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	专业（方向）	学号	姓名
15M2	互联网方向	15352194	梁杰鑫

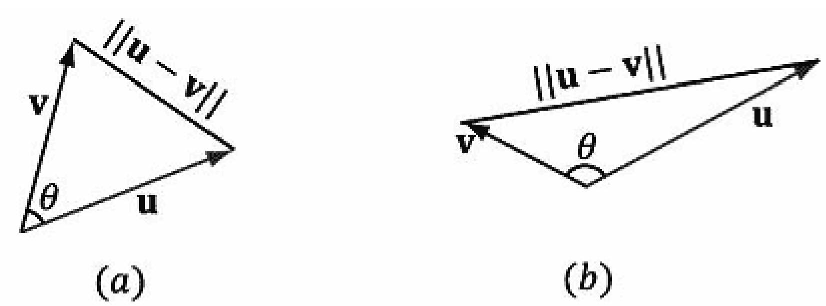
一、实验题目

实现Perceptron Learning Algorithm（感知机学习算法），进行二元分类。

二、实验内容

算法原理

我们先从二维平面上来分析PLA的判断方式。



我们假设$\vec{v}$为权重向量，$\vec{u}$为样本向量。在(a)中，$cos(\vec{u},\vec{v}) > 0$，在(b)中，$cos(\vec{u},\vec{v}) < 0$。同过余弦值的计算，我们就可以区分不同类型的向量，一种在$\vec{v}$向量所界定的法线正面，另一种在反面。这就是PLA的核心思想。

通常我们用以下模型来定义PLA:

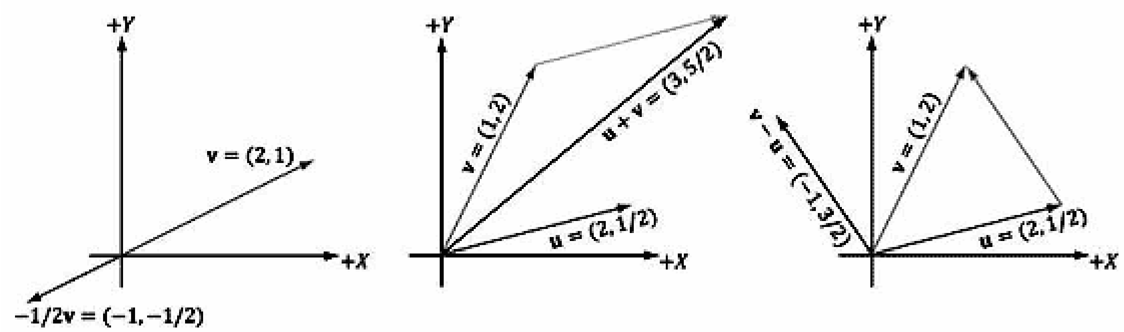
- 权重向量$w = \{w_1, \ldots, w_N\}$
- 样本向量$x = \{x_1, \ldots, x_N\}$
- 阈值$threshold$
- 输出y

$$y = sign(w \cdot x - threshold) \tag{1}$$

在这里我们可以使$w_0 = -threshold$，$x_0 = 1$。然后我们就有$w = \{-threshold, w_1, \dots, w_N\}$，$x = \{1, x_1, \dots, x_N\}$，于是式(1)就可以简化成：

$$y = sign(w \cdot x) \tag{2}$$

接下来时候如何训练权重向量的问题，我们可以利用训练样本中判断错误的样本来对权重向量进行纠正。



通过上图我们很容易知道$u+v$会与u成锐角，而$v-u$会与u成钝角，所以我们通过错误样本和它的标签来对权重向量进行偏移，以纠正权重向量和样本之间的夹角，使得$w = w + tag \cdot x$。

PLA伪代码

则对于一个训练集$U = \{X, T\}$：

- $X = \{x_1, \dots, x_M\}, x_k = \{x_{k1}, \dots, x_{kN}\}$
- $T = \{t_1, \dots, t_M\}, t_k \in \{+1, -1\}$

我们执行以下过程来对权重样本进行训练和测试：

values:

$w = \{1, 1, \dots, 1\}$

$X = \{\{1, x_{11}, \dots, x_{1N}\}, \dots, \{1, x_{M1}, \dots, x_{MN}\}\}$

train:

do

　$err = false$

　for x_k **in** X

　　if $t_k \cdot w \cdot x_k < 0$

　　　$w = w + t_k \cdot x_k, err = true$

while err

test:

$T_w = w^t X$

口袋算法

在初始PLA训练算法中，由于我们的数据集不一定是线性可分的，所以结果可能永远得不到收敛，因此我们要增加终止条件，比如规定迭代次数。另外，我们在训练过程中可能会出现一些非常优秀的不完全解，但是由于解的不完全，遇到判断错误的样本时它又会被覆盖掉。这时候我们就要引入口袋算法，即在训练过程中对w进行评估，将最优秀的w保存下来。

pocket-PLA训练伪代码

pocket train:

$w_{best} = w$

do

  $V_{error} = T - w^t X$

  for i **index** V_{error} **where** $error_i != 0$

    $w = w + t_i \cdot x_i$

    $V_{error} = T - w^t X$ // update error vector, but i is still increasing.

    if $evalution(w) > evalution(w_{best})$

      $w_{best} = w$

while $not\ match\ iteration$ **or** $|V_{error}| != 0$

这里的$evalution$方法可以自定义，可以用w对X的准确率或者F_1值对w进行评估。由于口袋算法每次更新都要更新V_{error}和评估w，所以复杂度会很高，为$O(|X|^2)$。

评估指标

对于一个模型，有很多种评估方法，通常我们会想到采用准确率来评估，但是这其实是不科学的。举个例子：

假设现在有10000个样本，其中9900个是-1，100个是+1。这时候我全部都判断为-1，准确率就有99%了，比绝大部分模型的准确率都要高。

从这里可以看出准确率非常依赖于样本分布，所以我们需要采用其他评估方法。

这里有四个统计量：

TP : true postive，实际为1，预测也为1。

FP : false postive，实际为-1，预测为1。

TN : true negetive，实际为-1，预测也为-1。

FN : false negetive，实际为-1，预测为1。

根据这四个统计量，我们可以计算另外四个指标：

$accuracy = \frac{TP+TN}{TP+FP+TN+FN}$，准确率。

$precision = \frac{TP}{TP+FP}$，精确率，预测为1的样本中实际也为1的。

$recall = \frac{TP}{TP+FN}$，召回率，实际为1的样本中预测也为1的。

$F_1 = \frac{2}{1/precision+1/recall} = 2 \cdot \frac{precision \times recall}{precision+recall}$，精确率和准确率的调和平均数。

关键部分代码

由于PLA测试只需要一行代码，故略过。

训练（省去读取参数的部分）

```
function [w] = pla_train(train, tag, param)
M = length(tag); N = size(train, 2)+1;
train = [ones(M, 1) train]; w = ones(1, N);
% ...prepareing params...
% init best w, err.
[b_e, err] = pla_eval(tag, sign(train*w'));
%init loop.
e = b_e; k = 1; perm = 1:M;
while k <= param.iteration && e.accuracy ~= 1
    if strcmp(param.access, 'random')
        perm = randperm(M); % random permutation of indices.
    end
    for i = perm
        if err(i)
            % renew current w.
            w = w+tag(i)*train(i, :);
            % evalute current w.
            [e, err] = pla_eval(tag, sign(train*w'));
            if strcmp(param.mode, 'pocket') % pocket algorithm.
                if strcmp(param.eval, 'f1')
                    better = e.f1 > b_e.f1;
                else
                    better = e.accuracy > b_e.accuracy;
                end
                % renew best w.
                if better
                    b_e = e; b_w = w;
                end
            end
        end
    end
    k = k + 1;
end
if strcmp(param.mode, 'pocket')
    w = b_w;
end
end
```

