Problem Set 3: Batch Gradient Descent

Justin Dang
CS 3113 - Artificial Intelligence
University of Arkansas — Fort Smith

March 10, 2023

1 Background

In linear regression, the prediction from a linear regression model takes the form of

$$\hat{y} = x\beta = xw$$

where $\hat{y}$ represents the prediction, x represent a feature, and w or β represents its corresponding weight. The error (or residual) is calculated by

$$\varepsilon = y - \hat{y}$$

where y is the true value. To measure the sum of squared errors:

$$SSE = (y - X\beta)^T(y - x\beta) = \sum_i (y - \hat{y}_i)^2$$

Recall that the goal for an accurate prediction based off the model is to *minimize error*. In order to do this, gradient descent can be used. **Gradient Descent** is an optimization algorithm and finds the local minimum of a given differentiable function. In this case, this function is the loss function:

$$\mathcal{L}(X, y, \hat{y}) = \mathcal{L}(X, y, h_w(\overrightarrow{x}))$$

To find $\hat{y}$, it is made into a hypothesis function that accepts some input of data.

$$h(\overrightarrow{x}) = \hat{y}$$

. Using the mean squared error, it is rigorously defined as

$$\mathcal{L}(X, y, h_w(\overrightarrow{x})) = \frac{1}{2n}[\sum_i^n (y_i - h_w(x^i))^2]$$

X is a matrix of observations in the row and features in the column. Taking this loss function, a partial derivative with respect to a given weight to find the minimum.

$$\frac{\partial}{\partial w_i}[\frac{1}{2n}[\sum_i^n (y_i - \hat{y}_i)^2]]$$

Using the chain rule:

$$\frac{1}{n}[\sum_{i}^{\hat{n}}(y_i - \hat{y_i}) \cdot \frac{\partial}{\partial w_3}[y_i - \hat{y_i}]]$$

which simplifies to

$$\frac{\partial}{\partial w_i}[\mathcal{L}()] = \frac{1}{n}[\sum_{i}^{n}(y_i - \hat{y_i}) \cdot (-x_i)]$$

Essentially for every single value in the vector of weights, update w_i where

$$w_i = w_i - \alpha \cdot \frac{\partial}{\partial w_i}[\mathcal{L}()]$$

The derived weights will be a $(p+1) \times 1$ vector. α and ϵ are used to control the learning rate and error tolerance respectively.

2 Implementation

In Java, the input to the loss and descent functions is a 2-dimensional array.

```
ArrayList<ArrayList<Double>> LRModel = new ArrayList<>();
```

The given input data had 60 observations with 15 features and 1 target column. Once recived into the `LRModel`, a normalization step was done using z-scores in order to reduce convergence computation. For all data points:

$$Z = \frac{x - \mu}{\sigma}$$

The data was split into parts for testing and training. 80% of the data was reserved for building the weights in training. For each feature, the corresponding μ and σ were saved to denormalize predicted values during testing. The remaining 20% (12 records) will represent that testing data and will use the derived weights from training.
Consider this hypothesis method that is based off of the loss formula. When run in a for loop, it will multiply weights with features to measure error.

```
1
2   public double hypothesis(int observation) {
3
4       double sum = 0;
5       for (int i = 0; i < featureCount; i++) {
6           sum = sum + (LRModel.get(observation).get(i) *
                modelWeights.get(i));
7       }
8       return sum;
9   }
```

It well follows the hypothesis function where $h_w(x) = w^T x$. It returns a double for that observation's predicted values. Paired with the loss function, it forms the basis of the gradient descent.

```
1     public double lossFunction(double threshold) {
2
3        double summation = 0;
4
5        for (int j = 0; j < recordCount * threshold; j++) {
6            summation = summation + Math.pow((targetFeature.get(j) -
                 hypothesis(j)), 2);
7        }
8
9        return (summation / (2 * (recordCount * threshold)));
10    }
```

