PhD Proposal, Nov 2003 (Also Appears as Technical Report, UT-AI-TR-03-307)

# Semi-supervised Clustering: Learning with Limited User Feedback

by

**Sugato Basu**

**Doctoral Dissertation Proposal**

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

**Doctor of Philosophy**

**The University of Texas at Austin**

November 2003

# Semi-supervised Clustering: Learning with Limited User Feedback

Publication No. ______________

Sugato Basu
The University of Texas at Austin, 2003

Supervisor: Dr. Raymond J. Mooney

In many machine learning domains (e.g. text processing, bioinformatics), there is a large supply of unlabeled data but limited labeled data, which 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. In the proposed thesis, our research focus is on semi-supervised clustering, which uses a small amount of supervised data in the form of class labels or pairwise constraints on some examples to aid unsupervised clustering. Semi-supervised clustering can be either search-based, i.e., changes are made to the clustering objective to satisfy user-specified labels/constraints, or similarity-based, i.e., the clustering similarity metric is trained to satisfy the given labels/constraints. Our main goal in the proposed thesis is to study search-based semi-supervised clustering algorithms and apply them to different domains.

In our initial work, we have shown how supervision can be provided to clustering in the form of labeled data points or pairwise constraints. We have also developed an active learning framework for selecting informative constraints in the pairwise constrained semi-supervised clustering model, and proposed a method for unifying search-based and similarity-based techniques in semi-supervised clustering.

In this thesis, we want to study other aspects of semi-supervised clustering. Some of the issues we want to investigate include: (1) effect of noisy, probabilistic or incomplete supervision in clustering; (2) model selection techniques for automatic selection of number of clusters in semi-supervised clustering; (3) ensemble semi-supervised clustering. In our work so far, we have mainly focussed on generative clustering models, e.g. KMeans and EM, and ran experiments on clustering low-dimensional UCI datasets or high-dimensional text datasets. In future, we want to study the effect of semi-supervision on other clustering algorithms, especially in the discriminative clustering and online clustering framework. We also want to study the effectiveness of our semi-supervised clustering algorithms on other domains, e.g., web search engines (clustering of search results), astronomy (clustering of Mars spectral images) and bioinformatics (clustering of gene microarray data).

# Contents

# Chapter 1

# Introduction

Two of the most widely-used methods in machine learning for prediction and data analysis are classification and clustering (Duda, Hart, & Stork, 2001; Mitchell, 1997). Classification is a purely supervised learning model, whereas clustering is completely unsupervised. Recently, there has been a lot of interest in the continuum between completely supervised and unsupervised learning (Muslea, 2002; Nigam, 2001; Ghani, Jones, & Rosenberg, 2003). In this chapter, we will give an overview of traditional supervised classification and unsupervised clustering, and then describe learning in the continuum between these two, where we have partially supervised data. We will then be presenting the main goal of our proposed thesis.

## 1.1 Classification

Classification is a supervised task, where supervision is provided in the form of a set of labeled training data, each data point having a class label selected from a fixed set of classes (Mitchell, 1997). The goal in classification is to learn a function from the training data that gives the best prediction of the class label of unseen (test) data points. Generative models for classification learn the joint distribution of the data and class variables by assuming a particular parametric form of the underlying distribution that generated the data points in each class, and then apply Bayes Rule to obtain class conditional probabilities that are used to predict the class labels for test points drawn from the same distribution, with unknown class labels (Ng & Jordan, 2002). In the discriminative framework, the focus is on learning the discriminant function for the class boundaries or a posterior probability for the class labels directly without learning the underlying generative densities (Jaakkola & Haussler, 1999). It can be shown that the discriminative model of classification has better generalization error than the generative model under certain assumptions (Vapnik, 1998), which has made discriminative classifiers, e.g., support vector machines (Joachims, 1999) and nearest neighbor classifiers (Devroye, Gyorfi, & Lugosi, 1996), very popular for the classification task.

## 1.2 Clustering

Clustering is an unsupervised learning problem, which tries to group a set of points into clusters such that points in the same cluster are more similar to each other than points in different clusters, under a particular similarity metric (Jain & Dubes, 1988). Here, the learning algorithm just observes a set of points without observing any corresponding class/category labels. Clustering problems can also be categorized as generative or discriminative. In the generative clustering model, a parametric form of data generation is assumed, and the goal in the maximum likelihood formulation is to find the parameters that maximize the probability (likelihood) of generation of the data given the model. In the most general formulation, the number of clusters $K$ is also considered to be an unknown parameter. Such a clustering formulation is called a "model selection" framework, since it has to choose the best value of $K$ under which the clustering model fits the data. We will be assuming that $K$ is known in the clustering frameworks that we will be considering, unless explicitly mentioned otherwise. In the discriminative clustering setting (e.g., graph-theoretic clustering), the clustering algorithm tries to cluster the data so as to maximize within-cluster similarity and minimize between-cluster similarity based on a particular similarity metric, where it is not necessary to consider an underlying parametric data generation model. In both the generative and discriminative models, clustering algorithms are generally posed as optimization problems and solved by iterative methods like EM (Dempster, Laird, & Rubin, 1977), approximation algorithms like KMedian (Jain & Vazirani, 2001), or heuristic methods like Metis (Karypis & Kumar, 1998).

## 1.3 Semi-supervised learning

In many practical learning domains (e.g. text processing, bioinformatics), there is a large supply of unlabeled data but limited labeled data, which 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. The framework of semi-supervised learning is applicable to both classification and clustering.

### 1.3.1 Semi-supervised classification

Supervised classification has a known, fixed set of categories, and category-labeled training data is used to induce a classification function. In this setting, the training can also exploit additional unlabeled data, frequently resulting in a more accurate classification function. Several semi-supervised classification algorithms that use unlabeled data to improve classification accuracy have become popular in the past few years, which include co-training (Blum & Mitchell, 1998), transductive support vector machines (Joachims, 1999), and using Expectation Maximization to incorporate unlabeled data into training (Ghahramani & Jordan, 1994; Nigam, McCallum, Thrun, & Mitchell, 2000). Unlabeled data have also been used to learn good metrics in the classification setting (Hastie & Tibshirani, 1996). A good review of semi-supervised classification methods is given in (Seeger, 2000).

### 1.3.2 Semi-supervised clustering

Semi-supervised clustering, which uses class labels or pairwise constraints on some examples to aid unsupervised clustering, has been the focus of several recent projects (Basu, Banerjee, & Mooney, 2002; Klein, Kamvar, & Manning, 2002; Wagstaff, Cardie, Rogers, & Schroedl, 2001; Xing, Ng, Jordan, & Russell, 2003). If the initial labeled data represent all the relevant categories, then both semi-supervised clustering and semi-supervised classification algorithms can be used for categorization. However in many domains, knowledge of the relevant categories is incomplete. Unlike semi-supervised classification, semi-supervised clustering (in the model-selection framework) can group data using the categories in the initial labeled data as well as extend and modify the existing set of categories as needed to reflect other regularities in the data.

Existing methods for semi-supervised clustering fall into two general approaches that we call *search-based* and *similarity-based* methods.

#### Search-based methods

In search-based approaches, the clustering algorithm itself is modified so that user-provided labels or constraints are used to bias the search for an appropriate partitioning. This can be done by several methods, e.g., modifying the clustering objective function so that it includes a term for satisfying specified constraints (Demiriz, Bennett, & Embrechts, 1999), enforcing constraints to be satisfied during the cluster assignment in the clustering process (Wagstaff et al., 2001), doing clustering using side-information from conditional distributions in an auxiliary space (Sinkkonen & Kaski, 2000), and initializing clusters and inferring clustering constraints based on neighborhoods derived from labeled examples (Basu et al., 2002).

#### Similarity-based methods

In similarity-based approaches, an existing clustering algorithm that uses a similarity metric is employed; however, the similarity metric is first trained to satisfy the labels or constraints in the supervised data. Several similarity metrics have been used for similarity-based semi-supervised clustering, including string-edit distance trained using EM (Bilenko & Mooney, 2003), Jensen-Shannon divergence trained using gradient descent (Cohn, Caruana, & McCallum, 2000), Euclidean distance modified by a shortest-path algorithm (Klein et al., 2002), or Mahalanobis distances trained using convex optimization (Hillel, Hertz, Shental, & Weinshall, 2003; Xing et al., 2003). Several clustering algorithms using trained similarity metrics have been employed for semi-supervised clustering, including single-link (Bilenko & Mooney, 2003) and complete-link (Klein et al., 2002) agglomerative clustering, EM (Cohn et al., 2000; Hillel et al., 2003), and KMeans (Hillel et al., 2003; Xing et al., 2003).

However, similarity-based and search-based approaches to semi-supervised clustering have not been adequately compared in previous work, and so their relative strengths and weaknesses are largely unknown. In Section 3.4, we will be presenting a new semi-supervised clustering algorithm that unifies these two approaches.

## 1.4 Goal of proposed thesis

In the proposed thesis, the main goal is to study semi-supervised clustering algorithms, characterize some of their properties and apply them to different domains. In our completed work, we have already shown how supervision can be provided to clustering in the form of labeled data points or pairwise constraints. We have also developed an active learning framework for selecting informative constraints in the pairwise constrained semi-supervised clustering model, and proposed a method for unifying search-based and similarity-based techniques in semi-supervised clustering. Details of the completed work are given in Chapter 3.

In future, we want to look at the following issues, details of which are given in Chapter 4:

- Investigate the effects of noisy supervision, probabilistic supervision (e.g., soft constraints) or incomplete supervision (e.g., labels not specified for all clusters) in clustering;
- Study model selection issues in semi-supervised clustering, which will help to characterize the difference between semi-supervised clustering and classification;
- Study the feasibility of semi-supervising other clustering algorithms, especially in the discriminative clustering or online clustering framework;
- Create a framework for ensemble semi-supervised clustering;
- Apply the semi-supervised clustering model on other domains apart from text, especially web search engines, astronomy and bioinformatics;
- Study the relation between different evaluation metrics used to evaluate semi-supervised clustering;
- Investigate other forms of semi-supervision, e.g., attribute-level constraints;
- Do more theoretical analysis of certain aspects of semi-supervision, especially semi-supervised clustering with labeled data and the unified semi-supervised clustering model.

# Chapter 2

# Background

This chapter gives a brief review of clustering algorithms on which our proposed semi-supervised clustering techniques will be applied. It also gives an overview of different popular clustering evaluation measures, and describes the measures we will be using in our experiments.

## 2.1 Overview of clustering algorithms

As explained in Chapter 1, clustering algorithms can be classified into two models — generative or discriminative. There are other categorizations of clustering, e.g., hierarchical or partitional (Jain, Myrthy, & Flynn, 1999), depending on whether the algorithm clusters the data into a hierarchical structure or gives a flat partitioning of the data.

### 2.1.1 Hierarchical clustering

In hierarchical clustering, the data is not partitioned into clusters in a single step. Instead, a series of partitions take place, which may run from a single cluster containing all objects to $N$ clusters each containing a single object. This gives rise to a hierarchy of clusterings, also known as the cluster dendrogram. Hierarchical clustering can be further categorized as:

- Divisive methods: Create the cluster dendrogram in a top-down divisive fashion, starting with every data point in one cluster and splitting clusters successively according to some measure till a convergence criterion is reached, e.g., Cobweb (Fisher, 1987), recursive cluster-splitting using a statistical transformation (Dubnov, El-Yaniv, Gdalyahu, Schneidman, Tishby, & Yona, 2002), etc.;

- Agglomerative methods: Create the cluster dendrogram in a bottom-up agglomerative fashion, starting with each data point in its own cluster and merging clusters successively according to a similarity measure till a convergence criterion is reached, e.g., hierarchical agglomerative clustering (Kaufman & Rousseeuw, 1990), Birch (Zhang, Ramakrishnan, & Livny, 1996), etc.

### 2.1.2 Partitional clustering

Let $\mathcal{X} = \{x_i\}_{i=1}^N$ be the set of $N$ data-points we want to cluster with each $x_i \in \mathbb{R}^d$. A partitional clustering algorithm divides the data into $K$ partitions ($K$ given as input to the algorithm) by grouping the associated feature vectors into $K$ clusters. Partitional algorithms can be classified as:

- Graph-theoretic: These are discriminative clustering approaches, where an undirected graph $G = (V, E)$ is constructed from the dataset, each vertex in $v_i \in V$ corresponding to a data point $\mathbf{x}_i$ and the weight of each edge $e_{ij} \in E$ corresponding to the similarity between the data points $\mathbf{x}_i$ and $\mathbf{x}_j$ according to a domain-specific similarity measure. The $K$ clustering problem becomes equivalent to finding the $K$-mincut in this graph, which is known to be a NP-complete problem for $K \geq 3$ (Garey & Johnson, 1979). So, most graph-based clustering algorithms try to use good heuristic methods to group nodes so as to find low-cost cuts in $G$. Several different graph-theoretic algorithms have been proposed: methods like Rock (Guha, Rastogi, & Shim, 1999) and Chameleon (Karypis, Han, & Kumar, 1999) group nodes based on the idea of defining neighborhoods using inter-connectivity of nodes in $G$, Metis (Karypis & Kumar, 1998) performs fast multi-level heuristics on $G$ at multiple resolutions to give good partitions, while Opossum (Strehl & Ghosh, 2000) uses a modified cut criterion to ensure that the resulting clusters are well-balanced according to a specified balancing criterion.

- Density-based: These methods model clusters as dense regions and use different heuristics to find arbitrary-shaped high-density regions in the data space and group points accordingly. Well-known methods include Denclue, which tries to analytically model the overall density around a point (Hinneburg & Keim, 1998), and WaveCluster, which uses wavelet-transform to find high-density regions (Sheikholesami, Chatterjee, & Zhang, 1998). Density-based methods typically have difficulty scaling up to very high dimensional data ($>$ 10000 dimensions), which are common in domains like text.

- Mixture-model based: In mixture-model based clustering, the underlying assumption is that each of the $N$ data points $\{\mathbf{x}_i\}_{i=1}^N$ to be clustered are generated by one of $K$ probability distributions $\{p_h\}_{h=1}^K$, where each distribution $p_h$ represents a cluster $C_h$. The probability of observing any point $\mathbf{x}_i$ is given by:

$$\mathbf{Pr}(\mathbf{x}_i|\Theta) = \sum_{i=1}^{K} \alpha_h p_h(\mathbf{x}_i|\theta_h)$$

where $\Theta = (\alpha_1, \cdots, \alpha_K, \theta_1, \cdots, \theta_K)$ is the parameter vector, $\alpha_h$ are the prior probabilities of the clusters ($\sum_{h=1}^K \alpha_h = 1$), and $p_h$ is the probability distribution of cluster $C_h$ parameterized by the set of parameters $\theta_h$. The data-generation process is assumed to be as follows – first, one of the $K$ components is chosen following their prior probability distribution $\{\alpha_h\}_{i=1}^K$; then, a data-point is sampled following the distribution $p_h$ of the chosen component.

Since the cluster assignment of the points are not known, we assume the existence of a random variable $\mathcal{Z}$ that encodes the cluster assignment $z_i$ for each data point $\mathbf{x}_i$. It takes values in $\{h\}_{h=1}^K$ and is always conditioned on the data-point $\mathbf{x}_i$ under consideration. The goal of clustering in this model is to find the estimates of the parameter vector $\Theta$ and the cluster assignment variable $\mathcal{Z}$ such that the log-likelihood of the data:

$$\mathcal{L}(\mathcal{X}, \mathcal{Z}|\Theta) = \sum_{i=i}^{N} \log \mathbf{Pr}(\mathbf{x}_i, \mathcal{Z}|\Theta)$$

is maximized. Since $\mathcal{Z}$ is unknown, the log-likelihood cannot be maximized directly. So, traditional approaches iteratively maximize the *expected* log-likelihood in the Expectation Maximization (EM) framework (Dempster et al., 1977). Starting from an initial estimate of $\Theta$, the EM algorithm iteratively improves the estimates of $\Theta$ and $p(\mathcal{Z}|\mathcal{X},\Theta)$ such that the expected value of the complete-data log-likelihood computed over the class conditional distribution $p(\mathcal{Z}|\mathcal{X},\Theta)$ is maximized. It can be shown that the EM algorithm converges to a local maximum of the expected log-likelihood distribution (Dempster et al., 1977), and the final estimates of the conditional distribution $p(\mathcal{Z}|\mathcal{X},\Theta)$ are used to find the cluster assignments of the points in $\mathcal{X}$.

