# Self-Train LogitBoost for Semi-supervised Learning

Stamatis Karlos[1(✉)], Nikos Fazakis[2], Sotiris Kotsiantis[1], and Kyriakos Sgarbas[2]

[1] Department of Mathematics, University of Patras, Patras, Greece
{stkarlos,kotsiantis}@upatras.gr
[2] Department of Electrical and Computer Engineering, University of Patras, Patras, Greece
fazakis@ece.upatras.gr, sgarbas@upatras.gr

**Abstract.** Semi-supervised classification methods are based on the use of unlabeled data in combination with a smaller set of labeled examples, in order to increase the classification rate compared with the supervised methods, in which the total training is executed only by the usage of labeled data. In this work, a self-train Logitboost algorithm is presented. The self-train process improves the results by using the accurate class probabilities for which the Logitboost regression tree model is more confident at the unlabeled instances. We performed a comparison with other well-known semi-supervised classification methods on standard benchmark datasets and the presented technique had better accuracy in most cases.

**Keywords:** Semi-supervised learning · Logitboost · Classification method · Labeled and/or unlabeled data

## 1 Introduction

Supervised machine learning algorithms need a large number of labeled data to assign an unlabeled example to a class. As a consequence, this characteristic demands too much effort from a specialist, as the stage of labeling all the instances, is necessary. On the contrary, semi-supervised techniques are more automated, since their needs for labeled data are dramatically reduced and can be easily applied in a variety of fields, such as text mining, image or speech classification etc. [2].

Sun [11] reviews theories developed to understand the properties of multi-view learning and gives a taxonomy of approaches according to the supervised and semi-supervised machine learning mechanisms involved. In this work, a self-training method that combines the power of Logitboost and regression trees for semi-supervised tasks is proposed. We performed a comparison with other well-known semi-supervised classification methods on standard benchmark datasets and the presented technique had better accuracy in most cases.

## 2 Semi-supervised Techniques

The main idea of the self-training technique is the tuning of more than one classifiers, so as to achieve a better classification accuracy [1]. The whole procedure is split into


L. Iliadis et al. (Eds.): EANN 2015, CCIS 517, pp. 1–10, 2015.
DOI: 10.1007/978-3-319-23983-5_14

several phases. During the first one, a classifier of the user's choice is trained using the small set of labeled examples. After this phase has been completed, unlabeled examples are examined and classified according to the knowledge that our learner has acquired from the previous stage. When the second phase has been finished, all the instances that were previously unlabeled and for which the learner's prediction exceeds a trust-threshold, are added to the training set along with their predicted labels. Next steps include the re-training of the classifier, under the assumption that the new training set provides us more useful information, until all the stopping criteria are met.

Co-Training is based on the hypothesis that the attribute space can be split into two disjoint subsets and that each subset may contribute to correct classification [4]. According to this guess, a single learner is trained on each subset. To begin with, both learners are trained only on labeled data. The next phase comprises the classification of usually a small number of unlabeled examples by both previous learners. During this phase, the most confident predictions of each one learner are added to the training set of the other one. This procedure continues to be repeated for a number of times, until a stopping criteria to be satisfied. Didaci et al [16] evaluated co-training performance as a function of the size of the labeled training set. It seems that the concept of using co-training can work even with very few instances per class. However, Du et al [20] made a number of experiments and came to the conclusion that relying on small labeled training sets, verification of both the sufficiency and independence assumptions of splitting single view into two views are unreliable.

Sun and Jin [10] proposed robust co-training, in which the predictions of co-training on the unlabeled data are tested through Canonical correlation analysis (CCA) with the intention to enlarge the training set, adding only these instances whose predicted labels are consistent with the outcome of CCA. Wang et al [5] proposed to combine the probabilities of class membership with a distance metric between unlabeled instances and labeled instances. If two instances have the same class probability value, the one with the smallest distance will have larger chance to be selected. Xu et al [6] proposed DCPE co-training algorithm. For each classifier, if unlabeled instances have the same prediction labels and the highest class probability values differences between this classifier and the other one, then these unlabeled examples are added into the training set of the classifier. COTRADE [18] uses a number of predicted labels with higher confidence of either learner are passed to the other one, if a number of constraints are imposed to avoid introducing noise.

