Robust Partitional Clustering by Outlier and Density Insensitive Seeding

Mohammad Al Hasan, Vineet Chaoji, Saeed Salem, and Mohammed J. Zaki

Dept. of Computer Science, Rensselaer Polytechnic, Troy, NY, 12180

Abstract

The leading partitional clustering technique, k-means, is one of the most computationally efficient clustering methods. However, it produces a local optimal solution that strongly depends on its initial seeds. Bad initial seeds can also cause the splitting or merging of natural clusters even if the clusters are well separated. In this paper, we propose, ROBIN, a novel method for initial seed selection in k-means types of algorithms. It imposes constraints on the chosen seeds that lead to better clustering when k-means converges. The constraints make the seed selection method insensitive to outliers in the data and also assist it to handle variable density or multi-scale clusters. Furthermore, they (constraints) make the method deterministic, so only one run suffices to obtain good initial seeds, as opposed to traditional random seed selection approaches that need many runs to obtain good seeds that lead to satisfactory clustering. We did a comprehensive evaluation of ROBIN against state-of-the-art seeding methods on a wide range of synthetic and real datasets. We show that ROBIN consistently outperforms existing approaches in terms of the clustering quality.

Key words: k-means, seed selection, robust initialization, partitional clustering

1 Introduction and Background

Clustering is one of the most fundamental tasks in exploratory data analysis that groups similar points in an unsupervised fashion. The clustering problem has been studied in many disciplines such as statistics, pattern recognition, signal processing (e.g., vector quantization), biology, and so on. As

⋆ This work was supported by NSF Grants EMT-0829835, and CNS-0103708, and NIH Grant 1R01EB0080161-01A1.

Email address: {alhasan, chaojv, salems, zaki}@cs.rpi.edu (Mohammad Al Hasan, Vineet Chaoji, Saeed Salem, and Mohammed J. Zaki).

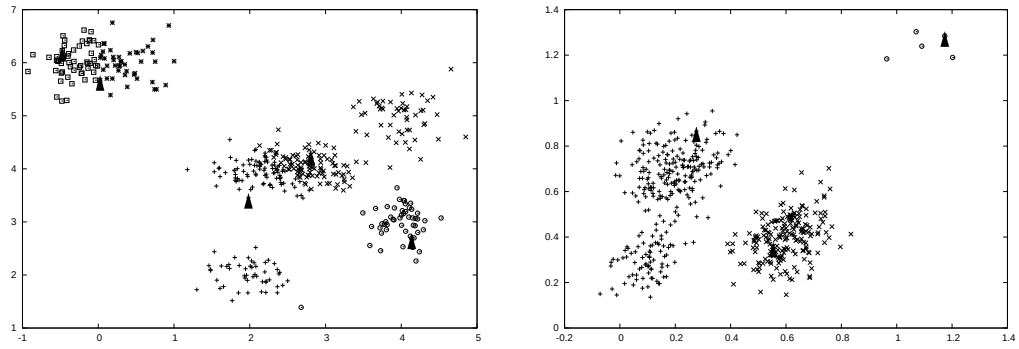
a consequence numerous clustering algorithms have been proposed in these different communities, spanning different clustering paradigms such as partitional (Forgy, 1965; Lloyd, 1982; MacQueen, 1967), hierarchical (Jardine et. al., 1967), spectral (Shi & Malik, 2000), density-based (Ester et. al., 1996), mixture-modeling (Jain & Dubes, 1988), and so on. Subspace clustering methods (Parsons, 2004) have also been proposed, but in this paper our focus is on full-dimensional clustering methods.

k-means is one of the earliest clustering algorithms which has been proposed independently in (Forgy, 1965; MacQueen, 1967) and in (Lloyd, 1982). The typical formulation of k-means is as follows: Given k, the number of clusters to find, and a set of n data points in $\mathbb{R}^d$, the goal is to choose k points as centers so as to minimize the sum of the distance between each point and its closest center. This formulation, which we adopt in this paper, is also named as *distortion minimization* clustering. Considered in an optimization framework, the objective function is non-convex and hence, it is difficult to obtain a global optimal solution. In fact, if we require that the cluster centers be actual data points, the problem is NP-Hard even for $k = 2$ (Garey & Johnson, 1979).

Even though it is difficult to obtain a global optimal solution for k-means, finding a local optimal solution is very cheap. Lloyd's algorithm (Lloyd, 1982) can be used to obtain such a solution. It chooses a set of random k data points as centers. All the remaining data points are assigned to their nearest centers. This forms the initial clusters. Then, for each cluster, its center is recomputed as the center of mass of all points assigned to it. These two steps, cluster assignment and center recomputation, are repeated until the clusters assignments converge. It is easy to show that every successive iteration improves the distortion, and the algorithm terminates with a local optimal solution.

Though not optimal in the global sense, k-means is still the most popular clustering method for a variety of applications (Berkhin, 2002). Recently k-means was even voted as one of the top ten algorithms in data mining (Wu et. al., 2008). The main reason for the popularity of k-means is its simplicity and efficiency. In fact, it has been shown that the convergence rate of k-means is comparable to Newton's method, which is quadratic (Bottou & Bengio, 1995). Moreover, it does not require the computation of $O(n^2)$ distances/similarities as in hierarchical or spectral clustering algorithms. Many distance computations can also be saved by using triangular inequality (Elkan, 2003).

