



北京大学

AI 系统实践 - 调参与ML0ps



内容部分参考: Stanford CS 329P, Practical Machine Learning

主讲人: 王乐业 2023年春

本讲主写人 王乐业



AI系统实践流水线



预测房屋售价

问题形式化



获取历史房屋信息
及售价



数据获取



数据清理、分析
及特征提取



数据预处理



训练线性模型



建模与调参



在线部署模型预
测新房屋的售价



系统部署



持续维护

获得新数据、更
新训练模型、预
测准确率监控



AI系统实践 - 调参



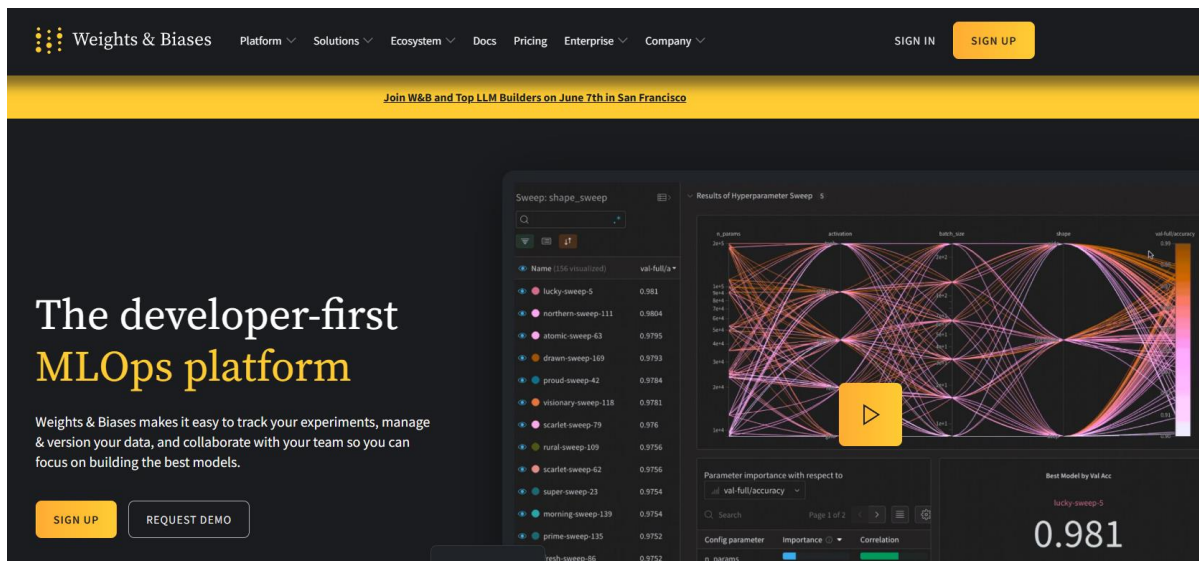
超参数调参

- 从默认设置开始：机器学习工具包默认设置、论文设置……
- 一次只变更一个超参数
 - 如果同时更改多个超参数,很难判断性能变化是由哪一个超参数导致的
- 对每个超参数重复并系统地尝试3-5个不同的值
 - 这样可以让你了解超参数值与性能之间的关系,找到最优值的范围
- 记录每个尝试的超参数设置和相应的性能指标
 - 搜索结束时回顾整个超参数调优过程,理解哪些值起作用以及它们的交互作用
- 有时会出现鸡肋,即超参数的值对性能几乎没有影响
 - 固定这个超参数,继续调整其他的超参数
- 探索较广范围超参数值但性能并未明显提高
 - 可能需要对模型架构进行改进



结果记录和复现

- 最简单的方法是将日志保存为文本,并将关键指标放入Excel
 - 存在更好的选项,例如tenesorboard和weights & bias
- 复现相关因素
 - 环境(硬件和库)
 - 代码
 - 随机性(种子)



自动超参数调参

- 计算成本呈指数下降,而人力成本不断上升
- 典型机器学习任务的每次训练成本:
 - 例如,100万用户日志,10000张图像
- 每天的数据科学家成本>\$500
- 自动调参结果通常超过90%的数据科学家

	Time	Cost on cloud
Trees	10min on CPU	\$0.4
Neural networks	1hour on GPU	\$5

- 自动化机器学习应用于解决实际问题的每一步
 - 数据清理、特征提取、模型选择...
- **超参数优化**(HPO, Hyper Parameter Optimization)
 - 通过搜索算法找到一组好的超参数
- **神经网络架构搜索**(NAS, Neural Architecture Search)
 - 构建一个好的神经网络模型

THE DATA SCIENTIST'S #1 EXCUSE FOR
LEGITIMATELY SLACKING OFF:
"THE AUTOML TOOL IS OPTIMIZING MY MODELS!"



