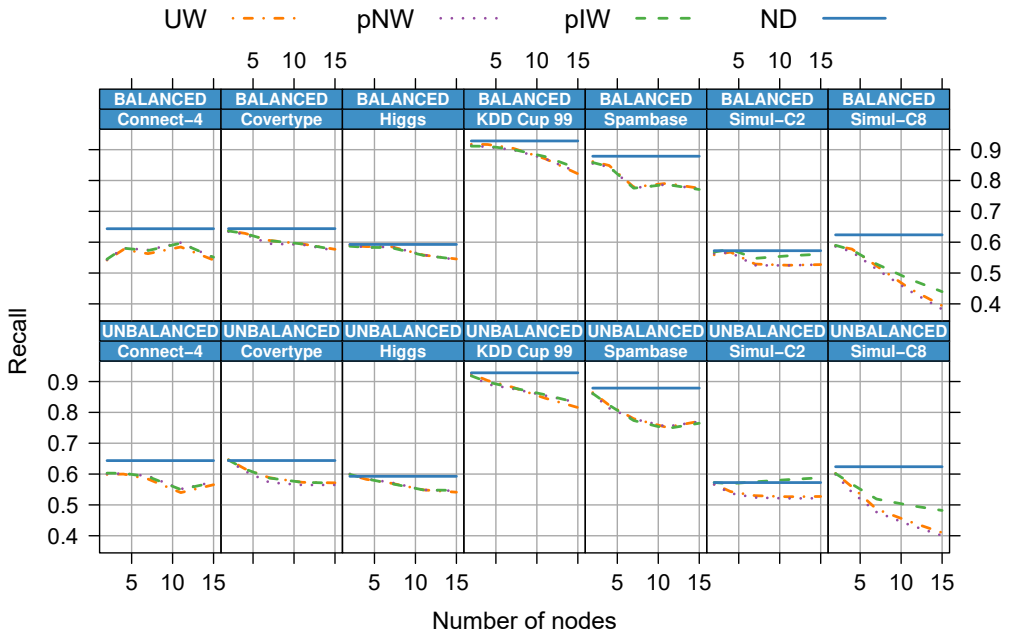

Figure A.12: Average recall as function of the partition size for each data set. The horizontal blue lines indicate the average recall for the ND model.

Table A.8: Average precision values conditional on classifier type.

	Classifier				
Model	RF	SVM	XGB	LDA	Mult
BALANCED					
pNW	0.7016	0.5631	0.6896	0.5833	0.6118
pIW	0.7113	0.5723	0.6989	0.5895	0.6200
UW	0.7060	0.5665	0.6935	0.5818	0.6103
UNBALANCED					
pNW	0.6875	0.5577	0.6827	0.5815	0.6098
pIW	0.7001	0.5760	0.6956	0.5948	0.6249
UW	0.6929	0.5569	0.6879	0.5819	0.6090
MEAN					
pNW	0.6945	0.5604	0.6862	0.5824	0.6108
pIW	0.7057	0.5741	0.6973	0.5922	0.6224
UW	0.6994	0.5617	0.6907	0.5819	0.6097
ND	0.7516	0.6586	0.7392	0.6405	0.6496

Table A.9: Average precision values conditional on the decision rules.

	Decision rule					
Model	MAJ	MAX	MIN	PROD	SUM	Mean
BALANCED						
pNW	0.6392	0.6239	0.6145	0.6282	0.6436	0.6299
pIW	0.6550	0.6459	0.6030	0.6280	0.6601	0.6384
UW	0.6408	0.6266	0.6166	0.6283	0.6459	0.6316
Mean	0.6450	0.6321	0.6113	0.6282	0.6499	0.6333
UNBALANCED						
pNW	0.6378	0.6193	0.5945	0.6226	0.6451	0.6238
pIW	0.6637	0.6531	0.5799	0.6224	0.6724	0.6383
UW	0.6420	0.6178	0.5985	0.6226	0.6477	0.6257
Mean	0.6478	0.6300	0.5910	0.6225	0.6551	0.6293
ND	—	—	—	—	—	0.6879

