**Lab 1 实验报告**

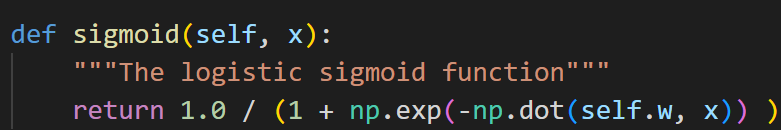
PB20000296 郑滕飞

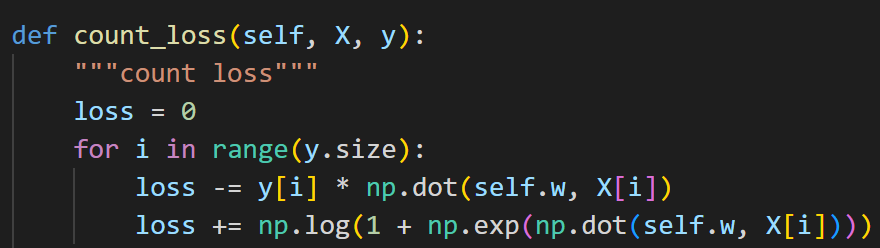
**py文件：**

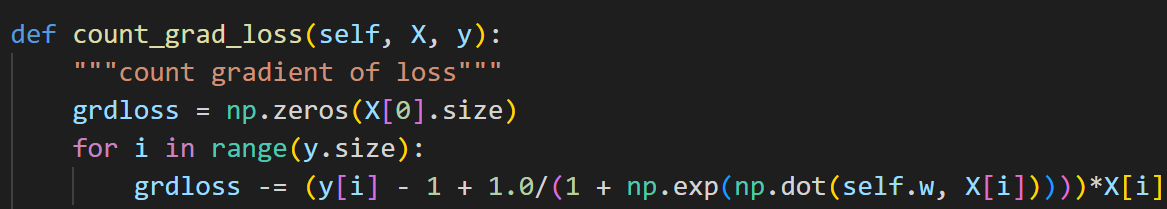
**1、整体框架**

由于采用梯度下降法并需要绘制损失曲线，需要计算损失函数及其梯度。而为了作出预测，需要计算Sigmoid函数。这三个函数直接根据ppt提供的公式实现即可：

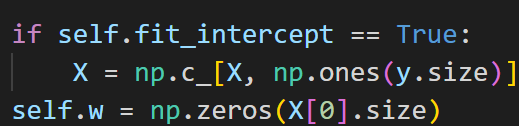
[此处计算损失函数及其梯度是按照ppt的公式，并没有对样本数归一化，具体对比见第三部分]



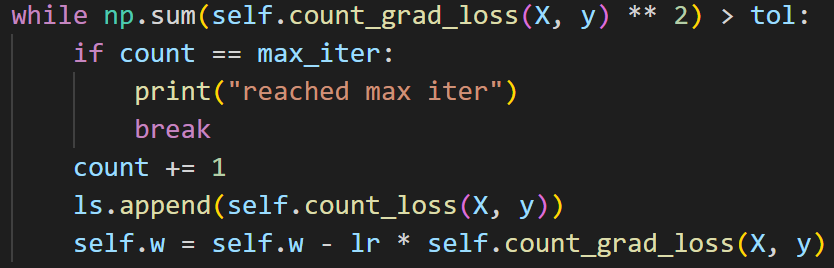




对于学习的部分，需要先考虑fit的写法。注意到，匹配截距相当于给X增加一列全为1的列，因此这个操作可直接于最开始进行，之后再根据当前X的列数建立w即可。如果匹配截距，w的最后一个分量即为截距：



