Neural Networks and Deep Learning

CSCI 5922 – Assignment 1 (Fall 2017) by Akshit Arora

Part 1:

Question 1a

w_1 = -2.04424, w_2 = 3.99686, b = -0.92429

(Additionally, Final mean squared error: 0.04)

Question 1b

Used linear regression from scikit-learn to find the solution. This method computes the least squares solution using a singular value decomposition of X. If X is a matrix of size (n, p) this method has a cost of O(n * p^2), assuming that n >= p. In this case, n = 100 (number of rows) and p = 3 (number of features, including bias). Used because it was available in the library.

Code used for Part 1: assign1_p1.py

Part 2

Upon experimenting with different settings of learning rate, number of epochs and batch/minibatch/online (say, setting 'A'). I found two useful conditions that lead to the solution to our objective of minimizing the number of epochs.

1. Given the number of epochs (10,000), what is the value of learning rate and setting A that give least error (or best performance):

Learning rate ↓	Online	Batch	Mini-Batch (Size = 5)
0.00001	0.408347	1.40768	0.85823
0.0001	0.039411	1.10794	0.04644
0.001	**0.03940897**	0.13747	**0.03940884**
0.01	0.039409	0.03941	0.03940885
0.1	0.04823	**0.039408**	0.03999
1	0.2106	-	-

Observe that minimum error for all the settings comes out to be similar. All are just below 0.039409. I can use that as a maximum error threshold while calculating number of sweeps in the next condition given below. Codes used in Part 2 first condition: assign1_p2_batch.py, assign1_p2_minibatch.py and assign1_p2_online.py

2. Given the maximum error threshold (0.039409), which setting gives least number of sweeps (with maximum number of sweeps limited to 100,000):

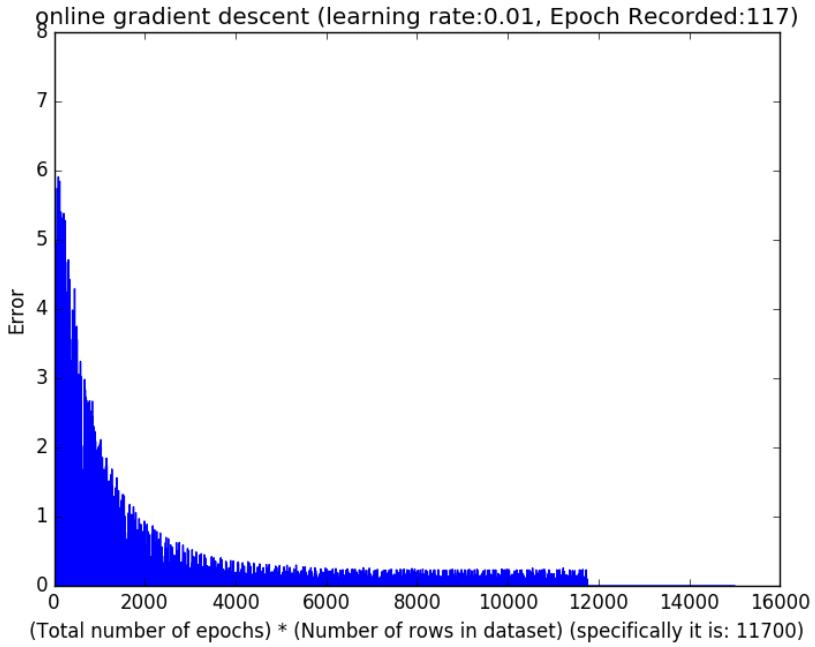
Learning rate ↓	Online	Batch	Mini-Batch (Size = 5)
0.00001	Sweeps Exceeded	Sweeps Exceeded	Sweeps Exceeded
0.0001	Sweeps Exceeded	Sweeps Exceeded	Sweeps Exceeded
0.001	1192	61484	3035
0.01	**112**	6144	**289**
0.1	1039	**610**	466
1	Sweeps Exceeded	Sweeps Exceeded	Sweeps Exceeded

Codes used in Part 2 second condition (inside sub-folder "stopping_criteria_error"): assign1_p2_batch.py, assign1_p2_minibatch.py and assign1_p2_online.py

From the above two tables, it looks like Online gradient descent with a learning rate of 0.01 would give best performance (less than 0.039409 error) in minimum number of sweeps (112).

REPORTING WEIGHTS AT THE SETTING: **ONLINE GRADIENT DESCENT, LEARNING RATE: 0.01, ERROR: 0.039409, EPOCHS: 112, MAX ITERATIONS ALLOWED: 150** => w_1 = -2.04388, w_2 = 3.99781, b = -0.92498 {Code available in file: assign1_p2_online.py}

