

PPO-DDPG-SAC

032190032 - Mine Korkmaz

032190060 - Fatma Büşra Tilki

032190080 - Fatma Nur Ayyıldız

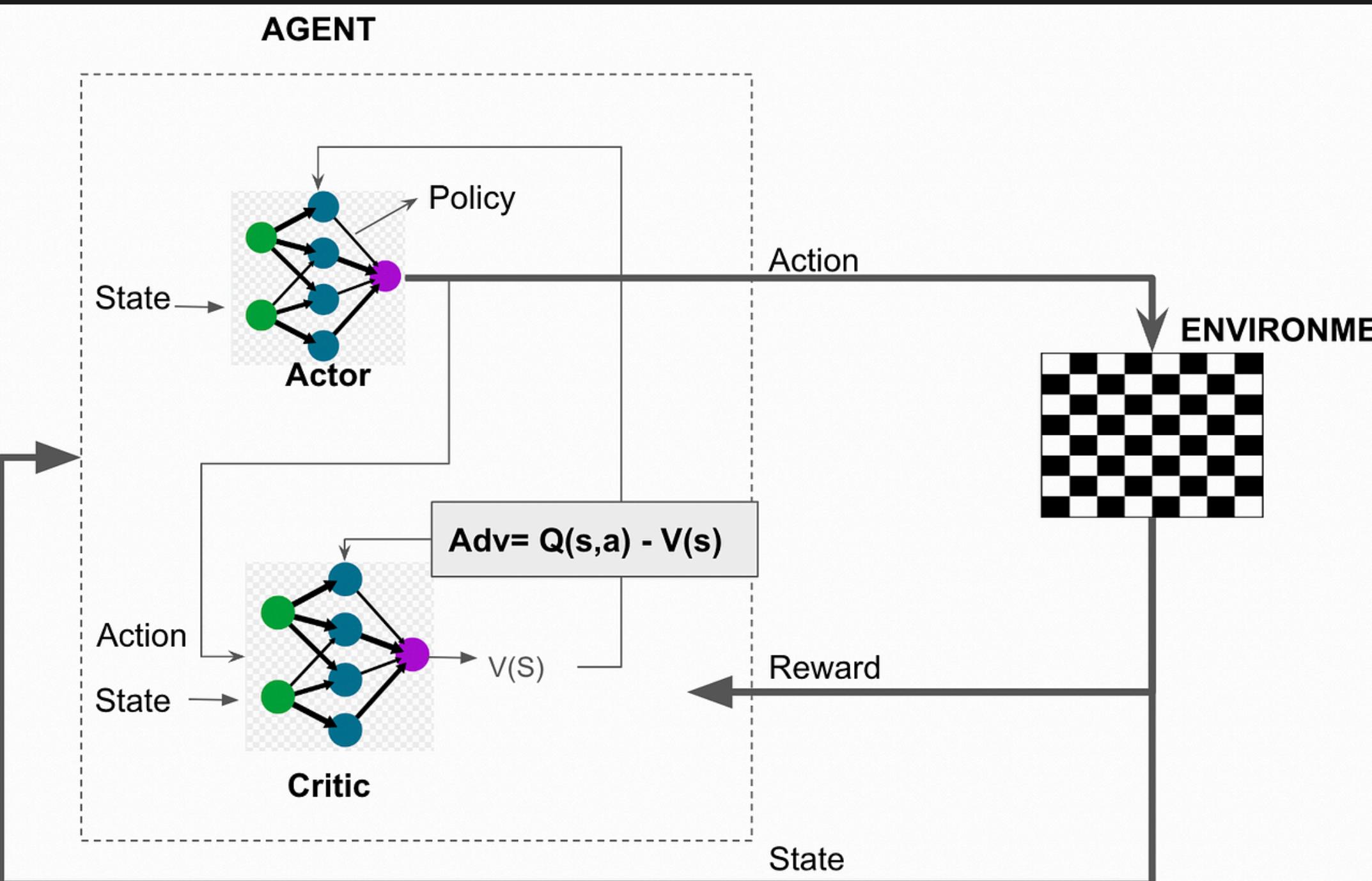
PPO

Proximal Policy Optimization

PPO NEDİR?

Bir bilgisayar ajanının karar işlevini, zor görevleri yerine getirmek için eğiten takviye öğrenme alanında bir algoritmadır. PPO, 2017 yılında John Schulman tarafından geliştirilmiş, Amerikan yapay zeka şirketi OpenAI'de varsayılan takviye öğrenme(RL) algoritması haline gelmiştir.

Gözetimli öğrenmeyle maliyet fonksiyonunu kolayca uygulayabilir, üzerinde gradyan inişi çalıştırabilir ve nispeten az hiperparametre ayarlamasıyla mükemmel sonuçlar elde edeceğimizden oldukça emin olabiliriz. .



PPO, uygulama kolaylığı, örnek karmaşıklığı ve ayarlama kolaylığı arasında bir denge kurar ve her adımda maliyet fonksiyonunu en aza indirirken önceki politikadan sapmanın nispeten küçük olmasını sağlayan bir güncelleme hesaplamaya çalışır. Aslında çevrimiçi verilerden de öğrenen bir politika gradyan yöntemidir. Sadece eğitimde düşük varyans sağlamak için güncellenen politikanın eski politikadan çok farklı olmamasını sağlar. PPO, actor-critic yapısına dayanır: en uygun politikayı öğrenen aktör ve değer fonksiyonunu tahmin eden eleştirmen.

PPO'NUN NASIL ÇALIŞTIĞINA DAİR KISA BİR GENEL BAKIŞ:

1. Politika Gradyan Yöntemleri: PPO, eylemleri durumlarla eşleştirilen bir politika fonksiyonunu doğrudan optimize eden bir politika gradyan yöntemidir. Bu yöntem, bir değer fonksiyonunu tahmin etmeye dayalı algoritmaların farklı olarak, doğrudan ajanların yaptığı eylemleri en iyi hale getirmeye çalışır.
2. Amaç Fonksiyonu: PPO'nun amacı, ajan ile çevre arasındaki etkileşimden elde edilen beklenen toplam ödül最大化 etmektir. Bu amaçla, politika iyileştirme sürecinde doğrudan kullanılabilen bir vekil amaç fonksiyonu oluşturulur. Bu fonksiyon, doğrudan politikayı güncellerek ödüllerini artırmak için kullanılır.
3. Kırılmış Vekil Hedefi: PPO'nun en temel özelliklerinden biri, büyük politika güncellemelerini sınırlayan kıırılmış (clipped) vekil hedefidir. Bu özellik, yeni politika ile eski politika arasında eylem olasılıkları açısından büyük değişikliklerin önüne geçerek güvenli bir güncelleme sağlar. Bu sayede, politika güncellemesi ani değişiklikler yerine küçük ve güvenli adımlarla yapılır.

