CS145 HW1 Report
Shunji Zhan
405030387

Linear Regression:

No.

DATE . .

Linear Regression

$$X = \begin{bmatrix} 1 & 60 \\ 1 & 70 \\ 1 & 62 \\ 1 & 72 \\ 1 & 65 \end{bmatrix} \quad y = \begin{bmatrix} 130 \\ 155 \\ 125 \\ 162 \\ 150 \end{bmatrix}$$

$$X^T X = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 60 & 70 & 62 & 72 & 65 \end{bmatrix} \begin{bmatrix} 1 & 60 \\ 1 & 70 \\ 1 & 62 \\ 1 & 72 \\ 1 & 65 \end{bmatrix} = \begin{bmatrix} 5 & 329 \\ 329 & 21753 \end{bmatrix}$$

$$(X^T X)^{-1} = \frac{1}{5 \times 21753 - 329^2} \begin{bmatrix} 21753 & -329 \\ -329 & 5 \end{bmatrix} = \frac{1}{524} \begin{bmatrix} 21753 & -329 \\ -329 & 5 \end{bmatrix}$$

$$X^T y = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 60 & 70 & 62 & 72 & 65 \end{bmatrix} \begin{bmatrix} 130 \\ 155 \\ 125 \\ 162 \\ 150 \end{bmatrix} = \begin{bmatrix} 722 \\ 47814 \end{bmatrix}$$

$$\hat{\beta} = (X^T X)^{-1} X^T y = \frac{1}{524} \begin{bmatrix} 21753 & -329 \\ -329 & 5 \end{bmatrix} \begin{bmatrix} 722 \\ 47814 \end{bmatrix} = \begin{bmatrix} -47.98 \\ 2.92 \end{bmatrix}$$

$$\hat{Y} = X\hat{\beta} = \begin{bmatrix} 1 & 60 \\ 1 & 70 \\ 1 & 62 \\ 1 & 72 \\ 1 & 65 \end{bmatrix} \begin{bmatrix} -47.98 \\ 2.92 \end{bmatrix} = \begin{bmatrix} 127.5 \\ 156.7 \\ 133.3 \\ 162.5 \\ 142.1 \end{bmatrix} \begin{matrix} \text{Student 1} \\ \text{Student 2} \\ \text{Student 3} \\ \text{Student 4} \\ \text{Student 5} \end{matrix}$$

```
Result:
-----------------------------------------------
Closed Form Without Normalization
Beta:
[-0.0862246   0.05340575  0.65803045  0.41731923 -0.01772481  0.30069864
  1.02871152  0.48383363  0.26685697  0.04573456  0.31944742  1.14776959
  0.29366213  0.41491543  0.85180482 -0.05950309  0.47235562  0.46198106
  0.00497427  0.0205398   0.41310473  0.98508025  0.15573467  0.8618602
  0.41974331 -0.06893699  0.33317496  0.27766637 -0.04184791 -0.23599504
  0.15020297  0.37745027  0.80256455  0.16053288  0.2744667   0.63461071
  0.74135259  0.56079776  0.94058723 -0.0432542   0.80803615  0.93967722
  0.12225161 -0.19933624  0.09398732  0.11412993  0.35479619  0.78582876
  0.38900433  0.11804526  0.67618837  0.70377377  0.05526258 -0.24919095
  0.87339793 -0.01381723  0.83138416  0.90569236  0.39980648  0.25235308
  0.69692397 -0.00949757  0.17676599  0.45822485  0.02743899  1.16718165
  0.04176352  1.01993881  0.56015024 -0.29761224  0.3177761   0.55781578
  1.1376088   0.55190283  0.4099807   0.91987238  1.34076835  0.53297825
  0.63648277  0.22140583  0.21469531 -0.00609269  0.82898663  0.46891532
 -0.25571565  0.1972989   1.38639797  0.87219453  0.65782257  0.54983464
  1.11698567  0.94267463  0.79030138  0.30055848  0.53288973  0.22873689
  0.86702876  0.98591924  0.08132528  0.30834368  0.70121488]

MSE:  4.39609786082
-----------------------------------------------
Batch Gradient Without Normalization
Beta:
[ 0.22766935  0.04926912  0.49075821  0.53316896  0.03312045  0.4291802
  0.92704285  0.38053823  0.54931205  0.32240208  0.36203713  1.02347259
  0.33157719  0.53755143  0.79293124  0.34421494  0.30348383  0.46472404
  0.29458572  0.26602136  0.54775439  0.90914347  0.28603106  0.50597984
  0.42788019  0.03880525  0.1978098   0.45151963  0.12783095  0.306327
  0.27974695  0.23015537  0.44117103  0.27154554  0.38328989  0.3524848
  0.50892727  0.43214341  0.83088654  0.26915302  0.6026654   0.88776574
  0.10887242  0.1701851   0.11400057  0.23809648  0.40552648  0.69369281
  0.44956139  0.32764342  0.66680446  0.37717538  0.05244999 -0.01382757
  0.58671581  0.20255529  0.88555671  0.65250047  0.32283663  0.1529379
  0.58433231  0.38708742  0.38569954  0.49123304  0.23010377  0.89345829
  0.04454032  0.74480553  0.68698152 -0.11399285  0.52433376  0.57866989
  0.78585681  0.34671584  0.4832255   0.6288695   0.95743216  0.64639908
  0.79010401  0.15680372  0.19309826  0.29357565  0.63720365  0.33732385
 -0.16104665  0.10673337  1.01468903  0.76834802  0.51525088  0.44584999
  0.79544821  0.78357295  0.75251839  0.55825906  0.59423384  0.20030567
  0.48726899  0.75329125  0.07284554  0.35503306  0.69429324]
```

