# Hacettepe Üniversitesi

# Yazılım Mühendisliği Yüksek Lisans Programı

## 2021 Bahar Dönemi

# Q-Learning ile Yol Bulma

VBM688 Dönem Projesi

Mert Karagöz

N20134084

# İçerik



## Figürler Tablosu



*Not: İçeriklere tıklayarak istenilen sayfaya gidebilirsiniz.*

# 1. Q-Learning Nedir?

Q-Learning, belirli bir durumda bir eylemin değerini öğrenmek için "**Model-free**", "**Reinforcement Learning**" algoritmasıdır. Yani bir ortam modeli gerektirmez ve stokastik*(değişken, rastlantısal)* geçişler ve ödüller ile ilgili sorunları, uyarlamalar gerektirmeden halledebilir. Gerçek hayattan açıklamak gerekirse, hayvanları eğitirken istenilen davranışları yaptıkları takdirde onları mamayla ödüllendirmek, istemediğimiz davranışlarda bulunduklarında onları cezalandırmak veya ilkokul öğretmenimizin ödevlerimizi yaptığımız takdirde bize yıldız verip yapmadığımız durumda puanımızı kırması gibi durumları örnek gösterebiliriz.

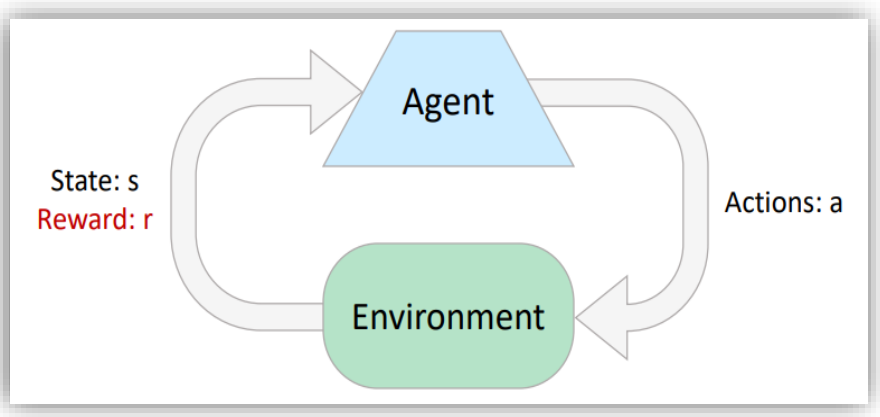


*Figure 1 Q Learning Akış Diagramı*

# 2. Q-Learning Formülasyonu

Q-Learning'deki Q değeri; yapay zeka ajanının bulunduğu state'te ulaştığı ödülü ve bir sonraki state'lerde ulaşacağı ödülleri hesaba katarak hesaplanır.

