# 逻辑回归算法报告

## 实验内容

数据集来源：

https://archive.ics.uci.edu/ml/datasets/HIV-1+protease+cleavage

## 实验任务

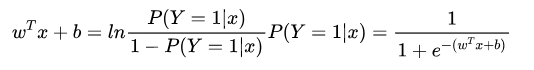
总共有20种氨基酸，数据集中是8个氨基酸组成的数据特征，对数据进行2分类，输出表示已切割（+1）或未切割（-1）。

## 实验算法

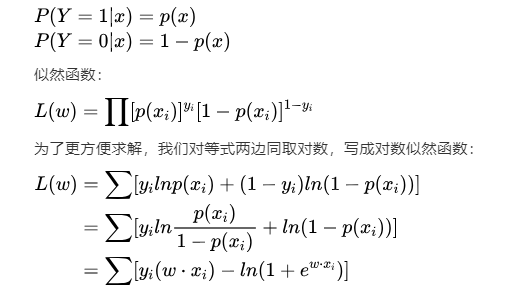
本次实验采用逻辑回归算法进行二分类。

逻辑回归的本质是：假设数据服从某个分布，就可以用极大似然对参数做估计。

#### 3.1 Logistic 回归



#### 3.2 代价函数





## 实验数据处理

对20种氨基酸用one-hot的编码方式。将数据集中8种氨基酸，每个都用20维的向量表示其出现的是哪一个，所以每个样本就有20\*8=160维。经过这种转化后，得到的矩阵X作为输入。

## 实验过程和结果

有4个数据集，采用其中某个数据训练，另外三个来测试。

#### 5.1调包的方式

需要使用的包有：sklearn.linear\_model中的LogisticRegression，用来构建逻辑回归模型。sklearn.metrics中的classification\_report用来评估实验结果，能显示出每个类的精确度，召回率，F1值。

