# 实验3《logistic回归预测二分类》

实验学时： 2 实验地点： 二综204 实验日期： 2019/11/01

1. **问题描述**

学会使用学习到的逻辑回归的知识，手动使用梯度下降方法，通过给定的相关数据来完成年薪是否高于50k的二分类预测任务。。

1. **设计简要描述**

**1．**数据集以及任务介绍

#### 1.1 任务

二分类任务：确定一个人是否年收入超过5万美元。

#### 1.2 数据集

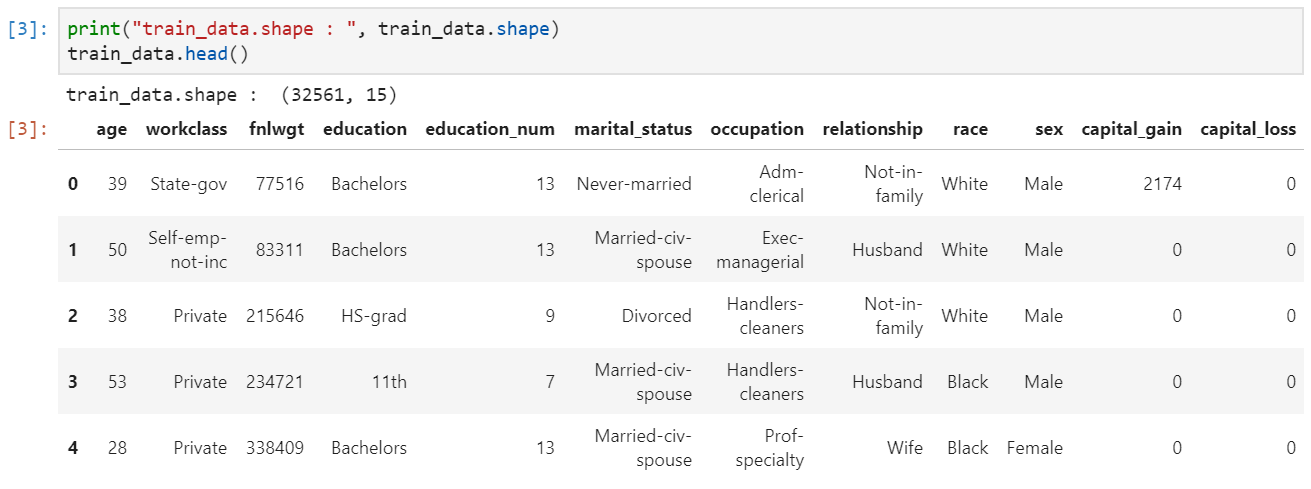
贝克尔(Barry Becker)从1994年的人口普查数据库中提取的数据。利用以下条件提取出一组相当清晰的记录:

((AGE>16) && (AGI>100) && (AFNLWGT>1) && (HRSWK>0)).

#### 1.3数据属性信息

总共是提供了两个数据集：train.csv，test.csv。并且一共是有以下属性：

age, workclass, fnlwgt, education, education num, marital-status, occupation relationship, race, sex, capital-gain, capital-loss, hours-per-week, native-country, make over 50K a year or not（income）



#### 1.4 提供的特征属性格式

对于数据集：X\_train,Y\_train,X\_test：

1. 对于train.csv中离散的特征属性（例如work\_class,education...）——使用独热（onehot）编码来处理
2. 对于train.csv中连续的特征属性（例如age,capital\_gain...）——则是直接用数据集中的原值
3. 在X\_train, X\_test中 : 每行包含一个表示样本的106-维特性
4. Y\_train: label = 0 表示  “<= 50K” 、 label = 1表示“ >50K ”

#### 1.5 提交的数据格式

结果输出到csv文件中：

第一行必须为id,label，从第二行开始为预测结果

每行分别为id以及预测的label，请以逗号分隔

2.实现方法—Logistic Regression

#### 2.1 加载数据

首先通过pandas加载数据集：



#### 2.2 逻辑回归模型

从算法描述，我们实现:

* \_sigmoid: 来计算输入的sigmoid. 使用np.clip来避免溢出，将数据夹在[ 1e-6 ,1-1e-6]之间.

