CS 5785 – Applied Machine Learning – Lec. 12

Prof. Serge Belongie,
Cornell Tech Scribe: Kuowei Tseng, Mengjue Wang,
Ming Chen, Haotian Zhang, Zaid Haque, Brook Du, Han Nie, Kelly Wang,
Feston Kastrati, Vijay Pillai

Oct. 12, 2017

1 Clustering

1.1 Review from Last Time

K-means is an iterative clustering algorithm that divides a set of feature vectors into K clusters. One caveat of using K-means is that it assumes our data lives in a vector space and that we can compute Euclidean distance between these data points. In practice, not all clustering problems come to us with this representation.

1.2 Definitions

A clustering algorithm is an unsupervised method that finds a cluster C such that $C : \{1, ..., N\} \rightarrow \{1, ..., k\}$ in order to minimize the with-in cluster variance.

Multi-dimensional scaling methods use a distance Matrix $D \in R^{N \times N}$ to find vectors $x_1, ..., x_n \in R^N$ such that $||x_i - x_j||_2 \approx D_{ij}$

1.3 K-Medoids

In cases where we have a table of dissimilarities without access to the vector space from which they arose, as seen in Figure 1, we can't use K-Means, but we can use K-Medoids instead. K-Medoids restricts the cluster centers to lie on existing data points (as opposed to K-Means that uses vector averages) but other than that it is very similar to K-means, see Algorithm 1. K-Medoids is a central clustering method, as opposed to K-means, which is a pairwise clustering method.

Later in the course we'll learn about another unsupervised learning method called *Multidimensional Scaling* (MDS) that also operates on dissimilarity matrices. Whereas K-Medoids simply aims to cluster the data, MDS seeks a low dimensional vector embedding that fits the observed dissimilarities.

Like K-Means clusteing, K-Medoids clustering has an objective function given by

$$w'(c) = \sum_{k=1}^{K} \sum_{C(i)=k} D^*_{ii_k}$$

TABLE 14.3. *Data from a political science survey: values are average pairwise dissimilarities of countries from a questionnaire given to political science students.*

	BEL	BRA	CHI	CUB	EGY	FRA	IND	ISR	USA	USS	YUG
BRA	5.58										
CHI	7.00	6.50									
CUB	7.08	7.00	3.83								
EGY	4.83	5.08	8.17	5.83							
FRA	2.17	5.75	6.67	6.92	4.92						
IND	6.42	5.00	5.58	6.00	4.67	6.42					
ISR	3.42	5.50	6.42	6.42	5.00	3.92	6.17				
USA	2.50	4.92	6.25	7.33	4.50	2.25	6.33	2.75			
USS	6.08	6.67	4.25	2.67	6.00	6.17	6.17	6.92	6.17		
YUG	5.25	6.83	4.50	3.75	5.75	5.42	6.08	5.83	6.67	3.67	
ZAI	4.75	3.00	6.08	6.67	5.00	5.58	4.83	6.17	5.67	6.50	6.92

Figure 1: The missing data is symmetric to the existing data

Data: Dissimilarities table D, K
Result: K clusters
initialize C;
while C *changes* **do**

- Find the medoid for each cluster:

$$i_k^* = \underset{\{i:C(i)=k\}}{\arg\min} \sum_{C(i')=k} D(x_i, x_i')$$

- Set $m_k = x_{i_k^*}$ for $k = 1, 2, \ldots, K$.
- Assign each observation to the closest medoid:

$$C(i) = \underset{1 \le k \le K}{\arg\min} \quad D(x_i, m_k)$$

end

Algorithm 1: K-Medoids [HTF Algorithm 14.2]

Note that because the objective function decreases in each iteration step and because there is a finite number of elements in the dataset, the K-Medoids algorithm will terminate in finite time.

1.4 Hierarchical Clustering

K-medoids, like K-means, requires us to choose K, which as we've seen before can be tricky. One way around this problem is to use an *agglomerative* (bottom up) or *divisive* (top down) method, which result in a hierarchical clustering. Agglomerative clustering works as follows:

1. Initialization: Start with every data point as a singleton cluster ($|C| = N$).
2. Step: Merge the closest two clusters ($|C_{new}| = |C_{old}| + 1$).
3. Repeat 1 and 2 until everything is merged ($|C| = 1$).
4. Return the list of merges preformed in the correct order (See Figure 2).

