Advanced Algorithms. Project Final Report

Remaldeep Singh, Mattia Medina-Grespan

December 5, 2018

1 Introduction

Clustering is a fundamental data mining task. It consists of partitioning the input data set into k clusters according to a defined notion of similarity. In the Euclidean space, the quality of a clustering solution is usually determined by the variance of the points within the same cluster. One way to obtain this clusters is to set k centers and then assign each point to its nearest center. Therefore, in this case, the "best" clustering would be the one that minimizes the sum of squared distances between every point and its nearest center. The latter is known as the $k-means$ objective and it is one of the most popular settings for the clustering problem.

However, even if the $k-means$ problem has been studied widely across the data mining literature, the algorithms developed for it usually have a poor performance on real data. This is due to the fact that the $k-means$ cost, defined for a set of "centers" C on the dataset U as follows

$$cost_{KM}(C;U) = \sum_{v \in U} d(v,C)^2,$$

is an objective function that assumes no noise. This is, it assumes the dataset U to be "clusterable", i.e. all of the datapoints can be naturally partitioned into k distinct clusters. Realistically, datasets will be "noisy", which means that they will have outliers points that can drastically affect the quality of the clustering solution.

On this papers, an approach to the problem of $k-means$ taking outliers into account is proposed. Using local search heuristics, the authors provide an algorithm that handles outliers achieving constant factor approximation solutions and that can be scaled to large datasets.

2 Formal description of the algorithmic problem

The authors propose an algorithm that handles datasets with noise. This algorithm discards potential outliers from the input, allowing the clustering to focus on the bulk of the dataset that is, potentially, noise-free.

The algorithm proposed, called **LS-outliers**, builds on a previous studied local search algorithm called **LS** for the $k-means$ problem with no outliers. The **LS** algorithm starts with a random set C of (k) centers given as input and a dataset U. Then it checks for each center $v \in C$ and each non-center $u \in U$ if swapping them would produce a lower $k-means$ cost. If that is the case, the algorithm updates the set of centers replacing C by $C \cup \{u\} \setminus \{v\}$. The algorithm stops when the cost value does not decrease significantly.

In order to achieve the generalization of **LS** with outliers, the algorithm **LS-outliers** relies on three main definitions provided:

1. The $k-means$ cost for a set of centers C with respect to a given subset Z of the dataset U. Defined as $cost(C,Z) = cost_{KM}(C; U \setminus Z)$.

2. For $S, S' \subseteq U$, the *farthest* α points in S from S' are the ones that maximize $\sum_{i=1}^{\alpha} d(v_i, S')$ (being α a non-negative integer).

3. $outliers(S,X)$ denotes the z farthest points from S in $U \setminus X$. For simplicity, $outliers(S) = outliers(S, \emptyset)$.

LS-outliers takes as input the dataset U of n points, the number of centers k, the number of the "target" outliers z, and a constant $0 < \epsilon < \frac{1}{6}$ or $\epsilon = 10^{-4}$. The algorithm outputs a final set C of centers and a set Z of outliers. It starts with a random set C of k data points as centers and the z farthest points from C as the set Z of outliers. In the same fashion as **LS**, the algorithm use local search (locally converge) to find good sets C and Z taking the following steps:

(i) It runs **LS** with centers C and the given dataset without inluding the initial z outliers in Z.

(ii) It checks if adding z more outliers to Z can improve the *cost* with a factor of $(1 - \frac{\epsilon}{k})$. This is, adding z more outliers to the set of outliers Z. These extra outliers are the z farthest points, in the dataset the were not in Z, from C.

(iii) Checks, for every pair i.e center-non center points, if the swapping and discarding z additional outliers improves the solution.

(iv) The algorithm selects the set of z additional outliers to discard, out of the steps (ii) and (iii). That is, if a better improvement of the *cost* was obtained without swapping (step ii) or with swapping (step ii) and accordingly either updates the set of outliers from step (ii) or updates set of centers and outliers from step (iii).

(v) This process continues until convergence.

