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	**1515**	专业（方向）	**软件工程**
学号	**15352334**	姓名	**吴佳卫**

一、 实验题目

感知机学习算法-PLA

二、 实验内容

1. 算法原理

从整体上来说，PLA 是一种二分类的分类算法模型。及其本原理是很简单的：用一个向量 w 划分所有的点，根据向量相乘结果划分为 1 或-1，然后不断的迭代寻找被划分错误的点。根据这个被划分错误点重新修正向量 w，基本算法是梯度下降法。

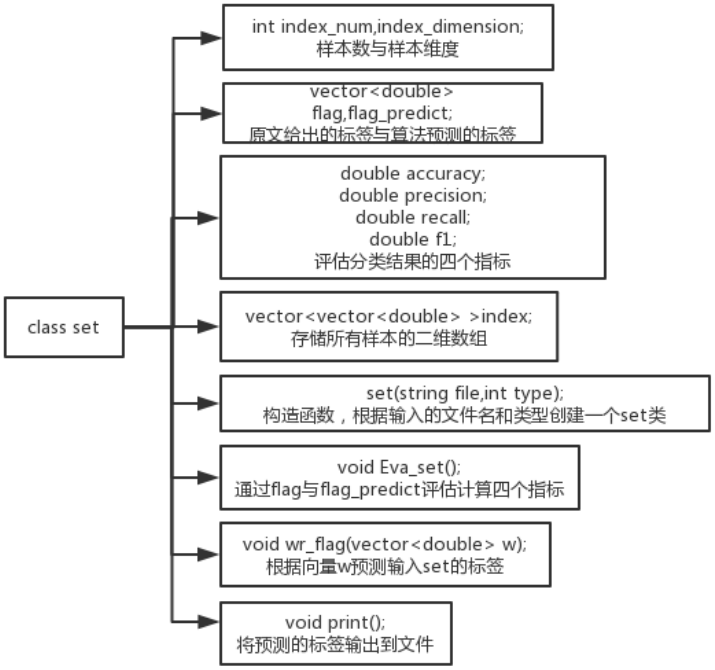
由于 PLA 是一个线性的过程，因此结果很快就会收敛。并且整个迭代的过程并不是逐步趋向最优解的，而是一个来回摆动的过程，因此我们还引入了口袋算法来确保去到最优的 w。

