k-means--: A unified approach to clustering and outlier detection

Sanjay Chawla* Aristides Gionis†

Abstract

We present a unified approach for simultaneously clustering and discovering outliers in data. Our approach is formalized as a generalization of the k-MEANS problem. We prove that the problem is **NP**-hard and then present a practical polynomial time algorithm, which is guaranteed to converge to a local optimum. Furthermore we extend our approach to all distance measures that can be expressed in the form of a Bregman divergence. Experiments on synthetic and real datasets demonstrate the effectiveness of our approach and the utility of carrying out both clustering and outlier detection in a concurrent manner. In particular on the famous KDD cup network-intrusion dataset, we were able to increase the precision of the outlier detection task by nearly 100% compared to the classical nearest-neighbor approach.

1 Introduction

Despite their close complementarity, clustering and anomaly detection are often treated as separate problems in the data-mining community. This distinction is not without justification. Often applications are defined in terms of outliers (like in fraud detection, network anomaly detection, etc.) in which case a direct approach is likely to be more efficient [13, 14]. However, an important point worth considering is that if there is no inherent clustering in the data it is unlikely that there exist any natural outliers.

In this paper we will propose a generalization of the k-MEANS problem with the aim of simultaneously clustering data and discovering outliers. A naïve approach is to apply the k-means algorithm and list as outliers the top ℓ points that are the furthest away from their nearest cluster centers. However, there is a subtle point that needs to be noted: the k-means algorithm itself is extremely sensitive to outliers, and such outliers may have a disproportionate impact on the final cluster configuration. This can result in many false negatives: i.e., data points that should be declared outliers are masked by the clustering and also false positives: data points that are incorrectly labeled as outliers. Thus what is required is a more robust version of the k-means algorithm that gracefully handles the presence of outliers.

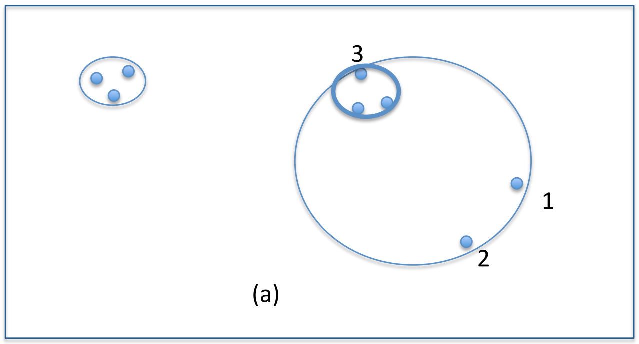

Figure 1: The k-means algorithm is extremely sensitive to outliers. By removing two points (1) and (2), we can obtain much tighter clusters (the bold circle near 3). The objective of this paper is to obtain a tight clustering and report the outliers in an automatic fashion.

Figure 1 shows a hypothetical scenario where the k-means algorithm could potentially be severely affected by the presence of outliers. In Figure 1(a), if $k = 2$, the k-means will force the five data points on the right into one cluster. On the other hand, if the k-means algorithm is designed to simultaneously form clusters and track outliers, a more natural outcome is also shown in Figure 1(a), where the big cluster has become more compact (bold circle) and the two data points (1) and (2) are the outliers. This is precisely what the method proposed in this paper will be able to achieve in a computationally efficient manner.

Our main contributions are the following:

- We formulate the (k, ℓ)-MEANS problem of simultaneously discovering clusters and outliers as an **NP**-hard optimization problem. The optimization problem paves the way for a systematic and formal analysis of the outlier detection problem.
- We propose the k-means-- algorithm,[1] which given an input parameter of (k, ℓ), will discover k clusters

*University of Sydney, Australia.

†Aalto University, Finland. This work was done while the author was with Yahoo! Research.

[1]Our algorithm is pronounced "k means minus minus" and in LaTeX is typeset as `$k$-means{-}{-}`.

and ℓ outliers in a unified fashion. We show that the k-means-- algorithm provably converges to a local optima and the running time is linear in the number of data points. The algorithm extends to cases when the similarity metric is a Bregman divergence.

- We have tested the effectiveness of k-means-- on a large class of synthetic and real datasets, including an interesting dataset about hurricanes originating in the Atlantic Ocean in the last five decades.

The rest of the paper is structured as follows. In Section 2 we describe related work with a special emphasis on the use of a robust version of Mahalanobis distance to finding outliers. The (k, ℓ)-MEANS problem is formalized in Section 3. In Section 4 we describe our algorithm and prove its properties. In Section 5 we report on a comprehensive set of experiments on synthetic and real data. Section 6 is a short conclusion.

2 Related work

Outlier detection is a deeply researched problem in both communities of statistics and data mining [5, 11] — but with different perspectives.

In data mining, Knorr and Ng [12] proposed a definition of distance-based outlier, which is free of any distributional assumptions and is generalizable to multidimensional datasets. Intuitively, outliers are data points that are far away from their nearest neighbors.

Following Knorr and Ng, several variations and algorithms have been proposed to detect distance-based outliers [2, 12, 17]. However, the outliers detected by these methods are *global outliers*, i.e., the "outlierness" is with respect to the whole dataset. Breunig et al. [4] have argued that in some situations local outliers are more important than global outliers and cannot be easily detected by standard distance-based techniques. They introduced the concept of local outlier factor (*LOF*), which captures how isolated an object is with respect to its surrounding neighborhood. The concept of local outliers has subsequently been extended in several directions [5, 7, 16].

