加入语雀，获得更好的阅读体验
注册 或 登录 后可以收藏本文随时阅读，还可以关注作者获得最新文章推送
立即加入

24. 神经网络优化算法们

批量梯度下降 ==》 随机梯度下降 ==》 mini batch 梯度下降 ==》 动量（momentum） ==》 牛顿（Nesterov） ==》 Adagrad ==》 RMSprop ==》 AdsDelta ==》 Adam

张晨 ML&DL知识点总结

1．什么是优化算法?

优化算法的功能，是通过改善训练方式，来最小化(或最大化)损失函数E(x)。模型内部有些参数，是用来计算测试集中目标值Y的真实值和预测值的偏差程度的，基于这些参数，就形成了损失函数E(x)。在有效地训练模型并产生准确结果时，模型的内部参数起到了非常重要的作用。 这也是为什么我们应该用各种优化策略和算法，来更新和计算影响模型训练和模型输出的网络参数，使其逼近或达到最优值。

2．优化算法类别

- 一阶优化算法：这种算法使用各参数的梯度值来最小化或最大化损失函数E(x)。最常用的一阶优化算法是梯度下降。函数梯度：导数dy/dx的多变量表达式，用来表示y相对于x的瞬时变化率。往往为了计算多变量函数的导数时，会用梯度取代导数，并使用偏导数来计算梯度。梯度和导数之间的一个主要区别是函数的梯度形成了一个向量场。因此，对单变量函数，使用导数来分析；而梯度是基于多变量函数而产生的。
- 二阶优化算法：二阶优化算法使用了二阶导数(也叫做Hessian方法)来最小化或最大化损失函数。由于二阶导数的计算成本很高，所以这种方法并没有广泛使用。

3．批量梯度下降

梯度下降的功能是：通过寻找最小值，控制方差，更新模型参数，最终使模型收敛。

网络更新参数的公式为：$\theta=\theta-\eta\times\nabla(\theta).J(\theta)$，其中η是学习率，∇(θ).J(θ)是损失函数J(θ)的梯度。

4．批量梯度下降的缺点

- 传统的批量梯度下降将计算整个数据集梯度，但只会进行一次更新，因此在处理大型数据集时速度很慢且难以控制，甚至导致内存溢出。
- 权重更新的快慢是由学习率η决定的，并且可以在凸曲面中收敛到全局最优值，在非凸曲面中可能趋于局部最优值。
- 使用标准形式的批量梯度下降还有一个问题，就是在训练大型数据集时存在冗余的权重更新。

5．随机梯度下降

随机梯度下降（Stochastic gradient descent，SGD）对每个训练样本进行参数更新，每次执行都进行一次更新，且执行速度更快。

$\theta=\theta-\eta\cdot\nabla(\theta)\times J(\theta;x(i);y(i))$，其中x(i)和y(i)为训练样本。

6．随机梯度下降的优缺点

优点：

- 对于训练速度来说，随机梯度下降法由于每次仅仅采用一个样本来迭代，训练速度很快
- 频繁的更新使得参数间具有高方差，损失函数会以不同的强度波动。这实际上是一件好事，因为它有助于我们发现新的和可能更优的局部最小值，而标准梯度下降将只会收敛到某个局部最优值。

缺点：

- 对于精准度来说，随机梯度下降法每次训练仅仅用一个样本决定梯度的方向，可能得到局部最小值，精准度不高。
- 对于收敛速度来说，由于随机梯度下降法一次迭代一个样本，导致迭代方向变化很大，不能很快的收敛到局部最优解。

7．mini batch 梯度下降

为了避免SGD和标准梯度下降中存在的问题，一个改进方法为小批量梯度下降（Mini Batch Gradient Descent），因为对每个批次中的n个训练样本，这种方法只执行一次更新。

张昌 ML&DL知识点总结

8．mini batch 梯度下降的优缺点

优点：

- 可以解决高方差的参数更新和不稳定收敛的问题

缺点：

- 很难选择出合适的学习率。太小的学习率会导致网络收敛过于缓慢，而学习率太大可能会影响收敛，并导致损失函数在最小值上波动，甚至出现梯度发散。
- 相同的学习率并不适用于所有的参数更新。如果训练集数据很稀疏，且特征频率非常不同，则不应该将其全部更新到相同的程度，但是对于很少出现的特征，应使用更大的更新率。
- 在神经网络中，最小化非凸误差函数的另一个关键挑战是避免陷于多个其他局部最小值中。

