• 计算机工程与应用 •

基于强化学习的模型选择和超参数优化

吴 佳，陈森朋*，陈修云，周 瑞

(电子科技大学信息与软件工程学院 成都 610054)

【摘要】随着机器学习技术的不断发展，机器学习算法种类的增多以及模型复杂度提高，造成了实践应用中的两大难题：算法模型选择及模型超参数优化。为了实现模型选择和超参数优化的自动处理，该文提出了一种基于深度强化学习的优化方法。利用长短期记忆 (LSTM) 网络构建一个智能体 (Agent)，自动选择机器学习算法模型及对应的超参数组合。该智能体以最大化机器学习模型在验证数据集上的准确率为目标，利用所选择的模型在验证数据集上的准确率作为奖赏值 (reward)，通过强化学习算法不断学习直到找到最优的模型以及超参数组合。为了验证该方法的可行性及性能，在 UCI 标准数据集上将其与传统优化方法中基于树状结构 Parzen 的估计方法和随机搜索方法进行比较。多次实验结果证明该优化方法在稳定性、时间效率、准确度方面均具有优势。

关 键 词 深度强化学习； 超参数优化； LSTM 网络； 机器学习； 模型选择
中图分类号 TP391 文献标志码 A doi:10.12178/1001-0548.2018279

Reinforcement Learning for Model Selection and Hyperparameter Optimization

WU Jia, CHEN Sen-peng*, CHEN Xiu-yun, and ZHOU Rui

(School of Information and Software Engineering, University of Electronic Science and Technology of China Chengdu 610054)

Abstract With the development of machine learning technology, the number of machine learning algorithms grows rapidly and the models become more and more complex. That causes two major problems in practice: the selection of machine learning models and the hyperparameter optimization. In order to tackle these issues, this paper proposes a new method based on deep reinforcement learning. Long short-term memory (LSTM) network is used to build an agent which automatically selects the machine learning model and optimizes hyperparameters for a given dataset. The agent aims to maximize the accuracy of the selected machine learning model on the validation dataset. At each iteration, it utilizes the accuracy of the selected model on the validation dataset as a reward signal to improve its decision for the next time. The reinforcement learning algorithm is used to guide the learning process for the agent. To verify the idea, the proposed method is compared with two widely optimization methods, tree-structured Parzen estimator and random search on UCI datasets. The results show that the proposed method outperforms other methods in terms of stability, time efficiency and accuracy.

Key words deep reinforcement learning; hyperparameter optimization; LSTM network; machine learning; model selection

近年来，机器学习已广泛应用于如机器翻译[1-2]、语音识别[3-4]、图像识别[5-6] 和游戏[7] 等众多领域。针对某一问题，如何快速构建一个成熟、可靠的机器学习模型就显得尤为重要。为了满足行业需要，使机器学习算法能够得到快速、高效的利用，一大批企业针对普通用户开发出了一些应用系统，如 DataRobot.com[8]、BigML.com[9]、Wise.io[10] 等。在机器学习算法的应用中，不可避免涉及两个重要问题：算法模型选择和超参数优化。

现有的机器学习算法众多，具有代表性的算法有逻辑回归 (logistic regression)、支持向量机 (support vector machine)、决策树 (decision tree) 和随机森林 (random forest) 等。针对不同的问题，没有一个机器学习算法模型能够适用于所有问题。在同一问题上，不同的方法所达到的性能也存在不同程度的差异。这给机器学习算法的使用者造成了不小的麻烦。算

收稿日期：2018－10－31；修回日期：2019－09－04
基金项目：国家自然科学基金 (61503059)；四川省科技厅重点研发计划 (2018GZ0464)
作者简介：吴佳 (1980－)，女，博士，副教授，主要从事深度强化学习、智能交通系统等方面的研究.
通信作者：陈森朋，E-mail：3040661835@qq.com

法模型选择成了机器学习算法广泛应用的一大障碍。

另外，超参数优化同样成为了机器学习算法应用中的难点之一。超参数不同于算法模型内部的参数，它是在算法模型训练之前设置的参数。在训练开始之前，往往希望找到一组超参数的值，即超参数组合，使得算法模型可以在合理的时间范围内对某一数据集的分类或拟合达到最佳性能。这个过程被称为超参数优化，它对机器学习算法的性能起着至关重要的作用。在实践中通常需要不断调整超参数的值，最终选择最佳的超参数组合。若算法模型的超参数搜索空间较大，该过程将非常耗时。

