Int J Software Informatics, Vol.2, No.2, December 2008, pp. 95–124 E-mail: ijsi@iscas.ac.cn
International Journal of Software and Informatics, ISSN 1673-7288 http://www.ijsi.org
 Tel: +86-10-62661040

Co-Training by Committee: A Generalized Framework for Semi-Supervised Learning with Committees*

Mohamed Farouk Abdel Hady and Friedhelm Schwenker

(Institute of Neural Information Processing, Ulm University, Ulm 89069, Germany)

Abstract Many data mining applications have a large amount of data but labeling data is often difficult, expensive, or time consuming, as it requires human experts for annotation. Semi-supervised learning addresses this problem by using unlabeled data together with labeled data to improve the performance. Co-Training is a popular semi-supervised learning algorithm that has the assumptions that each example is represented by two or more redundantly sufficient sets of features (views) and additionally these views are independent given the class. However, these assumptions are not satisfied in many real-world application domains. In this paper, a framework called Co-Training by Committee (*CoBC*) is proposed, in which an ensemble of diverse classifiers is used for semi-supervised learning that requires neither redundant and independent views nor different base learning algorithms. The framework is a general single-view semi-supervised learner that can be applied on any ensemble learner to build diverse committees. Experimental results of *CoBC* using *Bagging*, *AdaBoost* and the Random Subspace Method (*RSM*) as ensemble learners demonstrate that error diversity among classifiers leads to an effective Co-Training style algorithm that maintains the diversity of the underlying ensemble.

Key words: data mining; semi-supervised learning; co-training; classification; ensemble learning; decision tree; visual object recognition

Abdel Hady MF, Schwenker F. Co-Training by committee: A generalized framework for semi-supervised learning with committees. *Int J Software Informatics*, **2008, 2(2): 95–124. http://www.ijsi.org/1673-7288/2/95.pdf**

1 Introduction

Supervised learning algorithms require a large amount of labeled data in order to achieve high accuracy and this accuracy degards as the amount of available labeled data decreases. However, labeling data is often difficult, expensive, or time consuming, as it requires the efforts of experienced human annotators. In many practical data mining applications such as content-based image retrieval, computer-aided medical diagnosis[1], object detection and tracking, web page categorization[2], or e-mail

* This work was supported by the German Science Foundation (DFG) under grant SCHW623/4-3 and a scholarship of the German Academic Exchange Service (DAAD).
Corresponding author: Mohamed Farouk Abdel Hady, Email: mohamed.abdel-hady@uni-ulm.de
Manuscript received 28 Sept., 2008; revised 4 Dec., 2008; accepted 13 Dec., 2008; published online Dec.29, 2008.

classification[3], there is often an extremely large pool of data available. In the machine learning literature, there are mainly three paradigms for addressing the problem of combining labeled and unlabeled data to boost the performance: semi-supervised learning, transductive learning and active learning. *Semi-supervised learning* (SSL) refers to methods that attempt to either exploit the unlabeled data for supervised learning where the unlabeled examples are different from the test examples or to exploit the labeled data for unsupervised learning. *Transductive learning* refers to methods which also attempt to exploit unlabeled examples but assuming that the unlabeled examples are exactly the test examples. *Active learning*[4] refers to methods which selects the most important unlabeled examples, and an oracle can be asked for labeling these instances, where the aim is to minimize data labelling. Sometimes it is called selective sampling or sample selection.

Recent research on SSL concentrates into four directions: semi-supervised classification[5−8], semi-supervised regression[9], semi-supervised clustering[10] and semi-supervised dimensionality reduction[11]. Readers interested in recent advances of semi-supervised learning are directed to[12] for an excellent survey.

Co-Training is a popular SSL paradigm introduced by Blum and Mitchell[6]. In this approach, two classifiers are incrementally trained based on two sufficient and independent views. That is, two sets of features each of which is sufficient for learning and conditionally independent given the class. At the initial iteration, two classifiers are trained using the available labeled training examples. Then at each further iteration, each classifier selects and labels some unlabeled examples. The aim is that one classifier can improve the accuracy of the other by providing it with unknown information.

In this paper, we address the problem of how to use unlabeled data to boost the classification performance in application domains with: (1) a small set of labeled examples, (2) a large set of unlabeled examples, and (3) redundantly sufficient and independent views are not available. We present a *single-view committee-based framework* that uses a committee constructed by any ensemble learning algorithm to create a set of diverse classifiers with any base learning algorithm. The main contribution of this study is to replace the hard impractical requirement of *Co-Training* for two or more independent and redundant feature sets by using a set of diverse classifiers. Hence, we call this new framework *Co-Training by Committee* ($CoBC$).

There are two motivations for our study: firstly the recent successful development in ensemble learning algorithms such as *Bagging*[13], *AdaBoost*[14] and the Random Subspace Method (RSM)[15]. Secondly the success of the framework of *active learning with committees* known as *Query by Committee* (QBC)[4]. It iteratively constructs a committee based on the current training set then receives a stream of unlabeled examples as input. Each committee member then classifies the candidate example, QBC measures the degree of disagreement among the committee members and the most informative example is selected for labeling by the user.

In this study, three variants of $CoBC$ were implemented using *Bagging* (denoted by $CoBag$), *AdaBoost* (denoted by $CoAdaBoost$) and RSM (denoted by $CoRSM$) as ensemble methods and C4.5 decision tree induction as base learner. Then it was evaluated using a number of UCI data sets[18] and three object recognition tasks problems[32]. In addition, the influence of changing ensemble size and training set size

was investigated. The results emphasize that *CoBC* variants outperform *Bagging*, *AdaBoost* and the *RSM*, when the labeled data is limited and a large amount of unlabeled data is available. For the sake of comparison, *CoBC* variants were compared with two existing *SSL* methods, *Self-Training* and *Co-Training*.

The rest of the paper is organized as follows: We first give a short overview of semi-supervised learning in Section 2 and ensemble learning in Section 3. Section 4 presents the proposed *CoBC* framework and in Section 5 it is discussed why *CoBC* framework should work. The results of experiments on 15 UCI datasets are reported in Section 6. In Section 7, additional experiments are conducted and kappa-error diagrams are drawn to show the influence of *SSL* on the diversity-accuracy relationship for all the studied ensemble methods. Section 8 describes the application of *CoBC* to three visual object recognition tasks. Finally, Section 9 concludes the paper and outlines the directions of future work in Section 10.

2 Semi-Supervised Learning

The settings of *SSL* can be divided into single-view and multi-view learning (see Table 1). In single-view learning, such as EM[19] and *Self-Training*[5], the *SSL* algorithm receives a single set of features that is used for learning. The Expectation-Maximization (*EM*) is an iterative statistical technique for maximum likelihood estimation in problems with missing data[19]. Given a model for data generation such as Naive Bayes, and data with some missing values, *EM* locally maximizes the likelihood of the parameters and estimate the missing values. *EM* is used as a single-view *SSL* algorithm by treating the labels of the unlabeled examples as missing data[2]. *EM* begins with an initial classifier trained on the labeled examples. It then iteratively uses the current classifier to temporarily label all the unlabeled examples and then trains a new classifier on all labeled examples (the original and the newly labeled) until it converges. While the *EM* algorithm works well when the assumed model of data holds, violations of these assumptions result in poor performance[2].

Table 1 Taxonomy of *SSL* algorithms

Description	SSL algorithm
Single-view, Single-learner	*EM*[19]
Single-classifier	*Self-Training*[5]
Multi-view, Single-learner	*Co-EM*[5]]
Multiple classifiers	*Co-Training*[6]
Single-view, Multi-learner	*Statistical Co-Learning*[23]
Multiple classifiers	*Democratic Co-Learning*[7]
Single-view, Single-learner	*Tri-Training*[8], *Co-Forest*[1]
Multiple classifiers	*Co-Training by Committee*

Self-Training[5] is an incremental algorithm that initially builds a single classifier using the available labeled data. Then it iteratively labels all the unlabeled data, rank the examples by the confidence in their prediction and adds permenantly the most confident examples into the training set. It retrains the underlying classifier with the augmented training set.

In multi-view learning, such as *Co-Training* and *Co-EM*[5], the learning algorithm receives two or more sets of features (views). Blum and Mitchell[6] state two strong

assumptions for successful *Co-Training*: the views should be individually sufficient and conditionally independent given the class. The pseudo-code is shown in Algorithm 1. Recently, researchers have proposed approaches derived from *Co-Training* in three different directions as will be shown in the following subsections.

Algorithm 1. Standard Co-Training

Require: L: set of m labeled training examples
U: set of unlabeled examples
V_1,V_2: feature sets (views) representing an example
T: maximum number of co-training iterations
$BaseLearn$ - base learning algorithm
$\{Pr_k\}_{k=1}^{K}$ - prior probability of class k

1: $h_1^{(0)} = BaseLearn(V_1(L))$ and $h_2^{(0)} = BaseLearn(V_2(L))$
2: **for** $t = 1$ to T
3: **if** U is empty
4: Set $T = t$-1 and abort loop
5: **end if**
6: Apply $h_1^{(t-1)}$ on U.
7: Select a subset S_1 as follows: for each class k, select the $n_k \propto Pr_k$ most confident examples assigned to class k
8: Move S_1 from U to L
9: Apply $h_2^{(t-1)}$ on U.
10: Select a subset S_2 as follows: for each class k, select the $n_k \propto Pr_k$ most confident examples assigned to class k
11: Move S_2 from U to L
12: Re-train classifiers $h_1^{(t)}$ and $h_2^{(t)}$ using the new L
$h_1^{(t)} = BaseLearn(V_1(L))$ and $h_2^{(t)} = BaseLearn(V_2(L))$
13: **end for**
14: **return** combination of the predictions of $h_1^{(T)}$ and $h_2^{(T)}$

2.1 Co-Training with natural multiple views

The standard *Co-Training* was applied in domains with truly independent feature splits satisfying its conditions. In Ref.[3], *Co-Training* is used for email classification where the bags of words that represent email messages were split into two sets: the words from headers and the words from bodies. In Ref.[20], *Co-Training* was used for object recognition using color histograms and orientation histograms extracted from 2D images. In Ref.[21], *Co-Training* is used to improve visual detector for cars in traffic surveillance video where one classifier detects cars in the original grey level images. The second classifier detects cars in images where the background has been removed. Although there are many cases in which there are two or more independent and redundant views, there exists a lot of real-world domains in which such views are not available. Nigam and Ghani[5] showed that *Co-Training* is sensitive to this view independence assumption. Therefore, *Co-Training* with multiple views is impractical in many real-world applications.

2.2 Co-Training with artificial multiple views

In some work, *Co-Training* was applied in domains without natural feature splits

through splitting the available feature set into two views. The random feature split was used by Ref.[5] but it does not consider the required *Co-Training* criteria. In Ref.[22], a method based on artificial feature splitting called *maxInd* has been introduced. It splits the feature set into two views where the aim is to maximize the independence between the two feature subsets but this study had highlighted the need for re-investigating the view independence assumption in *Co-Training*. Also, it was found that there is a trade-off between the dependence of the features within each view, and the independence between views.

