

## Problem Set 4

Due via turn-in on Friday, March 17, 2006

**Question 1.** [15 points]

What are the VC-dimensions of the following hypothesis spaces?

**part a.** The set of axis-aligned rectangles in the plane.

**part b.** The set of circles in the plane.

**part c.** On the line: the set of functions  $\Theta(ax^2 + bx + c)$ , for all  $a$ ,  $b$ , and  $c$ .<sup>1</sup>

**part d.** On the line: the set of functions  $\Theta(\sin(ax + b))$ , for all  $a$  and  $b$ .

**Question 2.** [15 points]

What happens when the perceptron learning rule is used on the dataset below?

$x_1$	$x_2$	$y$
-1	-1	+1
-1	+1	-1
+1	-1	-1
+1	+1	+1

Show the first few steps (at least the first 8) of the algorithm (how the weight vector is updated) and then describe qualitatively how things continue (for example, does it converge?). I would recommend starting with the weight vector equal to  $[1 \ 0]^T$  and set  $\eta$  (the update step size) relatively small (certainly less than 1, say 0.5 or 0.1).

---

<sup>1</sup>Remember that the function  $\Theta$  returns the sign of its argument: 1 if the input is non-negative and  $-1$  if the input is negative.

**Question 3.** [15 points]

Scanning for the best decision split point can take a lot of time when the data is continuous. A naïve approach might look like this

1. foreach dimension,  $d$ :
  - (a) sort dataset according to its value in dimension  $d$
  - (b) foreach data point index,  $i$ :
    - i. initialize histogram1 and histogram2 to be empty (zeros)
    - ii. foreach data point index,  $j < i$ :
      - A. record data point  $j$  in histogram1
    - iii. foreach data point index,  $j \geq i$ :
      - A. record data point  $j$  in histogram2;
    - iv. calculate the entropy of histogram1
    - v. calculate the entropy of histogram2
    - vi. if the weighted sum of these two entropies (weighted by the number of examples less than  $i$  compared to the number greater than or equal to  $i$ ) is smaller than any seen thus far,
      - A. record this weighted entropy
      - B. pick  $d$  as the best feature on which to split
      - C. pick the value halfway between point  $i - 1$  and point  $i$  as the best split point

Unfortunately, this algorithm has two problems. First, it will try to split the data between two points that actually have the same value in the  $d$  dimension. This isn't possible. Second, it takes  $O(DN^2)$  time, where  $D$  is the number of dimensions in the dataset and  $N$  is the number of samples in the dataset. The calculation of the entropy of the points to either side of  $i$  requires  $O(N)$  time. This is inside a loop that has  $O(N)$  iterations and that is inside a loop that has  $O(D)$  iterations.

**part a.** Fix this algorithm so that it works even if multiple data points have the same value along a particular dimension.

**part b.** Fix this algorithm so that it doesn't take  $O(DN^2)$  time, but rather takes  $O(DN \log N)$  time.

Write down the resulting algorithm (with both fixes) in pseudo-code. Use this algorithm for the next question.

**Question 4.** [35 points]

At <http://www.cs.ucr.edu/~cshelton/courses/cs170-w2006/> you will find a .tgz file with starter code and two datasets. Your task is to write a program that will build a decision tree, prune that decision tree (using the “reduced error pruning” method described in class), and calculate the tree’s accuracy on a separate testing set. The code supplied implements a dataset class (that can read in the datasets supplied and perform simple, but useful, operations on that data). There is also a simple routine (in the drawing.cpp and drawing.h files) that can help you visualize the data from the second dataset. Both datasets have been split into training (extension .tra) and testing (extension .tes) sets. You should build and prune your decision tree based on the training data and then report the resulting tree’s accuracy on the testing set.

The first dataset is the “iris” dataset. It is a classic and has been used since the 1930s. The goal is to build a function that will classify irises into one of three types: Setosa, Versicolor, or Virginica. The datasets consist of a series of flowers (each labeled according to its type, or class) and each one has four measurements (in cm): sepal length, sepal width, petal length, and petal width.

The second dataset is the “NIST digit” dataset (optdigits). It consists of many images of digits, optically scanned from the zipcodes of US mail envelopes. The goal is to build a function that can classify such an image into the correct digit (0, 1, . . . , 9). Therefore, there are 10 classes for this dataset. Each image has 64 pixels (it is an 8-by-8 image). Each pixel’s value is one feature.

The *exact same* algorithm that you use on one dataset should also work on the other dataset (or any other third dataset). Do not build code that relies on the particulars of these datasets. You may assume that all features are real-valued, however. Here are a few hints that might help you figure out if your algorithm is working correctly.

- Using 20% of the training data for pruning (leaving 80% for building the decision tree) seems to work pretty well.
- Your algorithm achieve about 95% accuracy on the iris testing dataset.
- Your algorithm achieve about 85% accuracy on the digits dataset.
- Before pruning, your decision tree should be 100% accurate on the training data used to build it. (No big surprise there, but it is a useful test.)
- Your decision tree should be more accurate on the pruning set after pruning than before pruning.

Of course, there are incorrect algorithms that pass all of those tests too, but certainly if your algorithm doesn’t meet the criteria above, it isn’t working correctly.