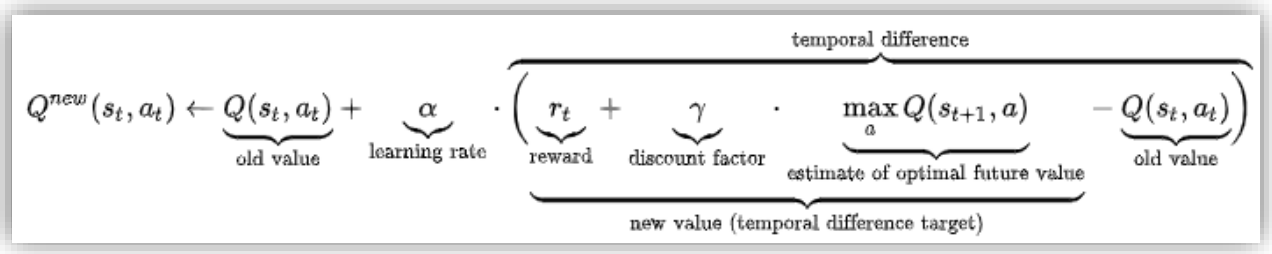


*Figure 2 Q Değerinin Hesaplanması*

## 2.1 Öğrenme Hızı(Learning Rate):

Öğrenme hızı veya adım boyutu, yeni edinilen bilgilerin ne ölçüde eski bilgileri geçersiz kıldığını belirler. 0 faktörü, aracının hiçbir şey öğrenmemesini sağlarken (yalnızca önceki bilgilerden yararlanarak), 1 faktörü ise temsilcinin yalnızca en yeni bilgileri dikkate almasını sağlar (olasılıkları keşfetmek için önceki bilgileri göz ardı ederek). Tamamen deterministik ortamlarda, $a_t$ = 1'de bir öğrenme oranı optimaldir. Problem stokastik olduğunda, algoritma bazı teknik koşullar altında, sıfıra düşmesini gerektiren öğrenme hızında birleşir. Uygulamada, genellikle tüm t için $a_t$ = 0.1 gibi sabit bir öğrenme oranı kullanılır.

## 2.2 İndirim Katsayısı(Discount Factor):

İndirim faktörü γ, gelecekteki ödüllerin önemini belirler. 0 faktörü, ajanı yalnızca mevcut ödülleri, dikkate alarak dar görüşlü yapar, 1'e yaklaşan bir faktör ise uzun vadeli yüksek bir ödül için çabalamasını sağlar. İndirim faktörü 1'i karşılar veya aşarsa, eylem değerleri farklı olabilir. γ = 1 için, bir uç durum olmadan veya aracı hiçbir zaman birine ulaşmazsa, tüm ortam geçmişleri sonsuz uzunlukta olur ve katkı maddeli hizmet programları, indirilmemiş ödüller genellikle sonsuz olur. 1'den biraz daha düşük bir indirim faktörüyle bile, Q işlevi öğrenimi, değer işlevi yapay bir sinir ağı ile yaklaştırıldığında hataların ve kararsızlıkların yayılmasına yol açar. Bu durumda, daha düşük bir indirim faktörüyle başlamak ve bunu nihai değerine doğru artırmak öğrenmeyi hızlandırır.

## 3. Q-Learning Algoritması

Q-Learning Algoritması Q-Table'ı oluşturularak başlar. Başlangıçta yapay zeka ajanımızın çevre hakkında bir bilgisi olmadığı için Q-Table'daki değerlerin hepsi 0'dır. Fakat ajanımız hamle yaptıkça bulunduğu state'lerdeki duruma göre Q-Table'ı güncellemeye devam edecektir. Yapay zeka ajanımız ne kadar çok hamle yapar ise doğru hamleleri yapma olasılığı bi o kadar artacaktır. Bu yüzden yapay zeka ajanının yeteri sayıda hamle yapması, algoritmanın doğru çalışması açısından büyük önem taşımaktadır.

