Winter 2024 CMPUT 466/566: Machine Learning
Instructor: Bailey Kacsmar
Teaching Assistant: Alex Ayoub
Assignment #2
Student Name: Khac Nguyen Nguyen

Instructions

Deadline: February 29

Submission instructions: You need to submit a zip file containing a PDF file, named `a02_<name>.pdf` where `<name>` is your name, and an .ipynb file, named `a02_<name>.ipynb` where `<name>` is your name, containing your code downloaded from Google's colab. The PDF file should include your typed up solutions (we strongly encourage to use pdfLaTeX) and the colab notebook should have working and documented code. Write your name in the title of your PDF file. We provide a LaTeXtemplate that you are encouraged to use.

IMPORTANT Collaboration and sources Work on your own. You can consult the problems with your classmates, use books or web (like Wikipedia), papers, etc. Also, the write-up (and code) must be your own and you must acknowledge all the sources (names of people you worked with, books, webpages etc., including class notes.) Failure to do so will be considered cheating. Identical or similar write-ups (and code) will be considered cheating as well. Students are expected to understand and explain all the steps of their solutions.

Scheduling Start early: It takes time to solve the problems, as well as to write down the solutions. Most problems should have a short solution. The code should be short and clean as well.

Problems

Question 1. Newton's method can be used to find local minimums. Let

$$f(x) = 4x^5 + 10x^4 + \sqrt{24}x^3 + x\,.$$

Compute the first five iterations of Netwon's method (for finding local minimums) starting at $x_0 = -1$, i.e. compute

$$\begin{aligned} x_0 &= -1 \\ x_1 &= x_0 + \dots \\ x_2 &= x_1 + \dots \\ x_3 &= x_2 + \dots \\ x_4 &= x_3 + \dots \\ x_5 &= x_4 + \dots \end{aligned}$$

and show your steps.

Solution 1. First, we need to calculate the derivative

$$f'(x) = 20x^4 + 40x^3 + 3\sqrt{24}x^2 + 1$$

$$
\begin{aligned}
x_0 &= -1\\
x_1 &= x_0 - \frac{f(x_0)}{f'(x_0)} = -0.9765235720156236,\\
x_2 &= x_1 - \frac{f(x_1)}{f'(x_1)} = -0.9757863578498447\\
x_3 &= x_2 - \frac{f(x_2)}{f'(x_2)} = -0.9757856016813968\\
x_4 &= x_3 - \frac{f(x_3)}{f'(x_3)} = -0.9757856016806005\\
x_5 &= x_4 - \frac{f(x_4)}{f'(x_4)} = -0.9757856016806005
\end{aligned}
$$

Since we need to find the local minimum, instead of $x_n = x_{n-1} + \ldots$, we subtract instead. The code below show the computing process

```python
import math

def function(x):
    return 4*x**5 + 10*x**4 + math.sqrt(24)*x**3 + x
def derivative(x):
    return 20*x**4 + 40*x**3 + 3 * math.sqrt(24) * x**2 + 1

lst = [-1]
for i in range (5):
    x = lst[-1]
    lst.append(x - function(x)/derivative(x))
print(lst)
```

Total: **10 points**

Question 2. Redo the computations for Example 1 of the Day 8/9 lecture slides, linked here, *but with* $\mathrm{Loss}(h_w) = y \log \frac{1}{\hat{y}} + (1-y)\log\frac{1}{1-\hat{y}}$.

Solution 2.

$$
\begin{aligned}
&\frac{\partial}{\partial w_{3,5}} \text{Loss}(h_w) \\
=&\frac{\partial}{\partial w_{3,5}} \left(y \log \frac{1}{\hat{y}} + (1-y) \log \frac{1}{1-\hat{y}} \right) \\
=&y \cdot \frac{\partial}{\partial w_{3,5}} \log \frac{1}{\hat{y}} + (1-y) \frac{\partial}{\partial w_{3,5}} \log \frac{1}{1-\hat{y}} \\
=&y \cdot \hat{y} \cdot \frac{-1}{\hat{y}^2} \cdot \frac{\partial}{\partial w_{3,5}} \hat{y} + (1-y)(1-\hat{y}) \cdot \frac{1}{(1-\hat{y})^2} \frac{\partial}{\partial w_{3,5}} \hat{y} \\
=&y \cdot \frac{-1}{\hat{y}} \cdot \frac{\partial}{\partial w_{3,5}} \hat{y} + (1-y) \cdot \frac{1}{1-\hat{y}} \frac{\partial}{\partial w_{3,5}} \hat{y} \\
=&\frac{\partial}{\partial w_{3,5}} \hat{y} \left[-\frac{y}{\hat{y}} + \frac{1-y}{1-\hat{y}} \right] \\
=&g_5'(\text{in}_5) \frac{\partial}{\partial w_{3,5}} \text{in}_5 \left[-\frac{y}{\hat{y}} + \frac{1-y}{1-\hat{y}} \right] \\
=&g_5'(\text{in}_5) \frac{\partial}{\partial w_{3,5}} (w_{0,5} + w_{3,5} a_3 + w_{4,5} a_4) \left[-\frac{y}{\hat{y}} + \frac{1-y}{1-\hat{y}} \right] \\
=&g_5'(\text{in}_5) a_3 \left[\frac{-y + \hat{y}}{\hat{y}(1-\hat{y})} \right]
\end{aligned}
$$

Total: **10 points**

Question 3. In this question you will be tasked with implementing a logistic regressor.

