运筹学基础答辩 -- 深度学习中的优化器

小组成员：曾伟豪 鲍方龙 杨宗元

目录

一 符号标记与问题介绍

1.1 符号标记

以下为本次学习中会用到的符号：

η :学习率

θ^t :在时刻t时模型的参数

$L(\theta^t)$:参数为θ_t时的损失函数

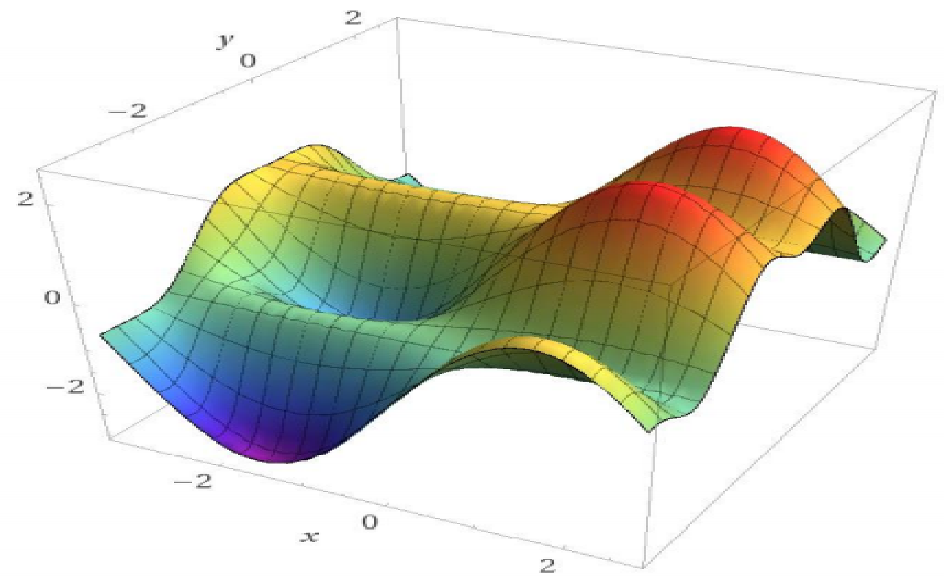
$\nabla L(\theta^t)$ or g^t :在模型参数为θ_t时，损失函数的梯度

一 符号标记与问题介绍

1.2 问题介绍

在机器学习或者深度学习中，我们总是希望找到使得损失函数$\Sigma_x L(\theta; x)$最小或者说使得$L(\theta)$最小的参数θ:

$$\theta^* = arg\ min_\theta L(\theta)$$



二 常见的优化方法

2.1 Greadient Descent

梯度下降的步骤：

1. 随机选取初始化参数值θ^0
2. 计算梯度$\nabla L(\theta^0)$
3. 更新参数$\theta^1 \leftarrow \theta^0 - \eta \nabla L(\theta^0)$
4. $\cdots\cdots$
5. 计算梯度$\nabla L(\theta^t)$
6. 更新参数$\theta^{t+1} \leftarrow \theta^t - \eta \nabla L(\theta^t)$

二 常见的优化方法

2.1 Greadient Descent

Gradient Descent存在的问题：

1. 学习率η如何确定，一直保持不变是否合适?
2. 遇到局部优化(local optimal)的情况怎么办?
3. 迭代的速度如何，有没有提高迭代速度的方法?

二 常见的优化方法

2.2 Stochastic Gradient Descent(SGD)

我们首先考虑解决上述提到的迭代速度的问题

SGD的步骤：

1. 随机选取初始化参数值θ^0
2. 计算梯度$\nabla L(\theta^0)$ （计算单个样本的损失函数的梯度）
3. 更新参数$\theta^1 \leftarrow \theta^0 - \eta\nabla L(\theta^0)$
4. $\cdots\cdots$
5. 计算梯度$\nabla L(\theta^t)$（同上）
6. 更新参数$\theta^{t+1} \leftarrow \theta^t - \eta\nabla L(\theta^t)$

