Robust One-Class Clustering using Hybrid Global and Local Search

Gunjan Gupta GGUPTA@ECE.UTEXAS.EDU
Joydeep Ghosh GHOSH@ECE.UTEXAS.EDU
Department of Electrical & Computer Engineering, University of Texas at Austin, Austin, TX 78712 USA

Abstract

Unsupervised learning methods often involve summarizing the data using a small number of parameters. In certain domains, only a small subset of the available data is relevant for the problem. *One-Class Classification* or *One-Class Clustering* attempts to find a useful subset by locating a dense region in the data. In particular, a recently proposed algorithm called One-Class Information Ball (OC-IB) shows the advantage of modeling a small set of highly coherent points as opposed to pruning outliers. We present several modifications to OC-IB and integrate it with a global search that results in several improvements such as deterministic results, optimality guarantees, control over cluster size and extension to other cost functions. Empirical studies yield significantly better results on various real and artificial data.

1. Introduction

Unsupervised learning methods often involve summarizing the data using a small number of parameters. In some domains, only a small subset of the available data is relevant while the rest shows poor correlation to the problem and needs to be pruned for building a good model. One major type of data in bioinformatics that exhibits such properties is micro-array data recording an organism's biochemical response to a specific set of conditions such as stress (Gasch A. P. et al., 2000) or disease (Ash A. Alizadeh et al., 2000). Such a dataset consists of a group of experiments (arrays) capturing mRNA expression for a large number of genes. Typically, only a small subset of the measured genes show correlated (anomalous) expression across a subset of the experiments, when compared to a control experiment. In another application, Crammer and Chechik (2004) used One-Class Clustering to design a document retrieval system. Often, a reasonable sampling of only the positive examples of the relevant document classes is available, while the input space of irrelevant documents is hard to sample and/or is ill-defined. Furthermore, in such a system, a user is typically interested in retrieving a small list of highly relevant documents (high precision) rather than obtaining a list of all positive examples (high recall). In such a scenario, a One-Class Clustering based classifier designed to discover a dense region for each document class may be more suitable than building a supervised classifier.

Earlier work on One-Class clustering involved detection of large-scale structures (Schölkopf et al., 1995; Tax & Duin, 1999; Schölkopf et al., 2001). However, Crammer and Chechik (2004) subsequently showed that for many such problems (such as the ones mentioned above), the opposite approach of directly modeling the dense region, and ignoring points outside the region, is more appropriate. One way of modeling a small dense region is by searching for the location of a ball of small radius that contains a large number of data points. This was the strategy used in *One Class Information Ball* (OC-IB) (Crammer & Chechik, 2004), which tries to find a local minimum of an appropriate cost function, and starts the search by randomly selecting one of the data points. However, such a local search can get stuck in a bad local minimum. In this paper we present a simple yet powerful global search method using an approximation algorithm and use its output to initialize a local search. The "hybrid" method gives substantially superior results on various real and artificial datasets. Our method extends to some new types of cost functions, is deterministic and provides control over cluster size as well as optimality bounds for some of the cost functions proposed.

The rest of the paper is organized as follows: In Section 2 we define two general forms of cost functions that work with all *Bregman divergences.* In Section 3.1 we

Appearing in *Proceedings of the 22nd International Conference on Machine Learning*, Bonn, Germany, 2005. Copyright 2005 by the author(s)/owner(s).

present an algorithm that finds the global optima for an approximation of the original One-Class problem. In Section 3.2 we present a local search algorithm that can take either size or radius as input. In Section 3.3, we present a hybrid algorithm that we refer to as *Hyper-BB* that combines the local and global search methods. In Section 4 we present a simple extension of our method that works with a biologically relevant distance measure, and in Section 5 we present results on some real and artificial datasets.

2. Cost Function

Given a set Z of n points in $\mathbb{R}^d$, and a measure of one-class cluster cost, we define the problem of one-class clustering as one of the following:

Definition 1*: Find the cluster $G = \{\mathbf{p}_1, \mathbf{p}_2, .\mathbf{p}_i, .., \mathbf{p}_s\} \subset Z$ of size s that has the smallest cost.*

Definition 2*: Find the largest cluster G of cost less than or equal to q_{max}.*

In practice, definitions 1 and 2 are functionally similar, since for a given q_{max} there exists a largest s, and for a given s there exists a smallest q_{max}. The main difference from a user's point of view is that in definition 1 the algorithm takes an integer s as input and searches for a solution with the smallest cost, while in definition 2, it takes a threshold $q_{max} \in \mathbb{R}$ as input and searches for a cluster with the largest size with cost less than q_{max}.

