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	周五 5-6 节	专业（方向）	软件工程（移动信息工程）M2
学号	15352204	姓名	林丹

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

- 算法简述：

  PLA 是一个监督学习算法，二分类的线性分类模型。

  输入：样本的 n 维特征向量

  输出：{+1,−1} （+1：正例 positive，-1：反例 negative）

- 数学基础：线性回归

  对于每一个样本，输入 n 个自变量，输出因变量 y，

  目标，找到权重 $[w_0, w_1, \ldots, w_n$，使得 $y = w_0 + w_1x_1 + w_2x_2, \ldots, +w_nx_n$ (理想情况)

  我们需要不断修改权重$[w_0, w_1, \ldots, w_n]$，希望能得到一个准确分类所有权重。

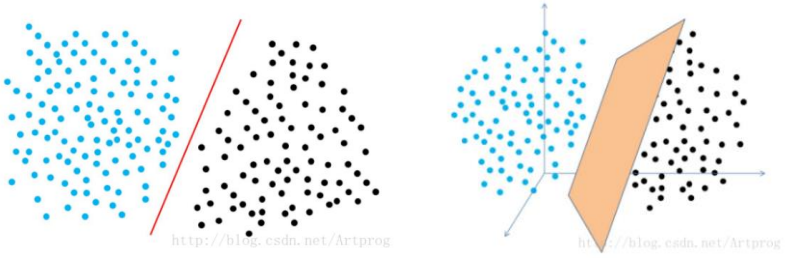
- 感性认识：

  左图：特征向量只有 2 维；

  权重$[w_0, w_1, x_2]$分割的是一条线$y = w_0 + w_1x_1 + w_2x_2 = 0$。

  右图：特征向量只有 3 维；

  权重$[w_0, w_1, w_1, w_3]$分割的是一条线$y = w_0 + w_1x_1 + w_2x_2 + w_3x_3 = 0$。



（图片来源网络）

- 修改权重 **W** 的方法是，通过设定一个阈值，来进行分类：
  大于阈值，判定为 positive：

$$\sum_{k=1}^{d} w_k x_k > threshold$$

  小于阈值，判定为 negative：

$$\sum_{k=1}^{d} w_k x_k < threshold$$

  借助符号函数 sign，得到感知机模型 h(x)

$$h(\mathbf{x}) = sign\left(\left(\sum_{k=1}^{d} w_k x_k\right) - threshold\right)$$

  转化为矩阵运算：

$$\begin{aligned} h(\mathbf{x}) &= sign\left(\left(\sum_{k=1}^{d} w_k x_k\right) - threshold\right) \\ &= sign\left(\left(\sum_{k=1}^{d} w_k x_k\right) + \underbrace{(-threshold)}_{w_0} \cdot \underbrace{(+1)}_{x_0}\right) \\ &= sign\left(\sum_{j=0}^{d} w_j x_j\right) \\ &= sign\left(\tilde{\mathbf{W}}^{\mathrm{T}} \tilde{\mathbf{X}}\right) \end{aligned}$$

  还发现，分割面（线或更高维的超平面，以下简称分割线）和权重向量是正交关系。
  因为分割面就是 $y = w_0 + w_1 x_1 + w_2 x_2, \ldots, + w_n x_n = 0$ 即 $\boldsymbol{W^T X = 0}$

- PLA 口袋算法：
  定义一个全局最优权重向量 **W**，每次更新 **W** 的时候，计算出当前这个权重的错误率，并且和全局最优权重的错误率进行比较。如果这个更新的 **W** 错误率更小，那么更新这个全局最优权重向量。这个错误率的指标可以根据需要修改为其他指标，如 Accuracy、F1 等。

2. 伪代码

2.1 原始 PLA 算法：

Input：$W \in \{1\}^{1\times m}$, $X \in \mathbb{R}^{m\times n}$, $Y \in \mathbb{R}^{1\times m}$, limitTimes
Initialize: Error = 0
Loop:
for t=0 to limitTimes
    for i=1 to n
    if sign($W_t X_i$) != Y_i
        $W_{t+1} = W_t + Y_i X_i$
        Error = Error +1
    if Error = 0, break the loop
end
Output：W_best

2.2 口袋算法：

Input：$W \in \{1\}^{1\times m}$, $X \in \mathbb{R}^{m\times n}$, $Y \in \mathbb{R}^{1\times m}$, limitTimes
Initialze: W_best, min_Error
For t=0,1,2..., limitTimes
    find a mistake of Wt , Y_i, X_i
    $W_{t+1} = W_t + Yi\ Y_i X_i$
    cout the error of W_{t+1}, E_t+1
    if error < min_Error, W_best = W_{t+1}
end
Output：W_best

