

Homework 2

姓名: 毕秋宇 学号: 171860624

2019 年 3 月 31 日

第一部分 作业

✓ Problem 1 (Multi-Label Logistic Regression)

In multi-label problem, each instance x has a label set $y = \{y_1, y_2, \dots, y_l\}$ and each label $y_i \in \{0, 1\}$. Assume the post probability $p(y|x)$ follows the conditional independence:

$$p(y|x) = \prod_{i=1}^l p(y_i|x)$$

Please use the logistic regression method to handle the following questions.

1. [15pts] Please give the 'log-likelihood' function of your logistic regression model;
2. [10pts] Please calculate the gradient of your 'log-likelihood' function.

✍ Solution

Because the post probability $p(y|x)$ follows the conditional independence, so every label is independent. We can make $z_i = w_i^T x + b_i$, that is

$$\begin{pmatrix} z_1 \\ \vdots \\ z_m \end{pmatrix} = \begin{pmatrix} w_1^T \\ \vdots \\ w_m^T \end{pmatrix} \times \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} + \begin{pmatrix} b_1 \\ \vdots \\ b_m \end{pmatrix} = \begin{pmatrix} w_{11} & \cdots & w_{1n} \\ \vdots & \cdots & \vdots \\ w_{m1} & \cdots & w_{mn} \end{pmatrix} \times \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} + \begin{pmatrix} b_1 \\ \vdots \\ b_m \end{pmatrix}$$

(1) The likelihood function of the logistic regression model is:

$$L(x_1, x_2, \dots, x_n) = \prod_{j=1}^n p(y|x_j) = \prod_{j=1}^n \prod_{i=1}^l p(y_i|x_j)$$

So the log-likelihood function of it is:

$$\log L(x_1, x_2, \dots, x_n) = \log \prod_{j=1}^n p(y|x_j) = \log \prod_{j=1}^n \prod_{i=1}^l p(y_i|x_j) = \sum_{j=1}^n \sum_{i=1}^l \log p(y_i|x_j)$$

(2)

$$\nabla \log L(x_1, \dots, x_n) = \left[\frac{\partial \sum_{i=1}^l \log p(y_i|x_1)}{\partial x_1}, \dots, \frac{\partial \sum_{i=1}^l \log p(y_i|x_n)}{\partial x_n} \right]^T$$

✓ Problem 2 (Linear Discriminant Analysis)

Suppose we transform the original \mathbf{X} to $\hat{\mathbf{Y}}$ via linear regression. In detail, let

$$\hat{\mathbf{Y}} = \mathbf{X}(\mathbf{X}^T \mathbf{X}^{-1})^{-1} \mathbf{X}^T \mathbf{Y} = \mathbf{X} \hat{\mathbf{B}},$$

where \mathbf{X} and \mathbf{Y} are the feature and label matrix, respectively. Similarly for any input \mathbf{x} , we get a transformed vector $\hat{\mathbf{y}} = \hat{\mathbf{B}}^T \mathbf{x}$. Show that *LDA* using $\hat{\mathbf{Y}}$ is identical to *LDA* in the original space.

✍ Solution

Because we have that $\hat{\mathbf{y}} = \hat{\mathbf{B}}^T \mathbf{x}$, so $\hat{\mu}_k^{\hat{\mathbf{y}}} = \hat{B}^T \hat{\mu}_k^x$ and $\hat{\mu}_l^{\hat{\mathbf{y}}} = \hat{B}^T \hat{\mu}_l^x$.

That is to say,

$$\begin{aligned} \log \frac{\Pr(G = k | \hat{Y} = \hat{y})}{\Pr(G = l | \hat{Y} = \hat{y})} &= \log \frac{f_k(\hat{y})}{f_l(\hat{y})} + \log \frac{\pi_k}{\pi_l} = \log \frac{\pi_k}{\pi_l} - \frac{1}{2} (\hat{\mu}_k^{\hat{\mathbf{y}}} + \hat{\mu}_l^{\hat{\mathbf{y}}})^T \Sigma_y^{-1} (\hat{\mu}_k^{\hat{\mathbf{y}}} - \hat{\mu}_l^{\hat{\mathbf{y}}}) + \hat{y}^T \Sigma_y^{-1} (\hat{\mu}_k^{\hat{\mathbf{y}}} - \hat{\mu}_l^{\hat{\mathbf{y}}}) \\ &= \log \frac{\pi_k}{\pi_l} - \frac{1}{2} (\hat{\mu}_k^x + \hat{\mu}_l^x)^T \hat{B} \Sigma_y^{-1} \hat{B}^T (\hat{\mu}_k^x - \hat{\mu}_l^x) + \hat{x}^T \hat{B} \Sigma_y^{-1} \hat{B}^T (\hat{\mu}_k^x - \hat{\mu}_l^x) \end{aligned}$$

According to the equation in ESL 4.9.

