Logistic Regression

袁雨 PB20151804

一、实验内容

In `Logistic.py`, write your own Logistic Regression class

In `Load.ipynb`

1. Deal with NULL rows, you can either choose to drop them or replace them with mean or other value
2. Encode categorical features
3. Split the dataset into X_train, X_test, y_train, y_test, also you can then use normalization or any other methods you want
4. Train your model and plot the loss curve of training
5. Compare the accuracy(or other metrics you want) of test data with different parameters you train with, i.e. learning rate, regularization methods and parameters .etc

二、实验要求

- Do not use sklearn or other machine learning library, you are only permitted with numpy, pandas, matplotlib, and Standard Library, you are required to write this project from scratch.
- You are allowed to discuss with other students, but you are not allowed to plagiarize the code, we will use automatic system to determine the similarity of your programs, once detected, both of you will get zero mark for this project.
- Report
    - The Loss curve of one training process
    - The comparation table of different parameters
    - The best accuracy of test data

三、实验步骤

1. **数据集分析**

```
df.head()
```

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
0	LP001002	Male	No	0	Graduate	No	5849	0.0	NaN	360.0	1.0	Urban	Y
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0	Rural	N
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	Urban	Y
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	Urban	Y
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	1.0	Urban	Y

查看数据集的前五行。

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 614 entries, 0 to 613
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Loan_ID            614 non-null    object 
 1   Gender             601 non-null    object 
 2   Married            611 non-null    object 
 3   Dependents         599 non-null    object 
 4   Education          614 non-null    object 
 5   Self_Employed      582 non-null    object 
 6   ApplicantIncome    614 non-null    int64  
 7   CoapplicantIncome  614 non-null    float64
 8   LoanAmount         592 non-null    float64
 9   Loan_Amount_Term   600 non-null    float64
 10  Credit_History     564 non-null    float64
 11  Property_Area      614 non-null    object 
 12  Loan_Status        614 non-null    object 
dtypes: float64(4), int64(1), object(8)
memory usage: 62.5+ KB
```

查看数据集基本信息。该数据集一共13列，包括12个特征和1个目标。一共614行。特征有int、float、object等类型，包含缺失值。

```
df['Loan_Status'].value_counts()
```

	Loan_Status
Y	422
N	192

观察目标列，包含正例422个，负例192个。

2. 数据预处理

（1）特征选择

去掉在预测目标上没有意义的特征"Loan_ID"列，再统计缺失值。

```
df.drop("Loan_ID", axis=1, inplace=True)
# Checking the Missing Values
df.isnull().sum()
```

	data
Gender	13
Married	3
Dependents	15
Education	0
Self_Employed	32
ApplicantIncome	0
CoapplicantIncome	0
LoanAmount	22
Loan_Amount_Term	14
Credit_History	50
Property_Area	0
Loan_Status	0

（2）特征编码

将object型的分类特征转换成数字型，方便机器学习模型计算。

```
GenderMap = {'Male':1,'Female':0}
MarriedMap = {'Yes':1,'No':0}
DependentsMap={'3+':3,'2':2,'1':1,'0':0}
EducationMap = {'Graduate':1,'Not Graduate':0}
Self_EmployedMap = {'Yes':1,'No':0}
Property_AreaMap = {'Urban':2,'Semiurban':1,'Rural':0}
Loan_StatusMap = {'Y':1,'N':0}
df['Gender'] = df['Gender'].map(GenderMap)
……
```

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
0	1.0	0.0	0.0	1	0.0	5849	0.0	NaN	360.0	1.0	2	1
1	1.0	1.0	1.0	1	0.0	4583	1508.0	128.0	360.0	1.0	0	0
2	1.0	1.0	0.0	1	1.0	3000	0.0	66.0	360.0	1.0	2	1
3	1.0	1.0	0.0	0	0.0	2583	2358.0	120.0	360.0	1.0	2	1
4	1.0	0.0	0.0	1	0.0	6000	0.0	141.0	360.0	1.0	2	1
...	...	...	...	...	...	...	...	...	...	...	...	...
609	0.0	0.0	0.0	1	0.0	2900	0.0	71.0	360.0	1.0	0	1
610	1.0	1.0	3.0	1	0.0	4106	0.0	40.0	180.0	1.0	0	1
611	1.0	1.0	1.0	1	0.0	8072	240.0	253.0	360.0	1.0	2	1
612	1.0	1.0	2.0	1	0.0	7583	0.0	187.0	360.0	1.0	2	1
613	0.0	0.0	0.0	1	1.0	4583	0.0	133.0	360.0	0.0	1	0

（3）缺失值处理

可以丢弃缺失值、用均值填充等。实验证明若丢弃缺失值则只剩下480个样本，训练效果较差，故选择均值填充。