Cost as a function of Distance: Given a distance [1] measure $D(\mathbf{x}, \mathbf{y}) \mapsto [0, \infty)$, and a cluster representative $\mathbf{c} \in \mathbb{R}^d$, we define the **Average Distance** cost Q_{AD} as the average distance of all points in G from $\mathbf{c}$:

$$Q_{AD}(G, \mathbf{c}) = \frac{1}{s} \sum_{i=1}^{s} D(\mathbf{p}_i, \mathbf{c}), \tag{1}$$

and the **Maximum Distance** cost Q_{MD} as the maximum distance among all points in G from $\mathbf{c}$:

$$Q_{MD}(G, \mathbf{c}) = \max_{i=1}^{s} D(\mathbf{p}_i, \mathbf{c}). \tag{2}$$

We shall see in Section 3.1 and 3.2 how the choice between Equation 1 and 2 affects the design of the algorithm for searching for a good cluster. Also note that our cluster cost is now uniquely defined by the set G and representative $\mathbf{c}$.

Choice of Distance Measure: *Bregman divergences* form a family of distance measures, defined as follows: Let $\phi : S \mapsto \mathbb{R}$ be a strictly convex function defined on a convex set $S \subseteq \mathbb{R}^d$, such that ϕ is differentiable on $int(S)$, the interior of S (Rockafeller, 1970). The Bregman divergence $D_\phi : S \times int(S) \mapsto [0, \inf)$ is defined as $D_\phi(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{x}) - \phi(\mathbf{y}) - (\mathbf{x} - \mathbf{y}, \bigtriangledown\phi(\mathbf{y}))$ 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 Bregman divergences such as Logistic Loss, Itakura-Saito Distance, Hinge Loss, Mahalanobis Distance and KL Divergence (Pietra et al., 2001; Banerjee et al., 2004). A k-means clustering algorithm based on iterative relocation that is applicable to all Bregman divergences has been recently described by Banerjee et al. (2004). The local search used by algorithms presented in this paper for one class clustering uses a different iterative relocation scheme that also generalizes to all Bregman divergences.

[1] In this paper, we use "distance" in an informal way to also include divergences that are not a metric.

Properties of Q_{AD} and Q_{MD}:

Proposition 2.1. *For a given s and $\mathbf{c}$, a cluster G that minimizes either Q_{AD} or Q_{MD} consists of the s points closest to $\mathbf{c}$.*

Proposition 2.1 follows from the observation that for any given center $\mathbf{c}$, for both Q_{AD} and Q_{MD}, adding the points closest to $\mathbf{c}$ first into the cluster increases the cost by the smallest amount.

Theorem 2.2. (Banerjee et al., 2004): *Let X be a random variable taking values in $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^n \subset C \subseteq \mathbb{R}^d$ following ν [2]. Given a Bregman divergence $D_\phi : C \times int(C) \mapsto [0, \inf)$, the problem*

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

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

When the cost function is Q_{AD}, and $D = D_\phi$, Theorem 2.2 allows us to compute the optimal center for a One-Class cluster G as the mean of the corresponding data points.

The cost function Q_{MD} for a cluster is uniquely defined by the cluster representative and the farthest point, and therefore the following is true:

Proposition 2.3. Order Preserving: *The solutions for Q_{MD} for any strictly monotonically increasing family of distance functions are identical.*

For example, for Euclidean Distance (a metric) versus the Squared Euclidean Distance (a Bregman divergence), the solutions for Q_{MD} are equivalent, with all solutions having the same same cost ordering. For

[2] Theorem 2.2 is more general in that it holds for any measure ν defined on the samples. For our one class clustering formulation, we assume all points to have the same weight.

Algorithm 1 HOCC

```
Input: Distance matrix $M$ containing distance between all points, desired list of cluster sizes $s_{list}$.
Output: Pairs of solution set containing indexes of best centroid followed by member points indices for each cluster size specified in $s_{list}$.
  $[radM, idxM] = sortrows(M)$
  for $j = 1$ to $|s_{list}|$ do
    $bestIdx = min(Q(\{radM(i, s_{list}(j)\}_{i=1}^{n}))$
    $solution(j) = \{bestIdx, \{idxM(bestIdx, j)\}_{j=1}^{s_{list}(j)}\}$
  end for
```

a given cluster G, Q_{MD} using the Squared Euclidean distance is simply the square of the Q_{MD} using Euclidean Distance. Furthermore, for Q_{MD}, the One-Class clustering problem reduces to finding a ball of smallest radius with s points in it, or finding a ball of a given radius that encloses maximum number of points.

