CSE4088 Introduction to Machine Learning

Homework 2

- **Generalization Error**

1. The modified Hoeffding Inequality provides a way to characterize the generalization error with a probabilistic bound

$$\mathbb{P}\left[|E_{in}(g) - E_{out}(g)| > \epsilon\right] \leq 2Me^{-2\epsilon^2 N}$$

for any $\epsilon > 0$. If we set $\epsilon = 0.05$ and want the probability bound $2Me^{-2\epsilon^2 N}$ to be at most 0.03, what is the least number of examples N (among the given choices) needed for the case $M = 1$?

[a] 500
[b] 1000
[c] 1500
[d] 2000
[e] More examples are needed.

2. Repeat for the case $M = 10$.

[a] 500
[b] 1000
[c] 1500
[d] 2000
[e] More examples are needed.

3. Repeat for the case $M = 100$.

[a] 500
[b] 1000
[c] 1500
[d] 2000
[e] More examples are needed.

- **The Perceptron Learning Algorithm**

In this problem, you will create your own target function f and data set $\mathcal{D}$ to see how the Perceptron Learning Algorithm works. Take $d = 2$ so you can visualize the problem, and assume $\mathcal{X} = [-1, 1] \times [-1, 1]$ with uniform probability of picking each $\mathbf{x} \in \mathcal{X}$.

In each run, choose a random line in the plane as your target function f (do this by taking two random, uniformly distributed points in $[-1, 1] \times [-1, 1]$ and taking the line passing through them), where one side of the line maps to $+1$ and the other maps to -1. Choose the inputs $\mathbf{x}_n$ of the data set as random points (uniformly in $\mathcal{X}$), and evaluate the target function on each $\mathbf{x}_n$ to get the corresponding output y_n.

Now, in each run, use the Perceptron Learning Algorithm to find g. Start the PLA with the weight vector $\mathbf{w}$ being all zeros (consider $\text{sign}(0) = 0$, so all points are initially misclassified), and at each iteration have the algorithm choose a point randomly from the set of misclassified points. We are interested in two quantities: the number of iterations that PLA takes to converge to g, and the disagreement between f and g which is $\mathbb{P}[f(\mathbf{x}) \neq g(\mathbf{x})]$ (the probability that f and g will disagree on their classification of a random point). You can either calculate this probability exactly, or approximate it by generating a sufficiently large, separate set of points to estimate it.

In order to get a reliable estimate for these two quantities, you should repeat the experiment for 1000 runs (each run as specified above) and take the average over these runs.

4. Take $N = 10$. How many iterations does it take on average for the PLA to converge for $N = 10$ training points? Pick the value closest to your results (again, 'closest' means: |your answer − given option| is closest to 0).

   [a] 1
   [b] 15
   [c] 300
   [d] 5000
   [e] 10000

5. Which of the following is closest to $\mathbb{P}[f(\mathbf{x}) \neq g(\mathbf{x})]$ for $N = 10$?

   [a] 0.001
   [b] 0.01
   [c] 0.1
   [d] 0.5

   [e] 0.8

6. Now, try $N = 100$. How many iterations does it take on average for the PLA to converge for $N = 100$ training points? Pick the value closest to your results.

[a] 50
[b] 100
[c] 500
[d] 1000
[e] 5000

7. Which of the following is closest to $\mathbb{P}[f(\mathbf{x}) \neq g(\mathbf{x})]$ for $N = 100$?

[a] 0.001
[b] 0.01
[c] 0.1
[d] 0.5
[e] 0.8

● **Linear Regression**

In these problems, we will explore how Linear Regression for classification works. As with the Perceptron Learning Algorithm , you will create your own target function f and data set $\mathcal{D}$. Take $d = 2$ so you can visualize the problem, and assume $\mathcal{X} = [-1, 1] \times [-1, 1]$ with uniform probability of picking each $\mathbf{x} \in \mathcal{X}$. In each run, choose a random line in the plane as your target function f (do this by taking two random, uniformly distributed points in $[-1, 1] \times [-1, 1]$ and taking the line passing through them), where one side of the line maps to $+1$ and the other maps to -1. Choose the inputs $\mathbf{x}_n$ of the data set as random points (uniformly in $\mathcal{X}$), and evaluate the target function on each $\mathbf{x}_n$ to get the corresponding output y_n.

8. Take $N = 100$. Use Linear Regression to find g and evaluate E_{in}, the fraction of in-sample points which got classified incorrectly. Repeat the experiment 1000 times and take the average (keep the g's as they will be used again in Problem 9). Which of the following values is closest to the average E_{in}? (*Closest* is the option that makes the expression |your answer − given option| closest to 0. Use this definition of *closest* here and throughout.)

[a] 0
[b] 0.001
[c] 0.01
[d] 0.1
[e] 0.5

9. Now, generate 1000 fresh points and use them to estimate the out-of-sample error E_{out} of g that you got in Problem 8 number of misclassified out-of-sample points / total number of out-of-sample points). Again, run the experiment 1000 times and take the average. Which value is closest to the average E_{out}?

