早在二十世纪五十年代，深度学习就已经作为连接主义学习的一大分支发展起来，受困于连接主义学习产生的是“黑箱”模型，并不能产生明确的概念，当时的深度学习技术并未被主流人工智能研究界看好。梯度下降算法与反向传播算法的组合很好地解决了这个问题，梯度下降算法是求解最优化问题最重要、最基础的方法，近年来，随着对搜索方向和步长的不断探索，涌现出随机梯度下降算法等众多改进版本，本文对部分算法的主要研究进展进行了综述。

1951年，Robbins和Monro提出了随机逼近，并被应用于模式识别和神经网络。这种方法实在迭代过程中随机选择一个或几个梯度样本来替代总体梯度，虽然大大降低了计算复杂度，但损失了数据集中大量特征，产生的模型泛化能力较差。1958年Rosenblatt等人研制出的感知器模型采用了随机梯度下降法的思想，即每轮随机选取一个误分类样本，求其对应损失函数的梯度，再基于给定的步长更新参数。一方面，固定的步长存在着在数据鞍部震荡的可能性；另一方面，随着数据估摸的扩大，为了提高拟合效率，该方法每经过数轮迭代就需要更换步长大小，工作量较大且很难快速找到当前模型环境下的最佳值。1986年Rumelhart 等分析了多层神经网络的误差反向传播算法，该算法每次按顺序或随机选取一个样本来更新参数，这实际上是小批量梯度下降法的一个特例，但不失为一种有效的算法，使得深度学习在很多实际问题上发挥作用，导致了大量的研究者涌向了神经网络方向，也一定程度上导致了深度学习的最大局限——试错性。

随着大数据的不断普及和对优化算法的深入研究，衍生出随机梯度下降算法的许多不同版本。这些改进算法在传统的随机梯度下降算法的基础上引入了许多新思想，从多个方面不同程度地提升了算法性能。搜索方向的选取和步长的确定是梯度下降算法研究的核心。按照搜索方向和步长选取的方式不同，将随机梯度下降算法的改进策略大致分为动量、方差缩减、增量梯度和自适应学习率等四种类型。

在传统梯度下降算法的基础上添加动量项可以有效避免震荡， 加速逼近最优解。 采用动量更新策略的方法主要包括经典动量算法（Classical Momentum，CM）和Nesterov加速梯度算法（Nesterov’s Accelerated Gradient，NAG）。简单版本的随机梯度下降算法在随机取样的过程中产生了方差并且随着迭代次数的增加而不断累加，无法保证达到线性收敛。为此，研究者们相继提出了一系列基于方差缩减的随机梯度下降算法，主要包括随机方差缩减梯度算法（Stochastic Variance Reduced Gradient，SVRG）、近端随机方差缩减梯度算法、Katyusha和 MiG等。前述方法没有充分利用历史梯度信息，而增量梯度策略通过“以新梯度替代旧梯度”的方式，充分考虑了历史梯度且达到了减少梯度计算量的目的，该类型的主要算法包括随机平均梯度算法（Stochastic Average Gradient，SAG）、SAGA和Point-SAGA。在深度神经网络中，自适应学习率的随机梯度下降法通过使用反向传播所计算出的梯度来更新参数。与前三类算法不同，自适应算法在训练过程中会根据历史梯度信息，针对参数的不同分量自动调整其对应的学习率。这类算法主要包括Adagrad、Adadelta、Adam(Adaptive Moment Estimation)和Nadam（Nesterov-accelerated Adaptive Moment Estimation）等。

1. 预备知识

有了神经网络的模型之后，首先，我们选取二次代价函数来衡量网络的分类精度：



其中，C被称为二次代价函数或者均方误差、MSE，是一种经典有效的代价函数。通过梯度下降算法，能够找到一系列让代价尽可能小的权重和偏置，减小模型的误差。

在神经网络中，被正确分类的样本数量所关于权重偏置的函数并不是一个平滑的函数。大多数情况下，对权重和偏置做出的微小变动完全不会影响被正确分类的样本的数量。这会导致我们很难去解决如何改变权重和偏置来取得改进的性能。而用一个类似二次代价的平滑代价函数则能更好的去解决如何用权重和偏置中的微小的改变来取得更好的结果。只有先专注于代价函数的选择，之后才能测试分类精度。假设C是一个只有两个变量v1和v2的函数：

