Sparse and Robust Methods for K-Means Clustering

Adam Norton
Guanglei Cao
Ying He

Dec 11, 2013

1 Motivation

K-means is a very broadly used clustering method. It is derived by minimizing the within cluster sum of squared distances in order to assign each sample point to one of K clusters. The implementation of this algorithm is strongly dependant on the initialization and finding the local minimum of the performance function during each iteration. The traditional, or nave k-means suffers from several defects. First, it is easy to construct examples where k-means performs rather poorly in the presence of a large number of noise variables, ie., variables that do not change from clusters to cluster. Secondly, naive k-means is not robust to data with many irrelevent features (sparcity), nor outliers. To overcome these defects, we propose a trimed, sparse k-means method. Finally, when we do not know what the real data looks like and what the final clusters tend to be, even with repeated initializations, we may select inappropriate initial values, which will make the "nave" k-means converge to a "too local" minimum. We use the weighted k-means algorithm, which is robust against bad initialization.

2 Project Goals

By proposing the trimmed k-means method, the Kernelized k-means algorithm and the weighted k-means algorithm, we hope to compare the behavior and robustness of these methods. Specifically, by doing classification an the abnormal data set, i.e. with outliers, sparse features and bad initializations. Through this project, we hope to gain a deeper understanding of the variations of the k-means clustering method and the general problems associated with sparsity, bad initialization and outliers.

3 Related Work

To refine the "naive" k-means method, Kaufman, L and Rousseeuw, P.J.(1987) proposed the k-medoids method, which applies to skewed data. Witten and Tibshirani (2010) proposed an alternative to classical k-means - called sparse K-means (SK-means) which simultaneously finds the clusters and the clustering variables. For the purpose of online learning, Yumi Kondo, Matias Salibian-Barrera, Ruben Zamar proposed the online k-means algorithm to deal with the sequence arrival data. Ahmed, Mohamed N.; Yamany, Sameh M.; Mohamed, Nevin; Farag, Aly A.; Moriarty, Thomas (2002) proposed another robust fuzzy-clustering method.

4 Weighted K-Means

In this section, we focus on the situation when the cluster initialization is very far from the real cluster centers, or more specifically, the intialized clusters cannot well reflect the real clustering. If an algorithm can deal with this senario, it should be able to behave well in a more general or lenient situation. The performance function for K-means is:

$$J_K = \sum_{i=0}^{N} \overset{K}{\underset{j=0}{min}} ||x_i - m_j|| \tag{1}$$

Note that this performance function detects only the prototypes (cluster centers) closest to data points and then distributes them to give the minimum performance, thus determining the cluster partitioning. However, prototypes which are far away from the data are not utilized. This may result in a bad prototype which will never reflect any cluster. To overcome the "too local" solution produced by the above performance function, we instead consider the following

$$J_A = \sum_{i=1}^{N} \sum_{j=1}^{K} ||x_i - m_j||^2 \tag{2}$$

While this takes into consideration the relationship between all data points and prototypes, it provides no useful clustring information because

$$\frac{\partial J_A}{\partial m_k} = 0 \quad \Rightarrow \quad m_k = \frac{1}{N} \sum_{i=1}^{N} x_i \ \ \forall k \tag{3}$$

This gives a too "global solution". Thus a better performance function will seek a balance between finding a minimum performance with respect to an the existing valid clusters while also taking into consideration the relationship between all data points and prototypes. Thus, a good idea is to combine the above two functions to produce a one which produces neither a too "local" nor a too "global" solution. The performance function can be defined as:

$$J_1 = \sum_{i=1}^{N} \left[\sum_{j=1}^{K} ||x_i - m_j|| \right] \overset{K}{\underset{k=1}{min}} ||x_i - m_k||^2 \tag{4}$$

In this way we utilize the minimum distance and retain the global intersection which is necessary to insure that all prototypes join the intersection. Define

$$J_2(x_i) = \left[\sum_{j=1}^{K} \frac{1}{||x_1 - m_j||^p} \right] \overset{K}{\underset{k=1}{min}} ||x_i - m_{k_*}||^n, \qquad k_* = \overset{K}{\underset{k=1}{argmin}} \left(||x_i - m_k|| \right) \tag{5}$$

By taking the first order derivative with respect to m_r, we can derive the closed form solution:

$$m_r(t+1) = \frac{\sum\limits_{i \in V_r} x_i a_{ir} + \sum\limits_{i \in V_r, j \neq r} x_i b_{ir}}{\sum\limits_{i \in V_r} a_{ir} + \sum\limits_{i \in V_r, j \neq r} b_{ir}} \tag{6}$$

