Unsupervised Learning

MInDS @ Mines

In this lecture, we discuss some clustering methods and how they differ. We will cover hierarchical clustering, specifically agglomerative clustering, and nonnegative matrix factorization as an approach to clustering. We will also go over manifold vs Euclidean space and cover an example of clustering in the manifold space, namely spectral clustering.

We discussed clustering as the grouping of data into similar portions tht are different from others where each portion is called a cluster. We now examine some methods for determining the clusters in our data.

Hierarchical Clustering

Hierarchical clustering is an approach to clustering that creates a hierarchy of data samples to determine the clusters that each sample belongs to. There are two main types of hierarchical clustering; agglomerative, and divisive. Agglomerative clustering is a set of hierarchical clustering methods that initialize each sample as their own cluster then agglomerate them into larger groups until they form the desired number of clusters. Divisive clustering is a set of hierarchical clustering methods that initialize all samples as one cluster then proceed to divide the cluster into smaller clusters.

Agglomerative Clustering

In a clustering problem with n samples of data and a desired clustering of k clusters, we begin by defining n clusters, where every point is a cluster. In every iteration, we then merge the two closest clusters together until we reach just 1 cluster.[1] We can then construct a dendrogram and start at the root then move down to the child clusters until we reach our desired k clusters.

Now there are two hyperparameters to this method, mainly related to determining the two closest clusters. First, we select a distance metric between any two points, this can be the Euclidean distance (ℓ_2-norm), Manhattan distance (ℓ_1-norm) or any other distance metric. The second hyperparameter is how we calculate the distance from one cluster to the next. This is called the *linkage criteria*. We will merge the two clusters that have the minimal metric for the linkage criteria.

Three straightforward linkage criteria are the maximum distance, minimum distance, and average distance. The maximum distance, also called *complete linkage*, is the farthest distance between any two points in the clusters. The minimum distance, also called *single linkage*, is the closest distance between any two points in the clusters. The average distance, also called the Unweighted Pair Group Method with Arithmetic Mean (UPGMA), is the av-

[1] This approach allows us to calculate the clusters for all possible values of k at once. We can also stop the dendrogram at k and determine the clusters at the k level.

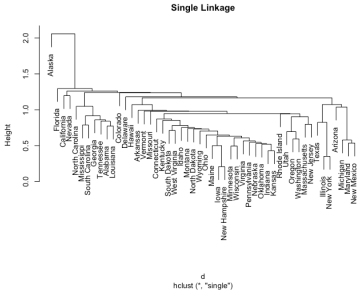

Figure 1: An example dendogram using single linkage; $\min(\text{dist}(c_i, c_j))$.

erage of the distances between every pair of points from the two clusters. Another linkage criterion, *Ward's method*, utilizes the variance change in each cluster. Ward's method's criterion calculates the increase in variance of merging every two clusters. The method selects the cluster with the minimal criterion increase.

Algorithm 1: Agglomerative Clustering Solution Algorithm

Input: Features from data

Output: k cluster labeling for each point

Initialize n clusters where every point is a cluster

while *cluster count reached greater than* k **do**

  Determine the two clusters closest to each other based on distance metric and linkage criteria

  Merge the two clusters into one

end

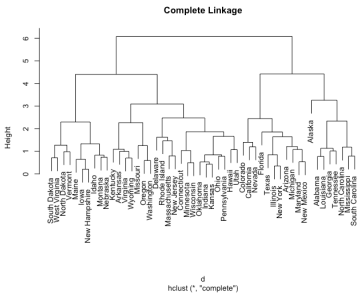

Figure 2: An example dendogram using complete linkage; $\max(\text{dist}(c_i, c_j))$.

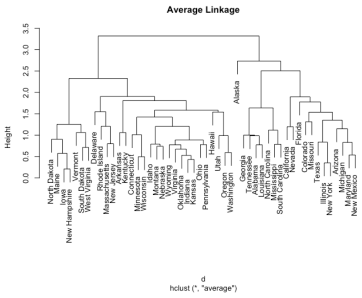

Figure 3: An example dendogram using average linkage; $\text{mean}(\text{dist}(c_i, c_j))$.

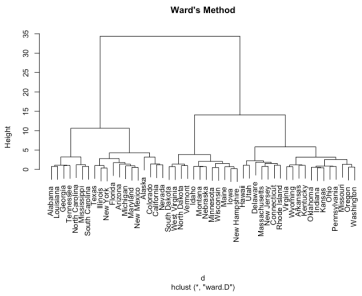

Figure 4: An example dendogram using Ward linkage; $\text{variance}(\text{merge}(c_i, c_j))$.

