When Does Co-Training Work in Real Data?

Jun Du, Charles X. Ling, *Senior Member, IEEE,* and Zhi-Hua Zhou, *Senior Member, IEEE*

Abstract—Co-training, a paradigm of semi-supervised learning, is promised to alleviate effectively the shortage of labeled examples in supervised learning. The standard two-view co-training requires the dataset to be described by two views of features, and previous studies have shown that co-training works well if the two views satisfy the sufficiency and independence assumptions. In practice, however, these two assumptions are often not known or ensured (even when the two views are given). More commonly, most supervised datasets are described by one set of attributes (one view). Thus, they need be split into two views in order to apply the standard two-view co-training. In this paper, we first propose a novel approach to empirically verify the two assumptions of co-training given two views. Then, we design several methods to split single view datasets into two views, in order to make co-training work reliably well. Our empirical results show that, given a whole or a large labeled training set, our view verification and splitting methods are quite effective. Unfortunately, co-training is called for precisely when the labeled training set is small. However, given small labeled training sets, we show that the two co-training assumptions are difficult to verify, and view splitting is unreliable. Our conclusions for co-training's effectiveness are mixed. If two views are given, and known to satisfy the two assumptions, co-training works well. Otherwise, based on small labeled training sets, verifying the assumptions or splitting single view into two views are unreliable, thus it is uncertain whether the standard co-training would work or not.

Index Terms—semi-supervised learning, co-training, sufficiency assumption, independence assumption, view splitting, single-view.

1 Introduction

Traditional supervised classification learning builds classifiers based on labeled training examples. As labeled examples can be expensive to obtain, many semi-supervised approaches, such as the generative-based methods, the graph-based methods, and co-training, tend to utilize unlabeled examples to improve the predictive accuracy (see Section 6 for a review).

Co-training, a paradigm of semi-supervised learning, has drawn considerable attentions and interests recently (see, for example, (Chapelle et al., 2006; Zhu, 2006; Zhou & Li, in press) for review). The standard two-view co-training (Blum & Mitchell, 1998) assumes that the data can be described by two disjoint sets of features or views.[1] The standard co-training utilizes an initial (small) set of labeled training data and a (large) set of unlabeled data from the same distribution, and it works roughly as follows (Blum & Mitchell, 1998). Two classifiers are first trained on the initial labeled training set using the two views separately. Then, alternately, each classifier classifies the unlabeled data, chooses the few unlabeled examples whose labels it predicts most confidently, and adds those examples (with the predicted labels) to the training set. The classifiers are retrained, and the process repeats, until some stopping criterion is met. That is, the two classifiers "teach" each other with the additional examples whose labels are given by the other classifier, so as to improve the classification accuracy.

Two assumptions are proposed for co-training to work well (Blum & Mitchell, 1998). The first one assumes that the views are sufficient; that is, each view (thus also the combined view) is sufficient to predict the class perfectly. We call it the *sufficiency assumption*. The second assumption requires that the two views are conditionally independent; that is, the two views are independent given the class. We call it the *independence assumption*. Theoretical results have shown that if the sufficiency and independence assumptions are satisfied, co-training is guaranteed to work.[2] In addition, the two-view co-training has been applied quite successfully to many real-world tasks. See Section 6 for a quick review of successful applications of co-training.

However, the two assumptions that make co-training always work are usually not known or ensured in real-world applications. Given a dataset with two views, how can we judge if the two-view co-training would work well? How can we verify if the sufficiency and independence assumptions are satisfied? If the dataset has only one view as in most real-world situations, can the two-view co-training still work if we split the single view into two views?

This is an extension of the paper published in PAKDD'09 (Ling et al., 2009).

- *J. Du and C.X. Ling are with the Department of Computer Science, The University of Western Ontario, London, Ontario, Canada N6G 5B7. E-mail: {jdu42, cling}@csd.uwo.ca.*
- *Z.-H. Zhou is with National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China 210093. E-mail: zhouzh@lamda.nju.edu.cn.*

1. Another form of co-training, called single-view co-training in our paper, generates diverse learners on the single view of features (Goldman & Zhou, 2000; Zhou & Li, 2005). See Section 6 for a review of different forms of co-training. In this paper, we mainly study the standard two-view co-training (Blum & Mitchell, 1998), which will be referred to as two-view co-training, or simply co-training.

2. The assumptions can be relaxed for co-training to still work well (Abney, 2002; Balcan et al., 2005; Wang & Zhou, 2007). Nevertheless, the sufficiency and independence assumptions are a "sufficient condition" for co-training to work.

In this paper we attempt to answer the following crucial questions in order to apply the standard two-view co-training successfully:

1) If a whole (or large) labeled training dataset with two views is given, how can we verify if the sufficiency and independence assumptions of co-training are satisfied?
2) If a whole (or large) labeled training dataset with only single view (such as most UCI datasets (Asuncion & Newman, 2007)) is given, can we split the single view into two sufficient and independent views, such that the two-view co-training could be applied?[3]
3) The above two questions are both based on the whole (or large) labeled dataset. If a small training set is given (as in most real-world applications where co-training is called for), can we still verify the sufficiency and independence assumptions? Can we still split single view into two views to apply co-training? How is co-training compared with other popular semi-supervised learning methods (such as EM with naive Bayes) given small training sets?

We attempt to answer these important questions regarding co-training in this paper. In Section 2 we describe a simple but novel method to empirically verify the sufficiency and independence assumptions (answering question 1 above). In Section 3 we study the empirical relation between the two assumptions and the performance of co-training, and propose four increasingly sophisticated methods to split single views into two views, in order to make the standard two-view co-training work more reliably on single-view datasets (answering question 2 above). In Section 4 we study those questions again based only on small labeled training sets (answering question 3 above). In Section 5 we compare the performance between the proposed view-splitting co-training and EM with naive Bayes on small training sets (answering question 3 above). Section 6 relates our work to previous works, and Section 7 contains discussions and conclusions.

Our conclusions for co-training's effectiveness are mixed. If two views are given, and known to satisfy the two assumptions, co-training works well. Otherwise, based on small labeled training sets, verifying the assumptions or splitting single view into two views are unreliable, thus it is uncertain whether the standard co-training would work or not.

3. Most UCI datasets are given in a single view of features, and thus, the single-view co-training (Goldman & Zhou, 2000; Zhou & Li, 2005) could be applied directly. However, the single-view co-training algorithms require some additional control to avoid the accumulation of the negative influence of unlabeled examples mislabeled by one learner for the other learner, and the control is often complicated.

2 Verifying Co-training Assumptions Empirically

Given a whole (or large) labeled dataset and two views of features ($X = x_1, \ldots, x_m$ and $Y = y_1, \ldots, y_n$), how can we verify if the two assumptions on sufficiency and independence for the standard co-training are satisfied? It has been proved (Blum & Mitchell, 1998) that if the assumptions are satisfied, co-training is guaranteed to work, and thus can be applied. Note that sometimes the domain knowledge can ensure the satisfaction of the two assumptions, but in most real-world applications, such assumptions are usually unknown, or cannot be guaranteed. Thus it is important that these assumptions are empirically verified based on the dataset given. Here we will use the whole (or large) labeled dataset that *represents the learning task* to verify the two assumptions. This is because the theoretical assumptions on sufficiency and independence are based on the whole domain (for example, it is assumed that there exist target functions that map from the whole view ($X \times Y$), the X view, and the Y view to the class labels perfectly (Blum & Mitchell, 1998)). In Section 4 we will only use small sets of training data to verify the assumptions.

Empirical verification of the co-training assumptions seems to be a crucial task to determine if co-training can be successfully applied in real-world applications where only data is given. However, as far as we know, little work on empirical verification has been done in the past.

The sufficiency assumption is relatively easy to verify. Sufficiency means that $X \times Y$ can accurately predict the class, so can X and Y individually. We can build a classifier to estimate the accuracy on the whole (or large) dataset D using $X \times Y$ with the 10-fold cross-validation. We denote this accuracy as p. The sufficiency requires that p should be close to 1. Note that the theoretical results assume that there exist (target) functions that map from $X \times Y$, X, and Y to the class labels perfectly. As we are verifying the assumption empirically, we use learning algorithms on the whole (or large) dataset to establish if such functions exist or not. Similarly, we build two classifiers using features in X and Y individually to estimate the accuracy of X (call it p_x) and the accuracy of Y (call it p_y).

