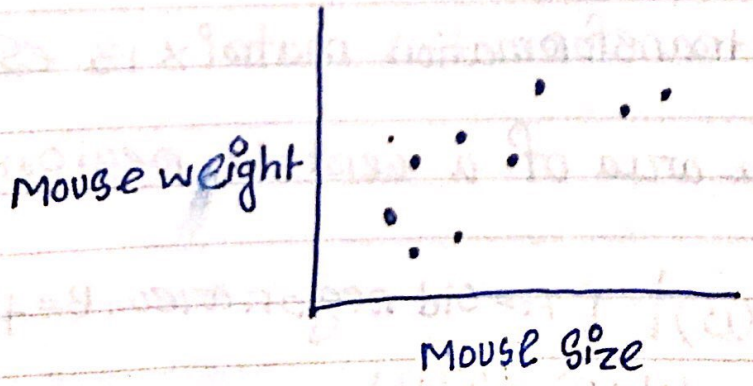
Homework-2

Problem 1

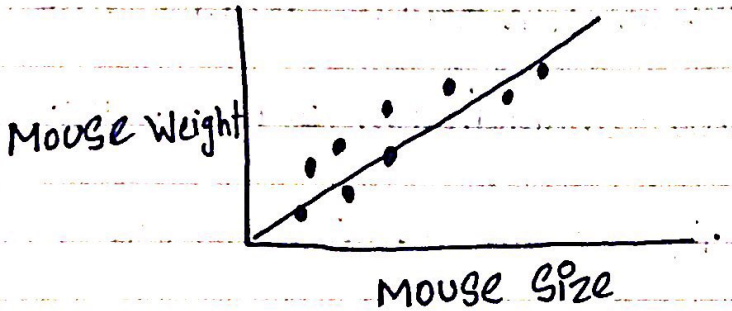
(a) Ridge regression is used when there is not enough data in training set or there is multicollinearity amongst regression predictors variables in a model.

But if there is enough data in the training set and we entirely know about the nature of independent variables in the dataset. Standard linear regression can appropriately predict from the test set.

Let us suppose that we have a decent amount of dataset where we predict mice weight from mice size.

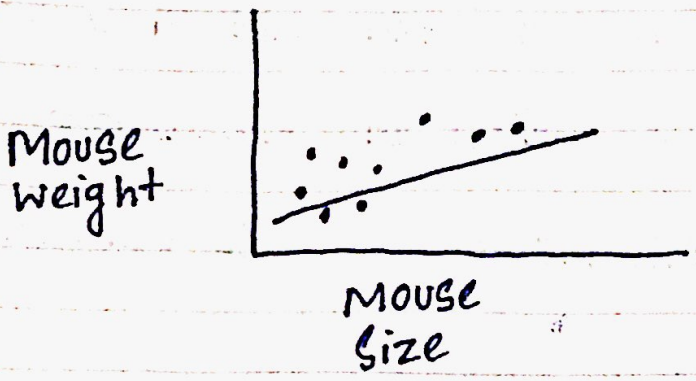


In Standard linear regression, we use AKA least square to model the relationship between weight and size. Least square line accurately reflects the relationship between size and weight.



On the other hand if we use ridge regression in this case it introduces a small amount of bias into how the new line is fit to the data. It returns a small amount of bias and there is a significant drop in variance.

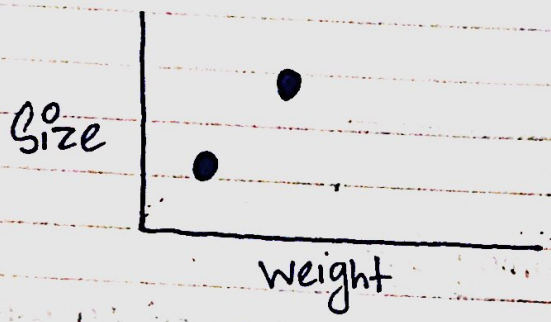
from equation (2), we can see that $\|w\|^2$ add a penalty to the traditional least square method. $(\frac{1}{2})=\lambda$ determines how severe the penalty is. As a result the model finally looks like the following:



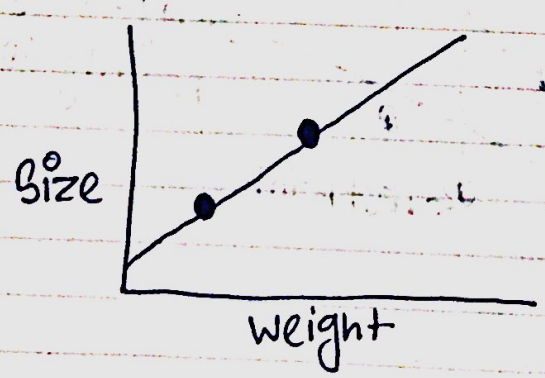
So from the above description we can finally realize that the ridge regression clearly underfits our model, where linear regression fit the model accurately. Ridge regression makes the slope smaller making mouse size less sensitive to mouse weight.

(b)

Let us get back to ovr previous model. This time we just have the two measurements in training set.



When we have a lot of measurements, we can be fairly confident that the least squares line accurately reflects the relationship between size and weight. But as we have only two measurments now we fit a new line with least square.



Here the minimum sum of squared residuals = 0
So, the new line is over fit to the training data.

