PEKİŞTİRMELİ ÖĞRENME ALGORİTMASI İLE TAKIM OYUNU OYNAYABİLEN YAPAY ZEKA GELİŞTİRME

BİTİRME PROJESİ 2

Furkan Kaya

191216002

DANIŞMAN

Öğr. Gör. Ezgi Özer

20 Kasım 2020

ÖZET

Bu projede pekiştirmeli öğrenme algoritması kullanılarak takım oyunlarından plaj voleybolu oynayabilen yapay zeka geliştirilmektedir. Bu doğrultuda öncelikli olarak ML-Agents kütüphanesi kurulmuştur. Ardından kullanılacak algoritmanın testini yapabilmek adına daha basit bir problem belirlenip üzerinde ön çalışma yapılmıştır. Modelleme programı öğrenilmiş ve problem için gerekli modeller tasarlanmıştır.

Kısacası bu raporda kurulması gereken programlardan, bilinmesi gereken algoritmalardan, yapılmış ve yapılacak olan adımlardan bahsedilmiştir.

Algoritmalar:

Unity ML-Agents Toolkitte yapay zekalar pekiştirmeli öğrenme sırasında Proximal Policy Optimization (PPO) tekniğini kullanılarak eğitilebilmektedir.

Proximal Policy Optimization Algorithms (PPO):

OpenAI tarafından 2017 yılında yayınlanmış bir pekiştirmeli öğrenme algoritmasıdır. Atari oyunlarından robotik simülasyonlara ve hatta geniş çaplı karmaşık AAA (bkz. Dota 2) oyunlarına kadar birçok alanda yapay zeka geliştirirken kullanılabilmektedir. Matematiksel formülü aşağıda görüldüğü gibidir.

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

Proximal Policy Optimization Formülü[1]

Algoritmanın sözde kodu ise görseldeki[2] gibidir.

Algorithm 1 PPO-Clip

1: Input: initial policy parameters θ_0, initial value function parameters ϕ_0
2: **for** $k = 0, 1, 2, \ldots$ **do**
3: Collect set of trajectories $\mathcal{D}_k = \{\tau_i\}$ by running policy $\pi_k = \pi(\theta_k)$ in the environment.
4: Compute rewards-to-go $\hat{R}_t$.
5: Compute advantage estimates, $\hat{A}_t$ (using any method of advantage estimation) based on the current value function V_{ϕ_k}.
6: Update the policy by maximizing the PPO-Clip objective:

$$\theta_{k+1} = \arg\max_{\theta} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \min\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)} A^{\pi_{\theta_k}}(s_t, a_t),\ \ g(\epsilon, A^{\pi_{\theta_k}}(s_t, a_t)) \right),$$

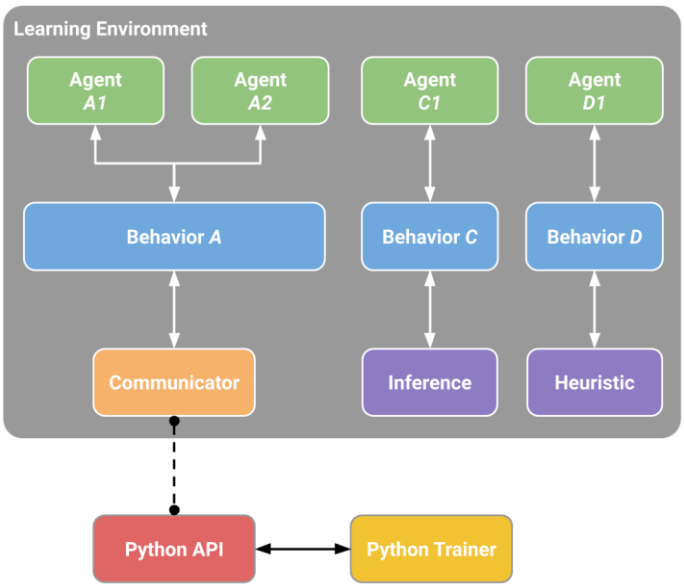
typically via stochastic gradient ascent with Adam.
7: Fit value function by regression on mean-squared error:

$$\phi_{k+1} = \arg\min_{\phi} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \left(V_\phi(s_t) - \hat{R}_t \right)^2,$$

typically via some gradient descent algorithm.
8: **end for**

[1] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, Oleg Klimov. Proximal Policy Optimization Algorithms (2017). arXiv:1707.06347v2

[2] OpenAI. Pseudo code of PPO Algorithm. https://spinningup.openai.com/en/latest/algorithms/ppo.html



Ml-Agents dökümantasyonundan alınan görselde de görüldüğü gibi; Ml-Agent Toolkitte ise bu algoritma TensorFlow ile implemente edilmekte ve eğitim sırasında ayrı bir python süreciyle çalışmaktadır. Unity ile ortak bir port üzerinden haberleşerek yapay zekalar eğitilmektedir.

