# ECE/CS 559 - Fall 2020 - Midterm
# Due: 10/14/2020, 11:00pm Chicago time.

**WARNING: Any plagiarism regarding any part of the exam will result in a full negative grade (-100) for the entire exam. Write all computer codes yourself and refrain from using online resources as well as codes from your peers. As usual, a copy of all answers and codes should be posted to Gradescope.**

**Q1.** Write a competitive learning based image compressor.

1. Download barbara.png from the "General Resources" section of the course Piazza website. Parse it to a variable. The result should be a $512 \times 512$ matrix, say $A$, where each entry of the matrix $A$ ranges from 0 to 255, where 0 represents a black pixel and 255 represents a white pixel. Correct parsing is important as otherwise your error calculations will be wrong.

2. (5 pts) **How many bits are needed to represent the (uncompressed) barbara.png image as described above? Justify your answer.**

3. Consider representing $A$ as

$$A = \begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1n} \\ A_{21} & A_{22} & \cdots & A_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ A_{n1} & A_{n2} & \cdots & A_{nn} \end{bmatrix} \tag{1}$$

   where each $A_{ij}$ is a $d \times d$ matrix, $d$ divides 512, and $n = \frac{512}{d}$. The $n^2$ submatrices $A_{ij}$, $i, j \in \{1, \ldots, n\}$ of dimensions $d \times d$ each will be your training set.

4. Initialize neuron weights $X_1, \ldots, X_K$ represented as $d \times d$ matrices arbitrarily (e.g. each entry can be chosen uniformly at random on $[0, 255]$). Do not be confused that the neuron weights are matrices in this case; if you like, you can view them as $d^2$-dimensional vectors as well.

5. Train the weights using competitive learning. Training will be slightly different than what we learned in class. In particular, given that the input to the neural network is $A_{ij}$ for some $i, j \in \{1, \ldots, n\}$, the winning neuron should be chosen as the neuron whose weights are closest to $A_{ij}$:

$$k^\star = \arg\min_k \|A_{ij} - X_k\|^2, \tag{2}$$

where $\|\cdot\|^2$ represents the sum of squares of entries of a matrix. As usual, only the weights of the winning neuron should be updated as

$$X_{k^\star} \leftarrow (1-\eta)X_{k^\star} + \eta A_{ij}, \tag{3}$$

where $\eta$ is the learning rate.

6. (10 pts) **For the competitive learning algorithm in class, the winning neuron is the one that maximizes the inner product. Here, the winner neuron is the one that minimizes the Euclidean distance, while the weight update method remains the same. Do the neuron weights converge under this modified competitive learning algorithm? Mathematically prove your claim.**

7. Now, suppose we go through many epochs of training to come up with some weights $X_1, \ldots, X_K$. We now begin the testing phase. First, find the weights of the winning neuron for each subblock in the same manner as before

$$k^\star(i,j) = \arg\min_k \|A_{ij} - X_k\|^2,\ i,j \in \{1,\ldots,n\}. \tag{4}$$

We do not update any weights during this testing phase. We consider the image

$$A' = \begin{bmatrix} X_{k^\star(1,1)} & X_{k^\star(1,2)} & \cdots & X_{k^\star(1,n)} \\ X_{k^\star(2,1)} & X_{k^\star(2,2)} & \cdots & X_{k^\star(2,n)} \\ \vdots & \vdots & \ddots & \vdots \\ X_{k^\star(n,1)} & X_{k^\star(n,2)} & \cdots & X_{k^\star(n,n)} \end{bmatrix} \tag{5}$$

We can also calculate the root mean squared error (RMS):

$$\frac{1}{512}\|A' - A\|. \tag{6}$$

8. (5 pts) **How many bits are needed to represent (5)? Justify your answer.**

9. (35 pts) **Upload a SINGLE .py or .m file to the code repository named "MQ1-IdNumber-LastName.ext," where IdNumber is your UIC NetID, LastName is your last name, and ext is the appropriate extension. IPYNB, MTX, are not allowed. This script should generate (5) for the case $d = 4$ and $K = 8$ and display it as a grayscale image (just like the original image). Your code should also print the resulting RMS after each epoch (i.e., after each epoch of training, which is desribed in Item 5, you should calculate the resulting RMS using the method described in Item 7). The code should not print anything else! Your code should execute without any errors or need modification and should provide all output within 3 minutes at most. Assume**

**barbara.png is in the same directory as the script. Your code should be general enough as I will modify the variables $d$ and $K$ and run it for different test values.**

**In your written report, print the final images (5) and the corresponding final RMS values for $d = 4$ and $K \in \{2, 8, 32\}$. Also include the epoch number vs RMS graphs for these cases.**

Hint: You will observe that some neurons never become winners due to bad initialization after certain epochs. You should reinitialize the weights of such neurons, or otherwise, you will not achieve a good RMS performance. A good strategy is to simply set the weights of such neurons to be equal to one of the training inputs.

10. (10 pts) **Explain how and why competitive learning achieves compression while remaining faithful to the original image?**

**Q2.** We will write a program on gradient descent and linear least squares fits.

1. Let $x_i = i$, $i = 1, \ldots, 50$. Let $y_i = i + u_i$, $i = 1, \ldots, 50$, where each $u_i$ should be chosen to be an arbitrary real number between $-1$ and 1.

2. (5 pts) **Find a mathematical expression for the linear least squares fit to $(x_i, y_i)$, $i = 1, \ldots, 50$.** Note that the linear least squares fit is the line $y = w_0 + w_1 x$, where $w_0$ and $w_1$ should be chosen to minimize $\sum_{i=1}^{50} (y_i - (w_0 + w_1 x_i))^2$.

3. (5 pts) **Upload a <u>SINGLE</u> .py or .m file to the code repository named "MQ2-IdNumber-LastName.ext." This script should plot the points $(x_i, y_i)$, $i = 1, \ldots, 50$ together with their linear least squares fit found in Item 2. Label this fit as "Analytical."**

4. (10 pts) **Find a mathematical expression for the gradient of $\sum_{i=1}^{50} (y_i - (w_0 + w_1 x_i))^2$ (derivatives with respect to $w_0$ and $w_1$).**

5. (10 pts) **What your script in Item 3 should also do is to (re)find the linear least squares fit using the gradient descent algorithm. Put the corresponding fit on the same plot as in Item 3, and label it as "Gradient Descent."**

6. (5 pts) **Compare and contrast the two fits: If they are the same explain why. If they are different, explain why.**