Neurocomputing 132 (2014) 30–41

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

On the characterization of noise filters for self-training semi-supervised in nearest neighbor classification

Isaac Triguero [a,*], José A. Sáez [a], Julián Luengo [b], Salvador García [c], Francisco Herrera [a]

[a] Department of Computer Science and Artificial Intelligence, CITIC-UGR (Research Center on Information and Communications Technology), University of Granada, 18071 Granada, Spain
[b] Department of Civil Engineering, LSI, University of Burgos, 09006 Burgos, Spain
[c] Department of Computer Science, University of Jaén, 23071 Jaén, Spain

ARTICLE INFO

Article history:
Received 22 October 2012
Received in revised form
18 February 2013
Accepted 30 May 2013
Available online 12 November 2013

Keywords:
Noise filters
Noisy data
Self-training
Semi-supervised learning
Nearest neighbor classification

ABSTRACT

Semi-supervised classification methods have received much attention as suitable tools to tackle training sets with large amounts of unlabeled data and a small quantity of labeled data. Several semi-supervised learning models have been proposed with different assumptions about the characteristics of the input data. Among them, the self-training process has emerged as a simple and effective technique, which does not require any specific hypotheses about the training data. Despite its effectiveness, the self-training algorithm usually make erroneous predictions, mainly at the initial stages, if noisy examples are labeled and incorporated into the training set.

Noise filters are commonly used to remove corrupted data in standard classification. In 2005, Li and Zhou proposed the addition of a statistical filter to the self-training process. Nevertheless, in this approach, filtering methods have to deal with a reduced number of labeled instances and the erroneous predictions it may induce. In this work, we analyze the integration of a wide variety of noise filters into the self-training process to distinguish the most relevant features of filters. We will focus on the nearest neighbor rule as a base classifier and ten different noise filters. We provide an extensive analysis of the performance of these filters considering different ratios of labeled data. The results are contrasted with nonparametric statistical tests that allow us to identify relevant filters, and their main characteristics, in the field of semi-supervised learning.

1. Introduction

The construction of classifiers can be considered one of the most important and challenging tasks in machine learning and data mining [1]. Supervised classification, which has attracted much attention and research efforts [2], aims to build classifiers using a set of labeled data. By contrast, in many real-world tasks, unlabeled data are easier to obtain than labeled ones because they require less effort, expertise and time-consumption. In this context, semi-supervised learning (SSL) [3] is a learning paradigm concerned with the design of classifiers in the presence of both labeled and unlabeled data.

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, SSL encompasses several settings such as semi-supervised classification (SSC) [4] and semi-supervised clustering [5]. 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 [6], aims to obtain better defined clusters than the ones obtained from unlabeled data. There are other SSL settings, including regression with labeled and unlabeled data, or dimensionality reduction [7] to find a faithful low dimensional mapping, or selection of the high dimensional data in a SSL context. We focus on SSC.

SSC can be categorized into two slightly different settings [8], denoted as transductive and inductive learning. On 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 to carry out an extensive analysis of the performance of the studied methods.

Many different approaches have been proposed to classify using unlabeled data in SSC. They usually make different assumptions

* Corresponding author. Tel.: +34 958 240598; fax: +34 958 243317.
E-mail addresses: triguero@decsai.ugr.es (I. Triguero), smja@decsai.ugr.es (J.A. Sáez), jluengo@ubu.es (J. Luengo), sglopez@ujaen.es (S. García), herrera@decsai.ugr.es (F. Herrera).

0925-2312/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.neucom.2013.05.055

related to the link between the distribution of unlabeled and labeled data. Generative models [9] 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 [10]. The standard co-training [11] 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 [12–15]. It trains one classifier in each specific view, and then the classifiers teach each other the most confident predicted examples. Multiview learning [16,17] can be viewed as a generalization of co-training, without requiring explicit feature splits or the iterative mutual-teaching procedure. Instead, it focuses on the explicitly hypothetical agreement of several classifiers [18]. There are also other algorithms such as transductive inference for support vector machines [19,20] 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 [21]. 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 [22]. In addition, there are recent studies which address multiple assumptions in one model [8].

Self-training [23,24] is a simple and effective SSL methodology which has been successfully applied in many real instances [25,26]. In the self-training process, a classifier is trained with an initially 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.

However, this idea can lead to erroneous predictions if noisy examples are classified as the most confident examples and incorporated into the labeled training set. In [27], the authors propose the addition of a statistical filter [28] to the self-training process, naming this algorithm SETRED. Nevertheless, this method does not perform well in many domains. The use of a particular filter which has been designed and tested under different conditions is not straightforward. Although the aim of any filter is to remove potentially noisy examples, both correct examples and examples containing valuable information may also be removed. Thus, detecting true noisy examples is a challenging task because the success of filtering methods depends on several factors [29] such as the kind and nature of data errors, the quantity of noise removed or the capabilities of the classifier to deal with the loss of useful information related to the filtering. In the self-training approach, the number of available labeled data and the induced noisy examples are two decisive factors when filtering noise. Hence, the performance of the combination of filtering techniques and self-training relies heavily on the filtering method chosen. It is so much so that the inclusion or the absence of one prototype into the labeled training set can alter the following stages of the self-training approach, especially in early steps. For these reasons, the inclusion and analysis of the most suitable filtering method into the self-training is mandatory in order to diminish the influence of noisy data.

Filtering techniques follow different approaches to determine whether an example could be noisy or not. We distinguish two types of noise detection mechanism: local and global. We call local methods to those techniques in which the removal decision is based on a local neighborhood of instances [30,31]. Global methods create different models from the training data. Mislabeled examples can be considered noisy depending on the hypothesis agreement of the used classifiers [32,33]. It is necessary to mention that there are other related approaches in which unlabeled data are used to identify mislabeled training data [34,35].

In this work we deepen in the integration of different noise filters and we further analyze recent proposals in order to establish their suitability with respect to the self-training process. We will adopt the Nearest Neighbor (NN) rule [36] as the base classifier, which has been highlighted as one of the most influential techniques in data mining [37]. For each filtering family, the most representative noise filters will be tested. The analysis of the behavior of noise filters in self-training motivates the global purpose of this paper, which pursues three objectives:

- To determine which characteristics of noise filters are more appropriate to be included in the self-training process.
- To perform an empirical study for analyzing the transductive and inductive capabilities of the filtered and non-filtered self-training algorithm.
- To check the behavior of this approach when dealing with data sets with different ratios of labeled data.

We will conduct experiments involving a total of 60 classification data sets with different ratios of labeled data: 10%, 20%, 30% and 40%. In order to test the behavior of noise filters, the experimental study will include a statistical analysis based on nonparametric statistical tests [38]. A web page with all the complementary material is available at ⟨http://sci2s.ugr.es/SelfTraining+Filters⟩, including this paper's basic information, all the data sets created and the complete results obtained for each algorithm.

The rest of the paper is organized as follows: Section 2 defines the SSC problem and the self-training approach. Section 3 explains how to combine self-training with noise filters. Section 4 introduces the filtering algorithms used. Section 5 presents the experimental framework and Section 6 discusses the analysis of results obtained. Finally, in Section 7 we summarize our conclusions.

2. Background: semi-supervised learning via the self-training approach

This section provides the necessary information to understand the proposed integration of noise filters into the self-training process. Section 2.1 defines the SSC problem. Then, Section 2.2 presents the self-training approach used to address the SSC problem.

2.1. Semi-supervised classification

This section presents the definition and notation for the SSC problem. A specification of this problem 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 i-th feature of the p-th sample. Then, let us assume that there is a labeled set L which consists of $\mathbf{n}$ instances $\mathbf{x}_p\omega$ with ω known. Furthermore, there is an unlabeled set U which consists of $\mathbf{m}$ instances $\mathbf{x}_q\omega$ with ω unknown, let $\mathbf{m}\gg\mathbf{n}$. The $L\cup U$ set forms the training set 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 $\mathbf{x}_q\omega$ of U with their correct class. The class assignation 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 $\mathbf{x}_r\omega$ 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.

2.2. Self-training

The self-training approach is a wrapper methodology characterized by the fact that the learning process uses its own predictions to teach itself. This process is also known as bootstrapping or self-teaching [39]. In general, self-training can be used either as inductive or transductive learning depending on the nature of the classifier. Self-training follows an iterative procedure in which a classifier is trained using labeled data to predict the labels of unlabeled data in order to obtain an enlarged labeled set *L*.

Fig. 1 outlines the pseudo-code of the self-training methodology. In the following we describe the most significant instructions enumerated from 1 to 22.

