ML模型调试概述

调试ML模型以使模型工作。一旦您的模型开始工作，您就可以优化模型的质量满足生产的需求。本节介绍调试和优化步骤。

ML调试

模型表现不好的原因包括：

特征预测能力不足

超参没有调整到最优值

数据包含错误或异常值

特征工程代码存在bug

由于运行实验所需的时间，调试ML模型非常复杂。考虑到迭代周期更长，错误空间更大，调试ML模型是一项独特的挑战。

ML模型调试过程

如果遵循开发ML模型的最佳实践，那么调试ML模型将更简单。这些最佳做法如下：

1. 从使用一个或两个特征的简单模型开始。从一个简单、易于调试的模型开始，可以帮助您缩小模型性能差的许多可能原因。
2. 通过尝试不同的特征和超参数值使模型正常工作。保持模型尽可能简单以简化调试。
3. 通过迭代尝试这些更改来优化模型：

增加特征

调整超参

增加模型容量（样本数）

1. 对模型进行每次更改后，重新检查度量并检查模型质量是否提高。如果没有，则按照本课程中的说明调试您的模型。
2. 在迭代过程中，确保您慢慢地、渐进地增加模型的复杂性。

在调试ML模型的时候，您应该首先尝试诊断问题并采用合适的解决方法，如果你使用sgd的优化方法，模型会收敛的快一点，但是模型收敛慢的实际原因可能是学习速率。使用sgd比adam快的原因是，sgd默认的学习速率要比adam快。

当然，也可以在默认学习速率下训练模型更久一点，但是在真实世界中，模型训练的时间本来就很长了，所以你应该尽量缩短训练周期，因此，提高学习率是正确的方法。

这些选项演示了ML中的调试是多维的，因此您必须使用对模型机制的理解来缩小选项的范围。由于在ML中运行实验非常耗时，需要仔细的设置，并且可能会受到再现性问题的影响，因此使用对模型机制的理解来缩小选项范围而不必进行实验是很重要的。

最后，根据开发最佳实践，您应该适当地转换特性数据。此Colab没有转换特征数据，因为收敛不需要转换。但是，您应该始终适当地转换数据在这里，您可以使用z-score规范化您的特征数据，或者将特征数据缩放到[0,1]。

模型训练中的一个常见问题是“爆炸”或变为“nan”的损失。一个常见的原因是异常的特征数据，例如异常值和nan值，或者是高学习率。以下各节说明这些原因。。

在ML中调试的n维特性使ML调试变得困难。

为了有效地调试，理解模型力学是很重要的。

从一个简单的模型开始。

与包含NaNs的原始数据相比，爆炸梯度在模型中不正确的规范化、特性列的错误配置等。

Data和Feature调试

低质量的数据将显著影响模型的性能。相比于在您的模型预测不好的情况下猜测数据是否存在问题，在输入处检测低质量数据更容一些。按照本节中的建议检查数据。

使用数据架构验证输入数据

要监视数据，应通过编写数据必须满足的规则，不断检查数据是否符合预期的统计值。此规则集合称为数据架构。通过以下步骤定义数据架构：

1. 对于特性数据，请了解范围和分布。对于分类特征，请理解可能的值集。
2. 将您的理解编码为模式中定义的规则。规则示例如下：

用户提交的评分数据必须在1~5之间

检查“the”出现的频率是否最高（对于英文文本功能）。

检查分类特征是否具有来自固定集的值。

1. 根据数据架构测试数据。架构应捕获数据错误，例如：

异常值

类别特征不希望的值

不希望的数据分布

确保数据分割质量

您的测试和培训拆分必须同样代表您的输入数据。如果测试和训练分割在统计学上是不同的，那么训练数据将无助于预测测试数据。要了解如何对数据进行采样和拆分，请参阅ML课程中“数据准备和特征工程”中的“采样和拆分数据”部分。

测试特征工程之后的数据

当你的原始数据通过验证，但输入模型的是特征工程之后的数据，这和原始数据存在差异，所以还需要检查特征工程之后的数据。基于对特征工程之后数据的理解，编写测试单元。例如，可以编写单元测试来检查以下条件：

* 所有数字特征都将缩放，例如，在0和1之间。
* 独热编码向量只包含一个1和N-1零。
* 丢失的数据将替换为平均值或默认值。
* 转换后的数据分布符合预期。例如，如果使用z-scores进行规范化，则z-scores的平均值为0。
* 处理异常值，例如通过缩放或剪裁。

模型调试

调试模型的第一步是数据调试。在调试数据之后，请按照以下步骤继续调试模型，详细信息请参见以下部分：

* 检查数据是否可以预测标签。
* 建立一个基线。
* 编写并运行测试。
* 调整超参数值。

检查数据可以预测标签

在调试模型之前，尝试确定您的特征是否编码成可预测的信号。通过使用相关矩阵，可以找到单个特征和标签之间的线性相关性。

然而，相关矩阵不能检测特征和标签之间的非线性相关性。相反，从您的数据集中选择10个示例，您的模型可以很容易地从中学习。或者，使用易于学习的合成数据。例如，分类器可以很容易地学习线性可分离的例子，而回归器可以很容易地学习与特征交叉高度相关的标签。然后，确保您的模型能够在这10个易于学习的示例上实现非常小的损失。

使用几个易于学习的示例可以减少出现错误的机会，从而简化调试。通过切换到更简单的梯度下降算法而不是更高级的优化算法，可以进一步简化模型。

建立一个基线

将模型与基线进行比较是对模型质量的快速测试。当开发一个新的模型时，使用一个简单的启发式方法来预测标签来定义基线。如果你的训练模型表现比基线差，你需要改进你的模型。

基线示例如下：

* 使用仅针对最具预测性特征训练的线性模型。
* 在分类中，总是预测最常见的标签。
* 在回归中，总是预测平均值。

在生产环境中验证模型版本后，可以将该模型版本用作较新模型版本的基线。因此，你可以有多个不同复杂的基线。对基线进行测试有助于证明增加模型的复杂性。一个更复杂的模型应该总是比一个不太复杂的模型或基线表现得更好。

实现ML代码的测试

在ML代码中捕获错误的测试过程类似于传统调试中的测试过程。您将编写单元测试来检测错误。ML中的代码错误示例如下：

* + 未正确配置的隐藏层。
  + 返回NaNs的数据规范化代码。

对于代码错误的存在，一个明智的检查是在特性中包含您的标签并训练您的模型。如果您的模型不工作，那么它肯定有一个bug。

超参调整

超参包括学习速率、正则项、训练迭代次数、批大小、神经网络的深度和宽度