Homework Assignment # 2

Submitted by Prerit Anwekar

Total marks: 100

Question 1. [25 MARKS]

Let $X_1, \ldots, X_n$ be i.i.d. Gaussian random variables, each having an unknown mean θ and known variance σ_0^2.

(a) [5 MARKS] Assume θ is itself selected from a normal distribution $\mathcal{N}(\mu, \sigma^2)$ having a known mean μ and a known variance σ^2. What is the maximum a posteriori (MAP) estimate of θ?

Answer(1a).

We know that MAP estimate can be written as,

$$P(\theta|D) \propto argmax_\theta\{P(D|\theta)P(\theta)\}$$

Now we are given each $X_1, X_2..., X_n$ each are gaussian iid and has unknown θ and known variance σ_0^2, So we can estimate θ for any one of the random variables and it will be true for all. Now, Let x_j denote the realization of random variable.

$$P(D|\theta) = \prod_{j=1}^{n} \frac{1}{\sqrt{2\pi\sigma_0^2}} e^{\frac{-1}{2\sigma_o^2}(x_j-\theta)^2}$$

$$P(\theta) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-1}{2\sigma^2}(\theta-\mu)^2}$$

$$P(\theta|D) = (\prod_{j=1}^{n} \frac{1}{\sqrt{2\pi\sigma_0^2}} e^{\frac{-1}{2\sigma_o^2}(x_j-\theta)^2})(\frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-1}{2\sigma^2}(\theta-\mu)^2})$$

$$= (\prod_{j=1}^{n} \frac{1}{\sqrt{2\pi\sigma_0^2}}) e^{\frac{-1}{2\sigma_o^2}\sum_{j=1}^{k}(x_j-\theta)^2} (\frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-1}{2\sigma^2}(\theta-\mu)^2})$$

$$log(P(\theta|D)) = log(\prod_{j=1}^{n} \frac{1}{\sqrt{2\pi\sigma_0^2}}) - \frac{1}{2\sigma_o^2}\sum_{j=1}^{n}(x_j-\theta)^2 + log(\frac{1}{\sqrt{2\pi\sigma^2}}) - \frac{1}{2\sigma^2}(\theta-\mu)^2$$

Differentiating this equation w.r.t. to θ.

$$\frac{\partial ll}{\partial \theta} = \frac{1}{\sigma_o^2}\sum_{j=1}^{n}(x_j-\theta) - \frac{1}{\sigma^2}(\theta-\mu)$$

Now,

$$\frac{\partial ll}{\partial \theta} = 0$$

$$\begin{aligned}
\frac{1}{\sigma^2}(\theta_{MAP} - \mu) &= \frac{1}{\sigma_o^2}\sum_{j=1}^{n}(x_j - \theta_{MAP}) \\
\frac{\theta_{MAP}}{\sigma^2} - \frac{\mu}{\sigma^2} &= \frac{\sum_{j=1}^{n} x_j}{\sigma_0^2} - \frac{n\theta_{MAP}}{\sigma_0^2} \\
\frac{\theta_{MAP}}{\sigma^2} + \frac{n\theta_{MAP}}{\sigma_0^2} &= \frac{\sum_{j=1}^{n} x_j}{\sigma_0^2} + \frac{\mu}{\sigma^2} \\
\theta_{MAP}(\sigma_0^2 + n\sigma^2) &= \sigma^2\sum_{j=1}^{n} x_j + \sigma_0^2\mu \\
\theta_{MAP} &= \frac{\sigma^2\sum_{j=1}^{n} x_j + \sigma_0^2\mu}{\sigma_0^2 + n\sigma^2}
\end{aligned}$$

(b) [10 MARKS] Assume θ is itself selected from a Laplace distribution $\mathcal{L}(\mu, b)$ having a known mean (location) μ and a known scale (diversity) b. Recall that the pdf for a Laplace distribution is

$$p(x) = \frac{1}{2b}\exp\left(\frac{-|x-\mu|}{b}\right)$$

For simplicity, assume $\mu = 0$. What is the maximum a posteriori estimate of θ? If you cannot find a closed form solution, explain how you would use an iterative approach to obtain the solution.

