The unsymmetrical-style co-training

Wang, Bin; Zhang, Harry; Spencer, Bruce; Guo, Yuanyuan

Publisher's version / Version de l'éditeur:
https://doi.org/10.1007/978-3-642-20841-6_9
Advances in Knowledge Discovery and Data Mining, Lecture Notes in Computer Science; Volume 6634, pp. 100-111, 2011-05

The Unsymmetrical-Style Co-training

Bin Wang[1], Harry Zhang[1], Bruce Spencer[2], and Yuanyuan Guo[1]

[1] Faculty of Computer Science, University of New Brunswick
P.O. Box 4400, Fredericton, NB, Canada E3B 5A3
[2] National Research Council of Canada,
Fredericton, NB, Canada E3B 9W4
bin.wang@unb.ca

Abstract. Semi-supervised learning has attracted much attention over the past decade because it provides the advantage of combining unlabeled data with labeled data to improve the learning capability of models. Co-training is a representative paradigm of semi-supervised learning methods. Typically, some co-training style algorithms, such as *co-training* and *co-EM*, learn two classifiers based on two views of the instance space. But they have to satisfy the assumptions that these two views are sufficient and conditionally independent given the class labels. Other co-training style algorithms, such as *multiple-learner*, use two different underlying classifiers based on only a single view of the instance space. However, they could not utilize the labeled data effectively, and suffer from the early convergence. After analyzing various co-training style algorithms, we have found that all of these algorithms have symmetrical framework structures that are related to their constraints. In this paper, we propose a novel unsymmetrical-style method, which we call the *unsymmetrical co-training algorithm*. The unsymmetrical co-training algorithm combines the advantages of other co-training style algorithms and overcomes their disadvantages. Within our unsymmetrical structure, we apply two unsymmetrical classifiers, namely, the self-training classifier and the EM classifier, and then train these two classifiers in an unsymmetrical way. The unsymmetrical co-training algorithm not only avoids the constraint of the conditional independence assumption, but also overcomes the flaws of the early convergence and the ineffective utilization of labeled data. We conduct experiments to compare the performances of these co-training style algorithms. From the experimental results, we can see that the unsymmetrical co-training algorithm outperforms other co-training algorithms.

1 Introduction

Over the course of the past decade, researchers have developed various types of semi-supervised learning methods. *Co-training* [1] is a representative paradigm of semi-supervised learning methods that are based on the multiple representations from difference views. Co-training was inspired by the observation discovered in the Web pages classification [1], in which a Web page has two different representations (views): the words occurring on the page itself; and the words

contained in the anchor text of hyperlinks pointing to the page. The initial form of co-training is to train two classifiers separately on two sufficient and redundant views of data, and let these two classifiers label some unlabeled instances for each other. Like other semi-supervised learning methods, co-training requires its own assumptions to guarantee its success. Blum and Mitchell [1] proved that co-training can be successful if the two sufficient and redundant views are conditionally independent given the class label. Many researchers have supported the observation that co-training is sensitive to this theoretical assumption [18] [2]. However, the sufficient and redundant views are rarely found in most real-world application scenarios.

In order to tease out the effect of view-splitting from the effect of labeling, Nigam and Ghani [2] proposed a hybrid algorithm of expectation-maximization (EM) and co-training, called *co-EM*. Like co-training, co-EM tries to divide the instance space into two conditional independent views, and to train two EM classifiers based on these two views, respectively. But unlike co-training, co-EM uses all the unlabeled data every time, instead of incrementally selecting some confident predictions to update the training set. Nigam and Ghani [2] also provided a method for ideally splitting the view of instance space based on the conditional mutual information criteria between two subsets of attributes. However, this method is NP-hard and difficult to apply in practice.

Since both co-training and co-EM suffer from the conditional independence assumption, variants of co-training have been developed based on only a single view (without splitting the attribute set). For example, Goldman and Zhou [3] used two different learning algorithms in the paradigm of co-training without splitting the attribute set. Steedman et al. [4] developed a similar co-training algorithm that applies two diverse statistical parsers. Wang and Zhou [5] proved that if the two classifiers are largely diverse, co-training style algorithms are able to succeed. Because these variants substitute multiple views by multiple classifiers, these algorithms are referred to as *multiple-learner* algorithms. Since the multiple-learner algorithms are trained on the same attribute set, it is important to keep the two classifiers different during the process in order to prevent early convergence. Maintaining separated training sets is one approach for this purpose. However, assigning labeled instances to two different initial training sets will cause the ineffective utilization of labeled data sets in a semi-supervised learning scenario.