Classical Multivariate Outlier Detection. We now briefly describe the classical statistical approach for finding anomalies in multivariate data as it is germane to our approach: Given an $N \times d$ dataset (N rows, d columns), the (square of) Mahalanobis Distance between two points x and y is given as

$$D_M(x, y) = (x - y)\Sigma^{-1}(x - y)^T,$$

where Σ is the $d \times d$ covariance matrix.

For a set S of N d-dimensional points from a normal distribution, the square of the Mahalanobis distance follows a χ^2_d distribution with d degrees of freedom. It has been experimentally observed that even when the data does not strictly follow the normal distribution, the chi-square distribution is still a good approximation.

As in the case of k-means it is well known that both the mean and standard deviation are extremely sensitive to outliers, and one "bad point" can skew the mean and the variance. We are using the Mahalanobis distance to find outliers and yet it itself is being effected by outliers. Thus, a "robustification" procedure is necessary to make the statistical estimator less sensitive to outliers.

In the statistics literature, the most famous method for performing such a robustification process is the Minimum Covariance Determinant (MCD) [18]. The MCD estimator is determined by a subset of points of size $N - \ell$, which minimizes the determinant of the variance-covariance matrix *over all subsets of size* $N - \ell$. Recall that the determinant of a $d \times d$ matrix Σ is also given by the product of its eigenvalues, $\det(\Sigma) = \lambda_1\lambda_2 \ldots \lambda_d$. Thus the objective is to identify is a set of size $N - \ell$ for which the determinant is minimum. The MCD algorithm is based on the following observation:

THEOREM 2.1. *[18] Let S be an $N \times d$ dataset. Let S_1 be a subset of S of size $N - \ell$, and let μ_{S_1} and Σ_{S_1} be the mean and covariance matrix of S_1. Compute, the Mahalanobis distance of all points of S based on μ_{S_1} and Σ_{S_1}. Sort the Mahalanobis distance in decreasing order and select $N - \ell$ points with smallest distance. Let the set be S_2. Then*

$$\det(S_2) \leq \det(S_1).$$

This is a k-means-type of algorithm for $k = 1$. At each iteration the value of the determinant does not increase. However the computation of the determinant requires a computation of order $O(d^3)$ and is practically infeasible for high-dimensional datasets.

Complexity of MCD. The MCD algorithm proposed by Rousseeuw [18] only guarantees a local optima. The computational complexity of MCD was settled by Bernholt et al. [3] who first proposed a polynomial time algorithm of complexity $O(N^{d^2})$ and then proved that the decision version of MCD when the dimension of the data varies is **NP**-hard. Clearly an $O(N^{d^2})$ algorithm, while polynomial time, is not practical for modern data-mining applications.

One of the most interesting aspects of the proposed approach is that it provides an optimization framework for an integrated approach to clustering and outlier detection. Similar optimization problems have been studied in the theoretical computer-science community, in the context of p-center, k-median, and facility-location problems [6][8]. In this line of research, approximation algorithms have been proposed, but their running times are not practical for very large datasets as they often require a reduction which entails solving a very large linear program. Our proposed algorithm is a highly scalable algorithm, very easy to implement, and in the spirit of the practical k-means setting, which is one of the most commonly used algorithms in data mining.

3 Problem definition

In this section we introduce our notation and we present our problem statement for clustering data with outliers.

We consider a dataset of n points $X = \{\mathbf{x}_1, ..., \mathbf{x}_n\}$. We assume a distance function $d : X \times X \to \mathbb{R}$ defined over pairs of points of X. The most common setting to consider is when the dataset X consists of points in the d-dimensional Euclidean space $\mathbb{R}^d$, and the distance function between pairs of points in X is defined as the Euclidean distance $d(\mathbf{x}_i, \mathbf{x}_j) = (\sum_{t=1}^{d} |x_{it} - x_{jt}|^2)^{\frac{1}{2}}$, where $\mathbf{x}_i = (x_{i1}, \ldots, x_{id})$ is the representation of $\mathbf{x}_i$ in $\mathbb{R}^d$, or in general, any other L_p-norm-based distance. Under certain assumptions, the base setting can be extended to more general cases, in which the dataset X consists of points in a *metric space* and the distance d is a metric function in the underlying space. Extensions to non-metric similarity measures (like Bregman divergence) are also possible [1].

Consider next a set $C = \{\mathbf{c}_1, \ldots, \mathbf{c}_k\}$ of k points, representing potential *cluster centers* that can be used for clustering the dataset X. For each $\mathbf{x} \in X$ we define

$$c(\mathbf{x} \mid C) = \arg\min_{\mathbf{c} \in C}\{d(\mathbf{x}, \mathbf{c})\},$$

the closest center of $\mathbf{x}$ in C. The distance of $\mathbf{x} \in X$ to its closest center $c(\mathbf{x} \mid C)$ is simply

$$d(\mathbf{x} \mid C) = \min_{\mathbf{c} \in C}\{d(\mathbf{x}, \mathbf{c})\} = d(\mathbf{x}, c(\mathbf{x} \mid C)).$$

We define the *error* of clustering the set X using as centers the k-point set C as

$$E(X, C) = \sum_{\mathbf{x} \in X} d(\mathbf{x} \mid C)^2. \tag{3.1}$$

Note that in the Euclidean setting the set of cluster centers C may consist of any points of $\mathbb{R}^d$. In the metric-space setting, sometimes it is necessary to restrict C to be a subset of the points X provided in the input. The latter restriction can be overcome when it is possible to "construct" new objects in the underlying space of X. For instance, for clustering a set of strings it is possible to consider the cluster centers to be new strings that are not part of the input; any string over the same alphabet as the input strings is a candidate cluster center.

