CODE A BETTER LIFE
一 行 代 码　　亿 万 生 活

深度强化学习技术在智能调度中的应用

王超

美团点评

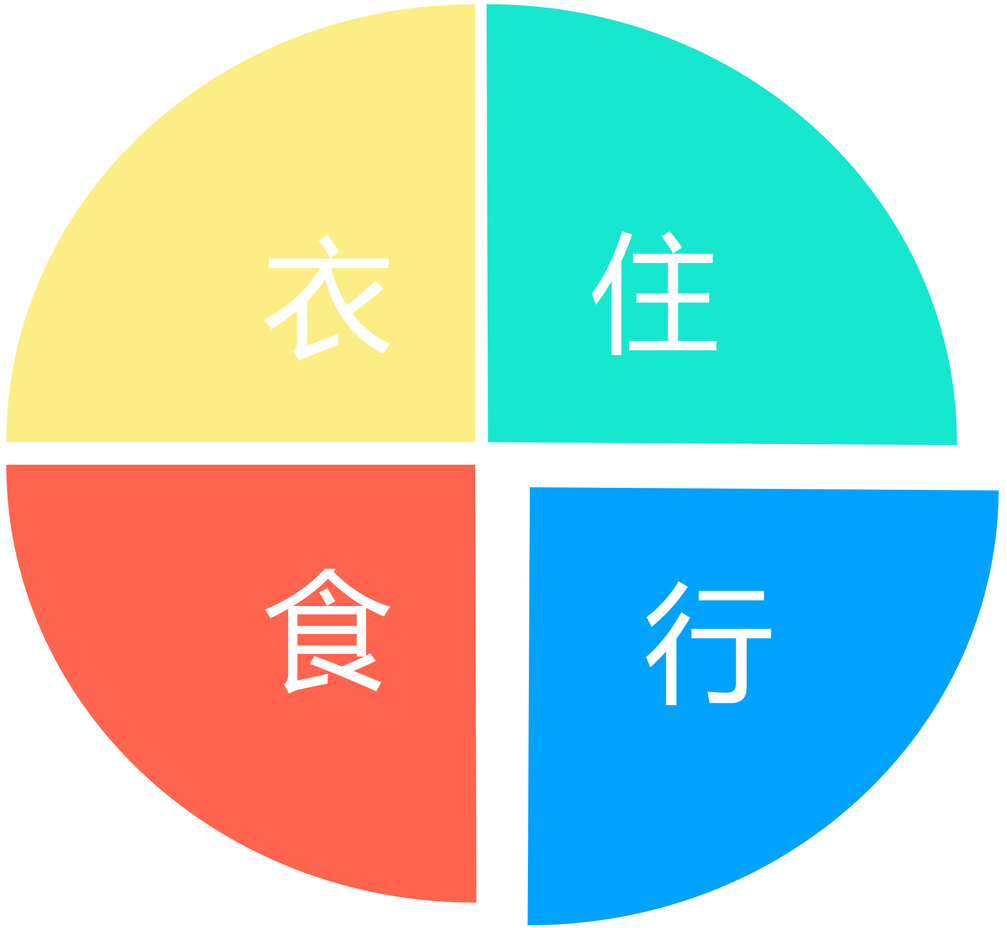
王超

美团打车 – 数据智能部，高级技术专家

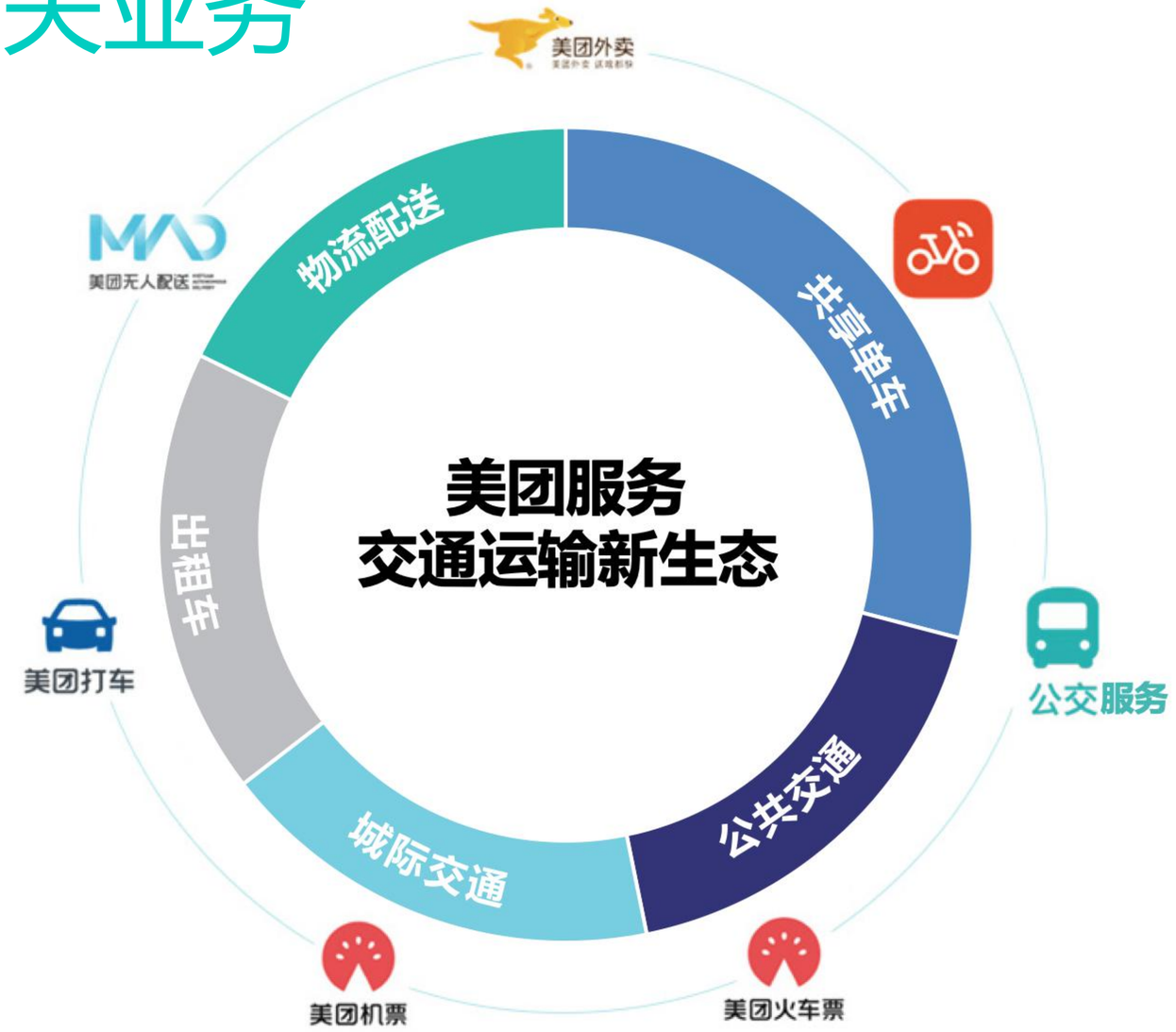
毕业后先后在雅虎、京东等公司从事个性化推荐、搜索等方向的研发工作，专注大规模机器学习系统、深度学习等方向的探索。

