控制与决策
Control and Decision

基于种群演化的超参数异步并行搜索

蒋云良， 赵康， 曹军杰， 范婧， 刘勇

引用本文:
蒋云良, 赵康, 曹军杰, 等. 基于种群演化的超参数异步并行搜索[J]. 控制与决策, 2021, 36(8): 1825−1833.

在线阅读 View online: https://doi.org/10.13195/j.kzyjc.2019.1743

基于种群演化的超参数异步并行搜索

蒋云良[1,2†], 赵 康[3], 曹军杰[4], 范 婧[4], 刘 勇[4]

(1. 湖州师范学院 信息工程学院，浙江 湖州 313000；2. 湖州师范学院 浙江省现代农业资源智慧管理与应用研究重点实验室，浙江 湖州 313000；3. 浙江师范大学 数学与计算机科学学院，浙江 金华 321004；4. 浙江大学 智能系统与控制研究所，杭州 310027)

摘 要: 近年来随着深度学习尤其是深度强化学习模型的不断增大,其训练成本即超参数的搜索空间也在不断变大,然而传统超参数搜索算法大部分是基于顺序执行训练,往往需要等待数周甚至数月才有可能找到较优的超参数配置. 为解决深度强化学习超参数搜索时间长和难以找到较优超参数配置问题,提出一种新的超参数搜索算法——基于种群演化的超参数异步并行搜索(PEHS). 算法结合演化算法思想,利用固定资源预算异步并行搜索种群模型及其超参数,从而提高算法性能. 设计实现在Ray并行分布式框架上运行的参数搜索算法,通过实验表明在并行框架上基于种群演化的超参数异步并行搜索的效果优于传统超参数搜索算法,且性能稳定.

关键词: 超参数搜索；种群；演化算法；异步并行；深度学习；并行框架

中图分类号: TP273 **文献标志码:** A

DOI: 10.13195/j.kzyjc.2019.1743 **开放科学(资源服务)标识码(OSID):**

引用格式: 蒋云良,赵康,曹军杰,等. 基于种群演化的超参数异步并行搜索[J]. 控制与决策, 2021, 36(8): 1825-1833.

Asynchronous parallel hyperparameter search with population evolution

JIANG Yun-liang[1,2†], *ZHAO Kang*[3], *CAO Jun-jie*[4], *FAN Jing*[4], *LIU Yong*[4]

(1. School of Information Engineering，Huzhou University，Huzhou 313000，China；2. Zhejiang Province Key Laboratory of Smart Management & Application of Modern Agricultural Resources，Huzhou University，Huzhou 313000，China；3. College of Mathematics and Computer Science，Zhejiang Normal University，Jinhua 321004，China；4. Institute of Cyber Systems and Control，Zhejiang University，Hangzhou 310027，China)

Abstract: In recent years, with the continuous increase of deep learning models, especially deep reinforcement learning models, the training cost, that is, the search space of hyperparameters, has also continuously increased. However, most traditional hyperparameter search algorithms are based on sequential execution of training, which often takes weeks or even months to find a better hyperparameter configuration. In order to solve the problem of the long search time hyperparameters and the difficulty in finding a better hyperparameter of deep reinforcement learning configuration, this paper proposes a new hyper-parameter search algorithm, named asynchronous parallel hyperparameter search with population evolution. This algorithm combines the idea of evolutionary algorithms and uses a fixed resource budget to search the population model and its hyperparameters asynchronously and in parallel, thereby improving the performance of the algorithm. It is realized that a parameter search algorithm can run on the Ray parallel distributed framework. Experiments show that the parametric asynchronous parallel search based on population evolution on the parallel framework is better than the traditional hyperparameter search algorithm, and its performance is stable.

Keywords: hyperparameter search；population；evolutionary algorithm；asynchronous parallelism；deep learning；parallel framework

0 引 言

近些年,随着硬件计算能力特别是大规模分布式并行计算的飞速发展,机器学习领域得到了长足的发展. 在训练数据足够充分的前提下,机器学习算法的超参数配置问题是其取得较好效果的关键. 超参数是在某个机器学习算法运行之前,首先需要选取的参数,例如深度学习算法中控制神经网络学习速度的学习率. 超参数搜索的目的是为某个应用的算法选择一组好的超参数,使此算法性能达到最佳[1]. 在以往机器学习超参数选择问题中,领域内研究者一般都是

收稿日期: 2019-12-13；**修回日期:** 2020-04-23.

基金项目: 国家自然科学基金项目(61771193)；浙江省重点研发计划项目(2020C01097, 2020C02020).

责任编委: 陈家伟.

†通讯作者. E-mail: jyl@zjhu.edu.cn.

基于个人经验对超参数进行人工选择. 随着数据规模指数级上升,虽然大规模计算加速设备的飞速进展使得深度学习特别是深度强化学习算法,在处理海量的图片等数据方面显示出强大的优势,但是在超参数选择问题上仍然是一个未解决的难题.

深度学习主要是通过多层的神经网络训练以使其达到期望的学习效果. 其概念由Hinton等于2006年提出[2],通过组合低层特征形成更加抽象的高层表示属性类别或特征,以发现数据的分布式特征表示. 强化学习,又称再励学习,是机器学习中一种重要的学习方法, 也被认为是属于马尔科夫决策过程(markov decision process, MDP)和动态优化方法的一个独立分支. 强化学习是智能体(Agent)以“试错”的方式进行学习,随着人工智能的发展,强化学习不再局限于动作空间和状态空间很小的离散环境,特别是深度强化学习面临更复杂更接近现实的连续环境. 神经网络在机器学习领域有着显著的进展,已经成为许多深度学习尤其是深度强化学习中非线性问题的一种逼近器. 一个特定神经网络的性能,不但依赖于模型结构,而且训练数据以及模型参数优化细节也非常重要. 模型框架每一部分都是由参数控制,只有通过适当的参数优化,才能充分体现模型框架性能. 在模型优化过程中,随着学习算法模型框架不断增大,超参数搜索过程会变的越来越复杂. 特别在深度学习和强化学习领域,一个较差超参数配置会导致训练结果出现异常现象(如不收敛,梯度爆炸等),而一个较优的超参数配置不仅能省去大量训练时间且可以得到预期结果.