Where V_r contains the indices of data points that are closest to m_r. V_j contains the indices of all the other points and

$$a_{ir} = ||x_i - m_r|| + 2 \sum_{j=1}^{K} ||x_i - m_j||, \qquad b_{ir} = \frac{||x_i - m_{k_*}||^2}{||x_i - m_r(t)||} \tag{7}$$

There is another adapted version of the performance function, which is called the inverse weighted K-means Algorithm , and the performance function assiciated with this algorithm is:

$$J_2(x_i) = \left[\sum_{j=1}^{K} \frac{1}{||x_1 - m_j||^p} \right] \overset{K}{\underset{k=1}{min}} ||x_i - m_{k_*}||^n \tag{8}$$

The intuition behind this performance function is that it is minimized when the distance between the data and the closest prototype is minimal and the other prototypes are scattered widely throughout the data to give the largest possible value in the denominator of the second part. Taking the first order dirivative with respect to m and apply some basic algebra, we can derive that in the expression of a_{ir} and b_{ir} as follows:

$$\begin{aligned} a_{ir} &= -(n+p)||x_i - m_r(t)||^{n-p-2} - n||x_i - m_r(t)||^{n-2} \sum_{j \neq k_*} \frac{1}{||x_i - m_j||^p} \\ b_{ir} &= p\frac{||x_i - m_{k_*}||^n}{||x_i - m_r(t)||^{p+2}} \end{aligned} \tag{9}$$

5 Kernelized K-Means

Kernel methods are widely used in pattern recognition. The basic idea of kernel methods is to transform the data into a higher dimensional feature space represented by an inner product. One advantage of using a kernel method is that it can better detect outliers and thus be more robust. In this model, we will use the Gaussian Kernel method, which is expressed by the equation:

$$K_G(x, y) = e^{\frac{-||x-y||^2}{\beta}} \tag{10}$$

Where β is the dispersion estimation expressed by

$$\beta = \frac{1}{n}\sum_{i=1}^{n} ||x_i - \bar{x}||^2 \tag{11}$$

For each step, we update our cluster center using the following equation:

$$a_i = \frac{\sum\limits_{j=1, j\in a_i}^{n} K_G(x_j, a_i)x_j}{\sum\limits_{j=1, j\in a_i}^{n} K_G(x_j, a_i)} \tag{12}$$

6 Robust-Sparse K-Means

The RSK-Means algorithm, seeks to simultaneously solve the problem of feature sparsity and the the problem with outliers.

The traditional k-means algorithm attempts to minimize the within-cluster sum of squared distances. This is defined as:

$$WCSS = \sum_{j=1}^{d}\sum_{k=1}^{K}\frac{1}{n_k}\sum_{i,i' \in C_k} d_{i,i',j} \tag{13}$$

Where d is the number of features, K is the number of user selected clusters, C_k is the set of points assigned to cluster k, and $d_{i,i',j}$ is some distance function defined between points i and i' on feature j. Any distance function can be used, but we focus on the traditional Euclidean distance.

In order to deal with the sparity problem, we introduce a set of weights to the objective function in order to decrease the importance of the irrelevant features. However, this is not feasible with the current objective function. If we were to attempt this, we would want to do the following minimization:

$$\underset{w,C}{min} \sum_{j=1}^{d} w_j \sum_{k=1}^{K} \frac{1}{n_k} \sum_{i,i' \in C_k} d_{i,i',j} \tag{14}$$

However, this results in the trivial solution where all weights equal 0. Therefore, we must develop a new objective function. Instead, we will attempt to maximize the weighted between-cluster sum of squared distances.

$$BCSS = \sum_{j=1}^{d} \left[\frac{1}{n} \sum_{i=1}^{n} \sum_{i'=1}^{n} d_{i,i',j} - \sum_{k=1}^{K} \frac{1}{n_k} \sum_{i,i' \in C_k} d_{i,i',j} \right] \tag{15}$$

Intuitively, we would like to maximize the seperation of the clusters. Therefore, in order to deal with sparsity, we will impose a lasso constraint on the weights. This implies the following objective function

$$\begin{gathered} \underset{w,C}{max} \sum_{j=1}^{d} w_j \left[\frac{1}{n} \sum_{i=1}^{n} \sum_{i'=1}^{n} d_{i,i',j} - \sum_{k=1}^{K} \frac{1}{n_k} \sum_{i,i' \in C_k} d_{i,i',j} \right] \\ \text{such that} \quad ||w||^2 \leq 1, \quad ||w||_1 \leq s, \quad w_j \geq 0 \; \forall j \end{gathered} \tag{16}$$

The lasso constraint promotes sparsity in the weights through the L_1 norm constraint (as discussed in class), while the L_2 norm constraint promotes solutions with more than a single non-zero quantity.