4. Çoklu Dönemler ve Mini Toplu Güncellemeler: PPO, genellikle ajan ve çevre arasındaki etkileşimlerden oluşan birden fazla dönemi kapsar. Toplanan bu deneyim yörüngeleri, daha sonra mini-batch güncellemeler kullanılarak optimize edilir. Bu işlem, vekil amaç fonksiyonunu her güncelleme turunda en iyi hale getirmek için yapılır.
5. Değer Fonksiyonu Tahmini: PPO, öğrenme sürecindeki belirsizliği azaltmak için değer fonksiyonu tahmini de kullanır. Bu, gradyan hesaplamalarındaki varyansı azaltır, böylece daha kararlı bir eğitim sağlar ve örnek verimliliğini artırır. Değer fonksiyonu, her durumun getireceği ödüllerini tahmin ederek politika güncellemelerini daha doğru hale getirir.
6. Paralelleştirme: PPO, aynı anda birçok farklı ortamdan yörünge toplayabilir, yani paralelleştirilebilir. Bu sayede, eğitim süreci hızlanır ve daha verimli hale gelir. Aynı anda birden fazla çevreyle etkileşime girerek daha fazla veri toplayabilir, bu da algoritmanın daha hızlı öğrenmesine yardımcı olur.

CLIP LOSS FONKSİYONU

$$L^{CLIP}(\theta) = \hat{E}_t[\min(r_t(\theta))\hat{A}_t, \text{clip}(r_t(\theta), 1 - \varepsilon, 1 + \varepsilon)\hat{A}_t]$$

θ : Politika parametresi

E^T : zaman adımları üzerindeki empirik bekleniyi belirtir.

