EECE-6822 Machine Learning

K-Means, GMMs

Cris Ababei
Dept. of Electrical and Computer Engineering
MARQUETTE UNIVERSITY
BE THE DIFFERENCE.

1

PART 1
K-Means clustering, GMMs

2

Unsupervised vs. Supervised learning

Although most of the applications of Machine Learning today are based on supervised learning (and as a result, this is where most of the investments go to), the vast majority of the available data is unlabeled: we have the input features **X**, but we do not have the labels **y**. The computer scientist Yann LeCun famously said that "if intelligence was a cake, unsupervised learning would be the cake, supervised learning would be the icing on the cake, and reinforcement learning would be the cherry on the cake." In other words, there is a huge potential in unsupervised learning that we have only barely started to sink our teeth into.

K-Means Clustering

Clustering

- Supervised learning used labeled data pairs $(\mathbf{x}, \mathrm{y})$ to learn a function $f : \mathrm{X} \rightarrow \mathrm{Y}$
  - But, what if we don't have labels?

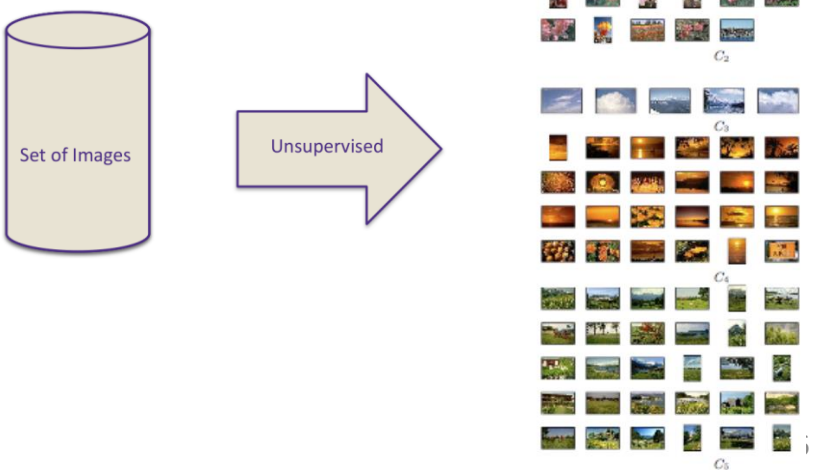
- No labels = **unsupervised learning**
- Only some points are labeled = **semi-supervised learning**
  - Labels may be expensive to obtain, so we only get a few

- **Clustering** is the unsupervised grouping of data points. It can be used for **knowledge discovery**.

5

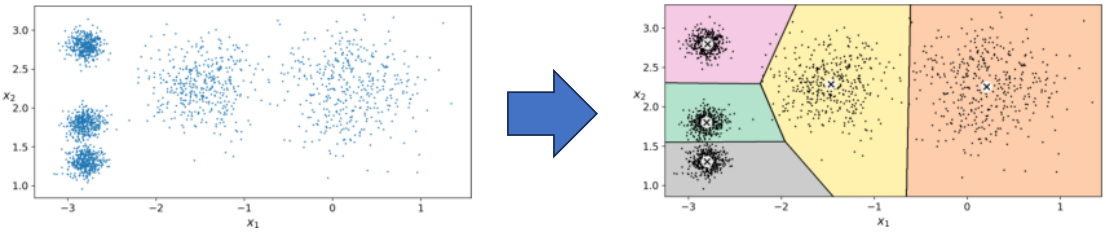
5

Clustering – a fundamental unsupervised task



6

K-means Idea



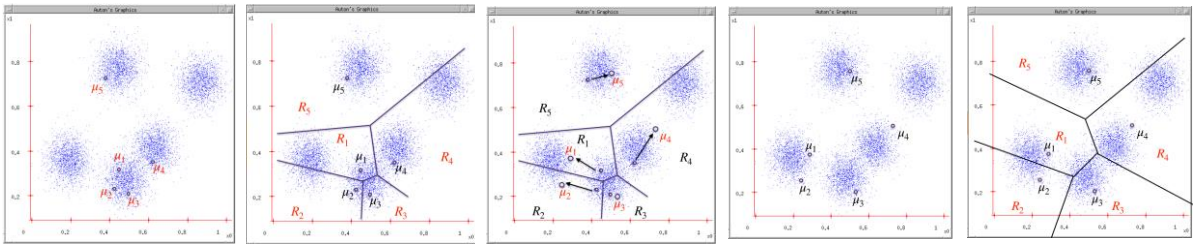
Consider the unlabeled dataset represented in Figure 9-2: you can clearly see five blobs of instances. The K-Means algorithm is a simple algorithm capable of clustering this kind of dataset very quickly and efficiently, often in just a few iterations. It was proposed by Stuart Lloyd at Bell Labs in 1957 as a technique for pulse-code modulation, but it was only published outside of the company in 1982.[1] In 1965, Edward W. Forgy had published virtually the same algorithm, so K-Means is sometimes referred to as Lloyd–Forgy.

[***B3-Geron***] Aurelien Geron, Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, O'Reilly, 2022.

7

7

K-means Algorithm



1. Ask user how many clusters they'd like. (e.g. k=5)
2. Randomly guess k cluster Center locations $\{\mu_1, \ldots, \mu_5\}$

1. Ask user how many clusters they'd like. (e.g. k=5)
2. Randomly guess k cluster Center locations $\{\mu_1, \ldots, \mu_5\}$
3. Each datapoint finds out which Center it's closest to. (Thus each Center "owns" a set of datapoints)

1. Ask user how many clusters they'd like. (e.g. k=5)
2. Randomly guess k cluster Center locations $\{\mu_1, \ldots, \mu_5\}$
3. Each datapoint finds out which Center it's closest to.
4. Each Center finds a new centroid of the points it owns

1. Ask user how many clusters they'd like. (e.g. k=5)
2. Randomly guess k cluster Center locations $\{\mu_1, \ldots, \mu_5\}$
3. Each datapoint finds out which Center it's closest to.
4. Each Center finds a new centroid of the points it owns...
5. ...and jumps there
6. ...Repeat until terminated!

