CPS 제어 및 분석 알고리즘 특론

- Reinforcement Learning #8 Actor-Critic Methods -

부산대학교

김호원

부산대
정보보호 및 IoT 연구실,
사물인터넷연구센터

2018.12

I. Monte Carlo Method

1. Beyond Dynamic Programming
2. The Monte Carlo method

https://mpatacchiola.github.io/blog/2017/01/15/dissecting-reinforcement-learning-2.html

Beyond Dynamic Programming

- **In previous slide, two main algorithms(namely value iteration and policy iteration) are presented based on Bellman algorithm**

We modelled the environment as a Markov decision process (MDP), and we used a transition model to describe the probability of moving from one state to the other. Th transition model was stored in a matrix `T` and used to find the utility function U^* and the best policy π^*.

- Model free method를 제시함 (Model free ~ transition model free를 의미함)
- The second thing we miss is the reward function R(s) which gives to the agent the reward associated to a particular state.
-
-

https://mpatacchiola.github.io/blog/2017/01/15/dissecting-reinforcement-learning-2.html

II. Actor-Critics Method

1. Basics on Actor-Critics Method

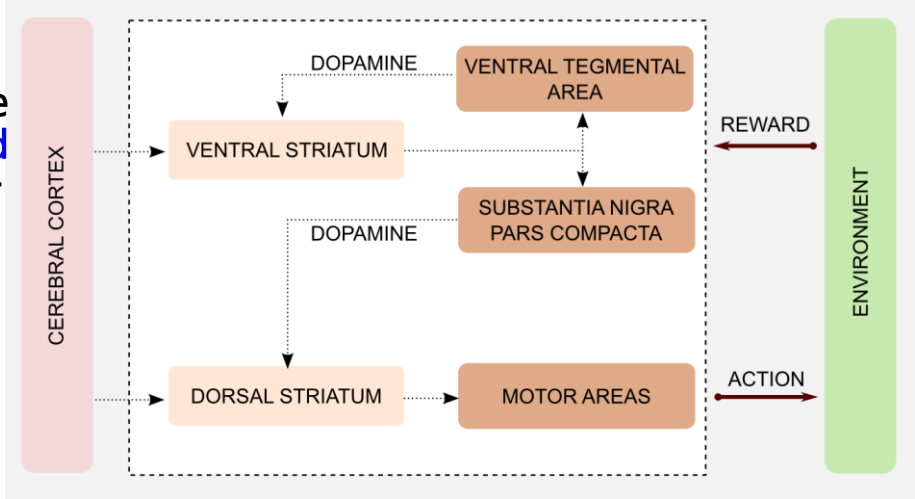
https://mpatacchiola.github.io/blog/2016/12/09/dissecting-reinforcement-learning.html

Actor-Critic Method

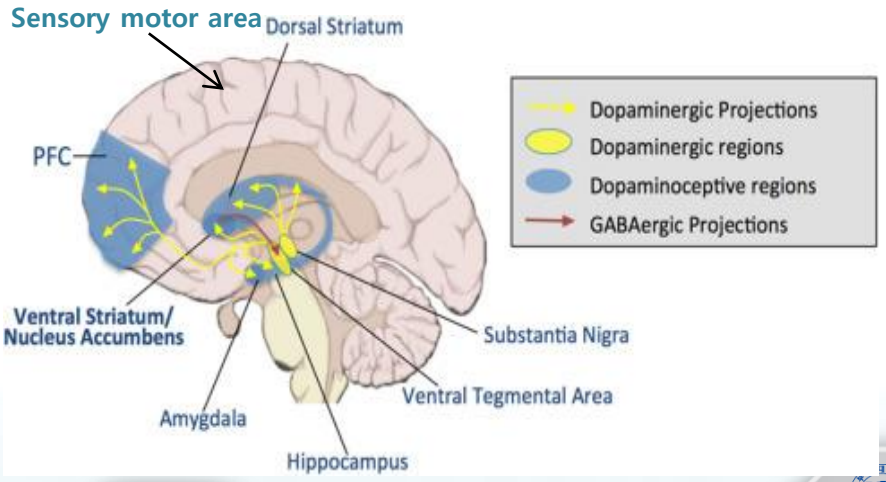
Basal ganglia(기저핵)에서의 인식 기능

- Group 1: Ventral striatum, substantia nigra, ventral tegmental area → This group can evaluate the saliency of a stimulus based on the associated reward. At the same time it can estimate an error measure comparing the result of the action and the direct consequences, and use this value to calibrate an executor. "Critic"

