Knowl Inf Syst
DOI 10.1007/s10115-013-0706-y

SURVEY PAPER

Self-labeled techniques for semi-supervised learning: taxonomy, software and empirical study

Isaac Triguero · Salvador García · Francisco Herrera

Received: 14 May 2013 / Revised: 21 August 2013 / Accepted: 5 November 2013

Abstract Semi-supervised classification methods are suitable tools to tackle training sets with large amounts of unlabeled data and a small quantity of labeled data. This problem has been addressed by several approaches with different assumptions about the characteristics of the input data. Among them, self-labeled techniques follow an iterative procedure, aiming to obtain an enlarged labeled data set, in which they accept that their own predictions tend to be correct. In this paper, we provide a survey of self-labeled methods for semi-supervised classification. From a theoretical point of view, we propose a taxonomy based on the main characteristics presented in them. Empirically, we conduct an exhaustive study that involves a large number of data sets, with different ratios of labeled data, aiming to measure their performance in terms of transductive and inductive classification capabilities. The results are contrasted with nonparametric statistical tests. Note is then taken of which self-labeled models are the best-performing ones. Moreover, a semi-supervised learning module has been developed for the Knowledge Extraction based on Evolutionary Learning software, integrating analyzed methods and data sets.

Keywords Learning from unlabeled data · Semi-supervised learning · Self-training · Co-training · Multi-view learning · Classification

I. Triguero (✉) · F. Herrera
Department of Computer Science and Artificial Intelligence, Research Center on Information and Communications Technology (CITIC-UGR), University of Granada, 18071 Granada, Spain
e-mail: triguero@decsai.ugr.es

F. Herrera
e-mail: herrera@decsai.ugr.es

S. García
Department of Computer Science, University of Jaén, 23071 Jaén, Spain
e-mail: sglopez@ujaen.es

1 Introduction

The semi-supervised learning (SSL) paradigm [1] has attracted much attention in many different fields ranging from bioinformatics to Web mining, where it is easier to obtain unlabeled than labeled data because it requires less effort, expertise and time consumption. In this context, traditional supervised learning [2] is limited to using labeled data to build a model [3]. Nevertheless, SSL is a learning paradigm concerned with the design of models in the presence of both labeled and unlabeled data. Essentially, SSL methods use unlabeled samples to either modify or reprioritize the hypothesis obtained from labeled samples alone.

SSL is an extension of unsupervised and supervised learning by including additional information typical of the other learning paradigm. Depending on the main objective of the methods, we can divide SSL into semi-supervised classification (SSC) [4] and semi-supervised clustering [5,6]. The former focuses on enhancing supervised classification by minimizing errors in the labeled examples, but it must also be compatible with the input distribution of unlabeled instances. The latter, also known as constrained clustering, aims to obtain better-defined clusters than those obtained from unlabeled data. We focus on SSC.

SSC can be categorized into two slightly different settings [7], denoted transductive and inductive learning. On the one hand, transductive learning concerns the problem of predicting the labels of the unlabeled examples, given in advance, by taking both labeled and unlabeled data together into account to train a classifier. On the other hand, inductive learning considers the given labeled and unlabeled data as the training examples, and its objective is to predict unseen data. In this paper, we address both settings in order to carry out an extensive analysis of the performance of the studied methods.

Many different approaches have been suggested and studied in order to classify using unlabeled data in SSC. They usually make different assumptions related to the link between the distribution of unlabeled and labeled data. Generative models [8] learn a joint probability model that depends on the assumption that the data follow a determined parametric model. There are also other algorithms such as transductive inference for support vector machines [9] that assume that the classes are well separated and do not cut through dense unlabeled data. Alternatively, SSC can also be viewed as a graph min-cut problem [10]. If two instances are connected by a strong edge, their labels are likely to be the same. In this case, the graph construction determines the behavior of this kind of algorithm [11]. In addition, there are recent studies that address multiple assumptions in one model [7,12].

A successful methodology to tackle the SSC problem is based on traditional supervised classification algorithms [2]. These techniques aim to obtain one (or several) enlarged labeled set(s), based on their most confident predictions, to classify unlabeled data. We denote these algorithms self-labeled techniques. In the literature [1], self-labeled techniques are typically divided into self-training and co-training. Self-training [13,14] is a simple and effective SSL methodology that has been successfully applied in many real instances. In the self-training process, a classifier is trained with an initial small number of labeled examples, aiming to classify unlabeled points. Then it is retrained with its own most confident predictions, enlarging its labeled training set. This model does not make any specific assumptions for the input data, but it accepts that its own predictions tend to be correct. The standard co-training [15] methodology assumes that the feature space can be split into two different conditionally independent views and that each view is able to predict the classes perfectly [16–18]. It trains one classifier in each specific view, and then the classifiers teach each other the most confidently predicted examples. Multi-view learning [19] for SSC is usually understood to be a generalization of co-training, without requiring explicit feature splits or the iterative mutual-teaching procedure [20–22]. However, these concepts are sparse and frequently confused in the literature.

There is currently no general categorization focused on self-labeled techniques. In the literature, we find several SSL surveys [23,24]. These include a general classification of SSL methods, dividing self-labeled techniques into self-training, co-training and multi-view learning, but they are not exclusively focused on self-labeled techniques nor especially on studying the similarities among them. Furthermore, we can find a good theoretical survey about disagreement-based models in [25], introducing research advances in this paradigm. Nevertheless, this categorization is a subset of self-labeling approaches without an explicit definition of a taxonomy.

Because of the absence of a focused taxonomy in the literature, we have observed that the new algorithms proposed are usually compared with only a subset of the complete family of self-labeled methods. Furthermore, in most of the studies, no rigorous analysis has been carried out. They do not follow a complete experimental framework. Instead, the new proposal is usually contrasted with a reduced number of data sets, only analyzing either the transductive or the inductive capabilities of the algorithms.

These are the reasons that motivate the global purpose of this paper, which can be divided into four objectives:

- To propose a new and complete taxonomy based on the main properties observed in self-labeled methods. The taxonomy will allow us to discern the advantages and drawbacks from a theoretical point of view. As a result, many frequently confused concepts will be clarified.
- To establish an experimental methodology to analyze the transductive and inductive capabilities of these kinds of algorithms.
- To make an empirical study of the state of art of self-labeled techniques. Our goal is to identify the best methods in each family, depending on the ratio of labeled data, and to stress the relevant properties of each one.
- To provide an open-source SSL module for the Knowledge Extraction based on Evolutionary Learning (KEEL) software tool [26]. This is a research tool for solving data mining problems which contains a great number of machine learning algorithms. The source code of the analyzed algorithms and a wide variety of data sets are available in this module.

We will conduct experiments involving a total of 55 UCI/KEEL classification data sets with different ratios of labeled data: 10, 20, 30 and 40 %. The experimental study will include a statistical analysis based on nonparametric statistical tests [27,28]. Then, we will test the performance of the best-performing methods over nine data sets obtained from the book of Chapelle [4]. These problems have a great number of features and a very reduced number of labeled data. A Web site with all the complementary material is available at http://sci2s.ugr.es/SelfLabeled, including this paper's basic information, the source code of the analyzed algorithms, all the data sets involved and the complete results obtained.

The rest of the paper is organized as follows: Sect. 2 provides a description of the properties and an enumeration of the methods, as well as related and advanced work on SSC. Section 3 presents the taxonomy proposed. In Sect. 4 we describe the experimental framework, and Sect. 5 examines the results obtained and presents a discussion on them. In Sect. 6 we summarize our conclusions. "Appendix" shows details and characteristics of the SSL software module.

2 Self-labeled techniques: background

The SSC problem can be defined as follows: Let $\mathbf{x}_p$ be an example where $\mathbf{x}_p = (\mathbf{x}_{p1}, \mathbf{x}_{p2}, \ldots, \mathbf{x}_{pD}, \omega)$, with $\mathbf{x}_p$ belonging to a class ω and a D-dimensional space in which $\mathbf{x}_{pi}$ is the value of the ith feature of the pth sample. Then, let us assume that there is a labeled set L which consists of $\mathbf{n}$ instances $\mathbf{x}_p$ with ω known. Furthermore, there is an unlabeled set U which consists of $\mathbf{m}$ instances $\mathbf{x}_q$ with ω unknown, let $\mathbf{m} > \mathbf{n}$. The $L \cup U$ set forms the training set (denoted as TR). The purpose of SSC is to obtain a robust learned hypothesis using TR instead of L alone, which can be applied in two slightly different settings: transductive and inductive learning.

Transductive learning is described as the application of an SSC technique to classify all the $\mathbf{m}$ instances $\mathbf{x}_q$ of U correctly. The class assignment should represent the distribution of the classes efficiently, based on the input distribution of unlabeled instances and the L instances.

Let TS be a test set composed of $\mathbf{t}$ unseen instances $\mathbf{x}_r$ with ω unknown, which has not been used at the training stage of the SSC technique. The inductive learning phase consists of correctly classifying the instances of TS based on the previously learned hypothesis.

This section presents an overview of self-labeled techniques. Three main topics will be discussed:

- In Sect. 2.1, the common characteristics in self-labeled techniques which will define the categories of the taxonomy proposed in this paper will be outlined.
- In Sect. 2.2, we briefly enumerate all the self-labeled techniques proposed in the literature. The complete and abbreviated name will be given together with the proposal reference.
- Finally, Sect. 2.3 explores other areas related to self-labeled techniques and provides a summary of advanced work in this research field.

2.1 Common characteristics in self-labeled techniques

This section provides a framework, establishing different properties of self-labeled techniques, for the definition of the taxonomy in the next section. Other issues that influence some of the methods are presented in this section although they are not involved in the proposed taxonomy. Finally, some criteria will be set in order to compare self-labeled methods.

2.1.1 Main properties of self-labeled techniques

Self-labeled methods search iteratively for one or several enlarged labeled set(s) (EL) of prototypes to efficiently represent the TR. For simplicity in reading, in what follows we restrict the description of these properties to one EL. The taxonomy proposed will be based on these characteristics:

- *Addition mechanism*: There are a variety of schemes in which an EL is formed.
  - **Incremental**: A strictly incremental approach begins with an enlarged labeled set $EL = L$ and adds, step-by-step, the most confident instances of U if they fulfill certain criteria. In this case, the algorithm crucially depends on the way in which it determines the confidence predictions of each unlabeled instance, that is, the probability of belonging to each class. Under such a scheme, the order in which the

instances are added to the EL determines the learning hypotheses and therefore the following stages of the algorithm. One of the most important aspects of this approach is the number of instances added in each iteration. On the one hand, this number could be defined as a constant parameter and/or independent of the classes of the instances. On the other hand, it can be chosen as a proportional value of the number of instances of each class in L. In our experiments, we implement the latter as suggested in [15]. This is the most simple and intuitive way of addressing the SSL problem which often corresponds to classical self-labeled approaches. One advantage of this approach is that it can be faster during the learning phase than nonincremental algorithms. Nevertheless, the main disadvantage is that strictly incremental algorithms can add instances with erroneous predictions to the class label. Hence, if it occurs, the learned hypotheses could produce a low performance in transductive and inductive phases.

- **Batch**: Another way to generate an EL set is in batch mode. This involves deciding whether each instance meets the addition criteria before adding any of them to the EL. Then, all those that do meet the criteria are added at once. In this sense, batch techniques do not assign a definitive class to each unlabeled instance during the learning process. They can reprioritize the hypotheses obtained from labeled samples. Batch processing suffers from increased time complexity over incremental algorithms.
- **Amending**: Amending models appeared as a solution to the main drawback of the strictly incremental approach. In this approach, the algorithm starts with $EL = L$ and iteratively can add or remove any instance that meets the specific criterion. This mechanism allows rectifications to already performed operations, and its main advantage is to make the achievement of a good accuracy-suited EL set of instances easy. As in the incremental approach, its behavior can also depend on the number of instances added per iteration. Typically, these methods have been designed to avoid the introduction of noisy instances into EL at each iteration [14,29]. However, under the rubric of amending model, many other proposals can be developed. For instance, incremental and batch mode algorithms in combination with a prior or a later cleaning phase of noisy instances are considered to be an amending model. Despite its flexibility, this scheme usually requires high computational demands compared to incremental and batch algorithms.

• *Single-classifier versus multi-classifier*: Self-labeled techniques can use one or more classifiers during the enlarging phase of the labeled set. As we stated before, all of these methods follow a wrapper methodology using classifier(s) to establish the possible class of unlabeled instances.
In a single-classifier model, each unlabeled instance belongs to the most probable class assigned by the uniquely used classifier. It implies that these probabilities should be explicitly measured. There are different ways in which these confidences are computed. For example, in probabilistic models, such as naive Bayes, the confidence predictions can usually be measured as the output probability in prediction, and other methods, such as nearest neighbor [30], could approximate confidence in terms of distance. In general, the main advantage of single-classifier methods is their simplicity, allowing us to compute faster confidence probabilities.
Multi-classifier methods combine the learned hypotheses with several classifiers to predict the class of unlabeled instances. The underlying idea of using multi-classifiers in SSL is that several weak classifiers, learned with a small number of instances, can produce bet-

ter generalization capabilities than only one weak classifier. These methods are motivated, in part, by the empirical success of ensemble methods. Two different approaches are commonly used to calculate the confidence predictions in multi-classifier methods: agreement of classifiers and combination of the probabilities obtained by single-classifiers. These techniques usually obtain a more accurate precision than single-classifier models. Another effect of using several classifiers in self-labeled techniques is their complexity. In both approaches, different confidence measures have been analyzed in the literature. They will be explained in Sect. 2.1.2.

- *Single-learning versus multi-learning*: Apart from the number of classifiers, a key concern is whether they are constituted by the same (single) or different (multiple) learning algorithms. The number of different learning algorithms used can also determine the confidence predictions of unlabeled data.
  In a multi-learning approach, the confidence predictions are computed as the integration of a group of different kinds of learners to boost the performance classification. It works under the hypothesis that different learning techniques present different behaviors, using the bias classification between them, which generate locally different models. Multi-learning methods are closely linked with multi-classifier models. A multi-learning method is itself a multi-classifier method in which the different classifiers come from different learning methods. Hence, the general properties explained above for multi-classifiers are also extrapolated to multi-learning. A specific drawback of this approach is the choice of the most adequate learning approaches.
  By contrast, a single-learning approach could be linked to both single and multi-classifiers. With a single-learning algorithm, the confidence prediction measurement is relaxed with regard to the rest of the properties, type of view and number of classifiers. The goal of this approach, which has been shown to perform well in several domains, is simplicity [20].
- *Single-view versus multi-view*: This characteristic refers to the way in which the input feature space is taken into consideration by the self-labeled technique. In a multi-view self-labeled technique, L is split into two or more subsets of the features L_{sub_k} of dimension M ($M < D$), by projecting each instance $\mathbf{x}_p$ of L onto the selected M-dimensional subspace. Multi-view requires redundant and conditionally independent views, provided that the class attribute and each subset are sufficient to train a good classifier. Therefore, the performance of these kinds of techniques depends on the division procedure followed. One important advantage of this kind of view is that if this assumption is met [31], the multi-view approach makes fewer generalization errors, using the hypothetical agreement of classifiers. Nevertheless, this could lead to adverse results being obtained because the compatibility and independence of the features' subsets obtained are strong assumptions and many real data sets cannot satisfy these hypotheses.
  Otherwise, a single-view self-labeled technique does not make any specific assumptions for the input data. Hence, it uses the complete L to train. Most of the self-labeled techniques adhere to this due to the fact that in many real-world data sets the feature input space cannot be appropriately divided into conditionally independent views.
  In the literature, many methods have been erroneously classified as multi-view or co-training methods without the requirement of sufficient and redundant views. This term is frequently confused with single-view methods that use multi-classifiers. Note that multi-view methods need, by definition, several classifiers. However, a single-view method could use single and multi-classifiers.

2.1.2 Other properties

We may remark upon other properties that explore how self-labeled methods work. Although they influence the operation and hence the results obtained with some of these techniques, we have not included them in the proposed taxonomy for the sake of simplicity and usability.