示例：np.clip(1 / (1.0 + np.exp(-z)), 1e-6, 1-1e-6)

* get\_prob: 在给定权重和偏差的情况下，找出模型预测输出1的概率

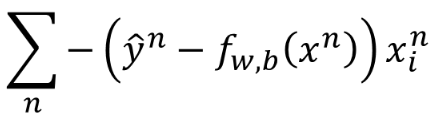
示例：\_sigmoid(np.add(np.matmul(X, w), b))

* infer: 如果概率>为0.5，则输出1，否则输出0。

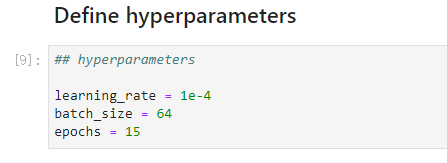
示例： np.round(get\_prob(X, w, b))

* \_crossentropy: 计算模型输出和真实标签之间的交叉熵。
* \_gradient : 通过数学推导损失函数，各个参数的梯度为:

各个权重w的梯度和如下：



#### 2.3训练过程



#### 2.4画图函数

导入画图函数包：

import matplotlib.pyplot as plt

%matplotlib inline

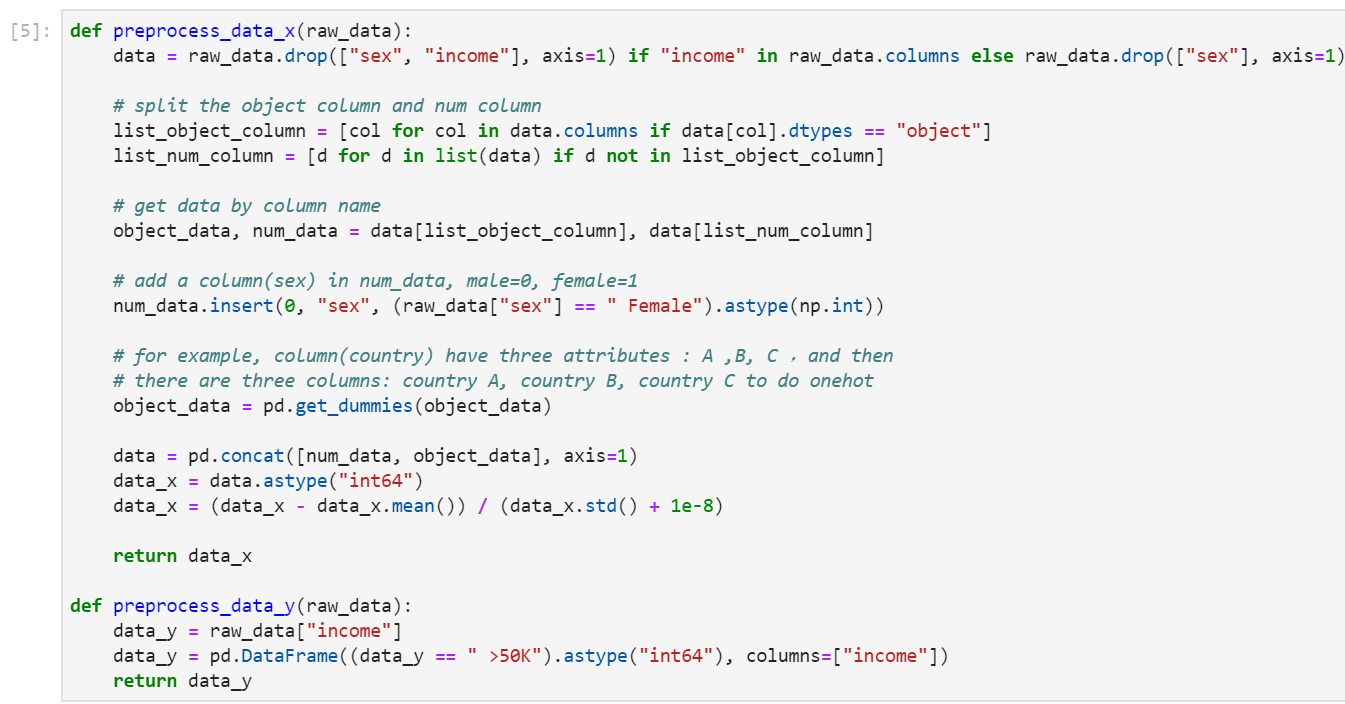