18年加入美团，目前负责智能策略相关的模型在美团打车中的应用。

EAT BETTER
LIVE BETTER



美团出行相关业务


美团外卖
物流配送
美团无人配送
共享单车
美团服务
交通运输新生态
公交服务
出租车
公共交通
美团打车
城际交通
美团机票
美团火车票

美团打车

2017

南京

开始在南京试点打车业务。

2018初

上海

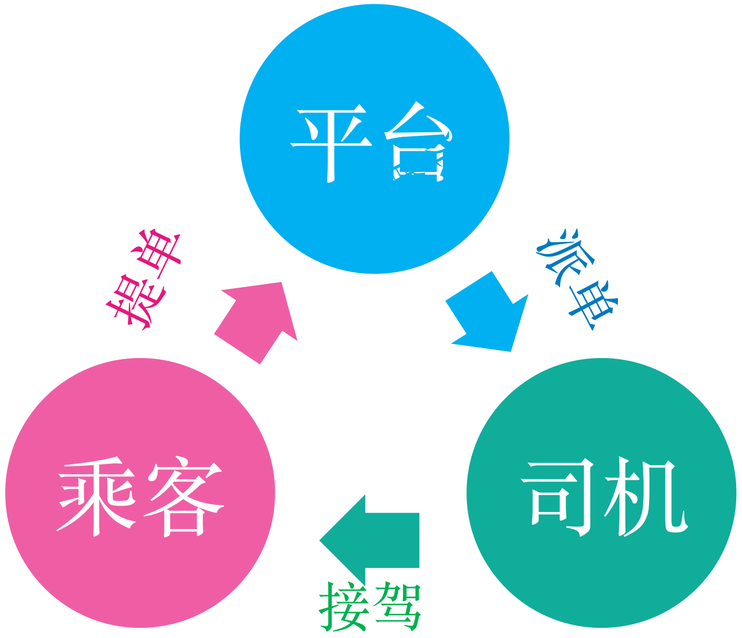
在上海上线打车业务，快速取得一定市场份额

2018底

出租车等业务

为满足不同的出行需求，发展出租车等多种业务

快车业务模式



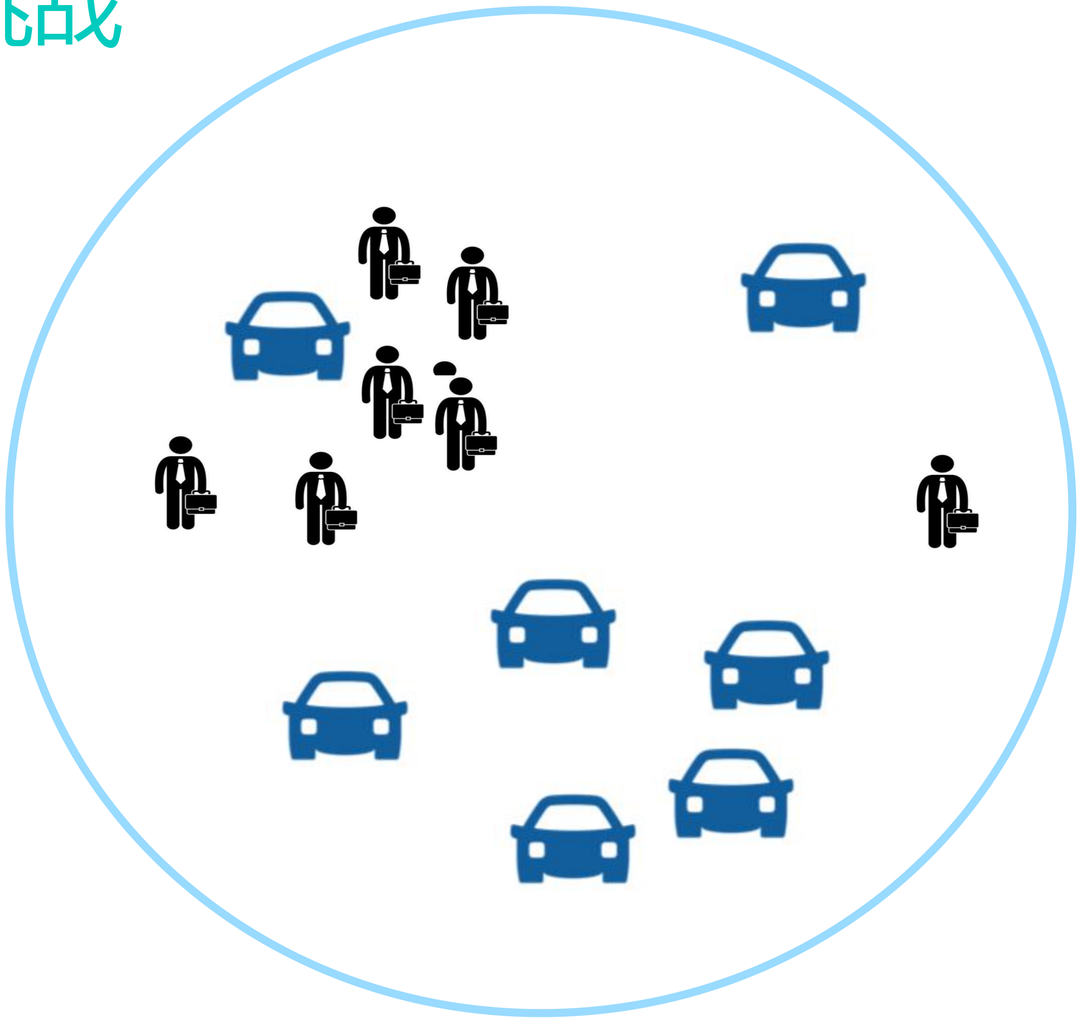
平台就是给乘客和司机建立关联

满足乘客出行需求

乘客

司机

挑战



乘客
高峰期打车难

司机
提高服务频次

平台
茫茫人（车）海中
牵线搭桥

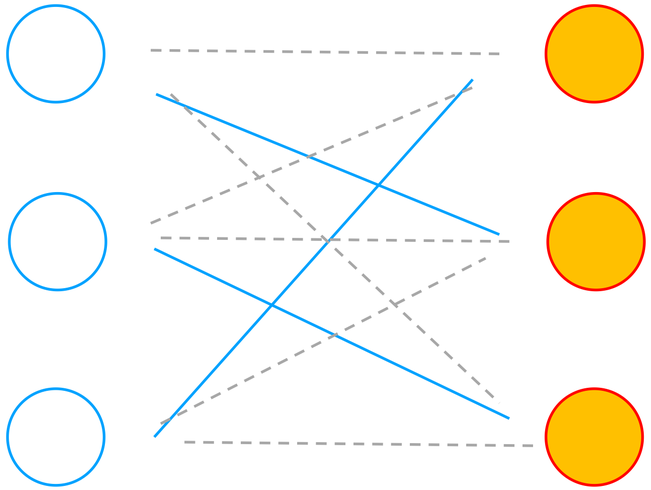
平衡供需

提高打车体验

提高乘客+司机体验

调度问题

乘客　　订单　　司机



- 当乘客面对多个司机时，分配给他哪个司机?
- 当司机面对多个乘客时，分配给他哪一单？还是都不给?

- 按照收益目标，找到最佳匹配!

