Homework 2

Spring 2021
(Due: Friday, Feb 19, 2020, 11:59 pm Eastern Time)

Please submit your homework through **gradescope**. You can write, scan, type, etc. But for the convenience of grading, please merge everything into a **single PDF**.

Objective

The objective of this homework are:

(a) Familiarizing yourself with basic tools in Python that can perform the following tasks: reading and processing of data files (csv format), and numerically solving specified optimization problems;

(b) Review of important concepts from linear algebra and optimization theory, with emphasis on linear least square estimation;

(c) Implement a linear classification algorithm, visualize its decision boundary, and test its accuracy.

You will be asked some of these questions in Quiz 2. The Quiz will be open on Feb 20, 8am Eastern Time, and close on Feb 21, 8am Eastern Time. The Quiz is 30 minutes long.

Get Started

In this homework, you need to numerically solve several convex optimization problems in Python. We will use CVXPY. CVXPY is a Python-embedded modeling language with a user-friendly API for convex optimization problems.

In Google Colab, you can call CVXPY. Here is a toy example taken from the software's website:

```
import cvxpy as cp
import numpy as np

# Problem data.
m = 30
n = 20
np.random.seed(1)
A = np.random.randn(m, n)
b = np.random.randn(m)

# Construct the problem.
x = cp.Variable(n)
objective = cp.Minimize(cp.sum_squares(A@x - b))
constraints = [0 <= x, x <= 1]
prob = cp.Problem(objective, constraints)

# The optimal objective value is returned by `prob.solve()`.
result = prob.solve()
```

```
# The optimal value for x is stored in `x.value`.
print(x.value)
# The optimal Lagrange multiplier for a constraint is stored in
# `constraint.dual_value`.
print(constraints[0].dual_value)
```

For more examples, head over to `https://www.cvxpy.org/examples/index.html`.
Caution: In the newest version of CVXPY, matrix-matrix and matrix-vector multiplication uses `@` (@), e.g. if `A` is a matrix and `x` is a vector, then `A@x` is the matrix-vector multiplication of the pair. The basic example on `https://www.cvxpy.org/` still uses the deprecated `A*x`, which could cause instability in certain scenarios.

Exercise 1: Loading Data via Python

The National Health and Nutrition Examination Survey (NHANES) is a program to assess the health and nutritional status of adults and children in the United States [1]. The complete survey result contains over 4,000 samples of health-related data of individuals who participated in the survey between 2011 and 2014. In this homework, we will focus on two categories of the data for each individual: the height (in mm) and body mass index (BMI). The data is divided into two classes based on gender. Table 1 contains snippets of the data.

index	female bmi	female stature mm
0	28.2	1563
1	22.2	1716
2	27.1	1484
3	28.1	1651

index	male bmi	male stature mm
0	30	1679
1	25.6	1586
2	24.2	1773
3	27.4	1816

Table 1: Male and Female Data Snippets

Use `csv.reader` to read the training data files for the two classes of data. Specifically, you may call the commands below:

```
import numpy as np
import matplotlib.pyplot as plt
import cvxpy as cp
import csv

# Reading csv file for male data
with open("male_train_data.csv", "r") as csv_file:
    reader = csv.reader(csv_file, delimiter=',')
    # Add your code here to process the data into usable form
csv_file.close()

# Reading csv file for female data
with open("female_train_data.csv", "r") as csv_file:
    reader = csv.reader(csv_file, delimiter=',')
    # Add your code here to process the data into usable form
csv_file.close()
```

Important: Before proceeding to the problems, please be careful that numerical solvers can perform very badly if the inputs to the problem are badly scaled, e.g. many coefficients being orders of magnitude apart; this is known as **numerical instability**. It is exactly the case for our matrix $\boldsymbol{X}$. To avoid CVXPY (by default, the ECOS solver in it) returning bizarre solutions, please

[1] `https://www.cdc.gov/nchs/nhanes/index.htm`

- normalize the number in `male_stature_mm` and `female_stature_mm` by dividing them with 1000, and
- normalize that of `male_bmi` and `female_bmi` by dividing them with 10.