Li and Zhou [7] proposed Co-Forest algorithm. According to this algorithm, a number of Random Trees are trained on bootstrap sample data from the data set. Then each Random Tree is refined with a small number of unlabeled instances during the training process and the final prediction is produced by majority voting. Deng and Guo [12] proposed a new Co-Forest algorithm named ADE-Co-Forest which uses a data editing technique to identify and discard probably mislabeled instances during the iterations. Co-training by committee has been proposed by Hady and Schwenker [8]. In their work, an initial committee was built with the labeled data set. Three ensemble methods were used: Bagging, AdaBoost and Random Subspace and these semi-supervised learning algorithms were named as CoBag, CoAdaBoost and CoRSM, respectively. Liu and Yuen [22] also proposed a boosted co-training algorithm.

Tri-training algorithm has been proposed by Zhou and Li [3]. In each round of tri-training algorithm, an unlabeled instance is labeled for a learner if the other two learners agree on the labeling. Guo and Li [17] proposed improved tri-training algorithm (im-tri-training) that addresses some issues existed in tri-training such as unsuitable error estimation. Democratic co-learning [9] also uses multiple classifiers. Initially, each classifier is trained with the same data. The classifiers are then used to label the unlabeled data. Each instance is then labeled with the majority voting, and the labeled instance is added to the training set of the classifier whose prediction disagree with the majority. Sun and Zhang [19] proposed an ensemble of classifiers to be trained from each view, and the consensus prediction of the ensemble to be used to select confident labeled instances from the unlabeled data to teach the other ensemble from the other view.

## 3 Proposed Algorithm

Regression trees are obtained using a fast divide and conquer greedy algorithm that recursively partitions the given training set into smaller subsets. The most well-known regression tree inducer is the M5 [14]. In spite of their advantages regression trees are also known for their instability, since a small change in the training data can lead to a different choice when building a node, which in turn can produce a dramatic change in the regression tree, particularly if the change occurs in top level nodes. A well-known technique to improve the accuracy of tree-based classifiers is the boosting procedure. The idea of boosting is to combine the prediction of many simple classifiers to form a powerful 'committee'. The simple classifiers are trained on reweighted versions of the training data, such that training instances that have been misclassified by the classifiers built so far, receive a higher weight and the new classifier can concentrate on these hard instances.

Additive logistic regression algorithm: Logitboost [13] is based on the observation that boosting is in essence fitting an additive logistic regression model to the training data. An additive model is an approximation to a function F(x) of the form:

$$F(x) = \sum_{m=1}^{M} c_m f_m(x) \tag{1}$$

where the $c_m$ are constants to be determined and $f_m$ are basis functions. It must be mentioned that m is number of classifiers and is set equal to 10 in our implementation.

If we assume that F(x) is the mapping that we seek to fit as our strong aggregate hypothesis, and f(x) are our weak hypotheses, then it can be shown that the two-class boosting algorithm is fitting such a model by minimizing the criterion:

$$J(F) = E(e^{-yF(x)}) \tag{2}$$

where $y$ is the true class label in {-1,1}. Logitboost minimises this criterion by using Newton-like steps to fit an additive logistic regression model to directly optimise the binomial log-likelihood:

$$-\log(1+e^{-2yF(x)}) \tag{3}$$

Friedman et al [13] used Logitboost algorithm with one level decision tree as base learner for solving supervised classification problems. A very useful property of this classification method is that it directly yields class conditional probability estimates that are crucial for constructing self-train classifiers. Regression trees are known for their simplicity and efficiency when dealing with domains with large number of features and instances. In this work, we propose a self-training method that combines the power of Logitboost and regression trees for semi-supervised tasks. The proposed algorithm (Self-Logit-M5) is presented in Figure 1. The self-train process produces good results by using the more accurate class probabilities of Logitboost regression tree model for the unlabeled instances.

**Input**: An initial set of labeled instances L and a set of unlabeled instances U
**Initialization:**
- Train Logitboost using M5 as base model on L

**Loop for a number of iterations (40 in our implementation)**
- Use Logitboost classifier to select the most confident predictions, remove them from U and add them to L. In each iteration about 2 instances per class are removed from U and added to L.

**Output**: Use Logitboost trained on L to predict class labels of the test cases.

**Fig. 1.** The Self-Logit-M5 Algorithm

For the implementation, it must be mentioned that we made use of the free available code of Weka [24] and KEEL [25].

## 4 Experiments

The experiments are based on standard classification datasets taken from the KEEL-dataset repository [23]. These data sets have been partitioned using the 10-fold cross-validation procedure. For each generated fold, a given algorithm is trained with the examples contained in the rest of folds (training partition) and then tested with the current fold. Each training partition is divided into two parts: labeled and unlabeled examples. In order to study the influence of the amount of labeled data, we take two different ratios when dividing the training set: 10% and 20%.

