第十四章 超参数调整

Markdown Revision 1;
Date: 2018/10/25
Editor: 乔成磊-同济大学
Contact: qchl0318@163.com
Updater: sjsdfg，王超锋

14.1 写在前面

关于训练深度学习模型最难的事情之一是你要处理的参数的数量。无论是从网络本身的层宽（宽度）、层数（深度）、连接方式，还是损失函数的超参数设计和调试，亦或者是学习率、批样本数量、优化器参数等等。这些大量的参数都会有网络模型最终的有效容限直接或者间接的影响。面对如此众多的参数，如果我们要一一对其优化调整，所需的无论是时间、资源都是不切实际。结果证实一些超参数比其它的更为重要，因此认识各个超参数的作用和其可能会造成的影响是深度学习训练中必不可少的一项重要技能。

目前，超参数调整一般分为手动调整和自动优化超参数两种。本章节不会过多阐述所有超参数的详细原理，如果需要了解这部分，您可以翻阅前面的基础章节或者查阅相关文献资料。当然，下面会讲到的一些超参数优化的建议是根据笔者们的实践以及部分文献资料得到认知建议，并不是非常严格且一定有效的，很多研究者可能会很不同意某些的观点或有着不同的直觉，这都是可保留讨论的，因为这很依赖于数据本身情况。

14.2 参数和超参数的区别

区分两者最大的一点就是是否通过数据来进行调整，模型参数通常是有数据来驱动调整，超参数则不需要数据来驱动，而是在训练前或者训练中人为的进行调整的参数。例如卷积核的具体核参数就是指模型参数，这是有数据驱动的。而学习率则是人为来进行调整的超参数。这里需要注意的是，通常情况下卷积核数量、卷积核尺寸这些也是超参数，因为网络设计完以后，这些参数同样不是由数据驱动的，注意与卷积核的核参数区分。

14.2 神经网络中一般包含哪些超参数

通常可以将超参数分为三类：网络参数、优化参数、正则化参数。

网络参数：可指网络层与层之间的交互方式（相加、相乘或者串接等）、卷积核数量和卷积核尺寸、网络层数（也称深度）和激活函数等。

优化参数：一般指学习率（learning rate）、批样本数量（batch size）、不同优化器的参数以及部分损失函数的可调参数。

正则化：权重衰减系数，丢弃法比率（dropout）

14.3 模型优化寻找最优解和正则项之间的关系

网络模型优化调整的目的是为了寻找到全局最优解（或者相比更好的局部最优解），而正则项又希望模型尽量拟合到最优。两者通常情况下，存在一定的对立，但两者的目标是一致的，即最小化期望风险。模型优化希望最小化经验风险，而容易陷入过拟合，正则项用来约束模型复杂度。所以如何平衡两者之间的关系，得的最优或者较优的解就是超参数调整优化的目的。

14.4 超参数的重要性顺序

- 首先， 学习率，损失函数上的可调参数。在网络参数、优化参数、正则化参数中最重要的超参数可能就是学习率了。学习率直接控制着训练中网络梯度更新的量级，直接影响着模型的**有效容限能力**；损失函数上的可调参数，这些参数通常情况下需要结合实际的损失函数来调整，大部分情况下这些参数也能很直接的影响到模型的的有效容限能力。这些损失一般可分成三类，第一类辅助损失结合常见的损失函数，起到辅助优化特征表达的作用。例如度量学习中的Center loss，通常结合交叉熵损失伴随一个权重完成一些特定的任务。这种情况下一般建议辅助损失值不高于或者不低于交叉熵损失值的两个数量级；第二类，多任务模型的多个损失函数，每个损失函数之间或独立或相关，用于各自任务，这种情况取决于任务之间本身的相关性，目前笔者并没有一个普适的经验由于提供参考；第三类，独立损失函数，这类损失通常会在特定的任务有显著性的效果。例如RetinaNet中的focal loss，其中的参数γ，α，对最终的效果会产生较大的影响。这类损失通常论文中会给出特定的建议值。
- 其次，批样本数量，动量优化器（Gradient Descent with Momentum）的动量参数β。批样本决定了数量梯度下降的方向。过小的批数量，极端情况下，例如batch size为1，即每个样本都去修正一次梯度方向，样本之间的差异越大越难以收敛。若网络中存在批归一化（batchnorm），batch size过小则更难以收敛，甚至垮掉。这是因为数据样本越少，统计量越不具有代表性，噪声也相应的增加。而过大的batch size，会使得梯度方向基本稳定，容易陷入局部最优解，降低精度。一般参考范围会取在[1:1024]之间，当然这个不是绝对的，需要结合具体场景和样本情况；动量衰减参数 β是计算梯度的指数加权平均数，并利用该值来更新参数，设置为 0.9 是一个常见且效果不错的选择；