超参数调优有两种常见方法:并行搜索和顺序优化[3]. 并行搜索方法执行多个并行优化过程,每个过程都有不同超参数,其目的是从其中一个优化过程中找到单个最佳输出. 顺序优化方法从早期的训练中获得信息来逐步执行超参数优化,以向后续训练提供经验信息. 顺序优化通常会提供最佳解决方案,但是多次顺序优化训练会耗费巨大的时间资源.

为了解决深度学习和深度强化学习模型中训练时间过长且找到的超参数较差的问题,本文提出一种新的异步并行超参数搜索方法,即基于种群演化的超参数异步并行搜索(PEHS),将并行搜索方法和顺序优化方法进行了结合和扩展. PEHS算法的优点是异步并行计算,可以大大缩减超参数搜索时间,并利用更少的计算资源快速找到较优的超参数配置. 另外,由于演化算法是借鉴大自然中生物的进化操作,它一般包括基因编码、种群初始化、交叉变异算子、经营保留机制等基本操作,具有一定的自组织、自适应、自学习特性,能够不受问题性质限制,有效地处理传统优化算法难以解决的复杂问题. 因此,PEHS算法还具有一定的稳定性和自适应性.

1 相关工作

超参数调优大致可以分为并行搜索和顺序调优两部分. 并行搜索是并行执行多个优化过程,从多个优化过程中选择最优输出超参数. 经典并行搜索有随机搜索[4]、 网格搜索[3]等. 随着大型并行分布式计算集群的兴起,近两年又出现了连续减半算法[4]和Hyperband[5]等并行算法. 顺序优化方法从早期的训练中获得信息来逐步执行超参数优化,向后续训练提供经验信息,最终找到较优超参数配置. 常见的顺序超参数配置有手动调优和贝叶斯优化等. 顺序优化需要大量的时间和先验知识才可能找到较优的超参数配置,且可能会陷入局部最优. 本文提出的基于种群演化的超参数异步并行搜索算法结合了并行搜索和顺序优化的优点,优化了超参数调优陷入局部最优的情况,提高了训练效率.

随机搜索[4]算法是一种经典的超参数优化算法,它的缺点是盲目性和随机性. 但是在训练资源足够多情况下,也可以找到较优超参数配置. 网格搜索[5](又名暴力搜索)是在所有候选的超参数中选择,通过循环遍历所有超参数,从中找到表现最好的超参数配置. 优点是能找到较优的超参数配置,缺点是需要大量时间和硬件资源. Jamieson等[6]提出了改进的随机搜索算法 —— 连续减半(successive halving)算法. 连续减半的思想:统一分配预算(资源)到一组配置(超参数),评估所有配置的性能,抛出较差的一半,重新平均分配预算到剩余配置,重复,直到剩余一个配置保留,即为找到的最佳配置. 该算法以指数的方式将更多的资源分配至更有希望的配置. 另外,此算法需要将配置数n作为算法输入,给定有限预算b(例如,选择超参数配置所需的1 h训练时间),b/n资源平均分配给各个配置. 但是,对于固定的b,不清楚是否应该考虑使用更多的平均训练时间在较短的配置(较大的n),如超参数收敛速度较快;或考虑较少平均训练时间在较长的配置(较小的n),如超参数收敛的速度较慢. Li等[7]为解决连续减半算法的资源配置问题,提出了一种新的超参数优化算法Hyperband,此算法主要是动态地将资源分配给一组随机配置,并使用连续减半算法停止性能不佳的配置. 与SMAC[8]和TPE[9]等贝叶斯算法相比,Hyperband表现出了优越的性能、 良好的灵活性和对高维空间的可扩展

性. 对于训练模型超大参数空间特征,为了更充分地探索,此算法必须通过估计大量的配置和并行来进行优化搜索. 考虑训练模型成本的不断增加,希望模型可以在尽量少的时间内搜索完成更多超参数配置. Li 等[10] 提出了一种异步的Hyperband超参数搜索算法AsyHyperband来解决这些问题,利用并行运算和主动提前停止规则,加快算法速度进而搜索更多的超参数. 在分布式环境中,规模与worker的数量成线性关系,特别在大规模集群服务器上表现出的性能更加优越. 虽然AsyHyperband算法在Hyperband算法的基础上添加了并行搜索,但是在硬件资源较少的情况下,对于大型超参数搜索空间超参数的选择还是具有一定的随机性.本文提出的基于种群演化的超参数异步并行搜索算法不但适用于硬件不足的情况,而且解决了AsyHyperband算法在大型超参数搜索空间超参数选择的随机性.

$$\begin{cases} l(x) = p(y < a|x, D); \\ g(x) = p(y > a|a, D). \end{cases} \tag{1}$$

