Bregman Bubble Clustering: A Robust Framework for Mining Dense Clusters

Gunjan Gupta Joydeep Ghosh

{ggupta/ghosh}@ece.utexas.edu

IDEAL-2006-TR04*

Intelligent Data Exploration & Analysis Laboratory (IDEAL)

(Web: http://www.ideal.ece.utexas.edu/)

Department of Electrical and Computer Engineering
The University of Texas at Austin
Austin, Texas 78712
U.S.A.

September 4, 2006

Abstract

In classical clustering, each data point is assigned to at least one cluster. However, in many applications only a small subset of the available data is relevant for the problem and the rest needs to be ignored in order to obtain good clusters. Certain non-parametric density-based clustering methods find the most relevant data as multiple dense regions, but such methods are generally limited to low-dimensional spatial data or are too slow for larger datasets. On the other hand, the promising One Class Information Bottleneck (OC-IB) method uses a fast iterative relocation scheme, and works on a large class of distortion measures known as Bregman divergences, but can only find a *single* dense region. In this paper, we first present a series of improvements and extensions over OC-IB to find k dense clusters that includes a seeding algorithm that can automatically estimate the best k. When k is forced to 1, our method gives rise to an improved version of OC-IB that yields optimality guarantees. Furthermore, we construct a generative model that gives rise to the proposed iterative algorithm for finding k dense regions as a special case. Our analysis reveals an interesting and novel connection between the problem of finding dense regions and exponential mixture models. The proposed methods provide a scalable approach for finding multiple dense regions as measured using any Bregman divergence, thus extending density based clustering to a variety of problems not addressable by earlier methods. We present empirical results on various artificial and real-life datasets to show the relevance and effectiveness of our methods.

1 Introduction

Clustering data is an important unsupervised learning problem that has been extensively applied in various domains [JD88]. In classical clustering, each data point is assigned to at least one cluster. However, in many real-world problems, only a small subset, often containing multiple natural groupings, clusters well, while the rest of the data shows little or no clustering tendencies. In a wide variety of domains where a large amount of (often noisy) real-life data is available, situations frequently arise where it is more important to cluster a small subset of the data very well, rather than optimizing clustering over all the data points.

For example, in Market-basket data, typically only a small subset of customers show coherent buying behavior [SG03, GG01]. These are often the loyal customers who purchase regularly. Another area is text mining, where recovering the most relevant items of key categories is more important than obtaining an exhaustive list. Biometric identification systems is another domain where there are many positive examples from multiple known users, but the real goal is to distinguish them from a variety of uncharacterized negative examples. Many types of bioinformatics datasets also exhibit this property. For example, gene-expression datasets measure expression level of genes compared to a control across a few thousand genes. The experiments typically cover only a specific "theme" such as stress-response, and therefore only a few genes related to the conditions show good clustering. Biologists are interested in recovering small, multiple clusters formed from a small subset of genes that show strongly correlated expression patterns. Often such clusters map to biological processes that are involved in the specific context. For example, on the Gasch dataset [Gas00], which consists of only stress response experiments and is a popular benchmark for testing gene clustering, over 5,500 genes out of 6,151 genes are not directly involved in stress response. These genes show insignificant change in expression level with respect to the control sample, and should be pruned for good clustering. Other types of biological data that share similar properties include protein mass spectroscopy and phylogenetic profile data.

One way to handle such scenarios would be to prune the large fraction of "don't care" data as a pre-processing or a post-processing step, and use existing "exhaustive" clustering methods. However, optimal preprocessing requires the knowledge of what subset would cluster well, which can only be defined well in the context of the clustering step. Post-processing is also not ideal since the optimization in exhaustive clustering is over the full dataset, and not on the small subset that is relevant. A more natural approach would involve finding the multiple dense regions and the "don't care" set simultaneously.

Density based clustering methods such as DBSCAN [EKSX96] find relevant subsets as multiple dense regions, while the rest of the data get assigned to a "don't care" set. These methods focus on directly identifying the densest possible regions, and unlike exhaustive clustering, work well in situations where only a small, densest fraction of the data is relevant. However, DBSCAN implicitly exploits triangle in-equality to find a set of densely connected points. This may not be the appropriate for many clustering problems where the suitable divergence criteria between points is not a metric. Furthermore, DBSCAN (and its derivatives) requires an efficient database index to be scalable to large data sets, which is usually available for only low-dimensional metric spaces [KPPS03]. Hence, applications of these algorithms have been largely limited to spatial datasets.

One Class Information Bottleneck (OC-IB) [CC04] provides a novel local search based approach for finding a single dense region in the data that is extremely fast, scalable to very

high-dimensional datasets, and works with a large family of distortion measures known as *Bregman divergences* (Section 2.1). However OC-IB can only find a single dense region, whereas in many problems dense regions can form multiple natural clusters. Furthermore, OC-IB can get stuck into a bad local minima and does not allow control over the size of the cluster returned, which can vary greatly depending upon the quality of the local minima.

Our approach, which is motivated by OC-IB, is able to resolve all these issues and greatly extends the current scope of density-based clustering. Our method is scalable to large, high dimensional datasets, works with all Bregman divergences, can find multiple dense regions simultaneously, and gives high quality clusters that are robust to initialization. We also propose a generative model for our method that provides deeper insight into our framework, and show several interesting connections between the problem of finding multiple dense regions, and (1) mixture modeling, (2) a recent unification of iterative relocation based methods known as Bregman Clustering [BMDG05], and (3) the problem of One Class Classification or Clustering as formulated by OC-IB. These contributions are summarized as follows:

1.1 Contributions

1. For the problem of finding *one* dense region, we present an alternative local search method, BBOCC-Q that optimizes the same cost function as OC-IB but gives better results.

2. We present a dual formulation, BBOCC-S that takes cluster size as input, provides control over the cluster size, and shows empirical improvements over OC-IB that are even more significant than that of BBOCC-Q.

3. We extend BBOCC to Pearson distance, a biologically relevant distance measure. The algorithmic difference between BBOCC and OC-IB is what makes BBOCC with Pearson distance computationally efficient; a corresponding extension using OC-IB runs much slower.

4. We present a generalization of BBOCC called Bregman Bubble Clustering (BBC) that can simultaneously find k dense clusters. We present two versions of BBC: BBC-Q and BBC-S corresponding to BBOCC-Q and BBOCC-S respectively. For both versions, BBC reduces to the corresponding version of BBOCC for $k = 1$. With a time-complexity and space complexity of $O(nd)$ for each step (for a dataset with n data points in d dimensions), BBC is scalable to much larger and higher-dimensional datasets than existing density-based methods. BBC also extends density-based clustering to *all* Bregman divergences, which includes a wide variety of popular measures such as Squared Euclidean distance, K-L Divergence, Mahalanobis distance, Itakura-Saito distance, Hinge Loss and Logistic Loss.

5. For medium-sized problems, we develop deterministic seeding algorithms called Hypersphere One Class Clustering (HOCC) and Density Gradient Enumeration (DGRADE) for initializing the local search in BBOCC and BBC respectively. HOCC and DGRADE give extremely good empirical results at the cost of somewhat increased time complexity [1]. HOCC also provides strong optimality guarantee on the quality of the results. DGRADE

[1] and also somewhat increased space complexity for DGRADE.

uses a novel approach to estimate the "density gradient" at all data points, and is able to automatically identify all the distinct dense regions in the data, thus allowing an automatic estimation of the best k. For many problems such as clustering gene-expression datasets, the more expensive time complexity of the seeding method is acceptable as it results in substantial improvement in the quality of results. Furthermore, the deterministic nature of the seeded results is also a desirable property when using clustering for discovering biochemical pathways, since the pathways within a biological organism that we are attempting to discover are well-defined [2], albeit often unknown (hence the need for clustering).

6. As a faster alternative to seeding, we present an extension to BBC called *Pressurization* that substantially improves the quality of the local search and overcomes the problem of local minima, while keeping the time and space complexity at $O(nd)$. This is especially important for very large problems (e.g. clustering millions of customers in a market-basket dataset) where the deterministic seeding approach is too slow to apply against the full dataset. In empirical evaluations, Pressurization gives results that are very robust to initialization and have very small variations in quality over multiple trials.

7. We develop a generative model consisting of a mixture of k exponentials and a uniform "background" distribution that leads to several insights into the problem of finding dense clusters using Bregman divergences. We show that BBOCC-S falls out as a "hard" version of a mixture of one uniform distribution and an exponential component, and that BBC-S is a "hard" version of the mixture of k exponentials and one uniform distribution [3]. Another subtle relationship that falls out from our framework is that the unified framework of hard and soft (exhaustive) clustering methods presented in Bregman Clustering [BMDG05], a generalization of K-Means and mixture modeling, can be viewed as a degenerate case of our method that is obtained when the size (or the weight for the soft model) of the don't care set is zero.

8. Empirical studies are presented on a variety of datasets showing the effectiveness of our framework on low, medium and very high-dimensional problems, as compared to Bregman Clustering, Single Link Agglomerative and DBSCAN.

A brief word on notation: bold faced variables, e.g. $\mathbf{x}$, represent vectors whose i^{th} element are accessed as either x_i or $x(i)$. Sets are represented by calligraphic upper-case alphabets such as $\mathcal{X}$ and are enumerated as $\{\mathbf{x}_i\}_{i=1}^{n}$ where $\mathbf{x}_i$ are the individual elements. $|\mathcal{X}|$ represents the size of set $\mathcal{X}$. Capital letters such as X are random variables. $\mathbb{R}$ and $\mathbb{R}^d$ represent the domain of real numbers and a d-dimensional vector space respectively. Bold-faced capital letters such as $\mathbf{M}_D$ represent a two-dimensional matrix.

[2] The relationships between genes only change/evolve (with a few rare exceptions) over evolutionary time-scales.

[3] This relationship was mentioned in passing by [CC04] for the Square Euclidean case for OC-IB, but we present formal derivations for Bregman divergences and generalization to the k class case.

2 Background

We now describe some key concepts and related work that will be important in describing our methods.

2.1 Partitional clustering using Bregman Divergences

Bregman Divergences: *Bregman divergences* form a family of distance measures, defined as follows: Let $\phi : S \mapsto \mathbb{R}$ be a strictly convex function defined on a convex set $S \subseteq \mathbb{R}^d$, such that ϕ is differentiable on $int(S)$, the interior of S. The Bregman divergence $D_\phi : S \times int(S) \mapsto [0, \inf)$ is defined as:

$$D_\phi(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{x}) - \phi(\mathbf{y}) - (\mathbf{x} - \mathbf{y}, \bigtriangledown\phi(\mathbf{y})), \tag{1}$$

where $\bigtriangledown\phi$ is the gradient of ϕ.

For example, for $\phi(\mathbf{x}) = \| \mathbf{x} \|^2$, $D_\phi(\mathbf{x}, \mathbf{y}) = \| \mathbf{x} - \mathbf{y} \|^2$, which is the Squared Euclidean Distance. Similarly, other forms of ϕ lead to other popular divergences such as Logistic Loss, Itakura-Saito Distance, Hinge Loss, Mahalanobis Distance and KL Divergence [PPL01, BMDG05].

Bregman Information: An important property of all Bregman divergences is as follows:

Theorem 2.1. [BMDG05]: *Let X be a random variable taking values in $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^n \subset C \subseteq \mathbb{R}^d$ following ν [4]. Given a Bregman divergence $D_\phi : C \times int(C) \mapsto [0, \inf)$, the problem*

$$\min_{\mathbf{c} \in C} E_\nu[D_\phi(X, \mathbf{c})]$$

has a unique minimizer given by $\mathbf{c}^* = \mu = E_\nu[X]$.

Banerjee et al. [BMDG05] refer to the corresponding minimum $E_\nu[D_\phi(X, \mathbf{c}^*)]$ as the *Bregman Information* of X. Both variance and mutual information are special cases of Bregman Information.

Bregman Hard Clustering: Banerjee et al. [BMDG05] describe a partitional clustering algorithm called *Bregman Hard Clustering* that exploits Theorem 2.1. Starting with a random initialization of k centers Bregman Hard Clustering repeats the following until convergence to a local minima: (1) assign each point to the closest center, as measured by the particular choice of D_ϕ, and (2) update the centers as the mean of points within each cluster. When D_ϕ is Squared Euclidean distance, Bregman Hard Clustering reduces to the K-Means algorithm, so one could view K-Means as a special case of Bregman Hard Clustering. A visible achievement of [BMDG05] was to show that a K-Means type algorithm exists for any Bregman divergence. However, a subtle but perhaps more consequential property of Bregman Hard Clustering is that different choices of D_ϕ result in clustering algorithms that are appropriate for very different types of datasets and problems; many of the special forms had been proposed, proven and applied as independent algorithms, such as the celebrated Linde-Buzo-Gray algorithm [LBG80, BGGM80], before Bregman Hard Clustering was formulated.

[4] Theorem 2.1 is more general in that it holds for any measure ν defined on the samples. Unless stated explicitly otherwise, we assume all points to have the same weight.

2.2 Density-based: Clustering a subset of data into multiple dense clusters

A variety of density-based methods have been developed that use different notions of "local" density to cluster only a part of the data and to prune the rest. For example, in DBSCAN [EKSX96], given a point that has at least *MinPts* points enclosed by a hypersphere of radius ϵ centered at the point, all points within the ϵ sphere are assigned to the same cluster. DBSCAN has the ability to find arbitrary shaped clusters, which is useful in certain problems. However, different choices for ϵ and *MinPts* can give dramatically different clusterings. OPTICS [ABKS99] proposed a visualization to make it easier to select these two parameters. DBSCAN is computationally efficient only for low-d spatial data where efficient indexing schemes are available, and is therefore popular in the database community for indexing 2-d and 3-d images.

DHC [JPZ03] is perhaps the first published work on applying density-based clustering to biological data. It proposes a density-based hierarchical clustering algorithm for time-series data. DHC provides a hierarchical grouping of time-series data that can be used to visually browse similar genes. The cluster hierarchy built by DHC uses the heuristic of *attraction* that assumes the data is uniformly distributed in a d-dimensional space. However, points in many real-life high dimensional datasets tend to reside in much lower dimensional manifolds [TdSL00] within the embedded space.

2.3 Finding a single dense region as a One Class

One Class Clustering: In One Class Clustering, also known as One Class Classification, the goal is to find a single, relevant subset of the data. One Class Classification algorithms are well-suited for problems where the space is well-sampled for the known positive classes, but the negative class is hard to characterize or sample. For example, if the goal is granting access to users of a facility based on a fingerprint identification system, the negative class can comprise of all the people on Earth *not* belonging to the set of people granted access to a facility. In such situations, it is more appropriate to characterize each target class separately rather than learning a k-class classifier. One way to characterize each class separately is by using a One Class Classifier.

OC-IB: Earlier approaches to One Class Classification [TD99, SBV95, SPST$^+$01] used convex cost functions that are good for finding large-scale structures, or correspondingly, for finding a small number of outliers. However, [CC04] showed that such methods are not appropriate when we want to find distinct dense regions covering only a small fraction of the data. For example, suppose the data is generated by two low-variance Gaussians and a larger variance background. On such data, One Class SVM based methods end up finding a solution centered in between the two Gaussians. Instead, [CC04] proposed a One Class Clustering algorithm called *One Class-Information Bottleneck* (OC-IB) that finds a *Bregmanian ball* centered on one of the two low-variance Gaussians.

Formally, a Bregmanian ball $B_\phi(r, \mathbf{c})$ with radius r and centroid $\mathbf{c}$ defines a volume in $\mathbb{R}^d$ such that all points $\mathbf{x}$ where $D_\phi(\mathbf{x}, \mathbf{c}) \leq r$ are enclosed by the ball. Furthermore, given a set $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^n$ of n points in $\mathbb{R}^d$, the cost of the ball is defined as the average $D_\phi(\mathbf{x}, \mathbf{c})$ of all points $\mathbf{p}_i \in \mathcal{G} \subseteq \mathcal{X}$ enclosed by it:

$$Q_{one}(\mathcal{G}, \mathbf{c}) = \frac{1}{|\mathcal{G}|} \sum_{i=1}^{|\mathcal{G}|} D(\mathbf{p}_i \in \mathcal{G}, \mathbf{c}), \tag{2}$$

OC-IB attempts to find the most number of enclosing points for some given cost threshold q_{max}, where the cost of the ball is defined by equation 2. OC-IB uses an Information Bottleneck approach described by [Slo02] to formulate the optimization problem. OC-IB takes a cost threshold q_{max} as input and starts with some seed location in $\mathcal{X}$ as the initial centroid $\mathbf{c}$ of the Bregmanian ball. Then, it uses a randomized iterative relocation algorithm to find a new centroid that encloses a progressively "denser" Bregmanian ball. More specifically, each iteration of OC-IB involves the following steps: (1) Randomly permute all the n data points. (2) For each of the n data points in the randomly permuted order do the following: if the point is not in $\mathcal{G}$, then add it to $\mathcal{G}$ if adding it keeps $Q_{one} < q_{max}$, else if the point is already in $\mathcal{G}$, remove it from $\mathcal{G}$ if before removing $Q_{one} >= q_{max}$. After each addition or deletion of a point, the Bregmanian ball centroid is re-estimated as the mean of the current member points and corresponding Q_{one} is updated in an efficient manner (see Appendix A for more detail). Steps (1) and (2) are repeated until convergence.

2.4 Clustering a subset of data into multiple overlapping clusters

In the context of clustering microarray data, discovering overlapping gene clusters is popular since many genes participate in multiple biological processes. Gene Shaving [Has00] uses PCA to find a small subset of genes that show strong expression change compared to the control sample, and allows them to be in multiple clusters. As we mentioned earlier, since only a small fraction of the genes are relevant for clustering in a given dataset, the ability of Gene Shaving to prune a large fraction of genes is particularly attractive. However, Gene Shaving greedily extracts one cluster at a time, and is computationally very expensive ($\geq O(n^3)$). Other greedy methods such as Plaid [LO02] decompose treat the original data $\mathcal{X}$ as a matrix, and decompose it into a set of sub-matrices that when added together reconstruct $\mathcal{X}$. This allows Plaid to also find overlapping clusters. However, matrix approximation methods for gene-expression datasets have only had partial success, in large part due to the highly unbalanced nature of the matrix; there are typically to the order of 10^2 experiments while there are to the order of 10^4 genes. A critical issue therefore continues to be the ability to select a small number of highly relevant genes for clustering, while selecting all the experiments as relevant.

3 Batch Ball One Class Clustering (BBOCC)

For the problem of finding a single dense region, OC-IB overcomes several limitations of traditional density-based clustering including: (1) being applicable to all Bregman divergences, (2) scalable to large datasets, with each iteration having $O(nd)$ space and time complexity, and (3) scalable to very high-dimensional datasets. However, one problem with OC-IB is its inability to control the size [5] of the cluster; because of problems with local minima, for a fixed q_{max} as input, each trial of OC-IB starting from a different seed location can result in a different

[5] In this paper, the *size* of a cluster denotes the number of data-points in that cluster.

and quite unpredictable cluster size and quality. Moreover, for some applications, the ability to control what fraction of data would form the dense cluster is important.

There are two possible problem formulations corresponding the cost function Q_{one} (Equation 2) used by OC-IB:

Problem definition 1*: Find the largest cluster $\mathcal{G} \subset \mathcal{X}$ with cost $Q_{one} \leq q_{max}$.*

and,

Problem definition 2*: Find a cluster $G \subset \mathcal{X}$ of size s that has the smallest cost.*

OC-IB corresponds to problem definition 1. However, problem definitions 1 and 2 are dual of each other, since for a given q_{max} there exists a largest s, and for a given s there exists a smallest q_{max}. The main difference from a user's point of view is that an algorithm based on definition 1 takes an integer s as input and searches for a solution with the smallest cost, while in definition 2, it takes a threshold $q_{max} \in \mathbb{R}$ as input and searches for a cluster with the largest size with cost less than q_{max}.

3.1 BBOCC-Q

OC-IB uses a randomized algorithm to find a local minima for problem definition 1; points are randomly added or discarded until convergence while constantly updating the cluster representative after each addition/deletion. Alternatively, a "batch update" local search algorithm can also be developed where points are added and discarded in batches. We start with the observation:

Proposition 3.1. *For a given centroid $\mathbf{c}$ and a threshold q_{max}, the largest cluster $\mathcal{G}$ with cost $Q_{one}(\mathcal{G}, \mathbf{c}) < q_{max}$ consists of the points closest to $\mathbf{c}$.*

Batch Ball One Class Clustering-Q (BBOCC-Q, Algorithm 1), utilizes the same set of inputs as OC-IB; a Bregman divergence D_ϕ, a seed start location $\mathbf{c}$ and a cost threshold q_{max}. Then, it iterates through the following steps until convergence: (1) sort $\mathcal{X}$ by distance from $\mathbf{c}$ (lines 9-10), (2) add points closest to $\mathbf{c}$ to an empty set $\mathcal{G}$ until adding more would cause cost to exceed q_{max} (lines 12-17), and (3) update $\mathbf{c}$ as the mean of all the points in $\mathcal{G}$ (line 18).

Proposition 3.2. *Algorithm 1 is guaranteed to converge to a local optima.*

For the sake of completeness, we have included the proofs for Propositions 3.1 and 3.2 in Appendix B.

If heap-sort is used at line 10 of Algorithm 1 and when clusters found are smaller than $n/\log(n)$, the resulting time complexity of BBOCC-Q is $O(nd)$, same as OC-IB. In our empirical comparisons however, we found that OC-IB takes lesser number of iterations to converge (Figure 6, (c)). However, with an $O(max(nd, n\log(n)))$ time complexity per iteration in general for large cluster sizes, and a space complexity of$O(nd)$, BBOCC-Q is still quite fast and scalable to very large datasets.

Algorithm 1 BBOCC-Q

Input: Data points $\mathcal{X} = \{\mathbf{x}\}_{i=1}^{n}$ in $\mathbb{R}^d$, Bregman divergence D_ϕ, initial cluster centroid $\mathbf{c}_{in}$ (optional), desired cost threshold q_{max}.

Output: Solution set $\mathcal{G}$ containing member points, and the corresponding cluster centroid $\mathbf{c}^*$.

if $\mathbf{c}_{in} = \varnothing$ **then**
  Pick randomly a point from $\mathcal{X}$ and assign it to $\mathbf{c}_{in}$
end if
$\mathbf{c} = \mathbf{c}_{in}; \mathcal{G}_l = \varnothing; \mathcal{G} = \varnothing; s_p = \infty; s_q = \infty;$
repeat
  $\mathbf{c}_p = \mathbf{c}; s_{pp} = s_p; s_p = s_q; \mathcal{G}_l = \mathcal{G}$
  distAll $= \{D_\phi(\mathbf{x}_i, \mathbf{c})\}_{i=1}^{n}$
  $[\mathbf{val}, \mathbf{idx}] = sort(\mathbf{distAll})$
  $\mathcal{G}_t = \varnothing; q_t = 0; s_t = 0;$
  while $(q_t < q_{max})$ **do**
    $q = q_t; \mathcal{G} = \mathcal{G}_t; s_q = s_t$
    $s_t = s_t + 1$
    $q_t = (q_t(s_t - 1) + val(s_t))/s_t$
    $\mathcal{G}_t = \mathcal{G}_t \cup \mathbf{x}_{idx(s_t)}$
  end while
  $\mathbf{c} = \frac{1}{|\mathcal{G}|} \sum_{i=1}^{|\mathcal{G}|} \mathbf{x}_{idx(i)}$
until $(\mathcal{G}_l == \mathcal{G}) \wedge (s_{pp} == s_q)$
$\mathbf{c}^* = \mathbf{c}$

3.2 BBOCC-S

We now propose an alternative to BBOCC-Q that uses the dual problem definition (definition 2) where the input is a fixed cluster size s, instead of q_{max}. We begin with a property similar to Proposition 3.1 that also holds true:

Proposition 3.3. *For a given centroid* $\mathbf{c}$ *and cluster size* s *the cluster* $\mathcal{G}$ *with the smallest cost* $Q_{one}(\mathcal{G}, \mathbf{c})$ *consists of the* s *points closest to* $\mathbf{c}$*.*

This follows from the observation that for any given $s < n$, the subset $\mathcal{P}$ of size s picked from a set $\mathcal{Q}$ of n real numbers having the smallest possible average value consists of the s smallest numbers in $\mathcal{Q}$. We can now modify Algorithm 1 by using Proposition 3.3 instead of 3.1 for selecting points around a given centroid in each iteration, and obtain BBOCC-S. Algorithm 2 requires a Bregman divergence D_ϕ, a seed start location $\mathbf{c}$ and the desired cluster size s as input (instead of a cost threshold). Then, it iterates through the following steps until convergence: (1) sort $\mathcal{X}$ by distance from $\mathbf{c}$ (lines 9 to 10), (2) form $\mathcal{G}$ as the set of s closest points to $\mathbf{c}$ (line 11), and (3) update $\mathbf{c}$ as the mean of all the points in $\mathcal{G}$ (line 12).

