Assignment 1 Part 1 (Report, Machine Learning)

By **Gargeya Sharma** - MSc Artificial Intelligence - (220278025) – ECS708P

Lab 3: Regression

Q5. What conclusion if any can be drawn from the weight values? How does gender and BMI affect blood sugar levels?

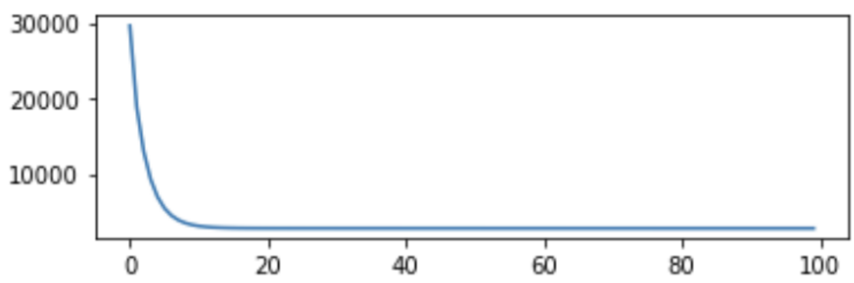
```
tensor([[   1.9400, -11.4488,   26.3047,   16.6306,   -9.8810,   -2.3179,   -7.6995,
            8.2121,   21.9769,    2.6065, 153.7365]])
```

Ans. There are 11 values in the weight matrix shown above, each corresponding to a single feature in the training dataset in addition with bias term (last term). These values are coefficients of their respective feature and determines how and how much they relate with the target variable. Positive values show directly proportional relation while negative values show inverse relation. The scale of these values shows the strength of interdependence while their sign shows direction (as discussed above).

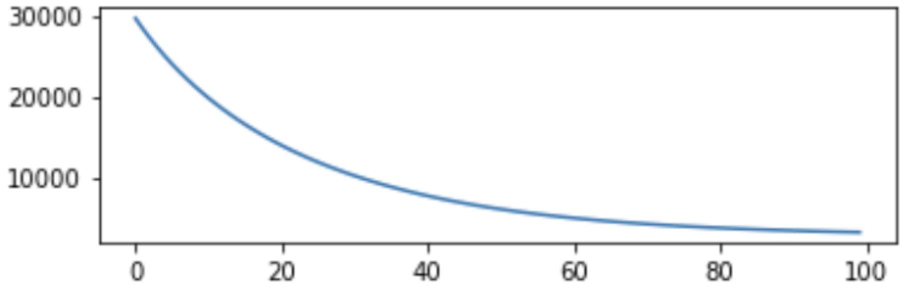
- Gender (value at index: 1) has a respective weight value of -11.448 which signifies that change in a single value of blood sugar level is affected by 11 times the value of gender in an inversely proportional way.
- And for BMI (value at index: 2), it has a respective weight value of 26.304 which signifies that change in a single value of blood sugar level is affected by 26 times the value of BMI in a directly proportional way.

Q6. Try the code with a number of learning rates that differ by orders of magnitude and record the error of the training and test sets. What do you observe on the training error? What about the error on the test set?

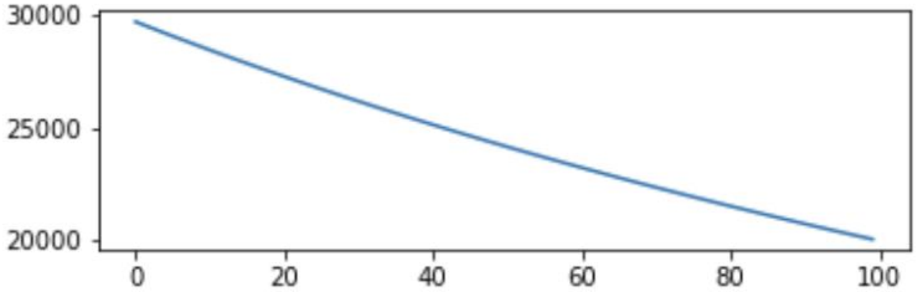
```
Minimum cost at alpha==0.1: 2890.406494140625
```



```
Minimum cost at alpha==0.01: 3356.778076171875
```




```
Minimum cost at alpha==0.001: 20040.58203125
```



Ans. The slope of the training error curve significantly changes to more uniform in nature rather than a sudden drop in the loss value and then staying dead (no optimization) for the rest of the iterations. Reducing the learning rate as the name suggests also slows down the process of learning the cost function optimally as the weights are learned with much smaller step. It might require a greater number of iterations as we move lower on the scale of learning rate to optimize the function because smaller steps would require 10s of times of more iterative step to minimize the error distance than the higher learning rates.