---

3. 关键代码截图（带注释） 基于 MATLAB 的实现

3.1 PLA 原始算法：

```
TP = 0;  % Truely postive   正确地预测为+1
TN = 0;  % Truely negative  正确地预测为-1
FP = 0;  % Falsely postive   错误地预测为+1，本来为-1
FN = 0;  % Falsely negative  错误地预测为+1，本来为+1

[trainRow,trainColumn] = size(trainVectors);  % 获取 [样本个数，特征向量的长度+1]
[valRow,valColumn] = size(valVectors);        % 获取 [样本个数，特征向量的长度+1]
[testRow,testColumn] = size(testVectors);     % 获取 [样本个数，特征向量的长度]
w = ones(1,trainColumn);   % 初始化权重向量w
valP = zeros(1,valRow);     % P 验证集预测结果向量
LimitTimes = 100;
```

```
Iteration = 0;                %迭代次数
while true                    %一直循环到满足收敛条件
    Iteration = Iteration + 1;
    Continue = false;         %是否需要需要修改w
    for i = 1 : trainRow      %验证每一个样本是否预测正确
        x = [ 1, trainVectors(i,1: trainColumn - 1)];
        % x 增广矩阵,前面补1 ，后面去掉正确答案
        if sign( dot(w, x )) ~= trainVectors(i,trainColumn)
            % 预测≠正确答案
            w = w + trainVectors(i,trainColumn) .* x ;
            Continue = true;  %预测错误，需要修改
            break;
        end
    end
    if ~Continue || Iteration >= LimitTimes    % 3 * trainRow
    % 停止迭代条件：全部预测正确 or 超出迭代次数（2*trainRow）
        break;
    end
end
```

3.2 用验证集验证各个指标：

```
Error = 0;
for i = 1 : valRow
    x = [ 1, valVectors(i,1: valColumn - 1)];
    predict = sign( dot(w, x )) ;
    valP(i) = predict ;  % 验证集预测结果
    ground_truth = valVectors(i,valColumn) ;
    if predict == 1 && ground_truth == 1
        TP = TP + 1;
    elseif predict == -1 && ground_truth == -1
        TN = TN + 1;
    elseif predict == 1 && ground_truth == -1
        FP = FP + 1;
    else
        FN = FN + 1;
    end
end
AccuracyRate = (TP+TN) / valRow
Recall = TP / (TP+FN)
Precision = TP / (TP+FP)
F1 = 2*Precision*Recall / (Precision+Recall)
```

3.3 口袋算法：

（分别用了以 **Accuracy** 和 **F1** 值作为 **W** 比较的指标）

```
w_best = w;                % 全局最优w
w_next = w ;
valP = zeros(1,valRow);     % P 验证集预测结果向量
max_F1 = 0;
Iteration = 0;              %迭代次数
while true                  %一直循环到满足收敛条件
    Iteration = Iteration + 1;
    Continue = false;       %是否需要需要修改w
    TP = 0;  % Truely postive    正确地预测为+1
    TN = 0;  % Truely negative   正确地预测为-1
    FP = 0;  % Falsely postive    错误地预测为+1，本来为-1
    FN = 0;  % Falsely negative   错误地预测为+1，本来为+1
    Error = 0;
    for i = 1 : trainRow    %验证每一个样本是否预测正确
        x = [ 1, trainVectors(i,1: trainColumn - 1)];
        % x 增广矩阵,前面补1 ，后面去掉正确答案
        predict = sign( dot(w, x )) ;
        ground_truth = trainVectors(i,trainColumn) ;
        if predict ~= ground_truth
            % 预测≠正确答案
            w_next = w + trainVectors(i,trainColumn) .* x ;
            Continue = true;  %预测错误，需要修改
            Error = Error + 1 ;
        end
```

```
        if predict == 1 && ground_truth == 1
            TP = TP + 1;
        elseif predict == -1 && ground_truth == -1
            TN = TN + 1;
        elseif predict == 1 && ground_truth == -1
            FP = FP + 1;
        else
            FN = FN + 1;
        end
    end
    if ~Continue || Iteration >= LimitTimes
    % 停止迭代条件：全部预测正确 or 超出迭代次数
        break;
    end
    Recall = TP / (TP+FN);
    Precision = TP / (TP+FP);
    F1 = 2*Precision*Recall / (Precision+Recall) ;
    if F1 > max_F1
        w_best = w ;
        max_F1 =  F1 ;
    end
    w = w_next ;
end
```

