

Logistic Regression Model

陈欣鸿 刘金杨

前次实验的问题

- 实验原理

- 1. 算法原理

- PLA 算法就是预先初始化向量 \mathbf{w} ，根据训练集的数据来修正 \mathbf{w} 向量，使得在已知的验证集真实数据中的准确率增高，最后得到的 \mathbf{w} 可以用来判断测试集中数据的二元分类问题。

- 1. 算法原理

- PLA 算法是指感知机尝试找到一个能够线性划分训练集数据的平面（如将本次实验训练集中的样本划分为 1 或 -1 两种）。过程中用梯度下降法使误差减小。

- 经过一系列数学运算后可以用权值向量和样本特征向量的点乘的 sign 函数得到结果。（ sign 函数：当 $k > 0$ 时， $\text{sign}(k) = +1$ ；当 $k < 0$ 时， $\text{sign}(k) = -1$ 。）

前次实验的问题

- 伪代码正例

Algorithm 1 PLA原始算法

```
1: for each epoch do  
2:    $newY \leftarrow sign(w, X)$   
3:    $\vec{a} \leftarrow \text{vectors of zero}$   
4:   for each  $i \in \text{shape of } newY$  do  
5:     if  $newY[i] \neq Y[i]$  then  
6:        $a[i] \leftarrow Y[i]$   
7:     end if  
8:   end for  
9:    $\vec{w} \leftarrow \vec{w} + dot(a, X)$   
10: end for
```

前次实验的问题

- 伪代码正例

Input: Train Data matrix $X = \{x_{ji}\}_{i=1}^d \in R_+^{m \times d}$ and Label matrix $Y = \{y_j\}_{j=1}^m \in R_+^{m \times 1}$

Output: Best Weight matrix $\tilde{W}_{(t+1)} = \{w_j\}_{j=0}^d \in R_+^{1 \times (d+1)}$

Begin

Change matrix X into $X = \{+1, x_{j1}, x_{j2}, \dots, x_{ji}, \dots, x_{jd}\}_{i=1}^d \in R_+^{m \times (d+1)}$

Initialize W to $W = \{w_0, w_1, \dots, w_i, \dots, w_d\}_{i=0}^d = 1$

Repeat

Error = 0

For i = 0 to m **do**

If $\text{sign}(\tilde{W}_t^T \tilde{X}_i) \neq y_j$

$\tilde{W}_{(t+1)} \leftarrow \tilde{W}_{(t)} + y_{i(t)} \tilde{X}_{i(t)}$

Error += 1

break

End

End

Until Error = 0

End

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

前次实验的问题

- 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？

改变一些导致非线性的点的标签：

发现陷入死循环后，将 $abs(w^T x)$ 的值（“距离”）最大的一个的标签反转，

然后继续迭代

有什么其他的手段可以解决数据集非线性可分的问题？

【交叉验证】

基础概念

- 软分类模型
 - 概率模型，对每个分类求概率后取概率最大的分类
 - 如 NB
- 硬分类模型
 - 非概率模型，由决策函数决定
 - 如决策树，PLA等
- 逻辑回归（Logistic Regression Model）
 - 属于软分类算法
 - 通过计算数据权重，根据权重了解预测目标的可能性

逻辑回归

- 理论推导（非常重要！）

- 对于一个软分类问题，目标函数定义如下：
- $f(x) = P(\text{label}|x) \in [0,1]$
- 即在给定特征向量 x 的情况下，属于 label 类的可能性多大
- 特征向量的每一个维度，都会对结果产生影响，那么与 PLA 一样，可以模拟一个带权重的分数：
- $s = \sum_{i=0}^d w_i x_i = \mathbf{w}^T \mathbf{x}$
- 这里为什么 t 从0开始，以及为何把 s 这样表示就不再细讲了，有问题的参照 PLA
- 表达式中的 w_i 表示第 i 维特征的权重， $w_i > 0$ 表示该特征对正类别有正面影响，且值越大，正面影响越大，反之亦然

