# Assignment 1-B

We have to fit various regression models to a dataset containing 546 samples, where the -th sample is of the form:

where are the features and is the target value.

We create a 80-20 split of the dataset for training and testing respectively.

### Structure of matrices

Our feature matrix is stored as:

and the target vector is stored as:

# POLYNOMIAL REGRESSION

### Our model

where for all training examples, and correspond to the features in dataset and the remaining ’s refer to higher degree terms found by multiplying and .

### Normalization

We calculate the mean () and standard deviation () of the training data. We normalize training data as:

We also store and vectors to normalize the testing data using the same and .

### Generating Feature Matrix

We generate feature matrix for polynomial regression using nested for loops. We generate a matrix with all the polynomial features till degree 9, and later slice rows from this matrix to run polynomial regression for lower degrees.

Our generated feature matrix is as follows:

### Cost Function

We calculate cost function for regression as follows:

### Error Function

The error function will be the of the difference between the predicted value and the given value

### Gradient Descent

We implement gradient descent by updating the weights for a fixed number of iterations:

where is the learning rate, and

Which in matrix form will be

$$ $$

### Stochastic Gradient Descent

We implement stochastic gradient descent similarly, the cost and its gradient is calculated with respect to a randomly chosed sample (instead of all samples) on every iteration.

Cost function for SGD:

$$$$

Gradient of cost function for SGD:

Finally Stochastic Gradient Descent looks like -

is chosen randomly from for every iteration.

### Regression

The function takes three parameters

$ Y $ The target attribute matrix

$ X $ The complete feature matrix

$ descent\_type $ Specifies wether gradient descent or stochastic gradient descent has to be performed

The function, iterates through degrees to and finds the weights by performing the appropriate descent on the given data.

The errors through each iteration is stored in which is then plotted. The final errors for each degree is stored in

The final weights for each degree is stored in which is returned along with

##### Getting the feature matrix for degree

The feature matrix passed to the function, contains features upto degree . The feature matrix is created in such a way that the first rows of , give the feature matrix for degree

### Testing

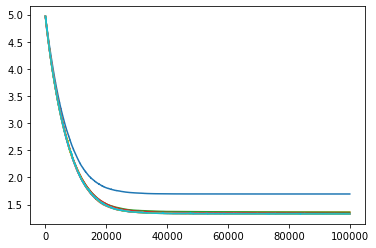
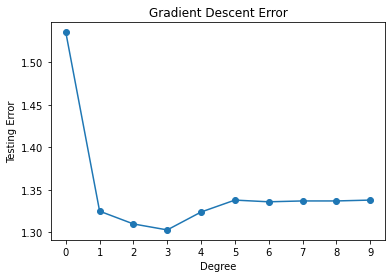
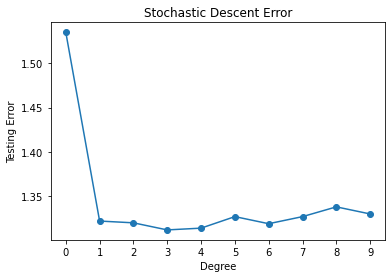
The funcion takes arguments

The target attribute for the testing data

The complete feature matrix of the testing data

A -d array. The row of which stores the weights for degree

The final testing error of each degree is stored in which is then returned by the function.

| Degree | Training Error (Gradient) | Training Error (Stochastic) | Testing Error (Gradient) | Testing Error (Stochastic) |
| --- | --- | --- | --- | --- |
| 0 | 1.6944 | 1.6944 | 1.534838 | 1.534624 |
| 1 | 1.3606 | 1.3608 | 1.324867 | 1.323926 |
| 2 | 1.3325 | 1.3570 | 1.309989 | 1.318076 |
| 3 | 1.3269 | 1.3370 | 1.303107 | 1.309406 |
| 4 | 1.3187 | 1.3306 | 1.323781 | 1.316062 |
| 5 | 1.3101 | 1.3302 | 1.338475 | 1.325186 |
| 6 | 1.3055 | 1.3295 | 1.336455 | 1.323378 |
| 7 | 1.3022 | 1.3291 | 1.337395 | 1.330144 |
| 8 | 1.2991 | 1.3280 | 1.337464 | 1.334542 |
| 9 | 1.2970 | 1.3271 | 1.338169 | 1.335491 |

Since the testing error is the least for degree 3, we can conclude that polynomial regression of degree 3 best fits the given dataset.

# REGULARIZATION

### Regularized Cost Function

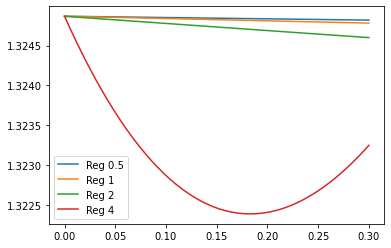
We calculate cost function for regularized linear regression as follows:

The bias, is not regularized.

### Regularized Linear Regression

We implement regularized linear regression by updating the weights for a fixed number of iterations:

where is the learning rate, and



regularization

| Type | Training RMS | Testing RMS |
| --- | --- | --- |
| Unregularized Linear | 1.360600 | 1.324867 |
| Unregularized Cubic | 1.326900 | 1.303107 |
| Regularized 0.5 | 1.375088 | 1.307830 |
| Regularized 1 | 1.375890 | 1.308257 |
| Regularized 2 | 1.365160 | 1.320221 |
| Regularized 4 | 1.362957 | 1.322391 |

##### Comparison between the best regularized and the best non-regularized model

Of all the regularized models, the one where q is 0.5 performs the best in testing data. Comparing this with the best performing non-regularized polynomial regression model (cubic regression) we can see that cubic regression performs better.

This can be explained as non-regularized linear regression does not overfit the training dataset hence adding regularization does not improve the performance of the model significantly.

# Surface plots of regression models

