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

实验五——逻辑回归模型

二、 实验内容

1. 算法原理

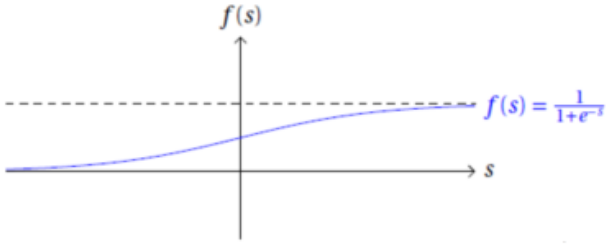
回忆 PLA 算法，希望找到一个权重向量$\mathbf{W}[w_0, w_1, \dots, w_d]$（列向量），使得对于每一个样本$\boldsymbol{X}[x_1, x_2, \dots x_d]$（列向量）有
$w_0 + w_1x_1 + w_2x_2, \dots, +w_dx_d = \boldsymbol{W}^T\boldsymbol{X}$ (理想情况)，通过训练样本得到最优的 $\mathbf{W}$。
对测试样本 X’ 预测，$\boldsymbol{W}^T\boldsymbol{X}' > 0$ 判断为正例$y = 1$，$\boldsymbol{W}^T\boldsymbol{X}' < 0$，判断为反例$y = -1$。
注：转置是因为 $\mathbf{W}$ 是列向量。

逻辑回归：
设正例为事件发生，反例为事件不发生。$p = p("1"|\boldsymbol{X})$
找一种映射关系，使得$\boldsymbol{W}^T\boldsymbol{X}$，$(-\infty, \infty)$映射到区间$[0,1]$，含义是事件发生的概率 p。
这样的函数就是$sigmod$函数。

$$\log\left(\frac{\mathrm{p}}{1-\mathrm{p}}\right) = \boldsymbol{W}^T\boldsymbol{X}$$

$$p = \frac{1}{1 + e^{-\boldsymbol{W}^T\boldsymbol{X}}} = sigmod(\boldsymbol{W}^T\boldsymbol{X})$$

$sigmod$函数图像，$(-\infty, \infty)$映射到 $[0,1]$



设一个数据集有 n 个训练样本，第 i 个样本的标签为$y_i \in \{0,1\}$

每个样本有 d 维的特征，第 i 个样本的特征为$[x_i^{(1)}, x_i^{(2)}, \dots, x_i^{(d)}]$

增广特征向量为

$$\boldsymbol{X_i} = [1, x_i^{(1)}, x_i^{(2)}, \dots, x_i^{(d)}]$$

根据贝叶斯法则，对数据集 D 有似然度：

$$likelihood(\boldsymbol{W}) = \prod_i^n p_i^{y_i}(1-p_i)^{1-y_i} \ ,\ p_i = sigmod(\boldsymbol{W}^T\boldsymbol{X_i})$$

其对数似然度是对上式取对数：$log\big(likelihood(\boldsymbol{W})\big)$

极大似然法求 $\max\limits_{\boldsymbol{W}}\ likelihood(\boldsymbol{W})$，转为求极小值 $\min\limits_{\boldsymbol{W}} -log\big(likelihood(\boldsymbol{W})\big)$

令

$$Err(\boldsymbol{W}) = -log\big(likelihood(\boldsymbol{W})\big)$$

$$= -\sum_i^n [y_i\, log(p_i) + (1-y_i)\log(1-p_i)]$$

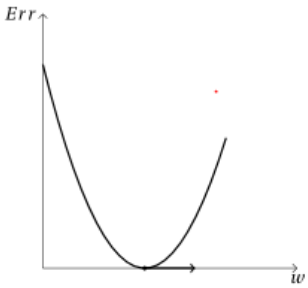
$$= \sum_i^n [\log\big(1+e^{\boldsymbol{W}^T\boldsymbol{X_i}}\big) - y_i\,\boldsymbol{W}^T\boldsymbol{X_i}] = \sum_i^n \log\big(1+e^{\boldsymbol{W}^T\boldsymbol{X_i}}\big) - \sum_i^n y_i\boldsymbol{W}^T\boldsymbol{X_i}$$

