Research Article

A Semisupervised Cascade Classification Algorithm

Stamatis Karlos,¹ Nikos Fazakis,² Sotiris Kotsiantis,¹ and Kyriakos Sgarbas²

¹*Department of Mathematics, University of Patras, 26504 Rio, Greece*

²*Department of Electrical and Computer Engineering, University of Patras, 26504 Rio, Greece*

Correspondence should be addressed to Stamatis Karlos; stkarlos@upatras.gr

Received 30 November 2015; Accepted 27 January 2016

Academic Editor: Samuel Huang

Copyright © 2016 Stamatis Karlos et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Classification is one of the most important tasks of data mining techniques, which have been adopted by several modern applications. The shortage of enough labeled data in the majority of these applications has shifted the interest towards using semisupervised methods. Under such schemes, the use of collected unlabeled data combined with a clearly smaller set of labeled examples leads to similar or even better classification accuracy against supervised algorithms, which use labeled examples exclusively during the training phase. A novel approach for increasing semisupervised classification using Cascade Classifier technique is presented in this paper. The main characteristic of Cascade Classifier strategy is the use of a base classifier for increasing the feature space by adding either the predicted class or the probability class distribution of the initial data. The classifier of the second level is supplied with the new dataset and extracts the decision for each instance. In this work, a self-trained NBVC4.5 classifier algorithm is presented, which combines the characteristics of Naive Bayes as a base classifier and the speed of C4.5 for final classification. We performed an in-depth comparison with other well-known semisupervised classification methods on standard benchmark datasets and we finally reached to the point that the presented technique has better accuracy in most cases.

1. Introduction

Pattern recognition and data mining from massively collected datasets have attracted extensive research the last decades. More and more recommendations systems from many scientific domains demand high quality information for providing accurate predictions. The most common procedure that is applied in such cases is the classification task. The aim of classification is to build a model, according to the base classifier that has been preferred each time, for matching each tested instance into predefined and distinct classes with the highest possible accuracy under any time restrictions. The nature of the classes depends on the scientific domain that each examined problem belongs to. Moreover, the number of the classes affects the expected classification accuracy and also determines how explanatory the prediction can be. For example, a simplified problem could include some data for deciding if a patient suffers from one disease or not. This problem can be characterized as a binary problem, since the unique two classes are “Yes” and “No.” On the other hand, in a more in-depth analysis problem, the classes could represent all the different diseases that a patient may suffer and the

prediction should be in this case based on more extensive and clearly more time-consuming procedures. The two important queries that arise from examples like the previous one are firstly how the classification task can be accurate when much available data do not exist and secondly how the processing time could be eliminated in order to be deemed acceptable in relation to the current computational resources.

The traditional division of classification theory contains the supervised and the unsupervised approaches. According to the former, all the available categorized data are used for building an appropriate classification model. Consequently, provided an initial dataset (D) is given along with a well-defined vector of classes c , which can be described as $c = \{\text{class}_1, \text{class}_2, \dots, \text{class}_k\}$ in the generalized case of k distinct classes, supervised methods build a mapping (M) for assigning the most probable class to each instance of D . Another variant of the same algorithms is the computation of conditional distribution probability vectors, whose dimension is $(c \times 1)$ and each value depicts the probability of i th instance for belonging to the j th class $P(\text{target_class} = \text{class}_j \mid \text{instance}_i)$ with $1 \leq j \leq k$. The latter approach is determined by the lack of knowledge of neither the categorization of the provided

data nor the number of the classes. On the contrary, these algorithms try to generate useful information from the available data usually using some stochastic assumptions for acquiring some initial information. The conditions under such algorithms operate and explain the inferior performance in respect to the supervised learning algorithms. However, a new group of methods have been suggested recently [1], which are commonly called semisupervised learning (SSL) algorithms and are considered as a mixture of both supervised and unsupervised algorithms. The merging of the properties from the original approaches and the generation of new semisupervised algorithms are still a field which has attracted the interest of many researchers. However, the common property of all these approaches is the search for an effective combination of few labeled data with much more unlabeled ones for increasing both their accuracy rate and learning ability. Moreover, the improved accuracy performance of these algorithms against their ancestors which has been noticed along many known scientific fields has highlighted semisupervised techniques as a great tool to the machine learning community.

The attribute which defines the quantitative relation between labeled and unlabeled data is called labeled ratio and is computed as the fraction of the number of labeled examples divided by the number of all the examples. The abbreviation of labeled ratio would be R in the following text. The dependence between parameter R and the performance of different algorithms is examined in depth in Section 4.

Taking into account that many practical applications and machine learning concepts suffer from practical difficulties on collecting enough reliable data in relation to the unlabeled examples, which in domains like Web mining [2], Speaker Recognition, and Object Detection [3] are plenty, the values of R tend to be small. Judging by the fact that in the most of these domains either there are not any automated procedures for labeling correctly new examples or the cost for such procedures is not always affordable, the solution of semisupervised concept seems to meet the current needs of real-life scenarios [4].

Triguero et al. [5] introduced a taxonomy of self-labeled techniques for semisupervised learning, having made an extensive study about the applicability of these methods to classification concepts. Some of the properties that were used for establishing this taxonomy are the influence of the number of labeled examples and the time requirements of each algorithm. Both of these play a cardinal role in real-life problems. Another generic review of semisupervised learning algorithms has been made by Schwenker and Trentin recently [6]. Concerning the classification concept, they refer to four different categories of algorithms whose behavior is based on one of the following methods:

- (i) incremental techniques,
- (ii) generative models,
- (iii) support vector machines,
- (iv) graphs.

Schwenker and Trentin [6] introduced the terms of partially or weakly supervised learning and indicated some promising challenges of this scientific domain.

The largest family of algorithms for learning tasks, such as classification, is the ensemble learning schemes. The basic idea behind this category is the combination of multiple classifiers for improving the average behavior and the robustness of any of the involved single classifiers. Another benefit is the avoiding of lengthy tuning procedures or retuning of a single classifier for each different dataset. The two different architectures that ensemble classifiers can meet are the sequential or cascade architecture and the parallel. Voting, which uses Bayesian learning theory, and Stacking, which learns from predictions of level-0 classifiers via a single metaclassifier, are the most common schemes. Cascade Generalization was proposed in 1997 by Gama and Bradzil [7] and it follows the theory of Stacked Generalization. According to this, provided a dataset which is described by a raw feature vector of dimensions $(f \times 1)$, we could define t levels with $t \geq 2$. In each level, one or more chosen classifiers examine the dataset independently and output their prediction. These predictions are added to the initial dataset and are passed to the next level. This chain of classifiers is symbolized as follows:

$$\begin{aligned} \text{Cascade Classifier} &= (\text{Classifier_Level}_1) \\ &\nabla (\text{Classifier_Level}_2) \nabla \dots \nabla (\text{Classifier_Level}_t), \quad (1) \\ &t \geq 2. \end{aligned}$$

As a result, the classifier of the last level is supplied with an enlarged dataset and its output is clearly guided and affected by the lower levels decisions. This strategy is called incremental batch learning. Two different schemes were suggested in this work and are compared with many different algorithms. They also present some explanatory examples and speak about the syntactic formalism of the new feature spaces that are generated. It is important to mention that Cascade Classifiers theory has been used only for supervised concepts [8–10] until now.

The aim of our work was to present a self-trained Cascade Classifier algorithm and compare it with other well-known semisupervised classification methods on standard benchmark datasets. Since time requirements for real-life applications are considered and respecting the increased complexity of complex cascade chains, we selected the minimum amount of levels for completing a Cascade Classifier. At the same time we combined a Bayesian (NB) [11] and a decision tree classifier (C4.5) [12] for inducing fast response under an ensemble scheme. Besides the examination of classification accuracy rate, we performed statistical comparisons of the proposed method (self-trained (NBVC4.5)) with other algorithms and represented an illustrative visualization for recording the average accuracy of each algorithm against the others for different labeled ratios. Our developed technique presented better accuracy in most cases and a better overall performance in different scenarios, rendering this algorithm as a robust tool. For the proposed classifier, its specific chain is depicted in Figure 1.

