训练的神经网络不工作？一文带你跨过这37个坑

2017-07-25 机器之心

选自Medium

作者：Slav Ivanov

机器之心编译

参与：黄小天、Smith

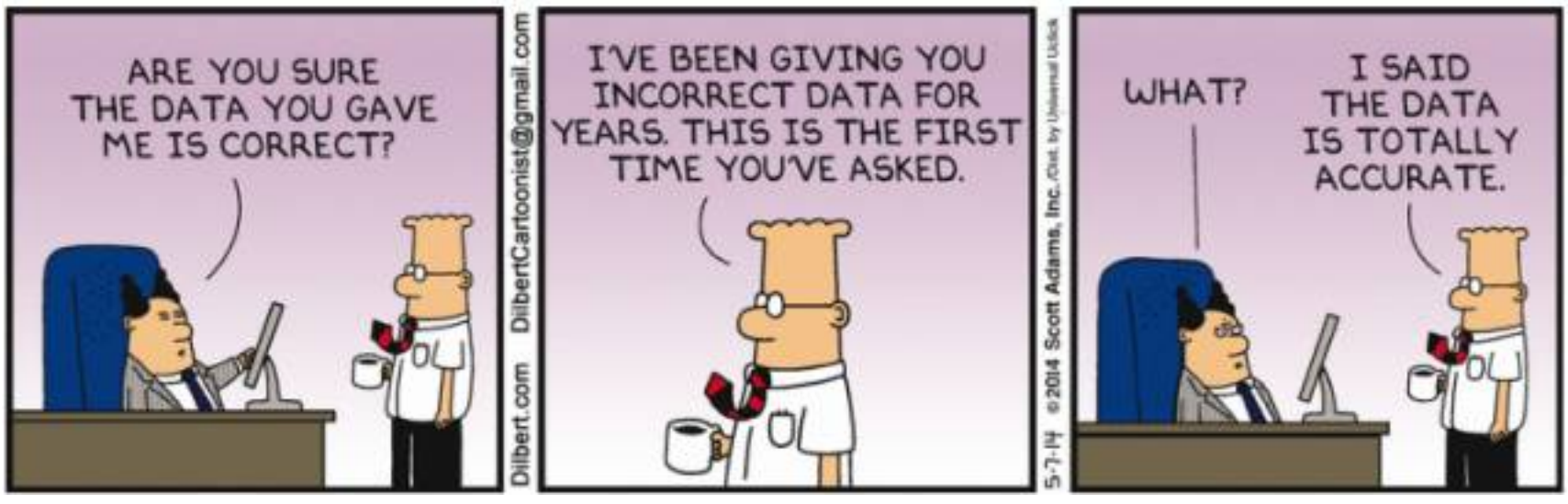
> *近日，Slav Ivanov 在 Medium 上发表了一篇题为《37 Reasons why your Neural Network is not working》的文章，从四个方面（数据集、数据归一化 / 增强、实现、训练），对自己长久以来的神经网络调试经验做了 37 条总结，并穿插了不少出色的个人想法和思考，希望能帮助你跨过神经网络训练中的 37 个大坑。机器之心对该文进行了编译，原文链接请见文末。*

神经网络已经持续训练了 12 个小时。它看起来很好：梯度在变化，损失也在下降。但是预测结果出来了：全部都是零值，全部都是背景，什么也检测不到。我质问我的计算机：「我做错了什么？」，它却无法回答。

如果你的模型正在输出垃圾（比如预测所有输出的平均值，或者它的精确度真的很低），那么你从哪里开始检查呢?

无法训练神经网络的原因有很多，因此通过总结诸多调试，作者发现有一些检查是经常做的。这张列表汇总了作者的经验以及最好的想法，希望也对读者有所帮助。

I. 数据集问题



1. 检查你的输入数据

检查馈送到网络的输入数据是否正确。例如，我不止一次混淆了图像的宽度和高度。有时，我错误地令输入数据全部为零，或者一遍遍地使用同一批数据执行梯度下降。因此打印 / 显示若干批量的输入和目标输出，并确保它们正确。

2. 尝试随机输入

尝试传递随机数而不是真实数据，看看错误的产生方式是否相同。如果是，说明在某些时候你的网络把数据转化为了垃圾。试着逐层调试，并查看出错的地方。

3. 检查数据加载器

你的数据也许很好，但是读取输入数据到网络的代码可能有问题，所以我们应该在所有操作之前打印第一层的输入并进行检查。

4. 确保输入与输出相关联

检查少许输入样本是否有正确的标签，同样也确保 shuffling 输入样本同样对输出标签有效。

5. 输入与输出之间的关系是否太随机?

