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	**m2**	专业（方向）	**软件工程（移动工程信息）**
学号	**15352446**	姓名	**钟展辉**

一、 实验题目

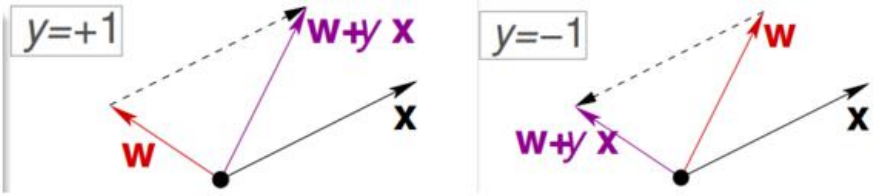
实验三——感知机学习算法

二、 实验内容

1. 算法原理

PLA 算法用于解决二维线性可分问题，二维是指评测指标和结果可理解为正实例或负实例，而休闲型可分是指可以找到一条线把正实例点与负实例点分开成两个集合。

PLA 算法的基本思想是，最开始的时候随便设置一条直线向量 w(n)，这个向量应用到已知的训练数据集中，那么会有错误的划分(x(n),y(n)),这时候我们去纠正我们的向量 w(n+1)=w(n)+y(n)*x(n)，将能达到使向量像该 x(n)向量靠近或远离，能更好的划分正负实例点的效果。



如上图所示，当 y 为正数，而 w 应用到 x 向量上出现错误时，说明实例 x 不在 w 分割线正实例的一边，因此在纠正 w 后，w 会向实例 x 靠近，试图将该实例容纳在 w 分割线正实例的一边。

当 y 为负数，而 w 应用到 x 向量上出现错误时，说明实例 x 在 w 分割线正实例的一边，因此在纠正 w 后，w 会远离实例 x，试图将该实例排除出 w 分割线正实例的一边。

最后得到的最佳向量 w，把它应用在测试集中，将能达到很好的预测实例的正负效果。

而算法的大概思路、步骤为：给每个 x 实例样本向量前加常数项 1-->初始化 w-->遍历所有样本并检测样本标签与预其测结果是否相符-->不相符则纠正 w 并从再次头开始遍历所有样本直至样本全部遍历完毕。

还有一个问题是，PLA 算法什么时候停止，按理想情况来考虑，训练集数据是线性可分的且数据量较小，则可以循环运算直至判断到索引已经到达最后一个样本（说明 w 已经能

将所有样本实例预测正确）。但是实际情况下，数据量都是非常大而且一般无法找到一条能将所有正负实例都划分开的直线。所以需要添加一个循环次数控制，以免程序进入死循环，还可以将修改一下步骤方法，每次遇到错误纠正 w 后不从头开始，而是继续向下面的样本运行下去，这样可以避免程序因为在前面少数的样本中就陷入循环而没有使用后面多数样本的问题。

上面是原始的 PLA 算法步骤，对于那些有杂质的数据来说，要做到线性可分是非常困难的，那么如何找到一条直线，它是所有可能直线当中犯错误最少的？这就用到口袋 PLA 算法，每次 b 纠正 w 后都与口袋中的最优 w 相比较，如果当前 w 能更准确的预测样本，则将其替换进口袋中，如此算法结束后，口袋中的就是最优 w 了。

2. 伪代码

原始 PLA 算法代码：

```
int main(){
    getnumber1();//从 csv 文件中读取数据进数组 x 和 ylabel
    初始化 w 为{1,1,1.....1};
    while(time<10)//time 为限定循环次数
    {
        for 循环遍历所有训练集样本
        {
            bool flag=Calculate(0,i);//计算 w(n)*x(n)，结果为自然数则返回 true，否则返回 false ;第一个参数 0 训练集 1 验证集 2 测试集
            if((flag=0&&label=1) or (flag=1&&label= -1))//预测不准确
            {
                UpdateWeight(i);//w=w+y(n)*x(n)
            }
        }
    }
    Use_val();
    Use_test();
}
Use_val()
{
    getnumber2(1);//从验证集读取数据进二维数组 xx 和一维数组 yylabel
    Prepare(1);//计算四个指标 TP FP TN FN
    double cur_f=F1(),cur_ac=Accuracy(),cur_re=Recall(),cur_pc=Precision();// 计算召回率 准确率 精确率 F 值，并输出
}
Use_test()
{
```