The authors prove that the algorithm computes a constant-approximation factor to the $k-means$ objective. However, the algorithm is allowed to discard more outliers than specified in the input in order to gain analytic tractability. They argue that the exact number of outliers is rarely known, then labelling some extra points as outliers should not affect the quality of the clustering solution.

3 Algorithm provided in the paper

LS-Outlier (U,k,z)

C $\leftarrow$ an arbitrary set of k points from U
Z $\leftarrow$ $outliers(C)$
$\alpha \leftarrow \infty$
while $\alpha\left(1 - \frac{\epsilon}{k}\right) > cost(C, Z)$ **do**
  $\alpha \leftarrow cost(C, Z)$
  (local search with no outliers)
  $C \leftarrow$**LS**$(U \setminus Z, C, k)$
  $\overline{C} \leftarrow C$ **(improved current set of centers)**
  $\overline{Z} \leftarrow Z$ **(improved current set of outliers)**
  (cost of discarding z additional outliers)
  if $cost(C, Z)\left(1 - \frac{\epsilon}{k}\right) > cost(C, Z) \cup outliers(C, Z)$ **then**
    $\overline{Z} \leftarrow Z \cup outliers(C, Z)$
  (for each center and non-center, perform a swap and discard additional outliers)
  for each $u \in U$ and $v \in C$ **do**
    (if this is the most improved swap found so far)
    if $cost(C \cup \{u\} \setminus \{v\}, Z \cup outliers(C \cup \{u\} \setminus \{v\}) < cost(\overline{C},\overline{Z})$ **then**
      (update the current solution)
      $\overline{C} \leftarrow C \cup \{u\} \setminus \{v\}$
      $\overline{Z} \leftarrow Z \cup outliers(C \cup \{u\} \setminus \{v\})$
  (update the solution allowing additional outliers if the solution value improved significantly)
  if $cost(C, Z)\left(1 - \frac{\epsilon}{k}\right) > cost(\overline{C}, \overline{Z})$ **then**
    $C \leftarrow \overline{C}$
    $Z \leftarrow \overline{Z}$
return C *as the* k centers and Z as the outliers

We have explained how the algorithm works in section 2.

4 Baseline algorithm

Since we are proposing changes on the **LS-outlier** (an consequently **LS**) algorithm, we will use it as the baseline algorithm.

5 Dataset

We ran the Principal Component Analysis and t-Distributed Stochastic Neighbor Embedding on four different datasets from the UCI repository (including sampled SUSY). The data set that showed has the best relation in terms of clusterability and size (number of examples) was the "Shuttle" dataset. This dataset has 58000 examples, and 9 attributes. Therefore, we run LS-outliers and our approach for LS-outlier on the latter dataset to compare results.

We have also created a synthetic dataset with Gaussian distribution, random gaussian noise and structured noise. Specifically, this dataset consists in 2,200 points in 2 dimensions; 2 clusters of 1000 points each, 1 outlier cluster of 100 points and 100 sparse outliers. Let us call this dataset SYN. We are testing both, LS-outliers and our approach on this SYN.

6 Work and results

6.1 Part I

1. As mentioned above **LS-outliers** algorithm uses **LS** algorithm as a subroutine.

   Therefore, we begun the work for this project implementing **LS** and **LS-outliers** finding some important challenges while coding it. The implementation, (based on the pseudocode given in Apendex of this proposal) is considerably slow on the Shuttle dataset.

2. We run our implementation of **LS-outliers** on both, Shuttle and **SYN** datasets. Finding out that the obtained output set of outliers, for both dataset, consisted of all the points in the data except for the centers. This undesired behavior had to do the fact that the datasets we are studying are "small" enough to make the cost change threshold of $(1 - \frac{\epsilon}{k})$ with respect to the minimum cost used in the pseudocode of the paper (which is approximately 99.5% of the α value), to be too tight for the quality of the clusters. Indeed, as explained in the paper, this threshold works for exponential sizes of the datasets.

   Hence, we decided to use a smoother threshold of 85% of alpha obtaining a much better results, namely, better quality of the clusters and better size of the clusters.

6.2 Part II

1. The possibility of choosing outliers datapoints as initial centers during the starting random process is one of the potential problems with **LS-outliers**. This situation will result in the selection of "clusterable" points, as outliers, when choosing the z *farthest points* from the set of centers.

