BIOS 835: Unsupervised Learning: Clustering

Alexander McLain

November 16, 2023

Clustering

- Clustering is an unsupervised learning technique that is one of the most natural exploratory data analysis (EDA) methods.
- **Goal:** arrange large quantities of data into groups
  - divide organisms into hierarchical orders
  - cluster patients for treatment diagnosis
- **Differences from classification**
  - we don't know the number of classes
  - we don't know anybody's class
  - the goal is more exploratory and less prediction
  - we can cluster observations (i.e., the Y_i's), the predictor variables (i.e., the $\boldsymbol{X}_i$'s), or both.

Definition

- Defining what a cluster is can be difficult and seems subjective.
- We might say that clusters are a *highly dense* region of a multivariate feature space.
- A natural way to think of clusters is as a mixture distribution.

$$p(\boldsymbol{X}|\Phi) = \sum_{k=1}^{K} I(\boldsymbol{X} \in \mathcal{C}_k) p_k(\boldsymbol{X}|\phi_k)$$

where the $f_k(\boldsymbol{X}|\phi_k)$ could be, for example, different multivariate normal distributions.

Full likelihood for cluster membership

- Suppose that $\mathcal{D} = \{\boldsymbol{x}_i, \boldsymbol{\delta}_i, i = 1, 2, \ldots, n\}$ is the complete data where $\boldsymbol{\delta}_i = \{\delta_{i1}, \ldots, \delta_{iK}\}$ where $\delta_{ik} = 1$ if $i \in \mathcal{C}_k$ and zero otherwise.
- Here, we may have $\Phi = (\phi_1, \phi_2, \ldots, \phi_K)$ where ϕ_k are the parameters for the kth distribution.
- For example, if each k follows a multivariate normal distribution we can have

$$\phi_k = (\boldsymbol{\mu}_k, \Sigma_k),$$

  but this is not always the case.
- The full likelihood is then

$$L(\Phi|\mathcal{D}) = p(\mathcal{D}|\Phi) = \prod_{i=1}^{n} \prod_{k=1}^{K} \{p_k(\boldsymbol{x}_i|\phi_k)\}^{\delta_{ik}}.$$

Full likelihood for cluster membership

- And the log-likelihood is

$$\ell(\Phi|\mathcal{D}) = \sum_{i=1}^{n}\sum_{k=1}^{K}\delta_{ik}\log\{p_k(\boldsymbol{x}_i|\phi_k)\}.$$

- However, $\boldsymbol{\delta}$ will not be known.
- Let $\mathcal{D} = \{\mathcal{D}_{obs}, \mathcal{D}_{mis}\}$,
  - $\mathcal{D}_{obs} = \{\boldsymbol{x}_i; i = 1, 2, \ldots, n\}$ and $\mathcal{D}_{mis} = \{\boldsymbol{\delta}_i; i = 1, 2, \ldots, n\}$
- The observed data likelihood is then

$$L_{obs}(\Phi|\mathcal{D}_{obs}) = \prod_{i=1}^{n}\int p(\mathcal{D}_i|\Phi)f(\mathcal{D}_{mis}|\boldsymbol{\pi})d\mathcal{D}_{mis}$$

Full likelihood for cluster membership

- In our situation, this would be

$$L_{obs}(\Phi|\mathcal{D}_{obs}) = \prod_{i=1}^{n}\sum_{k=1}^{K}\pi_k p_k(\boldsymbol{x}_i|\phi_k)$$

  where $\pi_k = \Pr(i \in \mathcal{C}_k)$ (note, this is not conditional on $\boldsymbol{x}_i$).
- We've added $\boldsymbol{\pi}$ to Φ (it could have been in the full likelihood as well).
- Optimization of this is very challenging with this model (e.g., identifiability issues, label switching, non-finite p_k, difficult derivatives).
- We'll seek other optimization options.

