Journal of Computers Vol. 30 No. 6, 2019, pp. 110-122
doi:10.3966/199115992019123006009

Impr-Co-Forest: The Improved Co-forest Algorithm Based on Optimized Decision Tree and Dual-confidence Estimation Method

Fei-Fei Hou[1], Wen-Tai Lei[1*], Hong Li[1], Jing-Wei Ren[1], Geng-Ye Liu[2], Qing-Yuan Gu[2]

[1] School of Computer Science and Engineering, Central South University, Shaoshan South Road 410075, Changsha, China
{154611044, leiwentai, lihongcsu, 174612183}@csu.edu.cn

[2] Time Varying Transmission Co., LTD
{glenn.liu, qy.gu}@timevary.com

Received 16 February 2018; Revised 21 June 2018; Accepted 1 August 2018

Abstract. Co-forest is a classical semi-supervised collaborative training algorithm. Aming at solving the unavoidable mislabeling problem and the defect of decision tree classifier with low performance. In this paper, a novel Improved Co-forest (Impr-Co-Forest) algorithm is proposed to address the above issues. First, considering the creation and selection of optimized decision trees, in training process, we applied a weight scheme into adaboost method in order to pay more attention to the samples that are difficult to classify. Because newly labeled data are not always valid, the Out-Of-Bag-Error (OOBE) of single decision tree is compared with the decision trees with smaller OOBE. Then, to solve the mislabeling problem throughout the co-labeling iterations, a dual-confidence estimation method is proposed for the newly labeled data that is effectively chosen to update classifiers. Finally, the weighted vote, which replaces the simple majority vote, achieves the estimation of probability for each class. Experimental results on eight UCI datasets and Pascal VOC show that the proposed Impr-Co-Forest algorithm has better classification performance than both supervised and semi-supervised algorithms.

Keywords: dual-confidence estimation, Improved Co-Forest (Impr-Co-Forest), optimized decision tree, semi-supervised collaborative training, weighted vote

1 Introduction

It is well-known that collecting tremendous high-quality labeled data is expensive, yet we could easily collect abundant unlabeled data in many real applications. Hence, the semi-supervised learning algorithms which make full use of large amount of unlabeled data to improve the performance of classifiers have become a hot topic [1]. Many methods have been proposed to tackle semi-supervised learning, we only introduce the most related ones. For more information of semi-supervised learning, see [2-4]. Disagreement-based semi-supervised learning started from the seminal paper of Blum and Mitchell [5] on co-training. Co-training first learns two classifiers from two views and then lets them label unlabeled data for each other to improve performance. For purpose of obtaining better generalization, Li and Zhou proposed Co-forest algorithm using N random decision tree classifiers, where N is always greater than 3 [6]. In order to improve the performance of co-training-style algorithms, the proposed methods have focused on solving the unavoidable mislabeling problem. Li and Zhou proposed a method using cutting edge weights-based neighborhood graph to identify and remove mislabeled data in the self-training process [7]. In [8], Deng proposed a new algorithm named ADE-Co-forest, an effective adaptive strategy is proposed to trigger the editing mechanism according to different situations. Co-training has been combined with deep model for the tasks which have two views [9-10]. Among them, Co-training

* Corresponding Author

paradigm has gained considerable attention with the advantage of fast convergence. Nevertheless, the weights of training samples in the co-forest algorithm are updated at the same time, which cannot highlight the misclassified samples during training process.

A natural idea is to combine adaboost with co-forest learning. Boosting is an effective tool to improve the learning ability of basic learning algorithms. It is regarded as the most common and effective method in ensemble learning. The first applicable approach of boosting is Adaboost proposed by Freund and Schapire [11], which has been rated as one of the top ten algorithms in data mining [12]. The basic Adaboost classification algorithm is implemented for binary classification problems, the main idea of Adaboost is as follows: A distribution D on training space is generated in which each training sample is assigned a weight of 1/m initially, where m is the number of training samples. Adaboost works by sequentially applying a weak classifier to train the reweighed training dataset (generated by the distribution D) and taking a majority vote to integrate all the weak hypotheses generated by weak classifiers into the final hypothesis. In each iteration, the sample distribution D is updated according to the hypothesis generated in this iteration. The rule of updating is, samples that failed to be assigned to the right class gain higher weights, so that in the next iteration the classifier will focus more on learning those failed classified samples.

In this paper, aming at solving the unavoidable mislabeling problem and the defect of decision tree classifier with low performance, we propose the Impr-Co-Forest algorithm that bases on the optimated decision tree and the dual-confidence estimation. First, the weight scheme of adaboost is added into the training process to optimize the creation of the decision trees, simutaneously, at each iteration, the decision tree with smaller error rate is choosen by comparing Out-Of-Bag-Error (OOBE) from each decision tree, and all the decision tree achieves selectively updated. Then, the dual-confidence estimation method is proposed to solve the inevitable mislabeling problem. Later, the weight class probability estimation method is exploited to predict the final class label. Experiments on eight UCI datasets and Pascal VOC dataset show that the proposed Impr-Co-Forest can more effectively and stably improve the classification accuracy compared with S4VM, Co-training, Random Forestmethods, et al.

This paper is organized as follows: Section 2 optimizes the creation and selection of decision tree, Section 3 presents dual-confidence estimation, in the following Section 4 introduces the weighted probability estimation method and summarizes the overall framework of Impr-Co-Forest algorithm. Section 5 shows the experimental results on two datasets. Finally, Section 6 concludes this paper and proposes some suggestions for future work.