- *Confidence measures*: An inaccurate confidence measure leads to adding mislabeled examples to the EL, which implies a performance degradation of the self-labeling process. The previously explained characteristics define in a global manner the way in which the confidence predictions are estimated. Nevertheless, the specific combination of these characteristics leads to different ideas to establish these probabilities.
  - **Simple**: As we stated before, the probability predictions can be extracted from the used learning model. This characteristic is essentially presented in single-classifier methods. For example, probabilistic models return an associated probability of belonging to each class for each unlabeled instance. In decision tree algorithms [32], the confidence probabilities can be estimated as the accuracy of the leaf that makes the prediction. Instance-based learning approaches can estimate probabilities in terms of dissimilarity between instances.
  - **Agreement and combination**: Multi-classifier methods can approximate their confidence probabilities based on the predictions obtained for each classifier. One way to calculate them is the hypothetical agreement of the used classifiers. Typically, a majority voting rule is used in which ties are commonly solved by a random process. By contrast, there are other methods that generate their confidence predictions as the aggregation of the obtained probabilities for each classifier in order to find a final confidence value for each unlabeled instance. Furthermore, it would also be possible to combine the agreement of classifiers with the calculated probabilities, developing a hybrid confidence prediction model. Not much research is currently underway with regard to this latter scheme.

    When considering these schemes, some questions should be taken into account depending on the number of different learning algorithms used. In a single-learning proposal, the method should retain multiple different hypotheses and combine (agreement or combination) their decisions during the computation of confidence probabilities. In this framework, a mandatory step is to generate new labeled sets based on the original L. For this purpose, self-labeled techniques usually apply bootstrapping techniques, such as resampling [33]. Nevertheless, there are other proposals such as a tenfold cross-validation scheme to create new labeled sets as is proposed in [34]. The effect obtained is related to the improvement of generalization capabilities with respect to the use of a single-classifier. However, in SSL, the labeled set tends to be small and the diversity obtained by the bootstrap sampling is limited, due to the fact that the obtained bootstraps are very similar. In multi-learning approaches, diversity is obtained from the different learning algorithms. Thus, they do not generate new labeled sets.

- *Self-teaching versus mutual-teaching*: Independently of the learning procedure used, one of these characteristics appears in multi-classifier methods. In a mutual-teaching approach, the classifiers teach each other their most confident predicted examples. Each C_i classifier has an associated EL_i, which is initialized in different ways. At each stage, all the classifiers are trained with its respective EL_i, and then EL_i is increased, with the most confident examples obtained as the hypotheses combination (or agreement) of the

remaining classifiers. Under this scheme, a classifier C_i does not use its own predictions to increase its EL_i. However, if C_i is unable to detect some interesting unlabeled instances as target examples, the rest of the classifiers may teach different hypotheses to C_i.
To construct the final learned hypothesis, two approaches can be followed: join or voting procedures. The join procedure consists of forming a complete EL through the combination of each EL_i without repetitions. With a voting procedure, the final hypothesis is obtained as the application of a majority voting rule, using all the EL_i.
By contrast, the self-teaching property refers to those multi-classifiers that maintain a single EL. In this case, the combination of hypotheses is blended to form a unique EL. As far as we know, there is no proposed model that combines mutual-teaching and self-teaching.
- *Stopping criteria*: This is related to the mechanism used to stop the self-labeling process. It is an important factor due to the fact that it implies the size of the EL formed and therefore in the learned hypothesis. Three main approaches can be found in the literature. Firstly, in classical approaches such as self-training, the self-labeling process is repeated until all the instances from U have been added to EL. If erroneous unlabeled instances are added to the EL, they can damage the classification performance. Secondly, in [15], the authors suggest choosing examples from a smaller pool instead of the whole unlabeled set to form the EL, establishing a limited number of iterations. This approach has been successfully applied to many self-labeled methods, outperforming the overall classification rate of the previous stopping criteria. However, the maximum number of iterations is usually prefixed, and it is not adaptive to the size of the data set used. Thirdly, the termination criteria can be satisfied when the used classifiers, in the self-labeling process, do not change the learned hypotheses. This criteria limits the number of unlabeled instances added to EL; however, it does not ensure that erroneous unlabeled instances will not be added to EL.

2.1.3 Criteria to compare self-labeled methods

When comparing self-labeled methods, there are a number of criteria that can be used to compare the relative strengths and weaknesses of each algorithm. These include transductive and inductive accuracy, influence of the number of labeled instances, noise tolerance and time requirements.

- *Transductive and inductive accuracy*: A successful self-labeling algorithm will often be able to appropriately enlarge the labeled set, increasing the transductive and inductive accuracy.
- *Influence of the number of labeled instances*: The number of labeled instances that a self-labeled technique manages determines the learned hypothesis. A good technique should be able to learn an appropriate hypothesis with the lowest possible number of labeled instances.
- *Noise tolerance*: Noisy instances can be harmful if they are added to the labeled set, as they bias the classification of unlabeled data to incorrect classes, which could make the enlarged labeled set in the next iteration even more noisy. This problem may especially occur in the initial stages of this process, and it can also be more harmful with a reduced number of labeled instances. Two types of noisy instances may appear during the self-labeling process. The first one is caused by the distribution of the instances of L in their respective classes. It can lead the classifier to erroneously label some instances. Second,

there may be outliers within the original unlabeled data. These can be detected, avoiding their labeling and inclusion in L.

- *Time requirements*: The learning process is usually carried out just once on a training set. If the objective of a specific application is related to transductive learning, this learning process becomes more important as it is responsible for managing unlabeled data. For inductive learning, it does not seem to be a very important evaluation method because test instances are classified based on the learned model. However, if the learning phase takes too long, it can become impractical for real applications.

2.2 Self-labeled methods

We have performed a deep search of the specialized literature, identifying the most relevant self-labeled algorithms. At the time of writing, more than 20 methods have been proposed in the literature. This section is devoted to enumerating and designating them according to the standard followed in this paper. For more details on their implementations, the reader can visit the URL http://sci2s.ugr.es/SelfLabeled. Implementations of most of the algorithms in java can be found in the KEEL software [35] (see "Appendix").

Table 1 presents an enumeration of self-labeled methods reviewed in this paper. The complete name, abbreviation and reference are provided for each one. In the case of there being more than one method in a row, they were proposed together or by the same authors and the best-performing method (indicated by the respective authors) is depicted in bold. We

Table 1 Self-labeled methods reviewed

Complete name	Abbr. name	References
Standard self-training	Self-Training	[13]
Standard co-training	Co-Training	[15]
Statistical co-learning	Statistical-Co	[34]
ASSEMBLE	ASSEMBLE	[36]
Democratic co-learning	Democratic-Co	[37]
Self-training with editing	SETRED	[14]
Tri-training	TriTraining	[20]
Tri-training with editing	DE-TriTraining	[38]
Co-forest	CoForest	[21]
Random subspace method for co-training	Rasco	[39]
Co-training by committee: AdaBoost	Co-Adaboost	[40,41]
Co-training by committee: bagging	Co-Bagging	
Co-training by committee: RSM	Co-RSM	
Co-training by committed: Tree-structured ensemble	Co-Tree	[42]
Co-training with relevant random subspaces	Rel-Rasco	[43]
Classification algorithm based on local clusters centers	CLCC	[44]
Ant-based semi-supervised classification	APSSC	[45]
Self-training nearest neighbor rule using cut edges	SNNRCE	[46]
Robust co-training	R-Co-Training	[17]
Adaptive co-forest editing	ADE-CoForest	[47]
Co-training with NB and SVM classifiers	Co-NB-SVM	[18]

will test some representative methods therefore only the methods in bold will be compared in the experimental study.

2.3 Related and advanced work

Nowadays, the use of unlabeled data in conjunction with labeled data is a growing field in different research lines. Self-labeled techniques form a feasible and promising group to make use of both kinds of data, which is closely related to other methods and problems. This section provides a brief review of other topics related to self-labeled techniques and describes other interesting work and future trends which have been studied in the last few years.

With the same objective as self-labeled techniques, we group the three following methodologies according to Zhu and Goldberg's book [1]:

- *Generative models and cluster-then-label methods*: The first attempts to deal with unlabeled data correspond to this area. It includes those methods that assume a joint probability model $p(x, y) = p(y)p(x|y)$, where $p(x|y)$ is an identifiable mixture distribution, for example, a Gaussian mixture model. Hence, it follows a determined parametric model [48] using both unlabeled and labeled data. Cluster-then-label methods are closely related to generative models. Instead of using a probabilistic model, they apply a previous clustering step to the whole data set, and then they label each cluster with the help of labeled data. Recent advances in these topics are [8,49].
- *Graph-based*: This represents the SSC problem as a graph min-cut problem [10]. Labeled and unlabeled examples constitute the graph nodes, and the similarity measurements between nodes correspond to the graph edges. The graph construction determines the behavior of this kind of algorithm [50,51]. These methods usually assume label smoothness over the graph. Its main characteristics are nonparametric, discriminative and transductive in nature. Advanced proposals can be found in [11,52].
- *Semi-supervised support vector machines* (S^3VM): S^3VM is an extension of standard support vector machines (SVM) with unlabeled data [53]. This approach implements the *cluster-assumption* for SSL, that is, examples in data cluster have similar labels, so classes are well separated and do not cut through dense unlabeled data. This methodology is also known as *transductive SVM*, although it learns an inductive rule defined over the search space. Advanced works in S^3VM are [54–56].

Regarding other problems connected with self-labeled techniques, we briefly describe the following topics:

- *Semi-supervised clustering*: This problem, also known as constrained clustering, aims to obtain better-defined clusters than the ones obtained from unlabeled data [57]. Labeled data are used to define pairwise constraints between examples, *must-links* and *cannot-links*. The former link establishes the examples that must be in the same cluster, and the latter refers to those examples that cannot be in the same cluster [58]. A brief review of this topic can be found in [59].
- *Active learning*: With the same objective as SSL, avoiding the cost of data labeling, active learning [60] tries to select the most important examples from a pool of unlabeled data. These examples are queried by an expert and are then labeled with the appropriate class, aiming to minimize effort and time consumption. Many active learning algorithms select as query the examples with maximum label ambiguity or least confidence. Several hybrid methods between self-labeled techniques and active learning [41,61–63] have been proposed and show that active learning queries maximize the generalization capabilities of SSL.

- *Semi-supervised dimensionality reduction*: This area studies the curse of dimensionality when it is addressed in an SSL framework. The goal of dimensionality reduction algorithms is to find a faithful low-dimensional mapping, or selection, of the high-dimensional data. Traditional dimensionality reduction techniques designed for supervised and unsupervised learning, such as linear discriminant analysis [64] and principal component analysis [65], are not appropriate to deal with both labeled and unlabeled data. Recently, many different frameworks have been proposed to use classical methods in this environment [66,67]. Two well-known dimensionality reduction solutions are feature selection and feature extraction [68] which have attracted much attention in recent years [69].
- *Self-supervised*: This paradigm has been recently presented in [70]. It integrates knowledge from labeled data with some features and knowledge from unlabeled data with all the features. Thus, self-supervised algorithms learn novel features from unlabeled examples without destroying partial knowledge previously acquired from labeled examples.
- *Partial label*: It is a paradigm proposed in [71]. This problem deals with partially labeled multi-class classification, in which instead of a single label per instance, it has a candidate set of labels, only one of which is correct. In these circumstances, a classifier should learn how to disambiguate the partially labeled training instance and generalize to unseen data.

3 Self-labeled techniques: taxonomy

The main characteristics of the self-labeled methods have been described in Sect. 2.1.1, and they can be used to categorize the self-labeled methods proposed in the literature. The type of view, number of learning algorithms, number of classifiers and addition mechanism constitute a set of properties that define each method. This section presents the taxonomy of self-labeled techniques based on these properties.

Figure 1 illustrates the categorization following a hierarchy based on this order: type of view, number of learning algorithms, number of classifiers and addition mechanism. Considering this figure and the year of publication of each analyzed method, some interesting observations about the existing and nonexisting proposals can be made:

- The number of single-classifier methods is smaller than multi-classifier. They constitute four of the 15 methods proposed in the literature. Although these methods may obtain great results, they do not have a refined confidence prediction mechanism because, in general, they are limited to extracting the most confident predictions from the learner used. Nevertheless, two of the most recent approaches belong to this category.
- Amending models appeared a few years ago. Most of the recent research efforts are focused on these kinds of models because they have reported a great synergy with the iterative scheme presented in self-labeled methods. In different ways, they remove those instances that are harmful to the classification task in order to alleviate the main drawback of the incremental addition mechanism.
- Only two multi-learning approaches have been proposed for self-labeled algorithms, and the most recent was published in 2004. In our opinion, more research is required in this area. For instance, there is no amending model that avoids introducing noisy instances into the enlarged labeled set. Similarly, no amending approaches have been designed for the family of methods which uses multiple views.
- Standard co-training has been widely used in many real applications [72]. However, there are a reduced number of advanced multi-view approaches, which, for example, use multi-learning or amending addition schemes.

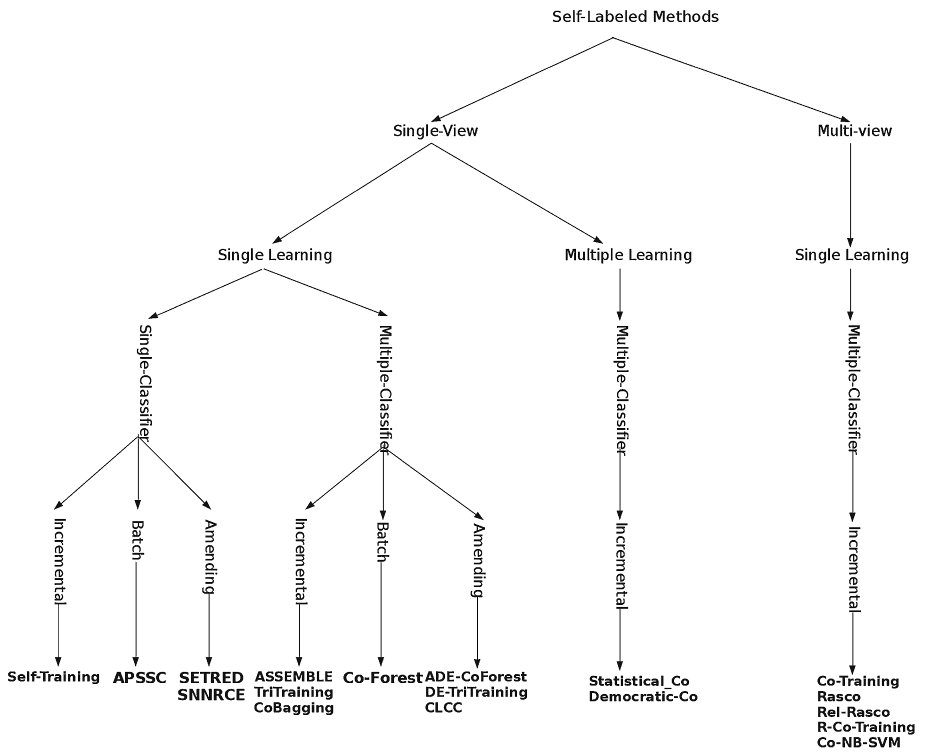


Fig. 1 Self-labeled techniques hierarchy

The properties studied here can help us to understand how the self-labeled algorithms work. In the following sections, we will establish which methods perform best for each family, considering several metrics of performance with a wide experimental framework.

4 Experimental framework

This section describes all the properties and issues related to the experimental framework followed in this paper. We provide the measures used to observe differences in the performance of the algorithms (Sect. 4.1), the main characteristics of the problems used (Sect. 4.2), the parameters of the algorithms and the base classifiers used (Sect. 4.3) and finally a brief description of the nonparametric statistical tests used to contrast the results obtained (Sect. 4.4).

4.1 Performance measures

Two performance measures are commonly applied because of their simplicity and successful application when multi-class classification problems are dealt with. As standard classification methods, the performance of SSC algorithms can be measured in terms of accuracy [1] and Cohen's kappa rate [73]. They are briefly explained as follows:

- *Accuracy*: It is the number of successful hits (correct classifications) relative to the total number of classifications. It has been by far the most commonly used metric for assessing the performance of classifiers for years [2,74].
- *Cohen's kappa (Kappa rate)*: It evaluates the portion of hits that can be attributed to the classifier itself, excluding random hits, relative to all the classifications that cannot be attributed to chance alone. Cohen's kappa ranges from -1 (total disagreement) through 0 (random classification) to 1 (perfect agreement). For multi-class problems, kappa is a very useful, yet simple, meter for measuring a classifier's accuracy while compensating for random successes.

Both metrics will be adopted to measure the efficacy of self-labeled methods in transductive and inductive phases.

4.2 Data sets

The experimentation is based on 55 standard classification data sets taken from the UCI repository [75] and the KEEL-data set repository[1] [35]. Table 2 summarizes the properties of the selected data sets. It shows, for each data set, the number of examples (#Examples), the number of attributes (#Features) and the number of classes (#Classes). The data sets considered contain between 100 and 19,000 instances, the number of attributes ranges from 2 to 90, and the number of classes varies between 2 and 28.