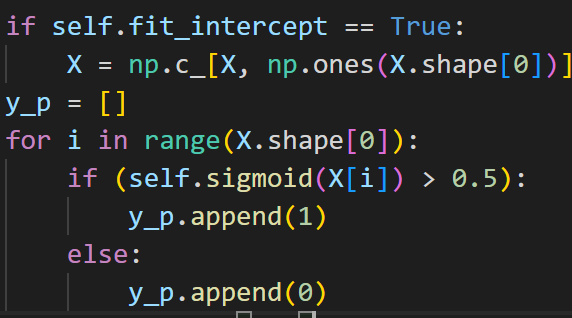
梯度下降的过程则直接按照算法进行：



由于在count\_loss中计算出了损失的值，直接将这些值储存成向量ls返回，即可绘制损失曲线。

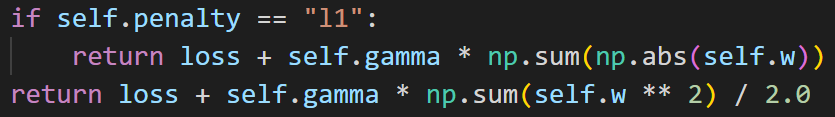
lr用于控制学习率，也即梯度下降的速度，而tol和max\_iter从两个不同方向确定循环的界。在初值设定tolerance 1e-4与max\_iter 1e3时，一般是由于到达最大迭代次数而返回。

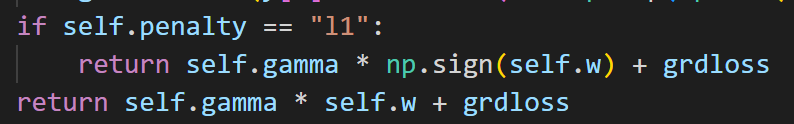
预测部分，直接通过Sigmoid函数生成预测值：



**2、正则化**

刚才的过程中，并没有考虑penalty参数与gamma参数形成的正则化项。通过查阅资料发现，penalty为l1相当于在损失中添加一项，而为l2则相当于添加。前者的梯度为gamma倍的sgn(w)，其中sgn为符号函数，而后者的梯度即为gamma倍的w。于是，通过正则化参数可以得到最终的损失与损失梯度：



