Cost-Sensitive Measures of Algorithm Similarity for Meta-Learning

Carlos Eduardo Castor de Melo
Universidade Federal de Pernambuco
Centro de Informática, Recife, Pernambuco
Email: cecm2@cin.ufpe.br

Ricardo Bastos Cavalcante Prudêncio
Universidade Federal de Pernambuco
Centro de Informática, Recife, Pernambuco
Email: rbcp@cin.ufpe.br

***Abstract*—Knowledge about algorithm similarity is an important aspect of meta-learning, where the information gathered from previous learning tasks can be used to guide the selection of algorithms for new datasets. Usually this task is done by comparing global performance measures across different datasets or alternatively, comparing the performance of algorithms at the instance-level. In both cases, the previous similarity measures do not consider misclassification costs, and hence they neglect an important information that can be exploited in different learning contexts. In this paper we present algorithm similarity measures that deals with cost proportions and different threshold choice methods for building crisp classifiers from learned models. Experiments were performed in a meta-learning study with 50 different learning tasks. The similarity measures were adopted to cluster algorithms according to their aggregated performance on the learning tasks. The clustering process revealed similarity between algorithms under different perspectives.**

I. Introduction

Meta-learning is a framework developed in supervised machine learning for acquiring knowledge from empirical case studies and relating features of learning problems to algorithm performance [1]. The knowledge acquired in meta-learning has been used to support algorithm selection in different contexts [2] [3]. An important assumption of meta-learning is that algorithm performance is similar for similar problems. Hence, to know how similar is the performance obtained by different algorithms is important for meta-learning. This information can be used to discover similarities between algorithms and between datasets [4] and also to support the prediction of suitable algorithms for a given dataset [3].

Deploying global metrics, such as accuracy and AUC, to compare the performance of algorithms on a dataset has been the most common approach to deal with the above issue. Despite the popularity of this approach, it may lose important knowledge about algorithm similarity since it is based on average algorithm performance without considering differences of algorithm behaviour in an instance-level. In order to overcome this limitation, alternative measures of algorithm similarity (e.g., error correlation [4] or classifier output difference [5]) have been adopted. Here, algorithms are considered similar if they produce the same results on the same instances.

Algorithm similarity measures adopted in these previous work are limited since they are not flexible enough to consider different misclassification costs and different strategies to build classifiers. Algorithms usually produce models (scoring functions) that return scores or class probabilities for the input examples. A classifier is then built from a model by adopting a decision threshold. The prediction for a given example depends on both the score returned by the learned model for that example and the decision threshold. It is very common to adopt a fixed decision threshold (e.g., 0.5) to produce classifiers. Alternatively, decision thresholds can be chosen according to the operating conditions or contexts (e.g., class frequencies and misclassification costs) observed when the learned model is deployed. In [6], the authors showed that different threshold choice methods require the use of different performance measures to evaluate a learned model. Similarly, concerning algorithm similarity, specific measures have to be defined when misclassification costs and adaptive threshold choice methods are taken into account. Previous work on algorithm similarity measures does not consider such aspects. The previous similarity measures are defined implicitly assuming fixed decision thresholds and thus are only suitable for comparing the performance of crisp classifiers.

Based on the above motivation, we developed similarity measures for algorithm performance taking into account cost proportions and different threshold choice methods. As in [2], we adopted an instance-level strategy. For that, we initially proposed instance hardness measures to indicate how difficult is an instance to be correctly classified by a model. Specific instance hardness measures were proposed for two threshold choice methods: (1) score-driven method [7]; and (2) rate-driven method [8]. By assuming a threshold choice method and a learned model, we built for each instance a corresponding *cost curve*, which indicates the model loss for that instance along different misclassification costs. The instance hardness is defined as the area under the cost curve. The dissimilarity[1] between two algorithms for a given dataset is defined as the average absolute difference between the hardness values over all instances of the dataset.

In our work, we applied the proposed measures in a meta-learning study in which 50 datasets and 8 learned models were adopted. Clusters of models were produced by adopting both the score-driven and the rate-driven measures. Each cluster reveals which algorithms produced similar learned models on the 50 datasets under each cost-sensitive perspective. We observe that the algorithm behaviour among the datasets was quite distinct by using the score-driven and rate-driven measures.

