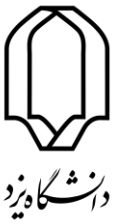
In the name of Allah



Computer Engineering Faculty of Yazd University
Artificial Intelligence

Machine Learning Course

Homework 1 - Part 1

(Linear Regression)

Instructor: Dr. Yazdian

Shiva Zeymaran

Fall 2022

A. Linear Regression with one variable

1. Cost function equation for LR:

$$J(\theta_0, \theta_1) = MSE = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x_i) - y_i)^2$$

Which m is the number of samples and h_θ is:

$$h_\theta(x) = \theta_0 + \theta_1 x$$

2. Fit an LR model on data using three approaches:
Note: Details of this part is available through jupyter notebook. Here, we will shortly talk about each of them.

2.1. Closed Form

After reading x and y of each sample from text file, our training set is a 97x1 matrix; Which means we have 97 samples and one feature for each of them.

For using CF approach, we should first add a single column to each sample with the value of 1. The reason is that we want to find 2 Weights for this problem (As we show in Q1).
Now, it is time to calculate desired thetas using closed form formula:

```
# main part of approach
theta = np.dot(np.linalg.inv(np.dot(X.T, X)), np.dot(X.T, y_sample))
theta_CF = theta
print('The solution for theta are:', theta_CF[0], theta_CF[1])
```

2.2. Stochastic/Online Gradient Descent

In this approach we need to assign some hyper-parameters such as: learning rate and number of iterations or epochs. Then its important to generate a random list of weights. Note that the number of random numbers as weights should be 1 + number of features, which is 2 in this problem.

Now, we can code the main part of GD:

```
J_online = []   # cost values in each epoch
grad = np.zeros(len(W))

# main part of approach
for i in range(iterations):
    y_hat = X@W

    # update parameters after considering each sample
    for j in range(n):   # iterate on samples
        for k in range(len(W)):   # iterate on features (to find all thetas)
            grad[k] = (X[j][k]*(y_hat[j] - y_sample[j]))
            W[k] -= learning_rate * grad[k]

    # calculate cost(J) in each epoch
    J_online.append(np.sum((y_hat - y_sample)**2)/(2*n))

theta_online = W
print('The solution for theta is:', theta_online)
```

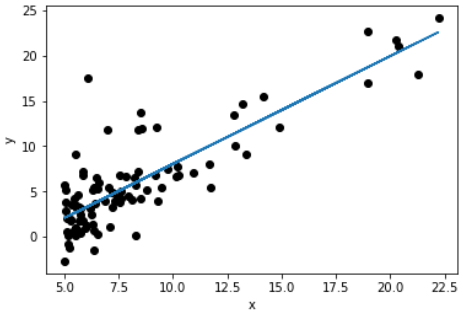
According to the above code, in stochastic GD we should consider each sample in calculating each weight. Therefore, inside of the global loop of GD, we will have another loop for considering each sample and the other nested loop is for calculating each weight by using corresponding feature values.

2.3. Batch Gradient Descent

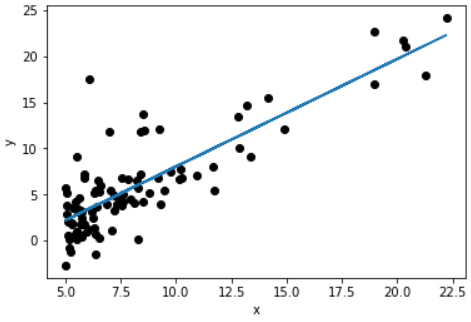
In Batch mode, there is a little difference with previous code. In the main loop of GD we do not iterate on samples and the only loop is for calculating various weights. We should use the value of all samples in each feature to calculate the corresponding weight.

```
J_Batch = []   # cost values in each epoch
grad = np.zeros(len(W))

# main part of approach
for i in range(iterations):
    y_hat = X@W

    # calculate cost(J) in each epoch
    J_Batch.append(np.sum((y_hat - y_sample)**2)/(2*n))


    # update parameters after considering all samples
    for k in range(len(W)):   # iterate on features (to find all thetas)
        grad[k] = (X[:,k]@(y_hat - y_sample))/n
        W[k] -= learning_rate * grad[k]


theta_batch = W
print('The solution for theta is:', theta_batch)
```