- 最后，Adam优化器的超参数、权重衰减系数、丢弃法比率（dropout）和网络参数。在这里说明下，这些参数重要性放在最后**并不等价于这些参数不重要。**而是表示这些参数在大部分实践中**不建议过多尝试**，例如Adam优化器中的$\beta1，\beta2，\epsilon$，常设为 0.9、0.999、10−8就会有不错的表现。权重衰减系数通常会有个建议值，例如0.0005 ，使用建议值即可，不必过多尝试。dropout通常会在全连接层之间使用防止过拟合，建议比率控制在[0.2,0.5]之间。使用dropout时需要特别注意两点：一、在RNN中，如果直接放在memory cell中，循环会放大噪声，扰乱学习。一般会建议放在输入和输出层；二、不建议dropout后直接跟上batchnorm，dropout很可能影响batchnorm计算统计量，导致方差偏移，这种情况下会使得推理阶段出现模型完全垮掉的极端情况；网络参数通常也属于超参数的范围内，通常情况下增加网络层数能增加模型的容限能力，但模型真正有效的容限能力还和样本数量和质量、层之间的关系等有关，所以一般情况下会选择先固定网络层数，调优到一定阶段或者有大量的硬件资源支持可以在网络深度上进行进一步调整。

14.5 超参数如何影响模型性能

超参数	如何影响模型容量	原因	注意事项
学习率	调至最优，提升有效容量	过高或者过低的学习率，都会由于优化失败而导致降低模型有效容限	学习率最优点，在训练的不同时间点都可能变化，所以需要一套有效的学习率衰减策略
损失函数部分超参数	调至最优，提升有效容量	损失函数超参数大部分情况都会可能影响优化，不合适的超参数会使即便是对目标优化非常合适的损失函数同样难以优化模型，降低模型有效容限。	对于部分损失函数超参数其变化会对结果十分敏感，而有些则并不会太影响。在调整时，建议参考论文的推荐值，并在该推荐值数量级上进行最大最小值调试该参数对结果的影响。
批样本数量	过大过小，容易降低有效容量	大部分情况下，选择适合自身硬件容量的批样本数量，并不会对模型容限造成。	在一些特殊的目标函数的设计中，如何选择样本是很可能影响到模型的有效容限的，例如度量学习（metric learning）中的N-pair loss。这类损失因为需要样本的多样性，可能会依赖于批样本数量。
丢弃法	比率降低会提升模型的容量	较少的丢弃参数意味着模型参数量的提升，参数间适应性提升，模型容量提升，但不一定能提升模型有效容限	
权重衰减系数	调至最优，提升有效容量	权重衰减可以有效的起到限制参数变化的幅度，起到一定的正则作用	

超参数	如何影响模型容量	原因	注意事项
优化器动量	调至最优，可能提升有效容量	动量参数通常用来加快训练，同时更容易跳出极值点，避免陷入局部最优解。	
模型深度	同条件下，深度增加，模型容量提升	同条件，下增加深度意味着模型具有更多的参数，更强的拟合能力。	同条件下，深度越深意味着参数越多，需要的时间和硬件资源也越高。
卷积核尺寸	尺寸增加，模型容量提升	增加卷积核尺寸意味着参数量的增加，同条件下，模型参数也相应的增加。	

