FoDS Assignment

Om Agarwal, Saransh Dwivedi, Ojashvi Tarunabh

Preprocessing

Given that the scale of all the variables was different, we decided to normalised our data-frame before training any models to bring all the data values in the range 0 to 1. We used the formula

$$X_{normalized} = \frac{X - min(X_i)}{max(X_i) - min(X_i)}$$

We then shuffled the dataset rows randomly using built in functions and selected a 70-30 train-test split to run our models on.

Different models

As a part of this project, we implemented 3 different models –

Polynomial Regression

This model contained no regularization term and hence there was no restriction on the values that the weights could take. The cost function of polynomial regression is as follows:

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{n}(x_i^T\theta - y_i)^2$$

The basic idea behind it is that we start at a random point on the loss function curve and take small steps towards the minimum based on the current errors.

$$\theta_j := \theta_j - \frac{\alpha}{m}\sum_{i=1}^{m}(\hat{y}^i - y^i)x_j^i$$

The size of our steps is determined by a hyperparameter “learning rate”. A higher learning rate means we take bigger steps towards the minimum, but also leaves room for the possibility of overshooting and oscillating around the minimum.

Polynomial Regression with Lasso Regularisation

The problem with plain regression is the model can have a tendency to overfit the training data, not making it generalise well.

$$J(\theta) = \frac{1}{2}(\sum_{i=1}^{n}(x_i^T\theta - y_i)^2 + \lambda\sum_{j=0}^{n}|\theta_j|)$$

To combat this problem, we impose a restriction on the sum of absolute values of the weights. This is done by updating the cost function as follows –

$$\theta_j := \theta_j - \frac{\alpha}{m}(\sum_{i=1}^{m}(\hat{y}^i - y^i)x_j^i + \lambda sgn(\theta_j))$$

The degree of regularisation is determined by the hyperparameter λ

Polynomial Regression with Ridge Regularisation

Another regularisation technique follows the same principle of imposing restrictions on the possible weight values but has a slightly different cost function, given by,

$$J(\theta) = \frac{1}{2}(\sum_{i=1}^{n}(x_i^T\theta - y_i)^2 + \lambda\sum_{j=0}^{p}\theta_j^2)$$

Here, we use the squared value of the weights instead of the absolute value. This is done by updating the cost function as follows –

$$\theta_j := \theta_j - \frac{\alpha}{m}(\sum_{i=1}^{m}(\hat{y}^i - y^i)x_j^i + \lambda\sum_{j=0}^{n}\theta_j))$$

The degree of regularisation is determined by the hyperparameter λ
All these 3 were modelled with both gradient descent and stochastic gradient descent.

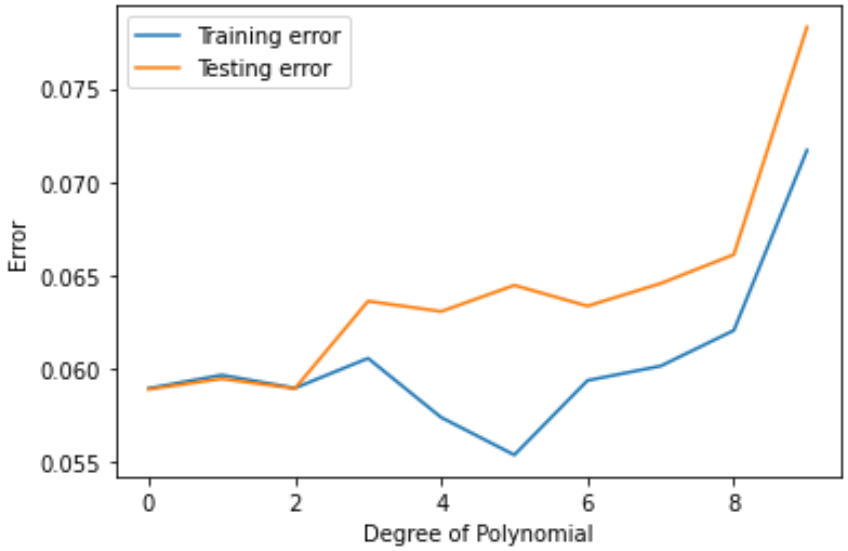
Runs of models

Polynomial regression with no regularization

Here, we ran the polynomial regression model for each degree from 0 to 9 and tabulated the train and test errors at each degree. These results can be seen below –

