

# Lab2 实验报告

PB19151769 马宇骁

实验目标:

在 conda 的 python3.6 上实现线性分类算法, 朴素贝叶斯分类器, SVM 算法等。

## Part1 机器学习

线性分类算法:

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(X^{(i)}) - y^{(i)})^2 + \lambda ||\theta||_2^2$$

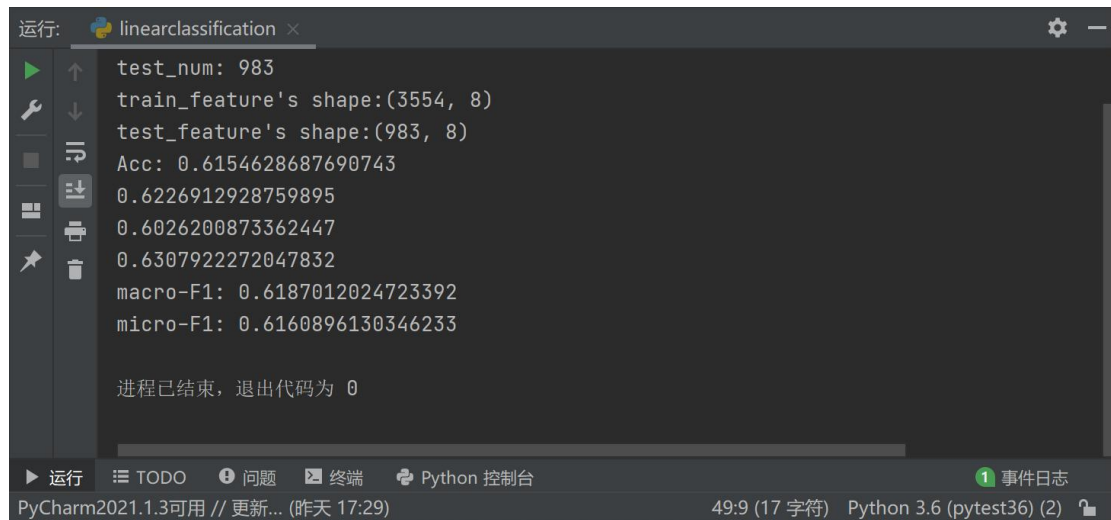
根据最小二乘法, 求导:

$$\frac{\partial J(\theta)}{\partial \theta} = \frac{1}{m} X^T (X\theta - Y) + 2\lambda \theta = 0$$

故,

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

最终得到迭代矩阵, 算得结果如下:



```
运行: linearclassification x
test_num: 983
train_feature's shape:(3554, 8)
test_feature's shape:(983, 8)
Acc: 0.6154628687690743
0.6226912928759895
0.6026200873362447
0.6307922272047832
macro-F1: 0.6187012024723392
micro-F1: 0.6160896130346233

进程已结束, 退出代码为 0

运行 | TODO | 问题 | 终端 | Python 控制台 | 1 事件日志
PyCharm2021.1.3可用 // 更新... (昨天 17:29) 49:9 (17 字符) Python 3.6 (pytest36) (2)
```

分析:

在处理中途不论怎么运行 acc 都为 0, 预测准确度也很低, 经过 print 和思考, 发现预测结果必须为整型而线性回归结果为浮点数, 因此考虑  $y = \text{np.round}(y, 0)$  处理为整数, 再次运行无误。

朴素贝叶斯分类器：

$$P(C) = \frac{|D_c|+1}{|D|+N} \quad P(X_i|C) = \frac{|D_{c,x_i}|+1}{|D|+N_i}$$

根据合适的数据集即可得到贝叶斯分类器的分类结果，但是，若某个属性值在训练集中没有与某个类同时出现过，则直接基于条件概率与先验概率的计算公式进行计算，将出现不论其它属性如何，该类的判别概率都为 0 的问题。为了避免避免其它属性携带的信息被训练集中未出现的属性值归 0，在估计概率值时要使用“平滑”的手段，通常使用的方法为拉普拉斯修正。现在用  $N$  表示训练集  $D$  中的类别数， $N_i$  表示第  $i$  个属性的取值数。