Thus, the sufficiency assumption of co-training can be defined as: there exists a small positive number δ_1 (such as 0.1) such that

$p > 1 - \delta_1$,
$p_x > 1 - \delta_1$, and
$p_y > 1 - \delta_1$.

We call δ_1 the *sufficiency parameter*. In Section 3, we will discover the empirical relation between δ_1 and the performance of co-training.

Conditional independence is a bit harder to verify. It means that given the class, the two views are independent. One way to verify this is, for each class label, we need to verify if each x_i is independent of Y, and each y_i is independent of X. To verify if x_i is independent of

Y empirically, we build a classifier (or many classifiers) to predict x_i using Y on the whole (or large) dataset. If x_i is independent of Y, then Y cannot predict x_i well — not better than the default accuracy of x_i. Specifically, assume that the 10-fold cross-validated accuracy of Y predicting x_i on the whole (or large) dataset is p_{x_i}, then it should not be much greater than the default accuracy of x_i — the accuracy (denoted as p'_{x_i}) of the majority value of the class. The same is true for using X to predict y_j. Thus, the independence assumption can be defined as: there exists a small positive number δ_2 (such as 0.1) such that for each class value

$p_{x_i} < p'_{x_i} + \delta_2$ for all $1 \le i \le m$, and
$p_{y_i} < p'_{y_i} + \delta_2$ for all $1 \le i \le n$.

We call δ_2 the *independence parameter*. We will discover the empirical relation between δ_2 and the performance of co-training in Section 3.

Several natural language datasets (such as *WebKB Course*, *20-newsgroups*, and *Reuters*) have been used repeatedly in the previous publications to show that the standard two-view co-training works well (for example, (Blum & Mitchell, 1998; Nigam & Ghani, 2000)). Here we verify empirically the two assumptions of co-training on one of them, the *WebKB Course* dataset. After calculating the sufficiency and independence parameters on the whole dataset of *WebKB Course*, we obtain $\delta_1 = 0.12$ and $\delta_2 = 0.15$, which are both rather small. According to the general pattern discovered in Section 3, co-training is very likely to work when δ_1 and δ_2 are both small. This thus explains why co-training works well on this dataset in many previous publications.

What about other datasets, especially datasets with single views?

3 Splitting Single Views to Two Views

In the previous section we described an empirical approach to verify, when given the whole (or large) dataset and two views, if the two views satisfy the sufficiency and independence assumptions for co-training to work. However, the standard two-view co-training has limited success in most real-world datasets with *single* views, such as most UCI datasets (Asuncion & Newman, 2007). (One could also directly apply the single-view co-training to the datasets with single views, but other complications may be entailed.)

In this section we propose four increasingly sophisticated methods to split single views into two views, all based on the whole datasets.[4] Via these splitting methods, we study the empirical relation between the sufficiency and independence parameters and the working of co-training, and furthermore, make the standard two-view co-training work reliably well on single-view real-world datasets.

4. Note that, both sufficiency and independence assumptions are based on the whole domain (corresponding to the whole dataset), thus we split single views into two views by verifying these two assumptions on the whole datasets in this section. See Section 4 for feature splitting based only on small training sets.

3.1 Configurations

To study the working of co-training on a variety of situations, we choose 30 UCI datasets coming with the WEKA package (Hall et al., 2009). The datasets and their basic properties are listed in Table 1. The continuous features are discretized into 10 equal-width bins.[5] Datasets with multiple classes are converted to binary by using the majority class value as one class, and the rest of the other values as the other class. These datasets are named with "_new" appended on the end of their original names. The naive Bayes (Langley et al., 1992) is used to check the sufficiency and independence assumptions, as well as construct co-training classifiers, due to its generally good performance in classification (Langley et al., 1992).

For each feature splitting method, experiments on these UCI datasets are conducted in the following three high-level steps. In the first step, all the features of each dataset are split into two disjoint sets as two views, according to four different splitting strategies.

In the second step, the standard two-view co-training is applied to see if it works. Specifically, each whole dataset D is first split randomly into three disjoint subsets: the training set (R), the unlabeled set (U), and the test set (T). The test set T is always 25% of D. To make sure that co-training can possibly show improvement when the unlabeled data are added, we choose a small training set for each dataset such that the "optimal gain"[6] is large enough (greater than 10%). The training sizes of the 30 UCI datasets range from 1/500 to 1/50 of the whole datasets, as shown in Table 1. The unlabeled set (U) is the whole dataset (D) taking away the test set (T) and the training set (R). The standard co-training (Blum & Mitchell, 1998) is then applied. The process is repeated 20 times with different splits of R, U, and T. A significance test, a paired t-test with 95% confidence, is then applied to see if co-training works (i.e., if co-training with the unlabeled data significantly outperforms the original labeled-data-only-learning).

In the third step, the general relation between the two parameters and the working of co-training is explored, and the proposed splitting methods are compared.

3.2 Splitting Methods

In this section, we study four increasingly sophisticated splitting methods: random split (Section 3.2.1), entropy split (Section 3.2.2), entropy-start hill climbing (Section 3.2.3), and random-restart hill climbing (Section 3.2.4). Then we compare the performance of the standard two-view co-training, as well as the sufficiency and independence parameters, among these four methods (Section 3.2.5).

5. As we will use the naive Bayes to implement co-training, features need to be discretized.

6. The "optimal gain" is the difference between the accuracy on the initial training set R plus all unlabeled data with correct labels and the accuracy on R alone (without any benefit of unlabeled examples). The "optimal gain" may reflect the upper bound that co-training can achieve in accuracy.

	No. of Features	No. of Examples	Class Distribution	Training Size
breast-cancer	9	277	196/81	1/50
breast-w	9	699	458/241	1/100
colic	22	368	232/136	1/50
credit-a	15	690	307/383	1/50
credit-g	20	1000	700/300	1/100
diabetes	8	768	500/268	1/50
heart-statlog	13	270	150/120	1/50
hepatitis	19	155	32/123	1/50
ionosphere	33	351	126/225	1/50
kr-vs-kp	36	3196	1669/1527	1/200
mushroom	22	8124	4208/3916	1/500
sonar	60	208	97/111	1/50
tic-tac-toe	9	958	332/626	1/100
vote	16	435	267/168	1/100
anneal_new	38	898	214/684	1/50
audiology_new	69	226	169/57	1/50
autos_new	25	205	138/67	1/50
cmc_new	9	1473	629/844	1/100
cylinder-bands_new	39	540	228/312	1/50
dermatology_new	34	366	112/254	1/50
ecoli_new	7	336	143/193	1/50
flags_new	28	194	125/69	1/50
glass_new	9	214	138/76	1/50
heart-c_new	13	303	165/138	1/50
heart-h_new	12	294	188/106	1/50
primary-tumor_new	17	339	84/255	1/50
solar-flare_1_new	12	323	235/88	1/50
solar-flare_2_new	11	1066	735/331	1/100
spambase_new	57	4601	2788/1813	1/300
splice_new	60	3190	1535/1655	1/200

TABLE 1
The 30 UCI datasets used in the experiments

3.2.1 Random Split

Although random split is the simplest and most straightforward method to split single views into two views, it is a natural way to discover empirical relation between the sufficiency and independence assumptions and the working of co-training. It also sets up a baseline to compare other splitting methods described later in this section. In our experiments, each dataset is split into two views randomly for 20 times, thus we can obtain 20 different feature splits for each dataset, and 600 different splits for the whole 30 datasets.

The experimental results show that the working of co-training varies dramatically among these 30 UCI datasets, and Figure 1 plots all the splits on four representative datasets of these 30 datasets. We can see from Figure 1 that, co-training wins with all the 20 splits on "breast-w" and with most splits (17 out of 20) on "dermatology_new", but loses with all the 20 splits on "kr-vs-kp". It shows that, for these three datasets, co-training either wins or loses, regardless of splits.[7] However, for all the rest 27 datasets (such as "credit-a" in Figure 1), various splits yield various co-training results. This thus gives us a chance to discover the relation between the splits and the working of co-training.

