Lecture 14 - 2: Limitations of the K Means Algorithm

Tabaré Pérez

May 6, 2020

Agenda

- K-Means algorithm.
- K-Medioids.
- Determining the number of clusters.

So today, we will continue the conversation about unsupervised learning. And we will continue talking about clustering. And I will start my lecture by briefly summarizing what we've seen last time, which covered the discussion about K-means algorithm. And specifically, we would see what are the limitations of this algorithm and think how we can improve it in some ways. So this will be our introduction to the algorithm that I will introduce today, which is called **K-MEDOIDS**.

And we will talk about this algorithm and what kind of new freedom this algorithm allows us to achieve, which K-means didn't let us do.

And then I would answer the question that many people are always thinking about clustering: How do we determine this K, the number of clusters?

So this would be our next question, determining the number of clusters. So I want to be honest upfront. I would not give you a magic formula on how to determine K. But I will give you some consideration how we can be thinking about deciding about K and what different reasons you may want to take into account when you are selecting K.

So let's first go back and think about general clustering problem.

So what we've said last time is that whenever we are thinking about clustering, we start with some similarity or distance measure, and then we are defining the cost of the partitioning.

Once we settle on the cost of the partitioning, we will need to have an algorithm which optimizes this cost, which find the best partition according to this cost.

And the cost that we introduced last time:

$$\text{cost}(C_1 \dots C_K, z^{(1)}, \dots, z^{(K)}) = \sum_{j=1}^{K} \sum_{C \in C_j} \left\| x^{(i)} - z^{(j)} \right\|^2 \tag{1}$$

actually looked at both partitioning themselves. So from C_1 and to C_K, but it also introduced this representatives, $z^{(1)}$ to $z^{(K)}$, and specifically the cost that we had the following: It went through all the clusters from 1 to K, and then within each cluster, it measured the distance between the representative and the particular point and it computed the Euclidean squared distance between the two.

So the algorithm that we've seen last time, the K-means algorithm, was working to find the best partitioning, which optimizes this costs. And let me remind you the structure of this algorithm, because as I said you earlier, the K-medoids algorithm that we are going to be using is actually a variant in some ways of K-means.

1. Ramdomly initialize $z^{(1)} \dots z^{(K)}$
2. Iterate unitl no change in cost:
   - 2a. for $i = 1 \dots n$:
     $C_j = \{i|$ s.t. $z^{(j)}$ is closest to $x^{(i)}\}$
   - 2b. for $j = 1 \dots K$:
     $z^{(j)} = \frac{\sum_{i \in C_j} x^{(i)}}{|C_j|}$

So we started first by randomly randomly initializing $z^{(1)} \dots z^{(K)}$. So we already picked up the z somehow randomly.

So after the initialization, the next step for us is to continue iterations.

At each point, we are going to come up with the new partitions, and the new representatives compute the cost, and if there is no change in cost, we can just stop. Otherwise, we continue iterations.

So we're going to right now iterate until no change in cost.

And for the first step of the algorithm, which we'll call it **2a**, we will go for all the points in our set, so i from 1 to n.

And for each one of the points, we will find the cluster, which whose centroid is closest to this point.

So in other words, we would say that C_j, the cluster would contain all of the points i, such that $z^{(j)}$ is closest to $x^{(i)}$.

So now for the fixed representatives, we re-assign all the points grouping them closer to the corresponding centers.

The next step for us is to do reassignment of the reference points.

So we will stand for j from 1 to K, and we will make $z^{(j)}$ to be the centroid of the corresponding partitioning, which means we go and sum all the points of the partitioning and divide it by this size of the partitioning.

So what we've done here, again, we summed over all the points and divided by the size of the partitioning.

So it's like an average of all the points.

So now let's look at this algorithm and think about limitations of the algorithm.

The first significant limitation of this algorithm is the fact that the z's are actually not guaranteed to be the members of the original set of points x.

So in this particular case, we may get another point.

In some application, it's OK. In other application, it's really a concern if we're thinking about looking at Google News, and we created a representative of the cluster of the story, which doesn't correspond to any story, we actually have nothing to show. This will be really problematic issue for many visualization applications.

So we would ideally want to have an algorithm wich actually gives us the freedom, if you wish so to find the z's, to find a representative which is one of the original points.

The second limitation of this algorithm has to do with the fact, again, with the step **2b**.

In order for us to say that the representative is actually a centroid of the points, we had to utilize the squared Euclidean distance.

And as you remember from the exercise that you did, that we can make the exact computation and, if this is our cost function and we want to minimize it, we need to compute the derivative and make it equal to 0. It will work this way, only if the original function that you are considering is squared Euclidean distance.

Now, in many applications of clustering, you want to work with other functions, not necessarily with square Euclidean distance. You may want to use $L1$ (Manhattan distance or Taxicab) or something else. This algorithm will not cut, so you would need to come up with some other way to find the clustering algorithm, which works with your metrics.

So with these two limitation in mind, again, making a representative part of the original points and ability to work with any distance metrics, we are moving towards the new algorithm that we need to consider, **K-MEDOIDS**.

So we've done with summarizing K-means, and we can start now talking about **K-MEDOIDS**, which will resolve two of those constraints.