

# Penta-Training: Clustering Ensembles with Bootstrapping of Constraints

Carlotta Domeniconi<sup>1</sup> and Muna Al-Razgan<sup>1</sup>

## Abstract.

In this paper we combine clustering ensembles and semi-supervised clustering to address the ill-posed nature of clustering. We introduce a mechanism which leverages the ensemble framework to bootstrap informative constraints directly from the data and from the various clusterings, without intervention from the user. Our approach is well suited for problems where the information available from an external source is very limited. We demonstrate the effectiveness of our proposed technique with experiments using real datasets and other state-of-the-art semi-supervised techniques.

## 1 Introduction

Clustering is the process of discovering homogeneous groups of data according to a given similarity measure. Clustering is well suited for data analysis. However, clustering is susceptible to several difficulties. It is well known that off-the-shelf clustering methods may discover very different structures in a given set of data. This is because each clustering algorithm has its own bias resulting from the optimization of different criteria. Furthermore, there is no ground truth against which the clustering result can be validated. Thus, no cross-validation technique can be carried out to tune input parameters involved in the clustering process.

Recently, cluster ensembles have emerged as a technique for overcoming problems with clustering algorithms. A cluster ensemble consists of different partitions. These partitions can be obtained from multiple applications of any single algorithm with different initializations, from various bootstrapped samples of the available data, or from the application of different algorithms to the same dataset. Cluster ensembles offer a solution to challenges inherent to clustering arising from its ill-posed nature: they can provide more robust and stable solutions by making use of the consensus across multiple clustering results, while averaging out emergent spurious structures that arise due to the various biases to which each participating algorithm is tuned.

To address the ill-posed nature of clustering, semi-supervised clustering has also emerged. Semi-supervised clustering uses prior knowledge to guide the clustering process, thus providing results that adhere to the user’s preference. Prior knowledge is often expressed in terms of pairwise constraints (must-link or cannot-link) on the data. Semi-supervised clustering has its own challenges. Often, the number of constraints available is very limited. Furthermore, the given constraints may not be effective for the improvement of clustering results. Active learning strategies to identify informative constraints have been developed. These techniques select pairs of

points which maximize some measure of “relevance”. The user (or oracle) is then queried on the nature of the relationship existing between these points.

In this work, we combine clustering ensembles with semi-supervised clustering. We use the ensemble framework to bootstrap informative constraints directly from the data, and from the different clustering components. Our approach is well suited for problems where the information available from an external source (e.g., domain expert) is very limited. We also demonstrate the feasibility of our technique to situations where prior knowledge is absent. Our work is motivated by co-training [5] and tri-training [13]. As co-training and tri-training, we leverage the ensemble methodology to perform semi-supervised learning. While co-training and tri-training use classifiers as learning components, and propagate labels among them, our technique uses a collection of clusterings (five, hence the name *Penta-Training*) to derive constraints. To the best of our knowledge, this is the first attempt of its kind.

We design a constrained version of subspace clustering, and use it as the basic component of our penta-training framework. To discover subspace clusters, we use a Locally Adaptive Clustering (or LAC) algorithm which has been proven to be effective [8, 7]. LAC is an iterative algorithm that assigns weights to features according to the local variance of data along each dimension. Dimensions along which data are loosely clustered receive a small weight, which has the effect of elongating distances along that dimension. Features along which data manifest a small variance receive a large weight, which has the effect of constricting distances along that dimension. Thus, the learned weights perform a directional local reshaping of distances which allows a better separation of clusters, and therefore the discovery of different patterns in different subspaces of the original input space. LAC depends on an input parameter (called  $h$ ) that controls the strength of the incentive to cluster on more features. Diverse clusterings can be generated using different  $h$  values.

We modify the dynamic of the LAC algorithm to embed the given constraints in the cluster assignment and weight computation processes. We call the resulting algorithm CLAC (Constrained-LAC). The individual clusterings are iteratively refined using constraints generated during the penta-training process. In particular, in each iteration of penta-training, constraints are generated for a clustering if the other four clusterings agree on them. The process is repeated until convergence, i.e., until no change in all five clusterings is observed.