14.6 为超参数选择合适的范围

超参数	建议范围	注意事项
初始学习率	SGD: [1e-2, 1e-1] momentum: [1e-3, 1e-2] Adagrad: [1e-3, 1e-2] Adadelta: [1e-2, 1e-1] RMSprop: [1e-3, 1e-2] Adam: [1e-3, 1e-2] Adamax: [1e-3, 1e-2] Nadam: [1e-3, 1e-2]	这些范围通常是指从头开始训练的情况。 若是微调，初始学习率可在降低一到两个数量级。
损失函数部分超参数	多个损失函数之间，损失值之间尽量相近，不建议超过或者低于两个数量级	这是指多个损失组合的情况，不一定完全正确。单个损失超参数需结合实际情况。
批样本数量	[1:1024]	
丢弃法比率	[0, 0.5]	
权重衰减系数	[0, 1e-4]	
卷积核尺寸	[7x7],[5x5],[3x3],[1x1], [7x1,1x7]	

14.7 为什么卷积核设计尺寸都是奇数

主要原因有两点：

- 保证像素点中心位置，避免位置信息偏移
- 填充边缘时能保证两边都能填充，原矩阵依然对称

14.8 权重共享的形式有哪些，为什么要权重共享

权重共享的形式：

- 深度学习中，权重共享最具代表性的就是卷积网络的卷积操作。卷积相比于全连接神经网络参数大大减少；
- 多任务网络中，通常为了降低每个任务的计算量，会共享一个骨干网络。
- 一些相同尺度下的结构化递归网络

权重共享的好处：

权重共享一定程度上能增强参数之间的联系，获得更好的共性特征。同时很大程度上降低了网络的参数，节省计算量和计算所需内存（当然，结构化递归并不节省计算量）。此外权重共享能起到很好正则的作用。正则化的目的是为了降低模型复杂度，防止过拟合，而权重共享则正好降低了模型的参数和复杂度。因此一个设计优秀的权重共享方式，在降低计算量的同时，通常会较独享网络有更好的效果。

14.9 什么是微调（fine-tune）

微调（fine-tune），顾名思义指稍微调整参数即可得到优秀的性能，是迁移学习的一种实现方式。微调和从头训练（train from scratch）的本质区别在于模型参数的初始化，train from scratch通常指对网络各类参数进行随机初始化（当然随机初始化也存在一定技巧），随机初始化模型通常不具有任何预测能力，通常需要大量的数据或者特定域的数据进行从零开始的训练，这样需要训练到优秀的模型通常是稍困难的。而微调的网络，网络各类参数已经在其他数据集（例如ImageNet数据集）完成较好调整的，具备了较优秀的表达能力。因此，我们只需要以较小的学习速率在自己所需的数据集领域进行学习即可得到较为优秀的模型。微调通常情况下，无须再重新设计网络结构，预训练模型提供了优秀的结构，只需稍微修改部分层即可。在小数据集上，通常微调的效果比从头训练要好很多，原因在于数据量较小的前提下，训练更多参数容易导致过度拟合。

14.10 微调有哪些不同方法?

以图像分类为例，通常情况下由于不同数据集需要的类别数不同，我们需要修改网络的输出顶层。这种情况下有两种微调方式：

- 不冻结网络模型的任何层，对最后的改动层使用较大的学习率，对未改动层以较小的学习率进行训练全模型训练，进行多轮训练即可。即一步完成训练。
- 冻结除了顶部改动层以外的所有层参数，即不对冻结部分的层进行参数训练更新，进行若干轮的微调训练后，放开顶部层以下的若干层或者全部放开所有层的参数，再次进行若干轮训练即可。即分多步训练。

  以上两种都属于微调。目前由于存在大量优秀的预训练模型，如何确定哪个模型适合自己的任务并能得到最佳性能需要花大量的时间探索。此时，上述的前者是种不错训练方式，你无须进行过多分步的操作。而当探索到一个比较适合的模型时，你不妨可以再次重新尝试下以第二种方式进行训练，或许能得到相比于前者稍高些的性能，因为小数据集上调整过多的参数过拟合的机率也会增大，当然这并不是绝对的。

14.11 微调先冻结底层，训练顶层的原因?

14.10中第二种冻结多步训练的方式。首先冻结除了顶部改动层以外的所有层参数，对顶层进行训练，这个过程可以理解为顶层的域适应训练，主要用来训练适应模型的现有特征空间，防止顶层糟糕的初始化，对已经具备一定表达能力的层的干扰和破坏，影响最终的性能。之后，在很多深度学习框架教程中会使用放开顶层往下一半的层数，继续进行微调。这样的好处在于越底层的特征通常是越通用的特征，越往上其整体的高层次语义越完备，这通过感受野很容易理解。所以，若预训练模型的数据和微调训练的数据语义差异越大（例如ImageNet的预模型用于医学图像的训练），那越往顶层的特征语义差异就越大，因此通常也需要进行相应的调整。

14.12 不同的数据集特性下如何微调?

