CIS 419/519 Applied Machine Learning
Assignment 4

Due: April 9, 2019 11:59pm

Instructions

Read all instructions in this section thoroughly.

Collaboration: Make certain that you understand the collaboration policy described on the course website.

You should feel free to talk to other members of the class in doing the homework. I am more concerned that you learn how to solve the problem than that you demonstrate that you solved it entirely on your own. You may *discuss* the homework to understand the problems and the mathematics behind the various learning algorithms, but you must **write down your solutions yourself**. In particular, **you are not allowed to share problem solutions or your code with any other students.**

We take plagiarism and cheating very seriously, and will be using automatic checking software to detect academic dishonesty, so please don't do it.

You are also prohibited from posting any part of your solution to the internet, even after the course is complete. Similarly, please don't post this PDF file or the homework skeleton code to the internet.

Formatting: This assignment consists of two parts: a problem set and program exercises.

For the problem set, you must write up your solutions electronically and submit it as a single PDF document. We will not accept handwritten or paper copies of the homework. Your problem set solutions must use proper mathematical formatting.

All solutions must be typeset using LaTeX; handwritten solutions of any part of the assignment (including figures or drawings) are not allowed.

Your solutions to the programming exercises must be implemented in python, following the precise instructions included in Part 2. Portions of the programming exercise will be graded automatically, so it is imperative that your code and output follows the specified API. A few parts of the programming exercise asks you to create plots or describe results; these should be included in the same PDF document that you create for the problem set.

Homework Template and Files to Get You Started: All files needed for this assignment are available in an archive posted online at
https://www.seas.upenn.edu/~cis519/spring2019/homework/hw4/hw4_skeleton.zip.

Citing Your Sources: Any sources of help that you consult while completing this assignment (other students, textbooks, websites, etc.) ***MUST*** be noted in the your PDF file. This includes anyone you briefly discussed the homework with. If you received help from the following sources, you do not need to cite it: course instructor, course teaching assistants, course lecture notes, course textbooks or other readings.

Submitting Your Solution: You will submit this assignment through **both** Gradescope and Canvas; submission will open approximately one week before the due date.

Both the PDF with your written responses and your python code must be submitted to **both** Canvas and Gradescope. Why do we require that you submit to both sites? Gradescope has better automatic unit testing functionality, and Canvas allows us to run plagiarism detection software. Unfortunately this means that we need you to submit the assignment in two places so we can get the best of both worlds.

All of the files should be uploaded directly; there should not be any directory structure.

Assignment Version 20190326

CIS 519 ONLY Problems: Some parts of the assignment will be marked as "[CIS 519 ONLY]". Only students enrolled in CIS 519 are required to complete these problems. However, we do encourage students in CIS 419 to read through these problems, although you are not required to complete them.

All homeworks will receive a percentage grade, but CIS 519 homeworks will be graded out of a different total number of points than CIS 419 homeworks. Students in CIS 419 choosing to complete CIS 519 ONLY exercises will not receive any credit for answers to these questions (i.e., they will not count as extra credit nor will they compensate for points lost on other problems).

Files to submit:

1. **HW4_report.pdf:** Submit to both Canvas and Gradescope.
2. **HW4.ipynb:** Submit to both Canvas and Gradescope.
3. **HW4_preds.txt:** Submit to gradescope.

PART I: PROBLEM SET

Your solutions to the problems will be submitted as a single PDF document. Be certain that your problems are well-numbered and that it is clear what your answers are.

