Data Science

Week 12

Fitting a Model to Data (4)- Clustering

Outline

- Similarity and distance
- K-means clustering
- Gaussian Mixture Models

Similarity

- Similarity underlies many data science methods and solutions to business problems. If two things (people, companies, products) are similar in some ways they often share other characteristics as well. Data mining procedures often are based on grouping things by similarity or searching for the "right" sort of similarity.
- We saw this implicitly in previous classes where modeling procedures create boundaries for grouping instances together that have similar values for their target variables.

Similarity in different tasks

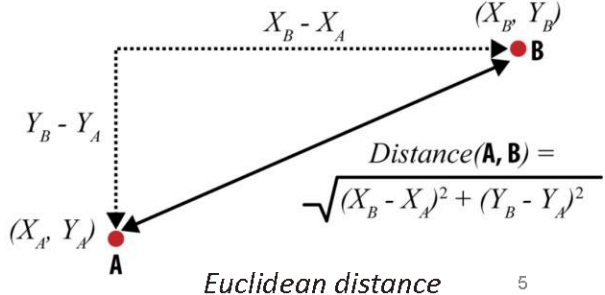
- Different sorts of tasks involve reasoning from similar examples:
  - We may want to retrieve similar things directly. For example, IBM wants to find companies that are similar to their best business customers, in order to have the sales staff look at them as prospects.
  - Similarity can be used for doing *classification*. Since we now know a good bit about classification, we will illustrate the use of similarity with a classification example below.
  - We may want to group similar items together into clusters, for example to see whether our customer base contains groups of similar customers and what these groups have in common.
  - Modern retailers such as Amazon and Netflix use similarity to provide *recommendations* of similar products or from similar people.

Basic method for measuring similarity or distance

- Consider two instances from our simplified credit application domain:

Attribute	Person A	Person B
Age	23	40
Years at current address	2	10
Residential status (1=Owner, 2=Renter, 3=Other)	2	1

These data items have multiple attributes. There are many different ways to measure the similarity or distance between Person A and Person B. A good place to begin is with measurements of distance from basic geometry.



Euclidean distance

Euclidean distance

- we can compute the overall distance by computing the distances of the individual dimensions—the individual features in our setting. This is called the Euclidean distance between two points, and it's probably the most common geometric distance measure.
- When an object is described by *n* features, *n* dimensions *(d_1, d_2, …, d_n)*, the general equation for Euclidean distance in *n* dimensions is shown in

$$\sqrt{(d_{1,A} - d_{1,B})^2 + (d_{2,A} - d_{2,B})^2 + \ ... + (d_{n,A} - d_{n,B})^2}$$

Common Distance measures:

- *Distance measure* will determine how the *similarity* of two elements is calculated and it will influence the shape of the clusters.

1. The Euclidean distance (also called 2-norm distance) is given by:

$$d(\mathbf{p}, \mathbf{q}) = d(\mathbf{q}, \mathbf{p}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2}$$

$$= \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}.$$

2. The Manhattan distance (also called taxicab norm or 1-norm) is given by:

$$d_1(\mathbf{p}, \mathbf{q}) = \|\mathbf{p} - \mathbf{q}\|_1 = \sum_{i=1}^{n} |p_i - q_i|,$$

Common Distance measures:

3.The maximum norm is given by:

$$d(\mathbf{p}, \mathbf{q}) = \max_{1 \le i \le n} |p_i - q_i|$$

4. Inner product space: The angle between two vectors can be used as a distance measure when clustering high dimensional data

5. Hamming distance (sometimes edit distance) measures the minimum number of substitutions required to change one member into another.

Clustering

- Recall the first application of data science that we looked at deeply: supervised segmentation—finding groups of objects that differ with respect to some target characteristic of interest.
- For example, find groups of customers that differ with respect to their propensity to leave the company when their contracts expire.
- Why, in talking about supervised segmentation, do we always use the modifier “supervised”?

Clustering

- In other applications we may want to find groups of objects, for example groups of customers, but not driven by some prespecified target characteristic.
- Do our customers naturally fall into different groups? This may be useful for many reasons. For example, we may want to step back and consider our marketing efforts more broadly. Do we understand who our customers are? Can we develop better products, better marketing campaigns, better sales methods, or better customer service by understanding the natural subgroups?

