强化学习基础

极客时间

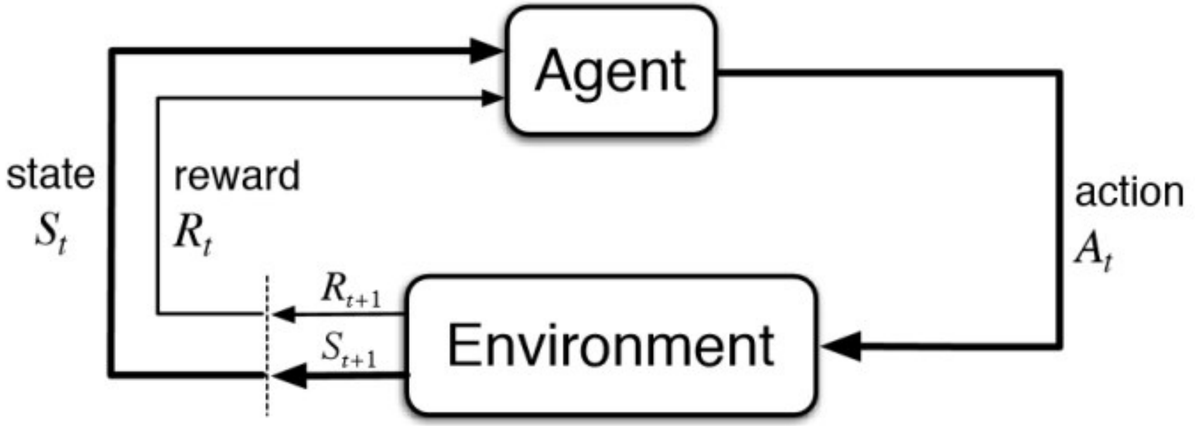
扫码试看/订阅

《NLP 实战高手课》视频课程

强化学习基础

- 强化学习设定
- 强化学习和一般的优化算法的关系
- Single-agent和Multi-agent强化学习

强化学习设定



强化学习设定：马尔科夫性

强化学习和优化算法

- 强化学习是一种优化算法
  - ✓ 动态
  - ✓ 非参数
  - ✓ 基于神经网络

Single-agent和Multi-agent强化学习

- Single-Agent
  - ✓ 对人类难以决策（高频）的问题进行优化
  - ✓ 应用：AutoML
- Multi-agent
  - ✓ 通过去中心化的方法，构建复杂的强化学习系统
  - ✓ 应用：游戏AI，交通优化，推荐系统

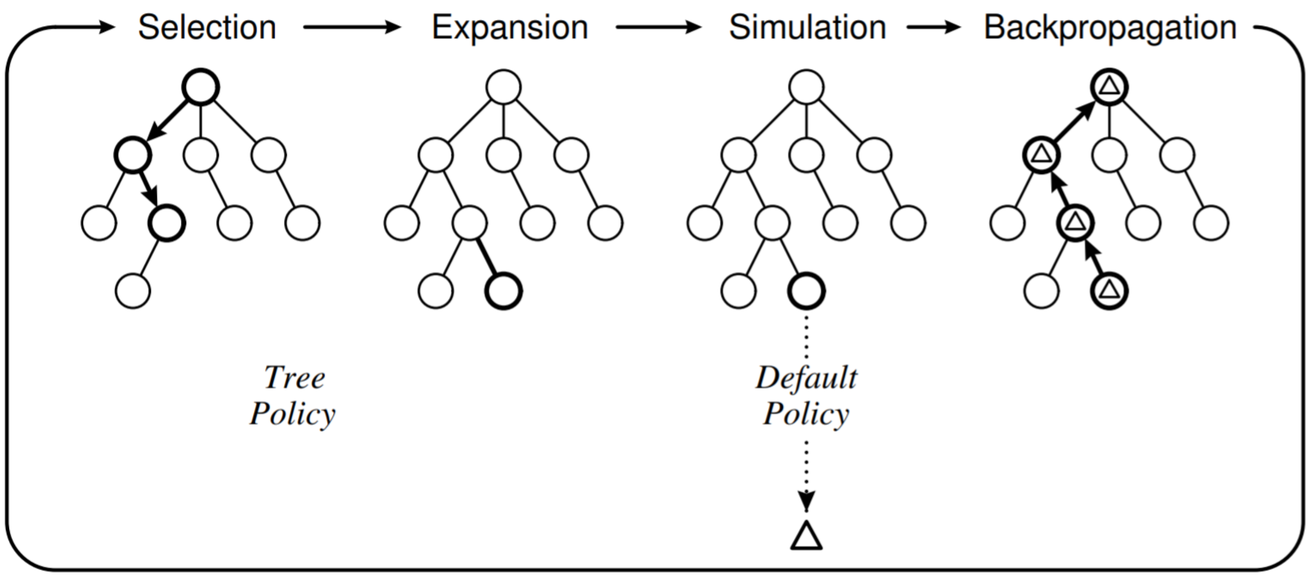
MCTS 简介：如何将“推理”引入到强化学习框架中

MCTS 简介：如何将“推理”引入到强化学习框架中

- 游戏 AI 中的“直觉”和“推理”
- MCTS 的基本方法
- 一些提升 MCTS 的方法

游戏 AI 中的“直觉”和“推理”

MCTS 的基本方法


Selection
Expansion
Simulation
Backpropagation
Tree Policy
Default Policy

MCTS 的一些改进

- 采用深度神经网络而不是纯粹模拟
- 采用更有效率的搜索
- “连续可微”的“MCTS”

AutoML 及 Neural Architecture Search 简介

AutoML 及 Neural Architecture Search 简介

- AutoML 问题简介
- Neural Architecture Search 问题分解

AutoML 简介

- 定义：任何自动化的机器学习过程均可以称为 AutoML
- 常见的领域：
  - 超参数选择
  - 基于经验的建模 pipeline 组合
  - Neural Architecture Search

Neural Architecture Search 问题分解

- 核心问题：计算效率
- 常见子问题：
  - Search Space
  - Search Strategy
  - Performance Estimation Strategy