- 搜索空间随超参数个数指数级增加

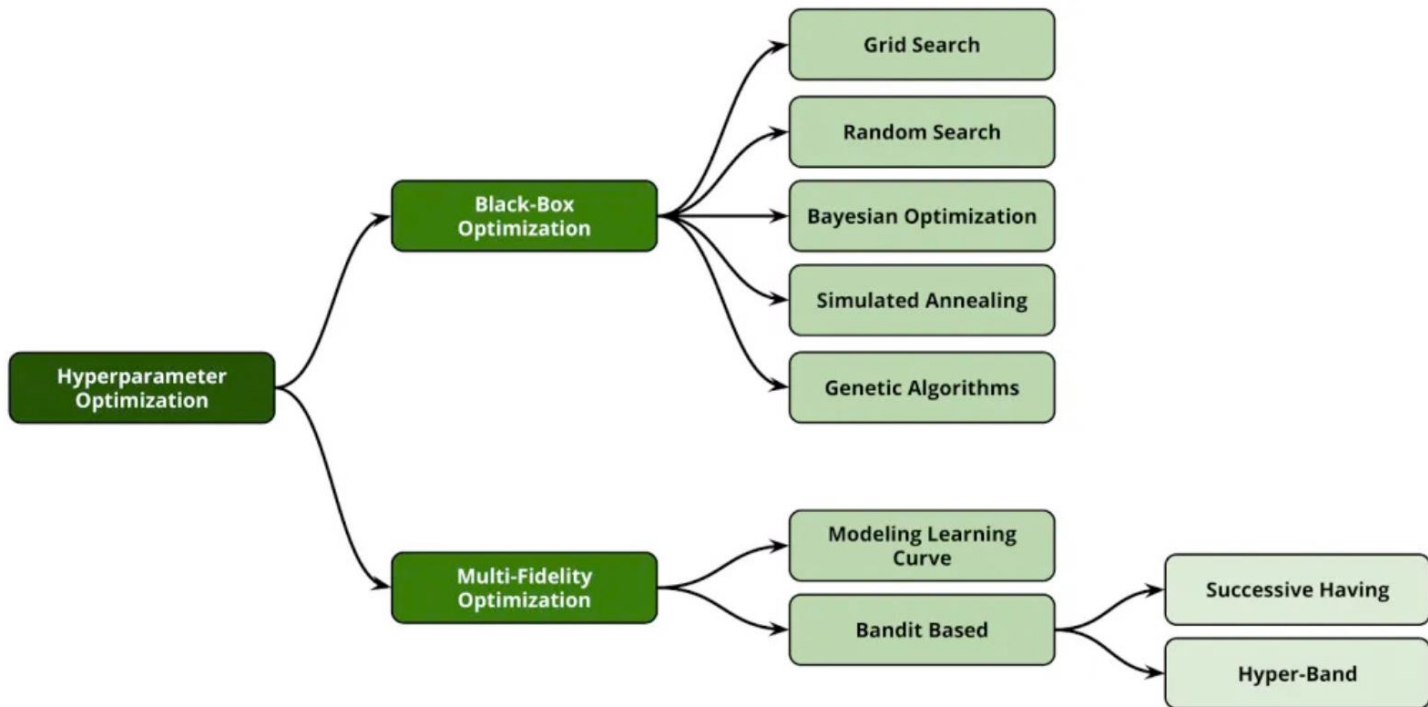
Hyper-Parameter	Range	Distribution
model(backbone)	[mobilenetv2_0.25, mobilenetv3_small, mobilenetv3_large, resnet18_v1b, resnet34_v1b, resnet50_v1b, resnet101_v1b, vgg16_bn, se_resnext50_32x4d, resnest50, resnest200]	categorical
learning rate *	[1e-6, 1e-1]	log-uniform
batch size *	[8, 16, 32, 64, 128, 256, 512]	categorical
momentum **	[0.85, 0.95]	uniform
weight decay **	[1e-6, 1e-2]	log-uniform
detector	[faster-rcnn, ssd, yolo-v3, center-net]	categorical