2 Optimized Decision Tree

2.1 Optimizing the Creation of Decision Trees

Adaboost is an excellent ensemble algorithm, which has been widely used in practical applications due to its pretty classification capability and high precision performance [13-14]. The main principle of adaboost algorithm is that different weak classifiers are trained for the same training sets, then these weak classifiers are put together to constitute a strong classifier.

Co-forest is composed of multiple decision trees and each tree is an independent classifier. Therefore, in order to improve the classification accuracy of each decision tree, we introduce boosting scheme to the training process. Each tree in forest is a weak classifier, the classification performance will be improved as the introduction of the weight scheme of adaboost. The creation of the optimized decision tree is described in Table 1.

Table 1. The creation of the optimized decision tree

Algorithm 1: Optimizing the Creation of Decision Trees

Input: Random forest R, the decision tree $R(i)$, $i=1,2,\ldots,N$, N represents the number of decision tree in the forest. The number of samples in each decision tree is m, $(x_1,y_1),(x_2,y_2),\ldots,(x_m,y_m)$ denotes the samples, where $x_i \in X$, $y_i \in Y$, and $Y=\{-1,+1\}$ is the class label of each sample. Labeled samples L, the maximum number of cycles S, the sample weight D_k.

Output: The optimized decision trees.

for $R(i) \in R$, $i=1,2,\ldots,N$ do

Resampled tagged samples L to obtain N initial training sets. Set the sample weight in each training set to $1/m$, where m is the number of samples in the training set.

for $j=1:N$

Obtain N weak classifiers $h_{1,2,\ldots,N}$ based on different training sets.

for $k=1:S$

① Train each weak classifier h_k to get the error rate ε_k:

$$\varepsilon_k = \frac{1}{m}\left[\sum_{l=1}^{m} D_k(l)\,\delta\left(h_k(x_l) \neq y_l\right)\right]$$

if $\varepsilon_k > 0.5$, then reset the weight of each sample to 1/m, generate a training set by random sampling, train the classifier, and calculate the error rate ε_k.

② Use ε_k to calculate the weight of weak classifier h_k:

$$\alpha_k = \frac{1}{2}\ln\left(\frac{1-\varepsilon_k}{\varepsilon_k}\right)$$

③ For each sample, update its weight:

$$D_{k+1}(i) = \frac{D_k(i)}{Z_k} \times \begin{cases} e^{-\alpha_k} & if \quad h_k(x_i) = y_i \\ e^{\alpha_k} & if \quad h_k(x_i) \neq y_i \end{cases}$$

where, Z_k is a formal factor used to ensure $\sum_{l=1}^{m} D_{k+1}(i) = 1$.

end

The strong classifier $H(x) = sign\left(\sum_{k=1}^{S} \alpha_k h_k(x)\right)$ is obtained by integrating the weak classifier decision trees.

end

end

2.2 Optimizing the Selection of Decision Tree

In the iterative process of the Co-Forest algorithm, the newly marked samples do not always serve to improve the performance of the classifier. In some cases, the error rate of the classifier obtained after retraining is higher than that before retraining. In the bootstrap aggregation, the elements which are not chosen in training sets are called OOB elements. So, we use the OOBE which is computed by OOB elements to evaluate performance of single decision tree at each iteration. The OOBE is a good estimate of the generalization error. This method is used to evaluate the performance of the Co-Forest algorithm and avoid cross-checking with the test set. Leistner et al. [15] proposed using OOBE to measure the performance of random forests, by comparing the OOBE of all single trees in end of the $(i-1)-th$ iteration with that in end of the *i-th* iteration, they decide whether to remain or discard the tree in forest. Leistner's method has certain flaws, in that some affected trees will be discarded.

Inspired by the method proposed by Leistner et al. [15], in the Co-Forest training process, we separately compare the OOBE of each decision tree. After each iteration is completed, the OOBE of each

tree in the forest is calculated and compared with the previous one, and the decision trees with smaller OOBEs will be retained. In other words, for each decision tree, if the OOBE obtained at the end of the $i-th$ training is greater than that at the end of the $(i-1)-th$ training, then we keep the decision tree obtained at the end of the $(i-1)-th$ iteration without updating the classifier. If the OOBE obtained at the end of the $i-th$ training is less than that at the end of the $(i-1)-th$ training, the classifier is updated and the latest decision tree is retained in the classification model. In this paper, the tree with smaller OOBE is retained and this method can avoid the error accumulation, the decision trees with the highest performance can be chosen to compose forest. The learning process of the proposed optimize the selection of decision tree method is shown in Table 2.

Table 2. The selection of the optimized decision tree

Algorithm 2: optimizing the selection of decision tree

Input: Labeled training data X_l, the number of the decision forests *N*, the training times *T*

Output: The learned F

for j=1:N

For the j^{th} decision tree, generate a initial labeled set X_l^j using the bootstrap aggregation

end for

for i=1:T

Initialize an empty forest F.

for j=1:N

Retraining the tree with new training set X_i^j, $h_i^j = \mathrm{Re}\,trainTree(X_i^j)$

Compute the OOBE, $e_i^j = oobe(h_i^j, X_l - X_l^j)$

Retraining the tree with new training set X_{i-1}^j, $h_{i-1}^j = \mathrm{Re}\,trainTree(X_{i-1}^j)$

Compute the OOBE, $e_{i-1}^j = oobe(h_{i-1}^j, X_l - X_l^j)$

if $e_i^j > e_{i-1}^j$ then

$F = F \cup h_{i-1}^j$

else

$F = F \cup h_i^j$

end if

end for

end for

Return F.

