CS 6350: Machine Learning – Homework 2

Clinton Fernandes, u1016390 09/27/2016

1 Warm up: Feature expansion

The function $\phi(x_1, x_2) \equiv (\ x_1,\ x_2,\ 4x_1^4 + 16x_2^4)$

Weight vector : $\mathbf{w} = [0\ 0\ -1]^T$

bias term : $\mathbf{b} = -r$

Now putting the above terms in the form $w^T\phi(x_1, x_2) \geqslant b$:

$[0\ 0\ -1][\ x_1,\ x_2,\ 4x_1^4 + 16x_2^4]^T \geqslant -r$

$-(x_1,\ x_2,\ 4x_1^4 + 16x_2^4) \geqslant -r$

$4x_1^4 + 16x_2^4 \leqslant r$

The above equation $w^T\phi(x_1, x_2) \geqslant b$ is satisfied when $f_r(x_1, x_2)$ is +1, i.e. when $4x_1^4 + 16x_2^4 \leqslant r$

2 Mistake Bound Model of Learning

1. The size of the concept class is $|C|$ = 80
   Concept class is the set of functions from which the true target function is drawn. Here, the number of possible functions depend on the number of possible values that r takes, which is from 1 to 80. Hence $|C|$ = 80.

2. $label = y^t - sgn(r^2 - (x_1^2 + x_2^2))$
   or
   $label = y^t - f_r(x_1^t, x_2^t)$, here $f_r(x_1^t, x_2^t) = sgn(r^2 - (x_1^2 + x_2^2))$
   If y^t and $f_r(x_1^t, x_2^t)$ are not same, i.e., when one is +1 and other is -1, $label$ = $\pm$ 2, then mistake is made. When y^t and $f_r(x_1^t, x_2^t)$ are same, i.e., when both are +1 or both are -1, label = 0, no mistake is made by f_r.

3. If label == +2 :
   - r_min = r + 1
   - pick r randomly from C {r_min, r_max}

   If label == -2 :
   - r_max = r - 1
   - pick r randomly from C {r_min, r_max}

   If label == 0 : do not change r

To elaborate the above steps:

- The label is +2 when y^t is +1 and $f_r(x_1^t, x_2^t)$ is -1; r should be bigger than current radius. For this, we redefine the concept class by changing its minimum value to one integer more than current radius. And then we pick r randomly from the concept class.
- The label is -2 when y^t is -1 and $f_r(x_1^t, x_2^t)$ is +1; r should be smaller than current radius. For this, we redefine the concept class by changing its maximum value to one integer less than current radius. And then we pick r randomly from the concept class.
- If label is 0 than do not update r.

4. let C be the concept class having r; C = [1 $\leqslant$ r $\leqslant$ 80]; C = [r_min $\leqslant$ r $\leqslant$ r_max]

initialize r_min = 1 and r_max = 80;
initialize Choose randomly r $\in$ Ci;
for *All examples* **do**
  if *r_min == r_max* **then**
    return r;
  $label = y^t - f_r(x_1, x_2)$;
  if *label == 0* **then**
    continue;
  else if *label == +2* **then**
    r_min = r + 1;
    Choose randomly r $\in C_{i+1}$;
  else if *label == -2* **then**
    r_max = r - 1;
    Choose randomly r $\in C_{i+1}$;
end

Algorithm 1: Mistake-driven learning algorithm

Maximum number of mistakes made:

In the final set of Concepts C_n, only 1 element remains. Also, in the worst case, exactly 1 element is removed form the concept class after each mistake is made.

If the algorithm was initialized from an extreme value in the range of concepts, i.e. either r = 1 or r = 80, then the maximum possible number of mistakes made is |C|-1. In this case it is 79.

5. (a) For this particular problem, the set of hypotheses consistent with all examples seen so far can be defined by storing two integers, the extreme values of the current concept class (minimum and maximum values of the concept class).

   for example, if the range of C is [5, 6, 7, 8, 9, 10, 11], then the two integers are '5' and '11'. These integers define the range of hypotheses that are consistent with all the examples seen so far.

(b) For all the hypotheses' in the current concept class, find the value of $f_r(x_1, x_2)$.
Note; $f_r(x_1^t, x_2^t) = sgn(r^2 - (x_1^2 + x_2^2))$
Depending on whether the value of $f_r(x_1, x_2)$ is +1 or -1, store it in either of two temporary sets.
Now take the label corresponding to the set with more size as the 'predicted label'
Now check if the 'predicted label' is equal to y^t
If 'predicted label' $\neq$ y^t then an error has occurred.