Proposition 3.4. *Algorithm 2 is guaranteed to converge to a local optima.*

The proof for the convergence of BBOCC-S is similar to that of BBOCC-Q, and comes from the observation that in each iteration the cost declines monotonically; after the center update (line 12, Algorithm 2), the cost declines monotonically because of Theorem 2.1, and after the reassignment (line 11) the cost declines because of Proposition 3.3.

The time and space complexity of BBOCC-S is similar to that of BBOCC-Q, with $O(max(nd, s\log(n)))$ time-complexity for each iteration (when a heap sort is used at line 10 in Algorithm 2) and

Algorithm 2 BBOCC-S

Input: Data points $\mathcal{X} = \{\mathbf{x}\}_{i=1}^{n}$ in $\mathbb{R}^d$, Bregman divergence D_ϕ, initial cluster centroid $\mathbf{c}_{in}$ (optional), desired cluster size s.
Output: Solution set $\mathcal{G}$ containing s member points, and the corresponding cluster centroid $\mathbf{c}^*$.
  if $\mathbf{c}_{in} = \varnothing$ **then**
    Pick randomly a point from $\mathcal{X}$ and assign it to $\mathbf{c}_{in}$
end if
  $\mathbf{c} = \mathbf{c}_{in}; \mathcal{G}_l = \varnothing; \mathcal{G} = \varnothing; q_p = \infty; q = \infty;$
  repeat
    $\mathbf{c}_p = \mathbf{c}; q_{pp} = q_p; q_p = q; \mathcal{G}_l = \mathcal{G}$
    $\mathbf{distAll} = \{D_\phi(\mathbf{x}_i, \mathbf{c})\}_{i=1}^{n}$
    $[\mathbf{val}, \mathbf{idx}] = sort(\mathbf{distAll})$
    $\mathcal{G} = \{\mathbf{x}_{idx(i)}\}_{i=1}^{s}$
    $\mathbf{c} = 1/s \sum_{i=1}^{s} \mathbf{x}_{idx(i)}$
    $q = Q_{one}(\mathcal{G}, \mathbf{c})$ (from Equation 2)
  until $(\mathcal{G}_l == \mathcal{G}) \wedge (q_{pp} == q)$
$\mathbf{c}^* = \mathbf{c}$

a space complexity of $O(nd)$. BBOCC-S contains two "enhancements" over OC-IB: (1) the randomized updates are replaced by a batch update, and (2) the Bregmanian ball has a fixed size rather than a cost in each iteration. BBOCC-Q can be considered as an algorithm with only the first enhancement over OC-IB, i.e. the batch update.

Enhancement (1), i.e. the batch update, also makes BBOCC faster than OC-IB with certain types of distance measures (see Note 2, Appendix A). Enhancement (2), i.e. the ability to control the size of the resultant cluster also provides BBOCC-S a major usability advantage over BBOCC-Q. However, we present empirical evidence in the Section 11.3 that BBOCC-S also significantly outperforms OC-IB in terms of the quality of the local minima, and that both these enhancements contribute to the improvement.

4 Bregman Bubble Clustering

Although the BBOCC-S algorithm proposed in the previous section provides some key advantages over OC-IB, while inheriting all the other desirable traits [6], it can only find one dense region. However, in many clustering problems, the useful subset consists of multiple dense regions. One way to discover multiple clusters using BBOCC would be to apply it sequentially to extract the clusters one by one. However, a more direct approach would be an algorithm that can simultaneously find multiple, i.e. $k > 1$ dense regions, but preserves all the desirable properties of BBOCC. In this section we present a straightforward generalization of BBOCC that has such properties. First, the notion of a single dense Bregmanian ball is extended to the idea of multiple dense regions called *Bregman Bubbles*. We then present an algorithm called *Bregman Bubble Clustering* or BBC, that can find k dense Bregman bubbles using a local search approach. BBC gives rise to BBOCC-Q and BBOCC-S as special cases when $k = 1$.

[6] such as scalability to large and high dimensional datasets, and applicability to all Bregman divergences.

4.1 Cost Function

Let $\mathcal{X} = \{\mathbf{x}\}_{i=1}^{n} \subset C \subseteq \mathbb{R}^d$ be the set of data points. Let $\mathcal{G} \subset \mathcal{X}$ represent a non-exhaustive clustering consisting of k clusters $\{\mathcal{C}_j\}_{j=1}^{k}$ with $\mathcal{X} \setminus \mathcal{G}$ points that are "don't care", i.e., they do not belong to any cluster. For a given Bregman Divergence $D_\phi(\mathbf{x}, \mathbf{y}) \mapsto [0, \infty)$, and a set of k cluster representatives $\{\mathbf{c_j}\}_{j=1}^{k} \in \mathbb{R}^d$ for the k clusters in clustering $\mathcal{G} = \{\mathcal{C}_j\}_{j=1}^{k}$, we define the cost Q_b as the average distance of all points in $\mathcal{G}$ from their assigned cluster representative:

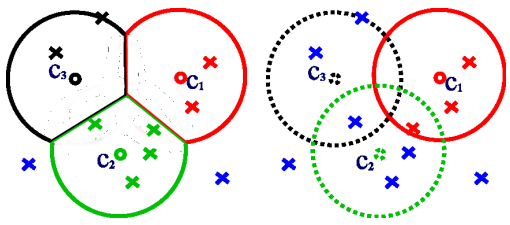
$$Q_b(\mathcal{G}, \{\mathbf{c_j}\}_{j=1}^{k}) = \frac{1}{|\mathcal{G}|} \sum_{j=1}^{k} \sum_{i:\mathbf{x}_i \in \mathcal{C}_j}^{|\mathcal{C}_j|} D_\phi(\mathbf{x}_i, \mathbf{c}_j), \tag{3}$$

4.2 Problem Definition

Given s, k and D_ϕ as inputs, where s out of n points from $\mathcal{X}$ are to be clustered into a clustering $\mathcal{G} \subseteq \mathcal{X}$ consisting of k clusters, where $1 \leq k < n$ and $k \leq s \leq n$, we define the clustering problem as:

Problem Definition 3: *Find the clustering $\mathcal{G}$ with smallest cost Q_b such that $|\mathcal{G}| = s$.*

4.3 Bregman Bubbles



(a) (b)

Figure 1: An illustration showing (a) three Bregman bubbles, and (b) a Bregmanian ball (solid line), and two other possible balls (dotted lines). The union of the points enclosed by the three possible balls in (b) is the same as the set of points enclosed by the three bubbles.

Given a set of k cluster representatives, and a fixed s, it can be shown that the clustering that minimizes Q consists of: (1) the assignment phase, where each point is assigned to the nearest cluster representative, and (2) picking points closest to their representatives first until s points are picked. Let r_{max} represent the distance of the last (s^{th}) picked point from its cluster representative.

These clusters can be viewed as k *Bregman bubbles* such that: (1) they are either pure Bregmanian balls of radius $r \leq r_{max}$ or are (2) *touching* bubbles that form when two or more Bregmanian balls, each of radius r_{max} overlap. Two Bregmanian balls $B_\phi(\mathbf{c1}, r_1)$ and $B_\phi(\mathbf{c2}, r_2)$ are said to overlap when $\exists \mathbf{x} : (D_\phi(\mathbf{x}, \mathbf{c1}) < r_1) \wedge (D_\phi(\mathbf{x}, \mathbf{c2}) < r_2)$. At the point of contact, the touching bubbles form linear boundaries[7] that result from assigning points to the

[7]This can be shown to be true for all Bregman divergences [BMDG05].

closest cluster representative. For the part of its boundary where a bubble does not touch any other bubble, it traces the contour of a Bregmanian ball of radius r_{max}. Therefore, bubbles arise naturally as the optimum solution for Q for a given s, k and D_ϕ.

Figure 1 illustrates a 2-D example of Bregman bubbles vs. balls. Unlike Bregmanian balls, the boundary of the Bregman bubbles can only be defined in the context of other bubbles touching it. It is important to note that the volume of the convex hull of points in one bubble could be smaller than that of the adjacent touching bubble, and the bubbles could also have different number of points assigned to them.

4.4 BBC-S: Bregman Bubble Clustering with fixed clustering size

For most real life problems, even for a small s, finding the globally optimal solution for problem definition 1 would be too slow. However, a fast iterative relocation algorithm that guarantees a local minima exists. *Bregman Bubble Clustering-S* (BBC-S, Algorithm 3) starts with k centers and a size s as input. Conceptually, it consists of three stages: (1) the assignment phase, where each point is assigned to the nearest cluster representative, (2) picking points closest to their representatives first until s points are picked, and (3) updating the centers. It is interesting to note that stages 1 and 3 of BBC-S are identical to the Assignment Step and the Re-estimation step of the Bregman Hard Clustering (Section 2.1), properties that lead to the unification described in Section 7. Stages 1, 2 and 3 are repeated until there is no change in assignment between two iterations - i.e. the algorithm converges. Algorithm 3 describes a more detailed implementation of BBC-S where line number 10 represents Stage 1, lines 14 to 18 map to Stage 2, while lines 22-24 represent Stage 3. We randomly pick k data points from $\mathcal{X}$ as the starting cluster representatives, but alternative initialization schemes could be implemented.

Proposition 4.1. *Algorithm 3 is guaranteed to converge to a local minima for all Bregman divergences.*

This follows from the observation that at each iteration the cost Q_b either declines or stays the same. It is easy to show that for a given set of cluster representatives, Stage 1 and 2 result in assignments with the lowest possible cost. Therefore, after Stage 1 and 2, the cost cannot increase but can decrease. If no cluster assignments change in Stage 1 and 2, the cost stays the same and the algorithm converges. Similarly the cost Q_b at Stage 3 can either decline or stay the same because of Theorem 2.1. Using heap-sort at line 12 of Algorithm 3, each iteration of BBC-S takes $O(max(nkd, s\log(n)))$ time making it really fast.

4.5 BBC-Q, dual formulation of Bregman Bubble Clustering with fixed cost

Just as BBOCC-Q is the dual of BBOCC-S, an alternative formulation of the Bregman Bubble Clustering called BBC-Q is possible where a threshold cost q_{max} is specified as input rather than the size s. Given q_{max}, k and D_ϕ as inputs:

Problem Definition 4*: Find the largest $\mathcal{G}$ with cost $Q_b \leq q_{max}$.*

We can show that this definition also results in Bregman bubbles as the optimal solution for a set of k cluster representatives. Definitions 3 and 4 are equivalent, since for a given q_{max} there exists a largest s for k bubbles, and for the same s, the same solution has the same smallest

Algorithm 3 BBC-S

Input: Set $\mathcal{X} = \{\mathbf{x}\}_{i=1}^{n} \subset C \subseteq \mathbb{R}^d$, Bregman divergence D_ϕ, no. of clusters k, desired clustering size s.

Output: Partitioning $\mathcal{G}^*$ containing k clusters$\{\mathcal{C}_j\}_{j=1}^{k}$, and the corresponding k cluster representatives $\{\mathbf{c_j^*}\}_{j=1}^{k}$.

Method:

if $\{\mathbf{c_j}\}_{j=1}^{k} = \varnothing$ **then**

Initialize cluster representatives: $\{\mathbf{c_j}\}_{j=1}^{k} \in C$

end if

$\mathcal{G}^l = \varnothing; \mathcal{G} = \varnothing; q = \infty; q_p = \infty;$

repeat

for $i = 1$ to n **do**

$[d_i^{min}, lab_i] = \min_{j=1}^{k}(D_\phi(\mathbf{x_i}, \mathbf{c_j}))$

end for

$[val, idx] = sort(\mathbf{d}^{min})$

$q^{tmp} = 0; s^c = 0; \{\mathcal{C}_j\}_{j=1}^{k} = \varnothing$

while $(s^c < s)$ **do**

$s^c = s^c + 1;$

$q^{tmp} = q^{tmp} + val(s^c)$

Add $\mathbf{x_{idx(s^c)}}$ to cluster $\mathcal{C}_{lab(idx(s^c))}$

end while

$\{\mathbf{c_j^p}\}_{j=1}^{k} = \{\mathbf{c_j}\}_{j=1}^{k}$

$q^{pp} = q; q^p = q; q = q^{tmp}/s$

$\mathcal{G}^l = \mathcal{G}; \mathcal{G} = \{\mathcal{C}_j\}_{j=1}^{k}$

for $j = 1$ to k **do**

$\mathbf{c_j} = \frac{1}{|\mathcal{C}_j|} \sum_{i:\mathbf{x}_i \in \mathcal{C}_j}^{|\mathcal{C}_j|} \mathbf{x}_i$

end for

until $(\mathcal{G}^l == \mathcal{G}) \wedge q_{pp} == q$

Return $\{\mathbf{c_j^*}\}_{j=1}^{k} = \{\mathbf{c_j}\}_{j=1}^{k}; \mathcal{G}^* = \mathcal{G}$

possible cost q_{max}. Algorithm 3 can be easily modified to work with q_{max} by modifying Stage (2) to stop adding points when the cost is more than q_{max}.

Proposition 4.2. *BBC-Q is guaranteed to converge to a local minima for all Bregman divergences.*

The proof for the above follows along similar lines as that for Proposition 4.1. It can also be shown that BBC-S and BBC-Q reduce to BBOCC-S and BBOCC-Q respectively when $k = 1$ (Section 7). BBC-S has the same type of advantage over BBC-Q that BBOCC-S has over BBOCC-Q; while bubbles in BBC-S iterations in sparse regions expand until s points are covered, BBC-Q has a fixed q_{max} and does not have this property. We show later (Figure 7, Section 11.3) that this difference results in BBOCC-S performing better than BBOCC-Q when measured in terms of the intrinsic cost Q_{one}. Similarly, as one might expect, BBC-S also gives better results than BBC-S in terms of Q_b, and it is not possible to control the number of points that get assigned to clusters in BBC-Q. For brevity, we do not repeat these comparisons between BBC-Q and BBC-S as they are analogous to that between BBOCC-Q and BBOCC-S, the special cases (for $k = 1$) of BBC-Q and BBC-S respectively. Unless stated explicitly otherwise, the discussion on Bregman Bubble Clustering in the rest of the paper is restricted to BBC-S, i.e. BBC with a fixed s as input.

5 Soft Bregman Bubble Clustering (Soft BBC)

5.1 Bregman Soft Clustering

In hard clustering, each point is assigned to one cluster. In *soft clustering*, each point can be a "partial" member of all of the clusters. If the total assignment weights for a given point over all clusters is normalized to 1, we can interpret the soft assignments as probabilities. One popular way to model such probabilistic assignments is to assume that the set of observed points come from a mixture of k distributions whose parameters are estimated based on the observed data. Once the parameters are estimated, the probabilistic membership of each point to each of the clusters can be computed. Banerjee et al. [BMDG05] proposed a soft clustering algorithm called *Bregman Soft Clustering* as a mixture model consisting of k distributions, taken from the family of *regular exponential distributions*. They went on to prove the following important result:

Theorem 5.1. *There is a bijection between regular exponential families and regular Bregman divergences.*

This bijection is expressed by:

$$p_{(\psi,\theta)}(\mathbf{x_s}) = exp(-\beta D_\phi(\mathbf{x_s}, \mu)) f_\phi(\mathbf{x_s}) \tag{4}$$

where ϕ is a convex function, and the conjugate function of ψ, D_ϕ is the corresponding Bregman divergence, $p_{(\psi,\theta)}$ is the corresponding regular exponential distribution with cumulant ψ, f_ϕ is a uniquely determined normalizing function that depends on the choice of ϕ, β is a scaling factor, μ is the expectation parameter, θ are the natural parameters of p^ϕ, and $\mathbf{x_s}$ is the sufficient statistics vector corresponding to $\mathbf{x}$. For the sake of notational simplicity, for the rest of the paper, unless stated explicitly otherwise, when we mention $\mathbf{x}$ we implicitly refer to the sufficient statistics of $\mathbf{x}$, i.e. $\mathbf{x_s}$. The exact transformation of $\mathbf{x}$ to $\mathbf{x}_s$ depends upon the choice of the distribution $p_{(\psi,\theta)}$. For example, for the Squared Euclidean distance as D_ϕ, the sufficient statistics space $\mathbf{x_s} = \mathbf{x}$.

Regular Bregman divergences form a large class of divergences for which a corresponding regular exponential distribution exists. Examples of regular Bregman divergences (and the corresponding exponential distribution) that have been popular for both hard and soft clustering models include squared Euclidean Distance (Gaussian distribution), KL-divergence (multinomial distribution) and Itakura-Saito distance.

Banerjee et al. [BMDG05] not only showed a formal unification of the various hard partitional clustering methods as special cases of Bregman hard clustering, and the corresponding exponential distribution soft clustering models as special cases of Bregman soft clustering, but went on to show that for all regular Bregman divergences, Bregman Hard Clustering falls out as a special case of Bregman Soft Clustering. For example, for the Squared Euclidean distance as D_ϕ, Bregman Hard Clustering maps to the standard K-Means algorithm, and the corresponding Bregman Soft Clustering maps to a mixture of spherical Gaussians with a fixed variance σ^2, popularly known as *soft K-Means*, and μ maps to Gaussian mean $\mathbf{a}$, $f_\phi(\mathbf{x_s}) = \frac{1}{\sqrt{(2\pi\sigma^2)^d}}, \beta = \frac{1}{2\sigma^2}, D_\phi(\mathbf{x_s}, \mu) = \beta||\mathbf{x}-\mathbf{a}||^2$, $\theta = \frac{\mathbf{a}}{\sigma^2}$, and $\psi(\theta) = \frac{\sigma^2}{2}||\theta||^2$. The soft K-Means model reduces to K-Means when the variance σ^2 of the k Gaussians is set to 0^+ that corresponds to $\beta \to \infty$ in Equation 4.

5.2 Motivations for developing Soft BBC

Bregman Bubble Clustering can be thought of as a non-exhaustive hard clustering where points can belong to either one of the k clusters or to a "don't care" group, while there is no such "don't care" grouping in Bregman Hard Clustering. The generative model for Bregman Soft Clustering consists of a mixture of k regular exponential distributions of the form p^{ϕ} corresponding to the k clusters. Correspondingly, *Soft Bregman Bubble Clustering* (Soft BBC) can be formulated as modeling the data as a mixture of k distributions from the exponential family and an additional "background" distribution that corresponding to the "don't care" points. Since we are trying to find k dense clusters, for a good solution the "don't care" group should be the least dense. One way to model this low density background is with a uniform distribution. The goal of building such a Soft BBC model is to give us deeper insights into the implicit modeling assumptions behind BBC.

5.3 Generative Model

Let $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^n$ be the dataset consisting of n i.i.d. points and k be the desired number of clusters. We propose Soft BBC as a generative model containing k mixture components corresponding to k dense clusters labeled 1 to k and one uniform background distribution labeled 0, where each data point is assumed to be generated by a unique but *unknown* component. Let $\mathcal{Y} = \{Y_i\}_{i=1}^n$ be the hidden random variables corresponding to the mixture components associated with the data points, where Y_i can take one of $k+1$ possible values from 0 to k. In the absence of any other information, the distribution of $\mathcal{Y}$ only depends upon the priors. Hence the model probability of the data points is given by:

$$p(\mathbf{x_i}) = \sum_{j=1}^{k} \alpha_j p_{(\psi,\theta)}(\mathbf{x}_i|\theta_j) + \alpha_0 p_0, [i]_1^n \tag{5}$$

where $\{\alpha_j\}_{j=1}^k$ and $\{p_{(\psi,\theta)}(\cdot|\theta_j)\}_{j=1}^k$ denote the priors and the conditional distributions of the k clusters, while α_0 and p_0 denotes the prior probability and the probability density of the uniform distribution. Since the data points are assumed to be i.i.d., the log-likelihood of the observed data (or the incomplete log-likelihood) is given by:

$$L(\Theta|\mathcal{X}) = \sum_{i=1}^{n} \log(\sum_{j=1}^{k} \alpha_j p_{(\psi,\theta)}(\mathbf{x}_i|\theta_j) + \alpha_0 p_0) \tag{6}$$

where Θ denotes all the parameters (priors and mixture component parameters). Maximizing the above data likelihood is a natural approach to fitting this generative model to the data. However, it is non-trivial to directly optimize the likelihood function due to the presence of mixture components.

5.4 Soft BBC EM Algorithm

Soft BBC (Algorithm 4) uses an EM [MK97, NH98] algorithm for mixture density estimation using the hidden variables $\mathcal{Y}$. For the observed information as the data-points $\mathcal{X}$, the hidden variables $\mathcal{Y}$, the possible parameters over which we could try to maximize the likelihood

Algorithm 4 Soft BBC

Input: Set $\mathcal{X} = \{\mathbf{x}\}_{i=1}^{n} \subset C \subseteq \mathbb{R}^d$, Bregman divergence D_ϕ, no. of clusters k, p_0, specifying the background distribution, α_0 for Case B.

Output: Θ^*, local maximizer of $L(\Theta|\mathcal{X})$ (equation 6) where $\Theta = \{\{\theta_j, \alpha_j\}_{j=1}^{k}, \alpha_0\}$ for case (A) and $\{\theta_j, \alpha_j\}_{j=1}^{k}$ for case (B), soft partitioning $\{\{p(Y_i = j|\mathbf{x}_i)\}_{j=0}^{k}\}_{i=1}^{n}$.

Method:

Initialize p_0, $\{\theta_j, \alpha_j\}_{j=1}^{k}$ with some $0 \leq p_0 < 1$, $\theta_j \in C$, $\alpha_j \geq 0$, such that $\sum_{j=0}^{k} \alpha_j = 1$.

repeat

{The **E** Step}

for $i = 1$ to n **do**

for $j = 0$ to k **do**

$p(Y_i = j|\mathbf{x}_i)$ is computed from equation (8) and (9), where $p_{(\psi,\theta)}(\mathbf{x}_i|\theta_j)$ is defined by equation 4.

end for

end for

{The **M** Step}

for $j = 0$ to k **do**

Update α_j using equation 11 for case A and 14 for case B.

Update θ_j using equation 13.

end for

until convergence

(Equation 6) are $\{\alpha_j\}_{j=0}^{k}$, $\{\theta_j\}_{j=1}^{k}$ and p_0, with the only constraint being that the priors should sum to 1, i.e. $\sum_{j=0}^{k} \alpha_j = 1$.