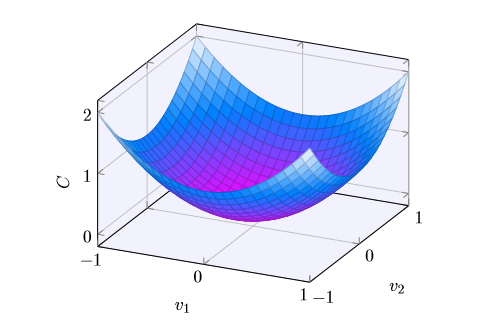


图1 二元二次代价函数图

1. 梯度下降算法

求解图1模型的梯度先将算法包括全梯度下降（Full Gradient Descent,FGD）、随机梯度下降（SGD）和小批量梯度下降（Mimi-batch Gradient Descent,Mini-batch）。

* 1. FGD

FGD以目标函数的全梯度的平均值迭代求解，首先，在v1和v2方向分别移动一个很小的量，记为v1和v2，C的变化量为：



为了使C为负，定义v为v变化的向量，即，定义C的梯度为偏导数的向量，即，记为梯度向量，从数学专业的角度解释，这是一个微分操作。

有了以上定义，C的表达式可以重写为：



上式中的把v的变化关联为C的变化，假设这时选取：



其中被称为学习速率，是一个很小的正数。所以，这保证了0，即C会一直减小，不会增加。以此来更新规则计算下一次的移动，记为：



反复持续进行上述步骤，即可持续减小C，直到获得最小值，多元代价函数同理。

在神经网络中，用权重和偏置代替变量v1和v2，即可得到更新规则：





FGD具有较快的收敛速度，但其迭代成本线性地依赖样本总数m，在求解大规模机器学习问题时，FGD运行时间长，优化效率低。

* 1. SGD

随机梯度下降算法在每轮更新参数时，随机选取一个样本或子成分计算梯度，并以此梯度作为全局梯度的估计值。这有助于加速梯度下降，进而加速学习过程。

但由于数据集通常存在噪声，使用SGD一般难以沿着最佳的更新方向逼近最优参数。如图2所示，SGD与FGD相比较，在每次迭代中，FGD都能改变目标参数，而SGD则震荡地收敛至最优解。

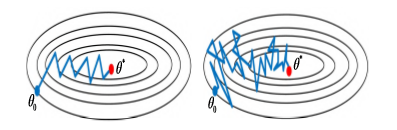


图2 FGD与SGD优化轨迹对比图

* 1. Mini-batch

小批量梯度下降在一定程度上兼顾了FGD与SGD的优点。其思想就是在每轮迭代中随机抽取若干个样本，以这些样本的梯度平均值作为本轮全局梯度的估计值。将随机选取的训练输入标记为，，…，，并把它们称为小批量数据（mini-batch），那么有：



即：



得到权重和偏置的更新规则为：





1. Hessian技术

当直接思考最小化代价函数C的抽象问题，Hessian技术有时能够带来比小批量随机梯度下降算法更好的效果。在数学上，海森矩阵(Hessian matrix或Hessian)是一个自变量为向量的实值函数的二阶偏导数组成的方块矩阵, 此函数如下：



如果f的所有二阶导数都存在，那么f的Hessian矩阵即：



其中，即H（f）为：



设C是多个参数的函数，，所以C=C（w），则代价函数可以在点w出被近似为：



可以用Hessian矩阵压缩为：



其中H是矩阵形式的Hessian矩阵，假设通过丢弃较高阶项来近似C，可得：



根据该近似表达式，可知从w移动到可以显著地降低代价函数的值。所以在神经网络中，我们通过重复地使用改变量来改变w。

在理论上和实践中的结果都表明Hessian技术比标准的梯度下降方法收敛速度更快，通过引入代价函数的二阶变化信息，可以让Hessian方法避免在梯度下降中常碰到的多路径问题。但由于在实践中权重和偏置的网络对应的Hessian矩阵过于庞大，导致计算极其困难，难以用于大规模机器学习问题。

下图为Hessian技术与梯度下降方法求解最优化问题的收敛速度对比图，红色曲线为Hessian技术迭代求解，绿色曲线为利用梯度下降法求解：

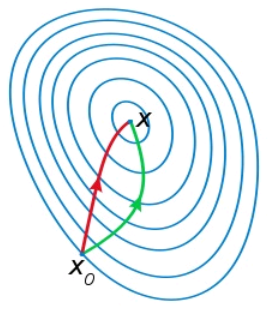


图3 Hessian与FGD优化轨迹对比图

1. 基于Momentum的梯度下降

Momentum技术借用了物理中的动量概念，它模拟的是物体运动时的惯性，即更新的时候在一定程度上保留之前更新的方向，同时利用当前batch的梯度微调最终的更新方向。这样一来，可以在一定程度上增加稳定性，从而学习地更快，并且还有一定摆脱局部最优的能力。