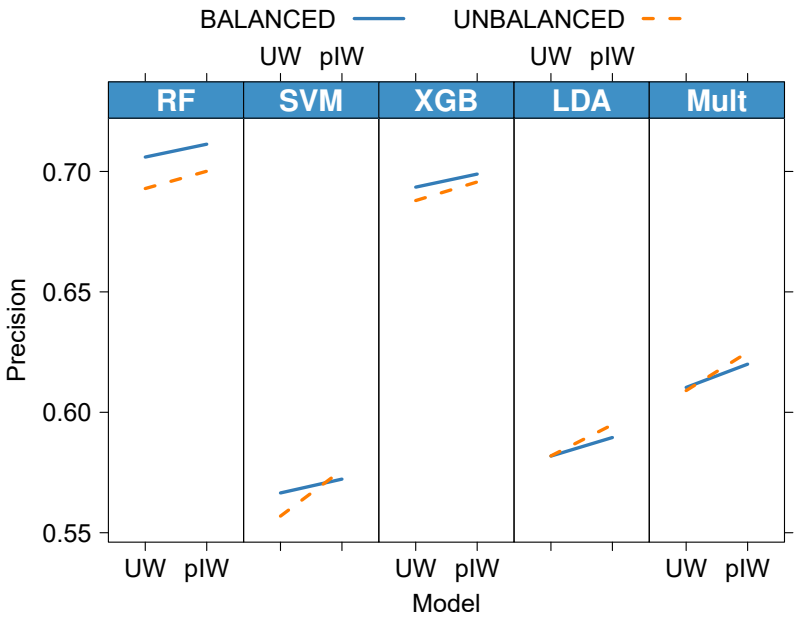

Figure A.13: Precision-based interaction plot to check the joint effect of classifier, learning model and scenario.

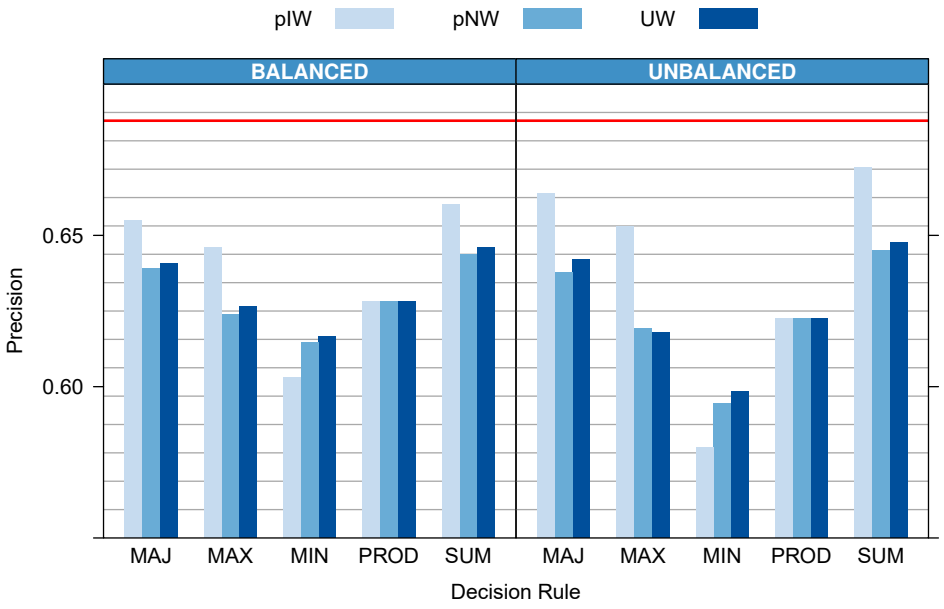

Figure A.14: Average precision values aggregated by decision rules. The horizontal red line indicates the average precision for the ND model.

