ECE 20875
Python for Data Science
Milind Kulkarni and Chris Brinton
clustering
what is clustering?

- Given a set of data points, group them into clusters (subsets of the data set that are “similar”)

- What “similar” means depends on what clustering algorithm you use
  - K-means: Points are “near” each other
  - Gaussian mixture models: Points come from same Gaussian distribution (i.e., distribution with same mean and variance)

- Basic goal: Identify structure in data without any labels

- Lack of labels for the data points makes this an example of unsupervised learning
  - We will discuss supervised learning later
what is clustering?

• Given a set of data points, group them into **clusters** (subsets of the data set that are “similar”)

• What “similar” means depends on what clustering algorithm you use
  
  • K-means: Points are “near” each other
  
  • Gaussian mixture models: Points come from same Gaussian distribution (i.e., distribution with same mean and variance)

• Basic goal: Identify structure in data **without any labels**

• Lack of labels for the data points makes this an example of **unsupervised learning**
  
  • We will discuss supervised learning later
two approaches

• K-means
  • No model required
  • Can only find “simple” structure in data (points that are close together)

• Gaussian Mixture Models
  • Requires having a model in mind
  • Can find interesting structure in data (based on how complex the model is)

• Both models require initial assumption about number of clusters

• Both models use iterative algorithms to find clusters: Guess clusters, then refine them

• Both models can be sensitive to initial guess: Different initial guesses can lead to different clusters
k-means clustering

• A simpler approach than Gaussian Mixture Models

• No need for an *a priori* model

• But is less sophisticated: Clusters that k-means finds have limitations

• Formally, consider $n$ observations $x_1, x_2, \ldots, x_n$ of the data. *k-means* seeks to divide the dataset into $k$ clusters $\{S_1, S_2, \ldots, S_k\}$, i.e., assign each $x_j$ to a cluster $S_i$, according to

$$\arg\min_{S_1, \ldots, S_k} \sum_{i=1}^{k} \sum_{x \in S_i} ||x - \mu_i||^2$$

where $\mu_i$ is the mean of data points in $S_i$
k-means algorithm

- Start out by initializing $k$ “centroids” that define the clusters
- Random choice
k-means algorithm

- Start out by initializing k “centroids” that define the clusters

- **Assignment step**: Assign each data point to a cluster
  - Each data point is assigned to the cluster it is closest to
  - According to Euclidean distance, i.e., \( \arg\min_i | |x_j - S_i| | \)
k-means algorithm

• Start out by initializing k “centroids” that define the clusters

• **Assignment step**: Assign each data point to a cluster

  • Each data point is assigned to the cluster it is closest to

  • According to Euclidean distance, i.e., \( \arg \min_i ||x_j - S_i|| \)
k-means algorithm

• Start out by initializing k “centroids” that define the clusters

• **Assignment step**: Assign each data point to a cluster

• **Update step**: Move each centroid to the “middle” of its cluster
  
  • Compute the average position of the data points
  
  • Compute mean according to

\[
\mu_i = \frac{1}{|S_i|} \sum_{x \in S_i} x
\]
k-means algorithm

- Start out by initializing k “centroids” that define the clusters

- **Assignment step**: Assign each data point to a cluster

- **Update step**: Move each centroid to the “middle” of its cluster
  - Compute the average position of the data points
  - Compute mean according to
    \[
    \mu_i = \frac{1}{|S_i|} \sum_{x \in S_i} x
    \]
k-means algorithm

- Start out by initializing k “centroids” that define the clusters

- **Assignment step**: Assign each data point to a cluster

- **Update step**: Move each centroid to the “middle” of its cluster

- Repeat assignment step with new centroid locations
k-means algorithm

- Start out by initializing k “centroids” that define the clusters

- **Assignment step**: Assign each data point to a cluster

- **Update step**: Move each centroid to the “middle” of its cluster

- Repeat assignment step with new centroid locations
k-means algorithm

- Start out by initializing k “centroids” that define the clusters

  - **Assignment step**: Assign each data point to a cluster

  - **Update step**: Move each centroid to the “middle” of its cluster

- Repeat assignment step with new centroid locations

- Repeat update step with new clusters
k-means algorithm

• Start out by initializing k “centroids” that define the clusters

• **Assignment step**: Assign each data point to a cluster

• **Update step**: Move each centroid to the “middle” of its cluster

• Repeat assignment step with new centroid locations

• Repeat update step with new clusters
k-means clustering

• Start out by initializing k “centroids” that define the clusters

• **Assignment step**: Assign each data point to a cluster

• **Update step**: Move each centroid to the “middle” of its cluster

• Repeat assignment, update, assignment, update, … until convergence

• In Python: KMeans class from sklearn.cluster
Choosing $k$

- How do we know how many centroids to start with?

- Can pick a $k$ and see how far points in a cluster are from each centroid
  - If there are too few centroids, average distance is high
  - As $k$ increases, average distance drops
    - But drop “slows down” after a while
    - **Knee or elbow method**: Look for the “knee” of the curve

- Can also use cross validation!
  - Too many clusters: New data points are not well represented by the clusters
gaussian mixture models

• A Gaussian mixture model (GMM) with $k$ components (clusters) is a probability distribution that is a weighted sum of $k$ Gaussians:

$$p_X(x) = \sum_{i=1}^{k} \pi_i \mathcal{N}(x | \mu_i, \sigma_i^2)$$

- $\mu_i$ : mean of $i$th Gaussian
- $\sigma_i^2$ : variance of $i$th Gaussian
- $\pi_i$ : weight of $i$th Gaussian

Note: $\sum_{i=1}^{k} \pi_i = 1$
gaussian mixture models

• A Gaussian mixture model (GMM) with $k$ components (clusters) is a probability distribution that is a weighted sum of $k$ Gaussians:

$$p_X(x) = \sum_{i=1}^{k} \pi_i \mathcal{N}(x | \mu_i, \sigma_i^2)$$

• **Problem**: Given $N$ data points, how do we determine what the parameters of the $k$ Gaussians (and their weights) are that best fit the data?