getnumber2(2);//从测试集读取数据进二维数组 xx 和一维数组 yylabel

ofstream 打开文件流

for 循环遍历测试集样本

{

if(Calculate(i)==1)//即计算 w*x>=0

{输出 1 到文件中}

else {输出-1 到文件中}

}

}

PLA 口袋算法代码：

代码基本与 PLA 原始算法一样，不同之处在于添加了变量

double w_pocket[100];//口袋，装着一个最优 w

double best_f,best_ac,best_pc,best_re;//口袋里那个最优 w 的四个评测指标

每次更新 w 时都将其与口袋中的 w 比较，以 f 值为指标选出最优的加进口袋中，如此算法结束时口袋里的 w 就是最优的权值向量了，再拿它去验证集测试准确率或者拿去测试集预测 label。

3. 关键代码截图（带注释）

原始 PLA 算法：

所有变量及函数：

```
14  double w[100];//正在用的向量
15  double x[5000][100];//训练集可容纳5000个x样本
16  double xx[1005][100];//验证集 or测试集
17  int ylabel[5000];//每个样本的标签（正负）
18  int yylabel[1005];//验证集 or测试集的标签
19  double TP=0,FN=0,TN=0,FP=0;//四个评测指标
20  int Length;//w、x向量的长度
21  int textcnt;//样本数
22  int new_textcnt;//验证集 or测试集的样本数
23
24  void getnumber1();//从csv文件中读取数据进数组x和ylabel 、xx和yylabel
25  bool Calculate(int which_set,int text_index);//计算w(n)*x(n)，结果为自然数则返回true，否则返回false ，
26  void UpdateWeight(int text_index);//w=w+y(n)*x(n)
27  void Prepare(int which_set);//计算四个评测指标   //0 训练集 1 验证集  2测试集
28  double Accuracy();//计算准确率，
29  double Recall();//计算召回率
30  double Precision();//计算精确率
31  double F1();//计算F1值
32  void Use_val();//使用验证集，验证模型准确率
33  void getnumber2(int v_or_t);//0 训练集 1 验证集  2测试集
34  void Use_test();//使用测试集预测，将结果输出到文件
```

main 函数中，利用训练集训练 w 向量

```
int main()
{
    getnumber1();//从csv文件中读取数据进数组x和ylabel 、xx和yylabel
    for(int i=0;i<Length;i++) w[i]=1;//初始化w为{1,1,1.....1};
    int time=0;//遍历样本集次数， 限制算法运行时间
    while(time<20)
    {
        int i;
        for(i=0;i<textcnt;i++)//遍历训练集
        {
            bool flag = Calculate(0,i);
            if((flag&&ylabel[i]==1) || (!flag&&ylabel[i]==-1));//正确，则不更新w
            else UpdateWeight(i);//纠正w
        }
        time++;
    }
```

然后再选择验证集或测试集进行预测：

```
//Use_val();
Use_test();
```

```
void Use_val()
{
    getnumber2(1);//从验证集csv文件中读取数据进数组xx和yylabel
    Prepare(1);//计算四个指标TP FP TN FN
    double cur_f=F1(),cur_ac=Accuracy(),cur_re=Recall(),cur_pc=Precision();
    cout<<"Accuracy_in_val_set="<<cur_ac<<endl;
    cout<<"Recall_in_val_set="<<cur_re<<endl;
    cout<<"Precision_in_val_set="<<cur_pc<<endl;
    cout<<"F1_in_val_set="<<cur_f<<endl;
}
```

```
void Use_test()
{
    ofstream fout;
    fout.open("15352446_zhongzhanhui_PLA.csv");
    getnumber2(2);
    for(int i=0;i<new_textcnt;i++)
    {
        bool flag2=Calculate(1,i);
        if(flag2==1) fout<<1<<endl;
        else fout<<-1<<endl;
    }
    fout.close();
}
```

计算评测指标的函数 Prepare();Accuracy();Recall();Precision();F1();的实现很简单，不展示。

PLA 口袋算法：

在原始 PLA 算法的基础上加上了以下代码：

```
for(i=0;i<textcnt;i++)
{
    bool flag = Calculate(0,i);
    if((flag&&ylabel[i]>0) || (!flag&&ylabel[i]<0));
    else
    {
        UpdateWeight(i);
        Prepare(0);
        double cur_f=F1(),cur_ac=Accuracy(),cur_re=Recall(),cur_pc=Precision();
        if(cur_f>best_f)//当前w更优，将其填进口袋中 ， 且更新变量 best_f....
        {
            for(int i=0;i<Length;i++)
            {
                w_pocket[i]=w[i];
            }
            best_f=cur_f;
            best_ac=cur_ac;
            best_re=cur_re;
            best_pc=cur_pc;
        }
    }
}
```

