Hyper-Parameter Optimization: A Review of Algorithms and Applications

作者：Tong Yu、Hong Zhu（浪潮）

总概

神经网络的设计和训练是一个挑战，为了降低用户的门槛，自动hyper-parameter optimization（HPO）成为一个流行的主题。这篇文章首先介绍了关于training和structrue的关键超参，它们的重要性与如何确定取值范围。然后讨论主要的优化算法的适用性，包括准确度和效率（针对深度网络），并对比主要的HPO工具。文章总结了在网络深层的时候HPO面临的问题，优化算法的比较，如何做高效的model evaluation。

Key parameter

- learning rate（LR）scheduler
  - 固定的LR
    - 最简单
    - 学习率的初值对于后面的step而言过大
  - Linear LR decay
    - $lr = \frac{lr_0}{1+kt}$
  - Exponential decay
    - $lr = \mathrm{lr}_0 \cdot \exp(-kt)$
    - 相比Linear，在初始时下降更快，在后期下降缓
  - Cyclic schedule
  - layer-wise adaptive rate scaling (LARS)
    - 每层有自己的local LR
- optimizer
  - Mini-batch SGD
    - Batch size
  - SGD with momentum
    - momentum β
  - RMSprop
  - Adam

        - $\beta 1, \beta 2, \epsilon$
- Model Design-Related
    - hidden layers 数
        - 决定特征复杂度、 receptive field
    - hidden neurons数
        - 训练时间
        - 导致underfit、overfit
    - regularization
        - L1, L2及其weight
    - data augmentation
        - image transformation, cropping, flipping, color adjustment, and image rotation
    - Dropout
        - dropout rate
    - activation dunction
        - tanh
        - ReLU（及其变种）
        - Maxout
        - Swish
        - Sigmoid
        - softmax

搜索算法（for sampling）

方法	简介
Grid Search	· 尝试参数集合的所有组合 · 适合有足够经验给出小的搜索空间的情况 · 不超过3个需要同时调整的超参 · 易于并行 · 数学形式简单
Random Search	· 每个超参以独立特定分布随机选择 · 每个超参可以设置不同的budget · 相比grid search更易找到更好的超参 · 原理由Monte Carlo保证 · 仍然计算密集

	· 作为HPO的baseline · 易于结合early stopping策略，并取得不错的效果和效率
Baysian Optimize（BO）	· Sequential model-based / guided search方法 · 相比grid search和random search更加高效（更少的尝试，获得更好的效果） · 标准版本不适用于integers, categorical values, conditional search space · reliable、 convincing
Tree Parzen Estimator（TPE）	· 一个树结构，处理条件搜索空间 · 相比原始的Baysian方法能处理更多超参数据类型 · 不能model param-interactions

现实场景中，尤其是在深度网络的应用里，完整训练模型代价高，考虑在训练过程中引入判断，是否有必要继续训练：early-stopping policy

Trial Scheduler（for early stopping）

方法	简介
Median stopping	· 每个training step维护一个关于所有trials的average metrics，如果某个trial当前step最好的metric低于 平均水平，则停止该trial · model-free 适用于广泛的performance curve 这个策略没有超参需要用户去调 · 被集成进Google Vizier、Tune、NNI
curve fitting	· 利用一个单独模型去预测最终的obj value · 预测模型也需要额外训练，带来bias和计算量 · 被集成进Google Vizier and NNI
Successive Halving（SHA）	· 设初始budget, 评估所有trials, 把表现差的后50%丢弃并double budgets, 持续评估丢弃直到只剩一个trial · budget和trials number之间trade off · computation efficient · easy understand
HyperBand（HB）	· 每次丢弃最差的后 $\frac{\eta-1}{\eta}$, 并且budget乘 η · 是SHA的扩展 · easy to deploy in parallel · adaptive allocation of resources · 被集成进Tune and NNI

Population-Based Training（PBT）	· 由DeepMind提出，算法类似于遗传算法，组合了并行搜索和序列优化。主要两个过程**遗传**和**变异**，对应Exploitation 和 exploration · 比以上算法都更高效 · 用户没有必要在训练前确定使用什么LR schedule（训练中改参数） · 缺乏理论基础 · 难以掌握进化和突变的trade-off · 超参的变化使得计算图复杂(训练中改参数)