First of all, it is necessary to determine the number of unlabeled instances which will be added to *L* in each iteration. Note that this parameter can be a constant, or it can be chosen as a proportional value of the number of instances of each class in *L*, as Blum and Mitchell suggest in [11]. We apply this idea in our implementations to determine the amount of prototypes per class which will be added to *L* in each iteration (Instructions 1–11).

Then, the algorithm enters into a loop to enlarge the labeled set *L* (Instructions 14–20). Instruction 15 calculates the confidence predictions of all the unlabeled instances, as the probability of belonging to each class. The way in which the confidence predictions are measured is dependant on the type of classifier used. Unlike probabilistic models such as Naive Bayes, whose confidence predictions can be measured as the output probability in prediction, the NN rule has no explicitly measured confidence for an instance. For the NN rule, the algorithm approximates confidence in terms of distance, hence, the most confident unlabeled instance is defined as the closest unlabeled instances to any labeled one (as defined in [27,3]).

Next, instruction 16 creates a set L' consisting of the most confident unlabeled data for each class, keeping the proportion of instances per class previously computed. L' is labeled with its predictions and added to *L* (Instruction 17). Instruction 18 removes the instances of L' from *U*.

In the original description of the self-training approach [23], this process was repeated until all the instances from *U* had been added to *L*. However, following [11], we have established a limit to the number of iterations, *MAXITER* (Instruction 14). Hence, a pool of unlabeled examples smaller than *U* is used in our implementations.

Finally, the obtained *L* set is used to classify the *U* set for transductive learning and the *TS* for inductive learning.

Require: L, U, TS
1: $PrototypesPerClass[1..NumberOfClasses] = 1$
2: Minimum $= +\infty$
3: **for** $i = 1$ to $NumberOfClasses$ **do**
4: $PrototypesPerClass[i]$ = |(Prototypes of Class i∈L)| / | L |
5: **if** PrototypesPerClass[i] < Minimum **then**
6: Minimum = $PrototypesPerClass[i]$
7: **end if**
8: **end for**
9: **for** $j = 1$ to $NumberOfClasses$ **do**
10: PrototypesPerClass[j] = Round (PrototypesPerClass[j] / Minimum)
11: **end for**
12: $L' = \emptyset$
13: z=0
14: **while** $z < MAXITER$ and $|U| > 0$ **do**
15: Confidence [1..| U |] = ConfidenceClassification(L, U)
16: $L' = PrototypesPerClass[j]$ most confident unlabeled instances for each class j.
17: $L = L \cup L'$
18: Removing L' from U
19: z=z+1
20: **end while**
21: Transductive Phase < − Classify(L, U)
22: Inductive Phase < − Classify(L, TS)

Fig. 1. Self-training pseudo-code.

3. Combining self-training and filtering methods

In this section we explain the combination of self-training with noise filters in depth. As mentioned above, the goal of the self-training process is to find the most adequate class label for unlabeled data with the aim of enlarging the *L* set. However, in SSC, the number of initial labeled examples tends to be too small to train a classifier with good generalization capabilities. In this scenario, 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.

Two types of noisy instances may appear during the self-labeling process:

- The first one is caused by the distribution of classes in *L*. It can lead to the classifier to label erroneously some instances.
- There may be outliers within the original unlabeled data. This second kind can be detected, avoiding its labeling and its inclusion into *L*.

These ideas motivate the treatment of noisy data during the self-training process. Filtering methods have been commonly used to deal with noisy data. Nevertheless, most of the proposed filtering schemes have been designed into the supervised classification framework. Hence, the number of labeled data can determine the way in which one filter decides whether an example is noisy or not. If incorrect examples are appropriately detected and removed during the labeling process, the generalization capabilities of the classifier are expected to be improved.

The filtering technique should be applied at each iteration, after L' is formed, in order to detect both types of noisy instances. The identification of noisy examples is performed using the set $L \cup L'$ as a training set. If one example of L' is annotated as a possible noisy example, it is removed from L', and it will not be added to *L*. Nevertheless, this instance should be cleaned from *U*. Note that this scheme does not try to relabel suspicious examples and thereby avoids the introduction of new noise into the training set [27].

Fig. 2 shows a case of study of filtered self-training in a two-class problem. In this figure, we can observe the first iteration of

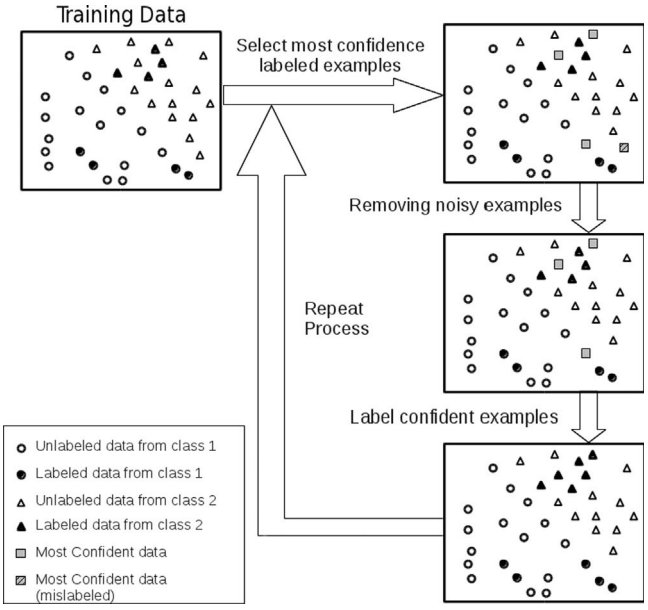

Fig. 2. Example of labeling process with editing.

the process and how the selection of the most confident examples can fail due to the distribution of the given labeled instances. One example of Class 2 has been selected as one of the most confident instances of Class 1. A filtering technique is needed to remove incorrectly labeled instances in order to avoid the erroneous future labeling in subsequent iterations.

4. Filtering methods

This section describes the filters adopted in our study. Filtering methods are preprocessing mechanisms to detect and eliminate noisy examples in the training set. The separation of noise detection and learning has the advantage that noisy examples do not influence the model building design [40].

As we explained before, each method considers an example to be harmful depending on its nature. Broadly speaking, we can categorize filters into two different types: local (Section 4.1) and global filters (Section 4.2).

In all descriptions, we use *TR* to refer to the training set, *FS* to the filtered set, and *ND* to refer to the noisy data identified in the training set (initially, $ND = \emptyset$).

4.1. Local filters

These methods create neighborhoods of instances to detect suspicious examples. Most of them are based on the distance between prototypes to determine their similarity. The best known distance measure for these filters is Euclidean distance (Eq. (1)). We will use it throughout this study, since it is simple, easy to optimize, and has been widely used in the field of instance based learning [41]

$$EuclideanDistance(X,Y) = \sqrt{\sum_{i=0}^{D} (x_{pi} - x_{qi})^2} \quad (1)$$

here we offer a brief description of the local filtering methods studied:

- *Edited Nearest Neighbor* (ENN) [42]: This algorithm starts with $FS = TR$. Then each instance in *FS* is removed if it does not agree with the majority of its k nearest neighbors.
- *All kNN* (AllKNN) [43]: The All kNN technique is an extension of ENN. Initially, $FS = TR$. Then the NN rule is applied k times. In each execution, the NN rule varies the number of neighbors considered between 1 and k. If one instance is misclassified by the NN rule, it is registered as removable from the *FS*. Then all those that do meet the criteria are removed at once.
- *Relative Neighborhood Graph Edition* (RNGE) [44]: This technique builds a proximity undirected graph $G = (V, E)$, in which each vertex corresponds to an instance from *TR*. There is a set of edges E, so that $(x_i, x_j) \in E$ if and only if x_i and x_j satisfy some neighborhood relation (Eq. (2)). In this case, we say that these instances are *graph neighbors*. The graph neighbors of a given point constitute its *graph neighborhood*. The edition scheme discards those instances misclassified by their graph neighbors (by the usual voting criterion)

$$(x_i, x_j) \in E \Leftrightarrow d(x_i, x_j) \leq \max(d(x_i, x_k), d(x_j, x_k)), \quad \forall x_k \in TR, \quad k \neq i, j. \quad (2)$$

- *Modified Edited Nearest Neighbor* (MENN) [45]: This algorithm starts with $FS = TR$. Then each instance $\mathbf{x}_p$ in *FS* is removed if it does not agree with all of its $k+l$ nearest neighbors, where l is the number of instances in *FS* which are at the same distance as the last neighbor of $\mathbf{x}_p$.

