Beijing Forest Studio
北京理工大学信息系统及安全对抗实验中心

深度学习优化算法概述

董思佳　　硕士研究生

2018年1月21日

内容提要

- 背景简介
- 基本概念
- 算法原理
- 应用总结
- 参考文献

背景简介

背景介绍

- 机器学习基础架构[1]

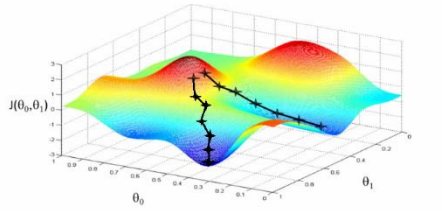
unknown target function
$f: \mathcal{X} \to \mathcal{Y}$
(ideal credit approval formula)

hypothesis set
$\mathcal{H}$
(set of candidate formula)

training examples
$\mathcal{D}: (\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_N, y_N)$
(historical records in bank)

learning algorithm
$\mathcal{A}$

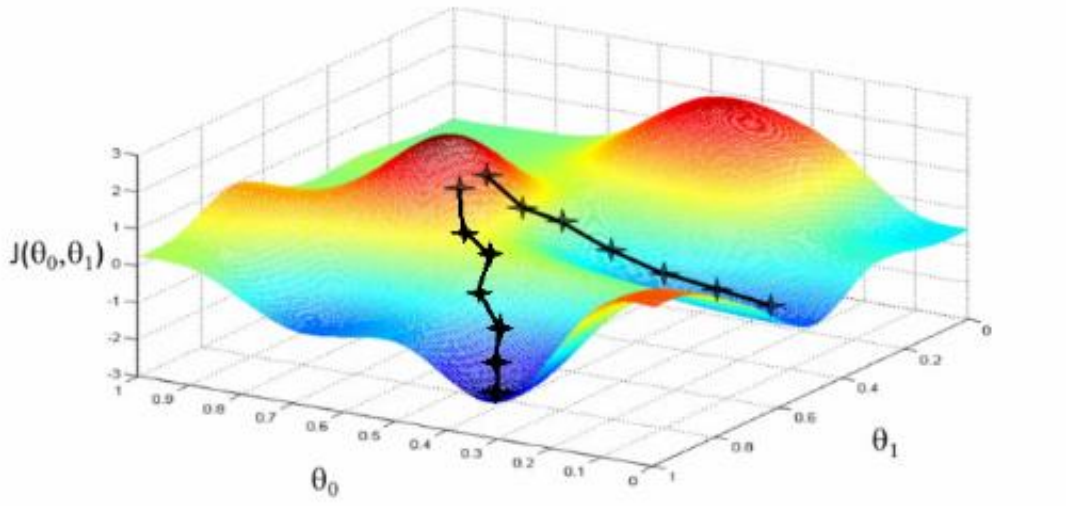
final hypothesis
$g \approx f$
('learned' formula to be used)



[1] 林轩田，机器学习基石

背景介绍

- 优化算法
  - 深度学习常用的优化算法是梯度下降(GD)，分为批量梯度下降法(BGD)、随机梯度下降法(SGD)、mini-batch梯度下降法。
  - 优化算法的发展经历了SGD -> SGDM -> NAG -> AdaGrad -> AdaDelda/RMSprop -> Adam 的过程。



基本概念

基本概念

- 优化算法的通用框架
  - 首先定义：待优化参数ω，目标函数$f(\omega)$，初始学习率α。
  - 然后开始进行迭代优化，在每个epoch t：
    - 步骤1：计算目标函数关于当前参数的梯度：

$$g_t = \nabla f(\omega_t)$$

    - 步骤2：根据历史梯度计算一阶动量和二阶动量：

$$m_t = \emptyset(g_1, g_2, \cdots, g_t)$$

$$V_t = \varphi(g_1, g_2, \cdots, g_t)$$

    - 步骤3：计算当前时刻的下降梯度：

$$\eta_t = \alpha \cdot m_t / \sqrt{V_t}$$

    - 步骤4：根据下降梯度进行更新：

$$\omega_{t+1} = \omega_t - \eta_t \qquad \text{即} \Delta\omega = -\eta_t$$

算法原理

固定学习率的优化算法

- SGD
  - 随机梯度下降法(Stochastic Gradient Descent)
  - 没有引入动量的概念
    - 步骤1：计算目标函数关于当前参数的梯度