The k-MEANS problem is defined as follows.

PROBLEM 1. (k-MEANS) *Given a set of points $X = \{\mathbf{x}_1, ..., \mathbf{x}_n\}$, a distance function $d : X \times X \to \mathbb{R}$ and a number k, find a set of k points $C = \{\mathbf{c}_1, \ldots, \mathbf{c}_k\}$ in order to minimize the error $E(X, C)$, defined in (3.1).*

Note the differentiation between k-MEANS, the optimization problem, and k-means, the popular algorithm used to find a solution to the k-MEANS problem.

We now define the problem of clustering with outliers. We consider two parameters: k the number of clusters, and ℓ the number of outliers. We refer to our optimization problem as (k, ℓ)-MEANS. As in the k-MEANS problem our goal is to find a set of k cluster center points C, which can be subset of X or points in the underlying space. Additionally we aim at identifying a set $L \subseteq X$ of outliers and we require $|L| = \ell$.

PROBLEM 2. ((k, ℓ)-MEANS) *Given a set of points $X = \{\mathbf{x}_1, ..., \mathbf{x}_n\}$, a distance function $d : X \times X \to \mathbb{R}$ and numbers k and ℓ, find a set of k points $C = \{\mathbf{c}_1, \ldots, \mathbf{c}_k\}$ and a set of ℓ points $L \subseteq X$ so as to minimize the error*

$$E(X, C, L) = E(X \setminus L, C). \tag{3.2}$$

Intuitively, in the (k, ℓ)-MEANS problem we want to find the best clustering of X, defined by cluster centers C, *after* we identify the set of outliers $L \subseteq X$. The inherent difficulty of this problem is that

(i) the outlier set L depends on the set of cluster centers C — the points in L should be those points that are "far away" from the points in C, and

(ii) the set of centers C depends on the outlier set L — if we knew the outlier set L we could run a standard k-means algorithm on the remaining points $X \setminus L$.

It should come as no surprise that the (k, ℓ)-MEANS is as hard as the k-MEANS.

THEOREM 3.1. *The (k, ℓ)-MEANS problem is* **NP**-*hard for $k > 1$ and $\ell \geq 0$. For $k = 1$ and $l > 1$ and fixed dimension d, there exists a polynomial time algorithm of complexity $O(n^{d^3})$. For varying dimension d the problem is* **NP**-*hard.*

Proof. For $\ell = 0$ the (k, ℓ)-MEANS problem becomes identical to the k-MEANS problem, which is known to be **NP**-hard for general metric spaces, or for the Euclidean case when the dimension is $d > 1$. For $k = 1$ and $\ell \geq 1$, the problem is a special case of the MCD problem.

Discussion. 1. In this paper we treat finding the parameters k and ℓ as an orthogonal problem, and we assume that they are given as input to our algorithm. Finding k is an open problem in data mining, although some approaches give reasonable results, for example the *minimum-description length* principle (MDL) and *Bayesian information criterion* (BIC).

For the outlier parameter ℓ, we note that all existing outlier-detection methods use one or more parameters. For example, if data is modeled as being generated from a normal distribution $\mathcal{N}$, then a data point x is an outlier if $p_{\mathcal{N}}(x) < 0.001$. Here 0.001 is the parameter. Often the application requirement is to report the top-ℓ outliers; and this is the approach we have taken. Sometimes more than one parameter is used. For example when PCA techniques are used for anomaly detection then we have to set two parameters (k, t). Here k represents the rank of the approximation and t represents the distance threshold.

2. The k-means algorithm is a widely-used algorithm, applicable to many different settings. For example, k-means is a component for *spectral clustering*, which is used to cluster not only objects in arbitrary metric spaces, but also Euclidean points lying on non-linear manifolds and having "local" cluster structure. Our algorithm gives a principled approach to cluster with outliers in all those different settings that k-MEANS is used. For example, replacing k-MEANS with (k, ℓ)-MEANS in above-mentioned component of spectral clustering allows to perform spectral clustering with outlier detection. Such an example is presented in Section 5, where we use the k-means-- algorithm in a spectral-clustering setting to identify outlier trajectories.

4 Algorithm

The proposed algorithm, which we call k-means--, is an extension of the k-means algorithm. The pseudocode is shown as Algorithm 4.1

The algorithm works as follows. The initialization part (lines 1-4) is identical to k-means. In line 5, all the points are ranked in decreasing order by their distance to their nearest cluster center. In lines 6 and 7, the top ℓ points of this ranked list are inserted into L_i which is then removed from the set to form the set X_i. In lines 8 to 9, each of the element of X_i is assigned to its nearest cluster to form a new set P_j for j ranging from 1 to k. New cluster centers are estimated from P_j in line 10, and the process is repeated till the solution stabilizes.

We are able to show that the k-means-- algorithm converges to a locally optimal solution. To see why such a local-optimality property holds, we note that in the i-th iteration the algorithm computes a new set of outliers L_i and a new set of cluster centers C_i. The outlier set L_i is computed in line 7, and the cluster-center set C_i is computed in lines 7–11. We can that in each consecutive update of L_i and C_i the total error improves. The proofs of the lemmas are given in the supplementary document.