Bergstra 等[9] 提出了一种新的贝叶斯超参数优化算法(tree parzen estimator, TPE),它使用核密度估计器对输入配置空间上的密度建模,而不是直接用 $p(f|g)$ 对目标函数 f 建模. 选择一个新的候选点 x,它最大化了 $l(x)/g(x)$ 的比率($l(x), g(x)$,见式(1)). 由于核密度估计器的性质,TPE很容易支持混合连续空间和离散空间,模型构建在数据点的数量上是线性的. Gonzalez 等[11] 提出了一种新的启发式贝叶斯超参数优化方法:通过局部惩罚实现批处理贝叶斯优化. 该算法假设目标函数是一个李普希茨连续(Lipschitz continuous)函数,围绕采集函数的一个环用于收集一定大小的批量点,从而最小化不可并行计算工作量. 相对于传统超参数优化算法,该算法对超参数优化具有一定的加速作用. Falkner 等[12] 根据贝叶斯超参数优化算法性能的稳定和Hyperband随机搜索的搜索速度快的优点,将两种超参数优化算法融合,提出顺序优化和并行搜索相结合的BOHB (TPE和Hyperband)超参数优化算法,实现了两种超参数优化算法性能的结合,优化了贝叶斯超参数算法的训练时间长且可能陷入局部最优和Hyperband随机搜索算法随机性问题,表现出强劲的随时性、并行性、鲁棒性及可扩展性. 本文提出的基于种群演化的超参数异步并行搜索算法与BOHB算法的优化方法有所不同,本文在超参数优化上是结合演化算法和种群进化的原则,能充分对超参数空间进行利用,进而减小了进入局部最优的概率.

Golovin 等[13] 提出了一种中值停止规则超参数优化算法,也是一种提前停止的随机搜索优化算法,主要思想是提前停止训练中超参配置性能低于整体性能中间值(可以是中值,中位数,或者分位数等)的超参数配置. 通过提前停止性能差的中间值,可以训练更多的超参数配置,增大配置搜索范围,使找到最优配置的概率更大. 由于该算法是可以并行的,在分布式集群上的性能更优. 本文提出的基于种群演化的超参数异步并行搜索算法结合早期停止规则超参数优化算法,使算法在并行的基础上可以找到性能更优的超参数.

2 基于种群演化的超参数异步并行搜索算法

由于深度学习尤其是深度强化学习模型在超参数选择上具有以下两个问题: 1) 对超参数的敏感和对经验的依赖; 2) 搜索到较优的超参数需要较大的计算资源和时间复杂度较高等问题. 本文提出一种针对深度学习的超参数搜索算法,即基于种群演化的超参数异步并行搜索(PEHS)算法. 该算法继承演化算法[14] 的思想,利用异步并行计算优化过程中种群与个体之间信息共享和提前停止规则,并允许种群成员根据性能异步在线传播/传输参数和超参数. 此外,与传统超参数搜索方法不同的是,PEHS能够对超参数进行在线的筛选,特别是在动态优化学习(深度强化学习和在线深度学习)中尤为重要. 该算法不仅可以平行计算,而且在训练的过程中可以在线地进行自动调整超参数,使整个训练过程具有一定的自适应能力.

2.1 PEHS算法构建过程

机器学习算法通常是优化模型 f 的参数 θ 以最大化给定目标函数 Q^* (例如分类、重建或预测). 可训练参数 θ 一般是通过优化程序进行更新,例如随机梯度下降. 然而,深度学习和强化学习关注的重点是实际性能指标 Q,与 Q^* 不同[1](Q 可能是验证集的准确性,或者强化学习中的环境奖励). PEHS的主要目的是提供一种实际指标上同时优化参数 θ 和超参数 h 的方法.

首先定义一个评估函数 eval( ),使用模型的当前状态来评估目标函数. 为了简单起见,忽略除了参数 θ 的所有影响训练的因素,只将评估函数定义为可训练参数 θ 的函数. 评估函数不需要是可微的,也不需要与优化步骤中用于计算迭代更新的函数相同(它们可能是相关的). 找到最大化目标函数最佳参数集的过程为

$$\theta' = \arg\min_{\theta\in\Phi} \text{eval}(\theta), \tag{2}$$

其中Φ是可训练参数θ的集合.

当模型是一个神经网络时,通常以迭代(函数)的方式优化参数θ,例如在目标函数上使用随机梯度下降法. 通过迭代优化,更新模型参数,其会受到其自身超参数$h \in H$(H参数搜索空间)的约束. 参数更新迭代步骤为

$$\theta = \text{step}(\theta|h). \tag{3}$$

通过将评估函数与迭代函数连接以形成一系列更新,理想地收敛到最优解,即

$$\theta' = \text{opt}(\theta|h) = \text{opt}(\theta|(h_t)_{t=1}^T) = \text{step}(\text{step}(\dots\text{step}(\theta|h_1)\dots|h_T-1)|h_T). \tag{4}$$

因为在每步迭代训练中得到参数θ'的计算成本代价很高,步骤数量T比较大,所以优化θ的过程可能需要几天、几周甚至几个月. 另外,超参数优化算法对超参数$h = (h_t)_{t=1}^T$的选择非常敏感,选择错误超参数可能导致错误的解决方案,甚至导致训练失败. 选择正确超参数需要对h具有很强的先验知识才可能被发现(通常需要不同的h进行多个优化训练过程). 然而h与迭代步骤的依赖性,可能值的个数随时间呈指数增长. 通常的做法是: 1) 让所有参数相等(例如整个训练保持恒定的学习速度); 2) 预先制定一个简单的计划(如annealing的学习速度). 这两种情况都需要搜索多个超参数h,即

$$\theta' = \text{opt}(\theta|h')$$

$$h' = \arg\min_{h\in H^T} \text{eval}(\text{opt}(\theta|h)). \tag{5}$$

式(5)对应种群中超参数的一次搜索选择. 考虑在种群P中训练N个的模型$\{\theta'^N_{i=1}\}$,通过不同的超参数$\{h^i\}_{i=1}^N$进行优化,目标是找到整个种群P中较优模型. 为了找到较优的模型h,PEHS算法对群体中每一个成员(即每个训练样本)使用两种独立的调用方法: 1) 利用函数(exploit),考虑到整个群体的表现(训练性能),可以决定成员是否应该放弃当前的解决方案(参数和超参数),而将注意力集中在更有前途的成员上; 2) 探索函数(explore),考虑到当前的解决方案,提出新的解决方案以更好地探索解决方案空间(参数空间).