   The idea we proposed to try to handle an adversarial scenario like the one described above, is to change the *outlier* selection criterion/definition mention above (3.) as follows: $outliers(S, X)$, denoted as $outliers^*(S, X)$ to be the set of z points from S in $U \setminus X$ wherein the z points are chosen at random based on the probability proportional to the distance $d(x, S)$, for $x \in U \setminus X$, and $d(x, S)$ the minimum distance of x from the set of points S.

   In this way, we are adding randomness on the setting of the outliers which is a commonly used strategy that could lead to better results.

   While implementing this changes (including the change of the new threshold) over **LS-outliers**, we run into problems at procedure (iii) of the algorithm.

   This problems came from the fact that on this step of the pseudocode provided in the paper, the algorithm swaps center and non-center points and evaluates the cost discarding the z additional outliers from the previous procedure (i). But notice that the non-center points of the for loop consist of **every** point in the dataset U including the outliers. This means that it allows a previously chosen outliers to be swapped as centers. In contrast, given the the *outliers* are defined as the farthest points from the set of centers, the cost of the swapping was always going to be high enough to not satisfy the (small enough) threshold and not be used

to update the set of centers. With our probabilistic modification to select outliers, namely, $outliers^*$, it is possible that an outlier in Z that is "very close" to a cluster, or even within a cluster is chosen to replace a center point, resulting in an improvement of the cost such that the threshold is satisfies thus the set of centers updated. In this case, we get the scenario where a data point is a center as well as an outlier and the program crashes.

Therefore, in order to solve this situation, for every non center point in U that is also in the current set of outliers Z, we delete it from the set Z. Thus avoiding the situation described above.

We will call $\mathbf{LS^* - outliers}$ the algorithm obtained from applying the modifications mentioned above to **LS-outliers**.

6.3 Results

We run **LS-outliers** over **SYN** dataset under the scenario of an outliers being selected as a center in the initial random process. Obtaining the following results:

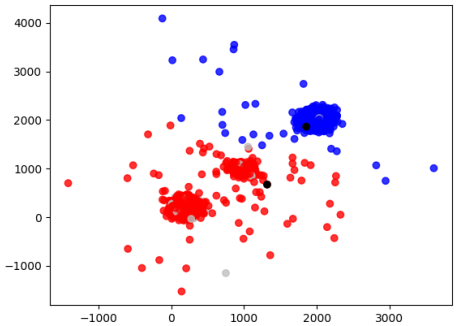
Round	Cost	Cluster 1 size	Cluster 2 size	Adjusted rand score
1	1550753113.0	1112	1018	-
2	122499044.0	1041	1019	-
3	51509689.0	975	1015	-
4	34733468.0	949	971	**0.91**

Table 1: **LS-outliers** on **SYN** dataset with outlier as a center.

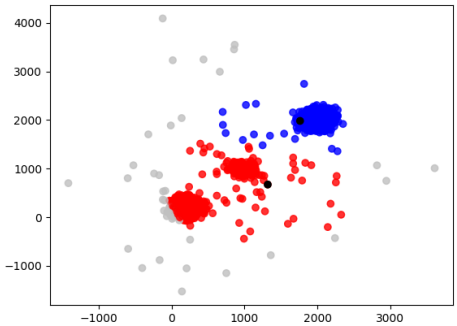
Round	Cost	Cluster 1 size	Cluster 2 size	Adjusted rand score
1	1634998832.0	1107	1023	-
2	215711030.0	1035	1025	-
3	144606139.0	986	1004	-
4	92328044.0	945	975	-
5	62694222.0	908	942	-
6	52093199.0	885	896	**0.75**

Table 2: $\mathbf{LS^* - outliers}$ on **SYN** dataset with outlier as a center.

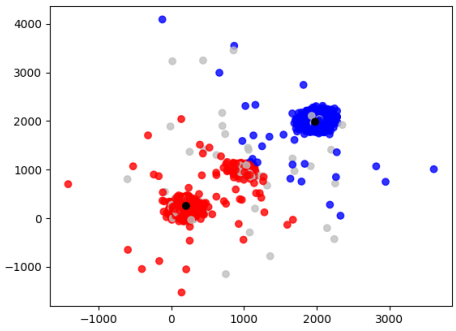
Figure 1: The black dots indicate the centers for clusters.



