Due Date:2019/12/17

Linear Regression – 30%

1. Description

   Write a linear regression program to find the best fitting line on the data with the matrix operations which are written by yourself.

2. Input Parameter

   (a) A set of data points(comma seperated :x,y):

```
-5.0,51.76405234596766
-4.795918367346939,45.42306433039972
-4.591836734693878,41.274448104888755
```

   Figure 1: Data points

   (b) The number of polynomial bases n:

   For example, if the number of polynomial bases is 3, then we are going to find the curve that fits the data points by $\mathrm{ax}^2 + bx^1 + cx^0 = y$.

3. Function Requirement

$$
\begin{aligned}
Ax &= b \\
x &= (A^T A)^{-1} A^T b \\
x &= A^{-1} b
\end{aligned}
$$

   You should implement the formula above with your **handcrafted inverse function**, you can use LU decomposition or Gauss-Jordan elimination to find the inverse matrix and plot the result including points and the fitting line.

4. Grading Policy

   (a) The weight of your fitting line. 10%

   (b) The error(L2-norm) of your fitting line. 10%

   (c) Plot your result with datapoints and your fitting line. 10%

5. Sample Input & Output

   (a) Input:

```
-5,-140.8244956385661
-4.69132,-123.31105767143112
-4.38264,-106.88511830330546
-4.07396,-91.54667753418906
-3.76528,-77.29573536408205
-3.4566,-64.13229179298432
-3.14792,-52.05634682089592
-2.83924,-41.06790044781682
-2.53056,-31.166952673747026
-2.22188,-22.353503498686564
-1.9132,-14.627552922635395
-1.60452,-7.989100945593549
-1.29584,-2.4381475675610105
-0.98716,2.0253072114622146
-0.67848,5.401263391476128
-0.3698,7.689720972480728
-0.06112,8.890679954476017
0.24756,9.004140337461992
0.55624,8.030102121438656
0.86492,5.968565306406006
1.1736,2.819529892364045
1.48228,-1.4170041206872313
1.79096,-6.741036732747816
2.09964,-13.152567943817708
2.40832,-20.651597753896922
2.717,-29.238126162985445
3.02568,-38.912153171083276
3.33436,-49.67367877819044
3.64304,-61.522702984306896
3.95172,-74.45922578943265
4.2604,-88.48324719356773
4.56908,-103.59476719671213
4.87776,-119.79378579886585
```

(b) Output:

```
Fitting line: 2.1290970665110547X^1 + -40.27912612124099
Total error: 63987.84575778297
```

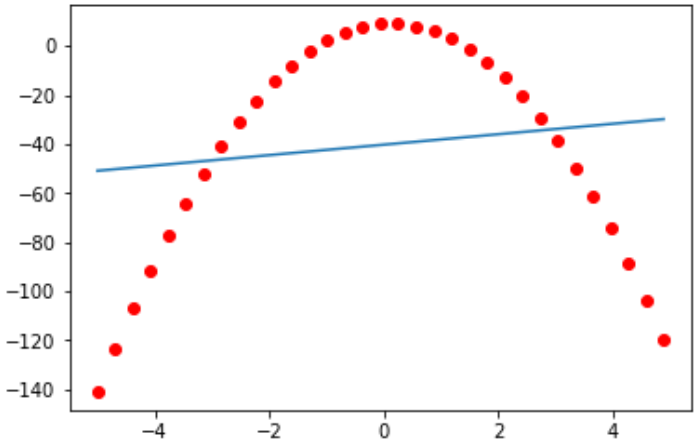

Figure 2: n = 2

```
Fitting line: -5.706656439716332X^2 + 1.43151538332013X^1 + 8.999492270942884
Total error: 1.3819265337661685e-26
```

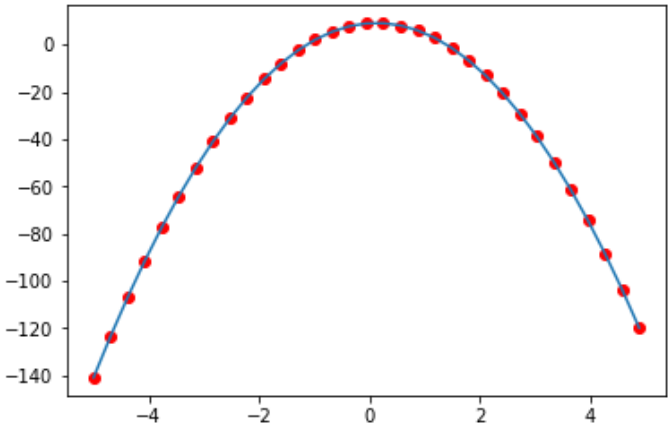

Figure 3: n = 3

Logistic Regression – 70%

1. Description

   Write a logistic regression program with two different error functions, L2-norm and cross entropy, to classify the data correctly. Plot the result and compare those two methods.

