Neural Network and Deep Learning: Assignment 1

Submitted To:

Dr.Alioune Ngom

Submitted By

Aayushi Navinchandra Patel (Student ID:110087817)

Aaditya Pradipbhai Parekh (Student ID:- 110084734)

Dhruvkumar Arvind Patel (Student ID:- 110055817)

Mrinal Walia (Student ID:- 110066886)

Date: 23/May/2022

GitHub Link: https://github.com/abhiwalia15/COMP8610_Assignment_1

Question_1: In this question you are to create some simulated data sets and then use *Ordinary Least Square Regression* to perform some prediction. Use whatever programming language you want to use.

Generate 5000 synthetic data points (x, y) as follows:

- **Using the rnorm() function in R (or equivalent in Matlab or Python or etc), create a vector, x, containing 5000 observations drawn from a Gaussian distribution N(0, 1) [ie, a normal distribution with mean 0 and variance 1]. This vector x represents your set of inputs x.**

- **Using the rnorm() function in R (or equivalent in Matlab or Python or etc), create a vector, eps, containing 5000 observation drawn from a N(0, 0.25) distribution; ie, a normal distribution with mean 0 and variance 0.25.**

- **Using vectors x and eps, generate a vector y according to the following model**

$$y = -1 + 0.5x - 2x2 + 0.3x3 + eps.$$

Your 5000 data-points (x, y) are generated upon completion of this last part. Note that the true function is a cubic function with true weight vector being

$$wtrue = (w0, w1, w2, w3) = (-1, +0.5, -2, +0.3).$$

- **Implement Ordinary Least Squares method (for linear regression) given the dataset you have created above, for predicting the value of any given input x. Use the whole data set as your training set; no test set. Perform cross-validation (LOOCV or 10-fold-cv) of your choice but without a test set.**

- **Repeat the above with cross-validation method of your own choice (LOOCV or 10-fold-cv) to find the best polynomial degree d which yield best accuracy. You must first randomly create a test set of size between 20% and 30% drawn from your original full data set. If this is correctly done, your methods should not only find d = 3 to be the best degree, but they should also find the best weight vector, wbest, to be as close as possible to wtrue = (-1, +0.5, -2, +0.3).**

- **Compare your results of OLS with or without cross-validation scheme**

- **Use your creativity and do whatever experiments you want to test, and then tell me whatever story your experiments told you.**

Solution:

Here is the solution of the first question with codes and screenshots.

1) Here we create a vector x using the rnorm() function in python, and having 5000 observation.

```
x = np.random.normal(loc=0, scale=1, size=5000)
```

2) Now we create a vector eps using the same rnorm() method containing 5000 observation.

```
eps = np.random.normal(loc=0, scale=0.25, size=5000)
```

3) Now, using vector x and eps, we generate our vector y.

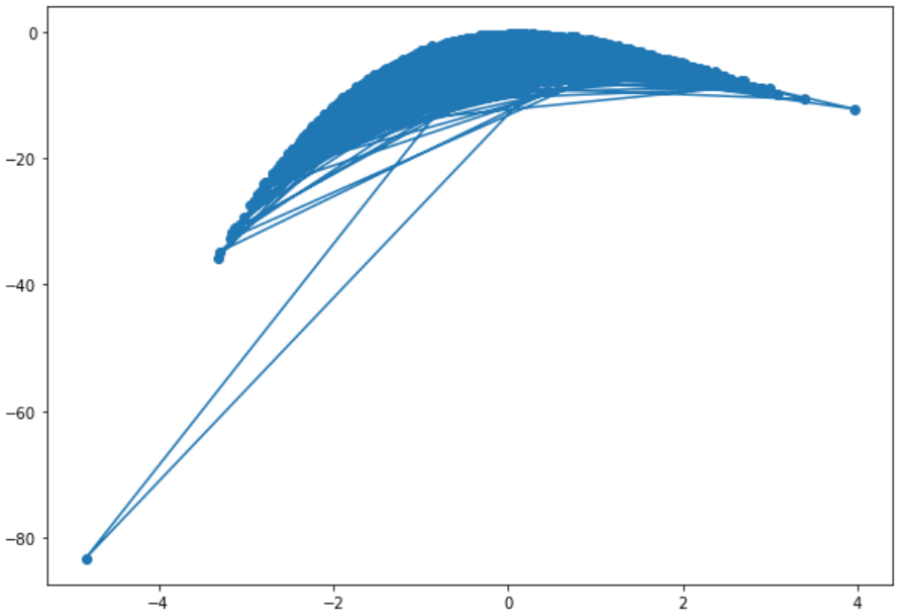
y = -1 + 0.5x – 2x2 + 0.3x3 + eps.

```
y = -1+0.5*x-2*(x**2)+0.3*(x**3)+eps
```