  Furthermore, MENN works with a prefixed number of pairs (k, k'). k is employed as the number of neighbors used to perform the editing process, and k' is employed to validate the edited set *FS* obtained. The best pair found is employed as the final reference set. If two or more sets are found to be optimal, then both are used in the classification of the test instances. A majority rule is used to decide the output of the classifier in this case.
- *Nearest Centroid Neighbor Edition* (NCNEdit) [46]: This algorithm defines the neighborhood, taking into account not only the proximity of prototypes to a given example, but also their symmetrical distribution around it. Specifically, it calculates the k nearest centroid neighbors (k NCNs). These k neighbors can be searched for through an iterative procedure [47] in the following way:
  1. The first neighbor of $\mathbf{x}_p$ is also its nearest neighbor, $\mathbf{x}_q^1$.
  2. The ith neighbor, $\mathbf{x}_q^i$, $i \gg 2$ is such that the centroid of this and previously selected neighbors, $\mathbf{x}_q^1, \ldots, \mathbf{x}_q^i$, is the closest to $\mathbf{x}_p$.

  The NCN Editing algorithm is a slight modification of ENN, which consists of discarding from *FS* every example misclassified by the k NCN rule.
- *Cut edges weight statistic* (CEWS) [28]: This method generates a graph with a set of vertices $V = TR$, and a set of edges, E, connecting each vertex to its nearest neighbor. An edge connecting two vertices that have different labels is denoted as a cut edge. If an example is located in a neighborhood with too many cut edges, it should be considered as noise. Next, a statistical procedure is applied in order to label cut edges as noisy examples. This is the filtering method used in the first self-training approach with edition [27] (SETRED).
- *Edited Nearest Neighbor Estimating Class Probabilistic and Threshold* (ENNTh) [48]: This method applies a probabilistic NN rule, where the class of an instance is decided as a weighted probability of the class of its nearest neighbors (each neighbor has the same *a priori* probability, and its associated weight is the inverse of its distance). The editing process is performed starting with $FS = TR$ and deleting from *FS* every prototype misclassified by this probabilistic rule.
  Furthermore, it defines a threshold in the NN rule, which will not consider instances with an assigned probability lower than the established threshold.
- *Multiedit* (Multiedit) [49,50]: This method starts with $FS = \varnothing$ and a new set R defined as $R = TR$. Then this technique splits R into nf blocks: $R_1, \ldots, R_b$ ($nf > 2$). For each instance of the block nf_i, it applies a k NN rule with $R_{(nf_i+1) mod\ b}$ as the training set. All misclassified instances are discarded. The remaining instances constitute the new *TR*. This process is repeated while at least one instance is discarded.

4.2. Global filters

We denote as global filters those methods which apply a classifier to several subsets of *TR* in order to detect problematic examples. These methods use different methodologies to divide the *TR*. Then, these methods create models over the generated subsets and use different heuristics to determine noisy examples. From here, nf is the number of folds in which the training data are partitioned by the filtering method.

- *Classification Filter* (CF) [32]: The main steps of this filtering algorithm are the following:
  1. Split the current training data set *TR* using an nf-fold cross validation scheme.

2. For each of these *nf* parts, a learning algorithm is trained on the other $n-1$ parts, resulting in n different classifiers. Here, C4.5 is used as the learning algorithm [51].
3. These n resulting classifiers are then used to tag each instance in the excluded part as either correct or mislabeled, by comparing the training label with that assigned by the classifier.
4. The misclassified examples from the previous step are added to *ND*.
5. Remove the noisy examples from the training set: $FS \leftarrow TR \backslash ND$.

- *Iterative Partitioning Filter* (IPF) [33]: This method removes noisy instances in multiple iterations until a stopping criterion is reached. The iterative process stops if, for a number of consecutive iterations, the number of identified noisy examples in each of these iterations is less than a percentage of the size of the original training data set. The basic steps of each iteration are as follows:
  1. Split the current training data set *TR* into *nf* equal sized subsets.
  2. Build a model with the C4.5 algorithm over each of these *nf* subsets and use them to evaluate the whole current training data set *TR*.
  3. Add to *ND* the noisy examples identified in *TR* using a voting scheme.
  4. Remove the noisy examples from the training set: $FS \leftarrow TR \backslash ND$.

  Two voting schemes can be used to identify noisy examples: consensus and majority. The former removes an example if it is misclassified by all the classifiers, whereas the latter removes an instance if it is misclassified by more than half of the classifiers. Furthermore, a noisy instance should be misclassified by the model which was induced in the subset containing that instance. In our experimentation we consider the majority scheme in order to detect most of the potentially noisy examples.

5. Experimental framework

This section describes the experimental study carried out in this paper. We provide the measures used to determine the performance of the algorithms (Section 5.1), the characteristics of the problems used for the experimentation (Section 5.2), an enumeration of the algorithms used with their respective parameters (Section 5.3) and finally a description of the nonparametric statistical tests applied to contrast the results obtained (Section 5.4).

5.1. Performance measures

Two measures are widely used for measuring the effectiveness of classifiers: accuracy [1,2] and Cohen's kappa rate [52]. 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 [1,2].
- *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.

Table 1
Summary description of the original data sets.

Data set	#Ex.	#Atts.	#Cl.	Data set	#Ex.	#Atts.	#Cl.
abalone	4174	8	28	monks	432	6	2
appendicitis	106	7	2	movement	360	90	15
australian	690	14	2	mushroom	8124	22	2
autos	205	25	6	nursery	12,690	8	5
balance	625	4	3	pageblocks	5472	10	5
banana	5300	2	2	penbased	10,992	16	10
bands	539	19	2	phoneme	5404	5	2
breast	286	9	2	pima	768	8	2
bupa	345	6	2	PostOper	90	8	3
chess	3196	36	2	ring	7400	20	2
coil2000	9822	85	2	saheart	462	9	2
contraceptive	1473	9	3	satimage	6435	36	7
crx	125	15	2	segment	2310	19	7
dermatology	366	33	6	sonar	208	60	2
ecoli	336	7	8	spambase	4597	55	2
flare	1066	9	2	spectheart	267	44	2
german	1000	20	2	splice	3190	60	3
glass	214	9	7	tae	151	5	3
haberman	306	3	2	texture	5500	40	11
heart	270	13	2	thyroid	7200	21	3
hepatitis	155	19	2	tic-tac-toe	958	9	2
ionosphere	351	33	2	titanic	2201	3	2
housevotes	435	16	2	twonorm	7400	20	2
iris	150	4	3	vehicle	846	18	4
led7digit	500	7	10	vowel	990	13	11
letter	20,000	16	10	wdbc	569	30	2
lym	148	18	4	wine	178	13	3
magic	19,020	10	2	wisconsin	683	9	2
mammograph	961	5	2	yeast	1484	8	10
marketing	8993	13	9	zoo	101	16	7

5.2. Data sets

The experimentation is based on 60 standard classification data sets taken from the KEEL-data set repository[1] [53,54]. Table 1 summarizes the properties of the selected data sets. It shows, for each data set, the number of examples (#Ex.), the number of attributes (#Atts.), and the number of classes (#Cl.). The data sets considered in this study contain between 100 and 20,000 instances, the number of attributes ranges from 2 to 85 and the number of classes varies between 2 and 28. Their values are normalized in the interval [0,1] to equalize the influence of attributes with different range domains when using the NN rule.

These data sets have been partitioned using the ten fold cross-validation procedure. Each training partition is divided into two parts: labeled and unlabeled examples. Using the recommendation established in [24], 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 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 which contains 1000 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*

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

Table 2
Parameter specification for all the methods employed in the experimentation.

Algorithm	*Parameters*
SelfTraining	*MAX_ITER*=40
ENN	Number of neighbors=3
AllKNN	Number of neighbors=3
RNGE	Order of the graph=1st order
MENN	Number of neighbors=3
NCNEdit	Number of neighbors=3
CEWS	Threshold=0.1
ENNTh	Noise threshold=0.7
Multiedit	Number of sub-blocks=3
CF	Number of partitions: n=5, Base algorithm: C4.5
IPF	Number of partitions: n=5, Filter type: *majority*, Iterations for stop criterion: i=3, Examples removed pct.: p=1%, Base algorithm: C4.5
SNNRCE	Threshold=0.5

without their labels. In summary, this experimental study involves a total of 240 data sets (60 data sets∗4 labeled rates). Note that test partitions are kept aside to evaluate the performance of the learned hypothesis.