具体实现的代码中提供了各种参数，并将口袋算法和普通算法结合在一起。

参数说明：

`train` 训练矩阵

`tag` 训练样本标签

`param`参数结构体

- `iteration`：迭代次数，可以是`Inf`。
- `access`：向量的访问顺序，可以是顺序，随机。
- `mode`：初始算法与口袋算法的选择
- `init`：权重向量的初始化方式，`ones`，`zeros`，`rand`。
- `progress`：展示中间结果
- `eval`：评估方式可选择根据准确率或者F_1评估。

评估函数

```
function [evals, err] = pla_eval(actual, predict)
tfpn = actual+2*predict;
fn = tfpn == -1;
fp = tfpn == 1;

err = fn | fp;
tp = sum(tfpn == 3); % sum the states。
tn = sum(tfpn == -3);
fn = sum(fn);
fp = sum(fp);

evals.accuracy = (tp+tn)/(tp+fp+tn+fn);
evals.recall = tp/(tp+fn);
evals.precision = tp/(tp+fp);
evals.f1 = 2*evals.precision*evals.recall/(evals.precision+evals.recall);
if isnan(evals.f1)
    evals.f1 = 0;
end
end
```

利用编码的方式快速统计4个统计量，其算法如下：

$$state = actual + 2 \times predict \tag{3}$$

因为$actual$和$predict$都只有两种取值，采用编码的方式可以将他们的全组合转化为编码，编码表格如下：

tag	actual	predict	state = actual+2*predict
TP	1	1	3
TN	-1	-1	-3
FN	1	-1	-1
FP	-1	1	1

优化

提供更多评估选项，考虑到$accuracy$的局限性，转而使用F_1值作为口袋算法更新w的指标。由于此次的训练样本非常不平衡，所以更换评估指标会取得较好的效果。

三、实验结果及分析

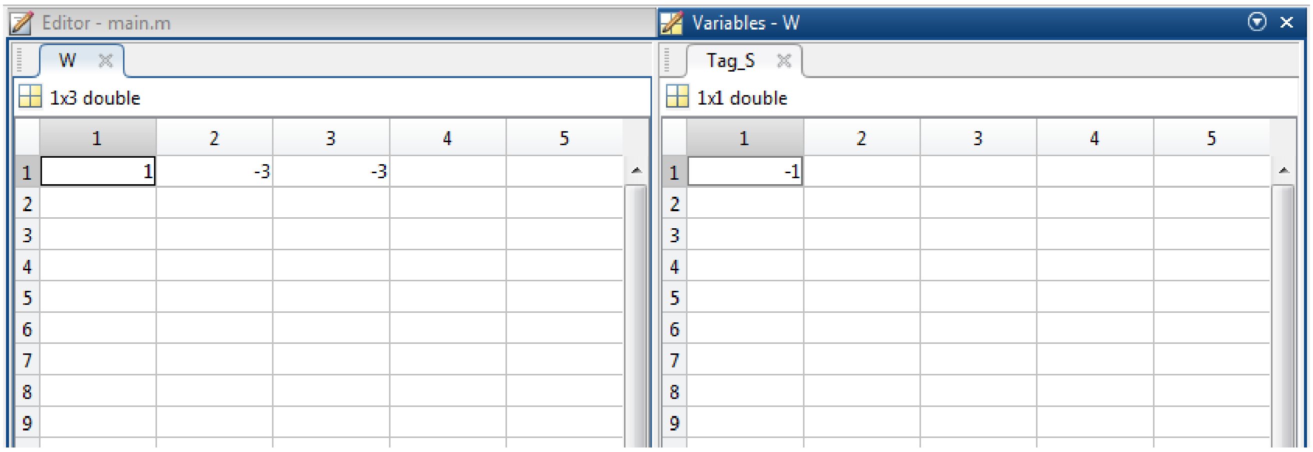
小数据集验证

name	value	tag
train1	1, -4, 1	+1
train2	1, 0, 3	-1
test	1, -2, 3	?