逻辑回归

- 理论推导（非常重要！）

- 既然对于软分类来说，要算出属于每个分类的概率，而我们之前所学习过的模型均属于硬分类模型，即结果非此即彼，无法知道相关概率，所以需要一个新的决策函数
- 利用某种函数将加权分数映射到另一个更合理的数据空间，使加权分数的大小能够反映概率的大小
- 在逻辑回归里使用的是：logistic (sigmoid) 函数

$$\theta(s) = \frac{e^s}{1+e^s} = \frac{1}{1+e^{-s}}$$

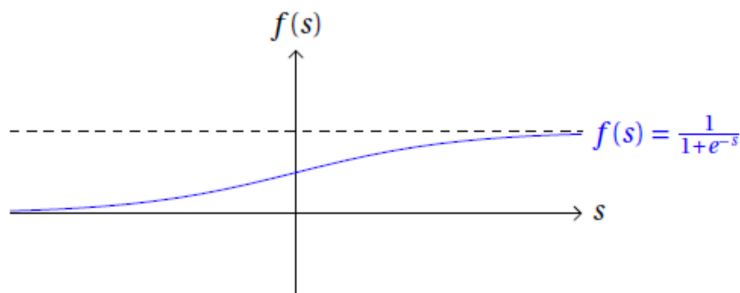
逻辑回归

- 理论推导（非常重要！）

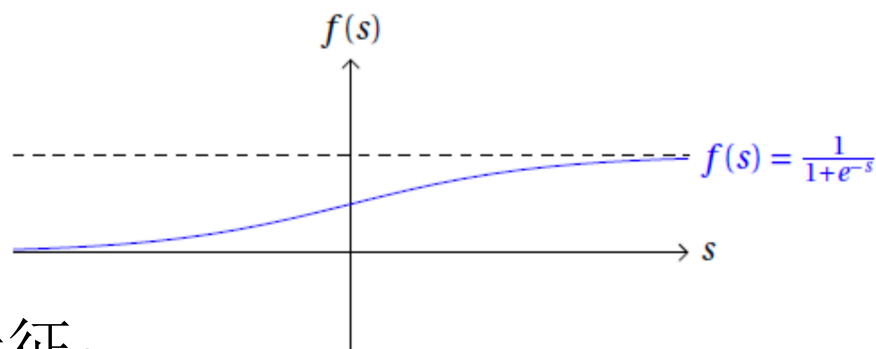
- 在逻辑回归里使用的是： *logistic* (*sigmoid*) 函数

$$\theta(s) = \frac{e^s}{1+e^s} = \frac{1}{1+e^{-s}}$$

- 将数据从 $(-\infty, \infty)$ 映射到 $(0,1)$



逻辑回归



- 理论推导（非常重要！）

- *logistic* (*sigmoid*) 函数的特征：
- $\theta(-\infty) = 0$ ，当加权分数无穷小，该数据属于正类别的概率为 0
- $\theta(0) = 0.5$ ，当加权分数为 0，该数据属于正/负类别的概率为 0.5，即该数据属于任一类别的概率相同
- $\theta(+\infty) = 1$ ，当加权分数无穷大，该数据属于正类别的概率为 1

逻辑回归

- 理论推导（非常重要！）

- 利用 logistic 函数，我们可以构成一个新的假说模型：

- $$h(x) = \frac{1}{1+e^{-w^T x}}$$

- 要求解的是 w

- 根据上面的假说模型， $h(x)$ 算得的是属于正类的概率，属于负类别的概率即为 $1 - h(x)$

- 当 $h(x)$ 大于 0.5 的时候，说明该数据更大可能属于正类别；

逻辑回归

- 理论推导（非常重要！）

- 那么我们可以把最开始提及的目标函数 $f(x)$ 与 $h(x)$ 联合起来：
- $f(x) = P(\text{label}|x) = h(x)^y (1 - h(x))^{1-y}$
- y 表示 x 对应的分类标签
- 当 $y = 1$, $f(x) = P(\text{label}|x) = h(x)$
- 当 $y = 0$, $f(x) = P(\text{label}|x) = 1 - h(x)$