Clustering

- This idea of finding natural groupings in the data may be called unsupervised segmentation, or more simply *clustering*.
- Clustering is another application of our fundamental notion of similarity. The basic idea is that we want to find groups of objects (consumers, businesses, whiskeys, etc.), where the objects within groups are similar, but the objects in different groups are not so similar.

Outline

- Similarity and distance
- K-means clustering
- Gaussian Mixture Models

K-means clustering

- The **k-means algorithm** is an algorithm to cluster n objects based on attributes into k partitions, where $k < n$.
- It is similar to the **expectation-maximization algorithm** for mixtures of Gaussians in that they both attempt to find the centers of natural clusters in the data.
- It assumes that the object attributes form a vector space.

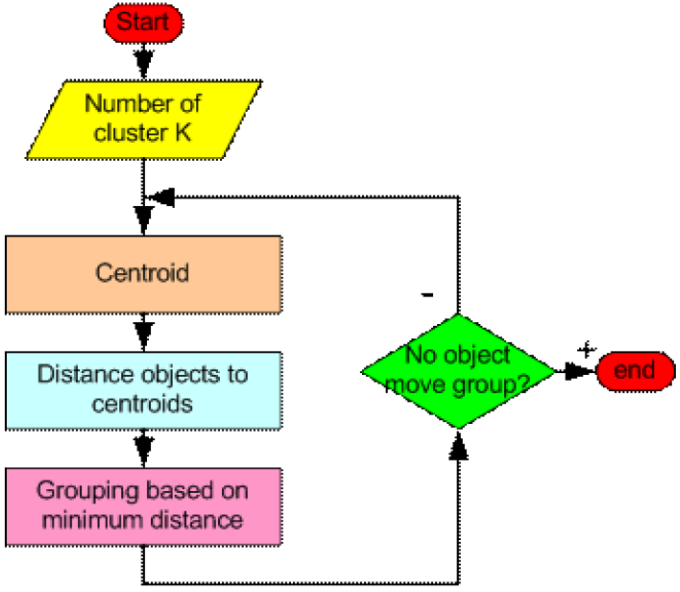
K-means clustering

- An algorithm for partitioning (or clustering) N data points into K disjoint subsets S_j containing data points so as to minimize the sum-of-squares criterion

$$J = \sum_{j=1}^{K} \sum_{n \in S_j} \left| x_n - \mu_j \right|^2,$$

where x_n is a vector representing the n^{th} data point and u_j is the geometric centroid of the data points in S_j.

How the K-Mean Clustering algorithm works?



Steps in K-Mean Clustering

- **Step 1:** Begin with a decision on the value of k = number of clusters .
- **Step 2**: Put any initial partition that classifies the data into k clusters. You may assign the training samples randomly, systematically as the following:
    1. Take the first k training sample as single-element clusters
    2. Assign each of the remaining (N-k) training sample to the cluster with the nearest centroid. After each assignment, recompute the centroid of the gaining cluster.

Steps in K-Mean Clustering

- **Step 3:** Take each sample in sequence and compute its distance from the centroid of each of the clusters. If a sample is not currently in the cluster with the closest centroid, switch this sample to that cluster and update the centroid of the cluster gaining the new sample and the cluster losing the sample.
- **Step 4 .** Repeat step 3 until convergence is achieved, that is until a pass through the training sample causes no new assignments.

A Simple example showing the implementation of k-means algorithm (using K=2)

Individual	Variable 1	Variable 2
1	1.0	1.0
2	1.5	2.0
3	3.0	4.0
4	5.0	7.0
5	3.5	5.0
6	4.5	5.0
7	3.5	4.5

- **Step 1**:
- Initialization: Randomly we choose following two centroids (k=2) for two clusters.
- In this case the 2 centroid are: m1=(1.0,1.0) and m2=(5.0,7.0).

Individual	Variable 1	Variable 2
1	1.0	1.0
2	1.5	2.0
3	3.0	4.0
4	5.0	7.0
5	3.5	5.0
6	4.5	5.0
7	3.5	4.5

	Individual	Mean Vector
Group 1	1	(1.0, 1.0)
Group 2	4	(5.0, 7.0)

Step 2:

- Distance from individual to two Centroids:
- Thus, we obtain two clusters (based on the distance) containing: {1,2,3} and {4,5,6,7}.