3 Dual-Confidence Estimation Method

In Co-Forest, let *N* denotes the size of the forest, *L* denotes the original labeled dataset, *U* denotes the original unlabeled dataset. First, the *N* labeled dataset $L_i\,(i=1,2,\ldots,N)$ will be obtained through bootstap sampling from the initially labeled dataset. The initial classifiers $h_i\,(i=1,2,\ldots,N)$ are trained on them. Then, the initial classifiers would be updated along with the co-training iterations. Let the H_i denotes the entire classifiers except h_i. The $h_i\,(i=1,2,\ldots,N)$ is retaining by the new labeled instance set $L_i \cup L_i^{'}$, $L_i^{'}$ is labeled by the concomitant ensemble $H_i\,(i=1,2,\ldots,N)$. In the co-labeling process, the unlabeled samples may be labeled erroneously, much noise are introduced in the newly labeled set. Thus, it will degrade the performance of the classifier.

In order to raise the selection criteria of unlabeled samples and reduce the introduction of noise to unlabeled data, the dual-confidence estimation method is proposed in this paper. In the iterations, first, we use the K-Nearest-Neighbor with Attibute and Distance Weighted (ADWKNN) method to predict the class label of the sample in the newly labeled datasets implicitly. If the class labels are predicted conformably by this method and corresponding ensemble $H_i\,(i=1,2,\ldots,N)$, the samples and its

corresponding labels will be put into buffer pool. Then, the difference value maximization confidence estimation method based on adaboost is proposed to estimate the confidence of unlabeled samples in the buffer pool explicitly, the samples with high confidence will be added into the corresponding labeled sets to update classifiers.

3.1 KNN Based on Attibute and Distance Weighted

K-Nearest-Neighbor (KNN) has been widely used in pattern classification on account of its effectiveness and easy implementation [16-18]. The standard Euclidean distance is often used as distance function to measure the dissimilarities among instances. In this paper, we use the more accurate distance function Heterogeneous Value Difference Metric (HVDM) [19]. As we all know that the standard HVDM calculates the distance among samples based on all attributes, it gives equal participation of all attributes. However, the standard HVDM becomes inaccurately when existing large numbers of irrelevant attributes with our research. In order to solve the problem of equal participation of attributes, an effective approach is proposed. All attributes are assigned a weight coefficient, that is to say, each attribute will be weighted based on the different degree of importance. The attribute with zero weight coefficient is called irrelevant attribute. Information gain (i.e. attributes weight coefficients) is used to measure the importance degree of attribute [20]. We utilize the attribute weighted in the HVDM. The improved HVDM method $d(x,y)$ is defined as follows.

$$d(x,y)=\sqrt{\sum_{r=1}^{n} w_r^2 d_r^2(x_r,y_r)}\,. \tag{1}$$

Where n is the number of an instance's attributes. $w_r(r=1,2,...,n)$ is the weight of the attribute A_r. w_r is defined as follows :

$$w_r = Gain(A) = Info(D) - Info_A(D) = -\sum_{i=1}^{n} p_i \log_2(p_i) - \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j)\,. \tag{2}$$

Among them, D represents a meta-component area, A represents an attribute. According to A, the D is divided into v subsets $D_1, D_2, \ldots, D_v$ and P_i is the probability that any tuple in D belongs to class C.

The distance between two instances x and y for attribute r is shown by $d_r(x, y)$, the $d_r(x, y)$ can be defined as follows:

$$d_r(x_r,y_r)=\begin{cases} 1, if\ x_r\ or\ y_r\ is\ unknow; \\ \dfrac{|x_r-y_r|}{4\sigma_r}, if\ r\ is\ continuous; \\ \displaystyle\sum_{c=1}^{C}\left|\frac{N_{r,x,c}}{N_{r,x}}-\frac{N_{r,y,c}}{N_{r,y}}\right|, if\ r\ is\ no\min al \end{cases}. \tag{3}$$

Where x_r is the value of the rth attribute of x. y_r is the value of the rth attribute of y. The σ_r is the standard deviation of the values of the rth attribute. $N_{r,x}$ denotes the number of examples that the values of the rth attribute is x_r, and $N_{r,x,c}$, denotes the number of examples that the values of the rth attribute is x_r and the class label is c. C denotes the number of classes.

The traditional KNN uses the majority voting method to produce class probability estimation, it is thought that the samples in the neighborhood have the equal voting weight, the method ignored that different samples should be assigned different voting weight. An improved method is to weight the votes of each neighborhood differently according to the distance from each neighborhood to the test sample [21]. This improved algorithm can make the result more accurate. It is called K-Nearest Neighborhood with Distance Weighted (KNNDW). The $c(x)$ class with the highest value will be chosen as the final

class.

The weighted class probability estimation method can be obtained as below:

$$c(x) = \arg\max_{c \in label} \sum_{i=1}^{k} \delta(c, c(y_i)). \tag{4}$$

where,

$$\delta(c, c(y_i)) = \begin{cases} 1 & c = c(y_i) \\ 0 & c \neq c(y_i) \end{cases}. \tag{5}$$

Where k is the number of neighbors, $y_1, y_2, ..., y_k$ are the k nearest neighbors of test example x, c represents the class label of examples.

