# 优化算法

## Mini-batch梯度下降

本次将学习优化算法，这个可以让我们的神经网络训练的更快。神经网络训练是一个需要快速迭代的过程，但是大规模的数据集往往需要很长时间训练，为此，我们需要一些能够加快神经网络训练的算法，。

那么，我们首先来探讨mini-batch梯度下降法。

我们在训练时，使用向量化技术，将训练集一次性的传入神经网络中，然后对整个网络执行梯度下降，也就是说，我们要遍历一整个数据集才能对网络执行一次梯度下降。那么，如果我们在遍历完整个训练集之前，就执行梯度下降，这会让我们的算法加快很多。我们可以把数据集分为小一点的数据集，每一部分都称为mini-batch。假设每个训练集只有1000个样本，那么对于整个数据集来说，1-1000是一个mini-batch，1000-2000是一个mini-batch，以此类推。对标签Y也要相同的处理，我们也需要分出相应标签的mini-batch。**mini-batch**的数量组成了和，我们用不同的t来代表不同的mini-batch。

接下来我们解释一下这个算法的名称，batch梯度下降来源于我们原来是将整个数据集喂入神经网络，当遍历一遍数据集后就执行一次梯度下降，所以叫做batch梯度下降。Mini-batch梯度下降就是我们这次介绍的算法，每次处理单个的mini-batch，并进行分别的梯度下降。

Mini-batch梯度下降的原理就是，我们传入一个mini-batch，并进行一次梯度下降，并计算出损失函数。在更大的范围上，我们遍历所有的mini-batch。但是如果我们仍然使用正则化，那么正则化中参数λ/m中的m，仍然是指总的样本数。

我们可以看出，mini-batch和batch梯度下降十分类似。同时，我们定义一个新的词称为epoch，我们通过mini-batch梯度下降遍历完整个数据集，就称为1epoch

## 理解mini-batch

现在，我们将要来理解mini-batch具体是如何执行的。

使用batch梯度下降时，我们每次都要遍历一整个数据集，可以预期每次迭代成本都会下降，即J是迭代成本的单调函数。如果J在某次迭代中出了问题，那就说明是你的学习率太大了。

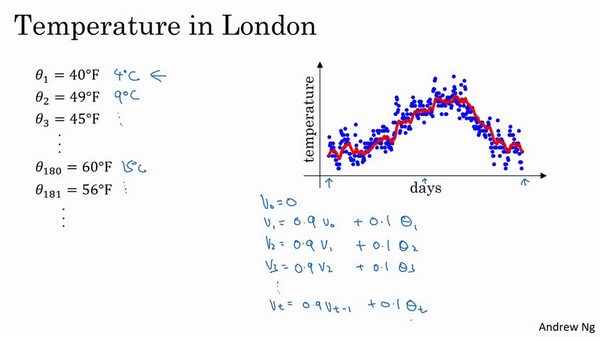
但是在mini-batch下降中可不同，如果你作出整个训练过程中成本函数的图，你可以发现，并不是每次迭代成本函数都是在下降的。因为每次进行梯度下降，你都是在用不同的mini-batch，所以成本函数并不是每次都在减少，但是总的趋势是向下减少的。

你需要决定的变量之一是mini-batch的大小，在极端情况下，这个大小等于整个训练集的大小，因此这个mini-batch就变成了batch梯度下降。

另一种极端情况下，假设mini-batch的大小变成了1，那么这个算法就变成了随机梯度下降算法。Batch梯度下降噪声较小，而随机梯度下降噪声较大。在样本数量不大的情况下，使用batch梯度下降效果很好。用随机梯度下降通常会失去向量化带来的加速。少于2000个样本时，适合使用batch梯度下降,否则，样本数目较大的话，一般的mini-batch选为64-512之间比较合适。

## 指数加权平均数

我们还有一些算法，比普通的梯度下降更加快，要学习这些算法，首先要理解指数加权平均值。

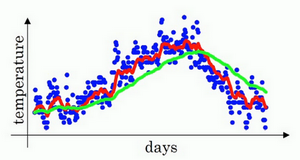
如图所示，我们有一些伦敦气温的散点图

这些数据看起来杂乱无章，但是我们是有办法计算趋势的，即计算各段时间的平均气温，我们不使用传统的计算平均数的方法，我们有更方便的做法。

我们要做的，首先就是令V0=0,然后使用0.9乘上之前的v，再加上0.1乘今天的数值。第二天，第三天，以此类推，继续这样计算。这样，我们得到某天的v就是0.9\*昨天的v加上当天温度的0.1

