Ritesh Sharma
Assignment 2: Machine Learning.

Problem 1 - Warm Up: Linear Classifiers and Boolean Functions

Built a truth table first and then tried different combinations till I found one that worked.

1. ¬x1 ∧ x2 ∧ ¬x3

0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

Answer: -x1 + x2 - x3 >= 1
This is linearly separable.

2. (x1 XNOR x2) XOR x3

0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1

1	1	0	1
1	1	1	0

Answer: It is not linearly separable because it has XOR in the equation.

3. x1 ∧ (¬x2 ∨ ¬x3)

0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

Answer: 2x1 - x2 - x3 >= 1
This is linearly separable.

4. (x1 ∧ ¬x2) ∨ (¬x1 ∧ x2) ∨ x3

0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1

1	1	0	0
1	1	1	1

Answer: This function is not linearly separable.

5. ¬(x1 ∧ ¬x2) ∨ x3

0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Answer: -x1 + 2x2 + 2x3 >= 1
This is linearly separable.

2 Feature Transformation.

1. **Answer**:
   Proof,
   We know,
   wTx + b >= 0, for every x ∈ X+ ------1
   and,
   wTx + b < 0, for every x ∈ X- -------2

   Now, for example 1
   [-1, -1] -
   Let w = [1], b = 1

wTx + b = [1] * [-1][-1] + 1
= -1 + 1 not strictly less than zero.
So, wTx + b >= 0 for [-1, -1] -

Example 2
[1][1] -
Let w = [1], b = 1

wTx + b = [1] * [1][1] + 1
= 2 not strictly less than zero.
So, wTx + b >= 0 for [1, 1] -

From two examples above we can see that the two conditions, 1 and 2 cannot be satisfied given any w and b.
Hence the following labeled set of points are not linearly separable.

2)
Answer: The function from number 1 cannot be separated linearly. Also a new transformation function that maps the same examples to a new space linear separation is not possible. The solution can logically be classified using two lines.
But,
φ(x 1 , x 2) -.> if x1 = x2 then 0 otherwise 1.

3)
Answer:
Since the function above behaves as a XOR function it cannot be linearly separated.

3. Mistake Bound Model of Learning

a) Determine |Cn,l|, the size of concept class

Answer:
= 2^l (hCl) [the term reads as 2 to the power l times h choose l]

T1 = $\sum_{i=1}^{l} 2^i$ (hCi) (this is equivalent to T1)

T2 = ($\sum_{i=1}^{l} 2^i$ (hCi)) - 1 (this is equivalent to T2

T3 = ($\sum_{i=1}^{l} 2^i$ (hCi)) - 2 (this is equivalent to T3

So the concept class is,
T1 * T2 * T3

= $\sum_{i=1}^{l} 2^i$ (hCi) * ($\sum_{i=1}^{l} 2^i$ (hCi)) - 1 * ($\sum_{i=1}^{l} 2^i$ (hCi)) - 2

= $\sum_{i=1}^{l} 2^i$ (hCi) * 2^i (hCi)) - 1 * 2^i (hCi)) - 2

b) Write a learning algorithm for this concept class that will only make a number of mistakes polynomial in the dimensionality. Please write the algorithm concisely in the form of pseudocode. Prove the mistake bound for this algorithm.
Input x, Clr. Output true function p(x)
Answer:
- Initialize C = Clr, the set of all possible functions
- For every example x
  - Prediction = wTx + b <= 0
  - If Prediction != y
    update.

This algorithm can make mistakes polynomial in the dimensionality. This is basically a perceptron algorithm. We can see that the concept class in not mutually exclusive as it is the polynomial function.

The sequence of the examples do not matter as the non zero output and the weight function will give a 0. The prediction will be completely based on what the previous weight was, because the prediction depends on the previous weight. The functions remaining in Clr will make polynomial mistakes.

Hence, we haven't learnt the correct function by making no more that polynomial mistakes.

2. **Extra Credit:**

Answer

a) Input x, C1. Output true function f(x)

Initialize C = Co, the set of all possible functions
For every example x
    Predict = majority of the function in Ci
    if Prediction != y
        Eliminate those functions which formed majority.

b) This functions behave as a halving function. It cannot make more that O(logn) mistakes.
If a majority function gives a 0, the prediction will be 0 which is in agreement with the true function.
x = z0 even when the function reports 1 and the majority function is 0. The function has an error in prediction resulting in elimination of majority.

