Introduction to ML (CS771), 2024-2025-Sem-I	**Total Marks**	**100**
End-sem exam. November 21, 2024	**Duration**	**3 hours**
Name	**Roll No.**	

Instructions:

1.	Clearly write your name (in block letters) and roll number in the provided boxes above.
2.	Write your final answers concisely in the provided space. You may use blue/black pen.
3.	We may not be able to provide clarifications during the exam. If any aspect of some question appears ambiguous/unclear to you, please state your assumption(s) and answer accordingly.
4.	The booklet has a total of 6 pages and 9 questions.

1.1 Given N i.i.d. coin toss outcomes $y_1, y_2, \dots, y_N$ from the distribution Bernoulli$(y|\mu)$, is the MLE solution for μ guaranteed to be unique? Justify your answer. **(10 marks)**

The log-likelihood is $L = \sum_{n=1}^{N}[y_n \log\mu + (1-y_n)\log(1-\mu)]$.

Its first derivative is $\frac{\partial L}{\partial \mu} = \sum_{n=1}^{N}\left[\frac{y_n}{\mu} - \frac{(1-y_n)}{(1-\mu)}\right]$

The second derivative is $\frac{\partial^2 L}{\partial \mu^2} = \sum_{n=1}^{N}\left[-\frac{y_n}{\mu^2} - \frac{(1-y_n)}{(1-\mu)^2}\right]$ which is always negative, so the log-likelihood is concave and thus there is a unique maxima of the log-likelihood function

1.2 Consider a kernelized LwP classifier with training data $D = \{(\boldsymbol{x}_n, y_n)\}_{n=1}^{N}$ from C classes. Given a test input $\boldsymbol{x}_*$, give the expression (with brief derivation) for its predicted label y_* assuming a kernel k. Your expression must only be in terms of D, $\boldsymbol{x}_*$, and k. **(10 marks)**

The kernelized LwP will use a kernelized Euclidean distance to predict the label. The rule will be $y_* = \text{argmin}_{k\in\{1,2,\dots,C\}}\|\phi(\boldsymbol{\mu}_k) - \phi(\boldsymbol{x}_*)\|^2$ where $\phi(\boldsymbol{\mu}_k) = \frac{\sum_{n:y_n=k}\phi(\boldsymbol{x}_n)}{N_k}$ with N_k being the number of inputs from label $y_n = k$. It is important to note that because ϕ is a nonlinear mapping, $\phi(\boldsymbol{\mu}_k) \neq \phi\left(\frac{\sum_{n:y_n=k}\boldsymbol{x}_n}{N_k}\right)$

Expanding the kernelized Euclidean distance expression

$$\|\phi(\boldsymbol{\mu}_k) - \phi(\boldsymbol{x}_*)\|^2 = \phi(\boldsymbol{\mu}_k)^\top\phi(\boldsymbol{\mu}_k) + \phi(\boldsymbol{x}_*)^\top\phi(\boldsymbol{x}_*) - 2\phi(\boldsymbol{\mu}_k)^\top\phi(\boldsymbol{x}_*)$$

Substituting the expression for $\phi(\boldsymbol{\mu}_k)$, we get

$$\phi(\boldsymbol{\mu}_k)^\top\phi(\boldsymbol{\mu}_k) = \frac{\sum_{n:y_n=k}\sum_{m:y_m=k}\phi(\boldsymbol{x}_n)^\top\phi(\boldsymbol{x}_m)}{N_k^2} = \frac{\sum_{n:y_n=k}\sum_{m:y_m=k}k(\boldsymbol{x}_n,\boldsymbol{x}_m)}{N_k^2}$$

$$\phi(\boldsymbol{\mu}_k)^\top\phi(\boldsymbol{x}_*) = \frac{\sum_{n:y_n=k}\phi(\boldsymbol{x}_n)^\top\phi(\boldsymbol{x}_*)}{N_k} = \frac{\sum_{n:y_n=k}k(\boldsymbol{x}_n,\boldsymbol{x}_*)}{N_k}$$

$$\phi(\boldsymbol{x}_*)^\top\phi(\boldsymbol{x}_*) = k(\boldsymbol{x}_*,\boldsymbol{x}_*)$$

