COMP.4220 - Machine Learning

Homework 8

student name:

---

1. In this question, you practice the K-means algorithm in action. Consider a small data set including 5 data points in 2D plane:

$$\mathbf{X} = \begin{bmatrix} 2 & 5 \\ 3 & 4 \\ 1 & 6 \\ -3 & -3 \\ -4 & -2 \\ 0 & 0 \end{bmatrix}$$

Starting from

$$\boldsymbol{\mu}_0 = \begin{bmatrix} -3 & 3 \\ 2 & -2 \end{bmatrix},$$

(a) Calculate $\mathbf{r}$ using the first E-step of the K-means algorithm.

(b) Use $\mathbf{r}$ from previous part to update the cluster centers by performing the first M-step of the K-means algorithm.

(10 marks)

2. (Programming) In this question, you investigate clustering using K-means and Gaussian Mixture Models.

   ⇒**Note:** For overall consistency, activate the `random_state` argument in all functions using `seed=1234`.

   (a) Generate a data set using the `make_blob` function from scikit-learn. This data set should include 3 clusters with a total of 500 samples. Set the standard deviation of the clusters as $[1.0, 2.5, 0.5]$. Make sure you set the `random_state` argument correctly. Assign the data to X_{blob} and t_{blob}. Note that the clustering methods are unsupervised and do not use the target values. We only need t_{blob} as ground truth and for evaluation of our models. (1)

   (b) Generate a second data set using the `make_circles` function from scikit-learn. This data set should include a total of 500 samples. Set the scale factor to 0.5, and the noise level to 0.05. Make sure you set the `random_state` argument correctly. Assign the data to X_{circle} and t_{circle}. (1)

   (c) Instantiate a `Kmeans` class and train two models, one for each data set. When the algorithm converges, extract the cluster centers and predicted labels. Make sure you set the `random_state` argument correctly. (2)

   (d) Instantiate a `GaussianMixture` class and train two models, one for each data set. When the algorithm converges, extract the cluster centers, "full" covariances, and predicted labels. Make sure you set the `random_state` argument correctly. (2)

   (e) Generate a figure including three subplots in one row. The first subplot should show the clusters with colors according to their ground truth labels. The middle subplot should show the clusters according to the K-means algorithm together with cluster centers marked using $\times$ symbols. The third subplot should show the clusters according to the Gaussian Mixture Model results together with the cluster centers marked using $\times$ symbols. For all subplots use the predicted labels to assign proper color to each data point. Include this figure of 1×3 subplots in your report. (2)

   (f) Generate another figure of 1×3 subplots for the second data set. This figure should also include all the details explained in the previou spart. Make sure all the axes are labeled accordingly and each figure has a proper title. (2)

   (g) Write python function for plotting an ellipse per cluster to visualize 2D GMM covariances. To plot an ellipse, you can use the `Ellipse` function from `patches` in matplotlib. The center of the ellipses are defined by the GMM means. The width and height of the ellipse along the two major axes can be calculated from the eigenvalues, $\mathbf{v}$, of the covariance matrix using this transformation: $2\sqrt{2}\mathbf{v}$. The angle of the ellipse can be calculated using eigenvectors of the covariance

matrix according to

$$\alpha = \arctan(\mathbf{u}_1, \mathbf{u}_0) + 180$$

where $\mathbf{u} = \frac{\mathbf{w}_0}{\|\mathbf{w}_0\|}$ is the first normalized eigenvector of the covariance. (2)

(h) Generate a table and report the score of clustering results for both data sets and both algorithms. For each case explain what you observed. (1)

	D_{blob}	D_{circle}
Kmeans		
GMM		

(i) Discover and report what method is used by default to initialize cluster centers in GMM. Train a Gaussian Mixture model on the data set generated using `make_circles`, this time making sure you initialize the parameters of the GMM (e.g., initial centers, weights, and covariances) *randomly*. Compare your results with when you initialize the parameters using `kmeans`. Repeat the test for a few times (e.g., 5) and calculate an average score for comparing the two scenarios. Report the average scores in a table. Which initialization method works better? Explain why? In all cases make sure to set the `random_state` argument correctly. (2)

(j) (+1 Extra Credit) Describe how the results can be improved. Verify your claim. (1)

(15 marks)

grading: parts (a), (b), (h) 1 mark each. Other parts 2 marks each. Part (j) +1 extra credit mark.