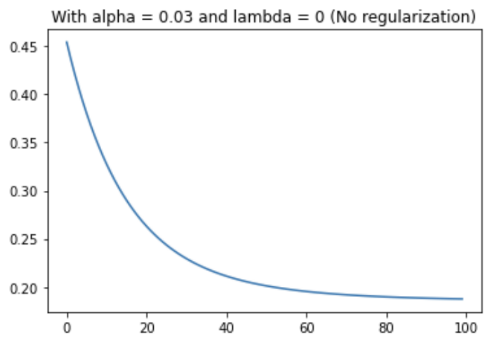
```
{'0.1': {'train error': tensor(2890.4065), 'test error': tensor(2929.9829)},
 '0.01': {'train error': tensor(3356.7781), 'test error': tensor(3005.3503)},
 '0.001': {'train error': tensor(20040.5820),
  'test error': tensor(17473.3184)}}
```

Training error as well as Testing error both are increasing when we are significantly reducing our learning rate. In fact, if the model hasn't learned the generalized representation of the dataset properly then the difference between the training and testing error also seems to increase. This is exactly what is happening in our case, learning rate of 0.1 is giving us an optimal well-fit model with a relatively smaller difference between training and testing errors, while also being the minimum cost values among all the other learning rates.

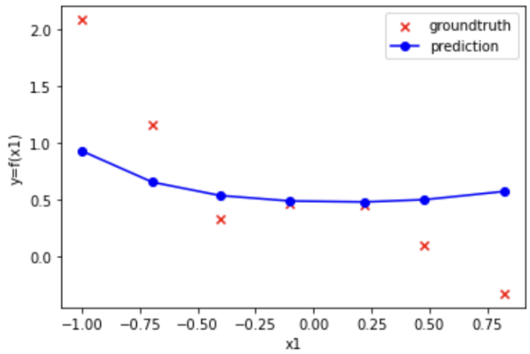
An interesting point to note here is that when our learning rate is 0.001 (100 times smaller than our optimal rate), the testing error is smaller than the model's training error. This happened because the learning was so slow that the model has still got a lot of room of minimizing the mean squared distance between the model's prediction line and ground truth values. When cost was calculated on the test set (fewer values than the training set: 1/5 the size of training set), the aggregate mean squared distance came out to be smaller than of training set.

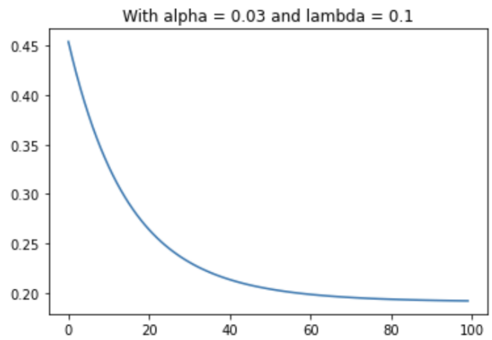
Q8. First of all, find the best value of alpha to use in order to optimize best. Next, experiment with different values of λ and see how this affects the shape of the hypothesis.

Ans. I did the experimentation with different alpha rates like [0.5, 0.1, 0.05, 0.04, 0.03, 0.01] and found that 0.03 as the alpha gives me the satisfactory result with relatively smoother error minimization as well as no scope of further training with the flat/dead line for the last 20-30 iterations.