Considering the framework structures of co-training, co-EM, and multiple-learner algorithms, we can see that each structure is symmetrical. The co-training algorithm splits the instance space into two symmetrical views, trains two classifiers symmetrically, and lets two classifiers teach each other in a symmetrical way. Similarly, the co-EM algorithm sets up two symmetrical EM classifiers based on their related views. And likewise, the multiple-learner algorithm also has a symmetrical structure, where two classifiers are trained in parallel and combined together to score the unlabeled instances. Therefore, we define these algorithms as the *symmetrical-style co-training algorithms*.

In this paper, we propose an *unsymmetrical co-training algorithm* - a novel, semi-supervised, unsymmetrical-style algorithm. The unsymmetrical co-training algorithm combines the advantages of other co-training style algorithms and overcomes their disadvantages. The unsymmetrical co-training algorithm combines two unsymmetrical classifiers, namely, an EM classifier and a self-training classifier. In the algorithm, the self-training classifier takes the responsibility for a section of unlabeled instances in a data pool; and the EM classifier maintains a global view over the entire instance set. Without the two-view splitting, the unsymmetrical co-training algorithm uses the full set of attributes so that it can avoid the intractable constraint of the conditional independence assumption. Although both classifiers are initially trained by labeled instances, the EM classifier and the self-training classifier have different training sets once they enter the iteration procedure. The unsymmetrical co-training algorithm does not need to hold different initial training sets and is therefore able to utilize the labeled instances more effectively. Furthermore, according to the study of Wang and Zhou [5], the multiple-learner algorithm could not further improve the performance after a number of learning rounds because the difference between the two learners become smaller and smaller. Since the unsymmetrical co-training algorithm uses two unsymmetrical classifiers in an unsymmetrical structure, it does not need to worry about the difference between these two classifiers fading too quickly. We conduct the experiments to compare the performances of co-training, co-EM, multiple-learner, and unsymmetrical co-training algorithms on 30 data sets from Weka [6]. From the experimental results, we can see that the unsymmetrical co-training algorithm outperforms other algorithms.

The remainder of this paper is organized as follows. After introducing some preliminaries about co-training, co-EM, and multiple-learner algorithm in Section 2, we present our unsymmetrical co-training algorithm in Section 3. Then, the experiments to compare the performances of algorithms are reported in Section 4. Finally, we give our conclusion and look toward future work in Section 5.

2 Preliminaries

Suppose we have the instance space $X = X^1 \times X^2$, where X^1 and X^2 correspond to the two different views of the instance space, respectively, and the class label space Y. Given the data set $L \cup U$, we have a labeled data set $L = \{\langle (x_1^1, x_1^2), y_1 \rangle, \cdots, \langle (x_l^1, x_l^2), y_l \rangle\} \subset X \times Y$ and an unlabeled data set $U = \{(x_{l+1}^1, x_{l+1}^2), \cdots, (x_{l+u}^1, x_{l+u}^2)\} \subset X$. In addition, we have two classifiers h_1 and h_2, which are used to compose the following algorithms.

2.1 Co-training

Co-training [1] first tries to divide the instance space X into two different views X_1 and X_2, which are conditionally independent given the class label. Then, two classifiers h_1 and h_2 are trained based on these two different views, respectively.

Classifier h_1 classifies the unlabeled instances and "teaches" the other classifier h_2 the predicted class labels of unlabeled instances about which it feels most confident. These confident unlabeled instances are added into the training set of h_2 together with their predicted class labels. At the same time, classifier h_2 teaches h_1 the predicted class labels about which it feels more confident. After that, each classifier is retrained with the updated training set. Such a process can be repeated until a certain stopping condition is satisfied. The framework structure of co-training is shown in Figure 1. From Figure 1, we observe that the framework structure of co-training is symmetrical: the instance space is divided into two views symmetrically; the two classifiers are trained symmetrically; and they update each other's training set in a symmetrical way.

