Machine Learning
Homework 2

Aaron Templeton

September 25, 2019

1: Linear Classifiers and Boolean Functions

(1) $\neg x_1 \wedge x_2 \wedge \neg x_3$

$$-x_1 + x_2 - x_3 - 1 \geq 0$$
$$w = [-1, 1, -1]$$
$$b = -1$$

(2) (x_1 XNOR x_2) XOR x_3

this is not linearly separable

(3) $x_1 \wedge (\neg x_2 \vee \neg x_3)$

$$2x_1 - x_2 - x_3 - 1 \geq 0$$
$$w = [2, -1, -1]$$
$$b = -1$$

(4) $(x_1 \wedge \neg x_2) \vee (\neg x_1 \wedge x_2) \vee x_3$

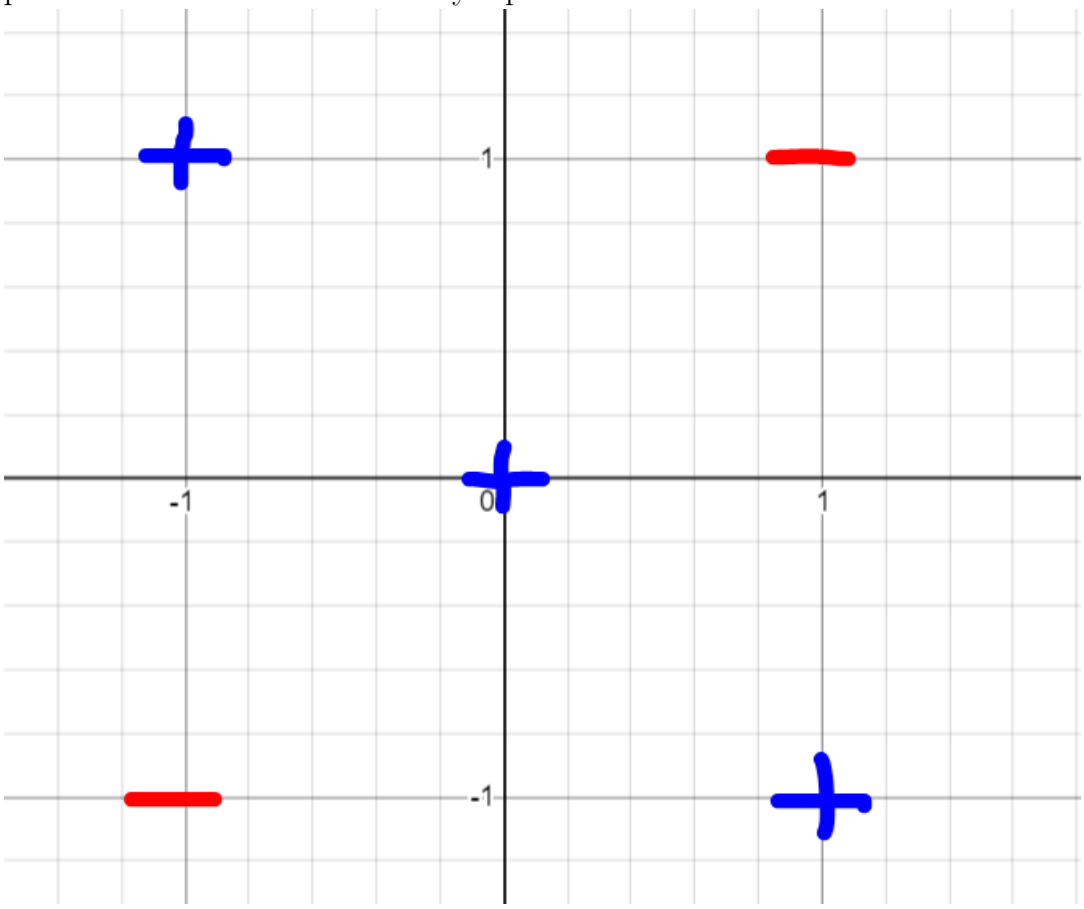
not linearly separable. parity is not separable