1. Ask user how many clusters they'd like. (e.g. k=5)
2. Randomly guess k cluster Center locations $\{\mu_1, \ldots, \mu_5\}$
3. Each datapoint finds out which Center it's closest to.
4. Each Center finds the centroid of the points it owns...
5. ...and jumps there
6. ...Repeat until terminated!

8

8

K-means Algorithm

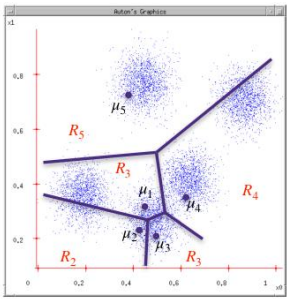
1. Choose k, how many clusters to find
2. Randomly initialize k centers
   - $\mu^{(0)} = [\mu_1^{(0)}, \ldots, \mu_k^{(0)}] \in \mathbb{R}^{d \times k}$
   - Usually randomly chosen from the data points, to make sure they are in the right domain
- For t=0,1,2,… repeat
3. Assign each point j∈{1,…,n} to its nearest center:

$$C_j^{(t)} \leftarrow \arg\min_{i \in \{1,2,\ldots,k\}} \|x_j - \mu_i^{(t)}\|^2$$

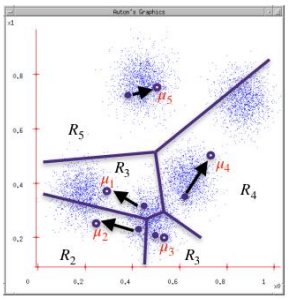
4. Recenter: μ_i becomes centroid of its point:

$$\mu_i^{(t+1)} \leftarrow \arg\min_{\mu} \sum_{j:C_j^{(t)}=i} \|\mu - x_j\|^2$$

- Equivalent to $\mu_i^{(t+1)} \leftarrow$ average of all the points assigned to $\mu_i^{(t)}$



Recenter:



9

9

Does k-means converge?

- k-means is trying to minimize the following objective

$$\min_{\{\mu_i\}_{i=1}^k} \min_{\{C_\ell\}_{\ell=1}^n} \sum_{j=1}^{n} \|x_j - \mu_{C_j}\|^2$$

via alternating minimization
(equivalent to coordinate descent)

- Fix μ, optimize C
- Fix C, optimize μ

- Does this converge? Does this terminate in finite time?

10

10

Does k-means converge?

- there is only a finite set of values that $\{C_j\}_{j=1}^{n} \in \{1,\ldots,k\}^n$ can take (k^n is large but finite)
- so there is only finite, k^n at most, values for cluster-centers, $\{\mu_i\}_{i=1}^{k}$, also
- each time we update them, we will never increase the objective function:

$$\sum_{j=1}^{n} \|x_j - \mu_{C_j}\|_2^2$$

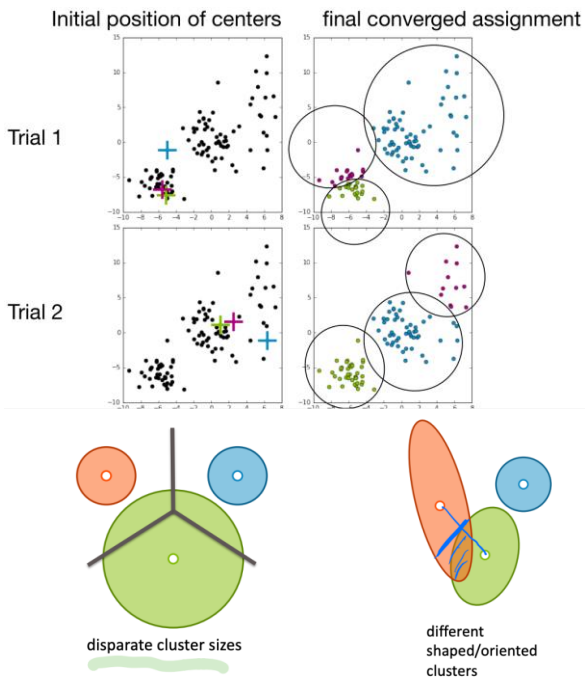
- the objective is lower bounded by zero
- after at most k^n steps, the algorithm must converge (as the assignments $\{C_j\}_{j=1}^{n}$ cannot return to previous assignments in the course of k-means iterations)

11

11

Downsides of k-means

1. Requires the number of clusters to be specified by us
2. Final solution depends on the initialization (does not necessarily find global minimum of the objective)
3. Does not behave well when clusters have varying sizes, different densities or nonspherical shapes



12

Finding the optimal number of clusters k

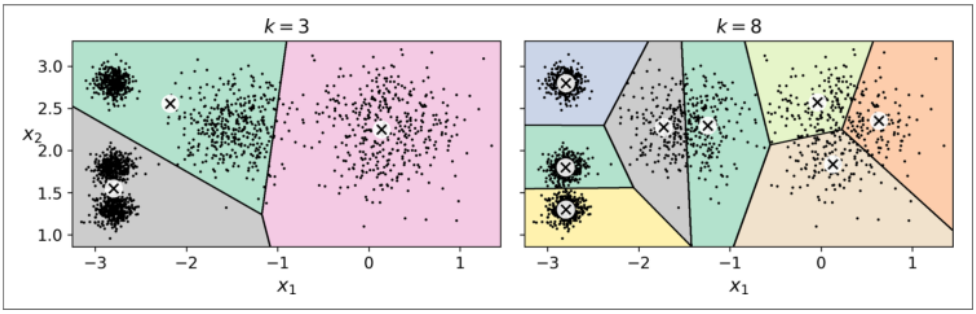


Figure 9-7. Bad choices for the number of clusters: when k is too small, separate clusters get merged (left), and when k is too large, some clusters get chopped into multiple pieces (right)

