Bregman Bubble Clustering: A Robust Framework for Mining Dense Clusters

Joydeep Ghosh[1] and Gunjan Gupta[2]

[1] Department of Electrical & Computer Engineering, The University of Texas at Austin, Austin, TX 78712, USA, ghosh@ece.utexas.edu
[2] Microsoft, One Microsoft Way, Redmond, WA 98052, USA, gunjang@microsoft.com

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 data and do not scale well to large, high-dimensional datasets. Also, they use a specific notion of "distance", typically Euclidean or Mahalanobis distance, which further limits their applicability. On the other hand, the recent One Class Information Bottleneck (OC-IB) method is fast and works on a large class of distortion measures known as Bregman Divergences, but can only find a *single* dense region. This paper presents a broad framework for finding k dense clusters while ignoring the rest of the data. It includes a seeding algorithm that can automatically determine a suitable value for k. When k is forced to 1, our method gives rise to an improved version of OC-IB with optimality guarantees. We provide a generative model that yields 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; a hard model corresponding to k exponential mixtures with a uniform background results in a set of k dense clusters. The proposed method describes a highly scalable algorithm for finding multiple dense regions that works with any Bregman Divergence, thus extending density based clustering to a variety of non-euclidean problems not addressable by earlier methods. We present empirical results on three artificial, two microarray and one text dataset to show the relevance and effectiveness of our methods.

1 Introduction

Clustering, which involves dividing data into groups of similar objects, is an important unsupervised learning problem that has been extensively applied in various domains [36], and a variety of hierarchical [39, 38, 29, 32], partitional [3, 17, 51, 48, 8, 41], graphical [34, 50, 61, 32, 25, 41, 42] and overlapping [4, 6, 33, 2] clustering algorithms have been proposed and have found applications in a wide va-

riety of domains such as identifying customer and product groups using market-basket data [71, 63, 31, 32, 25], document/text categorization [17, 18, 4, 72], and identifying functional groupings of genes and proteins in bioinformatics [38, 34, 29, 61, 6, 33].

In classical clustering, each data point either fully belongs to one cluster or is softly assigned to multiple clusters. However, in certain real-world problems, natural groupings are found among only on a small subset of the data, while the rest of the data shows little or no clustering tendencies. In such situations it is often more important to cluster a small subset of the data very well, rather than optimizing a clustering criterion over all the data points, particularly in application scenarios where a large amount of noisy data is encountered.

For example, consider a large, high-dimensional transactional market-basket data that consists of product purchase records of a large number of customers of a retail chain, gathered over a considerable period of time. For a multitude of reasons, including rapid growth of the customer base and product offerings, rapid evolution of the product catalog, and customer churn/inactivity, such data can be very sparse, with the majority of the customers buying only a very small set of products from a catalog of thousands of items [49, 30]. Such a dataset can be used for clustering either products (with customers as features) or customers (with products as features). Clustering on such data is useful in many applications including product recommendations, customer and product segmentation, and identifying various customer and market trends. However, typically only a small subset of customers show statistically significant coherent buying behavior and that too when one focuses only a small subset of products [64, 14, 69]. Therefore, a clustering algorithm for such datasets should have the ability to prune out (potentially large) sparse and noisy portions of the data to uncover the highly coherent clusters. There are also non-algorithmic reasons for desiring such a capability, e.g. the marketing department of a retailer might want to target only a fraction of the customers and ignore the rest so as to maximize the ROI of their usually limited budgets. Another reason for desiring higher accuracy in such models at the cost of less coverage could be to minimize the chance targeting of a customer with the wrong product, which can negatively impact a customer's shopping experience.

As a second example, consider microarray datasets that record the relative expression levels of a few thousand genes across multiple experimental conditions. The conditions typically cover only a specific "theme" such as stress-response, and therefore only a few genes that are related to the conditions show good clustering. Biologists are interested in identifying small groups of genes that show strongly correlated expression patterns, as they indicate common participation in biological processes that are involved in the specific context. For example, for the Gasch dataset [21], which consists of only stress response experiments and is a popular benchmark for clustering microarray data, according to the authors, 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 in order to better identify and charac-

terize the genes that are actually involved in specific types of stress responses. Similar characteristics have been observed and exploited in other microarray as well as protein mass spectroscopy and phylogenetic profile datasets [33, 37, 16, 53], where available features are often focused towards a few important contexts that are suitable for resolving only a small number of well-defined genetic pathways.

Finally, consider the grouping of documents according to their relevance to certain search queries. Certain documents are not relevant for any of the queries of interest. Moreover, the user is often interested in finding the top few matches for a broad-topic query rather than all possible matches, so that a system that returns a small number of highly relevant documents might be preferable over the one that returns hundreds of somewhat relevant documents, i.e., precision is more important than recall. By pruning out irrelevant or less relevant documents, precision can be improved without compromising much on recall [12].

One way to handle such scenarios is to prune the large fraction of "don't care" data as a preprocessing 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 relevant subset. A more natural approach would involve finding the multiple dense regions and the "don't care" set simultaneously. Specifically, one desires clustering algorithms that are (1) scalable, (2) can cluster only a specifiable fraction of the whole dataset, (3) find multiple clusters, and (4) can work with a wide variety of data types. Existing density-based methods such as DBSCAN [20] naturally cluster only a subset of the data but are not suitable for many such situations because of implicit metric assumptions, and are not scalable to very large problems since they either require an in-memory $O(n^2)$ distance matrix, or an efficient index [7, 44] [3]. In contrast, the One Class Information Bottleneck (OC-IB) [12] provides a local search based approach for finding a single dense region in the data that is fast and 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 minimum and does not allow control over the size of the cluster returned, which can vary greatly depending upon the quality of the local minimum. A subsequent technique called BBOCC enhanced the capabilities of the OC-IB type approach by providing the ability to control the size of the resultant cluster, as well as to use Pearson Correlation and Cosine similarity, in addition to Bregman Divergences [28]. This expanded the applicability of BBOCC to many

[3] DBSCAN finds small dense regions of points by connecting nearest neighbor dense points. DBSCAN (and its derivatives) requires an efficient database index to be scalable to large data sets, since it uses the indexes to find the nearest neighbors. However, such indexes are usually efficient for only low-dimensional datasets.

types of biological and textual clustering problems; however the limitation of identifying only a single cluster remained.

This paper substantially generalizes the single-cluster approach of BBOCC while retaining its key desirable properties, resulting in a robust and scalable framework for finding multiple dense clusters. Our main contributions are as follows:

1. We present a generalization of BBOCC called Bregman Bubble Clustering (BBC) that can simultaneously find k dense clusters. BBC inherits $O(nd)$ time and space complexity of BBOCC for each iteration and is scalable to much larger and higher-dimensional datasets than existing density-based methods. It also goes beyond Euclidean distance centric density-based clustering, and is applicable to all Bregman Divergences. This extension allows the method to be relevant for a very wide class of data properties (and corresponding loss functions) while retaining the simplicity of the squared-loss solution.
2. We develop a generative (soft) 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. BBC and many existing clustering algorithms are shown to be special cases of this model. Our main contribution here was to show how the seemingly distinct problem of finding dense clusters could be viewed as arising out of a generalization of the well-known mixture of exponential distributions model. This relationship also shows (1) how the problem setup of BBC is not just a convenient heuristic, but arises as a special (hard) case of a much more fundamental generative model, and (2) how BBC relates to the partitional clustering problem (that involves *all* data points) at a fundamental level.
3. We introduce a mechanism called *Pressurization* that substantially improves the quality of the local search in BBC 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 robust to initialization and have very small variations in quality over multiple trials.
4. For medium-sized problems, we describe a deterministic seeding algorithm for BBC called Density Gradient Enumeration (DGRADE). At the cost of somewhat increased time and space complexity, DGRADE gives good empirical results when seeding BBC, and can determine k automatically. DGRADE uses a novel "density gradient estimation" at all the data points to identify all the distinct dense regions in the data, which then allows it to automatically estimate the best k, and the corresponding k cluster seeds. For many problems such as clustering gene-expression datasets where the number of relevant clusters in a dataset are often unknown initially and vary greatly, the more expensive time complexity of the seeding method as compared to Pressurization provides a useful trade-off; it provides a meaningful seeding

algorithm for BBC in a completely unsupervised setting. It also makes the BBC results deterministic, a desirable property for discovering deterministic albeit unknown biochemical pathways in an organism. Moreover, DGRADE can be used in conjunction with Pressurization for further improving clustering quality while also determining k.

5. We performed evaluations 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. We performed two types of experiments: (a) three artificial Gaussian datasets of 2, 10 and 40 dimensions were used to show the stability of observed results to the increasing dimensionality of the data, keeping the number and type of clusters relatively similar, and (b) pertinence to real-life applications was demonstrated using three different types of problems: using microarray data to cluster genes (medium size, high dimensional), clustering of conditions/experiments from microarray data (small, very high dimensional), and text clustering (large, very high dimensional).

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 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 $\text{int}(S)$, the interior of S. The Bregman Divergence $D_\phi : S \times \text{int}(S) \mapsto [0, \text{inf})$ 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 [56, 3].

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

Theorem 21 [3]: *Let X be a random variable taking values in $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^n \subset C \subseteq \mathbb{R}^d$ (C is convex) following a probability measure ν [4], and let $E[\,]$ denote the expectation operator. 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 $\mathbf{c}^* = \mu = E_\nu[X]$.

[3] 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. Theorem 21 essentially states that given any set of data points, the mean vector (or more generally, the *expectation* given a probability measure defined over the points) is the best single representative of the set in the sense of minimizing the average loss when each point gets replaced by a common representative. This result is well known for squared loss, but as per this Theorem it holds true for all Bregman Divergences. An immediate implication is that the k-means type algorithm will have the same guarantee of convergence to a local minima of the cost function for any Bregman Divergence. This result is also used in the BBC algorithm formulated later in Section 3.

Bregman Hard Clustering: [3] describe a partitional clustering algorithm called *Bregman Hard Clustering* that exploits Theorem 21. Starting with a random initialization of k centers Bregman Hard Clustering repeats the following until convergence to a local minimum: (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. An important result from [3] was to prove the bijection that a K-Means type algorithm exists for any Bregman Divergence, and only for Bregman Divergences. 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, proved and applied as independent algorithms, such as the celebrated Linde-Buzo-Gray algorithm [48, 8], before Bregman Hard Clustering was formulated.

2.2 Density-based and Mode Seeking Approaches to Clustering

A variety of non-parametric 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. The classic work in this area is Wishart's mode analysis [70], which is closely related to the more recent DBSCAN algorithm [20]. Other notable works include the application of mean-shift algorithm to clustering [11, 22].

[4] Unless stated explicitly otherwise, we assume all points to have the same weight, i.e., ν is a uniform measure.

The mean-shift algorithm performs (adaptive) gradient ascent on the estimated density of the data, as obtained by convolving a suitable localized kernel function with the raw data, to find modes or local peaks of the density. If only modes that are sufficiently dominant are selected, then points attracted to less important modes could be discarded. By varying the widths of the kernels, one can investigate clustering behavior at different scales [10]. DBSCAN has a slightly different flavor: 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 [1] proposed a visualization to make it easier to select these two parameters. Like other mode-seeking algorithms, 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 [38] 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 [67] within the embedded space.

We have recently proposed a non-parametric approach, Auto-HDS [29] for detecting a few dense clusters in data. Inspired by Wishart's work but much more computationally efficient, Auto-HDS simultaneously detects clusters at multiple resolutions and provides a powerful visualization mechanism for cluster exploration. It also provides much superior results as compared to DBSCAN. Since this paper focuses on parametric approaches, we do not discuss Auto-HDS further, but point the interested reader to Chapter 11 of [26], which provides a detailed theoretical as well as empirical comparison of Auto-HDS with BBC. In summary, BBC is more scalable than Auto-HDS and other non-parametric approaches; and works better when one has a fairly good generative model of the data. However, if one has little idea about the nature of the data, or if the data has very odd-shaped dense regions at different resolutions, the additional flexibility of a non-parametric approach is helpful.

