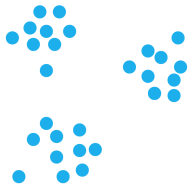
EE 16B Lecture Notes

k-means Clustering

February 10, 2016

This lecture is a continuation on the topic of clustering, specifically k-means clustering.

As an example, through some process, we have some cloud of points.



By some means, we know that there are, for example, 3 clusters of points (thus k=3 for our example). This data could be from neural electrophysiology measurements. In such a case, an individual point would represent a time-series of sampling, we remove their means and perform SVD on the data. Then we look for the 2 dimensions of greatest variation (because we can visually plot in 2-dimensions, although this can be done with higher-dimensions as well) and plot the data along those two axes. Thus it's plotted location would represent the shape of the signal. Two points that are in the same cluster means that for those two signals, the shape is similar.

Our job is to find the centers (the k-means) of the clusters of points. We assume the clusters are "round" (circular/spherical/not limited to 2-D or 3-D). In the illustration above, we can visually see that the three clusters are distinct, however, in some cases, clusters will blur together. As humans, we are very good at pattern recognition, however, computers are not.

Thus we address the issue of how we train computers to find the centers of the clusters in unsupervised computation.
For such a problem we first look at our input and our output:

Input:

a collection of data points ⟵ vectors (vector $\vec{x}$ in Eq. 1)

k ⟵ desired number of clusters

Output:

k different cluster "centers"

The objective of our algorithm is to minimize the distortion, D:

$$D = \sum_{\vec{x} \in data} \parallel \vec{x} \ - \vec{m}_{c(\vec{x})} \parallel^2 \tag{1}$$

In which $\vec{m}_{c(\vec{x})}$ is the proposed cluster center, and it is subtracted from every data point. The function $c(\vec{x})$ returns the cluster index point for specified $\vec{x}$, which is basically a label for which cluster we think it belongs to (in the next paragraph, it is basically S_i). Essentially, how far is each data point from it's cluster center. Thus, this procedure looks for the minimum squared error, similar to the idea of least squares.

We use k-means, an iterative algorithm, to solve our problem: In this algorithm, we will track $\vec{m}_i$ and collections S_i (essentially labels of all the data points currently believed to be in cluster i). Think of each S_i like a labeled box filled with all the points we proclaim to be in one cluster.

0. Initialize each $\vec{m}_i$ somehow
1. Assign data points to clusters via Eq 2, shown below.
2. Pick updated $\vec{m}_i$'s given S, described in Eq 3, shown below.

If no $\vec{m}_i$ changed, STOP. Otherwise, go back to step 1 and repeat.

Equations used in the algorithm:

$$S_i = \{\vec{x} \in \text{data} \mid i = \underset{j}{\text{argmin}} \parallel \vec{m}_j - \vec{x} \parallel\} \qquad j \in [0, \ldots, k-1] \tag{2}$$

$$\text{set } \vec{m}_i = \frac{1}{\mid S_i \mid} \sum_{\vec{x} \in S_i} \vec{x} \tag{3}$$

Proof of Minimization

Proof that Step 2 (Eq. 3) of the algorithm does actually minimize the sum of squared error:

$$\min_{\vec{m}_i} \sum_{\vec{x} \in S_i} \| \vec{m}_i - \vec{x} \|^2 \tag{4}$$

$$= \min_{\vec{m}_i} \sum_{\vec{x} \in S_i} (\vec{m}_i - \vec{x})^T (\vec{m}_i - \vec{x}) \tag{5}$$

$$= \min_{\vec{m}_i} \sum_{\vec{x} \in S_i} \vec{m}_i^T \vec{m}_i - 2\vec{x}^T \vec{m}_i + \vec{x}^T \vec{x} \tag{6}$$

Since the $\vec{x}^T \vec{x}$ term doesn't have any effect when minimizing $\vec{m}$, we can drop the last term from Eq. 6. To mathematically find the minimum, we look at the derivative.

$$\frac{d}{d\vec{m}_i} \left[\mid S_i \mid \sum_{t=0}^{n-1} m_i[t]^2 - 2 \sum_{\vec{x} \in S_i} \sum_{t=0}^{n-1} m_i[t] x[t] \right] = 0 \tag{7}$$

Going through some minor algebra will yield a minimum, $\hat{m}$:

$$\hat{m}[t] = \frac{1}{\mid S_i \mid} \sum_{x \in S_i} x[t] \tag{8}$$

Although we didn't give any specifics for how to do the initialization, the initialization can be critical. As an example of how poor initialization can go wrong, see the below illustration:

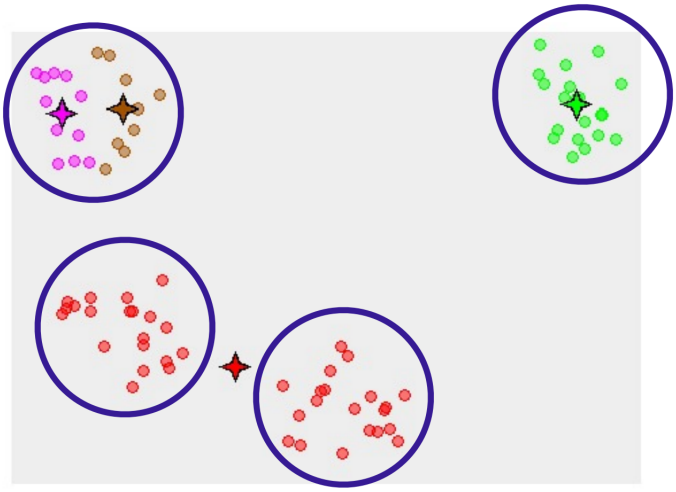


Illustration sourced from "Clustering and the K-means algorithm" MIT 18.304 Seminar Talk I by Yihui Saw from March 6, 2013.