EM Algorithm

Basic idea of the approach:

1. Initialize the parameters at $\Phi^{(0)}$
2. For $m = 0, 1, 2, \ldots$ iterate between

E-step Compute

$$Q(\Phi|\Phi^{(m)}) = E\{\ell(\Phi|\mathcal{D})|\mathcal{D}_{obs}, \Phi^{(m)}\}$$

as a function of Φ where $\ell(\Phi|\mathcal{D}) = \log\{L(\Phi|\mathcal{D})\}$

M-step Find

$$\Phi^{(m+1)} = argmax_{\Phi}\{Q(\Phi|\Phi^{(m)})\}$$

3. Iterate through E and M steps until the parameters converge, i.e., $||\Phi^{(m)} - \Phi^{(m+1)}|| < \epsilon$.

EM Algorithm properties

- For our situation, the E- and M- steps can be simplified because

$$\begin{aligned} Q(\Phi|\Phi^{(m)}) &= E\{\ell(\Phi|\mathcal{D})|\mathcal{D}_{obs}, \Phi^{(m)}\} \\ &= \sum_{i=1}^{n} E\{\ell_i(\Phi|\mathcal{D}_i)|\mathcal{D}_{obs}, \Phi^{(m)}\} \\ &= \sum_{i=1}^{n}\sum_{k=1}^{K} E\{\ell_i(\Phi_k|\mathcal{D}_i)|\mathcal{D}_{obs}, \Phi^{(m)}\} \end{aligned}$$

where $\Phi_1, \Phi_2, \ldots, \Phi_K$ can be found separately such that

$$\phi_k^{(m+1)} = \text{argmax}\left[\sum_{i=1}^{n} E\{\ell_i(\Phi_k|\mathcal{D}_i)|\mathcal{D}_{obs}, \Phi^{(m)}\}\right]$$

EM Algorithm properties

- It can be shown that

$$\ell(\Phi^{(m+1)}|\mathcal{D}) - \ell(\Phi^{(m)}|\mathcal{D}) \geq Q(\Phi^{(m+1)}|\Phi^{(m+1)}) - Q(\Phi^{(m)}|\Phi^{(m)}) \geq 0$$

  thus, at each iteration the likelihood is non-decreasing (i.e., getting larger or not changing).
- The convergence properties of the EM algorithm are discussed in detail by McLachlan and Krishnan, they show that it's possible for the algorithm to converge to local minima or saddle points in some cases.
- The EM can be sensitive to starting values and be slow to converge.

EM Algorithm for finite mixtures

- As stated before, the log-likelihood for the ith subject is

$$\ell_i(\Phi|\mathcal{D}_i) = \sum_{k=1}^{K} \ell_i(\Phi_k|\mathcal{D}_i) = \sum_{k=1}^{K} \delta_{ik} \log\{p_k(\boldsymbol{x}_i|\phi_k)\}$$

and

$$E\{\ell_i(\Phi_k|\mathcal{D}_i)|\mathcal{D}_{obs}, \Phi^{(m)}\} = E(\delta_{ik}|\boldsymbol{X}_i, \Phi^{(m)}) \log\{p_k(\boldsymbol{X}_i|\phi_k)\}$$

- The marginal probabilities $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_K)$ are such that $\boldsymbol{\delta} \sim Multi(K, \boldsymbol{\pi})$
- Using Bayes rule

$$E(\delta_{ik}|\boldsymbol{X}_i, \Phi^{(m)}) = \delta_{ik}^{(m)} = \frac{\pi_k^{(m)} p_k(\boldsymbol{X}_i|\phi_k^{(m)})}{\sum_{\ell=1}^{K} \pi_\ell^{(m)} p_\ell(\boldsymbol{X}_i|\phi_k^{(m)})}$$

EM Algorithm for finite mixtures

$$
\begin{aligned}
Q(\Phi|\Phi^{(m)}) &= \sum_{i=1}^{n}\sum_{k=1}^{K} E\{\ell_i(\Phi_k|\mathcal{D}_i)|\mathcal{D}_{obs}, \Phi^{(m)}\} \\
&= \sum_{i=1}^{n}\sum_{k=1}^{K} \delta_{ik}^{(m)} \log\{p_k(\boldsymbol{X}_i|\phi_k)\}
\end{aligned}
$$

and when the ϕ_k are variation independent

$$
\phi_k^{(m+1)} = \text{argmax}_{\phi_k}\left[\sum_{i=1}^{n} \delta_{ik}^{(m)} \log\{p_k(\boldsymbol{X}_i|\phi_k)\}\right]
$$

which is a weighted likelihood.