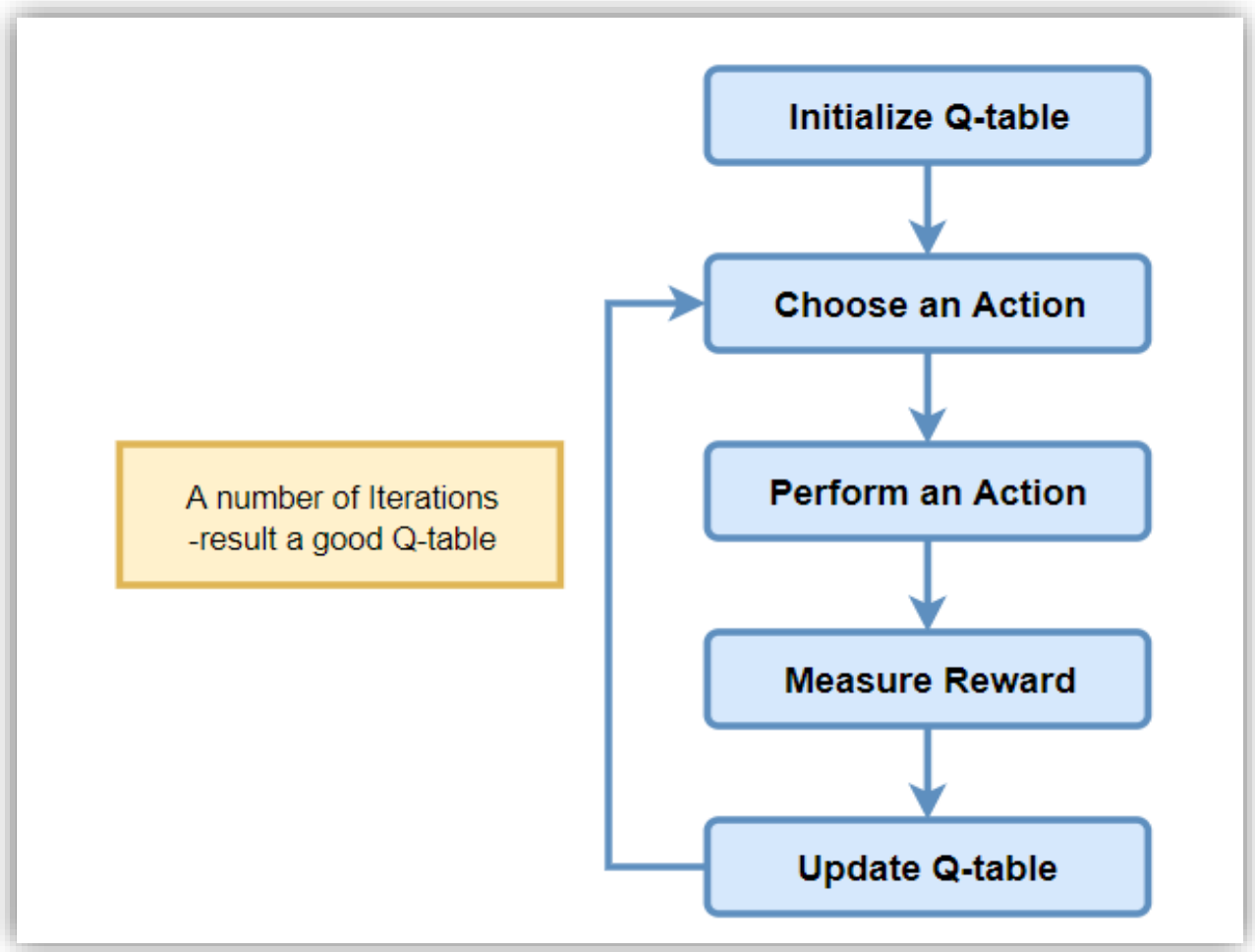


*Figure 3 Q-Learning Algoritması*

# 4. Q-Learning ile Yol Bulma Uygulaması

## 4.1 Giriş

Projem'de yukarıdaki kısımlarda bahsedilen Q-Learning ile alakalı bir uygulama yapmak istedim. Görsel olarak anlatımı daha kolay olacağı için halihazırda birkaç mobil oyun projesi yapmak için kullandığım Unity'i kullanmaya karar verdim. Bu kısımda projemi nasıl yaptığımı detaylı olarak anlatacağım.

## 4.2 Projenin Yapım Aşaması

Bu kısımda Unity'de nasıl proje oluşturulur vb. gibi detaylara girmeden, dersimiz kapsamında yazılım kısmını işin içine katarak projeyi anlatacağım. Proje'nin C# ile yapıldığını da belirtmek isterim.

Not: Uygulama Unity'de yapıldığı için karmaşıklığı önlemek adına oyun motoruna has fonksiyonları ve kodları adım adım anlatırken paylaşmadım(Kaynak kod, en son kısımda bulunabilir).

### 4.2.1 Parametrelerin ve Q Matrix'inin oluşturulması

İlk adım olarak Q Learning algoritmasında kullanacağım parametreleri ve matrix'leri tanımlıyoruz.

```
[SerializeField]private float learningParameter;
[SerializeField]private int currentStateIndex;
[SerializeField]private int action;
[SerializeField]private int nextAction;
public int iterations;

public float[,] Q = new float[5, 5] { { 0, 0, 0, 0, 0 },
                                      { 0, 0, 0, 0, 0 },
                                      { 0, 0, 0, 0, 0 },
                                      { 0, 0, 0, 0, 0 },
                                      { 0, 0, 0, 0, 0 } };


public float[,] R = new float[5, 5] { {-1, 20, -1, -1, -1 },
                                      { 0, -1, 0, 100, -1 },
                                      { -1, 20, -1, -1, 0 },
                                      { -1, 20, -1, 100, 0 },
                                      { -1, -1, 0, 100, -1 } };
```