因此，针对某一问题 (或数据集)，最终结果很大程度上是由机器学习算法模型和算法对应的超参数组合共同决定的。本文提出了一种基于深度强化学习的方法，用于自动实现机器学习算法的选择和超参数的优化。该方法利用长短期记忆 (LSTM) 网络[8] 构建一个智能体 (Agent) 来代替机器学习使用者选择最优的机器学习算法及其超参数；Agent 在训练集上训练所选择的机器学习算法及超参数组合所对应的算法模型，在验证数据集上验证该算法模型的性能；以在验证集上的准确度作为奖赏值，利用策略梯度算法 (policy gradient)[9] 优化 Agent 的决策。经过多次迭代，Agent 选择出适合该问题的最优模型及对应的超参数。在 Agent 训练过程中，梯度方差较大，本文提出引导数据池来解决该问题。本文主要的贡献在于以下 3 点：

1) 使用强化学习框架来解决模型选择和超参数优化问题；

2) 提出了数据引导池结构来提高方法的稳定性；

3) 通过在标准数据集上对 8 种机器学习算法进行优化，相比于其他方法，本文提出的方法达到了最好的优化结果。

1 相关工作

模型选择和超参数优化问题通常称为 CASH (combined algorithm selection and hyperp-arameter optimization)[10] 问题。CASH 问题定义如下：

针对某一数据集，寻找使得式 (1) 的值最小的算法A^*及相应的超参数配置λ^*：

$$A^*_{\lambda^*}\in \underset{A^{(j)}\in \boldsymbol{A}\ \lambda\in\Lambda^{(j)}}{\operatorname{argmin}} \frac{1}{K}\sum_{i=1}^{K}\mathcal{L}\left(A^{(j)}_{\lambda},\boldsymbol{D}^{(i)}_{\text{train}},\boldsymbol{D}^{(i)}_{\text{valid}}\right) \tag{1}$$

式中，$\boldsymbol{A}=\left\{A^{(1)},A^{(2)},\cdots,A^{(N)}\right\}$表示可供选择的机器学习算法集合；$\boldsymbol{\Lambda}=\left\{\Lambda^{(1)},\Lambda^{(2)},\cdots,\Lambda^{(N)}\right\}$表示算法集合$\boldsymbol{A}$中每个算法对应的超参数空间的集合；$\Lambda^{(i)}$为算法$A^{(i)}$的超参数空间，$i=[1,N]$；$\boldsymbol{D}_{\text{train}}=\{(x_1,y_1),(x_2,y_2),\cdots,(x_n,y_n)\}$表示训练集，其在训练过程中并被分割成均等大小的 K 份$\left\{\boldsymbol{D}^{(1)}_{\text{valid}},\boldsymbol{D}^{(2)}_{\text{valid}},\cdots,\boldsymbol{D}^{(K)}_{\text{valid}}\right\}$用于 K 折交叉验证，$\boldsymbol{D}^{(i)}_{\text{train}}=\boldsymbol{D}_{\text{train}}\backslash\boldsymbol{D}^{(i)}_{\text{valid}},i=[1,K]$；$\mathcal{L}\left(A^{(j)}_{\lambda},\boldsymbol{D}^{(i)}_{\text{train}},\boldsymbol{D}^{(i)}_{\text{valid}}\right)$表示拥有超参数配置 λ 的机器学习算法$A^{(j)}$通过在训练数据集$\boldsymbol{D}^{(i)}_{\text{train}}$上训练，在验证数据集$\boldsymbol{D}^{(i)}_{\text{valid}}$上验证得到的损失函数 (loss function)。

