机器学习项目中的测试与调试

本文翻译自谷歌开发者文档<https://developers.google.com/machine-learning/testing-debugging/>（最后更新于2019年3月27日）

因译者水平有限，如遇疑义，还请尽量查询原版文档。

# 综述

本文主要介绍模型调试和优化的步骤。

## 机器学习调试有哪些不同？

与调试传统程序相比，我们说一个模型质量差，并不是说模型里有个bug。通常为了调试一个表现差的模型，需要调查的起因范围比传统程序多得多。比如说，下面是一部分会导致模型表现不好的原因：特征缺乏预测效果、超参数设置的不够好、数据中有错误和异常值、特征工程部分代码有误。

考虑到漫长的模型训练时间，和更多难以避免的坑，调试机器学习模型是非常有挑战性的工作。

## 机器学习模型的开发步骤

1)从只使用一个或者两个特征的简单模型起步。使用简单易调试的模型可以让你在一开始就走到正确方向（对于NLP和CV方面，当然开始就应该使用一个复杂模型）。

2)尝试不同的特征和超参数来使模型运作正常。切记让模型足够简单来简化调试。

3)通过反复迭代以下步骤优化模型：

增加特征、调参、提高模型容量。

4)每次改变后，重新查看衡量标准和检查模型质量是否提升。如果没有提升，那就需要调试了。

5)在迭代的过程中，确保你增加模型复杂度的举措足够缓慢和持续。

# 数据和特征

低质量的数据显而易见会影响模型表现。在输入的时候就仔细评估低质量数据，比训练完模型后盲估数据哪里不对要容易得多。

## 使用Data Schema来评估输入数据

为了评估数据，工程师需要持续地写一些规则来检测数据符合预期的统计规律。这些规则被称为data schema。通过以下步骤来定义data schema：

1)了解特征数据的范围和分布；对于类别特征，了解它的可能值。

2)将你对数据的了解写成规则。比如：

* 用户允许的评分范围在1到5之间
* 在英文文本中，the总是出现的最多的词
* 类别特征的可能值总是有限的

3)用data schema来检测数据。工程师设计的data schema需要能够找到如下的一些错误：

* 异常值
* 不符合预期的类别特征中的值
* 不符合预期的数据分布

## 确保数据切割的质量

训练数据和测试数据必须能够同等地表征你的输入数据。如果训练数据和测试数据的切割在统计分布上就不同，那么训练数据无法帮助你预测测试数据。监控你切分数据的统计特性分布。如果这些特性分布不一样了，做个标记。另外，确保每次切分时样本的比例保证不变。比如，如果某个项目中你的数据切分为80:20，那么这个比例应该保持不变。

## 测试特征工程后的数据

即使你的原始数据很可靠，喂给模型的是你做过处理的数据。这部分数据和原始的数据看起来已经不太一样了，你需要分别检测它们。基于你对特征工程的了解，写一些单元测试。比如说：

* 所有的数值化特征都是有范围的，比如，在0和1之间。
* One-hot编码的特征向量只有1和N-1个0.
* 缺失数据都由平均值或者默认值替换了。
* 经过转换的数据分布符合预期，比如说，你使用z-scores来归一化数据，那么z-scores的平均值应该是0.
* 异常值(outlier)都处理过了，按比例变化或者裁剪。

# 模型调试

在调试过数据后，工程师们开始调试模型。

## 确保数据可以预测出标签

在调试模型前，先要保证特征确实有预测效果。比如说，可以通过相关矩阵(correlation matrix)来发现单个特征和标签之间的线形相关性。

对于非线形相关性，首先从数据集中挑选出10个容易被模型学习的样本，或者用容易学习的合成(synthetic data)数据。比如说，分类器可以容易学习到线形可分的数据，回归模型可以容易学习到与特征组合(feature cross)相关性高的标签。接下来，需要确保你的模型在这10个容易学习的样本上的损失函数非常小。

由于降低了bug出现的可能性，使用少量易学习的样本能够简化调试过程。你也可以更进一步简化模型，比如用梯度下降法来代替别的高级的优化算法。

## 建立一个基准线（baseline）模型

将训练好的模型与基准线模型相比，是一种非常快速的评估模型优劣的方法。在设计新模型时，可以用一个简单的启发式模型来作为基准线。如果你训练出的模型表现得比基准线差，那么就得提升模型了。

基准线模型举例：

* 只用最有预测效果特征训练出的线形模型。
* 在分类时，将所有数据都分到出现最多的类别里。
* 在回归种，对所有数据只返回平均值。

当你在生产中部署好一个版本后，你可以用这个模型版本做为新版本的基准线。因此，对于不同的复杂度你可以有不同的基准线模型。和基准线模型做比较有益于你捋清往模型增加复杂度的思路。更复杂的模型应该总是比简单模型或者基准线表现得要好。