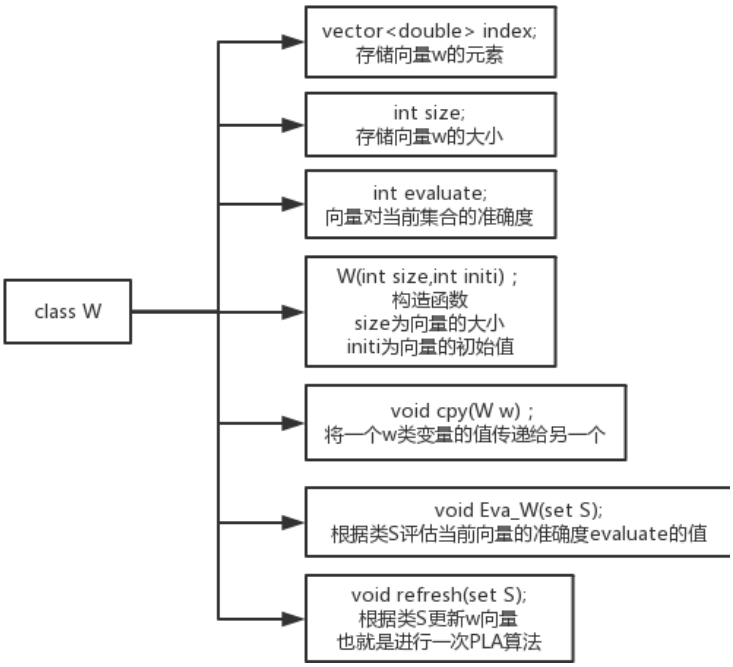
2. 伪代码

首先定义了两个类：set 与 W。

set 用于存储一组点集（训练集、验证集、测试集），并且提供与之相关的函数。

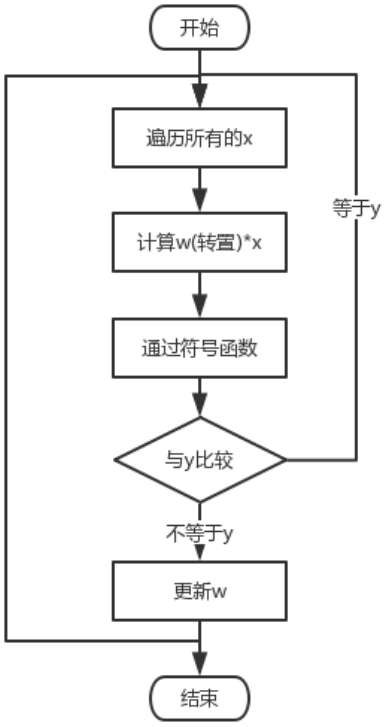
W 类用于存储 PLA 算法的向量 w，并且提供线管的函数（评估，更新等）。

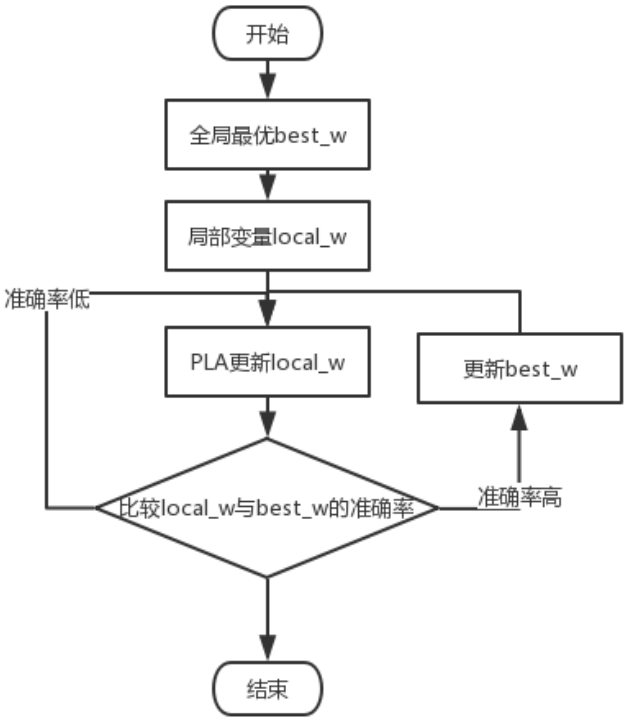




（由于修改过一次代码，最终提交版本的 cpy 函数已经包含在了 refresh 函数内）

其次是 PLA 的核心算法，整体来说非常简单，只要规定好迭代次数，然后一直用 PLA 的剧本法则更新 w 即可。由于是线性的，PLA 算法很快就收敛到最优解。同时为了解决最优解的问题，我们还引入了口袋算法，也就是每次更新完 w 就





3. 关键代码截图

```
class set // 用于存储被分类的所有点
{
    public:
        int index_num;    //点的个数
        int index_dimension;  //点的维度
        double accuracy;  //准确率
        double precision;  //精确率
        double recall;  //召回率
        double f1;  //F1值
        vector<vector<double> >index;  //用于存储每一个x
        vector<double> flag;  //用于存储每一个y
        vector<double> flag_predict;  //用于存储PLA计算出的y
        set();
        set(string file,int type);  //构造函数
        void Eva_set();  //评估四个指标
        void wr_flag(vector<double> w);  //借助w向量生成评估的y值
        void print();  //写结果
};
```

定义了类 set，用于存储所有被分类的点。

```
while(getline(fin,buffer))
{
    index[index_num][0]=1;  //插入1
    int tmp_dimension=1;
    int p1=0;int p2=0;
    string tmp;
    while(p2<buffer.length())
    {
        if(buffer[p2]==',')
        {
            tmp=buffer.substr(p1,p2-p1);
            index[index_num][tmp_dimension]=stod(tmp);
            tmp_dimension++;
            p1=p2+1;
            p2++;
        }
        else
        {
            p2++;
        }
    }
    index_dimension=tmp_dimension;
    if(type==0)  //如果不是测试集，输入标准的y
    {
        if(buffer[buffer.size()-2]=='-')
        {
            flag[index_num]=-1;
        }
        else
        {
            flag[index_num]=1;
        }
    }
    index_num++;
}
```

