# E9 205 – Machine Learning For Signal Processing

Practice Final Exam

Date: April 7, 2022

**Instructions**

1. This exam is open book. However, computers, mobile phones and other handheld devices are not allowed.
2. Any reference materials that are used in the exam (other than materials distributed in the course channel) should be pre-approved with the instructor before the exam.
3. No additional resources (other than those pre-approved) are allowed for use in the exam.
4. Academic integrity and ethics of highest order are expected.
5. Notation - bold symbols are vectors, capital bold symbols are matrices and regular symbols are scalars.
6. Answer all questions.
7. All answer sheets should contain your name and SR number in the top.
8. Question number should be clearly marked for each response.
9. Total Duration - **180 minutes**
10. Total Marks - **100 points**

Name - .....

Dept. - .....

SR Number - .....

1. **Output Non-linearity and Loss Function** - Leena and Kiran are using DNNs for a term project. Both of them start with the same data. Let $\{\mathbf{x}_i\}_{i=1}^M$ and $\{\mathbf{y}_i\}_{i=1}^M$ denote the input and desired targets to the DNN, where $M$ is the number of samples. The DNN is trained for a classification task with hard targets $\mathbf{y} \in \mathcal{B}^{C\times 1}$, where $\mathcal{B}$ denotes boolean variable (0 *or* 1), and $C$ is the number of classes. Note that every data point $\mathbf{x}_i$ is associated with only one class label $c_i$ where $c_i \in \{1, 2, .., C\}$ classes. The output of the network is denoted as $\hat{\mathbf{y}}_i$ where $\hat{\mathbf{y}}_i \in \mathcal{R}^{C\times 1}$. Both of them use $L$ layer DNN with each layer having weights $\mathbf{W}^l$ and bias $\mathbf{b}^l$ with $l = 1, .., L$ and a sigmoidal non-linearity at the hidden layers. The only difference in their DNN architectures is that Leena uses a linear output layer with a mean square error cost function while Kiran uses a softmax output layer with a cross entropy cost function.

   More specifically, let $\mathbf{z}^l$ denotes the input vector to the non-linearity at layer $l$ for the DNN and $\mathbf{h}^l$ denote the output vector of the hidden layer $l$. For both DNNs, $\mathbf{h}^l = \sigma(\mathbf{z}^l), l = 1, .., L-1$. For Leena's DNN, $\hat{\mathbf{y}} = \mathbf{h}^L = \mathbf{z}^L$ and for Kiran's DNN,

$$\hat{\mathbf{y}} = \mathbf{h}^L = softmax(\mathbf{z}^L)$$

   The cost function for Leena's DNN and Kiran's DNN are given by (in that order),

$$J_{MSE} = \sum_{i=1}^{M} ||\hat{\mathbf{y}}_i - \mathbf{y}_i||^2$$

$$J_{CE} = -\sum_{i=1}^{M} \mathbf{y}_i^T \log(\hat{\mathbf{y}}_i)$$

   Using this setup, each of them writes the weight and bias parameter updates in gradient descent to implement backpropagation algorithm for their choice of DNN. Is there any connection between the gradient update equation for weights $\mathbf{W}^L$ and bias $\mathbf{b}^L$ parameters for both of them ?

   (**Points 15**)

2. **HMMs on Count data** A Poisson process HMM is one in which the observations are postive integers and the emission probabilities are given by the Poisson density,

$$b_j(o_t = k) = \frac{\lambda_j^k e^{-\lambda_j}}{k\,!}$$

where $j = 1, ..., N$ denotes the HMM states and $o_t, t = 1, ..., T$ denotes the observation sequence.

- Derive the update formula for the Poisson density parameter $\lambda_j$ using the EM algorithm. Assume $N$ states for the HMM.
- How will the update equations be modified if density parameter $\lambda_j$ is shared among all states, i.e., $\lambda_j = \lambda$ for $j = 1, ..., N$.