```
# Task1 deal with NULL rows, you can either choose to drop them or replace
them with mean or other value
for column in list(df.columns[df.isnull().sum() > 0]):
    df[column].fillna(df[column].mean(), inplace=True)
```

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
0	1.0	0.0	0.0	1	0.0	5849	0.0	146.412162	360.0	1.0	2	1
1	1.0	1.0	1.0	1	0.0	4583	1508.0	128.000000	360.0	1.0	0	0
2	1.0	1.0	0.0	1	1.0	3000	0.0	66.000000	360.0	1.0	2	1
3	1.0	1.0	0.0	0	0.0	2583	2358.0	120.000000	360.0	1.0	2	1
4	1.0	0.0	0.0	1	0.0	6000	0.0	141.000000	360.0	1.0	2	1
...	...	...	...	...	...	...	...	...	...	...	...	...
609	0.0	0.0	0.0	1	0.0	2900	0.0	71.000000	360.0	1.0	0	1
610	1.0	1.0	3.0	1	0.0	4106	0.0	40.000000	180.0	1.0	0	1
611	1.0	1.0	1.0	1	0.0	8072	240.0	253.000000	360.0	1.0	2	1
612	1.0	1.0	2.0	1	0.0	7583	0.0	187.000000	360.0	1.0	2	1
613	0.0	0.0	0.0	1	1.0	4583	0.0	133.000000	360.0	0.0	1	0

614 rows × 12 columns

数据清理后查看数据集的若干统计值。

```
df.describe()
```

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
count	614.000000	614.000000	614.000000	614.000000	614.000000	614.000000	614.000000	614.000000	614.000000	614.000000	614.000000	614.000000
mean	0.813644	0.651391	0.762938	0.781759	0.140893	5403.459283	1621.245798	146.412162	342.000000	0.842199	1.037459	0.687296
std	0.385564	0.475752	1.002718	0.413389	0.339000	6109.041673	2926.248369	84.037468	64.372489	0.349681	0.787482	0.463973
min	0.000000	0.000000	0.000000	0.000000	0.000000	150.000000	0.000000	9.000000	12.000000	0.000000	0.000000	0.000000
25%	1.000000	0.000000	0.000000	1.000000	0.000000	2877.500000	0.000000	100.250000	360.000000	1.000000	0.000000	0.000000
50%	1.000000	1.000000	0.000000	1.000000	0.000000	3812.500000	1188.500000	129.000000	360.000000	1.000000	1.000000	1.000000
75%	1.000000	1.000000	1.000000	1.000000	0.000000	5795.000000	2297.250000	164.750000	360.000000	1.000000	2.000000	1.000000
max	1.000000	1.000000	3.000000	1.000000	1.000000	81000.000000	41667.000000	700.000000	480.000000	1.000000	2.000000	1.000000

查看正例和负例不同特征的均值差异。

```
df.groupby('Loan_Status').mean()
```

Loan_Status	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area
0	0.802439	0.588542	0.753008	0.729167	0.142021	5446.078125	1877.807292	150.945488	344.000000	0.562232	1.000000
1	0.818742	0.679986	0.767456	0.805687	0.140380	5384.068720	1504.516398	144.349606	341.090047	0.969577	1.054502

（4）标准化

可以使用z-score、min_max等标准化方法。因为Logistic Regression的参数优化采用了梯度下降法，如果不对特征进行归一化，可能会使得损失函数值得等高线呈椭球形，这样花费更多的迭代步数才能到达最优解；损失函数加入了正则项，参数的大小决定了损失函数值等。经实验验证，min_max归一化方法效果更好。故对特征进行归一化。