*Figure 4 Parametre ve Matrixlerin oluşturulması*

### 4.2.2 İterasyon Sayısının Durumunu ve Bir Sonraki Adımı Belirleyen Fonksiyon

İkinci adım olarak iterasyon sayısını kontrol eden ve yapay zeka ajanının çevreyi keşfetmesi adına alacağı aksiyonu ve ondan sonraki aksiyonu tanımlayan iki adet while dönhüsünden sonra yapay zeka ajanının bulunduğu durumu Q Matrix'i üzerine yazan "CalculateQ" fonksiyonu içine giriyoruz.

```
1 başvuru
public void CalculateQMatrixValues(int currentStateIndex)
{
    if (iterations <= 0 )
        return;

    while(R[currentStateIndex, action] < 0)
        action = Random.Range(0, 5);
    while (R[action, nextAction] < 0)
        nextAction = Random.Range(0, 5);

        if (iterations > 0)
        {
            CalculateQ(currentStateIndex, action);
        }
}
```

*Figure 5 Q Matrix'i Hesaplayan Fonksiyonun oluşturulması*

### 4.2.3 Q Matrix'indeki Değerleri Hesaplayan Kısım(CalculateQ fonksiyonu)

Bu kısımda yapay zeka ajanının bulunduğu state'deki yeni Q değerinin hesaplandığı **Figure 2'nin** yazılıma dökülmüş halini görebilirsiniz. Bu kısımda Q değeri güncellendikten sonra iterasyon değeri 1 azaltılacak ve **Figure 5'**deki kısma geri döneceğiz. İterasyon sayısı 0 olana dek bu döngü devam edecektir.

```
Q[currentState, action] = Q[currentState, action] + learningParameter
                                                  * ( R[ currentState, action ]
                                                      + discountFactor
                                                        * Mathf.Max(Q[nextAction, 0],
                                                                    Q[nextAction, 1],
                                                                    Q[nextAction, 2],
                                                                    Q[nextAction, 3],
                                                                    Q[nextAction, 4])
                                                         - Q[currentState, action]);
```

*Figure 6 Q Değerini Hesaplayan Kısım*

## 4.3 Proje’nin Gösterimi

### 4.3.1 Lejant:

Bir önceki kısımda yapay zeka ajanının Q matrix’ini her iterasyonda nasıl güncellediğini anlatmıştım. Bu kısımda yaptığım uygulamayı tanıtacağım.

1: Tüm parametreleri sıfırlayarak uygulamayı resetler.

2: Yapay zeka ajanı Q Matrix’ini hesaplamaya başlar.

3: Q Matrix’i

4: Belirlenen iterasyon sayısı bittikten sonra yapaz zeka ajanını manuel olarak yönlendirmeyi başlatır.

5: Kalan İterasyon sayısı

6: Beş grid’den oluşan yapay zeka ajanının yer değiştirebildiği ortam.

