Homework 1

CENG506 - Deep Learning

March 16, 2018

How to submit

You will submit the homework via CMS. There will be a homework submission link, clicking to which will lead you a file browser that you can choose your file. Please zip your files (even if there is only one file) before submission and name the file as yourstudentno-hw1.zip. Submit the homework until 11 April 2018 23:59.

Files included in this exercise

NeuralNet.py is currently almost empty. It just contains essential parts that should exist in a neural network implementation. You are going to write the necessary code and submit this file. Please do not change how init, predict and train functions are called from main.py. Ideally your NeuralNet.py should be able to be called from original main.py.

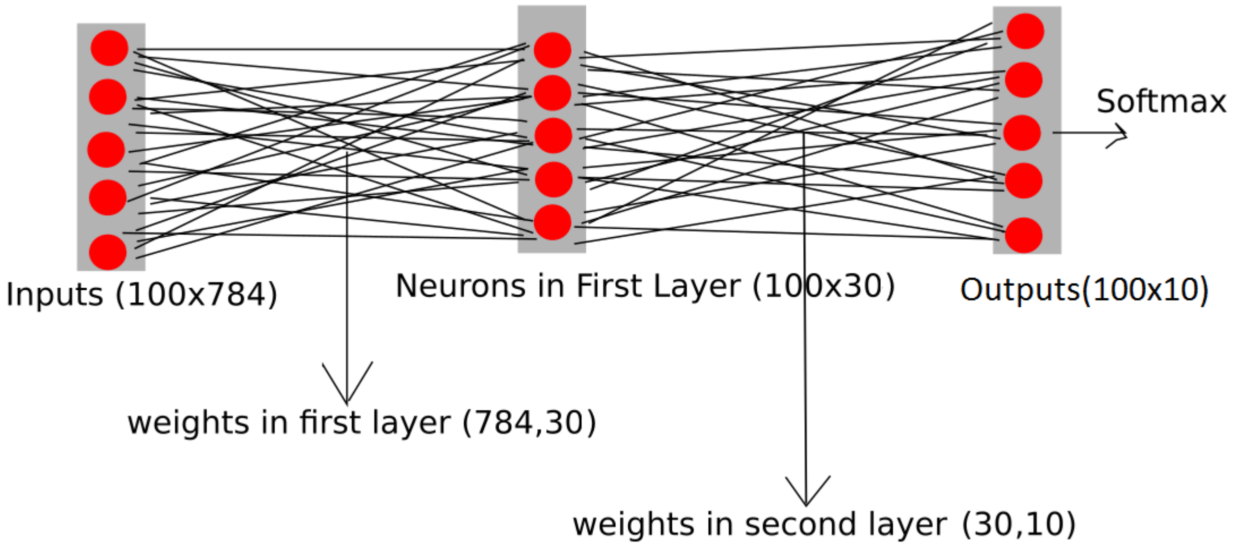
main.py contains lines to read the data, to assign some parameters, to call NeuralNet.py for training and testing. Finally it plots the recorded errors of each epoch. The hyperparameters already initialized in main.py (like hidden layer size, batch size, learning rate) are the suggested ones. Ideally you are not supposed to change any parameters in main.py. But if you have to change any parameters for better optimization/learning then you are allowed to change. If you changed any parameters in main.py, please also submit main.py with NeuralNet.py

Dataset

Network implemented by you, is going to be trained and tested on MNIST handwritten dataset. To do that you need to install this dataset, namely python-mnist 0.5. Please see the documentation: https://pypi.python.org/pypi/python-mnist

Network architecture

In this homework, you are going to develop a fully connected neural network with one hidden layer. You are allowed to change hidden layer size. If you keep it as 30, your architecture will look like the figure given below.


Softmax
Inputs (100x784)
Neurons in First Layer (100x30)
Outputs(100x10)
weights in first layer (784,30)
weights in second layer (30,10)

In the figure above, layers are denoted as matrices of (no of neurons x 100). Here 100 represents your batch size. With a clever implementation you can run your forward pass and backpropagation once for each batch. Otherwise, you need to write a for loop that turns as many as batch size.

Forward pass, Loss calculation and Backpropagation

You will perform the forward pass, computing the class scores for an input. Next, taking the loss as the sum of the classification loss (use softmax) and $L2$ regularization loss, you will calculate gradients for all parameters.

Weights will be updated at the end of each batch using simple gradient descent.

Weight initialization

The weights will be initialized randomly, by sampling from a Gaussian distribution of zero mean. As we have seen in the lecture, calibrate the variance of weights at each layer by multiplying with $1/\sqrt{n}$ where n is the number of input to a neuron.

Activation function, Data normalization

The ReLU will be used as the activation function. No data normalization is required. No batch normalization is required.