**中山大学数据科学与计算机学院**

**移动信息工程专业-人工智能**

**本科生实验报告**

**（2017-2018学年秋季学期）**

课程名称：**Artificial Intelligence**

# 实验题目

1. **实现感知机学习算法**

# 实验内容

1. 算法原理

感知机学习算法理解起来比较简单，其就是二类分类的线性分类模型，简单说的就是输入一个测试数据，然后进行非对即错的判断。

对于感知机进行分类，通常都是使用{-1，+1}这两类，不过对于这个分类实际上不仅限于{-1，+1}这两类，这个类别完全可以根据自己的喜好定个离散的数值。实验中给出的数据就是这两类，所以其算法也是围绕这个展开。

对于一个测试数据输入哪一类可以表述为如下：

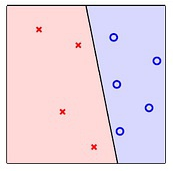
其中为测试数据，为权重向量，为维度，为阈值。这里面有一点那就是=的情况如何处理，一般是分为+1或者是直接忽视。

那么此时我们得到一个判断函数y

化简一下

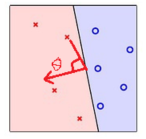
此时我们的测试数据

至此总结一下我们训练集的目标，是以利用已有的训练数据，得到一个可以尽量分开训练集的，然后其在测试集上也应该有很好的区分能力，所以训练的最终目标，在二维上可以如下图表示：



图来自：台湾国立大学《机器学习基石》

所以这个算法的实现的就要先基于对于这个w的拟合。

首先理解一下这个w，w是我们用来区分两个种类的直线的法向量。

那么用一个特征向量点乘w，其实就是在求这个这两个向量之间的余弦夹角，此时大于0的就是在同一侧，小于零的在另一侧。

然后就是拟合中出现错误判决时的修正w过程，使用如下公式：

|  |  |
| --- | --- |
|  |  |
| y=1,预测为-1的情况 | y=-1预测为+1的情况 |

其中的是特征向量*x*的真实类别也就是{-1,1}，为学习率，实验中没有提及默认是为1。对于上述两种情况这个公式都可以修正w。

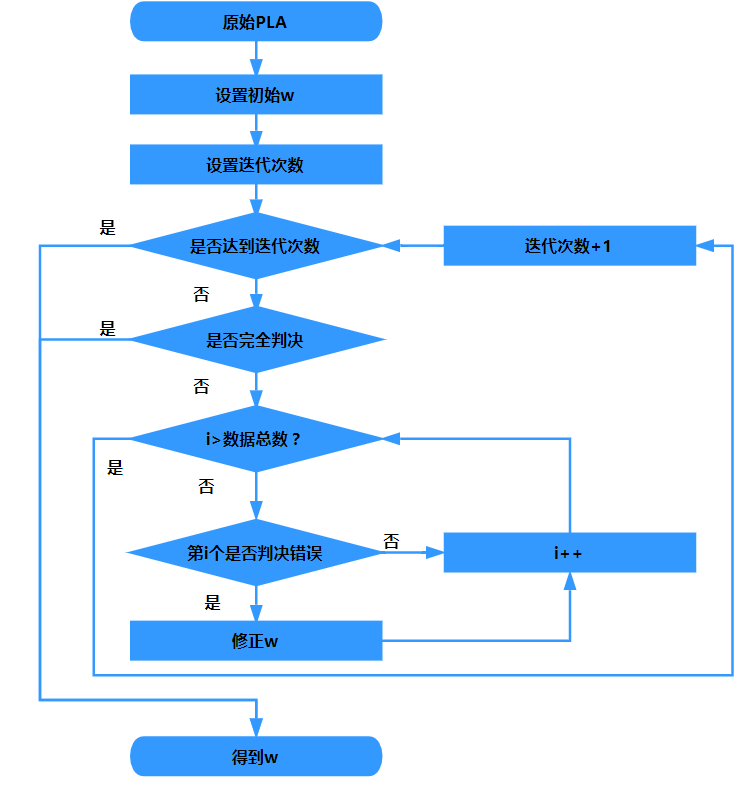
但是有种情况PLA不能迭代出正确的w，如果这个数据集不是线性可分的时候时候无论如何迭代都不会出现能够完全划分数据集的w。

