ECAI 2010
H. Coelho et al. (Eds.)
IOS Press, 2010

doi:10.3233/978-1-60750-606-5-815

Combining Local and Global KNN With Cotraining

Víctor Laguna and **Alneu de Andrade Lopes** [1]

Abstract.

Semi-supervised learning is a machine learning paradigm in which the induced hypothesis is improved by taking advantage of unlabeled data. It is particularly useful when labeled data is scarce. Cotraining is a widely adopted semi-supervised approach that assumes availability of two views of the training data a restrictive assumption for most real world tasks. In this paper, we propose a one-view Cotraining approach that combines two different k-Nearest Neighbors (KNN) strategies referred to as *global* and *local* k-NN. In *global KNN*, the nearest neighbors selected to classify a new instance are given by the training examples which include this instance as one of their own k-nearest neighbors. In *local KNN*, on the other hand, the neighborhood considered when classifying a new instance is computed with the traditional KNN approach. We carried out experiments showing that a combination of these strategies significantly improves the classification accuracy in Cotraining, particularly when one single view of training data is available. We also introduce an optimized algorithm to cope with time complexity of computing the global KNN, which enables tackling real classification problems.

1 INTRODUCTION

The ever increasing storage capacity of electronic devices, coupled with the growth and popularization of the Web, promoted the proliferation of large and diverse collections of data in electronic format. Research by Gantz et. al. [4] forecasts a digital universe of nearly 1,800 exabytes in 2011, a ten-fold increase since 2006. In most cases, however, the quality of this data is not appropriate for automatic knowledge discovery tasks and considerable effort is required from experts in cleansing, structuring, and labeling the data. Although unlabeled data is usually easy to collect, this preprocessing effort is difficult, expensive, time consuming, and often impractical. In this scenario, semi-supervised algorithms have been proposed since they address this problem of using a small quantity of labeled instances with a large quantity of unlabeled ones for classification purposes.

Cotraining [2] is a very effective semi-supervised algorithm that provides a framework for using unlabeled examples to improve classification accuracy. It boosts accuracy by using a small number of labeled examples in a training set iteratively augmented with the most confident predictions from two classifiers. This approach differs from the Selftraining algorithm [10] by the number of representations or views employed in the training process. While Selftraining uses only one representation and iteratively trains a single classifier, Cotraining employs two views and trains two classifiers, one for each representation.

[1] Universidade de São Paulo - ICMC, Brazil. Email: vlaguna@icmc.usp.br, alneu@icmc.usp.br

In that sense, Cotraining is classified as a multi-view based algorithm. It assumes representations obtained from independent sources, and therefore conditionally independent given the label. Nevertheless, obtaining two representations that satisfy the Cotraining assumptions is not a trivial task, particularly the assumption on conditional independence between representations [8]. In fact, several research contributions relax the independence assumption in order to extend Cotraining usage and application domains [8, 1, 18].

Recent contributions consider the multi-view based algorithms as a category of a new paradigm referred to as *semi-supervised learning based in disagreement* [12, 17]. In this paradigm, the existence of a large disagreement between base learners is the key for a successful learning process. In this context, the existence of two explicit views is a sufficient condition, rather than a necessary one, for disagreement-based approaches. The required diversity between classifiers may be ensured in many other ways, from employing different learning algorithms [6], or ensembles [15], to adopting variations of the same learning algorithm [16].

Following the above rationale, we propose employing two alternative strategies to find the nearest neighbors from an example and combine them with Cotraining. We show that the combination of these two strategies ensures the diversity between classifiers required for a successful Cotraining process. Such results support the theory of semi-supervised learning by disagreement when just one explicit set of attributes is available.

The remainder of the paper is organized as follows. In the following Section we briefly introduce the background and related work. In Section 3 we present the concepts of Cotraining, as well as the theoretical background on semi-supervised learning by disagreement. In Section 4 we present the classification strategies to be combined and define the proposed approach, as well as an optimized classifier for the semi-supervised framework. In Section 5 we present experimental results on a collection of UCI datasets. Finally, in Section 6 we present the conclusions and discuss future work.

2 RELATED WORK

Cotraining is a widely adopted algorithm in domains where two explicit views can be obtained (see [3, 13] as examples). However, its effectiveness and usefulness are limited in most real world applications, as typically just a single attribute set is available. In this context, methods that not require two explicit views of the examples have been introduced.