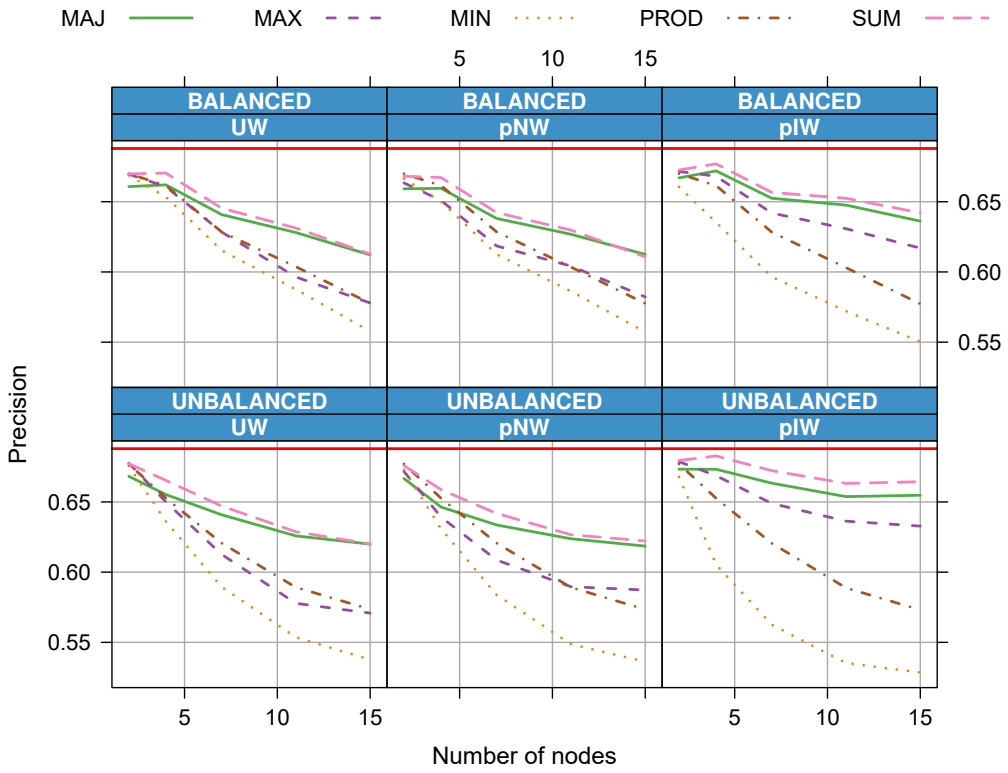

Figure A.15: Average precision as function of the partition size. The horizontal red lines indicate the average precision for the ND model.

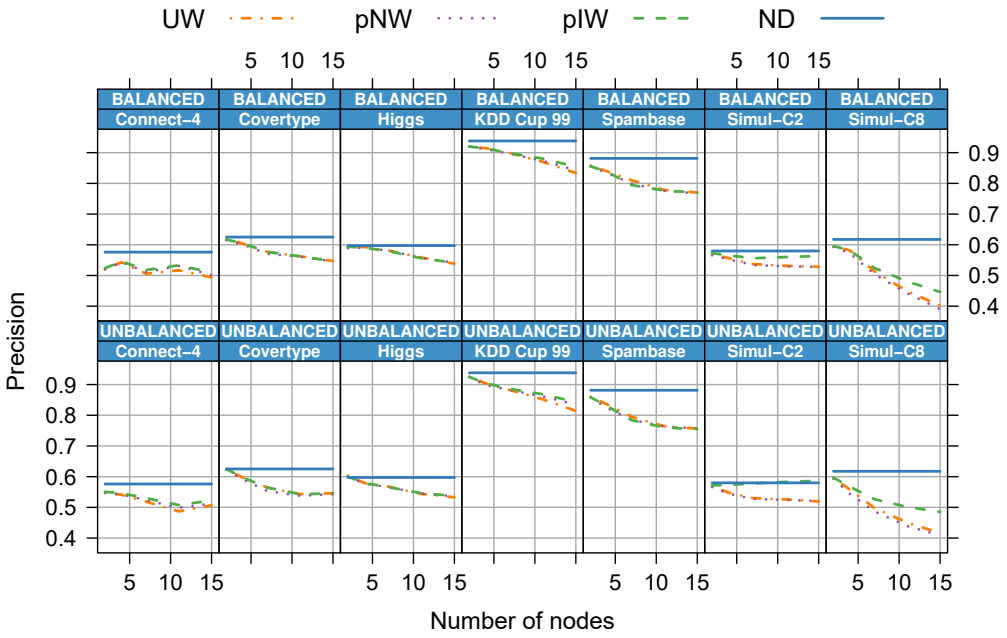

Figure A.16: Average precision as function of the partition size for each data set. The horizontal blue lines indicate the average precision for the ND model.