Answer(1b).
Given,

$$p(x) = \frac{1}{2b}e^{\frac{-|x-\mu|}{b}}$$

also, $\mu = 0$.
Then,

$$p(\theta) = \frac{1}{2b}e^{\frac{-|\theta|}{b}}$$

$$P(\theta|D) \propto argmax_{\theta}\{P(D|\theta)P(\theta)\}$$

$$P(D|\theta) = \prod_{j=1}^{n}\frac{1}{\sqrt{2\pi\sigma_0^2}}e^{\frac{-1}{2\sigma_o^2}(x_j-\theta)^2}$$

Now we can define,

$$P(\theta|D) = [\prod_{j=1}^{n}\frac{1}{\sqrt{2\pi\sigma_0^2}}e^{\frac{-1}{2\sigma_o^2}(x_j-\theta)^2}](\frac{1}{2b}e^{\frac{-|\theta|}{b}})$$

Taking log of both sides,

$$ll = log(\prod_{j=1}^{n} \frac{1}{\sqrt{2\pi\sigma_0^2}}) - \frac{1}{2\sigma_o^2}\sum_{j=1}^{n}(x_j - \theta)^2 + log(\frac{1}{2b}) - \frac{|\theta|}{b}$$

Differentiating it w.r.t. to θ

$$\frac{\partial ll}{\partial \theta} = \frac{1}{\sigma_o^2}\sum_{j=1}^{n}(x_j - \theta) - \frac{\partial|\theta|}{b\partial\theta}$$

Now,

$$\frac{\partial ll}{\partial \theta} = 0$$

$$\frac{\partial|\theta|}{\partial\theta} = \frac{b}{\sigma_o^2}\sum_{j=1}^{n}(x_j - \theta)$$

$$\frac{\partial|\theta|}{\partial\theta} = \frac{b}{\sigma_o^2}\sum_{j=1}^{n}x_j - n\theta$$

A simple iterative method would be to subtract this gradient that we have found out from θ at each step of the iteration. This can be helpful in doing a gradient descent algorithm. One can also use Newton Raphson method to find the subsequent values.

Now as for the close form solution,

Let y = $|\theta|$, which can be written as,

$$y = \sqrt{\theta^2}$$

On differentiating w.r.t.θ,

$$\frac{\partial y}{\partial\theta} = \frac{1}{2}(\theta^2)^{-\frac{1}{2}}(2\theta)$$
$$= \frac{\theta}{\sqrt{\theta^2}}$$
$$= \frac{\theta}{|\theta|}$$

$$\frac{\theta}{|\theta|} = \frac{b}{\sigma_o^2}\sum_{j=1}^{n}x_j - n\theta$$

$$\frac{\theta}{|\theta|} + n\theta = \frac{b}{\sigma_o^2}\sum_{j=1}^{n}x_j$$

We can have two conditions, when $|\theta| = \theta$

$$1 + n\theta_{MAP} = \frac{b}{\sigma_o^2}\sum_{j=1}^{n}x_j$$

$$\theta_{MAP} = \frac{\frac{b}{\sigma_o^2}\sum_{j=1}^{n} x_j - 1}{n}$$

or when $|\theta| = -\theta$,

$$-1 + n\theta_{MAP} = \frac{b}{\sigma_o^2}\sum_{j=1}^{n} x_j$$
$$\theta_{MAP} = \frac{\frac{b}{\sigma_o^2}\sum_{j=1}^{n} x_j + 1}{n}$$

