Submitted for publication

Active Semi-Supervision for Pairwise Constrained Clustering

Sugato Basu SUGATO@CS.UTEXAS.EDU
Department of Computer Sciences, University of Texas, Austin, TX 78712

Arindam Banerjee ABANERJE@ECE.UTEXAS.EDU
Department of Electrical and Computer Engineering, University of Texas, Austin, TX 78712

Raymond Mooney MOONEY@CS.UTEXAS.EDU
Department of Computer Sciences, University of Texas, Austin, TX 78712

Abstract

Semi-supervised clustering uses a small amount of supervised data to aid unsupervised learning. One typical approach specifies a limited number of *must-link* and *cannot-link* constraints between pairs of examples. This paper presents a pairwise constrained clustering framework and a new method for actively selecting informative pairwise constraints to get improved clustering performance. Experimental and theoretical results confirm that this active querying of pairwise constraints significantly improves the accuracy of clustering when given a relatively small amount of supervision.

1. Introduction

In many learning tasks, there is a large supply of unlabeled data but limited labeled data since it can be expensive to generate. Consequently, *semi-supervised learning*, learning from a combination of both labeled and unlabeled data, has become a topic of significant recent interest (Nigam et al., 2000). More specifically, *semi-supervised clustering*, the use of class labels or *pairwise constraints* on some examples to aid unsupervised clustering, has been the focus of several recent projects (Wagstaff et al., 2001; Basu et al., 2002; Klein et al., 2002; Xing et al., 2003).

However, in order to maximize the utility of the limited labeled data available in a semi-supervised setting, supervised training examples should be actively selected as maximally informative ones rather than chosen at random (McCallum & Nigam, 1998). In this paper, we present a new method for actively selecting good pairwise constraints for semi-supervised clustering, where pairwise constraints specify that two examples must be in the same cluster (*must-link*) or different clusters (*cannot-link*) (Wagstaff et al., 2001). By actively selecting the best examples to supervise, we show that fewer constraints are required to significantly improve clustering accuracy.

Section 2 outlines the pairwise constrained clustering framework, and Section 3 presents a refinement of the popular `KMeans` clustering algorithm (Duda et al., 1999) called `PCKMeans` that utilizes pairwise constraints. In Section 4, we present a method for actively picking good queries of the form "Are these two examples in same or different classes?". Experimental results on clustering high-dimensional text data and UCI data demonstrate that active `PCKMeans` achieves significantly steeper learning curves compared to `PCKMeans` with random pairwise queries.

2. Pairwise Constrained Clustering

Centroid-based partitional clustering algorithms (e.g., `KMeans`) find a disjoint k partitioning $\{\mathcal{X}_h\}_{h=1}^{k}$ (with each partition having a centroid $\boldsymbol{\mu}_h$) of a dataset $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^{n}$ such that the total distance between the data points and the cluster centroids is (locally) minimized. We introduce a framework for pairwise constrained clustering that has pairwise *must-link* and *cannot-link* constraints (with an associated cost of violating each constraint) between points in a dataset, in addition to having distances between the points. Since centroid-based clustering cannot handle pairwise constraints explicitly, we formulate the goal of clustering in the pairwise constrained clustering framework as minimizing a combined objective function, which is defined as the sum of the total distance between the points and their cluster centroids and the cost of violating the pairwise constraints.

For the pairwise constrained clustering framework

with both *must-link* and *cannot-link* constraints, let $\mathcal{M}$ be the set of *must-link* pairs such that $(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{M}$ implies $\mathbf{x}_i$ and $\mathbf{x}_j$ should be assigned to the same cluster, and $\mathcal{C}$ be the set of *cannot-link* pairs such that $(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{C}$ implies $\mathbf{x}_i$ and $\mathbf{x}_j$ should be assigned to different clusters. Let $W = \{w_{ij}\}$ and $\overline{W} = \{\overline{w}_{ij}\}$ be two sets that give the weights corresponding to the *must-link* constraints in $\mathcal{M}$ and the *cannot-link* constraints in $\mathcal{C}$ respectively. Let l_i be the cluster assignment of a point $\mathbf{x}_i$, where $l_i \in \{1, \ldots, k\}$. Let d_M and d_C be two metrics that quantify the cost of violating *must-link* and *cannot-link* constraints: $d_M(l_i, l_j) = \mathbb{1}[l_i \neq l_j]$ and $d_C(l_i, l_j) = \mathbb{1}[l_i = l_j]$, where $\mathbb{1}$ is the indicator function ($\mathbb{1}[true] = 1$, $\mathbb{1}[false] = 0$). Using this model, the problem of pairwise constrained clustering under *must-link* and *cannot-link* constraints is formulated as minimizing the following objective function, where point $\mathbf{x}_i$ is assigned to the partition $\mathcal{X}_{l_i}$ with centroid $\boldsymbol{\mu}_{l_i}$:

$$\mathcal{J}_{\text{pckm}} = \sum_{\mathbf{x}_i \in \mathcal{X}} \|\mathbf{x}_i - \boldsymbol{\mu}_{l_i}\|^2 + \sum_{(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{M}} w_{ij} \mathbb{1}[l_i \neq l_j] + \sum_{(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{C}} \overline{w}_{ij} \mathbb{1}[l_i = l_j] \quad (1)$$

