CSE 40625 — Machine Learning
Assignment 4 (10 points)
Due Date: March. 30, 2017 (Sakai Drop Box)

Single-layer Neural Network

Overview
The purpose of this assignment is for you to implement a single-layer neural network, a supervised machine learning inspired by the way the brain works. 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. A single-layer neural network has a single layer of hidden neurons. 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 single-layer neural network model with 100 hidden units. Initialize the weights at each layer with random samples from a uniform distribution over [0, 1). 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 0.625
50 0.962
100 0.973
150 0.978
200 0.981
250 0.983
300 0.984
350 0.984
400 0.985
450 0.986

Predicted   0   1   2   3   4   5   6   7   8   9
Actual
0         549   1   0   0   1   1   1   0   1   0
1           0 560   0   0   0   0   0   0  10   1
2           0   0 556   1   0   0   0   0   0   0
3           0   0   0 562   0   3   0   0   3   4
4           0   0   0   0 562   0   4   1   0   1
5           0   0   0   1   0 547   1   0   2   7
6           0   1   0   0   1   0 555   0   1   0
7           0   0   0   0   1   0   0 563   1   1
8           0  10   0   1   0   0   0   0 541   2
9           0   3   0   1   0   0   0   0   8 550
```

Submission
Please submit a Python executable (singlelayer_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.*