CPSC429/529: Machine Learning

Program 3: Linear Regression

1 Program Description

In this programming assignment, you can **form a group of two** to implement linear regression model. To test the correctness of your model, you need to apply your linear regression model on two linear regression problems.

The first problem is to predict `rental price` based on descriptive features of `size, floor` and, `broadband rate`. The dataset can be found in `prog3_input1.txt`. You need to do the followings:

1. Using the initial weights (w =([-0.146], [0.185], [-0.044], [0.119])) to calculate the `prediction, error, squared error, erroDelta(D, w[0]), erroDelta(D, w[1]), erroDelta(D, w[2]), erroDelta(D, w[3])`. Output the calculations similar to screenshots on lecture slides.

2. Print out the new weights after the first iteration of gradient descent algorithm, using learning rate $\alpha = 0.00000002$

3. Using the new weights generated from your algorithm to calculate the `prediction, error, squared error, erroDelta(D, w[0]), erroDelta(D, w[1]), erroDelta(D, w[2]), erroDelta(D, w[3])`. Output the calculations similar to screenshots on lecture slides.

4. Print out the new weights after the first iteration of gradient descent algorithm.

5. Print out the new weights after 100 iterations, and the final sum of squared errors.

6. Do the plot between cost function and iterations.

The second problem is to predict `oxyen consumption` (Column 2) based on descriptive features of `age` and, `heart rate`. The dataset can be found in `prog3_input2.txt`. Do the same work as described above, using the initial weights ($w = ([[-59.50], [-0.15], [0.60]])$) and learning rate ($\alpha = 0.000002$).

2 Useful Help

You should not use `scikit-learner` for this program, but you are allowed to use a slightly modified version of linear regression model (`lr_house.py`) I provide to you, based on online source code.

The online *Machine Learning with Python - Linear Regression* can be found at `http://aimotion.blogspot.com/2011/10/machine-learning-with-python-linear.html`. Read the tutorial and understand how linear regression can be used for predicting house prediction using dataset `ex1data2.txt`.

You can use the majority of source code for your program, and modify based on that. The main goal of this program is to understand how the gradient descent algorithm is implemented, and dig into the details of the gradient descent algorithm by printing out `errors, deltaErrors` for each iteration. You need to pay attention to the usage of `dot()`, `transpose()`, matrix addition, matrix multiplication, array cancatenation with `numpy`.

Important: Our program uses the following cost function and gradient update rule:

$$L_2(M_W, D) = \frac{1}{2}\sum_{i=1}^{m}(t^{(i)} - M_W(x^{(i)}))^2 \tag{1}$$

$$W_j = W_j + \alpha * errorDelta(D, W_j) \tag{2}$$

3 Submission

Demo your program during class time, and upload the following items on D2L dropbox, including:

1. The source code (.py code).
2. Program outputs and plots for input datasets of `prog3_input1.txt` and `prog3_input2.txt`.

Note: If you programmed with another group member, only one submission is sufficient. Make a note (your partner name) in your submission comment.

