PEKİŞTİRMELİ ÖĞRENME ALGORİTMASI İLE TAKIM OYUNU OYNAYABİLEN YAPAY ZEKA GELİŞTİRME

BİTİRME PROJESİ 2

Furkan Kaya

191216002

DANIŞMAN

Öğr. Gör. Ezgi Özer

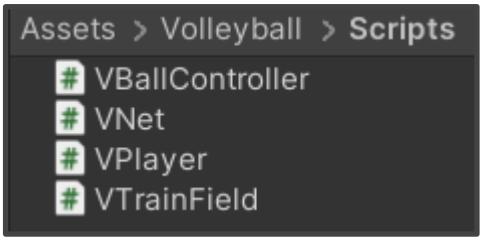
20 Kasım 2020

ÖZET

Daha önceki raporda bahsedilen ortam kurulumu bitirilmiş olup gerekli ödül/ceza fonksiyonlarının tanımlanması yapılmıştır. Ardından farklı parametreler ile eğitimler çalıştırılmış ve 1-1.5 haftalık eğitim sonucunda yaklaşık %80 başarı oranı sağlanmıştır. Oyuncular topu iyi bir şekilde kontrol etmeseler de topa vurmaları gerektiğini başarıyla öğrenmiş durumdalar.

Projenin son haline ve dosyalarına https://github.com/Wijt/graduation-project/ adresinden ulaşabilirsiniz.

Kodlar



Eğitim sahasını oluşturabilmek için 4 sınıf kodlanıp oyun objelerine atanmıştır.

Ball Controller:

```
void OnCollisionEnter(Collision col)
{
    if (col.gameObject.layer == LayerMask.NameToLayer("player"))...

    if (col.gameObject.tag == "AreaA")...

    if (col.gameObject.tag == "AreaB")...

    if (col.gameObject.name == "BetweenWall")...
    if (col.gameObject.tag == "wall")...
    if (col.gameObject.tag == "AreaA" || col.gameObject.tag == "AreaB")...
}
```

Ödül fonksiyonunun büyük bir kısmının olduğu sınıftır. Topun hareketlerini, çarptığı oyuncuları ve atıldığı yönü kontrol ederek oyuncuya ya da takıma ödül/cezalar verilmiştir.

Net:

Bu sınıfın MonoBehaviour adlı Unity oyun objesi sınıfından türetilmiş bir sınıftır. Herhangi bir objenin çarpması durumunda çalışan

```
private void OnCollisionEnter(Collision collision)
{
    if (collision.gameObject.layer == LayerMask.NameToLayer("player"))
    {
        collision.gameObject.GetComponent<VPlayer>().AddReward(-0.1f);
    }
}
```

OnCollisionEnter fonksiyonunun içine dokunan oyuncuya 0.1 ceza puanı verecek kodlar yazılmıştır.

```
36      int invert;
37
38      public bool debug;
39
40      public float jumpCoolDown = 1;
41      float jumpTimer;
42      public bool jumpRight { get => jumpTimer <= 0 && CheckGroundStatus(); }
43      public float hitForce = 0;
44
45      public override void Initialize()...
58
59      public void MoveAgent(ActionSegment<int> act)...
102
103     bool CheckGroundStatus()...
111
112     public override void CollectObservations(VectorSensor sensor)...
171
172     public override void OnActionReceived(ActionBuffers actionBuffers)...
199
200     public override void Heuristic(in ActionBuffers actionsOut)...
243
244     public override void OnEpisodeBegin()...
253
254     private void FixedUpdate()...
270
```

VPlayer:

ML-Agents kütüphanesinin içinde bulunan Agent sınıfından türetilmiş ve yapay zekanın giriş çıkışlarından sorumlu olan sınıftır.