- 用贝叶斯派的观点来看待这个问题
- 不同的参数设置代表着不同的模型，在某种模型下利用给定数据 x 得到给定标签 y 的概率，是这个问题中的似然 (*likelihood*)

逻辑回归

- 理论推导（非常重要！）

- 考虑整个数据集，似然函数如下：
- $likelihood = \prod_{i=1}^M P(label|x_i) = \prod_{i=1}^M h(x_i)^{y_i} (1 - h(x_i))^{1-y_i}$
- 根据最大似然估计算法，要找到一组模型参数，使得上式最大
- 对 $likelihood$ 取对数，再取负数之后，即可得到以下的函数：
- $-\log(likelihood) = -\log \prod_{i=1}^M P(label|x_i)$
- $= -\sum_{i=1}^M y_i \log(h(x_i)) + (1 - y_i) \log(1 - h(x_i))$
- 对以上的函数取最小，即达到最大似然的目的

逻辑回归

- 理论推导（非常重要！）

- $-\sum_{i=1}^M y_i \log(h(x_i)) + (1 - y_i) \log(1 - h(x_i))$ 对 w 求导
- 利用**梯度下降法**，通过不断地迭代使 w 逼近最优解直至收敛
- 求导的步骤在最后，有需要的同学查看辅助文档

$$\begin{aligned}\text{Repeat: } \tilde{\mathbf{W}}_{new}^{(j)} &= \tilde{\mathbf{W}}^{(j)} - \eta \frac{\partial C(\tilde{\mathbf{W}})}{\partial \tilde{\mathbf{W}}^{(j)}} \\ &= \tilde{\mathbf{W}}^{(j)} - \eta \sum_{i=1}^n \left[\left(\frac{e^{\tilde{\mathbf{W}}^T \tilde{\mathbf{x}}_i}}{1 + e^{\tilde{\mathbf{W}}^T \tilde{\mathbf{x}}_i}} - y_i \right) \tilde{\mathbf{x}}_i^{(j)} \right]\end{aligned}$$

Until convergence

- η 表示学习率, j 表示第几维, i 表示第几个样本

逻辑回归

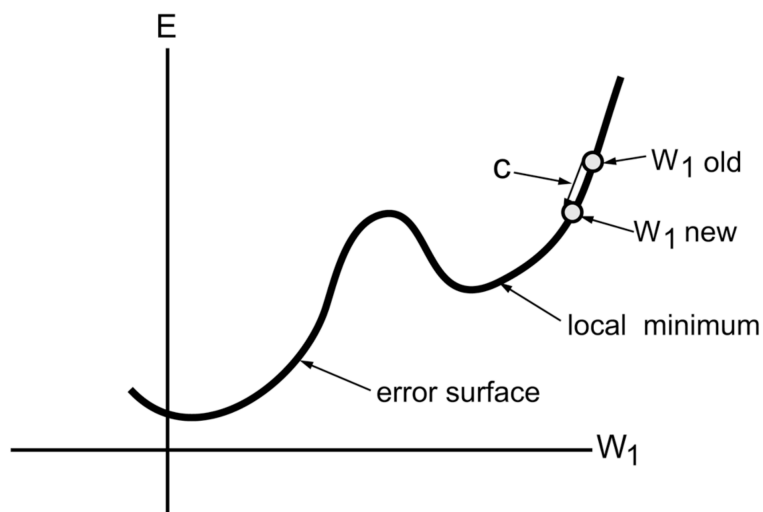
- **理论推导（非常重要！）**

- 综上，逻辑回归算法流程如下：
- 输入：特征向量集合 $\{x\}$ ，标签集合 $\{y\}$
- 输出：最优解 w
- 初始化 w_0
- 利用梯度下降法更新 w
- 直至梯度为 0 或者迭代足够多次
- 利用最优 w 来预测测试集特征向量所对应的标签，计算属于正/负类别的概率