Now, with ridge regression we introduce a small amount of bias into how the new line is fit to the data

and we get a significant droop in variance.

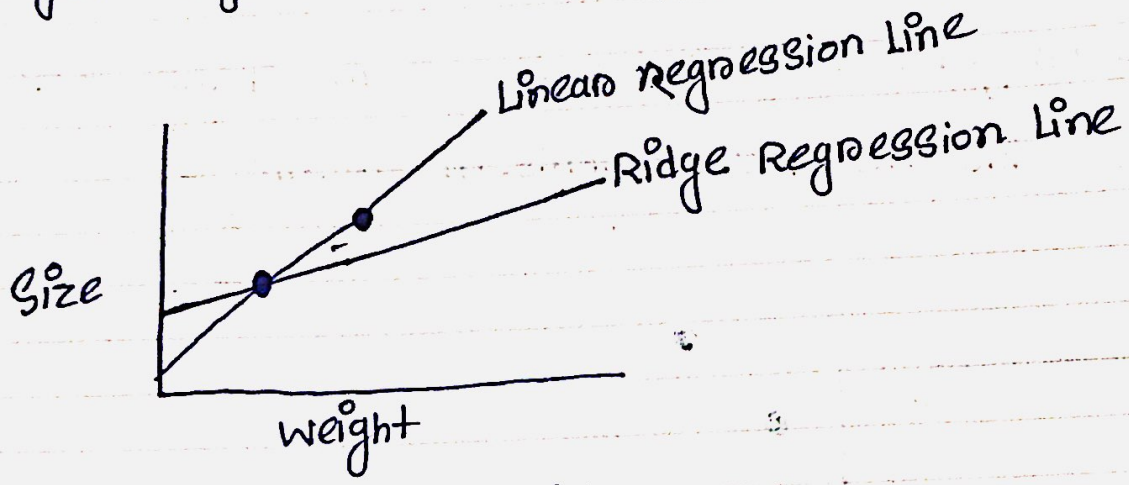


Fig: Training Set

For ridge regression line here, the sum of squared residuals plus the ridge regression penalty is smaller than the sum of the squared residuals of linear regression line.

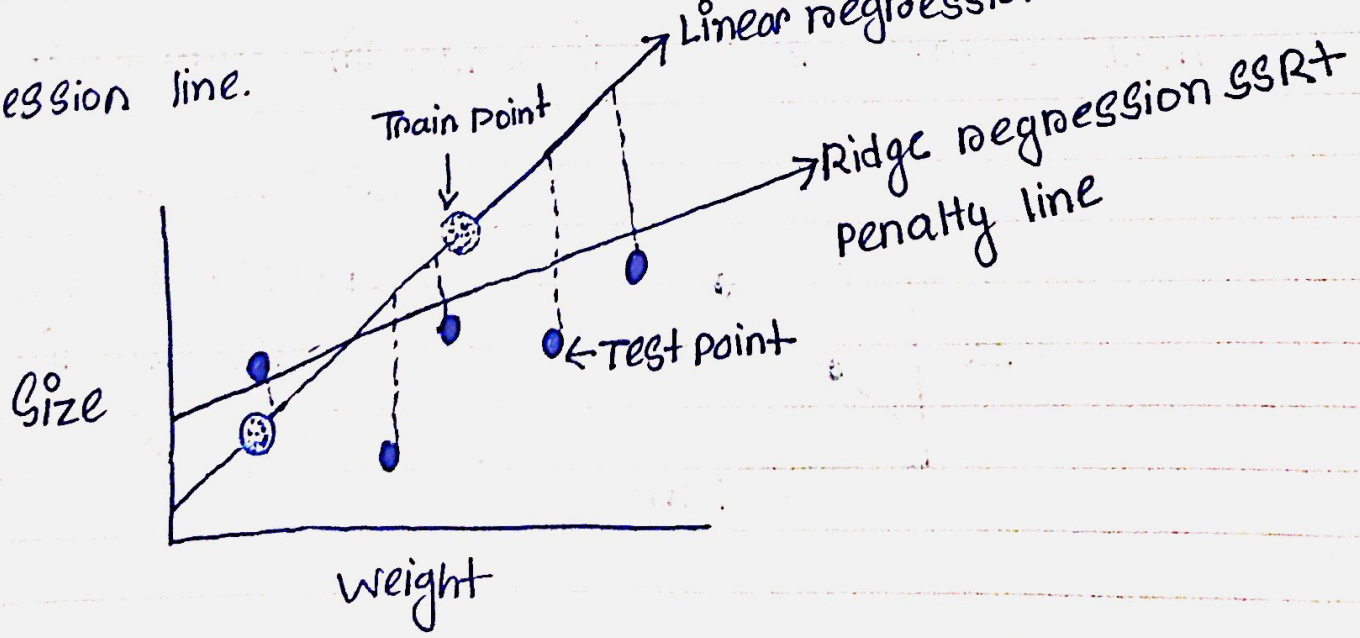
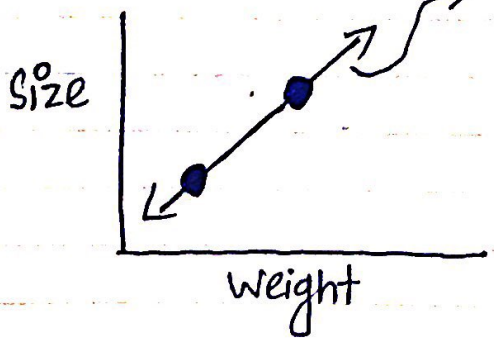


Fig: Test Set

Ridge regression finds new line that doesn't fit the training data by introducing bias with penalizer but in return provides better long term predictions.