Most of the work in this area has assumed that the individual mixture density components $p_h$ are Gaussian, and in this case the parameters of the individual Gaussians are estimated by the EM procedure. The popular KMeans clustering algorithm (MacQueen, 1967) can be shown to be an EM algorithm on a mixture of $K$ Gaussians under certain assumptions. Details of this derivation are shown in Section 2.2.1.

## 2.2 Our representative clustering algorithms

In our work, we have chosen KMeans and Hierarchical Agglomerative Clustering as two representative clustering algorithms, from the partitional and hierarchical clustering categories respectively, on which our proposed semi-supervised schemes will be applied. The following sections give brief descriptions of KMeans and Hierarchical Agglomerative Clustering.

### 2.2.1 KMeans and variants

KMeans is a partitional clustering algorithm. It performs iterative relocation to partition a dataset into $K$ clusters, locally minimizing the average squared distance between the data points and the cluster centers. For a set of data points $\mathcal{X} = \{x_i\}_{i=1}^N, x_i \in \mathbb{R}^d$, the KMeans algorithm creates a $K$-partitioning [1] $\{\mathcal{X}_h\}_{h=1}^K$ of $\mathcal{X}$ so that if $\{\mu_h\}_{h=1}^K$ represent the $K$ partition centers, then the following objective function

$$\mathcal{J}_{\text{kmeans}} = \sum_{h=1}^{K} \sum_{x_i \in \mathcal{X}_h} \|x_i - \mu_h\|^2 \tag{2.1}$$

is locally minimized. Note that finding the global optima for the KMeans objective function is an NP-complete problem (Garey, Johnson, & Witsenhausen, 1982). Let $l_i$ be the cluster assignment of the point $\mathbf{x}_i$, where $l_i \in \{h\}_{h=1}^K$. An equivalent form of the KMeans clustering objective function, which we will be using interchangeably, is:

$$\mathcal{J}_{\text{kmeans}} = \sum_{x_i \in \mathcal{X}} \|x_i - \mu_{l_i}\|^2 \tag{2.2}$$

The pseudocode for KMeans is given in Figure 2.1. If we have the additional constraint that the centroids $\{\boldsymbol{\mu}_h\}_{h=1}^K$

**Algorithm:** `KMeans`
**Input:** Set of data points $\mathcal{X} = \{\mathbf{x}_1, \cdots, \mathbf{x}_N\}, \mathbf{x}_i \in \mathbb{R}^d$,
number of clusters $K$
**Output:** Disjoint $K$ partitioning $\{\mathcal{X}_h\}_{h=1}^K$ of $\mathcal{X}$ such that
KMeans objective function is optimized
**Method:**
1. Initialize clusters: Initial centroids $\{\boldsymbol{\mu}_h^{(0)}\}_{i=1}^K$ are selected 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 $h^* = \arg\min_h \|\mathbf{x} - \boldsymbol{\mu}_h^{(t)}\|^2$
2b. `estimate_means`: $\boldsymbol{\mu}_h^{(t+1)} \leftarrow \frac{1}{|\mathcal{X}_h^{(t+1)}|} \sum_{\mathbf{x} \in \mathcal{X}_h^{(t+1)}} \mathbf{x}$
2c. $t \leftarrow (t+1)$

Figure 2.1: KMeans algorithm

are restricted to be selected from $\mathcal{X}$, then the resulting problem is called KMedian clustering. KMedian clustering corresponds to an integer programming problem, for which many approximation algorithms have been proposed (Jain & Vazirani, 2001; Mettu & Plaxton, 2000).

In certain high dimensional data, e.g. text, Euclidean distance is not a good measure of similarity. Certain high dimensional spaces like text have good directional properties, which has made directional similarity measures like $L_2$ normalized dot product (cosine similarity) between the vector representations of text data a popular measure of similarity in the information retrieval community (Baeza-Yates & Ribeiro-Neto, 1999). Note that other similarity measures, e.g., probabilistic document overlap (Goldszmidt & Sahami, 1998), have also been used successfully for text clustering, but we will be focusing on cosine similarity in our work.

Spherical KMeans (SPKMeans) is a version of KMeans that uses cosine similarity as its underlying similarity metric. In the SPKMeans algorithm, standard KMeans is applied to data vectors $\{x_i\}_{i=1}^N$ that have been normalized to have unit $L_2$ norm, so that the data points lie on a unit sphere (Dhillon & Modha, 2001). Note that in SPKMeans, the

[1] $K$ disjoint subsets of $\mathcal{X}$, whose union is $\mathcal{X}$

centroid vectors $\{\mu_h\}_{h=1}^K$ are also constrained to lie on the unit sphere. Assuming $\|x_i\| = \|\mu_h\| = 1, \forall i, h$ in (2.1), we get $\|x_i - \mu_h\|^2 = 2 - 2x_i^T \mu_h$. Then, the clustering problem can be equivalently formulated as that of maximizing the objective function:

$$\mathcal{J}_{\text{spkmeans}} = \sum_{h=1}^{K} \sum_{x_i \in \mathcal{X}_h} x_i^T \mu_h \tag{2.3}$$

The SPKMeans algorithm gives a local maximum of this objective function. The SPKMeans algorithm is computationally efficient for sparse high dimensional data vectors, which are very common in domains like text clustering. For this reason, we have used SPKMeans in our experiments with text data.

Both KMeans and SPKMeans are model-based clustering algorithms, having well-defined underlying generative models. As mentioned earlier, KMeans can be considered as fitting a mixture of Gaussians to a dataset under certain assumptions. The assumptions are that the prior distribution $\{\alpha_h\}_{h=1}^K$ of the Gaussians is uniform, i.e., $\alpha_h = 1/K, \forall h$, and that each Gaussian has identity covariance. Then, the parameter set $\Theta$ in the EM framework consists of just the $K$ means $\{\mu_h\}_{h=1}^K$. With these assumptions, one can show that (Bilmes, 1997):

$$\begin{aligned} \mathbf{E}_{\mathcal{Z}|\mathcal{X},\Theta}[\log \mathbf{Pr}(\mathcal{X}, \mathcal{Z}|\Theta)] &= \sum_{h=1}^{K} \sum_{i=1}^{N} \log(\alpha_h \cdot \frac{1}{(2\pi)^{d/2}} e^{-\|x_i - \mu_h\|^2})\, p(z_h|x_i, \Theta) \\ &= -\sum_{h=1}^{K} \sum_{i=1}^{N} \|x_i - \mu_h\|^2\, p(z_h|x_i, \Theta) + c \end{aligned} \tag{2.4}$$

where $c$ is a constant and $(\mathcal{Z} = h)$ is denoted by $z_h$. Further assuming that

$$p(z_h|x_i, \Theta) = \begin{cases} 1 & \text{if } h = \arg\min_l \|x_i - \mu_l\|^2, \\ 0 & \text{otherwise,} \end{cases} \tag{2.5}$$

and replacing it in (2.4), we note that the expectation term comes out to be the negative of the well-known KMeans objective function with an additive constant.[2] Thus, the problem of maximizing the expected log-likelihood under these assumptions is same as that of minimizing the KMeans objective function. Keeping in mind the assumption in (2.5), the KMeans objective can be written as

$$\mathcal{J}_{\text{kmeans}} = \sum_{h=1}^{K} \sum_{i=1}^{N} \|x_i - \mu_h\|^2\, p(z_h|x_i, \mu_h) \tag{2.6}$$

In a similar fashion, SPKMeans can be considered as fitting a mixture of von Mises-Fisher distributions to a dataset under some assumptions (Banerjee, Dhillon, Ghosh, & Sra, 2003).

In this proposal, we will be comparing our proposed semi-supervised KMeans algorithms to another semi-supervised variant of KMeans, called COP-KMeans (Wagstaff et al., 2001). In COP-KMeans, initial background knowledge, provided in the form of constraints between instances in the dataset, is used in the clustering process. It uses two types of constraints, *must-link* (two instances have to be together in the same cluster) and *cannot-link* (two instances have to be in different clusters). The pseudo-code for COP-KMeans is given in Figure 2.2.

### 2.2.2 Hierarchical Agglomerative Clustering

Hierarchical agglomerative clustering (HAC) is a bottom-up hierarchical clustering algorithm. In HAC, points are initially allocated to singleton clusters, and at each step the "closest" pair of clusters are merged, where closeness is defined according to a similarity measure between clusters. The algorithm generally terminates when the specified "convergence criterion" is reached, which in our case is when the number of current clusters becomes equal to the

[2] The assumption in (2.5) can also be derived by assuming the covariance of the Gaussians to be $\epsilon\mathbb{I}$ and letting $\epsilon \to 0^+$ (Kearns, Mansour, & Ng, 1997).

**Algorithm:** `COP-KMeans`
**Method:**
1. Initialize clusters: Cluster centers are chosen randomly, but as each one is chosen, any must-link constraints that it participates in are enforced (i.e. all items that the chosen instance must link to are assigned to the new cluster, so that they cannot later be chosen as the center of another cluster).
2. Repeat until *convergence*
2a. `assign_cluster`: Assign each data point to the closest cluster such that no must-link or cannot-link constraint is violated by the assignment. If no such assignment exists, **abort**.
2b. `estimate_means`: Update each cluster centroid to be the average of all the points assigned to it.

Figure 2.2: COP-KMeans algorithm

number of clusters desired by the user. Different cluster-level similarity measures are used to determine the closeness between clusters to be merged – single-link, complete-link, or group-average (Manning & Schütze, 1999).

Different HAC schemes have been recently shown to have well-defined underlying generative models – single-link HAC corresponds to the probabilistic model of a mixture of branching random walks, complete-link HAC corresponds to uniform equal-radius hyperspheres, whereas group-average HAC corresponds to equal-variance configurations (Kamvar, Klein, & Manning, 2002). So, the HAC algorithms can be categorized as generative clustering algorithms.

The pseudo-code for HAC is given in Figure 2.3.

**Algorithm:** `Hierarchical Agglomerative Clustering`
**Input:** Set of data points $\mathcal{X} = \{\mathbf{x}_1, \cdots, \mathbf{x}_N\}, \mathbf{x}_i \in \mathbb{R}^d$
**Output:** Dendogram representing hierarchical clustering of $\mathcal{X}$
**Method:**
1. Initialize clusters: Each data point $\mathbf{x}_i$ is placed in its own cluster $C_i$. These clusters form the leaves of the dendogram, and form the set of *current clusters*.
2. Repeat until *convergence*
2a. Merge two *closest* clusters $C_i$ and $C_j$ from the *current clusters* to get cluster $C$
2b. Remove $C_i$ and $C_j$ from *current clusters*, add cluster $C$ to *current clusters*
2c. Add parent links from $C_i$ and $C_j$ to $C$ in the cluster dendogram

Figure 2.3: HAC algorithm

## 2.3 Clustering evaluation measures

Evaluation of the quality of output of clustering algorithms is a difficult problem in general, since there is no "gold-standard" solution in clustering. The commonly used clustering validation measures can be categorized as *internal* or *external*. Internal validation measures, e.g., ratio of average inter-cluster to intra-cluster similarity (higher the better), need only the data and the clustering for their measurement. External validation measures, on the other hand, match the clustering solution to some known prior knowledge, e.g., an underlying class labeling of the data. Many datasets in supervised learning have class information. We can evaluate a clustering algorithm by applying it to such a dataset (with the class label information removed), and then using the class labels of the data as the gold standard against which we can compare the quality of the data clustering obtained.

In our experiments, we have used three metrics for cluster evaluation: *normalized mutual information* (NMI), *pairwise F-measure*, and *objective function*. Of these, normalized mutual information and pairwise F-measure are external clustering validation metrics that estimate the quality of the clustering with respect to a given underlying class labeling of the data.

For clustering algorithms which optimize a particular objective function, we report the value of the objective function when the algorithm converges. For KMeans and SPKMeans, the objective function values reported are $\mathcal{J}_{\text{kmeans}}$ and $\mathcal{J}_{\text{spkmeans}}$ respectively. For the semi-supervised versions of KMeans, we report their corresponding objective function values. Since all the semi-supervised clustering algorithms we propose are iterative methods that (locally) minimize the corresponding clustering objective functions, looking at the objective function value after convergence would give us an idea of whether the semi-supervised algorithm under consideration generated a good clustering.

Normalized mutual information (NMI) 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 compute NMI following the methodology of Strehl et al. (Strehl, Ghosh, & Mooney, 2000). NMI measures how closely the clustering algorithm could reconstruct the underlying label distribution in the data. If $C$ is the random variable denoting the cluster assignments of the points, and $K$ is the random variable denoting the underlying class labels on the points (Banerjee et al., 2003), then the NMI measure is defined as:

$$NMI = \frac{I(C;K)}{(H(C)+H(K))/2}$$

where $I(X;Y) = H(X) - H(X|Y)$ is the mutual information between the random variables $X$ and $Y$, $H(X)$ is the Shannon entropy of $X$, and $H(X|Y)$ is the conditional entropy of $X$ given $Y$.

Pairwise F-measure is defined as the harmonic mean of pairwise precision and recall, the traditional information retrieval measures adapted for evaluating clustering by considering pairs of points. For every pair of points that do not have explicit constraints between them, the decision to cluster this pair into same or different clusters is considered to be correct if it matches with the underlying class labeling available for the points (Bilenko & Mooney, 2002). Pairwise F-measure is defined as:

$$Precision = \frac{\#PairsCorrectlyPredictedInSameCluster}{\#TotalPairsPredictedInSameCluster}$$

$$Recall = \frac{\#PairsCorrectlyPredictedInSameCluster}{\#TotalPairsInSameCluster}$$

$$F{-}measure = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

Pairwise F-measure is a generalization of measures like Rand Index (Klein et al., 2002; Wagstaff et al., 2001; Xing et al., 2003) that are frequently used in other semi-supervised clustering projects. Mutual information (Cover & Thomas, 1991) has also become a popular clustering evaluation metric (Banerjee et al., 2003; Dom, 2001; Fern & Brodley, 2003). Recently, a symmetric cluster evaluation metric based on mutual information has been proposed, which has some useful properties, e.g., it is a true metric in the space of clusterings (Meila, 2003). Interestingly, we empirically observed from some of our experimental results that NMI and pairwise F-measure are highly correlated, an observation which we want to investigate further in future.

Note that the external cluster validation measures we have used (e.g., pairwise F-measure and NMI) are not completely definitive, since the clustering can find a grouping of the data that is different from the underlying class structure. For example, in our initial experiments on clustering articles from the CMU 20 Newsgroups data (where the main Usenet newsgroup to which an article was posted is considered to be its class label), we found one cluster that had articles from four underlying classes — `alt.atheism`, `soc.religion.christian`, `talk.politics.misc`, and `talk.politics.guns`. On closer observation, we noticed that all the articles in the cluster were about the David Koresh episode; this is a valid cluster, albeit different from the grouping suggested by the underlying class labels.

If we had human judges to evaluate the cluster quality, we could find an alternate external cluster validation measure — we could ask the human judges to rank data categorizations generated by humans and the clustering algorithm, and the quality of a clustering output would be considered to be high if the human judges could not reliably discriminate between a human categorization of the data and the grouping generated by the clustering algorithm. Since this is a time- and resource-consuming method of evaluation in the academic setting, we have used automatic external cluster validation methods like pairwise F-measure and NMI in our experiments.