The convergence criteria is $\Delta_{\%cost} < \epsilon$, where the change of the percent cost is defined by measuring the difference of change between the previous and current loss numbers. With final weights, the testing phase predicts accurate numbers with a fair amount of epochs (how many derivatives it needs to take before the function converges).

```
    Testing Phase:
-----------------------------------------------------------------
Loss of Testing Data    363.668

Test Record: 1    True: 790.73    Prediction: 800.227   Error: 9.494
Test Record: 2    True: 899.26    Prediction: 923.619   Error: 24.355
Test Record: 3    True: 904.16    Prediction: 881.371   Error: 22.784
Test Record: 4    True: 950.67    Prediction: 926.439   Error: 24.233
Test Record: 5    True: 972.46    Prediction: 927.731   Error: 44.733
Test Record: 6    True: 912.20    Prediction: 900.326   Error: 11.876
Test Record: 7    True: 967.80    Prediction: 1009.285  Error: 41.482
Test Record: 8    True: 823.76    Prediction: 887.051   Error: 63.287
Test Record: 9    True: 1003.50   Prediction: 935.804   Error: 67.698
Test Record: 10   True: 895.70    Prediction: 899.057   Error: 3.361
Test Record: 11   True: 911.82    Prediction: 934.492   Error: 22.675
Test Record: 12   True: 954.44    Prediction: 926.751   Error: 27.691

=> Number of Test Entries (n):         12
```

3 Outputs

```
    /PS3/ps3data.csv
****************************************
Problem Set:  Problem Set 4:  Gradient Descent
Name:         Justin Dang
```