• 思考题：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？

梯度下降

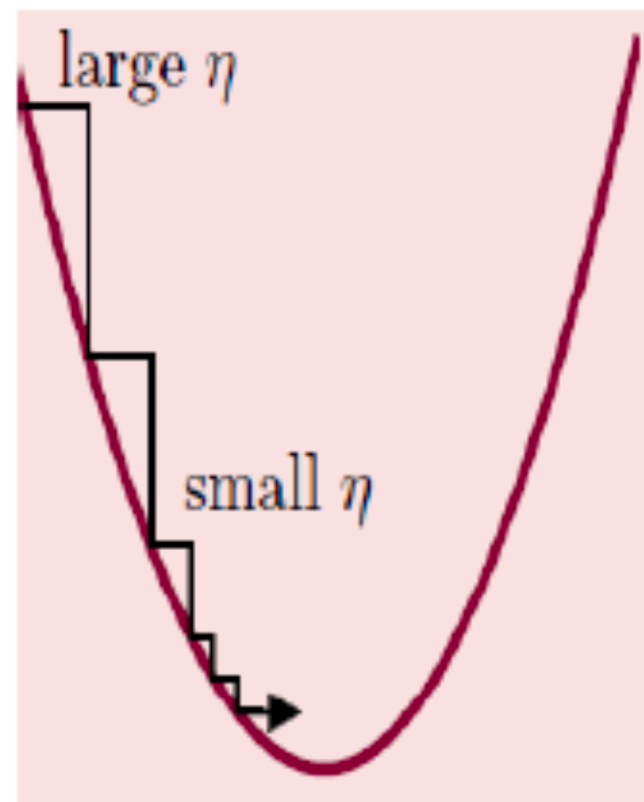
- 学习率 η
 - 也叫学习步长，计算梯度是找到了更新 w 的方向，往这个方向更新的幅度则由 η 确定
 - η 的设置会直接影响迭代解能否求解到梯度的全局最优值



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

梯度下降

- 学习率 η
 - 一般来说没有一个通用的办法和理论来确定这个学习步长，下面给出两个方法：
 - **动态学习率**：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



梯度下降

- 学习率 η

- 除此之外还可以通过**验证集**的方式确定学习率
- 设置不同的学习率，如果模型都可以较好拟合数据，选择该模型为最优模型，用于测试集的预测。

例子讲解

- 数据集如下，学习步长设为 1，只迭代 1 次
- 初始化 w_0 为 $\{1, 1, 1\}$
- 更新 w
 - 计算每一维梯度
 - 更新每一维的权重

No	Attribute 1	Attribute 2	Label
train 1	1	2	1
train 2	2	-1	0
test 1	3	3	?

例子讲解

- 计算每个样例的权重分数

- $s1 = 1 * 1 + 1 * 1 + 2 * 1 = 4$
- $s2 = 1 * 1 + 2 * 1 + (-1) * 1 = 2$

No	Attribute1	Attribute2	Label
train 1	1	2	1
train 2	2	-1	0
test 1	3	3	?

- 每一维的梯度计算

- $\nabla Cost(w_{0,0}) = \left(\frac{1}{1+e^{-4}} - 1\right) * (1) + \left(\frac{1}{1+e^{-2}} - 1\right) * (1) = -0.1372$
- $\nabla Cost(w_{0,1}) = \left(\frac{1}{1+e^{-4}} - 1\right) * (1) + \left(\frac{1}{1+e^{-2}} - 1\right) * (2) = -0.2564$
- $\nabla Cost(w_{0,2}) = \left(\frac{1}{1+e^{-4}} - 1\right) * (2) + \left(\frac{1}{1+e^{-2}} - 1\right) * (-1) = 0.0832$

例子讲解

No	Attribute1	Attribute2	Label
train 1	1	2	1
train 2	2	-1	0
test 1	3	3	?