Cannot optimize for p_0: Since p_0 is a uniform distribution by definition, $1/p_0$ defines the volume of its domain. This domain should include the convex hull of $\mathcal{X}$, which yields an upper bound for p_0. In Equation 6, keeping all other parameters constant, a lower value of p_0 will always result in a lower likelihood. For now, we only consider the case where p_0 is set to a fixed value.

Now, the only parameters we need to optimize over are the priors $\{\alpha_j\}_{j=0}^{k}$ and the exponential mixture parameters $\{\theta_j\}_{j=1}^{k}$. We consider two slightly different scenarios: (A) where α_0 is a variable parameter, and (B) where α_0 is a fixed value ≤ 1. To maximize the log-likelihood function, we adopt a standard EM-based approach and first construct the negative free energy function:

$$F(\tilde{P}, \Theta) = \sum_{i=1}^{n} E_{\tilde{p}(Y_i,\mathbf{x}_i)}[\log p(\mathbf{x}_i, Y_i|\Theta)] - \sum_{i=1}^{n} E_{\tilde{p}(Y_i,\mathbf{x}_i)}[\log p(Y_i|\mathbf{x}_i)]$$

where $\tilde{P} = \{\{\tilde{p}(Y_i = j|\mathbf{x}_i)\}_{i=1}^{n}\}_{j=1}^{k}$ are the current estimates of $\mathcal{Y}$. It can be shown that the EM procedure with the **E** and **M** steps alternately optimizing $F(\tilde{P}, \Theta)$ over $\tilde{P}$ and Θ is guaranteed to converge to a local maxima $\tilde{P}^*$ and Θ^*. Furthermore, it can be shown that a local maxima of $F(\tilde{P}, \Theta)$ leads to a local maxima on the original likelihood given by Equation 6. Hence we will now focus on obtaining the updates involved in the **E** and **M** steps for the two cases.

Case (A): α_0 **is not fixed**

E-Step: In this step we optimize $F(\tilde{P}, \Theta)$ (Equation 7) over $\tilde{P}$ under the constraints that

the $\sum_{j=0}^{k} \tilde{p}(Y_i = j|\mathbf{x}_i) = 1, [i]_1^n$, and $\tilde{p}(Y_i = j|\mathbf{x}_i) \geq= 0, \forall i, j$. Using Lagrange multipliers $\{\lambda_i\}_{i=1}^n$ for the n equality constraints and taking derivatives w.r.t. $\tilde{p}(Y_i = j|\mathbf{x}_i)$, we obtain the update equation for re-estimating the probability of each point coming from any of the 0 to k components, given the current model parameters:

$$\log p(\mathbf{x}_i, Y_i = j|\Theta) - 1 - \log \tilde{p}(Y_i = j|\mathbf{x}_i) - \lambda_i = 0 \tag{7}$$

where $p(\mathbf{x}_i, Y_i = j|\Theta)$ is $\alpha_j p_{(\psi,\theta)}(\mathbf{x}_i|\theta_j)$ for $1 \leq j \leq k$ and $\alpha_0 p_0$ for $j = 0$. On eliminating the Lagrange multipliers, we obtain:

$$\tilde{p}(Y_i = j|\mathbf{x}_i)^* = \frac{\alpha_j p_{(\psi,\theta)}(\mathbf{x_i}|\theta_j)}{\sum_{j=1}^{k} \alpha_j p_{(\psi,\theta)}(\mathbf{x}_i|\theta_j) + \alpha_0 p_0}, 1 \leq j \leq k \tag{8}$$

$$= \frac{\alpha_0 p_0}{\sum_{j=1}^{k} \alpha_j p_{(\psi,\theta)}(\mathbf{x}_i|\theta_j) + \alpha_0 p_0}, j = 0 \tag{9}$$

M-Step: In this step we optimize $F(\tilde{P}, \Theta)$ over Θ under constraints $\sum_{j=0}^{k} \alpha_j = 1$ and $\alpha_j \geq 0, \forall j$. It can be shown that the inequality constraints are not binding. Using Lagrange multiplier ζ for the constraint and taking derivatives w.r.t. $\alpha_j, [j]_0^k$, we obtain:

$$\sum_{i=1}^{n} \frac{\tilde{p}(Y_i = j|\mathbf{x}_i)}{\alpha_j} + \zeta = 0, [j]_0^k \tag{10}$$

and on eliminating ζ, we obtain:

$$\alpha_j^* = \frac{\sum_{i=1}^{n} \tilde{p}(Y_i = j|\mathbf{x}_i)}{n}, [j]_0^k \tag{11}$$

Note that the update equation for the background distribution prior, α_0, turns out to be the same as that for the exponential mixture distributions α_1 to α_k. The optimal mixture component parameter estimation can be obtained by setting derivatives over $\{\theta_j\}_{j=1}^n$ to 0 as follows:

$$\sum_{i=1}^{n} \tilde{p}(Y_i = j|\mathbf{x_i}) \nabla_{\theta_j} p_{(\psi,\theta)}(\mathbf{x}_i|\theta_j) = 0 \tag{12}$$

This results in the update equation for the exponential distribution mixtures $\{\theta\}_{j=1}^k$ as the weighted average of $\mathbf{x}$ [BMDG05]:

$$\theta_j = \frac{\sum_{i=1}^{n} p(Y_i = j|\mathbf{x}_i)\mathbf{x}_i}{\sum_{i=1}^{n} p(Y_i = j|\mathbf{x}_i)} \tag{13}$$

An example of re-estimation of mixture component parameters for Gaussians is described in more detail in Section 8.

Case (B): α_0 is fixed

E-Step: Since keeping α_0 fixed does not result in any additional constraints, this step is identical to that of case A.

M-Step: Keeping α_0 constant modifies the constraints on the priors so that we now require $\sum_{j=1}^{k} \alpha_j = 1 - \alpha_0$ and $\alpha_j \geq 0, \forall j$. As before, the inequality constraints are not binding and by using a Lagrange multiplier and taking derivatives, we arrive at:

$$\alpha_j^* = (1-\alpha_0)\frac{\sum_{i=1}^{n}\tilde{p}(Y_i=j|\mathbf{x}_i)}{\sum_{j=1}^{k}\sum_{i=1}^{n}\tilde{p}(Y_i=j|\mathbf{x}_i)} \tag{14}$$

The optimal mixture component parameters are obtained exactly as in case A.

5.5 Choosing an appropriate p_0

For case (A) of the Soft BBC algorithm, one can argue that the parameter α_0 is essentially a function of p_0 given by the relation (from the M step):

$$\alpha_0 = \frac{1}{n}\sum_{i=1}^{n}\frac{\alpha_0 p_0}{\sum_{j=1}^{k}\alpha_j p_{(\psi,\theta)}(\mathbf{x}_i|\theta_j)+\alpha_0 p_0} \tag{15}$$

Using this relation, for a given α_0 and a set of mixture component parameters, it is possible to solve for p_0. But one cannot do this in the EM framework since the best value for p_0 is always the highest possible one. However this relationship allows us to calculate the value of p_0 for the initial seed parameters. Although an optimization routine can also be used, a fast approximation of p_0 can be estimated by (1) performing the first E step (Equations 8 and 9), then (2) computing the $p_{max}^i = \max_{j=0}^{k}(p(Y_i=j|\mathbf{x}_i))$ for each $\mathbf{x}_i$, and then (3) picking p_0 as the s^{th} largest value in $p_{max}^i[i]_1^n$ where $s = \lceil \alpha_0 n \rceil$. In practice the following works better: (i) Compute $p_0 = p_0^1$ using some initial $\{\theta_j\}_{j=1}^k$ and $\{\alpha_j\}_{j=1}^k$ and run Soft BBC to convergence to obtain $\{\theta_j^1\}_{j=1}^k$ and $\{\alpha_j^1\}_{j=1}^k$, and then (ii) recompute p_0 using $\{\theta_j^1\}_{j=1}^k$ and $\{\alpha_j^1\}_{j=1}^k$ from Step (i) and run Soft BBC a second time to convergence with this new p_0.

6 Improving local search: Pressurization

6.1 Motivation

In Section 3.2, we theorized that BBOCC-S performs better than BBOCC-Q because of the ability of the Bregmanian ball in BBOCC-S to expand to cover s points in each iteration, and then shrink as the ball moves to a denser region. In contrast, BBOCC-Q uses a fixed cost q_{max}, which causes problems in sparse regions; very few points get enclosed in the ball causing premature convergence. BBC-S shows a similar advantage over BBC-Q, since the number of points assigned to k clusters in BBC-S remains constant.

BBC is able to find locally dense regions because of its ability to explicitly ignore large amounts of data by considering only points close to the cluster representatives for cluster membership. During each iteration, the bubble representatives move to a better nearby location. But when the dense bubbles are naturally small, i.e. threshold s is small, only a few close neighbors get assigned, thereby decreasing the mobility of the representatives at each iteration. Although BBC-S and BBOCC-S "expand" the bubbles in sparse regions, this effect is still limited for very small values of s. This makes it difficult for even BBC-S or BBOCC-S to find small, dense regions far from the initial seed locations. On the other hand, starting with a large s would be contrary to the goal of finding *small* dense regions. Is there a way to improve the quality of the local search while still discovering clusters encompassing only a small number of points?

6.2 BBC-Press

We first introduce a concept called *Bregman bubble pressure* that has properties analogous to that of pressure around air bubbles in a body of water on Earth; when air-bubbles rise in a column of water, the outside pressure drops, and the bubbles expand. In the case of BBC-S, we define this external pressure to be inversely proportional to the input threshold s; a larger threshold corresponds to a smaller external pressure, leading to larger bubbles.

We now propose an algorithmic enhancement to BBC-S that we call *Pressurization* that is designed to improve upon the quality of the local minima discovered. We start the first iteration of BBC-S with a small enough pressure to cause all points to be assigned to some cluster, and slowly increase the pressure after each iteration. An additional parameter $\gamma \in [0, 1)$ that controls the rate of pressure increase is used as an exponential decay parameter, and $s_j = s + \lfloor (n - s)\gamma^{j-1} \rfloor$ is used instead of s for the j^{th} iteration. Convergence is tested only after $(n - s)\gamma^{j-1} < 1$. A slower but more robust alternative involves running BBC-S to full convergence after each recomputation of s, but in practice only yields slightly better or similar quality results. Pressurization can similarly be implemented for the alternate formulation BBC-Q, by varying the fixed cost q_{max}.

6.3 Soft BBC-Press

The Pressurization scheme can also be extended to Soft BBC for Case B when α_0 is not updated. When α_0 and p_0 are large (close to 1), only a small amount of data is "explained" by the k exponential mixtures. This may lead to bad local minima problems similar to (although less severe than) the one faced in BBC. Therefore, we propose a soft version of Pressurization that takes a decay parameter $\tau \in [0, 1)$ and runs Soft BBC (Case B) multiple times as follows: (1) start with some initial model parameters $\{\theta_j^1\}_{j=1}^k$ and run Soft BBC to convergence. (2) at trial r set α_0 to $\alpha_r = \alpha_0(1 - \tau^{r-1})$, and for $r > 1$ set current model parameters to the output of last trial: $\{\theta_j^r\}_{j=1}^k = \{\theta_j^{r-1}\}_{j=1}^k$. Repeat step (2) until $\alpha_r - \alpha_0$ is smaller than ϵ, and then perform a final run with $\alpha_r = \alpha_0$.

6.4 Pressurization vs. Deterministic Annealing

Although our concept of Pressurization conceptually resembles the approach of deterministic annealing [UN98], they are not the same. For example, deterministic annealing in a Gaussian mixture modeling setting would involve gradually reducing the variance term σ^2 (Equation 19), whereas Soft Pressurization involves gradually increasing the probability mass α_0 (Equation 5) of the uniform background distribution. For the Gaussian setting where the variance is an updatable parameter (Section 8.2), it may even be possible to combine deterministic annealing with Pressurization. However, it is not clear if the concepts of deterministic annealing and Pressurization share any common theoretical foundation, for exponential distributions in general, or even just for the Gaussian distribution; it would be an interesting problem to explore such a connection. A notable property of Pressurization is that it works on both hard and Soft BBC, whereas deterministic annealing is only applicable in a soft setting. This has great significance for practical applications for large high dimensional datasets; a deterministic annealing approach for improving local search would require us to use Soft BBC which contains exponential mixtures, and exponential mixtures are generally hard to compute on high-dimensional datasets because of rounding errors.

7 A unified framework

7.1 Unifying Soft BBC & Bregman Soft Clustering

At first glance, the **E** and **M** steps of Soft BBC for case (A) and (B) resemble those of Bregman Soft Clustering [BMDG05]; the **M** step in (A) is identical to the **M** step of Bregman Soft Clustering, while that of (B) has an extra rescaling (Equation 14). The **E** step of Soft BBC has an extra update equation for the uniform background distribution (Equation 9). More specifically, we can show that:

Proposition 7.1. *Soft BBC*[8] *becomes identical to Bregman Soft Clustering when* $p_0 = 0$.

Proof. When p_0 is set to 0, we note that: (1) Equation 9 becomes inconsequential making the **E** step of Soft BBC identical to that of Bregman Soft Clustering, (2) the **M** step of case (B) for Soft BBC becomes the same as the **M** step of (A) since the denominator becomes n, and the **M** step of (A) is identical to that of Bregman Soft Clustering, and (3) the contribution of the uniform distribution to the objective function(Equations 6 and 7) vanishes making it identical to the objective function for Bregman Soft Clustering. □

7.2 Unifying Soft Bregman Bubble & Bregman Bubble Clustering

We are now ready to look at how the generative model Soft BBC relates to the BBC problem, specifically the formulation where the number of points classified into the k real clusters (excluding the "don't-care" cluster) is fixed (**Definition 3**, Section 4.1), and show the following:

Proposition 7.2. *BBC optimizes a lower bound on the log-likelihood objective function of Soft BBC.*

Proof. Let us consider the cost function:

$$L_2(\Theta|\mathcal{X}) = \sum_{i=1}^{n} E_{p^\dagger(Y_i=j|\mathbf{x}_i,\Theta)}[\log p(\mathbf{x}_i, Y_i = j|\theta_j)] \tag{16}$$

where $p^\dagger(Y_i = j|\mathbf{x}_i, \Theta) = 1$ for $j = \underset{0\leq j\leq k}{\operatorname{argmax}}\, p(\mathbf{x}_i, Y_i = j|\theta_j)$ and 0 otherwise, which is essentially equivalent to the posterior class probabilities based on the hard assignments used in BBC. It can be shown [KMN97] that for a fixed set of mixture parameters $\Theta = \{\theta\}_{j=1}^{k}$, and $L(\Theta|\mathcal{X})$ being the log-likelihood objective of Soft BBC(Equation 6):

$$L_2(\Theta|\mathcal{X}) \leq L(\Theta|\mathcal{X}) \tag{17}$$

This result is independent of the choice of priors $\{\alpha_j\}_{j=0}^{k}$. Note that while $L(\cdot)$ depends upon the priors while $L_2(\cdot)$ does not. For our choice of mixture components, based on Equations 4 and 17, one can readily obtain the following form for $L_2(\cdot)$:

[8] For both cases A and B.

$$L_2(\Theta|\mathcal{X}) = \sum_{j=1}^{k} \sum_{\forall Y_i = j}^{k} \log p^{\phi}(\mathbf{x}_i) - \tag{18}$$
$$\beta D_{\phi}(\mathbf{x}_i, \theta_j) + \sum_{\forall Y_i = 0} \log(p_0)[i]_{i=1}^{n}$$

If the number of points assigned to the uniform distribution is fixed to $n - s$, s points are assigned to the k exponential distributions, and p_0 and β are fixed, we can see from Equation 18 that:

Proposition 7.3. *Maximizing $L_2(\Theta|\mathcal{X})$ is identical to minimizing the BBC objective function Q_b (Equation 3).*

From Proposition 7.3 and Equation 17 we have the proof for Proposition 7.2. □

Proposition 7.4. *BBC with a fixed s as input (Definition 3, Section 4.1) is a special case of Soft BBC with fixed α_0.*

Proof. Let us consider an extreme case when $\beta \to \infty$ for Soft BBC (see Equation 6 and 4). Then the class posterior probabilities in Soft BBC converge to hard assignment (BBC) ensuring that $L(\Theta|\mathcal{X}) = L_2(\Theta|\mathcal{X})$ in Equation 18. Since BBC is equivalent to optimizing $L_2(\Theta|\mathcal{X})$ (Proposition 7.3), we can also view BBC with fixed s (**Definition 3**) as input as a special case of Soft BBC with fixed α_0. □

7.3 Unification of Bregman Bubble Clustering with BBOCC & Bregman Hard Clustering

For $k = 1$, the cost function Q_b (Equation 3) used in Bregman Bubble clustering becomes identical to Q_{one} (Equation 2). Furthermore, for $k = 1$, BBC-S and BBC-Q become identical to BBOCC-S and BBOCC-Q respectively.

Proposition 7.5. *Bregman Bubble Clustering becomes Batch Ball One Class Clustering when k=1*

The defining characteristic of BBC is its ability to find small, dense regions by modeling a small subset of the data. However, when we force BBC to cluster all the data points by setting $s = n$ in BBC-S, or $q_{max} = \infty$ in BBC-Q, Stage 2 of BBC becomes inconsequential, and it degenerates into the Bregman Hard Clustering algorithm, the generalization of K-Means to Bregman divergences:

Proposition 7.6. *Bregman Bubble Clustering becomes Bregman Hard Clustering [BMDG05] when $q_{max} = \infty$ (for BBC-Q) or when $s = n$ (for BBC-S).*

Proposition 7.7. *BBC is a special case of BBC-Press when $\gamma = 0$.*

When $\gamma = 0$, BBC-Press reduces to BBC as the modifications to the input q_{max} and s become inconsequential. Furthermore, if we think of BBC as a search under "constant pressure", then for Bregman Hard Clustering this pressure is zero.

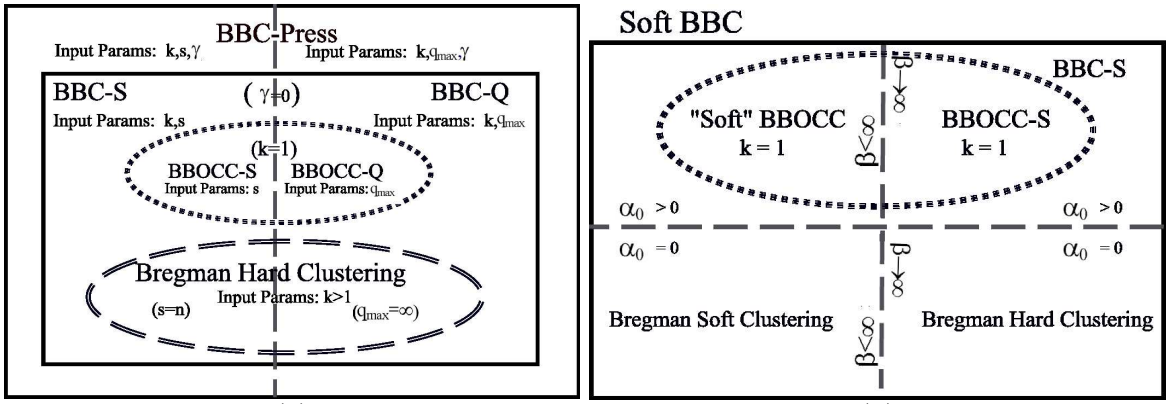


Figure 2: Unification of various algorithms for a given Bregman divergence D_ϕ: (a) BBC, BBOCC and Bregman Hard Clustering are all special cases of BBC-Press and are obtained by fixing specific input parameters. (b) Bregman Hard and Soft Clustering, BBC-S, BBOCC-S and a "soft" BBOCC consisting of a mixture of one exponential and a uniform background are all special cases of Soft BBC and are obtained from specific combinations of (i) whether $\beta \to \infty$ or not, (ii) whether the background distributions weight α_0 is 0 or not, and (iii) whether k is 1 or not. Note that Bregman Clustering (both hard and soft) with $k = 1$ does not result in a useful algorithm.

Figure 2 summarizes the hierarchy of algorithms descending from BBC-Press. It is interesting to note that for $k = 1$, Bregman Clustering is not very meaningful [9], whereas BBC gives rise to BBOCC. BBC can therefore be thought of as a conceptual bridge between the problem of one class clustering and exhaustive k class clustering, especially in the context of finding dense regions in the data. It combines the salient characteristics of both Bregman Hard Clustering and BBOCC resulting in a more powerful algorithm, and these combinations work across all Bregman divergences. Furthermore, the connections described above show that the generative models behind BBOCC-S and BBC-S are essentially the same; the difference is only in having a single vs. multiple exponential distributions mixed with a uniform background. This result also highlights the fact that BBC-S as an extension of BBOCC-S follows directly from the underlying generative model and is not just a heuristic.

8 Example: Bregman Bubble Clustering with Gaussians

Now that we have developed the theoretical framework for Soft BBC to work with all regular exponential distributions, we describe a concrete example with the Gaussian distribution, which is popularly used for many real-life applications. Let us consider spherical d-dimensional Gaussian distributions of the form:

$$N(\mathbf{x}|\mathbf{a},\sigma) = \frac{1}{\sqrt{2\pi\sigma^2)^d}} exp(-\frac{||\mathbf{x}-\mathbf{a}||^2}{2\sigma^2}) \tag{19}$$

[9] For $k = 1$, Bregman Soft Clustering returns a single exponential distribution fit to the whole data while Bregman Hard Clustering simply returns the mean of the whole data.

where $\mathbf{a} \in \mathbb{R}^d$ is the the mean, and $\sigma^2 \in \mathbb{R}$ is the variance that is the same across all the d dimensions. There are two major variations of the Soft BBC algorithm depending upon how we treat the variance σ^2:

8.1 σ^2 is fixed

Soft BBC: The parameters $\{\theta_j\}_{j=1}^k$ in equation 5 correspond to the parameters of the k exponential distributions of the mixture model that are updated in the **M** step of Soft BBC (Algorithm 4). For the Gaussian example, if we fix the values of $\{\sigma_j^2\}_{j=1}^k$ for the k spherical Gaussians mixtures then only parameter that can be updated are the k Gaussian means $\{\mathbf{a}_j\}_{j=1}^k$. For the Soft BBC algorithm this corresponds to $\theta = a$ and the sufficient statistics $\mathbf{x}_s$ is simply $\mathbf{x}$. $\frac{1}{2\sigma^2}$ is the scaling parameter β (equation 4) for the exponential function p^ϕ, $D_\phi(\mathbf{x}, \mathbf{a}) = ||\mathbf{x} - \mathbf{a}||^2$ corresponds to the Squared Euclidean distance of $\mathbf{x}$ from Gaussian mean $\mathbf{a}$, and $f_\phi(\mathbf{x}) = \frac{1}{\sqrt{(2\pi\sigma^2)^d}}$.

Therefore, the **E** step of Algorithm 4 involves computing p^ϕ as $N(\mathbf{x}|\mathbf{a}, \sigma)$ using the current Gaussian means $\{\mathbf{a}_j\}_{j=1}^k$ and the fixed variances $\{\sigma_j^2\}_{j=1}^k$ and then performing the rescaling given by equation (8) and (9) to get $p(Y_i = j|\mathbf{x}_i)$ for all the n points. For the **M** step, the new priors $\alpha_{j=1}^k$ can now be re-estimated using either equation 11 or 14 depending upon whether we keep α_0 fixed or not (case A vs. B). The θ_j for j^{th} exponential component represented by the Gaussian mean $\mathbf{a}_j$ can then be re-estimated as the weighted average of $\mathbf{x}$ as described by equation 13.