Initialize(): Eğitim ilk başladığında çalışıp daha sonra çalışmayacak olan script için gerekli tanımlamaların yapıldığı fonksiyon.

```
public override void Initialize()
{
    existinal = 1f / MaxStep;
    m_BehaviorParameters = gameObject.GetComponent<BehaviorParameters>();

    team = (Team)m_BehaviorParameters.TeamId;

    agentRb = GetComponent<Rigidbody>();
    agentRb.maxAngularVelocity = 500;
    agentRb.centerOfMass = new Vector3(0, -0.3f, 0.039f);

    ballRb = ballController.GetComponent<Rigidbody>();
}
```

MoveAgent(): Verilen axis değerlerine göre hareket sağlayan fonksiyon.

```
public void MoveAgent(ActionSegment<int> act)
{
    var dirToGo = Vector3.zero;
    var rotateDir = Vector3.zero;

    var forwardAxis = act[0];
    var rightAxis = act[1];
    var rotateAxis = act[2];

    switch (forwardAxis) {
        case 1: dirToGo = transform.forward * ForwardSpeed; break;
        case 2: dirToGo = transform.forward * -ForwardSpeed; break;
    }

    switch (rightAxis)...

    switch (rotateAxis)...

    transform.Rotate(rotateDir * Time.deltaTime * 100f);
    agentRb.AddForce(dirToGo * Time.deltaTime, ForceMode.VelocityChange);
}
```

CheckGroundStatus(): Oyuncunun altından gönderilen ışın zemine değerse true değmez ise false değer döndüren, oyuncunun yerde mi havada mı olduğunu kontrol eden fonksiyon.

```
bool CheckGroundStatus() {
    RaycastHit hit;
    Ray landingRay = new Ray(transform.position, Vector3.down);
    return Physics.Raycast(landingRay, out hit, .45f, LayerMask.GetMask("trainfield"));
}
```

OnActionReceived(): Agent sınıfından override edilmiştir. Fonksiyonun değişkeni olan actionBuffers içine yapay zekadan aldığımız çıktılar atanmaktadır.

```
public override void OnActionReceived(ActionBuffers actionBuffers)
{
    MoveAgent(actionBuffers.DiscreteActions);
    timePenalty -= existinal;
    var continuousActions = actionBuffers.ContinuousActions;
    hitForce = Mathf.Clamp(continuousActions[1], 0f, 1f);

    if ((continuousActions[0] > 0) && jumpRight) {
        agentRb.AddForce(Vector3.up * continuousActions[0] * JumpForce, ForceMode.Impulse);
        jumpTimer = jumpCoolDown;
    }
}
```

Heuristic(): Yapay zekanın ürettiği sonuçları kendimiz vererek yazdığımız kodları test edip yapay zekayı kontrol edebileceğimiz bir fonksiyon.

```
public override void Heuristic(in ActionBuffers actionsOut)
{
    var discreteActionsOut = actionsOut.DiscreteActions;
    var continuousActionsOut = actionsOut.ContinuousActions;

    continuousActionsOut.Clear();
    discreteActionsOut.Clear();

    //forward
    if (Input.GetKey(KeyCode.W))
    {
        discreteActionsOut[0] = 1;
    }
    if (Input.GetKey(KeyCode.S))...
    //rotate
    if (Input.GetKey(KeyCode.A))...
    if (Input.GetKey(KeyCode.D))...
    //right
    if (Input.GetKey(KeyCode.E))...
    if (Input.GetKey(KeyCode.Q))...

    //jump
    if (Input.GetKey(KeyCode.Space))...
    hitForce = 1;
}
```

OnEpsidoBegin(): Her bölüm başında çalışacak fonksiyonlar oyuncunun değerlerini resetlemek için kullanıyoruz.

```
public override void OnEpisodeBegin() {
    invert = team == Team.A ? 1 : -1;
    timePenalty = 0;
    jumpTimer = 0;
    agentRb.velocity = Vector3.zero;
    agentRb.angularVelocity = Vector3.zero;
    area.MatchReset();
}
```

