

1.1 Performance evaluation of Logistic-Regression models (12 pts.)

In this problem, your task is to evaluate the performance of two Logistic-Regression models (M_1 and M_2) for a binary classification problem. The attributes in the test set you have chosen are represented by \mathbf{X} . Table 1 shows the output ($P(+|M_1)$ and $P(+|M_2)$) of the models, which represent the posterior probabilities for the positive class. As this is a binary classification problem, $P(-) = 1 - P(+)$ and $P(- | \mathbf{X}) = 1 - P(+ | \mathbf{X})$.

Instance	True Class	$P(+ \mathbf{X}, M_1)$	$P(+ \mathbf{X}, M_2)$
1	+	0.78	0.61
2	+	0.62	0.08
3	-	0.44	0.62
4	-	0.55	0.39
5	+	0.61	0.48
6	+	0.47	0.09
7	-	0.07	0.38
8	-	0.14	0.09
9	+	0.48	0.06
10	-	0.32	0.01

Table 1: Output of two models

- (1) For the two models M_1 and M_2 , suppose you choose the decision threshold to be $\tau = 0.5$. In other words, any test instances whose output of the model is greater than τ will be classified as a positive example. Write down the confusion matrix and calculate the precision, recall and accuracy of the two models respectively.

Solution:

Model M_1 :		Predicted Negative (-)	Predicted Positive (+)
Actual Negative (-)		$TN = 4$	$FP = 1$
Actual Positive (+)		$FN = 2$	$TP = 3$

$$\text{Precision} = \frac{TP}{TP + FP} = \frac{3}{3+1} = 0.75$$

$$\text{Recall} = \frac{TP}{TP + FN} = \frac{3}{3+2} = 0.6$$

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} = \frac{3+4}{3+4+1+2} = 0.7$$

Model M_2		Predicted Negative (-)	Predicted Positive (+)
Actual Negative (-)		$TN = 4$	$FP = 1$
Actual Positive (+)		$FN = 4$	$TP = 1$

$$\text{Precision} = \frac{TP}{TP + FP} = \frac{1}{1+1} = 0.5$$

$$\text{Recall} = \frac{TP}{TP + FN} = \frac{1}{1+4} = 0.2$$

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} = \frac{1+4}{1+4+1+4} = 0.5$$

(2) Plot the ROC for both M_1 and M_2 , and calculate their AUC. Which model performs better based on the AUC value? (Hint: In this scenario, the ROC is a step line.)

Solution: Model M1

Threshold = 0.0 $TP = 5 \quad FN = 0 \quad FP = 5 \quad TN = 0$
 $TPR = \frac{TP}{TP+FN} = \frac{5}{5+0} = 1 \quad FPR = \frac{FP}{FP+TN} = \frac{5}{5+0} = 1$

Threshold = 0.14 $TP = 5 \quad FN = 0 \quad FP = 4 \quad TN = 1$
 $TPR = \frac{TP}{TP+FN} = \frac{5}{5+0} = 1 \quad FPR = \frac{FP}{FP+TN} = \frac{4}{4+1} = 0.8$

Threshold = 0.32 $TP = 5 \quad FN = 0 \quad FP = 3 \quad TN = 2$
 $TPR = \frac{TP}{TP+FN} = \frac{5}{5+0} = 1 \quad FPR = \frac{FP}{FP+TN} = \frac{3}{3+2} = 0.6$

Threshold = 0.44 $TP = 5 \quad FN = 0 \quad FP = 2 \quad TN = 3$
 $TPR = \frac{TP}{TP+FN} = \frac{5}{5+0} = 1 \quad FPR = \frac{FP}{FP+TN} = \frac{2}{2+3} = 0.4$

Threshold = 0.47 $TP = 5 \quad FN = 0 \quad FP = 1 \quad TN = 4$
 $TPR = \frac{TP}{TP+FN} = \frac{5}{5+0} = 1 \quad FPR = \frac{FP}{FP+TN} = \frac{1}{1+4} = 0.2$

Threshold = 0.48 $TP = 4 \quad FN = 1 \quad FP = 1 \quad TN = 4$
 $TPR = \frac{TP}{TP+FN} = \frac{4}{4+1} = 0.8 \quad FPR = \frac{FP}{FP+TN} = \frac{1}{1+4} = 0.2$

Threshold = 0.55 $TP = 3 \quad FN = 2 \quad FP = 1 \quad TN = 4$
 $TPR = \frac{TP}{TP+FN} = \frac{3}{3+2} = 0.6 \quad FPR = \frac{FP}{FP+TN} = \frac{1}{1+4} = 0.2$

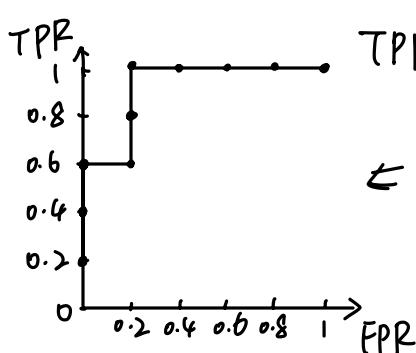
Threshold = 0.61 $TP = 3 \quad FN = 2 \quad FP = 0 \quad TN = 5$
 $TPR = \frac{TP}{TP+FN} = \frac{3}{3+2} = 0.6 \quad FPR = \frac{FP}{FP+TN} = \frac{0}{0+5} = 0$

Threshold = 0.62 $TP = 2 \quad FN = 3 \quad FP = 0 \quad TN = 5$
 $TPR = \frac{TP}{TP+FN} = \frac{2}{2+3} = 0.4 \quad FPR = \frac{FP}{FP+TN} = \frac{0}{0+5} = 0$

Threshold = 0.78 $TP = 1 \quad FN = 4 \quad FP = 0 \quad TN = 5$

$TPR = \frac{TP}{TP+FN} = \frac{1}{1+4} = 0.2 \quad FPR = \frac{FP}{FP+TN} = \frac{0}{0+5} = 0$

← The ROC curve
for Model M1



	1	2	3	4	7	8	10
1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1
5	1	1	1	1	1	1	1
6	1	0	1	1	1	1	1
9	1	0	1	1	1	1	1

$$AUC = \frac{1}{m^+ m^-} \sum_{i=1}^{m^+} \sum_{j=1}^{m^-} u(c_{ij}) = \frac{23}{25} = 0.92$$

Model M2

Threshold = 0.0 | $TP = 5$ $FN = 0$ $FP = 5$ $TN = 0$
 $TPR = \frac{TP}{TP+FN} = \frac{5}{5+0} = 1$ $FPR = \frac{FP}{FP+TN} = \frac{5}{5+0} = 1$

Threshold = 0.06 | $TP = 5$ $FN = 0$ $FP = 4$ $TN = 1$
 $TPR = \frac{TP}{TP+FN} = \frac{5}{5+0} = 1$ $FPR = \frac{FP}{FP+TN} = \frac{4}{4+1} = 0.8$