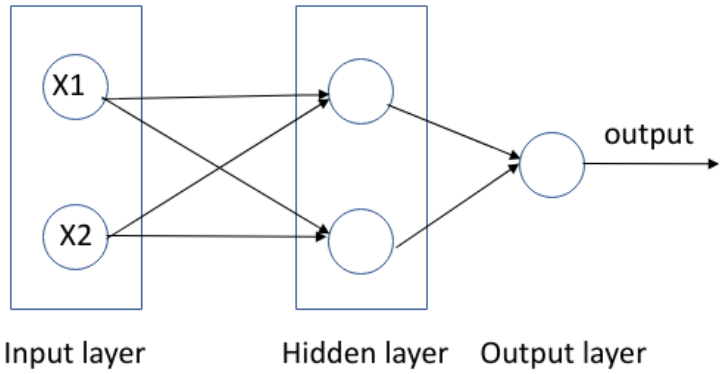
1 Thank U, Nets: Logical Functions with Neural Nets [10 pts]

For each of the logical functions below, draw the neural network that computes the function, and give a truth table showing the inputs, the value of the logical function, and the output of the neural network verifying that the neural network is correct. Show your work for the computations of the neural network's output.

(a) The NAND of two binary inputs

(b) The parity of three binary inputs

2 Raindrop, Backprop: Calculating Backprop by Hand [20 pts]



Consider the two-layer neural network shown above. You will use the sign function as the activation function in the hidden layer, but the output layer has a sigmoid activation function. The input vector is $\mathbf{x} = [5 \quad 4]^\intercal$. Compute the output gradient with respect to each of the weights. Be sure to show your work.

Use the following values for the input and weights:
Weight matrix for the first layer (between input and the hidden layer):

$$\mathbf{W}^1 = \begin{bmatrix} 0.1 & 0.2 \\ -0.4 & 0.3 \end{bmatrix}$$

Weight matrix between the hidden layer and output:

$$\mathbf{W}^2 = \begin{bmatrix} 0.1 & 0.2 \end{bmatrix}$$

$$Sign(z) = \begin{cases} 1 & \text{if } z \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

$$\nabla Sign(z) = \begin{cases} 1 & \text{if } -1 \leq z \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

Notation: W_{ij} denotes the weight between j^{th} component of the input vector (the left layer) and the i^{th} neuron of the next layer to the right. For example, to compute the input to the first neuron (start counting from the top) of the hidden layer:

$$[0.1 \quad 0.2] \times [5 \quad 4]^T = 1.3$$

.

PART II: PROGRAMMING EXERCISES

3 Where are Ü Now: Using Neural Nets for Classification [50 pts]

In this section of the assignment, you are a software engineer in a robotics start-up in Silicon-Valley! Your company is trying to revolutionize the future of personal robotics by offering assistant robots in the workplace. The robots navigate around the office and perform various task depending on their locations. As a software engineer, you are being tasked to train the robots to identify their location based on the images of the environment. There are 5 locations that need to be identified. The locations are One-person office/Barbara's office (label 0), Corridor (label 1), Two-persons office/Elin's office (label 2), Kitchen (label 3) and Printer area (label 4). After a great amount of research, you come to the conclusion that deep learning is the most versatile approach to this problem. You decide to use 1) a feed-forward neural network and 2) and convolutional neural network (CNN) in your pursuit of the best indoor scene classifier!

Dataset:
We will be using an edited version of the INDECS Database taken from the KTH Royal Institute of Technology. The database consists of several sets of pictures taken in five rooms under various illumination and weather conditions at different periods of time. Each image has a size of $64 \times 85 \times 3$ in the RGB scale. A more detailed explanation of the data set can be found here: https://www.nada.kth.se/cas/INDECS. Please have a look at the website and explore the given information about the dataset.

To make it easier for you, we've already downloaded the data set and stored the images and their labels in numpy matrices, available in files stored in the `data` folder.

Pytorch:
We will use Pytorch, one of the most popular deep learning frameworks. In this part, you will need to read and understand our Pytorch tutorial before starting to use it. After fully understanding the tutorial code, you should be able to implement the simple feed-forward networks and convolutional neural networks using Pytorch. To help debug your code and ensure that the network is learning, a useful suggestion is to plot the loss versus training epoch to check if the loss decreases during training. You can use the Pytorch tutorial as your reference, and create a new python file to implement the following tasks.