ALGORITHM 4.1. (k-MEANS--)

```
Input: Set of points $X = \{\mathbf{x}_1, ..., \mathbf{x}_n\}$
    A distance function $d : X \times X \to \mathbb{R}$
    Numbers $k$ and $\ell$
Output: A set of $k$ cluster centers $C$
    A set of $\ell$ outliers $L \subseteq X$
1: $C_0 \leftarrow \{k$ random points of $X\}$
2: $i \leftarrow 1$
3: while (no convergence achieved) do
4:     Compute $d(\mathbf{x} \mid C_{i-1})$, for all $\mathbf{x} \in X$
5:     Re-order the points in $X$ such that
       $d(\mathbf{x}_1 \mid C_{i-1}) \geq \ldots \geq d(\mathbf{x}_n \mid C_{i-1})$
6:     $L_i \leftarrow \{\mathbf{x}_1, \ldots, \mathbf{x}_\ell\}$
7:     $X_i \leftarrow X \setminus L_i = \{\mathbf{x}_{\ell+1}, \ldots, \mathbf{x}_n\}$
8:     for ($j \in \{1, \ldots, k\}$) do
9:         $P_j \leftarrow \{\mathbf{x} \in X_i \mid c(\mathbf{x} \mid C_{i-1}) = \mathbf{c}_{i-1,j}\}$
10:        $\mathbf{c}_{i,j} \leftarrow \mathrm{mean}(P_j)$
11:    $C_i \leftarrow \{\mathbf{c}_{i,1}, \ldots, \mathbf{c}_{i,k}\}$
12:    $i \leftarrow i + 1$
```

LEMMA 4.1. $E(X, C_{i-1}, L_i) \leq E(X, C_{i-1}, L_{i-1})$.

LEMMA 4.2. $E(X, C_i, L_i) \leq E(X, C_{i-1}, L_i)$.

Combining Lemmas 4.1 and 4.2 we have the following.

THEOREM 4.1. *The k-means-- algorithm converges to a local optimum.*

4.1 Extension to Bregman divergences. The k-means-- algorithm is applicable to any Bregman divergence in a straightforward manner. Furthermore, it can be shown that it still converges to a local optimum. This result is analogous to the fact that a k-means-type algorithm converges to a local optimum for any Bregman divergence, as it was shown by Banerjee et al. [1].

We remind that the Bregman divergence $d_\phi : S \times \mathrm{ri}(S) \to [0, +\infty)$ is defined as

$$d_\phi(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{x}) - \phi(\mathbf{y}) - \langle \mathbf{x} - \mathbf{y}, \nabla\phi(\mathbf{y})\rangle,$$

where $\phi : S \to \mathbb{R}$ is a strictly convex function defined on a convex set $S \subseteq \mathbb{R}^d$, which is differentiable on the relative interior set $\mathrm{ri}(S)$, and $\nabla\phi(\mathbf{y})$ represents the gradient vector of ϕ evaluated at $\mathbf{y}$. Many well-known measures are special cases of Bregman divergence, for example, the squared Euclidean distance, which we have considered so far in this paper, the KL-divergence, the Mahalanobis distance, the Logistic loss, and others. For more details please see the work of Banerjee et al. [1].

One important property of the Bregman divergence is that given a finite set of points $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ one can optimize the Bregman information

$$I_\phi(X) = \min_{\mathbf{s} \in \mathrm{ri}(S)} \sum_{i=1}^{m} d_\phi(\mathbf{x}_i, \mathbf{s}),$$

that is, it is possible to find the optimal representative $\mathbf{s}$ for the set X. In fact, the representative $\mathbf{s}$ is the mean of X, independently of the Bregman divergence used.

Thus, to apply the k-means-- algorithm for a Bregman divergence, we only need to use $d_\phi(\cdot,\cdot)$ instead of the squared Euclidean distance function $d(\cdot,\cdot)^2$. We are able to show that the proofs of Lemmas 4.1 and 4.2 still hold, and thus we have the following.

THEOREM 4.2. *The k-means-- algorithm converges to a local optimum for any Bregman divergence.*

5 Experiments

We validate our proposed algorithm on a mix of synthetic and real datasets.

5.1 Synthetic data. We generate synthetic datasets as follows. We start with parameters k for the number of clusters, m for the number of points per cluster, ℓ for the number of outliers, d for dimension, and σ as a sampling error parameter. We first sample k cluster centers $C^* = \{c_1^*, \ldots, c_k^*\}$ in the space $[0,1]^d$. These are the *ground-truth* cluster centers. We then generate m points per cluster by sampling each coordinate from the normal distribution $\mathcal{N}(0,\sigma)$ and adding it to the corresponding cluster center. We finally sample ℓ outliers, uniformly at random, from the space $[0,1]^d$.

We then run the k-means-- algorithm on the synthetic datasets, using the correct values of the parameters k and ℓ. As already discussed, determining these values is a very hard problem, which we do not study in this paper. We assume that the data miner has an estimate of these parameters from the application domain, or that he/she experiments with different values.

We evaluate the performance of the algorithm by varying different parameters of the data-generation process. Our objective is to create increasingly difficult settings so that the outliers eventually become indistinguishable from the points that belong to clusters.