Nigam and Ghani [8] compared the Cotraining model to the semi-supervised version of the Expectation Maximization (EM). In order to compare the two algorithms in scenarios where the independence assumption is disregarded, they split the features into two sub-

sets, showing that Cotraining still improves classification accuracy in those scenarios. Zhang and Zheng [14] extended this feature partitioning strategy by selecting orthogonal attributes to artificially generate two views. The authors projected the data into an orthogonal subspace using Principal Component Analysis (PCA).

Goldman and Zhou [5] applied the Cotraining algorithm to train two classifiers with different learning algorithms and just one set of features. In order to achieve a good performance, each classifier must split the instance space into a number of equivalence classes (a similar idea to equivalence classes from decision trees). When the classifiers agree, the example is used to update the labeled training set. Otherwise, the predicted label is decided based on the equivalence classes. Despite promising results, few classifiers split the instance space into equivalence classes and hence such approach is considered restrictive.

Zhou and Li [6] proposed the Tritraining, which trains three classifiers with different training sets. It differs from the Cotraining multi-view setting because Tritraining does not require explicit multi-view attribute sets. The key idea is that the majority teaches the minority, even if the three classifiers employ the same learning algorithm. Note that this approach can be extended to multiple classifiers, improving the semi-supervised classification accuracy with ensembles of learning techniques [15]. In the same line, Zhou and Li [16] generated two regressors based on the k-Nearest Neighbor algorithm (KNN) and combined them with Cotraining. Diversity between the regressors was achieved by changing the parameters of the Minkowsky distance measure.

We propose a semi-supervised approach similar to the one by Zhou and Li [16], but for classification purposes. Our approach adopts two different strategies to find the nearest neighbors, as originally proposed by Motta [7]. The author constructed a special graph called K-Associated which has a similar construction process as KNN-Networks, differing in that only instances holding the same label are connected. A new unlabeled instance is connected into the network by applying two alternative strategies called "TestInstance-Network" and "Network-TestInstance". The "TestInstance-Network" strategy connects the new instance to its k-nearest neighbor, similarly to the traditional approach. The "Network-TestInstance" strategy connects the new instance with the instances that include the new instance within their k-nearest neighborhood. Whilst a K-Associate classifier is a supervised graph-based classification approach, our approach aims at classifying data in a semi-supervised setting, adopting the combined KNN setting with Cotraining.

Variations of the KNN algorithm have been proposed in the literature that tried to take into account the global distribution of the data. The one most related to our approach was proposed by Nock et. al. [9], who introduced the Symmetric Nearest Neighbor learning rule (SKNN). It increases the nearest neighbors setting by adding to the usual k-nearest neighbors (local neighborhood) those instances considered into a global neighborhood of the target instance. Rather than merging those local and global sets, our approach proposes to handle them separately to better integrate the information from each strategy.

3 SEMI-SUPERVISED LEARNING BY DISAGREEMENT

In this Section, the Cotraining algorithm, introduced by Blum and Mitchell [2], is briefly introduced, as well as its assumptions. We also present a brief explanation of the semi-supervised learning by disagreement as a generalization of the Cotraining framework.

3.1 The Cotraining Algorithm

Cotraining is employed when multiple views of the training data are available. Its purpose is to train weak classifiers (one with each view) by using a few number of labeled examples and exploit unlabeled data to boost them in a cooperative process, where one classifier labels instances which the other classifier assumes as labeled instances. This integration of classifiers aims at overcoming the lack of labeled instances in the training process.

Formally, the learning framework of Cotraining is defined as follows. The two-view instance space is defined as $X = \left(X^{(1)} \cup X^{(2)}\right)$ over a distribution D. Let $X_L = \left(\vec{x_i}^{(1)}, \vec{x_i}^{(2)}, y_i\right)_{i=1}^{l}$ be the labeled training set and $X_U = \left(\vec{x_j}^{(1)}, \vec{x_j}^{(2)}\right)_{j=l+1}^{l+u}$ the unlabeled training set where $l = |X_L|$ and $u = |X_U|$. In order to succeed, Cotraining defines a compatibility between target function and the unknown data distribution D. Given two views $X^{(1)}$ and $X^{(2)}$, the compatibility is defined as $f_1(X^{(1)}) = f_2(X^{(2)}) = y$. Therefore, $f = (f_1, f_2)$ is compatible with the distribution D if D assigns probability zero to any example $x = (x^{(1)}, x^{(2)})$ such that $f_1(x^{(1)}) \neq f_2(x^{(2)})$.

