Computer Science and Application 计算机科学与应用, 2019, 9(2), 351-358
Published Online February 2019 in Hans. http://www.hanspub.org/journal/csa
https://doi.org/10.12677/csa.2019.92040

Hans汉斯

Ada_Nesterov Momentum Algorithm

—The Nesterov Momentum Algorithm with Adaptive Learning Rate

Xibin Jia, Jiashuai Shi

Faculty of Information Technology, Beijing University of Technology, Beijing
Email: shijiashuai@126.com

Received: Jan. 29th, 2019; accepted: Feb. 8th, 2019; published: Feb. 15th, 2019

Abstract

The Nesterov momentum algorithm can efficiently improve the gradient descent direction. However, all parameters of the Nesterov momentum algorithm share the same learning rate, and the learning rate needs to be set by workers. Adadelta algorithm has the characteristic of adaptive learning rate, and each dimension parameter has independent learning rate. Therefore, we firstly deduced the learning rate formula of each dimension, which was based on Adadelta algorithm. Secondly we introduced it into the Nesterov momentum algorithm. And finally, Ada_Nesterov momentum algorithm was proposed. To verify Ada_Nesterov momentum algorithm, two experiments were designed, which indicated that with momentum parameter 0.5, and VggNet_16, the CIFAR_100 dataset, Ada_Nesterov momentum algorithm achieved the highest test and evaluation accuracy, the lowest valuation loss, and fastest rate of convergence. Therefore, Ada_Nesterov momentum algorithm improved Nesterov momentum algorithm with adaptive learning rate.

Keywords

Nesterov Momentum Algorithm, Adadelta Algorithm, Ada_Nesterov Momentum Algorithm, Adaptive Learning Rate

Ada_Nesterov动量法

——一种具有自适应学习率的Nesterov动量法

贾熹滨，史佳帅

北京工业大学信息学部，北京
Email: shijiashuai@126.com

收稿日期：2019年1月29日；录用日期：2019年2月8日；发布日期：2019年2月15日

摘　要

Nesterov动量法可以很好地改进梯度下降方向，但是其所有参数都具有相同的学习率，并且学习率需要

文章引用: 贾熹滨, 史佳帅. Ada_Nesterov 动量法[J]. 计算机科学与应用, 2019, 9(2): 351-358.
DOI: 10.12677/csa.2019.92040

人为设定。Adadelta算法可以自适应学习率，并且每维参数具有独立的学习率。因此，本文首先基于Adadelta算法推导出每一维的学习率公式，其次将其带入Nesterov动量法中，得到了Ada_Nesterov动量法。为了验证提出的Ada_Nesterov动量法，本文设计了两个实验。实验结果表明：动量参数0.5时，Ada_Nesterov动量法在VggNet_16神经网络架构上，基于CIFAR_100数据集的验证准确率最高，损失最小，收敛速度最快。即Ada_Nesterov动量法改进了Nesterov动量法，具有自适应学习率。

关键词

Nesterov动量法，Adadelta算法，Ada_Nesterov动量法，自适应学习率

1. 引言

优化算法是深度学习的一个重要组成部分，近年来深度学习能够取得瞩目成绩的一个原因为其优化算法的改进。在求解深度学习网络时，优化算法按照反向求导梯度的阶数可以分为一阶方法(梯度下降法[1]、随机梯度下降法[2]、动量法[3]等)和二阶近似法(牛顿法[4] [5]、共轭梯度法[6] [7]和存储受限的BFGS(L-BFGS)算法等)。但是由于二阶近似方法计算量很大，对于高维度的数据的优化效果不好，所以本文主要研究一阶方法。

深度学习中一阶梯度优化算法的改进均为以梯度下降法为基础，对其梯度下降方向或者学习率进行改进。Nesterov 动量法可以很好地改进梯度下降方向，但是其所有参数都具有相同的学习率，并且学习率需要人为设定。Adadelta 算法具有自适应学习率的功能，但是在训练后期存在梯度下降方向不准确的现象。因此，本文结合 Nesterov 动量法的梯度下降策略和 Adadelta 算法的自适应学习率策略，提出了具有自适应学习率，且梯度下降方向准确的 Ada_Nesterov 动量法。