(c) [10 MARKS] Now assume that we have **multivariate** i.i.d. Gaussian random variables, $\mathbf{X}_1, \ldots, \mathbf{X}_n$ with each $\mathbf{X}_i \sim \mathcal{N}(\boldsymbol{\theta}, \boldsymbol{\Sigma}_0)$ for some unknown mean $\boldsymbol{\theta} \in \mathbb{R}^d$ and known $\boldsymbol{\Sigma}_0 = \mathbf{I} \in \mathbb{R}^{d\times d}$, where $\mathbf{I}$ is the identity matrix. Assume $\boldsymbol{\theta} \in \mathbb{R}^d$ is selected from a zero-mean multivariate Gaussian $\mathcal{N}(\boldsymbol{\mu} = \mathbf{0}, \boldsymbol{\Sigma} = \sigma^2\mathbf{I})$ and a known variance parameter σ^2 on the diagonal. What is the MAP estimate of $\boldsymbol{\theta}$?

Answer(1c).

$$P(\theta|D) \propto argmax_\theta\{P(D|\theta)P(\theta)\}$$

The pdf for multivariate gaussian variable is,

$$p(D|\theta) = \prod_{i=1}^{n} \frac{1}{\sqrt{(2\pi)^d|\Sigma_0|}} e^{\frac{-1}{2}(x_i-\theta)^T\Sigma_0^{-1}(x_i-\theta)}$$
$$p(\theta) = \prod_{i=1}^{n} \frac{1}{\sqrt{(2\pi)^d|\Sigma|}} e^{\frac{-1}{2}(\theta)^T\Sigma^{-1}(\theta)}$$

Substituting values from the question, $\Sigma_0 = I$ and $\Sigma = \sigma^2 I$

$$P(\theta|D) = p(D|\theta)P(\theta)$$
$$= (\prod_{i=1}^{n} \frac{1}{\sqrt{(2\pi)^d}} e^{\frac{-1}{2}(x_i-\theta)^T I(x_i-\theta)})(\prod_{i=1}^{n} \frac{1}{\sqrt{(2\pi)^d\sigma^2}} e^{\frac{-1}{2\sigma^2}(\theta)^T I(\theta)})$$
$$ll = log(\prod_{i=1}^{n} \frac{1}{\sqrt{(2\pi)^d}}) - \sum_{i=1}^{n} \frac{1}{2}(x_i-\theta)^T I(x_i-\theta)$$
$$+log(\prod_{i=1}^{n} \frac{1}{\sqrt{(2\pi)^d\sigma^2}}) - \frac{1}{2\sigma^2}(\theta)^T I(\theta)$$

Differentiating w.r.t. to θ and equating it to zero,

$$\frac{\partial ll}{\partial \theta} = \frac{1}{2}[(I + I^T)]\sum_{i=1}^{n}(x_i-\theta) - \frac{1}{2\sigma^2}[I + I^T](\theta)$$

$$\sum_{i=1}^{n}(x_i - \theta) = \frac{1}{\sigma^2}(\theta)$$
$$\sum_{i=1}^{n} x_i - n\theta = \frac{1}{\sigma^2}(\theta)$$
$$n\theta = \sum_{i=1}^{n} x_i - \frac{1}{\sigma^2}(\theta)$$
$$\theta\frac{(n\sigma^2 + 1)}{\sigma^2} = \sum_{i=1}^{n} x_i$$
$$\theta_{MAP} = \frac{\sigma^2 \sum_{i=1}^{n} x_i}{1 + n\sigma^2}$$

Question 2. [75 MARKS]

In this question, you will implement variants of linear regression. We will be examining some of the practical aspects of implementing regression, including for a large number of features and samples. An initial script in python has been given to you, called `script_regression.py`, and associated python files. You will be running on a UCI dataset for CT slices[1], with 385 features and 53,500 samples. Baseline algorithms, including mean and random predictions, are used to serve as sanity checks. We should be able to outperform random predictions, and the mean value of the target in the training set.

