Assignment 1: Basics of Regression

UVA CS 4774
Machine Learning Foundation, Deep Learning and Good Uses

January 20, 2022

a *The assignment should be submitted in the PDF format through Collob. If you prefer hand-writing QA parts of answers, please convert them (e.g., by scanning or using an app like Genuis Scan) into PDF form.*

b *For questions and clarifications, please post on Slack.*

c *Policy on collaboration:*

Homework should be done individually: each student must hand in their own answers. It is acceptable, however, for students to collaborate in figuring out answers and helping each other solve the problems. We will be assuming that, with the honor code, you will be taking the responsibility to make sure you personally understand the solution to any work arising from such collaboration.

d *Policy on late homework:*

Homework is worth full credit at the midnight on the due date. Each student has 10 *extension days to be used at his or her own discretion throughout the entire course. You could use the days in whatever combination you like. For example, all late days on 1 assignment or 1 each day over 10 assignments (for a maximum grade of 90% on each). After you've used all 10 days, you cannot get credit for anything turned in late.*

e *Policy on grading:*
1*: 30 points for code submission (and able to run), 10 point for successfully loading data, 60 points for correct implementation and good discussion of each optimization function (a total of 4).*
The overall grade will be divided by 10. Therefore, you can earn 10 out of 10.

Please provide proper steps to show how you get the answers.

1 Linear Regression Model Fitting (Programming)

There are **TWO** portions in this section.

1.1 Coding

First, you must fill in the provided code template. This code will perform linear regression on a data file named “regression-data.txt” which is also provided. In the given data, first column represents the intercept term, second is the x value, third column is the y value.

(BTW: This coding assignment does not include validation or testing which is a bad practice but we will do them later in the semester.)

Please submit your python code as “linearRegression.py” and use Python3.

ATT: If you highly prefer Jupyter notebook, you can submit the notebook version following exactly the same code template we have provided. (Coding via google collab notebook avoids the step of setting up python and related library locally on your computer. We highly recommend the non-notebook way because installing locally is a necessary and basic skill.)

Using the “numpy” arrays and functions from `http://www.numpy.org` is required. (BTW: Numpy arrays and functions perform mathmatical operations faster because they allow for vectorization and use optimized libraries. For this dataset size, it is irrelevant, but for larger datasets it means, for example the difference between waiting 1 hour or 4 days. So we are forcing you to use them as practice.)

1.2 Written Portion

Second, you must complete a written portion and turn it in as part of the pdf with the rest of the assignment. For the written portion you must describe what happens to the loss function per epoch as the learning rate changes in both gradient descent and stochastic gradient descent and explain why.

- Function load_data_set() should also output a figure plotting the data; Please submit the plot in the written part of the homework.
- For each optimization method you implement to learn the best LR line, please submit a figure showing the data samples and also draw the best-fit line which has been just learned. Please also include the concrete value of the derived theta in the written part of the homework.
- For each optimization method you implement to learn the best LR line (your GD, SGD or MiniSGD implementation):
  - The functions should output a figure with x-axis showing epoch number (the updating iteration t, t denotes the total number of times the entire training set is iterated over), and y-axis showing the mean training loss at that epoch.
  - The functions should use mean square error(MSE) as the loss function.
  - The functions should stop when t reaches a predefined value $t_{max} = 100$;
  - In the written part of the submission, you should discuss the behavior of the function when varying the value of the learning rate (for instance varying values from $\{0.001, 0.005, 0.01, 0.05, 0.1, 0.3\}$.
  - It is good practice to perform a random shuffle of your training samples before each epoch of (mini-batch) SGD;
  - In mini-batch SGD, you should try to observe the convergence behaviors by varying the size B you used for sizing the mini-batch.

1.3 Recommendations

- Implement the code in order that it is given in the template main function.
- Shuffle the x and y before using stochastic gradient descent. (Make sure to shuffle them together.)
- 0s will not work as hyperparameters for learning rate or number of iterations.