对比

	Advantage	**Disadvantage**	**Applicability for DNN**
Grid Search	- Simple - Parallelism	- curse of dimensionality	- Applicable if only a few HPs to tune
Random search	- Parallelism - Easy to combine with early stopping methods	- Low efficiency - Cannot promise an optimum	- Convenient for early stage
Bayesian optimization	- Reliable and promising - Foundation of many other algorithms	- Difficult for parallelism - Conceptually complex	- Default algorithm for tools - Variants of BO could be more applicable (TPE)
Multi-bandit methods	- Conceptually simple - Computationally efficient	- Balance between budget and number of trials	- Could be a default choice - Implemented by open-sourced libraries.
PBT methods	- Combine HPO and model training - Parallelism	- Constant changes to computation graph - Not extendable to advanced evolution	- For computationally expensive models

Toolkits

工具	**简介**
HyperOpt	· 基于Gaussian processes 和 regression trees（TPE） · 实现异步并行
Xcessive	· beginner-friendly（GUI） · 只支持Baysian方法的自动搜索，不便于神经网络的高效搜索
Scikit-Optimize	· Bayesian methods, random search, random forest和一些其他可靠的优化算法 · 不便神经网络的高效搜索

Google Vizier	· 闭源、收费 · Google cloud支持 · 容易使用（用户只需提交一个config文件、有GUI，监控实验状态，方便调试） · 具备扩展性（有经验的用户可以定制自己的算法） · 巨大的error-correcting能力 · transfer learning（拥有数据中心，加速） · 提供benchmark（performance-over-time metrics） · 支持多种数据类型（float, integer, discrete, and categori cal parameters）
Advisor	· 一个Google Vizier的开源版本 · 有交互式GUI · 没有Google cloud支持 · 受限于计算资源
Amazon SageMaker	· 闭源、收费 · AWS支持 · 易于使用 · Jupyter 设置和可视化结果 · 支持early stop and warm start · 用户能扩展使用自己的算法 · transfer learning（拥有数据中心，加速） · 搜索算法局限于random search 和 BO 收费
Neural Network Intelligence (NNI) 微软	· 开源 、免费 · 支持平台环境丰富（local machines, remote servers, and Dockers） · 相比Google Vizier 和 SageMaker实现了更多算法，几乎内建所有SOTA算法（除了 PBT） · 为新算法提供扩展接口（只需要继承一个base class） · 能部署到众多平台（local machines, remote servers, 和 Kubernetes-based Dockers） · 兼容大多数流行的deep learning框架（PyTorch, Keras, TensorFlow, MXNet, Scikit-learn，XGBoost） · 通过Web UI来可视化监控中间结果和过程（兼容TensorBoard 和 TensorBoardX） · 对beginners不友好（相比其他工具，封装得没那么死，导致需要一定模型训练经验） · Web UI功能不够完善（相比Google Vizier，只显示状态和结果，没有交互功能） · debug信息不够直接（log文件细节不够） · 缺少population-based算法（PBT是唯一能联合超参调整与模型训练的算法）
Ray.Tune by Berkeley	· 开源、免费

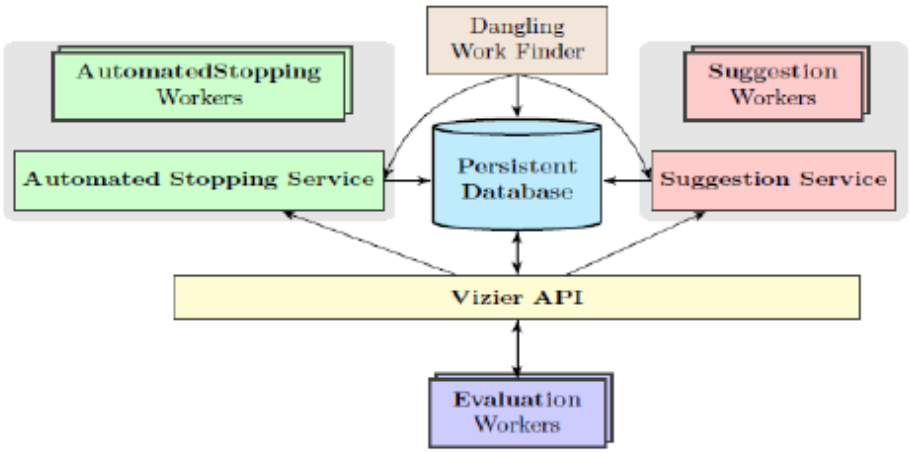
RISELab	· 和NNI有很多相似的地方，不同点主要在多了PBT支持、通过import方式实现大部分搜索算法 · 扩展性好 · 目前还不适用于Kubernetes-based Dockers · 没有交互GUI