(c)

let C be the concept class having r; C = [1 $\leqslant$ r $\leqslant$ 80]; C = [r_min $\leqslant$ r $\leqslant$ r_max]
initialize r_min = 1 and r_max = 80;
for *All examples* **do**
  if *r_min == r_max* **then**
    return r;
  for *all values of r in* C_i **do**
    if $f_r(x_1, x_2)$ == *+1* **then**
      add r to set_of_positive_label;
    else if $f_r(x_1, x_2)$ == *-1* **then**
      add r to set_of_negative_label;
  end
  predicted label = label of majority set;
  if *predicted label* $\neq$ y^t **then**
    Update *r_min = minimum value of minority set;*
    Update *r_max = maximum value of minority set;*
end

Algorithm 2: Halving algorithm

Mistake bound on this halving algorithm = log|C|
Suppose that the algorithm makes n mistakes and the final set of concepts C_n has one element.
$1 = |C_n| < \frac{1}{2}|C_{n-1}| < \frac{1}{2}|C_{n-2}| < ... < \frac{1}{2^n}|C_0|$
$|C_0| > 2^n$, $n = log_2(|C_0|) = log_2(|C|)$
For this particular problem, |C| = 80, so mistake bound is $log_2(80)$

3 The Perceptron Algorithm and Its Variants

1. The weight vector (with first element as the bias term): $w^T = [0\ 0\ 1\ 0\ -1\ 2]$
   Number of mistakes made: 4

2.

	Perceptron	Margin Perceptron
Number of Updates	1373	1389
Accuracy (training) %	81.35	82.02
Accuracy (test) %	81.09	81.47

- The Perceptron algorithm and the margin Perceptron were run on the Adult data for one pass and the results obtained form the exercise is shown in the above table.
- Experimenting with different hyperparameters, it was seen that a rate of '1' and Margin of '1e-06' gave the best results.

3.

Results for 'Perceptron'

	3 epochs		5 epochs	
	no shuffle	with shuffle	no shuffle	with shuffle
Number of Updates	4078	4056	6752	6736
Accuracy (training) %	79.83	82.46	83.29	81.26
Accuracy (test) %	78.89	81.34	82.79	80.14

Results for 'Margin Perceptron'

	3 epochs		5 epochs	
	no shuffle	with shuffle	no shuffle	with shuffle
Number of Updates	4054	4097	6747	6696
Accuracy (training) %	79.62	82.85	80.23	80.87
Accuracy (test) %	79.4	81.79	79.78	80.46

- The results for the Perceptron and the Margin perceptron for 3 and 5 epochs, along with shuffled and unshuffled data are shown in the two tables above tables.
- For the Perceptron implementation, it is seen that the best accuracy on the training as well as the test data was obtained when the number of epochs were 5 and the data was not shuffled after every epoch. The accuracy did not seem to improve whne the data was shuffled for 5 epochs. {It is to be noted that, when the code was run during differernt times, the above values in the table changed}
- For the Margin perceptron implementation, the best accuracy on the training as well as the test data was at the end of 3 epochs when the data was shuffled. It can also be noted that the accuracy was more during 5 epochs, when the data was shuffled.
- When we try to compare the Perceptron and the Margin perceptron implementation, specially for the 3 epoch condition, the number of Updates, accuracies appear to be similar. While for the 5 epoch case, Perceptron seems to do well than the margin perceptron. As such, it is difficult ot compare both the variants of perceptrom due to the random initialization of the weights and bias, which was different in both the implementations.

4. **Results for 'Aggressive Margin Perceptron'**

	3 epochs		5 epochs	
	no shuffle	with shuffle	no shuffle	with shuffle
Number of Updates	4072	4102	6720	6722
Accuracy (training) %	80.57	83.26	80.34	81.04
Accuracy (test) %	79.77	82.47	79.57	81.34

- The best accuracy on the training and the test data were found when the number of epochs were 3 and the data was not shuffled before each epoch.
- In this experiment, it can be noted that the accuracy improves with shuffling.
-

Conclusions/Notes about experiments (Question 3)

- When the implementation of the above variants of perceptron was run using python, for multiple times, it was observed that the number of updates and the accuracy of the final weight vector on both the train data and the test data would change with every run.
- This is due to the fact that the weights and the bias for each of the algorithms were initialized randomly. It was seen that, if the weights and bias were initialized to zeros, the accuracy would remain the same for different runs of the code.
- It can be observed that shuffling the data before each epoch would yield good accuracy, for most of the experiments.