Recall that in logistic regression the learner is presented with a dataset $D = \{(X_i, Y_i)\}_{i=1}^n$ of size n where $X_i \in \mathbb{R}^d$ is a d dimensional feature vector and $Y_i \in [0, 1]$ is a random observation. The expected value of an observation Y_i conditioned on a feature vector X_i is given by the following model

$$\mathbb{E}[Y_i \mid X_i] = \frac{1}{1 + \exp(-X_i^\top \theta_\star)}$$

where $\theta_\star \in \mathbb{R}^d$ is the d dimensional true parameter vector that generates the observations.

For this question you will implement code that takes input a dataset of size $n = 1,000,000$ (see FEATURE.NPY and OBS.NPY) and returns a parameter vector $\theta \in \mathbb{R}^{10}$, since $d = 10$ on this problem. If done correctly, the parameter vector your code returns should be "close" to the true parameter vector $\theta_\star$ that generated the observations, i.e. Y_i's.

Your code will be evaluated on

1. How fast it runs (my code runs in under 5 seconds) on a Google colab notebook.
2. How close the returned parameter vector θ is true parameter vector $\theta_\star$ (i.e. the parameter vector the minimizes the *training loss*) of some arbitrary (different) dataset.

For full marks, your code should also run in under 5 seconds (on a colab notebook) and produce a "good" estimate of $\theta_\star$. You are *expected* to Google solution methods and *encouraged* to use python libraries such as numpy. We are placing *no restrictions* on which python libraries you import and use. Thus we strongly encourage you to read the documentation of different libraries and familiarize yourself with their best practices.

Solution 3. We will separate the explanations by block, block 5 is simply a verification process and should not be considered in the timed process.

Block 1. Import necessary library.

Block 2. Logistic Regression using scipy BFGS method to minimize the loss function.

- ”sigmoid” are simply function to calculate the sigmoid.
- ”LogLoss” and ”derivative” are functions necessary for BFGS. The method is a branch of Newton’s method thus need the function that we want to minimize which is the LogLoss and the derivative of LogLoss.
- The solve function simply apply BFGS to find the weights that minimize the loss function. Even though the method minimize has a few RuntimeWarning it is safe to say that it does not affect the performance even though the message for the minimize method is: ”Desired error not necessarily achieved due to precision loss.” Here are 2 reasons why:

  If RuntimeWarning appears, it appears as a consequence of $\hat{y}$ reaching close to 0 or 1. WLOG, let’s say it is reaching close to 0, in this case, it is likely that the correct label for that data point is also 0. Thus the loss function of that data point return 0 (because of np.nansum) which means that that data point have no error, which is what we want.

  In the case where $\hat{y}$ is close to 0 but y is 1, this should still work because of the second reason, which is the amount of RuntimeWarning observed was little compared to the total amount of data.

Block 3. Import data, split data and feed the data into the model

Block 4. The timed block

Block 5. Verification of the weights using other methods and accuracy. Note that we will use the threshold as 0.5 here as logistic regression do not return the probability, it returns the clasess with the higher probability for any data point.

- LogisticRegression2 is Logistic Regression using Gradient Descent. The solve function of this class takes in data and iterate using gradient descent to update the weights that minimize the loss function until the changes are less than a certain amount, where we will break out of the loop.
- Verification using accuracy: since we split the datasets earlier, we can simply test the accuracy on the test sets using the ”accuracy” function, the function compare the predicted class using the weight and their true class. We can see that both methods have really high accuracy.
- Verification by comparing the weights with Gradient Descent: looking at the weights of both methods, they are not the same for the following reason:
  Since the threshold is 0.5, sigmoid$(x_i \cdot w) > 0.5$ whenever $x_i \cdot w > 0$ and is ≤ 0.5 whenever $x_i \cdot w \leq 0$. This means that if w_{opt} is the optimal weights then $2w_{\text{opt}}$ is also an optimal weights.
  This could be easily verified easily when dividing the 2 weights elementwise from the 2 methods. We can see that all the values would be approximately the same by looking at the standard deviation of the division (really small).

Output:
BGFS classification accuracy: 0.99882
GD classification accuracy: 0.99913
division of the 2 weights from 2 methods: [0.97633602 0.97503745 0.97235876 0.97718222 0.97655073 0.97355438 0.97970106 0.97715803 0.97779888 0.97753152]
standard deviation of the division: 0.002038887190360162

Total: **30 points**

Total for all questions: 50.

References

[1] https://github.com/aadeshd/MLAlgorithms-Scratch/blob/master/Logistic%20Regression/Logistic-Scratch.ipynb

[2] https://machinelearningmastery.com/bfgs-optimization-in-python/