The rest of this work is organized as follows. In Section 2, a brief description of the semisupervised classification techniques is provided. In Section 3, a presentation of the proposed algorithm is taking place. In Section 4, there are the

FIGURE 1: Cascading NB and C4.5 (NBVC4.5).

results of the comparison of the proposed algorithm with other well-known semisupervised classification methods on standard benchmark datasets. Finally, some conclusion remarks and future research examples are presented in Section 5. In the Appendix, there is the corresponding link for a tool that implements our proposed algorithm.

2. Semisupervised Techniques

Many variants of self-labeled techniques have been proposed over the last years, because many practical problems can be simulated under the scenario that both labeled and unlabeled data exist [13]. Any choice of the labeled ratio is allowed and this leads to large enough number of possible experiments or simulations. The only restriction is the limited number of labeled examples that are provided in some scientific fields. Even worse, the available data may not describe some or all the views of a problem sufficiently. For example, in a noise retrieval problem, the detection of the noisy segments should be described both in time and in frequency domain. But, the absence of labeled data either in one domain or in both domains would affect the performance of the self-labeled method. Some methods, such as boosting, tackle these problems but do not manage always to affect the learned hypotheses, especially in extreme situations. The most promising strategy for eliminating such phenomena has been proposed by Triguero et al. [14]. According to this work, the generation of new synthetic labeled data seems capable of fulfilling the labeled data distribution.

Having assured that there are a few labeled examples, implementation of self-labeled methods has shown promising results. In particular, the information that is exploited by the unlabeled data when they are combined with the labeled subset has been proved adequate for modifying the learned hypothesis obtained in comparison with the

supervised scenario. Having chosen the labeled ratio, all the available data (D) split into two distinct subsets: the Labeled (L) and the Unlabeled (U) set, with $D = L \cup U$. The generic representation of the examples that are included in each of these subsets is $x_j = \{(\text{Feature set}) \mid \text{class}\}$. When j equals L , then the class is known and these examples contribute to training phase. On the other hand, for the case when j equal U , the class is totally unknown. Depending on the theory of each self-labeled method, these subsets interact with various ways. There could be a movement of examples from U to L , or an iterative reweighting of the L examples for classifying, with higher accuracy, the examples of U , or even the generation of multiple subsets of L for classifying the U 's examples.

The most well-known method of semisupervised techniques is self-training method. It is usually called a wrapper method and its simplicity has attracted many researchers from different domains. Its main asset is the shortage of restrictions about the labeled data that are given. This happens because it acts with the assumption that in each iteration the information that is extracted is correct and sufficient to lead itself to better results. The whole function of this scheme as it concerns the semisupervised classification (SSC) task can be separated to five different steps [3]. First of all, there is an initial step during which all the stopping criteria are described and are set for the rest of the procedure. Also the accuracy threshold (AccT) for the accepted examples is set. After this, the original dataset (D) is split into L and U subsets, as it has been mentioned previously. Thirdly, a classifier of user's choice is chosen and is trained with the examples of L subset, which have been chosen randomly from D . During the fourth step, the classification of unlabeled examples takes place and then a procedure of assessment follows. More specifically, each point that scored a probability value over or equal to the specific threshold ($P(U_i) \geq \text{AccT}$) is considered informative enough for enhancing the learning ability of the algorithm

in the next training phases. At the final step, all these high-valued examples are subtracted from subset U and are inserted to the initial training set (L), increasing in this way its cardinality under the assumption that the new examples are also correctly labeled. These five steps compose a complete step of the simplified self-training scheme. Retraining of the classifier is done using the new enlarged training set until stopping criteria are satisfied. Although self-training has met great success in many practical problems, observation of fluctuations to its performance is still often compared with similar supervised algorithms. A good explanation is the fact that, during the training phase of the former algorithms, some of the unlabeled examples will not get labeled, since the termination of the algorithm will have been preceded [6]. This fact means that a part of the total information provided through the dataset will not be exploited under this scheme.

Another critical point for which self-training scheme has to be assessed is the principle of high confidence that governs itself and may lead to the opposite results [15]. Suppose that the quality of the initial labeled data which will be provided by a human-expert or by a specialized mining tool is poor; the final prediction will also be distorted, not providing a totally correct set of labeled data. For this reason, some statistical tests or other criteria have been added so as to reassure that the self-training scheme will continue to perform well, forming in this way new algorithms. An algorithm which derives from the primary self-training scheme is the Self-Training with Editing (SETRED) [16]. The prominent modification is the establishing of a restriction related to the acceptance or the rejection of the examples that the algorithm evaluates as trustworthy. The origin of this restriction stems from the graph theory. Therefore, a neighborhood graph in f -dimensional feature space is being built, whereas f is the dimension of the feature row vector. All the candidate examples for being appended to the initial training set are being filtered through a hypothesis test and only these that passed successfully that test are finally added to the L subset before the end of each iteration. The generic performance of SETRED over many datasets has certified the need of intermediate filtering stages into self-labeled methods.

Cotraining method is another popular generalized scheme under which a family of semisupervised algorithms has been formed utilizing more than one view of any problem that needs to be mined for any learning task. Sun [17] reviews theories in order to describe the characteristics of multiview learning. The basic idea that is examined is the chance for enhancing the classification accuracy by obtaining examples from multiple sources which are characterized from different feature vectors. Two of the theories that are under scrutiny by Sun are the Canonical Correlation Analysis (CCA), first proposed by Hotelling et al. [18, 19], for the case of two views and the effectiveness of cotraining method. The integration of multiple views and their editing is also being reviewed by Xu et al. [19] who are trying to organize the different strategies that have been proposed about multiview learning. Another equally important issue that is discussed is the potentiality of constructing multiple views and their evaluation.

Cotraining was introduced by Blum and Mitchell [20]. In contrast with self-training method that respects single-view

learning and demands one feature vector independently of its physical consistency, cotraining method requires two distinct feature vectors in which each one represents a different view [13, 14, 17, 21]. Similar to self-training theory, for each different view, a weak-classifier is trained over the corresponding training set. The aim is again to select the appropriate examples that can be considered reliable enough for the tested case and enlarge the initial training set with them. All this procedure is executed alternately for maximizing the mutual agreement on the two feature vectors. The stopping criteria also are slightly modified for capturing the existence of the two classifiers. Even though this assumption seems to be more effective when these two subsets of features are obtained from different views with some clear natural meaning, Nigam and Ghani [22] showed experimentally that cotraining scheme can perform well enough even if the different views come from random splits of an original feature vector. Moreover, the concatenation of many feature sets and their applications to single-view methods can cause overfitting phenomena, especially in cases in which the initial dataset is characterized by low labeled ratios.

The power of cotraining method as a function of the size of the initial training set has been examined by Didaci et al. [23]. The recordings of their study showed that cotraining managed to accomplish high quality results even in cases where the algorithm was provided with very few examples per class. This does not mean that there are not any weak examples of cotraining scheme. Du et al. [24], after having executed a big trial of experiments, reached to the point that running semisupervised algorithms based on small labeled training sets cannot ensure the accuracy of multiview consideration assumptions. A lot of variants of basic cotraining scheme have been developed trying to improve cotraining performance. Furthermore, since the hazard of false acceptance of unlabeled examples still remains possible and may deteriorate the classification accuracy, Multiple Kernel Learning (MKL) and Subspace learning-based theory have been used to filter the examples for higher reliability [19]. One approach according to the latter theory was proposed by Sun and Jin [25] who trust only the examples that satisfy both the high confidence of the used classifier and the CCA's restrictions for being finally appended to the initial training set. Another algorithm with great success over many domains is the Democratic-Co [26] that also follows the multiview theory but from another aspect. Instead of asking for more than one view of the data, it uses multiple algorithms for producing the necessary information and adopts a voted majority process for the final decision.

