Shrinking Methods Used In Non-Parametric Clustering

Yongyi Wu
Department of Computer Science
Simon Fraser University
Burnaby, BC V5A 1S6
yongyiw@sfu.ca

Abstract

One approach to realize non-parametric clustering methods is preprocessing the unlabeled data to find the parameters like the number of clusters by the algorithm itself. Shrinking methods is such a way that moves data points towards the cluster center in order to gain a better view of the data. In this report, I implemented two different kinds of shrinking methods: gravitational[2] and KNN median shrinking[3] algorithms. The performance and features of both methods are described and analyzed.

1 Introduction

Clustering [1] is known as a kind of unsupervised learning whose purpose is trying to find hidden structure in unlabeled data. The criterion of clustering is maximizing the intra-class similarity and minimizing the inter-class similarity. One of the classic clustering methods - K-means methods optimizing the *distortion measure J* by iteratively minimizing *J* with respect to $\{r_{nk}\}$, the assignment of every data points, keeping the set of cluster centers $\{\mu_k\}$ fixed and minimizing *J* with respect to the $\{\mu_k\}$, keeping the $\{r_{nk}\}$ fixed.

Because K-means is based on least square estimate, one problem is that it will be spoiled when noise is introduced. Moreover, the other limitation is that users should provide K as the parameter to the algorithm.

When the number of cluster is not known, the clustering techniques are called unsupervised or non-parametric. To handle the non-parametric clustering problems with noise and estimating the number of clusters, one idea is to shrink the data points towards cluster centers. Therefore we can get the number of clusters by counting the convergence points, and also some weak noise can be eliminated. Methods about shrinking the data points are proposed from two angles, one is gravitational clustering which regards each data point as a particle of unit mass with zero velocity gradually moving toward the cluster center due to the gravitation, and the other one is mean-shift clustering[7] which originates from the idea in kernel density estimation.

In this report, I implement two shrinking methods, one is as representative of gravitational clustering from [2], and another is a variation of mean-shift algorithm, the local shrinking method, mentioned in [3]. The two methods will be described in detail in Section 2. In Section 3, there are experiment results and analysis of the performance and features of both algorithms. At last, we will present our conclusions in Section 4.

2 Approach

2.1 Gravitational Method (G-Method)

The first shrinkage method is gravitational method based on the Gravitational Law and the second Newton's motion Law [2]. The basic ideas are: 1) If a data point o in the cluster A, then o must be exerted a higher gravitational force from $o_i \in A$ than $o_j \notin A$. Then o will move in the direction of the center of the cluster. 2) If o is a noise point, then the gravitational force exerted on in from other points is so small that the point is almost immobile. Therefore, noise points will not be assigned to any clusters.

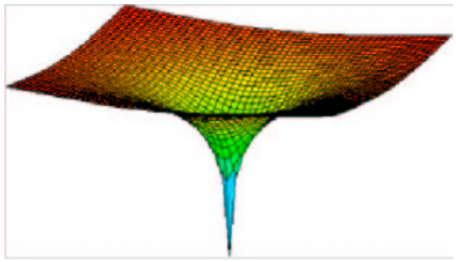

Figure 1. The Gravitational Field

With the guidance of the intuition, according to this paper we first built a fundamental physical model, and derived the formula of approximation of the movement of object, which is:

$$x\big(t + \Delta(t)\big) = x(t) + v(t)\Delta(t) + \frac{a(t)\Delta(t)^2}{2}$$

Combining with the gravitational force equation:

$$F(t) = \frac{Gm_x m_y}{\|d(t)\|^2}$$

Together we derive:

$$y\big(t + \Delta(t)\big) = y(t) + v(t)\Delta(t) + d(t)\frac{Gm_x\Delta(t)^2}{2\|d(t)\|^3}$$

By assuming the velocity at any time, *v(t)*, as the zero vector and $\Delta(t) = 1$. We can simply the formula as:

$$x(t+1) = x(t) + d(t)\frac{G}{\|d(t)\|^3}$$

As we get the approximated shrinking movement function, we can set a threshold ε such that if $dist(x_i, x_j)$ is less than ε, the points x_i and x_j are considered as from the same cluster and stored in a union-find set.