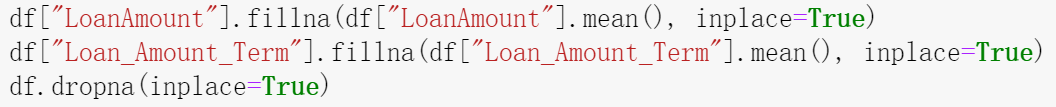


可以发现，正则化由于加入了w的模长作为最小化条件，控制了w的模不会过大，这点在迭代次数升高时尤为明显[见后方参数比较]。

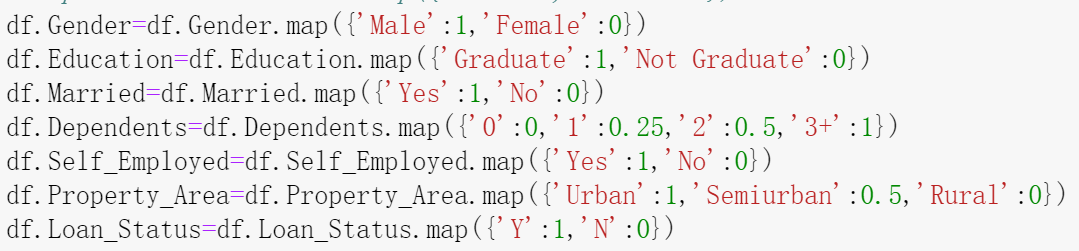
**ipynb文件：**

**1、Data Cleaning & Encode**

由于较多项目有Null项，对连续的属性采取平均值填充后再去掉Null：



而编码时，为了之后归一化方便，离散属性的编码均使用0到1间的实数：

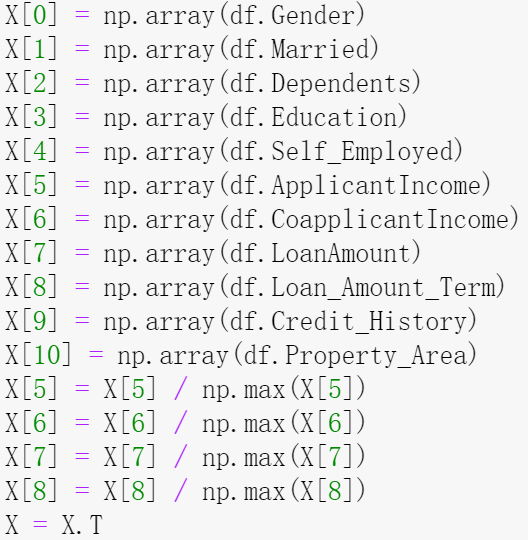


对后代数量，此处假设1、2、3+的比例是1:2:4。

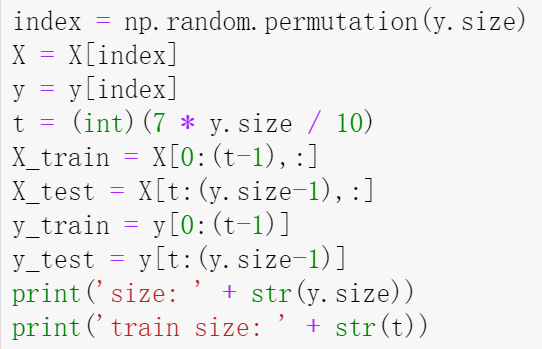
**2、Data Process**

这块分为四个部分：导入numpy向量/矩阵、归一化、随机打乱与按比例划分训练集/测试集。

导入部分先用行导入列，再进行转置，归一化则需要利用属性的最大值进行：

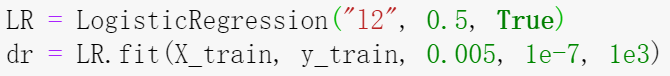


由于数据集不大，且正例与反例比例接近2:1，差别不大，不需要进行过采样/欠采样等。此外，原数据中分布就已经随机排列，事实上不打乱也可以直接学习。不过为了检验效果，仍随机打乱样本，并将前一部分划分为训练集[默认比例70%]，后一部分为测试集：