问题定义

- 目标：
订单分配全局收益最大化

- 表示意义：
x_{ij}代表第i个乘客与第j个司机的匹配关系，0代表不匹配，1代表匹配

- 约束：
每个订单只匹配最多一个乘客
每个乘客只分配给最多一个司机

$$max \sum_{ij} w_{ij} x_{ij}$$

$$s.t. \sum_{j} x_{ij} \leq 1 \quad \forall i$$

$$\sum_{i} x_{ij} \leq 1 \quad \forall j$$

$$x_{ij} \in \{0,1\}, \qquad \forall i,j$$

如何定义收益？

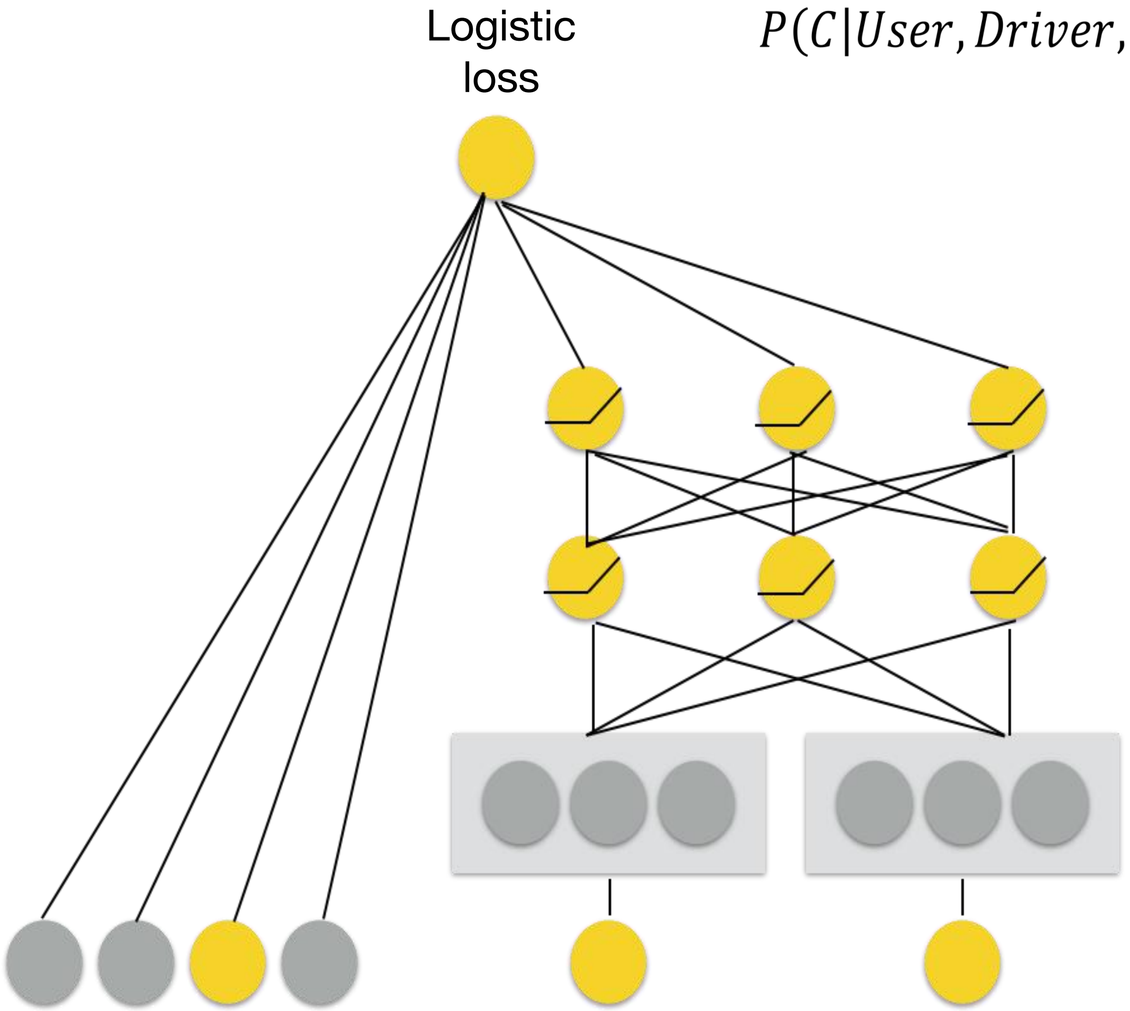
$$max \sum_{ij} w_{ij} x_{ij}$$

↓

$$max \sum_{ij} c_{ij} x_{ij}$$

短期收益
乘客体验：是否成单

成单率模型



Wide & Deep

模型：
- 泛化性
- 非线形网络结构

特征：
- LBS、司机、乘客、订单、供需、时间、天气…

如何定义司机+乘客收益？

$$max \sum_{ij} c_{ij} * d_{ij} * x_{ij}$$

↓

司机服务时长

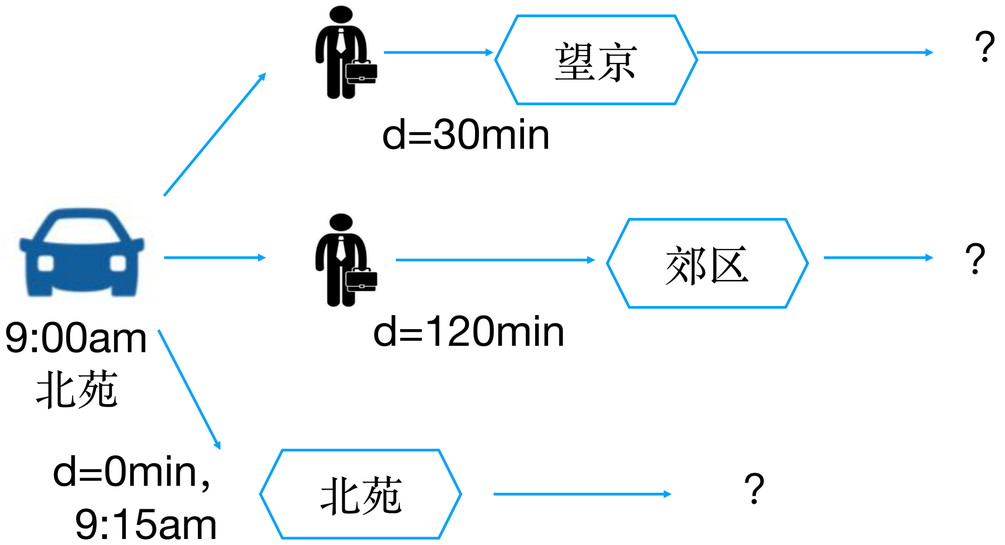
↓

乘客整体被服务时长提高

司机服务时长为目标

短期收益：每单服务时长

$$max \sum_{ij} c_{ij} * d_{ij} * x_{ij}$$



- 平台在早高峰给司机派个去市区的单还是郊区的?
- 司机要不要再等等，也许有个更好的单?

长期司机服务时长收益

$$max \sum_{ij} c_{ij} * a_{ij} * x_{ij}$$

$$a_{ij} = r_{ij} + V_{end,t'} - V_{start,t}$$

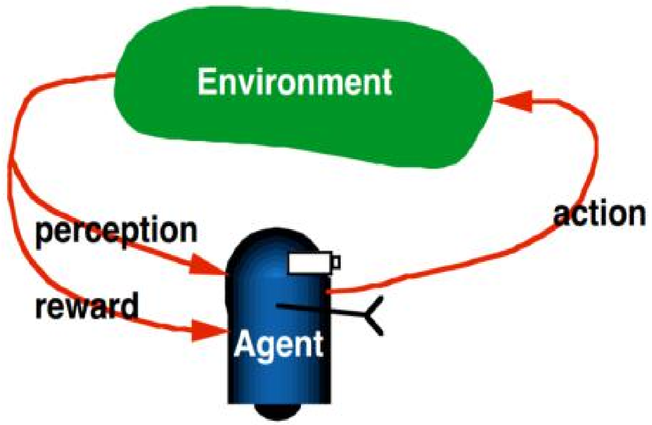
如果接单，去目的地的未来一天的收益

$V_{d,t}$ ：在时空（位置d，时间t）未来一天的收益
r_{ij} ：订单时长
a_{ij} ：未来收益的增量

优化全天司机服务时长，且是序列化问题，
尝试采用强化学习！

什么是强化学习？

Reinforcement learning (RL) is an area of machine learning concerned with how software agents ought to take actions in an environment so as to maximize some notion of cumulative reward.
-- Wikipedia



- Agent：作决策和动作的智能体
- Environment：交互环境，根据Agent动作产生反馈

- State：状态
- Action：动作
- Reward：环境根据动作产生的回报
- Policy (π)：在State下采取什么Action的策略
- P：状态转移的概率

强化学习目标：通过Agent与环境的多轮交互学习找到最优化Policy，使得长期Reward最大

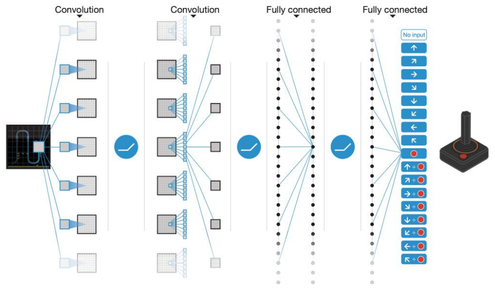
强化学习历史

1950s Bellman提出MDP建模，并提出动态规划的求解方式Bellman equation

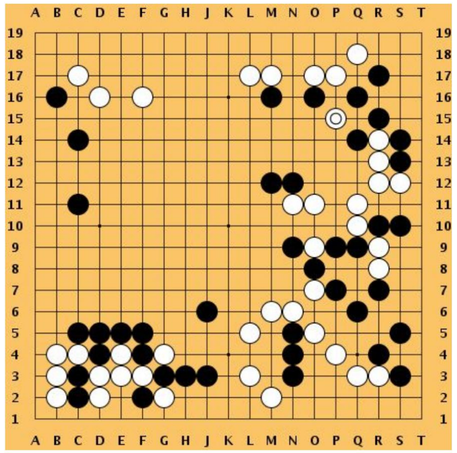
1950s-1990s 出现TD更新算法、Q-Learning、Sarsa等算法

