Learning Note Aug 2020

Contextual Reinforcement Learning

Baizhi (Daniel) Song

- Theoretical Note for CMDP's formation
- Proposed algorithm for Contextual RL

1 Contextual Setting and Function Approximation

Here we summarize and discuss several contextual settings in different problems. To recapitulate, the form of context and reduction of parameters is related to whether the context information is independent with state space $\mathcal{S}$ and action space $\mathcal{A}$.

1.1 Example for contextual setting

1.1.1 Contextual MNL

Multinomial logit (MNL) model offer a sequence of assortments of at most K items from a set of N possible items that minimize regret. The utility or the degree of attractive for a specific item may depend on both user and item's features (context $X(i) \subseteq \mathbb{R}^d$) and the probability of being chose is dynamically depends on that context.

- In MNL's setting (Agrawal et al., 2017), the probability of an item being chose is depend on all items in the selected subset, we intuitive use "utility" v_i for item i in subset $\mathcal{S}$ as parameters of its probability. The number of parameter to sample is S.

$$P_i(S) = \frac{v_i}{\sum_{j \in \mathcal{S}} v_j + 1} \qquad i \in \mathcal{S}$$

- By introducing contextual setting (Min-hwan et al., 2019), we represent the utility of each item as a function of feature x(i): $v_i = exp(\theta x(i))$. The number of parameter to sample is d.

$$P_i^x(S) = \frac{exp(\theta^{(i)} x(i))}{\sum_{j \in \mathcal{S}} exp(\theta^{(j)} x(j)) + 1} \qquad \theta \subseteq \mathbb{R}^d$$

1.1.2 Clinical trial CMDP

Patients with different health condition (context $X \subseteq \mathbb{R}^d$) may have different clinical settings (MDP with different $P(s'|s,a)$ and $r(s,a)$)

- Given n patients, we have n different $M_i(A, S, P_i(s'|s,a), R_i(s,a))$ with $n \times S \times S \times A$ parameters for transition matrix. If we consider the MDP with all different contexts, the state and parameter become infinite.
- By introducing contextual setting (Modi et al., 2020), we reduce the number of parameters from $n \times S \times S \times A$ to $d \times S \times S \times A$:

$$P_x(s_i|s,a) = \frac{exp(\theta_{sa}^{(i)} x)}{\sum_{j=1}^{S} exp(\theta_{sa}^{(j)} x)} \qquad \theta_{sa} \subseteq \mathbb{R}^{S \times d}$$

1.2 Function approximation

Use selected feature to approximate large scale MDP, the approximate function can be linear or more complex version (deep neural network)

1.2.1 Q-learning with linear approximation

For a large state set S, we approximate the $Q(s,a)$ by a linear combination of feature set $\overrightarrow{x}(s,a)$:

$$Q(s,a) \approx Q_\theta(s,a) = \theta_0 + \theta_1 x_1(s,a) + ... + \theta_d x_d(s,a) = \theta \overrightarrow{x}(s,a)$$

In this case we reduce the parameters to estimate from $S \times A$ to d.

1.2.2 Transit matrix with function approximation(proposed)

Here we propose four forms of approximation:

- Case-1; feature set $x(s,a,s')$

$$P(s'|s,a) = \frac{exp(\theta x(s,a))}{\sum_{j=1}^{S} exp(\theta x(s,a))} \qquad \theta \subseteq \mathbb{R}^d$$

  $\Rightarrow d$ parameters

- Case-2: feature set $x(s,a)$

$$P(s'|s,a) = \frac{exp(\theta^{(i)} x(s,a))}{\sum_{j=1}^{S} exp(\theta^{(j)} x(s,a))} \qquad \theta \subseteq \mathbb{R}^{S \times d}$$

  $\Rightarrow d \times S$ parameters

- Case-3: feature set $x(s)$

$$P(s'|s,a) = \frac{exp(\theta_a^{(i)} x(s))}{\sum_{j=1}^{S} exp(\theta_a^{(j)} x(s))} \qquad for\ a \in \mathcal{A},\ \theta_a \subseteq \mathbb{R}^{S \times d}$$

