Statistic Machine Learning

Unsupervised Learning
Clustering

Presented By: Nadav Gover

Introduction

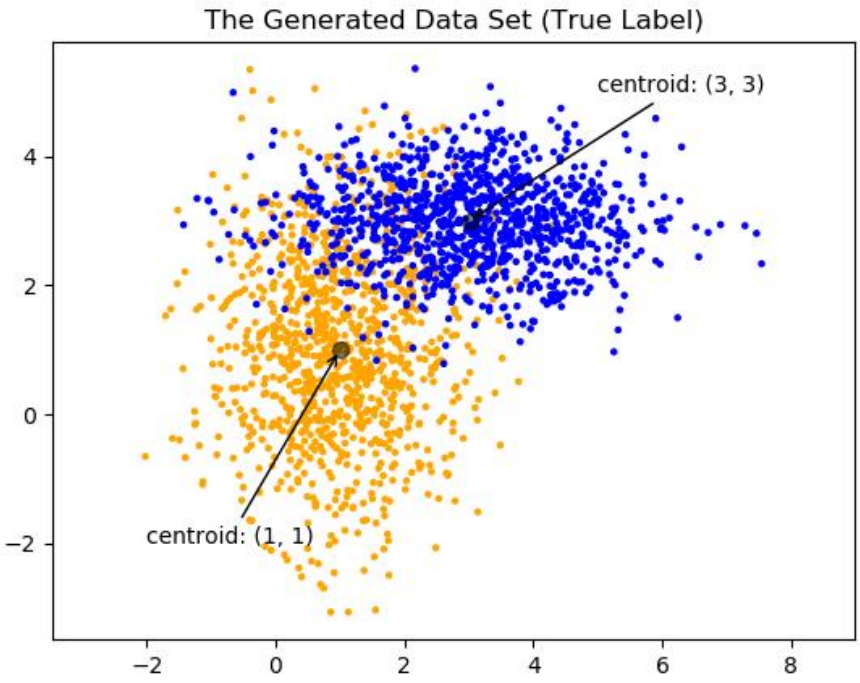
In this assignment the following process was given:

$$w_1 = 0.5,\ w_2 = 0.5, \mu_1 = (1,1)^T,\ \mu_2 = (3,3)^T, \Sigma_1 = \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix}, \Sigma_2 = \begin{pmatrix} 2 & 0 \\ 0 & 0.5 \end{pmatrix}$$

As we can see we have two 2-dimensional multivariate gaussian distribution.

Part A

(a) The data set for this assignment was created from the given process with $N = 2{,}000$ samples and is shown below:



Because $w_1 = w_2 = 0.5$ we have two clusters with even number of samples in it (1,000 in each).

This data set will use us for the rest of the assignment.

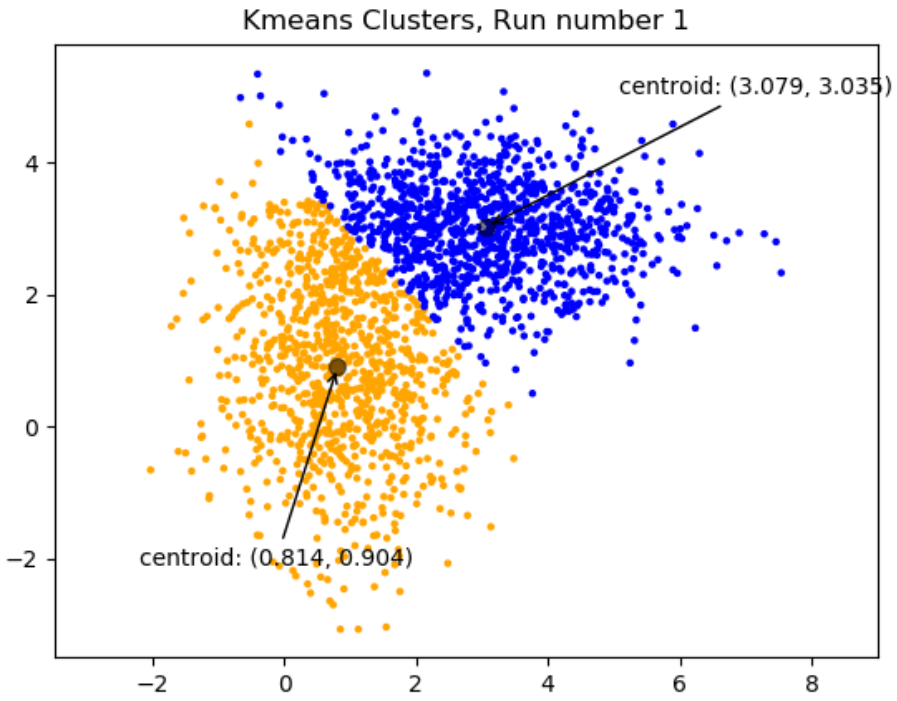
(b) In this section the K-means algorithm was implemented with $K = 2$ (meaning two clusters) and the Euclidean distance metric and was run for 100 iterations.

