CS F320 Assignment 3

Abhinav Sukumar Rao[1], Pratyush Banerjee[2], and Rohan Daniel[3]

[1]2018A7PS0172H
[2]2018A7PS0312H
[3]2018A7PS0584H

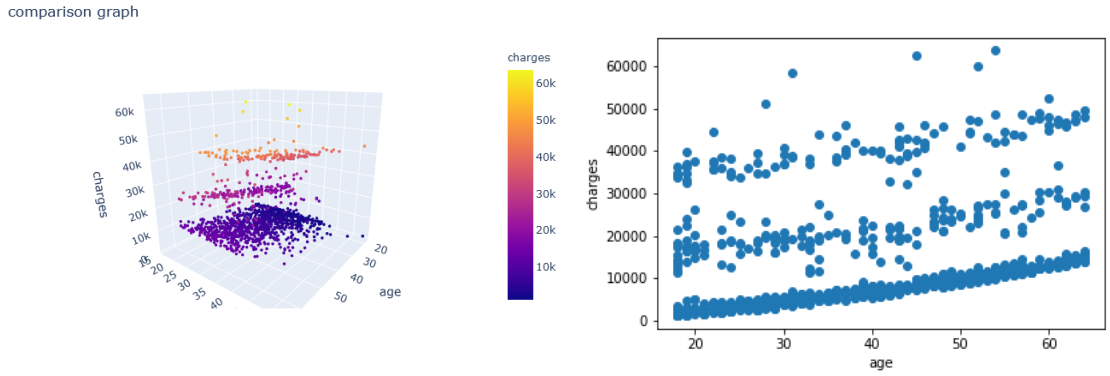
1 Introduction and Overview

This is our report on the implementation of Polynomial Regression of degrees 1-10 with 2 types of regularization i.e. a)Lasso regression b)Ridge regression.

2 Pre-Processing

The independent variable consists of three features 'age', 'bmi' and 'children'. The 'children' column was dropped as specified in the assignment. The target variable is 'charges'. We plotted 'charges' against the individual features to look for any correlation.

We constructed matured polynomial features using the PolynomialFeatures class



For feature scaling we normalized all the features and the the target variable. We decided to use this instead of standardization because the normalized data is between [0,1] and hence wont explode upon higher powers.

We then created a random 70-20-10 split to aid in training,validation and testing.

3 Model

$y = f(X, W, b, d)$ is a polynomial with two variables of degree d.

e.g. For degree 2

$$y = b + w_1x_1 + w_2x_2 + w_3x_1^2 + w_4x_1x_2 + w_5x_2^2$$

3.1 Gradient Descent

Gradient descent is an iterative optimization algorithm that minimizes a function by moving in the direction of negative of gradient. All the training examples are used at once to calculate the gradient.

3.1.1 Implementation

```
def ridge_GD(W, b, X, y, lamb):
    grad_w = (X.T.dot(X.dot(W) + b - y) + np.sum(lamb*(W)))
    grad_b = np.sum(X.dot(W) + b - y)
    return grad_w, grad_b

def lasso_GD(W, b, X, y, lamb):
    grad_w = (X.T.dot(X.dot(W) + b - y) + np.sign(lamb*(W)))
    grad_b = np.sum(X.dot(W) + b - y)
    return grad_w, grad_b

def GD(X, y,reg="ridge", alpha = 0.01, epochs = 1000):
  W = np.random.rand(X.shape[1], 1) * 0.01
  b = 1
  if reg == 'ridge':
        grad_GD = ridge_GD
  elif reg == 'lasso':
        grad_GD = lasso_GD
  for j in range(epochs):
      grad_w, grad_b = grad_GD(W, b, X, y, lamb)
      W = W - alpha/len(X) * grad_w
      b = b - alpha/len(X) * grad_b
  return W, b
```

3.2 Stochastic Gradient Descent

Stochastic gradient descent is another iterative optimization algorithm like gradient descent. However, it uses one training example at a time to calculate the gradient. This greatly reduces the computational cost.

3.2.1 Implementation

```
def ridge_SGD(W, b, X, y, lamb):
    X.reshape(-1,1)
    grad_w = np.array(X * (X.dot(W) + b - y) ).reshape(-1,1)  + lamb*(W)
    grad_b = (X.dot(W) + b - y).item(0)
    return grad_w, grad_b

def lasso_SGD(W, b, X, y, lamb):
    X.reshape(-1,1)
    grad_w = np.array(X * (X.dot(W) + b - y) ).reshape(-1,1)  + lamb*np.sign(W)
    grad_b = (X.dot(W) + b - y).item(0)
    return grad_w, grad_b

def SGD(X, y,alpha = 0.01, epochs = 1000):
    W = np.random.rand(X.shape[1], 1) * 0.01
```

```
b = 1
if reg == 'ridge':
    grad_GD = ridge_GD
elif reg == 'lasso':
    grad_GD = lasso_GD
for j in range(epochs):
    i = randint(0,len(X)-1)
    grad_w , grad_b = grad_SGD(W, b, X[i], y[i], lamb)
    W = W - alpha * grad_w
    b = b - alpha * grad_b
return W
```