直接通过LogisticRegression构建模型，然后去fit训练数据，最后predict测试数据。

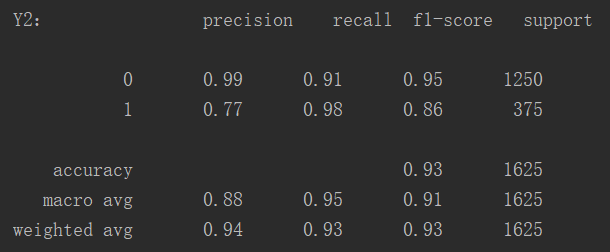
Y1: 746Data.txt

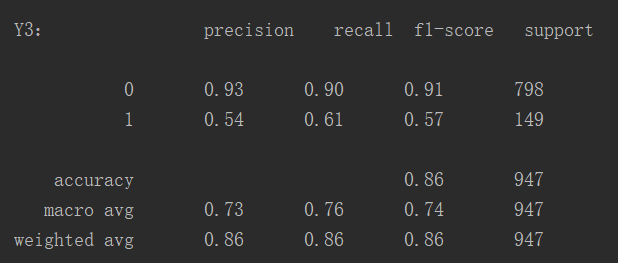
Y2: 1625Data.txt

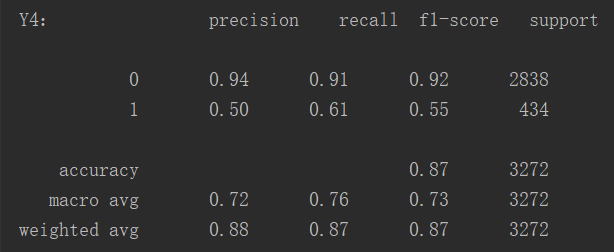
Y3: impensData.txt

Y4: schillingData.txt

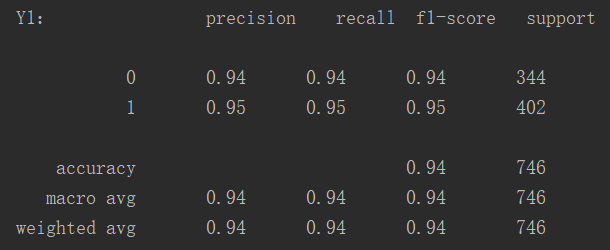
1. 用“746Data.txt”做训练，其他做测试得到的结果如下：

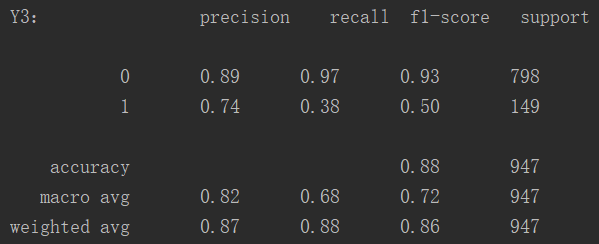


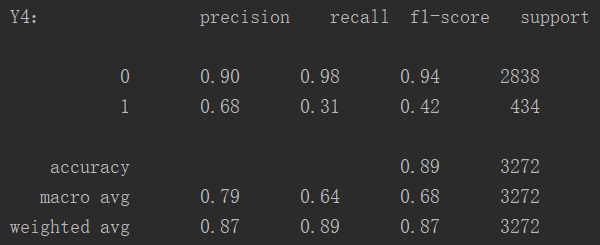




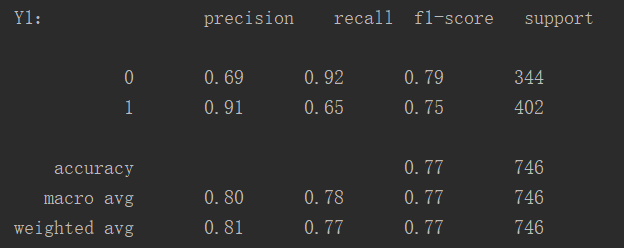
1. 用“1625Data.txt”做训练，其他数据集做测试。

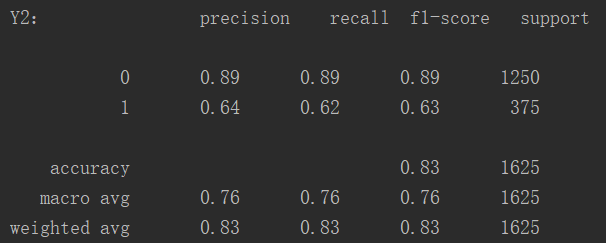


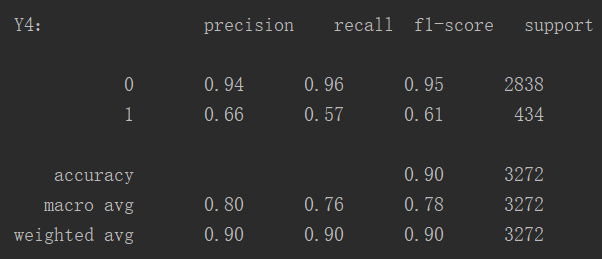




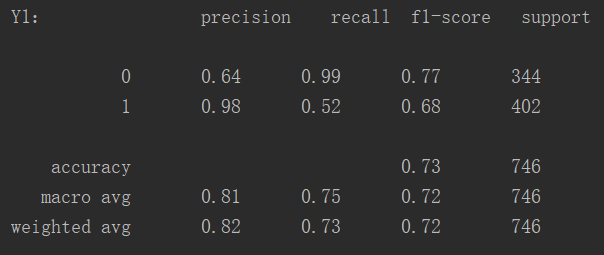
1. 用“impensData.txt”做训练，其他数据集做测试。

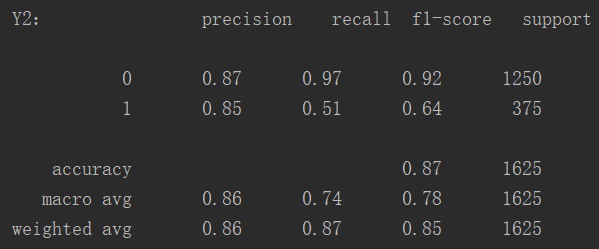


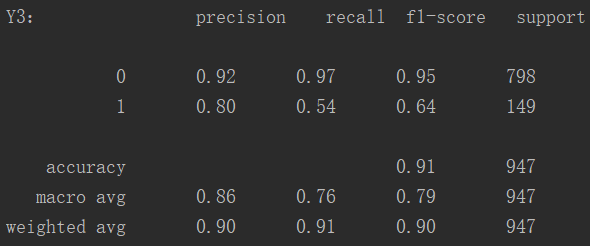




1. 用“schillingData.txt”做训练，其他数据集做测试。







#### 5.2用numpy手写logistic回归

（1）设置logistic回归模型的初始化参数

penalty="l2",gamma=0,fit\_intercept=True

默认设置beta作为要学习的参数。

默认设置惩罚项为l2,gamma为正则化系数，fit\_intercept为截距b,为正就在X中的首列增加一列1。

（2）定义负对数似然函数作为代价函数（def \_NLL）