2.3 Co-Training without multiple views

In a number of studies[23, 7, 8, 1], the applicability of *Co-Training* using a single view without feature splitting has been investigated. Goldman and Zhou[23] first presented a single-view SSL method, called *Statistical Co-learning*. It used two different supervised learning algorithms with the assumption that each of them produce a hypothesis that partition the input space into a set of equivalence classes. For example, a decision tree partitions the input space with one equivalence class per leaf. In their empirical study, they used ID3 (a decision tree induction algorithm) and HOODG (an algorithm for building decision graphs). They used 10-fold cross validation: (1) to select the most confident examples to label at each iteration and (2) to combine the two hypotheses producing the final decision. There are three drawbacks: first the assumptions concering the used algorithms limits its applicability. Second the amount of available labeled data was insufficient for applying cross validation which is time-consuming.

Then Zhou and Goldman[7] presented another single view method, called *Democratic Co-learning* which is applied to three or more supervised learning algorithms and reduce the need for statistical tests. Therefore, it resolves the drawbacks of *Statistical Co-learning* but it still uses the time-consuming cross-validation technique to measure confidence intervals. These confidence intervals are used to select the most confident unlabeled examples and to combine the hypotheses decisions.

Zhou and Li[8] present a new *Co-Training* style SSL method called *Tri-Training* where initially three classifiers are trained on bootstrap subsamples generated from the original labeled training set. These classifiers are then refined during the *Tri-Training* process, and the final hypothesis is produced via majority voting. The construction of the initial classifiers looks like training an ensemble from the labeled data with *Bagging*[13]. At each *Tri-Training* iteration, an unlabeled example is added to the training set of a classifier if the other two classifiers agree on their prediction under certain conditions. *Tri-Training* is more applicable than previous *Co-Training*-Style algorithms because it neither requires different views as in Ref.[6] nor does it depend on different supervised learning algorithms as in Refs.[23, 7]. There are two limitations: the ensemble size is limited to three classifiers and *Bagging* is used only at the initial iteration. Although the results have shown that using bagged ensemble of three classifiers can improve the generalization ability, better performance is expected when larger-size ensembles and other ensemble learners are used. Also, the influence of *Tri-Training* on the diversity of the 3-member ensemble was not studied.

Recently, Li and Zhou[1] proposed an extension to *Tri-Training*, called *Co-Forest*. It uses an ensemble of $N \geqslant 3$ members, which is denoted by H^*, in order to estimate

the confidence in more efficient way. When determining the most confident examples for each ensemble member h_i ($i = 1, \ldots, N$), all the other classifiers members in H^* except h_i are used. These ensemble members form a new ensemble, which is called the concomitant ensemble of h_i, denoted by $H_i(= H^* - h_i)$. Now the confidence for an unlabeled example can be simply estimated by the degree of agreement on the labeling, i.e. the number of classifiers that agree on the label assigned by H_i. By using this method, *Co-Forest* firstly constructs an initial ensemble of random trees using L and a well-known ensemble method named *Random Forest*[23]. Then at each *Co-Forest* iteration, it refines each ensemble member with unlabeled examples selected by its concomitant ensemble. Note that the unlabeled examples selected by H_i are not removed from U, so they might be selected again by other H_j ($j \neq i$) or the concomitant ensembles in the following iterations. As a result, the training sets of different ensemble members become similar. Therefore, the learning process of *Co-Forest* can hurt the diversity of classifiers if the underlying ensemble method based on training set manipulation to enforce the diversity such as *Bagging*. This degradation of diversity leads to reducing the error improvement caused by exploiting the unlabeled data. To maintain the diversity in the SSL process, *Co-Forest* uses *Random Forest* instead of *Bagging* because *Random Forest* depends also on feature set manipulation to enforce diversity. The main drawback of *Co-Forest*[1] is that it places constraints on the underlying ensemble learners and the size of ensemble. Developing a generalized framework that can maintain the diversity of any ensembles during the SSL process will extend the idea of *Co-Forest* to more general cases, such that it can be applied to more practical data mining applications.

The common reason of the success of the above mentioned approaches is the creation of diversity among a set of classifiers by exploiting different techniques: the use of redundant and sufficient views[6], artificial feature splits[21], different supervised learning algorithms[22, 7], training set manipulation by *Bagging*[8] or feature set manipulation by *Random Forest*[1]. Brown et al. presented in Ref.[24] an exhaustive survey of the various techniques used for creating diverse ensembles, and categorise them, forming a preliminary taxonomy of diversity creation methods. Therefore, the data mining community needs a more general SSL framework that can exploit this treasure of ensemble methods.

3 Ensemble Learning

An ensemble consists of a set of individual predictors (such as neural netwroks or decision trees) whose predictions are combined when classifying a given sample. In Ref.[17], Dietterich explains three fundamental reasons why an ensemble of classifiers is often more accurate than its individual members. An essential condition for an effective ensemble is the error diversiy and the accuracy of its member classifiers[24]. Two classifiers are diverse if they produce different errors for a given set of instances. Many ensemble methods have been developed that use different ways to promote diversity such as: manipulation of training set[13,14], manipulation of input feature set[15], manipulation of the output targets[16] and randomness injection. In the following subsections, the ensemble learners used for $CoBC$ will be presented.

3.1 Bagging

Given a training set L of size m, standard *Bagging*[13] creates N base classifiers $h_i : i = 1, \ldots, N$ (See Algorithm 2). Each classifier is constructed by the base learning algorithm on a bootstrap sample of size m created by random resampling with replacement from the original training set. Each bootstrap sample contains about 63%[13] of the original training set, where each example can appear multiple times. The final prediction of the ensemble for a given sample x is the average of the N class probability distributions produced by ensemble members $P_i(x)$ then select the class with the maximum probability.

Algorithm 2. *Bagging* Algorithm

Require: L - Original training set
$BaseLearn$ - Base learning algorithm
N - number of bagging iterations
1: **for** $i = 1$ to N do
2: $S_i = BootstrapSample(L)$
3: $h_i = BaseLearn(S_i)$
4: **end for**
5: The final hypothesis:
$H(x) = arg\max_{1 \leqslant k \leqslant K} P(x)$ where $P(x) = \frac{1}{N} \sum_{i=1}^{N} P_i(x)$
6: **return** ensemble H

This technique works well for unstable learning algorithms, where a small change in the input training set can lead to a major change in the output hypothesis. Decision trees and neural networks are well-known as unstable algorithms but linear classifiers, k-nearest neighbor and Naive-Bayes algorithms are generally stable especially when the training set size is large.

3.2 Boosting

Boosting is a family of ensemble learning algorithms that are very effective in improving performance compared to the ensemble members. *AdaBoost* described in Ref.[14] is the most popular algorithm (See Algorithm 3). It introduces the diversity through generating a sequence of base classifiers h_i using weighted training sets (weighted by D_1, ..., D_N) such that the weights of training examples misclassified by classifier h_{i-1} are increased and the weights of correctly classified examples are decreased in order to enforce the next classifier h_i to focus on the *hard-to-classify* examples at the expense of the correctly classified examples (bias correction). That is, for each iteration i, the aim of *AdaBoost* is to construct a classifier h_i that improve the training error of classifier h_{i-1}. Consequently, *AdaBoost* stops if the training error of the classifier is zero or worse than random guessing. The outputs of the committee members are combined by *weighted majority vote* to produce the final prediction of the committee where each member is assigned a weight based on its training error. Applying *AdaBoost* to decision trees is successful and is considered one of the best off-the-shelf classification methods. Despite its popularity, *AdaBoost* has two drawbacks[26]: it performs poorly given a small training set and also when there is a lot of training examples with incorrect class label (*mislabeling noise*).

Algorithm 3. *AdaBoost* Algorithm

Require: $L = \{(x_j, y_j)\}_{j=1}^{m}$ - Original training set
$BaseLearn$ - Base learning algorithm
N - number of boosting iterations

1: Initialize $D_1(j) = 1/m \quad \forall j \in \{1, \ldots, m\}$
2: **for** $i = 1$ to N
3: $h_i = BaseLearn(L, D_i)$
4: Calculate the training error of h_i: $\epsilon_i = \sum_{j=1}^{m} D_i(j) \times I(h_i(x_j) \neq y_j)$
5: **if** $\epsilon_i = 0$ or $\epsilon_i \geqslant 1/2$
6: Set $N = i - 1$ and abort loop
7: **end if**
8: Set the weight of h_i: $w_i = \log(\beta_i)$ where $\beta_i = \frac{1-\epsilon_i}{\epsilon_i}$
9: Update weights of training examples:

$$D_{i+1}(j) = D_i(j) \times \begin{cases} \beta_i & if \quad h_i(x_j) \neq y_j; \\ 1 & \text{otherwise.} \end{cases}$$

10: Normalize, $D_{i+1}(j) = D_{i+1}(j) / \sum_{j'=1}^{m} D_{i+1}(j')$
11: **end for**
12: The final hypothesis (*weighted majority vote*):
$p_k(x) = \sum_{h_i(x)=k} w_i$ where $P(x) = \{p_k(x) : k = 1, \ldots, K\}$
$H(x) = arg \max_{1 \leqslant k \leqslant K} P(x)$
13: **return** ensemble H

3.3 The random subspace method

The Random Subspace Method (RSM) is an ensemble learning algorithm proposed by Ho[15]. The diversity is promoted through feature set manipulation instead of training set manipulation. That is, if the given data set is represented by D features, then d features are randomly selected resulting in a d-dimensional random subspace of the original D-dimensional feature space. Then for each random subspace a classifier is constructed. The prediction of the committee for a given sample x is the average of the N class probability distributions produced by ensemble members $P_i(x)$ then select the class with the maximum probability. (in our study, $d = \frac{D}{2}$)

4 Co-Training by Committee

The formal description of the $CoBC$ family of SSL algorithms is provided in Algorithm 4. Given an ensemble learner $EnsembleLearn$, a base learner $BaseLearn$, a set L of labeled examples, and a set U of unlabeled examples, $CoBC$ works as follows: firstly it constructs a committee of N diverse accurate classifiers $H^{(0)}$ by applying $EnsembleLearn$ and $BaseLearn$ to the examples in L. Secondly, $CoBC$ creates a set U' of $poolsize$ examples drawn randomly from U without replacement. Then the following steps will be repeated until a maximum number of iterations T is reached or U' becomes empty. For each iteration t, $CoBC$ applies $H^{(t-1)}$ to the examples in U'. It is computationally more efficient to use U' instead of using the whole set U. $CoBC$ estimates the confidence of $H^{(t-1)}$ predictions (maximum class membership probability) then it creates the set S, which consists of the n_k most confident examples assigned to class k. Then S is removed from U' and added to the labeled training set L. U' is replenished with $|S|$ training examples chosen at

random from U without replacement. Then, $CoBC$ completely discards $H^{(t-1)}$ and constructs the committee $H^{(t)}$ using the updated training set. This is expensive in terms of time but it maintains the diversity among the committee members in the SSL process. It also results in classifiers that are usually more accurate, since they are being totally retrained on the most recent set L.