R^T : sırasıyla yeni ve eski politikalar altında olasılık oranıdır

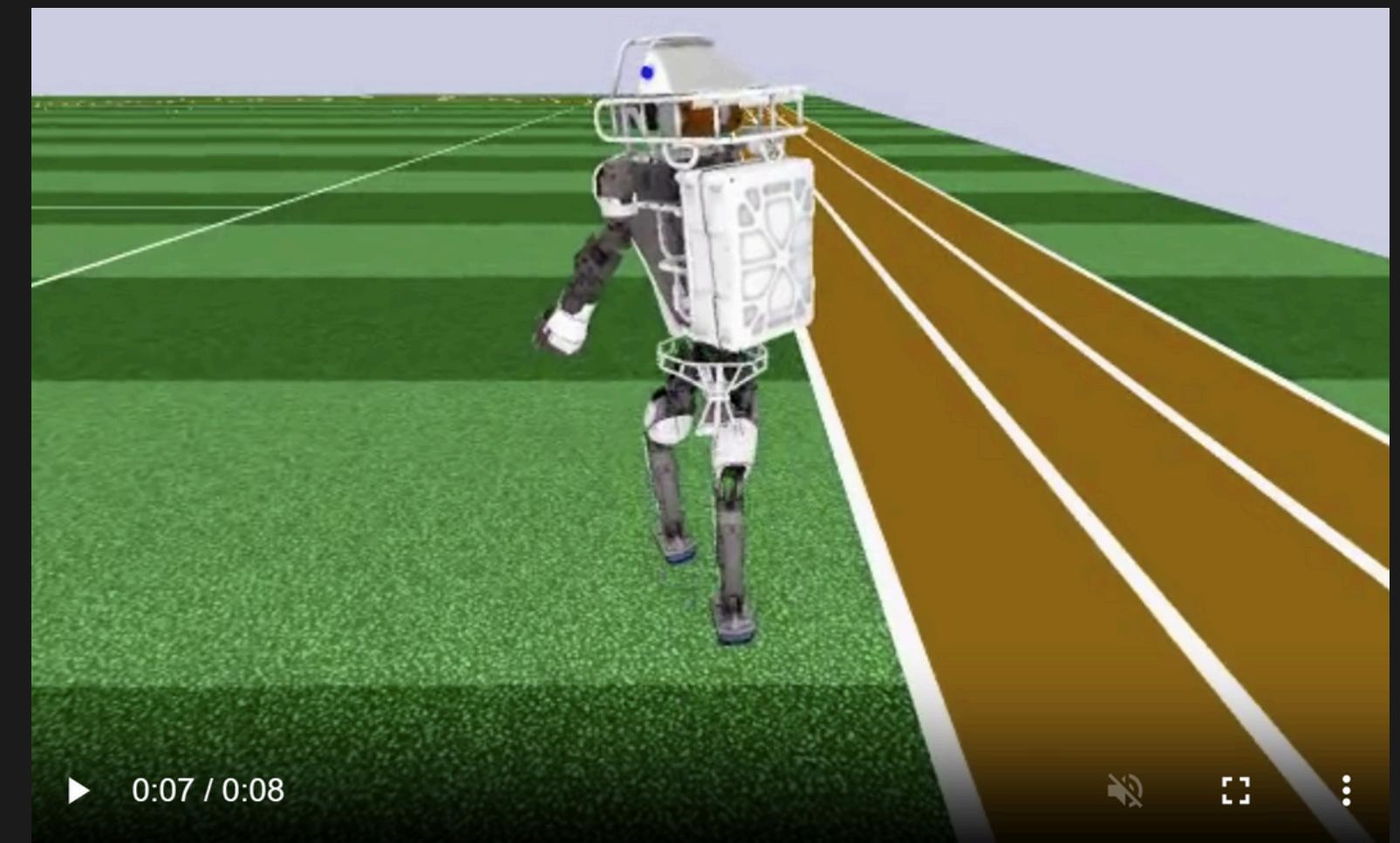
A^T : T zamanındaki tahmini avantajdır

ε : Bir hiperparametredir, genellikle 0.1 veya 0.2

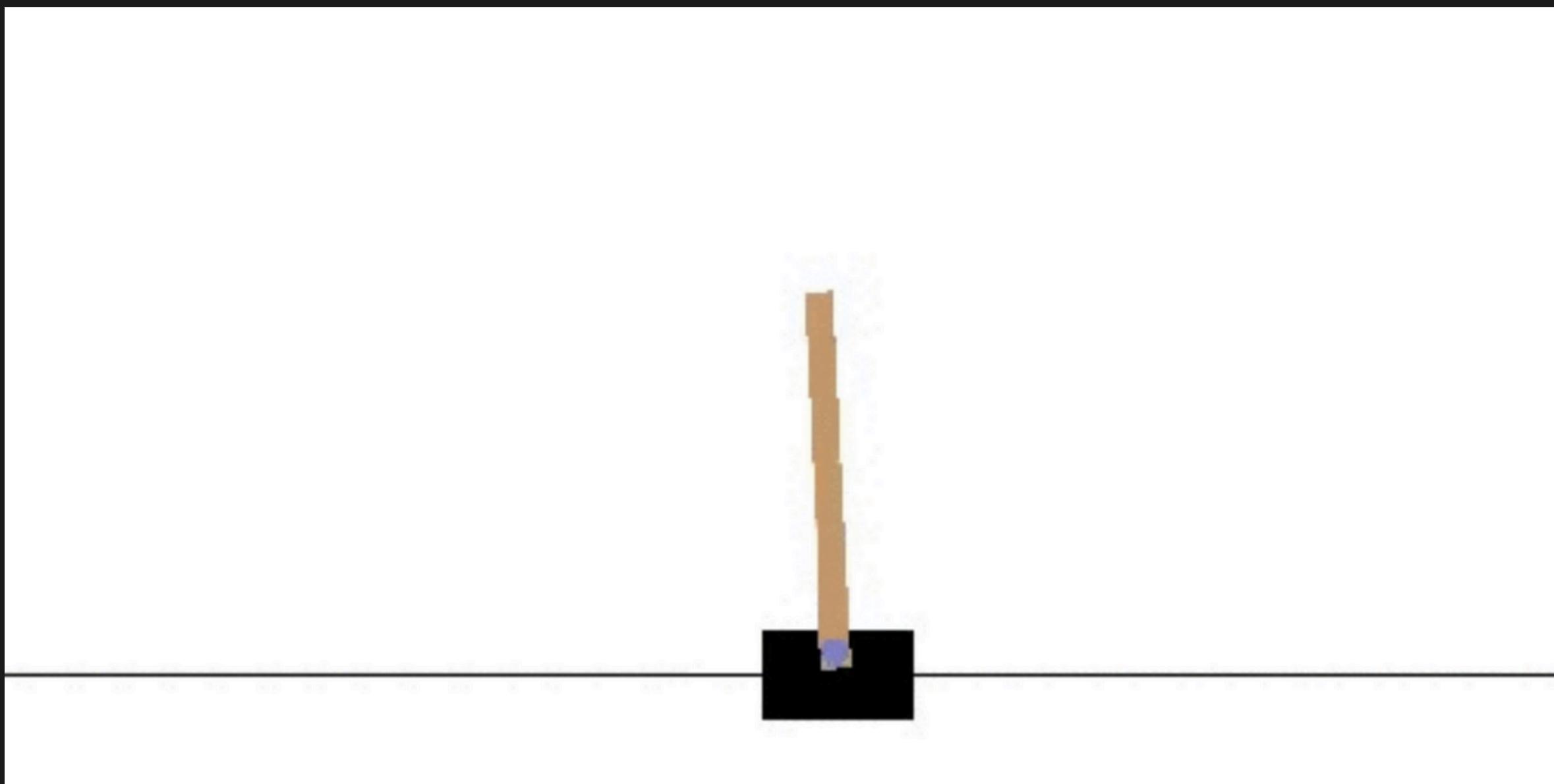
Politika güncellemelerinin stabilitesini artırmak amacıyla bu özel kayıp fonksiyonunu kullanır.

PPO-Clip, olasılık oranı terimine basitçe bir kırpma aralığı uygular ve bu aralık $[1 - \varepsilon, 1 + \varepsilon]$ aralığına kırpılır; burada ε bir hiper-parametredir. Ardından fonksiyon, orijinal oran ile kırpılmış oran arasındaki minimumu alır.

OpenAI testlerinde, bu algoritma sürekli kontrol görevlerinde en iyi performansı göstermiştir ve uygulaması çok daha basit olmasına rağmen ACER'in Atari'deki performansıyla neredeyse eşleşmektedir. Yine OpenAI yaptığı çalışmada Roboscholl içindeki bir ortamda bir robot için; giriş dizileri ajanın eğitildiğinden farklı olsada genelleme yapmayı başarmıştır.



Başka bir çalışmada ise araba üzerindeki direk sabit tutulmaya çalışılıyor. PPO'nun kümülatif ödülü en üst düzeye çıkarmayı hızla öğrendiği gözlemleniyor. Sepet direği ortamı aşırı zorlayıcı olmadıgından, algoritmamız sadece ~20 adımdan sonra ortalama ödülü maksimize eder.



PPO SÖZDE KOD:

1: Başlangıç

- Aktör (actor) ve Kritiği (critic) başlat
- Politika ağı π_θ ve değer ağı V_ϕ 'yi başlat
- Hiperparametreleri ayarla:
 - M (ana döngü sayısı)
 - N (mini-batch güncelleme sayısı)
 - T (her döngüdeki zaman adımı sayısı)
 - γ (indirim faktörü)
 - ϵ (kırpmacı oranı - clip ratio)

2: for i = 1 to M do

- # Politika çalışma ve veri toplama
- Boş yörüngeleri başlat: trajectories = []
- Durum s_0 'yı başlangıç durumu olarak ayarla

for t = 1 to T do

- Aktör ağı π_θ kullanarak bir aksiyon a_t seç
- Aksiyon a_t 'yi çevreye uygula, yeni durum s_{t+1} ve ödül r_t al
- Yörüngelere (s_t, a_t, r_t) olarak kaydet
- Eğer bölüm sona ererse (done), yeni bölüm başlat (state = env.reset)

Avantaj hesaplama

- Toplanan yörüngelere göre her durum için avantaj tahminini hesapla:

$$\hat{A}_t = \sum_{t' = t \text{ to } T} \gamma^{(t' - t)} * r_{t'} - V_\phi(s_t)$$

Politika güncellemeye hazırlık

- Eski politikayı güncelle: $\pi_{\text{old}} \leftarrow \pi_\theta$

3: for $j = 1$ to N do

Aktör güncellemesi - Kırılmış Vekil Hedefi

- Her mini-batch için:

- Eski politika olasılığı $\pi_{\text{old}}(a_t|s_t)$ ve yeni politika olasılığı $\pi_\theta(a_t|s_t)$ hesapla

- Oran (ratio) hesapla: $r_t(\theta) = \pi_\theta(a_t|s_t) / \pi_{\text{old}}(a_t|s_t)$

- Kırılmış kayıp fonksiyonunu (clipped loss) tanımla:

$$L^{\text{clip}}(\theta) = \min(r_t(\theta) * \hat{A}_t, \text{clip}(r_t(\theta), 1 - \varepsilon, 1 + \varepsilon) * \hat{A}_t)$$

- Politika kaybını minimize etmek için aktör ağını güncelle

Kritiği güncelleme

- Değer kaybını (value loss) hesapla:

