Partitioning Clustering

Mads Møller

LinkedIn: https://www.linkedin.com/in/madsmoeller1/

Mail: mads.moeller@outlook.com

This paper is the seventh in the series about machine learning algorithms. Partitioning Clustering methods is used for *clustering* problems. Partitioning Clustering includes different types of unsupervised machine learning algorithms, meaning that our data do not need to be labelled. So you do not need to know your target classes in order to use Partitioning Clustering methods. The most popular partition-based clustering algorithm is called **K-Means**. However, there does exist other partition-based algorithms such as **K-Medians**, etc.

1 Intuition

In many ways clustering can relate to the task of nearest-neighbor algorithms. We would like to group observations that are similar given a dissimilarity. Each observation is in a group and different groups do not overlap. A group is often in unsupervised learning referred to as a **cluster**. Within each group we have *inter homogeneity*, meaning that observations within a cluster should be similar. We call the collection of clusters a *clustering*. Within clustering we have *inter heterogeneity*, meaning that clusters should be dissimilar to other clusters.

Often clustering methods are used for *customer segmentation*, *grouping of products*, *recommender systems* (segmentation of customers and products). So indeed clustering methods are useful in the real-world as well.

2 Partitioning Clustering

The last machine learning paper was about *Hierarchical Clustering*. In Hierarchical Clustering the algorithm made a hierarchy of clusters where a threshold set for the cluster dendrogram decided how many clusters the data would be split into. In partitioning clustering the amount of clusters is set beforehand.

Partitioning Clustering is at its core a combinatorial problem. The partitioning of a dataset of n observations into K clusters is a computational expensive problem because it is combinatorial (there are a lot of possible combinations). If we had to calculate the number of possible partitions it could be done with the following formula:

$$\sum_{l=1}^{K}(-1)^{K-l}\frac{1}{(K-l)!\cdot l!}l^n$$

If we take an example of a dataset with 20 observations ($n = 20$) and we want to divide the observations into 4 clusters ($K = 4$) we would have more than 4×10^{10} possible partitions. However this would of course not work in practice as it would take too long time to compute. Therefore, in order to deal with the combinatorial character of the problem we will find the partition using iterative procedures that alleviate the computational complexity. When we use partitioning clustering we also have a measurement of dissimilarity δ which will be a distance, d, to measure how similar clusters and observations are at each other.

In partitioning clustering we use the concept of **prototype**. Each cluster will have a prototype. A prototype is a representative of the cluster. We define p_l as the prototype of the cluster l. The prototypes are used for measuring the intra-homogeneity in a cluster. We will measure the distance from the points in the cluster to its prototype, to have an estimate of the intra.homogeneity of the cluster:

$$\sum_{i\in C_l} d(x_i, p_l)$$

So we measure the intra-homogeneity by finding the sum of the distances between all observations in a cluster and its prototype p_l. The objective of a partitioning clustering method must be to minimize the within cluster distance to its prototype, across all clusters:

$$\text{minimize} \quad \sum_{l=1}^{K}\sum_{i\in C_l} d(x_i, p_l) \tag{1}$$

This minimization problem exactly aims at improving the total intra-homogeneity.

2.1 Iterative partitioning algorithms

First the clustering algorithm will choose K *initial prototypes*, $p_1, p_2, \ldots, p_K$. The prototypes will be randomly selected when the algorithm is initialized. You can imagine prototypes as being imaginary observations (prototypes) in our solution space. When the prototypes are set the algorithm clusters around the prototypes, i.e., each observation is assigned to the **closest** prototype. In math what the algorithm does at this point in time:

$$i \in C_l \quad \text{if} \quad d(x_i, p_l) = \min_{l'=1,\ldots,K} d(x_i, p_{l'}) \tag{2}$$

In other words the cluster an observation will be assigned will be the prototype it is closest to. The next step of the algorithm is to check if the clusters has changed. If the clusters has not changed the algorithm should stop. If the clusters has changed the prototypes will be (re-)calculated and equation 2 will be repeated until convergence (the clusters has not changed). How exactly the prototypes will be re-calculated is where we split partitioning clustering methods into different specified algorithms such as **K-Means**, **K-Medians** etc. The crucial elements in the different partitioning-based clustering methods are:

- The K initial prototypes.
- The definition of the prototypes.
- The distance used to define the similarity.

There is a whole family of Partitioning-based Clustering methods defined from their choices of the three bullets above. We will now dive into some of the different partitioning-based methods.

2.2 K-Means

In the K-means algorithm the K initial prototypes are randomly selected. The definition of the prototypes are the vector of means of the cluster (therefore the name K-means). We can write the definition of prototypes in K-means as:

$$p_l = \frac{1}{\text{cardinality}(C_l)} \sum_{i \in C_l} x_i$$

The cardinality is just the number of observations within a cluster. So the prototypes are defined as the mean vector based on the observations within a cluster. The distance measurement used in the K-means algorithm are the *Squared Euclidean Distance*:

$$d(x_i, p_l) = (x_{i1} - p_{l1})^2 + (x_{i2} - p_{l2})^2 + \ldots + (x_{ik} - p_{lk})^2$$

Therefore, from equation 1 the goal of K-means is to minimize the *total within sum of squares*:

$$\text{minimize} \sum_{l=1}^{K} \sum_{i \in C_l} (x_{i1} - p_{l1})^2 + (x_{i2} - p_{l2})^2 + \ldots + (x_{ik} - p_{lk})^2$$

If you want to get some more knowledge about the euclidean norm measurement the paper about K-Nearest-neighbors include more information on the topic. A thing to aware of it that K-means can only be used for continuous data because the euclidean norm only works on continuous data.

2.3 K-Medoids

In the K-medoids algorithm the K initial prototypes are randomly selected. The K-medoids algorithm can use any dissimilarity δ to measure how similar observations are. The prototype of cluster C_l defined as:

$$\sum_{i \in C_l} \delta(x_i, p) = \min_{x \in C_l} \sum_{i \in C_l} \delta(x_i, x)$$

From equation 2.3 it might be a little hard to understand the prototype in K-medoids. However, it is fairly simple, the prototype of a cluster will be calculated as the individual in the cluster that minimizes the sum of dissimilarities to the other individuals in the cluster. The nice thing about K-medoids is that it is suitable when categorical variables are present. This is because it works with any dissimilarity measurement δ.