2. 深度学习优化算法比较

2.1. 梯度下降法

深度学习优化算法求解损失函数最小化的问题，其基础的算法为梯度下降法(Gradient Descent, GD)[1]。在深度学习领域，梯度下降法又被称为批量梯度下降法(BGD)。在批量梯度下降法中，学习率可以理解为梯度下降法的步长，梯度可以理解为梯度下降法的下降方向，学习率决定以多大的步长更新参数。批量梯度下降法在每次迭代的时候，通过计算数据集上所有数据来完成一次迭代，该算法可以得到较准确的局部最优解或者全局最优解，但是随着数据集的增多，算法速度会越来越低。同时，该算法不能在训练过程中添加数据。

2.2. 随机梯度下降法

为了解决批量梯度下降法的求解速度问题，Bottou [2]于 1998 年提出了随机梯度下降法。与批量梯度下降法不同的是随机梯度下降法首先把数据集分成 m 个小批量样本，然后通过计算他们的梯度均值，计算梯度无偏估计，最后进行参数迭代。当 $m = 1$ 时，称为在线梯度下降法；当 $m > 1$ 时，称为小批量随机梯度下降法。

随机梯度下降法的优点为收敛速度快，其计算时间不取决于数据集的大小。对于大容量的数据集，往往在没有全部计算完数据之前，其收敛精度已经达到规定的范围。随机梯度下降法的缺点为需要人为调参。由于在随机梯度下降法随机采样 m 个训练样本，引入了梯度估计的噪声源(在极点处不消失)，因此随机梯度下降法不能像批量梯度下降法固定学习率的数值，而是要根据实验结果人为的调整学习率的大小。

2.3. 动量法

随机梯度下降法虽然比梯度下降法的收敛速度提高，但是在接近最优点时有时很难通过陡谷，即在一个维度上的表面弯曲程度远大于其他维度的区域[3]。为了提高随机梯度下降法的收敛速度 Polyak 等人[8]提出了动量法。

与随机梯度下降法不同的是动量法在速度向量上乘以了一个超参数 $\alpha \in [0,1)$。和学习率一样，超参数也需要不断调整。对于梯度点处具有相同方向的维度时，超参数增大；相反，超参数减小。从而，通过减少摇摆，提高收敛速度[9]。

2.4. Nesterov 动量法

动量法梯度下降改进了梯度的方向，但是该梯度方向具有一定的盲目性，不能预测以后的梯度方向是上升还是下降。受 Nesterov [10] [11]影响，Sutskever 等人[12]提出了动量算法的一个变种——Nesterov 动量法。Nesterov 动量法在动量法的梯度计算之前，对参 θ 进行了校正，其更新规则如(公式(1)、公式(2))所示：

$$\theta \leftarrow \theta + \Delta\theta \tag{1}$$

$$\Delta\theta = \partial\theta - \eta\nabla_\theta\left(\frac{1}{m}\sum_{i=1}^{m} L\left(f\left(x^{(i)};\theta\right), y^{(i)}\right)\right) \tag{2}$$

实验表明在凸批量梯度的情况下，Nesterov 动量法将额外误差收敛率从 $O(1/k)$ (k 步后)改进到$(1/k^2)$[12]。但是，在随机梯度的情况下，Nesterov 动量法没有改进收敛率。

2.5. Adagrad 算法

由于损失通常敏感于参数空间的某些方向，动量法可以改变梯度下降的方向，但是却引入了超参数。基于 Delta-bar-delta 算法[13]中自动适应模型参数各自学习率的方法，Duchi 等人[14]提出了 Adagrad 算法。该算法基于一种很简单的思想：具有损失最大偏导的参数相应地有一个快速下降的学习率，而具有小偏导的参数在学习率上有相对较小的下降，其更新规则如公式(3)~公式(6)所示：