There are a number of ways to measure dissimilarities between two groups of data points resulting from the merging steps:

1. Single Linkage - The distance between the groups is the distance between the closest pair of data points, one from each group:

$$d_{SL}(G, H) = \min_{\substack{i \in G \\ i' \in H}} d_{ii'}$$

2. Complete Linkage - The distance between the groups is the distance between the farthest pair of data points, one from each group:

$$d_{CL}(G, H) = \max_{\substack{i \in G \\ i' \in H}} d_{ii'}$$

3. Group Average - The distance between the groups is the average distance between each pair of data points, one from each group:

$$d_{GA}(G, H) = \frac{1}{N_G N_H} \sum_{i \in G} \sum_{i' \in H} d_{ii'}$$

See Figure 3 for the difference in the results of the algorithm when using the different merging techniques. Similarly, divisive clustering starts with all data points in one merged cluster and iterates until each data point is in its own singleton cluster.

If we want to return a partitioning of the original dataset to clusters we choose a level in the returned hierarchy and return the partitioning up to that level. We can imagine a dial that we can turn to move up and down in the levels of the hierarchy and see the clusters change, enabling us to choose the level most appropriate to our use case.

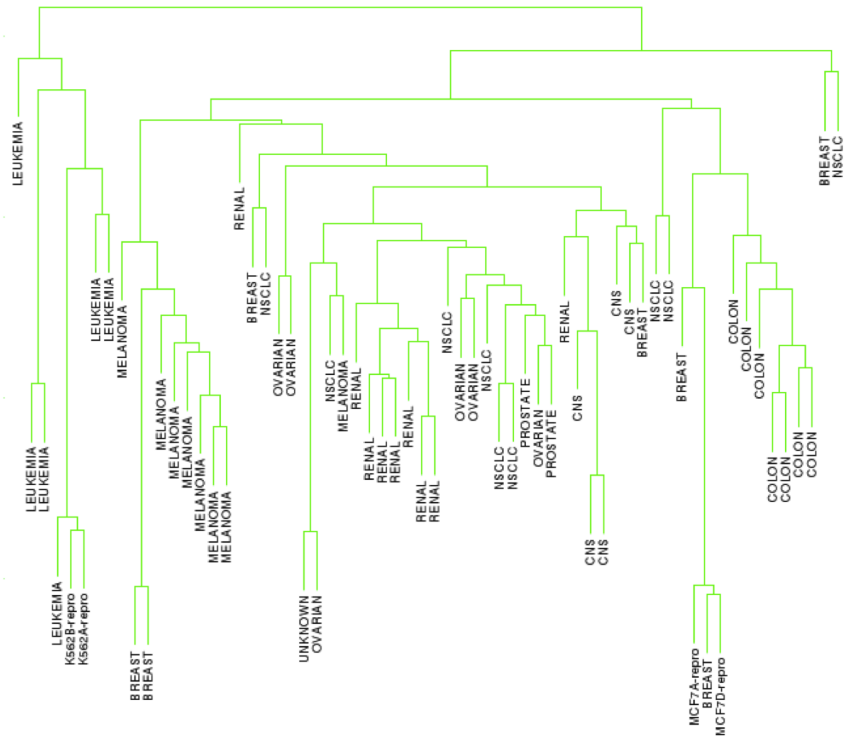

FIGURE 14.12. *Dendrogram from agglomerative hierarchical clustering with average linkage to the human tumor microarray data.*