MSE: 4.37759592671

```
-------------------------------------------------
```

Stochastic Gradient Without Normalization

Beta:

```
[-0.01185402  0.05502465  0.59066682  0.51656768  0.3127531   0.42484285
  0.77140998  0.6116552   0.58555048  0.0437524   0.51680984  0.99282054
  0.52586939  0.46452354  0.81679006 -0.02168758  0.49846207  0.52072191
  0.14213335  0.36819207  0.25994595  0.70263936  0.24072741  0.55881525
  0.54218245  0.35382061  0.48651257  0.49048785  0.11264291  0.17736387
  0.18872542  0.18502001  0.46833438  0.10578863  0.42944319  0.41399556
  0.43193366  0.4281315   0.81539793  0.17171139  0.76674725  0.8820437
  0.44211809  0.09868028  0.36049616  0.1160615   0.44482727  0.72706628
  0.46053535  0.06682069  0.74305567  0.44836829  0.14009563 -0.01874977
  0.72614565 -0.01069598  0.69121445  0.63311067  0.24875391  0.14697651
  0.55263444  0.02686585  0.34318297  0.60920143  0.44055768  1.10250063
  0.07929965  0.92723962  0.65397616  0.11989741  0.46731019  0.67822534
  0.92499347  0.36428803  0.29838152  0.72442646  0.96082815  0.50163245
  0.43303453  0.16550037  0.18995654  0.30379008  0.47154151  0.47277384
  0.12451971  0.48115684  0.79064542  0.77181377  0.48166054  0.55451245
  0.53549879  0.64948011  0.79964462  0.28620202  0.56009697  0.26445081
  0.79842373  0.56042837  0.35628587  0.15468497  0.6152241 ]
```