Kurulumlar:

Unity ML-Agents Toolkit'in kurulum sayfası[3] takip edilerek eğitim için gerekli programların kurulumu yapılmıştır.

Unity Kurulumu

Unity ML-Agents Toolkit'in kurulum sayfasında Unity Engine sürüm 2018.4 ya da daha yeni bir versiyonun kurulması önerilmiştir. Unity'nin yeni versiyonlarında olan özelliklerinden yararlanabilmek için bu projenin geliştirilmesinde 2020.02 seçilip kurulmuştur.

[3] ML-Agents Toolkit Kurulum Sayfası: https://github.com/Unity-Technologies/ml-agents/blob/master/docs/Installation.md

Python Kurulumu

Anaconda programı ile kurulumu yapılmıştır. Terminal ekranı açılıp sırasıyla aşağıdaki adımlar izlenmiştir.

- "conda create -n graduation-project python=3.8.5" komutu yazılarak 3.8.5 sürümüyle graduation-project isimli bir python ortamı kurulmuştur.
- "conda activate graduation-project" komutu ile kurulan ortam aktive edilmiştir.
- "pip install ml-agents" komutu ile ml-agents paketi kurulmuştur.
- "pip3 install torch==1.7.0 -f https://download.pytorch.org/whl/torch_stable.html" komutu ile ml-agents paketine gerekli olan paketin kurulumu yapılmıştır.

Blender Kurulumu

Blender açık kaynaklı ücretsiz herkesin kullanımına açık bir 3 boyutlu modelleme programıdır. Bu projede bulunan modellemeler blender kullanılarak yapılmıştır. Resmi sitesi üzerinden versiyon 2.91 indirilip kurulumu yapılmıştır.

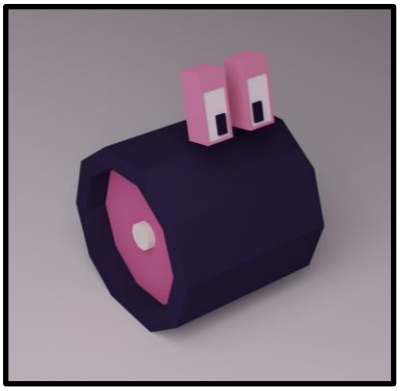
Projesinin Kurulumu

Proje dosyalarının ve raporlarının saklanacağı bir GitHub reposu[4] oluşturulmuş ardından Unity ML-Agents Toolkitin github sayfasından örnek projeleri de içeren proje dosyaları indirilip oluşturulan bu repoya eklenmiştir.

[4] Oluşturulan GitHub Reposu: https://github.com/Wijt/graduation-project

Ön çalışma:

Asıl proje olan voleybol oynayabilen yapay zekaları tasarlamadan önce daha basit bir proje geliştirerek programların ve algoritmaların testinin yapılması planlanmıştır.



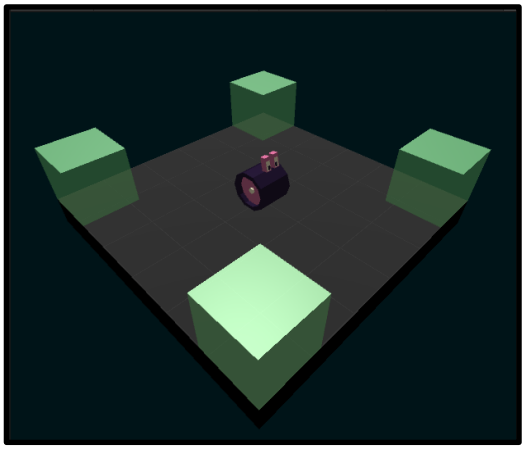
Karakter Tasarımı

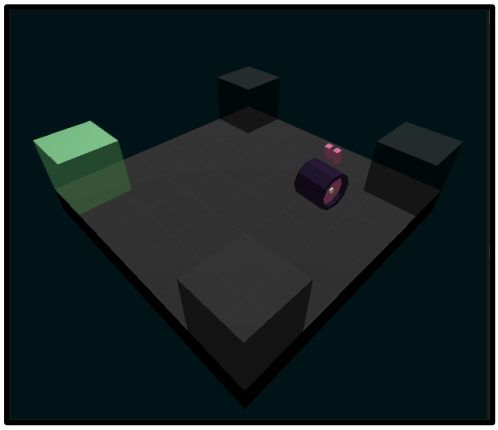
Açık kaynaklı ve ücretsiz bir 3 boyutlu modelleme programı olan Blender kullanılarak soldaki görselde görülen karakterin modellemesi yapılmıştır. FBX formatında çıktı alınıp Unity'e aktarılmıştır.