The enormous popularity of k-means has motivated research to remedy its limitations, specially safeguarding against bad local optimal solutions. Note that the solution space of any clustering is just a k-partition of the data points, so a neighborhood consists of other k-partitions that are very similar to it. Hence, once a k-partition is obtained by k-means, random re-shuffling of a few points across different clusters generally does not produce a better



(a) Merging/splitting of natural clusters (b) Cluster composed of only noise points

Fig. 1. Random initialization produces poor clustering. The dark triangles indicate the initial cluster seeds. Points with the same symbol belong to the same cluster.

clustering, since the modified partition, most likely, would actually be in the neighborhood of the previous k-partition. To get around the local minima, a common practice is to run the k-means algorithm repeatedly, each time starting with a different set of random initial centers. With a high probability, different runs (with different seeds) would explore different regions of the solution space and thus increasing the likelihood of escaping local minima. The solution that achieves the best objective function value is finally reported as the clustering solution. However, for large datasets, repeated runs can be costly. Previous research (Arthur & Vassilvitskii, 2007; Astrahan, 1970; Ball & Hall, 1967; Bradley & Fayyad, 1998; Katsavounidis, 1994; Khan & Ahmad, 2004) has thus attempted to intelligently choose the initial centers (seeds) such that the clustering is substantially better than that obtained by just random seeding. The success of these approaches depend on finding seeds that yield an initial partition which is close to a reasonably good local optimal. Based on our extensive experimental evaluation, we found that while previous approaches fare better than random initialization, they are very sensitive to outliers, as well as variable density regions.

In this research, we propose a novel seed initialization algorithm, named ROBIN (**ROB**ust **IN**itialization) that outperforms previous approaches in terms of the clustering quality (distortion), and it is also simple and efficient. As the name suggests ROBIN is robust to outliers or variable density. It works by imposing hard constraints on the seeds as they are selected. Other than k, the number of centers to choose, it takes only one other parameter, mp, the number of neighbors to consider, which is used when computing a measure of the degree to which a potential center is an outlier. Furthermore, we also show that ROBIN is not very sensitive to the choice of mp. We made extensive experiments on synthetic and real world dataset. In synthetic cases, we found that ROBIN finds solutions which are nearly as good as the (known) global optimal solutions. For real world datasets, it typically obtains superior distortion scores compared to other methods.

2 k-means Initialization

Most of the seed selection ideas emerged from the analysis of poor clustering solutions obtained by k-means. In the following, we explain the reasons why k-means may fail to produce a good clustering solution and how different seed selection strategies attempt to safeguard against those. However, we also like to clarify that for k-means, the poor seed selection is just one reason that result in poor clustering. The objective function that k-means optimize can also be responsible for this fact. For example, if a small cluster is in close proximity to a large cluster, a k-means algorithm that minimizes the distortion, will always merge a set of nearby points from the large cluster with the small cluster. This is inevitable even for the case when the geometric centers of the clusters are chosen as the initial seeds. Similar problem also arises when the clusters are of non-globular shape. In those cases, a cluster may contain points that are very far from each other, whereas the points that lie on the intersecting line of the above points belong to a different cluster (think of a circular cluster surrounded by a u-shape cluster). So, general k-means algorithm will always fail for the above cases irrespective of the seed selection methods. So, throughout our discussion, we assume clusters with comparable sizes and nearly globular shapes. We also assume that the user has the correct guess about the value of k.

In k-means, a poor clustering solution is obtained if two seeds are in close proximity. In such cases k-means can end up partitioning a natural cluster, and as a consequence, since k is fixed, it merges a pair of distant clusters into one. Figure 1(a) shows an example when this happens. As we can see two random seeds were chosen from the top left cluster, which results in that cluster being partitioned into two, and the center and top right clusters are merged into one. A poor solution can also be obtained when a chosen seed is actually an outlier. In such cases k-means can form a cluster which is not a natural cluster, but a set of noise points. As a consequence, some real cluster will be merged or missed. For example, we can see in Figure 1(b) that the initial seed on the top right results in a cluster composed solely of outliers, whereas the two larger clusters on the left are merged into one. Certain other traits of the dataset are responsible for poor performance of k-means. For instance, if the sizes of the clusters are substantially different, the likelihood is small that a seed would be chosen from the smaller cluster; as a result, in the final clustering solution, the smaller cluster may be merged with an adjacent large cluster. Different densities of the data points in different clusters, also called multi-scale clusters, can also have adverse affects on the initialization and clustering quality.

Many of the problems discussed above can be avoided by applying constraints in the seed selection process. If the clusters are globular and their sizes are

comparable, the splitting or merging of clusters can be entirely eliminated by adopting the constraints that the seeds are well separated. Formation of a noise cluster can also be avoided by constraining that a seed point is not an outlier point. However, seed selection is just an initialization step for k-means, so it has to be inexpensive. We list below the desiderata for a good cluster seed selection scheme:

(1) It should be computationally inexpensive.
(2) It should ideally be parameter free. For the parameters it does require, it should not be very sensitive to the parameter values. Furthermore, it should be intuitive to set reasonable parameter values.
(3) It should ideally be deterministic and should be insensitive to the order in which the data points are considered.
(4) It should not be sensitive to outliers, i.e., a noise point should not be selected as a center.
(5) It should be robust in presence of multi-scale (varying densities) clusters.

We now briefly discuss previous initialization schemes and the extent to which they fulfill the above criteria. One of the first schemes of center initialization was proposed by Ball and Hall (Ball & Hall, 1967). They suggested the use of a user defined threshold, d, to ensure that the seed points are well apart from each others. They consider each point in the dataset in an arbitrary order. The first point is chosen as a seed, and for any subsequent point considered, it is selected as a seed if it is at least d distance apart from the already chosen seeds, until k seeds are found. With a right choice of the value of d, this approach can restrict the splitting of natural clusters, but guessing a right value of d is very difficult and the quality of seeds depends on the order in which the data points are considered. Note that, it only satisfies the criteria (1) above, since its complexity is just $O(kn)$. A similar approach was also suggested by Tou and Gonzales under the name Simple Cluster Seeking (SCS) (Tou & Gonzales, 1974).

Astrahan (Astrahan, 1970) suggested using two distance parameters, d_1 and d_2. The method first computes the *density* of each point in the dataset, which is given as the number of neighboring points within the distance d_1. It then sorts the data points according to decreasing value of density. The highest density point is chosen as the first seed. Subsequent seed point are chosen in order of decreasing *density* subject to the condition that each new seed point be at least at a distance of d_2 from all other previously chosen seed points. This step is continued until no more seed points can be chosen. Finally, if more than k seeds are generated from the above approach, hierarchical clustering is used to group the seed points into the final k seeds. The main problem with this approach is that it is very sensitive to the values of d_1 and d_2. Furthermore, users have very little knowledge regarding the good choices of these parameters, and the method is computationally very expensive. A range

search query needs to be made for every data point followed by a hierarchical clustering of a set of points. Small values of d_1 and d_2 may produces enormously large number of seeds, and hierarchical clustering of those seeds can be very expensive ($O(n^2 \log n)$ in the worst case). This method also performs poorly when there exist different clusters in the dataset with variable density and size. In summary, this approach satisfies only criteria (3) and (4) above.

