单位代码：11414
学　　号：2016011871

中国石油大学（北京）
CHINA UNIVERSITY OF PETROLEUM , BEIJING

本科生毕业论文

题　　目　深度学习中的梯度下降优化算法对比分析研究

学院名称　理学院

专业名称　数学与应用数学

学生姓名　杜婷

指导教师　孙娜老师

起止时间：2019 年 10 月 22 日 至 2020 年 5 月 21 日

摘　要

深度学习作为人工智能的重要研究领域，现今仍在不断发展中，而在深度学习所需的优化算法中，梯度下降算法是最重要、最常用的算法[1]，该算法从出现以来就在不断地演变发展，目前已经出现了很多的改进算法。传统的梯度下降算法存在两大挑战：一是收敛速度上难以达到最佳，二是对训练的最终结果有巨大影响的学习率的选择，其中包括局部极小值以及鞍点等问题[2]。于是本文通过研究有关的文献资料分析各个算法的特点，基于这两个挑战分别对改进算法进行分析，同时结合实际的应用对比分析不同算法的优劣，主要是进行理论上的研究，然后利用 Python 在深度学习中的应用来编程实现部分算法的图像可视化，直观地展现常见的流行的算法的区别，通过最终的图像得到:三个自适应学习优化器 Adagrad、RMSprop 和 Adadelta 的下降速度明显快于 SGD 算法，其中 Adagrad 与 RMSprop 算法齐头并进，比 Adadelta 算法要快，两个动量优化器 Momentum 和 NAG 算法初期下降慢，但后期速度越来越快，而比较新的 Adam 算法相比之下比上述算法更快；面对鞍点，三个自适应学习率算法以及 Adam 算法没有进入鞍点，两个动量优化算法进入鞍点后抖动一会儿又离开了，而传统的 SGD 算法没能离开鞍点。之后本文结合目前流行的并行和分布式框架对梯度下降算法的并行和分布式进行了研究，分析了其发展前景，最后通过这些研究提出了一些进一步改进算法的思路，比如可以在学习率的自适应上结合一些函数进行变换从而使得训练结果更佳。

关键词：梯度下降算法；SGD 算法；Adam 算法；自适应学习率；并行和分布式算法

Contrast and Analysis of Gradient Descent Optimization Algorithm in Deep Learning

ABSTRACT

As an important research field of artificial intelligence, deep learning is still evolving today. Among the optimization algorithms required for deep learning, the gradient descent algorithm is the most important and commonly used algorithm[1]. The algorithm has been continuously evolving since its emergence,and nowadays there have been many improved algorithms. The traditional gradient descent algorithm has two major challenges: one is that it is difficult to achieve the best convergence speed, and the other is that the choice of learning rate that has a huge impact on the final result of training, including local minimum and saddle points[2]. So this paper analyzes the characteristics of each algorithm by studying relevant literature,and analyzes the improved algorithm based on these two challenges, and analyzes the advantages and disadvantages of different algorithms with actual applications. The main purpose is to conduct theoretical research and then use Python which is applied in deep learning to visualize some algorithms, and intuitively show the differences between common popular algorithms. The final images show that the decline rate of the three adaptive learning optimizers (Adagrad, RMSprop and Adadelta) is significantly faster than SGD Algorithm,and Adagrad and RMSprop algorithms go hand in hand, faster than Adadelta algorithm;two momentum optimizers (Momentum and NAG algorithms) decline slowly at the beginning, but later the speed is getting faster and faster; by comparison,the newer Adam algorithm is more fast than the above algorithms;about the saddle point, the three adaptive learning rate algorithms and the Adam algorithm do not enter the saddle point, and the two momentum optimization algorithms enter the saddle point and get away for a while, while the traditional SGD algorithm fail to leave the saddle point. Afterwards, this paper studies the parallel and distributed gradient descent algorithm in combination with the popular parallel and distributed frameworks, and analyzes its development prospects, and finally puts forward some ideas for further improvement of the algorithm through these studies, such as adding some functions to transform the self-learning rate to make the training result better.

Key Words：Gradient descent algorithm；SGD algorithm；Adam algorithm；Adaptive learning rate；Parallel and distributed algorithms

目　录

摘　要.....I
ABSTRACT.....II
第 1 章　绪论.....1
1.1　课题研究的背景、意义及研究现状.....1
1.1.1　研究背景及意义.....1
1.1.2　国内外研究现状.....2
1.2　本文研究内容.....3
1.3　本文研究内容构架.....3
第 2 章　深度学习中的梯度下降算法的理论基础.....5
2.1　深度学习.....5
2.1.1　特征提取.....5
2.1.2　与浅层学习的关系.....5
2.1.3　基本思想.....6
2.1.4　训练过程.....7
2.1.5　挑战与机遇.....7
第 3 章　基本梯度下降算法.....9
3.1　基本概念.....9
3.1.1　梯度下降.....9
3.1.2　算法代数方式描述.....9
3.1.3　算法矩阵方式描述.....10
3.2　基于样本使用量的算法变式.....11
3.2.1　批量梯度下降算法.....11
3.2.2　随机梯度下降算法.....11
3.2.3　SAG（Stochastic Average Gradient）.....12
3.2.4　小批量梯度下降算法.....13
3.2.5　存在的问题.....13
第 4 章　常见梯度下降优化算法.....15
4.1　基于动量的改进算法.....15
4.1.1　动量梯度下降法（Momentum 算法）.....15
4.1.2　NAG（Nesterov Accelerated Gradient）.....16
4.1.3　SVRG.....18
4.2　基于学习率问题的改进算法.....19

4.2.1 Adagrad 算法.....19
4.2.2 Adadelta 算法.....19
4.2.3 RMSprop 算法.....21
4.2.4 Adam 算法.....22
4.2.5 AdaMax.....23
4.2.6 Nadam.....24
4.2.7 其他有关研究点.....25
第 5 章 可视化算法与总结.....27
5.1 重要的 SGD.....27
5.2 动量的引入.....29
5.3 各优化器可视化.....30
5.3.1 鞍点处的表现.....30
5.3.2 普通曲面上的表现.....32
第 6 章 并行和分布式研究.....35
6.1 基本概念.....35
6.1.1 参数服务器.....35
6.1.2 同步更新.....36
6.1.3 异步更新.....36
6.2 一些流行的算法和框架.....36
6.2.1 Hogwild!.....36
6.2.2 BigDL.....37
6.2.3 Downpour SGD.....37
6.2.4 小结.....38
第 7 章 总结与展望.....39
7.1 本文总结.....39
7.2 未来展望.....39
参 考 文 献.....40
致　　谢.....44

第 1 章　绪论

人工智能的发展需求使得机器学习也在逐步发展完善，机器学习中一个重要的研究领域是优化算法，随着所涉及模型越来越复杂，在对模型参数进行优化中，梯度下降算法成为最常采用的方法之一。为了更好地满足实际中模型复杂度提高以及数据大规模化等问题的需求，梯度下降算法也在不断地改进和发展。但是在实际应用中，各种改进算法如何取舍仍是难以明确的，故本文基于已有的相关研究，对常用的流行的各种梯度下降算法进行对比分析，结合理论研究和实际应用，对各算法的收敛速度、适用范围、存在的优缺点等内容进行对比，给出相应的结论；同时本文会利用 Python 进行图像可视化，将部分算法的区别通过图像直观展现，这样也可以更方便有关人员的使用。虽然目前对于梯度下降算法的研究已经有很多，但通过具体的对比分析描述这些算法的文章很少，从而难以满足实际中有关研究的需求，因此，本文通过研究分析国内外各研究内容来对比分析各个算法，以供有关人员使用。同时，现今为了适应大数据时代，并行和分布式深度学习算法和框架的有关研究越来越多，本文选取了比较流行的 Hogwild![3]、BigDL[4]-[5]和 Downpour SGD[6]算法进行研究分析，对如今的并行和分布式算法的发展进行剖析，但是限于设备资源的欠缺，没能进一步对并行和分布式框架进行实践，期望可以在以后的工作中实现。

1.1　课题研究的背景、意义及研究现状

1.1.1　研究背景及意义

基于目前流行的人工智能和大数据，深度学习作为其迅速发展的重要部分已成为研究应用的热门方向。深度学习被广泛运用于图像搜索、自然语言处理、计算机视觉等领域[7]-[10]，并且随着计算机处理能力的提升，深度学习的研究更加深入并且获得了巨大的进步。随着它的迅速发展，各种优化算法也在不断发展中，比如最小二乘法、牛顿法、梯度下降法等[11]，而梯度下降法作为求解最优化问题的最基础、最重要的方法，其应用是最广泛的。于是为了满足研究应用的需要，传统的梯度下降算法的各种改进方案不断涌出，这些改进算法迎接传统的算法中存在的挑战（主要是学习率选择和收敛速度慢的困难），通过适当的变形来弥补一些缺陷，从而使得梯度下降算法这个大家族更加适用于模型具有不同复杂度的问题。同时最近几年，在大数据时代下，对于梯度下降算法的并行和分布式研究越来越多[10]，这让梯度下降法可以适用于更加大规模的数据和模型参数增多的情况。

梯度下降算法对于理论数学和应用数学而言都是一项重要的内容，故本课题

针对深度学习中的梯度下降优化算法的对比分析研究在理论和实际应用上都具有一定的意义。一方面，传统的梯度下降算法在经过不断地改进调整中需要结合理论数学的相关知识，其提升也是数学知识界的一个进步，展现了数学的可发展性和开创性，不仅仅对于梯度下降算法而言是一个改进，此类思想也可运用于其他算法的理论研究中，有助于数学理论知识的发展进步；另一方面，梯度下降算法作为深度学习的一个重要的研究方向，对其进行改进优化也是对深度学习甚至人工智能的一个提升，该算法的改进有助于在实际运用时满足不同的需求，使得训练数据时得到的运行结果更加符合实际情况，从而使结果更精确。因此，对这些梯度下降算法进行对比分析研究，有助于有关人员查阅从而选择最佳的算法来应用于自己的研究应用中，会使相应的研究应用更有意义或价值。

1.1.2 国内外研究现状

作为机器学习一个最重要的分支，深度学习源于人工神经网络模型的学习研究，近年来发展迅猛，在国内外都引起了广泛的关注，而梯度下降算法作为其最常用的优化算法也随之在不断地发展进步中。

早在 18 世纪就已经出现了关于梯度下降算法的研究，当时对其研究还是比较少的，传统的梯度下降算法适合解决的问题复杂度低并且对模型函数要求比较高。于是首先基于训练中样本量的使用，得出了 3 种基本的梯度下降算法[12]，第一个就是最原始的算法 BGD，该算法在训练中使用所有的样本量；其次是在每次迭代中只使用一个样本量的 SGD 算法；另一个是综合以上两个算法得到的 MBGD 算法，但这三种算法并没有从根本上解决最初的梯度下降算法的问题，于是成为后来各种改进算法的基础。

对于收敛速度这个问题，1998 年 Yann LeCun 等人[13]引入动量的概念，提出了动量法这个改进方案，该算法是针对 SGD 算法中出现的摇摆以及收敛速度较慢的问题提出来的；随后 Nesterov[14]在此基础上提出了 NAG 算法，该算法可以更加“智能”地进行训练；后来在 2018 年景立森、丁志刚、郑树泉, 等人[15]基于 NAG 算法对 BP 神经网络进行了进一步的改进，提出了针对隐层神经元的一种黄金比例动量方案，使得梯度计算进一步加快，并将其应用于 MNIST 手写字体识别中，获得了比较好的效果。

2013 年 3 月 Nicolas Le Roux 等人[16]提出了一种加速版本的 SGD 算法－－SAG 算法，该算法利用空间换取时间的思想加速了算法的训练。后来 Rie Johnson 等人[17]在此基础上又提出了改进的随机方差消减梯度法（Stochastic Variance Reduction Gradient，SVRG），这种方法进一步提高了 SGD 的收敛性。2018 年王建飞、亢良伊等人[13]又在 SVRG 的基础上提出了 topkSVRG 算法，实验证明该算法收敛性更高并且具有线性收敛性。

而在改进方案中，更多的是对于学习率的改进策略。2011 年 John Duchi 等人[18]提出了 Adagrad 算法，该算法让参数可以自适应学习率，从而可以更好地调整学习速度；随后 2012 年 Mattew D.Zeiler 等人[19]-[20]在此基础上提出了 Adadelta 算法，该算法已经可以不用设定默认的学习率，同年 Geoff Hinton[6]在他的 Coursra 课程中提出了 RMSprop 算法，这个算法可以看做是 Adadelta 的一个特例，可见对于梯度下降算法的研究真的是层出不穷；之后 Diederik P.Kingma 等人[21]结合动量和 Adadelta 算法的思想提出了 Adam 算法，该算法比已有的自适应学习率方法更加有效，之后在这些算法的基础上对于学习率的研究还有很多。

后来随着数据的大规模化和模型复杂度的提高，对于梯度下降算法的研究扩展到了并行与分布式研究，John 等人[11]提出了最早的并行 SGD 方法——Round-robin 方法[45]，该算法通过共享内存中的模型实现串行更新，更加适用于高维数据集。后来，Niu 等人[3]在此基础上提出了 Hogwild!算法，此算法是无锁的并行 SGD 算法，主要用于稀疏数据集的问题；Zhao 等人[22]在 SVRG 的基础上提出了并行的无锁的 AsySVRG，该算法可以达到线性收敛，效果更加好。2010 年 Martin 等人[23]提出了基于 MapReduce 的分布式 SGD 算法，该算法可以减少训练时间，后来 Meng 等人[24]-[25]又基于 Spark 提出了 MLlib 库，其实现了 MBGD 算法的分布式；类似的分布式算法改进还有很多，这些算法更加适用于大数据下的问题，很多公司（例如 Google、百度等）还成立了专门的研究团队开发更合适的深度学习库，例如 Google 开发的 BigDL 库[4]-[5]等。随着计算机运算能力的提升，基于这些改进的优化算法以及关于并行和分布式研究，未来深度学习将会更加高深从而推动人工智能等科技的发展。

1.2 本文研究内容

本课题旨在通过对比分析研究展现不同的优化梯度下降算法的具体特点，对比其推导准则以及应用条件等来评估不同方法的优劣好坏，有助于相关人员使用和其他有关的研究。

该课题的研究内容主要分为三点：(1) 研究分析最常用的优化算法的具体形式，包括这些算法在解决不同挑战时的动机以及如何得到更新规则的推导形式，在理论层面上熟悉这些优化算法。(2) 简单讨论如今较热门的并行和分布式环境中优化梯度下降的算法和框架，对其有具体系统的研究认识，结合有关的应用实例学习及分析。(3) 在对比分析过程中研究不同改进方案的优缺点，并同时思考对优化梯度下降算法有用的一些其他策略以对梯度下降算法未来的发展趋势进行展望。

1.3 本文研究内容构架

作为深度学习中常用的优化算法之一，梯度下降算法已经有了很多的改进方案，这些算法主要针对传统的梯度下降算法中的收敛速度和学习率等方向进行改进的。本文的主要工作就是对这些改进算法进行研究并且对比分析其优劣。