## 2 Related Work

Our approach is related to co-training [5] and tri-training [13]. Co-training [5] assumes that the given features can be split into two sets, each of which is sufficient to train a classifier that will produce accu-

<sup>1</sup> Department of Computer Science, George Mason University, USA, email: carlotta@cs.gmu.edu, malrazga@gmu.edu

rate results. Each resulting classifier makes predictions on unlabeled data, and then provides new training data (those labeled with highest confidence) to its counterpart, for iterating rounds of training. Although this method broke ground in first designing collaborative classifiers to perform semi-supervised learning, its requirement for two sufficient feature sets severely limits its applicability.

Tri-training [13] does not assume redundant feature sets. Diverse classifiers (three) are generated via bootstrap sampling of the original labeled data. An unlabeled example is labeled for a classifier if the other two classifiers agree on the labeling, under certain conditions. Both co-training and tri-training require initial labeled data to train the component classifiers.

Recently, several semi-supervised clustering algorithms have been proposed. The authors in [12] propose the COP-Kmeans algorithm as a variation of  $k$ -means, where constraints are embedded during the clustering process: each point is assigned to the closest cluster, which will enact the least violation of constraints. The algorithm will not assign the point if no such cluster can be found. Two additional constraint-based variants of  $k$ -means are Seeded-KMeans and Constrained-KMeans [4]. In both algorithms, the given labeled data are used to initialize a seeded set; the constraints obtained from this labeled set are then used to guide the  $k$ -means algorithm. Seeded-KMeans allows its constraints to be violated in successive iterations, while Constrained-KMeans enforces the constraints in each iteration.

Most semi-supervised learning approaches have focused on developing new algorithms. Only recently, more attention has been given to the nature of the available knowledge, and strategies to active learn relevant side-information have been developed. In [9], constraints are imputed from the information provided by the co-association values between pairs of points in a clustering ensemble. In [6], the authors define two measures to characterize the informativeness and coherence of a set of constraints. All these methods require the existence of a domain expert. Our approach, instead, leverages the ensemble methodology to derive constraints which are completely data-driven.

### 3 Penta-Training

When building ensembles, we are faced with a difficult dilemma: accuracy versus diversity. We are in need of diverse, and yet accurate, components. In our work, to construct effective clustering ensembles, we rely on the sensitivity of the underlying subspace clustering algorithm (LAC) on its input parameter. Thus, we run the LAC algorithm multiple times with different input parameter values. (In our experiments, we observed that five components provide a sufficient range of values for the input parameter.) Furthermore, our objective is to improve the quality of the individual clusterings using a collaborative approach among the components. To achieve this goal, we need to ensure that the information shared across the components is accurate and useful. To this end, we adopt the following heuristic: we look for the pairs of points on which four (out of the five) clusterings agree, i.e., all four clusterings group the two points together, or separately. In the first case, a must-link constraint is generated for the fifth component; in the second case, a cannot-link constraint is generated. The process is iterated until no further constraints can be generated. The requirement for an agreement across four components ensures a high level of accuracy of the derived constraints. To ensure constraint relevance, we introduce a ranking mechanism among the generated constraints, and distribute only the top ranked ones to the fifth component.

The following Sections describe the details of our approach. In

order to embed constraints into subspace clustering, we organize the constraints into a graph called *Chunklet Graph*. In the following, we describe the procedure to construct such graph.

### 3.1 Chunklet Graph

We assume that two sets of constraints are given:  $M$  is the set of must-link constraints, and  $C$  is the set of cannot-link constraints. Such constraints are either provided by a domain expert, or they are bootstrapped from the data (as explained in Section 3.4).

A chunklet is a group of points that belong to the same cluster, although the identity of the cluster is unknown [3]. The size of the chunklet is equal to the number of points it contains, e.g., the chunklet  $\Delta = (\mathbf{x}_1, \mathbf{x}_2)$  has size  $|\Delta| = 2$ .