Katsavounidis et. al. (Katsavounidis, 1994) suggested a parameterless approach, which we call the KKZ method based on the initials of all the authors. KKZ chooses the first center near the "edge" of the data, by choosing the vector with the highest norm as the first center. Then, it chooses the next center to be the point that is farthest from the nearest seed in the set chosen so far. This method is very inexpensive ($O(kn)$) and is easy to implement. It does not depend on the order of points and is deterministic by nature; as single run suffices to obtain the seeds. Thus KKZ satisfies criteria (1), (2), and (3) above. However, KKZ is sensitive to outliers, since the presence of noise at the edge of the dataset may cause a small set of outlier/noise points to make up a cluster (e.g. the dataset in Figure 1(b)). Note here that the KKZ method is similar to the 2-approximation solution of the k-center problem (Hochbaum & Shmoys, 1985).

Bradley and Fayyad (Bradley & Fayyad, 1998) proposed an initialization method that is suitable for large datasets. We call their approach Subsample, since they take a small subsample (less than 5%) of the dataset and use k-means clustering on the subsample and record the cluster centers. This process is repeated and cluster centers from all the different iterations are accumulated in a dataset. Finally, a last round of k-means is performed on this dataset and the cluster centers of this round are returned as the initial seeds for the entire dataset. This method generally performs better than k-means and converges to the local optimal faster. However, it still depends on the random choice of the subsamples and hence, can obtain a poor clustering in an unlucky session. Among the criteria above, Subsample satisfies (1) and (2), and also (4) with high probability.

More recently, Arthur and Vassilvitskii (Arthur & Vassilvitskii, 2007) proposed the k-means++ approach where the seed selection is similar to the KKZ method. However for seed, k-means++ does not take the farthest point from the already chosen seeds, but it selects a point with a probability proportional to its distance from the already chosen seeds. Thus, in KKZ the farthest point is chosen with probability 1, but in k-means++, probability of this selection is proportional to the minimum distance of this point from already chosen seeds. k-means++ satisfies criteria (1) and (2), and also (4) with high probability. Note that due to the probabilistic selection of points, different runs have to be performed to obtain a good clustering.

The problem of obtaining poor local optimal solution by k-means is also addressed in the domain of fuzzy clustering (Bezdek et. al., 2005). The most notable method in this domain named "Competitive Agglomeration"(CA) is proposed by Frigui and Krishnapuram (Frigui & Krishnapuram, 1997). CA starts with a large number of cluster seeds that provide a good initial sampling of the entire dataset. As the algorithm progresses, adjacent clusters compete for data points, and the weaker ones progressively disappear. Authors show that the remaining seeds (and the final clustering) has enhanced ability to overcome the poor local optimal solution of regular k-means that generally arises from the bad seed selection. In this article we, however, restrict our discussion on hard (non-fuzzy) seed selection methods.

Our proposed method, ROBIN, satisfies all of the criteria that we discuss at the beginning of this section. It is computationally inexpensive. It takes only one parameter mp, the number of neighboring points to consider, and is not very sensitive to the value of this parameter. It is deterministic, insensitive to outliers, and is robust against variable density.

3 The ROBIN Approach

The ROBIN approach to seed selection is essentially tied to the concept of avoiding outliers as seeds. For this ROBIN first computes the degree to which a point is an outlier, which in turn must consider the local density of the neighboring points. Outliers are those points whose density is very different compared to neighbor densities. In essence the local outlier measure automatically takes into account variable density regions and variable size clusters. The key aim here is to avoid the computation of outlier measure for each point in the dataset, which would yield a worst case $O(n^2)$ method, but rather the challenge is in keeping the complexity very close to linear in n.

Subject to the outlier measure, ROBIN ensures that the seeds are as far apart as possible. More formally, let $\mathcal{D}$ be a set of n points in $\mathbb{R}^d$. Given $2 \leq k < n$, we wish to find a *maximally separated subset* $I \subseteq \mathcal{D}$ of size k, for which the minimum distance among the $\binom{k}{2}$ pairs of points in I is as large as possible. The decision problem associated with this is to determine whether there exists $I \subseteq \mathcal{D}$, with $|I| = k$, so that all $\binom{k}{2}$ distances in I are at least 2. The decision version arises from the following transformation. Let us place a d-dimensional unit ball at each point. We can then construct an intersection graph G that has an edge between a pair of points, if the corresponding unit balls have a non-empty intersection. The set of seeds I is then an independent set in the graph G. Clearly, if G has an independent set of size k, we have $|I| = k$, where all the $\binom{k}{2}$ distances are at least 2 (since the *unit* radius balls do not intersect, they are at least a distance of 2 apart). For a given k the problem whether

an independent set of size k exists is known to be NP-Complete. Hence, the *maximally separated point-set* problem is also NP-Complete when k is part of the input (Clark et. al., 1990). ROBIN's seed selection is therefore just a greedy solution to the above problem.

3.1 Local Outlier Factor

To compute the degree to which a point is an outlier, we use the notion of *local outlier factor (LOF)* which was proposed in (Breunig et al., 2000). For a point $x \in \mathcal{D}$, define the local neighborhood of x, given the minimum points threshold mp as follows:

$$N(x, mp) = \{y \in \mathcal{D} \mid distance(x, y) \leq distance(x, x_{mp})\}$$

where x_{mp} is the mp-th nearest neighbor of x. Thus $N(x, mp)$ contains at least mp points. The *density* of x is then computed as follows:

$$density(x, mp) = \left(\frac{\mid N(x, mp) \mid}{\sum_{y \in N(x,mp)} distance(x, y)} \right)$$

Essentially, the lower the distance between x and neighboring points, the higher the density of x. The *average relative density (ard)* of x, is then computed as the ratio of the density of x and the average density of its nearest neighbors, given as follows:

$$ard(x, mp) = \frac{density(x, mp)}{\left(\frac{\sum_{y \in N(x,mp)} density(y,mp)}{|N(x,mp)|} \right)}$$

Finally the LOF score of x is just the inverse of the average relative density of x:

$$LOF(x, mp) = ard(x, mp)^{-1}$$

If a point is in a low density neighborhood compared to all its neighbors, then its *ard* score is low and hence its LOF value is high. Thus LOF value represents the extent to which a point is an outlier. A point that belongs to a cluster has an LOF value approximately equal to 1, since its density and the density of its neighbors is approximately the same.