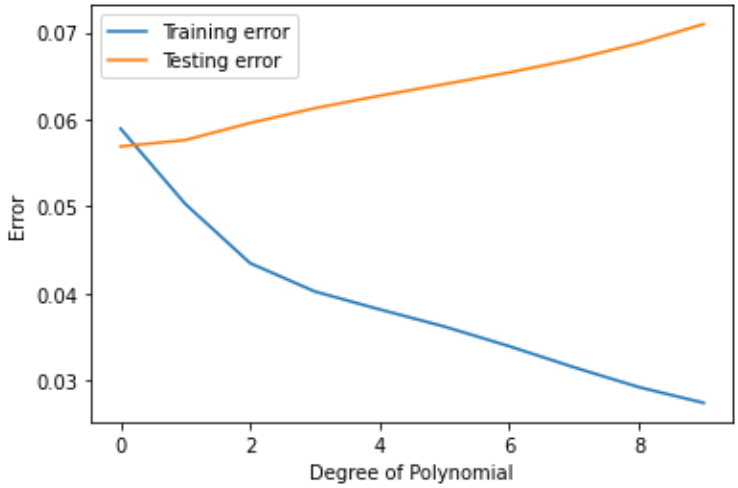
Stochastic Gradient Descent

```
For degree 0, The training error is 0.058952628328976645 and the testing error is 0.056901858735573664
For degree 1, The training error is 0.05966625741134713 and the testing error is 0.0594691041193624
For degree 2, The training error is 0.058968859300459586 and the testing error is 0.05894415244737253
For degree 3, The training error is 0.06056499205582103 and the testing error is 0.05862111495248691
For degree 4, The training error is 0.06462287803009328 and the testing error is 0.059799905990587605
For degree 5, The training error is 0.035391373037451775 and the testing error is 0.06447134556556884
For degree 6, The training error is 0.0593773880696733 and the testing error is 0.06335986163857217
For degree 7, The training error is 0.06014960715693898 and the testing error is 0.06456849953029567
For degree 8, The training error is 0.062056606812284115 and the testing error is 0.06611298597147529
For degree 9, The training error is 0.07169879870746182 and the testing error is 0.07828559683754148
```



Gradient Descent

```
For degree 0, The training error is 0.05895262832405689 and the testing error is 0.05690191713048854
For degree 1, The training error is 0.04291105116820716 and the testing error is 0.05977013626385724
For degree 2, The training error is 0.0498723625470454 and the testing error is 0.05770126674002394
For degree 3, The training error is 0.039532855905616254 and the testing error is 0.06154567352289141
For degree 4, The training error is 0.03739854133340181 and the testing error is 0.06307035868184892
For degree 5, The training error is 0.035391373037451775 and the testing error is 0.06447134556556884
For degree 6, The training error is 0.03394482614862911 and the testing error is 0.06540707853851632
For degree 7, The training error is 0.030798066827536854 and the testing error is 0.06745907345381878
For degree 8, The training error is 0.02863525107902966 and the testing error is 0.06932159098025953
For degree 9, The training error is 0.02695913726483443 and the testing error is 0.07153830167842566
```



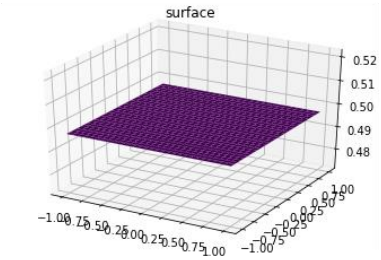
3D Surface Plots

We also visualised the 3D surface plot of our independent variables (Strength and Temperature) with respect to the predicted value (Pressure).