MSE: 4.55321973906

```
-------------------------------------------------
```

Closed Form With Normalization

Beta:

```
[  2.27729720e+01   1.53267685e-01   1.85400036e-01   1.20001101e-01
  -5.02894960e-03   8.91855522e-02   2.85477509e-01   1.40249729e-01
   7.58001703e-02   1.29653087e-02   9.40114997e-02   3.31501951e-01
   8.48405150e-02   1.19998020e-01   2.42101087e-01  -1.70904428e-02
   1.37119556e-01   1.35350218e-01   1.41619004e-03   5.96423043e-03
   1.15830867e-01   2.84837752e-01   4.40248244e-02   2.49185633e-01
   1.20285952e-01  -1.97966211e-02   9.78939759e-02   8.05403060e-02
  -1.21241111e-02  -6.77059821e-02   4.42940642e-02   1.07814670e-01
   2.27982170e-01   4.72154203e-02   7.98729034e-02   1.82957097e-01
   2.10609705e-01   1.62079663e-01   2.74455584e-01  -1.24456123e-02
   2.32346197e-01   2.68821067e-01   3.49745502e-02  -5.73174263e-02
   2.74558199e-02   3.22366923e-02   1.03219840e-01   2.23792899e-01
   1.12445398e-01   3.34223468e-02   1.96611852e-01   2.04171370e-01
   1.61259528e-02  -7.12316220e-02   2.51757075e-01  -3.88735810e-03
   2.31055679e-01   2.65481860e-01   1.14239087e-01   7.19519080e-02
   2.03225977e-01  -2.77922653e-03   5.10043840e-02   1.31478537e-01
   7.74623329e-03   3.36781203e-01   1.19518825e-02   2.98145298e-01
   1.64253970e-01  -8.57326109e-02   9.04810592e-02   1.57878654e-01
```

```
   3.30578812e-01   1.58142457e-01   1.17519641e-01   2.66603450e-01
   3.90619100e-01   1.54573813e-01   1.82230684e-01   6.25165215e-02
   6.11873098e-02  -1.74345404e-03   2.34361003e-01   1.35158424e-01
  -7.34879378e-02   5.72871764e-02   4.02966409e-01   2.50642329e-01
   1.87572968e-01   1.57855445e-01   3.18225717e-01   2.66144412e-01
   2.29911686e-01   8.53225833e-02   1.56706806e-01   6.57087894e-02
   2.52781623e-01   2.90068806e-01   2.33284776e-02   9.01229905e-02
   2.03998476e-01]

MSE:  4.40454594906
---------------------------------------------
Batch Gradient With Normalization
Beta:
[  2.26314370e+01   1.56472069e-01   1.66211092e-01   1.22311607e-01
  -1.23730807e-03   9.27912698e-02   3.00889509e-01   1.71530166e-01
   9.65016213e-02   1.24019695e-02   9.73546485e-02   3.44150847e-01
   7.37193551e-02   1.34926956e-01   2.50479685e-01   4.44237824e-03
   1.29705174e-01   1.33816710e-01  -1.20262171e-03   1.27782110e-02
   1.09214197e-01   2.77963281e-01   5.85014678e-02   2.59493309e-01
   1.23482024e-01  -1.01901790e-02   9.61310754e-02   7.91369297e-02
   1.78208772e-02  -4.02205054e-02   2.57410489e-02   1.20550801e-01
   2.45588378e-01   4.94254324e-02   8.84012432e-02   1.87802672e-01
   1.91767489e-01   1.66042065e-01   2.77006607e-01  -1.13363276e-02
   2.35250747e-01   2.58430264e-01   1.21489249e-02  -3.35672386e-02
   3.08828976e-02   2.93577082e-02   1.08348343e-01   2.19164328e-01