The mathematical formulation of this framework was motivated by the *metric labeling* problem and the *generalized Potts* model (Kleinberg & Tardos, 1999; Boykov et al., 1998), which only considers the set $\mathcal{M}$ of *must-link* constraints. We extended this formulation to the pairwise constrained clustering framework by adding the set $\mathcal{C}$ of *cannot-link* constraints. The metric labeling problem is generally solved by approximation algorithms. Our proposed pairwise constrained `KMeans` (`PCKMeans`) algorithm greedily optimizes $\mathcal{J}_{\text{pckm}}$ using a `KMeans`-type iteration with a modified cluster-assignment step. For experiments with text documents, we used a variant of `KMeans` called spherical `KMeans` (`SPKMeans`) (Dhillon & Modha, 2001) that has computational advantages for sparse high dimensional text data vectors. We will present our algorithm and its motivation based on `KMeans` in Section 3, but all of it can be easily extended for `SPKMeans`. Note that in the domains that we will be considering, e.g., text clustering, different costs for different pairwise constraints are not available in general, so for simplicity we will be assuming all elements of W and $\overline{W}$ to have the same constant value w in (1).

3. Clustering Algorithm

Given a set of data points $\mathcal{X}$, a set of *must-link* constraints $\mathcal{M}$, a set of *cannot-link* constraints $\mathcal{C}$, the weight of the constraints w and the number of clusters to form k, `PCKMeans` finds a disjoint k partitioning $\{\mathcal{X}_h\}_{h=1}^k$ of $\mathcal{X}$ (with each partition having a centroid $\boldsymbol{\mu}_h$) such that $\mathcal{J}_{\text{pckm}}$ is (locally) minimized.

In the initialization step of `PCKMeans`, we take the transitive closure of the *must-link* constraints (Wagstaff et al., 2001) and augment the set $\mathcal{M}$ by adding these entailed constraints. Let the number of connected components in the augmented set $\mathcal{M}$ be λ. These λ connected components are used to create λ neighborhood sets $\{N_p\}_{p=1}^{\lambda}$, where each neighborhood set consists of points connected by *must-links* from the augmented set $\mathcal{M}$. For every pair of neighborhoods N_p and $N_{p'}$ that have at least one *cannot-link* between them, we add *cannot-link* constraints between every pair of points in N_p and $N_{p'}$ and augment the *cannot-link* set $\mathcal{C}$ by these entailed constraints. We will overload notation from this point and refer to the augmented *must-link* and *cannot-link* sets as $\mathcal{M}$ and $\mathcal{C}$ respectively. Note that the neighborhood sets N_p, which contain the neighborhood information inferred from the *must-link* constraints and are unchanged during the iterations of the algorithm, are different from the partition sets $\mathcal{X}_h$, which contain the cluster partitioning information and get updated at each iteration of the algorithm.

After this preprocessing step, we get λ neighborhood sets $\{N_p\}_{p=1}^{\lambda}$. If $\lambda \geq k$, where k is the required number of clusters, we select the k neighborhood sets of largest size and initialize the k cluster centers with the centroids of these sets. If $\lambda < k$, we initialize λ cluster centers with the centroids of the λ neighborhood sets. We then look for a point $\mathbf{x}$ that is connected by *cannot-links* to every neighborhood set. If such a point exists, it is used to initialize the $(\lambda + 1)^{th}$ cluster. If there are any more cluster centroids left uninitialized, we initialize them by random points obtained by random perturbations of the global centroid of $\mathcal{X}$.

The algorithm `PCKMeans` alternates between the cluster assignment and the centroid estimation steps (see Figure 1). In the cluster assignment step of `PCKMeans`, every point $\mathbf{x}$ is assigned to a cluster such that it minimizes the sum of the distance of $\mathbf{x}$ to the cluster centroid and the cost of constraint violations incurred by that cluster assignment (by equivalently satisfying as many *must-links* and *cannot-links* it can by the assignment). Note that the cluster assignment step is order-dependent, since the subsets of $\mathcal{M}$ and $\mathcal{C}$ associated with each cluster may change with the assignment of a point. The centroid re-estimation step remains the same as `KMeans`.