CollectObservation(): Bu fonksiyon yapay zekanın giriş işlemlerinden sorumlu olan fonksiyondur. Agent sınıfındaki CollectObservation fonksiyonu override edilerek bir önceki raporda belirtilen giriş değerlerimiz yapay zekaya iletilmektedir.

```
public override void CollectObservations(VectorSensor sensor)
{   //Topun hız değerleri
    sensor.AddObservation(ballRb.velocity.x * invert);
    sensor.AddObservation(ballRb.velocity.y);
    sensor.AddObservation(ballRb.velocity.z * invert);
    //Topun Pozisyon değerleri
    sensor.AddObservation(ballRb.transform.localPosition.x * invert);
    sensor.AddObservation(ballRb.transform.localPosition.y);
    sensor.AddObservation(ballRb.transform.localPosition.z * invert);
    //Topa olan uzaklığı
    sensor.AddObservation(Vector3.Distance(transform.localPosition, ballRb.transform.localPosition));

    sensor.AddObservation(jumpRight); //Zıplama hakkı olup olmadığı
    sensor.AddObservation(jumpTimer); //Zıplaması için kalan süre
    //Sahada bulunan her bir oyuncunun
    foreach (var playerStates in area.transform.GetComponentsInChildren<VPlayer>())
    {
        Transform playerPos = playerStates.agentRb.transform;
        Rigidbody playerRb = playerStates.agentRb;

        sensor.AddObservation(playerPos.transform.localPosition.x * invert);
        sensor.AddObservation(playerPos.transform.localPosition.y); //Pozisyon değerleri
        sensor.AddObservation(playerPos.transform.localPosition.z * invert);

        sensor.AddObservation(playerRb.velocity.x * invert);
        sensor.AddObservation(playerRb.velocity.y); //Hız değerleri
        sensor.AddObservation(playerRb.velocity.z * invert);
    }

    sensor.AddObservation(transform.rotation.y); //Oyuncunun bakış yönü
}
```

Trainfield:

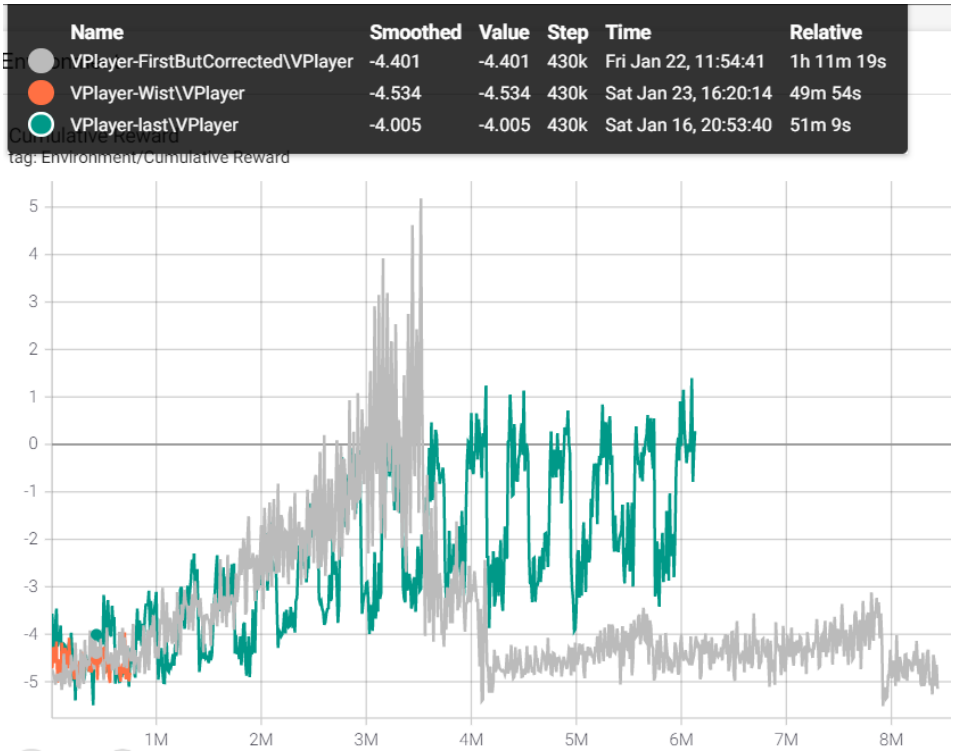
Oyuncuların bilgisini tutan, bölümü bitirip yeniden başlatmakla sorumlu olan sınıftır. MonoBehaviour sınıfından türetilmiştir.