9．动量（momentum）

SGD方法中的高方差振荡使得网络很难稳定收敛，所以有研究者提出了一种称为动量（Momentum）的技术，**通过优化相关方向的训练和弱化无关方向的振荡，来加速SGD训练。**换句话说，这种新方法**将上个步骤中更新向量的分量’γ’添加到当前更新向量：** $V(t)=\gamma V(t-1)+\eta\nabla(\theta).J(\theta)$。最后通过$\theta=\theta-V(t)$来更新参数。

当其梯度指向实际移动方向时，动量项γ增大；当梯度与实际移动方向相反时，γ减小。这种方式意味着动量项只对相关样本进行参数更新，减少了不必要的参数更新，从而得到更快且稳定的收敛，也减少了振荡过程。

10．牛顿（Nesterov）

如果一个滚下山坡的球，盲目沿着斜坡下滑，这是非常不合适的。**一个更聪明的球应该要注意到它将要去哪，因此在上坡再次向上倾斜时小球应该进行减速。实际上，当小球达到曲线上的最低点时，动量相当高。由于高动量可能会导致其完全地错过最小值，因此小球不知道何时进行减速，故继续向上移动。**

在该方法中，他提出**先根据之前的动量进行大步跳跃，然后计算梯度进行校正，从而实现参数更新。**这种预更新方法能防止大幅振荡，不会错过最小值，并对参数更新更加敏感。Nesterov梯度加速法（NAG）是一种**赋予了动量项预知能力的方法，通过使用动量项$\gamma V(t-1)$来更改参数θ。通过计算**

张昊 ML &DL知识点总结

θ−γV(t−1)，得到下一位置的参数近似值，这里的参数是一个粗略的概念。因此，我们不是通过计算当前参数θ的梯度值，而是通过相关参数的大致未来位置，来有效地预知未来：$V(t)=\gamma V(t-1)+\eta\nabla(\theta)J(\theta-\gamma V(t-1))$，然后使用$\theta=\theta-V(t)$来更新参数。

11．Adagrad

Adagrad方法是通过参数来调整合适的学习率η，对稀疏参数进行大幅更新和对频繁参数进行小幅更新。因此，Adagrad方法非常适合处理稀疏数据。在时间步长中，**Adagrad方法基于每个参数计算的过往梯度，为不同参数θ设置不同的学习率。**

先前，每个参数θ(i)使用相同的学习率，每次会对所有参数θ进行更新。在每个时间步t中，Adagrad方法为每个参数θ选取不同的学习率，更新对应参数，然后进行向量化。为了简单起见，我们把在t时刻参数θ(i)的损失函数梯度设为g(t,i)：

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

参数更新公式

Adagrad方法是在每个时间步中，根据过往已计算的参数梯度，来为每个参数θ(i)修改对应的学习率η。**Adagrad方法的主要好处是，不需要手工来调整学习率。Adagrad方法的主要缺点是，学习率η总是在降低和衰减。因为每个附加项都是正的，在分母中累积了多个平方梯度值，故累积的总和在训练期间保持增长。这反过来又导致学习率下降，变为很小数量级的数字，该模型完全停止学习，停止获新的额外知识。因为随着学习速度的越来越小，模型的学习能力迅速降低，而且收敛速度非常慢，需要很长的训练和学习，即学习速度降低。**

12．RMSprop

为了使学习率下降的慢点，提出了RMSprop。AdaGrad 是学习率除以梯度的平方和开根号，**RMs则变为了求梯度平方和的平均数开根号。**

13．AdsDelta

这是一个AdaGrad的延伸方法，它倾向于解决其学习率衰减的问题。Adadelta不是累积所有之前的平方梯度，而是**将累积之前梯度的窗口限制到某**

张晨-ML&DL知识点总结

个固定大小w。

14．Adam

Adam算法即自适应时刻估计方法（Adaptive Moment Estimation），能计算每个参数的自适应学习率。这个方法不仅存储了AdaDelta先前平方梯度的指数衰减平均值，而且保持了先前梯度M(t)的指数衰减平均值，这一点与动量类似：

M(t)为梯度的第一时刻平均值，V(t)为梯度的第二时刻非中心方差值。

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}$$
$$\hat{v}_t = \frac{v_t}{1-\beta_2^t}.$$