For the experiments, the proposed method has been compared with other state of the art algorithms integrated into the KEEL tool [23] such as Self-Training (C45) [1], Self-Training (SMO) [27], Co-Bagging (C45) [8], TriTraining (C45) [3], Democratic-Co [9], CoForest [7] and Co-Training (C45) [4]. It must be mentioned that the default parameters of KEEL were used for all the tested algorithms. The classification

accuracy of each tested algorithm using 10% and 20% as labeled ratio is presented in Table 1 and Table 2 respectively. As it concerns the values in bold style, they actually point the best accuracy value in each row among the different algorithms.

**Table 1.** Classification accuracy (Labeled Ratio 10%)

| | **Algorithms**[1] | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Datasets** | Alg1 | Alg2 | Alg3 | Alg4 | Alg5 | Alg6 | Alg7 | Alg8 |
| appendicitis | **0.862** | 0.832 | 0.754 | 0.805 | 0.805 | 0.822 | 0.823 | 0.832 |
| australian | 0.828 | 0.828 | 0.796 | 0.828 | **0.845** | **0.845** | 0.841 | 0.835 |
| banana | 0.877 | 0.848 | **0.896** | 0.855 | 0.848 | 0.842 | 0.527 | 0.848 |
| breast | 0.672 | 0.722 | 0.691 | 0.725 | 0.722 | 0.729 | **0.734** | 0.677 |
| bupa | 0.606 | 0.539 | **0.633** | 0.612 | 0.574 | 0.510 | 0.585 | 0.574 |
| chess | 0.956 | 0.954 | 0.896 | 0.954 | **0.958** | 0.920 | 0.944 | 0.952 |
| coil2000 | 0.922 | 0.937 | 0.572 | 0.935 | 0.936 | 0.932 | 0.930 | **0.940** |
| contraceptive | **0.500** | 0.489 | 0.450 | 0.483 | 0.481 | 0.436 | 0.485 | 0.446 |
| crx | 0.801 | **0.866** | 0.832 | 0.850 | 0.856 | 0.850 | 0.821 | 0.816 |
| dermatology | 0.863 | 0.856 | 0.840 | 0.876 | 0.882 | 0.876 | **0.905** | 0.843 |
| ecoli | 0.641 | 0.647 | 0.622 | 0.656 | **0.659** | 0.637 | 0.628 | 0.577 |
| flare | **0.721** | **0.721** | 0.516 | 0.714 | 0.716 | **0.721** | 0.402 | 0.574 |
| german | 0.713 | 0.706 | 0.592 | 0.711 | **0.717** | 0.716 | 0.686 | 0.690 |
| glass | 0.553 | 0.485 | 0.496 | 0.490 | 0.492 | 0.487 | **0.559** | 0.450 |
| haberman | 0.650 | 0.705 | 0.601 | 0.712 | 0.709 | 0.716 | 0.601 | **0.719** |
| heart | 0.763 | 0.678 | 0.770 | 0.704 | 0.715 | **0.800** | 0.693 | 0.700 |
| hepatitis | 0.793 | **0.834** | **0.834** | **0.834** | **0.834** | **0.834** | 0.811 | **0.834** |
| housevotes | 0.921 | **0.941** | 0.876 | 0.920 | 0.916 | 0.890 | 0.922 | 0.823 |
| iris | 0.900 | 0.840 | **0.940** | 0.800 | 0.727 | 0.913 | 0.933 | 0.847 |