Read more at: https://en.wikipedia.org/wiki/Determining_the_number_of_clusters_in_a_data_set

13

13

Centroid initialization methods

If you happen to know approximately where the centroids should be (e.g., if you ran another clustering algorithm earlier), then you can set the `init` hyperparameter to a NumPy array containing the list of centroids, and set `n_init` to `1`:

```
good_init = np.array([[-3, 3], [-3, 2], [-3, 1], [-1, 2], [0, 2]])
kmeans = KMeans(n_clusters=5, init=good_init, n_init=1)
```

Another solution is to run the algorithm multiple times with different random initializations and keep the best solution. The number of random initializations is controlled by the `n_init` hyperparameter: by default, it is equal to `10`, which means that the whole algorithm described earlier runs 10 times when you call `fit()`, and Scikit-Learn keeps the best solution. But how exactly does it know which solution is the best? It uses a performance metric! That metric is called the model's *inertia*, which is the mean squared distance between each instance and its closest centroid. It is

14

14

Plotting inertia as a function of the number of clusters k

Model's **inertia** is the mean squared distance between each instance and its closest centroid.

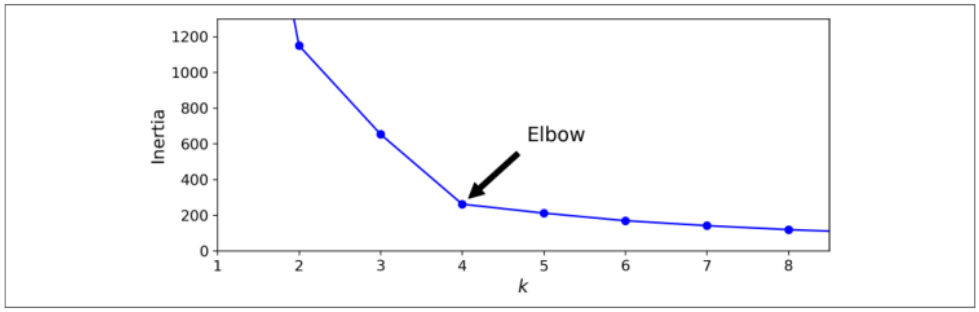


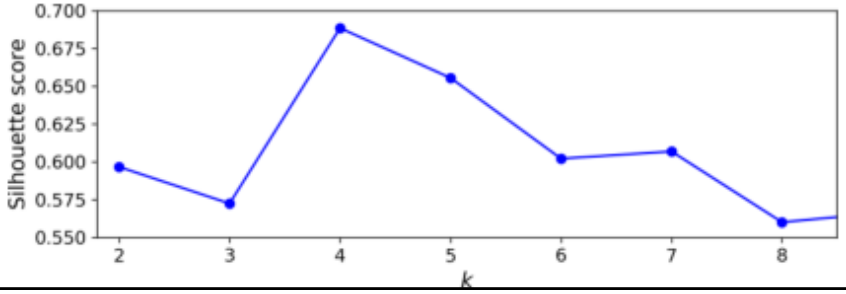
Figure 9-8. When plotting the inertia as a function of the number of clusters k, the curve often contains an inflexion point called the "elbow"

15

15

Silhouette score - the mean silhouette coefficient over all the instances

instance's silhouette coefficient is equal to $(b - a) / \max(a, b)$, where a is the mean distance to the other instances in the same cluster (i.e., the mean intra-cluster distance) and b is the mean nearest-cluster distance (i.e., the mean distance to the instances of the next closest cluster, defined as the one that minimizes b, excluding the instance's own cluster). The silhouette coefficient can vary between −1 and +1. A coefficient close to +1 means that the instance is well inside its own cluster and far from other clusters, while a coefficient close to 0 means that it is close to a cluster boundary, and finally a coefficient close to −1 means that the instance may have been assigned to the wrong cluster.



16

16

Use Example 1: Using Clustering for **Image Segmentation**

Figure 9-12. Image segmentation using K-Means with various numbers of color clusters

[***B3-Geron**] Aurelien Geron, Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, O'Reilly, 2022.

17

17

Use Example 2: Using Clustering for **Semi-Supervised Learning**

- We have plenty of unlabeled instances and very few labeled instances
- MNIST digits dataset: assume only 50 labeled instances are labeled
  - Train a Logistic Regression model on the sample of 50 labeled instances
  - **The accuracy is just 83.3%**
- Better approach:
  - First, cluster whole training set into 50 clusters using k-means
  - Then for each cluster, find the image closest to the centroid; call these images the representative images; label them manually
  - Train a Logistic Regression model on the new representative 50 labeled images
  - **The accuracy jumps from 83.3% accuracy to 92.2%,** although we are still only training the model on 50 instances.

