Homework Assignment 5

Due date: Tuesday, November 16 (in class)

In this assignment we review some ideas and methods from class and experiment with the Perceptron algorithm.

1 Review of Methods

1. You are given the following data for the FOIL learning algorithm for learning a property “nice” of nodes in a graph.

   positive examples: nice(2), nice(3), nice(5)

   negative examples: nice(4), nice(9)

   background knowledge:
   red(1), red(2), green(3), green(4), blue(5), red(6),
   blue(7), green(8), green(9)
   edge(1,2), edge(2,3), edge(2,4), edge(2,5),
   edge(3,6), edge(4,7), edge(5,8), edge(8,3), edge(7,9)

   FOIL will be used to learn a set of rules defining nice(X). One such potential concept is “a node from which one can reach a red node”; however, in this question, we are not concerned with the actual concept.

   For this question consider the initial rule with an empty right hand side

   nice(X) <--

   and consider refining the rule by adding red(X) or edge(X,Y) or edge(Y,X). For each of these 3 options, calculate the FOIL information gain (make sure to show and explain your work), and then explain which literal will be preferred.

2. Professor Cubic proposed to use neural networks to approximate real valued functions by using a concrete polynomial $\sigma(s) = 3s^2 - 5s + 2$ as the activation function. In other words a node with inputs $\vec{x} = (x_1, \ldots, x_n)$ and corresponding weights $\vec{w} = (w_1, \ldots, w_n)$ will output $o = \sigma(\vec{w} \cdot \vec{x}) = \sigma(\sum w_i x_i)$.

   Develop a gradient descent algorithm for a single node of this type. In particular, given a training set $(x^1, y^1), (x^2, y^2), (x^3, y^3), \ldots$ where superscripts index examples and their labels, use the error function $E = \frac{1}{2} \sum k [y^k - \sigma(\vec{w} \cdot x^k)]^2$ to derive the gradient descent update for $\vec{w}$.

3. Consider learning a single neural unit with sigmoid output using a “regularized” error function $E = \frac{1}{2} \sum_k [y^k - \sigma(\vec{w} \cdot x^k)]^2 + \frac{1}{2} \gamma \| \vec{w} \|^2$ where $\| \vec{w} \|^2 = \sum w_i^2$ and $\sigma(a) = \frac{1}{1+e^a}$. The penalty term on $\vec{w}$ gives preference to weight vectors without extreme weights and is thought to help avoid overfitting.

   Derive the gradient descent update formula for weights. Show that the update is equivalent to a “weight decay” strategy (see problem 4.10 in textbook).
2 Experiments with Perceptron

In this part of the assignment we extend the code from assignment 4 to run experiments with the Perceptron algorithm and the Perceptron with margin variant.

2.1 The Algorithm

The algorithm (with parameters $\eta, \tau, I$) is as follows where labels $(y^i)$ are in $\{-1, +1\}$ and the sign function gives a value in $\{-1, +1\}$:

1. Initialize $w_k = 0$ for all $k$.
2. Repeat for $I$ Iterations
   (a) For each example $(\vec{x}^i, y^i)$ in training set do:
      • (Classify): $O = \text{sign}(\vec{w} \cdot \vec{x}^i)$.
      • (Update): conditions for updates are different in the two algorithms
         Perceptron: if $O \neq y^i$:
         Perceptron with Margin: if $y^i(\vec{w} \cdot \vec{x}^i) < \tau$
         For all $k = 1 \ldots n$, $w_k = w_k + \eta y^i x^i_k$
3. Output the last weight vector $\vec{w}$ as the final hypothesis.

Thus, although the algorithm is on-line we can treat it as a batch algorithm using the final hypothesis as its output.

2.2 Experiments

Implement this algorithm as a function in your code, and integrate it with your code to evaluate algorithms using stratified $k$-fold cross validation. Your implementation should add a feature with constant value 1 to all examples to account for the threshold (this should be done inside the algorithm so that the dataset files would not require changing).

Then evaluate Perceptron and Perceptron with margin on the datasets heart-statlog.arff, sonar.arff, and sonar-withmanyfeatures.arff from the previous assignment. (We avoid glass because it has more than two classes and your implementation does not handle this case.) For parameters use $\eta = 0.1$ and $\tau = 0.1 \times V$ where $V = \frac{1}{m} \sum_{i=1}^{m} \sqrt{||x^i||^2}$ is the average norm of training examples that needs to be calculated before training starts.

For results plot the performance (accuracy and standard deviation) as a function of iterations from 1 to 100. Comment on any trends or lack thereof in the results. Additionally compare the results to the performance of $k$-nearest neighbor from the previous assignment.

3 What to Submit

4 What You Need to Submit

Please submit hard-copies only. Your submission should include:

1. Solutions for the questions in part 1.
2. Source code of all the programs used to run your experiments. If it’s not obvious please give instructions on how to run your code. Please also make sure that your code is well documented and written with good style.
3. A short report on the experiments you ran and their results. Please include all numerical results and plots as requested above and please discuss any trends or lack thereof in the results.

The experimental portion of this assignment is graded based on the quality of the code, quality of documentation, and the quality of the report and observations.