In the cluster assignment step, each point moves to a new cluster only if the component of $\mathcal{J}_{\text{pckm}}$ contributed

Algorithm: `PCKMeans`
Input: Set of data points $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^n$,
set of *must-link* constraints $\mathcal{M} = \{(\mathbf{x}_i, \mathbf{x}_j)\}$,
set of *cannot-link* constraints $\mathcal{C} = \{(\mathbf{x}_i, \mathbf{x}_j)\}$,
number of clusters k, weight of constraints w.
Output: Disjoint k partitioning $\{\mathcal{X}_h\}_{h=1}^k$ of $\mathcal{X}$ such that objective function $\mathcal{J}_{\text{pckm}}$ is (locally) minimized.
Method:
1. Initialize clusters:
1a. create the λ neighborhoods $\{N_p\}_{p=1}^{\lambda}$ from $\mathcal{M}$ and $\mathcal{C}$
1b. sort the indices p in decreasing size of N_p
1c. if $\lambda \geq k$
initialize $\{\boldsymbol{\mu}_h^{(0)}\}_{h=1}^k$ with centroids of $\{N_p\}_{p=1}^k$
else if $\lambda < k$
initialize $\{\boldsymbol{\mu}_h^{(0)}\}_{h=1}^{\lambda}$ with centroids of $\{N_p\}_{p=1}^{\lambda}$
if $\exists$ point $\mathbf{x}$ *cannot-linked* to all neighborhoods $\{N_p\}_{p=1}^{\lambda}$
initialize $\boldsymbol{\mu}_{\lambda+1}^{(0)}$ with $\mathbf{x}$
initialize remaining clusters at random
2. Repeat until *convergence*
2a. `assign_cluster`: Assign each data point $\mathbf{x}$ to the cluster h^* (i.e. set $\mathcal{X}_{h^*}^{(t+1)}$), for $h^* = \arg\min_h(\|\mathbf{x} - \boldsymbol{\mu}_h^{(t)}\|^2 + w\sum_{(\mathbf{x},\mathbf{x}_i)\in\mathcal{M}} \mathbb{1}[h \neq l_i] + w\sum_{(\mathbf{x},\mathbf{x}_i)\in\mathcal{C}} \mathbb{1}[h = l_i])$
2b. `estimate_means`: $\{\boldsymbol{\mu}_h^{(t+1)}\}_{h=1}^k \leftarrow \{\frac{1}{|\mathcal{X}_h^{(t+1)}|}\sum_{\mathbf{x}\in\mathcal{X}_h^{(t+1)}} \mathbf{x}\}_{h=1}^k$
2c. $t \leftarrow (t+1)$

Figure 1. `PCKMeans` algorithm

by this *point* decreases. So when all points are given their new assignment, $\mathcal{J}_{\text{pckm}}$ will decrease or remain the same. In the centroid re-estimation step, the cluster centroids $\boldsymbol{\mu}_h$ are re-estimated using the points in $\mathcal{X}_h$ so that the component of $\mathcal{J}_{\text{pckm}}$ contributed by this *partition* is minimized. As a result only the first term (the distance component) of $\mathcal{J}_{\text{pckm}}$ is minimized in this step. Hence the objective function decreases after every cluster assignment and re-estimation step till convergence, implying that the `PCKMeans` algorithm will converge to a local minima of $\mathcal{J}_{\text{pckm}}$.

4. Active Learning Algorithm

In the semi-supervised setting, getting labels on data point pairs may be expensive. In this section, we discuss an active learning scheme in the pairwise semi-supervised setting in order to improve clustering performance with as few queries as possible. Formally, the scheme has access to a noiseless oracle that can assign a *must-link* or *cannot-link* label on a given pair $(\mathbf{x}_i, \mathbf{x}_j)$, and it can pose Q queries to the oracle. Unlike most other active learning strategies, our scheme makes all the queries up-front before starting the clustering algorithm.

In order to get pairwise constraints that are more informative than random in the pairwise constrained clustering model, we have developed an active learning scheme for selecting pairwise constraints using the *farthest-first* traversal scheme. The basic idea in farthest-first traversal is to first select a starting point at random, choose the next point to be farthest from it and add it to the traversed set, then pick the following point farthest from the traversed set (using the standard notion of distance from a set: $d(\mathbf{x}, S) = \min_{y\in S} d(\mathbf{x}, \mathbf{y})$), and so on. Farthest-first traversal gives an efficient approximation of the *k-center* problem (Hochbaum & Shmoys, 1985), and has also been used to construct hierarchical clusterings with performance guarantees at each level of the hierarchy (Dasgupta, 2002). We prove another interesting property of the farthest-first traversal for our data model (see Appendix A.2) that justifies the use of farthest-first traversal for active learning.

In (Basu et al., 2002), it was observed that initializing `KMeans` with centroids estimated from a set of labeled examples for each cluster gives significant performance improvements. Since good initial centroids are very critical for the success of greedy algorithms such as `KMeans`, we follow the same principle for the pairwise case: we will try to get as many points (proportional to the actual cluster size) as possible per cluster, so that `PCKMeans` is initialized from a very good set of centroids.

Our active learning scheme first explores the given data to get k pairwise disjoint non-null neighborhoods as fast as possible. Note that even if there is only one point per neighborhood, this neighborhood structure defines a correct skeleton of the underlying cluster. Once such a skeleton is ready, the remaining queries are used to consolidate this structure. Now, we present the details of the algorithms for performing the exploration and the consolidation.

4.1. Exploration

In the exploration phase, we use a very interesting property of the farthest-first traversal. Given a set of k disjoint balls of unequal size in a metric space, we show that the farthest-first scheme is sure to get one point from each of the k balls in a reasonably small number of attempts (see Appendix A.2). Hence, our algorithm `Explore` (see Figure 2) uses farthest-first traversal for getting a skeleton structure of the neighborhoods.