LOF has two excellent properties: (1) It is very robust when the dataset has clusters with different sizes and densities. (2) Even though the LOF value may vary somewhat with mp, it is generally robust in making the decision whether a point is an outlier or not. That is, for a large range of values of mp, the outlier points will have LOF value well above 1, whereas points belonging to a cluster will assume an LOF value close to 1.

3.2 The ROBIN Algorithm

```
ROBIN(D, k, mp):
1.  Take any reference point, r (origin suffices)
2.  m = 0;
3.  while (| C |≤ k)
4.     if (m == 0)
5.        sort the points in D in decreasing order of
                              distances from r
6.     else
7.        sort the points in D in decreasing order of
                    minimum distances from points in C
8.     endif
9.     for each x in sorted order
10.       if (LOF(x, mp) ≈ 1)
11.          insert x in C
12.          break
13.       endif
14.    endfor
15.    m++;
16. endwhile
17. return C
```

Fig. 2. ROBIN: Robust Initialization Algorithm

Unlike previous work in seed selection (Astrahan, 1970), or the work in outlier detection (Breunig et al., 2000), we do not desire to compute the density or LOF value of all the points, since that is computationally expensive. Our aim is to judiciously compute the LOF values on demand, while selecting the seeds. The ROBIN robust initialization algorithm is outlined in Figure 2. In the figure, $\mathcal{D}$ is the dataset; k is the number of clusters or seeds desired, and mp is the number of neighbors to consider while computing the LOF.

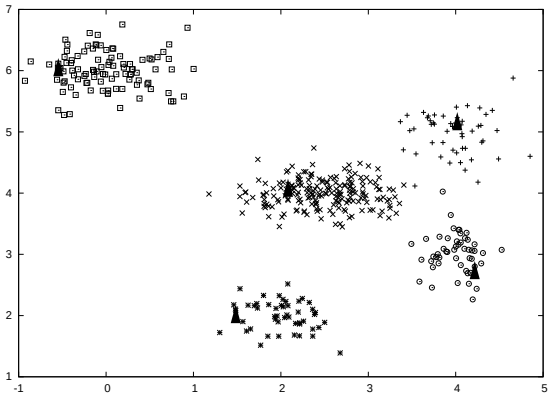
For any reference point r, ROBIN first sort the points in decreasing order of their distances from r (line 5). In this sorted order, it selects the first point that is not an outlier as validated through its LOF value. Note that the point r is not a center, it is only used to find the first seed center, which we take to be the point at the "edge" of the dataset, i.e., one having the largest distance from r and one which also has an LOF value close to one.

The subsequent seed points are obtained in a similar manner, by first sorting the points in decreasing order of their minimum distance to seed centers already in the set $\mathcal{C}$ (line 7). Note that computing the distances of all points from a chosen seed center is not an overhead, as the k-means algorithm would compute these distances anyway in its first iteration; we can simply skip that

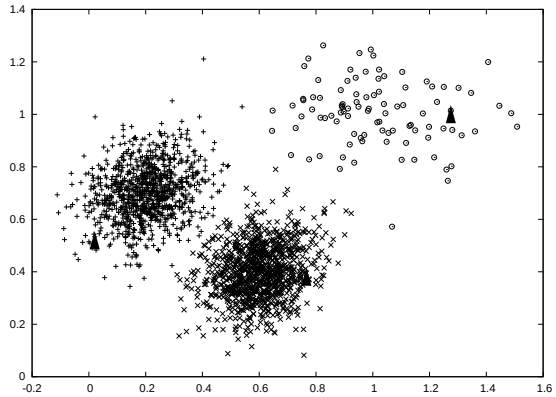
step in the subsequent k-means run. In each iteration of the while loop in line 3, ROBIN finds a new seed center. The inner for loop (line 9) considers the points in sorted (decreasing) order of distance, and breaks as soon as it obtains a point which has an LOF value approximately equal to one (in our implementation, this threshold is kept fixed at 1.05). Once k center points are found, the algorithm breaks. Note that if the initial reference point is chosen as the origin, the output of ROBIN is deterministic and point order insensitive.

Complexity: In terms of the computational complexity of ROBIN, we can see that the while loop (lines 3-16) is repeated k times. Thus, in aggregation, line 5 (and line 7) takes $O(knd)$ time to compute the distances of each point to each of the seeds in $\mathcal{C}$. Running the sort routine (on the same lines) for k times takes $O(kn \log n)$ time. At first glance, the for loop (line 9) appears to be expensive, since it can potentially loop over the $O(n)$ points in sorted order of distance, when all points are outliers; and also, computing the LOF value for x (line 10) can also take time $O(n)$, for a total of $O(n^2)$ time. In fact, this will never happen because of the definition of LOF, which computes the outlier factor locally; so a point is outlier with respect to other points in the neighborhood that are not outlier. So, in realistic cases, the for loop in line 9 is in fact repeated only a few times, which can practically be estimated to be a constant, L. Of-course, for very noisy dataset, L will assume a higher value, but is still negligible with respect to n. Computing the LOF takes $O(n)$ in high dimensions, but in lower dimensions one can use a range search index like kd-trees (Samet, 2006), which can compute the mp nearest neighbors in $O(n^{1-1/d} + mp)$ time, where d is the dimensionality of the dataset. Since, $mp \ll n$ (for example, mp is 25, whereas n is in the order of 10k), we can ignore the mp in the above complexity term, and aggregating for the k runs, yields an overall complexity of the for loop to be of $O(k \cdot n^{1-1/d} \cdot L)$. Combining all the three costs (distance computation, sorting and LOF computation) together, the total complexity is equal to $O(ndk + kn \cdot \log n + kL \cdot n^{1-1/d})$ Thus, the total worst case computation complexity is $O(n \log n)$ in terms of n and linear in terms of k, and d.