These data sets have been partitioned using the tenfold cross-validation procedure, that is, the data set has been split into ten folds, each one containing 10 % of the examples of the data set. For each fold, an algorithm is trained with the examples contained in the rest of folds (training partition) and then tested with the current fold. It is noteworthy that test partitions are kept aside to evaluate the performance of the learned hypothesis.

Each training partition is divided into two parts: labeled and unlabeled examples. Using the recommendation established in [46], in the division process we do not maintain the class proportion in the labeled and unlabeled sets since the main aim of SSC is to exploit unlabeled data for better classification results. Hence, we use a random selection of examples that will be marked as labeled instances, and the class label of the rest of the instances will be removed. We ensure that every class has at least one representative instance.

In order to study the influence of the amount of labeled data, we take different ratios when dividing the training set. In our experiments, four ratios are used: 10, 20, 30 and 40 %. For instance, assuming a data set that contains 1,000 examples, when the labeled rate is 10 %, 100 examples are put into L with their labels, while the remaining 900 examples are put into U without their labels. In summary, this experimental study involves a total of 220 data sets (55 data sets $\times$ 4 labeled rates).

Apart from these data sets, the best methods will be also tested with nine high-dimensional problems. These data sets have been extracted from the book of Chapelle [4]. To analyze transductive and inductive capabilities, these data sets have been also partitioned using the methodology explained above, except for the number of labeled data. We will use two splits for training partitions with 10 and 100 labeled examples, respectively. The remaining instances are marked as unlabeled points. Table 3 presents the main characteristics of these data sets.

All the data sets created can be found on the Web site associated with this paper.[2]

[1] http://sci2s.ugr.es/keel/datasets.

[2] http://sci2s.ugr.es/SelfLabeled.

Table 2 Summary description of the original data sets

Data set	#Examples	#Features	#Classes	Data set	#Examples	#Features	#Classes
abalone	4,174	8	28	movement_libras	360	90	15
appendicitis	106	7	2	mushroom	8,124	22	2
australian	690	14	2	nursery	12,690	8	5
autos	205	25	6	pageblocks	5,472	10	5
banana	5,300	2	2	penbased	10,992	16	10
breast	286	9	2	phoneme	5,404	5	2
bupa	345	6	2	pima	768	8	2
chess	3,196	36	2	ring	7,400	20	2
cleveland	297	13	5	saheart	462	9	2
coil2000	9,822	85	2	satimage	6,435	36	7
contraceptive	1,473	9	3	segment	2,310	19	7
crx	125	15	2	sonar	208	60	2
dermatology	366	33	6	spambase	4,597	55	2
ecoli	336	7	8	spectheart	267	44	2
flare-solar	1,066	9	2	splice	3,190	60	3
german	1,000	20	2	tae	151	5	3
glass	214	9	7	texture	5,500	40	11
haberman	306	3	2	tic-tac-toe	958	9	2
heart	270	13	2	thyroid	7,200	21	3
hepatitis	155	19	2	titanic	2,201	3	2
housevotes	435	16	2	twonorm	7,400	20	2
iris	150	4	3	vehicle	846	18	4
led7digit	500	7	10	vowel	990	13	11
lymphography	148	18	4	wine	178	13	3
magic	19,020	10	2	wisconsin	683	9	2
mammographic	961	5	2	yeast	1,484	8	10
marketing	8,993	13	9	zoo	101	17	7
monks	432	6	2				

Table 3 Summary description of high-dimensional data sets

Data set	#Examples	#Features	#Classes
bci	400	117	2
coil	1,500	241	6
coil2	1,500	241	2
digit1	1,500	241	2
g241c	1,500	241	2
g241n	1,500	241	2
secstr	83,679	315	2
text	1,500	11,960	2
usps	1,500	241	2

4.3 Parameters and base classifiers

In this subsection we show the configuration parameters of all the methods used in this study. The selected values are common for all problems, and they were selected according to the recommendation of the corresponding authors of each algorithm, which are also the default parameter settings included in the KEEL software [26] that we used to develop our experiments. The approaches analyzed should be as general and as flexible as possible. A good choice of parameters facilitates their better performance over different data sources, but their operations should allow good enough results to be obtained although the parameters are not optimized for a specific data set. This is the main purpose of this experimental survey, to show the generalization in performance of each self-labeled technique. The configuration parameters of all the methods are specified in Table 4.

Some of the self-labeled methods have been designed with one or more specific base classifier(s). In this study, these algorithms maintain their used classifier(s). However, the interchange of the base classifier is allowed in other approaches. Specifically, they are: Self-Training, Co-Training, TriTraining, DE-TriTraining, Rasco and Rel-Rasco. In this study, we select four classic and well-known classifiers in order to find differences in performance among these self-labeled methods. They are K-nearest neighbor, C4.5, naive Bayes and SVM. All of these selected base classifiers have been considered as one of the ten most influential data mining algorithms in [76].

A brief description of the base classifiers and their associated confidence prediction computation are enumerated as follows:

- *K-nearest neighbor (KNN)*: This is one of the simplest and most effective methods based on dissimilarities among a set of instances. It belongs to the lazy learning family of methods [77], which do not build a model during the learning process. With this method, confidence predictions can be approximated by the distance to the currently labeled set.
- *C4.5*: This is a well-known decision tree algorithm [32]. It induces classification rules in the form of decision trees for a given training set. The decision tree is built with a top-down scheme, using the normalized information gain (difference in entropy) that is obtained from choosing an attribute for splitting the data. The attribute with the highest normalized information gain is the one used to make the decision. Confidence predictions are obtained from the accuracy of the leaf that makes the prediction. The accuracy of a leaf is the percentage of correctly classified train examples from the total number of covered train instances.
- *Naive Bayes (NB)*: Its aim is to construct a rule which will allow us to assign future objects to a class, assuming independence of attributes when probabilities are established. For continuous data, we follow a typical assumption in which continuous values associated with each class are distributed according to a Gaussian distribution [78]. The extraction of probabilities is straightforward, due to the fact that this method explicitly computes the probability belonging to each class for the given test instance.
- *Support vector machines (SVM)*: It maps the original input space into a higher-dimensional feature space using a certain kernel function [79]. In the new feature space, the SVM algorithm searches the optimal separating hyperplane with maximal margin in order to minimize an upper bound of the expected risk instead of the empirical risk. Specifically, we use the SMO training algorithm, proposed in [80], to obtain the SVM base classifiers. Using a logistic model, we can use the probability estimate from the SMO [80] as the confidence for the predicted class.

Table 4 Parameter specification for all the base learners and self-labeled methods used in the experimentation

Algorithm	Parameters
KNN	Number of neighbors = 3, Euclidean distance
C4.5	Confidence level: $c = 0.25$
	Mininum number of item-sets per leaf: $i = 2$
	Prune after the tree building
NB	No parameters specified
SMO	$C = 1.0$, tolerance parameter = 0.001
	Epsilon = 1.0×10^{-12}
	Kernel type = polynomial
	Polynomial degree = 1
	Fit logistic models = true
Self-Training	$MAX\_ITER = 40$
Co-Training	$MAX\_ITER = 40$, initial unlabeled pool = 75
Democratic-Co	Classifiers = 3NN, C4.5, NB
SETRED	$MAX\_ITER = 40$, threshold = 0.1
TriTraining	No parameters specified
DE-TriTraining	Number of neighbors $k = 3$, minimum number of neighbors = 2
CoForest	Number of RandomForest classifiers = 6, threshold = 0.75
Rasco	$MAX\_ITER = 40$, number of views/classifiers = 30
Co-Bagging	$MAX\_ITER = 40$, committee members = 3
	Ensemble learning = *Bagging*, pool U = 100
Rel-Rasco	$MAX\_ITER = 40$, number of views/classifiers = 30
CLCC	Number of RandomForest classifiers = 6, Threshold = 0.75
	Manipulative beta parameter = 0.4, initial number of cluster = 4
	Running frequency z = 10, best center sets = 6, Optional Step = True
APSSC	Spread of the Gaussian = 0.3, evaporation coefficient = 0.7, MT = 0.7
SNNRCE	Threshold = 0.5
ADE-CoForest	Number of RandomForest classifiers = 6, threshold = 0.75
	Number of neighbors $k = 3$, minimum number of neighbors = 2

4.4 Statistical test for performance comparison

Statistical analyses are highly recommended in the field of data mining to find significant differences between the results obtained by the studied methods . We consider the use of nonparametric tests according to the recommendation made in [27,81], where a set of simple, safe and robust nonparametric tests for statistical comparisons of classifiers is presented. In these studies, the use of nonparametric tests will be preferred to parametric ones, since the initial conditions that guarantee the reliability of the latter may not be satisfied, causing the statistical analysis to lose credibility.

The Wilcoxon signed-ranks test [28,82] will be adopted to conduct pairwise comparisons between all the methods considered in this study. Considering the ratio of the number of data sets to the number of methods that we will compare throughout this paper, we fix the significance level $\alpha = 0.1$ for all comparisons.

Furthermore, we will use the Friedman test [83] in the global analysis to analyze differences between the methods considered outstanding with a multiple comparison analysis. The Bergmann–Hommel [84] procedure is applied as a post hoc procedure, which was highlighted as the most powerful test in [81], to find out which algorithms are distinctive in $n * n$ comparisons. Any interested reader can find more information about these tests at http://sci2s.ugr.es/sicidm/, together with the software for applying the statistical tests.

4.5 Other considerations

We want to stress that the implementations are based only on the descriptions and specifications given by the respective authors in their papers. No advanced data structures and enhancements for improving the suitability of self-labeled methods have been applied.

5 Analysis of results

This section presents the average results collected in the experimental study and some discussion on them. Due to the extent of the experimental analysis carried out, we report the complete tables of results on the Web page associated with this paper (see footnote 2). This study will be divided into two different parts: analysis of the results obtained in transductive learning (see Sect. 5.1) and inductive learning (see Sect. 5.2) considering different ratios of labeled data. A global discussion and the identification of outstanding methods is added in Sect. 5.3. Some representative methods will be also tested under high-dimensional data sets with small labeled ratio (see Sect. 5.4). Finally, a comparison with the supervised learning paradigm is performed in Sect. 5.5.

5.1 Transductive results

As we claimed before, the main objective of transductive learning is to predict the true class label of the unlabeled data used to train. Hence, a good exploitation of unlabeled data can lead to successful results. Within the framework of transductive learning, we have analyzed which are the best or the most appropriate proposals attending to their characteristics as explained before.

Table 5 presents the average accuracy results obtained in the transductive phase. Specifically, it shows the overall results of the analyzed algorithms over the 55 used data sets with 10, 20, 30 and 40 % of labeled data, respectively. For those methods that work with different classifiers, we have tested various base classifiers, specifying them between brackets. **Acc** presents the average accuracy obtained. The algorithms are ordered from the best to the worst accuracy obtained. Even though kappa results are not reported in the paper, we want to check the gain (or loss) of each method in the established ranking of algorithms (in terms of accuracy measure) when the kappa measure is taken into account.

For this purpose, ΔK shows the oscillation of each method in the classification order established with accuracy with respect to the position obtained with kappa. This information reveals whether or not a certain algorithm benefits from random hits in comparison with the rest of the methods. Complete kappa results can be found on the associated Web site. Furthermore, in this table, we highlight those methods whose performance is within 5 % of the range between the best and the worst method, that is, $value_{best} - (0.05 \cdot (value_{best} - value_{worst}))$. We use boldface for the accuracy measure and italic for kappa. They should

Table 5 Average results obtained by self-labeled methods in the transductive phase

10%			20%			30%			40%		
	Acc	△K		Acc	△K		Acc	△K		Acc	△K
Democratic-Co	0.7305	-2	***TriTraining (C45)***	0.7543	0	***Democratic-Co***	0.7714	0	***Democratic-Co***	0.7794	-1
TriTraining (C45)	0.7256	+1	***Democratic-Co***	0.7534	-1	***TriTraining (C45)***	0.7689	-1	***TriTraining (C45)***	0.7777	-2
Co-Bagging (C45)	0.7235	+1	***Co-Bagging (C45)***	0.7530	-1	***Co-Bagging (C45)***	0.7676	-1	***Co-Training (SMO)***	0.7771	+2
SNNRCE	0.7213	-5	*Self-Training (C45)*	0.7466	-1	*Co-Training (SMO)*	0.7635	+2	Co-Training (C45)	0.7725	-4
Self-Training (C45)	0.7184	-2	SNNRCE	0.7446	-6	Co-Training (C45)	0.7630	-2	Self-Training (C45)	0.7713	-2
SETRED	0.7178	+2	Co-Training (C45)	0.7442	0	Self-Training (C45)	0.7623	+1	*Self-Training (SMO)*	0.7710	+3
TriTraining (KNN)	0.7147	-4	*Co-Training (SMO)*	0.7417	+5	SNNRCE	0.7576	-3	SNNRCE	0.7689	-2
Self-Training (KNN)	0.7142	+2	SETRED	0.7409	0	Self-Training (SMO)	0.7543	+2	Co-Bagging (SMO)	0.7676	+3
CoForest	0.7110	+1	Co-Bagging (KNN)	0.7378	-8	SETRED	0.7530	-3	TriTraining (SMO)	0.7655	+3
Co-Bagging (KNN)	0.7106	-3	TriTraining (KNN)	0.7376	-4	Co-Bagging (SMO)	0.7530	+2	DE-TriTraining (C45)	0.7626	-5
Co-Training (C45)	0.7084	-1	Self-Training (KNN)	0.7371	-1	Self-Training (KNN)	0.7520	-2	SETRED	0.7609	-1
Co-Training (SMO)	0.7070	+7	CoForest	0.7354	-1	DE-TriTraining (C45)	0.7501	-3	Co-Bagging (KNN)	0.7594	-4
DE-TriTraining (C45)	0.7053	-5	Co-Bagging (SMO)	0.7336	+4	Co-Bagging (KNN)	0.7495	-4	DE-TriTraining (SMO)	0.7592	+3
DE-TriTraining (KNN)	0.7007	-7	DE-TriTraining (C45)	0.7333	-7	DE-TriTraining (SMO)	0.7490	+3	Self-Training (KNN)	0.7591	0
DE-TriTraining (SMO)	0.7005	-2	DE-TriTraining (SMO)	0.7316	-1	CoForest	0.7488	+1	CoForest	0.7585	+2
Self-Training (SMO)	0.6967	+6	Self-Training (SMO)	0.7286	+9	TriTraining (KNN)	0.7477	0	Co-Bagging (C45)	0.7574	+5
Co-Training (NB)	0.6955	+1	TriTraining (SMO)	0.7283	+7	TriTraining (SMO)	0.7476	+8	TriTraining (KNN)	0.7555	0
Co-Training (KNN)	0.6923	-1	DE-TriTraining (KNN)	0.7263	-5	DE-TriTraining (KNN)	0.7407	-5	DE-TriTraining (KNN)	0.7537	0
TriTraining (NB)	0.6919	+4	Co-Training (KNN)	0.7239	-3	Co-Training (KNN)	0.7392	-2	Co-Training (KNN)	0.7473	-2
Co-Bagging (SMO)	0.6901	-2	TriTraining (NB)	0.7209	+5	TriTraining (NB)	0.7367	+2	Rel-Rasco (C45)	0.7471	+1
Co-Bagging (NB)	0.6874	+1	Co-Bagging (NB)	0.7195	+2	Co-Bagging (NB)	0.7328	+1	Rasco (C45)	0.7454	-2
DE-TriTraining (NB)	0.6865	-2	Co-Training (NB)	0.7183	+2	Rasco (C45)	0.7289	-2	Co-Bagging (NB)	0.7364	-2
ADE-CoForest	0.6858	-5	DE-TriTraining (NB)	0.7116	-2	Co-Training (NB)	0.7287	+1	ADE-CoForest	0.7363	-8
TriTraining (SMO)	0.6856	+1	ADE-CoForest	0.7103	-2	Rel-Rasco (C45)	0.7264	-5	Co-Training (NB)	0.7353	-1
APSSC	0.6762	+11	APSSC	0.7008	+7	ADE-CoForest	0.7247	-5	TriTraining (NB)	0.7342	+3
Rel-Rasco (NB)	0.6684	0	Self-Training (NB)	0.6943	+2	DE-TriTraining (NB)	0.7236	-2	DE-TriTraining (NB)	0.7309	0
CLCC	0.6683	-6	Rel-Rasco (NB)	0.6684	0	Rasco (NB)	0.7207	0	Rel-Rasco (NB)	0.7219	-1
Rasco (NB)	0.6673	+1	CLCC	0.6683	-5	Rel-Rasco (NB)	0.7203	+3	Rasco (NB)	0.7217	+1
Self-Training (NB)	0.6651	+4	Rasco (NB)	0.6673	+1	APSSC	0.7139	+10	APSSC	0.7216	+9
Rel-Rasco (C45)	0.6510	+1	Rel-Rasco (C45)	0.6517	-1	Self-Training (NB)	0.7099	+4	Rasco (SMO)	0.7208	0
Rasco (C45)	0.6492	-1	Rasco (C45)	0.6482	-1	Rasco (SMO)	0.7039	0	Self-Training (NB)	0.7185	+2
Rel-Rasco (SMO)	0.6285	+2	Rasco (SMO)	0.6343	+3	Rel-Rasco (SMO)	0.7008	0	Rel-Rasco (SMO)	0.7176	0
Rasco (SMO)	0.6284	+2	Rel-Rasco (SMO)	0.6316	+3	CLCC	0.6856	-2	Rel-Rasco (KNN)	0.6933	0
Rel-Rasco (KNN)	0.6091	0	Rel-Rasco (KNN)	0.6091	0	Rel-Rasco (KNN)	0.6720	+1	Rasco (KNN)	0.6926	0
Rasco (KNN)	0.6075	0	Rasco (KNN)	0.6075	0	Rasco (KNN)	0.6716	+1	CLCC	0.6900	0

be considered as a set of outstanding methods, regardless of their specific position in the table.