[1]In this paper we treat *dissimilarity* as the complement of *similarity*, henceforth we will use only the latter term

978-1-4799-5618-0/14 $31.00 © 2014 IEEE
DOI 10.1109/BRACIS.2014.13

II. Measuring algorithm similarity

Meta-learning deals with methods to exploit knowledge gathered from learning on different tasks [1]. Differently from base-level learning, which focuses on acquiring experience on a single task, meta-learning acquires knowledge from a *meta-data* set produced when a pool of algorithms is applied on distinct learning tasks. Given a learning task, a meta-example usually stores characteristics of the task and the performance obtained by a pool of candidate algorithms. Meta-learning can be applied: (1) in a descriptive sense aiming to discover similarities between datasets and algorithms and (2) in a predictive sense to select candidate algorithms for new datasets based on the knowledge acquired in the meta-learning process.

Different meta-learning approaches rely on the use of similarity measures of algorithm performance. For instance, clustering algorithms based on their performance can provide useful information to guide algorithm selection and combination [4], [5]. As another line of research, we can cite the landmarking approach [3], which uses the performance of simple algorithms (called landmarkers) to guide the selection of more complex ones. In this approach, datasets are characterized based on the performance obtained by the landmarkers (which are usually simple and diverse algorithms, such as decision stumps and linear models). Given a new dataset, the most similar meta-examples are retrieved from the meta-data based on the similarity of performance obtained by the landmarkers. The best candidate algorithms adopted in the retrieved problems are then suggested for the new dataset.

The most common approach for measuring algorithm similarity is the use of global performance measures, like accuracy, estimated from an empirical evaluation process (such as cross-validation). Similarity between algorithms can be obtained by computing the differences or ratios between the performance measures, as performed in [9]. Although this approach is widely applied, it is strongly criticized since it may lose important information about algorithms behaviour and may not characterize similarity properly. For example, if a dataset has 100 instances and a given algorithm misclassifies 20 instances and another one misclassifies 20 instances completely different from the first ones, the accuracy of both algorithms will be the same but their behaviour is quite different.

A more fine-grained approach to algorithm similarity is to consider the performance at each instance of the dataset. This approach has the advantage of showing local discrepancies of performance among the space of instances. In [4], error correlation is adopted as similarity measure between two algorithms, in such a way that two algorithms are similar with they produce the same errors. In [5], the authors present the measure Classifier Output Difference (COD) as an alternative for this approach. This metric is defined as the probability of two distinct classifiers make different predictions [10].

Although this approach represents a more refined view about algorithm performance, the measures proposed in the literature are computed from predictions generated by crisp classifiers (i.e., with fixed decision thresholds chosen a priori). However as stated in section I, in the context of misclassification costs, decision thresholds can be adaptively defined to minimize the loss of a learned model. Hence, two instances considered equally easy by a classifier with a fixed threshold can be rather difficult under different decision thresholds and costs. In this work, we derived similarity measures for algorithm performance in cost sensitive scenarios, by considering different methods to choose decision thresholds of classifiers based on the knowledge about misclassification costs.

III. Notation and Basic Definitions

The basic definitions adopted in our work are mostly based on [6]. Instances can be classified into one of the classes $Y = \{0, 1\}$, in which 0 is the positive class and 1 is the negative class. A learned model m is a scoring function that receives an instance x as input and returns a score $s = m(x)$ that indicates the chance of a negative class prediction. A model can be transformed in a crisp classifier assuming a decision threshold t in such a way that if $s \leq t$ then x is classified as positive, and it is classified as negative if $s > t$.

The errors of a classifier are associated to costs related to the classes. The cost of a *false negative* is represented as c_0 and the cost of a *false positive* in turn is represented as c_1. As in [6], we normalize the costs by setting $c_0 + c_1 = b$ and adopt the *cost proportion* $c = c_0/b$ to represent the operating condition faced by a model this deployment. For simplicity, we adopted $b = 2$ and hence $c \in [0, 1]$, $c_0 = 2c$ and $c_1 = 2(1-c)$.

The loss function produced assuming a decision threshold t and a cost proportion c is defined as:

$$\begin{aligned} Q(t,c) &= c_0\pi_0 FN(t) + c_1\pi_1 FP(t) \\ &= 2\{c\pi_0 FN(t) + (1-c)\pi_1 FP(t)\} \end{aligned} \tag{1}$$

In the above equation $FN(t)$ and $FP(t)$ are respectively the *False Negative* and *False Positive* rates produced by a model when a threshold t is adopted. The variables π_0 and π_1 represent the proportion of positive and negative examples.