(**Points 20**)

3. **Combining DNNs** Richa is using DNNs for approximating a scalar function $h(\mathbf{x})$. In order to understand the impact of different initializations, she trains $N$ different DNNs and compare the errors obtained on a held out set. Assume that every DNN makes a random error $\epsilon_i(\mathbf{x}) = (v_i(\mathbf{x}) - h(\mathbf{x}))$. Let $J_i, i = 1..., N$ denote the expected error of $i$ th DNN,

$$J_i = \mathcal{E}\big[(v_i(\mathbf{x}) - h(\mathbf{x}))^2\big]$$

where $v_i(\mathbf{x})$ denotes the output of the $i$ th DNN. Varun proposes an idea of combining the outputs from $N$ DNNs using a simple averaging.

$$v_A(\mathbf{x}) = \frac{1}{N}\sum_{i=1}^{N} v_i(\mathbf{x})$$

He claims that expected error $J_A = \mathcal{E}\big[(v_A(\mathbf{x}) - h(\mathbf{x}))^2\big]$ by using the average output $v_A$ is better than the average expected error , i.e., $J_A \leq J_{avg}$, where,

$$J_{avg} = \frac{1}{N}\sum_{i=1}^{N} J_i$$

(a) Richa is not convinced. She takes the simple case when networks are all unbiased and uncorrelated, i.e., $\mathcal{E}[\epsilon_i(\mathbf{x})] = 0$ and $\mathcal{E}[\epsilon_i(\mathbf{x})\epsilon_j(\mathbf{x})] = 0\ \forall i \neq j = 1, ..., N$. She is able to prove this for the simple case. Would you also be able to ? (**Points 5**)

(b) Richa now considers a generic and more realistic case when the errors from individual DNNs are correlated. She is unable to prove the result for the generic case. But Varun still argues that it is indeed true that $J_A$ is better than $J_{avg}$. Do you think Varun is right ? Justify your answer.
*Remark* - Cauchy's Inequality states $\big(\sum_k a_k b_k\big)^2 \leq \sum_k a_k^2 \sum_k b_k^2$ (**Points 5**)

(c) When they take their discussion and findings to Prof. Srini, he modifies their solution to a weighted average,

$$v_W(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i v_i(\mathbf{x})$$

where $\alpha_i$ are constants and $\sum_{i=1}^{N} \alpha_i = 1$. He claims that the optimal value of these constants can be found using the correlation matrix of errors $\boldsymbol{C} = [C_{ij}]$ where $C_{ij} = \mathcal{E}[\epsilon_i \epsilon_j]$. Varun and Richa are asked to find these optimal values for weights $\alpha_i$. What would be your solution for these weights ? (**Points 10**)

4. **Kernel Feature Extraction** - Harry and Aruna are attempting a non-linear feature extraction $\boldsymbol{\phi}(\mathbf{x})$ for their image processing course project where $\mathbf{x} \in \mathcal{R}^D$ and $\boldsymbol{\phi}(\mathbf{x}) \in \mathcal{R}^M$. They have a sequence of images denoted as $\mathbf{x}_1, ..., \mathbf{x}_l$ in their training dataset. Using the images, Aruna computes the Gram matrix $\mathbf{G}$ where $[G]_{i,j} = k(\mathbf{x}_i, \mathbf{x}_j)$, $k(\mathbf{x}_i, \mathbf{x}_j) = \boldsymbol{\phi}(\mathbf{x}_i)^T \boldsymbol{\phi}(\mathbf{x}_j)$ and $\mathbf{G}$ is a square matrix of dimension $l$.

