Overview: Reinforcement Learning in Crowdsourcing and Crowdsensing

Oct. 28, 2020
1. Problem introduction
2. Recent work
3. Typical models
4. Personal research
5. Summary
Problem Introduction

- Motivation:
  - Proposal of novel reinforcement learning (RL) methods
  - Effective and efficient approximations are hard to design
  - Task assignment and worker arrangement can be modeled as MDP

- Difficulties:
  - Deep learning, especially Deep RL, has much higher computation costs than simple heuristic algorithms
  - It’s hard to achieve significant improvements on the state-of-the-art models using approximation algorithms
  - The design and training of networks can be really tricky
Recent Work

- **2020 ICDE:**
  - Curiosity-Driven Energy-Efficient Worker Scheduling in Vehicular Crowdsourcing: A Deep Reinforcement Learning Approach
  - An End-to-End Deep RL Framework for Task Arrangement in Crowdsourcing Platforms

- **2020 CIKM:**
  - Auxiliary-task Based Deep Reinforcement Learning for Participant Selection Problem in Mobile Crowdsensing

- **2020 PerCom:**
  - Participants Selection for From-Scratch Mobile Crowdsensing via Reinforcement Learning
Recent Work

- **2020 IEEE Internet Things of Journal:**
  - Energy-Efficient Mobile Crowdsensing by Unmanned Vehicles: A Sequential Deep Reinforcement Learning Approach

- **2019 DASFAA:**
  - Reinforced Reliable Worker Selection for Spatial Crowdsensing Network

- **2019 Computer Networks:**
  - Reinforcement Learning-Based Cell Selection in Sparse Mobile Crowdsensing (From ICDCS 2018)
Recent Work

- Common RL methods:
  - Multi-Arm Bandit (MAB)
  - Deep Q Network (DQN)
  - Proximal Policy Optimization (PPO)

- Deep RL is utilized in almost every work

- Most works use RNN and attention transformer

- Sometimes the real world is formulated as rectangular grids and CNN is used to extract spatial patterns
Typical Model I

- **Title:** An End-to-End Deep RL Framework for Task Arrangement in Crowdsourcing Platforms
- **Conference:** ICDE, 2020
- **RL Algorithm:** Double Deep Q Network (Double DQN)
- **Network Kernel:** Attention transformer
At timestamp $i$, a worker $w_i$ comes and there is an available task set $\{T_i\}$. (So a timestamp is triggered by a coming worker)

- $w_i$ has a feature $f_{w_i}$ (completion history) and a quality $q_{w_i}$

- The model contains two different Q networks to optimize MDP (w) and MDP (r)
MDP for Workers

- MDP (w):
  - State $s_i = f_{w_i}$
  - Action $a_i = t_{ij}$ or $a_i = \sigma(T_i) = \{t_{i1}, t_{i2}, \ldots\}$
  - Reward $r_i = \begin{cases} 1, & \text{if } w_i \text{ completes } t_{ij} \\ 0, & \text{otherwise} \end{cases}$
  - Future state $s_{i+1}$ happens when $w_i$ comes again

- From the definition of MDP (w), the objective is to optimize the cumulative completion rate of workers in the long run.
Problem in MDP (w)

- Parameters in Q-network(w) are shared by all workers
- We need to predict $s_{i+1}$ so that we can update immediately

- Learn the distribution $\phi(g)$ for $\text{Time}_{i+1}$
- Check for expired tasks and compute $r_{i+1}$
- Use expectation to update the parameters

Distribution $\phi(g)$ in History
MDP for Requesters

- MDP \((r)\):
  - State \(s_i = [f_{wi}, f_{Ti}, q_{wi}, q_{Ti}]\)
  - Action \(a_i = t_{ij}\) or \(a_i = \sigma(T_i) = \{t_{i1}, t_{i2}, \ldots\}\)
  - Reward \(r_i = \Delta q_i\) (quality gain)
  - Future state \(s_{i+1}\) happens when the next worker \(w_{i+1}\) comes

- From the definition of MDP \((r)\), the objective is to optimize the cumulative quality gain of tasks in the long run.
There are too many possibilities for \((w_i, s_i, w_{i+1}, s_{i+1})\)