To evaluate the performance of our algorithm we use three measures. The first measure is the distance ratio $R_N = d(\mathbf{x} \mid C)/d(\mathbf{x} \mid C^*)$ averaged over all *non outliers* reported by our algorithm. Here C are the cluster centers found by our algorithm and C^* are the ground truth outliers. The smaller the value of R_N the better is the performance of our algorithm. In fact, $R_N < 1$ indicates that our algorithm found a solution in which the distances of the non-outlier points is smaller than that in the ground truth solution.

The second measure is the same distance ratio $R_O = d(\mathbf{x} \mid C)/d(\mathbf{x} \mid C^*)$ but this time averaged over all *outliers* reported by our algorithm. In this case, the larger the value of R_O the better is the performance of our algorithm, since it means that the algorithm finds outliers that are further away from their cluster centers than in the ground-truth solution.

The final measure is the Jaccard coefficient between the outliers found by our algorithm and the ground-truth outliers. Thus if O is the set of outliers returned by the outliers and O^* are the ground-truth outliers, the the Jaccard coefficient (J) is defined as

$$J(O, O^*) = \frac{|O \cap O^*|}{|O \cup O^*|}$$

The results of our experiments are shown in Figures 2, 3, and 4, in which we vary the parameters d, ℓ, and σ, respectively. The rest of the parameters remain fixed for each setting and are reported in the captions of the figures. In each figure we show the three measures described above, R_N, R_O and J. The box-plots indicate running each experiment 30 times.

We see that the performance of the k-means-- algorithm is extremely good. The non-outlier distance ratio R_N is almost always less than 1.1, and the outlier distance ratio R_O is almost always greater than 0.9. In some settings the outliers become indistinguishable from the other points, and in these cases the Jaccard coefficient may become as low as 0. However in those cases the other two measures R_N and R_O are very good, implying that the algorithm did not find the ground-truth outliers, but found equally good, or even better solutions. Such a case that our algorithm finds different but better solutions is when the standard deviation increases. A case that all three measures are low, is shown in Figure 3, for 1 000 outliers. Note that this is a very difficult settings as the fraction of outliers is as high as 50% of the total number of points.

5.2 Intrusion detection. Next we discuss the performance of our algorithm on real datasets. The first dataset is from the 1999 KDD·CUP and contains instances describing connections of sequences of TCP packets. Each instance is annotated with respect to being *normal* or an *intrusion*, and in the latter case, with the intrusion type. We experiment with a 10% sample, provided by the organizers, which contains 494 021 instances. The dataset has a mix of numerical and categorical attributes, and for simplicity we consider only the numerical attributes. In total there are 38 numerical attributes, and we normalize each one of them so that they have 0 mean and standard deviation equal to 1.

In total there are 23 classes, and 3 of them account for 98.3% of the whole dataset. In particular, the class `normal` has frequency 19.6%, the class `neptune` 21.6%, and the class `smurf` 56.8%. We consider these three classes as non-outliers, and we target to discover all

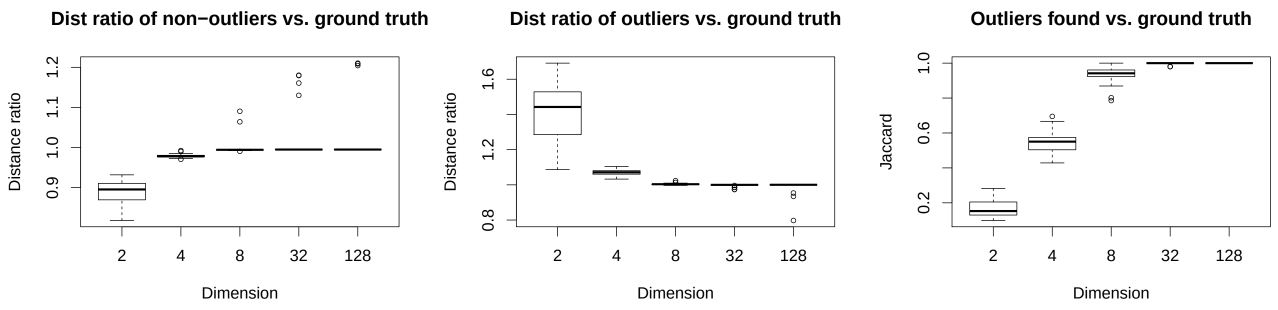

Figure 2: Behavior of the algorithm on synthetic datasets with respect to increasing dimension. The fixed parameters are $k = 10$, $m = 100$, $\ell = 100$, and $\sigma = 0.1$.

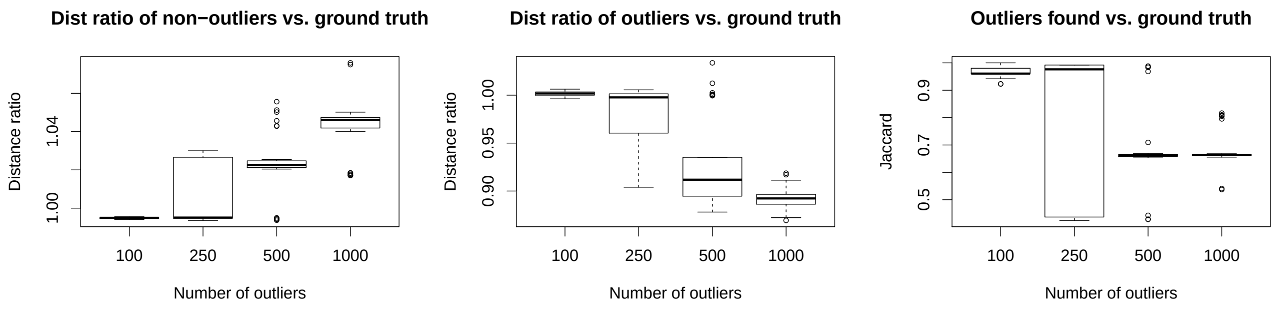

Figure 3: Behavior of the algorithm on synthetic datasets with respect to the number of outliers. The fixed parameters are $k = 10$, $m = 100$, $d = 32$, and $\sigma = 0.2$.

other 20 classes as outliers. Thus we set the number of outliers equal to $494\,021 \times 1.7\% \approx 8\,400$.