```
X=df.iloc[:,:-1]
X_MinMax = (X-X.min())/(X.max()-X.min())
print(X_MinMax)
```

```
     Gender  Married  Dependents  Education  Self_Employed  ApplicantIncome  \
0       1.0      0.0    0.000000        1.0            0.0         0.070489
1       1.0      1.0    0.333333        1.0            0.0         0.054830
2       1.0      1.0    0.000000        1.0            1.0         0.035250
3       1.0      1.0    0.000000        0.0            0.0         0.030093
4       1.0      0.0    0.000000        1.0            0.0         0.072356
..      ...      ...         ...        ...            ...              ...
609     0.0      0.0    0.000000        1.0            0.0         0.034014
610     1.0      1.0    1.000000        1.0            0.0         0.048930
611     1.0      1.0    0.333333        1.0            0.0         0.097984
612     1.0      1.0    0.666667        1.0            0.0         0.091936
613     0.0      0.0    0.000000        1.0            1.0         0.054830

     CoapplicantIncome  LoanAmount  Loan_Amount_Term  Credit_History  \
0             0.000000    0.198860          0.743590             1.0
1             0.036192    0.172214          0.743590             1.0
2             0.000000    0.082489          0.743590             1.0
3             0.056592    0.160637          0.743590             1.0
4             0.000000    0.191027          0.743590             1.0
..                 ...         ...               ...             ...
609           0.000000    0.089725          0.743590             1.0
610           0.000000    0.044863          0.358974             1.0
611           0.005760    0.353111          0.743590             1.0
612           0.000000    0.257598          0.743590             1.0
613           0.000000    0.179450          0.743590             0.0

     Property_Area
0              1.0
1              0.0
2              1.0
3              1.0
4              1.0
..             ...
609            0.0
610            0.0
611            1.0
612            1.0
613            0.5

