CMPUT 466/566
University of Alberta **Fall 2022**

Assignment 1

Due October 4, 2022, 11:59 pm on eClass

This assignment contains a written portion, and a programming portion. Please hand in a pdf for your written answers, and a zip of your code for the programming portion. Your pdf can contain scans of handwritten answers, so long as they are legible.

Questions

1) [Perceptron Algorithm] (25 marks)

Consider training a two-class classifier for a training dataset $\mathcal{D} = \{(x_n, y_n)|n = 1, \cdots, N\}$ with input vector x_n and label $y_n \in \{-1, +1\}$. The perceptron algorithm with weight w makes predictions as

$$h(x, w) =: \begin{cases} +1, \text{ if } w^\top x \geq 0, \\ -1, \text{ if } w^\top x < 0. \end{cases}$$

a) **(5 marks)** Write the perceptron update rule of the weight w_t at step t using a single data point (x_n, y_n). (i.e. $w_{t+1} = ...$) Use a stepsize of 1 for the update rule. How does this update influence the model's predictions for this data point (x_n, y_n) in the next step?

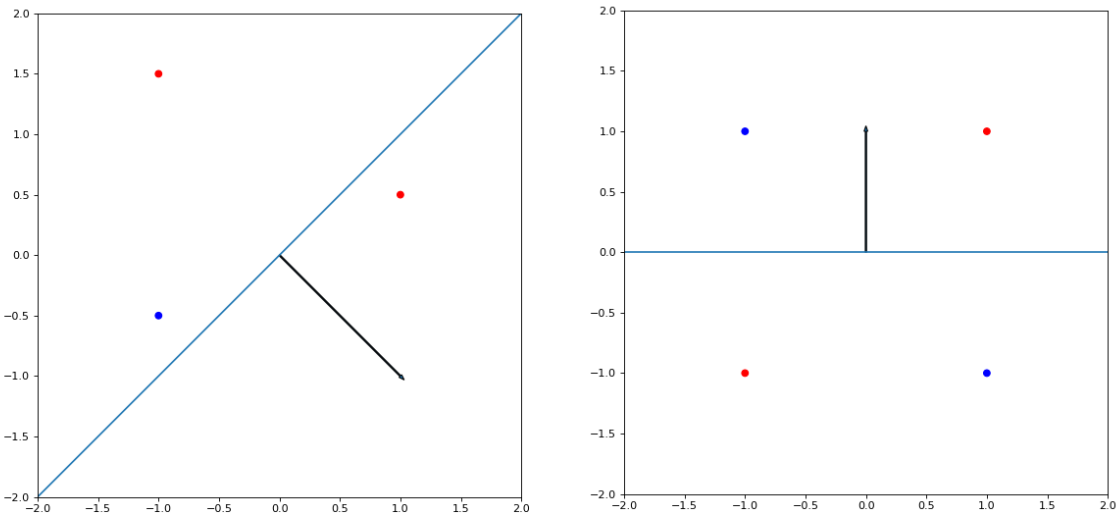

Figure 1: Question **b**) Dataset Plotted on the Left and Question **d**) on the Right. Class 1 data with label +1 is shown in red and class 2 with label −1 in blue.

b) **(10 marks)** Now consider the dataset={([-1,1.5],1),([1,0.5],1),([-1,-0.5],-1)}, and the weight vector is initialized to $w_0 = [1, -1]^T$. On the left of Figure 1 above, class 1 data points (label +1) are shown as red dots, and class 2 data points (label −1) are blue dots. The separation plane is defined as $h(x, w) = w^\top x = 0$, plotted as the blue line in Figure 1. Note that the weight vector

is always perpendicular to the plane shown as the black vector, since given any two points x_A and x_B lying on the plane, $w^\top(x_A - x_B) = 0$.

i. Plot out the update vector of the weight vector using point $([-1, 1.5], 1)$ on the left. [Plot the update vector starting from the current weight initialization.]

ii. Plot out the new weight vector and the separation plane in a new figure.

iii. This time, compute the update rule using any data point. What do you find?

c) **(3 marks)** The perceptron algorithm will stop when it arrives at any weight vector that properly separates training data. Are all possible weight vector solutions (that separate the training data) equally likely to generalize to a larger dataset? Please explain your answer.

d) **(7 marks)** Now consider the dataset={([-1,-1],1),([1,1],1),([1,-1],-1),([-1,1],-1)}, and the weight is initialized as $w_0 = [0, 1]^T$.

i. As above, plot out the update direction of the weight using point $([-1, -1], 1)$ on the right of Figure 1. [Plot the update direction starting from the current weight point.]

ii. Then, plot out the new weight vector and the separation plane in a new figure. Again, plot out the update direction of the weight using point $([1, 1], 1)$.

iii. After these updates, are all training data points correctly predicted? If the algorithm uses other data points to update, will it yield a separation line that correctly predicts for all points? Why or why not?