计算梯度：

$$g \leftarrow \nabla_\theta\left(\frac{1}{m}\sum_{i=1}^{m} L\left(f\left(x^{(i)};\theta\right), y^{(i)}\right)\right) \tag{3}$$

累积平方梯度：

$$r \leftarrow r + g \odot g \tag{4}$$

通过逐元素地应用除和求平方根，计算 θ 更新：

$$\Delta\theta \leftarrow -\frac{\varepsilon}{\xi + \sqrt{r}} \odot g \tag{5}$$

应用更新：

$$\theta \leftarrow \theta + \Delta\theta \tag{6}$$

Adagrad 算法具有良好的理论依据，但是由于其在分母中累加了梯度的平方根，在训练过程中累加会持续增长，导致学习率变小。在学习率无限小时，Adagrad 算法将无法获得额外的信息。Adagrad 算法在某些模型[15] [16]上可以得到很好的收敛，但不是全部。为了解决批量梯度下降法的求解速度问题，Bottou [2]于 1998 年提出了随机梯度下降法。与批量梯度下降法不同的是随机梯度下降法首先把数据集分成 m 个小批量样本，然后通过计算他们的梯度均值，计算梯度无偏估计，最后进行参数迭代(公式(3))。当 $m = 1$ 时，称为在线梯度下降法；当 $m > 1$ 时，称为小批量随机梯度下降法。

随机梯度下降法的优点为收敛速度快，其计算时间不取决于数据集的大小。对于大容量的数据集，往往在没有全部计算完数据之前，其收敛精度已经达到规定的范围。随机梯度下降法的缺点为需要人为调参。由于在随机梯度下降法随机采样 m 个训练样本，引入了梯度估计的噪声源(在极点处不消失)，因此随机梯度下降法不能像批量梯度下降法固定学习率的数值，而是要根据实验结果人为的调整学习率的大小。

2.6. Adadelta 算法

Adadelta 算法[17]是 Adagrad 算法的扩展，较 Adagrad 算法的改进有以下几点：a) 将梯度的平方递归地表示成所有历史梯度平方的均值，而不是计算所有的梯度的平方(公式(7))；

$$E\left[g^2\right]_t = \gamma E\left[g^2\right]_{t-1} + (1-\gamma) g_t^2 \tag{7}$$

b) 为了解决非凸优化问题，Adadelta 算法更新规则中设计参数具有相同的假设单位，首次定义了另一个指数衰减均值，不是梯度平方，而是参数平方的更新(公式(8))；

$$E\left[\Delta\theta^2\right]_t = \gamma E\left[\Delta\theta^2\right]_{t-1} + (1-\gamma)\Delta\theta_t^2 \tag{8}$$

改进后的 Adadelta 算法的更新规格为：

$$\Delta\theta_t = -\frac{RMS\left[\Delta\theta\right]_{t-1}}{RMS\left[\Delta g\right]_t} \cdot g_t \tag{9}$$

Adadelta 算法只需要小的计算量，并且一直是一阶梯度计算。此外，Adadelta 算法不需要调节学习率。在模型训练初期和中期，Adadelta 算法具有很好的收敛速度，但是在训练后期，Adadelta 算法会反复在局部最小值附近抖动，很难脱离局部最小值吸引盆。这时如果换 Adadelta 算法为 Nesterov 动量法，就会发现模型测试准确率将提高 5%左右，说明 Adadelta 算法在局部最小值附近时，梯度下降方向不准确。

此外，为了使 Adam 算法和 AdaGrad 算法有更快的收敛速度，张慧[18]在 Adam 算法和 AdaGrad 算法中引入了动量的思想，提出了带动量的 Adam 算法和 Adarad 算法：AMM 算法。