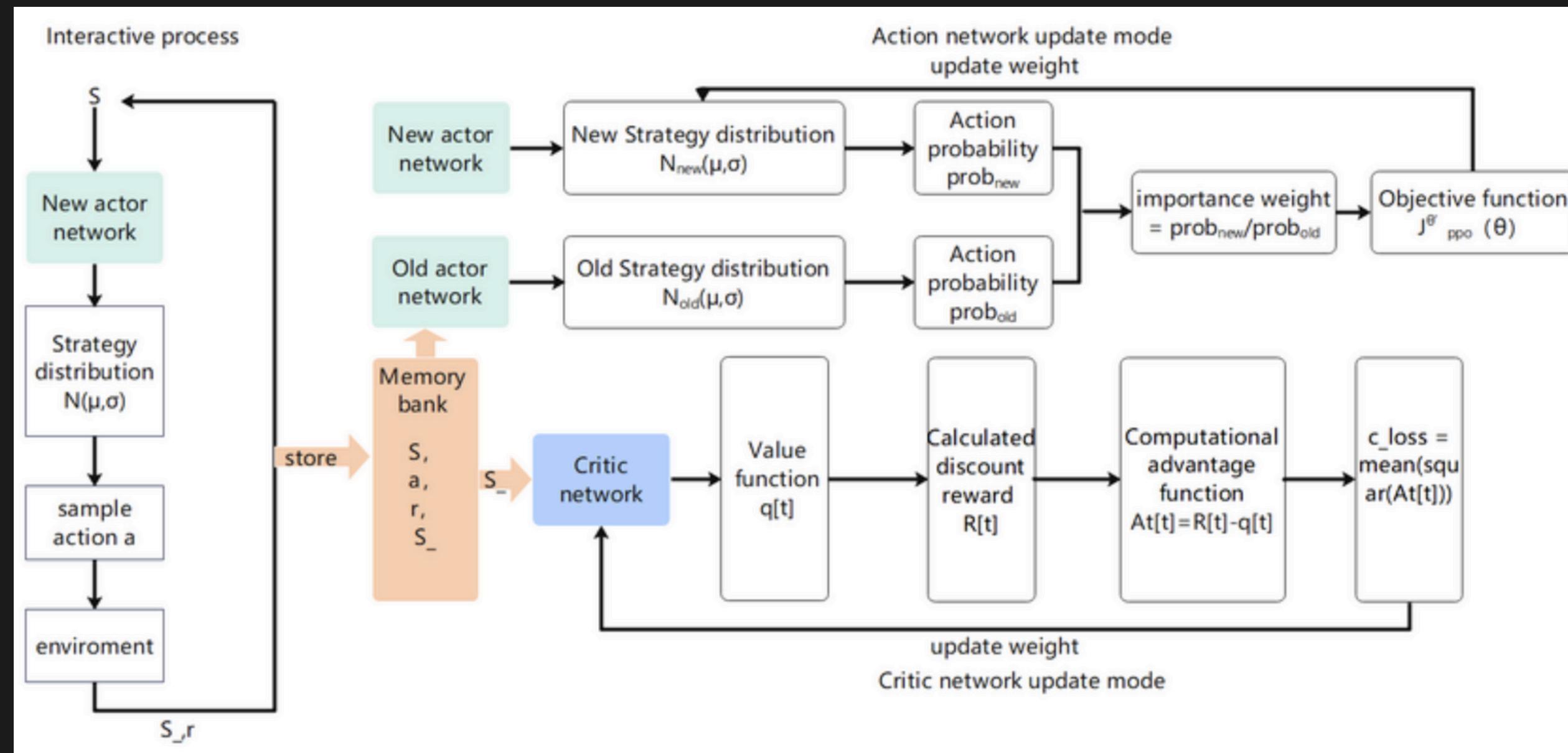
$$L_v(\phi) = \sum (V_\phi(s_t) - R_t)^2$$

- Kritiği optimize etmek için ϕ ağı güncelle

4: end for (N döngüsü)

5: end for (M döngüsü)

PPO AKIŞ DİYAGRAMI



DDPG

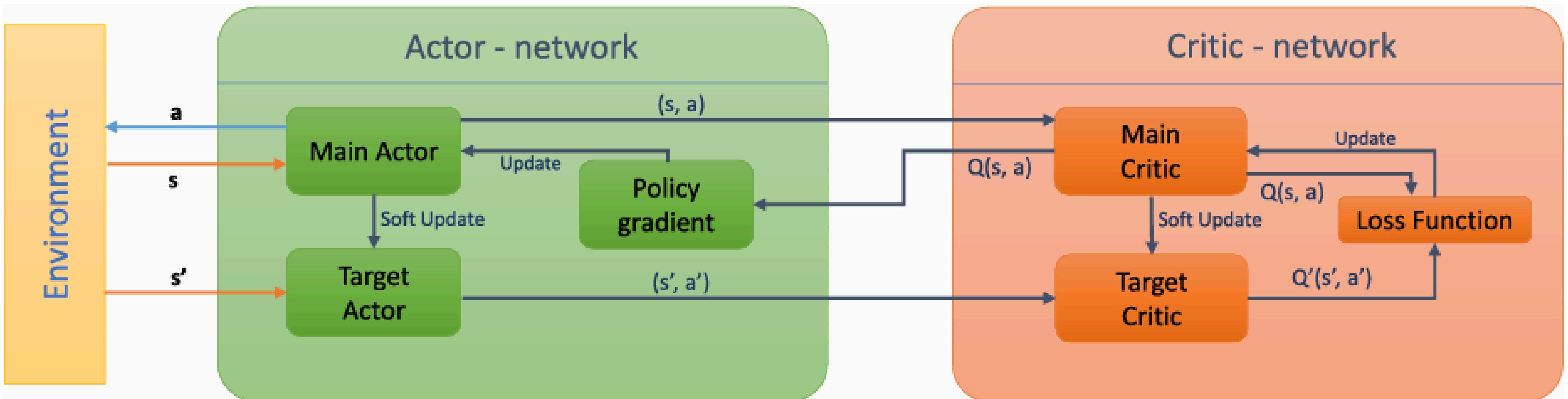
Deep Deterministic Policy Gradient

DDPG

DDPG (Deep Deterministic Policy Gradient), Derin Q-Öğrenme (DQN) ve Deterministik Politika Gradyanı (DPG) algoritmalarını birleştiren, sürekli eylem alanları için tasarlanmış, modelsiz bir aktör-eleştirmen algoritmasıdır. Orijinal DQN ayrık eylem alanlarında çalışırken, DDPG bu yaklaşımı sürekli eylem alanlarına genişletir ve kesin bir politika öğrenmeyi amaçlar.

