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	15M2	专业（方向）	互联网
学号	15352218	姓名	林燕娜

一、 实验题目

逻辑回归实现二元分类

二、 实验内容

1. 算法原理

- **逻辑回归：**

之所以叫逻辑回归，是与概率有关，比起之前的 PLA，更多了一种概率度量。Lab3 的 PLA 是根据W^TX的符号判断其标签为 1 或者-1.不管W^TX的值多大，也不管W^TX的值多小，缺少了一种逻辑。

而逻辑回归就是基于 PLA 的基础上，给予了一种概率。

对于二元分类，逻辑回归的推导如下：

=> 如果概率 P>0.5,则判定为类别 1，如果概率 P<0.5,则判定类别为 0

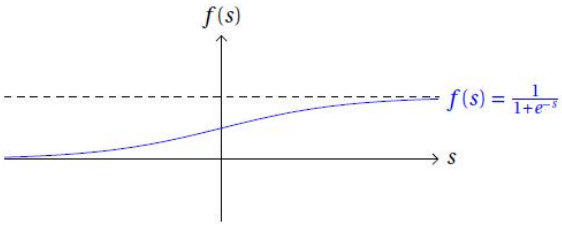
=> 引入几率比：$\frac{p}{1-p}=\begin{cases}>1 & p>0.5\\ <1 & p<0.5\end{cases}$

=> 对几率比取对数：$\ln(\frac{p}{1-p})=\begin{cases}>0 & p>0.5\\ <0 & p<0.5\end{cases}$

可以看到，几率比的对数大于 0 的时候，我们判定为类别 1，当对数小于 0 的时候，判定为类别 0，这与 lab3 的 PLA 一致。

所以，可得：$\ln(\frac{p}{1-p})=W^TX$，也就是：$p=\frac{e^{W^TX}}{1+e^{W^TX}}=\frac{1}{1+e^{-W^TX}}$

曲线大致如下，其中的 s 是W^TX：



曲线公式也叫作 logistics 变换，将一个（-∞,+∞）的值转换为一个[0,1]之间的值。

这就将 PLA 的 $W^T X$ 的值转化为逻辑回归的 P，这就是 PLA 和逻辑回归的关系。

- **极大似然度**

当 P 越接近 1 的时候，说明把样本判为 1 的可信度就越大，当 P 越靠近 0 的时候，同样说明把样本判为 0 的可信度就越大。

也就是样本为 1 的时候：P 越大越好；

样本为 0 的时候，P 越小越好，1-P 越大越好。

而对整个样本来说，将所有的 P 以及 1-P 连乘，结果越大越好，这就是最大似然度：

$$\prod_{i=1}^{n}(p_i)^{y_i}(1-p_i)^{1-y_i}$$

其中，n 指的是训练集的数据个数，而 y_i 是指第 i 个样本的类别是 0 还是 1.

考虑到 P 和 1-P 都是小于 0 的数，当样本个数过大的时候，连乘会导致精度不够。所以对极大似然度取对数：

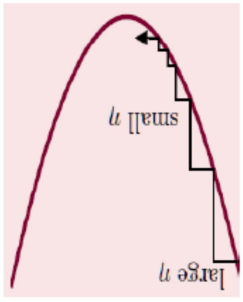
$$L(W)=\ln(\prod_{i=1}^{n}(p_i)^{y_i}(1-p_i)^{1-y_i})=\sum_{i=1}^{n}(y_i\ln p_i+(1-y_i)\ln(1-p_i))$$

$$=\sum_{i=1}^{n}(y_i\ln\frac{p_i}{1-p_i}+\ln(1-p_i))=\sum_{i=1}^{n}(y_i W^T X-\ln(1+e^{W^T X}))$$