In this algorithm, the shrinkage is inside the cluster at first, but after the clusters are dense, they would attract to each other which results in the merge among clusters. This will be depicted in Section 4 in detail. The method to deal with this is using a constant proportion called decay term (ΔG), which reduce the effect of G for each iteration.

The pseudo-code is given below:

```
GRAVITATIONAL( x, G, △(G), M, ε)
1   for i=1 to N do
2     MAKE(i)
3   for i=1 to M do
4     for j=1 to N do
5       k = random point index such that k ≠ j
6       MOVE( x_j , x_k )  (see Eq (2.22))
7       if dist²( x_j , x_k ) ≤ε  then UNION( j, k)
8    G = (1-△(G))*G
9   for i=1 to N do
10    FIND(i)
11  return disjoint-sets
```

2.2 Local Shrinking Method (LS Method)

Local shrinking method [3] does well on noise eliminating and data sharpening which is beneficial for us to find the number of clusters. In this algorithm, each data point will be "shrunken" toward a denser region. The movement is determined by the coordinate-wise median of its K-nearest neighbors as the median is more robust than the mean. Therefore the iteration is defined as:

$$y_i^{[t+1]} = median_{y_j^{[t]} \in N_k(y_i^{[t]})} y_j^{[t]}$$

Where $N_k(y_i^{[t]})$ is the smallest neighborhood that contains K-nearest neighbors of the data point $y_i^{[t]}$. The amusing thing is that we are trying hard to get rid of K, the number of clusters, but here we need to fix another K in K-nearest-neighbor. Fortunately, the method is not that sensitive to K in K-near neighbor which allow us choose K more flexible than that in K-means. Besides, we also need to fix the low bound threshold ε for the diameter of the cluster and upper bound M for the iteration.

The pseudo-code is given below:

Step 1: *Initialize the iteration number* $t \leftarrow 1$. *Backup original data points* $Y^{[t]} \leftarrow Y$ *and* $Y^{[t+1]} \leftarrow Y$.
Step 2: *For each point* $\mathbf{y}_i^{[t]}$, *update coordinates*: $\mathbf{y}_i^{[t+1]} \leftarrow \text{median}_{\mathbf{y}_j^{[t]} \in N_K(\mathbf{y}_i^{[t]})} \mathbf{y}_j^{[t]}$.
Step 3: $d \leftarrow \max_{i=1}^{n} \max_{j=1}^{p} |y_{ij}^{[t]} - y_{ij}^{[t+1]}|$.
Step 4:
  Step 4.1: *If* $d < \varepsilon$ *or* $t > M$, *then go to Step* 5.
  Step 4.2: *Otherwise*, $y_{ij}^{[t]} \leftarrow y_{ij}^{[t+1]}$, $t \leftarrow t + 1$, *and then go to Step* 2.
Step 5: *Output* $\mathbf{y}_i^{[t+1]}$, $i = 1, \ldots, n$.

3 Experiment

In this report, we use 2-D clustering dataset s_set and a2_set from [4]. These two datasets are like:

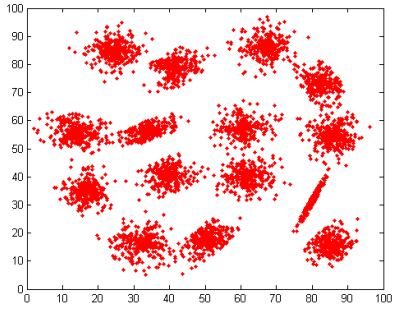

Figure 2. s_set (5000*2, rescalar=1/10000)

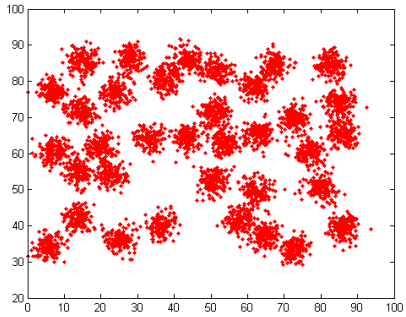

Figure 3. a2_set (5250*2, rescalar=1/700)

3.1 Experiment on G-algorithm

We first implement the G-algorithm. Because it is very sensitive to the initialization of the G which is similar to the notion of step size, for the two dataset we choose the G as 6.67/2 according to the definition and decayG = 0.0001. Because we have two tables contains the original dataset of D^T and a new moving dataset D^{T+1}, the order of the data points being processed is no worry. One thing embarrassing need to note is that because of the time complexity, instead of using a defined ε as the convergence condition, we simply use *MaxIteration* to control the shrinking process.

