

极客大学机器学习训练营

AutoML 介绍

王然

众微科技 AI Lab 负责人

二〇二一年五月十日

1 AutoML 介绍

2 RL 介绍

3 NAS

4 参考文献

1 AutoML 介绍

2 RL 介绍

3 NAS

4 参考文献

- ▶ AutoML 一词可以指很多领域；
- ▶ 传统来说，AutoML 主要指超参数寻找；
- ▶ 在相当一段时间，AutoML 也指自动特征构建；
- ▶ 在本章中，我们主要介绍 NAS (Neural Architecture Search)；
- ▶ AutoML 的主要问题：优化空间过大 vs. 计算能力有限。

- ▶ 传统模型（例如 LightGBM）的超参数调优；
- ▶ Sequential Bayes（HyperOpt）方法；
- ▶ 实践表明：
 - ▶ 不同的选择方法差异极小（随机搜索 vs. 各种基于随机优化的搜索）；
 - ▶ 根据历史经验所构建的搜索方式比任何复杂方式在相同算力的情况下都更有效果；
- ▶ 关于深度学习的参数搜索研究更少，效果不明。

- ▶ 主要针对表格化数据；
- ▶ 大部分方法的实际效果难以判定，最著名的是谷歌 AutoTabular；
- ▶ 一些研究发现，在一些数据集上，AutoTabular 的效果即使在给予极大算力情况下，仍然不如 XGBoost；
- ▶ 大部分 AutoML 比赛主要依靠规则而非任何算法，但是撰写这些规则过于复杂，不建议提前准备，Ensemble Learning 除外。

- ▶ 目前来看是最有可能起作用的领域；
- ▶ 在过去，这需要大到荒唐的算力，但最近的进展已经使得其可用性大幅度提升；
- ▶ 提升有限，不应该作为最主要的努力方向；
- ▶ 引入随机性的 Ensemble 效果很好；
- ▶ 如何在 NAS 过程中进行有效训练是非常困难的问题，且学术界少有研究。

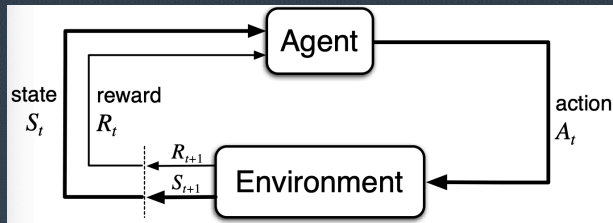
1 AutoML 介绍

2 RL 介绍

■ Q-learning ■ Policy Gradient ■ Dreamer 和其他 RL 算法提升

3 NAS

4 参考文献



- ▶ 在测试集训练；
- ▶ 关注最终精度也关注训练有效性；
- ▶ 由于训练样本的收集跟算法本身（或其他原因）有关，RL 的可复现性极差；
- ▶ 其他训练问题，我们会简要提及。

- ▶ 常见类别：Q-learning、Policy-gradient、Model-based Methods；
- ▶ 样本使用；Online、Offline；
- ▶ Single Agent vs. Multi Agent。

1 AutoML 介绍

2 RL 介绍

■ Q-learning ■ Policy Gradient ■ Dreamer 和其他 RL 算法提升

3 NAS

4 参考文献

- ▶ Action: A_t ;
- ▶ State: S_t ;
- ▶ Reward: R_t ;
- ▶ Cumulative Rewards: $G_t = \sum_{k=0} \gamma^k R_{t+k+1}$;
- ▶ 我们有: $G_t = R_{t+1} + \gamma G_{t+1}$;

- ▶ 我们一般假设 Markov Property;
- ▶ 具体而言, S_t, R_t 的联合概率分布仅取决于 S_{t-1}, A_{t-1} ;
- ▶ 同时我们假设策略是由概率分布 π 给定的, 且策略仅由当前状态决定;
- ▶ 出于简化讨论的目的, 我们假设所有的概率分布都是离散的, 连续的概率分布推导类似, 可在课后尝试。