- 黑盒 Black-box
 - 在HPO中将训练任务视为黑盒
 - 完成每次试验的训练过程
- 多粒度 Multi-Fidelity: 显著加速HPO,但得到的性能评估是相对的
 - 在子数据集上训练
 - 使用训练数据子集快速训练。无法得到精确性能,但可以比较不同配置之间的相对性能。
 - 减小模型大小
 - 使用更小的模型(例如少层或少通道的网络)进行快速训练
 - 早停
 - 如果某个配置的性能明显较差,可以提前停止其训练,以节省时间
- HPO算法通常**先使用多粒度方法**进行粗略搜索,找到较好的配置,**然后使用黑盒方法**训练这些配置并获得精确性能,最后选择最佳配置。



HPO 算法分类



```
for config in search_space:  
    train_and_eval(config)  
return best_result
```

- 流程

- 1. 遍历超参数搜索空间中的每个网格点,即每个超参数值组合。
- 2. 对每个超参数组合进行评估,获得其性能。
- 3. 选择性能最佳的超参数组合。

- 保证找到全局最优解

- 维数灾难



随机搜索 Random search

```
for _ in range(n):
    config = random_select(search_space)
    train_and_eval(config)
return best_result
```

- 实际中非常好用
- 比网格搜索更加高效
 - 具有理论分析
 - 绝大部分超参数其实没有太大影响
- 通常可以从随机搜索开始

Journal of Machine Learning Research 13 (2012) 281-305

Submitted 3/11; Revised 9/11; Published 2/12

Random Search for Hyper-Parameter Optimization

James Bergstra

Yoshua Bengio

Département d'Informatique et de recherche opérationnelle

Université de Montréal

Montréal, QC, H3C 3J7, Canada

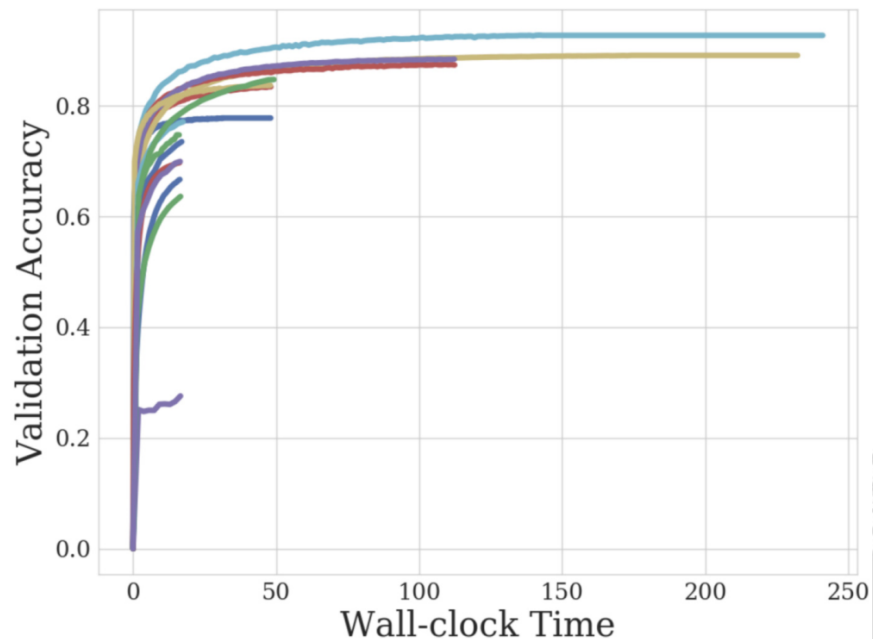
JAMES.BERGSTRA@UMONTREAL.CA

YOSHUA.BENGIO@UMONTREAL.CA

逐次减半 Successive Halving

1. 从搜索空间中随机选择一定数量(例如 n)个超参数配置。
2. 对这 n 个配置同时进行训练,训练时间为 t 个epoch。
3. 比较这 n 个配置的性能,保留性能最优的一半($n/2$)个配置,丢弃其余配置。
4. $t = 2t$, 对保留的配置继续训练 t 个epoch。
5. 重复步骤3和4,每次减少一半的配置,直到剩下一个配置。
6. 最终选择的配置即为HPO的结果。

