STAT 542: Statistical Learning

Clustering

Ruoqing Zhu, Ph.D. <rqzhu@illinois.edu>

Course Website: https://teazrq.github.io/stat542/

October 5, 2020

Department of Statistics
University of Illinois at Urbana-Champaign
Unsupervised Learning
Unsupervised Learning

• No response variable $Y$, only $\{x_i\}_{i=1}^n$.

• Goal: learn patterns in $X$.

• Examples
  • Estimate the density, covariance, graph (network), etc. of $X$ — could be difficult in high-dimensional settings
  
  • Cluster analysis: identify multiple regions of the feature space that contains modes of density.

  • Dimension reduction: identify low-dimensional manifold within the feature space $\mathcal{X}$ that represents high data density.

• Oftentimes, there is no clear measure of success.
Cluster Analysis
Cluster Analysis

- **Group the dataset into subsets** so that those within each subset are more closely related (similar) to each other than those objects assigned to other subsets. Each subset is called a **cluster**.

- Flat clustering vs. hierarchical clustering: flat clustering divides the dataset into $k$ clusters, and hierarchical clustering arranges the clusters into a natural hierarchy.

- Clustering results are crucially dependent on the **measure of similarity** (or distance) between the “points” to be clustered.
• A distance metric or a distance function is a function that defines the similarity of two elements (points, sets, etc.)

• For the distance of two points (with continuous entries), the most commonly used measurement is the Euclidian distance:

\[ d(u, v) = \|u - v\|_2 = \sqrt{\sum_{j=1}^{p} (u_j - v_j)^2} \]

• For categorical entries, the Hamming distance is usually used

\[ d(u, v) = \sum_{j=1}^{p} 1\{u_j \neq v_j\} \]

• Distance measures should be defined based on the application. There is no universally best approach.
• Suppose we have a set of \( n \) data points

• We want to form \( K \ll n \) clusters, indexed by \( k \in \{1, \ldots, K\} \).

• Let \( C(\cdot) \) be a cluster index function that assign the \( i \)th observation or cluster \( C(i) \).

• Consider: search for a function \( C: \{1, \ldots, n\} \rightarrow \{1, \ldots, K\} \) to minimize the within cluster distance:

\[
W(C) = \frac{1}{2} \sum_{k=1}^{K} \sum_{C(i), C(i') = k} d(x_i, x_{i'}). 
\]
Clustering

- This is equivalent to maximizing the between cluster distance

\[ B(C) = \frac{1}{2} \sum_{k=1}^{K} \sum_{C(i)=k} \sum_{C(i') \neq k} d_{ii'} \]

- Note that the total distance can be broken down into

\[ T = \frac{1}{2} \sum_{i=1}^{n} \sum_{i'=1}^{n} d_{ii'} = \frac{1}{2} \sum_{k=1}^{K} \sum_{C(i)=k} \left[ \sum_{C(i')=k} d_{ii'} + \sum_{C(i') \neq k} d_{ii'} \right] \]

\[ = W(C) + B(C) \]

- The total distance is fixed for a given set of data, hence

\[ \text{minimize } W(C) \iff \text{maximize } B(C) \]
• Given a specific distance measure $d(\cdot, \cdot)$, several algorithms can be used to find the clusters
  – Combinatorial algorithm
  – $K$-means clustering
  – Hierarchical clustering
Combinatorial Algorithm
• For small \( n \) and \( K \), we could minimize \( W \) by brute-force search.

• However, the number of “tries” needed to complete the search is

\[
S(n, K) = \frac{1}{K!} \sum_{k=1}^{K} (-1)^{K-k} \binom{K}{k} k^n
\]

• For example \( S(10, 4) = 34,105 \); \( S(19, 4) \approx 10^{10} \).

• This is not feasible for large \( n \) and \( K \), since the number of distinct assignments can be extremely large.

• It calls for more efficient algorithms: may not be optimal but a reasonably good suboptimal partition.
$K$-means Clustering
Consider an enlarged optimization problem:

\[
\min_{C, \{m_k\}_{k=1}^{K}} \sum_{k=1}^{K} \sum_{C(i)=k} \|x_i - m_k\|^2
\]

Hence, we are solving both

- the cluster index function \( C(\cdot) \),
- and also the cluster centers \( m_k \), \( k = 1 \ldots K \).

This problem is NP-hard for \( \geq 2 \) dimensions.
• Combinatorial algorithm is too expansive.

