CS 540 (Shavlik) HW 4 –ANNs and SVMs

Assigned: 11/5/15
Due: 11/19/15 at 11:55pm (not accepted after 11:55pm on 11/25/14)
Points: 100

Be sure to show your work for all of the questions.

Problem 1 – Perceptrons (20 points)

Consider the following examples:

ex1: F1 = 1 F2 = 0 F3 = 1 category = 1
ex2: F1 = 0 F2 = 1 F3 = 1 category = 0
ex3: F1 = 1 F2 = 1 F3 = 0 category = 1

Create a perceptron (i.e., an artificial neural network with no hidden units) for this task. Do not forget about the input unit whose value is always ‘-1’ and whose weight represents the threshold for the output unit. Initially set all the weights to 0.1, then show the resulting network after each of the following scenarios. Assume that if the weighted input equals or exceeds the threshold the perceptron outputs 1 otherwise it outputs 0 (see top of page 724 of the textbook). Use the perceptron learning rule (equation 18.7 of the textbook) with $\alpha = 0.1$.

i. training on ex1

ii. training on ex1 followed by training on ex2

iii. training on ex1 followed by training on ex2 followed by training on ex3

Each of the above parts picks up where the last one left off so you only need to show your work for the final step in each part, but do show THREE network states, that is, one for each of the above three questions.

iv. What output does your trained perceptron produce after Part iii above for this example?

exNew: F1 = 1 F2 = 0 F3 = 0 category = ?
Problem 2 – Hidden Units (20 points)

i. Using the dataset in Problem 1, draw a neural network with two hidden units. Connect every input unit to every hidden unit and every hidden unit to the output unit. The only connection from the input layer to the output unit should be from the ‘-1’ input unit. Initially set all weights and thresholds to 0.1. As done in Problem 1, set $\alpha = 0.1$.

The activation function for the hidden units for this question should be the Rectified Linear Unit (ReLU), where $x$ is the weighted sum and includes the ‘-1’ unit’s contribution. The activation function for the output unit should be ‘purely’ linear, i.e. $f(x) = x$.

ii. Show, in your answer to Part i, the activations of the hidden and output units for ex1 from Problem 1. Show your work below the network.

iii. Apply the back-propagation algorithm (Figure 18.24 of the text) to your answer from Part ii. (However, rather than assigning random weights, assign all weights to 0.1 to simplify grading.) Draw the resulting neural network and show your work above it (i.e., re-draw your answer to Part i, but use the new weights). Note: the line “for 1 = L-1 to 1 do” in Fig 18.24 should be “for 1 = L-1 to 2 do” since Layer 1 contains the input units.

The derivative of the ReLU ($g'(in)$ in Figure 18.24) is:

$$0 \text{ if } in \leq 0 \text{ and } 1 \text{ otherwise.}$$

The derivative for the linear output unit is 1.

Problem 3 – Weight Space (20 points)

Assume we have a perceptron with one input unit and one output unit, where the output unit’s activation function is the sigmoid, i.e. $output = 1 / (1 + e^x)$ where $x$ is the weighted sum and includes the ‘-1’ unit’s contribution. For simplicity, in this problem we will fix the threshold to be 1.5. Hence, we have only one free parameter, the weight between the input unit and the output unit.

Also assume we have this dataset:

- ex1: F1 = 1, category = 1
- ex2: F1 = 2, category = 0
- ex3: F1 = 3, category = 0

We will define the error of the network on $ex_i$, to be the square of the difference between the category of $ex_i$ and the predicted output of the network for that example.

Draw a graph where the $x$-axis is the weight (not the weighted sum) and the $y$-axis is the total error over the three examples above. Create this graph by letting the weight be one of the values in this set and then ‘connect the dots’ to create a curve: $\{-4, -2, -1, 0, 1, 2, 4\}$.

Hint: I recommend writing some Java code to compute the errors (but no need to turn in the code).
Problem 4 – Support Vector Machines (20 points)

Consider the following new features (F1 through F3), where ex1 through ex3 are from Prob. 1:

\[ F1(\text{ex}_i) = \text{similarity between ex1 and ex}_i \text{ features (i.e., number of bits in common)} \]

\[ F2(\text{ex}_i) = \text{similarity between ex2 and ex}_i \]

\[ F3(\text{ex}_i) = \text{similarity between ex3 and ex}_i \]

i. Convert Problem 1’s dataset into this representation, i.e. a table with 3 rows and 4 columns, where the 4th column is the output category (which is not used in the above similarity functions and hence is unchanged). Write out the dataset in a manner similar to that used at the start of Problem 1, but instead using F1(\text{ex}_i), F2(\text{ex}_i), and F3(\text{ex}_i).

ii. Repeat Problem 1i through Problem 1iii, but this time use the weight-update rule below (called “weight decay”), with \(a = 0.1\) and \(\lambda = 0.2\) (i.e., draw THREE network states). Do not forget about the ‘dummy’ input unit whose weight represents the output unit’s threshold and again initialize all weights to 0.1.

\[ w_i \leftarrow w_i + \left[ a \times (\text{category(ex)} - \text{perceptronOutput(ex)}) \times \text{feature}_i(\text{ex}) \right] - a \times \lambda \times w_i \]

This is essentially what a Support Vector Machine (SVM) does, though here we are using gradient descent rather than linear or quadratic programming to find good weights.

iii. Repeat Problem 1iv using the final network state from Problem 4ii.

Problem 5 – Short Answers (20 points)

Show and explain how each of the following algorithms might partition feature space:

i. Decision trees

ii. k-NN (using k=1 is fine)

iii. Naïve Bayes

iv. Perceptrons

v. SVMs using a non-linear kernel