(@) The regularization parameter ($\frac{n}{2} = \lambda$) controls the size of coefficient and amount of regularization. Let's get back to our previous model.

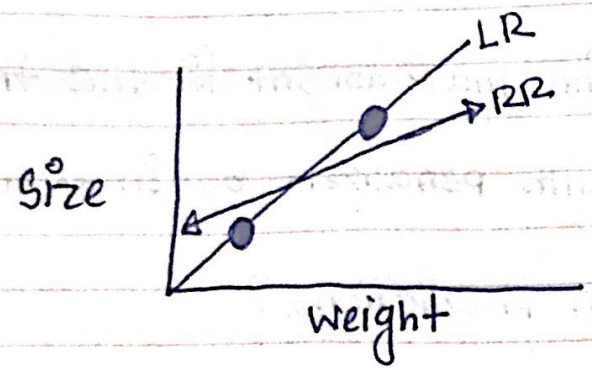
Suppose, $\lambda = 0$, then the ridge regression penalty is also zero. That means the ridge regression line will only minimize the sum of the squared residuals.



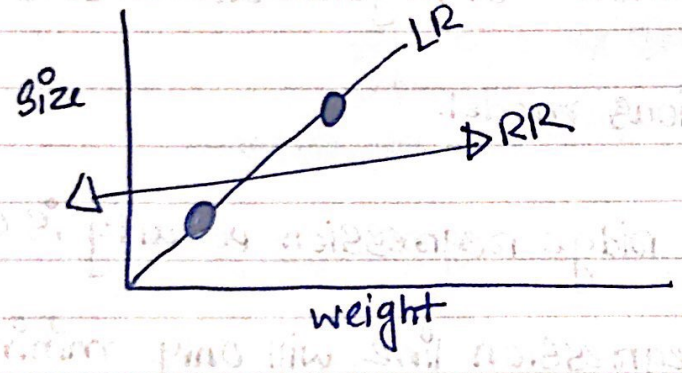
Least square line = ridge regression line

In this case, the bias and variance will be the same as standard linear regression

When $\lambda = 1$, the ridge regression line ended up with a smaller slope than the least square line.



when $\lambda=2$, the slope gets smaller.



The larger we make λ the slope gets asymptotically close to zero. with the increase of λ, our prediction of size become less and less sensitive to weight.

So, from above discussion we find out that, as the model's n, the regularizer gets larger, the model becomes more biased, drops variance and

vice-versa. Summary:

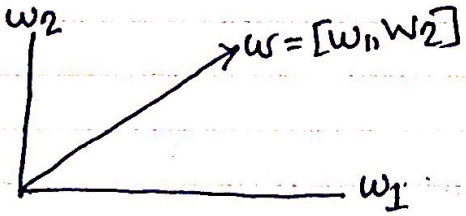
Large λ:
high bias, low variance
Small λ:
low bias, high variance

(c) Given equation:

$$\min_w \|Xw - y\|_2^2 + \frac{\eta}{2}\|w\|_2^2$$

So if we have a vector w in a two dimensional space

the norm of the vector:



Euclidian norm $= \sqrt{w_1^2 + w_2^2}$
$= l_2$ norm

lp norms:

$(w_1^p + w_2^p + \ldots\ldots\ldots + w_d^p)^{\frac{1}{p}}$ [w has d elements]

$= \|w\|_p$

Suppose, $F(w) = \|Xw - Y\|_2^2$

Regularizing linear models:

$F(w) + \lambda \|w\|_2^2 \rightarrow$ l2 regularization.

Ridge regression is l2 norm regularization of linear regression.

$$F(w) = \sum_{i=1}^{N} (y_i - w^T x_i)^2 + \lambda \|w\|_2^2$$

We want to find a w that minimizes $F(w)$, objective function.

$\|w\|_2^2$ can be written as $w^T w$

$$\frac{\partial F(w)}{\partial w} = 0$$

$F(w)$ can also be written as:

$$F(w) = (Y - Xw)^T (Y - Xw) + \lambda w^T w$$

$$\Rightarrow \frac{\partial F(w)}{\partial w} = -2(Y - Xw)X^T + 2\lambda w \qquad \left[\frac{\partial}{\partial x}(x^T x) = 2x\right]$$

Setting $\frac{\partial F(w)}{\partial w} = 0$

$\Rightarrow -2X^T(Y - Xw) + 2\lambda w = 0$

$\Rightarrow w = (X^TX + \lambda I)^{-1} X^TY$

(e) (i) The cost function of linear regression:

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

We need to choose θ to minimize $J(\theta)$

Lets suppose we have 4 features. For each observation we will have: x, x^2, x^3, x^4, y. For each x, x^2, x^3, x^4 we will have parameter θ_i

$$h(x) = \theta_0 + \theta_1 * x + \theta_2 * x^2 + \theta_3 * x^3 + \theta_4 x^4$$

And, we have two data points: (2, 3, 0, 4, 1) and (1, 2, 0, 1, 1)

Now for $J(\theta)$ to be zero, there are numerous solution.