Output:

```
[ 0.14174648  0.4352657  -0.33750165 ... -0.37290923  2.21035671
  1.05374669]
[-3.91267525 -3.81981284 -4.06480856 ... -4.46748062 -3.22505826
 -3.15443227]
```

Figure 1: Plotting 5000 observation



4) Implementing OLS (ordinary Least Square) regression method and predicting the value of any given input x

```
# OLS implementation
x = sm.add_constant(x)
result = sm.OLS(y, x).fit()
result.predict([[x[1]]])
```

Output:

Predicted value for x(1): array([[-3.85826726]])

Figure 2: Implementation of OLS on dataset

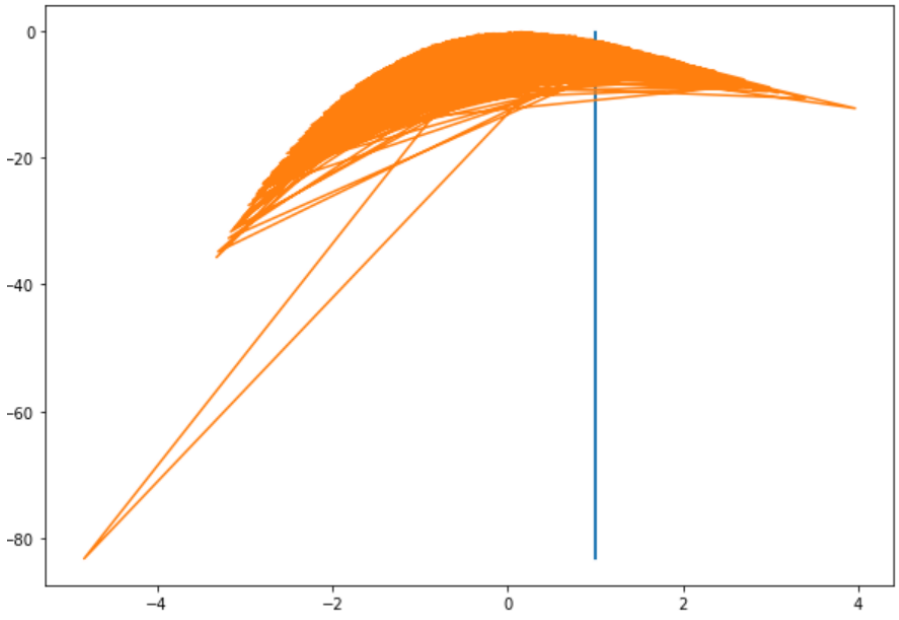


Figure 3: Summary of OLS regression

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.177
Model:                            OLS   Adj. R-squared:                  0.177
Method:                 Least Squares   F-statistic:                     1075.
Date:                Mon, 23 May 2022   Prob (F-statistic):          9.64e-214
Time:                        06:30:13   Log-Likelihood:                -12737.
No. Observations:                5000   AIC:                         2.548e+04
Df Residuals:                    4998   BIC:                         2.549e+04
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         -2.9871      0.044    -68.319      0.000      -3.073      -2.901
x1             1.4370      0.044     32.788      0.000       1.351       1.523
==============================================================================
Omnibus:                     5422.540   Durbin-Watson:                   2.054
Prob(Omnibus):                  0.000   Jarque-Bera (JB):          1135060.852
Skew:                          -5.094   Prob(JB):                         0.00
Kurtosis:                      76.106   Cond. No.                         1.01
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

5) Performing 10-fold-cv cross validation method on whole dataset

```
# cross validation 10-fold-cv
ols2 = LinearRegression()
ols_cv_mse = cross_val_score(ols2, x, y, scoring='neg_mean_squared_error', cv=10)
ols_cv_mse.mean()
```