(5) $\neg(x_1 \wedge \neg x_2) \vee x_3$

$$-2x_1 + x_2 + x_3 + 1 \geq 0$$
$$w = [-2, 1, 1]$$
$$b = 1$$

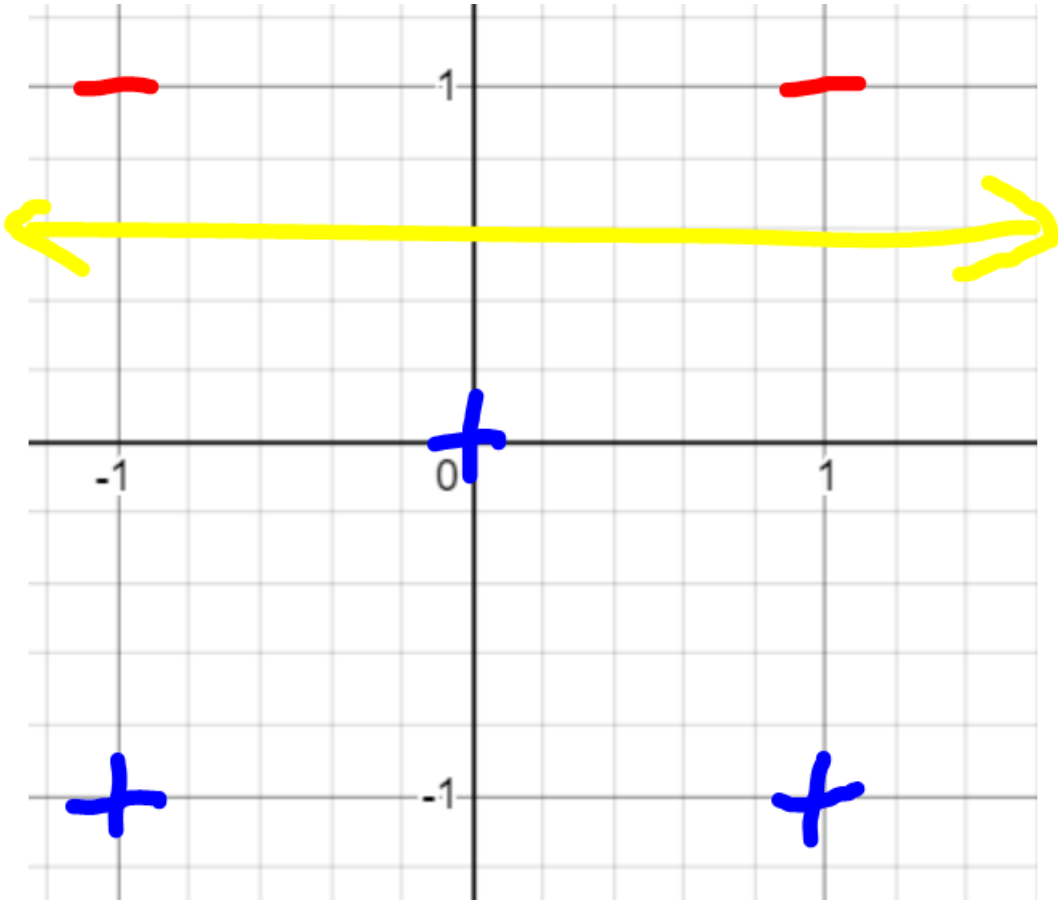
2: Feature Transformations

(1) prove that the data is not linearly separable in R^2



the labeled points cannot be separated in half or grouped together, by any line, therefore it is not linearly separable in R^2

(2) let the transformation function ϕ be $\phi(x_1, x_1x_2) = (z_1, z_2, z_3) = (x_1, x_2, x_1x_2)$
the map of the space is $\phi : R^2 \implies R^3$
the points on the graph after the function transformation are clearly separable by a line



(3)

$$w^T = [0, 0, -1]\ \phi(x_1, x_2) - 1 \geq 0$$

$$b = -1$$

3: Mistake Bound Model of Learning

(1) (a) determine $|C_{n,l}|$ the size of the concept class

the size of C is at most n

in the example, n =4 so the size of C is 4

(b) write a mistake bound learning algorithm for this concept class that will run in time polynomial to n.

Algorithm 1 Mistake bound algorithm

for all x_i in x and y_i **do**
  if $y^T = 1$ and $T1 + T2 + T3 \geq 2$ **then**
    $w = w + ry_ix_i$
  else
    if $y^T = -1$ and $T1 + T2 + T3 < 2$ **then**
      $w = w - ry_ix_i$
    end if
  end if
end for
return w

4: The Perceptron Algorithm

1. Design Decisions
   i used python in my implementation because the jupyter notebook was helpful and so was the libsvm module to read the libsvm files. I represented the vectors using simple arrays. I made my implementation step-by-step by first building around the provided test data. I then made cvfolds and started using the train and test data provided.