Ensemble classifiers or committees of classifiers can also be used under semisupervised scheme for exploiting the power of more than one weak learner [27]. The key point in this situation is the diversity of the included classifiers. Many artificial tactics have been presented for injecting diversity on a group of classifiers when the original diversity does not reach the expected levels. Bagging is maybe the most popular strategy for achieving such results. In this case, each base classifier is produced by random sample of the initial training set. A great research on the field of ensemble classifiers' properties has been made by Kuncheva and her

partners over the last fifteen years [28, 29]. Jiang et al. [30] introduced a hybrid method which weights the participation of two different classifiers for pushing the total accuracy against the atomic behavior of each participant classifier. It is worth mentioning that the combined classifiers were Naive Bayes (NB) and Support Vector Machine (SVM) which are a generative classifier and a discriminative classifier, respectively. A representative approach that uses ensemble theory is the TriTraining scheme that does not require any redundant view for being applied [31]. It is based on the decisions of three classifiers that classify each tested instance according to majority voting. The training procedure of each classifier is taking place over a different sampled subset of L . An enhanced variant of TriTraining scenario is the improved TriTraining algorithm (im-tri-training) [32]. Specific weaknesses of the default scheme, such as unsuitable error estimation, are tackled and eliminated for achieving a more robust behavior.

Following similar theories, Li and Zhou [15] developed CoForest algorithm, in which a number of Random Trees are trained on bootstrap data from the dataset. This technique has inherited the asset of ensemble methods about the robustness even if the number of the available labeled examples is reduced. One serious reason why this behavior is generalized over multiple datasets is the utilization of Random Tree classifier for random samples of the collected labeled data. Majority voting extracts the final prediction. ADE-CoForest [33] originated from the previous algorithm. Its powerful asset is an embedded editing technique for preventing misclassified examples to impair its learning ability. The solution of distance metrics has also been applied with cotraining scheme and one of the most well-defined products was the cotraining by committee which has been proposed by Hady and Schwenker [34]. Three ensemble methods were used (Co-Bag, CoAdaBoost, and CoRSM) for testing these metrics. This method does not reclaim the multiview concept and remains a single-view method.

Wang et al. [35] developed Rasco algorithm (Random Subspace Method for co-training) which follows the ensemble theory and tries to produce diversity among its base classifiers. According to this algorithm, a random split of the feature vector is suitable for training a group of different learners. After having completed the training of each base classifier on a randomly chosen subset of the initial feature vector, the enlargement of the training set by the unlabeled examples along with their class assignment is filtered from a number of decisions of the base learners. An extension of the original Rasco theory is Rel-Rasco [36]. The elimination of the inaccurate produced learners by Rasco method motivated Yaslan and Cataltepe to suggest an algorithm which produces relevant random subspaces and then does semisupervised ensemble learning using those subspaces together with unlabeled data.

3. Proposed Algorithm

Cascade Generalization may be regarded as a special case of Stacking Generalization mainly due to the layered learning

structure. Some aspects that make Cascade Generalization are the following:

- (i) All classifiers have access to the original attributes. Any new attribute built at lower layers is considered exactly in the same way as any of the original attributes. The new attributes are categorical by using the predicted class or continuous by taking the form of a probability class distribution.
- (ii) The goal of Cascade Generalization is to obtain a model that can use terms in the representation language of lower level classifiers.

Cascading classifiers is particularly useful for models that have highly combinatorial or counting rules (e.g., class 1 if exactly two features are negative, class 2 otherwise), which cannot be fitted without looking at all the interaction terms. Having cascading classifiers enables the successive stage to gradually approximate the combinatorial nature of the classification or to add interaction terms in classification algorithms that cannot express them in one stage.

In our case for the Cascade Generalization, the new attributes are first derived from the class prediction given by the Naive Bayes (NB) learner. This constructive step extends the representational language for the high level learner—C4.5. The presented ensemble can be symbolized as NBVC4.5 and is described by pseudocode in Algorithm 1.

In this work, we propose a self-training method that uses the power of cascade ensemble for semisupervised tasks. The proposed algorithm (self-trained (NBVC4.5)) is presented in Algorithm 2. The self-training process produces good results by using the more accurate class probabilities of NBVC4.5 model for the unlabeled examples.

Concerning which data examples are removed from U and added to L , the explanation is that if the probability of the most probable class exceeds the predefined threshold T , then this instance is assigned a label. In the proposed algorithm, experimental results that were performed by the authors showed that a good option for the threshold parameter is the value of 0.9, which gave decent results irrespective of the data set, as far as the classification accuracy is concerned. It was noticed that only a small amount of examples per class in each iteration meets the restriction above. For the implementation, we used the open-source environments of WEKA [37] and KEEL [5].

4. Experiments

The experiments are based on standard classification datasets taken from the KEEL-dataset repository [38] covering a wide range of scientific fields. These datasets have been partitioned using the 10-fold cross-validation procedure. For each generated fold, a given algorithm is trained with the examples contained in the rest of the folds (training partition) and then tested with the current fold. Each training partition is divided into two parts: labeled and unlabeled examples. In order to study the influence of the amount of labeled data, we examined four different ratios for dividing the training set: 10%, 20%, 30%, and 40%.

Algorithm *CascadeGeneralization*($L1, L2, Dataset$)

begin

 $Dataset' = \text{ExtendDataset}(Dataset, L1, Dataset)$ $M2 = \text{model induced by } L2 \text{ from } Dataset'$ For each new query instance q $q' = \text{ExtendDataset}(Dataset, L1, \{q\})$ $\text{Class}(q) = M2(q')$

end

where:

(i) $L1$ is the base level learning algorithm (NB Classifier in all the training set e.g calculate $P(c)$ for each class c , and returns the class with the max probability)(ii) $L2$ is the meta-level learning algorithm: C4.5**Algorithm** *ExtendDataset*(D, L, D')

begin

 $M = \text{model induced by } L \text{ from } D$ $newD = \emptyset$ For each $e = (x, y) \in D'$ For $j = 1$ to C $p_j = \text{probability that } e \text{ belongs to } c_j \text{ according to } M$ $e' = (x, \text{predicted class of } M: c_{\text{index}(\max(p_j))})$ $newD = newD \cup \{e'\}$ Return $newD$

end

where:

(i) L is a learning algorithm, C is the number of classes(ii) D is the dataset from which L builds a model, D' is a dataset to which the model learnt from D may be applied

ALGORITHM 1: Cascading NB and C4.5 (NBVC4.5).

Algorithm *Self-trained* (NBVC4.5)**Input:**

NBVC4.5 – Cascading NB and C4.5, as base classifier

 D – Initial training dataset L – initial labeled examples, $L \subseteq D$ U – initial unlabeled examples, $U \subseteq D$ x_{MCP} – Examples with Most Confident Predictions

AccT – Threshold of acceptance

MaxIter – number of maximum iterations performed

(1) Initialization:Train NBVC4.5 as base model on L **(2) Loop for a number of iterations (MaxIter is equal to 40 for our implementation)**Use NBVC4.5 classifier to select the examples with Most Confident Predictions per iteration (x_{MCP})Remove x_{MCP} from U and add them to L In each iteration a few examples per class are removed from U and added to L Re-train NBVC4.5 as base model on new enlarged L **Output:**Use NBVC4.5 trained on L to predict class labels of the test cases.

ALGORITHM 2: The self-trained (NBVC4.5) algorithm.