- ▶ Value Function: $v_{\pi}(s) = \mathbb{E}_{\pi} [G_t \mid S_t = s] = \mathbb{E}_{\pi} [\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s];$
- ▶ Q-Function:
 $q_{\pi}(s, a) = \mathbb{E}_{\pi} [G_t \mid S_t = s, A_t = a] = \mathbb{E}_{\pi} [\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s, A_t = a];$

推导：Value Function 的 Bellman Equation

$$v_{\pi}(s) = \sum_a \pi(a | s) \sum_{s', r} p(s', r | s, a) [r + \gamma v_{\pi}(s')]$$

我们还可以证明以下的等式：

- ▶ $v_{\pi}(s) = \sum_a \pi(a|s) q_{\pi}(s, a);$
- ▶ $q_{\pi}(s, a) = r(s, a) + \gamma \sum_{s'} p(s'|s, a) v_{\pi}(s');$
- ▶ $q_{\pi}(s, a) = \sum_{s'} p(s'|s, a) (r + \gamma \sum_{a'} \pi(a'|s') q_{\pi}(s', a')).$

- ▶ 当 π 为最优策略时，我们有 $q_*(s, a) = \sum_{s'} p(s'|s, a)(r + \gamma \max_{a'} q_*(s', a'))$;
- ▶ 由于不知道实际的动态，无法直接求解；
- ▶ 但是可以利用 Bootstrap 的思想；
- ▶ 具体细节和提升见 Hessel et al. (2018)。

1 AutoML 介绍

2 RL 介绍

■ Q-learning ■ Policy Gradient ■ Dreamer 和其他 RL 算法提升

3 NAS

4 参考文献

- ▶ 为了表示方便，我们用 τ 表示 state 和 action 的联合分布；
- ▶ 同时，我们用 $r(\tau)$ 表示 Cumulated Discounted Returns，为了简便，我们假设 $\gamma = 1$ 并且我们仅仅有有限个时间点
- ▶ $J(\theta) = E_{\tau \sim p_{\theta}(\tau)}[r(\tau)]$ ；
- ▶ 我们下面对 $J(\theta)$ 进行求导。

- ▶ $\nabla_{\theta} J(\theta) = \int \nabla_{\theta} p_{\theta}(\tau) r(\tau) d\tau = \int p_{\theta}(\tau) \nabla_{\theta} \log p_{\theta}(\tau) r(\tau) d\tau = E_{\tau \sim p_{\theta}(\tau)} [\nabla_{\theta} \log p_{\theta}(\tau) r(\tau)]$;
- ▶ 由于 $\nabla_{\theta} \log p_{\theta}(\tau)$ 借助于 Markov Properties 可以进行化简, 所以我们可以得到相当简单的表达式;
- ▶ $\nabla_{\theta} J(\theta) = E_{\tau \sim p_{\theta}(\tau)} \left[\left(\sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(\mathbf{a}_t | \mathbf{s}_t) \right) \left(\sum_{t=1}^T r(\mathbf{s}_t, \mathbf{a}_t) \right) \right]$ 。

- ▶ REINFORCE 算法直接通过 Monte Carlo 方法估计该期望：效果不好；
- ▶ 一个非常流行的架构是 A2C 和 A3C(Mnih et al. 2016)。
- ▶ 大体流程如下 (batch AC):
 - 收集历史 $\{s_i, a_i\}$ 信息；
 - 通过 $\min L(\phi) = \frac{1}{2} \sum_i \|\hat{V}_\phi(s_i) - y_i\|^2$ 拟合 $\hat{V}_\phi(\sim)$ ；这里 $y_{it} = r(s_{i,t}, a_{i,t}) + \hat{V}_\phi(s_{i,t+1})$ ；
 - 计算 Advantage Function $\hat{A}(s_i, a_i) = r(s_i, a_i) + \hat{V}_\phi(s'_i) - \hat{V}_\phi(s_i)$ ；
 - 优化 $\nabla_\theta = \sum_i \nabla_\theta \log \pi_\theta(a_i|s_i) \hat{A}(s_i, a_i)$ ；