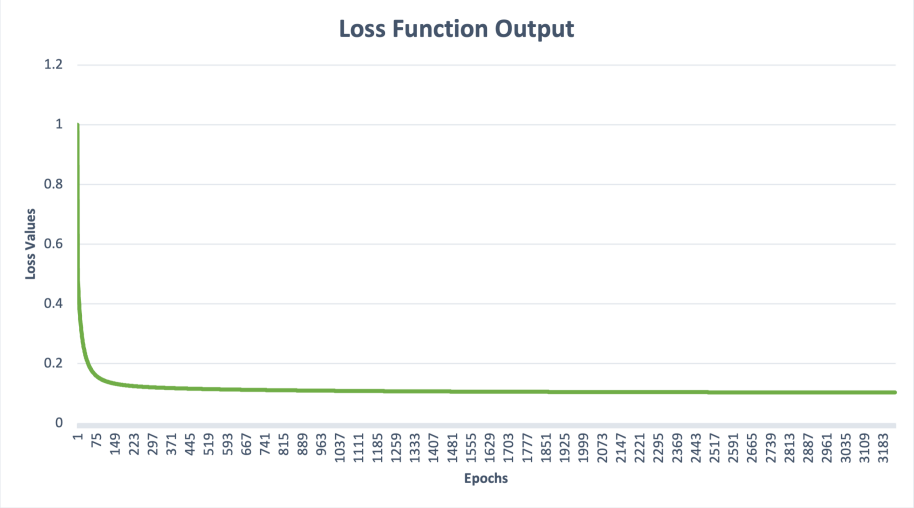
```
Syntax:       java PS3 PS3/ps3data.csv 15 PS3/output.txt  0.01 0.001
************************************************************

Training Phase:PS3/ps3data.csv
------------------------------------------------------------
=> Number of Entries(n): 60
=> Number of Features(p): 17


Starting Gradient Descent:
------------------------------------------------------------

Epoch 1   :     Loss of 1.000      Delta = N/A          Epsilon =  N/A
Epoch 2   :     Loss of 0.500      Delta = 4.08454%   Epsilon =  0.001%
Epoch 3   :     Loss of 0.480      Delta = 3.94492%   Epsilon =  0.001%
Epoch 4   :     Loss of 0.461      Delta = 3.80852%   Epsilon =  0.001%
Epoch 5   :     Loss of 0.443      Delta = 3.67552%   Epsilon =  0.001%
Epoch 6   :     Loss of 0.427      Delta = 3.54609%   Epsilon =  0.001%
Epoch 7   :     Loss of 0.412      Delta = 3.42036%   Epsilon =  0.001%
Epoch 8   :     Loss of 0.398      Delta = 3.29843%   Epsilon =  0.001%
Epoch 9   :     Loss of 0.384      Delta = 3.18037%   Epsilon =  0.001%
Epoch 10  :     Loss of 0.372      Delta = 3.06623%   Epsilon =  0.001%
Epoch 11  :     Loss of 0.361      Delta = 2.95603%   Epsilon =  0.001%
Epoch 12  :     Loss of 0.350      Delta = 2.84977%   Epsilon =  0.001%
Epoch 13  :     Loss of 0.340      Delta = 2.74743%   Epsilon =  0.001%
Epoch 14  :     Loss of 0.331      Delta = 2.64896%   Epsilon =  0.001%
Epoch 15  :     Loss of 0.322      Delta = 2.55431%   Epsilon =  0.001%
Epoch 16  :     Loss of 0.314      Delta = 2.46341%   Epsilon =  0.001%
Epoch 17  :     Loss of 0.306      Delta = 2.37616%   Epsilon =  0.001%
Epoch 18  :     Loss of 0.299      Delta = 2.29248%   Epsilon =  0.001%
Epoch 19  :     Loss of 0.292      Delta = 2.21226%   Epsilon =  0.001%
Epoch 20  :     Loss of 0.286      Delta = 2.13539%   Epsilon =  0.001%
Epoch 21  :     Loss of 0.279      Delta = 2.06176%   Epsilon =  0.001%
Epoch 22  :     Loss of 0.274      Delta = 1.99125%   Epsilon =  0.001%
Epoch 23  :     Loss of 0.268      Delta = 1.92374%   Epsilon =  0.001%
Epoch 24  :     Loss of 0.263      Delta = 1.85911%   Epsilon =  0.001%
Epoch 25  :     Loss of 0.258      Delta = 1.79724%   Epsilon =  0.001%
Epoch 26  :     Loss of 0.254      Delta = 1.73801%   Epsilon =  0.001%
Epoch 27  :     Loss of 0.249      Delta = 1.68130%   Epsilon =  0.001%
Epoch 28  :     Loss of 0.245      Delta = 1.62700%   Epsilon =  0.001%
Epoch 29  :     Loss of 0.241      Delta = 1.57500%   Epsilon =  0.001%
Epoch 30  :     Loss of 0.237      Delta = 1.52518%   Epsilon =  0.001%
Epoch 31  :     Loss of 0.234      Delta = 1.47744%   Epsilon =  0.001%
Epoch 32  :     Loss of 0.230      Delta = 1.43167%   Epsilon =  0.001%
Epoch 33  :     Loss of 0.227      Delta = 1.38779%   Epsilon =  0.001%
Epoch 34  :     Loss of 0.224      Delta = 1.34570%   Epsilon =  0.001%
```

```
Epoch 35  :     Loss of 0.221      Delta = 1.30530%   Epsilon =  0.001%
Epoch 36  :     Loss of 0.218      Delta = 1.26652%   Epsilon =  0.001%
Epoch 37  :     Loss of 0.215      Delta = 1.22927%   Epsilon =  0.001%
Epoch 38  :     Loss of 0.212      Delta = 1.19347%   Epsilon =  0.001%
Epoch 39  :     Loss of 0.210      Delta = 1.15906%   Epsilon =  0.001%
Epoch 40  :     Loss of 0.207      Delta = 1.12596%   Epsilon =  0.001%
Epoch 41  :     Loss of 0.205      Delta = 1.09411%   Epsilon =  0.001%
Epoch 42  :     Loss of 0.203      Delta = 1.06345%   Epsilon =  0.001%
Epoch 43  :     Loss of 0.201      Delta = 1.03392%   Epsilon =  0.001%
Epoch 44  :     Loss of 0.199      Delta = 1.00546%   Epsilon =  0.001%
Epoch 45  :     Loss of 0.197      Delta = 0.97802%   Epsilon =  0.001%
Epoch 46  :     Loss of 0.195      Delta = 0.95156%   Epsilon =  0.001%