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

5.3. Algorithm used and parameters

Apart from the original self-training proposal, two of the main variants of this algorithm proposed in the literature are SETRED [27] and SNNRCE [24]. The former corresponds to the first attempt to use a particular filter (CEWS) [28] during the self-training process. Hence, we will consider SETRED as equivalent to self-training with a CEWS filter. The latter algorithm is a recent approach which introduces several steps into the original self-training approach, such as a re-labeling stage and a relative graph-based neighborhood to determine the confidence level during the labeling process. We include these proposals in the experimental study as comparative techniques.

Table 2 shows the configuration parameters, which are common to all problems, of the comparison techniques and filters used with self-training. We focus this experimentation on the recommended parameters proposed by their respective authors, assuming that the choice of the values of the parameters was optimally made. For those filtering methods which are based on the NN rule, we have established the number of nearest neighbors as k=3. In filtering algorithms, a value $k>1$ may be convenient, when the interest lies in protecting the classification task against noisy instances, as Wilson and Martinez suggested in [30]. In all of the techniques, we use the Euclidean distance. Due to the fact that CEWS, Multiedit, CF, IPF and SNNRCE are stochastic methods, they have been run three times per partition.

Implementations of the algorithms can be found on the web site associated with this paper.

5.4. Statistical tools for analysis

The use of hypothesis testing methods to support the analysis of results is highly recommended in the field of Machine Learning. The aim of these techniques is to identify the most relevant differences found between the methods [55,56]. To this end, 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 [57].

Throughout the empirical study, we will focus on the use of the Friedman Aligned-Ranks (FAR) test [38], as a tool for contrasting the behavior of each proposal. Its application will allow us to highlight the existence of significant differences between methods. The Finner test is applied as a post hoc procedure to find out which algorithms present significant differences. More information about these tests and other statistical procedures can be found at http://sci2s.ugr.es/sicidm/.

6. Analyzing the integration of noise filters in the self-training approach

In this section, we analyze the results obtained in our experimental study. In particular, our aims are as follows:

- To compare the transductive capabilities achieved with the different kinds of filters under different ratios of labeled data (Section 6.1).
- To study how filtering techniques help the self-training methodology within the generalization process (Section 6.2).
- To analyze the behavior of the best filtering techniques in several data sets (Section 6.3).
- To present a global analysis of the results obtained in terms of the properties of the filtering methods (Section 6.4).

Due to the extension of the experimental analysis carried out, we report the complete experimental results on the web page associated with this paper. In this section we present summary figures and the statistical tests conducted. Tables 3–6 tabulate the information of the statistical analysis performed by nonparametric multiple comparison procedures over 10%, 20%, 30% and 40% of labeled data, respectively. In these tables, filtering methods have been sorted according to their family, starting from classic to more recent methods. In each table, we carry out a total of four statistical tests for accuracy and kappa measures, differentiating between transductive and test phases. The rankings computed, according to the FAR test [38], represent the effectiveness associated with each algorithm. The best (lowest) ranking obtained in each FAR test is marked with '∗', which determines the control algorithm for the post hoc test. Next, together with each FAR ranking, we present the adjusted p-value with Finner's test (Finner APV) based on the control algorithm. Those APVs highlighted in bold show the methods outperformed by the control, at $\alpha=0.1$ level of significance.

In these tables, we include as a baseline the NN rule trained only with labeled data (NN-L), to determine the goodness of the SSC techniques. Note that this technique corresponds to the initial stage of all the self-training schemes. However, it is also known that, depending on the problem, unlabeled data can lead to worse performance [3], hence, the inclusion of NN-L shows whether the self-training scheme is an outstanding methodology for SSC.

6.1. Transductive results

As we stated 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.

Observing Tables 3–6 we can make the following analysis:

- Considering 10% of labeled instances, the FAR procedure highlights the global filter CF as the best performing in terms of transductive learning. With this filter, self-training is able to

[2] http://sci2s.ugr.es/SelfTraining+Filters

Table 3
Average rankings of the algorithms (Friedman aligned-ranks+Finner test) over 10% labeled rate.

Algorithm	Transductive phase				Test phase			
	FAR (Accuracy)	Finner APV	FAR (Kappa)	Finner APV	FAR (Accuracy)	Finner APV	FAR (Kappa)	Finner APV
SelfTraining-ENN	416.5580	**0.0041**	390.2920	**0.0154**	421.3917	**0.0006**	397.9667	**0.0018**
SelfTraining-AllKNN	405.1160	**0.0080**	395.7580	**0.0125**	392.8583	**0.0060**	368.2917	**0.0125**
SelfTraining-RNGE	321.7920	0.3526	293.7170	0.7979	374.0583	**0.0155**	342.4333	**0.0532**
SelfTraining-MENN	364.0000	**0.0762**	408.7833	**0.0066**	371.4833	**0.0165**	405.9417	**0.0013**
SelfTraining-NCNEdit	402.9254	**0.0080**	353.3750	0.1134	424.4250	**0.0006**	395.1250	**0.0020**
SelfTraining-CEWS	330.2333	0.2775	294.9000	0.7979	379.4667	**0.0120**	367.8167	**0.0125**
SelfTraining-ENNTh	363.2833	**0.0762**	397.6667	**0.0125**	354.0833	**0.0387**	379.2000	**0.0063**
SelfTraining-Multiedit	377.2250	**0.0398**	413.2080	**0.0061**	370.5500	**0.0165**	403.9167	**0.0013**
SelfTraining-CF	**281.6174***	–	**282.8255***	–	**268.6083***	–	**261.4167***	–
SelfTraining-IPF	314.1333	0.4293	297.7164	0.7805	355.075	**0.0387**	341.5000	**0.0532**
SelfTraining	432.5500	**0.0015**	381.6421	**0.0243**	464.9833	**0.0000**	452.3417	**0.0000**
SNNRCE	345.4421	0.1577	445.5000	**0.0005**	388.3083	**0.0072**	454.2000	**0.0000**
NN-L	721.6250	**0.0000**	721.1176	**0.0000**	511.2083	**0.0000**	506.3500	**0.0000**

Table 4
Average rankings of the algorithms (Friedman aligned-ranks+Finner test) over 20% labeled rate.

Algorithm	Transductive phase				Test phase			
	FAR (Accuracy)	Finner APV	FAR (Kappa)	Finner APV	FAR (Accuracy)	Finner APV	FAR (Kappa)	Finner APV
SelfTraining-ENN	414.1167	**0.0012**	396.7750	**0.0066**	434.2333	**0.0087**	420.5583	**0.0235**
SelfTraining-AllKNN	424.6750	**0.0006**	405.3250	**0.0045**	431.6667	**0.0087**	415.0333	**0.0235**
SelfTraining-RNGE	312.8417	0.3315	306.2333	0.5485	316.0083	0.7926	319.8167	0.7618
SelfTraining-MENN	363.8083	**0.0358**	412.0750	**0.0031**	378.9333	0.1075	395.9500	**0.0530**
SelfTraining-NCNEdit	393.1583	**0.0051**	353.2250	**0.0968**	397.6000	**0.0419**	372.9000	0.1453
SelfTraining-CEWS	358.9417	**0.0391**	350.5750	0.1006	402.0917	**0.0366**	383.9083	**0.0926**
SelfTraining-ENNTh	360.5083	**0.0391**	398.9083	**0.0064**	355.1500	0.2630	367.0833	0.1731
SelfTraining-Multiedit	396.5667	**0.0045**	432.3583	**0.0008**	376.4000	0.1097	400.1250	**0.0476**
SelfTraining-CF	**270.9667***	–	**279.6167***	–	**305.1917***	–	**307.3500***	–
SelfTraining-IPF	297.7083	0.5156	287.8333	0.8417	350.3833	0.2927	343.1667	0.4105
SelfTraining	485.0083	**0.0000**	416.2333	**0.0027**	482.1667	**0.0002**	439.4583	**0.0079**
SNNRCE	545.1667	**0.0000**	599.0000	**0.0000**	436.1000	**0.0087**	494.4333	**0.0001**
NN-L	453.0333	**0.0000**	438.3417	**0.0007**	410.5750	**0.0248**	416.7167	**0.0235**

Table 5
Average rankings of the algorithms (Friedman aligned-ranks+Finner test) over 30% labeled rate.