Each chunklet is formed by must-link constraints. For example, if a must-link between points  $\mathbf{x}_1$  and  $\mathbf{x}_2$  exist, then the chunklet  $\Delta_1 = (\mathbf{x}_1, \mathbf{x}_2)$  is formed. Following the formation of chunklets through must-link constraints, a transitive closure process, in which chunklets are merged, is initiated. For example, if there is a must-link constraint between  $(\mathbf{x}_1, \mathbf{x}_2)$  and  $(\mathbf{x}_1, \mathbf{x}_3)$ , then by transitive closure the chunklet  $\Delta_2 = (\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)$  is formed.

Once chunklets are formed and all transitive closures completed, a graph is created using the cannot-link constraints. These constraints prevent the assignment of some chunklets to the same cluster. If, for example, there is a cannot-link constraint between the pair  $(\mathbf{x}_3, \mathbf{x}_5)$  and we have the chunklet  $\Delta_3 = (\mathbf{x}_4, \mathbf{x}_5)$ , then our previously cited chunklet  $\Delta_2 = (\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)$  will be prevented from the assignment to the same cluster as chunklet  $\Delta_3$ .

A cannot-link constraint is represented in the graph as an edge between two vertices, where each vertex corresponds to one chunklet. In this way the entire chunklet graph  $G_{ch} = (V, E)$  is constructed, where  $V$  is a set of vertices (or chunklets) constructed from the must-link constraints, and  $|V|$  is the total number of chunklets.  $E$  is the set of edges, and an edge  $E_{ij}$  exists between vertices (chunklets)  $v_i$  and  $v_j$  iff there exist  $\mathbf{x}_i \in v_i, \mathbf{x}_j \in v_j$  such that  $(\mathbf{x}_i, \mathbf{x}_j) \in C$ .

### 3.2 Chunklet Initialization

Similar to  $k$ -means, the subspace clustering algorithm LAC depends on the initial choice of centroids. Thus, we make use of the given constraints to achieve a good initialization.

Once we have constructed the chunklet graph, we select the vertices (or chunklets) with cannot-link constraints between them, i.e., the vertices  $v_i$  and  $v_j$  such that  $E_{ij} = 1$ . We then compute the mean vectors of the points contained in each corresponding chunklet. These mean vectors become the initial centroids. If the number of selected chunklets is less than  $k$  (the number of desired clusters), we choose as additional initial centroids the points that are the farthest from the already chosen ones. The selection is iterated until we reach  $k$  initial centroids.

### 3.3 Constrained Locally Adaptive Clustering (CLAC)

The chunklet initialization procedure provides  $k$  initial centroids. We use the subspace clustering algorithm LAC to generate the clustering components of our ensemble. A full derivation of the LAC algorithm is given in [7]. Here it suffices to know that LAC is an iterative algorithm that assigns local weights to features according to the local variance of data along each dimension. Thus, at each iteration, LAC

provides a set of  $k$  centroids  $\{\mathbf{c}_1, \dots, \mathbf{c}_k\}$ , and a set of  $k$  weight vectors  $\{\mathbf{w}_1, \dots, \mathbf{w}_k\}$ , where each weight vector reflects the relevance of features within the corresponding cluster.

We modify the dynamic of the LAC algorithm to embed the given constraints into the cluster assignment process, which in turn affects the computation of weights. We call the resulting algorithm CLAC (Constrained LAC). We first present the cluster assignment strategy for points in the chunklet graph.

Chunklet assignment is the process of assigning vertices (chunklets) in the graph to the appropriate centroid without violating any of the must-link or cannot-link constraints. We consider each chunklet as a group of points, and assign all points in the chunklet to the closest centroid which does not violate any constraint. Given a vertex (chunklet)  $v_i$ , we calculate the weighted Euclidean distances between all the points  $\mathbf{x}_j \in v_i$  and each centroid  $\mathbf{c}_l$ , where  $l = 1, \dots, k$ , and look for the centroid  $\mathbf{c}_t$  that satisfies  $\mathbf{c}_t = \arg \min_l (d(v_i, \mathbf{c}_l))$ , where  $d(v_i, \mathbf{c}_l) = \sum_{j=1}^{|v_i|} \sqrt{\sum_{s=1}^D w_{ls}(x_{js} - c_{ls})^2}$ ,  $D$  is the dimensionality of the data, and  $\mathbf{w}_l$  is the weight vector associated with centroid  $\mathbf{c}_l$ . To assign a chunklet to a centroid, we need to consider possible cannot-link constraints involving the chunklet. Three cases which require different centroid assignment strategies are given below.

