An incremental self-trained ensemble algorithm

Stamatis Karlos
Department of Mathematics
University of Patras
Rio Achaia, Greece
stkarlos@upatras.gr

Nikos Fazakis
Department of Electrical and Computer Engineering
University of Patras
Rio Achaia, Greece
fazakis@ece.upatras.gr

Konstantinos Kalleris
Department of Electrical and Computer Engineering
University of Patras
Rio Achaia, Greece
kkaleris@upnet.gr

Vasileios G. Kanas
Department of Electrical and Computer Engineering
University of Patras
Rio Achaia, Greece
vaskanas@upatras.gr

Sotiris Kotsiantis
Department of Mathematics
University of Patras
Rio Achaia, Greece
sotos@math.upatras.gr

***Abstract*—Incremental learning has boosted the speed of Data Mining algorithms without sacrificing much, or sometimes none, predictive accuracy. Instead, by saving computational resources, combination of such kind of algorithms with iterative procedures that improve the learned hypothesis utilizing vast amounts of available unlabeled data could be achieved efficiently, in contrast to supervised scenario where all this information is rejected because no exploitation mechanism exists. The scope of this work is to examine the ability of a learning scheme that operates under shortage of labeled data for classification tasks, based on an incrementally updated ensemble algorithm. Comparisons against 30 state-of-the art Semi-supervised methods over 50 publicly available datasets are provided, supporting our assumptions about the learning quality of the proposed algorithm.**

Keywords—self-training scheme, incremental learners, shortage of labeled data, semi-supervised theory, ensemble base learner

I. Introduction

Despite the fact that the power of cutting edge computer systems has been increased all these years with a stable trend, the corresponding increase rate of the volume of produced data is dramatically sharper, leading to vast amounts of datasets while, simultaneously, rising issues of efficient data manipulation. Taking also into consideration the complex cases of "concept drift" [1] – where the inputs to general Machine Learning (ML) recognition systems change dynamically – and fast evolving systems [2] – which occur mainly due to the nature of the tackled application and the aspects of online learning – it is easily perceived that more and more applications need to be satisfied under both time and capacity restrictions, avoiding to mention big data era. Some usual ways of coping with such kind of tasks are: i) to increase the computational power of employed working systems [3], ii) design more efficient algorithms that could operate under such environments [4] and iii) convert effective algorithms to parallel versions [5].

In order to be more compatible with the theoretical concepts of Data Mining (DM) and Machine Learning (ML) fields, this study is oriented towards the second option from the aforementioned. Following this choice, there would not be any demands on hardware material, but all the intuitiveness would be interwoven with implementation of an incrementally updated framework that occupies an ensemble algorithm into the heart of its learning kernel. The two assets that rise from this combination are:

a) computationally simpler model: avoiding of wasting time for building from scratch the corresponding classification model each time that a new example is met and prediction is needed,

b) improved learning behavior: more accurate predictions are longed by the embedded base learner, which consists of more than one learners, providing also more robust behavior according to [6]. Moreover, all the advantages of ensemble learners are absorbed in this case, leading to a more flexible learner over generic datasets.

Tuning stages with validation subsets, intermediate optimization of learner parameters or amending design of algorithms are systematically met at the literature. However, the majority of these strategies is based on the acquisition of enough labeled data, a prerequisite that is not totally satisfied in current technological environment. The main reason why this happens, is the defect of collecting numerous labeled examples, due to either large expenses of employing human entities – experts or less specialized ones – for fulfilling the process of assigning labels to newly mined data or restricted privacy, as it happens in medical and/or economic fields. On the other hand, Semi-supervised learning (SSL) overcomes this fact by incorporating iterative tools that operate under the existence of both labeled and unlabeled subsets – from here on they will be symbolized as *L* and *U*, respectively – assessing hidden information patterns inside the *U* pool.

Although similar combinations have been proposed recently, our framework gives an additional boost over this direction, achieving decent results over 50 publicly available datasets against several state-of-the-art SSL algorithms, without achieving inferior time responses, since the simplicity of the whole process is of utmost priority. Thus, its adoption from applications that aim to support online learning without being vulnerable to shortage of large *L* is feasible, taking into consideration that collecting and storing *U* sets of great cardinality is a cheap and easily resolved task. The produced framework seems robust enough, judging by its overall behavior over a large collection of datasets stemming from various scientific fields. The internal ensemble learner is highly blamed for this behavior, along with the augmenting strategy of SSL to add the most confident instances by *U* to the training set (*L*), and this assumption is tested over two different labeled

978-1-5386-1376-4/18/$31.00 ©2018 IEEE

ratios (R), scrutinizing the learning ability of all the learning algorithms under different initial knowledge points.

The remainder of this work is organized as follows: in Section II the basic principles of Incremental Learning (IncL) and SSL are discussed briefly together with the most outstanding combinations of these two methods. Next, in Section III the specifications of the proposed learning framework are provided thoroughly. Afterwards, details of experimental procedure are given and the results are placed together with suitable statistical comparisons for discerning the significance of the obtained classification behaviors. Finally, discussion of future work points is made in last Section, summarizing the milestones of the whole paper.

II. Incremental Learning and Semi-Supervised Learning

In supervised learning, the task is to construct a model from the available data. That is, denoting the available data as T = ((*x1*, y1), (*x2*, y2), (*x3*, y3), …, (*xm*, ym)), with *xi* (symbolizes a vector) and outputs yi a model M ≈ p(y|*x*) has to be inferred from the data. Most of the proposed frameworks restrict to the classical batch setting, where data are given a priori for model training, assuming that the data and its underlying structure remain static. However, in real world scenarios and applications, such as intrusion detection systems [7],[8],[9],[10] brain computer interfaces [11],[12] financial time series forecasting [13], with de facto limited memory resources, data T are not available priorly, instead they are provided sequentially over time (online). In such cases, the task is to infer a reliable model M_t after every time step based on the example (*xt*, yt) and the previous model M_{t-1} only. Incremental learning approaches are based on constant model adaptation to incorporate additional streaming data without re-training the learning model from scratch [14], while simultaneously preserving performance.