输入部分的核心代码，将文件读入变量 buffer 以后通过 “，” 来分割每一个数，然后存入相关的变量。

```
void set::wr_flag(vector<double> w)
{
    for(int i=0;i<index_num;i++)
    {
        double tmp_result=0;
        for(int j=0;j<index_dimension;j++)
        {
            tmp_result+=index[i][j]*w[j];
        }
        if(tmp_result>0)
        {
            flag_predict[i]=1;
        }
        else if(tmp_result<0)
        {
            flag_predict[i]=-1;
        }
        else
        {
            flag_predict[i]=0;
        }
    }
}
```

根据向量 w 和 x 来预测每一个 y。

```
void set::Eva_set()
{
    double TP=0;double FN=0;double TN=0;double FP=0;
    for(int i=0;i<index_num;i++)
    {
        if(flag[i]==1)
        {
            if(flag_predict[i]==1) TP++;
            else if(flag_predict[i]=-1) FN++;
        }
        else if(flag[i]==-1)
        {
            if(flag_predict[i]==-1) TN++;
            else if(flag_predict[i]=1) FP++;
        }
    }
    cout<<TP<<" "<<FN<<" "<<TN<<" "<<FP<<endl;
    accuracy=(TP+TN)/(TP+FP+TN+FN);
    recall=TP/(TP+FN);
    precision=TP/(TP+FP);
    f1=(2*precision*recall)/(precision+recall);
}
```

统计 TP、FTN 和 FP，计算四个指标。

```
class W //存储向量w
{
    public:
        vector<double> index; //存储w
        int size; //向量的维度
        int evaluate; //记录向量针对该模型的准确度
        W(int size,int initi)
        {
            this->size=size;
            index.resize(size+1);
            for(int i=0;i<size;i++)
            {
                index[i]=initi; //初始化向量
            }
            evaluate=0;
        }
        void Eva_W(set S); //评估evaluate值
        void refresh(set S,W *best_w); //PLA算法更新w
};
```

定义类 W，存储向量 w。

```
void W::refresh(set S,W *best_w) //best_w是口袋里最优的w
{
    for(int i=0;i<S.index_num;i++)
    {
        double tmp_result=0;
        for(int j=0;j<size;j++) //计算w*x
        {
            tmp_result+=S.index[i][j]*this->index[j];
        }
        if(tmp_result*S.flag[i]<=0) //如果预测错误
        {
            for(int j=0;j<size;j++)
            {
                this->index[j]+=S.index[i][j]*S.flag[i];
            }
            this->Eva_W(S);
            if(this->evaluate>best_w->evaluate) //如果准确率更高
            {
                cout<<i<<endl;
                best_w->evaluate=this->evaluate;
                for(int i=0;i<size;i++) //更新口袋
                {
                    best_w->index[i]=this->index[i];
                }
            }
        }
    }
}
```

PLA 算法的核心部分。

```
int main()
{
    string train_set="DATA\\train.csv"; //训练集
    string val_set="DATA\\val.csv"; //验证集
    string test_set="DATA\\test.csv"; //测试集
    set S1(train_set,0);
    W best_w(S1.index_dimension,1); //口袋中最优的w
    best_w.Eva_W(S1);
    W local_w(S1.index_dimension,1);
    local_w.Eva_W(S1);
    for(int i=0;i<iterate;i++)
    {
        local_w.refresh(S1,&best_w);
        if(best_w.evaluate==S1.index_num)
        {
            cout<<"best_w find!"<<endl;
            break;
        }
    }

    set S2(val_set,0);
    S2.wr_flag(best_w.index);
    S2.Eva_set(); //评估结果准确率
    cout<<"Accuracy="<<S2.accuracy<<endl;
    cout<<"Recall="<<S2.recall<<endl;
    cout<<"Precision="<<S2.precision<<endl;
    cout<<"F1="<<S2.f1<<endl;
    set S3(test_set,1);
    S3.wr_flag(best_w.index);
    S3.print();

    return 0;
}
```