难点: 很难确定初始的 n



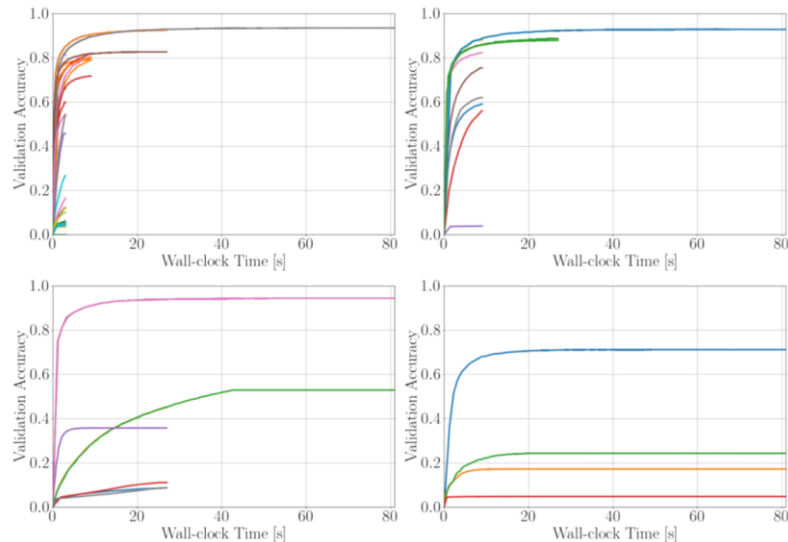
Hyper-Band

- 结合了“逐次减半”和并行化的思想

Algorithm 1: HYPERBAND algorithm for hyperparameter optimization.

```

input      :  $R, \eta$  (default  $\eta = 3$ )
initialization:  $s_{\max} = \lfloor \log_{\eta}(R) \rfloor, B = (s_{\max} + 1)R$ 
1 for  $s \in \{s_{\max}, s_{\max} - 1, \dots, 0\}$  do
2    $n = \lceil \frac{B}{R} \frac{\eta^s}{(s+1)} \rceil, \quad r = R\eta^{-s}$ 
   // begin SUCCESSIVEHALVING with  $(n, r)$  inner loop
3    $T = \text{get\_hyperparameter\_configuration}(n)$ 
4   for  $i \in \{0, \dots, s\}$  do
5      $n_i = \lfloor n\eta^{-i} \rfloor$ 
6      $r_i = r\eta^i$ 
7      $L = \{\text{run\_then\_return\_val\_loss}(t, r_i) : t \in T\}$ 
8      $T = \text{top\_k}(T, L, \lfloor n_i/\eta \rfloor)$ 
9   end
10 end
11 return Configuration with the smallest intermediate loss seen so far.
```



	$s = 4$		$s = 3$		$s = 2$		$s = 1$		$s = 0$	
i	n_i	r_i	n_i	r_i	n_i	r_i	n_i	r_i	n_i	r_i
0	81	1	27	3	9	9	6	27	5	81
1	27	3	9	9	3	27	2	81		
2	9	9	3	27	1	81				
3	3	27	1	81						
4	1	81								

Table 1: The values of n_i and r_i for the brackets of HYPERBAND corresponding to various values of s , when $R = 81$ and $\eta = 3$.

- 神经网络有不同类型的超参数:
 - 拓扑结构:resnet式的,mobilenet式的层数
 - 单独层:卷积层中的kernel_size,通道数

- NAS自动设计神经网络
 - 如何指定NN的搜索空间
 - 如何探索搜索空间
 - 性能估计

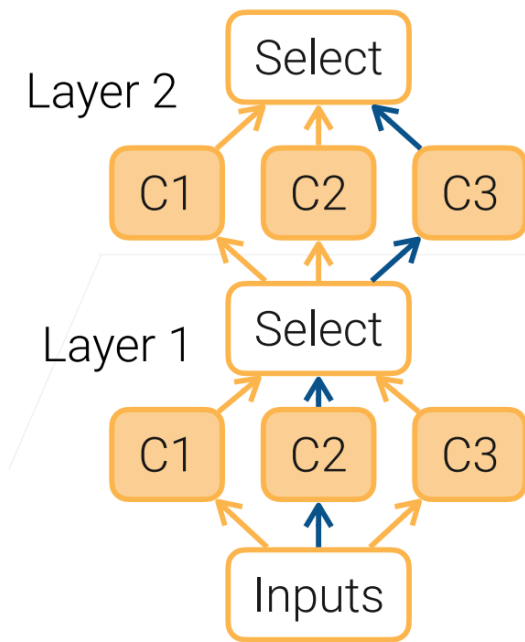
层/运算	超参数
卷积	卷积核数量 卷积核通道数 卷积核宽度 卷积核高度 水平方向步长 垂直方向步长
池化	池化核高度 池化核宽度 水平方向步长 垂直方向步长
全连接	神经元数量
激活	激活函数类型 各种激活函数的参数
相加 (add)	无
拼接 (concat)	无