[614 rows x 11 columns]
```

3. 实现Logistic Regression

(1) 模型

$\hat{y} = \frac{1}{1+e^{-z}}$，$z = wX + b$

$\hat{y}$：预测值。使用sigmoid 函数，表示$P(y=1|X)$，y=1的概率；

X：输入特征；

w：weight，权重；

b：bias，偏差。

(2) 优化目标

使用交叉熵损失函数，并加入正则化。

L1正则化：$J(w,b) = -\frac{1}{m}\sum_{i=1}^{m}[y_i \log(\hat{y}_i) + (1-y_i)log(1-\hat{y}_i)] + \frac{1}{m}\gamma||w||_1$

L2正则化：$J(w,b) = -\frac{1}{m}\sum_{i=1}^{m}[y_i \log(\hat{y}_i) + (1-y_i)log(1-\hat{y}_i)] + \frac{1}{m}\gamma w^T w$

m：样本数。

γ：正则化参数。

(3) 优化方法

使用梯度下降法更新参数w和b。

$w = w - lr\frac{\partial L}{\partial w}$

$$b = b - lr\frac{\partial L}{\partial b}$$

$$\frac{\partial J}{\partial w} = -\frac{1}{m}X^T(\hat{y} - y) + \begin{cases} \frac{1}{m}\gamma\sum_{i=1}^{m} sign(w_i) & \text{if L1} \\ \frac{1}{m}\gamma w & \text{if L2} \end{cases}$$

$$\frac{\partial J}{\partial b} = -\frac{1}{m}(\hat{y} - y)$$

lr：learning rate，学习率。

（4）算法实现

见`Logistic.py`。

3. 训练

使用`Logistic.py`里实验的LogisticRegression进行训练，返回训练得到的分类器、迭代次数与cost列表。

使用迭代次数与cost列表作出loss curve。

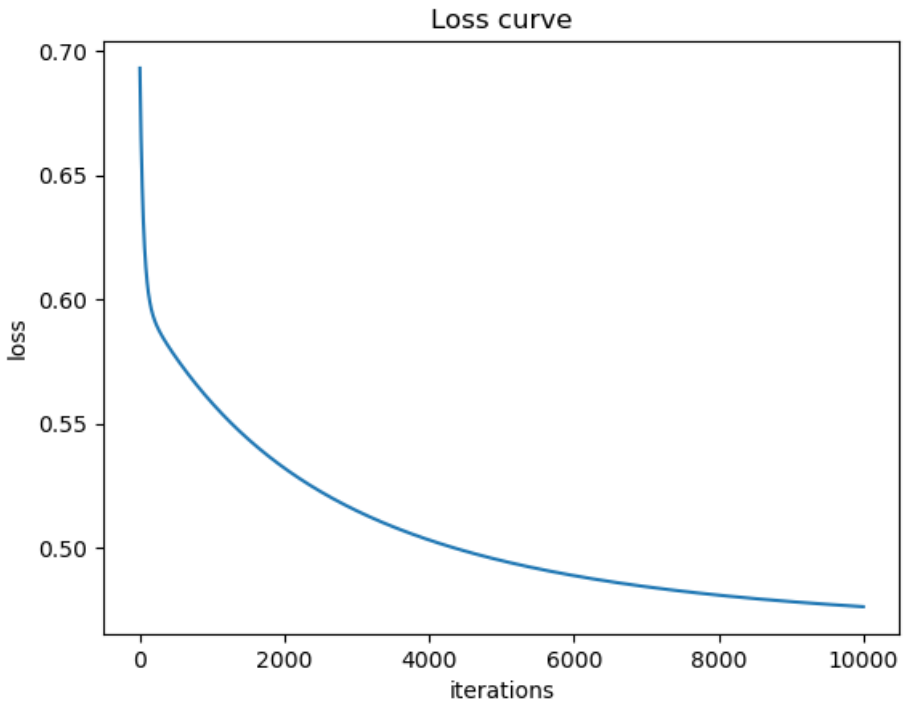
4. 测试

使用训练得到的分类器进行测试，采用十折交叉验证，最终得到十轮结果的accuracy均值。分别保存训练集与测试集的accuracy，以比较判断是否过拟合等。

四、实验结果

1. The Loss curve of one training process

(penalty="l2"，gamma=6，fit_intercept=True，lr=0.01，tol=1e-7，max_iter=1e4)



2. The comparation table of different parameters

初始：penalty="l2"，gamma=10，fit_intercept=True，lr=0.01，tol=1e-7，max_iter=1e4

（1）预处理方法

① 标准化方法

标准化方法	training accuracy	testing accuracy
归一化（min-max）	0.8081967213114754	0.8095840867992766
正规化（z-score）	0.7901639344262296	0.7867992766726943

可见归一化得到的accuracy更高，故选择归一化。

② 缺失值处理

缺失值处理	training accuracy	testing accuracy
丢弃	0.8083333333333333	0.8083333333333332
均值填充	0.8081967213114754	0.8095840867992766

丢弃之后只剩下480个样本。均值填充的样本数更多，得到的accuracy更高，故选择均值填充。

（2）调参

① penalty

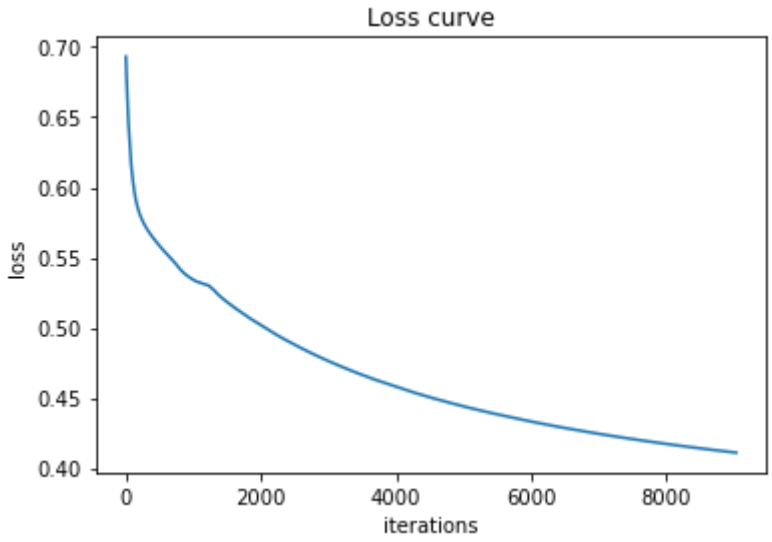
l1：L1正则化；
l2：L2正则化。

penalty	training accuracy	testing accuracy
l1	0.8081967213114754	0.8095840867992766