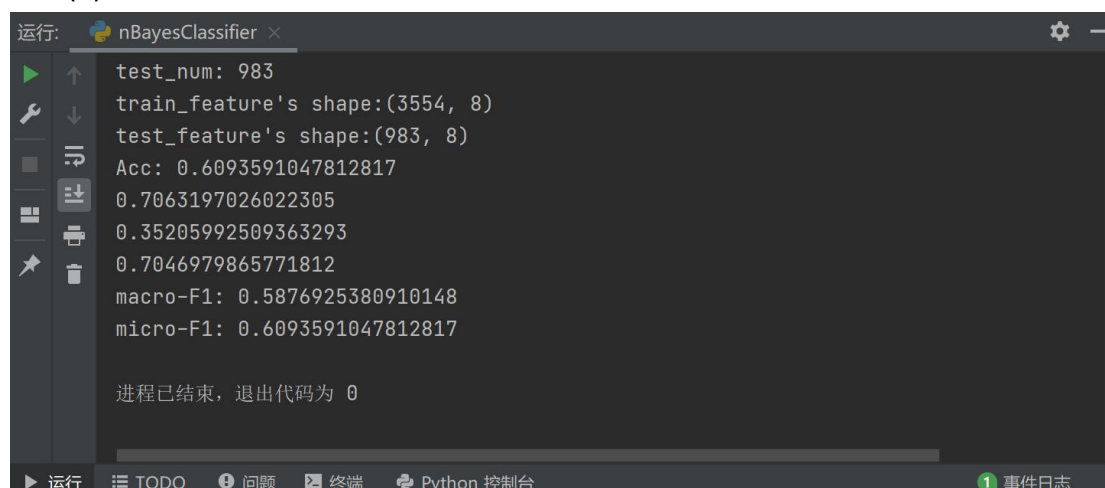
若数据连续，则考虑用高斯分布近似处理：

$$P(X_i|C) = \frac{1}{\sqrt{2\pi}\sigma_{c,i}} \exp\left(-\frac{(x_i - \mu_{c,i})^2}{2\sigma_{c,i}^2}\right)$$

最终计算

$$P(C) \prod_{i=1}^d P(X_i|C)$$

取  $P(C)$  最大值的  $C$  记为预测值。结果如下：



```
运行: nBayesClassifier ×
test_num: 983
train_feature's shape: (3554, 8)
test_feature's shape: (983, 8)
Acc: 0.6093591047812817
0.7063197026022305
0.35205992509363293
0.7046979865771812
macro-F1: 0.5876925380910148
micro-F1: 0.6093591047812817

进程已结束，退出代码为 0
```

SVM:

数据集:  $T = \{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$

$(x_i, y_i)$  为一个训练样本， $x_i$  为一个实例， $y_i$  为该实例的分类。 $y_i$  为 1 时为正例， $y_i$  为 -1 时为负例。

学习目标：在特征空间寻找一个分离超平面（将两类正确分类并且使得两类几何间隔最大）： $w \cdot x + b = 0$ ，该超平面由法向量  $w$  和截距  $b$  决定。法向量指向的一侧为正类。

相应的分类决策函数： $f(x) = \text{sign}(w \cdot x + b)$  称为线性可分支持向量机

求解  $a$ ：

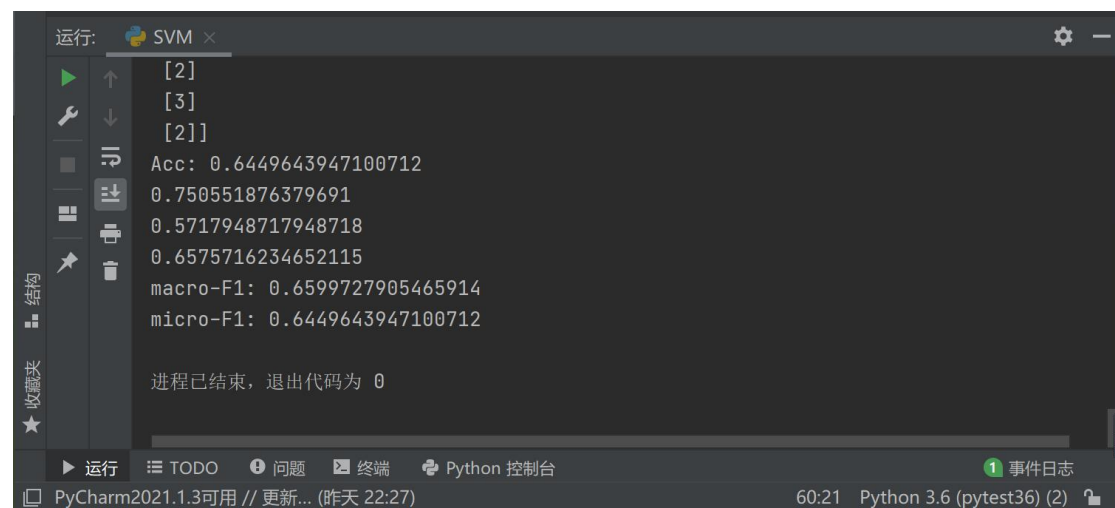
$$\begin{aligned} \min_{\alpha} \quad & \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j (x_i \cdot x_j) - \sum_{i=1}^N \alpha_i \\ \text{s.t.} \quad & \sum_{i=1}^N \alpha_i y_i = 0 \\ & 0 \leq \alpha_i \leq C, \quad i=1, 2, \dots, N \end{aligned}$$