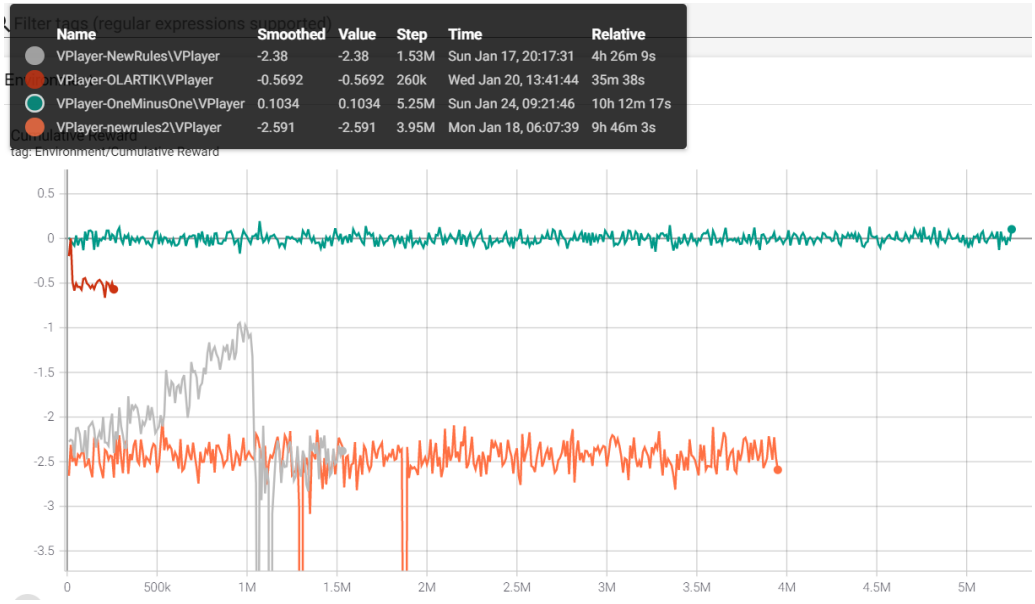
```
public class VTrainField : MonoBehaviour
{
    public List<VPlayerState> playerStates = new List<VPlayerState>();
    public VBallController ballController;

    private void Awake()
    {
        foreach (VPlayer player in GetComponentsInChildren<VPlayer>())
        {
            var playerState = new VPlayerState
            {
                agentRb = player.agentRb,
                startingPos = player.transform.position,
                startingRot = player.transform.rotation,
                agentScript = player,
            };

            playerStates.Add(playerState);
            player.PlayerIndex = playerStates.IndexOf(playerState);
            playerState.playerIndex = player.PlayerIndex;
        }
    }

    void Start() {
        MatchReset();
    }

    public void MatchReset()...
    public void EndEpsido()...
    public void AddRewardToTeam(VPlayer.Team team, float reward)...
}
```