# Chapter 3

# Completed Research

This chapter presents the results of our research so far on semi-supervised clustering. Section 3.1 describes how supervision in the form of labeled data can be incorporated into clustering. Section 3.2 describes a framework for considering pairwise constraints in clustering, while Section 3.3 outlines a method for selecting maximally informative constraints in a query-driven framework for pairwise constrained clustering. Finally, Section 3.4 presents a new semi-supervised clustering approach that unifies search-based and similarity-based techniques (see Section 1.3.2).

## 3.1 Semi-supervised clustering using labeled data

In this section, we give an outline of our initial work where we considered a scenario where supervision is incorporated into clustering in the form of labeled data. We used the labeled data to generate seed clusters that initialize a clustering algorithm, and used constraints generated from the labeled data to guide the clustering process. The underlying intuition is that proper seeding biases clustering towards a good region of the search space, thereby reducing the chances of it getting stuck in poor local optima, while simultaneously producing a clustering similar to the user-specified labels.

The importance of good seeding in clustering is well-known. In partitional clustering algorithms like EM (Dempster et al., 1977) or K-Means (MacQueen, 1967; Selim & Ismail, 1984), some commonly used approaches for initialization include simple random selection, taking the mean of the whole data and randomly perturbing to get initial cluster centers (Dhillon, Fan, & Guan, 2001), or running K smaller clustering problems recursively to initialize K-Means (Duda et al., 2001). Some other interesting initialization methods include the Buckshot method of doing hierarchical clustering on a sample of the data to get an initial set of cluster centers (Manning & Schütze, 1999), running repeated K-Means on multiple data samples and clustering the K-Means solutions to get initial seeds (Fayyad, Reina, & Bradley, 1998), and selecting the K densest intervals along each co-ordinate to get the K cluster centers (Bradley, Mangasarian, & Street, 1997). Our approach is different from these because we use labeled data to get good initialization for clustering.

In the following section, we present the goal of clustering in the presence of labeled data.

### 3.1.1 Problem definition

Given a dataset $\mathcal{X}$, as previously mentioned, KMeans clustering of the dataset generates a $K$-partitioning $\{\mathcal{X}_h\}_{h=1}^K$ of $\mathcal{X}$ so that the KMeans objective is locally minimized. Let $\mathcal{S} \subseteq \mathcal{X}$, called the *seed set*, be the subset of data-points on which supervision is provided as follows: for each $x_i \in \mathcal{S}$, the user provides the cluster $\mathcal{X}_h$ of the partition to which it belongs. We assume that corresponding to each partition $\mathcal{X}_h$ of $\mathcal{X}$, there is typically at least one seedpoint $x_i \in \mathcal{S}$. Note that we get a disjoint $K$-partitioning $\{\mathcal{S}_h\}_{h=1}^K$ of the seed set $\mathcal{S}$, so that all $x_i \in \mathcal{S}_h$ belongs to $\mathcal{X}_h$ according to the supervision. This partitioning of the seed set $\mathcal{S}$ forms the *seed clustering*. The goal is to guide the KMeans algorithm towards the desired clustering of the whole data as illustrated by the seed clustering.

### 3.1.2 Seeded and constrained KMeans algorithms

We propose two algorithms for semi-supervised clustering with labeled data: seeded KMeans (`S-KMeans`) and constrained KMeans (`C-KMeans`).

In `S-KMeans`, the seed clustering is used to initialize the KMeans algorithm. Thus, rather than initializing KMeans from $K$ random means, the centroid of the $h$th cluster is initialized with the centroid of the $h$th partition $\mathcal{S}_h$ of the seed set. The seed clustering is only used for initialization, and the seeds are not used in the following steps of the algorithm. The algorithm is presented in detail in Fig. 3.1.

**Algorithm:** `S-KMeans`
**Input:** Set of data points $\mathcal{X} = \{x_1, \cdots, x_N\}, x_i \in \mathbb{R}^d$, number of clusters $K$, set $\mathcal{S} = \cup_{h=1}^{K} \mathcal{S}_h$ of initial seeds
**Output:** Disjoint $K$ partitioning $\{\mathcal{X}_h\}_{h=1}^{K}$ of $\mathcal{X}$ such that KMeans objective function is optimized
**Method:**
1. `initialize`: $\mu_h^{(0)} \leftarrow \frac{1}{|\mathcal{S}_h|} \sum_{x \in \mathcal{S}_h} x$, for $h = 1, \ldots, K; t \leftarrow 0$
2. Repeat until *convergence*
2a. `assign_cluster`: Assign each data point $x$ to the cluster $h^*$ (i.e. set $\mathcal{X}_{h^*}^{(t+1)}$), for $h^* = \arg\min_h \|x - \mu_h^{(t)}\|^2$
2b. `estimate_means`: $\mu_h^{(t+1)} \leftarrow \frac{1}{|\mathcal{X}_h^{(t+1)}|} \sum_{x \in \mathcal{X}_h^{(t+1)}} x$
2c. $t \leftarrow (t+1)$

Figure 3.1: Seeded KMeans algorithm

In `C-KMeans`, the seed clustering is used to initialize the KMeans algorithm as described for the `S-KMeans` algorithm. However, in the subsequent steps, the cluster memberships of the data points in the seed set are not re-computed in the *assign_cluster* steps of the algorithm – the cluster labels of the seed data are kept unchanged, and only the labels of the non-seed data are re-estimated. The algorithm is given in detail in Fig. 3.2.

**Algorithm:** `C-KMeans`
**Input:** Set of data points $\mathcal{X} = \{\mathbf{x}_1, \cdots, \mathbf{x}_N\}, \mathbf{x}_i \in \mathbb{R}^d$, number of clusters $K$, set $\mathcal{S} = \cup_{h=1}^{K} \mathcal{S}_h$ of initial seeds
**Output:** Disjoint $K$ partitioning $\{\mathcal{X}_h\}_{h=1}^{K}$ of $\mathcal{X}$ such that the KMeans objective function is optimized
**Method:**
1. Initialize clusters: $\boldsymbol{\mu}_h^{(0)} \leftarrow \frac{1}{|\mathcal{S}_h|} \sum_{\mathbf{x} \in \mathcal{S}_h} \mathbf{x}$, for $h = 1, \ldots, K; t \leftarrow 0$
2. Repeat until *convergence*
2a. `assign_cluster`: For $\mathbf{x} \in \mathcal{S}$, if $\mathbf{x} \in \mathcal{S}_h$ assign $\mathbf{x}$ to the cluster $h$ (i.e., set $\mathcal{X}_h^{(t+1)}$). For $\mathbf{x} \notin \mathcal{S}$, assign $\mathbf{x}$ to the cluster $h^*$ (i.e. set $\mathcal{X}_{h^*}^{(t+1)}$), for $h^* = \arg\min_h \|\mathbf{x} - \boldsymbol{\mu}_h^{(t)}\|^2$
2b. `estimate_means`: $\boldsymbol{\mu}_h^{(t+1)} \leftarrow \frac{1}{|\mathcal{X}_h^{(t+1)}|} \sum_{\mathbf{x} \in \mathcal{X}_h^{(t+1)}} \mathbf{x}$
2c. $t \leftarrow (t+1)$

Figure 3.2: Constrained KMeans algorithm

`C-KMeans` seeds the KMeans algorithm with the user-specified labeled data and keeps that labeling unchanged throughout the algorithm. In `S-KMeans`, the user-specified labeling of the seed data may be changed in the course of the algorithm. `C-KMeans` is appropriate when the initial seed labeling is noise-free, or if the user does not want the labels on the seed data to change, whereas `S-KMeans` is appropriate in the presence of noisy seeds.

### 3.1.3 Motivation of semi-supervised KMeans algorithms

The two semi-supervised KMeans algorithms presented in the last section can be motivated by considering KMeans in the EM framework, as shown in Section 2.1. The only "missing data" for the KMeans problem are the conditional distributions of the cluster labels given the points and the parameters, i.e., $p(z_h|x_i, \mu_h)$. Knowledge of these distributions solves the clustering problem, but normally there is no way to compute it. In the semi-supervised clustering framework, the user provides information about some of the data points that specifies the corresponding conditional distributions. Thus, semi-supervision by providing labeled data is equivalent to providing information about the conditional distributions $p(z_h|x_i, \mu_h)$.

In standard KMeans without any initial supervision, the $K$ means are chosen randomly in the initial M-step and the data-points are assigned to the nearest means in the subsequent E-step. As explained above, every point $x_i$ in the dataset has $K$ possible conditional distributions associated with it (each satisfying (2.5)) corresponding to the $K$ means to which it can belong. This assignment of data point $x_i$ to a random cluster in the first E-step is similar to picking one conditional distribution at random from the $K$ possible conditional distributions.

In `S-KMeans`, the initial supervision is equivalent to specifying the conditional distributions $p(z_h|x_i, \mu_h)$ for the seed points $x_i \in \mathcal{S}$. The specified conditional distributions of the seed data are just used in the initial M-step of the algorithm, and $p(z_h|x_i, \mu_h)$ is re-estimated for all $x_i \in \mathcal{X}$ in the following E-steps of the algorithm.

In `C-KMeans`, the initial M-step is same as `S-KMeans`. The difference is that for the seed data points, the initial labels, i.e., the conditional distributions $p(z_h|x_i, \mu_h)$, are kept unchanged throughout the algorithm, whereas the conditional distribution for the non-seed points are re-estimated at every E-step.

Note that in the `SPKMeans` framework (Sec. 2.2.1), since every point lies on the unit sphere so that $||x_i|| = ||\mu_h|| = 1$, the expectation term in (2.4) becomes equivalent to

$$\mathbf{E}_{\mathcal{Z}|\mathcal{X},\Theta}[\log p(\mathcal{X}, \mathcal{Z}|\Theta)] = \sum_{h=1}^{K}\sum_{i=1}^{N} x_i^T \mu_h \, p(z_h|x_i, \Theta) + c$$

So, maximizing the `SPKMeans` objective function is equivalent to maximizing the expectation of the complete-data log-likelihood in the E-step of the EM algorithm.

It can also be shown that getting good seeding is very essential for centroid-based clustering algorithms like KMeans. As shown in Section 2.2.1, under certain generative model-based assumptions, one can connect the mixture of Gaussians model to the KMeans model. A direct calculation using Chernoff bounds shows that if a particular cluster (with an underlying Gaussian model) with true centroid $\boldsymbol{\mu}$ is seeded with $m$ points (drawn independently at random from the corresponding Gaussian distribution) and the estimated centroid is $\hat{\boldsymbol{\mu}}$, then

$$\mathbf{Pr}(|\hat{\boldsymbol{\mu}} - \boldsymbol{\mu}| \geq \delta) \leq e^{-\delta^2 m/2} \tag{3.1}$$

where $\delta \in \mathbb{R}+$ (Banerjee, 2001). Thus, the probability of deviation of the centroid estimates falls exponentially with the number of seeds, and hence seeding results in good initial centroids.

### 3.1.4 Experimental results

Our experiments demonstrated the advantages of `S-KMeans` and `C-KMeans` over standard random seeding and `COP-KMeans` (Wagstaff et al., 2001), a previously developed semi-supervised clustering algorithm described in Section 2.2.1. Details of the experimental results can be found in (Basu et al., 2002).

#### Datasets

We are showing results of our experiments on 2 high-dimensional text data sets – a subset of CMU 20 Newsgroups (2000 documents, 21631 words) and Yahoo! News K-series (2340 documents, 12229 words) datasets. Both datasets had 20 classes, and the clustering algorithms were asked to generate the same number of clusters. For each dataset, we ran 4 clustering algorithms – `S-KMeans`, `C-KMeans`, `COP-KMeans`, and random KMeans. In random KMeans, the $K$ means were initialized by taking the mean of the entire data and randomly perturbing it $K$ times (Fayyad et al., 1998). This technique of initialization has given good results in unsupervised KMeans in previous work (Dhillon et al., 2001). We compared the performance of these 4 methods on the 2 datasets with varying seeding and noise levels, using 10-fold cross validation. For each dataset, SPKMeans was used as the underlying KMeans algorithm for all the 4 KMeans variants.

### Learning curves with cross validation

For all the algorithms, on each dataset, we generated learning curves with 10-fold cross-validation. For studying the effect of seeding, 10% of the dataset was set aside as the test set at any particular fold. The training sets at different points of the learning curve were obtained from the remaining 90% of the data by varying the seed fraction from 0.0 to 1.0 in steps of 0.1, and the results at each point on the learning curve were obtained by averaging over 10 folds. The clustering algorithm was run on the whole dataset, but the evaluation metrics (objective function and NMI measure) were calculated only on the test set. For studying the effects of noise in the seeding, learning curves were generated by keeping a fixed fraction of seeding and varying the noise fraction.

### Results

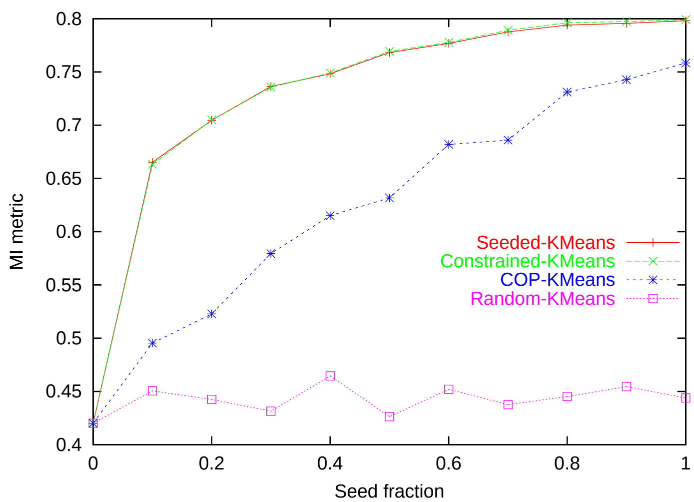


Figure 3.3: Comparison of NMI values on the reduced 20 Newsgroup dataset with increasing seed fraction, with noise fraction = 0

The semi-supervised algorithms (`S-KMeans` and `C-KMeans`) performed better than the unsupervised algorithm (random KMeans) in terms of the NMI evaluation measure, as shown in Figs. 3.3 and 3.4. For a balanced datasets (e.g., reduced 20 Newsgroups), the NMI value of the `S-KMeans` algorithm (see Figure 3.3) seems to increase exponentially with increasing amount of labeled data along the learning curve, a phenomenon which we will explain later. Similar results were also obtained for objective function. Both `S-KMeans` and `C-KMeans` performed better than `COP-KMeans` in most cases, when the seeds have no noise.

We also performed experiments where we simulated noise in the seed data by changing the labels of a fraction of the seed examples to a random incorrect value. The results in Figure 3.5 show that `S-KMeans` is quite robust against noisy seeding, since it takes the benefit of initialization using the supervised data but does not enforce the noisy constraints to be satisfied during every clustering iteration (like `C-KMeans` and `COP-KMeans` do).

We also ran initial experiments with *incomplete* seeding, where seeds are not specified for every cluster – from Fig. 3.6, it can be seen that the NMI metric did not decrease substantially with increase in the number of unseeded categories, showing that the semi-supervised clustering algorithms could extend the seed clusters and generate more clusters, in order to fit the regularity of the data.