This will significantly reduce the numerical error.

Print the first 10 elements of each column of the dataset. That is, print

- The first 10 entries of female BMI;
- The first 10 entries of female stature;
- The first 10 entries of male BMI;
- The first 10 entries of male stature.

Please submit your code, and submit your results.

Exercise 2: Build a Linear Classifier via Optimization

Consider a linear model:

$$g_{\boldsymbol{\theta}} = \boldsymbol{\theta}^T \boldsymbol{x}, \tag{1}$$

The regression problem we want to solve is

$$\widehat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}\in\mathbb{R}^d}{\operatorname{argmin}} \sum_{n=1}^{N} \left(y_n - g_{\boldsymbol{\theta}}(\boldsymbol{x}_n)\right)^2,$$

where $\mathcal{D} = \{(\boldsymbol{x}_n, y_n)\}_{n=1}^N$ is the training dataset. Putting the equation into the matrix form, we know that the optimization is equivalent to

$$\widehat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}\in\mathbb{R}^d}{\operatorname{argmin}} \underbrace{\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta}\|^2}_{\mathcal{E}_{\text{train}}(\boldsymbol{\theta})}.$$

(a) Derive the solution $\widehat{\boldsymbol{\theta}}$. State the conditions under which the solution is the unique global minimum in terms of the rank of $\boldsymbol{X}$. Suggest two techniques that can be used when $\boldsymbol{X}^T\boldsymbol{X}$ is not invertible.

(b) For the NHANES dataset, assign $y_n = +1$ if the n-th sample is a male, and $y_n = -1$ if the n-th sample is a female. Implement your answer in (a) with Python to solve the problem. Report your answer, and submit your code.

(c) Repeat (b), but this time use CVXPY. Report your answer, and submit your code.

(d) Derive the gradient descent step for this problem. That is, find the gradient $\nabla\mathcal{E}_{\text{train}}(\boldsymbol{\theta}^k)$ and substitute it into the equation:

$$\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k - \alpha^k \nabla\mathcal{E}_{\text{train}}(\boldsymbol{\theta}^k).$$

If you use the exact line search, what is the optimal step size α^k (for each iteration k)?

(e) Implement the gradient descent algorithm in Python. Report your answer, and submit your code. Initialize $\boldsymbol{\theta}^0 = 0$. Use 50000 iterations.

(f) For the gradient descent algorithm you implemented in (e), plot the training loss as a function of iteration number using `plt.semilogx`. Use `linewidth = 8`.

(g) Implement the momentum method, with $\beta = 0.9$. Initialize $\boldsymbol{\theta}^0 = 0$. Use 50000 iterations.

$$\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k - \alpha^k \left(\beta\nabla\mathcal{E}_{\text{train}}(\boldsymbol{\theta}^{k-1}) + (1-\beta)\nabla\mathcal{E}_{\text{train}}(\boldsymbol{\theta}^k)\right).$$

Here, the step size α^k can be determined through the exact line search.

(h) For the momentum method you implemented in (g), plot the training loss as a function of iteration number using `plt.semilogx`. Use `linewidth = 8`.

Hint: If you do everything right, the answers found by the analytic expression, CVXPY, and gradient descent should be the same.

Exercise 3: Visualization and Testing

We want to do a classification based on the linear model we found in Exercise 2. The classifier we will use is

$$\text{predicted label} = \text{sign}(g_{\boldsymbol{\theta}}(\boldsymbol{x})), \tag{2}$$

where $\boldsymbol{x} \in \mathbb{R}^d$ is the a test sample. Here, we label +1 for male and -1 for female. Because the dataset we consider in this exercise has only two columns, the linear model is

$$g_{\boldsymbol{\theta}}(\boldsymbol{x}) = \theta_0 + \theta_1 x_1 + \theta_2 x_2,$$

where $\boldsymbol{x} = [1, x_1, x_2]^T$ is the input data, and $\boldsymbol{\theta} = [\theta_0, \theta_1, \theta_2]^T$ is the parameter vector.