The initialization vectors were chosen randomly to be 2 different samples of the data set.

Following are the results of this algorithm: (The script prints those results)

Iteration #	Mean Vectors
Initialization (iteration 0)	$\mu_1 = (3.335, 2.802)^T, \mu_2 = (6.239, 1.497)^T$
2	$\mu_1 = (1.492, 1.812)^T, \mu_2 = (4.517, 3.037)^T$
10	$\mu_1 = (0.819, 0.942)^T, \mu_2 = (3.115, 3.037)^T$
100	$\mu_1 = (0.814, 0.904)^T, \mu_2 = (3.079, 3.035)^T$

The clusters at the end of the run are as follows:



The mixing coefficients calculated by the k-means algorithm are: $w_1 = 0.478,\ w_2 = 0.522$

The centroids (mean vectors) are conveniently marked on the graph above.

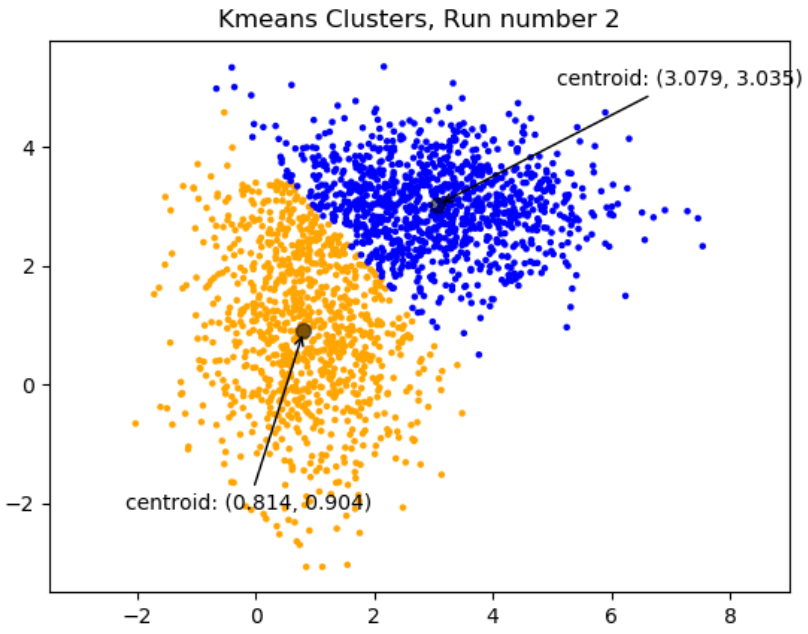
Also, the algorithm converged after 13 iterations.

(c) In this section the mean vectors were reinitialize to another random choice of two samples points and the algorithm was ran for the second time.

Following are the results for this run: (The script prints those results)

Iteration #	Mean Vectors
Initialization (iteration 0)	$\mu_1 = (3.841, 3.344)^T, \mu_2 = (4.307, 3.164)^T$
2	$\mu_1 = (1.220, 1.629)^T, \mu_2 = (4.020, 3.025)^T$
10	$\mu_1 = (0.817, 0.936)^T, \mu_2 = (3.108, 3.035)^T$
100	$\mu_1 = (0.814, 0.904)^T, \mu_2 = (3.079, 3.035)^T$

The clusters at the end of the run are as follows:



The mixing coefficients calculated by the k-means algorithm are: $w_1 = 0.478,\ w_2 = 0.522$

The centroids (mean vectors) are conveniently marked on the graph above.

Also, the algorithm converged after 13 iterations.

Comparison of the two runs: The only thing different between those two runs is the initialization vectors. As we can see from the results above the fact that we had two different initialization vectors did not matter at all. The final means vector and the convergence rate were the same. This means the method of initialization (choosing a random sample from the data set) is stable.

Part B

In this part the EM (Expectation Maximization) algorithm was implemented.

