Clustering with Propagated Constraints

by

Eric Robert Eaton

Thesis submitted to the Faculty of the Graduate School
of the University of Maryland in partial fulfillment
of the requirements for the degree of
Master of Science
2005

ABSTRACT

Title of Thesis: Clustering with Propagated Constraints

Eric Robert Eaton, Master of Science, 2005

Thesis directed by: Dr. Marie desJardins, Assistant Professor
Department of Computer Science and
Electrical Engineering

Background knowledge in the form of constraints can dramatically improve the quality of generated clustering models. In constrained clustering, these constraints typically specify the relative cluster membership of pairs of points. They are tedious to specify and expensive from a user perspective, yet are very useful in large quantities. Existing constrained clustering methods perform well when given large quantities of constraints, but do not focus on performing well when given very small quantities.

This thesis focuses on providing a high-quality clustering with small quantities of constraints. It proposes a method for propagating pairwise constraints to nearby instances using a Gaussian function. This method takes a few easily specified constraints, and propagates them to nearby pairs of points to constrain the local neighborhood. Clustering with these propagated constraints can yield superior performance with fewer constraints than clustering with only the original user-specified constraints. The experiments compare the performance of clustering with propagated constraints to that of established constrained clustering algorithms on several real-world data sets.

To my parents, Robert and Eileen, for their support and guidance.

ACKNOWLEDGMENTS

First, I would like to thank my adviser, Marie desJardins, who has mentored me since I was an undergraduate. She gave me the freedom to explore a broad range of topics, and tolerated my wandering interests, while helping me focus on tangible goals. Marie's enthusiasm and personality makes working with her very enjoyable.

I would also like to thank my thesis committee, Tim Oates and Tim Finin. I especially appreciate Tim Oates for being a great mentor and unofficial adviser to my research.

Many of my ideas were refined during afternoon discussions on constrained clustering with Matt Gaston and Qianjun Xu — the ones where we filled white boards. I especially enjoyed my late-night conversations with Robert Schroll, who reached outside his field to discuss this research. I also thank Blazej Bulka, who commiserated with me on many occasions and provided invaluable technical help, and Priyang Rathod, who generously donated computer time to my experiments. Thanks also to Craig Cambias, who helped in implementing early stages of this work and the evaluation code. I would like to thank Misha Bilenko for providing me the Protein data set.

I could never have gotten this far without my parents, Robert and Eileen Eaton, who supported my education and encouraged me in all of my endeavors. They exposed me to a huge range of subjects, and tirelessly supported me as I pursued them. Thanks to my brother Michael, my grandparents Louis and Jackie, and the rest of my family for all of their support. Finally, a huge thank you to Marielle Latrick for the friendship and support that gave me the courage to complete this thesis, and for the laughter that helped me to do so.

This research was supported by a fellowship from the Goddard Earth Sciences and Technology (GEST) Center at UMBC, and NSF ITR grant #0325329.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

INTRODUCTION

Consider the following scenario. You need to create a clustering model that will group newspaper articles based on their contents. Constrained clustering methods have been shown to outperform standard unsupervised clustering methods for this problem (Basu, Bilenko, & Mooney 2004). These methods require a set of unlabeled articles and pairs of these articles (called constraints) that should be grouped together (or grouped apart). In order to specify these pairs, you need to read the articles. Existing constrained clustering methods require many pairs in order to perform well, up to several hundred to obtain high performance. Would you want to read that many articles?

By propagating the constraints provided by the user to similar articles, the proposed method infers further (possibly inaccurate) constraints between pairs of articles. This allows the user to specify significantly fewer constraints in order to reach the same level of performance.

1.1 Constrained Clustering

Recent work on constrained clustering (Wagstaff 2002; Bilenko, Basu, & Mooney 2004; Xing *et al.* 2003; Bar-Hillel *et al.* 2005) has resulted in methods to cluster unlabeled data using background knowledge provided in the form of relative membership labels for some of the data points. These algorithms perform significantly better than standard unsupervised clustering in a variety of domains.

These methods typically take labeled background information in the form of *pairwise constraints* or *equivalence sets*. Both pairwise constraints and equivalence sets are simple, natural, and useful forms of background knowledge. Pairwise constraints specify the relative clustering for a pair of points. Research has focused on two types of pairwise constraints: *must-link* constraints that specify pairs of data points which belong in the same cluster, and *cannot-link* constraints that specify pairs which belong in different clusters. Equivalence sets specify *sets* of points that belong in the same cluster. Since it is possible to translate between pairwise constraints and equivalence sets, this thesis focuses on using pairwise constraints.

1.2 Problem Definition and Motivation

The constraints for constrained clustering must be provided by a domain expert. Providing such a relative labeling is expensive, compared to data collection. In order to generate a high-performance clustering, these algorithms often require many pairwise constraints. Most users want to specify as few constraints as possible, so this thesis focuses on providing a high-quality clustering when given few constraints.

Law et al. (2004) note that it is important to propagate the effect of constraints to the nearby neighborhood. Several methods do this by warping a distance metric based on the constraints (Xing *et al.* 2003; Bar-Hillel *et al.* 2005; Basu 2005), which implicitly constrains nearby points. This thesis takes this idea one step further by *explicitly* propagating the constraints to the nearby neighborhood.

Constraint propagation assumes that the specified constraints are representative of their neighborhood. The approach introduced in this thesis, Gaussian Propagated K-Means

(GPK-Means), infers additional constraints in the local neighborhood of the given constraint, with the weight of the propagated constraints decreasing by a Gaussian function as the endpoints move farther from the source constraint. By inferring additional constraints from the source constraints, GPK-Means can generate a higher-quality clustering than other algorithms when given few constraints.

This thesis explores a method for propagating constraints to nearby points using a Gaussian function. The hypothesis of this thesis is that clustering using the propagated constraints will outperform clustering using the original constraints.

1.3 Contribution

This thesis proposes a method for propagating user-specified constraints to nearby instances using a Gaussian function, and provides an algorithm, GPK-Means, that uses these propagated constraints in clustering. GPK-Means can be used with any clustering algorithm that supports weighted constraints.

The experiments in this thesis compare the performance of clustering with propagated constraints to several constrained clustering methods (Bilenko, Basu, & Mooney 2004). These experimental results support the hypothesis that clustering with the propagated constraints can result in improved clustering performance than using only the source constraints.

Chapter 2

BACKGROUND

2.1 Notation

This section introduces notation that will be used throughout the thesis. This notation is based on that of Bilenko, Basu, and Mooney (2004).

$\mathcal{X} = \{x_i\}_{i=1}^{N}, x_i \in \mathbb{R}^n$, is the set of all data instances. μ_{x_i}, Σ_{x_i}, and matrix A_{x_i} represent the centroid, covariance matrix, and metric matrix, respectively, of the cluster to which instance x_i belongs. Similarly, μ_h, Σ_h, and matrix A_h represent the centroid, covariance matrix, and metric matrix, respectively, for cluster $h \in \{1, \ldots, K\}$. The radius $radius_h$ of cluster h is the Mahalanobis distance from the centroid μ_h to the farthest point assigned to the cluster.

$\mathcal{M}$ and $\mathcal{C}$ are the sets of must-link and cannot-link constraints, respectively. $\mathcal{M}_h \subseteq \mathcal{M}$ and $\mathcal{C}_h \subseteq \mathcal{C}$ refer to the respective sets of must- and cannot-link constraints that involve points currently assigned to the cluster h. The notation $\langle x_i, x_j, w \rangle \in \mathcal{M}$ means that x_i and x_j are required to be in the same cluster, with a penalty cost w for violating the constraint. Similarly, $\langle x_i, x_j, \overline{w} \rangle \in \mathcal{C}$ implies that x_i and x_j must be in different clusters, with a penalty cost $\overline{w}$.

The function $\mathbb{1}[expr]$ returns 1 when $expr$ evaluates to true, and 0 when it evaluates to false. The pair (x'_h, x''_h) are defined as the points with the greatest separation using the

A_h metric. The Mahalanobis distance between x_i and μ_{x_i} using the metric A_{x_i} is notated as $||x_i - \mu_{x_i}||_{A_{x_i}}$, where

$$||x_i - \mu_{x_i}||_{A_{x_i}} = \sqrt{(x_i - \mu_{x_i})^T A_{x_i} (x_i - \mu_{x_i})} \ . \tag{2.1}$$

For convenience, Table 2.1 summarizes the notation used in this thesis.

2.2 Semi-Supervised Clustering

Clustering is the unsupervised grouping of similar instances. Each cluster is defined by its center, or centroid, μ. Clustering methods can be grouped into *partitioning* clustering algorithms, which construct flat clusters of the data, and *hierarchical* clustering algorithms, which generate a hierarchical grouping of the data instances.

While traditional clustering algorithms are unsupervised, semi-supervised clustering generates a model for a set of unlabeled data, aided by *side-information* for some of the data. Semi-supervised clustering algorithms use the side-information to reduce the search through the space of possible clusterings, which increases the accuracy of the final clustering relative to the side-information.

The side-information includes specific or relative cluster labels for some data items. Specific cluster labels are equivalent to class labels. Relative cluster labels define an equivalence relation between data items, where all items within an equivalence class belong in the same cluster. These equivalence relations typically take the form of constraints between pairs of data items, specifying whether the pair belongs in the same cluster (a *must-link* constraint) or in different clusters (a *cannot-link* constraint). Pairwise constraints and equivalence sets are discussed further in Section 2.4. With relative cluster labels, the domain expert providing the constraints does not need to know the number of clusters. Semi-supervised clustering using equivalence relations is known as *constrained clustering*.

2.3 K-Means Clustering

The work presented in this thesis is based on the *K-Means* partitional clustering algorithm (MacQueen 1967), which groups the data into K clusters, creating a K-partitioning of the data set. K-Means begins by picking K initial seed centroids, and then iteratively refines the clustering by repeatedly assigning each instance to the nearest centroid, then recomputing each centroid as the mean of the instances assigned to that cluster. The K-Means algorithm is given in Figure 2.1.

```
Algorithm: K-Means

Inputs:
  • the data set $\mathcal{X}$, and
  • the number of clusters $K$.
Output:
  • a disjoint $K$-partitioning $\{\mathcal{X}_h\}_{h=1}^K$ of $\mathcal{X}$ .
Method:
  1. Choose $K$ initial cluster centroids $\{\mu_1, \mu_2, \ldots, \mu_K\}$.
  2. Repeat until no change in $\{\mu_1, \mu_2, \ldots, \mu_K\}$:
       (a) Assign each instance $x_i$ to the cluster of the nearest centroid.
           For each cluster $h$, let $\mathcal{X}_h$ be the set of instances assigned to that
           cluster.
       (b) Recompute each $\mu_h$ as the mean of the instances assigned to that
           cluster $\mathcal{X}_h$.
  3. Return the clusters $\{\mathcal{X}_h\}_{h=1}^K$.
```

FIG. 2.1. The K-Means clustering algorithm.

Traditionally, the initial seed centroids are chosen randomly from the data set, but recent work has focused on developing better selection methods. The final clusters determined by K-Means are sensitive to the initial seed centroids. The labeled data provided to semi-supervised clustering can be used to seed these centroids, resulting in better clusters than by using random centroids (Basu, Banerjee, & Mooney 2002).

A K-Means model is equivalent to a Gaussian mixture model under the assumptions that the prior probability of each cluster (i.e., each Gaussian) is equal and that each Gaussian has an identity covariance (Kearns, Mansour, & Ng 1997; Hastie, Tibshirani, & Friedman 2001). K-Means can be viewed as trying to minimize the total mean squared error of each point to its assigned cluster by minimizing the objective function (Duda, Hart, & Stork 2001):

$$\mathcal{J}_{Kmeans} = \sum_{h=1}^{K} \sum_{x_i \in \mathcal{X}_h} ||x_i - \mu_h||^2 \quad . \tag{2.2}$$

Equivalently, using the notation conventions described earlier,

$$\mathcal{J}_{Kmeans} = \sum_{x_i \in \mathcal{X}} ||x_i - \mu_{x_i}||^2 \quad . \tag{2.3}$$

The computational complexity of the standard K-Means algorithm is $O(NKnT)$, where N is the size of the data set, n is the number of dimensions, and T is the number of iterations to convergence (Duda, Hart, & Stork 2001). Elkan (2003) provides an accelerated K-Means algorithm that uses the triangle inequality to avoid unnecessary computations. The accelerated K-Means algorithm has an empirical complexity closer to $O(N)$.

2.4 Equivalence Relations in Constrained Clustering

As mentioned previously, constrained clustering algorithms typically take side-information in the form of *pairwise constraints* or *equivalence sets*. Both constraints and equivalence sets form an equivalence relation over points in the data set.

Recall that *must-link* constraint $\langle x_i, x_j, w\rangle$ specifies that points x_i and x_j belong in the same cluster. This constraint can be violated with a penalty cost w. Similarly, a *cannot-link* constraint $\langle x_i, x_j, \overline{w}\rangle$ specifies that points x_i and x_j belong in different clusters, with a penalty cost $\overline{w}$ for placing them in the same cluster. The set of all must-link constraints is notated as $\mathcal{M}$, and the set of all cannot-link constraints is notated as $\mathcal{C}$.

Equivalence sets specify *sets* of points that belong in the same cluster. Each cluster may have more than one equivalence set; points in different equivalence sets may still belong in the same cluster. Bar-Hillel et al. (2005) generate equivalence sets by taking the transitive closure of a set of must-link constraints. Each connected component then becomes an equivalence set.

It is simple to extract pairwise constraints from equivalence sets, and simple to generate equivalence sets from a set of must-link constraints. Constructing an equivalence set using cannot-link constraints is more difficult, because cannot-link constraints are not transitive. Bar-Hillel et al. (2005) argue against the use of cannot-link constraints, because must-link constraints are more informative, and because the use of cannot-link constraints imposes an increased computational cost. Their method, Relevant Component Analysis (RCA), uses equivalence sets generated from must-link constraints only.

2.5 Approaches to Constrained Clustering

2.5.1 Hard Constrained Clustering

Wagstaff (2002) proposed the COP-Kmeans algorithm, which enforces the pairwise constraints during K-Means clustering. At initialization, K random instances are chosen as the starting centroids, such that no constraints are violated. COP-Kmeans follows the K-Means algorithm described in Figure 2.1 with one modification: COP-Kmeans assigns points to the nearest cluster *such that no constraint is violated*. The algorithm aborts if such an assignment is not possible. COP-Kmeans uses both must-link and cannot-link pairwise constraints.

COP-Kmeans is a *hard* constrained clustering algorithm, since it does not allow any constraint violations. In constrast, *soft* constrained clustering algorithms allow constraint violations, typically with some penalty. Wagstaff (2002) has also developed a soft constrained clustering version of COP-Kmeans, called SCOP-Kmeans.

2.5.2 Soft Constrained Clustering

Bilenko, Basu, and Mooney (2004) have developed the MPCK-Means algorithm for soft pairwise constrained clustering with metric learning. MPCK-Means uses weighted must-link and cannot-link constraints; a constraint can be violated with a penalty equal to its weight. Bilenko et al.'s algorithm locally minimizes an objective function that incorporates constraint violations with the K-Means objective function (Equation 2.3). During the clustering process, MPCK-Means learns a distance metric. It can learn either a single distance metric for all clusters, or one metric for each cluster. Details on the MPCK-Means algorithm are given later in Section 2.6; MPCK-Means forms the basis for the implementation of the method proposed in this thesis.

Basu et al. (2005; 2004) have also developed a probabilistic framework for semi-supervised clustering that is closely related to MPCK-Means. This model formalizes the combination of constraint-based and distance-based clustering used in MPCK-Means, using ideas from Hidden Markov Random Fields.

Shental et al. (2004) propose a constrained Expectation-Maximization (EM) procedure that fits a Gaussian mixture model to a data set. They provide an EM algorithm for using only must-link constraints, and a generalized EM algorithm for use with both must-link and cannot-link constraints. Lange et al. (2005) provide an alternative method of incorporating soft constraints into fitting a mixture model using maximum likelihood. Their method models both must-link and cannot-link constraints using maximum entropy, while bounding the number of violated constraints. Lu and Leen (2005) propose an alternative method of fitting a Gaussian mixture model using EM. Lange et al. and Lu et al. have similar approaches that incorporate the constraint information into the prior probabilities of the data to each mixture component.