画出训练集以及验证集中loss 以及 准确率 的对比图形

#### 2.5 测试数据的使用预测二分类

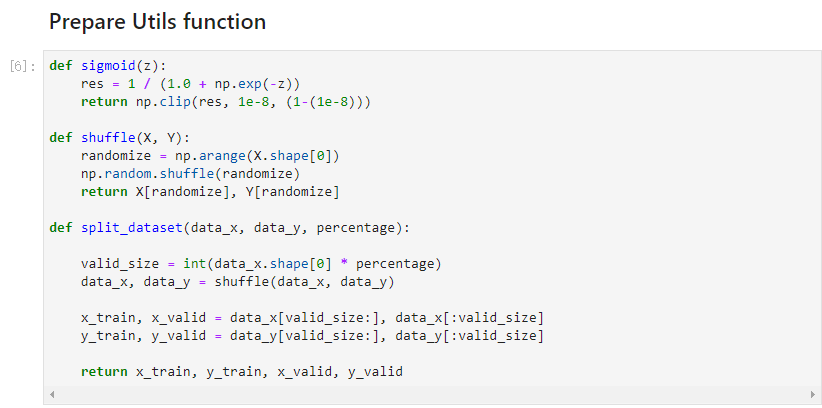
1. **程序清单**
2. **数据读取**

****

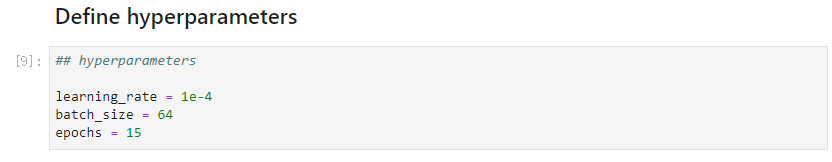
1. **准备训练集，测试集数据**

****

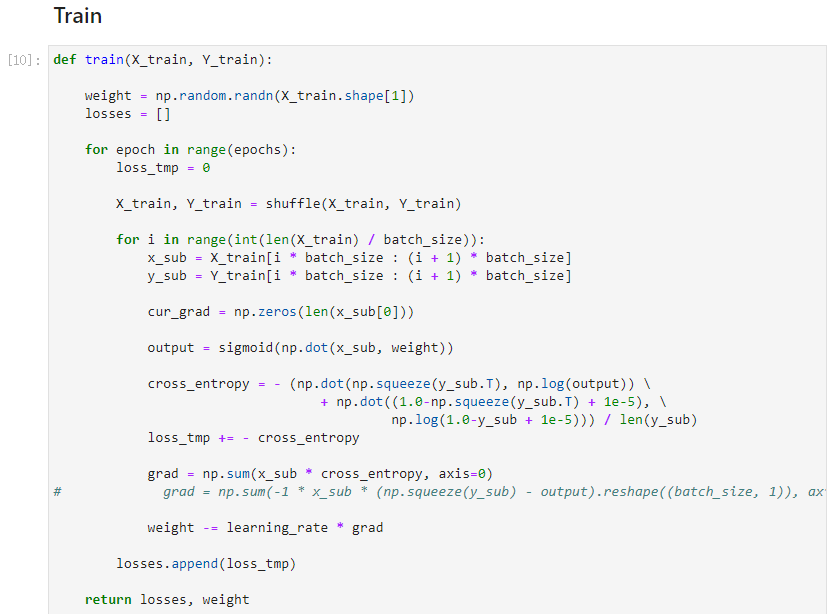
1. **书写所要用到的工具类**



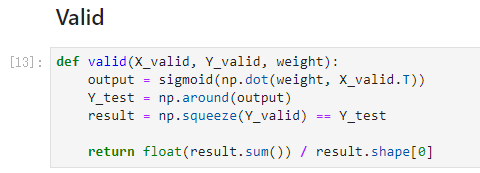
1. **定义超参数**



1. **训练**

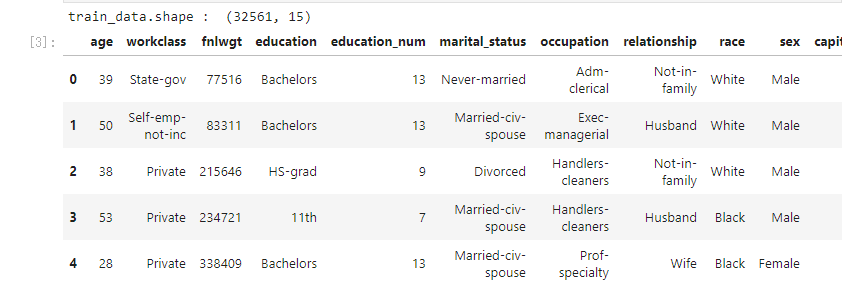


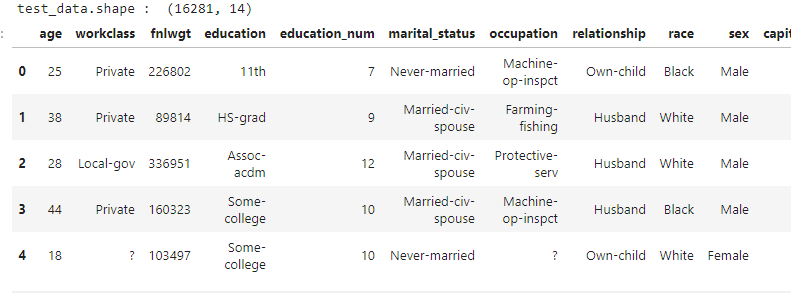
1. **测试**



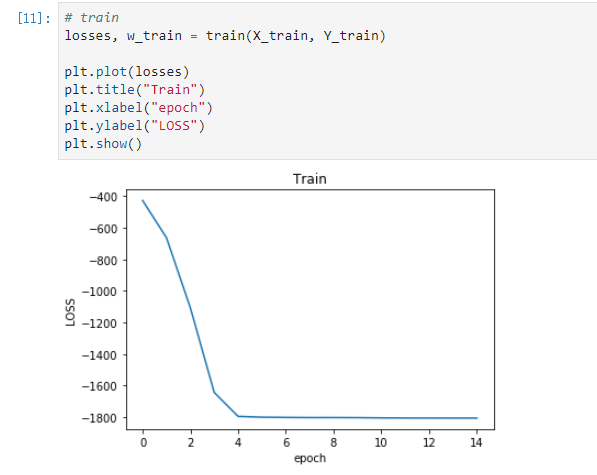
1. **保存测试结果**

