PMDS505P Data Mining and Machine Learning
Experiment 3

January 2025

1 Work to do today

Note: Make a single pdf file of the work you are doing in jupyter notebook. Upload with proper format. Please mention your name and roll no properly with Experiment number in the first page of your submission.

Multiple Linear Regression

Q1. Today we will implement multiple linear regression technique to fit a model in connection with the dataset "Book1.csv" available for you to download in moodle.

- Download the dataset 'Book1.csv' from moodle. This dataset has information regarding the house price and many features depending on it. Open the CSV file and see the different features and the target variable Y (house price) also.
- Load the dataset to a dataframe.
- Drop the furnishing status column and then we will be left out with 5 features $(X_1, X_2, X_3, X_4, X_5)$ to predict the house price (Y)
- We are looking for a model $h_\theta(\mathbf{X}) = \theta_0 + \theta_1 X_1 + \theta_2 X_2 + ... + \theta_5 X_5$.
- USe MinMaxScaler() to scale the data in the range of 0 to 1.
- Split the data into training and testing sets using appropriate functions. Use a 80:20 split.
- Also, use the inbuilt LinearRegression class and create an object of this class and fit the model using training data and check for the values of the parameters of your model. If you print the intercept and coefficients as you did in the previous lab you will get the model parameters $\theta_0, \theta_1, ..., \theta_5$.

Q2. Next write an appropriate gradient descent algorithm and determine the values of the parameters involved in the prediction function.

Notation:

n = number of features
$x^{(i)}$ = input (features) of i^{th} training example.
$x_j^{(i)}$-value of feature j in i^{th} training example.

Here, $h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_4$

we also use $x_0^{(i)} = 1$ for all i

$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \in R^{n+1} \quad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix} \in R^{n+1}$$

$$\begin{aligned} h_\theta(x) &= \theta_0 x_0 + \theta_1 x_1 + \cdots + \theta_n x_n \\ &= \theta^T x. \end{aligned}$$

Hypothesis: $h_\theta(x) = \theta^T x = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$
Parameters: $\theta_0, \theta_1, \ldots, \theta_n$
Cost function:

$$J(\theta) = J(\theta_0, \theta_1, \ldots, \theta_n) = \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)} \right)^2$$

Gradient descent: Repeat {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \ldots, \theta_n) \qquad \}$$

(simultaneously update for every $j = 0, \ldots, n$) New algorithm ($n \geq 1$) : Repeat {

$$\theta_j := \theta_j - \alpha \frac{2}{m} \sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)} \right) x_j^{(i)}$$

(simultaneously update θ_j for $j = 0, \ldots, n$)

that is,

$$\theta_0 := \theta_0 - \alpha \frac{2}{m} \sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)} \right) x_0^{(i)}$$

$$\theta_1 := \theta_1 - \alpha \frac{2}{m} \sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)} \right) x_1^{(i)}$$

....

$$\theta_n := \theta_n - \alpha \tfrac{2}{m} \textstyle\sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)} \right) x_n^{(i)}$$

- Also print the testing error for the model you have created using the test set you have.

Now if you are not able to do it right now take your time and try to finish up by next week. Time being you can move on to implement Stochastic gradient descent and mini batch gradient descent.

Q3. Implement the Linear regression problem what we attempted using the Trainingsetheights200 dataset in the last lab using stochastic gradient descent and mini batch gradient descent and present the results and plots of your model. Also find the testing error in both the cases. (Use the learning rate to be 0.5 and

Hint: you can use random module and random.sample function can be used to generate say 30 (a mini batch) of indices and the derivative can be calculated for only those data points and summed. which can be used to update the model parameters values.

Q4. When we look at the scatter plot of the data in trainingheights200.csv. It can be identified that there is a quadratic nature for the data. So it would be good we can go for such a model.
Let us try to now implement polynomial regression in the case of same dataset trainingheights200.csv. Let us say we want to fit a polynomial of degree 3 to this data we can do so. That is the model we are trying to fit is $h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3$
So here we need to model the variable Y in terms of the features X, X^2, X^3. We can take Y has weight and X as height from the data set.
Here we first make the data ready doing necessary preprocessing steps and do the train test split.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import PolynomialFeatures, MinMaxScaler
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error

data = pd.read_csv("Training_set_heights200.csv")

scaler = MinMaxScaler()
x_scaled = scaler.fit_transform(data)

X_train, X_test, y_train, y_test = train_test_split(
    x_scaled[:, 0].reshape(-1, 1), x_scaled[:, 1], test_size=0.30, random_state=0
)
```

Figure 1: Enter Caption

Now we can generate our polynomial features X, X^2 and X^3 etc. This can be done as given below.

```
poly = PolynomialFeatures(degree=3, include_bias=True)
X_train_poly = poly.fit_transform(X_train)
X_test_poly = poly.transform(X_test)
```

Figure 2:

The bias term is also included in the X_train_poly. You can print and check the values in X_train_poly. Then we can fit the model using the LinearRegression inbuild class and we have the results.

```
model = LinearRegression()
model.fit(X_train_poly, y_train)

y_train_pred = model.predict(X_train_poly)
y_test_pred = model.predict(X_test_poly)

train_mse = mean_squared_error(y_train, y_train_pred)
test_mse = mean_squared_error(y_test, y_test_pred)

print("Training MSE:",{train_mse})
print("Testing MSE:", {test_mse})

print("Polynomial Coefficients:", model.coef_)
print("Intercept:", model.intercept_)
```

Figure 3:

Further we can plot and check the results obtained as below.

```
plt.scatter(X_train, y_train, label="Training Data")
plt.scatter(X_train,y_train_pred, color="red", label="Fitted Polynomial (Degree 3)"
)
plt.xlabel("X_train")
plt.ylabel("y_train")
plt.legend()
plt.title("Polynomial Regression (Degree 3)")
plt.show()
```

Figure 4: Enter Caption