Artificial Intelligence and Machine Learning - Unit II
12 July 2022

First and last name, Student ID: ______________________________ Seat: ______

1. Answer the following questions by reporting also the procedure that leads you to the numerical values not just the values itself.

   (a) In Eq. (1) left, $\mathbf{X}$ is a design matrix where each row is a sample. Each row indicates RGB pixel triplet. What is the dimensionality of the samples in $\mathbf{X}$? Complete the $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ to compute the empirical RGB average and the covariance matrix associated to $\mathbf{X}$. $\boxed{1}$

$$\mathbf{X} = \begin{bmatrix} 172 & 47 & 117 \\ 192 & 67 & 251 \\ 195 & 103 & 9 \\ 211 & 21 & 242 \\ 36 & 87 & 70 \end{bmatrix} \quad \boldsymbol{\mu} = \begin{bmatrix} \quad\quad\quad\quad\quad\quad \end{bmatrix} \quad \boldsymbol{\Sigma} = \begin{bmatrix} \quad\quad\quad\quad\quad\quad \\ \\ \\ \end{bmatrix} \tag{1}$$

Solution: The number of samples is 5 and dimensionality is 3. $\boldsymbol{\Sigma}$ is $\frac{1}{N}(\mathbf{X}-\boldsymbol{\mu})^T(\mathbf{X}-\boldsymbol{\mu})$

$$\boldsymbol{\mu} = [161.2 \;\; 65. \;\; 137.8] \quad \boldsymbol{\Sigma} = \begin{bmatrix} 4072.56 & -758.8 & 2517.24 \\ -758.8 & 838.4 & -2074. \\ 2517.24 & -2074. & 9058.16 \end{bmatrix} \tag{2}$$

The dimension of 3 needs to be the same for $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$. If they do not match, there is a problem.

   (b) Let's say that you have a RGB image $\mathbf{I}$ and each color channel is quantized over 8 bit. Colors range from 0 to 2^8-1 with height 640 pixels and width 480 pixels. $\mathbf{I}$ shape is thus $640 \times 480 \times 3$. ◇ How many images are possible to be sampled in total in the space where the image lives in? ◇ You want to transform the image into grayscale. Assume that you can do that by computing the mean across color channel. Write the `one-liner numpy` code that implements that. Please, be specific on the axis. ◇ What is the size of the tensor after you have computed the mean? $\boxed{3}$

Solution: The range of colors consist of 256 possibiles integer values that can be arranged in a tensor $640 \times 480 \times 3$. Each cell of the tensor can take 256 possible values. The total number of images that live in this space is thus $256^{640\times480\times3}$. We can do it as `I = I.mean(axis=2)`. The size after will be 640×480.

   (c) A point cloud $\mathbf{X} \doteq \{\mathbf{x}_i\}_{i=1}^N$ is sampled from a multi-variate Gaussian distribution in 2D and shown in Fig. 1 (a), the black diamond indicate the mean of the distribution. Fig. 1 (b) shows the same point cloud after "sphering" the data. Describe the mathematical process of sphering in the data. What happens to the covariance matrix of the point cloud in Fig. 1 (b)? *Though the example is in 2D, the solution should work in N dimensions too.* $\boxed{3}$

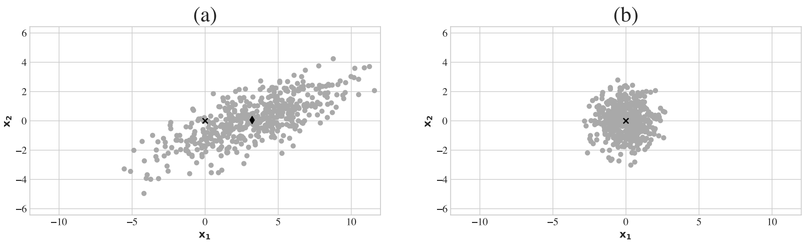

Figure 1: Point cloud and "Sphering"

Solution: Sphering means that the data is centered, rotated and then scaled on the axis by the square root of the eigenvalues. More formally this is achieved as:

$$\mathbf{X}_{sphr} = \boldsymbol{\Sigma}^{-1/2}\mathbf{U}^T\bar{\mathbf{X}}^T,$$

where $\boldsymbol{\Sigma}$ is the diagonal matrix of eigenvalues and $\mathbf{U}$ are the principal components. Those can be obtained by eigendecomposition of the centered covariance matrix $\frac{1}{N}\bar{\mathbf{X}}^T\bar{\mathbf{X}}$. We obtain $\bar{\mathbf{X}}$ by subtracting the empirical mean of $\mathbf{X}$. After the transformation the covariance matrix is the identity matrix.