EM Algorithm for finite mixtures

- Using the EM algorithm results in parameter estimates $\hat{\boldsymbol{\phi}} = \{\hat{\phi}_1, \ldots, \hat{\phi}_K\}$ and $\hat{\boldsymbol{\pi}}$.
- Further, we have estimates of the probability of cluster membership $\hat{\boldsymbol{\delta}}_i = (\hat{\delta}_{i1}, \ldots, \hat{\delta}_{iK})$ and
$$\hat{C}_k = \mathit{argmax}_k(\hat{\delta}_{ik})$$
- It is common to use the multivariate normal density as the mixing density. This results in:
  - the centroids (i.e., means) and covariances of all classes,
  - probability of class membership for each observation and class.

EM Algorithm for finite mixtures

- While the MVN is the most popular, this method can be used with any probabilistic model.
  - generalized linear models (e.g., logistic regression, Poisson regression, mixture of Poisson and logistic)
  - mixed models for longitudinal data.
- The EM for finite mixtures is a powerful approach for parametric clustering.
- There are methods to get the standard errors of the parameter (i.e., Louis' method), but they are harder than in standard MLE.

K-means algorithm

- The term "k-means" was first used by James MacQueen in 1967, though the idea goes back to Hugo Steinhaus in 1957.
- The standard algorithm was first proposed by Stuart Lloyd in 1957 as a technique for pulse-code modulation, though it wasn't published outside of Bell Labs until 1982.
- In 1965, E. W. Forgy published essentially the same method, which is why it is sometimes referred to as Lloyd-Forgy.

Dissimilarity

- We first need to choose a dissimilarity (or distance) metric to use.
- The dissimilarity between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ has the following properties
  1. $d(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq 0$
  2. $d(\boldsymbol{x}_i, \boldsymbol{x}_j) = 0 \leftrightarrow \boldsymbol{x}_i = \boldsymbol{x}_j$
  3. $d(\boldsymbol{x}_i, \boldsymbol{x}_j) = d(\boldsymbol{x}_j, \boldsymbol{x}_i)$

Dissimilarity

- We first need to choose a dissimilarity (or distance) metric to use.
- The dissimilarity between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ has the following properties
  1. $d(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq 0$
  2. $d(\boldsymbol{x}_i, \boldsymbol{x}_j) = 0 \leftrightarrow \boldsymbol{x}_i = \boldsymbol{x}_j$
  3. $d(\boldsymbol{x}_i, \boldsymbol{x}_j) = d(\boldsymbol{x}_j, \boldsymbol{x}_i)$
- **Examples**
  - Euclidean: $d(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sqrt{(\boldsymbol{x}_i - \boldsymbol{x}_j)'(\boldsymbol{x}_i - \boldsymbol{x}_j)}$
  - Manhattan: $d(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sum_{k=1}^{p} |x_{ik} - x_{jk}|$
  - $1-$correlation: $d(\boldsymbol{x}_i, \boldsymbol{x}_j) = 1 - \rho_{ij}$
- Let $\boldsymbol{D} = (d_{ij})_{i,j,=1,\ldots,n}$ where $d_{ij} = d(\boldsymbol{x}_i, \boldsymbol{x}_j)$ be the *proximity matrix*

Dissimilarity

- Overall, we can say that our goal is to find $\boldsymbol{\delta} = (\boldsymbol{\delta}_1, \ldots, \boldsymbol{\delta}_n)$ to minimize

$$W(\boldsymbol{\delta}) = \frac{1}{2} \sum_{k=1}^{K} \sum_{i=1}^{n} \sum_{j=1}^{n} I(\delta_{ik} = \delta_{jk} = 1) d(\boldsymbol{x}_i, \boldsymbol{x}_j) = \frac{1}{2} \sum_{k=1}^{K} \sum_{i=1}^{n} \sum_{j=1}^{n} \delta_{ik} \delta_{jk} d_{ij}$$

- Note that

$$\begin{aligned} T &= \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} \sum_{k=1}^{K} \delta_{ik} \{\delta_{jk} + (1 - \delta_{jk})\} \\ &= \frac{1}{2} \sum_{k=1}^{K} \sum_{i=1}^{n} \delta_{ik} \left(\sum_{j=1}^{n} \delta_{jk} d_{ij} + \sum_{j=1}^{n} (1 - \delta_{jk}) d_{ij} \right) \end{aligned}$$

is a constant.