**Case 1**  $v_i$  is an isolated chunklet that does not have an edge in the graph  $G_{ch}$ . We assign this chunklet to the centroid  $\mathbf{c}_t = \arg \min_l (d(v_i, \mathbf{c}_l))$ .

**Case 2**  $v_i$  is a chunklet that has at least one neighbor in the graph  $G_{ch}$ , and none of its neighbors has been assigned to a centroid. As before, we assign  $v_i$  to the centroid  $\mathbf{c}_t = \arg \min_l (d(v_i, \mathbf{c}_l))$ .

**Case 3**  $v_i$  is a chunklet that has at least one neighbor in the graph  $G_{ch}$  that has been assigned to a centroid. We construct the set of centroids  $R_i$  to which the neighboring nodes of  $v_i$  have been assigned. We then assign  $v_i$  to the centroid  $\mathbf{c}_t = \arg \min_l (d(v_i, \mathbf{c}_l))$  such that  $\mathbf{c}_t \notin R_i$ .

This procedure assigns chunklets to centroids according to the local similarities being discovered by the subspace clustering algorithm.

The overall CLAC algorithm is summarized in Algorithm 1. CLAC computes a partition of the data that satisfies the given constraints (under the assumption that the satisfaction of all constraints is feasible). The inputs of the algorithm are the number of desired clusters  $k$ , the value of the  $h$  parameter (which controls the strength of the incentive to cluster on more features [7]), and the sets of constraints  $M$  and  $C$ . Centroids are initialized according to the procedure described in Section 3.2. Equal weights are assigned initially to all features. The chunklet assignment procedure is used to compute the first partition. Points which are not contained in any chunklet are assigned to the closest centroid (according to the weighted Euclidean distance). The weights are updated, according to an exponential scheme which assigns larger weights to features where points have low variance (the parameter  $h$  is used here). Thus, a new partition and new centroids are computed. The procedure is iterated until convergence, i.e., until clusters do not change.

### 3.4 The Penta-training Algorithm

Penta-Training assembles multiple (five) clusterings obtained by CLAC, and bootstraps constraints to improve the quality of the components, and ultimately of the ensemble.

Given an initial collection of constraints ( $M, C$ ), we run CLAC five times with different values of the  $h$  parameter. Each run of CLAC

---

#### Algorithm 1 CLAC Algorithm

---

**Input:**  $n$  points  $\mathbf{x} \in \mathbb{R}^D$ , number of clusters  $k, h$ , set  $M$  of must-link constraints, set  $C$  of cannot-link constraints.