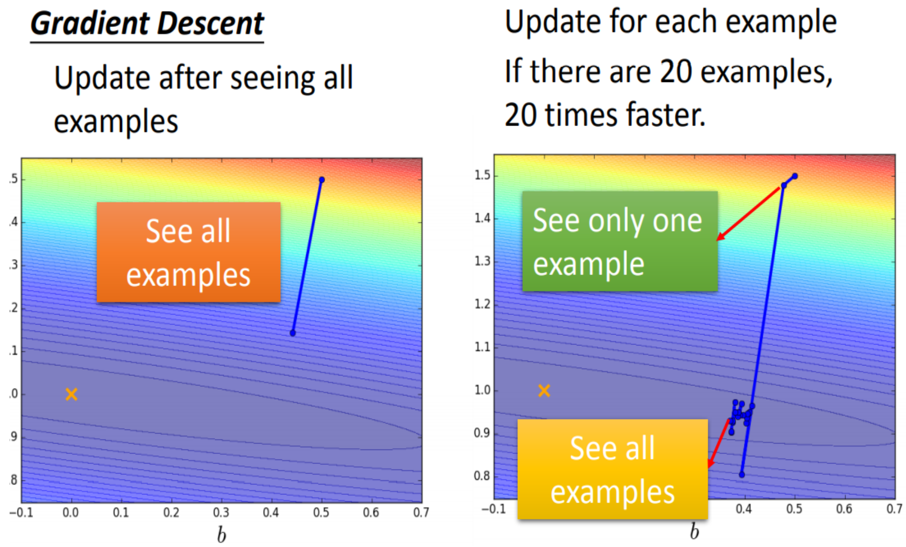
二 常见的优化方法

2.2 Stochastic Gradient Descent(SGD)

Gradient Descent与SGD的比较：

Gradient Descent: 更新一次参数需要计算所有的数据。

SGD: 每计算一次数据就迭代一次参数，如果有二十个样本数据，相当于速度提高到原来的20倍。



二 常见的优化方法

2.3 SGD with Momentum(SGDM)

引入了Movement: 前几步迭代过程中的movement减去当前状态的梯度。

SGDM步骤

1. 随机选取初始化参数值θ^0，令$Movement\ v^0 = 0$
2. 计算梯度$\nabla L(\theta^0)$ （在迭代过程中只计算一个样本的损失函数）
3. $Movement\ v^1 = \lambda v^0 - \eta \nabla L(\theta^0)$
4. 更新参数$\theta^1 = \theta^0 + v^1$
5. $\cdots\cdots$
6. 计算梯度$\nabla L(\theta^t)$（在迭代过程中只计算一个样本的损失函数）
7. $Movement\ v^{t+1} = \lambda v^t - \eta \nabla L(\theta^t)$
8. 更新参数$\theta^{t+1} = \theta^t + v^{t+1}$

二 常见的优化方法

2.3 SGD with Momentum(SGDM)

为什么使用Momentum?

v^i不仅仅基于当前的梯度，还基于曾经的 $movement$, 实际上是以前梯度 $\nabla L(\theta^0), \nabla L(\theta^1), \cdots, \nabla L(\theta^{i-1})$的权重和:

$$v^i = -\lambda^{i-1}\eta\nabla L(\theta^0) - \lambda^{i-2}\eta\nabla L(\theta^1)\cdots - \eta\nabla L(\theta^{i-1})$$

momentum可以类比于运动过程中的惯性，在如右图所示的图形中，在到达了局部最优点后，由于惯性，搜索继续进行下去，直到全局最优。

二 常见的优化方法

在之前提到的两种方法里，实际上我们已经解决了之前提到的三个问题中的两个问题：迭代过程中的更新速率问题，局部优化的问题。但是**学习率**的问题还没解决:
当学习率η过大的时候，从上图我们可以发现迭代过程没有办法收敛，η过小，迭代速率太慢。
解决上述问题的想法：

1. 我们能不能每一个epoch都适当地减少learning rate, 在开始的时候离目标较远，可以把η设置的小一点，之后不断地随迭代的进行降低η, 比如说 $\eta^t = \eta/\sqrt{t+1}$
2. 给不同类型的系数不同的η

二 常见的优化方法

2.4 Adagrad

给每一步的η^t除以σ^t,其中σ^t是之前梯度均方值。

Adagrad的步骤：

1. 随机选取初始化参数值θ^0
2. 计算梯度$\nabla L(\theta^0)$
3. 更新参数$\theta^1 = \theta^0 - \frac{\eta^0}{\sigma^0}\nabla L(\theta^0), \eta^0 = \frac{\eta}{\sqrt{1}}, \sigma^0 = \sqrt{(\nabla L(\theta^0))^2}$
4. $\cdots\cdots$
5. 计算梯度$\nabla L(\theta^t)$
6. 更新参数$\theta^{t+1} = \theta^t - \frac{\eta^t}{\sigma^t}\nabla L(\theta^t), \eta^t = \frac{\eta}{\sqrt{t+1}}, \sigma^{t+1} = \sqrt{\frac{1}{t+1}\Sigma_{i=0}^t(\nabla L(\theta^i))^2}$

推导易得：

$$\theta^{t+1} = \theta^t - \frac{\eta}{\sqrt{\Sigma_{i=0}^{t}(\nabla L(\theta^i))^2)}}$$

如图所示：

由于Adagrad的使用，在较为崎岖的路线(gradient 比较大)上采取较大的learning rate, 在较为平坦的路线上采用较大的learning rate.

提出问题:

如果一开始的gradient过大，是否会导致Adagrad没走几步就停止?