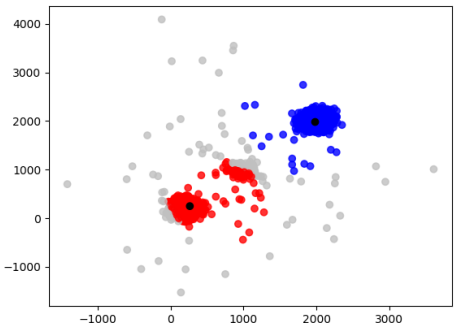
(a) $\mathbf{LS^* - outliers}$ Round 1.



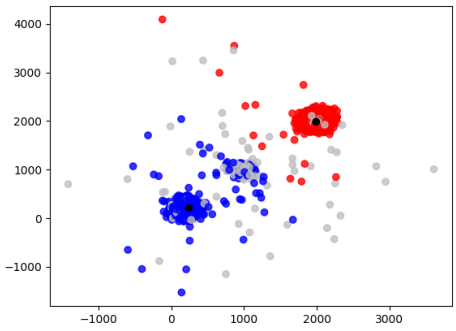
(b) **LS-outliers** Round 1.



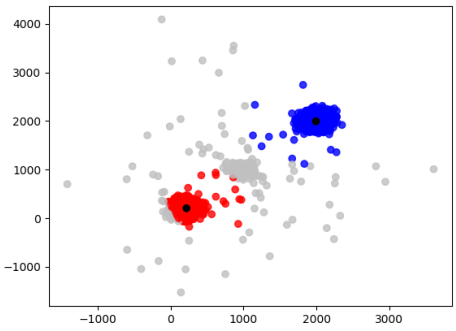
(a) **LS*** − **outliers** Round 2.



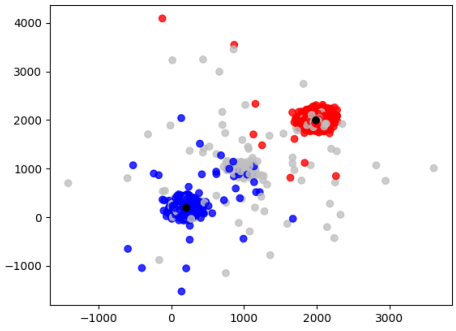
(b) **LS-outliers** Round 2.



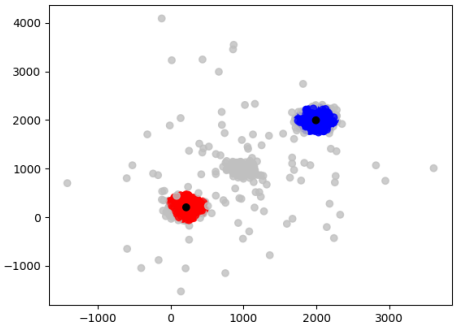
(a) **LS*** − **outliers** Round 3.



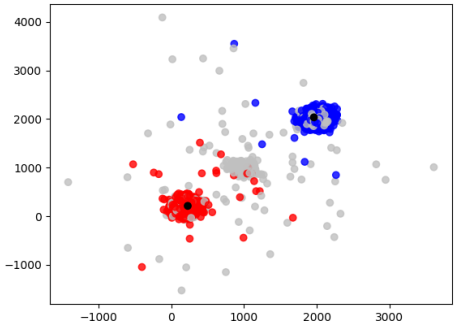
(b) **LS-outliers** Round 3.



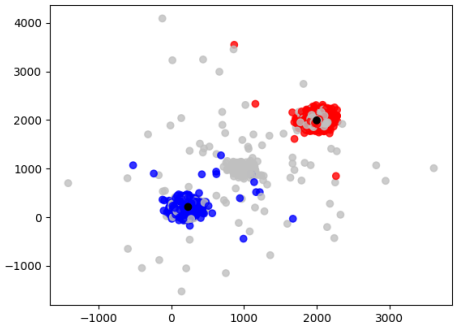
(a) **LS*** − **outliers** Round 4.