在这个集合中w的训练过程应该是：

$$w = \{1,1,1\} \rightarrow w = w + train_1 * tag_1 = \{2,-3,0\} \rightarrow w = w + train_2 * tag_2 = \{1,-3,-3\}$$

测试的结果是：$y = w^t test = -1$



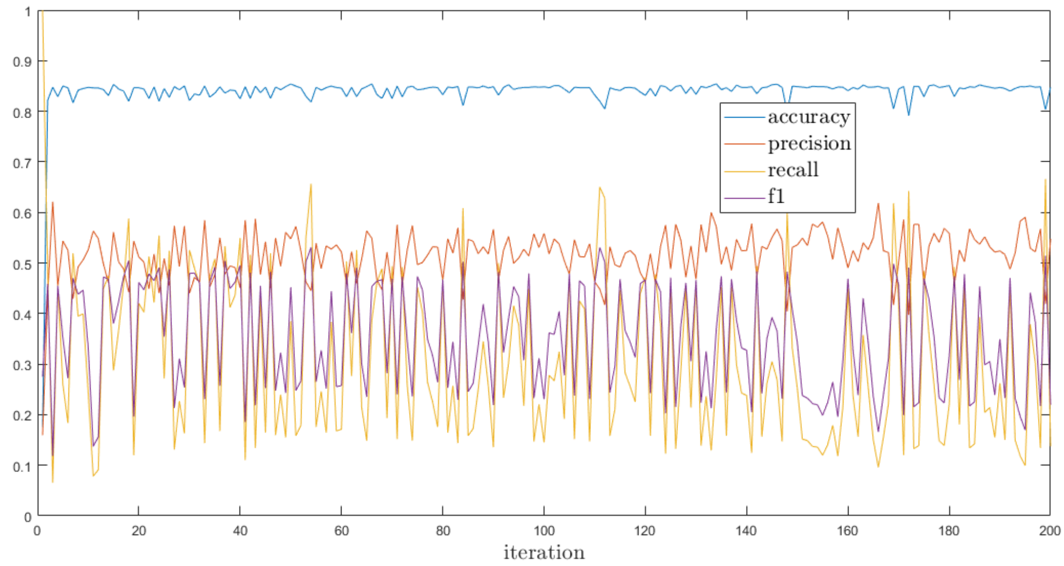
W是训练好的权重向量，Tag_S是测试结果。

参数调整

统一的参数：初始化为ones，迭代次数200，访问方式为顺序访问。

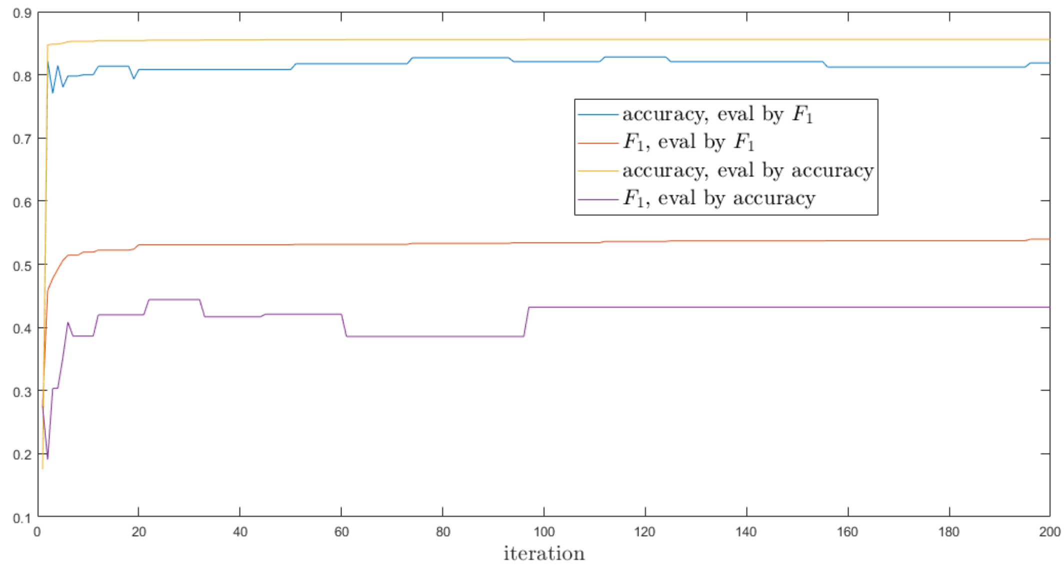
原始算法

原始算法因为没有筛选，所以得出的w是与迭代次数无关的随机量，从下图可以看出，我们得到的w对训练集自身的评估值一直在波动。

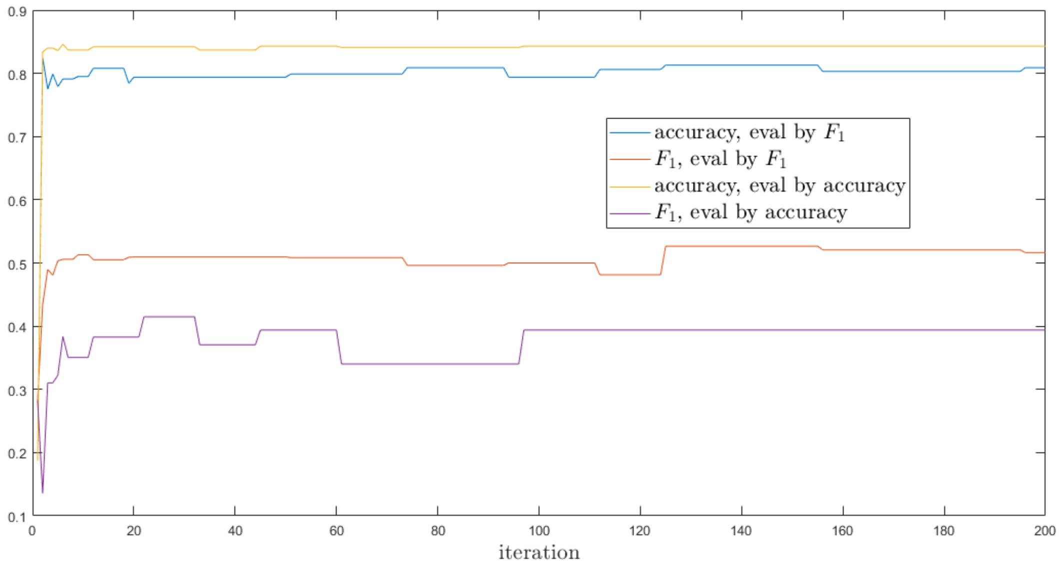


所以引入口袋算法是有必要的，下面给出口袋算法下，以$accuracy$作为评估条件和以F_1作为评估条件对评估参数的影响。

图一，不同的评估方法下w在训练集中的表现：



图二，不同评估方法下w在验证集的表现：



无论从训练集看还是从验证集看，不同评估标准下，准确率$accuracy$相差并不远，而F_1则有很大区别。综合来看，选择F_1作为口袋算法的评优参量对训练是有帮助的。

四、思考题

有什么其他的手段可以解决数据集非线性可分的问题?