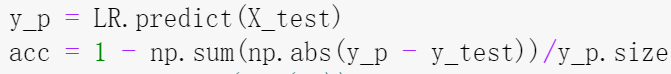


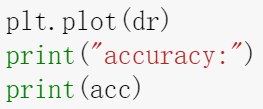
**3、Train & Test**

采用py文件中写好的类进行训练：



这里的dr保存了每次迭代损失所构成的向量，测试后直接用其进行画图：





**展示与对比：**

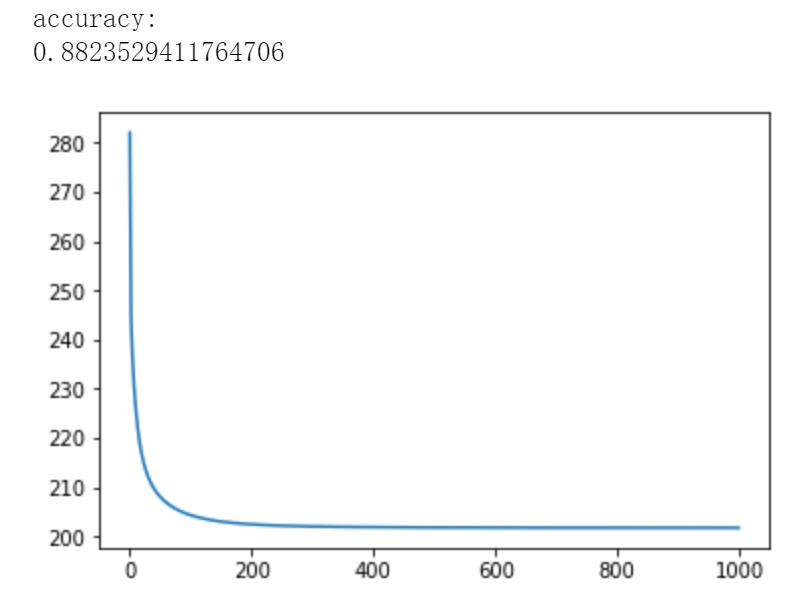
**1、最佳准确率及其损失曲线**

数据集分割：数据集大小511 训练集大小408(80%) 随机分割

参数：

penalty = ’l2’ gamma = 0 fit\_intercept = False

lr = 0.005 tol = 1e-7 max\_iter = 1e3

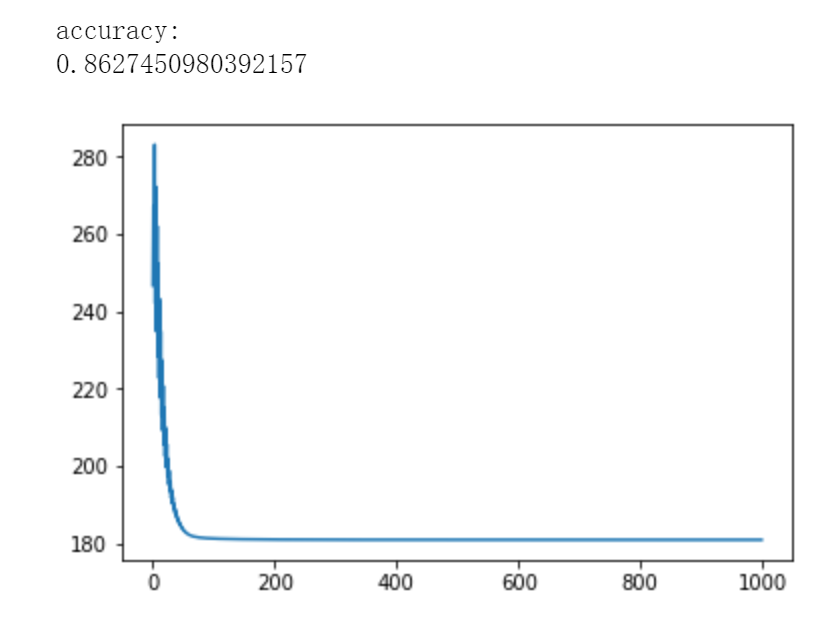


数据集分割：数据集大小511 训练集大小357(70%) 随机分割

参数：

penalty = ’l2’ gamma = 0.45 fit\_intercept = True

lr = 0.007 tol = 1e-7 max\_iter = 1e3



**2、参数影响**

默认参数为

penalty = ’l2’ gamma = 0 fit\_intercept = True

lr = 0.01 tol = 1e-4 max\_iter = 1e3

以下叙述参数影响时，未提及的参数均为默认参数。

首先，关于loss的归一化问题，根据计算过程不难发现，在计算过程中对loss进行归一化，相当于对学习率乘了样本量的倒数，因此，loss归一化的影响可直接通过学习率进行判别。

其次，由于数据集本身是乱序分布，以下为了对比不同参数造成的影响，去掉了随机打乱的步骤，而是直接以前70%作为训练集，后30%作为测试集。

学习率与迭代次数的影响，从直觉上来说，学习率越低则越稳定，但收敛速度会有一定下降，具体的列表如下[表格最左代表lr，最上代表max\_iter]：

|  |  |  |  |
| --- | --- | --- | --- |
| acc | 5e2 | 1e3 | 2e3 |
| 1e-3 | 0.824 | 0.824 | 0.810 |
| 5e-3 | 0.804 | 0.797 | 0.797 |
| 1e-2 | 0.824 | 0.824 | 0.824 |
| 2e-2 | 0.418 | 0.817 | 0.424 |

|  |  |  |  |
| --- | --- | --- | --- |
| loss | 5e2 | 1e3 | 2e3 |
| 1e-3 |  |  |  |
| 5e-3 |  |  |  |
| 1e-2 |  |  |  |
| 2e-2 |  |  |  |

可以发现，lr在0.005时曲线都较为稳定，而0.01开始已经出现了明显的震荡。当学习率进一步增大时，过拟合也变得更加明显(lr为0.02时迭代次数升高后准确率大幅下降)。

下面观察正则化项的影响[作为参考，上方默认参数时准确率为0.824]。

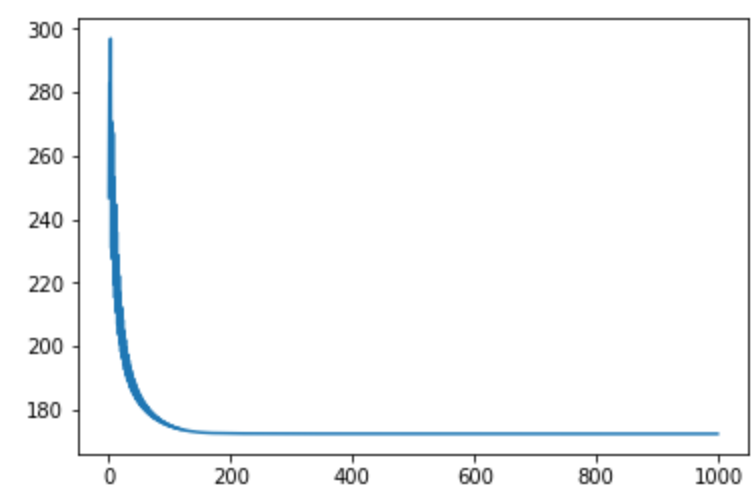
|  |  |  |  |
| --- | --- | --- | --- |
| acc | 0.2 | 0.5 | 2 |
| l1 | 0.837 | 0.837 | 0.778 |
| l2 | 0.837 | 0.837 | 0.837 |