- 更新每一维的权重

- $w_{1,0} = w_{0,0} - \nabla \text{Cost}(w_{0,0}) = 1.1372$

- $w_{1,1} = w_{0,1} - \nabla \text{Cost}(w_{0,1}) = 1.2564$

- $w_{1,2} = w_{0,2} - \nabla \text{Cost}(w_{0,2}) = 0.9168$

- 迭代 1 次，结束学习，利用 w_1 对测试集进行预测

- $$P(1|test1, w_1) = \frac{1}{1 + e^{-(1*w_{1,0} + 3*w_{1,1} + 3*w_{1,2})}} = 0.9995 > 0.5$$

优化思路（参考）

- 随机梯度下降
- 向量化运算(python/matlab)
- 标准化
- 正则化
- 动态学习率调整

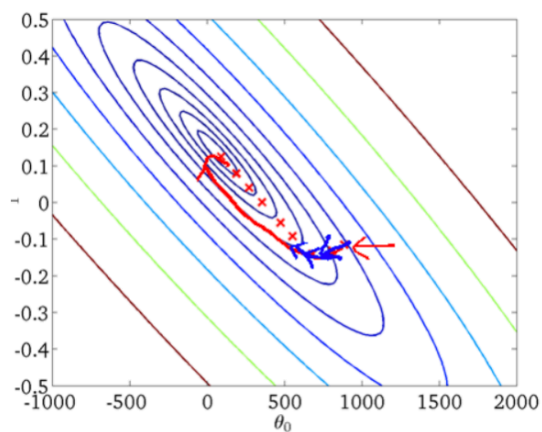
随机梯度下降

思考题：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

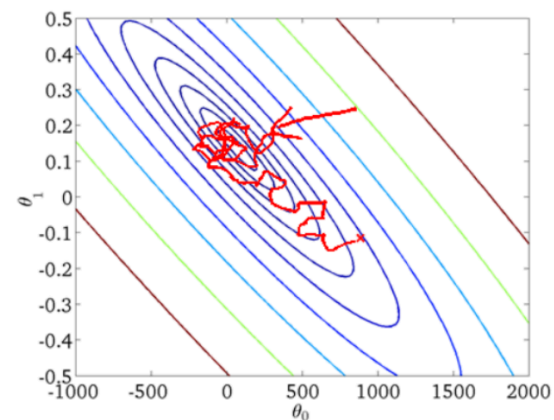
$$\tilde{\mathbf{W}}^{(j)} - \eta \sum_{i=1}^n \left[\left(\frac{e^{\tilde{\mathbf{W}}^T \tilde{\mathbf{x}}_i}}{1 + e^{\tilde{\mathbf{W}}^T \tilde{\mathbf{x}}_i}} - y_i \right) \tilde{\mathbf{x}}_i^{(j)} \right]$$



随机梯度下降

每次更新参数，考虑1个样本

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



向量化运算(python)

```
a = np.random.rand(1000000)
b = np.random.rand(1000000)

tic = time.time()
c = np.dot(a,b)
toc = time.time()

print(c)
print("Vectorized version:" + str(1000*(toc-tic)) + "ms")

c = 0
tic = time.time()
for i in range(1000000):
    c += a[i]*b[i]
toc = time.time()

print(c)
print("For loop:" + str(1000*(toc-tic)) + "ms")
```