## 对机器学习代码进行测试

在机器学习代码中寻找bug的过程与传统的调试过程类似，同样是写单元测试。ML codes中的bug举例：

* 隐藏层没有正确设置。
* 数据归一化后返回值为NaNs。

做一个合理性检验(sanity check)，至少模型需要跑起来，否则代码里肯定是有错误的。

## 调参

|  |  |
| --- | --- |
| 超参数 | 描述说明 |
| 学习率  Learning Rate | 通常来说，ML库中都会自动设置学习率。比如，在TensorFlow中，绝大多数TF Estimators使用AdagradOptimizer，初始学习率为0.05并在训练过程中会逐渐调整。另一个很流行的优化器，AdamOptimizer，使用初始学习率为0.001。然而，如果你的模型使用初始值无法收敛，那就人为在0.0001和1.0之间设定学习率，每次提高或者降低需要用对数比例直到模型最终收敛。记住你要解决的问题越困难，在损失函数开始降低前模型需要训练越多的次数。 |
| 正则化  Regularization | 首先，确保你的模型在训练数据上没有正则化也可以预测。接下来，如果你的模型过拟合了，可以增加正则化。正则化的方法分为线性和非线形模型。  对于线形模型，选择L1正则化来降低模型大小，选择L2正则化来增加模型稳定性。增加模型的稳定性能够使得模型的训练更能复现。对于正则化参数todo，从1e-5开始，通过反复实验调试得到正确值。  想要对一个深度模型进行正则化，使用dropout。每步梯度下降，dropout会随机移除一层上的固定比例的神经元。通常，在10%和50%之间的dropout比例能提高模型的generalization. |
| 训练次数  Training epochs | 模型至少用训练集中的全部样本训练一次（epoch），只要没有过拟合，就可以继续训练。 |
| 批大小  Batch size | 通常，mini-batch的大小在10和1000之间。对于随机梯度下降SGD，batchsize的值为1。批大小的上限取决于你的硬件内存能够装下多少数据做训练，下限取决于你的数据和算法。使用较小的batchsize在每个epoch中可以使梯度更新更多次，这样损失函数在每个epoch里降低得更多。进一步说，用更小的batch训练的模型普适性更好（generalize better） |
| 层的深度和宽度  Depth and width | 在神经网络中，深度指层的个数，宽度指每层中神经元的个数。当问题更复杂时，提高模型的深度和宽度。通过以下方法来调整深度和宽度：   * 开始时，只增加一个全连接的隐藏层，这个隐藏层的宽度和你的输入层一样。 * 对于回归类问题，将输出层的宽度设置为1.对于分类问题，将输出层的宽度设置为数据类别的个数。 * 如果模型并不work，并且你觉得模型需要更复杂来解决问题，那么每次增加一个全连接的隐藏层来线性增加模型的深度。隐藏层的宽度取决于具体的问题。通常使用的规则是用和前面的隐藏层相同的宽度，并且通过反复尝试来挑选合适的宽度。   后续层的宽度的调整取决于具体问题。一个比较实际的做法是把每一层的宽度设置为不大于之前一层的宽度。模型的深度和宽度并不一定要完全正确，后续优化模型的时候，你还是会调整它们的值的。 |

## 模型调试小测试

### 模型小测试

你和你的好友热爱独角兽。因为太喜欢独角兽了，你决定用机器学习来预测独角兽的出现。你有一个一万条独角兽现身数据的数据集。对于每一次现身，数据集里有其现身的位置，当天的时间点，海拔，温度，湿度，人群密度，植被，是否有彩虹出现以及其他特征。

你现在要开始开发机器学习模型了，下面的哪一种方法是开始开发的好方法？

1. 从一个简单的线性模型开始，但是要用上所有的特征来保证简单的模型也有预测能力。
2. 独角兽常出现在黎明和黄昏，因此，使用特征‘当天的时间点’来训练一个线性模型。
3. 预测独角兽的出现太困难了，因此，用一个能容纳所有特征的深度神经网络。

### 基准线小测试

使用一个MSE作为损失函数的回归模型，你要乘车费用。可以使用的特征有这趟行程的时间，距离，起点和终点。你已经知道：

* 平均乘车费用为15￥。
* 乘车费用每公里增加固定数额。
* 在市区乘坐要额外收费。
* 起步价3￥。

确定以下基准线模型是否有用，选择是或者否：

1. 每次乘车费用都是15￥。
2. 只用持续时间和起点作为特征的模型
3. 乘车费用就是用乘车距离乘以每公里的费用

4)每次乘车费用都是1￥。因为模型必须总是要超过这个基准线，如果一个模型不能超过这个基准线，那么我们可以确定模型有bug。

### 超参数

接下来的问题会描述训练分类器时遇到的一些问题，挑选能够解决这些问题的措施。

1)训练损失为0.24，验证损失为0.36，下列哪两个行为可以减少训练损失和验证损失之间的差距？

A. 确保训练集和验证集有相同的统计特性。

B. 使用正则化来防止过拟合。

C. 减小学习率。

D. 增大训练的epoch次数。

2)在上一步骤中你选择了正确答案，这时你的训练损失和验证损失经过很多epoch从1.0变成了0.24左右，接下来你可以选择哪一个行为来进一步降低你的训练损失？

A. 增加神经网络的深度和宽度。

B. 增加训练的epoch。

C. 增大学习率。

3)在上一个步骤中你又选择了正确答案，这时候你的训练损失到了0.20.假设你还需要再减小你的训练损失。你增加了一些看起来有预测效果的特征。然而，训练损失在0.20左右反复横跳，接下来你可以选择哪三个行为来进一步降低你的训练损失？