最后总结一下算法的训练过程：通过训练集不断训练修正w，得到一个最优的w用于测试集的判决。我们算法实现的就是寻找这个最优w。

1. 伪代码&流程图
   1. 原始PLA

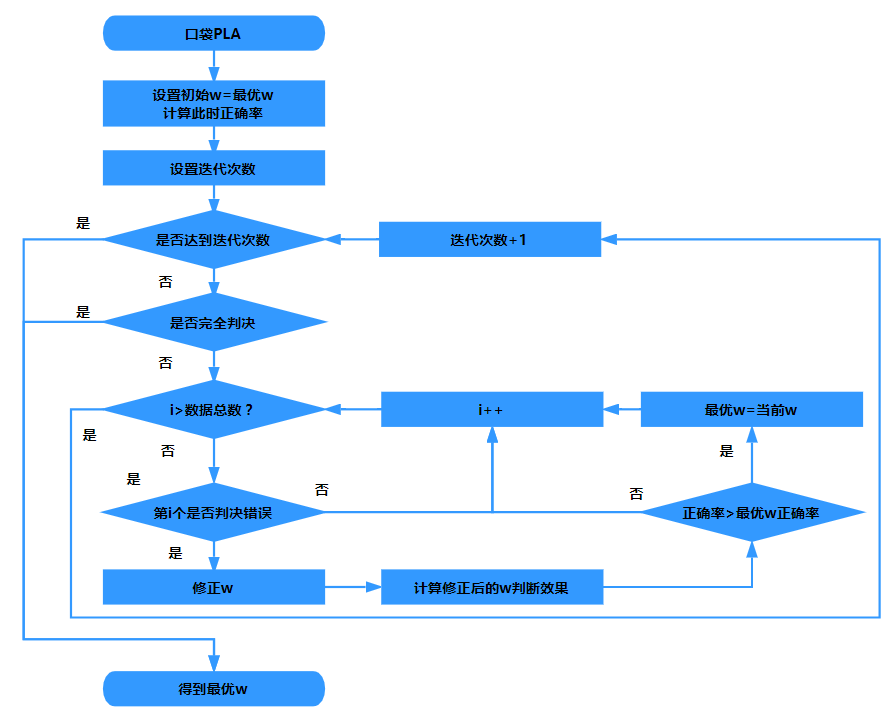
|  |
| --- |
| 设置初始w  **for** 迭代次数=0 to 设置迭代次数  **for** i=0 to 训练集数据数  **if** 出现符号判断错误 **then** 修正w  **else** i++ |

此处的最后w就是我们得到的w。当然此处应该加入如果完全分类直接退出迭代，虽然空着运行也行。



* 1. 口袋PLA

|  |
| --- |
| 初始化w=最优w  最优正确率=初始正确率  **for** 迭代次数=0 to 最大迭代次数  **for** i=0 to 训练集数据数  **if** 出现符号判断错误  **then** 修正w并计算正确率  **if** 正确率>最优正确率  **then** 最优正确率=当前正确率，最优w=当前w  **else** i++ |

当然此处无视完全分类直接退出迭代，当然也可以加。

1. 关键代码截图（带注释）

实验中使用了matlab作为编程语言，给出理由如下：这都是矩阵，向量计算，matlab有当前最快的矩阵算法，对于PLA这种能可能需要迭代的矩阵算法有很好的时间优化作用，在原始PLA中可能体现不出来，但是到了口袋PLA中O(n^2)的差距就体现出来了。

此处先使用python对csv处理了一下，把所有 ，分隔符换成了’\t’然后写入txt给matlab读取。

|  |
| --- |
| tmp\_train=[ones(train\_row,1) ,train(:,1:train\_col-1)];  sign\_train=train(:,train\_col); |