(a) [5 MARKS] The main linear regression class is `FSLinearRegression`. The FS stands for FeatureSelect. The provided implementation has subselected features and then simply explicitly solved for $\mathbf{w} = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{y}$. Increase the number of selected features (including all the way to including all the features). What do you find? How can this be remedied?

Answer(2a).

Number of Featuers	Error	Comments
5	2413.75371962	Things look good, weights are ok.
50	1023.15931218	Things look good, weights are ok.
100	*error*	We ended with a singular matrix. Somethings wrong!
200	*error*	We ended with a singular martrix again.
385	*error*	Let's go all out. We still ended with a singular matrix.

Table 1: Comparision of Linear Regression Algorithm as number of features changes.

After running the algorithm a number of times with different number of features, it was observed that we are ending up with singular matrix. This is not by chance because we have a feature matrix whose determinant is either zero or near zero due to which we are getting error. This can be remedied by either dropping those features or dropping the rows which makes our determinant zero or we can use regularization to add a bias to each features.

(b) [5 MARKS] Modify the code to average the error over multiple splits of the data, reporting both the mean and standard error. When running your experiments, how might you choose the number of splits over which to average?

Answer(2b).

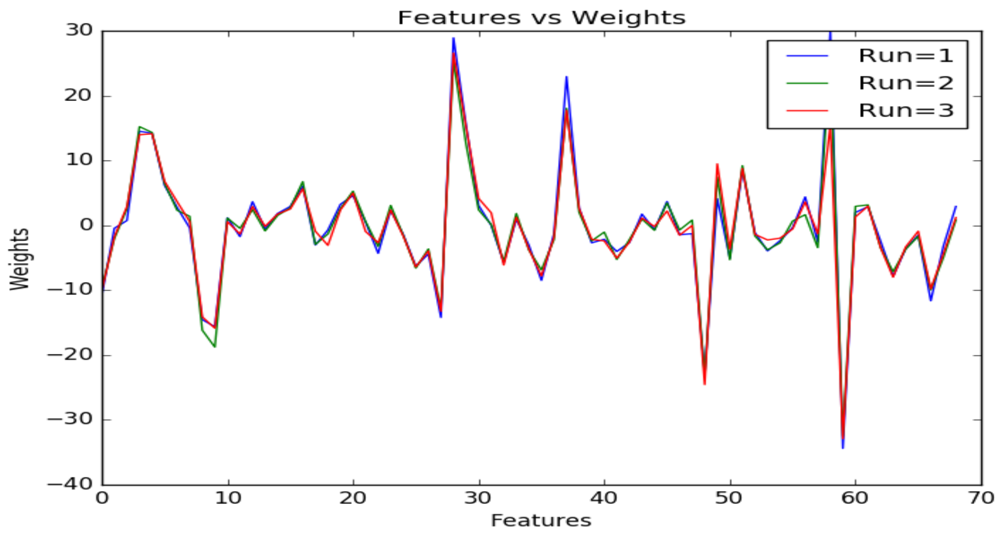
[1] `https://archive.ics.uci.edu/ml/datasets/Relative+location+of+CT+slices+on+axial+axis`

For this question I have considered the fact the splitdataset function is choosing random samples to create the training data and if we sample it enough times we will get an almost accurate model. As per my knowledge this question talks about a similar approach to boosting aggregating where we sample data many times and then average out the values to get an approximate stable model.

On three runs of the experiment.

Mean Error= 0.13410017288 Standard Error= 0.000122801353278

Here's a graph for how weights varied with features in each run.



The splits are chosen on the basis of the fact that as we sample more and more times from the dataset we will get different models which on averaging gives a representation of our true model.

(c) [5 MARKS] Now imagine that you want to compare an algorithm with different meta-parameter values (e.g., regularization parameter). Modify the code to enable this comparison. Explain your choices.