2.5.3 Distance Metric Learning

Xing et al. (2003) use gradient ascent combined with iterative projections to learn a Mahalanobis metric. Their method learns a metric A that minimizes the total mean squared error for must-linked points, such that other "dissimilar" points are not collapsed into a single location. It explicitly uses must-link constraints, and uses cannot-link constraints only to identify "dissimilar" points. In the absence of cannot-link constraints, their method assumes that all pairs of points that are not must-linked can be considered "dissimilar." This assumption is incorrect in most situations, since points which are not must-linked might still belong in the same cluster.

Bar-Hillel et al. (2005) use Relevant Component Analysis (RCA) (Shental *et al.* 2002) with must-link equivalence sets to learn a Mahalanobis metric. Bar-Hillel et al. argue against using cannot-link constraints, based on an argument that a cannot-link constraint provides less information than a must-link constraint, and due to the increased computational cost of using cannot-link constraints. Their algorithm uses only must-link constraints given as *equivalence sets* of points that belong in the same cluster. RCA's computation is similar to the optimization problem solved by Xing et al. (2003), but is more computationally efficient, and prohibits the volume of the entire data set (rather than just the "dissimilar" points) from collapsing. In this manner, the RCA method avoids the problematic assumption made in Xing et al.'s method. Both Xing et al. and Bar-Hillel et al. first train the distance metric based on the labeled data, and then use the distance metric to cluster the entire data set.

The constrained complete-link algorithm presented by Klein et al. (2002) uses pairwise constraints to warp a similarity matrix of the data points. It forces must-linked points to have a distance of zero and cannot-linked points to have the maximum distance of all pairs of points. After each distance adjustment, it warps the similarity matrix by computing the shortest distance between every pair of points. Klein et al.'s method then performs hierarchical complete-link clustering using the warped similarity matrix.

2.5.4 Other Related Work

Several methods have been developed for active learning of constraints during clustering (Basu, Banerjee, & Mooney 2004; Klein, Kamvar, & Manning 2002). Cohn et al.'s (2003) approach to semi-supervised clustering combines interaction with the user with EM to cluster the data, repeatedly clustering and then warping the distance metric in response to user feedback on the clustering.

In a different approach, Zhu et al. (Zhu, Ghahramani, & Lafferty 2003; Zhu, Lafferty, & Ghahramani 2003) propose a framework for semi-supervised learning with binary class labels using Gaussian random fields and Gaussian processes.

2.6 Details on PCK-Means and MPCK-Means

The implementation used in the experiments builds on the MPCK-Means algorithm of Bilenko, Basu, and Mooney (2004). This algorithm combines constrained K-Means clustering with metric learning by minimizing a single objective function. MPCK-Means uses soft pairwise constraints between instances both to seed the initial cluster centroids and to influence the clustering via the objective function. Bilenko et al.'s (2004) PCK-Means algorithm is effectively MPCK-Means without the metric learning component.

Bilenko et al. represent Euclidean distance with a symmetric positive-definite metric matrix A. A is a Mahalanobis metric, based on Equation 2.1. As a reminder, $\mathcal{X}$ is the set of data instances; $\mathcal{M}$ and $\mathcal{C}$ are the sets of must-link and cannot-link constraints, respectively; μ_h represents the centroid of the cluster h, where $h \in \{1, \ldots, K\}$; and matrix A_h represents the metric matrix for cluster h. MPCK-Means generates a K-partitioning of the data set $\mathcal{X}$ that (locally) minimizes the objective function (Bilenko, Basu, & Mooney 2004):

$$
\begin{aligned}
\mathcal{J}_{MPCKmeans} &= \sum_{x_i \in \mathcal{X}} \Big(||x_i - \mu_{x_i}||^2_{A_{x_i}} - log(det(A_{x_i})) \Big) \\
&+ \sum_{\langle x_i, x_j, w \rangle \in \mathcal{M}} w f_{\mathcal{M}}(x_i, x_j) \mathbb{1}[\mu_{x_i} \neq \mu_{x_j}] \qquad (2.4) \\
&+ \sum_{\langle x_i, x_j, \overline{w} \rangle \in \mathcal{C}} \overline{w} f_{\mathcal{C}}(x_i, x_j) \mathbb{1}[\mu_{x_i} = \mu_{x_j}]
\end{aligned}
$$

$$f_{\mathcal{M}}(x_i, x_j) = \frac{1}{2}||x_i - x_j||^2_{A_{x_i}} + \frac{1}{2}||x_i - x_j||^2_{A_{x_j}} \qquad (2.5)$$

$$f_{\mathcal{C}}(x_i, x_j) = ||x'_{x_i} - x''_{x_i}||^2_{A_{x_i}} - ||x_i - x_j||^2_{A_{x_i}} \quad . \qquad (2.6)$$

This equation is the same as used by Bilenko et al. with one minor modification: the sets W and $\overline{W}$ described in their paper (Bilenko, Basu, & Mooney 2004) have been eliminated by incorporating the weights into the individual constraints.

The first term of $\mathcal{J}_{MPCKmeans}$ is an attempt to maximize the log-likelihood of the K-

Means clustering (see Equation 2.3). The second and third terms incorporate the costs of violating the constraints in $\mathcal{M}$ and $\mathcal{C}$.

The MPCK-Means algorithm uses Expectation-Maximization (EM) to generate the clustering of $\mathcal{X}$ that locally minimizes $\mathcal{J}_{MPCKmeans}$. The E-step consists of assigning each point to the cluster that minimizes $\mathcal{J}_{MPCKmeans}$ from the perspective of that data point, given the previous assignments of points to clusters. The M-step consists of two parts: re-estimating the cluster centroids given the E-step cluster assignments, and updating the metric matrices $\{A_h\}_{h=1}^{K}$ to decrease $\mathcal{J}_{MPCKmeans}$. Each metric A_h is updated according to the following equation:

$$\begin{aligned} A_h &= |\mathcal{X}_h| \Bigg(\sum_{x_i \in \mathcal{X}_h} (x_i - \mu_h)(x_i - \mu_h)^T \\ &+ \sum_{\langle x_i, x_j, w \rangle \in \mathcal{M}_h} w f'_{\mathcal{M}}(x_i, x_j) \mathbb{1}\left[\mu_{x_i} \neq \mu_{x_j}\right] \\ &+ \sum_{\langle x_i, x_j, \overline{w} \rangle \in \mathcal{C}_h} \overline{w} f'_{\mathcal{C}}(x_i, x_j) \mathbb{1}[\mu_{x_i} = \mu_{x_j}] \Bigg)^{-1} \end{aligned} \tag{2.7}$$

$$f'_{\mathcal{M}}(x_i, x_j) = \frac{1}{2}(x_i - x_j)(x_i - x_j)^T \tag{2.8}$$

$$\begin{aligned} f'_{\mathcal{C}}(x_i, x_j) &= \left(\left(x'_h - x''_h\right)\left(x'_h - x''_h\right)^T \right. \\ &- \left. (x_i - x_j)(x_i - x_j)^T \right) . \end{aligned} \tag{2.9}$$

The original paper by Bilenko et al. (2004) includes further details on the MPCK-Means algorithm and the initialization steps for cluster seeding.

2.7 Details on Relevant Component Analysis

Relevant Component Analysis (RCA) (Shental *et al.* 2002) uses equivalence sets to “identify and down-scale global unwanted variability within the data (Bar-Hillel *et al.* 2005).” It generates a Mahalanobis metric for the data set from the equivalence sets. RCA uses the equivalence sets to identify and emphasize important dimensions by assigning them large weights in the metric. RCA computes the metric A as follows (Bar-Hillel *et al.* 2005):

$$A = \frac{1}{N} \sum_{j=1}^{|E|} \sum_{i=1}^{|E_j|} (x_{ji} - m_j)(x_{ji} - m_j)^T \quad , \tag{2.10}$$

where E is the set of equivalence sets. The jth equivalence set E_j contains a set of points $E_j = \{x_{ji}\}_{i=1}^{|E_j|}$ that belong in the same cluster. m_j denotes the mean of E_j.

This metric A can then be used directly as the Mahalanobis metric for all clusters. The full RCA algorithm (Shental *et al.* 2002) includes optional dimensionality reduction of the data set. Bar-Hillel et al. (2005) provide details on applying RCA to constrained clustering.

2.8 Clustering Evaluation

Measuring the performance of constrained clustering requires a measure of agreement between the desired clustering (as viewed by the domain expert providing the constraints) and the generated clustering. Following the methodology of other researchers, this thesis uses two objective measures to evaluate the clustering: the pairwise F-Measure (Equation 2.13) and the adjusted Rand index (Equation 2.15).

2.8.1 Pairwise F-Measure

The pairwise F-Measure (Equation 2.13) (Basu 2005) is the typical information-theoretic F-Measure, adapted to measure the number of same-cluster pairs. The F-Measure is the harmonic mean of precision (Equation 2.11) and recall (Equation 2.12). The pairwise F-measure has been used by other researchers to evaluate constrained clustering (Basu 2005).

$$precision = \frac{NumPairsCorrectlyPredictedInSameCluster}{NumTotalPairsPredictedInSameCluster} \tag{2.11}$$

$$recall = \frac{NumPairsCorrectlyPredictedInSameCluster}{NumTotalPairsInSameCluster} \tag{2.12}$$

$$\textit{F-Measure} = \frac{2 \cdot precision \cdot recall}{precision + recall} \tag{2.13}$$

2.8.2 Adjusted Rand Index

The adjusted Rand index (Hubert & Arabie 1985) measures the agreement between the partitions imposed by the class labels and the partitions generated by the clustering. It is related to the Rand index (Rand 1971), which has previously been used in the evaluation of constrained clustering (Bar-Hillel *et al.* 2005; Xing *et al.* 2003; Wagstaff *et al.* 2001). The general form of the adjusted Rand index is:

$$ARI = \frac{Index - ExpectedIndex}{MaximumIndex - ExpectedIndex} \quad . \tag{2.14}$$

The form of the adjusted Rand index used in the evaluation is based on the confusion matrix for the clustering, where c_{ij} is the number of data points from the ith class placed in the jth cluster, and N is the total number of clustered data points (Yeung & Ruzzo 2001):

(2.15)
$$ari = \frac{\sum_{i,j}\binom{c_{ij}}{2} - \left[\sum_i \binom{c_{i\cdot}}{2} \sum_j \binom{c_{\cdot j}}{2}\right] / \binom{N}{2}}{\frac{1}{2}\left[\sum_i \binom{c_{i\cdot}}{2} + \sum_j \binom{c_{\cdot j}}{2}\right] - \left[\sum_i \binom{c_{i\cdot}}{2} \sum_j \binom{c_{\cdot j}}{2}\right] / \binom{N}{2}} \ .$$

N	The number of data instances.
n	The number of dimensions of each data instance.
K	The number of clusters.
x_i	A data instance, $x_i \in \mathbb{R}^n$.
$\mathcal{X}$	The set of data instances $\mathcal{X} = \{x_i\}_{i=1}^{N}, x_i \in \mathbb{R}^n$.
$\mathcal{X}_h$	The set of data instances assigned to cluster h, where $h \in \{1, \ldots, K\}$.
μ_h	The centroid of the cluster h.
A_h	The metric matrix for cluster h.
Σ_h	The covariance matrix for cluster h.
(x'_h, x''_h)	The points with the greatest separation using the A_h metric.
$radius_h$	The Mahalanobis distance from μ_h to the farthest point assigned to cluster h.
μ_{x_i}	The centroid of the cluster to which x_i belongs.
A_{x_i}	The metric matrix of the cluster to which x_i belongs.
Σ_{x_i}	The covariance matrix for the cluster to which x_i belongs.
(x'_{x_i}, x''_{x_i})	The points with the greatest separation using the A_{x_i} metric.
$\mathcal{M}$	The set of must-link constraints.
$\mathcal{C}$	The set of cannot-link constraints.
$\mathcal{M}_h$	The set of must-link constraints involving points assigned to cluster h.
$\mathcal{C}_h$	The set of cannot-link constraints involving points assigned to cluster h.
$\langle x_i, x_j, w \rangle$	A constraint between the instances x_i and x_j with penalty cost w.
$[0]$	The $n \times n$ zero matrix.
$\|\|x_i - \mu_{x_i}\|\|_{A_{x_i}}$	The Mahalanobis distance (Equation 2.1) between x_i and μ_{x_i}.
$\mathbb{1}[expr]$	Returns 1 if $expr$ evaluates to true, 0 otherwise.

Table 2.1. Summary of notation.

Chapter 3

CLUSTERING WITH PROPAGATED CONSTRAINTS

3.1 Method Overview

The novel method presented in this thesis, called Gaussian Propagated K-Means (GPK-Means), requires an unlabeled data set, and sets of pairwise must-link and cannot-link constraints over that data set. Using a Gaussian function and the current cluster estimates, the algorithm infers new constraints in the neighborhoods of the original source constraints, then uses these constraints in clustering to generate new estimates for the clusters. GPK-Means can wrap around any partitional constrained clustering algorithm that uses weighted constraints; this thesis uses MPCK-Means as the base clustering algorithm.

3.2 The Gaussian Function for Propagating Constraints

For a given constraint $\langle x_A, x_B \rangle$, the constraint can potentially be propagated to two other related points, x_i and x_j. The weight of this new constraint should fall off smoothly, as the constraint $\langle x_i, x_j \rangle$ moves farther away from $\langle x_A, x_B \rangle$. GPK-Means uses a Gaussian centered at the given constraint $\langle x_A, x_B \rangle$ to determine the weight of $\langle x_i, x_j \rangle$, because a Gaussian function will emphasize propagated constraints that are closest to the source constraint and will fall off smoothly. Gaussian functions have been used similarly for weighting in other successful applications (Lowe 2004).

The standard d-dimensional Gaussian density function is:

$$N(v, u, \sigma) = exp\left[-\frac{1}{2}(v-u)^T\sigma^{-1}(v-u)\right] \quad , \tag{3.1}$$

where u is the d-dimensional mean and σ is the $d \times d$ covariance matrix.

The weight of the propagated constraint should decrease as the *pair* $\langle x_i, x_j\rangle$ moves farther from $\langle x_A, x_B\rangle$. Since each data point is n-dimensional, constraint propagation centers the Gaussian at a point in $2n$-dimensional space, accounting for the *pair* of points. Using a Gaussian allows the weight to fall off smoothly from 1 at the source constraint. Under this construction, $v = \begin{bmatrix} x_i \\ x_j \end{bmatrix}$, and $u = \begin{bmatrix} x_A \\ x_B \end{bmatrix}$. The weight function $W(x_i, x_j, x_A, x_B, \Sigma_{x_A}, \Sigma_{x_B})$ is given as:

$$W(x_i, x_j, x_A, x_B, \Sigma_{x_A}, \Sigma_{x_B}) = N\left(\begin{bmatrix} x_i \\ x_j \end{bmatrix}, \begin{bmatrix} x_A \\ x_B \end{bmatrix}, \Sigma_{x_A x_B}\right) \quad , \tag{3.2}$$

where

$$\Sigma_{x_A x_B} = \begin{bmatrix} \Sigma_{x_A} & [0] \\ [0] & \Sigma_{x_B} \end{bmatrix} \quad , \tag{3.3}$$

Σ_{x_A} is the covariance matrix of the cluster containing x_A, Σ_{x_B} is the covariance matrix of the cluster containing x_B, and $[0]$ denotes the $n \times n$ zero matrix. In Equation 3.2, the order of the pairs $\langle x_i, x_j\rangle$ and $\langle x_A, x_B\rangle$ is important. To circumvent this problem in applying it to clustering, $W()$ must be called with $x_i \prec x_j$ and $x_A \prec x_B$, where $\prec$ is a total ordering on all the points.

Note that this construction of the covariance matrix $\Sigma_{x_A x_B}$ assumes that x_i and x_A are independent of x_j and x_B. Appendix A shows the construction of the covariance matrix

$\Sigma_{x_A x_B}$ given this independence assumption.