Subsequently, we compared the proposed method with other state-of-the-art algorithms into the KEEL tool [38] such as self-training (C45) [3], self-training (NB) [5], SETRED [16], co-training (C45) [13], Democratic-Co [26], TriTraining (C45) [31], TriTraining (NB) [5], DE-TriTraining (C45) [39], DE-TriTraining (NB) [5], CoForest [15], Rasco (C45) [35], Rasco (NB) [35], Rel-Rasco (C4.5) [36], Rel-Rasco (NB) [36],

Co-Bagging (C45) [34], Co-Bagging (NB) [5], and ADE-CoForest [33]. For all tested algorithms, the default parameters of KEEL were used. The classification accuracy of each tested algorithm using 10%, 20%, 30%, and 40% as labeled ratio is presented in Tables 1, 2, 3, and 4, respectively. The best accuracy value among the different algorithms tested in each experiment is shown in bold style. For our experiments, we

TABLE 1: Classification accuracy ($R = 10\%$).

Datasets	Algorithms						
	Self (NB/C45)	Self (C45)	CoTrain (C45)	Democ-Co	SETRED	TriTrain (C45)	TriTrain (NB)
appendicitis	0.8218	0.8318	0.8318	0.8218	0.7373	0.8045	0.8018
automobile	0.4174	0.4052	0.3730	0.3601	0.4388	0.3889	0.3101
banana	0.8549	0.8479	0.8481	0.8417	0.8638	0.8481	0.6147
breast	0.7365	0.7216	0.6774	0.7287	0.6835	0.7216	0.7365
bupa	0.5910	0.5392	0.5739	0.5104	0.5339	0.5742	0.5331
chess	0.9409	0.9543	0.9515	0.9199	0.8104	0.9578	0.8407
cleveland	0.5431	0.5248	0.5736	0.5223	0.5294	0.4761	0.5563
coil2000	0.9321	0.9373	0.9403	0.9322	0.8926	0.9357	0.9222
crx	0.8483	0.8660	0.8159	0.8495	0.8111	0.8555	0.7789
dermatology	0.8847	0.8562	0.8429	0.8760	0.9182	0.8816	0.5595
flare	0.7402	0.7214	0.5742	0.7214	0.6445	0.7158	0.7411
glass	0.5604	0.4847	0.4504	0.4868	0.5402	0.4921	0.4435
haberman	0.7057	0.7054	0.7190	0.7156	0.6211	0.7088	0.7447
heart	0.7593	0.6778	0.7000	0.8000	0.7444	0.7148	0.7741
housevotes	0.9191	0.9414	0.8225	0.8899	0.9129	0.9158	0.8914
iris	0.8733	0.8400	0.8467	0.9133	0.9133	0.7267	0.8133
led7digit	0.6240	0.6140	0.5140	0.6160	0.6180	0.6040	0.5760
lymphography	0.7819	0.6312	0.5726	0.4901	0.6833	0.6118	0.4724
magic	0.8210	0.8217	0.8204	0.7842	0.7840	0.8245	0.7269
mammographic	0.7964	0.8025	0.8074	0.7963	0.7580	0.8183	0.7491
movement_libras	0.1944	0.2556	0.2361	0.1972	0.4444	0.2750	0.1889
mushroom	0.9922	0.9966	0.9968	0.9927	0.9945	0.9955	0.9301
nursery	0.9092	0.9064	0.9034	0.8951	0.8101	0.9039	0.8941
page-blocks	0.9497	0.9523	0.9492	0.9077	0.9359	0.9561	0.8560
penbased	0.9172	0.8916	0.8957	0.9474	0.9778	0.9027	0.8536
phoneme	0.7937	0.7770	0.7652	0.7874	0.8046	0.7770	0.7526
pima	0.6656	0.6643	0.6704	0.6967	0.6565	0.6564	0.7007
ring	0.9761	0.8396	0.8366	0.8741	0.6691	0.8542	0.9797
satimage	0.8104	0.8045	0.8056	0.8462	0.8570	0.8224	0.7953
segment	0.9004	0.8900	0.9017	0.9026	0.9065	0.9000	0.7909
spambase	0.8688	0.8669	0.8884	0.8777	0.8281	0.8810	0.8360
spectfheart	0.7788	0.6819	0.7235	0.7379	0.7201	0.7574	0.7604
splice	0.9282	0.8266	0.8310	0.8978	0.6997	0.8254	0.9185
thyroid	0.9921	0.9922	0.9924	0.9393	0.9090	0.9918	0.9267
tic-tac-toe	0.7046	0.7108	0.6931	0.6900	0.7255	0.7088	0.6619
titanic	0.7747	0.7751	0.7783	0.7756	0.6402	0.7765	0.7701
twonorm	0.9692	0.8136	0.8085	0.9645	0.9358	0.8616	0.9766
wine	0.9327	0.7405	0.8075	0.9493	0.9438	0.8203	0.8980
wisconsin	0.9422	0.9093	0.9064	0.9650	0.9478	0.9312	0.9358
zoo	0.7444	0.6794	0.6356	0.9314	0.9347	0.7192	0.9131
							0.5694
							0.8692
							0.7456
							0.9197
							0.8255
							0.7866
							0.9284
							0.7837
							0.6535
							0.9254
							0.9009
							0.8248
							0.7571
							0.8951
							0.8569
							0.7541
							0.8205
							0.7929
							0.9160
							0.8533
							0.9049
							0.7889
							0.7772
							0.6533
							0.6342
							0.8582
							0.9793
							0.7929
							0.7541
							0.8569
							0.7900
							0.8248
							0.9009
							0.9254
							0.6535
							0.7837
							0.6919
							0.8370
							0.9774
							0.8597
							0.7866
							0.9284
							0.7456

TABLE 2: Classification accuracy ($R = 20\%$).