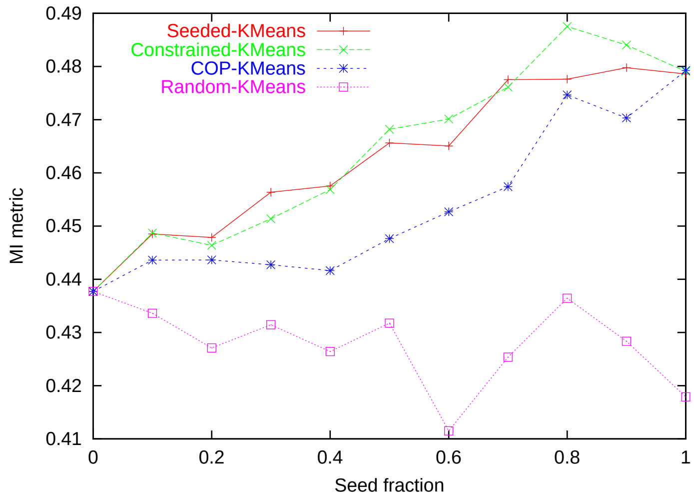


Figure 3.4: Comparison of NMI values on the Yahoo! News dataset with increasing seed fraction, with noise fraction = 0

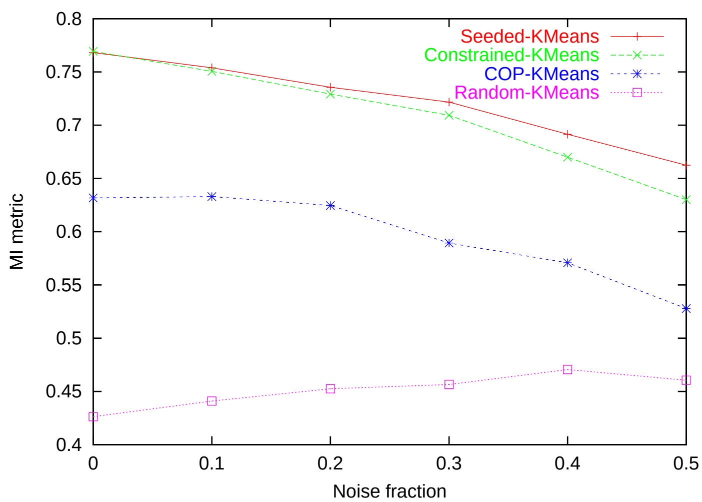


Figure 3.5: Comparison of NMI values on the reduced 20 Newsgroup dataset with increasing noise fraction, with seed fraction = 0.5

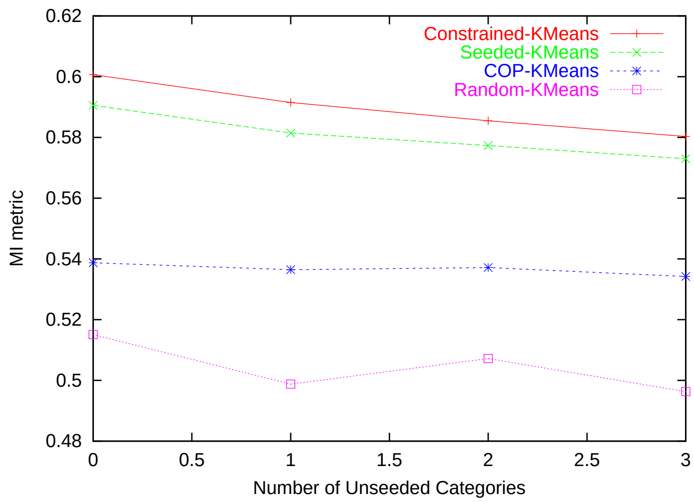


Figure 3.6: Comparison of NMI values on the full 20 Newsgroups data with increasing incompleteness fraction, with seed fraction = 0.1

Further experimental results on other datasets can be found in (Basu et al., 2002).

## 3.2 Semi-supervised clustering using pairwise constraints

In this work, we considered a framework that has pairwise *must-link* and *cannot-link* constraints between points in a dataset (with an associated cost of violating each constraint), in addition to having distances between the points. These constraints specify that two examples must be in the same cluster (must-link) or different clusters (cannot-link) (Wagstaff et al., 2001). In real-world unsupervised learning tasks, e.g., clustering for speaker identification in a conversation (Hillel et al., 2003), visual correspondence in multi-view image processing (Boykov, Veksler, & Zabih, 1998), clustering multi-spectral information from Mars images (Wagstaff, 2002), etc., considering supervision in the form of constraints is generally more practical than providing class labels, since true labels may be unknown a priori, while it can be easier for a user to specify whether pairs of points belong to the same cluster or different clusters. Constraints are a more general way to provide supervision in clustering than labels — given a set of labeled points one can always infer an unique equivalent set of pairwise must-link and cannot-link constraints, but not vice versa.

We proposed a cost function for *pairwise constrained clustering* (PCC) that can be shown to be the energy of a configuration of a Markov random field (MRF) over the data with a well-defined potential function and noise model. Then, the pairwise-constrained clustering problem becomes equivalent to finding the MRF configuration with the highest probability, or, in other words, minimizing its energy. We developed an iterative KMeans-type algorithm for solving this problem. Previous work in the PCC framework include the hard-constrained `COP-KMeans` algorithm (Wagstaff et al., 2001) and the soft-constrained `SCOP-KMeans` algorithm (Wagstaff, 2002), which have heuristically motivated objective functions. Our formulation, on the other hand, has a well-defined underlying generative model. Bansal et al. (Bansal, Blum, & Chawla, 2002) also proposed a theoretical model for pairwise constrained clustering, but their clustering model uses only pairwise constraints for clustering, whereas our formulation uses both constraints and an underlying distance metric. Pairwise clustering models have also been proposed for other non-parametric clustering algorithms (Dubnov et al., 2002).

### 3.2.1 Problem definition

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 points and the cluster centroids is (locally) minimized. We introduce a framework for *pairwise constrained clustering* (PCC) that has pairwise must-link and cannot-link constraints (Wagstaff et al., 2001) between a subset of points in the dataset (with a cost of violating each constraint), in addition to distances between points. Since centroid-based clustering cannot handle pairwise constraints explicitly, we formulate the goal of clustering in the PCC framework as minimizing a combined objective function: the sum of the total distance between the points and their cluster centroids and the cost of violating the pairwise constraints.

For the PCC 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. Note that the tuples in $\mathcal{M}$ and $\mathcal{C}$ are order-independent, i.e., $(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{C} \Rightarrow (\mathbf{x}_j, \mathbf{x}_i) \in \mathcal{C}$, and so also for $\mathcal{M}$. 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 $l_i \in \{h\}_{h=1}^K$. Let $d_M$ and $d_C$ be two metrics that quantify the cost of violating must-link and cannot-link constraints (Kleinberg & Tardos, 1999). We restrict our attention to $d_M(l_i, l_j) = \mathbb{1}[l_i \neq l_j]$ 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 PCC 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 $\boldsymbol{\mu}_{l_i}$:

$$\mathcal{J}_{\text{pckmeans}} = \frac{1}{2} \sum_{\mathbf{x}_i \in \mathcal{X}} \|\mathbf{x}_i - \boldsymbol{\mu}_{l_i}\|^2 + \sum_{(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{M}} w_{ij} \mathbb{1}[l_i \neq l_j] + \sum_{(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{C}} \overline{w}_{ij} \mathbb{1}[l_i = l_j] \tag{3.2}$$

This objective function tries to minimize the total distance between points and their cluster centroids such that the minimum number of specified constraints between the points are violated. The goal of any clustering algorithm in the PCC framework is to minimize this combined objective function.

### 3.2.2 Pairwise constrained KMeans 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 $K$, we propose an algorithm `PC-KMeans` that 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 $\mathcal{J}_{\text{pckmeans}}$ is (locally) minimized.

In the initialization step of `PC-KMeans`, assuming consistency of constraints, 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 $\lambda$, which are used to create $\lambda$ neighborhood sets $\{N_p\}_{p=1}^{\lambda}$. 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, again assuming consistency of constraints. From now on, we will 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 $\lambda$ neighborhood sets $\{N_p\}_{p=1}^{\lambda}$, which are used to initialize the cluster centroids.

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 $\lambda$ cluster centers with the centroids of the $\lambda$ 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 perturbations of the global centroid of $\mathcal{X}$.

The algorithm `PC-KMeans` alternates between the cluster assignment and centroid estimation steps (see Figure 3.7). In the cluster assignment step of `PC-KMeans`, 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

**Algorithm:** PC-KMeans
**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 $\lambda$ 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 $h^* = \arg\min_h(\frac{1}{2}\|\mathbf{x} - \boldsymbol{\mu}_h^{(t)}\|^2$
$+w \sum_{(\mathbf{x},\mathbf{x}_j)\in\mathcal{M}} \mathbb{1}[h \neq l_j] + w \sum_{(\mathbf{x},\mathbf{x}_j)\in\mathcal{C}} \mathbb{1}[h = l_j])$
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 3.7: Pairwise Constrained KMeans algorithm

assignment (by equivalently satisfying as many must-links and cannot-links as given by the assignment). The centroid re-estimation step is the same as KMeans. It is easy to show that the objective function decreases after every cluster assignment and re-estimation step until convergence, implying that the `PC-KMeans` algorithm will converge to a local minima of $\mathcal{J}_{\text{pckmeans}}$. The proof is shown in detail in (Basu, Banerjee, & Mooney, 2003a).

### 3.2.3 Motivation of the PCC framework

The mathematical formulation of the PCC framework was motivated by the *metric labeling* problem and the *generalized Potts* model (Boykov et al., 1998; Kleinberg & Tardos, 1999), for which Kleinberg et al. (Kleinberg & Tardos, 1999) proposed an approximation algorithm. Their formulation only considers the set $\mathcal{M}$ of must-link constraints, which we extended to the PCC framework by adding the set $\mathcal{C}$ of cannot-link constraints. Our proposed pairwise constrained KMeans (`PC-KMeans`) algorithm greedily optimizes $\mathcal{J}_{\text{pckmeans}}$ using a KMeans-type iteration with a modified cluster-assignment step.

The PCC formulation can be motivated by considering a Markov Random Field (MRF) (Boykov et al., 1998) defined over $\mathcal{X}$ such that the field (or set) $F = \{F_i\}_{i=1}^N$ of random variables over $\mathcal{X}$ can take values $\{l_i\}_{i=1}^N$ with $l_i \in \{h\}_{h=1}^K, \forall i$. Let a configuration $l$ denote the joint event $\{F = l\} = \{F_i = l_i\}_{i=1}^N$. Restricting the model to MRFs whose clique potentials involve pairwise points, the prior probability of a configuration $l$ is $P(l) \propto \exp(-\sum_i \sum_j V_{(i,j)}(l_i, l_j))$, where

$$V_{(i,j)}(l_i, l_j) = \begin{cases} w_{ij}\mathbb{1}[l_i \neq l_j] & \text{if } (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{M} \\ \overline{w}_{ij}\mathbb{1}[l_i = l_j] & \text{if } (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{C} \\ 0 & \text{otherwise} \end{cases}$$

Assuming an identity covariance Gaussian noise model for the observed data (von-Mises Fisher distribution (Mardia & Jupp, 2000) was considered as the noise model for high-dimensional text data)[1] and assuming the observed data has been drawn independently, if $\{\boldsymbol{\mu}_h\}_{h=1}^K$ denote the true representatives corresponding to the labels $\{h\}_{h=1}^K$, the conditional probability of the observation $\mathcal{X}$ for a given configuration $l$ is $P(\mathcal{X}|l) \propto \exp(-\frac{1}{2}\sum_{\mathbf{x}_i} \|\mathbf{x}_i - \boldsymbol{\mu}_{l_i}\|^2)$. Then, since the posterior probability of a configuration $l$ is $P(l|\mathcal{X}) = P(l)P(\mathcal{X}|l)$, finding the MAP configuration boils down to minimizing the energy of the configuration that is exactly as given by (3.2). Hence, the PCC objective function is really the energy function over the specified MRF.

### 3.2.4 Experimental results

The experimental results showing the improved performance of the PCC clustering framework over unsupervised clustering are shown in Section 3.3.4, where we show the benefit of the PCC framework as well as the usefulness of selecting pairwise constraints using our proposed active learning algorithm.

## 3.3 Active learning for semi-supervised clustering

In order to maximize the utility of the limited supervised data available in a semi-supervised setting, supervised training examples should be *actively* selected as maximally informative ones rather than chosen at random, if possible (McCallum & Nigam, 1998). In the PCC framework, this would imply that fewer constraints will be required to significantly improve the clustering accuracy. To this end, we developed a new method for actively selecting good pairwise constraints for semi-supervised clustering in the PCC framework.

Previous work in active learning has been mostly restricted to classification, where different principles of query selection have been studied, e.g., reduction of the version space size (Freund, Seung, Shamir, & Tishby, 1997), reduction of uncertainty in predicted label (Lewis & Gale, 1994), maximizing the margin on training data (Abe & Mamitsuka, 1998), and finding high variance data points by density-weighted pool-based sampling (McCallum & Nigam, 1998). However, active learning techniques in classification are not applicable in the clustering framework, since the basic underlying concept of reduction of classification error and variance over the distribution of examples (Cohn, Ghahramani, & Jordan, 1996) is not well-defined for clustering. In the unsupervised setting, Hofmann et al. (Hofmann & Buhmann, 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, using expected value of information estimated from the existing data, such that the risk of making wrong estimates about the true underlying clustering from the existing incomplete data is minimized. In contrast, our model assumes that we have complete similarity information between all pairs of points, along with pairwise constraints whose violation cost is a component of the objective function (3.2), and the active learning goal is to select pairwise constraints which are most informative about the underlying clustering. Klein et al. (Klein et al., 2002) also consider active learning in semi-supervised clustering, but instead of making example-level queries they make cluster level queries, i.e., 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.

### 3.3.1 Problem definition

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 a constant number of queries to the oracle.[2] Given a fixed number of queries that can be made to the oracle, the goal is to select the $\mathcal{M}$ and $\mathcal{C}$ sets such that the `PC-KMeans` clustering algorithm, using those pairwise constraints, can get to a local optimum of the objective function $\mathcal{J}_{\text{pckmeans}}$ which is better than the local optima it would reach with the same number of randomly chosen constraints.

[1] The framework can be shown to hold for arbitrary exponential noise models.

[2] The oracle can also give a *don't-know* response to a query, in which case that response is ignored (pair not considered as a constraint) and that query is not posed again later.

### 3.3.2 `Explore` and `Consolidate` algorithms

The proposed active learning scheme has two phases. The first phase explores the given data to get $K$ pairwise disjoint non-null neighborhoods, each belonging to a different cluster in the underlying clustering of the data, 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. For this phase, we propose an algorithm `Explore` that essentially uses the *farthest-first* scheme (Dasgupta, 2002; Hochbaum & Shmoys, 1985) to form appropriate queries for getting the required pairwise disjoint neighborhoods. For our data model, we prove another interesting property of farthest-first traversal (see (Basu et al., 2003a) for more details) that justifies its use for active learning.

At the end of `Explore`, at least one point has been obtained per cluster. The remaining queries are used to consolidate this structure. The cluster skeleton obtained from `Explore` is used to initialize $K$ pairwise disjoint non-null neighborhoods $\{N_p\}_{p=1}^{K}$. Then, given any point $\mathbf{x}$ not in any of the existing neighborhoods, we will have to ask at most $(K-1)$ queries by pairing $\mathbf{x}$ up with a member from each of the disjoint neighborhoods $N_p$ to find out the neighborhood to which $\mathbf{x}$ belongs. Note that it is practical to sort the neighborhoods in increasing order of the distance of their centroids from $\mathbf{x}$ so that the correct must-link neighborhood for $\mathbf{x}$ is encountered sooner in the querying process. This principle forms the second phase of our active learning algorithm, and we call the algorithm for this phase `Consolidate`. In this phase, we are able to get the correct cluster label of $\mathbf{x}$ by asking at most $(K-1)$ queries. So, $(K-1)$ *pairwise labels are equivalent to a single pointwise label* in the worst case, when $K$ is known.

The details of the algorithms for performing the exploration and the consolidation phases are given in Figures 3.8 and 3.9.

**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 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 $N_\lambda$ with $\mathbf{x}$
   else
      add $\mathbf{x}$ to the neighborhood with which it is must-linked

Figure 3.8: Algorithm `Explore`

### 3.3.3 Motivation of active constraint selection algorithm

In Section 3.1.3, 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 `PC-KMeans` is initialized from a very good set of centroids.