$Err(\boldsymbol{W})$ 是一个连续可导，并且存在全局最优解（证明略）

求导：

$$\nabla Err(\boldsymbol{W}) = \sum_i^n [\,(sigmod(\boldsymbol{W}^T\boldsymbol{X_i}) - y_i\,)\,\boldsymbol{X_i}\,] = \sum_i^n [\,(sigmod(A_i) - y_i\,)\,\boldsymbol{X_i}\,]$$

$$A_i = \boldsymbol{W}^T\boldsymbol{X_i}$$



当$\nabla Err(\boldsymbol{W}) = 0$ 时，有$\min\limits_{\boldsymbol{W}} Err(\boldsymbol{W}) = 0$，此时 W 为最优解。

更新$\boldsymbol{W}$

$$\boldsymbol{W} \coloneqq \boldsymbol{W} - \alpha\nabla Err(\boldsymbol{W})$$

直到$\nabla Err(\boldsymbol{W}) = 0$或者迭代次数达到上限。

学习率 α，即每次往梯度下降的方向移动的距离。

2. 伪代码

2.1 原始的逻辑算法：

Function $\boldsymbol{W}$ *= LR_makeW (* $\boldsymbol{X}$*,* $\boldsymbol{Y}$*, limit, alpha)*
Input：
m-samples-dataset with n attributes $\boldsymbol{X} \in \mathbb{R}^{m\times n}$*,*
corresponding label set $\boldsymbol{Y} \in \mathbb{R}^{m\times 1}$
limit, maximum iterations
alpha, learning rate (step size)
Initialize:
$\boldsymbol{X}$ *:= [1,* $\boldsymbol{X}$ *] as the augmented characteristic matrix*
$\boldsymbol{W} := \{1\}^{(n+1)\times 1}$ *initialized augmented weight matrix*
Iteration = 1
1:Loop:
2: $\boldsymbol{A} = \boldsymbol{X} * \boldsymbol{W}$
3: $likelihood(\boldsymbol{W})$ *= sum(Y .* A - log(1+exp(A)))*
*4: gradient = X' * (sigmod(A) - Y)*
5: if gradient == 0 or Iteration >= limit, then
6: return $\boldsymbol{W}$*; break ;*
7: $\boldsymbol{W} = \boldsymbol{W}$ *- alpha * gradient ;*
8: Iteration = Iteration + 1 ;
9:EndLoop
Output: $\boldsymbol{W}$

2.2 利用 W 进行预测：

Function $\boldsymbol{Y}$ *= LR_predict (* $\boldsymbol{X}$*,* $\boldsymbol{W}$ *)*
Input：
m-samples-dataset with n attributes $\boldsymbol{X} \in \mathbb{R}^{m\times n}$*,*
$\boldsymbol{W} \in \mathbb{R}^{(n+1)\times}$ *augmented weight matrix*

Initialize:

$\boldsymbol{X}$ *:= [1,* $\boldsymbol{X}$ *] as the augmented characteristic matrix*

1: $\boldsymbol{P}$*= sigmod(*$\boldsymbol{X}$ * $\boldsymbol{W}$*);*

2: if P_i *>= 0.5, then*

3: $Y_i = 1$

4: else

5: $Y_i = 0$

Output: $\boldsymbol{Y}$*, corresponding label set* $\boldsymbol{Y} \in \mathbb{R}^{m\times 1}$

---

3. 关键代码截图（带注释） 基于 MATLAB 的实现

3.1 构建一棵决策树：

```
function W = LR_makeW(trainVectors, limit, alpha)
[trainRow,trainColumn] = size(trainVectors);  % 获取 [样本个数，特征向量的长度+1]
Y = trainVectors( : ,trainColumn) ;   % 1：发生  0:未发生
data = trainVectors( : ,1:trainColumn-1 ) ;

X = [ ones(trainRow,1), data ] ; % 添加第一个1

W = ones(trainColumn,1);

Iteration = 1 ;
while(true)
    A = X * W ; % 计算“分数”
    L = Y .* A - log(1+exp(A)) ; % 对数似然度 样本数×1
    J = sum(L)/trainRow ; % 代价函数
    grad = X' * (sigmod(A) - Y);
    if Iteration >= limit ||sum(grad) == 0.0000001
        break ;
    end
     W = W - alpha * grad ;
    Iteration = Iteration + 1 ;
end
```