Semi-supervised learning (SSL) [15] considers the problem of classification when only a small subset of the observations have corresponding class labels. In many applications, including image search [16], genomics [17] and speech analysis [18], unlabeled data is available in large quantity and easy to collect, but obtaining training labels is nontrivial. In SSL the goal is to combine both *L* and *U* subsets of data for achieving higher prediction accuracy than pure supervised learning or unsupervised learning. The main issue of SSL is how to exploit information from the *L* data. A number of different algorithms for SSL have been presented, such as the Expectation Maximization (EM) based algorithms [19], Self-training [20], Co-training [21], Semi- Supervised SVM (S3VM) [22], graph-based methods[23], and boosting based SSL methods [24].

An interesting separation of SSL methods is according to the number of views that are used. A view typically is a subset of the feature vector that initial dataset contains. Single-view methods exploit at once all the full set of features (*F*). The most representative scheme is Self-training that does request any assumption about the provided *F* and trains directly a base learner for augmenting the cardinality of *L* subset through instances mined from *U*. On the other hand, multi-view methods assume that specific subsets of F describe the same problem measuring quantities from different sources or describing it from other scope. Co-training splits F into 2 disjoint subsets, while other approaches make even more splits, such as RASCO [25] and Rel-RASCO [26], which expands the former by weighing the importance of features and creating random subsets of initial F over which new learners are trained. Although Co-training theoretically needs redundant views for operating efficient, TriTraining scheme [27] uses a pool of 3 learners, whose level of disagreement is applied as a criterion for selecting trustworthy unlabeled examples. Popular methods that expand the previously referred, either by inserting editing mechanisms or by exploiting ensemble classifiers, are SETRED [28], DE-TriTraining [29] and CoForest [30].

A. *Incremental semi-supervised learners*

In [31] an incremental semi-supervised Support Vector Machine (SVM) is proposed for reducing stored data and learning time over traditional semi-supervised methods. In their algorithm, F. Gao et al adopt a "soft-start" initial training phase where two classifiers (main and assistant classifier) are used to ensure classification accuracy of the semi-labeled samples, imposing strict rules not only on the distance of each semi-labeled sample to the optimal hyperplane but also to all labeled samples. When sufficient high-confidence semi-labeled samples are absorbed into the training set, the assistant classifier is abandoned and classification is carried out through the main classifier. New labeled data are utilized for error correction and performance estimation, while a data cleaning mechanism is proposed for effectively reducing the training data. In another relative work [32], incremental semi-supervised techniques are incorporated on an Extreme Learning Machine (ELM), a feedforward neural network with non-tuned and randomly initialized hidden layers. The proposed algorithm implements novelty detection using Extreme Value Theory (EVT), where extreme statistics are assumed to follow the Gumbel distribution. Through novelty detection, data that are identified as members of an unknown class are rejected, thus, mislabeling is reduced and the data added to the training set is kept to a minimum.

The IS^{3}RS algorithm proposed in [33] uses self-representation selection for choosing exemplar data within large data chunks to be used for semi-supervised training. The data available in time are split into chunks which are clustered using kNN. From the clusters, a small amount of exemplar samples is selected to be used for SSL of the core classifier. The exemplar data samples are labeled using co-training, where the data feature set is split into two dependent feature sets and the most confident exemplars classified into the same class according to the two distinct feature sets are added to the algorithm's training set. Finally, the core classifier is updated incrementally by using the newly labeled data samples.

In [34], an incremental semi-supervised learner is designed to address mixed and imbalanced data classification in real-time problems, specifically intrusion detection. Core of the algorithm is a Mixed Self-Organizing Incremental Neural Network (MSOINN) running in off-line and on-line phases. The off-line phases constitute non-real-time training phases that are run each time a new training data set is available. The data is clustered by unsupervised MSOINN, in order for the initial cluster set to be formed, and then the clusters are adjusted through the supervised Cluster Adjustment method. The online phases correspond to the real-time intrusion detection phases. New incoming data are classified through KNN and the clusters are readjusted through the incremental neural network. Finally, H.R. Loo et al. [35] propose a selective self-training method for online incremental semi-supervised classification of streaming data with limited labeled instances. After the initial training using labeled data, the algorithm is incrementally updated through selection of instances from the data stream that are classified with a high confidence level, thus reducing false learning. Further performance improvement is achieved through periodic execution of a cluster reduction step that removes outdated or unused clusters selecting only specific clusters for the classification process.

III. Implemented Algorithm

Our scope in this work is to integrate an incrementally updateable ensemble learner into a Self-training scheme, so as to both obtain accurate predictions during the exploiting stage of *U* pool, and do not spend much resources for the building of the newly refined classifier at the end of each SSL iteration. Therefore, three separate incremental learners have been embedded into the kernel of the proposed algorithm, as well as their decisions are manipulated appropriately by a simple voting scheme. Before jumping to the outer level of the whole algorithm, more details will be given here.

To be more specific, since all the three selected classifiers are probabilistic, they return a row vector of class possibilities, equal in size with the number of the predefined classes of the examined dataset. Then, average value is computed and the class for which the greater probability appeared, is considered as the true label either when mining of *U* pool happens or when assigning the predicted label during testing phase. The three selected incrementally updateable learners are the following:

a. *NB*: simple Naive Bayes algorithm [36], expressing the simple theory of Bayes about independent relationships among all the entities of F.

b. *A1DE*: (or Aggregating One-Dependence Estimators) coming from family of Bayesian classifiers, but weakening the independence assumption of NB≡A0DE [37]. Hence, it is allowed to exactly one feature to be connected with any other feature, besides the class feature that already does. Holding these one-dependence estimators, underlying patterns are revealed easier and overall operation is still time efficient.

c. *SGD*: (or Stochastic Gradient Descent) that learns various linear models through gradient descent method, but not over all the training data, since it would be much computationally expensive for large datasets, but applies this process over random samples of the *L* subset, updating the computed terms so as to converge to a minimum. Hinge loss function was used and normalization stage was disabled for being able to operate incremental operation [38].