为了解决 CASH 问题，研究者提出了一系列解决方案，如随机搜索，贝叶斯优化等。随机搜索在算法和对应的超参数构成的搜索空间中随机采样。该方法执行起来效率高且操作简单，经过少量的尝试就可以搜索到性能较好的机器学习算法及相应的超参数的值。但文献 [10] 表明，随机搜索方法只有在达到或接近最优值的组合的数量占总的组合数量的比重超过 5% 时，搜索效率较高；其他情况下，随机搜索方法的表现较差，很难搜索到最优值。Auto-WEKA 是一个基于机器学习工具包 WEKA[11] 的自动化机器学习框架。CASH 问题首先在 Auto-WEKA[12] 系统中被解决，其核心是贝叶斯优化方法，主要包括基于高斯过程的贝叶斯优化方法[13]，基于模型的顺序算法配置方法 (sequential model-based algorithm configuration, SMAC)[14] 及其改进版本的基于树状结构 Parzen 的估计方法 (TPE)[15]。文献 [16] 使用热启动技术提升 SMAC 的性能。自适应协方差矩阵进化策略 (CMA-ES) 算法[17]，是基于进化算法的一种改进算法，主要用来解决非线性、非凸的优化问题，在解决模型选择和超参数优化问题也具有很好的效果。最近，BOHB[18] 被提出，该方法将贝叶斯优化与 HyperBand 方法相结合用于解决模型选择和超参数优化问题，并具有很好的优化效果。上述这些方法存在一定的局限性，如基于高斯过程的贝叶斯优化方法只适用于低维空间的超参数优化问题。在搜索性能方面，基于贝叶斯优化的方法容易陷入局部最优，很难探索出模型性能最好的算法及超参数组合。在时间性能方法，对于拥有较大的搜索空间的问题，贝叶斯优化方法时间效率会大幅降低。

强化学习 (reinforcement learning，RL)[19] 是从动物学习、参数扰动自适应控制等理论发展而来的。其基本原理是：智能体的行为决策得到环境的反馈，即奖赏值；通过最大化累积奖赏值，以学习到最优的行动策略。通常利用马尔可夫决策过程对强化学习问题进行建模。随着问题复杂度增加，谷歌的人工智能团队将具有感知能力的深度学习和具有决策能力的强化学习相结合，即深度强化学习

(deep reinforcement learning，DRL)[20]，成功解决了诸如与人类进行围棋对弈[21]等复杂任务。通过将深度学习与强化学习结合，实现了从感知 (perception) 到动作 (action) 的端对端的学习。目前，深度强化学习在视频[22]、游戏[23]、机器人[24]等领域获得广泛的应用。本文利用深度强化学习的优势来解决 CASH 问题。

2　模型选择和超参数优化方法

2.1　Agent 模型结构

模型选择和超参数优化可看作一个多阶段决策问题，在每个阶段 (时刻) 针对某个模型或者超参数做出相应决策，因此不同的时刻产生不同的输出。由于模型以及超参数之间存在相关性，每个阶段的决策又是相互关联、相互影响的。根据上述特点，模型和超参数设置的过程可由一个可变的字符串来表示，利用长短时记忆神经网络 (LSTM) 构造的智能体来生成这样的字符串，具体优化过程如图 1 所示。

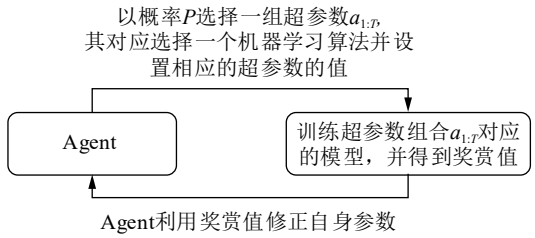


图 1　Agent 选择机器学习算法与优化超参数的流程

智能体以概率 P 为选择一组超参数 $a_{1:T}$，其中，动作 a_1 对应选择的机器学习算法；动作序列 $a_{2:T}$ 表示 a_1 选择的算法模型中超参数的取值。T–1 为 a_1 选择的算法模型中超参数的总个数，不同的机器学习算法 T 的取值不同。接下来，在训练数据集上训练智能体选择 $a_{1:T}$ 所对应的算法模型；以训练好模型在验证集上的准确率作为奖赏值 (reward)，利用强化学习中的策略梯度算法来训练 Agent。奖赏值引导 Agent 在下次迭代中以更高的概率选择准确率高的算法及对应的超参数的值。随着时间的推移，智能体将学会如何针对某一问题 (或数据集) 自主选择最优的机器学习算法和相关超参数。