Answer (2c). To compare different meta-parameter values a new parameter *lambda_param* is included into the learn function, please check the (Ridge regression class in code/regressionalgorithms.py). There are two choices to enable this feature:

1. If we don't have a closed form solution, we add the meta-parameter to the cost function and then use gradient descent to optimize the value until it converges.

2. If we have a closed form solution we will just add λI to the X^TX and then take the inverse, where λ is our meta-parameter. (This is basically ridge regression).

(d) [5 MARKS] Now implement Ridge Regression, where a ridge regularizer $\lambda\|\mathbf{w}\|_2^2$ is added to the optimization. Run this algorithm on all the features. How does the result differ from (a)? Discuss results for three different regularization parameters: $\lambda = 0.01, 0.1, 1.0$.

Answer (2d). I have implemented the ridge in closed form by adding λI as given in the notes. So what we are doing here is adding a small value to the diagonal of our feature matrix which will make all the features different and that is the reason we will not get singular matrix anymore.

Compared to (a) : Firstly, the problem of Singular matrix is removed.Secondly, the error is reduced, that means the model complexity is decreasing.

The result of using Ridge regression are following:

1. The algorithm runs for all the features. No singular matrix observed anymore.

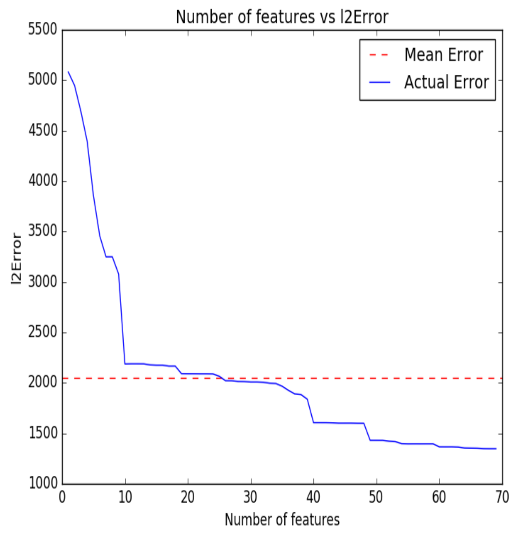
2. As we increased the value of meta-parameter λ the L2 Error starts to decrease. If we increase it too much we may under-fit the model.

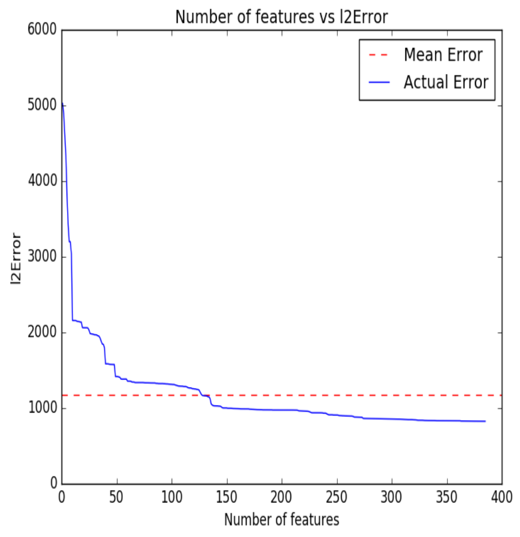
3. Although we are penalizing the weights they will never become zero.
4. As we increase the value of lambda the model complexity reduces.