Datasets	Algorithms									
	Self (NBVC45)	Self (C45)	Cotrain (C45)	Democ-Co	SETRED	TriTrain (C45)	TriTrain (NB)	De-TriTrain (C45)	Co-Bag (C45)	Co-Bag (NB)
appendicitis	0.8509	0.8509	0.8500	0.8600	0.8100	0.8318	0.8682	0.8409	0.8318	0.8318
automobile	0.4739	0.5158	0.4803	0.4744	0.4951	0.4938	0.4496	0.4120	0.5355	0.4412
banana	0.8706	0.8804	0.8743	0.8794	0.8691	0.8736	0.6125	0.8774	0.8804	0.6123
breast	0.7291	0.7156	0.7227	0.7184	0.6788	0.7225	0.7399	0.7475	0.7224	0.7469
bupa	0.5334	0.6055	0.6055	0.5504	0.5655	0.6199	0.5414	0.5589	0.6026	0.5510
chess	0.9621	0.9778	0.9759	0.9393	0.8489	0.9781	0.8611	0.9333	0.9778	0.8501
cleveland	0.5826	0.4929	0.5270	0.5530	0.5257	0.5399	0.5904	0.5128	0.5535	0.5705
coil2000	0.9395	0.9400	0.9402	0.9330	0.8994	0.9371	0.9167	0.9366	0.9338	0.9158
crx	0.8372	0.8571	0.8498	0.8587	0.8142	0.8540	0.7957	0.8555	0.8510	0.7927
dermatology	0.9382	0.9100	0.8790	0.9300	0.9467	0.9214	0.7618	0.8739	0.9186	0.7981
flare	0.7354	0.7280	0.6897	0.7402	0.6660	0.7270	0.7392	0.7082	0.7176	0.7439
glass	0.5481	0.5085	0.5348	0.4799	0.5935	0.6104	0.4671	0.5036	0.5631	0.4861
haberman	0.7122	0.7089	0.7353	0.7222	0.6600	0.7089	0.7285	0.7322	0.7348	0.7285
heart	0.8074	0.7519	0.7556	0.8222	0.7741	0.7926	0.8407	0.7556	0.7481	0.8185
housevotes	0.9586	0.9586	0.9363	0.9023	0.8957	0.9586	0.8839	0.9099	0.9528	0.8895
iris	0.9267	0.8933	0.8933	0.9533	0.9200	0.8800	0.9467	0.9600	0.8667	0.9467
led7digit	0.6480	0.6780	0.6440	0.6620	0.6280	0.6760	0.6440	0.6860	0.6580	0.6220
lymphography	0.7966	0.7055	0.7750	0.4612	0.7392	0.7545	0.5091	0.7461	0.7474	0.7529
magic	0.8378	0.8304	0.8320	0.8017	0.7934	0.8335	0.7281	0.8250	0.8409	0.7262
mammographic	0.8228	0.8229	0.8228	0.8084	0.7473	0.8157	0.7843	0.8087	0.8248	0.7819
movement_libras	0.3861	0.3472	0.3917	0.3667	0.6000	0.3833	0.3944	0.2889	0.3556	0.3389
mushroom	0.9966	0.9991	0.9991	0.9980	0.9984	0.9989	0.9441	0.9970	0.9989	0.9439
nursery	0.9340	0.9235	0.9260	0.9108	0.8327	0.9258	0.8971	0.9072	0.9235	0.8934
page-blocks	0.9585	0.9602	0.9609	0.9115	0.9448	0.9611	0.8942	0.9454	0.9613	0.9123
penbased	0.9370	0.9241	0.9247	0.9632	0.9871	0.9287	0.8512	0.9293	0.9370	0.8506
phoneme	0.8120	0.7839	0.8024	0.8055	0.8344	0.7977	0.7507	0.8029	0.8244	0.7478
pima	0.6825	0.6810	0.6874	0.7319	0.6369	0.6939	0.7176	0.6849	0.7084	0.7254
ring	0.9778	0.8658	0.8630	0.8969	0.7007	0.8795	0.9788	0.7142	0.8918	0.9795
satimage	0.8317	0.8238	0.8261	0.8612	0.8740	0.8353	0.7983	0.8318	0.8410	0.7961
segment	0.9242	0.9264	0.9294	0.9307	0.9338	0.9247	0.7939	0.9139	0.9277	0.7835
spambase	0.9071	0.8912	0.8908	0.8941	0.8529	0.8941	0.8279	0.8695	0.8960	0.8303
spectfheart	0.7272	0.7192	0.7611	0.7305	0.7085	0.7491	0.7268	0.7426	0.7833	0.7231
splice	0.9361	0.8834	0.8793	0.9119	0.6978	0.8843	0.9417	0.8292	0.8875	0.9276
thyroid	0.9933	0.9935	0.9932	0.9419	0.9149	0.9939	0.9264	0.9371	0.9942	0.9229
tic-tac-toe	0.7380	0.7568	0.7213	0.7329	0.7621	0.7505	0.7004	0.6931	0.7286	0.6900
titanic	0.7837	0.7824	0.7824	0.7797	0.6407	0.7815	0.7706	0.6915	0.7828	0.7701
twonorm	0.9720	0.8165	0.8276	0.9707	0.9411	0.8674	0.9765	0.8318	0.8741	0.9769
wine	0.9209	0.8369	0.7863	0.9542	0.9154	0.8422	0.9601	0.8931	0.8255	0.9379
wisconsin	0.9534	0.9343	0.9343	0.9637	0.9463	0.9253	0.9608	0.9460	0.9360	0.9490
zoo	0.8236	0.8311	0.7344	0.8900	0.9372	0.8264	0.9022	0.7869	0.8231	0.9106

TABLE 3: Classification accuracy ($R = 30\%$).

Datasets	Self (NBVC45)	Self (C45)	Cotrain (C45)	Democ-Co	SETRED	Algorithms				
						TriTrain (C45)	TriTrain (NB)	De-TriTrain (C45)	Co-Bag (C45)	Co-Bag (NB)
appendicitis	0.8300	0.8409	0.8409	0.8691	0.8109	0.8327	0.8591	0.8091	0.8318	0.8400
automobile	0.5232	0.5484	0.5710	0.5432	0.5515	0.5707	0.5071	0.4500	0.5737	0.5402
banana	0.8819	0.8819	0.8785	0.8728	0.8700	0.8806	0.6130	0.8834	0.8811	0.6102
breast	0.7437	0.7179	0.7079	0.7219	0.6495	0.7000	0.7470	0.7304	0.7111	0.7293
bupa	0.6076	0.5842	0.6137	0.5988	0.5848	0.5850	0.5326	0.5774	0.6159	0.5241
chess	0.9706	0.9856	0.9840	0.9596	0.8648	0.9843	0.8717	0.9634	0.9819	0.8652
cleveland	0.5841	0.5364	0.5467	0.5700	0.5566	0.5362	0.5703	0.5490	0.5599	0.5632
coil2000	0.9400	0.9384	0.9400	0.9320	0.8992	0.9387	0.9135	0.9383	0.9317	0.9169
crx	0.8528	0.8526	0.8570	0.8482	0.8067	0.8571	0.7808	0.8570	0.8615	0.7850
dermatology	0.9550	0.9209	0.9098	0.9180	0.9380	0.9240	0.8397	0.9047	0.9184	0.8878
flare	0.7420	0.7299	0.7224	0.7485	0.6594	0.7298	0.7392	0.7176	0.7308	0.7430
glass	0.5350	0.5437	0.5746	0.5079	0.5933	0.5687	0.5072	0.5465	0.5635	0.4823
haberman	0.7154	0.7153	0.7288	0.7447	0.6761	0.7089	0.7545	0.7157	0.7282	0.7515
heart	0.7926	0.7556	0.7704	0.8296	0.7852	0.7556	0.8296	0.7704	0.7704	0.8407
housevotes	0.9475	0.9475	0.9531	0.9037	0.9078	0.9531	0.8914	0.9112	0.9531	0.9063
iris	0.9267	0.9200	0.9133	0.9533	0.9267	0.9133	0.9467	0.9400	0.9067	0.9400
led7digit	0.6660	0.6880	0.6520	0.6820	0.6360	0.6760	0.6820	0.6760	0.6820	0.6800
lymphography	0.7752	0.7587	0.7444	0.7442	0.7439	0.7315	0.6082	0.7331	0.7445	0.6724
magic	0.8349	0.8380	0.8385	0.8016	0.7950	0.8411	0.7283	0.8283	0.8456	0.7280
mammographic	0.8497	0.8438	0.8437	0.8300	0.7620	0.8425	0.8108	0.8125	0.8352	0.8084
movement libras	0.5139	0.4083	0.4639	0.4917	0.6917	0.4889	0.5056	0.3611	0.4250	0.4611
mushroom	0.9986	0.9996	0.9996	0.9995	0.9991	0.9995	0.9467	0.9995	0.9991	0.9453
nursery	0.9484	0.9377	0.9363	0.9212	0.8357	0.9362	0.9004	0.9228	0.9382	0.8964
page-blocks	0.9622	0.9635	0.9618	0.9289	0.9461	0.9635	0.8918	0.9541	0.9622	0.9002
penbased	0.9495	0.9409	0.9391	0.9729	0.9901	0.9431	0.8543	0.9385	0.9485	0.8543
phoneme	0.8225	0.8137	0.8218	0.8029	0.8470	0.8249	0.7581	0.8177	0.8275	0.7506
pima	0.6809	0.7252	0.7123	0.7305	0.6694	0.7045	0.7463	0.7318	0.7085	0.7242
ring	0.9762	0.8754	0.8784	0.9089	0.7104	0.8881	0.9793	0.7326	0.8943	0.9796
satimage	0.8437	0.8270	0.8407	0.8693	0.8862	0.8438	0.7969	0.8437	0.8528	0.7950
segment	0.9346	0.9303	0.9385	0.9416	0.9411	0.9381	0.8022	0.9359	0.9329	0.7957
spambase	0.9139	0.8999	0.9052	0.9052	0.8695	0.9073	0.8284	0.8801	0.9117	0.8308
spectfheart	0.7339	0.7496	0.7538	0.7121	0.7127	0.7644	0.6930	0.7085	0.7608	0.7083
splice	0.9514	0.9166	0.9176	0.9188	0.7154	0.9157	0.9505	0.8721	0.9201	0.9455
thyroid	0.9944	0.9946	0.9947	0.9521	0.9183	0.9944	0.9317	0.9435	0.9939	0.9310
tic-tac-toe	0.7713	0.7610	0.7495	0.7630	0.7923	0.7641	0.7025	0.7224	0.7745	0.6973
titanic	0.7792	0.7787	0.7783	0.7792	0.6407	0.7783	0.7733	0.6928	0.7797	0.7733
twonorm	0.9730	0.8255	0.8323	0.9701	0.9439	0.8743	0.9772	0.8458	0.8778	0.9776
wine	0.9660	0.8415	0.8255	0.9660	0.9275	0.9039	0.9778	0.8987	0.8817	0.9833
wisconsin	0.9579	0.9443	0.9474	0.9666	0.9535	0.9502	0.9637	0.9387	0.9431	0.9577
zoo	0.8381	0.8231	0.8256	0.9133	0.9331	0.7781	0.8931	0.8008	0.7731	0.8711

TABLE 4: Classification accuracy ($R = 40\%$).

