1 什么是优化器?

（1）解释

一言以蔽之，优化器就是在深度学习反向传播过程中，指引损失函数（目标函数）的各个参数往正确的方向更新合适的大小，使得更新后的各个参数让损失函数（目标函数）值不断逼近全局最小。

（3）公式和定义

待优化参数： ω ，目标函数： $f(x)$ ，初始学习率： α ，迭代epoch： t

参数更新步骤如下：

Ⅰ.计算目标函数关于当前参数的梯度：

> $g_t = \nabla f(w_t)$

Ⅱ. 根据历史梯度计算一阶动量和二阶动量：

> $m_t = \phi(g_1, g_2, \cdots, g_t); V_t = \sum_{i=0}^{t} x_i^2$

Ⅲ. 计算当前时刻的下降梯度：

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

Ⅳ. 根据下降梯度进行更新参数：

> $w_{t+1} = w_t - \eta_t$

步骤Ⅲ、Ⅳ对于各个算法都是一致的，主要的差别就体现在步骤Ⅰ、Ⅱ上。

2.1 **随机梯度下降法（Stochastic Gradient Descent，SGD）**

随机梯度下降算法每次从训练集中随机选择一个样本来进行学习，SGD没有动量的概念，因此

> $$m_t = g_t; V_t = I^2$$

代入步骤Ⅲ，可以得到下降梯度

> $$\eta_t = \alpha \cdot g_t$$

SGD参数更新公式如下，其中 α 是学习率， g_t 是当前参数的梯度

> $$w_{t+1} = w_t - \eta_t = w_t - \alpha \cdot g_t$$

优点：

（1）每次只用一个样本更新模型参数，训练速度快

（2）随机梯度下降所带来的波动有利于优化的方向从当前的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，最终收敛于一个较好的局部极值点，甚至全局极值点。

缺点：

（1）当遇到局部最优点或鞍点时，梯度为0，无法继续更新参数

（2）沿陡峭方向震荡，而沿平缓维度进展缓慢，难以迅速收敛

2.2 **SGD with Momentum**

为了抑制SGD的震荡，SGDM认为梯度下降过程可以加入惯性。下坡的时候，如果发现是陡坡，那就利用惯性跑的快一些。SGDM全称是SGD with momentum，在SGD基础上引入了一阶动量：

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

SGD-M参数更新公式如下，其中 α 是学习率， g_t 是当前参数的梯度

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

一阶动量是各个时刻梯度方向的指数移动平均值，也就是说， t 时刻的下降方向，不仅由当前点的梯度方向决定，而且由此前累积的下降方向决定。 β_1 的经验值为0.9，这就意味着下降方向主要是此前累积的下降方向，并略微偏向当前时刻的下降方向。想象高速公路上汽车转弯，在高速向前的同时略微偏向，急转弯可是要出事的。

特点：

因为加入了动量因素，SGD-M缓解了SGD在局部最优点梯度为0，无法持续更新的问题和振荡幅度过大的问题，但是并没有完全解决，当局部沟壑比较深，动量加持用完了，依然会困在局部最优里来回振荡。

2.3 **SGD with Nesterov Acceleration**

SGD 还有一个问题是困在局部最优的沟壑里面震荡。想象一下你走到一个盆地，四周都是略高的小山，你觉得没有下坡的方向，那就只能待在这里了。可是如果你爬上高地，就会发现外面的世界还很广阔。因此，我们不能停留在当前位置去观察未来的方向，而要向前一步、多看一步、看远一些。

NAG全称Nesterov Accelerated Gradient，是在SGD、SGD-M的基础上的进一步改进，改进点在于步骤Ⅰ。我们知道在时刻 t 的主要下降方向是由累积动量决定的，自己的梯度方向说了也不算，那与其看当前梯度方向，不如先看看如果跟着累积动量走了一步，那个时候再怎么走。因此，NAG在步骤Ⅰ，不计算当前位置的梯度方向，而是计算如果按照累积动量走了一步，那个时候的下降方向：

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

NAG参数更新公式如下，其中 α 是学习率， g_t 是当前参数的梯度

> $$w_{t+1} = w_t - \alpha \cdot g_t = w_t - \alpha * (\nabla f(w_t - \alpha \cdot m_{t-1}/\sqrt{V_{t-1}}))$$

然后用下一个点的梯度方向，与历史累积动量相结合，计算步骤Ⅱ中当前时刻的累积动量。

特点：

有利于跳出当前局部最优的沟壑，寻找新的最优值，但是收敛速度慢。

2.4 **AdaGrad（自适应学习率算法）**

SGD系列的都没有用到二阶动量。二阶动量的出现，才意味着“自适应学习率”优化算法时代的到来。SGD及其变种以同样的学习率更新每个参数，但深度神经网络往往包含大量的参数，这些参数并不是总会用得到（想想大规模的embedding）。对于经常更新的参数，我们已经积累了大量关于它的知识，不希望被单个样本影响太大，希望学习速率慢一些；对于偶尔更新的参数，我们了解的信息太少，希望能从每个偶然出现的样本身上多学一些，即学习速率大一些。