Search Space

- 构建神经网络的零件
- 不同粒度
- 常常借用成熟的框架
- 搜索 cell 而不是整个模块

Search Strategy

- 本质上来说，NAS 是组合优化问题
  - 任何求解组合优化的方法都可以应用于之上
- 如何将 NAS 问题 formulate 成为组合优化问题很关键

Performance Estimation Strategy

- 作为组合优化的信号
- 一些常见方法
  - Lower Fidelity Estimates
  - Learning Curve Extrapolation
  - Network Morphism
  - One-shot Search

AutoML 网络架构举例

AutoML 网络架构举例

- Swiss
- NASNet
- Evolved Transformer

Swiss

- Paper： https://arxiv.org/pdf/1710.05941.pdf

NASNet

- Paper: https://openaccess.thecvf.com/content_cvpr_2018/papers/Zoph_Learning_Transferable_Architectures_CVPR_2018_paper.pdf

Evolved Transformer

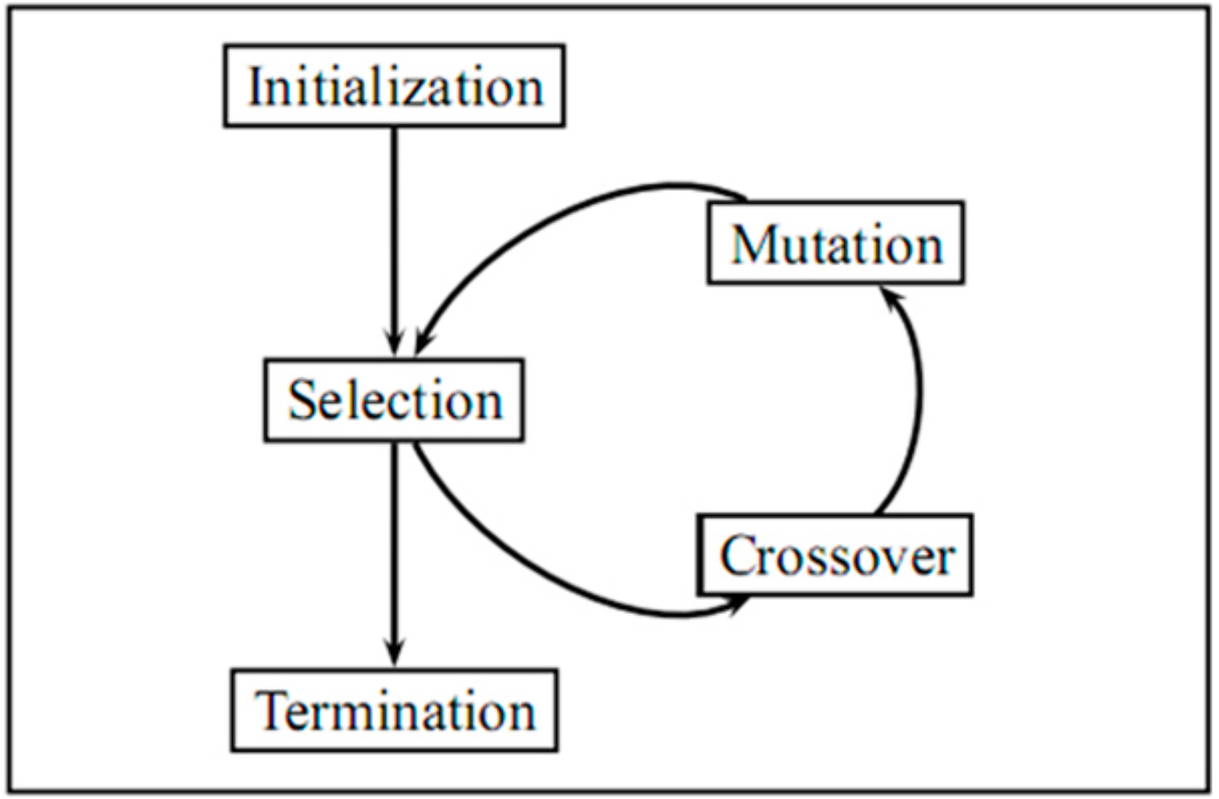
- Paper：https://arxiv.org/pdf/1901.11117.pdf

RENAS：如何结合遗传算法和增强学习

RENAS

- 遗传算法回顾
- 遗传算法和增强学习之争
  - NAS 的子战场
- 是否可以两者和睦共存

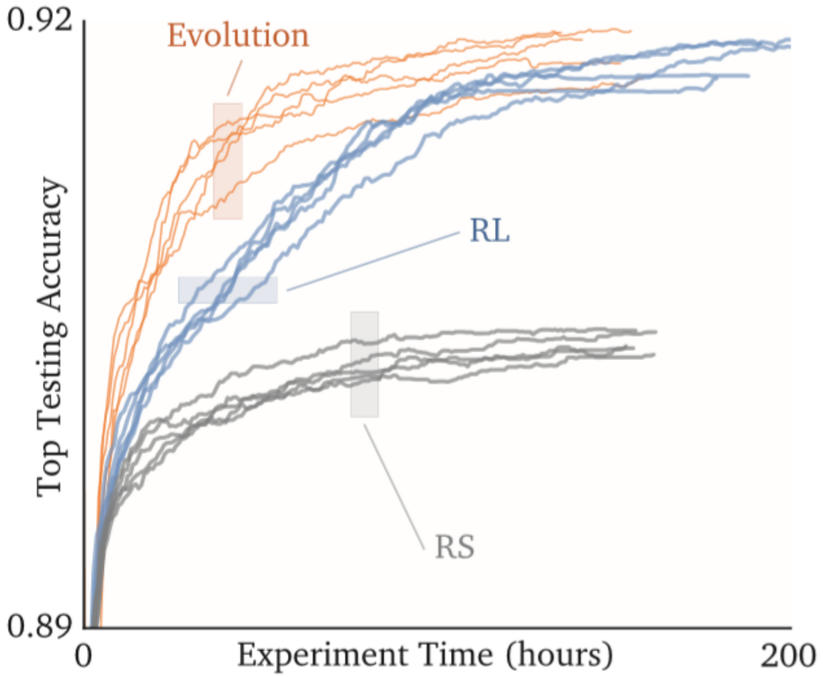
遗传算法回顾


Initialization
Mutation
Selection
Crossover
Termination

遗传算法和增强学习之争

- 增强学习和遗传算法均被应用于 NAS 中，并均取得了不错的效果
- 究竟何种效果最好并没有定论

Regularized Evolution



RENAS基本算法

- https://openaccess.thecvf.com/content_CVPR_2019/papers/Chen_RENAS_Reinforced_Evolutionary_Neural_Architecture_Search_CVPR_2019_paper.pdf