Individual	Distance to Centroid 1	Distance to Centroid 2
1	0	7.21
2	1.12	6.10
3	3.61	3.61
4	7.21	0
5	4.72	2.5
6	5.31	2.06
7	4.30	2.92

- After grouping, their new centroids are:

$$m_1 = \left(\frac{1}{3}(1.0+1.5+3.0), \frac{1}{3}(1.0+2.0+4.0)\right) = (1.83, 2.33)$$

$$m_2 = \left(\frac{1}{4}(5.0+3.5+4.5+3.5), \frac{1}{4}(7.0+5.0+5.0+4.5)\right) = (4.12, 5.38)$$

Step 3:

- Now using these centroids we compute the Euclidean distance of each object, as shown in table.

- Therefore, the new clusters are:

  {1,2} and {**3**,4,5,6,7}

Individual 3 are moved to second group because of distance

- Next centroids are: m1=(1.25,1.5) and m2 = (3.9,5.1)

Individual	Distance to Centroid 1	Distance to Centroid 2
1	1.57	5.38
2	0.47	4.28
3	2.04	1.78
4	5.64	1.84
5	3.15	0.73
6	3.78	0.54
7	2.74	1.08

- **Step 4 :**

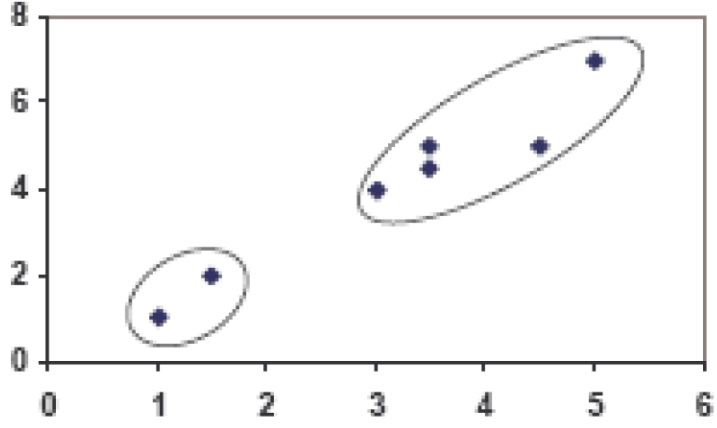
  The clusters obtained are:

  {1,2} and {3,4,5,6,7}

- Based on the distance listed in the table, there is no change in the cluster.
- Thus, the algorithm comes to a halt here and final result consist of 2 clusters {1,2} and {3,4,5,6,7}.

Individual	Distance to Centroid 1	Distance to Centroid 2
1	0.56	5.02
2	0. 56	3.92
3	3.05	1.42
4	6.66	2.20
5	4.16	0.41
6	4.78	0.61
7	3.75	0.72

Visualization of the result



Weaknesses of K-Mean Clustering

1. When the numbers of data are not so many, initial grouping will determine the cluster significantly.
2. The number of cluster, K, must be determined before hand. Its disadvantage is that it does not yield the same result with each run, since the resulting clusters depend on the initial random assignments.
3. We never know the real cluster, using the same data, because if it is inputted in a different order it may produce different cluster if the number of data is few.
4. It is sensitive to initial condition. Different initial condition may produce different result of cluster. The algorithm may be trapped in the *local optimum*.

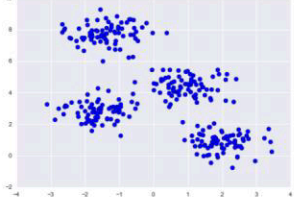
Outline

- Similarity and distance
- K-means clustering-programming
- Gaussian Mixture Models