Combining attribute weighting and distance weighting techniques, a new efficient method was proposed by Li et al. [22]. The new method is called Attribute and Distance Weighting K-Nearest Neighborhood (ADWKNN) in this paper. It can obtain higher classification accuracy rate than AWKNN and KNNDW. Therefore, in order to produce more accurate results, we use this improved method in our experiment.

3.2 Difference Value Maximization Confidence Estimation Method Based on Adaboost

The mislabeling problem throughout the co-labeling iterations degrades the performance of classification, in order to solve this problem, we adapt dual-confidence estimation in this paper. The novel mechanism can identify and filter the possibly mislabeled data effectively. Adaboost is an ensemble learning method which is used to estimate the confidence of unlabeled data explicitly. The Difference Value Maximization Confidence Estimation Method based on Adaboost (DVM-AB) is proposed in this paper.

In order to understand this method more clearly, we consider the binary classification problems. Assuming that $P(+1 \mid x_u)$ represents the posterior probability of the instance x_u classified in positive, and $P(-1 \mid x_u)$ represents the posterior probability of the instance x_u classified in negative. If the class probability value of $P(-1 \mid x_u)$ and $P(+1 \mid x_u)$ are close to 0.5, the classifier shows the uncertainty for the prediction. To the contrary, if the difference between the two numbers is greater, the predicted results are more credible. Similarly, In the multiple class classification problem(the number of classes is C), assuming that $P(c_i \mid x_u)$ represents the posterior probability for class $C_i (i = 1, 2, ..., C)$ of the instance, if the class probability value $P(c_i \mid x_u)$ is close to $1/C$, the classifier shows the uncertainty for the prediction. To the contrary, the difference between the posterior probabilities for class C_i and the posterior probabilities for class $non - C_i$ is greater, the predicted results are more credible.

Thus, the method will be defined as follows: Assuming that $P(c_i \mid x_u)$ represents the posterior probability for class $C_i (i = 1, 2, ..., C)$ of the instance x_u, the posterior probability for class $non - C_i$ accordingly are $P(c_i \mid x_u)$, and $\sum_{i=1}^{C} P(c_i \mid x_u) = 1$. If the unlabeled sample x_u are classified to class c_i, the $Confidence\ (x_u, c_j)$ is the difference between the $P(c_i \mid x_u)$ and $\frac{1}{C-1} \sum_{c_j \neq c_j} P(c_{\bar{j}} \mid x_u)$. Its definition is as follow:

$$Confidence\ \ (x_u, c_j) = P(c_j \mid x_u) - \frac{1}{C-1} \sum_{c_j \neq c_j} P(c_{\bar{j}} \mid x_u). \tag{6}$$

A higher value of $Confidence\ (x_u, c_i)$ indicates a higher credibility of the prediction. That is to say, the classifier classified the unlabeled instance x_u as class c_i more certainly. To the contrary, the smaller of the value $Confidence\ (x_u, c_i)$ presents the less credible of the prediction.

4 Impr-Co-Forest Algorithm

Based on the above mentioned three methods, we propose the Impr-Co-Forest algorithm which based on optimized decision tree and dual-confidence estimation method. This algorithm contains three main parts.

At first, we introduce boosting scheme in the training process, the performance of tree will be improved with the introduction of weight method. At the same time, because the newly labeled samples which from the unlabeled sets don't always work, we utilize OOBE to evaluate the performance of single decision tree at the end of each iteration. By comparing the OOBE of tree at the end of two consecutive iterations, we retain the tree with smaller OOBE. Then, dual-confidence estimation is proposed in this paper in order to solve the mislabeling problem throughout the co-labeling iterations. Finally, the majority voting is replaced by weighted vote to produce the class probability estimation. The overall learning process of the proposed Impr-Co-Forest algorithm is shown in the Table 3.

Table 3. The proposed Impr-Co-Forest algorithm

Algorithm 4: Impr-Co-Forest

Input: Labeled training data X_l, unlabeled training data X_u, the number of iterations: T,
The size of the forest: N

Output: The Impr-Co-Forest.

Process:

Step 1:

for j=1:N

For the j^{th} decision tree, generate a initial labeled set X_l^j using the bootstrap aggregation

end for

Step 2:

for $i \in \{1,2,...,T\}$ do

for $j \in \{1,2,...,N\}$ do

(1) Introducing weight scheme of adaboost in the training process. The final decision tree classifier is the weighted combination of weak classifiers $H(x) = sign\left(\sum_{k=1}^{S} \alpha_k h_k(x)\right)$.

Using the dual-confidence estimation method effectively choose the data with high confidence from the newly labeled data, these data are added into corresponding labeled dataset X_i^j to update classifiers.

(2) Retraining the tree with new training set X_i^j, $h_i^j = \mathrm{Re}trainTree(X_i^j)$ compute the OOBE, $e_i^j = oobe(h_i^j, X_l - X_l^j)$.

(4) Retraining the tree with new training set X_{i-1}^j, $h_{i-1}^j = \mathrm{Re}trainTree(X_{i-1}^j)$

Compute the OOBE, $e_{i-1}^j = oobe(h_{i-1}^j, X_l - X_l^j)$

Remain the tree with smaller OOBE.

end for

end for

Step 3:

During the testing, the weighted vote is used to predict the instance in testing set, the weight of each tree is computed using the rate of classification accuracy of its OOB data.