补充材料

- 关于遗传算法和其他类似算法的总结可见：*An Introduction to Metaheuristics for Optimization*

Hierarchical Search

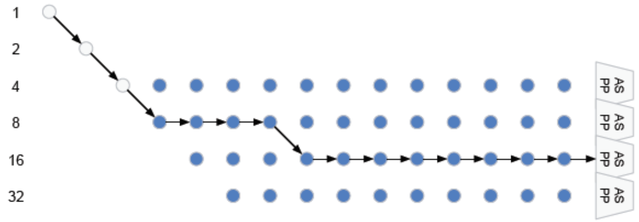
Hierarchical Search

- Motivation
- 基本方法

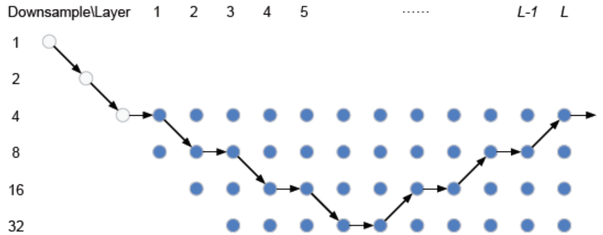
Motivation

- 绝大多数 NAS 方法都对 cell 内结构进行搜索
- 对标准问题（尤其是分类）来说，cell 的组合是固定的
- 对于非标准问题，如何组合 cell 是一个问题

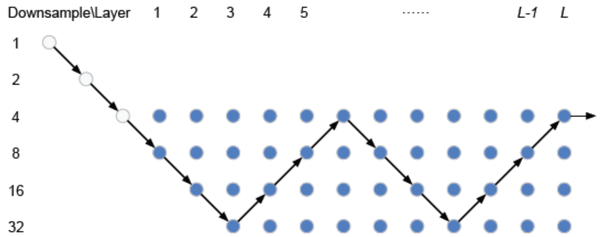
基本思路



(a) Network level architecture used in DeepLabv3 [9].



(b) Network level architecture used in Conv-Deconv [56].



(c) Network level architecture used in Stacked Hourglass [55].

LaNAS：如何搜索 Action Space

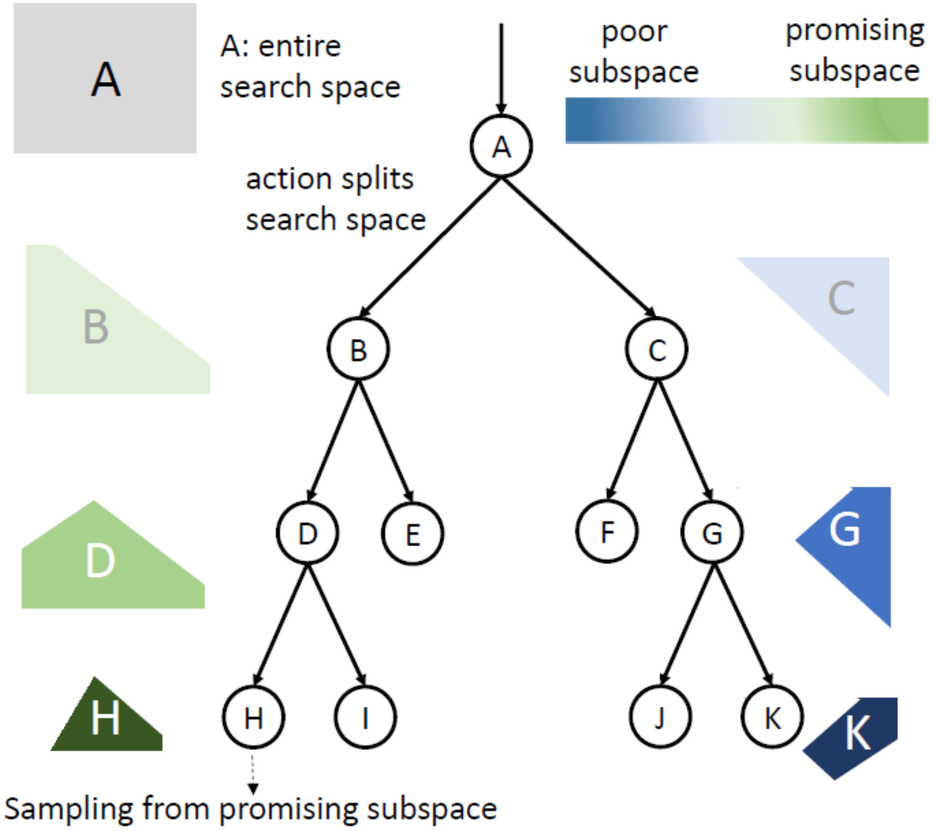
LaNAS：如何搜索 Action Space

- Motivation
- 基本思想
- 实现细节

Motivation

- 不同的 NAS 文献方法不同
  - Search Strategy 不同（遗传算法，RL，Differentiable Optimization）
  - 构建不同
  - 采用的基础构成不同（卷积，自注意力，全连接……）
  - ……
- 过小的 Action Space 导致错过最优；过大的 Action Space 导致收敛缓慢
- 很多 Action 从来都不会被采用
- 是否可以学习一个 Action Space 呢