本文利用 LSTM 网络构造 Agent 来自动选择算法模型及超参数组合。该 Agent 的网络结构如图 2 所示，它的核心由 3 层 LSTM 网络构成，每层拥有 35 个神经元节点；输出层由 softmax 函数构成；输入层、输出层与 3 层 LSTM 网络结构之间各有一个全连接层。Agent 中 3 层 LSTM 网络结构在任意时刻的结构、参数共享。Agent 在不同时刻输出不同的模型/超参数选择，并把不同时刻选择值在候选值中的索引位置作为下一时刻的输入数据，当所有超参数值生成后，Agent 输出停止。Agent 在任意时刻的输出为对某个模型/超参数所有候选值的评估。该值越大，对应的预选值被选中的概率越高；反之，越低。Agent 根据这些评估值做出最优的选择。

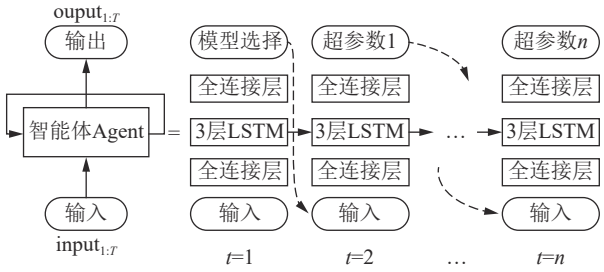


图 2　Agent 的网络结构图

2.2　Agent 训练学习

当 Agent 以概率 P 选择模型/超参数序列 $a_{1:T}$ 后，将 $a_{1:T}$ 对应的算法模型在训练数据集上训练至收敛，再在验证数据集上运行得到的准确率作为奖励信号 R 来优化 Agent 的参数 θ，使得随着时间的推移，Agent 学会选择准确率更高的模型/超参数组合。Agent 训练方法采用强化学习算法中的策略梯度[12]，算法的优化目标为最大化期望总奖赏：

$$J(\theta)=\max E_{P(a_{1:T};\theta)}[R] \tag{2}$$

式中，$P(a_{1:T};\theta)$表示表示 Agent 输出模型/超参数序列 $a_{1:T}$ 的概率。

由于优化目标是找到一个参数 θ，使得期望总奖赏最大化。根据梯度下降算法，通过求解目标函数的梯度，进而更新参数 θ，最终可求得局部最优值：

$$\nabla_\theta(J(\theta))=\sum_{t=1}^{T}E_{P(a_{1:T};\theta)}[R\nabla_\theta\log P(a_t|a_{(t-1):1};\theta)] \tag{3}$$

根据上式，可以看出$\nabla_\theta(J(\theta))$为函数$\nabla_\theta\log P(a_t|a_{(t-1):1};\theta)$的期望。本文利用在固定参数 θ 下 m 次采样的均值作为梯度更新的无偏估计：

$$\nabla_\theta(J(\theta))\approx\frac{1}{m}\sum_{k=1}^{m}\sum_{t=1}^{T}\nabla_\theta\log P(a_t|a_{(t-1):1};\theta)(R_k-b) \tag{4}$$

式中，R_k 为第 k 次采样的模型在验证数据集上的准确率；b 为基准值，其值为已采样到的算法模型的准确率的指数滑动平均值。基准值设置的目的在

于减小 Agent 训练过程中的方差。

2.3 减小方差

虽然策略梯度算法中采用了基准值降低训练中的方差，但 Agent 训练过程中仍存在方差过大问题。为了进一步减小方差，提高 Agent 决策的稳定性，本文为 Agent 添加一个“引导数据池”。具体来说，设置一个大小为 m 的数据池，保存到当前时刻性能表现最好的 m 条数据，定期把这批数据送入到 Agent 中进行训练。这样做起到了一个把握优化方向、防止模型方差过大的作用，故称之为引导数据池。具体算法如下。

输入：无

输出：最优超参数组合

1) 初始化 Agent 的模型参数 θ;
2) 初始化引导数据池 top_data;
3) for i = 1 : (N/m)
4) for j = 1 : m
5) 初始化 Agent 的输入数据 input 为全 1 向量;
6) for t = 1 : T
7) 将 t 时刻的输入数据添加进输入数据列表;
8) 将 Agent 在 t 时刻输出值作为下一时刻的输入数据;
9) 将 t 时刻的选择添加进动作列表 actions;
10) end for;
11) end for;
12) 在训练数据集上训练与动作列表 actions 对应的模型，在验证数据上验证模型准确性，得到奖赏值并存入奖励值列表 rewards;
13) 更新引导数据池 top_data;
14) if i == 0
15) b = mean(rewards);
16) end if;
17) if (i+1) % n_step == 0
18) 获取引导数据池 top_data 中的数据；
19) 利用策略梯度算法更新 Agent 的参数 θ;
20) else
21) 利用策略梯度算法更新 Agent 的参数 θ;
22) end if;
23) b=b*r + mean(rewards)*(1-r);
24) end for