Total for Question 1: 7

2. We are in the i-th step of the Expectation-Maximization (EM) for learning the parameters of a GMM. Let us assume the Expectation part just finished. The responsibilities $\boldsymbol{\gamma}$ for each training point x are given in Tab. 1 along with the training points x. Assume the estimate for GMM is maximum likelihood.

x	18	8	-6	-13	0
$\boldsymbol{\gamma}$	[0.8, 0.1, 0.1]	[0.3, 0.2, 0.5]	[0.1, 0.7, 0.2]	[1, 0, 0]	[0.99, 0.005, 0.005]

Table 1: Training set of a GMM with responsibilities.

(a) How many modes does the GMM described above have? Please, motivate your answer. $\boxed{1}$

Solution: The modes are three given that $\boldsymbol{\gamma}$ size is 3; each index is associated to each mode.

(b) Define the responsibilities $\boldsymbol{\gamma}$ from a mathematical point of view and explain which kind of information they provide. For an arbitrary point $\mathbf{x}$ associated to a responsibility $\boldsymbol{\gamma}$ what is the meaning of $\boldsymbol{\gamma}[0] = 0.8$, where 0 indexes the first value of the $\boldsymbol{\gamma}$ vector? $\boxed{2}$

Solution: $\boldsymbol{\gamma}_k$ indicates the probability that a point x may have been generated by the k-th Gaussian. It is defined as:

$$\boldsymbol{\gamma}_k = p(z == k|x) = \frac{\mathcal{N}(x; \mu_k, \sigma_k)\pi_k}{\sum_{k\in\{0,1,2\}} \mathcal{N}(x; \mu_k, \sigma_k)\pi_k}$$

where π_k indicates the mixing coefficients. $\boldsymbol{\gamma}[0] = 0.8$ means that there is 80% probability that the point x is associated to Gaussian with $k = 0$.

(c) Given the responsibilities and the training point defined in Tab. 1, compute the Maximization Step, that is, estimate the probability density function (pdf) of the GMM at that step. Please, write the equations you are using for the computation. *(Hint: to compute the pdf you just have to find the parameters of the GMM and then say it distributes according to e.g.,* $\frac{1}{5} \cdot \mathcal{N}(\frac{1}{2}, 29) + \ldots$ *etc.)* $\boxed{3}$

Solution: We have to estimate $\{\pi_k, \mu_k, \sigma_k\} \quad \forall\ k = 0..2$. In this case the responsibility index k tells how much to weight that point in the maximization step. First, we compute $N_k = \sum_i^N \gamma_{ik}$ the summed weights for k-th mode where $N = 5$ is the total number of points. Then $\pi_k = \frac{N_k}{N}$ and $\mu_k = \frac{1}{N_k}\sum_i^N \gamma_{ik} \cdot x_i$ while $\sigma_k^2 = \frac{1}{N_k}\sum_i^N \gamma_{ik} \cdot (x_i - \mu_k)^2$.

	$k = 0$	$k = 1$	$k = 2$
N_k	3.19	1.005	0.805
π	0.638	0.201	0.161
μ	1.003	-0.796	5.714
σ^2	140.37	69.41	56.29

(d) You have written your own implementation of EM to fit GMM and you are fitting some data, but at some point, you suddenly see that the log-likelihood goes to infinity—you see `np.nan` or `np.inf` appearing in the log-likelihood. What may be happened? *(Hint: division by zero is a good starting point but needs to be motivated in to the context of GMM.)* $\boxed{1}$

Solution: Irrespective of numerical errors, it may be that we have encountered a singularity. That is the learning places infinity mass over a training point. This causes one of mode to be centered exactly on a training point and the standard deviation σ collapses to zero. The distribution becomes something similar to Delta Dirac. Given that the σ is at the denominator in the Gaussian pdf, the log-likehood goes to infinity.