Matlab读入后处理成两个矩阵，一个是+1之后的训练集一个是训练集的符号。

* 1. 首先展示一下正确率计算部分

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 指标 | 说明 | 实际\*10+预测 | | S |
| TP | 本来为+1，预测为+1 | 1\*10 | 1 | 11 |
| FN | 本来为+1，预测为-1 | 1\*10 | -1 | 9 |
| TN | 本来为-1，预测为-1 | -1\*10 | -1 | -11 |
| FP | 本来为-1，预测为+1 | -1\*10 | 1 | -9 |
| S=sign(tmp\_val \* w') + 10 \* sign\_val; %原理见上表  TP = sum(S( : ) == 11);  FN = sum(S( : ) == 9);  TN = sum(S( : ) == -11);  FP = sum(S( : ) == -9);  Accuracy = (TP + TN) / (TP + FP + TN + FN);  Recall = TP / (TP + FN);  Precision = TP / (TP + FP);  F1 = (2 \* Precision \* Recall) / (Precision + Recall); | | | | |

解释一下我们直接通过一次转换使得不同指标得到不同的值，此时就可以直接用sum统计各个指标的数量了。

* 1. 原始PLA

这个没什么说的，因为使用matlab实际上跟伪代码差不多了，除了就是无视了完全划分之后跳出，因为完全划分之后多跑几次空循环好像也没什么问题，除了效率不高就是了。

|  |
| --- |
| **for** it = 1 : 1000 %设置迭代次数  **for** i = 1 : train\_row  **if**(sign(tmp\_train(i,:) \* w') ~= sign\_train(i,:))  % 跑出来的符号 != 正确符号 说明不匹配 调整直线  w=w + sign\_train( i , : ) \* tmp\_train(i , : );  %调整 w  **end**  **end**  **end** |

* 1. 口袋PLA

这个还是没什么还是跟着之前的伪代码差不多，同样无视了一下完全划分。

|  |
| --- |
| %初始化w  w=ones(1,val\_col);  max\_w = w;  %求当前w 和训练集中所有数据相乘 跑出来的符号 - 正确符号  flag=sign(tmp\_train \* w')-sign\_train;  %计算初始正确数量  cnt = sum( flag( : ) == 0);  max = cnt;  **for** it = 1 : 1000 %迭代  **for** i = 1 : train\_row  % 跑出来的符号 - 正确符号！=0 说明不匹配 调整直线  **if**(flag(i) ~= 0)  %调整 w  w=w + sign\_train( i , : ) \* tmp\_train(i , : );  %调整后的w 从新跑出来的符号 - 正确符号  flag = sign(tmp\_train \* w') - sign\_train;  %计算 跑出来的符号 == 正确符号 的个数  cnt = sum(flag( : ) == 0);  **if**(cnt > max) %贪心  max = cnt;  max\_w = w;  **end**  **end**  **end**  **end** |

1. 创新点&优化
2. 实验中使用F1指标代替了准确率作为测试指标主要考虑到了这个训练集中-1太多，而改用了F1，对于训练会有更好的效果。
3. 调整w的时候使用了随机梯度下降法，这样接近最优值得速度越快，但是可能不会最终迭代到最优值。(虽然不存在完全划分最优值)，随机梯度下降法更新w时公式如下：

以上公式应该是每维度的特征向量，但是我们此处二元分类，输出只有{-1，+1}所以可以直接合并。此处t为真实值，o为输出值。

在感知机训练法则中也有（t-o）但是经过符sgn(w\*x)计算的。而在随机梯度下降中，是直接使用 w\*x的结果。

感知机在线性可分时能够找到理想的权值正确分类所有样本，而随机梯度下降则收敛到误差最小，无论是否线性可分。

# 实验结果及分析

1. 实验结果展示示例

小数据测试，训练集如下

|  |  |  |
| --- | --- | --- |
| **x** | **y** | **label** |
| **0** | **2** | **1** |
| **2** | **2** | **-1** |
| **3** | **1** | **-1** |
| **-1** | **2** | **1** |
| **2** | **0** | **-1** |