(b) **LS-outliers** Round 4.



(a) **LS*** − **outliers** Round 5.



(a) **LS*** $-$ **outliers** Round 6.

Contrary to our expectations, Table 1, Table 2 and the plots above show that our approach **LS*** $-$ **outliers**, over the **SYN** dataset, does not give better results than the original algorithm **LS-outliers** in the adversarial scenario we are experimenting with. Indeed, we can see that it took **LS-outliers** only 4 rounds to stop getting a better final cost than $LS^* - outliers$ which performed 6 rounds to finish.

Note: We are using adjusted rand score to identify the quality of the clusters.The Rand Index computes a similarity measure between two clustering by considering all pairs of samples and counting pairs that are assigned in the same or different clusters in the predicted and true clustering.

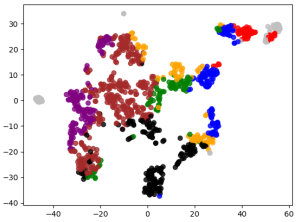
The next step of our experiment, we run **LS*** $-$ **outliers** and **LS-outliers** on the Shuttle UCI dataset in order to have a better sense about how our approach worked compared with the original algorithm.

Due to the time complexity of our implementation for **LS*** $-$ **outliers** and **LS-outliers** we decided to take the following three steps:

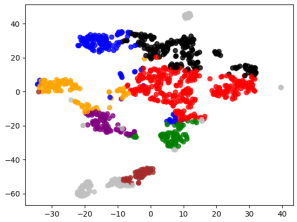
1. We randomly sampled 1000 examples as from the Shuttle dataset to work with.
2. Therefore, we consider a bigger threshold of 15% as the stop condition for both algorithms.
3. We used t-SNE as the dimensionality reduction technique to visualize our results. PCA was giving not so visible clusters.

The following figures show the performance of both algorithm over the sampled Shuttle dataset. **Note:** Since we are using **t-SNE** for visualization the figures will differ for each round.

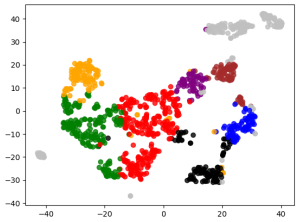
Figure 7: t-SNE analysis over **LS-outliers** on Shuttle dataset