Total for Question 2: 7

3. We have 2D training points below to be used for binary classification; each point is paired with its label $y \in \{0, 1\}$:

$$\mathbf{a} = [1/4, \sqrt{2}]; y_a = 1 \qquad \mathbf{b} = [4, -3\sqrt{2}]; y_b = 0 \qquad \mathbf{c} = [-2, 2\sqrt{2}]; y_c = 1 \tag{3}$$

(a) Irrespective of Eq. (3), give a generic definition of the distance between two points in D dimensions $\mathbf{x} \in \mathbb{R}^D$ and $\mathbf{z} \in \mathbb{R}^D$ using $\diamond$ ℓ_2 (Euclidean), $\diamond$ ℓ_1 (Manhattan) and $\diamond$ ℓ_∞ norms. $\boxed{2}$

Solution: Assume $\mathbf{x}_i$ indicates i-th value of the vector, then: $\ell_2 = \sqrt{\sum_{i=1}^{D} (\mathbf{x}_i - \mathbf{z}_i)^2}$ $\ell_1 = \sum_{i=1}^{D} |\mathbf{x}_i - \mathbf{z}_i|$ while $\ell_\infty = \max_i\{|\mathbf{x}_i - \mathbf{z}_i|\}$ (greatest component of the absolute difference.)

(b) Using the training set specified above, classify a new point $\mathbf{x}' = [0, 0]$ using **K Nearest Neighbor (K-NN)** with k=2 using: $\diamond$ ℓ_2 $\diamond$ ℓ_1 and $\diamond$ ℓ_∞ norms to measure the distance between points. Show your computation of the distances and the way you classify $\mathbf{x}'$. $\boxed{2}$

Solution: Basic k-NN computes all distances of $\mathbf{x}'$ wrt to the training set and sort the training labels based on the distances in ascending order. Clip the sorted labels to first $k = 2$. Take majority vote. In this case $\mathbf{x}' = [0, 0]$ is the center of the space and thus the distance is simply the norm of the training point.

$\diamond$ $\boldsymbol{\ell_2}$: $d(\mathbf{x}', \mathbf{a}) = \sqrt{(\frac{1}{16} + 2)} = \frac{\sqrt{33}}{4}$ $\quad d(\mathbf{x}', \mathbf{b}) = \sqrt{16 + 18} = \sqrt{34}$ $\quad d(\mathbf{x}', \mathbf{c}) = \sqrt{4 + 8} = \sqrt{12}$. The order is thus $[\mathbf{a}, \mathbf{c}, \mathbf{b}]$ and labels of a, c is both +1. Thus will be classified as +1.

$\diamond$ $\boldsymbol{\ell_1}$: $d(\mathbf{x}', \mathbf{a}) = \frac{1}{4} + \sqrt{2}$ $\quad d(\mathbf{x}', \mathbf{b}) = 4 + 3\sqrt{2}$ $\quad d(\mathbf{x}', \mathbf{c}) = 2 + 2\sqrt{2}$ The order is thus $[\mathbf{a}, \mathbf{c}, \mathbf{b}]$ and labels of a, c is both +1. Thus will be classified as +1.

$\diamond$ $\boldsymbol{\ell_\infty}$: $d(\mathbf{x}', \mathbf{a}) = \sqrt{2}$ $\quad d(\mathbf{x}', \mathbf{b}) = 3\sqrt{2}$ $\quad d(\mathbf{x}', \mathbf{c}) = 2\sqrt{2}$ The order is thus $[\mathbf{a}, \mathbf{c}, \mathbf{b}]$ and labels of a, c is both +1. Thus will be classified as +1.