A parametric approach wherein a mixture of Gaussians plus a uniform background component is fitted to data in order to detect peaks was recently presented in [57]. The current approach and accompanying software is specifically for detecting peaks in one-dimensional data, with additional constraints such as no other peak allowed within a certain distance to the left or right of a given peak. This constrained one-dimensional setting is designed for specific applications such as detecting transcription start sites from gene annotation data. If one properly generalizes this approach to multivariate data and to all exponen-

tial family mixture models, one will obtain the soft BBC model, which the new method proposed in this paper compares favorably against (see Section 10).

2.3 Iterative relocation algorithms for finding a single dense region

Traditional density-based clustering algorithms (Section 2.2) were aimed at low-dimensional, spatial datasets. However, they have two major shortcomings for broader clustering applications: (1) they typically rely on a Euclidean distance type metric to determine distance, even though such measures are not suitable for many datasets and (2) they do not scale well to large, higher-dimensional datasets. A recently proposed algorithm for finding a *single* dense region[5] called One Class Information Bottleneck (OC-IB) [12] breaks these two barriers by proposing an iterative relocation based approach that is also generalizable to all Bregman Divergences, and whose scaling properties are akin to K-Means even though K-Means itself is not designed for finding dense clusters.

OC-IB uses the notion of a Bregmanian ball to find a single, locally dense region. Earlier approaches to One Class Clustering [66, 58, 59, 13] used convex cost functions for finding large-scale structures, or correspondingly, for finding a small number of outliers. However, [12] 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 embedded within a relatively uniform background. Previously proposed convex One Class methods end up finding a solution centered in-between the two Gaussians. In contrast, OC-IB is able to find one of the two Gaussians, and could be applied sequentially to recover both. More discussion and evidence on this important conceptual difference between the two One Class approaches and why the local approach used by OC-IB is more relevant for finding dense regions can be found in [12].

In an earlier paper [27], we described an algorithm called Batch Ball One Class Clustering (BBOCC) that provides several improvements over OC-IB, including the ability to control the size (number of data points) of the dense cluster, improved quality of local search, optimality guarantee using seeding [6] and extension to Pearson Correlation. However, BBOCC can also only find a single dense region. An obvious solution that comes to mind is to apply OC-IB or BBOCC sequentially: removing points belonging to the first dense cluster and then running the One Class algorithm again on the reduced data. Unfortunately, unless a correspondence problem is solved, this could result in a "cookie-cutter" clustering where additional clusters found are comprised of left-over dense points

[5] A problem that is also referred to as One Class Classification or Clustering.

[6] Guarantees that the solution is within two times of lowest possible cost (as given by Equation 2, only applicable for $k = 1$, since k is always 1 for BBOCC), when the Bregman Divergence was Squared Euclidean, and is constant times optimal for other Bregman Divergences. See [28] for more details.

surrounding the hole created by the removal of the first dense cluster discovered. This limitation was also hinted upon in the conclusion section of [12] [7].

This paper addresses the problem of simultaneously finding k dense regions in the data while ignoring a specified fraction of the data-points. While the approach taken is not a straightforward generalization of BBOCC and entails several new concepts, familiarity with BBOCC [27] will provide an enriched understanding of this paper.

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 [33] 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 ($\Omega(n^3)$). Other greedy methods such as Plaid [46] 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 (biological experiment) conditions 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 conditions as relevant.

3 Bregman Bubble Clustering

In this section we first generalize the notion of a single dense *Bregmanian Ball* (used by One Class algorithms OC-IB and BBOCC) 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.

3.1 Cost Function

Let $\mathcal{X} = \{\mathbf{x}\}_{i=1}^{n} \subset C \subseteq \mathbb{R}^d$ (where C is convex) 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

[7] Interestingly, the seeding algorithm DGRADE presented in this paper in Section 9, solves exactly this correspondence problem by identifying all the distinct "basins of attraction" corresponding to the densest Bregmanian balls in the data.

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:

$$Q_b(\mathcal{G}, \{\mathbf{c_j}\}_{j=1}^{k}) = \frac{1}{|\mathcal{G}|} \sum_{j=1}^{k} \sum_{i:\mathbf{x}_i \in \mathcal{C}_j}^{|\mathcal{C}_j|} D_\phi(\mathbf{x}_i, \mathbf{c}_j), \tag{2}$$

3.2 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:

Definition 1*: Find the clustering $\mathcal{G}$ with smallest cost Q_b such that $|\mathcal{G}| = s$.*

Definition 1 builds upon the cost formulation stated in 3.1, where the cost contributed by each point in each cluster is proportional to the distance of the member points from their cluster centroid, with an additional constraint that exactly s points are clustered. For $k = 1$, this problem definition reduces to one used in one of the two form of BBOCC for finding a single dense cluster [8].

3.3 Bregmanian Balls & Bregman Bubbles

A *Bregmanian ball* [12] $B_\phi(r, \mathbf{c})$ 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. 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 enclosed by it.

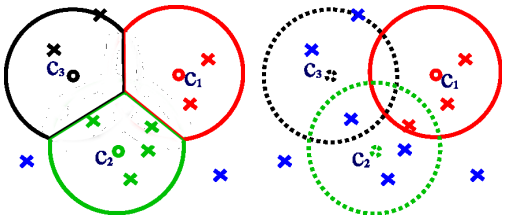


Fig. 1. An illustration showing (a) three Bregman bubbles, and (b) a Bregmanian ball (solid line), and two other possible balls (dotted lines). The union of the points enclosed by the three possible balls in (b) is the same as the set of points enclosed by the three bubbles.

For a specified set of k cluster representatives, and a fixed s, it can be shown using Theorem 21 [9] that the clustering that minimizes Q consists of: (1) the

[8] Section 6 discusses the connection with BBOCC in more detail.
[9] A more formal proof is presented after Proposition 31.

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 they are either (1) pure Bregmanian balls of radius $r \leq r_{max}$, or (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[10] that result from assigning points to the 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_b 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.

3.4 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 minimum exists. *Bregman Bubble Clustering-S* (BBC-S, Algorithm 1) 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) the selection phase, where points are selected in ascending order of their distance from their corresponding cluster representative, until s points are picked, and (3) the update step, where cluster "means" are re-estimated and updated. 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 6. Stages 1, 2 and 3 are repeated until there is no change in assignment between two iterations - i.e. the algorithm converges. Algorithm 1 describes a more detailed implementation of BBC-S where line number 11 represents Stage 1, lines 16 to 20 map to Stage 2, while lines 26-28 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 31 *Algorithm 1 terminates in a finite number of steps at a locally optimal solution, i.e., the cost function Q_b cannot be decreased by (a)the assign-*

[10] This can be shown to be true for all Bregman Divergences [3].

ment step, (b) the data selection step or (c)changing the means of any existing clusters.

Proof: The local optimality of steps (a) and (c) has been established in [3] Prop. 3, and can be summarized as follows: local optimality of step (a) can be readily shown by the contradiction that if a point is not assigned to the nearest centroid, then the total cost Q_b can be decreased by assigning it to a closer centroid. Theorem 21 guarantees that step (c) is locally optimal; a representative other than the mean would lead to a higher cost for a given cluster. Local optimality of step (b) can also be shown by contradiction - if a point x_p that is not among the first s points (in the sorted order at line 14 of the algorithm) was part of the optimal solution, then the cost Q_b could be decreased by replacing this point with a point within the first s points that is not picked. Thus no such x_p can be part of the best solution at step (b). So the algorithm monotonically decreases the objective function value, while the number of distinct clusterings is finite, thus assuring convergence in a finite number of steps.

If heap-sort is used at line 14 of Algorithm 1, then each iteration of BBC-S takes
$O(max(nkd, s\log(n)))$ time, making it quite fast.

3.5 BBC-Q: dual formulation of Bregman Bubble Clustering with fixed cost

An alternative dual 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:

Definition 2*: 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 1 and 2 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 possible cost q_{max}. Algorithm 1 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}. The proof of convergence for BBC-Q follows along similar lines as that for Proposition 31.

The seemingly minor difference between BBC-S and BBC-Q results in two very different algorithms. For a fixed s as input (BBC-S), for iterations in sparse regions the bubbles expand until s points are covered. As the bubbles move into denser regions, their radii shrink. BBC-Q does not have this property and generally gives worse performance when the bubbles are small [28]. 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.

4 Soft Bregman Bubble Clustering (Soft BBC)

4.1 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 sum of the assignment weights of a given point to 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. [3] 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*, and showed that there is a *bijection* between this family and regular Bregman Divergences. This bijection is expressed by:

$$p_{(\psi,\theta)}(\mathbf{x_s}) = exp(-\beta D_\phi(\mathbf{x_s}, \mu)) f_\phi(\mathbf{x_s}) \tag{3}$$

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 $\mathbf{x}$.

Well-known examples of regular Bregman Divergences (and the corresponding exponential distribution) include squared Euclidean Distance (Gaussian distribution), KL-divergence (multinomial distribution) and Itakura-Saito distance [48, 8].

[3] 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 3.

4.2 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 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.

4.3 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:

$$p(\mathbf{x_i}) = \sum_{j=1}^{k} \alpha_j p_{(\psi,\theta)}(\mathbf{x}_i|\theta_j) + \alpha_0 p_0, [i]_1^n \tag{4}$$

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:

$$L(\Theta|\mathcal{X}) = \sum_{i=1}^{n} \log(\sum_{j=1}^{k} \alpha_j p_{(\psi,\theta)}(\mathbf{x}_i|\theta_j) + \alpha_0 p_0) \tag{5}$$

where Θ denotes all the parameters (priors and mixture component parameters). Maximizing the above data likelihood is a natural approach for 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.

4.4 Soft BBC EM Algorithm

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 5, 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. Therefore, the only parameters we can 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 [55]:

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

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 maximum $\tilde{P}^*$ and Θ^*. Furthermore, it can be shown that a local maximum of $F(\tilde{P}, \Theta)$ leads to a local maximum on the original likelihood given by Equation 5. 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 6) 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 for the n equality constraints, taking derivatives w.r.t. $\tilde{p}(Y_i = j|\mathbf{x}_i)$, and then eliminating the Lagrange multipliers, we obtain:

$$\tilde{p}(Y_i = j|\mathbf{x}_i)^* = \frac{\alpha_j p_{(\psi,\theta)}(\mathbf{x_i}|\theta_j)}{\sum_{j=1}^{k} \alpha_j p_{(\psi,\theta)}(\mathbf{x}_i|\theta_j) + \alpha_0 p_0}, 1 \leq j \leq k \tag{6}$$

$$= \frac{\alpha_0 p_0}{\sum_{j=1}^{k} \alpha_j p_{(\psi,\theta)}(\mathbf{x}_i|\theta_j) + \alpha_0 p_0}, j = 0 \tag{7}$$

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. On applying the standard Lagrange procedure, one obtains:

$$\alpha_j^* = \frac{\sum_{i=1}^{n} \tilde{p}(Y_i = j|\mathbf{x}_i)}{n}, [j]_0^k \tag{8}$$

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:

$$\sum_{i=1}^{n} \tilde{p}(Y_i = j|\mathbf{x_i}) \nabla_{\theta_j} p_{(\psi,\theta)}(\mathbf{x}_i|\theta_j) = 0 \tag{9}$$

This results in the update equation for the exponential distribution mixtures $\{\theta\}_{j=1}^{k}$ as the weighted average of $\mathbf{x}$ [3]:

$$\theta_j = \frac{\sum_{i=1}^{n} p(Y_i = j|\mathbf{x}_i)\mathbf{x}_i}{\sum_{i=1}^{n} p(Y_i = j|\mathbf{x}_i)} \tag{10}$$

An example of re-estimation of mixture component parameters for Gaussians is described in more detail in Section 7.