Compatibility between target functions may be ensured with the following assumptions about the representations: they should be redundant for prediction (i.e., they should agree in the predicted label in most cases); and they should be conditionally independent given the class (i.e., information represented by each view should not be correlated). Blum and Mitchell analytically showed that a good Cotraining performance is expected if those assumptions are satisfied.

Given the two data representations, the Cotraining algorithm is as follows. Given a set X_L of labeled examples and a set X_U of unlabeled examples, each represented by two views, two weak classifiers trained with the respective representation of X_L are generated. These classifiers label every instance in set A, which is an arbitrarily sized subset from X_U. Each classifier chooses the $n + p$ most confident labeled examples from A to augment set X_L. With $n + p$ examples selected from each classifier, there will be $2n + 2p$ less examples in u, which must be replenished by drawing $2n + 2p$ random examples from X_U. This procedure iterates a certain number times, finally obtaining two semi-supervised trained classifiers. The values for n and p are user-defined and, according to Blum and Mitchell, their choice must consider the underlying data distribution. This process is summarized in Algorithm 1.

Algorithm 1: Cotraining.

input : $X_L \leftarrow$ Labeled Examples, $X_U \leftarrow$ Unlabeled Examples, $loops \leftarrow$ Loops, $p + n \leftarrow$ Number of examples to label
output: Classifiers h_1 and h_2

```
Create a random pool A from U;
for it ← 1 to loops do
    Train C1 with X(1);
    Train C2 with X(2);
    Allow C1 to label p + n more confident examples from A;
    Allow C2 to label p + n more confident examples from A;
    Add these self-labeled examples to XL;
    Replenish A with 2p + 2n examples from U;
end
return h1 and h2;
```

Blum and Mitchell justify the existence of set A to force the classifiers to select examples respecting the underlying class distribution

(i.e. A always has the same proportion of negative and positive examples). They also define a third "combined" classifier, which computes the probability of an example to belong to a given class by multiplying the probabilities calculated by each trained classifier.

3.2 Learning with Single View Multiple Classifiers

Several research works attempted to relax the independence assumption [8, 5, 1, 18, 14] to increase Cotraining's applicability. Although many were successful in applying the algorithm to mono-view domains, classic Cotraining theory can not explain those results [8, 6, 16].

In that context, an interesting theoretical contribution by Wang and Zhou [12] attempts to define a formal theoretical background to explain why information diversity, when combining classifiers, is important for the success of classification tasks. Formally, let H denote the hypothesis space and D represent the data distribution generated by the ground-truth hypothesis h^*. Let $d(h, h^*)$ be the difference between an arbitrary hypothesis and the ground-truth hypothesis defined by

$$d(h^i, h^*) = Pr_{x \in D}\left[h^i(x) \neq h^*(x)\right] \quad (1)$$

and let h^i_j be the hypothesis of $j{-}th$ classifier on the $i{-}th$ iteration of the Cotraining algorithm. As claimed in [17], their main result is presented in the following Theorem.

Theorem 1 *Given an initial labeled data X_L, with size l which is sufficient to train two classifiers h^0_1 and h^0_2 whose upper bounds on the generalization error are, respectively, $a_0 < 0.5$ and $b_0 < 0.5$ with high probability (above $1 - \delta$) in the probably approximately correct PAC model. Then h^0_1 selects u examples and puts them in σ_2 and h^1_2 is trained from σ_2 by minimizing the empirical risk. If $l * b_0 \leq e * \sqrt[M]{M!} - M$ then*

$$P(d(h^1_2, h^*) \geq b_1) \leq \delta \quad (2)$$

where $M = u * a_0$ and $b_1 = max\left[\frac{l*b_0+u*a_0-u*d(h^0_1,h^1_2)}{l}, 0\right]$.

In other words, the more information diversity is considered by classifiers, the closer will the final classifiers get to the ground-truth hypothesis h^*, with probability higher than $1 - \delta$, by learning with examples labeled from each other. Therefore, keeping a large disagreement among learners leads to a better selection of the examples to increase the labeled set X_L in the Cotraining framework.