1.  $S_1 = \emptyset, \dots, S_k = \emptyset$
2. Let  $(G_{ch})$  be the chunklet graph constructed from  $M$  and  $C$ ;  
**Chunklet Initialization**  
 $\forall v_i \in G_{ch}$  such that  $E_{ij} = 1$  for some  $j$   
 Compute the mean of points in  $v_i$  and assign it as initial centroid;  
 Set  $z =$  number of selected centroids;  
**while** ( $z < k$ )  
   Select the farthest point from the already selected centroids, and assign it as initial centroid;  
   Let  $\{\mathbf{c}_1, \dots, \mathbf{c}_k\}$  be the resulting centroids;
3. Set  $w_{sj} = 1/D$ , for each centroid  $\mathbf{c}_j, j = 1, \dots, k$  and each feature  $s = 1, \dots, D$ ;
4. For each  $v_i \in G_{ch}$ , let  $t_i$  be the assigned cluster centroid (as determined by the chunklet assignment procedure)  
 $\forall \mathbf{x} \in v_i, S_{t_i} = S_{t_i} \cup \{\mathbf{x}\}$ ;
5. For each centroid  $\mathbf{c}_j$ , and for each point  $\mathbf{x} \notin v_i, \forall i$   
 $S_t = S_t \cup \{\mathbf{x} | t = \arg \min_l L_w(\mathbf{c}_l, \mathbf{x})\}$   
 where  $L_w(\mathbf{c}_l, \mathbf{x}) = (\sum_{s=1}^D w_{ls}(c_{ls} - x_s)^2)^{1/2}$ ;
6. **Compute new weights**  
 For each centroid  $\mathbf{c}_j$ , and for each feature  $s$ :  
 Set  $X_{js} = \sum_{\mathbf{x} \in S_j} (c_{js} - x_s)^2 / |S_j|$ ;  
 Set  $w_{js} = \frac{\exp(-X_{js}/h)}{\sum_{s=1}^D \exp(-X_{js}/h)}$ ;
7. For each  $v_i \in G_{ch}$ , let  $t_i$  be the assigned cluster centroid (as determined by the chunklet assignment procedure)  
 $\forall \mathbf{x} \in v_i, S_{t_i} = S_{t_i} \cup \{\mathbf{x}\}$
8. For each centroid  $\mathbf{c}_j$ , and for each point ( $\mathbf{x} \notin v_i, \forall i$ )  
 $S_t = S_t \cup \{\mathbf{x} | t = \arg \min_l L_w(\mathbf{c}_l, \mathbf{x})\}$   
 where  $L_w(\mathbf{c}_l, \mathbf{x}) = (\sum_{s=1}^D w_{ls}(c_{ls} - x_s)^2)^{1/2}$ ;
9. **Compute new centroids**  
 Set  $\mathbf{c}_j = \frac{\sum_{\mathbf{x} \in S_j} \mathbf{x}}{|S_j|}$ , for each  $j = 1, \dots, k$ , where  $1_S(\cdot)$  is the indicator function of set  $S$ ;
10. Iterate 5-10 until convergence (no change in cluster assignment).

**Output:** The final partition  $S = \{S_1, \dots, S_k\}, \{\mathbf{c}_1, \dots, \mathbf{c}_k\}, \{\mathbf{w}_1, \dots, \mathbf{w}_k\}$

---

is provided with the entire data set and the entire constraint set. We obtain five clusterings of the data. Penta-training leverages the consensus achieved across such partitions to bootstrap and propagate constraints: we look for pairs of points on which four (out of the five) clusterings agree (and the fifth disagrees), i.e., all four clusterings group the two points together, or separately. In the first case, a must-link constraint is generated for the fifth component; in the second case, a cannot-link constraint is generated. Once all constraints for a given component have been generated, they are added to the current set of constraints of that component, and CLAC is re-run. The process is iterated for all combinations of four components, until no change in all five clusterings is observed. To ensure that only relevant constraints are propagated, we rank the candidate constraints, and use only the top ranked ones. In particular, for each candidate must-link constraint  $(\mathbf{x}_n, \mathbf{x}_m)$ , we compute the four weighted Euclidean distances, using the corresponding weights of the clusters the two points

$\mathbf{x}_n$  and  $\mathbf{x}_m$  are assigned to, and compute their average. The average distances are then sorted in ascending order. We select the top ranked pairs (with smallest distances) as must-link constraints for the fifth component. We proceed similarly for the cannot-link constraints. For each candidate cannot-link constraint we compute their Euclidean distance (note that in this case, four clusterings place the points in different clusters, and therefore there is no single weight vector associated with them). We then sort the distances in descending order. We select the top ranked pairs (with largest distances) as cannot-link constraints for the fifth component. In our experiments, at each iteration of penta-training, we select the top five ranked must-link and the top five ranked cannot-link constraints.

We observe that penta-training can also be applied when no constraints are initially available. In this case, we start building the ensemble by simply running the original LAC algorithm (with no side-information) using different values of  $h$ . As constraints are bootstrapped during the rounds of penta-training, LAC is substituted by CLAC. We test this scenario as well in our experiments.

At convergence, we have available five partitions (precisely, each partition corresponds to  $k$  centroids, and corresponding weight vectors). We map the problem of finding a consensus function to a graph partitioning problem, by applying the WBPA (Weighted Bipartite Partitioning) algorithm [1], which has been demonstrated to be effective. The WBPA algorithm takes into account not only how often points are grouped together across the clusterings, but also the degree of confidence of the groupings (by means of the weights). The Penta-Training algorithm is summarized in Algorithm 2.