核心思想


A
A: entire
search space
poor
subspace
promising
subspace
A
action splits
search space
B
C
B
C
D
E
F
G
D
G
H
I
J
K
H
K
Sampling from promising subspace

实现细节

- https://arxiv.org/pdf/1906.06832.pdf

超参数搜索：如何寻找算法的超参数

超参数搜索

- 超参数寻找的基本设定
- 超参数寻找的方法
- 实际建议
- Bayesian Optimization 为基础的优化

超参数寻找的基本设定

- 特指在训练过程开始前的参数
  - 一般用于结构化数据；尤其是非深度学习模型
- 为什么不用于深度学习模型
  - 穷
  - 深度学习模型算法的动态性

超参数寻找的基本算法

- Grid Search
- Random Search
- Bayesian Search

实践建议

- 区分重要和非重要变量
- 区分互相之间有交互影响的变量
- 有很多变量最好采用 grid search 来寻找
- 但也有很多变量影响很小
- 很多时候超参数寻找仅仅在最终建模阶段再使用比较复杂的方法

Bayesian Optimization 的优化

- Bayesian 分析基本想法
- Bayesian Optimization 基本流程
- 实现：https://github.com/hyperopt/hyperopt

Learning to Optimize

Learning To Optimize

- 超参数优化和 Learning To Optimize 的区别
- 一些初步的实验结果

超参数优化和 Learning to Optimize 的区别

- 为什么超参数优化方法不适合用于深度学习炼丹
  - 超参数方法大部分决定了训练开始时候的超参数
  - 深度学习训练过程的参数有些需要动态指定

核心思想

- Motivation：Learning to Optimize：https://arxiv.org/pdf/1606.01885.pdf
- 问题：从头学习一个优化器问题过难
- 突破口：
  - 是否可以仅仅学习优化器的一些参数
  - 优化过程可逆：效果不好即可回炉重练

基本思路

- 目标：在基本参数确定的范围内，对最终结果做稍许提升
- 步骤：
  - 确定基本合适的参数
  - 将模型训练到基本稳定
  - 采用 MCTS 对关键参数（learning-rate，label smoothing weight 等）进行周期性调整

基本结果

- 有提升空间
- 为什么采用 MCTS
  - Reinforcement Learning 不稳定，学习速率慢，试错成本太高
  - Sequential Bayesian 难以找到合适的先验和后验
- 核心问题：
  - Baseline 是什么：一个有无限算力的专家么？
  - 怎么利用反馈数据（Direct Policy Gradient？）
  - 如何收集更多数据并使用
  - 多起点问题
  - 如何减少训练成本

遗传算法和增强学习的结合

遗传算法和增强学习的结合

- 遗传算法和增强学习算法的缺点
- 结合方式

增强学习算法的缺点

- 不 WORK！
- 不稳定
- 对超参数敏感
- 探索问题

遗传算法的缺点

- 弱信号
- 学习过程慢
- 对于超大空间探索慢

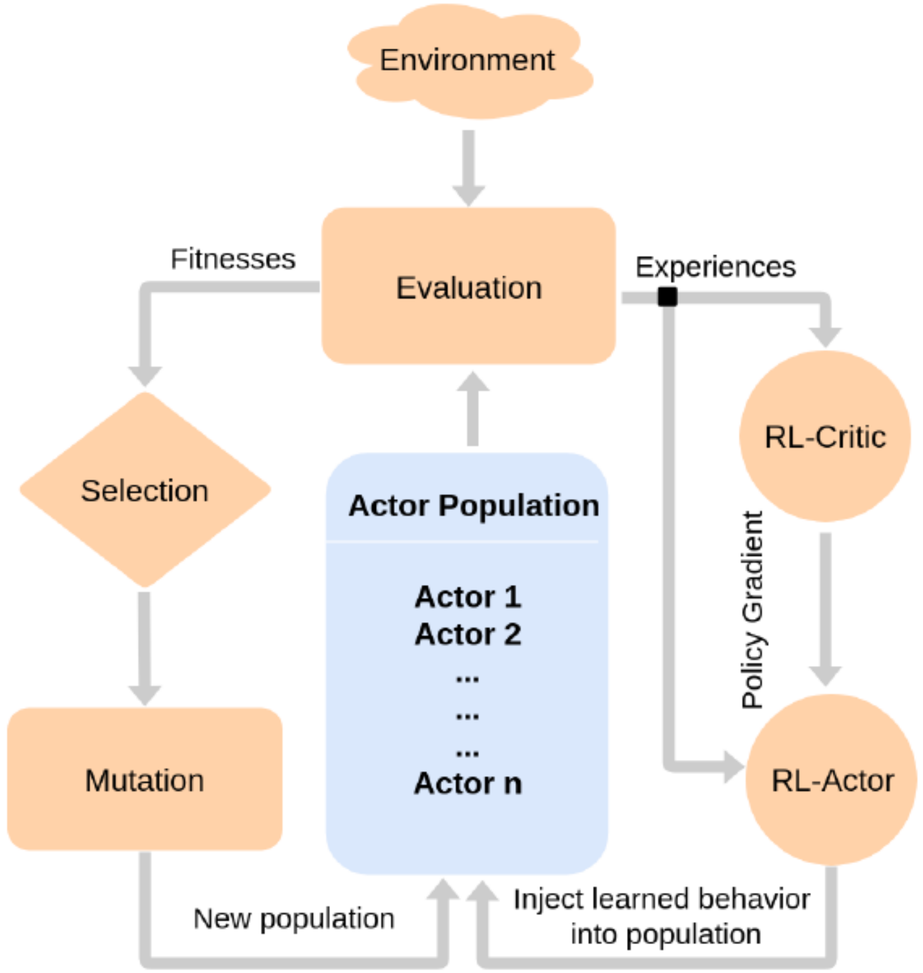
核心思路

- 不要指望 RL 每次都 work
- 但是如果只要有一个可以 work 就可以，那么问题就变得简单很多
  - 超参数寻找
  - 网络架构搜索
  - 优化过程
- 不包括自动驾驶等