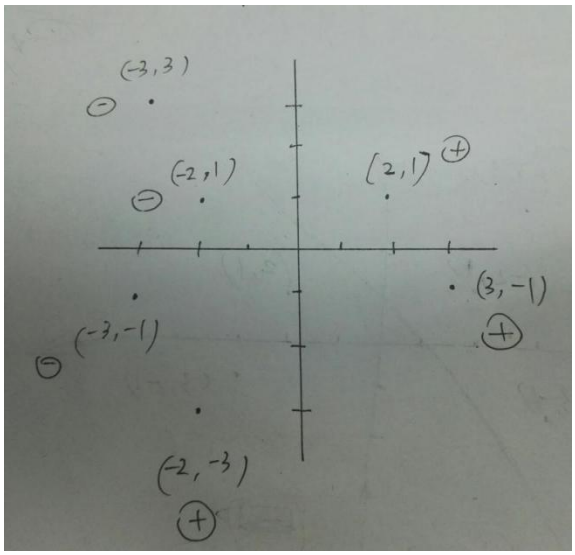
三、 实验结果及分析

1. 实验结果展示示例（可图可表可文字，尽量可视化）

一下是 PLA 原始算法，且每次遇到错误不是从头重新开始，而是继续遍历剩余实例：

以下是我自己设计的小数据集，共 6 个点，线性可分，尝试用 PLA 原始算法找出能将正负实例点分割开的直线（即找出权值 w 向量），每次需要更新 w 向量时，都输出样例的索引位置以及更新后的 w 向量；

```
1  -3,3,-1
2  -2,1,-1
3  -3,-1,-1
4  2,1,1
5  3,-1,1
6  -2,-3,1
7
```



分割线是 w 向量的法向量所在直线，而初始 w0 设置为(1,1,1)，其中第一项是截距，所以对应分割线为 y=-x+1；即下图中直线 w0

第一个样例，x0=(1,-3,3)；y0= -1； w0*x0=1 >0; 错误 w1=w0+(-1)*x0=(0,4,-2); 此时分割线为 y=2x,即下图中的直线 w1，可见仍没有把所有正负实例分开。

第二个样例，我选择继续选下一个点运算下去（不是从头开始），因此选用下一个点 x1=(1,-2,1); y1= -1; w1*x1= -10<0 正确，w 不需要更新

第三个样例，x2=(1,-3,-1); y2= -1 ; w1*x2=-10<0,正确，不需要更新；

第四第五个样例均正确，w 不需要更新；

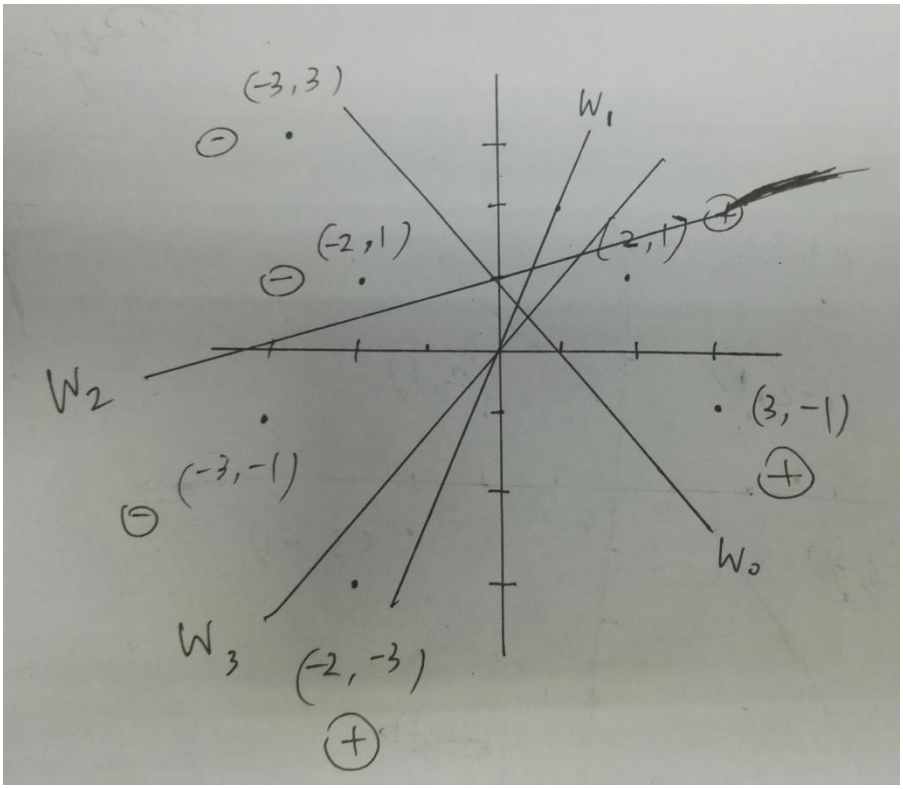
第六个样例，x5=(1,-2,-3) ; y5=1 ; w1*x5=-2<0 ,错误，更新 w2=w1+x5=(1,2,-5);对应直线 y=0.4x+1,如下图所示的直线 w2