第一章：绪论。这一章节介绍了本课题的研究背景和意义，体现出梯度下降算法的重要性，同时叙述了该课题的国内外研究现状，为后文做好铺垫。

第二章：本章节主要介绍深度学习的有关知识，包括其基本思想以及未来发展的机遇与挑战等，这更加体现了本研究的重要意义。

第三章：这一章节主要介绍基本的四种梯度下降算法，包括 BGD、SGD、SAG 和 MBGD，这四种算法的创新点比较小，但是具有不同的特点，其中的 SGD 算法的应用最为广泛，同时本章还介绍了传统的梯度下降算法中存在的困难，为后续的改进算法提供了基础。

第四章：针对第三章中介绍的困难，分别从动量和学习率两个角度分别叙写比较流行的梯度下降算法变式，同时还提出了一些其他有关研究点，可供后续改进算法使用。

第五章：本章利用 Python 进行编程对部分算法实现图像可视化，从而直观展现各个算法的区别。

第六章：本章节对现今流行的并行和分布式算法进行研究，首先是介绍了并行和分布式的相关概念，随后选取了几个比较热门或者有发展前景的算法和框架进行了介绍，同时对该研究的未来发展进行了分析。

第七章：最后一个章节对本文工作进行了总结，同时对未来的工作进行了一些展望。

第 2 章　深度学习中的梯度下降算法的理论基础

2.1　深度学习

2.1.1　特征提取

数学上，特征是指经典特征函数在局部域上的一种推广。机器学习中，特征是一个重要的概念，是机器学习的原材料，与最终训练出的模型密切相关。就如向量中确定基一样，提取特征[26]-[27]是进行机器学习的基础，而对许多机器学习而言，这并不是一件简单的事情。对一些复杂的问题，需要通过人工的方式进行特征提取，但这要投入非常多的时间和精力。例如，假设想要从一张图片中识别汽车，而汽车是有轮子的，那么就需要在图片中提取“轮子”这个特征，但实际操作是非常难的，因为图片中的汽车可能出现被遮挡、有反光等情况。

由于人工提取特征会出现很多的不确定因素，于是就有了深度学习的出现，深度学习解决的核心问题就是自动地从简单的特征中提取更复杂的特征，从而更加简单且有效地解决问题，而这也是深度学习与传统机器学习的差异，为之后人工智能的发展开辟了全新的道路。

对每一个实际的问题和方法，特征越多，所给信息就越多，从而会提升准确性，但是特征多意味着计算的时间复杂度和空间复杂度越大，可以用于训练的数据在每个特征上就会变得稀疏，从而产生其他问题，所以并不一定特征越多越好。

尤其是在高维空间中的好的特征是不易直接看出来的，因此如何提取好的特征成为了一个难题，而特征又是后续工作的决定性因素，所以对于这个问题，机器学习产生了一个专门的领域——特征工程，研究如何提取特征的方法论；对不同的数据、不同的问题，好的特征的定义是不同的，所以就产生了不同的特征提取算法，常见的算法有 LBP 算法（局部二值模式）、HOG 特征提取算法（方向梯度直方图）、SIFT 算子（尺度不变特征变换）等[28]，如今，卷积作为机器学习中的“特征提取器”在解决实际问题中广泛运用。

2.1.2　与浅层学习的关系

20 世纪 80 年代末期，深度学习之父 Geoffrey Hinton[29]提出的用于人工神经网络的反向传播算法（BP 算法）掀起了机器学习的研究浪潮，这种方法比起以往基于人工规则的系统在很多方面都展现出优势，随后各种浅层学习的模型被提出，例如支持向量机（SVM,Support Veector Machines）、Boosting、最大熵方法（如 LR,Logistic Regression）等[30]，但这些模型实际上是最多只含有两层隐层节点的浅层模型，当所研究的问题复杂度增加时就会出现很多局限性。

2006 年 Geoffrey Hinton 和他的学生 Salakhutdinov[30]在世界顶级学术刊物《科

学》上发表的文章中正式提出了深度学习的概念，从而开启了机器学习的第二次浪潮－－深度学习的研究。深度学习的实质是通过构建具有很多隐层的机器学习模型和海量的训练数据来学习更加有用的特征，从而最终提升分类或者预测的准确性。区别于浅层学习，深度学习更加强调模型结构的深度，更加突出特征学习的重要性，这样更加适合大数据时代下各种问题的解决。

2.1.3 基本思想

机器学习是人工智能领域的一种方法，而神经网络和深度学习又是机器学习的算法，深度学习可以看作是神经网络的发展，神经网络一般有输入层、隐藏层、输入层，而隐藏层大于 2 的神经网络叫做深度神经网络，深度学习就是采用像深度神经网络这种深层架构的一种机器学习方法。

假设有一个系统 S, 它有 n 层（$S_1, \cdots S_n$），其输入是 I、输出是 O，可表示为：

$$I \geq S_1 \geq S_2 \geq \cdots S_n \geq 0 \tag{2.1}$$

若输出 O 等于输入 I，意味着输入 I 经过这个系统之后无任何的信息损失，与原有信息保持一致，这意味着输入 I 经过每一层 S_i 都没有任何的信息损失，即在任何一层 S_i，它都是原有信息（即输入 I）的另外一种表示形式，但这在理论上是不符合实际的，因为信息论中的信息处理不等式表示信息会逐层丢失。而对于深度学习，它是自动地学习特征，假设有输入 I（例如图像或文本），通过调整一个系统 S（有 n 层）中的参数，使得它的输出 O 仍然是输入 I，那就可以自动地获取得到输入 I 的一系列层次特征，即 $S_1, \cdots, S_n$。对于深度学习而言，其思想就是滤过多个层，也就是说这一层的输出作为下一层的输入，通过这种方式，就可以实现对输入信息进行分级表达了。另外，输出 I 严格地等于输入 O 的假设太严格，可以略微地放松这个限制，例如只要使得输入与输出的差别尽可能地小即可，这个放松会导致另外一类不同的深度学习方法。

深度学习有着类似于 Dropout[31]-[32]的形式，Dropout 是一种正则化形式，它本质上体现着集成学习的思想。在集成学习中，采用一些较弱的分类器分别训练数据，也可将各个分类器进行组合产生更强的分类器，这与深度学习中的数据处理和特征提取等有着相同的思想，该思想在深度学习中得到非常好的应用。

如今深度学习已经取得了非常多的成功，深度神经网络的层数逐渐增加，更加适用于包含复杂的层次关系的图像和文本等问题的研究[9]，正是这种深层结构提取文本、图像、语言等原始数据抽象的本质特征，体现了深度学习的逐步抽象的特征层次结构的思维。提取本质的、好的特征，舍弃非本质的特征，这个过程

本来就是一个逐渐抽象的过程。

深度学习算法的另一个基本思想是每一层进行非线性变换[27]。大数据时代下，数据的复杂性提高，并且在生成架构中的层数增加，从而构造的非线性变化就越复杂。比如，利用深度学习算法进行人脸识别[33]，在第一层需要学习边缘特征，在第二层就需要组合这些边缘以进行更加复杂特征的学习，以此类推得到最终的人脸图像。因此，非线性思维在深度学习中得以充分的体现。

当然，深度学习所包含的思想远不仅这些，随着大数据的发展和问题的复杂性的提升，深度学习也在不断地发展变化中。

2.1.4 训练过程

深度学习适用于研究多层神经网络问题，而由于层数很多，如果对所有层同时进行训练，那么时间复杂度就会很高，而如果每次训练一层，那么每一次产生的误差就会逐层推进，导致最终的模型偏差很大，从而使得结果具有非常大的不准确性，所以有关人员对如何训练进行了进一步的研究分析。2006 年，Hinton[34]提出了一个有效方法，该方法用于在非监督数据上建立多层神经网络，将解决问题分为两步，第一步每次训练一层网络，第二步再进行调优，调优所使用的方法是 wake-sleep 算法。该算法分为 wake 和 sleep 两个阶段，wake 阶段也就是认知过程，通过外界的特征和认知权重来产生每一层的抽象表示，并且使用梯度下降修改层间的生成权重；而 sleep 阶段即为生成过程，通过前一阶段得到的结果生成底层的状态，同时修改层间向上的权重。这个训练过程在后面的研究中得到了具体的运用，使得深度学习更加具有系统性和完整性。具体而言，深度学习的训练过程如下：首先进行数据预处理；然后采用自下上升的非监督学习（即从底层开始，一层层地向顶层进行训练），具体的，先用无标定数据训练第一层以得到第一层的参数，由于模型性能的限制和稀疏性的约束使得训练可以得到比输入更具有表示能力的特征，以此类推，学习得到的第 n-1 层的输出将作为第 n 层的输入，由此就可以分别得到每一层的参数；接下来就是自顶向下的监督学习（即通过带标签的数据进行训练，在此过程中误差会进行传输，从而可以对结果进行微调），基于上个阶段得到的每一层的参数进一步的调整整个模型的参数，从而使得结果达到最优。

2.1.5 挑战与机遇

人工智能越来越热，而深度学习作为该领域中受到大家密切关注的领域，其研究与发展一直都在持续进行中，无论是理论还是应用方面都在不断地走向成熟，但是深度学习的发展仍然面临着很多的挑战与机遇。

深度学习是近 20 年来曝光度最高的技术，但它不是人工智能的全部创新，在上世纪 80 年代末就已经掀起了人工智能的多层神经网络技术热潮，但由于当

时没有现在的大数据和高性能的计算力，所以其研究受到了阻碍。中国科学院数学与系统科学研究院数学研究所研究员、复旦大学教授陆汝铃[35]在为《机器学习》一书撰写序言的时候提及了深度学习的缺陷：一是深度学习的理论创新还不明显，二是目前的深度学习只适用于神经网络，这些都是值得进一步的研究分析的。另外，深度学习具有不可解释性，他从原始的输入信息到提取特征再到输出信息的过程是一个黑盒子，使得有关人员在使用时难以对其有特别清晰具体的认识。有界人士认为：“如何将深度学习过程和人类已经积累的大量高度结构化知识融合，发展出逻辑推理甚至自我意识等人类的高级认知功能，是下一代深度学习的核心理论问题”。同时，随着数据量的增大和模型的复杂性提高等，深度学习目前仍有大量工作需要研究。

如今深度学习已经渗透到了各个领域，无论是在计算机科学还是金融、医学等研究中，深度学习都发挥着重要的作用。当然深度学习不是全部，但是对其的深入研究是必不可少的。随着网络技术的发展，深度学习将不仅仅用于语音识别、信息检索、图像处理等，在以后的研究中，深度结构的层次模型将更加复杂且合乎实际，网络的学习和适应能力将更加强大，对于深度学习的具体研究也将更加深入，从而使得人工智能的水平得到进一步提升。例如，无人机被用于许多实时图像分析任务以及基础设施检查，深度学习对于该领域的预测和决策任务有很大的影响，可以推动无人机达到最佳性能[5]；深度学习还有助于虚拟/增强现实这个应用领域的发展，增强现实可以用于提供比如行为识别、图像分类、目标跟踪等服务[36]，这对教育、科研等有着重大的贡献。因此，深度学习作为人工智能、大数据、物联网等前沿产业发展的催化剂，其未来发展和研究值得重视。

第 3 章　基本梯度下降算法

3.1　基本概念

3.1.1　梯度下降

梯度是微积分里面一个重要的概念，对多元函数的参数求偏导，再把求得的各个参数的偏导数以向量的形式写出来，得到的就是梯度。比如二元函数 $f(x,y)$，分别对 x 和 y 求偏导，得到的向量 $(\partial f/\partial x, \partial f/\partial y)^T$ 就是该函数的梯度，记作 $\nabla f(x,y)$ 或者 $grad(x,y)$。

梯度的几何意义[37]是函数变化增加最快的地方，在函数的某一点处求梯度得到的向量的方向就是函数增加最快的方向，沿着该方向更容易找到函数的最大值，而反过来讲，沿着梯度向量相反的方向，更加容易找到函数的最小值。如图 3.1 中的黑线即为梯度下降算法的路径：

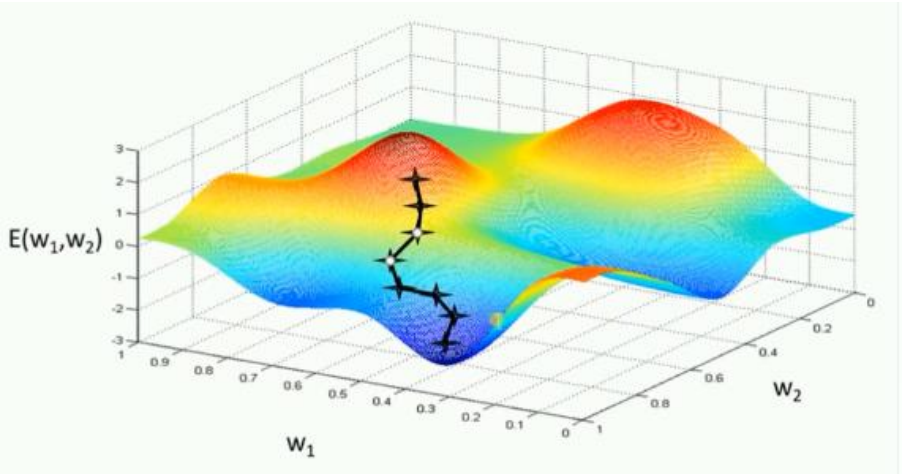

图 3.1 梯度下降算法

Fig.3.1 Gradient descent algorithm

因此，在机器学习中最小化损失函数[38]时，就是利用这个梯度下降概念进行迭代求解最小化的损失函数和模型参数值。直观地讲，梯度下降就相当于在一座大山中一步步地寻找最陡峭的方向从而以最快的速度抵达山脚。同时这个内容也涉及一些相关的概念，首先是函数的样本输入部分（即特征，相当于初始值）；其次是步长，它决定了在迭代的过程中每一步前进的长度；接着是为拟合输入样本所使用的假设函数；还有就是为了评估模型拟合好坏，常用损失函数来度量拟合的程度，当损失函数达到极小值时，常意味着拟合程度最好，此时训练得到的模型参数也是最优的。

3.1.2　算法代数方式描述

对于变量比较少的模型，采用代数法的形式描述更加容易理解。在使用梯度下降算法时，首先需要已经确定优化模型的假设函数和损失函数，比如对于最简单的线性回归，假设函数为：

$$h_\theta(x_1, x_2, \cdots x_n) = \theta_0 + \theta_1 x_1 + \cdots \theta_n x_n \tag{3.1}$$

其中的$\theta_i (i = 1,2,\cdots n)$就是模型的参数，损失函数令为：

$$J(\theta_0, \theta_1, \cdots \theta_n) = \frac{1}{2m}\sum_{j=0}^{m}\left(h_\theta\left(x_0^{(j)}, x_1^{(j)}, \cdots x_n^{(j)}\right) - y_j\right)^2 \tag{3.2}$$

然后对相关的参数（包括模型参数、迭代的步长以及算法的终止距离等）进行初始化，一般情况下，直接将模型参数初始化为 0，将步长初始化为 1，然后在后续的实际操作中调优。具体的算法过程如下：1. 计算损失函数在当前位置的的梯度，记作∇f；2. 计算在当前位置需要下降的距离，即步长乘以∇f；3. 在所有θ_i都大于终止距离时，对模型参数进行更新，更新表达式为：

$$\theta_i = \theta_i - \alpha \frac{\partial}{\partial \theta_i} J(\theta_0, \theta_1, \cdots \theta_n) \tag{3.3}$$