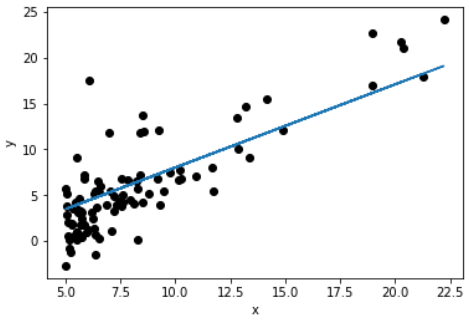
3. Plot dataset samples and each predicted model (using above approaches), separately:



Closed Form



Stochastic GD

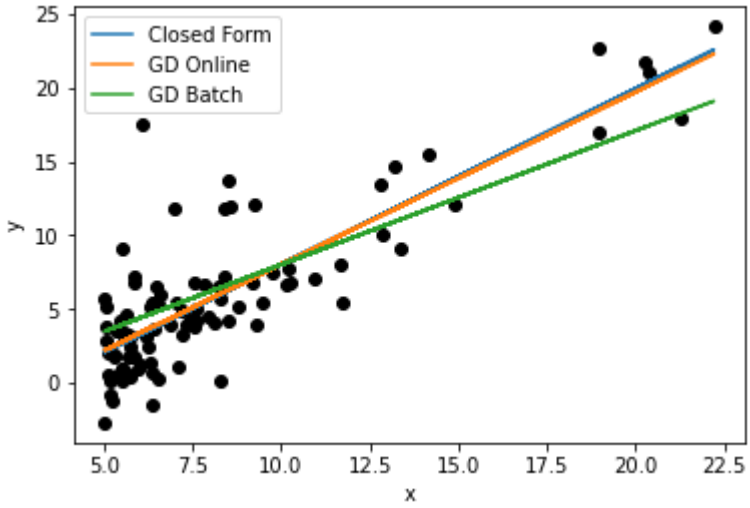


Batch GD

4. Predict y for the following x values using the above models:

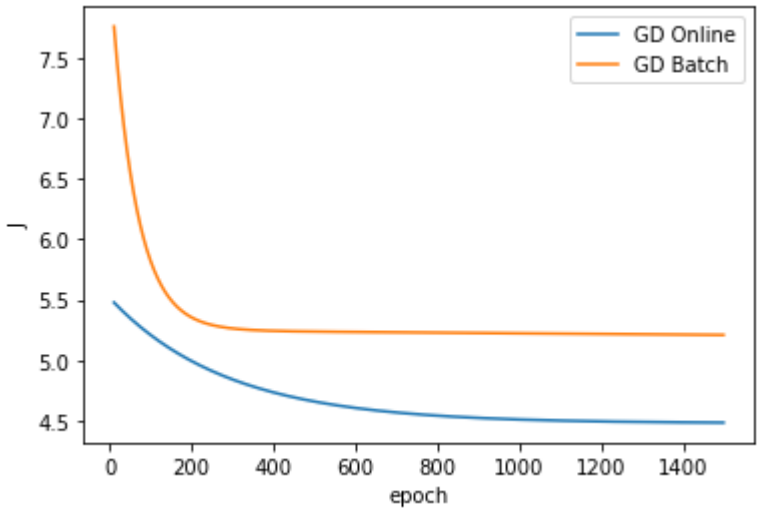
X	***Y Closed Form***	***Y Stochastic GD***	***Y Batch GD***
6.2	3.50102772	3.59316554	4.57025694
12.8	11.37504977	11.30520328	10.56448442
22.1	22.47026266	22.17216556	19.01089587
30	31.89522845	31.40324104	26.18580451

5. Plot all models together:



The comparison between these three different lines shows that the result of the Online GD approach is near to the Closed Form (Which we know that CF is the most accurate one in this problem). The Batch GD result is also a good estimate to model our data but it is a little far from two other approaches.

6. Plot J values along the epochs for Stochastic and Batch GD:
   Note: The plot is started from epoch 10 to 1500 to have better sense of the output diagram.