Algorithm 4. Generalized Co-Training by Committee

Require: L - set of labeled training examples
U - set of unlabeled training examples
T - maximum number of co-training iterations
$EnsembleLearn$ - ensemble learning algorithm
$BaseLearn$ - base learning algorithm
N - number of committee members (*ensemble size*)
K - number of classes
$poolsize$ - number of unlabeled examples in the pool
$\{Pr_k\}_{k=1}^{K}$ - prior probability of class k
1. Construct a committee of classifiers,
$H^{(0)} = EnsembleLearn(L, BaseLearn, N)$
2: Create a set U' of $poolsize$ examples chosen randomly from U without replacement
3: **for** $t = 1$ to T
4: **if** U' is empty
5: Set $T = t$-1 and abort loop
6: **end if**
7: $\forall x_j \in U'$, measure $Confidence(x_j, H^{(t-1)}, K)$
8: Select a subset S as follows: For each class k, select the $n_k \propto Pr_k$ with the highest $confidence$ and $classlabel$ is k
9: Set $U' = U' - S$ and $L = L \cup S$
10: Replenish U' with $|S|$ training examples chosen at random from U
11: $H^{(t)} = EnsembleLearn(L, BaseLearn, N)$
12: **end for**
13: **return** committee $H^{(T)}$

Each variant of the $CoBC$ framework is uniquely defined by the choice of the function $EnsembleLearn$. This choice depends on the properties of the application domain, the required base learning algorithm, the required ensemble size and the available training set size.

An important factor that affects the performance of any *Co-Training* style algorithm is how to measure the confidence of a given unlabeled example which determine its probability of being selected. An ineffective confidence measure can lead to selecting mislabeled examples and adding them to the training set. The confidence measure used in this study is shown in Algorithm 5. The labeling confidence is estimated through consulting the class probability distribution provided by the committee $H^{(t-1)}$ for a given unlabeled example.

In a future work, we will investigate other measures for improving classifcation accuracy. For example, we can measure confidence based on all the previous committees or the previous $windowSize$ committees rather than just the previous one. In addition, in the current version of *AdaBoost*, an ensemble class probability does not depend on the class probability distributions of its ensemble members. A potential

confidence measure can depend on other methods for combining the class probability distributions of the ensemble members.

Algorithm 5. Current implementation of $Confidence(x)$

Require: x - an unlabeled training example
H - a committee of classifiers
K - number of classes
1. Apply H to generate class probability distribution for the given x
$P(x) = \{p_k(x) : k = 1, \ldots, K\}$
2. Set $confidence = \max_{1 \leqslant k \leqslant K} P(x)$
3. Set $classlabel = arg \max_{1 \leqslant k \leqslant K} P(x)$
4. **return** $confidence$ and $classlabel$

5 Why *CoBC* Should Work

Ensemble error reduction is the difference between the average error of the ensemble members and the overall ensemble error. *Error diversity* is the essential requirement to construct an effective ensemble and it is directly proportional to *ensemble error reduction*[24]. *Bagging* and *AdaBoost* rely on the available training data for encouraging diversity. So if the size of the training set is small, then the diversity among the ensemble members will be limited. Consequently, the *ensemble error reduction* will be small. Although the *RSM* does not depend on the training data for promoting diversity, but if the labeled training set size is small, then average error of the ensemble members will be high. Consequently, the *ensemble error* will be high. On the other hand, *CoBC* incrementally adds newly-labeled examples to the training set. Therefore, it can improve the diversity among ensemble members constructed by *Bagging* and *AdaBoost* and improve the average error of ensemble members constructed by *Bagging*, *AdaBoost* and the *RSM*. This leads to the first hypothesis:

Hypothesis 1: *CoBC* will outperform *Bagging*, *AdaBoost* and the *RSM* when the available labeled training set is small.

An ensemble of classifiers is often more accurate than its individual members if their set of misclassified examples are different. Through using a diverse ensemble creation method, the prediction of class labels of unlabeled examples and the measure of confidence based on ensemble is more accurate than using a single classifier. If the ensemble members are identical, *CoBC* degrades to *Self-Training* which is based only on a single classifier. This leads to the second hypothesis:

Hypothesis 2: *CoBC* will outperform *Self-Training*.

To apply *Co-Training with random feature splits*, the original feature set is divided randomly into two subsets, which are not often redundant and conditionally independent given the class label. Therefore, the two classifiers trained on these subsets might be similar. That is, an unlabeled example could be mislabeled by both classifiers. In addition, the performance of co-trained classifiers trained using randomly selected half of the feature set could be worse than a single classifier trained using the full feature set with the same amount of training data. That is, *Self-Training* will outperform *Co-Training with random feature splits*. This leads to the third hypothesis:

Hypothesis 3: *CoBC* will outperform *Co-Training with random feature splits*.

6 Experimental Evaluation

6.1 Methodology

An experimental study is conducted to compare *CoBC* with *Bagging*, *AdaBoost* and the *RSM*. In all ensemble learners, the C4.5 pruned decision tree[28] was used as the base classifier and the used decision tree induction method is *J*48 from the WEKA library[29]. The implementations of *Bagging*, *AdaBoost* and the *RSM* are also from that library. The parameters of *J*48, *Bagging*, *AdaBoost* and the *RSM* were kept as their default values in WEKA unless we state a change. The 15 data sets from UCI Machine Learning repository[18] used in this study are described in Table 2. These data sets are not represented by two sufficient and redundant views.

Table 2 Description of the UCI data sets

Data set	K	D	N_{total}	N_{test}	$\|L\|$ (100%)	$\|L\|$ (20%)	$\|U\|$ (20%)
australian	2	14	690	173	517	103	414
bupa	2	6	345	76	227	45	182
colic	2	22	368	92	276	55	221
diabetes	2	8	768	192	576	115	461
heart-c	2	13	303	76	227	46	181
hepatitis	2	19	155	39	116	23	93
hypothyroid	4	29	3772	943	2829	565	2264
ionosphere	2	34	351	88	263	53	210
segment	7	19	2310	578	1732	346	1386
sick	2	29	3772	943	2829	566	2263
splice	3	60	3190	798	2392	478	1914
tic-tac-toe	2	9	958	240	718	144	574
vehicle	4	18	846	212	634	127	507
vote	2	16	435	109	326	65	261
wdbc	2	30	569	143	426	85	341

In addition, the performance of *CoBC* is compared with two existing *SSL* algorithms, *Self-Training* and standard *Co-Training*[6]. Since *Co-Training* requires two redundant and sufficient views and this condition is not satisfied for the UCI data sets, we randomly partition the D original features of each data set into two disjoint subsets with equal sizes and train two *J*48 decision trees each based on one subset as a separate view. The final output results from multiplying the class membership probabilities of these two co-trained trees. *Self-Training* algorithm uses a single *J*48 decision tree based on the original feature set. Each reported result is the average of performing 5 runs of 4-fold cross-validation. That is, for each data set about 25% of the N_{total} data samples are randomly chosen as test set of size N_{test}. The rest of the data is used as training set which is partitioned randomly into two sets L and U under different labeling rates 20%, 40%, 60% and 80%.

For fair comparison, we set the maximum number of iterations for all *SSL* methods to 30 but the methods can terminate earlier if all the unlabeled examples are labeled. At each iteration and for each class k, the committee selects from U' the n_k most confident examples and adds them to L where $n_k = 3$ and $poolsize = 100$.

6.2 Results

Tables 3 and 4 present the performance of the supervised base learner and ensemble learners when trained on the full training set ($L \cup U$). Tables 5, 6 and 7 present the test error rates of the models at iteration 0 (*initial*) trained only on the labeled data L, after the SSL iteration that achieves the minimum error rate during the SSL process of exploiting the unlabeled data (*best*), and the relative improvement percent ($improv = \frac{initial-best}{initial}$). Note that the learning curve is sometimes non-monotonic, which means that the best iteration is not necessary the last iteration. For each data set, the lowest error rate and the highest improvement achieved are bold faced. The averaged results in these tables verify the hypotheses mentioned in Section 5. That is, the performance of the best committees constructed by *CoBag*, *CoAdaBoost* and *CoRSM* effectively outperform the initial ensembles constructed by *Bagging*, *AdaBoost* and *RSM* under different labeling rates and ensemble sizes. In addition, the results show that at least one variant of *CoBC* can outperform *Self-Training* and *Co-Training with random feature split*.

Table 3 Test error rates of *J*48, *Bagging*, *AdaBoost* and *RSM*, using 100% labeling rate and committees of size 3

Data set	*J*48	*Bagging*-3	*AdaBoost*-3	*RSM*-3
australian	15.83(2.75)	**14.53(1.93)**	16.85(2.36)	20.64(3.70)
bupa	37.39(5.45)	34.14(5.16)	34.37(4.26)	**32.40(4.45)**
colic	**14.68(2.48)**	15.11(2.36)	18.75(2.77)	20.60(3.02)
diabetes	25.55(3.13)	**25.47(2.82)**	27.14(2.64)	31.02(2.55)
heart-c	24.30(4.67)	22.78(3.66)	23.50(4.04)	**19.28(4.55)**
hepatitis	19.60(6.10)	19.73(4.74)	**18.20(5.92)**	18.71(3.78)
hypothyroid	0.44(0.17)	**0.42(0.15)**	0.57(0.32)	1.88(0.41)
ionosphere	11.12(3.31)	9.53(3.57)	9.81(3.30)	**8.44(3.02)**
segment	3.54(0.77)	3.29(0.63)	**2.77(0.74)**	8.88(0.92)
sick	1.25(0.51)	1.38(0.58)	**1.23(0.43)**	5.43(0.38)
splice	**6.34(0.70)**	6.36(0.74)	8.14(8.47)	10.53(1.14)
tic-tac-toe	16.04(2.89)	13.03(3.06)	**10.21(2.22)**	20.48(2.33)
vehicle	27.67(2.54)	27.29(2.44)	**25.96(1.97)**	26.67(2.89)
vote	**4.05(1.77)**	4.51(1.81)	5.71(2.16)	9.33(2.90)
wdbc	6.37(1.86)	5.59(1.87)	**5.42(1.80)**	5.98(1.92)
ave.	14.28	**13.54**	13.91	16.02

Table 8 shows the summary of a pairwise comparison for four different parameter settings based on the 15 UCI data sets. The entry a(i,j) indicates the number of data sets at which the method of the column(j) outperforms the method of the row(i). The number in the parenthesis shows how many of these data sets the difference is statistically significant at 0.05 level of significance using *corrected paired t-test*[29].

Interestingly, we find that for many data sets *supervised ensemble learners Bagging*, *AdaBoost* and *RSM*, using labeled data only, outperform *semi-supervised learners Self-Training* and *Co-Training*, that exploit the unlabeled data, and the number of these data sets increases as the ensemble size increases. For example, from Table 8(a) and Table 8(b), *Boosted ensemble* of size 3 outperforms *Self-Training* and *Co-Training* in 4 and 5 data sets, respectively, compared to 8 and 6 data sets, respectively, for *Boosted ensemble* of size 10. This emphasizes the claim that ensemble learning can

be a good solution to improve the performance when the amount of labeled data is limited even if the unlabeled data is unused.

The error of *CoRSM* is not satisfactory compared to *CoBag* and *CoAdaBoost*. This poor performance is attributed to constructing small-sized ensembles using *RSM*. This results emphasizes the conclusions in Ref.[15] which states that *RSM* does not work well and is outperformed by *Bagging* and *AdaBoost* when the ensemble size is small. In *RSM* each classifier is trained based on a random feature subset, therefore using a small number of classifiers increases the possibility of neglecting some features. That means, it leads to loss of some discriminating information.