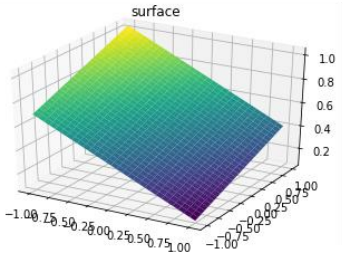
Without Regularisation

Gradient Descent

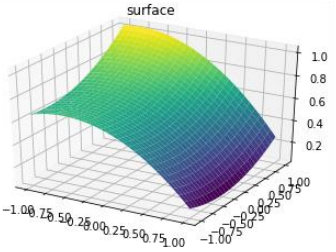
Degree 0



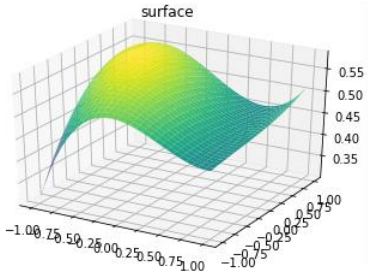
Degree 1



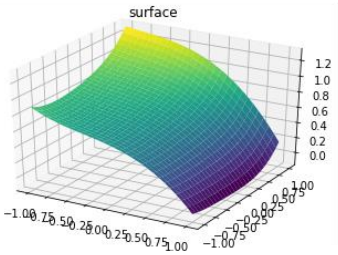
Degree 2



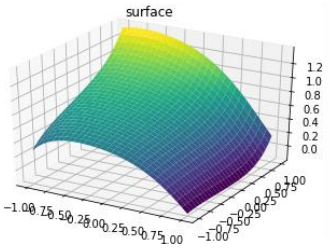
Degree 3



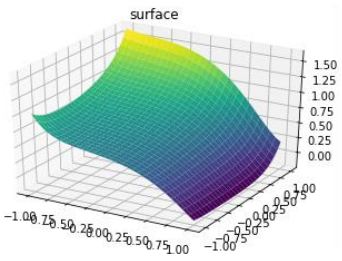
Degree 4



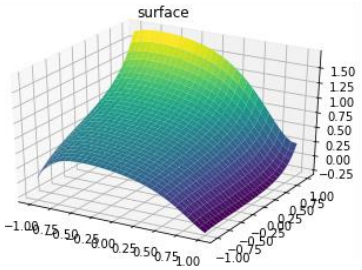
Degree 5



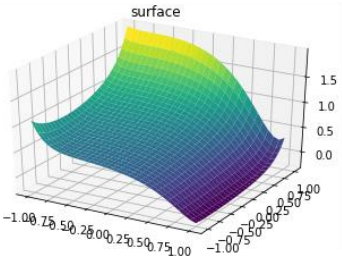
Degree 6



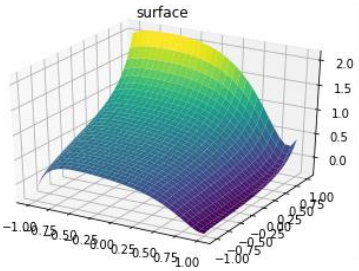
Degree 7



Degree 8



Degree 9

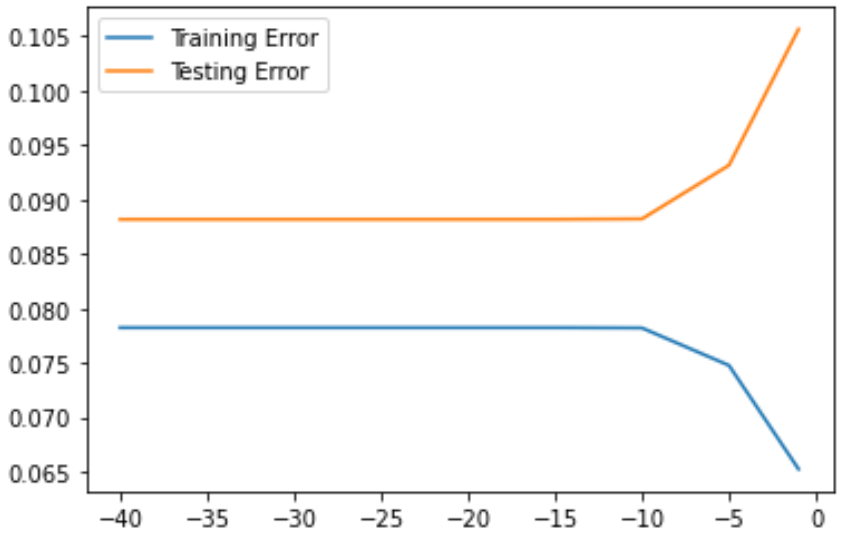


Polynomial regression with ridge regularization

Here, we ran this model for degree 9 polynomial along with 6 different values of lambda to observe how the value of lambda plays a role in training the model. Again, we tabulated the results for train and test errors, and they can be seen as follows –