- Group 2: Dorsal striatum and motor areas → The second group has direct access to actions but no way to estimate the utility of a stimulus, because of that I will call it the actor. "Actor"

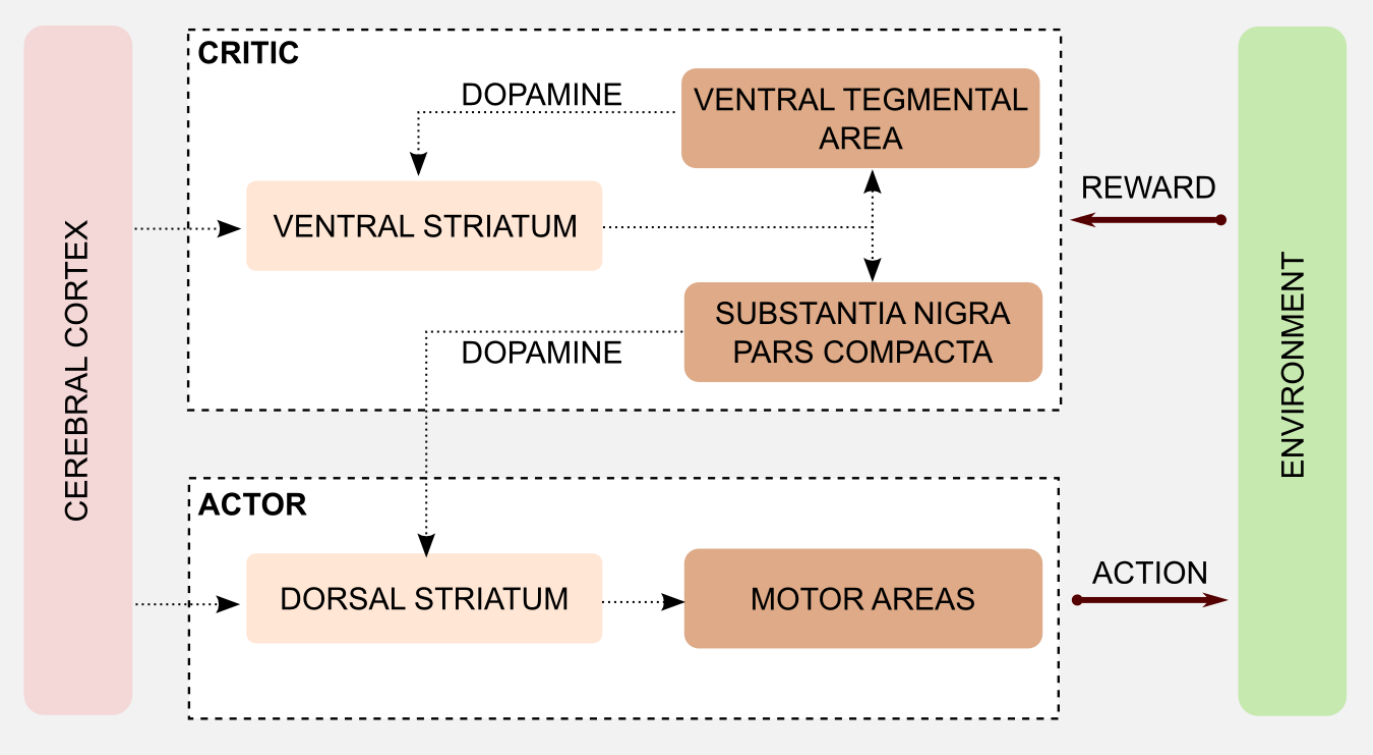


- 배쪽 피개부: ventral tegmental area
- **Ventral striatum: 배쪽 선조체**
- 흑질(substantia nigra) 밀집부(pars compacta).
- **등쪽선조: Dorsal Striatum**
- 운동영역: Motor Area