怎么样去度量历史更新频率呢？

那就是二阶动量——该维度上，记录到目前为止所有梯度值的平方和：

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

我们再回顾一下步骤Ⅲ中的下降梯度：

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

AdaGrad参数更新公式如下，其中 α 是学习率， g_t 是当前参数的梯度

$$w_{t+1} = w_t - \alpha \cdot m_t / \sqrt{V_t} = w_t - \alpha \cdot m_t / \sqrt{\sum_{\tau=1}^{t} g_\tau^2}$$

可以看出，此时实质上的学习率由 α 变成了 $\alpha / \sqrt{V_t}$ 。 一般为了避免分母为0，会在分母上加一个小的平滑项。因此 $\sqrt{V_t}$ 是恒大于0的，而且参数更新越频繁，二阶动量越大，学习率就越小。

优点：

（1）在稀疏数据场景下表现非常好

（2）此前的SGD及其变体的优化器主要聚焦在优化梯度前进的方向上，而AdaGrad首次使用二阶动量来关注学习率（步长），开启了自适应学习率算法的里程。

缺点：

（1）因为 $\sqrt{V_t}$ 是单调递增的，会使得学习率单调递减至0，可能会使得训练过程提前结束，即便后续还有数据也无法学到必要的知识。

2.5 **AdaDelta / RMSProp**

由于AdaGrad单调递减的学习率变化过于激进，考虑一个改变二阶动量计算方法的策略：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。这也就是AdaDelta名称中Delta的来历。

修改的思路很简单。前面讲到，指数移动平均值大约就是过去一段时间的平均值，因此我们用这一方法来计算二阶累积动量：

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

AdaDelta / RMSProp参数更新公式如下，其中 α 是学习率， g_t 是当前参数的梯度

> $$\begin{aligned} w_{t+1} &= w_t - \alpha \cdot m_t / \sqrt{V_t} \\ &= w_t - \alpha \cdot m_t / \sqrt{\beta_2 \cdot V_{t-1} + (1-\beta_2) g_t^2} \end{aligned}$$

优点：

避免了二阶动量持续累积、导致训练过程提前结束的问题了。

2.6 Adam

谈到这里，Adam和Nadam的出现就很自然而然了——它们是前述方法的集大成者。SGD-M在SGD基础上增加了一阶动量，AdaGrad和AdaDelta在SGD基础上增加了二阶动量。把一阶动量和二阶动量都用起来，就是Adam了——Adaptive + Momentum。

SGD的一阶动量：

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

加上AdaDelta的二阶动量：

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

Adam参数更新公式如下，其中 α 是学习率， g_t 是当前参数的梯度

$$\begin{aligned} w_{t+1} &= w_t - \alpha \cdot m_t / \sqrt{V_t} \\ &= w_t - \alpha \cdot (\beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t) / \sqrt{\beta_2 \cdot V_{t-1} + (1 - \beta_2) g_t^2} \end{aligned}$$

优化算法里最常见的两个超参数 β_1, β_2 就都在这里了，前者控制一阶动量，后者控制二阶动量。

优点：

（1）通过一阶动量和二阶动量，有效控制学习率步长和梯度方向，防止梯度的振荡和在鞍点的静止。

缺点：

（1）**可能不收敛**：

二阶动量是固定时间窗口内的累积，随着时间窗口的变化，遇到的数据可能发生巨变，使得V_t可能会时大时小，不是单调变化。这就可能在训练后期引起学习率的震荡，导致模型无法收敛。

修正的方法。由于Adam中的学习率主要是由二阶动量控制的，为了保证算法的收敛，可以对二阶动量的变化进行控制，避免上下波动。

$$V_t = max(\beta_2 * V_{t-1} + (1 - \beta_2) g_t^2, V_{t-1})$$

通过这样修改，就保证了 $||V_t|| \geq ||V_{t-1}||$ ，从而使得学习率单调递减。

（2）**可能错过全局最优解：**

自适应学习率算法可能会对前期出现的特征过拟合，后期才出现的特征很难纠正前期的拟合效果。后期Adam的学习率太低，影响了有效的收敛。

2.7 Nadam

最后是Nadam。我们说Adam是集大成者，但它居然遗漏了Nesterov，这还能忍？必须给它加上，按照NAG的步骤：

> $$g_t = \nabla f(w_t - \alpha \cdot m_{t-1}/\sqrt{V_t})$$

这就是Nesterov + Adam = Nadam了。