CS 189 Introduction to Machine Learning
Spring 2022 Marvin Zhang HW5

Due 4/17/22 at 11:59pm

- Homework 5 consists entirely of coding questions.
- We prefer that you typeset your answers using LaTeX or other word processing software. If you haven't yet learned LaTeX, one of the crown jewels of computer science, now is a good time! Neatly handwritten and scanned solutions will also be accepted for the written questions.
- In all of the questions, **show your work**, not just the final answer.

Deliverables:

1. Submit a PDF of your homework ,**with an appendix listing all your code**, to the Gradescope assignment entitled "HW5 Write-Up". **Please start each question on a new page.** If there are graphs, include those graphs in the correct sections. **Do not** put them in an appendix. We need each solution to be self-contained on pages of its own. If a question only requires coding, simply answer it "See Jupyter Notebook," and make sure to assign the correct pages of code to the question.
   - In your write-up, please state with whom you worked on the homework. This should be on its own page and should be the first page that you submit.
   - In your write-up, please copy the following statement and sign your signature next to it. (Mac Preview and FoxIt PDF Reader, among others, have tools to let you sign a PDF file.) We want to make it *extra* clear so that no one inadvertently cheats. *"I certify that all solutions are entirely in my own words and that I have not looked at another student's solutions. I have given credit to all external sources I consulted."*
   - **Replicate all your code in an appendix**. Begin code for each coding question in a fresh page. Do not put code from multiple questions in the same page. When you upload this PDF on Gradescope, *make sure* that you assign the relevant pages of your code from appendix to correct questions.
2. Submit a zip file containing your code to "HW5 Code". We will use this to check your implementation for academic dishonesty

1 Administrivia (2 points)

1. Please fill out the Check-In Survey if you haven't already. Please write down the 10 digit alphanumeric code you get after completing the survey.
2. **Declare and sign the following statement:**

   "I certify that all solutions in this document are entirely my own and that I have not looked at anyone else's solution. I have given credit to all external sources I consulted."

   Signature: ______________

   While discussions are encouraged, *everything* in your solution must be your (and only your) creation. Furthermore, all external material (i.e., anything outside lectures and assigned readings, including figures and pictures) should be cited properly. We wish to remind you that consequences of academic misconduct are particularly severe

2 Principal Canine Analysis (11 points)

In this question we will perform and analyze PCA on a dataset consisting of images of dogs. Each datapoint, $\mathbf{x}$, consists of a flattened $64x64$ pixel image (i.e. $\mathbf{x} \in \mathbb{R}^{4096}$).

For this problem, you may use *matplotlib*, *numpy.**, *numpy.linalg.eig* and *numpy.linalg.svd*.

(a) (2 points) Recall that in order to perform PCA, we must first center our data. Complete `center_data()` to center the data. Compute the average dog of the dataset, center the data, and plot the average dog face.

(b) (4 points) Perform PCA on the dataset by 1) Computing the eigendecomposition of the covariance matrix, 2) Projecting the data matrix onto the k first eigenvectors of the eigenvector matrix, and 3) Reconstructing the data matrix X from the projected data matrix X_k.

Answer the following question: How similar are the reconstructed dog faces to the original dog faces? Why do we observe this?

(c) (5 points) Recall from lecture that we can compute the percent variance explained by a certain number of PCs by using the eigenvalues of the covariance matrix. We have written a function to plot the percent variance explained as a function of the number of PCs. Answer the following questions:

1) How many PCs are needed to explain 95% of the variance?

2) What is the rank of the data matrix?

3) What is the rank of the covariance matrix?

4) Given the rank of the covariance matrix, what is the upper bound on the number of non-zero eigenvalues?

5) Explain why your answer to question 1) seems reasonable given the number of non-zero eigenvalues.

3 Fully Connected Neural Networks (25 points)

In this problem, you will implement a basic fully-connected neural network for classification on the CIFAR-10 dataset, which contains images for 10 different classes of data. You are not allowed to use an off-the-shelf implementation for your neural network. To implement the neural network, you are only allowed to use numpy.