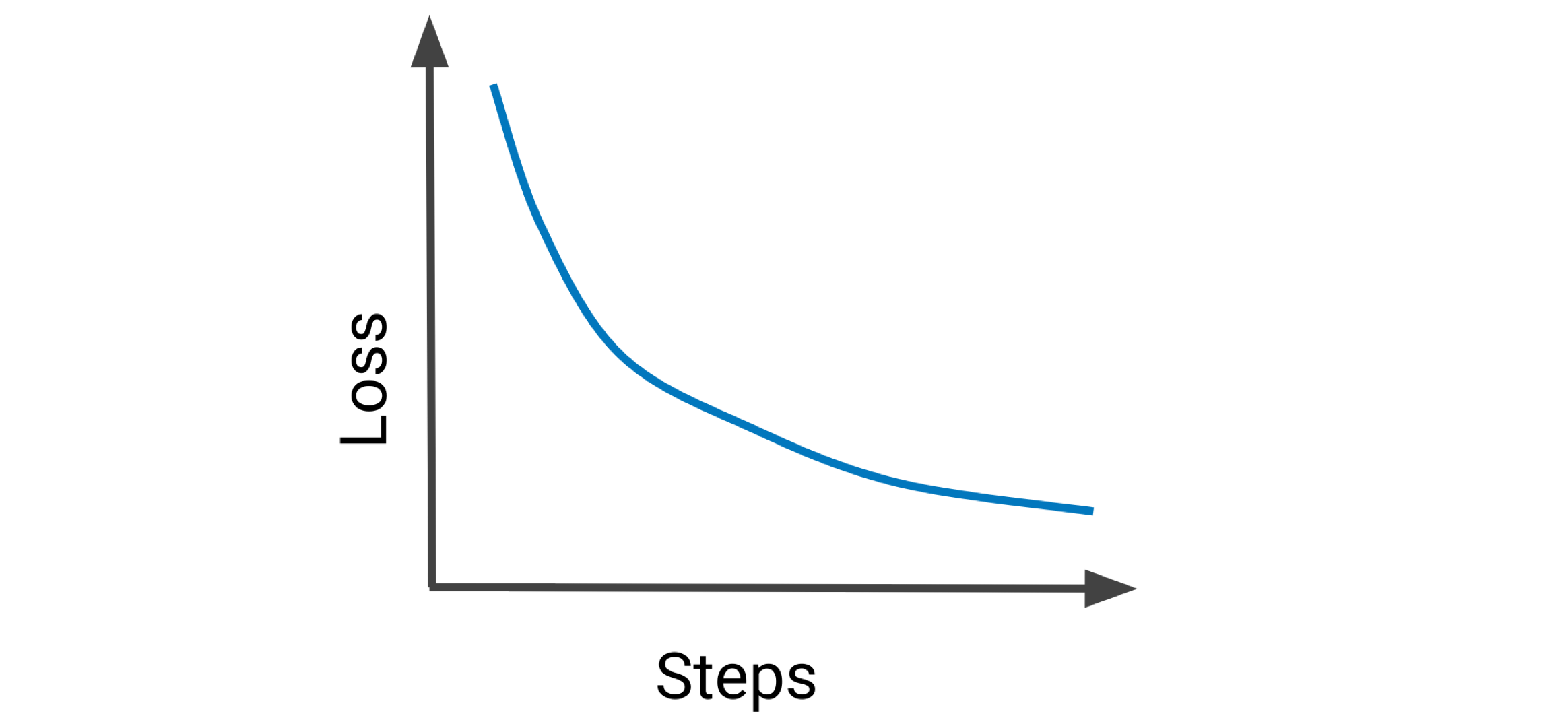
A. 增加训练的epoch

B. 减小学习率

C. 这个特征相对于已有特征来说并不是那么有用，试试别的特征。

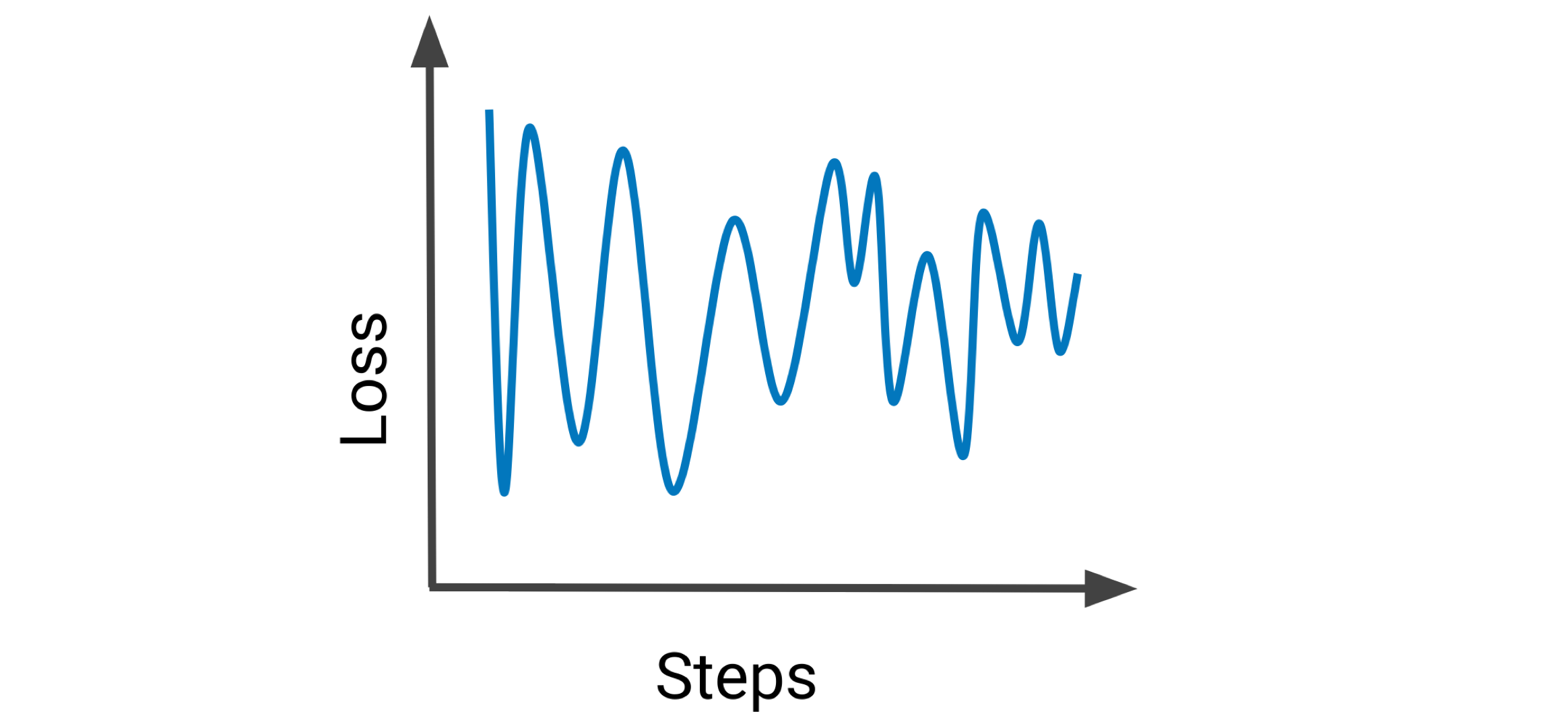
D. 增加网络的深度和宽度。

# Loss Curve损失曲线的解读



如果机器学习项目的损失曲线一开始都有如上图分布的话，生活该多么美好。

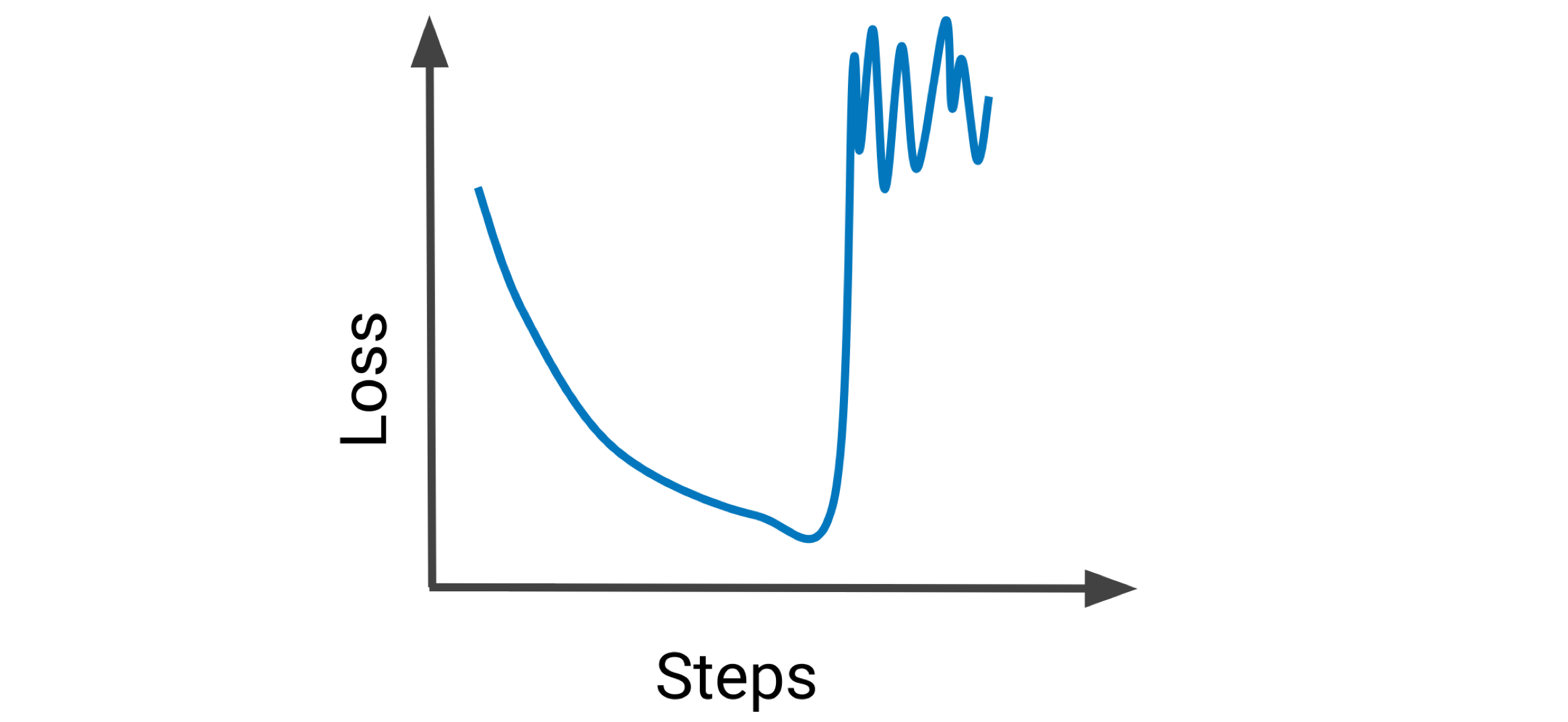
## 模型无法收敛



这种情况下，模型并不收敛。可以尝试以下步骤：

* 用模型调试中的方法检查特征是否能够成功地预测标签。
* 使用data schema来检出坏样本。
* 如果训练过程并不稳定，如上图所示，降低学习率来防止模型在参数空间里反复横跳。