DDPG

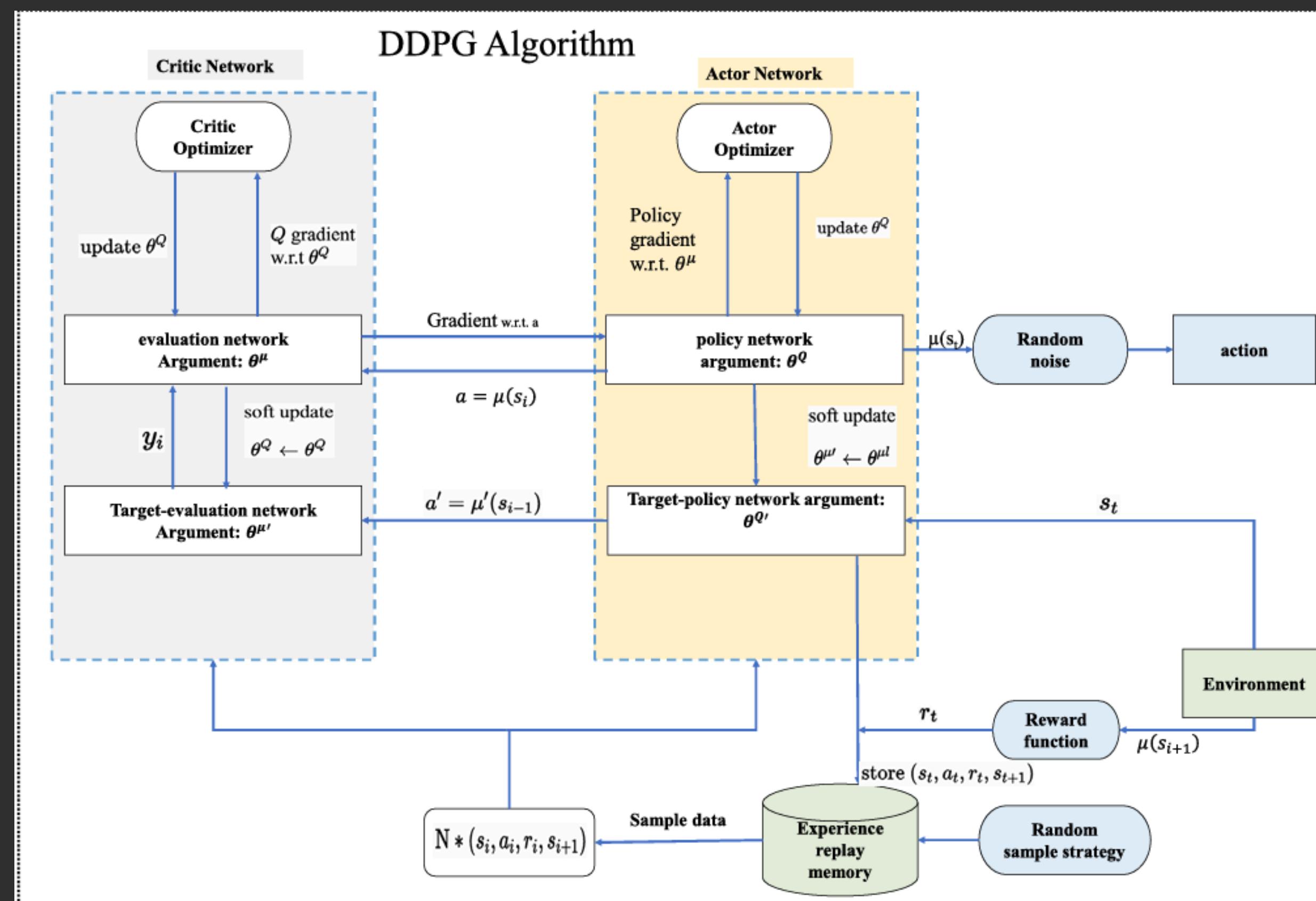
Bu algoritma, aktör ve eleştirmen ağları ile çalışarak en uygun politikayı öğrenir. Aktör ağı, mevcut durumdan (state) bir aksiyon (action) üretir ve bu aksiyon çevreye uygulanır. Eleştirmen ağı ise her durum-aksiyon çiftinin kalitesini (Q değeri) tahmin ederek geri bildirim sağlar. DDPG, stabil bir öğrenme süreci sağlamak için ana ve hedef ağlar (main ve target) arasında yavaş bir güncelleme (soft update) kullanır. DDPG'nin temel amacı, her adımda ajanı daha iyi aksiyonlar seçmeye yönlendirerek sürekli olarak ödülu maksimize etmektir.



DDPG GENEL ÖZELLİKLERİ

1. Aktör ve Eleştirmen Ağları: Aktör ağı, mevcut durum için en iyi eylemi tahmin ederken, eleştirmen ağı bu eylemi değerlendirerek bir Q-değeri üretir. Aktör ağı, deterministik bir politika kullanarak durumları eylemlere dönüştürken, eleştirmen ağı ödül sinyallerini kullanarak gelecekteki toplam ödülü optimize eden Q-değerlerini öğrenir.
2. Deneyim Tekrarı (Experience Replay): Algoritma, geçmiş deneyimlerin bir bellekte saklanması yoluyla eğitilir. Böylece, ağ farklı deneyimlerden öğrenerek istikrar kazanır ve aşırı öğrenme gibi riskler azaltılır.
3. Hedef Ağlar (Target Networks): Hedef ağlar, öğrenme sürecindeki kararsızlıkları önlemek için kullanılır. Bu ağlar, mevcut ağıın gecikmeli bir kopyası olarak çalışır ve güncellemeler sırasında yavaşça ana ağa yaklaşır.
4. Sürekli Eylem Uzayları için Uygunluk: DDPG algoritması, kesintisiz eylem uzaylarında maksimum Q-değerlerini türev tabanlı bir öğrenme kuralı kullanarak hesaplar. Bu özellik, algoritmayı otonom sürüş, robotik kontrol ve enerji yönetimi gibi çeşitli alanlarda çekici kılar.
5. Deterministik Politika : Eylemler üzerinde bir olasılık dağılımı çıkarılan stokastik politikaların aksine, DDPG'nin politikası deterministik ve durumları doğrudan belirli eylemlere eşler.

DDPG Algorithm



DDPG SÖZDE KODU

1. ***Input:*** politika parametreleri (θ), Q-fonksiyonu parametreleri (ϕ) başlat, ve boş bir tekrarlama tamponu D oluştur
2. Hedef parametreleri ana parametrelere eşitle:
 - $\theta_{\text{targ}} \leftarrow \theta$
 - $\phi_{\text{targ}} \leftarrow \phi$
3. ***Repeat***
 4. Durumu Gözlemle ve Eylem Seç: Durum s gözle ve eylemi seç:
 5. Eylemi Uygula: Seçilen aaa eylemini ortamda uygula.
 6. Yeni Durumu ve Ödülü Gözlemle: Yeni durum s' , ödül r ve terminal durum sinyali d (eğer s' terminal durumdaysa) gözleme.

7. Geçisi Kaydet: (s,a,r,s',d) geçişini tekrarlama tamponu D 'ye kaydet.
8. **If** s' terminal durumdaysa, enviroment state sıfırla.
9. **If** güncelleme zamanı geldiyse
10. **for** belirli güncellemeyi **do** (belirli güncellemeyi yapmak için)
 11. Örnekleme Yap: D 'den rastgele bir geçiş örneği grubu seç: $B=\{(s,a,r,s',d)\}$
 12. Hedefleri Hesapla: Seçilen geçişler için hedef değer (y) hesapla.
13. Q-Fonksiyonunu Güncelle: Q-fonksiyonunun parametreleri (ϕ) gradyan inişi yöntemiyle güncelle
14. Politikayı Güncelle: Politika parametreleri (θ) gradyan çıkıştı yöntemiyle güncelle.
15. Hedef Ağları Güncelle: Hedef parametreleri (hem politika hem de Q-fonksiyon için) yumuşak güncelleme ile güncelle
16. **end for**
17. **end if**
18. **Döngüyü Yakınsama Sağlanana Kadar Sürdür.**

SAC

Soft Actor-Critic

SAC - Soft Actor-Critic

SAC Nedir?

Stokastik politikayı politika dışı bir şekilde optimize eden ve stokastik politika optimizasyonu ile DDPG tarzı yaklaşımalar arasında köprü oluşturan bir algoritmadır.