nll=-np.log(y\_pred[y==1]).sum()-np.log(1-y\_pred[y==0]).sum()

penalty=0.5\*self.gamma\*np.linalg.norm(self.beta,ord=order)\*\*2

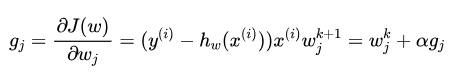
return (penalty+nll)/N





(3) 定义代价函数的梯度（def\_NLL\_grad）

采用随机梯度下降





l1norm=lambda x:np.linalg.norm(x,1)

d\_penalty=gamma\*beta if p=="l2" else gamma\*l1norm(beta)\*np.sign(beta)

return -(np.dot(y-y\_pred,X)+d\_penalty)/N

(4)定义fit拟合函数（def fit(X,y,lr=0.01,tol=1e-3,max\_iter=1e5)）

先初始化beta参数，循环max\_iter次数，通过sigmoid(XW)得到预测值，计算当前loss,

如果Loss<tol，停止迭代，否则就更新beta的参数，因为使用的是负对数似然函数，所以要梯度下降。用（参数-=学习率\*梯度）

self.beta=np.random.rand(X.shape[1])

for \_ in range(int(max\_iter)):

y\_pred=sigmoid(np.dot(X,self.beta))

loss=self.\_NLL(X,y,y\_pred)

if l\_prev-loss<tol:

return

l\_prev=loss

self.beta-=lr\*self.\_NLL\_grad(X,y,y\_pred)

（5）预测(def predict)

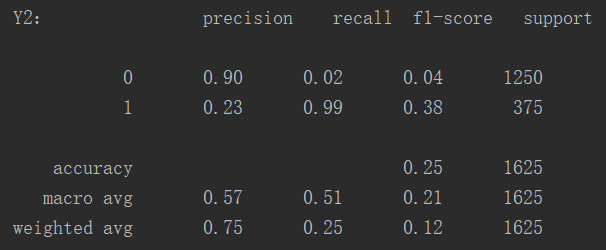
if self.fit\_intercept:

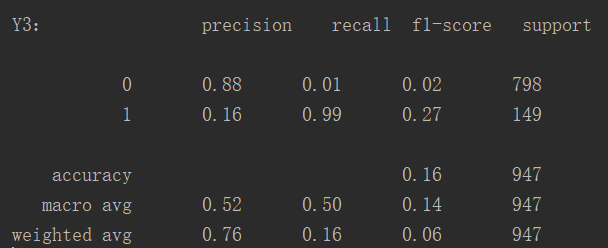
X=np.c\_[np.ones(X.shape[0]),X]

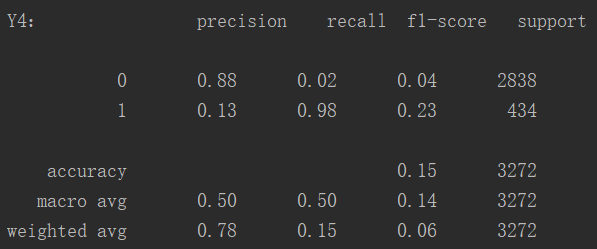
return sigmoid(np.dot(X,self.beta))