Threshold = 0.08 | $TP = 4$ $FN = 1$ $FP = 4$ $TN = 1$
 $TPR = \frac{TP}{TP+FN} = \frac{4}{4+1} = 0.8$ $FPR = \frac{FP}{FP+TN} = \frac{4}{4+1} = 0.8$

Threshold = 0.09 | $TP = 3$ $FN = 2$ $FP = 4$ $TN = 1$
 $TPR = \frac{TP}{TP+FN} = \frac{3}{3+2} = 0.6$ $FPR = \frac{FP}{FP+TN} = \frac{4}{4+1} = 0.8$

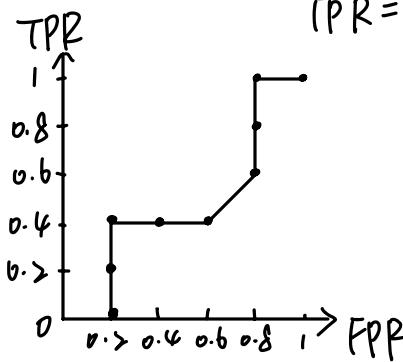
Threshold = 0.38 | $TP = 2$ $FN = 3$ $FP = 3$ $TN = 2$
 $TPR = \frac{TP}{TP+FN} = \frac{2}{2+3} = 0.4$ $FPR = \frac{FP}{FP+TN} = \frac{3}{3+2} = 0.6$

Threshold = 0.39 | $TP = 2$ $FN = 3$ $FP = 2$ $TN = 3$
 $TPR = \frac{TP}{TP+FN} = \frac{2}{2+3} = 0.4$ $FPR = \frac{FP}{FP+TN} = \frac{2}{2+3} = 0.4$

Threshold = 0.48 | $TP = 2$ $FN = 3$ $FP = 1$ $TN = 4$
 $TPR = \frac{TP}{TP+FN} = \frac{2}{2+3} = 0.4$ $FPR = \frac{FP}{FP+TN} = \frac{1}{1+4} = 0.2$

Threshold = 0.61 | $TP = 1$ $FN = 4$ $FP = 1$ $TN = 4$
 $TPR = \frac{TP}{TP+FN} = \frac{1}{1+4} = 0.2$ $FPR = \frac{FP}{FP+TN} = \frac{1}{1+4} = 0.2$

Threshold = 0.62 | $TP = 0$ $FN = 5$ $FP = 1$ $TN = 4$
 $TPR = \frac{TP}{TP+FN} = \frac{0}{0+5} = 0$ $FPR = \frac{FP}{FP+TN} = \frac{1}{1+4} = 0.2$



The ROC curve
For model M2

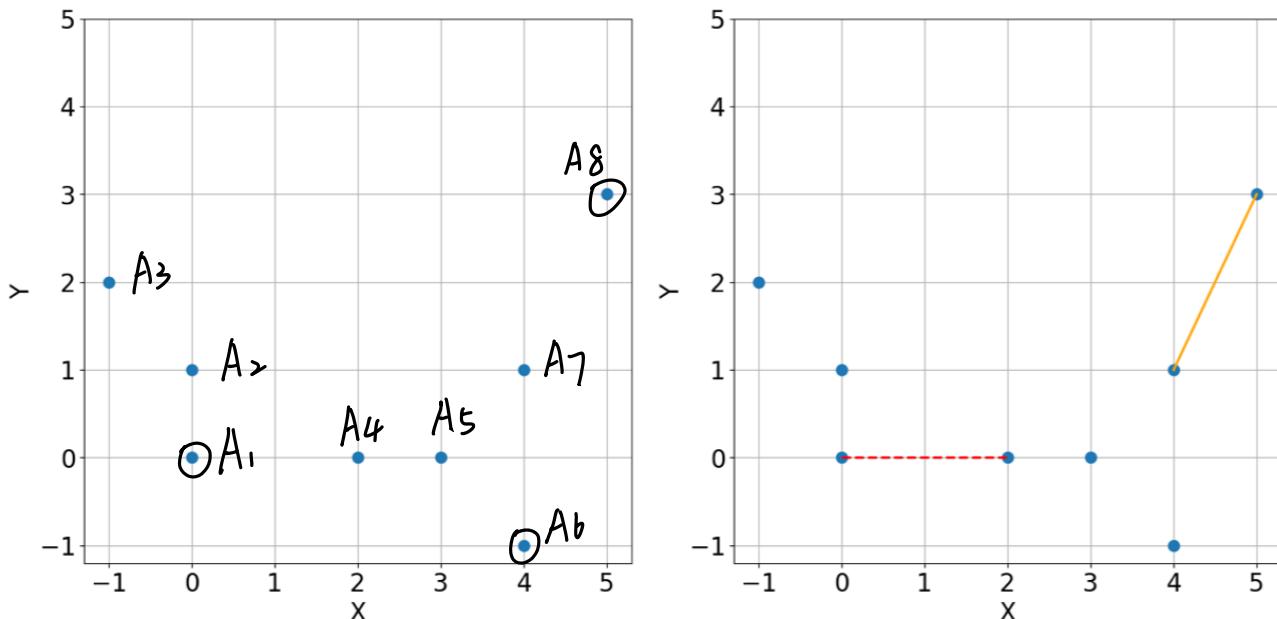
	3	4	7	8	10
1	0	1	1	1	1
2	0	0	0	0	1
5	0	1	1	1	1
6	0	0	0	0.5	1
9	0	0	0	0	1

$AUC = \frac{1}{m^+ m^-} \sum_{i=1}^{m^+} \sum_{j=1}^{m^-} u(c_{ij})$
 $= \frac{11 \times 1 + 0.5}{75} = 0.46$

$AUC_{(M_1)} > AUC_{(M_2)}$

\therefore Model M1 performs better.