- ▶ 是 Model-free 方法中，最为流行的 baseline；
- ▶ 定义 $r_t(\theta) = \frac{\pi_{\theta}(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ ；
- ▶ 目标 $L^{\text{CLIP}}(\theta) = \mathbb{E}_t \left[\min \left(r_t(\theta) \hat{A}_t, \text{clip} (r_t(\theta), 1 - \epsilon, 1 + \epsilon) \hat{A}_t \right) \right]$ ；
- ▶ \hat{A}_t 为 Advantage Function。

1 AutoML 介绍

2 RL 介绍

■ Q-learning ■ Policy Gradient ■ Dreamer 和其他 RL 算法提升

3 NAS

4 参考文献

- ▶ Q-learning 和 Policy Gradient 都被称之为 Model-free 方法；其原因是这两种方法都避免估计具体的转移状态；其主要思想是减少转移状态所带来的问题；
- ▶ 但是在近几年来，Model-based 方法逐渐成为了主流；Model-based 方法估计转移概率、回报等函数，并根据这些信息进行模拟，从而提升 RL 算法的样本效率；
- ▶ 目前认为最好的方法是 Dreamer V2，其中包含了大量的设计技巧，我们将对之进行介绍。

- ▶ 核心思想 A：学习潜在动态，并且根据动态函数进行模拟（dream）；并进一步根据模拟的结果优化策略；
- ▶ 核心思想 B：如果直接对原始的像素级别元素进行学习，其效率过低；目标是得到更简单的状态的表达。

- ▶ Representation Model: $p_{\theta}(s_t | s_{t-1}, a_{t-1}, o_t)$;
- ▶ Observation Model: $q_{\theta}(o_t | s_t)$;
- ▶ Reward Model: $q_{\theta}(r_t | s_t)$;
- ▶ Transition Model: $q_{\theta}(s_t | s_{t-1}, a_{t-1})$;

这其中 o_t 为原始像素，而 s_t 为我们希望能够学习到的表达式；

Hafner et al. (2019) 对于系统的学习建立在以下的损失函数基础上：

- ▶ $\mathcal{J}_{\text{REC}} \doteq \mathbb{E}_p(\sum_t (\mathcal{J}_{\text{O}}^t + \mathcal{J}_{\text{R}}^t + \mathcal{J}_{\text{D}}^t));$
- ▶ $\mathcal{J}_{\text{O}}^t \doteq \ln q(o_t | s_t);$
- ▶ $\mathcal{J}_{\text{R}}^t \doteq \ln q(r_t | s_t);$
- ▶ $\mathcal{J}_{\text{D}}^t \doteq -\beta \text{KL}(p(s_t | s_{t-1}, a_{t-1}, o_t) \| q(s_t | s_{t-1}, a_{t-1})).$

Algorithm 1: Dreamer

Initialize dataset \mathcal{D} with S random seed episodes.
 Initialize neural network parameters θ, ϕ, ψ randomly.
while not converged do
 for update step $c = 1..C$ do
 // Dynamics learning
 Draw B data sequences $\{(a_t, o_t, r_t)\}_{t=k}^{k+L} \sim \mathcal{D}$.
 Compute model states $s_t \sim p_\theta(s_t | s_{t-1}, a_{t-1}, o_t)$.
 Update θ using representation learning.
 // Behavior learning
 Imagine trajectories $\{(s_\tau, a_\tau)\}_{\tau=t}^{t+H}$ from each s_t .
 Predict rewards $E(q_\theta(r_\tau | s_\tau))$ and values $v_\psi(s_\tau)$.
 Compute value estimates $V_\lambda(s_\tau)$ via Equation 6.
 Update $\phi \leftarrow \phi + \alpha \nabla_\phi \sum_{\tau=t}^{t+H} V_\lambda(s_\tau)$.
 Update $\psi \leftarrow \psi - \alpha \nabla_\psi \sum_{\tau=t}^{t+H} \frac{1}{2} \|v_\psi(s_\tau) - V_\lambda(s_\tau)\|^2$.
 // Environment interaction
 $o_1 \leftarrow \text{env.reset}()$
 for time step $t = 1..T$ do
 Compute $s_t \sim p_\theta(s_t | s_{t-1}, a_{t-1}, o_t)$ from history.
 Compute $a_t \sim q_\phi(a_t | s_t)$ with the action model.
 Add exploration noise to action.
 $r_t, o_{t+1} \leftarrow \text{env.step}(a_t)$.
 Add experience to dataset $\mathcal{D} \leftarrow \mathcal{D} \cup \{(o_t, a_t, r_t)_{t=1}^T\}$.