   [a] 0

   [b] 0.001

   [c] 0.01

   [d] 0.1

   [e] 0.5

10. Now, take $N = 10$. After finding the weights using Linear Regression, use them as a vector of initial weights for the Perceptron Learning Algorithm. Run PLA until it converges to a final vector of weights that completely separates all the in-sample points. Among the choices below, what is the closest value to the average number of iterations (over 1000 runs) that PLA takes to converge? (When implementing PLA, have the algorithm choose a point randomly from the set of misclassified points at each iteration)

   [a] 1

   [b] 15

   [c] 300

   [d] 5000

   [e] 10000

• Nonlinear Transformation

In these problems, we again apply Linear Regression for classification. Consider the target function:

$$f(x_1, x_2) = \text{sign}(x_1^2 + x_2^2 - 0.6)$$

Generate a training set of $N = 1000$ points on $\mathcal{X} = [-1, 1] \times [-1, 1]$ with a uniform probability of picking each $\mathbf{x} \in \mathcal{X}$. Generate simulated noise by flipping the sign of the output in a randomly selected 10% subset of the generated training set.

11. Carry out Linear Regression without transformation, i.e., with feature vector:

$$(1, x_1, x_2),$$

to find the weight $\mathbf{w}$. What is the closest value to the classification in-sample error E_{in}? (Run the experiment 1000 times and take the average E_{in} to reduce variation in your results.)

[a] 0
[b] 0.1
[c] 0.3
[d] 0.5
[e] 0.8

12. Now, transform the $N = 1000$ training data into the following nonlinear feature vector:

$$(1, x_1, x_2, x_1x_2, x_1^2, x_2^2)$$

Find the vector $\tilde{\mathbf{w}}$ that corresponds to the solution of Linear Regression. Which of the following hypotheses is closest to the one you find? Closest here means agrees the most with your hypothesis (has the highest probability of agreeing on a randomly selected point). Average over a few runs to make sure your answer is stable.

[a] $g(x_1, x_2) = \text{sign}(-1 - 0.05x_1 + 0.08x_2 + 0.13x_1x_2 + 1.5x_1^2 + 1.5x_2^2)$
[b] $g(x_1, x_2) = \text{sign}(-1 - 0.05x_1 + 0.08x_2 + 0.13x_1x_2 + 1.5x_1^2 + 15x_2^2)$
[c] $g(x_1, x_2) = \text{sign}(-1 - 0.05x_1 + 0.08x_2 + 0.13x_1x_2 + 15x_1^2 + 1.5x_2^2)$
[d] $g(x_1, x_2) = \text{sign}(-1 - 1.5x_1 + 0.08x_2 + 0.13x_1x_2 + 0.05x_1^2 + 0.05x_2^2)$
[e] $g(x_1, x_2) = \text{sign}(-1 - 0.05x_1 + 0.08x_2 + 1.5x_1x_2 + 0.15x_1^2 + 0.15x_2^2)$

13. What is the closest value to the classification out-of-sample error E_{out} of your hypothesis from Problem 12 Estimate it by generating a new set of 1000 points and adding noise, as before. Average over 1000 runs to reduce the variation in your results.)

[a] 0
[b] 0.1
[c] 0.3
[d] 0.5
[e] 0.8

You need to show all your work step by step to receive credit. **You may be asked to make a demo of your code.** The answers are as follows for you to check your solutions: 1(b), 2(c), 3(d), 4(b), 5(c), 6(b), 7(b), 8(c), 9(c), 10(a), 11(d), 12(a), 13(b)

Note: Questions have been taken from "Learning From Data, by Yaser Abu-Mostafa".

Evaluation: Each question is worth 10 points. The overall score will be scaled to 100.

Submission Instructions:

1. Each student should submit his/her own homework. You can discuss the questions with your friends, but you must write your own solutions and code. Group work is not allowed. You can not exchange any written material, code or pseudocode. This also includes material found on the web.

2. Write a detailed report using Word/Latex and convert it to a pdf file. The report must include explanations about each part in each question. You should solve some of the questions by pencil and paper. Then, you can scan your solution and add it to your report as an image. (you can use free Mobile applications such as Adobe Scan). As an alternative, you can also print and hand in your homework during class hours to the instructor.

3. Explain how your scripts and functions work, i.e., which parts of your functions/scripts accomplish which task and how it is accomplished. Include the outputs of your functions and figures to your report. Each figure should have a caption and should be explained in the text.

4. You can use MATLAB or Python for programming assignments. Don't forget to put detailed comments into your functions/scripts to explain what your code is doing. Also indicate the inputs and outputs in the comment section.

5. Combine your report and codes into a single zip file. Plots and figures should go into the report.

6. Name your zip file as "CSE4088_name_surname_hw_no.zip". For example, a student whose name is Ayşe Çalışkan will name her file as: "CSE4088_ayse_caliskan_hw1.zip" for the first homework. Also, write your name, surname and student number as comments at the beginning of your codes.

7. Submit your homework via the class web page at classroom.google.com before the deadline. As an alternative, you can also print and hand in your homework during class hours to the instructor (you should still submit your code via the class web page).

8. Late submissions will loose 10 points for each day after the deadline.