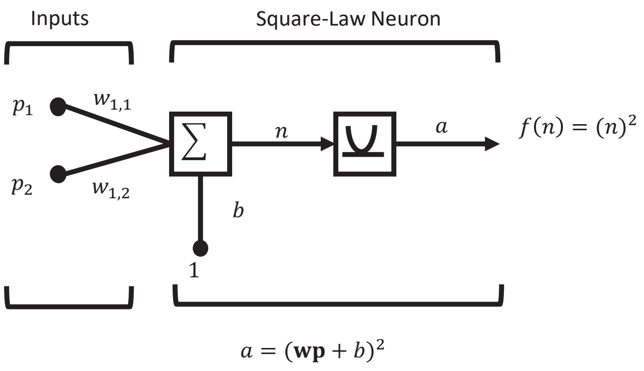
Machine Learning II
Exam #1
A. Jafari
Due: *October-29-2018*

Q1:Stepest Decent

Consider the following network



i. The following inputs and targets are given.

$$\mathbf{p}_1 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, t_1 = 0 \text{ and } \mathbf{p}_2 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, t_2 = 2$$

the initial weights and bias are

$$\mathbf{W}(0) = \begin{bmatrix} -1 & 1 \end{bmatrix}, b(0) = 0$$

Perform ONE iteration of the steepest descent algorithm, using a learning rate of one. Show the forward passes through network, the backward passes to compute the sensitivities and the weight and bias updates. Show every step clearly and completely.

Q2:Training MLP networks with PyTorch on CIFAR10 dataset

Notes

The objective of this exam is to become familiar with PyTorch for training deep networks on large datasets. You will be provided with a sample program that loads in the CIFAR10 data set and sets up a simple multilayer network to be trained on it. (See `https://www.cs.toronto.edu/~kriz/cifar.html` for a description of the data set and a list of various efforts to train on it.) The instructions below provide a rough framework of what you should do for the exam. You should experiment with using PyTorch on this large dataset. The project is open-ended. The goal is to learn as much as you can about using deep multilayer networks, and to communicate what you have learned in your project report.

Very Important Note: Your workflow for the exam should be as follows:

- **First:** Create a folder and title it as your name followed by an underscore and exam1. For example, I would create a folder and save it as `amirjafari_exam1`. **All** of your work for the exam should be saved in this folder.
- **Second:** Since part of the goal is to communicate what you have learned about multilayer networks by experimenting in PyTorch, you will write a report to document your learning process. Type your answers for each question into a word document. Be sure to include program listings, plots, command line output, etc. if necessary. Save your word document as a single PDF file and name it `exam1_report`.
- **Third:** Create a new folder, name it `report`, and move the PDF file of your report into it.
- **Fourth:** Create another new folder and name it `code`. All the code you write for the exam should be placed in this folder. You should have different scripts for each question and you should give them informative names such as `Q1_sol.py`, `Q2_sol.py`, `Q4_3layer_sol.py`, etc.
- **Fifth:** Zip the main folder and upload it to Blackboard. It should be titled similarly to the main folder (i.e., `amirjafari_exam1.zip`.
- Failing to do these steps result in a reduced grade.

* We will be using the Python implementation of Torch, PyTorch. If you are using GCP or AWS, PyTorch is already installed.

* Download the `CIFAR10.py` file from GitHub (See `https://github.com/amir-jafari/Deep-Learning/blob/master/Pytorch_/Mini_Project/CIFAR10.py`). It is based on the following website `https://github.com/yunjey/pytorch-tutorial/blob/master/tutorials/02-intermediate/convolutional_neural_network/main.py`, which you may want to review before proceeding.

Questions

1. Run the program in Pycharm and investigate the errors. I have manually added some bugs into the sample program. The first step is to get an understanding of the code and fix the errors. Once you have fixed the errors, investigate and verify its performance.

2. Modify the program to run on the GPU.

3. Find out, if program is set up to perform stochastic gradient descent or mini-batch gradient descent. Explain your answer.

4. Experiment with different numbers of layers and different numbers of neurons (Deep Network vs Shallow Network). While increasing the number of layers in the network, make sure the total number of weights and biases in the network are roughly the same as the original. Describe how the performance changes as the number of layers increases – both in terms of training time and performance.

5. Try several training function from the list on this page: `http://pytorch.org/docs/master/optim.html`. Compare the performance with gradient descent. Hint: Use the same seed number.

6. Try different transfer functions and do a comparison study between the transfer functions. Explain your results.

7. Write a script to save all gradients with respect to inputs. Save it as CSV file. Calculate the standard deviation of the gradients for each trail and sort the highest 10 std of the input.

8. Try to use dropout nodes and check if the performance is better or not. Explain the advantages/disadvantages of using dropout nodes.

9. Make a chart and or table to compare the difference in performance between batch, mini-batch, and stochastic gradient descent. Calculate the time it takes to train the network for each method by using the system (sys) clock command in Python.

10. Plot the confusion matrix (as table or a figure) for your predictions and explain each element of the confusion matrix. Additionally, Calculate the accuracy and misclassification rate.

11. Write a script or function to visualize the network response (output) for an arbitrary input and compare it with the target associated with that input.

12. **Bonus:** Plot the ROC curve and explain the results. Calculate AUC and explain the results. Note: You need to explain ROC and AUC and the use of these techniques in pattern recognition. Hint: There is more than one ROC curve in a multi-class problem.

13. **Bonus:** You can use torchvison capability of PyTorch to visualize the network. Try using tensorboard and look into weight and biases of network and find out how this information can be useful to improve your results.

14. **Bonus:** The 3 highest performance on the test set (f1score and accuracy) will get a bonus point.