Figure 2: The return value of the *Hierarchical Clustering* algorithm

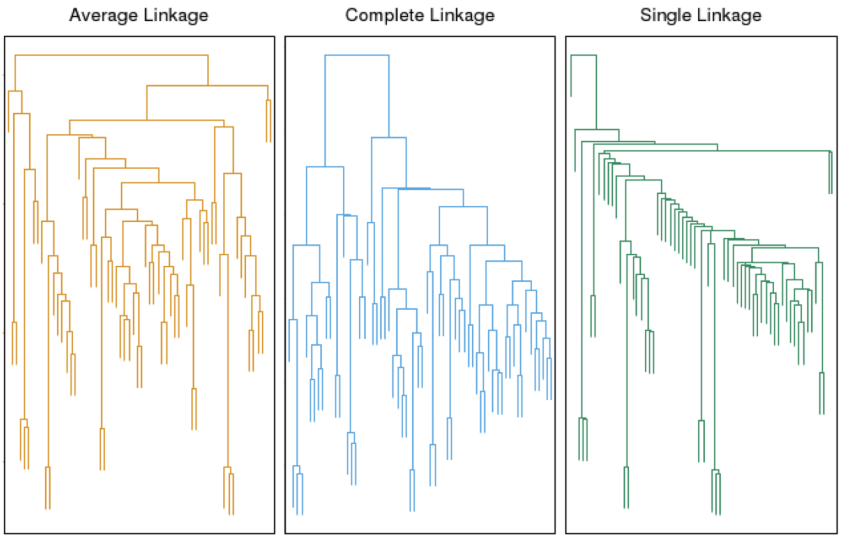

FIGURE 14.13. *Dendrograms from agglomerative hierarchical clustering of human tumor microarray data.*

Figure 3: The output of the hierarchical clustering algorithm when using different merging techniques.

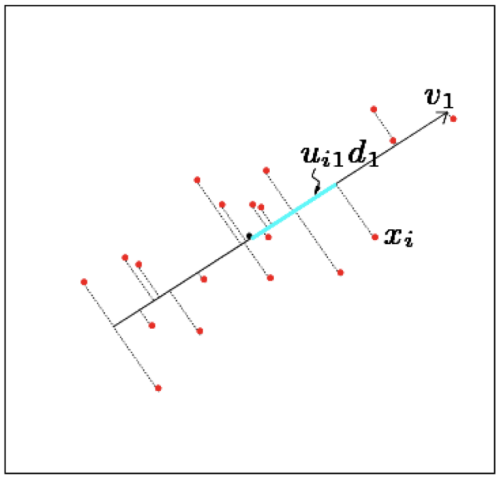

FIGURE 14.20. *The first linear principal component of a set of data. The line minimizes the total squared distance from each point to its orthogonal projection onto the line.*

Figure 4:

2 Principal Components Analysis (PCA)

PCA is an unsupervised dimensionality reduction technique used to find axes of variations in a dataset. In a sense, clustering may also be viewed as an unsupervised dimensionality reduction technique, in that it assigns possibly high dimensional feature vectors to one of K tokens (cluster labels). Clustering is useful when the data is *clumpy*, e.g., arranged in a manner reminiscent of Gaussian point clouds. For example, if a group of people play darts on a single target the darts would be arranged in a clump around the bullseye. Clumpy data is well represented by tokens or cluster centers. PCA is useful for "manifoldy" data (specifically, linear manifolds), which is well represented in low dimension coordinate system. Often, problems that are well suited to K-means are ill suited to PCA and vice versa.

As a simple example of dimensionality reduction, suppose you collect the feet length and width measurements for a group of people. This yields a list of 2D feature vectors per person. We would like to "project this down" to 1D in a manner that captures as much of the variability in the data ase possible. In particular, if we can find the axis with greatest variance as shown in Figure 4, the single coordinate of each data point along that axis could be a good choice for a reduced dimensionality representation of the data.

PCA attempts to capture the axes of variation by a sequence of projections of a dataset, mutually uncorrelated and ordered in variance. Being mutually

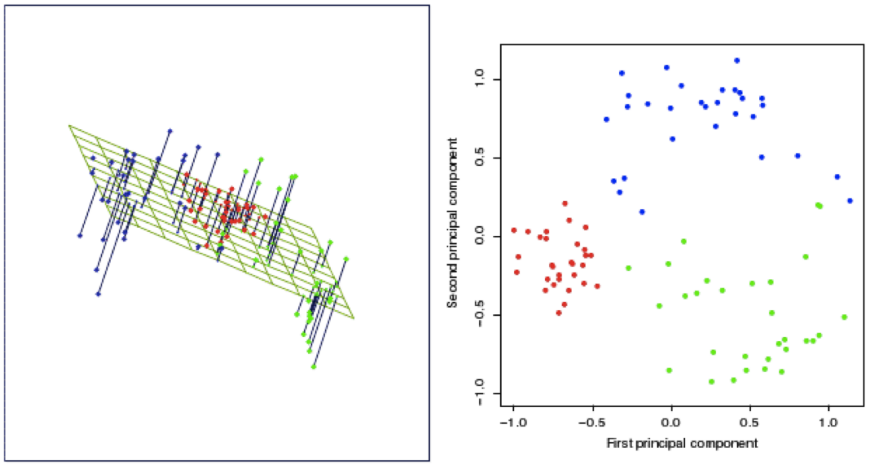

FIGURE 14.21. *The best rank-two linear approximation to the half-sphere data. The right panel shows the projected points with coordinates given by* $\mathbf{U}_2\mathbf{D}_2$*, the first two principal components of the data.*

Figure 5:

uncorrelated we get a set of projections that do not contain duplication of data and assure we have the minimal set of projections containing the amount of data we need. By ordering the projections in variance we gain the ability to drop projections at the end of the list that are uninformative, that can perhaps be considered as noise, and by doing so reduce the dimentionality.