Figure 2 depicts a star plot representing the average transductive accuracy obtained in each method for the four labeled ratios considered. This star plot presents the performance as the distance from the center; therefore, a higher area determines the best average performance. This illustration allows us to easily visualize the average performance of the algorithms comparatively for each labeled ratio and in general. Figure 3 presents the same results in a bar chart aiming to compare the specific accuracy values.

Apart from the average results, we use the Wilcoxon test to statistically compare self-labeled methods in the different labeled ratios. Table 6 summarizes all possible comparisons involved in the Wilcoxon test between all the methods considered, considering accuracy results. Again, kappa results and the individual comparisons are exhibited on the aforementioned Web site, where a detailed report of statistical results can be found. This table presents, for each method in the rows, the number of self-labeled methods outperformed by using the Wilcoxon test under the column represented by the "+" symbol. The column with the "±" symbol indicates the number of wins and ties obtained by the method in the row. The maximum value for each column is highlighted by bold.

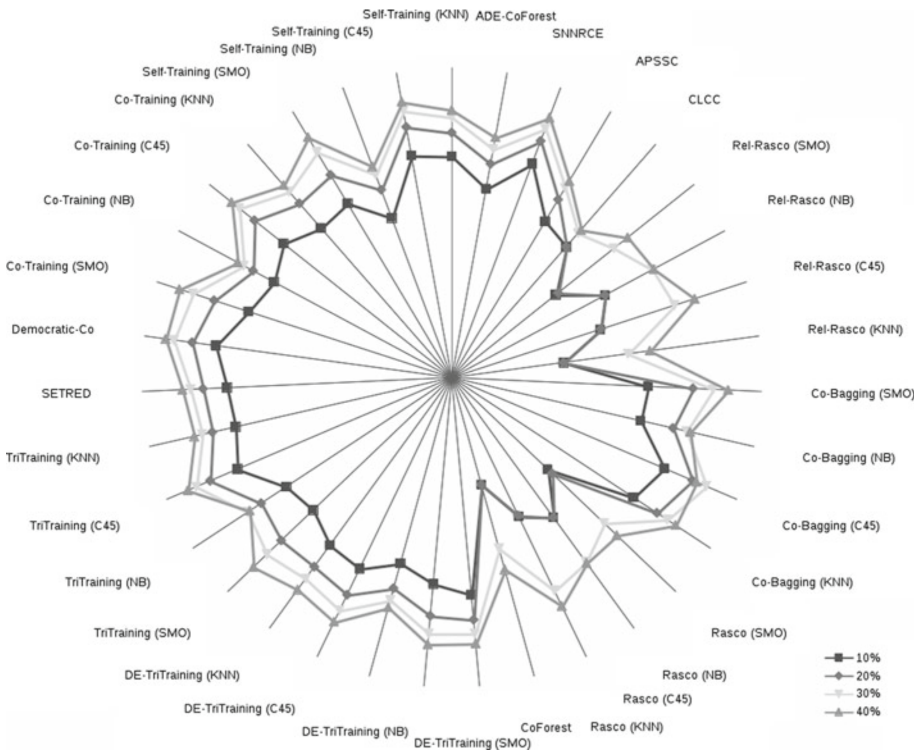


Fig. 2 Labeled ratio comparison (*star plot*): transductive phase

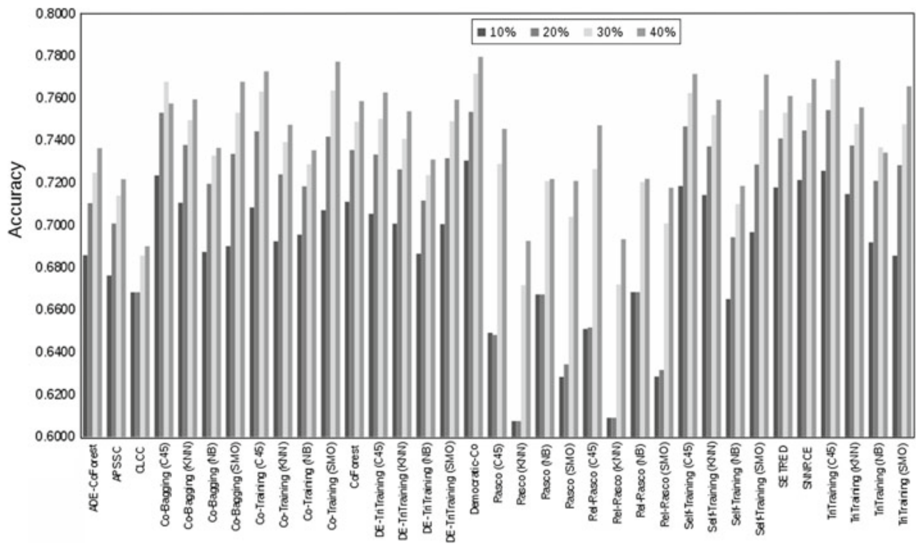


Fig. 3 Labeled ratio comparison (*bar chart*): transductive phase

Table 6 Wilcoxon test summary results: transductive accuracy

Method	10 %		20 %		30 %		40 %	
	+	±	+	±	+	±	+	±
Self-Training (KNN)	13	31	12	31	10	26	10	21
Self-Training (C45)	17	31	18	31	19	30	18	31
Self-Training (NB)	4	10	8	11	2	10	0	10
Self-Training (SMO)	9	29	11	30	14	33	17	33
Co-Training (KNN)	7	22	9	18	5	19	3	19
Co-Training (C45)	13	30	17	31	21	32	21	33
Co-Training (NB)	9	25	10	25	4	22	6	19
Co-Training (SMO)	12	31	15	**34**	26	**34**	24	**34**
Democratic-Co	**27**	**34**	**28**	**34**	27	**34**	26	**34**
SETRED	17	33	15	32	12	29	11	29
TriTraining (KNN)	15	31	12	28	8	22	6	19
TriTraining (C45)	25	**34**	26	**34**	**28**	**34**	**27**	**34**
TriTraining (NB)	10	28	13	28	10	27	8	23
TriTraining (SMO)	6	28	12	30	9	30	16	33
DE-TriTraining (KNN)	13	33	12	28	9	25	9	28
DE-TriTraining (C45)	14	30	14	28	15	28	15	30
DE-TriTraining (NB)	9	21	10	19	4	20	4	22
DE-TriTraining (SMO)	12	31	14	31	15	30	13	31
CoForest	21	**34**	20	**34**	21	**34**	24	**34**
Rasco (KNN)	0	1	0	1	0	2	0	6
Rasco (C45)	4	10	2	8	5	26	9	29
Rasco (NB)	4	14	4	8	3	14	1	10
Rasco (SMO)	2	3	2	6	2	13	4	17
Co-Bagging (KNN)	14	32	16	31	13	30	14	27
Co-Bagging (C45)	24	**34**	26	**34**	**28**	**34**	11	31
Co-Bagging (NB)	7	21	10	24	7	23	6	21
Co-Bagging (SMO)	8	31	13	33	14	30	20	33
Rel-Rasco (KNN)	0	1	0	1	0	2	0	6
Rel-Rasco (C45)	4	10	2	8	4	26	9	29
Rel-Rasco (NB)	5	13	4	8	3	14	1	10
Rel-Rasco (SMO)	2	4	2	6	2	10	2	14
CLCC	3	19	2	9	0	8	0	4
APSSC	4	19	9	16	3	15	1	14
SNNRCE	22	**34**	19	33	17	33	19	33
ADE-CoForest	11	31	11	29	6	25	6	28

Once the results are presented in the above tables and graphics, we outline some comments related to the properties observed, pointing out the best-performing methods in terms of transductive capabilities:

- Considering the influence of labeled ratio, Fig. 2 shows that, as could be expected, most of the algorithms obtain a regular increment on their accuracy transductive capabilities when the number of labeled data is increased. By contrast, we can observe how several methods are highly affected by the labeled ratio. For instance, multi-view methods obtain higher accuracy and kappa rankings when the labeled ratio is increased. In Table 5, we can also point out that two techniques are always at the top in accuracy and kappa rate independently of the labeled ratio: Democratic-Co and TriTraining (C45). They are also noteworthy as the methods that always obtain a transductive performance within 5 % of the range between the best and the worst method in both accuracy and kappa measures. Moreover, Co-Training (SMO) and Co-Bagging (C45) are considered as outstanding methods in most of the labeled ratios, mainly in the kappa measure. We can observe the validity of these statements in the Wilcoxon test, which confirms the averaged results obtained in accuracy and kappa rates.

- For those methods whose wrapper classifier can be set, we can find significant differences in their transductive behavior. In general, C4.5 offers the best transductive accuracy results in most of the techniques. In depth, classical self-training and co-training approaches obtain better results when C4.5 or SMO are used as base classifiers. Tri-Training also works with C4.5 and with KNN. The edited version of Tri-Training and Co-bagging presents a good synergy with C4.5. Finally, the use of Rasco and Rel-Rasco is more appropriate when NB or C4.5 is established as the base classifier.
- The classical Co-training with SMO or C4.5 as base classifier could be stressed as the best-performing method from the multi-view family. Rasco and Rel-Rasco are based on the idea of using random feature subspace (relevant or not) to construct different learned hypotheses. This idea performs well in several domains as shown in the corresponding papers. Nevertheless, to deal with a wide variety of data sets, such as in this experimental study, a more accurate feature selection methodology should be used to enhance their performance.
- Regarding single-view algorithms, several subfamilies deserve particular mention: In general, incremental approaches obtain the best results in accuracy and kappa rates. The results obtained by CoForest show it to be the best batch model. Moreover, CoForest shows that it is at least statistically similar to the rest of the methods in transductive capabilities. From amending approaches, we can highlight SNNRCE as one important method which is able to clearly outperform the Self-Training (KNN) method on which it is based in both measures.
- Usually, there is no significant difference ($\triangle$K) between the rankings obtained with accuracy and kappa rates, except for some concrete algorithms. For example, we can observe that DE-Training usually obtains a lower ranking with the kappa measure; this probably indicates that it benefits from random hits. In contrast, other algorithms, such as APSSC and Co-Training (SMO), improve their rankings when the kappa measure is used.

Furthermore, we perform an analysis of the results depending on the number of classes. On the Web site associated with this paper, we show the rankings obtained in accuracy and kappa for all the methods differentiating between binary and multi-class data sets. Figure 4 displays a summary representation of this study. It shows, for each method and labeled ratio, the differences in the ranking obtained in binary problems and the ranking achieved in multi-class data sets. Hence, positive bars indicate that the method performs better in multi-class problems, and negative bars show that the method obtains a higher ranking over binary domains. We can analyze several details from the results collected, which are as follows:

- When the transductive analysis is divided into binary and multi-class data sets, we find wide differences in the ranking previously obtained. Democratic-Co and TriTraining

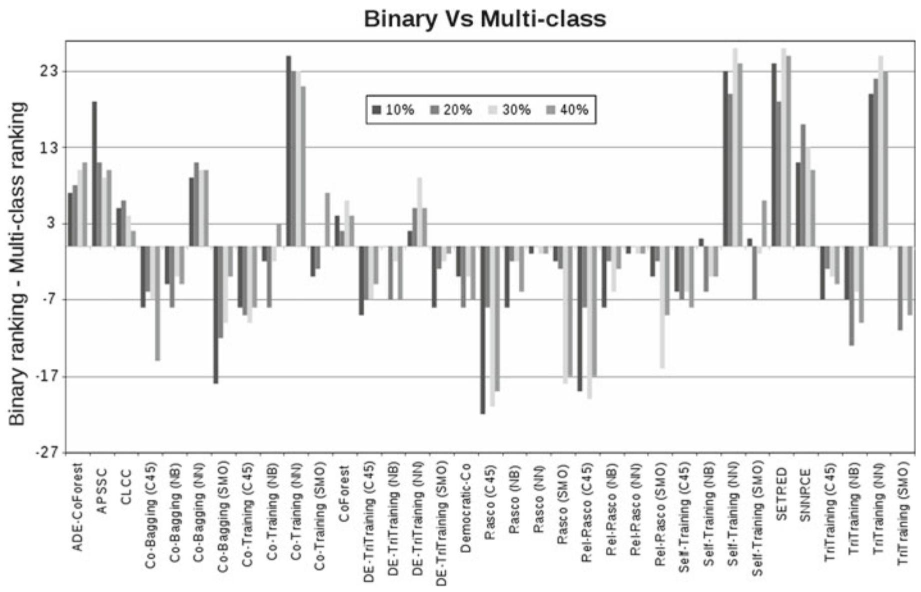


Fig. 4 Differences between rankings in binary and multi-class domains: transductive phase

(C45) continues to lead the ranking if we take into consideration only binary data sets. Nevertheless, in multi-class problems they are not the best-performing methods, although they maintain a good behavior.

- Single-classifier methods with an amending addition scheme, such as SETRED and SNNRCE, are noteworthy as the best-performing methods to deal with multi-class problems.
- Many differences appear in this study depending on the base classifier. In contrast to the previous analysis in which C4.5 was, in most cases, highlighted as the best base classifier, C4.5 is now stressed when tackling binary data sets, whereas for multi-class data sets, we can highlight the KNN rule as the most adequate base classifier for most of the self-labeled techniques. Specifically, Self-Training, Co-training, Co-Bagging and Tri-Training with KNN obtain a higher accuracy and kappa rates in comparison with the rest of the base classifiers.

5.2 Inductive results (test phase)

In contrast to transductive learning, the aim of inductive learning is to classify unknown examples. In this way, inductive learning proves the generalization capabilities of the self-labeled methods, checking whether the previous learned hypotheses are appropriate or not. Table 7 shows the average results obtained, and Figs. 5 and 6 illustrate the comparison between labeled ratios with a star plot and a bar graph, respectively. Finally, the summary of the results of applying the Wilcoxon test to all the techniques in test data is presented in Table 8.

