第十一章 实践方法论

使用深度学习

- 不仅仅要知道存在的算法和原理
- 还要知道如何选择方法并根据反馈改进现有的系统

实践者需要决定

- 是否收集更多的数据
- 增加或减少模型容量
- 添加或删除正则化项
- 改进模型的优化
- 改进模型的近似推断
- 调试模型的软件实现

第十一章 实践方法论

实践设计流程

- 确定目标——使用什么样的误差度量，并为此误差度量指定目标值。这些目标和误差度量取决于该应用旨在解决的问题
- 尽快建立一个端到端的工作流程，包括估计合适的性能度量
- 搭建系统，并确定性能瓶颈。检查哪个部分的性能差于预期，以及是否是因为过拟合、欠拟合，或者数据或软件缺陷造成的
- 根据具体观察反复地进行增量式的改动，如收集新数据、调整超参数或改进算法

11.1 性能度量

确定目标，即使用什么误差度量，是必要的第一步

- 不可能实现绝对零误差
  - 输入特征可能无法包含输出变量的完整信息
  - 因为系统可能本质上是随机的
  - 受限于有限的训练数据
- 减少误差的价值，并与收集更多数据的成本做权衡

确定合理的性能期望

- 根据先前公布的基准结果来估计预期错误率
- 一旦你确定了想要达到的错误率，那么你的设计将由如何达到这个错误率来指导

11.1 性能度量

更高级的度量

- 一种错误可能会比另一种错误更严重
  - 将正常邮件错误地归为垃圾邮件
- 精度和召回
  - 精度是模型报告的检测是正确的比率
  - 召回率则是真实事件被检测到的比率
  - PR 曲线
  - F 分数 $$F = \frac{2pr}{p + r}.$$
  - PR 曲线下方的总面积
- 覆盖
  - 覆盖是机器学习系统能够产生响应的样本所占的比率

11.2 默认的基准模型

不同情况下使用哪种算法作为第一个基准方法

- 根据问题的复杂性
  - 线性权重就能解决问题——逻辑回归
  - “AI-完全”类——深度学习模型
- 根据数据的结构选择一类合适的模型
  - 固定大小的向量作为输入的监督学习，那么可以使用全连接的前馈网络
  - 如果输入有已知的拓扑结构（例如，输入是图像），那么可以使用卷积网络
- 优化算法
  - 具有衰减学习率以及动量的SGD
  - Adam算法
- 正则化

11.3 决定是否收集更多数据

怎样判断是否要收集更多的数据

- 确定训练集上的性能是否可接受
  - 训练集上的性能就很差就没必要收集更多的数据
  - 如果更大的模型和仔细调试的优化算法效果不佳，那么问题可能源自训练数据的质量
- 如果更大的模型和仔细调试的优化算法效果不佳，那么问题可能源自训练数据的质量
- 测试集上的性能比训练集的要差得多，那么收集更多的数据是最有效的解决方案之一
- 需要考虑是收集更多数据的代价和可行性
  - 代价高时，替代的简单方法是降低模型大小或是改进正则化
  - 如果收集更多的数据是不可行的，那么改进泛化误差的唯一方法是改进学习算法本身

11.4 选择超参数

有两种选择超参数的基本方法：

- 手动选择
  - 手动选择超参数需要了解超参数做了些什么，以及机器学习模型如何才能取得良好的泛化
- 自动选择
  - 自动选择超参数算法大大减少了解这些想法的需要，但它们往往需要更高的计算成本

11.4.1 手动调整超参数

必须了解超参数、训练误差、泛化误差和计算资源（内存和运行时间）之间的关系

手动搜索超参数的目标

- 通常是最小化受限于运行时间和内存预算的泛化误差
- 主要目标是调整模型的有效容量以匹配任务的复杂性
- 记最终目标：提升测试集性能

有效容量受限于三个因素：

- 模型的表示容量
- 学习算法成功最小化训练模型代价函数的能力
- 代价函数和训练过程正则化模型的程度

学习率可能是最重要的超参数

11.4

必
运
手
有
学

超参数	容量何时增加	原因	注意事项
隐藏单元数量	增加	增加隐藏单元数量会增加模型的表示能力。	几乎模型每个操作所需的时间和内存代价都会随隐藏单元数量的增加而增加。
学习率	调至最优	不正确的学习速率，不管是太高还是太低都会由于优化失败而导致低有效容量的模型。	
卷积核宽度	增加	增加卷积核宽度会增加模型的参数数量。	较宽的卷积核导致较窄的输出尺寸，除非使用隐式零填充减少此影响，否则会降低模型容量。较宽的卷积核需要更多的内存存储参数，并会增加运行时间，但较窄的输出会降低内存代价。
隐式零填充	增加	在卷积之前隐式添加零能保持较大尺寸的表示。	大多数操作的时间和内存代价会增加。
权重衰减系数	降低	降低权重衰减系数使得模型参数可以自由地变大。	
Dropout 比率	降低	较少地丢弃单元可以更多地让单元彼此“协力”来适应训练集。	