更新完成回到第一步进行迭代；4. 若存在θ_i小于终止距离则迭代算法终止，此时得到的模型参数解即是最优解。

3.1.3 算法矩阵方式描述

而对于模型变量比较多的情况，则采用矩阵表示比较简易且有利于后续的工作。同样的，在已经确定所优化模型的假设函数和损失函数的情况下，此时的线性回归的假设函数表示为：

$$h_\theta(X) = X\theta \tag{3.4}$$

其中$h_\theta(X)$为$m \times 1$的向量，θ为$(n+1) \times 1$的代表模型参数的向量，m为样本的个数，$n+1$为样本的特征数。而对多元线性回归，一般令均方误差（Mean Square Error,MSE）为损失函数：

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}\left(\theta^T \cdot X^{(i)} - Y^{(i)}\right)^2 \tag{3.5}$$

Y是样本的输出向量。然后同样对相关参数进行初始化，算法过程与代数形式的步骤类似，此时损失函数对θ求偏导为：

$$\frac{\partial}{\partial\theta}J(\theta)=X^T(X\theta-Y) \tag{3.6}$$

而θ的更新表达式则为：

$$\theta=\theta-\alpha X^T(X\theta-Y) \tag{3.7}$$

与代数法相比较，矩阵法更简洁，逻辑也更加清晰，更加适用于如今模型参数多、复杂度高的问题的解决。

3.2 基于样本使用量的算法变式

3.2.1 批量梯度下降算法

批量梯度下降法（Batch Gradient Descent，BGD）[39]-[40]又名全梯度下降法（FG），对于该算法，计算损失函数是在整个训练集上进行的，使用所给的所有样本进行训练，即在对模型参数进行更新时将带入整个数据集计算梯度，代数表示为：

$$\theta=\theta-\eta\cdot\nabla_\theta J(\theta) \tag{3.8}$$

其实，$J(x;\theta)$表示的是总损失，它等于样本平均值，即：

$$J(x;\theta)=\frac{1}{m}\sum_{i=1}^{m}J_i(x_i,\theta) \tag{3.9}$$

其中的$J_i(x_i,\theta)$表示第 i 个数据样本x_i的损失，那么所求的$\nabla_\theta J(\theta)$也是对各个样本点损失函数梯度之和。如果数据量 m 很大，则求解$\nabla_\theta J(\theta)$的计算量相当大。

由于整个数据集比较大，一方面会导致运算和收敛速度较慢，特别是随着数据量 n 的增大，算法的时间复杂度$O(n)$会显著提高，增加时间成本；另一方面可能会面临内存不足的问题使得问题不能得以解决。但是 BGD 算法在解决损失函数为凸函数（例如逻辑回归问题）的问题时，它利用了更多的已知信息，从而可以使参数的每一次更新都朝着正确的方向进行，直到收敛于全局最优解。

因此，随着数据量的增大，直接使用 BGD 算法的频率在降低，但对于某些数据量不太大的问题，该算法才是最佳算法。

3.2.2 随机梯度下降算法

随机梯度下降算法（Stochastic Gradient Descent，SGD）[39]-[40]是对上一个算法的极端改进，它每次迭代中在所给样本中随机选择一个样本来对模型参数进行

更新，即每次迭代只需要一个样本对损失函数$J(\theta)$进行训练，代数表示为：

$$\theta = \theta - \eta \cdot \nabla_{\theta} J\left(\theta; x^{(i)}; y^{(i)}\right) \tag{3.10}$$

其中$\left(x^{(i)}, y^{(i)}\right)$即为所选样本。而由于每次选择的单个样本可能不同，则式中的$J\left(\theta;\ x^{(i)}; y^{(i)}\right)$是动态变动的。

可见，该算法的运算量明显降低，从而加快了训练过程，使得每次学习很快，运算效率很高。对于大数据量的机器学习而言，该算法以其速度快和易收敛的优点颇受欢迎。

但是，该算法只利用了极少的样本信息，而高频率的参数更新会导致高方差，所以有可能会出现目标函数值剧烈震荡的现象[6]，如图 3.2 所示：

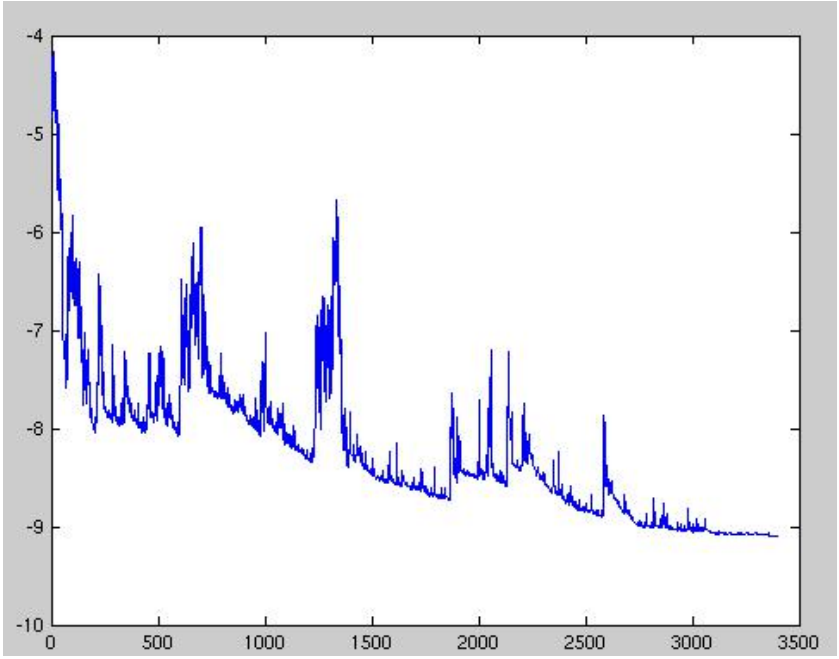

图 3.2 SGD 震荡现象

Fig.3.2 SGD shock phenomenon

因此，SGD 算法会带来一定的问题，因为计算得到的并不是一个准确的梯度，所以每次更新可能不会按照正确的方向前进，容易陷入局部最优解；另外，该算法的波动性使得收敛过程更加复杂，其次，SGD 对所有参数更新时使用的学习率都是相同的，这使得在数据稀疏时更新次数太多引起不必要的成本。但是对于类似盆地区域（有很多局部极小值点），该算法的波动性可能会反而带来好处，使得优化方向从当前局部极小值点跳到另一个更好的局部极小值点，这样对于非凸函数而言，可能最终会收敛于一个较好的局部极值点甚至是全局极值点。

3.2.3 SAG（Stochastic Average Gradient）

标准的 FG 算法的收敛速度与 n 有很大的关系，而 SGD 算法的不稳定性又

很大，于是 Nicolas Le Roux 等人[16]基于 SGD 算法提出了一种对于数量较大的有限的训练集可以达到指数级收敛速度的算法－－SAG 算法。该算法的迭代形式为：

$$x^{k+1} = x^k - \frac{\alpha_k}{n}\sum_{i=1}^{n} y_i^k \tag{3.11}$$

在每次迭代计算中会使用两个梯度的值，一个是前一次迭代的梯度值，另一个是新随机选择的样本的梯度值。具体的过程可简单表示为：

Algorithm 1 Basic SAG method for minimizing $\frac{1}{n}\sum_{i=1}^{n} f_i(x)$ with step size α.

$d = 0$, $y_i = 0$ for $i = 1, 2, \ldots, n$
for $k = 0, 1, \ldots$ **do**
  Sample i from $\{1, 2, \ldots, n\}$
  $d = d - y_i + f_i'(x)$
  $y_i = f_i'(x)$
  $x = x - \frac{\alpha}{n} d$
end for

图 3.3 SAG 算法[16]

Fig.3.3 SAG algorithm[16]

这样所利用的信息比较多，收敛速度就比 SGD 算法更快，但是，该算法需要另开内存来保存旧的梯度值，所以这个算法就相当于用空间成本换取时间成本。

3.2.4 小批量梯度下降算法

小批量梯度下降（Mini-batch Gradient Descent，MBGD）[41]可以看做是上述前两个算法的折中方案，尽量保留这两个算法的优点而避免一些缺点，该算法选取训练数据集中的小批量的样本进行训练，其代数表示式为：

$$\theta = \theta - \eta \cdot \nabla_\theta J\left(\theta; x^{(i:i+n)}; y^{(i:i+n)}\right) \tag{3.12}$$

其中 n 即为所选取的样本数量。

该算法在更新参数时只选取小批量的样本，使得在降低训练过程运算量的情况下又提升了收敛的稳定性，通常根据不同的数据集的数量级所设定的小批量亦是不同级别的。如今很多的深度学习库中利用矩阵优化方法已经可以高效地求解所设定的小批量的梯度。因此，在机器学习中该算法运用比较广泛。

3.2.5 存在的问题

传统的梯度下降算法存在的困难有：

1. 梯度的计算上：在机器学习和统计参数估计问题中目标函数经常是求和函数的形式 $J_x(\theta) = \sum_i J_{x_i}(\theta)$（其中每一个函数 $J_{x_i}(\theta)$ 都对应于一个样本 x_i），而当

样本量极大时，梯度的计算就变得非常耗时耗力。

2. 学习率的选择上：学习率选择过小会导致算法收敛太慢从而浪费运行时间，过大则容易导致算法不收敛产生偏离最小值等问题，故如何选择学习率需要具体问题具体分析。

3. 虽然 SGD 使用比较广泛，但是具有很大的不稳定性。

而主要困难即为第二个问题：学习率的选择。具体而言：

(1) 局部梯度的反方向不一定是函数整体下降的方向：对于图象比较崎岖的函数，尤其是隧道型曲面，梯度下降表现不佳。

(2) 预定学习率减法的问题：学习率衰减很难根据当前数据进行自适应。

(3) 对不同参数采取不同的学习率的问题：在数据有一定稀疏性时，希望对不同特征采取不同的学习率。

(4) 神经网络训练中梯度下降法容易被困在鞍点附近的问题：比起局部极小值，鞍点更加可怕。

基于传统的梯度下降算法，上述四种基本的改进算法是无法满足实际研究应用的需要的，因此现今出现了很多改进的梯度下降算法变式，通过解决以上所述问题使得梯度下降算法更加适用，从而更好地解决相关问题。

第 4 章　常见梯度下降优化算法

针对上述的困难以及基本的梯度下降算法（尤其是 SGD 算法）存在的挑战，陆续地提出了很多新的改进算法，以下将对基于不同的问题而提出的改进算法进行具体的研究分析。

4.1　基于动量的改进算法

上文中提到 SGD 算法具有很大的波动性，从而使得收敛很不稳定。于是有学者借助物理中动量的概念对算法进行优化，一个物体的动量是指该物体在它运动方向上保持这个运动的趋势，物理表示为物体质量与速度的乘积，借鉴此思想，在使用梯度下降算法中对参数的更新方向进行优化，使得参数在梯度方向不变的时候更新加快，而在梯度方向改变的时候更新速度减慢，以此来加快收敛并且减少震荡。

4.1.1　动量梯度下降法（Momentum 算法）

Momentum 算法[42]就是在参数更新时一方面一定程度上保留之前的更新方向，另一方面又利用当前计算的梯度微调最终的更新方向，简单的讲就是积累之前的动量来加速当前的梯度。Momentum 算法也可以看作是基于梯度的移动指数加权平均，数学表达式为：

$$v_t = \gamma v_{t-1} + \eta \nabla_\theta J(\theta_t) \tag{4.1}$$

$$\theta_t = \theta_{t-1} - v_t \tag{4.2}$$

其中 γ 为动量项，通常取 0.9 或近似值；v_t、v_{t-1} 分别为第 t 次和第 t-1 的下降动量；$\nabla_\theta J(\theta_t)$即为第 t 次更新时的梯度值。简单示意图如下图 4.1 所示：

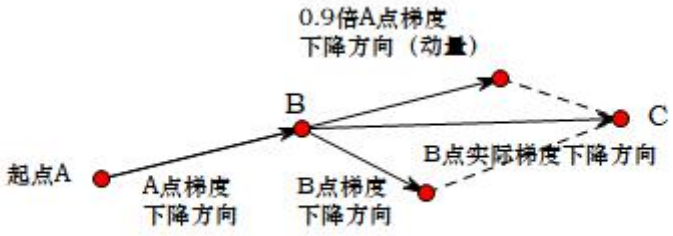

图 4.1 momentum 下降法示意图

Fig.4.1 Diagram of momentum reduction method

然后再以简单的二维函数 $f(x_1, x_2) = 0.5x_1^2 + 2x_2^2$ 为例展现 SGD 与动量法的区别，如下图所示：

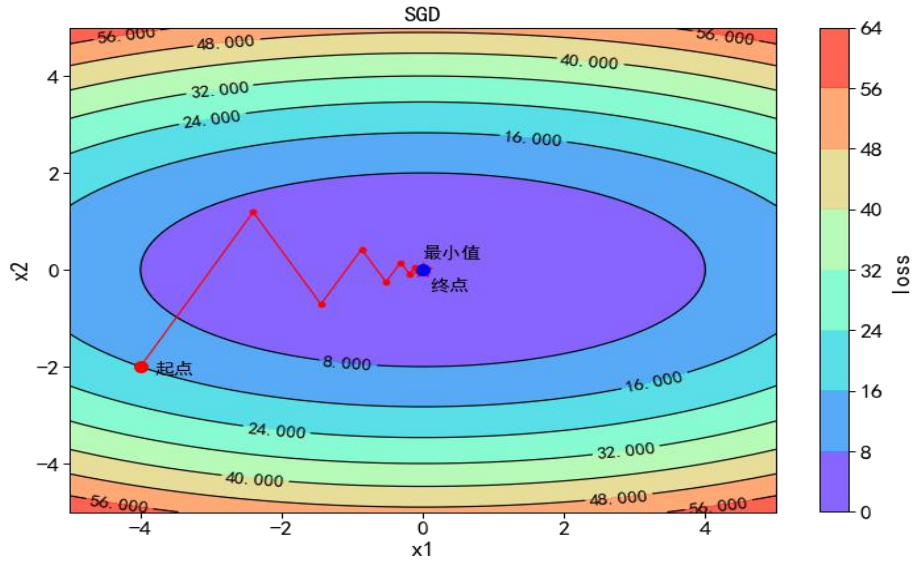

图 4.2 SGD 结果

Fig.4.2 SGD result

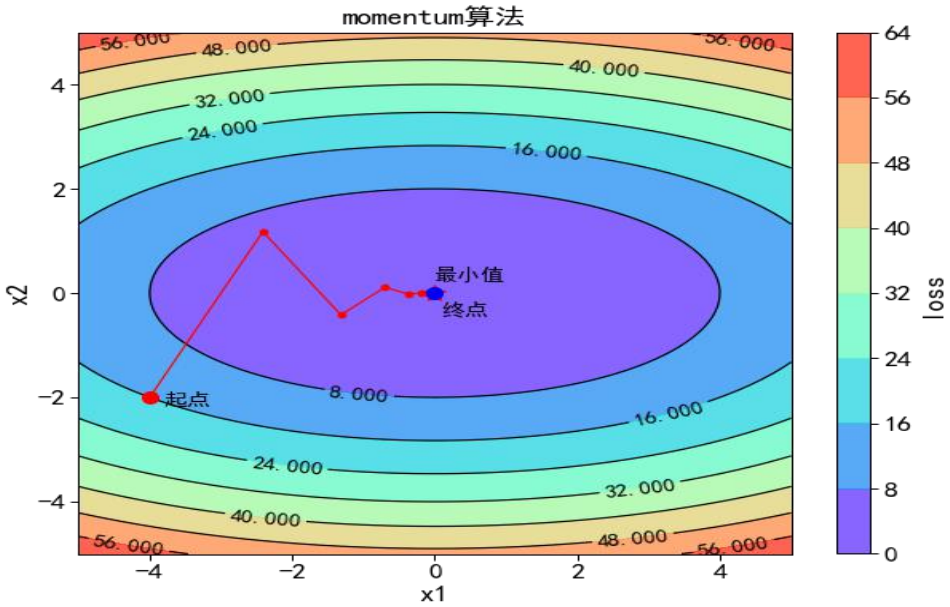

图 4.3 momentum 结果

Fig.4.3 momentum result

从这个简单的案例中可以看出动量法的收敛速度快于 SGD 算法，在迭代的初期梯度方向比较一致，动量法会起到加速的作用从而更快到达最优点，而在迭代后期，梯度方向会不太一致，在收敛值附近震荡，动量法可以起到减速的作用从而增加稳定性，但是如果下降到极值点附近时速度过快则无法及时停止。