* 将数据集简化为10个样本（你相信模型可以预测出的样本数据），在这个精简版的数据集上获取一个非常小的loss，接下来在完整的数据集上进行调试。
* 简化模型，确认这个模型能够超过你的基准线，接下来慢慢地增加模型的复杂度。

## 损失函数突然爆炸

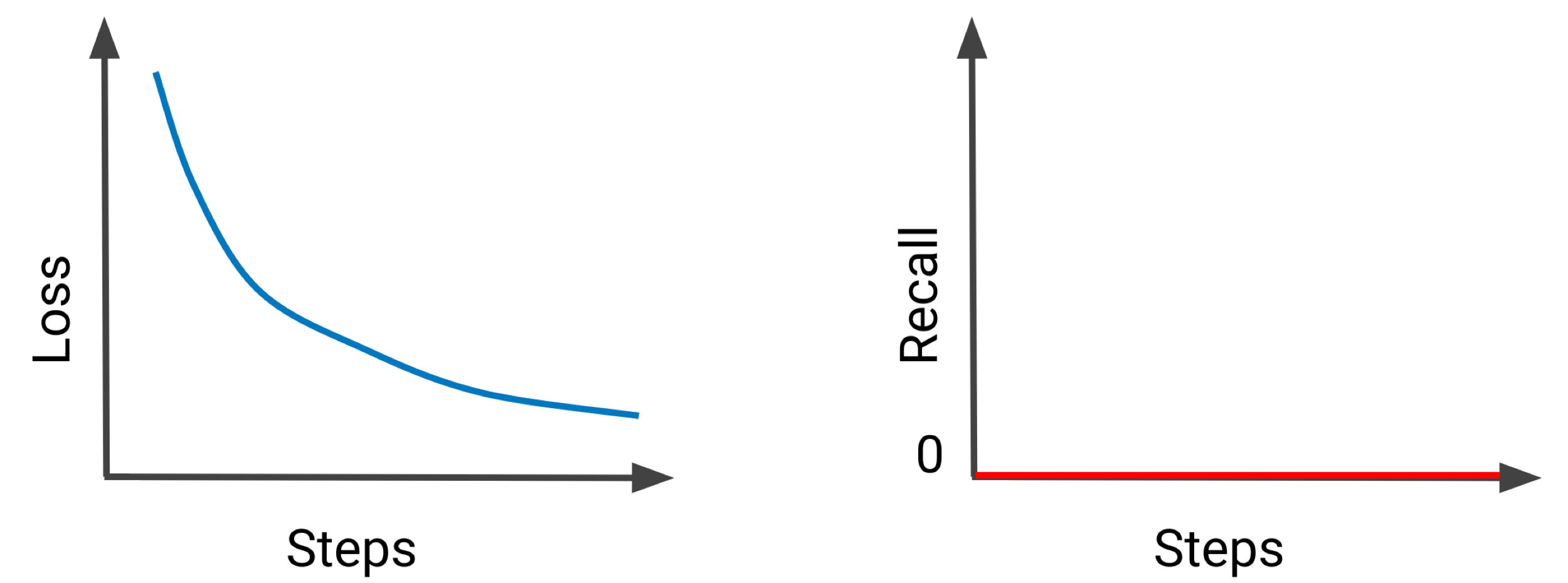


损失函数的突然增加通常是由输入中的异常数据引起的，可能的原因如下：

* 输入数据中的NaN
* 异常数据引起的梯度爆炸
* 有对0的除法
* 对0或者负数做对数

为了解决这类问题，检查这批次数据中的异常数据，和你处理过的特征数据。如果这个异常看起来问题很大，仔细调查原因。否则，如果这个异常看起来只是由于一点异常数据引起的，那么通过混洗（shuffle）数据，确保这类异常数据在你的批数据中同等分布。