As stated by Bilenko et al. (2004), the inverted metric matrix A_h^{-1} corresponds to the covariance matrix Σ_h for the Gaussian mixture component for cluster h. Bar-Hillel et al. (2005) note that in practice, metric learning typically constructs the metric modulo a scale factor. This scaling factor does not make a difference in clustering, since clustering uses relative distances. However, constraint propagation must have *absolute* distances in order to propagate the constraints correctly. The cluster covariance matrices cannot be generated directly from the data, since MPCK-Means takes the violated constraints into account when generating the metric, and the violated constraints depend directly on the current clustering.

Constraint propagation generates absolute distances by scaling the metric A_h^{-1} so that the bulk of the Gaussian fits within the cluster, then using the scaled metric as the covariance matrix for the cluster. The scale is computed by determining the scale factor n_h that will place approximately 99% of the first principal component of the Gaussian closer than the outermost point currently assigned to this cluster. Since the experiments are limited only to diagonal matrices, this scale factor n_h can be computed as:

$$n_h = \frac{radius_h}{3\sigma_{pc1_h}} \quad , \tag{3.4}$$

where σ_{pc1_h} is the standard deviation of the first principal component of the Gaussian, and $radius_h$ is the distance from the centroid μ_h to the farthest point currently assigned to cluster h. Note that three standard deviations is the offset from the mean that corresponds to containing roughly 99.7% of the data values in a normal distribution. As the first principal component is the largest dimension and the covariance matrix is already fit to the cluster, this will scale the Gaussian such that most (approximately 99%) of the cluster is contained within the Gaussian. The tails of the Gaussian will not have a large effect on the clustering, since propagated constraints within the tails will be trivial due to their small weight.

Therefore, the covariance matrix for cluster h is:

$$\Sigma_h = n_h A_h^{-1} \quad . \tag{3.5}$$

The distance that a particular constraint should be propagated varies with the constraint's location within the cluster. For example, a constraint located at the center of a cluster should ideally be propagated all the way to the cluster's edges. However, a constraint located at the edge of a cluster, if it were propagated the same overall distance, would extend into neighboring clusters. Constraint propagation uses another scaling factor s_{x_A} to vary the amount of propagation based on the source constraint's location in the cluster. The scaling factor s_{x_A} for a particular constraint endpoint x_A is defined as the value of a Gaussian centered at the cluster's centroid μ_{x_A} with the cluster's covariance matrix:

$$s_{x_A} = N(x_A, \mu_{x_A}, n_{x_A} A_{x_A}^{-1}) \quad . \tag{3.6}$$

The final Gaussian weighting function $Weight(x_i, x_j, x_A, x_B)$ used in GPK-Means is based on Equations 3.2, 3.3, and 3.6 as follows:

$$Weight(x_i, x_j, x_A, x_B) = W(x_i, x_j, x_A, x_B, s_{x_A} n_{x_A} A_{x_A}^{-1}, s_{x_B} n_{x_B} A_{x_B}^{-1}) \quad . \tag{3.7}$$

Using this equation, constraint propagation can now determine the weight for the propagated constraint between every pair of data points, and use these weighted constraints directly in a constrained clustering algorithm.

By taking advantage of the independence assumptions and using memoization, $Weight(x_i, x_j, x_A, x_B)$ can be calculated efficiently over repeated computations, as shown in Appendix B.

3.3 Using the Propagated Constraints in Clustering

Given a data set $\mathcal{X}$, a set of must-link constraints $\mathcal{M}$, a set of cannot-link constraints $\mathcal{C}$, a set of defined clusters $\{\mathcal{X}_h\}_{h=1}^K$, and the metric matrices for those clusters $\{A_h\}_{h=1}^K$, constraint propagation can now generate new sets of propagated must-link constraints $P(\mathcal{M})$ and cannot-link constraints $P(\mathcal{C})$. By construction, $\mathcal{M} \subseteq P(\mathcal{M})$ and $\mathcal{C} \subseteq P(\mathcal{C})$. Constrained clustering algorithms can directly use the data set $\mathcal{X}$ and the sets of propagated constraints, $P(\mathcal{M})$ and $P(\mathcal{C})$, to generate a new clustering and set of metrics.

GPK-Means first gets an initial estimate of the clusters and metrics from the base clustering algorithm — in this case, MPCK-Means — using the data set $\mathcal{X}$, and the sets of original constraints $\mathcal{M}$ and $\mathcal{C}$. MPCK-Means outputs a set of defined clusters, $\{\mathcal{X}_h^0\}_{h=1}^K$, and the metric matrices for those clusters, $\{A_h^0\}_{h=1}^K$. GPK-Means then propagates the given constraints using the learned clusters and metrics, and runs MPCK-Means again with the new sets of propagated constraints $P(\mathcal{M})^0$ and $P(\mathcal{C})^0$ to generate the clusters $\{\mathcal{X}_h^1\}_{h=1}^K$, and the metrics $\{A_h^1\}_{h=1}^K$. The ith run of MPCK-Means uses $P(\mathcal{M})^{i-1}$ and $P(\mathcal{C})^{i-1}$. After each run of MPCK-Means, excluding the initial seed run, GPK-Means checks for convergence between the final objective function values of MPCK-Means. GPK-Means runs repeatedly in this manner until convergence. The GPK-Means algorithm is given in Figure 3.1.

Due to the complex nature of the Gaussian propagation, it is difficult to use EM (as used in MPCK-Means) to ensure convergence. GPK-Means is not required to converge, since there is no strict requirement that the clustering must move closer to the optimal clustering at each step. MPCK-Means suffers from the same problem under some conditions that occur in practice, and is not theoretically guaranteed to converge in these cases (Bilenko, Basu, & Mooney 2004). In practice, however, the convergence of both GPK-Means and MPCK-Means has not been a problem.

The constraint propagation algorithm includes an optional step that reduces the set of propagated constraints. This thesis refers to performing the optional step as *reduced propagation* and skipping the optional step as *full propagation*. Full propagation allows each pair of points to have multiple propagated must-link constraints and multiple propagated cannot-link constraints. Reduced propagation forces the set of propagated constraints to include, for each pair of points, at most one must-link constraint and one cannot-link constraint. The propagated constraint with the maximum weight for a given pair is selected as the constraint in both cases.

3.4 Complexity Analysis of GPK-Means

Figure 3.3 examines the complexity of constraint propagation and Figure 3.4 provides a breakdown of the algorithmic complexity for GPK-Means. Since GPK-Means is composed of repeated runs of a constrained clustering algorithm and constraint propagation, this section analyzes each component in turn. Only an upper bound on the computational complexity is provided, because determining an exact lower bound for any K-Means variant is an open question (Elkan 2003).

3.4.1 Complexity of Constraint Propagation

This section examines the computational complexity of constraint propagation. Figure 3.3 shows the computational complexity for the primary steps of the $Propagate$ algorithm given in Figure 3.2.

This analysis assumes that the time taken to compute the Gaussian functions $W()$ and $N()$ depends only on the dimensions of the data set, and therefore represents the complexity of these functions as $\Theta(G(n))$ for an n-dimensional data set. $G(n)$ will vary depending on the specific implementation, but will generally be a low-order polynomial.

The determination of the scale factors for each cluster in Step 1 takes $\Theta(|\mathcal{X}| + Kn)$

time, where K is the number of clusters and n is the number of dimensions of the data set. Determining the radius of each cluster takes $\Theta(|\mathcal{X}|)$ time, and determining the first principal component for each of the K clusters takes $\Theta(Kn)$ time using diagonal metric matrices. If the farthest point in each cluster is determined and cached during the point assignment step of MPCK-Means, the complexity of Step 1 reduces to $\Theta(Kn)$.

The complexity of $Propagate()$ is primarily determined by the main loop in Step 3, which repeats $|C|$ times, for a set of constraints C. Computing the scale factors within this loop (Steps 3a–3b) requires a combined time of $\Theta(G(n))$, which is dominated by the complexity of Step 3d. Step 3d loops over every pair of data points, of which there are $\Theta(|\mathcal{X}|^2)$. For each pair, the step computes the weight of the propagated constraint between these points in $\Theta(G(n))$ time. For the optional reduction step, $Propagate()$ can track and keep the constraint with the maximum weight during the weight computation step without any additional computational cost. This yields a complexity of $\Theta(|\mathcal{X}|^2 G(n))$ for Step 3d, and therefore a complexity of $\Theta(|C||\mathcal{X}|^2 G(n))$ for the main loop of Step 3. Since this dominates the complexity of $\Theta(Kn)$ for Step 1, the overall complexity for $Propagate$ is $\Theta(|C||\mathcal{X}|^2 G(n))$ for both full and reduced propagation.

3.4.2 Complexity of GPK-Means

Figure 3.4 shows the computational complexity for the primary steps of the GPK-Means algorithm given in Figure 3.1.

The computational complexity of GPK-Means is highly dependent on the chosen constrained clustering algorithm. This analysis assumes that MPCK-Means has a computational complexity of $O(C_{MPCKmeans})$.[1]

The initial run of MPCK-Means in Step 1 has complexity $O(C_{MPCKmeans})$. GPK-

[1]To the best of my knowledge, no paper on MPCK-Means or any related algorithms analyzes the computational complexity of this algorithm.

Means then repeatedly propagates constraints (Steps 3a–3b) with complexity $\Theta((|\mathcal{M}| + |\mathcal{C}|)|\mathcal{X}|^2 G(n))$ and runs MPCK-Means (Step 3c) with complexity $O(C_{MPCKmeans})$ until convergence. Since the number of iterations taken to converge is not know a priori, let c represent this number. The initial seed run of MPCK-Means is absorbed in the final complexity, yielding a computational complexity of $O(cC_{MPCKmeans} + c(|\mathcal{M}| + |\mathcal{C}|)|\mathcal{X}|^2 G(n))$ for GPK-Means.

```
Algorithm: GPK-Means

Inputs:
  • the data set $\mathcal{X}$,
  • the set of weighted must-link constraints $\mathcal{M}$,
  • the set of weighted cannot-link constraints $\mathcal{C}$,
  • the number of clusters $K$, and
  • the cutoff threshold for propagation $g$.

Output:
  • a disjoint $K$-partitioning $\{\mathcal{X}_h\}_{h=1}^K$ of $\mathcal{X}$ .

Method:
  1. Obtain the initial clustering $\{\mathcal{X}_h^0\}_{h=1}^K$, $\{\mu_h^0\}_{h=1}^K$, and initial metrics
     $\{A_h^0\}_{h=1}^K$ by running MPCK-Means with $\mathcal{X}$, $\mathcal{M}$, $\mathcal{C}$, and $K$.
  2. Set $i := 0$.
  3. Repeat until the MPCK-Means objective function values converge:
     (a) Propagate the must-link constraints:
         $P(\mathcal{M})^i := Propagate(\mathcal{M}, \{\mathcal{X}_h^i\}_{h=1}^K, \{A_h^i\}_{h=1}^K, g)$.
     (b) Propagate the cannot-link constraints:
         $P(\mathcal{C})^i := Propagate(\mathcal{C}, \{\mathcal{X}_h^i\}_{h=1}^K, \{A_h^i\}_{h=1}^K, g)$.
     (c) Set $\{\mathcal{X}_h^{i+1}\}_{h=1}^K$, $\{\mu_h^{i+1}\}_{h=1}^K$, and $\{A_h^{i+1}\}_{h=1}^K$ by running MPCK-
         Means with $\mathcal{X}$, $P(\mathcal{M})^i$, $P(\mathcal{C})^i$, and $K$ starting with initial cen-
         troids $\{\mu_h^i\}_{h=1}^K$ .
     (d) Set $i := i + 1$ .
  4. Return $\{\mathcal{X}_h^i\}_{h=1}^K$ .
```

FIG. 3.1. The GPK-Means algorithm.

```
Algorithm: Propagate

Inputs:
  • the set of weighted constraints $C$,
  • the disjoint $K$-partitioning of the data $\{\mathcal{X}_h^0\}_{h=1}^K$,
  • the set of metrics $\{A_h^0\}_{h=1}^K$, and
  • the cutoff threshold for propagation $g$.

Output:
  • a set of weighted constraints.

Method:
  1. Compute the scale factors $\{n_h\}_{h=1}^K$ using Equation 3.4.
  2. Let the set of propagated constraints be empty: $P(C) = \{\}$.
  3. For each $\langle x_A, x_B, w\rangle \in C$, do:
      (a) Let $s_{x_A} := N(x_A, \mu_{x_A}, n_{x_A} A_{x_A}^{-1})$.
      (b) Let $s_{x_B} := N(x_B, \mu_{x_B}, n_{x_B} A_{x_B}^{-1})$.
      (c) Add the original constraint to the propagated set:
          $P(C) := P(C) \bigcup \{(x_A, x_B, w)\}$.
      (d) For each pair of data points $(x_i, x_j)$ such that $(x_i \neq x_A) \wedge (x_j \neq x_B) \wedge (x_i \prec x_j)$, where $\prec$ is a global ordering to ensure a pair is only checked once, do:
            i. Let $w_{ij1} = W(x_i, x_j, x_A, x_B, s_{x_A} n_{x_A} A_{x_A}^{-1}, s_{x_B} n_{x_B} A_{x_B}^{-1})$.
           ii. Let $w_{ij2} = W(x_j, x_i, x_A, x_B, s_{x_A} n_{x_A} A_{x_A}^{-1}, s_{x_B} n_{x_B} A_{x_B}^{-1})$.
          iii. Let $w_{ij} = max(w_{ij1}, w_{ij2})$.
           iv. If $w_{ij} \geq g$, then add the constraint to the propagated set:
               $P(C) := P(C) \bigcup \{\langle x_i, x_j, w_{ij} w\rangle\}$.
      (e) (Optional) Reduce the number of propagated constraints to only the maximum weighted constraint between each pair of points. For each constraint $\langle x_i, x_j, w\rangle$ such that $\exists \langle x_i, x_j, w'\rangle \in P(C)$ with $w <= w'$, $P(C) := P(C) - \{\langle x_i, x_j, w\rangle\}$.
  4. Return $P(C)$.
```

FIG. 3.2. The constraint propagation algorithm.

Propagate Algorithm: $\Theta(|C||\mathcal{X}|^2 G(n))$.

- Compute the scale factors for each cluster's Gaussian (step 1): $\Theta(Kn)$.
- Loop over each constraint (step 3): $\Theta(|C||\mathcal{X}|^2 G(n))$.
  - Compute scaling factors (steps 3a–3b): $\Theta(G(n))$.
  - Loop over each pair of data points (step 3d): $\Theta(|\mathcal{X}|^2 G(n))$.
    - Computing the constraint weights (steps 3(d)i–3(d)ii): $\Theta(G(n))$.

FIG. 3.3. The computational complexity of constraint propagation.

GPK-Means Algorithm: $O(cC_{MPCKmeans} + c(|\mathcal{M}| + |\mathcal{C}|)|\mathcal{X}|^2 G(n))$.

- Initialize the estimates of the clusters (step 1): $O(C_{MPCKmeans})$.
- Repeat until convergence (c iterations) (step 3): $O(cC_{MPCKmeans} + c(|\mathcal{M}| + |\mathcal{C}|)|\mathcal{X}|^2 G(n))$.
  - Propagate the constraints (steps 3a–3b): $\Theta((|\mathcal{M}| + |\mathcal{C}|)|\mathcal{X}|^2 G(n))$.
  - Run MPCK-Means (step 3c): $O(C_{MPCKmeans})$.

FIG. 3.4. The computational complexity of the GPK-Means algorithm.

Chapter 4

EXPERIMENTS

4.1 Data Sets

The experiments were conducted using five data sets: the Crabs-Gender, Iris, Digits, and Letters data sets from the UCI machine learning repository (Newman *et al.* 1998); and the protein data set used by Xing et al. (2003). Following Bilenko et al. (2004), the Digits and Letters data sets were reduced to include only the items $\{3, 8, 9\}$ and $\{I, J, L\}$, respectively. Table 4.1 summarizes the properties of these data sets.

Name	# Instances	# Features	# Classes
Crabs-Gender	200	5	2
Digits389	317	16	3
Iris	150	4	3
LettersIJL	227	16	3
Protein	116	20	6

Table 4.1. Properties of the data sets.

4.2 Methodology

On each data set, the experiments compared three methods of constrained clustering: *PCK-Means* (MPCK-Means without metric learning), *MPCK-Means*, and *GPK-Means*.