表 11.1: 各种超参数对模型容量的影响。

11.4.2 自动超参数优化算法

理想的学习算法应该是只需要输入一个数据集，就可以输出学习的函数，而不需要手动调整超参数

- 流行算法如逻辑回归和支持向量机
- 部分原因是这类算法只有一到两个超参数需要调整，它们也能表现出不错的性能
- 所需调整的超参数数量较少时，神经网络可以表现出不错的性能

原则上有可能开发出封装学习算法的超参数优化算法

- 自动选择超参数，
- 使用者不需要指定学习算法的超参数
- 超参数优化算法往往有自己的超参数

11.4.3 网格搜索

当有三个或更少的超参数时，常见的超参数搜索方法是网格搜索

- 使用者选择一个较小的有限值集去探索
- 这些超参数笛卡尔乘积得到一组组超参数
- 网格搜索使用每组超参数训练模型
- 挑选验证集误差最小的超参数作为最好的超参数

如何选择搜索集合的范围

- 网格搜索大约会在对数尺度下挑选合适的值
- 通常重复进行网格搜索时，效果会最好
- 超参数优化算法往往有自己的超参数

计算代价会随着超参数数量呈指数级增长

11.4.4 随机搜索

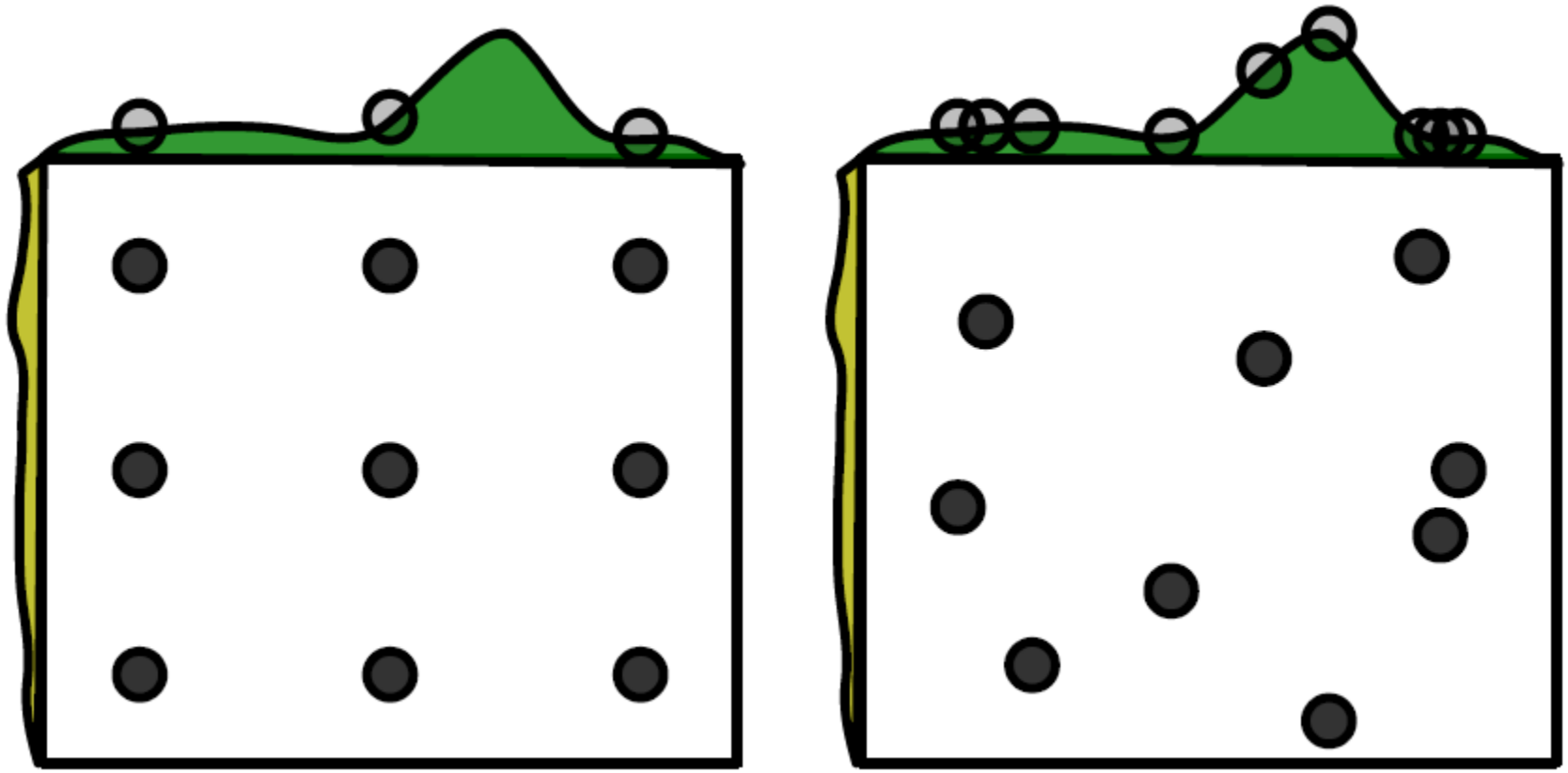
一种替代网格搜索的办法

- 首先，我们为每个超参数定义一个边缘分布
  - 不需要离散化超参数的值
- 会重复运行不同版本的随机搜索，以基于前一次运行的结果改进下一次搜索
- 挑选验证集误差最小的超参数作为最好的超参数

当有几个超参数对性能度量没有显著影响时，随机搜索相比于网格搜索指数级地高效

- 没有浪费的实验

11.4.4 随机搜索



这个图说明了通常只有一个超参数对结果有着重要的影响。在这个例子中，只有水平轴上的超参数对结果有重要的作用。网格搜索将大量的计算浪费在了指数量级的对结果无影响的超参数中，相比之下随机搜索几乎每次测试都测试了对结果有影响的每个超参数的独一无二的值。

11.4.5 基于模型的超参数优化

超参数搜索问题可以转化为一个优化问题

- 决策变量是超参数