相较于随机的部分（可以认为股票价格也是这种情况），输入与输出之间的非随机部分也许太小，即输入与输出的关联度太低。没有一个统一的方法来检测它，因为这要看数据的性质。

6. 数据集中是否有太多的噪音?

我曾经遇到过这种情况，当我从一个食品网站抓取一个图像数据集时，错误标签太多以至于网络无法学习。手动检查一些输入样本并查看标签是否大致正确。

7. Shuffle 数据集

如果你的数据集没有被 shuffle，并且有特定的序列（按标签排序），这可能给学习带来不利影响。你可以 shuffle 数据集来避免它，并确保输入和标签都被重新排列。

8. 减少类别失衡

一张类别 B 图像和 1000 张类别 A 图像？如果是这种情况，那么你也许需要平衡你的损失函数或者尝试其他解决类别失衡的方法。

9. 你有足够的训练实例吗?

如果你在从头开始训练一个网络（即不是调试），你很可能需要大量数据。对于图像分类，每个类别你需要 1000 张图像甚至更多。

10. 确保你采用的批量数据不是单一标签

这可能发生在排序数据集中（即前 10000 个样本属于同一个分类）。可通过 shuffle 数据集轻松修复。

11. 缩减批量大小

巨大的批量大小会降低模型的泛化能力（参阅：https://arxiv.org/abs/1609.04836）

II. 数据归一化/增强

12. 归一化特征

你的输入已经归一化到零均值和单位方差了吗?

13. 你是否应用了过量的数据增强?

数据增强有正则化效果（regularizing effect）。过量的数据增强，加上其它形式的正则化（权重 L2，中途退出效应等）可能会导致网络欠拟合（underfit）。

14. 检查你的预训练模型的预处理过程

如果你正在使用一个已经预训练过的模型，确保你现在正在使用的归一化和预处理与之前训练模型时的情况相同。例如，一个图像像素应该在 [0, 1]，[-1, 1] 或 [0, 255] 的范围内吗?

15. 检查训练、验证、测试集的预处理

CS231n 指出了一个常见的陷阱：「任何预处理数据（例如数据均值）必须只在训练数据上进行计算，然后再应用到验证、测试数据中。例如计算均值，然后在整个数据集的每个图像中都减去它，再把数据分发进训练、验证、测试集中，这是一个典型的错误。」此外，要在每一个样本或批量（batch）中检查不同的预处理。

III. 实现的问题

16. 试着解决某一问题的更简易的版本。

这将会有助于找到问题的根源究竟在哪里。例如，如果目标输出是一个物体类别和坐标，那就试着把预测结果仅限制在物体类别当中（尝试去掉坐标）。

17.「碰巧」寻找正确的损失

还是来源于 CS231n 的技巧：用小参数进行初始化，不使用正则化。例如，如果我们有 10 个类别，「碰巧」就意味着我们将会在 10% 的时间里得到正确类别，Softmax 损失是正确类别的负 log 概率： -ln(0.1) = 2.302。然后，试着增加正则化的强度，这样应该会增加损失。

18. 检查你的损失函数

如果你执行的是你自己的损失函数，那么就要检查错误，并且添加单元测试。通常情况下，损失可能会有些不正确，并且损害网络的性能表现。

19. 核实损失输入

如果你正在使用的是框架提供的损失函数，那么要确保你传递给它的东西是它所期望的。例如，在 PyTorch 中，我会混淆 NLLLoss 和 CrossEntropyLoss，因为一个需要 softmax 输入，而另一个不需要。