BBC-S: The proof for Proposition 7.4 tells us that the BBC-S algorithm will fall out from the Soft BBC when $\beta \to \infty$. For our Gaussian model with fixed variance, since $\beta = \frac{1}{2\sigma^2}$, this corresponds to setting the variances $\{\sigma_j^2\}_{j=1}^k \to 0$. This results in BBC-S, Definition 1 (Section 4.1) using Squared Euclidean distance as the Bregman divergences. Furthermore, when we set $s = n$, this version of BBC-S also gives us the classical K-Means algorithm, or Bregman Hard Clustering with Squared Euclidean distance as D_ϕ. An example of output from such a BBC-S variant is shown in Figure 3 (d).

8.2 σ^2 is optimized

Soft BBC: If the variances $\{\sigma_j^2\}_{j=1}^k$ are also updated as a part of the EM, then both $\{\mathbf{a_j}\}_{j=1}^k$ and $\{\sigma_j^2\}_{j=1}^k$ get updated in the **M** step of Soft BBC (Algorithm 4) and the sufficient statistics $\mathbf{x}_s$ becomes $[\mathbf{x}, \mathbf{x}^2]^T$. The **E** step of Algorithm 4 still involves computing p^ϕ as $N(\mathbf{x}|\mathbf{a}, \sigma)$ using the current Gaussian means $\{\mathbf{a}_j\}_{j=1}^k$ and the current variances $\{\sigma_j^2\}_{j=1}^k$ and then performing the rescaling given by equation (8) and (9) to get $p(Y_i = j|\mathbf{x}_i)$ for all the n points. For the **M** step, the new priors can also be re-estimated as before using either equation 11 or 14 depending upon whether we keep α_0 fixed or not. However, the θ_j for j^{th} exponential component, now a function of both the Gaussian mean $\mathbf{a}_j$ and the variance σ_j^2, needs to be re-estimated as the weighted average over the sufficient statistics. It can be shown that this maps to: (1) re-estimating the mean $\mathbf{a}_j$ as the average of $\mathbf{x}$ over the n points weighted by $p(Y_i = j|\mathbf{x}_i)$, and (2) re-estimating the variance as a weighted average of $(\mathbf{x} - \mathbf{a}_j)^2$ over the n points also weighted by $p(Y_i = j|\mathbf{x}_i)$. An example of output from such a Soft BBC variant is shown in Figure 3 (b).

BBC-S: Unlike for the fixed variance case, the scaling parameter β cannot be thought of as a function of variance since σ^2 is a part of the updatable parameters. The corresponding D_ϕ,

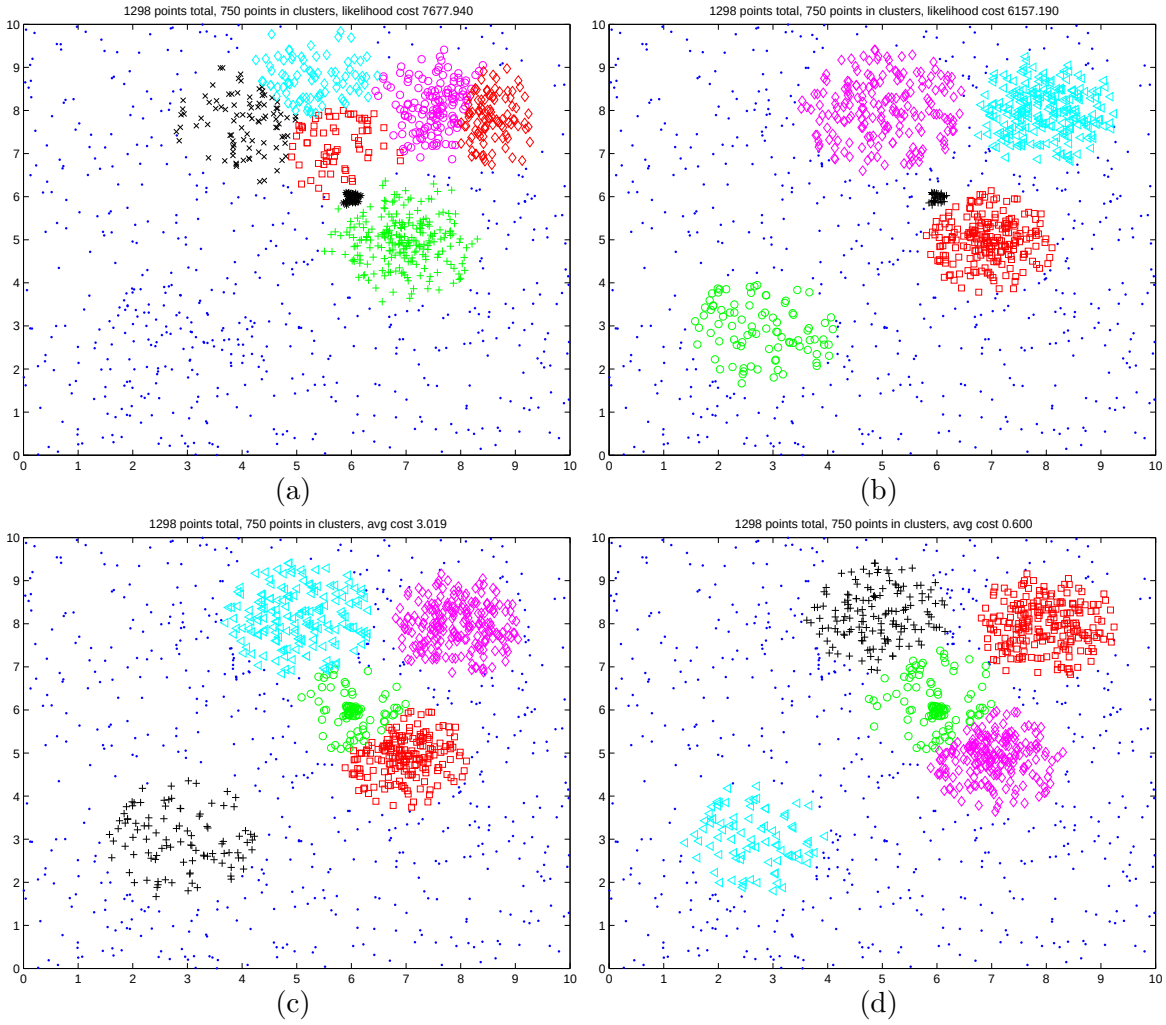


Figure 3: Comparison of bubbles generated using three variants of BBC on the simulated 2-D dataset: (a) Soft BBC with updated σ^2 for $k = 7$ highlights the nonlinear nature of boundaries where the bubbles touch each other. (b) Same as (a) except with $k = 5$, (c) BBC-S corresponding to Soft BBC in (b), where the distances to clusters get scaled in proportion to cluster variance, (d) BBC-S with fixed σ^2. The boundaries of bubbles where they touch each other (not shown) for cases (c) and (d) are linear. Note: Each point was assigned at convergence to the cluster to which it had the largest soft assignment value.

(which is not the Squared Euclidean distance) can be derived from the relationship defined by equation 4 and corresponds to Mahalanobis distance in the original space of $\mathbf{x}$. A corresponding BBC-S algorithm obtained when $\beta \to \infty$ is different from the BBC-S algorithm described for the scenario where σ^2 was fixed. One property of such a generative model is that in the original space of $\mathbf{x}$, bubbles of varying diameters can be discovered by both the Soft BBC and the corresponding BBC-S algorithm, which could be suitable for domains where the natural clusters have very different diameters. It can be shown that in each iteration of such an implementation, the estimated distances to clusters need to be rescaled in proportion of the variances of the respective clusters in that iteration. An example of output from such a BBC-S variant is shown in Figure 3 (c).

8.3 "Flavors" of Soft BBC for Gaussians

For Soft BBC built using spherical Gaussians, there are eight possible flavors (Table 1) depending upon whether (1) α_0 is updated (Case A vs. B, Section 5.4), (2) the Gaussian mixture variances are updated (Section 8.1 vs. 8.2), or (3) all cluster variances are forced to be equal. For the cases where variance could be updated, forcing them to be equal requires computing a weighted average of the variances of the k Gaussians after updating the variances in the **M** step as described in Section 8.2, and then assigning this weighted average to the variances of all the k Gaussians.

Table 1: 8 flavors of Soft BBC for spherical Gaussians arise depending upon the choice of Θ

Flavor	Update σ	$\{\sigma_j\}_{j=1}^{k} = \sigma_1$	Fixed α_0
1	No	No	No
2	No	No	Yes
3	No	Yes	No
4	No	Yes	Yes
5	Yes	No	No
6	Yes	No	Yes
7	Yes	Yes	No
8	Yes	Yes	Yes

8.4 Illustration on a simulated dataset

Figure 3 shows a comparison of bubbles generated using three of the many variants that arise out of BBC depending upon whether σ^2 is updated or not, and whether we use Soft BBC or the corresponding hard algorithm BBC-S. The results are shown for the simulated 2-D dataset (Sim-2 dataset, Table 2, Section 11) that has points generated from 5 Gaussians of variances varying from small to large and a uniform background. The results are summarized below:

Figure 3(a) shows results using a Soft BBC variant with Pressurization when σ^2 are unequal and updated (Flavor 6, Table 1) and k=7. The k was kept large to highlight the nonlinear nature of the boundaries between touching bubbles when σ^2 is an updatable parameter. We can also see an interesting scenario where a smaller Bregman bubble is completely engulfed by a larger bubble.

Figure 3(b) shows results using a flavor of Soft BBC with Pressurization that is the as that in **Figure 3(a)** but with $k = 5$. The algorithm finds the five densest Gaussians and fits them very well, especially the very dense but small cluster that corresponds to a class containing only 30 out of 1,298 data points.

Figure 3(c) shows results using BBC-S with Pressurization when σ^2 is updated with $k = 5$ and corresponds to the hard version of the soft model in Figure 3(b). We can see that although such a hard model can give clusters with varying diameters, it is not as sensitive as the soft algorithm. However, the hard algorithm scales well on very high dimensional datasets, which is problematic for Soft BBC and exponential mixtures in general due to rounding errors (discussed in more detail in the experiments section).

Figure 3(d) shows results using the simplest BBC algorithm with Pressurization; BBC-S without σ^2 update, when k was set to 5. Such a model falls out of a Soft BBC comprised of a mixture of k Gaussians with equal and fixed spherical variances and a uniform background (Flavor 4, Table 1).

8.5 Mixture-6: An alternative to BBC using a Gaussian background

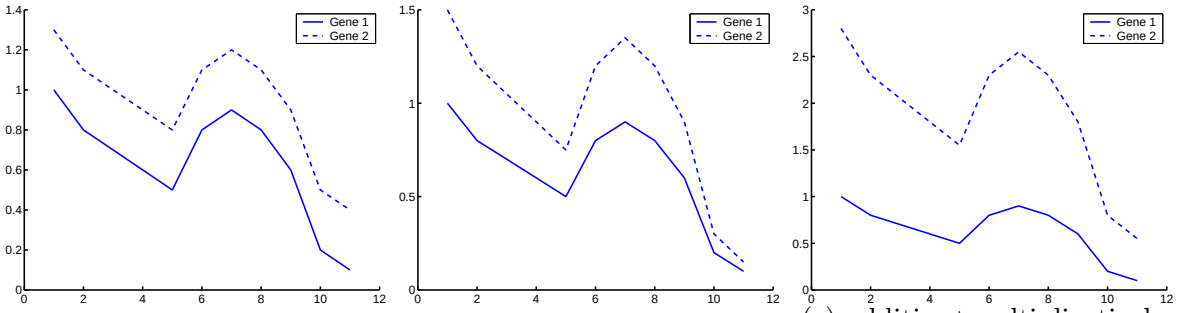
For the Gaussian case where σ^2 is optimized, we could also define an alternative mixture model where the uniform background distribution is replaced by a background Gaussian distribution with a large variance[10]. This results in a mixture of Gaussians model with $k + 1$ Gaussians where the 0^{th} Gaussian has a fixed large variance and only its mean is updated, while for all other Gaussians both the mean and the variance are updated. Such a model can be viewed as a "hybrid" of the models in Section 8.1 and 8.2 with the following update steps: (1) the E step involves computing p^{ϕ} as $N(\mathbf{x}|\mathbf{a}, \sigma)$ using the current means and variances for Gaussians 1 to k and using the current mean and the fixed variance for Gaussian 0, and (2) For the **M** step, the new priors can be estimated as before using either equation 11 or 14 depending upon whether we keep α_0 fixed or not. The variance of the Gaussian 0 is not updated, while that of 1 to k Gaussians can be re-estimated using the procedure as described in Section 8.2. The convergence guarantees for these steps follow directly from Bregman Soft Clustering since we only use a mixture of Gaussians.

We call this model *Mixture-6* since it is analogous to the flavor 6 of Soft BBC (Table 1). Unlike in the Soft BBC where the background mass (and the corresponding fraction of data assigned to the background after converting to hard assignment at convergence) is easy to control and predict (Section 5.5), using a large variance background does not result in a stable background; the final background mass varies substantially depending upon where the center of the background Gaussian lies at convergence. Furthermore, it is not clear if (1) this model could be generalize to all regular Bregman divergences, and (2) if a corresponding generalized hard model exists. Mixture-6 mainly serves as another baseline for empirically evaluating Soft BBC.

9 Extending BBOCC & BBC to Pearson Distance

In biological organisms, genes involved in the same biological processes are often correlated in an additive or multiplicative manner, or both (Figure 4). *Pearson Correlation* captures

[10] Much larger than the cluster variances.



(a) additively correlated (b) multiplicatively correlated (c) additive+multiplicatively correlated

Figure 4: Three common types of correlations observed between expression levels of genes. The x-axis represents distinct measurements at different time points/across experiments, while the y-axis represents the expression level of the gene.

the similarity between two variables in $\mathbb{R}^d$ that is invariant to linear scaling, such as a multiplicative and/or additive offset, and is therefore a popular similarity measure for clustering gene-expression and other biological data [SS00, MTea04].

For two data points $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$, the Pearson correlation P can be computed as $P(\mathbf{x}, \mathbf{y}) = \frac{zscore(\mathbf{x}) \bullet zscore(\mathbf{y})}{d-1}$, where $zscore(\mathbf{x})$ represents the vector-based z-scoring of the data point vector $\mathbf{x}$, i.e. we z-score each of the data points separately across features values [11], and is equal to $\frac{\mathbf{x}-\mu(\mathbf{x})}{\sigma(\mathbf{x})}$, where $\mu(\mathbf{x})$ is the mean of the elements of the vector $\mathbf{x}$ and $\sigma(\mathbf{x})$ is the standard deviation. We define the *Pearson Distance* as $D_P = 1 - P$. Since $P \mapsto [-1, 1]$, therefore $D_P \mapsto [0, 2]$. It can be shown that Pearson Distance is equal to the Squared Euclidean distance between z-scored points normalized by $2(d-1)$:

$$D_P(\mathbf{x}, \mathbf{y}) = \frac{\parallel zscore(\mathbf{x}) - zscore(\mathbf{y}) \parallel^2}{2(d-1)} \tag{20}$$

D_P can also be viewed as the Squared Euclidean distance between points that have been (1) rotated by subtracting the mean, and (2) by variance normalization, have been projected onto a hypersphere of radius 1 (radius $\frac{1}{\sqrt{2}}$ in Euclidean space) centered at the origin. When D_ϕ is replaced by D_P in Equation 3, we refer to Q_b as *Average Pearson Distance*(APD).

Proposition 9.1. *For any cluster $\mathcal{C}_j$ in $\mathcal{G}$, the cluster representative $\mathbf{c_j}^*$ that minimizes contribution to APD by that cluster is equal to the mean vector of the points in $\mathcal{C}_j$ projected onto a sphere of unit radius, i.e. $\mathbf{c_j}^* = \underset{\mathbf{c_j}}{argmin}(APD(\mathcal{C}_j, \mathbf{c_j})) = \frac{\mathcal{C}_j^m}{\|\mathcal{C}_j^m\|}$, where $\mathcal{C}_j^m = \frac{1}{|\mathcal{C}_j|} \sum_{i:\mathbf{x}_i \in \mathcal{C}_j}^{|\mathcal{C}_i|} zscore(\mathbf{x}_i)$.*

The proof for the above proposition follows directly from the result used by [DM01] for updating the center for their *Spherical K-Means* algorithm, which is a K-Means type of algorithm that uses Cosine Similarity as the similarity measure. Pearson Distance and Cosine Similarity are closely related; if we estimate the Squared Euclidean distance between points normalized

[11] This is different from the way z-scoring is often used in statistics, where it is performed for each column/dimension. We are performing it across rows of a data matrix, if the rows were to represent the data points.

by their L2-norm ($\sqrt{\sum_{i=1}^{d} \mathbf{x}_i^2}$) instead of between z-scored points, we obtain $2\times$(1−Cosine Similarity). Note that (1−Cosine Similarity) of z-scored points is the same as Pearson Distance. Because of Proposition 9.1, for $D = D_P$ the optimal representative computation in BBC involves the averaging of the z-scored points rather than the original points, and then re-projecting of the mean onto the sphere. This minor modification to BBC allows it work with D_P and ensures convergence to a local minima [12]. Since BBOCC is a special case of BBC, the same modification works for BBOCC too for the problem of One Class Clustering.

Note: Because of the relationship between Cosine Similarity and Pearson Distance, BBC will also work with Cosine Similarity, which is a popular similarity measure for clustering textual data [DM01]. Note that for $s = n$, BBC with Cosine Similarity degenerates to Spherical K-Means. For running BBC with Pearson Distance, since the z-scored data points have zero mean across the d dimensions, the mean of the z-scored points $\mathcal{C}_i^m$ used for center update only needs to be normalized by its l2-norm to obtain $zscore(\mathcal{C}_i^m)$. For this reason, if we z-score the individual data points in advance and run BBC using Cosine Similarity, it produces the same results as running BBC with Pearson Distance. However, it is important to note that the effect of not subtracting the mean gives rise to different distance measures (Pearson Distance vs. (1−Cosine Similarity)) and can result in very different clusterings; points with additive offsets will not necessarily be close when using only the Cosine Similarity. This difference could be important depending upon the application. For example, Pearson Distance/Correlation is more suitable for gene-expression data (Figure 4), while Cosine Similarity works better for document clustering. A similar distinction should be kept in mind about Bregman divergences in general; although BBC works with all Bregman divergences, it would produce radically different results depending upon the choice of the divergence; the particular problem domain and the underlying exponential distribution (Equation 4 and Proposition 7.4) should guide the selection of the appropriate Bregman divergence for BBC.

10 Global Search, Model Selection & Seeding

We now present an alternative to Pressurization for alleviating the problem of local minima in our local search. For $k = 1$, in Section 10.1 we present a deterministic global search algorithm called HOCC that gives strong optimality guarantee for cost Q_{one}. The output of HOCC can then be used to seed BBOCC resulting in a "hybrid" search algorithm (Section 10.2) that gives better results than either HOCC or BBOCC, and inherits the performance guarantee and determinism. Then, in Section 10.3, for $k > 1$, we present another deterministic clustering algorithm called DGRADE that is motivated by HOCC, but that can automatically determine the number of distinct dense regions in the data, the location of these dense regions, and their representative centroids in $\mathcal{X}$. These k representatives can then be used to seed BBC, thus alleviating the problem of local minima and resulting in deterministic results. By automatically determining k, DGRADE also addresses the issue of determining the number of clusters in the data.

[12] and the corresponding local maxima for average Pearson Correlation.

Algorithm 5 HOCC

1: **Input:** Distance matrix $\mathbf{M}$ containing distance between all points, desired list of cluster sizes s_{list}.
2: **Output:** Pairs of solution set containing indexes of best centroid followed by member points indices for each cluster size specified in s_{list}.
3: $[\mathbf{radM}, \mathbf{idxM}] = sortrows(\mathbf{M})$
4: **for** $j = 1$ to $|s_{list}|$ **do**
5: $\quad s = s_{list}(j)$
6: $\quad bestIdx = min(Q_{one}(\{radM(i,s)\}_{i=1}^{n}))$
7: $\quad solution(j) = \{bestIdx, \{idxM(bestIdx, j)\}_{j=1}^{s}\}$
8: **end for**

10.1 Global Search using One Class Enumeration: HOCC

For the problem of finding a single Bregmanian ball as described by Definitions 1 and 2 in Section 3 using the cost function defined by Equation 2, BBOCC-S provides a local minima solution, but does not provide any guarantee on the quality of the local minima. A local search *can* get stuck in a bad local minima when the ball to be discovered is in a small region isolated and far from the initial centroid input to BBOCC-S. One could improve upon the local search by picking the best solution out of multiple trials of BBOCC-S with random seeding or use Pressurization with BBOCC-S (by running BBC-Press with $k = 1$). However, neither approach or their combination provides any performance guarantees. Is there an alternative? It turns out that at least for the problem of One-Class clustering with a Bregmanian ball, there is. The key idea is to first restrict the cluster representative to be one of the data points, i.e. $\mathbf{c} \in \mathcal{X}$, and then performing an exhaustive (global) search on this reduced problem. For the restricted problem, the following holds true for cost Q_{one}:

Proposition 10.1. *If* $\mathbf{c}$ *is restricted to one of the sample data points* $\mathcal{X}$*, then the number of distinct* [13] *clusters of size* 2 *through* n *is* $n(n-1)$*, and can be enumerated.*

Proposition 10.1 is the basis of *Hypersphere One Class Clustering* (HOCC, Algorithm 5). It follows from Proposition 3.3 and the observation that the cluster representative and the farthest point determine members of $\mathcal{G}$ and that such a tuple can only be picked from $\mathcal{X}$. Such clusters could be visualized as bounded by "hyperspheres" in the corresponding divergence space. HOCC enumerates all such spheres and finds the one with the lowest cost for the specified size s. To summarize Algorithm 5: the s^{th} sorted index of row i represents the farthest point in a cluster of size s with center as the i^{th} point (line 3). The cumulative summation of each row j of **radM** followed by division by the column index s gives the cost Q_{one} of the Bregmanian ball of size s centered at the i^{th} data-point. The optimal solution is then picked as the one with the smallest cost (line 6 and 7) from the n solutions of size s centered on each of the n data points .

Note: It is also possible to use HOCC to find the best cluster if the cost is defined as the *maximum* divergence rather than the average divergence value used in Q_{one}. This can be achieved by simply omitting the cost computation process in Algorithm 5 and using the sorted distances (line 3) directly as the costs, and a variation of HOCC and BBOCC based on this approach is discussed in detail in [GG05]. Our empirical evaluations in that paper showed that the maximum distance formulation results in similar clusters on high-dimensional datasets.

[13] If a cluster is uniquely defined by G and $\mathbf{c}$, the cost changes even if G remains the same but $\mathbf{c}$ changes.

Complexity: Using HOCC, we can enumerate the lowest cost clusters of *all* possible sizes in $O(n^2 \log(n))$ time. However, for seeding BBC-S, since s is known, using heap sort the complexity reduces to $O(ns \log(n))$. The corresponding space complexity is only $O(n)$, since n Bregmanian ball costs need to be saved before picking the best solution. Taking the initial pairwise distance computation into account the total complexity of HOCC with a single s as input is $O(max(ns \log(n), n^2))$.