结合方法：将 RL 作为 Mutation 的工具

- 核心解决问题：对于超大空间的探索
- RENAS 当中实现

结合方法：Evolutionary Reinforcement Learning



多代理增强学习设定

多代理增强学习设定

- 基本思想
- 数学形式

基本思想

- 自然界中大量的任务是由多个 Agent 联合起来完成的
- 理论上来说我们可以采用 Centralized Policy
  - 问题：Action Space 过大
- Information Sharing

早期例子

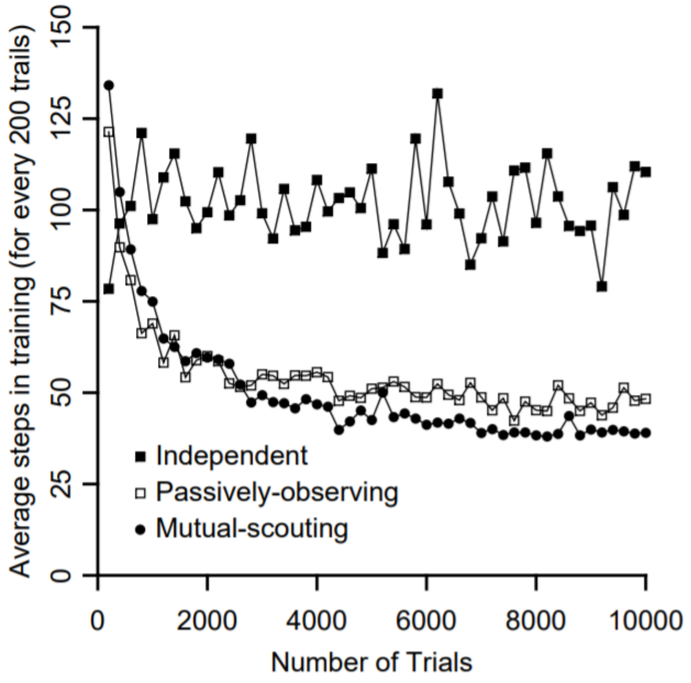


Figure 7: Typical runs for the 2-prey/2-hunter joint task.

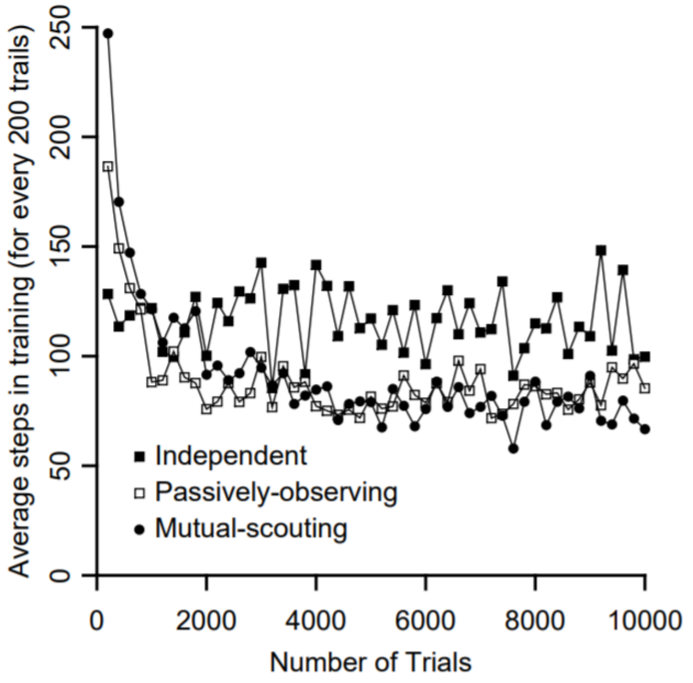


Figure 8: Typical runs for the 1-prey/2-hunter joint task.

数学形式

整体概况

- 最著名的应用：游戏 AI
  - 例子：AlphaStar
- 学术上可以有很多 idea
- 问题：
  - 一切 Single-Agent RL 算法的问题
  - Instability
  - Communication and Cooperation