The reason NMF has this ability to produce results similar to KMeans clustering is described in Section 3 of "Orthogonal Nonnegative Matrix Tri-Factorizations for Clustering"

Nonnegative Matrix Factorization

A matrix $\mathbf{X}$ can be deconstructed and represented as the product of two matrixes, $\mathbf{F}$, $\mathbf{G}$, $\mathbf{X} = \mathbf{FG}$. If $\mathbf{X}$ is a $d \times n$ matrix, we can produce a $d \times k$, $\mathbf{F}$ matrix, and $k \times n$, $\mathbf{G}$ matrix.

From a clustering perspective, $\mathbf{F}$ represents the centroids of the k clusters and $\mathbf{G}$ represents the clustering membership of the n samples. This creates a direct relationship between matrix factorization and clustering. However, in order for this to be effective, both $\mathbf{F}$ and $\mathbf{G}$ must be nonnegative. This method is therefore called Nonnegative Matrix Factorization (NMF).

The objective for NMF is,

$$\min_{\mathbf{F},\mathbf{G}} ||\mathbf{X} - \mathbf{FG}||_F^2 \quad s.t.\ \mathbf{F} \geq 0, \mathbf{G} \geq 0. \tag{1}$$

In other words, the goal is to find the values for $\mathbf{F}$ and $\mathbf{G}$ where their product is as close to $\mathbf{X}$ as possible while ensuring that they are both nonnegative.

Euclidean vs Manifold Space

So far, whenever we've considered data, we consider its position in Euclidean space, by that we mean its x and y coordinates for example. This approach to data representation can be extremely effective but it is not the only approach and others may be effective in different cases. One such approach to data representation is the *manifold space*. The manifold space looks at the relationship between points instead of the position of points in a constant Euclidean space. Instead of points receiving an x and y coordinate, they receive a distance metric to all other points. We can represent our data as a graph and investigate its topology.

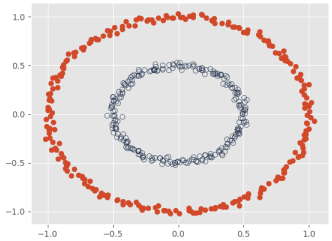

Figure 5: An example of concentric circles data that can be better represented using its topology than its absolute position in space.

A good example of data that is better represented using its topology instead of position in space is a set of concentric circles. If each circle in the set is considered a class, using the relationship between the points instead of their absolute position will allow algorithms to perform significantly better in classifying them.

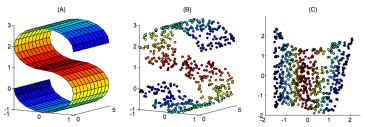

Figure 6: Example transformation using the manifold representation of data.

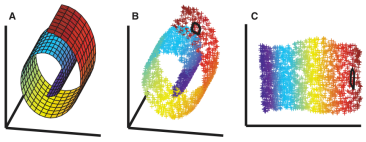

Figure 7: Example transformation using the manifold representation of data.

Spectral Clustering

Spectral clustering is an example of a very powerful clustering method in the manifold space. It is in fact simply applying the KMeans algorithm to data in the transformed manifold space. First, we must investigate how to transform our data into a representation extracted from the manifold.

The way we represent our data as a graph can be extracted from the points' position in absolute space. We can represent a graph using an $n \times n$ affinity matrix, $\mathbf{W}$, where w_{ij} is the distance between point i and point j.[2] In addition to the affinity matrix, $\mathbf{W}$, we define the degree matrix, $\mathbf{D}$, which is a diagonal matrix where $d_{ii} = \sum_{j=1}^{n} w_{ij}$. We also define the Laplacian matrix $\mathbf{L} = \mathbf{D} - \mathbf{W}$.

From the Laplacian matrix, we can learn a new representation, called an *embedding*[3], of our points from the first r eigenvectors of the matrix. We define $\mathbf{U}$ as the $n \times r$ matrix where column $\mathbf{u}_i$ is the i^{th} eigenvector of $\mathbf{L}$. The new representation of each data sample is the corresponding row in $\mathbf{U}$ which is called the *Laplacian embedding*. Spectral clustering is applying the KMeans algorithm on the points using this new representation.

This interpretation of Spectral Clustering should help explain the general concept. In reality we are actually solving an objective relating to the graph constructed by the data. We want to minimize the cut size, which is the number of cut edges in graph partitioning.

We can define our cluster indicator as,

$$q_i = \begin{cases} 1 & \text{if } i \in A \\ -1 & \text{if } i \in B \end{cases}$$

and we define the objective as,

$$\begin{aligned} \text{min Cut Size} = J &= \frac{1}{4}\sum_{i,j} w_{ij}\left[q_i - q_j\right]^2 = \frac{1}{4}\sum_{i,j} w_{ij}\left[q_i^2 + q_j^2 - 2q_iq_j\right] \\ &= \frac{1}{2}\sum_{i,j} q_i\left[d_i\delta_{ij} - w_{ij}\right]q_j = \frac{1}{2}\mathbf{q}^T(D-W)\mathbf{q} \end{aligned}$$

We can then relax the cluster indicators from discrete to continuous resulting in a 2-way partitioning of,

$$\begin{aligned} A &= \{i | q_2(i) < 0\}, \\ B &= \{i | q_2(i) \geq 0\}. \end{aligned}$$

[2] Here again, we can decide on the distance metric we wish to use whether it be Euclidean, Manhattan or others. Usually this is most effective when the data has some relational metrics.

