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

(2016年秋季学期)

课程名称:Artificial Intelligence

教学班	专业(方向)	学号	姓名
M1	移动(互联网)	15352426	张彧

一、实验题目

感知机学习算法(PLA)

1.算法原理

1. 假设输入空间是 $x \subset R^n$,输出空间是y={+1,-1}。由输入空间到输出空间的如下函数

   $$f(x) = sign(w \bullet x + b)$$

   称为感知机。其中，$w \subset R^n$叫做权值或者权值向量，$b \subset R$叫做偏置。其中偏置项b可以揉和到增广向量点积中

   $$f(x) = sign(\tilde{w} \bullet \tilde{x})$$

2. 感知机是一种线性分类模型，属于判别模型。其几何解释为:

   线性方程$w \bullet x + b = 0$对应于特征空间R^n中的一个超平面S，其中w是超平面的法向量(w与超平面中每一个向量都垂直),b是超平面的截距。这个超平面将特征空间划分为两个部分。位于两部分的点(特征向量)分别被分为正、负两类。

3. 一般的感知机的学习策略是极小化损失函数：

   $$\min_{w,b} L(w,b) = \sum_{x_j \in M} y_i(w \bullet x_i + b)$$或者

   $$\min_{w,b} L(w,b) = \sum_{x_j \in M} y_i(w \bullet x_i + b)^2$$

   损失函数对应于误分类点到分离超平面的总距离或距离平方和

4. 本次实验中使用的学习策略是基于四个不同的评价指标

1. 准确度(accuracy)
2. 精确度(precision)
3. 召回率(recall)
4. F1值

5. 感知机学习算法是基于梯度下降法的对损失函数的最优化算法，算法简单且容易实现。首先选取一个超平面，然后用梯度下降法，每次选取一个误分类点，根据学习率（本次实验中取1），不断极小化目标函数。
6. 对线性可分的数据集合，感知机学习算法一定收敛。

2.伪代码

initial

输入：训练数据集$T=\{(x_1,y_1),(x_2,y_2).....(x_N,y_N)\}$，其中$x_i \in R^n$，$y_i \in \{-1,+1\}, i=1,2,3.....N$；学习率$\eta(0<\eta \leq 1)$

输出：$\tilde{w}$；感知机模型$f(x)=sign(\tilde{w} \bullet \tilde{x})$

(1)：选取初值$\tilde{w_0}$，拓展输入x为增广向量$\tilde{x}$

(2)：对训练集中每一个实例(x_i,y_i)

如果$y_i(\tilde{w} \bullet \tilde{x_i}) \leq 0$

$\tilde{w} \leftarrow \tilde{w}+\eta y_i \tilde{x_i}$

转至(3)

(3)转至(2)，直至训练集中没有误分类点或者迭代次数达到上限

pocket

输入：训练数据集$T=\{(x_1,y_1),(x_2,y_2).....(x_N,y_N)\}$，其中$x_i \in R^n$，$y_i \in \{-1,+1\}, i=1,2,3.....N$；学习率$\eta(0<\eta \leq 1)$

输出：$\tilde{w}$；感知机模型$f(x)=sign(\tilde{w} \bullet \tilde{x})$

(1)：选取初值$\tilde{w_0}$，拓展输入x为增广向量$\tilde{x}$，设置当前最优值$\tilde{w}_m=\tilde{w}_0$

(2)：对训练集中每一个实例(x_i,y_i)

如果$y_i(\tilde{w} \bullet \tilde{x_i}) \leq 0$

$\tilde{w} \leftarrow \tilde{w}+\eta y_i \tilde{x_i}$

转至(3)

(3)如果对于当前评测指标，当前的w,b得到的结果优于$\tilde{w}_m$，设置当前最优值$\tilde{w}_m=\tilde{w}$

(4)转至(2)，直至迭代次数达到上限

3.关键代码截图(带注释)