两个公式分别为梯度的第一个时刻平均值和第二个时刻方差

则参数更新的最终公式为：

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

在实际应用中，Adam方法效果良好。与其他自适应学习率算法相比，其**收敛速度更快，学习效果更为有效**，而且可以**纠正其他优化技术中存在的问题，如学习率消失、收敛过慢或是高方差的参数更新导致损失函数波动较大**等问题。

- https://zhuanlan.zhihu.com/p/50289138
  <https://zhuanlan.zhihu.com/p/50289138>
- https://mp.weixin.qq.com/s/fQkuHnbre1FtNygXCPFIJw
  <https://mp.weixin.qq.com/s/fQkuHnbre1FtNygXCPFIJw>
- https://blog.csdn.net/u012033832/article/details/79460384
  <https://blog.csdn.net/u012033832/article/details/79460384>

15．梯度下降原理

直观解释：比如我们在一座大山上的某处位置，由于我们不知道怎么下山，于是决定走一步算一步，也就是在每走到一个位置的时候，求解当前位置的梯

张晨-ML&DL知识点总结

度，沿着梯度的负方向，也就是当前最陡峭的位置向下走一步，然后继续求解当前位置梯度，向这一步所在位置沿着最陡峭最易下山的位置走一步。这样一步步的走下去，一直走到觉得我们已经到了山脚。当然这样走下去，有可能我们不能走到山脚，而是到了某一个局部的山峰低处。梯度法思想的三要素：出发点、下降方向、下降步长。

https://zhuanlan.zhihu.com/p/55358013 <https://zhuanlan.zhihu.com/p/55358013>

16．为什么梯度下降可以收敛

我们在优化逻辑回归或者神经网络的时候，经常会用到梯度下降法，那么为什么梯度下降法是有效的呢？为什么最终可以让函数f(x)得到最优结果呢？

梯度下降法的过程的表示：

1.选择初始化 $x_0 \in R^d$,和步长(learn_rate) $\eta > 0$

2.for t=0,1...,k 有

$$x_{t+1} = x_t - \eta \nabla f(x_i)$$

假设函数 $f(x)$ 满足L-Lipschitz条件,常量L>0，并且是凸函数，设定 $x^* = \arg\min f(x)$ ，为最优解,

我们假步长 $\eta \leq \frac{1}{L}$ ，进行k次的梯度下降,得到以下公式：

$$f(x_k) \leq f(x) + \frac{\|x_0 - x\|_2^2}{2\eta k}$$

从上面公式可以看出，当k不断的进行变大的时候， $f(x_t)$ 会越来越接近 $f(x^*)$,所以这个就说明了梯度下降是一定能够收敛不的，收敛的率是 $O(\frac{1}{k})$

f(x)满足L-Lipschitz证明及收敛公式推导：

https://zhuanlan.zhihu.com/p/92151073 <https://zhuanlan.zhihu.com/p/92151073>

17．梯度下降一定可以收敛到全局最优值吗

梯度下降不一定能够找到全局的最优解，有可能是一个局部最优解。当然，如果损失函数是凸函数，梯度下降法得到的解就一定是全局最优解。

18．如何判断函数是凸函数

张晨-ML&DL知识点总结

对于一元函数f(x)，我们可以通过其二阶导数f″(x) 的符号来判断。如果函数的二阶导数总是非负，即f″(x)≥0 ，则f(x)是凸函数；

对于多元函数f(X)，我们可以通过其Hessian矩阵（Hessian矩阵是由多元函数的二阶导数组成的方阵）的正定性来判断。如果Hessian矩阵是半正定矩阵，则f(X)是凸函数。

https://www.cnblogs.com/always-fight/p/9377554.html
<https://www.cnblogs.com/always-fight/p/9377554.html>

19．Hessian矩阵的正定性判断

如果 F 的所有二阶导数都存在，则 F 的Hessian矩阵[9]为：

$$H(F)(\boldsymbol{x}) = \begin{bmatrix} \frac{\partial^2 F}{\partial x_1^2} & \frac{\partial^F}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^F}{\partial x_1 \partial x_n} \\ \frac{\partial^2 F}{\partial x_2 \partial x_1} & \frac{\partial^F}{\partial x_2^2} & \cdots & \frac{\partial^F}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 F}{\partial x_n \partial x_1} & \frac{\partial^F}{\partial x_n \partial x_2} & \cdots & \frac{\partial^F}{\partial x_n^2} \end{bmatrix}$$