Data for K-means

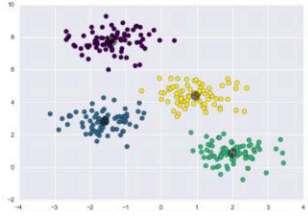
- let's generate a two-dimensional dataset containing four distinct blobs. To emphasize that this is an unsupervised algorithm, we will leave the labels out of the visualization

```
from sklearn.datasets.samples_generator import make_blobs
X, y_true = make_blobs(n_samples=300, centers=4,
                       cluster_std=0.60, random_state=0)
plt.scatter(X[:, 0], X[:, 1], s=50);
```



k-means in Scikit-Learn

- It is relatively easy to pick out the four clusters. The *k*-means algorithm does this automatically, and in Scikit-Learn uses the typical estimator API:



```
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=4)
kmeans.fit(X)
y_kmeans = kmeans.predict(X)
```

```
plt.scatter(X[:, 0], X[:, 1], c=y_kmeans, s=50, cmap='viridis')

centers = kmeans.cluster_centers_
plt.scatter(centers[:, 0], centers[:, 1], c='black', s=200, alpha=0.5);
```

k-Means Algorithm: Expectation–Maximization

- Expectation–maximization (E–M) is a powerful algorithm that comes up in a variety of contexts within data science.
- K-means is a particularly simple and easy-to understand application of the algorithm.

- In short, the expectation–maximization approach consists of the following procedure:
  1. Guess some cluster centers
  2. Repeat until converged
     a. *E-Step*: assign points to the nearest cluster center
     b. *M-Step*: set the cluster centers to the mean

Outline

- Similarity and distance
- K-means clustering
- Gaussian Mixture Models

Motivating GMM: Weaknesses of k-Means

- Let's take a look at some of the weaknesses of *k*-means and think about how we might improve the cluster model.
- As we saw in the previous section, given simple, well separated data, *k*-means finds suitable clustering results.
- For example, if we have simple blobs of data, the *k*-means algorithm can quickly label those clusters in a way that closely matches what we might do by eye.

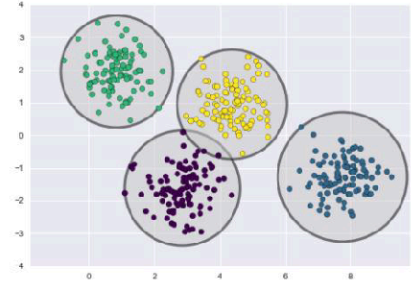
Motivating GMM: Weaknesses of k-Means

- From an intuitive standpoint, we might expect that the clustering assignment for some points is more certain than others;
- for example, there appears to be a very slight overlap between the two middle clusters, such that we might not have complete confidence in the cluster assignment of points between them.
- Unfortunately, the *k*-means model has no intrinsic measure of probability or uncertainty of cluster assignments (although it may be possible to use a bootstrap approach to estimate this uncertainty). For this, we must think about generalizing the model.

Motivating GMM: Weaknesses of k-Means

- One way to think about the *k*-means model is that it places a circle (or, in higher dimensions, a hyper-sphere) at the center of each cluster, with a radius defined by the most distant point in the cluster.
- This radius acts as a hard cutoff for cluster assignment within the training set: any point outside this circle is not considered a member of the cluster. We can visualize this cluster model with the following function

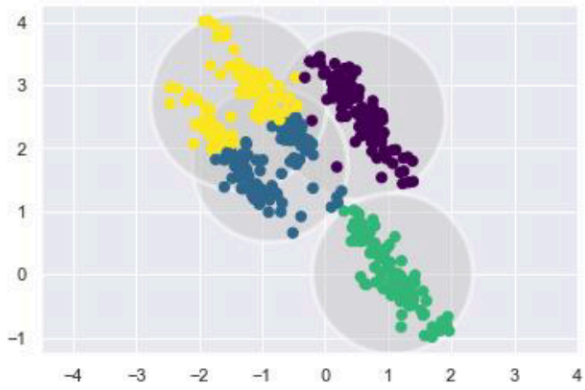
Visualize this cluster model



```
from sklearn.cluster import KMeans
from scipy.spatial.distance import cdist

def plot_kmeans(kmeans, X, n_clusters=4, rseed=0, ax=None):
    labels = kmeans.fit_predict(X)

    # plot the input data
    ax = ax or plt.gca()
    ax.axis('equal')
    ax.scatter(X[:, 0], X[:, 1], c=labels, s=40, cmap='viridis', zorder=2)

    # plot the representation of the k-means model
    centers = kmeans.cluster_centers_
    radii = [cdist(X[labels == i], [center]).max()
             for i, center in enumerate(centers)]
    for c, r in zip(centers, radii):
        ax.add_patch(plt.Circle(c, r, fc='#CCCCCC', lw=3, alpha=0.5, zorder=1))


kmeans = KMeans(n_clusters=4, random_state=0)
plot_kmeans(kmeans, X)
```

Weaknesses of k-Means

An important observation for *k*-means is that these cluster models *must be circular*: *k*-means has no built-in way of accounting for elliptical clusters.