3. Algorithms

For definition 1 described in Section 2, the best solution can be found exhaustively in $O(n^s)$ time. For most domains, this polynomial time solution for finding the global optima would be too slow. Rather, faster approximation algorithms that find a "good enough" solution are required.

3.1. Global Search: HOCC

The key idea is to perform a simpler exhaustive global search by restricting the cluster representative to be one of the data points, i.e. $\mathbf{c} \in Z$. Then, the following holds true for both cost functions Q_{AD} and Q_{MD}:

Proposition 3.1. *If* $\mathbf{c}$ *is restricted to* $\mathbf{c} \in Z$*, then the number of distinct* [3] *clusters of size* 2 *through* n *is* $n(n-1)$*, and can be enumerated.*

Proposition 3.1 is the basis of our Hypersphere One Class Clustering (HOCC, Algorithm 1) and follows from Proposition 2.1 based on the observation that the cluster representative and the farthest point determine members of G, and such a tuple can only be picked from Z. Such clusters could be visualized as bounded by "hyperspheres" in the corresponding divergence space and HOCC enumerates all such spheres and picks the one with the lowest cost for the specified size s. HOCC involves three steps: (1) sort elements of each row of M and save the corresponding matrices $radM$ and $idxM$ (represented by *sortrows* in Algorithm 1). (2) compute the cost Q for each position in $radM$, and (3) pick the cluster of size s from the s^{th} column that has the smallest Q. After step 1, the s^{th} sorted index of row i represents the farthest point in a cluster of size s with center as the i^{th} point. Picking the smallest element of the s^{th} column of matrix $Q(radM)$ then gives the optimal solution for size s. The only change when running HOCC for the maximum cost Q_{MD} versus the average cost Q_{AD} is the cost computed by a call to a function Q. For Q_{MD}, the value of element $radM(i, s)$ represents the cost, while the cumulative average $\frac{1}{s}\sum_{j=1}^{s} radM(i, j)$ represents the cost Q_{AD}. We can enumerate the lowest cost clusters of all size in $O(n^2 log(n))$ time, and if a *Heap* data structure is used, sorting can be performed only up to the maximum desired cluster size $m = max(s_{list})$ in $O(nmlog(m))$ time. In practice this could be much faster than $O(n^2 log(n))$, since the desired cluster size m is often much smaller than n. The memory requirement of the HOCC algorithm as presented is $O(n^2)$, but for a fixed cluster size s can be reduced to $O(n)$ if distances for each row sort are computed on the fly, only the s^{th} member of each sorted row is kept, and the $s-2$ nearest points (excluding the farthest and the center itself that are already known) for the optimal center are recomputed at the end. The time complexity for the whole process still remains $O(n^2 logn)$.

Optimality bound for Q_{MD} for Squared Euclidean Distance: Proposition 2.3 allows us to do perform the following analysis: If we put points in a cluster G on a spherical shell of radius r in a Euclidean Space, then the minimum Q_{MD} for Squared Euclidean Distance would be r^2, and the optimal cluster representative is the mean of the points in G. If we pick a cluster representative from G, the maximum distance to other points in G would be $2r$ and the Q_{MD} for Squared Euclidean distance would be $4r^2$. This is 4 times the optimal.

It is hard to give a general optimality bound for all Bregman divergences for Q_{MD}, but it is possible to do so for Q_{AD}:

Proposition 3.2. Optimality bound for Q_{AD}: *For a Bregman Divergence* D_ϕ *where* $l \le \phi'' \le u$*, the average distance cost* (Q_{MD}) *of the cluster found by HOCC algorithm is within* $1 + \frac{u}{l}$ *times the optimal solution.*

We can prove the above using the following lemma:

Lemma 3.3. (Banerjee et al., 2004) *For all Bregman Divergences* D_ϕ*, and points* $\{\mathbf{p}_i\}_{i=1}^{s}, \mathbf{w}, \mathbf{c}^* \in \mathbb{R}^d$*, where* $\mathbf{c}^*$ *is the minimizer for* Q_{AD} *for* $D = D_\phi$*, the following is true:*

$$\frac{1}{s}\sum_{i=1}^{s} D_\phi(\mathbf{p}_i, \mathbf{c}^*) + D_\phi(\mathbf{c}^*, \mathbf{w}) = \frac{1}{s}\sum_{i=1}^{s} D_\phi(\mathbf{p}_i, \mathbf{w}) \quad (3)$$

For the Squared Euclidean Distance, $u = l = 2$, therefore the above optimality guarantee reduces to within

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

2 times of optimal. For an unbounded divergence applied to a given dataset, it will be possible to bound l and u, in which case we would get corresponding optimality bounds.