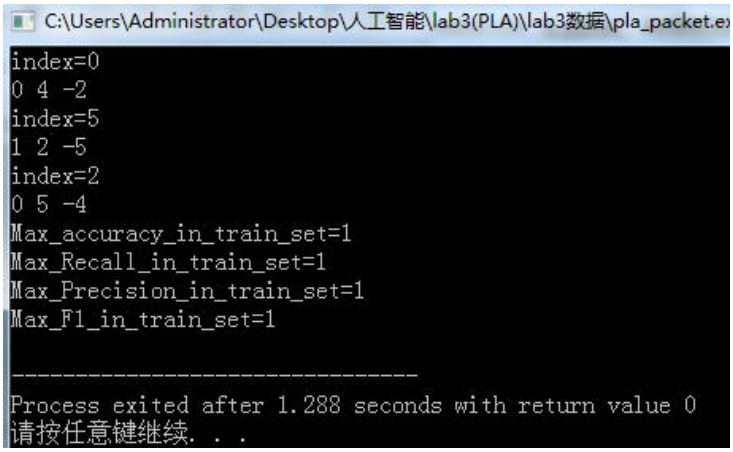
从头开始再次遍历所有样例：

第一个样例。。。

第三个样例，x2=(1,-3,-1); y2=-1 ;w2*x2=0 错误，更新 w3=w2-x2=(0,5,-4),对应直线 y=1.25x；如下图的直线 w3；

至此直线 w3 已经能完美划分正负实例点，w 不再更新；

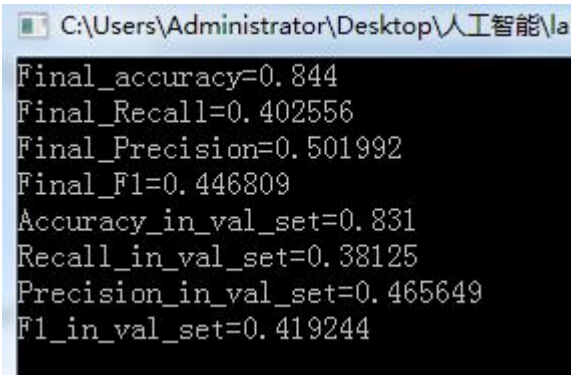




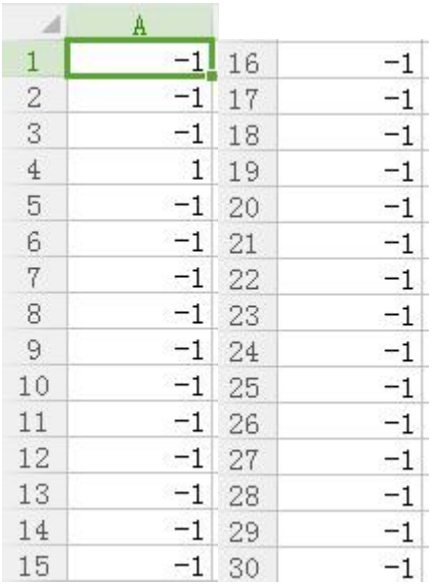
代码运行结果与上述一致.