To find out the relation in general, we combine all the splits on the whole 30 datasets together for analysis. Figure 2 plots all the 600 random splits on the whole 30 datasets. We can see from Figure 2 that, co-training works (wins) when δ_1 and δ_2 are both very small. For example, when δ_1 is smaller than 0.08 and δ_2 is smaller than 0.22 (as "Winning Area" shown in Figure 2), co-training always works. This observation coincides well with the previous theoretical findings on co-training assumptions, but more importantly, it provides us with a practical approach to check these assumptions on real-world datasets. Specifically, given a split (on a dataset), we could check if δ_1 and δ_2 are smaller than the empirical thresholds (such as, 0.08 and 0.22 respectively) to see if the sufficiency and independence assumptions are likely to be satisfied. Note that, all the splits falling into "Winning Area" in Figure 2 are from only two of these 30 datasets ("breast-w" and "dermatology_new"). This indicates that, with random split, co-training is guaranteed to work only on these two datasets.

In addition, Figure 2 also reveals a general pattern of the working of co-training: the winning cases appear denser in the lower left part of the figure, while the tying and losing cases are denser in the upper right part. In order to make it clearer, Table 3 presents the numbers of splits where co-training wins, ties and loses, and the averages of δ_1 and δ_2 on these splits. We can clearly

7. We check the predictive ability of every single feature on these three datasets via naive Bayes. On "breast-w", every single feature can build a classifier with higher than 79% accuracy; on "dermatology_new", all the accuracies of these single-feature classifiers are also higher than 70%; however, on "kr-vs-kp", most single-feature classifiers have only around 55% accuracy. High predictive ability on every feature makes co-training win with all (most) splits on "breast-w" and "dermatology_new", and low predictive ability makes it lose with all splits on "kr-vs-kp".

Datasets	Random Section 3.1	Entropy Section 3.2	ES-HC Section 3.3	RR-HC Section 3.4	Local-HC Section 4	EM+naiveBayes Section 5
breast-cancer	T	T	T	W	T	T
breast-w	W	W	W	W	W	W
colic	T	T	T	T	W	T
credit-a	T	T	W	W	T	W
credit-g	L	T	L	T	T	L
diabetes	T	T	T	W	T	W
heart-statlog	W	W	W	W	W	W
hepatitis	T	T	T	T	T	T
ionosphere	T	T	T	W	W	T
kr-vs-kp	L	L	L	L	L	L
mushroom	L	T	T	W	L	W
sonar	T	T	T	T	T	T
tic-tac-toe	T	T	T	T	T	T
vote	T	T	T	T	T	T
anneal_new	L	L	T	T	L	T
audiology_new	L	T	T	T	T	L
autos_new	T	T	T	T	T	T
cmc_new	T	T	T	T	T	T
cylinder-bands_new	L	L	L	T	L	L
dermatology_new	W	W	W	W	W	W
ecoli_new	W	W	W	W	W	W
flags_new	T	T	T	T	T	T
glass_new	T	T	T	T	T	T
heart-c_new	T	T	W	W	T	T
heart-h_new	T	W	T	W	W	W
primary-tumor_new	L	T	T	T	T	L
solar-flare_1_new	T	T	W	T	T	T
solar-flare_2_new	L	T	W	W	L	T
spambase_new	W	W	T	T	W	T
splice_new	T	T	T	W	T	T
Average of $\delta 1$	0.260	0.222	0.205	0.230	0.385	-
Average of $\delta 2$	0.229	0.243	0.130	0.040	0.001	-
Performance (w/t/l)	5/17/8	6/21/3	8/19/3	13/16/1	8/17/5	8/17/5

TABLE 2
Comparison of all methods on 30 UCI datasets. "Random", "Entropy", "ES-HC" and "RR-HC" (Sections 3.2.1 to 3.2.4) represent co-training with random split, entropy split, entropy-start hill climbing and random-restart hill climbing (minimizing "$\delta_1 + \delta_2$") respectively, all based on the whole dataset. "Local-HC" (Section 4) represents co-training with random-restart hill climbing, based on only training set. "EM+naiveBayes" (Section 5) represents EM with naive Bayes, based on only training set. "W", "T" or "L" in each entry means the method at the corresponding column wins, ties or loses on the dataset at the corresponding row. The averages of δ_1 and δ_2 on the whole 30 datasets are also presented in the bottom of the table.

see from Table 3 that, δ_1 and δ_2 for co-training to win are both significantly smaller than those for co-training to tie, and δ_2 for co-training to tie is also significantly smaller than that for co-training to lose. It gives us a clue that, co-training is more likely to work with smaller δ_1 and δ_2, even when the sufficiency and independence assumptions might not be satisfied.

	No. of Splits	δ_1	δ_2
Win	137	0.194 ± 0.097	0.157 ± 0.089
Tie	331	0.281 ± 0.082	0.230 ± 0.146
Lose	132	0.278 ± 0.079	0.299 ± 0.174

TABLE 3
The numbers of splits where co-trainig wins, ties and loses, and the averages of δ_1 and δ_2 on these splits.

Column "Random" of Table 2 lists the working of co-training (i.e., if co-training wins, ties or loses) on each dataset and the averages of δ_1, δ_2 on the whole 30 datasets.[8] We can see from Column "Random" that, co-training with random split wins on 5, loses on 8, and ties on the rest 17 datasets. This can be considered as a performance baseline of feature splitting methods, and will be compared with the other heuristic feature splitting methods in the following sections.

To summarize the observations from the random split experiments:

1. Co-training always works when δ_1 and δ_2 are both very small (such as, smaller than 0.08 and 0.22 respectively). This observation coincides well with the previous theoretical findings.

2. When δ_1 and δ_2 are greater, co-training may still work. Overall, with smaller δ_1 and δ_2, co-training is more likely to win; with greater δ_1 and δ_2, co-training is more likely to tie, or even lose.

3. Co-training with random split wins on 5, loses on

8. Since features are randomly split for 20 times on each dataset, "W", "T" or "L" in each entry of Column "Random" shows the majority co-training result of 20 times run on the specific dataset.

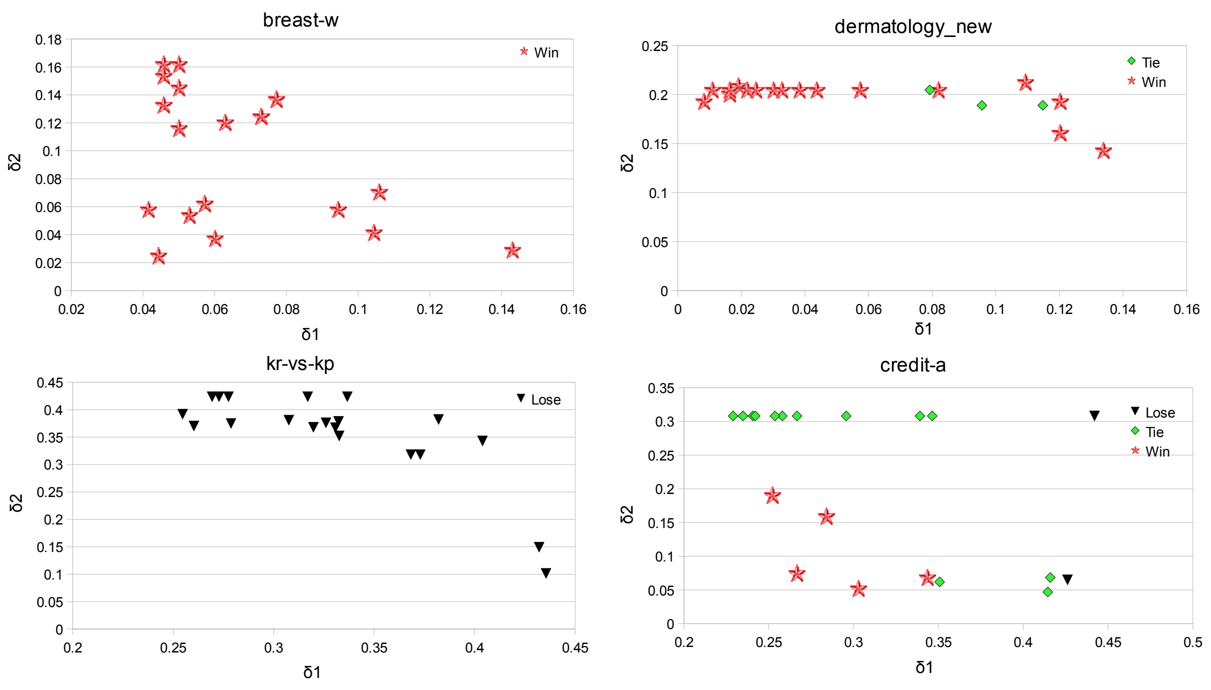

Fig. 1. Random splits on four representative UCI datasets. On each dataset, splits are labeled as “Win”, “Tie” or “Lose”, which means co-training wins, ties or loses with the split. On “breast-w” and “dermatology_new”, co-training wins with all (or most) splits; on “kr-vs-kp”, co-training loses with all splits; on “credit-a” (and all the other datasets not shown), various splits yield various co-training results.