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

3.2. Local Search: BBOCC

Our local search algorithm that we call Batch Ball One Class Clustering or BBOCC (Algorithm 2) is motivated along similar lines as the One-Class IB proposed by Crammer and Chechik (2004), but uses the update rule based on Theorem 2.2 to find the optimal center at each iteration. It also provides the ability to handle either Q_{AD} or Q_{MD} cost functions, and allows local search to be based on either a fixed cluster size or fixed cost. In Algorithm 2, a routine named *computeSize* is executed when q_{max} is passed as an input. Using proposition 2.1, *computeSize* finds the right cluster size in $O(n)$ and $O(logn)$ time respectively for Q_{AD} and Q_{MD} by searching for the largest cluster size that has cost below q_{max}.

Proposition 3.4. *Algorithm 2 is guaranteed to converge to a local minima for Q_{AD} for all Bregman divergences, for a fixed cost q_{max} as input.*

Proof. Let the set of members at iteration i be G_1. Let the center before update be $\mathbf{c}$, and cost be Q_{AD1}. Computing the optimal center $\mathbf{c}^*$ using Theorem 2.2 gives us a new cost $Q_{AD2} \leq Q_{AD1}$. Since $\mathbf{c}^*$ can be different than $\mathbf{c}$, there can be a point $\mathbf{f} \in G_1$ and another point $\mathbf{g} \notin G_1$ such that $D_\phi(c^*, g) \leq D_\phi(c^*, f)$. Let us assume this is true (if not, the algorithm has already converged). $G_2 = \{(G_1 \setminus \mathbf{f}) \cup \mathbf{g}\}$ has cost $Q_{AD3} \leq Q_{AD2} \leq Q_{AD1} \leq q_{max}$. Therefore, adding p closer points and removing p farthest points from the new center $\mathbf{c}^*$ will result in a cost $Q_{ADP} \leq Q_{AD1} \leq q_{max}$. Hence the number points in G cannot decline at each iteration, but can increase. If the members do not change in an iteration or if the cost fails do decline even with some membership change for two consecutive iterations, the algorithm converges. □

Algorithm 2 BBOCC

```
Input: Data $Z$, initial cluster center $\mathbf{c}_{in}$ (optional), desired
    cluster size $s$ or cost threshold $q_{max}$, distance function
    $D$, cost function $Q$.
Output: Solution set $G$ containing the member points,
    and the center $\mathbf{c}^*$.
  if $\mathbf{c}_{in} = \varnothing$ then
    Pick randomly a point from $Z$ and assign it to $\mathbf{c}_{in}$
  end if
  $\mathbf{c} = \mathbf{c}_{in}; lG = \varnothing; G = \varnothing; q_{pp} = \infty; q_p = \infty;$
  repeat
    for $i = 1$ to $n$ do
      $distAll(i) \leftarrow D(Z_i, \mathbf{c})$
    end for
    $[val, index] = sort(distAll)$
    Set $s_c = s$ else $s_c = computeSize(val, \mathbf{c}, Q, q_{max})$
    if $(Q = Q_{MD}) \wedge (val(s_c) \geq q_p)$ then
      $\mathbf{c} = \mathbf{c}_p$
      return
    else
      $\mathbf{c}_p = \mathbf{c}; q_{pp} = q_p; q_p = val(s_c)$
    end if
    $lG = G; G = Z(index(i))_{i=1}^{s_c}$
    $\mathbf{c} = \frac{1}{s_c}\sum_{i=1}^{s_c} Z(index(i))$
  until $(lG = G) \wedge q_{pp} = q_p$
  $\mathbf{c}^* = \mathbf{c}$
```

Along similar lines, we can also prove:

Proposition 3.5. *Algorithm 2 is guaranteed to converge to a local minima for Q_{AD} for all Bregman divergences, for a fixed cluster size s as input.*

For cost Q_{MD}, a call to function *computeSize* does not return an optimum center. In case of Euclidean distance, finding the optimal center is equivalent to finding the minimum bounding sphere. Since the computation of the minimum bounding sphere for a set of points is non-trivial, *computeSize* uses a simple heuristic: for good solutions involving dense regions, the points are likely to be symmetrically distributed, and in such cases the average center could be a good approximation (White & Jain, 1996). Thus, the function *computeSize* checks to see if computing the mean center reduces cost Q_{MD}, i.e. the distances from the farthest point. If it fails to find a smaller radius, the algorithm terminates. Therefore, for cost Q_{MD}, Algorithm 2 does not guarantee a local minima, but in practice, as we will see in Section 5, it seems to work well.

