# 深度学习入门

## 基于Python的理论与实现

Stephen CUI

March 14, 2023

# Chapter 1

# 与学习相关的技巧

本章将介绍神经网络的学习中的一些重要观点，主题涉及寻找最优权重参数的最优化方法、权重参数的初始值、超参数的设定方法等。此外，为了应对过拟合，本章还将介绍权值衰减、Dropout等正则化方法，并进行实现。

## 1.1 参数的更新

神经网络的学习的目的是找到使损失函数的值尽可能小的参数。这是寻找最优参数的问题，解决这个问题的过程称为**最优化**（optimization）。遗憾的是，神经网络的最优化问题非常难。这是因为参数空间非常复杂，无法轻易找到最优解（无法使用那种通过解数学式一下子就求得最小值的方法）。而且，在深度神经网络中，参数的数量非常庞大，导致最优化问题更加复杂。

使用参数的梯度，沿梯度方向更新参数，并重复这个步骤多次，从而逐渐靠近最优参数，这个过程称为**随机梯度下降法**（stochastic gradient descent），简称**SGD**。

### 1.1.1 SGD

用数学式可以将SGD写成如下式：

$$\boldsymbol{W} \leftarrow \boldsymbol{W} - \eta \frac{\partial L}{\partial \boldsymbol{W}}$$

SGD是朝着梯度方向只前进一定距离的简单方法。

### 1.1.2 SGD的缺点

虽然SGD简单，并且容易实现，但是在解决某些问题时可能没有效率。

$$z = \frac{1}{20}x^2 + y^2$$

上式表示的函数是向 $x$ 轴方向延伸的“碗”状函数。

SGD的缺点是，如果函数的形状非均向（anisotropic），比如呈延伸状，搜索的路径就会非常低效。因此，我们需要比单纯朝梯度方向前进的SGD更聪明的方法。SGD低效的根本原因是，梯度的方向并没有指向最小值的方向。

### 1.1.3 Momentum

$$\boldsymbol{v} \leftarrow \alpha \boldsymbol{v} - \eta \frac{\partial L}{\partial \boldsymbol{W}} \tag{1.1a}$$

$$\boldsymbol{W} \leftarrow \boldsymbol{W} + \boldsymbol{v} \tag{1.1b}$$

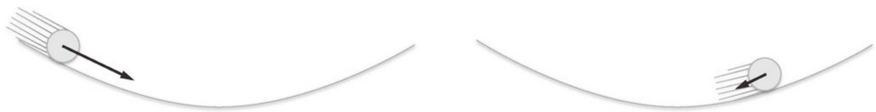

Figure 1.1: Momentum-The ball rolls on an incline

这里新出现了一个变量$\boldsymbol{v}$，对应物理上的速度。 Equation 1.1a表示了物体在梯度方向上受力，在这个力的作用下，物体的速度增加这一物理法则。如Figure 1.1所示，Momentum方法给人的感觉就像是小球在地面上滚动。

式Equation 1.1a中有$\alpha\boldsymbol{v}$这一项。在物体不受任何力时，该项承担使物体逐渐减速的任务（$\alpha$设定为0.9之类的值），对应物理上的地面摩擦或空气阻力。

### 1.1.4 AdaGrad

在关于学习率的有效技巧中，有一种被称为**学习率衰减**（learning rate decay）的方法，即随着学习的进行，使学习率逐渐减小。实际上，一开始“多” 学，然后逐渐“少”学的方法，在神经网络的学习中经常被使用。

逐渐减小学习率的想法，相当于将“全体”参数的学习率值一起降低。而AdaGrad进一步发展了这个想法，针对“一个一个”的参数，赋予其“定制”的值。AdaGrad会为参数的每个元素适当地调整学习率，与此同时进行学习。

$$\boldsymbol{h} \leftarrow \boldsymbol{h} + \frac{\partial L}{\partial \boldsymbol{W}} \odot \frac{\partial L}{\partial \boldsymbol{W}} \tag{1.2a}$$

$$\boldsymbol{W} \leftarrow \boldsymbol{W} - \eta \frac{1}{\sqrt{\boldsymbol{h}}} \frac{\partial L}{\partial \boldsymbol{W}} \tag{1.2b}$$