## 4 Empirical Evaluation

### 4.1 Experimental Design

**Table 1.** Characteristics of the datasets

Dataset	$k$	$D$	n (points-per-class)
Iris	3	4	150 (50-50-50)
WDBC	2	31	424 (212-212)
Breast	2	9	478 (239-239)
Wine	3	13	144 (48-48-48)
Ionosphere	2	33	239 (126-113)

In our experiments, we used five real datasets. The characteristics of all datasets are given in Table 1. Iris, Breast, Wine and Ionosphere are from the UCI Machine Learning Repository [2]. WDBC is the Wisconsin Diagnostic Breast Cancer dataset [11].

The clustering ensemble algorithm WBPA uses METIS [10] to compute the  $k$ -way partitioning of a graph. Since METIS requires balanced datasets, we performed random sampling on Breast, WDBC, Wine, and Ionosphere. In each case, we sub-sampled the most populated class: from 357 to 212 for WDBC, from 444 to 239 for Breast, from 59 to 48 and 71 to 48 for Wine, and from 225 to 113 for Ionosphere.

We tested our penta-training framework with two scenarios: in one case, a limited number of constraints is initially available; in the second case, no constraints are available. For the first scenario, to generate the initial set of constraints, we follow the procedure introduced in [9]. We run the LAC algorithm 10 times, for  $h = 1, \dots, 10$ . We then build the co-association matrix resulting from the 10 partitions. We select all pairs of points ( $\mathbf{x}_n, \mathbf{x}_m$ ) with a co-association value in the interval  $[0.45, 0.55]$ . This means that, roughly, half of the components clusters the two points together, and the other half places them in separate groups. Thus, it is unclear whether the two

---

### Algorithm 2 Penta-Training Algorithm

---

**Input:**  $n$  points  $\mathbf{x} \in \mathbb{R}^D$ , number of clusters  $k$ ,  $h$ , set  $M$  of must-link constraints, set  $C$  of cannot-link constraints.  
 Let  $(G_{ch})$  be the chunklet graph constructed from  $M$  and  $C$ ;  
 $S^{(h_i)} = \text{CLAC}(\{\mathbf{x}\}, k, h_i, M, C)$ , for  $i = 1, \dots, 5$ ;  
 Let  $T_n^{(h_i)} \in S^{(h_i)}$ , be the set in partition  $S^{(h_i)}$  to which  $\mathbf{x}_n$  is assigned;  
 [Initialization of constraints for each component]  
**for**  $i = 1, \dots, 5$   
    $M^{(h_i)} = M, C^{(h_i)} = C$ ;  
**repeat**  
   **for**  $l = 1, \dots, 5$   
     [Bootstrapping of must-link constraints]  
     **for** every pair  $(\mathbf{x}_n, \mathbf{x}_m) \notin M$   
       **if**  $(T_n^{(h_l)} \neq T_m^{(h_l)})$  and  $(\forall i \neq l, T_n^{(h_i)} = T_m^{(h_i)})$   
         Calculate the average weighted distance:  
          $d(\mathbf{x}_n, \mathbf{x}_m) = \frac{1}{4} \sum_{i \neq l} (\sum_{s=1}^D w_{ts}(x_{n_s} - x_{m_s})^2)^{1/2}$ ;  
         [ $\mathbf{w}_t$  is the weight vector of the cluster the points  $\mathbf{x}_n$  and  $\mathbf{x}_m$  are assigned to]  
       **end if**  
     **end for**  
     Sort above distances in ascending order;  
     Select a percentage of top ranked pairs (with smallest distances), and add them to  $M_{h_l}$ ;  
     [Bootstrapping of cannot-link constraints]  
     **for** every pair  $(\mathbf{x}_n, \mathbf{x}_m) \notin C$   
       **if**  $(T_n^{(h_l)} = T_m^{(h_l)})$  and  $(\forall i \neq l, T_n^{(h_i)} \neq T_m^{(h_i)})$   
         Calculate the Euclidean distance:  
          $d(\mathbf{x}_n, \mathbf{x}_m) = \sum_{s=1}^D w_{ts}(x_{n_s} - x_{m_s})^2)^{1/2}$ ;  
       **end if**  
     **end for**  
     Sort above distances in descending order;  
     Select a percentage of top ranked pairs (with largest distances), and add them to  $C_{h_l}$ ;  
     Run  $\text{CLAC}(\{\mathbf{x}\}, k, h_l, M_{h_l}, C_{h_l})$ ;  
   **end for**  
**until** convergence [all five clusterings do not change]  
 Input the obtained five partitions (and corresponding weights) to WBPA [1];  
**Output:** Partition of the  $n$  data points into  $k$  clusters.