Table 4 Test error rates of *J*48, *Bagging*, *AdaBoost* and *RSM*, using 100% labeling rate and committees of size 10

Data set	*J*48	*Bagging*-10	*AdaBoost*-10	*RSM*-10
australian	15.83(2.75)	**14.09(2.09)**	15.71(2.51)	16.90(2.91)
bupa	37.39(5.45)	31.12(5.04)	33.46(4.19)	33.45(3.77)
colic	**14.68(2.48)**	14.52(2.37)	17.94(2.92)	14.95(2.34)
diabetes	25.55(3.13)	**24.25(3.33)**	26.91(2.90)	25.53(2.05)
heart-c	24.30(4.67)	22.18(3.50)	21.92(4.01)	**19.35(3.61)**
hepatitis	19.60(6.10)	18.70(4.88)	**16.51(3.99)**	18.96(3.73)
hypothyroid	0.44(0.17)	0.41(0.17)	0.39(0.14)	0.57(0.21)
ionosphere	11.12(3.31)	9.93(3.96)	7.81(3.00)	**8.04(3.74)**
segment	3.54(0.77)	2.85(0.59)	**1.87(0.51)**	4.23(0.63)
sick	1.25(0.51)	1.24(0.50)	**1.15(0.35)**	5.23(0.39)
splice	**6.34(0.70)**	5.84(0.46)	6.48(1.15)	6.28(0.88)
tic-tac-toe	16.04(2.89)	9.81(2.12)	**4.66(1.49)**	23.16(1.77)
vehicle	27.67(2.54)	26.70(3.10)	**23.15(2.41)**	25.89(2.17)
vote	**4.05(1.77)**	4.19(1.85)	4.55(1.70)	6.81(2.10)
wdbc	6.37(1.86)	4.82(1.94)	**4.40(1.77)**	4.86(1.91)
ave.	14.28	12.71	**12.46**	14.28

7 Additional Experiments

7.1 Dominance rank

The dominance of a method is the difference between the number of data sets for which it significantly outperforms the others (W) and the number of data sets for which it significantly underperforms the others (L). Table 9(a) shows the top 24 *SSL* methods sorted by dominance rank, due to a pairwise comparison of each method against the others. In addition, Table 9(b) shows the dominance ranks of *CoBC* variants, in descending order, due to a pairwise comparison of each of them against *Self-Training* and *Co-Training with random feature split*.

7.2 Influence of labeled training set size

From Tables 9(a) and 9(b), we can observe the influence of increasing the labeling rate from 20% to 40% while keeping the committee size (N) fixed. The dominance rank of the *CoAdaBoost* increases from 37 to 39 (with N=3) and from 44 to 53 (with N=10). The dominance rank of the *CoBag* increases from 33 to 48 (with N=3) and from 30 to 36 (with N=10). This increment for *CoBC* variants is at the expenses of the ranks of *Self-Training* and *Co-Training* that decreases, see Table 9(a) for more details. From Table 9(b), the rank of *CoAdaBoost* increases from 9 to 14 (with N=10) and do not change (with N=3). In addition, Fig.1 plots test errors for seven

Table 5 Test error rates of the initial and best models and the relative improvements of *Self-Training*, *Co-Training* and *CoBC* variant rate and using committees of size 3

SSL Method	Self-Training-20%			Co-Training-20%			CoBag-20%-3			CoAdaBoost-20%-3			C
Dataset	*initial*	*best*	*improv*	*initial*	*best*	*improv*	*initial*	*best*	*improv*	*initial*	*best*	*improv*	*initia*
australian	17.19	16.09	6.40%	21.60	18.30	15.28%	**16.24**	**13.66**	15.89%	19.34	14.93	**22.80%**	23.95
bupa	40.46	37.56	7.17%	43.13	40.98	4.98%	39.72	32.87	**17.25%**	**38.71**	**32.57**	15.86%	40.41
colic	**17.83**	16.74	6.11%	19.19	16.91	11.88%	19.79	**15.60**	21.17%	21.69	16.74	**22.82%**	23.21
diabetes	30.78	27.82	9.62%	29.95	26.49	11.55%	**28.83**	**25.32**	12.17%	30.84	25.55	**17.15%**	33.47
heart-c	26.26	24.09	8.26%	24.09	20.93	13.12%	25.28	20.33	19.58%	**23.76**	**18.95**	**20.24%**	24.35
hepatitis	20.25	19.35	4.44%	21.41	19.99	6.63%	21.68	**17.56**	**19.00%**	21.83	18.19	16.67%	**19.74**
hypothyroid	**1.03**	0.95	7.77%	5.68	5.28	7.04%	1.22	**0.77**	**36.89%**	1.54	1.11	27.92%	2.55
ionosphere	16.47	14.31	13.11%	**10.60**	**8.84**	16.60%	14.08	9.63	**31.61%**	16.99	11.69	31.19%	12.48
segment	8.67	7.87	9.23%	**6.32**	6.84	-8.23%	8.40	6.68	20.48%	7.40	**5.68**	**23.24%**	15.39
sick	**2.10**	2.06	1.90%	5.35	3.97	**25.79%**	2.15	**1.77**	17.67%	2.42	2.10	13.22%	6.01
splice	13.15	12.69	3.50%	13.80	14.49	-5.00%	12.91	10.16	**21.30%**	**11.08**	**9.90**	10.65%	16.19
tic-tac-toe	27.96	27.29	2.40%	31.63	28.05	11.32%	28.23	24.62	**12.79%**	**24.35**	**21.59**	11.33%	27.71
vehicle	36.39	33.79	7.14%	**27.88**	**24.73**	11.30%	35.84	30.21	**15.71%**	33.81	30.41	10.06%	36.10
vote	5.89	5.66	3.90%	5.94	4.92	17.17%	5.85	**4.37**	**25.30%**	**5.80**	4.74	18.28%	10.72
wdbc	9.46	7.63	19.34%	**7.18**	**4.68**	**34.82%**	7.49	5.49	26.70%	7.77	7.14	8.11%	7.67
ave.	18.26	16.93	7.30%	18.25	16.36	10.36%	17.85	**14.60**	**18.18%**	**17.82**	14.75	17.22%	20.00
no. of wins	3	0	0	4	3	2	2	**7**	**8**	**5**	5	5	1

Table 6 Test error rates of the initial and best models and the relative improvements of *Self-Training*, *Co-Training* and *CoBC* variant rate and using committees of size 10

SSL Method	Self-Training-20%			Co-Training-20%			CoBag-20%-10			CoAdaBoost-20%-10			C
Dataset	*initial*	*best*	*improv*	*initial*	*best*	*improv*	*initial*	*best*	*improv*	*initial*	*best*	*improv*	*initial*
australian	17.19	16.09	6.40%	21.60	18.30	15.28%	**15.31**	**12.99**	15.15%	18.79	13.51	**28.10%**	18.58
bupa	40.46	37.56	7.17%	43.13	40.98	4.98%	38.31	35.24	8.01%	**37.15**	**33.04**	**11.06%**	39.94
colic	17.83	16.74	6.11%	19.19	16.91	11.88%	17.89	**14.35**	19.79%	21.58	15.98	**25.95%**	**16.96**
diabetes	30.78	27.82	9.62%	29.95	26.49	11.55%	**27.48**	**24.40**	11.21%	29.40	24.48	**16.73%**	28.91
heart-c	26.26	24.09	8.26%	24.09	20.93	13.12%	23.44	20.47	12.67%	22.57	**16.84**	**25.39%**	**22.50**
hepatitis	20.25	19.35	4.44%	21.41	19.99	6.63%	**19.87**	**17.55**	11.68%	21.68	17.68	**18.45%**	20.25
hypothyroid	1.03	0.95	7.77%	5.68	5.28	7.04%	**0.93**	**0.85**	8.60%	1.13	0.91	**19.47%**	1.48
ionosphere	16.47	14.31	13.11%	**10.60**	8.84	16.60%	12.82	10.43	18.64%	15.73	9.29	**40.94%**	12.37
segment	8.67	7.87	9.23%	6.32	6.84	-8.23%	7.85	6.63	15.54%	**6.24**	**4.51**	**27.72%**	8.68
sick	2.10	2.06	1.90%	5.35	3.97	**25.79%**	**2.05**	**1.77**	13.66%	2.20	1.77	19.55%	5.76
splice	13.15	12.69	3.50%	13.80	14.49	-5.00%	11.12	9.83	11.60%	**10.14**	**8.24**	**18.74%**	12.59
tic-tac-toe	27.96	27.29	2.40%	31.63	28.05	11.32%	25.50	23.51	7.80%	**20.90**	**17.06**	**18.37%**	29.51
vehicle	36.39	33.79	7.14%	**27.88**	**24.73**	**11.30%**	32.37	29.72	8.19%	30.93	27.93	9.70%	30.81
vote	5.89	5.66	3.90%	5.94	4.92	17.17%	6.08	**4.42**	**27.30%**	**5.47**	4.56	16.64%	8.65
wdbc	9.46	7.63	19.34%	7.18	**4.68**	**34.82%**	**7.07**	5.17	26.87%	7.07	5.49	22.35%	7.63
ave.	18.26	16.93	7.30%	18.25	16.36	10.36%	**16.54**	14.49	12.40%	16.73	**13.42**	**19.80%**	17.64
no. of wins	0	0	0	2	2	3	**6**	**7**	1	5	5	**11**	2

Table 7 Test error rates of the initial and best models and the relative improvements of *Self-Training*, *Co-Training* and *CoBC* variant
rate and using committees of size 10

SSL Method	Self-Training-40%			Co-Training-40%			CoBag-40%-10			CoAdaBoost-40%-10			C
Dataset	*initial*	*best*	*improv*	*initial*	*best*	*improv*	*initial*	*best*	*improv*	*initial*	*best*	*improv*	*initia*
australian	15.77	14.44	8.43%	18.81	17.05	9.36%	**14.73**	**12.06**	18.13%	15.89	12.62	**20.58%**	17.98
bupa	39.77	35.13	11.67%	41.27	37.86	8.26%	37.45	32.81	12.39%	**37.39**	**31.01**	**17.06%**	39.42
colic	16.36	15.55	4.95%	17.40	15.93	8.45%	16.31	**14.08**	13.67%	19.03	15.22	**20.02%**	**16.3**
diabetes	28.39	25.89	8.81%	27.87	25.79	7.46%	**25.50**	**23.05**	9.61%	27.82	23.52	**15.46%**	27.19
heart-c	25.48	22.58	11.38%	23.44	20.20	13.82%	21.78	18.62	14.51%	22.58	**15.92**	**29.50%**	**20.7**
hepatitis	20.51	20.12	1.90%	**17.94**	15.63	12.88%	18.19	17.02	6.43%	19.62	**15.24**	**22.32%**	18.58
hypothyroid	0.68	0.65	4.41%	5.26	5.01	4.75%	**0.67**	**0.54**	19.40%	0.74	0.54	**27.03%**	0.97
ionosphere	13.85	11.57	16.46%	**9.07**	8.55	5.73%	10.61	7.98	24.79%	11.12	**7.30**	**34.35%**	10.72
segment	6.23	5.43	12.84%	5.05	5.35	-5.94%	5.50	4.38	20.36%	**3.96**	**2.89**	**27.02%**	6.80
sick	1.90	1.84	3.16%	4.63	3.98	14.04%	1.88	1.56	17.02%	**1.86**	**1.42**	**23.66%**	5.61
splice	9.23	9.08	1.63%	11.96	11.13	6.94%	**8.16**	**7.03**	13.85%	10.58	8.52	**19.47%**	8.65
tic-tac-toe	22.82	21.07	7.67%	27.96	26.06	6.80%	19.32	16.42	15.01%	**12.07**	**9.11**	**24.52%**	26.60
vehicle	33.10	30.03	9.27%	**26.15**	**23.74**	9.22%	29.70	26.81	9.73%	28.37	24.64	**13.15%**	28.97
vote	5.06	4.37	13.64%	6.21	5.01	19.32%	**4.65**	**3.73**	19.78%	5.93	3.87	**34.74%**	7.91
wdbc	7.25	6.51	10.21%	6.16	4.12	33.12%	5.59	4.29	23.26%	**5.38**	**3.34**	**37.92%**	5.77
ave.	16.43	14.95	8.99%	16.61	15.03	9.54%	**14.67**	12.69	13.48%	14.82	**11.68**	**21.22%**	16.15
no. of wins	0	0	0	3	1	0	5	6	0	**5**	**8**	**15**	2

data sets versus different labeling rates using committees of size 10. For all labeling rates, one or more variants of *CoBC* outperforms both *Self-Training* and *Co-Training*.