• **Intuition**: Move the Gaussians around until their sum best fits the red curve (i.e., the dataset)
expectation maximization

- An iterative approach to finding the parameters of a statistical model, where the model depends on unobserved, latent variables
  - Here, our latent variables are the parameters of the Gaussians

- Start with a random guess for the Gaussians

- Compute **expectation** (E-step)
  - Given the current parameters, what is the likelihood that each point comes from a particular Gaussian?

- Perform **maximization** (M-step)
  - Given these new likelihoods (which are essentially weights), update the means, covariances, and weights of the Gaussians using weighted averages
E step

- For each data point $x_j$, compute the likelihood that the data point comes from Gaussian $i$'s random variable $G_i$:

$$
\gamma_{ij} = P(G_i | x_j) = \frac{P(x_j | G_i)P(G_i)}{P(x_j)}
$$

- $P(x_j | G_i)$ is the (conditional) probability of observing $x_j$ from $G_i$
- $P(G_i)$ is the (unconditional) probability of observing the Gaussian $G_i$
- $P(x_j)$ is the (unconditional) probability of observing $x_j$ (from any Gaussian)
E step

- For each data point $x_j$, compute the likelihood that the data point comes from Gaussian $i$'s random variable $G_i$:

$$
\gamma_{ij} = P(G_i \mid x_j) = \frac{P(x_j \mid G_i)P(G_i)}{P(x_j)}
$$
E step

- For each data point $x_j$, compute the likelihood that the data point comes from Gaussian $i$’s random variable $G_i$:

$$\gamma_{ij} = P(G_i | x_j) = \frac{P(x_j | G_i)P(G_i)}{P(x_j)}$$

Weight of this Gaussian
E step

- For each data point $x_j$, compute the likelihood that the data point comes from Gaussian $i$'s random variable $G_i$:

$$
\gamma_{ij} = P(G_i \mid x_j) = \frac{P(x_j \mid G_i)P(G_i)}{P(x_j)}
$$

Estimate from overall distribution
E step

- For each data point $x_j$, compute the likelihood that the data point comes from Gaussian $i$'s random variable $G_i$:

$$\gamma_{ij} = P(G_i \mid x_j) = \frac{P(x_j \mid G_i)P(G_i)}{P(x_j)}$$

$$= \frac{\pi_i \mathcal{N}(x_j \mid \mu_i, \sigma_i^2)}{\sum_{g=1}^{k} \pi_g \mathcal{N}(x_j \mid \mu_g, \sigma_g^2)}$$
• Now that we have the likelihoods for each data point (how likely each is to come from each Gaussian), we re-estimate the parameters of the Gaussian using those weights:

\[ N_i = \sum_{j=1}^{N} \gamma_{ij} \quad \pi_i' = \frac{N_i}{N} \quad \mu_i' = \frac{\sum_{j=1}^{N} \gamma_{ij} x_j}{N_i} \quad \sigma_i'^2 = \frac{\sum_{j=1}^{N} \gamma_{ij} (x_j - \mu_i')^2}{N_i} \]

• These expressions are the maximum likelihood estimators for Gaussian distributions

• Derived by setting the derivative of \( \log \prod_j p_X(x_j) \) to 0 for each parameter
Now that we have the likelihoods for each data point (how likely each is to come from each Gaussian), we re-estimate the parameters of each Gaussian $i$ using those weights:

- $N_i = \sum_{j=1}^{N} \gamma_{ij}$
  - Total likelihood of points in this Gaussian
- $\pi_i' = \frac{N_i}{N}$
  - Proportion of points that come from this Gaussian
- $\mu_i' = \frac{\sum_{j=1}^{N} \gamma_{ij} x_j}{N_i}$
  - Weighted mean of this Gaussian
- $\sigma_i'^2 = \frac{\sum_{j=1}^{N} \gamma_{ij} (x_j - \mu_i')^2}{N_i}$
  - Weighted variance of this Gaussian. Note that this uses the updated mean!
learning GMMs

• Repeat E and M steps until convergence

• Note that what you converge to can be sensitive to the initial estimates (like KMeans)

• When you are done, you have multiple Gaussians defined that “fit” the data you have

• This is a useful starting point for building Naïve Bayes classifiers!
  • We will discuss this next

parametric vs. non-parametric

- K-means vs. GMM is an instance of a distinction that shows up over and over when we build models:
  - A **parametric** approach starts with some assumptions about the underlying data (what kinds of distribution they have, for example)
  - A **non-parametric** approach makes no assumptions about the underlying data
- K-means is **non-parametric**: Do not assume that the data has any particular distribution (even though we do have one parameter, $k$)
- GMM is **parametric**: Approach assumes something about the structure of the data (that the clusters are normally distributed)
- What about other modeling techniques we’ve looked at?
  - **Regression?**