l2	0.8081967213114754	0.8095840867992766

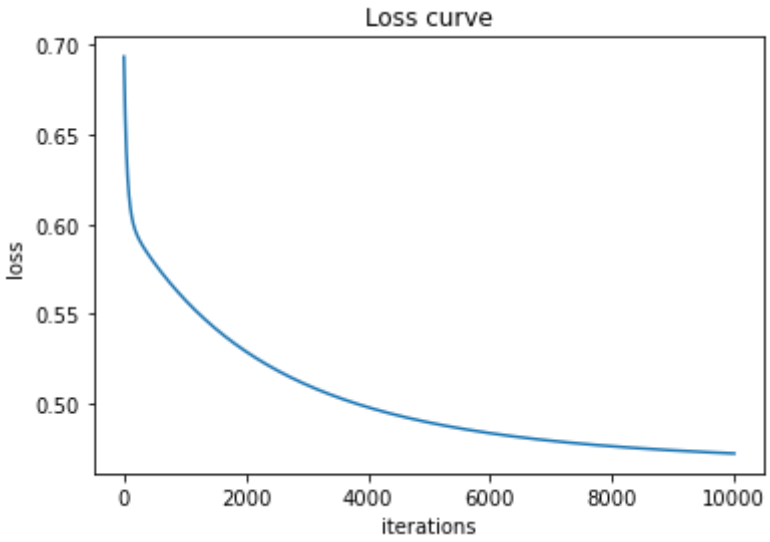
可见二者最终的accuracy相同。

观察二者的loss curve：

l1：



l2：



L2正则化的loss curve下降略快，且更平滑，故选择l2。

② gamma

正则化参数。

gamma	training accuracy	testing accuracy
0	0.8065573770491803	0.8092224231464737
3	0.8081967213114754	0.8095840867992766
6	0.8081967213114754	0.8095840867992766
10	0.8081967213114754	0.8095840867992766
20	0.8081967213114754	0.8095840867992766
30	0.7606557377049181	0.7712477396021701

正则化参数取3~20左右时accuracy达最高，选择 gamma = 6。

③ fit_intercept

True：包含bias；

False：不包含bias。

fit_intercept	training accuracy	testing accuracy
True	0.8081967213114754	0.8095840867992766
False	0.8081967213114754	0.8095840867992766

当max_iter = 10000, tol = 1e-7时，二者的accuracy相同。

接着比较当max_iter = 3000时：

fit_intercept	training accuracy	testing accuracy
True	0.8081967213114754	0.8090415913200723
False	0.8032786885245902	0.8057866184448462

可见fit_intercept = True时的accuracy略高，取fit_intercept = True。

④ lr

学习率。

lr	training accuracy	testing accuracy
0.001	0.6868852459016394	0.6873417721518987
0.01	0.8081967213114754	0.8095840867992766
0.1	0.8081967213114754	0.8095840867992766

可见学习率过低会导致收敛速度变慢，取lr=0.01。

⑤ tol

停止迭代的tolerance。

tol	training accuracy	testing accuracy
10^{-5}	0.8081967213114754	0.8079566003616637
10^{-7}	0.8081967213114754	0.8095840867992766
10^{-9}	0.8081967213114754	0.8095840867992766

tol < 10^{-7}时，accuracy几乎不变化，故没必要继续减小。取 tol = 1e-7。

⑥ max_iter

最大迭代次数。

max_iter	training accuracy	testing accuracy
1000	0.6916666666666667	0.6916666666666667
3000	0.8081967213114754	0.8090415913200723
5000	0.8081967213114754	0.8095840867992766
10000	0.8081967213114754	0.8095840867992766
20000	0.8081967213114754	0.8095840867992766

max_iter = 10000后，accuracy几乎不变化，故没必要继续增大。取max_iter = 10000。

总得来说该模型收敛较快，调参起的作用不大。

3. The best accuracy of test data

0.8095840867992766

（penalty="l2"，gamma=6，fit_intercept=True，lr=0.01，tol=1e-7，max_iter=1e4）