# CS 5350 - Homework 2

Jakob Horvath - u1092049

October 1, 2020

## 1 Warm Up: Linear Classifiers and Boolean Functions

1. Linearly separable:
   $\neg x_1 \vee \neg x_2 \vee x_3 \equiv (1 - x_1) + (1 - x_2) + x_3 \geq 1 \equiv -x_1 - x_2 + x_3 + 1 \geq 0 \therefore w = [-1, -1, 1], b = 1$

2. Linearly separable:
   $(x_1 \vee x_2) \wedge (\neg x_2 \vee \neg x_3) \equiv x_1 + x_2 + (1 - x_2) + (1 - x_3) \geq 2 \equiv x_1 - x_3 \geq 0 \therefore w = [1, 0, -1], b = 0$

3. Linearly separable:
   $(x_1 \wedge x_2) \vee \neg x_3 \equiv x_1 + x_2 + (1 - x_3) \geq 2 \equiv x_1 + x_2 - x_3 - 1 \geq 0 \therefore w = [1, 1, -1], b = -1$

4. Not linearly separable:
   $x_1 \ xor \ x_2 \ xor \ x_3 \implies w_1 \geq 0$ and $w_2 \geq 0$, but $w_1 + w_2 < 0$.

5. Linearly separable:
   $\neg x_1 \wedge x_2 \wedge x_3 \equiv (1 - x_1) + x_2 + x_3 \geq 3 \equiv -x_1 + x_2 + x_3 - 2 \geq 0 \therefore w = [-1, 1, 1], b = -2$

## 2 Mistake Bound Model of Learning

1. (a) $P_k = \sum_{r=1}^{k} C(n, r) = \sum_{r=1}^{k} \frac{n!}{r!(n-r)!}$

   (b) $\log_2(\sum_{r=1}^{k} \frac{n!}{r!(n-r)!})$

2. (a) Total number of functions $= 2^n$

   (b) This Halving algorithm will make at most 1 mistake:
   Proof: Given that we initialize $C$ to be the finite concept class and receive example $x$ to evaluate, only one *indicator function* in the entire set will output $True$. There is a $\frac{1}{n}$ chance that the example we receive is the exact dimensional vector the desired function will accept as $True$. In that case, unless we are working in one dimension, the Halving algorithm will predict False and then throw out all other functions in the set when it knows that the prediction was incorrect. This would leave us with only the one desired function remaining in the set,

and learning would end. Therefore, the Halving algorithm would only make one mistake, possibly seeing every other example for all other functions before arriving at the one example which agrees with the desired function.

(c) The above mistake bound algorithm does not resemble the general Halving algorithm in the number of mistakes it is likely to make. This is due to the unusual nature of each function only outputting $True$ for a single, particular dimension vector. The algorithm will only be able to make a mistake and throw out other possible functions when it has received the desired function's lone example input.

3. Yes, the mistake bound for this new $C$ is still the same:
Proof: Given that we initialize $C$ to be the finite concept class and receive example $x$ to evaluate, only one *indicator function* in the entire set will output $False$. It then follows that the sequence of trial and error will be the exact same, except that the function we are searching for is the one we're looking to remove, not keep. Therefore, in this scenario the Halving algorithm would only make one mistake, again possibly seeing all examples for all other possible functions before arriving at the one example which *disagrees* with the "imposter" function.

# 3 The Perceptron Algorithm and its Variants

1. I chose to write my program in Python, since it is a widely accepted language for machine learning. Lists were used to represent the vector data. Although I read the SVM formatted data, these vectors were then expanded to include the missing attributes to make vector computations easier. Each variant of the perceptron algorithm inhabits its own function. Constants such as the random seed (49) and the learning rates ([1, 0.1, 0.01]) were initialized at the beginning of execution.