The results of our algorithm on this dataset is shown on Table 1. We measure the *precision* with respect to the outliers found by the algorithm compared to the ground-truth outliers, as well as the *purity* of the clusters found by the algorithm. Here purity is defined as the fraction of the majority class of each cluster with respect to the size of the cluster. We see that the algorithm is capable of discovering the outliers with fairly good precision. We also see that the resulting clustering matches very well the class labels of the dataset, as shown by the purity scores. Finally, from Table 1 we see that the performance of the algorithm remains stable for the different values of k that we try.

For reference, we compare our algorithm with a standard k-NN algorithm for distance-based outlier detection. This algorithm works by computing the k-th nearest neighbor of each point in the dataset and reporting as outliers the ℓ points whose distance to their k-th nearest-neighbor is the largest. This is an inherently quadratic algorithm. Even though we implement the algorithm using the smart-pruning heuristic proposed by Bay and Schwabacher [2], it was not possible to run it on the whole dataset. Thus, we run the k-NN algorithm on a sample of 50 thousand instances. We again apply the rule of searching for 1.7% outliers, giving $\ell = 850$.

The results for the k-NN algorithm are reported in Table 1, as well. Note that in this case, the parameter k is associated to the k-th nearest neighbor and not to the number of clusters. Since the k-NN algorithm does not produce a clustering, we do not report any purity scores. With respect to the precision scores, we observe that the k-means-- algorithm outperforms the k-NN algorithm. We also see that the k-NN algorithm is more sensitive to the value of k. We also note that when run on the same 50 K sample, the performance of the k-means-- algorithm actually improves: the precision score becomes 0.61.

5.3 Shuttle dataset. We use the SHUTTLE dataset, which is publicly available in the UCI Machine Learning Repository [10], and we perform a similar analysis as

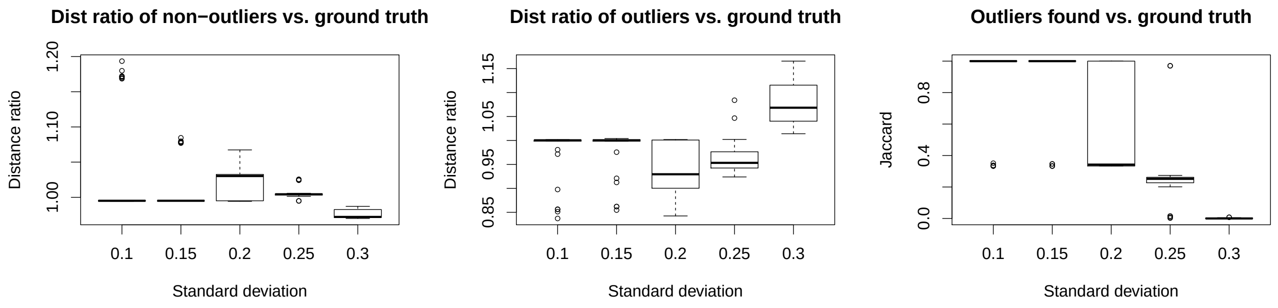

Figure 4: Behavior of the algorithm on synthetic datasets with respect to the standard deviation used to generate the clusters. The fixed parameters are $k = 10$, $m = 100$, $\ell = 200$, and $d = 64$.

Table 1: Results on the KDD·CUP dataset

k-means-- algorithm			
k	ℓ	Precision	Purity
5	8400 (1.7%)	0.564	0.987
7	8400 (1.7%)	0.568	0.990
13	8400 (1.7%)	0.593	0.981
k-NN algorithm (50 K sample)			
k	ℓ	Precision	
10	850 (1.7%)	0.241	
30	850 (1.7%)	0.331	
50	850 (1.7%)	0.301	

Table 2: Results on the SHUTTLE dataset

k-means-- algorithm			
k	ℓ	Precision	Purity
10	175 (0.4%)	0.155	0.945
15	175 (0.4%)	0.160	0.957
20	175 (0.4%)	0.172	0.974
k-NN algorithm			
k	ℓ	Precision	
10	175 (0.4%)	0.114	
20	175 (0.4%)	0.132	
30	175 (0.4%)	0.155	

Table 3: Stability results on the SHUTTLE dataset

k-means-- algorithm			
k	ℓ	Precision	Purity
10	87 (0.2%)	0.207	0.963
15	87 (0.2%)	0.207	0.967
20	87 (0.2%)	0.207	0.974
10	175 (0.4%)	0.155	0.945
15	175 (0.4%)	0.160	0.957
20	175 (0.4%)	0.172	0.974
10	348 (0.8%)	0.112	0.945
15	348 (0.8%)	0.123	0.969
20	348 (0.8%)	0.137	0.974

with the KDD·CUP dataset. The dataset contains 9 numerical attributes, and one categorical that can be interpreted as a class label. We use the training part of the dataset, which consists of 43 500 instances. There are 7 distinct class labels. The three largest classes account for the 99.6% of the dataset, with frequencies 78.4%, 15.5%, and 5.6%. We consider these three classes as non-outliers, and the set to identify as outliers the rest four classes that account for 0.4% of the dataset. We set appropriately $\ell = 43\,500 \times 0.4\% = 175$.