Some theoretical studies have analyzed why and how the co-training algorithm can succeed. With proposing the co-training algorithm, Blum and Mitchell [1] defined the co-training model in a PAC-style theoretical framework and proved that the two different views are supposed to satisfy the following conditions: (1) each view is sufficient and consistent to train a good classifier; (2) each view is conditionally independent to the other one given the class label. Dasgupta et al. [20] also provided a PAC-style theoretical analysis for co-training. Yu et al. [7] proposed a graphical model for the co-training algorithm based on the conditional independence assumption. Abney [8] showed that weak dependence can also guarantee co-training's working. Balcan et al. [9] proposed a much weaker "expansion" assumption on the underlying data distribution, and proved that it is sufficient for the iterative co-training to succeed. Wang and Zhou [10] analyzed the co-training algorithm as a combinative label propagation over two views, and provided the sufficient and necessary condition for co-training to succeed.

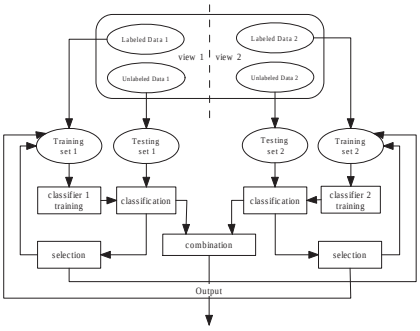

Fig. 1: The structure of co-training algorithm.

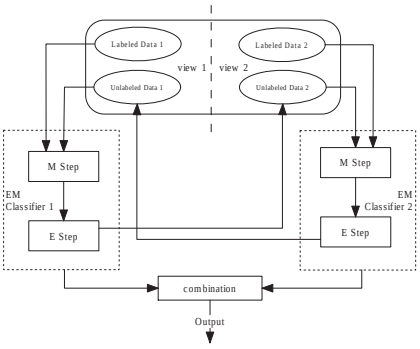

Fig. 2: The structure of co-EM algorithm.

2.2 Co-EM

Nigam and Ghani [2] compared the performances of the co-training algorithm with the EM algorithm, and studied how sensitive the co-training algorithm is to the conditional independence assumption. They the proposed a hybrid

algorithm of EM and co-training, called co-EM. The co-EM algorithm is similar to the EM algorithm, which is an iterative procedure that uses all the unlabeled instances every time, instead of incrementally selecting some unlabeled instances to update the training set. On the other hand, the co-EM algorithm is also like the co-training algorithm, which tries to divide the instance space into two conditionally independent views. Nigam and Ghani [2] argued that the co-EM algorithm is a closer match to the theoretical argument established by Blum and Mitchell [1].

In the co-EM algorithm, two EM classifiers, h_1 and h_2, are chosen to correspond to the two different views X_1 and X_2. Initially, classifier h_1 is trained only based on the labeled data set L with view X_1. Then, classifier h_1 probabilistically labels all the unlabeled instances in data set U. Next, classifier h_2 is trained using the labeled instances from the view X_2 of data set L, plus the unlabeled instances from the view X_2 of data set U with the class labels given by h_1. Classifier h_2 then relabels the instances for the retraining of h_1. This process iterates until the classifiers converge. The framework structure of co-EM is shown in Figure 2. From Figure 2, we can see the symmetrical structure of the co-EM algorithm.

2.3 Multiple-learner

Since the paradigmatic assumptions of co-training are difficult to satisfy in real-world application scenarios, many researchers begin to study the variants of co-training that do not require the two-view splitting [3] [4] [11]. Because those algorithms usually use multiple learners, they are referred to as the multiple-learner algorithms. Ng and Cardie [12] summarized multiple-learner algorithms and proposed their own algorithm[3]. In their multiple-learner algorithm, two different classifiers h_1 and h_2 are used and trained based on the single view of instance space. At each iteration, each classifier labels and scores all the instances in a data pool. Some instances with scores found to be high by classifier h_1 are added to the training set of classifier h_2 together with their predicted class labels from h_1, and vice verse. Then, the entire data pool is flushed and replenished using instances drawn from the unlabeled data set U after each iteration. The process is repeated until no further instances can be labeled. The framework structure of multiple-learner algorithm is shown in Figure 3. Here we can see that the structure of multiple-learner algorithm is also symmetrical, where two classifiers are trained in parallel and combined together to score the unlabeled instances in the data pool.