因为要取极大似然度最大，所以 $L(W)$ 也是取最大。判断 $L(W)$ 什么时候最大，就需要求出 $L(W)$ 导数。

- **梯度上升（下降）法**

梯度上升法：（在梯度最大的地方，函数取得最大值）



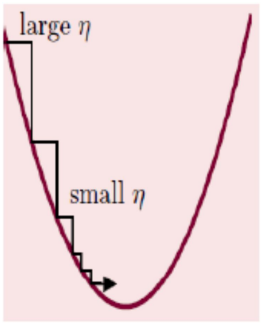
如图，梯度为正，说明在梯度最值的左边，W 需要往右走；梯度为负，说明在梯度最值

的右边，W需要往左走。

所以，W的走向和梯度的正负成正相关，所以W的更新公式为：

$Wnew = W + \eta\Delta$，其中Δ是$L(W)$的梯度。

梯度下降法：



刚好与梯度上升法相反，W的走向和梯度的正负成负相关，所以W的更新公式为：

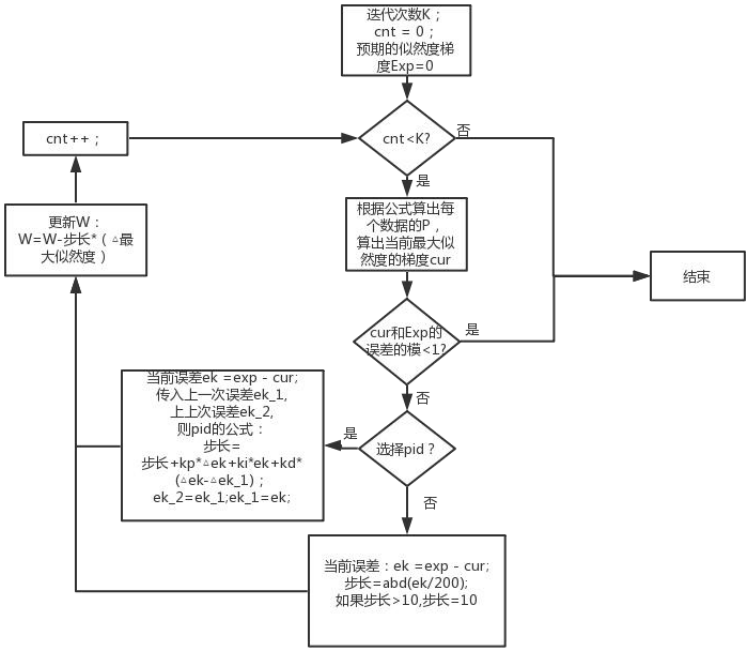
$Wnew = W - \eta\Delta$，其中Δ是$L(W)$的梯度，η是更新的步长。

而$L(W)$的倒数正好是开口向下，所以需要用梯度上升法找到导数为0，也就是$L(W)$取得最大值的点：

$$Wnew = W + \eta\Delta = W + \eta\frac{\partial L(W)}{\partial W} = W + \eta\sum_{i=1}^{n}((y_i - \frac{1}{1+e^{-W^T X}})X_i)$$

而$L(W)$取负数，就变成开口向上，同样可以用梯度下降法求解。

2. 伪代码/流程图



3. 关键代码（带注释）

- **动态更新 W：**

->迭代 1000 次

->根据当前梯度与预期梯度，利用 pid 或者简单的 p 动态算出每个维度当前的学习率

->当达到迭代次数或者是当前梯度与预期梯度的误差在允许范围内，完成更新 W。

```
for k=1:1000
    s = train*W;
    predict = 1.0 ./ (1.0+exp(-s));
    %当前的梯度大小current%
    current = (train.')*(predict-train_label);
    %PID或者简单的P算法得到动态变化的学习率%
    %[learning,ek,ek_1] = pid_self(current,expected,ek,ek_1);%
    learning= P_self(current,expected);
     %如果目前与梯度的预期值的差距的模小于1，则完成%
    if(norm(ek)<1)
        disp(k);
        break;
    end
    %更新w%
    W = W - learning.*current;
end
```

- **增量式 pid：**

->根据期望值和当前值得本次误差、上次误差。上上次误差，根据公式推出当前的 W 各个维度需要的步长。

```
function [y,ek,ek_1] = pid_self(current,expected,e_k1,e_k2,learning)
%算出本次的误差ek%
ek = expected-current;
kp =3;
ki =1.5;
kd = 3;
%根据增里式pid公式得到当前的pid输出，作为步长%
y = kp*(ek-e_k1)+ki*ek+kd*(ek-2*e_k1+e_k2)+learning;
y = abs(y);
%更新ek_1,记录上次的误差%
ek_1 = e_k1;
```

- **简单的线性分段 P：**

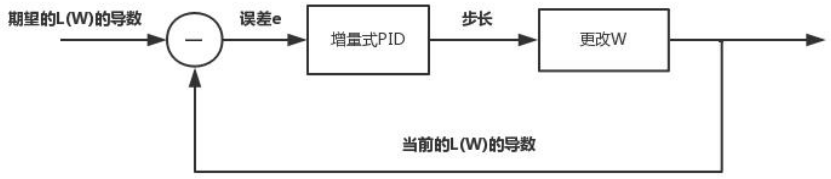
->根据当前值和期望值的误差，做一个正比例函数，并且限制步长在 0-10 的范围内。

```
function y= P_self(current,expected)
 %假设当前梯度值误差大都在0-200之间%
 max = 200;
 ek = expected-current;
 %用一次函数正比例函数，误差大，用大步长，误差小，用小步长%
 y = abs(ek/max);
 %一般不会有需要太大的步长，现在在0-10之间%
for i = 1:length(y)
    if(y(i)>10)
        y(i) = 10;
    end
end
```