求解 w, b:

$$\begin{aligned} w^* &= \sum_{i=1}^N \alpha_i^* y_i x_i \\ b^* &= y_j - \sum_{i=1}^N y_i \alpha_i^* (x_i \cdot x_j) \end{aligned}$$

$$0 < \alpha_j^* < C$$

当数据中有一些奇异点，如下图中跑上来的两个红色的点。只是这极个别的点而导致的非完全线性可分。此时，需要对每个样本点增加一个松弛变量。结果如下：



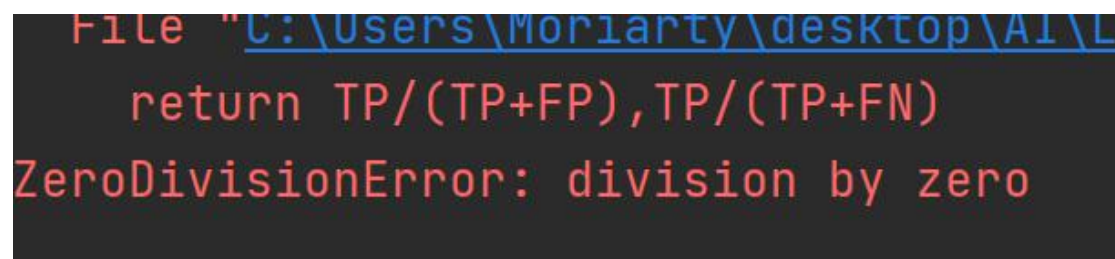
```

运行: SVM x
[2]
[3]
[2]]
Acc: 0.6449643947100712
0.750551876379691
0.5717948717948718
0.6575716234652115
macro-F1: 0.6599727905465914
micro-F1: 0.6449643947100712

进程已结束，退出代码为 0

```

问题分析：



```

File "C:\Users\Moriarty\Desktop\AI\L
return TP/(TP+FP), TP/(TP+FN)
ZeroDivisionError: division by zero

```

实验过程中速度很慢据目测是由于调用核函数引起的。最初代码编译总是报 A 的矩阵大小不匹配，经过推导，将 A 最终确定为 `(np.ones(row) * y, (1, row))` 的矩阵。实验过程中总是在进行一半的时候因为出现除以零而中断，此时的准确率只有 0.23 左右，经过改动参数，直接调用 self 中的数据最终实现完成 svm 且准确度正常。

## Part2 深度学习

### 手写感知机模型并进行反向传播

实验目的：

对矩阵链式求导的掌握

实验内容：

实现一个 4 层的感知机模型（隐层神经元设置为 5，4，4，3，即输入的特征尾为 5，输出的类别个数的 3，激活函数设置为 sigmoid）；实现 BP 算法；实现梯度下降算法。