- 优化的代价是超参数训练出来的模型在验证集上的误差
- 计算验证集上可导误差函数关于超参数的梯度

这个梯度是不可用的

- 因为其高额的计算代价和存储成本
- 验证集误差在超参数上本质上不可导

弥补梯度的缺失

- 对验证集误差建模
- 通过优化该模型来提出新的超参数猜想
- 使用贝叶斯回归模型来估计每个超参数的验证集误差期望和该期望的不确定性

缺点

- 在它们能够从实验中提取任何信息之前，它们需要运行完整的训练实验

11.5 调试策略

机器学习系统很难调试

- 在大多数情况下，我们不能提前知道算法的行为
- 大部分机器学习模型有多个自适应的部分

大部分神经网络的调试策略都是解决这两个难题的一个或两个。

- 可以设计一种足够简单的情况，能够提前得到正确结果，判断模型预测是否与之相符
- 也可以设计一个测试，独立检查神经网络实现的各个部分

11.5 调试策略

一些重要的调试检测

- 可视化计算中模型的行为
  - 有助于确定模型达到的量化性能数据是否看上去合理
- 可视化最严重的错误
  - 通常可以发现该数据预处理或者标记方式的问题
- 根据训练和测试误差检测软件
  - 训练和测试误差能够提供一些线索
  - 拟合极小的数据集，确定问题是真正的欠拟合，还是软件错误
  - 比较反向传播导数和数值导数
- 监控激活函数值和梯度的直方图
  - 隐藏单元的预激活值可以告诉我们该单元是否饱和，或者它们饱和的频率如何

11.6 示例：多位数字识别

街景转录系统

- 首先这个过程要采集数据
  - 收集原始数据，进行标注
- 数据处理工作
  - 使用其他机器学习技术探测房屋号码
- 度量的选择
  - 保证高准确率，所以覆盖成了这个项目优化的主要性能度量
- 基准系统是带有整流线性单元的卷积网络
- 反复细化这些基准，并测试每个变化是否都有改进
- 理论问题解决之后
  - 综合训练集和测试集性能，以确定问题是否是欠拟合或过拟合
- 调试策略：可视化模型最糟糕的错误
  - 可视化不正确而模型给了最高置信度的训练集转录结果
- 调整超参数