Entropi Düzenlemesi

Politika, beklenen getiri ile politikadaki rastgeleliğin bir ölçüsü olan entropi arasındaki dengeyi maksimize etmek için eğitilmiştir.

SAC (Soft Actor-Critic) Özellikleri

Q-Fonksiyonu Öğrenimi:

- SAC, Q-fonksiyonlarını gerileme yoluyla öğrenir ve MSBE (Mean Squared Bellman Error) minimizasyonu kullanır.
- Hedef Q-fonksiyonları, hedef Q-ağları kullanılarak hesaplanır ve bu ağlar, eğitim süreci boyunca Q-ağ parametrelerinin Polyak ortalaması alınarak elde edilir.
- Çift-Q Hilesi, hedef Q-fonksiyonu hesaplamasında kullanılır, bu da overestimation problemine karşı bir önlem alır.

Hedef Hesaplama:

- SAC, hedef Q-fonksiyonunda, entropi düzenlemesi ekler. Bu, daha keşif odaklı ve istikrarlı bir öğrenme süreci sağlar.

Politika ve Eylemler:

- SAC, stokastik bir politika eğitir. Bu sayede, politika doğal olarak gürültü içerir ve bu stokastiklikten gelen gürültü, politika yumuşatma işlemini sağlar.
- Hedefteki bir sonraki durum eylemi, geçerli politikadan alınır. Bu, hedef politikadan alınan eylemlerden farklıdır.

Entropi-Düzenlemeli RL

- Entropi, rastgele bir değişkenin ne kadar rastgele olduğunu söyleyen bir niceliktir.
- Bir x rastgele değişkeni, P dağılımına sahip olsun. $H(P)$ entropisi, P dağılımına göre aşağıdaki şekilde hesaplanır:

$$H(P) = \mathbb{E}_{x \sim P} [-\log P(x)].$$

- Entropi düzenlemeli RL'de, ajan her zaman adımında politikanın o zamandaki entropisine orantılı bir ödül bonusu alır. Bu, RL problemini şu şekilde değiştirir:

$$H(P) = \mathbb{E}_{x \sim P} [-\log P(x)].$$

SAC Sözde Kodu

1. Başlangıç Değişkenlerini Tanımla:

- İndirim oranı ($\gamma\backslash\gamma$)
- Batch güncelleme sayısı (BBB)
- Öğrenme oranı ($\lambda\backslash\lambda$)
- Sıcaklık ($\alpha\backslash\alpha$)

2. Buffer'ı Başlat:

- Deneyimler için buffer D başlatılır.

3. Öğrenilebilir Parametreleri Başlat:

- Aksiyon politikası ve Q-fonksiyonu için parametreler ($\theta\backslash\theta$ ve $\psi\backslash\psi$) başlatılır.

4. Ana Döngü:

- Her zaman adımı t için aşağıdaki işlemler gerçekleştirilir:

a. Aksiyon Uygula:

- Politika $\pi\backslash\pi$ üzerinden aksiyon ata_tat seçilir.

b. Çevreden Gözlem Al:

- Çevreden ödül rtr_trt ve yeni durum $st+1s_{\{t+1}st+1$ gözlemlenir.

c. Deneyim Sakla:

- (durum, aksiyon, ödül, yeni durum) deneyimi D buffer'ına eklenir.

5. Batch Güncelleme Döngüsü:

- B sayısında güncelleme için:

a. Mini-Batch Seçimi:

- Buffer D üzerinden bir mini-batch seçilir.

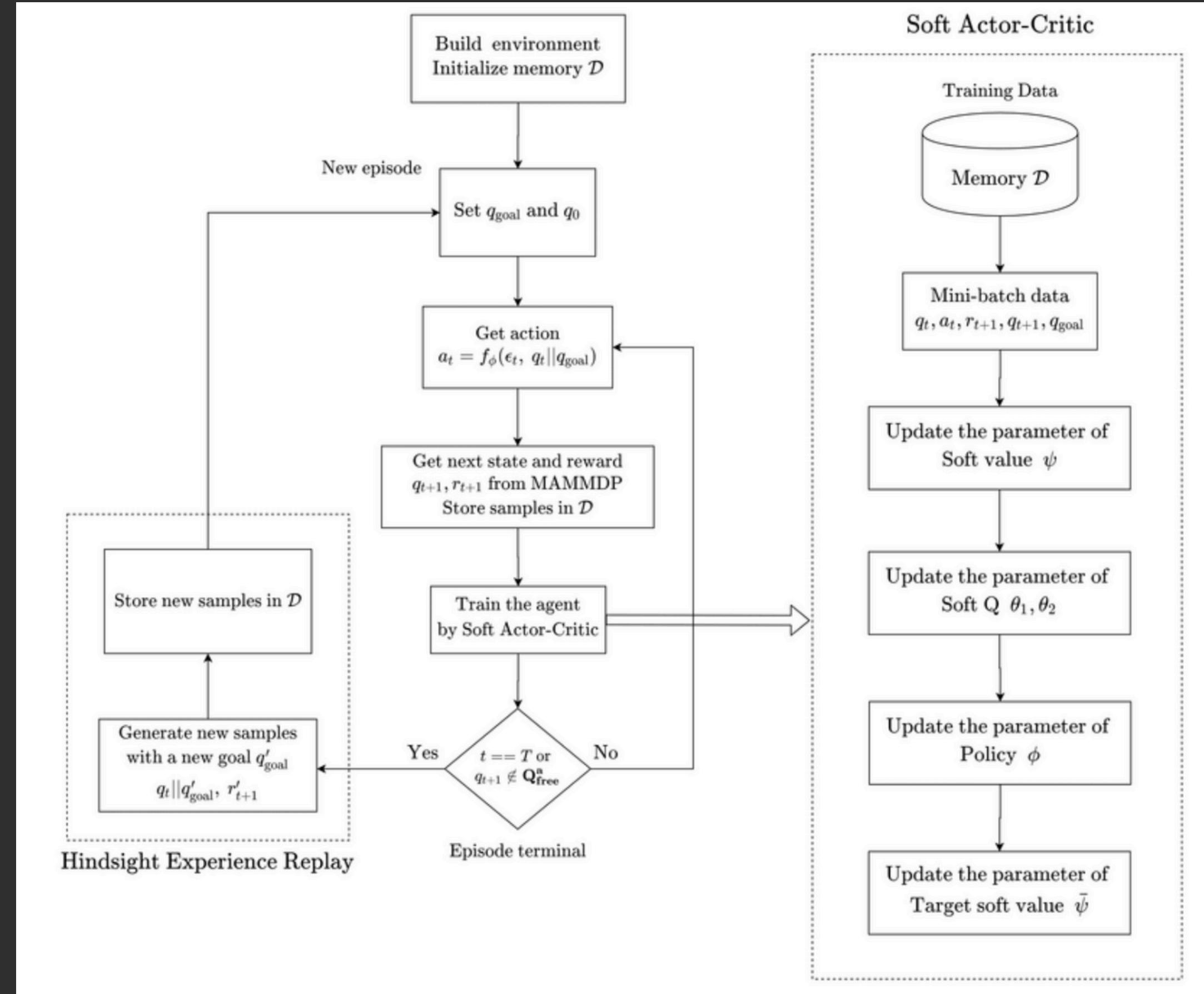
b. Critic Güncellemesi:

- Yeni bir aksiyon a' seçilir.
- Q-fonksiyonu, ödül ve gelecekteki Q-değerleri ile güncellenir.