1.2 Problem: Manual implementation of K-means (13 pts.)



(a) Training data set for K-means clustering w/o constraints (b) Training data set for K-means clustering w/ constraints

Figure 1: Manual implementation of K-means

Given the data set as shown in Figure 1 and assume that points A_1 , A_6 and A_8 are chosen to be the initialized cluster centers. The coordinates of the data points are:

$$A_1 = \underbrace{(0, 0)}_{\text{underlined}}, A_2 = (0, 1), A_3 = (-1, 2), A_4 = (2, 0), A_5 = (3, 0), A_6 = \underbrace{(4, -1)}_{\text{underlined}}, A_7 = (4, 1), A_8 = \underbrace{(5, 3)}_{\text{underlined}}$$

(1) Use the K-means algorithm and Euclidean distance to cluster the 8 data points shown in 1a into $K = 3$ clusters. Show the new clusters (i.e. the examples belonging to each cluster) and cluster centers after the first iteration, does the algorithm converge after the first iteration?

$$\text{Solution: (1)} \quad d_{21} = 1, \quad d_{26} = \sqrt{4^2 + 2^2} = 2\sqrt{5}, \quad d_{28} = \sqrt{5^2 + 2^2} = \sqrt{29} \quad \therefore A_2 \rightarrow A_1$$

$$d_{31} = \sqrt{1^2 + 2^2} = \sqrt{5}, \quad d_{36} = \sqrt{5^2 + 3^2} = \sqrt{34}, \quad d_{38} = \sqrt{6^2 + 1^2} = \sqrt{37} \quad \therefore A_3 \rightarrow A_1$$

$$d_{41} = \sqrt{2^2} = 2, \quad d_{46} = \sqrt{2^2 + 1^2} = \sqrt{5}, \quad d_{48} = \sqrt{3^2 + 3^2} = 3\sqrt{2} \quad \therefore A_4 \rightarrow A_1$$

$$d_{51} = 3, \quad d_{56} = \sqrt{1^2 + 1^2} = \sqrt{2}, \quad d_{58} = \sqrt{2^2 + 3^2} = \sqrt{13} \quad \therefore A_5 \rightarrow A_6$$

$$d_{71} = \sqrt{4^2 + 1^2} = \sqrt{17}, \quad d_{76} = 2, \quad d_{78} = \sqrt{1^2 + 2^2} = \sqrt{5} \quad \therefore A_7 \rightarrow A_6$$

\therefore The new cluster after the first iteration:

① Cluster C1 (Initial center $A_1(0, 0)$): A_1, A_2, A_3, A_4

$$\text{New X center} = \frac{0+0-1+2}{4} = \frac{1}{4}, \quad \text{New Y center} = \frac{0+1+2+0}{4} = \frac{3}{4}$$

New cluster center: $(\frac{1}{4}, \frac{3}{4})$

② Cluster C2 (Initial center $A_6(4, 1)$): A_5, A_6, A_7

$$\text{New X center} = \frac{3+4+4}{3} = \frac{11}{3}, \quad \text{New Y center} = \frac{0-1+1}{3} = 0$$

New cluster center: $(\frac{11}{3}, 0)$

③ Cluster C3 (Initial center $A_8(5, 3)$): A_8

New center is the same as initial: $(5, 3)$

\because The cluster centers have shifted and a new assignment of points might occur

\therefore The algorithm does not converge after the first iteration

New cluster center: $P_1 = (\frac{1}{4}, \frac{3}{4})$ $P_2 = (\frac{11}{3}, 0)$ $P_3 = (5, 3)$

$$d_{A_1-P_1} = \sqrt{(\frac{1}{4})^2 + (\frac{3}{4})^2} = \frac{\sqrt{10}}{4} \quad d_{A_1-P_2} = \frac{11}{3} \quad d_{A_1-P_3} = \sqrt{5^2 + 3^2} = \sqrt{34} \quad A_1 \rightarrow P_1$$

$$d_{A_2-P_1} = \sqrt{(\frac{1}{4})^2 + (\frac{1}{4})^2} = \frac{\sqrt{2}}{4} \quad d_{A_2-P_2} = \sqrt{(\frac{11}{3})^2 + 1^2} = \frac{\sqrt{130}}{3} \quad d_{A_2-P_3} = \sqrt{5^2 + 2^2} = \sqrt{29} \quad A_2 \rightarrow P_1$$

$$d_{A_3-P_1} = \sqrt{(\frac{5}{4})^2 + (\frac{5}{4})^2} = \frac{5\sqrt{2}}{4} \quad d_{A_3-P_2} = \sqrt{(\frac{14}{3})^2 + 2^2} = \frac{2\sqrt{58}}{3} \quad d_{A_3-P_3} = \sqrt{6^2 + 1^2} = \sqrt{37} \quad A_3 \rightarrow P_1$$

$$d_{A_4-P_1} = \sqrt{(\frac{7}{4})^2 + (\frac{3}{4})^2} = \frac{\sqrt{58}}{4} \quad d_{A_4-P_2} = \sqrt{(\frac{5}{3})^2} = \frac{5}{3} \quad d_{A_4-P_3} = \sqrt{3^2 + 3^2} = 3\sqrt{2} \quad A_4 \rightarrow P_2$$

$$d_{A_5-P_1} = \sqrt{(\frac{11}{4})^2 + (\frac{3}{4})^2} = \frac{\sqrt{130}}{4} \quad d_{A_5-P_2} = \frac{2}{3} \quad d_{A_5-P_3} = \sqrt{2^2 + 3^2} = \sqrt{13} \quad A_5 \rightarrow P_2$$

$$d_{A_6-P_1} = \sqrt{(\frac{15}{4})^2 + (\frac{7}{4})^2} = \frac{\sqrt{274}}{4} \quad d_{A_6-P_2} = \sqrt{(\frac{1}{3})^2 + 1^2} = \frac{\sqrt{10}}{3} \quad d_{A_6-P_3} = \sqrt{1^2 + 4^2} = \sqrt{17} \quad A_6 \rightarrow P_2$$

$$d_{A_7-P_1} = \sqrt{(\frac{15}{4})^2 + (\frac{1}{4})^2} = \frac{\sqrt{226}}{4} \quad d_{A_7-P_2} = \sqrt{(\frac{1}{3})^2 + 1^2} = \frac{\sqrt{10}}{3} \quad d_{A_7-P_3} = \sqrt{1^2 + 2^2} = \sqrt{5} \quad A_7 \rightarrow P_2$$

$$d_{A_8-P_1} = \sqrt{(\frac{19}{4})^2 + (\frac{9}{4})^2} = \frac{\sqrt{442}}{4} \quad d_{A_8-P_2} = \sqrt{(\frac{4}{3})^2 + 3^2} = \frac{\sqrt{49}}{3} \quad d_{A_8-P_3} = 0 \quad A_8 \rightarrow P_3$$

\therefore The new cluster after the second iteration:

① Cluster C1 (previous center: $(\frac{1}{4}, \frac{3}{4})$): A_1, A_5, A_3

$$\text{New X center} = \frac{0+0-1}{3} = -\frac{1}{3} \quad \text{New Y center} = \frac{0+1+2}{3} = 1$$

$$\text{New cluster center: } (-\frac{1}{3}, 1)$$

② Cluster C2 (previous center: $(\frac{11}{3}, 0)$): A_4, A_5, A_6, A_7

$$\text{New X center} = \frac{2+3+4+4}{4} = \frac{13}{4}, \quad \text{New Y center} = \frac{0+0-1+1}{4} = 0$$

$$\text{New cluster center: } (\frac{13}{4}, 0)$$

③ Cluster C3 (previous center: $(5, 3)$): A_8

New center is the same as previous: $(5, 3)$

Since the assignment at the second iteration changes compared to the first one, the algorithm does not converge after the first iteration.

