NCTU Pattern Recognition, Homework 1

Deadline: April 3, 23:55

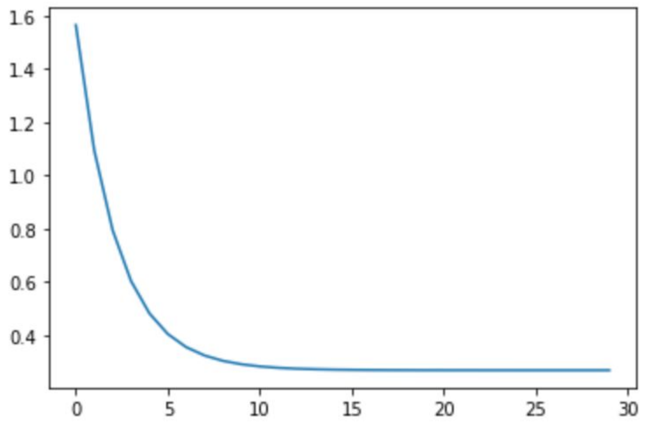
Part. 1, Coding (60%):

In this coding assignment, you need to implement linear regression by using only NumPy, then train your implemented model using **Gradient Descent** by the provided dataset and test the performance with testing data. Find the sample code and data on the GitHub page https://github.com/NCTU-VRDL/CS_DCP3121/tree/master/HW1

We suggest using the hyper-parameters below:

- Loss function: Mean Square Error
- Learning rate: 1e-4
- Number of training iteration: 100

Please note that only NumPy can be used to implement your model, you will get no points by simply calling sklearn.linear_model.LinearRegression. Moreover, please train your regression model using Gradient Descent, not the closed-form solution.

1. (15%) Plot the learning curve of the training, you should find that loss decreases after a few iterations (x-axis=iteration, y-axis=loss, Matplotlib or other plot tools is available to use)
    a. 
2. (15%) What's the Mean Square Error of your prediction and ground truth (prediction=model(x_test), ground truth=y_test)
    a. 0.06877978498537816
3. (15%) What're the weights and intercepts of your linear model?
    a. weight: 0.8169229484250446
    b. intercept: 0.7837563476889158
4. (10%) What's the difference between Gradient Descent, Mini-Batch Gradient Descent, and Stochastic Gradient Descent?
    a. Batch Gradient Descent

        i. is a variation of the gradient descent algorithm that calculates the error for each example in the training dataset, but **only updates the model after all training examples have been evaluated.**

    b. Mini-Batch Gradient Descent

        i. is a variation of the gradient descent algorithm that **splits the training dataset into small batches** that are used to calculate model error and update model coefficients.

    c. Stochastic Gradient Descent

        i. is a variation of the gradient descent algorithm that calculates the error and updates the model **for each example** in the training dataset.

References:

a. https://pse.is/R8PVP
b. https://www.geeksforgeeks.org/ml-mini-batch-gradient-descent-with-python/

5. (5%) All your codes should follow the PEP8 coding style and with clear comments

Part. 2, Questions (40%):

1. (20%) Suppose that we have three colored boxes R (red), B (blue), and G (green). Box R contains 3 apples, 4 oranges, and 3 guavas, box B contains 2 apples, 0 orange, and 2 guavas, and box G contains 12 apples, 4 oranges, and 4 guavas. If a box is chosen at random with probabilities p(R)=0.2, p(B)=0.4, p(G)=0.4, and a piece of fruit is removed from the box (with equal probability of selecting any of the items in the box), then what is the probability of selecting guava? If we observe that the selected fruit is in fact an apple, what is the probability that it came from the blue box?

	apples	oranges	guavas	total
Red	3	4	3	10
Blue	2	0	2	4
Green	12	4	4	20

$$(1) \quad \tfrac{3}{10} \times 0.2 + \tfrac{2}{4} \times 0.4 + \tfrac{4}{20} \times 0.4 = 0.34$$

$$(2) \quad \tfrac{2}{4} \times 0.4 \,/\, (\tfrac{3}{10} \times 0.2 + \tfrac{2}{4} \times 0.4 + \tfrac{12}{20} \times 0.4) = 0.4$$

2. (20%) Using the definition $var[f] = E[(f(x) - E[f(x)])^2]$ show that $var[f(x)]$ satisfies $var[f] = E[f(x)^2] - E[f(x)]^2$

3. $var[f] = E[(f(x) - E[f(x)])^2]$

$$= E[f(x)^2 - 2f(x) \cdot E[f(x)] + E[f(x)]^2]$$

$$= E[f(x)^2] - 2E[f(x) \cdot E[f(x)]] + E[E[f(x)]^2]$$

$$= E[f(x)^2] - 2E[f(x)] \cdot E[f(x)] + E[f(x)]^2]$$

$$= E[f(x)^2] - E[f(x)]^2$$