4.1.2 NAG（Nesterov Accelerated Gradient）

对于上述策略，就像是受训练的小球从山上盲目的沿着梯度方向往下滚，其

实这得到的不一定是最优解，于是就有人研究“智能的球”，它可以在下坡的时候加速而预判到要上坡的时候减速。

NAG 算法[43]就是一种实现此功能的算法，它给予动量一种预测能力，从而使得训练在加速收敛的同时抑制摇摆。如下图所示为 NAG 算法的简单示意图：

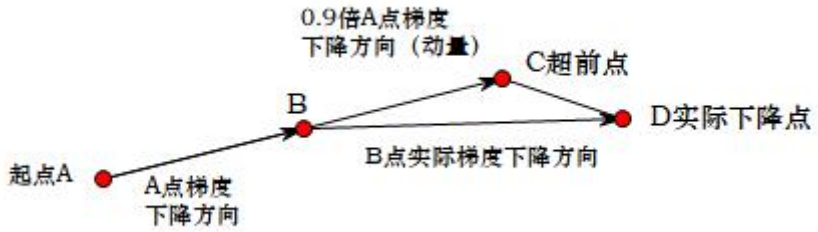

图 4.4　NAG 下降法示意图

Fig.4.4 Schematic diagram of NAG descent method

将图 4.1 和图 4.4 对比就可以看出，NAG 算法区别于动量法之处就在于 B 点和 C 点梯度的不同。NAG 算法会向前看一步，预估超前点处的梯度，然后将该点处的梯度和历史累积梯度加权得到下一个迭代点的更新方向，该方法可视为超前预测，可以在往错的方向运动之前阻止向前，适应性比较好。

该算法具体的推导准则为：令 v_t 为第 t 次梯度的累积，由以下式子：

$$v_0 = 0 \tag{4.3}$$

$$v_1 = \eta \nabla_\theta J(\theta) \tag{4.4}$$

$$v_2 = \gamma v_1 + \eta \nabla_\theta J(\theta - \gamma v_1) \tag{4.5}$$

可以得到：

$$v_t = \gamma v_{t-1} + \eta \nabla_\theta J(\theta - \gamma v_{t-1}) \tag{4.6}$$

而参数更新公式就为：

$$\theta' = \theta - v_t \tag{4.7}$$

其中 $-\gamma v_{t-1}$ 即为上图中的向量 BC，γv_{t-1} 叫做动量项；$\theta - \gamma v_{t-1}$ 即为超前点 C，γ 为衰减率，常设为 0.9；η 为学习率。

对于 Lipshitz 凸函数，NAG 算法的收敛速度达到 $O\left(1/t^2\right)$，Nesterov 曾证明该算法对任何基于一阶梯度的方法可能是最佳速率，但对于非凸的模型，该方法并不能达到这个收敛速度。这个智能的具有预见性的算法可以在梯度上升的时候及

时停止，增强了算法的响应力，提高了基本的动量法的稳定性。

4.1.3 SVRG

对于 SGD 算法存在的受噪声干扰问题提出了另一个改进算法——SVRG[17]，它的全称是 Stochastic Variance Reduction Gradient（改进的随机方差消减梯度法），具有线性收敛速度。在 SGD 或 SAG 算法中都是用一个样本值去估计梯度，但在 SVRG 算法中是用一个“批”去估计梯度，这并不是每次计算都要用新的一批样本去估计梯度，而是在一段时间内（如 m 次迭代中）使用这 n 个样本来估计梯度，这段时间结束就重新选择 n 个样本来估计梯度，每次更新最多计算两次梯度，和 SAG 算法对比可见，该算法不需要在内存中为每个样本都维护一个梯度从而大大节省了内存资源；另外，该算法提出了一个重要的概念叫做方差缩减，在 SGD 算法中需要假设样本梯度的方差有上界从而使得其无法线性收敛，而 SVRG 算法可以使方差有一个不断减小的上界从而可以做到线性收敛。

该算法的具体训练过程为：首先计算每 w 次迭代中所有样本的梯度来维持区间平均梯度：

$$\tilde{\mu}=\frac{1}{N}\sum_{i=1}^{N}g_i\left(\tilde{\theta}\right) \tag{4.8}$$

其中 $\tilde{\theta}$ 是间隔更新参数，$\tilde{\mu}$ 包含每个时间间隔 w 过去时间中所有样本梯度的平均内存，SVRG 算法随机选择 $i_t\in\{1,\cdots,\ N\}$，并对当前参数执行梯度更新：

$$\theta_t=\theta_{t-1}-\eta\cdot\left(g_{i_t}\left(\theta_{t-1}\right)-g_{i_t}\left(\tilde{\theta}\right)+\tilde{\mu}\right) \tag{4.9}$$

该算法的提出者将该算法的训练过程表示为：

Procedure SVRG

Parameters update frequency m and learning rate η
Initialize $\tilde{w}_0$
Iterate: for $s=1,2,\ldots$
  $\tilde{w}=\tilde{w}_{s-1}$
  $\tilde{\mu}=\frac{1}{n}\sum_{i=1}^{n}\nabla\psi_i(\tilde{w})$
  $w_0=\tilde{w}$
  Iterate: for $t=1,2,\ldots,m$
    Randomly pick $i_t\in\{1,\ldots,n\}$ and update weight
    $w_t=w_{t-1}-\eta(\nabla\psi_{i_t}(w_{t-1})-\nabla\psi_{i_t}(\tilde{w})+\tilde{\mu})$
  end
  option I: set $\tilde{w}_s=w_m$
  option II: set $\tilde{w}_s=w_t$ for randomly chosen $t\in\{0,\ldots,m-1\}$
end

图 4.5 SVRG 算法过程[17]

Fig.4.5 SVRG algorithm process[17]

SVRG 算法具有更快的收敛速度并可以实现线性收敛，在实际应用中要求精确的训练精度时，该算法表现出优于 SGD 算法的性能，更加有效。

4.2 基于学习率问题的改进算法

学习率控制迭代的步长，在梯度下降算法中占有举足轻重的地位，影响着搜索函数的全局最优值的实现。一种最简单的方式是固定学习率，即每次迭代时每个参数使用相同的学习率，上述提到的算法都是这样的，但这种方法适用于那些目标函数是凸函数的模型，这个固定值的选择并不是随意的，通常为了保证收敛会选择一个比较小的值，比如 0.001、0.01 等，但是这种方式有不足之处，因为不同参数的重要性肯定是不同的，学习率取值不同会对梯度下降产生不同的影响。于是就有了许多对如何设定学习率的研究，以下将具体介绍一些相关的算法。

4.2.1 Adagrad 算法

Adagrad 算法[18]就是解决不同参数应该使用不同的更新速率的问题，该算法自适应地为各个参数分配不同的学习率，对出现频率高的参数提供较小的学习率，而对出现频率低的参数使用较大的学习率。每一个参数 θ_i 的具体的表达式为：

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{G_{t,ii} + \varepsilon}} \nabla_\theta J\left(\theta_{t,i}\right) \tag{4.10}$$

其中 G_t 是一个对角矩阵，其对角上的每个元素 $G_{t,ii}$ 为 t 时刻前所有累积到 θ_i 上的梯度的平方和，通常为防止出现除数为 0 的情况，会用 ε 设置为很小的数（例如 10^{-8}）。所以 Adagrad 算法就是将每一个参数的每次迭代的梯度取平方累加、再开方，再用全局学习率除以这个数从而得到各参数新的学习率。

从算法更新准则可以看出，随着该算法迭代次数不断增加，累积梯度会越来越大而使得整体的学习率会越来越小，因此，Adagrad 算法在最初是激励收敛，但到后面会变成惩罚收敛，即收敛速度越来越慢，这样更加符合实际，因为在最初阶段，受训模型距离最优解还很远，学习率大点训练比较快，而随着更新次数增加，越来越接近最优解，此时放慢训练过程的效果会更好。

该算法的优点是不用手动调节学习率，并且在数据分布稀疏的情形中可以更好地利用稀疏梯度（即自然语言和计算机视觉问题）的信息，会比标准的 SGD 算法更有效地收敛；但是缺点也存在：随着时间的延长和迭代次数额增加，学习率会衰减逐渐接近于 0，可能会导致学不到信息甚至是训练过早结束（依赖于人工设定的学习率）。

4.2.2 Adadelta 算法

针对 Adagrad 算法的学习率持续退火以及需要人工选择学习率问题，

Matthew D.Zeiler 提出的 Adadelta 算法[44]对其进行了相应的改进。Matthew D.Zeiler 的改进思路分为两种，第一种改进思路是在一个窗口内对梯度进行累和来进行运算，限制一个固定的大小 ω 而不是对所有梯度一直累加，这样可使 Adagrad 算法中的分母不止于趋于无限，就确保了训练迭代过程即使进行很多次也不至于提前结束训练。由于存放 ω 之前的梯度是低效率的，所以可以用对先前所有梯度均值（使用 RMS 即均方根值）的一个指数衰减作为替代的实现方法。令从当前 t 时刻开始之前的梯度的平方的期望值是 $E\left[g^2\right]_t$，然后 Adadelta 算法使用类似动量因子的平均方法计算：

$$E\left[g^2\right]_t = \rho E\left[g^2\right]_{t-1} + (1-\rho) g_t^2 \tag{4.11}$$

其中 ρ 是一个类似于动量法中使用的衰减常数，当 $\rho = 0.5$ 时此式为求梯度平方和的平均值，此式相当于历史梯度信息的累计乘以一个衰减系数 ρ 与 $(1-\rho)$ 乘以当前梯度的平方相加；然后再求根的话即为 RMS：

$$RMS\left[g\right]_t = \sqrt{E\left[g^2\right]_t + \varepsilon} \tag{4.12}$$

其中 ε 是一个用来防止分母为 0 的常数，再将这个作为梯度的正则化得到更新规则：

$$\Delta x_t = -\frac{\eta}{RMS\left[g\right]_t} g_t \tag{4.13}$$

这样就得到了 Adagrad 的改进版，它解决了对历史梯度一直累加而导致的学习率衰减得问题，但仍需要人工选择初始学习率。

Matthew D.Zeiler 的进一步改进的第二个思路是用 Hessian 方法得到正确的更新单元。在之前的 SGD、动量法及 Adagrad 算法中的参数更新都是无单位的，并且假设成本函数也是无单位的，有：

$$units\ of\ \Delta x \propto units\ of\ \mathrm{g} \propto \frac{\partial f}{\partial x} \propto \frac{1}{units\ of\ x} \tag{4.14}$$

于是利用二阶导 Hessian 矩阵法思想，其中 Hessian 矩阵为高阶的牛顿法迭代的步长，采用 Hessian 矩阵的对角线近似 Hessian 矩阵，公式为：

$$\Delta x \approx \frac{\dfrac{\partial f}{\partial x}}{\dfrac{\partial^2 f}{\partial x^2}} \tag{4.15}$$

从而有

$$\frac{1}{\frac{\partial^2 f}{\partial x^2}}=\frac{\Delta x}{\frac{\partial f}{\partial x}} \tag{4.16}$$

而更新公式为

$$x_{t+1}=x_t-\frac{1}{\frac{\partial^2 f}{\partial x^2}}*g_t=\frac{\Delta x}{\frac{\partial f}{\partial x}}*g_t \tag{4.17}$$

假设 x 附近的曲率平滑，且x_{t+1}可以近似x_t，则有：

$$\Delta\theta_t=\frac{RMS\left[\Delta\theta\right]_{t-1}}{RMS\left[g\right]_t}*g_t \tag{4.18}$$

$$\theta_{t+1}=\theta_t+\Delta\theta_t \tag{4.19}$$

其中g_t为本次迭代的梯度。由于*RMS*始终为正，所以可保证更新方向一直为梯度的负方向；分子作为一个加速项，作为动量在时间窗口ω上累积之前的梯度。可见此时已不需要设定缺省学习率，因更新规则已不受其影响。

4.2.3 RMSprop 算法

RMSprop 算法[6]全称是 Root Mean Square Prop（均方根反向传播），它是 Geoffrey E. Hinton 在 Coursera 课程中提出的一种优化算法，该算法与上述的 Adadelta 算法同年被提出，都是针对 Adagrad 算法的学习率问题而提出的解决方案。其实 Adadelta 算法中的第一个改进思路得到的改进版的 Adagrad 算法就是 RMSprop 算法，当时对应的研究者是独立提出这两个算法的。故 RMSprop 算法的具体表达式即为：

$$E\left[g^2\right]_t=\rho E\left[g^2\right]_{t-1}+\left(1-\rho\right)g_t^2 \tag{4.20}$$

更新规则为：

$$\theta_{t+1}=\theta_t-\frac{\eta}{\sqrt{E\left[g^2\right]_t+\varepsilon}}*g_t \tag{4.21}$$

Hinton 将其中的ρ设置为 0.9，η设为 0.001，经有关实验证明，这样设定的效果确实很好，步长更新更加缓和。

RMSprop 算法也是 RProp（弹性反向传播）算法的改良版，RProp 算法首先

为各权重变化初始化一个值，设定权重变化加速因子与减速因子，在网络前馈迭代中当连续误差梯度符号不变时采用加速策略从而加快训练速度，当误差梯度符号变化时采用减速策略以期稳定收敛，网络结合当前误差梯度符号与变化步长实现 BP，同时为了避免网络学习发生震荡或下溢，算法要求设定权重变化的上下限，这种算法适用于全梯度训练，但是不适用于 MBGD。

