CS 5350/6350: Machine Learining Fall 2016

Homework 2

Gopal Menon

September 25, 2016

1 Warm up: Feature expansion

1. Consider the following function that maps the examples in $\Re^2$ into a higher space $\Re^3$:

$$\phi : (x_1, x_2) \rightarrow (x_1, x_2, f_r(x_1, x_2))$$

This has the effect of raising the positively labelled examples in the newly introduced dimension. I'm not sure about the mathematical notation, but my intention is to map the function from two-dimensional to three-dimensional space.

2. In order to verify that this achieves a linear separation between the positive and negative examples, I will define a weight vector w which includes a bias term as follows:

$$w = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

The dot product of the weight vector transpose and an example (here the example has a dimension in addition to the added third dimension in order to accommodate the bias) will be

$$\begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ f_r(x_1, x_2) \end{bmatrix} = \begin{cases} +1 & 4x_1^4 + 16x_2^4 \leq r; \\ -1 & \text{otherwise} \end{cases}$$

This means that the weight vector w can be used to separate out the positive and negative labelled examples. The hyperplane that separates the positive and negative labelled examples will pass through the origin (since the bias is zero) and will be perpendicular to the weight vector w. The dot product of the weight vector and any example will give the distance of the example point from the linearly separating plane (since it's a projection of the example vector on the weight vector) that separates the positive and negative examples. We can see that the dot product is $+1$ for positively labelled examples and is -1 for negatively labelled examples. Hence the weight vector w separates out the examples with different labels.

2 Mistake Bound Model of Learning

1. Each function f_r in the concept class $\mathcal{C}$ is defined by a radius r. Since $1 \leq r \leq 80$ and r is being compared with the sum of the squares of two integers, we need only consider integral values of r. So each function f_r in the concept class $\mathcal{C}$ that needs to be considered, will have a different integral value of r. So $|\mathcal{C}| = 80$.

2. [5 points] We need to check if the following equality is true

$$y^t = \begin{cases} +1 & (x_1^t)^2 + (x_2^t)^2 \leq r^2; \\ -1 & \text{otherwise} \end{cases}$$

   If it is not true then it means that the hypothesis f_r has made a mistake.

3. [10 points] Consider the case when the label is -1 and the prediction is $+1$ because $x_1^2 + x_2^2 \leq r^2$. In order to correct this, we need to update r to make it $x_1^2 + x_2^2 > r^2$ or $r = \left\lfloor \sqrt{x_1^2 + x_2^2 - 1} \right\rfloor$.

   Consider the case when the label is $+1$ and the prediction is -1 because $x_1^2 + x_2^2 > r^2$. In order to correct this, we need to update r to make it $r = \left\lceil \sqrt{x_1^2 + x_2^2 + 1} \right\rceil$.

   In both cases above, we need to consider only the positive value of the square root.

4. [20 points] Here is a mistake-driven learning algorithm to learn the function.

Algorithm 1 Mistake-Driven Learning Algorithm

```
1: procedure MISTAKE-DRIVEN LEARNING ALGORITHM($x_1, x_2, y$)
2:     if $x_1^2 + x_2^2 \leq r^2$ then
3:         if y == $-1$ then
4:             $r = \left\lfloor \sqrt{x_1^2 + x_2^2 - 1} \right\rfloor$
5:     else
6:         if y == $+1$ then
7:             $r = \left\lceil \sqrt{x_1^2 + x_2^2 + 1} \right\rceil$
```

   Here the algorithm receives as input the values of x_1, x_2 and the label y. It then uses these values to update the value of r that it maintains in its internal state. In the algorithm above, == represents the test for equals and = represents an assignment.

Since the correct function will use a value of r between 1 and 80, the worst case scenario for learning the correct function will be the case where all the functions with the incorrect value of r are first tried and the test data results in a wrong prediction in each such case. So the correct function will be the last one tried and will be found after making 79 (that is $|\mathcal{C} - 1|$) mistakes.

