logo içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Bursa Uludağ Üniversitesi**

**Bilgisayar Mühendisliği**

**2024 – 2025 Eğitim Öğretim Yılı Güz Yarıyılı**

**Robot Tasarımı ve Uygulamaları Dersi**

**Dönem Ödevi**

| **İsim** | **Numara** |
| --- | --- |
| Atilla Erdinç | 032190098 |
| Muhammed Ali Gedikli | 032190010 |
| Osman Atalay Kayalar | 032190043 |

**Giriş**

**Mountain Car Continuous**

Mountain Car Continuous probleminde, bir araba iki tepe arasında, yerçekiminin negatif etkisi altında kalan bir vadide sıkışmıştır. Aracın amacı, sağdaki tepeden yukarı tırmanarak belirli bir yüksekliğe ulaşıp vadiyi terk etmektir. Problem zorlayıcıdır çünkü araç motoru, bu eğimi tek başına çıkacak kadar güçlü değildir. Bu yüzden araç, vadi boyunca ivme kazanmak için ileri-geri hareket etmek zorundadır.

Mountain Car Continuous problemi, Mountain Car probleminden şu farklılıklara sahiptir:

1. **Kontrol Sürekliliği**: İki seçenek yerine (-1 veya +1 gibi) sürekli bir eylem uzayı vardır, yani aracın motoruna verilen güç değeri herhangi bir sürekli sayı aralığında olabilir (örneğin, [-1, 1] aralığı).
2. **Eylemler**: Araba, hem hızlanmak hem de yavaşlamak için sürekli bir itme gücü kullanabilir.
3. **Ödül Yapısı**: Genellikle, başarılı bir şekilde sağdaki tepeye ulaşana kadar her adımda negatif bir ödül verilir. Böylece, model arabanın hedefine daha hızlı ulaşması için teşvik edilir.

### Problem Parametreleri ve Gözlemler

* **Durum Uzayı**: Arabanın pozisyonu ve hızı.
* **Eylem Uzayı**: Aracın motoruna verilen güç, sürekli bir değerde olabilir.
* **Başarı Kriteri**: Araba, sağdaki tepenin üst sınırına ulaştığında problem çözülmüş sayılır.
* **Ödül Fonksiyonu**: Her adımda ödül -1 verilir, hedefe ulaşıldığında yüksek bir ödül elde edilir.

**Mountain Car Continuous Probleminde Kullanılan Yöntemler ve Çalışmalar**

1. **Q-Öğrenme (Q-Learning) ve Sürekli D-Q-Öğrenme (Continuous D-Q Learning):**

* Geleneksel Q-öğrenme algoritması, sürekli aksiyon uzayları için yeterli değildir. Bu nedenle, sürekli aksiyon uzayları için geliştirilmiş Deep Q-Network (DQN) algoritmaları gibi varyantlar kullanılır.
* Ancak, Q-öğrenme sürekli aksiyon uzayına doğrudan uygulanamaz. Bu nedenle, genellikle yaklaşım olarak DDPG (Deep Deterministic Policy Gradient) gibi sürekli aksiyonları öğrenme odaklı algoritmalar tercih edilir.

1. **Proksimal Politika Optimizasyonu (PPO)**:

* PPO, politikanın optimizasyonu sırasında yüksek stabilite ve performans sağlayan bir başka güçlü algoritmadır.
* PPO’nun öğrenme stabilitesinin yüksek olması, Mountain Car Continuous gibi sürekli aksiyon içeren karmaşık görevlerde avantaj sağlar.

**3)Soft Actor-Critic (SAC)**:

* SAC, sürekli aksiyon problemlerinde iyi performans gösteren, entropi düzenlemesi ile politikanın keşfetme yeteneğini arttıran bir algoritmadır.
* Mountain Car Continuous probleminde de genellikle başarılı sonuçlar elde eder.

**Mountain Car Continuous Problemi Üzerine Yapılan Çalışmalar**

**Daha Etkili Politika Gradyanı Yöntemleri**:

* Politika gradyanı yöntemleri ile elde edilen başarılar, Mountain Car Continuous gibi görevlerde verimliliği arttırmayı amaçlamaktadır. Bazı çalışmalarda PPO ve SAC gibi algoritmalarla standart politika gradyanı yöntemlerine kıyasla daha yüksek başarılar gözlemlenmiştir.

**Hızlı Keşif ve Optimizasyon**:

* Entropi düzenlemesi ve temsili öğrenme yöntemleriyle keşfi hızlandırmak amacıyla yapılan çalışmalarda, Mountain Car Continuous gibi problemler test alanı olarak kullanılmıştır. Özellikle SAC gibi algoritmalar, entropi düzenlemesi sayesinde daha kapsamlı keşif yaparak öğrenme sürecini hızlandırmıştır.