3 Unsymmetrical Co-training

As we have already emphasized, the co-training algorithm, the co-EM algorithm, and the multiple-learner algorithm all have symmetrical framework structures.

[3] The multiple-learner algorithm that appears in later sections refers to the version of Ng and Cardie [12]

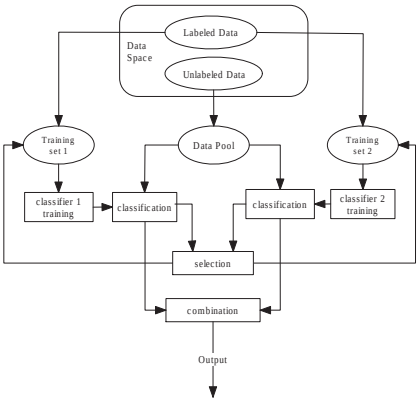

Fig. 3: The structure of multiple-learner algorithm.

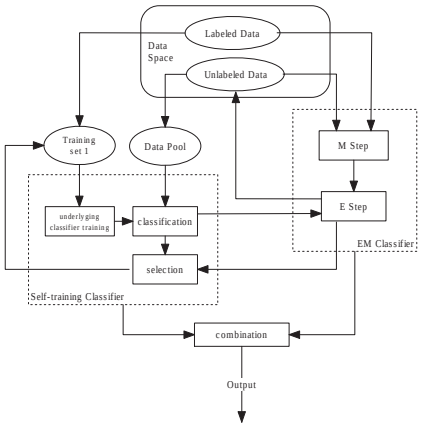

Fig. 4: The structure of unsymmetrical co-training algorithm.

Some of the constraints that restrict these algorithm are related to their symmetrical structures. For example, both the co-training algorithm and the co-EM algorithm are required to symmetrically divide the instance space into two sufficient and conditionally independent views, and their performances are quite sensitive to these assumptions. However, fulfilling these assumptions is a NP-hard problem, and these assumptions are intractable in practice. Although the multiple-learner algorithm does not suffer from the same intractable assumptions, it nevertheless still requires two different classifiers that are trained in a symmetrical way. Wang an Zhou [10] reported that if the two initial classifiers have large difference, they can be improved together using the multiple-learner algorithm. They also discovered that, as the multiple-learner algorithm proceeds, more and more unlabeled data are labeled, which makes the difference between the two learners become smaller and smaller. In other words, even though the two classifiers have big difference initially, they become more and more similar after several learning rounds since they are trained in a symmetrical way, and the performance of the multiple-learner algorithm cannot be further improved. Moreover, if the two selected classifiers are only slightly different from one another, two different training sets are required in order to avoid convergence of the algorithm at an early stage in the symmetrical structure. However, in the scenario of semi-supervised learning, assigning labeled instances to two different initial training sets will cause the ineffective utilization of labeled data set. To escape the constraints of symmetrical structures, we attempt to design a new algorithm that still performs with the style of co-training but has an unsymmetrical framework structure.

In this paper, we propose the unsymmetrical co-training algorithm. This algorithm uses two unsymmetrical classifiers, namely, the EM classifier and the self-training classifier. The EM classifier is a kind of generative model that has been successfully applied in semi-supervised learning [13]. The EM algorithm includes two steps: the E-step and the M-step. The E-step estimates the expec-

tations of the class information of unlabeled instances, and the M-step maximizes the likelihood of the model parameters using the expectations from the previous E-step. The EM classifier performs an iterative hill-climbing process to find a local maxima of model probability and then assigns the class labels in terms of the established model. Self-training might be the earliest technique for semi-supervised learning [14] [15] and is commonly applied in many domains [17] [16]. In the self-training classifier, an underlying classifier is iteratively trained and used to label the unlabeled instances, and then some unlabeled instances having the confident predictions are used to update the training set for the next round training of the underlying classifier.