2015 Deep-mind发表Human-level control through deep reinforcement learning，提出DQN

2015 Atari Games

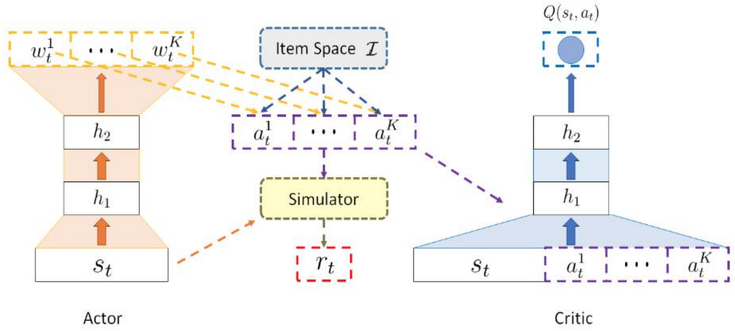


2016 AlphaGo



强化学习在互联网领域的应用

2016 淘宝双11在搜索中应用深度强化学习



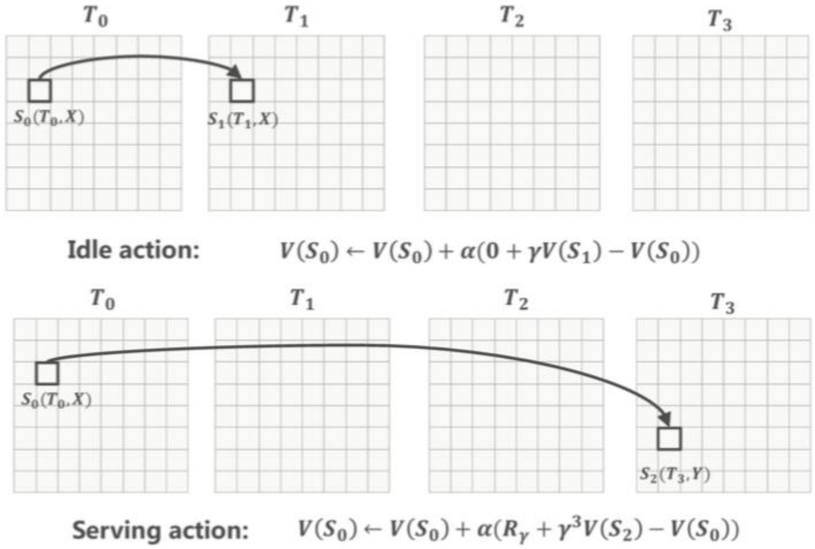
京东发表 Deep Reinforcement Learning for List-wise Recommendations

百度2018年一季度财报报道，百度凤巢应用强化学习大幅提高收入

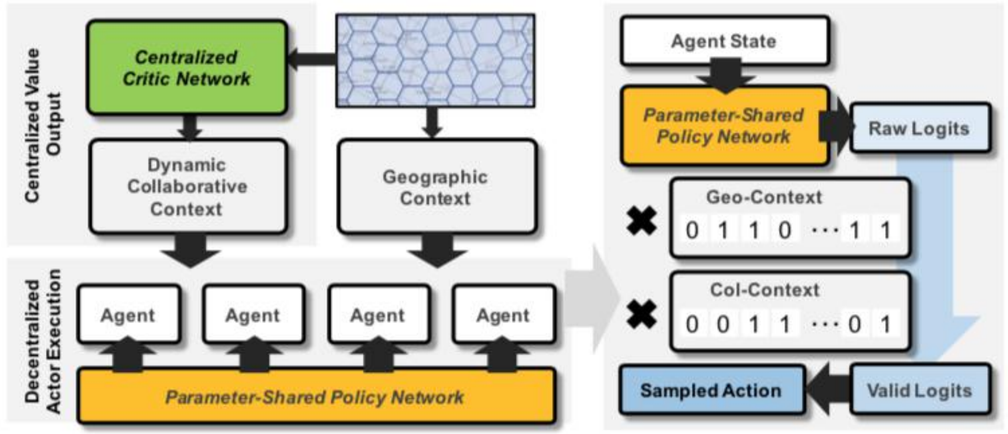
强化学习在出行领域的应用

滴滴出行 2018 KDD

Large-Scale Order Dispatch in On-Demand Ride-Hailing Platforms: A Learning and Planning Approach



Efficient Large-Scale Fleet Management via Multi-Agent Deep Reinforcement Learning



MDP建模

从最开始状态，通过一些列交互得到序列样本

↓

$(s_0, a_0, r_0, s_1, a_1, r_1, s_2, a_2, r_2, \ldots, s_t, a_t, r_t)$

↓

$$P(S_{t+1}, R_{t+1} | S_0, A_0, R_1, \ldots S_t, A_t) = P(S_{t+1}, R_{t+1} | S_t, A_t)$$

Markov Decision Process：假设下一个状态和回报，只和当前状态、动作有关

Policy

通过MDP对RL建模，就是学习出每个状态下的Decision（Policy），决定动作和下一个状态，使得长期回报最大化

Policy就是在State下采取什么Action的策略

- Deterministic Policy　　$a = \pi(s)$
- Stochastic Policy　　$p = \pi(a|s)$

价值函数

R_t只是t时刻的直接回报，但t时刻的未来长期收益如何表述？

$$G_t = \sum_{n=0}^{N} \gamma^n r_{t+n}$$

t时刻长期回报

$$V_\pi(s) = E_\pi[G_t | S_t = s]$$

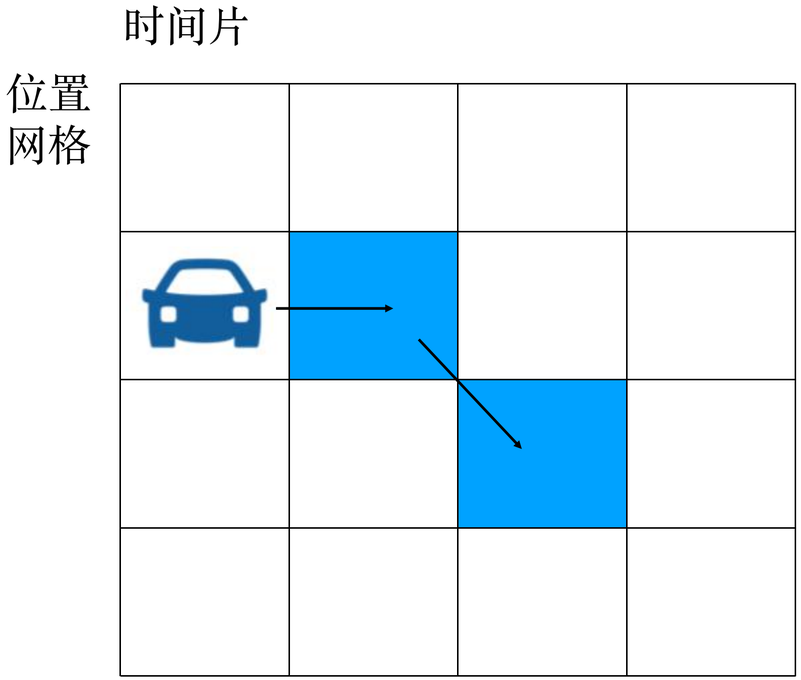
状态s的价值函数：在状态s 、采用Policy π的长期回报期望

通过引入价值函数，对于获取最优的Policy就有两种方法：

1、Policy iteration：直接优化策略$\pi(a|s)$或$a = \pi(s)$，每次迭代改进Policy
2、Value iteration：通过评估价值函数V，间接获得最优策略（通过知道每个状态的好坏，就知道如何选择了），每次迭代改进Value function，间接改进Policy

本次分享介绍的方案采用的是Value iteration方法

尝试方法：以单个司机视角对时空价值建模



Agent：平台的派单算法
Environment：调度环境，由供需、环境变量构成的系统

State：时间、空间
Action：接单去某个时空，或停留原地
Reward：接单则为订单时长、停留则为0

MDP建模：司机每个接单行为构成的序列

$$P(S_{t+1}, R_{t+1}|S_0, A_0, R_1, \ldots S_t, A_t) = P(S_{t+1}, R_{t+1}|S_t, A_t)$$

Policy定义

Value iteration：通过动作前后的价值变化，间接得到Policy：
Policy Evaluation

$$Advantage = r_{ij} + V(S_{t+1}) - V(S_t)$$
$$a = \pi(s) \Rightarrow argmax_a A(\mathrm{S}, \mathrm{a})$$

Greedy Policy!