Eğitim Sahasının Tasarımı

Eğitimin hızlı tamamlanabilmesi için olabildiğince basit bir ortam kurulumu yapılmıştır. Eğitim sahasında şu objeler bulunmaktadır:

- 4 olası hedef
- Karakter
- Zemin





Amaç

Bu projede karakterin, rastgele konumda başlayarak 4 hedef içinden rastgele seçilen hedefe, olabilecek en kısa sürede ulaşmasını öğretmek amaçlanmıştır.

Kullanılan Algoritma:

Pekiştirmeli öğrenme algoritması, PPO

Ödül/Ceza Fonksiyonu:

- Her adımda 0.002 ceza puanı
    - Olabilecek en kısa sürede görevi yerine getirmesini teşvik etmek için bu puan verilmelidir.
- Karakter platformdan aşağı düşerse 10 ceza puanı
- Karakter belirlenen hedefe ulaşması durumunda 15 ödül puanı

Girdiler:

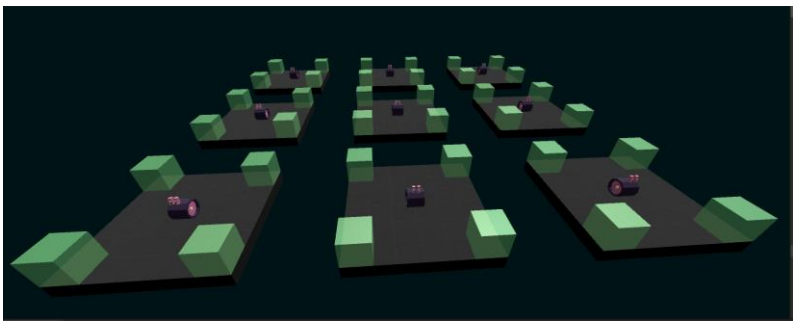
- X, Y ve Z olarak kendi pozisyonu
- X, Y ve Z olarak hedefin pozisyonu
- Kendisinin hedefe olan açısı

Çıktılar:

Ağ 2 adet sonuç üretebilmektedir.

- -1 ile +1 arasında sağa yada sola dönüş hızı
- -0.2 ile +1 arasında ileri ya da geri hareket hızı
    - İleri yönlü hareketi teşvik etmek için geri yönlü hareket hızı düşürülmüştür.

Eğitim Adımları:



Yapay zekaya aynı anda birden çok deneyim yaşatabilmek ve eğitim hızını arttırmak için eğitim ortamı kopyalanarak 3 boyutlu uzayda yan yana dizilmiştir.

Daha sonra "project/config/ppo" yolunda eğitim için ayar dosyası oluşturulmuştur.
Solda bulunan parametreler kullanılmıştır:

```
behaviors:
  CT:
    trainer_type: ppo
    hyperparameters:
      batch_size: 10
      buffer_size: 100
      learning_rate: 0.0003
      beta: 0.0005
      epsilon: 0.2
      lambd: 0.99
      num_epoch: 3
      learning_rate_schedule:
linear
    network_settings:
      normalize: true
      hidden_units: 128
      num_layers: 2
      vis_encode_type: simple
      memory: null
    reward_signals:
      extrinsic:
        gamma: 0.99
        strength: 1.0
    init_path: null
    keep_checkpoints: 100
    checkpoint_interval: 500000
    max_steps: 5000000
    time_horizon: 64
    summary_freq: 10000
    threaded: true
    self_play: null
    behavioral_cloning: null
    framework: pytorch
```

Eğitimi başlatmak için projenin bulunduğu konumda terminal açılmış ve sırasıyla aşağıdaki komutlar girilmiştir.

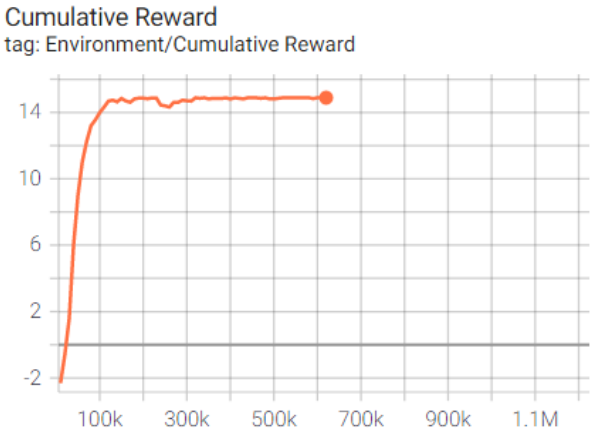
"conda activate graduation-project" yazarak python ortamı aktive edilmiştir.