In `Explore`, while queries are still allowed and k pairwise disjoint neighborhoods have not yet been found, the point $\mathbf{x}$ farthest from all the existing neighborhoods is chosen as a candidate for starting a new neighborhood. Queries are posed by pairing $\mathbf{x}$ with a random point from each of the existing neighborhoods. If $\mathbf{x}$ is *cannot-linked* to all the existing neighborhoods, a new neighborhood is started with $\mathbf{x}$. If a *must-link* is

obtained for a particular neighborhood, $\mathbf{x}$ is added to that neighborhood. This continues till the algorithm runs out of queries, or, k pairwise disjoint neighborhoods have been found. In the latter case, the active learning scheme enters the consolidation phase.

Algorithm: `Explore`
Input: Set of data points $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^{n}$, access to an oracle that answers pairwise queries, number of clusters k, total number of queries Q.
Output: $\lambda \leq k$ disjoint neighborhoods $\{N_p\}_{h=1}^{\lambda}$ corresponding to the true clustering of $\mathcal{X}$ with at least one point per neighborhood.
Method:
1. Initialize: set all neighborhoods $\{N_p\}_{p=1}^{k}$ to null
2. Pick the first point $\mathbf{x}$ at random, add to N_1, $\lambda \leftarrow 1$
3. While queries are allowed and $\lambda < k$
   $\mathbf{x} \leftarrow$ point farthest from all existing neighborhoods $\{N_p\}_{p=1}^{\lambda}$
   if, by querying, $\mathbf{x}$ is *cannot-linked* to all existing neighborhoods
   $\lambda \leftarrow \lambda + 1$, start a new neighborhood N_λ with $\mathbf{x}$
   else
   add $\mathbf{x}$ to the neighborhood with which it is *must-linked*

Figure 2. Algorithm `Explore`

4.2. Consolidation

The basic idea in the consolidation phase is that since we now have points from all the clusters, the proper neighborhood of any random point $\mathbf{x}$ can be determined within a maximum of k queries. The queries will be formed by taking a point $\mathbf{y}$ from each of the neighborhoods in turn and asking for the label on the pair $(\mathbf{x}, \mathbf{y})$ till a *must-link* has been obtained. A *must-link* reply has to come within k queries. Note that it is practical to sort the neighborhoods in increasing order of the distance of their centroids from $\mathbf{x}$ so that the *must-link* neighborhood is encountered sooner in the querying process. The outline of the algorithm `Consolidate` is given in Figure 3.

Algorithm: `Consolidate`
Input: Set of data points $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^{n}$, access to an oracle that answers pairwise queries, number of clusters k, total number of queries Q, k disjoint neighborhoods corresponding to the true clustering of $\mathcal{X}$ with at least one point per neighborhood.
Output: k disjoint neighborhoods corresponding to the true clustering of $\mathcal{X}$ with higher number of points per neighborhood.
Method:
1. Estimate centroids $\{\boldsymbol{\mu}_h\}_{h=1}^{k}$ of each of the neighborhoods
2. While queries are allowed
2a. randomly pick a point $\mathbf{x}$ not in the existing neighborhoods
2b. sort the indices h with increasing distances $\|\mathbf{x} - \boldsymbol{\mu}_h\|^2$
2c. for $h = 1$ to k
   query $\mathbf{x}$ with each of the neighborhoods in the sorted order till a must-link is obtained, add $\mathbf{x}$ to that neighborhood

Figure 3. Algorithm `Consolidate`

Finally, we briefly address the case when the number of clusters k is not known to the active learning scheme. In this case, only `Explore` is used while queries are allowed. `Explore` will keep discovering new clusters as fast as it can. When it has obtained all the clusters, it will not have any way of knowing this. However, from this point onwards, for every farthest-first $\mathbf{x}$ it draws from the dataset, it will always find a neighborhood that is *must-linked* to it. Hence, after discovering all the clusters, `Explore` will essentially become equivalent to `Consolidate`. However, when k is known, it makes sense to invoke `Consolidate` since it picks random samples following the underlying data distribution. The random samples have certain nice properties in terms of estimating good centroids, e.g., Chernoff bounds on the centroid estimates, that the samples obtained using farthest-first traversal need not have.

5. Experiments

5.1. Methodology

In our experiments with high-dimensional text documents, we used datasets created from the `20 Newsgroups` collection. It has messages collected from 20 different Usenet newsgroups, 1000 messages from each newsgroup.[1] From the original dataset, a reduced dataset `News-all20` was created by taking a random subsample of 100 documents from each of the 20 newsgroups. By selecting 3 categories from the reduced dataset `News-all20`, two other datasets were created: `News-sim3` that consists of 3 newsgroups on similar topics (comp.graphics, comp.os.ms-windows, comp.windows.x), and `News-diff3` that consists of 3 newsgroups on different topics (alt.atheism, rec.sport.baseball, sci.space). Another dataset we used in our experiments is a subset of `Classic3` (Dhillon & Modha, 2001) containing 400 documents — 100 *Cranfield* documents, 100 *Medline* documents, and 200 *Cisi* documents. This `Classic3-subset` dataset was specifically designed to create clusters of unequal size. Similarities between data points in the text datasets were computed using cosine similarity. For experiments on low-dimensional data, we selected the UCI dataset `Iris`. The Euclidean metric was used for computing distances between points in this dataset.