**Daha Az Epizod Sayısında Başarı**:

* Bazı araştırmalar, çeşitli parametre incelemeleriyle aracın daha az epizodda başarıya ulaşmasını sağlamayı hedeflemiştir. Bunun için algoritmanın ödül fonksiyonları ve keşif oranları optimize edilmiştir. Özellikle DDPG gibi algoritmalar, sürekli aksiyon alanı olduğu için bu tür problemlerle başarılı bir şekilde entegre edilmiştir.

**Method**

1. **Ortamın Tanımlanması ve Başlatılması:**

* MountainCarContinuous ortamı tanımlanır ve başlatılır. Bu ortamda bir araba, sürekli bir aksiyon alanında kontrol edilerek tepeyi tırmanmaya çalışır.

1. **Modellerin Tanımlanması:**

* Her algoritma için bir model tanımlanır (PPO, A2C ve SAC). Bu modeller, MlpPolicy adı verilen bir sinir ağı mimarisini kullanır.

1. **Eğitim Döngüsü:**

* Her model için 5 döngülük bir eğitim süreci uygulanır.
* Her bir döngüde model learn() metodu kullanılarak total\_timesteps (10,000 adım) boyunca eğitilir.
* Eğitim sonrası modelin performansı, bir bölüm (episode) süresince elde edilen ödüllerin toplanması ile değerlendirilir.

1. **Performansın Kaydedilmesi:**

* Her bölümde elde edilen toplam ödül ppo\_rewards, a2c\_rewards, sac\_rewards listelerine kaydedilir. Bu ödüller daha sonra grafik üzerinde karşılaştırılır.

1. **Grafik Çizimi:**

* Her algoritmanın elde ettiği ödüller grafik üzerinde karşılaştırılır ve görselleştirilir.

**Akış Diyagramı**

**Başla**

1. MountainCarContinuous-v0 ortamını oluştur
2. PPO, A2C ve SAC modellerini sırayla tanımla
3. Her bir model için (PPO, A2C, SAC):
4. Eğitim döngüsüne gir (5 kez)
5. Her bir döngü için:
   1. Modeli total\_timesteps kadar eğit
   2. Bölüm başlat
   3. Her adım için (maksimum 200 adım):
      1. Aksiyon tahmin et
      2. Ortamda aksiyonu uygula, yeni gözlemi ve ödülü al
      3. Toplam ödülü güncelle
      4. Eğer bölüm tamamlandıysa çık
   4. Toplam ödülü kaydet
6. Tüm modeller için döngü tamamlandığında: Grafik çiz
7. Bitir

**Algoritmaların Çalışma Mantığı**

#### PPO (Proximal Policy Optimization)

1. Eğitim Başlat: learn fonksiyonu çağrılarak model, belirli bir adım boyunca eğitim yapar.
2. Aksiyon Seçimi: Model mevcut duruma göre bir aksiyon seçer.
3. Çevreden Geri Bildirim: Seçilen aksiyon çevreye uygulanır ve çevre yeni bir durum, ödül ve bitiş durumu sağlar.
4. Politika Güncellemesi: PPO, ödülleri maksimize etmek için küçük adımlarla güncelleme yapar. Aşırı güncellemelerden kaçınmak için özel bir kısıtlama mekanizması (proksimal güncelleme) kullanır.

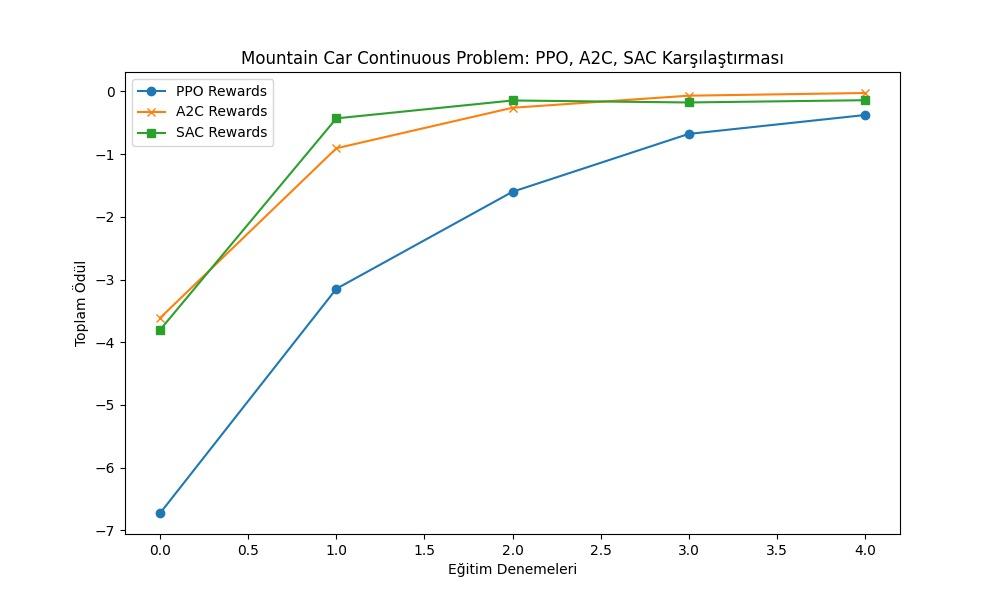
#### A2C (Advantage Actor-Critic)