5 Experiments

All programs are based on MATLAB R2016a language. The experimental results are generated on a PC equipped with an Intel Core i7-7700 with 3.6GHz and Nvidia GTX 1060 6G graphics card.

5.1 Dataset Details

The paper provides two kinds of datasets, which are eight UCI datasets and VOC2007-Person dataset. They are described in the form of text and table.

UCI dataset. In the experiments, eight UCI datasets (http://archive.ics.uci.edu/ml/datasets.html) are used to evaluate the performance of the Impr-Co-Forest. Table 4 shows the detail of these datasets, where the pos/neg in the table denotes the percentage of positive examples against that of negative ones.

Table 4. The detail of eight UCI datasets used in experiments

Dataset	Attribute	Size	Class	Pos/neg
Bupa	6	345	2	42.0%/58.0%
Diabetes	8	768	2	65.1%/34.9%
German	24	1000	2	70.0%/30.0%
Ionosphere	34	351	2	35.9%/64.1%
Kr-vs-kp	36	3196	2	52.2%/47.8%
Tic-tac-toe	9	958	2	65.3%/34.7%
Vote	16	435	2	61.4%/38.6%
Wdbc	31	569	2	37.3%/62.7%

VOC2007-person dataset. We add dataset related to the image which is provided by Pascal VOC Challenge in 2007 (http://host.robots.ox.ac.uk/pascal/VOC/voc2007/). The image dataset contains four categories: person, animal, vehicle, indoor. To simplify the experiment, we only select the category of person as our experimental subject. The category of person contains 9963 pictures, can be divided into three classes: 0, +1, -1. The class of 0 has no contribution to experiment and affect our judgment, so it is manually removed. The rest 9786 examples contain two kinds: +1 (there exists human face in the image) and -1 (there no exists human face in the image).

Then we extract characteristic values based on the principle that the extracted characteristics are fixed, they don't change with the size of the image. We can get three datasets by different extracted methods. The first category named colMoment dataset expresses the color distribution of the images by extracting color moments. Color moment feathers consist of the first, second and third moment. Because RGB owns three components, each image get nine attributes. The second category named glcm dataset reflects image texture features by extracting Gray-level co-occurrence matrix. Contrast, correlation, entropy and evenness are calculated respectively in four directions (0°, 45°, 90°, 135°), and the power is computed at last. So each image gets 17 attributes. The third category named colMomentGlcm dataset reflects image color and texture features by extracting color moments and Gray-level co-occurrence matrix, consisting of 26 attributes. Table 5 shows us the details of the dataset.

Table 5. The VOC2007-person datasets in experiments

Extracted feature	The number of attributes	The number of samples	Class	Pos/neg
colMoment	9	9786	2	50.3%/49.7%
glcm	17	9786	2	50.3%/49.7%
colMomentGlcm	26	9786	2	50.3%/49.7%

5.2 Experimental Setups

For each dataset, 75% of samples are kept as training data, while the remaining 25% of samples are used as testing data to evaluate the performance of algorithms. The training dataset is partitioned into L (samples with labels) and U (samples without labels) under different unlabeled rates including 60% and 80%. In general, the L, U, test dataset and original dataset have similar pos/neg ratio. In order to obtain reliable results in the experiment, there are two different L and U are generated by random selection under each unlabeled rate, ten independent runs are implemented for each experiment, the final result is the average error rate of twenty runs.

For all experiments, C4.5 decision trees are used as basic classifiers, both Random Forest and three Co-Forest-style semi-supervised algorithms have same parameters, the size of ensemble $N = 6$, the confidence threshold $\theta = 0.75$. In order to improve the accuracy of Tri-training algorithm, a new algorithm which is based on traditional collaborative training algorithm named Improved Tri-training (ITri) is proposed, the main idea is to enhance the difference among classifiers, three classifiers adopt C4.5 classifier, BP neural network, naive bayes respectively. The kernel of S4VM [24] algorithm adopt

Linear, the number of low-density separator is set to 10, and the sampling number is set to100. For all the experiments, we use HVDM [19] as the distance function to calculate the distance between two examples.

To comprehensively evaluate the performance of our algorithm comparing with other algorithms, five sets of experiments are carried out to prove the effectiveness of our algorithm. We provide the details for each experiment as follows:

Experiment 1. For UCI datasets, Table 6 and Table 7 show the average classification error rates of our algorithm compared with three classical co-training-style algorithms under different unlabeled rates: (1) Co-Training [5], (2) Tri-Training [23], (3) Co-Forest [6].

Experiment 2. For UCI datasets, Table 8 and Table 9 present the average classification error rates of Random Forest [7] and three Co-Forest-style algorithms under different unlabeled rates: (1) Co-Forest, (2) DE-Co-Forest [8], (3) Our method (Impr-Co-Forest).

Experiment 3. For UCI datasets, Figure 1 describes the error rates variation of Random Forest and three Co-Forest-style algorithms along with the iterative times under different unlabeled rates: (1) Co-Forest, (2) DE-Co-Forest, (3) Our method (Impr-Co-Forest).

Experiment 4. For VOC-person dataset, Tables 10, 11, 12 and 13 summarize the average classification error rates of our algorithm compared with the other three semi-supervised algorithms under different unlabeled rates: (1) Co-Forest, (2) S4VM [24], (3) ITri.