## 召回率近似于0

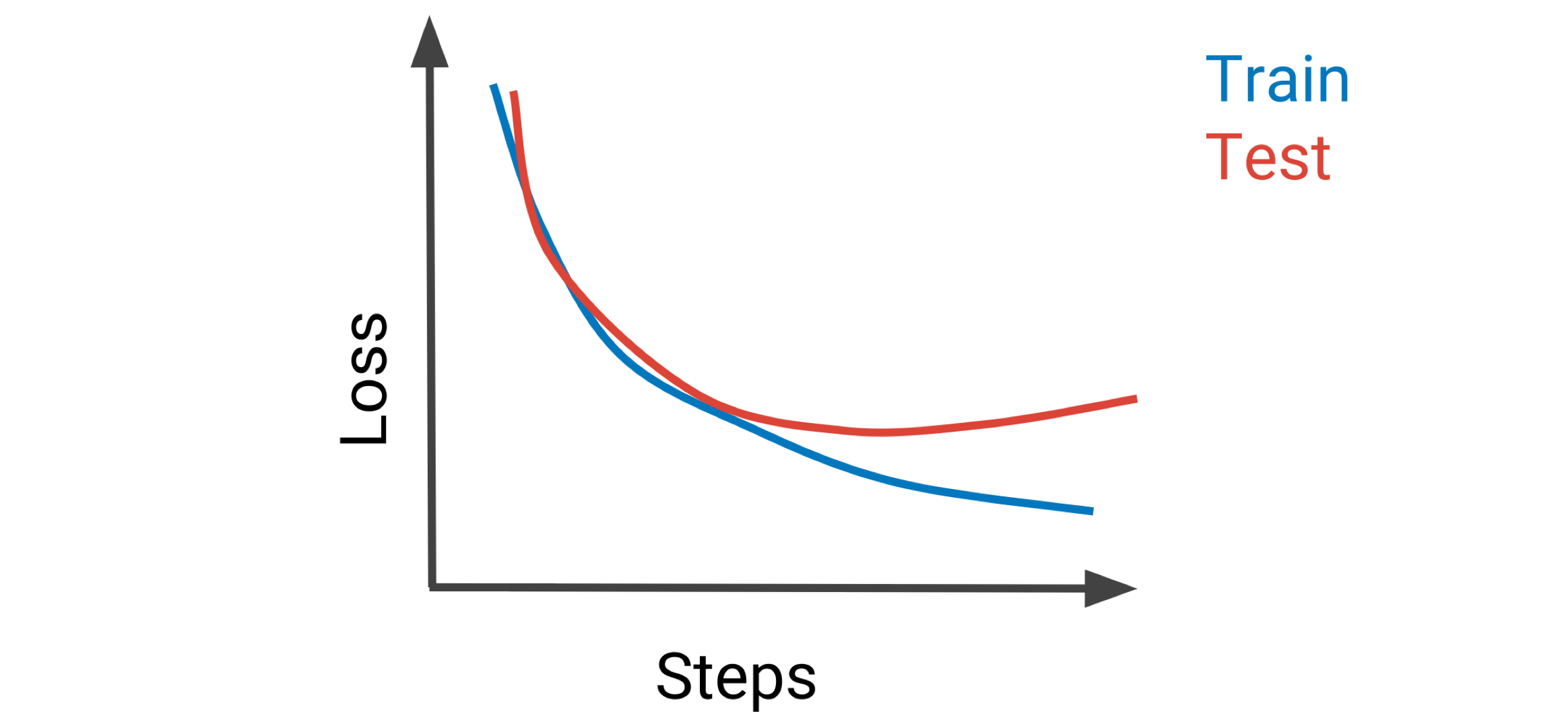


召回率一直是0，因为模型输出的分类概率值一直比认定为positive的阈值（threshold）要小。这种情况经常发生在样本非常不平衡时。记住很多机器学习的库，比如TF Keras，默认使用0.5作为阈值来衡量分类。