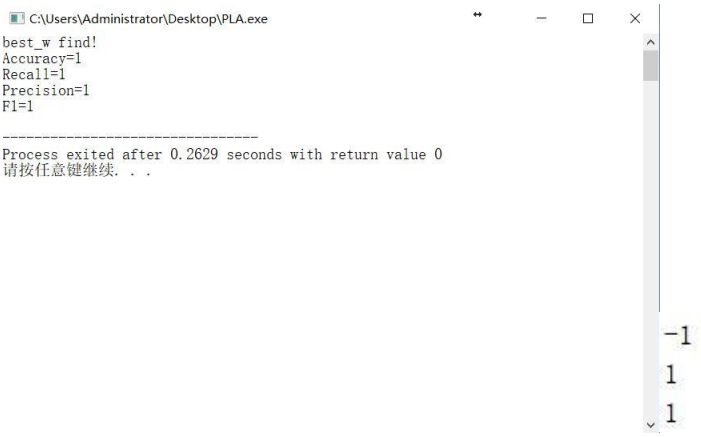
主函数。

三、 实验结果及分析

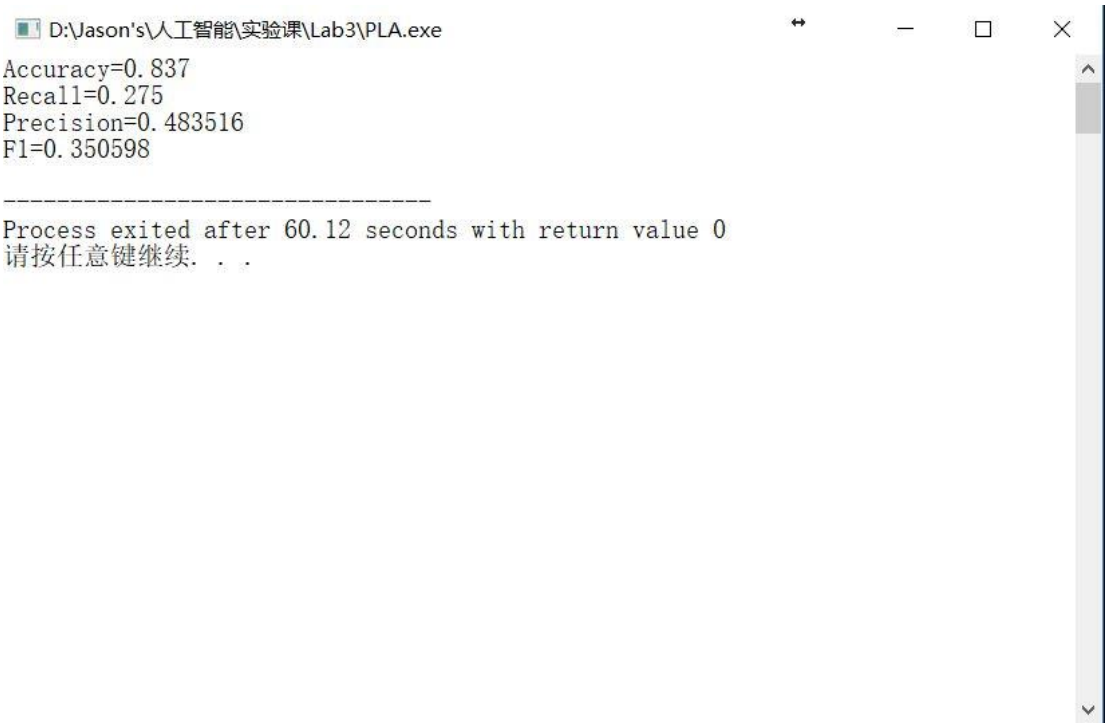
1. 实验结果展示示例

1, 1, 0, −1	1, 1, 0, ?
3, 3, 1, 1	3, 3, 1, ?
4, 3, −1, 1	4, 3, −1, ?
训练集	测试集

运行结果：



2. 评测指标展示即分析



迭代次数为 10 时的运行结果

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？

可以把非线性问题转换为多段的线性问题。分段可以参考样条差值法的思路。

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。

准确率：整体上而言，评估算法整体上的可靠度；

精确率：本质是对预测结果而言的，衡量模型的查准率；

召回率：本质是对原来的样本而言的，衡量模型的查全率；

F1值 :是精确率与召回率的调和平均数 ,因为精确率和召回率有可能出现矛盾，所以引入了F1值。