4. 创新点&优化（如果有）

优化思路：

1）初始化 W 为各个特征的平均值。但是想不到具体的数学证明，这个平均值和分割面是否正交，所以弃

用。

2）口袋算法评判一个 W 的好坏

评判一个 W 的好坏通过计算预测值 dot(WX)与正确值{+1，-1}取差值的平方。类似欧式距离。

但是优化效果一般。

2）每次修改 W 选择哪一个 X 和 Y

原来有 break 的算法，是指每次迭代，都用第一次遇到的错误 X 和 Y 来更新 W，导致后面的样本没有机会修改 W。

或者用一个数组保存错误的样本，然后随机选择一个。

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）【小数据集】

	增广 1	特征向量		标签
样本 1	1	3	3	1
样本 2	1	4	3	1
样本 3	1	1	1	-1

初始化 w 为 [0,0,0]

输出结果：

1）原始 PLA（更新了 W 后不从头开始）：

W 的更新（MATLAB 命令窗口截图如下）

```
w =
     0     0     0

w =
     1     3     3

w =
     0     2     2

w =
    -1     1     1

w =
    -2     0     0

w =
    -1     3     3

w =
    -2     2     2

w =
    -3     1     1

AccuracyRate =
     1

Recall =
     1

Precision =
     1

F1 =
     1
```

分析：经过手算验证，过程正确。

2） 口袋算法：

```
w =
     0     0     0
w =
     1     4     3
w =
     0     3     2
w =
     0     3     2
w =
    -1     2     1
w =
    -2     1     0
AccuracyRate =
     1
Recall =
     1
Precision =
     1
F1 =
     1
```

分析：经过手算验证，过程正确。

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）

使用算法	验证集评测指标			
	Accuracy	Recall	Precision	**F1**
PLA 原始算法，100 次（循环 break）	0.7910	0.2562	0.3130	0.2818
PLA 原始算法，1000 次（循环 break）	0.3590	0.9125	0.1889	0.3130
PLA 原始算法，5000 次（循环 break）	0.7780	0.7063	0.3924	0.5045
PLA 原始算法，10000 次（循环 break）	0.7450	0.6750	0.3473	0.4586
PLA 原始算法，100 次（循环不 break）	0.8370	0.2813	0.4839	0.3557
PLA 原始算法，1000 次（循环不 break）	0.8430	0.2000	0.5246	0.2896
PLA 原始算法，10000 次（循环不 break）	0.8040	0.6438	0.4256	0.5124
口袋算法，F1 为指标，1000 次	0.6000	0.7750	0.2541	0.3827
口袋算法，F1 为指标，5000 次	0.7530	0.5500	0.3346	0.4161
口袋算法，Accuracy 为指标，1000 次	0.4960	0.6312	0.1850	0.2861
口袋算法，错误率为指标，随机选错修改，1000 次	0.8340	0.1250	0.4348	0.1942
口袋算法，错误率为指标，随机选错修改，5000 次	0.8380	0.2062	0.4853	0.2895

