# LB1 机器学习概论

PB19151769　　马宇骁

## 1　实验要求

### 1.1　Tasks

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

### 1.2　Requirements

- 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.

## 2　实验原理

实验主要运用 Logistic Regression 的知识进行分类回归和预测，在建立回归的时候需要理解梯度下降法并注意收敛条件。

## 2.1 数据转换

### 2.1.1 类型转换

机器学习模型需要的数据是数字型的，因为只有数字类型才能进行计算。因此，对于各种特殊的特征值，我们都需要对其进行相应的编码，也是量化的过程。因此，对于实验数据集，做出两种不同的数据类型转换方式：

1. Label encoding

   用标签进行编码的意思，即我们给特征变量自定义数字标签，量化特征。此数据集中将 Gender，Married，Dependents，Education，Self_Employed 和 Loan_Status 做该转换

2. One-hot encoding

   独热编码（哑变量）定类型数据。将原始特征变量转换成以原始特征值分类的多维度的变量，并用是否（0,1）这种方式的新特征值替代和量化。此数据集中将 Property_Area 的三类数据分离出来。

### 2.1.2 归一化

正则化的方式有很多，例如：Sigmiod，arctan，log 函数转换，z-score 标准化，min-max 标准化等。由于该数据集需要转换的数据都是非负数据，且使用 arctan 使得数据接近 1 区分很小，故考虑使用 min-max 标准化对表格中的 4 类数据转化。

- min-max 标准化 (Min-max normalization)

$$x^* = \frac{x - min}{max - min}$$

## 2.2 Logistic Regression 逻辑回归

Logistic 回归是一种统计方法，用于根据先前的观察结果预测因变量的结果。它是一种回归分析，是解决二元分类问题的常用算法。逻辑回归也称为二项逻辑回归或二元逻辑回归。如果响应变量有两个以上类别，则称为多项逻辑回归。

逻辑回归的表示方式类似于使用直线方程定义线性回归的方式。与线性回归的显着区别是输出将是二进制值（0 或 1）而不是数值。

$$h_\theta(x) = g\left(\theta^T x\right) = \frac{1}{1 + e^{-\theta^T x}} \tag{1}$$

其中，$\theta_0 + \theta_1 x_1 +, \ldots, +\theta_n x_n = \sum_{i=1}^n \theta_i x_i = \theta^T x.$

应用逻辑回归来预测分类因变量。换句话说，当预测是分类的时使用它，例如，是或否，真或假，0 或 1。逻辑回归的预测概率或输出可以是其中之一，没有中间立场。

对于预测变量，它们可以是以下任何类别的一部分：

- 连续数据：可以在无限尺度上测量的数据。它可以取两个数字之间的任何值。例如以磅为单位的重量或以华氏度为单位的温度。

- 离散的名义数据：适合命名类别的数据。一个简单的例子是头发颜色：金色、黑色或棕色。
- 离散、有序的数据：符合某种规模顺序的数据。例如，以 1 到 5 的等级说明您对产品或服务的满意程度。

逻辑回归分析对于预测事件的可能性很有价值。它有助于确定任何两个类之间的概率。
因此，此次实验使用逻辑回归对 Loan_Status 进行回归，预测判断这个人是否可以贷款。

## 2.3 损失

对数似然函数为：

$$l(\theta)=\log L(\theta)=\sum_{i=1}^{m}\left(y_i \log h_\theta\left(x_i\right)+\left(1-y_i\right) \log \left(1-h_\theta\left(x_i\right)\right)\right)$$

记 $D$ 为对数似然去掉符号的函数，则定义平均损失为：

$$J(\theta)=\frac{1}{n} \sum_{i=1}^{n} D\left(h_\theta\left(x_i\right), y_i\right)$$

## 2.4 梯度下降

### 2.4.1 原理与实现

循环执行以下 3 步骤:

1. 环顾周围找到最陡的一段路
2. 在最陡的一段路上走一段距离
3. 重复以上步骤直到山底

对一个函数应用梯度下降法，就是为了最快地求出函数的全局最小值或者局部最小值；再对应到机器学习问题上，梯度下降法就是为了尽快求出模型代价函数最小值，进而得到模型参数；
所以梯度下降法要解决的问题就是：以最快速度求函数最小值。

$$\theta_j=\theta_j-\alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

计算梯度就是代价函数对 $\theta_i$ 进行复合求导：

$$J\left(\theta_0, \theta_1\right)^{\prime}=2 * \frac{1}{2 m} \sum_{i=1}^{m}\left(h_\theta\left(x_i\right)-y^{(i)}\right) * h_\theta\left(x_i\right)^{\prime}$$

$$h_\theta\left(x_i\right)^{\prime}=\left(\theta_0 x_0+\theta_1 x_1\right)^{\prime}=x_i$$

