Assignment 4, Neural Networks and Application

CSC736 Machine Learning
Max Score: 100

Objectives

The purpose of this assignment is to familiarize ourselves with neural networks and implement a simple three layer neural network (input layer, hidden layer, and output layer) with the Backpropagation learning algorithm on a handwritten digits recognition problem. This problem is important in applications such as automatic zip code reading on letters, which is in current use by the U.S. Postal Service. As you will see below the assignment asks you often to practice your own judgment and curiosity in the experiments. Feel free to use your imagination and tell me about what you thought.

The Data

The data file, called optdigits-3.tra, is an ASCII file containing 1535 examples, one per line. Each example is a comma-separated (without any white space) list of 65 integer values, the first 64 specifying the input and the last value specifying the digit which is the desired output. The input values are integers in the range [0..16]. You should first normalize the input values by converting them to reals in the range [0..1] by dividing every value by 16.0. This is useful because the derivative of the sigmoid function is often very close to 0, which can cause the network to converge very slowly to a good set of weights.

You'll have to convert the desired output digit to a target output vector for the four output units (0, 1, 2, 3). For example, if the digit is a "3" then create the target vector [0.1 0.1 0.1 0.9]. Using this set of teacher output values is preferred because the sigmoid function cannot produce the exact output values of 0 and 1 using finite weights, and so the weight values may get very, very large causing overflow problems.

Training

You are to implement a validation set to evaluate the performance of your network during the training process. To do this, you should divide the input file into two parts, each containing 80% and 20% data samples. For the training process, you will use the 80% of the training data as the training set and the remaining 20% will be the validation set. (Note: You should

experiment with the number of epochs to train based on the MSE curve (see below) your results produce.) After every 10 epochs compute the mean squared error (MSE) on the training set and validation set as follows:

$$MSE = \frac{1}{m}\sum_{j=1}^{m}(\times \sum_{i=1}^{4}(T_{ij} - O_{ij})^2)$$

where m is the number of examples in the training/validation set, T_{ij} is the target output value (either 0.1 or 0.9) of the ith output unit for the jth training/validation example, and O_{ij} is the actual real-valued output of the ith output unit for the jth training/validation example. You should compute this MSE value of your training examples after every 10 epochs by stopping backpropogation and running through all of the training examples (not validation examples) to compute this error value. You should not compute this "on the fly" after each training example is used to update the weights because then the error for each example would be based on a different network (i.e., set of weights). You should not use your validation examples to tune your weights in the network.

Plot these MSE values as a training curve. The x-axis will show the epoch number and the y-axis will show the MSE value. You can create this plot manually or by entering the data into any plotting program.

Testing

Test your network using the examples in the test set. Report the percentage correct classification on the test set. Define the output digit computed by the network as the corresponding output unit with maximum activation (i.e., output) value.

Experiments with Varying Numbers of Hidden Units

Repeat the training and testing steps given above after varying the number of hidden units in your network. Use values of 5 and 50. Show the training plot and the validation plot using the same training and testing sets. Comment on how performance changes with the number of hidden units, both in terms of the percentage correctly classified on the training set and the test set, and also in terms of the number of iterations that seem necessary for the network to "converge." Specify what you consider to be the "best" set of parameters (alpha, number of hidden units, number of iterations) for this problem based on your experiments.