  $\Rightarrow d \times S \times A$ parameters

- Case-4: feature set x, similar form to case 2 and 3, and this is the case in the common CMDP setting (clinical trail)

$$P(s'|s,a) = \frac{exp(\theta_{sa}^{(i)} x)}{\sum_{j=1}^{S} exp(\theta_{sa}^{(j)} x)} \qquad for\ s,a \in \mathcal{S} \times \mathcal{A},\ \theta_{sa} \subseteq \mathbb{R}^{S \times d}$$

  $\Rightarrow d \times S \times S \times A$ parameters

In the case of recommendation problem, if the state is a list of user's previous purchase like (x_1, x_3, x_4) then the state space it self could be infinite, in this case we could not take case 2,3 and 4's setting due to the infinite S. Case 1's setting is not covered a lot in the current CMDP's research.

1.2.3 Normalization and sparse structure

In previous part we discussed four different cases of the contextual approximation of transition matrix in MDP. It can be seen that the main difference between different cases is the dependency of contextual feature x with state space $\mathcal{S}$ and action space $\mathcal{A}$. Apart from the feature x, the rest of structure is quite similar: exponential like feature aggregation and normalization. Here we discuss more about the idea of normalization and potential way to solve the infinite state space problem for the case 2 to 4.

$$u(s'|s,a) = exp(\theta_{sa} x(s,a))$$

$$P(s'|s,a) = \frac{u(s'|s,a)}{\sum_{s_i \in S} u(s_i|s,a)}$$

The normalization method used in above section is based on the fact the the sum of the probability within all events is 1 ie. $\sum_{s' \in \mathcal{S}} P(s'|s,a) = 1$.Currently the problem in case 2 to 4 is that the state space $\mathcal{S}$ is extremely large, which make it highly costly to normalize along all states. However, under certain situations (many open AI games where agent moves step by step), it is reasonable to make such an assumption that the next state could only be in a certain 'near subset' ie. $P(s'|s,a) = 0$ for $s' \notin K_{|s,a}$, which also means that we assume the transition matrix is sparse. In this case the normalized probability can be represented as:

$$P(s'|s,a) = \frac{u(s'|s,a)}{\sum_{s_i \in S} u(s_i|s,a)} = \frac{u(s'|s,a)}{\sum_{s_i \in K_{|s,a}} u(s_i|s,a)}$$

So far it is not difficult to realize the above sparse form of transition matrix is similar to the problem setting in contextual MNL. In contextual MNL problem, the probability of an item being chose is normalized along the assortment subset S_t with a finite and tractable cardinality. Here in CMDP we achieve the same from by adding the sparse assumption.

2 Posterior Sampling for CMDP

To sample the parameter in contextual setting, we may not get a conjugate prior so we will do it in by the way of MCMC or variational inference.

3 Contextual PSRL Algorithm

Here we proposed the algorithm of PSRL in Contextual setting.

Initialization ;
for *epoch k = 1, 2, ...* **do**
  Sample contextual parameter θ via MCMC or variational inference;
  while *True* **do**
    Observe contextual vector x;
    Calculate transition probability and reward;
    Compute optimal policy π^t through extended value iteration;
    Execute π^t and collect data;
    $a_t = \pi^t(s_t)$;
    Observe next state $s_{t+1} = env(a_t)$;
    $t = t + 1$;
  end
end

Algorithm 1: Contextual PSRL

From above it can be seen that main difference between the original version and the contextual version of the PSRL algorithm is the number of value iteration, since previously we can calculate the optimal policy

only once at every beginning of each epochs and then repeat that policy through out the time. Now in contextual setting, as the transition matrix varies at each time steps, the optimal policy has to be calculated at every time step t.

Considering that the posterior sampling part would also be more time consuming due to the MCMC method, the total time of the Contextual PSRL should be slower if the state space is only middle size.

4 Contextual RL Example

Here we discussed several potential feasible examples for the applications of contextual RL. The focus is on the formation of state space S and contextual feature space X.

4.1 River Swim MDP

We first consider the simplest example that we tested in MDP's setting. In River Swim MDP, both the state and the action space are one dimension discrete space:

- **State**: $s \in \{0, 1, 2, ..., S\}$ as the s^{th} states in the river.
- **Action**: $a \in \{0, 1\}$ as swim left (0) or right (1).