| led7digit | **0.686** | 0.614 | 0.568 | 0.564 | 0.604 | 0.616 | 0.634 | 0.514 |
| lymphography | **0.737** | 0.631 | 0.549 | 0.595 | 0.612 | 0.490 | 0.646 | 0.573 |
| magic | **0.849** | 0.822 | 0.839 | 0.832 | 0.825 | 0.784 | 0.844 | 0.820 |
| mammographic | 0.809 | 0.803 | 0.782 | 0.809 | **0.818** | 0.796 | 0.794 | 0.807 |
| monk-2 | 0.953 | **0.973** | 0.783 | 0.966 | 0.966 | 0.908 | 0.939 | **0.973** |
| mushroom | 0.996 | **0.997** | 0.992 | 0.995 | 0.996 | 0.993 | 0.908 | **0.997** |
| nursery | **0.962** | 0.906 | 0.815 | 0.901 | 0.904 | 0.895 | 0.381 | 0.903 |
| page-blocks | 0.953 | 0.952 | 0.940 | 0.957 | 0.956 | 0.908 | **0.959** | 0.949 |
| penbased | 0.965 | 0.892 | **0.976** | 0.905 | 0.903 | 0.947 | 0.955 | 0.896 |
| phoneme | **0.829** | 0.777 | **0.829** | 0.789 | 0.777 | 0.787 | 0.801 | 0.765 |
| pima | 0.680 | 0.664 | 0.608 | 0.634 | 0.656 | **0.697** | 0.663 | 0.670 |
| ring | 0.904 | 0.840 | **0.970** | 0.858 | 0.854 | 0.874 | 0.882 | 0.837 |
| saheart | 0.645 | 0.652 | 0.613 | 0.650 | 0.678 | **0.682** | 0.656 | 0.636 |
| satimage | **0.870** | 0.805 | 0.825 | 0.821 | 0.822 | 0.846 | 0.860 | 0.806 |
| segment | **0.931** | 0.890 | 0.907 | 0.916 | 0.900 | 0.903 | 0.903 | 0.902 |
| sonar | 0.686 | 0.643 | 0.669 | 0.701 | 0.702 | 0.601 | **0.755** | 0.582 |
| spambase | 0.917 | 0.867 | 0.851 | 0.895 | 0.881 | 0.878 | **0.919** | 0.888 |
| spectfheart | 0.749 | 0.682 | 0.656 | 0.757 | 0.757 | 0.738 | **0.775** | 0.724 |
| splice | **0.938** | 0.827 | 0.541 | 0.825 | 0.825 | 0.898 | 0.507 | 0.831 |
| texture | 0.929 | 0.831 | **0.963** | 0.850 | 0.852 | 0.894 | 0.907 | 0.829 |
| thyroid | 0.987 | **0.992** | 0.932 | 0.991 | 0.992 | 0.939 | 0.986 | **0.992** |
| tic-tac-toe | **0.741** | 0.711 | 0.627 | 0.704 | 0.709 | 0.690 | 0.597 | 0.693 |
| titanic | 0.780 | 0.775 | 0.776 | **0.784** | 0.777 | 0.776 | 0.707 | 0.778 |
| twonorm | 0.949 | 0.814 | **0.973** | 0.860 | 0.862 | 0.965 | 0.899 | 0.809 |
| vehicle | **0.633** | 0.579 | 0.586 | 0.603 | 0.619 | 0.502 | 0.612 | 0.575 |
| vowel | 0.495 | 0.424 | 0.463 | 0.456 | 0.453 | 0.416 | **0.522** | 0.438 |
| wine | 0.854 | 0.741 | **0.955** | 0.787 | 0.820 | 0.949 | 0.859 | 0.808 |
| wisconsin | 0.945 | 0.909 | 0.952 | 0.928 | 0.931 | **0.965** | 0.936 | 0.906 |
| yeast | **0.506** | 0.462 | 0.458 | 0.477 | 0.491 | 0.489 | 0.456 | 0.489 |
| zoo | 0.841 | 0.679 | 0.493 | 0.746 | 0.719 | **0.931** | 0.909 | 0.636 |