1.3 Consider an "online" K-means clustering where we have already received $n-1$ inputs, using which we have current cluster means $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_K$. Now we receive the next input $\boldsymbol{x}_n$. Use SGD on the K-means loss function to derive updates of the cluster means and provide a brief justification why this SGD based update expression makes intuitive sense. **(10 marks)**

K-means loss function $\mathcal{L} = \sum_{n=1}^{N} \sum_{k=1}^{K} z_{nk} \|\boldsymbol{x}_n - \boldsymbol{\mu}_k\|^2$. Suppose $\boldsymbol{x}_n$ is assigned to its closest cluster with mean $\boldsymbol{\mu}_k$. Thus $z_{nk} = 1$ and $z_{n\ell} = 0, \forall \ell \neq k$

Considering a single data point $\boldsymbol{x}_n$, the stochastic gradient of the K-means loss function w.r.t. $\boldsymbol{\mu}_k$

$$\boldsymbol{g} = -2 \sum_{k=1}^{K} z_{nk}(\boldsymbol{x}_n - \boldsymbol{\mu}_k) = -2(\boldsymbol{x}_n - \boldsymbol{\mu}_k)$$

We will therefore update $\boldsymbol{\mu}_k$ using SGD as

$$\boldsymbol{\mu}_k^{new} = \boldsymbol{\mu}_k^{old} + 2\eta\left(\boldsymbol{x}_n - \boldsymbol{\mu}_k^{old}\right) = (1-2\eta)\boldsymbol{\mu}_k^{old} + 2\eta \boldsymbol{x}_n$$

The update makes intuitive sense since it is moving the mean of cluster k towards the x_n if this data-point is assigned to cluster k.

1.4 Given labeled data $D = \{(\boldsymbol{x}_n, y_n)\}_{n=1}^{N}$ with binary labels, consider projecting each input along a direction $\boldsymbol{w} \in \mathbb{R}^D$. Suppose we want that, after projection, the two class means should be far apart and the inputs within the same class should have a small spread (variance). Write down the expression of an optimization problem that will help achieve this goal. **(10 marks)**

The projected inputs are given by $\{\boldsymbol{w}^\top \boldsymbol{x}_1, \boldsymbol{w}^\top \boldsymbol{x}_2, \ldots, \boldsymbol{w}^\top \boldsymbol{x}_N\}$.

The class means for the projected inputs will be

$$\boldsymbol{\mu}_+ = \frac{\sum_{n:y_n=+1} \boldsymbol{w}^\top \boldsymbol{x}_n}{N_+} \text{ and } \boldsymbol{\mu}_- = \frac{\sum_{n:y_n=-1} \boldsymbol{w}^\top \boldsymbol{x}_n}{N_-}$$

The class spreads will be

$$\sigma_+^2 = \frac{\sum_{n:y_n=+1}\left(\boldsymbol{w}^\top \boldsymbol{x}_n - \mu_+\right)^2}{N_+} \text{ and } \sigma_-^2 = \frac{\sum_{n:y_n=-1}\left(\boldsymbol{w}^\top \boldsymbol{x}_n - \mu_-\right)^2}{N_-}$$

We want $\boldsymbol{\mu}_+$ and $\boldsymbol{\mu}_-$ to be far apart and want the sum of within class spreads, i.e., $\sigma_+^2 + \sigma_-^2$ to be small. This can be accomplished by maximizing the following objective

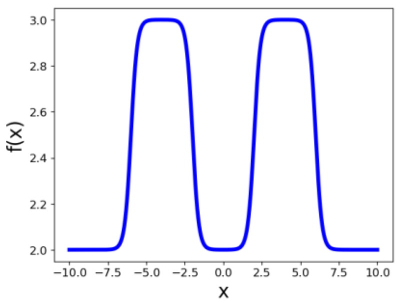
$$\hat{\boldsymbol{w}} = \text{argmax}_{\boldsymbol{w}} \frac{\|\boldsymbol{\mu}_+ - \boldsymbol{\mu}_-\|^2}{(\sigma_+^2 + \sigma_-^2)}$$