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:

$$\alpha_j^* = (1 - \alpha_0)\frac{\sum_{i=1}^{n} \tilde{p}(Y_i = j|\mathbf{x}_i)}{\sum_{j=1}^{k}\sum_{i=1}^{n} \tilde{p}(Y_i = j|\mathbf{x}_i)} \tag{11}$$

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

4.5 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):

$$\alpha_0 = \frac{1}{n}\sum_{i=1}^{n} \frac{\alpha_0 p_0}{\sum_{j=1}^{k} \alpha_j p_{(\psi,\theta)}(\mathbf{x}_i|\theta_j) + \alpha_0 p_0} \tag{12}$$

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. For a given value of α_0, one approach would be to rewrite Equation 12 as an optimization problem and solve for the best value of p_0:

$$f(p_0) = \alpha_0 - \frac{1}{n}\sum_{i=1}^{n} \frac{\alpha_0 p_0}{\sum_{j=1}^{k} \alpha_j p_{(\psi,\theta)}(\mathbf{x}_i|\theta_j) + \alpha_0 p_0} = 0 \tag{13}$$

Written in this form, one could now start with a seed value between 0 and 1, and then search for the value of p_0 that brings $f(p_0)$ closest to 0. An optimization routine such as Matlab *fsolve* (http://www.mathworks.com) could be used for this process. However, a faster approximation of p_0 can be obtained as follows:

1. Perform the first E step (equations 6 and 7).
2. Compute the $p_{max}^i = \max_{j=0}^{k}(p(Y_i = j|\mathbf{x}_i))$ for each $\mathbf{x}_i$.
3. Pick p_0 as the s^{th} largest value in $p_{max}^i[i]_1^n$ where $s = \lceil \alpha_0 n \rceil$.

The above formulation works well because of the following reason: the final soft BBC results are probabilistic, with each point having a probability of belonging to either one of the k clusters, or the background. The probabilities need to be converted into hard assignment to obtain clustering needed in many applications. Later in Section 6, we show that the natural hard assignment corresponds to assigning each point to the mixture with the maximum posterior probability (which could be either of the k clusters or the uniform background). In other words, selecting label j such that $p(Y_i = j|\mathbf{x}_i) = p^i_{max}$ using Step 2 above. For the hard assignment case, since the fraction of data points clustered is s, and since $s = \lceil \alpha_0 n \rceil$, picking p_0 equal to (or in theory, slightly larger than) the s^{th} largest value in $p^i_{max}[i]^n_1$ would result in very close to s points getting assigned to the clusters, while the remaining, having a $p^i_{max} \leq p_0$ get assigned to the background cluster. In practice, the value of p_0 obtained using this approach corresponds closely with the value computed using the more expensive optimization type approach.

The following enhancement works even better in practice: an initial estimate of p_0 is computed using the approach described above using the seed cluster parameters to Soft BBC ($\{\theta_j\}^k_{j=1}$ and $\{\alpha_j\}^k_{j=1}$). This initial value of p_0 is then used to run Soft BBC to convergence. The clustering parameters obtained at convergence are then used to compute p_0 again, and Soft BBC is then run to convergence a second time using the new p_0. This second p_0 estimate is better since it is based on parameters that are closer to the convergence values.

5 Improving local search: Pressurization

5.1 Bregman Bubble Pressure

We first introduce a concept called *Bregman bubble pressure* that has properties analogous to that of pressure around air bubbles rising from a seabed; as air bubbles rise in a column of water, the hydrostatic pressure outside drops, and the bubbles expand (see [40], Chapter 10, Section 10.4.4 for a description of this phenomena).

In the case of the generative or Soft BBC model, we can think of this external pressure as being driven by the relative weight of the background distribution, α_0 in Equation 4. Bregman bubble pressure can be seen as being proportional to the ratio of the background distribution weight vs. the weight of all the k bubbles combined, i.e. $\alpha_0 / \sum^k_{j=1} \alpha_j$, which is equal to $\frac{\alpha_0}{1-\alpha_0}$, since $\sum^k_{j=0} \alpha_j = 1$.

As α_0 tends to 1, the Bregman bubble pressure, being proportional to $\frac{\alpha_0}{1-\alpha_0}$, tends to infinity, causing the extent of the regions around the k exponential distribution centroids, where the corresponding distributions have higher weight than the background distribution in Equation 4, to shrink towards 0 (since the weights of the exponential distributions all get forced to 0). For the hard BBC, the increasing weight of α_0 corresponds to the number of points clustered, s, tending towards 0. The Bregman bubble pressure can also be thought of as being proportional to $(n - s)/s$, where s is the number of data points clustered, and is an input to the hard BBC algorithm (Algorithm 1).

Conversely, when α_0 tends to 0, the background pressure (proportional to $\frac{\alpha_0}{1-\alpha_0}$) tends to 0, and the Soft BBC model gives rise to Bregman Soft Clustering, where there is no background distribution. This also corresponds to the hard BBC model reducing to Bregman Clustering, where $s = n$ and all the data points are clustered.

Note that the behavior of the Bregman Bubble for the two extreme cases (0 and infinite pressure) is also analogous to the phenomena of water bubbles reducing to very small sizes under extreme external hydrostatic pressure, and expanding to unlimited extent when all pressure is removed (such as for free moving air molecules in a perfect vacuum).

5.2 Motivation

BBC-S 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. If the bubbles are initialized in sparse regions, they have to expand in order to enclose s points. Then, during each iteration, the bubble representatives move to lower cost nearby locations, and the bubbles shrink in their extent [11]. These mechanisms ensure that the results are less sensitive to initialization. However, when threshold s is small, only a few close neighbors get assigned, thereby decreasing the mobility of the representatives at each iteration. This makes it difficult for BBC-S to find small, dense regions far from initial seed locations. Addressing this issue by starting with a large s would be contrary to the goal of finding small dense regions. This problem is even more severe with BBC-Q, since the bubbles cannot expand automatically in sparser regions. Is there a way to improve upon the ability of BBC-S to "expand" in a sparse region, while still optimizing clustering over small, dense regions?

The pressure mechanism described in the previous section indicates a way of ameliorating this problem. The essential idea is to start BBC with a very small pressure, allowing it to reach out to all the data points, and then slowly increasing the pressure, which would correspond to increasing α_0 or s for Soft and Hard BBC respectively. This causes the bubbles to be "squeezed" by the increasing external pressure into denser regions. Moreover, bubbles that move to a denser region retain proportionately more points at the expense of bubbles in less dense regions. This is because when points compete for being assigned to one of the k clusters (Stage 2 of Algorithm 1), the ones nearest to their respective centroids get assigned first, while $n - s$ points farthest from their cluster centroids get dropped. Both of these trends help to improve solution quality. A demo of the BBC-Press algorithm in action on the Gauss-2 dataset illustrating this "squeezing" phenomena can be seen at *http://www.ideal.ece.utexas.edu/∼gunjan/bbc/bbcicdm.ppt.gz*, slides 31 to 36.

[11] A toy example demonstrating this phenomena can be seen in our power-point slides at *http://www.ideal.ece.utexas.edu/∼gunjan/bbc/bbcicdm.ppt.gz*, slides 16 through 20.

5.3 BBC-Press

Based on the ideas described in the last two sections, we propose an algorithmic enhancement to BBC-S that we call *Pressurization* that is designed to improve upon the quality of the local minimum 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 [12], 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 and using the resultant centroids to seed the next iteration, and in practice yields slightly better results. Algorithm 3 describes the steps for the full-convergence version of BBC-Press in more detail. Pressurization can similarly be implemented for the alternate formulation BBC-Q, by varying the fixed cost q_{max}.

5.4 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 (a small positive value close to 0, e.g. 0.001), and then perform a final run with $\alpha_r = \alpha_0$.

5.5 Pressurization vs. Deterministic Annealing

Although our concept of Pressurization conceptually resembles the approach of *Deterministic Annealing* [68], 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 17), whereas Soft Pressurization involves gradually increasing the probability mass α_0 (Equation 4) of the uniform background distribution. 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 is significant 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 [9].

[12] A smaller value gives better results, but runs slower. A value between 0.01 and 0.05 seems to work well for most real-world scenarios.

6 A unified framework

6.1 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 1**, Section 3.2), and show the following:

Proposition 61 *Maximizing $L_2(\Theta|\mathcal{X})$ is identical to minimizing the BBC objective function Q_b (Equation 2).*

Proof. Let us consider the cost function:

$$L_2(\Theta|\mathcal{X}) = \sum_{i=1}^{n} E_{p^\dagger(Y_i=j|\mathbf{x}_i,\Theta)}[\log p(\mathbf{x}_i, Y_i = j|\theta_j)] \tag{14}$$

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 [43] 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 5):

$$L_2(\Theta|\mathcal{X}) \leq L(\Theta|\mathcal{X}) \tag{15}$$

This result is independent of the choice of priors $\{\alpha_j\}_{j=0}^k$. Note that while $L(\cdot)$ depends upon the priors, $L_2(\cdot)$ does not. For our choice of mixture components, based on Equations 3 and 15, one can readily obtain the following form for $L_2(\cdot)$:

$$L_2(\Theta|\mathcal{X}) = \sum_{j=1}^{k} \sum_{\forall Y_i=j}^{k} \log p^\phi(\mathbf{x}_i) - \tag{16}$$

$$\beta D_\phi(\mathbf{x}_i, \theta_j) + \sum_{\forall Y_i=0} \log(p_0)[i]_{i=1}^n$$

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 16 that maximizing $L_2(\Theta|\mathcal{X})$ is identical to minimizing the BBC objective function Q_b (Equation 2).

Proposition 62 *BBC with a fixed s as input (Definition 1, Section 3.2) 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 equations 5 and 3). 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 16. Since BBC is equivalent to optimizing $L_2(\Theta|\mathcal{X})$ (Proposition 61), we can also view BBC with fixed s (**Definition 1**) as input as a special case of Soft BBC with fixed α_0.

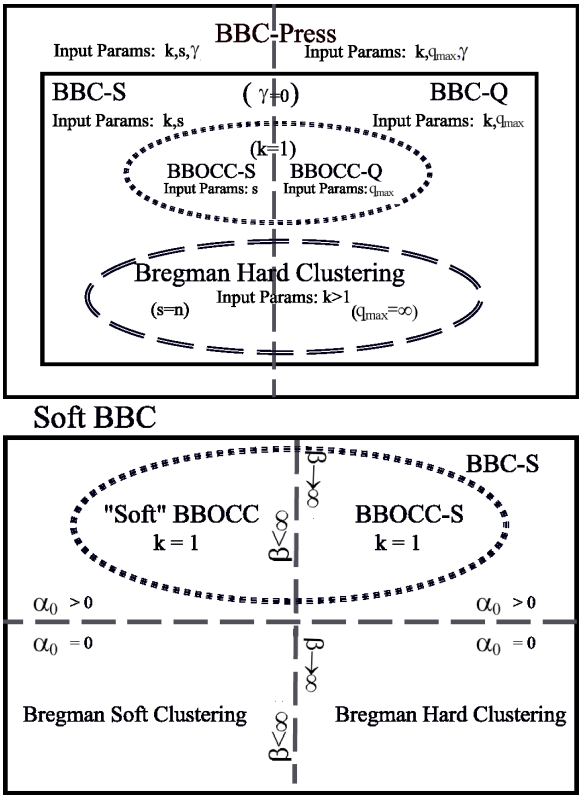


Fig. 2. Unification of various algorithms for a given Bregman Divergence D_ϕ: (top) BBC, BBOCC and Bregman Hard Clustering are special cases of BBC-Press. (bottom) Bregman Hard and Soft Clustering, BBC-S, BBOCC-S and a "soft" BBOCC (consisting of one exponential and a uniform background mixture) are special cases of Soft BBC obtained as specific combinations of (i) whether $\beta \to \infty$, (ii) whether α_0 is 0 (equation 4), and (iii) whether k is 1. Bregman Clustering (both hard and soft) for $k = 1$ does not result in a useful algorithm. BBOCC-S and BBOCC-Q represent BBOCC with fixed s or q_{max} as inputs respectively.

6.2 Other unifications

The following other interesting unifications can also be shown easily for our framework:

1. BBC is a special case of BBC-Press when $\gamma = 0$.
2. Bregman Bubble Clustering becomes BBOCC when k=1.
3. Soft BBC[13] reduces to Bregman Soft Clustering when $p_0 = 0$.
4. Bregman Bubble Clustering reduces to Bregman Hard Clustering (which is a special case of Bregman Soft Clustering) when $q_{max} = \infty$ (for BBC-Q) or when $s = n$ (for BBC-S).

[13] For both cases A and B.