[1] Alg1: Self-training (Logitboost), Alg2: Self-Training (C45), Alg3: Self-Training (SMO), Alg4: Co-Bagging (C45), Alg5: TriTraining (C45), Alg6: Democratic-Co, Alg7: CoForest, Alg8: Co-Training (C45)

**Table 2.** Classification accuracy (Labeled Ratio 20%)

| | Algorithms[2] | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Datasets** | Alg1 | Alg2 | Alg3 | Alg4 | Alg5 | Alg6 | Alg7 | Alg8 |
| appendicitis | **0.887** | 0.851 | 0.705 | 0.832 | 0.832 | 0.860 | 0.877 | 0.850 |
| australian | **0.859** | 0.858 | 0.830 | 0.836 | 0.852 | 0.858 | 0.842 | 0.838 |
| banana | 0.886 | 0.880 | **0.899** | 0.880 | 0.874 | 0.879 | 0.541 | 0.874 |
| breast | 0.687 | 0.716 | 0.636 | 0.722 | 0.723 | 0.718 | **0.727** | 0.723 |
| bupa | **0.641** | 0.606 | 0.611 | 0.603 | 0.620 | 0.550 | 0.602 | 0.606 |
| chess | **0.978** | **0.978** | 0.948 | **0.978** | **0.978** | 0.939 | 0.951 | 0.976 |
| coil2000 | 0.923 | **0.940** | 0.817 | 0.934 | 0.937 | 0.933 | 0.927 | **0.940** |
| contraceptive | **0.515** | 0.478 | 0.471 | 0.481 | 0.488 | 0.484 | 0.494 | 0.504 |
| crx | 0.833 | 0.857 | **0.860** | 0.851 | 0.854 | 0.859 | 0.841 | 0.850 |
| dermatology | **0.924** | 0.910 | 0.910 | 0.919 | 0.921 | 0.930 | **0.924** | 0.879 |
| ecoli | **0.774** | 0.720 | 0.670 | 0.732 | 0.721 | 0.724 | 0.748 | 0.733 |
| flare | 0.720 | **0.728** | 0.585 | 0.718 | 0.727 | 0.740 | 0.416 | 0.690 |
| german | 0.714 | 0.697 | 0.618 | 0.712 | 0.684 | **0.729** | 0.684 | 0.699 |
| glass | 0.628 | 0.509 | 0.551 | 0.563 | 0.610 | 0.480 | **0.634** | 0.535 |
| haberman | 0.686 | 0.709 | 0.660 | **0.735** | 0.709 | 0.722 | 0.640 | **0.735** |
| heart | 0.789 | 0.752 | 0.789 | 0.748 | 0.793 | **0.822** | 0.730 | 0.756 |
| hepatitis | 0.747 | 0.843 | 0.834 | 0.820 | 0.843 | 0.834 | **0.861** | 0.843 |
| housevotes | 0.931 | **0.959** | 0.888 | 0.953 | **0.959** | 0.902 | 0.945 | 0.936 |
| iris | 0.893 | 0.893 | 0.913 | 0.867 | 0.880 | **0.953** | 0.940 | 0.893 |
| led7digit | **0.692** | 0.678 | 0.644 | 0.658 | 0.676 | 0.662 | 0.678 | 0.644 |
| lymphography | **0.817** | 0.706 | 0.649 | 0.747 | 0.755 | 0.461 | 0.693 | 0.775 |
| magic | **0.855** | 0.830 | 0.842 | 0.841 | 0.834 | 0.802 | 0.852 | 0.832 |
| mammographic | 0.823 | 0.823 | 0.793 | **0.825** | 0.816 | 0.808 | 0.787 | 0.823 |
| monk-2 | 0.977 | **0.980** | 0.869 | 0.973 | 0.973 | 0.944 | 0.979 | **0.980** |
| mushroom | **1.000** | 0.999 | 0.997 | 0.999 | 0.999 | 0.998 | 0.910 | 0.999 |
| nursery | **0.984** | 0.924 | 0.576 | 0.924 | 0.926 | 0.911 | 0.384 | 0.926 |
| page-blocks | **0.965** | 0.960 | 0.957 | 0.961 | 0.961 | 0.912 | 0.959 | 0.961 |
| penbased | 0.979 | 0.924 | **0.983** | 0.937 | 0.929 | 0.963 | 0.966 | 0.925 |
| phoneme | **0.851** | 0.784 | 0.848 | 0.824 | 0.798 | 0.806 | 0.831 | 0.802 |
| pima | **0.734** | 0.681 | 0.646 | 0.708 | 0.694 | 0.732 | 0.711 | 0.687 |
| ring | 0.928 | 0.866 | **0.971** | 0.892 | 0.880 | 0.897 | 0.893 | 0.863 |
| saheart | 0.667 | 0.652 | 0.602 | 0.686 | 0.678 | 0.697 | 0.671 | **0.704** |
| satimage | **0.886** | 0.824 | 0.857 | 0.841 | 0.835 | 0.861 | 0.872 | 0.826 |
| segment | **0.953** | 0.926 | 0.945 | 0.928 | 0.925 | 0.931 | 0.934 | 0.929 |
| sonar | 0.725 | 0.664 | 0.691 | 0.659 | 0.663 | 0.647 | **0.745** | 0.634 |
| spambase | 0.926 | 0.891 | 0.900 | 0.896 | 0.894 | 0.894 | **0.931** | 0.891 |
| spectfheart | **0.802** | 0.719 | 0.671 | 0.783 | 0.749 | 0.731 | 0.791 | 0.761 |
| splice | **0.954** | 0.883 | 0.578 | 0.888 | 0.884 | 0.912 | 0.521 | 0.879 |
| texture | 0.960 | 0.867 | **0.982** | 0.884 | 0.893 | 0.918 | 0.935 | 0.863 |
| thyroid | 0.992 | **0.994** | 0.939 | **0.994** | **0.994** | 0.942 | 0.988 | 0.993 |
| tic-tac-toe | **0.779** | 0.757 | 0.672 | 0.729 | 0.751 | 0.733 | 0.607 | 0.721 |
| titanic | 0.777 | 0.782 | 0.781 | **0.783** | 0.782 | 0.780 | 0.723 | 0.782 |
| twonorm | 0.955 | 0.817 | **0.973** | 0.874 | 0.867 | 0.971 | 0.903 | 0.828 |
| vehicle | **0.679** | 0.649 | 0.675 | 0.655 | 0.662 | 0.483 | 0.655 | 0.649 |
| vowel | 0.644 | 0.530 | **0.705** | 0.557 | 0.555 | 0.503 | 0.640 | 0.530 |
| wine | 0.848 | 0.837 | **0.961** | 0.826 | 0.842 | 0.954 | 0.842 | 0.786 |
| wisconsin | 0.951 | 0.934 | 0.952 | 0.936 | 0.925 | **0.964** | 0.936 | 0.934 |
| yeast | **0.567** | 0.526 | 0.502 | 0.520 | 0.549 | 0.549 | 0.489 | 0.549 |
| zoo | 0.891 | 0.831 | 0.601 | 0.823 | 0.826 | 0.890 | **0.897** | 0.734 |

