Twin-Delayed Deep Deterministic (TD3) Policy Gradient Agents

The twin-delayed deep deterministic policy gradient (TD3) algorithm is an off-policy reinforcement learning method. A TD3 agent is an actor-critic reinforcement learning agent that searches for an optimal policy that maximizes the expected cumulative long-term reward.

The twin-delayed deep deterministic policy gradient (TD3) algorithm is an off-policy reinforcement learning method. A TD3 agent is an actor-critic reinforcement learning agent that searches for an optimal policy that maximizes the expected cumulative long-term reward.

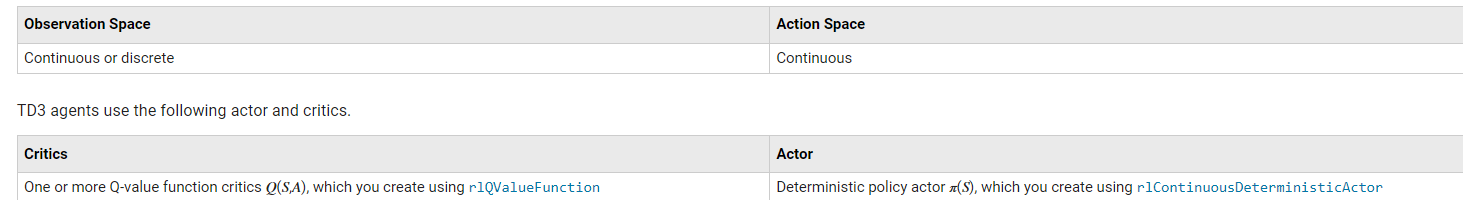
**The TD3 algorithm is an extension of the DDPG algorithm.** DDPG agents can overestimate value functions, which can produce suboptimal policies. To reduce value function overestimation, the TD3 algorithm includes the following modifications of the DDPG algorithm.

1. A TD3 agent learns two Q-value functions and uses the minimum value function estimate during policy updates.
2. A TD3 agent updates the policy and targets less frequently than the Q functions.
3. When updating the policy, a TD3 agent adds noise to the target action, which makes the policy less likely to exploit actions with high Q-value estimates.

You can use a TD3 agent to implement one of the following training algorithms, depending on the number of critics you specify.

* TD3 — Train the agent with two Q-value functions. This algorithm implements all three of the preceding modifications.
* Delayed DDPG — Train the agent with a single Q-value function. This algorithm trains a DDPG agent with target policy smoothing and delayed policy and target updates.

TD3 agents can be trained in environments with the following observation and action spaces.



During training, a TD3 agent:

* Updates the actor and critic properties at each time step during learning.
* Stores past experiences using a circular experience buffer. The agent updates the actor and critic using a mini-batch of experiences randomly sampled from the buffer.
* Perturbs the action chosen by the policy using a stochastic noise model at each training step.

### Actor and Critic Functions

To estimate the policy and value function, a TD3 agent maintains the following function approximators:

* Deterministic actor π(S;θ) — The actor, with parameters θ, takes observation S and returns the corresponding action that maximizes the long-term reward.
* Target actor πt(S;θt) — To improve the stability of the optimization, the agent periodically updates the target actor parameters θt using the latest actor parameter values.
* One or two Q-value critics Qk(S,A;ϕk) — The critics, each with different parameters ϕk, take observation S and action A as inputs and returns the corresponding expectation of the long-term reward.
* One or two target critics Qtk(S,A;ϕtk) — To improve the stability of the optimization, the agent periodically updates the target critic parameters ϕtk using the latest corresponding critic parameter values. The number of target critics matches the number of critics.

Both π(S;θ) and πt(S;θt) have the same structure and parameterization.

For each critic, Qk(S,A;ϕk) and Qtk(S,A;ϕtk) have the same structure and parameterization.

When using two critics, Q1(S,A;ϕ1) and Q2(S,A;ϕ2), each critic can have a different structure, though TD3 works best when the critics have the same structure. When the critics have the same structure, they must have different initial parameter values.

