蒋豆芽

深度学习——优化函数 已购

来自 【机器学习面试题汇总与解析（蒋豆芽面试题总结）】 | 79 浏览 | 0 回复 | 2021-04-19

蒋豆芽

+关注

机器学习面试题汇总与解析——优化函数

1. **说一下你了解的优化函数?** ☆☆☆☆☆
2. **SGD和Adam谁收敛的比较快？谁能达到全局最优解?** ☆☆☆☆☆
3. **说说常见的优化器以及优化思路，写出他们的优化公式**☆☆☆☆☆
4. **深度学习中的优化算法总结 Optimizer**☆☆☆☆☆
5. **adam用到二阶矩的原理是什么**☆☆☆☆☆
6. **Batch的大小如何选择，过大的batch和过小的batch分别有什么影响**☆☆☆☆☆
7. **梯度下降的思想**☆☆☆☆☆

==========

- 本专栏适合于**Python已经入门**的学生或人士，有一定的编程基础。
- 本专栏适合于**算法工程师、机器学习、图像处理求职**的学生或人士。
- 本专栏针对面试题答案进行了**优化，尽量做到好记、言简意赅。这才是一份面试题总结的正确打开方式。这样才方便背诵**
- 如专栏内容有错漏，欢迎在评论区指出或私聊我更改，一起学习，共同进步。
- **相信大家都有着高尚的灵魂，请尊重我的知识产权，未经允许严禁各类机构和个人转载、传阅本专栏的内容。**

==========

1. **说一下你了解的优化函数?** ☆☆☆☆☆

   参考回答

   1. **梯度下降法。**

      梯度下降法是原始的优化方法，梯度下降的**核心思想**：**负梯度**方向是使函数值下降最快的方向，因此我们的目标就是求取目标函数的**负梯度。**

      在梯度下降法中，因为每次都遍历了完整的训练集，**其能保证结果为全局最优**（优点），但是也因为我们需要对于每个参数求偏导，且在对每个参数求偏导的过程中还需要对训练集遍历一次，当训练集很大时，**计算费时（缺点）**。

针对一个批次的数据。

3. **随机梯度下降法。**

   再极端一点，就是随机梯度下降法，即每次从训练集中**随机抽取一个数据**来计算梯度。因此，其**速度较快（优点）**，但是其**每次的优化方向不一定是全局最优的（缺点）。**因为误差，所以每一次迭代的梯度受抽样的影响比较大,也就是说梯度含有比较大的噪声,不能很好的反映真实梯度，并且**SGD有较高的方差，其波动较大。而且SGD无法逃离鞍点。**

4. **Momentum随机梯度下降法**

   Momentum借用了物理中的**动量**概念,即前一次的梯度也会参与运算。为了表示动量，引入了**一阶动量**m(momentum)。m是之前的梯度的累加,但是每回合都有一定的衰减。

   总而言之，momentum能够加速SGD收敛，抑制震荡。并且动量有机会逃脱局部极小值(鞍点)。

5. **Nesterov动量型随机梯度下降法**

   Nesterov动量型随机梯度下降法是在momentum更新梯度时加入对当前梯度的校正，让梯度“多走一步”，可能跳出局部最优解。

   相比momentum动量法，Nesterov动量型随机梯度下降法对于凸函数在收敛性证明上有更强的理论保证。

6. **Adagrad法**

   SGD的学习率是线性更新的，每次更新的差值一样。后面的优化法开始围绕**自适应学习率**进行改进。Adagrad法引入**二阶动量**，根据训练轮数的不同，对学习率进行了动态调整。

   但缺点是Adagrad法仍然需要人为指定一个合适的全局学习率，同时网络训练到一定轮次后，分母上梯度累加过大使得学习率为0而导致训练提前结束。

7. **Adadelta法**

   Adadelta法是对Adagrad法的扩展，通过引入衰减因子ρ消除Adagrad法对全局学习率的依赖

   Adadelta法避免了对全局学习率的依赖。而且只用了部分梯度加和而不是所有，这样避免了梯度累加过大使得学习率为0而导致训练提前结束。

8. **RMSProp法**

   AdaGrad算法在迭代后期由于学习率过小，可能较难找到一个有用的解。为了解决这一问题，RMSprop算法对Adagrad算法做了一点小小的修改，RMSprop使用指数衰减只保留过去给定窗口大小的梯度，使其能够在找到凸碗状结构后快速收敛。RMSProp法可以视为Adadelta法的一个特例

   RMSProp法缺陷在于依然使用了全局学习率，需要根据实际情况来设定。可以看出分母不再是一味的增加，它会重点考虑距离它较近的梯度（指数衰减的效果）。优点是只用了部分梯度加和而不是所有，这样避免了梯度累加过大使得学习率为0而导致训练提前结束。