Algorithm	Transductive phase				Test phase			
	FAR (Accuracy)	Finner APV	FAR (Kappa)	Finner APV	FAR (Accuracy)	Finner APV	FAR (Kappa)	Finner APV
SelfTraining-ENN	394.2000	**0.0384**	381.1417	0.1484	419.4167	**0.0052**	405.9500	**0.0126**
SelfTraining-AllKNN	381.6917	**0.0700**	382.6500	0.1484	388.0500	**0.0280**	378.4333	**0.0467**
SelfTraining-RNGE	341.9833	0.3260	324.3333	0.6993	327.0500	0.4380	**293.5667***	–
SelfTraining-MENN	335.3250	0.3541	357.0083	0.2833	393.7167	**0.0248**	391.3500	**0.0232**
SelfTraining-NCNEdit	403.8500	**0.0247**	379.5167	0.1484	425.5667	**0.0039**	396.8750	**0.0205**
SelfTraining-CEWS	378.8083	**0.0714**	364.6333	0.2285	408.9500	**0.0098**	411.3167	**0.0126**
SelfTraining-ENNTh	338.2333	0.3440	355.2167	0.2833	393.0417	**0.0248**	393.1583	**0.0231**
SelfTraining-Multiedit	373.8500	**0.0829**	400.2833	**0.0670**	373.5167	**0.0607**	411.1667	**0.0126**
SelfTraining-CF	302.7250	0.8561	314.1583	0.8555	**293.1833***	–	296.8750	0.9359
SelfTraining-IPF	**295.2667***	–	**306.6667***	–	320.5833	0.5054	311.7000	0.6911
SelfTraining	437.8167	**0.0021**	409.8417	**0.0477**	451.6500	**0.0014**	429.3500	**0.0039**
SNNRCE	644.0250	**0.0000**	653.7667	**0.0000**	450.2833	**0.0014**	519.5583	**0.0000**
NN-L	448.7250	**0.0011**	447.2833	**0.0038**	431.4917	**0.0031**	437.2000	**0.0029**

significantly outperform 8 of the 12 comparison techniques for the accuracy and kappa measures. The IPF filter which also belongs to the global family of filters can be stressed as an excellent filter with this labeled ratio. Furthermore, we can highlight RNGE and CEWS as the most competitive local filters in comparison with CF. The comparison technique SNNRCE is outperformed in terms of kappa measure. By contrast, considering the accuracy measure, it is not overcome with a level of significance $\alpha = 0.1$. This fact indicates that SNNRCE benefits from random hits.

- When the number of labeled instances is increased to 20%, we observe a clear improvement in terms of accuracy and kappa rate for all the studied methods. Again, global filters obtain the two best rankings and the CF filter is stressed as the best performing method. The number of techniques outperformed has also been increased. RNGE is the most relevant local filtering technique in comparison with CF and IPF.
- When the data set has a relatively higher number of labeled instances (30% or 40%), either local and global filtering techniques display similar behavior. This is because all the filtering

Table 6
Average rankings of the algorithms (Friedman aligned-ranks+Finner test) over 40% labeled rate.

Algorithm	Transductive phase				Test phase			
	FAR (Accuracy)	Finner APV	FAR (Kappa)	Finner APV	FAR (Accuracy)	Finner APV	FAR (Kappa)	Finner APV
SelfTraining-ENN	375.1500	0.2488	359.7333	0.4135	404.8000	**0.0909**	394.6000	**0.0718**
SelfTraining-AllKNN	359.7917	0.2681	352.1083	0.4401	380.9417	0.2558	378.9333	0.1449
SelfTraining-RNGE	331.3083	0.5635	324.2500	0.8323	**322.9083***	–	**308.5917***	–
SelfTraining-MENN	363.6417	0.2559	384.6250	0.2378	350.3083	0.5620	363.1917	0.2380
SelfTraining-NCNEdit	368.7833	0.2488	354.9667	0.4401	426.7917	**0.0454**	403.9000	**0.0485**
SelfTraining-CEWS	375.0833	0.2488	379.7667	0.2450	418.6333	**0.0548**	427.5667	**0.0227**
SelfTraining-ENNTh	351.8833	0.3315	367.7417	0.3477	352.9083	0.5620	362.3667	0.2380
SelfTraining-Multiedit	340.5667	0.4534	354.6833	0.4401	353.1917	0.5620	377.9667	0.1449
SelfTraining-CF	322.5083	0.6813	323.7500	0.8323	343.9083	0.6097	339.9000	0.4466
SelfTraining-IPF	**305.6167***	–	**314.1167***	–	371.1250	0.3389	354.7583	0.2818
SelfTraining	448.5083	**0.0021**	423.7333	**0.0305**	453.4750	**0.0090**	424.6833	**0.0227**
SNNRCE	676.8667	**0.0000**	680.7000	**0.0000**	477.7833	**0.0020**	523.0667	**0.0000**
NN-L	456.7917	**0.0014**	456.3250	**0.0033**	419.7250	**0.0548**	416.9750	**0.0250**

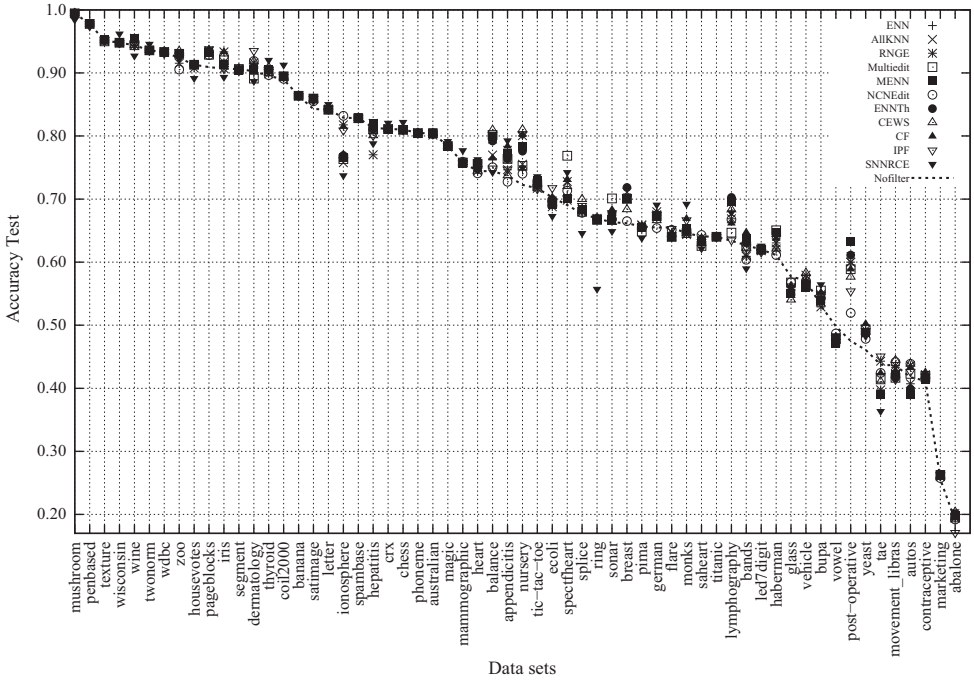

Fig. 3. Accuracy test over 10% of labeled data.

techniques are able to detect noisy examples in an easier way with a representative number of labeled data. Nevertheless, the IPF filter is outstanding as the best ranking in both kappa and accuracy measures for high labeled ratios. Note that it is able to obtain better results than the standard self-training or the baseline NN-L which shows the usefulness of the filtering process with a greater number of labeled data.

6.2. *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 analyzed methods, checking if the previous learned hypotheses are appropriate or not.

Apart from Tables 3–6, we include four figures representing the accuracy obtained by the methods in the different labeled ratios. Figs. 3 and 4 illustrate the accuracy test obtained in each data set over 10% and 40% of labeled instances. For the sake of simplicity, the figures with a 20% and 30% of labeled instances and their corresponding accuracy tables can be found on the associated web page.

The aim of these figures is to determine in which data sets the original self-training algorithm is outperformed. For this reason, we take the standard self-training as the baseline method to be overcome. For a better visualization, on the *x*-axis the data sets are ordered from the maximum to the minimum accuracy obtained by the standard self-training. The *y*-axis position is the accuracy test of each algorithm. Self-training without any filter is drawn as a line. Therefore, points above of this line correspond to data sets for which the other proposals perform better than the original algorithm.

Finally, Table 7 summarizes the main differences of each method over the basic self-training algorithm in each labeled ratio. In this table, we present the number of data sets in which the obtained accuracy and kappa rates for each technique are strictly greater (Wins) and greater or equal (Ties+Wins) than the baseline. Again, the best results in each column are highlighted in bold.