These results allow us to highlight some differences in the generalization capabilities of the analyzed methods:

- Some methods present clear differences when dealing with the inductive phase. For instance, the amending model SNNRCE obtains a lesser generalization accuracy/kappa,

Table 7 Average results obtained by self-labeled methods in the inductive phase

10%			20%			30%			40%		
	Acc	△K		Acc	△K		Acc	△K		Acc	△K
Democratic-Co	0.7304	-2	***TriTraining (C4.5)***	0.7609	-1	***Democratic-Co***	0.7744	-1	***Democratic-Co***	0.7834	-1
TriTraining (C4.5)	0.7297	+1	***Co-Bagging (C45)***	0.7597	+1	***TriTraining (C4.5)***	0.7708	-2	**TriTraining (C4.5)**	0.7812	-3
Co-Bagging (C4.5)	0.7278	+1	**Democratic-Co**	0.7545	-3	***Co-Bagging (C4.5)***	0.7707	0	*Co-Training (SMO)*	0.7781	+2
Self-Training (C4.5)	0.7214	0	Self-Training (C4.5)	0.7526	-1	Self-Training (C4.5)	0.7652	-2	Self-Training (C4.5)	0.7754	-3
SETRED	0.7178	0	Co-Training (C4.5)	0.7524	-2	*Co-Training (SMO)*	0.7642	+4	Co-Training (C4.5)	0.7754	-3
TriTraining (KNN)	0.7176	-11	*Co-Training (SMO)*	0.7447	+3	Co-Training (C4.5)	0.7639	-3	*Co-Bagging (SMO)*	0.7738	+3
SNNRCE	0.7151	-2	Co-Bagging (KNN)	0.7431	-8	Co-Bagging (SMO)	0.7568	-1	*TriTraining (SMO)*	0.7707	+3
CoForest	0.7124	+1	*TriTraining (SMO)*	0.7414	+4	Self-Training (SMO)	0.7553	+1	Self-Training (SMO)	0.7700	+2
Self-Training (KNN)	0.7101	-3	SETRED	0.7406	-2	SETRED	0.7550	-2	Co-Bagging (C4.5)	0.7638	0
Co-Bagging (KNN)	0.7097	-3	DE-TriTraining (SMO)	0.7394	0	TriTraining (SMO)	0.7545	+5	SETRED	0.7628	-1
Co-Training (C4.5)	0.7088	-3	DE-TriTraining (C4.5)	0.7387	-6	Co-Bagging (KNN)	0.7536	-3	Co-Bagging (KNN)	0.7628	-7
Co-Training (SMO)	0.7069	+6	Co-Bagging (SMO)	0.7386	+4	Self-Training (KNN)	0.7530	0	DE-TriTraining (C4.5)	0.7605	-5
DE-TriTraining (C4.5)	0.7058	-5	CoForest	0.7372	-1	DE-TriTraining (SMO)	0.7527	+3	DE-TriTraining (SMO)	0.7593	+3
DE-TriTraining (SMO)	0.7019	-1	Self-Training (KNN)	0.7367	+1	DE-TriTraining (C4.5)	0.7500	-1	Rasco (C4.5)	0.7591	+2
DE-TriTraining (KNN)	0.7016	-7	SNNRCE	0.7350	-4	CoForest	0.7497	+2	Self-Training (KNN)	0.7586	0
TriTraining (SMO)	0.6988	+6	TriTraining (KNN)	0.7303	-6	SNNRCE	0.7474	-4	Rel-Rasco (C4.5)	0.7578	+2
Self-Training (SMO)	0.6969	+9	Co-Training (KNN)	0.7302	-3	Co-Training (KNN)	0.7454	-2	CoForest	0.7565	-2
Co-Training (KNN)	0.6958	-5	Self-Training (SMO)	0.7298	+9	DE-TriTraining (KNN)	0.7426	-8	SNNRCE	0.7545	-3
Co-Training (NB)	0.6949	0	DE-TriTraining (KNN)	0.7294	-4	TriTraining (KNN)	0.7412	-8	Co-Training (KNN)	0.7539	-1
TriTraining (NB)	0.6934	+4	Co-Bagging (NB)	0.7288	+4	TriTraining (NB)	0.7391	+4	DE-TriTraining (KNN)	0.7519	-4
Co-Bagging (SMO)	0.6920	0	TriTraining (NB)	0.7269	+9	Rasco (C4.5)	0.7389	-1	Rasco (SMO)	0.7475	+8
Co-Bagging (NB)	0.6892	+2	Co-Training (NB)	0.7219	+4	Rel-Rasco (C4.5)	0.7378	-2	TriTraining (KNN)	0.7463	-5
DE-TriTraining (NB)	0.6881	-1	DE-TriTraining (NB)	0.7199	-1	Co-Bagging (NB)	0.7360	+2	Rel-Rasco (SMO)	0.7454	+7
ADE-CoForest	0.6839	-4	ADE-CoForest	0.7122	-2	Co-Training (NB)	0.7316	-1	Co-Training (NB)	0.7387	-2
APSSC	0.6770	14	APSSC	0.7033	+4	Rasco (SMO)	0.7311	+7	Co-Bagging (NB)	0.7374	0
Rel-Rasco (NB)	0.6733	+1	Self-Training (NB)	0.6967	+1	Rel-Rasco (SMO)	0.7306	+9	ADE-CoForest	0.7360	-6
Rasco (NB)	0.6705	0	Rel-Rasco (NB)	0.6733	0	Rel-Rasco (NB)	0.7261	-2	TriTraining (NB)	0.7354	+4
CLCC	0.6685	-5	Rasco (NB)	0.6705	0	DE-TriTraining (NB)	0.7258	-2	DE-TriTraining (NB)	0.7332	0
Self-Training (NB)	0.6655	+3	CLCC	0.6685	-4	Rasco (NB)	0.7258	+1	Rasco (NB)	0.7268	0
Rasco (C4.5)	0.6546	-2	Rasco (C4.5)	0.6539	-2	ADE-CoForest	0.7242	-2	Rel-Rasco (NB)	0.7252	0
Rel-Rasco (C4.5)	0.6529	0	Rel-Rasco (C4.5)	0.6535	0	APSSC	0.7179	+8	APSSC	0.7246	+9
Rasco (SMO)	0.6480	+2	Rasco (SMO)	0.6525	+3	Self-Training (NB)	0.7122	+1	Rasco (KNN)	0.7232	-1
Rel-Rasco (SMO)	0.6473	+4	Rel-Rasco (SMO)	0.6494	+3	Rel-Rasco (KNN)	0.6990	0	Rel-Rasco (KNN)	0.7212	-1
Rasco (KNN)	0.6273	0	Rasco (KNN)	0.6273	0	Rasco (KNN)	0.6988	0	Self-Training (NB)	0.7180	+3
Rel-Rasco (KNN)	0.6231	0	Rel-Rasco (KNN)	0.6231	0	CLCC	0.6826	0	CLCC	0.6866	0

and it is outperformed by the other member of its family SETRED. It may suffer from an excessive elimination rule of candidate instances to be incorporated into the labeled set. On the other hand, classical self-training and co-training methods are shown to perform well in the inductive phase, and they are at the top in accuracy and kappa rate.

- TriTraining (C45) and Democratic-Co remain at the top of the rankings, established by the accuracy measure, in conjunction with Co-bagging (C4.5) which is a very competitive method in the test phase. In Table 7, we observe that the kappa ranking penalizes the Democratic-Co algorithm and considers other methods to be outstanding. For example, the classical Co-Training (SMO) is positioned as one of the most important methods when the kappa rate is taken into consideration. Wilcoxon test supports these ideas, showing that, in most labeled ratios, TriTraining (C45) is the best proposal to the detriment of Democratic-Co.
- It is worth mentioning that, in general, those methods that are based on C4.5 and SMO as base classifier(s) obtain a higher number of outperformed methods (+) and, consequently, a higher number of wins and ties (±) than the numbers obtained in the transductive phase. Hence, these methods present good generalization capabilities, and their use is recommended for inductive tasks.

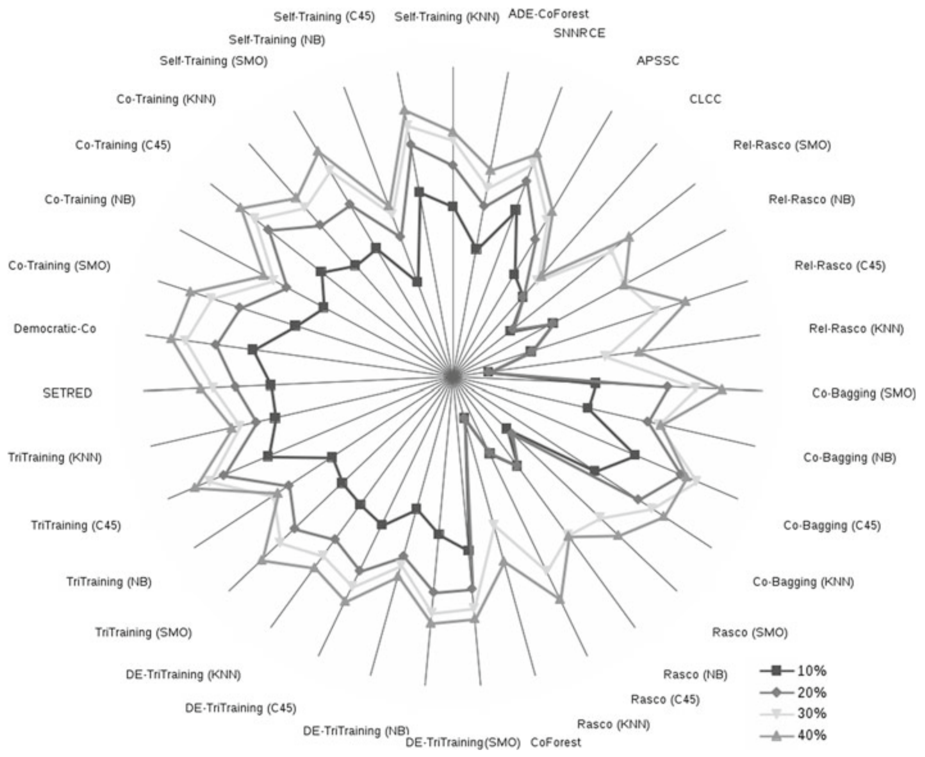


Fig. 5 Labeled ratio comparison (*star plot*): inductive phase

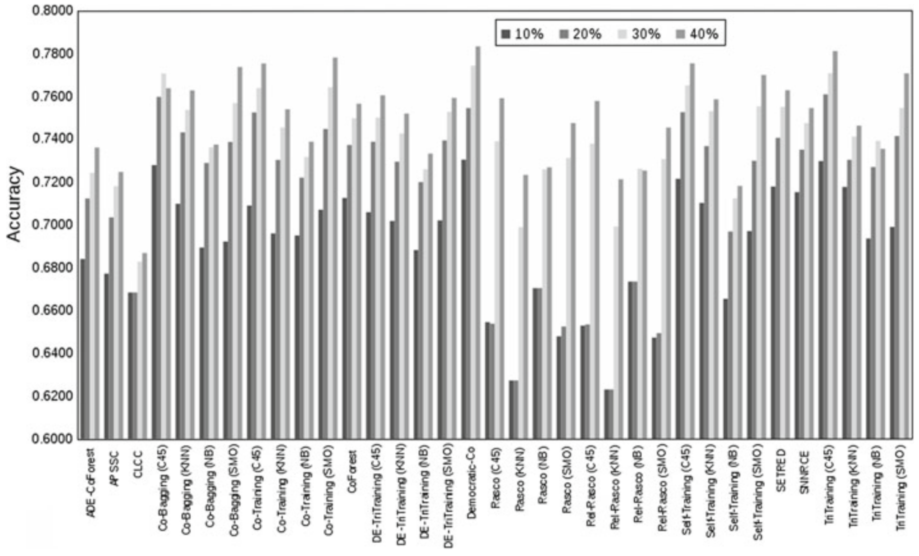


Fig. 6 Labeled ratio comparison (*bar chart*): inductive phase

Table 8 Wilcoxon test summary results: inductive accuracy

Method	10 %		20 %		30 %		40 %	
	+	±	+	±	+	±	+	±
Self-Training (KNN)	11	30	11	25	9	28	6	24
Self-Training (C4.5)	19	32	22	32	21	32	23	33
Self-Training (NB)	2	11	8	13	1	9	1	9
Self-Training (SMO)	9	29	10	28	11	**34**	17	32
Co-Training (KNN)	6	26	10	23	4	22	5	21
Co-Training (C4.5)	11	30	23	32	21	33	22	**34**
Co-Training (NB)	10	27	10	25	4	24	3	23
Co-Training (SMO)	11	32	14	33	21	**34**	25	**34**
Democratic-Co	26	**34**	26	**34**	25	**34**	26	**34**
SETRED	15	**34**	14	29	10	29	11	29
TriTraining (KNN)	13	**34**	10	23	3	20	2	16
TriTraining (C4.5)	**28**	**34**	**31**	**34**	24	**34**	**27**	**34**
TriTraining (NB)	10	29	11	28	9	28	4	20
TriTraining (SMO)	9	31	12	33	12	33	18	32
DE-TriTraining (KNN)	11	31	11	26	8	26	7	25
DE-TriTraining (C4.5)	12	29	13	29	11	28	12	28
DE-TriTraining (NB)	9	25	10	25	3	19	2	20
DE-TriTraining (SMO)	11	30	17	31	16	32	14	29
CoForest	19	**34**	18	**34**	20	**34**	20	**34**
Rasco (KNN)	0	1	0	1	0	5	0	11
Rasco (C4.5)	2	10	2	8	4	28	11	29
Rasco (NB)	3	12	2	7	4	17	2	15
Rasco (SMO)	2	9	2	8	3	21	3	23
Co-Bagging (KNN)	12	31	18	31	17	32	16	29
Co-Bagging (C4.5)	27	**34**	28	**34**	**27**	**34**	15	32
Co-Bagging (NB)	8	24	10	28	4	25	4	20
Co-Bagging (SMO)	7	31	16	33	15	33	24	**34**
Rel-Rasco (KNN)	0	1	0	1	0	3	0	10
Rel-Rasco (C4.5)	2	10	2	8	4	28	10	29
Rel-Rasco (NB)	3	14	3	8	4	17	1	14
Rel-Rasco (SMO)	2	10	2	8	3	21	3	22
CLCC	2	20	2	9	0	2	0	2
APSSC	2	20	9	16	2	17	1	15
SNNRCE	16	**34**	11	27	5	23	6	22
ADE-CoForest	7	30	10	28	3	23	4	27

- In this phase, the ratio of labeled instances is a more relevant issue for the obtained performance of all the algorithms. In comparison with transductive results, there are higher differences between the results obtained for each method, comparing, as extreme cases, 10 and 40 % ratios. As we can see in Fig. 5, the star plot shows abrupt changes, mainly from 10 to 20 % of labeled instances.

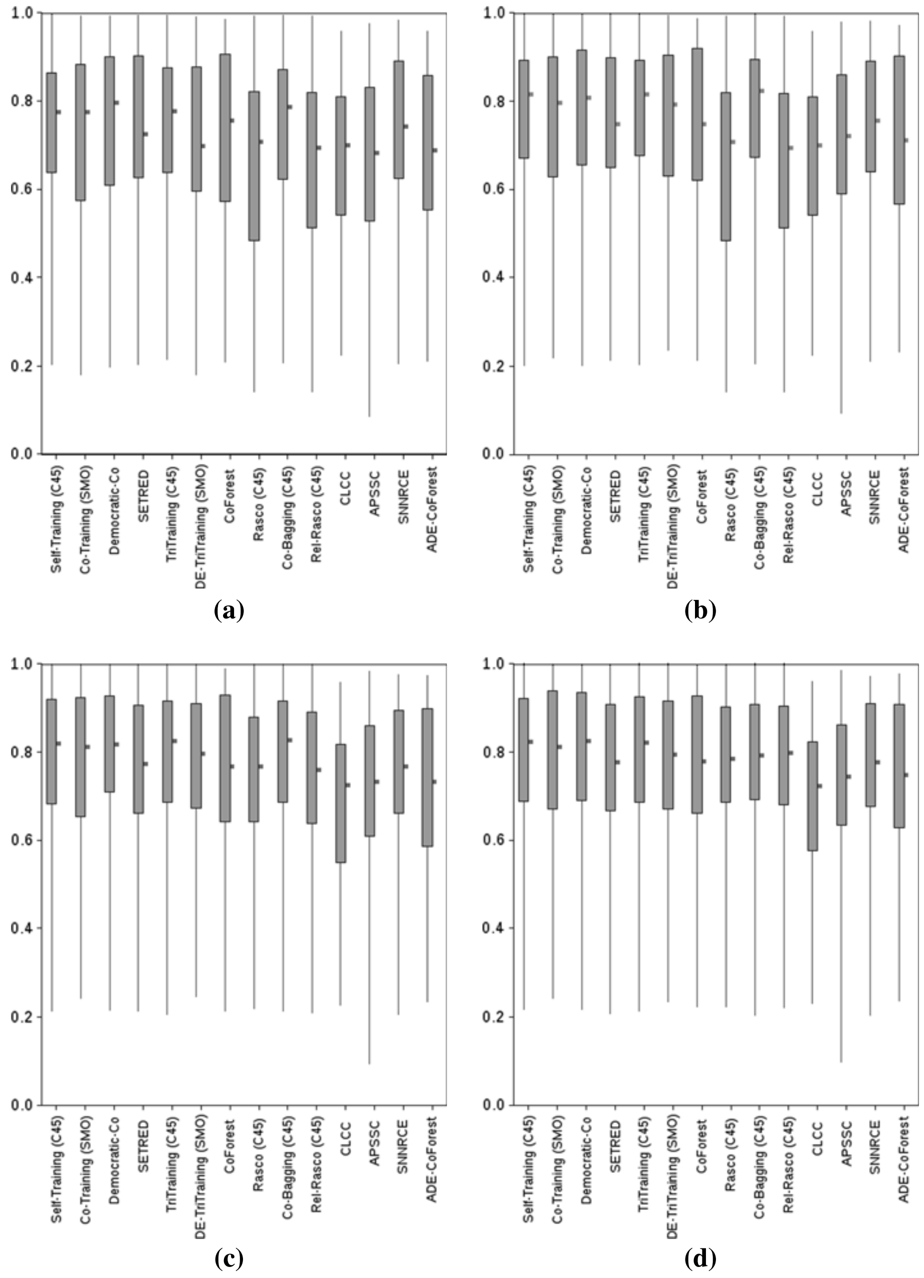


Fig. 7 *Box plot* accuracy inductive phase. **a** 10 % labeled ratio, **b** 20 % labeled ratio, **c** 30 % labeled ratio, **d** 40 % labeled ratio