如何训练得到V价值函数?

TD (Temporal Difference) Method

$$V_\pi(s) = E_\pi[G_t|S_t = s]$$

↓

Monte Carol update

$$V_\pi(S_t) \Leftarrow V_\pi(S_t) + \alpha(G_t - V_\pi(S_t))$$

α: $step\ size$，相当于learning rate

但Monte Carol法需要知道S到最终时刻时的未来价值，不适合在线场景

Bellman equation

$$V_\pi(s) = \sum \pi(a|s)E[R_{t+1} + \gamma V(s_{t+1})|S_t = s]$$

γ : future reward discount factor

↓

TD update

$$V_\pi(S_t) \Leftarrow V_\pi(S_t) + \alpha[\underbrace{R_t + \gamma V(S_{t+1}) - V_\pi(S_t)}_{\text{1-step TD error}}]$$

时空价值模型训练过程

参考：Large-Scale Order Dispatch in On-Demand Ride-Hailing Platforms:
A Learning and Planning Approach
From 滴滴出行

Algorithm 1 Policy evaluation (dynamic programming) for the local-view MDP

Input: Collect historical state transitions $D = \{(s_i, a_i, r_i, s_i')\}$; each state is composed of a time and space index: $s_i = (t_i, g_i)$.

Initialize $V(s)$, $N(s)$ as zeros for all possible states.
for $t = T - 1$ to 0 **do**
  Find a subset $D^{(t)}$ where $t_i = t$ in s_i.
  for each sample (s_i, a_i, r_i, s_i') in $D^{(t)}$ **do**
    $N(s_i) \leftarrow N(s_i) + 1$,
    $V(s_i) \leftarrow V(s_i) + \frac{1}{N(s_i)}[\gamma^{\Delta t(a_i)} V(s_i') + R_\gamma(a_i) - V(s_i)]$.
  end for
end for

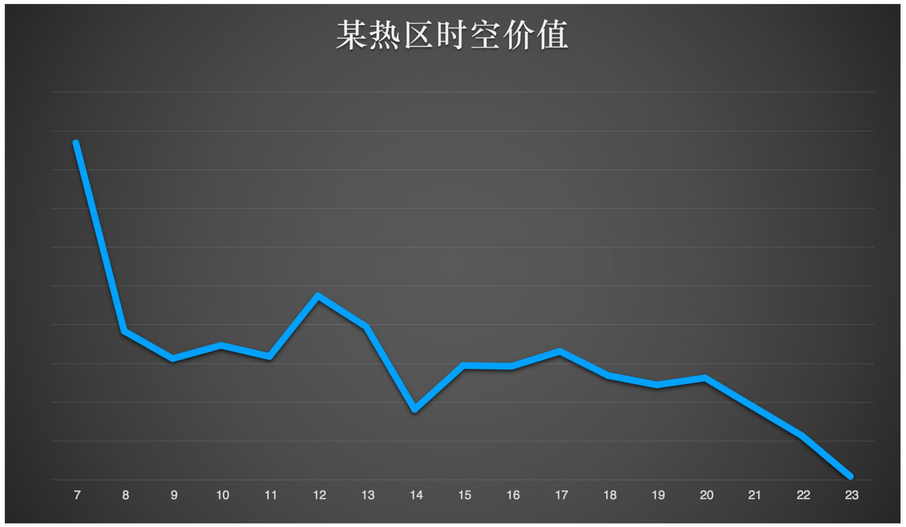
Return: Value function $V(s)$ for all states

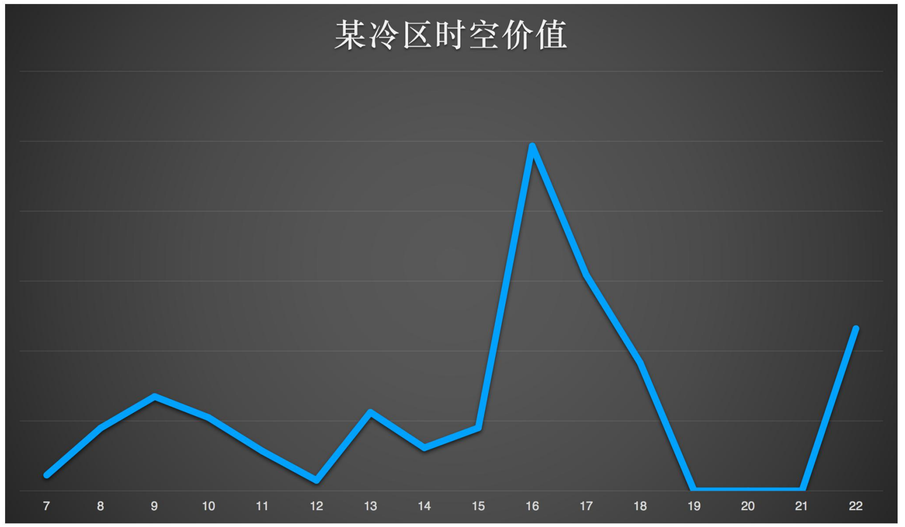
训练样本：根据订单和司机状态构造出 {<S, A, r, S'>}，每个序列为一天的长度
S：时间、空间
A：接单去某个时空，或停留原地
r：接单则为订单时长、停留则为0
S'：若接单，则为目的地时空；若停留，则为同一地点下一时间片的时空

初始V(s) table所有s的价值为0，按照TD update公式对每批样本进行更新

该方法问题之一：离散表示每个State泛化性较差

如果某时空上只出现很少几次司机，或根本没有出现过司机怎么办?

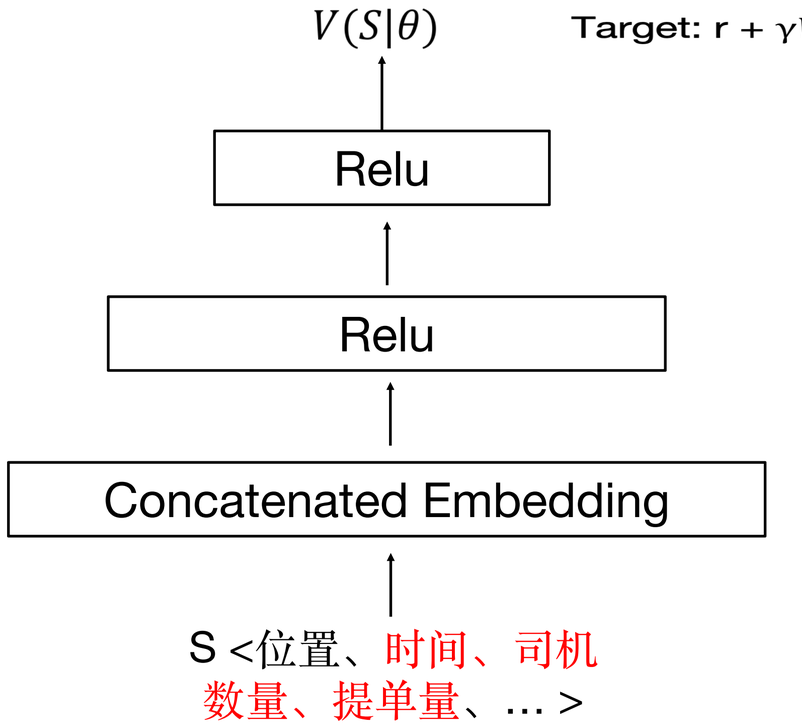




基于Tabular RL的方法泛化性较差，经常导致异常估值或估值为0的情况

改进1：使用深度神经网络参数化表示V(s)

$$V(S) \rightarrow V(S|\theta)$$



离散化并用embedding每个特征

改进2：使用Q-Learning直接评估动作的价值

我们希望直接评估时空+订单的价值，将订单、乘客等信息也能一起建模；

并直接评估动作的价值

新的建模方式：

- State：时空 + 候选匹配订单
- Action：{派单，不派单}
- Reward：{派单：订单时长，不派：0}

Q-Learning

Q函数：评估状态+动作的未来价值

$$Q_\pi(s,a) = E_\pi[G_t|S_t = s, A_t = a]$$

Q-Learning with TD update

$$Q_\pi(S_t,a_t) \Leftarrow Q_\pi(S_t,a_t) + \alpha[R_t + \gamma\, max_{a'}Q(S_{t+1},a') - Q_\pi(S_t,a_t)]$$