---

points should be clustered together or not. Therefore, the user feedback is most valuable for such points. In our experiments, we use the ground truth (class labels) to generate constraints for the selected points. For each dataset, the number of constraints generated is equal to 20% the number of data available. (If a larger number of pairs have a co-association value in the range  $[0.45, 0.55]$ , we random sample a subset.) The obtained constraints are given in input to all five CLAC components. (The same set of constraints is also given in input to the competitive techniques.) Each run of CLAC uses a different values of the  $h$  parameter. In our experiments we use the values  $\{1, 3, 5, 7, 10\}$ . According to our experience, this range of values provides in general diverse and accurate components. In the second case, when no initial constraints are available, we build the ensemble by running the LAC algorithm with the five values of  $h$ . As constraints are bootstrapped during the iterations of penta-training, LAC is substituted by CLAC.

At each iteration of penta-training, for each component, we bootstrap the top five ranked must-link constraints, and the top five ranked cannot-link constraints. We compare the following techniques:

**Table 2.** Error Rates and Standard Deviations

Methods	Iris	Breast	WDBC	Ionosphere	Wine
LAC	14.13±2.18	15.9±11.78	19.76±15.75	34.06±4.20	15.16±11.15
CLAC (20%)	13.06±1.74	16.32±11.41	16.69±8.92	34.33±6.88	15.69±11.59
COP-Kmeans (20%)	11.73±0.78	4.68±0.17	48.34± 0.5	29.87±9.17	56.46±2.17
Seeded-COP-Kmeans (20%)	<b>9.33</b>	4.39	48.34	<b>25.52</b>	55.74
Penta-Training (20%)	10.67	3.56	9.19	32.21	11.11
Penta-Training (w/o const.)	14	<b>3.14</b>	<b>8.73</b>	31.38	<b>9.03</b>

- **LAC** [7]. We run the LAC algorithm five times for  $h \in \{1, 3, 5, 7, 10\}$ , and report average error rates and standard deviations.
- **CLAC**. We run the CLAC algorithm five times for  $h \in \{1, 3, 5, 7, 10\}$ , and report average error rates and standard deviations. We provide CLAC the set of constraints generated according to the procedure described above.
- **COP-Kmeans** [12]. We run COP-Kmeans 10 times using random initialization of centroids, and report average error rates and standard deviations. Again, we provide COP-Kmeans the same set of constraints generated according to the procedure described above.
- **Seeded-COP-Kmeans** [4]. In this case centroids are initialized using the given constraints, according to the technique described in Section 3.2.
- **Penta-Training with initial constraints**. We start penta-training with the same initial set of constraints we feed all competitive semi-supervised techniques.
- **Penta-Training without initial constraints**. In this case, no external constraints are used. The ensemble starts off in an unsupervised mode (LAC components), and constraints are bootstrapped in successive iterations from the data.