(2) Consider the case that there exist 1 must link (solid orange line) and 1 cannot link (dashed red line) as shown in 1b. Show the new clusters and cluster centers after the first iterations, does the algorithm converge after the first iteration?

Solution: According to the clustering results in (1)

\because There is a cannot link between A_1 and A_4

$\therefore A_4$ should be assigned to the next closest cluster center: A_6

\therefore There is a must link between A₇ and A₈

\therefore A₇ should be assigned to the next closest cluster center : A₁

But this still violates the must link

\therefore A₇ should continue being assigned to the next closest cluster center:

\therefore The new cluster after the first iteration:

① Cluster C₁ (Initial center A₁ (0,0)) : A₁, A₂, A₃

$$\text{New X center} = \frac{0+0-1}{3} = -\frac{1}{3}, \text{ new Y center} = \frac{0+1+2}{3} = 1$$

$$\text{New cluster center: } \left(-\frac{1}{3}, 1\right)$$

② Cluster C₂ (Initial center A₆ (4,-1)) : A₄, A₅, A₆

$$\text{New X center} = \frac{2+3+4}{3} = 3, \text{ new Y center} = \frac{0+0-1}{3} = -\frac{1}{3}$$

$$\text{New cluster center: } \left(3, -\frac{1}{3}\right)$$

③ Cluster C₃ (Initial center A₈ (5,3)) : A₇, A₈

$$\text{New X center} = \frac{4+5}{2} = \frac{9}{2}, \text{ new Y center} = \frac{1+3}{2} = 2$$

$$\text{New cluster center: } \left(\frac{9}{2}, 2\right)$$

New cluster center: M₁ (- $\frac{1}{3}$, 1), M₂ (3, - $\frac{1}{3}$), M₃ ($\frac{9}{2}$, 2)

$$d_{A_1-M_1} = \sqrt{\left(\frac{1}{3}\right)^2 + 1^2} = \frac{\sqrt{10}}{3} \quad d_{A_1-M_2} = \sqrt{3^2 + \left(\frac{1}{3}\right)^2} = \frac{\sqrt{82}}{3} \quad d_{A_1-M_3} = \sqrt{\left(\frac{9}{2}\right)^2 + 2^2} = \frac{\sqrt{97}}{2} \quad A_1 \rightarrow M_1$$

$$d_{A_2-M_1} = \sqrt{\left(\frac{1}{3}\right)^2} = \frac{1}{3} \quad d_{A_2-M_2} = \sqrt{3^2 + \left(\frac{4}{3}\right)^2} = \frac{\sqrt{97}}{3} \quad d_{A_2-M_3} = \sqrt{\left(\frac{9}{2}\right)^2 + 1^2} = \frac{\sqrt{85}}{2} \quad A_2 \rightarrow M_1$$

$$d_{A_3-M_1} = \sqrt{\left(\frac{2}{3}\right)^2 + 1^2} = \frac{\sqrt{13}}{3} \quad d_{A_3-M_2} = \sqrt{4^2 + \left(\frac{7}{3}\right)^2} = \frac{\sqrt{193}}{3} \quad d_{A_3-M_3} = \frac{11}{2} \quad A_3 \rightarrow M_1$$

$$d_{A_4-M_1} = \sqrt{\left(\frac{1}{3}\right)^2 + 1^2} = \frac{\sqrt{58}}{3} \quad d_{A_4-M_2} = \sqrt{1^2 + \left(\frac{1}{3}\right)^2} = \frac{\sqrt{10}}{3} \quad d_{A_4-M_3} = \sqrt{\left(\frac{5}{2}\right)^2 + 2^2} = \frac{\sqrt{41}}{2} \quad A_4 \rightarrow M_2$$

This assignment does not violate the cannot link between A₁ and A₄.

$$d_{A_5-M_1} = \sqrt{\left(\frac{10}{3}\right)^2 + 1^2} = \frac{\sqrt{109}}{3} \quad d_{A_5-M_2} = \frac{1}{3} \quad d_{A_5-M_3} = \sqrt{\left(\frac{3}{2}\right)^2 + 2^2} = \frac{5}{2} \quad A_5 \rightarrow M_2$$

$$d_{A_6-M_1} = \sqrt{\left(\frac{13}{3}\right)^2 + 2^2} = \frac{\sqrt{105}}{3} \quad d_{A_6-M_2} = \sqrt{1^2 + \left(\frac{2}{3}\right)^2} = \frac{\sqrt{13}}{3} \quad d_{A_6-M_3} = \sqrt{\left(\frac{1}{2}\right)^2 + 3^2} = \frac{\sqrt{37}}{2} \quad A_6 \rightarrow M_2$$

$$d_{A_7-M_1} = \sqrt{\left(\frac{3}{3}\right)^2 + 0^2} = \frac{13}{3} \quad d_{A_7-M_2} = \sqrt{1^2 + \left(\frac{4}{3}\right)^2} = \frac{5}{3} \quad d_{A_7-M_3} = \sqrt{\left(\frac{1}{2}\right)^2 + 1^2} = \frac{\sqrt{5}}{2} \quad A_7 \rightarrow M_3$$

$$d_{A_8-M_1} = \sqrt{\left(\frac{16}{3}\right)^2 + 2^2} = \frac{2\sqrt{73}}{3} \quad d_{A_8-M_2} = \sqrt{2^2 + \left(\frac{10}{3}\right)^2} = \frac{2\sqrt{34}}{3} \quad d_{A_8-M_3} = \sqrt{\left(\frac{1}{2}\right)^2 + 1^2} = \frac{\sqrt{5}}{2} \quad A_8 \rightarrow M_3$$

This assignment does not violate the must link between A₇ and A₈

\therefore The new cluster after the second iteration:

① Cluster C₁ (previous center: (- $\frac{1}{3}$, 1)) : A₁, A₂, A₃

$$\text{New X center} = \frac{0+0-1}{3} = -\frac{1}{3} \quad \text{New Y center} = \frac{0+1+2}{3} = 1$$

$$\text{New cluster center: } \left(-\frac{1}{3}, 1\right)$$

② Cluster C2 (previous center: $(3, -\frac{1}{3})$): A4, A5, A6

$$\text{New X center} = \frac{2+3+4}{3} = 3 \quad \text{New Y center} = \frac{0+0+1}{3} = -\frac{1}{3}$$

$$\text{New cluster center: } (3, -\frac{1}{3})$$

③ Cluster C3 (previous center: $(\frac{9}{2}, 2)$): A7, A8

$$\text{New X center} = \frac{4+5}{2} = \frac{9}{2} \quad \text{New Y center} = \frac{1+3}{2} = 2$$

$$\text{New cluster center: } (\frac{9}{2}, 2)$$

Since the assignment at the second iteration does not change compared to the first iteration, the algorithm converges after the first iteration.

