Data Engineering 414 ~ Practical 2: Linear Regression in Python

Johan Neethling 24739286

2024 - 03 - 04

Introduction

In this practical we were given skeleton code and data from the MNIST dataset. We were asked to finish implementing a linear regression model. We defined weights for our training data, completed the model framework and tested the model's accuracy.

Report Overview

The practical was the next step towards understanding machine learning algorithms in Python.

I had a few problems with data shapes and the weights inputs, but with the help of one of the delightful demi's we managed to find a work around which led me to the correct answer.

To fix bugs, I implemented print statements that showed me where my code got stuck (numbers) and the shape of the data being used in the model. I left the print statements, but in comments to display to you how I moved through the project.

Report

Question 1

We were given two functions to load in the data.

I loaded the data in with the npv function as it is much faster than it's csv alternative, which allowed me to be much more efficient with the rest of the practical.

```
90   #************************ YOUR CODE HERE ***********************#
91
92   train_X, train_y = read_mnist_npz("data/train.npz")
93   test_X, test_y = read_mnist_npz("data/test.npz")
94
95   print("All files loaded in form X and y:")
96   print("test_X shape: ", test_X.shape)
97   print("test_y shape: ", test_y.shape)
98   print("train_X shape: ", train_X.shape)
99   print("train_y shape: ", train_y.shape)
100  print("")
101
102  #***************************************************************#
```

Question 2

We implemented the rest of the Linear Regression model.

Part a

I implemented the optimal weights theorem to train the model.

I had to add in the row of bias to the data, as shown below.

```
229          #************************ YOUR CODE HERE ***********************#
230
231          X = np.insert(X, 0, 1, axis=1)
232          #print(X.shape)
233          self.weights = np.linalg.inv(X.T@X)@X.T@Y
234
235          #***************************************************************#
```

Part b

For the forward function, I also had to change the shape of the input data by adding a column of ones. The returned value is just the data multiplied by the weights.

```
190         #************************ YOUR CODE HERE ***********************#
191
192         X = np.insert(X, 0, 1, axis=1)
193         #print("In forward: ", X.shape)
194         #print("Self.weights shape:", self.weights.shape)
195         #print("")
196         return X@self.weights
197
198
199         #***************************************************************#
```

Part c

We had to create an instance of the linear regression model (lr) and trained it on the training data.

```
271     #************************ YOUR CODE HERE ***********************#
272     #print("-1")
273     lr = LinearRegression(784, 10)
274     #print("0")
275     lr.train_normaleqs(train_X, train_y)
276     #print("0.5")
277     #***************************************************************#
```

Question 3

We tested the accuracy of the model with a predetermined accuracy function.

```
304     #************************ YOUR CODE HERE ***********************#
305
306     #print("1")
307     model_outputs = lr.forward(test_X)
308     #print("2")
309     model_accuracy = accuracy(test_y, model_outputs)
310     print("Model accuracy = ", model_accuracy)
311
312     #***************************************************************#
```

Question 4

We implemented our own error function that compares the variance between the real and predicted outputs of the testing data.

```
318  #************************ YOUR CODE HERE ************************#
319
320  def error(y, y_hat):
321
322      diff_sqr = (y-y_hat)**2
323      sum_diff_sqr = np.sum(diff_sqr)
324
325      return sum_diff_sqr/len(y)
326
327  print("Error of Linear Regression model: ", error(test_y, model_outputs))
328
329  #************************************************************#
```

Question 5

In this question we got a peak into gradient descent for linear regression models.

Part a

Implemented another linear regression model (gd) and trained it using gradient descent.

```
333  #************************ YOUR CODE HERE ************************#
334
335  gd = LinearRegression(784, 10)
336  gd.train(train_X, train_y)
337
338  #************************************************************#
```

Part b

Once again compared the error of this model instance.

```
344  #************************ YOUR CODE HERE ************************#
345
346  err_2 = error(train_y, gd.forward(train_X))
347  err_1 = error(test_y, gd.forward(test_X))
348  print("Error in training: ", err_2)
349  print("Error in testing: ", err_1)
350
351  #************************************************************#
```

Conclusion

We had a productive practical period that is adding to my knowledge and comfort of the work.

Much appreciated.