After the training of the ensemble learner (*Ens)* over training data, a Self-training wrapper method exploits it for assigning confidence values to any instance that belongs to *U* pool. Then, instances that scored the higher confidence values are extracted from *U* and are added to current *L* subset, along with their predicted values, maintaining the numerical equilibrium of the contained classes.

```
Algorithm InSelfTr(Ens)
Parameters:
Ens – NB, A1DE and SGD
iter – number of the current iteration, $1 \le iter \le MaxIter$
Input:
 $L^0$/ $U^0$ – Initial sets of labeled/unlabeled instances
  1. Initialization
      a. Compute number of instances per class to extract ($V^{perclass}$)
      b. Train Ens over $L^0$ and set iter = 0
  2. Loop for a number of iterations until stopping criteria are satisfied
      a. Apply Ens to $U^{iter}$ and select instances with the Most Confident
         Predictions ($X_{MCP}$) per class, according to vector $V^{perclass}$
      b. Average the accuracies of Ens components for each class of $X_{MCP}$
      c. Remove $X_{MCP}$ from $U^{iter}$ and add them to $L^{iter}$ with most probable
         class label
      d. Re-train Incrementally Ens over augmented $L^{iter}$
      e. Set iter = iter + 1
Output:
 Use Ens trained on $L^{MaxIter}$ to predict the unknown test cases.
```

Fig 1. Pseudocode of proposed Incremental SSL algorithm.

Summarizing, the decision of the whole process is produced automatically, without human intervention, while the cardinality of *L* increases, assuming that the mined instances enrich the initial information. This fact highlights the importance of acquiring accurate predictions by the base learner. After a predefined number of iterations (MaxIter) the process stops and the lastly formatted training set is applied to test set. Not any filtering stages have been added to the constructed algorithm, named as *InSelfTr(Ens)*, keeping its simplicity, induced by the adopted theories and strategies. For our implementation, MaxIter has been defined equal to 40.

IV. Experimental Procedure

In this section, all the technical information about the conducted experiments are given through three separate subparagraphs. Appropriate tables and figures are provided for convenience and visualization of achieved performance.

A. Description of compared algorithms

As it concerns the Semi-supervised algorithms that were selected to be compared with the proposed algorithm, for examining its learning behavior, they were extracted through KEEL (Knowledge Extraction Evolution Learning) environment [39]. This ML tool constitutes a flexible enough solution for executing SSL experiments, since a large variety of algorithms are included, as they are described in depth in [40]. These four well-known base learners are the following:

a) *C4.5*: (or J48 as renamed through its implementation in JAVA) [41],

b) *kNN*: this option refers to the *k*-Nearest-Neighbors algorithm. The choice of *k* has been fixed equal to 3 [42],

c) *NB*: simple Naive Bayes algorithm [36],

Co-financed by the European Social Fund and Greek national funds.

d) *SMO (Sequential Minimal Optimization)*: it follows the theory of Support Vector Machines (SVMs) algorithms [43]. Here, polynomial kernel of first degree is the default.

The SSL schemes that produce differentiated algorithms by using each one of the above base learners, and were contained into our comparisons, are Self-training, Co-training, Tri-training, DE-TriTraining, RASCO and Rel-RASCO. Besides these 24 algorithms, the rest 6 algorithms that follow their more sophisticated learning strategy, independently of the previously referred choices of base learners, are the SETRED and APSSC algorithms, as were seen in Section 2, CoForest along with ADE-CoForest [44] – the later algorithm constitutes a variant of the former, since an editing technique named RemoveOnly [45] has been employed for guarantying better predictions together with other assumptions – CLCC (Classification algorithm based on Local Cluster Centers) [46] that exploits local clustering principle for acquiring better insights of the underlying patterns and finally, SNNRCE (Self-training Nearest Neighbor Rule using Cut Edges) [47] approach that applies graph-based rules for avoiding mislabeled examples during the early stages of learning process.

Without covering all the details that accompany each included method, some basic available options are placed here:

a) Maximum number of iterations: set equal to 40, explaining the reason why the same number was defined also in the proposed learning scheme,

b) Number of views: this choice is applicable to multi-view algorithms that build a number of classifiers per each supplied view.

B. *Classification accuracy results*

Having selected the 30 different SSL approaches that are going to be compared with *InSelfTr(Ens)* algorithm, the next step is about the datasets that will get scrutinized by them. For the sake of reproducibility, the extracted classification datasets come from UCI and/or KEEL repositories and are related with various scientific fields. Thus, the obtained performances essentially cover a wide range of applications, enforcing our ambition to highlight the generalization ability of the proposed algorithm as a useful DM tool.

For the context of the current work, two different values of R parameter – ratio of labeled against total amount of instances – have been selected: 10% and 20%. Table III and Table IV show the performances of the best 10 algorithms, including the *InSelfTr(Ens)*, for all the datasets. The characterization "best" comes from the statistical ranking that is placed in the next subparagraph. The best achieved accuracy per dataset/line is highlighted through bold format. All the executed experiments could be found in http://ml.math.upatras.gr/wp-content/uploads/2018/03/results_InSelfTrEns_EAIS_2018.xlsx.

C. *Statistical testing*

Although the comparison of the pure classification accuracies could offer an initial opinion about the contained entities, it still seems less powerful than statistical testing procedures. For this reason, verification of the learning ability of the proposed algorithm is needed through a statistical testing process [48]. Regarding the Friedman rankings in both cases of R-valued scenarios, *InSelfTr(Ens)* achieved the best score, while TriTraining (C4.5) was ranked as the second best. Remarkable performance was also recorded for CoForest, Self-Training (C4.5) and De-TriTraining (C4.5) algorithms, as it concerns the individual behaviors of the learners. Table I and Table II contain only the top-5 ranked algorithms per case.

