Homework 2 by 161220097 戚赟

2019 年 4 月 3 日

1 [25pts] Multi-Label Logistic Regression

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

$$p(\boldsymbol{y}|x) = \prod_{i=1}^{l} p(y_i|x). \tag{1.1}$$

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:
(1)对数似然函数为：

$$\ell(\boldsymbol{\omega}, b) = \sum_{i=1}^{m} \ln p(\boldsymbol{y_i}|\boldsymbol{x_i}; \boldsymbol{\omega}, b) \tag{1.2}$$

$$= \sum_{i=1}^{m} \ln \prod_{j=1}^{l} p(y_{ij}|\boldsymbol{x_i}; \boldsymbol{\omega_j}, b_j) \tag{1.3}$$

$$= \sum_{i=1}^{m} \sum_{j=1}^{l} \ln p(y_{ij}|\boldsymbol{x_i}; \boldsymbol{\omega_j}, b_j) \tag{1.4}$$

令$\boldsymbol{\beta}=(\boldsymbol{\omega};b),\ \hat{\boldsymbol{x}}=(\boldsymbol{x};1)$,则对数似然函数为:

$$\ell(\boldsymbol{\beta})=\sum_{i=1}^{m}\sum_{j=1}^{l}\ln(y_{ij}p_1(\hat{\boldsymbol{x}}_{\boldsymbol{i}};\boldsymbol{\beta}_j)+(1-y_{ij})p_0(\hat{\boldsymbol{x}}_{\boldsymbol{i}};\boldsymbol{\beta}_j)) \tag{1.5}$$

$$=\sum_{i=1}^{m}\sum_{j=1}^{l}(y_{ij}\boldsymbol{\beta}_{\boldsymbol{j}}^{\boldsymbol{T}}\hat{\boldsymbol{x}}_{\boldsymbol{i}}-\ln(1+e^{\boldsymbol{\beta}_j^T\hat{\boldsymbol{x}}_{\boldsymbol{i}}})) \tag{1.6}$$

(2)梯度的计算如下：

$$\frac{\partial\ell(\boldsymbol{\beta})}{\partial\boldsymbol{\beta}_t}=\sum_{i=1}^{m}[y_{it}\hat{\boldsymbol{x}}_i-\frac{\hat{\boldsymbol{x}}_i e^{\boldsymbol{\beta}_t^T\hat{\boldsymbol{x}}_i}}{1+e^{\boldsymbol{\beta}_t^T\hat{\boldsymbol{x}}_i}}] \tag{1.7}$$

$$=\sum_{i=1}^{m}\hat{\boldsymbol{x}}_i[y_{it}-\frac{e^{\boldsymbol{\beta}_t^T\hat{\boldsymbol{x}}_i}}{1+e^{\boldsymbol{\beta}_t^T\hat{\boldsymbol{x}}_i}}] \tag{1.8}$$

2 [20pts] Linear Discriminant Analysis

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

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

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

Solution:
假设y的种类的数量为K，则因为$\hat{y}=\hat{\boldsymbol{B}}^T x$，则对于种类$k$

$$\hat{\mu}_k^{\hat{y}}=\frac{\sum_{g_i=k}\hat{y}_i}{N_k}=\frac{\sum_{g_i=k}\hat{\boldsymbol{B}}^T x_i}{N_k}=\hat{\boldsymbol{B}}^T\hat{\mu}_k^x \tag{2.1}$$

同理，对于种类l来说，$\hat{\mu}_l^{\hat{y}} = \hat{\boldsymbol{B}}^T\hat{\mu}_l^x$. 所以为了要证明是一致的，必须要证明对于任意两个种类$k, l, \frac{\log Pr(G=k|\hat{Y}=\hat{y})}{\log Pr(G=l|\hat{Y}=\hat{y})} = \frac{\log Pr(G=k|\hat{X}=x)}{\log Pr(G=l|\hat{X}=x)}$. 而

$$\frac{\log Pr(G=k|\hat{Y}=\hat{y})}{\log Pr(G=l|\hat{Y}=\hat{y})} = \frac{\log \pi_k}{\log \pi_l} - \frac{1}{2}(\hat{\mu}_k^x+\hat{\mu}_l^x)\hat{\boldsymbol{B}}(\hat{\boldsymbol{B}}^T\hat{\sum_x}\hat{\boldsymbol{B}})^{-1}\hat{\boldsymbol{B}}^T(\hat{\mu}_k^x-\hat{\mu}_l^x) + x^T\hat{\boldsymbol{B}}(\hat{\boldsymbol{B}}^T\hat{\sum_x}\hat{\boldsymbol{B}})^{-1}\hat{\boldsymbol{B}}^T(\hat{\mu}_k^x-\hat{\mu}_l^x) \tag{2.2}$$