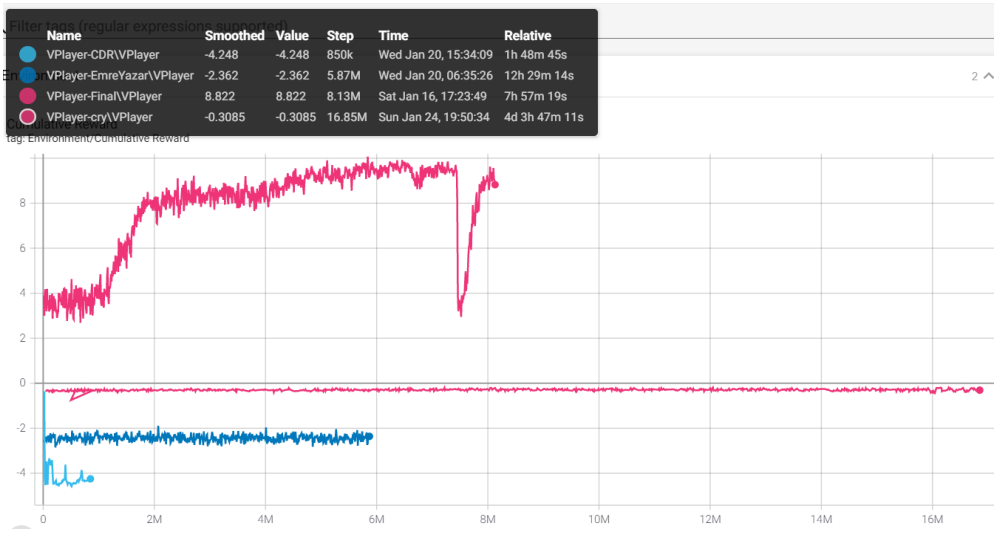
MonoBehaviour sınıfında; Awake fonksiyonu, Start fonksiyonundan önce çalışmaktadır. Start fonksiyonumuz maçı başlangıç pozisyonuna getirmekten sorumludur. Bu sebeple de start fonksiyonunda ihtiyacımız olan tanımlamaları awake fonksiyonunda yazıyoruz.

Eğitim Süreci

Olası en iyi yapay zekayı bulmak üretebilmek için defalarca kez farklı ödül fonksiyonu ve farklı parametreler ile eğitim verilmiştir. Aşağıda kayıt altına alınan eğitim süreçlerinin grafikleri verilmiştir. Eğitim sürecinde 15 farklı çalışma ID'si oluşturulmuş ve toplam 90 saatten fazla eğitim verilmiştir. Aşağıdaki grafiklerde yapılan eğitim denemelerini inceleyebilirsiniz.