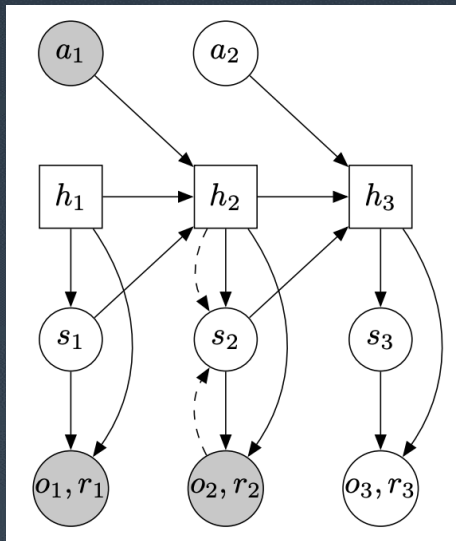
Model components

Representation	$p_\theta(s_t s_{t-1}, a_{t-1}, o_t)$
Transition	$q_\theta(s_t s_{t-1}, a_{t-1})$
Reward	$q_\theta(r_t s_t)$
Action	$q_\phi(a_t s_t)$
Value	$v_\psi(s_t)$

Hyper parameters

Seed episodes	S
Collect interval	C
Batch size	B
Sequence length	L
Imagination horizon	H
Learning rate	α

- ▶ 请注意这里 Value 函数的估计是建立在 imagined trajectories 的基础上的
- ▶ 设 $a_\tau \sim q_\phi(a_\tau | s_\tau)$, 并让 value model 为 $v_\psi(s_\tau) \approx \mathbb{E}_{q(\cdot|s_\tau)} \left(\sum_{\tau=t}^{t+H} \gamma^{\tau-t} r_\tau \right)$. 作者采用的方式是 $a_\tau = \tanh(\mu_\phi(s_\tau) + \sigma_\phi(s_\tau) \epsilon)$, $\epsilon \sim \text{Normal}(0, \mathbb{I})$.
- ▶ 对于 value function, Hafner et al. (2019) 采用的是
$$V_\lambda(s_\tau) \doteq (1 - \lambda) \sum_{n=1}^{H-1} \lambda^{n-1} V_N^n(s_\tau) + \lambda^{H-1} V_N^H(s_\tau),$$
$$V_N^k(s_\tau) \doteq \mathbb{E}_{q_\theta, q_\phi} \left(\sum_{n=\tau}^{h-1} \gamma^{n-\tau} r_n + \gamma^{h-\tau} v_\psi(s_h) \right), \quad h = \min(\tau + k, t + H),$$
$$V_R(s_\tau) \doteq \mathbb{E}_{q_\theta, q_\phi} \left(\sum_{n=\tau}^{t+H} r_n \right);$$
- ▶ 对于策略函数的估计, Hafner et al. (2019) 并没有采用典型的 Policy Gradient 损失, 而是直接对损失进行优化。



- ▶ 主要创新：将潜在表示转换成为了离散的表示；
- ▶ 并介绍了其他的一些非常有效的提升方法。

- ▶ Recurrent model: $h_t = f_\phi(h_{t-1}, z_{t-1}, a_{t-1})$;
- ▶ Representation model: $z_t \sim q_\phi(z_t | h_t, x_t)$;
- ▶ Transition predictor: $\hat{z}_t \sim p_\phi(\hat{z}_t | h_t)$;
- ▶ Image predictor: $\hat{x}_t \sim p_\phi(\hat{x}_t | h_t, z_t)$;
- ▶ Reward predictor: $\hat{r}_t \sim p_\phi(\hat{r}_t | h_t, z_t)$;
- ▶ Discount predictor: $\hat{\gamma}_t \sim p_\phi(\hat{\gamma}_t | h_t, z_t)$ 。