3.3 Ridge Regression

Ridge Regression is a regularization technique that penalizes the sum of squares of weights so that small values of weights can be obtained. This prevents overfitting.

$$E(w) = \frac{1}{2N}\sum_{i=1}^{N}(y_i - t_i)^2$$

$$\min E(w)\ s.t.\ \sum_{i=1}^{N} w_i^2 \leq \eta$$

$$E(w) = \frac{1}{2N}\sum_{i=1}^{N}(y_i - t_i)^2 + \lambda\sum_{i=1}^{N} w_i^2$$

3.4 Lasso Regression

Lasso Regression is a regularization technique that penalizes the absolute value of weights to prevent overfitting.

$$E(w) = \frac{1}{2N}\sum_{i=1}^{N}(y_i - t_i)^2$$

$$\min E(w)\ s.t.\ \sum_{i=1}^{N} |w_i| \leq \eta$$

$$E(w) = \frac{1}{2N}\sum_{i=1}^{N}(y_i - t_i)^2 + \lambda\sum_{i=1}^{N} |w_i|$$

4 Results

4.1 Varying Degrees without regularization

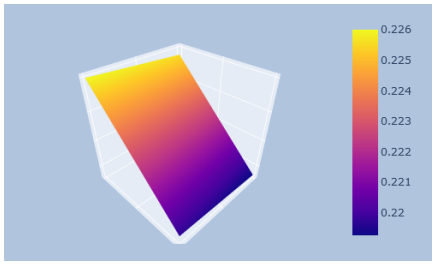
We have tabulated the minimum and training and testing error you get for varying degrees 1, 2, ..., 10

Degree	GD		SGD	
	Train	Test	Train	Test
1	0.021592475	0.025654525	0.021082305	0.02488318
2	0.02133971	0.025311755	0.021180645	0.02490499
3	0.02013052	0.023832495	0.01991128	0.023475785
4	0.019586365	0.023166585	0.019001695	0.02243806
5	0.019371065	0.022907355	0.01905171	0.022441125
6	0.019298645	0.02282811	0.019090205	0.02263292
7	0.01928634	0.0228229	0.01913996	0.022469475
8	0.019291145	0.022838475	0.018882345	0.022268025
9	0.01929522	0.022848155	0.01903992	0.02242288
10	0.01932618	0.022893015	0.019277185	0.02274675

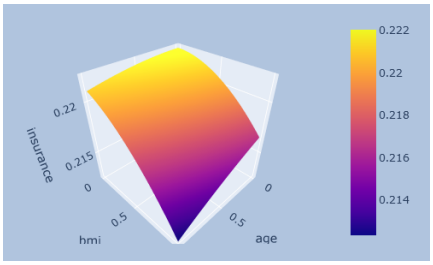
4.2 Varying Degrees with Regularization

We have tabulated the minimum and training and testing error you get by incorporating regularization for varying degrees.

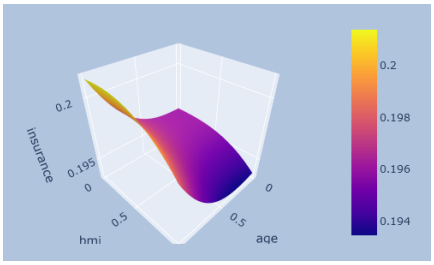
	Train			Validation			Test		
	Degree	Min_train error	lambda_train	degree	~min_test_error	lambda_test	Degree	min test error	lambda
GD_Lasso	6	0.01925	0.3472	6	0.0228	0.3472	6	0.02385	0.17155
GD_Ridge	8	0.0193	0.3472	7	0.02285	0.24405	7	0.02385	0.1532
SGD_Lasso	6	0.01965	0.48115	6	0.02325	0.48115	5	0.0241	0.4531
SGD_Ridge	4	0.01915	0.1968	4	0.02245	0.1968	3	0.02345	0.131