做预测：

```
function test_predicts = LR_predict(testVectors, W)
[testRow,~] = size(testVectors);        % 获取 样本个数×特征向量的长度

x_ = [ ones(testRow,1),testVectors ] ; % 添加第一个1
p_ = sigmod( x_ * W );
test_predicts = zeros(testRow,1);
test_predicts( find( p_ >= 0.5 ) ) = 1;  % 大于等于0.5的判为1  发生
```

4. 创新点&优化（如果有）

4.1 MATLAB 实现，向量化：

$$\boldsymbol{X}=\begin{bmatrix}\boldsymbol{X_1}\\ \boldsymbol{X_2}\\ \dots\\ \boldsymbol{X_n}\end{bmatrix}=\begin{bmatrix}1 & x_1^{(1)} & x_1^{(1)} & \dots & x_1^{(d)}\\ 1 & x_2^{(1)} & x_2^{(1)} & \cdots & x_2^{(d)}\\ \vdots & \vdots & \vdots & \ddots & \vdots\\ 1 & x_n^{(1)} & x_n^{(1)} & \dots & x_n^{(d)}\end{bmatrix}\quad Y=\begin{bmatrix}y_1\\ y_2\\ \dots\\ y_n\end{bmatrix}\quad W=\begin{bmatrix}w_1\\ w_2\\ \dots\\ w_d\end{bmatrix}$$

令$\mathbf{A}=\boldsymbol{W}^T\boldsymbol{X}$，$A_i=\boldsymbol{W}^T\boldsymbol{X_i}$

$$(sigmod(\boldsymbol{A_i})-y_i)=\begin{bmatrix}sigmod(\boldsymbol{A_1})-y_1\\ sigmod(\boldsymbol{A_2})-y_2\\ \dots\\ sigmod(\boldsymbol{A_n})-y_n\end{bmatrix}=sigmod(\boldsymbol{A})-\boldsymbol{Y}$$

$$\boldsymbol{X_i}=[1,x_i^{(1)},x_i^{(2)},\dots,x_i^{(d)}]$$

对于整个数据集 n 个样本：

$\nabla Err(\boldsymbol{W})=\sum_i^n[\,(sigmod(\boldsymbol{A_i})-y_i)\,\boldsymbol{X_i}\,]=\boldsymbol{X}^T(sigmod(\boldsymbol{A})-\boldsymbol{Y})$

更新$\boldsymbol{W}$公式

$$\boldsymbol{W}:=\boldsymbol{W}-\alpha\nabla Err(\boldsymbol{W})=\boldsymbol{W}-\alpha\boldsymbol{X}^T(sigmod(\boldsymbol{A})-\boldsymbol{Y})$$

注意：sigmod 函数实现需要支持向量运算。

```
function g = sigmod(z)
% z -- 可以是一个向量
    g = zeros(size(z));
    g = 1 ./ (1 + exp(-z));
end
```

$$Err(\boldsymbol{W})=\sum_i^n[\,\log(1+e^{\boldsymbol{W}^T\boldsymbol{X_i}})-y_i\boldsymbol{W}^T\boldsymbol{X_i}\,]$$

$$=\sum_i^n\log(1+e^{\boldsymbol{W}^T\boldsymbol{X_i}})-\sum_i^n y_i\boldsymbol{W}^T\boldsymbol{X_i}$$

$$=\mathbf{Y}.*\mathbf{A}-\log(1+e^{\boldsymbol{A}})$$

4.2 标准化。

思路一，利用公式：$X_s=\frac{X-X_{min}}{X_{max}-X_{min}}$，范围在[0,1]

但是问题在于，考虑最大值最小值的时候要考虑测试集的属性值。

思路二：利用 sigmod 函数，将数据从$(-\infty,\infty)$映射到 [0,1]

4.3 变步长α 。

思路：初始学习率较大，当梯度下降到接近最优值时，将学习率降低

如何判断梯度下降距离到最优值的距离？

① 看梯度，由于如果梯度的绝对值越小，那么距离最优值的距离越近，学习率减小。

② 看代价函数：

$likelihood(\boldsymbol{W})=\prod_i^n p_i^{y_i}(1-p_i)^{1-y_i}$ 的范围是[0,1]，所以 $Err(\boldsymbol{W})=-log\big(likelihood(\boldsymbol{W})\big)$