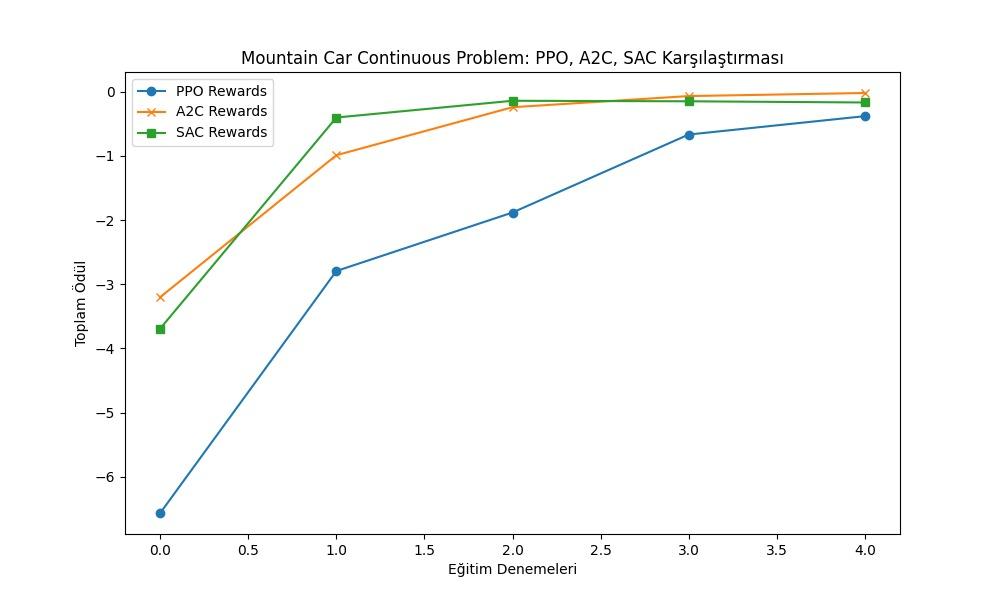
1. Aksiyon ve Durum Değerlendirmesi: Aksiyon ve duruma göre beklenen avantajı (elde edilecek ödül) hesaplar.
2. Aksiyon Seçimi: Mevcut duruma göre en uygun aksiyon tahmin edilir.
3. Politika Güncellemesi: Aksiyonların avantajına göre politika güncellemeleri yapılır.
4. Ödül Takibi: Eğitim süresince, beklenen ödüllere göre model güncellenir.

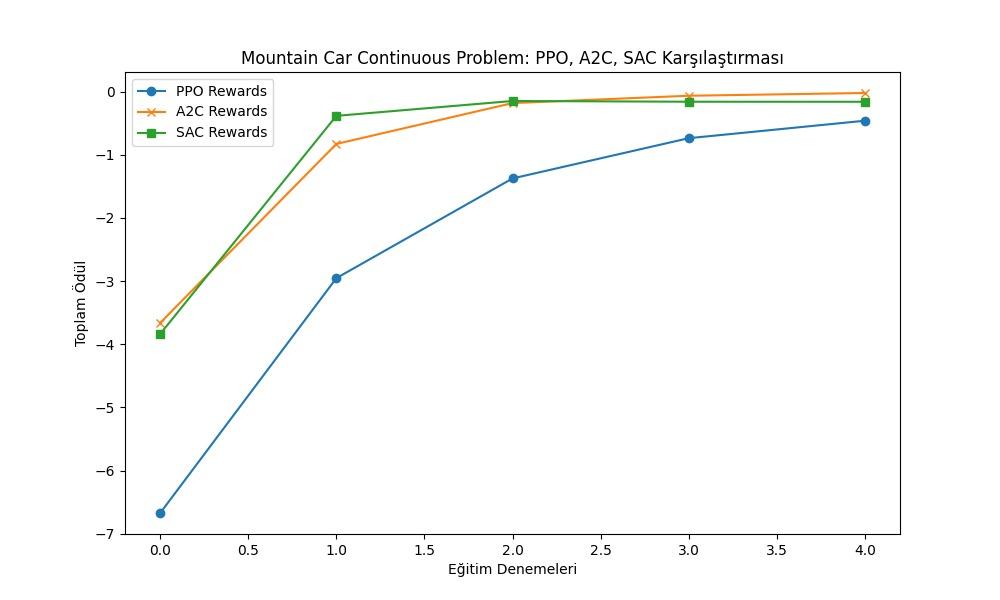
#### SAC (Soft Actor-Critic)