4 COMBINING LOCAL AND GLOBAL KNN WITH COTRAINING

In this section, we define two different strategies to find the nearest neighbors. Additionally, we present some evidence of the disagreement between the classifiers based on each strategy, which enables a good synergy between them in the semi-supervised learning by disagreement framework.

4.1 Local KNN

We shall refer to the traditional KNN classifier as Local KNN. Using this classifier to predict a new instance class, the k-nearest neighbors to the target instance are taken into account, as shown in Figure 1. This strategy is referred to as *local* because the chosen neighborhood reflects the nearby locality of a target instance. A simple linear search may be employed to find the k-nearest neighbors with time complexity $O(l)$, $l = |X_L|$. If required, other strategies may be implemented to reduce the time complexity to $O(k * \log l)$, e.g., by adopting more complex structures such as *kd-trees*).

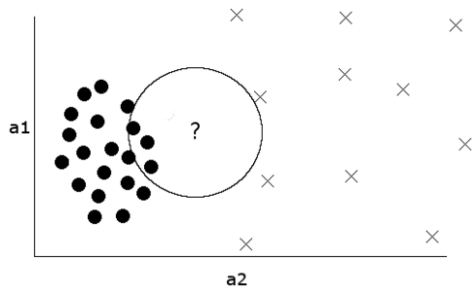

Figure 1. Neighborhood of instance "?" by the local KNN approach considering $k = 3$.

4.2 Global KNN

The Global KNN classifier selects the nearest neighbors of a target instance from a different perspective that considers the training set as a whole. The neighborhood considered is the set of training examples $\vec{x_i} \in X_L$ whose set of (local) k-nearest neighbors of $\vec{x_i}$ includes the target instance $\vec{x}_{new}$. A similar strategy was adopted by Wang et. al [11] considering the notion of citation and reference in scientific papers. Note that the neighborhood obtained with this strategy may be totally different from the one obtained with a local approach, as illustrated in Figure 2 where each of the 4 selected instances has the target instance "?" as one of its 3-nearest neighbors.

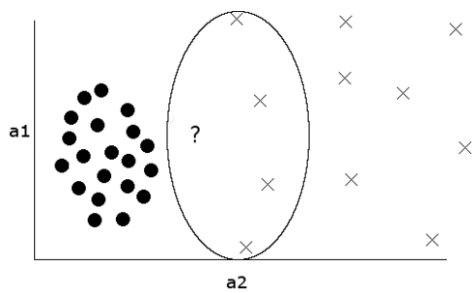

Figure 2. Neighborhood of instance "?" by the global KNN approach considering $k = 3$.

A naive implementation would execute an exhaustive search over the entire labeled training set to find the global k-nearest neighbors of each example. Let $Nk(\vec{x_i})$ be the k-nearest neighbors of example $\vec{x_i}$. Then, for each $Nk(\vec{x_i})$ with $i \in l$, we need to verify if $\vec{x}_{new} \in Nk(\vec{x_i})$. The worst-case time complexity is $O(l * (l + k)) = O(l^2)$, assuming $l > k$. Using *kd-trees* or other structures, the overall complexity of retrieving the global k-nearest neighbors may be reduced to $O(l * (k * \log l + k)) = O(l * k * \log l)$. Note that, unlike in the local approach, the number of neighbors retrieved with the global KNN approach is not fixed (and equal to k), i.e., $|Nk(\vec{x_i})_{global}| \in [0..l]$.

4.3 Disagreement Analysis

The difference between global and local KNN strategies may be analyzed from the KNN Networks perspective. Let G_k be the KNN

Network generated from the labeled training set X_L. Let's consider G_k a directed graph, i.e., the out-edges from a vertex v_i represent the connections with its k-nearest neighbors. Therefore, the in-edges connects vertex v_i with the vertices which include it as a k-nearest neighbor. When a new vertex v_{new} is connected to the network, the overall edge distribution of G_k will change. Furthermore, after the re-distribution of edges, finding both local and global nearest neighbors of v_{new} becomes a trivial process. As shown in Figure 3, the set of local nearest neighbors of "□" with $k = 3$ is $\{1, 2, 3\}$ and the set of global nearest neighbors is $\{1, 2, 3, 5\}$. Note that the bidirectional edges represent the connections with vertices in both neighborhoods.