Not only are two unsymmetrical classifiers applied in the unsymmetrical co-training algorithm, but these two classifiers are also applied within an unsymmetrical framework structure. In the structure, a data pool has been created for the self-training classifier by randomly selecting instances from the unlabeled instance set U. This data pool is the labeling objective on which the self-training classifier focuses in the process of the algorithm. But the EM classifier does not focus on a specific section of unlabeled instances. Instead, it faces the entire unlabeled instance set U during the algorithm. Moreover, the training sets used to train these two classifiers are different. For the self-training classifier, the training set consists of the labeled instances and the unlabeled instances from the data pool with their predicted class labels. For the EM classifier, the training set is the labeled instances plus the whole unlabeled instances together with the class labels assigned from the previous learning round. The framework structure of unsymmetrical co-training algorithm is shown in Figure 4.

The unsymmetrical co-training algorithm learns the two classifiers in an unsymmetrical way. Initially, both of the classifiers are trained based on the labeled instance set L with the single view (the full set of attributes). Then, the EM classifier labels all the unlabeled instances, and the self-training classifier predicts the labels of unlabeled instances in the data pool. The predicted class labels of unlabeled instances in the data pool will be used to substitute the class labels of corresponding unlabeled instances that have been assigned by the EM classifier. The EM classifier is then retrained by the updated training set and relabels the unlabeled instances. The unlabeled instances in the data pool, for which class labels from the self-training classifier are identical to the class labels from EM classifier, will be selected to update the training set of self-training classifier together with their predicted class labels. If there are not enough such unlabeled instances, the confidence degree metric will be used to select other unlabeled instances with high confidence in the data pool to update the training set together with the predicted class labels from the self-training classifier. Next, the data pool will be replenished by other unlabeled instances. The procedure is repeated until there are no further instances in the data pool. The predictions for new-coming instances are obtained using the combination of predictions from the EM classifier and the self-training classifier, just as the co-training algorithm does [1]. The formal description of unsymmetrical co-training algorithm is shown in Figure 5.

Input: Labeled instance set L, unlabeled instance set U, self-training classifier h_{self}, and EM classifier h_{EM}.
Initialization:

- Initialize the training set for h_{self} by L.
- Create the data pool for h_{self} by randomly selecting from U.
- Build h_{EM} by L, and use h_{EM} to label all the unlabeled instances in U.
- Create the training set for h_{EM} using all the labeled instances and unlabeled instances with predicted class labels from h_{EM}.

Loop if h_{self}'s data pool still has some instances

- Build h_{self} using its training set.
- Use h_{self} to label the instances in its data pool.
- Use the predicted labels in h_{self}'s data pool to replace the corresponding instances' class labels in h_{EM}'s training set.
- Build h_{EM} by the updated training set.
- Relabel all the unlabeled examples using h_{EM}, and return the predicted class labels of instances in the data pool to h_{self}.
- h_{self} selects some instances in the data pool if:
  1. their predicted labels are identical to the labels from h_{EM}; or
  2. they have higher scores ranked by the confidence degree metric of h_{self}.
- Add selected instances into the training set of h_{self} together with their predicted class labels.
- Replenish instances in the data pool using other unlabeled instances.

Output: The combination of predictions for new-coming instances from h_{self} and h_{EM}.

Fig. 5: The description of unsymmetrical co-training algorithm.

According to the theoretical study of Wang and Zhou [5], the co-training style algorithm is able to succeed if the two classifiers are vastly different. From the description above, we can see that the unsymmetrical co-training is consistent with their theory. The self-training classifier and the EM classifier not only display different characteristics on learning, but also are deployed differently in the unsymmetrical framework structure. Although they are both initially trained by the same labeled instances, these two classifiers have different training sets once they enter the iteration procedure. Therefore, the unsymmetrical co-training algorithm avoids the early convergence of both classifiers to the same hypothesis, and utilizes the labeled and unlabeled instances more effectively than does the multiple-learner algorithm. Moreover, unlike the multiple-learner algorithm, in which two classifiers become more and more similar as the algorithm proceeds, our algorithm always maintains the differences between the two classifiers due to its unsymmetrical way of learning. On the other hand, the unsymmetrical co-training algorithm uses the single view of instance space so that it avoids the intractable conditional independent view-splitting.