3.3. Hybrid Search: Hyper-BB

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

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

4. Extension to Pearson Distance

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 offsets), and is therefore a popular similarity measure for clustering gene-expression and other biological data (Sharan & Shamir, 2000; Mansson et al., 2004). For two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$, the Pearson correlation P can be computed as $P(\mathbf{x}, \mathbf{y}) = \frac{Z_s(\mathbf{x}) \bullet Z_s(\mathbf{y})}{d-1}$, where $Z_s(\mathbf{x})$ represents z-scoring [4] of $\mathbf{x}$ and is equal to $\frac{\mathbf{x}-\mu(\mathbf{x})}{\sigma(\mathbf{x})}$, $\mu(\mathbf{x})$ is the mean of the vector $\mathbf{x}$ and $\sigma(\mathbf{x})$ is the standard deviation, where σ is the unbiased estimator for a normal distribution. We define the *Pearson Distance* as $D_P = 1 - P$. Since $P \mapsto [-1, 1]$, therefore $D_P \mapsto [0, 2]$. It can be shown that Pearson Distance is equal to the Squared Euclidean Distance between z-scored points (between $Z_s(\mathbf{x})$ and $Z_s(\mathbf{y})$) normalized by $2(d-1)$:

$$D_P(\mathbf{x}, \mathbf{y}) = \frac{\| Z(\mathbf{x}) - Z(\mathbf{y}) \|^2}{2(d-1)} \tag{4}$$

Therefore, Pearson Distance is the Squared Euclidean Distance between the z-scored vectors projected onto a hypersphere of radius 1 (radius $\frac{1}{\sqrt{2}}$ in Euclidean space) centered at the origin.

Using HOCC and BBOCC with Pearson Distance: We refer to Q_{AD} and Q_{MD} as the *Average Pearson Distance* (APD) and the *Maximum Pearson Distance* (MPD) when $D = D_P$. The following directly follows from a proof given by Dhillon and Modha (2001):

Proposition 4.1. *The representative point $\mathbf{c}^*_P$ that minimizes APD is equal to the mean vector of the points in G projected onto a sphere of unit radius, i.e.* $\mathbf{c}^*_P = \underset{c}{argmin}(Q_{AD}(G, c)) = \frac{m_G}{\|m_G\|}$, *where* $m_G = \frac{1}{s}\sum_{i=1}^{s} Z(p_i)$.

[4]Often used in statistics, normally performed between points across a dimension. Here we perform it between dimensions for one data point.

Because of Proposition 4.1, for $D = D_P$ the optimal representative computation 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 Algorithm 2 makes it applicable to D_P, while Algorithm 1 can be used with Pearson Distance without any modifications. Furthermore, it can be shown that just like for Squared Euclidean Distance, the within two times optimality guarantee for HOCC and Hyper-BB also holds true for Pearson Distance.

5. Experimental Results

5.1. Datasets used

We test our algorithms on two real and two artificial datasets, and Table 1 summarizes the main aspects of these datasets. For real data, we chose to experiment with gene-expression data because we plan to incorporate Hyper-BB into methods for identifying biochemical pathways on such data.

Real Data: The two real data sets were obtained from Ash A. Alizadeh et al. (2000) and Gasch A. P. et al. (2000) respectively. The Alizadeh data consists of expression profile of 4,026 genes from 46 B-Cells Lymphoma cancer tissues, and also comes with the survival history for 40 patients, with 18 surviving long-term and 22 dying much earlier. The Gasch dataset 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. For each stress type there are multiple experiments with varying intensity of that stress, and all the 173 experiments are labeled by the type of stress that was applied. The Alizadeh data was also used by Crammer and Chechik (2004), and the Gasch data is a widely used benchmark for testing algorithms designed for clustering microarray data.

Simulated Data: We created two high-dimensional data-sets to test the hypothesis that a hybrid search becomes extremely important when the dense region is small in mass and is relatively isolated from the rest of the data. For this we created two high dimensional data-sets, each consisting of 40 dimensions and 4,026 data points. The *Hard* data consists of 3 spherical Gaussians with one of the Gaussians containing 5% of the probability mass, and being separated by a distance much larger than the standard deviation of two other wider and highly overlapping gaussians. For the

Table 1. A summary of the datasets used.