- 强化学习：速度慢
- One-shot方法
 - 结合架构学习和模型参数学习
 - 构建和训练一个模型展示各种各样的架构
 - 评估候选架构
 - 只关心候选项的排名
 - 使用代理指标:几个epoch后的精度
 - 从头开始重新训练最有希望的候选项



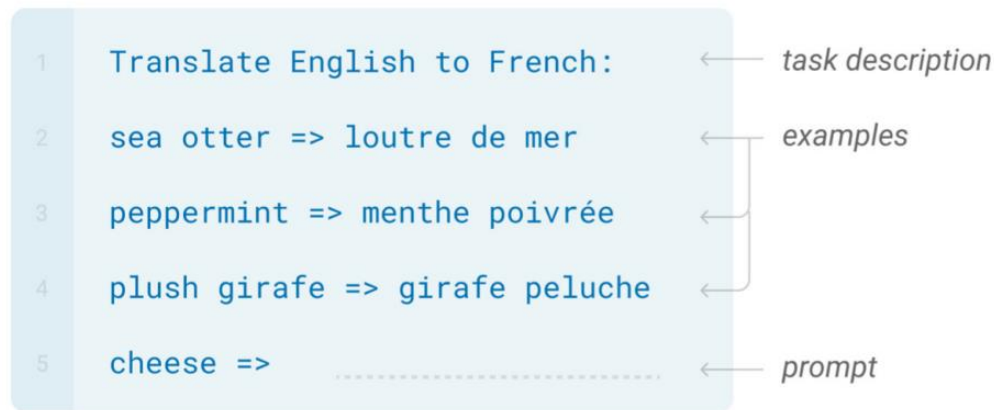
1. 为每层定义多个候选运算(如卷积、池化等)。
2. 每个候选运算的输出通过softmax获得一个权重,这些权重之和为1。
3. 每层的输入是所有候选运算输出的加权和,加权系数为各自的softmax权重。
4. 网络的前向过程实际上是学习这些softmax权重,优化模型参数的同时优化权重。
5. 权重最大的候选运算被选为该层的最终运算。
6. 最终选择的每层运算构成NAS的结果架构。

效率对比强化学习大幅提升!



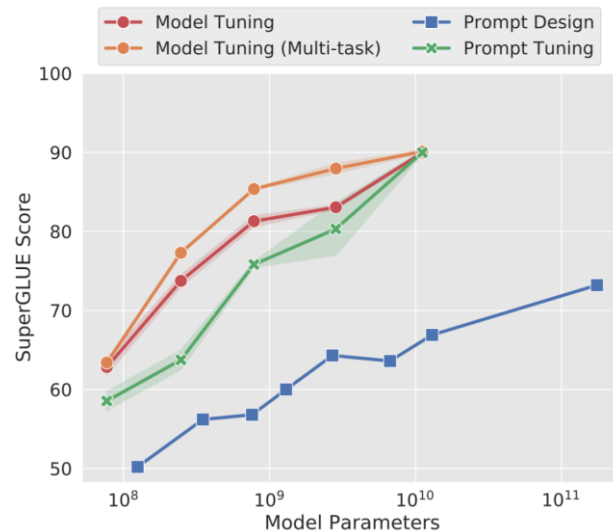
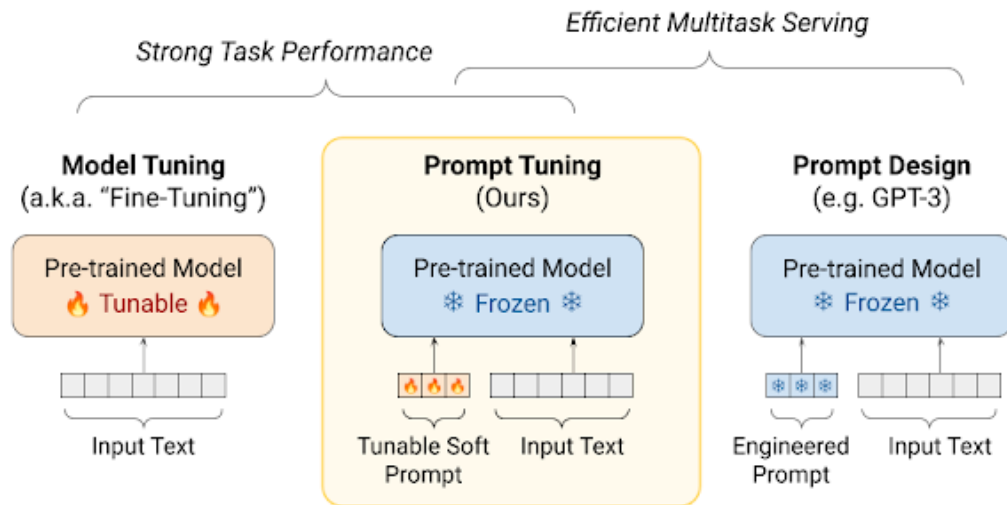
大语言模型的提示 Prompt