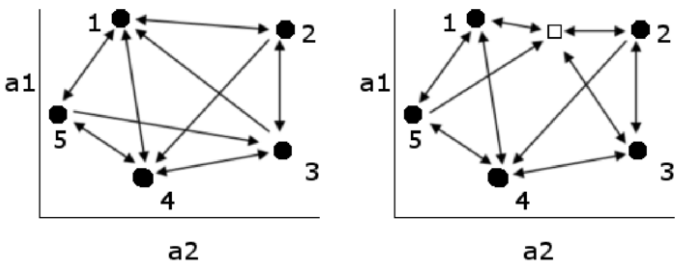

Figure 3. Re-distribution of edges in a KNN Network with $k = 3$. The square represents $\vec{x}_{new}$.

Considering a traditional way of predicting the label of an instance given its k nearest neighbors (such as majority or weighted vote), one may consider two general scenarios. The first is when the surrounding distribution SD has a unique class or a large majority class in which case the label predictions will likely be the same. This can be proved since the edge re-distribution affects just a local subgraph LS of the network which is a subset of SD. The classifiers will agree in the label predictions with high probability even though the neighborhoods do not share any vertex.

The other scenario is when the surrounding distribution SN has different classes. This is the case when the new vertex v_{new} is close to the decision boundary. In that sense, we claim that the variety between the nearest neighbors sets (i.e., the number of non bidirectional edges) will increase, specially, when the dispersion changes with the classes. As shown in Figure 4, the introduction of a new instance will have higher impact in the distribution of the out-edges when the class distribution is sparse. Moreover, the out-edges distribution of a class C will not be affected if C is compact enough. Thus, given $Nk(v_i)$ as the k-nearest neighbors of vertex v_i, there will exist an out-edge from v_i to v_{new} if

$$d(v_i, v_{new}) < argmax(d(v_i, v_j))_{v_j \in Nk(v_i)} \quad (3)$$

In summary, while the local KNN strategy considers the perspective of the added target instance as given by its out-edges, the global KNN assigns more importance to the network topology as given by its in-edges. This fact suggests that a combination of these strategies may improve classification accuracy, particularly in decision boundaries. In this context, considering such diversity between the classifiers one can expect that a Cotraining process might succeed.

4.4 The Co2KKN Algorithm

We called the proposed algorithm *Co2KNN* because it combines two KNN strategies with the Cotraining algorithm. In order to avoid a high time complexity for the global KNN strategy, we propose taking advantage of Equation 3 to optimize the algorithm. For this purpose, we maintain a list of distances Dk_{max} which stores the maximum distance from each training example $\vec{x}_i$ to its farthest nearest neighbor in $Nk(\vec{x}_i)$. It allows finding out the global nearest neighbors in linear time on the number of training instances.

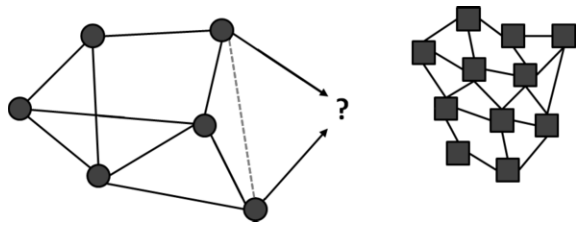

Figure 4. Impact of an added instance in the edges distribution

Algorithm 2 details this combination. Basically, it implements the traditional Cotraining algorithm with the following changes: (i) both classifier processes are based in the same view of the data; (ii) k is a new parameter; and (iii) just the combined classifier is returned. We choose the combined classifier because experiments have shown it leads to the better performance.

Algorithm 2: Co2KNN

input : $X_L \leftarrow$ Labeled Examples, $X_U \leftarrow$ Unlabeled Examples, $loops \leftarrow$ Loops, $p + n \leftarrow$ Number of examples to label and $k \leftarrow$ Number of neighbors
output: Combined Classifier h_3

```
Create A from U;
for it = 1 to loops do
    TrainLocalKNN(h1,X_L);
    TrainGlobalKNN(h2,X_L,k);
    foreach x_i ∈ A do
        V_i ←retrieveLocalKNN(X_L, x_i, k);
        Add confidence x_i to clist1;
    end
    Chose p + n examples from clist1 to increase X_L;
    foreach x_i ∈ A do
        V_i ←retrieveGlobalKNN(X_L, x_i, k);
        Add confidence x_i to clist2;
    end
    Chose p + n examples from clist2 to increase X_L;
    Chose randomly 2p + 2n from u to replenish A;
end
Define h3 as P_h3(c_j|x) ← P_h1(c_j|x^(1)) * P_h2(c_j|x^(2));
return h3
```