(c) Let's assume you have a k-NN algorithm and you want to cross-validate $k = \{7, 11, 21\}$ and three types of distanced ℓ_1, ℓ_2 and ℓ_∞. How many models do you have to train in total, assuming you do 10-fold cross-validation? Moreover, what is the complexity for training a k-NN classifier? $\boxed{2}$

Solution: We have to grid search for 3 values for k and for each of the k selected value, we have to search for 3 distances. Now given a fixed k and a fixed distance, we have to do 10-fold cross-validation. So within a selected pair (k, dist) we have to train 10 models. Given that the pairs are 3×3 in total we have to to train $3 \times 3 \times 10$ models. The k-NN classifier is a lazy learner and does not perform any training, just store the data so we can say training complexity is $\mathcal{O}(1)$, without considering loading the data etc.

Total for Question 3: 6

4. We want to perform some evaluation of a binary classifier.

y	1	0	1	1	0	0	0
s	0.7	-5	0.3	0.1	-1	5	0

Table 2: Labels and unnormalized scores for a binary classifier.

(a) Give a definition of True Positive Rate (TPR) and False Positive Rate (FPR). Given a binary classifier with unnormalized scores s—the higher the score, the more correlates with y—compute the ROC curve for the values in Tab. 2 by showing the TPR and FPR in a table. $\boxed{3}$

Solution: The True Positive Rate is the ratio between the positive samples that are correctly classified (TP) divided by the number of ground-truth positive we have in the population (P_{gt}). The False Positive Rate is the ratio between the negative samples that are incorrectly classified as positive FP divided by the total number of ground-truth negative in the population N_{gt}.

$$TPR = \frac{TP}{P_{gt}} \quad FPR = \frac{FP}{N_{gt}}.$$

In Tab. 2 N_{gt} is 4 and P_{gt} is 3, so the denominator in the ROC computation will be always the same. We just have to sort the score in descending order and compute the TPR, FPR for each score. We start from maximum score plus 1 to make sure the first point in in the ROC is TPR=0@FPR=0.

thrs	6.	5.	0.7	0.3	0.1	0.	-1.	-5.
FPR	$\frac{0}{4}$	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{2}{4}$	$\frac{3}{4}$	$\frac{4}{4}$
TPR	$\frac{0}{3}$	$\frac{0}{3}$	$\frac{1}{3}$	$\frac{2}{3}$	$\frac{3}{3}$	$\frac{3}{3}$	$\frac{3}{3}$	$\frac{3}{3}$

(b) Compute the Area Under the Curve (AUC) of the above ROC. $\boxed{2}$

Solution: We have all zero in both TPR and FPR until score=0.7, but there are lower scores that for the same FPR=1/4 give TPR=1. So we have 3 step of width 1/4 multiplied by 3/3 height, thus the area is $3 \cdot \frac{1}{4} \cdot 1 = \frac{3}{4}$.

(c) Let's assume that in Tab. 2 we replace the score $s = -5$ with $s = -2.2$; we also replace $s = 0.3$ with $s = 0.22$. Is the ROC going to change? Do we have to recompute it? Motivate your answer. $\boxed{1}$

Solution: The sorted thresholds are $[6., 5., 0.7, 0.3, 0.1, 0., -1., -5.]$. Even if we replace -5 with -2.2, the order will not change: that point will still be lower than -1. Same holds for the other swaps. Thus ROC will not change and no need to recompute it.

(d) Alice works for `IseekU`, a biometric company using AI, and she is happy since she developed a "perfect" classifier: it achieves 99.17% AUC in the validation set over $10K$ samples. Alice says "it is ready to be employed in practice since it will never generate false alarm". What would you tell Alice? What Alice should measure if the company wants a quota "X" on the false alarms? $\boxed{2}$

Solution: Though the AUC shows a very high value computed over a large set of points (10K), it is not guaranteed that will not generate false alarm. In fact, if they wanted to see the impact of the false alarm and they have a quota of X on those, Alice should measure the TPR@FPR=X%. That is, should fix the FPR to X% and report the TPR at that level of FPR. In this case the company may have a sense of what is the true positive rate given the "X" false alarms they can tolerate.

