Assignment 2 (12 Marks)

1. (4 Marks) Gradient descent is a first-order iterative optimization algorithm for finding a local minimum of a differentiable function. To find a local minimum of a function using gradient descent, we take steps proportional to the negative of the gradient (or approximate gradient) of the function at the current point. Implement algorithm keeping the value of \( \epsilon = 0.01 \) and \( f(\theta) = \theta^2 + 4 \). Initiate the value of \( \theta \) randomly such that \( \theta \in [100, 101] \). Plot the graph of number of iterations vs function value keeping \( \eta = 1, 0.1, 0.01, 0.001 \). Report the graph and the observations.

Algorithm 1 Gradient Descent Algorithm

Result: Local minima of the function \( f(\theta) \)

Initialization: \( f(\theta), \epsilon, \eta \)

while \( |f'(\theta)| > \epsilon \) do
   \( \theta \leftarrow \theta - \eta \nabla f(\theta) \)
end

2. (2 Marks) If \( E_i \in \mathcal{F}, i \geq 1 \), then show that

\[
P(\bigcup_i E_i) = \sum_i P(E_i) - \sum_{i<j} P(E_iE_j) + \sum_{i<j<k} P(E_iE_jE_k) - \sum_{i<j<k<l} P(E_iE_jE_kE_l) + \ldots + (-1)^{n+1} P(E_1E_2\ldots E_n)
\]

3. (1 Marks) Construct a hypothesis class \( \mathcal{H} \) and a sequence of examples on which Consistent algorithm will make \( |\mathcal{H}| - 1 \) mistakes.

4. (5 Marks) Consider the problem of prediction with expert advice with \( d = 10 \). Assume that the losses assigned to each expert are generated according to independent Bernoulli distributions. The adversary/environment generates loss for experts 1 to 8 according to \( Ber(0.5) \) in each round. For the 9th expert, loss is generated according to \( Ber(0.5\Delta) \) in each round. The losses for the 10th expert are generated according to different Bernoulli random variable in each round - for the first \( T/2 \) rounds, they are generated according to \( Ber(0.5 + \Delta) \) and the remaining \( T/2 \) rounds they are generated according to Bernoulli random variable \( Ber(0.52\Delta) \). \( \Delta = 0.1 \) and \( T = 105 \). Generate (pseudo) regret values for different learning rates \( \eta \) for Weighted Majority algorithm. The averages should be taken over at least 20 sample paths (more is better). Display 95% confidence intervals for each plot. Vary \( c \) in the interval \([0.1, 2.1]\) in steps of size 0.2 to get different learning rates. Implement Weighted Majority algorithm with \( \eta = c\sqrt{2\log(d)/T} \). Report the plot.