采用和调包的一样的划分数据集的方式，结果如下：

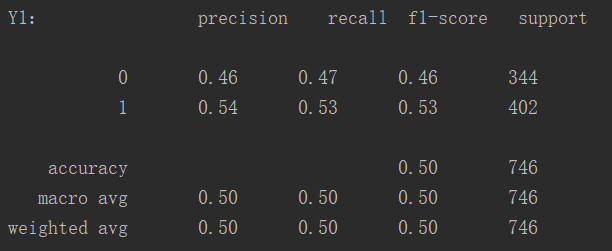
1. 用“746Data.txt”做训练，其他做测试得到的结果如下：

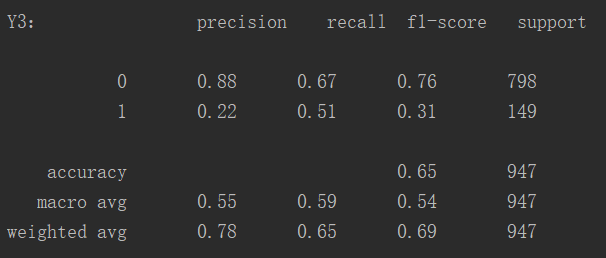


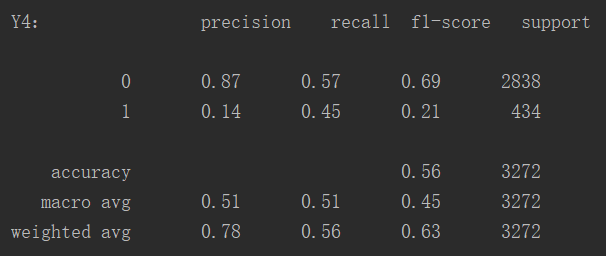




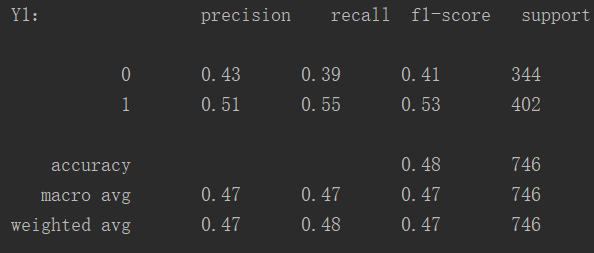
1. 用“1625Data.txt”做训练，其他数据集做测试。

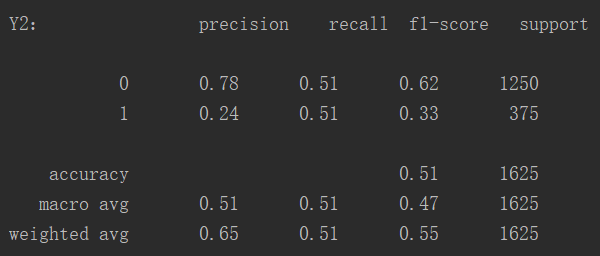


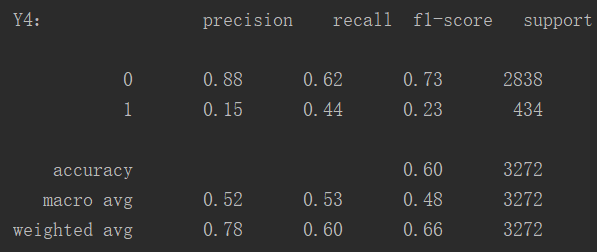




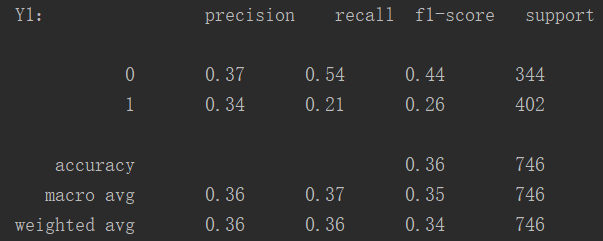
1. 用“impensData.txt”做训练，其他数据集做测试。

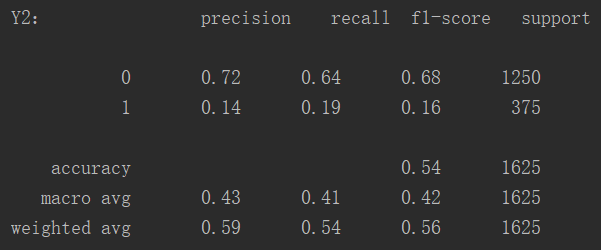


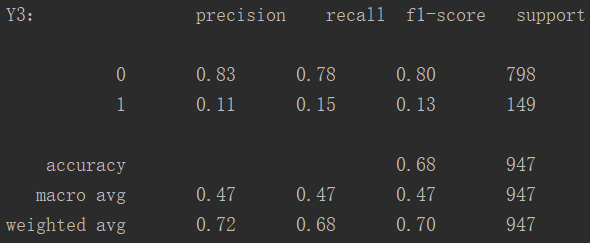