其中，N 表示 Agent 采样的总批次数 (迭代次数)；m 为 Agent 更新一次模型参数所需的数据量大小；n_step 为 Agent 利用数据引导池中的数据更新模型参数的步伐大小；r 控制着基准值 b 的滑动范围。

为了验证引导数据池的有效性，同样对随机森林算法的超参数进行优化实验。在相同的实验环境下，本文进行了共 20 次的未添加引导数据池的 Agent 与添加了引导数据池的 Agent 的对比实验，如图 3 所示。实验结果表明，添加了引导数据池的 Agent 在 18 次运行中都选择到了同一个最优的超参数组合，而未添加引导数据池的 Agent 只有 1 次选择到了最优的超参数组合。未添加引导数据池的 Agent 所存在的高方差、不稳定的问题得到了有效的解决。

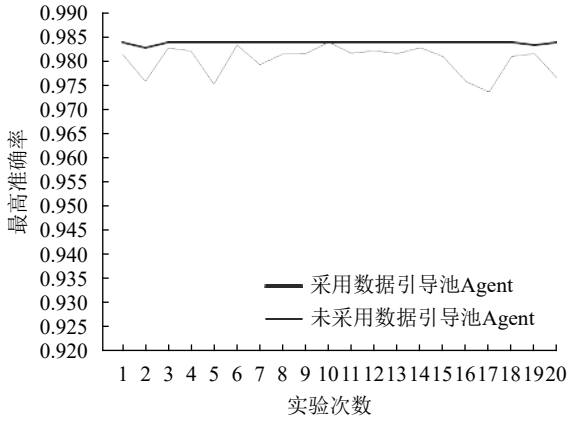


图 3 采用数据引导池 Agent 与未采用数据引导池 Agent 性能比较 (随机森林算法超参数优化)

3 实验结果及分析

3.1 搜索空间及数据集

根据文献 [25] 中对 179 种机器学习算法在 UCI machine learning repository 数据集上的评估结果，本文挑选了一些具有代表性的学习算法以及相应超参数，并为这些超参数设定了一些候选值，详细情况如表 1 所示。

UCI machine learning repository 数据集是一种常见的、用于分类任务的数据集。采用两种 UCI 标准数据集进行测试，数据集信息如表 2 所示。数据集的原始数据经过预处理后，将整个数据集分成验证集和测试集两部分，验证集占整个数据集的 80%，数据集中剩余的 20% 的数据将作为测试集，用于测试所选择的超参数组合对应的模型最终的性能。

在构建 Agent 的过程中，Agent 采用 3 层 LSTM，每一层有 35 个隐藏节点。采样的总次数 N 设置为 5 000 次；每次采样的超参数组合的数量 m 设置为 8；引导池大小设置为 8；数据引导池的利用间隔

