ML Lab 1

Logistic Regression

王世炟

PB20151796

October 13, 2022

Part 1: 实验要求

本次实验要求我们完成**逻辑回归**的模型实现，并根据Loan Data Set | Kaggle的数据进行训练和测试。具体需要完成以下部分：

- 数据预处理（包括数据清洗、数据填充、数据编码、训练集与测试集的划分）
- 逻辑斯蒂回归模型的初始化
- 实现优化算法
- 在训练集上训练模型，进行参数优化，精度达到某一程度停止，并得到损失函数曲线
- 在测试集上测试模型，获得准确率
- 进行 k 折交叉验证

Part 2: 数据集介绍

本次实验选取的数据集来自Loan Data Set | Kaggle，它描述了 614 位贷款人的有关信息，有关特征如下：

```
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)
```

图 1: Features of the data set

其中 Loan_Status 是本次实验要进行预测的类别。

Part 3: 数据预处理

数据预处理分为几个部分：数据填充、数据编码、归一化、数据集划分

3.1 数据填充

以下是数据集的缺失情况：

表 1: 数据集缺失情况

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

常见的数据缺失处理方法有：删除该条数据，使用均值、众数、中位数填充，使用周围数据的组合进行填充本次实验采用以下方法：

- 对于特征 Gender，采取把缺失数据直接删除的方法，理由是 Gender 的缺失数量较少，且用其他方法均不是十分合理
- 对于特征 LoanAmount，采取填充均值的方法，理由是可以观察到该特征取值连续，用平均值或中位数填充比较合理。
- 对于特征 Married、Dependents、Self_Employed、Loan_Amount_Term、Credit_History，采取填充众数的方法，理由是这些特征大多为离散值，用平均值不可取，众数可代表一般情况，较为合理。

3.2 数据编码

对于字符串型的数据，我们要进行编码处理，将其转化为可运算的数字，再进行模型的训练。对应编码如下：

```python
df1.Gender=df1.Gender.map({'Male':1,'Female':0})
df1.Married=df1.Married.map({'Yes':1,'No':0})
df1.Education=df1.Education.map({'Graduate':1,'Not Graduate':0})
df1.Self_Employed=df1.Self_Employed.map({'Yes':1,'No':0})
df1.Loan_Status=df1.Loan_Status.map({'Y':1,'N':0})
df1.Property_Area=df1.Property_Area.map({'Urban':1,'Semiurban':2,'Rural':3})
df1.Dependents=df1.Dependents.map({'0':0,'1':1,'2':2,'3+':3})
```

3.3 数据归一化

本实验采用 MIN-MAX 方法进行数据归一化
MIN-MAX 方法是对原始数据进行线性变换，将值映射到 $[0, 1]$ 之间

Algorithm 1 Algorithm of MIN-MAX

Input: X(X is the data set)

Output: X'

for each $i \in [1, X.length]$ do

  $X'[i] = \frac{X[i]-min(X[i])}{max(X[i])-min(X[i])}$

end for

return X'

3.4 数据集划分

本实验采用 5 次 5 折交叉验证进行模型检验，即训练集与测试集比例为 $4:1$
具体实现方法为首先进行数据洗牌，每次训练时生成 $[0, X.shape[0]-1]$ 内的整数乱序序列，然后按比例进行数据切片，划分训练集与测试集。这样做的好处是随机划分，可以提高模型的泛化能力。

Part 4: 理论基础

4.1 多元回归方程形式

多元回归方程形式:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p$$

写成矩阵形式为:

$$\boldsymbol{Y} = \boldsymbol{X\beta}$$

4.2 Logistics Regression **模型**

Logistics Regression 模型中, 利用了 sigmoid 函数来估计概率

$$P(\boldsymbol{Y}=1) = \frac{1}{1+e^{\boldsymbol{X\beta}}}$$

为了估计出参数 $\boldsymbol{\beta}$, 课本采用了 **最大似然估计**. 以二分类问题为例, 我们有:

$$P(y|x,\beta) = P(y=1|x,\beta)^y[1-P(y=1|x,\beta)]^{1-y}$$

由此可以写出似然函数:

$$\mathcal{L}(\beta) = \prod_{i=1}^{n} P(y_i|x_i,\beta) = \prod_{i=1}^{n} \left(\frac{1}{1+e^{-x_i\beta}}\right)^{y_i} \left(1-\frac{1}{1+e^{-x_i\beta}}\right)^{1-y_i}$$

对其取对数即得到对数似然函数:

$$\log\mathcal{L}(\beta) = \sum_{i=1}^{n} \left[y_i \log\left(\frac{1}{1+e^{-x_i\beta}}\right) + (1-y_i)\log\left(1-\frac{1}{1+e^{-x_i\beta}}\right)\right]$$

我们可以将它的相反数并取平均值当做损失函数:

$$J(\beta) = -\frac{1}{n}\log\mathcal{L}(\beta) = -\frac{1}{n}\sum_{i=1}^{n}\left[y_i\log\left(P(y_i)\right) + (1-y_i)\log\left(1-P(y_i)\right)\right]$$