5. a. The set of hypotheses consistent with all examples seen so far can be defined by storing the upper and lower values of the range of r values that satisfy the examples seen so far.

   b. [5 points] At any point in the iteration of the halving algorithm, we can check and see if the value of r^2 corresponding to the lowest value of r in the range of r values in the top half of the ranges of r satisfies the following

$$y^t = \begin{cases} +1 & (x_1^t)^2 + (x_2^t)^2 \leq r^2; \\ -1 & \text{otherwise} \end{cases}$$

   c. [5 points] The halving algorithm can be as follows:

Algorithm 2 Halving Algorithm

```
1: procedure HALVING ALGORITHM($x_1^t, x_2^t, y^t$)
2:     Construct sets $R_1$ and $R_2$ by splitting the sorted set $R$ of remaining $r$
       values down the middle. The split is made such that for the case of odd
       number of $r$ values, the set $R_2$ will contain one more element than $R_1$
3:     if $|R_1| == |R_2|$ and $|R_1| \neq 1$ then
4:         Remove largest vaue of $r$ from $R_1$ and put it into $R_2$
5:     $r_t$ = minimum value of r in set $R_2$
6:     if $(x_1^t)^2 + (x_2^t)^2 \leq r_t^2$ and $y^t == -1$ then
7:         $R = R_1$
8:         if $|R| == 1$ then
9:             The function has been learnt and is $f_r$ where $r$ = the element in set $R$
```

In the above halving algorithm, in step 4, the set R_2 is made the majority set if it is not already one. In the case where there is only one element left in each split set, there is no change made. The majority step is used to make a prediction. If the prediction is wrong, the entire set is dropped from the list of potential values of r that will be considered to be the value used in the target function. This step where the majority set is checked is shown in line 6.

The halving algorithm discards at least half the functions from the hypothesis set each time its makes a mistake in the prediction. In the worst case, the algorithm discards exactly half the functions from the hypothesis set. This means that it uses at most $log_2 |\mathcal{C}|$ steps to arrive at the correct function. Here $\mathcal{C}$ is the concept class that the algorithm searches over. This means that the mistake bound (the number of steps in the worst case) is $log_2 80$.

3 The Perceptron Algorithm and Its Variants

3.1 The Task and Data

3.2 Algorithms

3.3 Experiments

1. The weight vector at the end of the run was $[0.0, 0.0, 1.0, 0.0, -1.0, 2.0]^T$. The number of mistakes made was 4.

2. 6-fold cross validation was run for finding the hyper-parameters for the Perceptron and Margin Perceptron.

   For the Perceptron, a learning rate of 0.2 was selected after cross-validation. 1382 mistakes were made during the training process. The accuracy on the training set (calculated in all cases of this assignment by adding true positives and true negatives and dividing by total number of training samples) was 0.7989, and was 0.7941 on the testing set.

   For the Margin Perceptron, a learning rate of 0.1 and a μ value of 5.0 was selected after cross-validation. 2431 mistakes were made during the training process. The accuracy on the training set was 0.8458, and was 0.8413 on the testing set.

3. In this case, the data was shuffled for the case of multiple epochs before each subsequent epoch.

   For the Perceptron, a learning rate of 0.6 and 5 epochs was selected after cross-validation. 6600 mistakes were made during the training process. The accuracy on the training set was 0.7957, and was 0.7984 on the testing set.

   For the Margin Perceptron, a learning rate of 0.2, a μ value of 3.0 and 3 epochs was selected after cross-validation. 5903 mistakes were made during the training process. The accuracy on the training set was 0.8388, and was 0.8343 on the testing set.

   In both cases, the hyper-parameters selected when running over multiple epochs resulted in reduced accuracy over both training and testing sets. This is possible because the samples are not linearly separated and the separating hyperplane results in classification errors. Additional updates to weights will not increase accuracy and the accuracy will tend to be more or less the same. The hyperplane may shift a little with each update, but does not converge beyond a given point.

4. When the Aggressive Margin Perceptron was trained for 1 epoch, cross-validation resulted in a μ of 1.0. 2584 mistakes were made during the training process. The accuracy on the training set was 0.7919, and was 0.7904 on the testing set.

   When the number of epochs were also selected by cross-validation, the Aggressive Margin Perceptron used a μ of 2.0 and 3 epochs were selected. 7795 mistakes were made during the training process. The accuracy on the training set was 0.8240, and was 0.8269 on the testing set. The data was shuffled in this case of multiple epochs. The shuffling was done before each subsequent epoch.