2.2 PEHS算法实现

对群体中每个成员进行异步并行训练,通过调用迭代函数来更新成员的权重θ,评估函数来度量成员的当前性能. 当群体中的一个成员被视为准备就绪(通过使用最少的步骤进行优化或达到某个性能阈值),其权重和超参数将通过利用函数和探索函数进行更新. 例如,利用函数可以将当前权重替换为在种群中同步长且具有最高记录性能的模型参数,探索函数可以随机地用噪声干扰超参数. 在利用和探索之后,停止性能差的成员,重新生成一个新的成员,其余成员迭代训练像以前一样继续进行演化. 通过局部迭代训练和运用种群进行利用和探索的循环,直到模型收敛.

算法1 基于种群演化的超参数异步并行搜索(PEHS)

初始化: 超参数h,当前运行时间t,参数θ,种群P(h集合H),最大运行时间$\max_t$,Ω_{mt}为t时刻Ω中前比例为m的子集,选择算子α(包含两个值A, B),重建成员数量nt,性能p,性能优的h和p集合序列Ω,评估函数eval( ).

开始进行种群训练(异步并行运行)

for i in种群 P do

while $t < \max\_t$ do

$\theta = \text{step}(\theta|h_i) \quad H_i \in H$

if Ω为空且长度小于2(以下是探索和利用)

将 h_i和 p_i添加到Ω

else

if $p_{it} < \Omega_{mt}$且 $i \leqslant nt$

$\alpha = \text{random}(A, B)$

if $\alpha == A$

从Ω_{mt}随机选取h'_i $(h'_i \in \Omega)$

$h'_i = h'_i + \beta$,其中β为扰动因子

else

从种群P中随机选取h'_i

end if

终止成员h'_i,从种群P中移除

新建新成员h'_i,并添加到种群P中

end if

把h'_i和p_i添加到Ω

end if

更新种群 P中成员$(h, p, \theta, t+1)$

end while

end for

返回种群P中性能p最优的超参数h

end 训练

算法1是基于种群演化的超参数异步并行搜索算法的伪代码形式,其算法流程如图1所示. 具体步骤如下:

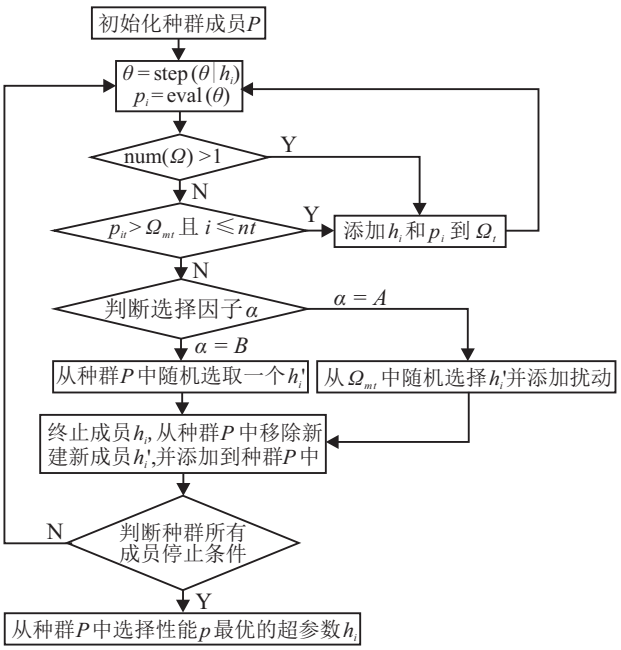


图 1　算法结构流程

step 1: 通过随机采样的方法随机选择超参数配置,初始化种群 P 中的成员(成员为需要搜索较优超参数的网络模型).

step 2: 种群中每个成员并行进行模型的迭代和评估.

step 3: 当种群成员模型达到提前停止的条件时,根据评估性能留下对应性能优越的成员模型,并保存到对应性能优的集合序列中.终止性能差的成员模型,新建种群新成员. 具体新成员通过选择因子进行探索或者利用,选择新成员模型的超参数.

step 4: 循环step 2和step 3,直至所有种群成员都达到终止条件,返回性能最优的超参数配置,实验完成.

注1　算法中提前停止条件是成员的性能在 t 时刻(t 时刻可以是迭代次数、时间等)低于成员性能序列 Ω_{mt} 中前 m 比例时,终止该性能差的成员,从种群 P 中移除.

注2　终止条件是种群中的成员达到设定的最大迭代次数、损失值、精度或奖励值时,终止该成员.

注3　新建成员有两种情况:一是种群初始化时随机选择超参数配置,新建成员;二是种群成员达到提前停止条件时,通过利用函数和探索函数生成超参数配置,新建成员.

群体中每个成员每训练 n 次后,通过评估成员得到的性能指标进行群体的优胜劣汰,留下性能指标较优的超参数配置,将较优的性能指标和超参数配置存入 t 时刻对应集合中, 终止性能指标较差的超参数配置,通过利用函数和探索函数重新选择超参数配置,作为新建成员重新开始训练. 如此循环下去,直至新建成员的总数达到人为设置的最大个数,且种群中所有成员训练次数达到最大的训练次数时,返回性能指标最优的超参数配置,实验完成.

评估函数在实现中的具体形式取决于应用程序,一般情况下,深度学习模型中评估函数主要通过评估模型优化过程的训练精确度或损失;在深度强化学习中评估函数主要通过评估强化学习模型的奖励值. PEHS 算法着重于搜索神经网络,深度强化学习等超参数模型. 在这些模型中一般采用的是通过梯度下降法进行优化 (SGD[15-16] 或 RMSProp[17]),评估函数是要优化度量的指标或验证集性能,利用函数从种群中性能优的个体中选择一个成员使用其超参数配置,探索函数进行超参数扰动,并复制给新建成员作为初始超参数. 终止性能差的成员,如此循环下去,直至找到性能最优的超参数配置.

