Homework 4

Spring 2023
(Due: Thursday, Mar 23, 2023, 4:59 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

There are three things you will learn in this homework:

(a) Understand some theoretical properties about logistic regression.
(b) Implement a logistic regression in CVX, and visualize the decision boundary.
(c) Apply kernel trick to logistic regression.

You will be asked some of these questions in Quiz 4.

Exercise 1. Logistic regression + gradient descent
We analyze the convergence behavior of the logistic regression when the data is **linearly separable**.

Recall that the logistic regression tries to minimize the cross-entropy loss:

$$\widehat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}\in\mathbb{R}^d}{\text{argmin}} -\sum_{n=1}^{N} \Big\{ y_n \log h_{\boldsymbol{\theta}}(\boldsymbol{x}_n) + (1-y_n)\log(1-h_{\boldsymbol{\theta}}(\boldsymbol{x}_n)) \Big\}, \tag{1}$$

where $h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \frac{1}{1+\exp\{-\boldsymbol{\theta}^T\boldsymbol{x}\}}$ is the sigmoid function. As usual, we assume that $\boldsymbol{\theta} = [\boldsymbol{w},\ w_0]$. We consider the gradient descent algorithm. We know that the objective function is convex, and so we consider the gradient descent algorithm. The iterations are (if you take the derivative of $J(\boldsymbol{\theta})$ and rearrange the terms):

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} - \alpha_k \left(\sum_{n=1}^{N} (h_{\boldsymbol{\theta}^{(k)}}(\boldsymbol{x}_n) - y_n)\boldsymbol{x}_n \right) \tag{2}$$

for some choices of the step size α_k.

(i) Prove that if two classes of data in $\mathbb{R}^d$ are linearly separable, then the magnitude of the slope $\|\boldsymbol{w}\|_2$ and the magnitude of the intercept $|w_0|$ would tend to ∞.

(ii) What happens if we restrict $\|\boldsymbol{w}\|_2 \le c_1$ and $|w_0| < c_2$ for some $c_1, c_2 > 0$? What other ways can you come up with to counter the nonconvergence issue?

(iii) Does linear separability of data cause nonconvergence for the other linear classifiers that we have studied? Why?

Exercise 2. Implement logistic regression
Download `hw4_data.zip` from Brightspace. There are two classes with class labels $y_n = 1$ and $y_n = 0$.

(a) Show that the logistic regression loss is given by

$$J(\boldsymbol{\theta}) = -\left\{ \left(\sum_{n=1}^{N} y_n \boldsymbol{x}_n \right)^T \boldsymbol{\theta} - \sum_{n=1}^{N} \log\left(1 + e^{\boldsymbol{\theta}^T \boldsymbol{x}_n}\right) \right\}.$$

(b) Introduce a regularization term $\lambda\|\boldsymbol{\theta}\|^2$ and normalize the data fidelity term so that the loss becomes

$$J(\boldsymbol{\theta}) = -\frac{1}{N}\left\{ \left(\sum_{n=1}^{N} y_n \boldsymbol{x}_n \right)^T \boldsymbol{\theta} - \sum_{n=1}^{N} \log\left(1 + e^{\boldsymbol{\theta}^T \boldsymbol{x}_n}\right) \right\} + \lambda\|\boldsymbol{\theta}\|^2. \tag{3}$$

Use CVXPY to minimize this loss function for the dataset I provided. Your $\boldsymbol{\theta}$ should be $\boldsymbol{\theta} = [\theta_2, \theta_1, \theta_0]^T$. Please use $\lambda = 0.0001$.

(c) Scatter plot the data points by marking the two classes in two colors. Then plot the decision boundary.

(d) Repeat (c), but this time using the Bayeisn decision rule. Note that since the covariance matrices are not identical, the decision boundary is not a straight line. To plot the decision boundary, you can create a grid of testing sites in the range of $[-5, 10] \times [-5, 10]$ (with 100 points along each dimension). Evaluate the decision on these testing sites. And then plot the decision using `plt.contour`.

Exercise 3. Kernel trick
Let us continue to use the dataset in Exercise 2. Our goal here is to implement the kernel trick.