All experiments were conducted using implementations of these algorithms incorporated into the Weka machine learning toolkit (Witten & Frank 2000). The PCK-Means and MPCK-Means algorithms were taken from the University of Texas Weka distribution provided by Bilenko et al. [1] The implementation of GPK-Means is based on the MPCK-Means algorithm from this distribution.

MPCK-Means can use either a single metric for the entire data set or multiple separate metrics for each cluster; consequently, so can GPK-Means. The experiments used both a single diagonal metric and multiple diagonal metrics. All experiments used unit constraint weights for the initial constraints.

GPK-Means was run with two propagation thresholds on each data set; the propagation thresholds are listed in each graph's key. The propagation thresholds were 0.5 and 0.7 for every data set except Protein. Protein required lower propagation thresholds of 0.01 and 0.05 to obtain any effect, because of the sparsity of the data set.

For each data set, learning curves were generated for each algorithm as the number of constraints was varied. Each data point on the learning curve was averaged over 50 trials of 5-fold cross validation. Constraints were selected randomly from the training set (four folds); the full data set was then clustered; and results were reported for only the test set (the fifth fold). All algorithms were tested on the same five folds with the same constraints.

The experiments compare each algorithm using the pairwise F-Measure (Section 2.8.1) and the adjusted Rand index (Section 2.8.2). Significance testing was performed on the results using 90%, 95%, 97.5%, and 99% confidence levels. Each graph has two components: an upper graph that displays clustering performance as the number of constraints varies, and a lower graph that gives the significance level against MPCK-Means for each data point.

[1]Available on-line at http://www.cs.utexas.edu/users/ml/risc/code/.

4.3 Clustering Evaluation of GPK-Means Using Full Propagation

This section evaluates the clustering performance of GPK-Means using full propagation. In this form of GPK-Means, all inferred constraints are used, even if there are multiple constraints between a pair of points. The experiments evaluated the performance of GPK-Means with full propagation against MPCK-Means and PCK-Means using the pairwise F-Measure and the adjusted Rand index (ARI).

4.3.1 F-Measure Evaluation of GPK-Means Using Full Propagation

The F-Measure (Section 2.8.1) evaluates the clustering from an information-theoretic perspective. Figure 4.1 depicts the F-Measure performance of PCK-Means, MPCK-Means, and GPK-Means on each data set as the number of constraints varies.

GPK-Means outperforms MPCK-Means and PCK-Means on the Iris and Protein data sets with a single metric. GPK-Means shows significant improvements in the F-Measure for a range of low quantities of constraints. As the number of constraints increases, the performance of GPK-Means and MPCK-Means becomes indistinguishable. At higher numbers of constraints, there is no longer a benefit to using the (possibly inaccurate) propagated constraints, because there are enough accurate source constraints to yield a high-performance clustering.

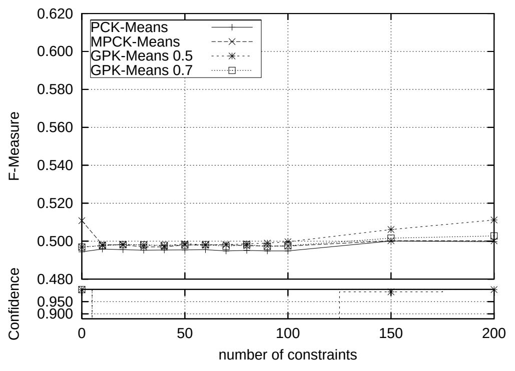
Crabs-Gender and Digits389 also show a benefit with using GPK-Means with single metrics, but over a much smaller range of constraint values. The improvement on Crabs-Gender is very slight and appears only with larger numbers of constraints (above 100). Given the marginal improvement with the data set, it appears that the Crabs-Gender data set is very difficult to cluster using a single metric. LetterIJL shows virtually no difference between using MPCK-Means and GPK-Means.

When using multiple metrics, only the Crabs-Gender and Iris data sets show a benefit of using GPK-Means over MPCK-Means. GPK-Means shows a large improvement over MPCK-Means on Crabs-Gender with multiple metrics. The performance of GPK-Means is indistinguishable from MPCK-Means on the other data sets with multiple metrics.

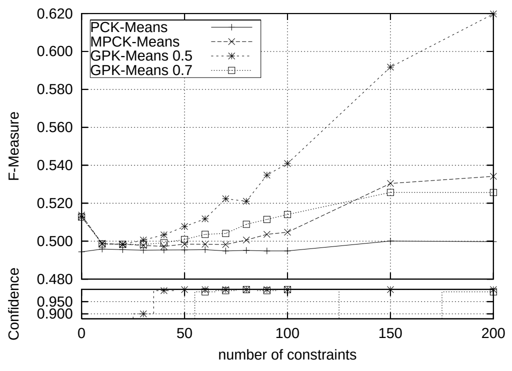
PCK-Means did not perform well on any data set, except initially on Protein against MPCK-Means and GPK-Means using multiple metrics.

Notice the sudden drop in performance as the number of constraints increases on the Iris data set. This problems occurs with full propagation on Iris, because the number of propagated constraints tends to increase rapidly with the number of source constraints. The huge numbers of propagated constraints are likely to be inaccurate, and decrease GPK-Means' performance. As Section 4.4 will show, reduced propagation avoids this problem, and reduces the sudden performance drop.

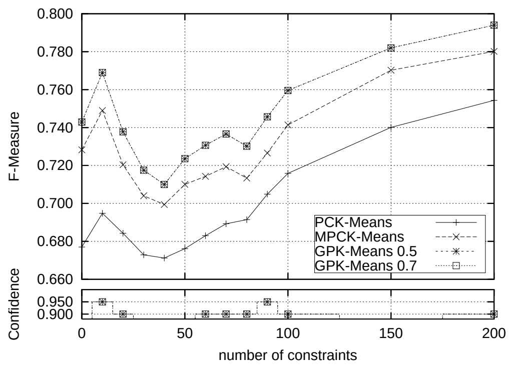
The experiments demonstrate a large performance difference between single and multiple metrics on several of the data sets. In some cases, using a single metric is better (e.g. Iris), while in other cases, using multiple metrics is better (e.g. Crabs-Gender). Multiple metrics are especially useful on data sets where the clusters are different shapes (Bilenko, Basu, & Mooney 2004). Because the benefit of using multiple metrics depends on each data set's underlying clustering, no conclusion can be drawn from these experiments as to whether using single or multiple metrics is generally better.



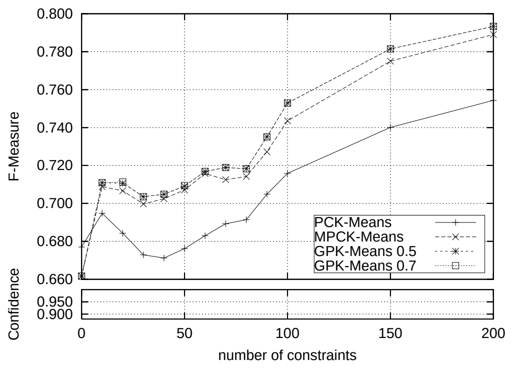
(a) Crabs-Gender - Single Metric



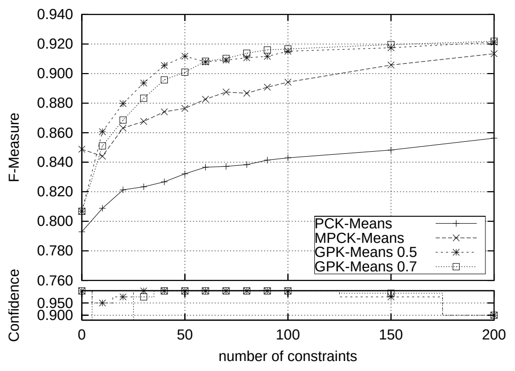
(b) Crabs-Gender - Multiple Metrics



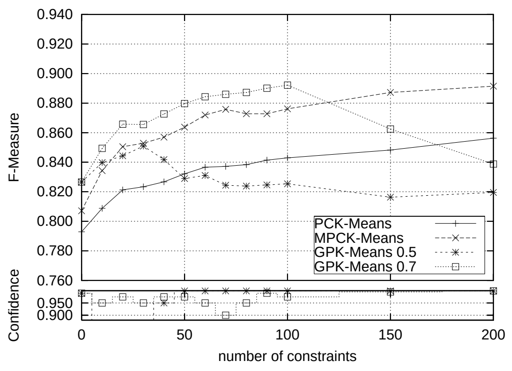
(c) Digits389 - Single Metric



(d) Digits389 - Multiple Metrics

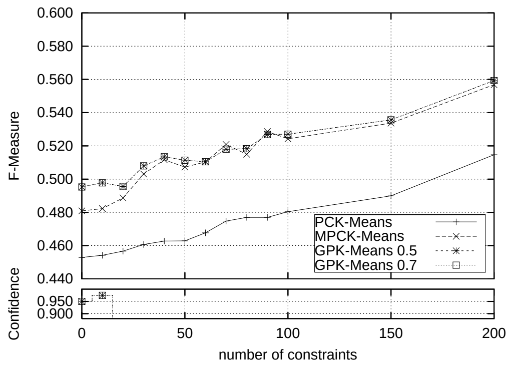


(e) Iris - Single Metric

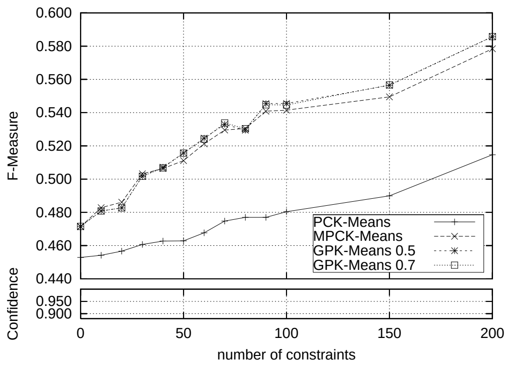


(f) Iris - Multiple Metrics

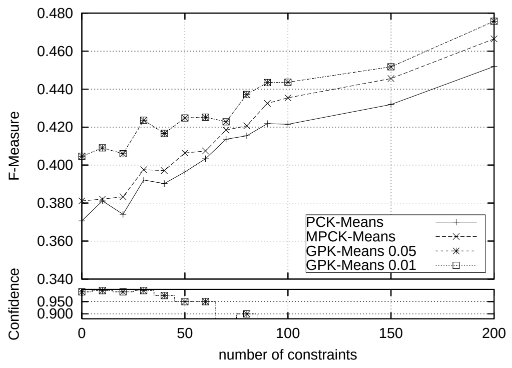
FIG. 4.1. The F-Measure performance of GPK-Means using full propagation. The bottom section of each graph depicts the significance level of GPK-Means against MPCK-Means.



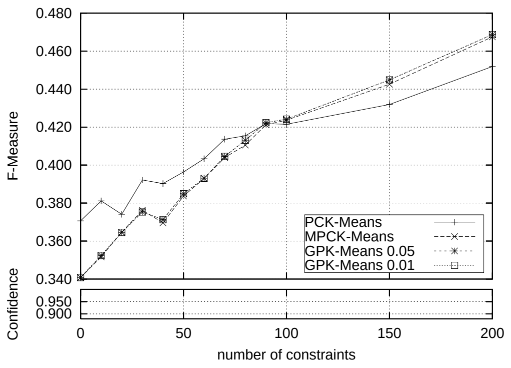
(g) LetterIJL - Single Metric



(h) LetterIJL - Multiple Metrics



(i) Protein - Single Metric



(j) Protein - Multiple Metrics

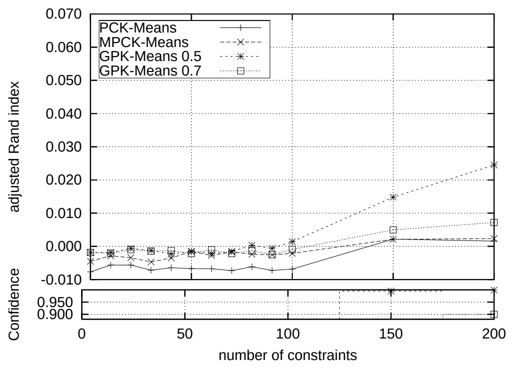
FIG. 4.1 (continued). The F-Measure performance of GPK-Means using full propagation. The bottom section of each graph depicts the significance level of GPK-Means against MPCK-Means.

4.3.2 Adjusted Rand Index Evaluation of GPK-Means Using Full Propagation

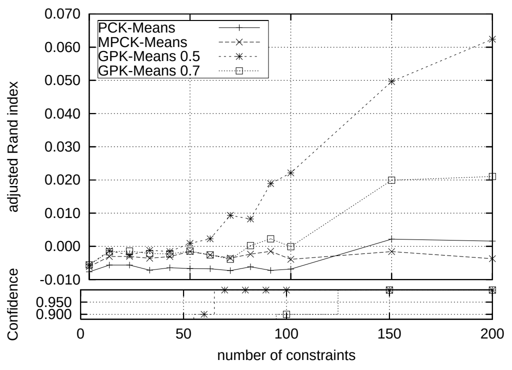
The adjusted Rand index (ARI) (Section 2.8.2) evaluates the level of agreement between the true class labels and the clustering. Figure 4.2 depicts the ARI performance of PCK-Means, MPCK-Means, and GPK-Means with full propagation on each data set as the number of constraints varies.

The ARI results are similar to the F-Measure results, with GPK-Means having significant benefits for low numbers of constraints on the iris data set with both single and multiple metrics. At higher numbers of constraints, the experiments on Iris with multiple metrics show the same sudden performance drop as with the F-Measure.

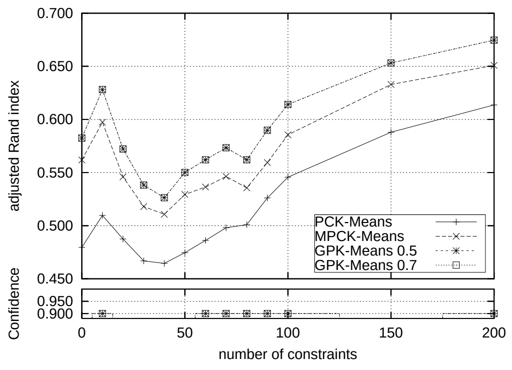
GPK-Means with multiple metrics significantly improves the clustering performance on Crabs-Gender and Iris. There is also an improvement on Crabs-Gender with a single metric for larger numbers of constraints. Protein also shows an improvement with GPK-Means, but only with single metrics. Digits389 and LettersIJL show only a small benefit for limited numbers of constraints. As with the F-Measure, GPK-Means with full propagation using multiple metrics does not have a significant benefit over MPCK-Means on Digits389, LetterIJL, and Protein.