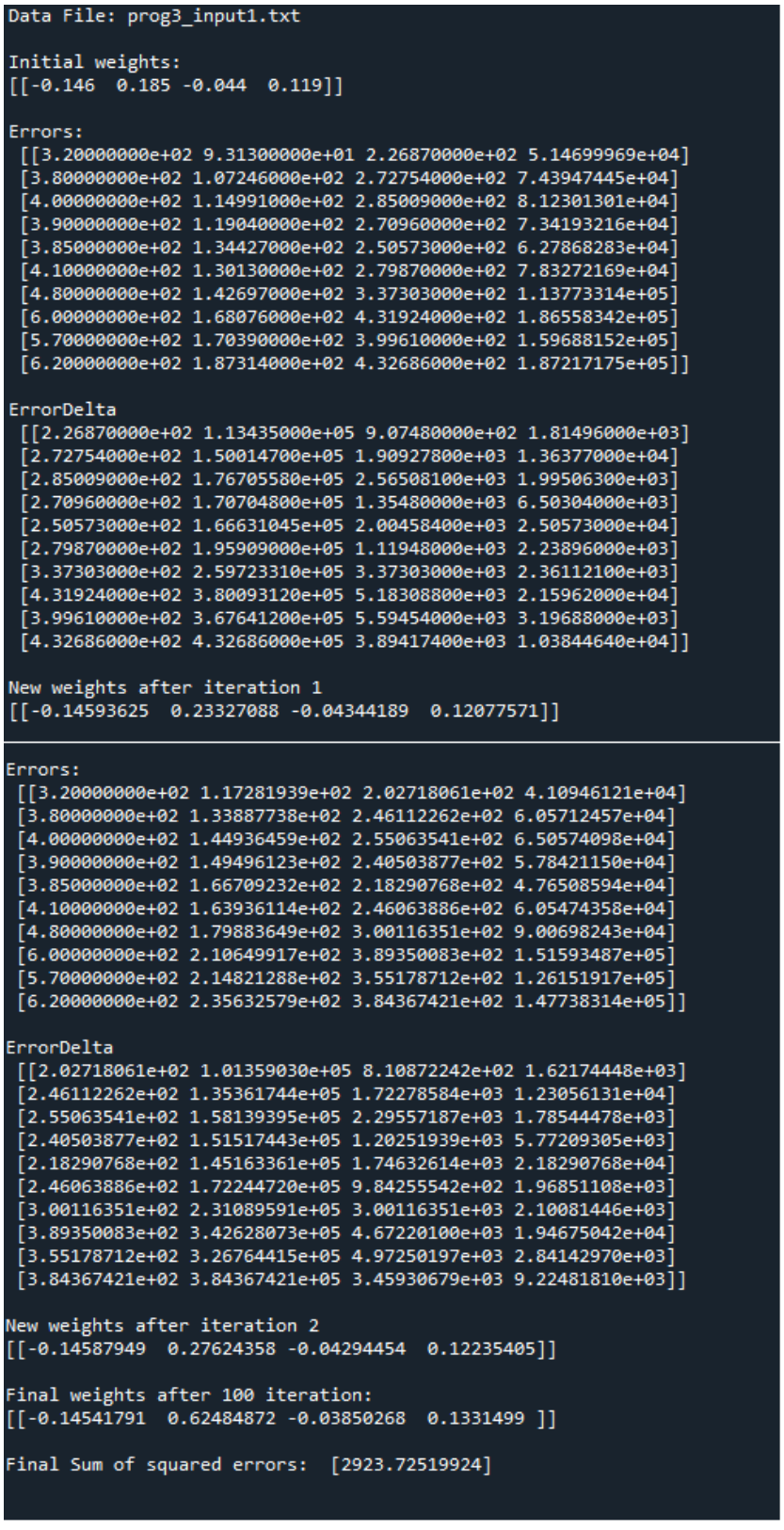

Figure 1: A sample program output on "`prog3_input1.txt`"