Dissimilarity

- Where

$$\begin{aligned} T &= \frac{1}{2}\sum_{k=1}^{K}\sum_{i=1}^{n}\sum_{j=1}^{n}\delta_{ik}\delta_{jk}d_{ij} + \frac{1}{2}\sum_{k=1}^{K}\sum_{i=1}^{n}\sum_{j=1}^{n}\delta_{ik}(1-\delta_{jk})d_{ij} \\ &= W(\boldsymbol{\delta}) + B(\boldsymbol{\delta}) \end{aligned}$$

- Since T is a constant, minimizing $W(\boldsymbol{\delta})$ is equivalent to maximizing $B(\boldsymbol{\delta}) = T - W(\boldsymbol{\delta})$.
- T is the total point scatter (fixed), W is the within-cluster scatter, B is the between-cluster scatter.
- Unfortunately, minimizing W or equivalently maximizing B over all possible assignments of the n data points to K clusters is feasible only for very small data sets (for $n = 19$ and $K = 4$ there are 10^{10} combinations).

K-means Algorithm

- K-means looks to minimize

$$W_K(\boldsymbol{\delta}) = \frac{1}{2}\sum_{k=1}^{K}\sum_{i=1}^{n}\sum_{j=1}^{n}\delta_{ik}\delta_{jk}||\boldsymbol{x}_i - \boldsymbol{x}_j||^2 = \frac{1}{2}\sum_{k=1}^{K}N_k\sum_{i=1}^{n}\delta_{ik}||\boldsymbol{x}_i - m_k||^2$$

  where N_k is the number of points in the kth cluster, and m_k is the cluster center.
- Algorithm:
  1. Group each observation to the closest cluster center in **Euclinean distance**.
  2. Calculate the the K centers.
- Steps 1 and 2 are iterated until convergence (i.e., the cluster centers stop moving or the groups stop changing).
- This is similar to the EM for a MVN.

Choosing K

- For all of the clustering algorithms we need to specify K, the number of clusters in the given dataset.
- Let W_k be equal to $W(\boldsymbol{\delta})$ for a given algorithm when the number of clusters is k.
- We'd expect that

$$\{W_k - W_{k+1} | k < k^*\} \gg \{W_k - W_{k+1} | k \geq k^*\}$$

  where k^* is the true number of clusters.
- An estimate of K is then obtained by identifying a "kink" in the plot of W_K as a function of K.

Gap Statistic

- A method to determine if $W_k - W_{k+1}$ is "large" or "small" is the Gap Statistic.
- To use this method, we start by generating B datasets using a uniform distribution over a rectangle containing the data.
- Let W_k^{*b} be the within-cluster differences for the bth dataset when k clusters are used.
- From these we calculate

$$\log W_k^* = \frac{1}{B} \sum_{b=1}^{B} \log W_k^{*b}$$

and let s_k be the standard deviation of $\log W_k^{*b}$ values.

Gap Statistic

- We calculate gap statistic by

$$G(k) = \log W_k^* - \log W_k$$

- Note that

$$G(k+1) - G(k) = (\log W_k - \log W_{k+1}) - (\log W_k^* - \log W_{k+1}^*)$$

  where
  - $\log W_k - \log W_{k+1}$ will be "large" if $k < k^*$ and "small" if $k \geq k^*$
  - $\log W_k^* - \log W_{k+1}^*$ will always be "small" since it has only 1 true cluster (expected small difference).
- If $G(k+1) - G(k) > C$ for some $C > 0$ then we should use **at least** $k+1$ clusters, if it's not we should use k clusters.

Gap Statistic

- For C we'll use the standard deviation (s_k) calculated from the B datasets.
- The Gap estimate of k^* is the smallest k producing a gap within one standard deviation of the gap at $k+1$, i.e.

$$\begin{aligned} k^* &= argmin_k\{k|G(k+1) - G(k) < s'_{k+1}\} \\ &= argmin_k\{k|G(k) > G(k+1) - s'_{k+1}\} \end{aligned}$$

  where $s'_{k+1} = s_{k+1}\sqrt{1+1/B}$.
- Mostly $G(k+1) > G(k)$ but we can have $G(k+1) < G(k)$, which would definitely indicate k is better than $k+1$.