最小值取极限是 0（但是到不了 0），$Err(\boldsymbol{W})$距离 0 越近意味着此时距离最优值的距离越近。

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】

这里的小数据利用老师上课演示的例子：

样本	增广特征向量			标签
A	1	3	3	1
B	1	4	3	1
C	1	1	1	0

初始的增广权向量：		增广特征向量	增广权向量的转置乘以增广特征向量	对数似然的值	-0.380
-3	A	1	3	-0.04859	
1		3			
1		3			
	B	1	4	-0.01815	
		4			
		3			
	C	1	-1	-0.31326	
		1			
		1			

初始的增广权向量 W=[-3,1,1]

增广权向量乘以增广特征向量$\mathbf{A} = \boldsymbol{X}\boldsymbol{W}^T$

$$\mathbf{A} = \boldsymbol{X}\boldsymbol{W}^T = \begin{bmatrix}1 & 3 & 3\\ 1 & 4 & 3\\ 1 & 1 & 1\end{bmatrix}\begin{bmatrix}-3\\ 1\\ 1\end{bmatrix} = \begin{bmatrix}3\\ 4\\ -1\end{bmatrix}$$

```
A =

     3
     4
    -1
```

对数似然的值：

$$log\big(likelihood(\boldsymbol{W})\big) = \sum_i^n [y_i\, log(p_i) + (1 - y_i)\log(1 - p_i)]$$

$$= -\sum_i^n [\log\big(1 + e^{\boldsymbol{W}^T\boldsymbol{X_i}}\big) - y_i\, \boldsymbol{W}^T\boldsymbol{X_i}] = \text{sum}(\begin{bmatrix}-0.04859\\ -0.1815\\ -0.31326\end{bmatrix}) = -0.380$$

```
L =

   -0.0486
   -0.0181
   -0.3133
```

```
   -0.3800
```

步长为0.1更新后的增广权向量：		增广特征向量	增广权向量的转置乘以增广特征向量	对数似然的值	-0.375
-3.02	A	1	2.92	-0.05253	
0.99		3			
0.99		3			
	B	1	3.91	-0.01984	
		4			
		3			
	C	1	-1.04	-0.30266	
		1			
		1			

以步长 0.1 更新 W

$$\boldsymbol{W} \coloneqq \boldsymbol{W} - \alpha\nabla Err(\boldsymbol{W}) = \boldsymbol{W} - \alpha\boldsymbol{X}^T(sigmod(\boldsymbol{A}) - \boldsymbol{Y}) = \begin{bmatrix}-3.02\\ 0.99\\ 0.99\end{bmatrix}$$

```
grad =

    0.2035
    0.0547
    0.0727
```

```
W =

   -3.0204
    0.9945
    0.9927
```

增广权向量乘以增广特征向量$\mathbf{A} = \begin{bmatrix}2.92\\ 3.91\\ -1.04\end{bmatrix}$

```
A =

    2.9414
    3.9359
   -1.0331
```

对数似然的值：

$$log\big(likelihood(\boldsymbol{W})\big) = \text{sum}(\begin{bmatrix}-0.0514\\ -0.0193\\ -0.3045\end{bmatrix})$$

```
L =

   -0.0514
   -0.0193
   -0.3045
```

一直迭代下去，直到梯度为 0，或者迭代次数达到上限。

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）

步长	验证 1	验证 2	验证 3	验证 4	验证 5	平均正确率
0.1	0.535	0.713125	0.716875	0.750625	0.57	0.657125
0.01	0.734375	0.719375	0.65875	0.701875	0.671875	0.69725
0.001	0.689375	0.515	0.549375	0.67875	0.74375	0.63525
0.0001	0.67125	0.735625	0.65125	0.698125	0.5525	0.66175
0.00001	0.773125	0.7825	0.763125	0.77	0.785	0.77475
0.000001	0.739375	0.754375	0.7375	0.734375	0.7325	0.739625