We used *normalized mutual information* (NMI) as our evaluation metric, which determines the amount of statistical information shared by the random variables representing the cluster assignments and the user-labeled class assignments of the data points. We computed NMI following the methodology of Strehl et al. (2000). The NMI metric correlates well with the Rand Index metric used in other projects (Wagstaff et al., 2001; Klein et al., 2002).

We generated learning curves with 10-fold cross-

[1] http://www.ai.mit.edu/people/jrennie/20_newsgroups

validation for each dataset to determine the effect of pairwise constraints and the effectiveness of the active learning scheme. Each point in the learning curve represents a particular number of pairwise constraints given as input to the algorithm. For non-active `PCKMeans` these pairwise constraints are selected at random, while for active `PCKMeans` the constraints are selected using our active learning scheme. The clustering algorithm is run on the whole dataset, but the NMI measure is calculated only on the test set. Note that for all datasets, we did not continue the learning curve beyond 1000 queries (5000 for `News-all20`) since the general nature of the curves was evident in this range. Moreover, in practical active learning applications, it is unrealistic to expect the oracle to answer even 1000 queries.

5.2. Results

We experimented with different values of the constraint weight parameter w. If w is set to 0, `PCKMeans` becomes equivalent to the `Seeded-KMeans` algorithm (Basu et al., 2002). Here, the algorithm is initialized with seed points derived from the given constraints and then normal `KMeans` iterations are run till convergence, with the algorithm free to violate the constraints in any iteration. If w is set to a very high value, `PCKMeans` becomes effectively equivalent to the `Constrained-KMeans` algorithm (Basu et al., 2002). In this case, the algorithm is initialized with seed points derived from the given constraints and the constraints have to be satisfied in every iteration, since the constraint cost violation component of the $\mathcal{J}_{\text{pckm}}$ objective function supersedes its distance component. If w is set to an intermediate value, which was chosen to be 0.001 for the text-datasets and 1 for `Iris`, the algorithm gives a tradeoff between minimizing the total distance between points and cluster centroids and the cost of violating the constraints.

The results of the experiments are shown in Figures 4-8. The results obtained for different values of w were similar for the datasets considered (see Figure 4), showing that our algorithm is not very sensitive to the choice of w. In Figures 5-8, we will only present the results for the intermediate value of w. Note that in datasets with overlapping clusters, e.g., `Iris`, non-zero values of w gave slightly better results, since the algorithm performs well if it gets constraints from the overlap regions.

Non-active schemes: As shown in Appendix A.1, if the number of random pairwise constraints is low, the probability that the k largest neighborhoods are in fact from k different clusters is very low. Till this point on the learning curve, some of the neighborhoods used to initialize `PCKMeans` can actually belong to the same cluster, so that we may not get any representative from some of the clusters. This gives a poor initialization of `PCKMeans` that may cause the algorithm to converge to bad local minima. Consequently the clustering produced by `PCKMeans` can be unstable, resulting in varying NMI values on the test set. This initial jitter can be observed in all the Figures 4-8. Beyond this point on the learning curve, non-active `PCKMeans` will most likely be initialized with points from each cluster. So after the initial jitter, the performance of non-active `PCKMeans` improves steadily along the learning curve.

Active schemes: For the active algorithms, we consistently get significant improvements over the non-active algorithms, for all datasets we have considered.