Despite this fact, the decent achieved results are not underestimated, since there are several subsets of datasets in which *InSelfTr(Ens)* outperforms all its 30 rivals SSL algorithms, but this is not the point of this work. In contrast, the better the generalization ability of the proposed algorithm, the larger its applicability. An even more important remark is the fact that all these competitive algorithms which scored similar classification performances, but still inferior against the proposed, are built by combining more complex SSL schemes than the simplified Self-training choice of *InSelfTr(Ens)* and are at the same time based on predictions that are exported by DTs-learners, which are not compatible with the property of incremental updating, or Lazy learners, which consume useful time during testing phase. Thus, much more computational resources are spent inside the learning kernel of these methods – without being applicable for online learning – and, of course, more complicated structures have been employed for reaching the same predictive quality of *InSelfTr(Ens)*.

It worth mentioning that each component of the embedded learners into the proposed algorithm was tested as it concerns both its classification accuracy and its overall time performance over all the 50 contained datasets against their non-incremental versions, and no accuracy loss was found, while extreme reduction of spent time was recorded. These results enforce our assumptions for proposing this kind of combination, supporting also concept of IncL, but due to lack of space, more specified results are not included here.

TABLE I. STATISTICAL COMPARISONS OF EXAMINED SSL ALGORITHMS FOR R = 10%

Table Head	**Two-stage statistical test**		
	Friedman Ranking	***Holm's test p-values***	***Adjusted p-values***
InSelfTr (Ens)	8	-	-
TriTraining (C45)	10.13	0.241461	0.05
CoForest	11.56	0.050261	0.025
Self-Training (C45)	11.83	0.035185	0.016667
ADE-CoForest	13.41	0.002929	0.0125

TABLE II. STATISTICAL COMPARISONS OF EXAMINED SSL ALGORITHMS FOR FOR R = 20%

Table Head	**Two-stage statistical test**		
	Friedman Ranking	***Holm's test p-values***	***Adjusted p-values***
InSelfTr (Ens)	7.94	-	-
TriTraining (C45)	10.2	0.213929	0.05
Co-Training (C45)	10.72	0.126315	0.025
CoForest	11.04	0.088237	0.016667
Self-Training (C45)	11.41	0.056359	0.0125

TABLE III. CLASSIFICATION ACCURACY FOR R = 10%

Data-sets	InSelfTr (Ens)	TriTraining (C45)	CoForest	Self-Training (C45)	ADE-CoForest	DE-Tri Training (C45)	DE-Tri Training (3NN)	Co-Training (C45)	SETRED	DE-Tri Training (SMO)
abalone	**0.239**	0.216	0.210	0.204	0.234	0.228	0.225	0.211	0.204	0.227
appendicitis	0.823	0.805	0.823	**0.832**	0.812	0.812	0.812	**0.832**	0.737	0.802
australian	**0.854**	0.845	0.841	0.828	0.833	0.830	0.839	0.835	0.804	0.832
automobile	0.417	0.389	**0.457**	0.405	0.407	0.414	0.425	0.373	0.439	0.401
breast	0.704	0.722	**0.734**	0.722	0.720	0.712	0.716	0.677	0.684	0.686
bupa	0.573	0.574	**0.585**	0.539	0.554	0.578	0.542	0.574	0.534	0.584
chess	0.909	**0.958**	0.944	0.954	0.835	0.912	0.776	0.952	0.810	0.871
cleveland	0.564	0.476	0.537	0.525	0.543	0.486	0.536	**0.574**	0.529	0.515
coil2000	0.901	0.936	0.930	0.937	0.934	0.938	0.920	**0.940**	0.893	0.880
contraceptive	0.447	0.481	0.485	**0.489**	0.439	0.441	0.432	0.446	0.415	0.428
crx	0.836	0.856	0.821	**0.866**	0.839	0.851	0.829	0.816	0.811	0.848
dermatology	**0.944**	0.882	0.905	0.856	0.857	0.859	0.859	0.843	0.918	0.738
ecoli	**0.700**	0.659	0.628	0.647	0.670	0.611	0.628	0.577	0.694	0.616
flare	0.714	0.716	0.402	**0.721**	0.365	0.697	0.615	0.574	0.645	0.643
german	0.699	**0.717**	0.686	0.706	0.688	0.703	0.689	0.690	0.666	0.697
haberman	0.725	0.709	0.601	0.705	0.660	**0.726**	0.683	0.719	0.621	0.689
heart	0.796	0.715	0.693	0.678	0.785	0.793	**0.811**	0.700	0.744	0.819
hepatitis	**0.847**	0.834	0.811	0.834	0.834	0.834	0.834	0.834	0.801	0.834
housevotes	0.903	0.916	0.922	**0.941**	0.904	0.864	0.894	0.823	0.913	0.904
iris	0.887	0.727	**0.933**	0.840	0.920	0.920	0.913	0.847	0.913	0.893
led7digit	0.630	0.604	**0.634**	0.614	0.628	0.590	0.630	0.514	0.618	0.514
lymphography	**0.790**	0.612	0.646	0.631	0.707	0.657	0.672	0.573	0.683	0.606
magic	0.799	0.825	**0.844**	0.822	0.814	0.814	0.788	0.820	0.784	0.826
mammographic	0.812	**0.818**	0.794	0.803	0.786	0.797	0.788	0.807	0.758	0.781
marketing	0.278	0.269	0.292	0.285	**0.298**	0.287	0.276	0.280	0.260	0.267
movement_libras	**0.353**	0.275	0.311	0.256	0.211	0.169	0.281	0.236	0.444	0.181
mushroom	**0.998**	0.996	0.908	0.997	0.908	0.991	0.992	0.997	0.995	0.991
page-blocks	0.932	0.956	**0.959**	0.952	0.940	0.937	0.937	0.949	0.936	0.934
penbased	0.910	0.903	0.955	0.892	0.958	0.894	0.974	0.896	**0.978**	0.976
phoneme	0.774	0.777	0.801	0.777	0.790	0.777	0.794	0.765	0.805	**0.815**
pima	**0.724**	0.656	0.663	0.664	0.679	0.655	0.673	0.670	0.657	0.675
ring	**0.967**	0.854	0.882	0.840	0.610	0.666	0.575	0.837	0.669	0.619
saheart	**0.691**	0.678	0.656	0.652	0.684	0.682	0.665	0.636	0.630	0.675
satimage	0.847	0.822	0.860	0.805	0.858	0.815	**0.861**	0.806	0.857	0.854
segment	0.870	0.900	0.903	0.890	0.885	0.881	0.881	0.902	**0.907**	0.896
sonar	0.702	0.702	**0.755**	0.643	0.625	0.620	0.645	0.582	0.663	0.639
spambase	0.912	0.881	**0.919**	0.867	0.859	0.855	0.789	0.888	0.828	0.826
spectfheart	0.742	0.757	**0.775**	0.682	0.768	0.720	0.701	0.724	0.720	0.686
splice	**0.885**	0.825	0.507	0.827	0.484	0.793	0.621	0.831	0.700	0.700
tae	0.396	**0.457**	0.378	0.424	0.404	0.384	0.417	0.311	0.411	0.397
texture	0.891	0.852	0.907	0.831	0.912	0.837	0.937	0.829	**0.951**	0.947
thyroid	0.960	0.992	0.986	**0.992**	0.929	0.932	0.927	**0.992**	0.909	0.928
tic-tac-toe	**0.786**	0.709	0.597	0.711	0.625	0.690	0.690	0.693	0.726	0.667
titanic	0.775	0.777	0.707	0.775	0.619	0.691	0.691	**0.778**	0.640	0.692
twonorm	**0.976**	0.862	0.899	0.814	0.916	0.837	0.927	0.809	0.936	0.971
vehicle	0.536	**0.619**	0.612	0.579	0.571	0.553	0.557	0.575	0.583	0.551
wine	**0.944**	0.820	0.859	0.741	0.887	0.932	0.933	0.808	0.944	0.922
wisconsin	0.967	0.931	0.936	0.909	0.956	0.945	0.955	0.906	0.948	**0.968**
yeast	**0.514**	0.491	0.456	0.462	0.464	0.502	0.485	0.489	0.489	0.496
zoo	**0.923**	0.719	0.909	0.679	0.579	0.569	0.579	0.636	0.935	0.568