   1.02330247e-01   2.51918396e-02   2.01546865e-01   2.05967393e-01
   2.44029928e-02  -6.19076257e-02   2.48507437e-01   9.27691703e-03
   2.46905873e-01   2.85875947e-01   1.29394746e-01   8.75080646e-02
   2.10823417e-01   2.02666978e-02   4.14111150e-02   1.29141830e-01
   1.91198779e-02   3.62870196e-01   2.24001342e-02   3.12990830e-01
   1.77761813e-01  -7.39828048e-02   8.21730110e-02   1.77154451e-01
   3.36492024e-01   1.74811868e-01   1.08921172e-01   2.63607960e-01
   4.09818557e-01   1.57795545e-01   1.97303661e-01   5.37669575e-02
   5.21850983e-02   3.58247178e-03   2.52169011e-01   1.53033209e-01
  -6.76204718e-02   7.82286698e-02   3.92781597e-01   2.44867492e-01
   1.91773252e-01   1.60529496e-01   3.19988309e-01   2.60654717e-01
   2.34634115e-01   8.75464460e-02   1.55468524e-01   7.99155768e-02
   2.63419427e-01   2.73817631e-01   3.71941936e-02   9.15361359e-02
   1.99412688e-01]

MSE:  4.35965874223
---------------------------------------------
Stochastic Gradient With Normalization
Beta:
```

```
[  2.26355654e+01   1.67994448e-01   1.66007867e-01   1.15506373e-01
  -4.92549005e-03   8.47808070e-02   2.88997721e-01   1.60296291e-01
   9.15977394e-02   2.58892409e-02   1.09694277e-01   3.45871525e-01
   8.68438408e-02   1.40420749e-01   2.38082692e-01  -8.04233387e-04
   1.40324029e-01   1.37910026e-01   2.14854946e-02   1.99968512e-02
   1.16408187e-01   2.90037213e-01   7.62915221e-02   2.49970230e-01
   1.11734009e-01  -1.52193845e-02   1.03093756e-01   9.98040109e-02
   2.15743666e-02  -4.00074942e-02   3.23151751e-02   1.20785461e-01
   2.27624002e-01   2.92833953e-02   9.41565155e-02   2.01272895e-01
   2.04297445e-01   1.70770785e-01   2.84032205e-01  -9.14514482e-03
   2.41035240e-01   2.65684584e-01   2.79125120e-02  -5.51663617e-02
   4.50753662e-02   2.75059553e-02   9.19485262e-02   2.23173183e-01
   1.07987631e-01   6.82650987e-02   1.97947076e-01   1.95971691e-01
   3.71991739e-02  -5.33601320e-02   2.50406040e-01  -2.80571896e-03
   2.33427493e-01   2.80246295e-01   1.24864923e-01   6.52156543e-02
   2.03020326e-01   3.69006429e-03   4.10700125e-02   1.35701587e-01
   3.80019248e-02   3.70744769e-01   2.60547509e-02   3.01661475e-01
   1.68269267e-01  -8.11566132e-02   9.02707702e-02   1.69281382e-01
   3.52828067e-01   1.65284950e-01   1.03858647e-01   2.71194237e-01
   4.01866382e-01   1.43872869e-01   1.79486330e-01   6.53913236e-02
   6.86866415e-02  -6.30923490e-06   2.41026136e-01   1.53596458e-01
  -6.39901983e-02   5.76930803e-02   4.09508578e-01   2.34961381e-01
   1.86717336e-01   1.77487273e-01   3.22373537e-01   2.67346795e-01
   2.44475587e-01   8.76909995e-02   1.51501960e-01   8.94086333e-02
   2.65467442e-01   2.85651782e-01   3.26714489e-02   9.48052948e-02
   1.95383968e-01]
```