Geron's GitHub: https://github.com/ageron/handson-ml3/blob/main/09_unsupervised_learning.ipynb

18

18

For fun

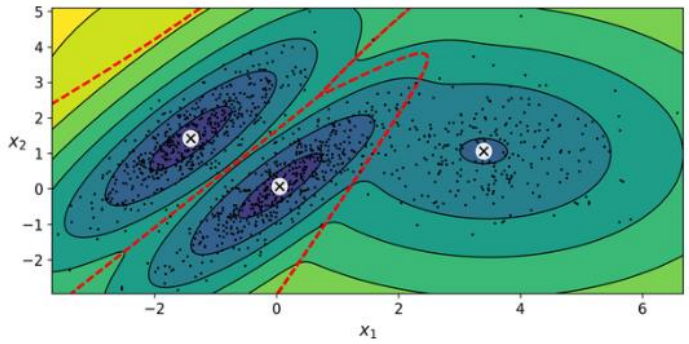
- https://www.naftaliharris.com/blog/visualizing-k-means-clustering/
- http://alekseynp.com/viz/k-means.html
- https://maayanlab.cloud/clustergrammer/
- https://user.ceng.metu.edu.tr/~akifakkus/courses/ceng574/k-means/
- https://www.philippe-fournier-viger.com/tools/kmeans_demo.php

Gaussian Mixture Models

Extension of k-means

Gaussian Mixture Model (GMM)

- A **Gaussian Mixture Model (GMM)** is a probabilistic model that assumes that the instances were generated from a mixture of several Gaussian distributions whose parameters are unknown.



See demo: https://lukapopijac.github.io/gaussian-mixture-model

21

21

Gaussian Mixture Model

- input: data $\{x_i\}_{i=1}^n$ in $\mathbb{R}^d$
- parameters of a **Gaussian Mixture Model**
  - mixing weights:
    - $\pi_j = \mathbf{P}(\text{cluster membership} = j)$ for $j \in \{1,\ldots,K\}$
  - means:
    - $\mu_j \in \mathbb{R}^d$ for $j \in \{1,\ldots,K\}$
  - covariance matrices:
    - $\mathbf{\Sigma}_j \in \mathbb{R}^{d\times d}$ for $j \in \{1,\ldots,K\}$
- we suppose that the given data has been generated from a GMM, and try to find the best GMM parameters (this naturally will define clustering of the training data)

- under the GMM, the i-th sample is drawn as follows
  - first sample a cluster $z_i \in \{1,\ldots,K\}$, from $\pi = [\pi_1, \ldots, \pi_K]$
  - conditioned on this cluster, x_i is sampled from

$$x_i \sim N(\mu_{z_i}, \mathbf{\Sigma}_{z_i})$$

Probability to pick a cluster is the cluster's weight. The index of the cluster is z_i

22

22

Maximum Likelihood Estimation (MLE)

- we can find the best GMM for given data by MLE
- for simplicity, suppose $d = 1$ and $K = 2$
- Model parameters are $\pi_1, \pi_2, \mu_1, \mu_2, \sigma_1^2, \sigma_2^2 \in \mathbb{R}$
- the probability of observing a sample x_i can be written as

$$\mathbf{P}(x_i; \pi_1, \pi_2, \mu_1, \mu_2, \sigma_1^2, \sigma_2^2) = \underbrace{\pi_1 \frac{1}{\sqrt{2\pi\sigma_1^2}} e^{-\frac{(x_i-\mu_1)^2}{2\sigma_1^2}}}_{\triangleq N(x_i;\mu_1,\sigma_1^2)} + \underbrace{\pi_2 \frac{1}{\sqrt{2\pi\sigma_2^2}} e^{-\frac{(x_i-\mu_2)^2}{2\sigma_2^2}}}_{\triangleq N(x_i;\mu_2,\sigma_2^2)}$$

- MLE tries to find

$$\arg \max_{\pi_1,\pi_2,\mu_1,\mu_2,\mathbf{C}_1,\mathbf{C}_2} \sum_{i=1}^{n} \log \mathbf{P}(x_i; \pi_1, \pi_2, \mu_1, \mu_2, \sigma_1^2, \sigma_2^2)$$

- however, unlike least squared or logistic regression, this is not a concave function of the parameters (thus hard to find the optimal solution)
- in general, MLE of a mixture model is not convex/concave optimization

Recall MLE from Lecture 1: fitting a single Gaussian model

- given $\{x_i\}_{i=1}^n \in \mathbb{R}$,fit the best Gaussian model with mean $\mu \in \mathbb{R}$ and variance $\sigma^2 \in \mathbb{R}$
- using MLE we want to solve

$$\text{maximize}_{\mu,\sigma^2} \ \mathscr{L}(\mu, \sigma^2) = \sum_{i=1}^{n} \underbrace{\left(-\frac{(x_i-\mu)^2}{2\sigma^2} - \log\left(\sqrt{2\pi\sigma^2}\right) \right)}_{\log N(x_i|\mu,\sigma^2)}$$

- we compute gradient and set it to zero:
  - $\nabla_\mu \mathscr{L}(\mu, \sigma^2) = \frac{1}{\sigma^2} \sum_{i=1}^{n} (\mu - x_i)$

    which is zero for $\mu = \frac{1}{n} \sum_{i=1}^{n} x_i$

    (which makes sense as it is the empirical mean)
  - $\nabla_{\sigma^2} \mathscr{L}(\mu, \sigma^2) = \frac{\sum_{i=1}^{n} (x_i - \mu)^2}{2(\sigma^2)^2} - \frac{n}{2\sigma^2}$

    which is zero for $\sigma^2 = \frac{1}{n} \sum_{i=1}^{n} (x_i - \mu)^2$

    (which makes sense as it is the empirical variance)

MLE for GMM

- we want to fit a model by solving

$$\max_{\pi_1,\pi_2,\mu_1,\mu_2,\sigma_1^2,\sigma_2^2} \sum_{i=1}^{n} \log\Big(\pi_1 \underbrace{\frac{1}{\sqrt{2\pi\sigma_1^2}} e^{-\frac{(x_i-\mu_1)^2}{2\sigma_1^2}}}_{\triangleq N(x_i;\mu_1,\sigma_1^2)} + \pi_2 \underbrace{\frac{1}{\sqrt{2\pi\sigma_2^2}} e^{-\frac{(x_i-\mu_2)^2}{2\sigma_2^2}}}_{\triangleq N(x_i;\mu_2,\sigma_2^2)}\Big)$$