分析：
整体上看，迭代次数都为 3000 次的时候，步长为 0.00001 时候，模型验证的正确率更高。

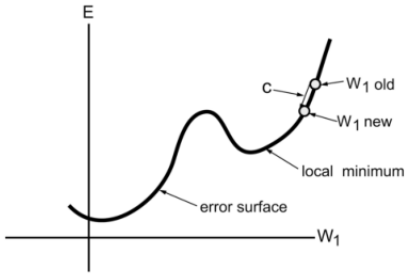
标准化：

步长	验证 1	验证 2	验证 3	验证 4	验证 5	平均正确率
0.1	0.775625	0.740625	0.735625	0.65125	0.735625	0.72775
0.01	0.574375	0.6025	0.664375	0.689375	0.7125	0.648625
0.001	0.758125	0.7575	0.578125	0.61375	0.75125	0.69175
0.0001	0.778125	0.773125	0.761875	0.760625	0.765	0.76775
0.00001	0.769375	0.785625	0.763125	0.7625	0.766875	0.7695
0.000001	0.711875	0.725625	0.7225	0.71625	0.698125	0.714875

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？

1) 当数据量大的时候，几乎不可能能做到梯度恰好为 0。可能会使得迭代无法停止，或者时间开销很大。
2) 如果代价函数有几个局部最优解，那么把梯度为 0 作为停止条件会错过全局最优解。
如下图所示：

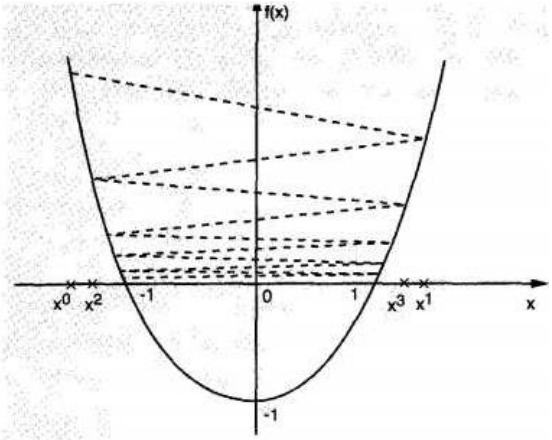


2. 思考题： η 的大小会怎么影响梯度下降的结果？给出具体的解释，可视化的解释最好，比如图形展示等.

学习率η 即步长（stepsize）。

η 决定了每一次迭代过程中，会往梯度下降的方向移动的距离。

如果η 太大，W 会在最优点附近来回跳动，甚至发散不会收敛：



（图片来源于网络）

但是如果步长太短，尽管精度较高，发散的风险减小。

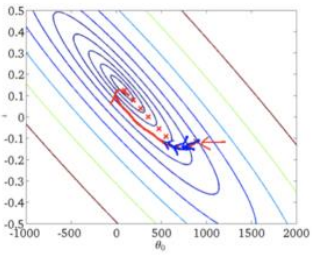
而且由于每步的移动距离很短，就会导致算法收敛速度很慢，时间开销大。

3.思考这两种优化方法的优缺点

批梯度下降

每次更新参数，考虑所有样本

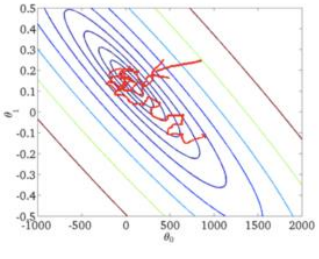
$$\tilde{\mathbf{W}}^{(j)} - \eta \sum_{i=1}^{n} \left[\left(\frac{e^{\tilde{\mathbf{W}}^{\mathrm{T}} \tilde{\mathbf{X}}_i}}{1 + e^{\tilde{\mathbf{W}}^{\mathrm{T}} \tilde{\mathbf{X}}_i}} - y_i \right) \tilde{\mathbf{X}}_i^{(j)} \right]$$



随机梯度下降

每次更新参数，考虑1个样本

$$\tilde{\mathbf{W}}^{(j)} - \eta \left[\left(\frac{e^{\tilde{\mathbf{W}}^{\mathrm{T}} \tilde{\mathbf{X}}_i}}{1 + e^{\tilde{\mathbf{W}}^{\mathrm{T}} \tilde{\mathbf{X}}_i}} - y_i \right) \tilde{\mathbf{X}}_i^{(j)} \right]$$



批梯度下降，综合考虑了每一个样本，找到对于数据集整体的梯度。

优点：验证的准确度更高，但是由于每一次迭代的每一维的 W 都要考虑所有样本，如果数据量大的话，时间开销大。

随机梯度下降就避免了时间开销大的问题，每次更新 W 就随机选择一个样本，缺点是由于是随机的，验证的准确度不一定好。