Its lack of flexibility in cluster shape and lack of probabilistic cluster assignment

Generalizing E–M: Gaussian Mixture Models

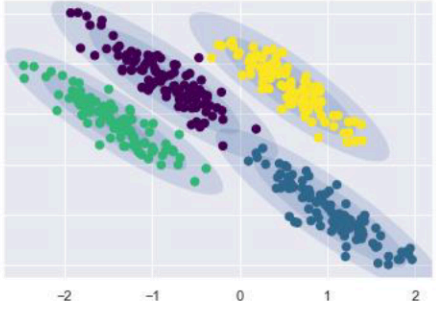
- A Gaussian mixture model (GMM) attempts to find a mixture of multi-dimensional Gaussian probability distributions that best model any input dataset.

- In the simplest case, GMMs can be used for finding clusters in the same manner as k-means

```python
from sklearn.cluster import KMeans
from scipy.spatial.distance import cdist
import numpy as np
import matplotlib.pyplot as plt

from sklearn.datasets.samples_generator import make_blobs
X, y_true = make_blobs(n_samples=400, centers=4,
cluster_std=0.60, random_state=0)
```

```python
rng = np.random.RandomState(13)
X_stretched = np.dot(X, rng.randn(2, 2))
```

```python
from matplotlib.patches import Ellipse
def draw_ellipse(position, covariance, ax=None, **kwargs):
    """Draw an ellipse with a given position and covariance"""
    ax = ax or plt.gca()
    # Convert covariance to principal axes
    if covariance.shape == (2, 2):
        U, s, Vt = np.linalg.svd(covariance)
        angle = np.degrees(np.arctan2(U[1, 0], U[0, 0]))
        width, height = 2 * np.sqrt(s)
    else:
        angle = 0
        width, height = 2 * np.sqrt(covariance)
    # Draw the ellipse
    for nsig in range(1, 4):
        ax.add_patch(Ellipse(position, nsig * width, nsig * height,
        angle, **kwargs))

def plot_gmm(gmm, X, label=True, ax=None):
    ax = ax or plt.gca()
    labels = gmm.fit(X).predict(X)
    if label:
        ax.scatter(X[:, 0], X[:, 1], c=labels, s=40, cmap='viridis', zorder=2)
    else:
        ax.scatter(X[:, 0], X[:, 1], s=40, zorder=2)
        ax.axis('equal')
    w_factor = 0.2 / gmm.weights_.max()
    for pos, covar, w in zip(gmm.means_, gmm.covariances_, gmm.weights_):
        draw_ellipse(pos, covar, alpha=w * w_factor)

from sklearn.mixture import GaussianMixture
gmm = GaussianMixture(n_components=4, random_state=42)
plot_gmm(gmm, X_stretched)
```



Outline

- Similarity and distance
- K-means clustering
- Gaussian Mixture Models - Theory

http://www.ee.bgu.ac.il/~haimp/ml/lectures/lec2/lec2.pdf

Preliminaries

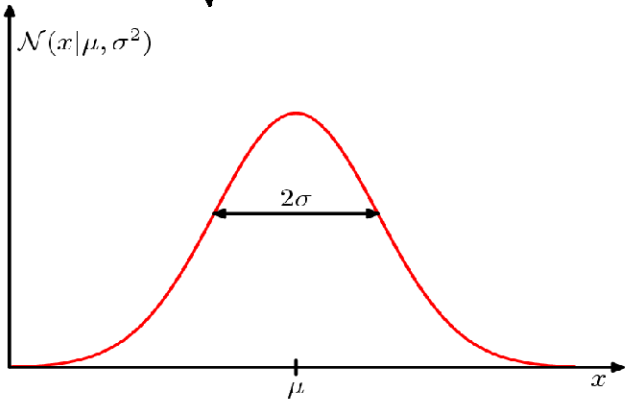
- We assume that the dataset $\mathbf{x}$ has been generated by a *parametric* distribution $p(\mathbf{x})$.
- Estimation of the parameters of p is known as *density estimation*.
- We consider Gaussian distribution.
  - *Mean* (μ): average value of $p(\mathbf{x})$, also called expectation.
  - *Variance* (σ): provides a measure of variability in $p(\mathbf{x})$ around the mean.