3. Ada_Nesterov 动量法定义

Ada_Nesterov 动量法在 Nesterov 动量法的基础上添加了 Adadelta 算法的自适应学习率策略。Adadelta 算法在 Adagrad 算法基础上做出来两个方面的改进，首先，为解决 Adagrad 算法在训练后期学习率非常小的问题，将梯度的平方递归地表示成所有历史梯度平方的均值，而不是计算所有的梯度的平方(公式(8))，得到公式(10)：

$$\Delta\theta_t = -\frac{\eta}{RMS\left[\Delta g\right]_t} \cdot g_t \tag{10}$$

其次，AdaDelta 算法通过计算指数衰减的累积梯度平方和(公式(8))，解决了 SGD 算法、Adagrad 算

法、动量法等算法在参数更新时单位不一致的问题，得到梯度下降的公式(9)。比较公式(10)和公式(9)，得出每一维的学习率为指数衰减均值的均方根(公式(11))：

$$\eta = RMS[\Delta\theta]_{t-1} \tag{11}$$

把公式(11)带入 Nesterov 动量法的梯度下降公式(2)，得 Ada_Nesterov 动量法的梯度下降公式(12)：

$$\Delta\theta = \rho[\Delta\theta]_{t-1} - RMS[\Delta\theta]_{t-1}\nabla_\theta\left(\frac{1}{m}\sum_{i=1}^{m}L\left(f\left(x^{(i)};\theta\right),y^{(i)}\right)\right) \tag{12}$$

Ada_Nesterov 动量法的具体更新规则如算法 1 所示：

算法 1：Ada_Nesterov 动量法

设置： $\rho = 0.9,\ \varepsilon = 1,\ \partial = 0.5$

初始化变量：θ

初始化参数：$\mathrm{E}[\Delta\theta^2]_0 = 0$

For t=1:T，循环：

计算指数衰减均值： $\mathrm{E}[\Delta\theta^2]_t = \rho\mathrm{E}[\Delta\theta^2]_{t-1} + (1-\rho)\Delta\theta^2{}_t$

求均方根： $RMS[\Delta\theta]_t = \sqrt{\mathrm{E}[\Delta\theta^2]_t + \varepsilon}$

更新： $\theta \leftarrow \theta + \Delta\theta$

计算下一步骤参数：$\Delta\theta = \partial[\Delta\theta]_{t-1} - RMS[\Delta\theta]_{t-1}\nabla_\theta\left({}^1\!/_m\sum_{i=1}^{m}L\left(f\left(x^{(i)};\theta\right),y^{(i)}\right)\right)$

End For

步骤 1：设置动量因子 ρ，常数项，衰减系数 α；初始化假设函数变量 θ，初始化梯度下降的初始期望的平方为零。

步骤 2：在设定的循环次数内，计算指数衰减均值，并对其求平方根，更新假设函数变量 θ，并计算下一步梯度下降参数。

步骤 3：判断循环是否达到终止条件，如果未达到终止条件，则循环步骤 2，如果达到了终止条件，则算法终止。

4. 实验验证

本文对 Ada_Nesterov 动量法进行了两个实验，实验一构建了一个简单的卷积神经网络的网络训练，以确定 Ada_Nesterov 动量法的最优动量因子。实验二构建了一个神经网络，并训练了 CIFAR-100 数据集，比较 Nesterov 动量法，Adadelta 算法和 Ada_Nesterov 动量法的性能。

4.1. 实验一

为了确定 Ada_Nesterov 动量法的最优动量因子，本文构建了一个具有两个卷积层，两个全连接层的简单卷积神经网络。此神经网络卷积层的卷积核大小为 5 × 5，第二个卷积成后进行 dropout 操作，激活函数为 ReLU [19]。训练和测试数据集为 MNIST 数据集。试验平台为 OS X EI Capitan 系统，Intel Core i5 主板，8G 内存，tensorflow 为 1.2.1 CPU 版本。

Ada_Nesterov 动量法的常数项为 0.00001，衰减系数 α 为 0.001。在经过 1 个 epoch 后，不同的动量因子，得到的测试正确率和损失结果如表 1 所示：