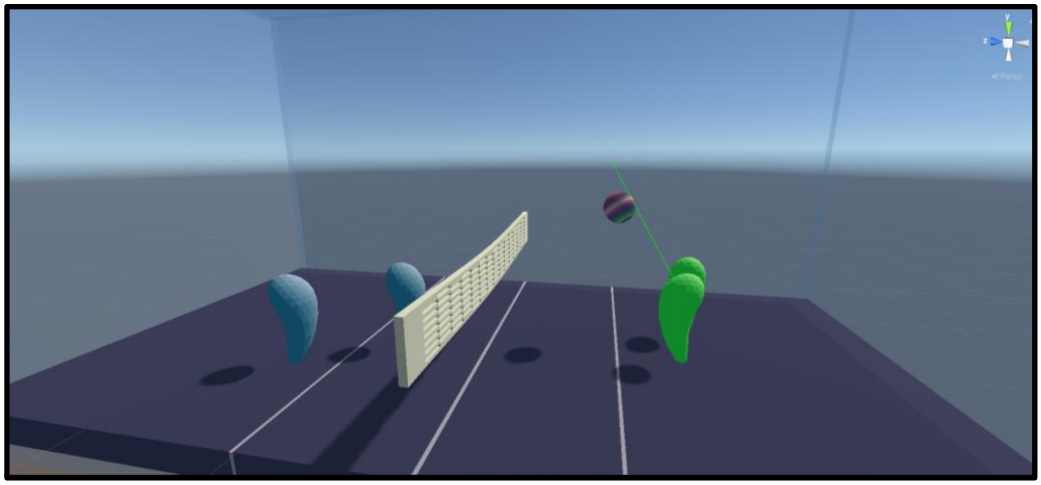


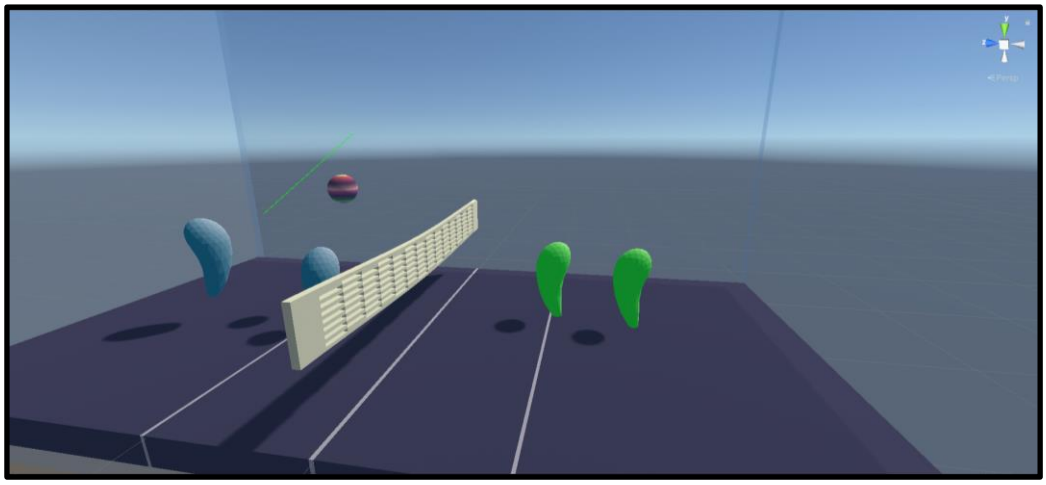




Ne yapılırsa yapılsın yapay zeka local bir minimuma inip oradan çıkmayı başaramadı.

Gelecek çalışmalarda genetik algoritma kullanılarak ve eğitim süresi arttırılarak geliştirme yapılması planlanmaktadır.





Sonuç:

Ulaşılan maksimum başarıda oyuncumuz 10 servis kullanımının 8'inde karşı sahaya topu atıp puan almayı başarmıştır. 10 servisten 1 inde ise rakipten gelen topu karşılamayı başarmıştır. Simülasyonunun karmaşıklığı nedeniyle eğitimin yeterli gelmediği düşünülmektedir. Eğitim süresi artırılırsa daha da güçlü bir yapay zekaya ulaşılabilir.

Başarı sağlanan parametreler:

Config.yaml:

```yaml
behaviors:
  VPlayer:
    trainer_type: ppo
    hyperparameters:
      batch_size: 2048
      buffer_size: 20480
      learning_rate: 0.0003
      beta: 0.005
      epsilon: 0.2
      lambd: 0.95
      num_epoch: 3
      learning_rate_schedule: linear
    network_settings:
      normalize: false
      hidden_units: 256
      num_layers: 8
      vis_encode_type: simple
    reward_signals:
      extrinsic:
        gamma: 0.99
        strength: 1.0
    keep_checkpoints: 50
    max_steps: 50000000
    time_horizon: 1000
    summary_freq: 10000
    threaded: false
    self_play:
      save_steps: 15000
      team_change: 100000
      swap_steps: 2000
      window: 10
      play_against_latest_model_ratio: 0.5
      initial_elo: 1200.0
```

Ödül fonksiyonu:

- Top takımın sahasına düşerse -0.75
- Top rakip takımın sahasına düşerse ve takım top üzerinde vuruş sahibiyse +1
- Topa dokunuş +0.1
- Topa çift dokunuş -0.1
- Topu fırlatmak +0.2
    - Doğru yöne fırlatmak +0.1
    - Yanlış yöne fırlatmak -0.1
- Oyuncunun fileye değmesi -0.1
- Topun rakip sahaya geçmesi +0.4
- Dokunulan topun duvara değmesi -0.02

Eğitim süresi ~36 saat.