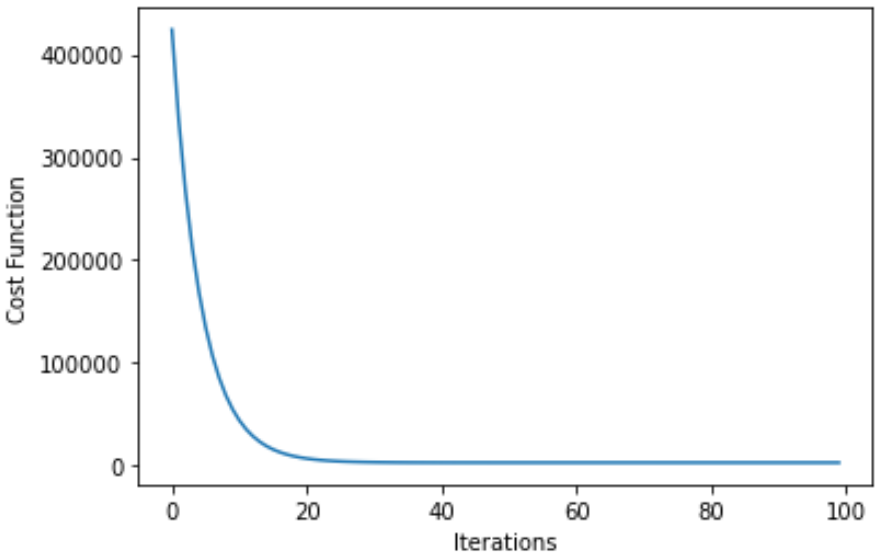

Figure 2: A sample plot on "`prog3_input1.txt`"

```
Data File: prog3_input2.txt

Initial weights:
[[-59.5   -0.15   0.6 ]]

Errors:
 [[ 37.99    17.15    20.84   434.3056]
 [ 47.34    26.      21.34   455.3956]
 [ 44.38    25.55    18.83   354.5689]
 [ 28.17    13.4     14.77   218.1529]
 [ 27.07     8.9     18.17   330.1489]
 [ 37.85    20.9     16.95   287.3025]
 [ 44.72    28.85    15.87   251.8569]
 [ 36.42    19.4     17.02   289.6804]
 [ 31.21    17.75    13.46   181.1716]
 [ 54.85    29.6     25.25   637.5625]
 [ 39.84    19.85    19.99   399.6001]
 [ 30.83    16.85    13.98   195.4404]]

ErrorDelta
 [[  20.84  854.44 2875.92]
 [  21.34  896.28 3265.02]
 [  18.83  696.71 2843.33]
 [  14.77  679.42 1964.41]
 [  18.17  872.16 2289.42]
 [  16.95  745.8  2457.75]
 [  15.87  682.41 2507.46]
 [  17.02  782.92 2433.86]
 [  13.46  498.02 1857.48]
 [  25.25  959.5  3989.5 ]
 [  19.99  859.57 2858.57]
 [  13.98  601.14 1929.24]]

New weights after iteration 1
[[-59.49956706  -0.13174326   0.66254392]]

Errors:
 [[ 37.99        26.53002024  11.45997976 131.3311361 ]
 [ 47.34        36.33643578  11.00356422 121.07842554]
 [ 44.38        35.67006424   8.70993576  75.86298094]
 [ 28.17        22.55858434   5.61141566  31.48798571]
 [ 27.07        17.65729038   9.41270962  88.59910239]
 [ 37.85        30.7725979    7.0774021   50.08962049]
 [ 44.72        39.51741212   5.20258788  27.06692065]
 [ 36.42        29.18402354   7.23597646  52.35935533]
 [ 31.21        27.05699328   4.15300672  17.24746482]
 [ 54.85        40.17612842  14.67387158 215.32250715]
 [ 39.84        29.57925332  10.26074668 105.28292243]
 [ 30.83        26.26653372   4.56346628  20.82522449]]

ErrorDelta
 [[  11.45997976  469.85917016 1581.47720688]
 [  11.00356422  462.14969724 1683.54532566]
 [   8.70993576  322.26762312 1315.20029976]
 [   5.61141566  258.12512036  746.31828278]
 [   9.41270962  451.81006176 1186.00141212]
 [   7.0774021   311.4056924  1026.2233045 ]
 [   5.20258788  223.71127884  822.00888504]
 [   7.23597646  332.85491716 1034.74463378]
 [   4.15300672  153.66124864  573.11492736]
 [  14.67387158  557.60712004 2318.47170964]
 [  10.26074668  441.21210724 1467.28677524]
 [   4.56346628  196.22905004  629.75834664]]

New weights after iteration 2
[[-59.49936833  -0.12338147   0.69131222]]

Final weights after 100 iteration:
[[-59.4992592   -0.12200374   0.71748235]]

Final Sum of squared errors:  [52.68931138]
```

Figure 3: A sample program output on "`prog3_input2.txt`"

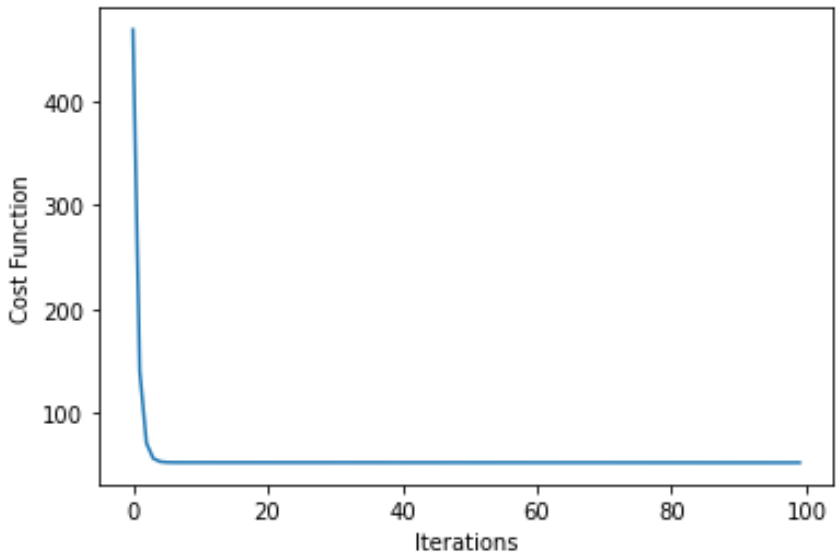

Figure 4: A sample plot on "`prog3_input2.txt`"