$$g_t = \nabla f(\omega_t)$$

    - 步骤2：不引入动量：

$$m_t = \emptyset(g_1, g_2, \dots, g_t) \rightarrow m_t = g_t$$

$$V_t = \varphi(g_1, g_2, \dots, g_t) \rightarrow V_t = I^2$$

    - 步骤3：计算当前时刻的下降梯度：

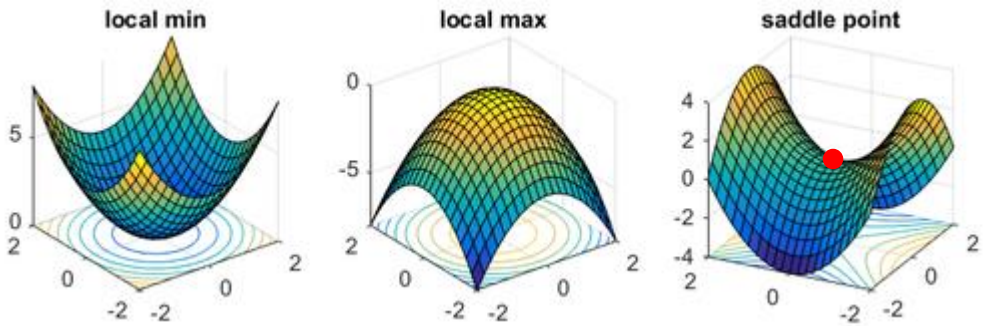
$$\eta_t = \alpha \cdot m_t / \sqrt{V_t} \rightarrow \eta_t = \alpha \cdot g_t$$

    - 步骤4：根据下降梯度进行更新：

$$\omega_{t+1} = \omega_t - \eta_t \qquad \text{即} \Delta\omega = -\eta_t$$

固定学习率的优化算法

- SGD
  - 出现问题：
    - 很难选择合适的学习率
    - 下降速度慢，可能会在沟壑两边持续震荡，陷入局部最优点
    - 在某些情况下会被困在“鞍点”(saddle point)



固定学习率的优化算法

- SGDM（ SGD with Momentum ）
  - BorisPolyak1964年提出，可以在梯度下降过程加入动量，加速SGD 在正确方向的下降并抑制震荡。
  - 在SGD基础上引入一阶动量，β_1经验值为0.9。
    - 步骤1：计算目标函数关于当前参数的梯度：

$$g_t = \nabla f(\omega_t)$$

    - 步骤2：计算一阶动量

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

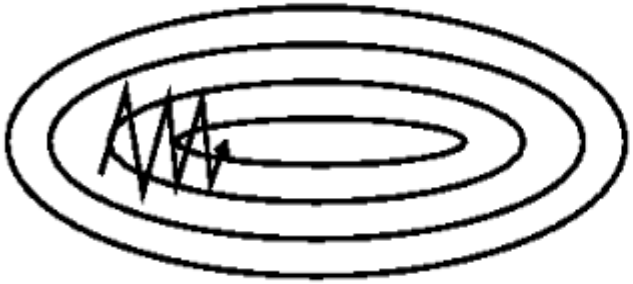
    - 步骤3：计算当前时刻的下降梯度：

$$\eta_t = \alpha \cdot m_t$$

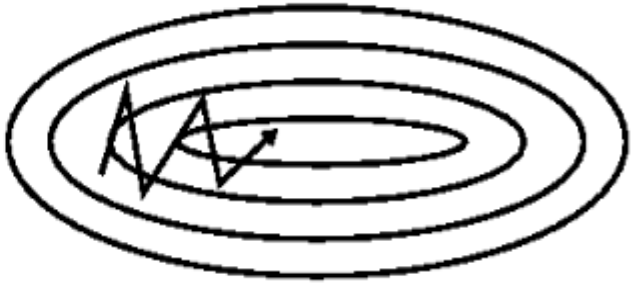
    - 步骤4：根据下降梯度进行更新：

$$\omega_{t+1} = \omega_t - \eta_t \qquad \text{即} \Delta\omega = -\eta_t$$

固定学习率的优化算法



传统SGD



SGDM

– 特点
  - 在梯度指向同一方向时会加速更新参数，而在梯度方向改变时会减小更新参数
  - 下降中后期在局部最小值附近来回震荡时，$g_t \to 0$，此时累积动量存在，可以抑制震荡