通过多次执行梯度下降优化的迭代函数,利用和探索群体成员的超参数,PEHS 算法不仅对当前的成员进行梯度下降优化,而且周期地进行模型选择和超参数细化. 演化算法中进行种群演化 (探索) 的扰动幅度逐渐衰减,可以促进算法的收敛性. 根据目标设计的评估函数保证了种群演化(利用)的趋优稳定性. 因此,PEHS 算法不仅能够提高算法的收敛性,而且还使算法的性能更加稳定. 另外,PEHS 算法是异步并行的,不需要一个集中过程来协调种群成员的训练,更适合在分布式集群上进行计算.

3　实验分析

本实验主要用到 Ray[18] 并行分布式框架和 docker[19] 容器技术作为实验平台,通过深度学习、深度强化学习和分布式框架性能3个方面实验对PEHS算法进行分析.

3.1　实验平台

分布式框架对实验性能有着至关重要的作用,考虑到实验分析采用深度学习和深度强化学习模型训练,需要频繁的参数收集和重新分配,故采用一种为深度强化学习而设计的高性能并行分布式训练平台 —— Ray. Ray 不同于传统的分布式计算框架(如图2),被赋予了比 Spark[20] 更深入的任务抽象能力,故更适合分布式训练算法的学习和计算. 利用 docker 虚拟化技术(如图3),增加平台的可移植性和灵活性. 另外,平台集成了 gym、tensorflow、tune[21]、rllib[22] 等常用的机器学习相关工具集,使平台功能更加健全. 实验平台是在 CPU 集群上运行的,具体的硬件设备及配置信息如表1所示.

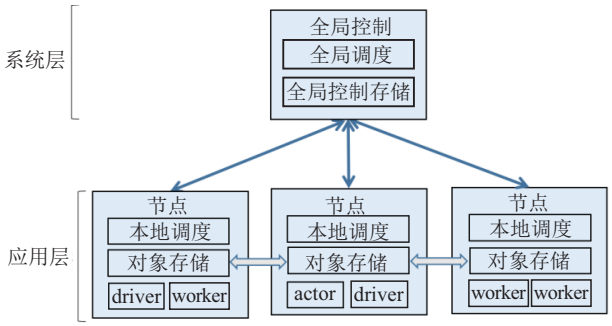
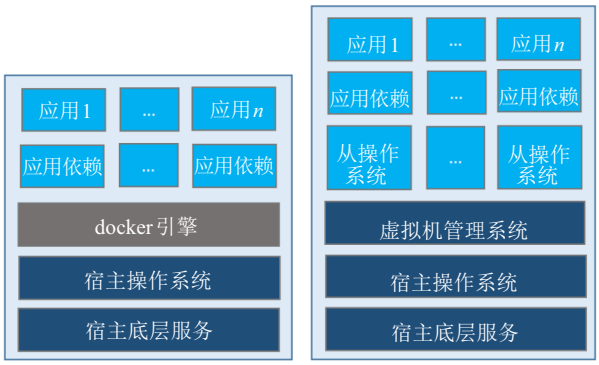


图 2 Ray 分布式训练平台计算框架



(a) docker (b) 虚拟机

图 3 docker 和虚拟机构架对比图

表 1 CPU 实验平台硬件及系统配置说明

硬件	描述	软件	描述名称
CPU	Intel Xeon E5-2630,2*6*2 cores per node	JDK	8u60
memory	128 G per node	python	3.6.5
drive	2T per node	docker	1.13.1
physical nodes	Huawei servers*18	spark	1.60
network	15.6Tbps	OS	CentOS7.2

3.2 实验结果分析

本实验通过 Hyperband、AsyHyperband、Random 三种优化算法和 PEHS 算法进行对比, 分析 PEHS 算法性能.

3.2.1 深度学习实验及结果分析

深度学习实验采用一个两层的卷积神经网络架构[23](convolutional neural networks, CNN) 进行图像识别训练, 其中 CNN 神经网络结构采用的卷积核都为 5×5, 池化步长为 2×2, 通道数分别为 32 和 64, 最后进行全连接和 dropout 处理. 实验的数据集是 mnist 数据, 超参数空间如表 2 所示. 实验用到了集群上 5 个节点, 每个节点 25 个 CPU 核心, 其中 PEHS 初始采样 10 次, 即每个样本训练使用 12 个 CPU(因使用 CPU 个数不能为小数, 故剩余 5 个没有使用), 重新采样 20 次 (即中间停止较差训练样本 20 次); AsyHyperband 和 Hyperband 初始采样 20 次, 每个训练样本使用 12 个 CPU, 其中部分实验样本需等待前边样本训练结束后开始训练; Random 采样 10 次, 每个训练样本使用 12 个 CPU. 实验的停止条件为训练精度达到 0.95.