After entering two data points:

$$J(\theta) = \frac{1}{2}(\theta_0 + 2\theta_1 + 3\theta_2 + 4\theta_4 - 1)^2 + \frac{1}{2}(\theta_0 + \theta_1 + 2\theta_2 + \theta_4 - 1)^2$$

From above equations we can say, there is no single solutions of those equations.. There will be infinite

number of solutions. So, deriving any generalized solution form is not possible.

When there is multicollinearity in featureset being cost function of linear model is somewhat convex, nothing stops steepest descent algorithm from converging.

For a given dataset, the mean square error of a linear model = $\sigma^2 Tr((X^TX)^{-1})$

Now with multicollinearity X^TX will have almost linearly dependent columns, leading some eigenvalues to be very small.

$$Tr((X^TX)^{-1}) = \frac{1}{\lambda_1(X^TX)} + \dots \frac{1}{\lambda_p(X^TX)} \geq \frac{1}{\lambda_{min}(X^T}$$

So, any closed form solution isn't possible

(ii) Ridge regression can handle efficiently the cases when there are more features than samples. Also, it can handle multicollinear

The closed form of ridge regression:

$$(X^TX + \lambda I)^{-1} X^TY$$

Ridge regression can find a solution with cross validation and the ridge regression penalty that favors smaller parameter values. As it performs L2 regularization, it adds penalty equivalent to square the magnitude of coefficient which minimizes the sum of square of coefficients to reduce the impact of correlated predictors.

Problem 2:

(a) From the 2D scatter plot for each feature, where each point associates with a sample, and the two coordinates are the feature value and the house price of the sample. The top three features that are most correlated:

1. RM (average number of rooms per dwelling)
- has strong positive correlation with price.

2. LSTAT (lowered status of the population)
- has strong negative correlation with price.

3. B (1000 (Bk - 0.63)^2 where Bk is the proportions of blacks by town - has positive correlation with price.

(b) By visualizing the matrix using heatmap we can find top three features:

1. RM - Stroongly & positively corrorelated with price
2. LSTAS - Strongly & negatively corrorelated with price
3. PTRATIO has next stroong negative corrorelation with price

So it is evident that the first two features (RM, LSTAS) also showed stroong corrorelation in 2(a). But we can notice from the heatmap that PTRATIO is negatively corrorelated with house price (corrorelation is bettero than othero features) that was not obseroved in 2(a)

Problem 2©

Linear Regression Coefficients:

index ▲	Feature	Coeff
0	CRIM	-0.0993237493
1	ZN	0.0522513375
2	INDUS	0.004515606
3	CHAS	2.9572610163
4	NOX	1.1279377483
5	RM	5.8541984993
6	AGE	-0.01495686
7	DIS	-0.9208437549
8	RAD	0.1595191043
9	TAX	-0.0089342717
10	PTRATIO	-0.4356744352
11	B	0.0149052352
12	LSTAT	-0.4747505148

Ridge Regression Coefficients when eta = 15:

Filter Rows

index ▲	Feature	Coeff
0	CRIM	-0.1006476846
1	ZN	0.0546323927
2	INDUS	0.0129580836
3	CHAS	2.2727834496
4	NOX	0.4576743373
5	RM	5.7281521134
6	AGE	-0.0100943724
7	DIS	-0.8969852778
8	RAD	0.1630844674
9	TAX	-0.0089823137
10	PTRATIO	-0.4061490574
11	B	0.0155177873
12	LSTAT	-0.484273584

Ridge Regression Coefficients when eta = 0.02:

index ▲	Feature	Coeff
0	CRIM	-0.0993282363
1	ZN	0.0522540587
2	INDUS	0.0045516127
3	CHAS	2.9561707142
4	NOX	1.1226184046
5	RM	5.8542341864
6	AGE	-0.0149461895
7	DIS	-0.9208461031
8	RAD	0.1595292071
9	TAX	-0.0089338279
10	PTRATIO	-0.4356325653
11	B	0.0149065509
12	LSTAT	-0.4747422626

Ridge regression puts constraint on the coefficients w. The penalty term (eta/2= lambda) regularizes the coefficients such that if the coefficients take large values the optimization function is penalized. Lower the constraint (low eta) on the features, the model will resemble linear regression model (for example when (eta/2 = 0.01). For higher value of eta (15), the magnitudes are considerably less compared to linear regression case.

Problem 2(D):

After implementing the prediction function and root mean square error, I got following results:

The RMSE of linear regression on train set (after concatenating a constant value (1) to each feature vector to learn some linear offset): 4.63142853427834
The RMSE of linear regression on test set (after concatenating a constant value (1) to each feature vector to learn some linear offset): 4.8699261725702

The RMSE of Ridge regression on train set (after concatenating a constant value (1) to each feature vector to learn some linear offset): 4.795434059479303 [When eta = 15 (eta= 15)]

The RMSE of Ridge regression on test set (after concatenating a constant value (1) to each feature vector to learn some linear offset): 5.1603378230671995 [When eta = 15 (eta = 15)]

Discussion:

Ridge regression puts constraint on the coefficients w. The penalty term eta regularizes the coefficients such that if the coefficients take large values the optimization function is penalized. When eta= 15, in the training dataset, eta puts more restriction on the coefficients by shrinking their magnitude that's why RMSE error is greater (4.80) in Ridge regression than linear regression (4.63). It is evident by noticing the RMSE value of test sets in both cases that Ridge regression underfits the Boston housing price model with a 5.16 RMSE where in linear regression the RMSE is 4.87.