• Instead, consider an algorithm that alternatively updates the two components:
  • $C$, the cluster assignments
  • $\{m_k\}_{k=1}^K$: the cluster means

• We will do an iterative update by:
  1) Fixing $C$, find the best $\{m_k\}_{k=1}^K$
  2) Fixing $\{m_k\}_{k=1}^K$, find the best $C$
• Fixing $C$, we know the cluster label of each subject. For any set \( \{ i : C(i) = k \} \), finding the mean is

$$m_k = \arg \min_m \sum_{C(i)=k} \| x_i - m \|^2.$$ 

• This is simply finding the mean within cluster $k$, i.e.

$$m_k = \frac{\sum_{C(i)=k} x_i}{\sum_i 1\{C(i) = k\}}.$$
Fixing the cluster means \( \{ m_k \}_{k=1}^{K} \), to find the new cluster assignments, we simply recalculate the distance from an observation to each of the cluster mean.

\[
C(i) = \arg\min_{k \in \{1, \ldots, K\}} d(x_i, m_k)
\]

Hence each point will be assigned to the closest cluster mean.
A $K$-means Clustering algorithm:

1) Randomly split the dataset into $K$ different subsets. Assign each subsets a cluster label. Then iterate between 2) and 3).

2) Given the cluster assignment $C$, calculate the cluster mean vectors $m_1, \ldots, m_K$.

3) Given the current set of means $\{m_1, \ldots, m_K\}$, assign each observation to the closest current cluster mean.

Stop the algorithm when $C$ does not change.
Demonstration
Demonstration
• **Note**: We usually initiate the cluster labels randomly. However, this algorithm does not guarantee to global minimum. Example?

• The algorithm still has a **descent property**, which leads to a local minimizer.
• *K*-medoids is an alternative version of *K*-means:

• Replace the second step by searching for the one observation that minimizes the distance to all others in the cluster

\[
i_k^* = \arg \min_{i: C(i) = k} \sum_{C(i') = k} D(x_i, x_{i'})
\]

• Use \(x_{i_k^*}\) as the “center” of cluster \(k\).
Applications

• See the supplementary R file

• Example 1: the iris data

• Example 2: cluster pixels in a picture
• Clustering may help in supervised learning
• How to choose the number of clusters $K$
• Other distance measures
• Categorical variables
Hierarchical Clustering
Hierarchical Clustering

- Choosing the number of clusters $K$ can be difficult
- A hierarchical representation which
  - at the lowest level, each cluster contains a single observation.
  - at the highest level there is only one cluster containing all observations.
- Use dendrogram to display the clustering result.
• Suppose we have a set of 6 observations
• A typical dendrogram from hierarchical clustering
• How to construct this?
Demonstration
An agglomerative algorithm is a “bottom up” approach:

- Begin with every observation representing a singleton cluster.
- At each step, merge two “closest” clusters into one cluster and reduce the number of clusters by one.
- Stop when all observations are in the same cluster

**How to choose which two clusters to merge?**

This requires:

- A distance measure between any two observations \( d(x_i, x_j) \)
- A distance measure between any two sets of observations \( d(G, H) \)
Distance Measures

• Distance $d(G, H)$ between two clusters $G$ and $H$ can be defined in different ways:
  - **Complete linkage** (default of `hclust()`): the furthest pair
    \[
    d(G, H) = \max_{i \in G, i' \in H} d_{ii'}
    \]
  - **Single linkage**: the closest pair
    \[
    d(G, H) = \min_{i \in G, i' \in H} d_{ii'}
    \]
  - **Average linkage**: average dissimilarity
    \[
    d(G, H) = \frac{1}{n_G n_H} \sum_{i \in G} \sum_{i' \in H} d_{ii'}
    \]
  - Different choice may results in different hierarchical structures
To perform a hierarchical clustering, a matrix of all the pair-wise distances is sufficient.

We don’t have to know the values of the original observations.

This is an \( n \times n \) matrix: the \((i, i')\)'s element represents the distance between \( x_i \) and \( x_{i'} \).

This matrix is also called a **dissimilarity matrix**.

- Symmetric
- diagonal elements are zero
Applications

- See the supplementary R file
- Example 1: the iris data
- Example 2: RNA expression data
Principle Component Analysis
Principle Component Analysis (PCA) is an old but very useful technique invented by Karl Pearson in 1901. The main purpose is data visualization (mostly in 2d). It also serves as a dimension reduction method. Unsupervised method, can be used for preprocessing the data.
Principle Component Analysis

- Given that we have a $n \times p$ design matrix $X$, there are many equivalent approaches (motivations):
  - **Explain the most variation**: Produce a derived set of uncorrelated variables $Z_k = X\alpha_k$, $k = 1, \ldots, q < p$ that are linear combinations of the original variables, and explain most of the variation in the original set
  - **Approximate the design matrix**: Approximate the design matrix $X$ by the best (using Frobenius norm) rank-$q$ matrix, which can be performed through SVD
• Suppose we have an $n \times p$ design matrix $X$

• The first step is to **center each variable**, i.e., subtract the column means from each column respectively.