In the sequel, in Table 3 and Table 4 the results of Friedman test [13] together with a statistical test [13] are presented, which are used in order to conduct comparisons among all algorithms considered in the study and the proposed algorithm for both situations that have already been examined.

[2] Alg1: Self-training (Logitboost), Alg2: Self-Training (C45), Alg3: Self-Training (SMO), Alg4: Co-Bagging (C45), Alg5: TriTraining (C45), Alg6: Democratic-Co, Alg7: CoForest, Alg8: Co-Training (C45)

**Table 3.** Average rankings of the algorithms in 10% labeled ratio (Friedman) and Holm / Hochberg (alpha=0.05)

| Algorithm | Friedman Ranking | p | Holm/Hochberg Test |
|---|---|---|---|
| Self-training (Logitboost) | 3.0408163265306114 | - | - |
| TriTraining (C45) | 3.9693877551020407 | 0.060602 | 0.05 |
| CoForest | 4.265306122448979 | 0.013347 | 0.025 |
| Co-Bagging (C45) | 4.316326530612245 | 0.009953 | 0.01666 |
| Democratic-Co | 4.591836734693876 | 0.001723 | 0.0125 |
| Self-Training (C45) | 4.979591836734695 | 8.94E-5 | 0.01 |
| Self-Training (SMO) | 5.336734693877552 | 3.49E-6 | 0.00833 |
| Co-Training (C45) | 5.500000000000001 | 6.72E-7 | 0.00714 |

**Table 4.** Average rankings of the algorithms in 20% labeled ratio (Friedman) and Holm / Hochberg (alpha=0.05)

| Algorithm | Friedman Ranking | p | Holm/Hochberg Test |
|---|---|---|---|
| Self-training (Logitboost) | 2.724489795918366 | - | - |
| TriTraining (C45) | 4.469387755102042 | 4.22E-4 | 0.05 |
| CoForest | 4.510204081632654 | 3.08E-4 | 0.025 |
| Co-Bagging (C45) | 4.5306122448979576 | 2.63E-4 | 0.01666 |
| Democratic-Co | 4.530612244897959 | 2.63E-4 | 0.0125 |
| Co-Training (C45) | 4.969387755102043 | 5.72E-6 | 0.01 |
| Self-Training (C45) | 5.030612244897958 | 3.16E-6 | 0.00833 |
| Self-Training (SMO) | 5.2346938775510194 | 3.93E-7 | 0.00714 |

As a result, the proposed algorithm gives statistical better results than all the tested algorithms. Better probability-based ranking and high classification accuracy could select the high-confidence predictions in the selection step of self-training and therefore, the proposed method improved the performance of self-training. The proposed approach performs better than the tested state of the art algorithms in the tested datasets.

## 5 Tool Presentation

In this section a short presentation of a tool that implements the semi-supervised proposed algorithm is provided. First of all, its display is illustrated in Figure 2. Secondly, it is easily observable that it is simple enough, as it contains only three buttons through which the user can interact with. Moreover, because it is developed using JAVA, it is platform independent. The only restriction is the hardware requirement of 64-bit CPU combined with 4GB RAM.