Table 1. The results of Ada_Nesterov momentum algorithm with different parameters
表 1. Ada_Nesterov 动量法在不同的动量因子下的测试结果

动量因子	正确率	损失
0.1	94%	0.2187
0.2	95%	0.1898
0.3	95%	0.1849
0.4	96%	0.1502
0.5	97%	0.1378
0.6	96%	0.1748
0.7	95%	0.1833
0.8	88%	0.4458
0.9	10%	2.3045

由表 1 可以看出，在影响因子 0.5 时，Ada_Nesterov 动量法的正确率最高(97%)，损失最小(0.1378)；在影响因子 0.9 时，Ada_Nesterov 动量法的正确率最低(10%)，损失最大(2.3.45)。

也可以看到随着动量因子在 0.5 附近其正确率和损失大小比较稳定，在动量因子减小或者增大时，正确率均会降低，损失均增加。同时，动量因子减小正确率缓慢降低，损失缓慢增加；但是，动量因子增大正确率急剧降低，损失急剧增加。

综上所述，动量因子在 0.5 时，Ada_Nesterov 动量法的正确率最高，损失最小。本文在进行实验二时，对动量因子 0.5 的 Ada_Nesterov 动量法进行验证。

4.2. 实验二

为了验证训练了 Ada_Nesterov 动量法，Nesterov 动量法和 Adadelta 算法的性能，本文基于 VggNet-16 网络架构，训练了 CIFAR-100 数据集。三种算法均设置 64 的批处理量，训练 6000 步。图 1 和图 2 分别为三种算法的验证正确率曲线和验证损失曲线。

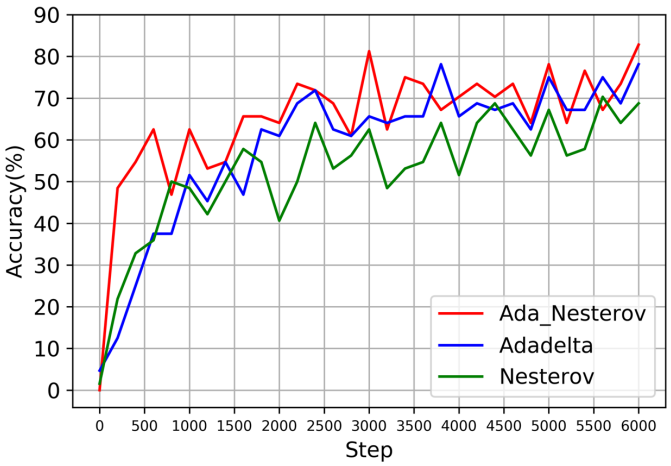

Figure 1. The accuracy of the three optimization algorithms on CIFAR-100 datasets
图 1. 三种优化算法在 CIFAR-100 数据集上的验证准确率

由图 1 可知，Ada_Nesterov 动量法的验证准确率曲线基本上处于 Adadelta 算法和 Nesterov 动量法之

上，即 Ada_Nesterov 动量法的验证准确率整体上高于 Adadelta 算法和 Nesterov 动量法。在训练 6000 步时，三种算法的准确率均为最高，Ada_Nesterov 动量法，Adadelta 算法和 Nesterov 动量法的验证准确率分别为：82.8%，78.1%和 68.8%。

