CS440 : Introduction to Artificial Intelligence **Spring 2014**

Problem Set 2 - Part 2

Dan McQuillan *Handed In: February 18, 2014*

1. **Solution to problem 2**
   The weight vector values were largely different as shown below:

old	**new**
$\theta = -54.0$	$\theta = 513.0$
$w_1 \rightarrow -752.0$	$w_1 \rightarrow -1719.0$
$w_2 \rightarrow 4771.0$	$w_2 \rightarrow 2216.0$
$w_3 \rightarrow 17714.0$	$w_3 \rightarrow 1357.0$
$w_4 \rightarrow 762.0$	$w_4 \rightarrow 1506.0$
$w_5 \rightarrow 6.0$	$w_5 \rightarrow 435.0$
$w_6 \rightarrow 676.0$	$w_6 \rightarrow 1152.0$
$w_7 \rightarrow 3060.0$	$w_7 \rightarrow -300.0$
$w_8 \rightarrow -2004.0$	$w_8 \rightarrow -1928.0$
$w_9 \rightarrow 5459.0$	$w_9 \rightarrow 1462.0$
$w_{10} \rightarrow 9591.2999999989$	$w_{10} \rightarrow 991.9000000000113$
$w_{11} \rightarrow 3832.0$	$w_{11} \rightarrow -775.0$
$w_{12} \rightarrow 14963.0$	$w_{12} \rightarrow 6726.0$
$w_{13} \rightarrow 20912.0$	$w_{13} \rightarrow 11721.0$

   The confusion matrix also was also more error prone when testing data set 1. It resulted in the following confusion matrix:

53	1
9	54

   The application to data set 3 produced largely different conclusions.

   However, the most important feature's weight value stayed relatively the same which shows that it has more influence on the Δw on each iteration of the inputs. This is as expected since because that feature is the most important.

2. **Solution to problem 3**

   (a) Find and report the list MAX for Dataset 2
   $(70.0, 1.0, 4.0, 170.0, 407.0, 1.0, 2.0, 194.0, 1.0, 4.0, 3.0, 3.0, 7.0)$

   (b) Repeat Problem 1

   Training Dataset 1x

The training took 14 epochs.
Threshhold : 5.0
$\gamma \rightarrow 0.0320963696907362$
$w_1 \rightarrow -1.2368421052631573$
$w_2 \rightarrow 3.0$
$w_3 \rightarrow 5.5$
$w_4 \rightarrow -0.2899999999999997$
$w_5 \rightarrow -0.13829787234042568$
$w_6 \rightarrow 0.0$
$w_7 \rightarrow 0.5$
$w_8 \rightarrow -13.317204301075275$
$w_9 \rightarrow 3.0$
$w_{10} \rightarrow 5.645161290322578$
$w_{11} \rightarrow 2.9999999999999987$
$w_{12} \rightarrow 5.0$
$w_{13} \rightarrow 5.7142857142857135$

Test Dataset 1x

Confusion Matrix:

54	0
0	63

Total loss: 0.0

Test Dataset 2x

False Positives:
Index: 7
Confusion matrix:

13	0
1	19

Total loss: 0.010623925503195387

Application Dataset 3x

$1 \rightarrow 0.0$
$2 \rightarrow 1.0$
$3 \rightarrow 0.0$
$4 \rightarrow 1.0$
$5 \rightarrow 1.0$
$6 \rightarrow 0.0$

$7 \rightarrow 0.0$
$8 \rightarrow 0.0$
$9 \rightarrow 0.0$
$10 \rightarrow 1.0$
$11 \rightarrow 1.0$
$12 \rightarrow 0.0$
$13 \rightarrow 1.0$
$14 \rightarrow 0.0$
$15 \rightarrow 0.0$
$16 \rightarrow 0.0$
$17 \rightarrow 1.0$
$18 \rightarrow 0.0$
$19 \rightarrow 0.0$
$20 \rightarrow 0.0$
$21 \rightarrow 0.0$

(c) Explanation
The results above show that the number of epoch were quite a bit less. This is a result of the perceptron having to oscillate because of the larger difference between input vectors. Instead it is more likely to increase with a smaller Δw and have a lesser chance to have to use a negative Δw to correct a movement when it moves past the margin of a "good" perceptron.