Table 8 Summary of results. The entry a(i,j) indicates the number of data sets at which the method of the column(j) outperforms the method of the row(i).

(a) 20%-3

	Supervised learners				Semi-Supervised learners				
	J48	*B*	*A*	*R*	*ST*	*CT*	*CB*	*CA*	*CR*
J48	-	10(0)	8(0)	7(0)	15(0)	9(2)	15(6)	13(6)	7(0)
Bagging (*B*)	5(0)	-	7(0)	4(0)	13(0)	10(1)	15(2)	15(4)	7(0)
AdaBoost (*A*)	7(0)	8(0)	-	3(0)	11(0)	10(1)	14(4)	15(4)	6(0)
RSM (*R*)	8(5)	11(5)	12(4)	-	14(6)	11(4)	15(8)	15(9)	15(0)
Self-Training (*ST*)	0(0)	2(0)	4(0)	1(0)	-	7(1)	15(2)	12(3)	5(0)
Co-Training (*CT*)	6(1)	5(1)	5(1)	4(0)	8(1)	-	12(4)	12(4)	4(1)
CoBag (*CB*)	0(0)	0(0)	1(0)	0(0)	0(0)	3(0)	-	5(0)	1(0)
CoAdaBoost (*CA*)	1(0)	0(0)	0(0)	0(0)	2(0)	3(1)	10(0)	-	2(0)
CoRSM (*CR*)	8(4)	8(5)	9(4)	0(0)	10(5)	11(4)	14(7)	13(8)	-
no.ofWins	35	44	46	19	73	64	**110**	100	47
Sig.	(10)	(11)	(9)	(0)	(12)	(14)	(33)	**(38)**	(1)

(b) 20%-10

	Supervised learners				Semi-Supervised learners				
	J48	*B*	*A*	*R*	*ST*	*CT*	*CB*	*CA*	*CR*
J48	-	13(0)	10(3)	8(1)	15(0)	9(2)	15(8)	15(7)	12(3)
Bagging (*B*)	2(0)	-	7(0)	4(0)	4(0)	8(1)	15(2)	15(6)	8(0)
AdaBoost (*A*)	5(0)	7(0)	-	7(0)	7(0)	9(1)	13(4)	15(7)	8(2)
RSM (*R*)	6(1)	11(1)	8(3)	-	10(1)	12(2)	15(8)	15(8)	15(0)
Self-Training (*ST*)	0(0)	11(0)	8(1)	4(0)	-	7(1)	15(3)	15(4)	7(0)
Co-Training (*CT*)	6(1)	7(2)	6(3)	3(1)	8(1)	-	12(3)	12(5)	10(1)
CoBag (*CB*)	0(0)	0(0)	2(0)	0(0)	0(0)	3(0)	-	7(2)	3(0)
CoAdaBoost (*CA*)	0(0)	0(0)	0(0)	0(0)	0(0)	3(0)	7(0)	-	2(0)
CoRSM (*CR*)	3(1)	7(1)	7(3)	0(0)	8(1)	5(1)	12(4)	13(5)	-
no.ofWins	22	56	48	26	52	56	104	**107**	65
Sig.	(3)	(4)	(13)	(2)	(3)	(8)	(32)	**(44)**	(6)

(c) 40%-3

	Supervised learners				Semi-Supervised learners				
	J48	*B*	*A*	*R*	*ST*	*CT*	*CB*	*CA*	*CR*
J48	-	11(0)	8(1)	6(0)	15(0)	10(2)	15(11)	13(7)	6(1)
Bagging (*B*)	4(0)	-	5(0)	2(0)	11(0)	10(1)	15(7)	13(5)	6(0)
AdaBoost (*A*)	7(0)	10(0)	-	3(0)	11(0)	10(1)	15(3)	15(7)	6(0)
RSM (*R*)	9(5)	12(6)	12(4)	-	12(8)	13(5)	15(12)	15(8)	15(2)
Self-Training (*ST*)	0(0)	3(0)	4(0)	3(0)	-	7(1)	14(4)	12(3)	5(0)
Co-Training (*CT*)	5(1)	5(3)	5(2)	2(0)	8(3)	-	12(4)	11(3)	5(0)
CoBag (*CB*)	0(0)	0(0)	0(0)	0(0)	0(0)	3(1)	-	6(0)	0(0)
CoAdaBoost (*CA*)	2(0)	2(0)	0(0)	0(0)	3(0)	4(0)	8(0)	-	1(0)
CoRSM (*CR*)	9(4)	9(5)	9(4)	0(0)	10(6)	10(3)	15(8)	14(6)	-
no.ofWins	36	52	43	16	70	67	**109**	99	44
Sig.	(10)	(14)	(11)	(0)	(17)	(14)	**(49)**	(39)	(3)

(d) 40%-10

	Supervised learners				Semi-Supervised learners				
	J48	*B*	*A*	*R*	*ST*	*CT*	*CB*	*CA*	*CR*
J48	-	15(1)	10(2)	9(0)	15(0)	10(2)	15(10)	15(10)	10(3)
Bagging (*B*)	0(0)	-	6(2)	2(0)	7(0)	8(1)	15(6)	14(6)	8(2)
AdaBoost (*A*)	5(0)	9(0)	-	6(0)	8(0)	8(1)	13(2)	15(8)	9(1)
RSM (*R*)	6(2)	12(3)	9(3)	-	9(4)	13(2)	15(7)	15(9)	15(3)
Self-Training (*ST*)	0(0)	8(0)	6(2)	6(0)	-	7(1)	15(4)	15(7)	8(1)
Co-Training (*CT*)	5(1)	8(3)	7(2)	2(2)	8(3)	-	12(4)	14(7)	9(2)
CoBag (*CB*)	0(0)	0(0)	2(1)	0(0)	0(0)	3(0)	-	9(2)	2(0)
CoAdaBoost (*CA*)	0(0)	1(0)	0(0)	0(0)	0(0)	1(0)	5(1)	-	2(0)
CoRSM (*CR*)	5(1)	7(2)	6(3)	0(0)	7(3)	6(1)	13(5)	12(5)	-
no.ofWins	21	60	46	25	54	56	103	**109**	63
Sig.	(4)	(9)	(15)	(2)	(10)	(8)	(39)	**(54)**	(12)

Table 9 Dominance ranking using the significant differences (*W* - *L*)

(a) Each model against others

#	Method	W-L	*W*	*L*	#	Method	W-L	*W*	*L*
1	CA-40%-10	53	54	1	13	B-40%3	1	14	13
2	CB-40%-3	48	49	1	14	CT-20%(3)	1	14	13
3	CA-20%-10	44	44	0	15	A-40%3	0	11	11
4	CA-40%-3	39	39	0	16	A-20%3	0	9	9
5	CA-20%-3	37	38	1	17	A-20%-10	-1	13	14
6	CB-40%-10	36	39	3	18	CT-40%-(3)	-2	14	16
7	CB-20%-3	33	33	0	19	J48-20%-(3)	-4	10	14
8	CB-20%-10	30	32	2	20	ST-40%-(10)	-5	10	15
9	ST-40%(3)	9	17	8	21	CR-40%-10	-8	12	20
10	ST-20%(3)	6	12	6	22	B-40%-10	-8	9	17
11	B-20%-3	4	11	7	23	CT-20%-(10)	-9	8	17
12	A-40%-10	3	15	12	24	CT-40%-(10)	-16	8	24

(b) Each *CoBC* variant against *Self-Training* and *Co-Training*

#	Method	W-L	*W*	*L*	#	Method	W-L	*W*	*L*
1	CA-40%-10	14	14	0	7	CB-20%-10	6	6	0
2	CA-20%-10	9	9	0	8	CB-20%-3	6	6	0
3	CB-40%-10	8	8	0	9	CR-20%-10	-1	1	2
4	CB-40%-3	7	8	1	10	CR-40%-10	-1	3	4
5	CA-20%-3	6	7	1	11	CR-20%-3	-8	1	9
6	CA-40%-3	6	6	0	12	CR-40%-3	-9	0	9

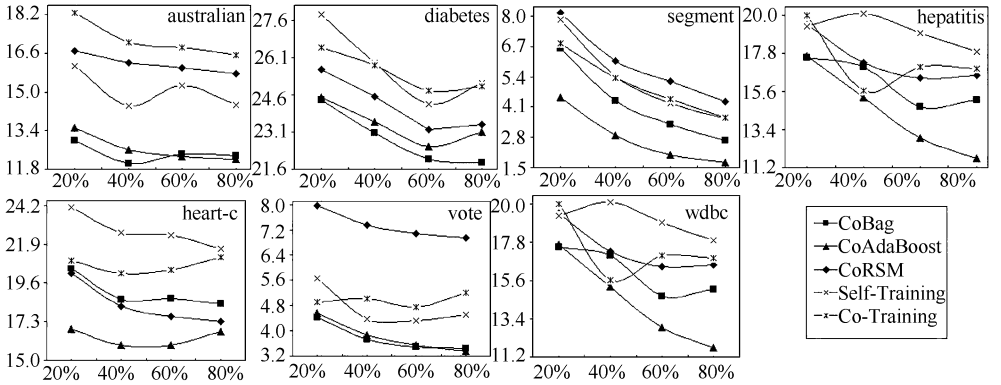

Figure 1. Influence of changing training set size on test error for seven data sets using committees of size 10 (x-axis = *labeling rate*, y-axis = *ensemble error*

7.3 Influence of ensemble size

From Table 9(a) and 9(b), we can conclude the influence of increasing the committees size from 3 to 10 while keeping labeling rates (LR) fixed. The dominance rank of *CoAdaBoost* increases from 39 to 53 (under LR=40%) at the expenses of both *Self-Training* that decreases from 9 to -5 and *Co-Training* that decreases from -2 to -16. Also, *CoAdaBoost* dominance rank increases from 37 to 44 (under LR=20%). Surprisingly, the dominance rank of *CoBag* decreases from 48 to 36 (under LR=40%) and from 33 to 30 (under LR=20%). From Table 9(b), the rank of *CoAdaBoost* increases from 6 to 14 (under LR=40%) and from 6 to 9 (under LR=20%). Also, the rank of *CoRSM* increases from -9 to –1 (under LR=40%) and from –8 to –1 (under LR=20%). Therefore, *CoAdaBoost* and *CoRSM* gained from increasing the ensemble size at the expenses of *CoBag*, *Self-Training* and *Co-Training*.