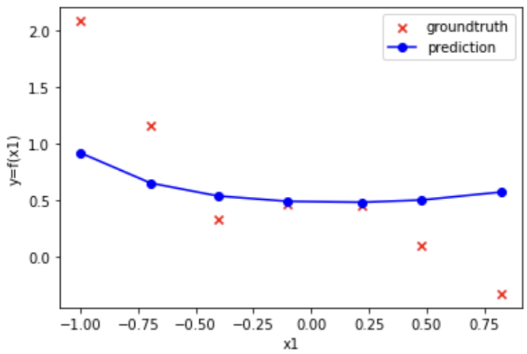


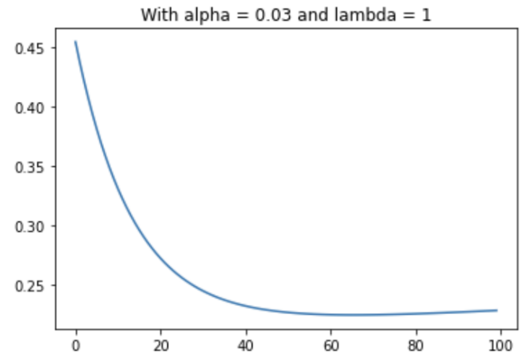

```
Parameter containing:
tensor([[-0.0468,  0.1794, -0.0497,  0.1214, -0.0524,  0.4820]])
Minimum cost: 0.1875617653131485
```

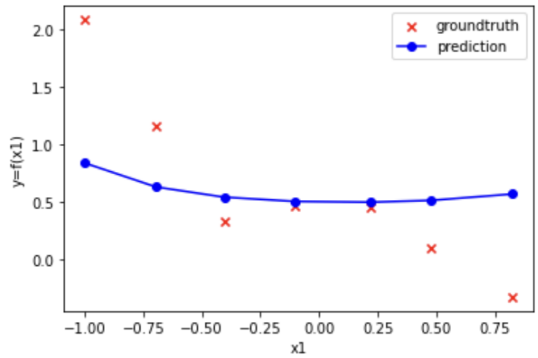





```
Parameter containing:
tensor([[-0.0455,  0.1744, -0.0483,  0.1180, -0.0509,  0.4839]])
Minimum cost: 0.19189567863941193
```

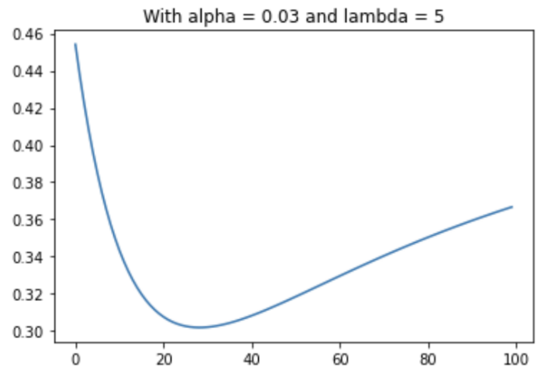


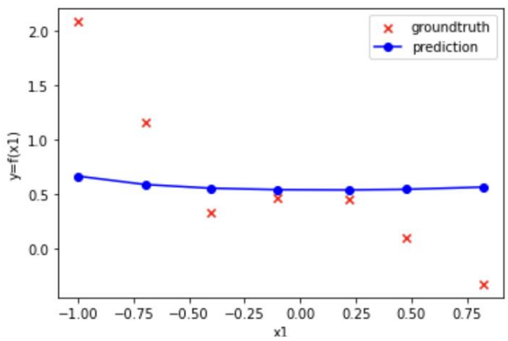





```
Parameter containing:
tensor([[-0.0355,  0.1361, -0.0377,  0.0921, -0.0397,  0.4995]])
Minimum cost: 0.2248290628194809
```

After selecting the learning rate, I experimented for the value of regularizer (lambda) as [0, 0.1, 1, 5] to observe the change in behavior of the hypothesis, learning curve and training error value. Just as we expected, lambda is applying restriction on the cost function to prevent overfitting the data, hence the training error value is increasing with each increasing lambda value but the shape of the hypothesis is barely changing.






```
Parameter containing:
tensor([[-0.0132,  0.0506, -0.0140,  0.0342, -0.0148,  0.5395]])
Minimum cost: 0.3019304573535919
```

This pattern disrupted while we are increasing the lambda value to a relatively large number like ‘5’, such a value disturbed the learning curve with significant increase in error after certain iterations (20-25 iterations) of optimizing the weight for minimum cost. Setting the lambda value so high is causing unwanted increase in the importance of regularization rather than on the mean squared distance between the predicted values and ground truth values, which we meant to minimize to fit the hypothesis to our data. **The shape of the hypothesis flattens out as a line rather than following the nature of exponential features in our training values.**