• In the following, we assume that $X$ is already centered.

• Centering $X$ does nothing but re-positioning the axis
• One way to understand the PCA is using a **singular value decomposition** (SVD)

\[
X_{n \times p} = U_{n \times n} D_{n \times p} V^T_{p \times p},
\]

where both \( U \) and \( V \) are orthogonal matrices, i.e. \( U^T U = U U^T = I \), and same for \( V \); and \( D_{n \times p} \) is a diagonal matrix.

• The diagonal elements of \( D_{n \times p} \) are of a decreasing order.
If we have to choose a rank-1 matrix $A$ to approximate $X_{n \times p}$, what would we do?

Turns out that the best choice is

$$U_1 d_{11} V_1^T,$$

where $U_1$ is the first column of $U$, $V_1$ is the first column of $V$, and $d_{11}$ is the first diagonal element of $D$.

In other words, $\|X - U_1 d_{11} V_1^T\|_2^2$ is minimized with this choice.

Hence, $U_1 d_{11} V_1^T$ is a rank-1 matrix that explained the variations in $X$ as much as possible.
Principle Component Analysis

- Let’s consider the sample covariance matrix $\hat{\Sigma} = \frac{X^T X}{n - 1}$, since $X$ is already centered.
- $\hat{\Sigma}$ can be diagonalize into
  \[ \hat{\Sigma} = V D^* V^T, \]
  where columns of $V$ are principle directions (loadings) and projecting $X$ on these loadings gives the principal components.
- On the other hand, based on SVD,
  \[ X = U D V^T, \]
  we can rewrite $\hat{\Sigma}$ as
  \[ \hat{\Sigma} = V D^T U^T U D V^T / (n - 1) = V \frac{D^2}{n - 1} V^T \]
So the right singular vectors $V$ of $X$ are just the principle directions, and the principal components are basically projecting each row (observation) of $X$ onto those directions:

$$XV = UDV^T V = UD$$

The first column of $U$ is the first PC, and $d_{11}$ is the variation along that direction, which is also the squared eigenvalue from SVD.

PCA should be performed by centering $X$ first (column-wise, i.e., by each variable).
• PCA is a **dimension reduction** tool, often used for visualization

• The leading PCs may display interesting information of the underlying (unobserved) clusters/manifold

• PCA is unsupervised, i.e., the directions does not necessarily reflect the relationship with the response (if there is any)
Self-organizing Maps
• Self-organizing Map (SOM) is another popular clustering method
• The main difference between SOM and k-means is that cluster means of a SOM has geometric relationship
• We will introduce the algorithm
• Please see R examples from course website
SOM Algorithm

- **Input data:** \( \{x_i\}_{i=1}^n \).

- **Initialize cluster means** (randomly): \( w_{ij}, i = 1, \ldots p, j = 1, \ldots q \). They are a grid of centers (the connected black dots). They are similar to the centers in a k-mean algorithm. However, they also preserves some geometric relationship.

- **Learning rate:** \( \alpha \in [0, 1] \). This controls how fast the \( w_{ij} \)'s are updated. It will decrease progressively.

- **Radius:** \( r \) that controls how many \( w_{ij} \)'s will be updated at each iteration. It will also decrease progressively.
SOM Algorithm

- For $k = 1, \ldots, n$, we will steam-in one new observation $x_k$ and perform the following update:
  - For all $w_{ij}$, calculate the distance between each $w_{ij}$ and $x_k$. Let $d_{ij} = \|x_k - w_{ij}\|$. By default, we use Euclidean distance.
  - Select the closest $w_{ij}$, denoted as $w_*$
  - Update each $w_{ij}$ based on the formula
    \[ w_{ij} = w_{ij} + \alpha h(w_*, w_{ij}, r) \|x_k - w_{ij}\| \]
  - Decrease the value of $\alpha$ and $r$

- In the ‘kohonen’ package, $\alpha$ starts at 0.05, and linearly decreases to 0.01, while $r$ is chosen to be 2/3 of all cluster means at the first iteration.
Spectral Clustering
In some applications, we do not have the value of each data point, instead, we have the similarities between data points.