K-mediods

- The k-medoids or Partitioning Around Medoids (PAM) algorithm is a clustering algorithm reminiscent to the k-means algorithm.
- K-means requires all of the variables to be of the quantitative type and squared Euclidean distance places the highest influence on the largest distances.
- This causes the procedure to lack robustness against outliers that produce very large distances.
- What part of k-means algorithm uses Euclidean distance?
- If we didn't use the means of the centers, what could we use?

K-mediods

A general form of the k-medoid algorithm is:

1. For a given cluster assignment $\boldsymbol{\delta}$ find the observation in the each cluster minimizing total distance to other points in that cluster:

$$i_k^* = \underset{\{i:\delta_{ik}=1\}}{\text{argmin}} \sum_{\{j:\delta_{jk}=1\}} D(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

   then $m_k = x_{i_k^* k}$ are the current estimates of the cluster centers.

2. Given a current set of cluster centers $\{m_1, \ldots, m_K\}$, minimize the total error by assigning each observation to the closest (current) cluster center:

$$\delta_{ik} = 1 \text{ if } k = \text{argmin}_k D(\boldsymbol{x}_i, m_k)$$

3. Iterate steps 1 and 2 until the assignments do not change.

Hierarchical Clustering

- Hierarchical Clustering is a procedure that results in different degrees of clustering.
- At the top (or bottom) there will be only two clusters and at the bottom (or top) every observation will be in their own cluster.
- There are two ways to do this:
  1. **Agglomorative:** Bottom-up, each point starts as it's own cluster, finish with one cluster.
  2. **Divisive:** Top-down, everyone starts as one cluster and end up in their own cluster.
- These have similarities to forward-backward selection methods.
- For either method, observations are joined (or separated) by how dissimilar that are.

Hierarchical Clustering

Algorithm for agglomerative clustering:

1. Let $\mathcal{L} = \{\boldsymbol{x}_i; i = 1, 2, \ldots, n\}$ be the n-clusters
2. Find the smallest element of $\boldsymbol{D} = d_{IJ}$. Put I and J in the same cluster.

Now we have to recompute the dissimilarities for the new group IJ.

3. Compute dissimiarities $d_{IJ,k}$ for all $k \neq I$ or J. Three options for doing this:
   - **single**: $d_{IJ,k} = \min(d_{Ik}, d_{Jk})$
   - **complete**: $d_{IJ,k} = \max(d_{Ik}, d_{Jk})$
   - **average**: $d_{IJ,k} = avg(d_{Ik}, d_{Jk})$
4. For $\boldsymbol{D}^{(1)}$ which removes row I and J and includes the row calculated in 3.
5. Repeat steps 2–4 $n - 1$ times.

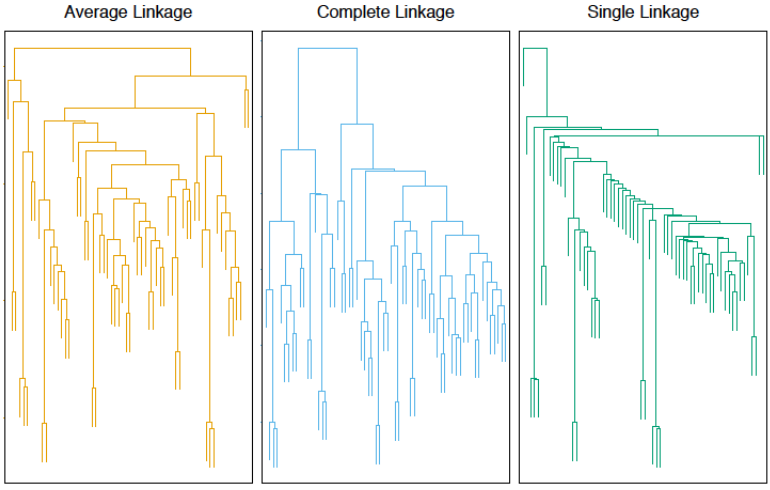

Figure: From page 524 in the online version of ESL.