1. 采用坐标系映射的方法，将非线性分布的数据变成线性的。比如要用PLA区分圆内圆外的数据时，可以将数据的坐标由(x, y)变成$(|x|+|y|, \sqrt{x^2+y^2})$就可以用PLA划分数据。
2. 将数据升维，在高维度空间对低纬度数据进行划分，可行的方法有核函数变换，将数据映射到希尔伯特空间等等。

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?

$accuracy = \frac{TP+TN}{TP+FP+TN+FN}$，准确率。

$precision = \frac{TP}{TP+FP}$，精确率，预测为1的样本中实际也为1的。

$recall = \frac{TP}{TP+FN}$，召回率，实际为1的样本中预测也为1的。

$F_1 = \frac{2}{1/precision+1/recall} = 2 \cdot \frac{precision \times recall}{precision+recall}$，精确率和准确率的调和平均数。

前面说过，当样本的分布不平衡的时候，准确率将变得没有意义，这时候就需要其他评估方法。

精确率在需要准确识别出正样本时需要用到，而回调率在需要尽量只（是只识别出）识别出正样本是需要用到。然而精确率和回调往往是不能兼得的，在综合评测模型的时候，我们就需要用到他们的调和平均数F_1 $score$。当然还有G $score$、F_α $score$等综合评分。