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 (Basu et al., 2003a) for details).

**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 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 sorted order till a must-link is obtained, add $\mathbf{x}$ to that neighborhood

Figure 3.9: Algorithm `Consolidate`

Both `Explore` and `Consolidate` add points to the clusters at a good rate. It can be shown using results proved in (Basu et al., 2003a) that the `Explore` phase gets at least one point from each of the $K$ underlying clusters in maximum $K\binom{K}{2}$ queries. When the active scheme has finished running `Explore` and is running `Consolidate`, it can also be shown using a generalization of the coupon collector's problem (Motwani & Raghavan, 1995) that with high probability it will get one new point from each cluster in approximately $K^2 \log K$ queries. `Consolidate` therefore adds points to clusters at a faster rate than `Explore` by a factor of $\mathcal{O}(\frac{K}{\log K})$.

### 3.3.4 Experimental results

In this section, we present our experiments with active selection of constraints in the PCC framework.

#### Datasets

We ran experiments on both high-dimensional text datasets and low-dimensional UCI (Blake & Merz, 1998) datasets. Here we present the results on a subset of the text data Classic3 (Dhillon & Modha, 2001), containing 400 documents – 100 *Cranfield* documents from aeronautical system papers, 100 *Medline* documents from medical journals, and 200 *Cisi* documents from information retrieval papers. This Classic3-subset dataset was specifically designed to create clusters of unequal size, and has 400 points in 2897 dimensions after standard pre-processing steps like stop-word removal, tf-idf weighting, and removal of very high-frequency and very low-frequency words (Dhillon & Modha, 2001). From the UCI collection we selected Iris, which is a well-known dataset having 150 points in 4 dimensions. We used the active pairwise constrained version of `KMeans` on Iris, and `SPKMeans` on Classic3-subset.

#### Learning curves with cross validation

For all algorithms on each dataset, we generated learning curves with 10-fold cross-validation. The same methodology was used to generate the learning curves as in Section 3.1.4, except for that fact that here the x-axis represents the number of pairwise constraints given as input to the algorithms. For non-active `PCKMeans` the pairwise constraints were selected at random, while for active `PCKMeans` the pairwise constraints were selected using our active learning scheme.

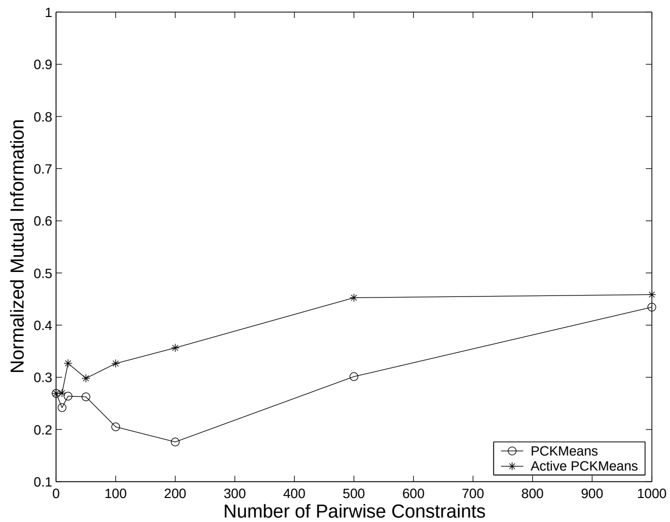


Figure 3.10: Comparison of NMI values on `Classic400`

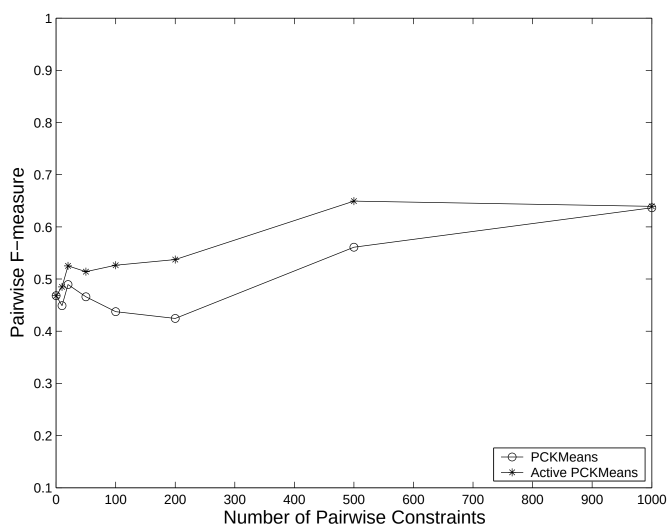


Figure 3.11: Comparison of pairwise F-measure values on `Classic400`

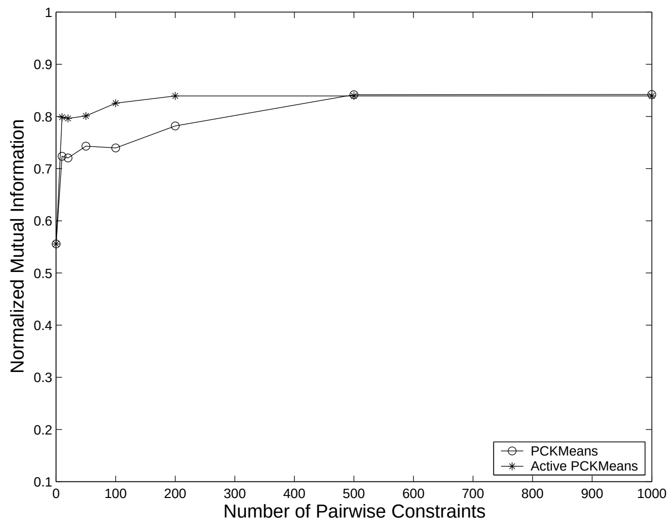


Figure 3.12: Comparison of NMI values on `Iris`

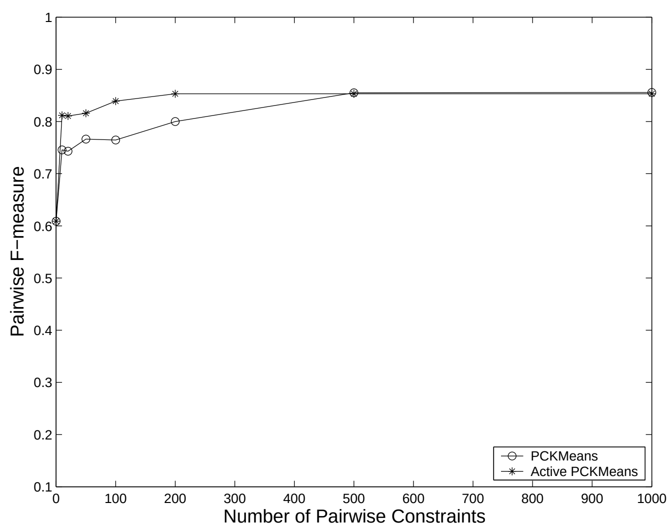


Figure 3.13: Comparison of pairwise F-measure values on `Iris`

**Results**

In the domains that we considered, 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 (3.2). The $w$ parameter acts as a tuning knob, giving us the continuum between a `S-KMeans`-like algorithm on one extreme ($w$ = 0), where there is no guarantee of the constraint satisfaction in the clustering, and a `C-KMeans`-like algorithm on the other extreme ($w = \infty$), where the clustering process is forced to respect all the given constraints. The parameter $w$ can be chosen either by cross-validation on a held-out set, or by the user according to the degree of confidence in the constraints. We chose an intermediate value of $w$ in (3.2) so that the algorithms gives a tradeoff between minimizing the total distance between points and cluster centroids and the cost of violating the constraints.

Figures 3.11-3.13 show that (1) semi-supervised clustering with constraints, for both active and non-active `PCKMeans`, performs considerably better than unsupervised KMeans clustering for the datasets we considered, since the clustering evaluation measures (NMI and pairwise F-measure) improve with increasing number of pairwise constraints along the learning curve. and (2) active selection of pairwise constraints in active `PCKMeans`, using our two-phase active learning algorithm, significantly outperforms random selection of constraints in non-active `PCKMeans`. Detailed discussions and results on other datasets can be found in (Basu et al., 2003a).

## 3.4 Unified model of semi-supervised clustering

In previous work, similarity-based and search-based approaches to semi-supervised clustering have not been adequately compared experimentally, so their relative strengths and weaknesses are largely unknown. Also, the two approaches are not incompatible, therefore, applying a search-based approach with a trained similarity metric is clearly an additional option which may have advantages over both existing approaches. In this work, we presented a new unified semi-supervised clustering algorithm derived from KMeans that incorporates *both* metric learning and using labeled data as seeds and/or constraints.

### 3.4.1 Problem definition

Following previous work (Xing et al., 2003), we can parameterize Euclidean distance with a symmetric positive-definite weight matrix $\mathbf{A}$ as follows: $\|\mathbf{x}_i - \mathbf{x}_j\|_{\mathbf{A}} = \sqrt{(\mathbf{x}_i - \boldsymbol{\mu}_{l_i})^T \mathbf{A} (\mathbf{x}_i - \boldsymbol{\mu}_{l_i})}$. If $\mathbf{A}$ is restricted to be a diagonal matrix, then it scales each axis by a different weight and corresponds to feature weighting; otherwise new features are created that are linear combinations of the original features. In our clustering formulation, using the matrix $\mathbf{A}$ is equivalent to considering a generalized version of the KMeans model described in Section 2.2.1, where all the Gaussians have a covariance matrix $\mathbf{A}^{-1}$ (Bilmes, 1997).

It can be easily shown that maximizing the complete data log-likelihood under this generalized KMeans model is equivalent to minimizing the objective function:

$$\mathcal{J}_{\text{mkmeans}} = \sum_{\mathbf{x}_i \in \mathcal{X}} \|\mathbf{x}_i - \boldsymbol{\mu}_{l_i}\|^2_{\mathbf{A}} - \log(\det(\mathbf{A})) \tag{3.3}$$

where the second term arises due to the normalizing constant of a Gaussian with covariance matrix $\mathbf{A}^{-1}$.

We combine objective functions (3.2) and (3.3) to get the following objective function that attempts to minimize cluster dispersion under a learned metric along with minimizing the number of constraint violations:

$$\begin{aligned} \mathcal{J}_{\text{mpckm}} &= \sum_{\mathbf{x}_i \in \mathcal{X}} \|\mathbf{x}_i - \boldsymbol{\mu}_{l_i}\|^2_{\mathbf{A}} - \log(\det(\mathbf{A})) + \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{M}} w_{ij} f_M(\mathbf{x}_i,\mathbf{x}_j) \mathbb{1}[l_i \neq l_j] \\ &+ \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{C}} \overline{w}_{ij} f_C(\mathbf{x}_i,\mathbf{x}_j) \mathbb{1}[l_i = l_j] \end{aligned} \tag{3.4}$$

where $f_M$ and $f_C$ are functions (defined in the next section) that specify the "seriousness" of violating a must-link or cannot-link constraint. The weights $w_{ij}$ and $\overline{w}_{ij}$ provide a way to specify the relative importance of the unlabeled versus labeled data while allowing individual constraint weights. This objective function $\mathcal{J}_{\text{mpckm}}$ is greedily optimized by our proposed metric pairwise constrained KMeans (`MPC-KMeans`) algorithm that uses a KMeans-type iteration.

### 3.4.2 Motivation of the unified framework

If we do not scale the weights $w_{ij}$ and $\overline{w}_{ij}$ by the functions $f_M$ and $f_C$ in (3.4), one problem with the resulting combined objective function would be that all constraint violations are treated equally. However, the cost of violating a must-link constraint between two *close* points should be higher than the cost of violating a must-link constraint between two points that are *far apart*. Such cost assignment reflects the intuition that it is a "worse error" to violate a must-link constraint between similar points, and such an error should have more impact on the metric learning framework. Multiplying the weights $w_{ij}$ with the penalty function $f_M(\mathbf{x}_i, \mathbf{x}_j) = \max(\alpha_{\min}, \alpha_{\max} - \|\mathbf{x}_i - \mathbf{x}_j\|^2_{\mathbf{A}})$ gives us the overall cost of violating a must-link constraint between two points $\mathbf{x}_i$ and $\mathbf{x}_j$, where $\alpha_{\min}$ and $\alpha_{\max}$ are non-negative constants that correspond to minimum and maximum penalties respectively. They can be set as fractions of the square of the maximum must-link distance $\max_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{M}} \|\mathbf{x}_i - \mathbf{x}_j\|^2$, thus guaranteeing that the penalty for violating a constraint is always positive. Overall, this formulation enables the penalty for violating a must-link constraint to be proportional to the "seriousness" of the violation.

Analogously, the cost of violating a cannot-link constraint between two *distant* points should be higher than the cost of violating a cannot-link constraint between points that are *close*, since the former is a "worse error". Multiplying weights $\overline{w}_{ij}$ with $f_C(\mathbf{x}_i, \mathbf{x}_j) = \min(\alpha_{\min} + \|\mathbf{x}_i - \mathbf{x}_j\|^2_{\mathbf{A}}, \alpha_{\max})$ allows us to take the "seriousness" of the constraint violation into account.

### 3.4.3 `MPC-KMeans` algorithm for the unified framework

Given a set of data points $\mathcal{X}$, a set of must-link constraints $\mathcal{M}$, a set of cannot-link constraints $\mathcal{C}$, corresponding weight sets $W$ and $\overline{W}$, and the number of clusters to form $K$, metric pairwise constrained KMeans (`MPC-KMeans`) 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 $\mathcal{J}_{\text{mpckm}}$ is (locally) minimized.

The algorithm `MPC-KMeans` has two components. Utilizing constraints during cluster initialization and satisfaction of the constraints during every cluster assignment step constitutes the search-based component of the algorithm. Learning the distance metric by re-estimating the weight matrix **A** during each algorithm iteration based on current constraint violations is the similarity-based component.

Intuitively, the search-based technique uses the pairwise constraints to generate seed clusters that initialize the clustering algorithm, and also uses the constraints to guide the clustering process through the iterations. Seeds inferred from the constraints bias the clustering towards a good region of the search space, thereby possibly reducing the chances of it getting stuck in poor local optima, while a clustering that satisfies the user-specified constraints is produced simultaneously.

The similarity-based technique distorts the metric space to minimize the costs of violated constraints, possibly removing the violations in the subsequent iterations. Implicitly, the space where data points are embedded is transformed to respect the user-provided constraints, thus capturing the notion of similarity appropriate for the dataset from the user's perspective.

**Initialization:** The `MPC-KMeans` algorithm is initialized from neighborhoods inferred from the $\mathcal{M}$ and $\mathcal{C}$ sets, in the same way as the `PC-KMeans` algorithm (see Section 3.2.2).

**E-step:** `MPC-KMeans` alternates between cluster assignment in the E-step, and centroid estimation and metric learning in the M-step (see Figure 3.14).

In the E-step of `MPC-KMeans`, every point $\mathbf{x}$ is assigned to a cluster so that the sum of the distance of $\mathbf{x}$ to the cluster centroid and the cost of constraint violations possibly incurred by this cluster assignment is minimized. Note that this 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. In the cluster assignment step, each point moves to a new cluster only if the component of $\mathcal{J}_{\text{mpckm}}$ contributed by this point decreases. So when all points are given their new assignment, $\mathcal{J}_{\text{mpckm}}$ will decrease or remain the same.

**M step:** In the M-step, the cluster centroids $\boldsymbol{\mu}_h$ are first re-estimated using the points in $\mathcal{X}_h$. As a result, the contribution of each cluster to $\mathcal{J}_{\text{mpckm}}$ is minimized. The pairwise constraints do not take in part in this centroid re-estimation step because the constraints are not an explicit function of the centroid. Thus, only the first term (the distance component) of $\mathcal{J}_{\text{mpckm}}$ is minimized in this step. The centroid re-estimation step effectively remains the same as KMeans.