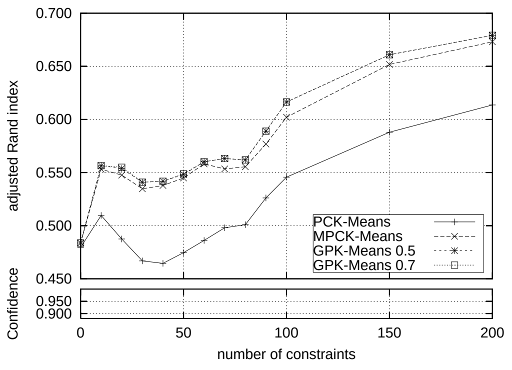
(a) Crabs-Gender - Single Metric



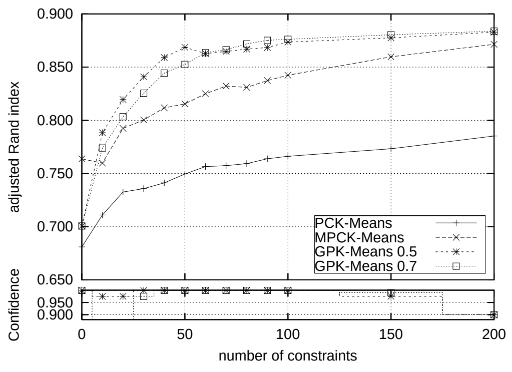
(b) Crabs-Gender - Multiple Metrics



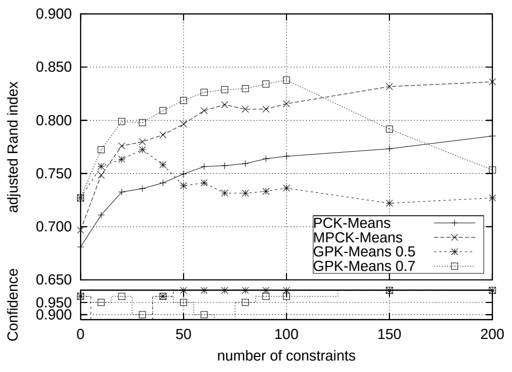
(c) Digits389 - Single Metric



(d) Digits389 - Multiple Metrics

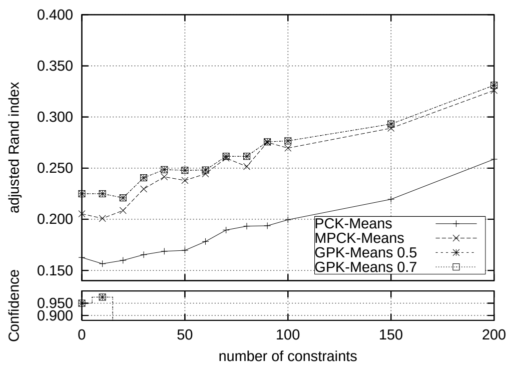


(e) Iris - Single Metric

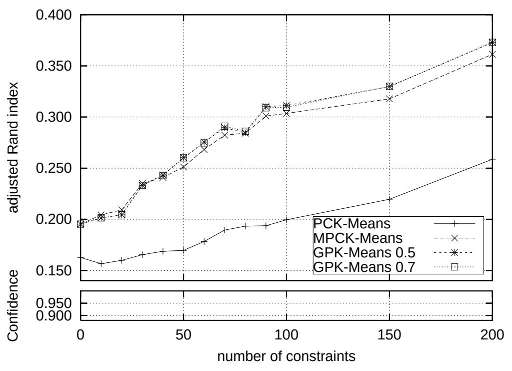


(f) Iris - Multiple Metrics

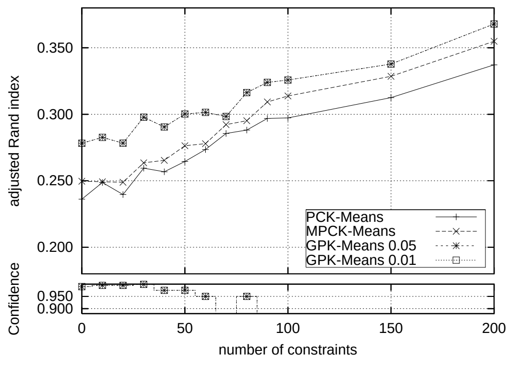
FIG. 4.2. The adjusted Rand index performance of GPK-Means using full propagation. The bottom section of each graph depicts the significance level of GPK-Means against MPCK-Means.



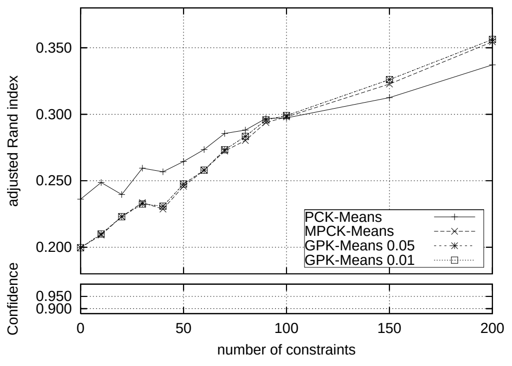
(g) LetterIJL - Single Metric



(h) LetterIJL - Multiple Metrics



(i) Protein - Single Metric



(j) Protein - Multiple Metrics

FIG. 4.2 (continued). The adjusted Rand index performance of GPK-Means using full propagation. The bottom section of each graph depicts the significance level of GPK-Means against MPCK-Means.

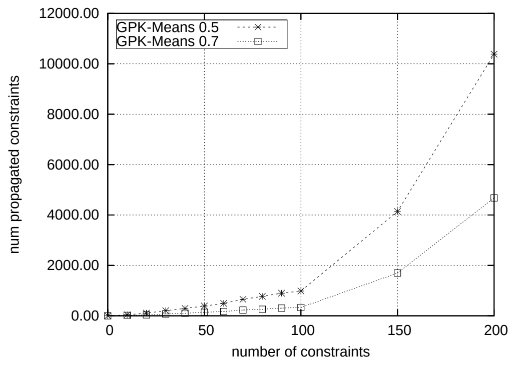
4.3.3 Evaluation of Full Constraint Propagation

Recall that on the Iris data set with multiple metrics, GPK-Means demonstrates a sudden decrease in performance as the number of source constraints increases. To investigate why this occurs, this section examines the number of constraints generated by full propagation as the number of source constraints increases (Figure 4.3). The counts of propagated constraints in Figure 4.3 *exclude* the source constraints, and were measured after the last iteration of constraints propagation in GPK-Means

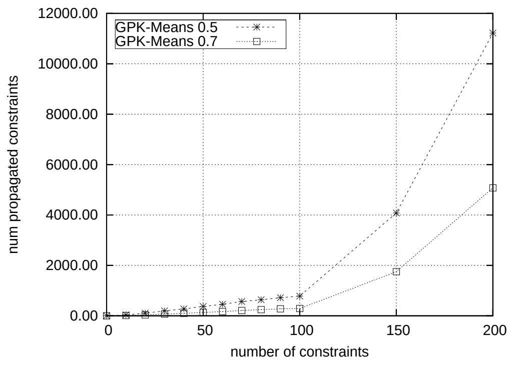
As the number of source constraints increases, the number of propagated constraints leaps dramatically. For the high-dimensional data sets (Digits389, LetterIJL, and Protein), the number of propagated constraints remains low (less than 3 times the number of source constraints). The low density of the high-dimensional data causes the propagation neighborhoods to be very small, resulting in few propagated constraints.

However, for the low-dimensional data sets (Crabs-Gender and Iris), the high density results in many propagated constraints, up to several hundred times the number of source constraints. Using multiple metrics appears to reduce the amount of propagation, because the per-cluster covariance matrices are more accurate to each cluster. With full propagation, multiple different constraints may propagate to the same pair of data points. Therefore, each pair of data points may be constrained multiple times, each with a different weight. The effect of these constraints is cumulative, and they act as one constraint with a cost equal to the sum of the multiple constraint weights. The weight of the summed constraint may be disproportionately high compared to the weight of the source constraints. Section 5.1 discusses this issue in more detail.

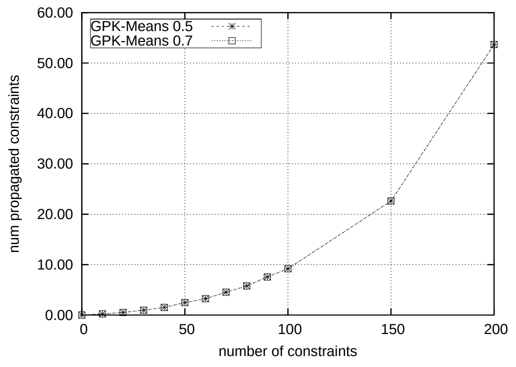
The huge number of constraints likely contains many which are inaccurate, resulting in decreased clustering performance. Also, this increases the computational cost of MPCK-Means, resulting in a longer run-time for the constrained clustering step of GPK-Means.



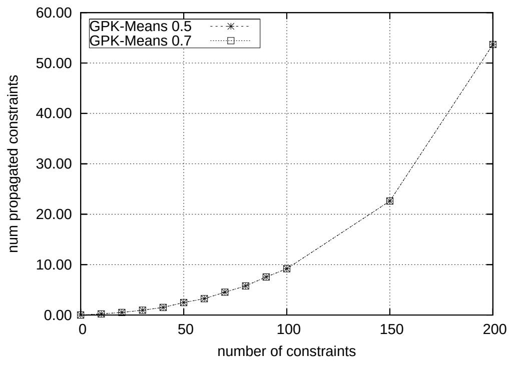
(a) Crabs-Gender - Single Metric



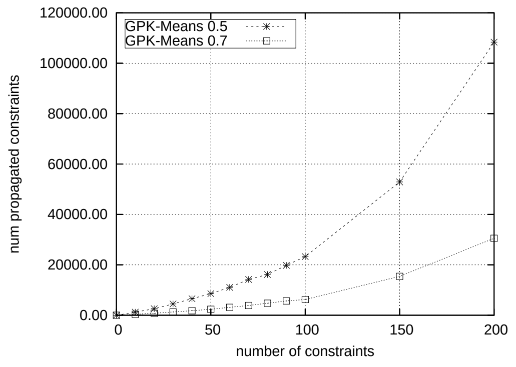
(b) Crabs-Gender - Multiple Metrics



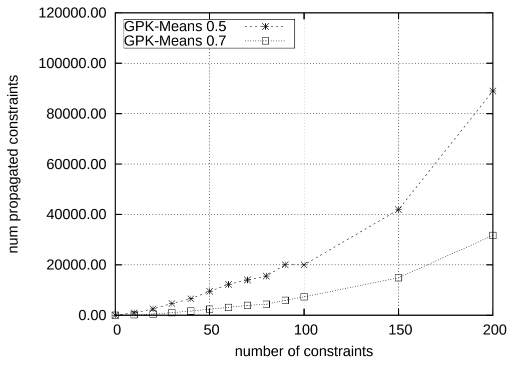
(c) Digits389 - Single Metric



(d) Digits389 - Multiple Metrics

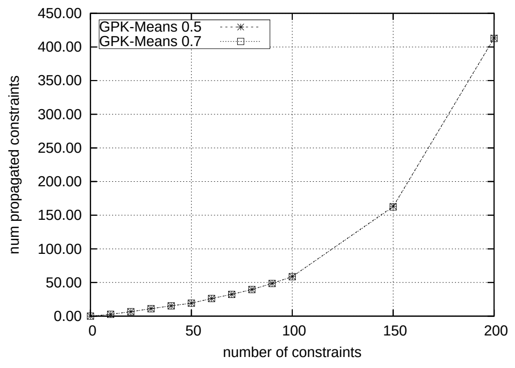


(e) Iris - Single Metric

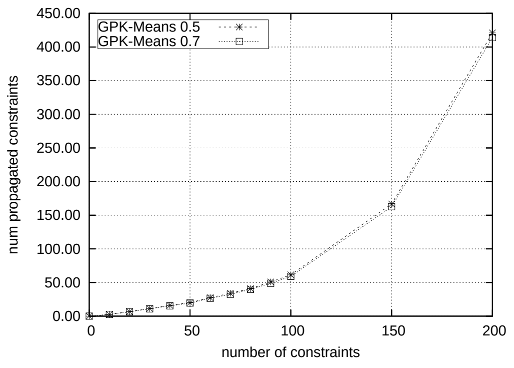


(f) Iris - Multiple Metrics

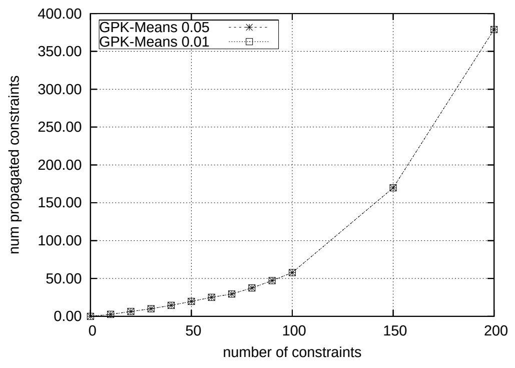
FIG. 4.3. The number of constraints inferred by GPK-Means using full propagation. Note that the counts *exclude* the original source constraints.



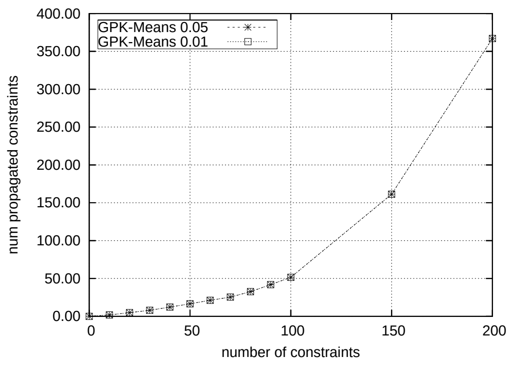
(g) LetterIJL - Single Metric



(h) LetterIJL - Multiple Metrics



(i) Protein - Single Metric



(j) Protein - Multiple Metrics

FIG. 4.3 (continued). The number of constraints inferred by GPK-Means using full propagation. Note that the counts *exclude* the original source constraints.

4.4 Clustering Evaluation of GPK-Means Using Reduced Propagation

The propagation reduction step was introduced into the GPK-Means algorithm to prevent the problem of each pair of data points having multiple associated constraints. This problem occurs frequently in the low-dimensional data sets with larger numbers of constraints, since the propagation neighborhoods are dense. With reduced propagation, if there are multiple inferred (or source) constraints of a particular type between a pair of points, GPK-Means uses only the constraint with the maximum weight. This limits the number of constraints involving a pair of points to be two, one must-link constraint and one cannot-link constraint. This also limits the total number of propagated constraints to be less than or equal to $N^2 - N$, where N is the number of data points. $N^2 - N$ corresponds to two constraints for every pair of data points.

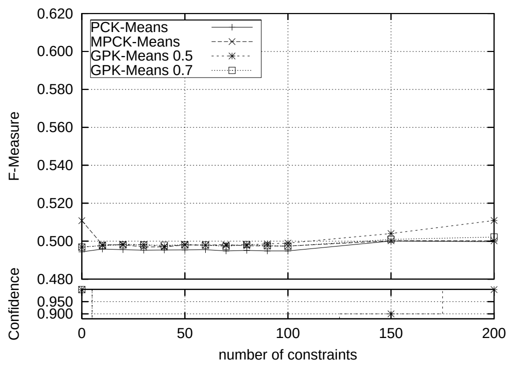
GPK-Means allows a pair of points to be both must-linked and cannot-linked at the same time, since the weight of a must-link constraint between the pair of points is not necessarily one minus the weight of the cannot-link constraint between those points, and vice versa. In practice however, if two points are must-linked with large weight, then the weight of a cannot-link constraint between them is likely to be small. The reverse is also true.

Section 4.4.1 examines the performance of GPK-Means with reduced propagation against MPCK-Means and PCK-Means using the pairwise F-Measure and the adjusted Rand index (ARI).

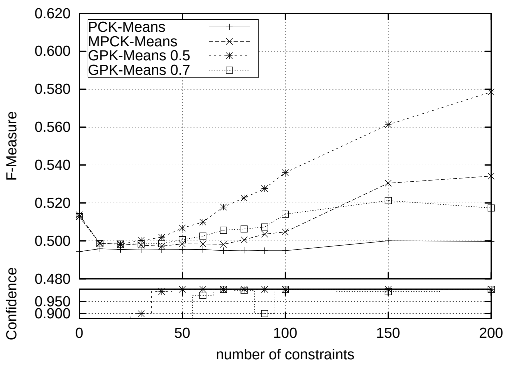
4.4.1 F-Measure and Adjusted Rand Index Evaluation of GPK-Means Using Reduced Propagation

Figures 4.4 and 4.5 depict the F-Measure performance and adjusted Rand index performance, respectively, of PCK-Means, MPCK-Means, and GPK-Means with reduced propagation on each data set as the number of constraints varies.

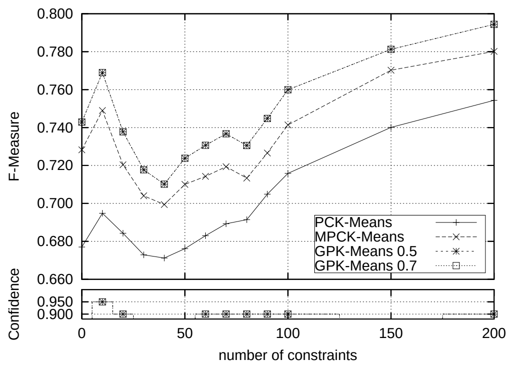
The F-Measure and adjusted Rand index performance GPK-Means using reduced propagation is about the same as GPK-Means using full propagation. It actually shows an improvement over full propagation on Iris with multiple metrics, and a very slight improvement on Protein with multiple metrics. However, reduced propagation decreases the performance of GPK-Means on Crabs-Gender with multiple metrics; GPK-Means with reduced propagation is still significantly better than MPCK-Means on Crabs-Gender for larger quantities of constraints. The most important aspect of Figures 4.4 and 4.5 is that they show a reduction in the severe performance decrease with GPK-Means on the Iris data set. A slight decrease still occurs, since the likelihood of inaccurate propagated constraints increases as the number of constraints increases. The increased number of inaccurate constraints degrades the clustering quality. However, the decrease is not as sudden or as severe as with full propagation.