二 常见的优化方法

2.5 RMSProp

参考了之前momentum的设置方法，让之前的梯度降低权重。使得梯度指数平均和不再线性递增。

RMSProp**的步骤：**

1. 随机选取初始化参数值θ^0
2. 计算梯度$\nabla L(\theta^0)$
3. 更新参数$\theta_1 = \theta_0 - \frac{\eta}{\sqrt{v_1}} \nabla L(\theta_0), v_1 = (\nabla L(\theta_0))^2$
4. $\cdots\cdots$
5. 计算梯度$\nabla L(\theta^t)$
6. 更新参数$\theta_t = \theta_{t-1} - \frac{\eta}{\sqrt{v_t}} \nabla L(\theta_{t-1}), v_t = \alpha v_{t-1} + (1-\alpha)(g_{t-1})^2$

二 常见的优化方法

2.7 Adam

RMSProp解决了学习率的问题，但是并没有解决局部优化的问题；但我们在前面用SGDM暂时解决了局部优化的问题，那我们为什么不尝试将SGDM与RMSProp结合起来呢?

RMSProp

$$\theta_t = \theta_{t-1} - \frac{\eta}{\sqrt{v_t}} \nabla L(\theta_{t-1})$$
$$v_1 = \nabla L(\theta_0)^2$$
$$v_t = \beta_2 v_{t-1} + (1-\beta_2)(g_{t-1})^2$$

$+$

SGDM

$$\theta_t = \theta_{t-1} - \eta m_t$$
$$m_t = \beta_1 m_{t-1} + (1-\beta_1)\nabla L(\theta_{t-1})$$

得到Adam的迭代表达式:

$$\theta_t = \theta_{t-1} - \frac{\eta}{\sqrt{\hat{v}_t}+\epsilon}\hat{m}_t$$
$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}, \hat{v}_t = \frac{v_t}{1-\beta_2^t}$$
$$\beta_1 = 0.9, \beta_2 = 0.999$$
$$\epsilon = 10^{-8}$$

$1-\beta_1^t, 1-\beta_t^2$**的作用:**

防止训练开始时，m_t与v_t过大，导致随着时间进行，$\hat{m}_t$与$\hat{v}_t$变化较大。

二 常见的优化方法

2.7 Adam vs SGDM

比较Adam与SGDM在训练集上的结果：

通过右图我们发现，Adam的训练速度似乎更快

实验链接

二 常见的优化方法

2.7 Adam vs SGDM

比较Adam与SGDM在验证集上的结果：

右图表明在该实验中SGDM的泛化能力更强

实验链接

二 常见的优化方法

2.7 Adam vs SGDM

比较Adam与SGDM的收敛性：

在本实验中SGDM的收敛性更好

论文地址

总结：

Adam : 训练速度较快，泛化能力不如SGDM, 不稳定

SGDM: 稳定，容易收敛（并不绝对）

三 对现有方法的改进

Adam从14年被提出到现在，与SGDM已经成为最常用的优化方法了，还能不能改进?

三 对现有方法的改进

3.1 将SGDM与Adam的优势结合?

SWATS:

15

论文网址：https://openreview.net/forum?id=rk6qdGgCZ&noteId=rk6qdGgCZ

三 对现有方法的改进

3.2 Adam的缺陷?

某些情况下:

step	$\cdots$	100000	100001	100002	100003	$\cdots$	100999	101000	$\cdots$
gradient		1	1	1	1		100000	1	
movement		η	η	η	η		$10\sqrt{10}\eta$	$10^{-3.5}\eta$	

考虑上述情况：如果之前的gradient都很小，此时一直找不到好的下降方向，而100999步时，遇到了突然好的下降方向。但此时由于之前无效梯度的累积，现在的movement只有$10\sqrt{10}\eta$,而之前的无效方向加起来却有100000，怎么办?

三 对现有方法的改进

3.3 运筹帷幄

每一种算法的提出后，在使用上都是有利有弊。只有真正能解决实际问题的算法才是好算法，这就是我们的运筹学要解决的问题。

OR

四 参考文献

- [Hung-yi Lee, et al., Lecture slides] "ML 2020"， Lecture slides, 2020
- [Ruder, arXiv'17] Sebastian Ruder, "An Overview of Gradient Descent Optimization Algorithms", arXiv, 2017
- [Hinton, et al., Lecture slides, 2013] Geoffrey Hinton, Nitish Srivastava and Kevin Swersky, "RMSProp", Lecture slides, 2013
- [Rumelhart, et al., Nature'86] David E. Rumelhart, Geoffrey E. Hinton and Ronald J. Williams, "Learning Representations by BackPropagating Errors", Nature, 1986
- [Kingma, et al., ICLR'15] Diederik P. Kingma and Jimmy Ba, "A Method for Stochastic Optimization", ICLR, 2015