During training, the agent tunes the parameter values in θ. After training, the parameters remain at their tuned value and the trained actor function approximator is stored in π(S).

### Training Algorithm

TD3 agents use the following training algorithm, in which they update their actor and critic models at each time step. To configure the training algorithm, specify options using an rlTD3AgentOptions object. Here, K = 2 is the number of critics and k is the critic index.

* Initialize each critic Qk(S,A;ϕk) with random parameter values ϕk, and initialize each target critic with the same random parameter values: *ϕtk*=*ϕk*.
* Initialize the actor π(S;θ) with random parameter values θ, and initialize the target actor with the same parameter values: *θt*=*θ*.
* For each training time step:
  1. For the current observation S, select action A = π(S;θ) + N, where N is stochastic noise from the noise model. To configure the noise model, use the ExplorationModel option.
  2. Execute action A. Observe the reward R and next observation S'.
  3. Store the experience (S,A,R,S') in the experience buffer.
  4. Sample a random mini-batch of M experiences (Si,Ai,Ri,S'i) from the experience buffer. To specify M, use the MiniBatchSize option.
  5. If S'i is a terminal state, set the value function target yi to Ri. Otherwise, set it to

*yi*=*Ri*+*γ*∗min*k*(*Qtk*(*Si*′,clip(*πt*(*Si*′;*θt*)+*ε*);*ϕtk*))

The value function target is the sum of the experience reward Ri and the minimum discounted future reward from the critics. To specify the discount factor γ, use the DiscountFactor option.

To compute the cumulative reward, the agent first computes a next action by passing the next observation S'i from the sampled experience to the target actor. Then, the agent adds noise ε to the computed action using the TargetPolicySmoothModel, and clips the action based on the upper and lower noise limits. The agent finds the cumulative rewards by passing the next action to the target critics.

If you specify a value of NumStepsToLookAhead equal to N, then the N-step return (which adds the rewards of the following N steps and the discounted estimated value of the state that caused the N-th reward) is used to calculate the target yi.

* + 1. At every time training step, update the parameters of each critic by minimizing the loss *Lk* across all sampled experiences.

*Lk*=12*MM*∑*i*=1(*yi*−*Qk*(*Si*,*Ai*;*ϕk*))2

* + 1. Every *D*1 steps, update the actor parameters using the following sampled policy gradient to maximize the expected discounted reward. To set *D*1, use the PolicyUpdateFrequency option.

∇*θJ*≈1*MM*∑*i*=1*GaiGπiGai*=∇*A*min*k*(*Qk*(*Si*,*A*;*ϕ*)) where *A*=*π*(*Si*;*θ*)*Gπi*=∇*θπ*(*Si*;*θ*)

Here, *Gai* is the gradient of the minimum critic output with respect to the action computed by the actor network, and *Gπi* is the gradient of the actor output with respect to the actor parameters. Both gradients are evaluated for observation *Si*.

8. Every *D*2 steps, update the target actor and critics depending on the target update method. To specify *D*2, use the TargetUpdateFrequency option.

For simplicity, the actor and critic updates in this algorithm show a gradient update using basic stochastic gradient descent.

### Target Update Methods

TD3 agents update their target actor and critic parameters using one of the following target update methods.

* **Smoothing** — Update the target parameters at every time step using smoothing factor τ. To specify the smoothing factor, use the TargetSmoothFactor option.

*ϕtk*=*τϕk*+(1−*τ*)*ϕtk* (critic parameters)*θt*=*τθ*+(1−*τ*)*θt*     (actor parameters)

* **Periodic** — Update the target parameters periodically without smoothing (TargetSmoothFactor = 1). To specify the update period, use the TargetUpdateFrequency parameter.

*ϕtk*=*ϕkθt*=*θ*

* **Periodic Smoothing** — Update the target parameters periodically with smoothing.

To configure the target update method, create a rlTD3AgentOptions object, and set the TargetUpdateFrequency and TargetSmoothFactor parameters as shown in the following table.