sigmoid 函数：

$$\sigma(x) = \frac{1}{1 + \exp(-x)}.$$

导数为：

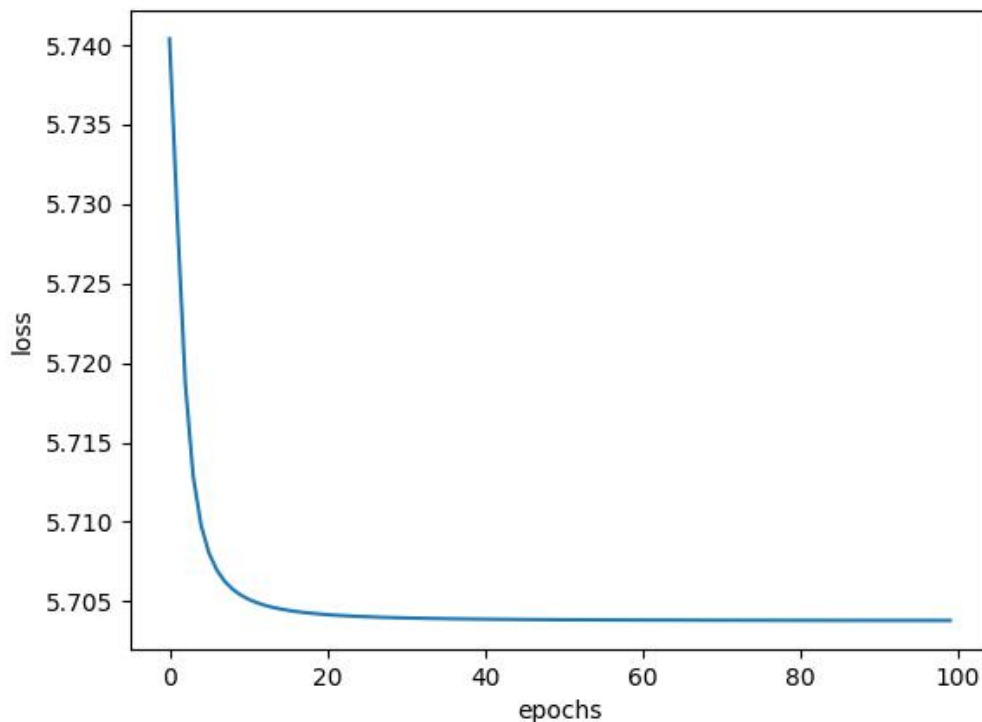
$$\sigma'(x) = \sigma(x)(1 - \sigma(x))$$

LogisticRegression 逻辑回归（softmax 回归）

我们要三层的 MLP，则只需要 HiddenLayer+LogisticRegression，

如果要四层的 MLP，则为 HiddenLayer+HiddenLayer+LogisticRegression.....以此类推。

LOSS 图像如下：



问题：

实验中遇到的问题主要集中在对模型的理解上以及最后的 **loss** 可视化中间，发现每次运行的时候 **loss** 的图像有变化，在少数情况下甚至出现曲线一直增长的情况，应该是由生成随机数的不确定导致的。

## 复现 MLP-Mixer

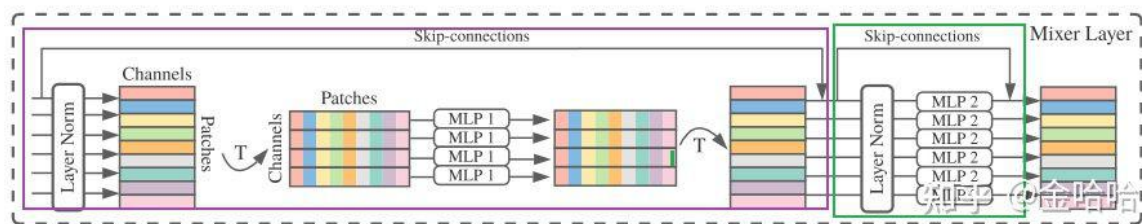
实验目的：

对深度学习的初步掌握，仅使用最基础的多层感知机。考察自行搜索相关资料学习的能力。

实验内容：

复现 MLP-Mixer 模型，并在 MNIST 数据集上进行测试。

MLP-Mixer 中，通过 Mixer Layer 使用 MLP 先后对列、行进行映射，实现空间域和通道域的信息融合。与传统卷积不同的是，Mixer Layer 将空间域和通道域分开操作，这种思想与 Xception 和 MobileNet 中的深度可分离卷积相似。



该实验根据 csdn 的思路跌跌撞撞完成函数代码最终还是未能成功测试数据集，但求给点分吧 orz。