- 通用语言模型具有令人印象深刻的文本生成能力
 - 零样本(zero-shot)/少样本(few-shot)学习能力
- 语言模型也理解任务说明
- 提示: 提供带有几个(~10)示例的任务描述和提示



Prompt Tuning vs. Fine-Tuning

• 基于提示的微调



*"As scale increases, prompt tuning matches model tuning, **despite tuning 25,000 times fewer parameters.**"*

<https://ai.googleblog.com/2022/02/guiding-frozen-language-models-with.html>

AI系统实践流水线



预测房屋售价

问题形式化



获取历史房屋信息
及售价



数据获取

数据清理、分析
及特征提取



数据预处理

训练线性模型



建模与调参

在线部署模型预
测新房屋的售价



系统部署



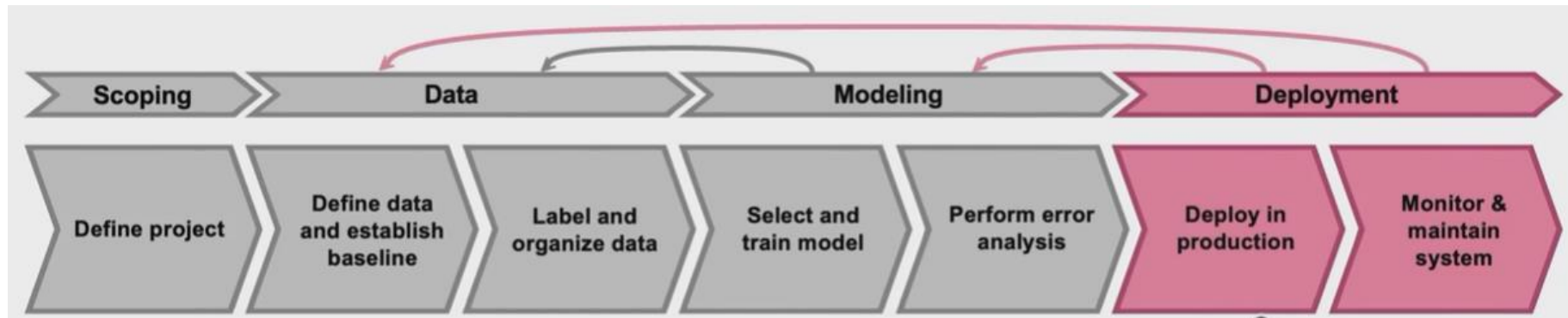
持续维护

获得新数据、更
新训练模型、预
测准确率监控

AI系统实践 – MLOps



MLOps: Machine Learning Operations

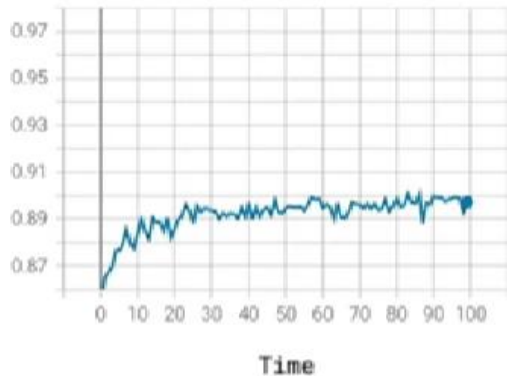


持续监控系统

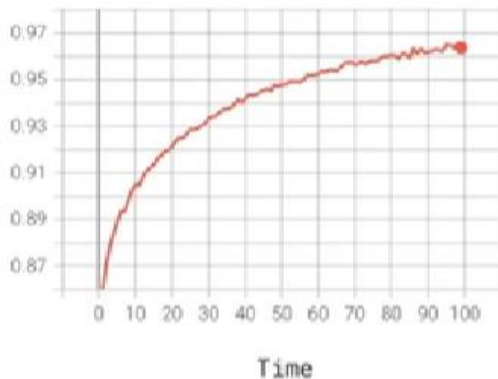