【注】循环 break：是指每次迭代，都用第一次遇到的错误 X 和 Y 来更新 W

分析：

1）原始的 PLA 算法

优点，运算速度快；缺点：取最后一次迭代的 W 作为最优 W，显然不一定迭代次数大就结果好，

2）口袋算法：

优点：选择全局最优 W；

缺点：运算速度满。

3）随机选错修改

Accuracy 指标较高，随着迭代次数增加，F1 值也增加。

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？

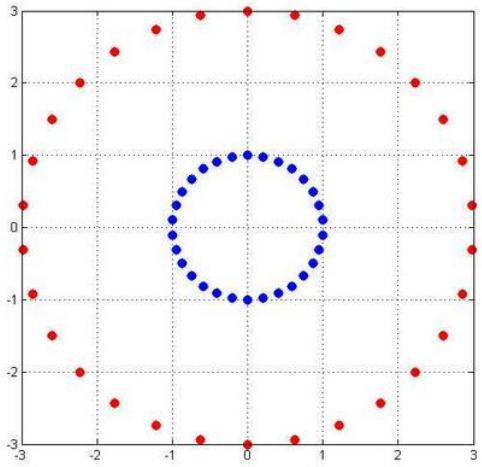
1）如果不是线性可分的，意味着任何一条直线分割都会有错误。

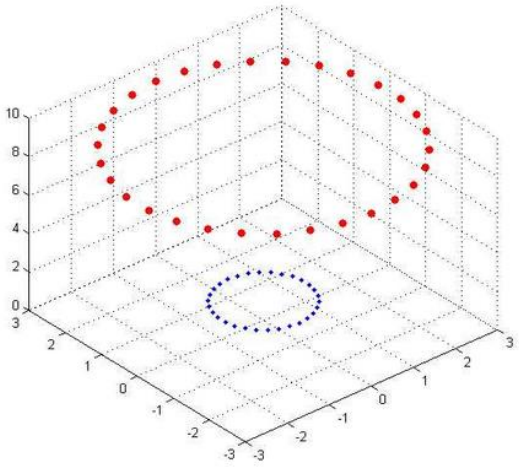
处理思路：找出一条判断错误最少的直线。即，

$$\boldsymbol{W} = arg\ \min_{\boldsymbol{W}} \sum_{i=1}^{N} \|y_i - sign(W^T X_i)\|$$

但是，由于这些分割线有无数条，所以要找到一条最优的不容易。

2）扩展到更高维的特征向量空间。





（图片来源于知乎）

对于左图，样本的特征向量是二维的，所以分割线是形如，$y = w_0 + w_1x_1 + w_2x_2 = 0$

即 $x_2 = -w_0/w_2 - w_1x_1/w_2$

但是通过一个函数关系，将二维的特征向量空间映射到三维的特征向量空间，这样就是线性可分的。

或者我们在二维平面不满足于用线性分割，即对于左图分割线可以考虑用一个曲线分割，比如二次曲线。转化最关键的部分就在于找到 x 到 y 的映射方法。

2、解释为什么用以下作为评测指标，各自的含义是什么？

先解释四个概念：

TP，True positive　正确地预测为+1　（搜到的也想要的）
FN，False negative　错误地预测为-1，本来为+1　（搜到的但没用的）
TN，True negative　正确地预测为-1　（没搜到，然而实际上想要的）
FP，False positive　错误地预测为+1，本来为-1　（没搜到也没用的）

对于样本的分类，有预测错的
用检索识别邮箱中垃圾邮件来举一个例子，
准确率(accuracy)，含义是，分类算法的准确程度

$$\text{Accuracy} = \frac{分类正确的个数}{所有样本个数} = \frac{TP + TN}{TP + FP + TN + FN}$$

精确率(precision)，含义是，如果我们检测的目标是+1 表示检索到的样本，比如判断是否是垃圾邮件，那么

$$\text{Precision} = \frac{系统预测是垃圾邮件中确实是垃圾邮件的个数}{系统预测是垃圾邮件的个数} = \frac{TP}{TP + FP}$$

显然，系统预测是垃圾邮件的个数包括了
系统预测是垃圾邮件中确实是垃圾邮件的个数+系统预测是垃圾邮件中确实是垃圾邮件的个数但实际上不是

对于我们的预测结果而言，或者说是检索出来的样本而言，有多好是检索正确的。

召回率(recall)，含义是，就是从关注领域中，召回（找到的）目标类别的比例

$$\text{Recall} = \frac{系统预测出来的垃圾邮件中确实是垃圾邮件的个数}{实际的垃圾邮件的个数} = \frac{TP}{TP + FN}$$

precision 和 recall 两个指标之间是有矛盾的，最极端的情况：

我们只检索了一次，这个刚好就是垃圾邮件，那么根据公式，precision 就是 100%，但是实际垃圾邮件的个数（一般情况不止一个，假如一个邮箱中有 100 个垃圾邮件 ），那么 recall 就只有 1%.

如果我们检索了全部的邮件，一般的算法不可能做到对所有的邮件分类正确，那么FP，也就是错误地预测为垃圾邮件的也会增加，导致 precision 降低；与此同时，由于正确检测

出来的是垃圾邮件的数量也会增加，所以 recall 增加。

所以需要综合考虑了精确率和召回率，引入了 F-值。

F1 值就是精确率与召回率的调和平均数：

$$F_1 = 2 \cdot \frac{1}{\frac{1}{\text{recall}} + \frac{1}{\text{precision}}} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

通用的 F 值的公式为：

$$F_\beta = (1 + \beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{(\beta^2 \cdot \text{precision}) + \text{recall}}.$$

附：参考资料：

https://www.zhihu.com/question/27210162

http://blog.csdn.net/on2way/article/details/47731455

完。

谢谢评阅 :)