Default Parameters:
As your default parameters, you should use:

- Adagrad as the optimizer
- 0.001 as the learning rate
- 64 as the batch size
- 40 as the maximum number of epochs
- Cross-entropy loss as the loss function

Data Visualization:
Before starting with the experiments, it is important to visualize the pictures that lie in each class. Seeing the differences in each class' pictures enables you to understand the task of the classification better. Then, you can decide on the model's structure that can take advantages of such differences to perform better in the task. Load the data from the files in the `data` folder, and examine the structure of the loaded data. Visualize 6 pictures per class in the function `data_visualization`. You will have 30 pictures in total because we have 5 classes in this classification task.

Experiment 1: Baseline Feed-Forward Neural Network [10 points]
In this part, you will implement a one-hidden-layer neural network, using the architecture shown in the table below. The original image size is $64 \times 85 \times 3$, so you need to reshape the input image as a 16320×1 vector and feed it into the network. The hidden layer has 2000 neurons with ReLu activation functions. The last layer outputs a 5-element vector, representing each class. Plot the training accuracy and loss of your

feed forward neural network over 40 epochs in your report. Report your final validation accuracy and all information needed to reconstruct your experiment.

Table of the structure and parameters:

Layer	Hyperparameters
Fully Connected1	Out channels = 2000. ReLu activation functions
Fully Connected2	Out channels = 5.

Note you should complete Experiment 1 by programming the class `FeedForwardNN` in the skeleton python file provided.

Experiment 2: Baseline Convolutional Neural Network [10 points]
Implement the CNN architecture shown in the table below. Plot the training accuracy and loss of your CNN over 40 epochs in your report. Report your final validation accuracy and all information needed to reconstruct your experiment. Note you should complete Experiment 2 by programming the class `ConvolutionalNN` in the skeleton python file provided.

Table of the structure and parameters:

Layer	Hyperparameters
Covolution1	Kernel=(3×3), Out channels = 16, stride=1, padding=0. ReLu activations
Pool1	MaxPool operation, Kernel size=(2×2)
Covolution2	Kernel=(3×3), Out channels = 32, stride=1, padding=0. ReLu activations
Pool2	MaxPool operation, Kernel size=(2×2)
Fully Connected1	Out channels = 200. ReLu activations
Fully Connected2	Out channels = 5.

Experiment 3: Image Preprocessing [10 points]
In this part, you will explore how image pre-processing can play an important role in the performance of a convolutional neural network. First, instead of using the raw images, you should normalize images before training. Specifically, do the following:

Take each image and normalize pixel values in each of the RGB channel by subtracting its mean and dividing by the standard deviation. For example, if you are normalizing the red channel for an image, then for each of the red pixel values RP_i, you should compute:

$$\frac{RP_i - mean(RP_i)}{std(RP_i)}$$

Similarly normalize the blue and green channel of each image. Note you should complete Experiment 3 by programming the function `normalize_image()` in the skeleton python file provided. You will then have to train your baseline convolutional neural network from Experiment 2 using these normalized images. Plot the training accuracy, validation accuracy and training loss of your baseline convolutional neural network over 40 epochs with the raw images and normalized images on the same graph in your report. Report your final validation accuracy and all information needed to reconstruct your experiment. Describe the graph by comparing and contrasting the performance of these methods over validation accuracy, training accuracy, and loss over 40 epochs.

Experiment 4: Hyper-parameterization [20 points]
Hyper-parameter tuning is a very important procedure when training a neural network. In this part, you will change different hyper-parameters for one of the models in Experiment 1 or 2 (your choice). You can change anything you want in this experiment and that includes number of hidden layers, learning rate,

choice of optimizer, etc. The aim in this experiment is to create your most optimized neural network for this classification problem.