- define $r_i = \mathbf{P}(z_i = 1 \mid x_i) = \dfrac{\mathbf{P}(z_i = 1, x_i)}{\mathbf{P}(z_i = 1, x_i) + \mathbf{P}(z_i = 2, x_i)}$ *i=1..n*

$$= \frac{\pi_1 N(x_i;\mu_1,\sigma_1^2)}{\pi_1 N(x_i;\mu_1,\sigma_1^2) + \pi_2 N(x_i;\mu_2,\sigma_2^2)}$$

- setting the gradient to zero, we get
  - $\pi_1 = \dfrac{N_1}{n}$ where $N_1 = \sum_{i=1}^{n} r_i$, and $\pi_2 = \dfrac{N_2}{n}$ where $N_2 = \sum_{i=1}^{n} (1 - r_i)$
  - $\mu_1 = \dfrac{1}{N_1} \sum_{i=1}^{n} r_i x_i$ and $\mu_2 = \dfrac{1}{N_2} \sum_{i=1}^{n} (1 - r_i) x_i$
  - $\sigma_1^2 = \dfrac{1}{N_1} \sum_{i=1}^{n} r_i (x_i - \mu_1)^2$ and $\sigma_2^2 = \dfrac{1}{N_2} \sum_{i=1}^{n} (1 - r_i)(x_i - \mu_2)^2$
- both LHS and RHS depend on the parameters, and no closed form solution exists
- **note that if we know r_i's it is trivial to compute parameters, and vice versa**

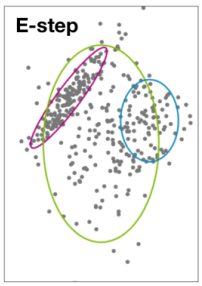
25

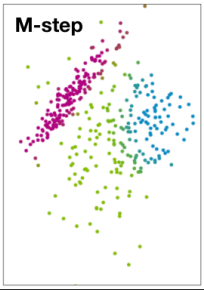
25

Expectation Maximization (EM) algorithm to approximate the solution of MLE

- EM is a popular method to solve MLE for mixture models
- input: training data $\{x_i\}_{i=1}^n$
- output: $\pi_1, \pi_2, \mu_1, \mu_2, \sigma_1^2, \sigma_2^2 \in \mathbb{R}$

- initialization: randomly initialize the parameters
- repeat
  - **E-step** (Expectation): parameters → soft membership
    - $r_i = \dfrac{\pi_1 N(x_i;\mu_1,\sigma_1^2)}{\pi_1 N(x_i;\mu_1,\sigma_1^2) + \pi_2 N(x_i;\mu_2,\sigma_2^2)}$ for all $i \in \{1,2,\ldots,n\}$
  - **M-step** (Maximization): soft membership → parameters
    - $\pi_1 = \dfrac{N_1}{n}$ where $N_1 = \sum_{i=1}^{n} r_i$, and $\pi_2 = \dfrac{N_2}{n}$ where $N_2 = \sum_{i=1}^{n} (1 - r_i)$
    - $\mu_1 = \dfrac{1}{N_1} \sum_{i=1}^{n} r_i x_i$ and $\mu_2 = \dfrac{1}{N_2} \sum_{i=1}^{n} (1 - r_i) x_i$
    - $\sigma_1^2 = \dfrac{1}{N_1} \sum_{i=1}^{n} r_i (x_i - \mu_1)^2$ and $\sigma_2^2 = \dfrac{1}{N_2} \sum_{i=1}^{n} (1 - r_i)(x_i - \mu_2)^2$





26

26

For general number of clusters *K* and dimension *d*

- we can derive EM for general case, in an analogous way
- Initialize parameters: $\pi_1, \ldots, \pi_K, \mu_1, \ldots, \mu_K, \Sigma_1, \ldots, \Sigma_K$

- **E-step**:
  - For k=1,…,K

$$r_{i,k} = \frac{\pi_k \, N(x_i \mid \mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \, N(x_i \mid \mu_j, \Sigma_j)}$$

- **M-step**:
  - For k=1,..,K

$$\pi_k = \frac{N_k}{n} \quad \text{where} \quad N_k = \frac{\sum_{i=1}^{n} r_{i,k}}{n}$$

$$\mu_k = \frac{1}{N_k} \sum_{i=1}^{n} r_{i,k} x_i \quad \text{and} \quad \Sigma_k = \frac{1}{N_k} \sum_{i=1}^{n} r_{i,k} (x_i - \mu_k)(x_i - \mu_k)^T$$

- **once GMM is learned, clustering is straight forward** **cluster according to the $r_{i,k}$'s**

27

27

Selecting the Number of Clusters

- Find the model that minimizes a theoretical information criterion, such as
  - the **Bayesian information criterion (BIC)** or
  - the **Akaike information criterion (AIC)**

Equation 9-1. Bayesian information criterion (BIC) and Akaike information criterion (AIC)

$$BIC = \log(m)p - 2\log(\hat{L})$$

$$AIC = 2p - 2\log(\hat{L})$$

In these equations:

- m is the number of instances, as always.
- p is the number of parameters learned by the model.
- $\hat{L}$ is the maximized value of the *likelihood function* of the model.

