# Deep Learning Technology and Application

Ge Li

Peking University

# 关于学习率的优化

梯度下降过程中的权重更新：

$$\theta = \theta - \alpha \nabla_{\theta} J(\theta)$$

# Momentum

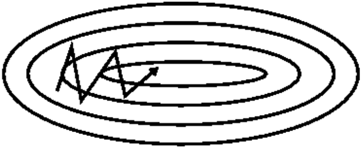
- Momentum based Gradient Descent:

$$v_t = \gamma v_{t-1} + \alpha \nabla_\theta \left( \frac{1}{m} \sum_t^m J(x^{(i)}, y^{(i)}; \theta) \right)$$

$$\theta = \theta - v_t$$

- 在更新模型参数时，对于那些当前梯度方向与上一次梯度方向相同的参数，进行加强，即在这些方向上的参数更新更快了；
- 对于那些当前梯度方向与上一次梯度方向不同的参数，进行削减，即在这些方向的参数更新上减慢了。

一般而已，动量项参数 $\gamma < 0.9$



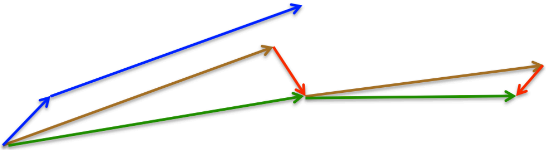
# NAG-Nesterov Accelerated Gradient

- NAG:

$$v_t = \gamma v_{t-1} + \alpha \nabla_\theta \left( \frac{1}{m} \sum_t^m J(\theta - \gamma v_{t-1}) \right)$$

$$\theta = \theta - v_t$$

- Computing $\theta - \gamma v_{t-1}$ thus gives us an approximation of the next position of the parameters (the gradient is missing for the full update). This is a rough idea where our parameters are going to be.
- We can now effectively look ahead by calculating the gradient not with regard to our current parameters $\theta$ but with regard to the approximate future position of our parameters.



# Adagrad

- 在上述模型中，每个模型参数 $\theta_i$ 使用相同的学习速率 $\alpha$，而 Adagrad 在每一个更新步骤中对于每一个模型参数 $\theta_i$ 使用不同的学习速率 $\alpha_i$；
- 设第 $t$ 次更新步骤中，目标函数的参数 $\theta_i$ 梯度为 $g_{t,i}$，即：

$$g_{t,i} = \nabla_\theta J(\theta_i)$$

- 则，传统的 SGD 的更新方程表示为：

$$\theta_{t+1,i} = \theta_{t,i} - \alpha \cdot g_{t,i}$$

- 而，Adagrad 对每一个参数使用不同的学习率，则其更新方程变为：

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\alpha}{\sqrt{G_{t,ii} + \epsilon}} \cdot g_{t,i}$$

其中，$G_t \in \mathcal{R}^{d\times d}$ 是一个对角矩阵，其中第 $i$ 行的对角元素 $e_{ii}$ 为过去到当前第 $i$ 个参数 $\theta_i$ 的梯度的平方和，$\epsilon$ 是一个平滑参数，为了使得分母不为 0（如可取 $\epsilon = 1e-8$）

# Adagrad

写成矩阵形式：

$$\Delta\theta_t = -\frac{\eta}{\sqrt{G_t + \epsilon}} \odot g_t$$

- Adagrad 主要优势在于它能够为每个参数自适应不同的学习速率，而一般的人工都是设定为 0.01。其缺点在于需要计算参数的整个梯度序列的平方和，并且学习速率趋势是不断衰减最终达到一个非常小的值，开始很大，最后很小。

# Adadelta

- Adadelta 提出的目的也是为了避免 Adagrad 对学习速率的调整过于“鲁莽”的问题；
- 同时，为了避免计算整个梯度序列的平方和，Adadelta 采用了“窗口”技术，即，仅对固定窗口内的 $w$ 个梯度序列进行计算；
- 当前的梯度平方的平均值（$E[g^2]_t$）仅依赖于前一个时刻的平均值和当前的梯度；

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

- 可以把 $\gamma$ 设为一个类似于动量的值，如 0.9 附近.

# Adadelta

- 下面给出 Adadelta 的表达式：从 $\Delta\theta_t$ 的表达式开始:

$$\Delta\theta_t = -\alpha \cdot g_{t,i}$$
$$\theta_{t+1} = \theta_t + \Delta\theta_t$$

对比 Adagrad 的公式：

$$\Delta\theta_t = -\frac{\alpha}{\sqrt{G_t + \epsilon}} \odot g_t$$

用 $E[g_t^2]$ 替换 $G_t$:

$$\Delta\theta_t = -\frac{\alpha}{\sqrt{E[g_t^2] + \epsilon}} \odot g_t$$

# Adadelta

- 可见，分母为梯度的均方根（Root Meam Square），简短表示为：$RMS[g]_t$ 得：

$$\Delta\theta_t = -\frac{\alpha}{RMS[g]_t} \cdot g_t$$

- 还注意到，梯度的更新中 $\alpha$ 并不平缓，做以下替换:

$$因为：E[\Delta\theta^2]_t = \gamma E[\theta^2]_{t-1} + (1-\gamma)\Delta\theta_t^2$$

- 于是，得到：

$$RMS[\Delta\theta]_t = \sqrt{E[\Delta\theta^2]_t + \epsilon}$$

# Adadelta

- 又因为 t 时刻的 $RMS[\Delta\theta]_t$ 并不知道，于是用 $t$ 时刻之前的参数更新的 RMS 来代替，即：用 $RMS[\Delta\theta]_{t-1}$ 代替 $\alpha$，最终得到 Adadelta 的更新规则：

$$
\begin{aligned}
\Delta\theta_t &= -\frac{RMS[\Delta\theta]_{t-1}}{RMS[g]_t} g_t \\
\theta_{t+1} &= \theta_t + \Delta\theta_t
\end{aligned}
$$

# RMSprop

RMSprop 由 Geoff Hinton 在他的 Coursera 课程中提出。
RMSprop 与 Adadelta 几乎在同一时间提出，只是可以看做 Adadelta 的简化版本，对比如下：

- Adadelta:

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

- RMSprop

$$E[g^2]_t = 0.9E[g^2]_{t-1} + 0.1 g_t^2$$
$$\theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{E[g_t^2] + \epsilon}} \cdot g_t$$

- 可见，RMSprop 方法也是用"衰减的梯度均方根误差"去除学习率。

# Adam-Adaptive Moment Estimation

- Adam（自适应的矩估计）也是一种不同参数自适应不同学习速率方法，与 Adadelta 与 RMSprop 区别在于，它计算历史梯度衰减方式不同，不使用历史平方衰减，其衰减方式类似动量：

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$$
$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$$

- $m_t$ 与 $v_t$ 分别是梯度的一阶矩和二阶矩的估计值，初始为 0 向量；

# Adam-Adaptive Moment Estimation

- 然而，它们通常被偏置化为趋向于 0 的向量，特别是当衰减因子(衰减率)$\beta_1, \beta_2$ 接近于 1 时；
- 为了改进这个问题，可以改进上式中的偏置项：利用经过偏置修正的一阶和二阶矩估计来计算 $m_t$ 与 $v_t$：

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

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

- 类似 Adadelta 与 RMSprop 方法，可以得到 Adam 方法的更新规则：

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

- Adam 提出者建议 $\beta_1 = 0.9, \beta_2 = 0.9999, \epsilon = 10^{-8}$. 实验证实，Adam 方法较其他方法有更好的应用效果。

***Thanks.***