2) [Perceptron Convergence Proof] (20 marks)

The following information is provided for use in your proof
The figure below provides a brief visual representation of a perceptron where input is $x \in \mathbb{R}^d$ with a d-dimensional parameter vector $w \in \mathbb{R}^d$. The output generated is $h(x, w) = \text{sign}(w^\top x)$, where for $\alpha \in \mathbb{R}$,

$$\text{sign}(\alpha) = \begin{cases} 1 & \text{if } \alpha \geq 0, \\ 0 & \text{if } \alpha < 0 \end{cases}.$$

Consider the following assumptions:

(a) Linear separability - There exists $w^* \in \mathbb{R}^d$ such that $\|w^*\| = 1$ and for $\gamma > 0$, for all $i \in \{1, 2, ..., n\}$,

$$y^i(w^{*\top} x^i) > \gamma. \tag{1}$$

(b) Bounded coordinates - There exists $R \in \mathbb{R}$ such that for all $i \in \{1, 2, ..., n\}$,

$$\|x^i\| \leq R. \tag{2}$$

Hint: it may help to visualize the algorithm by plotting simple data points in a two-dimensional plane. The perceptron algorithm reflects the fact that the magnitude of the weight vector increases with every update such that the weight vector is normal to the hyperplane.

Our main goal will be to bound the norm of w^{k+1} in terms of k for proving convergence Prove the following to prove convergence **(5 marks each)**:

a) Find a lower bound for the inner product of the weights at $(k+1)$-th iteration i.e. w^{k+1} and w^* using the perceptron algorithm update. *Hint*: Use induction, and assume $w^1 = 0$

b) Use Cauchy Schwartz inequality on the result of 1 to find a lower bound of w^{k+1}.

c) Find an upper bound for the inner product of the weights at $(k+1)$-th iteration using the perceptron algorithm update. *Hint*: Assuming bounded coordinates may help here. Also like (a), use induction.

d) Using the results of (a) and (c), prove the convergence and find the maximum number of updates it may take to converge.

3) [Maximum Likelihood Estimation] (10 points)

In a certain board game, players collect resources once every turn equal to the number of heads they get after flipping 10 coins. The probably of each coin of landing on heads is unknown, but all 10 coins have the same probability. What is the probability distribution of the number of resources players collect on their turn? Write the probability mass or density function of the distribution and derive the maximum likelihood estimate of the unknown parameter based on a sample of m iid observations.

4) [Optimization: CMPUT 466 only] (10 points)

Imagine you want to estimate a real-valued function $f(\boldsymbol{x}_t) = y_t \in \mathbb{R}$ using stochastic gradient descent, where $\boldsymbol{x}_t \in \mathbb{R}^p$ is a feature vector at time t and y_t is the output of the function. Although you don't know the function f, you suspect that y_t is a linear combination of the features. Thus, you decide to approximate $f(x)$ as the inner product between a parameter vector $\boldsymbol{w}_t \in \mathbb{R}^p$ and the feature vector, i.e., $\hat{f}(\boldsymbol{x}) = \boldsymbol{x}^\top \boldsymbol{w}$. Using the Absolute Error Loss with L2-regularization shown below, derive the first-order stochastic gradient descent update for the parameter vector at time $t+1$:

$$\ell(\boldsymbol{w}) = |\hat{f}(\boldsymbol{x}) - f(\boldsymbol{x})| + \lambda \sum_{i=1}^{p} w_i^2,$$

where λ is a non-negative scalars. Use the following definition in your solution

$$\frac{d}{dx}|x| = \text{sign}(x) = \begin{cases} 1, & \text{if } x > 0 \\ 0, & \text{if } x = 0 \\ -1, & \text{if } x < 0 \end{cases}.$$

5) [Optimization: CMPUT 566 only] (10 points)

Imagine you want to estimate a real-valued function $f(\boldsymbol{x}_t) = y_t \in \mathbb{R}$ using stochastic gradient descent, where $\boldsymbol{x}_t \in \mathbb{R}^p$ is a feature vector at time t and y_t is the output of the function. Although you don't know the function f, you suspect that y_t is linear combination of the features. Thus, you decide to approximate $f(x)$ as the inner product between a parameter vector $\boldsymbol{w}_t \in \mathbb{R}^p$ and the feature vector, i.e., $\hat{f}(\boldsymbol{x}) = \boldsymbol{x}^\top \boldsymbol{w}$. Using the Huber Loss with L2-regularization shown below, derive the first-order stochastic gradient descent update for the parameter vector at time $t+1$:

$$\ell(\boldsymbol{w}) = \begin{cases} \frac{1}{2}(\hat{f}(\boldsymbol{x}) - f(\boldsymbol{x}))^2 + \lambda \sum_{i=1}^{p} w_i^2, & \text{if } |\hat{f}(\boldsymbol{x}) - f(\boldsymbol{x})| \le \delta \\ \delta|\hat{f}(\boldsymbol{x}) - f(\boldsymbol{x})| - \frac{\delta^2}{2} + \lambda \sum_{i=1}^{p} w_i^2, & \text{otherwise} \end{cases}$$

where δ and λ are non-negative scalars. Use the following definition in your solution

$$\frac{d}{dx}|x| = \text{sign}(x) = \begin{cases} 1, & \text{if } x > 0 \\ 0, & \text{if } x = 0 \\ -1, & \text{if } x < 0 \end{cases}.$$

6) [Naive Bayes] (25 points)

Use the resources provided in *a1.zip* to finish this section. The code provided already handles preprocessing (converting discrete features to numeric values), inputting train data and making predictions. You just need to finish the parts indicated in the classifier.

You cannot use any libraries other than *numpy* and *pandas*. You cannot use code completion tools like *GitHub Copilot*.

Remember you can calculate the probability of each class using the following, where α is the same for all classes:

$$P(c|e_1,...,e_n) = \alpha P(e_1|c)...P(e_n|c)P(c)$$

a) **(20 marks)** Finish the implementation of the Naive Bayes Classifier with laplace smoothing found in *a1.zip*

b) **(2 marks)** What accuracy do you get on the *heart_test.csv* dataset predicting the *HeartDisease* value?

c) **(1 mark)** Is there a difference in performance on the *heart_test.csv* dataset if you remove laplace smoothing?

d) **(2 marks)** Why is there this difference/lack of difference?

7) [Decision Trees] (10 points)

Use the following decision tree and data for to answer the questions in this section

comment_count	has_location	caption_length	user
8	FALSE	905	ualberta
177	FALSE	465	ucalgary
2	FALSE	163	ualberta
8	FALSE	320	universityofbc
4	TRUE	152	ualberta
6	TRUE	219	ualberta
11	TRUE	103	ualberta
4	FALSE	530	ucalgary
18	FALSE	256	universityofbc
16	FALSE	222	universityofbc
6	FALSE	397	ualberta

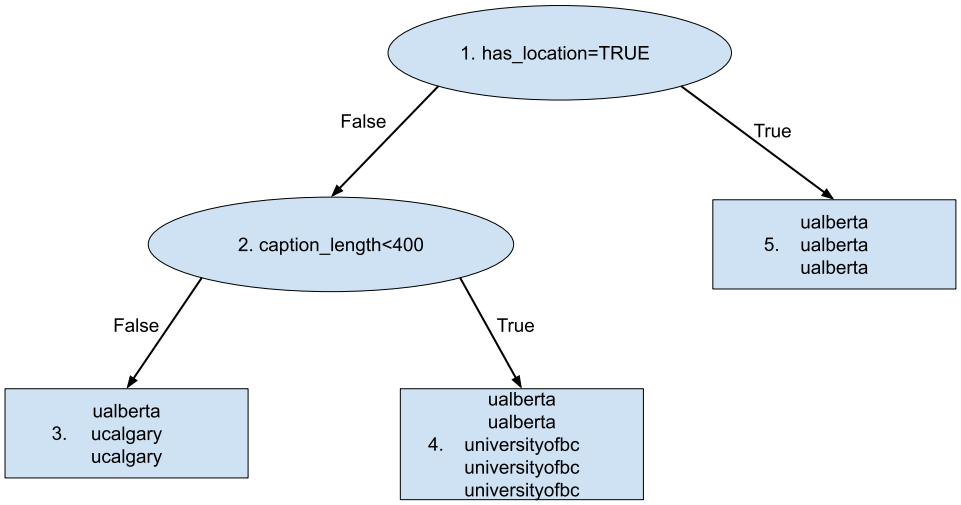

Figure 2: Decision Tree Predicting **user**

(a) **(3 marks)** Compute the entropy of *user* at leaf *Node 3* (the required data is shown in both the decision tree and the table). Show your work.

(b) **(3 marks)** Is *caption_length* or *comment_count* a better feature to make the **next** split on for leaf *Node 4* if an optimal threshold is chosen for each feature based on the data above? Why? (Hint: look at the purity of nodes after splitting on each feature, you may find that one possibility is the clear winner.)

(c) **(4 marks)** Assume we want to extend the tree by adding a new decision in place of leaf *Node 4*. After building the tree, we will pre-prune nodes if the entropy does not decrease by more than 0.1. If we add a new node that makes a decision on *caption_length* < 250 will the new node be pruned? Show your work.