Typical parameters

- *Covariance*: measures how much two variables vary together.
- *Covariance matrix*: collection of covariances between all dimensions.
  - Diagonal of the covariance matrix contains the variances of each attribute.

One-dimensional Gaussian

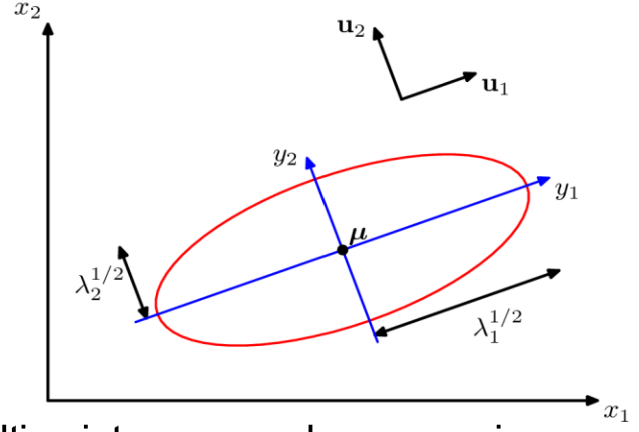
$$\text{Normal}(x \mid \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{ -\frac{1}{2\sigma^2}(x-\mu)^2 \right\}$$



- Parameters to be estimated are the mean (μ) and variance (σ)

Multivariate Gaussian (1)

$$\text{Normal}(\mathbf{x} \mid \mu, \Sigma) = \frac{1}{(2\pi)^{D/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mu)^T\Sigma^{-1}(\mathbf{x}-\mu)\right)$$



- In multivariate case we have covariance matrix instead of variance

The diagonal covariance matrix case

To get an intuition for what a multivariate Gaussian is, consider the simple case where $n = 2$, and where the covariance matrix Σ is diagonal, i.e.,

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \qquad \mu = \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix} \qquad \Sigma = \begin{bmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{bmatrix}$$

In this case, the multivariate Gaussian density has the form,

$$\begin{aligned} p(x; \mu, \Sigma) &= \frac{1}{2\pi \begin{vmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{vmatrix}^{1/2}} \exp\left(-\frac{1}{2} \begin{bmatrix} x_1 - \mu_1 \\ x_2 - \mu_2 \end{bmatrix}^T \begin{bmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{bmatrix}^{-1} \begin{bmatrix} x_1 - \mu_1 \\ x_2 - \mu_2 \end{bmatrix} \right) \\ &= \frac{1}{2\pi(\sigma_1^2 \cdot \sigma_2^2 - 0 \cdot 0)^{1/2}} \exp\left(-\frac{1}{2} \begin{bmatrix} x_1 - \mu_1 \\ x_2 - \mu_2 \end{bmatrix}^T \begin{bmatrix} \frac{1}{\sigma_1^2} & 0 \\ 0 & \frac{1}{\sigma_2^2} \end{bmatrix} \begin{bmatrix} x_1 - \mu_1 \\ x_2 - \mu_2 \end{bmatrix} \right), \end{aligned}$$