7.4 Diversity-Error diagrams

It is well known that the combination of the output of a set of classifiers is only useful if they have uncorrelated classification errors which is called the diversity of the ensemble. In the literature, there are several measures of ensemble diversity[26] that are used to prove that increasing diversity while maintaining the average accuracy of ensemble members, should increase ensemble accuracy[25].

In this study, the *Kappa agreement measure* is used as a pairwise diversity measure and the *Kappa-Error diagram* introduced in Ref.[31] is drawn as a way to visualize the influence of *CoBC* on the relationship between diversity-accuracy of the constructed ensembles. *Kappa* measures the level of agreement between two classifiers decisions and considers the agreement by chance. It is defined as follows: Given two classifiers h_1 and h_2, K classes and m examples, we can define a coincidence matrix C where element C_{ij} represents the number of examples that are assigned by the first classifier to class i and by the second classifier to class j.

Then, the agreement measure κ is defined as follows:

$$\kappa = \frac{\theta_1 - \theta_2}{1 - \theta_2} \tag{1.1}$$

where

$$\theta_1 = \sum_{i=1}^{K} \frac{C_{ii}}{m} \quad \text{and} \quad \theta_2 = \sum_{i=1}^{K} \left(\sum_{j=1}^{K} \frac{C_{ij}}{m} \right) \left(\sum_{j=1}^{K} \frac{C_{ji}}{m} \right) \tag{1.2}$$

If h_1 and h_2 are identical, only the main diagonal of M will contain non-zero elements (θ_1=1) and $\kappa = 1$. If h_1 and h_2 are totally different, their agreement (θ_1) will be the same as the agreement by chance (θ_2) and $\kappa = 0$. If h_1 and h_2 are negatively dependent, then $\kappa < 0$ and when one classifier is wrong, the other has more than random chance of being correct.

The x-axis of the Kappa-error diagram represents the $\kappa_{i,j}$ between two ensemble members h_i and h_j while the y-axis represents the average error $E_{i,j} = \frac{E_i + E_j}{2}$. Therefore, an ensemble of N members has a cloud of $\frac{N(N-1)}{2}$ points in the diagram. The Kappa-error diagrams were calculated on the test sets. Since small values of κ indicate better diversity and small values of $E_{i,j}$ indicate better accuracy, the most desirable cloud will lie in the bottom left corner.

Figure 2 shows only the centroids of the clouds of Kappa-error dots of different ensembles in the same plot for each data set instead of ploting the whole clouds because the clouds are heavily overlapping. This is useful for visual evaluation of the relative positions of clouds for the different ensembles. The axes of the plots are rescaled in order to make the relative positions of clouds clearly visible. The unfilled shape represents the *initial ensemble* before SSL while the corresponding filled one represents the *best ensemble* after SSL. From these plots, we can observe the following:

- At most cases, the clouds of *Co-Training* lie above the clouds of $CoBag$ and below the clouds of $CoAdaBoost$ and $CoRSM$.
- At most cases, the clouds of *Co-Training* are located to the left of the clouds for $CoRSM$, $CoBag$ and $CoAdaBoost$ due to random feature splitting.

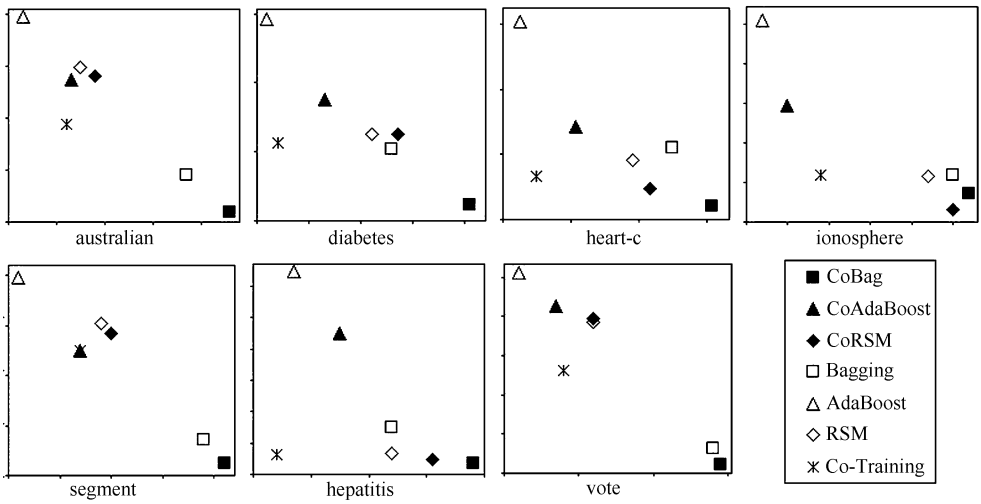

Figure 2. Centroids of the Kappa-error clouds for seven data sets with committees of size 10 and under 20% label rate (x-axis = average of $\kappa_{i,j}$, y-axis = average of $E_{i,j}$)

Figure 3 shows the trade-off between *ensemble diversity* and *ensemble accuracy*. We have the following observations:

- At most cases, the centroids of *Co-Training* lie above the centroids of *CoBag*, *CoRSM* and *CoAdaBoost* and below the centroids of *Bagging*, *RSM* and *AdaBoost*.
- At most cases, the centroids of *CoAdaBoost*, *CoBag* and *CoRSM* are located to the right and below the centroids for *AdaBoost*, *Bagging* and *RSM* respectively. That is, *CoBC* improves the accuracy at the expense of the diversity that is *slightly* reduced due to knowledge exchange between committee members.
- *CoAdaBoost* is more diverse than *CoBag* and *CoRSM* as *AdaBoost* is more diverse than *Bagging* and *RSM*.

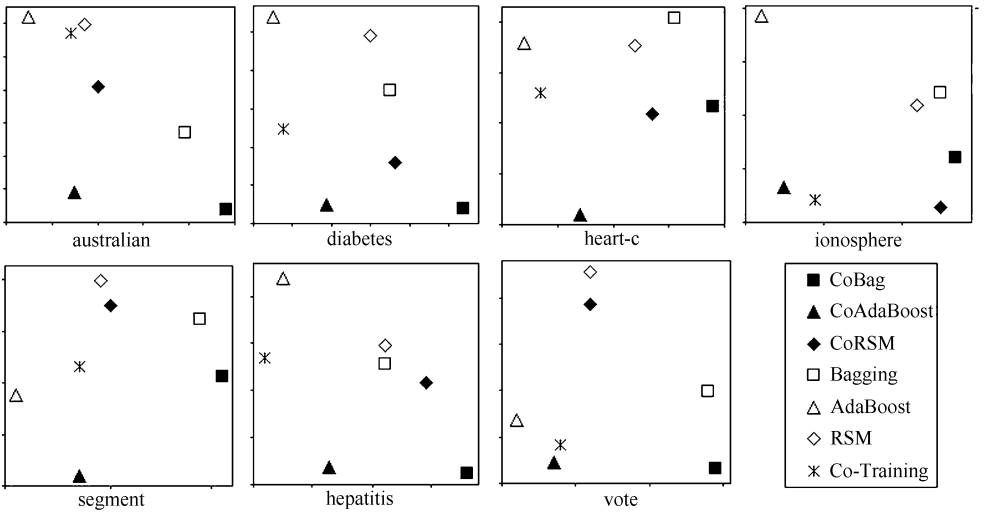

Figure 3. Kappa versus ensemble error for seven data sets with committees of size 10 and under 20% labeling rate (x-axis = average of $\kappa_{i,j}$, y-axis = *ensemble error)*

Table 10 shows the test set average errors of the ensemble members and error redution of ensembles constructed by *Co-Training* and different *CoBC* variants. From this table the following can be observed:

- The committee members constructed by *CoBag* achieves the lowest average error. This comes to show why *CoBag* outperforms *Co-Training*.
- The committee constructed by *CoAdaBoost* achieves the highest *ensemble error reduction*. This shows why *CoAdaBoost* outperforms *Co-Training* although the average error of committee members constructed by *CoAdaBoost* is higher than that of *Co-Training*.

8 Application to Visual Object Recognition

The recognition of 3-D objects from 2-D camera images is one of the most important goals in computer vision. *CoBC* will be applied to two 3-D object and one 2-D object recognition tasks.

8.1 Coil-20 Dataset

This benchmark dataset is a set of object images obtained from Columbia Object Image Library[32] (see Fig.4). The dataset contains the images of 20 different objects, for each object 72 training samples are available. Colour histogram is extracted from each image and then it is used as input feature vector for classification (V_1). Then, each image was divided into 2×2 overlapped sub-images. An orientation histogram utilising Sobel edge detection is extracted from each sub-image. The four histograms are concatenated together to form another feature vector (V_2).

Figure 4. COIL 20

8.2 Fruits dataset

The fruits images data set, collected at the University of Ulm, consists of seven different objects with 120 images per object[32] (see Fig.5). Colour histogram is extracted from each image and then it is used as input feature vector for classification (V_1). Then, each image was divided into 2×2 overlapped sub-images. An orientation histogram utilising Sobel edge detection is extracted from each sub-image. The 4 histograms are concatenated together to form another feature vector (V_2).

8.3 Handwritten digits dataset

The handwritten STATLOG digits data set[32] consists of 10,000 images (1,000 images per class) and each image is represented by a 16×16 matrix containing the 8-bit grey values of each pixel (see Fig.6). In our study we used only 200 images per class to save computation time. Each sample is represented by a 40-dim vector results from reshaping the 16×16 image matrix into 256-dim vector then performing dimensionality reduction using PCA and projecting the 256-dim vector onto the top

Table 10 Test average error and error reduction percentage of the initial and best models constructed by *Co-Training* and different *C*
20% labeling rate and committees of size 10