初始w全1为{1,1,1}

第2个出现判断错误 此时w改为{0，-1,-1}

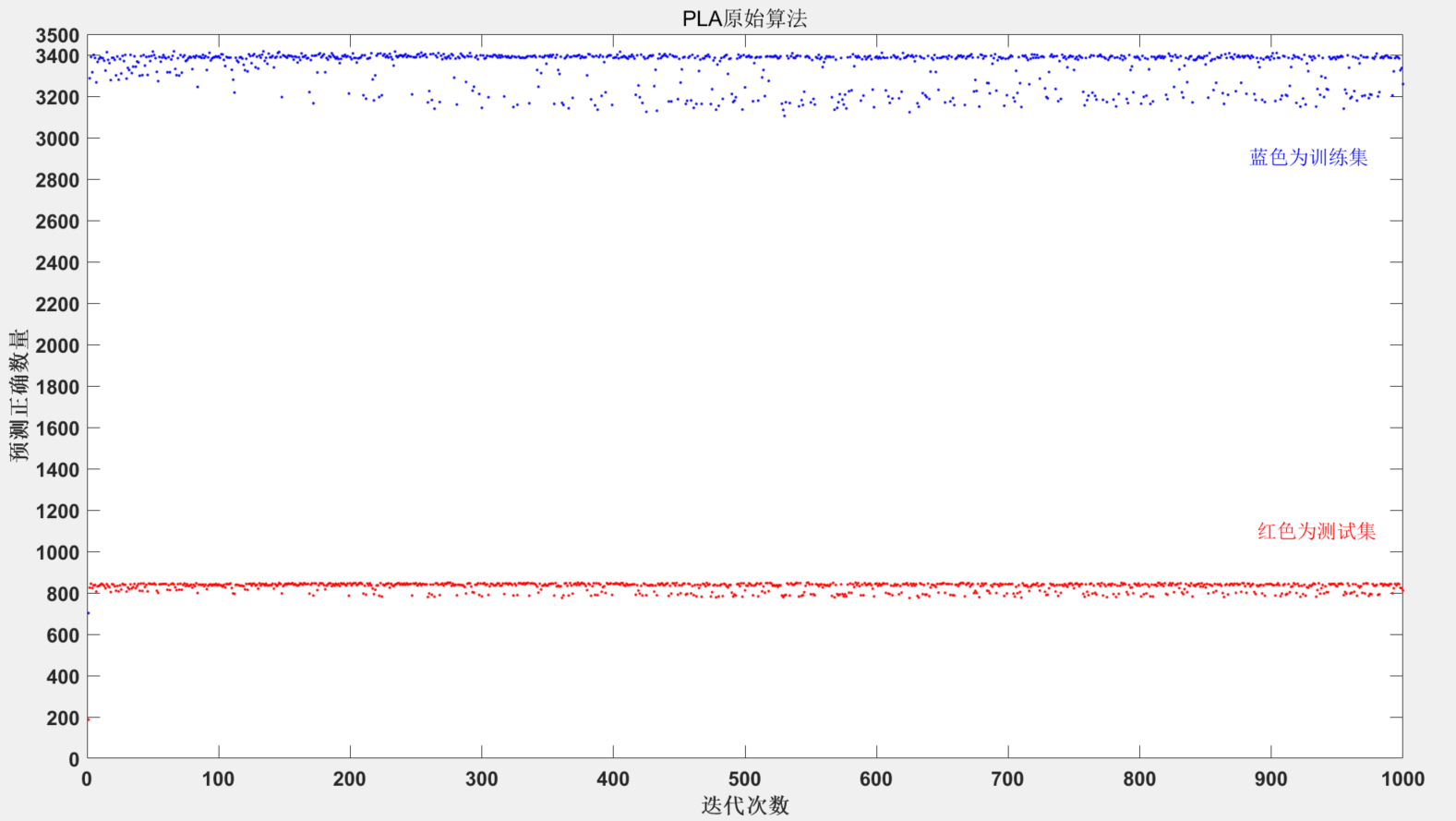
第4个出现判断错误 此时w改为{1，-2, 1 }

修改后全部判断正确

此时看到代码结果 和理论计算一致。