正定矩阵：给出一个 $n \times n$ 的矩阵 A，如果对于任意长度为 n 的非零向量，都有 $x^T Ax > 0$ 成立，则 A 是正定矩阵。等价于 A 的所有特征值都是正的。等价于 A 的顺序主子式都是正的。

半正定矩阵：给出一个 $n \times n$ 的矩阵 A，如果对于任意长度为 n 的非零向量，都有 $x^T Ax \geq 0$ 成立，则 A 是半正定矩阵。

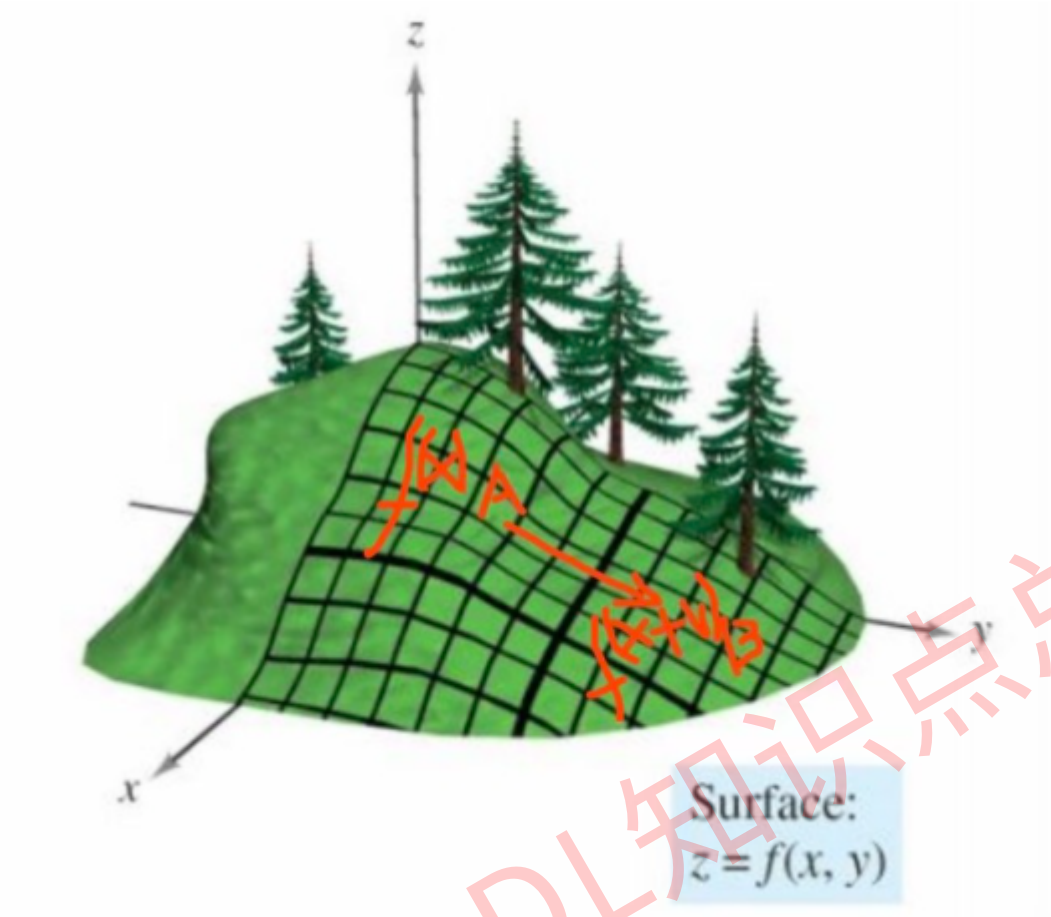
负定矩阵：给出一个 $n \times n$ 的矩阵 A，如果对于任意长度为 n 的非零向量，都有 $x^T Ax < 0$ 成立，则 A 是负定矩阵。等价于 A 的所有特征值都是负的。

半负定矩阵：给出一个 $n \times n$ 的矩阵 A，如果对于任意长度为 n 的非零向量，都有 $x^T Ax \leq 0$ 成立，则 A 是半负定矩阵。

20．为什么梯度的负方向是局部下降最快的方向

张晨-ML&DL知识点总结

当我们在某个要优化的函数，这里设为 [公式] ,我们在x点处，然后沿方向 v 进行移动，到达 $f(x+v)$ ，图示表示了移动过程：



上图显示了从A点移动到B点的过程。那么 v 方向是什么的时候，局部下降的最快呢?