## 4.2 Analysis of the Results

Error rates and standard deviations are reported in Table 2. In four problems, penta-training provided the lowest error rate, or an error rate very close to the minimum. For Ionosphere, Seeded-COP-Kmeans gives the lowest error. In some cases, penta-training provides huge improvements with respect to LAC, CLAC, and COP-Kmeans. This indicates that the collaborative approach adopted by penta-training allows the bootstrapping of accurate and relevant constraints for the clustering process. In particular, as expected, the largest improvements are achieved when LAC and CLAC have large standard deviations (i.e., on Breast, WDBC, and Wine). In these cases, the components are diverse, and the ensembles become most effective. Quite interesting is the fact that penta-training without initial constraints in most cases performs better than penta-training with initial constraints. This shows the efficacy of our that data-driven and ensemble-driven constraints. COP-Kmeans and Seeded-COP-Kmeans perform very poorly on WDBC and Wine. These data are sparse and have larger dimensionalities; in such conditions, COP-Kmeans (as  $k$ -means) has difficulties in finding the underlying cluster structure.

## 5 Conclusions and Future Work

We have introduced a semi-supervised framework for clustering ensembles which addresses the ill-posed nature of clustering. We use the ensemble framework to bootstrap informative constraints directly from the data, and from the different clustering components. Our approach is well suited for problems where the information available

from an external source is very limited, or not available at all. Furthermore, since our approach builds upon subspace clustering components, it is well suited for high dimensional data.

Several avenues can be taken for future work. As the iterations of penta-training progress, the clustering components may become correlated due to the distribution of constraints across the ensemble. As such, the best ensemble accuracy may be achieved before convergence. A criterion for an early stop of penta-training will be considered. Furthermore, a larger number of components can be used, and alternative majority schemes for the bootstrapping of constraints will be investigated. The use of soft constraints that reflect the uncertainty associated with prior knowledge will also be considered.

## ACKNOWLEDGEMENTS

This work was in part supported by NSF CAREER Award IIS-0447814.

## REFERENCES

- [1] M. Al-Razgan and C. Domeniconi, ‘Weighted clustering ensembles’, in *SIAM International Conference on Data Mining*, pp. 258–269, (2006).
- [2] A. Asuncion and D. J. Newman. UCI Machine Learning Repository, University of California, Irvine, School of Information and Computer Sciences, 2007.
- [3] A. Bar-Hillel, T. Hertz, N. Shental, and D. Weinshall., ‘Learning distance functions using equivalence relations’, in *International Conference on Machine Learning*, (2003).
- [4] S. Basu, A. Banerjee, and R. Mooney, ‘Semi-supervised clustering by seeding’, in *International Conference on Machine Learning*, (2002).
- [5] A. Blum and T. Mitchell, ‘Combining labeled and unlabeled data with co-training’, in *Conference on Computational Learning Theory*, pp. 92–100, (1998).
- [6] I. Davidson, K. Wagstaff, and S. Basu, ‘Measuring constraint-set utility for partitioning clustering algorithms’, in *European Conference on Principles and Practice of Knowledge Discovery in Databases*, pp. 115–126, (2006).
- [7] C. Domeniconi, D. Gunopulos, S. Ma, B. Yan, M. Al-Razgan, and D. Papadopoulos, ‘Locally adaptive metrics for clustering high dimensional data’, *Data Mining and Knowledge Discovery Journal*, **14**(1), 63–97, (2007).
- [8] C. Domeniconi, D. Papadopoulos, D. Gunopulos, and S. Ma, ‘Subspace clustering of high dimensional data.’, in *SIAM International Conference on Data Mining*, pp. 517–520, (2004).
- [9] D. Greene and P. Cunningham, ‘An ensemble approach to identifying informative constraints for semi-supervised clustering’, in *European Conference on Machine Learning*, pp. 140–151, (2007).
- [10] G. Karypis and V. Kumar, ‘A fast and high quality multilevel scheme for partitioning irregular graphs’, *SIAM Journal on Scientific Computing*, **20**(1), 359–392, (1998).
- [11] O. Mangasarian and W. Wolberg, ‘Cancer diagnosis via linear programming’, *SIAM News*, **23**(5), 1–18, (1990).
- [12] K. Wagstaff, C. Cardie, S. Rogers, and S. Schrödl, ‘Constrained  $k$ -means clustering with background knowledge’, in *International Conference on Machine Learning*, (2001).
- [13] Z. Zhou and M. Li, ‘Tri-training: exploiting unlabeled data using three classifiers’, *IEEE Transactions on Knowledge and Data Engineering*, **17**(11), 1529–1541, (2005).