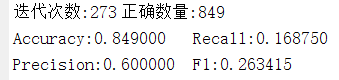
此处就不进行判断了…自己设的点没什么意义。

1. **评测指标展示即分析**
   1. **PLA原始算法w全为1**



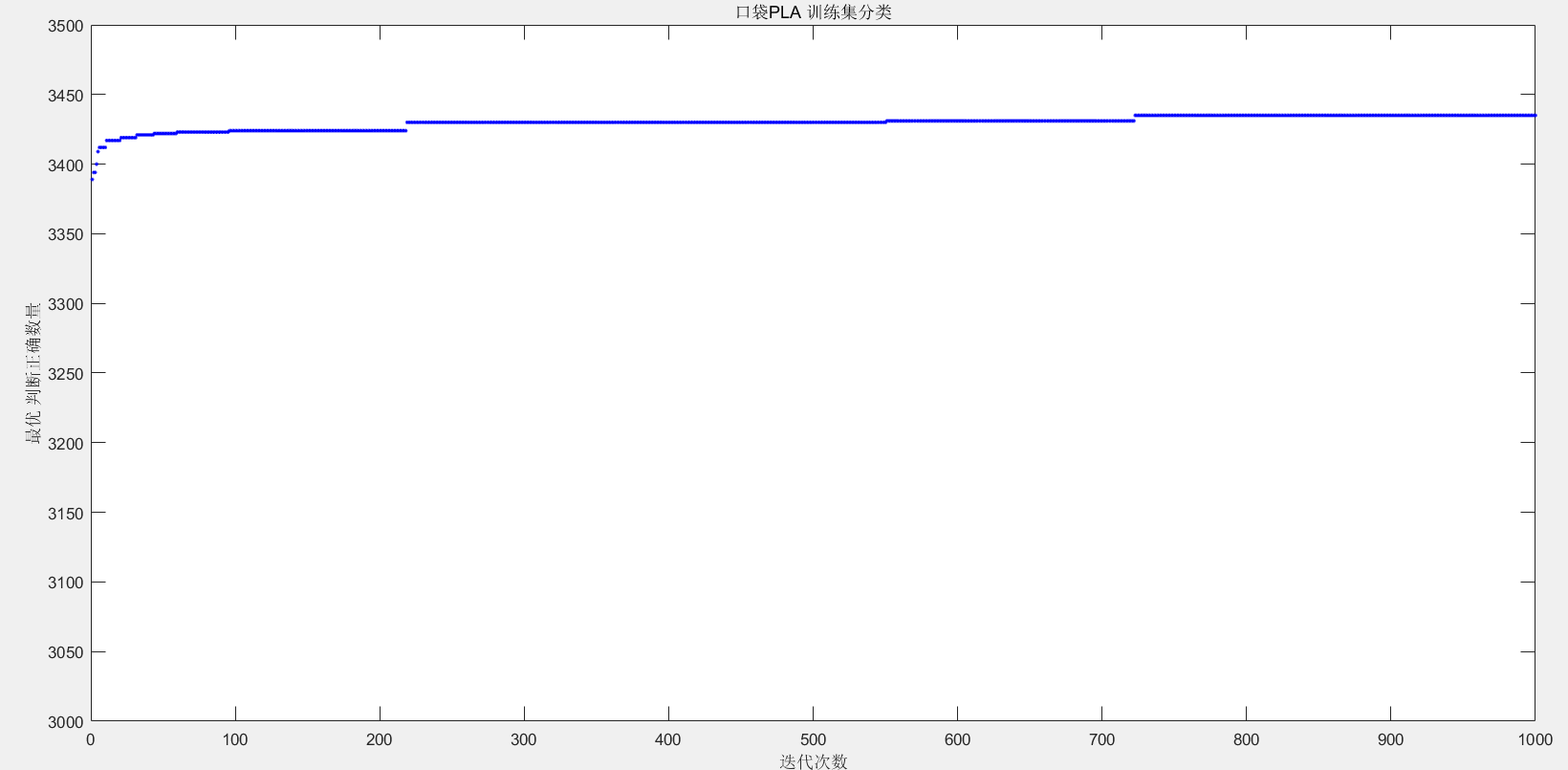
