SOLENT UNIVERSITY
SOUTHAMPTON

TEF Silver

INTRODUCTION TO AI (COM727)

(MSc Applied AI and Data Science)

Week 5

Dr Shakeel Ahmad, Assoc Professor in Computing
Course Leader

Practical Activities

SOLENT UNIVERSITY

- Let's create a simple linear regression model and tune it.
- Create a project in PyCharm.
- Download Regression_excercise.py from sol and add this to your project.
- You will see a number of sections (Part 1 to Part 8). For example, Part-1 looks like this.

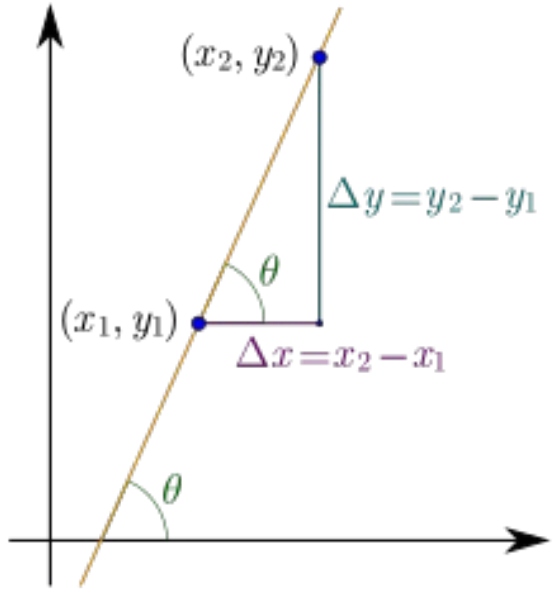
```
import matplotlib.pyplot as plt
months = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]
profit = [52, 74, 79, 95, 115, 110, 129, 126, 147, 146, 156, 184]
plt.plot(months, profit, "o")
plt.title("My Company-ABC")
plt.xlabel("Months")
plt.ylabel("Profit (£)")
plt.show()
```

Part-1 (Plot Data)

- Uncomment Part-1 (select all lines in this part and press control+/) while leaving all other parts commented (alternatively you can copy this part into a new file).
- Run it and examine the output plot, what do you think the profit will be in month 13, add this point to the plot.
- Play with the plot, try changing labels, and the title. Can you add a legend?

Questions:

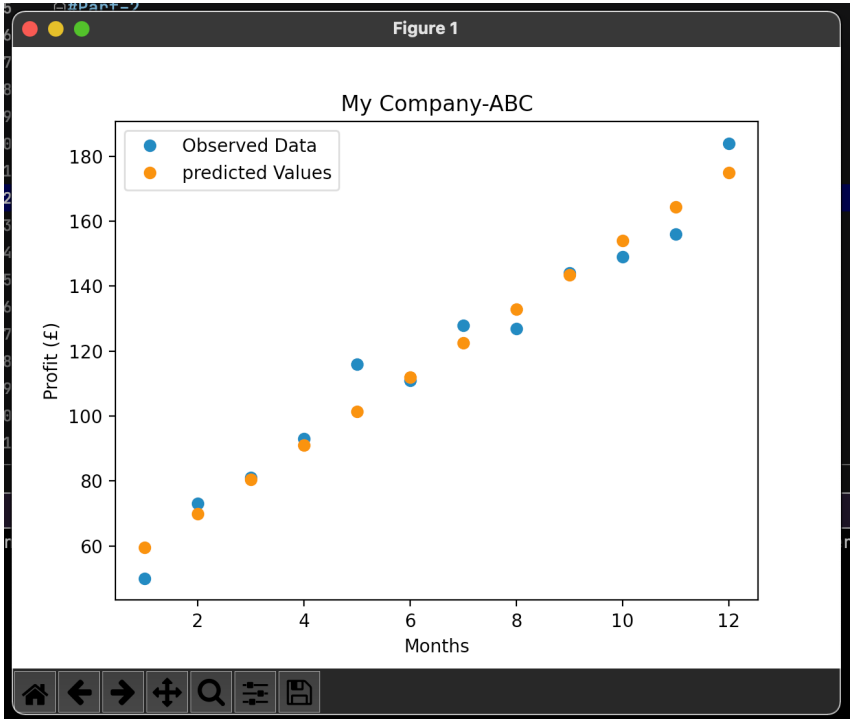
- **Do you think the slope of linear regression is always positive? Explain it. You may review what slope is.**



Part-2 (Use a line to predict data and plot it)

- We cannot make a rough guess, we have to do it mathematically.
- Let's fit the data using a linear regression model, a straight line. Remember line is
  - y=mx+b where m is the slope and b is the Y-intercept.
- Our goal is to find the best values of m and b so that this fits the data with "loss"
- Use the code in Part-1.
- For the time being, assume m=10.5 and b=49, use these values for the model.
- Create a new list y, that has every element in months, multiplied by m and added to b e.g., the first element of y will be equal to month[0]*m + b
- Add a new Plot to the same figure, the new plot will be between months and y (predicted profit)
- Play with m and b to fit the model best.

- **What are the best values for m and b ?**



Part-3 (Use Error Metric – Mean Square Error, and Try it on two given lines.)