Figure 2 summarizes the hierarchy of algorithms descending from BBC-Press and Soft BBC. We could think of BBC as a search under "constant pressure", and for Bregman Hard Clustering this pressure is zero. Note that for $k = 1$, BBC gives rise to BBOCC. In the context of finding dense regions in the data, BBC can be thought of as a conceptual bridge between the problems of one class clustering and exhaustive k class clustering. However, the defining characteristic of BBC is its ability to find small, dense regions by modeling a small subset of the data. BBC combines the salient characteristics of both Bregman Hard Clustering and BBOCC resulting in an algorithm more powerful than either, and that works across all Bregman Divergences. BBC-S is a natural extension of BBOCC-S following directly from a common underlying generative model, and is not just a heuristic; the difference in the generative model is only in having a single vs. multiple exponential distributions mixed with a uniform background.

7 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:

$$N(\mathbf{x}|\mathbf{a},\sigma) = \frac{1}{\sqrt{(2\pi\sigma^2)^d}} exp(-\frac{||\mathbf{x}-\mathbf{a}||^2}{2\sigma^2}) \tag{17}$$

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:

7.1 σ^2 is fixed

Soft BBC: The parameters $\{\theta_j\}_{j=1}^k$ in Equation 4 correspond to the parameters of the k exponential distributions of the mixture model that are updated in the **M** step of Soft BBC (Algorithm 2). For the Gaussian example, if we fix the values of $\{\sigma_j^2\}_{j=1}^k$ for the k spherical Gaussians mixtures then the only parameters 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 3) 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 2 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 equations 6 and 7 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 8 or 11 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 10.

BBC-S: The proof for Proposition 62 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 3.2) 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).

7.2 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 2) and the sufficient statistics $\mathbf{x}_s$ becomes $[\mathbf{x}, \mathbf{x}^2]^T$. The **E** step of Algorithm 2 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 equations 6 and 7 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 equations 8 or 11 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, (which is not the Squared Euclidean distance) can be derived from the relationship defined by Equation 3 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).

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

7.3 "Flavors" of 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 4.4), (2) the Gaussian mixture variances are updated (Section 7.1 vs. 7.2), 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 7.2, and then assigning this weighted average to the variances of all the k Gaussians. Corresponding BBC-S for these eight flavors could also be derived. Figure 3 shows a comparison of bubbles generated using some of these variants for the simulated 2-D dataset (Gauss-2 dataset, Table 2, Section 10) that has points generated from 5 Gaussians of variances varying from small to large and a uniform background.

7.4 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[14]. 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 sections 7.1 and 7.2, and update steps using EM can be readily derived.

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 4.5), using a large variance background does not result in a stable background; the final background mass varies substantially depending upon where the center

[14] Much larger than the cluster variances.

of the background Gaussian lies at convergence. Mixture-6 serves as another baseline for empirically evaluating Soft BBC.

8 Extending BBOCC & BBC to Pearson Distance & Cosine Similarity

8.1 Pearson Correlation and 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 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 [61, 52].

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 $zscore(\mathbf{x}) = \frac{\mathbf{x} - \mu(\mathbf{x})}{\sigma(\mathbf{x})}$ represents the vector-based z-scoring of the data point vector $\mathbf{x}$, $\mu(\mathbf{x})$ is the mean of the elements of the vector $\mathbf{x}$, and $\sigma(\mathbf{x})$ is the standard deviation. Note that we z-score each of the data points separately across features values [15]. We then 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 $2(d-1)$:

$$D_P(\mathbf{x}, \mathbf{y}) = \frac{\parallel zscore(\mathbf{x}) - zscore(\mathbf{y}) \parallel^2}{2(d-1)} \tag{18}$$

D_P can also be viewed as the Squared Euclidean distance between points that have been first rotated by subtracting the mean, and then, by variance normalization, 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 2, we refer to Q_b as *Average Pearson Distance*(APD).

Proposition 81 *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 [19] 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

[15] 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.

the Squared Euclidean distance between points normalized 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 81, for $D = D_P$ the optimum 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 to work with D_P and ensures convergence to a local minimum [16]. Since BBOCC is a special case of BBC, the same modification works for BBOCC too for the problem of One Class Clustering.

8.2 Extension to Cosine Similarity

Because of the relationship between Cosine Similarity and Pearson Distance described above, BBC will also work with Cosine Similarity, which is a popular similarity measure for clustering textual data [19]. 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.

8.3 Pearson Distance vs. (1-Cosine Similarity) vs. other Bregman Divergences - which one to use where?

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. In Section 10, we present results based on Pearson Distance for the biological datasets Lee and Gasch (see Table 2), while for the 20 Newsgroup data, where the features consist of words, (1-Cosine Similarity) is used.

A similar distinction should be kept in mind about Bregman Divergences in general; although BBC works with all Bregman Divergences, it can produce quite different results depending upon the choice of the divergence; the particular problem domain and the underlying exponential distribution (Equation 3 and Proposition 62) should guide the selection of the appropriate Bregman Divergence for BBC.

[16] and the corresponding local maximum for Average Pearson Correlation.

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

We now present an alternative to Pressurization for alleviating the problem of local minima in our local search. For medium-sized datasets [17], the seeding framework described in this section is computationally feasible on machines with modest resources, and provides two key advantages over Pressurization: (1) deterministic results, and (2) the ability to automatically determine the number of distinct dense regions in the data (k), the location of these dense regions, and their representative centroids in $\mathcal{X}$. These k representatives are then used to seed BBC.

9.1 Background

In [27], we presented a deterministic enumeration based algorithm called Hypersphere One Class Clustering (HOCC) that finds an approximate, restricted solution to the problem of finding a single dense cluster in the data. The centroid location determined by HOCC can be used to to seed BBOCC. HOCC is based on the observation that if $\mathbf{c}$ is restricted to one of the sample data points $\mathcal{X}$, then the number of distinct solutions is only $n(n-1)$, and can be enumerated efficiently.

The problem of finding a good seeding method for local search for 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 and also indicates a suitable value of k? This is answered positively via the DGRADE algorithm described next.

9.2 DGRADE Algorithm

The key idea behind the DGRADE seeding algorithm is to (i) limit the search for seeds to the actual data points and (ii) do the search in a computationally efficient manner. For each point, we consider the cost of a Bregmanian Ball that is centered at that location, and encompasses $s = s_{one}$ points. This cost can be viewed as the "potential" at the corresponding point. If the cost at a neighboring point is lower, then that point is (locally) more preferable. This leads to a chain of local preferences viewed as a potential gradient. Points that follow a chain of preferences to the same final point (lowest local potential) can be viewed as belonging to the same partition, the physical analogy being that they are all part of the same "basin of attraction". Thus the data points can be quickly grouped,

[17] Such as gene clustering datasets, where the number of genes is usually $O(10^4)$.

with the number of groups indicating the corresponding value of k. Note that this k is dependent on s_{one}, which acts as a scale or smoothing parameter.

Density Gradient Enumeration (DGRADE), described in detail as Algorithm 4, has the following key 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 [18]. 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 of $\mathbf{x}$ (including itself). 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 Gauss-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. DGRADE consists of two distinct phases: (1) sorting of points by Bregmanian ball cost to get the absolute measurement of cost in various regions of the data, and (2) pointing each data point to the direction of maximum decline in cost within the same Bregmanian ball to get 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 [19]. Although DGRADE can be used as a algorithm to find dense

[18] For the restricted One Class case ($k = 1$), returning this (lowest cost) point as a solution corresponds to the One Class seeding algorithm described in our earlier paper [27], and results in strong optimality guarantees for One Class that are described in more detail in [27, 28].

[19] Bregman Divergences are generally not symmetric; Squared Euclidean is a notable exception.

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.

A detailed time and space complexity of DGRADE is given in [28]. To a first approximation, time complexity is quadratic in data size, and space requirements are only $O(n)$.

9.3 Selecting s_{one}: the smoothing parameter for DGRADE

s_{one}, which is a parameter for DGRADE that needs to be input by the user, 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 Gauss-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 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 datasets with well-defined or prominent clusters, 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.

10 Experiments

10.1 Overview

Results in Section 10.4 show 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. Section 10.5 then presents 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. Results suggest that it is also possible to combine all three: BBC, Pressurization and DGRADE to achieve the best quality of clustering, and confirm that in practice, DGRADE can estimate k automatically, and the deterministic, high quality results generated are a good alternative to Pressurization for biological datasets.

10.2 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. k represents the number of clusters specified to the clustering methods that require it as an input, and is only needed for BBC when testing without DGRADE. When seeding with DGRADE, k output by DGRADE was used for all methods that required it as an input.

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

Real Data:

A. Microarray Datasets: A microarray dataset can be represented by a matrix that shows (suitably normalized) expression levels of genes (rows) across different experiments/conditions (columns). Researchers are interested in clustering either the rows, the columns, or simultaneously clustering both rows and columns, in order to find similar genes, similar conditions, or subsets of genes with related expressions across a subset of conditions, respectively [62, 60, 4]. For this paper, we report results on clustering the rows of the Lee dataset, which was obtained from [47], and consists of 591 gene-expression conditions on yeast obtained from the Stanford Microarray database [24] (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 Gasch dataset [21] 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. For Gasch, we clustered the columns instead, for three reasons: (1) we have good labels for experiments from [21], (2) the Gasch Array dataset viewing the conditions as the objects to be clustered, provides a high-dimensional biological testbed (6,151 dimensions), and (3) the 173 Gasch Array experiments are already incorporated in the 591 conditions contained in the Lee dataset, which we use for clustering genes.

B. Text Data: The 20-Newsgroup (20-NG) dataset is a popular dataset for text classification [45], 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 Gauss-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 Gauss-10 and Gauss-40 datasets.

10.3 Evaluation Methodology

Having shown the value of the local search approach and seeding for the One Class problem in [27], this paper presents results for the general case of finding multiple dense clusters. Additional results for the One Class case can be found in [28].

Evaluation Criteria: Evaluating clustering is a challenging problem even when labeled data is available [65]. Depending upon the type of the labeled data, we performed the following three different types of evaluations:

1. *Adjusted Rand Index*: Given a set of class labels $\mathcal{U}$, and a set of cluster labels $\mathcal{V}$ for a set of points $\mathcal{X}$, *Rand Index* (RI) is computed as:

$$RI = \frac{a_{uv} + a_u}{a_{uv} + a_u + a_v + d_{uv}} \tag{19}$$

   where: a_{uv} represents the number of pairs of points (in $\mathcal{X}$) that have the same label in $\mathcal{U}$ and $\mathcal{V}$, a_u represents the pairs of points that have the same label in $\mathcal{U}$ but not in $\mathcal{V}$, a_v represents the pairs of points that have the same label in $\mathcal{V}$ but not in $\mathcal{U}$, and d_{uv} represents the pairs of points that have different labels both in $\mathcal{U}$ and $\mathcal{V}$.
   A problem with Rand Index is that the expected value for Rand Index of two random partitions does not go to 0, but depends on a multitude of factors, including the number of distinct labels in $\mathcal{U}$ and $\mathcal{V}$, and the size of the sets, (i.e. the number of data points being clustered). Adjusted Rand Index was proposed by [35] as a normalized version of Rand Index that takes care of this problem, and returns 1 for a perfect agreement between the class label set $\mathcal{U}$ and the clustering label set $\mathcal{V}$, and 0 when the clustering is as bad as random assignments. The adjustment uses the following formula:

$$ARI = \frac{RI - (ExpectedRI)}{(MaximumRI) - (ExpectedRI)} \tag{20}$$

   Using a generalized hypergeometric model, [35] showed that the ARI computation can be reduced to the following form:

$$ARI = \frac{\sum_{i,j}\binom{n_{i,j}}{2} - [\sum_i \binom{n_{i\cdot}}{2}\sum_j \binom{n_{\cdot j}}{2}]/\binom{n}{2}}{\frac{1}{2}[\sum_i \binom{n_{i\cdot}}{2} + \sum_j \binom{n_{\cdot j}}{2}] - [\sum_i \binom{n_{i\cdot}}{2}\sum_j \binom{n_{\cdot j}}{2}]/\binom{n}{2}} \tag{21}$$

   where $n_{i,j}$ represents the number of data points that are in class i and cluster j, $n_{i\cdot}$ represents the number of data points in class i, $n_{\cdot j}$ represents the number of data points in cluster j.
   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 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 7.4).

