DL4J
DEEPLEARNING4J
(../index.html)

深度学习教科书
(https://www.amazon.com/Deep-Learning-Practitioners-Adam-Gibson/dp/1491914254)
快速入门
教程
深入学习介绍
神经网络
数据与ETL
调模和训练
部署
开源社区
自然语言处理
ND4J: JVM 的 Numpy
相关资源
其他语言
与我们在Gitter聊天

Deeplearning4j更新器介绍

本页内容主要面向已经了解随机梯度下降(glossary.html#stochasticgradientdescent)原理的读者。

下文介绍的各种更新器之间最主要的区别是对于学习速率的处理方式。

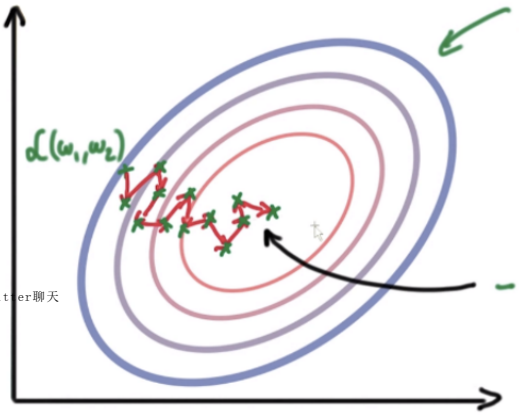
随机梯度下降

$$\theta_{t+1} = \theta_t - \alpha\delta L(\theta_t)$$

`Theta` （θ）是权重，每个theta按其对应的损失函数的梯度进行调整。

`Alpha` （α）是学习速率。如果alpha值很小，那么向最小误差收敛的过程会比较缓慢。如果alpha值很大，模型会偏离最小误差，学习将会停止。

由于定型样例之间的差异，损失函数(L)的梯度在每次迭代后变化很快。请看下图中的收敛路径。更新的步幅很小，在向最小误差逼近的过程中会来回振荡。



Github：Deeplearning4j中的SGDUpdater
(https://github.com/deeplearning4j/deeplearning4j/blob/b585d6c1ae75e48e06db86880a5acd22593d3889/deeplearning4j-core/src/main/java/org/deeplearning4j/nn/updater/SgdUpdater.java)

动量

我们用动量（*momentum*）来减少振荡。动量会根据之前更新步骤的情况来调整更新器的运动方向。我们用一个新的超参数μ（mu）来表示动量。

$$v_{t+1} = \mu v_t - \alpha\delta L(\theta_t)$$
$$\theta_{t+1} = \theta_t + v_{t+1}$$

上图为使用了动量的SGD算法。Github：Deeplearnign4j中的Nesterov动量更新器
(https://github.com/deeplearning4j/deeplearning4j/blob/b585d6c1ae75e48e06db86880a5acd22593d3889/deeplearning4j-core/src/main/java/org/deeplearning4j/nn/updater/NesterovsUpdater.java)

Adagrad

Adagrad会根据每个参数对应的历史梯度（之前更新步骤中的情况）来调整该参数的alpha。具体方法是将更新规则中的当前梯度除以历史梯度之和。其结果是，梯度很大时，alpha会减小，反之则alpha增大。

$$g_{t+1} = g_t + \delta L(\theta_t)^2$$

$$\theta_{t+1} = \theta_t - \frac{\alpha \delta L(\theta)^2}{\sqrt{g_{t+1}} + \epsilon}$$

参考：Deeplearning4j中的AdaGradUpdater
(https://deeplearning4j.org/doc/org/deeplearning4j/nn/updater/AdaGradUpdater.html)

RMSProp

RMSProp和Adagrad的唯一区别在于 `g_t` 项的计算方式是对梯度的平均值而非总和进行指数衰减。

$$g_{t+1} = \gamma g_t + (1-\gamma)\delta L(\theta)^2$$

此处的 `g_t` 称为δL的二阶矩。此外，还可以引入一阶矩 `m_t` 。

$$m_{t+1} = \gamma m_t + (1-\gamma)\delta L(\theta)$$
$$g_{t+1} = \gamma g_t + (1-\gamma)\delta L(\theta)^2$$

像第一个例子中那样加入动量……

$$v_{t+1} = \mu v_t - \frac{\alpha \delta L(\theta)}{\sqrt{g_{t+1} - m_{t+1}^2} + \epsilon}$$

……最后像第一个例子中一样得到新的 `theta` 。

$$\theta_{t+1} = \theta_t + v_{t+1}$$

Github：Deeplearning4j中的RMSPropUpdater
(https://github.com/deeplearning4j/deeplearning4j/blob/b585d6c1ae75e48e06db86880a5acd22593d3889/deeplearning4j-core/src/main/java/org/deeplearning4j/nn/updater/RmsPropUpdater.java)

AdaDelta

AdaDelta同样采用指数衰减的 `g_t` 平均值，也就是梯度的二阶矩。但它不采用通常作为学习速率的alpha，而是引入 `x_t` ，即 `v_t` 的二阶矩。

$$g_{t+1} = \gamma g_t + (1-\gamma)\nabla \mathcal{L}(\theta)^2$$
$$x_{t+1} = \gamma x_t + (1-\gamma)v_{t+1}^2$$
$$v_{t+1} = -\frac{\sqrt{x_t + \epsilon}\,\delta L(\theta_t)}{\sqrt{g_{t+1} + \epsilon}}$$
$$\theta_{t+1} = \theta_t + v_{t+1}$$

参考：Deepelearning4j中的AdaDeltaUpdater
(https://deeplearning4j.org/doc/org/deeplearning4j/nn/updater/AdaDeltaUpdater.html)

ADAM

ADAM同时使用一阶矩 `m_t` 和二阶矩 `g_t` ，但二者均会随时间衰减。步幅约为 `±α` 。当我们不断逼近最小误差时，步幅会逐渐缩小。

$$m_{t+1} = \gamma_1 m_t + (1-\gamma_1)\nabla \mathcal{L}(\theta_t)$$
$$g_{t+1} = \gamma_2 g_t + (1-\gamma_2)\nabla \mathcal{L}(\theta_t)^2$$
$$\hat{m}_{t+1} = \frac{m_{t+1}}{1-\gamma_1^{t+1}}$$
$$\hat{g}_{t+1} = \frac{g_{t+1}}{1-\gamma^{t+1}}$$

$$\theta_{t+1} = \theta_t - \frac{\alpha \hat{m}_{t+1}}{\sqrt{\hat{s}_{t+1}} + \epsilon}$$

参考：Deeplearning4j中的AdamUpdater
(http://deeplearning4j.org/doc/org/deeplearning4j/nn/updater/AdamUpdater.html)

^