固定学习率的优化算法

- NAG（ SGD with Nesterov Acceleration ）
  - YuriiNesterov在1983年提出，在SGDM基础上改进步骤1，不计算当前位置的梯度方向，而是计算按照累积动量走了一步，那时候的下降方向。
    - 步骤1：计算梯度：

$$g_t = \nabla f(\omega_t) \rightarrow g_t = \nabla f(\omega_t - \alpha \cdot m_{t-1}/\sqrt{V_{t-1}})$$

    - 步骤2：计算一阶动量

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

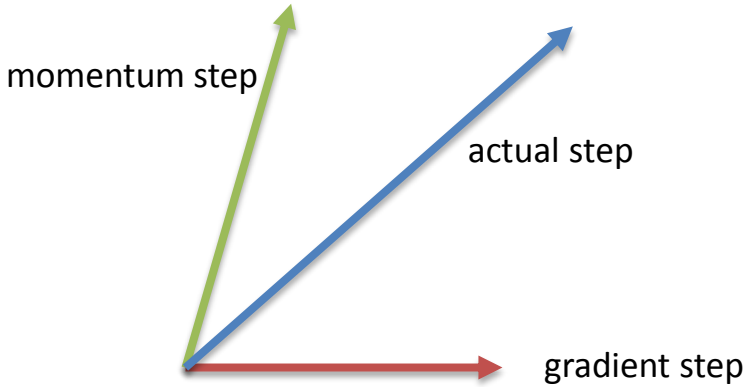
    - 步骤3：计算当前时刻的下降梯度：

$$\eta_t = \alpha \cdot m_t$$

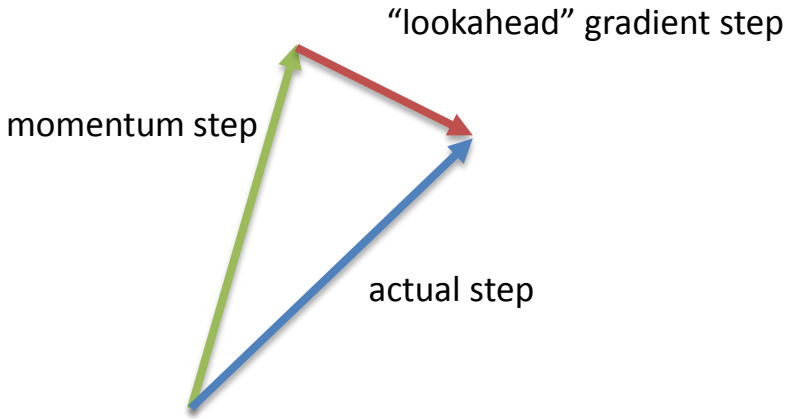
    - 步骤4：根据下降梯度进行更新：

$$\omega_{t+1} = \omega_t - \eta_t \qquad \text{即} \Delta\omega = -\eta_t$$

固定学习率的优化算法



SGDM



NAG

- 特点
  - 这种计算梯度的方式可以使算法更好的「预测未来」，提前调整更新速率
- 出现问题：
  - 以上SGD、SGDM、NAG算法对每个参数使用相同的学习率，但同一个学习率不一定适用于所有参数。

自适应学习率的优化算法

- AdaGrad
  - Duchi在2011年提出，引入二阶动量，即该维度上迄今为止所有梯度的平方和。
    - 步骤1：计算目标函数关于当前参数的梯度：

$$g_t = \nabla f(\omega_t)$$

    - 步骤2：根据历史梯度计算二阶动量

$$V_t = \sum_{\tau=1}^{t} g_\tau^2$$

    - 步骤3：计算当前时刻的下降梯度：

$$\eta_t = \alpha \cdot g_t / \sqrt{V_t} \longrightarrow \eta_t = \alpha \cdot g_t / \sqrt{V_t + \epsilon}$$

    - 步骤4：根据下降梯度进行更新：

$$\omega_{t+1} = \omega_t - \eta_t \qquad \text{即} \Delta\omega = -\eta_t$$

自适应学习率的优化算法

- AdaGrad
  - 特点
    - 适合处理稀疏特征数据
  - 出现问题：
    - 学习率单调递减，训练后期的学习率非常小，这有可能导致优化函数在收敛之前就停止更新。
    - 需要人工设置全局初始学习率