Datasets	Self (NBVC45)	Self (C45)	Co-Train (C45)	Democ-Co	SETRED	Algorithms				De-TriTrain (C45)	Co-Bag (C45)	Co-Bag (NB)
						TriTrain (C45)	TriTrain (NB)	De-TriTrain (NB)	TriTrain (C45)			
appendicitis	0.8491	0.8018	0.8000	0.8591	0.7564	0.7718	0.8309	0.8309	0.8673	0.8382	0.8109	
automobile	0.5703	0.5760	0.6168	0.5537	0.6089	0.6151	0.5404	0.5404	0.4836	0.6009	0.5265	
banana	0.8808	0.8828	0.8785	0.8811	0.8649	0.8825	0.6140	0.6140	0.8847	0.8836	0.6142	
breast	0.7254	0.7046	0.7082	0.7257	0.6632	0.7009	0.7543	0.7543	0.6977	0.7289	0.7508	
bupa	0.6283	0.6546	0.6002	0.6035	0.5874	0.6570	0.5391	0.5391	0.6132	0.6180	0.5482	
chess	0.9828	0.9853	0.9856	0.9712	0.8795	0.9872	0.8724	0.8724	0.9700	0.9872	0.8664	
cleveland	0.5656	0.5093	0.5364	0.5290	0.5586	0.5289	0.5261	0.5261	0.5216	0.5627	0.5662	
coil2000	0.9400	0.9401	0.9395	0.9327	0.8982	0.9399	0.9104	0.9104	0.9385	0.9354	0.9133	
crx	0.8542	0.8465	0.8556	0.8424	0.8161	0.8527	0.7822	0.7822	0.8571	0.8587	0.7898	
dermatology	0.9661	0.9385	0.9159	0.9439	0.9466	0.9353	0.8739	0.8739	0.8990	0.9267	0.8853	
flare	0.7326	0.7280	0.7072	0.7420	0.6679	0.7270	0.7420	0.7420	0.7223	0.7223	0.7420	
glass	0.5965	0.5722	0.6038	0.5535	0.6487	0.6718	0.4633	0.4633	0.5828	0.6272	0.4606	
haberman	0.7218	0.7186	0.7385	0.7449	0.6924	0.7218	0.7448	0.7448	0.7253	0.7449	0.7514	
heart	0.8000	0.7667	0.7407	0.8259	0.7741	0.7741	0.8407	0.8407	0.7444	0.7926	0.8296	
housevotes	0.9586	0.9586	0.9586	0.9164	0.9117	0.9591	0.9060	0.9060	0.9250	0.9702	0.8986	
iris	0.9067	0.9000	0.9000	0.9733	0.9333	0.9067	0.9333	0.9333	0.9400	0.9067	0.9467	
led7digit	0.6840	0.6840	0.6740	0.6820	0.6380	0.6740	0.6840	0.6840	0.6600	0.6880	0.6720	
lymphography	0.8100	0.7725	0.7653	0.6496	0.7639	0.7382	0.3010	0.3010	0.6764	0.7630	0.4695	
magic	0.8400	0.8416	0.8405	0.8111	0.7996	0.8409	0.7277	0.7277	0.8321	0.8532	0.7275	
mammographic	0.8354	0.8452	0.8440	0.8267	0.7486	0.8367	0.8168	0.8168	0.8038	0.8352	0.8146	
movement_libras	0.5639	0.4889	0.4611	0.6222	0.7500	0.5194	0.6139	0.6139	0.4556	0.4889	0.5667	
mushroom	0.9987	1.0000	1.0000	0.9998	0.9998	0.9995	0.9504	0.9504	1.0000	0.9993	0.9488	
nursery	0.9590	0.9442	0.9435	0.9307	0.8341	0.9439	0.8996	0.8996	0.9304	0.9433	0.8978	
page-blocks	0.9691	0.9677	0.9688	0.9371	0.9455	0.9673	0.8936	0.8936	0.9587	0.9667	0.9101	
penbased	0.9567	0.9456	0.9451	0.9761	0.9904	0.9478	0.8569	0.8569	0.9452	0.9567	0.8564	
phoneme	0.8307	0.8233	0.8261	0.8142	0.8560	0.8211	0.7600	0.7600	0.8364	0.8444	0.7570	
pima	0.7280	0.7292	0.7030	0.7527	0.6980	0.7318	0.7436	0.7436	0.7200	0.7330	0.7384	
ring	0.9770	0.8727	0.8846	0.9177	0.7174	0.8953	0.9800	0.9800	0.7442	0.9095	0.9795	
satimage	0.8457	0.8448	0.8483	0.8738	0.8940	0.8545	0.7975	0.7975	0.8446	0.8252	0.7958	
segment	0.9364	0.9385	0.9398	0.9424	0.9446	0.9437	0.7944	0.7944	0.9407	0.9160	0.7926	
spambase	0.9213	0.9056	0.9119	0.9154	0.8780	0.9089	0.8268	0.8268	0.8888	0.8951	0.8319	
spectfheart	0.7681	0.7752	0.7489	0.6972	0.7500	0.7604	0.6890	0.6890	0.7604	0.7370	0.7003	
splice	0.9524	0.9266	0.9276	0.9273	0.7201	0.9301	0.9520	0.9520	0.8987	0.8248	0.9492	
thyroid	0.9940	0.9940	0.9949	0.9526	0.9231	0.9944	0.9346	0.9346	0.9438	0.9901	0.9267	
tic-tac-toe	0.8069	0.7829	0.7797	0.7901	0.7975	0.8006	0.7067	0.7067	0.7599	0.7035	0.6973	
titanic	0.7878	0.7874	0.7856	0.7787	0.6407	0.7842	0.7733	0.7733	0.6951	0.7837	0.7733	
twonorm	0.9739	0.8269	0.8358	0.9722	0.9454	0.8788	0.9780	0.9780	0.8408	0.8597	0.9782	
wine	0.9663	0.8866	0.8987	0.9605	0.9441	0.8755	0.9601	0.9601	0.9157	0.7866	0.9431	
wisconsin	0.9593	0.9606	0.9417	0.9695	0.9623	0.9474	0.9593	0.9593	0.9535	0.9284	0.9578	
zoo	0.8908	0.8767	0.8875	0.9117	0.9397	0.9075	0.8997	0.8997	0.8556	0.7456	0.8817

FIGURE 2: Comparison of average accuracy on benchmark datasets.

used 40 datasets and all the above 19 algorithms, including self-NBVC4.5.

Here, we present only the best 10 of these algorithms, according to their classification accuracy. We also provide a more representative visualization of the average accuracy ability of the proposed algorithm in comparison with the remaining 18 algorithms, presented in Figure 2. In this figure, we have mapped each different ratio of labeled examples with different color and line format across a radar plot.