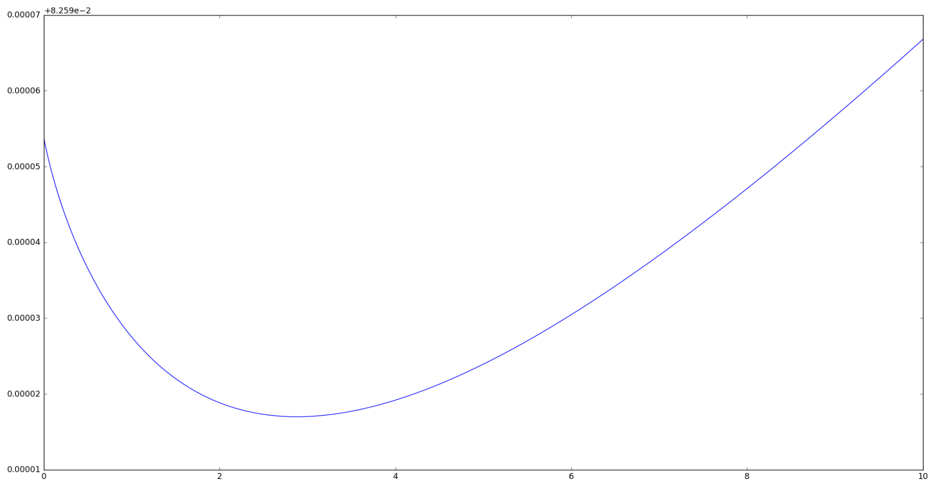
Following are results of experiments on values of λ = [0.01,0.1,1.0] with ridge regression.
Lambda = 0.01 Train Error = 0.0573079051576, Test Error = 0.0836743568769
Lambda = 0.1 Train Error = 0.0573080757009, Test Error = 0.0836733503694
Lambda = 1.0 Train Error = 0.0573140647888, Test Error = 0.0836666424011
The results from the above values of lambda shows no significant changes in test error.
train size = 20,000 ; test size = 10,000

I experimented a little bit on my own and actually ran on features when equal to [0,1,2,3...69] before regularization and see how error changes as ridge regularization is applied and all features starts to contribute in the model.

Below are couple of graphs which doesn't pertain to actual question asked but I was bit curious and wanted to share my findings.







This is a graph between different lambda values in range of [10e-05,10] and test error. Although the test error will increase as we try to underfit our model with higher lambda value, I was expecting a subtle graph.

(e) [15 MARKS] Imagine that the dataset size continues to grow, which causes the matrix $\boldsymbol{X} \in \mathbb{R}^{n \times d}$ for n samples and d features to become quite large. One option is to go back to subselecting features. In this question, you will implement a simple greedy algorithm for selecting a subset of features, often called forward greedy selection or matching pursuit (see "On the Consistency of Feature Selection using Greedy Least Squares Regression", Zhang, 2009). The idea is to greedily add one new feature on each iteration, based on its (Pearson) correlation with the residual: the feature with the maximum absolute dot product with the residual. On each step, for current subset s of features and residual $\mathbf{R} = \mathbf{X}(:, s)\mathbf{w} - \mathbf{y}$, one adds the feature i with maximal $|\mathbf{X}(:, i)^\top \mathbf{R}|$. The coefficients are then recomputed for the current subset of features, and another feature considered. This iteration ends once the residual error is below some threshold, or if $\max_i |\mathbf{X}(:, i)^\top \mathbf{R}|$ is below some threshold. Further details can be found in the cited paper by Zhang. Implement `MPLinearRegression` and report error.

Answer (2e).

Experiment 1:

corr_epsilon = 0.003 , res_epsilon=0.000000007

Note: These smaller values were chosen to see if max features kicks in. One can choose different values for these parameters.

Total number of features selected = 100

Features selected = [48, 132, 118, 378, 273, 4, 227, 255, 135, 115, 120, 294, 3, 35, 127, 106, 226, 114, 292, 224, 192, 63, 28, 180, 254, 0, 215, 5, 77, 27, 97, 42, 191, 264, 137, 225, 98, 331, 8, 76, 252, 40, 237, 228, 150, 212, 119, 174, 45, 23, 22, 6, 216, 220, 277, 193, 182, 369, 340, 103, 64, 116, 105, 291, 91, 71, 74, 301, 7, 38, 267, 274, 36, 170, 133, 136, 139, 148, 323, 143, 324, 374, 229, 121, 110, 236, 288, 156, 138, 55, 307, 218, 202, 247, 333, 96, 222, 295, 297, 358]