自适应学习率的优化算法

- AdaDelta
  - 改变二阶动量计算方法：不累积全部历史梯度，而只关注过去一段时间窗口的下降梯度。
    - 步骤1：计算目标函数关于当前参数的梯度：

$$g_t = \nabla f(\omega_t)$$

    - 步骤2：计算二阶动量

$$V_t = \sum_{\tau=1}^{t} g_\tau^2 \longrightarrow V_t = \beta_2 \cdot V_{t-1} + (1-\beta_2) \cdot g_t^2$$

    - 步骤3：计算当前时刻的下降梯度：

$$\eta_t = \alpha \cdot g_t / \sqrt{V_t + \epsilon}$$

    - 步骤4：根据下降梯度进行更新：

$$\omega_{t+1} = \omega_t - \eta_t \qquad \text{即} \Delta\omega = -\eta_t$$

自适应学习率的优化算法

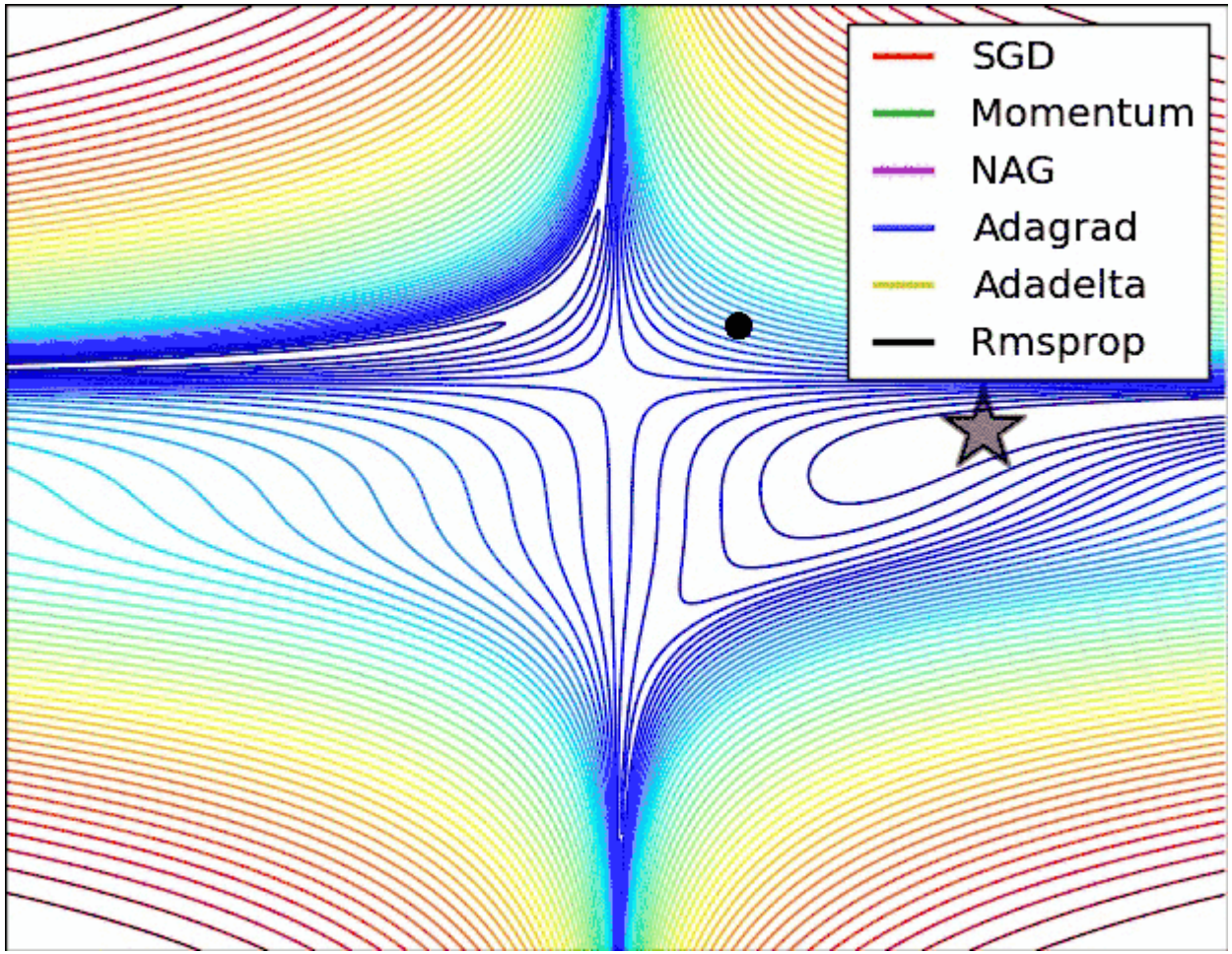
- AdaDelta
  - 经过近似牛顿迭代法之后：
    - $E|g^2|_t = \rho \cdot E|g^2|_t + (1-\rho) \cdot g_t^2$
    - $\eta_t = \alpha \cdot g_t / \sqrt{E|g^2|_t + \epsilon}$
    - $\Delta\omega = -\frac{\sqrt{E|\Delta x^2|_{t-1}}}{\sqrt{E|g^2|_t + \epsilon}} g_t$
  - 此时，Adadelta已经不依赖于全局的初始学习率