Note: for similarity measures, larger means more similar, while for distance measures, larger is further away.

A nice way to represent the data is the **Similarity Graph** 

\[ G = (V, E) \] — an undirected graph

- \( V \) is a set of vertices: \( \{x_1, x_2, \ldots, x_n\} \)
- \( E \) is the set of edges: \( \{(i, j)\}_{i,j} \)
• For our case, this graph is weighted by an adjacency matrix

\[ W_{n \times n} = \{w_{ij}\}_{ij} \]

• You can define \( W \) in many different ways

• Each \( w_{ij} \) is the similarity between vertices \( i \) and \( j \)

• \( W \) is symmetric: \( w_{ij} = w_{ji} \)

• We also define the degree matrix \( D \) as a diagonal matrix

\[ \text{diag}(d_1, \ldots, d_n) \]

where the \( d_i \) is the degree of vertex \( i \):

\[ d_i = \sum_{j=1}^{n} w_{ij} \]
There are many different ways to define a graph Laplacian matrix. We give a few examples.

- **Unnormalized graph Laplacian**
  \[ L = D - W \]

- **Normalized graph Laplacians**
  \[ L_{\text{sym}} = D^{-1/2}LD^{-1/2} = I - D^{-1/2}WD^{-1/2} \]
  \[ L_{\text{rw}} = D^{-1}L = I - D^{-1}W \]

- Each of them have some unique properties.
For the Unnormalized graph Laplacian, we have, for any $f$

$$f'Lf = f'Df - f'Wf = \sum_{i=1}^{n} d_i f_i^2 - \sum_{i,j} f_i f_j w_{ij}$$

$$= \frac{1}{2} \left\{ \sum_i d_i f_i^2 - 2 \sum_{i,j} f_i f_j w_{ij} + \sum_j d_j f_j^2 \right\}$$

$$= \frac{1}{2} \sum_{ij} w_{ij} (f_i - f_j)^2$$

We also have:

- $L$ is positive semi-definite
- the smallest eigen-value is 0, with eigen-vector $1$
- The number of 0 eigen-values depends on the number of connected components
• For the normalized graph Laplacian, we have,

\[ f' L_{\text{sym}} f = \frac{1}{2} \sum_{ij} w_{ij} \left( \frac{f_i}{\sqrt{d_i}} - \frac{f_j}{\sqrt{d_i}} \right)^2 \]

• The eigenvalue \( \lambda \) and eigenvector \( v \) of \( L_{\text{rw}} \) can be solved from

\[ L v = \lambda D v \]

• The smallest eigenvalue of both \( L_{\text{sym}} \) and \( L_{\text{rw}} \) are 0, with
eigen-vectors \( \mathbf{1} \) and \( D^{1/2} \mathbf{1} \), respectively.
The spectral clustering is very simple:

- Construct a weighted adjacency matrix $W$, using e.g.,
  $$w_{ij} = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)$$

- Compute the Laplacian $L$ (or normalized versions $L_{\text{sym}}$, $L_{\text{rw}}$)
- Compute the smallest $k$ eigenvectors, denote them collectively as $V_{n \times k}$
- Treat $V_{n \times k}$ as the matrix of the observed data, and perform $k$-means clustering
- Output the $k$ cluster labels
Principal Coordinates Analysis
• Principal Coordinates Analysis (PCoA) is also called the classical multidimensional scaling (MDS)

• Given a similarity matrix $S_{n \times n} = \{s_{ij}\}_{ij}$, we want to construct low-dimensional ($k$) coordinates $z_1, z_2, \ldots, z_n \in \mathbb{R}^k$ such that the inner product of two subjects mimics the corresponding similarity

• More specifically, we minimize the strain function:

$$\text{Strain} = \sum_{ij} (s_{ij} - \langle z_i, z_j \rangle)^2$$

by solving low-dimensional coordinates $z_1, \ldots, z_n$. 
• PCoA can also be solved using eigen decomposition:
  • We first obtain the similarity matrix $S$
  • Let the double centered matrix $B = -\frac{1}{2}HDH$, with $H = I - \frac{1}{n}11^T$
  • Perform SVD on $B$ and obtain the largest $k$ eigenvector $U_{n \times k}$ and the eigenvalues on the diagonal of $D_{k \times k}$
  • The principle coordinate matrix is $UD^{1/2}$

• If the similarity matrix $S$ is in fact the centered inner product of the original data $X$, then PCoA is the same as PCA.