Date: Monday, November 21, 2022. 11:00 am - 1:15 pm
There are 7 questions. All questions are compulsory. Each Question carries 6 points. Max Points: 42. There are four printed pages.
Answer to each question must start on a new page. You need to justify all your answers. Answers without justification may not get full points.

1. **VC-Dimension of Linear Classifiers in N-dimensions**: In the following, we will derive the characteristics of the VC-Dimension of linear classifiers in the n-dimensional Euclidean space $\mathcal{R}^n$. Consider the hypothesis class $\mathcal{H}^n$ such that $h_{w,b} \in \mathcal{H}^n$ takes the form $h_{w,b}(x) = \mathbb{1}\{w^T x + b \geq 0\}$ where where $w \in \mathcal{R}^n$ and $b \in \mathcal{R}$. We will first prove the desired result for set of classifiers represented by hyperplanes passing through the origin, and then generalize the result to the set of arbitrary linear classifiers. Specifically, let $\mathcal{H}_0^n \subset \mathcal{H}^n$ denote that subset of hypothesis represented by hyperplanes passing through the origin, i.e., $\forall h_{w,b} \in \mathcal{H}_0^n$, we have $b = 0$, and $h_{w,b}(x) = \mathbb{1}\{w^T x \geq 0\}$.

   (a) Show that VC-Dim$(\mathcal{H}_0^2) = 2$.

   (b) Show that for a general n, VC-Dim$(\mathcal{H}_0^n) = n$.
   Hint: Here is one way to prove it. (a) To prove VC-Dim is at least n, consider the set of points n, each of which lie along one of the axis. (b) To show, VC-Dim is less than equal to n, prove by contradiction using following steps: (i) Consider $n+1$ points, and assume that they can be shattered by $\mathcal{H}_0^n$. (ii) Consider the matrix XW, where X is the design matrix for $n+1$ points (i.e., each row represents an individual point), and W being the matrix of 2^{n+1} weight combinations which realize the possible labelings of $n+1$ points. (iii) Show that the rows of the matrix XW must be linearly independent by showing that $\forall a \in R^{n+1}, a \neq 0$, $a^T XW$ can never be zero (there will be at least one positive entry). (iv) On the other hand, using the fact that for any two matrices A, B, $Rank(AB) \leq min(Rank(A), Rank(B))$, the $Rank(XW) \leq n$, which is a contradiction. This is just a proof sketch. You have to fill in the required details. On the other hand, you are free to use any other method to prove the desired fact, if you desire so.

   (c) Use the result above to show that VC-Dim $(\mathcal{H}^n) \leq n + 1$.

2. **Principal Component Analysis:** Consider doing PCA over a set of points given as $\{x^{(i)}\}_{i=1}^m$. Let the number of PCA components be given by K $(K \leq n)$, and let the corresponding PCA components be given as $\{\mu_1, \mu_2, \cdots, \mu_K\}$.

   (a) Show that the k^{th} PCA component μ_k is given by the eigenvector of the empirical co-variance matrix Σ corresponding to the k^{th} largest eigenvalue. You have to prove this fact $\forall k \leq K$. Hint: here is one way to prove it. Start with $k = 1$ and prove this as done in class. Then, use induction over k (you need to carefully define your inductive hypothesis and then prove it).

   (b) Let λ_k denote the k^{th} largest eigenvalue of Σ. Show that the amount of variance in the data captured by the component μ_k is exactly equal to λ_k. Using this fact show that if $K = n$, the projected components will be able to capture the entire

variance of the data in the original space. You can use the fact that trace of a symmetric matrix is equal to the sum of its eigenvalues, where trace is sum of diagonal elements.

3. **Boosting (Adaboost):** In the class, we studied functional gradient boosting over decision trees. In this problem, we will look at a variation of this algorithm, called Adaboost, which works for any general classifier. Specifically, we work with binary classification problem, i.e., target $y \in \{-1, 1\}$, and also assume that $\forall k, h_k \in \mathcal{H}$ (hypothesis class), $h_k(x) \in \{-1, 1\}$. Recall that in boosting, we are looking for an ensemble of classifiers of the form $h_K^{agg}(x) = \sum_{k=1}^{K} \alpha_k h_k(x)$, where the classifiers h_k's and their coefficients are selected in an iterative manner. Whereas in the case of functional gradient boosting, they are obtained by optimizing over residuals, by taking functional gradients of the loss function, in Adaboost, they are obtained by simply choosing the next classifier h_{k+1} which minimizes the overall loss in the next iteration. Specifically, the loss function on an example (x, y) takes the form $L(y, h_k^{agg}(x)) = e^{-yh_k^{agg}(x)}$, where h_k^{agg} represents the current aggregate of the classifiers (up to to step k). Then, given a set of m examples $\{(x^{(i)}, y^{(i)})\}_{i=1}^m$, h_{k+1} is chosen to minimize $\sum_{i=1}^m L(y^{(i)}, h_{k+1}^{agg}(x^{(i)}))$ given h_k^{agg}. Once the classifier h_{k+1} has been chosen, α_{k+1} is picked which minimizes the overall loss (after adding h_{k+1} to the mix).