自适应学习率的优化算法

- RMSprop
  - 当ρ=0. 5时，$E|g^2|_t = \rho \cdot E|g^2|_t + (1-\rho) \cdot g_t^2$为求梯度平方和的平均数
  - 再求均方根：$RMS|g|_t = \sqrt{E|g^2|_t + \epsilon}$
  - $\Delta\omega = -\frac{\alpha}{\sqrt{E|g^2|_t+\epsilon}} g_t$
  - 特点
    - RMSprop依然依赖于全局学习率
    - RMSprop算是Adagrad的一种发展，和Adadelta的变体，效果趋于二者之间
    - 适合处理非平稳目标 - 对于RNN效果很好

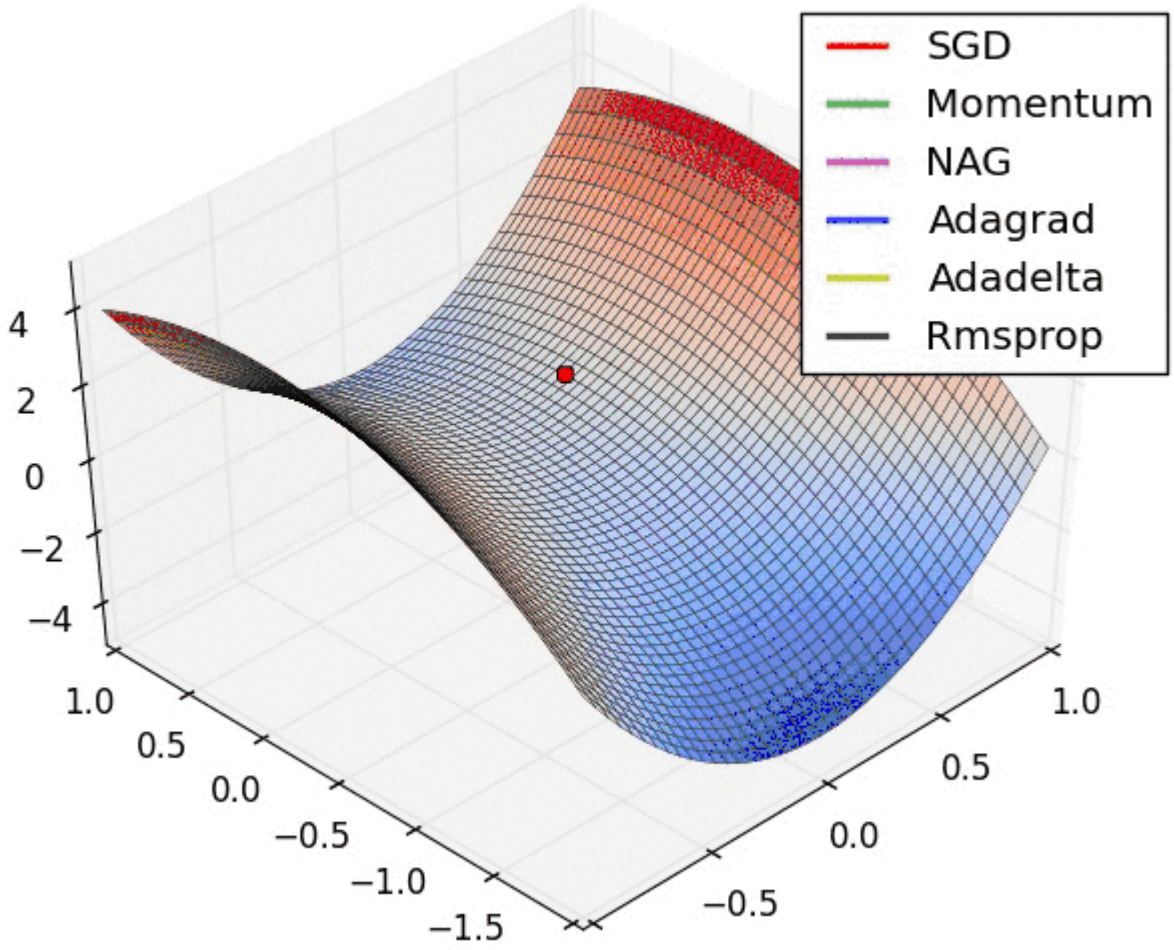
自适应学习率的优化算法

- 各优化算法在损失曲面上的表现



自适应学习率的优化算法

- 各优化方法在损失曲面鞍点处上的表现



自适应学习率的优化算法

- Adam（ Adaptive Moment Estimation ）
  - Kingma在2015年提出，本质上是带有动量项的RMSprop。
    - 步骤1：计算目标函数关于当前参数的梯度：

$$g_t = \nabla f(\omega_t)$$

    - 步骤2：计算一阶动量和二阶动量，并进行偏差校正

$$m_t = \beta_1 \cdot m_{t-1} + (1-\beta_1) \cdot g_t \qquad \widehat{m_t} = \frac{m_t}{1-{\beta_1}^t}$$

$$V_t = \beta_2 \cdot V_{t-1} + (1-\beta_2) \cdot g_t^2 \qquad \widehat{V_t} = \frac{V_t}{1-{\beta_2}^t}$$

    - 步骤3：计算当前时刻的下降梯度：

$$\eta_t = \alpha \cdot \widehat{m_t} / (\sqrt{\widehat{V_t}} + \epsilon)$$

    - 步骤4：根据下降梯度进行更新：

$$\omega_{t+1} = \omega_t - \eta_t \qquad \text{即} \Delta\omega = -\eta_t$$

自适应学习率的优化算法

- Adam（ Adaptive Moment Estimation ）
  - 特点：