(d) Test new perceptron on DataSet1
When testing data set 1 it resulted in a complete failure of classifying the input vectors from dataset one. This is a result of normalizing the input vectors before training which caused the weight vector to be much smaller. Although we normalized the input vectors that does not correlate to the weight vector itself being normalized.

(e) Is is possible to recover the problem 1 perceptron
It is not possible to recover the perceptron from problem 1 from this weight vector since the resultant weight vectors are dependent on the values chosen to be the initial training weights. It will not also be able to be recovered since the test on DataSet1 failed which means that the resultant vector from part d is not the weight vector normalized but instead it has a different direction entirely. Since it lacks this corollary it is not possible to recover the perceptron from problem 1. In addition, even if we used the resultant weight vector as the initialization of the perceptron, then trained the weight vector we would still be left with a different perceptron.

3. **Solution to problem 4**

(a) Train on DataSet4, test on DataSet1

Training Results:

The training took 1572 epochs.
Threshhold: 612.0
$\gamma \to 0.08551637281027642$

$w_1 \to -449.0$
$w_2 \to 2997.0$
$w_3 \to 2286.0$
$w_4 \to 927.0$
$w_5 \to 526.0$
$w_6 \to 361.0$
$w_7 \to -1940.0$
$w_8 \to -2171.0$
$w_9 \to 1543.0$
$w_{10} \to 3807.299999999868$
$w_{11} \to 166.0$
$w_{12} \to 5163.0$
$w_{13} \to 13713.0$

Testing Results
Confusion Matrix:

52	2
9	54

Total loss: 23.891007156505278

(b) Train on DataSet5, test on DataSet1

Training Results:
The training took 1737 epochs.
Threshhold: 650.0
$\gamma \to 0.07955877280170609$

$w_1 \to -1200.0$
$w_2 \to 3007.0$
$w_3 \to 1766.0$
$w_4 \to 1677.0$
$w_5 \to 555.0$
$w_6 \to -21.0$
$w_7 \to -1560.0$
$w_8 \to -2545.0$
$w_9 \to 1421.0$
$w_{10} \to 2497.999999999887$
$w_{11} \to -135.0$

$w_{12} \rightarrow 7571.0$
$w_{13} \rightarrow 12296.0$

Testing Results
Confusion Matrix:

52	2
10	53

Total loss: 31.41886183116968

(c) Train on DataSet6, test on DataSet1

Training Results:
The training took 3437 epochs.
Threshhold: 791.0
$\gamma \rightarrow 0.00675989841026674$

$w_1 \rightarrow 1374.0$
$w_2 \rightarrow 3520.0$
$w_3 \rightarrow 3954.0$
$w_4 \rightarrow -271.0$
$w_5 \rightarrow 538.0$
$w_6 \rightarrow 635.0$
$w_7 \rightarrow -5986.0$
$w_8 \rightarrow -2088.0$
$w_9 \rightarrow 1518.0$
$w_{10} \rightarrow 5392.699999999808$
$w_{11} \rightarrow 778.0$
$w_{12} \rightarrow 7076.0$
$w_{13} \rightarrow 18931.0$

Testing Results
Confusion Matrix:

52	2
8	55

Total loss: 16.247427541193762

(d) Which of the four yields the best performance?
Testing results dataset 2:
Confusion Matrix:

53	1
9	54

Total loss: 23.31131267461353

The test training from **DataSet6** had the best performance since we are looking at minimum loss as a determining factor in the success of the perceptron. The results will not always be identical since the permutations random. This causes the ordering of the operations performed on the weight vector during training to be different for each result of shuffling DataSet2.