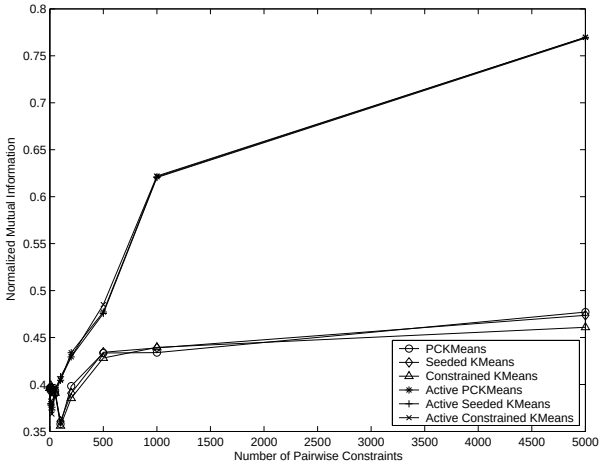

Figure 4. Comparison of NMI values on `News-all20`

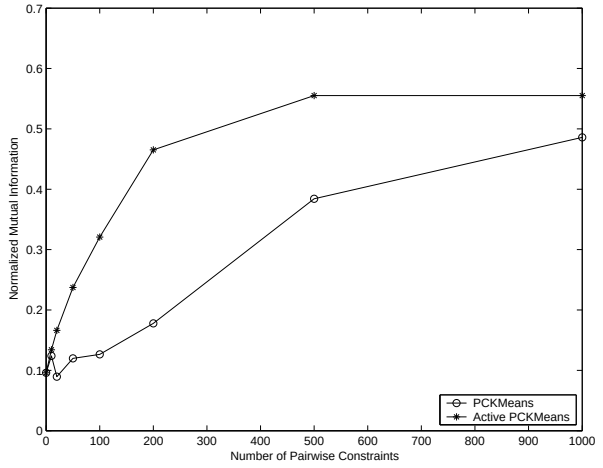

Figure 5. Comparison of NMI values on `News-sim3`

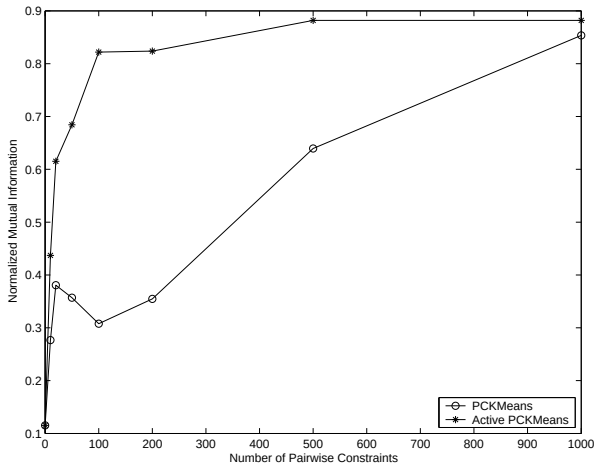

Figure 6. Comparison of NMI values on `News-diff3`

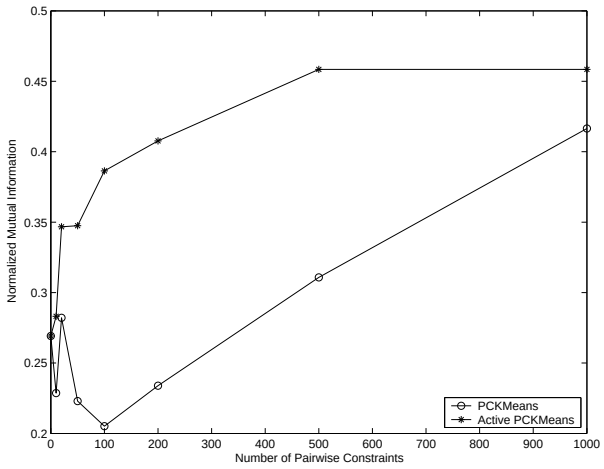

Figure 7. Comparison of NMI values on `Classic3-subset`

Firstly, we see the jitter only in the *very early* part of the learning curve. This is because the `Explore` phase creates only one neighborhood from each cluster and continues till k pairwise disjoint neighborhoods are found, creating all the neighborhoods within a small number of queries (see Appendix A.2). The jitter is so early in the learning curve that it cannot be even observed in the plots. The `Consolidate` phase grows the k neighborhoods already created, so that when the active learning scheme runs out of queries, `PCKMeans` is initialized using centroids constructed from good neighborhoods. The improvement of the active scheme is more pronounced for the difficult high-dimensional text datasets we have considered, e.g., Figures 4-7.

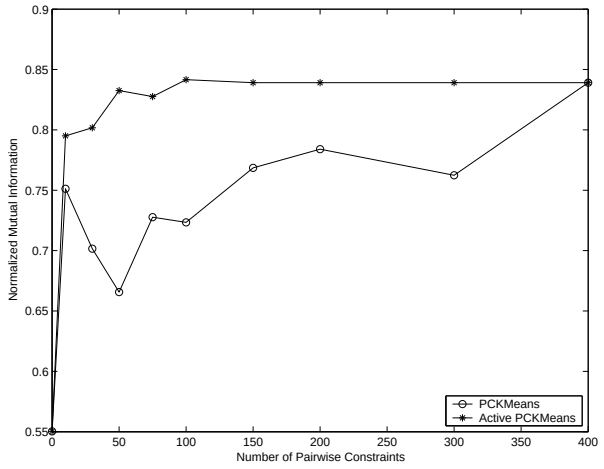

Figure 8. Comparison of NMI values on `Iris`

6. Related Work

`COP-KMeans` is another algorithm in the pairwise constrained clustering model (Wagstaff et al., 2001), but it does not handle soft-constraints, i.e., constraints that can be violated with an associated violation cost, which `PCKMeans` does. A soft-constrained algorithm `SCOP-KMeans` has been recently proposed (Wagstaff, 2002), whose performance would be interesting to compare with `PCKMeans`. Bansal et al. (2002) propose a theoretical model for pairwise constrained clustering, but their clustering model uses only pairwise constraints for clustering. Other work with the pairwise constrained clustering model includes learning distance metrics for clustering from pairwise constraints (Klein et al., 2002; Xing et al., 2003).