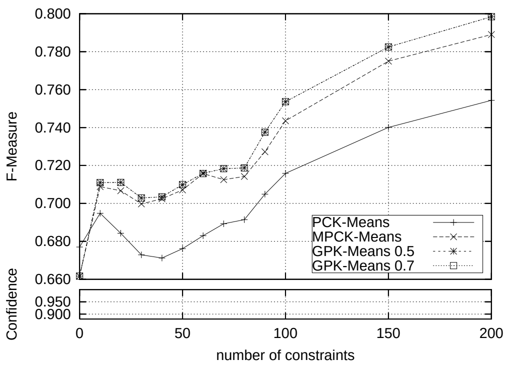
(a) Crabs-Gender - Single Metric



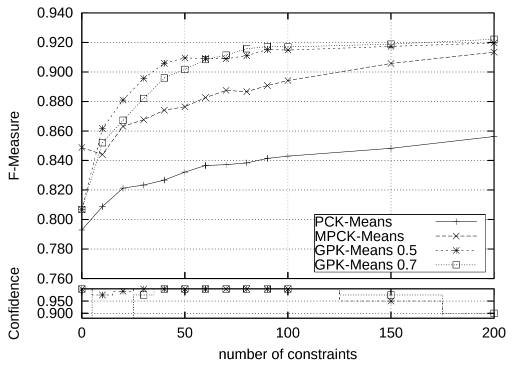
(b) Crabs-Gender - Multiple Metrics



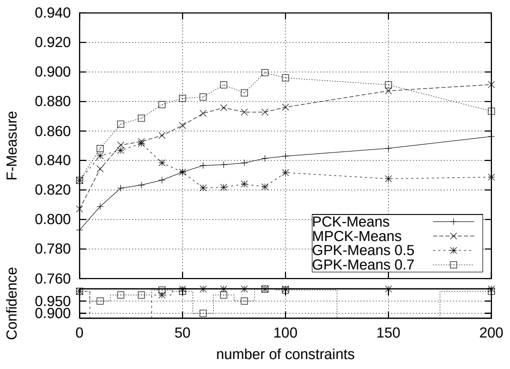
(c) Digits389 - Single Metric



(d) Digits389 - Multiple Metrics

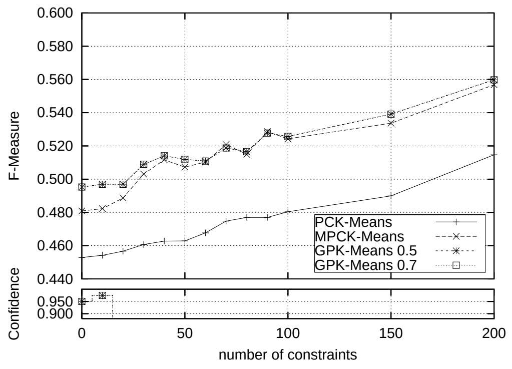


(e) Iris - Single Metric

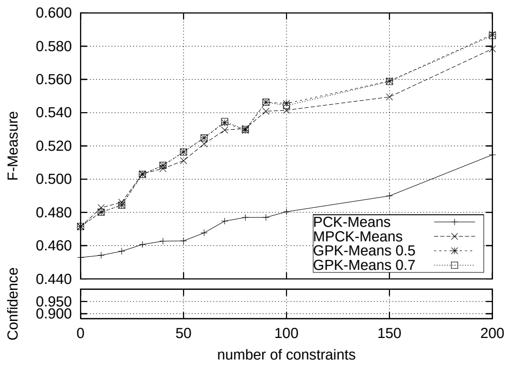


(f) Iris - Multiple Metrics

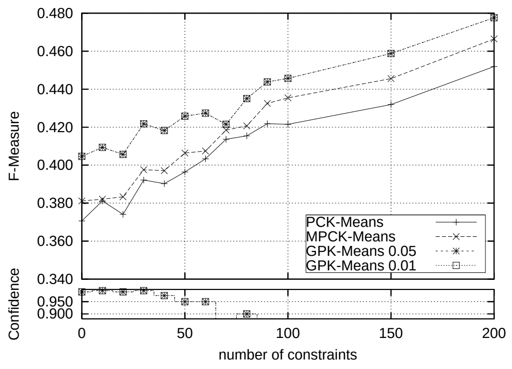
FIG. 4.4. The F-Measure of GPK-Means using reduced propagation. The bottom section of each graph depicts the significance level of GPK-Means against MPCK-Means.



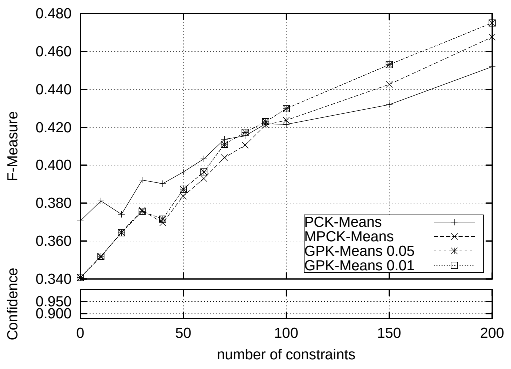
(g) LetterIJL - Single Metric



(h) LetterIJL - Multiple Metrics

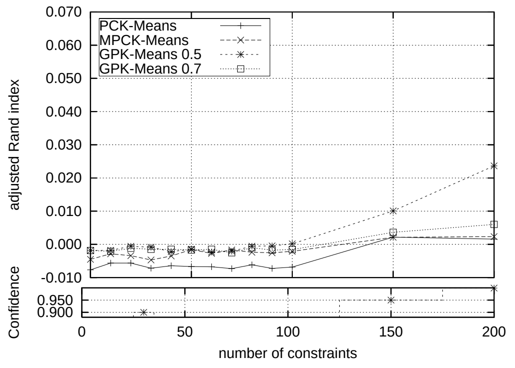


(i) Protein - Single Metric

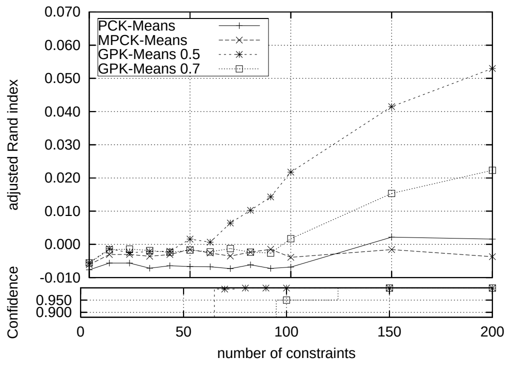


(j) Protein - Multiple Metrics

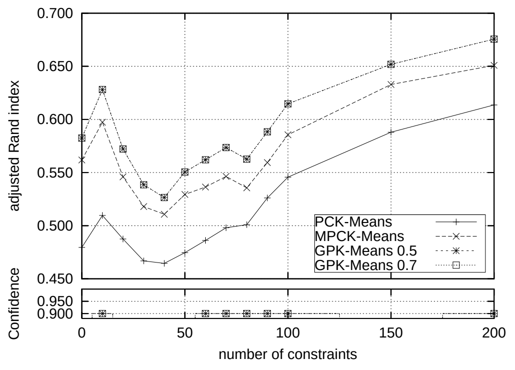
FIG. 4.4 (continued). The F-Measure of GPK-Means using reduced propagation. The bottom section of each graph depicts the significance level of GPK-Means against MPCK-Means.



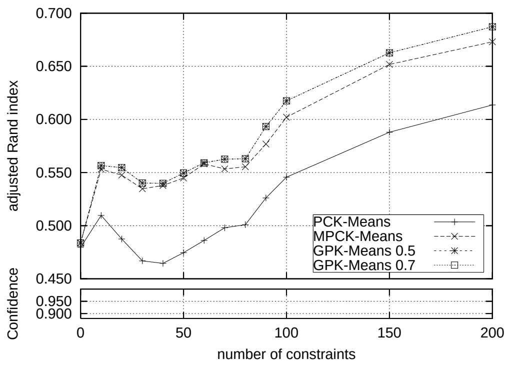
(a) Crabs-Gender - Single Metric



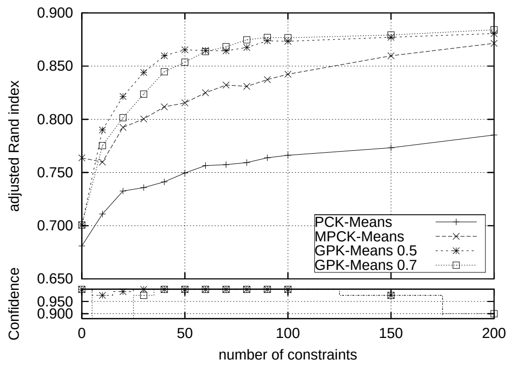
(b) Crabs-Gender - Multiple Metrics



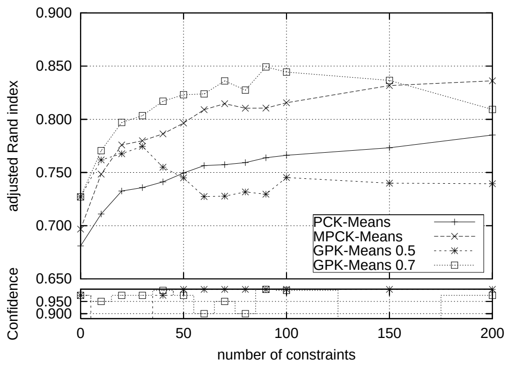
(c) Digits389 - Single Metric



(d) Digits389 - Multiple Metrics

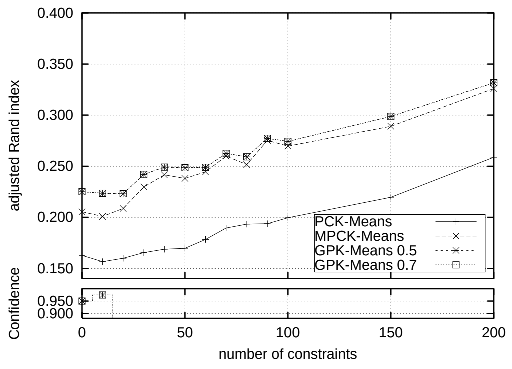


(e) Iris - Single Metric

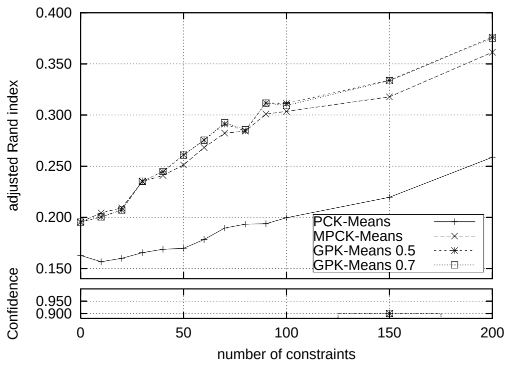


(f) Iris - Multiple Metrics

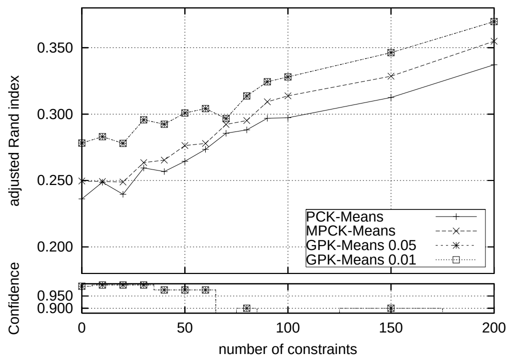
FIG. 4.5. The adjusted Rand index performance of GPK-Means using reduced propagation. The bottom section of each graph depicts the significance level of GPK-Means against MPCK-Means.



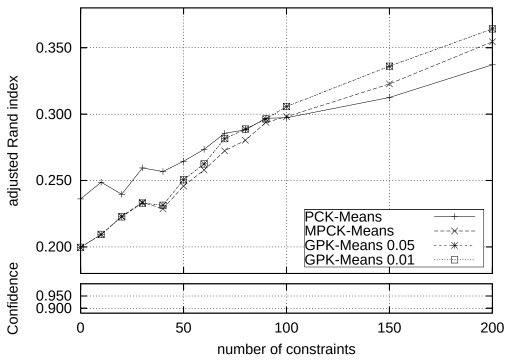
(g) LetterIJL - Single Metric



(h) LetterIJL - Multiple Metrics



(i) Protein - Single Metric



(j) Protein - Multiple Metrics

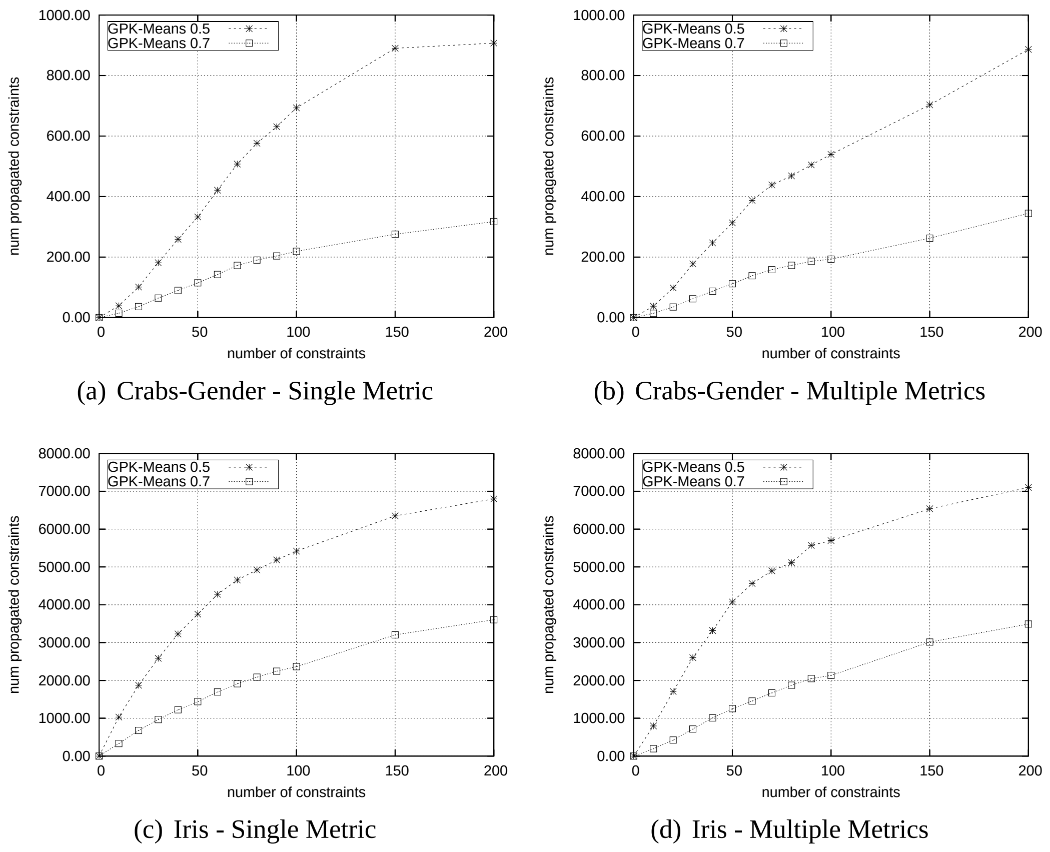
FIG. 4.5 (continued). The adjusted Rand index performance of GPK-Means using reduced propagation. The bottom section of each graph depicts the significance level of GPK-Means against MPCK-Means.

4.4.2 Evaluation of Reduced Constraint Propagation

Figure 4.6 depicts the number of propagated constraints under reduced propagation as the number of source constraints was varied. The quantities of propagated constraints reported in Figure 4.6 *exclude* the original constraints, and were measured after the last iteration of constraints propagation in GPK-Means.