Aside from these tables, Fig. 7 collects box plot representations for each labeled ratio. In this figure, we select a subset of methods whose performances are of interest. In particular, we select the most promising alternatives based on the best base classifier choice from the previous study. Box plots have been shown to be an effective tool in data reporting because

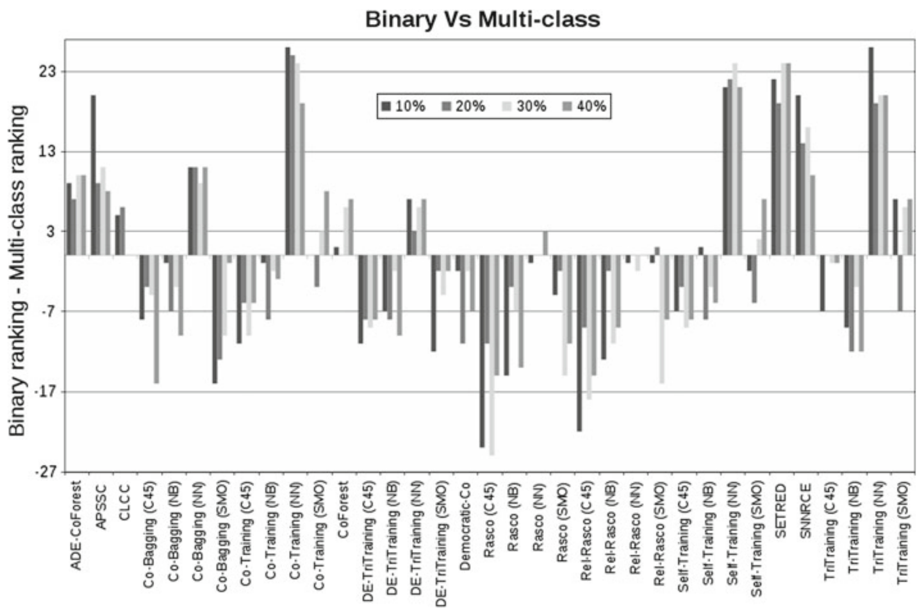


Fig. 8 Differences between rankings in binary and multi-class domains: inductive phase

they allow the graphical representation of the performance of algorithms, indicating important characteristics such as the median, extreme values and spread of values about the median in the form of quartiles (Q1 and Q3).

- The box plots show which results are more robust in the sense that the boxes are more compact. In general, Self-training and Co-bagging present the smallest box plots in all the labeled ratios. By contrast, other methods such as CoForest and Rasco are less robust, as was previously reflected in the average results and Wilcoxon test.
- Median results also help us to identify promising algorithms which perform well in many domains. It is interesting that Co-Bagging is shown to be the best median value in most of the labeled ratios in comparison with Democratic-Co and TriTraining (C4.5), which were pointed out as the best-performing methods.

Again, we differentiate between binary and multi-class data sets. Figure 8 depicts a summary graphic. The Web site associated with this paper presents the complete results. Observing these results, we can make several comments.

- Over binary data sets, we find an significant increment in the ranking obtained by Rasco (C45) and Rel-Rasco (C45). In general, C4.5 models lead the ranking established in accuracy and kappa rates.
- When only multi-class data sets are taken into consideration, as in the transductive phase, we also observe that TriTraining, SETRED and Self-Training based on KNN reach the top positions.

5.3 Global analysis

This section provides a global perspective on the obtained results. As a summary, we want to outline several remarks attending to the previous studies:

Table 9 Average Friedman rankings of outstanding algorithms in transductive and inductive phases

Algorithm	Ranking transductive	Algorithm	Ranking inductive
TriTraining (C45)	3.9477	TriTraining (C45)	3.9455
Democratic-Co	4.1205	Democratic-Co	4.2295
Co-Bagging (C45)	4.2409	Co-Bagging (C45)	4.2727
Co-Training (SMO)	4.3409	Co-Training (SMO)	4.4159
Self-Training (C45)	4.6136	Self-Training (C45)	4.4727
Self-Training (SMO)	4.7636	TriTraining (SMO)	4.8023
Co-Bagging (SMO)	4.9432	Co-Bagging (SMO)	4.8955
TriTraining (SMO)	5.0295	Self-Training (SMO)	4.9659

- In general, those self-labeled methods based on C4.5 or SMO have been shown to perform well in binary data sets. In contrast, the KNN rule is shown to work better with multi-class domains. Regarding the NB classifier, the continuous distribution version used [78] has not reported competitive results in comparison with the rest of classifiers. It will probably obtain a better performance if a discretization process is used for continuous attributes [85,86].
- According to the type of data sets, we can also claim that single-classifier models usually outperform those that are based on multi-classifiers when dealing with multi-class problems. By contrast, multi-classifier methods show a better behavior than single-classifier in tackling binary data sets.
- Eight self-labeled methods have been emphasized as outstanding methods according to the accuracy/kappa obtained, at least in one of the labeled ratios of the transductive or inductive phases: TriTraining (C45), TriTraining (SMO), Democratic-Co, Co-Bagging (C45), Co-Bagging (SMO), Co-Training (SMO), SelfTraining (C45) and SelfTraining (SMO).

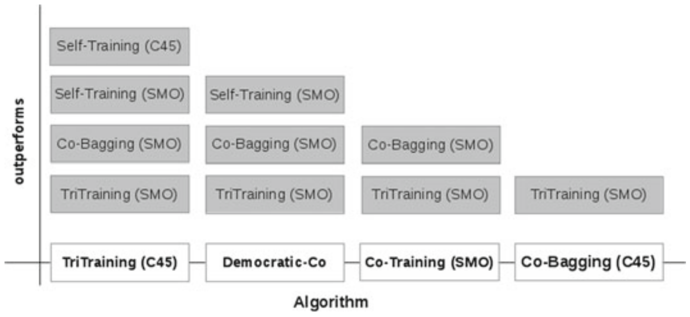
Now, we focus our attention on these eight outstanding methods, performing a multiple comparison statistical test between them. To analyze the global capabilities of these algorithms independently of labeled ratios, the statistical test is conducted with all of the 220 data sets (55 data sets $\times$ 4 labeled rates). Table 9 presents the results of the Friedman test, which has been carried out considering the accuracy results obtained in the transductive and inductive phases. In this table, the computed Friedman rankings are presented, representing the associated effectiveness of each method in both phases. Algorithms are ordered from the best (lowest) to the worst (highest) ranking.

Table 10 provides information about the state of retainment or rejection of all the hypotheses, comparing outstanding methods in transductive and inductive phases. They show the adjusted p value (APV) with Bergmann–Hommel's procedure for the 28 established comparisons. Each table is set out so that each row contains a hypothesis in which the first algorithm mentioned (left side of the comparison) outperforms the second one (right side). The hypotheses are ordered from the most to the least significant differences. Those APVs highlighted in bold correspond to hypotheses whose left method outperforms the right method, at an $\alpha = 0.1$ level of significance.

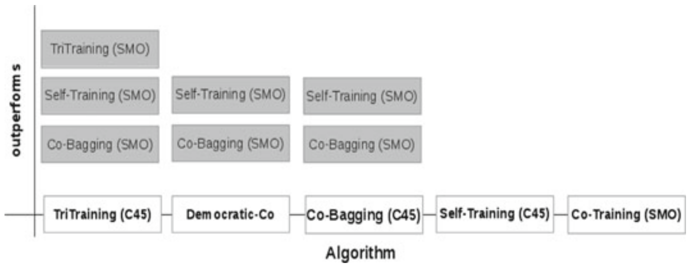
Figure 9 outlines two graphs for transductive and inductive statistical results, respectively. The x-axis presents those methods that are not outperformed by any other algorithm. For each of them, the y-axis collects the methods that they outperform according to the Bergmann–Hommel test.

Table 10 Multiple comparison test: Bergmann–Hommel's APVs

Hypothesis Transductive	Bergmann's APV	Hypothesis Inductive	Bergmann's APV
TriTraining (C45) vs TriTraining (SMO)	**0.0001**	TriTraining (C45) vs Self-Training (SMO)	**0.0003**
TriTraining (C45) vs Co-Bagging (SMO)	**0.0004**	TriTraining (C45) vs Co-Bagging (SMO)	**0.0010**
Democratic-Co vs TriTraining (SMO)	**0.0021**	TriTraining (C45) vs TriTraining (SMO)	**0.0039**
Democratic-Co vs Co-Bagging (SMO)	**0.0064**	Democratic-Co vs Self-Training (SMO)	**0.0339**
TriTraining (C45) vs Self-Training (SMO)	**0.0076**	Co-Bagging (C45) vs Self-Training (SMO)	**0.0480**
Co-Bagging (C45) vs TriTraining (SMO)	**0.0117**	Democratic-Co vs Co-Bagging (SMO)	**0.0653**
Co-Bagging (C45) vs Co-Bagging (SMO)	**0.0290**	Co-Bagging (C45) vs Co-Bagging (SMO)	**0.0843**
Co-Training (SMO) vs TriTraining (SMO)	**0.0415**	Democratic-Co vs TriTraining (SMO)	0.1562
TriTraining (C45) vs Self-Training (C45)	**0.0566**	Co-Training (SMO) vs Self-Training (SMO)	0.2408
Democratic-Co vs Self-Training (SMO)	**0.0648**	Co-Bagging (C45) vs TriTraining (SMO)	0.2408
Co-Training (SMO) vs Co-Bagging (SMO)	**0.0892**	TriTraining (C45) vs Self-Training (C45)	0.3116
Co-Bagging (C45) vs Self-Training (SMO)	0.2017	Self-Training (C45) vs Self-Training (SMO)	0.4166
Democratic-Co vs Self-Training (C45)	0.3124	Co-Training (SMO) vs Co-Bagging (SMO)	0.4166
Co-Training (SMO) vs Self-Training (SMO)	0.4921	TriTraining (C45) vs Co-Training (SMO)	0.4166
Self-Training (C45) vs TriTraining (SMO)	0.8993	Self-Training (C45) vs Co-Bagging (SMO)	0.4218
TriTraining (C45) vs Co-Training (SMO)	0.8993	Co-Training (SMO) vs TriTraining (SMO)	0.6865
Co-Bagging (C45) vs Self-Training (C45)	0.8993	Self-Training (C45) vs TriTraining (SMO)	0.7912
Self-Training (C45) vs Co-Bagging (SMO)	0.9494	TriTraining (C45) vs Co-Bagging (C45)	1.0000
TriTraining (C45) vs Co-Bagging (C45)	1.0000	TriTraining (C45) vs Democratic-Co	1.0000
Co-Training (SMO) vs Self-Training (C45)	1.0000	Democratic-Co vs Self-Training (C45)	1.0000
Self-Training (SMO) vs TriTraining (SMO)	1.0000	Co-Bagging (C45) vs Self-Training (C45)	1.0000
Democratic-Co vs Co-Training (SMO)	1.0000	Democratic-Co vs Co-Training (SMO)	1.0000
Self-Training (SMO) vs Co-Bagging (SMO)	1.0000	TriTraining (SMO) vs Self-Training (SMO)	1.0000
TriTraining (C45) vs Democratic-Co	1.0000	Co-Bagging (C45) vs Co-Training (SMO)	1.0000
Self-Training (C45) vs Self-Training (SMO)	1.0000	TriTraining (SMO) vs Co-Bagging (SMO)	1.0000
Democratic-Co vs Co-Bagging (C45)	1.0000	Co-Bagging (SMO) vs Self-Training (SMO)	1.0000
Co-Bagging (C45) vs Co-Training (SMO)	1.0000	Self-Training (C45) vs Co-Training (SMO)	1.0000
Co-Bagging (SMO) vs TriTraining (SMO)	1.0000	Democratic-Co vs Co-Bagging (C45)	1.0000



(a)



(b)

Fig. 9 Graphical comparison with Bergmann–Hommel's test. **a** Transductive results, **b** inductive results

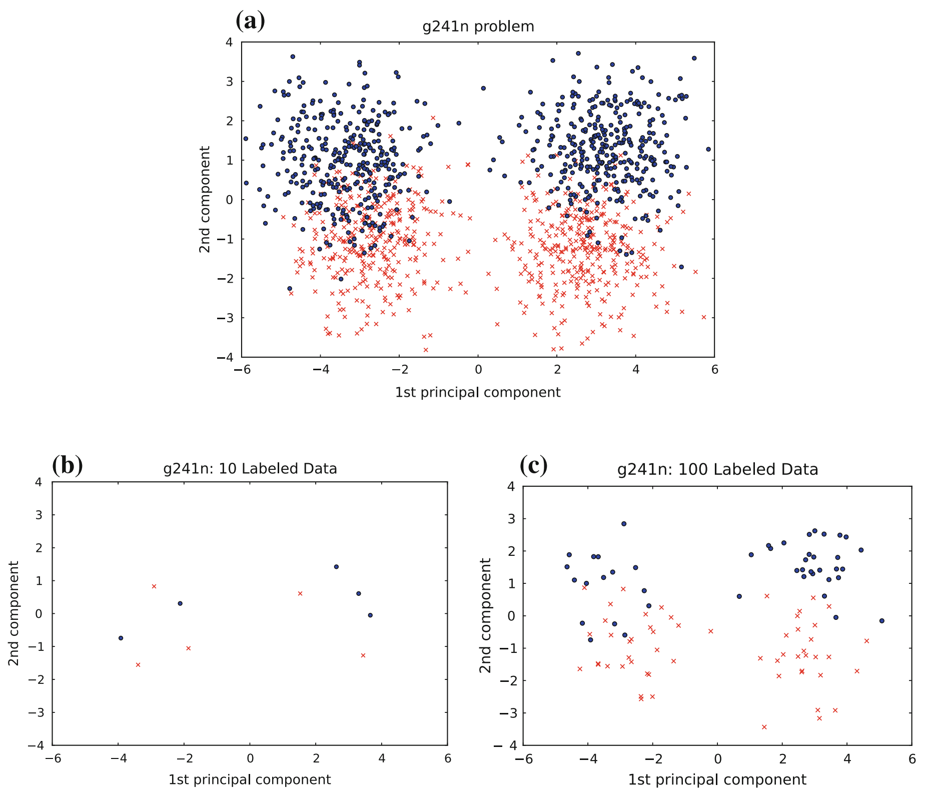


Fig. 10 Two-dimensional projections of g241n. *Red crosses* class +1, *blue circles* class −1. **a** problem, **b** g24ln: labeled data, **c** g24ln: 100 labeled data (color figure online)

Thus, observing the multiple comparison test and Fig. 9, we can highlight TriTraining (C45), Democratic-Co, Co-Bagging (C45) and Co-Training (SMO) as methods that appear in the x-axis in both Fig. 9a, b. Hence, they are not statistically outperformed by any algorithm. Self-Training (C45) is also highlighted as a nonoutperformed method in the inductive phase.

5.4 Experiments on high-dimensional data sets with small labeled ratio

The aim of this section is to check the behavior of self-labeling methods when they deal with high-dimensional data and a reduced labeled ratio. To do this, we focus our attention on four of the best methods highlighted above: Democratic-Co, TriTraining (C4.5), Co-Bagging(C4.5) and CoTraining(SMO). The used data sets were provided by Chapelle in [4], and their main characteristics were described in Sect. 4.2. To illustrate the complexity of these problems, Fig. 10 depicts an example of one partition of the g241n problem. This graph presents a two-dimensional projection (obtained with PCA [87]) of the problem and the 10 and 100 labeled data points used with self-labeled methods.