Active learning in the classification framework is a long-studied problem, where different principles of query selection have been studied, e.g., reduction of size of version space (Freund et al., 1997), reduction of uncertainty in predicted label (Lewis & Gale, 1994), finding high variance data points by density-weighted pool-based sampling (McCallum & Nigam, 1998), etc. These models are not applicable in the clustering framework, since the basic underlying concept of reduction of classification error and variance over the distribution of examples is not well-defined for clustering. In the unsupervised setting, Hofmann et al. (1998) consider a model of active learning which is different from ours – they have incomplete pairwise similarities between points, and their active learning goal is to select new data such that the risk of making wrong estimates about the true underlying clustering from the existing incomplete data is minimized. Klein et al. (2002) also consider active learning in semi-supervised clustering, but instead of making example-

level queries they ask the user whether or not two whole clusters should be merged. Answering example-level queries rather than cluster-level queries is a much easier task for a user, making our model more practical in a real-world active learning setting.

7. Conclusion

In this paper, we have presented a pairwise constrained clustering framework and a new theoretically well-motivated method for actively selecting good pairwise constraints for semi-supervised clustering. Experiments on text and UCI data show that our active learning scheme performs quite well, giving significantly steeper learning curves compared to random pairwise queries.

8. Acknowledgment

This research was supported in part by an IBM PhD Fellowship, by Intel and by NSF grant IIS-0117308.

References

Bansal, N., Blum, A., & Chawla, S. (2002). Correlation clustering. *IEEE Symp. on Foundations of Comp. Sci.*

Basu, S., Banerjee, A., & Mooney, R. J. (2002). Semi-supervised clustering by seeding. *Proc. of 18th Intl. Conf. on Machine Learning.*

Boykov, Y., Veksler, O., & Zabih, R. (1998). Markov random fields with efficient approximations. *IEEE Computer Vision and Pattern Recognition Conf.*

Dasgupta, S. (2002). Performance guarantees for hierarchical clustering. *Computational Learning Theory.*

Dhillon, I. S., & Modha, D. S. (2001). Concept decompositions for large sparse text data using clustering. *Machine Learning, 42*, 143–175.

Duda, R. O., Stork, D. G., & Hart, P. E. (1999). *Pattern classification, 2nd ed.* New York, NY: Wiley.

Freund, Y., Seung, H. S., Shamir, E., & Tishby, N. (1997). Selective sampling using the query by committee algorithm. *Machine Learning, 28*, 133–168.

Hochbaum, D., & Shmoys, D. (1985). A best possible heuristic for the k-center problem. *Mathematics of Operations Research, 10(2)*, 180–184.

Hofmann, T., & Buhmann, J. M. (1998). Active data clustering. *Advances in Neural Information Processing Systems 10.*

Klein, D., Kamvar, S. D., & Manning, C. (2002). From instance-level constraints to space-level constraints: Making the most of prior knowledge in data clustering. *Proc. of 18th Intl. Conf. on Machine Learning.*

Kleinberg, J., & Tardos, E. (1999). Approximation algorithms for classification problems with pairwise relationships: Metric labeling and Markov random fields. *IEEE Symp. on Foundations of Comp. Sci.*

Lewis, D., & Gale, W. (1994). A sequential algorithm for training text classifiers. *Proc. of 17th Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval.*

McCallum, A. K., & Nigam, K. (1998). Employing EM and pool-based active learning for text classification. *Proc. of 15th Intl. Conf. on Machine Learning.*

Motwani, R., & Raghavan, P. (1995). *Randomized algorithms.* Cambridge University Press.

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

Strehl, A., Ghosh, J., & Mooney, R. (2000). Impact of similarity measures on web-page clustering. *Workshop on Artificial Intelligence for Web Search (AAAI 2000).*

Wagstaff, K. (2002). *Intelligent clustering with instance-level constraints.* Doctoral dissertation, Cornell University.

Wagstaff, K., Cardie, C., Rogers, S., & Schroedl, S. (2001). Constrained K-Means clustering with background knowledge. *Proc. of 18th Intl. Conf. on Machine Learning.*

Xing, E. P., Ng, A. Y., Jordan, M. I., & Russell, S. (2003). Distance metric learning, with application to clustering with side-information. *Advances in Neural Information Processing Systems 16.*

A. Appendix

First of all, we present the formal model of the dataset based on which all analysis will be done: the data is assumed to be coming from k disjoint uniform density balls of unequal size in a metric space. The balls are defined in terms of the metric. All data points inside any particular ball are assumed to be in the same cluster, and points from different balls are assumed to be from different clusters. The oracle is assumed to know this model exactly.

Let n be the total number of points under consideration. Let $\{\pi_h\}_{h=1}^k$ be the probabilities of drawing a point randomly from the h-th ball B_h. Without loss of generality, we assume $\pi_1 \leq \pi_2 \leq \cdots \leq \pi_k$. Further, let $1/l \leq \pi_1$. Let m_h be the number of points in the dataset from B_h. Then, $\pi_h = m_h/n$ and $\pi_h \propto V_h$, the volume of B_h, $\forall h$. Now, the number of possible *cannot-links* is $\sum_{\{h,l,h<l\}} m_h m_l$ and the number of *must-links* is $\sum_h \binom{m_h}{2}$. Let $\alpha = \sum_{\{h,l,h<l\}} m_h m_l / \sum_h \binom{m_h}{2}$.