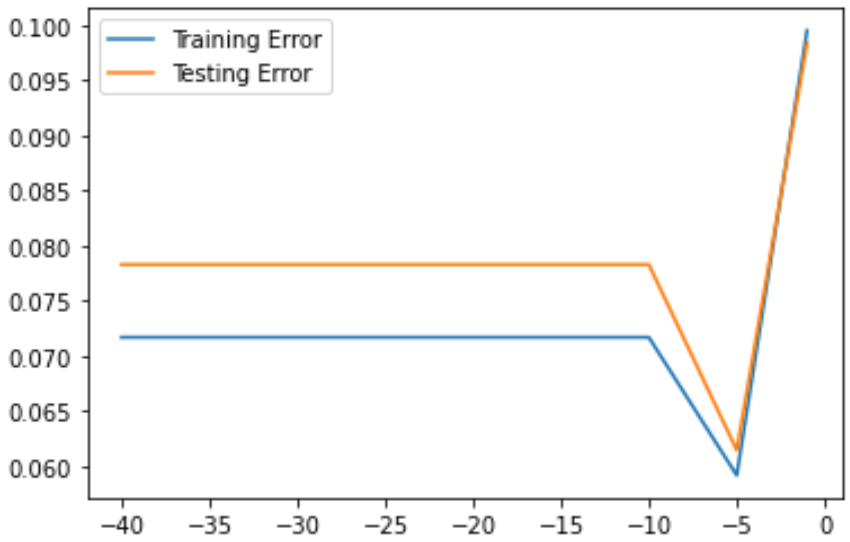
Gradient Descent

```
Value of lambda = e^-1
The training error is 0.06523510932918765 and the testing error is 0.10564810778586653
Value of lambda = e^-5
The training error is 0.07476397655945269 and the testing error is 0.09314651387751167
Value of lambda = e^-10
The training error is 0.07820329591166465 and the testing error is 0.08821928063344782
Value of lambda = e^-15
The training error is 0.07822835312660685 and the testing error is 0.08818070203157316
Value of lambda = e^-20
The training error is 0.07822852205036462 and the testing error is 0.08818044182886374
Value of lambda = e^-40
The training error is 0.0782285231962893 and the testing error is 0.08818044006372631
```



Stochastic Gradient Descent

Value of lambda = e^{-1}
The training error is 0.0983546945653303 and the testing error is 0.09948650132315712
Value of lambda = e^{-5}
The training error is 0.061477099671273336 and the testing error is 0.05919654595938126
Value of lambda = e^{-10}
The training error is 0.07828495534705615 and the testing error is 0.07169826920872237
Value of lambda = e^{-15}
The training error is 0.07828559683109965 and the testing error is 0.07169879870215172
Value of lambda = e^{-20}
The training error is 0.07828559683754142and the testing error is 0.07169879870746175
Value of lambda = e^{-40}
The training error is 0.07828559683754148 and the testing error is 0.07169879870746182