Tables 11 and 12 show the accuracy results obtained in the transductive and inductive phases with 10 and 100 labeled data, respectively. To measure the goodness of self-labeled techniques with this kind of data, we compare their results with the obtained with the base classifiers. Therefore, C4.5, KNN, SMO and NB have been trained with the available labeled

Table 11 High-dimensional data sets: self-labeled performance with ten labeled data

Data sets	Democratic-Co		TriTraining (C4.5)		Co-Bagging (C4.5)		Co-Training (SMO)	
	TRS	TST	TRS	TST	TRS	TST	TRS	TST
bci	0.5014	0.4875	0.5134	0.5050	0.5043	0.5100	0.5146	0.5250
coil	0.8328	0.8333	0.7864	0.7753	0.7792	0.7727	0.8239	0.8300
coil2	0.8050	0.8027	0.6897	0.7040	0.7139	0.7173	0.7601	0.7673
digit1	0.5945	0.6020	0.5337	0.5273	0.5261	0.5113	0.7372	0.7520
g241c	0.5290	0.5220	0.5169	0.5040	0.5213	0.5007	0.5925	0.6067
g241n	0.5043	0.5020	0.5029	0.5033	0.4990	0.4993	0.5340	0.5253
secstr	0.5719	0.5718	0.5097	0.5085	0.5298	0.5283	0.5281	0.5141
text	0.5604	0.5533	0.5272	0.5167	0.5190	0.5180	0.4993	0.5000
usps	0.8050	0.8027	0.6897	0.7040	0.7139	0.7173	0.7601	0.7673
Average	0.6338	0.6308	0.5855	0.5831	0.5896	0.5861	0.6389	0.6431

Table 12 High-dimensional data sets: self-labeled performance with 100 labeled data

Data sets	Democratic-Co		TriTraining (C4.5)		Co-Bagging (C4.5)		Co-Training (SMO)	
	TRS	TST	TRS	TST	TRS	TST	TRS	TST
bci	0.5027	0.5450	0.5588	0.5625	0.5604	0.5550	0.6573	0.6500
coil	0.8635	0.8773	0.8372	0.8393	0.8439	0.8480	0.9110	0.9047
coil2	0.8557	0.8413	0.7972	0.8033	0.8064	0.8333	0.8063	0.7927
digit1	0.9370	0.9347	0.8208	0.8600	0.8072	0.8127	0.9158	0.9173
g241c	0.6033	0.5213	0.5689	0.5413	0.5685	0.5660	0.7334	0.7453
g241n	0.5420	0.5053	0.5792	0.6067	0.5696	0.5733	0.7320	0.7313
secstr	0.5917	0.5915	0.5436	0.5421	0.5573	0.5500	0.5476	0.5515
text	0.6608	0.6667	0.6728	0.6800	0.6920	0.7333	0.6797	0.6735
usps	0.8557	0.8413	0.8184	0.8000	0.7904	0.7913	0.8063	0.7927
Average	0.7125	0.7027	0.6885	0.6928	0.6884	0.6959	0.7544	0.7510

examples to predict the class of the rest of unlabeled ones. Note that the information used for these techniques corresponds to the initial stage of all the self-labeled schemes. However, it is also known that, depending on the problem, unlabeled data can lead to worse performance [1]; hence, the inclusion of these baselines shows whether self-labeled techniques are appropriate for these high-dimensional problems. Results are presented in Tables 13 and 14.

In these tables we can appreciate that self-labeled techniques do not fit adequately to these kinds of problems. They are not able to significantly outperform baseline techniques which in some cases result in a better performance. When only ten labeled points are used, base classifiers perform equal or better than most of the self-labeled techniques. If the number of labeled points is increased to 100, we observe that some self-labeled techniques, such as TriTraining (C4.5) and Co-Bagging (C4.5), perform better than their base classifier. However, they are not really competitive with the results obtained with KNN or SMO with 100 labeled points. Figure 11 illustrates an example of the evolution of the transductive and inductive accuracy of Democratic-Co during the self-labeling process. With a self-labeling approach, it is expected that as the iterations go by, the accuracy should be increased. Nevertheless, in

Table 13 High-dimensional data sets: baselines performance with ten labeled data

Datasets	C4.5		KNN		SMO		NB	
	TRS	TST	TRS	TST	TRS	TST	TRS	TST
bci	0.5189	0.5175	0.4889	0.4725	0.5194	0.5000	0.4966	0.5000
coil	0.7945	0.7913	0.7938	0.7900	0.8398	0.8413	0.6921	0.6913
coil2	0.6997	0.7107	0.7767	0.7867	0.7794	0.7867	0.7676	0.7673
digit1	0.5353	0.5220	0.7738	0.7773	0.6889	0.6673	0.6748	0.6787
g241c	0.5175	0.4973	0.5466	0.5653	0.6020	0.6140	0.5446	0.5587
g241n	0.5160	0.5187	0.5431	0.5333	0.5091	0.5060	0.5048	0.5000
secstr	0.5209	0.5215	0.5121	0.5127	0.5155	0.5155	0.5232	0.5220
text	0.5034	0.5064	0.5143	0.5102	0.5201	0.5167	0.4936	0.4986
usps	0.6997	0.7107	0.7767	0.7867	0.7794	0.7867	0.7676	0.7673
Average	0.5895	0.5885	0.6362	0.6372	0.6393	0.6371	0.6072	0.6093

Table 14 High-dimensional data sets: baselines performance with 100 labeled data

Datasets	C4.5		KNN		SMO		NB	
	TRS	TST	TRS	TST	TRS	TST	TRS	TST
bci	0.5569	0.5525	0.5204	0.5600	0.6581	0.6500	0.5212	0.5275
coil	0.8226	0.8220	0.9422	0.9387	0.9182	0.9113	0.7684	0.7653
coil2	0.7762	0.7860	0.9245	0.9160	0.8386	0.8300	0.8624	0.8593
digit1	0.7726	0.7800	0.9361	0.9373	0.9146	0.9080	0.9365	0.9453
g241c	0.5446	0.5433	0.5919	0.5973	0.7405	0.7533	0.7202	0.7300
g241n	0.5377	0.5267	0.6286	0.6380	0.7371	0.7400	0.6877	0.6780
secstr	0.5298	0.5284	0.5156	0.5152	0.5240	0.5257	0.5340	0.5346
text	0.5054	0.5049	0.5064	0.5177	0.5196	0.5206	0.5003	0.4945
usps	0.7762	0.7860	0.9245	0.9160	0.8386	0.8300	0.8624	0.8593
Average	0.6469	0.6478	0.7211	0.7262	0.7433	0.7410	0.7103	0.7104

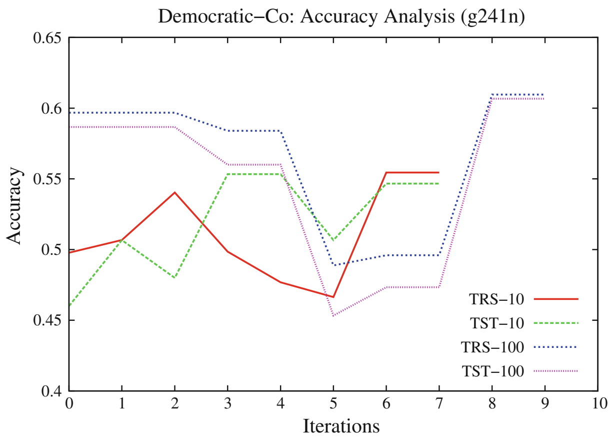


Fig. 11 Transductive (TRS) and inductive (TST) accuracy of Democratic-Co in the g241n problem: 10 and 100

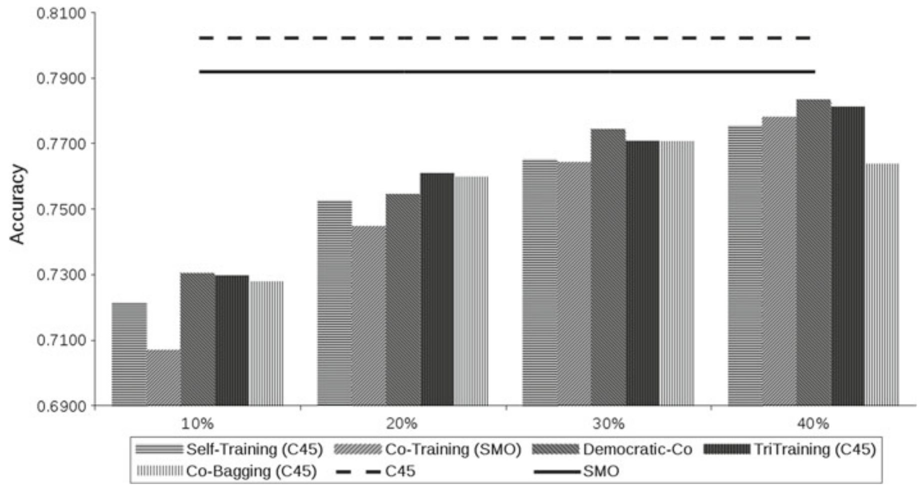


Fig. 12 Differences in average test results between outstanding models and C4.5 and SMO

these problems we find that this expectation is not satisfied in most of cases. In the plot we see that in intermediate iterations the accuracy is deteriorated. It means that the estimation of most confident examples is erroneous and much more complicated to be obtained in these domains. Therefore, these results have exemplified the difficulty of these problems when a very reduced number of labeled data are used. In our opinion, more research is required to provide to the self-labeled techniques the ability of deal with high-dimensional problem with a very reduced labeled ratio.

5.5 How far removed is semi-supervised learning from the traditional supervised learning paradigm?

This section is devoted to checking how far removed is the accuracy obtained with self-labeled methods, in the SSL context, in comparison with supervised learning. It is clear that SSL implies a more complex problem than the standard supervised learning problem. In SSL, algorithms are provided with a lesser number of labeled examples to learn a correct hypothesis. Specifically, the performance obtained with self-labeled methods is theoretically upper-bounded by the traditional supervised learning algorithms used as base classifiers.

To contrast this idea, we compare the inductive (test) results obtained with the best methods highlighted in the previous section with C4.5 and SMO classifiers. These classifiers are trained with completely labeled training sets, using the same tenfold cross-validation scheme to compute the accuracy test results. The complete results of this study are available on the associated Web site.

Figure 12 draws a graphical comparison between outstanding inductive methods, C4.5 and SMO classifiers. For each self-labeled method, the average result obtained is shown in each labeled ratio. The average result of C4.5 and SMO is represented as a line $y = AverageResult$, to show the differences between self-labeled methods and these classifiers.

As we can observe in this figure, it is noteworthy that with a reduced number of labeled examples (10 %), self-labeled techniques are far removed from the results obtained with base classifiers which use a completely labeled training set. Although an increment in the labeled ratio does not produce a proportional increase in the performance obtained with self-labeled techniques, it indicates that from 20 % of the labeled ratio, they offer an acceptable classification performance. As an extreme case, Co-Training (SMO) does not perform well with 10 % of labeled data, and it shows a great improvement when the labeled ratio is augmented, approaching the SMO classifier with all labeled training examples.

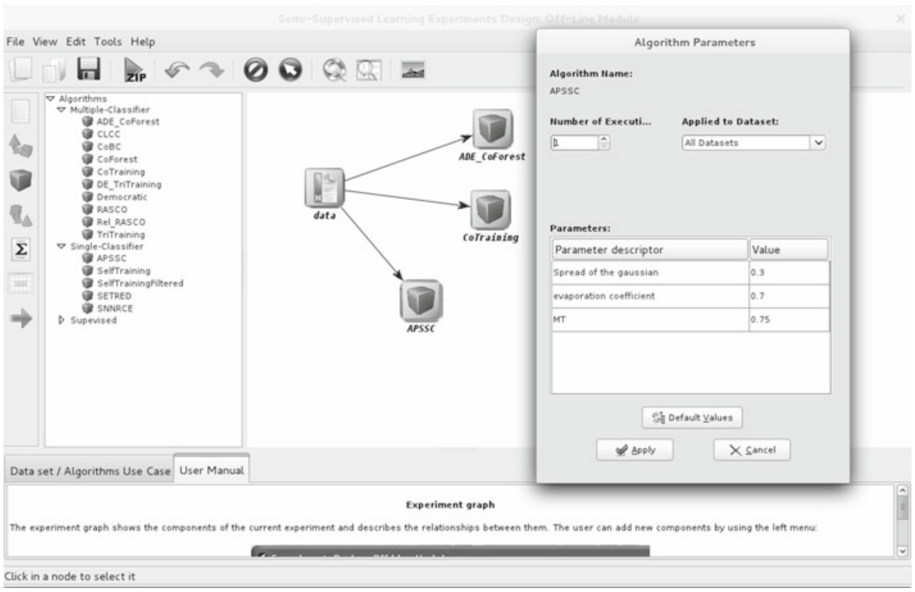
6 Concluding remarks and global guidelines

The present paper provides a complete overview of the self-labeled methods proposed in the literature. We have analyzed the basic and advanced features presented in them. Furthermore, existing and related work have also been reviewed. Based on the main properties studied, we have proposed a taxonomy of self-labeled methods.

The most important methods have been empirically analyzed in terms of transductive and inductive settings. In order to strengthen this experimental study, we have conducted statistical analyses based on nonparametric tests which help us to characterize the capabilities of each method, supporting the conclusions drawn. Several remarks can be made and guidelines suggested:

- This paper helps nonexperts in self-labeled methods to differentiate between them, to make an appropriate decision about their application and to understand their behavior.
- A researcher who needs to apply a self-labeled method should know the main characteristics of these kinds of methods in order to choose the most suitable, depending on the type of problem. The taxonomy proposed and the empirical study can help a researcher to make this decision.
- It is important to know the main advantages of each self-labeled method. In this paper, many methods have been empirically analyzed, but a specific conclusion cannot be drawn regarding the best-performing method. This choice depends on the problem tackled, but the results offered in this paper could help to reduce the set of candidates.
- SSC is a growing field, and more research studies should be conducted. In this paper, several guidelines about unexplored and promising families have been described.
- To propose a new self-labeled method, rigorous analyses should be considered to compare it with the most well-known approaches and those that fit with the basic properties of the new proposal in terms of transductive and inductive learning. To do this, the taxonomy and the proposed experimental framework can help guide a future proposal toward the correct method.
- The empirical study allows us to highlight several methods from among the whole set. In both transductive and inductive settings, TriTraining (C45), Democratic-Co, Co-Bagging (C45) and Co-Training (SMO) are shown to be the best-performing methods. Furthermore, in the inductive phase, the classical Self-Training with C45 as base classifier is also remarkable as an outstanding method.
- The experiments conducted with high-dimensional data sets and very reduced labeled ratio show that much more work is needed in the field of self-labeled techniques to deal with these problems.
- The developed software (see "Appendix") allows the reader to reproduce the experiments carried out and uses it as an SSL framework to implement new methods. It could be a useful tool to do experimental analyses in an easier and more effective way.

Acknowledgments This work is supported by the Research Projects TIN2011-28488, TIC-6858 and P11-TIC-7765.



Algorithm Parameters

Algorithm Name: APSSC

Number of Executi...: 1

Applied to Dataset: All Datasets

Parameters:

Parameter descriptor	Value
Spread of the gaussian	0.3
evaporation coefficient	0.7
MT	0.75

Experiment graph

The experiment graph shows the components of the current experiment and describes the relationships between them. The user can add new components by using the left menu:

Click in a node to select it

Fig. 13 A *snapshot* of the semi-supervised learning module for KEEL

7 Appendix

As a consequence of this work, we have developed a complete SSL framework which has been integrated into the Knowledge Extraction based on Evolutionary Learning (KEEL) tool[3] [26]. This research tool is an open-source software, written in Java, that supports data management and the design of experiments. Until now, KEEL has paid special attention to the implementation of supervised and unsupervised learning, clustering, pattern mining and so on. Nevertheless, it did not offer support for SSL. We integrated a new SSL module into this software.

The main characteristics of this module are as follows:

- All the data sets involved in the experimental study have been included into this module and can be used for new experiments. These data sets are composed of three files for each partition: training, transductive and test partitions. The former is composed of labeled and unlabeled instances (labeled as "unlabeled"). Transductive partition contains the real class of unlabeled instances and the latter collect the test instances. These data sets are included in the KEEL-data set repository and are static, ensuring that further experiments carried out will no longer be dependent on particular data partitions.
- It allows the design of SSL experiments which generate all the XML scripts and a JAR program for running it, by creating a zip file for an off-line run. The SSL module is designed for experiments containing multiple data sets and algorithms connected among themselves to obtain the desired experimental setup. The parameters configuration of the methods is also customizable as well as the number of executions, validation scheme and so on. Figure 13 shows a snapshot of an experiment with three analyzed self-labeled methods and the customization of the parameters of the algorithm APSSC. Note that every

[3] http://www.keel.es.

method could be executed apart from the KEEL tool with an appropriate configuration file.

- Special care has been taken to allow a researcher to be able to use this module to assess the relative effectiveness of his own procedures. Guidelines about how to integrate a method into KEEL can be found in [35].