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 4.5 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 $j = \underset{j=1\rightarrow k}{\text{argmax}}\, p(Y_i = j|\mathbf{x}_i)$. 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 [20] 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 [20], 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.

10.4 Results for 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 Gauss-10 data and an ARI between 0.8 and 0.9 for up to 40% coverage on Gauss-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 [21]. For Gauss-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 17 is substituted for $p_{(\psi,\theta)}$ in equation 6), 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 6).

On the Gauss-40 dataset, BBC-Press continues to give an ARI ≈ 1 for up to 40% coverage (Figure 6(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 7(a)) where most of the data is relevant, discovering small number of

[20] Bregman Hard Clustering reduces to K-Means when D_ϕ is Sq. Euclidean distance, which is the distance measure used for the Gaussian datasets (Table 2).

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

relevant gene clusters on high-dimensional microarray data (7 (e)), to clustering large and high-dimensional documents (7 (c)).

On both artificial and real datasets (figures 6, 7), 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 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 7(f) shows that (1) BBC-Press not only beats other methods by a wide margin but also shows high enrichment 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 fact, for some situations (Figure 6(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 three other 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 become 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 a small amount of labeled data is available, the best k can be estimated for a fixed s using an approach such as PAC-MDL [5], while a reasonable s can be picked by applying BBC-Press on a range of s and picking the "knee" (e.g. Figures 6(a),(b),(c) and 7(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 Gauss-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" [22]. 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.

10.5 Results on BBC with 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 three other 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 [20], while we searched 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 9. DGRADE then uses the same s_{one} for smaller coverages, which can result in a smaller k (Figure 9) 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 present results on DGRADE when it is used for seeding BBC, on two real and three synthetic datasets.

Ability to estimate s_{one} automatically: In the table in Figure 8(a), the first column shows the values of s_{one} determined automatically using one of the three methods described in Section 9, 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 Gauss-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 8 (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

[22] Which represents one of the class of experiments set up by [21]

only deterministic for varying values of s, but also have another useful property that follows directly from Algorithm 4; 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 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. The decline in k with s can be seen for the 5 different datasets in Figure 9, and provides us with a meaningful method for selecting centroids and k for seeding BBC (for varying fraction of data clustered), as compared with other seeding methods that require k as an input. The k corresponding to those shown in Figure 9 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 10 and 11 show results for BBC seeded with DGRADE as compared to the other alternatives; BBC with Pressurization and the other three benchmark algorithms. For the Gauss-2 dataset, when DGRADE was used to seed Soft BBC, the results were comparable to that of Soft BBC with Pressurization (Figure 10(b)), while the results of DGRADE as a clustering algorithm by itself were quite good (Figure 10(a)). On the Gauss-40 dataset (figures 10(e) and (f)), although DGRADE does not perform well by itself, when used for seeding BBC-Press, the combination gives results that are 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. Similar trends were seen on the Gauss-10 (figures 10(c) and (d)) dataset. One possibility is that the global search bias behind DGRADE provides additional advantages to a robust local 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 (Gauss-40 vs. Gauss-2), and is similar to the boost in performance observed when DGRADE-style seeding is combined with local search for the One Class scenario [27, 28]. This phenomenon of DGRADE and BBC-Press improving results as a combination is also observed on the really high-dimensional real datasets- the Gasch Array (figures 11(a) and (b)). However, this relationship is only seen for the lowest coverage of 0.05 on the Lee dataset (Figure 11(e)), perhaps because the fraction of genes that are usually dense when clustering genes is usually very small.

11 Concluding Remarks

Bregman Bubble Clustering 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 in such settings. The availability of appro-

priate Bregman Divergences [23] for a variety of problem domains where finding dense clusters is important, opens density-based clustering to many new domains. Moreover, the extension of BBC to Pearson Correlation (Pearson Distance) is particularly helpful for analyzing several biological datasets. 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 shows that BBC arises out of a more fundamental model involving a mixture of exponentials and a uniform background.

Empirical results show that BBC-Press gives good results on a variety of problems involving both low and high-dimensional feature spaces, often outperforming other alternatives by large margins. DGRADE provides effective seeding for BBC and BBC-Press, and provides an additional mechanism for indicating the appropriate number of dense clusters that one should seek. Overall, the combination of the three components; BBC, Pressurization, and DGRADE provides a robust framework for finding dense clusters with several key properties: deterministic results, reasonable scalability to large, high-dimensional datasets, and applicability to a wide variety of problem domains.

There are several properties of the Bregman Bubble Clustering framework that can be used to further expand the robustness and quality of clustering. We conclude this chapter by outlining two promising extensions:

Bregman Bubble Co-clustering: A generalized framework called Bregman Co-clustering was proposed by [54] that unifies many different *checkerboard* co-clustering algorithms, where the clusters are defined not just on the rows (data objects) but also simultaneously on the columns (features) when the dataset is viewed as a matrix. If one rearranges the rows and columns of this matrix such that rows/cols belonging to the same cluster are contiguous, then the permuted matrix shows checkerboard pattern of non-overlapping co-clusters. However, often one desires to co-cluster only parts of the data, and also to allow for co-clusters that overlap. For example, genes' interactions in gene-expression data are often overlapping. Such data also are typically highly skewed - the number of experiments is typically a few dozens or hundreds, while the number of genes is at least an order of magnitude more. For such data, it is better to prune a large number of genes before applying co-clustering. Since Bregman Bubble Clustering is a generalization of Bregman Clustering, and since Bregman Clustering is a component/step in Bregman Co-clustering, it is possible to modify Bregman Co-clustering to use Bregman Bubble Clustering, and to obtain a co-clustering algorithm that can find dense co-clusters. Such an algorithm also makes it possible to find overlapping dense co-clusters. This approach has been successfully applied to gene-expression data with results superior to a host of alternatives [15, 23].

Online Bregman Bubbles for detecting non-stationary dense clusters: OC-IB (Section 2.3) can be considered as a randomized equivalent of BBC-Q for $k = 1$ (Section 3.5). It is also possible to derive a *k-class* randomized

[23] e.g. Itakura-Saito for voice identification, Mahalanobis distance for digital Mammography, and KL-divergence for identifying most relevant documents.

version of the algorithm for BBC-Q or BBC-S (Section 3.4 and 3.5). This could be useful in online settings, e.g. for modeling dense clusters in scenarios where there is substantial *concept drift*, i.e. the distribution that the data is coming from is not stationary, but drifting gradually. With an online version of BBC, the dense cluster centers could move as the modes in the data shift. One application could be for modeling highly coherent sets of customers in market-basket data. Such customer groupings tend to shift slowly over time [30]. Another application could be to track rare but recurring non-stationary anomalies; as old anomalies stop appearing or shift, the online Bregman Bubble model could adapt quickly because of its localized nature. One area where such anomalous data tends to shift rather than randomly appear is in online fraud, where lots of very similar fraudulent content originates from a single individual, and the fraudsters tend to modify keywords only slightly to try to evade the detection system. Using previous and newly identified fraudulent data points as cluster centers, such similar but constantly drifting patterns could perhaps be better tracked using online BBC. Yet another application could be to follow visual movements of dense patterns In low-dimensional data: such as "blobs" in infra-red images of objects in a military, wildlife, or an oceanic setting, or as a part of an airport security system at an airport.

Acknowledgments: This research was supported by NSF grants IIS-0713142 and IIS-1017614. We are also grateful to Srujana Merugu and Arindam Banerjee for some useful discussions.

References

1. Ankerst, M., Breunig, M.M., Kriegel, H.P., Sander, J.: OPTICS: Ordering points to identify the clustering structure. In: Proc. ACM SIGMOD. pp. 49–60 (1999)
2. Arabie, P., Carroll, J.D., DeSarbo, W., Wind, J.: Overlapping clustering: A new method for product positioning. Journal of Marketing Research 18(3), 319–317 (August 1981)
3. Banerjee, A., Merugu, S., Dhillon, I., Ghosh, J.: Clustering with Bregman divergences. JMLR 6, 1705–1749 (2005)
4. Banerjee, A., Krumpelman, C., Ghosh, J., Basu, S., Mooney, R.J.: Model-based overlapping clustering. In: Proc. KDD'05. pp. 532–537. Chicago, Illinois, USA (2005)
5. Banerjee, A., Langford, J.: An objective evaluation criterion for clustering. In: KDD-04. Seattle, Washington, USA (August 2004)
6. Battle, A., Segal, E., Koller, D.: Probabilistic discovery of overlapping cellular processes and their regulation. In: Eighth Annual International Conference on Research in Computational Molecular Biology (RECOMB 2004) (April 2004)
7. Berchtold, S., Keim, D.A., Kriegel, H.P.: The X-Tree: An index structure for high-dimensional data. In: Proceedings of the 22nd International Conference on Very Large Databases. pp. 28–39. Morgan Kaufmann Publishers, San Francisco, U.S.A. (1996)
8. Buzo, A., Gray, A.H., Gray, R.M., Markel, J.D.: Speech coding based on vector quantization. IEEE Transactions on Accoustics, Speech and Signal Processing 28(5), 562–574 (1980)

9. Casella., G., Robert, C.P., Wells, M.T.: Mixture models, latent variables and partitioned importance sampling. In: Technical Report, RePEc:fth:inseep:2000-03, Institut National de la Statistique et des Etudes Economiques. http://ideas.repec.org/p/fth/inseep/2000-03.html (2003)
10. Chakaravathy, S.V., Ghosh, J.: Scale based clustering using a radial basis function network. IEEE Transactions on Neural Networks 2(5), 1250–61 (September 1996)
11. Cheng, Y.: Mean shift, mode seeking, and clustering. IEEE Trans. Pattern Anal. Mach. Intell. 17(8), 790–799 (1995)
12. Crammer, K., Chechik, G.: A needle in a haystack: Local one-class optimization. In: ICML-04. Banff, Alberta, Canada (2004)
13. Crammer, K., Singer, Y.: Learning algorithms for enclosing points in Bregmanian spheres. In: COLT-03. pp. 388–402 (2003)
14. Deodhar, M., Ghosh, J.: Simultaneous co-clustering and modeling of market data. In: Workshop for Data Mining in Marketing (DMM 2007). IEEE Computer Society, Leipzig, Germany (2007)
15. Deodhar, M., Ghosh, J., Gupta, G., Cho, H., Dhillon, I.: A scalable framework for discovering coherent co-clusters in noisy data. In: Bottou, L., Littman, M. (eds.) Proceedings of the 26th International Conference on Machine Learning. pp. 241–248. Omnipress, Montreal (June 2009)
16. Dettling, M., Bühlmann, P.: Supervised clustering of genes. Genome Biol. 3(12) (2002)
17. Dhillon, I., Mallela, S., Kumar, R.: A divisive information-theoretic feature clustering algorithm for text classification. JMLR 3, 1265–1287 (March 2003)
18. Dhillon, I.S., Guan, Y., Kogan, J.: Refining clusters in high-dimensional text data. In: 2nd SIAM International Conference on Data Mining (Workshop on Clustering High-Dimensional Data and its Applications) (April 2002)
19. Dhillon, I.S., Modha, D.S.: Concept decompositions for large sparse text data using clustering. Machine Learning 42(1-2), 143–175 (January-February 2001)
20. Ester, M., Kriegel, H.P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proc. KDD-96. pp. 226–231 (1996)
21. 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)
22. Georgescu, B., Shimshoni, I., Meer, P.: Mean shift based clustering in high dimensions: A texture classification example. In: ICCV '03: Proceedings of the Ninth IEEE International Conference on Computer Vision. pp. 456–463. IEEE Computer Society, Washington, DC, USA (2003)
23. Ghosh, J., Deodhar, M., Gupta, G.: Detection of Dense Co-clusters in Large, Noisy Datasets, appears in book: Pattern Recognition and Machine Vision (in memory of Professor King-Sun Fu) by P.S.P. Wang. River Publishers, Aalborg, Denmark (2010)
24. Gollub J. et al.: The Stanford Microarray Database: data access and quality assessment tools. Nucleic Acids Res 31, 94–96 (2003)
25. Guha, S., Rastogi, R., Shim, K.: Rock: A robust clustering algorithm for categorical attributes. In: 15th International Conference on Data Engineering. p. 512. Sydney, Australia (1999)
26. Gupta, G.: Robust methods for locating multiple dense regions in complex datasets. In: PhD Thesis, University of Texas at Austin (December 2006)
27. Gupta, G., Ghosh, J.: Robust one-class clustering using hybrid global and local search. In: Proc. ICML 2005. pp. 273–280. Bonn, Germany (August 2005)