The Positive Rate $R(t)$ is the proportion of instances predicted as positive at the decision threshold t and can be expressed as $\pi_0(1 - FN(t)) + \pi_1 FP(t)$.

IV. Instance Hardness and Cost Curves

In Eq. 1, the loss produced by a classifier is an aggregation of the errors observed for the positive and the negative instances. A positive instance will be associated to a cost $2c$ when it is incorrectly classified. An error for a negative instance in turn will be associated to a cost $2(1-c)$.

In our work, we decompose Eq. 1 to derive the loss functions for individual instances. The individual loss for a positive instance x is defined as:

$$QI(x,t,c) = 2cFN(x,t) \tag{2}$$

In the above equation, $FN(x,t) = 1$ if x is a false negative when threshold t is adopted and $FN(x,t) = 0$ otherwise. A similar definition of loss function can be done for a negative instance x:

$$QI(x,t,c) = 2(1-c)FP(x,t) \tag{3}$$

In the above equation, $FP(x,t) = 1$ is x is a false positive at threshold t and $FP(x,t) = 0$ otherwise.

Given a threshold choice method, $QI(x,t,c)$ produce a specific curve for the example x along the range of operating conditions ($c \in [0,1]$). The *instance hardness* measures in our work are defined as the area under the individual cost curves and compute the total expected loss for the range of operation conditions. In the general case, given a threshold choice method $t = T(c)$, the hardness of an instance is:

$$IH^T(x) = \int_0^1 QI(x, T(c), c)w(c)dc \tag{4}$$

In the above equation, $w(c)$ represents the distribution of c. In our work, we derived the instance hardness measures for different threshold choice methods assuming uniform distribution of cost proportions.

A. Score-Driven Instance Hardness

In the score-driven method [7], the decision threshold t is defined by simply setting it equal to the operating condition c:

$$T(c) = c \tag{5}$$

For instance, if $c = 0.7$, the cost of false negatives are higher than the costs of false positives. In this case by setting $t = c = 0.7$ the produced classifier tends to predict more instances as positive, thus minimizing the relative number of false negatives. According to [7], the score-driven method is a natural choice when the model scores are assumed to be class probability estimators and the scores are well calibrated.

Under the score-driven method, we can derive the loss function for positive instances as follows:

$$FN(x,c) = \begin{cases} 1, & \text{if } s > c \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

Replacing $FN(x,c)$ in Eq. 2 we have the following score-driven cost curve for a positive instance:

$$QI(x,t,c) = \begin{cases} 2c, & \text{if } s > c \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

Fig. 1(a) illustrates the score-driven cost curve for a positive instance with score s. For $s \leq c$ no cost is assumed; on the other hand, the cost varies linearly. The area under the cost curve is defined as an instance hardness measure:

$$IH^{sd}(x) = \int_0^s 2c\,\mathrm{d}c = \Big[c^2\Big]_0^s = s^2 \tag{8}$$

Since $y = 0$ for positive instances, the above measure can be replaced by $(y-s)^2$, which correspond to the squared-error obtained by the model.

Eq. 9 and 10 define the score-driven cost curve for negative instances. Figure 1(c) illustrates this curve when the negative instance has a score s.

$$FP(x,c) = \begin{cases} 1, & \text{if } s \leq c \\ 0, & \text{otherwise} \end{cases} \tag{9}$$

$$QI(x,t,c) = \begin{cases} 2(1-c), & \text{if } s \leq c \\ 0, & \text{otherwise} \end{cases} \tag{10}$$

Similar to the positive instance, instance hardness for a negative instance is defined as the area under the cost curve:

$$IH^{sd}(x) = \int_s^1 2(1-c)\,\mathrm{d}c = \Big[2c - c^2\Big]_s^1 = (1-s)^2 \tag{11}$$

For negative instances we have $y = 1$ and then the above measure corresponds to $(y-s)^2$. Hence, for both positive and negative instances, hardness is defined as the squared-error obtained by the model.