SSL Method	Co-Training-20%				CoBag-20%-10				CoAdaBoost-20%-10				CoF	
Dataset	*initial*		*best*		*initial*		*best*		*initial*		*best*		*initial*	
	avg.	reduc.	avg.	reduc.	avg.	reduc.	avg.	reduc.	avg.	reduc.	avg.	reduc.	avg.	redu
australian	22.43	3.70	22.17	17.46	**19.25**	20.47	**17.05**	23.81	28.35	**33.72**	24.75	**45.41**	25.46	27.0
bupa	42.99	-0.33	43.95	6.76	**41.95**	8.68	**39.95**	11.79	43.83	**15.24**	41.39	**20.17**	42.80	6.6
colic	25.39	24.42	24.22	30.18	**20.80**	13.99	**18.16**	20.98	36.34	**40.62**	31.34	**49.01**	23.97	29.1
diabetes	32.58	8.07	32.07	17.40	**31.84**	13.69	**29.41**	17.04	37.47	**21.54**	33.95	**27.89**	32.45	10.9
heart-c	27.27	11.66	26.34	20.54	27.19	13.79	**25.41**	19.44	31.10	**27.43**	27.85	**39.53**	**26.81**	16.0
hepatitis	**21.88**	2.15	22.45	10.96	23.28	14.65	**22.16**	20.80	28.25	**23.26**	26.29	**32.75**	22.48	9.9
hypothyroid	4.56	-24.56	4.62	-14.29	**1.28**	27.34	**1.21**	29.75	4.67	**75.80**	4.61	**80.26**	3.42	56.7
ionosphere	17.41	**39.12**	16.79	47.35	16.81	23.74	15.85	34.20	24.59	36.03	20.30	**54.24**	**16.71**	25.9
segment	13.99	54.82	13.41	48.99	**9.85**	20.30	**8.89**	25.42	16.32	**61.76**	13.40	**66.34**	14.50	40.
sick	4.35	-22.99	4.40	9.77	**2.37**	13.50	**2.19**	19.18	6.73	**67.31**	6.47	**72.64**	4.33	-33.
splice	23.80	42.02	24.27	40.30	**15.50**	28.26	**14.29**	31.21	24.35	**58.36**	24.83	**66.81**	20.14	37.4
tic-tac-toe	32.39	2.35	31.92	12.12	**30.30**	15.84	**29.44**	20.14	34.70	**39.77**	32.88	**48.11**	32.62	9.5
vehicle	**39.34**	29.13	38.21	**35.28**	39.70	18.46	**36.95**	19.57	44.15	**29.94**	40.89	31.69	39.81	22.0
vote	9.07	34.51	9.11	45.99	**6.16**	1.30	**5.53**	20.07	12.85	**57.43**	11.61	**60.72**	10.99	21.1
wdbc	21.77	20.75	8.17	42.72	9.33	24.22	8.61	39.95	11.68	**39.47**	11.41	**51.88**	**8.64**	11.0
ave.	22.61	16.16	21.47	23.81	19.71	16.08	18.34	21.00	25.69	34.87	23.46	42.81	21.68	18.0
no. of wins	2	1	0	1	**10**	0	**13**	0	0	**14**	1	**14**	3	0

40 principal components (V_1). Then, each image was divided into 2×2 overlapped sub-images. An orientation histogram is extracted from each sub-image. The 4 histograms are concatenated together to form another feature vector (V_2).

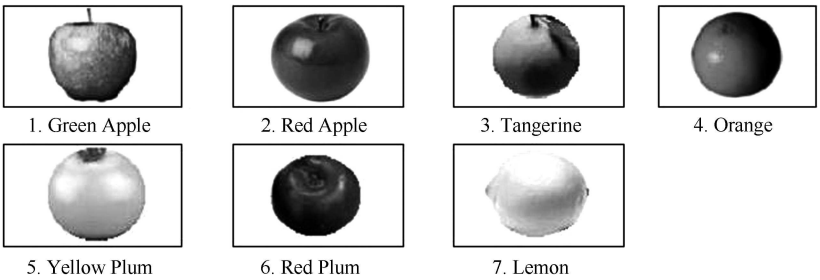

Figure 5. A sample of the images in the fruits dataset

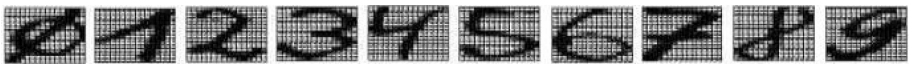

Figure 6. A sample of the handwritten digits

8.4 Experimental setup

The *CoBC* variants are evaluated on the three image classification tasks. *J*48 decision tree and k nearest neighbor classifier are used as base learners. To estimate the error, 5 runs of 4-fold cross validation have been performed. That is, for each data set about 25% of the data samples are randomly chosen as test set (360, 210 and 500). Then, the remaining 75% (1080, 630 and 1500 for Coil-20, fruits and handwritten digits, respectively) partitioned into 20% (216, 126 and 300, respectively) as initial labeled training set L and the remainder is the unlabeled data U to be exploited to boost the performance.

The performance of *CoBC* variants is compared with *Self-Training*, *Single-View Co-Training* and *Two-View Co-Training*. Note that these data sets are with two sufficient and redundant views. Therefore, *Two-View Co-Training* is applied and its performance as a *multi-view SSL* method is used for comparison to the other *single-view SSL* methods. For *Single-View Co-Training*, we randomly divide the original feature set of each data set into two disjoint subsets with equal sizes and train a classifier based on each subset as a separate view. The final output results from averaging the class probability estimates of these two co-trained classifiers. *Self-Training* algorithm uses a single classifier based on the original feature set.

For fair comparison, we set the maximum number of iterations for all SSL methods to 30 but the methods can terminate earlier if all the unlabeled examples are labeled. In addition, we set $poolsize = 100$ and $n_k = 3$ for all SSL methods except for *Two-View Co-Training* and *Single-View Co-Training* where $n_k = 1$.

8.5 Results

Table 11 presents the performance of the supervised base learner and ensemble learners when trained on the full training set ($L \cup U$). This represents the lower bound of the test error. Note that the bullet/circle mark indicates that the corresponding classifier significantly better/worse than the base classifier.

Table 11 Test error rates of the supervised learners, using 100% labeling rate and committees of size 10

(a) Base Learner: Pruned *J*48 decision tree without Laplace Correction

Data set	*J*48	*Bagging*-10	*AdaBoost*-10	*RSM*-10
COIL20	11.62(1.48)	9.06(1.26)$^\bullet$	5.80(1.25)$^\bullet$	9.60(1.46)$^\bullet$
Fruits	13.01(2.45)	8.67(2.18)$^\bullet$	6.43(1.56)$^\bullet$	6.51(1.55)$^\bullet$
Digits	23.28(2.45)	16.09(2.42)$^\bullet$	11.43(1.77)$^\bullet$	14.51(1.16)$^\bullet$
ave.	15.97	11.27	7.89	10.20

(b) Base Learner: Fuzzy 5-*NN* with weighting = 1 / distance

Data set	Fuzzy 5-*NN*	*Bagging*-10	*AdaBoost*-10	*RSM*-10
COIL20	8.80(1.28)	9.00(1.16)	8.80(1.28)	8.20(1.12)
Fruits	6.34(1.79)	6.03(1.89)	6.34(1.79)	5.34(1.52)
Digits	6.48(0.71)	6.64(0.97)	6.48(0.71)	6.38(0.98)
ave.	7.20	7.22	7.20	6.64

The average test errors and standard deviations are shown in Table 12, where *initial* presents the test error rate, at iteration 0, using only the available labeled data L, *best* denotes the minimum error rate achieved during the SSL iterations of exploiting the unlabeled data, and *improv* denotes the relative improvement, that is $improv = \frac{initial-best}{initial}$. Table 12 shows that on the three image classification tasks, different $CoBC$ variants can effectively exploit the unlabeled data to improve the learning performance. For *CoAdaBoost*, when *J*48 decision tree is used as base learner, the improvement of the best model constructed by *AdaBoost* is 16.37%, 20.55% and 11.48%, respectively. While when 5-*NN* classifier is used as base learner, the improvement is 8.03%, 14.08% and 34.34%. For *CoBag*, when *J*48 decision tree classifiers are used, the improvement of the best model constructed by *Bagging* is 3.93%, 11.32% and 1.60%, respectively. While when 5-*NN* classifiers are used, the improvement is 8.03%, 15.26% and 35.31%.

Table 12 also shows that the improvement of $CoBC$ variants in most of the cases is higher than that of *Self-Training*, *Single-View Co-Training* and *Two-View Co-Training*. Through observing Table 12(a), one can find that the *initial* ensembles of 10 (unstable) *J*48 decision trees constructed by *Bagging*, *AdaBoost* and the *RSM* perform better than the *initial* single *J*48 decision tree before SSL process. As a consequence, the *best* ensembles of trees constructed by *Bagging*, *AdaBoost* and the *RSM* perform better than the *best* single tree after exploiting the unlabeled data which emphasizes Hypothesis 2. Secondly, for the three $CoBC$ variants, it is clear that the *best* ensembles peform better than the *initial* ensembles which supports Hypothesis 1. Thirdly, $CoBC$ variants can outperform *Two-View Co-Training* evenif the former depends only on a single view (V_1) while the latter depends on two views (V_1 and V_2). In addition, $CoBC$ variants can outperform *Single-View Co-Training* which emphasizes Hypothesis 3.

By observing Table 12(b) where (stable) 5-nearest neighbors (*NN*) classifier is used, one can find that the ensembles of 10 (stable) *NN* classifiers constructed by *Bagging*, *AdaBoost* and *RSM* failed to outperform a single *NN* classifier. Bay[33] gave the reasons of the failure of *AdaBoost*: (1) *AdaBoost* stops when a classifier

Table 12 The performance of *Self-Training*, *Co-Training* and different *CoBC* variants on image classification problems, using 20% committees of size 10

(a) Base Learner: Pruned $J48$ decision tree without Laplace Correction

Method	Self-Training-20% (V_1)			Single-View Co-Training-20% (V_1)			Two-View Co-Training
Dataset	*initial* ($J48$)	*best*	*improv*	*initial*	*best*	*improv*	*initial*	*be*
COIL20	23.66(3.38)	22.60(2.72)	4.48%	24.67(3.76)	22.87(2.54)	7.30%	24.07(3.33)	21.38(
Fruits	25.15(3.79)	24.58(3.67)	2.27%	25.84(3.21)	21.79(2.84)	15.67%	29.86(4.38)	25.17(
Digits	35.19(2.42)	33.23(2.07)	5.57%	45.84(3.78)°	43.28(2.42)°	5.60%	38.23(2.41)	35.44(
ave.	28.00	26.80	4.28%	32.12	29.31	8.75%	30.72	27.
Method	*CoBag*-20%-10 (V_1)			*CoAdaBoost*-20%-10 (V_1)			*CoRSM*-20%	
Dataset	*initial* (*Bagging*)	*best*	*improv*	*initial* (*AdaBoost*)	*best*	*improv*	*initial* (*RSM*)	*be*
COIL20	19.07(2.57)•	18.32(2.31)•	3.93%	**16.43(2.38)**•	**13.74(2.09)**•	**16.37%**	21.69(3.07)•	20.81(
Fruits	22.69(3.48)	20.12(3.56)	11.32%	**19.70(2.63)**•	**15.65(2.67)**•	**20.55%**	18.96(3.08)•	16.81(
Digits	24.34(2.19)•	23.95(2.77)•	1.60%	**21.43(2.59)**•	**18.97(2.76)**•	11.48%	25.10(2.50)•	22.15(
ave.	22.04	20.80	5.62%	19.19	16.12	**16.00%**	21.91	19.