尝试以下步骤：

* 降低分类器的阈值
* 使用与阈值无关的衡量函数，比如AUC

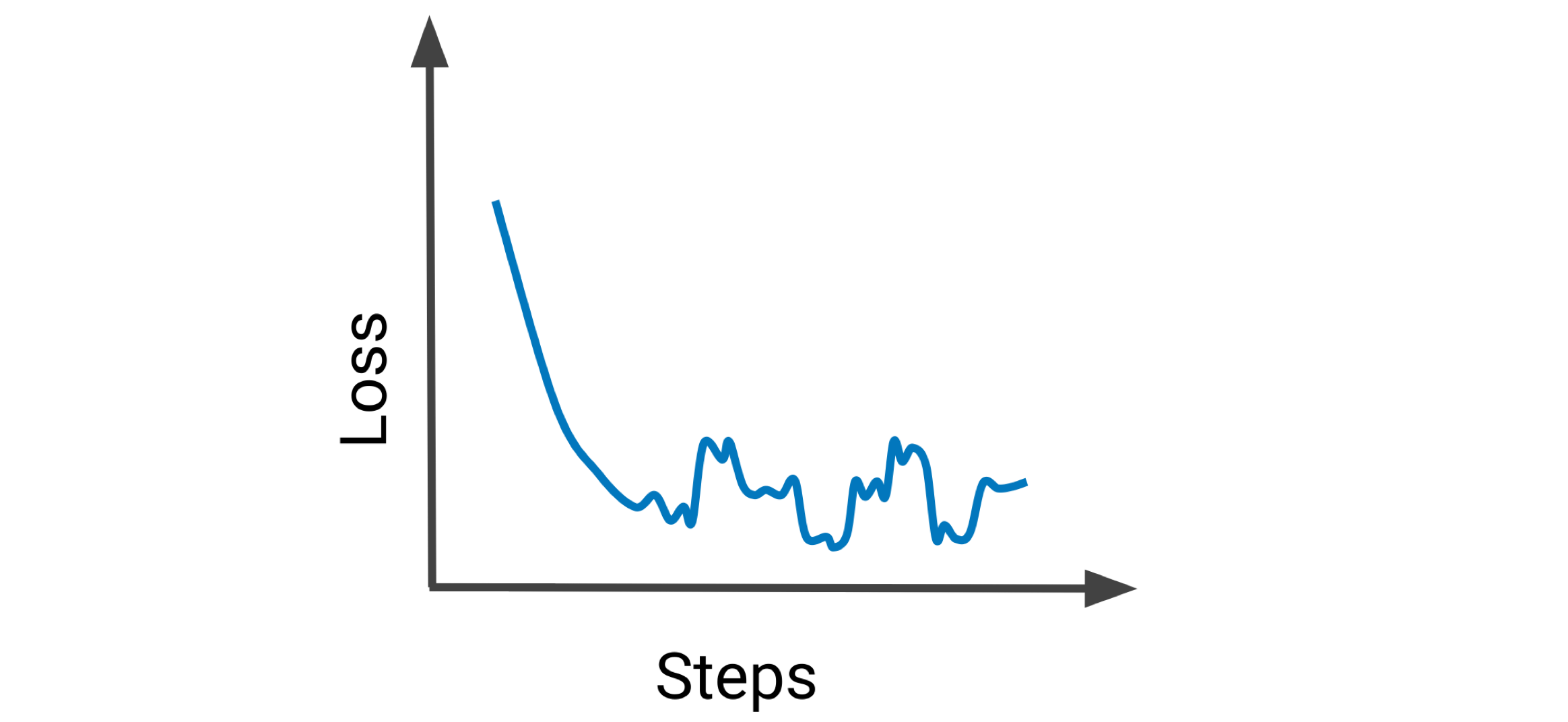
## 过拟合



过拟合，试试下面的措施：

* 降低模型容量。
* 增加正则化。
* 检查训练和测试数据的分割是统计平等的。

## 损失函数来回反复



如果你的损失函数体现出了反反复复重复的趋势，很可能是喂给模型的输入数据本身有着重复的行为特征。确保通过混洗(shuffling)将输入数据中的重复性去掉。

# Model Metrics

## 使用评估指标来评估质量

比较常用的metrics有：Loss、accuracy、precision&recall、AUC

|  |  |
| --- | --- |
| 问题类型 | 评估质量 |
| Regression | 除了降低绝对的均方误差Mean Square Error（MSE）之外，也可以根据你的标签的值来衡量相对的均方误差。比如说，假设你需要预测两个货物的价格，一个均价为5，一个均价为100。两种情况下，你训练的模型的均方误差都为5。对于第一个货物来说，均方误差和平均值一样大了，这个模型的误差显然非常大。对于第二个货物，均方误差仅为平均值的5%，这个误差看起来可以接受。 |
| Multiclass  classification | 对于多分类问题，如果需要预测的数据的类别很少，那么分开查看每种分类的评估指标即可。在多个类别上做分类时，你可以平均每种类别上的评估指标来获得对整个分类效果的评估指标；另外，也可以根据实际需求着重关注某些评估指标。比如说，在图片中寻找目标时，找到人的重要性要在找到其他类型的目标之上。 |

## 在重要的切片数据上检查评估指标

当你拥有一个表现很好的模型后，这个模型可能在少部分数据上仍然表现不佳。比如说，你的独角兽预测器必须能在撒哈拉沙漠和纽约有同样的效果，不分时间。然而，撒哈拉沙漠里的数据量很少。因此，你着重想要评估模型在撒哈拉沙漠上的质量。这种少部分的数据，叫做数据切片(data slices)。在评估模型效果时，你需要分别监控这种你特别关心效果的数据，以及模型可能表现不佳的数据。

通过你对数据的了解来确认你需要关心的data slices。接下来比较data slices的评估指标和你整个数据集上的评估指标。在所有的data slices上检验过模型表现后，你的模型能够有效移除bias。

## 使用实际世界中的评估指标

有时，模型的评估指标并不一定能够反映你的模型在现实世界中的影响。比如说，你可以调节一个超参数并且提高了模型的AUC值，但这个如何影响到用户体验？为了衡量实际应用场景的影响，你需要定义不同的评估指标。比如说，你可以问每一个看到独角兽预测出现的人，他们是否真的看到了独角兽。衡量现实中的影响能够帮助你比较模型不同迭代的质量。

## 模型度量小测试

### 回归问题

你正在预测独角兽出现的时长，单位是分钟。你的模型的验证损失MSE是15，对于一个MSE为25分钟的基准线模型来说，你的模型效果有提升吗？

是，否

### 现实指标

对于你的独角兽出现预测器，下列哪两个实际指标可以补充你的模型指标？

A.如果用户在你的app上看到一个预测，然后在下一次继续使用了你的app。

B.看到了流星，因为独角兽可以呼唤流星。

C.玫瑰产量，因为独角兽会吃掉很多玫瑰。

D.网上关于独角兽的报道。

# 模型优化

模型跑起来后，是时候来进行模型效果的优化了。

## 增加有用的特征

通过增加与当前特征中蕴含信息不同的特征可以提高模型的表现。可以用相关矩阵来评估单个特征和模型的线形相关性。如果想要评估非线性相关性，可以分别用包含这个特征和不包含这个特征的特征集来训练模型，并检查模型效果是否有提升。一个特征是否有效果，必须以能够提升模型效果为标准。

## 调参

之前的参数使得模型能够跑起来，但是想要达到更好的效果，参数是需要调整的。你可以通过反复尝试来人工炼丹，当然，人工调试总是很耗时间。考虑能够自动调参的服务吧，比如Cloud ML Hyperparameter Tuning。