Lower the constraint (low eta) on the features, the model will resemble linear regression model (for example when eta = 0.02).

Problem 2(e):

The RMSE of linear regression on train set with top three features: 5.273361751695365
The RMSE of linear regression on test set with top three features: 5.494723646664577

The RMSE of ridge regression on train set with top three features: 5.275045693942413
The RMSE of ridge regression on test set with top three features: 5.481154712581162

We can see that this time in both cases RMSE error is greater (with only top 3 features) than before (trained with all its feature). The features we deleted at least some of them played critical roles in the classification model. Moreover, if we observe the heat matrix, we can see there are some collinearity between RM and LSTAT (-0.6, negative collinearity) which also influenced the accuracy of the model.

Problem 2(f)

Feature Engineering:

I was able to reduce RMSE from 4.87 (2(d)) to 4.77

The techniques I used for feature engineering are following:

1. Zn and INDS are very closely related to each other. So, I added these two features and made them one feature
2. PTRATIO, TAX and B - made these features linear by using a logarithmic function

3. Took the variance of age and add this as a feature and at this point the correlation with MEDV increased then the previous feature AGE

4. Took square root of CRIM, NOX and RAD.

Problem 3(a).

Given that, the maximum likelihood estimation:

$$\min_W F(W) \text{ where } F(W) = \frac{1}{n}\sum_{i=1}^{n} -\log\left[Pr(y=y_i \mid x=X_i; W)\right] + \frac{\eta}{2}\|W\|_F^2$$

The gradient of $F(W) \rightarrow \frac{\partial F(W)}{\partial W}$.

Each data point is independent. So the probability of all the data is:

$$F(W) = \prod_{i=1}^{n} Pr(y=y^{(i)} \mid x=X^{(i)})$$

$$= \prod_{i=1}^{n} \kappa(\theta^T X^{(i)})^{y^{(i)}} \cdot \left[1-\kappa(\theta^T X^{(i)}\right]^{(1-y^{(i)})}$$

Here, $\kappa = \dfrac{\exp(z_k - \max_j z_j)}{\sum_{j=1}^{c} \exp(z_j - \max_j z_j)}$ where $z = \theta^T X$

Where, c = number of possible classes.

and $k \in \{1, 2, \ldots\ldots c\}$

Derivative of gradient for one datapoint (X,Y) without regularizer

$$\frac{\partial F(w)}{\partial w} = \frac{\partial}{\partial w} y \log \sigma(\theta^T x) + \frac{\partial}{\partial w}(1-y)\log[1-\sigma(\theta^T x)]$$

$$= \left[\frac{y}{\sigma(\theta^T x)} - \frac{1-y}{1-\sigma(\theta^T x)}\right] \frac{\partial}{\partial w} \sigma(\theta^T x) \quad \text{(derivative of } \log f(x)\text{)}$$

$$= \left[\frac{y}{\sigma(\theta^T x)} - \frac{1-y}{1-\sigma(\theta^T x)}\right] \sigma(\theta^T x)[1-\sigma(\theta^T x)] x$$

[chain rule + derivative of sigma]

$$= \left[\frac{y-\sigma(\theta^T x)}{\sigma(\theta^T x)[1-\sigma(\theta^T x)]}\right] \sigma(\theta^T x)[1-\sigma(\theta^T x)] x_j$$

[algebraic manipulation]

$$= [y - \sigma(\theta^T x)] x_j \quad \text{[cancelling terms]}$$

The gradient of theta is simply the sum of this term for each training datapoint.

Finally, the gradient of F(w) with regularizer

$$\frac{\partial F(w)}{\partial w} = -\frac{\eta}{2} w_i^{(t)} + \sum_j x_i^j [y^j - \hat{P}(Y^j = 1 | x^j, w^{(t)})]$$

The gradient descent rule for w:

- First we have to initialize $w^{(1)} \in R^D$ randomly
- The we have to iterate through the following convergence:

$$w_i^{(t+1)} \leftarrow w_i^{(t)} + \alpha \left\{ -\lambda w_i^{(t)} + \sum_j x_i^j \left[y^j - \hat{P}(Y^j = 1 | x^j, w^{(t)}) \right] \right\}$$

$w_i^{(t+1)}$: new value; $w_i^{(t)}$: previous value; α: learning rate; the braced term: gradient at previous value

The updates give larger weights to those examples on which the current model makes larger mistakes.