4. 创新点&优化（如果有）

动态学习率：

- **借鉴 PID 算法：**



$L(W)$ 的导数的 pid 输出，是 $L(W)$ 的导数的导数，也就是 $L(W)$ 的导数的变化率。而我们是通过步长 η，改变 W 的值，然后进一步更改 $L(W)$ 的导数的大小，可以抽象不太准确理解一下，就是步长越长，$L(W)$ 跨度越大，$L(W)$ 导数的跨度就越大，步长越小，$L(W)$ 导数跨度就越小。所以，步长是和 $L(W)$ 导数的值跨度相关的，而 $L(W)$ 导数大小的跨度就是 $L(W)$ 的导数的变化率。所以可以将其 PID 输出作为步长，不断调整步长的大小。

Pid 的优点就是当距离目标值的差值较大的时候，步长就大。随着差值逐渐变小，步长随之变小。这样子就可以既提高效率又提高准确率。

- **线性分段 P 算法：**

鉴于 PID 的参数过于难调，我们希望达到差值越大，步长越大，最简单粗暴的一种方法就是用一个正相关函数，步长和差值成正比。并且考虑到步长一般不会过大，所以再限制步长的范围在 0 到 10 之间。之所以是正数，是因为：

我们已经把正负相关的运算给予了 W 的更新公式，在公式里面，步长 η 是 W 向左向右更新的一个确切的跨度，是一个正数。

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）

数据集的分法：

选取了数据集的前 6000 数据作为训练集，后 2000 个数据作为验证集合。

结果展示：

注：下面图中的所有 zhibiao 向量是： [Accuracy Recall Precision F1]

原始的：（迭代了 10000 次，步长选择 0.068）

```
>> zhibiao

zhibiao =

    0.7650    0.8458    0.7200    0.7779
```

（迭代 200 次，步长选择 0.013）

```
zhibiao =

    0.7535    0.8561    0.7024    0.7717
```

Pid:

Kp=3，kp=1.5，ki = 3，迭代 10000 次：

```
zhibiao =

    0.7645    0.7575    0.7582    0.7578
```

线性 P:

1000 次：

```
zhibiao =

    0.7675    0.7626    0.7602    0.7614
```

小数据验证：

	属性 1	属性 2	label
train1	1	3	1
train2	3	-2	0
test	2	1	?

W0 = (1,1,1)，步长 $\eta = 0.1$，更新 1 次：

$$\vec{s} = XW^T = \begin{bmatrix} 1 & 1 & 3 \\ 1 & 3 & -2 \end{bmatrix} * \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 5 \\ 2 \end{bmatrix} \Rightarrow$$

$$\vec{p} = \frac{1}{1+e^{-\vec{s}}} = \begin{bmatrix} 0.9933 \\ 0.8808 \end{bmatrix} \Rightarrow$$

$$W_1^T = W_0^T - \eta * \sum_{i=1}^{2} X_i^T \left(\vec{p} - \overrightarrow{label} \right) = \begin{bmatrix} 0.91259 \\ 0.7364 \\ 1.17817 \end{bmatrix}$$

则 test 的预测是：

$$s = W^T * X = \begin{bmatrix} 0.91259 \\ 0.7364 \\ 1.17817 \end{bmatrix} * \begin{bmatrix} 1 & 2 & 1 \end{bmatrix} = 3.56356 \Rightarrow$$

$$p = \frac{1}{1+e^{-s}} = 0.97244314 > 0.5$$

判断测试集为类别 1；

代码结果图如下：

其中，s 是训练集的 s，p 是训练集的概率，W 是更新后的 W。

s_test 是计算测试集的 s，p_test 是测试集的预测概率。

```
>> smalldata
s
     5
     2

p
    0.9933
    0.8808

W
    0.9126
    0.7364
    1.1782
```

```
W
    0.9126
    0.7364
    1.1782

s_test
    3.5636

p_test
    0.9724

判断为类别1
```

明显，结果一致，结果正确。

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）

- **迭代次数不同**

：

算法	Accuracy	Recall	Precision	F1	迭代次数
原始	0.765	0.8458	0.72	0.7779	10000
PID	0.7645	0.7575	0.7582	0.7578	
线性分段	0.745	0.7369	0.7384	0.7377	
原始	0.5075	0.9908	0.4969	0.6619	1000
PID	0.7065	0.8243	0.6585	0.7321	
线性分段	0.7675	0.7626	0.7602	0.7614	