AlphaStar 介绍

AlphaStar 介绍

- 基本任务
- 训练流程
- 网络设计
- 实验结果

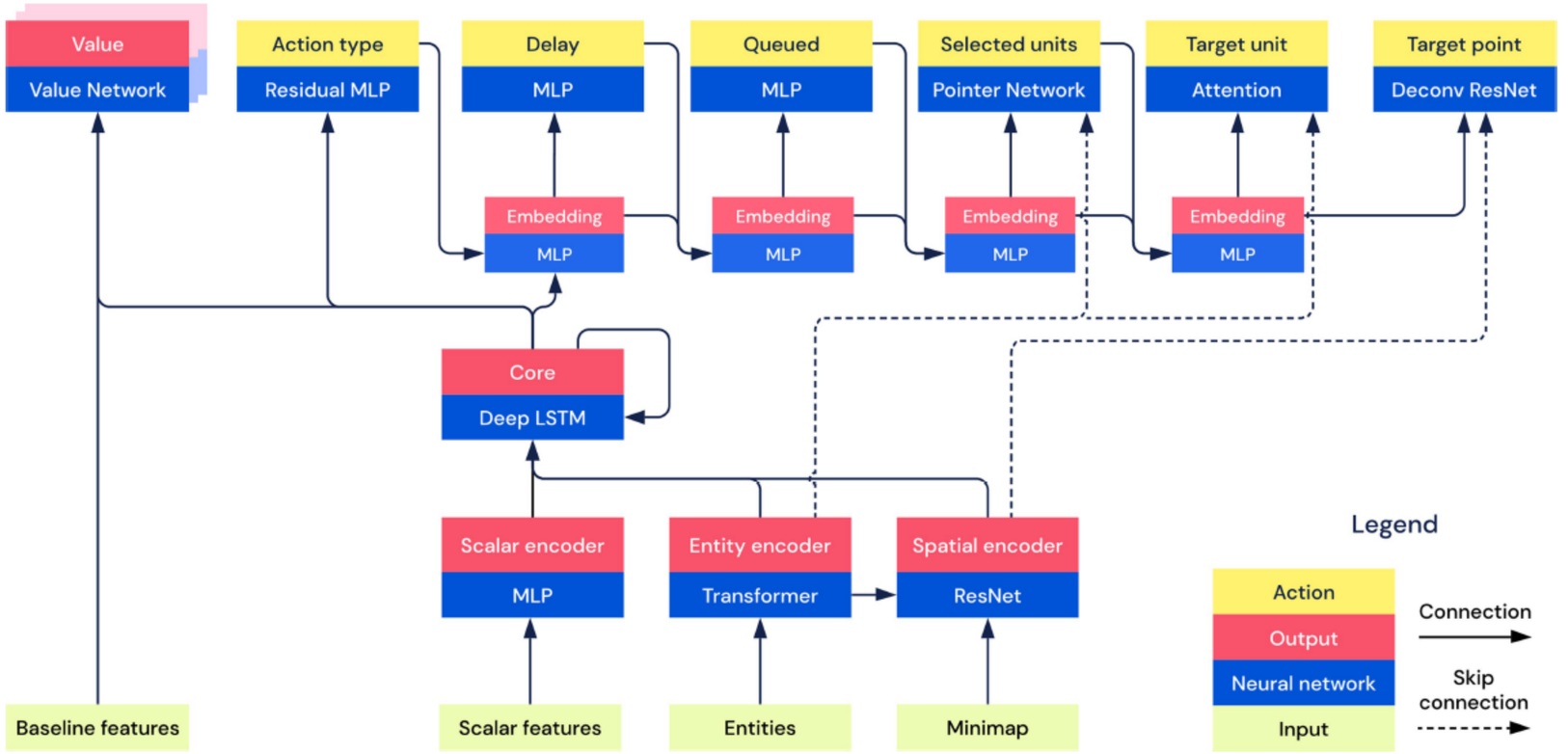
基本任务

- 基于游戏“星际争霸2”
- 核心挑战：巨大的 Action Space（每步约 10 的 26 次方的选择）
- 目标：
  - 尽可能以类似于人的方法击败对手，不依赖于人为规则
  - 以创新的策略击败对手
- 重要性：DeepMind 当时最主要的项目

基本训练流程

- 以模仿学习开始
- AlphaStar League
  - 主要的 Agent 需要应付各类的策略
  - 一部分 Agent 以 Cheese 为主
  - 一部分 Agent 不必考虑 Cheese 的威胁，所以可以专心考虑策略性的问题

网络设计


Value
Value Network
Action type
Residual MLP
Delay
MLP
Queued
MLP
Selected units
Pointer Network
Target unit
Attention
Target point
Deconv ResNet
Embedding
MLP
Embedding
MLP
Embedding
MLP
Embedding
MLP
Core
Deep LSTM
Scalar encoder
MLP
Entity encoder
Transformer
Spatial encoder
ResNet
Baseline features
Scalar features
Entities
Minimap
Legend
Action
Output
Neural network
Input
Connection
Skip connection

实验结果

- 烧钱机器：据估计耗费 100 万英镑
- 击败了几名著名的选手
  - 但是没有击败当时最顶尖的选手
- 得到了一些比较新的战术
  - 但是战术中充满的低级错误
- APM 和职业选手类似
  - 但是核心的一些操作效率可能更高（经济建设）

IMPALA：多 Agent 之间的 Actor-Critic 算法

IMPALA

- 核心问题
- 基本想法
- 数学形式

核心问题

- 如何进行 offline 学习？
- Actor 和 Critic 策略函数可能不同
- 理想状况下，进行频繁更新
- 实际情况，希望 Actor 可以长期和 Critic 的策略有一定偏差

基本想法

- 可以按照正常的 A3C 算法进行
- 但是：
  - 假设 Actor-Critic 之间策略函数分布较大，这部分观测应该对 V 函数的更新有更大的作用

数学形式

COMA：如何解决 Agent 之间的交流问题

AlphaStar 介绍

- 基本问题
- 解决方式
- COMA 概述

基本问题

- 多 Agent 的增强学习未必比单 Agent 效果好
- 很多情况可以归结为 Agent 之间的沟通和协调问题

解决方式

- 独立的学习
- Weight Sharing
- 各种疯狂的解决方式
- COMA

COMA概述

- 核心思想：每个 Agent 应该关心自己在系统中的贡献
- 核心公式：

$$A^a(s, \mathbf{u}) = Q(s, \mathbf{u}) - \sum_{u'^a} \pi^a(u'^a|\tau^a) Q(s, (\mathbf{u}^{-a}, u'^a)).$$

多模态表示学习

多模态表示学习简介

- 问题设定
- 基本原则
- 简单拼接方法
- Auto-Encoder

问题设定

- 很多问题数据源以多种形式出现：
  - 例：新闻中包含文本（标题，正文）、基础信息和图片信息
- 每个数据源的习惯性建模方式不一样
  - 结构化数据：树模型，MLP，xDeepFM
  - 文本：RNN，Transformer
  - 图像：CNN（ResNet，DenseNet 等）
- 如何将多个数据源结合

基本原则

- 对于大多数问题，某种模态将会是最重要的
  - 先检查加入其它模态数据后是否有助于人的判断
- 没有通用的多模态方法
  - 这里介绍的更多是一些标准问题引入多模态数据的 trick
- 注意模型的训练程度
  - 直接联合训练可能会导致模型崩溃

直接拼接

- 常见拼接方法
  - 全连接层
  - 外积
- 问题：希望不同模态之间能够保持一定的独立性（对问题贡献不同的角度）
- 通用解决方法：加入 Reconstruction Loss

Auto Encoder

- Shared Representation 接入到不同的 Reconstruction Loss
- 不一定所有 Reconstruction Loss 都具备
  - Tabular 和图像的 Reconstruction Loss 往往比文本要好
- 其他 Auto-encoder 的 trick 也适用