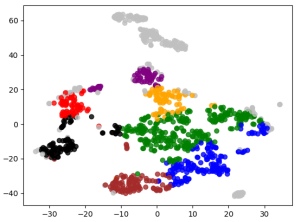
(a) Round 1



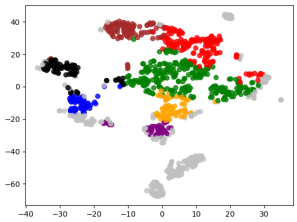
(b) Round 2



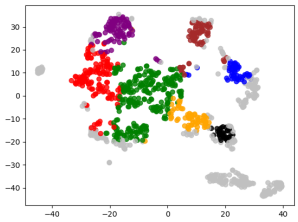
(c) Round 3



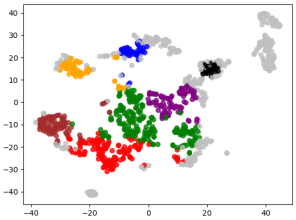
(a) Round 4



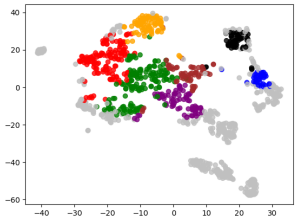
(b) Round 5



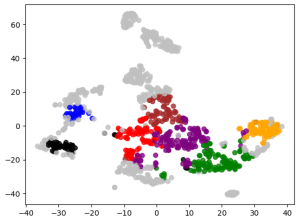
(c) Round 6



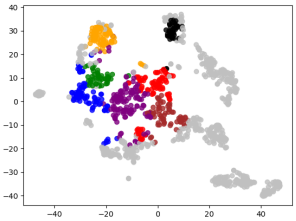
(a) Round 7



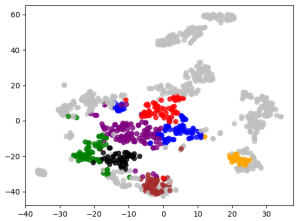
(b) Round 8



(c) Round 9

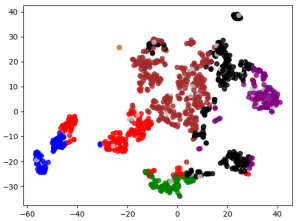


(a) Round 10

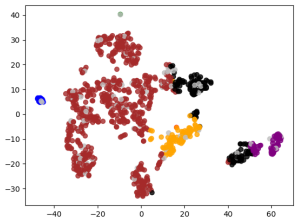


(b) Round 11

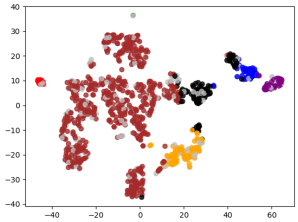
Figure 11: t-SNE analysis over **LS*** – **outliers** on Shuttle dataset



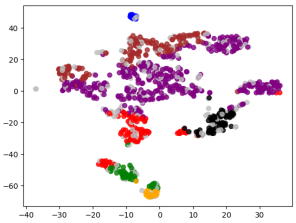
(a) Round 1



(b) Round 2



(c) Round 3



(a) Round 4

Round	Cost **LS-outliers**	Cost $\mathbf{LS^* - outliers}$ size
1	223,776	483,746
2	175,728	396,870
3	137,337	318,077
4	111,053	286,603
5	89,929	-
6	73,018	-
7	59353	-
8	46129	-
9	36251	-
10	27995	-
11	21805	-

Table 3: Cost for **LS-outliers** and $\mathbf{LS^* - outliers}$ on Shuttle dataset with random initial centers.

From above table (3) it looks like **LS-outliers** algorithm is working better than $\mathbf{LS^* - outliers}$ but in fact that is not the case. It is shown clearly in the figures that in the process of reducing the cost the **LS-outliers** algorithm is actually removing the data points that belong to a far away cluster, which might or might not be noise. In this case our algorithm, $\mathbf{LS^* - outliers}$, is working better than the **LS-outliers**.

7 Future work

- An observation we have about the algorithm is that it always adds z extra outliers. Rather than adding z outliers at every iteration we plan to decrease the set of points added since there is a higher chance with every iteration to include non-outliers in outliers. We can recall some of the outliers from z based on some distance heuristic or we can decrease z based on $\frac{z}{2^i}$, i being the iteration number of the algorithm.
- One more way to approach this problem, (also discussed during the meeting with Dr. Bhaskara) was by clustering using Lloyd's algorithm without outliers considerations. In order to tackle the data outliers, we can use a Probability Density function with binning on clusters to identify the outliers. We start by aplying this process to one cluster C_1 to get an outlier o. Then for every other cluster, we repeat the process considering o as part of it. If o turns out to be an outlier all of the cluster, we will just added it to an set of outliers O. Otherwise, we will add o to the first cluster where it is not an outlier. Then we repeat the entire process until convergence.

8 APPENDIX

1. **LS (U,C,k)**

   $\alpha \leftarrow \infty$
   (while the solution improves by performing swaps)
   while $\alpha\left(1 - \frac{\epsilon}{k}\right) > cost_{KM}(C; U)$ **do**
   $\alpha \leftarrow cost_{KM}(C; U)$
   $\overline{C} \leftarrow C$ **(improved current set of centers)**
   for each $u \in U$ and $v \in C$ **do**
   (if this is the most improved swap found so far)
   if $cost_{KM}((C \cup \{u\} \setminus \{v\}); U) < cost_{KM}(\overline{C}; U)$ **then**
   $\overline{C} \leftarrow C \cup \{u\} \setminus \{v\}$**(update the current solution)**
   (update the solution to the best swap found)
   $C \leftarrow \overline{C}$
   return C as the k centers

2. Dataset Used: UCI Shuttle Dataset