式中，$\odot$表示对应矩阵元素的乘法，在更新参数时，通过乘以$\frac{1}{\sqrt{\boldsymbol{h}}}$，就可以调整学习的尺度。这意味着，参数的元素中变动较大（被大幅更新）的元素的学习率将变小。也就是说， 可以按参数的元素进行学习率衰减，使变动大的参数的学习率逐渐减小。

> AdaGrad会记录过去所有梯度的平方和。因此，学习越深入，更新的幅度就越小。实际上，如果无止境地学习，更新量就会变为 0，完全不再更新。为了改善这个问题，可以使用 RMSProp方法。 RMSProp方法并不是将过去所有的梯度一视同仁地相加，而是逐渐地遗忘过去的梯度，在做加法运算时将新梯度的信息更多地反映出来。这种操作从专业上讲，称为“指数移动平均”，呈指数函数式地减小过去的梯度的尺度。

### 1.1.5 Adam

Adam是2015年提出的新方法。它的理论有些复杂，直观地讲，就是融合了Momentum和AdaGrad的方法。通过组合前面两个方法的优点，有望实现参数空间的高效搜索。此外，进行超参数的“偏置校正”也是Adam的特征。

> Adam会设置3个超参数。一个是学习率（论文中以$\alpha$出现），另外两个是一次momentum系数$\beta_1$和二次momentum系数$\beta_2$。根据论文，标准的设定值是$\beta_1$为0.9，$\beta_2$ 为0.999。设置了这些值后，大多数情况下都能顺利运行。

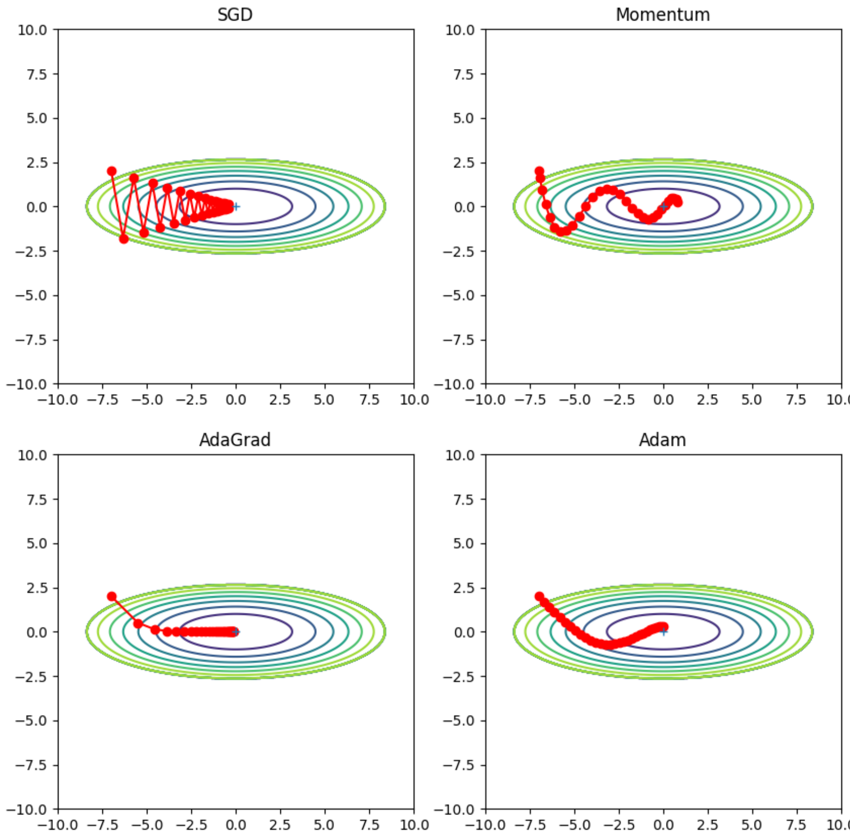


Figure 1.2: Comparison of Optimization Methods

### 1.1.6 使用哪种更新方法呢

如Figure 1.2所示，根据使用的方法不同，参数更新的路径也不同。只看这个图的话，AdaGrad似乎是最好的，不过也要注意，结果会根据要解决的问题而变。并且，很显然，超参数（学习率等）的设定值不同，结果也会发生变化。

## 1.2 权重的初始值

## 1.3 Batch Normalization

## 1.4 正则化

## 1.5 超参数的验证