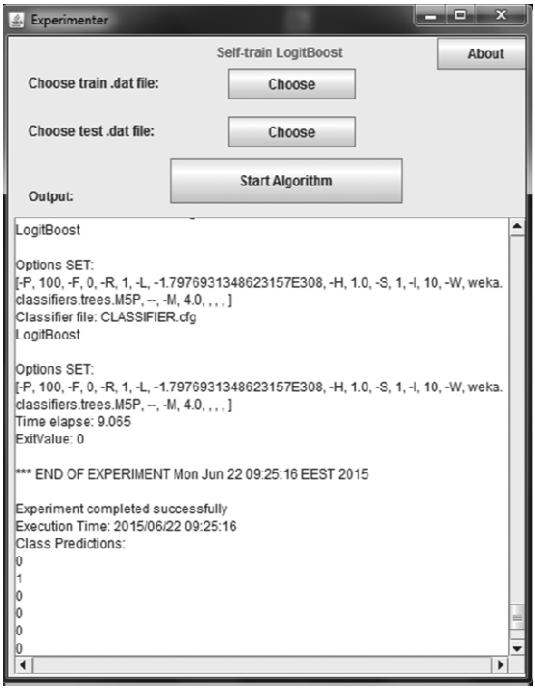


**Fig. 2.** Illustration of the corresponding tool for Self-train LogitBoost Algorithm.

To begin with the description, anyone who wants to run the proposed algorithm has to follow the standard steps that semi-supervised theory includes. Particularly, a train set has to be chosen through the first button Choose. It must be mentioned that the train set has to consist of both labeled and unlabeled instances. The provided sets contain 10% labeled instances, but the user can adjust the percentage of unlabeled instances over his train set. After the completion of this step, the user can choose the test set. The difference in comparison with the train set is that the column which refers to the class of each instance simply does not exist. After that, selecting the Start Algorithm button, a file named as predictions.txt will be created in the same folder with the Experimenter program, which finally will contain the class prediction of each tested instance.

As it considers the data sets that are used by this tool, are managed by plain ASCII text files, with the .dat extension. Each data file is composed by header (basic metadata describing the data set) and data (content of the dataset). The corresponding link for downloading this tool and some basics instructions is the following: http://www.math.upatras.gr/~sotos/SelfLogitBoost-Experiment.zip.

## 6 Conclusion

It is promising to develop techniques that use both labeled and unlabeled instances in classification tasks. The limited availability of labeled examples makes the learning

process difficult, as supervised learning methods cannot produce a classifier with good generalization performance.

The main difficulty in self-training task is to find a set of high confidence predictions of unlabeled instances. Although for a lot of domains, decision tree classifiers produce accurate classifiers, they provide poor probability estimates [28-29]. The reason is that the sample size at the leaves is small in most of the time, and all instances at a leaf get the same probability. In addition, the probability estimate is the proportion of the majority class at the leaf of a pruned decision tree.

In this work, a self-train Logitboost algorithm is presented. The self-train process improves the results by using the accurate class probabilities for which the Logitboost regression tree model is more confident at the unlabeled instances. We performed a comparison with other well-known semi-supervised classification methods on standard benchmark datasets and the presented technique had better accuracy in most cases. In spite of these results, no general method will work always.

In the near future, we will try to improve the results of proposed method combining the probabilities of class membership with a distance metric between unlabeled instances and labeled instances [21].