Data set	Source	n	d	Meaningful D
Alizadeh	Microarray	$4,026$	40	D_P
Gasch	Microarray	$6,151$	173	D_P
Hard	Simulated	$4,026$	40	Sq. Euclidean
Easy	Simulated	$4,026$	40	Sq. Euclidean

Easy dataset, we use the same 3 gaussians but place the small dense Gaussian close to the center of the rest of the data.

5.2. Evaluation

Effectiveness of Local Search: We compared OC-IB with BBOCC across multiple trials with various cluster sizes on Alizadeh data. Figure 1 (top) shows the difference in the cost of solutions for multiple trials over a range of cluster size for cost Q_{AD}, while the darker(pink) bars in Figure 1 (bottom) show the fraction of trials for which BBOCC gave results with lower cost than the lowest in all trials for OC-IB. Since the size of the cluster cannot be controlled in OC-IB, the comparison is across bucketed cluster sizes in the range shown on x-axis, with the number of trials in each of the 6 buckets varying between 95 and 126. OC-IB could not be tested on the gene-expression data with D_P since it only works with Bregman divergences, hence Squared Euclidean was used for this particular test. Such a test is still reasonable since we are performing an unsupervised evaluation. We found similar differences in performance between BBOCC and OC-IB for the other three datasets. We believe BBOC performs better because it uses an optimal center update at each iteration (Theorem 2.2).

Lesion Studies for Hyper-BB: We studied the importance of the local and global search in our hybrid algorithm by comparing Hyper-BB against running HOCC and BBOCC separately. We also compared our methods against "BBOC-10", where we pick the best of 10 trials of BBOC, and against a "Naive" algorithm that picks the s closest points to the mean of Z as the solution. For every tested cluster size, results of BBOCC and BBOCC-10 were averaged over 50 trials. Figure 2 shows the results on three different datasets for both Q_{AD} and Q_{MD} cost measures. Clearly, the global search (HOCC) by itself performs substantially better than the local search (BBOCC) for all cluster sizes, but even the combination (Hyper-BB) shows significant improvement over HOCC. The property that HOCC and Hyper-BB are deterministic while BBOCC's performance varies substantially across multiple trials (not plotted in Figure 2 for clarity) makes the improvement even more significant. On the Easy dataset (not shown) BBOCC performed as well as Hyper-BB for clusters of size larger than 10, while the naive algorithm that simply picks the data center gave results as good as Hyper-BB. But for the other three non-trivial datasets, the Hybrid algorithm clearly performs better than it's components, and even compared to BBOCC-10. One of the surprising things to notice is that for the Hard dataset (Figure 2, column 3) and for small cluster sizes Batch-BB performs worse than Naive, even though this dataset was not designed to have the dense region at the center of the data. These lesion studies confirm our intuition that both the local and global components of Hyper-BB are important in identifying small, dense regions.

Choice of cost function: We designed our algorithms to work for Q_{MD} as an alternative to Q_{AD}. The most surprising result was the discovery that for both Squared Euclidean Distance and Pearson Distance, Hyper-BB results on high dimensional datasets are almost independent of the cost function used to train Hyper-BB and and the one used to evaluate the results. Figure 3 shows that the results for various cluster sizes for matching and mismatching cost functions give almost identical performance for Hyper-BB. We found this to be true for all the four datasets, although plots for only two of them are shown. Our explanation for this phenomena is that since all the four datasets were high dimensional, and since Hyper-BB finds near-optimal clusters, the points mostly lie within a thin spherical shell, making the max and average distance almost equal. This tells us that given the local and global guarantees for Q_{AD}, it should be the preferred cost function for Hyper-BB.

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

6. Concluding Remarks

The HOCC enumeration algorithm is a fast global approximation and provides optimality bounds under certain conditions, while our local search BBOCC