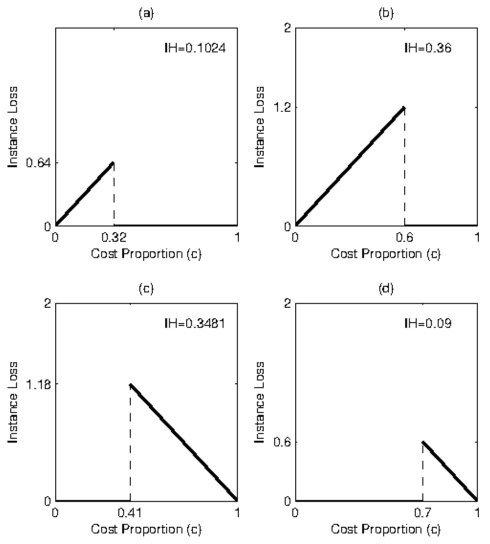

Fig. 1. Instance cost curves for an instance assuming the rate-driven and score-driven methods.

B. Rate-Driven Instance Hardness

According to [8], the score-driven method is sensitive to the estimation of the scores. For instance, if the scores are highly concentrated, a small change in the operating condition (and consequently in the decision threshold) may drastically affect the classifier performance. As an alternative, in [8] the authors proposed to use the proportion of instances predicted as positive (i.e., $R(t)$) to define the decision thresholds.

In the rate-driven method, the decision threshold is set to achieve a desired proportion of positive predictions. The threshold choice method is defined as:

$$T(c) = R^{-1}(c) \tag{12}$$

For instance, if $c = 0.7$ the threshold t is set in such a way that 70% of the instances are classified as positive. The operating condition c is then expressed as the desired positive rate: $c = R(t)$. The probability of a false negative under the rate-driven choice method can be defined as:

$$FN(x, R^{-1}(c)) = \begin{cases} 1, & \text{if } s > R^{-1}(c) \\ 0, & \text{otherwise} \end{cases} \tag{13}$$

Replacing $FN(x, R^{-1}(c))$ in Eq. 2 we have the following rate-driven cost curve for a positive instance:

$$QI(x, t, c) = \begin{cases} 2c, & \text{if } s > R^{-1}(c) \\ 0, & \text{otherwise} \end{cases} \tag{14}$$

Fig. 1(b) illustrates the rate-driven cost curve for a positive instance with a score s. For $s \leq R^{-1}(c)$, or alternatively for $R(s) \leq c$, no cost is assumed. When $R(s) > c$, the cost of the instance varies linearly. The area under the rate-driven cost can be adopted as an instance hardness measure:

$$IH^{rd}(x) = \int_0^{R(s)} 2c \, \mathrm{d}c = \Big[c^2\Big]_0^{R(s)} = R(s)^2 \tag{15}$$

The above measure states that the instance hardness is related to position of the instance in the ranking produced by the learned model. The worse is the ranking of the positive instance, the higher is the instance hardness.

Eq. 16 and 17 define the rate-driven cost curve for negative instances with score s, illustrated in Figure 1(d).

$$FP(x, R^{-1}(c)) = \begin{cases} 1, & \text{if } s \leq R^{-1}(c) \\ 0, & \text{otherwise} \end{cases} \tag{16}$$

$$QI(x, t, c) = \begin{cases} 2(1-c), & \text{if } s \leq R^{-1}(c) \\ 0, & \text{otherwise} \end{cases} \tag{17}$$

Similar to the positive instance, instance hardness for a negative instance is defined as the area under the cost curve:

$$IH^{rd}(x) = \int_{R(s)}^{1} 2(1-c) \, \mathrm{d}c = \Big[2c - c^2\Big]_{R(s)^1} = (1 - R(s))^2 \tag{18}$$

Notice that $(1 - R(s))$ corresponds to the negative rate of a classifier at the point s. The instance hardness for negative instances is then related to the ranking of the most likely negative instances produced by the learned model.

Different from the score-driven method, which measures the magnitude of the errors obtained by a model, the rate-driven method is more related to ranking performance. By adopting the score-driven method, an instance is considered hard if its score is not well calibrated. On other hand, the same instance may be easy by assuming the rate-driven method if it is well ranked relative to the other instances. Instance hardness by adopting the score-driven method only depends on the score of the instance. Instance hardness by adopting the rate-driven method in turn depends not only on the instance score but also on how the other instances are ordered.

V. Cost Sensitive Algorithm Similarity

The application of global metrics fails to properly measure algorithm similarity since it is based on average performance, loosing important information during the evaluation process. More fine-grained measures provide a solution for this limitation by verifying the algorithm performance at the instance level. However, the proposed measures are not well defined when misclassification costs are involved.