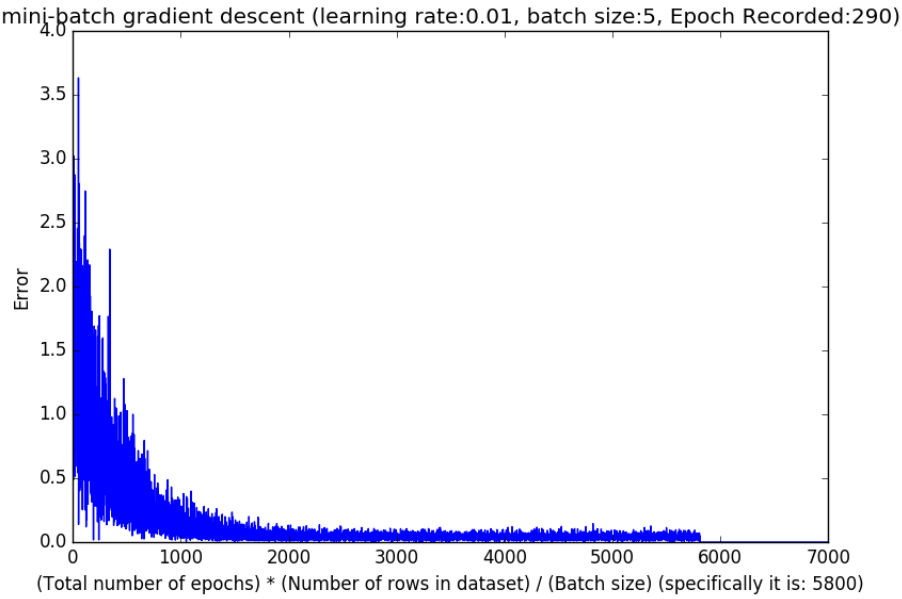
PLOT:



But, I have considered performance in terms of error only and not time taken to compute. If we consider the time taken to compute the weights then mini-batch gradient descent wins, because in mini-batch we can parallelize our computation, this facility is not available in online gradient descent.

REPORTING WEIGHTS AT THE SETTING: **MINI-BATCH GRADIENT DESCENT, LEARNING RATE: 0.01, ERROR: 0.039409, BATCH SIZE: 5, EPOCHS: 289, MAX ITERATIONS ALLOWED: 350** => w_1 = -2.04400, w_2 = 3.99593, b = -0.92397 {code available in file: assign1_p2_minibatch.py}

PLOT:

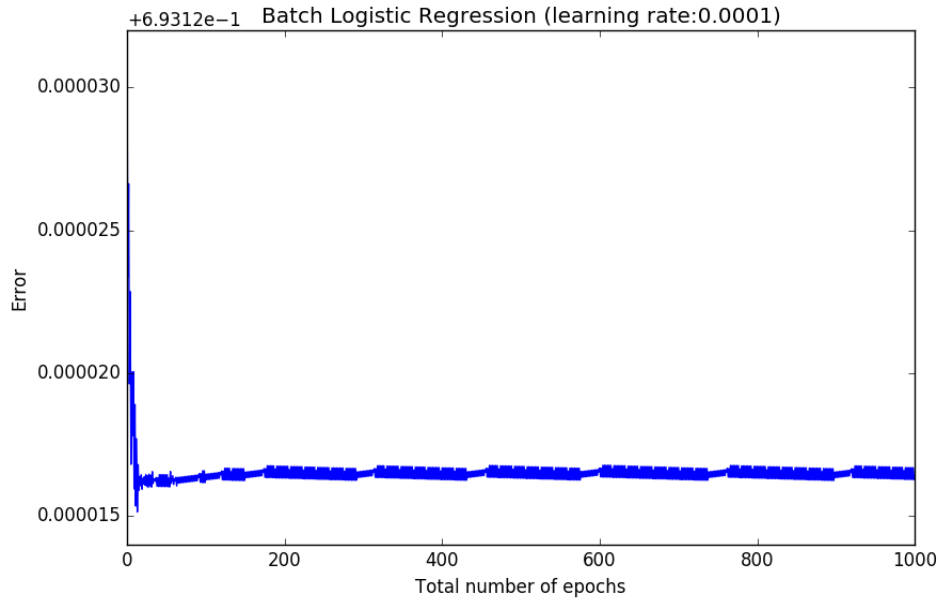


Additionally, I used stochastic gradient descent to implement online gradient descent. So, the actual number of epochs may vary slightly (+- 5 epochs) across different runs of the algorithm. It is because I randomize data points after every iteration, this is done in both online and mini-batch gradient descent implementations.

Part 3

Implemented basic logistic regression (batch) using sigmoid function. Learning Rate: 0.0001, Number of iterations: 1000.

Observation: Regular patterns appear from around x = 180. Look at the following graph:



In order to determine the weights in this case. We know that regular patterns appear. We spot that first such pattern appears between 170 and 300! (before 170 it is all irregular pattern)

We just need to look at the Error[170:300] and take the minimum error position (say, t). Then simply get B[t], W1[t] and W2[t] and that would be the answer.

But, it looks like there is another dip (in the irregular pattern before 170) in the very beginning around x = 10. So, including index 5 to 300 to find the minimum.

Upon following the procedure described above, I found that minimum error occurs at epoch = 13 with learning rate 0.0001.

w_1 = -3.64729000e-05; w_2 = 9.24969000e-05; b = -7.00000000e-06

Code used in Part 3: assign1_p3.py

Part 4

The performance of test set varies with amount of training data as follows:

# Training Examples	Average Error
5	0.684282227937
25	0.681142708119
50	0.682653247051
75	0.682698551011

Code used: assign1_p4.py