对于此时的逻辑回归有：

$$
\begin{aligned}
\frac{\partial}{\partial \theta_j} J(\theta) &= -\frac{1}{m} \sum_{i=1}^{m}\left(y_i \frac{1}{h_\theta\left(x_i\right)} \frac{\partial}{\partial \theta_j} h_\theta\left(x_i\right)-\left(1-\mathrm{y}_{\mathrm{i}}\right) \frac{1}{1-h_\theta\left(x_i\right)} \frac{\partial}{\partial \theta_j} h_\theta\left(x_i\right)\right) \\
&= -\frac{1}{m} \sum_{i=1}^{m}\left(y_i \frac{1}{g\left(\theta^{\mathrm{T}} x_i\right)}-\left(1-\mathrm{y}_{\mathrm{i}}\right) \frac{1}{1-g\left(\theta^{\mathrm{T}} x_i\right)}\right) \frac{\partial}{\partial \theta_j} g\left(\theta^{\mathrm{T}} x_i\right) \\
&= -\frac{1}{m} \sum_{i=1}^{m}\left(y_i \frac{1}{g\left(\theta^{\mathrm{T}} x_i\right)}-\left(1-\mathrm{y}_{\mathrm{i}}\right) \frac{1}{1-g\left(\theta^{\mathrm{T}} x_i\right)}\right) g\left(\theta^{\mathrm{T}} x_i\right)\left(1-g\left(\theta^{\mathrm{T}} x_i\right)\right) \frac{\partial}{\partial \theta_j} \theta^{\mathrm{T}} x_i \\
&= -\frac{1}{m} \sum_{i=1}^{m}\left(y_i\left(1-g\left(\theta^{\mathrm{T}} x_i\right)\right)-\left(1-\mathrm{y}_{\mathrm{i}}\right) g\left(\theta^{\mathrm{T}} x_i\right)\right) x_i^j \\
&= -\frac{1}{m} \sum_{i=1}^{m}\left(y_i-g\left(\theta^{\mathrm{T}} x_i\right)\right) x_i^j \\
&= \frac{1}{m} \sum_{i=1}^{m}\left(h_\theta\left(x_i\right)-y_i\right) x_i^j
\end{aligned}
$$

#### 2.4.2 收敛与停止

有 3 种情况一个判断收敛或者哪怕没有收敛但也应该停止：

1. 损失不再下降
2. 迭代次数过多
3. 梯度已经降到 0

否则学习可能无法停止。

### 2.5 预测

通过回归学习过程中得到的 $\theta$ 的值带回式1中，使用测试集计算得出的值如果大于 0.5 则为 1，小于则为 0. 以此作为预测。

## 3 实验实现

具体代码见附件 Loan.ipynb 与 Logis.py 文件，使用学习率为 0.01 时的学习过程记录如下：

```
  第2000次
  Train Loss: 0.468
用时0.078998秒
第4000次
  Train Loss: 0.466
用时0.148096秒
第6000次
  Train Loss: 0.465
用时0.217119秒
第8000次
```

```
  Train Loss: 0.465
用时0.293116秒
第10000次
  Train Loss: 0.465
用时0.366115秒
第12000次
  Train Loss: 0.465
用时0.435113秒
第14000次
  Train Loss: 0.465
用时0.506114秒
第16000次
  Train Loss: 0.465
用时0.582115秒
第18000次
...
用时3.250627秒
第92000次
  Train Loss: 0.465
用时3.321548秒
```

损失曲线绘制如图：可以发现，收敛速度很快。此时的预测准确率（随机打乱 9:1 的训练测试集比例）：

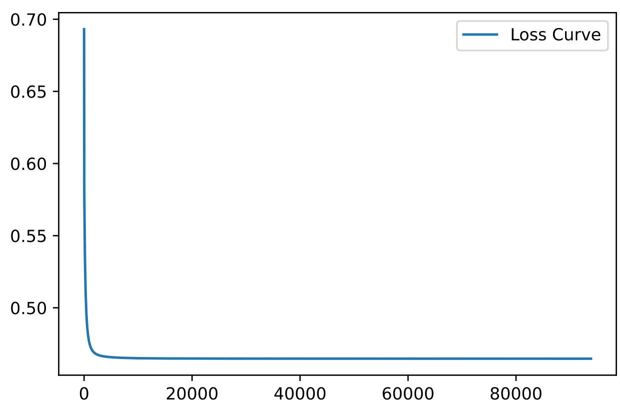


图 1: Loss Curve

0.8541666666666666

还是不错的。

经过尝试其他的学习率如 0.1，可以发现收敛速度更快（有运气的成分），但由于集合的原因此时看不出预测准确率的很大区别。