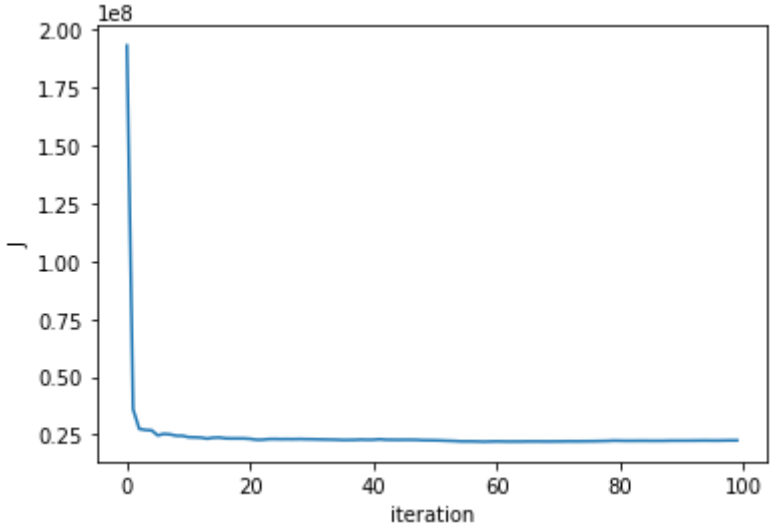
7. It is clear from the previous diagram that **GD Online is better than Batch mode**, here. Since, in online mode we can approximately reach the cost value 4 but in batch mode we can not reach less than 5! Although, the batch mode can converges sooner to the minimum cost.

B. Multiple Variable Regression

Q: Why is OHE better at encoding regional features compared to integer encoding?
A: Because using integer encoding will cause having weight for different values of region! This is not want we want since the different regions shouldn't have different effects for us. They are all the same for calculation purposes.

1. After data preprocessing (extract labels from features, integer encoding for gender and smoke features, one-hot encoding for region, bmi^2 instead of bmi and etc.), we should add a column of one to all samples. Then we can use the formula for Closed Form approach and the result will be 10 weights.

2. After doing the same preprocess we did for train-set on the test-set, we can define a loop with 100 iterations. In each iteration we select 10 more samples of train-set for learning and use them in Closed Form formula. Then we use the weights in each iteration to predict labels for test-set.
   The following plot is the result of calculated costs in each iteration:

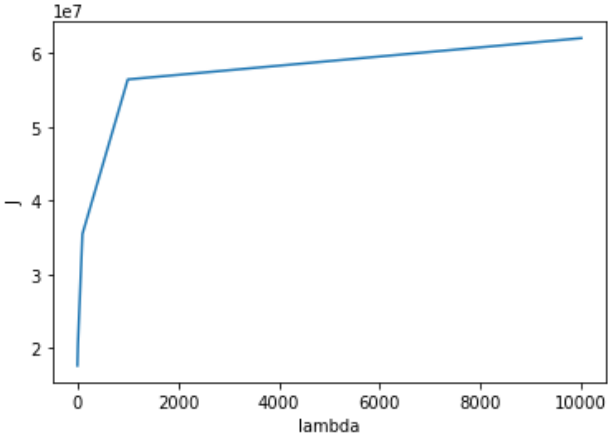


Shows that by increasing the number of samples in each iteration, our model will be more accurate. The cost decreases faster in the first iterations but then it slows down.

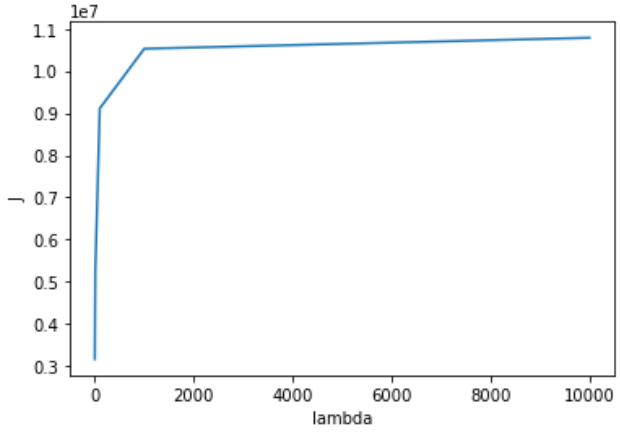
3. The implementation of Online GD and Batch GD are the same as what we did in part A. The details are available in code file. There is the result of each approach for test and train sets (cost values).

	Train Error	*Test Error*
Stochastic GD	1.008877046382688e+149	1.4072653284344393e+148
Batch GD	8.332509858622621e+88	1.1622875418685887e+88

4. According to the calculated costs using each lambda value, lambda = 10^{-4} is the best value because we will have minimum cost.
   The less value for lambda causes the less cost value. This is clear from the following diagram for train set:



   And for test set:



5. Results with Regularization is better than without it.