about Twin Delayed DDPG (TD3) for Reinforcement Learning

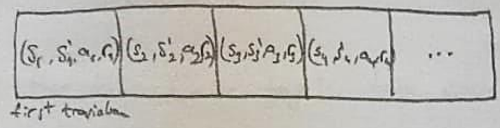
By Oğuzhan Bozoğlu

Twin-Delayed DDPG (TD3)

Bu algoritma modeli, çevrim içi, politikadışı (off-policy) bir RL modelidir. Bir TD3 ajanı, beklenen kümülatif uzun vadeli ödülü en üst düzeye çıkaran optimal bir politikayı arayan actor-critic RL örneğidir.

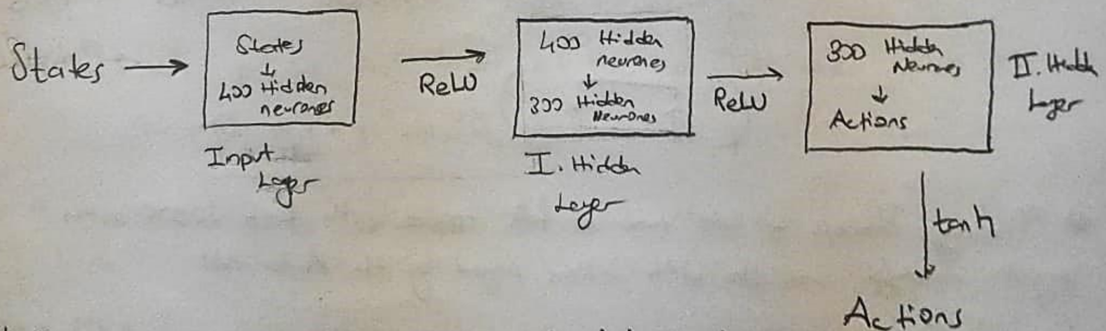
Initialization

Step 1: We initialize the Experience Replay memory, with a size of 20000. We will populated it with each new transition.



– Experience Replay Memory –

Step 2: We build one neural network for the Actor model and one neural network for the Actor target. (Yani 2 sinir ağı oluşturduk ve bunlar birbirinin aynısı)



Yukarıdaki sinir ağı modeli yazarlar tarafından önerilmiştir. Bu model hem Actor model hem de Actor target için yapılır.

about Twin Delayed DDPG (TD3) for Reinforcement Learning

By Oğuzhan Bozoğlu

Step 3: We build two neural networks for the two Critic models and two neural networks for the two Critic targets (4 adet kritik sinir ağı inşa edelim.)

States → | States + Actions ↓ 400 Hidden neurones | → ReLU | 400 Hidden Neurons ↓ 300 Hidden Neurones | → ReLU | 300 Hidden Neurones ↓ Q-Value | → Q1 Value

Actions ↗

\# Yukarıda bu sinir ağı modelleri yazarlar tarafından önerilen, kritik modeli içindir. Bu YSA 4 adet olacaktır.

! Elimizde toplam 6 adet Y.S.A. oldu. Bunların ikisi Actor, dördü Critic'ten oluşmakta. Unutmadan biz Actor'ler politikayı öğrenecek, Critic'ler Q değerlerini öğrenecek. İzleyeceğimiz yol şu şekilde.!

(Bunlar hep YSA)

Actor Target → Critic Target → Critic Target → Critic Model → Critic Model → Actor Model

\# Training Process – We run a full episode with first 10000 actions played randomly and then with actions played by the Actor model.

Step 4: We sample a batch of transitions (s, s', a, r) from the memory. Then for each element of the batch:

\# Uygulamada olarak 100 batch kullanılacağız.

\# Bunlar hep aynı batchler olacak. Yani her birinin batchi ve 4 adet batch var.