## 调节模型的深度与宽度

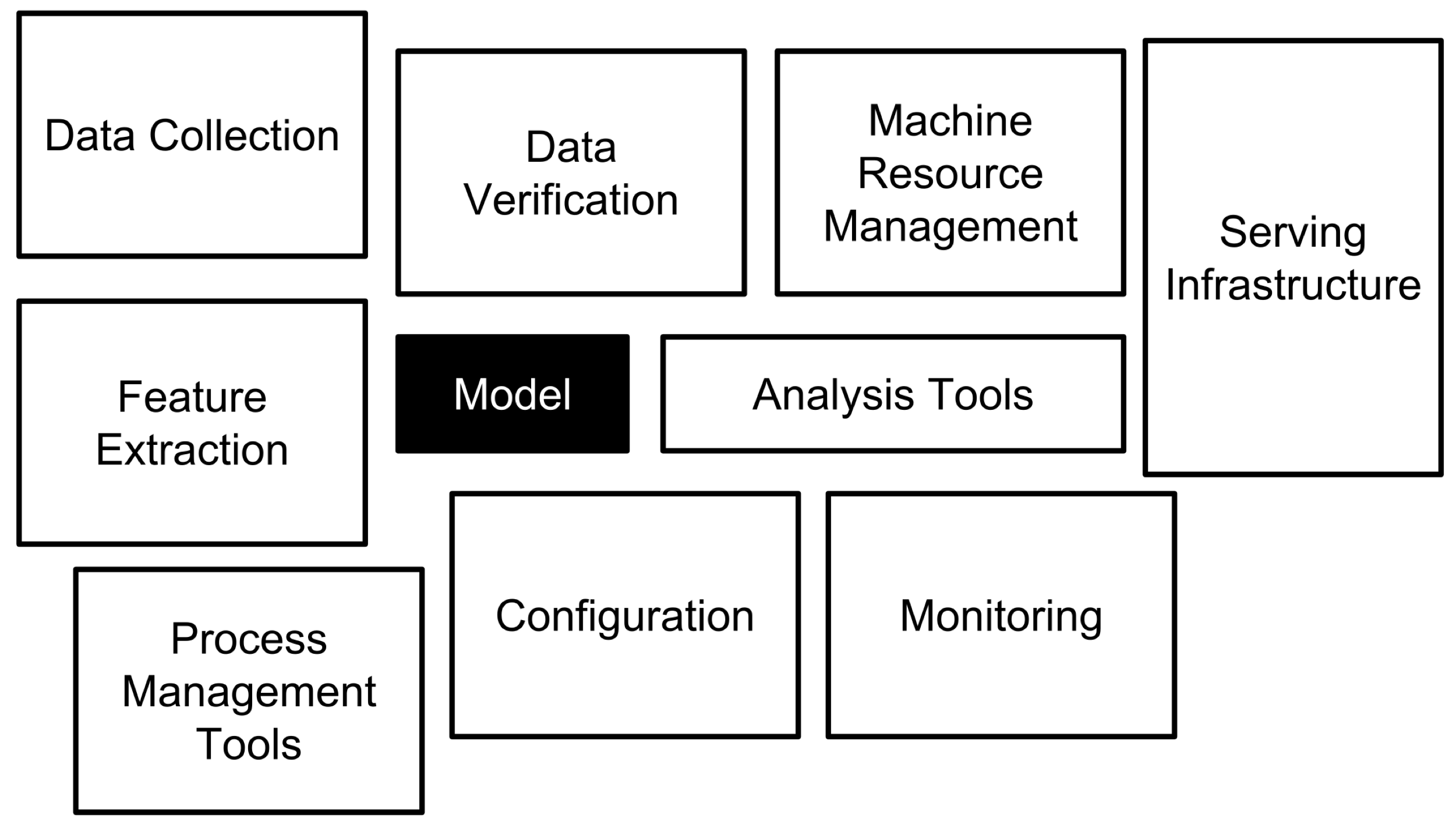
在调试模型时，你只是增加了模型的深度和宽度。而在模型优化过程中，你可能要根据需要增加或者降低深度和宽度。如果模型的表现足够好，为了避免发生过拟合和降低训练时间，可以减少深度和宽度。具体来说，试着减半每个后续层的宽度。这时模型的表现也会降低，你需要在过拟合和训练时间中挑选一个平衡点。

而反过来说，如果你需要更好的模型效果，试着增加深度和宽度吧。

当然，模型的深度和宽度也属于超参数，自动调参也可以用来优化深度和宽度。

# ML Pipelines 机器学习工作流

机器学习工作流包括许多组成部分，如下图所示，接下来我们会熟悉这些部分。注意黑盒标注的‘Model’仅仅是实际生产的机器学习架构中的一个小部分。



## 在机器学习工作流中测试的角色

传统软件开发中，理想的工作流是技术驱动开发(test-driven development, TDD)。然而在机器学习中，由测试开始一个项目很难直接实现。工程师的测试取决于数据，模型，和具体问题。比如说，在训练模型之前，人们无法写一个测试来确定损失值。相应地，在模型开发的过程中，工程师能够找到一个可以追求的损失值，新版本的模型就可以相对这个损失值来测试。

你需要测试的内容有：

* 确保输入数据无误。
* 确保特征工程无误。
* 确保新版本的模型质量没有问题。
* 确保部署架构没有问题。
* 测试机器学习工作流中各个组件的集成性(integration)。

# 部署机器学习模型的测试

## 使用可复现(reproducible)的训练过程来测试模型更新

毫无疑问，你想要提升你的独角兽出现预测工具。假设你重构了time of day这个特征的特征工程代码，那么如何来测试确保这段代码是正确的？你决定重新训练模型来查看是否得到了相同的结果，但是遗憾地发现模型的训练结果无法重复。为了继续预测独角兽的出现，你继续研究，接着发现了如下步骤可以实现模型的可复现：

* 人为设定随机数生成器（random number generator, RNG）。有关这部分的细节请参照另外一门数据准备和特征工程课程。
* 提前固定模型的各个组件的顺序，确保每个组件在每次运行时都从RNG中获取了同样的随机数。一般来说，机器学习库会自动为你解决这个需求。
* 对模型多次训练的结果取平均。
* 使用版本管理，即使在准备工作的迭代中，这样在调查模型和工作流的时候你就可以精确定位代码和参数了。

即使采取了上述措施，你仍有可能面对一些不确定性。

## 针对版本和API请求测试模型更新

将模型更新到独角兽预测器2.0后，你需要测试新模型的算法是否正确，和相应的API请求是否发生变化。

### 测试API请求

如何测试API请求的更新呢？你当然可以重复训练模型，但这非常耗费时间。你可以写一个单元测试，随机生成输入数据，接下来只跑一个步骤的梯度下降，按照预期，这将顺利完成并且没有是runtime错误。

### 测试算法是否正确

一个模型，不光要能预测正确地结果，还要保证它本身算法上就是正确的。比如说，假如99%的邮件都不是垃圾邮件，那么直接把所有的邮件都判定为不是垃圾邮件也可以获得99%的准确率，但这毫无意义。因此，你需要确保模型本身的算法就是对的。采取以下步骤：

* 训练几次迭代，确保损失值在下降。
* 去掉正则化来训练模型，如果模型足够复杂，它会记住所有的训练数据，并且最终训练集的损失函数会趋近于0。
* 将算法整个的计算过程分解，并测试其中特定的部分是否正确。比如说，你可以用输入数据中的每个元素一次来测试一部分RNN代码的准确性。

注意：由于训练过程可能耗时过长，在将模型效果与之前训练好的模型相比较时，你可以尝试只部分训练模型。不用测试这种部分训练的模型，因为这种测试无法控制和解读。

## 对工作流的组件编写集成测试

在一个机器学习工作流中，一个组件的改变可能会导致其他组件产生错误。通过写一个端对端的跑通整个工作流的测试来确保所有的组件一起工作。这样的一个测试叫做集成测试（integration test)。