MSE: 4.3888817496

For each version of the linear model, Beta and MSE are not the same. The reason is that for the same dataset there are lots of possible beta to construct a good linear model, the beta doesn't have to be the same. And different algorithms result in different Betas, so that the difference between real y and predicted y are slightly different in these algorithms, resulting in different MSE.

After applying the z score normalization, we saw that the beta changed totally. And from the result we can see that the MSE decrease really slightly, compared to non-normalized data.

Code Explanation:

compute_mse():

I just used the formula to compute each difference between predict_y and real_y, sqaure them and add together.

getBeta():

this is straight forward, just plug in the formula and return (X^T*X)^ -1)*(X^T*Y)

getBetaBatchGradient():
I set up a infinite loop and update beta using the formula in each iteration. I implemented a helper function derivative() to make the formula looks more clean.
In each iteration I calculated the previous cost and current cost after updating beta, if calculate the increase in cost. If the increase is less than 1 I will stop the iteration and return the beta.

getBetaStochasticGradient():
I set up a infinite loop and update beta using the formula in each iteration. What's different from batch gradient was that in each loop, each sample xi will update the beta, so it updates more constantly.
In each iteration I calculated the previous cost and current cost after updating beta, if calculate the increase in cost. If the increase is less than 1 I will stop the iteration and return the beta.

applyZScore():
use the formula directly

Logistic Regression:

$ python logicsticRegression.py
[-19.72536249   76.65485643 -45.49392387]

Logistic Regression

a) $$L = \sum_{i=0}^{2} y_i X_i^T \beta - \log\left(1 + e^{X_i^T \beta}\right)$$

where:

$$\beta = \begin{pmatrix} \beta_{10} \\ \beta_{11} \\ \beta_{12} \end{pmatrix} \quad Y = \begin{pmatrix} y_0 \\ y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} \quad X = \begin{pmatrix} X_0^T \\ X_1^T \\ X_2^T \end{pmatrix} = \begin{bmatrix} (1, 60, 155) \\ (1, 64, 135) \\ (1, 73, 170) \end{bmatrix}$$

b) $$\frac{\partial L(\beta)}{\beta_{10}} = \sum_{i=0}^{2} x_{i0}\left(y_i - \frac{\exp(\beta^T x_i)}{1 + \exp(\beta^T x_i)}\right)$$

$$\frac{\partial L(\beta)}{\beta_{11}} = \sum_{i=0}^{2} x_{i1}\left(y_i - \frac{\exp(\beta^T x_i)}{1 + \exp(\beta^T x_i)}\right)$$

$$\frac{\partial L(\beta)}{\beta_{12}} = \sum_{i=0}^{2} x_{i2}\left(y_i - \frac{\exp(\beta^T x_i)}{1 + \exp(\beta^T x_i)}\right)$$

After expansion, $L =$

$$\boxed{2\beta_0 + 137\beta_1 + 305\beta_2 - \log\left[(1+\beta_0+60\beta_1+155\beta_2)\cdot(1+\beta_0+64\beta_1+135\beta_2)\cdot(1+\beta_0+73\beta_1+170\beta_2)\right]}$$

$$\frac{\partial L(\beta)}{\partial \beta_{1j}\partial \beta_{1n}} = -\sum_{i=0}^{2} x_{ij}x_{in}\, p(x_i|\beta)\left(1-p(x_i|\beta)\right)$$

$$= -\sum_{i=0}^{2} x_{ij}\, x_{in} \frac{\exp(\beta^T x_i)}{[1+\exp(\beta^T x_i)]^2}$$

Hessian Matrix =

Let $\underline{\underline{A}}$ denote $\frac{\exp(\beta^T x_i)}{[1+\exp(\beta^T x)]^2}$

$$\begin{bmatrix} -\sum x_{i0}x_{i0}A & -\sum x_{i0}x_{i1}A & -\sum x_{i0}x_{i2}A \\ -\sum x_{i1}x_{i0}A & -\sum x_{i1}x_{i1}A & -\sum x_{i1}x_{i2}A \\ -\sum x_{i2}x_{i0}A & -\sum x_{i2}x_{i1}A & -\sum x_{i2}x_{i2}A \end{bmatrix}$$

In the codes, I just followed the formula, for the gradients I calculated directly, and for the Hessian Matrix I used a double for loop and update each of the item in the 3 * 3 matrix, one by one.

Decision Tree:

No.

DATE

Decision Tree

feature A: vote for handicapped-infants
feature B: vote for water-project-cost-sharing
feature C: vote for budget-resolution-adoption.