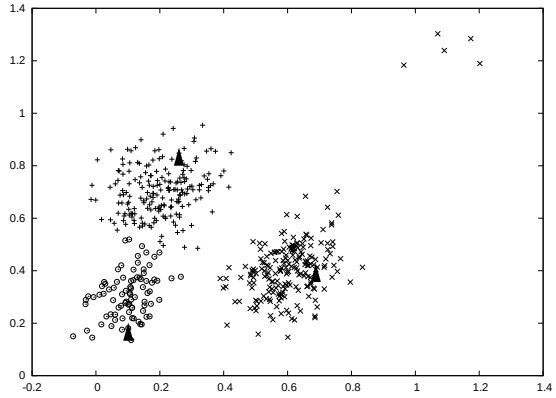
ROBIN Seed Selection: In Figure 3(a) we show an example dataset that has five clusters. We also show the initial seeds (triangles) chosen by ROBIN with $mp = 10$. We see that exactly one seed is chosen from each cluster. ROBIN can also handle clusters with different densities. Since LOF considers a point as outlier based only on the local density, it does not penalize any low density cluster while selecting a seed from that cluster. For example Figure 3(b) shows 3 clusters, with sizes 1000, 1000 and 100 points, respectively. ROBIN chose three different seeds (triangles), one from each of these clusters. Also, ROBIN always avoids the noise points while choosing the seeds. In Figure 3(c), there are three clusters and a few noise point at the upper left corner; ROBIN initialization did not consider any of those noise points as seeds.



(a) No merging/splitting



(b) Density Insensitive



(c) Outlier Insensitive

Fig. 3. ROBIN seeds (triangles)

4 Empirical Results

We performed extensive experiments to evaluate the benefits of ROBIN initialization scheme using synthetic and real life datasets. All the experiments were performed on Mac G5 machine with 1.66 GHz processor, running the Mac 10.4 OS.

Table 1
Comparison on synthetic datasets. The distortion scores are shown for each method.

d	k	Optimal	ROBIN	Random		Subsample		k-means++		KKZ
				min	avg	min	avg	min	avg	
8	10	7738	7755	7904	8421	7887	8092	8008	8508	9204
	25	9365	9382	9774	10185	9639	10044	9641	9951	10743
	50	8694	8754	9244	9565	9136	9407	9289	9598	17042
16	10	16865	16882	17406	18496	17356	18314	16870	17951	19346
	25	17241	17261	18298	19219	17647	18812	17732	18550	20567
	50	17580	17622	18866	19507	18469	19084	18974	19661	21632
24	10	26149	26150	26706	28733	26150	28340	26755	28860	29413
	25	22233	22261	23241	24582	23034	23942	22803	23787	27052
	50	21453	21467	22838	23818	22477	23387	23003	23762	26599

The synthetic datasets were generated as follows: For a given k (number of clusters) and d (number of dimensions), we generate k Gaussian clusters, each having m data points, where m is chosen uniformly between 100 to 1000. So, it is possible that the size of one cluster is about ten times larger than that of another. Each Gaussian cluster is obtained from a mean (μ) and a covariance matrix ($\mathbf{\Sigma}$). Each component/dimension of the mean (μ_i) was chosen uniformly within 0 and 10. So, all the cluster means reside in the length 10 hypercube in d-dimensions. The covariance matrix, $\mathbf{\Sigma}$ was first chosen as a diagonal matrix, which was later rotated with a random rotation matrix. Each entry along the diagonal of $\mathbf{\Sigma}$ is chosen randomly in the interval $[0.2{*}w, 0.8{*}w]$, where $w = s * \sqrt{d}$, and s is a parameter that can be used to control the cluster width. With higher w, the generated clusters are more noisy; i.e., the points are further away from the mean point. w is also used to separate the means of the clusters, i.e., when generating the cluster means, if the distance between a mean is within $2w$ of an already generated mean, the new mean is ignored, until all k mean points are obtained that are somewhat separated. The $\sqrt{d}$ part in w stretches the clusters in higher dimensions, so that the clusters cover the space uniformly over different dimensions. We also injected about 2% noise points in the dataset, which are distributed uniformly in the length 10 hypercube.

4.1 Results on Synthetic Datasets

The experimental results for the synthetic datasets are shown in Table 1. The distortion value is used to compare the performance of ROBIN against random initialization and other different initialization schemes: KKZ (Katsavounidis, 1994), Subsample (Bradley & Fayyad, 1998), and KMeans++ (Arthur & Vassilvitskii, 2007). We present the result for three different dimensions (d): 8, 16 and 24, and for three different number of clusters (k): 10, 25 and 50. For algorithms that are not deterministic, like k-means, Subsample and KMeans++, each algorithm was executed for 50 different runs. We show the minimum and average distortion values over these runs. For ROBIN, we set the minimum points threshold $mp = 10$. In fact, the clustering score does not vary much with different choices of mp, as we show in the sensitivity section below. For Subsample, we always take a 5% sample of the dataset.

Since these datasets were generated synthetically, the actual mean of each cluster is known. To obtain a benchmark score of distortion, we ran k-means algorithm with those known means as the initial seeds. The distortion score obtained from this run is recorded as the optimal score in Table 1. Note that, without the presence of noise, the above seeds indeed obtain the global optimal clustering. However, all the dataset in the above experiments had 2% noise, so, in some cases, the actual global optimal clustering mean may deviate from the generated means. Since, the noise level is not significant, we consider the score obtained with the known seeds as the optimal distortion value.