TABLE IV. CLASSIFICATION ACCURACY FOR R = 20%

Data-sets	**InSelfTr (Ens)**	**TriTraining (C45)**	**Co-Training (C45)**	**CoForest**	**Self-Training (C45)**	**DE-Tri Training (C45)**	**DE-Tri Training (SMO)**	**TriTraining (SMO)**	**Co-Training (SMO)**	**ADE-CoForest**
abalone	**0.251**	0.203	0.201	0.214	0.202	0.238	0.237	0.227	0.229	0.232
appendicitis	**0.879**	0.832	0.850	0.877	0.851	0.841	0.851	0.803	0.833	0.841
australian	**0.864**	0.852	0.838	0.842	0.858	0.859	0.855	0.844	0.830	0.859
automobile	0.520	0.494	0.480	**0.547**	0.516	0.412	0.402	0.399	0.405	0.454
breast	0.737	0.723	0.723	0.727	0.716	**0.748**	0.696	0.649	0.608	0.689
bupa	0.548	0.620	0.606	0.602	0.606	0.559	0.594	**0.646**	0.630	0.577
chess	0.935	**0.978**	0.976	0.951	**0.978**	0.933	0.913	0.946	0.951	0.883
cleveland	**0.567**	0.540	0.527	0.533	0.493	0.513	0.543	0.513	0.487	0.543
coil2000	0.885	0.937	**0.940**	0.927	**0.940**	0.937	0.892	0.690	0.821	0.936
contraceptive	0.486	0.488	**0.504**	0.494	0.478	0.450	0.449	0.471	0.479	0.461
crx	0.850	0.854	0.850	0.841	0.857	0.856	0.866	**0.870**	0.850	0.839
dermatology	**0.966**	0.921	0.879	0.924	0.910	0.874	0.905	0.891	0.876	0.939
ecoli	**0.762**	0.721	0.733	0.748	0.720	0.724	0.718	0.717	0.700	0.703
flare	**0.733**	0.727	0.690	0.416	0.728	0.708	0.661	0.633	0.602	0.400
german	**0.732**	0.684	0.699	0.684	0.697	0.708	0.683	0.658	0.641	0.693
haberman	0.725	0.709	**0.735**	0.640	0.709	0.732	0.706	0.630	0.663	0.671
heart	**0.856**	0.793	0.756	0.730	0.752	0.756	0.819	0.785	0.778	0.752
hepatitis	0.831	0.843	0.843	**0.861**	0.843	0.834	0.834	0.834	0.834	0.834
housevotes	0.911	**0.959**	0.936	0.945	0.959	0.910	0.908	0.897	0.882	0.921
iris	0.887	0.880	0.893	0.940	0.893	**0.960**	**0.960**	0.920	0.940	0.913
led7digit	0.668	0.676	0.644	**0.678**	**0.678**	0.686	0.606	0.610	0.622	0.662
lymphography	**0.803**	0.755	0.775	0.693	0.706	0.746	0.793	0.630	0.698	0.711
magic	0.800	0.834	0.832	0.852	0.830	0.825	0.839	0.845	**0.853**	0.822
mammographic	**0.826**	0.816	0.823	0.787	0.823	0.809	0.797	0.791	0.797	0.801
marketing	0.282	0.284	0.290	0.292	0.289	0.293	0.270	0.260	0.267	**0.306**
movement_libras	**0.467**	0.383	0.392	0.461	0.347	0.289	0.289	0.361	0.219	0.342
mushroom	**0.999**	**0.999**	**0.999**	0.910	**0.999**	0.997	0.996	0.998	0.997	0.909
page-blocks	0.941	**0.961**	**0.961**	0.959	0.960	0.945	0.943	0.957	0.957	0.944
penbased	0.927	0.929	0.925	0.966	0.924	0.929	0.985	0.991	**0.992**	0.972
phoneme	0.782	0.798	0.802	0.831	0.784	0.803	0.838	0.847	**0.850**	0.816
pima	**0.741**	0.694	0.687	0.711	0.681	0.685	0.672	0.634	0.656	0.686
ring	0.971	0.880	0.863	0.893	0.866	0.714	0.670	**0.972**	0.971	0.676
saheart	**0.704**	0.678	**0.704**	0.671	0.652	0.634	0.634	0.626	0.604	0.684
satimage	0.852	0.835	0.826	**0.872**	0.824	0.832	0.870	0.871	0.869	0.869
segment	0.902	0.925	0.929	0.934	0.926	0.914	0.926	0.936	**0.949**	0.916
sonar	0.731	0.663	0.634	**0.745**	0.664	0.736	0.692	0.696	0.716	0.672
spambase	0.917	0.894	0.891	**0.931**	0.891	0.870	0.871	0.901	0.901	0.884
spectfheart	0.735	0.749	0.761	**0.791**	0.719	0.743	0.675	0.723	0.674	0.772
splice	**0.926**	0.884	0.879	0.521	0.883	0.829	0.902	0.917	0.890	0.497
tae	0.410	0.430	0.423	0.457	0.423	0.437	**0.463**	0.437	0.376	**0.463**
texture	0.913	0.893	0.863	0.935	0.867	0.874	0.967	0.984	**0.987**	0.935
thyroid	0.970	**0.994**	0.993	0.988	0.994	0.937	0.932	0.939	0.940	0.933
tic-tac-toe	**0.811**	0.751	0.721	0.607	0.757	0.693	0.710	0.702	0.722	0.652
titanic	0.778	0.782	0.782	0.723	0.782	0.692	0.696	0.781	**0.783**	0.644
twonorm	**0.977**	0.867	0.828	0.903	0.817	0.832	0.973	0.975	0.973	0.915
vehicle	0.558	0.662	0.649	0.655	0.649	0.599	0.623	0.667	**0.693**	0.638
wine	**0.960**	0.842	0.786	0.842	0.837	0.893	0.921	0.927	0.956	0.916
wisconsin	0.961	0.925	0.934	0.936	0.934	0.946	**0.965**	0.950	0.949	0.951
yeast	**0.563**	0.549	0.549	0.489	0.526	0.545	0.555	0.517	0.512	0.505
zoo	**0.906**	0.826	0.734	0.897	0.831	0.787	0.626	0.538	0.628	0.762