2. Majority Baseline
   the accuracy is approximately 50%

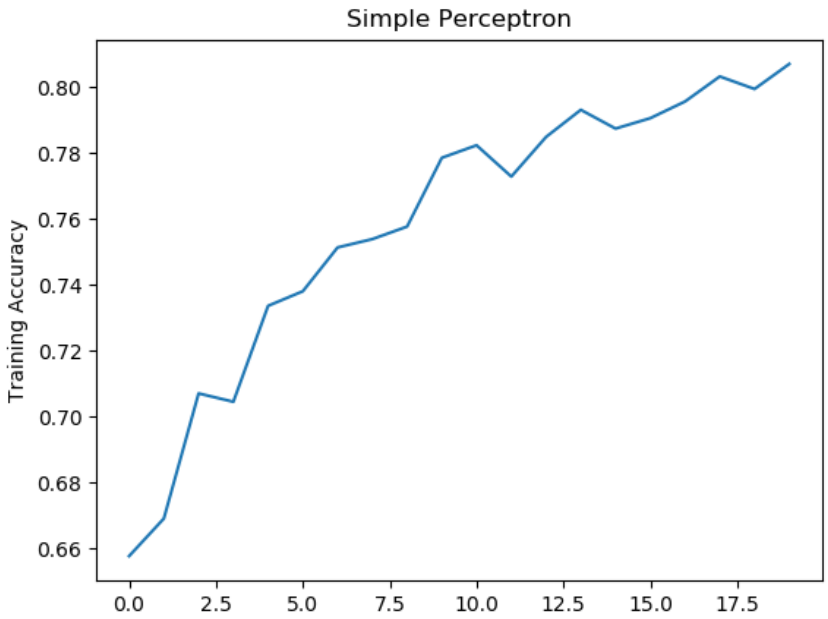
3. Highest/Lowest Weights
   Top 10 words
   cause
   health
   medical
   medicine
   experience
   normal
   med
   concerned
   surgery
   disease

   Bottom 10 words
   solar
   satellite
   station
   earth
   big
   launch
   jupiter
   nasa

orbit
space

Perceptron Variants

1. Simple Perceptron
   (a) best hyper-parameters - 0.1
   (b) cross-validation accuracy 0.75
   (c) number of updates - 21024
   (d) training accuracy - .807
   (e) test accuracy - .767

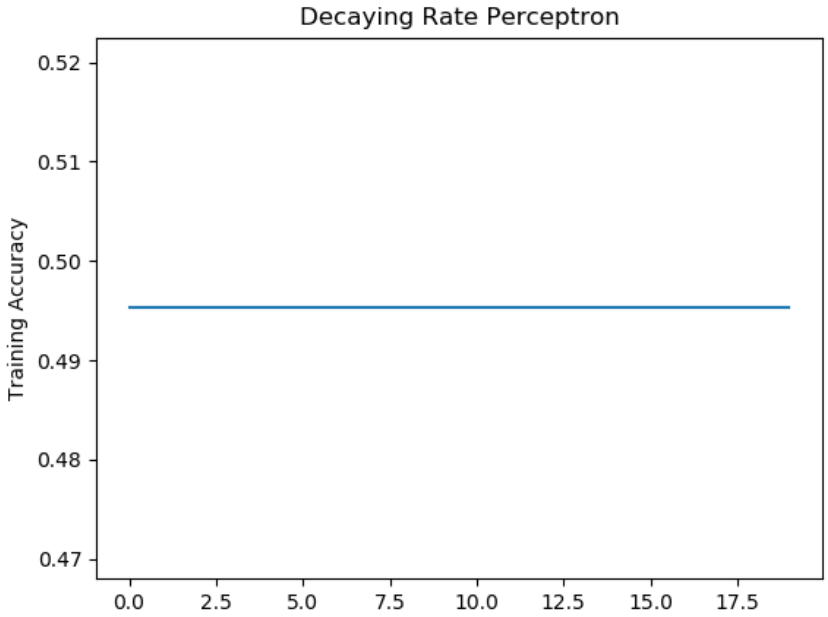


plots/simple.png
I built simple perceptron by following the guidelines in the provided jupyter notebook. Once i completed the jupyter notebook tutorial with the test data, I used the given train and test data files and did the report

2. Decaying Rate Perceptron
   (a) best hyper-parameters - 0.01
   (b) cross-validation accuracy 0..49
   (c) number of updates - 40084
   (d) training accuracy - .495

(e) test accuracy - .487

(f) plot - run code to see plot



plots/decay.png Decaying rate perceptron was the same as simple perceptron except in the train method, I added a boolean parameter to check if we want to run decaying-rate. if True, the rate is decayed per the assignment instructions on each epoch