Off-policy: greedy choose max Q action, 实际中还可以采用$\varepsilon - greedy$进行动作探索

最终结合DNN与Q-Learning: Deep Q-Network 方案

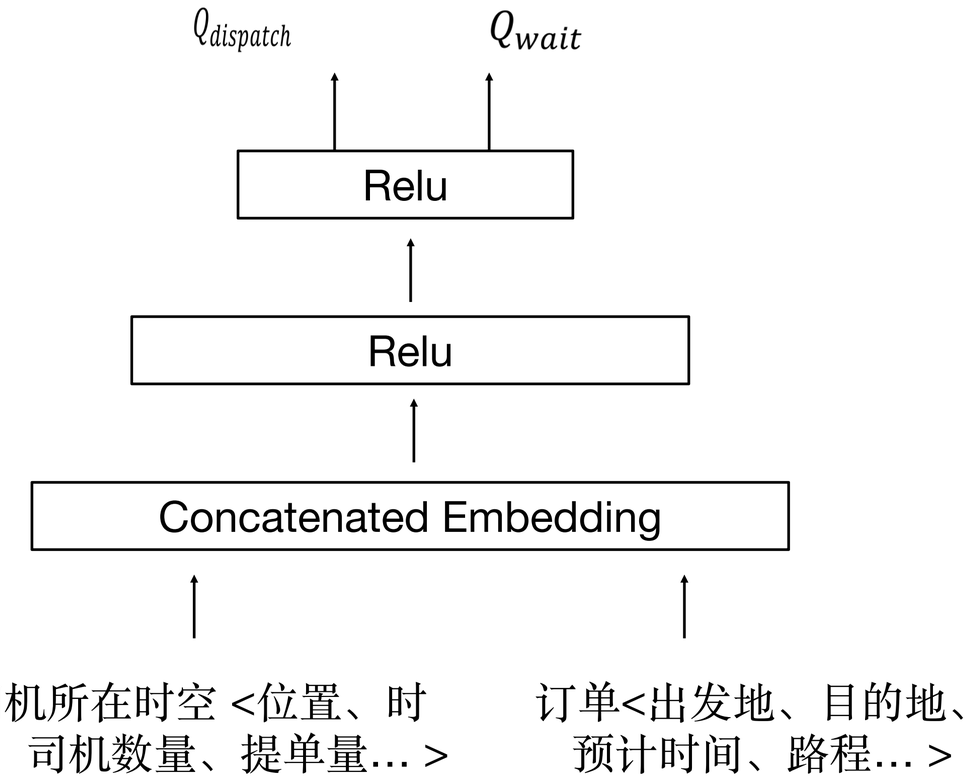
训练样本：观测线上每个state（订单）的action及reward，next-state，记录到日志：D={<S,A,r,S'>}

初始化模型参数θ

for episode = 1, M do
  sample mini-batch (s_i, a_i, r_i, s_{i+1}) from D

$$y_i = \begin{cases} r_i, & if\ episode\ terminate\ at\ s_{i+1} \\ r_i + \gamma\ max_{a_{i+1}} Q(s_{i+1}, a_{i+1}; \theta) & otherwise \end{cases}$$

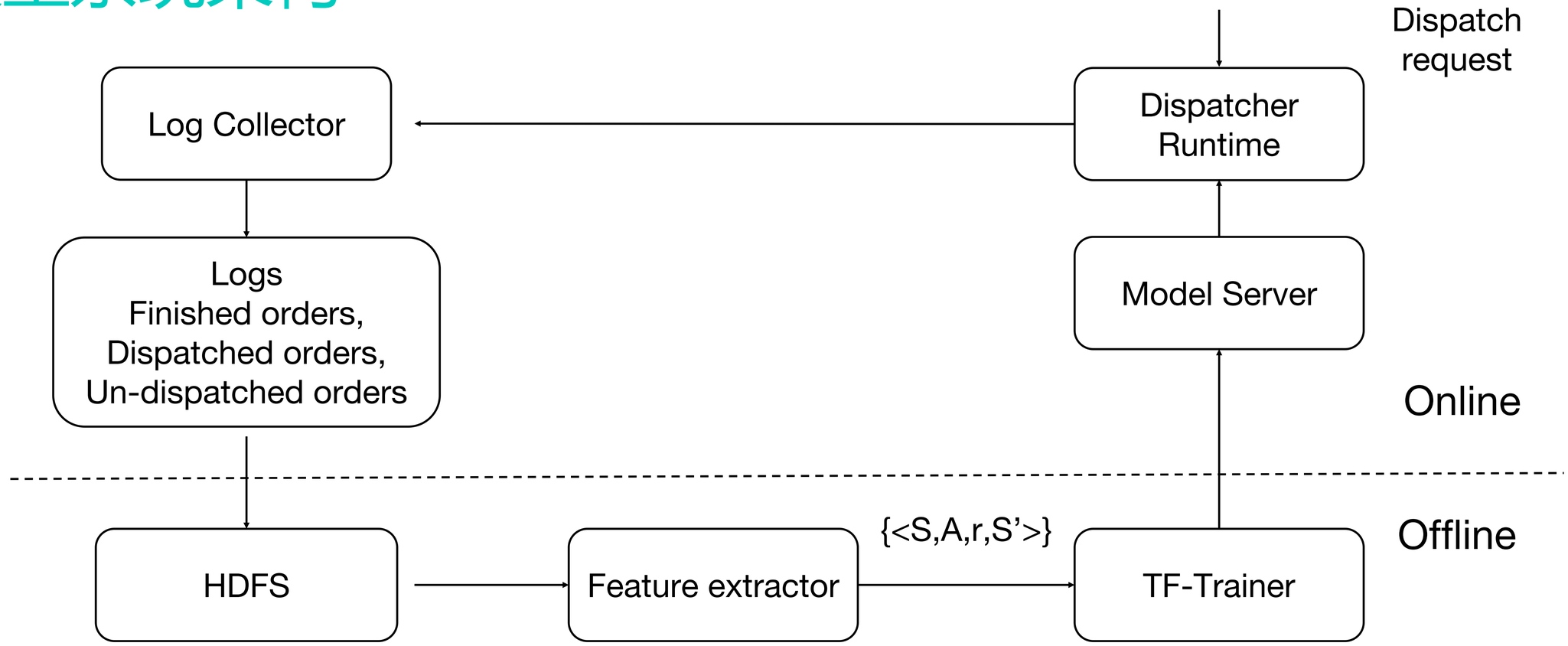
  perform SGD update on $(y_i - Q(s_i, a_i; \theta))^2$ with respect to θ



注：

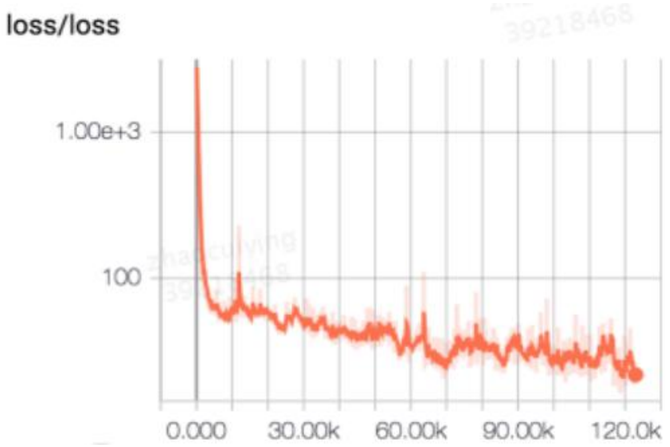
- D相当于experience-reply buffer
- Target network和当前Q-network使用的是相同参数，在现有工作中已经可以取得较好收敛