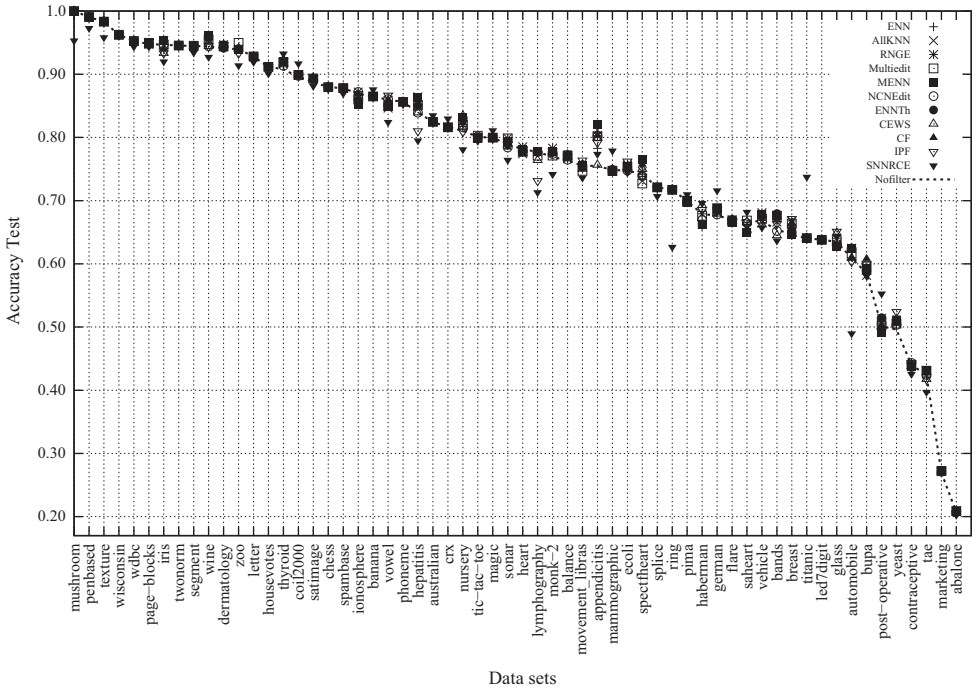

Fig. 4. Accuracy test over 40% of labeled data.

Table 7
Comparison of each method over the basic self-training approach.

Algorithm		10%		20%		30%		40%	
		Accuracy	Kappa	Accuracy	Kappa	Accuracy	Kappa	Accuracy	Kappa
SelfTraining-ENN	Wins	24	26	29	28	24	23	25	23
	Ties+Wins	40	41	49	43	45	42	49	45
SelfTraining-AllKNN	Wins	24	28	28	26	25	24	25	22
	Ties+Wins	40	43	45	42	43	41	47	42
SelfTraining-RNGE	Wins	27	32	**38**	**33**	**34**	**35**	29	**28**
	Ties+Wins	42	45	51	**47**	**53**	**52**	**52**	**49**
SelfTraining-MENN	Wins	26	26	30	30	27	25	26	26
	Ties+Wins	40	40	44	44	40	39	46	43
SelfTraining-NCNEdit	Wins	20	28	29	27	24	23	23	23
	Ties+Wins	38	46	**52**	**47**	48	47	52	50
SelfTraining-ENNTh	Wins	23	28	32	28	26	23	27	27
	Ties+Wins	38	42	48	44	42	38	45	44
SelfTraining-CEWS	Wins	26	26	24	22	19	22	20	16
	Ties+Wins	**49**	**49**	47	45	43	45	44	40
SelfTraining-Multiedit	Wins	24	24	31	28	28	24	**31**	27
	Ties+Wins	40	40	45	42	45	42	49	43
SelfTraining-CF	Wins	**34**	**35**	36	**33**	29	30	27	27
	Ties+Wins	45	47	50	46	49	46	51	**49**
SelfTraining-IPF	Wins	15	28	32	30	30	29	24	24
	Ties+Wins	31	44	48	46	50	47	46	45
SNNRCE	Wins	29	30	31	26	28	20	24	18
	Ties+Wins	29	31	32	28	29	21	24	19

Taking into account these figures, the previous statistical tests and Table 7, may make some comments to summarize:

- When the inductive learning problem is addressed with 10% of labeled data, there are significant and interesting differences between the methods. Concretely, the global CF filter is highlighted as the most promising filter. The Finner procedure signals the differences of this filter to the rest of the comparison techniques in both accuracy and kappa measures. Fig. 3 also corroborates this statement, because the majority of its points are above the baseline.
- With 20% of labeled instances, CF is still the most suitable algorithm in terms of generalization capabilities. Depending on the used measure, accuracy or kappa, the CF filter is able to obtain a significant improvement over different filters. Nevertheless, IPF, ENNTh and RNGE are established as the best performing filters which are not statistically outperformed in both accuracy and kappa measures. Table 7 shows that these methods obtain a similar number of victories and ties over the standard self-training.
- Using 30%, two main methods from different families, CF and RNGE, are the two best algorithms. CF is note-worthy in terms of accuracy test, however RNGE is established as a control algorithm in terms of the kappa measure. In both cases, the global filter IPF is still a robust filter, not clearly outperformed by the control ones.

- With 40% of labeled examples, RNGE works appropriately but it does not obtain a distinctive difference from most of the filters. By contrast, this method has significant differences in comparison with NN-L, standard self-training and SNNRCE.
- The figures reveal that when there is an increment in the labeled ratio, a notable number of points are closer to the used baseline. This means that when a considerable number of known data are available, the basic self-training algorithm is able to work fine, avoiding misclassification. Table 7 also confirms this idea, because the number of data sets in which basic self-training is outperformed, decreases over 30% and 40% of labeled data.

6.3. Analyzing the behavior of the noisy filters

The noise detection capabilities of filtering techniques determine the behavior of the self-training filtering scheme. In this subsection, we want to analyze how many instances are detected as noisy ones during the self-labeling process.

To perform this analysis, we have selected two data sets with a 10% of labeled data (we choose nursery and yeast as illustrative data sets in which the filters reach the same number of iterations in the self-training approach) and the three best filters (CF, IPF and RNGE). Table 8 collects the total number of removed instances (#RI) at the end of the self-training filtered process for each method. Furthermore, the improvement on average test results, regarding the standard self-training is also shown (Impr).

In addition, Fig. 5 shows a graphical representation of the number of removed instances in each self-training iteration for both Nursery and Yeast data sets. The *X*-axis represents the number of iterations carried out, and the *Y*-axis represents the number of instances removed in this iteration.

As we can observe in Table 8 and Fig. 5, we can establish that each method works in a different way, annotating different instances as noisy ones during the self-training process. For instance, the CF filter tends to remove a greater number of instances than IPF and RNGE. Nevertheless, Table 8 shows that this fact does not imply that its improvement capabilities are better.

In both Fig. 5(a) and (b) global filters, CF and IPF, present similar trends. We can see that this kind of filtering techniques detects analogous proportions of noisy instances according to the iterations. However, the local filter, RNGE, shows a different behavior from that of global filters.

Table 8
Removed instances for each filtering method.

Data set	SelfTraining-CF		SelfTraining-IPF		SelfTraining-RNGE	
	#RI	**Impr.**	**#RI**	**Impr.**	**#RI**	**Impr.**
Nursery	1313	0.0794	1196	0.0805	1129	0.0781
Yeast	284	0.0411	156	0.0303	201	0.0317

6.4. Global analysis

This section provides a global perspective on the obtained results. As a summary we want to outline several points about the self-training filtered approach in general and the characteristics of noise filters that are more appropriate to improve the self-training process:

- In all the studies performed, one of the self-training filtered always obtains the best results. Independent of the labeled rate and the established baselines, NN-L and self-training without a filter are clearly outperformed by at least one of these methods. This fact highlights the good synergy between noise filters and the self-training process. As we stated before, the inclusion of one erroneous example in the labeled data can alter the following stages, and therefore, the latter's generalization capabilities. Inductive results highlight the generalization abilities and the usefulness of the use self-training in conjunction with filtering techniques. Nevertheless, there are significant differences depending on the selected filter.
- Comparing transductive and test phases, we can state that, in general, the SSC methods used differ widely when tackling the inductive phase. It shows the necessity of some mechanisms, like appropriate filtering methodologies, to find robust learned hypotheses which allow the classifiers to predict unseen cases.
- In general, an increment in the labeled ratio indicates the lower benefit of the filtering techniques. It justifies the view that the use of SSC methods is more appropriate in the presence of a lower number of labeled examples. However, even in these cases, the analyzed self-training filtered algorithms can still be helpful, as has been shown in the reported results.
- The working process of one filter is an important factor in both transductive and inductive learning. In the conducted experimental study, we have observed that global methods are more robust in most of the experiments independent of the labeled