Last States	Next States	Actions	Rewards	
Last State 1	Next State 1	Action 1	Reward 1	I. transition
Last State 2	Next State 2	Action 2	Reward 2	II. transition
...	...	...	...	
Last State 100	Next State 100	Action 100	Reward 100	

about Twin Delayed DDPG (TD3) for Reinforcement Learning

By Oğuzhan Bozoğlu

Step 5: From the next state s', the Actor target plays the next action a'.

Step 6: We add Gaussian noise to this next action a' and we clamp it in a range of values supported by the environment.
(Ajan farklı aksiyonları keşfetmesi için gürültü ekliyoruz)

$$\tilde{a} \leftarrow \pi_{\phi'}(s') + \overset{\nearrow \text{noise}}{\epsilon}, \quad \epsilon \sim \text{clip}(\mathcal{N}(0, \tilde{\sigma}), -c, c)$$

$$\tilde{a} \leftarrow \text{clip}(\tilde{a})$$

Step 7: The two Critic targets take each the couple (s', a') as input and return two Q-values $Q_{t_1}(s', a')$ and $Q_{t_2}(s', a')$ as outputs

Step 8: We keep the minimum of those two Q-values: $\min(Q_{t_1}, Q_{t_2})$
It represents the approximated value of the next state.
(min alınca aşırı tahmini önler)

Step 9: We get the final target of the two Critic models, which is:

$$\underbrace{Q_t = r + \gamma * \min(Q_{t_1}, Q_{t_2})}$$ where γ is the discount factor.

Yukarıdaki formülü Q-func.'dan türettik. Max değer yerine min ile Q_t'leri inceledik.

Step 10: The two Critic models take each the couple (s, a) as input and return two Q-values $Q_1(s, a)$ and $Q_2(s, a)$ as outputs.
(İki Critic model de iki farklı tahmin Q değeri çıkarır ve bunu başlatırız.)

Step 11: We compute the loss coming from the two Critic models:

$$\text{Critic Loss} = \text{MSE\_Loss}(Q_1(s,a), Q_t) + \text{MSE\_Loss}(Q_2(s,a), Q_t)$$

iki Critic model arasındaki kaybı hesaplarız.

about Twin Delayed DDPG (TD3) for Reinforcement Learning

By Oğuzhan Bozoğlu

Step 12: We backpropagate this Critic loss and update the parameters of the two Critic models with a SGD (Stochastic Gradient Descent) optimizer
("Adam" optimizeri ile stokastik gradyan inişi kullanarak Critic kaybı azaltılır)

Training kısmının Q-Learning parti bitti şimdi Policy Learning kısmına geçelim.

* TD3'ün politika öğrenme bölümünün nihai amacı, her durumda (state) beklenen getiriyi en üst düzeye çıkaracak optimal aksiyonları gerçekleştirmek için Actor model'in (Ana politikanın) optimal parametreleri bulmak olduğunu hatırlayın.

* Q-value, bir tahminin değeridir ve elimizde bu var. Q-value ne kadar artarsa, optimum beklenen değere o kadar yaklaşırız.

Step 13: Once every two iterations, we update our Actor model by performing gradient ascent on the output of the first Critic Model:

policy

$$\nabla_\phi J(\phi) = N^{-1} \sum \nabla_a Q_{\theta_1}(s, a)\big|_{a=\pi_\phi(s)} \nabla_\phi \pi_\phi(s)$$

state
policy parameter

where ϕ and θ_1 are resp. the weights of the Actor and the Critic.
(Her iki adımda ilk Critic modelin çıktısında gradyan yükselişi ile Actor modeli güncelleriz.)

Step 14: Still once every two iterations, we update the weights of the Actor target by Polyak averaging (gradient descent gibi bir optimizasyon yolu.):

$$\theta_i' \leftarrow \tau\theta_i + (1-\tau)\theta_i'$$

teta tau

Step 15: Still once every two iterations, we update the weights of the Critic target by Polyak averaging:

$$\phi' \leftarrow \tau\phi + (1-\tau)\phi'$$

– The End –
(O.B.)