Training the global KNN classifier will generate the initial Dk_{max} list, which is then just updated with the added instances. Appropriate data structures such as heaps may reduce the global KNN training complexity to $O(|L| * \log k)$. Assuming a linear search for the local KNN, the overall complexity of the algorithm is $O(loops * (|L| * \log k + a * |L| + a * |L|)) = O(loops * a * |L|)$, with $a = |A|$. Moreover, choosing an appropriate value for $loops$, and assuming small enough values for p and n, the relation $|L| \ll u$ may be ensured, making the overall algorithm complexity suitable for application in real problems.

5 EXPERIMENTS

We consider a realistic scenario where a data collection must be classified and few labeled examples are available and just one feature set is available. We then generate two classifiers with each KNN strategy and combine them with Cotraining.

5.1 Datasets and Configurations

We considered 17 different datasets from the UCI repository. In order to simplify the evaluation of the proposed algorithm, we have selected datasets with two classes. As one may observe in Table 1, datasets with different class distributions were chosen. Furthermore, Cotraining parameters n and p have been chosen according to the class distribution and the data set size.

Table 1. Datasets detail

DataSet	Name	%maj class	size	*n*	*p*
1	adult	76%	48842	6	18
2	breast-cancer	70%	286	2	5
3	wisconsin-bcancer	65%	699	4	6
4	horse-colic	63%	368	4	6
5	credit-approval	55%	690	3	4
6	german-credit	70%	1000	3	7
7	diabetes	65%	768	4	6
8	heart-disease-cleav	54%	303	4	3
9	heart-disease-hunga	64%	294	3	5
10	heart-statlog	55%	270	3	2
11	hepatitis	79%	155	4	1
12	ionosphere	64%	351	1	2
13	kr-vs-kp	52%	3196	6	6
14	mushroom	51%	8124	8	8
15	sick	93%	3772	1	10
16	sonar	53%	208	2	2
17	vote	61%	435	5	3

In executing the Co2KNN algorithm we set the number of loops to 10. The size of subset A was fixed to 10% of the unlabeled training set X_U and the number of labeled examples was set to 20. Majority vote was chosen as the label prediction strategy for both Local and Global KNN. All the classifiers were implemented in the WEKA framework[2].

In order to confirm the improvement on classification precision provided by combining both strategies, we compared Co2KNN against several classifiers. Those classifiers were trained with Selftraining configured with the same parameter settings. We ran 10-fold cross validation for each configuration, using the same fold to compare the performance of each approach. The Area Under the Curve (AUC) was used as it is a more discriminative measure than accuracy, specially in scenarios with unbalanced classes.

We initially compared Co2KNN to traditional classifiers trained with Selftraining to obtain a baseline comparison. The following classifiers were chosen: Support Vector Machines (with the polynomial kernel), Decision Trees (with the J48 algorithm) and the Multinomial Naive Bayes all of them set with the default parameters of WEKA. Then, we compared Co2KNN against each strategy trained with Selftraining to ensure that the combination is, in fact, better than each approach separately. We also compare Co2KNN with SKNN, which may be considered as an alternative to Co2KNN, as stated in the background section. We perform a statistical non-parametric and paired comparison using the Friedman Test with $p < 0.05$ and the Dunn method for one-to-many comparison, and Wilcoxon test to compare Co2KNN with SKNN.

[2] http://www.cs.waikato.ac.nz/ml/weka/

5.2 Experimental Results

For the baseline comparison, Co2KNN was executed with different values of parameter k, $k = 1$, 3, 5, and 15. Results are shown in Table 2 where the significantly better classifiers for each data set are highlighted. Note that in all data sets, at least one Co2KNN classifier achieved the best performance. It is worth mentioning that, as in traditional KNN classification, k is a relevant parameter for classification performance, hence better results are expected from Co2KNN if a suitable value is selected for k, for each data set, rather than fixing it at an arbitrary value, say $k = 15$. We performed statistical comparisons which confirm the superiority of Co2KNN in all cases, except for $k = 1$ (this comparison is not presented due to space constraints).