**Algorithm:** `MPC-KMeans`
**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 $K$, sets of constraint weights $W$ and $\overline{W}$.
**Output:** Disjoint $K$ partitioning $\{\mathcal{X}_h\}_{h=1}^K$ of $\mathcal{X}$ such that
objective function $\mathcal{J}_{\text{mpckm}}$ is (locally) minimized.
**Method:**
1. Initialize clusters:
1a. create the $\lambda$ 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 $\mathbf{x}_i$ to cluster $h^*$
(i.e. set $\mathcal{X}_{h^*}^{(t+1)}$), for $h^* = \arg\min_h (\|\mathbf{x}_i - \boldsymbol{\mu}_h^{(t)}\|_{\mathbf{A}}^2 - \log(\det(\mathbf{A}))$
$+ \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{M}} w_{ij} f_M(\mathbf{x}_i,\mathbf{x}_j) \mathbb{1}[h \neq l_j]$
$+ \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{C}} \overline{w}_{ij} f_C(\mathbf{x}_i,\mathbf{x}_j) \mathbb{1}[h = l_j])$
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. `update_metric`: $\mathbf{A}^{-1} = \sum_{\mathbf{x}_i\in\mathcal{X}} (\mathbf{x}_i - \boldsymbol{\mu}_{l_i})(\mathbf{x}_i - \boldsymbol{\mu}_{l_i})^T$
$- \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{M}^*} w_{ij}(\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^T \mathbb{1}[l_i \neq l_j]$
$+ \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{C}^*} \overline{w}_{ij}(\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^T \mathbb{1}[l_i = l_j]$
2d. $t \leftarrow (t+1)$

Figure 3.14: Metric Pairwise Constrained KMeans algorithm

The second part of the M-step is metric learning, where the matrix $\mathbf{A}$ is re-estimated to decrease the objective function $\mathcal{J}_{\text{mpckm}}$. The updated matrix $\mathbf{A}$ is obtained by taking the partial derivative $\frac{\partial \mathcal{J}_{\text{mpckm}}}{\partial \mathbf{A}}$ and setting it to zero, resulting in:

$$\begin{aligned}\mathbf{A} \; = \; & \Big(\sum_{\mathbf{x}_i\in\mathcal{X}} (\mathbf{x}_i - \boldsymbol{\mu}_{l_i})(\mathbf{x}_i - \boldsymbol{\mu}_{l_i})^T - \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{M}^*} w_{ij}(\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^T \mathbb{1}[l_i \neq l_j] \\ & + \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{C}^*} \overline{w}_{ij}(\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^T \mathbb{1}[l_i = l_j]\Big)^{-1}\end{aligned}$$

where $M^*$ and $C^*$ are subsets of $M$ and $C$ that exclude the constraint pairs for which the penalty functions $f_M$ and $f_C$ take the threshold values $\alpha_{min}$ and $\alpha_{max}$ respectively. See (Basu, Bilenko, & Mooney, 2003b) for the details of the derivation.

Since estimating a full matrix $\mathbf{A}$ from limited training data is difficult, we limit ourselves to diagonal $\mathbf{A}$ for most of our experiments, which is equivalent to learning a metric via feature weighting. In that case, the $d$-th diagonal element of $\mathbf{A}$, $a_{dd}$, corresponds to the weight of the $d$-th feature:

$$a_{dd} = \Bigg( \sum_{\mathbf{x}_i \in \mathcal{X}} (\mathbf{x}_{id} - \boldsymbol{\mu}_{l_i d})^2 - \sum_{(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{M}^*} w_{ij} (\mathbf{x}_{id} - \mathbf{x}_{jd})^2 \mathbb{1}[l_i \neq l_j] + \sum_{(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{C}^*} \overline{w}_{ij} (\mathbf{x}_{id} - \mathbf{x}_{jd})^2 \mathbb{1}[l_i = l_j] \Bigg)^{-1}$$

Intuitively, the first term in the sum, $\sum_{\mathbf{x}_i \in \mathcal{X}} (\mathbf{x}_{id} - \boldsymbol{\mu}_{l_i d})^2$, scales the weight of each feature proportionately to the feature's contribution to the overall cluster dispersion, analogously to scaling performed when computing Mahalanobis distance. The second two terms that depend on constraint violations, $-\sum_{(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{M}^*} w_{ij} (\mathbf{x}_{id} - \mathbf{x}_{jd})^2 \mathbb{1}[l_i \neq l_j]$ and $\sum_{(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{C}^*} \overline{w}_{ij} (\mathbf{x}_{id} - \mathbf{x}_{jd})^2 \mathbb{1}[l_i = l_j]$, respectively contract and stretch each dimension attempting to mend the current violations. Thus, the metric weights are adjusted at each iteration in such a way that the contribution of different attributes to distance is equalized, while constraint violations are minimized.

The objective function decreases after every cluster assignment, centroid re-estimation and metric learning step till convergence, implying that the `MPC-KMeans` algorithm will converge to a local minima of $\mathcal{J}_{\text{mpckm}}$.

### 3.4.4 Experimental results

In this section, we present ablation experiments which show the improved performance of our unified semi-supervised clustering framework. For each dataset, we compared four semi-supervised clustering schemes:

- `MPC-KMeans` clustering, which involves both seeding and metric learning in the unified framework described in Section 3.4.3;
- `M-KMeans`, which is K-Means clustering with the metric learning component only, without utilizing constraints for seeding;
- `PC-KMeans` clustering, which utilizes constraints for seeding and cluster assignment without doing any metric learning;
- Unsupervised K-Means clustering.

#### Datasets

Experiments were conducted on several datasets from the UCI repository: *Iris*, *Wine*, and representative randomly sampled subsets from the *Pen-Digits* and *Letter* datasets. For *Pen-Digits* and *Letter*, we chose two sets of three classes: {**I, J, L**} from *Letter* and {**3, 8, 9**} from *Pen-Digits*, sampling 20% of the data points from the original datasets randomly. These classes were chosen from the handwriting recognition datasets since they intuitively represent difficult visual discrimination problems.

#### Learning curves with cross validation

We used the same learning curve generation methodology as in Section 3.3.4. Unit constraint weights $W$ and $\overline{W}$ were used, since the datasets did not provide information for setting individual weights for the constraints. The maximum square distance between must-link constraints was used as value for $\alpha_{\max}$, while $\alpha_{\min}$ was set to 0. All experiments were run using Euclidean KMeans.

#### Results

The learning curves in Figures 3.15-3.18 illustrate that providing pairwise constraints is beneficial to clustering quality. On the presented datasets, the unified approach (`MPC-KMeans`) outperforms individual seeding (`PC-KMeans`) and metric learning (`M-KMeans`) approaches. When both seeding and metric learning are utilized, the unified approach benefits from the individual strengths of the two methods.

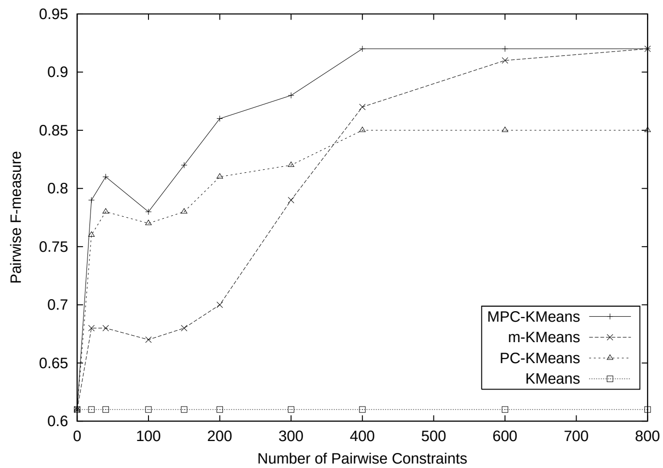


Figure 3.15: Comparison of pairwise F-measure values on Iris dataset

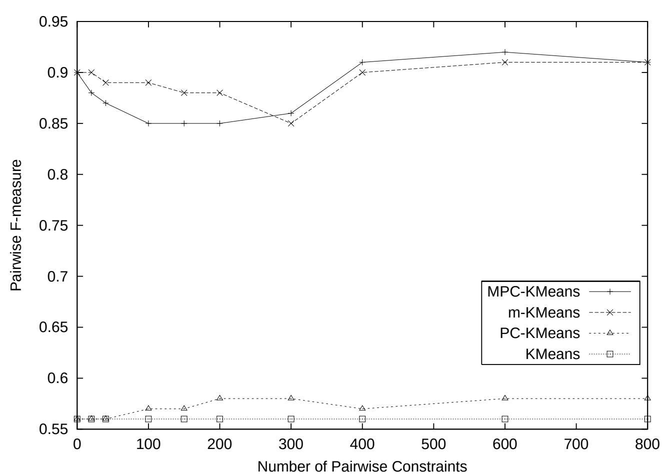


Figure 3.16: Comparison of pairwise F-measure values on Wine dataset

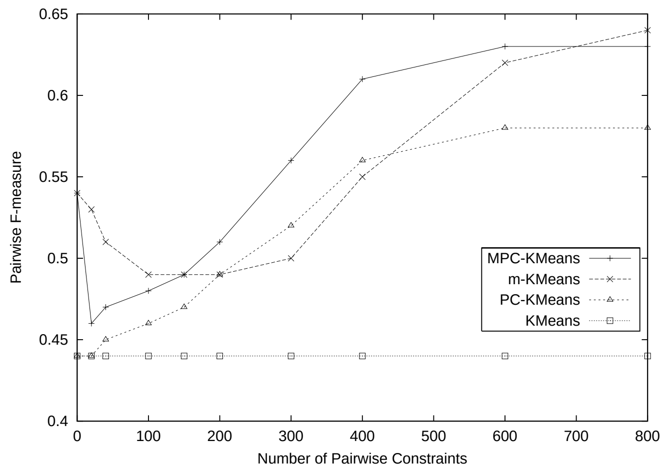


Figure 3.17: Comparison of pairwise F-measure values on Letter-IJL dataset

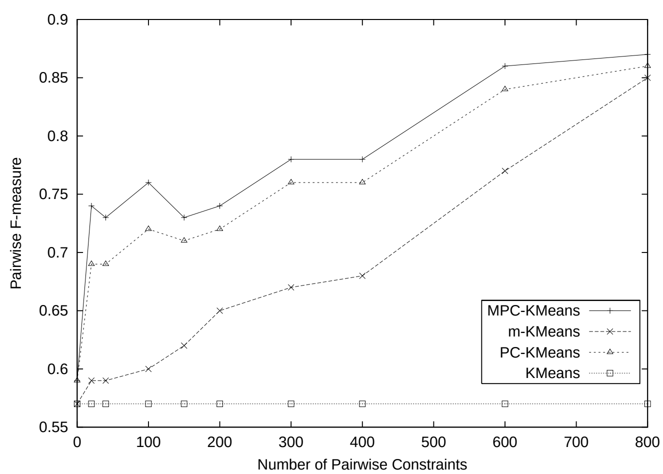


Figure 3.18: Comparison of pairwise F-measure values on Digits-389 dataset

Some of the metric learning curves display a characteristic "dip", where clustering accuracy decreases when initial constraints are provided, but after a certain point starts to increase and eventually outperforms the initial point of the learning curve. We conjecture that this phenomenon is due to the fact that feature weights learned from few constraints are unreliable, while increasing the number of constraints provides the metric learning mechanism enough data to estimate good metric parameters. On the other hand, seeding the clusters with a small number of pairwise constraints has an immediate positive effect on the final cluster quality, while providing more pairwise constraints has diminishing returns, i.e., `PC-KMeans` learning curves rise slowly. As can be seen from the `MPC-KMeans` results, the unified method has the advantages of both metric learning and seeding, and outperforms each of these individual methods of semi-supervised clustering.

# Chapter 4

# Proposed Research

This chapter outlines the issues that we want to further investigate in this thesis.

## 4.1 Short-term definite goals

This section describes our short-term goals, which we will definitely investigate in our future work.

### 4.1.1 Soft and noisy constraints

The labeled semi-supervised clustering framework can handle both soft or noisy labels, as shown in Section 3.1. We want to extend the formulation of the pairwise constrained semi-supervised clustering framework to be able to handle soft or noisy constraints also. This would involve extending the PCC clustering framework and the active constraint selection strategy to include a noise model, and formulating a model where violation of soft-constraints can be considered. The penalized cluustering objective function can handle soft or noisy constraints, but the active PCC model we have considered needs to be extended in two places to handle such constraints. Firstly, the initialization step takes the transitive closure of must-link constraints and adds entailed constraints inferred from cannot-links — this needs to be modified, since for soft or noisy constraints such direct constraint inference will not be possible. Secondly, the active learning strategy also uses the fact that the constraints are hard and noise-free. Both these steps need to be modified to be able to support soft or noisy constraints.

SCOP-KMeans (Wagstaff, 2002) supports soft-constraint violation, but they do not perform any algorithm analysis to show convergence guarantees.

### 4.1.2 Incomplete semi-supervision and class discovery

In semi-supervised classification, all classes are assumed to be known apriori and labeled training data is provided for all classes. In labeled semi-supervised clustering, when we consider clustering a dataset that has an underlying class labeling, we would like to consider incomplete seeding where labeled data are not provided for every underlying class. For such incomplete semi-supervision, we would like to see if the labels on some classes can help the clustering algorithm discover the unknown classes. An example of class discovery using incomplete seeding is provided in the Figure 4.1. Given the points in Figure 4.1, if we are asked to do a 2-clustering, we can get a clustering as shown in Figure 4.1. Now, if we give as input a pair of points labeled to be in the same cluster (shown by the annular points in Figure 4.2), we will get a clustering as in Figure 4.2. In this case, even though we did not provide any supervision about the top cluster, clustering using the provided supervision helped us to discover that cluster.

Initial experiments for class discovery under incomplete seeding were considered in (Basu et al., 2002), where seeds were not provided for different categories and the NMI measure was calculated on the whole test dataset. We want to refine these experiments, so that (1) when we remove seeds from one category in the labeled data, we still consider the same overall number of labeled data points provided to the clustering algorithm (by adding more

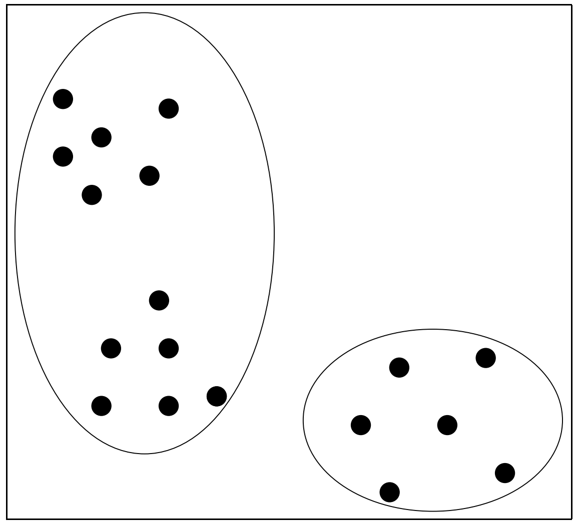

Figure 4.1: Incomplete seeding and class discovery

labeled data from the other categories), (2) the NMI measure is calculated only on the test data points which belong to categories for which no labeled data have been provided, and not on the whole test data (so that we see whether the unknown classes are discovered when supervision is provided for the other classes). We expect that in these experiments the semi-supervised clustering algorithm will be able to discover the categories for which no supervision was provided. We also want to extend the work of (Miller & Browning, 2003) to give a theoretically well-motivated model for class discovery.