蒋豆芽

每个参数的学习率。Adam法主要的优点在于经过偏置校正后，每一次迭代学习率都有一个**确定范围**，这样可以使得参数更新比较平稳。

答案解析

优化函数的作用

前一篇讲了损失函数，损失函数是用来度量模型**预测值**和**真实值**间的**偏差**，偏差越小，模型越好，所以我们需要**最小化损失函数。**如何求最小值呢？高中知识告诉我们，对于凸函数可以直接求导可以找到最值，但是神经网络的损失函数往往是**非凸函数**，我们并不能直接求导。所以**优化函数**就可以帮助我们解决这个问题，**优化函数就是用来最小化我们的损失函数的。**

好，我们一个个分析：

1. **梯度下降法（gradient descent）**，

   梯度下降（gradient descent）在机器学习中应用十分的广泛，不论是在线性回归还是Logistic回归中，它的主要目的是通过迭代找到目标函数的最小值，或者收敛到最小值。

   假设待优化的目标函数 $J(w)$是具有一阶连续偏导的函数，现在的目标是要求取最小的$J(w)$，即 $minJ(w)$

   梯度下降的核心思想：负梯度方向是使函数值下降最快的方向，在迭代的每一步根据负梯度的方向更新 w的值，从而求得最小的 $J(w)$。因此我们的目标就转变为求取 $J(w)$的梯度。

   当$J(w)$是凸函数的时候，用梯度下降的方法取得的最小值是全局最优解。但神经网络的损失函数往往是**非凸函数**，所以梯度下降法往往找到的是**局部最优解。**

   假设$h(x)$ 是我们的模型函数， n表示参数个数， m表示训练集的样本个数，$J(w)$是我们的损失函数，即待优化的目标函数。

   $$h(x)=\sum_{j=0}^{n} w_j x_j \tag{.}$$

   $$J(w)=\frac{1}{2m}\sum_{i=1}^{m}(y^i-h_w(x^i))^2$$

   对w求偏导，可以得到每个w对应的梯度：

   $$\nabla_{w_j}=\frac{\partial J(w)}{\partial w_j}=-\frac{1}{m}\sum_{i=1}^{m}(y^i-h_w(x^i))x_j^i \tag{.}$$

   接着按每个w的**负梯度**来更新每个w，

   $$w_j^*=w_j+\eta\cdot\nabla_{w_j} \tag{.}$$

   η是学习率。可以看出，在梯度下降法中，因为每次都遍历了完整的训练集，**其能保证结果为全局最优**（优点），但是也因为我们需要对于每个参数求偏导，且在对每个参数求偏导的过程中还需要对训练集遍历一次，当训练集（m）很大时，计算费时（缺点）。

数据，即m表示一个批次的数据个数。

3. **随机梯度下降（stochastic gradient descent)**

   再极端一点，我们一个批次都不要，而是让优化器自己**随机选择一个样本**，其每次对w的更新，都是针对单个样本数据，并没有遍历完整的参数。当样本数据很大时，可能到迭代完成，也只不过遍历了样本中的一小部分。因此，其速度较快（优点），但是其每次的优化方向不一定是全局最优的（缺点）。因为误差，所以每一次迭代的梯度受抽样的影响比较大,也就是说梯度含有比较大的噪声,不能很好的反映真实梯度，并且**SGD有较高的方差，其波动较大。**

   因为每一次迭代的梯度受抽样的影响比较大，学习率需要逐渐减少，否则模型很难收敛。在实际操作中，一般采用线性衰减：

$$\eta_k = (1-\alpha)\eta_0 + \alpha\eta_\tau \tag{.}$$
$$\alpha = \frac{k}{\tau}$$

   η_0是初始学习率，η_τ是最后一次迭代的学习率，τ代表自然迭代次数，η_τ设为η_0的1%。k我们一般设为100的倍数。

   为了克服上述缺点，我们可以将批次梯度下降法和随机梯度下降法结合起来——**随机批次梯度下降法**，即随机抽取一个批次的数据用于计算梯度，这样就可以减少噪声，使得训练收敛平稳。

   我们来总结一下梯度下降的优化方法，SGD最为常用。

   优点： 收敛速度快。

   缺点：1. 训练不稳定。2. 选择适当的学习率可能很困难。 太小的学习率会导致收敛性缓慢，而学习速度太大可能会妨碍收敛，并导致损失函数在最小点波动。3. 无法逃脱鞍点。