1.5 Consider a 4×4 input grid I and a 2×2 filter F as shown below. Perform a convolution followed by 2×2 max-pooling, and compute the feature map produced as a result of convolution as well as the final feature map produced after pooling. You may simply write the final values for both feature maps (no need to show all the basic operations). Assume no zero padding, and row/column strides of 1 for both convolution and max-pooling. **(8 marks)**.

$$I = \begin{bmatrix} 1 & 2 & 3 & 0 \\ 4 & 5 & 6 & 1 \\ 7 & 8 & 9 & 2 \\ 0 & 1 & 0 & 3 \end{bmatrix} \qquad F = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

Feature map after convolution: $\begin{bmatrix} -4 & -4 & 2 \\ -4 & -4 & 4 \\ 6 & 8 & 6 \end{bmatrix}$. After pooling: $\begin{bmatrix} -4 & 4 \\ 8 & 8 \end{bmatrix}$

1.6 We would like to learn the function $f(x)$ shown on the right using only a "single layer" combination of multiple sigmoid functions. What is the minimum number of sigmoid functions that you will need to construct this function? Clearly write down the expression of $f(x)$ in terms of these sigmoid functions. If we double the number of peaks in $f(x)$, will the same number of sigmoids suffice? Briefly justify your answer. **(12 marks)**



Each of the bumps can be constructed by superimposing a sigmoid with another sigmoid that is a "shifted and flipped" version of it.

The given function has two such bumps so we will need two pairs, and thus total 4 sigmoids.

The function $f(x)$ can thus be expression in the following manner:

$$f(x) = \sigma(a(x+b_1)) + \sigma(-a(x+b_2)) + \sigma(a(x+b_3)) + \sigma(-a(x+b_4))$$

The scale value a will control the rate at which the sigmoid increases from small to high value, and offsets b_1, b_2, b_3, b_4 will control the location around which it starts to increase rapidly.

The sigmoid first will cause the rise of the first bump, the second will cause the decline of the first bump, the third will cause the rise of the second bump, and the fourth will cause the decline of the second bump. If we want to double the number of peaks to 4 then just 4 sigmoids won't suffice but 8 sigmoid will be needed (2 for each bump).

1.7 Assume N observations $\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_N$ each drawn i.i.d. from a D-dimensional Gaussian $N(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{I})$. Derive the MAP estimate of $\boldsymbol{\mu}$ assuming a Gaussian prior $N(\boldsymbol{\mu}|\boldsymbol{\mu}_0, \boldsymbol{I})$ and, from the final expression of the MAP estimate of $\boldsymbol{\mu}$ that you obtain, provide a precise and brief interpretation of the role played by the prior's mean $\boldsymbol{\mu}_0$. **(15 marks)**

The PDF of D dimensional Gaussian $N(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{I}) \propto \exp\left(-0.5(\boldsymbol{x}_n - \boldsymbol{\mu})^\top(\boldsymbol{x}_n - \boldsymbol{\mu})\right)$
The PDF of D dimensional Gaussian $N(\boldsymbol{\mu}|\boldsymbol{\mu}_0, \boldsymbol{I}) \propto \exp\left(-0.5(\boldsymbol{\mu} - \boldsymbol{\mu}_0)^\top(\boldsymbol{\mu} - \boldsymbol{\mu}_0)\right)$
The MAP objective is equal to the sum of the negative log-likelihood + the negative of the log prior. Ignoring terms that don't depend on $\boldsymbol{\mu}$, it is given by

$$L(\boldsymbol{\mu}) = \sum_{n=1}^{N} (\boldsymbol{x}_n - \boldsymbol{\mu})^\top(\boldsymbol{x}_n - \boldsymbol{\mu}) + (\boldsymbol{\mu} - \boldsymbol{\mu}_0)^\top(\boldsymbol{\mu} - \boldsymbol{\mu}_0)$$

Note that the above MAP estimation is equivalent to doing MLE with N+1 observations $\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_N, \boldsymbol{\mu}_0$ where $\boldsymbol{\mu}_0$ acts as the $(N+1)^{th}$ observation (or rather a "pseudo-observation"). As we know, the MLE solution for the mean of this Gaussian would be simply the mean of the N+1 observations. Thus

$$\boldsymbol{\mu}_{MAP} = \frac{\boldsymbol{\mu}_0 + \sum_{n=1}^{N} \boldsymbol{x}_n}{N+1}$$