- 数据集数据量少，数据和原数据集类似。这是通常做法只需修改最后的输出层，训练即可，训练过多参数容易过拟合。
- 数据集数据量少，数据和原数据集差异较大。由于数据差异较大，可以在完成输出顶层的微调后，微调顶层往下一半的层数，进行微调，原因见14.11。
- 数据集数据量大，数据与原数据集差异较大。这种情况下，通常已经不需要用预训练模型进行微调，通常直接重新训练即可。
- 数据集数据量大，数据与原数据类似。这时预训练模型的参数是个很好的初始化，可利用预训练模型放开所有层以较小的学习率微调即可。

14.13 自动化超参数搜索方法有哪些?

目前自动化搜索主要包含网格搜索，随机搜索，基于模型的超参优化

14.14 GAN常用调参策略

14.15 调试策略

#

最后，关于如何搜索超参数的问题，我见过大概两种重要的思想流派或人们通常采用的两种重要但不同的方式。

一种是你照看一个模型，通常是有庞大的数据组，但没有许多计算资源或足够的 CPU 和 GPU 的前提下，基本而言，你只可以一次负担起试验一个模型或一小批模型，在这种情况下，即使当它在试验时，你也可以逐渐改良。比如，第 0 天，你将随机参数初始化，然后开始试验，然后你逐渐观察自己的学习曲线，也许是损失函数 J，或者数据设置误差或其它的东西，在第 1 天内逐渐减少，那这一天末的时候，你可能会说，看，它学习得真不错。我试着增加一点学习速率，看看它会怎样，也许结果证明它做得更好，那是你第二天的表现。两天后，你会说，它依旧做得不错，也许我现在可以填充下 Momentum 或减少变量。然后进入第三天，每天，你都会观察它，不断调整你的参数。也许有一天，你会发现你的学习率太大了，所以你可能又回归之前的模型，像这样，但你可以说是在每天花时间照看此模型，即使是它在许多天或许多星期的试验过程中。所以这是一个人们照料一个模型的方法，观察它的表现，耐心地调试学习率，但那通常是因为你没有足够的计算能力，不能在同一时间试验大量模型时才采取的办法。

另一种方法则是同时试验多种模型，你设置了一些超参数，尽管让它自己运行，或者是一天甚至多天，然后你会获得像这样的学习曲线，这可以是损失函数 J 或实验误差或损失或数据误差的损失，但都是你曲线轨迹的度量。同时你可以开始一个有着不同超参数设定的不同模型，所以，你的第二个模型会生成一个不同的学习曲线，也许是像这样的一条（紫色曲线），我会说这条看起来更好些。与此同时，你可以试验第三种模型，其可能产生一条像这样的学习曲线（红色曲线），还有另一条（绿色曲线），也许这条有所偏离，像这样，等等。或者你可以同时平行试验许多不同的模型，橙色的线就是不同的模型。用这种方式你可以试验许多不同的参数设定，然后只是最后快速选择工作效果最好的那个。在这个例子中，也许这条看起来是最好的（下方绿色曲线）。

所以这两种方式的选择，是由你拥有的计算资源决定的，如果你拥有足够的计算机去平行试验许多模型，那绝对采用鱼子酱方式，尝试许多不同的超参数，看效果怎么样。但在一些应用领域，比如在线广告设置和计算机视觉应用领域，那里的数据太多了，你需要试验大量的模型，所以同时试验大量的模型是很困难的，它的确是依赖于应用的过程。但我看到那些应用熊猫方式多一些的组织，那里，你会像对婴儿一样照看一个模型，调试参数，试着让它工作运转。尽管，当然，甚至是在熊猫方式中，试验一个模型，观察它工作与否，也许第二或第三个星期后，也许我应该建立一个不同的模型（绿色曲线），像熊猫那样照料它，我猜，这样一生中可以培育几个孩子，即使它们一次只有一个孩子或孩子的数量很少。