换成数学语言来说就是， $f(x+v)-f(x)$ 的值在 v 是什么的时候，达到最大!

下面进行讲解:

对 $f(x+v)$ 在 x 处进行 Taylor 一阶展开，

$$f(x+v) \approx f(x)+\nabla f(x)^T v ,$$

这里有，

$$f(x)-f(x+v) \approx -\nabla f(x)^T v$$

则 $f(x+v)-f(x)=df(x)v$,则我们可以得出： $df(x)v$ 为函数值的变化量，我们要注意的是 $df(x)$ 和 v 均为向量， $df(x)v$ 也就是两个向量进行点积，而向量进行点积的最大值，也就是两者共线的时候，也就是说 v 的方向和 $df(x)$ 方向相同的时候，点积值最大，这个点积值也代表了从A点到B点的上升量。点积说明如下：

张晨-ML&DL知识点总结

设二维空间内有两个向量 $\vec{a}$ 和 $\vec{b}$，它们的夹角为 $\theta(0 \leq \theta \leq \pi)$，则内积定义为以下实数：

$$\vec{a} \bullet \vec{b} = |\vec{a}||\vec{b}|\cos\theta$$

而 $df(x)$ 正是代表函数值在 x 处的**梯度**。前面又说明了 v 的方向和 $df(x)$ 方向相同的时候，点积值（变化值）最大，所以说明了梯度方向是函数局部上升最快的方向。也就证明了梯度的负方向是局部下降最快的方向！

21．神经网络训练不收敛或训练失败的原因总结

在面对模型不收敛的时候，首先要保证训练的次数够多。在训练过程中，loss并不是一直在下降，准确率一直在提升的，会有一些震荡存在。只要总体趋势是在收敛就行。若训练次数够多（一般上千次，上万次，或者几十个epoch）没收敛，再考虑采取措施解决。

没有对数据进行预处理。数据分类标注是否准确？数据是否干净？

- 没有对数据进行归一化。由于不同评价指标往往具有不同的量纲和量纲单位，这样的情况会影响到数据分析的结果，为了消除指标之间的量纲影响，需要进行数据标准化处理，以解决数据指标之间的可比性。
- 样本的信息量太大导致网络不足以fit住整个样本空间。
- 标签的设置是否正确。
- 网络设定不合理。如果做很复杂的分类任务，却只用了很浅的网络，可能会导致训练难以收敛。
- Learning rate不合适。如果太大，会造成不收敛，如果太小，会造成收敛速度非常慢。
- 隐层神经元数量错误。在一些情况下使用过多或过少的神经元数量都会使得网络很难训练。太少的神经元数量没有能力来表达任务，而太多的神经元数量会导致训练缓慢，并且网络很难清除一些噪声。
- 错误初始化网络参数。
- 没有正则化。正则化典型的就是dropout、加噪声等。即使数据量很大或者你觉得网络不可能出现过拟合，但是对网络进行正则化还是很有必要的。
- Batch Size 过大。Batch size 设置的过大会降低网络的准确度，因为它降低了梯度下降的随机性。另外，在相同情况下batch size 越大那么要达到相同的精确度通常需要训练更多的epoch。
- 网络存在坏梯度。如果你训练了几个epoch误差没有改变,那可能是你使用

张昌 ML&DL知识点总结

了Relu，可以尝试将激活函数换成leaky Relu。

https://zhuanlan.zhihu.com/p/464844046
<https://zhuanlan.zhihu.com/p/464844046>

22．如何通过train loss与test loss分析网络当下的状况

- train loss 不断下降，test loss不断下降，说明网络仍在学习;
- train loss 不断下降，test loss趋于不变，说明网络过拟合;
- train loss 趋于不变，test loss不断下降，说明数据集100%有问题;
- train loss 趋于不变，test loss趋于不变，说明学习遇到瓶颈，需要减小学习率或批量数目;
- train loss 不断上升，test loss不断上升，说明网络结构设计不当，训练超参数设置不当，数据集经过清洗等问题。

23．如何防止梯度下降陷入局部最优解

24．梯度下降的收敛条件是什么

张晨-ML&DL知识点总结