

Lab Assignment 4

Linear Regression

PART A : Prerequisite for linear regression implementation

1. Create an array $x = [1, 1, 2, 3, 4, 3, 4, 6, 4]$ using numpy. Calculate a function $h(x) = t_0 + t_1 * x$, where $t_0 = 1.2$ and $t_1 = 0.5$, for all values of x and plot a graph with x on one axis and $h(x)$ on another axis.
2. Create two arrays A and B with the following values using numpy array. Let (A_i, B_i) represent a data point with i th element of A and B . $A = [1, 1, 2, 3, 4, 3, 4, 6, 4]$ $B = [2, 1, 0.5, 1, 3, 3, 2, 5, 4]$ Find out the dot product of the vectors. [Hint use numpy `np.dot(a,b)`]
3. Plot a graph marking the data points (A_i, B_i) with A on the X-axis and B on the Y-axis.
4. Calculate Mean Square Error (MSE) of A and B with the formulae where n is the no: of sample data points.

$$MSE = \frac{1}{n} \sum_{i=1}^n (A^i - B^i)^2$$

5. Modify the above equation with the following cost function. Implement as a function with prototype `def compute_cost_function(n,t1,A,B):`

$$J(t_1) = \frac{1}{2n} \sum_{i=1}^n (h(A^i) - B^i)^2$$

Take $h(x) = t_1 * x$ and $t_1 = 0.5$ Modify the above code iterating for different values of t_1 and calculate $J(t_1)$. Try with $t_1 = 0.1, 0.3, 0.5, 0.7, 0.8$. Plot a graph with t_1 on X-axis and $J(t_1)$ on Y-axis. [hint `sum_squared_error = np.square(np.dot(features, theta) - values).sum()` `cost = sum_squared_error / (2*m)`]

PART B : Linear Regression Implementation

1. Linear regression with one variable.
 - a. Generate a new data set from student scores with one feature studytime and output variable average grade = $(G1 + G2 + G3) / 3$
 - b. Load the new data set
 - c. Plot data

- d. Implement linear regression using inbuilt package python Scikit

```
from sklearn.linear_model import LinearRegression

regressor = LinearRegression()

regressor.fit(X_train, y_train) y_pred =

regressor.predict(X_test) ]
```

- e. Implement gradient descent algorithm with the function prototype `def gradient_descent(alpha, x, y, max_iter=1500)`: where alpha is the learning rate, x is the input feature vector. y is the target. Subject the feature vector to normalisation step if needed. Convergence criteria: when no: of iterations exceed max_iter.

[hint `sum_squared_error = np.square(np.dot(features, theta) - values).sum()` cost = `sum_squared_error / (2*m)`]

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) x_j^{(i)} \quad (\text{simultaneously update } \theta_j \text{ for all } j).$$

- f. Vary learning rate from 0.1 to 0.9 and observe the learned parameter.
- g. Draw the contour plot of cost function and simulate the steps of gradient descent.

Example contour for a function

```
xmesh, ymesh = np.mgrid[-2:2:50j, -2:2:50j]

fmesh = f(np.array([xmesh, ymesh]))

plt.contour(xmesh, ymesh, fmesh) def f(x):

return 0.5*x[0]**2 + 2.5*x[1]**2
```

- h. Do simple k-fold and repeated k-fold. Compute error metrics ME, MAE, MSE, RMSE and compare.

PART B : Extra Credit

1. Implement gradient descend for multivariate linear regression to fit data in full data set.
2. Analyze impact of each input variable on the output variable average grade($g1+g2+g3/3$). Plot each input variable vs average grade and analyse the relationship.