V. Conclusions

Summarizing, this paper implements an incrementally updateable Semi-supervised classifier, based on an ensemble base learner and using Self-training scheme for exploiting pool of unlabeled data, named as *InSelfTr(Ens)*. Its asset to operate in incremental mode, instead of batch, provides much saved computational resources. Thus, an iterative SSL scheme has been used to exploit this property, mining – with an accurate and robust ensemble learner – unlabeled instances and providing them to initially quite few training data. Therefore, our proposed algorithm was examined only in low labeled ratio values (R) and managed to outperform 30 SSL algorithms that are created by state-of-the-art SSL schemes and well-known base learners on 50 generic datasets.

Although no generic results are easy to be drawn, the proposed algorithm tends to perform good classification behavior when binary-class datasets with a few instances are met, as well as multi-class datasets with small number of classes and modest or small cardinality of instances are provided. It worth mentioning that *InSelfTr(Ens)* performed the more robust behavior, since its emphatic fluctuations were numerically restricted enough and occurred in their majority during too small datasets. Another remarkable point is that algorithms based on SMO were favored by the increase of R value, in contrast with their corresponding approaches based on 3NN. Robust enough behavior was acquired by the choice of DTs as base learner, which are not applicable for incremental applications, scoring still weaker overall classification accuracy against the proposed. This fact verifies our choice of the embedded learners into such an ensemble structure, but its performance inside other SSL schemes should be investigated in the future.

Some evident expansions of the proposed learning framework that may lead to improved predictive behavior, taking of course advantage of the reduced demanded training time, constitute the manipulation of the provided feature vector, either by holding the most informative, based on appropriate feature selection methods inside SSL scheme [49], or by introducing feature weighting stages that facilitate the extraction of underlying relationships. In case that such methods could be harmonized with the incremental property of the embedded learner, then boosting of predictive ability could be achieved even for online learning scenarios [50], [51].

Besides the well-known methods of these procedures, more oriented directions could be inserted as pre-process stages inside learning procedure analog to the nature of the tackled dataset [52]. Great asset, would be the utilization of incrementally updateable algorithms along with such incrementally pre-processing methods, especially for real-life applications which tackle vast amounts of data, like social media analysis or profile characterization [53].

Finally, combination of SSL strategy together with Active Learning, supporting at the same time the incremental construction of the embedded learning kernel, could be proven an excellent solution, considering both accuracy and demanded time complexity, since the knowledge of human factor would be unified along with the automated nature of SSL, identifying probably hidden relationships and patterns with increased efficacy [54].

Acknowledgment

This research project (MIS) 5005767 has been co-financed by the Operational Program "Human Resources Development, Education and Lifelong Learning" and is co-financed by the European Union (European Social Fund) and Greek national funds.

References

[1] B. Mallick, D. Garg, and P. S. Grover, "Incremental mining of sequential patterns: Progress and challenges," *Intell. Data Anal.*, vol. 17, no. 3, pp. 507–530, 2013.