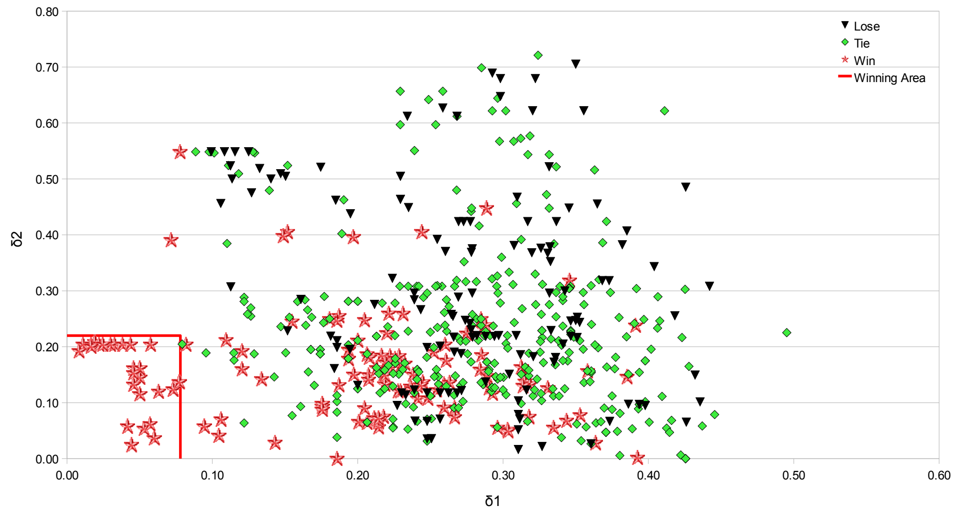

Fig. 2. Random splits on the whole 30 datasets. Each split is labeled as “Win”, “Tie” or “Lose”, which means co-training wins, ties or loses with the split. Co-training always wins when δ_1 is smaller than 0.08 and δ_2 is smaller than 0.22 (as “Winnining Area” shown in the figure).

8, and ties on the rest 17 datasets, and this is considered as a performance baseline for feature splitting methods. Due to winning on only 1/6 single-view datasets, the splitting method is clearly not very effective.

Motivated by the poor performance of the random split, we design the following three increasingly sophisticated splitting methods. With these heuristic algorithms, datasets are split into two views to make δ_1 and δ_2 as small as possible, thus co-training is expected to be more likely to work.

3.2.2 *Entropy Split*

In this section we propose a simple heuristic based on entropy to split single views into two views. The entropy split works as follows. We first calculate the entropy of each feature in the single view based on the whole dataset. This is similar to the entropy calculation when deciding which feature should be chosen as the root of the decision tree (Quinlan, 1993). Intuitively, the larger the entropy, the more predictive of the class that the feature would be. We simply assign features with the first, third, and so on (the odd number of), highest entropy to the first view. We then assign features with the second, fourth, and so on (the even number of), highest entropy to the second view. The rationale is to distribute the high-entropy features evenly in the two views, and thus, both views are more likely to be sufficient.

Based on the entropy split, the working of co-training on each dataset and the averages of δ_1 and δ_2 on the whole 30 datasets are presented in Column "Entropy" of Table 2. It shows co-training with the entropy split wins on 6, loses on 3, and ties on the rest 21 datasets. Compared with the previous random split (Column "Random"), the entropy split yields significantly smaller δ_1, slightly greater δ_2, and makes co-training win on more datasets, as well as lose on fewer datasets.

3.2.3 Entropy-Start Hill Climbing

As we mentioned in the previous section, the entropy split effectively decreases the sufficiency parameter (δ_1), but takes little effect on the independence parameter (δ_2). In order to minimize both δ_1 and δ_2, in this section, we combine the entropy split with hill climbing to further improve splitting performance.

Specifically, we first split features into two views based on the previous entropy split. Then, each feature is switched to the other view once a time, thus we can obtain a group of new generated splits. All these new generated splits are evaluated to check the sufficiency parameter (δ_1) and the independence parameter (δ_2), and the one that yields minimum $\delta_1 + \delta_2$ is remained.[9] The whole process repeats, until the split is not altered from the last iteration. This way, we can find the split that minimizes $\delta_1 + \delta_2$, which basically makes δ_1 and δ_2 as small as possible.

Based on the entropy-start hill climbing, the working of co-training on each dataset and the averages of δ_1 and δ_2 on the whole 30 datasets are presented in Column "ES-HC" of Table 2. It shows co-training with the entropy-start hill climbing wins on 8, loses on 3, and ties on the rest 19 datasets. Compared with the previous entropy split (Column "Entropy"), the entropy-start hill climbing yields smaller δ_1 and δ_2, and makes co-training win on two more datasets. Specifically, the entropy-start hill climbing slightly decreases the average of δ_1 from 0.22 to 0.21, and dramatically decreases the average of δ_2 from 0.24 to 0.13 (which is also much smaller than the average of δ_2 (0.23) of the random split).

3.2.4 Random-Restart Hill Climbing

The entropy-start hill climbing starts from the entropy based split, and deterministically ends at the split minimizing sum of δ_1 and δ_2, thus it might be easy to get stuck in local minima. In this section, we utilize random-restart hill climbing to get better optimization performance. The random-restart hill climbing works almost the same as the entropy-start hill climbing described in the previous section, except it starts from 20 random splits instead of only one deterministic entropy split. The splits obtained from all the 20 hill climbing searches are compared, and the one yields the minimum sum of δ_1 and δ_2 is remained as the best.

Based on the random-restart hill climbing, the working of co-training on each dataset and the averages of δ_1 and δ_2 on the whole 30 datasets are presented in Column "RR-HC" of Table 2. It shows co-training with the random-restart hill climbing wins on 13, loses on only 1, and ties on the rest 16 datasets. Compared with the previous entropy-start hill climbing, the random-restart hill climbing yields slightly greater δ_1, dramatically smaller δ_2, and makes co-training win on five more datasets, as well as lose on two fewer datasets. These comparisons clearly show that the random-restart hill climbing is superior to the previous entropy-start hill climbing.

So far, we minimize both δ_1 and δ_2 by directly minimizing $\delta_1+\delta_2$, in order to make co-training work on more datasets. However, there also exist many other ways to to minimize δ_1 and δ_2, such as, minimizing the *weighted* sum of δ_1 and δ_2. In order to compare the performance of different weights, we conduct experiments to check the change of δ_1, δ_2 and the working of co-training by minimizing various weighted sum of δ_1 and δ_2.

Table 4 presents the averages of δ_1, δ_2 and the working of co-training on the 30 UCI datasets. Seven different weights on δ_1 and δ_2 are utilized for comparison, which are "$9\delta_1+\delta_2$", "$6\delta_1+\delta_2$", "$3\delta_1+\delta_2$", "$\delta_1+\delta_2$", "$\delta_1+3\delta_2$", "$\delta_1+6\delta_2$", and "$\delta_1+9\delta_2$". It is clear from Table 4 that, with various weights, the decrease of one of δ_1 and δ_2 is always accompanied by the increase of the other, which indicates a clear trade-off between them. Thus, minimizing them equally (i.e., minimizing $\delta_1+\delta_2$) is expected to be a reasonable way to minimizing both of them. It is confirmed by Column "Ct works?" (i.e., the working of co-training) of Table 4 that, directly minimizing "$\delta_1+\delta_2$" makes co-training win on more datasets and lose on fewer datasets compared with the other weights.