Table 2. Baseline AUC Comparison

DataSet	Co2KNN				Bayes	J48	SVM
	k1	k3	k5	k15			
1	0.67	0.77	0.77	**0.80**	0.76	0.67	0.62
2	0.63	0.66	0.64	**0.70**	**0.70**	0.54	0.53
3	**0.97**	**0.98**	**0.99**	**0.98**	0.95	0.92	**0.97**
4	0.73	0.83	**0.85**	**0.85**	0.71	0.75	0.75
5	0.76	0.85	0.86	**0.89**	0.79	0.84	0.79
6	0.58	0.64	0.64	**0.66**	0.60	0.52	0.55
7	0.61	0.70	0.73	**0.75**	0.68	0.58	0.58
8	0.77	0.84	0.86	**0.90**	**0.89**	0.70	0.77
9	0.75	0.85	0.86	**0.89**	0.85	0.73	0.77
10	0.71	0.83	0.86	**0.88**	**0.88**	0.64	0.75
11	0.65	**0.79**	0.77	**0.80**	0.77	0.69	0.61
12	0.79	**0.86**	**0.86**	0.81	0.76	0.74	0.66
13	0.78	**0.82**	**0.83**	0.79	0.66	0.72	0.64
14	**0.96**	**0.97**	**0.96**	**0.96**	0.93	0.94	0.82
15	0.62	0.65	**0.70**	0.67	0.56	0.52	0.58
16	0.72	**0.77**	**0.76**	0.71	0.60	0.66	0.69
17	0.94	0.94	**0.98**	**0.97**	0.94	0.93	0.88

In order to compare the Cotraining combination of the global and local KNNs to performance of each individual strategy, we configured selftraining versions of the local and global KNN with $k = 1, 3, 5$, and 15 for all data sets. Results from comparing Co2KNN with both approaches trained with Selftraining are summarized in Table 3. Note that Co2KNN achieves significant better results in nearly all cases, suggesting that the Cotraining combination has a positive impact in classification performance.

Table 3. Co2KNN vs Selftraining

Classifier	k	Selftraining	
		Local KNN	Global KNN
Co2KNN	1	Not Significant	*yes*
	3	Not Significant	*yes*
	5	*yes*	*yes*
	15	*yes*	*yes*

Finally, we compared Co2KNN with the SKNN approach, which appears as a natural alternative. We employed Selftraining to train a

SKNN classifier with the same parameters and applied the Wilcoxon test for comparison. As shown in Table 4, Co2KNN shows no significant difference in the $k = 1$ scenario.

Table 4. Co2KNN vs SKNN

Classifier	k	Selftraining SKNN
Co2KNN	1	Not Significant
	3	yes
	5	yes
	15	yes

Our hypothesis is that the global KNN strategy requires more than a single nearest neighbor to reach a confident classification and improve the Cotraining process, because for $k = 1$ the set of Global KNN neighbors of an instance is usually empty. Figure5 shows the average size of the neighborhood set for $k = 1$ for both Local and Global KNN, in all data sets considered in the experiments. One observes that the average neighborhood size for Global KNN is less than 1 in most data sets, confirming that empty neighborhoods are found in several cases. In this setting, Co2KNN performance degrades to a Local KNN trained with Selftraining. These results indicate that Co2KNN must adopt values of $k > 1$. On the other hand, k should be less than the number of initial labeled examples.

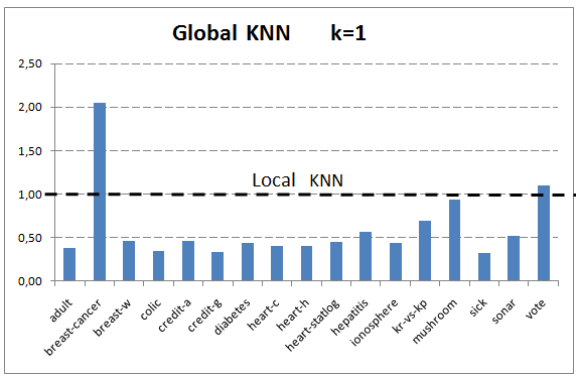

Figure 5. Average neighborhood size found by Global KNN for $k = 1$

6 CONCLUSION

We investigated the integration of two different KNN strategies into a Cotraining framework for application in a semi-supervised scenario. Results from employing the global and local KNN strategies show empirical evidence that supports the theory of semi-supervised learning by disagreement and the use of Cotraining in one-view scenarios. Although the global KNN may be seen as an expensive approach, we have proposed an optimized algorithm for real tasks. Experiments suggest that the combination of these two strategies improves classifier performance, particularly when labeled data is scarce.