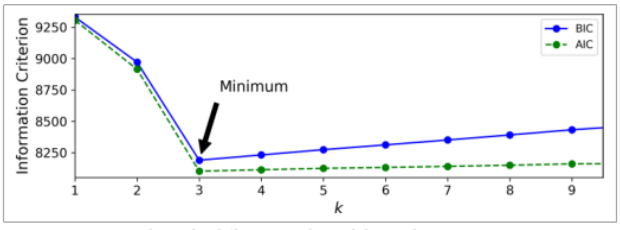


Figure 9-21. AIC and BIC for different numbers of clusters k

28

28

Bayesian Gaussian Mixture Models

Rather than manually searching for the optimal number of clusters, you can use the `BayesianGaussianMixture` class, which is capable of giving weights equal (or close) to zero to unnecessary clusters. Set the number of clusters `n_components` to a value that you have good reason to believe is greater than the optimal number of clusters (this assumes some minimal knowledge about the problem at hand), and the algorithm will eliminate the unnecessary clusters automatically. For example, let's set the number of clusters to 10 and see what happens:

```
>>> from sklearn.mixture import BayesianGaussianMixture
>>> bgm = BayesianGaussianMixture(n_components=10, n_init=10)
>>> bgm.fit(X)
>>> np.round(bgm.weights_, 2)
array([0.4 , 0.21, 0.4 , 0.  , 0.  , 0.  , 0.  , 0.  , 0.  , 0.  ])
```

29

29

Example: Anomaly Detection Using Gaussian Mixtures

Anomaly detection (also called *outlier detection*) is the task of detecting instances that deviate strongly from the norm. These instances are called *anomalies*, or *outliers*, while the normal instances are called *inliers*. Anomaly detection is useful in a wide variety of applications, such as fraud detection, detecting defective products in manufacturing, or removing outliers from a dataset before training another model (which can significantly improve the performance of the resulting model).

Using a Gaussian mixture model for anomaly detection is quite simple: any instance located in a low-density region can be considered an anomaly. You must define what density threshold you want to use. For example, in a manufacturing company that tries to detect defective products, the ratio of defective products is usually well known. Say it is equal to 4%. You then set the density threshold to be the value that results in having 4% of the instances located in areas below that threshold density. If

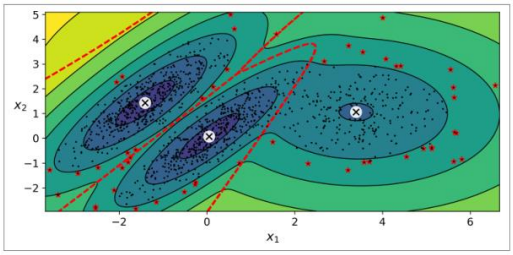
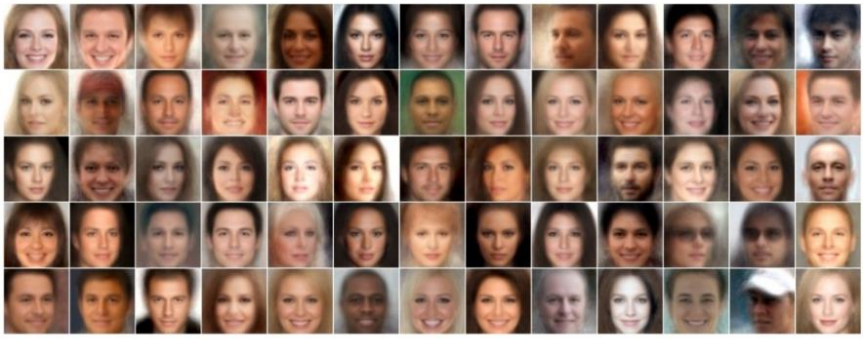


Figure 9-19. Anomaly detection using a Gaussian mixture model

30

30

Example: GMM for real data



- these are generated samples, from GMM trained on CelebA dataset
- image: 64*64*3=288 dimension
- covariance: restricted to rank-10 matrices
- mixture: K=1,000

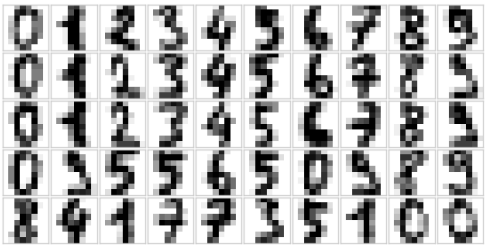
[1] Eitan Richardson and Yair Weiss. 2018. On GANs and GMMs. In Proceedings of the 32nd International Conference on Neural Information Processing Systems (NIPS'18). Curran Associates Inc., Red Hook, NY, USA, 5852–5863.

31

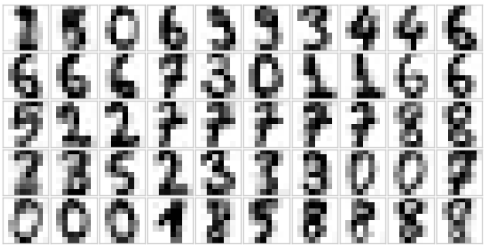
31

Summary

- Though the GMM is often categorized as a **clustering algorithm**, fundamentally it is an algorithm for ***density estimation***.
- That is to say, the result of a GMM fit to some data is technically not a clustering model, but a generative probabilistic model describing the distribution of the data.
- GMM gives us the recipe to generate new random data distributed similarly to our input.