4.3 **优化方法**

为了用梯度法优化参数, 应当将损失函数对参数 β 求导:

$$\frac{\partial J(\beta)}{\partial \beta_j} = -\frac{1}{n}\sum_{i=1}^{n}(y_i - f(x_i,\beta))\cdot x_{ij} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{1}{1+e^{-x_i\beta}} - y_i\right)\cdot x_{ij}$$

然后根据梯度下降法的原理:

$$\beta_{t+1} \leftarrow \beta_t - \alpha\nabla J(\beta)$$

即可进行迭代优化。

Algorithm 2 Gradient Descent

Input: 训练的 $epochs$; 初始化 $\boldsymbol{\beta} = (\boldsymbol{w}, b)$, 学习率 α

Output: $\boldsymbol{\beta}$

```
for each epoch do
    dβ = 0
    for each training sample x_i do
        dβ = dβ + Σ_{i=1}^{n} ( 1/(1+e^{-x_i β}) − y_i ) · x_ij
    end for
    β = β − α ∗ dβ
end for
return  Outputs
```

4.4 正则化

正则化是用来防止模型过拟合而采取的手段。我们对代价函数增加一个限制条件，限制其较高次的参数大小不能过大。

L1 正则化

对于逻辑回归，进行 L1 正则化后的损失函数和梯度分别变为:

$$J(\beta) = -\frac{1}{n}\log\mathcal{L}(\beta) = -\frac{1}{n}\sum_{i=1}^{n}\left[y_i\log\left(P(y_i)\right) + (1-y_i)\log\left(1-P(y_i)\right)\right] + \frac{\lambda}{n}||w||_1$$

$$\frac{\partial J(\beta)}{\partial \beta_j} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{1}{1+e^{-x_i\beta}} - y_i\right)\cdot x_{ij} + \frac{\lambda}{n}sgn(w)$$

L2 正则化

对于逻辑回归，进行 L2 正则化后的损失函数和梯度分别变为:

$$J(\beta) = -\frac{1}{n}\log\mathcal{L}(\beta) = -\frac{1}{n}\sum_{i=1}^{n}\left[y_i\log\left(P(y_i)\right) + (1-y_i)\log\left(1-P(y_i)\right)\right] + \frac{\lambda}{2n}||w||_2^2$$

$$\frac{\partial J(\beta)}{\partial \beta_j} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{1}{1+e^{-x_i\beta}} - y_i\right)\cdot x_{ij} + \frac{\lambda}{n}w$$

Part 5: 实验结果

5.1 总体对比

对于数据集进行了 5 **次** 5 **折交叉验证**，得到如下结果

表 2: Characteristics of the buck converter

模型	测试集准确率（平均）	迭代次数（平均）	参数
GD	80.80%	23099.72	lr=1
GD+L1 正则化	80.63%	22091.08	lr=1,$\lambda = 0.01$
GD+L2 正则化	80.80%	3286.76	lr=1,$\lambda = 1$

注：L1 的正则项系数调整为很小的原因是因为 $||w||_1$ 的导数为 $sgn(w)$, 对于梯度来说过大，导致迭代无法收敛，所以调小正则项系数；对于学习率，我认为只要迭代能收敛，学习率对于结果的影响不大 (有可能陷入局部最优)，所以此次试验不探讨学习率对于准确率的影响，学习率的取值合适即可。