2. 评测指标展示及分析（如果实验题目有特殊要求，否则使用准确率）

PLA 原始算法，20 次从头到尾遍历训练集样本，前四个数值是最终得到的评测指标，下面四个数值是将这个 w 应用到验证集后得到的四个评测指标。

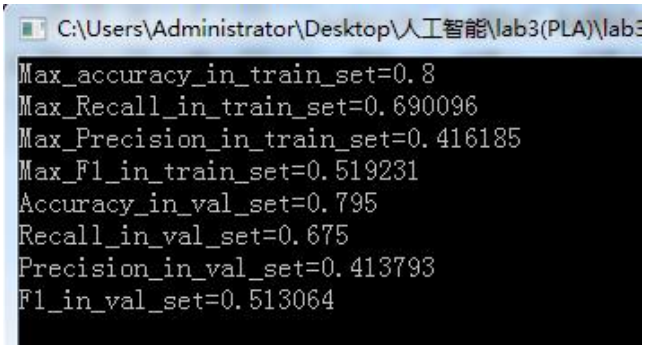


w 应用到测试集，预测的 label(前 28 个)



	A		
1	-1	16	-1
2	-1	17	-1
3	-1	18	-1
4	1	19	-1
5	-1	20	-1
6	-1	21	-1
7	-1	22	-1
8	-1	23	-1
9	-1	24	-1
10	-1	25	-1
11	-1	26	-1
12	-1	27	-1
13	-1	28	-1
14	-1	29	-1
15	-1	30	-1

口袋 PLA 算法，10 次从头到尾遍历训练集样本，前四个数值是用口袋中的最优 w（以更高 f 值为标准）计算出的评测指标，下面四个数值是将这个 w 应用到验证集后得到的四个评测指标。



将这个最优 w 应用到测试集，预测的 label(前 30 个)

	A		
1	-1	16	-1
2	-1	17	-1
3	1	18	1
4	1	19	-1
5	-1	20	-1
6	1	21	-1
7	1	22	-1
8	-1	23	1
9	-1	24	-1
10	1	25	-1
11	-1	26	-1
12	-1	27	1
13	1	28	-1
14	-1	29	-1
15	-1	30	-1

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？

①设置一个数组，每个训练集样例占用一个数组元素，记录 w 在每个样本更新的次数，如果发现在有的样本处更新次数过多，则很有可能这个点是杂质，比如是处于一大堆正实例点中间的一个负实例点，则将此样例去除，不在遍历它。

②每次遍历的时候遇到错误预测，此时更新 w，然后继续向下遍历剩余点，而不是从头开始再遍历，这样可以避免因为前面的样例非线性可分而陷入死循环中而没有遍历到下面的样例。因此也能更大限度的利用整个数据集的校正来降低几个杂质点的影响。

2、请查询相关资料，解释为什么要用召回率 准确率 精确率 F 值这四种评测指标，各自的意义是什么。

$$\text{Accuracy}=\frac{TP+TN}{TP+FP+TN+FN} \qquad \text{Recall}=\frac{TP}{TP+FN}$$

$$\text{Precision}=\frac{TP}{TP+FP} \qquad F1=\frac{2*\text{Precision}*\text{Recall}}{\text{Precision}*\text{Recall}}$$

准确率是你的全部判断有多准确，无论预测是正值还是负值，准确率有多少。

精确率是针对我们预测结果而言的，它表示的是预测为正的样本中有多少是真正的正样本。那么预测为正就有两种可能了，一种就是把正类预测为正类(TP)，另一种就是把负类预测为正类(FP)。意义是代表了你的预测有多少是对的

而召回率是针对我们原来的样本而言的，它表示的是样本中的正例有多少被预测正确了。那也有两种可能，一种是把原来的正类预测成正类(TP)，另一种就是把原来的正类预测为负类(FN)。意义是代表了正例里你的预测覆盖了多少

精确率与召回率其实就是分母不同，一个分母是预测为正的样本数，另一个是原来样本中所有的正样本数。

另外，精确率+误判率=1，召回率+漏判率=1；

F 值为精确率和召回率的调和平均值。相当于精确率和召回率的综合评价指标。

|---------- 如有优化，重复 1，2 步的展示，分析优化后结果 ----------|

PS：可以自己设计报告模板，但是内容必须包括上述的几个部分，不需要写实验感想