CSE 40625 — Machine Learning
Assignment 5 (10 points)
Due Date: April. 13, 2017 (Sakai Drop Box)

Multilayer Neural Network

Overview

The purpose of this assignment is for you to implement a multilayer neural network, a powerful supervised machine learning inspired by the way the brain works. Multilayer neural networks are systems of interconnected "neurons" or units that can compute values from inputs by feeding information through the network. The model makes predictions by chaining together layers of neural units, with the output of neurons at earlier layers used as the input to neurons at later ones. Materials for the assignment, including the dataset, expected output, and template code can be found here on GitHub.

You may use Python libraries for handling data preprocessing and visualization, including but not limited to NumPy, SciPy, pandas, and Matplotlib, but *you may NOT use any Python libraries that employ machine learning models, including but not limited to scikit-learn, StatsModels, TensorFlow, or Orange*. Your solution to the assignment should be individually submitted.

Dataset

You will use the "digits" dataset on handwritten digit classification with all 10 classes (labeled from 0 to 9) for this assignment. The data is provided in comma-separated (CSV) file format. For all rows, the last column designates the class (y) and the remaining columns designate features (X). The first row consists of the feature and class names.

Procedure

Use the above digits dataset as input to a multilayer neural network model with two hidden layers of 100 units each. Initialize the weights at each layer with random samples using normalized initialization. Initialize the activations as input x. For each batch of inputs x, feed the activations of the batch forward through the network. Add a fixed bias term of positive one to the hidden and final activations. At each layer, apply the hyperbolic tangent function to compute the activations for the next layer. Using squared loss, compute the sensitivities of the final layer as twice the difference of the final activations and the target values. Propagate the sensitivities backwards, using the sensitivities at the current layer to compute the sensitivities at the previous layer. Update the weights at each layer using the minimum gradient of the error function with a learning rate of 0.01 for at most 500 iterations. Predict each target value based on the unit in the final layer with the highest activation output.

Output

Your code should output the accuracy of the prediction results every 50 iterations while fitting the model, a blank line, and the confusion matrix of the final output results.

Example output:

```
0 0.676
50 0.972
100 0.982
150 0.988
200 0.990
250 0.992
300 0.994
350 0.994
400 0.995
450 0.995
```

Predicted	0	1	2	3	4	5	6	7	8	9
Actual										
0	554	0	0	0	0	0	0	0	0	0
1	0	569	0	0	0	0	0	1	1	0
2	0	0	556	0	0	0	0	0	1	0
3	0	0	0	570	0	1	0	0	0	1
4	0	0	0	0	566	0	1	0	0	1
5	0	0	0	1	0	554	0	0	0	3
6	0	1	1	0	1	0	555	0	0	0
7	0	0	0	1	0	1	0	564	0	0
8	0	6	0	0	0	0	0	0	548	0
9	0	1	0	0	1	1	0	0	1	558

Submission

Please submit a Python executable (multilayer_perceptron.py) file of your code to your Sakai Drop Box. *Should you run into any problems, please feel free to email or meet with the instructor.*