   所以后面发展了不少其他的优化方法改进了SGD。

4. **Momentum随机梯度下降法**

   核心思想：Momentum借用了物理中的**动量**概念,即前一次的梯度也会参与运算。为了表示动量，引入了**一阶动量**m(momentum)。m是之前的梯度的累加,但是每回合都有一定的衰减。公式如下：

$$m_t = \beta m_{t-1} + (1-\beta)\cdot g_t \tag{.}$$
$$w_{t+1} = w_t - \eta \cdot m_t$$

   g_t表示第t次epoch计算的梯度， β表示动量因子，一般取0.9，所以当前权值的改变受上一次改变的影响，类似加上了**惯性**。

   - 在梯度方向改变时，momentum能够降低参数更新速度，从而减少震荡；
   - 在梯度方向相同时，momentum可以加速参数更新， 从而加速收敛。

   总而言之，momentum能够加速SGD收敛，抑制震荡。并且动量有机会逃脱局部极小值(鞍点)。

蒋豆芽

“多走一步”，可能跳出局部最优解：

$$\begin{aligned} w_t^* &= \beta m_{t-1} + w_t \\ m_t &= \beta m_{t-1} + (1-\beta)\cdot g_t \\ w_{t+1} &= w_t - \eta \cdot m_t \end{aligned} \tag{.}$$

相比momentum动量法，Nesterov动量型随机梯度下降法对于凸函数在收敛性证明上有更强的理论保证。

6. **Adagrad法**

我们上面讲到了SGD的学习率是线性更新的，每次更新的差值一样。后面的优化法开始围绕自适应学习率进行改进。Adagrad法引入**二阶动量**，根据训练轮数的不同，对学习率进行了动态调整：

$$\begin{aligned} v_t &= \sqrt{\sum_{\tau=1}^{t} g_\tau^2 + \epsilon} \\ \eta_t &= \frac{\eta_{global}}{v_t} \\ w_{t+1} &= w_t - \eta_t \cdot g_t \end{aligned} \tag{.}$$

公式中， g_t表示第t次epoch计算的梯度， ϵ为一个小常数（通常设定为10-6数量级）以防止分母为零。在网络训练的前期，由于分母中梯度的累加（v_t）较小，所以一开始的学习率η_t比较大；随着训练后期梯度累加较大时，η_t逐渐减小，而且是自适应地减小。

但缺点是Adagrad法仍然需要人为指定一个合适的全局学习率，同时网络训练到一定轮次后，分母上梯度累加过大使得学习率为0而导致训练提前结束。

7. **Adadelta法**

Adadelta法是对Adagrad法的扩展，通过引入衰减因子ρ消除Adagrad法对全局学习率的依赖，具体可表示为：

$$\begin{aligned} v_t &= \rho \cdot v_{t-1} + (1-\rho)\cdot g_t^2 \\ \eta_t &= \frac{\sqrt{s_{t-1}+\epsilon}}{\sqrt{v_t+\epsilon}} \\ s_t &= \rho \cdot s_{t-1} + (1-\rho)\cdot (\eta_t \cdot g_t)^2 \end{aligned} \tag{.}$$

其中，ρ为区间[0,1]间的实值：较大的ρ会促进网络更新；较小的ρ值会抑制更新。两个超参数的推荐设定为$\rho = 0.95$，$\epsilon = 10^{-6}$。可以看出，Adadelta法确实避免了对全局学习率的依赖。而且只用了部分梯度加和而不是所有，这样避免了梯度累加过大使得学习率为0而导致训练提前结束。

8. **RMSProp法**

AdaGrad算法在迭代后期由于学习率过小，可能较难找到一个有用的解。为了解决这一问题，RMSprop算法对Adagrad算法做了一点小小的修改，RMSprop使用指数衰减只保留过去给定窗口大小的梯度，使其能够在找到凸碗状结构后快速收敛。RMSProp法可以视为Adadelta法的一个特例，即依然使用全局学习率替换掉Adadelta法中的s_t：

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

实际使用中推荐参数为$\eta_{global} = 1$，$\rho = 0.9$，$\epsilon = 10^{-6}$。RMSProp法缺陷在于依然使用了全局学习率，需要根据实际情况来设定。可以看出分母不再是一味的增加，它会重点考虑距离它较近的梯度（指数衰减的效果）。优点是只用了部分梯度加和而不是所有，这样避免了梯度累加过大使得学习率为0而导致训练提前结束。

9. **Adam法**

Adam法本质上是带有动量项的RMSProp法，它利用梯度的一阶矩估计和二阶矩估计动态调整每个参数的学习率。Adam法主要的优点在于经过偏置校正后，每一次迭代学习率都有一个确定范围，这样可以使得参数更新比较平稳。

$$\begin{aligned} m_t &= \beta_1 \cdot m_{t-1} + (1-\beta_1) \cdot g_t \\ v_t &= \beta_2 \cdot v_{t-1} + (1-\beta_2) \cdot g_t^2 \\ m_t^* &= \frac{m_t}{1-\beta_1^t} \\ v_t^* &= \frac{v_t}{1-\beta_2^t} \\ w_{t+1} &= w_t - \eta \frac{m_t^*}{\sqrt{v_t^*} + \epsilon} \end{aligned} \tag{.}$$