3.2.5 Comparison of Four Methods

To better compare the four splitting methods, we illustrate δ_1, δ_2, $\delta_1+\delta_2$ and the working of co-training on the 30 UCI datasets here.

Figure 3 illustrates the change of δ_1, δ_2 and $\delta_1+\delta_2$, and Figure 4 illustrates the working of co-training, where "Random", "Entropy", "ES-HC" and "RR-HC" represent random split, entropy split, entropy-start hill climbing and random-restart hill climbing (minimizing "$\delta_1+\delta_2$") respectively. From Figure 3, we can notice that, both δ_1 and δ_2 roughly decrease, and $\delta_1+\delta_2$ clearly decreases, based on the four splitting methods. From Figure 4, we can see that, these four methods also increasingly make

9. Minimizing the sum of δ_1 and δ_2 is only one way to minimize both of them, and there exist many other ways (such as, minimizing the *weighted* sum of δ_1 and δ_2), see Section 3.2.4 for details.

	Average of δ_1	Average of δ_2	Ct works?(w/t/l)
$9\delta_1 + \delta_2$	0.185	0.159	11/17/2
$6\delta_1 + \delta_2$	0.186	0.159	10/15/5
$3\delta_1 + \delta_2$	0.198	0.097	12/16/2
$\delta_1 + \delta_2$	0.230	0.040	13/16/1
$\delta_1 + 3\delta_2$	0.260	0.020	12/15/3
$\delta_1 + 6\delta_2$	0.271	0.017	10/16/4
$\delta_1 + 9\delta_2$	0.272	0.017	10/16/4

TABLE 4
Comparison on δ_1, δ_2 and the working of co-training between minimizing various weighted sum of δ_1 and δ_2, all based on the 30 UCI datasets. The decrease of one of δ_1 and δ_2 is always accompanied by the increase of the other. Directly minimizing "δ_1 + δ_2" works the best on making co-training win on more datasets (lose on fewer datasets).

co-training win on more datasets and lose on fewer datasets. Thus, the general relation between these two parameters and the working of co-training can again be clearly demonstrated: co-training is more likely to work with smaller δ_1 and δ_2.

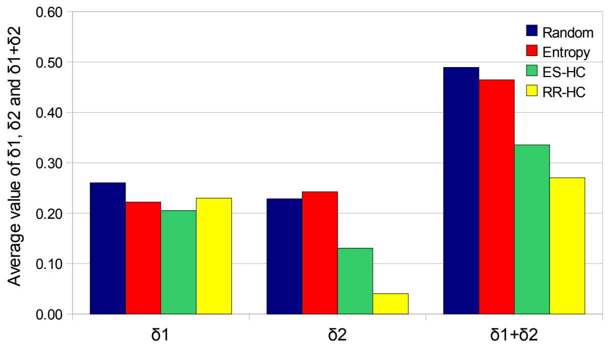

Fig. 3. Comparison on δ_1, δ_2 and $\delta_1 + \delta_2$ between all four splitting methods. "Random", "Entropy", "ES-HC" and "RR-HC" represent random split, entropy split, entropy-start hill climbing and random-restart hill climbing respectively. δ_1 and δ_2 are both decreased with the four methods.

In addition, the above observations seem to indicate that the random-restart hill climbing works the best compared with the other three methods; however, it is also the most expensive one. The random split is the simplest and cheapest method; the entropy split requires calculating entropy on all features; the entropy-start hill climbing implements one hill climbing search on the basis of entropy calculation; and the random-restart hill climbing implements 20 hill climbing searches. The four methods are increasingly effective, but also increasingly expensive. Therefore, which method to use in real-world application depends on the specific scenario.

3.3 Reconstruction of Two Views

Our view splitting methods described previously are quite effective in discovering sufficient and independent

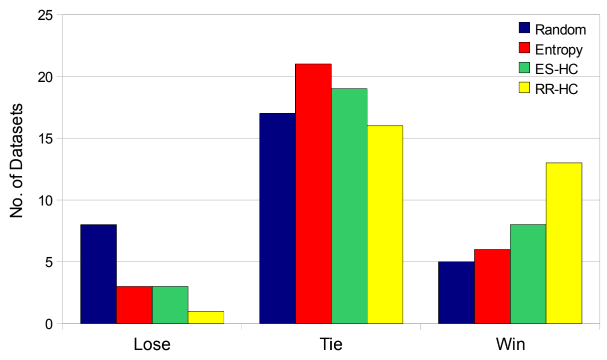

Fig. 4. Comparison on the working of co-training between all the four splitting methods. "Random", "Entropy", "ES-HC" and "RR-HC" represent random split, entropy split, entropy-start hill climbing and random-restart hill climbing respectively. The four methods increasingly make co-training win on more datasets and lose on fewer datasets.

views for the UCI datasets. However, we are not completely certain if all sufficient and independent views are discovered, as we do not have domain knowledge on these real-world datasets. In this subsection, we apply the four splitting methods to a synthetic dataset with two sufficient and independent views. As we know exactly what the true two views are for the synthetic data, we can verify if these splitting methods can successfully reconstruct them.

Specifically, we first generate a synthetic dataset with two groups of sufficient and conditionally independent features (i.e., two views). We mix these two views together to construct a single-view dataset. Then, we apply the splitting methods to see if they can successfully recover these two views.

We use 10 features in each view to generate the two-view data (i.e., $X = x_1, x_2, \ldots, x_{10}$ and $Y = y_1, y_2, \ldots, y_{10}$). We use two simple linear functions as the target functions for the two views: $f(X) = x_1 - x_2 + x_3 - x_4 + \ldots + x_9 - x_{10}$ and $f(Y) = -y_1 + y_2 - y_3 + y_4 - \ldots - y_9 + y_{10}$. For each instance, we randomly assign an integer ([1, 10]) to each of these 20 features as the feature value. Then we assign the corresponding label of the instance as 1 when both target functions are greater than 0 (i.e., $f(X) > 0$ and $f(Y) > 0$); and assign the label as -1 when both target functions are smaller than 0 (i.e., $f(X) < 0$ and $f(Y) < 0$). If the two target functions are not consistent with each other, we simply discard the current instance and generate the next one. As all the instances are directly generated according to the target functions $f(X)$ and $f(Y)$, both the X view and the Y view are sufficient. Moreover, given labels (denoted by L), all the feature values of the X view are not at all affected by the Y view (i.e., $P(X|L) = P(X|L, Y)$), thus the X view and the Y view are also conditionally independent. Therefore, we can construct a 20-feature dataset with the combined two views, both of which are sufficient and conditionally independent.

We generate 1,000 such labeled examples in our experiment. We apply the four splitting methods to this synthetic dataset, to verify if they could successfully reconstruct the two views. All the experimental configurations are the same as in the previous subsections. Table 5 shows the typical splits generated by the four methods (Rows "Random", "Entropy", "ES-HC" and "RR-HC"), as well as δ_1, δ_2, $\delta_1+\delta_2$ for each split. The true split (Row "True Split") is also presented for comparison.

We can make several interesting observations from Table 5:

1. The "True Split" (i.e., the true sufficient and conditionally independent split) yields the smallest δ_1 and δ_2 (thus $\delta_1 + \delta_2$ as well), compared with all the other splits.[10] This confirms the validity of the sufficiency and independence parameters we proposed in Section 2. In other words, by decreasing both δ_1 and δ_2, it is indeed more likely to obtain two sufficient and conditionally independent views.

2. The four proposed splitting methods yield increasingly smaller δ_1, δ_2 and $\delta_1 + \delta_2$. This confirms that more sophisticated methods can more effectively decrease the sufficiency and independence parameters.

3. Most importantly, the random-restart hill climbing method (RR-HC) yields exactly the same δ_1, δ_2 and $\delta_1 + \delta_2$ as the "True Split", and generates a very similar view-split as the true one (only one feature, $x1$, is mis-assigned). This clearly demonstrates the extraordinary splitting performance of RR-HC.[11]

To conclude, we propose a novel and effective way to measure the sufficiency and independence with given two views (Section 2), propose four increasingly sophisticated methods to split single views into two views (Sections 3.2.1 - 3.2.4), and discover the empirical relation between the sufficiency and independence assumptions and the working of co-training (Section 3.2.5).