RMSprop 算法相当于对梯度计算了微分平方加权平均数，这样更加有利于消除摆动幅度大的方向，可以修正幅度从而使得各维度的摆动幅度都较小，同时也使得训练收敛更快，该算法在在线以及非平稳问题中表现出很好的性能，但是此算法并不能完全解决局部最小值问题，只是使得参数收敛得更快。

4.2.4 Adam 算法

Adam 算法[46]名称来源于自适应矩估计（Adaptive Moment Estimation），该算法也是一种自适应学习率的算法，它基本上可以看做是 Momentum 和 RMSprop 算法的结合。一方面，它让每个参数都计算自己的学习率，就像 RMSprop 算法一样有一个历史梯度的存储器，同时还有像 Momentum 算法中的动量一样的用于保存梯度的指数衰减均值。Adam 算法同时获得了 Adagrad 和 RMSprop 算法的优点，它不仅像 RMSprop 算法那样基于一阶矩均值计算适应性参数学习率，还利用梯度的二阶矩（即还计算了梯度的指数移动平均），所以 Adam 算法的应用性以及表现性能会更佳，有很高的计算效率和较低的内存需求。

Adam 算法的具体训练过程为：在确定了参数 α、β_1、β_2 以及随机目标函数 $f(\theta)$ 后，就对参数向量、梯度一阶矩向量 m（梯度的指数移动均值）、梯度二阶矩向量 v（平方梯度）和时间步长进行初始化。对 m_t 有更新表达式：

$$m_t = \beta_1 \cdot m_{t-1} + (1-\beta_1)\cdot g_t \tag{4.22}$$

对 v_t 有更新表达式为：

$$v_t = \beta_2 \cdot v_{t-1} + (1-\beta_2)\cdot g_t^2 \tag{4.23}$$

其中 $g_t = \nabla_\theta f(\theta_{t-1})$，$\beta_1$、$\beta_2 \in [0,1)$ 用于控制这些移动均值指数衰减率；然后计算偏差修正的 m_t 和 v_t 来抵消偏差：

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t} \tag{4.24}$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t} \tag{4.25}$$

最后给出的参数更新规则为：

$$\theta_t = \theta_{t-1} - \alpha \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon} \tag{4.26}$$

当参数θ未收敛时，循环迭代地更新各个部分。在初始化时若这些移动均值初始化为零向量，则矩估计值会偏向于 0，特别是在初始步骤和衰减非常小（即β接近于 1）的情况下是这样的，但是初始化偏差很容易抵消，即用上述的$\hat{m}_t$和$\hat{v}_t$进行修正。另外，对 Adam 算法中的参数进行配置中，用于控制权重的更新比率的步长因子α是需要针对不同的训练集进行调参的，较大的值（如 0.4）在学习率更新前会有更快的初始学习，而较小的值（如 0.00001）会使训练收敛到 更好的性能；一阶矩估计的指数衰减率β_1常用缺省值是 0.9,；二阶矩估计的指数衰减率常设置为 0.999（该数值由 Adam 算法的提出者推荐）；ε一般令为10^{-8}；除了α需要进行调参外，其他参数一般直接使用默认缺省值，当然对于当前流行的深度学习库，某些参数值会有细微的差别。

Adam 算法较之前的算法都很多优势，具体包括：更加高效的计算、所需内存少、具有梯度对角缩放的不变性、更加适合解决大规模数据和高维空间问题、适合非稳态目标、适用于解决包含很高噪声或稀疏梯度的问题以及超参数可以很直观地解释且基本上只需要进行极小的调参等等，所以该算法比其他自适应学习率算法更有优势从而应用更加广泛；当然，该算法也可能会出现在局部最小值附近震荡导致的不收敛问题。

4.2.5 AdaMax

从上述的介绍知道 Adam 算法中单个权重的更新规则是将其梯度与当前和过去梯度的L^2范数（标量）成反比例缩放，而在 AdaMax 算法[47]中基于L^p范数的更新规则中的$p \to \infty$，这样可以得到一个极其稳定且简单的算法，在此种情况下，时间 t 下的步长和$v_t^{1/p}$成反比例变化。具体的推导准则如下：

$$v_t = \beta_2^p v_{t-1} + \left(1-\beta_2^p\right)\left|g_t\right|^p = \left(1-\beta_2^p\right)\sum_{i=1}^{t}\beta_2^{p(t-i)} \cdot \left|g_i\right|^p \tag{4.27}$$

由$p \to \infty$并令新的二阶矩$u_t = \lim_{p\to\infty}\left(v_t\right)^{1/p}$，则有：

$$u_t = \lim_{p\to\infty}\left(\left(1-\beta_2^p\right)\sum_{i=1}^{t}\beta_2^{p(t-i)} \cdot \left|g_i\right|^p\right)^{1/p}$$

$$
\begin{aligned}
&= \lim_{p\to\infty}\left(1-\beta_2^p\right)^{1/p}\left(\sum_{i=1}^{t}\beta_2^{p(t-i)}\cdot|g_i|^p\right)^{1/p} \\
&= \lim_{p\to\infty}\left(\sum_{i=1}^{t}\beta_2^{p(t-i)}\cdot|g_i|^p\right)^{1/p} \\
&= \max\left(\beta_2^{t-1}|g_1|,\beta_2^{t-2}|g_2|,\cdots,\beta_2|g_{t-1}|,|g_t|\right) \\
&= \max\left(\beta_2\cdot u_{t-1},|g_t|\right)
\end{aligned}
\tag{4.28}
$$

其中初始值 $u_0=0$，其他的一阶矩更新与纠正以及参数的更新规则都与 Adam 算法一样，但此时不需要修正初始化偏差，并且 AdaMax 算法的参数更新的量级比 Adam 算法更简单。

4.2.6 Nadam

Nadam 算法[47]是 Adam 和 NAG 算法的结合，其具体的推导公式如下：

$$\hat{g}_t=\frac{g_t}{1-\prod_{i=1}^{t}\mu_i} \tag{4.29}$$

$$m_t=\mu_t*m_{t-1}+(1-\mu_t)*g_t \tag{4.30}$$

$$n_t=\nu*n_{t-1}+(1-\nu)*g_t^2 \tag{4.31}$$

对 m_t 和 n_t 的更新有：

$$\hat{m}_t=\frac{m_t}{1-\prod_{i=1}^{t+1}\mu_i} \tag{4.32}$$

$$\hat{n}_t=\frac{n_t}{1-\nu^t} \tag{4.33}$$

令：

$$\overline{m}_t=(1-\mu_t)*\hat{g}_t+\mu_{t+1}*\hat{m}_t \tag{4.34}$$

则参数的更新规则为：

$$\Delta\theta_t=-\eta*\frac{\overline{m}_t}{\sqrt{\hat{n}_t}+\varepsilon} \tag{4.35}$$

可见，Nadam 算法对学习率有了更强的约束，同时对梯度的更新也有更直

接的影响。正因为 Nadam 算法是 Adam 和 NAG 算法的结合，所以在能使用这两种算法的情况下一般可以使用 Nadam 算法得到更好的效果。

4.2.7 其他有关研究点

上述各个算法的介绍中展现了学习率的重要性，确实对于梯度下降算法，只有调好学习率这个超参数，才能让梯度下降算法更好地运作从而使模型达到更好的效果。前面的算法中，有的算法的学习率是固定不变的，即每次迭代中每个参数都使用相同的学习率，但是找到一个最佳的固定学习率是一项比较关键但又很困难的工作；后来对于学习率出现了很多改进算法（在上文已介绍），对不同的参数采用不同的学习率，使得训练结果更加有效。对于学习率的设定，还有一些其他的策略。

第一种方法是学习率衰减策略，即随着训练的进行，学习率将逐渐衰减，前文介绍的 Adagrad、Adam 等算法就利用了这个思想。衰减方式可以是线性亦或是指数衰减，比如有如下方式：1、学习率呈多项式曲线下降；2、学习率随着迭代次数的增加而下降；3、学习率在每次迭代后减少 a 倍。这种学习率逐渐下降的方法更适用于目标函数不太复杂的时候，这种方式比固定学习率训练的速度加快，但是对于解决鞍点以及局部极小值问题的作用不大。

第二种方式是循环学习率，即学习率缓慢增加在缓慢减小并且这样循环进行，Leslie N.Smith 曾[48]提出了两种循环形式，如下图所示：

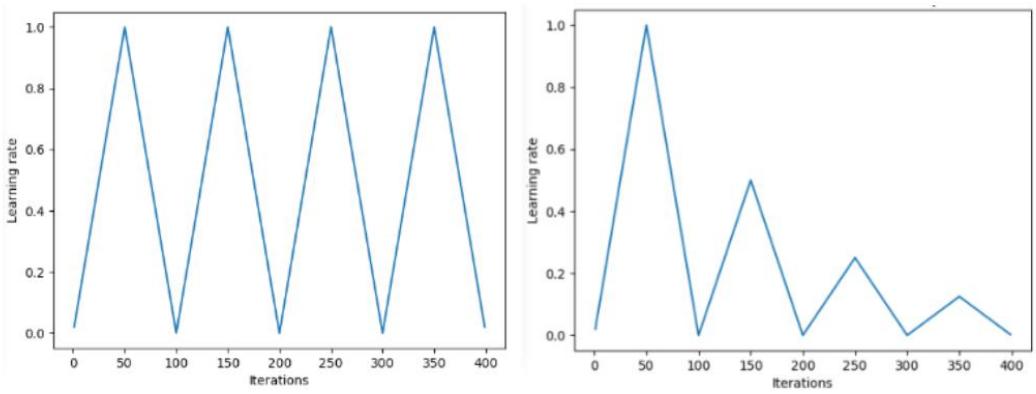

图 4.6 学习率循环形式

Fig.4.6 Learning rate cycle

第三种方式是余弦退火方法，这个也可以看作是学习率衰减的一种方式，如今也被普遍使用。余弦退火利用余弦函数来降低学习率，随着训练过程的延长，学习率首先缓慢下降，再加速下降，然后再缓慢下降，这种下降形式可以产生很好的效果，在很多深度学习库中有可以直接调用的函数来实现这个算法，比如 Fast.ai 库中的 learn.fit（）函数。

当然，学习率设定的策略还有很多，比如还有 Loshchilov 和 Hutter 提出的热重启随机梯度下降（SGDR）、基于 Armijo 准则的线性回溯搜索算法[49]等，本文将不进行逐一介绍。

第 5 章　可视化算法与总结

通过以上对于各改进算法的理论分析和应用的对比等，接下来将利用 Python[50]针对部分应用比较广泛的算法进行编程实现可视化操作，通过一些具体的数据集来绘图直观展现不同算法的区别。

5.1　重要的 SGD

传统的三种梯度下降算法本质上来说是没有太大区别的，但 SGD 算法由于训练速度较快，在很多的大规模机器学习中得到运用，接下来就以一个很简单的小例子展现 SGD 算法区别于另外两种算法的特点。令将要拟合的函数为 $f(x)=0.5x+1$，由生成随机数的函数随机生成带噪声的样本点数据，接下来用这三种算法分别进行拟合并分析不同。