注：原始算法：（步长 0.068），PID 算法：（PID 的取值：3，1.5，3）

对比上面的几种算法，可以看出，原始算法因为步长较小，所以比较依赖迭代次数。当迭代次数较少的时候，原始算法的准确率、F1 都比较低。

反观 PID 和线性分段 P 算法，因为其步长主要依靠于差值的大小自动调整，并且是每个维度自动调整。所以并不是太过于依靠迭代次数，数据都比较稳定。

四、 思考题

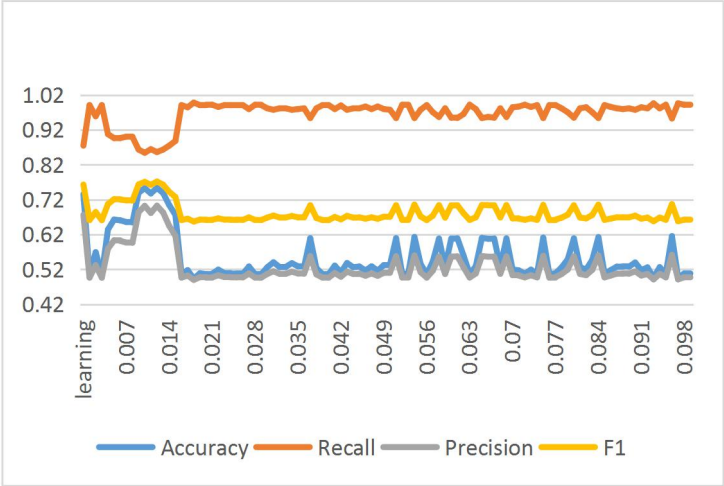
1. 如果把梯度为 0 作为算法停止的条件，可能存在怎样的弊端？

梯度为 0 的情况是很难发生的，可能因为步长设置的问题，使得梯度一直在 0 左右来回不会终止。这可以类比于变量类型 double 或者 float，当我们判断这种类型的变量是否为 0 的时候，不能直接等于 0，因为精度的问题几乎不会等于 0.

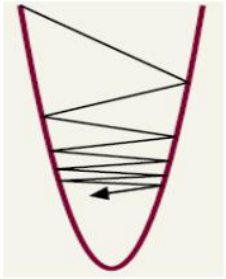
第二是在本次试验中，梯度是一个向量，就算很巧某个维度上的梯度达到 0 了，其他维度都没达到 0，一直继续迭代，所有的维度上的梯度都是 0 就更加困难。

2. η 的大小会怎么影响梯度下降的结果？给出具体的解释，可视化的解释最好，比如图形展示等。

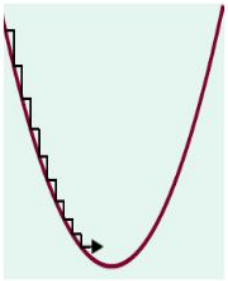
迭代次数 200 次：



- 步长设置过大，求解过程比较震荡，很可能无法求出最优解；
- 步长设置过小，模型一定可以求得最优解，但是耗时比较长。



(a) η过大



(b) η过小

当迭代次数一定的时候，步长大可能继续震荡，步长小的，可能因为迭代次数不够，没有达到最优解。根据下面两张图，就可以大概理解为何上面的曲线如此震荡。而且因为步长是对于每个维度来说的，可能某个维度比较合适的步长是 A，另一个维度合适的步长是 B，所以，就算是步长都过小的时候，也会出现震荡现象。

3. 思考批梯度下降法和随机梯度下降法两种优化方法的优缺点。

- **批梯度下降法**

  批梯度下降法是考虑所有的样本，所以其缺点就是：

  -> 当训练集非常大的时候，计算量非常大

  -> 需要等到所有的数据齐全了才能训练

  -> 最后 L(W)收敛的可能性非常小

  -> 收敛速度慢

  优点：

  -> 考虑到了全部数据，求解出来是全局最优解

  -> 在逼近最优解的过程中，比较稳定。

- **随机梯度下降法**

  随机梯度下降是每次只考虑一个样本，所以，其优点是：

  -> 不管训练集多大，都只需要计算一个样本，计算量少；

  -> 不需要等到所有的数据齐全；

  -> 随机梯度下降一般都可以收敛到最优解；

  其缺点是：

  -> 只用了一个样本，可能求解出来的 W 是对于该样本最优的解，但不是全体最优解；

  -> 选中的单个样本可能是噪声，选中的单个样本需要具有代表性，比较困难。

-> 在逼近最优解的过程中，抖动非常厉害。