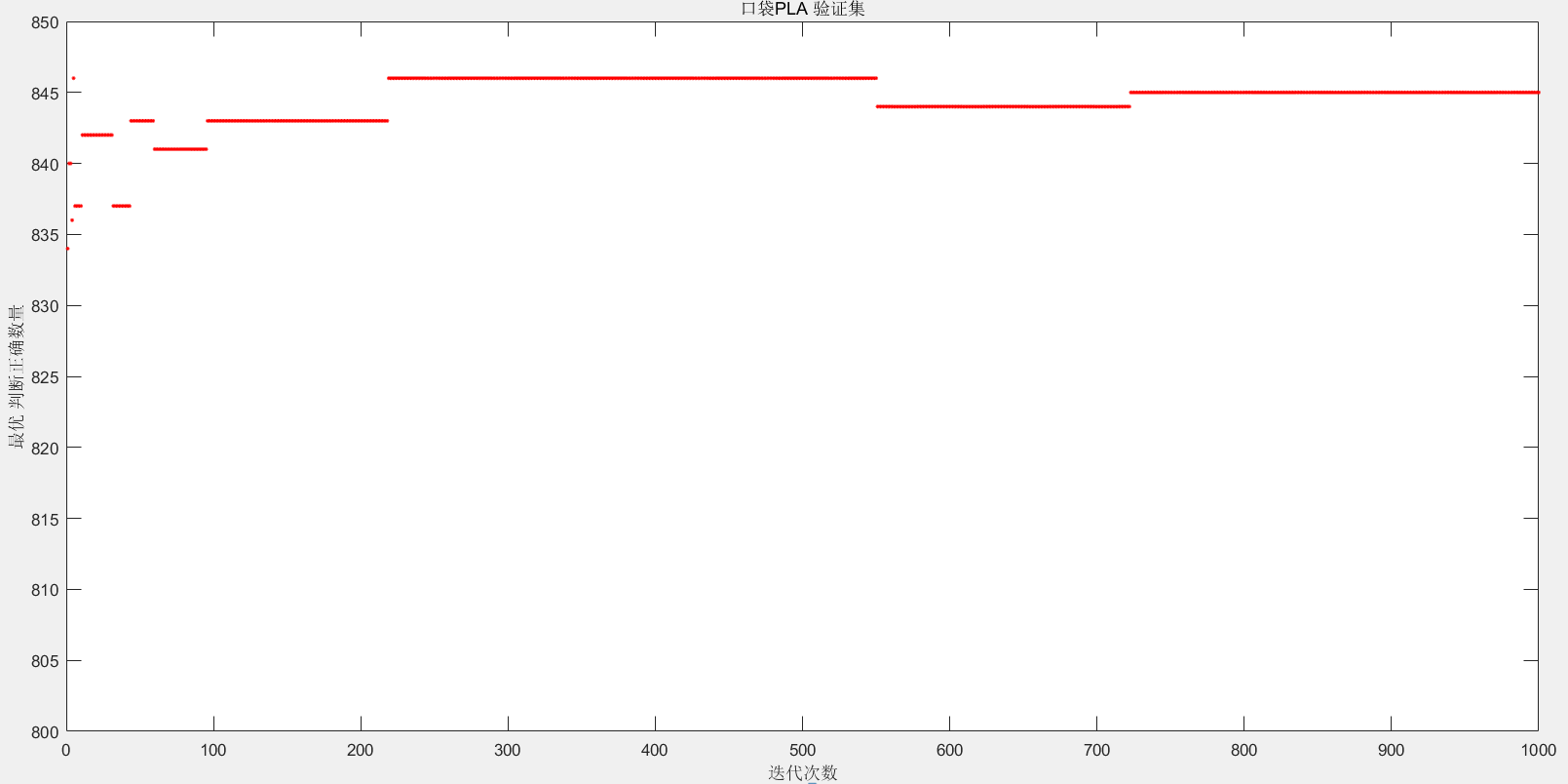
当我们迭代1000次来看，迭代1次过后这个准群率就进入到较为稳定的水平了，蓝线为这次迭代之后的w在训练集上的准确数量红线是这个w在测试集的准确数量。