表 2 CNN 实验超参数及取值范围

超参数	含义	取值范围名称
learning_rate	学习率	(1e-5, 1e-3)
activation-f_{n_1}	激励函数 1	[relu, elu, tanh, selu]
activation-f_{n_2}	激励函数 2	[relu, elu, tanh, selu]
batch	采样批次大小	(50, 200)
keep_pro	keep_pro 率	[0.5, 0.6, 0.7, 0.8]

根据实验结果, 从 3 个方面进行实验分析: 1) 在同样条件下重复 100 次实验, 且每次实验迭代 200 次后停止, 进行算法稳定性和收敛性分析; 2) 选取 4 种算法最先达到最大精确度的训练样本结果; 3) 达到最大精确度的平均迭代次数. 针对 3 种情况进行实验结果数据分析. 由 100 次重复的实验得到 100 组训练数据, 记录最终每组实验最优的训练精度, 表 3 所示为 PEHS 算法在该深度学习模型上最终最优精度在不同区间的个数, 可以看出 100 次实验最终的精度都在 0.95 以上, 故算法具有一定的稳定性. 另外, 该 100 次重复实验在迭代 10～20 次时已经全部达到 0.95 以上的精度, 且精度的幅度在 0.01 以内, 故算法的收敛性良好. 由 4 种算法最小迭代次数达到最大精确度的训练样本精度折线图 (图 4) 可知, AsyHyperband 和 Random 同时达到最大精度, 但 Random 的波动性较大, Hyperband 的训练时间最长, 但其波动性高于 Random; 说明 PEHS 优化算法训练收敛速度最快. 由 4 种算法达到最大精确度的平均迭代次数柱状图 (图 5) 可知, PEHS 的平均迭代次数最小, 性能最优, AsyHyperband 次之, Random 最差. 表 4 的深度学习实验结果分析通过数字的形式展示了 PEHS 算法最先找到最优的超参数模型, 且找到超参数模型的平均迭代时间也是最短. 综合以上图表的分析可知, PEHS 算法在深度学习超参数搜索上相对于 AsyHyperband, Hyperband 和 Random 三种算法, 超

表 3 PEHS 在深度学习方面重复 100 次实验结果分析

精度范围/%	个数
(95, 96)	58
[96, 97)	32
[97, 98)	10

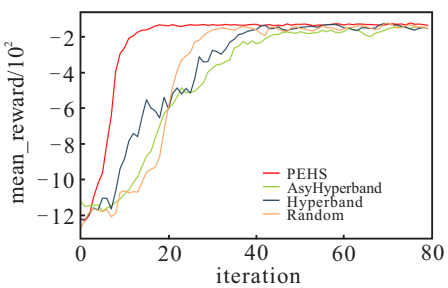


图 4 4 种超参数搜索算法最优实验精确度

参数搜索方面性能最好,能在较短的时间内找到性能较好的超参数模型,达到最优性能的平均迭代次数最少,表明PEHS算法具有一定的稳定性,收敛速度最快且性能最优.

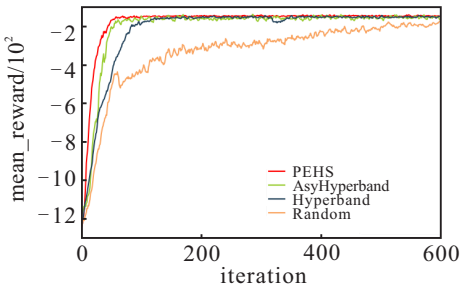


图 5　4种超参数搜索算法达到最大训练精度的平均迭代次数折线

表 4　深度学习实验结果分析

算法	训练精度0.95的迭代次数	训练精度0.95的平均迭代次数
PEHS	8	28.29
Hyperband	29	47.75
AsyHyperband	18	32.65
Random	18	87.16

3.2.2　深度强化学习实验结果及分析

本文深度强化学习实验环境选取gym中的pendulum仿真环境,策略选取OpenAI中默认的强化学习策略算法: PPO (proximal policy optimization)算法[24],超参数搜索空间如表5所示. 实验用到的节点数和训练样本数同深度学习实验. 实验停止条件为迭代600次.

表 5　深度强化学习实验超参数及取值范围

超参数	含义	取值范围名称
sample_batch_size	采样的批次大小	(100, 300)
lambda	GAE[25]中的参数lambda	(0.9, 1.0)
clip_parameter	PPO clip参数	(0.01, 0.5)
learning_rate	学习率	(1e-5, 1e-3)
num_sgd_iter	外部循环中SGD的迭代次数	(1, 30)
sgd_minibatch_size	driver中SGD的总批处理数	(128, 16 384)
trail_batch_size	每个SGD中的时间步长	(2 000, 160 000)

根据强化学习实验结果对3个方面进行实验分析: 1) 在同样条件下重复50次实验,进行算法稳定性和收敛性分析; 2) 选取4种算法中平均奖励最大的训练模型; 3) 4种算法中有效样本平均奖励的平均. 针对两种情况进行实验结果数据分析. 由50次重复的实验得到50组训练数据,记录最终每组实验最优奖励值,表6所示为PEHS算法在深度强化学习模型上最终奖励值在不同区间的个数. 50次重复实验的所有奖励值都在−150以上,故有一定的稳定性. 另外,该50次重复实验在迭代30～50次时已经全部达到−180以上的奖励值,且奖励值的幅度在10以内,故算法的收敛性良好. 4种算法最优训练样本平均奖励折线图如图6所示,在实验80次后4种算法最优实验平均奖励值几乎收敛,波动范围较小,故此实验选取了前80次的迭代次数进行实验分析并绘图. AsyHyperband和Random的曲线较平稳,但是收敛较慢,Hyperband性能最差,PEHS算法性能最优,且收敛最快. 选取4种算法中所有样本都达到最大迭代数600次的样本,画出这些样本的平均奖励的平均值的折线图,如图7所示. 由图7可知Random的波动较大,且收敛较慢,AsyHyperband和Hyperband性能次之, PEHS的平均收敛速度最快,且稳定. 表7所示的深度强化学习实验结果分析是对图6和图7结果的提取,PEHS算法达到最大的平均奖励需要迭代的次数最小,且达到最大平均奖励的平均迭代次数也最小. 综合以上图表的分析可知,PEHS算法在强化学习超参数搜索上的稳定性最好且算法的性能较优越.