Comparison with least mean square algorithm:

$$w(n+1) = w(n) + \frac{1}{2}\mu[-\nabla J(n)]$$

$J(n)$: cost function

LMS algorithm is based on the idea of gradient descent to search for the minimum error with a cost function equal to the mean squared error, where gradient descent doesn't force the use of any particular cost function, it hunts for the minimum cost solution

Problem 3(b):

1. For learning rate = 5.0e-3, Train Set Accuracy is 0.9755011135857461 and Test set Accuracy is 0.9644444444444444

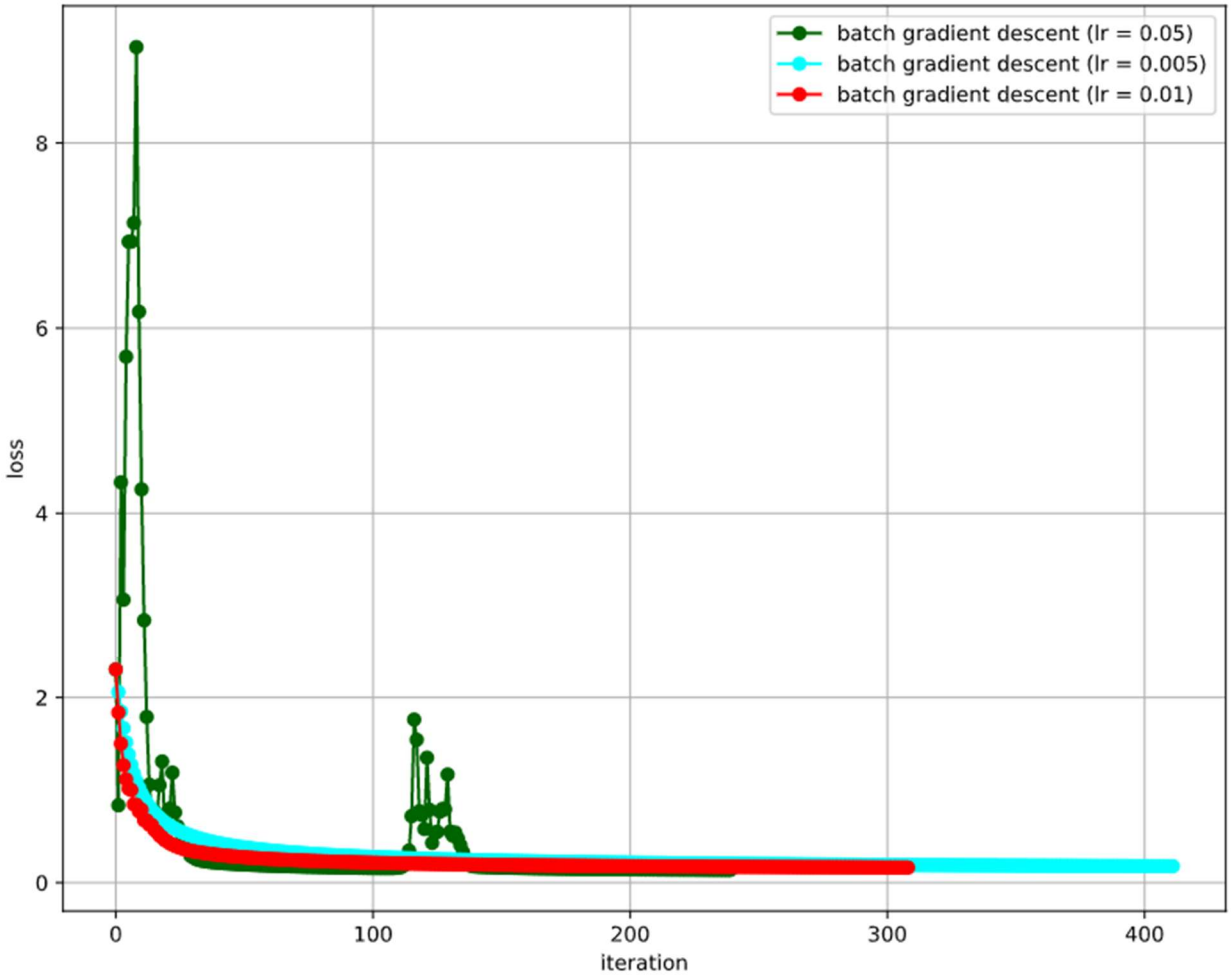
Final Value of F(w): 0.18985915843282414

2. For learning rate = 1.0e-2, Train Set Accuracy is 0.9836674090571641 and Test set Accuracy is 0.9733333333333334

Final Value of F(w): 0.1803129634628979

3. For learning rate = 5.0e-2, Train Set Accuracy is 0.9910913140311804 and Test set Accuracy is 0.9711111111111111

Final value of F(W): 0.1847604454138947



3C.

From the above experiment it was evident that while calculating the batch gradient descent for logistic regression, small learning rates made the gradient very slow. We can see from the image that learning rate 0.005 is taking about 400 iterations to converge. However, it converged ultimately. But when the learning rate is bigger like 0.05, it takes 250 iterations, but it is jumping/oscillating very rapidly overshooting the gradient descent to minima. So, when the learning rate is high although its very fast but sometimes it fails to converge, even diverge.

3d.

