DAT405 Introduction to Data Science and AI, 2019 –2020,

Assignment 7:

Read Section "Implementing our network to classify digits" in Chapter 1 of the online book:

http://neuralnetworksanddeeplearning.com/chap1.html

Download and prepare the code provided in this section. Note: if you are using Python 3 and above, use the following link for the code:

https://github.com/MichalDanielDobrzanski/DeepLearningPython35

Perform the following tasks and write a report about it. For each experiment, write a short text (few lines) about your observation. Include the codes that you write (including only the modified parts in the given code) in your report. Parts 1 and 2 gives you grade 3. Proceeding to parts 3 and 4 respectively improve your grade to 4 and 5.

1)Using the matplotlib library, add few lines to the program to

A) also calculate the training accuracy (the number of correctly classified training samples out of 50000) after each epoch.

B) Plot the training and testing accuracies vs the epoch as two curves in the same plot.

2)Use your program to perform the following experiments.

a) Plot the training and testing curves for the case of a single hidden layer with 30 units and step size 3 with 30 epochs.

b) Change the number epochs to 10 and the number of hidden layers to 100. Try different step sizes from 3 to 15. Repeat each step size 3 times. Report the testing result at the last epoch of each trial. For this learning and learning rate 3, make two separate plots of performance with 30 epochs.

c) Fix the number of iterations to 10. Create a chart of testing performance for different number of hidden layers (one hidden layer and repeat 3 times) with the best learning rate by repeating part 2 above. Report the best size and best learning rate with the plot for the performance with 30 epochs.

3) Experiment with noise:

a) Add few lines to the network.Network.SGD (after the line "*n = len(training_data)*") to add a centered i.i.d Gaussian noise with standard deviation (std) 1 to each training data point. Use

command "*np.random.randn()*" to create noise and note that the *training_data* variable is a list of tuples [x,y] of data and labels.

b) With a single hidden layer with 30 units and step size 3 with 30 epochs, report the performance for different noise levels (std) from 0 to 2 (3 realizations for each value).

4) Implement l_2 norm regularization: We want to change the risk minimization framework to a regularized one:

$$\min \frac{1}{2N} \sum_{i=1}^{N} \left(y_i - f_{W,b}(x_i) \right)^2 + \frac{0.001}{2} \|W\|_2^2$$

Where N=50000 is the number of training data points, W, b are respectively the vectors of all weights and biases in the network and $\|W\|_2^2$ is the squared l_2 norm of the weights (sum of squares of weights)

a) Calculate the gradient of $\frac{0.001}{2} \|W\|_2^2$ by hand, assuming that $W = (w_1, w_2, \dots, w_M)$.
b) Make necessary changes in the function "*network.Network.update_mini_batch"* to include this gradient (note that the negative of gradient is added and set the learning rate of this term to 1 (not eta)).
c) With a single hidden layer with 30 units, step size 3, noise std 1 and 30 epochs, report the performance by changing the regularization parameter (0.001) from 0 to 0.002 (repeat each value three times).