统计正负类预测情况

```
void dataSet::predict(vector<double> wn)//统计正负类预测情况
{
    pre.clear();
    TP=0;
    TN=0;
    FP=0;
    FN=0;
    for(int i=0;i<sampleNum;i++)
    {
        pre.push_back(sign((dot(x[i],wn))));
        if(label[i]==1)
        {
            if(pre[i]==1)TP++;
            else TN++;
        }
        else{
            if(pre[i]==1)FP++;
            else FN++;
        }
    }
}

void dataSet::PLA()
{
    w.resize(dimNum);
    for(int i=0;i<dimNum;i++)//初始化w
    {
        w[i]=1;
    }
    bool itFlag=0;
    int itCnt;
    for(itCnt=0;itCnt<upperLimit;itCnt++)//设置迭代上限
    {
        itFlag=0;
        for(int i=0;i<sampleNum;i++)
        {
            if(!verify(i,w))//找出第一个错分类的sample
            {
                itFlag=1;
                for(int j=0;j<dimNum;j++)
                {
                    w[j]+=x[i][j]*label[i];
                }
                break;
            }
        }
        if(itFlag==0)break;//正确划分所有样本则退出
    }
    //cout<<itCnt<<endl;
    predict(w);//对正负类预测情况进行统计
}
```

```
void dataSet::PLA()
{
    w_F1.resize(dimNum);
    w_Pre.resize(dimNum);
    w_Recall.resize(dimNum);
    w_Acc.resize(dimNum);
    w.resize(dimNum);
    for(int i=0;i<dimNum;i++)//初始化w以及四个口袋
    {
        w_Acc[i]=1;
        w_Recall[i]=1;
        w_Pre[i]=1;
        w_F1[i]=1;
        w[i]=1;
    }
    predict(w);
    int itCnt;
    for(itCnt=0;itCnt<upperLimit;itCnt++)//设置迭代上限
    {
        for(int i=0;i<sampleNum;i++)
        {
            if(!verify(i,w))//找出第一个错分类的sample
            {
                for(int j=0;j<dimNum;j++)
                {
                    w[j]+=x[i][j]*label[i];
                }
                break;
            }
        }
        predict(w);
        if(Acc_max<getAccuracy()){
            Acc_max=getAccuracy();
            w_Acc=w;
        }
        if(Recall_max<getRecall()){
            Recall_max=getRecall();
            w_Recall=w;
        }
        if(Pre_max<getPrecision()){
            Pre_max=getPrecision();
            w_Pre=w;
        }
        if(F1_max<getF1()){
            F1_max=getF1();
            w_F1=w;
        }
    }
    predict(w);
}
```

返回四个指标

```
double dataSet::getValue(int n){
    switch (n) {
        case 1:
            return getAccuracy();
            break;
        case 2:
            return getRecall();
            break;
        case 3:
            return getPrecision();
            break;
        default:
            return getF1();
    }
}
```

4.创新点&优化

1. 对误分类项采用随机选取的策略，而不是从第一项开始寻找，防止陷入局部最优。
2. 每次迭代更新所有误分类项，而不是只更新第一项后退出
3. 口袋算法中，对四种测评指标进行了加权后作为综合指标。

二、实验内容

1.实验结果展示示例（可图可表可文字，尽量可视化）

使用实验指导文档中的例子

编号	特征1	特征2	标签
Train1	-4	-1	+1
Train2	0	3	-1
Test1	-2	3	?

初始化w向量所有参数为1，并输出每一步更新过程中的w向量,得到的结果如下。

```
step 0 : w = 2 -3 0
step 1 : w = 1 -3 -3
step 2 : w = 1 -3 -3
1 0.5 1 0.666667
predict test = -1
```

每一步得到的w均与手动运算预期结果一致。

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）

initial版本

由于最终的结果只是与迭代次数有关，因此可以选取不同的迭代次数同时使用四个不同的测评指标进行分析。

```
迭代次数 : 500
train:
Acc : 0.7745 Recall : 0.5
Pre : 0.347007 F1 : 0.409686
val
Acc : 0.7745 Recall : 0.5
Pre : 0.347007 F1 : 0.409686
Program ended with exit code: 0
```

pocket版本

每次迭代过程中得到的w对应四个指标，可以分别记录最大值。同时可以给各项设置权重加和之后得到一个综合的测评系数作为放进口袋的比较依据，得到的w为最终应用于验证集的w。

```
迭代次数 : 500
train:
Acc : 0.8475 Recall : 0.364217
Pre : 0.518182 F1 : 0.427767
Acc_max : 0.85425 Recall_max : 0.71246
Pre_max : 0.625 F1_max : 0.530665
val
Acc : 0.8475 Precision : 0.364217
Pre : 0.518182 F1 : 0.427767
Program ended with exit code: 0
```