    - 经过偏置校正后，每一次迭代学习率都有确定范围，使参数较为平稳
    - 结合了Adagrad善于处理稀疏梯度和RMSprop善于处理非平稳稳目标的优点
    - 对内存需求小
    - 为不同的参数计算不同的自适应学习率
    - 也适用于大多非凸优化，适用于大数据集和高维空间
  - 出现问题：
    - 可能不收敛
    - 可能错过全局最优解

应用总结

应用总结

- SGD通常训练时间更长，但是在好的初始化和学习率的基础上，训练结果更可靠
- 如果需要更快的收敛，并且需要训练较深较复杂的网络时，推荐使用自适应学习率优化方法
- 对于稀疏数据，尽量使用自适应学习率的优化方法，不用手动调节，而且最好采用默认值
- Adadelta、RMSprop、Adam是比较相似的算法，在相似的情况下表现差不多
- 一般优先考虑 Adam

算法固然美好，数据才是根本！

参考文献

参考文献

[1] Sebastian Ruder. An overview of gradient descent optimization algorithms. Eprint arXiv: 1609.04747,2016.9

[2] D.Kingma, J.Ba. Adam:A Method for Stochastic Optimization. International Conference for Learning Representations, 2015.

[3] M. Zeiler. Adadelta: An Adaptive Learning Rate Method. arXiv preprint, 2012.

[4] Sutskever, J. Martens, G. Dahl, and G. Hinton. On the Importance of Initialization and Momentum in Deep Learning. Proceedings of the 30th International Conference on Machine Learning, 2013.

[5] T. Tieleman, and G. Hinton. RMSProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning.Technical report, 2012.

道德经

知人者智，自知者明。胜人者有力，自胜者强。知足者富。强行者有志。不失其所者久。死而不亡者，寿。

谢谢！