However, our conclusions so far are all based on the whole (or large) datasets. Often co-training is needed when a small set of labeled training data is given, and we need to decide empirically if co-training can be applied. In the next section we will study the verification of the co-training assumptions and the performance of the splitting methods given only small sets of training data.

4 Verifying and Applying Co-training on Small Training Sets

The results in Section 3 coincide well with the previous theoretical findings. In addition, they also reveal that, the sufficiency and independence assumptions can be reliably verified, and the single views datasets can also be reliably split into two sufficient and independent views (if they exist) for co-training to work well. However, all these results are based on the whole (or large) datasets. In real-world applications, co-training is needed when the labeled training set is small. Thus we need know if feature splitting is still reliable, and if co-training can still be applied in this situation.

We first empirically study the same 30 UCI datasets to answer these questions. Specifically, for each UCI dataset, a small training set is randomly sampled. The size of the training data is the same as the one in Section 3 (Table 1). Then the single view of features is split into two using the most effective strategy proposed in Section 3 — random-restart hill climbing (minimizing $\delta_1 + \delta_2$) *based only on the small labeled training set* (instead of on the whole dataset as in the previous section).

For each dataset, we apply the standard co-training to the training set, and compare the predictive accuracy after co-training with the one without co-training. The process is repeated 20 times with different splits of training set, unlabeled set, and test set.[12] The significance test, as described in Section 3, is then applied to see if co-training works.

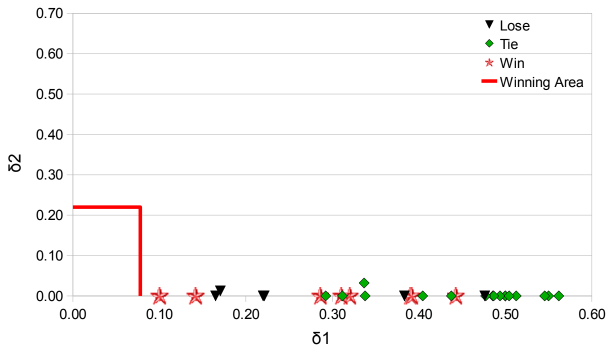

Fig. 5. The splits from random-restart hill climbing that make co-training "Win", "Tie" or "Lose" on 30 datasets. The feature splitting is based on small sets of training data.

	No. of Splits\Datasets	δ_1	δ_2
Win	8	0.298 ± 0.113	0.000 ± 0.000
Tie	17	0.455 ± 0.084	0.002 ± 0.008
Lose	5	0.283 ± 0.135	0.002 ± 0.005

TABLE 6
The numbers of splits (datasets) where co-training wins, ties and loses, and the averages of δ_1 and δ_2 on these splits (datasets), all based on small sets of training data.

Figure 5 plots 30 splits on the 30 UCI datasets, and Table 6 presents the numbers of splits (datasets) where co-training wins, ties and loses, and the averages of δ_1 and δ_2 on these splits (datasets). We can see no split in the "Winning Area" ($\delta_1 < 0.08$ and $\delta_2 < 0.22$, as

10. Note that, the sufficiency parameter (δ_1) is still 0.136 even for the true split, because we only generate 1,000 instances for the synthetic data. If we generate all valid instances, δ_1 naturally approaches to 0.

11. Here random-restart hill climbing method performs reliably well given the large synthetic data (1,000 instances). In Section 4, we evaluate its performance again, but given only small synthetic data (20 instances).

12. The split of the two views is based on the whole dataset in Section 3, and thus it remains unchanged in multiple runs of co-training. Here, the feature split changes with different training sets.

	View X	View Y	δ_1	δ_2	$\delta_1 + \delta_2$
True Split	x1, x2, x3, x4, x5, x6, x7, x8, x9, x10	y1, y2, y3, y4, y5, y6, y7, y8, y9, y10	0.136	0.000	0.136
Random	x1, x2, x3, x4, x5, x6, x8, y1, y2, y4, y5, y6, y7, y8	x7, x9, x10, y3, y9, y10	0.293	0.014	0.307
Entropy	x2, x3, x7, x8, x9, y1, y3, y5, y6, y7	x1, x4, x5, x6, x10, y2, y4, y8, y9, y10	0.202	0.000	0.202
ES-HC	x1, x2, x3, x4, x7, x8, x9, y1, y3, y5, y6	x5, x6, x10, y2, y4, y7, y8, y9, y10	0.184	0.000	0.184
RR-HC	x2, x3, x4, x5, x6, x7, x8, x9, x10	x1, y1, y2, y3, y4, y5, y6, y7, y8, y9, y10	0.136	0.000	0.136
Local-HC	x2, x3, x4, x5, x8, x9, x10, y2, y4, y5, y7, y8, y10	x1, x6, x7, y1, y3, y6, y9	0.250	0.000	0.250

TABLE 5
Comparison of the two views generated by the splitting methods (as well as the corresponding δ_1, δ_2 and $\delta_1 + \delta_2$) on the synthetic dataset. "True Split" represent the true two sufficient and independent views; "Random", "Entropy", "ES-HC" and "RR-HC" represent random split, entropy split, entropy-start hill climbing and random-restart hill climbing respectively, based on the whole dataset (1,000 instances). "Local-HC" (see Section 4) represents random-restart hill climbing, based on only training set (20 instances).

same as the one in the previous section) from Figure 5, and we can discover no general pattern from Table 6. This observation is inconsistent with what we have discovered in the previous section when the features are split based on large training sets. It indicates that, when only small sets of training data are given, the feature splitting is unreliable, and thus, difficult to make co-training work.

In addition, in Figure 5, values of δ_1 for all the splits (datasets) spread from 0.01 to 0.56, while most values of δ_2 (28 out of 30) are as small as 0. Similar phenomenon could also be discovered from Table 6 that, the averages of δ_1 are relatively great (0.283, 0.455 and 0.298), while the averages of δ_2 are very small (0.002, 0.002 and 0.000). This is again due to the small sets of training data. On one hand, sufficiency calculation (see Section 2 for details) on few training examples indicates relatively low predictive ability on both the whole view and the split two views, thus yielding greater δ_1; on the other hand, independence calculation (see Section 2 for details) on few training examples is more likely to show the independence relation between features, thus yielding smaller δ_2.

Moreover, even for a specific split (on a specific dataset), δ_1 and δ_2 both vary dramatically with different sets of training data. All of these indicate unreliable estimation of δ_1 and δ_2 given only small sets of training data, which makes the sufficiency and independence assumptions difficult to be verified.

Column "Local-HC" of Table 2 presents the working of co-training on each of the 30 UCI datasets. We can see that co-training wins on 8, loses on 5 and ties on the rest 17 datasets, given only small sets of training data. This result is slightly better than the random split (wins on 5 and loses on 8, as in Column "Random"); however, it is much worse than the same random-restart hill climbing on the whole datasets (wins on 13 and loses on 1, as in Column "RR-HC"). This result simply shows that the splitting methods do *not* work well when only given small sets of labeled training data, due to the unreliable estimation of δ_1 and δ_2.

We further study the co-training assumptions and the feature splitting on the synthetic data with a small size. In Section 3.3, we studied these issues on the large dataset (1,000 instances). Here, we conduct the same experiments on a small set of 20 instances. Specifically, we first randomly sample 20 instances from the whole dataset as the small training set, then we apply the most effective splitting method — random-restart hill climbing, to this small training set. The typical two views generated by the splitting method, as well as the corresponding δ_1, δ_2 and $\delta_1 + \delta_2$, are recorded in Table 5 (Row "Local-HC") for comparison.

We can clearly see from Table 5 that, given only limited training examples, even the most effective splitting method (random-restrat hill climbing) still cannot successfully reconstruct the true two views. It is again due to the unreliable verification of the co-training assumptions on the small training set.

To summarize, co-training is often called for in real-world applications when the labeled training sets are small. However, the experimental results on both the UCI datasets and the synthetic dataset reveal that, if only small training sets with single views are given, neither verifying the sufficiency and independence assumptions nor splitting single view into two views is reliable, thus it is uncertain whether co-training would work or not. This indicates the limitation of co-training in real-world applications.