除了持续集成测试外，每当布置新模型和有新的软件版本时，你都需要跑集成测试。运行整个工作流所耗费的时间使得持续集成测试很慢。为了让集成测试快一些，可以只跑一部分数据集或者使用一个更简单的模型，细节取决于你的模型和数据。为了保障持续集成，每当有新版本的模型和软件上线时候，你可以跑精简版的测试，期间，较慢的（完整的）测试持续在背景中运行着。

## 在部署前确保模型质量

在一个新的模型版本上线前，测试如下两种质量上的下降

* 突然下降：新版本的bug可能会导致严重的质量问题，比照着之前的版本来确保新版本。
* 缓慢下降：对突然下降的测试可能很难测试到几个版本间模型质量的缓慢下降。取而代之的，确保模型在验证集上的预测值能满足一个固定的阈值。如果你的验证集是从live data中获取的，那么及时更新你的验证集并确保模型始终满足同样的质量阈值。

## 在上线前确保模型框架的兼容性

如果模型版本更新得比服务器要快，那么很可能在软件依赖上会有一些问题导致不兼容。通过在沙盒版本的服务器中筹备模型，确保模型使用的操作存在于服务器中。

# 生产测试

终于成功上线独角兽出现预测器后，你希望这个预测器能够不出故障，7X24小时稳定运行。这时你意识到需要监控你的机器学习工作流。虽然监控工作流里所有的组件看上去很令人头大，但让我们先看看需求和解决方案吧。

## 检查训练-生产之间的偏离

训练-生产之间的偏离是指对于训练时和实际在线时，你的输入数据不同。下表中列出两种比较重要的偏离。

|  |  |  |  |
| --- | --- | --- | --- |
| 类型 | 定义 | 举例 | 解决方法 |
| 模式偏离  Schema  skew | 训练和生产的输入数据不符合相同的模式。 | 模型依然是用已有的（旧）数据训练的，生产数据的格式或者分布发生变化。 | 使用同样的data schema来验证训练和生产数据。记得单独检查一些没被你的schema检查到的统计学信息，比如缺失值的比例。 |
| 特征偏离  Feature  skew | 经过特征工程后的训练数据和生产数据的特征不同。 | 比如说，对训练数据和生产数据的特征工程部分代码不同，产生了不同的特征数据。 | 和模式偏离类似，在对训练数据和生产数据做特征工程时，使用同样的统计规则。跟踪被发现的有偏离的特征，以及每个特征中偏离的样本比例。 |

## 监视整个工作流的模型年龄

如果生产数据随着时间改变，而你的模型没有定期重新训练，那么你会发现模型的质量在下降。跟踪模型根据新数据重新训练的时间，并为警报设置一个阈值年龄。除了监视模型在生产中的年龄外，也需要监视整个工作流的模型年龄来捕捉工作流停滞。

## 测试模型的权重和输出在数值上是否稳定

在模型训练过程中，你的权重和层的输出不应该有NaN或者Inf。编写测试来检测你的权重和层输出的NaN和Inf值。另外，还需要测试每层超过半数的输出不是0。

## 监控模型表现

你的独角兽出现预测器比预想得还要成功！你收到了很多很多预测请求和更多的训练数据。这很好，直到你意识到了模型需要更多的内存和时间来训练。你决定通过以下步骤监视模型性能：

* 根据代码、模型和数据的版本来跟踪模型表现。这样的跟踪可以让你精准定位性能下降的确切原因。
* 根据以前的版本和固定的阈值每秒测试新版本模型的训练步骤。
* 设置内存阈值来捕捉内存泄漏。
* 监视API响应时间并跟踪它们的百分比。虽然API响应时间可能不在你的控制范围内，但是较慢的响应可能会导致较差的实际度量。
* 监视每秒应答的查询数量。

## 在生产数据上进行实时模型测试

你已经验证了你的模型。但是如果在现实场景中，比如独角兽行为，在记录验证数据之后发生了改变该怎么办？你的生产模型的表现会下降。然而，在实际中测试质量是困难的，因为现实的数据并不是总有标签的。如果你的生产数据没有标签，考虑以下测试：

* 使用人工来产生标签。
* 调查预测中显示出显著性统计偏差的模型。
* 跟踪模型在实际中的衡量结果。比如说，如果你在监控垃圾邮件，把你的预测结果和用户报告的垃圾邮件进行比较。
* 通过在一小部分查询上上线新模型来减少训练和在线数据间的潜在差异。在验证了新模型后，逐渐把所有查询切换到新版本。
* 使用了这些测试，也要记住监控预测质量的突然下降和缓慢下降。

## 生产测试小测试

在部署了你的独角兽探测器后，你必须不停训练新数据来保证模型不过时。因为你收集了太多的新数据，所以你决定通过在一个时间窗口内的采样来限制训练数据的规模。你还需要考虑独角兽出现的每天和每年的模式（pattern）。而且，你能推出新版本模型的最快时间是三个月。那么你会选择哪一个时间窗口？

A. 一周，这样你的数据集就不会太大，仍然可以平滑模式。

B. 一年，这样你的模型就不会受每天和每年的模式的影响。

C. 一天，因为更长的时间窗口会导致更多数据，模型训练会更久。

独角兽探测器上线了，运行得很好。你去休假了。三周后回来的你发现模型的质量显著下降。假如独角兽本身的行为在三周内不太可能显著变化，这个质量下降最有可能的解释是什么？

A. 你忘记在固定阈值下测试模型质量。

B. 训练和生产的偏离。

C. 你的模型过时了。

因为缺乏南极洲的数据，你机智地决定去检测那里的预测。你的预测结果会神秘地下降几天，特别是在冬日。你觉得理由可能是什么？

A. 没啥理由，机器学习模型本身就有随机性。

B. 你的模型过时了。

C. 一个环境因素。

你的独角兽探测器运行一年了。你已经解决了很多问题，现在模型的质量已经很好了。然而，你注意到了一个很小但是持续很长的问题。你的模型质量会在城市地区略有下降，可能的原因是什么？

A. 城市地区难以建模。

B. 在人流密度高的地方，独角兽的出现被报道了很多次，偏离了你的训练数据。

C. 高质量的预测使得用户很容易能发现独角兽，影响到了独角兽出现这一行为本身。