250286.989866
Vectorized version:1.5027523040771484ms
250286.989866
For loop:474.29513931274414ms

向量化：
1.50ms

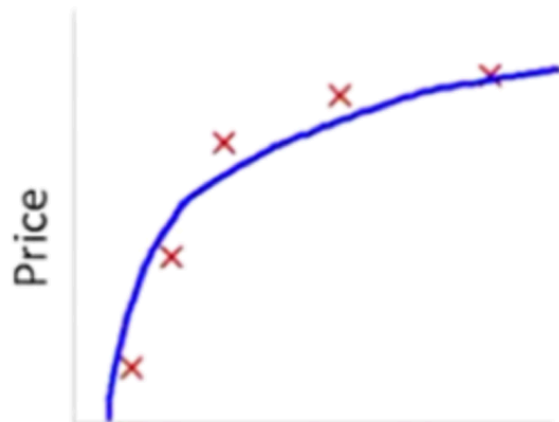
for循环：
474ms

使用matlab或python的同学
尽量不显式使用for循环可以大幅度加速计算

<https://mooc.study.163.com/smartSpec/detail/1001319001.htm>
第二周—神经网络基础

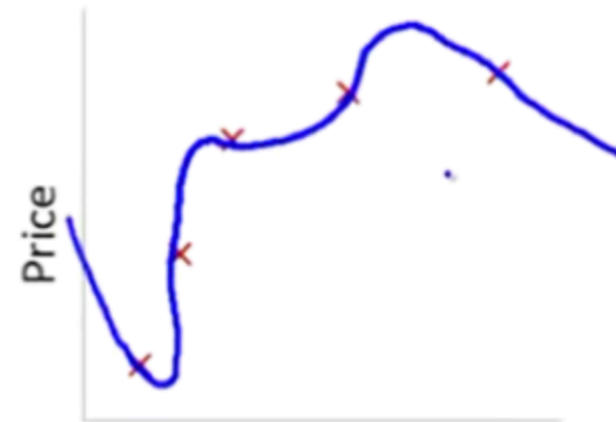
正则化

目的：减轻过拟合现象



Size of house

$$\theta_0 + \theta_1 x + \theta_2 x^2$$



Size of house

$$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$$

尽量使这两个参数的值变小

正则化

我们并不会事先知道要减小哪一个参数的值。但是，一般来讲参数的值越小，通常对应更加简单的函数，就不容易发生过拟合的问题。因此，我们通常在损失函数中惩罚比较大的参数，以得到更为简单的模型。

通过增加正则化项，惩罚较大的参数

$$-\sum_{i=1}^M y_i h(x_i) + (1 - y_i)(1 - h(x_i)) + \lambda \sum_{j=1} W_j^2$$

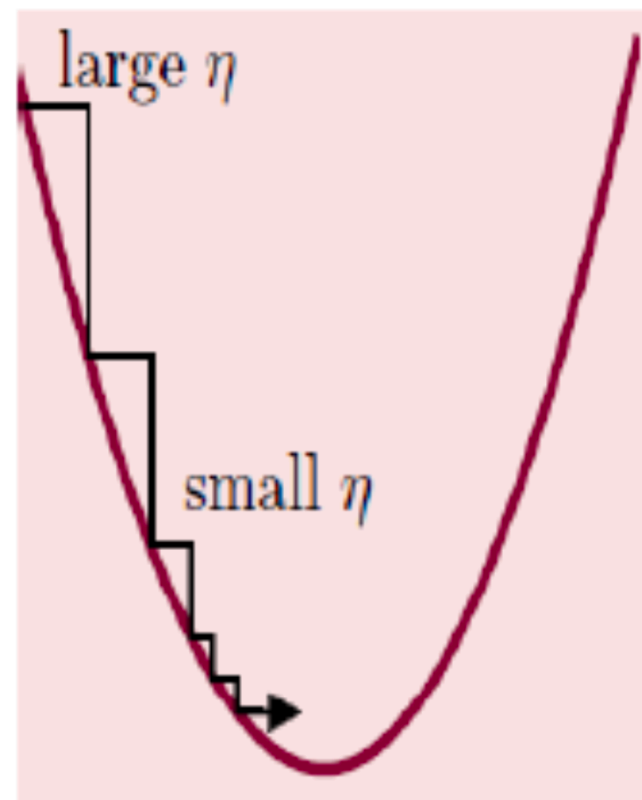
正则化项

增加正则化项后，更新公式需要重新推导，请自行推导，可作为优化。

注意：实现正则化的同学，报告中需要附上推导过程

动态调整学习率

动态学习率： 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



数据集介绍

- 训练集：8000个样本，40维度，二分类
- 验证集：自己划分
- 测试集：2000个样本，40维度，二分类

任务布置

- 必须实现梯度下降法
- 必须在给出的优化建议中任意选择一项实现
- 自己划分验证集（报告里说明是怎么分的） 调整参数
- 在测试集上预测，提交预测结果

注意事项

- 实验报告截止日期：
- **2017.11.22 晚 23:59:59 前**提交至 FTP 文件夹
- 提交文件：
 - 测试集结果：15*****_wangxiaoming.txt 每一行对应的是测试样例的标签。
 - 实验报告：15*****_wangxiaoming.pdf
 - 代码：15*****_wangxiaoming.zip 如果代码分成多个文件，最好写份 readme