[2] H. R. Loo, S. B. Joseph, and M. N. Marsono, "Online Incremental Learning for High Bandwidth Network Traffic Classification," *Appl. Comput. Intell. Soft Comput.*, vol. 2016, 2016.

[3] N. Altanaite and J. Langguth, "GPU-based Acceleration of Detailed Tissue-Scale Cardiac Simulations," in *Proceedings of the 11th Workshop on General Purpose GPUs - GPGPU-11*, 2018, pp. 31–38.

[4] J.-P. Huang, S.-J. Chen, and H.-C. Kuo, "An efficient incremental mining algorithm-QSD," *Intell. Data Anal.*, vol. 11, no. 3, pp. 265–278, Jan. 2007.

[5] B. E. Moutafis, C. K. Filelis-Papadopoulos, P. E. Kyziropoulos, and G. A. Gravvanis, "Parallel multi-projection type methods on hybrid CPU/MIC cluster," *ACM Int. Conf. Proceeding Ser.*, vol. Part F1325, pp. 1–6, 2017.

[6] L. I. Kuncheva, *Combining Pattern Classifiers: Methods and Algorithms: Second Edition*. 2014.

[7] A. A. Aburomman and M. Bin Ibne Reaz, "A novel SVM-kNN-PSO ensemble method for intrusion detection system," *Appl. Soft Comput.*, vol. 38, pp. 360–372, Jan. 2016.

[8] M.-J. Kang and J.-W. Kang, "Intrusion Detection System Using Deep Neural Network for In-Vehicle Network Security," *PLoS One*, vol. 11, no. 6, p. e0155781, Jun. 2016.

[9] J. Peng, K.-K. R. Choo, and H. Ashman, "User profiling in intrusion detection: A review," *J. Netw. Comput. Appl.*, vol. 72, pp. 14–27, Sep. 2016.

[10] I. Butun, S. D. Morgera, and R. Sankar, "A Survey of Intrusion Detection Systems in Wireless Sensor Networks," *IEEE Commun. Surv. Tutorials*, vol. 16, no. 1, pp. 266–282, 2014.

[11] K. Vo, D. N. Nguyen, H. H. Kha, and E. Dutkiewicz, "Subject-Independent P300 BCI Using Ensemble Classifier, Dynamic Stopping and Adaptive Learning," in *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, 2017, pp. 1–7.

[12] F. Lotte *et al.*, "A Review of Classification Algorithms for EEG-based Brain-Computer Interfaces: A 10-year Update," *J. Neural Eng.*, Feb. 2018.

[13] C.-J. Lu, T.-S. Lee, and C.-C. Chiu, "Financial time series forecasting using independent component analysis and support vector regression," *Decis. Support Syst.*, vol. 47, no. 2, pp. 115–125, May 2009.

[14] P. Laskov and F. K. Stefan uger KRUEGERS, "Incremental Support Vector Learning: Analysis, Implementation and Applications Christian Gehl Klaus-Robert uller," *J. Mach. Learn. Res.*, vol. 7, pp. 1909–1936, 2006.

[15] X. Zhu and A. B. Goldberg, *Introduction to Semi-Supervised Learning*, vol. 3, no. 1. Morgan & Claypool, 2009.

[16] R. Fergus, Y. Weiss, and A. Torralba, "Semi-Supervised Learning in Gigantic Image Collections", *Adv. Neural Inf. Process. Syst. 22*, pp.

522–530, 2009.

[17] M. Shi and B. Zhang, “Semi-supervised learning improves gene expression-based prediction of cancer recurrence,” *Bioinformatics*, vol. 27, no. 21, pp. 3017–3023, Nov. 2011.

[18] Y. Liu and K. Kirchhoff, “Graph-Based Semi-supervised Learning for Phone and Segment Classification,” in *INTERSPEECH*, 2013, pp. 1840–1843.

[19] Y. Gao, J. Ma, and A. L. Yuille, “Semi-Supervised Sparse Representation Based Classification for Face Recognition With Insufficient Labeled Samples,” *IEEE Trans. Image Process.*, vol. 26, no. 5, pp. 2545–2560, May 2017.

[20] Y. Li, C. Guan, H. Li, and Z. Chin, “A self-training semi-supervised SVM algorithm and its application in an EEG-based brain computer interface speller system,” *Pattern Recognit. Lett.*, vol. 29, no. 9, pp. 1285–1294, Jul. 2008.

[21] S. Samiappan and R. J. Moorhead, “Semi-supervised co-training and active learning framework for hyperspectral image classification,” in *2015 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, 2015, pp. 401–404.

[22] Y.-F. Li and Z.-H. Zhou, “Improving Semi-Supervised Support Vector Machines through Unlabeled Instances Selection *,” in *AAAI*, 2011.

[23] Y. Zhao, R. Ball, J. Mosesian, J.-F. de Palma, and B. Lehman, “Graph-Based Semi-supervised Learning for Fault Detection and Classification in Solar Photovoltaic Arrays,” *IEEE Trans. Power Electron.*, vol. 30, no. 5, pp. 2848–2858, May 2015.

[24] P. K. Mallapragada, R. Rong Jin, A. K. Jain, and Y. Yi Liu, “SemiBoost: Boosting for Semi-Supervised Learning,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 11, pp. 2000–2014, Nov. 2009.

[25] J. Wang, S. Luo, and X. Zeng, “A random subspace method for co-training,” in *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, 2008, pp. 195–200.

[26] Y. Yaslan and Z. Cataltepe, “Co-training with relevant random subspaces,” *Neurocomputing*, vol. 73, no. 10–12, pp. 1652–1661, Jun. 2010.

[27] Z.-H. Zhou and M. Li, “Tri-training: exploiting unlabeled data using three classifiers,” *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 11, pp. 1529–1541, Nov. 2005.

[28] M. Li and Z. Zhou, “SETRED : Self-training with Editing,” *LNAI*, vol. 3518, pp. 611–621, 2005.