引入速度变量，其中每一个对应变量，则可将梯度下降更新规则改为：





Momentum算法会观察历史梯度，若当前梯度的方向与历史梯度一致（表明当前样本不太可能为异常点），则会增强这个方向的梯度，若当前梯度与历史梯方向不一致，则梯度会衰减。一种形象的解释是：我们把一个球推下山，球在下坡时积聚动量，在途中变得越来越快，可视为空气阻力，若球的方向发生变化，则动量会衰减。该模型如下图所示：

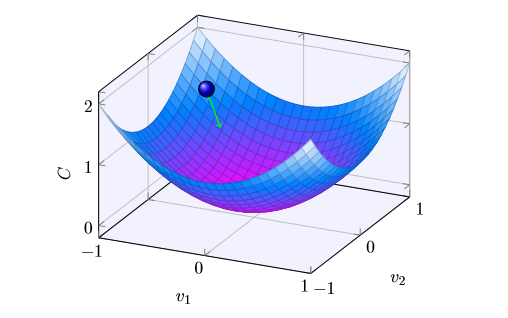


图4 梯度下降物理模拟图

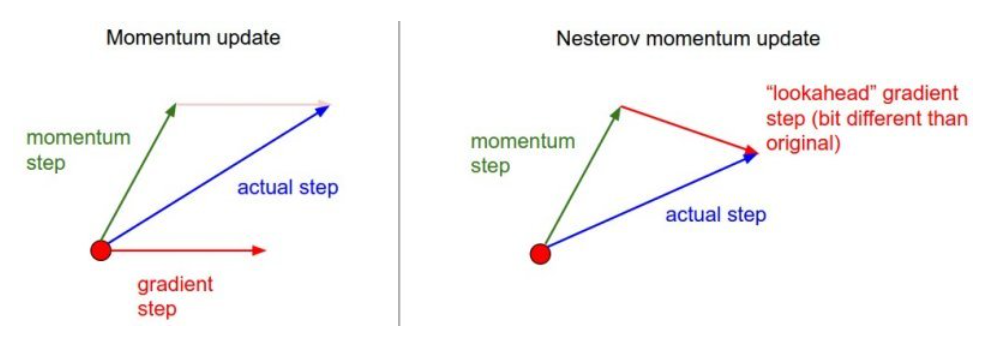
1. Nesterov Momentum

在小球向下滚动的过程中，我们希望小球能够提前知道在哪些地方坡面会上升，这样在遇到上升坡面之前，小球就开始减速。这方法就是Nesterov Momentum。并且，在实践中Nesterov Momentum也比单纯的 Momentum 的效果好，其参数更新公式为：





其核心思想是：在momentum方法中，如果只看项，那么当前的w经过 momentum的作用会变成。因此可以把这个位置看作是当前优化的一个展望位置。所以，可以在求导, 而不是原始的w。

图5 Nesterov Momentum

1. Adagrad

在大规模机器学习中，为了减少震荡、提高优化效率，每经过数轮迭代就需要更换一个较小的学习速率。手动调节学习率工作量较大且很难快速找到当前模型环境下的最佳值。若设置的学习率过小，会使得优化进程缓慢；若学习率过大，会导致震荡且难以逼近最优解甚至逐渐远离最优解。Adagrad算法能够在训练中自动地对学习速率进行调整，对于出现频率较低参数采用较大的更新；相反，对于出现频率较高的参数采用较小的更新。因此，Adagrad非常适合处理稀疏数据。

为了描述方便，引入以下符号：表示第t轮迭代时的第k个参数分量，其中；表示第t轮迭代时的梯度，即；表示对应的梯度分量，即。Adagrad的参数分量更新公式为：



其中，表示对前t轮的第k个梯度分量进行平方和累加；是全局学习速率；是一个很小的正数，其作用是避免分母为零；可将看作自适应调节的学习速率。Adagrad的缺点是在训练的中后期，分母上梯度平方的累加将会越来越大，从而梯度趋近于0，使得训练提前结束。

1. Adam

Adam(Adaptive Moment Estimation)是另一种自适应学习率的方法，他结合了矩估计思想，通过计算并修正每轮梯度的一阶矩和二阶矩来动态调节学习率。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，能够有效判断样本是否为噪声点或此轮更新不具有价值，使得参数比较平稳。分别计算的一阶矩和二阶矩的估计量和：



其中，, ∈[0,1) 是衰减常数。建议设置为= 0.9，= 0.999，初始化和均为d维零向量。分别对、的偏差进行修正：



Adam在结合了Adagrad善于处理稀疏梯度的优点的同时，也能够处理非平稳目标的优点，更适用于大数据集和高维空间。所以在实际应用中 ，Adam为最常用的方法，可以比较快地得到一个预估结果。