2. Input parameter

   (a) A set of data points(comma seperated :x,y):

   (b) The parameter to choose which method using:

3. Method

   Both L2-norm and cross entropy will do gradient descent to find the optimal result. If you are confused the formulas below, you should check the detail of these formulas in 05_LogisticRegression.pdf.

   (a) Sigmoid function

$$M_{\mathbf{w}}(\mathbf{x}) = L(\mathbf{w} \cdot \mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w} \cdot \mathbf{x}}}$$

   (b) L2-norm

$$w_j' = -\frac{\partial \sum_{i=1}^{m} \frac{1}{2} \left(y_i - M_{\mathbf{w}} \left(\mathbf{x}_i\right)\right)^2}{\partial w_j}$$

$$w_j' = \sum_{i=1}^{m} \left(y_i - M_{\mathrm{w}} \left(\mathbf{x}_i\right)\right) M_{\mathrm{w}} \left(\mathbf{x}_i\right) \left(1 - M_{\mathrm{w}} \left(\mathbf{x}_i\right)\right) x_{ij}$$

$$w_j = w_j + \alpha w_j'$$

   Here are the explanation of the parameters:

   - m: Numbers of your data.
   - j: The dimension which set by your data dimension. For example, if data has two dimension(x,y), then we will have three terms of w, one for x axis, one for y axis and the other is for bias.
   - y: Target. It will be 0 or 1 in this case.
   - x: The data points.
   - w: The weight of your result.

The steps of logistic regression is:

i. You got the data points, which is **x** in our formula, for example, we have some (x,y) points, we might make it become a vector like [x,y,1]. The last one will be the bias.

ii. Then it will do dot product with our **w**, which is a 3 by 1 vector here. Then put them into the sigmoid function, it will give you a result between 0 and 1.

iii. Now you have the target **y** and the predict result **y'**, just use two different error function doing the gradient descent, both two error function's result is given.

iv. Until your loss smaller than your threshold, you get the **w**.

(c) Cross Entropy

$$G(\mathbf{w}) = \frac{1}{m}\sum_{i=1}^{m} -y_i \log M_{\mathbf{w}}(\mathbf{x}_i) - (1-y_i)\log\left(1 - M_{\mathbf{w}}(\mathbf{x}_i)\right)$$

$$\frac{\partial G(L)}{\partial w_j} = \frac{1}{m}\sum_{i=1}^{m}\left(M_{\mathbf{w}}(\mathbf{x}_i) - y_i\right)x_{ij}$$

$$w_j = w_j + \alpha w_j'$$

Same as the SOP above, cross entropy will be easier to implement. You can find out that the result of the equation is shorter than l2-norm.

4. Grading Policy
   (a) Plot the ground truth of data points. 5%
   (b) The confusion matrix, precision and recall with the target. 25%
   (c) The weight after using gradient descent. 25%
   (d) Plot the result of your classification. 15%
5. Sample Input & Output
   (a) Input:

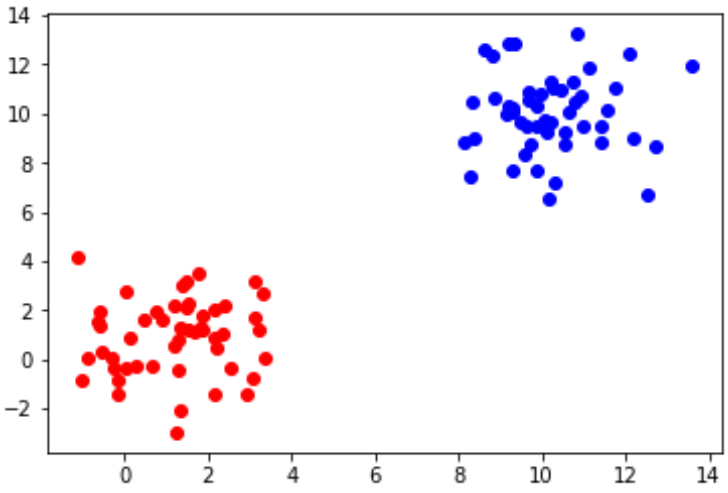

Figure 4: Data input example1

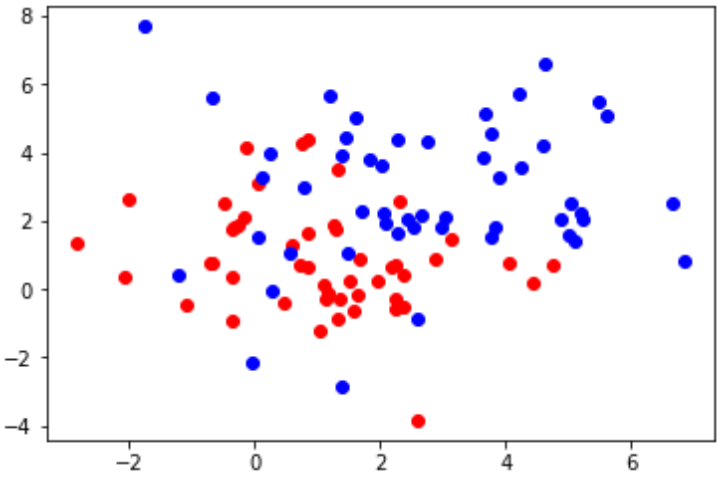

Figure 5: Data input example2

(b) Output:

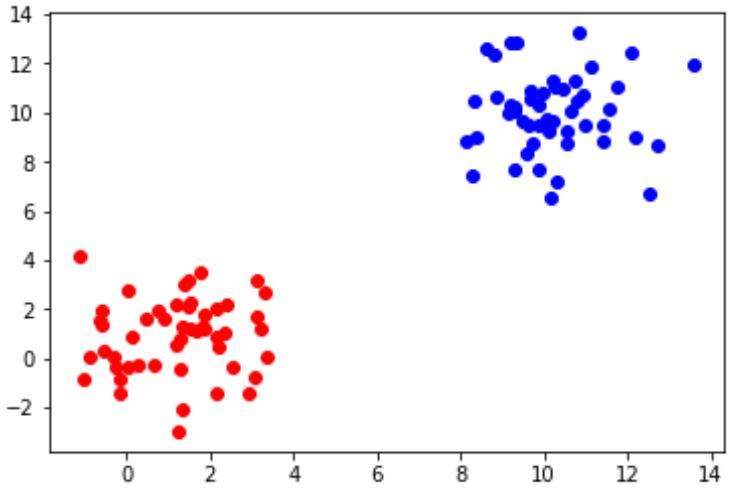

Figure 6: Data output example1

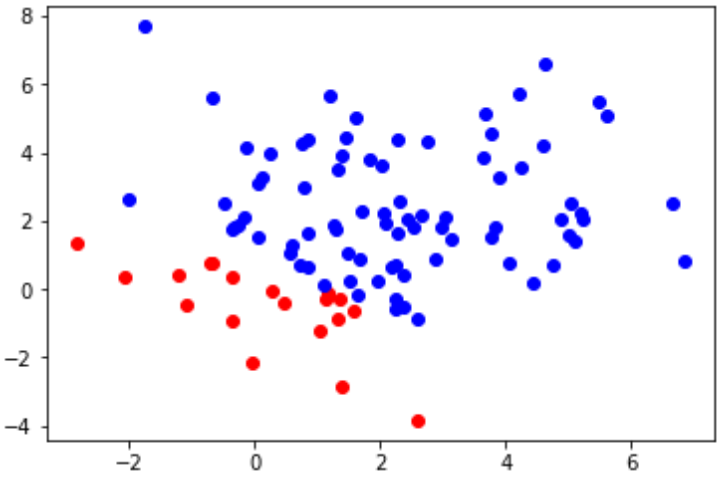

Figure 7: Data output example2

```
Confusion Matrix:
                    Is cluster 1    Is cluster 2
predict cluster 1        16              34
predict cluster 2         4              46

Precision :  0.32
Recall :  0.8
```

Figure 8: Result for example2

Your answer may be different from others even you are right, just make sure your loss as small as possible. If you have any problem of homework, just feel free to ask TA.

E3 Submission

Please submit a **zip** file, which contains the following, to the newE3 system.

1. Report
   (a) Explanation of how your code works.
   (b) All the content mentioned above.
   (c) Your name and student ID.
   (d) Accept formats: PDF, HTML
2. Source codes
   (a) Accept languages: C/C++, C#, R, python2/python3, MATLAB, JAVA
   (b) NO package-provided model allowed, so the algorithm must be done by you.

   However, numerical computing and visualization packages are allowed, solely for data manipulation and visualization. Take python3 for example:
   i. sklearn X
   ii. numpy O
   iii. pandas O
   iv. matplotlib O

- Your score will be determined mainly by the submitted report.
  - If there is any problem with your code, TA might ask you (through email) to demo it. Otherwise, no demo needed.
- Plagiarizing is not allowed.
  - You will get ZERO on that homework if you get caught the first time.

  - The second time, you'll FAIL this class.

- In order to meet the grading rules the school formulated, your final score of this homework will be normalized by the following formula:
  final_score = (raw_score - raw_score.mean()) * 15. / raw_score.std() + 80

My email address: barry84090371@gmail.com
Good Luck and HAPPY Coding!