(a) Consider a multivariate Gaussian mixture model with diagonal covariance matrices. The iteration formula of the EM algorithm for this case is developed below:

Let $x = (x_1, x_2, \dots, x_n)$ be a sample of n independent observations from a mixture of two multivariate normal distributions of dimension d, and let $z = (z_1, z_2, \dots, z_n)$ be the latent variables that determine the component from which the observation originates

$$X_i|(Z_i = 1) \sim N_d(\mu_1, \Sigma_1)$$

$$X_i|(Z_i = 2) \sim N_d(\mu_2, \Sigma_2)$$

Where

$$P(Z_i = 1) = w_1 \; and \; P(Z_i = 2) = w_2 = 1 - w_1$$

E step:

Given our current estimate of the parameters $\theta^{(t)}$, the conditional distribution of the Z_i is determined by Bayes theorem to be the proportional height of the normal density weighted by *w*:

$$\gamma_{k,i}^{(t)} := P\left(Z_i = k \middle| X_i = x_i; \theta^{(t)}\right) = \frac{w_k^{(t)} \, f\left(x_i; \mu_k^{(t)}, \Sigma_k^{(t)}\right)}{w_1^{(t)} \, f\left(x_i; \mu_1^{(t)}, \Sigma_1^{(t)}\right) + \, w_2^{(t)} \, f\left(x_i; \mu_2^{(t)}, \Sigma_2^{(t)}\right)}$$

These are called the "membership probabilities", or responsibilities, which are normally considered the output of the E step (although this is not the Q function of below).

This E step corresponds with setting up this function for Q:

$$Q\left(\theta \middle| \theta^{(t)}\right) = E_{Z|X,\theta^{(t)}}[logL(\theta; x, Z)]$$

$$= E_{Z|X,\theta^{(t)}}\left[log \prod_{i=1}^{n} L(\theta; x_i, Z_i)\right] = \; E_{Z|X,\theta^{(t)}}\left[\sum_{i=1}^{n} logL(\theta; x_i, Z_i)\right]$$

$$= \sum_{i=1}^{n} E_{Z_i|X,\theta^{(t)}}\left[logL(\theta; x_i, Z_i)\right]$$

$$= \sum_{i=1}^{n} \sum_{k=1}^{2} P\left(Z_i = k \middle| X_i = x_i; \theta^{(t)}\right) logL(\theta_k; x_i, z_i)$$

$$= \sum_{i=1}^{n} \sum_{k=1}^{2} \gamma_{k,i}^{(t)} [logw_k - \frac{1}{2}\log|\Sigma_k| - \frac{1}{2}(x_i - \mu_k)^T \Sigma_k^{-1}(x_i - \mu_k) - \frac{d}{2}\log(2\pi)]$$

Everything in the E step is known before the step is taken except $\gamma_{k,i}$ which is computed according to the equation at the beginning of the E step.

M step:

$$w_k^{(t+1)} = \frac{1}{n}\sum_{i=1}^{n}\gamma_{k,i}^{(t)}$$

$$\mu_k^{(t+1)} = \frac{\sum_{i=1}^{n}\gamma_{k,i}^{(t)} x_i}{\sum_{i=1}^{n}\gamma_{k,i}^{(t)}}$$

$$\Sigma_k^{(t+1)} = \frac{\sum_{i=1}^{n}\gamma_{k,i}^{(t)}(x_i - \mu_k^{(t+1)}) \cdot \left((x_i - \mu_k^{(t+1)}\right)^T}{\sum_{i=1}^{n}\gamma_{k,i}^{(t)}}$$

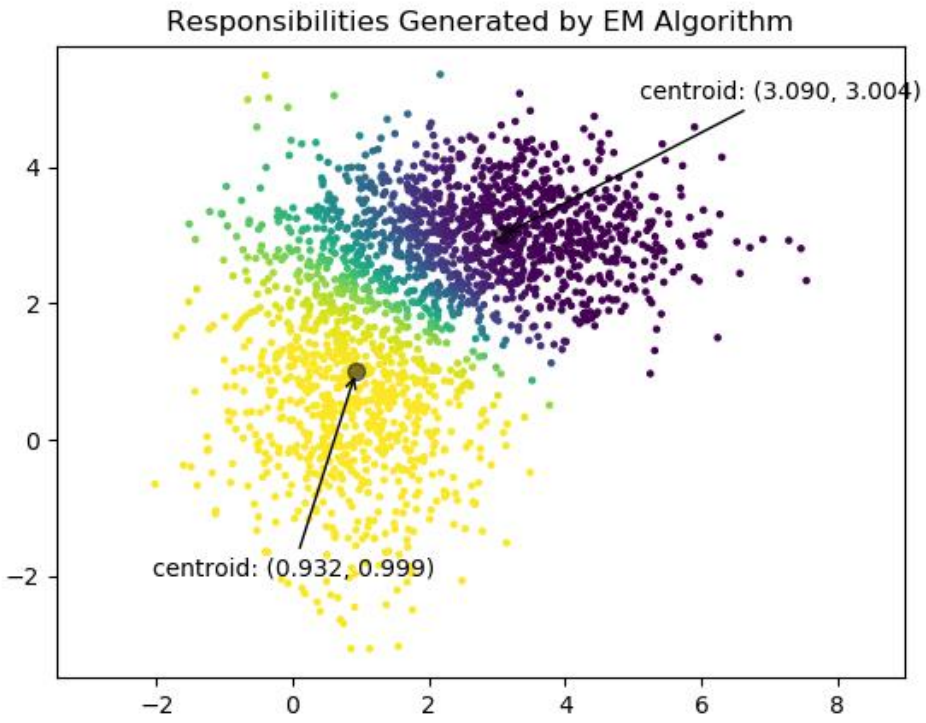
(b+c) In this section the EM algorithm was implemented using the iteration formula from section (a). The algorithm ran on the data set from part A and all the model parameters were initialized to be the results of the second run of the k-means algorithm in part A. The covariance matrix Σ was initialized randomly since it was not calculated with k-means.