在原始PLA算法中最好值：

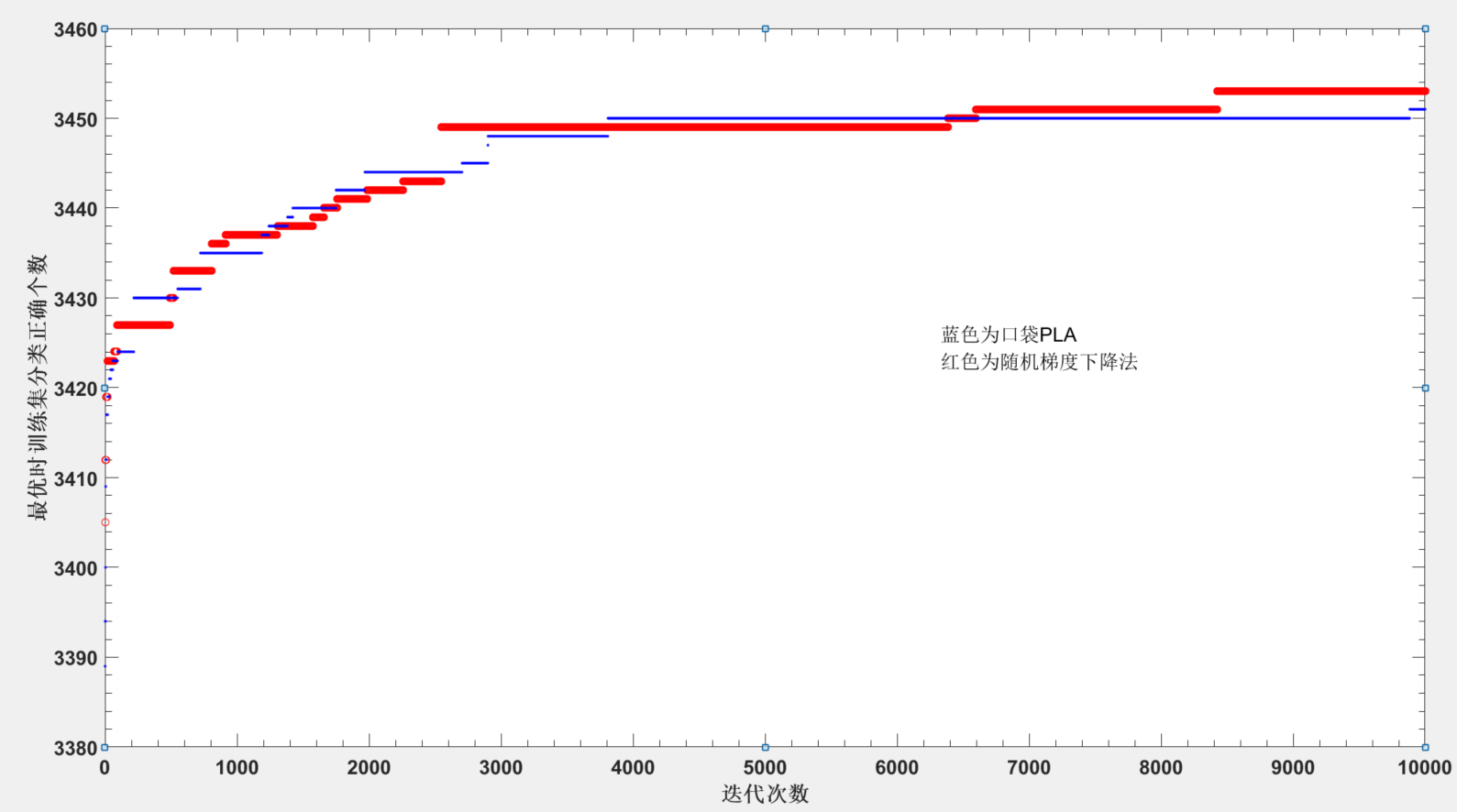


但是此处召回率过低，说很多正类都没有找到，按照综合评测指标来看并不是一个很好的结果。

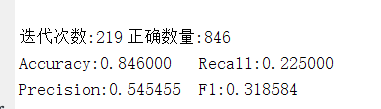
* 1. **PLA口袋算法w全为1**

 很明显我们可以看到实际上在训练集上划分效果好的，在训练集上效果不一定好，所以实际中还是要记录一下每一次迭代中最好的一次。用此w最为最终判断的w。

对比一下随机梯度下降法中的预测情况

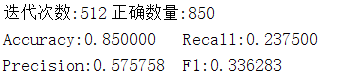
其实很明显我们可以看到随机梯度下降法对于加速达到一个更优解的速度更快。虽然也就高了那么一点没有什么太大的作用。

口袋PLA中最优结果如下：



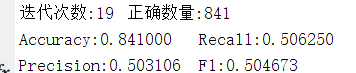
但是也有一个问题，他的召回率低也就是很多正类没有判决到。

然后看到随机梯度下降法，在没有使用F1作为判断参数的时候随机梯度下降法结果如图：



但是这里还是有个问题召回率偏低，说明还是有很多正类没有判决，所以需要引入F1参数。

所以我们最终使用梯度下降法和F1最为标准得到如下：



虽然此时准确率没有基础的口袋PLA高但是其判断了50%的正类，说明此时算法的比之前的更好。

# 思考题

* + - 1. **有什么其他的手段可以解决数据集非线性可分的问题？**

这个实际上可以通过提升数据维度实现，或者是找一种特殊的映射方法，举个例子两个同心的圆在2维平面上是不能线性可分的，如果映射到3维立体空间中，这两个圆可能就存在高度差就线性可分了。但是这个需要数据的特征，而且也不好实现，就算升到了无穷维也不一定能线性可分，只是能更好的分类而已。

或者我们可以把线性方法推广到曲线上。

不过都是玄学问题，目前无力解决。