As discussed in Section 4.4, the number of propagated constraints under reduced propagation is less than or equal to $N^2 - N$. The results in Figure 4.6 are consistent with this bound.

Figure 4.6 omits the plot of the number of propagated constraints for Digits389 because of an interesting effect with reduced propagation—there were zero propagated constraints for Digits389 after deducting the source constraints, using both single and multiple metrics. The Gaussian Propagation actually *reduced* the number of constraints by eliminating multiple constraints between the same pair of data points. This reduction seems to have improved the clustering quality on Digits389. Since the clustering performance with full propagation is approximately the same as with reduced propagation, it may be the case that all of the propagations occurred between a small set of already constrained points, causing the performance increase. This atypical case is worthy of further study, but this thesis leaves it to future work.



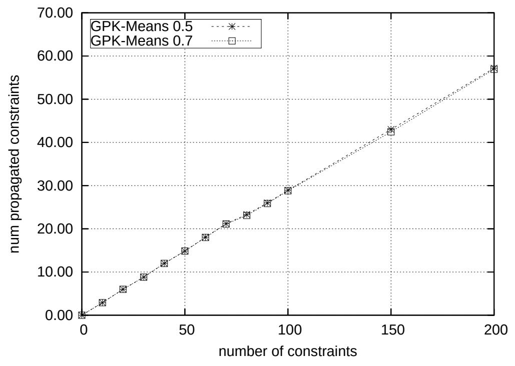
(a) Crabs-Gender - Single Metric

(b) Crabs-Gender - Multiple Metrics

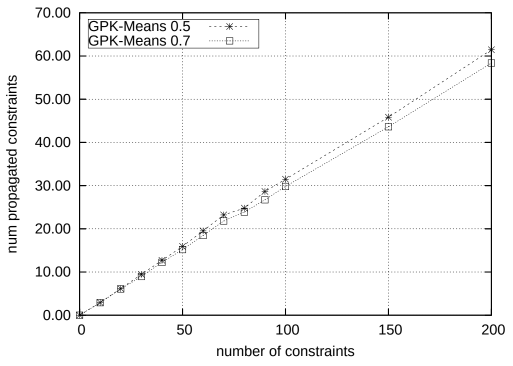
(c) Iris - Single Metric

(d) Iris - Multiple Metrics

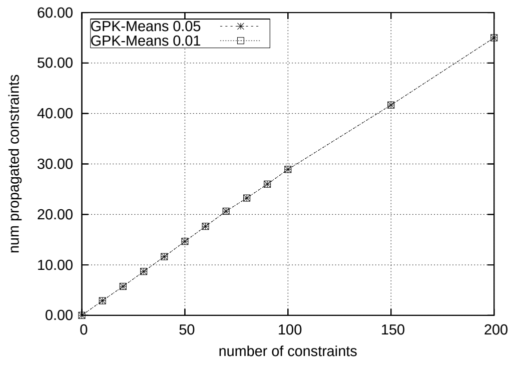
FIG. 4.6. The number of constraints inferred by GPK-Means using reduced propagation. Note that the counts *exclude* the original source constraints.



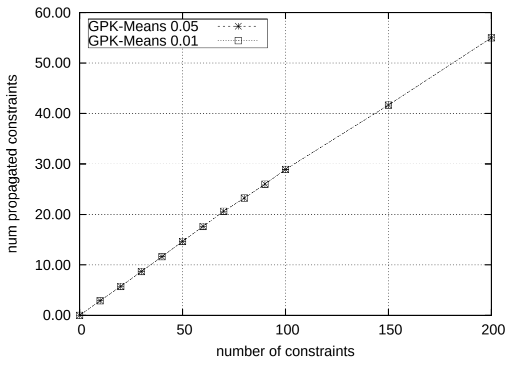
(g) LetterIJL - Single Metric



(h) LetterIJL - Multiple Metrics



(i) Protein - Single Metric



(j) Protein - Multiple Metrics

FIG. 4.6 (continued). The number of constraints inferred by GPK-Means using reduced propagation. Note that the counts *exclude* the original source constraints.

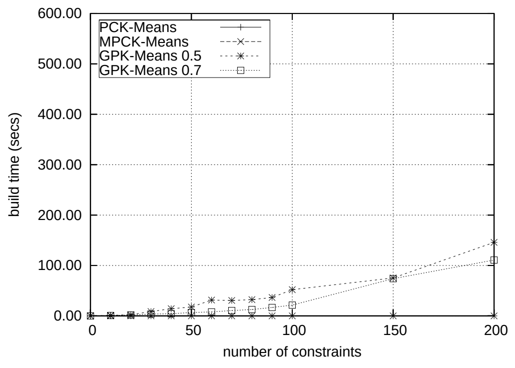
4.5 Clustering Time Evaluation

GPK-Means runs multiple iterations of MPCK-Means and constraint propagation. The MPCK-Means implementation by Bilenko et al. runs in under one second, so GPK-Means must take a minimum of several seconds. To determine whether GPK-Means could be used in a reasonable amount of time, this section examines the run-time of GPK-Means as the number of constraints was varied. The results were averaged over 20 trials of 5-fold cross-validation on the Iris data set. The experiment was conducted on a low-loaded 2.4 GHz Pentium 4 with hyper-threading running Linux. Figure 4.7 shows the empirical run-time of GPK-Means using full propagation and GPK-Means using reduced propagation.

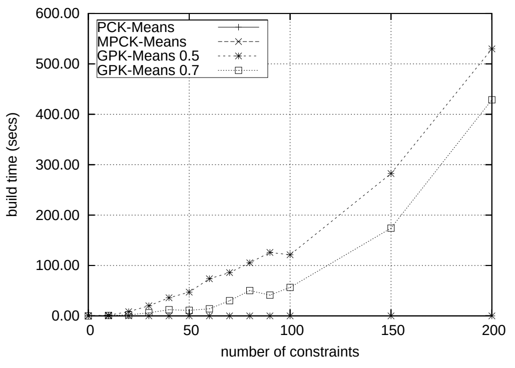
Full and reduced propagation have approximately the same empirical run-time, with only slight differences. Reduced propagation incurs a small overhead with keeping only the highest weighted constraints, but MPCK-Means runs faster under reduced propagation, because it is given fewer constraints than full propagation. These differences balance out, resulting in approximately the same run-time.

While the empirical run-time of GPK-Means appears much higher than MPCK-Means, the extra computational cost is inexpensive compared to that of obtaining more constraints. Consider the news article clustering scenario given in Chapter 1. Most people would gladly use GPK-Means and wait a few extra seconds (or minutes), rather than use MPCK-Means and have to read several more articles. With the cost of obtaining more constraints in mind, GPK-Means' run-time is reasonable for practical use.

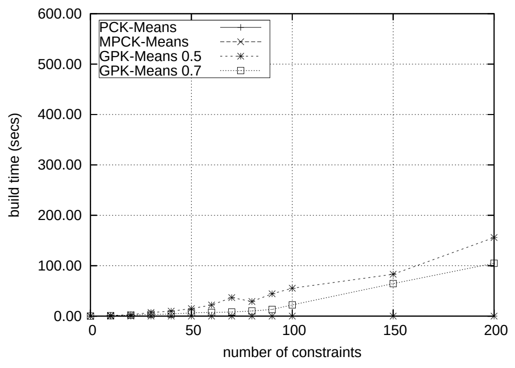
The discussion of future work (Section 5.5) includes ideas for improving the run-time of GPK-Means using sampling, and by eliminating unnecessary weight computations.



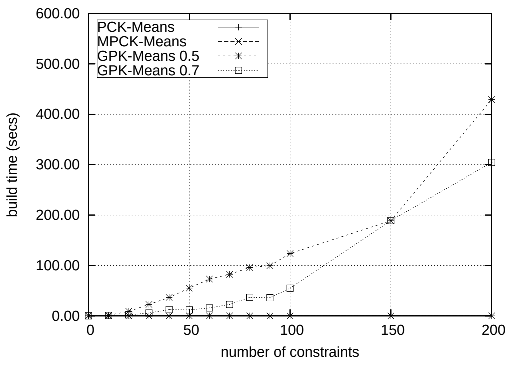
(a) Iris - Single Metric - Full Propagation



(b) Iris - Multiple Metrics - Full Propagation



(c) Iris - Single Metric - Reduced Propagation



(d) Iris - Multiple Metrics - Reduced Propagation

FIG. 4.7. Clustering time for GPK-Means using full and reduced propagation. All experiments were conducted on a low-loaded 2.4 GHz Pentium 4HT and averaged over 20 trials of 5-fold cross-validation.

Chapter 5

DISCUSSION

5.1 Full Versus Reduced Propagation

Full propagation and reduced propagation have approximately the same clustering performance and empirical run-time. Reduced propagation shows a slight performance improvement on the Iris and Protein data sets, and a performance decrease on Crabs-Gender. The largest benefit of it is a limit on the number of propagated constraints.

Full propagation allows multiple constraints between the same pair of points, which seems unreasonable at first. However, the multiple constraints between a pair of points could be interpreted as a single constraint between the pair with a weight equal to the sum of the multiple constraint weights. It is not possible to violate a subset of the multiple constraints, so they act and contribute as one.

The weight of the single (summed) constraint could be higher than the weight of any original source constraint. This is reasonable if the source constraints are uniformly distributed, but if they are concentrated in an area, constraints in that area will augment each other's weight. Reduced propagation limits the constraint weight, so that only the source constraints are given the highest weight in a neighborhood.

Since the benefit of reduced propagation varies between data sets, further study is necessary to determine whether reduced propagation should be used over full propagation.

5.2 Benefits of Constraint Propagation

Constraint propagation is most useful when the number of source constraints is small, as shown in the results for the Iris and Protein data sets. This thesis focuses on providing a high-quality clustering with small numbers of constraints, and constraint propagation is capable of doing that.

With larger numbers of source constraints, it is better to use the accurate source constraints than the (possibly inaccurate) propagated constraints. The accurate source constraints will yield a higher-quality clustering in most cases. Also, the cost of constraint propagation increases with the number of source constraints, making constraint propagation with larger numbers of source constraints prohibitively expensive.

Constraint propagation assumes that the constraints are representative of their local neighborhood. In situations where constraints are used to constrain outliers or atypical instances, this assumption is incorrect and constraint propagation may decrease clustering performance.

5.3 Using GPK-Means with Other Algorithms

GPK-Means is capable of using any base constrained clustering algorithm that supports weighted constraints. It is possible to use it with an algorithm that supports only unweighted constraints by inferring constraints if they are above the given threshold, and ignoring the actual weight of the constraint. Such a modification to the GPK-Means algorithm and an analysis of its performance is left to future work.

Most existing constrained clustering algorithms tend to focus on using “perfect” constraints, without errors in the relative labeling. MPCK-Means falls into this category. The constraints inferred by GPK-Means’ propagation method are likely to be inaccurate. While MPCK-Means assumes only accurate constraints, GPK-Means uses it successfully with possibly inaccurate propagated constraints.

Very little work on constrained clustering has focused on using imperfect constraints; this is left as an open question in many papers' future work sections. GPK-Means may work better with constrained clustering methods designed for imperfect constraints; they could easily replace MPCK-Means in GPK-Means.

5.4 Comparison of GPK-Means with Other Methods

This section compares GPK-Means to several other constrained clustering methods from a theoretical standpoint.

5.4.1 Bilenko et al.'s MPCK-Means

The primary difference between GPK-Means and MPCK-Means (Bilenko, Basu, & Mooney 2004) is in how each uses constraints. MPCK-Means assumes that constraint violations occur based on the clustering of individual points. Points are assigned individually to clusters in order to minimize the objective function, which is partially based on constraint violations.

In contrast, GPK-Means makes an explicit assumption that constraints are representative of their local neighborhood. The amount each neighborhood is constrained is directly based on its location in the current clustering. The effect of a constraint extends out into space, constraining two neighborhoods rather than two individual instances.

Note that GPK-Means with a propagation threshold of $g = 1$ corresponds to two runs of MPCK-Means, with the second run seeded by the centroids discovered by the first run. This correspondence holds under the assumption that for all data instances that are equal in feature space, either it is the case that none of those instances are involved in a constraint, or it is the case that all of those instances are involved in a constraint with some other instance

x_j. To illustrate the need for this assumption, consider a clustering task where x_1 and x_2 have the same location in feature space, and x_1 is involved in a constraint with x_3, but x_2 is not, violating the assumption. GPK-Means with $g = 1$ would add a constraint between x_2 and x_3, so it would no longer correspond directly to two runs of MPCK-Means.

5.4.2 Klein et al.'s Constrained Complete-Link Clustering

Klein et al.'s (2002) method for constrained complete-link clustering warps a similarity matrix of the data in response to constraints, pulling must-linked points closer, and pushing cannot-linked constraints away. Unlike other methods that learn a distance metric, Klein et al.'s method is based on distances between individual instances.

Similar to Klein et al.'s approach, GPK-Means consider individual pairs of instances during constraint propagation. These individual pairs are pulled together or pushed apart in response to the source constraint. By propagating the effect of a source constraint to nearby pairs, the source constraint affects a neighborhood, similar to Klein et al.'s approach. The effect on the neighborhood increases as the constraint moves closer to the center of the cluster.

However, once the effect is propagated to the local neighborhood, GPK-Means still relies on the underlying constrained clustering algorithm to generate the clustering. In the experimental setup, MPCK-Means learns a distance metric, so the effect of the propagated constraint is limited to adjusting the centroids and contributing to the distance metric. This lessens the effect of the constraints on the neighborhood, because the effects on all neighborhoods are combined together into a distance metric. Klein et al.'s method keeps the effects of the constraints separate, and allows them to interact only via the distances between the points.

Said another way, Klein et al.'s method can be thought of pinching a rubber sheet, pulling must-linked points to touch and cannot-linked points as far apart as possible. Points in between the constrained points adjust their distances based on the warping of the space. The distance between any two points is the shortest path through this space, which is warped based on the *combination* of individual constraints. Constraint propagation takes the rubber sheet and places weights in a greater-dimensional version of it to sink the neighborhoods of each must-link constraint together, and pull the neighborhoods of cannot-link constraints apart. The effect of these weighted neighborhoods is then combined together to form a distance metric that governs the distance between any two points based on the *average* of many constraints.

5.4.3 Bar-Hillel et al.'s Relevant Component Analysis

Bar-Hillel et al.'s (2005) application of RCA to constrained clustering learns a Mahalanobis distance metric based on the clusters defined by the equivalence sets. It estimates this distance metric as the covariance matrix for these equivalence sets. Essentially, it assumes that the equivalence sets are representative of the actual clusters, and estimates a metric based on the clusters defined by the equivalence sets. RCA does not do any form of iterative refinement; it generates the metric from only the labeled data. Clustering can then be performed using the unlabeled data transformed by the metric.

GPK-Means, in contrast, uses the best estimate of the clustering (from both the constraints *and* the unlabeled data) to determine how to propagate the constraints to the nearby neighborhoods. It interprets constraints as affecting a neighborhood, rather than acting as a sample for a cluster. Those neighborhoods are based on the current clustering, rather than just the estimates of the clustering from the labeled data.

GPK-Means could wrap around RCA, using RCA to learn the metric defined by the constraints and then K-Means to generate the clustering. This would require a relatively straightforward modification of the RCA algorithm to use weighted points in estimating the distance metric.

5.5 Future Work

The results have revealed several aspects of GPK-Means which could use improvement. The Gaussian constraint propagation does not work very well on high-dimensional data sets. It propagates relatively few constraints, most likely due to the decreased density of the high-dimensional data. Future work includes exploring the use of dimensionality reduction in the propagation step to avoid this problem and improve the clustering quality of GPK-Means on high-dimensional data sets.

The method used to scale the metrics to the cluster could also be at fault for the poor performance on high-dimensional data sets. Using a different method for scaling the metrics to the cluster could improve the performance of GPK-Means, and possibly eliminate the need for a propagation threshold.

GPK-Means could use RCA with dimensionality reduction to learn the metric, instead of MPCK-Means. By using RCA, GPK-Means would gain the benefits of built-in dimensionality reduction, and possible elimination of the metric scale problem.

This thesis investigated propagating constraints with a Gaussian function, with encouraging results. Another area of future work involves exploring other methods of propagating constraints, such as by Euclidean distance.