参考：生产中的机器学习(MLOps)-DeepLearning.ai

项目监控

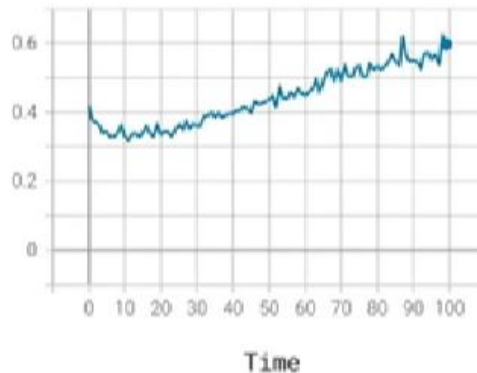
Server load



Fraction of non-null outputs



Fraction of missing input values

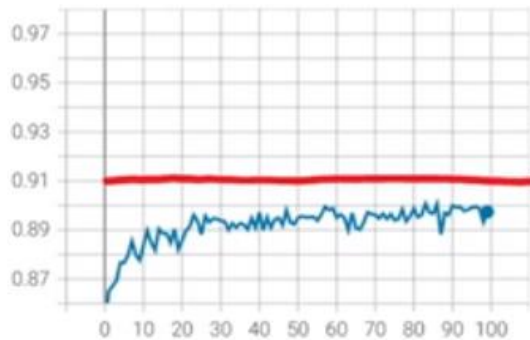


- 头脑风暴
 - 找到可能出问题的地方
 - 找出可以反应这些问题的指标并进行监控
- 开始可以多选择一些监控指标，再慢慢删掉没有用的那些



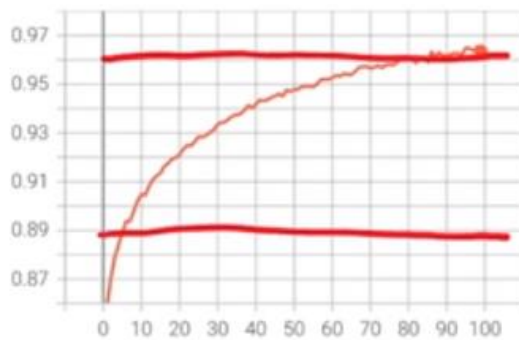
项目监控

Server load



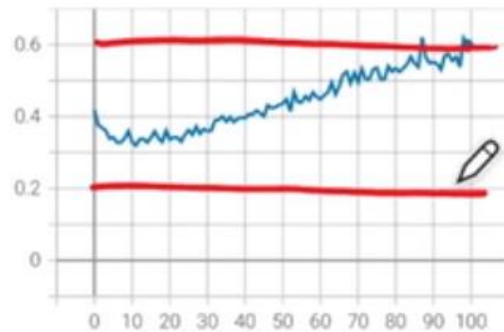
Time

Fraction of non-null outputs



Time

Fraction of missing input values



Time

- 设置阈值处罚警报
- 随着时间阈值需要进行调整



常见的监控指标

Software metrics:

Memory, compute, latency, throughput, server load

Input metrics:



Avg input length

Avg input volume

Num missing values

Avg image brightness

Output metrics:

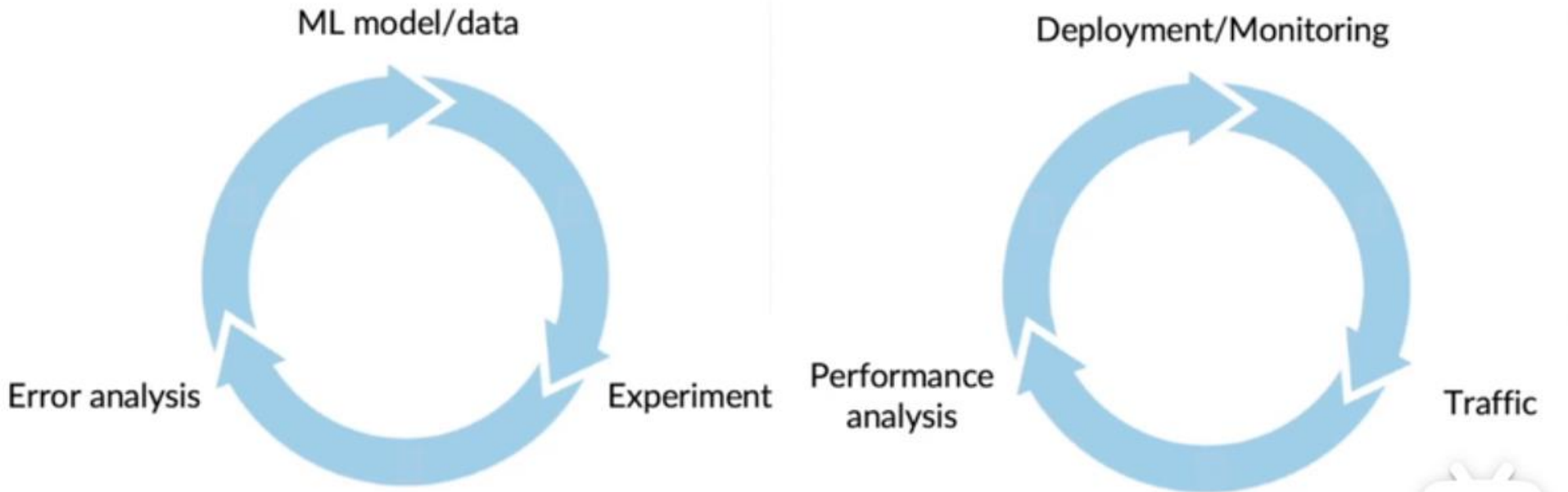
times return " " (null)

times user redoes search

times user switches to typing



部署也是一个迭代的过程





Speech recognition example:

Type	Accuracy
Clear Speech	94%
Car Noise	89%
People Noise	87%
Low Bandwidth	70%

应该着重去优化哪个场景？








建立基线模型

- 获取人类完成任务时的性能，作为模型对比



Speech recognition example:

Type	Accuracy	Human level performance
Clear Speech	94% 	95%
Car Noise	89% 	93%
People Noise	87% 	89%
 Low Bandwidth	<u>70%</u> 	70%

10/0

4/0

2/0

~0/0



基线模型主要方法

- 人类水平性能 Human Level Performance
- 论文SOTA方法/开源算法
- 一个快速实现方法
- 老系统的性能



数据比例

Type	Accuracy	Human level performance	Gap to HLP	% of data
<u>Clean Speech</u>	<u>94%</u>	<u>95%</u>	1%	60% → 0.6%
Car Noise	89%	93%	<u>4%</u>	4% → 0.16%
People Noise	87%	89%	2%	30% → 0.6%
Low Bandwidth	70%	70%	0%	6% → ~0%



- 根据以下因素决定工作的最重要类别:
- 改进空间有多大
 - 对于误分类较多或准确率较低的类别,改进空间最大,应优先考虑
- 该类别出现的频率有多高
 - 频率高的类别可以最大限度地提高总体准确率,应作为重点类别
- 提高该类别准确率的难易度有多大
 - 相对较易提高准确率的类别应先进行改进,以获得快速收益
- 提高该类别的重要性有多大
 - 对最终业务或任务影响更大的类别,其准确率更加重要,应优先进行改进



对于优先类别,可以采取以下措施:

1. 收集更多数据

- 获取更多的训练样本可以明显提高模型在该类别上的准确率

2. 使用数据增强获得更多数据

- 通过翻转、旋转、缩放、噪声添加等方法从原有数据中增强出新的样本

3. 提高标签准确率/数据质量

- 检测并纠正数据集中的错误样本和标签可以避免对模型的误导,提高其学习效果



谢 谢

