Winter 2026 DSC240: Introduction to Machine Learning
Homework 4

Due: **Monday, Mar 9th, 11:59 pm PST**

Notes:

1. **This assignment is to be done individually.** You may discuss the problems at a general level with others in the class (e.g., about the concepts underlying the question, or what lecture or reading material may be relevant), but the work you turn in must be solely your own.
2. Be aware of the late policy in the course syllabus – i.e., *no late days for all the assignments,* so **it is your responsibility to turn in your assignment to Gradescope by the due time.**
3. Justify every answer you give – **show the work** that achieves the answer or **explain** your response.
4. Any updates or corrections will be posted on Piazza, so check there occasionally.
5. This assignment requires **two parts of the submission**. Both of these submissions will be available on Gradescope, and **both submissions are MANDATORY** for receiving full credit for the assignment:
   (a) Written form of the submission as a **PDF** file format. This needs to be submitted in **Homework 4 - Report** on Gradescope. Please mention appropriate code sections/blocks in this report.
   (b) Code submission as a **.ipynb** jupyter notebook file. This needs to be submitted in **Homework 4 - Code** on Gradescope.
6. You can complete the assignment in the provided Jupyter notebooks, which allows you to write both the text (using LaTeX for math) and the code in one file. Once done, convert the notebook to a PDF and submit it to the **Homework 4 - Report** on Gradescope. Also, submit the .ipynb notebook files (with the code) to the **Homework 4 - Code** on Gradescope. This is just a suggestion – you can also submit the report using LaTeX/Markdown/handwriting and the code in a separate .ipynb file if you prefer.
7. Be sure to re-read the **"Academic Integrity"** on the course syllabus. You must complete the section below. If you answered Yes to either of the following two questions, give corresponding full details.

Did you receive any help whatsoever from anyone in solving this assignment?
Did you give any help whatsoever to anyone in solving this assignment?

Problem 1 (40 points + 14 Bonus points): Jax, Autograd and Neural Networks.

In this problem, you will learn how to work with an auto differentiation tool — jax. The wide availability of auto differentiation tool is one of the fundamental reasons why deep learning took off. In Part (a) and (b) you will learn the basic syntax on how you can differentiate over a piece of code. In part (c) and (d) you will learn how to apply it to implement a two layer neural network, and take gradient (without actually doing any math). In Part (e) you will study a from-scratch implementation of a more realistic neural network to the celebrated MNIST dataset and answer a few questions.

Objective: Learn how to use the auto differentiation tool `jax`. *Follow the exercises in the* `Q1_jax_and_autograd.ipynb` *notebook and discuss your observations.*

(a) Univariate Function Gradient(5 points)

Task: Complete the code for automatic gradient computation of a simple univariate function.

```
# Code for univariate function gradient
```

(b) Multivariate Function Gradient(5 points)

Task: Extend the implementation to handle multivariate functions.

```
# Code for multivariate function gradient
```

Bonus (2 points): Implement a function that computes the Hessian matrix for the given function.

(c) Soft-Max Regression and Neural Networks(15 points)

Task: Study the 'from scratch' implementation of soft-max regression and complete the neural network code using the 'tanh' activation function.

```
# Code for neural network with 'tanh' activation
```

(d) Neural Network Parameters and Gradient(15 points)

Task: Determine valid shapes for the parameters in a two-layer neural network and output the gradient as per the provided instructions.

```
# Code for determining parameter shapes and computing gradient
```

(e) Digit Classification from Scratch (Bonus 12 points)

Engage with the `digit_classification_from_scratch.ipynb` *notebook. Answer the following questions:*

- (1 point) How many layers are there in the neural network?

- (1 point) What are the shapes of each parameter in "params"?
- (2 points) Which activation function is used?
- (2 points) What are the input space and label space?
- (2 points) What does "predict" output?
- (2 points) What are the training accuracy and test accuracy after 10 epochs? Use the GUI, write your 1,2,3,4,...,10 using your mouse. Write a short report that shows your digits and the classification results.
- (2 points) What is the classification accuracy on your data? Is it better or worse than the accuracy on the test set? If it is very different, could you explain why? Any ideas to fix the issue?

Problem 2: K-Means Clustering(30 points)

In this problem, you will implement the Lloyd's algorithm for k-means clustering *from scratch* and apply it to two datasets. What does it mean by *from scratch*? You can use low-level packages such as numpy and scipy, but you cannot directly call kmeans from sklearn.

Objective: Implement and explore the K-means clustering algorithm. *Follow the instructions in the* `Q2_kmeans.ipynb` *notebook and respond to the associated queries.*

```
# Implementation  of the K-means algorithm and answer questions.
```

Answer the following questions based on your implementation:

1. Do you get the same results each time you run k-means? Why?
2. Describe what you see when you adjust k. You may reorganize the subplot layouts to see clearly.
3. What are the prominent differences you can see between the raw images and the "centroid faces"?

Problem 3: PCA for Dimension Reduction(30 points)

The goal of this problem is for you to explore tools in dimension reduction. What does it mean by *from scratch*? You can use low-level packages such as numpy and scipy (e.g., the eigendecomposition or SVD), but you cannot directly call pca from sklearn.

Objective: Learn to implement PCA and to reduce the dimensionality of data. *Complete the exercises in the 'Q3_dimension_reduction' notebook and discuss your observations.*

```
# Implementation and discussion for PCA dimension reduction
```

Answer the following questions based on your implementation:

1. When you rerun PCA with different choices of `number_of_components` do they return the same or different solutions? Why?
2. How would you interpret the eigenfaces?