模型系统架构

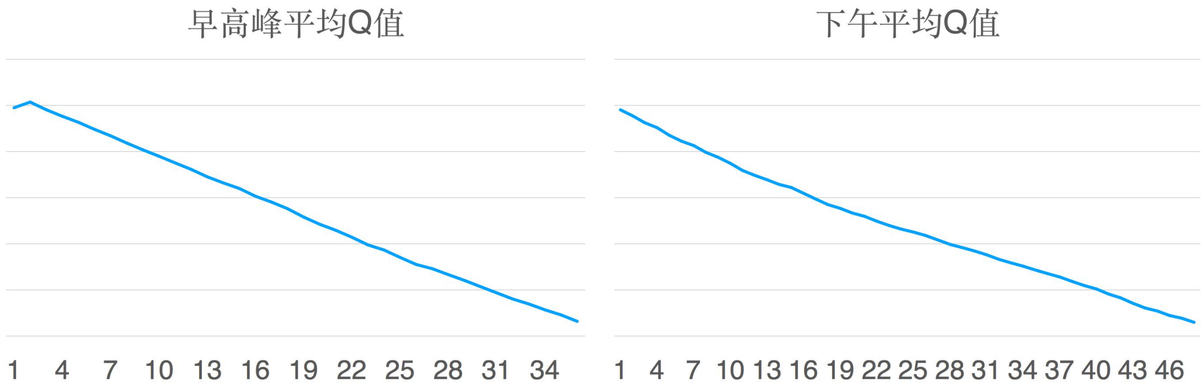

Dispatch request
Log Collector
Dispatcher Runtime
Logs
Finished orders,
Dispatched orders,
Un-dispatched orders
Model Server
Online
Offline
{<S,A,r,S'>}
HDFS
Feature extractor
TF-Trainer

训练及实验效果

Loss收敛



Q函数随时间下降



Tips:

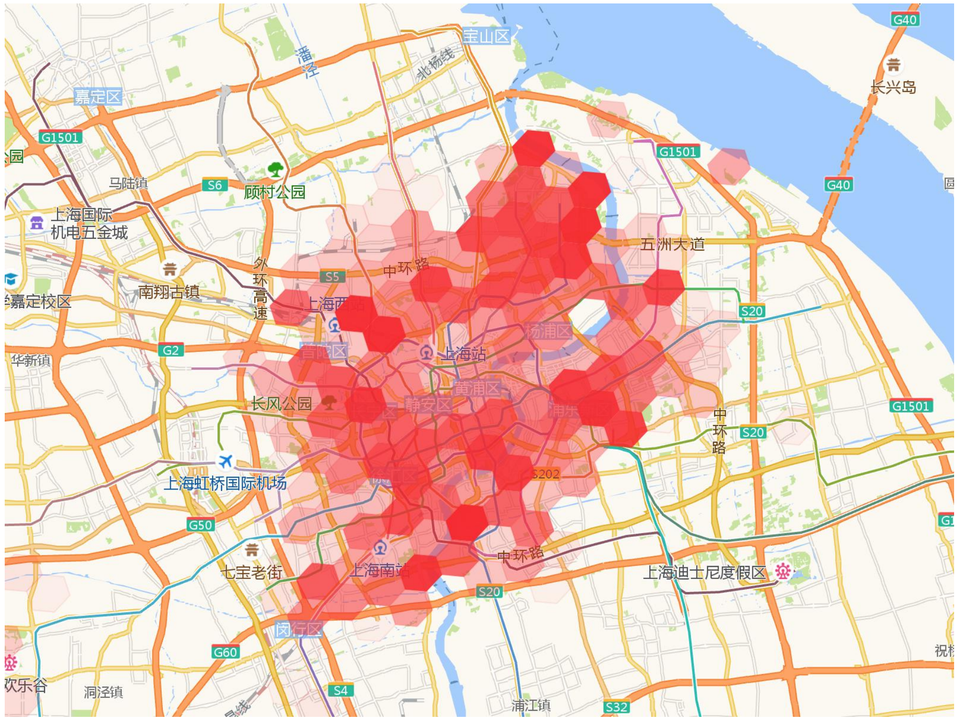
- 分时段作为一个session，而不是一整天，收敛更好
- 结合业务特点，对样本进行选择（sampling）
- discount factor对未来价值进行调节