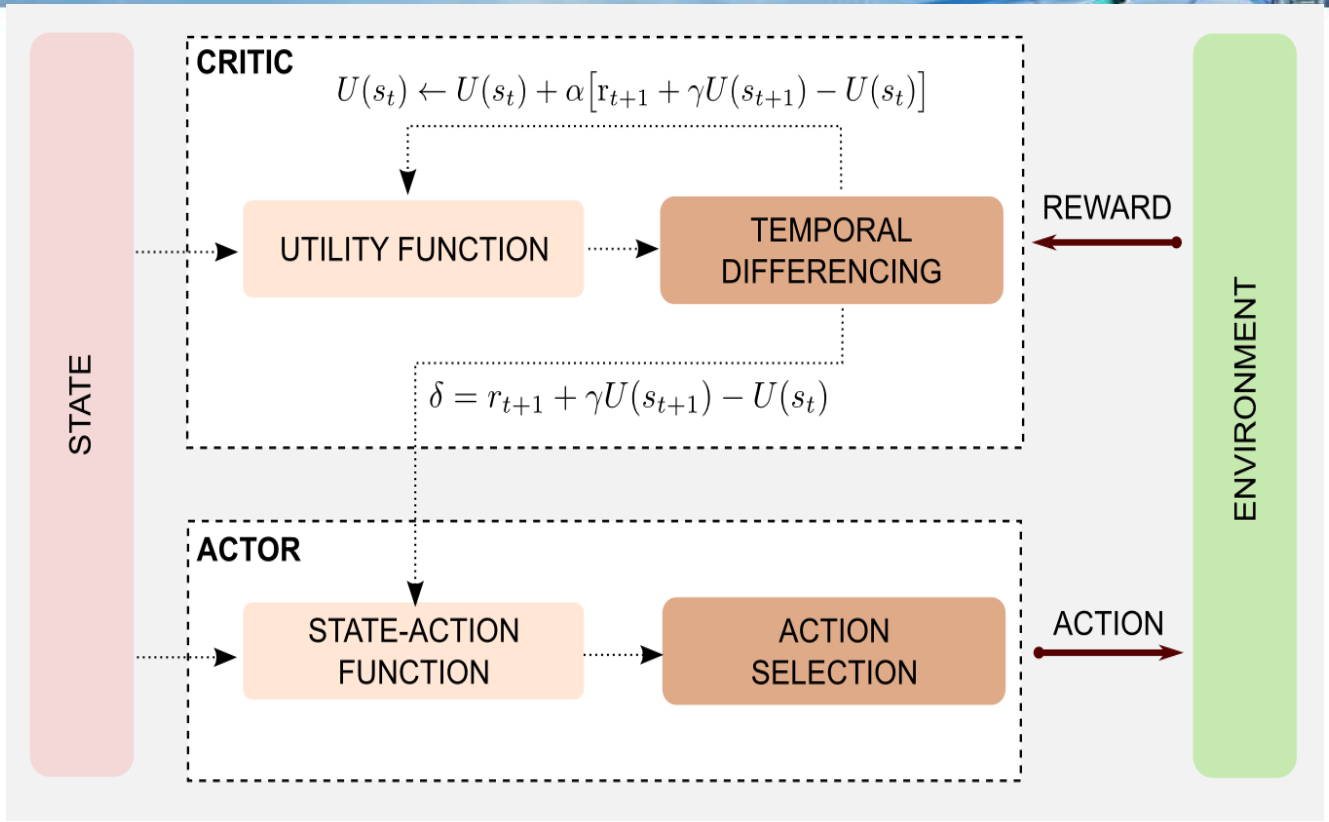


Actor-Critic Method

- **The interaction between actor and critic** has an important role in learning

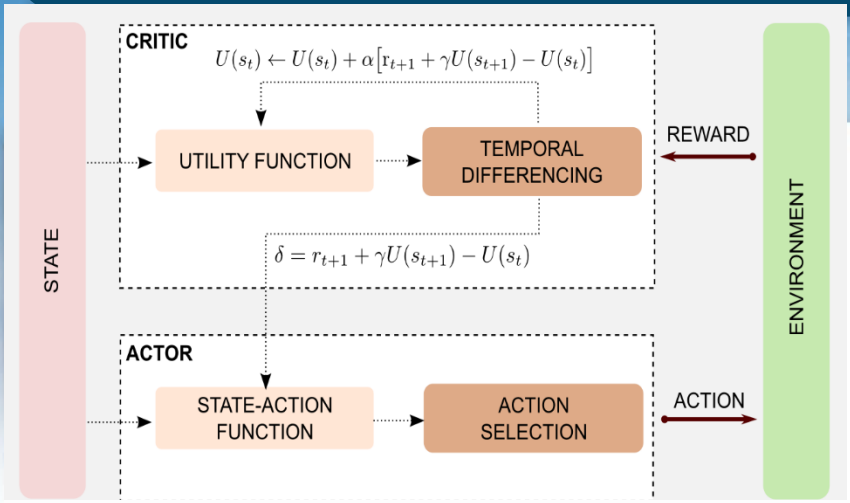


Making Actor-Critic Method from the Neural Version



We can summarise the steps of the AC algorithm as follow:

1. Produce the action a_t for the current state s_t
2. Observe next state s_{t+1} and the reward r
3. Update the utility of state s_t (critic)
4. Update the probability of the action using δ (actor)



In step 1, the agent produces an action following the current policy. In the previous posts I used an ϵ-greedy strategy to select the action and to update the policy. Here I will select a certain action using a softmax function:

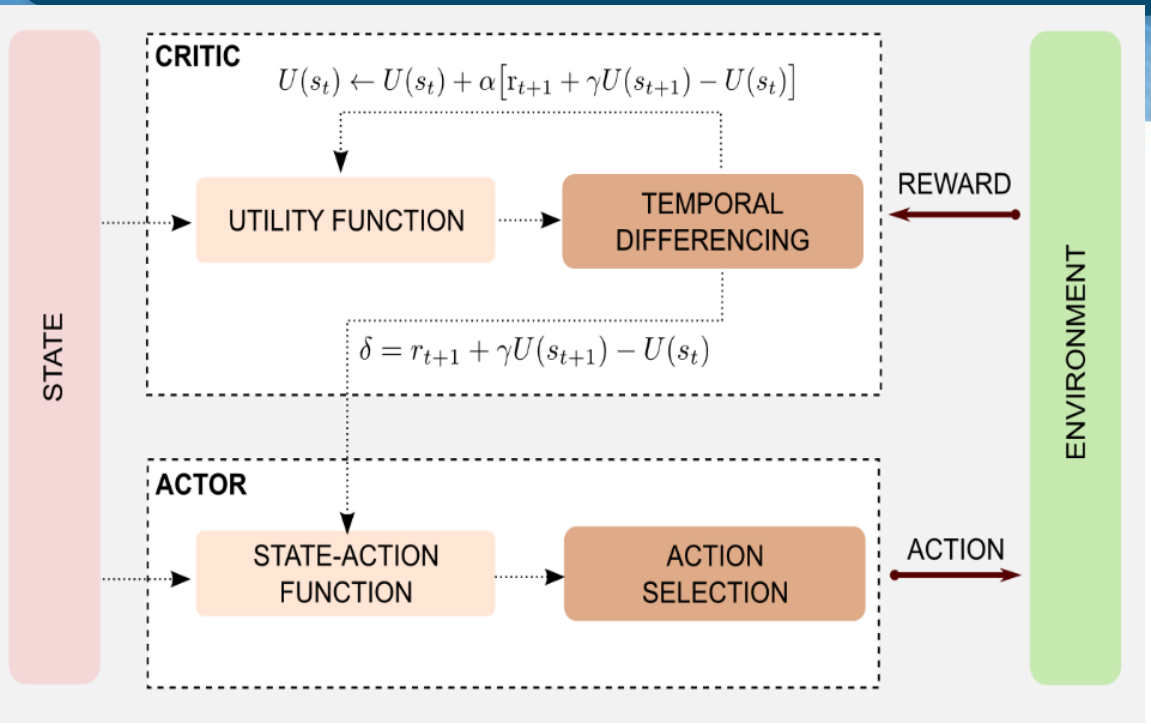
$$P\{a_t = a | s_t = s\} = \frac{e^{p(s,a)}}{\sum_b e^{p(s,b)}}$$

After the action we observe the new state and the reward (step 2). In step 3 we plug the reward, the utility of s_t and s_{t+1} in the standard update formula used in TD(0) (see third post):

$$U(s_t) \leftarrow U(s_t) + \alpha\left[r_{t+1} + \gamma U(s_{t+1}) - U(s_t)\right]$$