28. Gupta, G., Ghosh, J.: Bregman Bubble Clustering: A robust framework for mining dense clusters. In: Tech Report, Dept. of Elec. & Comp. Engineering, University of Texas at Austin. IDEAL-TR04,http://www.lans.ece.utexas.edu/techreps.html (September 2006)
29. Gupta, G., Liu, A., Ghosh, J.: Automated Hierarchical Density Shaving: A robust, automated clustering and visualization framework for large biological datasets. To Appear, IEEE/ACM Transactions on Computational Biology and Bioinformatics (March 2008)
30. Gupta, G.K.: Modeling Customer Dynamics Using Motion Estimation in A Value Based Cluster Space for Large Retail Data-sets. Master's thesis, University of Texas at Austin (August 2000)
31. Gupta, G.K., Ghosh, J.: 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)
32. Gupta, G.K., Ghosh, J.: Value Balanced Agglomerative Connectivity Clustering. In: SPIE conference on Data Mining and Knowledge Discovery III, SPIE Proc. vol. 4384, pp. 6–15. Orlando, Florida (April 2001)
33. 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)
34. Hendrickson, B., Leland, R.: An improved spectral graph partitioning algorithm for mapping parallel computations. SIAM Journal on Scientific Computing 16(2), 452–469 (1995)
35. Hubert, L., Arabie, P.: Comparing partitions. Journal of classification pp. 193–218 (1985)
36. Jain, A.K., Dubes, R.C.: Algorithms for Clustering Data. Prentice Hall, Englewood Cliffs, NJ, USA (1988)
37. Jiang, D., Pei, J., Ramanathan, M., Tang, C., Zhang, A.: Mining coherent gene clusters from gene-sample-time microarray data. In: KDD'04. pp. 430–439. Seattle, WA, USA (2004)
38. Jiang, D., Pei, J., Zhang, A.: DHC: A density-based hierarchical clustering method for time series gene expression data. In: BIBE '03. p. 393. IEEE Comp. Soc., Washington, DC, USA (2003)
39. Johnson, S.C.: Hierarchical clustering schemes. Psychometrika 32(3), 241–254 (September 1967)
40. Judd, A., Hovland, M.: Seabed Fluid Flow: The Impact of Geology, Biology and the Marine Environment. Cambridge University Press, Cambridge. (2007)
41. Karypis, G., Kumar, V.: A fast and high quality multilevel scheme for partitioning irregular graphs. SIAM Journal of Scientific Computing 20(1), 359–392 (1998)
42. Karypis, G., Aggarwal, R., Kumar, V., Shekhar, S.: Multilevel hypergraph partitioning: Applications in VLSI domain. In: Design and Automation Conference (1997)
43. Kearns, M., Mansour, Y., Ng, A.Y.: An information-theoretic analysis of hard and soft assignment methods for clustering. In: Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence. pp. 282–293. AAAI (1997)
44. Kriegel, H.P., Pfeifle, M., Pötke, M., Seidl, T.: The paradigm of relational indexing: A survey. In: BTW. LNI, vol. 26. GI (2003)
45. Lang, K.: Newsweeder: Learning to filter netnews. In: Proceedings of the Twelfth International Conference on Machine Learning. pp. 331–339 (1995)
46. Lazzeroni, L., Owen, A.B.: Plaid models for gene expression data. Statistica Sinica 12(1), 61–86 (January 2002)

47. Lee, I., Date, S.V., Adai, A.T., Marcotte, E.M.: A probabilistic functional network of yeast genes. Science 306, 1555–1558 (2004)
48. Linde, Y., Buzo, A., Gray, R.M.: An algorithm for vector quantizer design. IEEE Transactions on Communications 28(1), 84–95 (1980)
49. Linden, G., Smith, B., York, J.: Amazon.com recommendations: Item-to-item collaborative filtering. IEEE Internet Computing 7(1), 76–80 (Jan-Feb 2003)
50. Long, B., Zhang, Z.M., Wu, X., Yu, P.S.: Relational clustering by symmetric convex coding. In: ICML '07. pp. 569–576. ACM, New York, NY, USA (2007)
51. MacQueen, J.: Some methods for classification and analysis of multivariate observations. In: 5th Berkeley Symposium on Mathematical Statistics and Probability. pp. 281–297 (1967)
52. Mansson, R., Tsapogas, P., et al., M.A.: 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)
53. McGuire, A.M., Church, G.M.: Predicting regulons and their cis-regulatory motifs by comparative genomics. Nucleic Acids Research 28(22), 4523–4530 (2000)
54. Merugu, S.: Privacy-preserving distributed learning using generative models. In: PhD Thesis, The University of Texas at Austin (August 2006)
55. Neal, R., Hinton, G.: A view of the EM algorithm that justifies incremental, sparse, and other variants. In: Jordan, M.I. (ed.) Learning in Graphical Models. Kluwer (1998), http://www.cs.toronto.edu/~radford/em.abstract.html
56. Pietra, S.D., Pietra, V.D., Lafferty, J.: Duality and auxiliary functions for Bregman distances. In: Technical Report CMU-CS-01-109, School of Computer Science. Carnegie Mellon University (2001)
57. Schmid, C., Sengstag, T., Bucher, P., Delorenzi, M.: MADAP, a flexible clustering tool for the interpretation of one-dimensional genome annotation data. Nucleic Acids Res pp. W201–W205 (2007)
58. Schölkopf, B., Burges, C., Vapnik, V.: Extracting support data for a given task. In: KDD. AAAI Press, Menlo Park, CA (1995)
59. Schölkopf, B., Platt, J.C., Shawe-Taylor, J.S., Smola, A.J., Williamson, R.C.: Estimating the support of a high-dimensional distribution. Neural Computation 13(7), 1443–1471 (2001)
60. Segal, E., Battle, A., Koller, D.: Decomposing gene expression into cellular processes. In: 8th Pacific Symposium on Biocomputing (PSB), Kaua'i (January 2003)
61. Sharan, R., Shamir, R.: Click: A clustering algorithm with applications to gene expression analysis. Proc. 8th ISMB pp. 307–316 (2000)
62. Slonim, N., Atwal, G.S., Tkacik, G., Bialek, W.: Information-based clustering. PNAS 102(51), 18297–18302 (2005)
63. Strehl, A., Ghosh, J.: Value-based customer grouping from large retail data-sets. In: SPIE Conference on Data Mining and Knowledge Discovery: Theory, Tools, and Technology II, 24-25 April 2000, Orlando, Florida, USA. vol. 4057, pp. 33–42. SPIE (April 2000)
64. Strehl, A., Ghosh, J.: Relationship-based clustering and visualization for high-dimensional data mining. INFORMS Journal on Computing 15(2), 208–230 (2003)
65. Strehl, A., Ghosh, J., Mooney, R.J.: Impact of similarity measures on web-page clustering. In: AAAI Workshop on AI for Web Search (AAAI 2000). pp. 58–64. AAAI/MIT Press (July 2000)
66. Tax, D., Duin, R.: Data domain description using support vectors. In: Proceedings of the ESANN-99. pp. 251–256 (1999)
67. Tenenbaum, J.B., de Silva, V., Langford, J.C.: A global geometric framework for nonlinear dimensionality reduction. Science 290, 2319–2323 (2000)

68. Ueda, N., Nakano, R.: Deterministic annealing EM algorithm. Neural Networks 11(2), 271–282 (1998)
69. Wedel, M., Steenkamp, J.: A clusterwise regression method for simultaneous fuzzy market structuring and benefit segmentation. Journal of Marketing Research pp. 385 – 396 (1991)
70. Wishart, D.: Mode analysis: A generalization of nearest neighbour which reduces chaining effects. In: Proceedings of the Colloquium in Numerical Taxonomy. pp. 282–308. Academic Press, University of St. Andrews, Fife, Scotland (September 1968)
71. Yun, C.H., Chuang, K.T., Chen, M.S.: An efficient clustering algorithm for market basket data based on small large ratios. In: Computer Software and Applications Conference 2001. pp. 505–510 (2001)
72. Zhong, S.: Efficient streaming text clustering. Special issue IJCNN 2005: Neural Networks 18(5-6), 790–798 (2005)

Algorithm 1 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, k seed centroids for the k clusters (optional).

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 $\{\mathbf{c_j}\}_{j=1}^{k}$ with seed centroids if available, else randomly.

end if

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

repeat

/*Assign each point to the closest of the k centroids*/

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

/* Find the s points closest to their centroids and form the cluster*/

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

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

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

/* Save previous centroids, clustering costs, cluster memberships and update to new ones */

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

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

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

/* Recompute cluster centroids based on the new cluster memberships */

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 $(\mathcal{G}^l == \mathcal{G}) \wedge q_{pp} == q$ /* Convergence */

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

Algorithm 2 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 5) 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 equations 6 and 7, where $p_{(\psi,\theta)}(\mathbf{x}_i|\theta_j)$ is defined by equation 3.

end for

end for

{The **M** Step}

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

Update α_j using equation 8 for Case A and 11 for Case B.

Update θ_j using equation 10.

end for

until convergence

Algorithm 3 Hard BBC-Press

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, k seed centroids for the k clusters (optional).

Set γ to a value between 0 and 1, $j = 1$, and $s_{in} = n$.

Run Algorithm 1 until convergence with random starting centroids, unless seed centroids are available, and with $s = s_{in}$.

Set $j = 2$, $s_{in} = s + \lfloor (n - s)\gamma^1 \rfloor$

while $(n - s)\gamma^{j-1} >= 1$ **do**

$s_{in} = s + \lfloor (n - s)\gamma^{j-1} \rfloor$. /* Reduce s_{in} by pressurization rate */

Run Algorithm 1 until convergence, using the k centroids from last run to seed the centroids for this run, and $s = s_{in}$.

end while

Return resultant clustering from the final run of Algorithm 1.