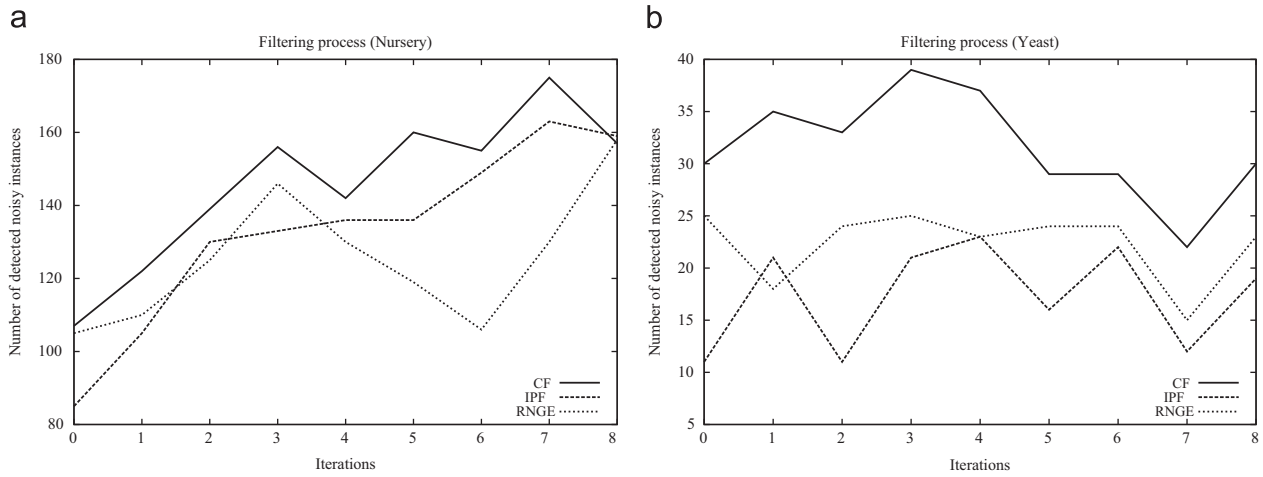

Fig. 5. Filtering process. (a) Nursery data set and (b) yeast data set.

ratio. These methods assumed that the label errors are independent of particular classifiers learned from the data, collecting predictions from different classifiers could provide a better estimation of mislabeled examples than collecting information from a single classifier only. This idea performs well under the semi-supervised learning conditions and especially when the number of labeled data is reduced. Local approaches also help to improve the self-training process, however, they are more useful when there are a higher number of labeled data. It implies that the idea of constructing a local neighborhood to determine if one instance should be considered as noise is not the most appropriate way to deal with SSC.

7. Conclusions

In this paper, we have analyzed the characteristics of a wide variety of noise filters, of a different nature, to improve the self-training approach in SSC. Most of these filters have been previously studied from a traditional supervised learning perspective. However, the filtering process can be more difficult in semi-supervised learning due to the reduced number of labeled instances.

The experimental analysis performed, supported from a statistical point of view, has allowed us to distinguish which characteristics of filtering techniques have reported a better behavior to address the transductive and inductive problems. We have checked that global filters (CF and IPF algorithms) highlight as the best performing family of filters, showing that the hypothesis agreement of several classifiers is also robust when the ratio of available labeled data is reduced. Most of the local approaches need more labeled data to perform better. The use of these filters has resulted in a better performance than that achieved by the previously proposed self-training methods, SETRED and SNNRCE.

Thus, the use of global filters is highly recommended in this field, which can be useful for further work with other SSC approaches and other base classifiers. As future work, we consider the design of new global filters for SSC that use fuzzy rough set models [58,59].

Acknowledgments

Supported by the Research Projects TIN2011-28488 and P11-TIC-7765. J.A. Sáez holds an FPU scholarship from the Spanish Ministry of Education and Science.

References

[1] E. Alpaydin, Introduction to Machine Learning, 2nd ed., MIT Press, Cambridge, MA, 2010.
[2] I.H. Witten, E. Frank, M.A. Hall, Data Mining: Practical Machine Learning Tools and Techniques, 3rd ed., Morgan Kaufmann, San Francisco, 2011.
[3] X. Zhu, A.B. Goldberg, Introduction to Semi-Supervised Learning, 1st ed., Morgan and Claypool, 2009.
[4] O. Chapelle, B. Schlkopf, A. Zien, Semi-Supervised Learning, 1st ed., The MIT Press, 2006.
[5] W. Pedrycz, Algorithms of fuzzy clustering with partial supervision, Pattern Recognition Lett. 3 (1985) 13–20.
[6] N. Seliya, T. Khoshgoftaar, Software quality analysis of unlabeled program modules with semisupervised clustering, IEEE Trans. Syst. Man Cybern. Part A: Syst. Humans 37 (2) (2007) 201–211.
[7] L. Faivishevsky, J. Goldberger, Dimensionality reduction based on non-parametric mutual information, Neurocomputing 80 (2012) 31–37.
[8] K. Chen, S. Wang, Semi-supervised learning via regularized boosting working on multiple semi-supervised assumptions, IEEE Trans. Pattern Anal. Mach. Intell. 33 (1) (2011) 129–143.
[9] A. Fujino, N. Ueda, K. Saito, Semisupervised learning for a hybrid generative/discriminative classifier based on the maximum entropy principle, IEEE Trans. Pattern Anal. Mach. Intell. 30 (3) (2008) 424–437.
[10] M. Belkin, P. Niyogi, V. Sindhwani, Manifold regularization: a geometric framework for learning from labeled and unlabeled examples, J. Mach. Learn. Res. 7 (2006) 2399–2434.
[11] A. Blum, T. Mitchell, Combining labeled and unlabeled data with co-training, in: Proceedings of the Annual ACM Conference on Computational Learning Theory, 1998, pp. 92–100.
[12] Z. Yu, L. Su, L. Li, Q. Zhao, C. Mao, J. Guo, Question classification based on co-training style semi-supervised learning, Pattern Recognition Lett. 31 (13) (2010) 1975–1980.
[13] J. Du, C.X. Ling, Z.-H. Zhou, When does co-training work in real data? IEEE Trans. Knowl. Data Eng. 23 (5) (2010) 788–799.
[14] Y. Yaslan, Z. Cataltepe, Co-training with relevant random subspaces, Neurocomputing 73 (10–12) (2010) 1652–1661.
[15] J. Xu, H. He, H. Man, Dcpe co-training for classification, Neurocomputing 86 (2012) 75–85.
[16] Z.-H. Zhou, M. Li, Tri-training: exploiting unlabeled data using three classifiers, IEEE Trans. Knowl. Data Eng. 17 (2005) 1529–1541.
[17] M. Li, Z.-H. Zhou, Improve computer-aided diagnosis with machine learning techniques using undiagnosed samples, IEEE Trans. Syst. Man Cybern., Part A: Syst. Humans 37 (6) (2007) 1088–1098.
[18] S. Sun, J. Shawe-Taylor, Sparse semi-supervised learning using conjugate functions, J. Mach. Learn. Res. 11 (2010) 2423–2455.
[19] T. Joachims, Transductive inference for text classification using support vector machines, in: Proceedings of the 16th International Conference on Machine Learning, Morgan Kaufmann, 1999, pp. 200–209.
[20] X. Tian, G. Gasso, S. Canu, A multiple kernel framework for inductive semi-supervised SVM learning, Neurocomputing 90 (2012) 46–58.
[21] A. Blum, S. Chawla, Learning from labeled and unlabeled data using graph mincuts, in: Proceedings of the 18th International Conference on Machine Learning, 2001, pp. 19–26.
[22] A. Mantrach, N. Van Zeebroeck, P. Francq, M. Shimbo, H. Bersini, M. Saerens, Semi-supervised classification and betweenness computation on large, sparse, directed graphs, Pattern Recognition 44 (6) (2011) 1212–1224.
[23] D. Yarowsky, Unsupervised word sense disambiguation rivaling supervised methods, in: Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics, 1995, pp. 189–196.
[24] Y. Wang, X. Xu, H. Zhao, Z. Hua, Semi-supervised learning based on nearest neighbor rule and cut edges, Knowl. Based Syst. 23 (6) (2010) 547–554.
[25] Y. Li, H. Li, C. Guan, Z. Chin, A self-training semi-supervised support vector machine algorithm and its applications in brain computer interface, in: ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing—Proceedings, 2007, pp. 385–388.
[26] U. Maulik, D. Chakraborty, A self-trained ensemble with semisupervised SVM: an application to pixel classification of remote sensing imagery, Pattern Recognition 44 (3) (2011) 615–623.
[27] M. Li, Z.-H. Zhou, SETRED: self-training with editing, in: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2005, pp. 611–621.
[28] F. Muhlenbach, S. Lallich, D. Zighed, Identifying and handling mislabelled instances, J. Intell. Inf. Syst. 39 (2004) 89–109.
[29] X. Wu, X. Zhu, Mining with noise knowledge: error-aware data mining, IEEE Trans. Syst. Man Cybern. Part A: Syst. Humans 38 (4) (2008) 917–932.
[30] D.R. Wilson, T.R. Martinez, Reduction techniques for instance-based learning algorithms, Mach. Learn. 38 (3) (2000) 257–286.
[31] S. García, J. Derrac, J. Cano, F. Herrera, Prototype selection for nearest neighbor classification: taxonomy and empirical study, IEEE Trans. Pattern Anal. Mach. Intell. 34 (3) (2012) 417–435.
[32] D. Gamberger, R. Boskovic, N. Lavrac, C. Groselj, Experiments with noise filtering in a medical domain, in: Proceedings of the 16th International Conference on Machine Learning, 1999, pp. 143–151.
[33] T.M. Khoshgoftaar, P. Rebours, Improving software quality prediction by noise filtering techniques, J. Comput. Sci. Technol. 22 (2007) 387–396.
[34] D. Guan, W. Yuan, Y.-K. Lee, S. Lee, Nearest neighbor editing aided by unlabeled data, Inf. Sci. 179 (13) (2009) 2273–2282.
[35] D. Guan, W. Yuan, Y.-K. Lee, S. Lee, Identifying mislabeled training data with the aid of unlabeled data, Appl. Intell. 35 (2011) 345–358.
[36] T.M. Cover, P.E. Hart, Nearest neighbor pattern classification, IEEE Trans. Inf. Theory 13 (1) (1967) 21–27.
[37] X. Wu, V. Kumar (Eds.), The Top Ten Algorithms in Data Mining, Chapman & Hall/CRC Data Mining and Knowledge Discovery, 2009.
[38] S. García, A. Fernández, J. Luengo, F. Herrera, Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: experimental analysis of power, Inf. Sci. 180 (2010) 2044–2064.
[39] N. Chawla, G. Karakoulas, Learning from labeled and unlabeled data: an empirical study across techniques and domains, J. Artif. Intell. Res. 23 (2005) 331–366.
[40] D. Gamberger, N. Lavrac, S. Dzeroski, Noise detection and elimination in data preprocessing: experiments in medical domains, Appl. Artif. Intell. 14 (2000) 205–223.
[41] D.W. Aha, D. Kibler, M.K. Albert, Instance-based learning algorithms, Mach. Learn. 6 (1) (1991) 37–66.
[42] D.L. Wilson, Asymptotic properties of nearest neighbor rules using edited data, IEEE Trans. Syst. Man Cybern. 2 (3) (1972) 408–421.
[43] I. Tomek, An experiment with the edited nearest-neighbor rule, IEEE Trans. Syst. Man Cybern. 6 (6) (1976) 448–452.