表 1　候选算法模型及对应超参数候选值

算法模型	超参数	候选值范围	间隔
RandomForestClassifier	n_estimators	[100～1 200]	100
	max_depth	[2～30]	2
	min_samples_split	[1～99]	2
	min_samples_leaf	[1～99]	2
	max_features	[sqrt,log2,None]	无
	criterion	[gini,entropy]	无
	bootstrap	[True,False]	无
XGBClassifier	max_depth	[3～25]	2
	gamma	[0.05～0.9]	0.05
	min_child_weight	[1～9]	2
	subsample	[0.1～0.9]	0.1
	colsample_bytree	[0.1～0.9]	0.1
	reg_alpha	[0.0～1.0]	0.1
	reg_lambda	[0.01～0.1]	0.01
	learning_rate	[0.005～0.1]	0.005
DecisionTreeClassifier	criterion	[gini,entropy]	无
	splitter	[best,random]	无
	max_depth	[2～30]	2
	min_samples_split	[1～99]	2
	min_samples_leaf	[1～99]	2
	max_features	[sqrt,log2,None]	无
SVC	C	[0.000 5～0.01]	0.000 5
	kernel	[linear,poly,rbf,sigmoid]	无
	class_weight	[balanced,None]	无
KneighborsClassifier	n_neighbors	[2～100]	2
	weights	[uniform,distance]	无
	algorithm	[auto,ball_tree,kd_tree,brut]	无
	leaf_size	[5～50]	5
	p	[1～5]	1
AdaBoostClassifier	n_estimators	[100～1 200]	100
	learning_rate	[0.1～1.0]	0.1
	algorithm	[SAMME,SAMME.R]	无
ExtraTreesClassifier	n_estimators	[100～1 200]	100
	criterion	[gini,entropy]	无
	max_features	[sqrt,log2,None]	无
	max_depth	[1～29]	2
	min_samples_split	[1～99]	2
	min_samples_leaf	[1～99]	2
BaggingClassifier	n_estimators	[100～1200]	100
	max_samples	[0.1～0.9]	0.1
	max_features	[0.1～0.9]	0.1
	bootstrap	[True,False]	无
	bootstrap_features	[True,False]	无
	warm_start	[True,False]	无

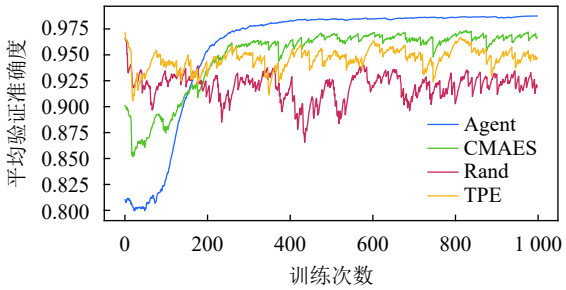
表 2　数据集基本信息

数据集名称	UCI手写数字数据集	UCI Spambase数据集	UCI Car Evaluation数据集
适用任务类别	多分类	二分类	多分类
标签类别	10种(0–9)整数	2种(0,1)	4种
特征数量/个	64	57	6
是否有缺失值	否	否	否
数据集大小/条	5 620	4 601	1 728

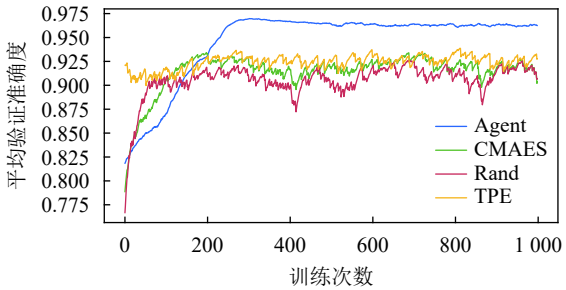
*n*_step 设置为 10；基准值 *b* 的控制率 *r* 设置为 0.8；以−0.2～0.2 之间的随机值对 Agent 的权重进行初始化，使用 Adam 优化器[26] 进行策略优化。

3.2　实验结果及分析

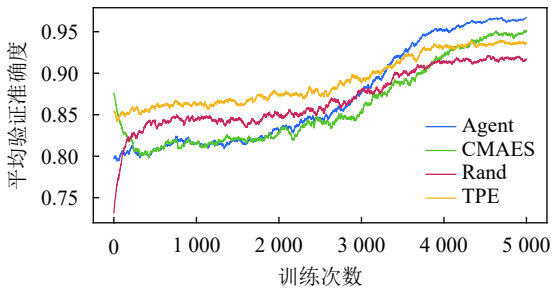
实验在两个 UCI 标准数据集下进行，对比了 CMAES 优化方法、TPE 优化方法、随机搜索方法的准确度、时间效率、稳定性。实验结果如图 4 和表 3 所示，所有结果为 3 次实验后统计值。图中所示为 3 次实验的平均值和 1 倍标准差。



a. UCI mnist数据集



b. UCI spambase数据集



c. UCI Car-Evaluation数据集

图 4　优化过程中准确度 (Agent 代表本文提出方法；Rand 代表随机搜索；TPE 代表基于树状结构 Parzen 的估计方法方法；CMAES 指 CMAES 优化方法)