The KEEL version with the SSL module is available on the associated Web site.

References

1. Zhu X, Goldberg AB (2009) Introduction to semi-supervised learning, 1st edn. Morgan and Claypool, San Rafael, CA
2. Witten IH, Frank E, Hall MA (2011) Data mining: practical machine learning tools and techniques, 3rd edn. Morgan Kaufmann, San Francisco
3. Zhu Y, Yu J, Jing L (2013) A novel semi-supervised learning framework with simultaneous text representing. Knowl Inf Syst 34(3):547–562
4. Chapelle O, Schlkopf B, Zien A (2006) Semi-supervised learning, 1st edn. The MIT Press, Cambridge, MA
5. Pedrycz W (1985) Algorithms of fuzzy clustering with partial supervision. Pattern Recognit Lett 3:13–20
6. Zhao W, He Q, Ma H, Shi Z (2012) Effective semi-supervised document clustering via active learning with instance-level constraints. Knowl Inf Syst 30(3):569–587
7. Chen K, Wang S (2011) Semi-supervised learning via regularized boosting working on multiple semi-supervised assumptions. IEEE Trans Pattern Anal Mach Intell 33(1):129–143
8. Fujino A, Ueda N, Saito K (2008) Semisupervised learning for a hybrid generative/discriminative classifier based on the maximum entropy principle. IEEE Trans Pattern Anal Mach Intell 30(3):424–437
9. Joachims T (1999) Transductive inference for text classification using support vector machines. In: Proceedings of 16th international conference on machine learning, Morgan Kaufmann, pp 200–209
10. Blum A, Chawla S (2001) Learning from labeled and unlabeled data using graph mincuts. In: Proceedings of the eighteenth international conference on machine learning, pp 19–26
11. Wang J, Jebara T, Chang S-F (2013) Semi-supervised learning using greedy max-cut. J Mac Learn Res 14(1):771–800
12. Mallapragada PK, Jin R, Jain A, Liu Y (2009) Semiboost: boosting for semi-supervised learning. IEEE Trans Pattern Anal Mach Intell 31(11):2000–2014
13. Yarowsky D (1995) Unsupervised word sense disambiguation rivaling supervised methods. In: Proceedings of the 33rd annual meeting of the association for computational linguistics, pp 189–196
14. Li M, Zhou ZH (2005) SETRED: self-training with editing. In: Lecture notes in computer science (including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics), vol 3518 LNAI, pp 611–621
15. Blum A, Mitchell T (1998) Combining labeled and unlabeled data with co-training. In: Proceedings of the annual ACM conference on computational learning theory, pp 92–100
16. Du J, Ling CX, Zhou ZH (2010) When does co-training work in real data? IEEE Trans Knowl Data Eng 23(5):788–799
17. Sun S, Jin F (2011) Robust co-training. Int J Pattern Recognit Artif Intell 25(07):1113–1126
18. Jiang Z, Zhang S, Zeng J (2013) A hybrid generative/discriminative method for semi-supervised classification. Knowl-Based Syst 37:137–145
19. Sun S (2013) A survey of multi-view machine learning. Neural Comput Appl 23(7–8):2031–2038
20. Zhou ZH, Li M (2005) Tri-training: exploiting unlabeled data using three classifiers. IEEE Trans Knowl Data Eng 17:1529–1541
21. Li M, Zhou ZH (2007) Improve computer-aided diagnosis with machine learning techniques using undiagnosed samples. IEEE Trans Syst Man Cybern A Syst Hum 37(6):1088–1098
22. Sun S, Shawe-Taylor J (2010) Sparse semi-supervised learning using conjugate functions. J Mach Learn Res 11:2423–2455
23. Zhu X (2005) Semi-supervised learning literature survey. Technical report 1530, Computer Sciences, University of Wisconsin-Madison
24. Chawla N, Karakoulas G (2005) Learning from labeled and unlabeled data: an empirical study across techniques and domains. J Artif Intell Res 23:331–366
25. Zhou Z-H, Li M (2010) Semi-supervised learning by disagreement. Knowl Inf Syst 24(3):415–439

26. Alcalá-Fdez J, Sánchez L, García S, del Jesus MJ, Ventura S, Garrell JM, Otero J, Romero C, Bacardit J, Rivas VM, Fernández JC, Herrera F (2009) KEEL: a software tool to assess evolutionary algorithms for data mining problems. Soft Comput 13(3):307–318
27. Demšar J (2006) Statistical comparisons of classifiers over multiple data sets. J Mach Learn Res 7:1–30
28. García S, Fernández A, Luengo J, Herrera F (2010) Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: experimental analysis of power. Inf Sci 180:2044–2064
29. Triguero I, Sáez JA, Luengo J, García S, Herrera F (2013) On the characterization of noise filters for self-training semi-supervised in nearest neighbor classification, Neurocomputing (in press)
30. Cover TM, Hart PE (1967) Nearest neighbor pattern classification. IEEE Trans Inf Theory 13(1):21–27
31. Dasgupta S, Littman ML, McAllester DA (2001) Pac generalization bounds for co-training. In: Dietterich TG, Becker S, Ghahramani Z (eds) Advances in neural information processing systems. Neural information processing systems: natural and synthetic, vol 14. MIT Press, Cambridge, pp 375–382
32. Quinlan JR (1993) C4.5 programs for machine learning. Morgan Kaufmann Publishers, San Francisco, CA
33. Efron B, Tibshirani RJ (1993) An Introduction to the bootstrap. Chapman & Hall, New York
34. Goldman S, Zhou Y (2000) Enhancing supervised learning with unlabeled data. In: Proceedings of the 17th international conference on machine learning. Morgan Kaufmann, pp 327–334
35. Alcalá-Fdez J, Fernandez A, Luengo J, Derrac J, García S, Sánchez L, Herrera F (2011) KEEL data-mining software tool: data set repository, integration of algorithms and experimental analysis framework. J Multiple-Valued Logic Soft Comput 17(2–3):255–277
36. Bennett K, Demiriz A, Maclin R (2002) Exploiting unlabeled data in ensemble methods. In: Proceedings of the ACM SIGKDD international conference on knowledge discovery and data mining, pp 289–296
37. Zhou Y, Goldman S (2004) Democratic co-learning. In: IEEE international conference on tools with artificial intelligence, pp 594–602
38. Deng C, Guo M (2006) Tri-training and data editing based semi-supervised clustering algorithm. In: Gelbukh A, Reyes-Garcia C (eds) MICAI 2006: advances in artificial intelligence, vol 4293 of lecture notes in computer science. Springer, Berlin, pp 641–651
39. Wang J, Luo S, Zeng X (2008) A random subspace method for co-training. In: IEEE international joint conference on computational intelligence, pp 195–200
40. Hady M, Schwenker F (2008) Co-training by committee: a new semi-supervised learning framework. In: IEEE international conference on data mining workshops, ICDMW ’08, pp 563–572
41. Hady M, Schwenker F (2010) Combining committee-based semi-supervised learning and active learning. J Comput Sci Technol 25:681–698
42. Hady M, Schwenker F, Palm G (2010) Semi-supervised learning for tree-structured ensembles of rbf networks with co-training. Neural Netw 23:497–509
43. Yaslan Y, Cataltepe Z (2010) Co-training with relevant random subspaces. Neurocomputing 73(10–12):1652–1661
44. Huang T, Yu Y, Guo G, Li K (2010) A classification algorithm based on local cluster centers with a few labeled training examples. Knowl-Based Syst 23(6):563–571
45. Halder A, Ghosh S, Ghosh A (2010) Ant based semi-supervised classification. In: Proceedings of the 7th international conference on swarm intelligence, ANTS’10, Springer, Berlin, Heidelberg, pp 376–383
46. Wang Y, Xu X, Zhao H, Hua Z (2010) Semi-supervised learning based on nearest neighbor rule and cut edges. Knowl-Based Syst 23(6):547–554
47. Deng C, Guo M (2011) A new co-training-style random forest for computer aided diagnosis. J Intell Inf Syst 36:253–281. doi:10.1007/s10844-009-0105-8
48. Nigam K, Mccallum A, Thrun S, Mitchell T (2000) Text classification from labeled and unlabeled documents using em. Mach Learn 39(2):103–134
49. Tang X-L, Han M (2010) Semi-supervised Bayesian artmap. Appl Intell 33(3):302–317
50. Joachims T (2003) Transductive learning via spectral graph partitioning. In: Proceedings of twentieth international conference on machine learning, vol 1, pp 290–297
51. Belkin M, Niyogi P, Sindhwani V (2006) Manifold regularization: a geometric framework for learning from labeled and unlabeled examples. J Mach Learn Res 7:2399–2434
52. Xie B, Wang M, Tao D (2011) Toward the optimization of normalized graph Laplacian. IEEE Trans Neural Netw 22(4):660–666
53. Burges C (1998) A tutorial on support vector machines for pattern recognition. Data Min Knowl Discov 2(2):121–167
54. Chapelle O, Sindhwani V, Keerthi SS (2008) Optimization techniques for semi-supervised support vector machines. J Mach Learn Re. 9:203–233

55. Adankon M, Cheriet M (2010) Genetic algorithm-based training for semi-supervised svm. Neural Comput Appl 19:1197–1206
56. Tian X, Gasso G, Canu S (2012) A multiple kernel framework for inductive semi-supervised svm learning. Neurocomputing 90:46–58
57. Sugato B, Raymond JM (2003) Comparing and unifying search-based and similarity-based approaches to semi-supervised clustering. In: Proceedings of the ICML-2003 workshop on the continuum from labeled to unlabeled data in machine learning and data mining, pp 42–49
58. Yin X, Chen S, Hu E, Zhang D (2010) Semi-supervised clustering with metric learning: an adaptive kernel method. Pattern Recognit 43(4):1320–1333
59. Grira N, Crucianu M, Boujemaa N (2004) Unsupervised and semi-supervised clustering: a brief survey. In: A review of machine learning techniques for processing multimedia content. Report of the MUSCLE European network of excellence FP6
60. Freund Y, Seung HS, Shamir E, Tishby N (1997) Selective sampling using the query by committee algorithm. Mach Learn 28:133–168
61. Muslea I, Minton S, Knoblock C (2002) Active + semi-supervised learning = robust multi-view learning. In: Proceedings of ICML-02, 19th international conference on machine learning, pp 435–442
62. Zhang Q, Sun S (2010) Multiple-view multiple-learner active learning. Pattern Recognit 43(9):3113–3119
63. Yu H (2011) Selective sampling techniques for feedback-based data retrieval. Data Min Knowl Discov 22(1–2):1–30
64. Belhumeur P, Hespanha J, Kriegman D (1997) Eigenfaces vs. fisherfaces: recognition using class specific linear projection. IEEE Trans Pattern Anal Mach Intell 19(7):711–720
65. Hastie T, Tibshirani R, Friedman J (2001) The elements of statistical learning: data mining, inference and prediction, 2nd edn. Springer, Berlin
66. Song Y, Nie F, Zhang C, Xiang S (2008) A unified framework for semi-supervised dimensionality reduction. Pattern Recognit 41(9):2789–2799
67. Li Y, Guan C (2008) Joint feature re-extraction and classification using an iterative semi-supervised support vector machine algorithm. Mach Learn 71:33–53
68. Liu H, Motoda H (eds) (2007) Computational methods of feature selection. Chapman &Hall/CRC data mining and knowledge discovery series. Chapman & Hall/CRC, Boca Raton, FL
69. Zhao J, Lu K, He X (2008) Locality sensitive semi-supervised feature selection. Neurocomputing 71(10–12):1842–1849
70. Gregory PA, Gail AC (2010) Self-supervised ARTMAP. Neural Netw 23:265–282
71. Cour T, Sapp B, Taskar B (2011) Learning from partial labels. J Mach Learn Res 12:1501–1536
72. Joshi A, Papanikolopoulos N (2008) Learning to detect moving shadows in dynamic environments. IEEE Trans Pattern Anal Mach Intell 30(11):2055–2063
73. Ben-David A (2007) A lot of randomness is hiding in accuracy. Eng Appl Artif Intell 20:875–885
74. Alpaydin E (2010) Introduction to machine learning, 2nd edn. MIT Press, Cambridge, MA
75. Asuncion A, Newman D (2007) UCI machine learning repository. http://www.ics.uci.edu/mlearn/MLRepository.html
76. Wu X, Kumar V (eds) (2009) The top ten algorithms in data mining. Chapman & Hall/CRC data mining and knowledge discovery. Chapman & Hall/CRC, Boca Raton, FL
77. Aha DW, Kibler D, Albert MK (1991) Instance-based learning algorithms. Mach Learn 6(1):37–66
78. John GH, Langley P (2001) Estimating continuous distributions in Bayesian classifiers. In: Proceedings of the eleventh conference on uncertainty in artificial intelligence. Morgan Kaufmann, San Mateo, pp 338–345
79. Vapnik VN (1998) Statistical learning theory. Wiley-Interscience, London
80. Platt JC (1999) Fast training of support vector machines using sequential minimal optimization. MIT Press, Cambridge, MA
81. García S, Herrera F (2008) An extension on statistical comparisons of classifiers over multiple data sets for all pairwise comparisons. J Mach Learn Res 9:2677–2694
82. Sheskin DJ (2011) Handbook of parametric and nonparametric statistical procedures, 5th edn. Chapman & Hall/CRC, Boca Raton, FL
83. Friedman M (1937) The use of ranks to avoid the assumption of normality implicit in the analysis of variance. J Am Stat Assoc 32:675–701
84. Bergmann G, Hommel G (1988) Improvements of general multiple test procedures for redundant systems of hypotheses. In: Bauer P, Hommel G, Sonnemann E (eds) Multiple hypotheses testing. Springer, Berlin pp 100–115
85. Yang Y, Webb G (2009) Discretization for naive-Bayes learning: managing discretization bias and variance. Mac Learn 74(1):39–74

86. García S, Luengo J, Saez JA, López V, Herrera F (2013) A survey of discretization techniques: taxonomy and empirical analysis in supervised learning. IEEE Trans Knowl Data Eng 25(4):734–750
87. Jolliffe IT (1986) Principal component analysis. Springer, Berlin

Author Biographies

Isaac Triguero received the M.Sc. degree in Computer Science from the University of Granada, Granada, Spain, in 2009. He is currently a Ph.D. student in the Department of Computer Science and Artificial Intelligence, University of Granada, Granada, Spain. His research interests include data mining, data reduction, biometrics, evolutionary algorithms and semi-supervised learning.

Salvador García received the M.Sc. and Ph.D. degrees in Computer Science from the University of Granada, Granada, Spain, in 2004 and 2008, respectively. He is currently an Associate Professor in the Department of Computer Science, University of Jaén, Jaén, Spain. He has published more than 30 papers in international journals. As edited activities, he has co-edited two special issues in international journals on different Data Mining topics. His research interests include data mining, data reduction, data complexity, imbalanced learning, semi-supervised learning, statistical inference and evolutionary algorithms.

Francisco Herrera received his M.Sc. in Mathematics in 1988 and Ph.D. in Mathematics in 1991, both from the University of Granada, Spain. He is currently a Professor in the Department of Computer Science and Artificial Intelligence at the University of Granada. He has published more than 230 papers in international journals. He is coauthor of the book "Genetic Fuzzy Systems: Evolutionary Tuning and Learning of Fuzzy Knowledge Bases" (World Scientific, 2001). He currently acts as Editor in Chief of the international journal "Progress in Artificial Intelligence" (Springer). He acts as area editor of the International Journal of Computational Intelligence Systems and associated editor of the journals: IEEE Transactions on Fuzzy Systems, Information Sciences, Knowledge and Information Systems, Advances in Fuzzy Systems, and International Journal of Applied Metaheuristics Computing; and he serves as member of several journal editorial boards, among others: Fuzzy Sets and Systems, Applied Intelligence, Information Fusion, Evolutionary Intelligence, International Journal of Hybrid Intelligent Systems, Memetic Computation, and Swarm and Evolutionary Computation. He received the following honors and awards: ECCAI Fellow 2009, 2010 Spanish National Award on Computer Science ARITMEL to the "Spanish Engineer on Computer Science", International Cajastur "Mamdani" Prize for Soft Computing (Fourth Edition, 2010), IEEE Transactions on Fuzzy System Outstanding 2008 Paper Award (bestowed in 2011), and 2011 Lotfi A. Zadeh Prize Best paper Award of the International Fuzzy Systems Association. His current research interests include computing with words and decision making, bibliometrics, data mining, biometrics, data preparation, instance selection, fuzzy rule based systems, genetic fuzzy systems, knowledge extraction based on evolutionary algorithms, memetic algorithms and genetic algorithms.