Constraint propagation by considering each pair of data points for each constraint is computationally expensive; sampling could potentially reduce the cost of constraint propagation while yielding the same performance. Constraint propagation currently considers many pairs of data points which are far apart, and therefore makes a large number of unnecessary computations. Tracking point distances, as in the method used by Elkan (2003) to accelerate K-Means, will eliminate many unnecessary computations and will accelerate constraint propagation.

Chapter 6

CONCLUSION

This thesis investigated propagating constraints to nearby points using a Gaussian function, with encouraging results. Clustering with the propagated constraints yielded higher-quality clusterings than clustering with only the original constraints on several data sets. Constraint propagation appears especially useful when there are few source constraints.

Constraint propagation does not appear to do well in high-dimensional data sets, possibly due to the low density of the propagation neighborhoods. Dimensionality reduction may improve the clustering quality. Constraint propagation when given high numbers of constraints does not perform as well as standard constrained clustering, since the given constraints are more accurate than the propagated constraints. However, in most cases, constraint propagation does not significantly degrade the clustering quality.

The results in this thesis support further exploration of constraint propagation and the development of methods for clustering with propagated constraints. Constraint propagation may be expensive compared to standard constrained clustering algorithms, but the cost is low compared to that of obtaining many constraints. Clustering with propagated constraints is capable of providing a high-quality clustering when given few source constraints; other constrained clustering algorithms perform poorly on this task.

Appendix A

CONSTRUCTION OF THE COVARIANCE MATRIX FOR CONSTRAINT PROPAGATION

This Appendix shows how to construct the covariance matrix given as Equation 3.3 with the assumption that the two endpoints of a constraint are independent.

$$\Sigma_{x_A x_B} = \begin{bmatrix} \Sigma_{x_A} & [0] \\ [0] & \Sigma_{x_B} \end{bmatrix} \tag{3.3}$$

Recall that $\Sigma_{x_A x_B}$ is a $2n \times 2n$ matrix. Let $v = \begin{bmatrix} x_i \\ x_j \end{bmatrix}$, and $u = \begin{bmatrix} x_A \\ x_B \end{bmatrix}$. The (g, h)th element σ_{gh} of the covariance matrix $\Sigma_{x_A x_B}$ is the covariance of the gth and hth elements of v. That is, $\sigma_{gh} = \sigma_{hg} = E[(v_g - u_g)(v_h - u_h)]$, for $g, h = 1 \ldots 2n$. For $g = h$, $\sigma_{gh} = \sigma_g^2$, which is the variance of v_g. Expanding $\Sigma_{x_A x_B}$ from this definition yields Equation A.1.

As shown by the solid black lines in Equation A.1, $\Sigma_{x_A x_B}$ can be divided into four quadrants. In the construction of the Gaussian weighting function, this thesis assumes that x_i is independent of x_j, so their covariance is 0. Therefore, in the construction of the covariance matrix, $\sigma_{gh} = 0$ and $\sigma_{hg} = 0$, for $g = 1 \ldots n$ and $h = (n+1) \ldots 2n$. All elements of the lower left quadrant and the upper right right quadrant go to zero, therefore, these quadrants can be replaced by the $n \times n$ zero matrix. The upper left quadrant corresponds to the covariance matrix Σ_{x_A}, and the lower right quadrant corresponds to Σ_{x_B}.

(A.1)

$$\Sigma_{x_A x_B} = \left[\begin{array}{cccc|cccc}
\sigma_1^2 & \sigma_{12} & \dots & \sigma_{1n} & \sigma_{1(n+1)} & \sigma_{1(n+2)} & \dots & \sigma_{1(2n)} \\
\sigma_{21} & \sigma_2^2 & \dots & \sigma_{2n} & \sigma_{2(n+1)} & \sigma_{2(n+2)} & \dots & \sigma_{2(2n)} \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
\sigma_{n1} & \sigma_{n2} & \dots & \sigma_n^2 & \sigma_{n(n+1)} & \sigma_{n(n+2)} & \dots & \sigma_{n(2n)} \\
\hline
\sigma_{(n+1)1} & \sigma_{(n+1)2} & \dots & \sigma_{(n+1)n} & \sigma_{(n+1)}^2 & \sigma_{(n+1)(n+2)} & \dots & \sigma_{(n+1)(2n)} \\
\sigma_{(n+2)1} & \sigma_{(n+2)2} & \dots & \sigma_{(n+2)n} & \sigma_{(n+2)(n+1)} & \sigma_{(n+2)}^2 & \dots & \sigma_{(n+2)(2n)} \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
\sigma_{(2n)1} & \sigma_{(2n)2} & \dots & \sigma_{(2n)n} & \sigma_{(2n)(n+1)} & \sigma_{(2n)(n+2)} & \dots & \sigma_{(2n)}^2
\end{array}\right]$$

$$\text{(A.2)} \quad = \left[\begin{array}{cccc|cccc}
\sigma_1^2 & \sigma_{12} & \dots & \sigma_{1n} & 0 & 0 & \dots & 0 \\
\sigma_{21} & \sigma_2^2 & \dots & \sigma_{2n} & 0 & 0 & \dots & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
\sigma_{n1} & \sigma_{n2} & \dots & \sigma_n^2 & 0 & 0 & \dots & 0 \\
\hline
0 & 0 & \dots & 0 & \sigma_{(n+1)}^2 & \sigma_{(n+1)(n+2)} & \dots & \sigma_{(n+1)(2n)} \\
0 & 0 & \dots & 0 & \sigma_{(n+2)(n+1)} & \sigma_{(n+2)}^2 & \dots & \sigma_{(n+2)(2n)} \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & \dots & 0 & \sigma_{(2n)(n+1)} & \sigma_{(2n)(n+2)} & \dots & \sigma_{(2n)}^2
\end{array}\right],$$

$$\text{(3.3)} \quad = \begin{bmatrix} \Sigma_{x_A} & [0] \\ [0] & \Sigma_{x_B} \end{bmatrix}.$$

Appendix B

EFFICIENT COMPUTATION OF PROPAGATED CONSTRAINT WEIGHTS

This Appendix demonstrates how to efficiently compute $Weight(x_i, x_j, x_A, x_B)$ (Equation 3.7) by taking advantage of the independence assumptions in the construction of $\Sigma_{x_A x_B}$ (Equation 3.3). For convenience, here are the relevant equations:

(3.1) $$N(v, u, \sigma) = exp\left[-\frac{1}{2}(v-u)^T \sigma^{-1}(v-u)\right] \ ,$$

(3.2) $$W(x_i, x_j, x_A, x_B, \Sigma_{x_A}, \Sigma_{x_B}) = N\left(\begin{bmatrix} x_i \\ x_j \end{bmatrix}, \begin{bmatrix} x_A \\ x_B \end{bmatrix}, \Sigma_{x_A x_B}\right) \ ,$$

(3.3) $$\Sigma_{x_A x_B} = \begin{bmatrix} \Sigma_{x_A} & [0] \\ [0] & \Sigma_{x_B} \end{bmatrix} \ ,$$

(3.4) $$n_h = \frac{radius_h}{3\sigma_{pc1_h}} \ ,$$

(3.6) $$s_{x_A} = N(x_A, \mu_{x_A}, n_{x_A} A_{x_A}^{-1}) \ ,$$

(3.7) $$Weight(x_i, x_j, x_A, x_B) = W(x_i, x_j, x_A, x_B, s_{x_A} n_{x_A} A_{x_A}^{-1}, s_{x_B} n_{x_B} A_{x_B}^{-1}) \ .$$

Recall that each data instance has n dimensions, and that $\Sigma_{x_A x_B}$ is a $2n \times 2n$ matrix.

Let $v = \begin{bmatrix} x_i \\ x_j \end{bmatrix}$, and $u = \begin{bmatrix} x_A \\ x_B \end{bmatrix}$. Since the experiments are restricted to diagonal metric matrices, $\Sigma_{x_A x_B}$ is diagonal.

By definition, for a diagonal $2n \times 2n$ (since $\Sigma_{x_A x_B}$ is $2n \times 2n$) covariance matrix σ ,

$$N(v, u, \sigma) = exp\left[-\frac{1}{2}(v-u)^T \sigma^{-1}(v-u)\right] \quad , \tag{B.1}$$

$$= exp\left[-\frac{1}{2}\sum_{k=1}^{2n}\left(\frac{v_k - u_k}{\sigma_k}\right)^2\right] \quad . \tag{B.2}$$

Let $\sigma = \Sigma_{x_A x_B}$. Expanding $\sum_{k=1}^{2n}(\frac{v_k-u_k}{\sigma_k})^2$ yields:

$$\sum_{k=1}^{2n}\left(\frac{v_k - u_k}{\sigma_k}\right)^2 = \sum_{k=1}^{n}\left(\frac{v_k - u_k}{\sigma_k}\right)^2 + \sum_{k=(n+1)}^{2n}\left(\frac{v_k - u_k}{\sigma_k}\right)^2 \quad , \tag{B.3}$$

$$= \sum_{k=1}^{n}\left(\frac{x_{ik} - x_{Ak}}{\Sigma_{x_A k}}\right)^2 + \sum_{k=1}^{n}\left(\frac{x_{jk} - x_{Bk}}{\Sigma_{x_B k}}\right)^2 \quad , \tag{B.4}$$

$$= (x_i - x_A)^T \Sigma_{x_A}^{-1}(x_i - x_A) + (x_j - x_B)^T \Sigma_{x_B}^{-1}(x_j - x_B) \quad . \tag{B.5}$$

By substituting back into the Gaussian equation,

$$N\left(\begin{bmatrix} x_i \\ x_j \end{bmatrix}, \begin{bmatrix} x_A \\ x_B \end{bmatrix}, \begin{bmatrix} \Sigma_{x_A} & [0] \\ [0] & \Sigma_{x_B} \end{bmatrix}\right)$$
$$= exp\left[-\frac{1}{2}(x_i - x_A)^T \Sigma_{x_A}^{-1}(x_i - x_A) - \frac{1}{2}(x_j - x_B)^T \Sigma_{x_B}^{-1}(x_j - x_B)\right] \quad . \tag{B.6}$$

The complexity of this computation over repeated evaluations can be reduced by memoizing the computations of $\Sigma_{x_A} = s_{x_A} n_{x_A} A_{x_A}$, and $\Sigma_{x_A}^{-1}$. The computation $n_h A_h$ can be calculated once for each cluster, and cached for future use. The computation $-\frac{1}{2}(x_i - x_A)^T \Sigma_{x_A}^{-1}(x_i - x_A)$ can also be memoized for efficient lookup as x_j varies. From experience, this memoization has significantly accelerated constraint propagation.

REFERENCES

[1] Bar-Hillel, A.; Hertz, T.; Shental, N.; and Weinshall, D. 2005. Learning a Mahalanobis metric from equivalence constraints. *Journal of Machine Learning Research* 6:937–965.

[2] Basu, S.; Banerjee, A.; and Mooney, R. 2002. Semi-supervised clustering by seeding. In *Proceedings of the Nineteenth International Conference on Machine Learning*, 19–26. Sydney, Australia: Morgan Kauffman.

[3] Basu, S.; Banerjee, A.; and Mooney, R. J. 2004. Active semi-supervision for pairwise constrained clustering. In *Proceedings of the SIAM International Conference on Data Mining*, 333–344. Lake Buena Vista, FL: SIAM.

[4] Basu, S.; Bilenko, M.; and Mooney, R. J. 2004. A probabilistic framework for semi-supervised clustering. In *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining*, 59–68. Seattle, WA: ACM.

[5] Basu, S. 2005. *Semi-Supervised Clustering: Probabilistic Models, Algorithms, and Experiments*. Ph.D. Dissertation, University of Texas at Austin.

[6] Bilenko, M.; Basu, S.; and Mooney, R. J. 2004. Integrating constraints and metric learning in semi-supervised clustering. In *Proceedings of the Twenty-First International Conference on Machine Learning*, 81–88. Banff, Alberta, Canada: ACM.

[7] Cohn, D.; Caruana, R.; and McCallum, A. 2003. Semi-supervised clustering with user feedback. Technical Report TR2003-1892, Cornell University.

[8] Duda, R. O.; Hart, P. E.; and Stork, D. G. 2001. *Pattern Classification*. Wiley-Interscience, 2nd edition.

[9] Elkan, C. 2003. Using the triangle inequality to accelerate k-means. In *Proceedings of the Twentieth International Conference on Machine Learning*, 147–153. AAAI Press.

[10] Hastie, T.; Tibshirani, R.; and Friedman, J. 2001. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer.

[11] Hubert, L., and Arabie, P. 1985. Comparing partitions. *Journal of Classification* 2:193–218.

[12] Kearns, M.; Mansour, Y.; and Ng, A. Y. 1997. An information-theoretic analysis of hard and soft assignment methods for clustering. In *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence*, 282–293. Morgan Kaufmann.

[13] Klein, D.; Kamvar, S. D.; and Manning, C. D. 2002. From instance-level constraints to space-level constraints: Making the most of prior knowledge in data clustering. In *Proceedings of the Nineteenth International Conference on Machine Learning*, 307–314. Sydney, Australia: Morgan Kauffman.

[14] Lange, T.; Law, M. H. C.; Jain, A. K.; and Buhmann, J. M. 2005. Learning with constrained and unlabeled data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 731–738. IEEE Press.

[15] Law, M. H. C.; Topchy, A.; and Jain, A. K. 2004. Model-based clustering with soft and probabilistic constraints. Technical Report MSU-CSE-04-51, Michigan State University.

[16] Lowe, D. G. 2004. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision* 60(2):91–110.

[17] Lu, Z., and Leen, T. 2005. Semi-supervised learning with penalized probabilistic

clustering. In Saul, L. K.; Weiss, Y.; and Bottou, L., eds., *Advances in Neural Information Processing Systems 17*, 849–856. Cambridge, MA: MIT Press.

[18] MacQueen, J. B. 1967. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, 281–297. University of California Press.

[19] Newman, D.; Hettich, S.; Blake, C.; and Merz, C. 1998. UCI repository of machine learning databases. Available on-line: http://www.ics.uci.edu/∼mlearn/MLRepository.html.

[20] Rand, W. M. 1971. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association* 66(336):846–850.

[21] Shental, N.; Hertz, T.; Weinshall, D.; and Pavel, M. 2002. Adjustment learning and relevant component analysis. In *Proceedings of the Seventh European Conference on Computer Vision*, 776–792. Springer-Verlag.

[22] Shental, N.; Bar-Hillel, A.; Hertz, T.; and Weinshall, D. 2004. Computing Gaussian mixture models with EM using equivalence constraints. In Thrun, S.; Saul, L.; and Schölkopf, B., eds., *Advances in Neural Information Processing Systems 16*. Cambridge, MA: MIT Press.

[23] Wagstaff, K.; Cardie, C.; Rogers, S.; and Schroedl, S. 2001. Constrained k-means clustering with background knowledge. In *Proceedings of the Eighteenth International Conference on Machine Learning*, 577–584. Williamstown, MA: Morgan Kaufmann.

[24] Wagstaff, K. L. 2002. *Intelligent Clustering with Instance-Level Constraints*. Ph.D. Dissertation, Cornell University.

[25] Witten, I. H., and Frank, E. 2000. *Data Mining: Practical Machine Learning Tools with Java Implementations*. San Francisco, CA: Morgan Kaufmann.

[26] Xing, E. P.; Ng, A. Y.; Jordan, M. I.; and Russell, S. 2003. Distance metric learning, with application to clustering with side-information. *Advances in Neural Information Processing Systems* 15:505–512.

[27] Yeung, K. Y., and Ruzzo, W. L. 2001. An empirical study of principal component analysis for clustering gene expression data. *Bioinformatics* 17(9):763–774.

[28] Zhu, X.; Ghahramani, Z.; and Lafferty, J. 2003. Semi-supervised learning using Gaussian fields and harmonic functions. In Fawcett, T., and Mishra, N., eds., *Proceedings of the Twentieth International Conference on Machine Learning*, 912–919. Washington, DC: AAAI Press.

[29] Zhu, X.; Lafferty, J.; and Ghahramani, Z. 2003. Semi-supervised learning: From Gaussian fields to Gaussian processes. Technical Report CMU-CS-03-175, Carnegie Mellon University.