## References


1. Rosenberg, C., Hebert, M., Schneiderman, H.: Semi-supervised self-training of object detection models. In: 7th IEEE Workshop on Applications of Computer Vision, pp. 29–36 (2005)
2. Friedhelm, S., Edmondo, T.: Pattern classification and clustering: A review of partially supervised learning approaches. Pattern Recognition Letters **37**, 4–14 (2014)
3. Zhou, Z.-H., Li, M.: Tri-Training: Exploiting Unlabeled Data Using Three Classifiers. IEEE Trans. on Knowledge and Data Engg. **17**(11), 1529–1541 (2005)
4. Chapelle, O., Schölkopf, B., Zien, A.: Semi-supervised learning. MIT Press, Cambridge (2006)
5. Wang, S., Wu, L., Jiao, L., Liu, H.: Improve the performance of co-training by committee with refinement of class probability estimations. Neurocomputing **136**, 30–40 (2014)
6. Xu, J., He, H., Man, H.: DCPE co-training for classification. Neurocomputing **86**, 75–85 (2012)
7. Li, M., Zhou, Z.: Improve computer-aided diagnosis with machine learning techniques using undiagnosed samples. IEEE Trans. Syst. Man Cybernet, 1088–1098 (2007)
8. Hady, M., Schwenker, F.: Co-training by committee: a new semi-supervised learning framework. In: Proceedings of the IEEE International Conference on Data Mining Workshops, pp. 563–572 (2008)
9. Zhou, Y., Goldman, S.: Democratic co-learning. In: Ictai, 16th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2004), pp. 594–202 (2004)
10. Sun, S., Jin, F.: Robust co-training. Int. J. Pattern Recognit. Artif. Intell. **25**, 1113–1126 (2011)
11. Sun, S.: A survey of multi-view machine learning. Neural Computing and Applications **23**(7–8), 2031–2038 (2013)
12. Deng, C., Guo, M.Z.: A new co-training-style random forest for computer aided diagnosis. Journal of Intelligent Information Systems **36**, 253–281 (2011)

13. Friedman, J., Hastie, T., Tibshirani, R.: Additive logistic regression: a statistical view of boosting. Ann. Statist. **28**(2), 337–407 (2000)
14. Torgo, L.: Inductive learning of tree-based regression models. AI Communications **13**(2), 137–138 (2000)
15. Jiang, Z., Zhang, S., Zeng, J.: A hybrid generative/discriminative method for semi-supervised classification. Knowledge-Based Systems **37**, 137–145 (2013)
16. Didaci, L., Fumera, G., Roli, F.: Analysis of co-training algorithm with very small training sets. In: Gimel'farb, G., Hancock, E., Imiya, A., Kuijper, A., Kudo, M., Omachi, S., Windeatt, T., Yamada, K. (eds.) SSPR&SPR 2012. LNCS, vol. 7626, pp. 719–726. Springer, Heidelberg (2012)
17. Guo, T., Li, G.: Improved tri-training with unlabeled data. In: Wu, Y. (ed.) Software Engineering and Knowledge Engineering: Vol. 2. AISC, vol. 115, pp. 139–148. Springer, Heidelberg (2012)
18. Zhang, M.-L., Zhou, Z.-H.: CoTrade: Confident co-training with data editing. IEEE Trans. Syst. Man Cybernet, Part B: Cybernetics **41**(6), 1612–1626 (2011)
19. Sun, S., Zhang, Q.: Multiple-View Multiple-Learner Semi-Supervised Learning. Neural Process. Lett. **34**, 229–240 (2011)
20. Du, J., Ling, C.X., Zhou, Z.-H.: When. does cotraining work in real data? IEEE Trans. on Knowledge and Data Engg. **23**(5), 788–799 (2011)
21. Zhu, X., Goldberg, A.: Introduction to semi-supervised learning. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool (2009)
22. Liu, C., Yuen, P.C.: A boosted co-training algorithm for human action recognition. IEEE Trans. on Circuits and Systems for Video Technology **21**(9), 1203–1213 (2011). 5739520
23. Alcalá-Fdez, J., Fernandez, A., Luengo, J., Derrac, J., García, S., Sánchez, L., Herrera, F.: KEEL Data-Mining Software Tool: Data Set Repository, Integration of Algorithms and Experimental Analysis Framework. Journal of Multiple-Valued Logic and Soft Computing **17**(2–3), 255–287 (2011)
24. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.: The WEKA Data Mining Software: An Update. SIGKDD Explorations **11**(1) (2009)
25. Triguero, I., Garca, S., Herrera, F.: Self-labeled techniques for semi-supervised learning: taxonomy, software and empirical study. Knowledge and Information Systems **42**(2), 245–284 (2015)
26. García, S., Fernández, A., Luengo, J., Herrera, F.: Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. Inf. Sciences **180**(10), 2044–2064 (2010)
27. Keerthi, S.S., Shevade, S.K., Bhattacharyya, C., Murthy, K.R.K.: Improvements to Platt's SMO Algorithm for SVM Classifier Design. Neural Computation **13**(3), 637–649 (2001)
28. Mease, D., Wyner, A.J., Buja, A.: Boosted classification trees and class probability/quantile estimation. J. Mach. Learn. Res. **8**, 409–439 (2007)
29. Provost, F.J., Domingos, P.: Tree induction for probability based ranking. Mach. Learn. **52**, 199–215 (2003)