Table 1 shows that for all the dimensions and number of clusters, the random initialization could not find a clustering with a distortion score as good as ROBIN in 50 different random runs. In fact, ROBIN achieves about 2% to 5% better (lower) distortion scores than the best of random, and 5% to 10% better than the average random score. Further, ROBIN is deterministic, hence is executed only once to achieve these clustering solutions. ROBIN also yields a distortion score close to the global optimal, which suggests that the cluster means obtained via ROBIN seeding are actually in very close proximity to that of the original cluster means. Comparing to other initialization schemes, ROBIN scores are much better.

4.2 Results on Real Datasets

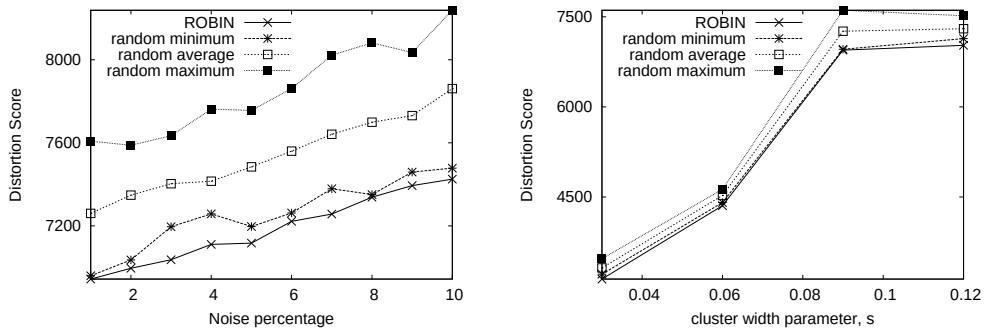
Table 2 shows the comparison of ROBIN with the random, Subsample, k-means++ and KKZ approaches. These datasets were taken from the UCI machine learning archive (`http://archive.ics.uci.edu/ml/`). These datasets are generally small, with the exception of poker, which has one million points.

Since we do not know the optimal distortion score for these, we simply compare the distortion values across the methods. Random, Subsample and k-means++ were run 50 times. For these datasets, the performance of ROBIN is as good as the best result obtained by subsample, KKZ, and k-means++. On the small datasets, Subsample does better, but for the large poker dataset, ROBIN is the best.

4.3 Sensitivity Experiments

4.3.1 Noise Sensitivity

We performed experiments to check how the performance of ROBIN varies with the noise level of the dataset. We define the noise from two standpoints: (1) completely random noise, where the random points are spread uniformly over the entire space; (2) cluster noise, where the noise level deviates a data point from the mean of a cluster. The second kind of noise can be controlled by varying the cluster width parameter of the synthetic data generator. We analyze the performance of ROBIN in the presence of both these kinds of noise.



(a) Random Noise (b) Cluster Noise

Fig. 4. Noise Sensitivity: ROBIN vs. Random

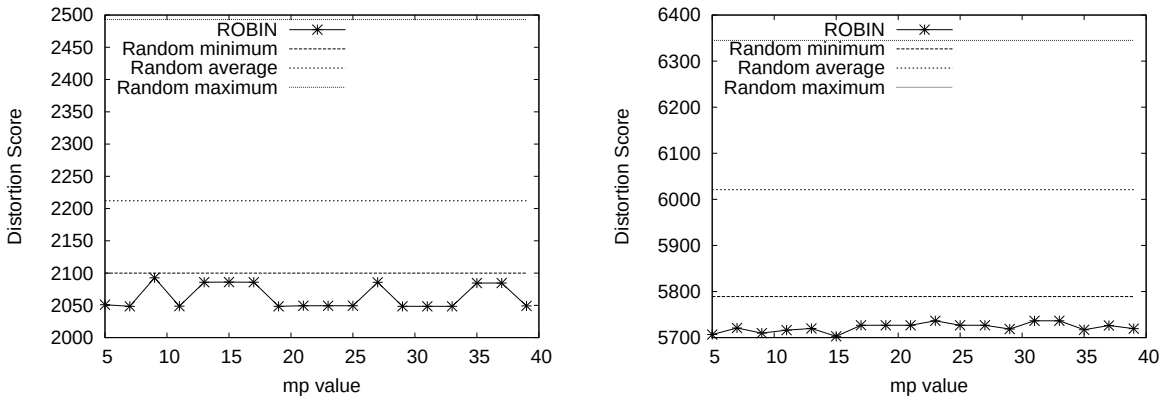
For the random noise experiment, we first generated a dataset without any noise. Then, we construct ten different datasets by adding an increasing number of noise points to the above dataset. The noise level of different datasets ranged from 1% to 10%. In Figure 4(a), we compare the distortion scores of random initialization with that of ROBIN. Over the 50 runs for k-means, we show minimum, average and maximum distortion scores. For ROBIN $mp = 10$ as before. In the base dataset used above, we set $d = 8$, and $k = 10$. Similar result were obtained for other values of dimensions and clusters, and hence are not shown. It is evident that the performance superiority of our ROBIN method over random prevails for different noise levels of the dataset, since the distortion score is always better (lower) than the minimum of random seeds distortion score.

For the cluster noise experiments, we generated datasets with different values of s for different noise levels of a cluster: we used s values of 0.03, 0.06, 0.09, and 0.12. As s value is increased, clusters will have more noise. Figure 4(b) shows that ROBIN performs better than the best of fifty runs of random seeding of k-means for all different cluster noise levels.