Similar to MDP \((w)\), learn \(\phi(g)\) from history

Learn the rate of new workers \(p_{new}\)

Obtain the probability for a coming worker \(w\):

\[
\Pr(w_{i+1} = w) = \begin{cases} 
(1 - p_{new}) \frac{\phi(g_w)}{\sum_{w' \in W_{old}} \phi(g_{w'})}, & w \in W_{old} \\
p_{new}, & w \text{ is new}
\end{cases}
\]

Use summation over \(g\) and \(w\) to update the parameters
Q Networks

- Network structure:

- Challenges:
  - The number of tasks (input size) are not fixed
  - The order of tasks should not affect the output
  - The value of a task is influenced by others
Q Networks

- Network structure:

- Solutions:
  - Set maximum number of available tasks & Use zero padding
  - Use multi-head self-attention layer
  - Structures to enhance the stability and learning ability
Typical Models II

- **Title:** Energy-Efficient Mobile Crowdsensing by Unmanned Vehicles: A Sequential Deep Reinforcement Learning Approach

- **Journal:** IEEE Int. Things of Journal, 2020

- **RL Algorithm:** PPO + Actor-Critic

- **Network Kernel:** CNN + LSTM
Optimization Object

- Energy efficiency of the entire system at timestamp $t$ as

$$\alpha_t = \frac{\beta_t f_t}{e_t}$$

- $\beta_t$: Data collection ratio
- $f_t$: Jane’s fairness index
- $e_t$: Energy consumption ratio
POMDP Formulation

- Observation: $o_t$ is a 3-D vector in size $m \times n \times 3$
  - The real-world 2-D positions are mapped to $m \times n$ grids
  - Channel 1 contains obstacles (-1) and sensor nodes (data)
  - Channel 2 contains UVs (energy) and charging stations (-1)
  - Channel 3 contains sensor nodes (visit times till timeslot $t$)
  - Other positions are filled with 0
POMDP Formulation

- **Action:** \( a_t = [a_t^1, a_t^2, \cdots, a_t^V] \), where \( a_t^v \in \mathbb{R}^3 \)
  - \( a_t^v(0) \) and \( a_t^v(1) \) denotes moving distance and direction of \( v \)
  - \( a_t^v(2) \) denotes charging (> 0) or collecting data (< 0)

- **Reward:** \( r_t = \frac{1}{V} \sum_v (r_t^v + p_t^v) \)
  - \( v \) gets reward \( r_t^v = f_t \frac{\Delta d_t^v}{\eta_t \Delta l_t^v + \eta_d \Delta l_t^v} \)
  - \( v \) gets penalty \( p_t^v \) when it hits obstacles or runs out of energy
Policy Network

- Use CNN structure to extract spatial patterns
- Flatten and linearly map 3-D features to 1-D features
- LSTM generates $h_t$ and $c_t$ with historical $h_{t-1}$ and $c_{t-1}$
- Obtain $\pi_\theta(a_t|h_t)$ and $V_\theta(h_t)$

Loss function: $L_t(\theta) = E_t[L_t^{CLIP}(\theta) - c_1L_t^{VF}(\theta) + c_2S[\pi_\theta](h_t)]$
In each iteration (episode), we collect $N$ pieces of transition which have a length $T$ and split them into pieces of length $k$.

With smaller pieces, we optimize the loss function in minibatch manners and update parameters of the network.

**PPO Network Optimization**
Personal Research

- Some previous work have paid attention to the fairness of data collection at all PoIs in crowdsensing space.

- My recent focus is to build an RL framework, which can effectively optimize an objective in crowdsensing system while keeping the load of all sensing devices balanced.

- At present I am to formulate a valuable and available problem.

- Co-workers are very welcome !!!
Another Direction

- Up till now, little focus on Multi-Agent RL approaches
- On behalf of the platform, MARL looks suitable
- Local observation indicates better privacy protection?
- “Cooperation” between workers leads to better solution?
- I might look into this type of model in a short time.
Summary

- The introduction of applying (Deep) RL in scheduling problem of crowdsourcing and crowdsensing
- Recent work and approaches on this topic
- Two representative models proposed this year
- Progress of my own research
Thanks!