首先是 BGD 算法，该算法拟合的结果是：

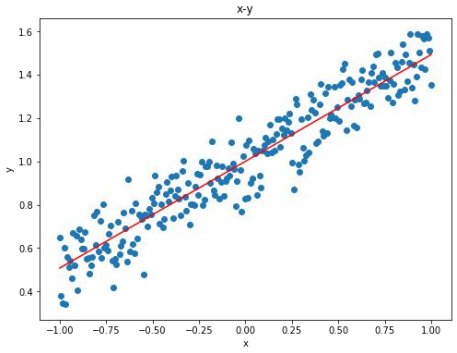

图 5.1 BGD 拟合结果

Fig.5.1 BGD fitting results

从图中可见，拟合结果不错，进一步分析其训练过程中的损失以及迭代过程如下：

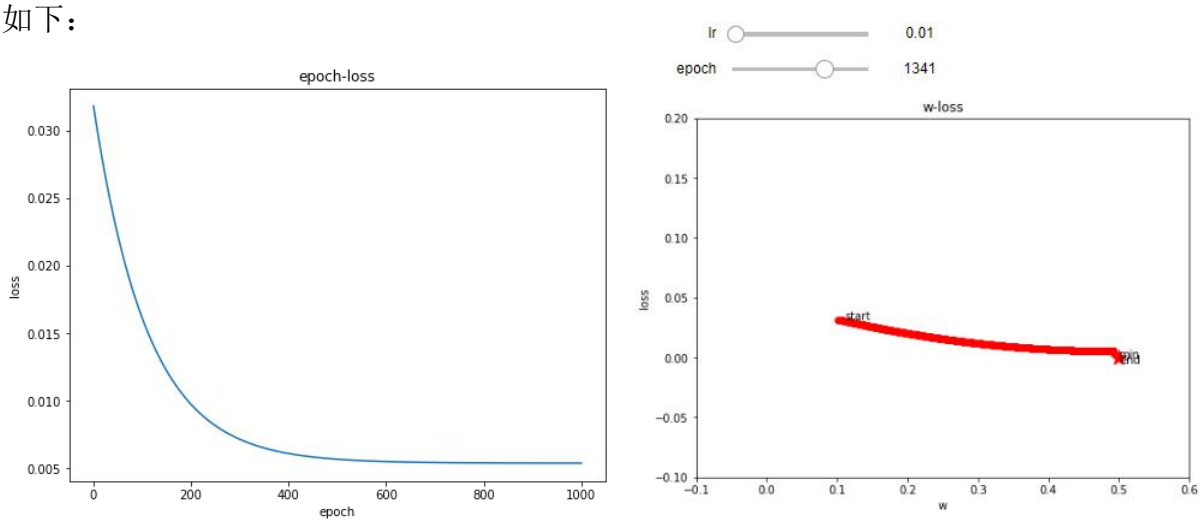

图 5.2 BDG 损失及迭代过程

Fig.5.2 BDG loss and iterative process

从训练过程的图像可见，BGD 算法所需要的迭代次数很多，但是训练过程中的损失函数是比较平稳的，没有出现大幅度的波动。

第二个运用 MBGD 算法进行拟合：

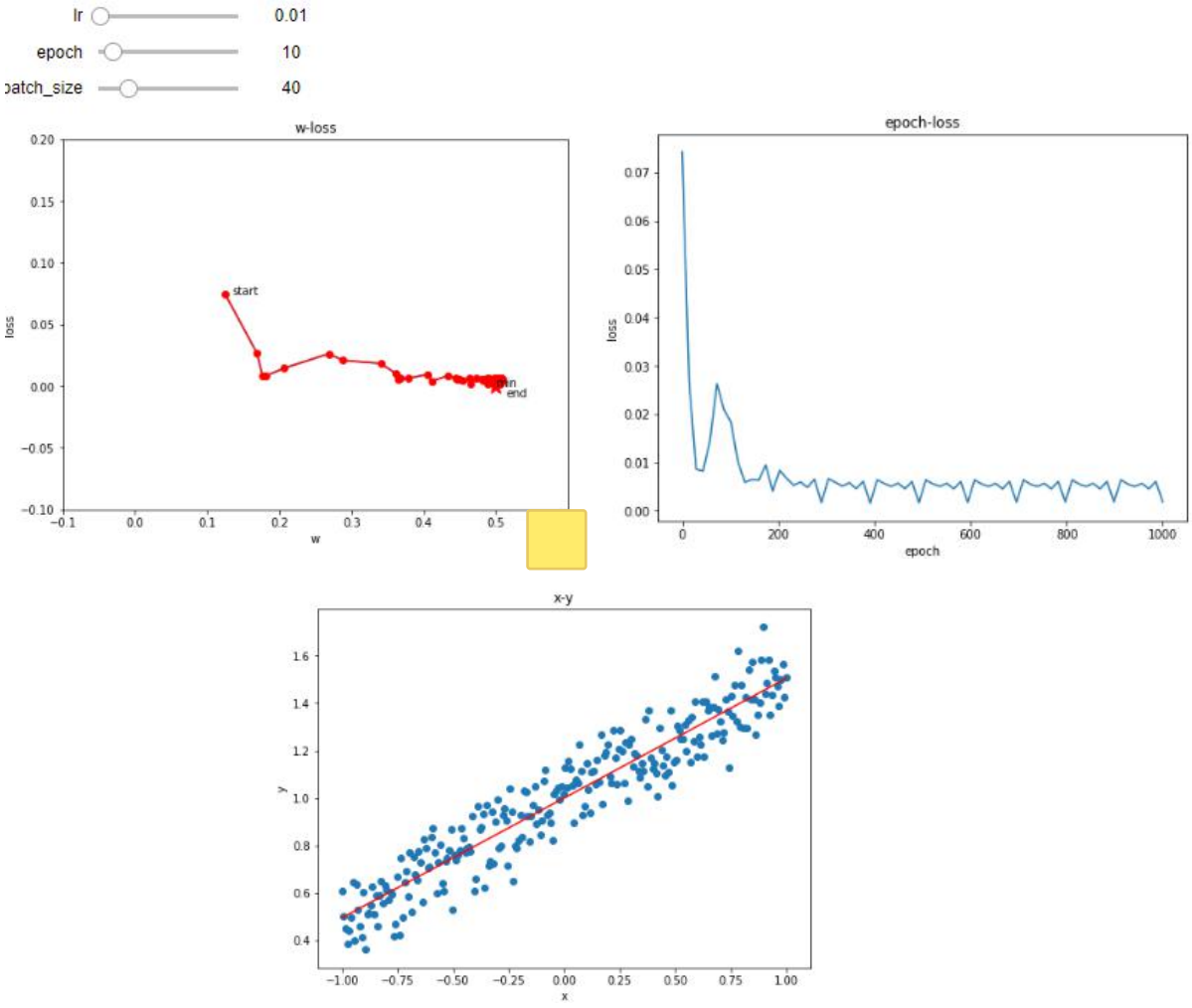

图 5.3 MBGD 算法运行结果

Fig.5.3 MBGD algorithm running result

从训练结果图可见，MBGD 算法的波动性明显强于 BGD 算法，从拟合结果来看差别不大，但是训练过程中损失函数的波动性是比较大的，当然这也与所选取的批量的大小有关。

接下来就是 SGD 算法：

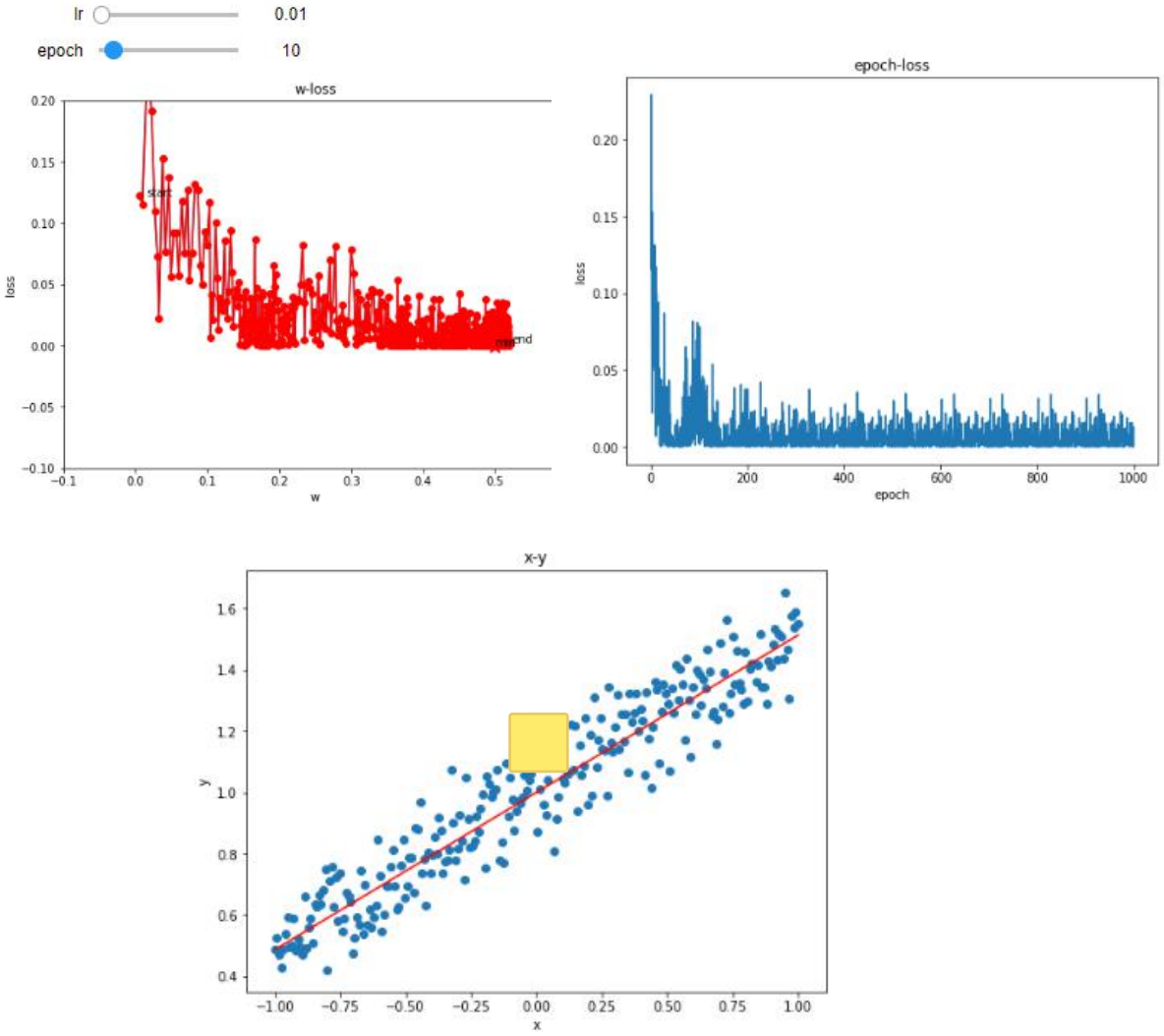

图 5.4 SGD 算法运行结果

Fig.5.4 SGD algorithm running result

从 SGD 算法运行的结果图可见，该算法的波动性很大，即使迭代次数比较小，损失函数的震荡仍很厉害，从拟合结果看，该算法的拟合效果要差一点。正也正证实了前文的理论研究分析，但是由于大数据时代下所研究问题的数据集很大，虽然 SGD 算法的波动性很大，但是训练速度快，并且总体而言其训练结果是可以收敛得，于是有了很多关于 SGD 算法的改进措施，在上文的介绍中，基本上所有的改进算法都是基于 SGD 进行改进的，让 SGD 算法可以更好地适应如今的研究应用。

5.2 动量的引入

上文专门针对动量对改进算法进行了研究，动量的引入加快了算法的训练速度，但是动量的设定不同对于算法的最终结果影响也是不同的，下文结合有关资

料的分析得到如图 5.5：

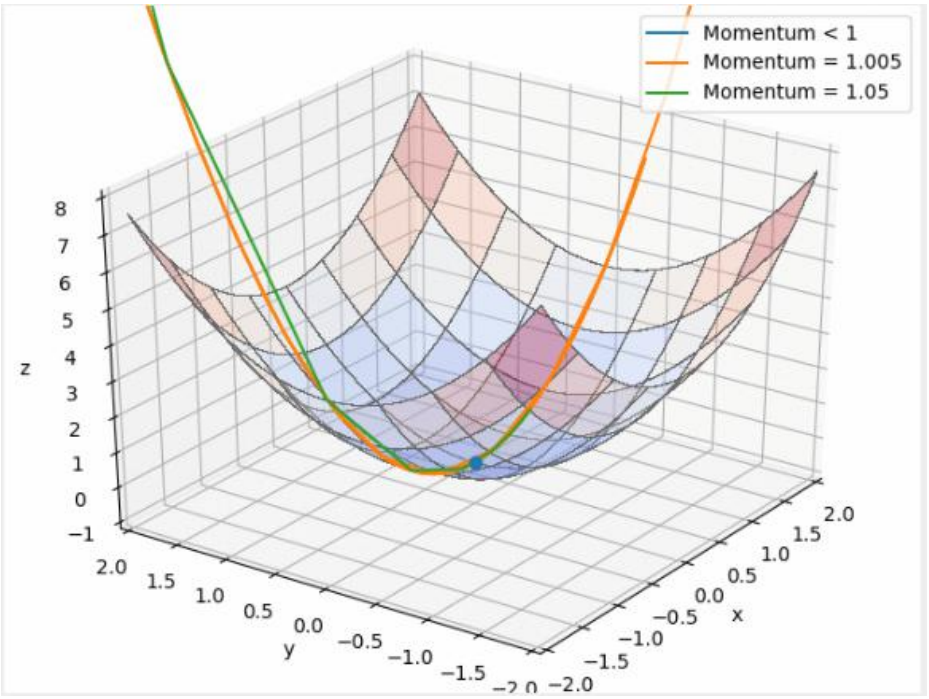

图 5.5 momentum 不同设定

Fig.5.5 Different momentum settings

根据图像可以得到，在动量的引入中，不能将其权值给的超过 1，数值超过 1 则会导致结果不收敛，尤其是在如今流行的 Adam 算法中，对 Adam 算法中衰减率和步长因子等的控制要尽量准确，如果实在不会设定就可以直接使用各大深度学习库中的系统默认值。

5.3 各优化器可视化

5.3.1 鞍点处的表现

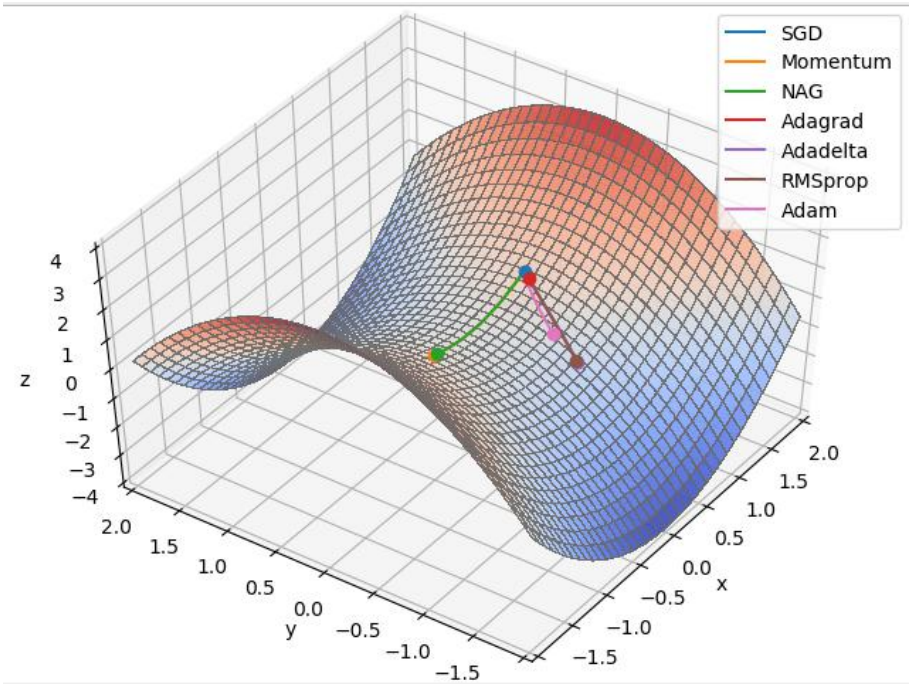

图 5.6 鞍点处各优化器表现 1

Fig.5.6 Performance of each optimizer at the saddle point 1

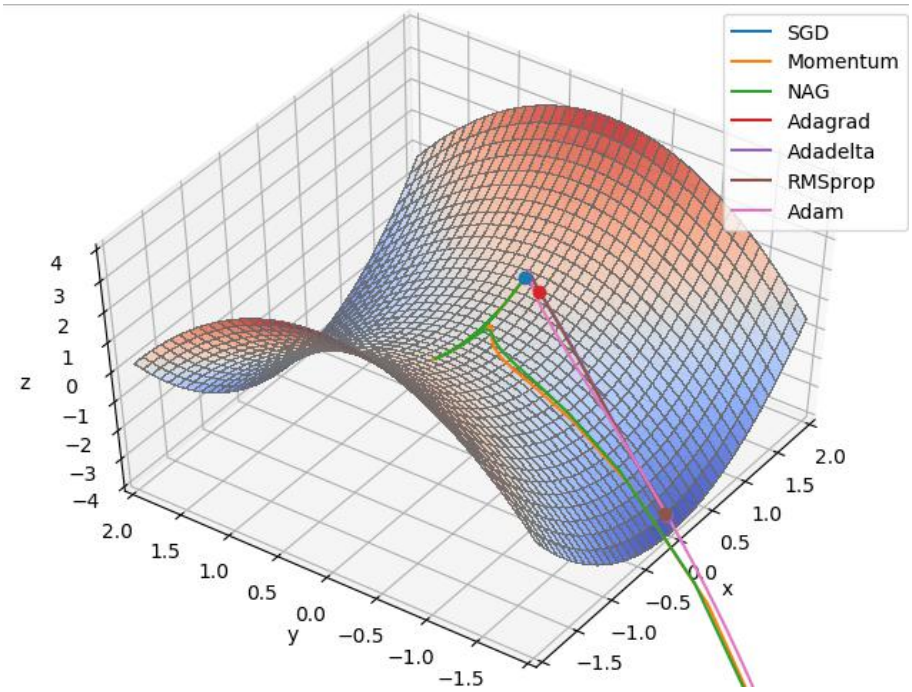

图 5.7 鞍点处各优化器表现 2

Fig.5.7 Performance of each optimizer at the saddle point 2

这是一个带鞍点的曲面，这些算法的不同表现如图所示，根据图形分析可得：

（1）SGD 算法进入了鞍点并停留在了鞍点处，没能跳过鞍点；

（2）三个自适应学习率优化器没有进入鞍点，其中，AdaDelta 算法下降最

快，Adagrad 和 RMSprop 则齐头并进；

（3）两个动量优化器 Momentum 和 NAG 算法进入了鞍点，但是在鞍点处抖动了一会儿后逃离了鞍点并迅速下降；

（4）Adam 算法在此数据集下也没有进入鞍点，并且可以看出该算法是最佳的优化器，结合上文中的研究分析，更加证实了 Adam 算法的优势，在如今很多的研究中得以应用。