1. Eğitim Başlat**:** learn fonksiyonu çağrılır ve model, belirli bir adım boyunca eğitim yapar.
2. Aksiyon ve Değer Fonksiyonu Öğrenimi: Hem politika (aktör) hem de değer fonksiyonu (kritik) öğrenilir.
3. Soft Maksimum Entropi: SAC, aksiyon seçiminde entropiyi maksimize etmeye çalışır; bu, politikayı daha keşif odaklı yapar.
4. Politika Güncellemesi: Güncellemeler, hem ödüller hem de entropiye göre yapılır.

**Uygulama**



****



**Car Racing**

### Çözümün Genel Mantığı

* Amaç: Aracı, pistte mümkün olan en uzun süre boyunca, en iyi hız ve doğrultuda tutarak maksimum ödül toplamaktır.
* Algoritmalar:
  + PPO (Proximal Policy Optimization): Adım adım değişiklikler yaparak istikrarlı bir şekilde öğrenmeyi sağlar ve çevreye duyarlı hareketler üretir.
  + A2C (Advantage Actor-Critic): Kritik unsurlar üzerinde çalışarak her hareketin avantajını hesaplar, ödülleri maksimize eder.
  + SAC (Soft Actor-Critic): Maksimum entropi yöntemi ile ödülleri artırarak kararlı hareketler öğrenir.

**Akış Diyagramı**

1. **Başla** - Ortam ve model tanımları yapılacak.
2. **Ortamı ve Modelleri Başlat** - CarRacing-v0 ortamını oluştur ve üç algoritma (PPO, A2C, SAC) için modelleri tanımla.
3. **Eğitim Döngüsüne Gir** - Her bir algoritma için eğitim döngüsüne başla:
4. **Ortamı Sıfırla** - Eğitim bölümünü başlat.
5. **Hareket Tahmini** - Algoritma kullanılarak hareket tahmin et.
6. **Hareket Uygula ve Ödül Al** - Ortama hareketi uygula, yeni durum ve ödülü al.
7. **Ödül Kaydet** - Toplam ödüle ekle.
8. **Bölüm Bitti mi?** - Evetse, eğitim bölümünü bitir. Hayırsa, tahmin ve hareket döngüsüne devam et.
9. **Grafikleri Çiz ve Karşılaştır** - Her algoritmanın ödüllerini karşılaştıran bir grafik oluştur.
10. **Bitir** - Eğitim döngüsünü bitir ve sonuçları incele.

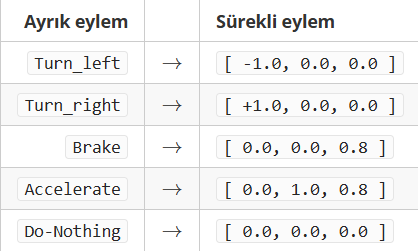
**Çalışma Mantığı**

**Ortam Tanımı:** CarRacing-v0 ortamı oluşturulur.

**Model Tanımı:** PPO, A2C, ve SAC modelleri CNN tabanlı politika ağı ile tanımlanır. (CarRacing görüntü tabanlı olduğu için CNN kullanılır)

**Eğitim Döngüsü:**

1. Her algoritma için belirlenen bölüm sayısı kadar eğitim yapılır.
2. Eğitim esnasında, model her adımda çevreden durumu alarak bir hareket tahmini yapar.
3. Çevreye yapılan hareketin sonucunda yeni durum, ödül ve bölüm sonu bilgileri elde edilir.
4. Ödüller bölüm sonunda toplanır ve kaydedilir.
5. **Grafik Oluşturma:** Her bir algoritmanın toplam ödüllerini gösteren bir grafik çizilir ve karşılaştırma yapılır.



ppo