CS 181 Spring 2016 Section 6 Notes (Clustering)

1 Motivation

We now move onto **unsupervised learning**, where the objective is to learn the structure of unlabeled data. In other words, we are looking for groups, or **clusters** among the data. Clustering algorithms are useful not only for finding groups in data, but also to extract features of the data that summarize the most important information about the data in a compressed way.
For most clustering algorithms, we need some kind of a metric that we can use to specify the notion of "distance" between the data points. If, for example, the points $\mathbf{x}$ and $\mathbf{x'}$ live in some Euclidean space $\mathbb{R}^N$, then the natural choice of such metric is the l_2 distance:

$$||\mathbf{x} - \mathbf{x'}||_2 = \sqrt{\sum_{n=1}^{N} (x_n - x'_n)^2}$$

Now that the metric is well-defined, the next thing we need to do is to decide how many groups you want. Sometimes you know the ideal number of groups in advance (*e.g.* clustering alphabet). Other times, you need to decide if you'd like a more compressed representation with more information loss by having the number of groups small, or a less compressed representation with less information loss by having the number of groups large.
Suppose our data set is $\{\mathbf{x}_n\}_{n=1}^N$, then our objective is to find the ideal assignment of the data set to the clusters, by assigning to each of the N data points, a binary **responsibility vector** $\mathbf{r}_n$, which is all zeros except one component, which corresponds to the assigned cluster.

2 K-Means Algorithm

The idea is to represent each cluster by the point in data space that is the average of the data assigned to it. The K-Means Algorithm loops over the mean vectors $\{\mu_k\}_{k=1}^K$ for some choice of cluster number K and changes them to be the mean of the cluster that it currently belongs to. After that, we iterate through the data points and update the responsibility vectors according to their closest means. We repeat this until convergence: until none of the responsibility vectors change.
(*The pseudocode for Lloyd's Algorithm should be reviewed in section.*)

2.1 Derivation

We begin by defining a loss function:

$$J(\{\mathbf{r}_n\}_{n=1}^{N}, \{\mu_k\}_{k=1}^{K}) = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} ||\mathbf{x}_n - \mu_k||_2^2$$

and the K-Means Algorithm minimizes this via coordinate descent.
We can minimize this by first choosing r_n such that:

$$r_{nk} = \begin{cases} 1 & \text{if } k = argmin_{k'} ||\mathbf{x}_n - \mu_{k'}||_2^2 \\ 0 & \text{otherwise} \end{cases}$$

Having fixed everything else, we see that the loss function is:

$$\begin{aligned} J(\mu_k) &= \sum_{n=1}^{N} r_{nk} ||\mathbf{x}_n - \mu_k||_2^2 \\ &= \sum_{n=1}^{N} r_{nk} (\mathbf{x}_n - \mu_k)^T (\mathbf{x}_n - \mu_k) \end{aligned}$$

Taking the derivative and setting it to zero,

$$\begin{aligned} \nabla_{\mu_k} J(\mu_k) &= -2 \sum_{n=1}^{N} r_{nk} (\mathbf{x}_n - \mu_k) = 0 \\ \mu_k &= \frac{\sum_{n=1}^{N} r_{nk} \mathbf{x}_n}{\sum_{n=1}^{N} r_{nk}} \end{aligned}$$

2.2 Number of Clusters

Our main approach is to use the idea of the **gap statistic**. We can define the within-cluster dispersion:

$$D_k = \sum_{n=1}^{N} \sum_{n'=1}^{N} r_{nk} r_{n'k} ||\mathbf{x}_n - \mathbf{x'}_n||^2$$

Then the dispersion for the clustering is the normalized sum of the within-cluster dispersion over all the clusters:

$$W_k = \sum_{k=1}^{K} \frac{1}{2 \sum_{n=1}^{N} r_{nk}} \sum_{n=1}^{N} \sum_{n'=1}^{N} r_{nk} r_{n'k} ||\mathbf{x}_n - \mathbf{x'}_n||^2$$