This objective function implies an alternating algorithm. We first hold the weights constant and optimze the cluster partitions, Ck. Then we hold the cluster partitions constant and optimize the weights.

The first step is done by performing weighted k-means, e.g. the distance function now becomes weighted by the weights that are assumed constant in this step. However, we would also like to make the algorithm robust to outliers. Therefore, we introduce a new parameter, α and in the second step of k-means, we compute the new cluster centers while ignoring the α percent of the samples in that cluster with the farthest distance from the current cluster center. We thus iterate our modified version of k-means to convergence.

The second step is done by maximizing the objective function

$$\underset{w}{max} \sum_{j=1}^{d} w_j a_j \qquad \text{such that} \quad ||w||^2 \leq 1, \quad ||w||_1 \leq s, \quad w_j \geq 0 \; \forall j \tag{17}$$

Where a_j is the jth element of the (unweighted) BCSS. Again, we want to make the algorithm robust to outliers. With this in mind, we want to calculate each a_j without the points, O_w,that were trimmed in the final iteration of the previous step. However, we must also consider a potential problem. If a point is an outlier in a feature that has a small weight in the current iteration, it would most likely not have been trimmed in the previous step. To cope with this, we also find the α percent of points that are have the farthest unweighted distance from their cluster centers, O_e. We then calculate each a_j excluding $O_w \cup O_e$.

We choose to view the problem from a geometric standpoint. We will maximize the objective function ignoring the L_1 constraint and modify the solution if the constraint is violated. This is feasible due to the similarity of the L_1 and L_2 constraints. Now, we realize that we are trying to maximize a linear objective function for which the L_2 constraint is a d-dimensional sphere of radius 1, centered at the origin. Since the objective function is linear, the solution to the maximization problem will be found at the boundary of the L_2 ball in the direction of the projection of the gradient of the objective function. In this case, we have

$$\nabla a^T w = a \tag{18}$$

Therefore, the solution can be found at radius 1 of the projection of the gradient:

$$w = \frac{a}{||a||} \tag{19}$$

Notice that a_j is a sum of squares and therefore, must be positive, thuse preserving our all-positive constraint. We now need to check the L_1 constraint however. If this constraint is violated, we just need to move the solution in the direction of the negative gradient until it lies on the boundary of the L_1 ball. The solution to this, while maintaining the all-positive constraint on w is:

$$w(\Delta) = \frac{(a - \mathbf{\Delta 1})_+}{||(a - \mathbf{\Delta 1})_+||} \tag{20}$$

Where delta is chosen to be the root of the equation $||w||_1 = s$. The root of this equation can be found rather easily by noting that its value must lie somewhere between 0 and max(w(0)). Therefore, we can do what is essentially a binary search by trying the midpoint and checking $||w(\text{midpoint})||$ against s.

All steps of the algorithm have been derived. Therefore, all that remains is to iterate until the stopping criterion:

$$\frac{\sum_{j=1}^{d} |w_j^r - w_j^{r-1}|}{\sum_{j=1}^{d} |w_j^{r-1}|} < 10^{-4} \tag{21}$$

Where w_j^r represents the jth weight at iteration r.

7 Evaluation

For our dataset, we used the MNIST handwritten digit numbers to test the different clustering methods. From digit numbers from 0 to 9, we choose the following numbers: 0, 1, 3, 4, 5, 6 to test our methods. Each digit number has roughly 1000 plots with dimension 28×28. During testing, we choose our number of clusters to be 6. To observe our result, we plotted each cluster center and computed the purity function: $purity(W, C) = \frac{1}{N} \sum_k \underset{j}{max} |w_k \cap c_j|$ where w_k is the set of points in cluster k, and c_j is the set of points in class j

We used two different initialization approaches to observe how our methods worked. For the first initialization approach, we shuffle the data and pick K random data points as the initialized cluster centers. For the second approach, we choose all K initializers very close to each other, i.e. all drawn from handwritten digit number 1.

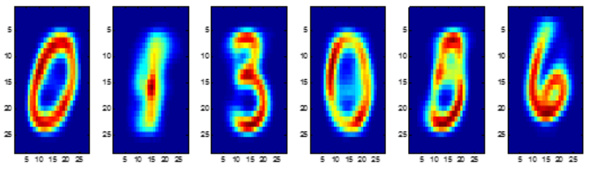
Figure 1: Traditional K-Means (Good Initialization)

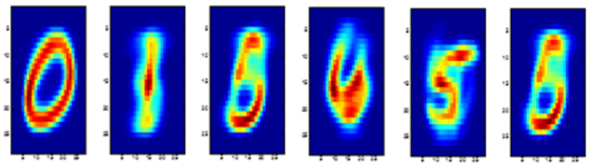
Figure 2: Traditional K-Means (Bad Initialization)