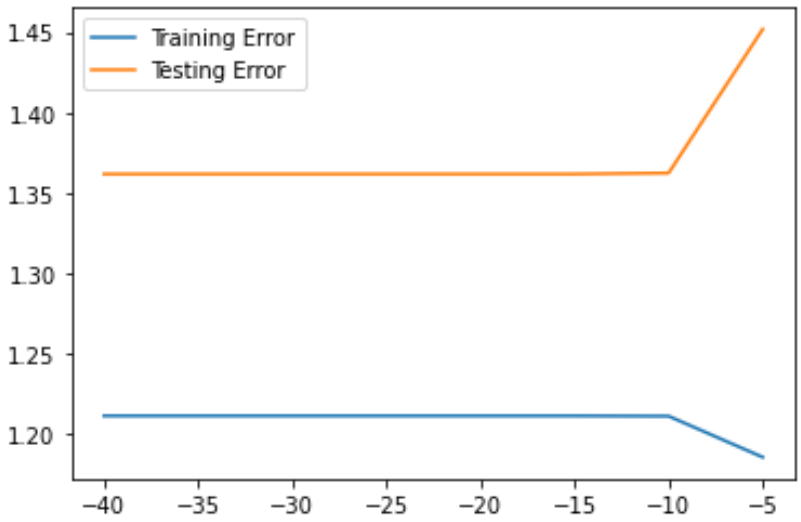


Polynomial regression with Lasso Regularisation

Similarly, we ran this model for degree 9 polynomial along with 6 different values of lambda to observe how the value of lambda plays a role in training the model. Again, we tabulated the results for train and test errors, and they can be seen as follows –

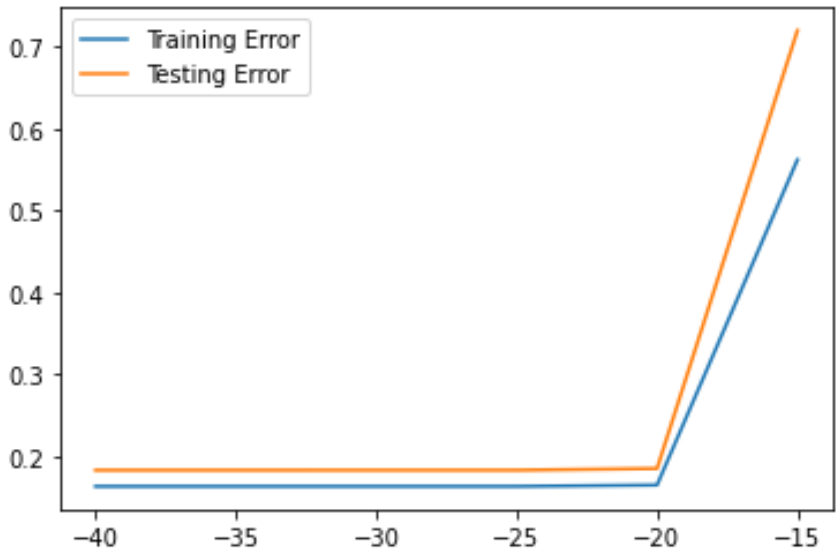
Gradient Descent

```
Value of lambda = e-5
The training error is 1.1857971103929787 and the testing error is 1.4518420182998597
Value of lambda = e-10
The training error is 1.2113231382190246 and the testing error is 1.362415055016698
Value of lambda = e-15
The training error is 1.2114963859804881 and the testing error is 1.3618034961496208
Value of lambda = e-20
The training error is 1.2114975533593715 and the testing error is 1.3617993750795174
Value of lambda = e-25
The training error is 1.2114975612251118 and the testing error is 1.3617993473119476
Value of lambda = e-40
The training error is 1.2114975612784686 and the testing error is 1.3617993471235805
```



Stochastic Gradient Descent

Value of lambda = e^{-5}
The training error is 59832084.246320225 and the testing error is 61497130.74759399
Value of lambda = e^{-10}
The training error is 2757.75054310437 and the testing error is 2883.240402988471
Value of lambda = e^{-15}
The training error is 0.5621027931776927 and the testing error is 0.7196716047867808
Value of lambda = e^{-20}
The training error is 0.16536721121205167 and the testing error is 0.185522722203134
Value of lambda = e^{-25}
The training error is 0.16374383844441665 and the testing error is 0.18347837223483657
Value of lambda = e^{-40}
The training error is 0.16373287891860902 and the testing error is 0.1834645531671622



Conclusion

• The training errors on average tend to decrease as we increase the degree of our equation as our model has more information to use to be able to accurately fit the training data. However, as we increase the degree of our equation, we run the risk of overfitting (memorising the training data). This would lead to low training errors but increasing test errors.

• Regularization is necessary in such a case as with limited training examples there is a high risk of overfitting the data.

• The regularization parameter lambda determines the extent to which we curtail the freedom of the weights. A very small lambda imposes less restrictions on them which could lead to overfitting, whereas a large value of lambda would impose very heavy restrictions on the weights, thus resulting in underfitting.

• Lasso regularization tends to perform better when there are a small number of significant parameters, and the others are close to zero. Ridge works well when there are many large parameters of around the same value.