1.3 Problem: Manual calculation of the M step for a GMM (13 pts.)

In this question we consider clustering 1D data with a mixture of 2 Gaussians using the EM algorithm. A GMM with 1D data represents a distribution as

$$p(x) = \sum_{k=1}^K \pi_k \mathcal{N}(x | \mu_k, \sigma_k)$$

with π_k the mixing coefficients, where: $\sum_{k=1}^K \pi_k = 1$ and $\pi_k \geq 0, \forall k$. And,

$$\mathcal{N}(x | \mu_k, \sigma_k) = \frac{1}{\sqrt{2\pi\sigma_k^2}} \exp\left(-\frac{(x - \mu_k)^2}{2\sigma_k^2}\right)$$

You are given the 1-D data points $X = \{1, 10, 20\}$. Suppose the output of the E step is the following matrix:

$$\mathbf{R} = \begin{bmatrix} 1 & 0 \\ 0.4 & 0.6 \\ 0 & 1 \end{bmatrix}$$

where entry $r_{i,c}$ is the probability of observation X_i belonging to cluster c (the responsibility of cluster c for data point i). You just have to compute the M step. You may state the equations for maximum likelihood estimates of these quantities (which you should know from the lecture) and apply to this data set.

- (1) Write down the likelihood function you are trying to optimize.
- (2) After performing the M step for the mixing weights π_1, π_2 , what are the new values?
- (3) After performing the M step for the means μ_1 and μ_2 , what are the new values?

Solution:

$$(1) L(\theta) = \sum_{i=1}^3 \sum_{c=1}^2 r_{ik} \log(\pi_k N(x_i | \mu_k, \sigma_k))$$

where θ represents the parameters of the GMM, namely the mixing coefficients π_k , the means μ_k and the standard deviations σ_k .

$$(2) \pi_{ik} = \frac{1}{N} \sum_{i=1}^N r_{ik} = \frac{1}{3} \cdot \sum_{i=1}^3 r_{ik}$$

$$\pi_{11} = \frac{1}{3} \cdot (r_{11} + r_{21} + r_{31}) = \frac{1}{3} \times (1 + 0.4) = \frac{7}{15}$$

$$\pi_{12} = \frac{1}{3} \cdot (r_{12} + r_{22} + r_{32}) = \frac{1}{3} \times (0 + 0.6 + 1) = \frac{8}{15}$$

$$(3) MR = \frac{\sum_{i=1}^N r_{ik}x_i}{\sum_{i=1}^N r_{ik}} = \frac{\sum_{i=1}^3 r_{ik}x_i}{\sum_{i=1}^3 r_{ik}} \quad X = \{1, 10, 20\}$$

$$M_1 = \frac{r_{11}x_1 + r_{21}x_2 + r_{31}x_3}{r_{11} + r_{21} + r_{31}} = \frac{1x_1 + 0.4x_{10} + 0x_{20}}{1+0.4+0} = \frac{25}{7}$$

$$M_2 = \frac{r_{12}x_1 + r_{22}x_2 + r_{32}x_3}{r_{12} + r_{22} + r_{32}} = \frac{0x_1 + 0.6x_{10} + 1x_{20}}{0+0.6+1} = \frac{65}{4}$$

1.4 Problem: PCA (12 pts.)

Consider the following 10 data points: $X = \{(2, 0, 1, -3, -2), (0, 2, -3, -3, -2), (1, 2, 1, 3, -2), (-1, 1, 3, 2, -1), (1, 0, 1, -1, 1), (2, 3, -1, 1, -2), (-2, 3, -3, 3, 2), (-2, -2, 2, 3, -2), (-2, -3, 1, -2, -3), (-3, 2, 0, -1, -2)\}$. Compute the unit-length principle components of X and choose two of them for PCA, then calculate the projection of each data on these two principal components. You could use python or matlab to obtain eigenvectors and eigenvalues.

Solution: (1) Calculate the mean of each dimension:

$$M_x = (-0.4, 0.8, 0.2, 0.2, -1.3)$$

(2) Center the data by subtracting the mean:

$$X - M_x = \begin{bmatrix} 2.4 & -0.8 & 0.8 & -3.2 & -0.7 \\ 0.4 & 1.2 & -3.2 & -3.2 & -0.7 \\ 1.4 & 1.2 & 0.8 & 2.8 & -0.7 \\ -0.6 & 0.2 & 2.8 & 1.8 & 0.3 \\ 1.4 & -0.8 & 0.8 & -1.2 & 2.3 \\ 2.4 & 2.2 & -1.2 & 0.8 & -0.7 \\ -1.6 & 2.2 & -3.2 & 2.8 & 3.3 \\ -1.6 & -2.8 & 1.8 & 2.8 & -0.7 \\ -1.6 & -3.8 & 0.8 & -2.2 & -1.7 \\ -2.6 & 1.2 & -0.2 & -1.2 & -0.7 \end{bmatrix}$$

(3) Calculate the covariance matrix of the centered data

$$\Sigma = \frac{1}{N} \sum_{n=1}^N (X - M_x)(X - M_x)^T$$

$$= \begin{bmatrix} -3.04 & 0.82 & -0.02 & -0.82 & -0.12 \\ 0.82 & 3.76 & -2.16 & 1.04 & 1.04 \\ -0.02 & -2.16 & 3.56 & 0.76 & -0.84 \\ -0.82 & 1.04 & 0.76 & 5.56 & 1.16 \\ -0.12 & 1.04 & -0.84 & 1.16 & 2.21 \end{bmatrix}$$

(4) Use python to obtain eigenvectors and eigenvalues

$$q_1 = \begin{bmatrix} -0.33 \\ 0.61 \\ 0.62 \\ -0.33 \\ 0.17 \end{bmatrix}$$

$$q_2 = \begin{bmatrix} 0.17 \\ -0.33 \\ 0.08 \\ -0.16 \\ 0.91 \end{bmatrix}$$

$$q_3 = \begin{bmatrix} 0.88 \\ 0.18 \\ 0.41 \\ 0.14 \\ -0.11 \end{bmatrix}$$

$$q_4 = \begin{bmatrix} 0.30 \\ 0.40 \\ -0.58 \\ -0.65 \\ 0.03 \end{bmatrix}$$

$$q_5 = \begin{bmatrix} -0.03 \\ 0.58 \\ -0.33 \\ 0.65 \\ 0.36 \end{bmatrix}$$

$$\lambda_1 = 0.82 \quad \lambda_2 = 1.53 \quad \lambda_3 = 3.08 \quad \lambda_4 = 5.93 \quad \checkmark \quad \lambda_5 = 6.77 \quad \checkmark$$