图 4 为优化过程中所有方法的准确度。横轴表示采样次数；纵轴表示每次采样后选择的模型在验证集上的准确率。图中数据为 3 次实验的均值和一倍标准差。表 3 结果为 5 000 次采样完成后， 4 种方法选择的模型的统计结果 (3 次实验平均值)。其中，准确度表示模型在验证集上最高准确率的平均值；耗时为完成 3 次训练所耗费时间的平均值，该

值大小反应了优化算法的时间性能；标准差为 3 次实验最高准确率统计的标准差，该项数据反应了算法的稳定性。

表 3 优化结束后统计结果

数据集	UCI手写数字数据集实验结果		
方法	准确度	耗时/min	标准差
Agent	**0.987 5**	**219.4**	**0.009 36**
TPE	0.983 9	1 167.5	0.026 43
Rand	0.985 3	1 825.9	0.038 62
CMAES	0.985 6	776.3	0.006 71
数据集	UCI Spambase数据集实验结果		
方法	准确度	耗时/min	标准差
Agent	**0.955 2**	**199.5**	**0.012 2**
TPE	0.953 5	462.1	0.019 4
Rand	0.952 7	870.5	0.028 0
CMAES	0.954 0	355.6	0.013 4
数据集	UCI Car Evaluation数据集实验结果		
方法	准确度	耗时/min	标准差
Agent	**0.975 4**	**60.7**	**0.003 4**
TPE	0.948 2	78.3	0.010 6
Rand	0.921 6	80.6	0.033 4
CMAES	0.952 1	66.3	0.009 7

通过分析实验数据可以看出，本文提出的优化算法能够在最短的时间搜索到最优的结果。虽然随机搜索、TPE 优化方法和 CMAES 优化方法也能达到较好的优化结果，相比之下，Agent 能够用远少于前两种方法的时间搜索出更优的算法模型和超参数组合，尤其是在问题规模增大时，Agent 优化方法仍具有很高的时间效率，综合性能更好。TPE 方法使用从开始到当前时刻所有采样的数据进行训练，这就造成了对数据的极大依赖，容易造成过拟合，最终陷入局部最优。相比之下，Agent 在对自身的模型参数进行更新时，每次都是由当前时刻采样到的新数据进行训练，能够搜索到更好最优解。随机搜索算法的搜索效率相比与 TPE 算法更低，究其原因在于随机搜索方法的采样具有随机性，随着搜索空间增大，搜索到相同模型超参数组合可能性越小，因而耗费在模型训练上的时间也就越多。通过实验还发现 Agent 通过数据引导池结构，相比于 TPE 和随机搜索方法能够有效的减小训练时的方差，使训练更加稳定。

4 结 束 语

本文提出了一种基于深度强化学习的超参数优化方法。该方法利用长短时记忆网络构建了一个 Agent，针对不同问题 (数据集) 自动进行算法选择超参数优化。Agent 以最大化模型在验证集上的准确率为目标，以 Agent 每次选择的所对应的模型在验证数据集上的准确率作为奖赏值，利用策略梯度算法来修正 Agent 的模型参数。经过多次迭代，Agent 最终收敛并选择出最优的算法模型及超参数组合。为了验证算法的可行性和性能，利用 Agent 对两种标准数据集进行优化实验。通过对比 TPE 和随机搜索两种具有代表性的超参数优化方法，本文提出的方法在准确率、运行时间效率和稳定性上均优于上述算法，特别是对于规模较大的问题，具有绝对优势，其完成优化所需的时长最低仅约为随机搜索方法的 12% 和 TPE 优化方法的 19%。

参 考 文 献

[1] WU Y, SCHUSTER M, CHEN Z, et al. Google's neural machine translation system: Bridging the gap between human and machine translation[J]. CoRR, 2016, 1: 1-10.

[2] ZHOU Shi-yu, DONG Lin-hao, XU Shuang, et al. Syllable-based sequence-to-sequence speech recognition with the transformer in Mandarin Chinese[J]. Interspeech, 2018, 10: 791-795.

[3] SAINATH T N, WEISS R J, WILSON K W, et al. Multichannel signal processing with deep neural networks for automatic speech recognition[J]. IEEE/ACM Transactions on Audio, Speech, and Language Processing, 2017, 25(5): 965-979.