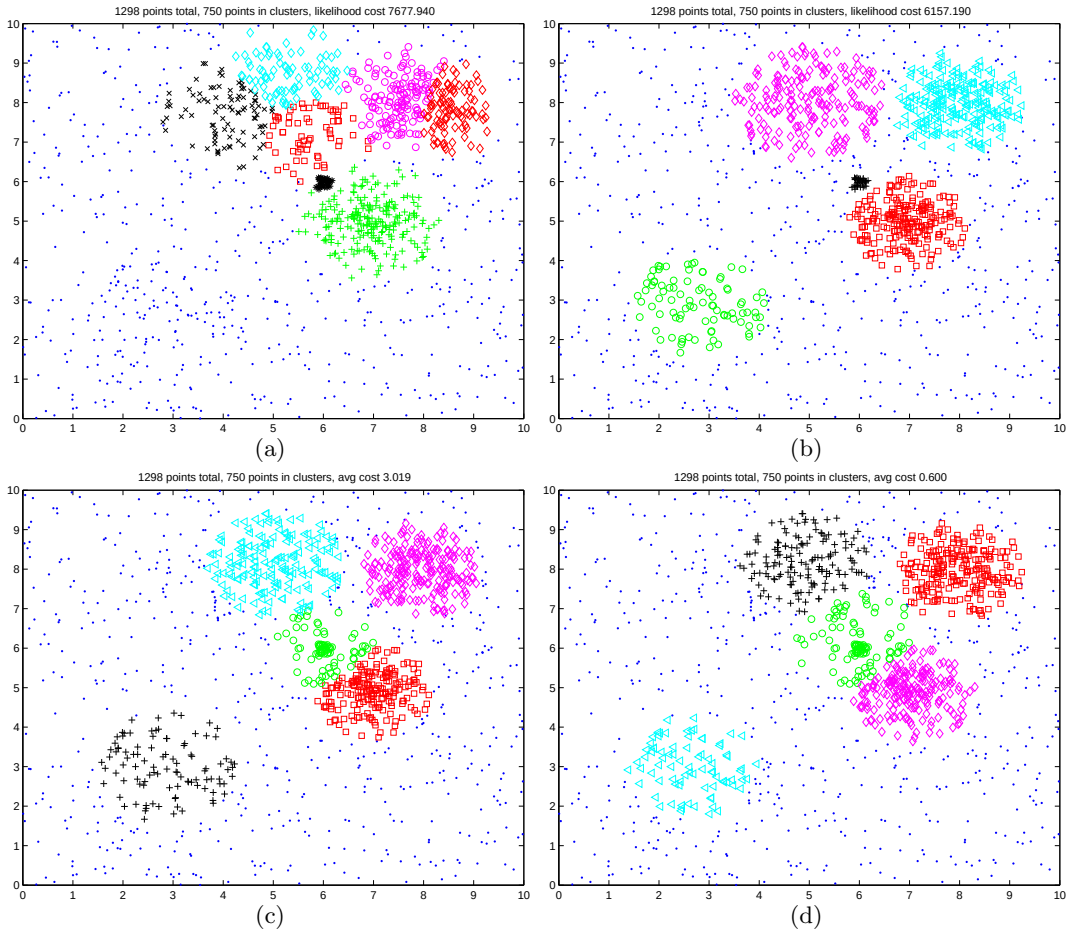
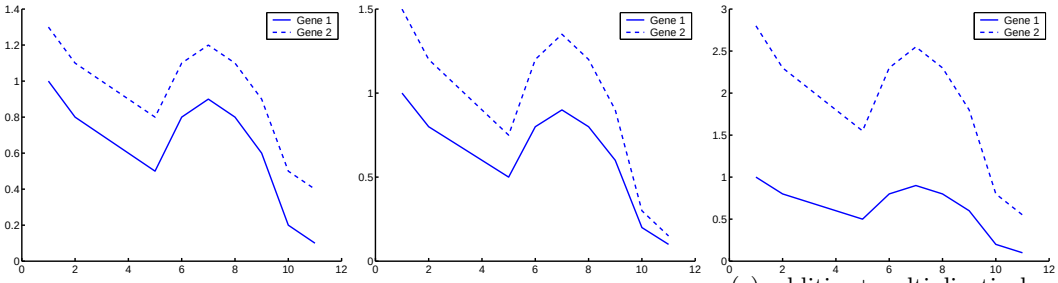


Fig. 3. Comparison of bubbles generated using a variant of Soft BBC and two variants of BBC-S on the simulated 2-D dataset: Soft BBC with updated σ^2 for (a) $k = 7$ and (b) $k = 5$. k was intentionally kept large in (a) to illustrate the non-linear boundaries produced by touching bubbles. BBC-S resulting from Soft BBC model where (c) σ^2 is updatable and (d) σ^2 is fixed. For Soft BBC, each point was assigned at convergence to the cluster to which it had the largest soft assignment value.



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

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

Algorithm 4 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 $\mathcal{G}^*$.

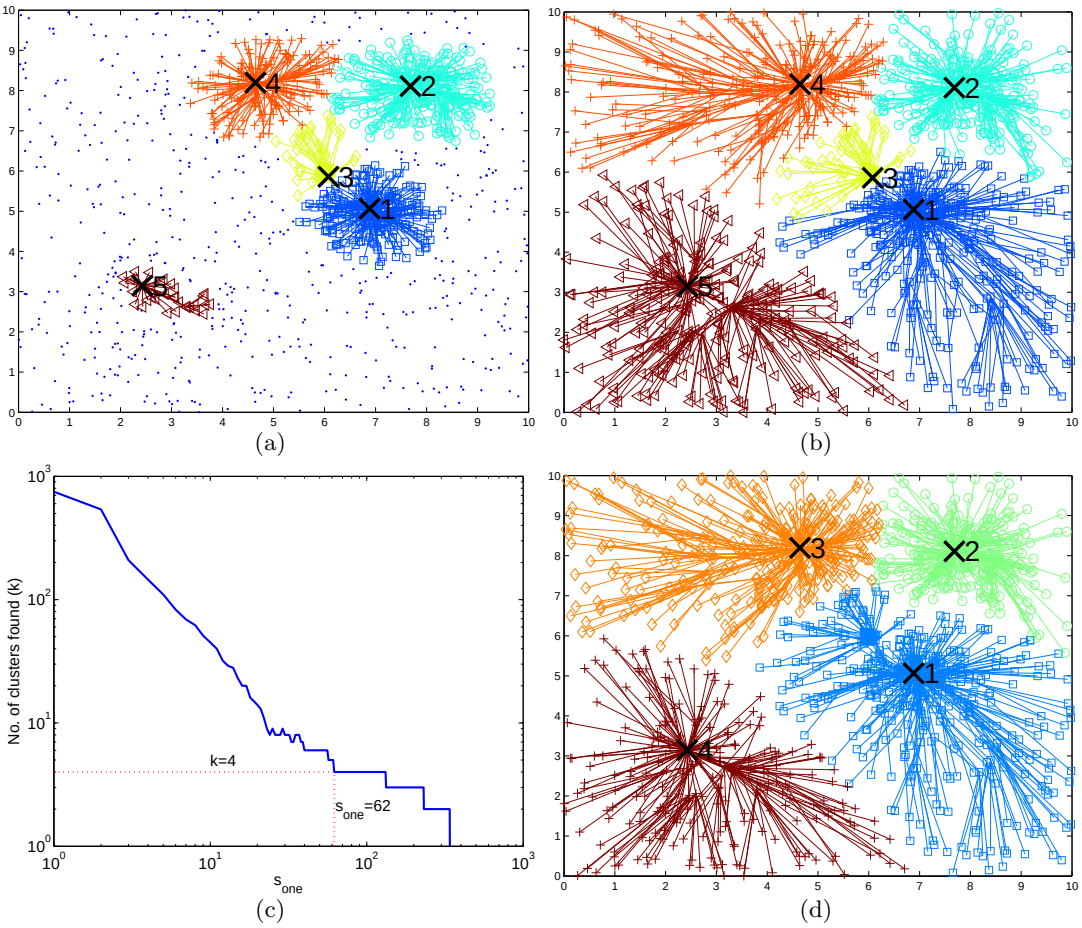
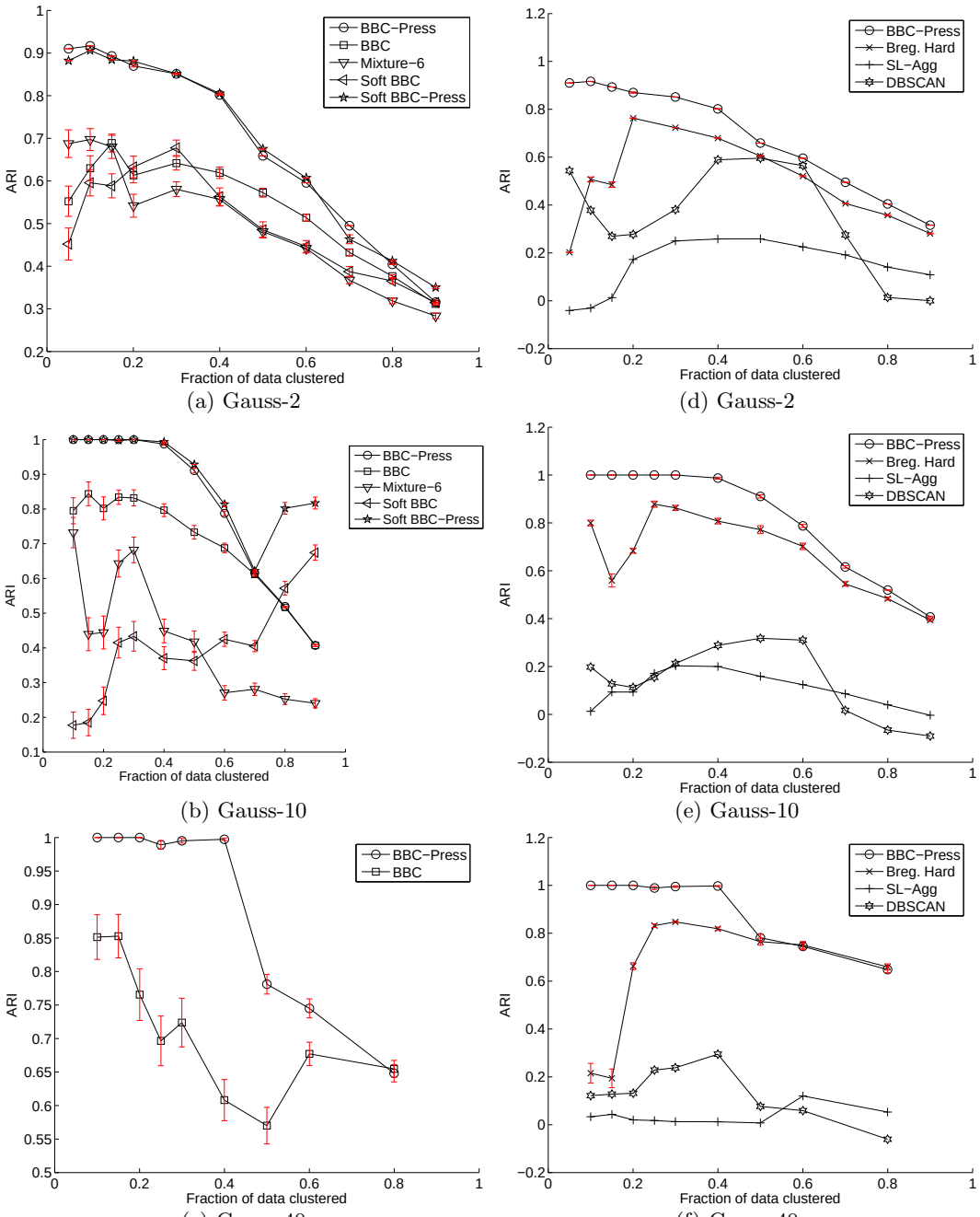


Fig. 5. Clustering of Gauss-2 data using DGRADE for various scenarios. For (a), (b) and (d), lines show the path of the simulated local search converging at the locally lowest cost/densest point, which also form the centroid of the corresponding cluster, and are marked as "x" and numbered 1 to k. For (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).