Proposition 10.2. Optimality bound for Q_{one} using HOCC: *For a Bregman Divergence D_ϕ where $l \leq \phi'' \leq u$, the average distance cost (Q_{one}) of the cluster found by HOCC algorithm is within $1 + \frac{u}{l}$ times the optimal solution.*

Proof. We can prove the above using the following Lemma:

Lemma 10.3. [BMDG05] *For all Bregman Divergences D_ϕ, and points $\{\mathbf{p}_i\}_{i=1}^s$, $\mathbf{w}$, $\mathbf{c}^* \in \mathbb{R}^d$, where $\mathbf{c}^*$ is the minimizer for Q_{one}, the following is true:*

$$\frac{1}{s}\sum_{i=1}^{s} D_\phi(\mathbf{p}_i, \mathbf{c}^*) + D_\phi(\mathbf{c}^*, \mathbf{w}) = \frac{1}{s}\sum_{i=1}^{s} D_\phi(\mathbf{p}_i, \mathbf{w}) \tag{21}$$

For a cluster of a given size, if Q^*_{one} is the cost of the optimal solution, and $Q^*_{one-HOCC}$ is the lowest cost that appears in the HOCC algorithm when $w = w^*$ is the representative data point that gives the lowest possible cost, then we can rewrite Equation 21 as:

$$Q^*_{one} + D_\phi(\mathbf{c}^*, \mathbf{w}^*) = Q^*_{one-HOCC} \tag{22}$$

For a Bregman Divergence D_ϕ where $l \leq \phi$" $\leq u$, it is also possible to show that:

$$D(c^*, w^*) \leq \frac{u}{l} \times \frac{1}{s}\sum_{i=1}^{s} D_\phi(\mathbf{p}_i, \mathbf{c}^*) = Q^*_{one} \tag{23}$$

Substituting inequality from 23 into 22 we get:

$$(1 + \frac{u}{l})Q^*_{one} \geq Q^*_{one-HOCC}$$

□

For the Squared Euclidean distance, $u = l = 2$, therefore the above optimality guarantee reduces to within 2 times of optimal. For an unbounded divergence applied to a given dataset, it will be possible to bound l and u, in which case we would get the corresponding optimality bounds.

HOCC for a specific cost q_{max}: A simple modification to the HOCC algorithm (which follows from Proposition 3.1) makes it work with a fixed cost q_{max} as input. After sorting each of the rows of the distance matrix $\mathbf{M}$ in Algorithm 5 (line 3), instead of picking the s^{th} element of a row i to compute cost, we perform a binary search on each of the sorted rows to find the largest column index j such that the value of $Q_{one}(radM(i, j)) \leq q_{max}$. The value of j then represents the size of the largest cluster at center i that has cost within the threshold q_{max}. The best cluster is then found simply by picking the points from 1 to j corresponding to the row that gives the largest j.

10.2 Seeding BBOCC with HOCC: Hyper-BB

The hybrid search algorithm Hyper Batch Ball (Hyper-BB) is a simple combination of the global approximation HOCC and the local search BBOCC. For either a fixed cluster size or radius, given a particular D_ϕ and data to cluster $\mathcal{X}$, it consists of the following two steps: (1) Find the optimal cluster representative $\mathbf{c}^*_{hocc}$ using HOCC, and (2) pass this center as seed to BBOCC: $\mathbf{c}_{in} = \mathbf{c}^*_{hocc}$.

Hyper-BB inherits the same global guarantees that HOCC provides but has the added advantage that the local search is likely to improve the results substantially. Empirical results presented later in Section 11.4 supports this intuition. The global search seeds the local search, and therefore Hyper-BB has a hybrid search bias that combines local and global search. Since HOCC is deterministic and the output of BBOCC is determined by the initial seed, the output of Hyper-BB algorithm is also deterministic.

Pressurization vs. HOCC, Speed tradeoff: When HOCC is used to seed BBOCC, it provides a strong optimality guarantee and deterministic results [14]. However, the tradeoff is computational complexity. For example, HOCC with BBOCC-S is $O(max(ns\log(n), n^2))$, which in practice runs slower than BBOCC-S with Pressurization whose iterations[15] take $O(s\log n)$ time. It is useful to have the choice of this tradeoff between optimality guarantee vs. speed for improving BBOCC local search; for very large problems, it may be more practical to run BBOC with Pressurization rather than performing a seeding using HOCC that has a time-complexity between $O(n^2)$ and $O(n^2\log n)$.

10.3 Seeding BBC and determining k using Density Gradient Enumeration (DGRADE)

The problem of finding a good seeding method for local search based on BBC for $k > 1$ is a harder problem than for the One Class case, since the search space for possible initializations gets much larger. For a given k, a simple extension of the strategy used in HOCC that involves restricting the search for cluster representatives to the given data points, and then enumerating all the $\binom{n}{k}$ combinations of centroids, is prohibitively expensive. On the other hand, picking the best solution over multiple trials of BBC-S or BBC-Press seems to give quite high quality solutions in practice, but also requires k as an input. Is there a fast, HOCC-type algorithm that can be used for seeding BBC [16] that could also be used to estimate k?

For a fixed $s = s_{one}$, if BBOCC-S is run on all the n possible seed locations, and after convergence the center is assigned to the nearest data point in $\mathcal{X}$, we would end up getting a small subset $\mathcal{C}_{dense}$ of $\mathcal{X}$ consisting of relatively dense points that would correspond to local minima where BBOCC converges for all possible starting locations. We could view all the starting points converging near the same final point in $\mathcal{C}_{dense}$ as belonging to same "basis of attraction". The size of set $\mathcal{C}_{dense}$ would give k, and the member points the centroids of the

[14] Pressurization on BBOCC would start from the mean of the data and would also be deterministic, but the same would not be true for BBC-Press.

[15] This is true for the faster version of BBC-Press with $k = 1$ (Section 6) where the cluster size is shrunk after each iteration. The average complexity of BBC-Press-Full would be $O(ts(\log(n))^2)$, where after each shrinkage BBC is run to full convergence. t represents the average number of trials to convergence for BBC.

[16] Which imposes an additional secondary requirement that just like BBC, the seeding algorithm should be meaningful for all Bregman divergences.

Algorithm 6 DGRADE

Input: Distance matrix $\mathbf{M}$ containing distance between all points, Bregmanian ball size s_{one}, number of points to be clustered $s \leq n$.

Output: k, Partitioning $\mathcal{G}^*$ containing k clusters$\{\mathbf{C_j}\}_{j=1}^{k}$, and the corresponding k cluster centroids $\{\mathbf{c_j^*}\}_{j=1}^{k}$.

$[\mathbf{radM}, \mathbf{idxM}] = sortrows(\mathbf{M})$
for $i = 1$ to n **do**
  $q_{list}(i) = Q_{one}(\{idxM(i,j)\}_{j=1}^{s_{one}}, \mathbf{x}_i)$
end for
$[val, idx] = sort(\mathbf{q}_{list})$
$k = 1; \{\mathbf{lab}\}_{i=1}^{n} = 0; \{\mathbf{head}\}_{i=1}^{n} = 0;$
$\mathbf{head}_{idx(1)} = \varnothing; \mathbf{lab}_{idx(1)} = 1$
for $i = 2$ to $s-1$ **do**
  $[hCost, hIdxIdx] = min(val(\{idxM(i,j)\}_{j=1}^{s_{one}}))$
  $hIdx = idxM(i, hIdxIdx)$
  if $hIdx == idx(i)$ **then**
    $k = k+1; lab(hIdx) = k; head(hIdx) = \varnothing;$
  else
    $lab(hIdx) = lab(idx(i)); head(hIdx) = idx(i);$
  end if
end for
Return k, $\{\mathbf{c_j^*}\}_{j=1}^{k}$ as the set of k points whose corresponding **head** pointers are $\varnothing$, the set of clusters formed by non-zero values in lab as $\mathcal{G}^*$.

k dense regions. However, for speed considerations it would be impractical to run BBOCC n times on large datasets.

For a faster solution, let us constrain the problem a bit further by restricting the centers of the Bregmanian ball movement during the local search also to one of the n data points. For a given s_{one} corresponding to s in BBOCC-S, a local search *simulation* is then possible that does not require us to run BBOCC, and a local search "simulation" can be performed to find the k densest locations in $O(ns \log n)$ time using heap sort. Algorithm 6 shows an implementation of such an algorithm called *Density Gradient Enumeration* (DGRADE) that has the following steps:

1. Input integers s_{one}, and the number of dense points s to be classified into clusters.
2. Sort each row of the distance matrix and save the corresponding sorted s_{one} nearest neighbors indices into **radM**, **idxM** (just like in HOCC).
3. Compute cost Q_{one} for each of n points as cost of a Bregmanian ball of size s_{one} centered on the point.
4. Sort the n points by increasing cost and save the cost of the first s points.
5. Set $k = 1$, labels of all points to 0. Create a pointer corresponding to each of the n points.
6. Assign cluster label 1 to the lowest cost point. Set its pointer to null.
7. Now pick the next $s-1$ points in the order of increasing cost and perform the following: for each point $\mathbf{x}$ find the lowest cost point $\mathbf{y}$ among the closest s_{one} neighbors (including

itself) of $\mathbf{x}$. If $\mathbf{y} = \mathbf{x}$, set $k = k + 1$ and set label of the point to k and set pointer of $\mathbf{x}$ to null. Else assign the label of $\mathbf{y}$ to $\mathbf{x}$ and set pointer of $\mathbf{x}$ to point to $\mathbf{y}$.

8. Return the clustering $\mathcal{G}$ consisting of the s densest points, and the k cluster centroids as the k points with pointers set to null.

At the end of the process, we get a set $\mathcal{G} \subseteq \mathcal{X}$ consisting of k clusters $\{\mathcal{C}_j\}_{j=1}^{k}$ formed by a subset of s points from $\mathcal{X}$ with the lowest cost. We also get a pointer from each of the s points leading to a point of lower cost Q_{one}. There are exactly k points from $\mathcal{G}$ that form k centroids, one for each cluster $\mathcal{C}_j$, and the centroid is the point in $\mathcal{C}_j$ with the lowest cost. The pointers from each of the members of a cluster $\mathcal{C}_j$ form a path of lower cost leading eventually to the centroid of the cluster. Figure 5 shows the output of DGRADE on the Sim-2 dataset when (b) the assignments of all the points is performed, i.e. when $s = n$, vs. (a) when only $s = 750$ points are assigned. One way to think about DGRADE is that it finds an approximation of the gradient of cost that would be observed if the BBOCC-S was executed n times with $s = s_{one}$ starting with each data point as a seeding point. DGRADE consists of two distinct phases: (a) the initial sorting of points by Bregmanian ball costs gives us the absolute measurement of cost in various regions of the data, and (b) pointing each data point to the direction of maximum decline in cost within the same Bregmanian ball in the second phase gives us an estimate of the direction of maximum cost decline. The second phase generates a global map of the distinct valleys that ultimately converge to the locally dense (restricted) centroids. Essentially, the algorithm performs a global search for local density cost estimate *and* the gradient direction, and unlike density-based clustering methods such as DBSCAN, is compatible with asymmetric Bregman divergences [17]. Interestingly, the cost function used (Q_{one}) can be replaced with another cost function such as the maximum distance cost function, but then the solution would not match well with the cost functions of BBOCC and BBC. Although DGRADE can be used as a algorithm to find dense regions, or as an exhaustive clustering algorithm (for $s = n$), our main goal in designing it was to obtain a seeding solution for BBC-S. Therefore, we simply use the centroids discovered by DGRADE to seed BBC-S.

Selecting DGRADE parameter s_{one}:

HOCC does not require any additional parameters beyond BBOCC-S. However s_{one} is a parameter for DGRADE that needs to be input by the user. s_{one} acts like a smoothing parameter; a larger value typically results in a smaller k. When s_{one} is increased, the number of clusters found drops rapidly (Figure 5(c)), and beyond a certain value of s_{one} a consecutive set of values result in the same k. This characteristic enables several alternatives for selecting s_{one} automatically. We now present three common scenarios and the corresponding solutions for them:

1. If k is known, we can find the smallest $s_{one} \geq 2$ that results in k clusters and a binary search could be performed in $O(n^2 \log(n))$ time for finding the best centroids of the k clusters using DGRADE. The clustering using this approach on the Sim-2 data is shown in Figure 5 (a) and (b).

2. If k is not known and somewhat "over-split" clusters are preferred, a user can specify a maximum *stability* integer value of m and a linear search could be performed for up

[17] Bregman divergences are generally not symmetric; Squared Euclidean is a notable exception.

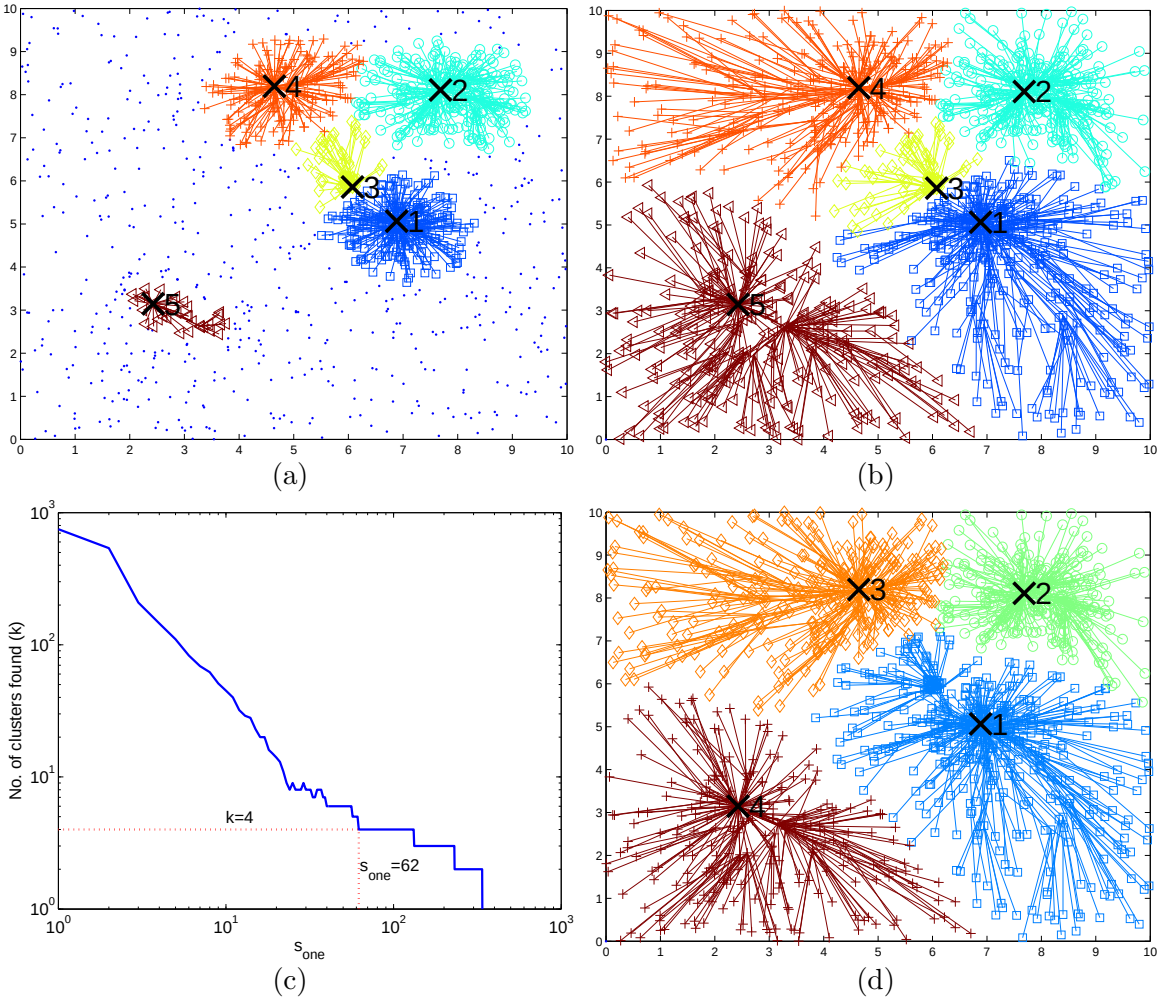


Figure 5: Clustering of Sim-2 data using DGRADE for various scenarios. For (a), (b) and (d), lines show the path of the simulated local search converging at the locally lowest cost/densest point, which also form the centroid of the corresponding cluster, and are marked as "x" and numbered 1 to 6. For (a) and (b), k is known and set to 5, and s_{one} was automatically determined as 57. (a) shows clustering for $s = 750$, while (b) for $s = n$. For unknown k, (c) shows the relationship between s_{one} and k when $s = n$ and how it can be used for choosing s_{one} and determining k automatically. The most stable k and the corresponding smallest s_{one} is shown using the dotted lines. (d) shows the four clusters discovered by DGRADE using the automatic model selection described in (c).

to a certain maximum value of s_{one} to find the value of s_{one} after which $s_{one}+1$ to $s_{one}+m-1$ values all result in k clusters. This value of s_{one} and the corresponding solution of DGRADE is then returned. This technique is more appropriate for many biological datasets where the signal-to-noise ratio is often very low, and the clustering present is extremely weak.

3. For more robust datasets, we can simply select k with the largest stability for s_{one} ranging from 1 to the smallest value that returns $k=1$. Then, we select the smallest s_{one} that gives k clusters. This selection method is shown in Figure 5 (c), where $k=4$ is obtained for $62 \leq s_{one} \leq 214$, corresponding to the largest stable interval. The smallest s_{one} for $k=4$ is 62, which is the value used for the final clustering output (for $s=n$) shown in Figure 5 (d). It is interesting to note that the densest cluster (numbered 3 in Figure 5 (b)) gets merged with a nearby larger cluster (numbered 1 in Figure 5 (b)) resulting in $k=4$, which is quite good for this completely parameterless, unsupervised setting.

Time and Space Complexity of DGRADE: The gradient traversal phase can be shown to be $O(ns)$, resulting in a total time complexity for DGRADE that is $O(max(n^2, s_{one}\log(n), ns)$, and is similar to that of the one class seeding algorithm HOCC. In practice, this results in an $O(n^2)$ algorithm, dominated by the initial distance computation. Since DGRADE performs a sequential search for discovering the cluster labels, in an efficient implementation, the indexes can be saved onto secondary storage and accessed sequentially, requiring only $O(n)$ memory storage of labels and the head pointers. Since only the graph traversal and column sorting needs to be recomputed for testing DGRADE with different values of s_{one}, the complexity of DGRADE with automatic selection of s_{one} remains reasonably low at $O(max(n^2, s_{one}\log(n), qn\log(n), nsq)$, where q is the number of different s_{one} values evaluated.

11 Experiments

11.1 Overview

We have divided our empirical evaluation into four main phases to verify various aspects of our framework:

1. **Evidence that the batch-update and fixed s approach is better for local search**: In Section 11.3, we present empirical evaluations to show the importance of the two improvements in BBOCC-S over OC-IB; using a batch update rather than a randomized update, and having a fixed size s as input rather than q_{max}.

2. **Effectiveness of hybrid search for One Class**: In Section 11.4, we present results on the one class Bregmanian ball problem and show that both the local and the global search components are important. We give comparison with the globally optimal solution using the lower bound defined by Proposition 10.2, i.e. with half of the cost of the HOCC solution. We show that the hybrid algorithm, Hyper-BB, which combines HOCC and BBOCC, gives solutions very close to this optimal lower bound.

3. **Effectiveness of BBC with Pressurization**: In Section 11.5 we describe the experimental setup, and in Section 11.6 the corresponding results for BBC-S, showing the

effectiveness of BBC with Pressurization in finding high-quality, robust results as compared with three other methods, against three real and three synthetic datasets.

4. **Effectiveness of BBC seeded with DGRADE**: In Section 11.5 we describe the experimental setup, and in Section 11.7 the corresponding results when using DGRADE to seed BBC. Although on some datasets DGRADE gave good results by itself, more consistently, seeding BBC with the centers found by DGRADE gave results that were significantly better than using either BBC or DGRADE separately, and generally better than even BBC with Pressurization. We show that it is also possible to combine all three: BBC, Pressurization and DGRADE to achieve the best quality of clustering. Our results also confirm that in practice, DGRADE can estimate k automatically, and the deterministic, high quality results using BBC seeded with DGRADE is a good alternative to Pressurization for biological datasets.

11.2 Datasets

We tested our algorithms on multiple real and synthetic datasets that are summarized in Table 2.

Table 2: A summary of the datasets used. D is the distance function used for clustering while $\mathcal{C}$ represents the number of classes for labeled datasets only.

Dataset	Source	n	d	D	k [a]	$\mathcal{C}$
Gasch Gene	Microarray	6,151	173	D_P	10	NA
Gasch Array	Microarray	173	6,151	D_P	12	12
Alizadeh	Microarray	4,026	40	D_P	1	NA
Lee	Microarray	5,612	591	D_P	9	NA
Cleaned 20-NG	Web documents	19,975	4,292	(1−Cosine Sim.)	6	6
Sim-2	Synthetic	1,298	2	Sq. Euclidean	5	5
Sim-10	Synthetic	2,600	10	Sq. Euclidean	5	5
Sim-40	Synthetic	1,298	40	Sq. Euclidean	5	5
Hard	Synthetic	4,026	40	Sq. Euclidean	1	1
Easy	Synthetic	4,026	40	Sq. Euclidean	1	1

[a] k represents the number of clusters specified to the clustering methods that require it as an input. Only used when testing BBC without seeding with DGRADE. When seeding with DGRADE, k output by DGRADE was used for all methods that required it as an input.

Real Data: The Alizadeh data consists of expression profile of 4,026 genes from 46 B-Cells Lymphoma cancer tissues, and also comes with the survival history for 40 patients, with 18 surviving long-term and 22 dying much earlier. The Gasch dataset [Gas00] consists of 6,151 genes of yeast *Saccharomyces cervisiae* responding to diverse environmental conditions over 173 microarray experiments. These experiments were designed to measure the response of the yeast strain over various forms of stress such as temperature shock, osmotic shock, starvation and exposure to various toxins. Each experiment is labeled with one of the 12 different categories of experiments. The Alizadeh data was also used by [CC04], and the Gasch data is a widely used benchmark for testing algorithms designed for clustering microarray data. Gasch Array vs. Gasch Gene shown in Table 2 are actually the same dataset but with the

dimension of clustering flipped: clustering genes vs. clustering the experiments (arrays). The Lee dataset was obtained from Lee et al. [LDAM04], and consists of 591 gene-expression experiments on yeast obtained from the Stanford Microarray database [Gol03] (http://genome-www5.stanford.edu/), and also contains a *Gold* standard based on Gene Ontology(GO) annotations (http://www.geneontology.org). The Gold standard contains 121,406 pairwise links (out of a total of 15,744,466 gene pairs) between 5,612 genes in the Lee data that are known to be functionally related. The Gold standard was generated using Gene Ontology biological process from level 6 through 10.

The 20-Newsgroup (20-NG) dataset is a popular dataset for text classification [Lan95], and is widely available on the web including the KDD UCI repository (http://kdd.ics.uci.edu/). It consists of 20,000 Usenet articles taken from 20 different newsgroup, 1,000 from each newsgroup. The 20 groups can also be categorized into 6 high-level categories shown in Table 3, which are then used as the class labels for evaluating the clustering algorithms. The 20-NG data contains over 50,000 distinct words after removing punctuations, common words such as articles, and stemming. Since we use the 20-NG data to evaluate (unsupervised) clustering algorithms, we used an unsupervised approach for selecting features; we selected 4,292 most frequent words (all words occurring ≥ 100 times over the 20,000 documents) as features.

Table 3: The 6 top-level classes ($\mathcal{C}$) in the 20-Newsgroup data.

