## L1和L2

作用

优化方法：

L1 坐标下降，LARS角回归

## 优化方法

### BGD

即Batch Gradient Descent. 在训练中,每一步迭代都使用训练集的所有内容. 也就是说,利用现有参数对训练集中的每一个输入生成一个估计输出,然后跟实际输出比较,统计所有误差,求平均以后得到平均误差,以此来作为更新参数的依据.

#### 具体实现:

需要:学习速率 ϵ, 初始参数 θ

每步迭代过程:

1. 提取训练集中的所有内容{,…,},以及相关的输出

2. 计算梯度和误差并更新参数:

#### 优点:

由于每一步都利用了训练集中的所有数据,因此当损失函数达到最小值以后,能够保证此时计算出的梯度为0,换句话说,就是能够收敛.因此,使用BGD时不需要逐渐减小学习速率

#### 缺点:

由于每一步都要使用所有数据,因此随着数据集的增大,运行速度会越来越慢.

### SGD

SGD全名 stochastic gradient descent， 即随机梯度下降。不过这里的SGD其实跟MBGD(minibatch gradient descent)是一个意思,即随机抽取一批样本,以此为根据来更新参数.

#### 具体实现:

需要:学习速率 ϵ, 初始参数 θ

每步迭代过程:

1. 从训练集中的随机抽取一批容量为m的样本{,…,},以及相关的输出

2. 计算梯度和误差并更新参数:

#### 优点:

训练速度快,对于很大的数据集,也能够以较快的速度收敛.

#### 缺点:

由于是抽取,因此不可避免的,得到的梯度肯定有误差.因此学习速率需要逐渐减小.否则模型无法收敛

因为误差,所以每一次迭代的梯度受抽样的影响比较大,也就是说梯度含有比较大的噪声,不能很好的反映真实梯度.

#### 学习速率该如何调整:

那么这样一来,ϵ如何衰减就成了问题.如果要保证SGD收敛,应该满足如下两个要求:

而在实际操作中,一般是进行线性衰减:

其中是初始学习率, 是最后一次迭代的学习率. τ自然代表迭代次数.一般来说, 设为的1%比较合适.而τ一般设为让训练集中的每个数据都输入模型上百次比较合适.那么初始学习率怎么设置呢?书上说,你先用固定的学习速率迭代100次,找出效果最好的学习速率,然后ϵ0设为比它大一点就可以了.

## Momentum

上面的SGD有个问题,就是每次迭代计算的梯度含有比较大的噪音. 而Momentum方法可以比较好的缓解这个问题,尤其是在面对小而连续的梯度但是含有很多噪声的时候,可以很好的加速学习.Momentum借用了物理中的动量概念,即前几次的梯度也会参与运算.为了表示动量,引入了一个新的变量v(velocity).v是之前的梯度的累加,但是每回合都有一定的衰减.

#### 具体实现:

需要:学习速率 ϵ, 初始参数 θ, 初始速率v，动量衰减参数

每步迭代过程:

1. 从训练集中的随机抽取一批容量为m的样本{,…,},以及相关的输出

2. 计算梯度和误差并更新速度v和参数:

其中参数表示每回合速率v的衰减程度.同时也可以推断得到,如果每次迭代得到的梯度都是g,那么最后得到的v的稳定值为

也就是说,Momentum最好情况下能够将学习速率加速倍.一般的取值有0.5,0.9,0.99这几种.当然,也可以让α的值随着时间而变化,一开始小点,后来再加大.不过这样一来,又会引进新的参数.

#### 特点:

前后梯度方向一致时,能够加速学习

前后梯度方向不一致时,能够抑制震荡

### Nesterov Momentum

这是对之前的Momentum的一种改进,大概思路就是,先对参数进行估计,然后使用估计后的参数来计算误差

#### 具体实现:

需要:学习速率 ϵ, 初始参数 θ, 初始速率v，动量衰减参数

每步迭代过程:

1. 从训练集中的随机抽取一批容量为m的样本{,…,},以及相关的输出

2. 计算梯度和误差并更新速度v和参数:

注意在估算ĝ 的时候,参数变成了+αv而不是之前的

### AdaGrad

AdaGrad可以自动变更学习速率,只是需要设定一个全局的学习速率ϵ,但是这并非是实际学习速率,实际的速率是与以往参数的模之和的开方成反比的.也许说起来有点绕口,不过用公式来表示就直白的多:

其中δ是一个很小的常亮,大概在10−7,防止出现除以0的情况.

#### 具体实现:

需要:全局学习速率 ϵ, 初始参数 θ, 数值稳定量δ

中间变量: 梯度累计量r(初始化为0)

每步迭代过程:

1. 从训练集中的随机抽取一批容量为m的样本{,…,},以及相关的输出

2. 计算梯度和误差,更新r,在根据r和梯度计算参数更新量:

#### 优点:

能够实现学习率的自动更改。如果这次梯度大,那么学习速率衰减的就快一些;如果这次梯度小,那么学习速率衰减的就满一些。

#### 缺点:

任然要设置一个变量ϵ

经验表明，在普通算法中也许效果不错，但在深度学习中，深度过深时会造成训练提前结束。

## RMSProp

RMSProp通过引入一个衰减系数，让r每回合都衰减一定比例，类似于Momentum中的做法。

#### 具体实现:

需要:全局学习速率 ϵ, 初始参数 θ, 数值稳定量δ，衰减速率

中间变量: 梯度累计量r(初始化为0)

每步迭代过程:

1. 从训练集中的随机抽取一批容量为m的样本{,…,},以及相关的输出

2. 计算梯度和误差,更新r,在根据r和梯度计算参数更新量:

#### 优点：

相比于AdaGrad,这种方法很好的解决了深度学习中过早结束的问题

适合处理非平稳目标，对于RNN效果很好

#### 缺点：

又引入了新的超参，衰减系数

依然依赖于全局学习速率

### RMSProp with Nesterov Momentum

当然，也有将RMSProp和Nesterov Momentum结合起来的

#### 具体实现:

需要:全局学习速率 ϵ, 初始参数 θ, 初始速率v，动量衰减系数α, 梯度累计量衰减速率

中间变量: 梯度累计量r(初始化为0)

每步迭代过程:

1. 从训练集中的随机抽取一批容量为m的样本{,…,},以及相关的输出

2. 计算梯度和误差,更新r,在根据r和梯度计算参数更新量:

### Adam

Adam(Adaptive Moment Estimation)本质上是带有动量项的RMSprop，它利用梯度的一阶矩估计和二阶矩估计动态调整每个参数的学习率。Adam的优点主要在于经过偏置校正后，每一次迭代学习率都有个确定范围，使得参数比较平稳。

#### 具体实现:

需要:步进值 ϵ, 初始参数 θ, 数值稳定量δ，一阶动量衰减系数, 二阶动量衰减系数

其中几个取值一般为：δ=10−8,,=0.9,=0.999

中间变量：一阶动量s，二阶动量r,都初始化为0

每步迭代过程:

1. 从训练集中的随机抽取一批容量为m的样本{,…,},以及相关的输出

2. 计算梯度和误差,更新r,在根据r和梯度计算参数更新量:

#### 参考：

http://blog.csdn.net/u014595019/article/details/52989301

http://www.360doc.com/content/17/0323/08/1489589\_639370019.shtml

## PCA，LDA，TSNE

## 有序数组的交集（要有代码）