   (a) Show that Adaboost can be seen as at each step selecting a classifier which minimizes the **classification error** of a modified training set with weighted set of examples, where the i^{th} example has a weight $w^{(i)}$ (note that in the beginning $w^{(i)} = 1, \forall i$). Express $w^{(i)}$ in terms of the loss of the current aggregate classifier h_k^{agg} on the i^{th} example.

   (b) Compute the value of the coefficient α_{k+1} in terms of the average training error $\hat{\epsilon}_{k+1} = \frac{1}{m} \sum_{i=1}^m w^{(i)} \mathbb{1}\{h_{k+1}(x^{(i)}) \neq y^{(i)}\}$ of the newly added classifier h_{k+1} on the weighted set of examples, where the $w^{(i)}$'s are as computed in the previous step.

4. **Naïve Bayes with Cont. and Count based Attributes:** Consider learning a Naïve bayes model with a mixed set of $(n + n')$ attributes, where the first n attributes are continuous and the next n' attributes are count based (i.e., $x_j \in \mathcal{R}$ for $j : 1 \leq j \leq n$, and, $x_j \in \{0, 1, 2, 3, \ldots\}$, for $j : n < j \leq n+n'$). Let the target variable $y \in \{1, 2, \cdots r\}$, r being the total number of classes. We make the following distributional assumptions:

   - For the target variable, $y \sim Multonomial(\phi)$, $\sum_{k=1}^r \phi_k = 1$.
   - For continuous attributes, $(x_j | y = k) \sim \mathcal{N}(\mu_{jk}, {\sigma_{jk}}^2)$. (Normal Distribution)
   - For count based attributes, $(x_j | y = k) \sim Poisson(\lambda_{jk})$. (Poisson distribution).

   (a) Assume the number of classes $r = 2$. Derive the form of the decision boundary in terms of the parameters of the model as expressed above. Describe the qualitative shape of the boundary that you obtain.

   (b) Using the part(a) above, derive the form of the decision boundary when $r > 2$. Describe the qualitative shape of the boundary that you obtain.

Recall: (a) For $z \in \mathcal{R}$, if $z \sim \mathcal{N}(\mu, \sigma^2)$, then $P(z) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(z-\mu)^2}{2\sigma^2}}$ (b) For $z \in \{0, 1, 2, 3, \ldots\}$, $z \sim Poisson(\lambda)$, then $P(z = c) = \frac{e^{-\lambda}\lambda^c}{c!}$.

5. **Clustering with K-Means and GMM**: Let $\{x^{(i)}\}_{i=1}^m$ be set of points such that $x^{(i)} \in \mathcal{R}^n$. Assume we are interested in finding a K-sized clustering for these set of points, using K-means (part a), K-Means and GMM (part b) below. Let the cluster representatives discovered by each algorithm be given as $\{\mu_k\}_{k=1}^K$ and let $\{C^{(i)}\}_{i=1}^m$, denote the cluster assignments.

   (a) Consider a slightly modified setting of the K-means algorithm, where in the M-step, instead of optimizing over the standard Euclidean distance, we instead optimize the Manhattan distance[1] to compute the cluster representatives, resulting in the following computation: $\forall k, \mu_k = \arg\min_{\mu_k} \sum_{i=1}^m \mathbb{1}\{C^{(i)} = k\}||x^{(i)} - \mu_k||_1$, where the operator $||\cdots||_1$ denotes the L1-norm. Derive a simplified expression for μ_k. In particular, show that μ_k's can be represented in terms of a very well known statistical quantity as a function of those $x^{(i)}$'s where $C^{(i)} = k$.