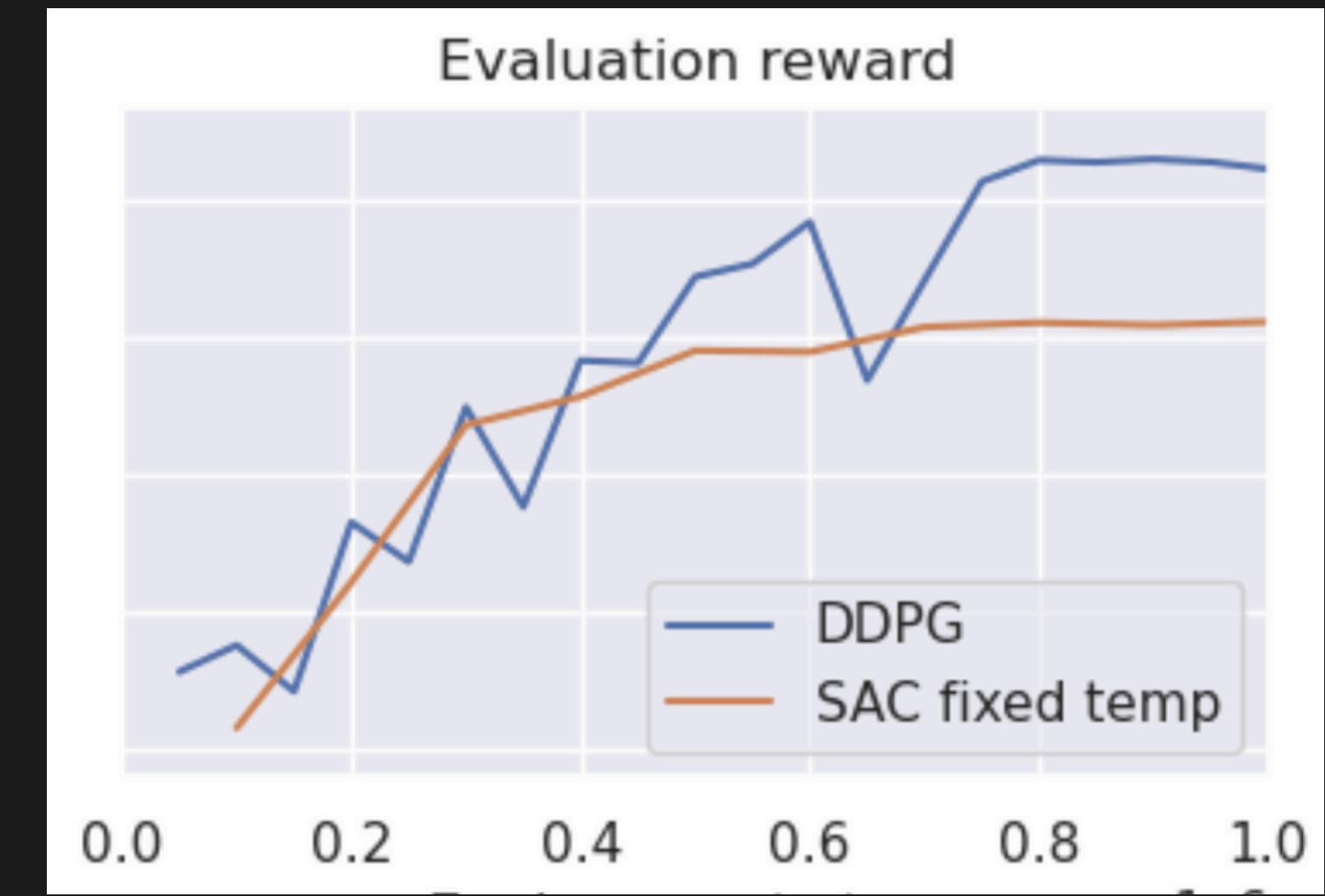
c. Actor Güncellemesi:

- Politika parametreleri θ , öğrenme oranı ve gradienleri kullanılarak güncellenir.

SAC AKIŞ DİYAGRAMI



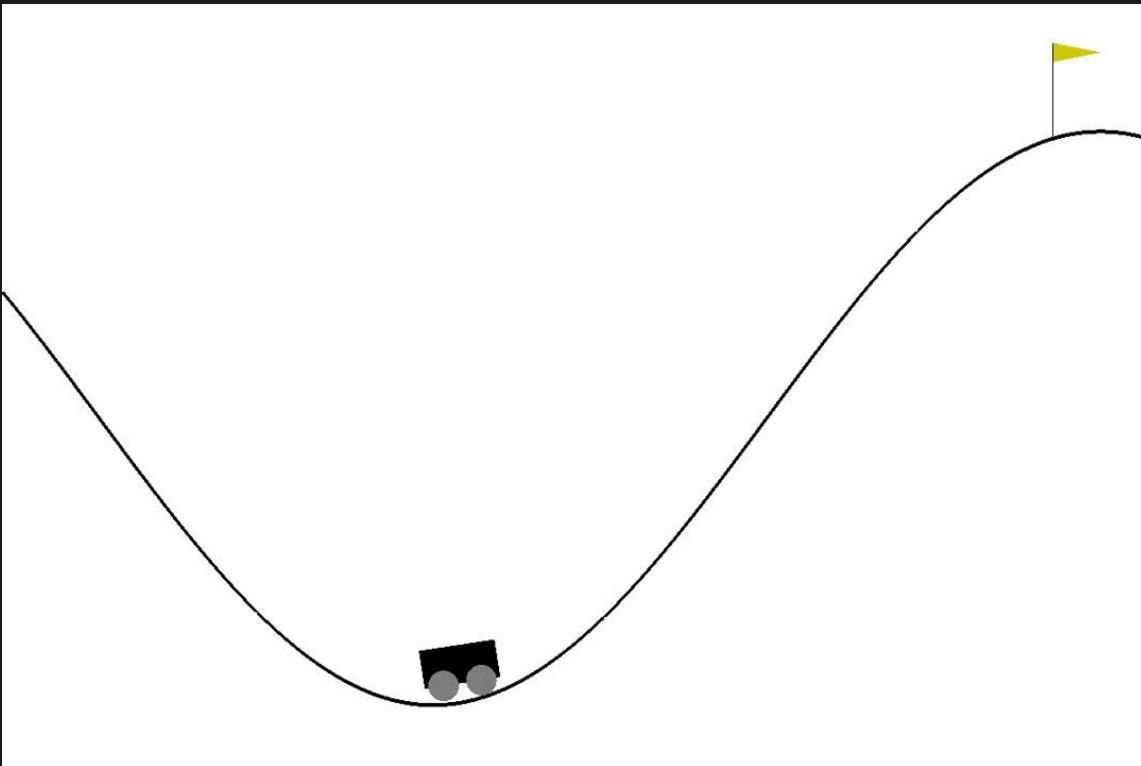
DDPG-SAC



Ignat Georgiev'in Cheetah problemi üzerinde yaptığı çalışmada, SAC iyi sonuç gösteriyor ancak hem eğitim hem de değerlendirme açısından DDPG'ye yaklaşamıyor.