4.4

What to report

1. [8 points] Briefly describe the design decisions that you have made in your implementation. (E.g, what programming language, how do you represent the vectors, etc.)

Answer:

Programming language : Python, Used numpy for vectors, main.py contains all the reading and manipulation, and perceptron class contains code for prediction.
List and arrays used.

2. [2 points] Majority baseline: Consider a classifier that always predicts the most frequent label. What is its accuracy on test and training set?

Answer:

Accuracy for training set = 50.54
Accuracy for test set = 48.23

3.

[10 points] For your best classifier, report the 10 words with the highest weights and

the 10 words with the lowest weights. Briefly discuss why the weights make sense for the words (Remember, these can be found in vocab idx.json).

Answer: This is done using the best classifier:

10 with highest: facility, has,inflatable,many,international,diseases,military,doctor,antibiotics,astronomy

10 with lowest: wife,with,would,241,sky,systems,saturn,program,results,160

The weights makes sense for the word because the words with the highest weight seems to be more common the data set we were given whereas it is opposite for the words with the lowest weights.

4. [10 points per variant] For each variant above (5 for 6350 students, 4 for 5350 students),
you need to report:
(a) The best hyper-parameters
(b) The cross-validation accuracy for the best hyperparameter
(c) The total number of updates the learning algorithm performs on the training set
(d) Training set accuracy
(e) Test set accuracy
(f) Plot a learning curve where the x axis is the epoch id and the y axis is the training set accuracy using the classifier (or the averaged classifier, as appropriate) at the end of that epoch. Note that you should have selected the number of epochs using the learning curve (but no more than 20 epochs).

Answers Below:

Answer:

Simple Perceptron

lr	Acc(mean)	Acc(std)
1.00	55.66	4.51
0.10	54.78	5.11
0.01	54.40	2.85

Best Learning Rate: 1
Total num of updates= 7270

Accuracy on best hyperparameter
train = 86.59
test = 60.10

Simple with Decaying

lr	Acc(mean)	Acc(std)
1.00	54.77	3.73
0.10	50.98	2.45
0.01	49.02	1.41

Best Learning Rate: 1
Total num of updates= 6662

Accuracy on best hyperparameter
train = 90.77
test = 64.39

Average Perceptron

mu	Acc(mean)	Acc(std)
1.00	61.04	2.42
0.10	62.05	1.40
0.01	59.52	1.76

Best Learning Rate: 0.1
Total num of updates= 7095

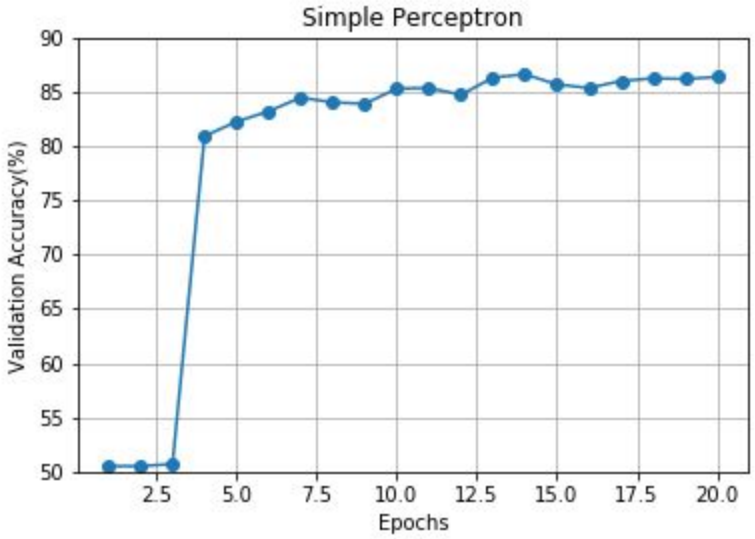
Accuracy on best hyperparameter
train = 81.21
test = 60.61