The results of the k-means-- algorithm on the SHUTTLE dataset are shown in Table 2. The clustering purity score are again high, indicating that the three majority classes are relatively easy to separate. However, compared with the results obtained on the KDD·CUP dataset, the precision scores are fairly low. The four classes that we identified as outliers, seem to be inherently difficult to distinguish from the three majority classes. This conjecture is verified by the performance of the k-NN algorithm on the same dataset, which is also shown in Table 2. The precision scores are comparable, in fact the performance of the k-means-- algorithm is somewhat better, while the performance of the k-NN algorithm is again sensitive to the choice of k.

5.4 Stability results. We experiment with the stability of our algorithm: we validate its performance as a function of the parameter ℓ. We report results on the SHUTTLE dataset, for which the ground truth value of ℓ is 0.4% of the total number of data points. We evaluate the precision and purity of our algorithm for two alternative values of ℓ, 0.2% and 0.8%, that is, half and double the ground-truth number of outliers. The results are given in Table 3. We observe that as ℓ increases the precision of the output decreases. This behavior is well-explained: the more outliers are asked, the more difficult it is to distinguish them from the rest of the data points. On the other hand, the purity figure does not change much, which means that the algorithm still returns very good clusters. Overall we see that even when overestimating the number of outliers by a factor of two, the k-means-- performs almost as good as the k-NN algorithm with the correct number of outliers.

5.5 Hurricane trajectories. We also experiment with clustering and finding outliers in trajectories. This is a different type of dataset than those in the previous sections, since the data points are not directly represented as vectors in a Euclidean space. A two-dimensional trajectory is a sequence of $\langle x, y, t\rangle$ coordinates, where x and y are spatial coordinates and t is the temporal dimension. We obtain trajectories of hurricanes from the Unisys weather site.[2] We compile a dataset of 479 hurricane trajectories from 1970 to 2010.

We then apply *spectral clustering* on the resulting set of trajectories [15]: For each pair of trajectories P and Q we compute their Fréchet distance [9]. Intuitively, the Fréchet distance $d_F(P,Q)$ between two curves P and Q is the minimum length of a leash required to connect a dog and its owner, constrained on two separate paths. Both the owner and the dog may vary their speed but backtracking is prohibited. Before computing the Fréchet distance of two curves we normalize them so that their center is located at the origin, so in fact we are measuring structural similarity rather than spatial distance. For applying the spectral-clustering algorithm we need a similarity measure between pairs of trajectories, and we use the similarity measure $s(P,Q) = e^{-d_F(P,Q)/\sigma}$, as proposed by Ng et al. [15]. We then project each trajectory to a point in $\mathbb{R}^r$, formed by considering its coordinates in the r eigenvectors that correspond to eigenvalues $\lambda_2, ..., \lambda_{r+1}$, where $0 = \lambda_1 \leq \ldots \leq \lambda_n$ are the eigenvalues of the Laplacian of the data similarity matrix.

For our experiment we use $r = 2$. Thus we project the data on the second and third "smallest" eigenvectors. The k-means step of spectral clustering is replaced by using the k-means-- algorithm, so in addition to clusters we also obtain outliers. We use our algorithm with values $k = 3$ and $\ell = 10$. Our algorithm correctly identifies the outliers that are evident in the spectral projection. The actual hurricane trajectories, are shown in Figure 5 using three different colors for the three clusters and black thick lines for the outliers. We see that the clustering nicely captures the three main directions of the trajectories.

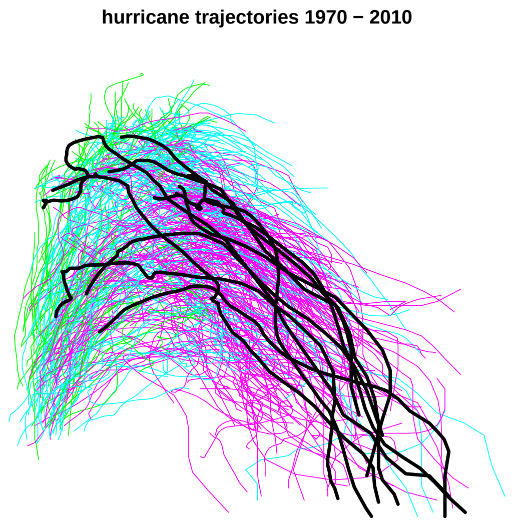

Figure 5: Clustering of the HURRICANE dataset with $k = 3$ and $\ell = 10$. Black trajectories are the outliers.

We have carried out a deeper analysis of the ten outlier hurricanes as shown in Table 4. Most (but not all) of the outlier hurricanes failed to make a US land fall. In fact their real trajectories show that, like all hurricanes, they originated near the equator and then moved towards the United States and then took a sharp turn towards Europe. Most of the hurricanes did not cause much damage in the US. On the other hand, Mitch caused extensive damage in Central America.

6 Conclusion and future work

Clustering and outlier detection are often treated as separate problems. However, both these problems are tightly coupled. For example, outliers can have a disproportionate impact on the shape of clusters which in turn can mask obvious outliers.