实验效果

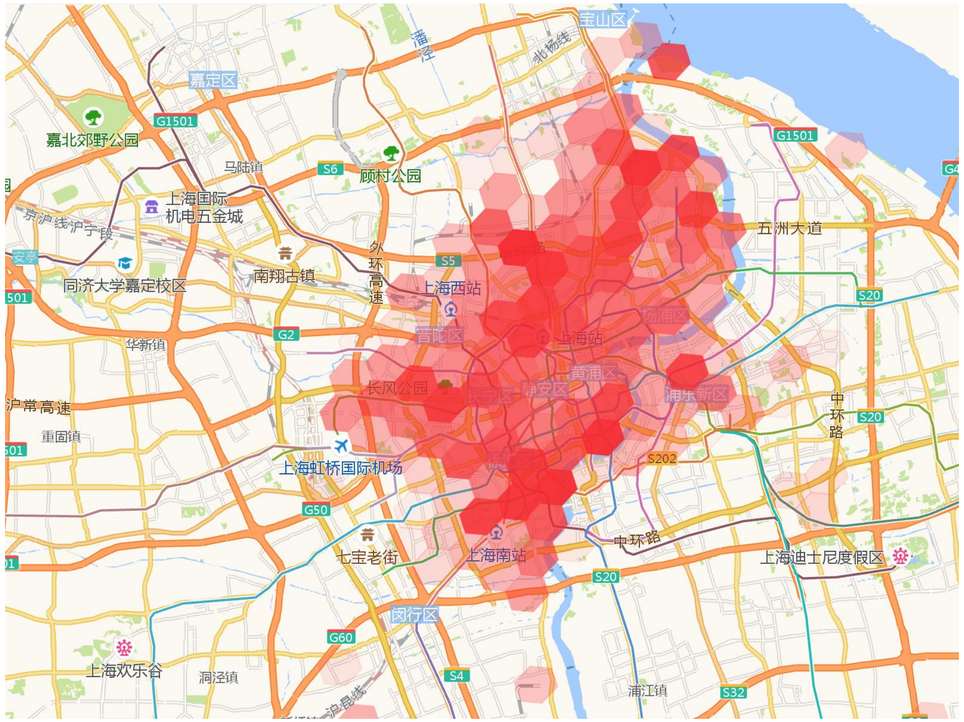
- 司机服务时长显著提高

应用场景 – 热点地区评估

上海：工作日早高峰

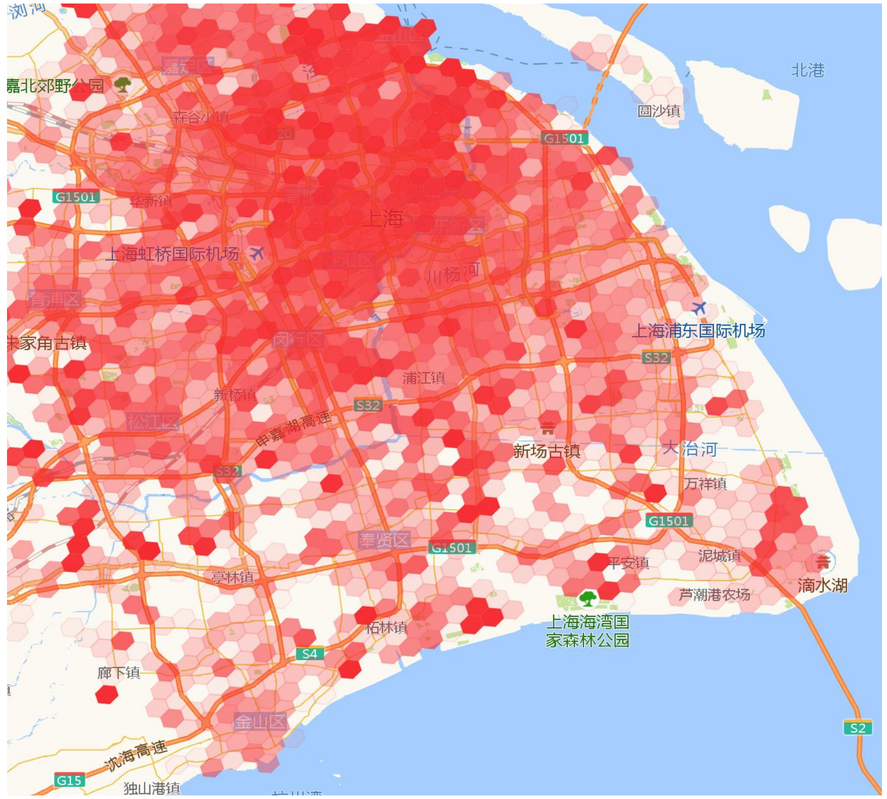


上海：工作日晚高峰

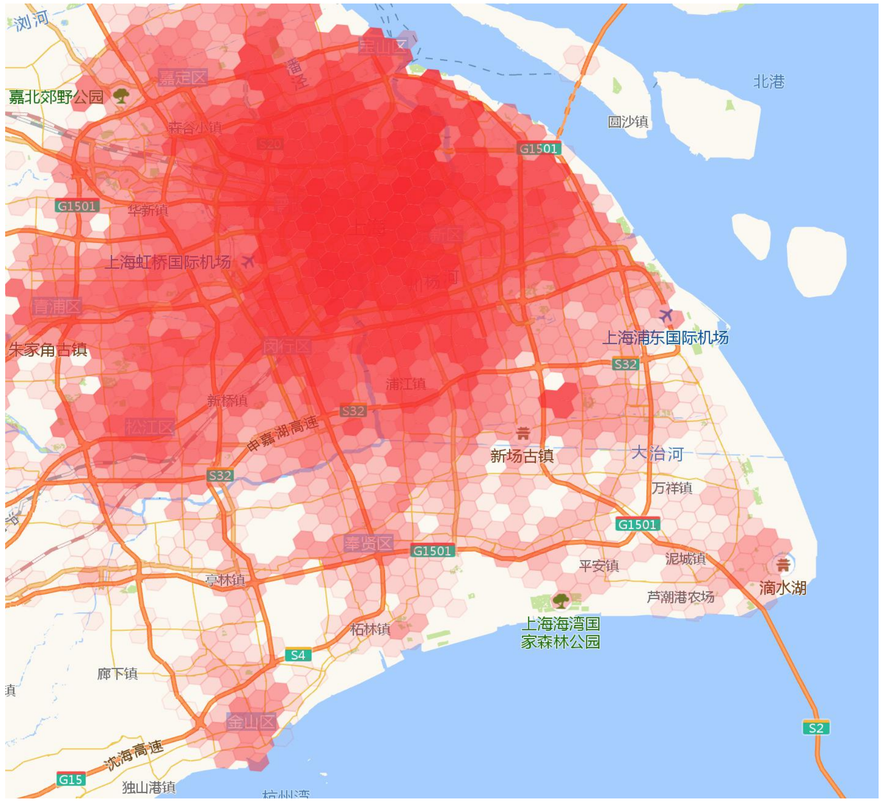


Tabular RL和DRL对比

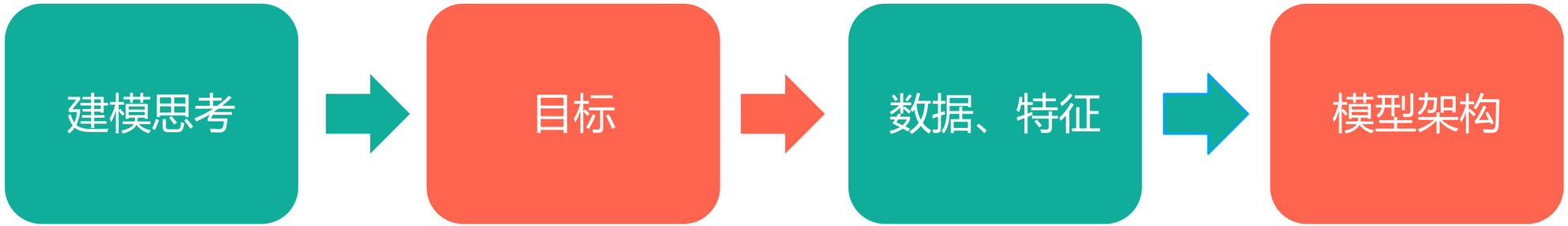
Tabular RL



Deep RL

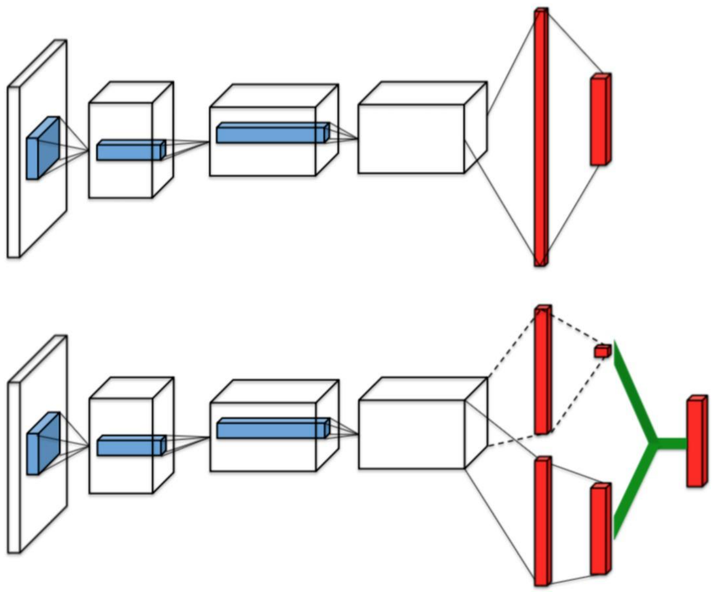


后续展望



后续展望

Dueling network

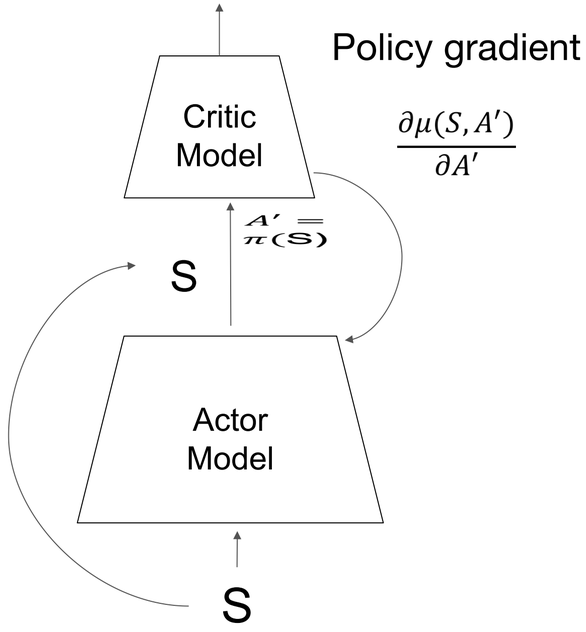


$$Q(s,a;\theta,\alpha,\beta) = V(s;\theta,\beta) + A(s,a;\theta,\alpha),$$

将DQN（上图）分拆成预估V(s)和A(s,a)的两个网络（下图），大量样本可以单独用于训练V(s)

DDPG

$$\frac{\partial L(w)}{\partial w} = r + \mu(S',A') - \mu(S,A)$$



通过DDPG，可尝试将RL用于连续值动作的场景

总结

1、介绍美团打车业务

2、调度环节要解决的问题

3、强化学习的初步实现

4、DQN在调度场景的实践

5、未来展望

Q&A

CODE A BETTER LIFE

一 行 代 码　亿 万 生 活

更多技术干货
欢迎关注“美团技术团队”

邮箱：
wangchao88@meituan.com

美团点评