Homework #6
Due Thursday, November 16 at 4 p.m.

Please drop the assignment in the HOMEWORK In box in the EE 278 drawer of the class file cabinet on the 2nd floor of the Packard Building.
Please work on problems 1-2 with your group, and submit problems 3-6 individually.

1. Neural 4-ary QAM. Quadrature Amplitude Modulation (QAM) is a modulation scheme that conveys two analog message signals, or two digital bit streams, by changing the amplitudes of two carrier waves. QAM is used extensively as a modulation scheme for digital telecommunication systems, such as in 802.11 Wi-Fi standards. In this problem, we consider a simplified case where we want to transmit a signal $H \in \{0, 1, 2, 3\}$, $p_H(i) = \frac{1}{4}$, $i = 0, 1, 2, 3$. The communication scheme is illustrated in figure 1.

![Figure 1: Communication scheme.](image)

When $H = i$, the transmitter transmits $s_i \in \mathbb{R}^2$. The values of $s_i$’s are shown in figure 2. The noise is $Z \sim \mathcal{N}(0, \sigma^2 I_{2 \times 2})$ and the observation, when $H = i$, is $Y = s_i + Z$.

a. Find the maximum likelihood decision rule and calculate its probability of error. You can express your answer in terms of the $Q$ function.

We want to find an estimate $\hat{H}$ using a neural network and compare its performance with the maximum likelihood decoder. The neural network takes input $Y$ and outputs a probability vector $p \in \mathbb{R}^4$, where $p_i$ represents the probability that $\hat{H} = i$. This is similar to the MNIST classification problem. For more information, please review the TensorFlow tutorial slides and code on the course website. The starter code for this problem is given in neural_qam.py.
b. Write the function \texttt{generate
data} to generate the training data. For \texttt{p}, use a one-hot representation where

\[ p_i = \begin{cases} 
1 & \text{if } H = i \\
0 & \text{otherwise.} 
\end{cases} \]

Assume \( d = 2 \) and \( \sigma^2 = 1 \).

c. Train a neural network decoder with 10,000 training samples. Use the softmax function at the output layer and the cross-entropy loss as the loss function. Tune the parameters, including but not limited to depth, number of neurons, initialization values, etc. What is the best set of parameters that you get?

d. Compare the empirical probability of error of the neural network with the probability of error of the maximum likelihood decoder computed in part a. on a separate test dataset of 1000 samples. How well does your neural network perform?

e. Now assume \( d = 3, \sigma^2 = 1 \). Train the neural network with 10,000 training samples and compare its empirical probability of error with the probability of error of the maximum likelihood decoder on a separate test dataset of 1000 samples.

2. \textit{Vector neural estimator}. Consider a channel with vector inputs \( X \) and vector outputs \( Y \). The channel is given in the illustration below in figure 3.

The input of the channel is a 6-dimensional vector \( X \in \mathbb{R}^6 \), and the output is \( Y \in \mathbb{R}^5 \). Each node performs an addition operation of the incoming nodes plus an iid noise. For example, \( N_{11} = X_1 + X_2 + Z_{11} \) and \( Y_1 = N_{21} + N_{32} + Z_{31} \). \( X \) and \( Z_{ij} \)'s are all independent. In this exercise you will implement a neural network estimator for this vector channel.

a. Write a function to generate training data. Assume \( X \sim \mathcal{N}(0, I_{6 \times 6}) \) and i.i.d. noise \( Z_{ij} \sim \text{Exp}(0.1) \).
b. Train a neural network estimator which takes input $Y$ and outputs an estimated $\hat{X}$. Tune the parameters, including but not limited to depth, number of neurons, non-linearity, initialization and so on. What’s the best set of parameters you get?

c. Calculate the covariance matrix for random vector $(X_1, \cdots, X_6, Y_1, \cdots, Y_5)$. Find the optimal linear estimator.

d. Compare the neural network with the linear estimator. Report their root mean-square errors (RMSE).