The illustration above depicts the association between the accuracy of the involved algorithms and the different labeled ratio's values. A noteworthy conclusion is that the increase of R (%) does not uniquely mean that the average accuracy of any algorithm will finally be increased. For instance, both versions of Rasco and Rel-Rasco did not manage to score a better accuracy between the 10% and 20% provided labeled examples. Similar irregularities are found in TriTraining (NB) and Co-Bagging (NB), where the choice of $R = 30\%$ caused better results than those of 40%. All these saturation phenomena can be avoided by visualizations like that in Figure 2. The full tables of comparisons can be found in supplemental excel file (see Supplementary Material available online at <http://dx.doi.org/10.1155/2016/5919717>).

A statistical comparison of the tested algorithms has been also applied to all the selected values of R . In order to perform

this task, Friedman test together with two similar post hoc statistical tests (Holm/Hochberg) described in [40] has been chosen. Concerning Friedman test, this is a nonparametric equivalent of the repeated-measures ANOVA. It produces a ranking of the algorithms for each dataset separately and compares the average ranks of the algorithms. The null hypothesis states that all the algorithms are equivalent, getting the same average ranking over any dataset. Holm procedure (1979) [41] was used in Demsar (2006) [42] for comparisons of multiple classifiers involving a control method. Hochberg Test (1988) [43] has more power than Holm's procedure, but the difference between them is not very notable when considering all pairwise comparisons. The results are presented in Tables 5 and 6.

Regarding the post hoc test, if the sorted p -values increase rapidly enough, the procedures by Holm and Hochberg give identical answers. Judging by the produced results and, more especially, by the average accuracy on benchmark datasets, as Figure 2 depicts, and the ranking of the proposed algorithm in Friedman tests, as presented in Tables 5 and 6, the proposed algorithm gives better results among all the tested algorithms. This is due to better probability-based ranking and higher classification accuracy that are induced by the series combination of Naive Bayes and C4.5 classifiers. In this point we have to mention that the same algorithm was examined

TABLE 5: Friedman ranking of the algorithms in all tested labeled ratios.

Algorithm	$R = 10\%$	$R = 20\%$	$R = 30\%$	$R = 40\%$
Self-NBVC4.5	5.85	6.225	6.2125	5.9375
Co-Bagging (C45)	7.35	6.625	7.5375	8.4375
Democratic-Co	7.55	6.7375	6.975	7.125
TriTraining (C45)	7.6	6.9875	8.1875	7.575
CoForest	7.9375	8.5	8.625	8.65
Self-training (C45)	8.2125	7.9375	8.575	8.425
Cotraining (C45)	8.9625	7.725	8.0875	8.7875
SETRED	9.35	10.125	10.6375	10.275
DE-TriTraining (C45)	9.8	9.225	10.5375	10.65
ADE-CoForest	10.125	10.775	11.025	10.5375
DE-TriTraining (NB)	10.425	10.2875	11.975	12
TriTraining (NB)	10.55	9.4875	9.8375	10.3875
Cotraining (NB)	10.9875	10.2125	11.375	11.1
Rel-Rasco (NB)	11.7375	13.3625	11.8125	12.6125
Co-Bagging (NB)	11.75	10.0625	10.6625	10.975
Rasco (NB)	12.5	14.575	11.725	12.175
Rasco (C45)	12.7125	14.4875	10.8125	10.7125
Rel-Rasco (C45)	12.95	14	11.775	10.4125
Self-training (NB)	13.65	12.6625	13.625	13.225

TABLE 6: Rankings of the algorithms in all tested labeled ratios according to Holm/Hochberg ($\alpha = 0.05$).

Algorithm	$R = 10\%$	$R = 20\%$	$R = 30\%$	$R = 40\%$
Co-Bagging (C45)	0.2332	0.7506	0.2923	0.0469
Democratic-Co	0.1767	0.6838	0.5445	0.3453
TriTraining (C45)	0.1643	0.5445	0.1165	0.1931
Cotraining (C45)	0.0971	0.0706	0.0552	0.0311
Self-training (C45)	0.0604	0.1735	0.0604	0.0481
CoForest	0.0134	0.2332	0.1362	0.0235
DE-TriTraining (C45)	0.0054	0.0019	0.0004	0.0006
TriTraining (NB)	0.0017	0.0171	0.0006	0.0002
Co-Bagging (NB)	0.0007	0.0003	0.0001	0.0003
SETRED	0.0003	0.0012	<0.0001	<0.0001
Cotraining (NB)	0.0002	0.0095	0.0040	0.0004
DE-TriTraining (NB)	<0.0001	0.0015	<0.0001	<0.0001
ADE-CoForest	<0.0001	0.0000	<0.0001	<0.0001
Self-training (NB)	<0.0001	0.0023	0.0004	0.0001
Rel-Rasco (NB)	<0.0001	<0.0001	<0.0001	<0.0001
Rel-Rasco (C45)	<0.0001	<0.0001	0.0003	0.0001
Rasco (C45)	<0.0001	<0.0001	<0.0001	0.0004
Rasco (NB)	<0.0001	<0.0001	<0.0001	<0.0001

with the modification of the Level-1 classifier. To be more specific, we changed the output stage of Naive Bayes classifier so as to append the distribution vector to the feature vector of original dataset. The results show that there was a small deterioration concerning the average ability of the final algorithm.

5. Conclusions

It is promising to implement techniques that use both labeled and unlabeled examples in classification tasks. The shortage of available labeled data impairs the accuracy of learning process, since supervised learning methods cannot produce a learner with worthy accuracy.

The strategy of combining classifiers using a cascade methodology seems to improve the average behavior of the final ensemble classifier under self-training scheme. In this work, a self-trained (NBVC4.5) algorithm has been proposed. We performed a comparison with other well-known semisupervised learning methods on standard benchmark datasets and the presented technique had better accuracy in most of the tested datasets. Due to the encouraging results obtained from these experiments, one can expect that the proposed technique can be applied to real classification tasks giving slightly better accuracy than the traditional semisupervised approaches.

The fact that there is no previous use of Cascade Classifiers under self-labeled techniques means that a new family of algorithms can be developed for achieving enhanced learning effectiveness. The choice of the classifiers in each step as well as the number of levels (t) should be selected with respect to the time restrictions that are usually imposed by real-life problems.

In spite of these results, no general method will work always. The main drawback of the semisupervised schemes is the needed time in the training phase. Some techniques that could enhance this property by saving both valuable operation time and computational resources are the feature selection algorithms that search for a subset of relevant features by removing the less informative features of the initial set [44].

Appendix

A java software tool implementing the proposed algorithm and some basic run instructions can be found at <http://www.math.upatras.gr/~sotos/SemiCascade-Experiment.zip>.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

- [1] X. Zhu and A. B. Goldberg, *Introduction to Semi-Supervised Learning*, Synthesis Lectures on Artificial Intelligence and Machine Learning, Morgan & Claypool, San Rafael, Calif, USA, 2009.
- [2] M. K. Dalal and M. A. Zaveri, "Semisupervised learning based opinion summarization and classification for online product reviews," *Applied Computational Intelligence and Soft Computing*, vol. 2013, Article ID 910706, 8 pages, 2013.
- [3] C. Rosenberg, M. Hebert, and H. Schneiderman, "Semi-supervised self-training of object detection models," in *Proceedings of the 7th IEEE Workshop on Applications of Computer Vision (WACV '05)*, pp. 29–36, Breckenridge, Colo, USA, January 2005.