(5) The largest two eigenvalues are $\lambda_5 = 6.77$, $\lambda_4 = 5.93$

So choose q_5 and q_4 to form the principle components

$$U = [q_5, q_4] = \begin{bmatrix} -0.03 & 0.30 \\ 0.58 & 0.39 \\ -0.33 & -0.58 \\ 0.65 & -0.65 \\ 0.36 & 0.03 \end{bmatrix}$$

(6) Project the data onto these two principal components

$$U^T(X - \mu_x) = \begin{bmatrix} -3.13 & 1.98 \\ -0.61 & 4.50 \\ 1.97 & -1.40 \\ 0.50 & -2.88 \\ -0.72 & 0.48 \\ 1.88 & 1.74 \\ 5.38 & 0.54 \\ -0.59 & -4.46 \\ -4.47 & -1.07 \\ -0.21 & 0.57 \end{bmatrix}$$



香港中文大學(深圳)
The Chinese University of Hong Kong, Shenzhen

K-means and GMM Implementation from scratch

DDA3020 Machine Learning HW4 Report

Jingquan Li
119010148

Date: May 8, 2024

1 Implement K-means

For $\mathbf{x} \in \mathbb{R}^D$ and finite set $\mathcal{C} \subset \mathbb{R}^D$ define $d(\mathbf{x}, \mathcal{C}) := \min_{\mathbf{c} \in \mathcal{C}} \|\mathbf{x} - \mathbf{c}\|_2$.

For finite set $\mathcal{X} \subset \mathbb{R}^D$ and finite set $\mathcal{C} \subset \mathbb{R}^D$ define $\phi(\mathcal{X}, \mathcal{C}) := \sum_{\mathbf{x} \in \mathcal{X}} d^2(\mathbf{x}, \mathcal{C})$.

Given $\mathcal{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\} \subset \mathbb{R}^D$, k-means optimizes $\mathcal{C} \subset \mathbb{R}^D$, where $|\mathcal{C}| = K$, to minimize the following loss:

$$\mathcal{L}(\mathcal{C}) = \phi(\mathcal{X}, \mathcal{C}) = \sum_{\mathbf{x} \in \mathcal{X}} \min_{\mathbf{c} \in \mathcal{C}} \|\mathbf{x} - \mathbf{c}\|_2^2$$

1.1 Implement the `solve_k_means()` function

1.1.1 Code Overview

The provided `solve_k_means()` function takes a dataset x and an initial set of cluster centers c , and iteratively updates the cluster centers until convergence or until a maximum number of iterations (`max_step`) is reached. The goal is to group the dataset into K clusters, where each point belongs to the cluster with the nearest mean.

1.1.2 Function Signature

Inputs:

- x (`ndarray[float]`): An array with shape $[N, D]$, where N is the number of data samples and D is the dimensionality of each sample.
- c (`ndarray[float]`): An array with shape $[K, D]$, representing the initial K cluster centers.
- `max_step` (`int`, optional): The maximum number of iterations to perform, defaulting to 10,000.

Outputs:

- c (`ndarray[float]`): The updated cluster centers after the algorithm has converged.
- `index` (`ndarray[int]`): An array indicating the nearest cluster center for each data sample.
- `losses` (`list[float]`): A list tracking the loss at each iteration, where the loss is defined as the within-cluster sum of squares.

1.1.3 Implementation Details

Initialization:

- N and D are derived from the shape of x , and K from c .
- `index` is initialized to store the index of the nearest cluster center for each sample.
- `losses` is a list to track the loss over iterations.

Main Iterative Process:

- **Distance Calculation:** For each data point, the Euclidean distance to each cluster center is calculated using vectorized operations, avoiding explicit loops over dimensions or samples.
- **Re-assignment:** Each data point is assigned to the cluster center to which it has the smallest distance.
- **Cluster Center Update:** New cluster centers are computed as the mean of all data points assigned to each cluster.
- **Loss Calculation:** At each step, the loss is calculated as the sum of squared distances of each point to its nearest cluster center.
- **Convergence Check:** The algorithm checks if the updated cluster centers are close to the old centers using `np.allclose`, which compares the arrays element-wise within a tolerance.

1.1.4 Hyperparameter Choices

- **max_step:** Set to a default of 10,000 as a balance between allowing sufficient iterations for convergence and limiting computational cost.
- **Convergence tolerance in np.allclose:** Default settings are used, striking a balance between precision and performance.

1.2 Plot loss function \mathcal{L}

The function `compute_wcss()` computes the Within-Cluster Sum of Squares (WCSS) to evaluate the performance of the k -means clustering algorithm. The WCSS measures the compactness of the clusters and is calculated by summing the squared distances between each point and its nearest centroid.

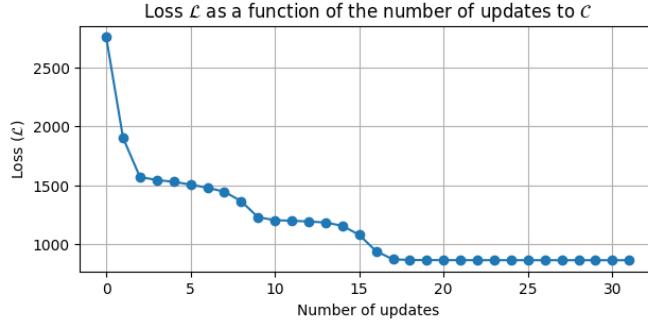


Figure 1: Loss function of the number of updates to c

1.3 Scatter plot of the training samples and cluster centers

1.3.1 Standard Normal Initialization:

This function initializes K cluster centers using random values drawn from a standard normal distribution.

- Input: K (number of cluster centers)
- Output: ndarray of shape $[K, 2]$ representing the initial cluster centers.

1.3.2 Cluster Initialization and Update

- `init_c = standard_normal_initialization(8)`: Initializes 8 cluster centers using the standard normal initialization function.
- `updated_c, index, losses = solve_k_means(train_x, init_c, max_step=1000)`: Solves the K-Means problem using the initial cluster centers, training data (`train_x`), and a maximum number of steps (`max_step=1000`). It returns the updated cluster centers (`updated_c`), index of cluster assignments for each data point (`index`), and the loss values during optimization (`losses`).

1.3.3 Choice of Hyperparameters

- **Number of Clusters (K):** The number of clusters (K) is set to 8.
- **Maximum Steps (max_step):** The maximum number of steps for the optimization process is set to 1000. This parameter determines the maximum number of iterations the algorithm will run before converging. Setting it too low may result in premature convergence, while setting it too high may increase computational overhead.