1. For batch = 10 and lr = 0.01, Train Set Accuracy is 0.991833704528582 and Test set Accuracy is 0.9733333333333334

Final Value of F(w) = 22.20

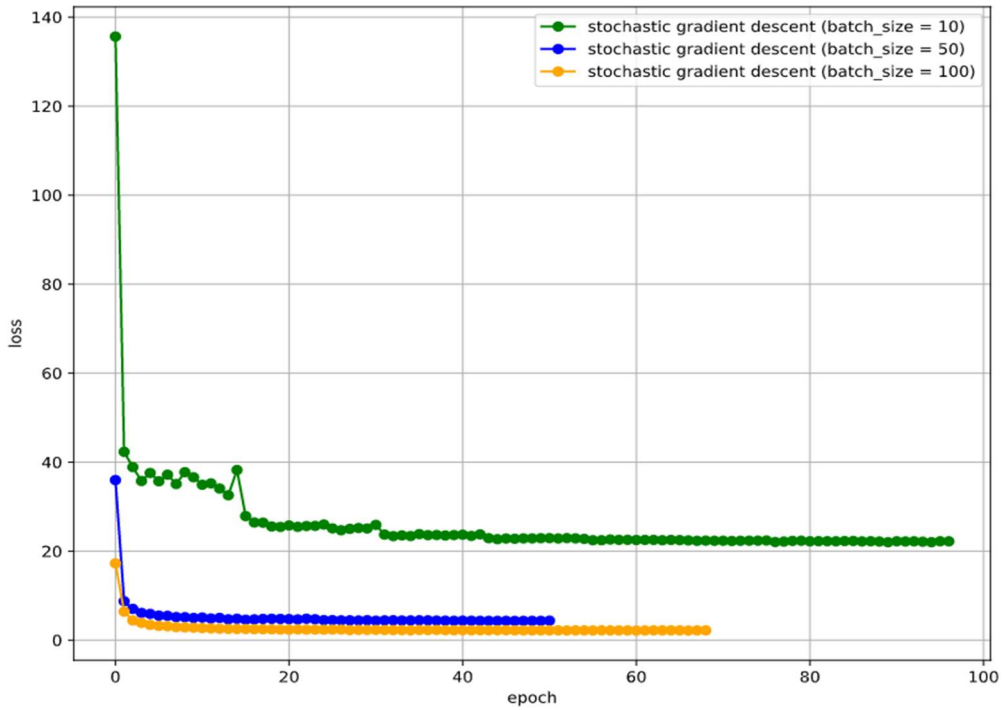
2. For batch = 50 and lr = 0.01, Train Set Accuracy is 0.9866369710467706 and Test set Accuracy is 0.9733333333333334

Final Value of F(w) = 4.3854857636758

3. For batch = 100 and lr = 0.01, Train Set Accuracy is 0.9866369710467706 and Test set Accuracy is 0.9733333333333334

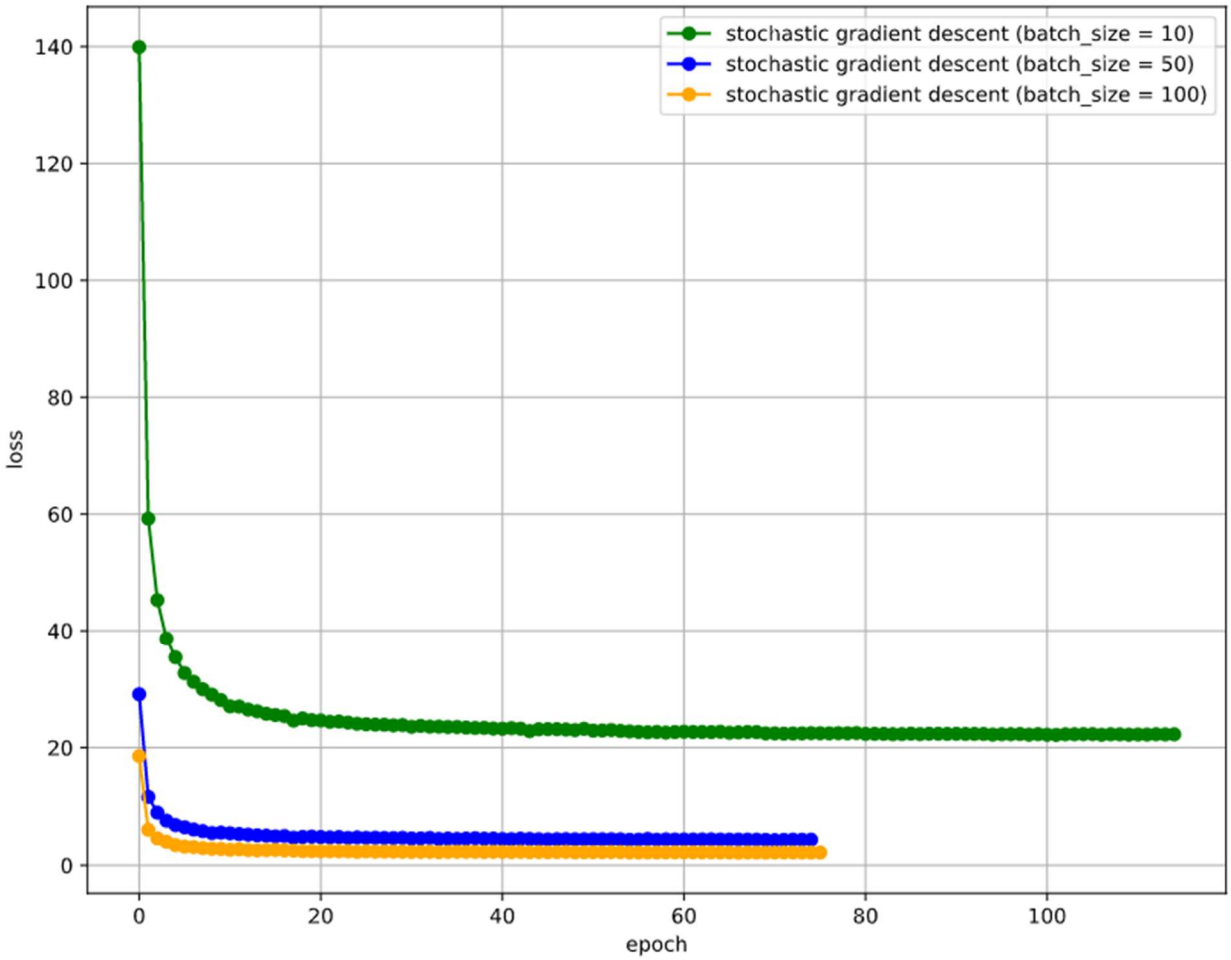
Final Value of F(w) = 0.1847604454138947

Stochastic Gradient Descent for 0.01 learning rate and different mini batches:



3e.