Following are the results for this run: (The script prints those results)

Iteration #	Model Parameters
Initialization (iteration 0)	$\mu_1 = (0.814, 0.904)^T, \mu_2 = (3.079, 3.035)^T$ $, \Sigma_1 = \begin{pmatrix} 3 & 0 \\ 0 & 3 \end{pmatrix},$ $\Sigma_2 = \begin{pmatrix} 3 & 0 \\ 0 & 3 \end{pmatrix}$ $w_1 = 0.478,\ w_2 = 0.522$
2	$\mu_1 = (1.032, 1.084)^T, \mu_2 = (3.066, 2.846)^T$ $, \Sigma_1 = \begin{pmatrix} 1.057 & 0.187 \\ 0.187 & 2.355 \end{pmatrix},$ $\Sigma_2 = \begin{pmatrix} 2.152 & 0.197 \\ 0.197 & 0.718 \end{pmatrix}$ $w_1 = 0.487,\ w_2 = 0.513$
10	$\mu_1 = (0.932, 1.000)^T, \mu_2 = (3.088, 3.003)^T$ $, \Sigma_1 = \begin{pmatrix} 0.919 & -0.062 \\ -0.062 & 2.029 \end{pmatrix},$ $\Sigma_2 = \begin{pmatrix} 1.830 & -0.049 \\ -0.049 & 0.486 \end{pmatrix}$ $w_1 = 0.496,\ w_2 = 0.504$
100	$\mu_1 = (0.932, 0.999)^T, \mu_2 = (3.090, 3.004)^T$ $, \Sigma_1 = \begin{pmatrix} 0.918 & -0.065 \\ -0.065 & 2.023 \end{pmatrix},$

	$\Sigma_2 = \begin{pmatrix} 1.825 & -0.052 \\ -0.052 & 0.486 \end{pmatrix}$ $w_1 = 0.496,\ w_2 = 0.504$

The clusters at the end of the run are as follows:



In the graph above each point has a color proportional to the probability of it belonging to a Gaussian. We can see that the algorithm is positive about the samples far from the overlapping area. Inside the overlapping area there is less confidence of which gaussians it belongs to, but we can see that the model parameters are predicted (almost) correctly.

Comparison between the results of both algorithms and conclusion:

In both the k-means and EM we got convergence early on. The main difference is in the mean vectors. While the mean vector of the cluster with mean $\mu_2 = (3,3)^T$ is similar in both algorithms, the mean vector $\mu_1 = (1,1)^T$ is much closer on the EM algorithm. The results of the mean vector of $k = 1$ are shown here again:
EM: $\mu_1 = (0.932, 0.999)^T$,
k-means: $\mu_1 = (0.814, 0.904)^T$

This is explained by the difference between the algorithms:

K-means:

1. Hard assign a data point to one particular cluster on convergence.
2. It makes use of the Euclidian distance

EM:

1. Soft assigns a point to clusters (so it gives a probability of any point belonging to any centroid).
2. It doesn't depend on the Euclidian distance, but is based on the Expectation, i.e., the probability of the point belonging to a particular cluster.

This makes k-means biased towards spherical clusters.

In conclusion, the accuracy of the k-means was lower as compared to that of the EM algorithm. The tendency of k-means to produce spherical clusters leads to bad results here, while EM benefits from the Gaussian distributions present in the data set.