5.3.2 普通曲面上的表现

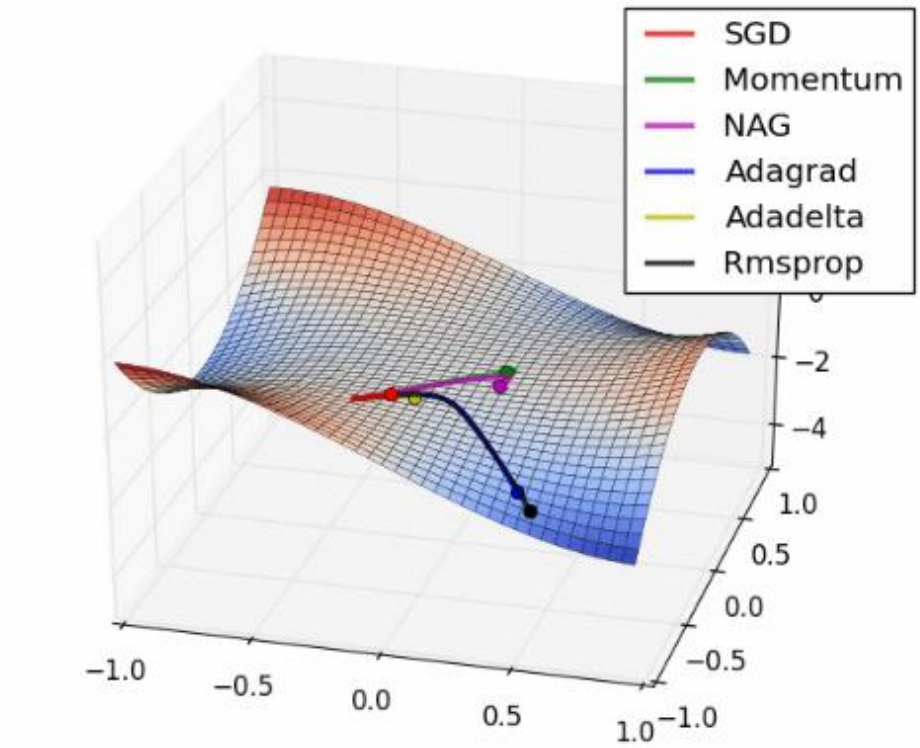

图 5.8 曲面上的优化器表现 1

Fig.5.8 Optimizer performance on the surface 1

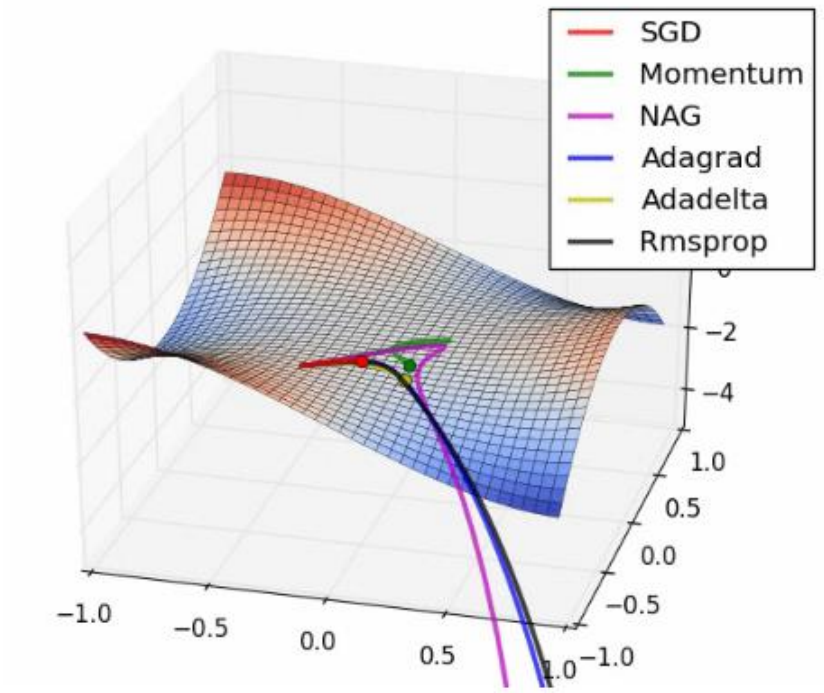

图 5.9 曲面上的优化器表现 2

Fig.5.9 Optimizer performance on the surface 2

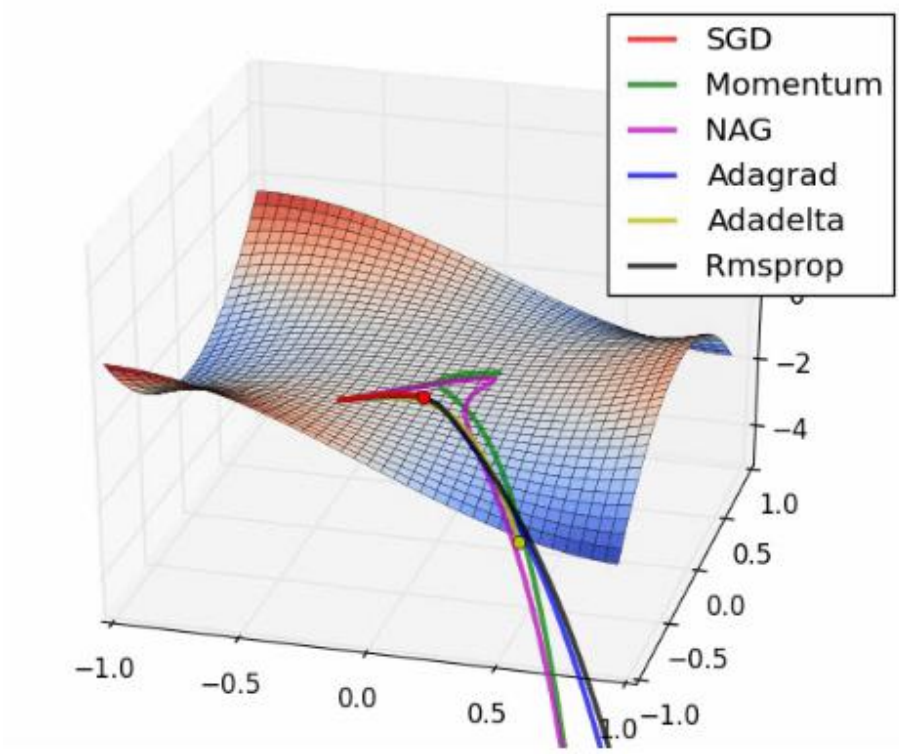

图 5.10 曲面上的优化器表现 3

Fig.5.10 Optimizer performance on the surface 3

此图描述了在一个曲面上 6 种优化器的具体表现，从图中可以得到：

（1）下降速度上：三个自适应学习优化器 Adagrad、RMSProp 与 AdaDelta 的下降速度明显比 SGD 要快，其中，Adagrad 和 RMSProp 齐头并进，要比 AdaDelta 要快；两个动量优化器 Momentum 和 NAG 由于刚开始走了岔路，初期下降的慢；随着慢慢调整，下降速度越来越快，其中 NAG 到后期甚至超过了领先的 Adagrad 和 RMSProp。

（2）下降轨迹上：SGD 和三个自适应优化器轨迹大致相同；两个动量优化器初期走了“岔路”，后期也调整了过来。

第 6 章　并行和分布式研究

随着大数据时代的来临，机器学习所研究的问题越来越复杂，仅仅只是利用上述单机梯度下降算法来进行研究已经不能满足实际的需求，于是有了机器学习的并行与分布式的有关研究。在机器学习中常用的优化算法除了上述的梯度下降算法以外，还有很多其他的算法包括二阶优化算法、交替方向乘子（alternating direction method of multipliers，简称 ADMM）算法等等[13]，这些算法可以解决不同类型的优化问题，从而拓宽了机器学习的应用领域。在深度学习的分布式训练中，一般分为数据并行和模型并行两类，数据并行是指把训练数据分到不同的工作节点上分别进行训练，它又分为同步更新和异步更新；而模型并行是指加模型的不同部分分别在不同的设备上进行训练，这样整个集群就形成了一种分布式算法，当然这两种方式也可以一起使用形成混合形式。接下来，下文将基于目前的有关研究对并行和分布式算法进行研究对比分析。

6.1　基本概念

6.1.1　参数服务器

参数服务器的概念最早是来自 Alex Smola 于 2010 年提出的并行 LDA 框架[51]-[52]，它是一个用来同步不同工作节点上的任务的方法，相当于一个通讯接口。无论是分布式中的数据并行还是模型并行，不同工作节点上的模型参数需要同步才能够使得最终训练结果收敛，参数服务器通常采用数学封装的形式来进行参数同步，比如向量、矩阵、张量等。

在进行分布式训练时，选择怎样的参数同步协议和参数的更新机制会对训练的最终结果产生重要的影响，研究应用中针对不同的数据集或者模型的复杂度可以选择不同的参数协议，会产生不一样的效果。一般情况下，参数同步协议分为三种：第一种是 BSP 协议，基于此协议，每一个工作节点完成一次迭代后将参数传给参数服务器之后就暂停等待下一次迭代，然后参数服务器基于这些结果对全局参数进行更新，更新完成后再传递给各工作节点进行下一次迭代，这种形式会产生多余的时间成本，但是其准确性是比较高的，常见的 Spark、GraphX 等框架就使用了这种房事；第二种协议是异步并行（Asynchronous Parallel,ASP）协议，此时工作节点向参数服务器传递了更新参数之后会立刻得到更新的全局参数，可见这种方式比上一种方式速度上快很多，但是这可能会导致梯度过期的现象，即模型更新参数与对应的梯度不同，从而出现收敛不有效的情况，如今 Tensorflow 和 Disbelief 框架就是使用这种协议；第三种协议名称是延迟同步并行（Stale Synchronous Parallel,SSP）协议，该协议相当于对 ASP 协议进行了优化，在协议中选取一个阈值，只要各工作节点的速度最大值和速度最小值之差没有超

过这个阈值，就用异步的方式训练，只有超过之后才会进行严格同步，很明显这种算法保留了 ASP 协议的优势，同时也使得算法的收敛更加有效。

6.1.2 同步更新

参数更新的过程可以简单表示为：参数服务器收集每台 worker 计算的梯度来更新参数，然后把更新到的参数再发到各个 worker，紧接着各个 worker 再重新进行下一次迭代。在同步更新中，每一台 worker 计算了一小批量样本的梯度后，得到相应的矩阵，并把其传递给参数服务器，参数服务器会等待所有的 worker 都已完成传送之后对所有梯度值求平均来更新参数，然后再将新的参数值传回每台 worker 并这样继续训练。同步更新的信息传输开销很大，若有一台 worker 的 GPU 较差，那么就会产生巨大的通信成本并且有短板效应，从而使得训练时间可能比单机时还要慢，但是这种方式的优点是模型收敛比较稳定，因为其使用了很大的批量从而使得这个批梯度下降的效果更加接近整体的梯度下降效果。

6.1.3 异步更新

而异步更新[57]中，对于每一次迭代，每台 worker 计算好参数的梯度后就把结果矩阵传递给参数服务器，参数服务器在接受到任意一台 worker 的矩阵后就会立即更新参数并把结果发回 worker，紧接着再根据新的参数选取同样数量的样本计算梯度来进行下一次迭代。异步更新时参数服务器不需要再等待全部的 worker 都计算好梯度后再更新，从而没有了同步更新中出现的短板效应问题，但是异步更新出现了过期梯度的问题，可以这样理解，worker1 完成传送后得到了第 100 版本的参数并开始进行下一次计算，而这时 worker2 刚刚将自己计算的梯度结果船到参数服务器，那么参数服务器中的参数将更新到第 101 版本，而这时 worker1 还在进行计算，以此类推，还有很多其他的 worker 可能也会出现类似问题，这种过期梯度的情况会导致梯度下降过程变得不稳定，因为模型在更新参数时使用的梯度不是它这一步实际算出来的梯度，但是梯度下降本身就是一个求近似解的过程，所以一般情况下最后训练仍会收敛，当然这对最后的精确度是有影响的。

6.2 一些流行的算法和框架

6.2.1 Hogwild!

由于 SGD 算法具有占用内存小、抗噪声和训练速度快等优点，它在研究应用中表现的性能较好，尤其是对于数据密集型的机器学习问题，于是在数据的大规模化时，有些人提出了并行化 SGD 策略，但是这些方案需要破坏性能的内存锁定和同步，于是 Feng Hiu 等人[3]利用新颖的理论研究分析提出了一种无需任何锁定的并行新算法－－Hogwild!，它是属于一种改进的异步的 SGD 算法。

该算法具体的运行步骤为：首先需要所有处理器假设存在一个共享内存（里面包含决定变量 x），所有处理器都可以访问决定变量 x，于是该算法对于每一个单独的处理器的更新准则为：从 M 中随机均匀得选取 m 样本，读取这些样本的当前状态并计算相应的梯度，然后对于每一个 $a \in m$ 有：

$$x_a \leftarrow x_a - \gamma b_a^T G_m(x) \tag{6.1}$$

其中 γ 为步长并且是固定的常数，b_a 表示 R^n 中的标准基础元素之一。该无锁算法相比于之前的并行算法收敛率虽然仍然是属于次线性收敛，但是大于之前的并行算法。Hogwild!算法可以在很多个 CPU 上并行执行 SGD 算法，但是这个算法适用于稀疏（高维）数据集，每一次训练只会对一部分参数进行修改或者更新，这样就不会出现重写的情况。

6.2.2 BigDL

在如今的大数据时代下，人工智能的逐步发展与突破将深度学习带入了一个新的前沿，很多已有的深度学习框架已经不能满足现有的需求，于是有关研究人员提出了很多深度学习的新的算法和框架，在此研究热门下，Intel 的有关研究人员提出了一种基于 Spark[55]的分布式框架 BigDL[4]-[5]，它利用已有的 Spark 集群来进行深度学习有关训练并且简化了存储在 Hadoop[57]中的大数据集的数据加载，该算法主要用于大数据平台，可以直接在现有的 Hadoop 以及 Spark 的集群上运行而不需要对集群做任何的修改，更有利于用户的使用。

由于该项目是建立在 Spark 框架之上的，所以它提供了更加全面的深度学习算法支持，并且它利用 Intel 已有的英特尔数学内核库（Intel MKL）以及高级编程等技术，在 Intel 的 Xeon 服务器上可以达到很高的性能，它甚至比一些开源深度学习库（比如 Caffe, TensorFlow 等）的训练速度快得到达数量级。

BigDL 框架还提供了基于 OpenCV[58]的图像预处理库，可以用于图像转换和增强等，用户还可以调用该库提供的 OpenCV 操作功能自定义图像转换，同时它让用户可以直接在 Hadoop 或者 Spark 框架下使用深度学习进行大数据分析，这样就可以用非常少的代码进行有关的研究从而更加有利于深度学习的应用和发展。