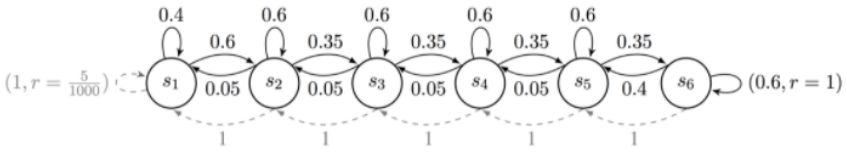

Figure 1: River Swim MDP

From the Fig. 1 it can be seen that the transition probability is quite simple and can be captured by following contextual feature:

- **Contextual feature**: $x(s, a) = (s, a, e(s))$ where $e : \{0, 1, 2, ..., S\} \mapsto \{0, 1, 2\}$ indicates if the state s is at the middle or the end of the river chain.

$$e(s) = \begin{cases} 0 & s = 0 \\ 1 & 0 < s < S \\ 2 & s = S \end{cases}$$

4.2 OpenAI Gym - Cart Pole

Now we makes the situation more complex, expanding to a larger, continuous state space. In cart pole game, agent gets rewards as long as the pole stays upright. Therefore, we need to choose optimal policy to keep the pole always stand upright.

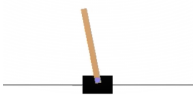

Figure 2: Atari Game: Cart pole

- **Action**: $a \in \{0, 1\}$ as push the cart to left (0) or right (1).
- **Observation feature**: $b = (x, x', \theta, \theta')$ as the features observed from the environment.
  - Cart position $x \in [-2.4, 2.4]$ is the position of the cart.
  - Cart velocity $x' \in \mathbb{R}$ describes the velocity of the cart.
  - Angle $\theta \in [-41.8°, 41.8°]$ is the angle of the pole.
  - Angle velocity $\theta' \in \mathbb{R}$ describe the speed of pole's angle.

It can be seen that in the this example, there is no obvious "discrete state space" exists but a four dimensional continuous observable feature space. There might be more works to be done to decide a finer state space. Here we proposed a naive form of state space:

- **State**: $s \in \{0, 1, 2\} \times \{0, 1, 2, 3\} \times \{0, 1, 2, 3\} \times \{0, 1, 2, 3\}$ as the discretized observation feature space. To be more specified, it can be illustrated as{left-hand side, middle, right-hand side } × { car fast to left, car slow to left, car slow to right, car fast to right } × { large angle to left, small angle to left, small angle to right, large angle to right } × { angle fast to left, angle slow to left, angle slow to right, angle fast to right}.
- **Contextual feature**: $x(s, a)$ could have two forms: $x(s, a) = (b, a)$ where the feature b is the observed continuous feature. The other form is $x(s, a) = (s, a)$ where the s is the discrete value as in state space.

4.3 OpenAI Gym - Breakout

Here we discuss another Atari game in OpenAI. The Breakout game is a video game where player hit the bricks for score by bouncing off the ball via a horizontal paddle. The player can observe the whole screen includes the position of the ball, the structure of the bricks and the position of the paddle.

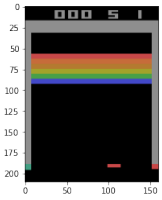

Figure 3: Atari Game: Breakout

- **Action**: $a \in \{0, 1, 2, 3\}$ as move the paddle to left(0), right(1), no move(2), launch the ball(3).
- **Observation feature**: $x \subseteq [0, 255]^{210 \times 160 \times 3}$ is the RGB value of the whole game screenshot as shown in Fig 3. This is an extremely large space to use directly.

In this example we have neither an obvious state space nor a direct contextual feature space. One of the solution is to use Deep Q-Network (DQN) and handle the input image through convolution layer. Apart from this approach, a proposed model-based approach is to first conduct feature engineering on the image to extract useful features such as ball position, bricks position, ball trajectory etc.. Given that low dimensional feature set, we can apply similar discretization method in previous cart pole example to get a relatively tractable state space.