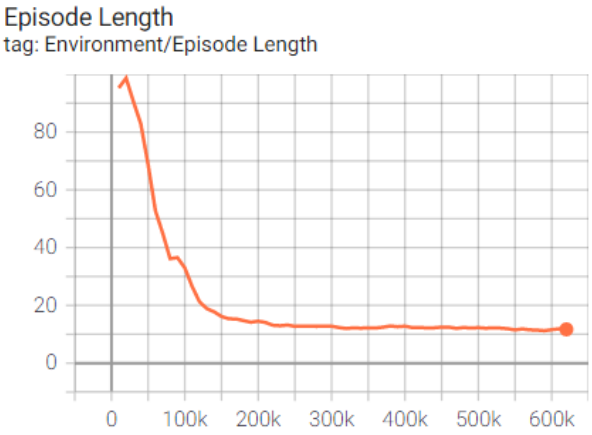
"mlagents-learn config/ppo/CT.yaml --run-id=CT-First" yazılmıştır.

Unity içinden sahne başlatılarak eğitim başlatılmıştır.

Sonuçlar:

Aşağıdaki grafiklerde de görüldüğü üzere yaklaşık 120,000. adımdan itibaren aldığı ödül miktarında artış yaşanmasa da görevi yerine getirme süresi azaldığı için eğitim 620,000. adıma kadar devam etmiştir. Toplam eğitim süresi 1 saat 20 dakika sürmüş ve istenilen başarıya ulaşılmıştır.

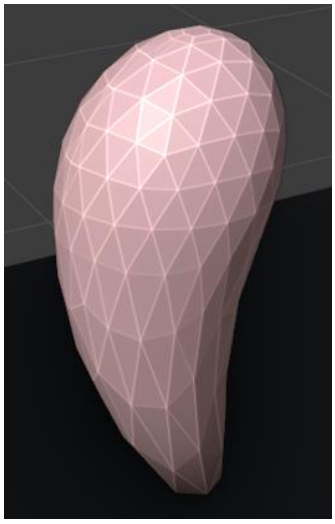




Plaj Voleybolu

Bu kısma kadar anlatılmış konular projenin ilerleyebilmesi için yapılması/öğrenilmesi gereken konulardır. Çalışmanın devamında ise projenin detaylarından bahsedilmiştir.

Tasarlanmış modeller:



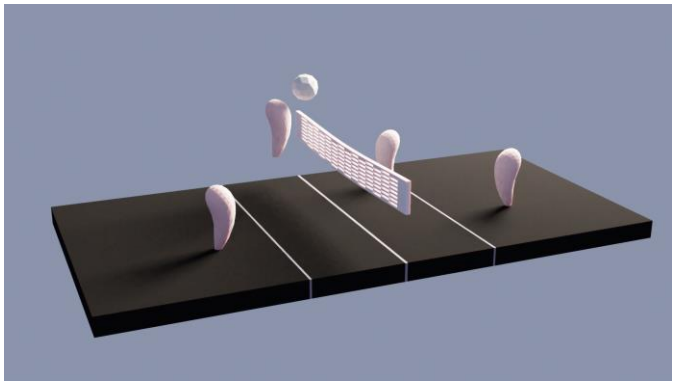
Ön çalışmada test edilip ve öğrenilen blender programı kullanılmıştır.

Oyuncu:

Karakterin top kontrolünü maksimize edebilmek için topun farklı açılardan gelme ihtimali düşünülerek karşılama açısını rahatlıkla tutturabilmesi amacıyla düz yüzeyleri artırılmış ve her açıyı karşılayabilmesi için de yuvarlağımsı bir şekilde modellenmiştir.

Saha:

Gerçek bir plaj voleybolu sahasının ölçüleri baz alınarak modellenmiştir.



Ödül/Ceza Fonksiyonu:

Top rastgele bir takımın tarafında başlayacaktır ve oyunun başlangıcından itibaren geçerli olacak kurallar şunlardır:

- Top rakip sahasına düşerse 10 ödül puanı
- Top kendi sahasına düşerse 1 ceza puanı
- Atılan top fileye değerse 1 ceza puanı
- Alınan her karar için 0.001 ceza puanı

Girdiler:

Her karakter aşağıdaki özellikleri görebileceklerdir:

- Kendi pozisyon ve rotasyon değerleri
  - X, Y ve Z bilgileri
- Takım arkadaşının pozisyon ve rotasyon değerleri
  - X, Y ve Z bilgileri
- Rakip takımdakilerin pozisyon ve rotasyon değerleri
  - X, Y ve Z bilgileri
- Topun X, Y ve Z bilgileri
- Duvarların ve filenin o anki pozisyonuna olan uzaklıkları

Çıktılar:

- -1 ile +1 arasında sağa yada sola dönüş hızı
- -1 ile +1 arasında ileri ya da geri hareket hızı
- -1 ile +1 arasında sağa ya da sola hareket hızı
- 0 ile 1 arasında zıplama gücü

Eğitim adımları ve sonuçlar gelecek raporda raporlanacaktır.