(e) Test averaged weight vector on dataset 1.
Training on dataset2:
The training took 1637 epochs.
Threshhold: 513.0
$\gamma \to 2.7031724654225244 \times 10^{-4}$

$w_1 \to -1719.0$
$w_2 \to 2216.0$
$w_3 \to 1357.0$
$w_4 \to 1506.0$
$w_5 \to 435.0$
$w_6 \to 1152.0$
$w_7 \to -300.0$
$w_8 \to -1928.0$
$w_9 \to 1462.0$
$w_{10} \to 991.9000000000113$
$w_{11} \to -775.0$
$w_{12} \to 6726.0$
$w_{13} \to 11721.0$

Averaged weight vector:
$\theta \to 0.03770903915138822$

$w_1 \to -0.041178942494859094$
$w_2 \to 0.17274616057058972$
$w_3 \to 0.13148707571728$
$w_4 \to 0.06475623501863624$
$w_5 \to 0.030680548706339232$
$w_6 \to 0.032487241176698975$
$w_7 \to -0.12672389222628322$
$w_8 \to -0.13156694037154992$
$w_9 \to 0.08926847383363394$
$w_{10} \to 0.17649885545001573$

$w_{11} \to -0.004544798237466599$
$w_{12} \to 0.3980499596908348$
$w_{13} \to 0.8258489971867217$

Results:
False Positives
Index: 5
Inputs: (67.0, 0.0, 3.0, 115.0, 564.0, 0.0, 2.0, 160.0, 0.0, 1.6, 2.0, 0.0, 7.0)
Index: 6
Inputs: (65.0, 0.0, 3.0, 140.0, 417.0, 1.0, 2.0, 157.0, 0.0, 0.8, 1.0, 1.0, 3.0)
Index: 8
Inputs: (65.0, 0.0, 3.0, 160.0, 360.0, 0.0, 2.0, 151.0, 0.0, 0.8, 1.0, 0.0, 3.0)
Index: 20
Inputs: (64.0, 0.0, 3.0, 140.0, 313.0, 0.0, 0.0, 133.0, 0.0, 0.2, 1.0, 0.0, 7.0)
Index: 31
Inputs: (69.0, 1.0, 1.0, 160.0, 234.0, 1.0, 2.0, 131.0, 0.0, 0.1, 2.0, 1.0, 3.0)
Index: 33
Inputs: (64.0, 0.0, 4.0, 180.0, 325.0, 0.0, 0.0, 154.0, 1.0, 0.0, 1.0, 0.0, 3.0)
Index: 53
Inputs: (74.0, 0.0, 2.0, 120.0, 269.0, 0.0, 2.0, 121.0, 1.0, 0.2, 1.0, 1.0, 3.0)
Index: 77
Inputs: (66.0, 0.0, 1.0, 150.0, 226.0, 0.0, 0.0, 114.0, 0.0, 2.6, 3.0, 0.0, 3.0)
Index: 114
Inputs: (60.0, 0.0, 3.0, 120.0, 178.0, 1.0, 0.0, 96.0, 0.0, 0.0, 1.0, 0.0, 3.0)

False Negatives
Index: 34
Inputs: (53.0, 1.0, 4.0, 140.0, 203.0, 1.0, 2.0, 155.0, 1.0, 3.1, 3.0, 0.0, 7.0)
Index: 69
Inputs: (60.0, 1.0, 4.0, 117.0, 230.0, 1.0, 0.0, 160.0, 1.0, 1.4, 1.0, 2.0, 7.0)

Confusion Matrix:

52	2
9	54

Total loss: 22.263379141433237

The results were successful in classifying data set 1 with minimal error. Unlike problem 3 we are not just dividing each input vector by it's corresponding list max. We are instead using the full dataset 2 and permuting it. Then by taking

the unit vector of the resultant weight vector of each and then averaging them, it results in giving us a resultant weight vector near the range of "good" perceptrons since we are not changing the direction of the weight vector, only the magnitude. This results from the concept that each learning of a permuted dataset would give us a perceptron in the range of its "good" perceptrons. Therefore, if we took more permutations of the input vectors and ran it we could get a resultant weight vector with even less loss.