- [4] C. Liu and P. C. Yuen, "A boosted co-training algorithm for human action recognition," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 21, no. 9, pp. 1203–1213, 2011.
- [5] I. Triguero, S. García, and F. Herrera, "Self-labeled techniques for semi-supervised learning: taxonomy, software and empirical study," *Knowledge and Information Systems*, vol. 42, no. 2, pp. 245–284, 2015.
- [6] F. Schwenker and E. Trentin, "Pattern classification and clustering: a review of partially supervised learning approaches," *Pattern Recognition Letters*, vol. 37, no. 1, pp. 4–14, 2014.
- [7] J. Gama and P. Brazdil, "Cascade generalization," *Machine Learning*, vol. 41, no. 3, pp. 315–343, 2000.
- [8] D. Ferraretti, G. Gamberoni, and E. Lamma, "Unsupervised and supervised learning in cascade for petroleum geology," *Expert Systems with Applications*, vol. 39, no. 10, pp. 9504–9514, 2012.
- [9] W.-C. Cheng and D.-M. Jhan, "A self-constructing cascade classifier with AdaBoost and SVM for pedestrian detection," *Engineering Applications of Artificial Intelligence*, vol. 26, no. 3, pp. 1016–1028, 2013.
- [10] P. Zhang, T. D. Bui, and C. Y. Suen, "A novel cascade ensemble classifier system with a high recognition performance on hand-written digits," *Pattern Recognition*, vol. 40, no. 12, pp. 3415–3429, 2007.
- [11] P. Domingos and M. Pazzani, "On the optimality of the simple Bayesian classifier under zero-one loss," *Machine Learning*, vol. 29, no. 2–3, pp. 103–130, 1997.
- [12] J. R. Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufmann, San Francisco, Calif, USA, 1993.
- [13] O. Chapelle, B. Schölkopf, and A. Zien, *Semi-Supervised Learning*, The MIT Press, Cambridge, Mass, USA, 2006.
- [14] I. Triguero, S. Garcia, and F. Herrera, "SEG-SSC: a framework based on synthetic examples generation for self-labeled semi-supervised classification," *IEEE Transactions on Cybernetics*, vol. 45, no. 4, pp. 622–634, 2015.
- [15] M. Li and Z.-H. Zhou, "Improve computer-aided diagnosis with machine learning techniques using undiagnosed samples," *IEEE Transactions on Systems, Man, and Cybernetics Part A: Systems and Humans*, vol. 37, no. 6, pp. 1088–1098, 2007.
- [16] M. Li and Z.-H. Zhou, "SETRED: self-training with editing," in *Advances in Knowledge Discovery and Data Mining: 9th Pacific-Asia Conference, PAKDD 2005, Hanoi, Vietnam, May 18–20, 2005. Proceedings*, vol. 3518 of *Lecture Notes in Computer Science*, pp. 611–621, Springer, Berlin, Germany, 2005.
- [17] S. Sun, "A survey of multi-view machine learning," *Neural Computing and Applications*, vol. 23, no. 7–8, pp. 2031–2038, 2013.
- [18] H. Hotelling, "Relations between two sets of variates," *Biometrika*, vol. 28, no. 3–4, pp. 321–377, 1936.
- [19] C. Xu, D. Tao, and C. Xu, "A survey on multi-view learning," <http://arxiv.org/abs/1304.5634>.
- [20] A. Blum and T. Mitchell, "Combining labeled and unlabeled data with co-training," in *Proceedings of the 11th Annual Conference on Computational Learning Theory (COLT '98)*, pp. 92–100, Morgan Kaufmann, Madison, Wis, USA, July 1998.
- [21] S. Sun and Q. Zhang, "Multiple-view multiple-learner semi-supervised learning," *Neural Processing Letters*, vol. 34, no. 3, pp. 229–240, 2011.
- [22] K. Nigam and R. Ghani, "Analyzing the effectiveness and applicability of co-training," in *Proceedings of the 9th International Conference on Information and Knowledge Management (CIKM '00)*, pp. 86–93, ACM, McLean, Pa, USA, November 2000.
- [23] L. Didaci, G. Fumera, and F. Roli, "Analysis of co-training algorithm with very small training sets," in *Structural, Syntactic, and Statistical Pattern Recognition: Joint IAPR International Workshop, SSPR&SPR 2012, Hiroshima, Japan, November 7–9, 2012. Proceedings*, vol. 7626 of *Lecture Notes in Computer Science*, pp. 719–726, Springer, Berlin, Germany, 2012.
- [24] J. Du, C. X. Ling, and Z.-H. Zhou, "When does cotraining work in real data?" *IEEE Transactions on Knowledge and Data Engineering*, vol. 23, no. 5, pp. 788–799, 2011.
- [25] S. Sun and F. Jin, "Robust co-training," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 25, no. 7, pp. 1113–1126, 2011.
- [26] Y. Zhou and S. Goldman, "Democratic co-learning," in *Proceedings of the 16th IEEE International Conference on Tools with Artificial Intelligence (ICTAI '04)*, pp. 594–602, Boca Raton, Fla, USA, November 2004.
- [27] S. Wang, L. Wu, L. Jiao, and H. Liu, "Improve the performance of co-training by committee with refinement of class probability estimations," *Neurocomputing*, vol. 136, pp. 30–40, 2014.
- [28] L. I. Kuncheva, "Using diversity measures for generating error-correcting output codes in classifier ensembles," *Pattern Recognition Letters*, vol. 26, no. 1, pp. 83–90, 2005.
- [29] L. I. Kuncheva and C. J. Whitaker, "Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy," *Machine Learning*, vol. 51, no. 2, pp. 181–207, 2003.
- [30] Z. Jiang, S. Zhang, and J. Zeng, "A hybrid generative/discriminative method for semi-supervised classification," *Knowledge-Based Systems*, vol. 37, pp. 137–145, 2013.
- [31] Z.-H. Zhou and M. Li, "Tri-training: exploiting unlabeled data using three classifiers," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 11, pp. 1529–1541, 2005.
- [32] T. Guo and G. Li, "Improved tri-training with unlabeled data," *Advances in Intelligent and Soft Computing*, vol. 115, no. 2, pp. 139–147, 2012.
- [33] C. Deng and M. Z. Guo, "A new co-training-style random forest for computer aided diagnosis," *Journal of Intelligent Information Systems*, vol. 36, no. 3, pp. 253–281, 2011.
- [34] M. F. A. Hady and F. Schwenker, "Co-Training by Committee: a new semi-supervised learning framework," in *Proceedings of the IEEE International Conference on Data Mining Workshops (ICDM '08)*, pp. 563–572, Pisa, Ital, December 2008.
- [35] J. Wang, S.-W. Luo, and X.-H. Zeng, "A random subspace method for co-training," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN '08)*, pp. 195–200, IEEE, Hong Kong, June 2008.
- [36] Y. Yaslan and Z. Cataltepe, "Co-training with relevant random subspaces," *Neurocomputing*, vol. 73, no. 10–12, pp. 1652–1661, 2010.
- [37] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software: an update," *ACM SIGKDD Explorations Newsletter*, vol. 11, no. 1, pp. 10–18, 2009.
- [38] J. Alcalá-Fdez, A. Fernández, J. Luengo et al., "KEEL data-mining software tool: data set repository, integration of algorithms and experimental analysis framework," *Journal of Multiple-Valued Logic and Soft Computing*, vol. 17, no. 2–3, pp. 255–287, 2011.
- [39] C. Deng and M. Z. Guo, "Tri-training and data editing based semi-supervised clustering algorithm," in *MICAI 2006: Advances in Artificial Intelligence*, vol. 4293 of *Lecture Notes in Computer Science*, pp. 641–651, Springer, Berlin, Germany, 2006.

- [40] S. García, A. Fernández, J. Luengo, and F. Herrera, “Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: experimental analysis of power,” *Information Sciences*, vol. 180, no. 10, pp. 2044–2064, 2010.
- [41] S. Holm, “A simple sequentially rejective multiple test procedure,” *Scandinavian Journal of Statistics*, vol. 6, no. 2, pp. 65–70, 1979.
- [42] J. Demsar, “Statistical comparisons of classifiers over multiple data sets,” *Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.
- [43] Y. Hochberg, “A sharper bonferroni procedure for multiple tests of significance,” *Biometrika*, vol. 75, no. 4, pp. 800–802, 1988.
- [44] Z. Xu, I. King, M. R.-T. Lyu, and R. Jin, “Discriminative semi-supervised feature selection via manifold regularization,” *IEEE Transactions on Neural Networks*, vol. 21, no. 7, pp. 1033–1047, 2010.

Hindawi

Submit your manuscripts at
<http://www.hindawi.com>