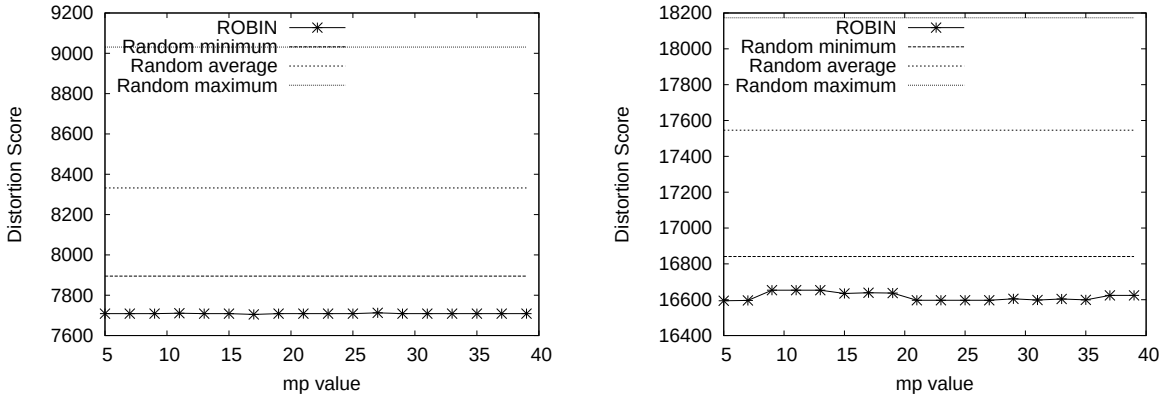
4.3.2 Parameter Sensitivity

ROBIN takes the minimum points parameter mp, to compute the LOF value of a point, to decide whether the point is an outlier or not. We observed that for a good range of values for mp, identical clustering solutions are obtained in both synthetic and real world datasets. In cases when different clustering solutions are obtained, they produce similar distortion values (clustering scores), which are mostly better than distortion score of random initialization. In (Breunig et al., 2000), the authors provided a guideline to choose the value of mp; the summary of their suggestions is that mp should not be higher than the smallest sized cluster and should not be lower than the largest sized noise cluster. For typical dataset, the difference of these two estimates is comfortably large; and, what is more important is that a reasonable guess of these values can be easily made. Note that cluster seed initialization methods that depends on a distance threshold to keep the seeds well apart do not have this flexibility and are very sensitive to the choice of parameter values. Moreover, it is hard for a user to guess such a value without data visualization, which is effectively possible only for low dimensional datasets. Although, one can visualize high dimension data using techniques like ISOMAP, but guessing a right value of distance threshold from such view is as hard as random guessing.

The range of acceptable values of mp varies depending on the noise level of a dataset. Intuitively, the acceptable range of values can be well characterized by the signal-to-noise-ratio (SNR), where the signal represents the clusters, and the noise the set of outlier points. The larger the SNR, the more flexible it is to choose the mp value. In Figure 5 we show the distortion scores for different values of mp for increasing dimensions, keeping the number of clusters fixed at $k = 15$. In each of the plots, we also show the random seeding scores: the minimum, the average and the maximum for 50 different runs. We observe that for a large range of mp values the distortion score for ROBIN does not change much, and more importantly ROBIN produces clusterings that have distortion scores less than the minimum of the random seeding.



(a) $d = 4$, 15 Clusters (b) $d = 8$, 15 Clusters



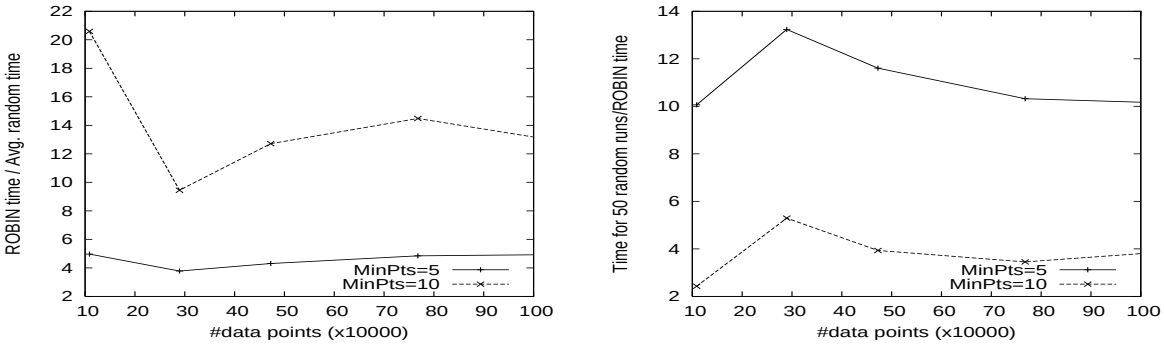
(c) $d = 16$, 15 Clusters (d) $d = 24$, 15 Clusters

Fig. 5. Distortion scores with varying values of mp

4.4 Scalability Experiments

We also compared how ROBIN scales with the number of points in the dataset. The experimental results for the scalability tests are shown Table 3. These results are for 16 dimension, 30 clusters and 5% random noise. The minimum and average scores for the random seeding approach are shown, as well as the average time for a single run. For ROBIN the table shows the distortion score and total time for $mp = 5$ and $mp = 10$. We can see that as the number of points increases, both methods scale linearly, as expected. Furthermore, ROBIN, still has the best distortion scores, which are not much different for $mp = 5$ or $mp = 10$. However in terms of time, we do see a big improvement when ROBIN uses $mp = 5$, a reduction by a factor of 3, since the lower the mp value the cheaper the LOF computation.

Comparing the time with the random seeding, we plot in Figure 6(a) the ratio of the running time of ROBIN to the average running time for a single run of the random seeding. In general, we see that a single random seeding run is about 10 times faster than our approach. On the other hand, if we let the



(a) (b)

Fig. 6. Running time comparison of ROBIN with a single (6(a)) and 50 runs of random seeding (6(b)).

random seeding to run for 50 runs, in Figure 6(b), which plots the ratio of the total time of ROBIN versus the total of 50 random runs, we see that overall ROBIN is 10 times faster, and of course yields a much better distortion score. If we let the random seeding run over many more iterations, it will likely improve the score, but at the cost of even more time. These results confirm that ROBIN is indeed robust, efficient, and scalable.

5 Conclusions

In this paper we proposed a new method, ROBIN, for robust initialization of seed centers for use in a partitional clustering method like k-means. We studied the state-of-the-art initialization methods, and noticed that either they were computationally expensive, or they were not able to handle outliers or multi-scale clusters with variable density of points. We outlined the desiderata for good seed selection methods, and showed that ROBIN satisfies all those criteria by imposing explicit constraints on the seeds to be chosen.

An extensive evaluation on real and synthetic datasets confirms that ROBIN is computationally efficient. It takes only one parameter mp, the number of neighboring points to consider, and is not very sensitive to the value of this parameter. It is deterministic, insensitive to outliers, and is robust against multi-scale clusters.