So We need to prove that

$$\hat{B} \Sigma_y^{-1} \hat{B}^T = \Sigma_x^{-1}$$

We also know

$$\begin{aligned} \Sigma_y &= \frac{\sum_{k=1}^N \sum_{g_i=k} (\hat{y}_i - \hat{\mu}_k^{\hat{\mathbf{y}}}) (\hat{y}_i - \hat{\mu}_k^{\hat{\mathbf{y}}})^T}{N - K} = \frac{\sum_{k=1}^N \sum_{g_i=k} \hat{B}^T (\hat{x}_i - \hat{\mu}_k^x) (\hat{x}_i - \hat{\mu}_k^x)^T \hat{B}}{N - K} = \hat{B}^T \Sigma_x \hat{B}. \\ &\Rightarrow \hat{B} \Sigma_y^{-1} \hat{B}^T = \hat{B} (\hat{B}^T \Sigma_x \hat{B})^{-1} \hat{B}^T = \hat{B} \hat{B}^{-1} \Sigma_x^{-1} (\hat{B}^T)^{-1} \hat{B}^T = \Sigma_x^{-1} \end{aligned}$$

So far, we have proved that *LDA* using $\hat{\mathbf{Y}}$ is identical to *LDA* in the original space.

✓ Problem 3 (Logistic Regression from scratch)

Implementing algorithms is a good way of understanding how they work in-depth. In case that you are not familiar with the pipeline of building a machine learning model, this article can be an example.

In this experiment, you are asked to build a classification model on one of UCI data sets, Letter Recognition Data Set.

In particular, the objective is to identify each of a large number of black-and-white.

rectangular pixel displays as one of the 26 capital letters in the English alphabet. The detailed statistics of this data set is listed in Table. The data set was then randomly split into train set and test set with proportion 7 : 3. Also, letters from 'A' to 'Z' are mapped to digits '1' to '26' respectively as represented in the last column of the provided data set.

Property	Value	Description
Number of Instances	20,000	Rows of the data set
Number of Features	17	Columns of the data set
Number of classes	26	Dimension of the target attribute

In order to build machine learning models, you are supposed to implement Logistic Regression (LR) algorithm which is commonly used in classification tasks. Specifically, in this experiment, you have to adapt the traditional binary class LR method to tackle the multi-class learning problem.

- (1) [5pts] You are encouraged to implement the code using *Python3* or *Matlab*, implementations in any other programming language will not be judged. Please name the source file (which contains the main function) as *LR_main.py* (for python3) or *LR_main.m* (for matlab). Finally, your code needs to print the testing performance on the provided test set once executed.
- (2) [30pts] Functions required to implement:
- Implement LR algorithm using gradient descent or Newton's method.
 - Incorporate One-vs-Rest (OvR) strategy to tackle multi-class classification problem.
- (3) [20pts] Explain implementing details in your submitted report (source code should not be included in your report), including optimization details and hyper-parameter settings, etc. Also, testing performance with respect to Accuracy, Precision, Recall, and F_1 score should be reported following the form of Table 2.

Performance Metric	Value (%)
accuracy	00.00
micro Precision	00.00
micro Recall	00.00
micro F_1	00.00
macro Precision	00.00
macro Recall	00.00
macro F_1	00.00

NOTE: Any off-the-shelf implementations of LR or optimization methods

are **NOT ALLOWED** to use. When submitting your code and report, all files should be placed in the same directory (without any sub-directory).

Solution


Introduction : While implementing Logistic Regression, I use matrix multiplication instead of training every case to save the running time and guarantee the large number of epoch. In terms of choosing the hyper-parameter α , which is normally called gradient in Maths, method of bisection is useful. I checked the gradient from 1 to 1^{-7} , and used the function $\alpha = \frac{0.01 + \frac{0.01}{1+i}}{m}$ of epoch i instead of just choosing a constant in order for a faster convergence. Finally, I implement the Logistic Regression with 100000 epoch and the performance is as follows

```
micro Precision=0.709667
micro Recall=0.709667
micro F1=0.709667
macro Precision=0.710653
macro Recall=0.710732
macro F1=0.707638
accuracy = 0.709667
```

Figure 1: Performance

Difficulties that I met : When writing the vote function, I create an array `[]` and assign it with prediction by parameters $1 \sim 26$, then use function `index(max(array)) + 1` to get the index of largest probability. However, the

assignment will replace the original value, so $index(max(array)) + 1$ will always return 1. After I using a loop to get the maximum of probability and the index of it, the problem is fixed.

第二部分 订正 

第三部分 反馈 