5.2 Loss curve

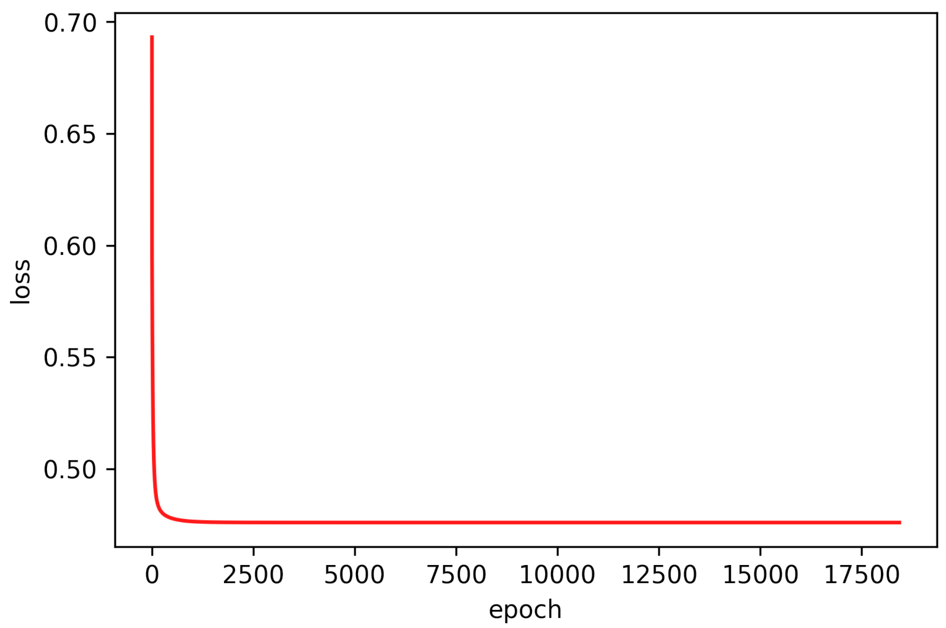

图 2: The Loss curve of GD,learning_rate=1

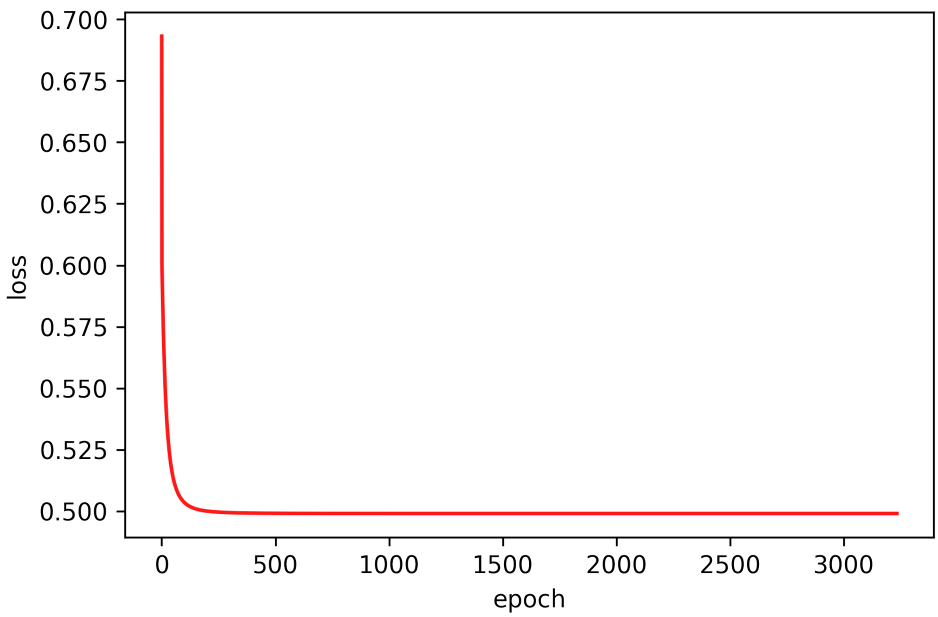

图 3: The Loss curve of GD and L2,$\lambda = 1$,learning_rate=1

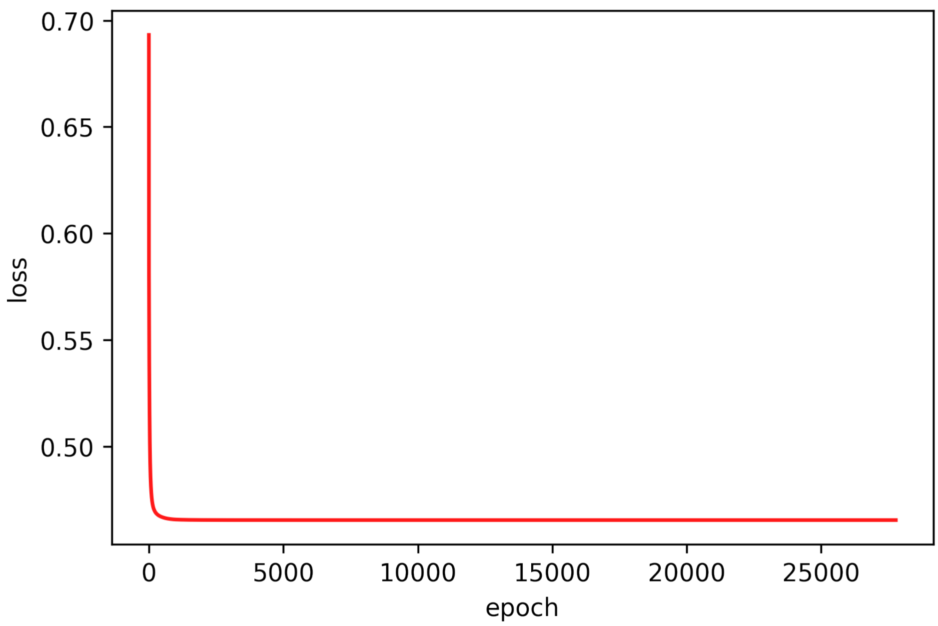

图 4: The Loss curve of GD and L1,$\lambda = 0.1$,learning_rate=1

5.3 实验总结

本次实验完成了逻辑回归的梯度下降求解及其正则化。在实验过程中遇到了以下问题:

- 模型需要迭代很多次，消耗时间长

- 对于 numpy 功能不熟悉，使用循环计算梯度以及损失函数
- 对于 LR 损失函数理解不透彻，矩阵运算维度未想清楚，导致写错，损失函数曲线出现未知原因造成的抖动

解决办法：

- 调大学习率，增加每次梯度变化的大小，但要注意不能太大，否则会不收敛
- 熟练运用 numpy 的矩阵乘法，能极大提高训练计算速度
- 重新手动推导一遍损失函数，熟悉相关知识后再进行代码的编写