|  |  |  |  |
| --- | --- | --- | --- |
| loss | 0.2 | 0.5 | 2 |
| l1 |  |  |  |
| l2 |  |  |  |

在正则化项增强时，损失曲线事实上变得更加波动了，这也符合其防止过拟合、保持一定损失的要求。大部分情况下，增加正则化项后结果可以变得更好。不过，当它过强时，由于造成的波动太大，拟合精度事实上降低了。

此外，l2正则化的效果比l1更加明显，这是由于其梯度直接与w的值相关，而不是只与符号相关。

最后，不匹配截距时，收敛事实上变得更加稳定了(这很可能是因为归一化后截距参数造成的影响是很大的)，但准确率下降到了0.804。



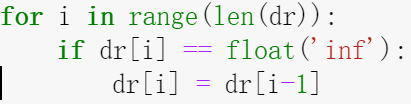
[对于tol参数，由于其事实上影响的是迭代次数，可以通过迭代次数的情况观察出它的可能影响。其在实际操作时可以作为一个终点，也有防止过拟合的作用。]

**2、数据集操作影响**

有关归一化：若不进行归一化，由于数字的量级不同，收敛速度会慢非常多，结果也不尽人意，在其他参数默认时不进行归一化的结果如下：

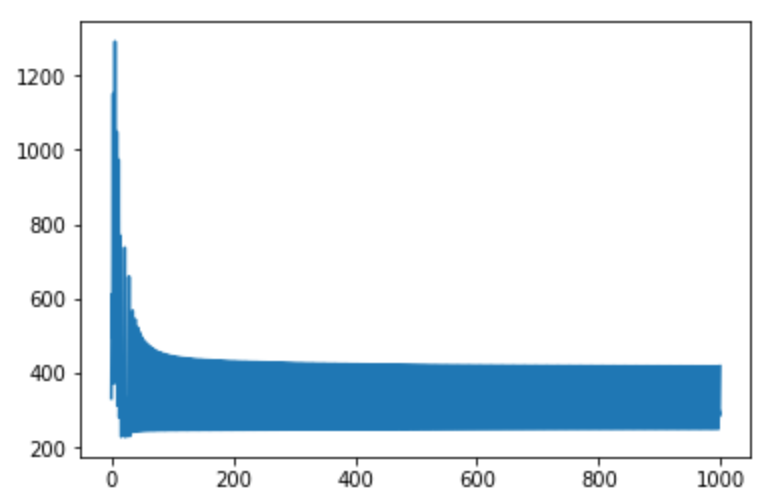
|  |  |  |  |
| --- | --- | --- | --- |
| iter | 1e3 | 5e3 | 1e4 |
| loss |  |  |  |
| acc | 0.294 | 0.706 | 0.307 |

\*由于loss计算过程出可能无现穷值，这时需要进行“去头”才能画出曲线：



正因如此，最终的“稳定”并不是真实的稳定值，而是由于无穷值被填充为了上一个值，此后的loss仍然非常大。

有关去掉NA值：如果把所有的Null全部去掉，样本个数会进一步减少。去掉离散值Null后剩余511个样本，而去掉全部后只剩下480个，学习效果也普遍更差[默认参数准确率0.811，随机打乱后平均基本小于80]：



**4、算法评价**

参数[默认参数加入一定正则化]：

penalty = ’l2’ gamma = 0.3 fit\_intercept = True

lr = 0.01 tol = 1e-4 max\_iter = 1e3

10次随机7:3留出的结果：

.817 .791 .817 .778 .778 .817 .830 .797 .843 .837

\*平均为81.05%

**总结：**

虽然乍看实验文档啥也没看明白，探索、写出自己的学习器还是颇有趣的过程。

\*调参真的会上瘾啊（恼