(a) First, we want to visualize the classifier.

(i) Plot the training data points of the male and female classes using `matplotlib.pyplot.scatter`. Mark the male class with blue circles, and the female class as red dots.

(ii) Plot the decision boundary $g_{\boldsymbol{\theta}}(\cdot)$, and overlay it with the data plotted in (a). Hint: $g_{\boldsymbol{\theta}}(\cdot)$ is a straight line in 2D. You can express x_2 in terms of x_1 and other parameter.

(b) Let us report the classification accuracy of `male_test_data.csv` and `female_test_data.csv`. To do so, take a testing data $\boldsymbol{x}$ and compute the prediction according to (2).

(i) What is the Type 1 error (False Alarm) of classifying male? That is, what is the percentage of testing samples that should be female but you predicted it as a male.

(ii) What is the Type 2 error (Miss) of classifying male? That is, what is the percentage of testing samples that should be male but you predicted it as a female.

(iii) What is the precision and recall for this classifier? For definition of precision and recall, you can refer to my book, Chapter 9.5.4, or Wikipedia.

Exercise 4: Regularization

Consider the following three optimization problems:

$$\widehat{\boldsymbol{\theta}}_\lambda = \underset{\boldsymbol{\theta}\in\mathbb{R}^d}{\text{argmin}} \quad \|\boldsymbol{X}\boldsymbol{\theta} - \boldsymbol{y}\|_2^2 + \lambda\|\boldsymbol{\theta}\|_2^2, \tag{3}$$

$$\widehat{\boldsymbol{\theta}}_\alpha = \underset{\boldsymbol{\theta}\in\mathbb{R}^d}{\text{argmin}} \quad \|\boldsymbol{X}\boldsymbol{\theta} - \boldsymbol{y}\|_2^2 \quad \text{subject to} \quad \|\boldsymbol{\theta}\|_2^2 \leq \alpha, \tag{4}$$

$$\widehat{\boldsymbol{\theta}}_\epsilon = \underset{\boldsymbol{\theta}\in\mathbb{R}^d}{\text{argmin}} \quad \|\boldsymbol{\theta}\|_2^2 \quad \text{subject to} \quad \|\boldsymbol{X}\boldsymbol{\theta} - \boldsymbol{y}\|_2^2 \leq \epsilon. \tag{5}$$

(a) Set `lambd = np.arange(0.1,10,0.1)`. Plot

- $\|\boldsymbol{X}\widehat{\boldsymbol{\theta}}_\lambda - \boldsymbol{y}\|_2^2$ as a function of $\|\widehat{\boldsymbol{\theta}}_\lambda\|_2^2$
- $\|\boldsymbol{X}\widehat{\boldsymbol{\theta}}_\lambda - \boldsymbol{y}\|_2^2$ as a function of λ
- $\|\widehat{\boldsymbol{\theta}}_\lambda\|_2^2$ as a function of λ

(b) (i) Write down the Lagrangian for each of the three problems. Note that the first problem does not have any Lagrange multiplier. For the second and the third problem, you may use the notations:

- γ_α = the Lagrange multiplier of (4), and
- γ_ϵ = the Lagrange multiplier of (5).

(ii) State the first order optimality conditions (the Karush-Kuhn-Tucker (KKT) conditions) for each of the three problems. See Tutorial 4 on Optimization, posted on the course website. Express your answers in terms of $\boldsymbol{X}$, $\boldsymbol{\theta}$, $\boldsymbol{y}$, λ, α, ϵ, and the two Lagrange multipliers γ_α, γ_ϵ.

(iii) Fix $\lambda > 0$. We can solve (3) to obtain $\widehat{\boldsymbol{\theta}}_\lambda$. Find α and the Lagrange multiplier γ_α in (4) such that $\widehat{\boldsymbol{\theta}}_\lambda$ would satisfy the KKT conditions of (4).