Epoch 47  :     Loss of 0.193      Delta = 0.92603%   Epsilon =  0.001%
Epoch 48  :     Loss of 0.191      Delta = 0.90139%   Epsilon =  0.001%
Epoch 49  :     Loss of 0.189      Delta = 0.87759%   Epsilon =  0.001%
Epoch 50  :     Loss of 0.188      Delta = 0.85460%   Epsilon =  0.001%
Epoch 51  :     Loss of 0.186      Delta = 0.83237%   Epsilon =  0.001%
Epoch 52  :     Loss of 0.185      Delta = 0.81089%   Epsilon =  0.001%
Epoch 53  :     Loss of 0.183      Delta = 0.79011%   Epsilon =  0.001%
Epoch 54  :     Loss of 0.182      Delta = 0.77001%   Epsilon =  0.001%
Epoch 55  :     Loss of 0.180      Delta = 0.75055%   Epsilon =  0.001%
Epoch 56  :     Loss of 0.179      Delta = 0.73171%   Epsilon =  0.001%
Epoch 57  :     Loss of 0.178      Delta = 0.71347%   Epsilon =  0.001%
Epoch 58  :     Loss of 0.176      Delta = 0.69580%   Epsilon =  0.001%
Epoch 59  :     Loss of 0.175      Delta = 0.67868%   Epsilon =  0.001%
Epoch 60  :     Loss of 0.174      Delta = 0.66209%   Epsilon =  0.001%
Epoch 61  :     Loss of 0.173      Delta = 0.64600%   Epsilon =  0.001%
Epoch 62  :     Loss of 0.172      Delta = 0.63040%   Epsilon =  0.001%
Epoch 63  :     Loss of 0.170      Delta = 0.61527%   Epsilon =  0.001%
Epoch 64  :     Loss of 0.169      Delta = 0.60060%   Epsilon =  0.001%
Epoch 65  :     Loss of 0.168      Delta = 0.58636%   Epsilon =  0.001%
Epoch 66  :     Loss of 0.167      Delta = 0.57254%   Epsilon =  0.001%
Epoch 67  :     Loss of 0.166      Delta = 0.55912%   Epsilon =  0.001%
Epoch 68  :     Loss of 0.166      Delta = 0.54610%   Epsilon =  0.001%
Epoch 69  :     Loss of 0.165      Delta = 0.53346%   Epsilon =  0.001%
---
Epoch 3228:     Loss of 0.103      Delta = 0.00100%   Epsilon =  0.001%
Epoch 3229:     Loss of 0.103      Delta = 0.00100%   Epsilon =  0.001%
Epoch 3230:     Loss of 0.103      Delta = 0.00100%   Epsilon =  0.001%
Epoch 3231:     Loss of 0.103      Delta = 0.00100%   Epsilon =  0.001%

Epochs required: 3231
```

```
Resulting weights:
W0  :    -0.000  (y-intercept)
W1  :     0.349
W2  :    -0.335
W3  :    -0.220
W4  :    -0.321
W5  :    -0.271
W6  :    -0.245
W7  :    -0.157
W8  :     0.260
W9  :     0.505
W10 :     0.051
W11 :     0.046
W12 :    -0.193
W13 :     0.188
W14 :     0.176
W15 :     0.026

Testing Phase:
------------------------------------------------------------
Loss of Testing Data    363.668

Test Record: 1    True: 790.73    Prediction: 800.227   Error: 9.494
Test Record: 2    True: 899.26    Prediction: 923.619   Error: 24.355
Test Record: 3    True: 904.16    Prediction: 881.371   Error: 22.784
Test Record: 4    True: 950.67    Prediction: 926.439   Error: 24.233
Test Record: 5    True: 972.46    Prediction: 927.731   Error: 44.733
Test Record: 6    True: 912.20    Prediction: 900.326   Error: 11.876
Test Record: 7    True: 967.80    Prediction: 1009.285  Error: 41.482
Test Record: 8    True: 823.76    Prediction: 887.051   Error: 63.287
Test Record: 9    True: 1003.50   Prediction: 935.804   Error: 67.698
Test Record: 10   True: 895.70    Prediction: 899.057   Error: 3.361
Test Record: 11   True: 911.82    Prediction: 934.492   Error: 22.675
Test Record: 12   True: 954.44    Prediction: 926.751   Error: 27.691

=> Number of Test Entries (n):         12
Process finished with exit code 0
```

4 Visualization

These data points produced by the loss function show a decrease in error as the algorithm comes closer to convergence.



Batch gradient descent is one of the fastest gradient descent methods to converge on minima. For more complex cases in machine learning, Stochastic Gradient Descent (SGD) or Mini-batch Gradient Descent (MBGD) may be considered.