5 Comparison between Co-Training and EM with Naive Bayes

To further assess the validity of co-training with view splitting, in this section we compare it with another popular semi-supervised learning method — EM with naive Bayes (Nigam et al., 2000). Roughly speaking, EM with naive Bayes first constructs a naive Bayes classifier based on the given labeled training data. This classifier is then used to estimate the class probability of the unlabeled data. Then, all the unlabeled data (with their estimated class probabilities) are included into the training set to construct the second classifier, and their class probability is again estimated according to this second classifier. The whole process repeats, till the probability estimation of all the unlabeled data converges.

Co-training and EM with naive Bayes have been previously compared in (Nigam & Ghani, 2000), and it shows that co-training significantly outperforms EM with naive Bayes when two sufficient and conditionally independent views are given. Here, we make no assumption about these two views, and only empirically split single views into two views. Given large training datasets, we have shown (Section 3) that view splitting is reliable, thus, co-training should outperform EM with naive Bayes. However, co-training is often expected to be applied when small single-view training sets are given, and it is unknown if co-training still outperforms EM with naive Bayes in this situation.

We apply EM with naive Bayes to the same 30 UCI datasets, and all the experimental configurations are the same as in the previous sections. Column "EM+naiveBayes" of Table 2 presents the working of EM with naive Bayes on each of the 30 UCI dataset. For better illustration, Table 7 summarizes the comparison between view-splitting co-training with whole domain, view-splitting co-training with small training data, and EM with naive Bayes with small training data.

View splitting based on the whole domain (whole dataset) can be considered as a way to discover the *ideal* two views such that co-training will be most likely to work. Indeed, we can clearly see from Table 7 that, given the whole domain, co-training with view splitting performs reasonably well on the tested UCI datasets (wins on 13, loses on 1, and ties on 16 datasets). This indicates that, if the single-view datasets are *appropriately* split into two views, co-training can indeed outperform EM with naive Bayes. However, this ideal situation is usually *unrealistic* in real-world applications, where a small set of training data is given and view splitting could only be applied accordingly. In this situation, Table 7 also shows that, both co-training with view splitting and EM with naive Bayes only work on around 1/4 tested UCI datasets (win on 8, lose on 5, and tie on 17 datasets). Clearly, co-training has no advantage over EM in this more common real-world situations.

To conclude, if the single-view datasets can be appropriately split into two views (based on the whole domain), co-training can indeed outperform EM. However, in common real-world situations where the co-training assumptions are not ensured and only small sets of training data are given, co-training with view splitting has no distinct advantage over EM with naive Bayes, and performs unreliable on the tested real-world datasets.

6 Relation to Previous Works

Current semi-supervised learning approaches can be roughly categorized into three major paradigms. In the first paradigm, a generative model is used for the classifier and the EM algorithm is utilized for label estimation and parameter estimation (Miller & Uyar, 1997; Nigam et al., 2000; Fujino et al., 2008). In the second paradigm, unlabeled instances are used to regularize the learning process, such as in the graph-based methods (Blum & Chawla, 2001; Zhu et al., 2003; Belkin & Niyogi, 2004; Zhou et al., 2005; Chapelle et al., 2003). The third paradigm is co-training. We study co-training in our paper.

The co-training paradigm was first proposed by Blum and Mitchell's seminal paper (Blum & Mitchell, 1998). It has achieved great success in many applications, such as statistical parsing (Sarkar, 2001; Steedman et al., 2003; Hwa et al., 2003), noun phrase identification (Pierce & Cardie, 2001), and image retrieval (Zhou et al., 2004; Zhou et al., 2006).

In Blum and Mitchell's paper (Blum & Mitchell, 1998), the existence of two sufficient and conditionally independent views is regarded as the required condition for co-training. Dasgupta et al. (2002) theoretically showed that when the requirement of sufficient and independent views is met, the co-trained classifiers are guaranteed to make few generalization errors, and some error bounds are derived. However, the requirement that the two sufficient views are conditionally independent given the class label is too rigid to be held in most real-world cases. Fortunately, Abney (2002) showed that the conditional independence can be relaxed to be *weak independence*. Later, Balcan et al. (2005) theoretically showed that given appropriately strong PAC-learners on each view, an assumption of *expansion* on the underlying data distribution, which is weaker than the assumption of sufficient and redundant views, is sufficient. Thus, it is not strange that even in datasets with two-views that are not independent, co-training may still be able to work well.

Most real-world data sets have only single views instead of two views. To exploit the advantages of co-training, effective single-view co-training algorithms are needed. Goldman and Zhou (2000) proposed an algorithm which does not exploit feature partition. This algorithm uses two different supervised learning algorithms to train the two classifiers. Zhou and Li (2005) proposed the tri-training approach, which uses three classifiers generated from bootstrap samples of the original training set. Wang and Zhou (2007) theoretically showed that given appropriately strong PAC-learners, the co-training process can improve performance when there is large difference between the two learners, which gives theoretical support to the success of single-view co-training algorithms.

Our work is closely related to the work of Nigam and Ghani (2000). Nigam and Ghani (2000) reported an earlier empirical study on splitting single views into two views. They found that when an independent and redundant feature split exists, co-training algorithms outperform many other algorithms using unlabeled data. Even when there is no natural feature divisions, if there are sufficient redundancy among the features, and a fairly reasonable division of the features can be identified, then co-training algorithms may show similar advantages to other algorithms. However, they only studied random

Method	Section	Setting	Performance on UCI (w/t/l)
Co-Training with View Splitting	Section 3	feature splitting on whole domain; co-training on small training sets	13/16/1
	Section 4	feature splitting on small training sets; co-training on small training sets	8/17/5
EM+naiveBayes	Section 5	on small training sets	8/17/5

TABLE 7
Comparison between co-training with view split and EM with naive Bayes.

split of the single views, and did not provide any empirical method for verifying if the two views are sufficient and independent. In our experiments we have shown that our feature splitting approaches are quite effective in producing two views for the standard co-training, and we have also discovered the general relation between the sufficiency and independence parameters and the working of co-training. It is worth noting that their empirical conclusions were mainly drawn from text data, while it is well-known that text data have large redundancy in features. Although there was no careful study on whether Nigam and Ghani's conclusions can also apply to common data (such as UCI data), those results have inspired many people to use random feature split for co-training in practice. Our study clearly shows that view splitting does not always work for co-training on single-view datasets.

7 Conclusions

This paper extends our preliminary study (Ling et al., 2009). Conclusions drawn in Section 3 coincide well with the previous theoretical findings. Moreover, they show that, if the whole (or large) training datasets are given, the feature splitting methods can successfully split single views into two views, and co-training can be applied, and expected to work well.

Unfortunately, in real-world situations, co-training is often called for when only small sets of training data are given. Conclusions drawn in Sections 4 and 5 show that, given small training datasets, the sufficiency and independence assumptions cannot be reliably verified, thus the splitting methods cannot work reliably either. In addition, in this situation, co-training with view splitting has no advantage over EM with naive Bayes, and performs unreliably.

To summarize, if the two views and the two corresponding assumptions of co-training can be ensured, co-training indeed works well. However, on the more common single-view data, verifying the assumptions or splitting single view into two views are unreliable given small labeled training sets, thus it is uncertain whether the standard co-training would work or not.

Acknowledgments

Z.-H. Zhou was supported by the National Science Foundation of China (60635030, 60721002), the Jiangsu Science Foundation (BK2008018) and the National Fundamental Research Program of China (2010CB327903).

References

Abney, S. (2002). Bootstrapping. *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics* (pp. 360–367). Philadelphia, PA.

Asuncion, A., & Newman, D. J. (2007). UCI machine learning repository.

Balcan, M. F., Blum, A., & Yang, K. (2005). Co-training and expansion: Towards bridging theory and practice. *Advances in Neural Information Processing Systems 17* (pp. 89–96). Cambridge, MA: MIT Press.

Belkin, M., & Niyogi, P. (2004). Semi-supervised learning on Riemannian manifolds. *Machine Learning, 56*, 209–239.