[29] C. Deng and M. Z. Guo, “Tri-training and Data Editing Based Semi-supervised Clustering Algorithm,” Springer, Berlin, Heidelberg, 2006, pp. 641–651.

[30] M. Li and Z. H. Zhou, “Improve computer-aided diagnosis with machine learning techniques using undiagnosed samples,” *IEEE Trans. Syst. Man, Cybern. Part ASystems Humans*, vol. 37, no. 6, pp. 1088–1098, 2007.

[31] F. Gao, J. Mei, J. Sun, J. Wang, E. Yang, and A. Hussain, “A novel classification algorithm based on incremental semi-supervised support vector machine,” *PLoS One*, vol. 10, no. 8, pp. 1–19, 2015.

[32] H. Al-Behadili, A. Grumpe, C. Dopp, and C. Wohler, “Extreme learning machine based novelty detection for incremental semi-supervised learning,” *2015 Third Int. Conf. Image Inf. Process.*, pp. 230–235, 2015.

[33] Z. Feng, M. Wang, S. Yang, and L. Jiao, “Incremental Semi-Supervised classification of data streams via self-representative selection,” *Appl. Soft Comput. J.*, vol. 47, pp. 389–394, 2016.

[34] F. Noorbehbahani, A. Fanian, R. Mousavi, and H. Hasannejad, “An incremental intrusion detection system using a new semi-supervised stream classification method,” *Int. J. Commun. Syst.*, vol. 30, no. 4, pp. 1–26, 2017.

[35] H. R. Loo and M. N. Marsono, “Online data stream classification with incremental semi-supervised learning,” *Proc. Second ACM IKDD Conf. Data Sci. - CoDS ’15*, vol. 0, pp. 132–133, 2015.

[36] G. H. G. John and P. Langley, “Estimating Continuous Distributions in Bayesian Classifiers,” *Proc. Elev. Conf. Uncertain. Artif. Intell. Montr. Quebec, Canada*, vol. 1, pp. 338--345, 1995.

[37] G. I. Webb, J. R. Boughton, and Z. Wang, “Not so naive Bayes: Aggregating one-dependence estimators,” *Mach. Learn.*, vol. 58, no. 1, pp. 5–24, 2005.

[38] J. H. Friedman, “Stochastic gradient boosting,” *Comput. Stat. Data Anal.*, vol. 38, no. 4, pp. 367–378, Feb. 2002.

[39] J. Alcalá-Fdez *et al.*, “KEEL data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework,” *J. Mult. Log. Soft Comput.*, vol. 17, no. 2–3, pp. 255–287, 2011.

[40] I. Triguero, S. García, and F. Herrera, “Self-labeled techniques for semi-supervised learning: taxonomy, software and empirical study,” *Knowl. Inf. Syst.*, pp. 1–40, 2013.

[41] J. R. (John R. Quinlan, *C4.5 : programs for machine learning*. Morgan Kaufmann Publishers, 1993.

[42] G. Guo, H. Wang, D. Bell, Y. Bi, and K. Greer, “kNN Model-Based Approach in Classification,” *Move to Meaningful Internet Syst. 2003 CoopIS, DOA, ODBASE*, vol. 2888, pp. 986–996, 2003.

[43] S. S. Keerthi, S. K. Shevade, C. Bhattacharyya, and K. R. K. Murthy, “Improvements to Platt’s SMO Algorithm for SVM Classifier Design,” *Neural Comput.*, vol. 13, no. 3, pp. 637–649, Mar. 2001.

[44] C. Deng and M. Z. Guo, “A new co-training-style random forest for computer aided diagnosis,” *J. Intell. Inf. Syst.*, vol. 36, no. 3, pp. 253–281, 2011.

[45] Y. Jiang and Z.-H. Zhou, “Editing Training Data for kNN Classifiers with Neural Network Ensemble,” Springer, Berlin, Heidelberg, 2004, pp. 356–361.

[46] T. Huang, Y. Yu, G. Guo, and K. Li, “A classification algorithm based on local cluster centers with a few labeled training examples,” *Knowledge-Based Syst.*, vol. 23, no. 6, pp. 563–571, 2010.

[47] Y. Wang, X. Xu, H. Zhao, and Z. Hua, “Semi-supervised learning based on nearest neighbor rule and cut edges,” *Knowledge-Based Syst.*, vol. 23, no. 6, pp. 547–554, 2010.

[48] J. Demšar, “Statistical Comparisons of Classifiers over Multiple Data Sets,” *J. Mach. Learn. Res.*, vol. 7, pp. 1–30, 2006.

[49] B.-J. Choi, K.-R. Kim, K.-D. Cho, C. Park, and J.-Y. Koo, “Variable Selection for Naive Bayes Semisupervised Learning,” *Commun. Stat. - Simul. Comput.*, vol. 43, no. 10, pp. 2702–2713, 2014.

[50] J. Lu, Y. Yang, and G. Webb, “Incremental discretization for naive-bayes classifier,” *Adv. Data Min. Appl.*, pp. 223–238, 2006.

[51] Y. J. Minho and L. Seiichi, “A real-time personal authentication system based on incremental feature extraction and classification of audiovisual information,” *Evol. Syst.*, vol. 2, no. 4, pp. 261–272, 2011.

[52] L. Zhang, L. Jiang, C. Li, and G. Kong, “Two feature weighting approaches for naive Bayes text classifiers,” *Knowledge-Based Syst.*, vol. 100, pp. 137–144, 2016.

[53] G. Rizos, S. Papadopoulos, and Y. Kompatsiaris, “Predicting News Popularity by Mining Online Discussions,” *Proc. 25th Int. Conf. Companion World Wide Web*, pp. 737–742, 2016.

[54] Ł. Korycki and B. Krawczyk, “Combining Active Learning and Self-Labeling for Data Stream Mining,” in *International Conference on Computer Recognition Systems*, 2017, pp. 481–490.