References

Arthur, D., & Vassilvitskii, S., 2007. k-means++: The Advantages of Careful Seeding, Proc. of Symposium of Discrete Analysis, 1027-1035.

Astrahan, M.M, 1970. Speech Analysis by Clustering, or the Hyperphoneme Method, Stanford A. I. Project Memo, Stanford University.

Ball, G.H., & Hall, D.J., 1967. PROMENADE– An online pattern recognition system, Stanford Research Inst. Memo, Stanford Univeristy.

Berkhin, P., 2002. Survey of clustering data mining techniques, Technical report, Accure Software, San Jose, CA.

Bezdek, J., Keller J., Krishnapuram, R., Pal, N, 2005 Fuzzy Models and Algorithms for Pattern Recognition and Image Processing Springer Publications

Bottou, L., & Bengio, Y., 1995. Convergence Properties of the k-means Algorithms, Advances in Neural Information Processing Systems, 7, 585-592.

Bradley, P.S., & Fayyad, U.M., 1998. Refining Initial Points for k-means Clustering, Proc. of the Fifteenth Int. Conf. on Machine Learning, 91-99

Breunig, M.M, Kriegel H., Ng, R.T., & Sander, J., 2000. LOF: Identifying Density-Based Local Outliers, ACM SIGMOD Record, 93-104.

Clark, B.N., Colbourn, C.J., & Johnson, D.S., 1990. Unit disk graphs, J. of Discrete Mathematics, 86(1-3), 165-177.

Garey, M.R., & Johnson, D.S., 1979. Computers and Intractability: A Guide to the Theory of NP-Completeness, W. H. Freeman.

Elkan, C., 2003. Using the Triangle Inequality to Accelarate k-Means, Proc. of the Twentieth International Conference on Machine Learning, 147-153.

Ester, M., Kriegel, H., Sander, J., & Xu, X., 1996. A Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise, Proc. of KDD.

Forgy, E. 1965. Cluster analysis of multivariate data: Efficiency vs. interpretability of classifications, Biometrics, 21, 768

Hichem, F., & Krishnapuram, R., 1997 Pattern Recognition, 30(7), 1109-1119.

Hochbaum, D., & Shmoys, D., 1985. A Best Possible Heuristic for the k-Center Problem, Mathematics of Operation Research, 10(2), 180-184.

Jain, A.K., & Dubes, R.C., 1988. Algorithms for Clusterind data, Pentice-Hill.

Jardine, C.J., Jardine, N., & Sibson, C., 1967. The structure and construction of Taxonomic Hierarchies, Math. Bioscience, 1(2), 173-179.

Katsavounidis, I., Kuo, C.C.J, & Zhen, Z., 1994. A new initialization technique for generalized Lloyd iteration, IEEE Signal Processing Letter, 1(10), 144-146.

Khan, S.S., & Ahmad, A., 2004 Cluster center initialization algorithm for k-means clustering, Pattern Recognition Letter, 25(11), 1293-1302.

Lloyd, S.P., 1982. Lease square quantization in pcm, IEEE Transactions on Information Theorey, 28(2), 129-136

MacQueen, J.B, 1967. Some Method for Classification and Analysis of Multivariate Observations, Proc. of Berkeley Symp. on Mathematical Statistics and Prob., Berkeley, U. of California Press, 1, 281-297

Parsons, L., Haque, E., & Liu, H., 2004. Subspace Clustering for High Dimensional Data: A Review, SIGKDD Explorations Newsl. 6, 90105.

Samet, H., 2006 Foundation of Multidimensional and Metric Data Structures, Morgan Kaufmann Publishers.

Shi, J. & Malik, J., 2000 Normalized Cuts and Image Segmentation, IEEE Transactions on Pattern Analysis and Machine Intelligence, 22(8), 888-905.

Tou, J., & Gonzales, R., 1974 Pattern Recognition Principles, Addision-Wesley, Reading, MA.

Wu, X., Kuman, V., Quinlan, J.R., Ghosh, J., Yang, Q., Motoda, H., Mclachlan, G.J., Ng, A., Liu, B., Yu, P.S., Zhou, Z., Steinbach, M., Hand, D.J., & Steinberg, D., 2008 Top 10 algorithms in data mining, Knowledge and Information Systems Journal, 14(1), 1-37.

Table 2. Comparison on real datasets

Dataset	$\lvert\mathcal{D}\rvert$	d	k	ROBIN	Random		Subsample		k-means++		KKZ
					minimum	average	minimum	average	minimum	average	
Yeast	1484	8	10	2465	2467	2564	2418	2514	2500	2525	2472
poker-hand	1000000	10	10	6.18 $\times 10^6$	6.27 $\times 10^6$	6.34 $\times 10^6$	6.28 $\times 10^6$	6.33 $\times 10^6$	6.26 $\times 10^6$	6.32 $\times 10^6$	6.22 $\times 10^6$
ecoli	336	7	8	65.34	67.17	67.29	64.36	67.63	64.20	66.38	67.41
wdbc	569	30	2	2402	2402	2407	2402	2410	2402	2434	2411
wine	178	13	3	500	500	505	500	505	500	527	510
Pendigit	7494	15	10	17051	16932	17776	16977	17706	16992	17477	17823
OptDigit	3823	64	10	110289	108699	111912	108557	111926	108940	112911	114673

Table 3
Scalability with varying number of data points

$\|D\|$	Random			ROBIN ($mp = 5$)		ROBIN ($mp = 10$)	
	min. score	avg. score	time	score	time	score	time
4986	17060	18019	0.29	16284	3.01	16287	6.98
11685	39073	42707	0.68	38505	5.07	37491	10.68
49043	159392	170279	2.99	155660	17.84	155630	65.92
108721	354538	374245	7.04	354835	34.98	343131	144.97
289517	934897	1007988	18.79	937745	70.99	943050	177.46
472385	1505000	1598921	31.91	1522880	137.46	1515540	405.66
767964	2477390	2621665	53.79	2564730	260.60	2462570	778.93
1048225	3356920	3491258	69.80	3298390	344.16	3344840	900.80