* + - 1. **请查询相关资料，解释为什么要用这四种评测指标，各自的意义是什么？**

|  |  |  |
| --- | --- | --- |
|  | 相关(Retrieved),正类 | 无关(NonRelevant),负类 |
| 被检索到(Retrieved) | true positives正类判定为正，也就是+1判定为+1 | false positives负类判定为正类，也就是-1判定为+1 |
| 未被检索到(Not Retrieved) | false negatives正类判定为负类，也就是+1判定为-1 | true negatives，负类判定为负类，也就是-1判断为-1 |

准确率(Accuracy)其实就是判断正确的数量/总数。

精确率(precision)的公式是，它计算的是所有被被检索到的项中，应该被检索到的比例。也就是所有判决为正中应该被判决为正的比例。

召回率(recall)的公式是，它计算的是所有检索到的项占所有应该检索到的项的比例。也就是正类判定为正占所有正类的比例

F1值就是精确值和召回率的调和均值，当P和R指标出现矛盾情况，这样就需要综合考虑他们，公式是：

我们所用的F1就是的公式。

首先这个准确率(Accuracy)实际上就是常用的我们判决正确的数量，日常使用中我们第一个想到的就是这个。

精确率(precision)这个指标主要就是在衡量我们判断为正的中有多少是判断正确的，他能体现我们判决结果的准确情况，换个词叫做查准率。

召回率(recall)找回了就是我们应该判断为正中，实际判断为正的比例，他能体现我们漏判的情况，换个词查全率。

P、R这两个指标在使用中根据不同的情况占有不同的比重，有的情况要的是精确率、有的要的是召回率。

本实验中，P，R指标需要综合考虑，所以就会用到综合评价指标(F-Measure)，使用F1指标可以很好的权衡P、R。在实验中可以起到替代准确率指标的作用。