(iv) Fix $\lambda > 0$. We can solve (3) to obtain $\widehat{\boldsymbol{\theta}}_\lambda$. Find ϵ and the Lagrange multiplier γ_ϵ in (5) such that $\widehat{\boldsymbol{\theta}}_\lambda$ would satisfy the KKT conditions of (5).

(v) This part follows from (iii). Fix $\lambda > 0$. By using the α and γ_α you found in (iii), you can show that $\widehat{\boldsymbol{\theta}}_\lambda$ would satisfy the KKT conditions of (4). Is it enough to claim that $\widehat{\boldsymbol{\theta}}_\lambda$ is the solution of (4)? If yes, why? If no, what else do we need to show? Please elaborate through a proof, if needed.

Exercise 5: Project: Check Point 2

In check point 1, I asked you to find the necessary resources for your project. I assume that you have already done it. The goal of check point 2 is to get your hands dirty.

No matter what project you do, you need to have a baseline. A baseline is an existing method that has been reported in the literature, typically with Python codes available. Understanding the baseline is useful for multiple reasons: (1) Since you are a beginner to the problem, you need to learn the baseline before you can innovate. (2) Instead of reinventing the wheel, it is usually more efficient to customize a baseline for your own problem. (3) Ultimately, you need to compare with the baseline. So it is better to set up the baseline early.

So, in check point 2, you should aim to *reproduce* the results reported in the main reference you choose. "Reproduce" can be a fuzzy word, because you may not have the dataset and computing power as the authors do. I normally suggest two steps here:

- (1) Run their pre-trained models (if available) on your testing dataset. Check if the results make sense. This step is mandatory for check point 2.
- (2) Modify the training dataset and try to retrain their model. Check if the results make sense. This step is optional for check point 2 but will be required for check point 3.

Many of you are new to Python and the project. I completely understand that it will take you some time to digest the code and type your hello world. Therefore, I only put (1) as the mandatory, and (2) as the optional.

Different project topics have different focuses, and therefore different approaches. You should make the best judgment of what constitutes your baseline, and what kind of baseline results you would need. Below are just some suggestions for you to consider:

- Topic 1: Noise2Noise: Download the existing noise2noise code, and run it on a set of noisy images. Plot the denoised image quality (measured in terms of PSNR) as a function of the noise level. To accomplish this task, you need to know how to generate noise, create a testing dataset, run the noise 2 noise pre-trained model, and plot the curve. This is what you should aim to achieve at this check point.
- Topic 2: PnP: Download several denoisers (including deep neural networks). Plot the denoising results as a function of the noise levels, for different denoisers, so that you can conclude whether there is a uniformly superior denoiser. To accomplish this task, you need to know how to test denosiers. We haven't integrated this with PnP, but you should start to think about it at this check point.
- Topic 3: Attack: You need to be able to attack, e.g., using FGSM on a simple classifier. There are plenty of codes available online. Download one, and plot the attack success rate as a function of attack strength. You should also be able to run adversarial training. Repeat the same exercise, but this time using an adversarially trained classifier. To accomplish this task, you need to know how to attack, and how to do adversarial training.

- Topic 4: Noisy label: You need to download and run an existing code. Try to perturb the labels according to some simple models. Plot the classification accuracy as a function of the amount of noisy labels. To accomplish this task, you need to be familiar with the code, and be able to perturb labels.

As you can see, for every topic, I am looking for **quantitative experiments**, and not qualitative descriptions. So, for this check point, I hope to see at least one figure (such as the ones I suggested above) and some text describing what you have done. (You can use a table instead of figure if you find it more suitable.) Add them to the LaTeX report you submitted for check point 1, and submit it with homework 2. To help us check whether you have done this task, please add a section called "check point 2 basline". Limit the number of pages for your check point 2 to one page. (Since the TAs are handling 100+ submissions, they will only check whether you have done the work. They will not grade your check point. I hope you understand. If you need help, you can post your questions on Piazza.)