STATS 306B: Unsupervised Learning **Spring 2014**

Lecture 6 — April 16

Lecturer: Lester Mackey *Scribe: Peyton Greenside, Natalie Telis*

6.1 Hierarchical Clustering

Last time, we introduced the task of hierarchical clustering, in which we aim to produce nested clusterings that reflect the similarity between clusters. This contrasts sharply with our former discussion of "flat" or structureless clustering methods like k-means which do not model relationships between clusters. In this lecture, we will continue our discussion of the standard model-free approaches to hierarchical clustering by considering both of the principal paradigms for hierarchical clustering:

1. Agglomerative / Bottom-Up Clustering, in which we recursively merge similar clusters
2. Divisive / Top-Down Clustering, in which we recursively sub-divide into dissimilar sub clusters

The standard approaches in both settings are greedy fashion and so typically not optimal in any sense.

6.2 Agglomerative Clustering

Agglomerative clustering is the more common of the two paradigms. The standard greedy algorithm proceeds as follows:

1. Start with all data points in their own clusters
2. Repeat until only one cluster remains:

   -Find 2 clusters (C_1, C_2) that are most similar (that have the smallest pairwise cluster dissimilarity $d(C_1, C_2)$)

   -Merge C_1, C_2 into one cluster

In order to perform the above algorithm we need to know how to construct cluster dissimilarity from a measure of data point similarity $d(x, x')$? The three most common measures, outlined below, are single linkage, complete linkage, and average linkage.

6.2.1 Single Linkage

The **single linkage** measure $d_{sl}(C_1, C_2) = \min_{x_1 \in C_1, x_2 \in C_2} d(x_1, x_2)$ judges cluster similarity by the most similar points in the clusters being compared. A direct implementation of this algorithm would require order of (# clusters)^2 work to choose clusters and update the minimum dissimilarity to the newly merged cluster. The complexity could be reduced significantly with appropriate data structures. A drawback of this choice is that single Linkage may yield long, extended clusters or chains of points ("chaining") so points in the same cluster could be quite dissimilar.

6.2.2 Complete Linkage

The complete linkage measure $d_{cl}(C_1, C_2) = \max_{x_1 \in C_1, x_2 \in C_2} d(x_1, x_2)$ judges cluster similarity by the least similar points in the clusters being compared. A direct implementation of this algorithm would require order of (# clusters)^2 work to choose clusters and update the maximum dissimilarity to the newly merged cluster. The complexity could be reduced significantly with appropriate data structures. A drawback of this choice is that points in one cluster may be closer to points in another cluster than to any of its own cluster.

6.2.3 Average Linkage

The average linkage measure $d_{al}(C_1, C_2) = \frac{1}{|C_1||C_2|} \sum_{x_1 \in C_1, x_2 \in C_2} d(x_1, x_2)$ judges cluster similarity by average similarity of all of the points in the clusters being compared. This metric can be considered a compromise between single linkage and complete linkage. A direct implementation of this algorithm would require order of (# clusters)^2 work to choose clusters and update the average dissimilarity to the newly merged cluster. The complexity could be reduced significantly with appropriate data structures. A drawback of this choice is that average linkage is not invariant to increasing or decreasing transformations of dissimilarity matrix d. This property could be regained by using the median dissimilarity instead of the average.

6.2.4 Summary

The three methods outlined above are the most common measures of cluster dissimilarity. They yield similar results when data is clustered and clusters are **compact** (more similarity within clusters than between). Otherwise the results may differ. With any of the above choices, the algorithm is **monotonic** meaning that the similarity between merged clusters at the t^{th} merge decreases monotonically with t.

6.3 Divisive Clustering

Divisive clustering is the second, less common paradigm for hierarchical clustering. It can be advantageous to use divisive clustering over agglomerative when you only want a small number of total clusters. Here are two common options for implementation.

6.3.1 Option 1

Option 1 recursively applies 2-means or 2-medoids to a selected cluster. The downsides of this approach are that

1. Results depend on the initialization of k-means / k-medoids at each division step because of multiple stationary points
2. The algorithm can violate the monotonicity property, as divided clusters at split t can be even more similar than those at $t + 1$. This is a problem for making dendrograms.