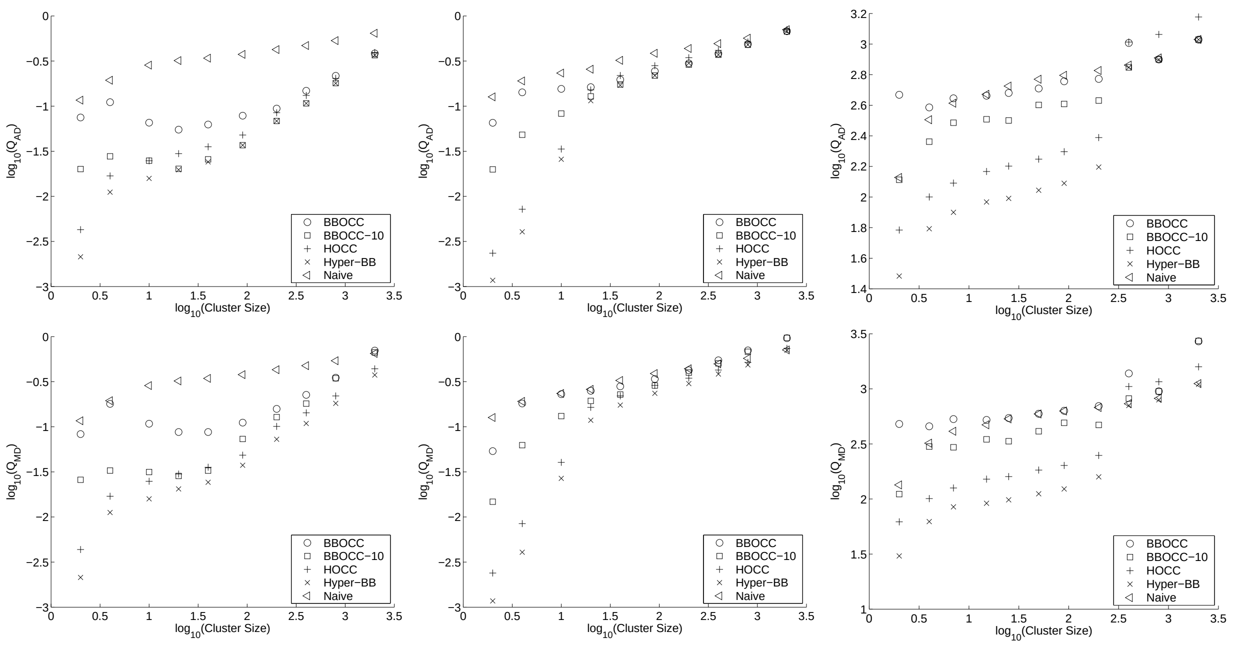

Figure 2. Comparison of performance of BBOCC, BBOCC-10, HOCC, Hyper-BB and the Naive algorithm using Q_{AD} (first row) and Q_{MD} (second row) as training and evaluation cost functions on three datasets: Gasch (left column), Alizadeh (center) and Hard(right). The y-axis is the evaluation cost, while the x-axis is the size of the cluster for which the evaluation was performed. For BBOCC and BBOC-10, the results were averaged over 50 trials. The variance in the performance of BBOC and BBOC-10 over the 50 trials was too large to plot meaningfully.

shows empirical improvement in the quality of the local minima compared to OC-IB. Both HOCC and BBOCC can take either a threshold cost or cluster size as input. Hyper-BB is a combination of HOCC and BBOC and inherits desirable properties of the two components such as optimality bound, deterministic output, and the ability to take cluster size or cost threshold as input. Empirical results show that both HOCC and BBOCC are important components of Hyper-BB, and that Hyper-BB is excellent at finding dense regions, with dramatically better results than a local search in situations where the dense region is small and isolated. We extended Hyper-BB to work on Pearson Distance that directly optimizes the Pearson Correlation, an important measure for clustering many kinds of biological data.

A key issue not addressed in this paper is that of model selection, which in this setting translates to the appropriate choice of size s or cost q_{max}. Several approaches readily suggest themselves, especially given that our methods incrementally grow the clusters until the threshold s or q_{max} is exceeded. For example, since we are seeking a cluster that is dense relative to the background, statistical testing against a null hypothesis that the cluster points are coming from the same distribution as the background, can readily be carried out. In other situations, the application domain dictates the choice of s.

Although we do not discuss it in this paper due to lack of space, our method can be extended to a diametric (Dhillon et al., 2003) version that captures both correlated and anti-correlated genes. We plan to apply one-class clustering to the problem of feature selection for gene expression data and answer questions such as "Is a biochemical pathway over or under-active in a set of gene-expression data, and if it is, which of the subset of experiments are those?" Another extension would be to use one-class clustering as a component to solve more complex clustering settings such as semi-supervised clustering, coclustering, and clustering with feature selection. We feel that given the simplicity and elegance of one-class clustering, such extensions might present a new way of looking at clustering in general.

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

References

Ash A. Alizadeh et al. (2000). Distinct types of diffuse large b-cell lymphoma identified by gene expression profiling. *Nature*, *403*, 503–511.

Banerjee, A., Merugu, S., Dhillon, I., & Ghosh, J.