3. Convergence to a random variable. Consider a coin with random bias $P \sim F_P(p)$. Flip the coin $n$ times independently to generate $X_1, X_2, \ldots, X_n$, where $X_i = 1$ if the $i$-th outcome is heads and $X_i = 0$ otherwise. Let $S_n = \frac{1}{n} \sum_{i=1}^{n} X_i$ be the sample average. Show that $S_n$ converges to $P$ in mean square.

4. Convergence examples. Consider the following sequences of random variables, where $\omega$ is uniformly distributed in the interval $[0, 1]$.

$$X_n(\omega) = \begin{cases} \frac{1}{n} & \omega < \frac{1}{n} \\ 0 & \text{otherwise} \end{cases} \quad Y_n(\omega) = \begin{cases} 2^n & \omega < \frac{1}{n} \\ 0 & \text{otherwise} \end{cases} \quad Z_n(\omega) = \begin{cases} 1 & \omega \in S_n \\ 0 & \text{otherwise} \end{cases}$$

where $\{S_n\}$ denotes the following sequence of intervals:

$$\{S_n\} = \{[0, 1], [0, 1/2), (1/2, 1], [0, 1/3), (1/3, 2/3), (2/3, 1], \ldots \}.$$

Which of these sequences converges to zero (a) with probability one, (b) in mean
square, and/or (c) in probability? Justify your answers.

5. *Convergence in probability implies convergence in distribution.* Prove that convergence in probability implies convergence in distribution following these steps:
   a. Show that for any random variable $A$, any real number $a$ and any $\epsilon > 0$,
      \[ P ( A \leq a - \epsilon ) - P ( |A_n - A| > \epsilon ) \leq P ( A \leq a + \epsilon ) + P ( |A_n - A| > \epsilon ) \]
   b. Assume that $A_n \rightarrow A$ in probability. Use the expression derived in (a) to prove that $A_n \rightarrow A$ in distribution. Hint: Remember that $A_n \rightarrow A$ in distribution if $\lim_{n \rightarrow \infty} F_{A_n}(a) = F_A(a)$ for points $a$ at which $F_A$ is continuous, which implies $\lim_{\epsilon \rightarrow 0} F_A(a \pm \epsilon) = F_A(a)$.

6. *Convergence experiments.* The purpose of this problem is to demonstrate different types of convergence and laws of large numbers.
   a. Use MATLAB to generate 200 samples $X_1, \ldots, X_{200}$ of i.i.d. zero mean, unit variance Gaussian random variables. Plot the sample average sequence $S_n = \frac{1}{n} \sum_{i=1}^{n} X_i$ as a function of $n$. Hint: use `randn` and `cumsum`.
   b. Generate 5000 sequences each consisting of 200 samples of i.i.d. zero mean, unit variance Gaussian random variables:
      \[ X_{i,1}, X_{i,2}, \ldots, X_{i,200}, \quad i = 1, 2, \ldots, 5000 \]
      Compute the sample average sequences $S_{i,1}, S_{i,2}, \ldots, S_{i,200}$ for $i = 1, \ldots, 5000$. You can generate a $5000 \times 200$ matrix of i.i.d. Gaussians using the `randn` command.
   c. *Strong law of large numbers.* To explore convergence with probability 1, let $N_m$ be the number of $S_{i,n}$ sequences with $|S_{i,n} - E(X)| > 0.1$ for some $n > m$. Use $E_m = N_m/5000$ as an estimate of the probability $P_m = P\{|S_n - E(X)| > 0.1\}$ for some $n > m$. Plot $E_m$ vs. $m$ for $1 \leq m \leq 200$.
   d. *Mean square convergence.* To explore convergence in mean square, compute
      \[ M_n = \frac{1}{5000} \sum_{i=1}^{5000} (S_{i,n} - E(X))^2, \]
      which are estimates of the mean square. Plot $M_n$ vs. $n$ for $1 \leq n \leq 200$.
   e. *Weak law of large numbers.* To explore convergence in probability, let $N_n$ be the number of $S_{i,n}$ sequences for which $|S_{i,n} - E(X)| > 0.1$. Use $E_n = N_n/5000$ as an estimate of the probability $P_n = P\{|S_n - E(X)| > 0.1\}$. Plot $E_n$ and $P_n$ vs. $n$ for $1 \leq n \leq 200$. 