$Info(D) = I(20,10) = 1.0$

$Info_A(D) = \frac{8}{20} I(6,2) + \frac{12}{20} I(8,4) = 0.875$
$Info_B(D) = \frac{10}{20} I(6,4) + \frac{10}{20} I(6,4) = 0.971$
$Info_C(D) = \frac{11}{20} I(9,2) + \frac{9}{20} I(8,1) = 0.603$

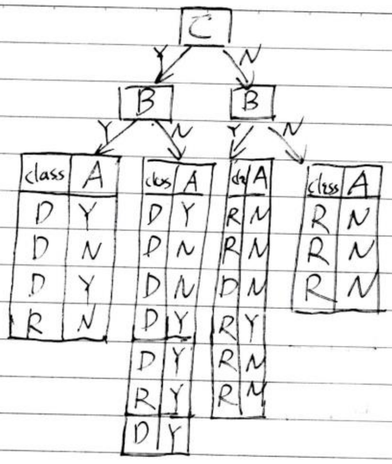
$Gain(A) = 1 - 0.875 = 0.125$
$Gain(B) = 1 - 0.971 = 0.029$
$Gain(C) = 1 - 0.603 = 0.397$ Gain(C) is largest.

$$\begin{tikzcd} & \boxed{\text{vote for budget-resolution-adoption}} \arrow[dl, "Y"'] \arrow[dr, "N"] & \\ \boxed{L_1} & & \boxed{R_1} \end{tikzcd}$$

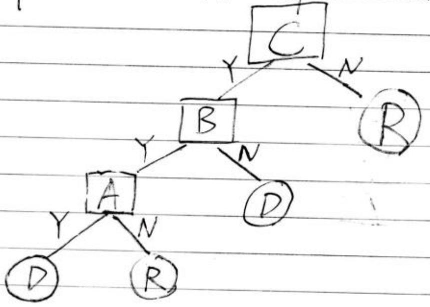
$Info_A(L_1) = \frac{7}{11}(6,1) + \frac{4}{11}(3,1) = 0.369$ $Info_A(R_1) = \frac{8}{9}(7,1) + \frac{1}{9}(1,0) = 0.217$
$Info_B(L_1) = \frac{4}{11}(3,1) + \frac{7}{11}(6,1) = 0.369$ $Info_B(R_1) = \frac{6}{9}(5,1) + \frac{3}{9}(3,0) = 0.195$

$InfoA(L_1) = InfoB(L_1)$
so split with A or B

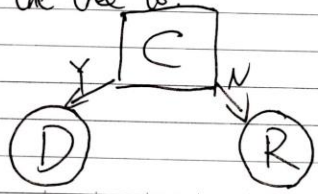
$InfoA(R_1) > InfoB(R_1)$
so split with B



Repeat the similar step we will finally get:



On the left branch if we pick D on the third level on equal gain, we can simplify the tree to:



```
$ python3 DecisionTree.py house-votes-84.data 0
Fold-1: 0.9310344827586207
Fold-2: 0.9080459770114943
Fold-3: 0.9655172413793104
Fold-4: 0.8850574712643678
Fold-5: 0.9310344827586207
5-CV Accuracy = 0.9241379310344827

$ python3 DecisionTree.py house-votes-84.data 1
Fold-1: 0.9195402298850575
Fold-2: 0.9195402298850575
Fold-3: 0.9540229885057471
Fold-4: 0.8850574712643678
Fold-5: 0.9310344827586207
5-CV Accuracy = 0.9218390804597701

$ python3 DecisionTree.py tic-tac-toe.data 0
Fold-1: 0.875
Fold-2: 0.8333333333333334
Fold-3: 0.8333333333333334
Fold-4: 0.8167539267015707
Fold-5: 0.8272251308900523
5-CV Accuracy = 0.837129144851658

$ python3 DecisionTree.py tic-tac-toe.data 1
Fold-1: 0.8802083333333334
Fold-2: 0.8229166666666666
Fold-3: 0.8177083333333334
Fold-4: 0.837696335078534
Fold-5: 0.837696335078534
5-CV Accuracy = 0.8392452006980804
```

For the house-vote dataset I would choose information gain method, since the accuracy is slightly higher. On the other hand, for the tic-tac-toe dataset I would choose gain ratio method, since this method produce better accuracy.