In this work, we derived different measures for algorithm similarity based on the concept of instance hardness. Given a pair of learned models, they will be similar if the hardness values computed on a test set by using the two models are similar. More specifically, in order to relate two models, we initially collect the scores produced by them on a test set of instances. Following, we compute the hardness value for each test instance considering each model. Finally, the similarity between the models is defined by aggregating the instance hardness values as follows:

$$\mathcal{D}(m_a, m_b, D) = \frac{1}{|D|} \sum_{x \in D} |IH^T_{m_a}(x) - IH^T_{m_b}(x)| \tag{19}$$

The above equation measures the pairwise distance between models m_a and m_b for each instance x belonging to the test set D. In our work, in order to deal with costs, we derived in the previous section two distinct instance hardness by adopting different threshold choice methods T. By adopting the score-driven method, two models will be similar with their scores are similar (the squared-errors produced by the models are similar at instance level). By adopting the rate-driven method in turn two models will be similar if the test instances are similarly ranked. The two methods correspond to two different evaluation strategies. Other instance hardness measures and their corresponding algorithm similarity measures can be developed in the future by considering other threshold choice methods, such as the probabilistic ones [6].

Once we have the algorithm similarity measure on a single dataset, we can compute the overall similarity over different datasets in order to achieve a wider view about the relative performance of algorithms. There are many strategies to do this aggregation, such as the use of the median or the average similarity as well as other consensus similarity methods [11] that can be taken over all datasets involved. In this work, we adopt a simple aggregation strategy by computing the average dissimilarities measured over all datasets observed:

$$\mathcal{A}(m_a, m_b) = \frac{1}{N} \sum_{j=1}^{N} \mathcal{D}(m_a, m_b, D_j) \tag{20}$$

In the above equation, N stands for the number of datasets available for measuring algorithm similarity. As it will be seen next section, this measure will be adopted in a case study to cluster algorithms based on average similarity over a set of learning tasks.

VI. Experimental Setup

In order to achieve a good diversity of problems, we computed the instance hardness on a group of 50 datasets[2] representing problems of binary classification. Most problems were collected from the UCI repository.

We compare the performance of 6 algorithms available on the Scikit-Learn Project[3]. The scores for each instance were calculated using a 10-fold cross-validation. Usually, the classification algorithms return the label predicted for an informed example but to produce real-values scores (varying between 0 and 1)for the input instances, we adopted specific procedures for each algorithm. For the Nearest Neighbors (KNN), the score returned is the number of neighbors belonging to the negative class divided by K. The procedure used for the Random Forest (RF) is similar: we count the number of votes for the negative class and divide this by the number of trees in the forest. For the Decision Tree (DT), Naive Bayes (NB) and Logistic Regression (LR) the score represents the probability of the instance being negative. For the Support Vector Machine (SVM), we get the decision function output and normalize it between 0 and 1.

In order to have a reference point, we compare variations of algorithm at the clustering process. For the KNN, we applied the algorithm with 3 and 5 nearest neighbors (3NN and 5NN) and for the SVM, we adopted the experiments with the linear and RBF kernel functions(SVM_LIN and SVM_RBF, respectively). We expect that the models learned with variations of a same algorithm will be clustered together. In order to cluster the results obtained by the similarity measures, we applied the agglomerative hierarchical clustering method with the average linkage function[12].

VII. Dataset Algorithm Similarity

After computing the scores for all datasets instances, we use the results to calculate the instance hardness measures for the two threshold choice methods presented in section IV. Then we measure the pairwise distance among the models for each dataset using the equation 19. In order to illustrate the use of the proposed measures, we initially presented the results obtained by a single dataset (the Ecoli dataset). Next section will present the clustering results obtained by considering all the collected datasets.