The results of both s_set and a2_set with MaxIteration = 1000 are:

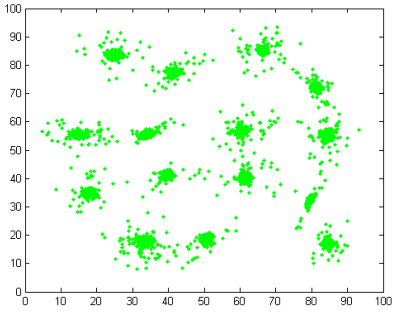

Figure 4. After G shrinking(s_set)

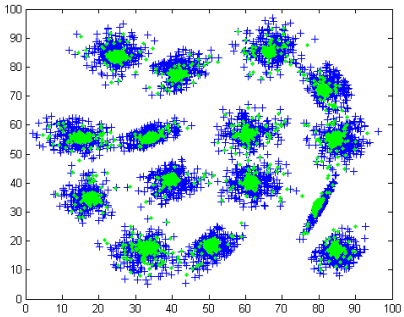

Figure 5. Comparison with original s_set

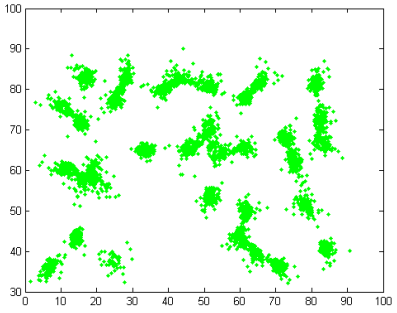

Figure 6. After G shrinking(a2_set)

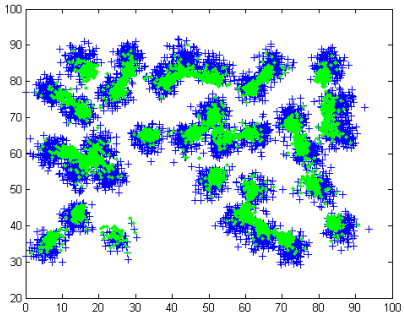

Figure 7. Comparison with original a2_set

In the figures, we observe that G-algorithm works well in s_set, as all clusters has a condensed center. However in a2_set, because some clusters are too close to each other which leads to cluster merging. Moreover, we find that most condense center are slightly shifted from their cluster center to the global center. In some sense, points from other clusters may also have some influence for those within same cluster, especially for the clusters which locate relatively outside. It is reasonable because the points on the outside will gradually move to the center while the center cluster will balance the force from opposite directions according to the law. This also tells us that the convergence condition is rather significant.

3.2 Experiment on LS algorithm

In implementation of LS algorithm, we use Matlab commands like *'knnsearch'* and *'median'* which makes the code is as clean as the result figures. In this method, we make a little change in the original implementation of the CLUES [3] by decaying the K as the iteration grows. Because points are in high density in later iterations, it is not necessary for finding as much as K neighbors. We set the $K = 10$ in the experiment and after 4 iterations the final figure is shown in Figure 8~11.

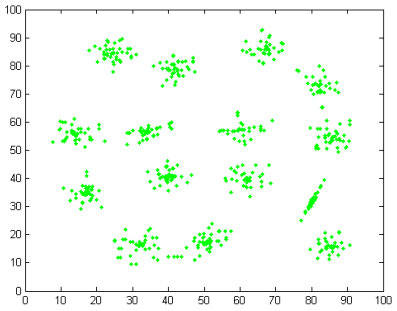

Figure 8. After LS shrinking(s_set)