In the unsymmetrical co-training algorithm, the self-training classifier and the EM classifier can complement each other. The EM algorithm essentially uses the naive Bayes method to assign class membership probabilities to unlabeled instances. The EM classifier is expected to do well when it satisfies the conditional independence assumption of naive Bayes. However, these probabilities are always poorly estimated because the conditional independence assumption is violated. Self-training, on the other hand, uses the class membership probabilities for ranking the confidences of unlabeled instances instead of directly using them for the classification. Thus, the conditional independence assumption influences the self-training classifier more weakly than does the EM classifier. From another point of view, self-training is an incremental procedure and always suffers from the reinforcement of any misclassifications from previous updates. In the unsymmetrical co-training algorithm, the EM classifier is like a supervisor

beside the self-training classifier, which checks the predicted class labels made by self-training and provides opinions for these predictions in terms of its own knowledge. It is useful to reduce the chance of adding misclassifications to the next iteration in the procedure. Moreover, the self-training classifier restricts its view only on the labeled instances and the unlabeled instances in the data pool. Since the EM classifier is able to see the entire set of labeled and unlabeled instances, the view of the EM classifier is much broader than that of the self-training classifier. Therefore, the EM classifier might be seen to make predictions from a global view to help self-training, and the self-training classifier likewise boosts EM from its local view.

We summarize the differences of co-training, co-EM, multiple-learner, and unsymmetrical co-training algorithms from several points of view in Table 1. From Table 1, we can see that the unsymmetrical co-training algorithm not only combines the advantages of other algorithms, but also overcomes their disadvantages. In the next section, we design the experiments to compare the performances of these algorithms in the next section.

Table 1: The differences of co-training, co-EM, multiple-learner and unsymmetrical co-training algorithms.

	co-training	co-EM	mult-learner	unsym co-training
Structure	Symmetrical	Symmetrical	Symmetrical	Unsymmetrical
Split attribute set	Yes	Yes	No	No
Split training set	No	No	Yes	No
Learning style	Incremental	Iterative	Incremental	Incremental and iterative
Num of instances added per iteration	Fixed	Variable	Fixed	Variable
Use data pool	Yes	No	Yes	Yes
Example selection for two learners	Higher scored	N/A (no need to select)	Agreed and higher scored	Agreed and higher scored

4 Experiments

In this section, we design the experiments to compare the performances of co-training, co-EM, multiple-learner, and unsymmetrical co-training algorithms. The experiments are conducted on 30 data sets from Weka [6], which are selected from the UCI repository. There are some preprocessing stages adopted on each data set. First, we use the filter $ReplaceMissingValues$ in Weka to replace the missing values of attributes in each data set. Second, we use the filter $Discretize$ in Weka, which is the unsupervised ten-bin discretization, to discretize numeric attributes. Thus, all the attributes are nominal. Moreover, we notice that some attributes do not contribute any information for the purpose of prediction if the numbers of these attributes are almost equal to the numbers of instances in the corresponding data sets. The third preprocessing stage is to use the filter *Remove* in Weka to delete such attributes. We implement co-training, co-EM, multiple-learner and unsymmetrical co-training algorithms in the Weka framework. The underlying classifiers used by algorithms are naive Bayes classifiers, except the co-EM and a part of the unsymmetrical co-training algorithms that use the EM classifiers.

Our experiments are configured as follows. In each data set, 10% of the instances are used as testing instances; 10% of the remaining data set is used

as the set of labeled instances; and all other instances are used as unlabeled instances. For the co-training and co-EM algorithms based on two views, the attribute set is randomly divided into two disjointed subsets. For the co-training, co-EM, and unsymmetrical co-training algorithms that need the data pool, the size of the data pool is set to 10% of the unlabeled instance set. For the co-training and multiple-learner algorithms that fix the number of instances added per iteration, the number of added instances for each class label is decided by the class label distribution in the original labeled instance set: for the class label with the minimum percentage, the number is set to 1; and for all the other class labels, the numbers are set to the times of that the class label has the minimum percentage. The accuracy score is used to evaluate the performances of algorithms. In our experiments, the accuracy scores of each algorithm are obtained via 10 runs of ten-fold cross-validation and evaluated on the same testing sets. Finally, we conduct two-tailed t-test with a 95% confidence level to compare the unsymmetrical co-training algorithm to the other algorithms. The results are shown in Table 2.