As a result, we will focus our attention on a second option.

6.3.2 Option 2

Option 2 is a greedy approach that was described in Smith et al. (1965). It follows the following steps:

1. Start with all data in one cluster
2. Repeat until all clusters are singletons (or until no dissimilarity remains between clusters):

   -Choose one cluster G

   -Remove the point in G with largest average dissimilarity with all other points in G; add that point to a new cluster H

   -Repeat the following until no point in G is closer to H on average:

   -Remove $x^* = \underset{x \in G}{\operatorname{argmax}} \frac{1}{|G-1|} \sum_{g \in G \setminus \{x\}} d(x,g) - \frac{1}{|H|} \sum_{h \in H} d(x,h)$ and add to H

Option 2 avoids the downsides of Option 1 (dependence on initialization, monotonicity violation), but a question that arises is how should we choose the cluster G to split? Two commonly used methods to choose G are:

1. Select G with the largest diameter, $\operatorname{diam}(G) = \max_{x,x' \in G} d(x,x')$
2. Select G with the largest average pairwise point dissimilarity.

6.4 Applications

Let us turn to particular examples and applications of hierarchical clustering; we will reference the accompanying slides for this lecture.

6.4.1 Agglomerative Clustering

A visualization of agglomerative clustering in action is found on slide 3 (beginning with "Data" and ending with "iteration 28"). Each slide represents an iteration of the clustering algorithm; all points in a common cluster are connected by red lines. In iterations 10-11, we see an example of two previously merged clusters also merging.

Agglomerative Clustering of Cancer Gene Expression Profiles

On slide 4, we see a **dendrogram**, a visual representation of cluster mergings in hierarchical clustering. Clusters are arranged by merging order and nested thereby. This dendrogram represents a series of clusterings of gene expression samples from 64 samples with 12 cancer types. Observe that many cancer types cluster together.

Comparison of Clustering Methods

On slide 5, we see displayed the results of Agglomerative Clustering on the dataset shown in slide 4 when using different similarity metrics. Here, in order, we see d_{al}, d_{cl}, and d_{sl}. Note that clustering is very method-dependent, and differences are evident between all three agglomerative clustering approaches.

As we note that clustering is very method-dependent, several peers note that single linkage, which uses the d_{sl} similarity metric, results in very "long" clusters, whereas complete linkage (which uses d_{cl}) results in a more balanced clustering. In general, the instructor notes that associations verified only by a single clustering method may be suspect and that evidence for clustering structure could be buttressed by agreement among disparate methods.

Clustering Human Microarray Data

Clustering can be done over genes which are expressed rather than the samples in which they are expressed as well. On slide 6 we see an example of a clustering of approximately 7,000 genes in cancer samples. Note that the order of the clusters is not unique, as any cluster ordering can be reversed by rotation around a node. Generally an ordering scheme can be very important to visual interpretation of clustering data. The *hclust* function in R follows the convention of ordering by tightness of cluster.

A Brief Note: Challenges in Interpreting Dendrograms

Selecting the "correct" number of clusters from a hierarchical clustering to can be challenging. This is generally done by "cutting" a tree off at a given point (truncation at a dissimilarity truncation level) and enumerating the remaining branches (see slide 7, fig3); the height of the "cut" will result in differing numbers of clusters. Choosing k intelligently and informatively to classify and understand data is crucial. Because there are many levels of clustering, it is possible to observe late-term spikes in the gap statistic for different values of k (slide 7, fig 4b). These spikes reflect smaller clusters nested within larger ones.

Breast Cancer Subtypes

By clustering gene expression in several manners on several tumor gene expression datasets, we can observe (slide 8) several different breast cancer types, as well as several suites of genes highly expressed per subtype. In particular this is noteworthy because it is well-validated and generates testable hypotheses to confirm validity of clustering predictions.

An Unusual Example: Clustering Great Paintings

After clustering 57 paintings by ratings for composition, color, drawing type, and expression, the authors discussed in slide 9 generated a dendrogram of great artwork labeled by painter. The authors note that the value of this analysis is dependent on the interesting hypotheses generated by the clustering, as in the example above.

