**20CS6037 Summer 2021**

**Assignment #2**

**Assigned 9/14/2021**

**Due on Canvas 9/24/2021**

We continue with the insurance data set, this time we will consider all attributes except the region attribute, for which the following **gradient descent algorithms** are to be applied**:**

* **Batch:**

without L2 regularization

with L2 regularization

with L1 regularization

* **Stochastic:**

without L2 regularization

with L2 regularization

with L1 regularization

* **Mini batch:**

without L2 regularization

with L2 regularization

with L1 regularization

Inspect / analyze the data set before you start the ML portion: Plot the data **before** proceeding, normalize the data as follows: if X is the data set (i.e., without the output y) standardize it by subtracting from each column its mean and dividing it by the standard deviation. **Do this, before you extend the data set by adding the extra column of 1s.**

Example of standardizing the data set X (in Matlab):

Let a data set be

X =

75 23 83 8

45 91 54 44

8 15 100 11

plot(X’). % Note: In Matlab, plot(X) will plot the columns not the rows of X



**muX=mean(X)**

muX = 42.6667 43.0000 79.0000 21.0000

**stdX=std(X)**

stdX = 33.5609 41.7612 23.2594 19.9750

Now, create two matrices of the same size as X with repeated **muX** and **stdX**, respectively. Use repmat Matlab command:

**repstd=repmat(stdX,3,1)**

repstd =

33.5609 41.7612 23.2594 19.9750

33.5609 41.7612 23.2594 19.9750

33.5609 41.7612 23.2594 19.9750

**repmu=repmat(muX, 3, 1)**

repmu =

42.6667 43.0000 79.0000 21.0000

42.6667 43.0000 79.0000 21.0000

42.6667 43.0000 79.0000 21.0000

**standardizedX = (x-repmu)./repstd**

standardizedX =

0.9634 -0.4789 0.1720 -0.6508

0.0695 1.1494 -1.0748 1.1514

-1.0329 -0.6705 0.9029 -0.5006

Then the column averages and column standard deviations will be 0 and 1, respectively.

**[mean(standardizedX) ; std(standardizedX)]**

ans =

0.0000 -0.0000 0 0

1.0000 1.0000 1.0000 1.0000

plot(standardizedX’):



Divide your data randomly, into two subsets as follows:

Train (approx. 50%): Xtrain, an m by (n+1) matrix

Test (remaining subset): Xtest, an m by (n+1) matrix

**Save the train and test data sets for comparison purposes.**

Ytrain: the ground truth for the training set

Ytest: the ground truth for the test set

Yhat: Model output for the training set

Train and test each model without regularization.

Recall the steps:

1. Let m be the size of the training set
2. Set up the weight vector (let’s call it **w**), of dimension n+1, where n is the number of attributes
3. Randomly initialize **w**
4. Calculate the output vector **Yhat** of dimension m, for the training data points, using the formula: for each data point (row vector in **Xtrain**) compute the dot product with the vector w
5. Training MSE is the mean of (**Ytrain – Yhat**).^2
6. Calculate **gradMSE** the vector of dimension n+1 according to the formula

**gradMSE(w)** = (2/m) x (**Xtrain**)’ x (**Xtrain** x **w** – **Ytrain**)

1. Update **w**:

**w** = **w** – l **gradMSE(w),**

where l is the learning rate.

1. Repeat Steps 4 – 8 until the termination condition is met (error less than some small value, bound on the number of epochs, or a combination of these).

For stochastic gradient descent, one updates **w** after each training point is used.

For mini-batch gradient descent, the steps above are carried out for subsets (mini-batches) of the training set.

**L2 Regularization**

Instead of **gradMSE** from step 6, use the vector of dimension n+1, with components

**(gradMSE)i+ 2** a **wi, i=1, …, n+1**

You will need to come up with a way to select the parameter a(maybe validation?)

**Eliminate the attribute (column in X) corresponding to the smallest component of the weight vector.**

Train again, on the reduced set of attributes, using batch gradient using Xtrain, Xtest, etc. previously saved.

Divide the data sets into training and test subsets of approximately equal, and train again, using the attributes selected by the L2-regularization step.

**L1 Regularization**

Perform L1 regularization for the batch Gradient Descent. That is, instead of **gradMSE** from step 6, use the vector of dimension n+1, with components

**(gradMSE)i+ 2** a **sign(wi), i=1, …, n+1**

**Eliminate the attribute (column in X) corresponding to the smallest component(s) (value 0) of the weight vector.**

Train again, on the reduced set of attributes, using batch gradient using **Xtrain**, **Xtest**, etc. previously saved.

Divide the data sets into training and test subsets of approximately equal, and train again, using the attributes selected by the L1-regularization step.

Provide the following figures:

Plot the regression lines for each attribute in each the regression for each of the attributes in each mode of training before and after L1 and L2 regularizations for batch gradient descent.

Please do not forget to:

1. Turn in your homework in a zipped folder named identified by the names in alphabetical order of your team members
2. Include in this folder the data set
3. Write at the top of your program files the names of your team members
4. Comment your program
5. Turn in an analysis of (up to one page) of your results

**Hint:** For those coding in Matlab, Matlab has a “publish” option which can be used to produce .doc, .pdf,.html, or .tex files, which can then be further edited.