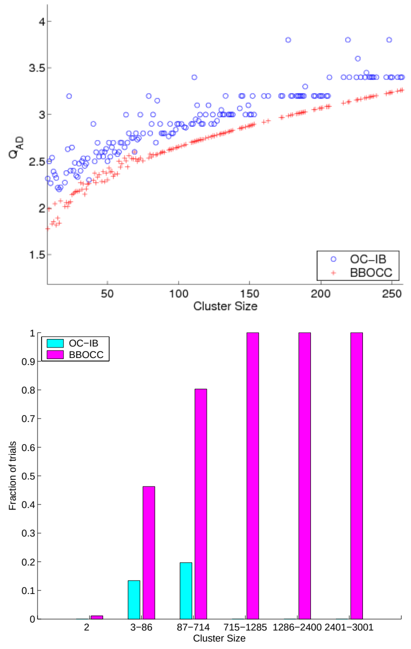

Figure 1. Comparison of performance of OC-IB (Crammer & Chechik, 2004) w.r.t. BBOCC on Alizadeh data: (top) Actual cluster cost distributions for various cluster sizes. (bottom) For 6 bucketed cluster sizes (x-axis), y-axis represents the fraction of trials in a range of cluster size that one algorithm had lower cost than the smallest of the other.

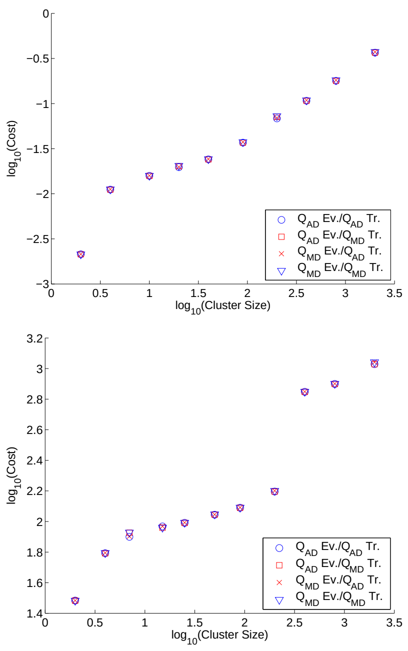

Figure 3. Evaluation of results using various combinations of Q_{AD} and Q_{MD} where training cost function matches evaluation (o & ∇) and where it does not match (x & $\Box$), for Gasch (top) and Hard (bottom) datasets. In the legends, Ev. stands for Evaluation and Tr. for Training.

(2004). Clustering with Bregman divergences. *Proc. SDM2004* (pp. 234–245).

Crammer, K., & Chechik, G. (2004). A needle in a haystack: Local one-class optimization. *ICML*. Banff, Alberta, Canada.

Dhillon, I. S., Marcotte, E. M., & Roshan, U. (2003). Diametrical clustering for identifying anti-correlated gene clusters. *Bioinformatics*, *19*, 1612–1619.

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

Gasch A. P. et al. (2000). Genomic expression programs in the response of yeast cells to environmental changes. *Molecular Biology of the Cell*, *11*, 4241–4257.

Mansson, R., Tsapogas, P., Akerlund, M., & et. al. (2004). Pearson correlation analysis of microarray data allows for the identification of genetic targets for early b-cell factor. *J. Biol. Chem.*, *279*, 17905–17913.

Pietra, S. D., Pietra, V. D., & Lafferty, J. (2001). Duality and auxiliary functions for bregman distances. *Technical Report CMU-CS-01-109, School of Computer Science*. Carnegie Mellon University.

Rockafeller, R. T. (1970). *Convex analysis*. Princeton University Press.

Schölkopf, B., Burges, C., & Vapnik, V. (1995). Extracting support data for a given task. *KDD*. Menlo Park, CA: AAAI Press.

Schölkopf, B., Platt, J. C., Shawe-Taylor, J. S., Smola, A. J., & Williamson, R. C. (2001). Estimating the support of a high-dimensional distribution. *Neural Computation*, *13*, 1443–1471.

Sharan, R., & Shamir, R. (2000). Click: A clustering algorithm with applications to gene expression analysis. *Proc. 8th ISMB*, 307–316.

Tax, D., & Duin, R. (1999). Data domain description using support vectors. *Proceedings of the European Symposium on Artificial Neural Networks* (pp. 251–256).

White, D., & Jain, R. (1996). Similarity indexing with ss-tree. *12th International Conf. on Data Engineering* (pp. 516 – 523). New Orleans.