知识蒸馏

知识蒸馏

- 目标
- 设定
- 训练目标

目标

- 基本：加速模型部署
  - 节省算力
  - 节省内存/显存消耗
- 其他可能作用
  - 减少推断方差
  - 对抗 Adversarial Attack

基本设定

- Teacher/Student Network
- Teacher Network 应该是已经进行充分调优的（Offline Distillation）
- Student Network 理论上可以是任何形式的网络
  - 但通常与 Teacher Network 一致

训练目标

- Soft Target 为最常用的训练目标
- 中间表示层的差别也可以进行
- 其他 Dark Knowledge（需要 Case-by-case 进行分析）
  - 训练 Embedding 时可以将两两之间的相似度作为训练目标

DeepGBM

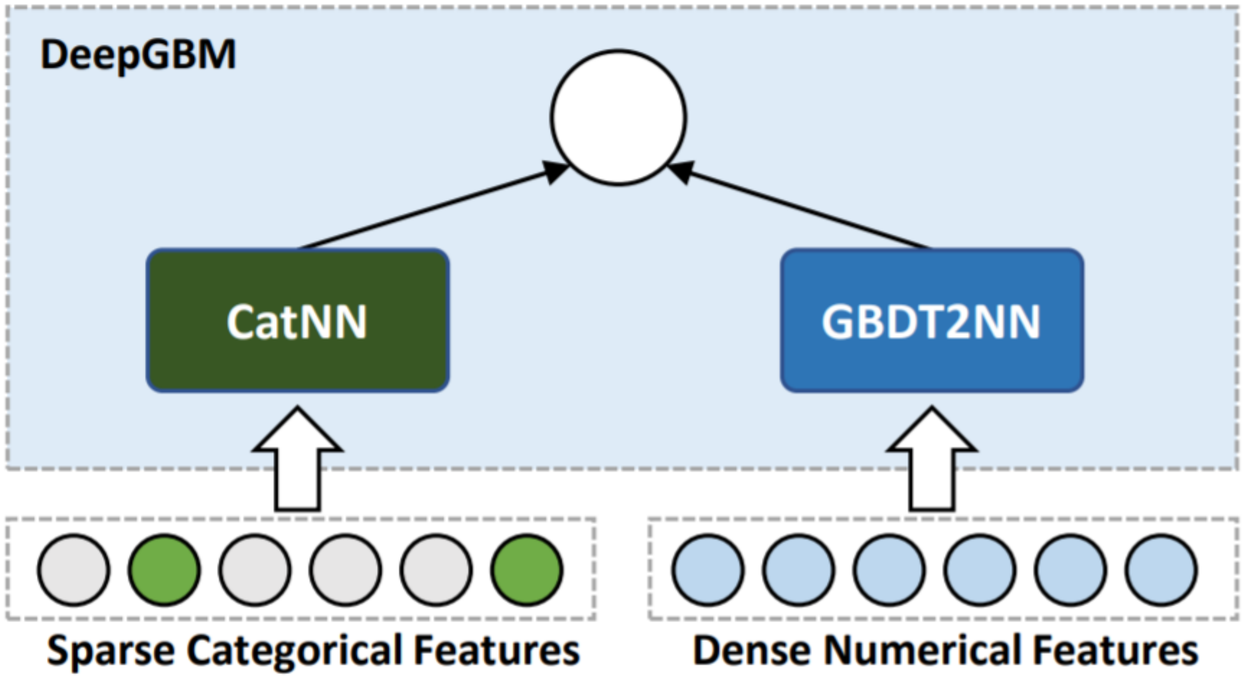
DeepGBM

- 问题
- 基础架构
- 训练目标

为何要将集成树模型变为神经网络模型

- 很多算法框架只能接受神经网络模型
  - 多模态学习
  - 增强学习
- 但是集成树模型在处理结构化的表现上（远远）优于神经网络

基础架构


DeepGBM
CatNN
GBDT2NN
Sparse Categorical Features
Dense Numerical Features

训练目标

- 核心思想：将树进行 Group，然后学习 Leaf Embedding
- Leaf Embedding 学习过程：
  - 使用 Leaf Index 进入全连接层，预测 Leaf Values
  - 使用 Embedding 后得到的结果作为学习目标