所以只需要证明$\hat{\boldsymbol{B}}(\hat{\boldsymbol{B}}^T\hat{\sum}_x\hat{\boldsymbol{B}})^{-1}\hat{\boldsymbol{B}}^T(\hat{\mu}_k^x - \hat{\mu}_l^x) = \hat{\sum}_x^{-1}(\hat{\mu}_k^x - \hat{\mu}_l^x)$,
也就是$\hat{\sum}_x\hat{\boldsymbol{B}}(\hat{\boldsymbol{B}}^T\hat{\sum}_x\hat{\boldsymbol{B}})^{-1}\hat{\boldsymbol{B}}^T(\hat{\mu}_k^x - \hat{\mu}_l^x) = (\hat{\mu}_k^x - \hat{\mu}_l^x)$,
所以只要证明：$\hat{\sum}_x\hat{\boldsymbol{B}}(\hat{\boldsymbol{B}}^T\hat{\sum}_x\hat{\boldsymbol{B}})^{-1}\hat{\boldsymbol{B}}^T\hat{\mu}_k^x = \hat{\mu}_k^x$
设$H = \hat{\sum}_x\hat{\boldsymbol{B}}(\hat{\boldsymbol{B}}^T\hat{\sum}_x\hat{\boldsymbol{B}})^{-1}\hat{\boldsymbol{B}}^T$，我们能发现H是一个投影矩阵，因为$HH = H$.所以只需要证明$H\hat{\mu}_k^x = \hat{\mu}_k^x$. 而$N_k\hat{\boldsymbol{\mu}}_{\boldsymbol{k}}^{\boldsymbol{x}} = \sum_{k_i=k} x_i = X^T y_k$,所以只需要证明$HX^TY = X^TY$即可。

$$\hat{\boldsymbol{B}}^T = Y^TX(X^TX)^{-1} \tag{2.3}$$

$$\hat{\sum_x} = \frac{1}{N-K}\sum_{k=1}^{K}\sum_{g_i=1}[(x_i - \hat{\mu}_k)(x_i - \hat{\mu}_k)^T] \tag{2.4}$$

$$= \frac{1}{N-K}\sum_{k=1}^{K}\sum_{g_i=1}[(x_ix_i^T - 2x_i\hat{\mu}_k^T + \hat{\mu}_k\hat{\mu}_k^T)] \tag{2.5}$$

$$= \frac{1}{N-K}[\sum_{i=1}^{N}x_ix_i^T - \sum_{k=1}^{K}N_k\hat{\mu}_k\hat{\mu}_k^T] \tag{2.6}$$

$$= \frac{1}{N-K}[\sum_{i=1}^{N}x_ix_i^T - \sum_{k=1}^{K}X^Ty_iD^{-1}y_i^TX] \tag{2.7}$$

$$= \frac{1}{N-K}[X^TX - X^TYD^{-1}Y^TX] \tag{2.8}$$

其中D矩阵是

$$\begin{bmatrix} N_1 & 0 & \dots & 0 \\ 0 & N_2 & \dots & 0 \\ 0 & \dots & \dots & 0 \\ 0 & \dots & N_{K-1} & 0 \\ 0 & 0 & \dots & N_K \end{bmatrix} \tag{2.9}$$

$$\hat{\sum_x}\hat{\boldsymbol{B}} = \frac{1}{N-K}X^TY(I - D^{-1}Y^TX(X^TX)^{-1}X^TY) \tag{2.10}$$

$$(\hat{\boldsymbol{B}}^T\hat{\sum_x}\hat{\boldsymbol{B}})^{-1} = (N-K)(Y^TX(X^TX)^{-1}X^TY(I - D^{-1}Y^TX(X^TX)^{-1}X^TY))^{-1} \tag{2.11}$$

$$set\ P = Y^TX(X^TX)^{-1}X^TY, so \tag{2.12}$$

$$(\hat{\boldsymbol{B}}^T\hat{\sum_x}\hat{\boldsymbol{B}})^{-1} = (N-K)(P(1 - D^{-1}P))^{-1} \tag{2.13}$$

$$= (N-K)(1 - D^{-1}P)^{-1}P^{-1}\ (P\ is\ invertable) \tag{2.14}$$

所以

$$HX^TY = \frac{1}{N-K}X^TY(I - D^{-1}P)(N-K)(1 - D^{-1}P)^{-1}P^{-1}Y^TX(X^TX)^{-1}X^TY \tag{2.15}$$

$$= X^TYP^{-1}P = X^TY \tag{2.16}$$

于是LDA用$\hat{y}$和在原空间上分析是一致的。

3 [55pts] 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 (here).

In this experiment, you are asked to build a classification model on one of UCI data sets, Letter Recognition Data Set (click to download). 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 1. 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.

表 1: Statistics of the 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.

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).

表 2: Performance of your implementation on test set.