Please follow the corresponding jupyter notebook as you work on this problem. It is more comprehensive than this page, as it includes test cases and other information to help you complete the problem. If you follow the jupyter notebook correctly, you should receive full points on this problem. In your submission, make sure to attach a pdf copy of your jupyter notebook so that we can examine your code and grade your submission.

(a) **(0 points) Setup** For this assignment, you need `numpy` and `matplotlib`. Make sure that you have these libraries running correctly either in a conda environment or locally. If these methods do not work, you are welcome to use Google Colab.

You will need to download the CIFAR-10 dataset.

Run the following from the 'hw5' directory:

```
cd deeplearning/datasets
./get_datasets.sh (or ./get_datasets_curl.sh if this does not work)
```

(b) **(2 points) Affine Layer** Implement `affine_forward()` and `affine_backward()` to construct the linear layers for our neural network. You can find these methods in `deeplearning/layers.py`

(c) **(2 points) ReLU Layer** Activation layers are non-linearities that allow our neural network to fit to complex patterns and functions. One example of a non-linearity that you may have seen in this course is the sigmoid non-linearity used for logistic regression. For this assignment, we will be using the ReLU (rectified linear unit) non-linearity. Implement `relu_forward()` and `relu_backward()` to construct the activation layers for our neural network. You can find these methods in `deeplearning/layers.py`

(d) **(6 points) Two Layer Network** Now, we are going to use the layers that we implemented to build a simple neural network! Complete the implementation of `TwoLayerNet` class in `deeplearning/classifiers/fc_net.py`. Then, in the Jupyter Notebook use a Solver instance to train a `TwoLayerNet` that achieves at least 50% validation accuracy

(e) **(6 points) Multilayer Network** Expanding upon the previous part, we are now going to implement a fully-connected network with an arbitrary number of hidden leayers. Complete the implementation of `FullyConnectedNet` class in `deeplearning/classifiers/fc_net.py`.

(f) **(3 Points) SGD + Momentum** So far we have used vanilla stochastic gradient descent (SGD) as our update rule. More sophisticated update rules can make it easier to train deep networks. For instance, stochastic gradient descent with momentum is a widely used update rule that tends to make deep networks converge faster than vanilla stochstic gradient descent. Complete the implementation of `sgd_momentum()` in `deeplearning/optim.py`.

(g) **(6 points) Training a Good Model** Train the best fully-connected model that you can on CIFAR-10, storing your best model in the `best_model` variable and the solver used in the `best_solver` variable. We require you to get at least 50% accuracy on the validation set using a fully-connected net.

4 PyTorch Tutorial (12 points)

In this assignment, you have implemented parts of a deep learning library that hopefully provide you with a clear understanding of the inner workings of different components in deep learning. For the last part of this assignment, we are going to leave behind your previous codebase to take advantage of the conveniences offered by a popular deep learning framework: Pytorch.

We recommend you run this notebook using Google Colab, which will already have PyTorch installed and ready to use, while also allowing you to utilize GPUs.

Please follow the corresponding jupyter notebook as you work on this problem. It is more comprehensive than this page, as it includes test cases and other information to help you complete the problem. If you follow the jupyter notebook correctly, you should receive full points on this problem. In your submission, make sure to attach a pdf copy of your jupyter notebook so that we can examine your code and grade your submission.

(a) **(2 points) Barebones PyTorch** Complete the Barebones PyTorch portion of the notebook to understand the barebones workings of PyTorch. Then, implement and train a simple 3-layer Convolutional Neural Network with barebones PyTorch.

(b) **(2 points) PyTorch Module API** Complete the PyTorch Module API portion of the notebook to understand the module API. Then, implement and train a simple 3-layer Convolutional Neural Network with the module API.

(c) **(2 points) PyTorch Sequential API** Complete the PyTorch Sequential API portion of the notebook to understand the Sequential API. Then, implement and train a simple 3-layer Convolutional Neural Network with the Sequential API.

(d) **(6 points) CIFAR-10 Challenge** Experiment with architectures, hyperparameters, loss functions, and optimizers to train a model that achieves **at least 70%** accuracy** on the CIFAR-10 validation set within 10 epochs. You can use the `check_accuracy()`and `train()` functions from above. You can use either `nn.Module` or `nn.Sequential` API.

Describe what you did here.