We have presented a unified approach by formulating the (k, ℓ)-MEANS optimization problem whose solution consists of k clusters and ℓ outliers. Being a generalization of the k-MEANS the problem is **NP**-Hard. We then present an iterative algorithm and prove that it converges to a local optima. We further show that the algorithm is not only suitable for Euclidean distance but also all similarity measures which can be expressed in the form of Bregman divergence. We first test the accuracy of the k-means-- on synthetic datasets which confirms that in a controlled set up, the algorithm his highly accurate and efficient. We have also conducted experiments on several real datasets including a novel

[2]http://weather.unisys.com/hurricane/atlantic/

Table 4: Information about the top 10 outlier hurricanes obtained from www.wunderground.com.

Name	Year	Dates	Max Winds (mph)	Min Pressure (mb)	Deaths	Damage	US Category
Charley	1986	08/13-08/30	80	980	5	15	Category 1
Chantal	1995	07/12/07/22	70	991	0	0	No US Landfall
Iris	1995	08/22-08/28	110	957	3	0	No US Landfall
Lilli	1996	10/14-10/29	115	960	8	0	No US Landfall
Earl	1998	08/31-09/08	100	964	3	700	Category 1
Mitch	1998	10/22-11/09	180	905	9086	40	Tropical Storm
Leslie	2000	10/04-10/10	70	973	0	0	No US Landfall
Maria	2005	09/01-09/13	115	962	0	0	No US Landfall
Ophelia	2005	09/06-09/23	85	985	0	0	No US Landfall
Alberto	2006	06/10-06/19	70	969	0	0	Tropical Storm

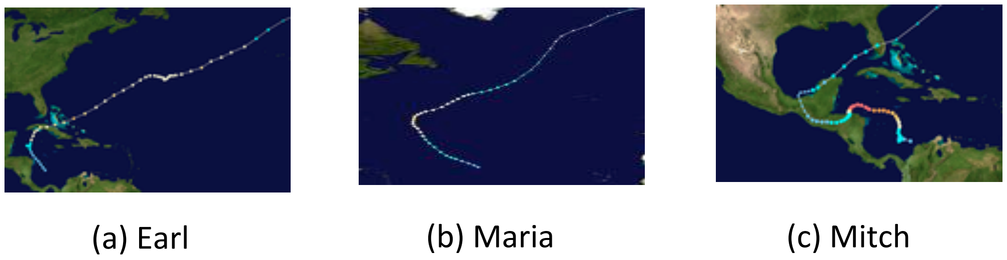

Figure 6: The tracks of three of the ten outliers. Earl and Mitch originated near (the top of) South America and then moved towards Florida. Maria on the other hand originated in the middle of the Atlantic Ocean and swerved towards Europe.

archive of hurricanes originating in the Atlantic Ocean. For the hurricane dataset, we actually embedded the k-means-- in a spectral clustering framework.

For future work we will integrate the k-means-- into a wider class of applications with a suitably chosen Bregman divergence which captures the semantics of the domain. We also plan to explore methodologies for automatically determining the number of outliers.

References

[1] A. Banerjee, S. Merugu, I. S. Dhillon, and J. Ghosh. Clustering with bregman divergences. *JMLR*, 2005.

[2] S. Bay and M. Schwabacher. Mining distance-based outliers in near linear time with randomization and a simple pruning rule. In *KDD*, 2003.

[3] T. Bernholt and P. Fischer. The complexity of computing the MCD-estimator. *TCS*, 2004.

[4] M. Breunig, H.-P. Kriegel, R. Ng, and J. Sander. LOF: Identifying density-based local outliers. In *SIGMOD*, 2000.

[5] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3), 2009.

[6] M. Charikar, S. Khuller, D. Mount, and G. Narasimhan. Algorithms for facility location problems with outliers. In *SODA*, 2001.

[7] S. Chawla and P. Sun. SLOM: A new measure for local spatial outliers. *KAIS*, 9(4):412–429, 2006.

[8] K. Chen. A constant factor approximation algorithm for k-median clustering with outliers. In *SODA*, 2008.

[9] T. Eiter and H. Mannila. Computing discrete fréchet distance. Technical report, TUW, 1994.

[10] A. Frank and A. Asuncion. UCI machine learning repository, 2010.

[11] D. Hawkins. *Identification of Outliers*. Chapman and Hall, London, 1980.

[12] E. Knorr and R. Ng. Algorithms for mining distance-based outliers in large datasets. In *VLDB*, 1998.

[13] Y. Kou, C. Lu, S. Sirwongwattana, and Y. Huang. Survey of fraud detection techniques. In *ICNSC*, 2004.

[14] A. Lazarevic, L. Ertöz, V. Kumar, A. Ozgur, and J. Srivastava. A comparative study of anomaly detection schemes in network intrusion detection. In *SDM*, 2003.

[15] A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *NIPS*, 2001.

[16] S. Papadimitriou, H. Kitagawa, P. Gibbons, and C. Faloutsos. LOCI: Fast outlier detection using the local correlation integral. In *ICDE*, 2003.

[17] S. Ramaswamy, R. Rastogi, and K. Shim. Efficient algorithms for mining outliers from large data sets. In *SIGMOD*, 2000.

[18] P. J. Rousseeuw and K. V. Driessen. A fast algorithm for the minimum covariance deteriminant estimator. *Technometrics*, 41:212–223, 1999.