Local Search Methods for k-Means with Outliers

Jesus Zarate

January 15, 2018

1 Introduction

Clustering using k-means is a widely studied data mining problem, and it is well known that finding an optimal solution is NP-hard. The k-means algorithms is known to be as one of the top ten algorithms used in data mining, other than having the guarantee of being a local minimum, it is simply a heuristic and has no guarantees on the quality of the solution. Although k-means is well studied, real world algorithms don't perform very well. In practice k-means peforms poorly due to k-means assuming that all of the points can be naturally partitioned into k distinct clusters. Also, real world data tends to contain noise and k-means is extremely sensitive to it, and it can drastically change the quality of the clusters.

In our project we took on the task of dealing with noisy data, or a way to deal with outliers in the data. The way the paper that we researched dealt with noisy data was to still perform the normal clustering algorithm, however, the algorithm also discards a small set of data points from the input data. The points removed from the input data will be referred to as outliers, and will not be included in part of the clustering solution. This will allow the clustering algorithm to focus on partitioning the rest of the data, which will be virtually noise-free.

2 Algorithm

Although the paper gives a very discriptive algorithm, it is not very easy to read, and we would like to give alternate pseudocode and try to make it more readable and provide more comments.

The algorithm behave just like good old vanilla k-means where it tries to minimize the sum of squared distances between every point and its nearest center, in euclidean space. Except that as the algorithm tries to find the best centers by swaping, this algorithm also removes the points farthest away from swapped centers, and calls them outliers. If the algorithm improves by a significant amount then these outliers will be discarded from the input data, and the algorthim will continue with this process until swaping centes and removing additional outliers does not improve the cost.

The first part of the algorithm is the local search for k-means without outliers

```
def LS(Input, C, k):
    alpha = infinity

    while alpha * (1 - epsilon/k) > Cost(C, Input):
        alpha = cost(C, Input)
        tempC = C # A temporary improved set of centers


        for u in range(Input):
            for v in range(C):

                swapC = swap(u, v, C) # Swap the values of u and v
                swapCost = cost(swapC, Input) # Calculate the cost
                                              # of the swapped center

                # If this is the most improved swap found so far
                if swapCost < cost(tempC, Input):
                    tempC = swapC

        # Update solution to the best swap found so far
        C = tempC
    return C # as the k centers
```

The second part of the algorithm is the local search for k-means with outliers
This algorithm is divided into 3 parts.

0. Perform local search with no outliers
1. Computing the cost after discarding z amount of additional outliers
2. Finally for each center and non-center, perform a swap and then discard additional outliers

```
def LS-Outlier(Input, C, k):
    alpha = infinity
    C = kCenters(Input) # An arbitrary set of k points from Input
    Z = outliers(C) # Compute outliers by getting points
                    # farthest away from the centers

    while alpha * (1 - epsilon/k) > Cost(C, Input):
        alpha = cost(C, Z)

        # Step 0
        inputNoOutliers = removeOutliers(Input, Z)
        C = LS(inputNoOutliers, C, k)

        tempC = C # A temporary improved set of centers
```

```
tempZ = Z # A temporary improved set of outliers

# Step 1
additionalOutliers = Z + outliers(C, Z)
costAdditionaOutliers = cost(C, additionalOutliers)

if cost(C, Z)(1 - epsilon/k) > costAdditionaOutliers
    tempZ = additionalOutliers
```

3 Algorithm Modifications

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed euismod quis odio sed malesuada. Praesent a massa ut magna fringilla imperdiet ut ac dolor. Nulla tempus arcu a elit mollis, id convallis dui lacinia. Nulla interdum lorem at orci semper, at faucibus mi vehicula. Morbi pulvinar tempus sem, nec accumsan lectus varius et. Proin eu cursus est. Nullam eu placerat eros, vel viverra neque.

4 Experiments

Quisque aliquam accumsan orci nec auctor. Suspendisse elit justo, iaculis ut orci porttitor, egestas pharetra urna. Fusce hendrerit nulla id ligula faucibus hendrerit. Aliquam libero purus, consectetur vel risus vitae, faucibus rhoncus odio. Sed fringilla elit eu dolor pharetra, non viverra nisi accumsan. Integer gravida metus quam, nec laoreet nunc tristique nec. Nullam in ex vel dolor pharetra scelerisque. Etiam vestibulum consequat tortor, id aliquet nunc ultricies a. Donec consequat consectetur sagittis. Nunc neque mi, sagittis eget tempor ut, posuere nec felis. Nam mollis lectus libero, vel laoreet neque dignissim a. Etiam laoreet velit nec nunc egestas, rutrum ullamcorper diam pellentesque. Nam placerat at nisi quis fermentum.

4.1 Synthetic Data

Nullam libero lectus, rutrum eget velit eu, posuere rutrum leo. Suspendisse cursus orci nisl, eget vestibulum risus mattis quis. Cras dapibus, tellus quis accumsan accumsan, ligula ligula eleifend diam, ut lobortis neque mi cursus nisl. Praesent lobortis ipsum tortor, ut aliquam tellus blandit nec. Nullam non luctus mauris. Quisque consectetur ultricies nunc, posuere commodo nisi condimentum dictum. Suspendisse potenti. Curabitur sed imperdiet arcu. Nulla aliquet nisi risus. Sed et aliquam elit, vel aliquam enim. Curabitur posuere porttitor tellus eget ornare. Morbi iaculis ultricies nisl, eget ullamcorper nulla facilisis eget. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. In auctor urna in congue egestas. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Sed posuere lectus in orci lacinia viverra.

4.2 Real Data

Nam sagittis metus sit amet accumsan efficitur. Donec molestie viverra ipsum, vitae hendrerit nisi tempor sit amet. Praesent a pulvinar lorem. Integer consequat vulputate turpis, a luctus velit posuere vel. Curabitur in placerat quam. Quisque bibendum leo at massa sollicitudin dignissim. Nullam in semper risus.

5 Results

Cras vitae tempor justo, in hendrerit erat. Quisque imperdiet dignissim elit, vitae efficitur velit tempus eget. Duis est ex, dapibus ac diam in, eleifend ornare urna. Phasellus vitae sem non felis ultrices sollicitudin. Sed lorem nisl, pharetra vel efficitur non, sollicitudin ut nisi. Sed eu metus efficitur, iaculis ligula vitae, tincidunt dui. Maecenas auctor tortor sed tempor volutpat. Proin scelerisque, erat eget dapibus placerat, lorem felis fringilla arcu, id eleifend felis ex vitae lacus. In lectus risus, finibus et lacus ut, mattis sagittis dui.