3. Averaged Perceptron

   (a) best hyper-parameters - 1

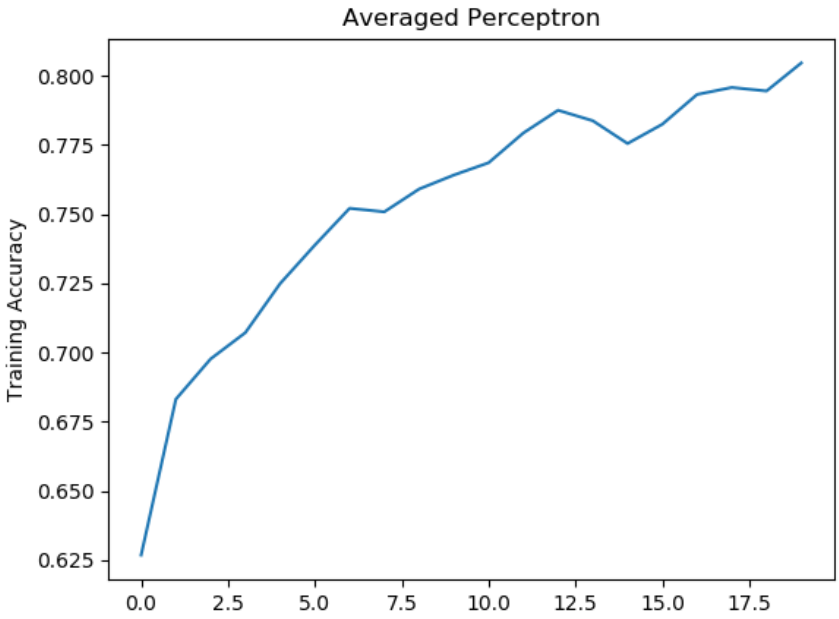
   (b) cross-validation accuracy 0.83

   (c) number of updates - 21074

   (d) training accuracy - .804

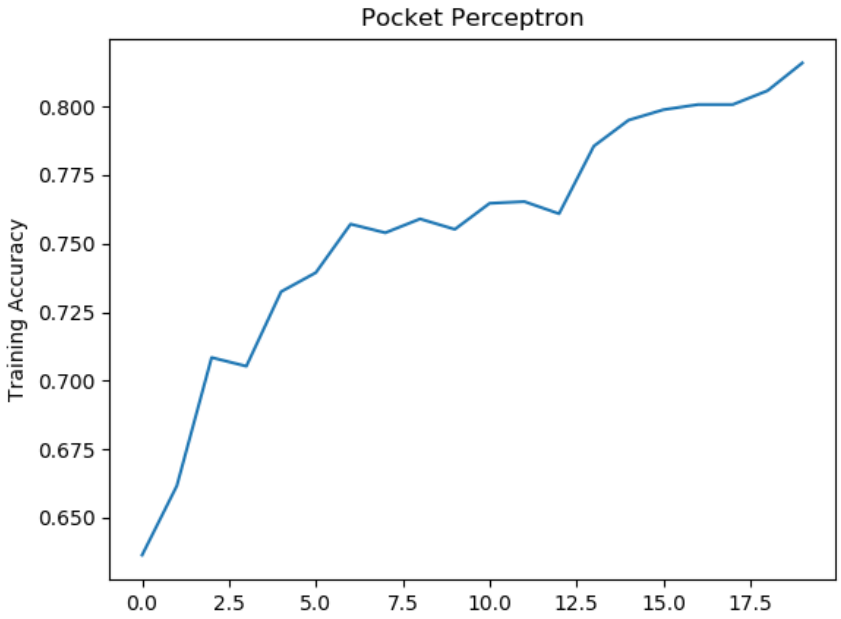
   (e) test accuracy - .873

   (f) plot - run code to see plot



plots/avg.png for averaged perceptron, I added another boolean paramater to the train method for averaged perceptron. i also used some global variables for the average weights and average bias. I kept a global highest accuracy and if there was higher then i updated the averaged weights and average bias

4. Pocket Perceptron
   (a) best hyper-parameters - 0.1
   (b) cross-validation accuracy 0.748
   (c) number of updates - 21088
   (d) training accuracy - .815
   (e) test accuracy - .775
   (f) plot - run code to see plot



plots/pocket.png

pocket perceptron was implemented similarly to averaged perceptron. I added another boolean param to train for pocket perceptron and i also added global variables for pocket weight and pocket bias. i updated them every epoch