2. Using a majority baseline, the accuracy on the training set was 0.51435, and on the test set it was 0.54122.

3. Simple Perceptron:

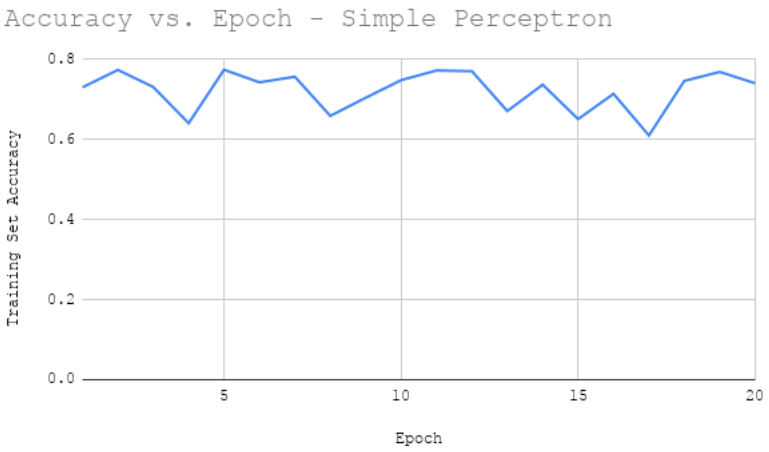
   (a) Best hyper-parameter (learning rate): 0.01

   (b) Corresponding cross-validation accuracy: 70.942%

   (c) Total number of updates performed on the training set: 11890

   (d) Training set accuracy: 74.081%

   (e) Test set accuracy: 74.552%




(f)

Decaying Perceptron:

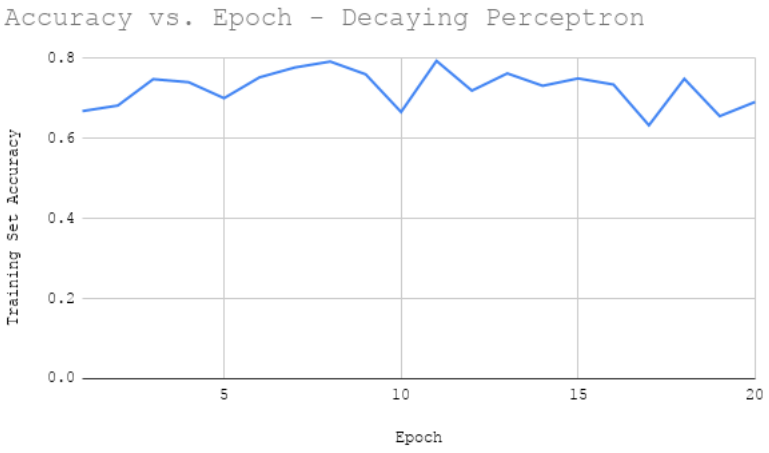
(a) Best hyper-parameter (learning rate): 0.1

(b) Corresponding cross-validation accuracy: 73.274%

(c) Total number of updates performed on the training set: 11773

(d) Training set accuracy: 69.148%

(e) Test set accuracy: 70.609%




(f)

Averaged Perceptron:

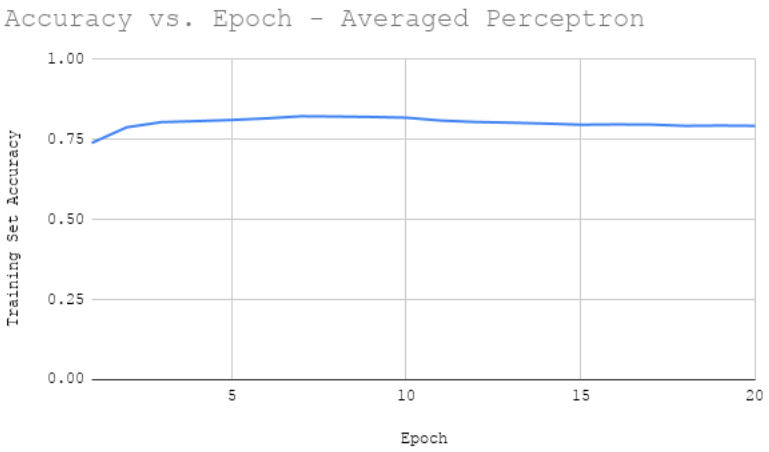
(a) Best hyper-parameter (learning rate): 0.1

(b) Corresponding cross-validation accuracy: 74.753%

(c) Total number of updates performed on the training set: 11802

(d) Training set accuracy: 79.283%

(e) Test set accuracy: 77.419%


Accuracy vs. Epoch - Averaged Perceptron
Training Set Accuracy
1.00
0.75
0.50
0.25
0.00
5
10
15
20
Epoch

(f)