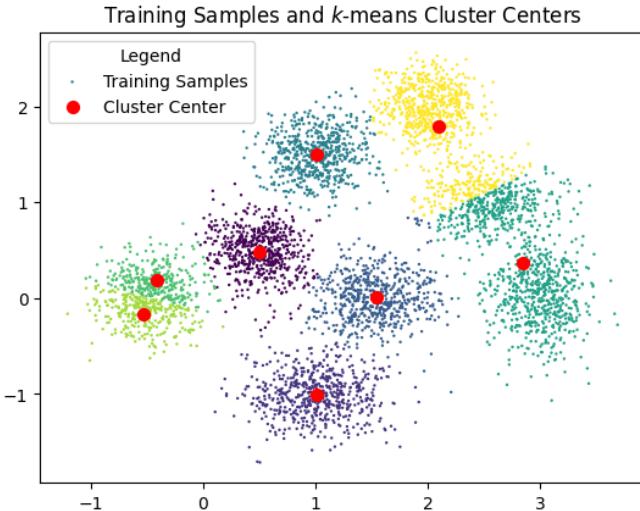


Figure 2: Training Samples and k -means Cluster Centers

1.4 Run k -means multiple times

Conduct a convergence analysis by running the K-Means algorithm multiple times (`num_runs`) with random initializations.

- `num_runs`: The number of times the K-Means algorithm is run with different initializations. In this case, it is set to 10.
- `converge_count`: A counter variable initialized to zero, which keeps track of the number of times the algorithm converges to the same solution as the initial one.
- `init_c`: The initial cluster centers are generated using the `standard_normal_initialization` function.
- `for _ in range(num_runs)`: Iterates through `num_runs` and runs the K-Means algorithm each time.
- `updated_c, index, losses = solve_k_means(train_x, init_c, max_step=1000)`: Solves the K-Means problem with the current initial cluster centers (`init_c`) and training data (`train_x`). It returns the updated cluster centers (`updated_c`), index of cluster assignments for each data point (`index`), and the loss values during optimization (`losses`).
- `if np.allclose(init_c, updated_c)`: Checks if the updated cluster centers are equal to the initial ones. If they are close enough, it implies that the algorithm has converged to the same solution.
- `converge_count += 1`: Increments the `converge_count` if convergence is detected.
- `init_c = updated_c`: Updates the initial cluster centers for the next iteration.
- `freq = converge_count / num_runs`: Calculates the frequency of convergence among all runs.

The calculated `freq` represents the proportion of runs in which the K-Means algorithm converged to the same solution as the initial one. In this case, the calculated frequency that the algorithm converges to a global minimum is 0.9.

2 Implement K-means++

Given \mathcal{X} , k -means++ generates a set of k vectors \mathcal{I} in \mathbb{R}^D using the following steps: 1. First, randomly pick a sample \mathbf{x} from \mathcal{X} . And set $\mathcal{I} := \{\mathbf{x}\}$. 2. While $|\mathcal{I}| < k$. Sample $\mathbf{x} \in \mathcal{X}$ with probability $p(\mathbf{x}) := d^2(\mathbf{x}, \mathcal{I})/\phi(\mathcal{X}, \mathcal{I})$. And add the sampled \mathbf{x} to set \mathcal{I} .

2.1 K-Means++ Initialization Method

This function initializes K cluster centers using the K-Means++ method.

2.1.1 Input

- x (ndarray): An array of shape $[N, D]$, storing N data samples with D as the feature dimension.
- K (int): Number of cluster centers to generate.

2.1.2 Output

- centers (ndarray): An array of shape $[K, D]$, representing the initial cluster centers generated by K-Means++.

2.1.3 Steps in K-Means++ Initialization

1. Randomly Select First Center: Choose the first center randomly from the data points.
2. Initialize Distance Array: Initialize an array to store the squared distances from each data point to the closest center.
3. Choose Remaining Centers:
 - (a) Update the distance from each data point to the closest center.
 - (b) Calculate probabilities for each data point based on the squared distances.
 - (c) Choose the next center based on weighted probabilities.

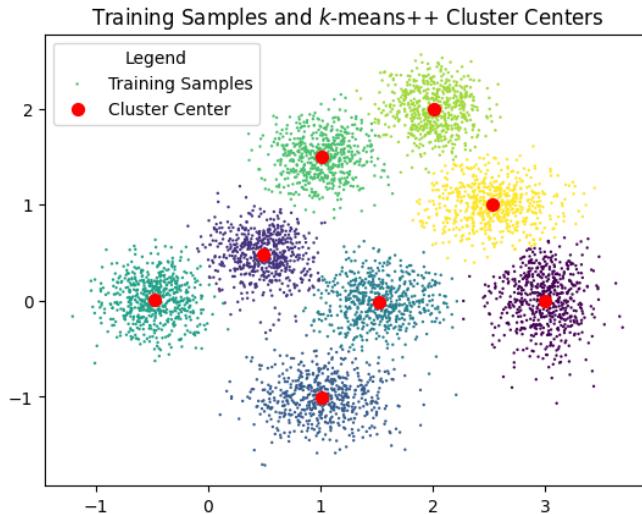


Figure 3: Training Samples and k -means++ Cluster Centers

2.2 Compare the loss and convergence speed

Put the loss function graph with standard normal initialization and k-means++ initialization, and compare their loss. We can see that the loss with k-means++ initialization is largely lower than that with standard normal initialization, which means the k-means++ initialization performs better and has higher efficiency while doing the clustering task.

We can also compare the convergence speed with standard normal initialization and k-means++ initialization. We can see from the graph that the convergence speed with k-means++ initialization is larger than that with standard normal initialization. To be specific, the clustering with k-means++ initialization converged at about the 7th update, but the clustering with standard normal initialization converges at about the 17th update.

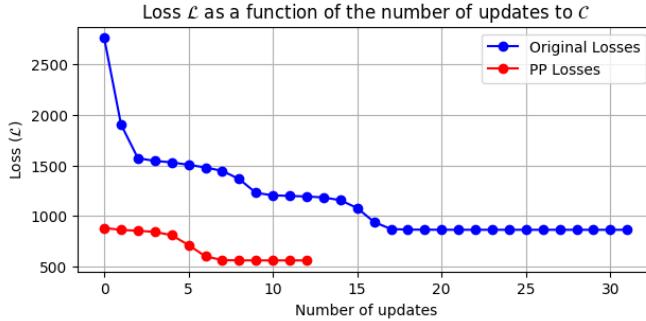


Figure 4: Loss \mathcal{L} as a function of the number of updates to C

3 Implement a Gaussian Mixture Model

In this task, I will implement the Expectation-Maximization (EM) algorithm for learning a Gaussian Mixture Model (GMM). Our GMM in \mathbb{R}^2 has a density given by:

$$p(x) = \sum_{k=1}^K \pi_k \mathcal{N}(x|\mu_k, \Sigma_k), \quad \mu_k \in \mathbb{R}^2, \quad \Sigma_k \in \mathbb{R}^{2 \times 2}$$

3.1 Implement EM algorithm

3.1.1 Code Overview

The GaussianMixtureModel class encapsulates an implementation of the Gaussian Mixture Model (GMM), a probabilistic model for representing normally distributed subpopulations within an overall population. The implementation uses the Expectation-Maximization (EM) algorithm to estimate the parameters that maximize the likelihood of the data given the model.