$\mathcal{C}$	$\lvert\mathcal{C}\rvert$	Member newsgroups
Computers	4,959	*comp.graphics*, *comp.os.ms-windows.misc*, *comp.sys.ibm.pc.hardware*,*comp.sys.mac.hardware*, *comp.windows.x*
Recreation	3,984	*rec.autos*, *rec.motorcycles*, *rec.sport.baseball*, *rec.sport.hockey*
Science	3,989	*sci.crypt*, *sci.electronics*, *sci.med*, *sci.space*
Miscellaneous	988	*misc.forsale*
Talk	2,994	*talk.politics.misc*, *talk.politics.guns*, *talk.politics.mideast*
Religion	2,994	*talk.religion.misc*, *alt.atheism*, *soc.religion.christian*

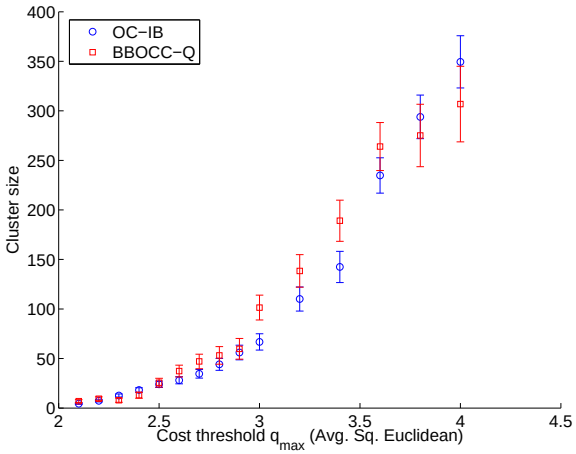
Synthetic Data: These datasets are useful for verifying algorithms since the true labels are known exactly. The Sim-2 dataset was generated using 4 2-D Gaussians of different variances (Figure 3) and a uniform distribution. Similar datasets were generated with 5 Gaussians in 10-D and 40-D to produce Sim-10 and Sim-40 datasets.

We also created two high-dimensional data-sets to test the hypothesis that seeding local search with a global search is more important when a dense region is small in mass and is relatively isolated from the rest of the data. For this we created two high dimensional data-sets, each consisting of 40 dimensions and 4,026 data points. The *Hard* data consists of 3 spherical Gaussians with one of the Gaussians containing 5% of the probability mass, and being separated by a distance much larger than the standard deviation of two other wider and highly overlapping Gaussians. For the *Easy* dataset, we use the same 3 Gaussians but place the small dense Gaussian close to the center of the rest of the data.

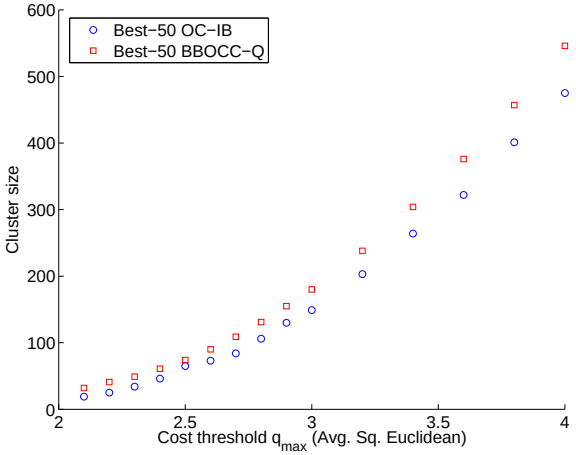
11.3 BBOCC-S: an improved local search for One Class

BBOCC-Q tends to give better results than OC-IB on a variety of datasets, as measured by the size of the clusters returned for the same cost threshold q_{max}. Figure 6 compares the two

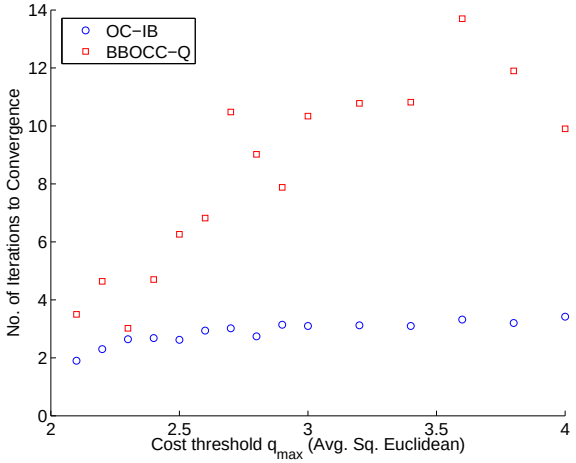
methods on two such datasets. Note that although OC-IB converges in a very small number of "iterations" (Figure 6 (c)), each iteration of OC-IB actually moves the Bregmanian ball n times.



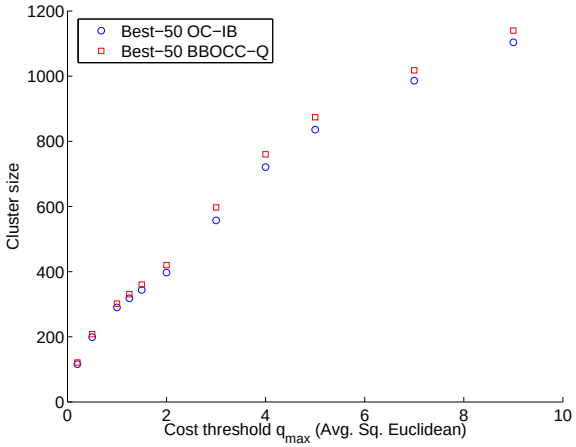
(a) Alizadeh, avg over 50 trials



(b) Alizadeh, best in 50 trials



(c) Alizadeh, No. of iterations to convergence



(d) Sim-2, best in 50 trials

Figure 6: Comparison of performance of OC-IB with BBOCC-Q using Sq. Euclidean as D_ϕ: (a) Comparison on Alizadeh dataset using average size of the cluster discovered (y-axis) for a given cost threshold as input to both the algorithms (x-axis), including one +/- standard deviation of the average. The results were averaged over 50 trials with random initialization. (b) Comparison on Alizadeh dataset when the solution is picked as best of 50 trials for both OC-IB and BBOCC, (c) Average number of trials to convergence for BBOCC-Q vs. OC-IB (y-axis) for a given cost threshold (x-axis), averaged over 50 trials. (d) Results over 50 trials on the Sim-2 dataset.

Does the use of a fixed size as input in BBOCC-S also have any positive effect on performance? If both the enhancements over OC-IB lead to an improved local search, one should expect BBOCC-S to do better than both OC-IB and BBOCC-Q, and BBOCC-S should beat OC-IB by a larger margin than the margin by which BBOCC-S beats BBOCC-Q. However, a straightforward comparison of BBOCC-S with the other two approaches is not possible since

we cannot control the output cluster sizes in either BBOCC-Q or OC-IB.

Comparing BBOCC-S with OC-IB and BBOCC-Q using the duality: The fact that BBOCC-S and BBOCC-Q (and OC-IB) use dual formulations of the same problem (Problem definitions 2 vs. 1 respectively) allowed us to experimentally compare the three methods as follows: BBOCC-S was run with a random initialization using a given size as input. Then, the output cluster's cost Q_{one} was measured, and a trial of BBOCC-Q and OC-IB each was run with random initialization using this cost as input q_{max}. The corresponding output cluster sizes and cluster costs for the two algorithms were then stored. This operation was repeated for the particular cluster size 50 times with random initialization and the results from all the three algorithms were stored. These experiments were repeated over a range of cluster sizes as inputs to BBOCC-S (27 different sizes ranging from 2 to 2000), with 50 trials for each cluster size, resulting in a total of 1,350 experimental trials for each of the three methods. Figure 7 (a) shows a plot of resulting individual runs using such an approach.

At the end of the process we obtained a range of cluster sizes generated by BBOCC-S, BBOCC-Q and OC-IB for a given cost threshold q_{max}, and we measured the fraction of times over the 50 trials that BBOCC-S gave a larger cluster size than BBOCC-Q. Similarly we also measured the fraction of time over the 50 trials that BBOCC-S gave a larger cluster size than OC-IB. We found that on the Alizadeh data, BBOCC-S does beat both OC-IB and BBOCC-Q in most of the trials. Figure 7 (b) compares BBOCC-S with BBOCC-Q and OC-IB over bucketed ranges of cost thresholds. Each of the six buckets had approximately equal number of trials in them. Interestingly, BBOCC-S generally beats OC-IB by a larger margin than BBOCC-Q. This supports the hypothesis that both the algorithmic enhancements in BBOCC-S over OC-IB improve local search. We found similar results on other datasets.

Explanation for the improvements: Our explanation for the improvement observed in BBOCC-S over BBOCC-Q is as follows: A fixed cost threshold as input required by definition 1 in BBOCC-Q produces an undesirable side-effect, when the goal is to find small dense clusters; for a random seeding on a dataset where the small dense region comprises a small fraction of the total number of data points, there is a high probability that the random seed is in a sparse region. In BBOCC-Q, the ball can only "grow" to a point where adding another point would cause the average distance of the points within the ball to be greater than q_{max}. In a sparse region this causes very few points to get added to the ball. For example, in the extreme case when the nearest point to the seed is farther than q_{max}, 0 points would get added to the cluster. Since the next operation updates the centroid by averaging all the member points, when very few points are present, the algorithm converges rapidly but prematurely, or in other words, the search is unable to "move" much towards the distant dense region. The fixed cluster size input used by problem definition 2 and BBOCC-S ameliorates this premature convergence problem by "scaling" the cluster neighborhood to a larger region when the initialization is in a sparse region. This happens because the number of points assigned in each iteration is fixed; a sparse region causes the ball to automatically expand, and the ball shrinks as it moves towards a denser region after each iteration.

To conclude, BBOCC-S and BBOCC-Q outperform OC-IB and that both the enhancements; the batch updated and fixed size s as input (see Figures 6 and 7) contribute to the improved quality of the local search.

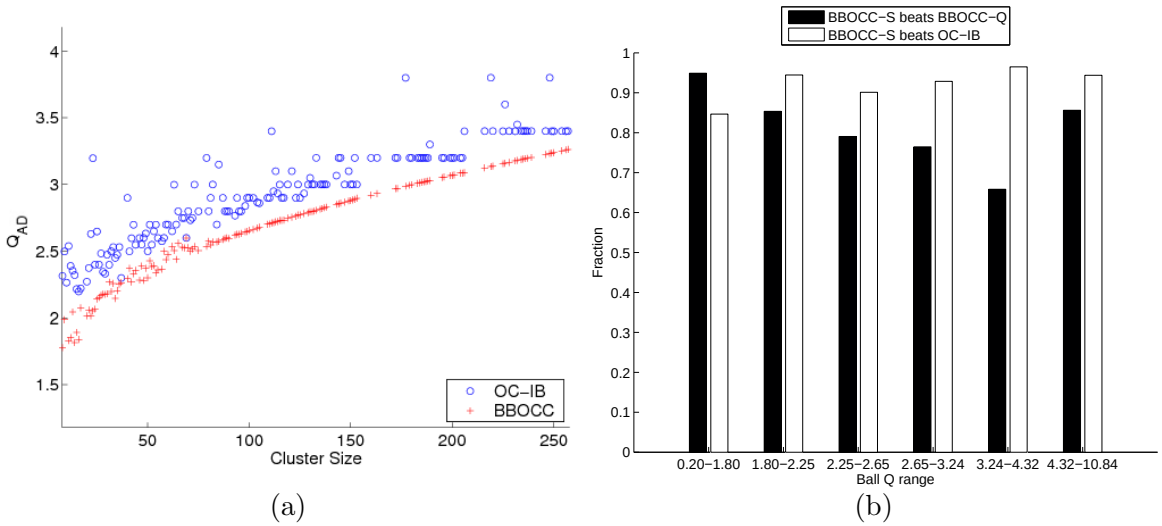


Figure 7: Comparison of performance of OC-IB [CC04] and BBOCC-Q w.r.t. BBOCC-S on Alizadeh data. (a) Actual cluster cost distributions for various cluster sizes for OC-IB as compared to BBOCC-S. (b) A comparison of BBOCC-S w.r.t to OC-IB and BBOCC-Q over a range of cluster costs. y-axis represents the fraction of trials in a range of costs (x-axis) that BBOCC-S had larger cluster sizes than BBOCC and OC-IB. For each cluster size used in BBOCC-S, experiments were run 50 times, and the output cluster cost in each experiment was used to run a corresponding trial of BBOCC-Q and OC-IB. Each bucket contains roughly 225, or 1/6th of the total number of trials. Each trial was randomly initialized.

11.4 Effectiveness of HOCC and Hybrid Search for One Class

In Section 11.3 we showed the effectiveness of the local search method BBOCC-S. We now present results that show the effectiveness of the global search algorithm HOCC presented in Section 10.1, and lesion studies to show the usefulness of the hybrid combination of the two: Hyper-BB and confirm that it finds solutions very close to the global optima.

Evaluation Methodology: We studied the importance of the local and global search in our hybrid algorithm by comparing Hyper-BB against running HOCC and BBOCC separately. We also compared our methods against "BBOCC-10", where we pick the best of 10 trials of BBOC, and against a "Naive" algorithm that picks the s closest points to the mean of data $\mathcal{X}$ as the solution. We also show comparisons against the lower bound of the global optima, guaranteed by Proposition 10.2, by plotting half of the cost of the solution HOCC. We plot this lower bound as the legend "Optimal-LB". For every tested cluster size, results of BBOCC and BBOCC-10 were averaged over 50 trials.

Lesion Results: Figure 8 shows the results on four different datasets measured in terms of the Bregmanian ball cost Q_{one}. Clearly, the global search (HOCC) by itself performs substantially better than the local search (BBOCC) for all cluster sizes, but even the combination (Hyper-BB) shows significant improvement over HOCC. The "Optimal-LB" legend shows the lower bound for the globally optimal solution as half of the cost of HOCC solution. Clearly, Hyper-BB gets very close to this theoretical lower bound, showing that in practice Hyper-BB finds a solution very close to the global optima. Note that the (unknown) true globally optimal

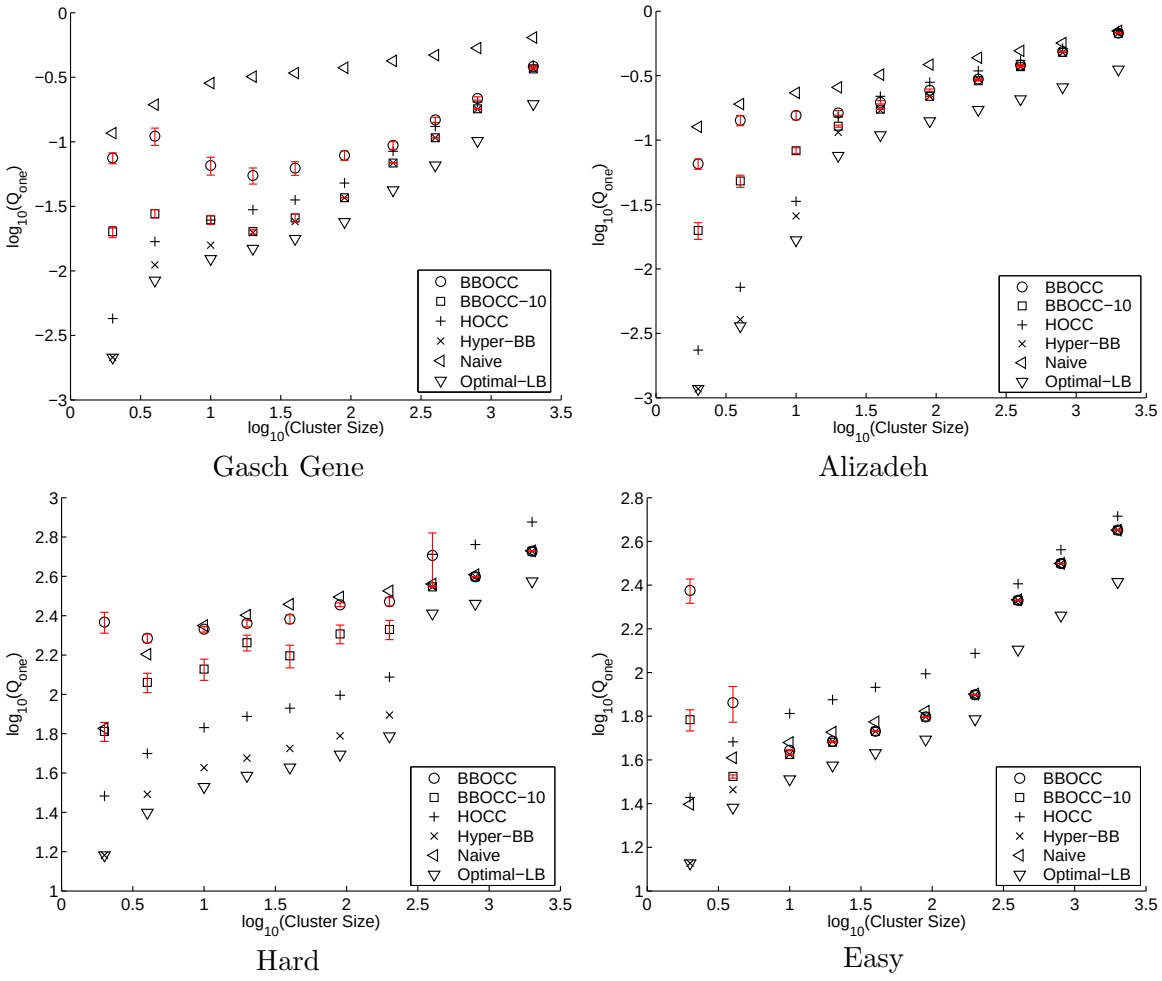


Figure 8: Comparison of performance of BBOCC, BBOCC-10, HOCC, Hyper-BB and the Naive algorithm using Q_{one} as training and evaluation cost functions on four datasets. BBOCC-S was used for all the BBOCC evaluations shown. The (unknown) true global optimal solution lies between the Optimal-LB (the theoretical lower bound) and the HOCC costs, which is the region where Hyper-BB costs fall. The y-axis is the evaluation cost, while the x-axis is the size of the cluster for which the evaluation was performed. For BBOCC and BBOC-10, the results were averaged over 50 trials. For BBOCC and BBOCC-10, one std. dev. error bars are plotted but are sometimes too small to be visible.

solution would have a cost in between that of the Optimal-LB (lower bound) and HOCC (upper bound). In light of this fact, the results for low coverages on the Hard, Gasch Gene and the Alizadeh dataset are remarkably close to the lower bound, thus empirically demonstrating that Hyper-BB gets very close to the globally optimal solution when finding a small, dense region.

The property that HOCC and Hyper-BB are deterministic while BBOCC's performance varies substantially across multiple trials (not plotted in Figure 8 for clarity) makes the improvement even more significant. On the Easy dataset BBOCC performed as well as Hyper-BB for clusters of size larger than 10, while the naive algorithm that simply picks the data center gave results as good as Hyper-BB. But for the other three non-trivial datasets, the Hybrid algorithm clearly performs better than its components, and even compared to BBOCC-10. One of the surprising observations to be made is that for the Hard dataset and for smaller cluster sizes, BBOCC by itself performs worse than Naive, even though this dataset was not designed to have the dense region at the center of the data. These lesion studies support our intuition that both the local and global components of Hyper-BB are important in identifying small, dense regions.

Results against labeled data: One-class clustering aims at finding one tightly knit cluster rather than classifying all the data. However, one indication of the quality of the detected cluster is its purity in terms of distribution of class labels, if such labels are available. In our experiments, small clusters were invariably very pure. For example, when we applied Hyper-BB to cluster the 173 experiments in Gasch Array data using the 6,151 genes as features, the closest 7 points to the densest region were all labeled as heat shock experiments. This represented a precision of 1 for a recall of 0.41 for recovering an experiment type whose prior in the data was 0.1 (17 out of 173).

11.5 Evaluation Methodology for the k-Class algorithms: BBC and DGRADE

Now that we have shown the effectiveness of the local search BBOCC-S and the value of combining global search and local search for the One Class problem, we present more extensive results for the general case of finding multiple dense clusters using BBC-S (and for the corresponding Soft BBC model), and the effectiveness of combining it with Pressurization and/or with the global search method, DGRADE. We now describe various aspects of our evaluation setup, before presenting the results:

Evaluation Criteria: Evaluating clustering is a challenging problem even when labelled data is available [SGM00]. Depending upon the type of the labeled data, we performed the following three different types of evaluations:

1. *Adjusted Rand Index*: Adjusted Rand Index was proposed by [HA85] as a normalized version of Rand Index, and returns 1 for a perfect agreement between clusters and class labels and 0 when the clustering is as bad as random assignments. ARI can be used on the Gasch Array, 20-NG and the synthetic datasets since the true class-labels are available.

2. *p-value*: We use p-value to evaluate individual clusters of Yeast genes found using BBC for the Lee dataset. *Funspec* (http://funspec.med.utoronto.ca/) is a popular Yeast database query interface on the Web that computes cluster p-values for individual clusters using the hypergeometric distribution, representing the probability that the intersection of a given list of genes with any given functional category occurs by random chance. $p - value$ is a commonly used measure of individual cluster quality used by bioinformatics researchers.

3. *Overlap Lift*: For evaluating the overall clustering quality, it is not possible to use ARI to evaluate against the links in the Lee Gold standard. In general, measuring overall clustering quality for genes is quite difficult since only an incomplete and partially verified ground truth is known, such as the links in the Lee Gold standard. We propose *Overlap Lift* as a measure of the statistical significance of our clustering results against the gold standard as follows: A cluster containing w genes in one cluster creates $w(w-1)/2$ links between genes, since every point within the cluster is linked to every other point. Therefore, k clusters of size $\{w_j\}_{j=1}^k$ would result in a total of $l_c = \sum_{j=1}^k w_j(w_j-1)/2$ links. The fraction of pairs in the Gold standard that are linked f_{linked} is known (for example for Lee dataset $f_{linked} = 121,406/15,744,466 = 0.007711$). If we construct a null hypothesis as randomly picking l_c pairs out of $n(n-1)/2$ pairs in the Gold standard, we can expect $l_{null} = f_{linked} l_c$ pairs to be correctly linked. A good clustering should result in (a lot) more correctly linked pairs than l_{null}. If l_{true} is the number of correct links observed (which will always be $\leq l_c$) in our clustering, then the Overlap Lift is computed as the ratio $\frac{l_{true}}{l_{null}}$, which represents how many times more correct links are observed as compared to random chance. A larger ratio implies better clustering.

Handling "don't care" points in evaluations: All the evaluations are performed across a range of coverage of the data; a coverage of $s/n = 0.4$ implies 40% of the points are in clusters while the remaining 60% are in the "don't care" or the background cluster. The points in the background or the "don't care" clusters are excluded from all evaluations. To keep the comparisons fair, all methods are compared against each other only across the same coverage.

Evaluating Soft BBC: We tested Soft BBC using Gaussians as the exponential mixture components. There are eight possible flavors of Soft BBC (Table 1) depending upon the choice of the updatable parameters. We present results on the Soft BBC implementation, flavor 6, i.e. with updatable, unequal variances with a fixed α_0. We also compared Soft BBC for Gaussians with the alternative soft model called Mixture-6 (Section 8.5).