In Table 2, the two-tailed t-test results are shown in the bottom row, where each entry has the format of $w/t/l$. This means that, comparing with the unsymmetrical co-training algorithm, the algorithm in the corresponding column wins w times, ties t times, and loses l times. From the experimental results, we observe that the unsymmetrical co-training algorithm outperforms other algorithms, where it wins 8 times and loses 2 times against the co-training and multiple-learner algorithms, and wins 10 times and never loses against the co-EM algorithm.

The evidences provided by the above experiments can be explained as follows. The unsymmetrical structure is more effective in the scenario of semi-supervised learning than are the symmetrical structures. The unsymmetrical co-training algorithm combines the EM classifier, which learns from a global view, and the self-training classifier, which boosts the learning from the local view. These two classifiers, which complement each other in the unsymmetrical structure, enhance the learning ability of the co-training style framework. The co-training and co-EM algorithms are sensitive to the conditional independence assumption. Randomly splitting the attribute sets decreases the overall performances of co-training and co-EM algorithms. Although the multiple-learner algorithm does not need two-view splitting, just as the unsymmetrical co-training algorithm does, the small number of labeled instances cannot be utilized effectively in the symmetrical structure. Besides, as the multiple-learner algorithm proceeds, its underlying classifiers become more and more similar so that the performance cannot be further improved.

5 Conclusion

In this paper, we propose the unsymmetrical co-training algorithm, which is a novel semi-supervised learning method in the co-training style. In the algorithm, two unsymmetrical classifiers, namely, the self-training classifier and the

Table 2: Experimental results on accuracy for co-training, co-EM, multiple-learner and unsymmetrical co-training algorithms.

Dataset	UnSymCoTrain	Co-training	Co-EM	Mult-Learner
zoo	85.36	79.48	76.24	80.19
labor	78.83	78.5	65.9	79.87
iris	89.2	89.33	79.47	90.4
vote	88.62	88	88.23	88.26
breast-cancer	72.25	65.94	71.63	70.97
lymph	74.21	55.1	64.65	57.63
primary-tumor	34.29	30.98	25.38	30.44
hepatitis	82.55	80.5	80.08	82.18
balance-scale	72.38	70.03	53.68	65.63
glass	45.43	42.27	42.42	43.53
audiology	31.46	34.27	26.55	35.16
heart-h	83.2	74.27	81.36	79.12
heart-c	81.99	67.27	82.75	74.53
colic.ORIG	58.45	59.02	54.88	57.38
heart-statlog	81.04	77.67	70.07	81.07
autos	46.79	45.58	41.22	45.67
credit-a	84.29	82.14	75.64	83.88
colic	72.2	74.17	67.37	73.54
breast-w	97.44	97.02	97.4	97.08
diabetes	71.66	71.79	69.37	70.75
anneal.ORIG	77.04	77.38	69.32	77.61
soybean	77.71	68.3	62.29	70.77
ionosphere	84.64	80.34	80.6	82.73
anneal	86.15	83.44	68.93	84.66
vowel	30.59	26.82	18	26.86
kr-vs-kp	65.55	74.27	51.63	72.82
credit-g	69.03	67.51	67.13	65.73
vehicle	50.33	47.26	43.91	46.34
sonar	65.23	56.55	55.7	56.82
mushroom	90.04	93.06	89.24	92.96
w/t/l		2/20/8	0/20/10	2/20/8

EM classifier, are learned in an unsymmetrical way within an unsymmetrical framework structure. Compared with other co-training style algorithms, such as co-training, co-EM, and multiple-learner, the unsymmetrical co-training algorithm has several advantages. First, the unsymmetrical co-training algorithm is based on the single view of instance space, so it does not suffer from the violation of conditional independence assumption as co-training and co-EM algorithms do. Second, the unsymmetrical structure makes the utilization of labeled instances more effective since there is no need to split the labeled instance set into two different initial training sets for two underlying classifiers. Moreover, the two unsymmetrical classifiers do not easily become more similar after several learning rounds because the unsymmetrical training of the algorithm prevents growing similarity. We conduct the experiments to compare the performances of these algorithms. The experimental results show that the unsymmetrical co-training algorithm overall outperforms other algorithms.