表 6　PEHS在深度强化学习重复50次实验结果分析

奖励值范围	个数
(−180, −170)	31
[−170, −160)	11
[−160, −140)	8

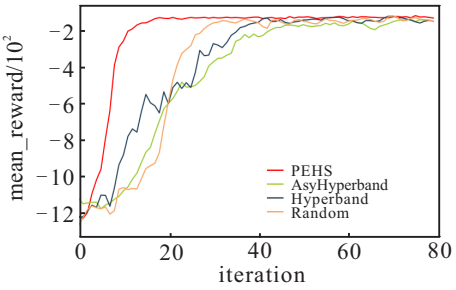


图 6　4种超参数搜索算法最优实验平均奖励折线

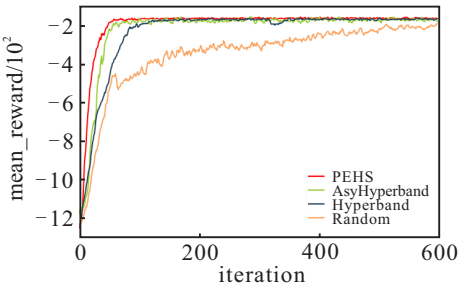


图 7　4种超参数搜索算法训练有效样本平均奖励的平均值的折线

表 7　深度强化学习实验结果分析

算法	达到平均奖励最大迭代次数	实验达到最大平均奖励的平均迭代次数
PEHS	19	37
Hyperband	79	68
AsyHyperband	41	143
Random	39	1 000

3.2.3 分布式实验结果及分析

分布式实验主要通过算法在不同节点数上进行不同采样数量的强化学习训练,判断并行数量对实验性能的影响. 强化学习采用的环境是pybullet中的HumanoidBulletEnv-0仿真环境,策略选取强化学习策略算法: A3C (asynchronous advantage actor-critic)算法[26],超参数搜索空间如表8所示. 由于实验硬件限制,分别在4个、8个、12个节点上进行实验. 4个节点上采样8次,最大重采用次数为20,每个训练样本使用11个CPU;8个节点上采样16次,最大重采用次数为40,每个样本用11个CPU;12个节点上采样24次,最大重采用次数为80,每个训练样本使用11个CPU. 实验的停止条件为迭代1 000次.

表8 分布式实验强化学习实验超参数及取值范围

超参数	含义	取值范围名称
sample_batch_size	采样的批次大小	(100, 300)
lambda	GAE中的参数lambda	(0.9, 1.0)
grad_clip	梯度clip参数	(30, 50)
vf_loss_coeff	值函数损失系数	(0.2, 0.6)
learning_rate	学习率	(1e-5, 1e-3)
entropy_coeff	熵系数	(0.001, 0.2)
min_iter_time_s	最小迭代次数值	(3, 10)

根据运行不同节点数量的实验结果进行两个方面实验分析: 1) 选取3种节点数实验中平均奖励最高的训练模型; 2) 3种节点数实验中所有有效样本平均奖励的平均. 针对两种情况进行实验结果数据分析: 1) PEHS的3种节点数实验最优训练样本平均奖励折线如图8所示,PEHS搜索算法的性能整体较为平稳,随着节点数量的增加,性能也在逐步提升; 2) 选取3种节点数实验中的有效实验样本结果,画出所有有效样本平均奖励平均值的折线如图9所示,可知节点数越多算法的整体平均奖励越高,说明随着节点数增多,采样量越大,找到最优参数模型的几率越大. 表9的分布式实验强化学习实验结果分析是对图8和图9结果的提取,PEHS算法的节点数和采样数越大找到最大平均奖励的训练次数越少,且达到最大平均奖励的平均训练次数也最小. 由分布式实验可知: PEHS在多节点的分布式实验上的性能更佳.

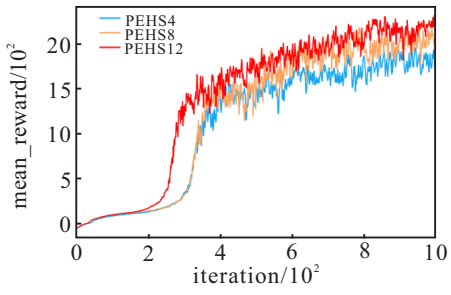


图8 PEHS算法3种节点数实验的最优平均奖励折线

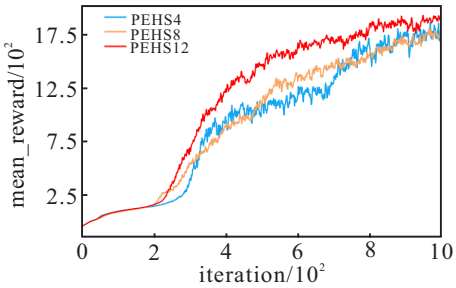


图9 PEHS算法3种节点数实验的有效样本平均奖励折线

表9 分布式实验强化学习实验结果分析

实验节点数	1 000次迭代的最大平均奖励	1 000次迭代平均的最大平均奖励
4	1 882	1 798
8	2 276	1 814
12	2 453	1 935

4 结 论

本文根据深度学习尤其是深度强化学习模型在超参数选择上对个人经验的依赖,且现有超参数搜索算法耗时较长、空间复杂度大等问题,提出了一种适应于深度学习的超参数搜索算法——基于种群演化的超参数异步并行搜索(PEHS)算法. 该算法融合异步并行计算和顺序优化优点,运用演化算法成熟的高鲁棒性和对全局优化广泛的适用性,结合利用与探索的平衡进行超参数搜索,有效地减少了超参数搜索的时间和计算复杂度. 另外,PEHS算法结合Ray并行分布式实验平台,不仅提高了算法的并行性,也使PEHS算法性能得到充分展示. 通过实验分析可知,PEHS算法在深度学习超参数搜索方面比传统的超参数搜索性能有较大提高,同时具备有效性和稳定性.

参考文献(References)

[1] Wistuba M, Schilling N, Schmidt-Thieme L. Hyperparameter search space pruning — A new component for sequential model-based hyperparameter optimization[C]. Proceedings of the 2015 European Conference on Machine Learning and Knowledge Discovery in Databases. Porto: ACM, 2015: 104-119.