可以看出，Adam法依然需要指定学习率η，对于其中的超参数，一般设定为：$\beta_1 = 0.9$，$\beta_2 = 0.999$，$\epsilon = 10^{-8}$，$\eta = 0.001$。可以看出，每一次迭代学习率都有一个确定范围，即$-\frac{m_t^*}{\sqrt{v_t^*}+\epsilon}$

优化器的选择

那种优化器最好？该选择哪种优化算法？目前还没能够达达成共识。Schaul et al (2014)展示了许多优化算法在大量学习任务上极具价值的比较。虽然结果表明，具有自适应学习率的优化器表现的很鲁棒，不分伯仲，但是没有哪种算法能够脱颖而出。

目前，最流行并且使用很高的优化器（算法）包括SGD、具有动量的SGD、RMSprop、具有动量的RMSProp、AdaDelta和Adam。在实际应用中，选择哪种优化器应结合具体问题；同时，优化器的选择也取决于使用者对优化器的熟悉程度（比如参数的调节等等）。

我们来总结一下：

从优化器发展历程可以看出，有两个参数很重要。一个是**优化方向**，决定“前进的方向是否正确”，在优化器中反映为**梯度或动量**。另一个是**步长**，决定“每一步迈多远”，在优化器中反映为**学习率**。

待优化参数：w，目标函数：$J(w)$，初始学习率：η，迭代epoch：t

参数更新步骤如下：

1. **计算目标函数关于当前参数的梯度**：

蒋豆芽

$$m_t = \phi(g_1, g_2, \ldots, g_t) \tag{.}$$

$$v_t = \sum_{i=0}^{t} g_i^2$$

3. **计算当前时刻的下降梯度**：

$$\nabla = \eta \cdot \frac{m_t}{\sqrt{v_t}} \tag{.}$$

4. **根据下降梯度进行参数更新**：

$$w_{t+1} = w_t - \nabla \tag{.}$$

优化器的步骤基本都是以上流程，差异体现在第一步和第二步上。SGD没有利用动量，MSGD在SGD基础上增加了一阶动量，AdaGrad、AdaDelta和RMSProp法在SGD基础上增加了二阶动量。**Adam把一阶动量和二阶动量都用起来了。**

类似的问题还有：

2. **SGD和Adam谁收敛的比较快？谁能达到全局最优解？** ☆☆☆☆☆

   参考回答

   SGD算法没有动量的概念，SGD和Adam相比，缺点是下降速度慢，对学习率要求严格。

   而Adam引入了一阶动量和二阶动量，下降速度比SGD快，Adam可以自适应学习率，所以初始学习率可以很大。

   SGD相比Adam，更容易达到全局最优解。主要是后期Adam的学习率太低，影响了有效的收敛。

   我们可以前期使用Adam，后期使用SGD进一步调优。

   答案解析

   无。

3. **说说常见的优化器以及优化思路，写出他们的优化公式** ☆☆☆☆☆

   参考回答

   回答参考上面。

   答案解析

   无。

4. **深度学习中的优化算法总结 Optimizer** ☆☆☆☆☆

答案解析

无。

5. **adam用到二阶矩的原理是什么**☆☆☆☆☆

参考回答

回答参考上面。

答案解析

无。

6. **Batch的大小如何选择，过大的batch和过小的batch分别有什么影响**☆☆☆☆☆

参考回答

Batch选择时尽量采用2的幂次，如8、16、32等

在合理范围内，增大Batch_size的**好处**：

1. 提高了**内存利用率**以及大矩阵乘法的并行化效率。
2. 减少了跑完一次epoch(全数据集）所需要的迭代次数，加快了对于相同数据量的处理速度。

盲目增大Batch_size的**坏处**：

1. 提高了内存利用率，但是内存容量可能不足。
2. 跑完一次epoch(全数据集)所需的迭代次数减少，要想达到相同的精度，其所花费的时间大大增加，从而对参数的修正也就显得更加缓慢。
3. Batch_size增大到一定程度，其确定的下降方向已经基本不再变化。

Batch_size过小的**影响**：

1. 训练时不稳定，可能不收敛
2. 精度可能更高。

答案解析

无。

7. **梯度下降的思想**☆☆☆☆☆

蒋豆芽

答案解析

无。

资源分享 python 机器学习 算法工程师 春秋招 面试题 面经

举报

收藏 赞

相关专栏

机器学习面试题汇总与解析（蒋豆芽面试题总结）

27篇文章 | 90订阅

已订阅

0条评论 默认排序

没有回复

请留下你的观点吧~

发布

牛客博客，记录你的成长

关于博客 | 意见反馈 | 免责声明 | 牛客网首页