Blum, A., & Chawla, S. (2001). Learning from labeled and unlabeled data using graph mincuts. *Proceedings of the 18th International Conference on Machine Learning* (pp. 19–26). Williamston, MA.

Blum, A., & Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. *Proceedings of the 11th Annual Conference on Computational Learning Theory* (pp. 92–100). Madison, WI.

Chapelle, O., Schölkopf, B., & Zien, A. (Eds.). (2006). *Semi-supervised learning*. Cambridge, MA: MIT Press.

Chapelle, O., Weston, J., & Schölkopf, B. (2003). Cluster kernels for semi-supervised learning. *Advances in Neural Information Processing Systems 15*.

Dasgupta, S., Littman, M., & McAllester, D. (2002). PAC generalization bounds for co-training. *Advances in Neural Information Processing Systems 14* (pp. 375–382). Cambridge, MA: MIT Press.

Fujino, A., Ueda, N., & Saito, K. (2008). Semisupervised learning for a hybrid generative/discriminative classifier based on the maximum entropy principle. *Pattern Analysis and Machine Intelligence, IEEE Transactions on, 30*, 424–437.

Goldman, S., & Zhou, Y. (2000). Enhancing supervised learning with unlabeled data. *Proceedings of the 17th International Conference on Machine Learning* (pp. 327–334). San Francisco, CA.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I. H. (2009). The weka data mining software: an update. *SIGKDD Explorations*, *11*, 10–18.

Hwa, R., Osborne, M., Sarkar, A., & Steedman, M. (2003). Corrected co-training for statistical parsers. *Proceedings of the Workshop on the Continuum from Labeled to Unlabeled Data in Machine Learning and Data Mining, International Conference of Machine Learning*. Washington, DC.

Langley, P., Iba, W., & Thompson, K. (1992). An analysis of bayesian classifiers. *In Proceedings of the 10th National Conference on Artificial Intelligence* (pp. 223–228). AAAI Press.

Ling, C. X., Du, J., & Zhou, Z. H. (2009). When does co-training work in real data? *PAKDD '09: Proceedings of the 13th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining* (pp. 596–603). Berlin, Heidelberg: Springer-Verlag.

Miller, D. J., & Uyar, H. S. (1997). A mixture of experts classifier with learning based on both labelled and unlabelled data. *Advances in Neural Information Processing Systems 9* (pp. 571–577). Cambridge, MA: MIT Press.

Nigam, K., & Ghani, R. (2000). Analyzing the effectiveness and applicability of co-training. *Proceedings of the 9th ACM International Conference on Information and Knowledge Management* (pp. 86–93). Washington, DC.

Nigam, K., Mccallum, A. K., Thrun, S., & Mitchell, T. (2000). Text classification from labeled and unlabeled documents using EM. *Machine Learning*, *39*, 103–134.

Pierce, D., & Cardie, C. (2001). Limitations of co-training for natural language learning from large data sets. *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing* (pp. 1–9). Pittsburgh, PA.

Quinlan, R. J. (1993). *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc.

Sarkar, A. (2001). Applying co-training methods to statistical parsing. *Proceedings of the 2nd Annual Meeting of the North American Chapter of the Association for Computational Linguistics* (pp. 95–102). Pittsburgh, PA.

Steedman, M., Osborne, M., Sarkar, A., Clark, S., Hwa, R., Hockenmaier, J., Ruhlen, P., Baker, S., & Crim, J. (2003). Bootstrapping statistical parsers from small data sets. *Proceedings of the 11th Conference on the European Chapter of the Association for Computational Linguistics* (pp. 331–338). Budapest, Hungary.

Wang, W., & Zhou, Z. H. (2007). Analyzing co-training style algorithms. *Proceedings of the 18th European Conference on Machine Learning* (pp. 454–465). Warsaw, Poland.

Zhou, D., Schölkopf, B., & Hofmann, T. (2005). Semi-supervised learning on directed graphs. *Advances in Neural Information Processing Systems 17* (pp. 1633–1640). Cambridge, MA: MIT Press.

Zhou, Z. H., Chen, K. J., & Dai, H. B. (2006). Enhancing relevance feedback in image retrieval using unlabeled data. *ACM Transactions on Information Systems*, *24*, 219–244.

Zhou, Z. H., Chen, K. J., & Jiang, Y. (2004). Exploiting unlabeled data in content-based image retrieval. *Proceedings of the 15th European Conference on Machine Learning* (pp. 525–536). Pisa, Italy.

Zhou, Z. H., & Li, M. (2005). Tri-training: exploiting unlabeled data using three classifiers. *IEEE Transactions on Knowledge and Data Engineering*, *17*, 1529–1541.

Zhou, Z.-H., & Li, M. (in press). Semi-supervised learning by disagreement. *Knowledge and Information Systems*.

Zhu, X. (2006). *Semi-supervised learning literature survey* (Technical Report 1530). Department of Computer Sciences, University of Wisconsin at Madison, Madison, WI.

Zhu, X., Ghahramani, Z., & Lafferty, J. (2003). Semi-supervised learning using gaussian fields and harmonic functions. *The Twentieth International Conference on Machine Learning, August 21-24, 2003, Washington, DC USA* (pp. 912–919).

Jun Du received the BSc and MSc degrees in computer science at the China University of Geosciences in 2000 and 2005. He is currently a PhD Candidate in the Department of Computer Science at The University of Western Ontario. His research interests include data mining, machine learning, and related real-world applications. He has published over 10 peer-reviewed research papers in journals and international conferences. He is currently Ontario Graduate Scholarship (OGS) holder.

Charles X. Ling earned his dual-BSc (Computer Science & EE) from Shanghai Jiao Tong University in China. Then he studied in the Department of Computer and Information Science at University of Pennsylvania, and obtained both MSc and PhD within four years. Since then he has been a faculty member in the Department of Computer Science at the University of Western Ontario, Canada. He is currently a Full Professor, and the Director of Data Mining and Business Intelligence Lab. He is IEEE Senior Member. His main research areas include machine learning (theory, algorithms, and applications), data mining (applications in Custom Relationship Management, Web data, search engine, medical data, and so on), cognitive modeling and child education. He has published over 120 peer-reviewed research papers in journals and international conferences. He is an Associate Editor for IEEE TKDE (IEEE Transaction on Knowledge and Data Engineering), ACM Transactions on Intelligent Systems and Technology (ACM TIST), and Computational Intelligence Journal. He has been Conference Chair, PC Chair, Senior PC Member, Area Chair, and Program Committee member for major international conferences on machine learning and data mining. See http://cling.csd.uwo.ca/ for more info.

Zhi-hua Zhou (S'00-M'01-SM'06) received the BSc, MSc and PhD degrees in computer science from Nanjing University, China, in 1996, 1998 and 2000, respectively, all with the highest honors. He joined the Department of Computer Science & Technology at Nanjing University as an assistant professor in 2001, and is currently Cheung Kong Professor and Director of the LAMDA group. His research interests are in artificial intelligence, machine learning, data mining, pattern recognition, information retrieval, evolutionary computation and neural computation. In these areas he has published over 70 papers in leading international journals or conference proceedings. Dr. Zhou has won various awards/honors including the National Science & Technology Award for Young Scholars of China (2006), the Award of National Science Fund for Distinguished Young Scholars of China (2003), the Microsoft Young Professorship Award (2006), etc. He is an Associate Editor-in-Chief of *Chinese Science Bulletin*, Associate Editor of *IEEE Transactions on Knowledge and Data Engineering* and *ACM Transactions on Intelligent Systems and Technology*, and on the editorial boards of various journals. He is a co-founder of ACML, Steering Committee member of PAKDD and PRICAI, Program Committee Chair/Co-Chair of PAKDD'07, PRICAI'08 and ACML'09, vice Chair or area Chair of conferences including IEEE ICDM'06, IEEE ICDM'08, SIAM DM'09, ACM CIKM'09, etc., and General Chair/Co-Chair or Program Committee Chair/Co-Chair of a dozen of native conferences in China. He is the chair of the Machine Learning Society of the Chinese Association of Artificial Intelligence (CAAI), vice chair of the Artificial Intelligence & Pattern Recognition Society of the China Computer Federation (CCF), and chair of the IEEE Computer Society Nanjing Chapter.