5.3 Results and Discussion

Result Ⅰ. Table 6 and Table 7 show the results of Experiment 1. For each row of dataset, the minimum error rate has been boldfaced. In order to compare the performance of different semi-supervised algorithms, the error rates are averaged across all the datasets under all unlabeled rates, and an overall average error rates is obtained. The overall average error rates of Impr-Co-Forest was calculated as 0.157, while that of Co-Forest and Tri-Training are 0.1755 and 0.1895, respectively, the average error rate of Co-Training is 0.2095. The numerical results show that Impr-Co-Forest can make full use of unlabeled sample to enhance the learning performance more effectively.

Table 6. Average error rates of algorithms under the unlabeled rate of 80%

error rate%	Co-Training	Tri-Training	Co-Forest	Impr-Co-Forest
Bupa	0.433	0.392	0.383	**0.355**
Diabetes	0.293	0.287	0.262	**0.243**
German	0.371	0.333	0.296	**0.273**
Ionosphere	0.139	0.139	0.091	**0.086**
Kr-vs-kp	0.075	0.035	0.028	**0.02**
Tic-tac-toe	0.278	0.233	0.251	**0.224**
vote	0.055	0.051	0.05	**0.033**
Wdbc	0.106	0.108	0.088	**0.058**
Average	0.219	0.197	0.181	**0.162**

Table 7. Average error rates of algorithms under the unlabeled rate of 60%

error rate%	Co-Training	Tri-Training	Co-Forest	Impr-Co-Forest
Bupa	0.412	0.372	0.364	**0.342**
Diabetes	0.269	0.278	0.261	**0.241**
German	0.342	0.320	0.278	**0.248**
Ionosphere	0.116	0.133	**0.083**	**0.083**
Kr-vs-kp	0.056	**0.016**	0.023	0.024
Tic-tac-toe	0.259	0.206	0.224	**0.200**
Vote	0.06	0.052	0.048	**0.035**
Wdbc	0.09	0.082	0.079	**0.045**
Average	0.2	0.182	0.17	**0.152**

Result II. Table 8 and Table 9 summarized the average classification error rates of four different algorithms under different unlabeled rates. Suppose the error rate is trained on original labeled datasets L by Random Forest, i.e. the combination of *N* C4.5 decision trees at round 0 is presented by column *initial*. When the iteration process is terminated, the average classification error rate of the *final* hypothesis

generated by the *N* updated classifiers is given by the *final* result of the column. The percentage difference between column *final* and column *initial* are calculated, it describes the improvement of the *final* hypothesis over the *initial* hypothesis. The average results over all the datasets is also shown.

Table 8. Average error rates of algorithms under the unlabeled rate of 80%

Dataset	Random Forest	Co-Forest		DE-Co-Forest		Impr-Co-Forest	
	initial	Final	Impr./%	Final	Impr./%	Final	Impr./%
Bupa	0.393	0.383	2.5	0.378	3.8	**0.355**	9.7
Diabetes	0.279	0.262	6.1	0.256	8.2	**0.243**	12.9
German	0.301	0.296	1.7	0.297	1.3	**0.273**	9.3
Ionosphere	0.122	0.116	4.9	0.124	-1.6	**0.095**	22.1
Kr-vs-kp	0.05	0.028	44	0.022	56	**0.02**	60
Tic-tac-toe	0.262	0.251	4.2	0.252	3.8	**0.224**	14.5
vote	0.068	0.050	26.5	0.044	35.3	**0.033**	51.5
Wdbc	0.1	0.088	12	0.086	14	**0.058**	42
Ave	0.197	0.184	12.7	0.182	15.1	0.163	27.8

Table 9. Average error rates of algorithms under the unlabeled rate of 60%

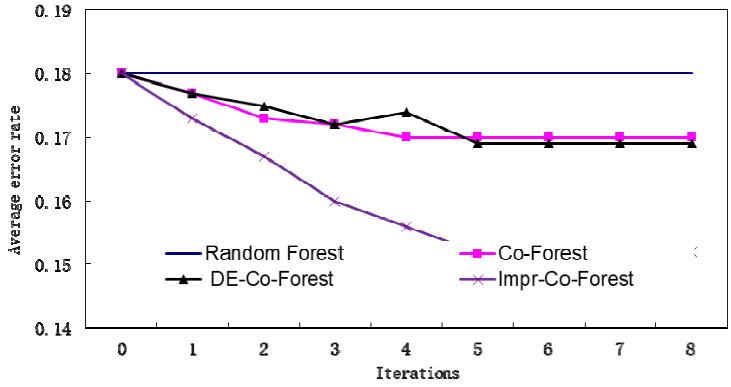
Dataset	Random Forest	Co-Forest		DE-Co-Forest		Impr-Co-Forest	
	initial	Final	Impr./%	Final	Impr./%	Final	Impr./%
Bupa	0.372	0.364	2.2	0.368	1.1	**0.342**	8.1
Diabetes	0.275	0.261	5.1	0.265	3.6	**0.241**	12.4
German	0.288	0.278	3.5	0.270	6.3	**0.248**	13.9
Ionosphere	0.101	**0.083**	17.8	0.086	14.9	**0.083**	17.8
Kr-vs-kp	0.031	**0.023**	25.8	0.024	22.6	0.024	22.6
Tic-tac-toe	0.232	0.224	3.4	0.216	6.9	**0.2**	13.8
vote	0.057	0.048	15.8	0.047	17.5	**0.035**	38.6
Wdbc	0.084	0.079	6.0	0.072	14.3	**0.045**	46.4
Ave	0.18	0.17	10	0.169	10.9	0.152	21.7