Output:

```
-8.35615645415144
```

6) Creating a test set of size between 20% and 30 % from original method and performing 10-fold cv method on it.

```
df_x = panda.DataFrame(x)

df_y = panda.DataFrame(y)

train_x, test_x, train_y, test_y = train_test_split(df_x, df_y, test_size=0.25)

lreg = LinearRegression()

lreg.fit(train_x, train_y)

kf = KFold(n_splits=10)

score = cross_val_score(lreg, test_x, test_y, cv=kf)

print(lreg.score(test_x, test_y))
print("\nCross Validation Scores are :\n{}\n".format(score))
print("Average Cross Validation score :\n{}\n".format(score.mean()))
```

Output:

Average cross validation score on performing 10-fold cv is `0.14279013064770485`

7)Predicting the best weight vector to be as close as true weight

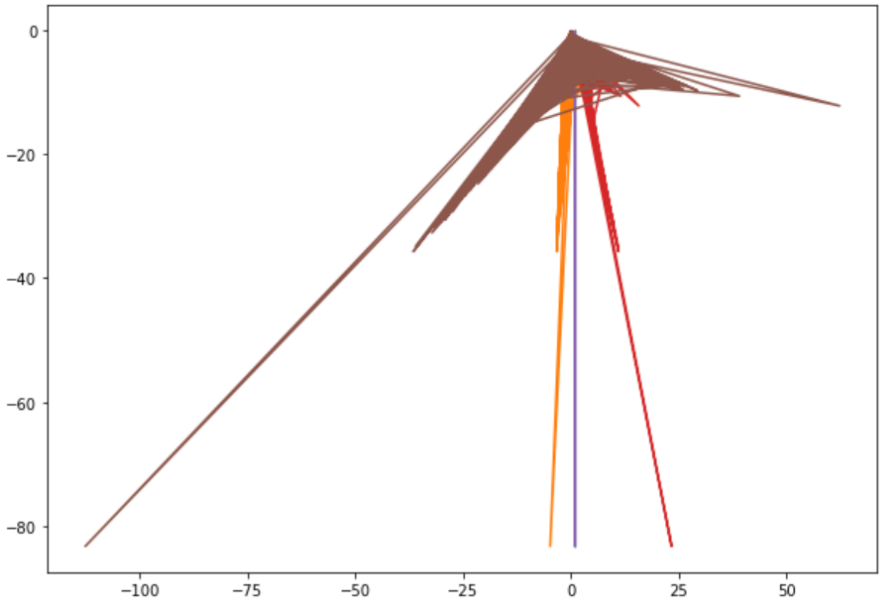
```
X = np.hstack([x, x**2, x**3])
reg = LinearRegression().fit(X, y)

wpred = [reg.intercept_.tolist()]+ reg.coef_.tolist()
print(wpred)
```

Output:

The Best predicted weight is:

```
[-1.00526815854367, 0.5050103881033143, -1.9994560127932701, 0.2981793217286884]
```



7) Comparing the result of OLS with 10-fold cv
The score on implementation of OLS is 0.177 and the mean score after implementation of 10-fold-cv method on same dataset is 0.143

Observation during Experiment:

Using Gaussian distribution, we created 5000 data points using a random normal function. On implementing OLS on the same dataset, we found a predicted value nearer to the actual value.

After that, we implemented polynomial regression on a dataset between 20 and 30% at d=3 to get the best weight as close as true weight, where we found the predicted weight as [-1.006672719524989, 0.4933532632180748, -1.995344105422205, 0.3011635460012912]

This is how we solve the problem by using the OLS method and performing cross-validation.

8) We have tried different training algorithms and learning rates. These changes had drastic effects on the performance of the Ordinary Lease Squares method for linear regression. Hence, after much trial and error, we came to the final values that we found to be optimal, yielding the best results.

There is a difference in OLS configurations with and without the cross-validation scheme. The main difference that we observed was in the score on performance. The performance score with cross-validation was significantly less than the error.

Reference:

- https://github.com/runawayhorse001
- https://scikit-learn.org/0.19/modules/classes.html#module-sklearn.model_selection
- https://sebastianraschka.com/faq/docs