三、实验结果及分析

initial版本

更改迭代次数得到不同的结果

```
迭代次数 : 5
train:
Acc : 0.84225 Recall : 0.00958466
Pre : 0.352941 F1 : 0.0186625
val
Acc : 0.828 Recall : 0
Pre : 0 F1 : nan
Program ended with exit code: 0
```

```
迭代次数 : 200
train:
Acc : 0.451 Recall : 0.65655
Pre : 0.171823 F1 : 0.272366
val
Acc : 0.455 Recall : 0.6
Pre : 0.166378 F1 : 0.260516
Program ended with exit code: 0
```

```
迭代次数 : 500
train:
Acc : 0.7745 Recall : 0.5
Pre : 0.347007 F1 : 0.409686
val
Acc : 0.76 Recall : 0.525
Pre : 0.33871 F1 : 0.411765
Program ended with exit code: 0
```

```
迭代次数 : 1000
train:
Acc : 0.349 Recall : 0.915335
Pre : 0.183419 F1 : 0.3056
val
Acc : 0.359 Recall : 0.9125
Pre : 0.188875 F1 : 0.312969
Program ended with exit code: 0
```

```
迭代次数 : 10000
train:
Acc : 0.7295 Recall : 0.678914
Pre : 0.325421 F1 : 0.439959
val
Acc : 0.745 Recall : 0.675
Pre : 0.347267 F1 : 0.458599
Program ended with exit code: 0
```

在迭代过程中，Accuracy一直处于波动状态；而其他三个指标在前五百次迭代中不断上升，之后在一个稳定值附近波动。

对于不可线性划分的数据集，initial版本得到模型的好坏仅由迭代次数决定。迭代次数达到一定值之后，再继续增加迭代次数得到的结果没有显著提升。每增加一次迭代次数较前一次可能得到较坏的结果，也可能得到较好的结果，但都在一个稳定值附近。

pocket版本

基础版

```
迭代次数 : 1000
train:
Acc : 0.78725 Recall : 0.664537
Pre : 0.393567 F1 : 0.494355
Acc_max : 0.84375 Recall_max : 0.950479
Pre_max : 1 F1_max : 0.500307
val
Acc : 0.785 Precision : 0.61875
Pre : 0.391304 F1 : 0.479419
Program ended with exit code: 0
```

```
迭代次数 : 2000
train:
Acc : 0.78725 Recall : 0.664537
Pre : 0.393567 F1 : 0.494355
Acc_max : 0.84375 Recall_max : 0.950479
Pre_max : 1 F1_max : 0.500307
val
Acc : 0.791 Precision : 0.575
Pre : 0.39485 F1 : 0.468193
Program ended with exit code: 0
```

由于口袋算法会记录一个最优结果，因此得到的结果肯定是随迭代次数增加不断优化的。

随机寻找误差项

```
迭代次数 : 2000
train:
Acc : 0.81025 Recall : 0.549521
Pre : 0.419001 F1 : 0.475466
Acc_max : 0.85 Recall_max : 0.992013
Pre_max : 0.833333 F1_max : 0.500705
val
Acc : 0.821 Precision : 0.5375
Pre : 0.450262 F1 : 0.490028
Program ended with exit code: 0
```

由于寻找到第一个误判项之后就更新w并进行下一次迭代，基础版本的口袋算法很有可能系数向量w始终只受到前几个输入文本的调控。在加入随机寻找误判项之后，可以解决这个问题。几项指标相比于基础版都有了一定的提升。

综合评价指标+更新所有误分类项

```
迭代次数 : 2000
train:
Acc : 0.81375 Recall : 0.670927
Pre : 0.437956 F1 : 0.529968
Acc_max : 0.85775 Recall_max : 0.897764
Pre_max : 0.702703 F1_max : 0.535463
val
Acc : 0.821 Precision : 0.5125
Pre : 0.448087 F1 : 0.478134
Program ended with exit code: 0
```

使用$(Accuracy + F1)/2$作为评价指标，得到的最终的权值向量w，在训练集上的结果有一定提升，但是在验证集上的结果并没有提升反而下降了。可能有一定程度的过拟合。

四、实验思考题

1. PLA不适用非线性的问题,有什么其他的手段可以解决数据集非线性可分的问题?
    1. 进行非线性假设，根据预先对数据进行预估，设定每个维度对应的指数项
    2. 使用非线性转换，拓展参数向量为参数矩阵，可以应对各类多项式数据集，但是复杂度较高
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。
    - 在机器学习二元分类的判别中，有两种错误判别：
        - 误判：将负类预测为正类
        - 漏判：将正类预测为负类

      实际问题当中，我们对两种错误判断的关心程度不同。例如安检更注重防止“漏判”等
    - 根据以上的的两种不当判断，四种评测指标的侧重点不同
        - Accuracy(准确率)：无正类负类的区分。
        - Precision(精确率)：只关心误判项
        - Recall(召回率)：只关心漏判项
        - F1(F值)：精确率和召回率的调和平均数，是两者的共同体现。由于调和平均数对极小值的敏感程度高于极大值，因此主要由精确率和召回率中较小的一项决定。
    - 当正负类的样本分布不同时，不同的评测方式适用性不同。例如，当正类很少时，即使将所有的类都预测为负类，得到的准确率依然比较高，但这显然不是一个有效的预测，这时候使用召回率的效果显著优于准确率。