For each row of the dataset, the smallest error rate has been boldfaced. The average classification error rate of thirty runs is calculated and recorded as the final result under different unlabeled rates. In the Tables 2 and Table 3, Co-Forest, DE-Co-Forest and Impr-Co-Forest have the same initial column. In order to compare the performance of four different algorithms, the error rates are averaged across all the datasets under all unlabeled rates, and an overall average error rate is obtained. The overall average error rates of Impr-Co-Forest was calculated as 0.1575, while that of Co-Forest and DE-Forest are 0.177 and 0.1755, respectively. Random Forest can't train on unlabeled data, so that it gets the biggest error rate 0.1885 as compared to the other algorithms. The results show that Impr-Co-Forest can make full use of unlabeled sample to enhance the learning performance more effectively.

For the purpose of comparing the performance enhancement among the three Co-Forest-style algorithms, the overall averaged performance improvement of each algorithm across all the datasets can be computed. The Impr-Co-Forest has the greatest performance enhancement, i.e. 24.75%, as compared to the other two algorithms, while DE-Co-Forest and Co-Forest increase by 8.0% and 11.35%, respectively. In detail, under unlabeled rate of 80%, the overall improvement of Impr-Co-Forest is 27.8%, while DE-Co-Forest and Co-Forest increase by 15.1% and 12.7%, respectively. When the unlabeled rate is 60%, the overall improvement of Impr-Co-Forest is 21.7%, which is greater than the 10.9% improvement of DE-Co-Forest and 10% improvement of Co-Forest. The results prove that Impr-Co-Forest algorithm can obtain the greatest performance improvement as compared to Co-Forest and DE-Co-Forest.

Result III. Fig. 1 reveals that the hypotheses generated by three Co-Forest-style algorithms have better performance than hypothesis generated by Random Forest under each different unlabeled proportion. It implies that the three semi-supervised learning algorithms can make full use of the unlabeled data to enhance the performance of initial hypothesis. At each iteration, the Impr-Co-Forest has better performance than that of DE-Co-Forest and Co-Forest. In addition, because the mislabeling data is introduced into the training set during learning process, the performance curve of the four different algorithms under each unlabeled rate shows that the average error rate of Co-Forest and DE-Co-Forest

algorithm has local fluctuations or rise phenomenon, by contrast, the average error rate of Impr-Co-Forest is almost continuously decreased till the termination. This shows that mislabeling problem has a low impact of Impr-Co-Forest which has stronger generalization ability and robustness.



(a) 60% unlabel rate

Fig. 1. Error rates averaged across all the datasets

Result IV. Table 10 and Table 11 described the experiment result of the experiments 4. In order to show a pretty classification performance of the proposed algorithm (Impr-Co-Forest), we introduce VOC2007-Person dataset to compare the average classification error rates of different semi-supervised algorithms under different unlabeled rates. For each dataset row, the minimum error rate has been boldfaced.

Table 10. Average error rates of the compared algorithms under the unlabeled rate of 80%

error rate%	Co-Forest	S4VM	ITri	Impr-Co-Forest
colMoment	0.4352	**0.4082**	0.4136	0.4088
glcm	0.4086	0.4217	0.4013	**0.3755**
colMomentGlcm	0.4037	0.4095	0.3958	**0.3776**
Average	0.4158	0.4131	0.4035	**0.3873**

Table 11. Average error rates of the compared algorithms under the unlabeled rate of 60%

error rate%	Co-Forest	S4VM	ITri	Impr-Co-Forest
colMoment	0.4123	0.4089	0.4097	**0.3902**
glcm	0.4115	0.4136	0.4092	**0.3566**
colMomentGlcm	0.4033	0.4014	0.3870	**0.3357**
Average	0.4090	0.4079	0.4019	**0.3608**

Firstly, we define the concept of the overall error rate: the error rate of the algorithm under two kinds of unlabeled proportion of all classification and then average. The overall error rate of our algorithm (Impr-Co-Forest) is 0.3741, the overall error rate of ITri algorithm is 0.4027, the overall error rate of S4VM algorithm is 0.4105, the overall error rate of Co-Forest algorithm is 0.4124. It is obvious that the overall error rate of Impr-Co-Forest is lower than the other three classic collaborative training algorithms, and it obtains the minimum average error rate under two kinds of unlabeled proportion. S4VM algorithm is regarded as the most popular semi-supervised algorithm in recent years, by extracting color moments, it obtains the minimum average error rate under the unlabeled proportion of 80% than other algorithms. ITri algorithm is second only to Impr-Co-Forest algorithm in the accuracy of image classification.

6 Conclusion

In this paper, the proposed Impr-Co-Forest algorithm applies the weight scheme of adaboost to training procedure. At the same time, the OOBE method is utilized to evaluate the performance of single decision tree at the end of each iteration, by comparing the decision tree's OOBE at the end of two consecutive iterations, the decision tree with smaller OOBE will be retained. By incorporating the dual-confidence

estimation method into co-labeling process, mislabeled samples are identified and removed. In order to prove effectiveness of the algorithm, a series of experiments are implemented on eight UCI datasets and VOC2007-person dataset. The Impr-Co-Forest algorithm is compared with the other five classification algorithms, including Co-Training, Tri-Training, Co-forest, Random Forest and DE-Co-forest. The experimental results present that the proposed algorithm is superior to other algorithms and has stronger generalization ability and robustness.