[3] We will further discuss embeddings in the feature learning portion of this course.

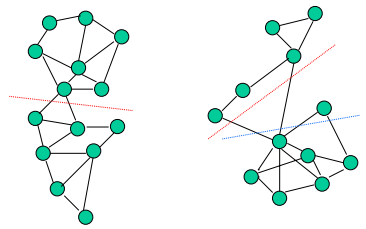

Figure 8: Example graph partitioning.

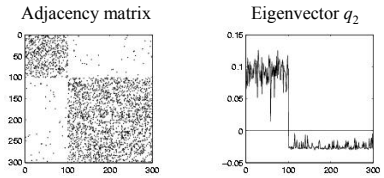

Figure 9: Example showing the relationship between adjacency and the second eigenvector.

We can aso recursively apply this formulation to k-way partitioning. The solution to the 2-way partitioning problem can be found be investigating the eigenvectors of $(\mathsf{D}-\mathsf{W})\mathsf{q}=\lambda\mathsf{q}$.

We define $s(A,B)=\sum_{i\in A}\sum_{j\in B}w_{ij}$ and $d_A=\sum_{i\in A}d_i$ and from there we can identify multiple types of cuts that we can optimize for. The **ratio cut** minimizes the smiilarity between two sets while balancing the size of clusters. The **normalized cut** minimizes the similarity between two sets while balancing their weight. The **min-max cut** minimizes the similarity between two sets while maximizing each set's similarity with itself. We define the objectives for each of these cuts as,

$$\min J_{\text{Ratio Cut}}(A,B)=\frac{s(A,B)}{|A|}+\frac{s(A,B)}{|B|}, \tag{2}$$

$$\begin{aligned}\min J_{\text{Normalized Cut}}(A,B)&=\frac{s(A,B)}{d_A}+\frac{s(A,B)}{d_B} \\ &=\frac{s(A,B)}{s(A,A)+s(A,B)}+\frac{s(A,B)}{s(B,B)+s(A,B)},\end{aligned} \tag{3}$$

$$\min J_{\text{Min-Max Cut}}(A,B)=\frac{s(A,B)}{s(A,A)}+\frac{s(A,B)}{s(B,B)}. \tag{4}$$

The solutions to these objectives are the second eigenvector of L for ratio cut, the second smallest eigenvector of $(\mathsf{D}-\mathsf{W})\mathsf{q}=\lambda\mathsf{D}\mathsf{q}$ for normalized cut, and second largest eigenvector of $(\mathsf{D}-\mathsf{W})\mathsf{q}=\lambda\mathsf{D}\mathsf{q}$ for min-max cut.

We can also expand the objective to k sets, $C_1,\ldots,C_k$, as,

$$\begin{aligned}\min J_{\text{Ratio Cut}}(C_1,\ldots,C_k)&=\sum_{<k,l>}\left(\frac{s\left(C_k,C_l\right)}{|C_k|}+\frac{s\left(C_k,C_l\right)}{|C_l|}\right) \\ &=\sum_k\frac{s\left(C_k,G-C_k\right)}{|C_k|},\end{aligned} \tag{5}$$

$$\begin{aligned}\min J_{\text{Normalized Cut}}(C_1,\ldots,C_k)&=\sum_{<k,l>}\left(\frac{s\left(C_k,C_l\right)}{d_k}+\frac{s\left(C_k,C_l\right)}{d_l}\right) \\ &=\sum_k\frac{s\left(C_k,G-C_k\right)}{d_k},\end{aligned} \tag{6}$$

$$\begin{aligned}\min J_{\text{Min-Max Cut}}(C_1,\ldots,C_k)&=\sum_{<k,l>}\left(\frac{s\left(C_k,C_l\right)}{s\left(C_k,C_k\right)}+\frac{s\left(C_k,C_l\right)}{s\left(C_l,C_l\right)}\right) \\ &=\sum_k\frac{s\left(C_k,G-C_k\right)}{s\left(C_k,C_k\right)}.\end{aligned} \tag{7}$$

In terms of a performance comparison for these objectives, if clusters are *well separated*, all three give similar and accurate results. When clusters are *marginally separated*, minimizing the **normalized cut or the min-max cut** produces better results. Finally, when clusters *overlap significantly*, minimizing the **min-max cut** produces the most compact and balanced clusters.