TABLE I. Mean Instance Hardness Measures for Ecoli Dataset

Model	$\overline{IH^{rd}}$	$\overline{IH^{sd}}$
3NN	0,269	0,0694
5NN	0,2583	0,0583
DT	0,2858	0,0774
LR	0,252	0,0671
NB	0,2585	0,1949
RF	0,256	0,051
SVM_LIN	0,2554	0,1828
SVM_RBF	0,2553	0,1811

[2]The list of datasets used is available at http://www.cin.ufpe.br/∼cecm2

[3]Available at http://scikit-learn.org/stable/

Table I presents the average instance hardness measures for the Ecoli dataset. The average hardness values observed suggest that the models are better at calibrating scores than at ranking instances for this dataset. In fact, the score-driven instance hardness measures are in general smaller than the rate-driven ones. Also, large differences in the quality of scores produced by some algorithms do not reflect in large differences in the ranking quality. For instance, although the NB produced the worst result for the score-driven method, its results for the rate-driven method are similar to the other algorithms.

Figures 2 and 3 present the dendrograms derived from the clustering process by adopting the proposed measures for the Ecoli dataset. The cluster formed by the SVM models is observed in both cases. The remaining clusters have a slight difference. In the score-driven method the NB model is a cluster by itself, but in the rate-driven case NB was considered similar to LR (i.e., they produced similar rankings of instances although their scores are different). In both cases, the KNN and the RF models were considered similar, which can be supported in [13].

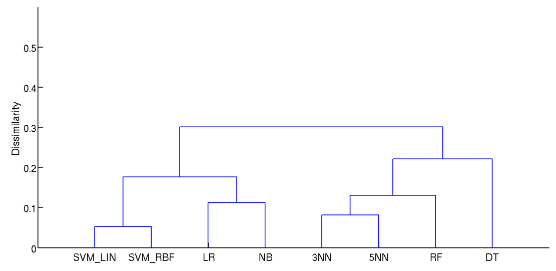

Fig. 2. Clustered score-driven performance for the models learned on the Ecoli dataset

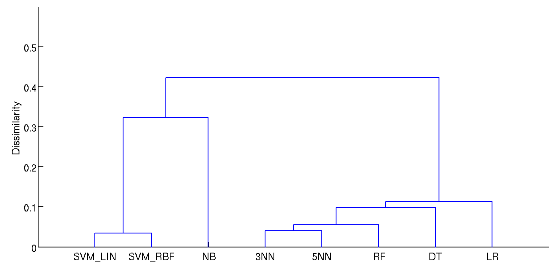

Fig. 3. Clustered rate-driven performance for the models learned on the Ecoli dataset

VIII. Aggregated Algorithm Similarity

In order to have a better view about the algorithms similarities, we applied the equation 20 to compute the average distance between algorithms across the 50 datasets adopted in our experiments. Figure 4 presents the dendrogram of algorithms considering the score-driven measure. This dendrogram shows that the models learned by KNN were the most similar, as expected. The second most similar pair of models was produced by the RF and the LR algorithms (with a low dissimilarity measure around 0.1). Depending on the cut-point (e.g., 0.3) adopted to produce clusters from the dendrogram, the DT algorithm is clustered together with 3NN, 5NN, RF and LR. The models obtained by NB are the ones that present the most distinct behaviour. Finally, the SVM models are similar between them (relative to the other algorithms), but their similarity level is not so high. The change in the kernel function produced in many datasets very distinct models. Some of the results derived from the dendrogram are expected (such as the similarity between 3NN and 5NN). Other results in turn, such as the similarity between RF and LR, were not expected and hence have to be better investigated in the future.

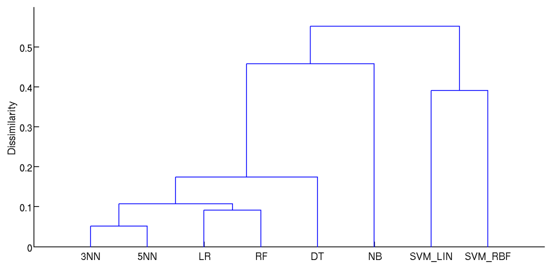

Fig. 4. Clustered average score-driven performance

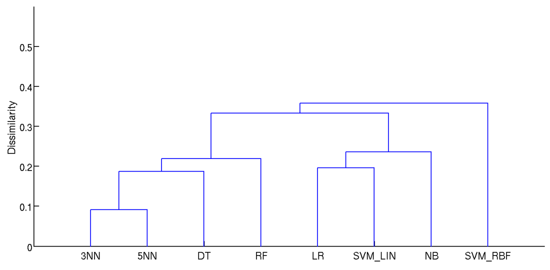

Fig. 5. Clustered average rate-driven performance

Figure 5 displays the dendrogram of algorithms produced by adopting the rate-driven measure. The algorithms are more similar to each other when similarity is measured in terms of ranking quality. As in the score-driven dendrogram, the 3NN, 5NN, DT and RF are clustered together. The algorithms LR, SVM_LIN and NB formed another cluster. Different from the score-driven dendrogram, we see that the SVM models do not belong to the same group. Again, a deeper investigation has been to be done to provide further explanations on why a given pair of algorithms was similar. In our work, we provide a general framework that can be extended with new algorithms and datasets and can be used to identify which algorithms are similar under different cost-sensitive perspectives.