Algorithm 1: Straight-Through Gradients with Automatic Differentiation

```
sample = one_hot(draw(logits))           # sample has no gradient
probs  = softmax(logits)                  # want gradient of this
sample = sample + probs - stop_grad(probs) # has gradient of probs
```

$$\mathcal{L}(\phi) \doteq \mathbb{E}_{q_{\phi}(z_{1:T} \mid a_{1:T}, x_{1:T})} \left[\sum_{t=1}^T \underbrace{-\ln p_{\phi}(x_t \mid h_t, z_t)}_{\text{image log loss}} \underbrace{-\ln p_{\phi}(r_t \mid h_t, z_t)}_{\text{reward log loss}} \underbrace{-\ln p_{\phi}(\gamma_t \mid h_t, z_t)}_{\text{discount log loss}} \right. \\ \left. \underbrace{+\beta \text{KL}[q_{\phi}(z_t \mid h_t, x_t) \parallel p_{\phi}(z_t \mid h_t)]}_{\text{KL loss}} \right]. \quad (2)$$

Algorithm 2: KL Balancing with Automatic Differentiation

```
kl_loss =      alpha * compute_kl(stop_grad(approx_posterior), prior)
            + (1 - alpha) * compute_kl(approx_posterior, stop_grad(prior))
```

- ▶ Dreamer V2 整体网络架构和 V1 类似，在这里我们省略；
- ▶ 对于策略函数的损失函数有所不同： $\mathcal{L}(\psi) \doteq \mathbb{E}_{p_\phi, p_\psi} \left[\sum_{t=1}^{H-1} \left(-\rho \ln p_\psi(\hat{a}_t | \hat{z}_t) \text{sg}(V_t^\lambda - v_\xi(\hat{z}_t)) - (1 - \rho) V_t^\lambda - \eta H[a_t | \hat{z}_t] \right) \right]$ 。

- ▶ RL 算法的训练非常困难；
- ▶ 最重要的两点：Reward 设计（非 Sparse、有区分度），初始化（Imitation Learning）；
- ▶ 一些训练细节见[该教程](#)。

- ▶ Prioritized Experience Replay;
- ▶ Exploration;
- ▶ Self-imitation Learning;
- ▶ Data Augmentation;
- ▶ Encoder Refinement;
- ▶ Multi-task training;
- ▶ Network Design。

1 AutoML 介绍

2 RL 介绍

3 NAS

■ 设计选择 ■ ReNAS ■ Gumbel-Softmax Trick 和 Entmax

4 参考文献

1 AutoML 介绍

2 RL 介绍

3 NAS

■ 设计选择 ■ ReNAS ■ Gumbel-Softmax Trick 和 Entmax

4 参考文献

- ▶ Search Space;
- ▶ Search Strategy;
- ▶ Performance Estimation Strategy。

- ▶ 对于大部分网络来说，Search Space 搜索的是 Cell 或者 Block Level；
- ▶ Cell 或者 Block 之间采用常用的方法（例如 Residual Connection）进行连接；
- ▶ Cell 内部在文献当中采取的经常是不同的卷积和池化层，为了保证结构可以进行链接，通常需要通过 padding 或者其他方式保证输入输出的相等。

- ▶ 理想状况下，对于每一个新的网络架构，我们均应该从头进行训练；
- ▶ 这里的消耗是早期 NAS Paper 不可行的主要原因；
- ▶ Performance Estimation Strategy 常常和 Search Strategy 高度相关；
- ▶ 一些常见方法：Lower Fidelity Estimation、Learning Curve Extrapolation、Network Morphisms、Weight Sharing。