[2] Hinton G E, Osindero S, Teh Y W. A fast learning algorithm for deep belief nets[J]. Neural Computation, 2006, 18(7): 1527-1554.

[3] Thakur N, Kumar S, Patle V K. Comparison of serial and parallel searching in multicore systems[C]. Proceedings of 2015 International Conference on Parallel, Distributed and Grid Computing. Solan: IEEE, 2015: 11-13.

[4] Yang E S, Kim J D, Park C Y, et al. Hyperparameter tuning for hidden unit conditional random fields[J]. Engineering Computations, 2017, 34(6): 2054-2062.

[5] Rahnama A H A, Toloo M, Zaidenberg N J. An LP-based hyperparameter optimization model for language modeling[J]. The Journal of Supercomputing,

2018, 74(5): 2151-2160.

[6] Jamieson K, Talwalkar A. Non-stochastic best arm identification and hyperparameter optimization[C]. Proceedings of the 19th International Conference on Artificial Intelligence and Statistics. Cadiz: PMLR , 2016, 240-248.

[7] Li L, Jamieson K, Desalvo G, et al. Hyperband: A novel bandit-based approach to hyperparameter optimization[J]. Journal of Machine Learning Research, 2018, 18: 1-52.

[8] Hutter F, Hoos H H, Leyton-Brown K. Sequential model-based optimization for general algorithm configuration[C]. Proceedings of the 5th International Conference on Learning and Intelligent Optimization. Rome: ACM, 2011: 507-523.

[9] Bergstra J, Bardenet R, Bengio Y, et al. Algorithms for hyper-parameter optimization[C]. Proceedings of the 24th International Conference on Neural Information Processing Systems. Granada: ACM, 2011: 2546-2554.

[10] Li L, Jamieson K, Rostamizadeh A, et al. Massively parallel hyperparameter tuning[J]. 2018, arXiv: 1810.05934.

[11] Gonzalez J, Dai Z, Hennig P, et al. Batch Bayesian optimization via local penalization[C]. Proceedings of the 19th International Workshop on Artificial Intelligence and Statistics. Cadiz: AISTATS, 2015: 648-657.

[12] Falkner S, Klein A, Hutter F. BOHB: Robust and efficient hyperparameter optimization at scale[J]. 2018, arXiv: 1807.01774.

[13] Golovin D, Solnik B, Moitra S, et al. Google vizier: A service for black-box optimization[C]. Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Halifax: ACM, 2017: 1487-1495.

[14] Gong D W, Sun J, Miao Z. A set-based genetic algorithm for interval many-objective optimization problems[J]. IEEE Transactions on Evolutionary Computation, 2016, 22(1): 47-60.

[15] Jaderberg M, Dalibard V, Osindero S, et al. Population based training of neural networks[J]. 2017, arXiv: 1711.09846.

[16] Goyal P, Dollár P, Girshick R, et al. Accurate, large minibatch SGD: Training ImageNet in 1 hour[J]. 2017, arXiv: 1706.02677.

[17] Valente P C, Dallas V. Spectral imbalance in the inertial range dynamics of decaying rotating turbulence[J]. Physical Review E, 2017, 95(2): 023114.

[18] Moritz P, Nishihara R, Wang S, et al. Ray: A distributed framework for emerging AI applications[C]. Operating Systems Design and Implementation. Carlsbad: USENIX, 2018: 561-577.

[19] Avino G, Malinverno M, Malandrino F, et al. Characterizing docker overhead in mobile edge computing scenarios[C]. Proceedings of the 17th Workshop on Hot Topics in Container Networking and Networked Systems. New York: ACM, 2017: 30-35.

[20] Zaharia M, Xin R S, Wendell P, et al. Apache spark: A unified engine for big data processing[J]. Communications of the ACM, 2016, 59(11): 56-65.

[21] Liaw R, Liang E, Nishihara R, et al. Tune: A research platform for distributed model selection and training[J]. 2018, arXiv: 1807.05118.

[22] Liang E, Liaw R, Moritz P, et al. RLlib: Abstractions for distributed reinforcement learning[C]. Proceedings of the 35th International Conference on Machine Learning. Stockholmsmässan: ICML, 2018: 3053-3062.

[23] Rocco I, Arandjelovic R, Sivic J, et al. Convolutional neural network architecture for geometric matching[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2019, 41(11): 2553-2567.

[24] Schulman J, Moritz P, Levine S, et al. High-dimensional continuous control using generalized advantage estimation[J]. 2015, arXiv: 1506.02438.

[25] Mnih V, Badia A P, Mirza M, et al. Asynchronous methods for deep reinforcement learning[C]. International Conference on Machine Learning. New York: ACM, 2016: 1928-1937.

[26] Schulman J, Wolski F, Dhariwal P, et al. Proximal policy optimization algorithms[J]. 2017, arXiv: 1707.06347.

作者简介

蒋云良 (1967−), 男, 教授, 博士, 从事智能信息处理、地理信息系统等研究, E-mail: jyl@zjhu.edu.cn;

赵康 (1994−), 男, 硕士生, 从事数据挖掘、强化学习的研究, E-mail: zhaokang@zjnu.edu.cn;

曹军杰 (1991−), 男, 博士生, 从事机器学习、决策与控制的研究, E-mail: cjunjie@zju.edu.cn;

范婧 (1986−), 女, 讲师, 博士生, 从事数据挖掘、数据可视化的研究, E-mail: fanjing105@zju.edu.cn;

刘勇 (1980−), 男, 教授, 博士生导师, 从事智能机器人系统、机器人感知等研究, E-mail: yongliu@iipc.zju.edu.cn.

(责任编辑：孙艺红)