   (a) Harry constructs a modified Gram matrix $\hat{\mathbf{G}} = (\mathbf{I} - \mathbf{1}_l)\mathbf{G}(\mathbf{I} - \mathbf{1}_l)$ where $\mathbf{I}$ is the $l \times l$ identity matrix and $\mathbf{1}_l$ is a square matrix with all elements equal to $\frac{1}{l}$. He claims that $\hat{\mathbf{G}}$ is the sample covariance matrix of the non-linear features $\boldsymbol{\phi}(\mathbf{x})$. Is Harry right ? Also show that $\hat{\mathbf{G}}$ is positive semidefinite with atleast one zero eigenvalue. (**Points 10**)

   (b) Aruna attempts to perform a $K$-means clustering on the images $\mathbf{x}_1, ..., \mathbf{x}_l$ in the non-linear feature space $\boldsymbol{\phi}(\mathbf{x})$. The K-means clustering involves the iterative computation of,

$$c_i = argmin_{\ k} \ ||(\boldsymbol{\phi}(\mathbf{x}_i) - \boldsymbol{m}_k)||^2$$
$$\boldsymbol{m}_k = \frac{\sum_{i:c_i=k} \boldsymbol{\phi}(\mathbf{x}_i)}{\sum_i \delta_{k,c_i}}$$

   where $\boldsymbol{m}_k$ for $k = 1, ..., K$ denotes the cluster means in the non-linear feature space, $\delta$ is the Kronecker delta function and $c_i$ denotes the cluster index for each data point $i = 1, ..., l$. Aruna argues that the iterative clustering of the entire dataset can be achieved using the Gram matrix $\mathbf{G}$ without explicitly computing the non-linear features $\boldsymbol{\phi}(\mathbf{x})$. How can she achieve this. (**Points 15**)

5. **Convolutional Networks** - A CNN realizes a convolution operation of input image $\mathbf{X}$ of size $(U, V)$ with a set of weights (filters) $\mathbf{W}^k$ for $k = 1, .., K$ where $K$ denotes the number of filters in a CNN layer. The convolution operation is given by,

$$\begin{aligned} \mathbf{Y}^k &= \mathbf{X} * \mathbf{W}^k \\ \mathbf{Y}^k(p,q) &= \sum_{i=0}^{S-1} \sum_{j=0}^{T-1} \mathbf{X}(p+i, q+j) \mathbf{W}^k(i,j) \end{aligned}$$

where $(S, T)$ is the size of the filter $\mathbf{W}^k$, $p$ ranges from $0, 1, ..., U - S$ and $q$ ranges from $0, 1, ..., V - T$. Note that the output image $\mathbf{Y}^k$ is of size $(U - S + 1, V - T + 1)$. Let $J$ denote the cost function used in CNN training. Assume that the partial derivative w.r.t. to output of filter has been computed as $\frac{\partial J}{\partial \mathbf{Y}^k}$. Prove the following gradient update rule for filter learning

$$\frac{\partial J}{\partial \mathbf{W}^k} = \mathbf{X} * \frac{\partial J}{\partial \mathbf{Y}^k}$$

(**Points 10**)

6. **Gambling** - Dhruv visits a casino to play the die game which is the following. He bets Rs $z$. He always rolls a fair die. Megha, who is the casino host, picks another die from a pack of two dice. This pack contains

   - a fair die $P(i) = 1/6$ where $i = 1, .., 6$ and,
   - a loaded die $P(i) = 1/15$ for $i = 1, .., 5$ and $P(i = 6) = 2/3$.

   Each time Megha can keep her current die with a probability of 0.9 or switch to a different die with a probabilty of 0.1. She also has a equal probability of picking either the fair or loaded die initially. After Dhruv rolls his die, Megha rolls her die. The house wins if Megha's die value is higher or equal to die value of Dhruv. On the other hand, if Dhruv's die has a higher value than Megha's, he wins and gets back Rs $2z$.
   In his game, Dhruv rolls his die twice and gets values 2 and 3. With this game of two rolls, what is the probability that he bets Rs 100 and wins back Rs 400 after the two rolls.

   *No need to use calculators, showing the steps to calculate is sufficient*

   (**Points 10**)