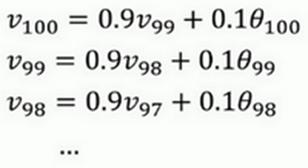
我们即可得到公式，我们计算的Vt，是天的平均温度，我们β等于0.9，我们就是计算每十天的平均温度。其计算结果和算术平均数相差无几。

我们来试试别的，将β设为更靠近1的数据，比如0.98。我们就能得到绿色的那条线。

我们设更高的β值时，曲线会变得更加平缓，因为我们多估计了几天的平均值，缺点是曲线进一步右移，因为现在平均的温度值更多，要平均更多的值，指数加权平均公式在温度变化时，适应地更缓慢一些，所以会出现一定延迟。

## 理解指数加权平均数

这是我们计算指数加权平均数的关键方程

我们进一步来分析，来理解它是如何计算出每日温度的平均值的。我们使β的0.9，如图所示，然后我们把公式展开，得到

这是一个加和于平均的过程，

指数加权平均数的好处之一在于它占用极少的内存，他不是最好最精确的做法，但是却很快，很省资源。

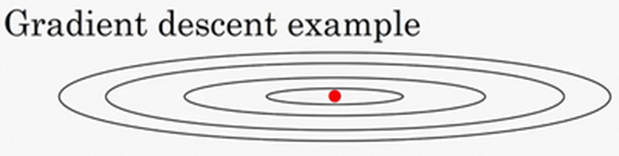
## 指数加权平均值的偏差修正

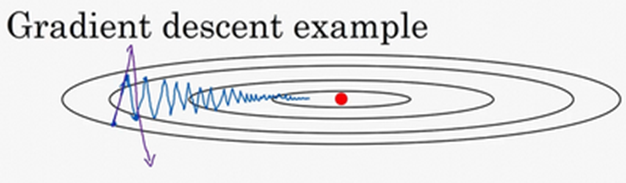
在计算指数加权平均值时，因为V0 = 0，所以当我们计算V的前几项时，通常会和真实值相差不少。有一个方法可以解决这个问题，即我们需要引入偏差修正项。也就是我们在估测的初期，不使用Vt,而使用，t是现在的天数，这样我们就能有效的进行偏差修正。不过在机器学习中，大家都不注意偏差修正，而是放过那几个初始值。

## 动量梯度下降法

有一种算法叫做动量梯度下降法，或者叫做Momentum，运行速度总是快于基本的梯度下降算法，简而言之，核心思想就是将梯度的指数加权平均数来更新梯度的权重。

假设我们的梯度下降过程图如图所示，



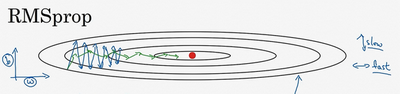
红点代表损失函数的最小值，我们随机从某个点开始梯度下降，如图所示。我们的梯度在水平方向总是朝中心点前进，但是在y轴方向却总是上下摆动。为了避免这种摆动过大，我们需要选择一个较小的学习率，而选择这种小的学习率，却会让我们的梯度下降速度大为减慢。但是这种扰动我们可以通过用指数加权平均数来更新我们的参数来使之减小。

我们要做的就是计算，接着再计算，然后再用Vdb和Vdw来更新我的参数。通过这样的办法，我们在纵轴方向的扰动就会减小，从而更快的进行梯度下降。

除了这个，我们还有其他更优化的方法。

## RMSprop

除了动量梯度下降可以加快梯度下降意以外，要有一个叫做RMSprop的算法也可以加快梯度下降，全称是root mean square prop算法。我们首先根据公式计算，，与，接着我们使用这样的公式来更新参数，

在变化较大的b方向，Sdb的值也大，而Sdw的值小，因此我们可以通过此方法加快w方向的下降，而减小b方向的变化。我们就可以得到如图所示的结果

## Adam优化方法

Adam优化方法是少有的，能被普遍使用的优化方法，Adam实际上是动量梯度下降和rmsprop的结合。我们来看看要如何使用rmsprop方法。

首先我们要初始化，，，，，接下来计算momentum，，同样的，，接着使用RMSprop计算，，。计算偏差修正项，，，，，最后更新W的公式为，，β1常用值为0.9，β2常用值为0.99

## 学习率衰减

通过这个公式更新学习率。

不过吴恩达认为，学习率衰减并不怎么重要，还不如设置一个不变的学习率。

## 局部最优问题

在神经网络发展的早期，人们总担心算法会被困在局部最优解中，但实际上这种担心是多余的。实际上，在高维空间中，局部最优的概率太低了，需要所有参数的梯度都为0，所以高维空间中大量分布的是鞍点，这会大大减慢我们的计算速度，但是仍然不会让我们陷入局部最优问题。