[44] J.S. Sánchez, F. Pla, F.J. Ferri, Prototype selection for the nearest neighbour rule through proximity graphs, Pattern Recognition Lett. 18 (1997) 507–513.
[45] K. Hattori, M. Takahashi, A new edited k-nearest neighbor rule in the pattern classification problem, Pattern Recognition 33 (3) (2000) 521–528.
[46] J. Sánchez, R. Barandela, A. Marques, R. Alejo, J. Badenas, Analysis of new techniques to obtain quality training sets, Pattern Recognition Lett. 24 (7) (2003) 1015–1022.
[47] B.B. Chaudhuri, A new definition of neighborhood of a point in multi-dimensional space, Pattern Recognition Lett. 17 (1) (1996) 11–17.
[48] F. Vázquez, J. Sánchez, F. Pla, A stochastic approach to Wilson's editing algorithm, in: Proceedings of the 2nd Iberian Conference on Pattern Recognition and Image Analysis, 2005, pp. 35–42.
[49] P.A. Devijver, J. Kittler, On the edited nearest neighbor rule, in: Proceedings of the Fifth International Conference on Pattern Recognition, 1980, pp. 72–80.
[50] F.J. Ferri, J.V. Albert, E. Vidal, Consideration about sample-size sensitivity of a family of edited nearest-neighbor rules, IEEE Trans. Syst. Man Cybern. 29 (4) (1999) 667–672.
[51] J.R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann Publishers, San Francisco, CA, USA, 1993.
[52] A. Ben-David, A lot of randomness is hiding in accuracy, Eng. Appl. Artif. Intell. 20 (2007) 875–885.
[53] J. Alcalá-Fdez, L. Sánchez, S. García, M.J. del Jesus, S. Ventura, J.M. Garrell, J. Otero, C. Romero, J. Bacardit, V.M. Rivas, J.C. Fernández, F. Herrera, KEEL: a software tool to assess evolutionary algorithms for data mining problems, Soft Comput. 13 (3) (2009) 307–318.
[54] J. Alcalá-Fdez, A. Fernandez, J. Luengo, J. Derrac, S. García, L. Sánchez, F. Herrera, KEEL data-mining software tool: data set repository, integration of algorithms and experimental analysis framework, J. Multiple-Valued Logic Soft Comput. 17 (2–3) (2010) 255–277.
[55] S. García, A. Fernández, J. Luengo, F. Herrera, A study of statistical techniques and performance measures for genetics-based machine learning: accuracy and interpretability, Soft Comput. 13 (10) (2009) 959–977.
[56] D.J. Sheskin, Handbook of Parametric and Nonparametric Statistical Procedures, 5th ed., Chapman & Hall/CRC, 2011.
[57] S. García, F. Herrera, An extension on "Statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons, J. Mach. Learn. Res. 9 (2008) 2677–2694.
[58] N. Mac Parthalain, R. Jensen, Fuzzy-rough set based semi-supervised learning, in: 2011 IEEE International Conference on Fuzzy Systems (FUZZ), 2011, pp. 2465–2472.
[59] J. Derrac, N. Verbiest, S. García, C. Cornelis, F. Herrera, On the use of evolutionary feature selection for improving fuzzy rough set based prototype selection, Soft Comput. 17 (2013) 223–238.

Isaac Triguero Velázquez 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, evolutionary algorithms and semi-supervised learning.

José Antonio Sáez Muñoz received his M.Sc. 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 the study of the impact of noisy data in classification, data preprocessing, fuzzy rule-based systems and imbalanced learning.

Julián Luengo Martín received his M.Sc. in Computer Science and Ph.D. from the University of Granada, Granada, Spain, in 2006 and 2011 respectively. He is currently an Assistant Professor in the Department of Civil Engineering, University of Burgos, Burgos, Spain. His research interests include machine learning and data mining, data preparation in knowledge discovery and data mining, missing values, data complexity and semi-supervised learning.

Salvador García López received his 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 Triguero 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 had more than 200 papers published in international journals. He is a coauthor of the book Genetic Fuzzy Systems: Evolutionary Tuning and Learning of Fuzzy Knowledge Bases (World Scientific, 2001).

He currently acts as an Editor in Chief of the international journal Progress in Artificial Intelligence (Springer) and serves as an Area Editor of the Journal Soft Computing (area of evolutionary and bioinspired algorithms) and International Journal of Computational Intelligence Systems (area of information systems). He acts as an Associated Editor of the journals: IEEE Transactions on Fuzzy Systems, Information Sciences, Advances in Fuzzy Systems, and International Journal of Applied Metaheuristics Computing; and he serves as a Member of several journal editorial boards, among others: Fuzzy Sets and Systems, Applied Intelligence, Knowledge and Information Systems, Information Fusion, Evolutionary Intelligence, International Journal of Hybrid Intelligent Systems, Memetic Computation, 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, and International Cajastur Mamdani Prize for Soft Computing (Fourth Edition, 2010), the 2011 IEEE Transactions on Fuzzy Systems Outstanding Paper Award and the 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, data mining, bibliometrics, data preparation, instance selection, fuzzy rule based systems, genetic fuzzy systems, knowledge extraction based on evolutionary algorithms, memetic algorithms and genetic algorithms.