****

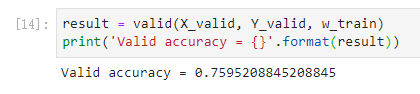
1. **结果分析**
2. **查看训练数据表表头部分**
3. **查看测试数据表表头部分**



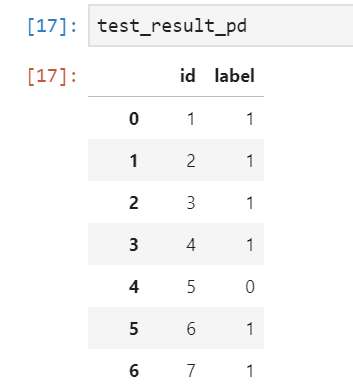
1. **训练过程的loss变化**



1. **验证结果**



1. **测试集后训练结果**

****

1. **调试报告**
2. **所遇到的问题**

本次实验使用了部分上次实验的代码，重写了train()和valid()函数，进展较快。在定义训练时的loss时，使用简单的加减操作所获得的loss比使用cross\_entropy所获得的loss验证效果要好，感觉应该是单层神经元的缘故吧。

矩阵计算方面，各种形状的矩阵乘积还有点让人头晕，目前只想到了debug去查看矩阵大小，不知道有没有其他好的办法。

1. **经验和体会**

感觉使用框架和自己手动写代码还是有点差距的，写简单的前馈神经网络还行，如果复杂起来就很难操作了，主要是手动计算梯度非常费劲。不过，这样确实能增强自己对原理的理解。