Hard Assignments for Soft BBC: To compare Soft BBC against BBC and other hard assignment methods, on convergence, the points are assigned to the mixture with the largest probability, i.e. to $j = \underset{j=0\rightarrow K}{\text{argmax}}\, p(Y_i = j|\mathbf{x}_i)$. The estimation of p_0 described in Section 5.5 results in *approximately* $(n \times \alpha_0)$ points getting assigned to the background. In order to ensure that exactly $(n-s)/n$ points are assigned to the "don't care" set, a post-processing is performed where we: (1) compute $p^i_{max} = \max_{j=0}^k (p(Y_i = j|\mathbf{x}_i))$ for each $\mathbf{x}_i$, (2) set $p_0^\dagger$ to the s^{th} largest value in $p^i_{max}[i]_{i=1}^n$, (3) put all points below $p_0^\dagger$ into the "don't care" cluster, and assign rest to cluster $j = \underset{j=1 to k}{\text{argmax}}\, p(Y_i = j|\mathbf{x}_i)$. A similar conversion was required for evaluating Mixture-6.

Comparison against other methods: We also compared our method with Bregman Hard Clustering, Single Link Agglomerative clustering and DBSCAN. Bregman Hard Clustering assigns every data point into a cluster. To be able to compare it meaningfully with BBC, we picked s points closest to their respective cluster representatives. This procedure was also used for Single Link Agglomerative clustering. For the two DBSCAN parameters, we set $MinPts$ to 4 as recommended by [EKSX96], while we searched for Eps that resulted in s points in clusters. k is automatically estimated by DBSCAN while for all the other methods and datasets, when evaluating BBC with Pressurization, k was set to $|\mathcal{C}|$ (Table 2), except for the Lee dataset (where $|\mathcal{C}|$ is not known) where we set k to 9.

All five methods use the same and appropriate distance measure that corresponds to the D listed for each of the datasets in Table 2; Sq. Euclidean for the synthetic Gaussian datasets, Pearson Distance for the gene-expression datasets, and (1−Cosine Similarity) for the 20-newsgroup data.

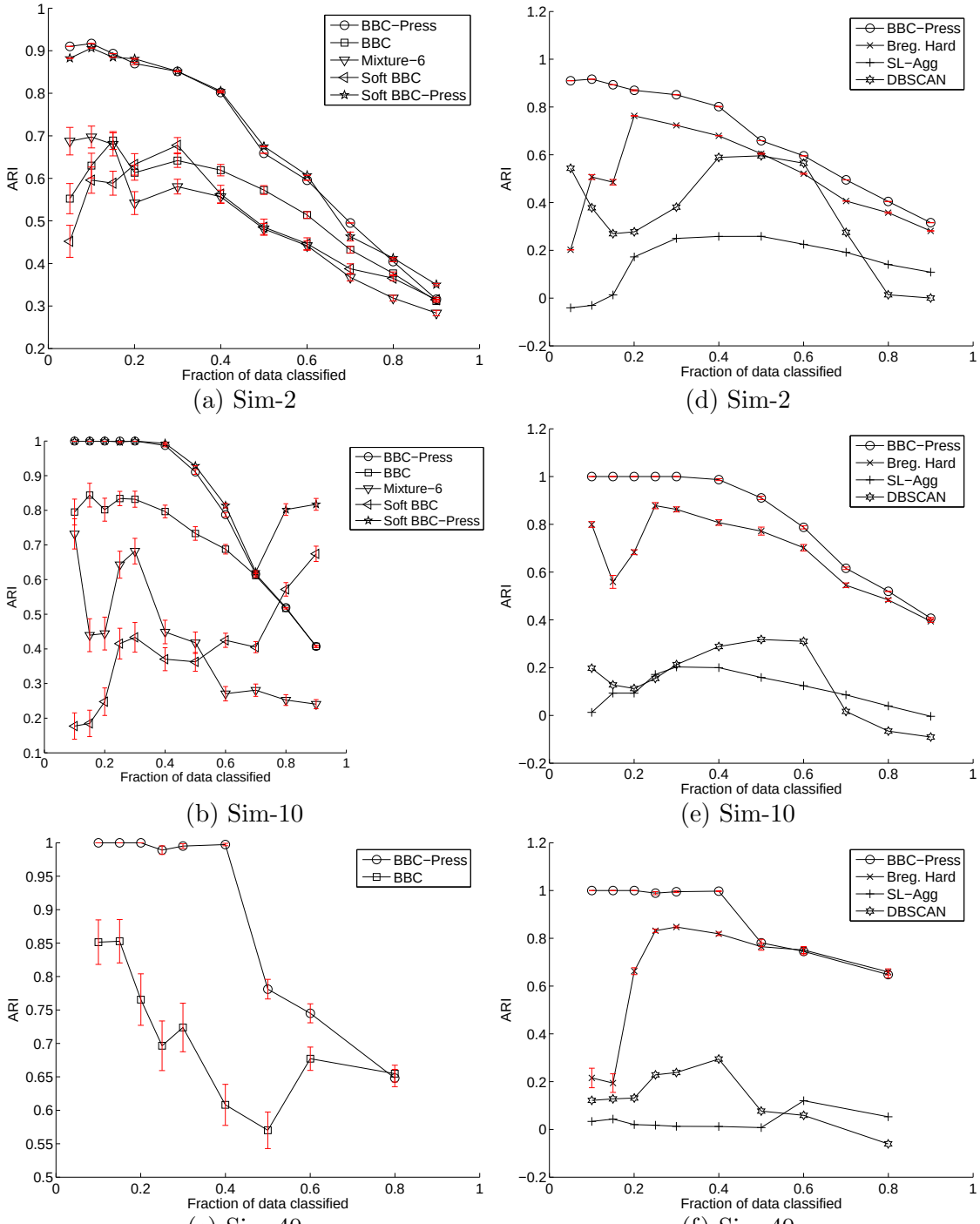
11.6 Results for $k > 1$: BBC with Pressurization

For the lower dimensional datasets, both Soft and Hard BBC with Pressurization perform extremely well, giving near-perfect results (ARI ≈ 1) for up to 40% coverage on Sim-10 data and an ARI between 0.8 and 0.9 for up to 40% coverage on Sim-2 data. As expected, both BBC and Soft BBC without Pressurization tend to be a lot more sensitive to initialization, thus exhibiting noticeable error-bars [18]. For Sim-40 and all the real datasets, results are only shown for Hard BBC with Pressurization (BBC-Press). This is because exponential mixture models in general, including Bregman Soft Clustering, Mixture-6 and Soft BBC, all suffer from an inherent problem that makes them impractical for high dimensional datasets: there are rounding errors while estimating the mixture membership probabilities, and these rounding errors worsen exponentially with the dimensionality of the data d (e.g., for Gaussians, when L.H.S. of equation 19 is substituted for $p_{(\psi,\theta)}$ in equation 8), so much so that the models often do not work well beyond $d = 10$. However, the main purpose of designing Soft BBC was to show that a fundamental generative model lies behind BBC (Section 7). We tested the Soft BBC and the Mixture-6 models on Sim-2 and Sim-10 datasets, mainly to validate Soft BBC and Soft BBC-Press; figure 9(a) and (b) show that Mixture-6 has no clear performance advantage over Soft BBC. Furthermore, Mixture-6 does not conform to the form required to incorporate Pressurization and does not correspond to any known "hard" model for Bregman divergences.

On the Sim-40 dataset, BBC-Press continues to give an ARI ≈ 1 for up to 40% coverage (Figure 9(c)). These results are impressive given that the ARI was obtained as averages of multiple runs with random seeding. The improvement against labeled data using BBC Press as compared to BBC is also quite remarkable for the two micro-array datasets Gasch Array and Lee, and a similar trend is seen for 20-Newsgroup as well. This indicates that Pressurization works well on a variety of real datasets; from very high dimensional gene experiments (Figure 10(a)) where most of the data is relevant, discovering small number of relevant gene clusters on high-dimensional microarray data (10 (e)), to clustering large and high-dimensional documents (10 (c)).

On both artificial and real datasets (Figure 9, 10), DBSCAN, Single Link Agglomerative and Bregman Hard Clustering all perform much worse than BBC-Press in general, and especially when clustering a part of the data. Note that these results are based on on labels that were *not* used for clustering; using ARI on Gaussians, Gasch Array and the 20-newsgroup data, and using Overlap Lift on Lee, and are therefore independent of the clustering methodology. Figure 10(f) shows that (1) BBC-Press not only beats other methods by a wide margin but also shows high enrichments of links for low coverages (over 6 times for 5 % coverage), and (2) Single Link Agglomerative clustering does not work well for clustering genes and gives results not much better than random. On all datasets, Single Link tends to perform the worst; one explanation might be its inability to handle noisy data. For 20-Newsgroup, the ARI of Single Link is not clearly visible although it has been plotted because it hovers close to 0 for all coverages. In

[18] Note that the error bars were plotted on all the local search algorithms, but are often too small to be visible.

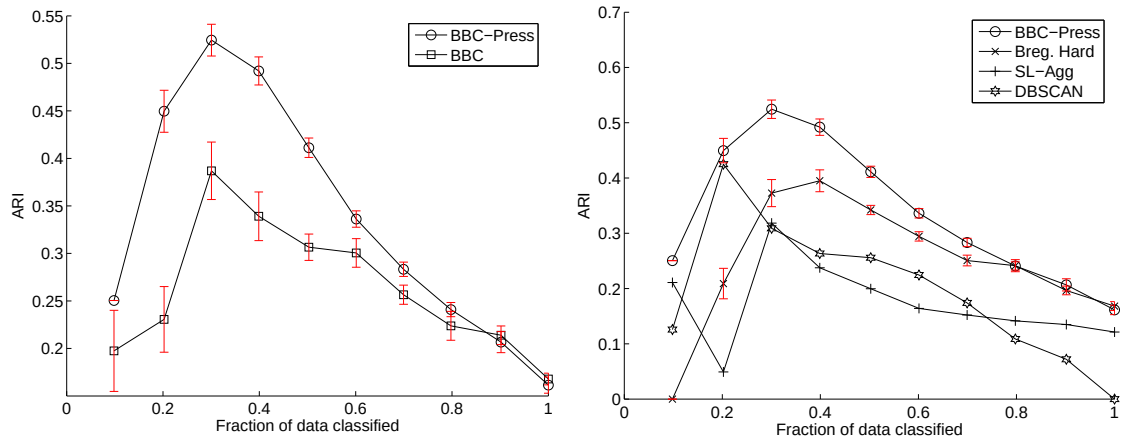


(a) Sim-2 (d) Sim-2

(b) Sim-10 (e) Sim-10

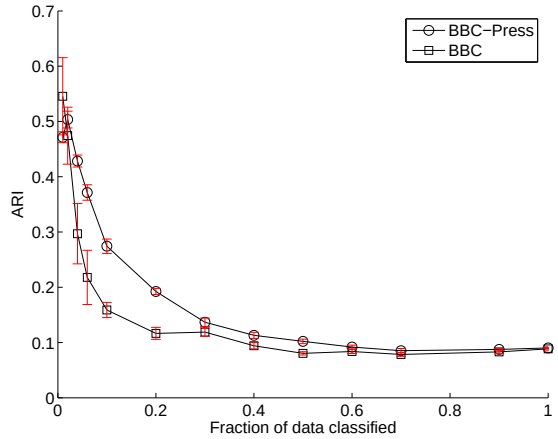
(c) Sim-40 (f) Sim-40

Figure 9: Evaluation on synthetic Gaussian data of increasing dimensionality using ARI: (a), (b) and (c) demonstrate the effectiveness of Pressurization. (d), (e) and (f) show effectiveness of BBC-Press as compared to three other methods: Bregman Hard Clustering, Single Link Agglomerative and DBSCAN. Error bars of one std. deviation are shown (but are sometimes too small to be visible) for local search methods for which ARI is plotted as the average over 100 trials with random initialization.

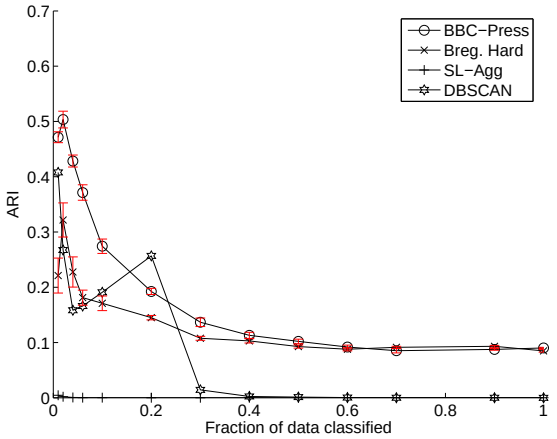


(a) Gasch Array

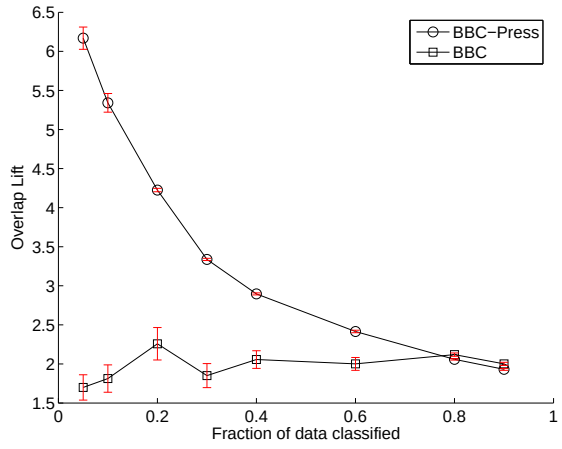
(b) Gasch Array



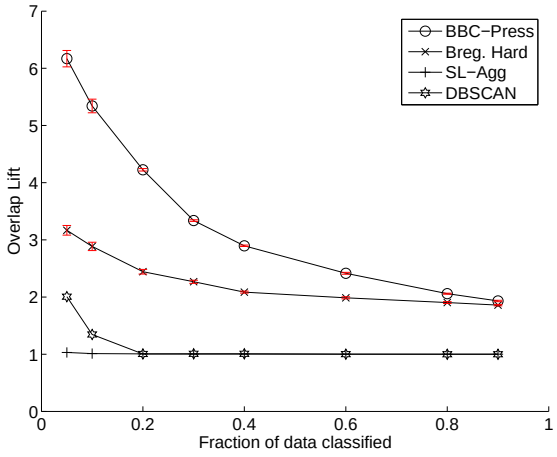
(c) Cleaned 20-Newsgroup



(d) Cleaned 20-Newsgroup



(e) Lee



(f) Lee

Figure 10: Evaluation of BBC-Press on real data using ARI for Gasch Array and Cleaned 20-NG and Overlap Lift for Lee, as compared to BBC, Bregman Hard Clustering, Single Link Agglomerative, and DBSCAN. Local search results were averaged over 20 trials for Gasch Array and Cleaned 20-NG, and over 10 trials for Lee. The corresponding one std. dev. error-bars are plotted, but are sometimes too small to be visible.

fact, for some situations (Figure 9(d) to (f)), DBSCAN and Single Link Agglomerative give slightly worse than random performance resulting in ARI values that are slightly below 0. The performance difference between our method (BBC-Press) and the other three methods is quite significant on all the six datasets, given the small error bars. Additionally, if we were to pick the minimum-cost solution out of multiple trials for the local search methods, the differences in the performance between BBC-Press vs. DBSCAN and Single Link becomes even more substantial.

Selecting size and number of dense clusters in the absence of DGRADE seeding: In BBC-Press, s controls the number of data points in dense clusters. The dense clusters were invariably very pure when using BBC-Press, with near-perfect clusters on the Gaussian data for s of up to 40% of n, while on the Gasch Array dataset the performance peaks at a coverage of around 0.3 but shows a general decline after that. The rapid increase in cluster quality with decreasing s is more pronounced in BBC-Press than in the other methods, and shows that on these datasets, dense regions are indeed highly correlated with the class labels. In practice, selecting dense clusters with BBC-Press requires choosing an appropriate s and k. If small amounts of labeled data is available, the best k can be estimated for a fixed s using an approach such as PAC-MDL [BL04], while a reasonable s can be picked by applying BBC-Press on a range of s and picking the "knee" (e.g. Figures 9(a),(b),(c) and 10(b) show a sudden decline in ARI near $s = 0.4 \times n$). Alternatively, in many problems k can be an input, while s simply has to be a small threshold (e.g. for finding a small number of relevant web documents or a small number of relevant genes).

Evaluating individual clusters: Although the results based on ARI and Overlap Lift show the effectiveness of our method, visual verification serves as another independent validation that the clusters are not only statistically significant but also useful in practice. For the Sim-2 dataset, it is easy to verify the quality of the clusters visually (Figure 3)(b). For the Gasch Array clustering, most clusters were generally very pure using BBC-Press for lower coverages. For example, when only 70 out of 173 experiments are clustered by repeating BBC-Press 20 times and picking the lowest cost solution, the average ARI is around 0.6 over 12 classes. Some clusters are even purer, for example, one of the clusters contained 12 out of 13 points belonging to the class "YPD" [19]. Similarly, for the Lee dataset, when clustering only 600 genes into 30 clusters and pruning the rest, we verified a high purity cluster using FunSpec; 10 out of 14 genes in one of the clusters belonged to the functional category "cytoplasmic and nuclear degradation" (p-value of $< 10^{-14}$). Many other gene clusters on the Lee dataset also had low p-values for some of the categories recovered by FunSpec.

11.7 Model Selection using DGRADE

In an alternative setting where DGRADE is used to seed BBC, we compared both DGRADE and BBC seeded with DGRADE with Bregman Hard Clustering, Single Link Agglomerative clustering and DBSCAN. The experimental setup and evaluation was similar to when comparing BBC with Pressurization against the other three methods, except that for a given coverage, k was automatically determined by DGRADE, and this k was then used as input for methods that require it; BBC, BBC-Press, Bregman Hard and Single Link. For DBSCAN, which determines k internally, as before, we set $MinPts$ to 4 as recommended by [EKSX96], while we searched

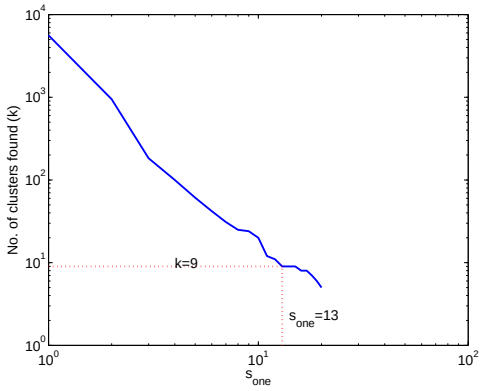
[19] Which represents one of the class of experiments set up by [Gas00]

for *Eps* that resulted in s points in clusters. The input parameter s_{one} required by DGRADE was determined automatically by using the approach described in Section 10.3. DGRADE then uses the same s_{one} for smaller coverages, which can result in a smaller k (Figure 12) as the lesser dense clusters become "don't care" points, and the corresponding k is then used as input to all the algorithms requiring it. When seeding (Hard/Soft/Pressurized) BBC with the output of DGRADE, the cluster centroids output by DGRADE were used as the seed/initial centroids. We now presents results on DGRADE when it used for seeding BBC, on two real and three synthetic datasets.

Dataset	s_{one}	k, when $s = n$	User input
Gasch Array	4	11 [a]	$k = 12$
Lee	10	9	$stability > 1$ [b]
Sim-2	62	4	None [c]
Sim-2	57	5	$k = 5$ [d]
Sim-10	104	5	$k = 5$
Sim-40	57	5	$k = 5$

[a]Closest possible k to 12 found when $s_{one} = 3$. For $s_{one} = 3$, k increases to 14.
[b]See plot (b) in this figure.
[c]Figure 5(c) and (d)
[d]Figure 5(b)

(a)

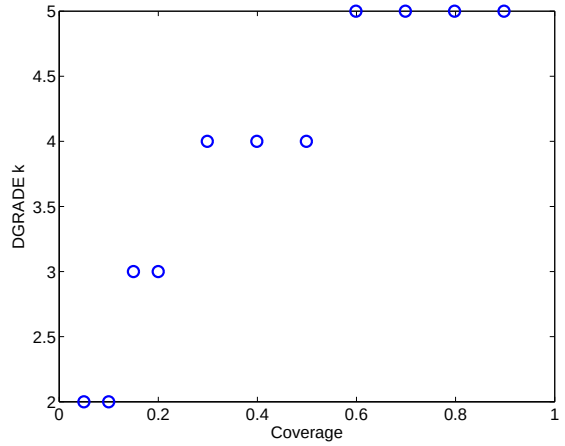


(b)

Figure 11: (a) Automatically determined s_{one} for DGRADE on various datasets, using the approach described in Section 10.3. (b) For the Lee data, selecting the largest k (smallest s_{one}) with $stability > 1$ gives a $stability = 3$, $s_{one} = 13$ and $k = 9$.

Ability to estimate s_{one} automatically: In the table in Figure 11(a), the first column shows the values of s_{one} determined automatically using one of the three methods described in Section 10.3, the second column shows the number of clusters discovered when clustering all of the data ($s = n$), while the third column shows the user input, if any required by DGRADE. For all the datasets except Lee, k was known and was used to determine s_{one} automatically. For the Sim-2 dataset, when both k and s_{one} were determined automatically using the maximum cluster stability criteria, we obtained $k = 4$ (Figure 5(c) and (d)). For the Lee dataset, by using a cluster stability threshold > 1 we obtained $k = 9$ and $s_{one} = 10$. Figure 11 (b) shows the process of automatically determining both k and s_{one} for the Lee dataset; interestingly, $k = 9$ also had the maximum stability of 3 in the range $2 \leq s_{one} \leq 20$, for which k ranged from 948 (for $s_{one} = 2$) to 5 (for $s_{one} = 20$).