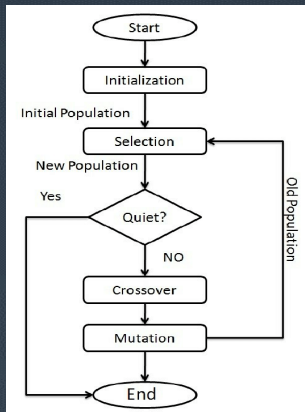
1 AutoML 介绍

2 RL 介绍

3 NAS

■ 设计选择 ■ ReNAS ■ Gumbel-Softmax Trick 和 Entmax

4 参考文献



- ▶ 遗传算法是相当一般的，并且被证明是广为成功的解决组合优化问题的启发式算法；
- ▶ 遗传算法和增强学习的组合主要目的是：
 - ▶ 使用遗传算法避免增强学习不稳定的问题；
 - ▶ 使用增强学习提升遗传算法突变的效率。
- ▶ 在 NAS 的应用见Chen et al. (2019)。

1 AutoML 介绍

2 RL 介绍

3 NAS

■ 设计选择 ■ ReNAS ■ Gumbel-Softmax Trick 和 Entmax

4 参考文献

- ▶ Gumbel 分布的 PDF 为 $e^{-(x+e^{-x})}$ ，其 CDF 为 $e^{-e^{-x}}$ ；
- ▶ Gumbel 分布的一个最大的应用是将不可微分的问题转化成可微分的问题，这个就是所谓的 Gumbel-max trick；
- ▶ 我们下面对之进行一个简单的证明。

- ▶ 假设 G_1, G_2, \dots, G_k 服从 Gumbel 分布;
- ▶ 假设 $\alpha_1, \alpha_2, \dots, \alpha_k$ 为 0 到 1 之间的数, 并且 $\sum_i \alpha_i = 1$;
- ▶ 设 $Z = \arg \max_k \{\log \alpha_k + G_k\}$;
- ▶ 则 $\mathbb{P}(Z = k) = \alpha_k$ 。

$$\begin{aligned} P(Z = k) &= P(u_k \geq u_j, \forall j \neq k) \\ &= \int_{-\infty}^{\infty} P(u_k \geq u_j, \forall j \neq k \mid u_k) p(u_k) du_k \\ &= \int_{-\infty}^{\infty} \prod_{j \neq k} P(u_k \geq u_j \mid u_k) p(u_k) du_k \\ &= \int_{-\infty}^{\infty} \prod_{j \neq k} e^{-e^{-u_k + \log \alpha_j}} e^{-\left(u_k - \log \alpha_k + e^{-(u_k - \log \alpha_k)}\right)} du_k \end{aligned}$$

$$\begin{aligned}P(Z = k) &= \int_{-\infty}^{\infty} e^{-\sum_{j \neq k} \alpha_j e^{-u_k}} \alpha_k e^{-(u_k + \alpha_k e^{-u_k})} du_k \\&= \alpha_k \int_{-\infty}^{\infty} e^{-u_k - (\alpha_k + \sum_{j \neq k} \alpha_j) e^{-u_k}} du_k \\&= \alpha_k\end{aligned}$$







- ▶ 在 NAS 的应用见Dong and Yang (2019);
- ▶ 注意, 可以通过 Entmax 改进 Softmax, 具体方法见Peters, Niculae, and Martins (2019)。

1 AutoML 介绍

2 RL 介绍

3 NAS

4 参考文献

-  Chen, Yukang et al. (2019). “Renas: Reinforced evolutionary neural architecture search”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4787–4796.
-  Dong, Xuanyi and Yi Yang (2019). “Searching for a robust neural architecture in four gpu hours”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1761–1770.
-  Hafner, Danijar et al. (2019). “Dream to control: Learning behaviors by latent imagination”. In: *arXiv preprint arXiv:1912.01603*.
-  Hafner, Danijar et al. (2020). “Mastering atari with discrete world models”. In: *arXiv preprint arXiv:2010.02193*.
-  Hessel, Matteo et al. (2018). “Rainbow: Combining improvements in deep reinforcement learning”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32. 1.
-  Mnih, Volodymyr et al. (2016). “Asynchronous methods for deep reinforcement learning”. In: *International conference on machine learning*. PMLR, pp. 1928–1937.



Peters, Ben, Vlad Niculae, and André FT Martins (2019). “Sparse sequence-to-sequence models”. In: *arXiv preprint arXiv:1905.05702*.



Schulman, John et al. (2017). “Proximal policy optimization algorithms”. In: *arXiv preprint arXiv:1707.06347*.