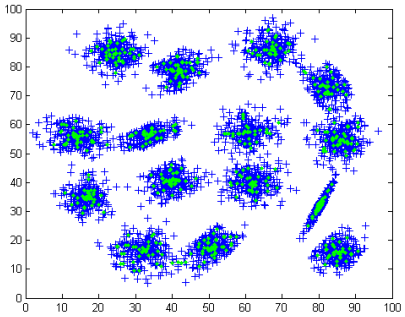

Figure 9. Comparison with original s_set

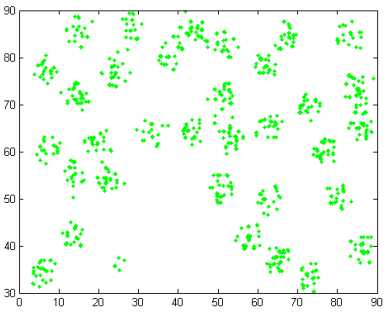

Figure 10. After LS shrinking(a2_set)

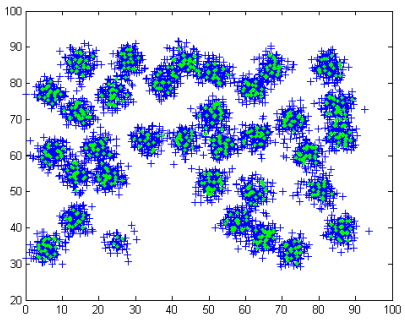

Figure 11. Comparison with original a2_set

Because we use the median of the K-nearest neighbors to update the points, the noise is gathered in the clusters. Also the points seem not that dense in comparison with those in previous graph but actually they are. We can confirm that by the graph given:

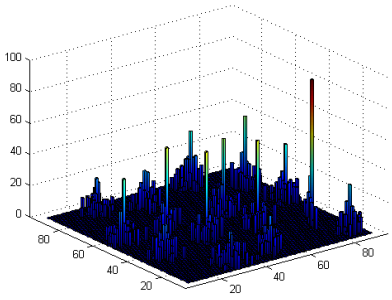

Figure 12. 3-d histogram of s_set after shrinking

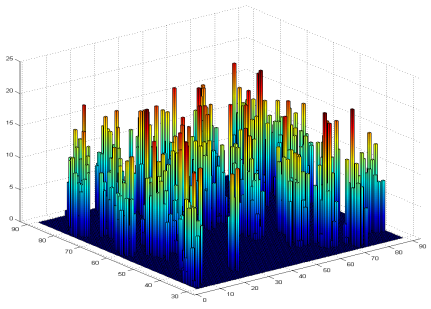

Figure 13. 3-d histogram of a2_set after shrinking

From the 3-d histograms of the dataset after shrinkage, we observe that the data set are actually very dense. Some extreme points in figure 12 contains nearly 80 points in a single bin. This is because the some data points located at the middle could be used as median of KNN of many other points, leading to all those points sharing the same value.

4 Conclusion

Both G-algorithm and LS-algorithm aim at shrinking the data points and gain a better view of the original dataset. However, G-algorithm is very sensitive to the constant G and needs the extra density information about the dataset to choose a reasonable convergence threshold. LS-algorithm has a good performance of sharpening the dataset and it is also robust because the using median instead of mean.

When handling datasets that have very near clusters, G-algorithm is more likely to merge the clusters because of the clusters will attract to each other, while this is not the case for the LS algorithm.

For future work, there are also demands on shrinking on multidimensional data. Moreover, to improve the efficiency of the shrinking methods by exploring parallel algorithms for the shrinking methods is also a great and open topic.

References

[1]*http://en.wikipedia.org/wiki/Cluster_analysis*

[2]Jonatan Gomez* Dipankar Dasguptat Olfa Nasraoui[C] *A New Gravitational Clustering Algorithm* Proceeedings of the Third SIAM International Conference on Data Mining. SIAM, 2003, 112: 83.

[3]Wang X, Qiu W, Zamar R H. *CLUES: A non-parametric clustering method based on local shrinking*. [J]Computational Statistics & Data Analysis, 2007, 52(1): 286-298.

[4]*http://cs.joensuu.fi/sipu/datasets/*

[5]Wang, J. H. and Rau, J. D. (2001), *"VQ-Agglomeration: A Novel Approach to Clustering,"* IEE Proceedings-Vision, Image and Signal Processing, 148, 36–44.

[6] Shi Y, Song Y, Zhang A. *A shrinking-based approach for multi-dimensional data analysis*[C] //VLDB. 2003: 440-451.

[7] Comaniciu D, Meer P. Mean shift: *A robust approach toward feature space analysis*[J]. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 2002, 24(5): 603-619.