In the future, we plan to investigate how to reduce the complexity of our algorithm without affecting the classification performance. Furthermore, in addition to the off-the-shelf semi-supervised learning algorithms, it is worthy to study other innovative algorithms to make full use of unlabeled samples.

Acknowledgements

This research is supported by the National Natural Science Foundation of China Nos.61572524 and 61102139, the Fundamental Research Funds for the Central Universities of Central South University Nos.2018zzts181, and 2017 Hunan High-Tech & Innovation Investment Project-'The High Precision 3D Gesture Radar'.

References

[1] X. Zhu, A.-B. Goldberg, R. Brachman, T. Dietterich, Introduction to semi-supervised learning, Semi-supervised Learning, 3(1)(2006) 130.

[2] Z.-X. Zhang, J. Han, J. Deng, Leveraging unlabeled data for emotion recognition with enhanced collaborative semi-supervised learning, IEEE Access 6(2018) 22196-22209.

[3] Y.-M. Yu, J. Wang, Q.-Y. Tan, Semi-supervised multi-label dimensionality reduction based on dependence maximization, IEEE Access 5(2018) 21927-21940.

[4] H. Wu, S. Prasad, Semi-supervised deep learning using pseudo labels for hyperspectral image classification, IEEE Transactions on Image Processing 27(3)(2018) 1259-1270.

[5] A. Blum, T. Mitchell, Combining labeled and unlabeled data with co-training, in: Proc. 1998 Conference on Computational Learning Theory, 1998.

[6] M. Li, Z.-H. Zhou, Improve computer-aided diagnosis with machine learning techniques using undiagnosed samples, IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans 37(6)(2007) 1088-1098.

[7] M. Li, Z. Zhou, SETRED: Self-training with Editing, Lecture Notes in Computer Sciences 3518(2005) 611-621.

[8] C. Deng, A new co-training-style random forest for computer aided diagnosis, Journal of Intelligent Information Systems 36(3)(2011) 253-281.

[9] Y. Cheng, X. Zhao, R. Cai, Z. Li, K. Huang, Y. Rui, Semi-supervised multimodal deep learning for RGB-D object recognition, in: Proc. 2016 International Joint Conference on Artificial Intelligence, 2016.

[10] E.-M. Ardehaly, A. Culotta, Co-training for demographic classification using deep learning from label proportions, in: Proc. 2017 International Conference on Data Mining, 2017.

[11] Y. Freund, R.-E. Schapire, Experiments with a new boosting algorithm, in: Proc. 1996 13th International Conference on International Conference on Machine Learning, 1996.

[12] Y. Cao, Q.-G. Miao, J.-C. Liu, advance and prospects of adaboost algorithm, Acta Automatica Sinica 39(6)(2013) 745-758.

[13] Y. Freund, R.-E. Schapire, A decision-theoretic generalization of on-line learning and an application to boosting, in: Proc. 1997 European Conference on Computational Learning Theory, 1997.

[14] R. Sindoori, K.-S. Ravichandran, B. Santhi, Adaboost technique for vehicle detection in aerial surveillance, International Journal of Engineering & Technology 5(2)(2013) 765-769.

[15] C. Leistner, A. Saffari, J. Santner, Semi-supervised random forests, in: Proc. 2009 IEEE 12th International Conference on IEEE, 2009.

[16] W. Zhao, C. Sun, J. Wang, The research on price prediction of second-hand houses based on KNN and stimulated snnealing slgorithm, International Journal of Smart Home 8(2)(2014) 191-200.

[17] J.-S. Keller, M.-R. Gray, J.-A. Givens, A fuzzy k-nearest neighbor algorithm, IEEE Transactions on Systems Man & Cybernetics SMC-15(4)(2012) 580-585.

[18] K. Patroumpas, C. Koutras, Probabilistic k-nearest neighbor monitoring of moving Gaussians, in: Proc. 2017 International Conference on Scientific and Statistical Database Management, 2017.

[19] D.-R. Wilson, T.-R. Martinez, Improved heterogeneous distance functions, Journal of Artificial Intelligence Research 11(1)(1997) 1-34.

[20] S. Taneja, C. Gupta, K. Goyal, D. Gureja, An enhanced k-nearest neighbor algorithm using information gain and clustering, in: Proc. 2014 International Conference on Advanced Computing & Communication Technologies, 2014.

[21] E. Parvinnia, M. Sabeti, M.-Z. Jahromi, R. Boostani, Classification of EEG signals using adaptive weighted distance nearest neighbor algorithm, Journal of King Saud University - Computer and Information Sciences 26(1)(2014) 1-6.

[22] J. Wu, Z.-H. Cai, S. Ao, Hybrid dynamic k-nearest-neighbour and distance and attribute weighted method for classification, International Journal of Computer Applications in Technology 43(4)(2012) 378-384.

[23] Z. Zhou, M. Li, Tri-training: exploiting unlabeled data using three classifiers, IEEE Transactions on Knowledge & Data Engineering 17(11)(2005) 1529-1541.

[24] Y.-F. Li, Z.-H. Zhou, Towards making unlabeled data never hurt, IEEE Transactions on Pattern Analysis & Machine Intelligence 37(1)(2015) 175-188.