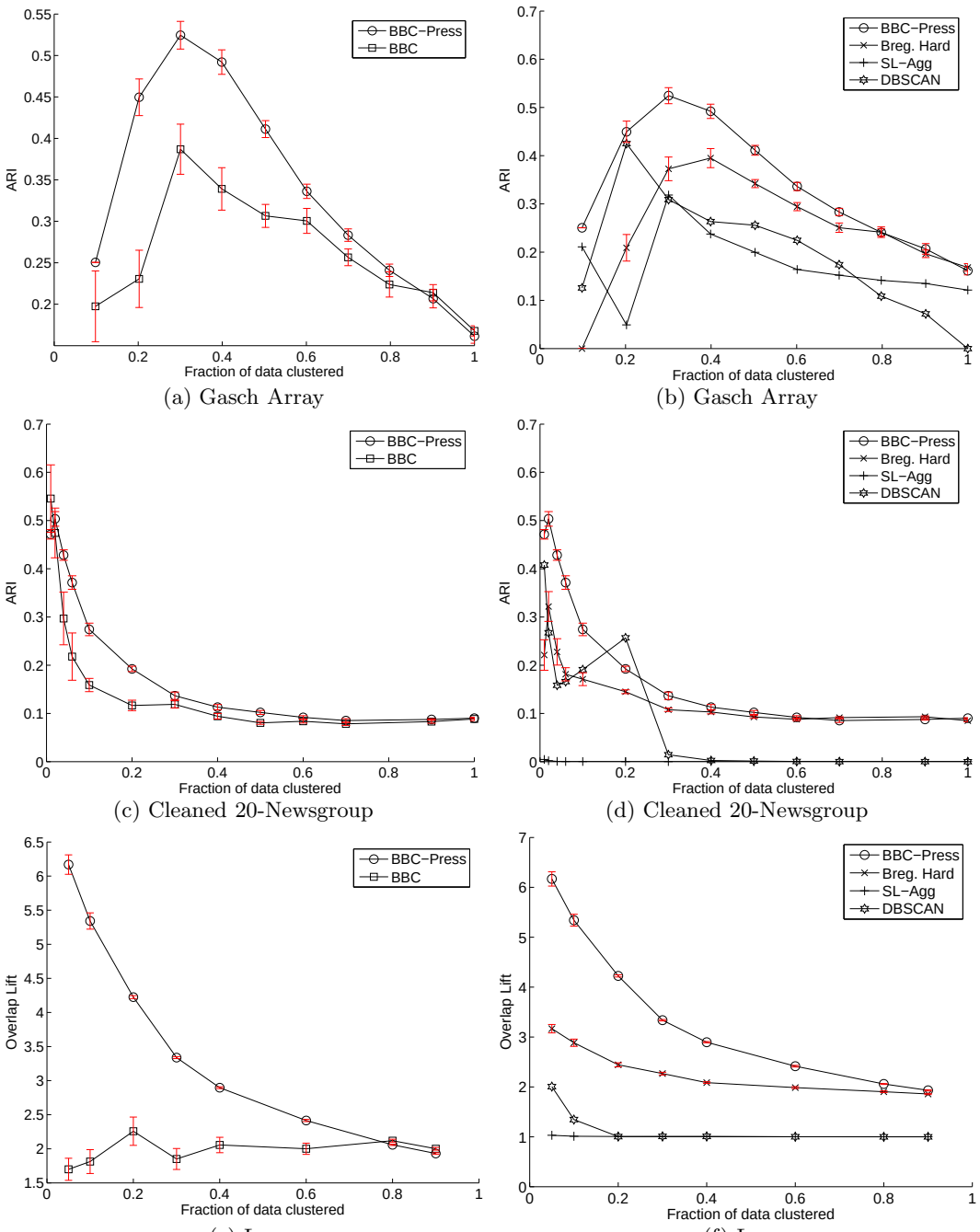


(a) Gauss-2 (d) Gauss-2

(b) Gauss-10 (e) Gauss-10

(c) Gauss-40 (f) Gauss-40

Fig. 6. 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 non-deterministic methods (i.e. excluding DBSCAN and Agglomerative) 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

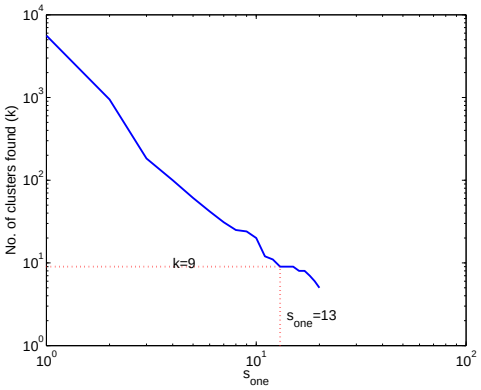
(e) Lee (f) Lee

Fig. 7. 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 (i.e. excluding DBSCAN and Agglomerative) 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.

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

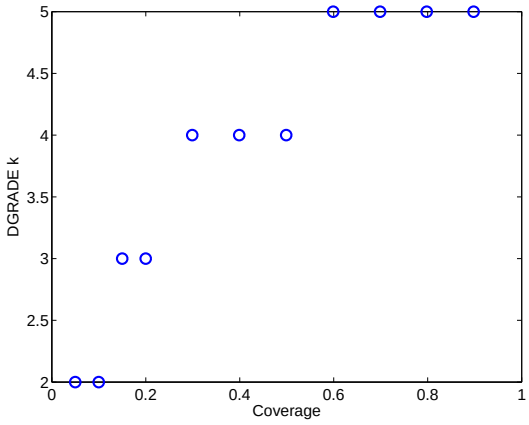
[a] Closest possible k to 12 found when $s_{one} = 3$. For $s_{one} = 3$, k increases to 14.
[b] See plot (b) in this figure.
[c] Figure 5(c) and (d)
[d] Figure 5(b)

(a)



(b)

Fig. 8. (a) Automatically determined s_{one} for DGRADE on various datasets, using the approach described in Section 9. (b) For the Lee data, selecting the largest k (smallest s_{one}) with $stability > 1$ gives a $stability = 3$, $s_{one} = 13$ and $k = 9$.

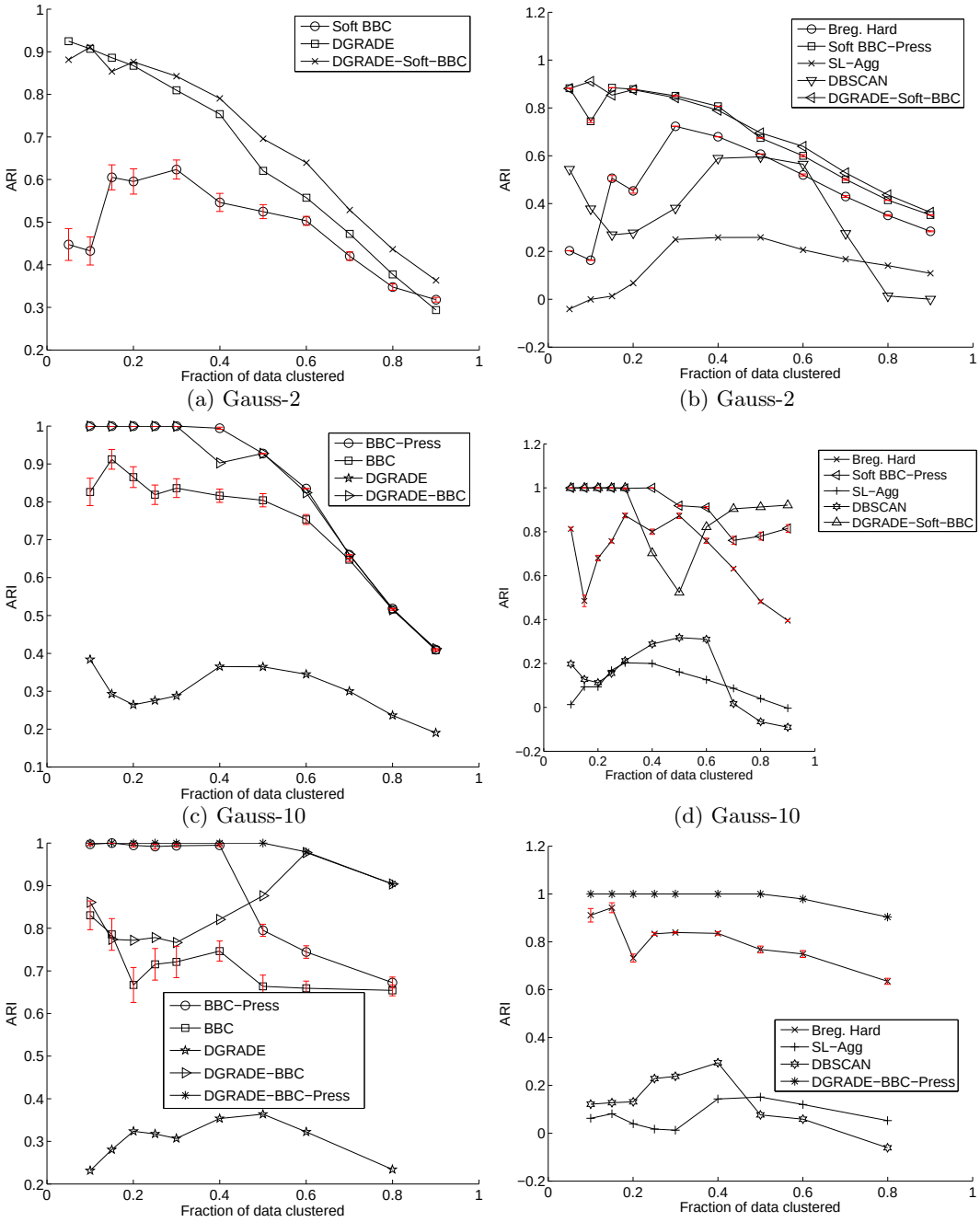


(a) Gauss-2

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
Gauss-10	0.1	3	1.0	5
Gauss-40	0.1	4	1.0	5

(b) Other datasets

Fig. 9. (a) On Gauss-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, Gauss-2, Gauss-10 and Gauss-40 datasets respectively.

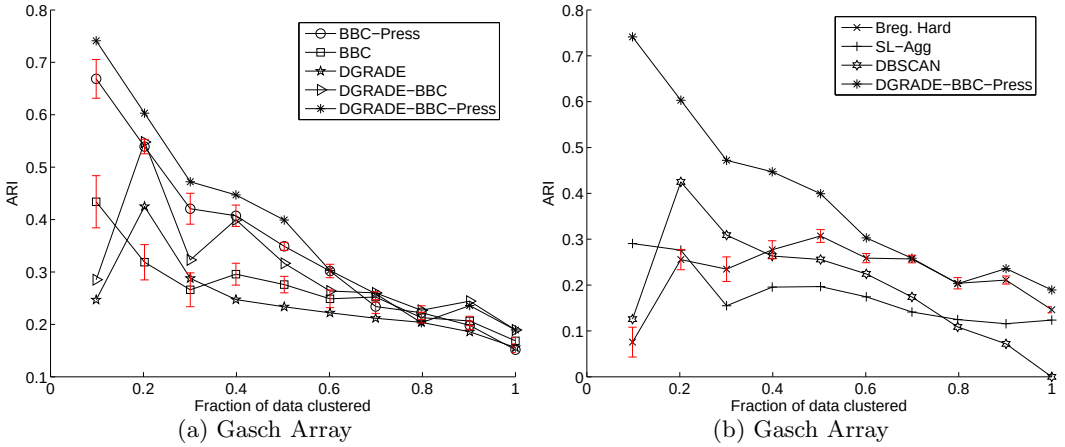


(a) Gauss-2 (b) Gauss-2

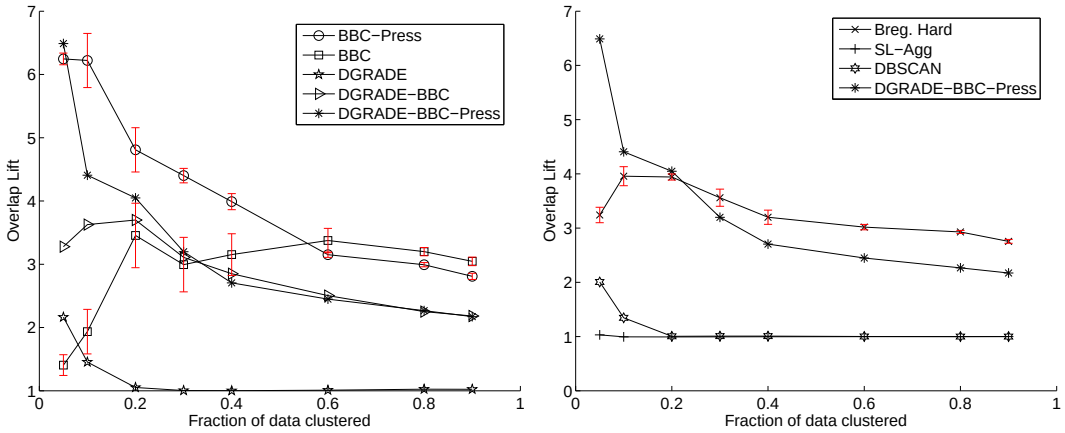
(c) Gauss-10 (d) Gauss-10

(e) Gauss-40 (f) Gauss-40

Fig. 10. 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 non-deterministic methods (i.e. all except DBSCAN and Agglomerative) for which ARI is plotted as the average over 100 trials with random initialization. Note: These experiments were setup differently from those without DGRADE; the k output by DGRADE was used as input to all algorithms except DBSCAN.



(a) Gasch Array (b) Gasch Array



(e) Lee (f) Lee

Fig. 11. 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). Note: These experiments were setup differently from those without DGRADE; the k output by DGRADE was used as input to all algorithms except DBSCAN.