Original



Generated

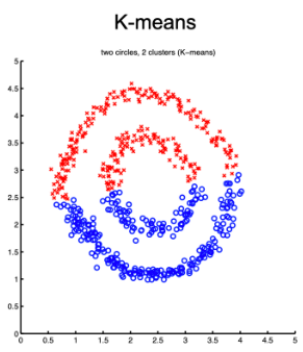
32

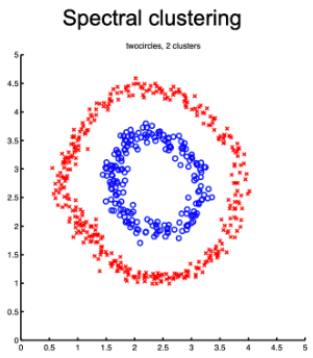
32

Spectral Clustering

Spectral Clustering

- Main idea:
  - Transform the dataset into a graph
  - Use eigenvalues (also called spectrum) and vectors of a graph to cluster





k-means and GMMs are inherently linear

[Shi,Malik,'00],[Ng,Jordan,Weiss,'01]

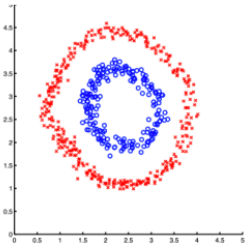
Step 1. From dataset to a graph

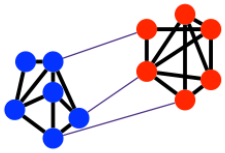
- Given $\mathcal{D} = \{x_i \in \mathbb{R}^d\}_{i=1}^n$, create a graph with n nodes and weighted edges $\{w_{ij}\}$, where each node represents each sample and each edge measures the similarity between the two nodes
  - Example 1: Gaussian kernel

$$w_{ij} = e^{-\frac{\|x_i - x_j\|_2^2}{\sigma^2}}$$

  - Example 2: k-nearest neighbor graph

$w_{ij} = 1$ if j is one of k-nearest neighbors of i or
i is one of k-nearest neighbors of j





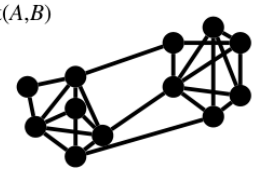
35

35

Step 2. Graph partitioning

- Once we have a similarity graph, how do we partition it?
- Can we use **minimum cut** for a graph $G(V, E)$?
  - Set of nodes $V = \{1, \ldots, n\}$
  - Set of edges $E = \{(i, j)\}$
  - If it is a weighted graph we have weights $\{w_{ij}\}_{(i,j)\in E}$
- **Minimum cut** of a graph is a partition $A \cup B = V$ and $A \cap B = \varnothing$ such that

$$\arg\min_{A,B} \underbrace{\sum_{i\in A}\sum_{j\in B} w_{i,j}}_{\text{cut}(A,B)}$$

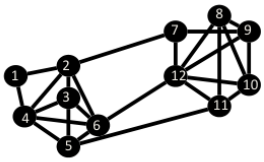


36

36

Step 2. Graph partition using Graph Laplacian

- Definitions (we will define it for unweighted graphs, but everything naturally generalizes to weighted graphs)
  - **Adjacency matrix** of a graph $A \in \mathbb{R}^{n \times n}$
    $A_{ij} = 1$ if $(i, j) \in E$
    0 otherwise
  - **Degree** of a node i, is $d_i = \sum_{j=1}^{n} A_{ij}$, which is number of edges connected to node i
  - Define $D \in \mathbb{R}^{n \times n}$ as a diagonal matrix with the degrees of each node in the diagonal
  - The **Graph Laplacian** of a graph is defined as
    $L_G = D - A$

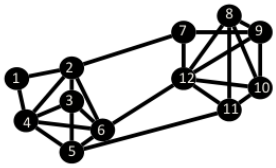


$$A = \begin{bmatrix} 0&1&0&1&0&0&0&0&0&0&0&0 \\ 1&0&1&1&0&1&1&0&0&0&0&0 \\ 0&1&0&1&1&1&1&0&0&0&0&0 \\ 1&1&1&0&1&1&1&0&0&0&0&0 \\ 0&0&1&1&0&1&1&0&0&0&0&1 \\ 0&1&1&1&1&0&0&0&0&0&0&1 \\ 0&1&0&0&0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&0&0&1&1&1&1 \\ 0&0&0&0&0&0&0&1&1&0&1&0 \\ 0&0&0&0&0&0&0&1&1&0&1&1 \\ 0&0&0&0&0&1&0&0&1&0&1&0 \\ 0&0&0&0&0&1&1&1&1&1&1&0 \end{bmatrix}$$

$$D = \begin{bmatrix} 2&0&0&0&0&0&0&0&0&0&0&0 \\ 0&5&0&0&0&0&0&0&0&0&0&0 \\ 0&0&4&0&0&0&0&0&0&0&0&0 \\ 0&0&0&5&0&0&0&0&0&0&0&0 \\ 0&0&0&0&3&0&0&0&0&0&0&0 \\ 0&0&0&0&0&5&0&0&0&0&0&0 \\ 0&0&0&0&0&0&3&0&0&0&0&0 \\ 0&0&0&0&0&0&0&3&0&0&0&0 \\ 0&0&0&0&0&0&0&0&4&0&0&0 \\ 0&0&0&0&0&0&0&0&0&4&0&0 \\ 0&0&0&0&0&0&0&0&0&0&4&0 \\ 0&0&0&0&0&0&0&0&0&0&0&6 \end{bmatrix}$$

Step 2. Graph partition using Graph Laplacian

- Graph Laplacian $L_G = D - A$ can capture some structure of the graph
- Consider placing each node in 1-dim line at positions $x = [x_1, x_2, \ldots, x_2]$

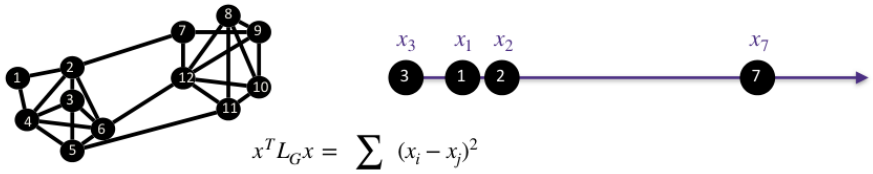


quadratic form of L_G is useful in capturing the structure of the graph:

$$\begin{aligned} x^T L_G x &= \sum_i d_i x_i^2 - \sum_{(i,j)\in E} 2x_i x_j \\ &= \sum_i \sum_{j:(i,j)\in E} x_i^2 - \sum_{(i,j)\in E} 2x_i x_j \\ &= \sum_{(i,j)\in E} 2x_i^2 - 2x_i x_j \\ &= \sum_{(i,j)\in E} x_i^2 + x_j^2 - 2x_i x_j \\ &= \sum_{(i,j)\in E} (x_i - x_j)^2 \end{aligned}$$