In step 4 we use the error estimation δ to update the policy. In practical terms step 4 consists in strengthening or weakening the probability of the action using the error δ and a positive step-size parameter β:

$$p(s_t, a_t) \leftarrow p(s_t, a_t) + \beta\delta_t$$

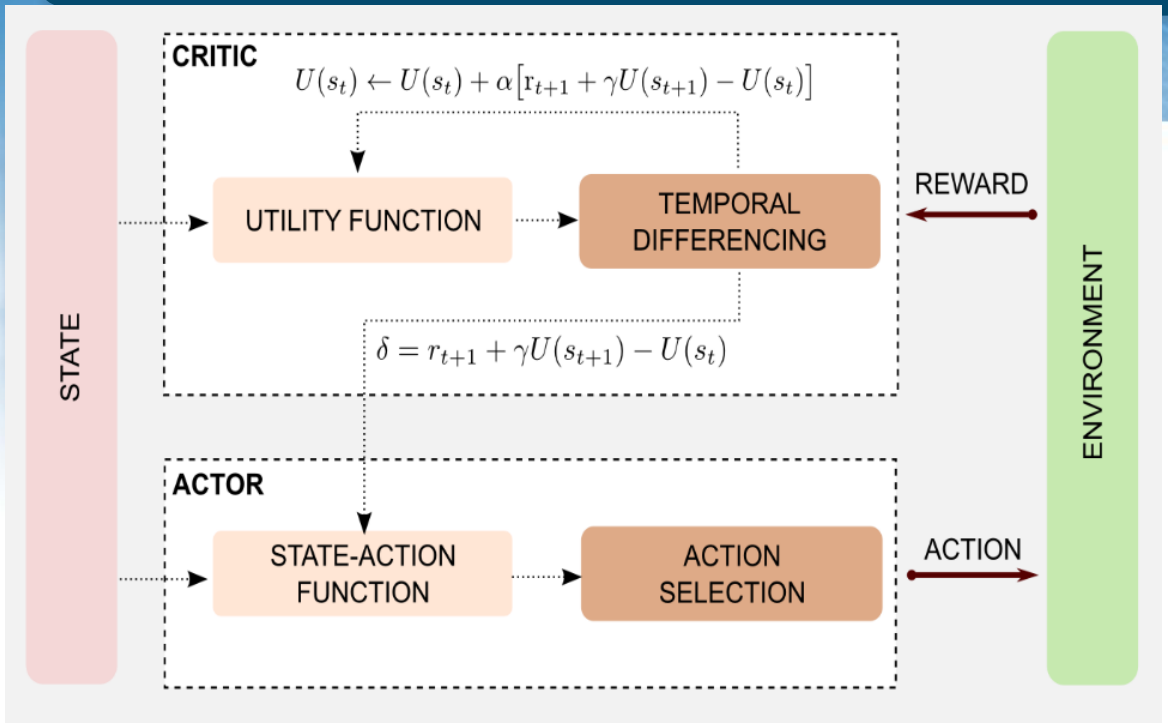


Like in the TD case, we can integrate the **eligibility traces** mechanism (see third post) in AC methods. However in the AC case we need two set of traces, one for the actor and one for the critic. For the **critic** we need to store a trace for each state and update them as follow:

$$e_t(s) = \begin{cases} \gamma\lambda e_{t-1}(s) & \text{if } s \neq s_t; \\ \gamma\lambda e_{t-1}(s) + 1 & \text{if } s = s_t; \end{cases}$$

There is nothing different from the TD(λ) method I introduced in the third post. Once we estimated the trace we can update the state as follow:

$$U(s_t) \leftarrow U(s_t) + \alpha\delta_t e_t(s)$$



For the actor we have to store a trace for each state-action pair, similarly to SARSA and Q-learning. The traces can be updated as follow:

$$e_t(s,a) = \begin{cases} \gamma\lambda e_{t-1}(s,a) + 1 & \text{if } s = s_t \text{ and } a = a_t; \\ \gamma\lambda e_{t-1}(s,a) & \text{otherwise;} \end{cases}$$

Finally the probability of choosing an action is updated as follow:

$$p(s_t, a_t) \leftarrow p(s_t, a_t) + \alpha\delta_t e_t(s)$$

Great, we obtained our generic computational model to use in a standard reinforcement learning scenario.

감사합니다

Q & A

Temporal Differencing(TD) Learning: 시간차 학습

- **TD learning solves some of the problem arising in MC learning**
  - Using MC methods it is necessary to wait until the end of the episode before updating the utility function.
  - This is a serious problem because some applications can have very long episodes and delaying learning until the end is too slow.
  - Moreover the termination of the episode is not always guaranteed. We will see how TD methods solve these issues.

https://mpatacchiola.github.io/blog/2017/01/29/dissecting-reinforcement-learning-3.html