Implement the new hyper-parameterized neural network architecture in the `OptimizedNN` class as seen in the provided skeleton python file. Explain your decisions and reasoning for your choices and report your final validation accuracy of your model in the report. Report all information needed to reconstruct your experiment. After finalizing your choice of model hyper-parameters, train your model using 100% of the given training data. Evaluate your 100%-trained model by predicting the labels for the given testing data. Your implementation should output a txt file called `HW4_preds.txt` with only one column with one prediction per line as follows:

```
0
3
2
1
...
```

Be very careful not to shuffle the instances; the first predicted label should correspond to the first unlabeled instance in the testing data. The number of predictions should match the number of unlabeled test instances. Plot the training accuracy and loss of your hyperparameterized neural network over 40 epochs in your report. Provide a table that summarizes the different hyperparameters that you have chosen. Explain your decisions and reasoning for your choices and report your final validation accuracy of your model in the report.

4 Better Now: Debugging Neural Nets [20 pts] (CIS 519 ONLY)

In this Post-Malone-inspired section of the homework, you will be tasked with debugging neural networks in PyTorch. Please read the following instructions carefully.

In Section 4.1, you will be asked to train a vanilla feed-forward DNN on the data we have provided. You will use this as a ground-truth comparison to the remainder of the assignment.

In Section 4.2-4.3, you will begin by training the DNN and plotting the training accuracy. Please report the training and test accuracy after 10 epochs. Then, you will identify which one of the following parts of the neural network is responsible for the faulty performance: (1) the DataLoader, (2) the Network, or (3) the Optimizer. Note that in each case, only *one* of the three items will be faulty. You will then be asked to replace the faulty item and retrain the network.

The code stub associated with this portion of the assignment is `HW_4_Part4_Code.ipynb`. It is stored in a folder called `Part4`, which contains some additional files that are important to this portion of the assignment. **You will need to use Colaboratory** https://colab.research.google.com/notebooks/welcome.ipynb#recent=true **to run this code.** When you run the code, **ensure that all the contents within the folder `Part4` are kept in the same directory in Google Drive** (you can simply upload the folder to drive). Note that the code itself will not be graded, but the writeup, which depends on the output of the code, will be evaluated.

4.1 Feed-Forward Implementation (for comparison)

In this section, you will implement the class `StudentNet`, a feed-forward network with no convolutions, which you will train and compare to the faulty networks. We will be working with a dataset of handwritten characters. The dataset consists of labels, 0-25, representing characters A-Z in the alphabet. We have provided functional DataLoaders in this section for your convenience.

Using an optimizer and criterion of your own choosing, you will also implement `run_net` which takes in a network, train loader, test loader, criterion, and optimizer, and runs the neural network.

In your writeup, please provide the following:

(a) Describe the architecture of your feed-forward Network.

(b) Plot the training accuracy of the network over 10 epochs.

(c) Report the final training accuracy and test accuracy of the network.

4.2 Faulty DNN 1

(a) Plot the training accuracy of the network over 10 epochs.

(b) Report the training and test accuracies of the network after 10 epochs.

(c) Describe which unit (DataLoader, Network, Optimizer) is faulty and what the problem is. Explain in detail why this is consistent with your observations in training.

(d) Retrain the network by substituting the faulty unit with one of your own (from part 4.1), keeping the rest the same. Plot the training accuracy over 10 epochs below.

(e) Report the final training and test accuracy after 10 epochs on the fixed DNN.

4.3 Faulty DNN 2

(a) Plot the training accuracy of the network over 10 epochs.

(b) Report the training and test accuracies of the network after 10 epochs.

(c) Describe which unit (DataLoader, Network, Optimizer) is faulty and what the problem is. Explain in detail why this is consistent with your observations in training.

(d) Retrain the network by substituting the faulty unit with one of your own (from part 4.1), keeping the rest the same. Plot the training accuracy over 10 epochs below.

(e) Report the final training and test accuracy after 10 epochs on the fixed DNN.