(b) Base Learner: Fuzzy 5-NN with weighting = 1 / distance

Method	Self-Training-20% (V_1)			Single-View Co-Training-20% (V_1)			Two-View Co-Train	
Dataset	*initial* (5-NN)	*best*	*improv*	*initial*	*best*	*improv*	*initial*	
COIL20	22.39(1.36)	20.41(1.36)•	8.84%	21.45(1.32)	19.23(1.11)•	10.35%	10.92(1.68)	9.
Fruits	16.05(3.49)	13.60(2.55)	15.26%	15.43(2.74)	13.93(2.15)	9.72%	7.53(1.74)	6.
Digits	14.27(2.09)	9.23(1.17)•	35.31%	17.38(1.81)°	14.56(1.80)	16.22%	11.48(1.67)	8.
ave.	17.57	14.41	**17.98%**	18.09	15.91	12.05%	**9.98**	
Method	*CoBag*-20%-10 (V_1)			*CoAdaBoost*-20%-10 (V_1)			*CoRSM*-2	
Dataset	*initial* (*Bagging*)	*best*	*improv*	*initial* (*AdaBoost*)	*best*	*improv*	*initial* (*RSM*)	
COIL20	23.21(1.73)	20.50(1.72)•	11.67%	22.39(1.36)	20.59(1.48)•	8.03%	20.93(1.59)•	19.
Fruits	16.72(3.28)	13.43(2.02)	**19.67%**	16.05(3.49)	13.79(2.82)	14.08%	15.41(3.60)	13
Digits	14.52(2.31)	9.17(1.48)•	**36.84%**	14.27(2.09)	9.34(1.08)•	34.34%	14.81(1.93)	10.

has zero training error, and this always happens for the NN classifier. (2) *AdaBoost* increases the weights of hard-to-classify example does not help in its classification because it depends on the weights of its neighbors not on its weight. Alpaydin[34] attributed the failure of *Bagging* with *NN* classifier to the fact that *Bagging* aims to reduce the variance of its underlying classifier while the variance of the stable *NN* classifier is already low so it is difficult to reduce it more. Investigating the best ensemble method to construct an effective ensemble of *NN* classifiers is beyond the scope of this study but this shows the need for a generalized ensemble-based SSL framework that can be applied on any ensemble method.

9 Conclusions

A new framework for semi-supervised learning with committees was introduced for application domains in which there are not redundant and independent views of the data ($CoBC$). Based on our experiments on 15 UCI data sets and three visual object recognition tasks, we have the following conclusions:

- Empirical results demonstrate that an ensemble of accurate and diverse classifiers can be used effectively to label unlabeled data.
- $CoBC$ can mitigate the degradation of a model performance when the training set size is small. That is, $CoBC$ outperform *Bagging*, *AdaBoost* and *RSM* when the available training data is limited.
- *CoBag* is able to minimize the average error of the underlying committee members and *CoAdaBoost* is able to maximize the *ensemble error reduction* without strongly degrading the diversity. Therefore, $CoBC$ can outperform *Self-Training*.
- The random feature splitting to apply *Co-Training* can result in base classifiers that are weaker than the single classifier used in *Self-Training*. Therefore, *Self-Training* can outperform *Co-Training with random split*.
- Kappa-error diagram is a useful tool to visualize the influence of SSL process on the diversity-accuracy relationship for ensembles.
- Interestingly *CoAdaBoost* achieves a performance comparable to *CoBag* and outperforms *CoRSM* although *AdaBoost* is sensitive to the scarce of training data and to the *mislabeling noise*.
- For some data sets, *supervised ensemble learners* outperform *semi-supervised learners Self-Training* and *Co-Training*, that exploit the unlabeled data, and the number of these data sets increases as the ensemble size increases. This emphasizes the hypothesis in Ref.[17] that ensemble learning can be a good solution to improve the performance of a single classifier when the amount of labeled data is limited even if the unlabeled data is not used.
- $CoBC$ is a generalized framework that can maintain the diversity of an ensemble constructed by any ensemble learner during the SSL process. *Co-Forest* places more constraints on the used ensemble learner and can hurt the diversity of the underlying ensemble[1].

- There is no SSL method that is the best for all real-world data sets. Because labeled data is scarce and there is no guarantee that unlabeled data will always help, each SSL method has strong model assumptions. One should use the method whose assumptions match the given problem structure. For instance, the following checklist was proposed in Ref.[12]: If the classes produce well clustered data, then EM with generative mixture models may be a good choice; If the features are naturally divided into two or more sets of features, then *Co-Training* and *Co-EM* may be appropriate; If SVM is already used, then *Transductive SVM* is a natural extension; If the existing supervised classifier is complicated and hard to modify, then *Self-Training* is a practical wrapper method.

- We can add the following statement to the above mentioned checklist: If the features are not naturally divided into more than one set and ensemble learning can be used, then *Co-Forest* is a good choice if the used ensemble learner does not depend on the training set to enforce the diversity and $CoBC$ may be more practical for application with any ensemble learner.

10 Future Work

There are many interesting directions for future work. Firstly, we plan to perform additional empirical studies using real data from several application areas without two views. Also as $CoBC$ is general, we plan to evaluate $CoBC$ using other ensemble learners and other base learners such as MLP, *Naive Bayes* and RBF Networks.

Secondly, since active learning is another direction for learning from unlabeled data where the labels of some selected unlabeled examples (*most informative examples*) are asked for labelling by the user. Currently, we are investigating the interleaving of $CoBC$ and QBC for a more robust committee-based learning (*ActiveCoBC*).

Thirdly, we found that the learning curve of $CoBC$, like *Co-Training*, is non-monotonic. Ideally, we would like it to have a more stable behavior. We investigate the selection of *most confident examples* based on all the previous ensembles rather than just the previous one. This change can leads to *smoothing* the learning curve and to increase the robustness to mislabeling noise (*SmoothedCoBC*).

Finally, in the current $CoBC$ implementation, when new data becomes available at each iteration, the previous committee is discarded and a new one is trained from scratch with the new training set. This leads to the non-monotonic behavior of the learning curve. We claim that $CoBC$ will be more stable and more efficient if incremental versions of ensemble learners and base learners are integrated (*OnlineCoBC*).

References

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

[2] Nigam K, McCallum AK, Thrun S, Mitchell T. Text classification from labeled and unlabeled documents using em. Machine Learning, 2000, 39(2-3): 103–134.

[3] Kiritchenko S, Matwin S. Email classification with co-training. In: Proc. of the 2001 Conference of the Centre for Advanced Studies on Collaborative Research (CASCON'01). IBM Press, 2001. 8–19.

[4] Freund Y, Seung H, Shamir E, Tishby N. Selective sampling using the query by committee algorithm. Machine Learning, 1997, 28(2-3): 133–168.
[5] Nigam K, Ghani R. Analyzing the effectiveness and applicability of co-training. In: Proc. of the 9th Int. Conf. on Information and Knowledge Management. New York: ACM, 2000. 86–93.
[6] Blum A, Mitchell T. Combining labeled and unlabeled data with co-training. In: Proc. of the 11th Annual Conf. on Computational Learning Theory (COLT 1998).Morgan Kaufmann, 1998, pp. 92–100. http://citeseer.ist.psu.edu/article/blum98combining.html
[7] Zhou Y, Goldman S. Democratic co-learning. In: Proc. of the 16th IEEE Int. Conf. on Tools with Artificial Intelligence (ICTAI'04). Washington, DC: IEEE Computer Society, 2004. 594–202.
[8] Zhou ZH, Li M. Tri-Training: Exploiting unlabeled data using three classifiers. IEEE Trans. on Knowledge and Data Engineering, 2005, 17(11): 1529–1541.
[9] Zhou ZH, Li M. Semi-Supervised regression with co-training. In: Proc. of the 19th International Joint Conference on Artificial Intelligence. Edinburgh, Scotland, 2005. 908–13.
[10] Wagstaff SRK, Cardie C, Schroedl S. Constrained k-means clustering with background knowledge. In: Proc. of the 18th Int. Conference on Machine Learning (ICML01). Williamstown, MA, 2001. 577–84.
[11] Zhou ZH, Zhang D, Chen S. Semi-Supervised dimensionality reduction. In: Proc. of the 7th SIAM International Conference on Data Mining (SDM'07). Minneapolis, MN, 2007. 629–634.
[12] Zhu X. Semi-Supervised learning literature survey. Department of Computer Sciences, University of Wisconsin at Madison, Madison, WI, Tech. Rep. 1530, 2006. http://www.cs.wisc.edu/jerryzhu/pub/sslsurvey.pdf
[13] Breiman L. Bagging predictors. Machine Learning, 1996, 24(2): 123–140.
[14] Freund Y, Schapire RE. Experiments with a new boosting algorithm. In: Proc. of 13th Int. Conf. on Machine Learning (ICML'97). San Francisco: Morgan Kaufmann, 1996. 148–156.
[15] Ho T. The random subspace method for constructing decision forests. IEEE Trans. Pattern Analysis and Machine Intelligence, 1998, 20(8): 832–844.
[16] Dietterich TG, Bakiri G. Error-Correcting output codes: a general method for improving multiclass inductive learning programs. In: Dean TL, McKeown K, eds. Proc. of the 9th AAAI National Conf. on Artificial Intelligence. AAAI Press, CA, 1991. 572–577.
[17] Dietterich TG. Ensemble Methods in Machine Learning. In: Proc. of the First Int. Workshop on Multiple Classifier Systems (MCS 2000). Cagliari, Italy, LNCS 1857, 2000.
[18] Blake C, Merz C. UCI repository of machine learning databases. 1998. http://archive.ics.uci.edu/ml
[19] Dempster A, Laird N, Rubin D. Maximum likelihood from incomplete data via the em algorithm. Journal of the Royal Statistical Society. Series B (Methodological), 1977, 39(1): 1–38.
[20] Hady MA, Schwenker F, Palm G. Semi-Supervised learning of tree-structured rbf networks using co-training. In: Proc. of the 18th Int. Conf. on Artificial Neural Networks (ICANN'08). Prague, Czech Republic, LNCS 5163, Springer-Verlag, 2008. 79–88.
[21] Levin A, Viola P, Freund Y. Unsupervised improvement of visual detectors using co-training. In: Proc. of the Intl. Conf. on Computer Vision. 2003.
[22] Feger F, Koprinska I. Co-Training using rbf nets and different feature splits. In: Proc. of the Int. Joint Conf. on Neural Networks (IJCNN'06). 2006. 1878–1885.
[23] Goldman S, Zhou Y. Enhancing supervised learning with unlabeled data. In: Proc. of the 17th Int. Conf. Machine Learning (ICML'00). 2000. 327–334.
[24] Breiman L. Random forests. Machine Learning, 2001, 45(1): 5–32.
[25] Brown G, Wyatt J, Harris R, Yao X. Diversity creation methods: a survey and categorisation. Information Fusion, 2005, 6(1): 5–20.
[26] Kuncheva LI, Whitaker CJ. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. Machine Learning, 2003, 51(2): 181-207.
[27] Dietterich TG. An experimental comparison of three methods for constructing ensembles of decision trees. In: Bagging, Boosting, and Randomization. Machine Learning, 2000. 139–157.
[28] Quinlan R. C4.5: Programs for Machine Learning. Morgan Kaufmann, 1993.
[29] Witten IH, Frank E. Data Mining: Practical Machine Learning Tools and Techniques with Java

Implementations. Morgan Kaufmann, October 1999.

[30] Nadeau C, Bengio Y. Inference for the Generalization Error. Machine Learning,2003, 62: 239–281.

[31] Margineantu DD, Dietterich TG. Pruning adaptive boosting. In: Proc. of the 14th Int. Conference on Machine Learning (ICML'97). Morgan Kaufmann, 1997. 211–218.

[32] Fay R. Feature selection and information fusion in hierarchical neural networks for iterative 3d-object recognition [Ph.D. dissertation]. Ulm University, 2007. http://vts.uni-ulm.de/doc.asp?id=6044

[33] Bay S. Nearest neighbor classification from multiple feature sets. Intelligent Data Analysis, 1998, 3: 191–209.

[34] Alpaydin E. Voting over multiple condensed nearest neighbors. Artificial Intelligence Review, 1997, 11: 115–132.