Further work on global KNN is required to ensure its relevance in the Cotraining process, as well as to compare it with other semi-supervised approaches. The relation between our approach and semi-supervised ensembles will be deeply studied by combining other classifiers. As Active Learning has become a hot topic in last years, we intend to investigate how the strategy employed in the Co2KNN training process may be adapted for an active learning scenario.

ACKNOWLEDGEMENTS

This research was partially supported by the Brazilian agencies CAPES and FAPESP.

REFERENCES

[1] S. Abney, 'Bootstrapping', in *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pp. 360–367, (2002).
[2] A. Blum and T. Mitchell, 'Combining labeled and unlabeled data with co-training', in *Proceedings of the eleventh annual conference on Computational learning theory*, pp. 92–100, New York, USA, (1998). ACM.
[3] M. Collins and Y. Singer, 'Unsupervised models for named entity classification', in *Conference on empirical methods in natural language processing and very large corpora*, pp. 100–110, University of Maryland, USA, (1999).
[4] J. F. Gantz, C. Chute, S. Minton, D. Reinsel, W. Schlichting, and A. Toncheva, 'The diverse and exploiding digital universe', *External Publication of IDC sponsored by EMC*, 1–16, (2008).
[5] S. A. Goldman and Y. Zhou, 'Enhancing supervised learning with unlabeled data', in *ICML '00: Proceedings of the Seventeenth International Conference on Machine Learning*, pp. 327–334, San Francisco, CA, USA, (2000). Morgan Kaufmann Publishers Inc.
[6] M. Li, 'Tri-training: Exploiting unlabeled data using three classifiers', *IEEE Trans. on Knowl. and Data Eng.*, **17**(11), 1529–1541, (2005). Member-Zhi-Hua Zhou.
[7] R. C. Motta. Uso de redes complexas na classificação relacional, June 2009. Master's Dissertation, ICMC-USP, São Carlos, Brazil.
[8] K. Nigam and R. Ghani, 'Understanding the behavior of co-training', in *Proceedings of KDD-2000 Workshop on Text Mining*, (2000).
[9] R. Nock, M. Sebban, and P. Jabby, 'A symmetric nearest neighbor learning rule', in *EWCBR '00: Proceedings of the 5th European Workshop on Advances in Case-Based Reasoning*, pp. 222–233, London, UK, (2000). Springer-Verlag.
[10] H. Scudder, 'Probability of error of some adaptive pattern-recognition machines', *IEEE Transactions on Information Theory*, 363–371, (1965).
[11] Jun Wang, 'Solving the multiple-instance problem: A lazy learning approach', in *In Proc. 17th International Conf. on Machine Learning*, pp. 1119–1125. Morgan Kaufmann, (2000).
[12] W. Wang and Z. Zhou, 'Analyzing co-training style algorithms', in *ECML '07: Proceedings of the 18th European conference on Machine Learning*, pp. 454–465, Berlin, (2007). Springer-Verlag.
[13] R. Yan and M. Naphade, 'Multi-modal video concept extraction using co-training', *Multimedia and Expo, 2005. ICME 2005. IEEE International Conference on*, 514–517, (July 2005).
[14] W. Zhang and Q. Zheng, 'Tsfs: A novel algorithm for single view co-training', in *CSO '09: Proceedings of the 2009 International Joint Conference on Computational Sciences and Optimization*, pp. 492–496, Washington, DC, USA, (2009). IEEE Computer Society.
[15] Z. Zhou, 'When semi-supervised learning meets ensemble learning', in *MCS '09: Proceedings of the 8th International Workshop on Multiple Classifier Systems*, pp. 529–538, Berlin, (2009). Springer-Verlag.
[16] Z. Zhou and M. Li, 'Semisupervised regression with cotraining-style algorithms', *IEEE Trans. on Knowl. and Data Eng.*, **19**(11), 1479–1493, (2007).
[17] Z. Zhou and M. Li, 'Semi-supervised learning by disagreement', *Knowledge and Information Systems, in press*, 1–27, (2009).
[18] Z. Zhou, D. Zhan, and Q. Yang, 'Semi-supervised learning with very few labeled training examples', in *Twenty-Second AAAI Conference on Artificial Intelligence (AAAI-07)*, pp. 675–680, (2007).