A.1. Analysis of random initialization

In `PCKMeans`, initialization is done using the k largest sized neighborhoods. We argue that within a small number of queries, the probability of getting even a 3-point neighborhood from any cluster is very low. Given Q pairs at random, on average there will be one *must-link* in every $(1+\alpha)$ pairs. Hence, there will be a total of $Q/(1+\alpha)$ *must-link* pairs in the expected behavior. Then, for the h-th cluster, there will be $r_h = \pi_h Q/(1+\alpha) \ll m_h$ *must-link* pairs on average. We focus on a particular cluster B_h on which r_h pairs have been selected at random. We will not get a 3-point neighborhood from B_h if none of the points $\mathbf{x} \in B_h$ gets drawn more than once in the random pair sampling. If the sampling of r_h pairs is replaced by the sampling of $2r_h$ vertices, the probability of getting a vertex twice is increased. Hence, the probability p_h of *not* getting a 3-point neighborhood is lower bounded by the probability of not getting a vertex twice in the vertex sampling setting. So,

$$\begin{aligned} p_h &\geq \sum_{\substack{\sum_l \beta_l = 2r_h \\ \beta_l < 2, \forall l}} \binom{2r_h}{\beta_1 \ \cdots \ \beta_{m_h}} \left(\frac{1}{m_h}\right)^{2r_h} \\ &= 1 \cdot \left(1 - \frac{1}{m_h}\right) \cdot \left(1 - \frac{2}{m_h}\right) \cdots \left(1 - \frac{2r_h - 1}{m_h}\right) \\ &\geq \left(1 - \frac{2r_h}{m_h}\right)^{2r_h} \approx 1 - \frac{4r_h^2}{m_h} = 1 - \frac{4m_h Q^2}{n^2(1+\alpha)^2} \end{aligned}$$

which is close to 1 for small values of Q. Hence, the probability of getting a 3-point neighborhoods is very low. Therefore, the initialization is essentially done by k random draws from a set of approximately $Q/(1+\alpha)$ 2-point neighborhoods. In this setting, the probability of getting exactly one neighborhood from all the clusters is quite low ($k! \prod_{h=1}^k \pi_h \leq k!/k^k$). This results in significant variance in the initializing neighborhoods and explains the initial jitter for the non-active algorithms for low values of Q.

A.2. Analysis of `Explore`

We shall refer to points from the same cluster as having the same color. If the probability of drawing points of different colors is given by $1/l \leq \pi_1 \leq \pi_2 \leq \cdots \leq \pi_k$, then, by an extension of the coupon collector's problem (Motwani & Raghavan, 1995), one can show that points of all colors will be drawn with high probability within $l \ln k + O(l)$ draws. We claim that the farthest first scheme gets points of all colors within l attempts with probability 1.

In the worst case, if the disjoint balls are placed by an adversary, the adversary will try to place the balls such that getting a point from at least one ball is very difficult. Using a packing argument, we show that irrespective of the placement of the balls, the farthest first traversal cannot avoid any particular ball for long. Consider two balls b, B with probabilities π_b, π_B. Let r_b, r_B be the radii of the two balls, and V_b, V_B be the volumes of the two balls. Further, let $\sigma_b(B)$ denote the packing number of B with b balls — the maximum number of disjoint b balls that can be packed inside the ball B. Now, if there are just these two balls in the universe and if farthest-first traversal starts in B, the points obtained from B before entering b must have pairwise distances (between their centers) of at least $2r_b$, because otherwise the traversal would have picked the farthest point from b and got a distance of at least $2r_b$. Hence, the traversal cannot stay in B for more that $\sigma_b(B)$ farthest-first jumps because there are exactly these many points inside B that can be at a distance of at least $2r_b$ from each other. Now, the packing number $\sigma_b(B) \leq V_B/V_b = \pi_B/\pi_b$, the ratio of their probabilities. This argument can be extended to the general case of k balls. In the general case, the number of times the farthest first traversal can continue without entering the ball B_i is

$$n_i \leq \sum_{\substack{h=1 \\ h \neq i}}^k \sigma_{B_i}(B_h) \leq \sum_{\substack{h=1 \\ h \neq i}}^k \pi_h/\pi_i$$

Clearly, this number is largest for the smallest ball B_1. So, the maximum number of farthest-first jumps before reaching B_1 is given by

$$\begin{aligned} n_1 &\leq \sum_{h=2}^k \pi_h/\pi_1 = (1-\pi_1)/\pi_1 \\ &\leq l(1-1/l) = (l-1) \end{aligned}$$

In the next jump, farthest-first gets a point from B_1. Hence, the farthest first traversal will find points of all the colors in $(l-1)+1 = l$ attempts. Note that this is a significant $\ln k$ factor improvement over the random scheme.