Error = 0.0906614762694

Experiment 2:

corr_epsilon = 0.3 , res_epsilon=0.07

Total number of features selected = 2

Features selected = [48, 132]

Error = 0.176140293075

(f) [15 MARKS] Now imagine that you want to try a different feature selection method and you've heard all about this amazing and mysterious Lasso. Lasso can often be described as an algorithm, or otherwise as an objective with a least-squares loss and ℓ_1 regularizer. It is more suitably thought of as the objective, rather than an algorithm, as there are many algorithms to solve the Lasso. Implement an iterative solution approach that uses the soft thresholding operator (also called the shrinkage operator). Discuss the impact of the choice of regularization parameter on the error of `LassoRegression`.

Answer (2f).

Following experiments were performed with l1 regularization.

Experiment 1:

Lambda= 0.105

Number of zero weights= 13

Test error= 0.318598989127

Experiment 2:

Lambda= 0.5

Number of zero weights= 81

Test error= 0.376618190624

Experiment 3:
Lambda= 1.0
Number of zero weights= 285
Test error= 0.465846052804

Discussion:
1) The weight coefficients are smaller for lasso regression compared to ridge regression.
2) Even for a small value of lambda we get significant decrease in value of weight coefficients.
3) Lasso regression starts to make weight coefficients of some features zero. This is the feature selection property of lasso.
4) Train error starts to increase and test error starts to improve as we increase the value of alpha.
5) After a particular value of lambda we start to underfit our model and errors starts increasing.

(g) [10 MARKS] Now imagine that your dataset has gotten even larger, to the point where dropping features is not enough. Instead of removing features, implement a stochastic optimization approach to obtaining the linear regression solution (see Section 4.5.3). Explain your implementation choices. Report the error.

Answer (2g).
If we do a warm start with weights = random values. We get following results:
Experiment 1:
alpha = 0.01
Error= 0.372312544474

Experiment 2:
alpha = 0.03
Error= 0.372312544474
If we initialize weights to zeros then we get following results.

Experiment 3:
alpha = 0.01
Error= 0.293507909732

Experiment 4:
alpha = 0.03
Error= 0.22066560369
The implementation choices that were made are following:
1) We work on one training example at a time. This is useful when we have large datasets or are dealing with stream of data. The weights can be updated on the fly.
2) It is much faster as we don't work on the "batch" of data with the trade-off that we always get an approximation on each iteration as we don't consider all the data at once. Each example is kind of new dataset (hypothetically) to the model.
3) The learning rate alpha is changed based on concept of diminishing alpha where we decrease the value of alpha on each iteration.
4) We will chose the number of epochs depending upon when we think the cost function is minimum.

5) Since our objective function is convex we will always get a global minimum with stochastic gradient.

(h) [15 MARKS] Implement batch gradient descent for linear regression. Compare stochastic gradient descent to batch gradient descent, in terms of the number of times the entire training set is processed. Set the step-size to 0.01 for stochastic gradient descent, and implement a reasonable approach to select the step-size for batch gradient descent. Report the error versus epochs, where one epoch involves processing the training set once. Report the error versus runtime.

Answer (2h).

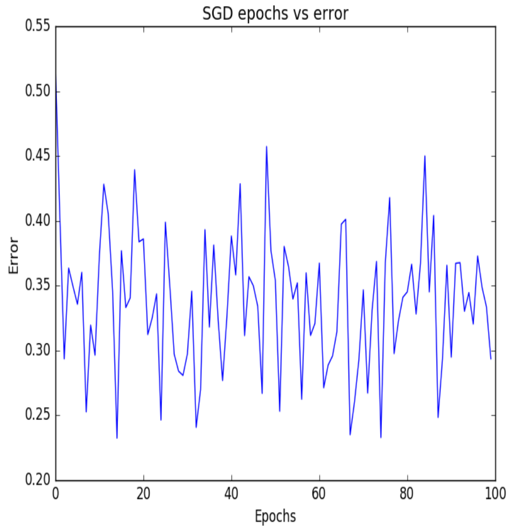
1) In stochastic gradient descent, the gradient is computed based on one sample at a time while in batch gradient descent whole dataset is used at once.