- How do we know which values are best for m and b? we need to define what is the best fit.
- We need to calculate the error, commonly known as loss, a best fit will have the least loss value.
- Consider three points (1, 5), (2, 1), and (3, 3) for which we are trying to find the best-fit line.
- Note that we have 3 values for x (1,2,3) and 3 values for y (5,1,3)
- Let's try two lines:
  - Line1: m1=1 and b1=0 and Line2: m2=0.5 and b2=1
- Use line 1 to model the 3 points, and find the Y_predicted1 values using Line1. Create a variable total_loss1 and set this to zero.
- Then, calculate the sum of the squared distance between the actual y-values and the y_predicted1 values using a for loop over the list, and set this equal to total_loss1.
  - Calculating the difference between y and y_predicted1
  - Square the difference
  - Add this to total_loss1
  - Do this for all three points.

- Similarly, use Line2 and calculate total_loss2
- Print out total_loss1 and total_loss2.
- Use the if statement to print which line model is better.

Question: Is there always one line that fits best?

Part-4 (Implement find_gradient() function for m and b)

- Define a function called find_gradient_b() that takes in a set of x values, a set of y values, the y-intercept value b and the slope m, and returns the b_gradient at the given value of b (see the formula, which can be obtained by taking derivative of the loss function with respect to b)

$$\frac{2}{N}\sum_{i=1}^{N} -(y_i - (mx_i + b))$$

- Define a function called find_gradient_m() that takes in a set of x values, a set of y values, the y-intercept value b and the slope m, and returns the m_gradient at the given value of m (see the formula, which can be obtained by taking derivative of the loss function with respect to m)

$$\frac{2}{N}\sum_{i=1}^{N} -x_i(y_i - (mx_i + b))$$

- **Question: Is gradient descent only apply to two dimensions?**

Part-5 (Implement step_function() to find new m and b values at the current m and b values)

- Now we know how to calculate b_gradient and m_gradient
- We need to iterate. To find a new (tweaked) value of b, we would use
  - New_b=current_b – (learning_rate * b_gradient)

- Take the code of part-4 and add a function called step_gradient() that takes in x, y, b_current, and m_current.
- Step_gradient() function will find m_gradient and b_gradient at the current value of m and b, and will return new values of m and b in order to move towards the minimum loss value.
  - new_b=b_current – (0.01*b_gradient)
  - new_m=m_current – (0.01*m_gradient)
- Call Step_gradient() function to find m and b values, and print them.

- **Questions:**
  - **Why m and b values are very different to the ones we chose in Part 2?**
  - **What is the step here?**

Part-6 (Implement gradient_descent() function)

- Combine everything together
- Take the code of Part-5, we had fixed learning_rate but now we will pass it as a parameter to step_gradient() function.
- Add a function gradient_descent() that takes input parameters x, y, learning_rate, and iterations_num. In this function:
  - Create two variables b and m, and set them both to zero. At the end, the function will return the best values of b and m
  - Code a for loop that loops iterations_num times. In each iteration,
    - Call step_gradient() by passing x, y, learning_rate and current values of b and m.
    - Update values of b, m to those returned by step_gradient()
- In your main programme (outside of any function)
  - Define two variables learning_rate=0.01 and iteration_num=1000
  - Call gradient_descent() function by passing x, y, learning_rate, iteration_num. The function will return the optimal values of b and m.
  - Use the returned value of b and m to calculate the profit_predicted values.
  - Plot the original months Vs profit
  - Add another plot of months Vs profit_predicted
  - Add appropriate labels and legends to the plot.
- **Question:**
  - **What happens if there are any outliers?**
  - **How to treat outliers, so that it has minimal effect on the slope and intercept?**

Congratulations!!

- You have implemented Gradient Descent algorithm yourself and have tested it for the month-profit data. Let's apply this to real data.

Part-7 – (Apply your very own implmented Gradient_Descent alogirthm on a real data set)

- Take code of Part-6,
- Use the following code to get a new data set (instead of month-profit data), don't forget to install pandas package first. Also, download the weight-height.csv from sol page.

```
import pandas as pd
df = pd.read_csv("weight-height.csv")
X_data = df[”Height"]
Y_data = df[”Weight"]
```

- Use learning rate=0.0001 and iterations_num=1000

Part-8 (Use linearRegression() of Sklearn package)

- You have now implemented a linear regression algorithm from scratch and developed an insight into how it works.
- Let's try how to do it efficiently using (sklearn) which has linearRegression() function inside the linear_model module.
- Important steps: (Assume we have X and Y data)
  - Install sklearn package
  - from sklearn.linear_model import LinearRegression as LR
  - line_fitter = LR()
  - line_fitter.fit(X, Y)
  - Y_predicted=line_fitter.predict(X)

- Use this on weight-height.csv and weight-height_full.csv

```
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np

df = pd.read_csv("weight-height_full.csv")
X_data = df['Height'].to_numpy()
Y_data = df['Weight'].to_numpy()

X_data=X_data.reshape(-1,1)
plt.plot(X_data, Y_data, 'o')

line_fitter = LinearRegression()
line_fitter.fit(X_data , Y_data)
Y_data_predict=line_fitter.predict(X_data)
plt.plot(X_data,Y_data_predict)
plt.xlabel("Height")
plt.ylabel("Weight")
plt.legend(['Observed Data','Predicted Data']);
plt.show()
```

Let's Recap what we have learnt

SOLENT UNIVERSITY

- We learnt how we can implement a linear regression algorithm in Python
- We learnt how to measure how well a line fits a given data set , the criteria is using loss function
- The best fit means minimum loss (loss is a general term showing prediction error i.e, how much the predicted data is off the actual data)
- To find the line of best fit, we try to find the Y-intercept value b and the slope m of the line that minimizes loss.
- Convergence refers to when the parameters stop changing with each iteration.
- Learning rate refers to how much the parameters are changed on each iteration.

- We also learnt how to use sklearn package to perform linear regression.