(a) In Python, construct the kernel matrix $\boldsymbol{K}$, where

$$[\boldsymbol{K}]_{m,n} = \exp\left\{-\|\boldsymbol{x}_m - \boldsymbol{x}_n\|^2/h\right\}, \tag{4}$$

where $h = 1$. Print `K[47:52,47:52]`.

(b) Let us assume that that $\boldsymbol{\theta} = \sum_{n=1}^{N} \alpha_n \boldsymbol{x}_n$ for some α_n's. Then

$$\boldsymbol{\theta}^T \boldsymbol{x} = \sum_{n=1}^{N} \alpha_n \langle \boldsymbol{x}_n, \boldsymbol{x} \rangle.$$

The kernel trick says that we can replace $\langle \boldsymbol{x}_n, \boldsymbol{x} \rangle$ by a kernel $K(\boldsymbol{x}_n, \boldsymbol{x})$. Apply the kernel trick to the loss function in (3). Show that the new loss is

$$J(\boldsymbol{\alpha}) = -\frac{1}{N}\left\{ \boldsymbol{y}^T \boldsymbol{K}\boldsymbol{\alpha} - \boldsymbol{1}^T \log(e^{\boldsymbol{0}} + e^{\boldsymbol{K}\boldsymbol{\alpha}}) \right\} + \lambda \boldsymbol{\alpha}^T \boldsymbol{K}\boldsymbol{\alpha}.$$

(c) Implement the kernel logistic regression training in CVXPY. Report the first two elements of the regression coefficients $\boldsymbol{\alpha}$.

(d) Scatter plot the data points and plot the decision boundary, just like what you did in Exercise 2(d).

Exercise 4. Project Check Point # 4
At this checkpoint, I expect you to have started reimplementing the assigned paper. Here are a couple important reminders for you.

- Start simple. For the final project, we don't need you to redo the experiment in the assigned paper. The purpose of this project is for you to reimplement the method so that it can be used for your own tasks. Therefore, start with as many assumptions as you need to keep your problem tractable. For example, you can assume the data you work on is low dimensional, or you can use simple dataset at first. The most important thing is to demonstrate something that works as soon as possible. Then, you can gradually make your program more complicated.
- Leverage existing Python packages. You are allowed and encouraged to use many popular Python packages, such as numpy, TensorFlow, PyTorch, opencv, etc. For many basic operations that you want to achieve, it is likely that there are already implementations in these packages. So always check or ask first before you implement something basic by yourself.
- Start early. Start early. Start early.

Here are some good programming habits I appreciate. I know it might be hard to follow at first, but if you follow them for long enough time, they will gradually become your muscle memory. Furthermore, these habits will benefit you A LOT in the long run if you plan to work for a coding company.

- Keep good naming conventions for your variables and functions, in which case you will not even need to write comments. Good code explains themselves. For example, here is a sample Python naming convention: `https://visualgit.readthedocs.io/en/latest/pages/naming_convention.html`.
- Do not write functions that is too long, which will be difficult to debug. Keep a good structure for your program. Do not copy and paste the same code snippet in your program. If you want to reuse the same code snippet, write a function for it.
- Do not use a lot of for loops. It will make your Python program super slow. Try to vectorize your program. See this page for more details. `https://www.geeksforgeeks.org/vectorization-in-python/`.
- Train yourself to master debugging. It is normal that your code does not do what you want. The time to debug your code is the most valuable in terms of improving your programming skills. Everyone will have their own way to debug. My favorite debug tool is adding breakpoint in the program using `pdb.set_trace()`.

For this checkpoint, you need to create a new section in your overleaf document. In the section, write down the following things:

- Write down the assumptions that you are making for your project.
- Draw a diagram that describes the structure of your code. If you don't know how to do it, simply use a box to represent each function in your program, specify what are the inputs and outputs, and connect the boxes according to your code structure. This will help us understand how you design your code.
- Report any errors that you encounter when programming. If you have resolved the problem, what did you do? If not, what have you tried and why do you suspect that you cannot solve the problem?