1. 用“schillingData.txt”做训练，其他数据集做测试。

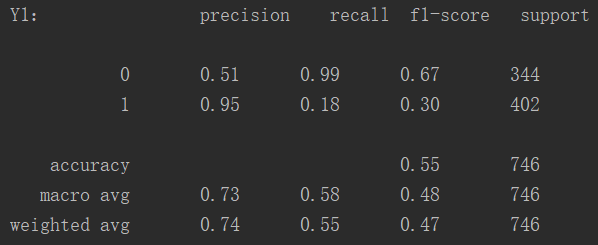


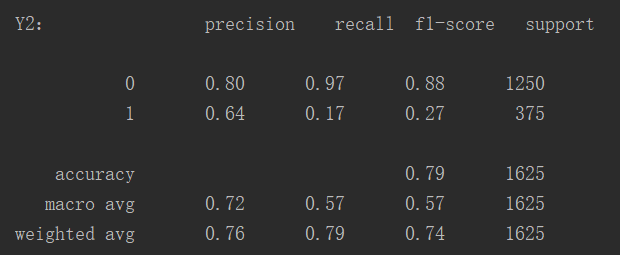


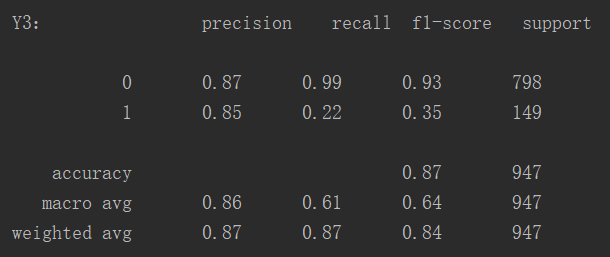


因为效果不理想，重新调整fit拟合参数后，效果变好了一些。学习率，误差阈值，最大迭代次数的选择。

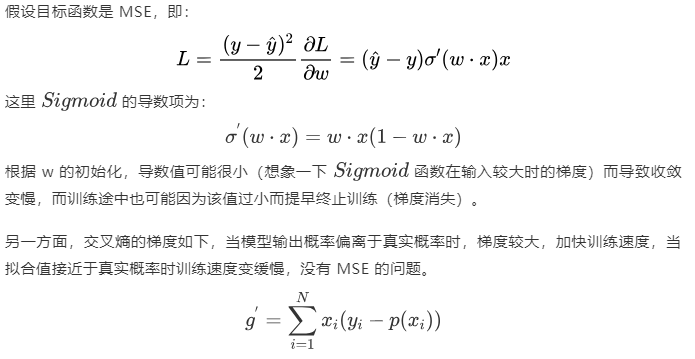
fit(self,X,y,lr=0.01,tol=1e-5,max\_iter=1e7):







## 总结



参考：

<https://mp.weixin.qq.com/s/M4mKicODrulLbXafJfN_AQ>（一文详尽系列之逻辑回归）

<https://github.com/ddbourgin/numpy-ml/blob/master/numpy_ml/linear_models/lm.py>

分享：

https://github.com/ddbourgin/numpy-ml