[4] KIM J, EL-KHAMY M, LEE J. Residual LSTM: Design of a deep recurrent architecture for distant speech recognition[J]. Interspeech, 2017, 12: 1591-1595.

[5] BIGDELI S A, JIN M, FAVARO P, et al. Deep mean-shift priors for image restoration[C]//Advances in Neural Information Processing Systems 30. Long Beach, CA: DBLP, 2017: 763-772.

[6] LIN C H, LUCEY S. Inverse compositional spatial transformer networks[C]//2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Honolulu, HI, USA: IEEE Computer Society, 2017: 2252-2260.

[7] SILVER D, SCHRITTWIESER J, SIMONYAN K, et al. Mastering the game of Go without human knowledge[J]. Nature, 2017, 550(7676): 354-359.

[8] HOCHREITER S, SCHMIDHUBER J. Long short-term memory[J]. Neural Computation, 1997, 9(8): 1735-1780.

[9] WILLIAMS R J. Simple statistical gradient-following algorithms for connectionist reinforcement learning[J]. Machine Learning, 1992, 8(3-4): 229-256.

[10] BERGSTRA J, BENGIO Y. Random search for hyper-parameter optimization[J]. Journal of Machine Learning Research, 2012, 13(1): 281-305.

[11] PFAHRINGER B, REUTEMANN P, WITTEN I H, et al. The WEKA data mining software: An update[J]. ACM SIGKDD Explorations Newsletter, 2009, 11(1): 10-21.

[12] THORNTON C, HUTTER F, HOOS H H, et al. Auto-

WEKA: Combined selection and hyperparameter optimization of classification algorithms[C]//International Conference on Knowledge Discovery and Data Mining.[S.l.]: ACM, 2012: 847-855.

[13] SNOEK J, LAROCHELLE H, ADAMS R P S. Practical bayesian optimization of machine learning algorithms[J]. Advances in Neural Information Processing Systems, 2012, 4: 2951-2959.

[14] HUTTER F, HOOS H H, LEYTONBROWN K. Sequential model-based optimization for general algorithm configuration[C]//Learning and Intelligent Optimization - International Conference. Rome, Italy: DBLP, 2011: 507-523.

[15] JAMES B, REMI B, YOSHUA B, et al. Algorithms for hyper-parameter optimization[C]//Advances in Neural Information Processing Systems. Granada, Spain: DBLP, 2011: 2546-2554.

[16] LINDAUER M, HUTTER F. Warmstarting of model-based algorithm configuration[C]//Association for the Advancement of Artificial Intelligence(AAAI). Louisiana, USA: DBLP, 2018: 1355-1362.

[17] HANSEN N. The CMA evolution strategy: A comparing review[J]. Studies in Fuzziness & Soft Computing, 2007, 192: 75-102.

[18] FALKNER S, KLEIN A, HUTTER F. BOHB: Robust and efficient hyperparameter optimization at scale[J]. Dy and Krause, 2018, 24: 1437-1446.

[19] SZEPESVÁRI C. Algorithms for reinforcement learning[J]. International Conference on Computing, 2009, 21: 234-243.

[20] MNIH V, KAVUKCUOGLU K, SILVER D, et al. Human-level control through deep reinforcement learning[J]. Nature, 2015, 518(7540): 529-533.

[21] SILVER D, HUANG A, MADDISON C J, et al. Mastering the game of go with deep neural networks and tree search[J]. Nature, 2016, 529(7587): 484.

[22] LEVINE S, FINN C, DARRELL T, et al. End-to-end training of deep visuomotor policies[J]. Journal of Machine Learning Research, 2015, 17(1): 1334-1373.

[23] MNIH V, KAVUKCUOGLU K, SILVER D, et al. Playing atari with deep reinforcement learning[J]. Computer Science, 2013, 10: 431-439.

[24] GU S, HOLLY E, LILLICRAP T, et al. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates[C]//International Conference on Robotics and Automation. Singapore: DBLP, 2017: 3389-3396.

[25] CERNADAS E, AMORIM D. Do we need hundreds of classifiers to solve real world classification problems?[J]. Journal of Machine Learning Research, 2014, 15(1): 3133-3181.

[26] KINGMA D P, BA J. Adam: A method for stochastic optimization[J]. Computer Science, 2014, 30: 1272-1282.

编　辑　税　红