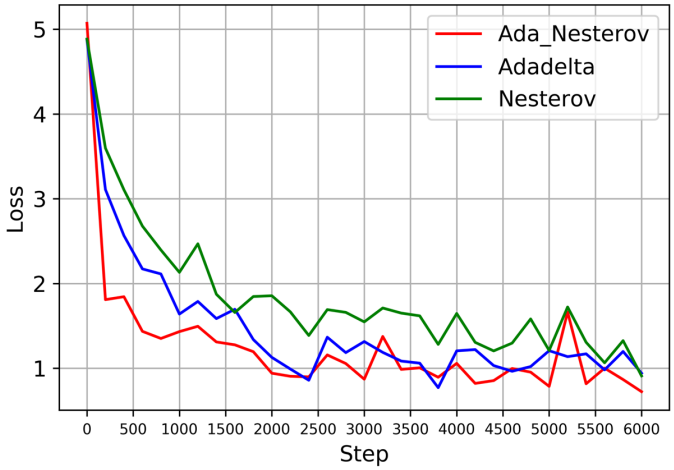

Figure 2. The loss of the three optimization algorithms on CIFAR-100 datasets
图 2. 三种优化算法在 CIFAR-100 数据集上的验证损失

由图 2 可知，Ada_Nesterov 动量法的验证损失曲线基本上处于 Adadelta 算法和 Nesterov 动量法之下，即 Ada_Nesterov 动量法的验证损失整体上高于 Adadelta 算法和 Nesterov 动量法。在训练 6000 步时，三种算法的损失均为最低，Ada_Nesterov 动量法，Adadelta 算法和 Nesterov 动量法的验证损失分别为：0.72、0.94 和 0.91。

本文对训练得到的各神经网络模型，基于 CIFAR-100 测试数据集进行了测试，测试结果如表 2 所示。由表 2 可知，基于 Nesterov 动量法、Adadelta 算法和 Ada_Nesterov 动量法训练得到的模型的测试正确率分别为：78.5%、75.2%和 61.7%；测试损失分别为：1.13、1.58 和 2.17。

此外，从三种优化算法的运行时间分析，运行 6000 时，Ada_Nesterov 动量法，Adadelta 算法和 Nesterov 动量法耗时分别约为 16 小时，19 小时和 10 小时。可见，Adadelta 算法复杂度最高，Nesterov 动量法复杂度最低，Ada_Nesterov 动量法复杂度介于 Adadelta 算法和 Nesterov 动量法之间。

Table 2. System resulting data of standard experiment
表 2. 各算法测试正确率和测试损失

算法	测试正确率	测试损失
Nesterov 动量法	61.7%	2.17
Adadelta 算法	75.2%	1.58
Ada_Nesterov 动量法	78.5%	1.13

综上所述，动量因子 0.5 的 Ada_Nesterov 动量法可以给每一维数据一个自适应学习率，算法复杂度适中，具有验证准确率高，损失小，收敛速度快，是一个综合性能较好的优化算法。

5. 结论

Nesterov 动量法可以很好的改变动量法上梯度的下降方向，但是 Nesterov 动量法的每一维度都具有相同

的学习率，本文借助 Adadelta 算法的自适应学习率的策略，提出了自适应学习率的 Ada_Nesterov 动量法。为了验证 Ada_Nesterov 动量法的性能，本文设计了两个实验。实验一为确定 Ada_Nesterov 动量法的最优的动量因子 ρ。我们分别在动量因子 0.1~0.9 的情况下，基于 MNIST 数据集训练了一个简单的卷积神经网络。对神经网络模型进行测试的结果表明：动量因子 0.5 时 Ada_Nesterov 动量法具有最高的测试正确率和损失。实验二为 Ada_Nesterov 动量法(动量因子 0.5)与 Adadelta 算法和 Nesterov 动量法的性能比较。我们在 VggNet_16 神经网络架构上，基于 CIFAR_100 数据集，分别对三种算法进行了训练，实验结果表明 Ada_Nesterov 动量法的准确率最高，损失最小，收敛速度最快。即 Ada_Nesterov 动量法改进了 Nesterov 动量法的自适应学习率。

基金项目

北京工业大学国际科研合作种子基金项目资助(项目编号：2018A02)。

参考文献

[1] Cauchy, A. (1847) Méthode générale pour la résolution des systemes d'équations simultanées. *Comp. Rend. Sci. Paris*, **25**, 536-538.

[2] Bottou, L. (1998) Online Learning and Stochastic Approximations. *On-Line Learning in Neural Networks*, **17**, 142.