### 4.1.3 Model selection

In all our experiments so far, we have considered that the number of clusters, $K$, has been provided as input to the algorithm. In the future, we want select the number of clusters automatically by using a model selection criterion in the KMeans or EM clustering objective function. Several model selection criteria exist in the literature for selecting the right number of clusters. Criteria like Minimum Description Length (Rissanen, 1978), Bayesian Information Criterion (Pelleg & Moore, 2000) or Minimum Message Length (Wallace & Lowe, 1999) encode the Occam's Razor principle in some form, penalizing models according to their model complexity. We would like to use one of these models, or explore a recent model proposed in (Hamerly & Elkan, 2003), where the $K$ in KMeans is selected based on a statistical test for the hypothesis that a subset of the data follows a Gaussian distribution. In hierarchical clustering, the right number of clusters can be selected by using some criteria for stopping the cluster merging, e.g., (Fern & Brodley, 2003) used a heuristic where they stopped merging clusters when the similarity value between the two closest current clusters in the algorithm had a sudden drop compared to the values in previous merges.

As mentioned in Section 1.3.2, automatic model selection is the main advantage of using semi-supervised clustering rather than semi-supervised classification in scenarios where knowledge of the relevant categories is incomplete.

### 4.1.4 Generalizing the unified semi-supervised clustering model

In Section 3.4, we proposed a framework for unifying search-based and similarity-based semi-supervised clustering that works only with Euclidean KMeans. I am working with Misha Bilenko, who is formulating an effective metric learning algorithm in high dimensions (Bilenko, 2003), on generalizing our unified PCC framework to work with SPKMeans so that we can apply it to domains like text.

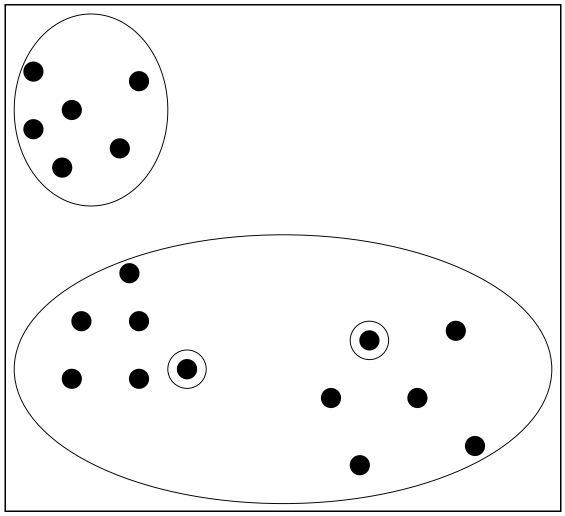

Figure 4.2: Incomplete seeding and class discovery

### 4.1.5 Other semi-supervised clustering algorithms

We want to apply our semi-supervision ideas to hierarchical algorithms, e.g., HAC and Cobweb. Incorporating constraints into hierarchical algorithms will be relatively straightforward. For example, to run constrained HAC, we can change the similarity $s(\mathbf{x}_i, \mathbf{x}_j)$ between 2 points $\mathbf{x}_i$ and $\mathbf{x}_j$ to $s(\mathbf{x}_i, \mathbf{x}_j) + w_{ij}$ if there is a must-link provided between $\mathbf{x}_i$ and $\mathbf{x}_j$ with weight $w_{ij}$, and $s(\mathbf{x}_i, \mathbf{x}_j) - \overline{w}_{ij}$ if there is a cannot-link provided between $\mathbf{x}_i$ and $x_j$ with weight $\overline{w}_{ij}$. Then, we can proceed and run the usual HAC algorithm on the data points using this modified similarity metric, so that at each cluster-merge step, we consider the similarity between the data points as well as the cost of constraint violation incurred during the merge operation.

A more interesting problem would be when the initial supervision is given in the form of a hierarchy, and the clustering problem will be to do hierarchical clustering "using" the initial hierarchy. We want to formalize the notion of using an initial seed hierarchy for hierarchical clustering. Such an approach would be useful for content management applications, e.g., if the requirement is to hierarchically cluster the documents of a company, and the initial seed hierarchy is a preliminary directory structure containing a subset of the documents.

So far we have mainly focussed on clustering algorithms that use a generative model. We also want to apply the pairwise constrained framework to discriminative clustering algorithms (e.g., graph partitioning), for which pairwise constraints are a natural way for providing constraints. Another interesting research direction would be online clustering in the semi-supervised framework.

### 4.1.6 Ensemble semi-supervised clustering

In our work so far, we have assumed constraints to be noise-free. We have also assumed the weights on the constraints to be uniform (`PCKMeans`) or changed the weights based on the "difficulty of satisfying the constraints" (unified model). An interesting problem in the PCC model would be the choice of the constraint weights in the general case of noisy constraints.

Given a set of noisy constraints, we can create an ensemble of semi-supervised clusterers, each of which put different weights on the constraints and possibly get different clusterings. We propose a scheme for creating an ensemble of PCC clusterers and combining their results using boosting (Freund & Schapire, 1996).

Each PCC clusterer can be considered as a weak learner taking pairwise data points as input, and giving an binary output decision of "same-cluster" or "different-cluster". The must-link and cannot-link constraints can be considered as the training data for each weak learner. Given a set of input constraints, the PCC clusterer initially sets all constraints to have uniform weight and performs clustering. After clustering is completed, the clusterer categorizes each pair of points as "same-cluster" or "different-cluster", based on whether the pair ended up in the same cluster or in different clusters. Since the given constraints are noisy, some of them will be violated by the clustering. The constraints are reweighted based on the number of errors made by the weak learner, and a new clusterer is created to perform the clustering with the new weights on the constraints. We use boosting for re-weighting of the constraints and combining the outputs of the clusterers in the ensemble.

The boosted ensemble will give as output a $N \times N$ symmetric matrix $M$ with 0/1 entries, where the $M[i, j]$ entry in the matrix would represent the ensemble decision of whether points $i$ and $j$ should be in the same cluster ($M[i, j] = 1$) or not ($M[i, j] = 0$). Once we get this matrix, we can apply any discriminative clustering algorithm (e.g., graph partitioning using min-cuts) on $M$, which we have to evaluate empirically to see whether it gives us better clustering performance than the initial PCC clusterer. Initial work along the lines of semi-supervised boosting in the product space has been proposed in (Hertz, Bar-Hiller, & Weinshall, 2003), which we want to investigate further.

### 4.1.7 Other application domains

We have been running our experiments on text data and UCI datasets. In future, we also want to run experiments on other domains.

#### Search result clustering

One domain we are very interested in is the clustering of web search results. It is an important tool for helping search engine users do better navigation of search results, and has become a part of some recent search engines, e.g., Vivisimo. For short or ambiguous queries, e.g., "jaguar", it is useful for a user if the retrieved documents are organized in different clusters based on their topics, e.g., Jaguar cars, jaguar cats and Apple Jaguar OS. The task we are proposing is using constraints derived from the click-through data in the query-logs to get constrained clustering of search results, which would correspond better with user browsing behavior.

(Joachims, 2002) proposed a model for a user browsing through search results of a query given to a search engine. The model considered that lower ranked search results clicked-on by the user in a ranked list (as on a search results webpage) are more relevant to the user than higher ranked results that were not clicked-on. As a natural corollary of this model, we consider that if a user clicks on two search results on a page, we can infer a must-link constraint between these pages since the user considered both of them relevant for his query. For example, let us consider a user who gives the query "`jaguar`" looking for automobiles. The top 5 corresponding search results in Google are: (1) Jaguar Apple (the Mac OS website), (2) Jaguar Racing (Grand Prix racing team sponsored by Jaguar), (3) Jaguar Cars (official Jaguar car website), (4) Jaguar Australia (Australian Jaguar car website), and (5) Jaguar Canada (Canadian Jaguar car website). The user in our example, who is interested in jaguar cars, would tend to skip the first two search results and click on some of the last 3 results, say (3) and (5). Looking at the query-logs, we can then infer a must-link constraint between (3)-(5). Note that these inferred must-link constraints will be individually noisy, but we can consider a subset of reliable constraints by aggregating the constraints over all instances of a particular query in the query-logs and considering only those constraints above a particular frequency threshold. For example, if the threshold is 5%, we consider only those must-link constraints that occur in more than 5% of all user sessions for that query in the query-logs. At the end of this aggregation, we will have a set of inferred must-link constraints corresponding to each query.

If we want more aggressive constraint inference, we can also consider that if the user skips a higher ranked search result and clicks on a result of lower rank, then there is a cannot-link between these results. In the jaguar example, we can infer cannot-link constraints between (1)-(3), (1)-(5), (2)-(3), and (2)-(5). Note, however, that we have to be careful while inferring cannot-link constraints, since in this example (4)-(5) will also be incorrectly inferred to be a cannot-link. To take care of such cases, we have to apply a higher cutoff frequency (say 10%) for selecting cannot-links from the query-logs. For constraints above the cutoff frequency thresholds, we can weight the importance of the constraints by the fraction of times they occurred in the query-log entries for the corresponding query.

The proposed algorithms in the PCC framework could be used to improve clustering of search results. The constraints corresponding to different queries could be collected from the query-logs and used while clustering the results of corresponding queries, to give better clustering. We could also re-use the constraints for one query to cluster other similar queries. To this effect, we can use a query-document bipartite graph, where there is a link between a query and a document if the document is clicked-on a significant fraction of times in the query-log entries for that query.

We can then cluster the queries, based on the overlap of the documents linked to each query in this graph (Dhillon, 2001; Mishra, Ron, & Swaminathan, 2003). Let $q_2$ be a query for which not many constraints have been inferred from the query-logs. Queries $q_1$ and $q_2$ can be considered similar if they belong to the same cluster, and the constraints of $q_1$ inferred from the query-logs can be now used (after weighting them down suitably, e.g., multiplying them by the fractional similarity between $q_1$ and $q_2$) while clustering the search results of $q_2$.

This proposed work is part of a research proposal submitted to Google. If the proposal gets approved, we can get access to anonymous query-logs on which we can run our experiments. If a constrained clustering prototype is developed, evaluation of the clustering results can be performed by "temps" (temporary contract workers) at Google. If such clustering is not feasible in real-time, a possible initial deployment model could be the following: do offline clustering of the top $N$ most popular queries using constraints from the query-logs, cache the results and show the clustering results to the restricted set of users giving those queries.

**Other datasets**

We want to apply our algorithms to clustering astronomical datasets, e.g., Mars spectral data (Wagstaff, 2002) and galaxy data (Pelleg & Moore, 2000). The Mars data will be especially useful — the dataset consists of spectral analysis of telescopic Mars images, and has domain knowledge (spatial relations, spectrum slope characteristics, etc.) encoded in the form of soft constraints, which will be very useful for testing the soft-constrained clustering algorithms we plan to develop. We also want to investigate clustering of bioinformatics data, especially gene micro-array data and phylogenetic profiles (Marcotte, Xenarios, van der Bliek, & Eisenberg, 2000). In these cases, constraints can be derived from other knowledge sources, e.g., while clustering a large set of genes, we can use the information that some of these genes belong to the same pathway (implying must-link constraints) or to different pathways (implying cannot-link constraints) in the KEGG database (Ogata, Goto, Sato, Fujibuchi, Bono, & Kanehisa, 1999).

## 4.2 Long-term speculative goals

This section outlines our long-term goals, some of which we plan to investigate as time permits.

### 4.2.1 Other forms of semi-supervision

Till now, we have only considered labeled data and constraints between data points as possible methods of supervision in clustering. Other forms of supervision have also been considered in clustering, e.g., (1) cluster-size constraints (Banerjee & Ghosh, 2002), where balancing the size of the clusters is considered as an explicit soft constraint, and (2) attribute-level constraints (Dai, Lin, & Chen, 2003), where clustering is performed under constraints on the numerical attributes (e.g., maximum difference between two points in a cluster along the "age" attribute should be 15).

We want to also explore supervision in the form of partial classification functions defined on subsets of the data space $\mathcal{X}$. In this case, along with the unsupervised data $\mathcal{X}$, we will have a set of classification functions, each of which classifies a subset of points from $\mathcal{X}$ to a set of class labels that can be different for each classifier. For example, while clustering search engine results, we can use the DMOZ and the Yahoo! hierarchies as classifiers for a subset of the documents returned for a query. Note that in the general case, the label sets for the classifiers will be different, e.g. in the example above, DMOZ and Yahoo! will have different categories.

Given a classifier $C$ that gives posterior probabilities corresponding to its set of class labels $L$ and two points $\mathbf{x}_1$ and $\mathbf{x}_2$, we can estimate the probabilities of a must-link and cannot-link between the two points, given the classifier, as follows:

$$\frac{\mathbf{Pr}(\text{must-link}(\mathbf{x}_1, \mathbf{x}_2)|C)}{\mathbf{Pr}(\text{cannot-link}(\mathbf{x}_1, \mathbf{x}_2)|C)} = \frac{\sum_{l \in L} \mathbf{Pr}(\mathbf{x}_1|l)\mathbf{Pr}(\mathbf{x}_2|l)}{\sum_{l_1, l_2 \in L, l_1 \neq l_2} \mathbf{Pr}(\mathbf{x}_1|l_1)\mathbf{Pr}(\mathbf{x}_2|l_2)}$$

Given $R$ classifiers $\{C_r\}_{r=1}^R$, the probability of constraints between the two points, as estimated from the classifiers, is:

$$\begin{aligned}
\mathbf{Pr}(\text{must-link}(\mathbf{x}_1, \mathbf{x}_2)) &= \sum_{r=1}^{R} \mathbf{Pr}(\text{must-link}(\mathbf{x}_1, \mathbf{x}_2) | C_r) \mathbf{Pr}(C_r) \\
\mathbf{Pr}(\text{cannot-link}(\mathbf{x}_1, \mathbf{x}_2)) &= 1 - \mathbf{Pr}(\text{must-link}(\mathbf{x}_1, \mathbf{x}_2))
\end{aligned}$$

In the absence of any other information on the distribution over the classifiers, $\mathbf{Pr}(C_r)$ would be assumed to be uniform.

In a generative clustering model, the probabilistic constraints inferred from the different classifiers can be directly incorporated into the soft-constrained PCC model. If we use kernel-based discriminative clustering, we can adapt the original kernel so that the kernel similarity between two points in the adapted kernel would be more similar if the points have a higher probability of being must-linked and less similar of they have a higher probability of being cannot-linked, using a framework similar to (Kwok & Tsang, 2003).

### 4.2.2 Study of cluster evaluation metrics

As observed in Figures 3.10-3.13 in Section 3.3.4, there seems to be a strong correlation between the NMI and pairwise F-measure clustering evaluation metrics. We want to further investigate this similarity between NMI and pairwise F-measure. Dom (2001) gives a nice formulation of evaluation metrics for hard clustering using the clustering confusion matrix, where he shows the relation between popular hard clustering evaluation metrics like the Rand Index, the Jaccard index, the Folkes-Mallows index and the Hubert $\Gamma$ statistic. We want to see if we can extend this formulation to evaluate soft-clustering.

### 4.2.3 Approximation algorithms for semi-supervised clustering

Another interesting research direction is considering how semi-supervision affects appromixation algorithms for some clustering methods, e.g., KMedian. The KMedian problem, which was explained briefly in Section 2.2.1, is similar to the facility location problem. In the facility location problem, we are given a set of demand points and a set of candidate facility sites with costs of building facilities at each of them. Each demand point is then assigned to its closest facility, incurring a service cost equal to the distance to its assigned facility. The goal is to select a subset of sites where facilities should be built, so that the sum of facility costs and the service costs for the demand points is minimized. The KMedian problem is similar to facility location, but with a few differences — in KMedian there are no facility costs and there is a bound $K$ on the number of facilities that can be opened. The KMedian objective is to select a set of $K$ facilities so as to minimize the sum of the service costs for the demand points.

We propose an semi-supervised extension to KMedian to handle constraints on the demand points. The constrained KMedian problem would be additionally given an input set of must-link and cannot-link constraints on the demand points (i.e., two demand points should be or should not be assigned to the same facility), and the goal would be to minimize an objective function that is the sum of the service costs for the demand points and the cost of violating the constraints.