20. 调整损失权重

如果你的损失由几个更小的损失函数组成，那么确保它们每一个的相应幅值都是正确的。这可能会涉及到测试损失权重的不同组合。

21. 监控其它指标

有时损失并不是衡量你的网络是否被正确训练的最佳预测器。如果可以的话，使用其它指标来帮助你，比如精度。

22. 测试任意的自定义层

你自己在网络中实现过任意层吗？检查并且复核以确保它们的运行符合预期。

23. 检查「冷冻」层或变量

检查你是否无意中阻止了一些层或变量的梯度更新，这些层或变量本来应该是可学的。

24. 扩大网络规模

可能你的网络的表现力不足以采集目标函数。试着加入更多的层，或在全连层中增加更多的隐藏单元。

25. 检查隐维度误差

如果你的输入看上去像（k,H,W）= (64, 64, 64)，那么很容易错过与错误维度相关的误差。给输入维度使用一些

「奇怪」的数值（例如，每一个维度使用不同的质数），并且检查它们是如何通过网络传播的。

26. 探索梯度检查（Gradient checking）

如果你手动实现梯度下降，梯度检查会确保你的反向传播（backpropagation）能像预期中一样工作。

IV. 训练问题

27. 一个真正小的数据集

过拟合数据的一个小子集，并确保其工作。例如，仅使用 1 或 2 个实例训练，并查看你的网络是否学习了区分它们。然后再训练每个分类的更多实例。

28. 检查权重初始化

如果不确定，请使用 Xavier 或 He 初始化。同样，初始化也许会给你带来坏的局部最小值，因此尝试不同的初始化，看看是否有效。

29. 改变你的超参数

或许你正在使用一个很糟糕的超参数集。如果可行，尝试一下网格搜索。

30. 减少正则化

太多的正则化可致使网络严重地欠拟合。减少正则化，比如 dropout、批规范、权重 / 偏差 L2 正则化等。在优秀课程《编程人员的深度学习实战》（http://course.fast.ai）中，Jeremy Howard 建议首先解决欠拟合。这意味着你充分地过拟合数据，并且只有在那时处理过拟合。

31. 给它一些时间

也许你的网络需要更多的时间来训练，在它能做出有意义的预测之前。如果你的损失在稳步下降，那就再多训练一会儿。

32. 从训练模式转换为测试模式

一些框架的层很像批规范、Dropout，而其他的层在训练和测试时表现并不同。转换到适当的模式有助于网络更好地预测。

33. 可视化训练

- 监督每一层的激活值、权重和更新。确保它们的大小匹配。例如，参数更新的大小（权重和偏差）应该是 1-e3。
- 考虑可视化库，比如 Tensorboard 和 Crayon。紧要时你也可以打印权重 / 偏差 / 激活值。
- 寻找平均值远大于 0 的层激活。尝试批规范或者 ELUs。

Deeplearning4j 指出了权重和偏差柱状图中的期望值：对于权重，一些时间之后这些柱状图应该有一个近似高斯的（正常）分布。对于偏差，这些柱状图通常会从 0 开始，并经常以近似高斯（这种情况的一个例外是 LSTM）结束。留意那些向 +/- 无限发散的参数。留意那些变得很大的偏差。这有时可能发生在分类的输出层，如果类别的分布不均匀。

- 检查层更新，它们应该有一个高斯分布。

34. 尝试不同的优化器

优化器的选择不应当妨碍网络的训练，除非你选择了一个特别糟糕的参数。但是，为任务选择一个合适的优化器非常有助于在最短的时间内获得最多的训练。描述你正在使用的算法的论文应当指定优化器；如果没有，我倾向于选择 Adam 或者带有动量的朴素 SGD。

35. 梯度爆炸、梯度消失

- 检查隐蔽层的最新情况，过大的值可能代表梯度爆炸。这时，梯度截断（Gradient clipping）可能会有所帮助。
- 检查隐蔽层的激活值。Deeplearning4j 中有一个很好的指导方针：「一个好的激活值标准差大约在 0.5 到 2.0 之间。明显超过这一范围可能就代表着激活值消失或爆炸。」

36. 增加、减少学习速率

- 低学习速率将会导致你的模型收敛很慢；
- 高学习速率将会在开始阶段减少你的损失，但是可能会导致你很难找到一个好的解决方案。
- 试着把你当前的学习速率乘以 0.1 或 10。

37. 克服 NaNs

据我所知，在训练 RNNs 时得到 NaN（Non-a-Number）是一个很大的问题。一些解决它的方法：

- 减小学习速率，尤其是如果你在前 100 次迭代中就得到了 NaNs。
- NaNs 的出现可能是由于用零作了除数，或用零或负数作了自然对数。
- Russell Stewart 对如何处理 NaNs 很有心得（http://russellsstewart.com/notes/0.html）。
- 尝试逐层评估你的网络，这样就会看见 NaNs 到底出现在了哪里。

原文地址：https://medium.com/@slavivanov/4020854bd607

阅读原文