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**

教学班级	**15M1**	专业（方向）	**软件工程（移动信息工程）**
学号	**15352033**	姓名	**陈黄胤**

一、 实验题目

Lab3：感知机学习算法 Perceptron Learning Algorithm

二、 实验内容

1. 算法原理

PLA 可以用于解决线性可分问题，最终将问题分成两类（1 或-1）

Ⅰ、传统 PLA 算法

Step1:读入训练集后，对每一个样本前加一个常数 1，初始化权重向量 W 为 1，权重向量长度等于训练集样本长度。

Step2:遍历训练集的所有样本，判断 sign(W*X)的值与标签是否一致，如果一致的话遍历下一个样本，如果不一致的话就更新 W=W+X*Y。

Step3:重复 Step2 直到次数达到某一个阈值次数。得到训练好的权重向量 W。

Step4:读取测试(验证)集数据，与 W 相乘求和，得到 sign(W*X)作为预测标签。

Ⅱ、口袋 PLA 算法

Step1：与传统 PLA 算法一致，读入训练集后，对每一个样本前加一个常数 1，初始化权重向量 W 为 1，权重向量长度等于训练集样本长度。

Step2：建立一个临时的局部权重向量 Wt，并先初始化 Wt 使得 Wt 与 W 一致。

Step3：遍历训练集的所有样本，判断 sign(Wt*X)的值与标签是否一致，如果一致的话遍历下一个样本，如果不一致的话就求解当前 Wt 下训练集的正确率，并保存这个正确率，如果正确率比最大正确率要大，那么就更新这个正确率，并把 Wt 保存到 W 中。重复 Step3 若干次。

Step4:读取测试(验证)集数据，与 W 相乘求和，得到 sign(W*X)作为预测标签。

2. 伪代码

```
Function Tradition_PLA()
{
    For i=0 -> i=iterations
        For j=0 -> j=row.size()
            if (sign(X*W)!=labels[j]
                W=W+X*W;
```

```
    Return W
}

Function Pocket_PLA()
{
    For i=0 -> i=iterations
        For j=0 -> j=row.size()
            if (sign(X*W)!=labels[j]
                W=W+X*W;
            If(Run(W) > Run(W_best)
            W_best = W;
    Return W_best
}
```

3. 关键代码截图（带注释）

```
60  void initial_algro()
61  {
62      int limit=8000;
63      for(int i=0;i<limit;i++)
64      {
65          bool best_fit = false;    //如果全部都预测正确
66          for(int j=0;j<train_set.size();j++)
67          {
68              double sum=0;
69              for(int k=0;k<train_set[j].size();k++)
70              {
71                  sum+=initial_weight[k]*train_set[j][k];  //训练集样本乘以权重向量
72              }
73              if(sign(sum)!=labels[j])  //与标签不符合
74              {
75                  for(int k=0;k<train_set[j].size();k++)
76                  {
77                      initial_weight[k]+=(double)labels[j]*train_set[j][k]; //W=W+X*Y
78                  }
79                  break;
80              }
81              else if(j==train_set.size()-1)
82              best_fit = true;
83          }
84          if(best_fit) break;
85      }
86  }
```

```
111 void pocket_algro()
112 {
113     int limit=8000;
114     vector<double> tmp_weight (pocket_weight);
115 
116     double cur=0;
117     for(int i=0;i<limit;i++)  //迭代8k次
118     {
119         for(int j=0;j<train_set.size();j++)
120         {
121             double sum=0;
122             for(int k=0;k<train_set[j].size();k++)
123             {
124                 sum+=tmp_weight[k]*train_set[j][k];  //训练集样本乘以权重向量
125             }
126             if(sign(sum)!=labels[j])
127             {
128                 for(int k=0;k<train_set[j].size();k++)
129                 {
130                     tmp_weight[k]+=(double)labels[j]*train_set[j][k]; //Wt=Wt+X*Y
131                 }
132                 if(pocket_count(tmp_weight) > cur)  //Wt的效果高于W_best
133                 {
134                     pocket_weight.assign(tmp_weight.begin(),tmp_weight.end()); //W_bset = Wt
135 
136                     cur = pocket_count(pocket_weight);   //最佳效果更新
137                 }
138 
139                 break;
140             }
141         }
142     }
143 }
```