Using cluster centroids from DGRADE for seeding and selecting k, for variable coverages: For a constant s_{one}, the results of DGRADE are not only deterministic for varying values of s, but also have another useful property that follows directly from Algorithm 6; using a smaller s gives clusters that are guaranteed to be subsets of the DGRADE clustering with a larger s. This effect can also be seen in Figure 5(b) vs. 5(a), when s was reduced from from 1298 to 750. Eventually, some of the less dense clusters disappear completely resulting in a decline in k returned by DGRADE . However, the remaining centroids output by DGRADE remain unchanged. This phenomena can be seen for the 5 different datasets in Figure 12 and provides us with a principled way of selecting centroids for seeding BBC; and for choosing the corresponding k for coverages less than 1, when comparing with other methods that require k



(a) Sim-2

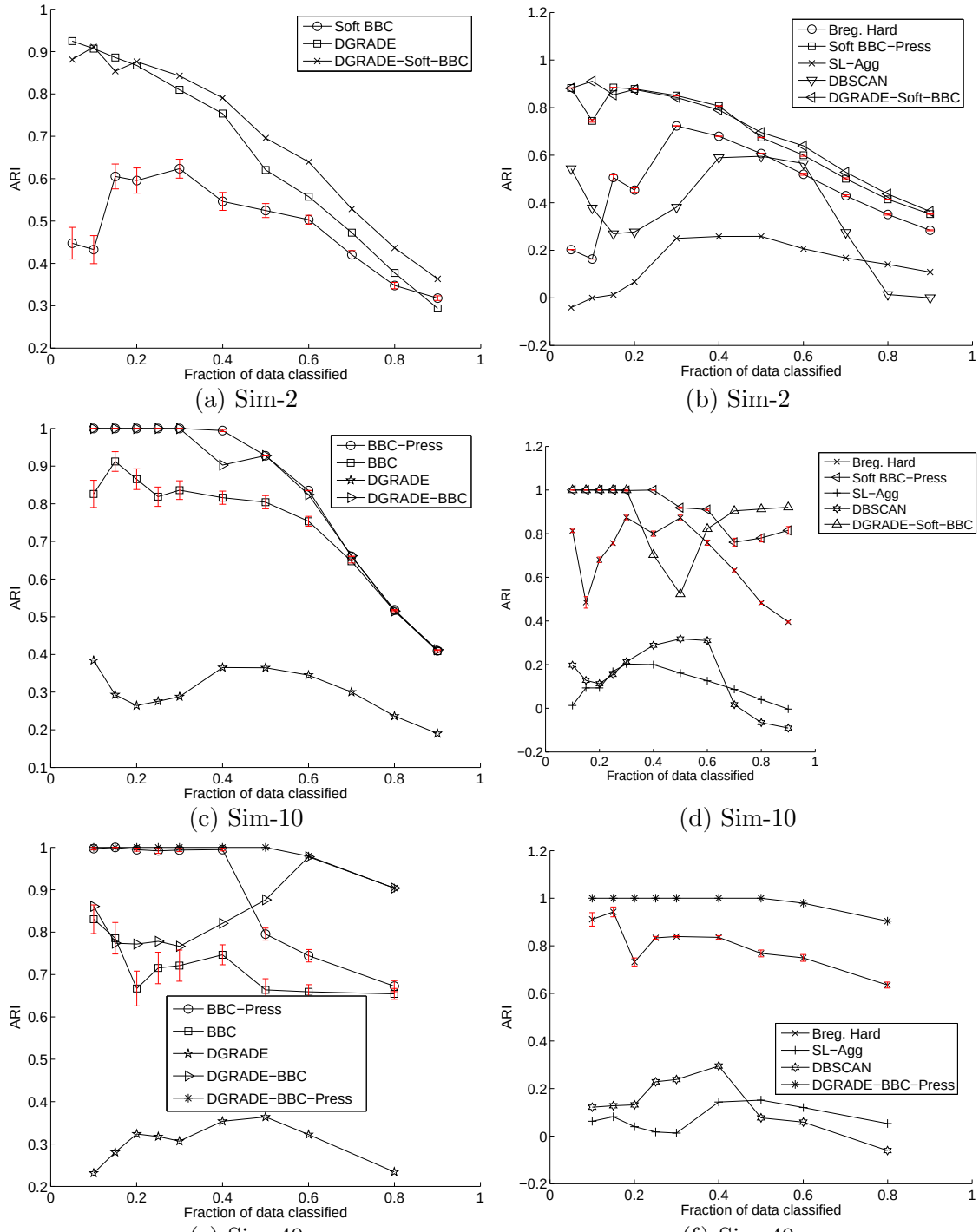
Dataset	cov.(s/n) vs. k			
	Min.		Max.	
	s/n	k	s/n	k
Gasch Array	0.1	3	1.0	11
Lee	0.05	6	1.0	9
Sim-10	0.1	3	1.0	5
Sim-40	0.1	4	1.0	5

(b) Other datasets

Figure 12: (a) On Sim-2 data, the number of clusters k found by DGRADE declines asymptotically with the fraction of densest data clustered (s/n). (b) A summary of a similar trend on the other datasets. The maximum k corresponds to $s = n$, and the minimum k corresponds to the smallest coverage. s_{one} was held constant for all coverages, and corresponded to the automatically determined values of 4,10, 57, 104, and 57 for the Gasch, Lee, Sim-2, Sim-10 and Sim-40 datasets respectively.

as an input. The k corresponding to those shown in Figure 12 for various coverages were used as inputs to Bregman Hard and Single Link Agglomerative Clustering. Also, the corresponding centroids output by DGRADE for various coverages were used as inputs for seeding any of the forms of BBC (Hard/Soft/Pressurized).

DGRADE seeding works well with BBC: Figure 13 and 14 show results for BBC seeded with DGRADE as compared to the other alternatives; BBC with Pressurization and the other three benchmark algorithms. For the Sim-2 dataset, when DGRADE was used to seed Soft BBC, the results were comparable to that of Soft BBC with Pressurization (Figure 13 (b)), while the results of DGRADE as a clustering algorithm by itself result were quite good (Figure 13 (a)). Similar trends were seen on the Sim-10 (13 (c) and (d)) dataset. On the Sim-40 dataset (13 (e) and (f)), although DGRADE does not perform well by itself, when used for seeding BBC-Press, the combination gives results that are far superior to all other algorithms, and beats even BBC-Press; an ARI close to 1 is observed for coverages as high as 0.6 as compared to only until 0.4 for BBC-Press. One possibility is that the global search bias behind DGRADE provides additional advantages to a robust locals search algorithm such as BBC-Press, and especially on higher-dimensional datasets. This agrees with the intuition that the local search problems should become more severe with increasing dimensionality of the data (Sim-40 vs. Sim-2), and is similar to boost in performance observed for the One Class scenario (Section 11.4). This phenomena of DGRADE and BBC-Press improving results as a combination is also observed on the really high-dimensional real datasets- the Gasch Array (Figure 14 (a) and (b)). However, this relationship is only seen for the lowest coverage of 0.05 on the Lee dataset (Figure 14 (e)), perhaps because the fraction of genes that are usually dense when clustering genes is usually very small.

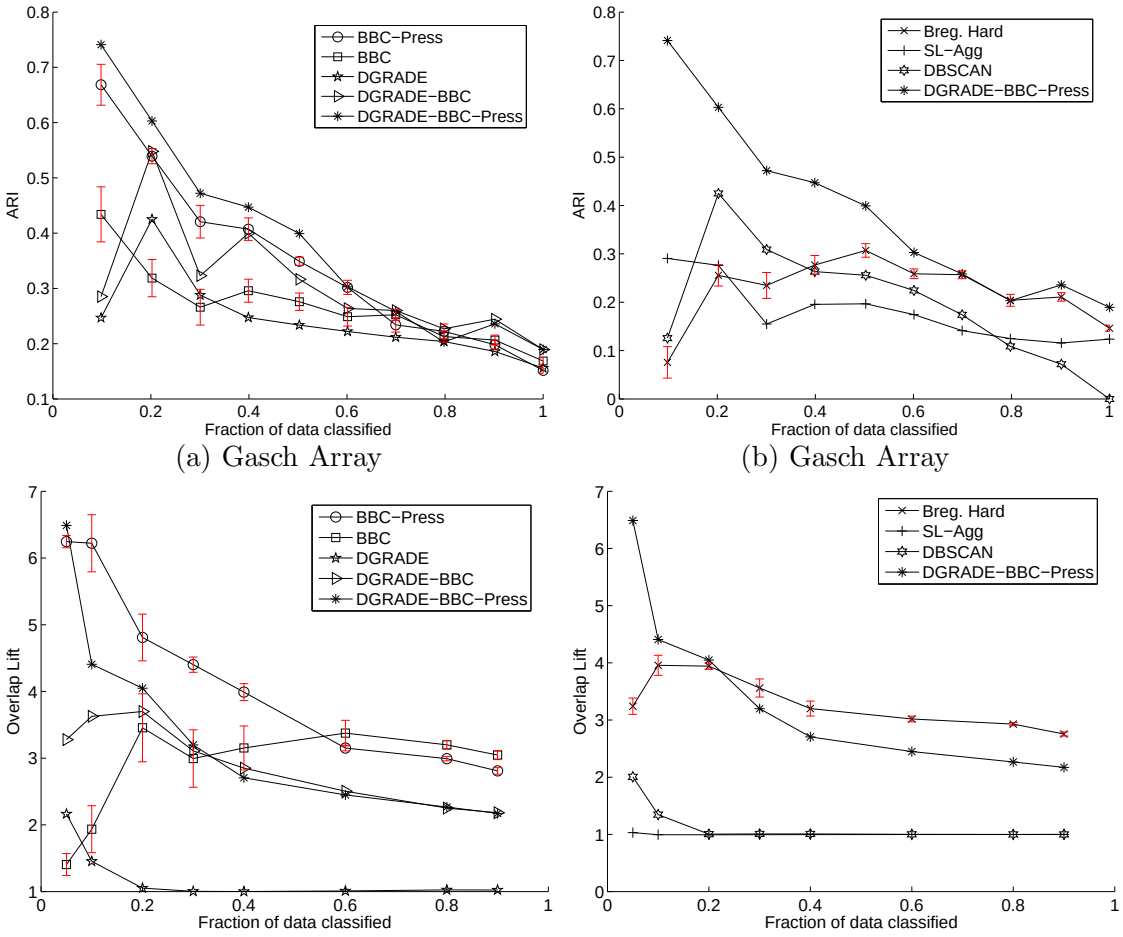


(a) Sim-2 (b) Sim-2

(c) Sim-10 (d) Sim-10

(e) Sim-40 (f) Sim-40

Figure 13: Evaluation of BBC seeded with DGRADE using the synthetic Gaussian datasets: as compared to BBC with seeding, and against three other methods. Error bars of one std. deviation are shown (but are sometimes too small to be visible) for local search methods for which ARI is plotted as the average over 100 trials with random initialization.



(a) Gasch Array (b) Gasch Array

(e) Lee (f) Lee

Figure 14: Evaluation of BBC seeded with DGRADE on two real datasets as compared to BBC without seeding, and against the three other methods. Performance was measured using ARI for Gasch Array and using Overlap Lift for the Lee dataset. Results for Bregman Hard clustering and BBC without seeding were averaged over 20 and 10 trials for Gasch and Lee respectively, and the corresponding one std. dev. error-bars are plotted (but are sometimes too small to be visible).

12 Concluding Remarks

First, we significantly improved upon existing One Class clustering work by incorporating a novel global search into an improved local search. The HOCC algorithm provides a global approximation with optimality bounds under certain conditions, while our local search BBOCC shows empirical improvement in the quality of the local minima compared to OC-IB. Hyper-BB as a combination of HOCC and BBOCC, inherits the desirable properties of the two components such as optimality bound and deterministic output, and the ability to take either the cluster size or cost threshold as input. Empirical results confirm that both HOCC and BBOCC are important components of Hyper-BB, and that Hyper-BB gives near-optimal solutions for finding a small, dense region.

Bregman Bubble Clustering (BBC) extends our One Class local search method to a more general setting of finding k dense regions. We also developed an alternative to seeding for improving local search called Pressurization. Empirical results show that when clustering only a (densest) fraction of the data, BBC-Press outperforms other potential alternatives by a large margin and gives good results on both low and high dimensional data, where such a bias is important. BBC-Press could therefore be seen as a powerful extension of One Class Clustering to a multi-class setting where the goal is to find a few most relevant clusters in the data. BBC extends the notion of "density-based clustering" to a large class of divergence measures, and is perhaps the first that uses a local search/parametric approach. The low time and space complexity of the local search approach, coupled with the robustness provided by Pressurization, makes it possible to find multiple dense regions on extremely large and high-dimensional datasets, thus opening up density-based clustering for applications to much larger problems.

Motivated by both BBOCC and HOCC, DGRADE provides a k-class equivalent of HOCC for seeding BBC. Empirical results show that DGRADE performs extremely well in combination with BBC, giving very high-quality results on many datasets that are competitive or better than BBC with Pressurization. Furthermore, DGRADE allows for automatically estimating the number of dense clusters in the data, and BBC seeded with DGRADE is ideal for applications such as bioinformatics where deterministic results are desirable. Also, for high-dimensional datasets, when DGRADE is further combined with Pressurization for BBC, i.e. DGRADE is used for seeding BBC-Press, the performance improves beyond that of either BBC-Press or DGRADE seeded BBC. Thus, the combination of the three components provides an extremely robust framework with several key properties in one method; robust, deterministic results, scalability to large, high-dimensional datasets, and applicability to a wide variety of problem domains.

Our local search framework, Bregman Bubble Clustering can also be thought of as a conceptual bridge between partitional clustering algorithms and the problem of One Class Clustering. The Soft BBC model we developed shows that BBC and its One Class equivalent special cases arise out of a more fundamental model involving a mixture of exponentials and a uniform background, and explains how, by incorporating a model for the "noisy" background, BBC performs better than Bregman Clustering when only some of the data should be clustered.

The extension of our methods to Pearson Correlation (Pearson Distance) makes our method applicable to a variety of biological datasets where finding small, dense clusters is critical, while the extension to Cosine Similarity makes it applicable to text clustering, where the goal of retrieving a few of the most relevant documents belonging to a category maps well to our

framework.

Acknowledgments: This research was supported by NSF grants IIS-0325116 and IIS-0307792. We are also grateful to Srujana Merugu for some useful discussions.

References

[ABKS99] M. Ankerst, M. Breunig, H. P. Kriegel, and J. Sander. Optics: Ordering points to identify the clustering structure. In *ACM SIGMOD Conference*, 1999.

[BGGM80] A. Buzo, A. H. Gray, R. M. Gray, and J. D. Markel. Speech coding based on vector quantization. *IEEE Transactions on Accoustics, Speech and Signal Processing*, 28(5):562–574, 1980.

[BL04] Arindam Banerjee and John Langford. An objective evaluation criterion for clustering. In *KDD-04*, Seattle, Washington, USA, August 2004.

[BMDG05] A. Banerjee, S. Merugu, I. Dhillon, and J. Ghosh. Clustering with Bregman divergences. *JMLR*, 6:1705–1749, 2005.

[CC04] Koby Crammer and Gal Chechik. A needle in a haystack: Local one-class optimization. In *In Proc. ICML*, Banff, Alberta, Canada, 2004.

[DM01] Inderjit S. Dhillon and Dharmendra S. Modha. Concept decompositions for large sparse text data using clustering. *Machine Learning*, 42(1-2):143–175, January-February 2001.

[EKSX96] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *In Proc. KDD-96*, 1996.

[Gas00] Gasch A. P. et al. Genomic expression programs in the response of yeast cells to environmental changes. *Mol. Bio. of the Cell*, 11(3):4241–4257, December 2000.

[GG01] Gunjan K. Gupta and Joydeep Ghosh. Detecting seasonal trends and cluster motion visualization for very high dimensional transactional data. In *Society for Industrial and Applied Mathematics (First International SIAM Conference on Data Mining (SDM2001))*, April 2001.

[GG05] Gunjan Gupta and Joydeep Ghosh. Robust one-class clustering using hybrid global and local search. In *Proc. ICML 2005*, pages 273–280, Bonn, Germany, August 2005.

[Gol03] Gollub J. et al. The stanford microarray database: data access and quality assessment tools. *Nucleic Acids Res*, 31:94–96, 2003.

[HA85] L. Hubert and P. Arabie. Comparing partitions. *Journal of classification*, pages 193–218, 1985.

[Has00] Hastie T. et al. Gene shaving as a method for identifying distinct sets of genes with similar expression patterns. *Genome Biology*, 1:1–21, 2000.

[JD88] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice Hall, Englewood Cliffs, NJ, USA, 1988.

[JPZ03] Daxin Jiang, Jian Pei, and Aidong Zhang. DHC: A density-based hierarchical clustering method for time series gene expression data. In *BIBE '03*, page 393, Washington, DC, USA, 2003. IEEE Comp. Soc.

[KMN97] Michael Kearns, Yishay Mansour, and Andrew Y. Ng. An information-theoretic analysis of hard and soft assignment methods for clustering. In *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence*, pages 282–293. AAAI, 1997.

[KPPS03] Hans-Peter Kriegel, Martin Pfeifle, Marco Pötke, and Thomas Seidl. The paradigm of relational indexing: A survey. In *BTW*, volume 26 of *LNI*. GI, 2003.

[Lan95] Ken Lang. Newsweeder: Learning to filter netnews. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 331–339, 1995.

[LBG80] Y. Linde, A. Buzo, and R. M. Gray. An algorithm for vector quantizer design. *IEEE Transactions on Communications*, 28(1):84–95, 1980.

[LDAM04] I. Lee, S. V. Date, A. T. Adai, and E. M. Marcotte. A probabilistic functional network of yeast genes. *Science*, 306:1555–1558, 2004.

[LO02] L. Lazzeroni and A. B. Owen. Plaid models for gene expression data. *Statistica Sinica*, 12(1):61–86, January 2002.

[MK97] Geoffrey J. McLachlan and Thriyambakam Krishnan. *The EM Algorithm and Extensions*. John Wiley & Sons, Inc, 1997.

[MTea04] Robert Mansson, Panagiotis Tsapogas, and Mikael Akerlund et al. Pearson correlation analysis of microarray data allows for the identification of genetic targets for early b-cell factor. *J. Biol. Chem.*, 279(17):17905–17913, April 2004.

[NH98] R. Neal and G. Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. In M. I. Jordan, editor, *Learning in Graphical Models*. Kluwer, 1998.

[PPL01] S. D. Pietra, V. D. Pietra, and J. Lafferty. Duality and auxiliary functions for Bregman distances. In *Technical Report CMU-CS-01-109, School of Computer Science*, Carnegie Mellon University, 2001.

[SBV95] B. Schölkopf, C. Burges, and V. Vapnik. Extracting support data for a given task. In *KDD*, Menlo Park, CA, 1995. AAAI Press.

[SG03] Alexander Strehl and Joydeep Ghosh. Relationship-based clustering and visualization for high-dimensional data mining. *INFORMS Journal on Computing*, 15(2):208–230, 2003.

[SGM00] Alexander Strehl, Joydeep Ghosh, and Raymond J. Mooney. Impact of similarity measures on web-page clustering. In *AAAI Workshop on AI for Web Search (AAAI 2000)*, pages 58–64. AAAI/MIT Press, July 2000.

[Slo02] Noam Slonim. The information bottleneck: Theory and applications. In *Doctoral dissertation, The Hebrew University*, 2002.

[SPST+01] B. Schölkopf, J. C. Platt, J. S. Shawe-Taylor, A. J. Smola, and R. C. Williamson. Estimating the support of a high-dimensional distribution. *Neural Computation*, 13(7):1443–1471, 2001.

[SS00] R. Sharan and R. Shamir. Click: A clustering algorithm with applications to gene expression analysis. *Proc. 8th ISMB*, pages 307–316, 2000.

[TD99] D. Tax and R. Duin. Data domain description using support vectors. In *Proceedings of the ESANN-99*, pages 251–256, 1999.

[TdSL00] J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290:2319–2323, 2000.

[UN98] N. Ueda and R. Nakano. Deterministic annealing EM algorithm. *Neural Networks*, 11(2):271–282, 1998.

A Fast update of Q_{one} in OC-IB

In this appendix we revisit the OC-IB algorithm introduced in Section 2.3 and examine the centroid update at each iteration in more detail to highlight the difference between OC-IB and BBOCC. One randomized iteration of OC-IB involves n updates of the centroid and the corresponding cost Q_{one}. Since the centroid is the mean of the member points, it can be updated quickly for every addition or deletion of a point. However, the n re-estimations of Q_{one} as the average Bregman divergence of all the member points to the corresponding n updated centroids could be quite slow. Fortunately, OC-IB uses a much faster update originally derived by [Slo02] using the principle of Information Bottleneck, but which can also be derived from the following lemma:

Let $\mathcal{G}_1 \subset \mathcal{X}$ be a set of data points, and the corresponding mean be μ_1. Furthermore, let μ_2 be the mean of set $\mathcal{G}_2 = \mathcal{G}_1 \cup \mathbf{x}_{new}$. Then, the following follows directly from Lemma 10.3:

$$Q_{one}(\mathcal{G}_1, \mu_2) = Q_{one}(\mathcal{G}_1, \mu_1) + D_\phi(\mu_1, \mu_2), \tag{24}$$

Therefore, $Q_{one}(\mathcal{G}_2, \mu_2)$ can be estimated as:

$$Q_{one}(\mathcal{G}_2, \mu_2) = Q_{one}(\mathcal{G}_1, \mu_1) + D_\phi(\mu_1, \mu_2) + D_\phi(\mathbf{x}_{new}, \mu_2) \tag{25}$$

Thus, Equation 25 enables update of the cost Q_{one} in OC-IB in constant time when a new point $\mathbf{x}_{new}$ is added, instead of having to re-estimate Q_{one} over all the points in $\mathcal{G}_2$. A similar constant time update is possible if a point is deleted from $\mathcal{G}$.

Note 1: The cost Q_{one} estimated by OC-IB after each center update using Equation 25 actually corresponds to the Bregman Information of the set $\mathcal{G}$, and at convergence OC-IB finds

a Bregmanian ball enclosing a locally maximum of points with Bregman Information less than q_{max}.

Note 2: Such an update will not work with either Pearson Distance and (1−Cosine Similarity) described in Section 9, since they are "convex projections" of Squared Euclidean distance, not directly a Bregman divergence. Thus, the standard OC-IB algorithm will not work on these measures, and a modified version that recomputes the cluster cost from scratch would be considerably slower for these two measures that are important for biological and textual datasets respectively.

B Proofs for BBOCC-Q

Following is the proof for Proposition 3.1 in Section 3.1:

Proof. Let us add points to $\mathcal{G}$ in the order of increasing distance from $\mathbf{c}$ until adding one more point would cause $Q_{one}(\mathcal{G}, \mathbf{c})$ to become greater than q_{max}. Such a set forms a Bregmanian ball enclosing $s = |\mathcal{G}|$ data points and is defined by the centroid $\mathbf{c}$ and a radius equal to the distance from $\mathbf{c}$ of the last point added to $\mathcal{G}$. Now, it can be observed that deleting one point in $\mathcal{G}$ and adding another point from outside the Bregmanian ball, i.e. from set $\mathcal{X}/\mathcal{G}$ cannot decrease the cost Q_{one} but can increase it. This follows from the fact that all the points in set $\mathcal{X}/\mathcal{G}$ are at a distance from $\mathbf{c} \geq$ the distance of the point in $\mathcal{G}$ farthest from $\mathbf{c}$. Furthermore, adding the point closest to $\mathbf{c}$ in set $\mathcal{X}/\mathcal{G}$ causes the average cost Q_{one} to be more than q_{max}. Therefore, the same would happen if any other points in $\mathcal{X}/\mathcal{G}$ is added to $\mathcal{G}$. Hence it is not possible to obtain a set larger than s with cost less than q_{max}. □

Following is the proof for Proposition 3.2 in Section 3.1:

Proof. From Proposition 3.1 and Theorem 2.1, it is straightforward to show that after each step of Algorithm 1, the number of points in cluster $\mathcal{G}$, cannot decline but can increase, and that eventually this leads to convergence. Let $\mathbf{c}_1$ and $\mathbf{c}_2$ be the two centroids before and after the centroid update in some iteration, and $\mathcal{G}_1$ be the set of points assigned to the cluster before the centroid update. Let $q_1 = Q_{one}(\mathcal{G}_1, \mathbf{c}_1)$ and $q_2 = Q_{one}(\mathcal{G}_1, \mathbf{c}_2)$ be the costs of $\mathcal{G}_1$ before and after the centroid update respectively. Then, $q_2 \leq q_1$ because of Theorem 2.1. Furthermore, because of Proposition 3.1, if we now reassign closest s_1 points and get a new cluster set $\mathcal{G}_2$, then $q_3 = Q_{one}(\mathcal{G}_2, \mathbf{c}_2) \leq Q_{one}(\mathcal{G}_1, \mathbf{c}_2)$ because of Proposition 3.1. Hence, $q_3 \leq q_2 \leq q_1 < q_{max}$. Therefore, potentially more points can now be added to $\mathcal{G}_2$ making it larger than s_1 while keeping the cost below q_{max}. Therefore, the size of the cluster $\mathcal{G}$ can potentially increase after each iteration of Algorithm 1, but cannot decline. If the size does not change, but the actual cluster members do change in an iteration, the centroid $\mathbf{c}$ can still drift and eventually lead to a larger cluster in some later iteration [20]. However, in the event that the size of the $\mathcal{G}$ does not change *and* the member points also remain unchanged, the algorithm converges (line 19). □

[20] An implementation detail: line 19 of Algorithm 1 also checks to see if the cluster size has not changed in two iterations. This prevents a rare but possible oscillation between two solutions of identical size and detects convergence.