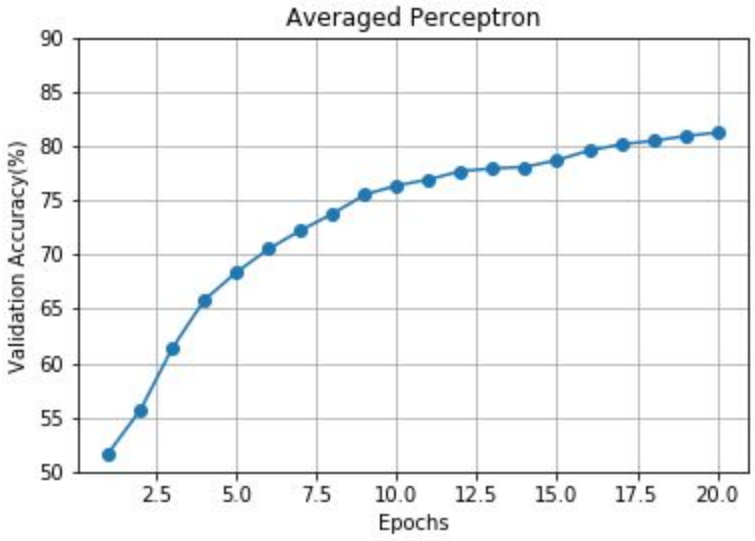
Pocket Perceptron

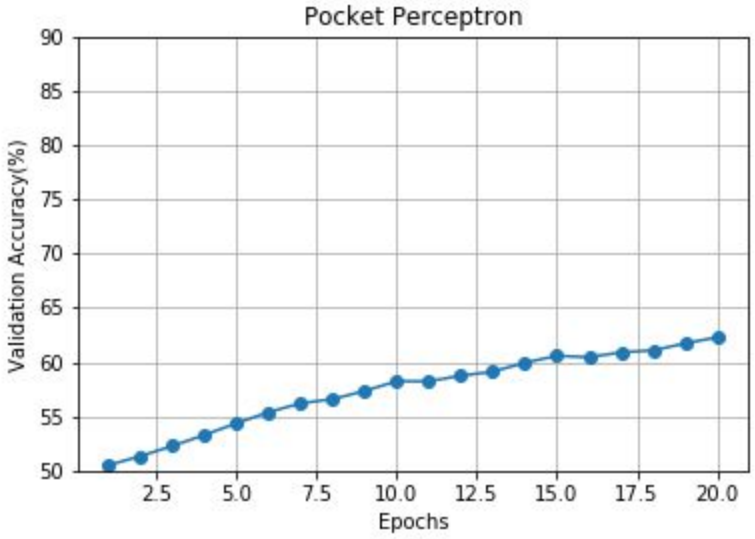
mu	Acc(mean)	Acc(std)
1.00	53.64	4.72
0.10	56.54	4.97
0.01	52.56	3.54

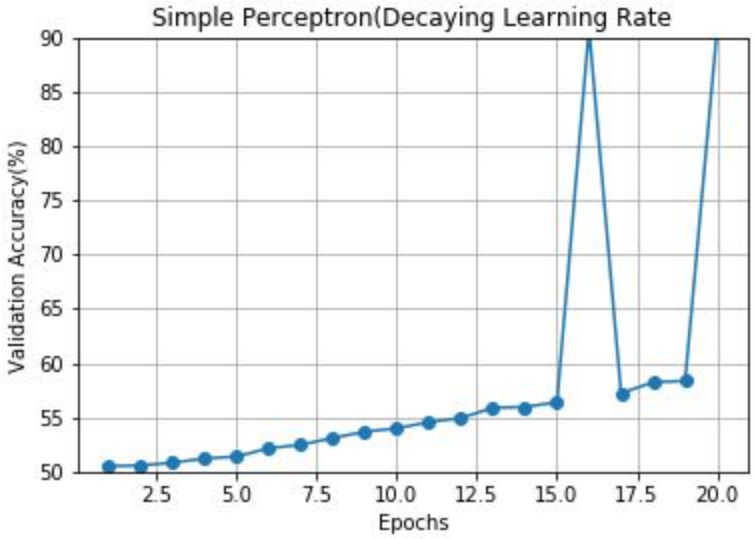
Best Learning Rate: 0.1
Total num of updates= 7113

Accuracy on best hyperparameter
train = 62.30
test = 50.51


Simple Perceptron
Validation Accuracy(%)
Epochs
90
85
80
75
70
65
60
55
50
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0


Averaged Perceptron
Validation Accuracy(%)
Epochs
90
85
80
75
70
65
60
55
50
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0


Pocket Perceptron
Validation Accuracy(%)
Epochs
90
85
80
75
70
65
60
55
50
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0


Simple Perceptron(Decaying Learning Rate
Validation Accuracy(%)
Epochs
90
85
80
75
70
65
60
55
50
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0