4. 创新点&优化（如果有）

无

三、 实验结果及分析

1. 实验结果展示示例（使用小数据集）

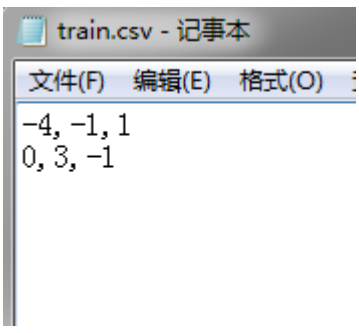
采用 ppt 上的小数据集作为对算法的检验：

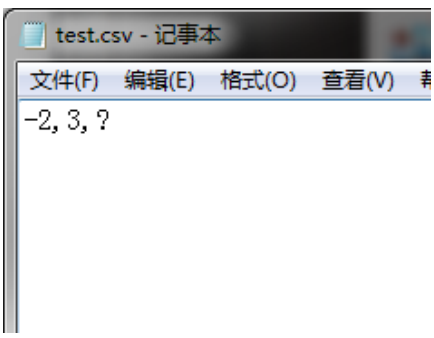
即样本一： 4 -1 标签+1

样本二： 0 3 标签-1

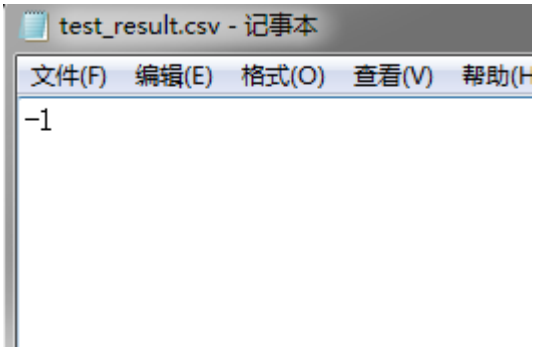
求样本三：-2 3 标签？

将数据输入程序：





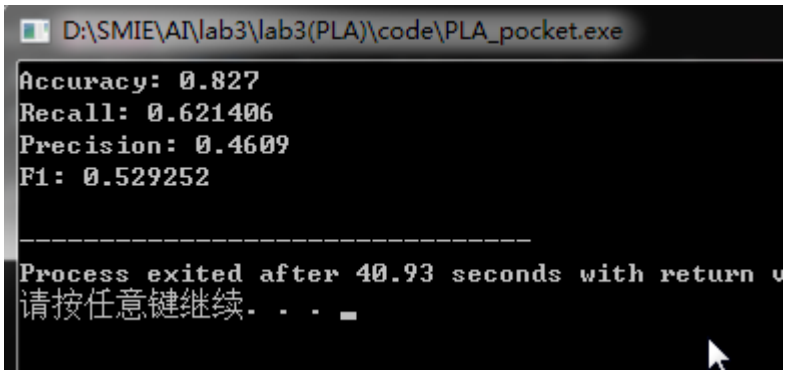
得到结果：



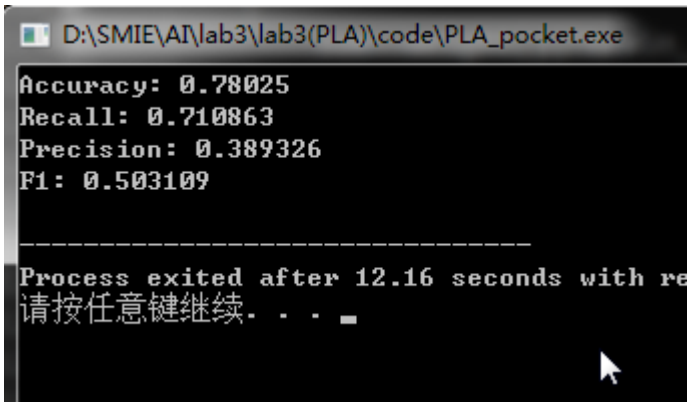
显然与答案一致，说明程序在这一测试集中预测正确。

2. 评测指标展示即分析（如果实验题目有特殊要求，否则使用准确率）

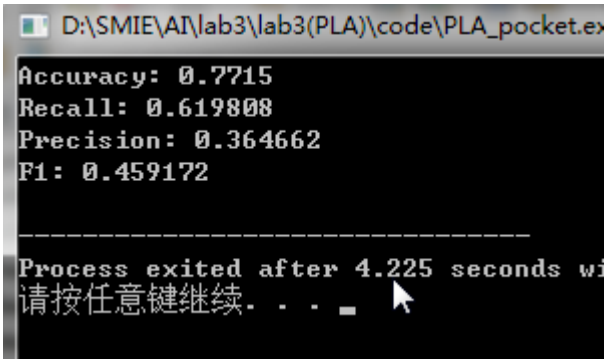
迭代 8k 次的 pocket 算法（以 F1 作为指标）



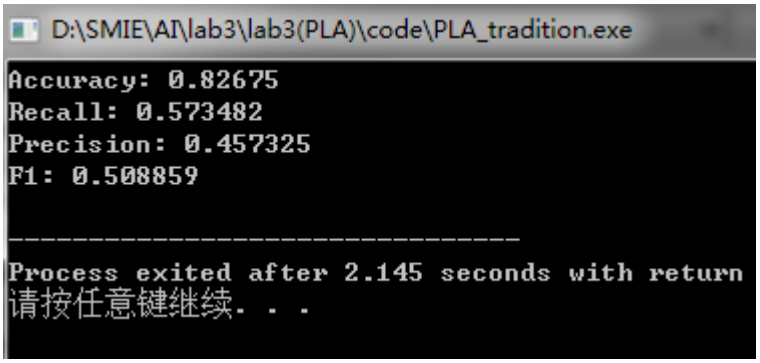
迭代 2k 次的 pocket 算法



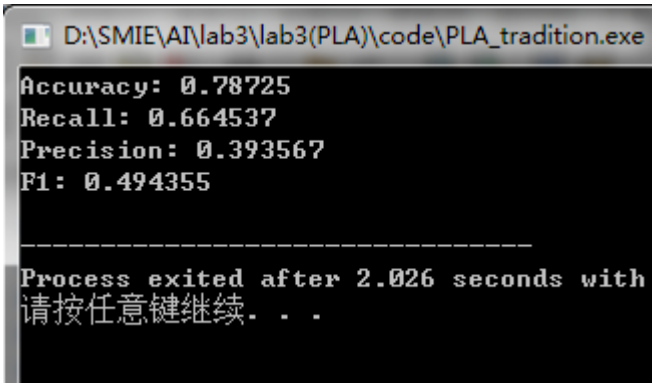
迭代 500 次的 pocket 算法



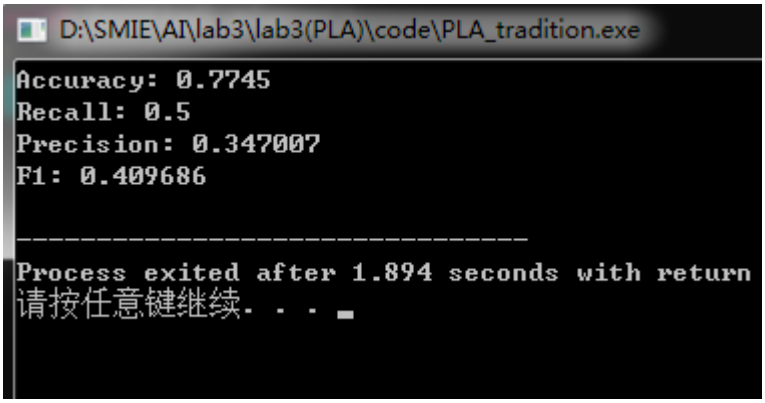