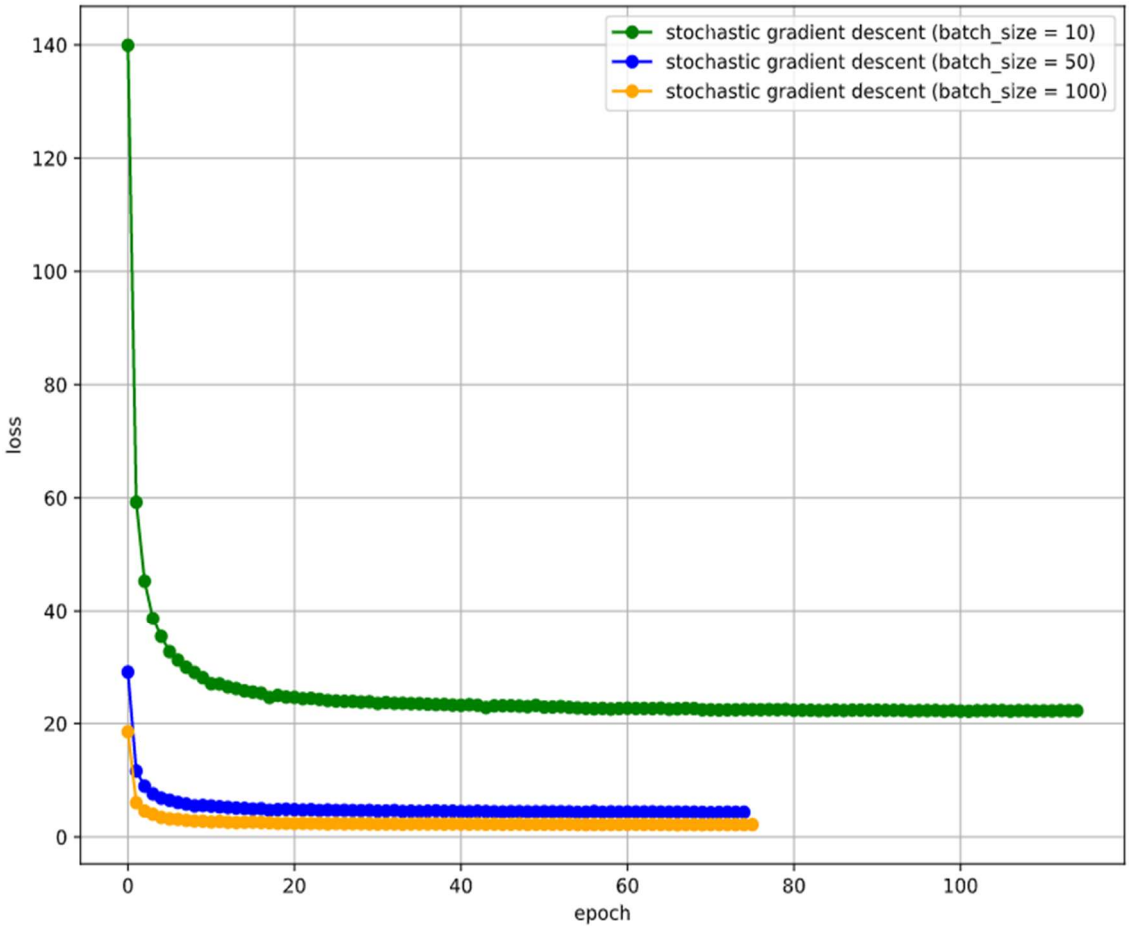
When the same learning rate (0.01) is used, the batches behave differently from each other. It is evident from the graph that, large values (50, 100) are more stable and smoother than small values. Small value (10) asserts instability by almost always oscillating and fluctuating until they converge. Also,



small value (50) take much time to converge than larger values (100)

New Convergence Curve after tuning:

As the rule of thumb is to scale the learning rate linearly with the batch size, for batch size 10, 0.001 learning, for batch size 50, 0.005 learning rate, for batch size 100, 0.1 learning rate has been used.



By comparing two graphs we can say that, when we tune the learning rate finely with respect to batch size the model becomes more stable and less fluctuates. Also, they converge very quickly in terms of wall clock time.

Mathematical Explanations:

Setting learning rate too high in SGD can cause the algorithm to diverge. Again, setting low learning rate can make the model to converge slowly. So, fast convergence requires large learning rates. The problem can be solved by considering implicit update using following equation, this is called iterative method of stochastic gradient descent:

$$w^{new} := w^{old} - \eta \nabla Q_i(w^{new}).$$