Initial investigation has shown we if we consider only must-link constraints, we can make a simple modification to an approximation algorithm for KMedian to get a constant-factor approximation algorithm for must-link constrained KMedian. However, if we consider both must-links and cannot-link constraints, then the constrained KMedian problem becomes NP-complete, which can be shown by a reduction to graph coloring (Plaxton, 2003). An interesting research problem to look into is the constrained KMedian problem with limited number of cannot-links (e.g., $\mathcal{O}(\log K)$ cannot-links, $K$ being the number of clusters), and to try and get approximation guarantees in this case.

### 4.2.4 Analysis of labeled semi-supervision with EM

As mentioned in Section 3.1, the objective function plots for `S-KMeans` seem to increase exponentially as the number of labeled points are increased along the learning curve. As shown in (3.1), if we consider a Gaussian model for each cluster, the probability of deviation of the centroid estimates falls exponentially with the number of seeds (Banerjee, 2001). I am working with Arindam Banerjee to extend this observation and show that when we add more unlabeled points along with the seeds and perform the EM iteration, we get a tighter bound for the centroid deviation with high probability. Ratsaby & Venkatesh (1995) gave a PAC analysis for labeled and unlabeled sample complexity when learning a classification rule, with an underlying data generation model of a mixture of 2 Gaussians. We want to use their style of analysis to give similar bounds for the semi-supervised EM algorithm on a mixture of 2 Gaussians.

## 4.3 Conclusion

Our main goal in the proposed thesis is to study search-based semi-supervised clustering algorithms and apply them to different domains. As explained in Chapter 3, our initial work has shown: (1) how supervision can be provided to clustering in the form of labeled data points or pairwise constraints; (2) how informative constraints can be selected in an active learning framework for the pairwise constrained semi-supervised clustering model; and (3) how search-based and similarity-based techniques can be unified in semi-supervised clustering. In our work so far, we have mainly focussed on generative clustering models, e.g. KMeans and EM, and ran experiments on clustering low-dimensional UCI datasets or high-dimensional text datasets.

In this thesis, we want to study other aspects of semi-supervised clustering, like: (1) the effect of noisy, probabilistic or incomplete supervision in clustering; (2) model selection techniques for automatic selection of number of clusters in semi-supervised clustering; (3) ensemble semi-supervised clustering. In future, we want to study the effect of semi-supervision on other clustering algorithms, especially in the discriminative clustering and online clustering framework. We also want to study the effectiveness of our semi-supervised clustering algorithms on other domains, e.g., web search engines (clustering of search results), astronomy (clustering of Mars spectral images) and bioinformatics (clustering of gene microarray data).

# Bibliography


Abe, N., & Mamitsuka, H. (1998). Query learning strategies using boosting and bagging. In *Proceedings of the Fifteenth International Conference on Machine Learning (ICML-98)*, pp. 1–10.

Baeza-Yates, R., & Ribeiro-Neto, B. (1999). *Modern Information Retrieval*. ACM Press, New York.

Banerjee, A., Dhillon, I., Ghosh, J., & Sra, S. (2003). Generative model-based clustering of directional data. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2003)*.

Banerjee, A. (2001). Large scale clustering of sequential patterns. Doctoral Dissertation Proposal, University of Texas at Austin.

Banerjee, A., & Ghosh, J. (2002). On scaling up balanced clustering algorithms. In *Proceedings of the SIAM International Conference on Data Mining*, pp. 333–349.

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

Basu, S., Banerjee, A., & Mooney, R. J. (2002). Semi-supervised clustering by seeding. In *Proceedings of 19th International Conference on Machine Learning (ICML-2002)*, pp. 19–26.

Basu, S., Banerjee, A., & Mooney, R. J. (2003a). Active semi-supervision for pairwise constrained clustering. Submitted for publication, available at `http://www.cs.utexas.edu/~sugato/`.

Basu, S., Bilenko, M., & Mooney, R. J. (2003b). Comparing and unifying search-based and similarity-based approaches to semi-supervised clustering. In *Proceedings of the ICML-2003 Workshop on the Continuum from Labeled to Unlabeled Data in Machine Learning and Data Mining* Washington, DC.

Bilenko, M. (2003). Learnable similarity functions and their applications to record linkage and clustering. Doctoral Dissertation Proposal, University of Texas at Austin.

Bilenko, M., & Mooney, R. J. (2002). Learning to combine trained distance metrics for duplicate detection in databases. Tech. rep. AI 02-296, Artificial Intelligence Laboratory, University of Texas at Austin, Austin, TX.

Bilenko, M., & Mooney, R. J. (2003). Adaptive duplicate detection using learnable string similarity measures. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2003)*, pp. 39–48 Washington, DC.

Bilmes, J. (1997). A gentle tutorial on the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models. Tech. rep. ICSI-TR-97-021, ICSI.

Blake, C. L., & Merz, C. J. (1998). UCI repository of machine learning databases. http://www.ics.uci.edu/~mlearn/MLRepository.html.

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

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

Bradley, P. S., Mangasarian, O. L., & Street, W. N. (1997). Clustering via concave minimization. In Mozer, M. C., Jordan, M. I., & Petsche, T. (Eds.), *Advances in Neural Information Processing Systems 9*, pp. 368–374. The MIT Press.

Cohn, D., Caruana, R., & McCallum, A. (2000). Semi-supervised clustering with user feedback. Unpublished manuscript. Available at `http://www-2.cs.cmu.edu/~mccallum/`.

Cohn, D. A., Ghahramani, Z., & Jordan, M. I. (1996). Active learning with statistical models. *Journal of Artificial Intelligence Research*, *4*, 129–145.

Cover, T. M., & Thomas, J. A. (1991). *Elements of Information Theory*. Wiley-Interscience.

Dai, B.-R., Lin, C.-R., & Chen, M.-S. (2003). On the techniques for data clustering with numerical constraints. In *Proceedings of the SIAM International Conference on Data Mining*.

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

Demiriz, A., Bennett, K. P., & Embrechts, M. J. (1999). Semi-supervised clustering using genetic algorithms. In *ANNIE'99 (Artificial Neural Networks in Engineering)*.

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B*, *39*, 1–38.

Devroye, L., Gyorfi, L., & Lugosi, G. (1996). *A Probabilistic Theory of Pattern Recognition*. Springer Verlag.

Dhillon, I. S., Fan, J., & Guan, Y. (2001). Efficient clustering of very large document collections. In *Data Mining for Scientific and Engineering Applications*. Kluwer Academic Publishers.

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

Dhillon, I. S. (2001). Co-clustering documents and words using bipartite spectral graph partitioning. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2001)*.

Dom, B. E. (2001). An information-theoretic external cluster-validity measure. Research report RJ 10219, IBM.

Dubnov, S., El-Yaniv, R., Gdalyahu, Y., Schneidman, E., Tishby, N., & Yona, G. (2002). A new nonparametric pairwise clustering algorithm based on iterative estimation of distance profiles. *Machine Learning*, *47*(1), 35–61.

Duda, R. O., Hart, P. E., & Stork, D. G. (2001). *Pattern Classification* (Second edition). Wiley, New York.

Fayyad, U. M., Reina, C., & Bradley, P. S. (1998). Initialization of iterative refinement clustering algorithms. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD-98)*, pp. 194–198.

Fern, X., & Brodley, C. (2003). Random projection for high dimensional data clustering: A cluster ensemble approach. In *Proceedings of 20th International Conference on Machine Learning (ICML-2003)*.

Fisher, D. H. (1987). Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, *2*, 139–172.

Freund, Y., & Schapire, R. E. (1996). Experiments with a new boosting algorithm. In Saitta, L. (Ed.), *Proceedings of the Thirteenth International Conference on Machine Learning (ICML-96)*. Morgan Kaufmann.

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

Garey, M. R., Johnson, D. S., & Witsenhausen, H. S. (1982). The complexity of the generalized Lloyd-max problem. *IEEE Transactions on Information Theory*, *28*(2), 255–256.

Garey, M., & Johnson, D. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, New York, NY.

Ghahramani, Z., & Jordan, M. I. (1994). Supervised learning from incomplete data via the EM approach. In *Advances in Neural Information Processing Systems 6*, pp. 120–127.

Ghani, R., Jones, R., & Rosenberg, C. (Eds.). (2003). *ICML Workshop on the Continuum from Labeled to Unlabeled Data in Machine Learning and Data Mining*, Washington, DC.

Goldszmidt, M., & Sahami, M. (1998). A probabilistic approach to full-text document clustering. Tech. rep. ITAD-433-MS-98-044, SRI International.

Guha, S., Rastogi, R., & Shim, K. (1999). Rock: a robust clsutering algorithm for categorical attributes. In *Proceedings of the Fifteenth International Conference on Data Engineering*.

Hamerly, G., & Elkan, C. (2003). Learning the $k$ in $k$-means. In *Advances in Neural Information Processing Systems 15*.

Hastie, T., & Tibshirani, R. (1996). Discriminant adaptive nearest-neighbor classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *18*(6), 607–617.

Hertz, T., Bar-Hiller, A., & Weinshall, D. (2003). Learning distance functions with product space boosting. Tech. rep. TR 2003-35, Leibniz Center for Research in Computer Science, The Hebrew University of Jerusalem.

Hillel, A. B., Hertz, T., Shental, N., & Weinshall, D. (2003). Learning distance functions using equivalence relations. In *Proceedings of 20th International Conference on Machine Learning (ICML-2003)*.

Hinneburg, A., & Keim, D. A. (1998). An efficient approach to clustering in large multimedia databses with noise. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD-98)*, pp. 58–65.

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. In *Advances in Neural Information Processing Systems 10*.

Jaakkola, T., & Haussler, D. (1999). Exploiting generative models in discriminative classifiers. In *Advances in Neural Information Processing Systems 11*, pp. 487–493.

Jain, A. K., & Dubes, R. C. (1988). *Algorithms for Clustering Data*. Prentice Hall, New Jersey.

Jain, A. K., Myrthy, M. N., & Flynn, P. J. (1999). Data clustering: A survey. *ACM Computing Survey*, *31*(3), 264–323.

Jain, K., & Vazirani, V. (2001). Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and Lagrangian relaxation. *Journal of the ACM*, *48*, 274–296.

Joachims, T. (1999). Transductive inference for text classification using support vector machines. In *Proceedings of the Sixteenth International Conference on Machine Learning (ICML-99)* Bled, Slovenia.

Joachims, T. (2002). Optimizing search engines through clickthrough data. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2002)*.

Kamvar, S. D., Klein, D., & Manning, C. D. (2002). Interpreting and extending classical agglomerative clustering algorithms using a model-based approach. In *Proceedings of 19th International Conference on Machine Learning (ICML-2002)*.

Karypis, G., Han, E. H., & Kumar, V. (1999). Chameleon: A hierarchical clustering algorithm using dynamic modeling. *IEEE Computer*, *32*(8), 68–75.

Karypis, G., & Kumar, V. (1998). A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, *20*(1), 359–392.

Kaufman, L., & Rousseeuw, P. (1990). *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley and Sons, New York.

Kearns, M., Mansour, Y., & Ng, A. Y. (1997). An information-theoretic analysis of hard and soft assignment methods for clustering. In *Proceedings of 13th Conference on Uncertainty in Artificial Intelligence (UAI-97)*, pp. 282–293.

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. In *Proceedings of the The Nineteenth International Conference on Machine Learning (ICML-2002)* Sydney, Australia.

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

Kwok, J. T., & Tsang, I. W. (2003). Learning with idealized kernels. In *Proceedings of 20th International Conference on Machine Learning (ICML-2003)*, pp. 400–407.

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

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability*, pp. 281–297.

Manning, C. D., & Schütze, H. (1999). *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA.

Marcotte, E. M., Xenarios, I., van der Bliek, A., & Eisenberg, D. (2000). Localizing proteins in the cell from their phylogenetic profiles. *Proceedings of the National Academy of Science*, *97*, 12115–20.

Mardia, K. V., & Jupp, P. (2000). *Directional Statistics*. John Wiley and Sons Ltd., 2nd edition.

McCallum, A., & Nigam, K. (1998). Employing EM and pool-based active learning for text classification. In *Proceedings of the Fifteenth International Conference on Machine Learning (ICML-98)* Madison, WI. Morgan Kaufmann.

Meila, M. (2003). Comparing clusterings. In *Proceedings of the 16th Annual Conference on Computational Learning Theory*.

Mettu, R., & Plaxton, C. G. (2000). The online median problem. In *Proceedings of the 41st Annual IEEE Symposium on Foundations of Computer Science*, pp. 339–348.

Miller, D. J., & Browning, J. (2003). A mixture model and EM algorithm for robust classification, outlier rejection, and class discovery. In *IEEE International Conference on Acoustics, Speech and Signal Processing*.

Mishra, N., Ron, D., & Swaminathan, R. (2003). On finding large conjunctive clusters. In *Computational Learning Theory*.

Mitchell, T. (1997). *Machine Learning*. McGraw-Hill, New York, NY.

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

Muslea, I. (2002). *Active learning with multiple views*. Ph.D. thesis, University of Southern California.

Ng, A. Y., & Jordan, M. I. (2002). On discriminative vs. generative classifiers: A comparison of logistic regression and naive Bayes. In *Advances in Neural Information Processing Systems 14*, pp. 605–610.

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

Nigam, K. (2001). *Using Unlabeled Data to Improve Text Classification*. Ph.D. thesis, Carnegie Mellon University.

Ogata, H., Goto, S., Sato, K., Fujibuchi, W., Bono, H., & Kanehisa, M. (1999). KEGG: Kyoto encyclopedia of genes and genomes. *Nucleic Acids Res.*, *27*, 29–34.

Pelleg, D., & Moore, A. (2000). X-means: Extending k-means with efficient estimation of the number of clusters. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML-2000)*.

Plaxton, C. G. (2003). Personal communication.

Rissanen, J. (1978). Modeling by shortest data description. *Automatica*, *14*, 465–471.

Seeger, M. (2000). Learning with labeled and unlabeled data. Tech. rep., Institute for ANC, Edinburgh, UK. See `http://www.dai.ed.ac.uk/~seeger/papers.html`.

Selim, S., & Ismail, M. (1984). K-means-type algorithms: A generalized convergence theorem and characterization of local optimality. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *6*, 81–87.

Sheikholesami, G., Chatterjee, S., & Zhang, A. (1998). Wavecluster: A muti-resolution clustering approach for very large spatial databases. In *Proceedings of the International Conference on Very Large Data Bases*, pp. 428–439.

Sinkkonen, J., & Kaski, S. (2000). Semisupervised clustering based on conditional distributions in an auxiliary space. Tech. rep. A60, Helsinki University of Technology.

Strehl, A., & Ghosh, J. (2000). A scalable approach to balanced, high-dimensional clustering of market-baskets. In *Proceedings of the Seventh International Conference on High Performance Computing (HiPC 2000)*.

Strehl, A., Ghosh, J., & Mooney, R. (2000). Impact of similarity measures on web-page clustering. In *Workshop on Artificial Intelligence for Web Search (AAAI 2000)*, pp. 58–64.

Vapnik, V. N. (1998). *Statistical Learning Theory*. John Wiley & Sons.

Wagstaff, K. (2002). *Intelligent Clustering with Instance-Level Constraints*. Ph.D. thesis, Cornell University.

Wagstaff, K., Cardie, C., Rogers, S., & Schroedl, S. (2001). Constrained K-Means clustering with background knowledge. In *Proceedings of 18th International Conference on Machine Learning (ICML-2001)*.

Wallace, C. S., & Lowe, D. L. (1999). Minimum message length and Kolmogorov complexity. In *The Computer Journal*.

Xing, E. P., Ng, A. Y., Jordan, M. I., & Russell, S. (2003). Distance metric learning, with application to clustering with side-information. In *Advances in Neural Information Processing Systems 15*. MIT Press.

Zhang, T., Ramakrishnan, R., & Livny, M. (1996). Birch: An efficient data clustering method for very large databases. In *In Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 103–114.