该框架由于是 Intel 内部研究开源的，它对计算机的要求（比如运行性能等）比较高，并且开源时间短，发展不太成熟，对 GPU 并没有良好的支持，所以 BigDL 是具有一定的研究价值的。目前，BigDL 技术可以在包括 AWS、阿里云和京东云等几乎所有的公有云平台上使用，随着研究的推进，该技术一定会有一个更好的发展前景。

6.2.3 Downpour SGD

Downpour SGD[6]是当下引用量比较大的一种形式，它是 Google 开发的 DistBelief 框架（该框架用来训练超大规模的深度神经网络）下的一种异步并行随机梯度下降法，这种方法利用分布式思想和参数服务器，同时实现了模型并行和数据并行，最关键的是该算法结合了前面提到的 Adagrad 算法，所以相比之下，该算法得到了明显的加速。

该算法的核心思想是把数据随机划分成多个子数据集并把模型变量划分到数个机器并对子数据集进行训练，然后主节点再更新模型变量，这个过程中各个训练是相互独立互不干扰的。

然后正因为该算法结合了自适应学习率的 Adagrad 算法，这极大地提高了 Downpour SGD 的鲁棒性[59]-[61]，并且使得可以训练的模型副本数量的增加。但是由于训练时各副本之间互不干扰，所以可能会出现参数发散的现象。

6.2.4 小结

在本小节中只选择了几种当下比较流行的分布式框架和算法进行了介绍，但事实上有关并行和分布式算法和框架的研究很多，比如还有异步并行算法——AASGD 算法[59]-[61]，这个算法适用于大规模稠密集或者高维数据集；还有 Google 开源的 Tensorflow 框架，这个框架的使用比较多，它主要用于实现和部署大规模机器学习的模型，这个框架如今已包含了很多深度学习模块，可以直接使用该系统中的函数或者模块内容等，方便了很多的大型研究；类似的算法还有很多。并且其研究不仅仅只针对梯度下降算法，在其他的优化算法中也有很多的并行和分布式研究，比如并行与分布式二阶优化算法等。正因为这些并行和分布式研究，让如今大数据时代下深度学习的研究更加深入，从而推动了人工智能的发展，使得未来更加可期。

第 7 章　总结与展望

7.1　本文总结

本文研究了非常重要的梯度下降算法的改进算法，一方面根据其提出者的引入思路对算法的重要部分进行了分析和评价，另一方面通过编程实现部分算法的图像可视化，直观展现各算法的区别。本文工作主要是基于理论的研究分析，梯度下降算法的变式主要是由国外有关研究人员提出来的，很多深度学习库是自带各种算法的使用框架的，所以本文的工作从一定角度上弥补了深度学习的部分空缺。但是由于更加偏向于理论研究，没有结合当下存在的深度学习库进行具体的细节研究，所以本文仍有完善的空间。

7.2　未来展望

限于机器设备的欠缺，本文对于并行和分布式的研究没有结合实际的训练来展现，所以没能对各个并行和分布式算法进行更加实际具体的对比分析，希望在以后条件允许的情况下，可以实际地运用这个方式，来看看它对于大规模数据的处理形式以及训练结果等。同时由于时间有限，对于 Tensorflow 的研究很少，希望未来可以进一步学习这个流行的深度学习框架，并利用它操作一些有关深度学习的具体案例。

参 考 文 献

[1] 马世龙,乌尼日其其格,李小平. 大数据与深度学习综述[J].智能系统学报,2016,11(06):728-742.

[2] 张慧. 深度学习中优化算法的研究与改进[D].北京邮电大学,2018.

[3] Niu, F., Recht, B., Christopher, R., & Wright, S. J. (2011). Hogwild! : A Lock-Free Approach to Parallelizing Stochastic Gradient Descent, 1–22.

[4] 章敏敏,徐和平,王晓洁,周梦昀,洪淑月. 谷歌 TensorFlow 机器学习框架及应用[J].微型机与应用,2017,36(10):58-60.

[5] 樊雅琴,王炳皓,王伟,唐烨伟.深度学习国内研究综述[J].中国远程教育,2015(06):27-33+79.

[6] Sebastian Ruder,Aylien Ltd.,Dublin. An Overview of Gradient Descent Optimization Algorithms. ar Xiv preprint ar Xiv:1609.04747,2017

[7] 张建明,詹智财,成科扬,詹永照. 深度学习的研究与发展[J].江苏大学学报(自然科学版),2015,36(02):191-200.

[8] Yuanming Zhou, Shifeng Zhao, Xuesong Wang, et al. Deep Learning Model and Its Application in Big Data. 2018.

[9] 张建华. 基于深度学习的语音识别应用研究[D].北京邮电大学,2015.

[10] 余滨,李绍滋,徐素霞,纪荣嵘. 深度学习:开启大数据时代的钥匙[J].工程研究-跨学科视野中的工程,2014,6(03):233-243.

[11] 舒娜,刘波,林伟伟,李鹏飞. 分布式机器学习平台与算法综述[J].计算机科学,2019,46(03):9-18.

[12] 郭跃东,宋旭东. 梯度下降法的分析和改进[J].科技展望,2016,26(15):115+117.

[13] 亢良伊,王建飞,刘杰,叶丹. 可扩展机器学习的并行与分布式优化算法综述[J].软件学报,2018,29(01):109-130.

[14] Nesterov, Y. (1983). A method for unconstrained convex minimization problem with the rate of convergence o(1/k2). Doklady ANSSSR (translated as Soviet.Math.Docl.), vol. 269, pp. 543–547.

[15] 景立森,丁志刚,郑树泉,等. 基于 NAG 的 BP 神经网络的研究与改进[J].计算机应用与软件,2018,35(11):272-277.

[16] Nicolas Le Roux, Mark Schmidt, and Francis Bach. A Stochastic Gradient Method with an Exponential Convergence Rate for Strongly-Convex Optimization with Finite Training Sets. arXiv preprint arXiv:1202.6258, 2012.

[17] Rie Johnson, Tong Zhang. Accelerating Stochastic Gradient Descent using Predictive Variance Reduction.2013.

[18] Duchi, J., Hazan, E., & Singer, Y. (2011). Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. Journal of Machine Learning Research, 12, 2121 – 2159. Retrieved from http://jmlr.org/papers/v12/duchi11a.html.

[19] 常永虎,李虎阳. 基于梯度的优化算法研究[J].现代计算机,2019(17):3-8+15.

[20] Huo,Z.,Huang,H. Asynchronous mini-batch gradient descent with variance reduction for non-convex optimization[J].USA:THIRTY-FIRST AAAI CONFERENCE ON ARTIFICIAL INTELLIGENCE,2017:2043-2049.

[21] Kingma, D. P., & Ba, J. L. (2015). Adam: a Method for Stochastic Optimization. International Conference on Learning Representations, 1 – 13.

[22] Zhang, S., Choromanska, A., & LeCun, Y. (2015). Deep learning with Elastic Averaging SGD. Neural Information Processing Systems Conference (NIPS 2015), 1 – 24. Retrieved from http://arxiv.org/abs/1412.6651.

[23] 李成华,张新访,金海,向文. MapReduce:新型的分布式并行计算编程模型[J].计算机工程与科学,2011,33(03):129-135.

[24] 孙科. 基于 Spark 的机器学习应用框架研究与实现[D].上海交通大学,2015.

[25] 李姚舜,刘黎志. 逻辑回归中的批量梯度下降算法并行化研究[J].武汉工程大学学报,2019,41(05):499-503+510.

[26] Ioffe,S.,Szegedy,C.(2015). Batch Normalization:Accelerating Deep Network Training by Reducing Internal Covariate Shift. ar Xiv Preprint ar Xiv:1502.03167v3.

[27] 袁冰清,陆悦斌,张杰. 神经网络与深度学习基础[J].数字通信世界,2018(05):32-33+62.

[28] 魏运才. 最优化梯度法的改进[J].工科数学,1996(01):77-79.

[29] N. Yağmur and B. B. Alagöz, "Comparision of Solutions of Numerical Gradient Descent Method and Continous Time Gradient Descent Dynamics and Lyapunov Stability," 2019 27th Signal Processing and Communications Applications Conference (SIU), Sivas, Turkey, 2019, pp. 1-4.

[30] 叶晓芸,秦鉴. 论浅层学习与深度学习[J].软件导刊,2006(02):19-21.

[31] 周安众,罗可. 一种卷积神经网络的稀疏性 Dropout 正则化方法[J].小型微型计算机系统,2018,39(08):1674-1679.

[32] 杨观赐,杨静,李少波,胡建军. 基于 Dopout 与 ADAM 优化器的改进 CNN 算法[J].华中科技大学学报(自然科学版),2018,46(07):122-127.

[33] 钱程. 基于深度学习的人脸识别技术研究[D].西南交通大学,2017.

[34] Geoffrey Hinton,Simon Osindero,Max Welling,Yee – Whye Teh. Unsupervised Discovery of Nonlinear Structure Using Contrastive Backpropagation[J]. Cognitive Science,2006,30(4).

[35] 陆汝铃. 知识科学及其研究前沿[J]. 中国青年科技, 2000, 8(6):48-51.

[36] 高君宇,杨小汕,张天柱,徐常胜. 基于深度学习的鲁棒性视觉跟踪方法[J].计算机学报,2016,39(07):1419-1434.

[37] 邓乃扬, 诸梅芳. 最优化计算方法[M]. 中国人民解放军空军工程学院, 1983.

[38] 孙娅楠,林文斌. 梯度下降法在机器学习中的应用[J].苏州科技大学学报(自然科学版),2018,35(02):26-31.

[39] Ashia C. Wilson, Rebecca Roelofs, Mitchell Stern, Nathan Srebro, Benjamin Recht. The Marginal Value of Adaptive Gradient Methodsin Machine Learning[C]. 31st Conference on Neural Information Processing Systems(NIPS)，2017.

[40] NeilZhang. 监督学习——随机梯度下降算法(sgd)和批梯度下降算法(bgd)[OL].[2018.2.20].https://www.cnblogs.com/NeilZhang/p/8454890.html.

[41] Gabidullina, Z. R. (2019). Adaptive Conditional Gradient Method. Journal of Optimization Theory and Applications. doi:10.1007/s10957-019-01585-w

[42] Ning Qian. On the momentum term in gradient descent learning algorithms[J]. Neural Networks,1999,12(1).

[43] A.Botev, G.Lever and D.Barber, "Nesterov's accelerated gradient and momentum as approximations to regularised update descent," 2017 International Joint Conference on Neural Networks (IJCNN), Anchorage, AK, 2017, pp. 1899-1903, doi: 10.1109/IJCNN.2017.7966082.

[44]Matthew D. Zeiler.ADADELTA:AN ADAPTIVE LEARNING RATE METHOD. https://arxiv.org/pdf/1212.5701.pdf

[45] H. Robinds and S. Monro, “A stochastic approximation method,” Annals of Mathematical Statistics, vol. 22, pp. 400–407, 1951.

[46] Kingma D, Ba J. Adam:A Method for Stochastic Optimization[J]. Computer Science, 2014.

[47] 李兴怡,岳洋. 梯度下降算法研究综述[J].软件工程,2020,23(02):1-4.

[48] 严晓明. 一种逻辑回归学习率自适应调整方法[J].福建师范大学学报(自然科学版),2019,35(03):24-28.

[49] 孟继东,马燕青,张冰. Armijo 型线搜索下一个修正 Hestenes-Stiefel 共轭梯度法的全局收敛性[J].内江师范学院学报,2012,27(04):27-30.

[50] 费良宏. Deep Learning with Python[M]. Apress, 2017.

[51] 唐淳. 分布式随机梯度下降算法研究[D]. 电子科技大学, 2018.

[52] Dean, J., Corrado, G. S., Monga, R., Chen, K., Devin, M., Le, Q. V, … Ng, A. Y. (2012). Large Scale Distributed Deep Networks. NIPS 2012: Neural Information Processing Systems, 1–11. http://doi.org/10.1109/ICDAR.2011.95.

[53] Bottou, L.,Curtis, F. E., & Nocedal, J. (2018). Optimization Methods for Large-Scale Machine Learning. SIAM Review, 60(2), 223–311. doi:10.1137/16m1080173.

[54] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition; 2014. arXiv preprint. arXiv:1409.1556.

[55] 吴信东,嵇圣硙. MapReduce 与 Spark 用于大数据分析之比较[J].软件学报,2018,29(06):1770-1791.

[56] Chao Liu,Pingyu Jiang,Wenlei Jiang. Web-based digital twin modeling and remote control of cyber-physical production systems[J]. Robotics and Computer-Integrated Manufacturing,2020,64.

[57] 李明. 基于 Spark 的分布式深度学习系统的研究与实现[D].电子科技大学,2019.

[58] Kennedy, R.K.L., Khoshgoftaar, T.M., Villanustre, F. et al. A parallel and distributed stochastic gradient descent implementation using commodity clusters. J Big Data 6, 16 (2019). https://doi.org/10.1186/s40537-019-0179-2

[59] Shiliang Sun,Jing Zhao,Jiang Zhu. A review of Nyström methods for large-scale machine learning[J]. Information Fusion,2015,26.

[60] Gonglin Yuan,Tingting Li,Wujie Hu. A conjugate gradient algorithm for large-scale nonlinear equations and image restoration problems[J]. Applied Numerical Mathematics,2020,147.

[61] 冯辉. 网络化的并行与分布式优化算法研究及应用[D].复旦大学,2013.

[62] Han Zhang, Gang Wu, Qing Ling. Distributed stochastic gradient descent for link prediction in signed social networks[J].EURASIP Journal on Advances in Signal Processing, 2019, Vol.2019 (1), pp.1-11.DOI:10.1186/s13634-019-0601-0

致　　谢

本论文是在武国宁老师和孙娜老师的悉心指导下完成的，从最开始论文的选题、研究思路和创新点的确定，到过程中的问题答疑以及论文的审阅，老师们都倾注了很多的精力和心血，真的很感谢老师们的耐心指导和关心，在这里向老师们表示最诚挚的感谢！特别感谢从大一就认识的武国宁老师，武老师学识渊博、工作兢兢业业、对学生认真负责并且总是耐心的回答同学们的问题、为人谦和，老师从大一开始就给予了我很多帮助和支持，他不仅仅是我的老师，更是我的榜样，在即将毕业之际，向武老师再次表示最真挚的谢意和最崇高的敬意！

感谢当时转专业时各位老师对我的肯定和支持，让我得以在自己喜欢的数学专业提升自己，感谢每一个曾耐心指导我的老师，正因为您们的指点和教诲，才让我一点点地进步。

感谢一直支持鼓励我的闺蜜刘彦君和罗霞，感谢一直宠着我的金倩倩、黄庆梅、王莹，感谢带给我无限温暖的小学妹王鑫，感谢在 610、213 和 336 宿舍遇到的各位室友们，谢谢大学遇到的每一位朋友和贵人，我的大学正是有了你们才这么丰富多彩。

感谢给予我支持的家人们，谢谢您们的关心和照顾，以后的路我会努力好好走，为您们带来一个更好的生活！

最后，感谢咱们中国石油大学（北京），是你让我来到北京，让我拥有现在的所有。时间真的太快，舍不得挥手再见，所有美好的点滴我都会铭记在心，不悔选择，未来可期！