1.8 Suppose there are N i.i.d. coin toss outcomes $y_1, y_2, \dots, y_N$ from the distribution Bernoulli$(y|\mu)$, where we only recorded the true values of the first M toss outcomes and the true values of the remaining $N - M$ toss outcomes are hidden ("latent"). Now consider two approaches to do MLE for μ: (1) Using only the first M outcomes, and (2) Using an "EM-like" procedure where we initialize μ using the MLE solution of the first approach and then alternate between estimating the **expectations** of each of the $N - M$ hidden coin toss outcomes and re-estimating μ using MLE the true values of the first M outcomes and the **expected** values of the remaining $N - M$ outcomes (and repeating this till convergence). Does the "EM like" approach 2 (which uses seemingly more data than approach 1) give a different MLE solution as compared to approach 1? Briefly justify your answer. **(15 marks)**

MLE using the first M toss outcomes will be given by $\hat{\mu} = \frac{\sum_{i=1}^{M} y_i}{M}$

MLE using the first M toss outcomes and expected value of the remaining $N - M$ outcomes will be given by $\mu = \frac{\sum_{i=1}^{M} y_i + \sum_{i=M+1}^{N} \mathbb{E}[y_i]}{N} = \frac{M}{N}\hat{\mu} + \frac{(N-M)}{N}\hat{\mu} = \hat{\mu}$

So we see that the "EM-like" approach 2 also gives the same answer.

The reason why we don't get a different answer is that it isn't really a latent variable model in the sense that $y_{M+1}, \dots, y_N$ don't depend on any latent variables but depend directly just on the parameter μ that is being estimated.

1.9 Consider modeling some data $D = \{(\boldsymbol{x}_n, y_n)\}_{n=1}^{N}$, $\boldsymbol{x}_n \in \mathbb{R}^D$, $y_n \in \{0,1\}$, using a mixture of logistic regression models, where we model each binary label y_n by first randomly picking one of the K logistic regression models with weight vectors $\boldsymbol{w}_1, \boldsymbol{w}_2, \dots, \boldsymbol{w}_K$, based on the value of a latent variable z_n drawn from $p(z_n|\boldsymbol{\pi}) = \text{multinoulli}(z_n|\pi_1, \pi_2, \dots, \pi_K)$, and then generating y_n conditioned on z_n from $p(y_n|z_n, \boldsymbol{x}_n) = \text{Bernoulli}(y_n|\sigma(w_{z_n}^\top \boldsymbol{x}_n))$

Now consider the marginal probability of the label $y_n = 1$, given $\boldsymbol{x}_n$, i.e., $p(y_n = 1|\boldsymbol{x}_n)$, and show that this can be quantity can also be thought of as the output of a neural network. Clearly specify what is the input layer, hidden layer(s), activations, the output layer, and the connection weights of this neural network. **(10 marks)**

The marginal probability $p(y_n = 1|\boldsymbol{x}_n)$ can be obtained by taking the joint probability distribution $p(y_n = 1, z_n|x_n)$ and summing it over all possibilities of the unknown z_n.

Thus

$$\begin{aligned} p(y_n = 1|\boldsymbol{x}_n) &= \textstyle\sum_{k=1}^{K} p(y_n = 1, z_n = k|x_n) \\ &= \sum_{k=1}^{K} p(y_n = 1|z_n = k, x_n)\, p(z_n = k|\boldsymbol{\pi}) \\ &= \sum_{k=1}^{K} \pi_k \sigma(w_k^\top x_n) \end{aligned}$$

This is equivalent to a neural network with one input layer (with D input features of $\boldsymbol{x}_n$) , one hidden layer with K neurons with sigmoid activations $\{\sigma(\boldsymbol{w}_k^\top x_n)\}_{k=1}^{K}$, and one output layer producing y_n (a binary value). The connection weights between input to the hidden layer are given by $\{\boldsymbol{w}_k\}_{k=1}^{K}$ and the connection weights between hidden layer to output layer are given by $\{\pi_k\}_{k=1}^{K}$