Step 2. Graph partition using Graph Laplacian

- Graph Laplacian $L_G = D - A$ can capture some structure of the graph
- Consider placing each node in 1-dim line at positions $x = [x_1, x_2, \ldots, x_2]$



$$x^T L_G x = \sum_{(i,j)\in E} (x_i - x_j)^2$$

- If we want a good graph partition, we want to place nodes such that the distance between connected nodes are smaller
- This naturally leads to the following problem:

$$\arg\min_{x\in\mathbb{R}^n} x^T L_G x = \sum_{(i,j)\in E} (x_i - x_j)^2$$

- There is a trivial solution to this problem: $x_i = 1$ for all i, which achieves the minimum value of zero, so we change it to

$$\arg\min_{x\in\mathbb{R}^n} x^T L_G x = \sum_{(i,j)\in E} (x_i - x_j)^2 \qquad \text{subject to } x^T \mathbf{1} = 0$$

39

39

Step 2. Graph partition using Graph Laplacian

- To solve graph partitioning, we solve

$$\arg\min_{x\in\mathbb{R}^n} x^T L_G x = \sum_{(i,j)\in E} (x_i - x_j)^2$$

$$\text{subject to } x^T \mathbf{1} = 0$$
$$\|x\|_2 = 1$$

  and place nodes as per x, and find a partition using simple algorithms like k-means
- It turns out that the above optimization has a efficient solver, because The optimal x turns out to be the second smallest eigen vector of the graph Laplacian L_G
- Since, eigen values of a matrix is also called a spectrum, this is called a spectral clustering algorithm

40

40

Spectral Clustering

- Step 1. Define a similarity graph $G(V, E, W)$
- Step 2. Compute the Graph Laplacian

$$L_G = D - W$$

where $D$is a diagonal matrix with $D_{ii} = \sum_{j=1}^{n} w_{ij}$

  - let x be the Eigen vector corresponding to the second smallest Eigen value
  - Place samples according to x and apply k-means clustering

- instead of using just the second smallest Eigen pair, you can use multiple smallest Eigen pairs

41

41

PART 2

Code Time!

42

42

Code Time

- See demonstration and discussion in class.
- See also links in the “Code Examples” for this lecture assignment.

43

Conclusion

Takeaways

44

K-Means vs. Gaussian Mixture Models (GMMs)

Aspect	K-Means	Gaussian Mixture Model (GMM)
Goal	Partition data into K clusters minimizing within-cluster variance	Estimate parameters of a mixture of K Gaussian distributions
Model assumption	Each cluster is spherical with equal variance	Each cluster is Gaussian $\mathcal{N}(\mu_k, \Sigma_k)$ with its own covariance
Cluster assignment	**Hard:** each sample belongs to exactly one cluster	**Soft:** each sample belongs to clusters with probabilities $r_{ik} = P(z_i = k \mid x_i)$
Objective function	$J = \sum_{i=1}^{n} \|x_i - \mu_{z_i}\|^2$	$\mathcal{L} = \sum_{i=1}^{n} \log \left[\sum_{k=1}^{K} \pi_k \mathcal{N}(x_i \mid \mu_k, \Sigma_k) \right]$
Optimization method	Alternating minimization (hard EM): E-step: assign nearest mean M-step: update centroids	Expectation–Maximization (EM): E-step: compute $r_{ik} = \frac{\pi_k \mathcal{N}(x_i \mid \mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(x_i \mid \mu_j, \Sigma_j)}$ M-step: update parameters $\pi_k = \frac{1}{n} \sum_i r_{ik}$, $\mu_k = \frac{\sum_i r_{ik} x_i}{\sum_i r_{ik}}$, $\Sigma_k = \frac{\sum_i r_{ik} (x_i - \mu_k)(x_i - \mu_k)^\top}{\sum_i r_{ik}}$
Output parameters	Cluster centers $\{\mu_k\}$	Mixture weights $\{\pi_k\}$, means $\{\mu_k\}$, covariances $\{\Sigma_k\}$
Cluster shape	Voronoi regions (spherical boundaries)	Elliptical contours (anisotropic Gaussian)
Relationship	Special case of GMM with $\Sigma_k = \sigma^2 I$ and equal priors $\pi_k = \frac{1}{K}$	Generalization of K-Means with probabilistic assignments

References and Credits

Many of the teaching materials for this course have been adapted from various sources. We are very grateful and thank the following professors, researchers, and practitioners for sharing their teaching materials (in no particular order):

- Yaser S. Abu-Mostafa, Malik Magdon-Ismail and Hsuan-Tien Lin. https://amlbook.com/slides.html
- Ethem Alpaydin. https://www.cmpe.boun.edu.tr/~ethem/i2ml3e/
- **Natasha Jaques**. https://courses.cs.washington.edu/courses/cse446/25sp/
- Lyle Ungar. https://alliance.seas.upenn.edu/~cis520/dynamic/2022/wiki/index.php?n=Lectures.Lectures
- Aurelien Geron. https://github.com/ageron/handson-ml3
- Sebastian Raschka. https://github.com/rasbt/machine-learning-book
- Trevor Hastie. https://www.statlearning.com/resources-python
- Andrew Ng. https://www.youtube.com/playlist?list=PLoROMvodv4rMiGQp3WXShtMGgzqpfVfbU
- Richard Povineli. https://www.richard.povinelli.org/teaching
- … and many others.