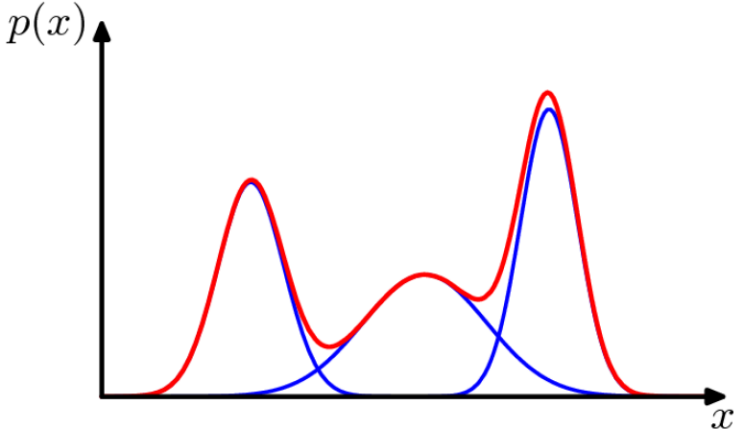
where we have relied on the explicit formula for the determinant of a 2×2 matrix[3], and the fact that the inverse of a diagonal matrix is simply found by taking the reciprocal of each diagonal entry. Continuing,

$$\begin{aligned} p(x; \mu, \Sigma) &= \frac{1}{2\pi\sigma_1\sigma_2} \exp\left(-\frac{1}{2} \begin{bmatrix} x_1 - \mu_1 \\ x_2 - \mu_2 \end{bmatrix}^T \begin{bmatrix} \frac{1}{\sigma_1^2}(x_1 - \mu_1) \\ \frac{1}{\sigma_2^2}(x_2 - \mu_2) \end{bmatrix} \right) \\ &= \frac{1}{2\pi\sigma_1\sigma_2} \exp\left(-\frac{1}{2\sigma_1^2}(x_1 - \mu_1)^2 - \frac{1}{2\sigma_2^2}(x_2 - \mu_2)^2 \right) \\ &= \frac{1}{\sqrt{2\pi}\sigma_1} \exp\left(-\frac{1}{2\sigma_1^2}(x_1 - \mu_1)^2 \right) \cdot \frac{1}{\sqrt{2\pi}\sigma_2} \exp\left(-\frac{1}{2\sigma_2^2}(x_2 - \mu_2)^2 \right). \end{aligned}$$

The last equation we recognize to simply be the product of two independent Gaussian densities, one with mean μ_1 and variance σ_1^2, and the other with mean μ_2 and variance σ_2^2.

Mixtures of Gaussians (1)

$$p(\mathbf{x}) = \sum_{k=1}^{M} \pi_k \text{Normal}(\mathbf{x} \mid \mu_k, \Sigma_k)$$



Mixtures of Gaussians (2)

- In addition to mean and covariance parameters, we have mixing coefficients π_k.

- Following properties hold for the mixing coefficients:

$$\sum_{k=1}^{M} \pi_k = 1 \qquad 0 \leq \pi_k \leq 1$$

It can be seen as the prior probability of the component *k*

Expectation Maximization (EM) for GMM

- Goal: Maximize the log likelihood of the whole data

$$\ln p(\mathbf{X} \mid \Pi, \mu, \Sigma) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{M} \pi_k \text{Normal}(\mathbf{x}_n \mid \mu_k, \Sigma_k) \right\}$$

- we can/should maximize the log likelihood individually for the means, covariances and the mixing coefficients!

EM Algorithm for GMM

- Initialize parameters
- while not converged
  - **E step:** Calculate posteriori probability .

$$\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_n, \boldsymbol{\Sigma}_n)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}$$

  - **M step:** Estimate new parameters

$$\boldsymbol{\mu}_k^{new} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) \boldsymbol{x}_n \qquad N_k = \sum_{n=1}^{N} \gamma(z_{nk})$$

$$\boldsymbol{\Sigma}_k^{new} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk})(\boldsymbol{x}_n - \boldsymbol{\mu}_k^{new})(\boldsymbol{x}_n - \boldsymbol{\mu}_k^{new})^T$$

$$\pi_k^{new} = \frac{N_k}{N}$$

  - Calculate log likelihood of the new parameters

$$\ln p(\boldsymbol{x}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^{N} \ln \left\{ \sum_{k\ 1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}_k|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\}$$

Reference

- Provost, Foster, and Tom Fawcett. Data Science for Business: What you need to know about data mining and data-analytic thinking. O'Reilly Media, Inc., 2013.

- http://www.ee.bgu.ac.il/~haimp/ml/lectures/lec2/lec2.pdf

Data Science

Week 13

Decision Analytic Thinking: What Is a Good Model?