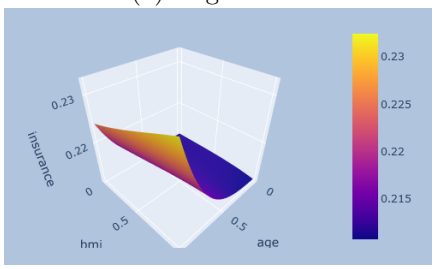
(a) Degree = 1



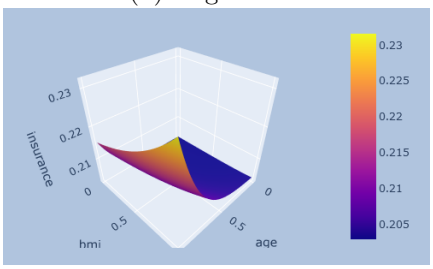
(b) Degree = 2



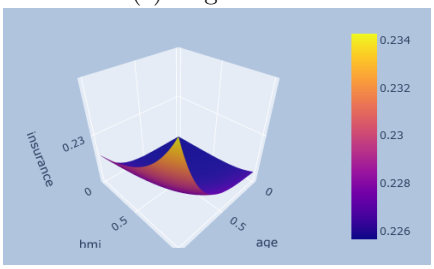
(c) Degree = 3



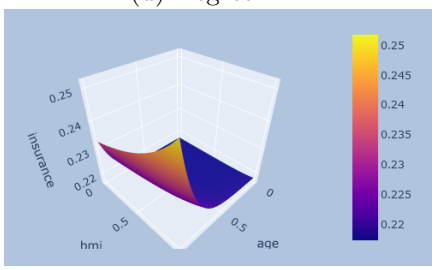
(d) Degree = 4



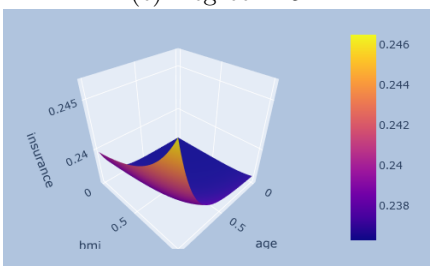
(e) Degree = 5



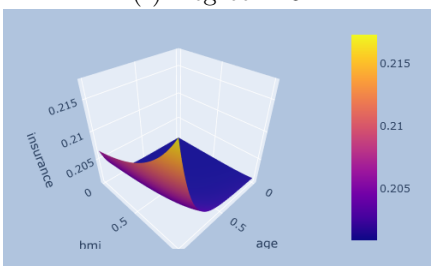
(f) Degree = 6



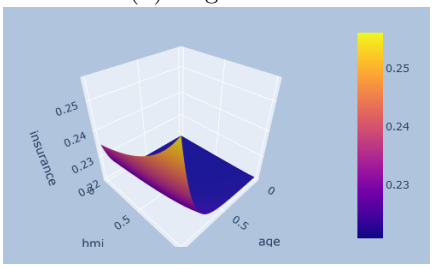
(g) Degree = 7



(h) Degree = 8



(i) Degree = 9



(j) Degree = 10

Figure 1: Surface Plot of Predictions with varying degrees

4.2.1 Overfitting

When we increase the degree beyond a certain optimal degree, the model becomes too complex and fits to the training points very closely. As a consequence, it performs poorly on the validation and testing sets although the training cost function decreases. This is called overfitting. We can combat overfitting by utilizing regularization.

5 Question and Answers

i) Training error decreases and testing error tends to decrease and then increase as the complexity of the model increases. This is because we will first approach an optimal model and then using higher degree polynomials may lead to overfitting.

ii) Since polynomial regression is another case of linear regression, the hessian will be positive semidefinite (the determinant of the principle minor will be positive). As a result the error function will be convex and then there will be a global minima.

iii) Both produce nearly identical results for Gradient descent, but ridge regression performs better for SGD. Lasso stands for: Least Absolute Shrinkage and Selection Operator.
Lasso regression give sparse coefficients, given the assumption that data isn't correlated. So, when we want simpler models, and we know that data isn't correlated (which can also be eliminated by machine learning techniques), we can use lasso regression.

iv) Small λ would not have much effect on the training and larger values of λ would restrict the coefficients too much, so the MSE actually has lesser effect on the error function

v) Yes, we agree with the statement that regularization is necessary when you have a large number of features but limited training instances. In this case, models tend to overfit to the training data. This reduces performance on new data since the model cannot generalize well.

vi) This is a simple combinatorics problem which involves finding D groups in N items. The answer to this problem is $\binom{N+D}{D}$

vii) The total error can be represented as a sum of $Bias^2$ and $Variance$ and an irreducible error η. This would mean that the bias and variance can't both increase or decrease with constant error. Bias can be thought of as a difference of your estimate from the true value of the function and variance can be thought of as the difference of values within your estimate itself, which means that bias shows the simplicity and variance shows the complexity.
As a result, we can say that a model with high bias generally underfits, and model with higher variance generally overfits, and you need to balance both of them to achieve an optimal value. Regularization attempts to penalize the weights in order to decrease the variance, and hence prevents overfitting.