In the future, we will continue the study of semi-supervised learning methods in the co-training style, especially within the unsymmetrical framework structure. More experiments will be conducted to compare our unsymmetrical co-training algorithm with other co-training style methods under various circumstances. The different underlying classifiers will be tested within this unsymmetrical structure to see whether the performance can be improved further. It will also be interesting to apply the unsymmetrical co-training algorithm to real-world applications, especially for the applications suitable for semi-supervised learning, such as natural Language processing (NLP) and bioinformatics.

References

1. Blum A. and Mitchell T.: Combining labeled and unlabeled data with co-training. Proceedings of the 1998 Conference on Computational Learning Theory. pages

92–100, Kaufmann Morgan 1998.
2. Nigam K., Ghani R.: Analyzing the Effectiveness and Applicability of Co-training. CIKM. pages 86–93. 2000.
3. Goldman S.A., Zhou Y.: Enhancing Supervised Learning with Unlabeled Data. Proceedings of the Seventeenth International Conference on Machine Learning, pages 327–334, 2000.
4. Steedman M., Osborne M., Sarkar A., Clark S., Hwa R., Hockenmaier J., Ruhlen P., Baker S., Crim J.: Bootstrapping statistical parsers from small datasets. Proceedings of the EACL, pages 331–338, 2003.
5. Wang W., Zhou Z-H.: Analyzing Co-training Style Algorithms. ECML '07: Proceedings of the 18th European conference on Machine Learning, pages 454–465, 2007.
6. Witten Ian H., Frank E.: Data mining: practical machine learning tools and techniques with Java implementations. Morgan Kaufmann, 2002.
7. Yu S., Krishnapuram B., Rosales R., Steck H., Rao R.B.: Bayesian co-training. NIPS 20, pages 1665–1672, 2008.
8. Abney S.: Bootstrapping. ACL, pages 360–367, 2002.
9. Balcan M., Blum A., Yang K.: Co-Training and Expansion: Towards Bridging Theory and Practice. Advances in Neural Information Processing Systems 17, pages 89–96, 2004.
10. Wang W., Zhou Z.H.: A new analysis on co-training. Proceedings of the 27th International Conference on Machine Learning, 2010.
11. Zhou Z-H., Li M.: Semi-supervised regression with co-training. IJCAI'05: Proceedings of the 19th international joint conference on Artificial intelligence, pages 908–913, San Francisco, CA, USA, 2005.
12. Ng V., Cardie C.: Bootstrapping Coreference Classifiers with Multiple Machine Learning Algorithms. Proceedings of the 2003 conference on empirical methods in natural language processing (EMNLP), pages 113–120, 2003.
13. Nigam K., McCallum A.K., Thrun S., Mitchell T.: Text Classification from Labeled and Unlabeled Documents using EM, Machine Learning, vol. 39, pages 103–134, 2000.
14. Hearst M.: Noun homograph disambiguation using local context in large text corpora. University of Waterloo, pages 1–22, 1991.
15. Yarowsky D.: Unsupervised word sense disambiguation rivaling supervised methods. Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics, pages 189–196, 1995.
16. Wang B.: Semi-supervised Learning and Opinion-oriented Information Extraction. PhD dissertation, University of New Brunswick, 2010.
17. Wang B., Spencer B., Ling C.X., Zhang H.: Semi-supervised self-training for sentence subjectivity classifiation. Conference of Canadian AI 2008, pages 344–355, Springer-Verlag Berlin Heidelberg, 2008.
18. Muslea I., Minton S., Knoblock C.: Active + Semi-Supervised Learning = Robust Multi-View Learning. Proceedings of the 19th International Conference on Machine Learning (ICML 2002), pages 435–442, 2002.
19. Ganchev K., Graca J.V., Blitzer J., Taskar B.: Multi-view learning over structured and non-identical outputs. Proceedings of the Converence on Uncertainty in Artificial Intelligence (UAI), 2008.
20. Dasgupta S., Littman M. L., Mcallester D.: Pac generalization bounds for co-training. NIPS, 2001.