Total for Question 4: 8

5. We have to analyze a neural network in the form of a multi-layer perceptron (MLP). The neural network details follow in the sub questions below.

(a) Deduce and write how many trainable parameters you have with a MLP with input feature vectors with dimension equal to 1024, a first layer with 512 units/neurons, a second layer with 256 units/neurons, and a final multi-class classification layer with 3 units/neurons. Assume all layers have the bias term. The network uses ReLU activation function after each layer except the classification that uses softmax. Write down the equation for the computation, not just the final value. Moreover, how many classes does the network classify? $\boxed{1}$

Solution: The network maps 1024 to 512 via a matrix plus a 512x1 bias vector. So first layer is $1024 \times 512 + 512$. We can repeate this for all layers and the total number is:

$$\underbrace{\overbrace{1024 \times 512}^{\text{matrix}} + \overbrace{512}^{\text{bias}}}_{\text{1st layer}} + \underbrace{512 \times 256 + 256}_{\text{2nd layer}} + \underbrace{256 \times 3 + 3}_{\text{3rd layer}}$$

Given that we have to 3 units in the last classification layer we have 3 classes to classify.

(b) Give a definition of the <u>softmax function</u> and <u>cross-entropy loss</u> used for training neural nets for classification. $\boxed{3}$

Solution: Given the logit response after the classification layer $\mathbf{z} \doteq \mathbf{W}\mathbf{x} + \mathbf{b}$, the softmax transforms it to a probability vector over K classes, for the j-th class as:

$$\mathbf{p}_j = \frac{e^{\mathbf{z}_j}}{\sum_{k=1}^{K} e^{\mathbf{z}_k}}.$$

The cross-entropy measures the deviation from the one-hot encoding distribution of the class labels $\mathbf{y}$ compared to the response of the network $\mathbf{p}$ as:

$$-\sum_{k=1}^{K} \mathbf{y}_k \log(\mathbf{p}_k).$$

Given $\mathbf{y}$ has only a 1 in correct class label, it functions as a class selector where y selects the index that contains the 1 in $\mathbf{y}$: $-\log(\mathbf{p}_{k==y})$.

(c) Let us suppose that in the previous network we arrive at the last classification layer and define $\mathbf{z} \doteq \mathbf{W}\mathbf{x} + \mathbf{b}$ as the response of the linear layer for classification. $\mathbf{z}$, as you may know, has to go through the softmax function. Given the vector $\mathbf{z}$ below and a $\mathbf{y}$ as one-hot encoding for class label $\diamond$ compute the probability values after softmax, by filling in the values in $\mathbf{p}$ $\diamond$ compute the value of the cross-entropy loss $\mathcal{L}$, given $\mathbf{p}$ and $\mathbf{y}$. When computing cross-entropy use natural logarithm. $\boxed{3}$

$$\mathbf{z} = \begin{bmatrix} -0.5 \\ 0.5 \\ -1 \end{bmatrix} \quad \mathbf{p} = \begin{pmatrix} \_\_\_\_ \\ \_\_\_\_ \\ \_\_\_\_ \end{pmatrix} \quad \mathbf{y} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \quad \mathcal{L} = \_\_\_\_$$

What is the value of the loss instead if the network perform random guess?

Solution: The softmax normalization to make it a probability can be precomputed and is $D = \exp(0.5) + \exp(-0.5) + \exp(-1)$

$$\mathbf{z} = \begin{bmatrix} -0.5 \\ 0.5 \\ -1 \end{bmatrix} \quad \mathbf{p} = \begin{pmatrix} \exp(-0.5)/D \approx 0.23 \\ \exp(0.5)/D \approx 0.62 \\ \exp(-1)/D \approx 0.14 \end{pmatrix} \quad \mathbf{y} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \quad \mathcal{L} = -\ln(0.62) \approx 0.478.$$

Random guess is uniform probability over $K = 3$ classes thus $-\ln(\frac{1}{3}) \approx 1.098$

Total for Question 5: 7

You can use this space for writing. Summary for points is at the bottom.

Question:	1	2	3	4	5	Total
Points:	7	7	6	8	7	35
Score:						