6.4.2 Practicalities and Challenges

Model selection presents challenges similar to k-means cluster selection.

Model selection (choosing truncation level, referred to in 6.4.1, in **A Brief Note...**) is a challenge in creating meaningful interpretations of hierarchical clusterings. Although there is no single solution to interpretation, many of the methods we have discussed for k selection in k-means may apply equally well here.

Interpreting dendrograms is challenging for large datasets.

It is difficult to interpret dendrograms even past the selection of a model because large datasets may make visualizing clusters impossible, as well as choosing a meaningful k difficult (if there are several thousand data points, there may be tens or hundreds of meaningful clusters).

One solution is labeling each interior node with a prototype data point. One could simply choose the average or median of those in the cluster. A much more informative choice would be labeling each cluster with a prototypical datapoint which is minimally dissimilar from every point in the cluster. This is called **minimax linkage**: when aiming to label a cluster C, choose the prototype $x^* = \operatorname{argmin}_{x \in C} \max_{x' \in C} d(x, x')$.

6.5 Spectral Clustering

We next consider another type of clustering known as **spectral clustering**. Spectral clustering is motivated by the desire to detect connected by non-compact and non-convex clusters that prior methods like k-means fail to detect. On slide 12 we see such an example of concentric rings of datapoints. On the left we see the results of a k-means clustering approach, compared with spectral clustering on the right. In this case, k-means attempts to find compact clusters based on Euclidean distance and so divides each concentric ring in half; spectral clustering instead perfectly identifies the two concentric rings of points. In the remainder of this lecture, we will study the spectral clustering algorithm in more detail.

6.5.1 Spectral Clustering

In the **spectral clustering** setup, we begin with a matrix $S \in \mathbb{R}^{nxn}$ of similarities s_{ij} between points x_i and x_j in our data set. We will use this matrix to derive a **weighted adjacency matrix**,

$$W = g(S) \in \mathbb{R}^{nxn}$$

in a problem-specific fashion. We then will view our data as vertices within a weighted graph $G = (V, E, W)$ where if $w_{ij} = 0$, there is no edge between vertices i and j, and, if the weight is non-zero, the edge weight is equal to w_{ij}.

Our goal is now to partition the vertices of G such that there is little weight between partitioned groups and high weight within partitioned groups. Spectral clustering atttempts to achieve this end with the following algorithm:

Spectal Clustering Algorithm

1. Construct $W = g(S) \in \mathbb{R}^{nxn}$ as above.
2. Form a graph Laplacian $L = f(W) \in \mathbb{R}^{nxn}$.

   -Different choices of Laplacian give rise to different spectral clustering variants.
3. Identify eigenvectors $U \in \mathbb{R}^{nxk}$ of L corresponding to the k *smallest* eigenvalues.

   -We can view each row $\tilde{x_i}$ of U as a new, k-dimensional representation of the datapoint x_i.
4. Cluster $\tilde{x_i}$ using k-means with k clusters.

6.5.2 Questions and Closing Remarks

At this point, the spectral algorithm is rather mysterious, and a number of questions remain to be answered:

- What is L?
- Why is U a useful alternative representation of our data?
- How do we choose W?

We will address each of these in detail next lecture, but let us end this lecture with a description of the standard choices of graph Laplacian L.

Graph Laplacians

To define the standard graph Laplacians, we will need some additional notation. We must first define the degree of a vertex i as $d_i = \sum_{j=1}^{n} w_{ij}$ and the the **degree matrix** as the diagonal matrix

$$D = \text{diag}(d_1, d_2, \ldots, d_n).$$

In these terms, the most commonly used Laplacian matrices are

- The unnormalized Laplacian, $L_{un} = D - W$
- The symmetric normalized Laplacian, $L_{sym} = I - D^{-1/2} W D^{-1/2}$
- The random walk normalized Laplacian, $L_{rw} = D^{-1} L = I - D^{-1} W$.

All three have similar desirable properties that make the minimum eigenvectors useful. Next time, we will elucidate the properties of L_{rw}.