Given a set of N points $x_i \in \mathbf{R}^p$, we suppose that they live on a linear manifold, rather than simply filing up $\mathbb{R}^p$ in an unstructured manner. The idea behind dimensionality reduction that PCA attempts to solve is to keep the best *rank* q *approximation* of the data for some $q < p$. The model for representing the data is a non-parametric representation of an affine hyperplane of rank q:

$$f(\lambda) = \mu + \mathbf{V}_q\lambda$$

Where:

- μ: a location vector in $\mathbf{R}^p$, can also be viewed as the "centered" origin of the data.
- $\mathbf{V}_q \in \mathbf{R}^{p\times q}$, with q orthogonal unit vectors as its columns.
- λ: a vector of q parameters.

Figure 4 is an example of a dimensionality reduction where $p = 2$ and $q = 1$. In Figure 5 $p = 3$ and $q = 2$.

We want to find the matrix $\mathbf{V}_q$ and vector λ that generate the smallest reconstruction error. This is a least squares fitting problem:

$$\min_{\substack{\mu \\ \lambda_i \\ \mathbf{V}_q}} \sum_{i=1}^{N} \|x_i - \mu - \mathbf{V}_q \lambda_i\|^2$$

It is relatively straightforward to show that the optimal μ and λ_is are given by:

- $\hat{\mu} = \bar{x}$: The new origin is the average of all the original data points, the center of the data.
- $\hat{\lambda}_i = \mathbf{V}_q^\top (x_i - \bar{x})$: The projection after centering around the mean.

We are left with finding the optimal orthogonal matrix $\mathbf{V}_q$:

$$\min_{\mathbf{V}_q} \sum_{i=1}^{N} \|(x_i - \bar{x}) - \mathbf{V}_q \mathbf{V}_q^\top (x_i - \bar{x})\|^2$$

For simplicity, assume we have a *centered* set of data, i.e., $\bar{x} = 0$, and we get:

$$\min_{\mathbf{V}_q} \sum_{i=1}^{N} \|x_i - \mathbf{V}_q \mathbf{V}_q^\top x_i\|^2$$

Where $\mathbf{V}_q \mathbf{V}_q^\top$ is a $p \times p$ projection matrix. Denote $\mathbf{M}_q = \mathbf{V}_q \mathbf{V}_q^\top$. $\mathbf{M}_q$ maps each x_i onto its rank q reconstruction, the orthogonal projection of x_i onto the subspace spanned by the columns of $\mathbf{V}_q$. If $p = q$, $\mathbf{V}_q \mathbf{V}_q^\top$ is an identity matrix.

We've already seen the machinery we need to solve this for $\mathbf{V}_q$: the SVD of $\mathbf{X}$, where $\mathbf{X}$ is formed by stacking the (centered) observations into the rows of an $N \times p$ matrix. The SVD of $\mathbf{X}$ is $\mathbf{UDV}^\top$, where $\mathbf{D}$ is a diagonal matrix with values $d_1, \ldots, d_p$ where: $d_1 \geq \ldots \geq d_p \geq 0$. For each rank q the solution to $\mathbf{V}_q$ above is given by the leading q singular vectors in $\mathbf{V}$ (from the SVD). The columns of $\mathbf{UD}$ are the principal components of $\mathbf{X}$. The projection onto the sungular vectors gives us coordinates in a linear subspace, as shown in the right graph in Figure 5.

The sample covariance for $\mathbf{X}$, assuming it is centered, is:

$$\mathbf{S} = \frac{1}{N} \mathbf{X}^\top \mathbf{X} \in \mathbb{R}^{p \times p}$$

PCA computes the eigenvectors of $\mathbf{S}$ to reveal the orthogonal axes of variation in the data that capture the most variance, meaning the axes where the data spreads the most. PCA does this by throwing away small eigenvalues and their corresponding eigenvectors to leave the highest q singular values.