We will run "python3 linearRegression.py" and it should work!
A few useful links for using numpy array:

- basics (more than we need): `https://docs.scipy.org/doc/numpy/user/quickstart.html`
- a good comprehensive list of math operations we normally need to use on numpy arrays `https://www.numpy.org/devdocs/user/numpy-for-matlab-users.html#array-or-matrix-which-should-i-use`

HW1 Programming report answer template

A. Dataset:

<Plot the given dataset (y vs x).>

Fig 1: Plot of given dataset

B. Linear Regression with Normal Equation:

<Plot the given dataset and the best-fit LR line for the Normal equation.>

Fig 2: Best-fit LR line with the Normal equation

C: Linear Regression with Gradient Descent (GD):

<Plot loss vs epoch for Gradient Descent with low learning rate. Plot for max_epochs = 100. Here, low rate means that the loss will not converge by the end of the 100th epoch.>

Fig 3: Gradient Descent loss vs epoch with learning rate of <your learning rate>

<Plot loss vs epoch for Gradient Descent with high learning rate. Plot for max_epochs = 100. Here, high rate means that the loss will diverge by the end of the 100th epoch.>

Fig 4: Gradient Descent loss vs epoch with learning rate of <your learning rate>

<Plot loss vs epoch for Gradient Descent with optimum learning rate. Plot for max_epochs = 100. Here, optimum rate means that the loss will converge before 100th epoch.>

Fig 5: Gradient Descent loss vs epoch with learning rate of <your learning rate>

<Plot the given dataset and the best-fit LR line for Gradient Descent.>

Fig 6: Best-fit LR line with Gradient Descent. Used learning rate=<learning rate>

C1: <Explain the effect of varying learning rate for GD>

D: Linear Regression with Stochastic Gradient Descent (SGD):

<Plot loss vs epoch for SGD with low learning rate. Plot for max_epochs = 100. Here, low rate means that the loss will not converge by the end of the 100th epoch.>

Fig 7: SGD loss vs epoch with learning rate of <your learning rate>

<Plot loss vs epoch for SGD with high learning rate. Plot for max_epochs = 100. Here, high rate means that the loss will diverge by the end of the 100th epoch.>

Fig 8: SGD loss vs epoch with learning rate of <your learning rate>

<Plot loss vs epoch for SGD with optimum learning rate. Plot for max_epochs = 100. Here, optimum rate means that the loss will converge before 100th epoch.>

Fig 9: SGD loss vs epoch with learning rate of <your learning rate>

<Plot the given dataset and the best-fit LR line for Gradient Descent.>

Fig 10: Best-fit LR line with SGD. Used learning rate=<learning rate>

D1: <Explain the effect of varying learning rate for SGD>

E. Linear Regression with Mini-batch Gradient Descent (MiniSGD):

<Plot loss vs epoch for MiniSGD with low learning rate and the batch size of 20. Plot for max_epochs = 100. Here, low rate means that the loss will not converge by the end of the 100th epoch.>

Fig 11: SGD loss vs epoch with learning rate=<used learning rate> and batch_size=20

<Plot loss vs epoch for MiniSGD with high learning rate and the batch size of 20. Plot for max_epochs = 100. Here, high rate means that the loss will diverge by the end of the 100th epoch.>

Fig 12: SGD loss vs epoch with learning rate=<used learning rate> and batch_size=20

<Plot loss vs epoch for SGD with optimum learning rate and the batch size of 20. Plot for max_epochs = 100. Here, optimum rate means that the loss will converge before 100th epoch.>

Fig 13: SGD loss vs epoch with learning rate=<used learning rate> and batch_size=20

E1: <Explain the effect of varying learning rate for MiniSGD>

<Plot loss vs epoch for MiniSGD with the batch size of 5 and optimum learning rate used for Fig 13. Plot for max_epochs = 100.>

Fig 14: SGD loss vs epoch with learning rate==<used learning rate> and batch_size=5

<Plot loss vs epoch for MiniSGD with the batch size of 40 and optimum learning rate used for Fig 13. Plot for max_epochs = 100.>

Fig 15: SGD loss vs epoch with learning rate=<used learning rate> and batch_size=40

<Plot loss vs epoch for MiniSGD with the batch size of 100 and optimum learning rate used for Fig 13. Plot for max_epochs = 100.>

Fig 16: SGD loss vs epoch with learning rate=<used learning rate> and batch_size=100

<Plot the given dataset and the best-fit LR line for MiniSGD.>

Fig 17: Best-fit LR line with MiniSGD. Here, learning rate=<used learning rate> and batch_size=<used batch size>

E2: <Explain the effect of varying batch size for MiniSGD>