迭代 8k 次的传统算法



迭代 2k 次的传统算法



迭代 500 次的传统算法



可以发现权重在训练集上的迭代次数越多会有更好的效果。在相同迭代次数下，可以发现口袋算法在同一验证集上的表现比传统算法要更加优秀，评测指标都要比传统算法更高。

四、 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？

可以采用多个 PLA 一起执行，然后再对多个 PLA 得到的权重再次进行加权取优，其做法类似于一层的神经网络。也可以把数据高维化，把相关特征先细分处理，然后再使用 PLA。

2. 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。

①准确率：准确率可以很直接地对分类器进行评价，只需判断正确率即可。但是正确率高只能代表在这一个验证集上表现得好。如果验证集数据不够全面，凑巧数据集的范围都属于分类器刚刚好能够分类正确的地方，这样的话正确率就缺乏代表性了，正确率极高但是并不能说明分类器的准确性。

②精确度：即分类器分类出正样本的能力。在某些分类器中，正样本极少的话，如果精确度足够高，就能在这类的分类中找出绝大多数的所需正样本，而不会错找太多负样本。

③召回率：即对于正结果的找全率。在一些不希望损失太多正样本的分类需求中，高召回率的分类器会有很出色的表现。

④F：调和平均率可以综合精确度和召回率，当两者都有较好表现的时候才能有较好的调和平均律