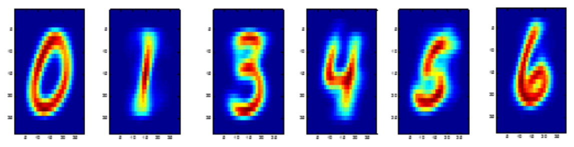
Figure 3: Kernelized K-Means (Good Initialization)

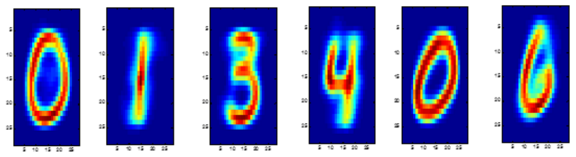
Figure 4: Kernelized K-Means (Bad Initialization)

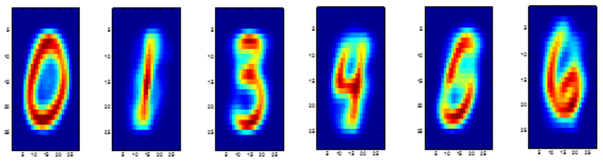
Figure 5: Weighted K-Means (Good Initialization)

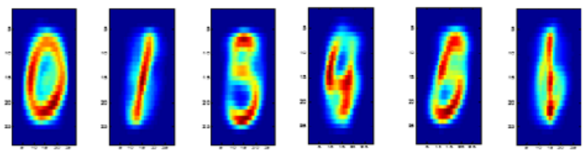
Figure 6: Weighted K-Means (Bad Initialization)

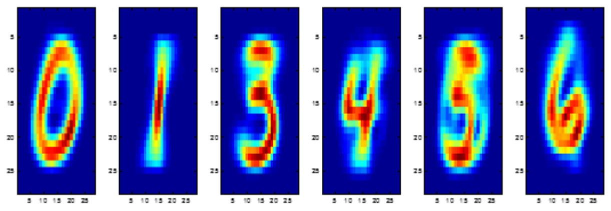
Figure 7: Robust-Sparse K-Means

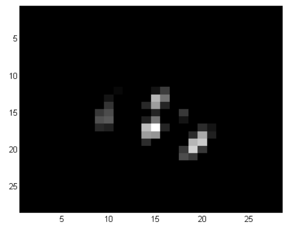
Figure 8: Relevant Features

	Naive(GI)	Naive(BI)	Kern(GI)	Kern(BI)	WKM(GI)	WKM(BI)	RSK
Purities	0.4967	0.4741	0.7742	0.5867	0.6872	0.5766	0.6535

From the above results, it can be seen that each method is relatively good at what it was designed for. The weighted k-means algorithm does a good job of recovering from a bad initialization. The RSK-Means algorithm does well at reducing the feature set to only those relavent, and is somewhat more robust to outliers. Interestingly, the kernelized k-means algorithm turned out to provide the best results. The gaussian kernel seems to be capable of assigning the best weights.

8 Contributions

All group members worked together on many aspects of the project. However, Adam Norton worked mainly on RSK-Means, Guanglei Cao worked mostly on Weighted K-Means and Ying He worked mostly on Kernelized K-Means.

References

[1] arbakh, Wesam, and Colin Fyfe. ”Online Clustering Algorithms.” (n.d.): n. pag. University of Paisley. Web.

[2] oyd, S. ”Linear and Quadratic Problems.” University of Siena. Web.

[3] uchi, John, Shai Shalev-Shwartz, Yoram Singer, and Tushar Chandra. ”Efcient Projections onto the 1-Ball for Learning in High Dimensions.” N Proceedings of the 25th International Conference on Machine Learning (2008): n. pag. Web.

[4] ondo, Yumi, Matias Salibian-Barrera, and Ruben Zamar. ”A Robust and Sparse K-means Clustering Algorithm.” (n.d.): n. pag. Web.

[5] ondo, Yumi. Robustication of the Sparse K-means Clustering Algorithm. Thesis. The University Of British Columbia, 2011. N.p.: n.p., n.d. Print.

[6] eCun, Yann, Corinna Cortes, and Christopher J.C. Burges. ”THE MNIST DATABASE.” N.p., n.d. Web.

[7] ubg-Kang, Tainan. ”Kernelized K-Means Algorithm Based on Gaussian Kernel.” Advances in Control and Communication 137 (2012): 657-64. Web.

[8] ou, Hui, and Trevor Hastie. ”Regularization and Variable Selection via the Elastic Net.” Journal of the Royal Statistical Society 67 (2005): 301-20. Web.