对比

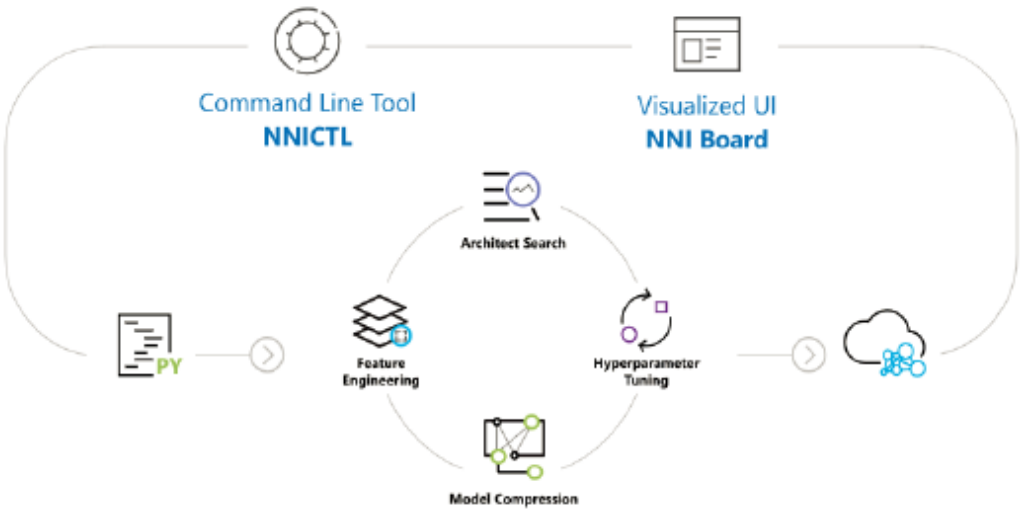
	Vizier & Sagemaker	**NNI & Tune**
Ease to use	Almost no configuration by users - Friendly to starters with simple workflow - Straightforward GUI for both task management and result visualization - Error correcting capability	- Need minimal configuration by users - Preliminary knowledge on model training - Visualization with Tensorboard
Scalability	- Very high with the support of cloud service	- Usually deployed on several units
State-of-the-art	- Support classical search algorithms - Some early stopping methods - Transfer learning	- Support almost all SOTA algorithms - Not support transfer or meta learning
Availability	- Need to pay for the convenience	- Open sourced and free
Flexibility	- Extendable search algorithm - Closed sourced in infrastructure	- Able to modify everything - High flexibility for experienced users

	NNI	**Tune**
Application	Toolkit for AutoML , HPO and model compression	Only for HPO
Deployment	Local, remote and Dockers	Local and remote
Algorithm	Support almost all SOTA algorithms except PBT	Support almost all SOTA algorithms including PBT
Implementation	By rewriting the algorithm in format of NNI	By importing existing libraries with the interface if Tune
Start an Experiment	Import NNI+ *config. file* + search space file	Define a trainable class + *tune.run* function
Modification on original code	Less. By adding individual files	More. By build a trainable class
Visualization	Web UI and TensorBoard (TensorBoardX)	TensorBoard (TensorBoardX)

Google Vizier



NNI



挑战

- 超参的数量多，复杂的超参数空间
- 对某一特定的结构和超参，evaluation代价昂贵
- 并行化
- 计算效率

Evaluation

训练完整模型到收敛再evaluate (最准确但耗资源)

subsample dataset训练（加速训练但是引入noise）

early stop（训练中performance低于一定标准便停止）

transfer learning (param reuse，用相似超参配置的参数来初始化)

meta learning (linear regression, curve fitting，MLP，etc. 预测performance)