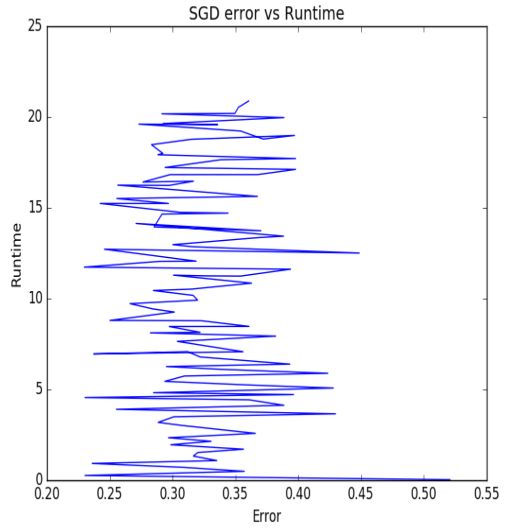
2) A stochastic gradient descent algorithm will be much faster than batch gradient descent but the strength of SGD will increase as we increase the number of epochs.

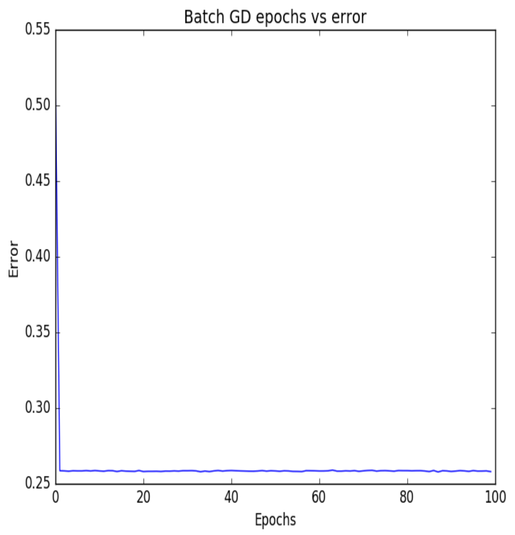
3) A stochastic gradient can be performed on a stream data and is much more beneficial to revise a model while Batch gradient descent can prove computationally expensive.

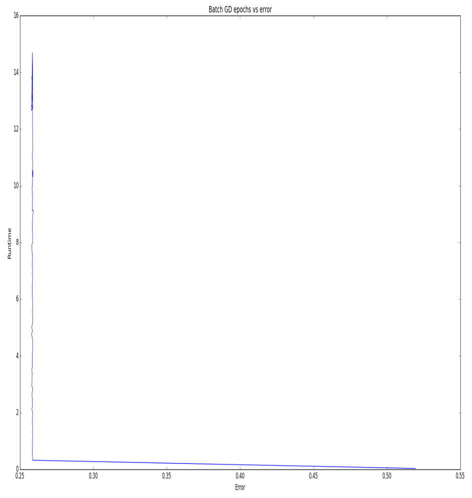
4) In stochastic gradient our error will fluctuate a lot as seen in the graph below on 100 epochs but will converge to minimum as we increase number of epochs. This fluctuation can also help us find new minima.

5) There is only update in Batch gradient descent while update is performed for each sample observed.






Batch GD epochs vs error
Error
Epochs
0.55
0.50
0.45
0.40
0.35
0.30
0.25
0
20
40
60
80
100


Batch GD epochs vs error
Runtime
Error
16
14
12
10
8
6
4
2
0
0.25
0.30
0.35
0.40
0.45
0.50
0.55