IX. Conclusion

In this work, we proposed similarity measures between algorithms by considering two threshold choice methods: rate-driven and score-driven. For each method, we proposed a corresponding instance hardness measure that was then deployed to define the algorithm similarity measure. The similarity between algorithms can be quite different depending on the dataset and the cost-sensitive scenario being tackled. For the score-driven method, two algorithms are similar if they produce similar scores. For the rate-driven method, in turn, two algorithms are similar if they produce similar rankings of instances.

In order to infer overall similarity on different datasets, we computed the average similarity between algorithms over 50 datasets and performed clustering of algorithms. The results revealed some unexpected similarities that can serve as starting points for further investigations to discover and explain hidden relationships between algorithms and learning strategies.

As future work, the ideas presented here can be applied on a predictive meta-learning strategy to select algorithms for new datasets depending on the threshold choice method and the input costs. In our work, we aggregate similarity across datasets using the mean method, which is simple but possibly naive. Other consensus similarity methods can be investigated. Finally, we highlight that our work is very limited concerning the number of algorithms and datasets adopted. Hence, more extensive meta-learning studies can be done in the future, with stronger conclusions and insights.

Acknowledgement

This research is supported by CNPq, CAPES and FACEPE (Brazilian agencies).

References

[1] P. Brazdil, C. Giraud-Carrier, C. Soares, and R. Vilalta, *Metalearning: Applications to Data Mining*, 1st ed. Springer Publishing Company, Incorporated, 2008.

[2] T. M. Michael R. Smith and C. Giraud-Carrier, "An instance level analysis of data complexity," *Machine Learning*, 2013.

[3] R. Leite, P. Brazdil, and J. Vanschoren, "Selecting classification algorithms with active testing," in *Machine Learning and Data Mining in Pattern Recognition*. Springer, 2012, pp. 117–131.

[4] A. Kalousis, J. Gama, and M. Hilario, "On data and algorithms: Understanding inductive performance," *Machine Learning*, pp. 275–312, 2004.

[5] J. Lee and C. Giraud-Carrier, "A metric for unsupervised metalearning," *Intelligent Data Analysis*, vol. 15, pp. 827–841, 2011.

[6] J. Hernández-Orallo, P. Flach, and C. Ferri, "A unified view of performance metrics: Translating threshold choice into expected classification loss," *J. Mach. Learn. Res.*, vol. 13, no. 1, pp. 2813–2869, Oct. 2012.

[7] ——, "Brier curves: A new cost-based visualisation of classifier performance," in *Proceedings of the 28th International Conference on Machine Learning*, 2011.

[8] ——, "ROC curves in cost space," *Machine Learning*, vol. 93, no. 1, pp. 71–91, Feb. 2013.

[9] J. Fürnkranz and J. Petrak, "An evaluation of landmarking variants," in *Working Notes of the ECML/PKDD 2000 Workshop on Integrating Aspects of Data Mining, Decision Support and Meta-Learning*, 2001, pp. 57–68.

[10] A. Peterson and T. Martinez, "Estimating the potential for combining learning models," *Proceedings of the ICML workshop on meta-learning*, 2005.

[11] F. de A.T. de Carvalho, Y. Lechevallier, and F. M. de Melo, "Relational partitioning fuzzy clustering algorithms based on multiple dissimilarity matrices," *Fuzzy Sets and Systems*, vol. 215, no. 0, pp. 1 – 28, 2013, theme : Clustering.

[12] C. Ding and X. He, "Cluster merging and splitting in hierarchical clustering algorithms," *Data Mining, 2002. ICDM 2003. Proceedings. . . .* , pp. 139–146, 2002.

[13] Y. Lin and Y. Jeon, "Random forests and adaptive nearest neighbors," *Journal of the American Statistical Association*, vol. 101, no. 474, pp. 578–590, 2006.