[3] Sutton, R.S. (1986) Two Problems with Backpropagation and Other Steepest-Descent Learning Procedures for Networks. In: *Proceedings of the* 8*th Annual Conference of the Cognitive Science Society*, Erlbaum, Hillsdale, NJ, 823-831.

[4] Levenberg, K. (1944) A Method for the Solution of Certain Non-Linear Problems in Least Squares. *Quarterly of Applied Mathematics*, **2**, 164-168. https://doi.org/10.1090/qam/10666

[5] Marquardt, D.W. (1963) An Algorithm for Least-Squares Estimation of Nonlinear Parameters. *Journal of the Society for Industrial and Applied Mathematics*, **11**, 431-441. https://doi.org/10.1137/0111030

[6] Møller, M.F. (1993) Efficient Training of Feed-Forward Neural Networks. *DAIMI Report Series*, **22**, No. 464.

[7] Le Roux, N., Bengio, Y. and Fitzgibbon, A. (2011) 15. Improving First and Second-Order Methods by Modeling Uncertainty. In: Sra, S., Nowozin, S. and Wright, S.J., Eds., *Optimization for Machine Learning*, The MIT Press, Cambridge, MA, 403.

[8] LeCun, Y., Boser, B., Denker, J.S., *et al*. (1989) Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Computation*, **1**, 541-551. https://doi.org/10.1162/neco.1989.1.4.541

[9] LeCun, Y.A., Bottou, L., Orr, G.B., *et al.* (2012) Efficient Backprop. In: *Neural Networks*: *Tricks of the Trade*, Springer Berlin Heidelberg, 9-48.

[10] Nesterov, Y. (1983) A Method of Solving a Convex Programming Problem with Convergence Rate O (1/k2). *Soviet Mathematics Doklady*, **27**, 372-376.

[11] Nesterov, Y. (2013) Introductory Lectures on Convex Optimization: A Basic Course. Springer Science & Business Media.

[12] Sutskever, I., Martens, J., Dahl, G., *et al*. (2013) On the Importance of Initialization and Momentum in Deep Learning. *Proceedings of the* 30*th International Conference on Machine Learning*, **28**, 1139-1147.

[13] Jacobs, R.A. (1988) Increased Rates of Convergence through Learning Rate Adaptation. *Neural Networks*, **1**, 295-307. https://doi.org/10.1016/0893-6080(88)90003-2

[14] Duchi, J., Hazan, E. and Singer, Y. (2011) Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research*, **12**, 2121-2159.

[15] Dean, J., Corrado, G., Monga, R., *et al.* (2012) Large Scale Distributed Deep Networks. *Proceedings of the* 25*th International Conference on Neural Information Processing Systems*, **1**, 1223-1231.

[16] Pennington, J., Socher, R. and Manning, C.D. (2014) Glove: Global Vectors for Word Representation. *EMNLP*, **14**, 1532-1543.

[17] Zeiler, M.D. (2012) ADADELTA: An Adaptive Learning Rate Method. arXiv preprint arXiv:1212.5701.

[18] 张慧. 深度学习中优化算法的研究与改进[D]: [硕士学位论文]. 北京: 北京邮电大学, 2018.

[19] Glorot, X., Bordes, A. and Bengio, Y. (2011) Deep Sparse Rectifier Neural Networks. *Proceedings of the* 14*th International Conference on Artificial Intelligence and Statistics*, PMLR **15**, 315-323.

知网检索的两种方式：

1. 打开知网页面 http://kns.cnki.net/kns/brief/result.aspx?dbPrefix=WWJD
   下拉列表框选择：[ISSN]，输入期刊 ISSN：2161-8801，即可查询
2. 打开知网首页 http://cnki.net/
   左侧“国际文献总库”进入，输入文章标题，即可查询

投稿请点击：http://www.hanspub.org/Submission.aspx
期刊邮箱：csa@hanspub.org