To compute the gap statistic, we use a null distribution for the data, from which we generate reference data. Then, the gap statistic can be computed:

$$Gap_N(K) = \mathbb{E}_{null}[\log W_K] - \log W_K$$

2.3 Notes

Lloyd's algorithm finds a local optimal solution and finding the global optimal turns out to be NP-hard. A common strategy, therefore, is to use random restarts. More recently, algorithm called **K-Means++** has enjoyed popular usage as an alternative to random initialization. The basic idea is to assign probability to each data point proportional to the squared of the distance to the nearest center of being the next center.
Also note that sometimes it is necessary to **standardize** the data to account for unsatisfying result due to dimension mismatch.
Lastly, when for the metric we are using for the given data set, a "mean" does not make sense, we might instead use a **K-Medoids Algorithm**. This algorithm requires the cluster centers to be a data point in the data set.

3 Hierarchical Agglomerative Clustering

Hierarchical clustering constructs a tree over the data, where the leaves are individual data items, while the root is a single cluster that contains all of the data. When drawing the dendrogram, for the clustering to be valid, the distances between the two groups being merged should be monotonically increasing.
(*The pseudocode for HAC should be reviewed in section.*)
The main decision is what the distance criterion should be between groups.

3.1 The Single-Linkage Criterion

The idea is to merge groups based on the shortest distance over all possible pairs:

$$DIST(\{x_n\}_{n=1}^N, \{y_m\}_{m=1}^M) = min_{n,m}||x_n - y_m||$$

this produces the minimum spanning tree for the data.

3.2 The Complete-Linkage Criterion

The idea is to merge groups based on the largest distance over all possible pairs:

$$DIST(\{x_n\}_{n=1}^N, \{y_m\}_{m=1}^M) = max_{n,m}||x_n - y_m||$$

3.3 The Average-Linkage Criterion

The idea is to average over all possible pairs between the groups:

$$DIST(\{x_n\}_{n=1}^N, \{y_m\}_{m=1}^M) = \frac{1}{NM}\sum_{n=1}^{N}\sum_{m=1}^{M}||x_n - y_m||$$

3.4 The Centroid Criterion

The idea is to look at the difference between the groups' centroids:

$$DIST(\{x_n\}_{n=1}^{N}, \{y_m\}_{m=1}^{M}) = || \left(\frac{1}{N} \sum_{n=1}^{N} x_n \right) - \left(\frac{1}{M} \sum_{m=1}^{M} y_m \right) ||$$

3.5 Divisive Clustering

While HAC is a bottom-up procedure, divising clustering is a top-down procedure, starting with a single group with all the data in it and then dividing and subdividing it into smaller groups using other clustering algorithms like K-Means or K-Medoids.

4 Practice Problems

1. **Convergence of K-Means (Bishop 9.1)**

Consider Lloyd's algorithm for finding a K-Means clustering of N data, i.e., minimizing the "distortion measure" objective function

$$J(\{\mathbf{r}_n\}_{n=1}^{N}, \{\mu_k\}_{k=1}^{K}) = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} ||\mathbf{x}_n - \mu_k||_2^2.$$

Show that as a consequence of there being a finite number of possible assignments for the set of responsibilities $r_{n,k}$, and that for each such assignment there is a unique optimum for the means $\{\mu_k\}_{k=1}^{K}$, the K-Means algorithm must converge after a finite number of iterations.

2. **K-Means++**

   $\boxed{\text{Assuming that } N > K\text{, can the K-Means++ algorithm choose the same datum twice to become a cluster center? Why or why not?}}$

3. **K-means and HAC**

   What is the difference between K-Means and HAC? Give answer in terms of both the number of clusters and determinism.

4. **Single-Linkage and Complete-Linkage Criterion**

Use the initial setting:

$$\{1\}\{2\}\{4\}\{5\}\{9\}\{11\}\{16\}\{17\}$$

(a) Using the Single-Linkage Criterion for the HAC Algorithm, what is the clustering sequence until there are two clusters remaining?

(b) Using the Complete-Linkage Criterion, what is the clustering sequence until there are two clusters remaining?

5. **Clustering Complexity**

   What would be the big-O complexity of an HAC? In terms of efficiency and determinism, when would HAC be a better method than divisive clustering?