CS 4780/5780: Machine Learning for Intelligent Systems **(Due: 09/24/19)**

Assignment #2: Hypothesis Testing, Linear Classifiers, Perceptrons

Instructor: Nika Haghtalab, Thorsten Joachims *Name:* Student name(s), *Netid:* NetId(s)

Course Policy: Read all the instructions below carefully before you start working on the assignment, and before you make a submission.

- Please include your name and NetIDs on the first page. We recommend typesetting your submission in LaTeX, and an Overleaf template is linked here.
- Assignments are due by noon on the due date in PDF form on Gradescope.
- Late assignments can be submitted on Gradescope up to Sunday, Sept 29. This is also when the solutions will be released.
- You can do this assignment in groups of 2-3. Please submit no more than one submission per group. Collaboration across groups is not permitted.
- All sources of material outside the course must be cited. The University Academic Code of Conduct will be strictly enforced.

Submission Instructions: All group members must be added to the Gradescope submission. If you're the one submitting, add your group members on Gradescope. Otherwise, make sure you are added to the submission. Put your names on the PDF, this helps us track the groups in case there are errors on Gradescope. See this Piazza post for more information.

Problem 1: Hypothesis Testing

(16 + 16 = 32 points)

We will evaluate the performance of two hypotheses. The file **hyp_test_pred.txt** contains the predictions of two hypotheses h_A and h_B. The first column is whether the node was cancerous or benign and the second and third columns are the predictions of h_A and h_B, respectively.

(a) Compute the 95% confidence interval for the prediction error for each model (recall theorem based on Hoeffding's Lemma in class). Also, specify the actual error rates on the given data. Show all your work including application of theorem.

(b) Use the exact binomial McNemar test to decide whether with 95% confidence the error rate of h_A is significantly different from h_B. Show each step of the calculation, i.e. the definitions of null and alternative hypotheses, the contingency table, and the test calculations.

Problem 2: Linear Classifiers

(5 + 5 + 5 + 10 + 10 = 35 points)

We will be using linear separators to recreate certain functions, which we will call $f(\cdot)$. Recall that a linear classifier is written as

$$h_{\vec{w},b}(\vec{x}) = \begin{cases} +1, & \text{if } \vec{w} \cdot \vec{x} + b > 0 \\ -1, & \text{otherwise} \end{cases}$$

with parameter vectors $\vec{w}$ and term b. A homogeneous linear classifier has $b = 0$. For the following questions, construct the linear separator by choosing $\vec{w}$ and b.

(a) First create a linear separator which recreates the logical **NOT** function. The instance space is $X = \{+1, -1\}$. Here $(+1)$ corresponds to TRUE and (-1) corresponds to FALSE, for both the instance space and output of the linear classifier. The function f_{NOT} which we want to recreate is:

$$f_{NOT}(+1) = -1$$
$$f_{NOT}(-1) = +1$$

Give the parameters w, b which create a linear classifier which recreates the logical NOT function f_{NOT}. Note that w is a number, not a vector.

(b) Consider the instance space $X = \mathbb{R}^d$ and let $\overrightarrow{x} = (x_1, \ldots, x_d)$ be a vector of length d. Consider the function

$$f_M(\overrightarrow{x}) = \begin{cases} +1, & \text{if } \sum_{i=1}^{d} x_i > M \\ -1, & \text{otherwise} \end{cases}$$

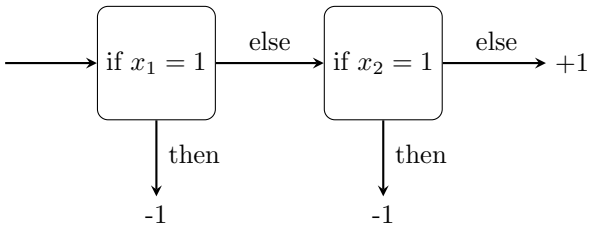
Giver parameters $\overrightarrow{w}, b$. Note here that $\overrightarrow{w}$ is a vector.

(c) Consider the instance space $X = \mathbb{R}^{120}$ and let $\overrightarrow{x} = (x_1, \ldots, x_{120})$. Consider the function

$$f_{120}(\overrightarrow{x}) = \begin{cases} +1, & \text{if } \sum_{i=1}^{10} x_i > \sum_{j=11}^{20} x_j \\ -1, & \text{otherwise} \end{cases}$$

Give the parameters $\overrightarrow{w}$ and b.

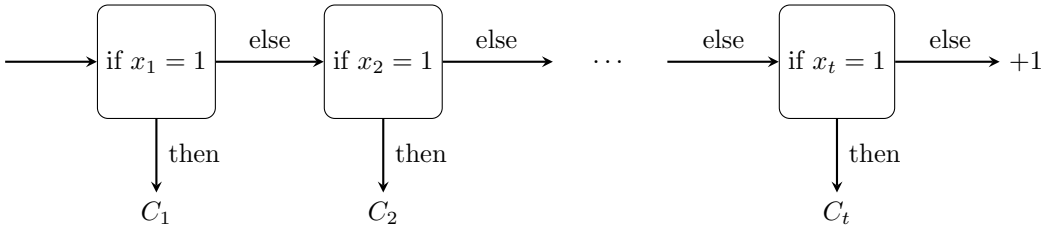
(d) Consider a decision list with two segments as shown.



Create a linear classifier which recreates this 2-segment decision list. Here x_1 refers to the first element and x_2 the second element of our data vector. The instance space will be $X = \{0, 1\}^2$.

Give the parameters $\overrightarrow{w}_{Cond2}, b_{Cond2}$ which create a linear classifier which recreate the 2-segment decison list. For example, we want the linear classifier to classify $x = (1, 1)$ to -1.

(e) Next consider a generalization of the 2-segment decision list. Now there are t segments, and the output of each segment is a variable $C_i = \{-1, +1\}$. However the instance space will be $X = \{0, 1\}^d$, where $d \geq t$.



Give parameters $\overrightarrow{w}_{Cond}, b_{Cond}$ which create a linear classifier which recreates this t-segment decision list using input data with number of features d.

Problem 3: Perceptrons (16 + 17 = 33 points)

(a) Consider the two-point 2D dataset $\{(\vec{x}_i, y_i) | i \in [2]\} = \{((1, 3), +1), ((-1, 4), -1)\}$.

Starting with $w_0 = (0, 0)$, how many updates will you have to perform to w until convergence for a homogeneous batch perceptron (i.e. bias term $b = 0$)? Write down the sequence of each updated w_i ($[w_1, w_2 ... w_n]$) by iterating the data points in the order: $[(1, 3), (-1, 4)]$.

(b) Your friend comes to you, desperate for your Perceptron expertise. Their dataset is massive (with more than 10 trillion training examples), and after hours of training their perceptron (until convergence), their code malfunctioned and did not save the final weight vector.

Thankfully, at every training iteration, the code saved which example was used for the update step. Surprisingly, only five of the 10 trillion+ training examples were ever misclassified. They are listed below, along with the number of times they were used in an update step.

Training Example	Number of Times Used in an Update Step
(0, 0, 0, 0, 4), +1	2
(0, 0, 6, 5, 0), +1	1
(3, 0, 0, 0, 0), -1	1
(0, 9, 3, 6, 0), -1	1
(0, 1, 0, 2, 5), -1	1

What is the final weight vector of this perceptron (i.e. the weight vector that would have been saved if the code had not malfunctioned)? Include a *one-sentence* explanation for how the final weight vector was computed.