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

Report of my lab:
(1)我严格遵守了实验要求，使用了**Python3**作为编程语言，实现的编程文件为**LR_mian.py**，能够正确读取数据，进行运算和预测，结果能够正确的显示.

(2)在**LR_main.py**中，我实现了**梯度下降法**和**牛顿法**，并且使用了OvR的思想，即对于每一个类别，分别将其看作正类，其余看作反类，进行预测和运算，得到一个结果，利用得到的26个向量进行预测，本应该是取预测最准的作为结果，但是实验发现这样做的精度很低，于是改用26个向量分别进行预测，取概率最大的作为结果预测，综合以上我完整的完成了要求.

(3)我的实验设计如下，包含了Solution类，包含如下的函数：
readTrain_set(self)：用来读取训练集数据并做处理
readTest_set(self)：用来读取测试集并做处理
readRes_set(self)：用来读取结果文件，type = 0读取的是梯度下降法的结果，type = 1读取的是牛顿法的结果
normalize_set(self,tempDataMat)：将测试的数据做归一化处理
OvR(self,type):利用OvR进行测试，根据type选择梯度下降法还是牛顿法进行处理
gradAscent(self, dataMatrix, labelsMatrix ,inX): 进行梯度下降处理，其中inX代表将第inX类看成正例
newton_method(self,dataMatrix, labelsMatrix,inX): 进行牛顿法处

理，其中inX代表将第inX类看成正例

predictTestData(self): 表示预测测试集

saveRes(self,type): 表示对于结果的保存,type = 0 代表梯度下降结果保存为res_gra.txt, type = 1 表示牛顿法结果保存为res_newton.txt,结果保留四位小数.

数据归一化:对于数据归一化，就是将数据矩阵进行处理，使得值范围处在[0,1]之间，这样效果显著，在归一化之前，梯度和牛顿的精确率达到40%，但是归一化之后，准确率稳定在70%左右，下面的显示的是已经归一化的结果.

梯度下降法的实现细节：目标函数：$J(\theta) = \sum_{i=1}^{m}[y^i \log h_\theta(x^i) + (1-y^i)] \log h_{1-\theta}(x^i)$

伪代码为：

1 每个回归系数初始化为0

2 重复R次:

3 计算整个数据集的梯度

4 使用 alpha * gradient 更新回归系数向量,其中alpha为学习率取很小

5 返回回归系数

对于参数的设置，当取alpha = 0.01 , 次数为5000轮时:

表 3: 梯度下降1

Performance Metric	Value (%)
accuracy	45.71
micro Precision	45.26
micro Recall	44.47
micro F_1	44.84
macro Precision	41.15
macro Recall	44.83
macro F_1	42.91

可见得并没有比较好得收敛性质，调整参数，和步长有关，当步长过小，

很容易导致不能到达最低点，但是如果步长过大，不能够达到最小顶点，所以需要不断尝试，到当取alpha = 0.0005，次数为 5000，结果为:

表 4: 梯度下降2

Performance Metric	Value (%)
accuracy	71.41
micro Precision	76.65
micro Recall	43.93
micro F_1	55.85
macro Precision	68.38
macro Recall	43.88
macro F_1	53.46

当调整之后，概率稳定在71%左右，达到极限，可以推测出极限大概是71%，但是由于差全率比较低，所以存在一定得过拟合现象.而且梯度下降得时间比较长，不优化得算法达到跑一次8分钟左右

牛顿下降法的实现细节：参考书上给出得原理，则牛顿法得关键在于求取一阶和二阶导数，牛顿是下降沿着切线方向进行下降得，则仅与迭代得次数是有关系得：

取迭代得次数为5时：

表 5: 牛顿法1

Performance Metric	Value (%)
accuracy	70.73
micro Precision	77.81
micro Recall	42.9
micro F_1	55.30
macro Precision	69.13
macro Recall	42.83
macro F_1	52.89

取迭代得次数为10时：

表 6: 牛顿法2

Performance Metric	Value (%)
accuracy	71.15
micro Precision	76.36
micro Recall	44.59
micro F_1	53.98
macro Precision	68.46
macro Recall	44.56
macro F_1	53.98

最后达到得精度差不多和梯度下降一致，可以看出不采用其他数据清洗策略，极限大概是71%左右.但牛顿法时间更快.

困惑: 在对于各项性能指标进行计算得时候，参考书上对于说明，是对多个混淆矩阵进行操作，但是并没有很清楚是用哪些混淆矩阵，利用库进行计算，但结果不是很令人信服，于是我采用了如下得方法：对于26个预测向量，分别计算它预测得混淆矩阵，对于这26个样本总和都为6000的混淆矩阵进行操作.于是得出以上的数据，对于这个计算方式有疑问.

参考：

[1]逻辑回归https://blog.csdn.net/m0_37393514/article/details/79005859
[2]梯度下降法https://www.jianshu.com/p/c7e642877b0e