   (b) In this question, we will analyze the equivalence of GMM and K-means (standard formulation) algorithms for clustering the set of m points given as above. While we run K-means in the standard setting, we run GMM with the following restrictions: (a) Each cluster has a fixed (not learnt) co-variance matrix given by $\beta * I$, where $\beta \in \mathcal{R}$ is a constant and $I \in \mathcal{R}^{n \times n}$ is the identity matrix (b) During E-step, instead of the soft assignment, we do a hard assignment, i.e., (hard) assign each point to the cluster which maximizes $P(z^{(i)}|x^{(i)}; \Theta)$, where $z^{(i)}$'s are hidden cluster assignments, and Θ denotes the set of model parameters in GMM as discussed in the class. Now, under these additional assumptions, would K-means and GMM necessarily discover the same clustering (i.e., for any given value of β, and any arbitrary set of points to be clustered)? You can assume identical initialization for the two algorithms. Why or why not? Argue. If your answer is no, can you impose additional assumptions on the β parameter, so that, K-means will necessarily discover the same clustering as GMM under the modified setting.

6. **Quadratic Optimization and SVM Dual**: Consider the following quadratic optimization problem in two variables $\alpha_1, \alpha_2 \in \mathcal{R}$:

$$\max_{\alpha_1} \; a\alpha_1^2 + b\alpha_1 + c$$
$$\text{subject to}$$
$$d\alpha_1 + e\alpha_2 = \gamma$$
$$0 \leq \alpha_1, \alpha_2 \leq C$$

   Note that $a, b, c, d, e, \gamma \in \mathcal{R}$. Also, $C \in \mathcal{R}$.

   (a) Derive a set of constraints on d, e, γ (expressed as a function of C) such that above optimization problem has at least some solution. Determine the range of values

[1] Manhattan distance, also equivalently the L1-norm, between two points in the Euclidean space, is simply the sum of the absolute difference between the coordinate values of the two points in each dimension.

that α_1 can take, expressed in terms of d, e, γ and C. Let us refer to this range as $[L, H]$.

(b) Find the optima of the above problem as a function of L, H and the parameters a, b and c.

(c) Argue that the above formulation can be used to solve the (linear) SVM dual soft-margin formulation using dual coordinate descent. Your answer should be mathematically precise.

7. **Backpropagation in RNNs with slight modification**: Consider the following set up. You are given a single training example (x, y). Let y take values from a discrete set, i.e., $y \in \{1, 2, \cdots, K\}$. Assume that x is represented as a sequence $(x^{(1)}, x^{(2)} \cdots x^{(T)})$ such that $x^{(t)} \in \mathcal{R}^n, \forall t, 1 \leq t \leq T$. Assume T to be even. Consider learning the target variable using a (modified) RNN with the following equations:

$$\forall t(\text{odd}),\ h^{(t)} = g(\Theta_I x^{(t)} + \Theta_h^1 h^{(t-1)})$$
$$\forall t(\text{even}),\ h^{(t)} = g(\Theta_I x^{(t)} + \Theta_h^2 h^{(t-1)})$$
$$\hat{y} = Softmax(\Theta_o h^{(T)} + b_o)$$

Assume that $h^{(t)} \in \mathcal{R}^d$. Therefore, $\Theta_I \in \mathcal{R}^{d \times n}$, $\Theta_h^1, \Theta_h^2 \in \mathcal{R}^{d \times d}$ and $\Theta_o \in \mathcal{R}^{K \times d}$. $b_o \in \mathcal{R}^K$. Assume that g is the activation function, applied point-wise. Assume g to be represented by the ReLu function. You can assume $h^{(0)} = \bar{0}$ (0 vector). Assuming the log-likelihood based loss $\mathcal{L}$ between $\hat{y}$ (predicted) and y (target), derive (from the first principles) the expressions for:

(a) $\nabla_{h^{(T)}} \mathcal{L}$ as a function of y

(b) $\nabla_{h^{(t)}} \mathcal{L}$ in terms of $\nabla_{h^{(t+1)}} \mathcal{L}$ (for both t odd and t even)

(c) $\nabla_{\Theta^1_{hj:}} \mathcal{L}$ and $\nabla_{\Theta^2_{hj:}} \mathcal{L}$ in terms of $\nabla_{h^{(t)}} \mathcal{L}$'s, where $\Theta^1_{hj:}$ represents the j^{th} row of matrix Θ_h^1. Similarly for $\Theta^2_{hj:}$

You should do the entire computation using vector calculus. Doing it using single variable calculus may not fetch full points. You can use the fact that if $\hat{y} = softmax(z)$, then:

$$\frac{\delta \hat{y}_{k'}}{\delta z_k} = \hat{y}_k(1 - \hat{y}_k) \text{ if } k = k'$$
$$= -\hat{y}_k \hat{y}_{k'} \text{ o.w.}$$