文本推荐系统和增强学习

文本推荐系统和增强学习

- 文本推荐系统面临的问题
- 增强学习和文本推荐系统结合的机遇
- 一些研究

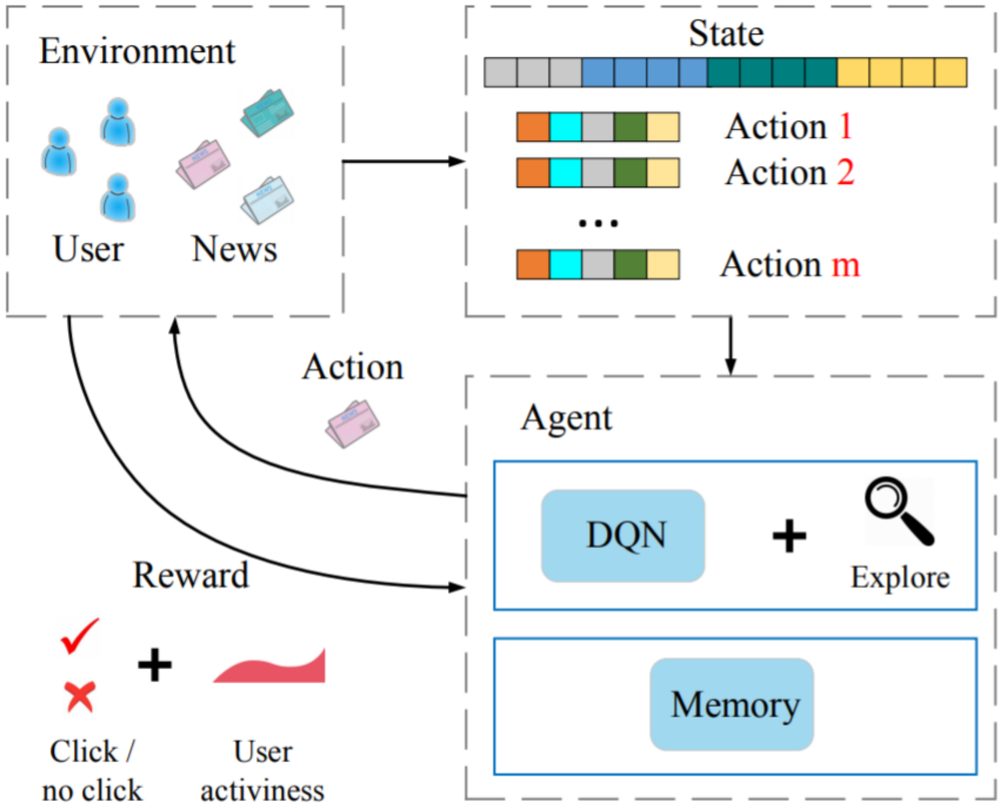
文本推荐系统面临的问题

- 冷启动
- 评估问题->用户粘性
- 探索问题（关于类似文章的推荐）
- 兴趣漂移

增强学习和文本推荐系统结合的机遇

- 冷启动->难以彻底解决，但是有系统性的探索方式总是好的
- 评估问题->RL 可以对各种 reward 进行学习
- 探索问题->可以借用 RL 中的探索方法

DRN


Environment
User
News
State
Action 1
Action 2
...
Action m
Action
Agent
DQN
+
Explore
Memory
Reward
+
Click /
no click
User
activiness

Personalized Attention

- https://arxiv.org/pdf/1907.05559.pdf

不靠谱的增强学习

不靠谱的增强学习

- 自杀的 Agent
- 不稳定的结果
- 敏感的性格
- 数据利用效率低下
- 巨大的实验成本
- Reward 的设计问题
- 这么不靠谱，还有价值研究么？

自杀的 Agent

- 多个故事版本
- 共性：Agent 学到一个速死的策略，以避免最糟糕的情况
- 通常问题根源：Reward 设计

不稳定的结果

- 大部分我们介绍的 Deep Learning 方法都很大概率不会导致整体崩溃的情况
  - RL 不一定
- 一切不变的情况下，仍然有相当大的概率可以失败
  - 30% 左右的失败率可能在 RL 实验中是可以接受的
- 这意味着：
  - 如何比较结果成为一个难题
  - 对于不能接受失败的系统（比如说自动驾驶），RL 落地极难
  - 对于Long Running 系统（尤其是有可能有状态改变的情况），RL 很难避免运维（如果运维能解决问题）

敏感的性格

- RL 结果可能跟各种参数有关系
  - 环境
  - 算法
  - Reward 设计
  - 各种超参数
  - Scale
  - 优化器
  - 初始化
  - ？？？

数据利用效率低下

- RL 算法对于数据利用臭名昭著的低下
- 可能是由于问题过难（从 pixel 得到战略）
- 但是这对于大部分不能通过模拟得到的问题很有可能是灾难性的
- 对于环境有变化的情况（比如说 Multi-agent RL），结果可能更加糟糕

巨大的实验成本

- RL 的实验成本可能由多个方面造成
  - 问题本身造成（自动驾驶）
  - 充分的探索需求（AlphaStar）
  - 失控的可能性

Reward 设计问题

- RL 结果对 Reward 设计十分敏感
  - RL 希望有信息的 Reward
- Reward 的稀疏问题很难解决

这么不靠谱，还有价值研究么？

- 有些任务不需要 RL 表现很好也可有可接受结果
- RL 的实现和实验有一定的 trick
- 一些情况可以根据策略进行补充
- RL 本身的进展近年来是迅速的

增强学习算法的实践注意事项

增强学习的算法实践事项

- 代码实现
- 记录
- 实验

代码实现

- 由于增强学习的不稳定性，尽量不要自己从头实现；
  - Github 上有大量的实现
- 采用一些小的单元测试
  - 比如说 OpenAI Gym
  - 包括已实现的代码（Github 上的代码未必没有 bug）
  - 不一定每次都成功；不一定所有问题都成功；但如果简单问题始终不成功，那基本可以考虑实现错误

记录

- 记录一切算法进行中出现的情况
  - 尤其注意截断发生的情况
- 记录 Policy 中 Entropy 的变化
  - 过小的 Entropy 意味着非随机的政策->出现了退化的问题
  - 过大的 Entropy 意味着纯随机的政策
- 注意 Policy 的变化（KL 距离）
- 所有记录都应该存在一个类当中（而不是打印在屏幕上）

实验

- 反复实验是必须的
- 关注超参数的影响
  - 尽可能减少超参数的影响
  - 如果稍微超参数变化一下影响就很大的话
- 初始值可能起到很大作用
- Be Patient

TRPO 和 PPO

TRPO 和 PPO

- TRPO
- PPO

Trust-Region Policy Optimization

- 基本数学形式

$$\begin{aligned} \underset{\theta}{\text{maximize}} \quad & \hat{\mathbb{E}}_t \left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)} \hat{A}_t \right] \\ \text{subject to} \quad & \hat{\mathbb{E}}_t[\text{KL}[\pi_{\theta_{\text{old}}}(\cdot \mid s_t), \pi_\theta(\cdot \mid s_t)]] \leq \delta. \end{aligned}$$

- 基本思想

PPO

- https://arxiv.org/pdf/1707.06347.pdf

Reward 设计

Reward 设计

- Reward 设计的重要性
- 一些基本准则
- Reward 的 scale 问题

Reward 设计的重要性

- 是 RL 算法设计的最大难点之一
- Reward 可能具有误导性
- Reward 经常面临过于稀疏的问题
- Reward 的信息量可能不够
- Reward 的 Scale 可能有问题

一些基本准则

- 信息量
- 噪声
- 避免鼓励 Agent 自杀的 Reward
- 观察可行性

Reward 的 scale 问题

- 一般建议是 Reward 应该进行 scale
- 不要采用 moving average 进行 scale！

极客时间

扫码试看/订阅

《NLP 实战高手课》视频课程