Mountain Car Continuous

Mountain Car MDP, sinüzoidal bir vadinin dibine stokastik olarak yerleştirilmiş bir arabadan oluşan deterministik bir MDP'dir ve tek olası eylemler, arbaya her iki yönde de uygulanabilen ivmelerdir. MDP'nin amacı, arbayı stratejik olarak hızlandırarak sağ tepenin üstündeki hedef durumuna ulaştırmaktır. Gymnasium'da dağ arabası alanının iki versiyonu vardır: biri ayrik eylemlere sahip ve diğerinin sürekli eylemlere sahip. Bu versiyon, sürekli eylemlere sahip olandır.



Gözlem Alanı

Gözlem, elemanlarının aşağıdakilere karşılık geldiği (2,) şeklinde bir ndarray 'dir:

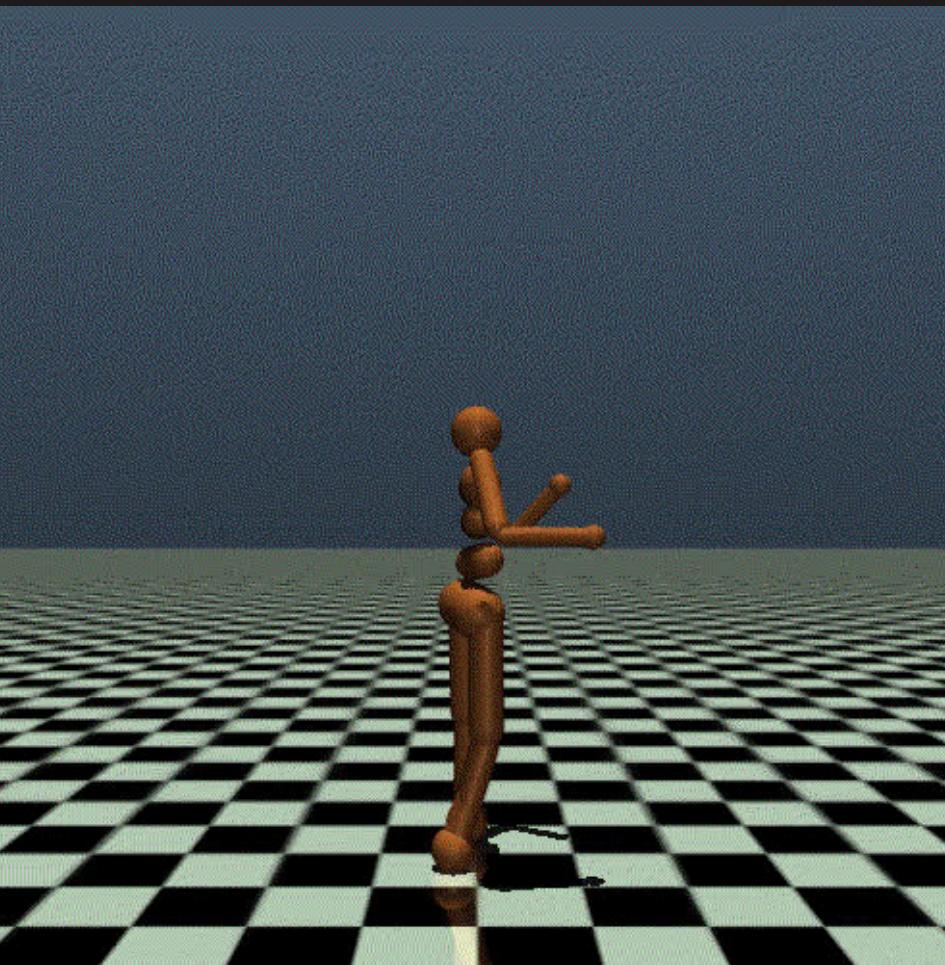
Sayı	Gözlem	Dakika	Maksimum	Birim
0	x ekseni boyunca aracın konumu	-Bilgi	Enf	pozisyon (e)
1	Arabanın hızı	-Bilgi	Enf	pozisyon (e)

Eylem Alanı

Eylem, arabaya uygulanan yönlü kuvveti temsil eden (1,) şeklinde bir ndarray 'dir. Eylem [-1,1] aralığında kırılır ve 0 , 0015'lik bir güçle çarpılır.

Humanoid

3D iki ayaklı robot, bir insanı simüle etmek için tasarlanmıştır. Bir çift bacak ve kol ile bir gövdeye (karın) sahiptir. Bacakların her biri iki bağlantidan oluşur ve böylece kollar (sırasıyla dizleri ve dirsekleri temsil eder). Ortamın amacı, düşmeden mümkün olduğunca hızlı bir şekilde ileriye doğru yürümektir.



Eylem Alanı

Eylem uzayı bir Box(-1, 1, (17,), float32)'dır. Bir eylem, menteşe bağlantılarında uygulanan torkları temsil eder.

Gözem Alanı

Gözlemler, insansı'nın farklı vücut parçalarının konumsal değerlerinden ve ardından bu parçaların hızlarından (türevleri) oluşur ve tüm konumlar tüm hızlardan önce sıralanır.

Kaynakça

- <https://www.gymlibrary.dev/environments>
- <https://spinningup.openai.com/en/latest/algorithms/ddpg.html#id1>
- <https://openai.com/index/openai-baselines-ppo/>
- <https://www.researchgate.net/>
- https://medium-com.translate.goog/@briancpulfer/ppo-intuitive-guide-to-state-of-the-art-reinforcement-learning-410a41cb675b?_x_tr_sl=en&_x_tr_tl=tr&_x_tr_hl=tr&_x_tr_pto=tc
- <https://medium.com/@danushidk507/ppo-algorithm-3b33195de14a>
- https://spinningup-openai-com.translate.goog/en/latest/algorithms/sac.html?_x_tr_sl=en&_x_tr_tl=tr&_x_tr_hl=tr&_x_tr_pto=tc
- Sumiea, Ebrahim Hamid, et al. "Deep deterministic policy gradient algorithm: A systematic review." *Heliyon* (2024).
- <https://medium.com/@amaresh.dm/how-ddpg-deep-deterministic-policy-gradient-algorithms-works-in-reinforcement-learning-117e6a932e68>
- <https://medium.com/intro-to-artificial-intelligence/deep-deterministic-policy-gradient-ddpg-an-off-policy-reinforcement-learning-algorithm-38ca8698131b>