3.1.2 Class Structure

- **n_components**: The number of Gaussian distributions in the mixture.
- **n_iter**: Maximum number of iterations for the EM algorithm.
- **tol**: Convergence threshold for the log likelihood improvement.
- **means**: Current means of the Gaussian components.
- **init_means**: Initial means used for the EM algorithm.
- **weights**: Mixture weights for each Gaussian component.
- **covariances**: Covariance matrices for each component.
- **responsibilities**: Posterior probabilities of each component given the data.

3.1.3 Methods and their functionalities

- **initialize_parameters(self, X)**: Initializes the model parameters using a k-means++ like initialization for means, sets equal weights, and uses the empirical covariance of the dataset for each component.
- **e_step(self, X)**: Computes the responsibilities using the current model parameters. This step uses the multivariate normal PDF to evaluate the density of each data point under each component, weighted by the component's current weight.
- **m_step(self, X)**: Updates the model's parameters based on the newly computed responsibilities. Calculates new weights as the average of the responsibilities across all data points. Updates the means as the weighted

average of the data points, with weights given by the responsibilities. Recomputes covariance matrices as the weighted outer product of the data deviations.

- **fit(self, X)**: Coordinates the running of the EM algorithm, including initializing parameters and iteratively performing the E and M steps. Monitors convergence by checking if the change in log likelihood between iterations falls below the tolerance.
- **compute_log_likelihood_init(self, X)** and **compute_log_likelihood(self, X)**: Computes the log likelihood of the data under the current model parameters, important for monitoring convergence and for model selection.
- **plot(self, X)**: Utility function for visualizing the results of the Gaussian mixture model.

3.1.4 Hyperparameter choices

- **n_components**: Determines the complexity of the model by specifying the number of Gaussian distributions.
- **n_iter**: Sets a cap on the number of iterations to prevent excessive computation.
- **tol**: Controls the precision of convergence; smaller values result in more precise fits at the cost of potentially longer run times.

3.2 Generate a 2D contour plot of the GMM density

The resulting plot provides a clear visual representation of how well the GMM components fit the data distribution. It highlights areas of high probability density and how these areas correspond to the clustering of data points.

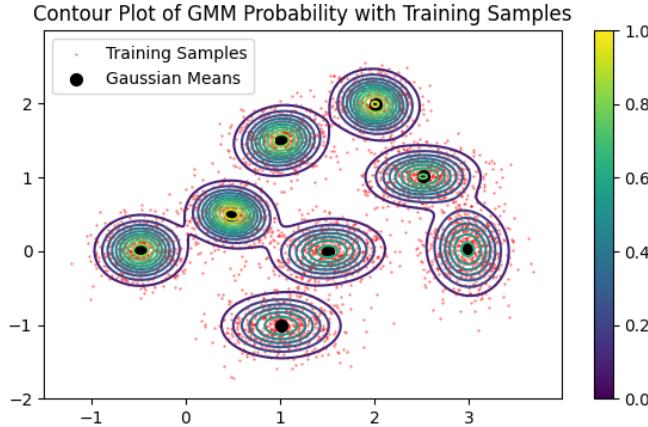


Figure 5: Contour Plot of GMM Probability with Training Samples

3.3 Report the log-likelihood on the training set and development set

The function `gmm_log_prob` computes the log probabilities of given data points under a Gaussian Mixture Model (GMM) using PyTorch's distribution libraries.

3.3.1 The log probability of the training set

`[-1.0763839, -1.54256438, -1.09837806, ..., -1.37186707, -1.0407084, -2.96304398]`

3.3.2 The log probability of the development set

`[-3.01289871, -4.88739487, -1.07392199, -2.01393966, -2.14995957, -1.21085158, -3.38868284, -1.37573343, -1.26428815, -2.95509965, -1.68947912, -2.55407091, -4.98363017, -1.26859155, -1.76087422, -1.87492613, -0.96467143, -2.09768647,`

-1.88104261, -2.79928273, -2.24732301, -1.56953075, -2.3042276, -1.22961184,
 -1.51911158, -1.39717186, -2.70263953, -1.2115244, -1.5618419, -4.03277753,
 -2.70584515, -1.10858013, -1.61158534, -1.68130234, -1.45925973, -2.68950916,
 -2.79277004, -1.88262046, -1.40505877, -2.21963341, -2.04068837, -3.00133505,
 -2.21992579, -3.12856726, -2.58021589, -1.62513441, -1.38604897, -1.51362573,
 -2.25566364, -1.61406873, -1.0153317, -1.85360387, -1.80630432, -2.07345289,
 -1.93103654, -2.49862097, -1.76349746, -2.1298992, -1.80916879, -3.16832383,
 -5.59332091, -1.14429242, -1.87010351, -2.59564254, -1.47673363, -1.2922715,
 -0.96234884, -1.91109783, -2.22535778, -1.80271351, -2.59757653, -2.11488784,
 -2.79615348, -1.7536742, -1.47598131, -5.83516238, -3.48589668, -1.17864577,
 -1.43565769, -2.26288939, -3.30993108, -1.17586267, -2.75953804, -2.4461565,
 -1.36348829, -1.43773212, -2.48290689, -2.04786243, -2.72534538, -2.27232864,
 -2.08175001, -1.8345003, -2.3468446, -2.92265787, -1.49162983, -1.60214014,
 -3.08305393, -0.99900345, -3.1781902, -1.15244191, -5.04278189, -0.96535059,
 -1.41211923, -2.15288795, -2.30826709, -4.20806702, -2.37590817, -1.43207572,
 -1.91537347, -0.95009799, -2.5980243, -2.45175154, -3.92277177, -2.85671865,
 -1.58637683, -2.5764404, -2.08453045, -1.33623315, -1.35174576, -2.78978737,
 -5.08351666, -1.933459, -0.99620505, -1.00624576, -1.74215811, -8.01509903,
 -1.08825007, -2.0378159, -2.72353339, -1.9166893, -1.3763935, -2.94626297,
 -2.61224128, -1.43918016, -2.33608867, -1.21240287, -2.51972785, -2.38931699,
 -2.22677916, -2.20828412, -1.41079291, -1.1147447, -3.38543888, -4.6671017,
 -2.25474493, -1.91855516, -1.23912108, -2.26197829, -1.5899188, -3.52805363,
 ...
 -1.10124684, -1.08899235, -0.99726571, -2.85781293, -7.2339675, -3.47261716,
 -3.98460437, -1.0371552, -1.35785851, -2.80623583, -1.57769946, -1.06214429,
 -2.07913459, -1.78010699, -3.1686996, -1.66845292, -1.96161519, -2.62642614,
 -1.24109268, -1.68223608]