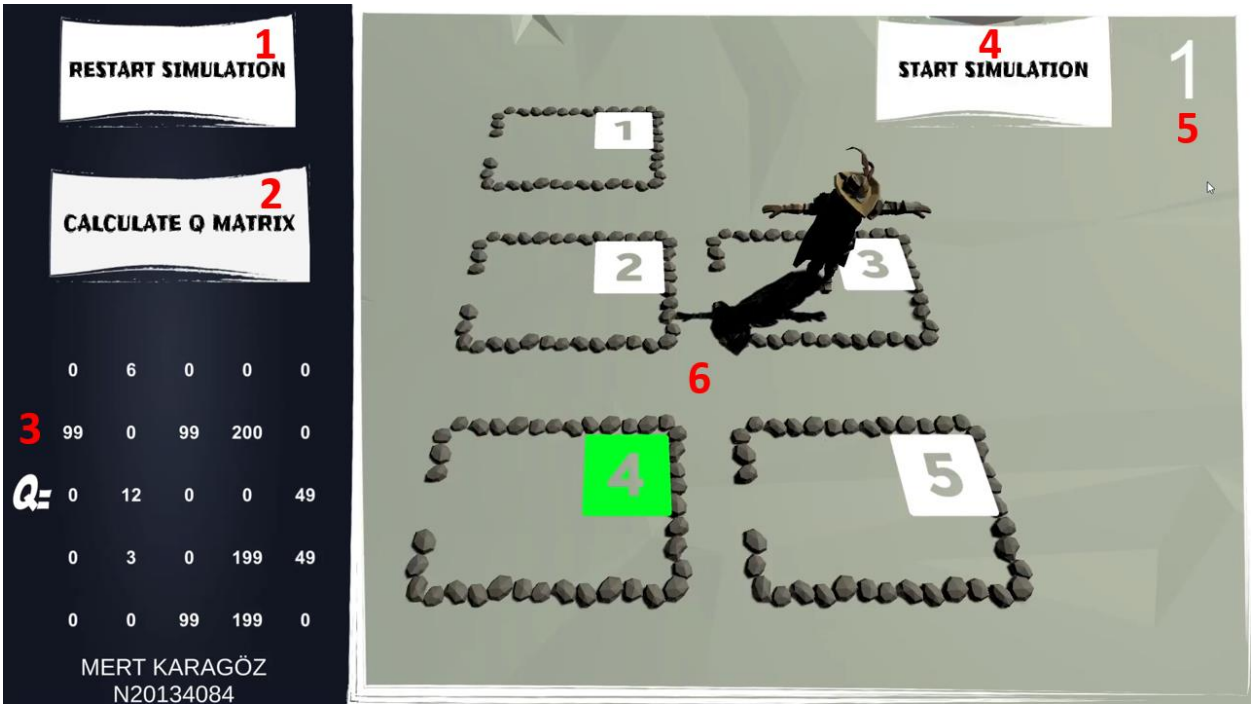


*Figure 7 Uygulamanın Tanıtım Görseli*

### 4.3.2 Uygulamanın Anlatımı

Q Matrix'ini hesaplaya bastıktan sonra ajanımız 100 iterasyon gerçekleştirerek Q matrix'ini güncellemeye başlayacaktır. 100 İterasyon sonunda "4" numara ile belirtilen Start Simulation butonu belirerek Ajanın doğru hareket edip etmediği gözlenebilir. Q matrix'inde satırlar bulunulan grid'i sütunlar ise o gridden gidilebilecek olası grid'leri göstermektedir. Ajanımız puanı en yüksek gride doğru gitmek için programlandı. 4 numaralı grid'e ulaştığı zaman oyunu kazanacağından dolayı Reward Matrix'i (**Figure 4**) oluştururken 4 numaralı grid'e en yüksek puanı verdim. Reward Matrix'inde "-1" ile belirtilen kısımlar bulunulan grid'den ulaşılamayacak olan grid'leri temsil etmektedir. Böylelikle Ajanımız hareket ederken zıplamalar yapmadan sadece komşu grid'lere doğru hareket edebilir.

### 4.3.3 Uygulamanın Youtube Bağlantısı

Yukarıda açıklamaya çalıştığım uygulamanın çalışmasını aşağıdaki bağlantıdan izleyebilirsiniz. Video'yu çektiğim sırada mikrofonum bozuk olduğu için ses kaydedememiştim, affınıza sığınıyorum.

Unity Q - Learning Example - YouTube

[ https://www.youtube.com/watch?v=0unmnL86RXE&ab_channel=MertKarag%C3%B6z ]

## 5. Referanslar


1. Q Learning'e giriş , Turkey, 15.05.2021 , https://medium.com/deep-learning-turkiye/q-learninge-giri%C5%9F-6742b3c5ed2b
2. How to use Q Learning in Video Games Easily, USA, 15.05.2021, https://www.youtube.com/watch?v=A5eihauRQvo&ab_channel=SirajRaval
3. A Beginners Guide to Q-Learning, USA, 16.05.2021, https://towardsdatascience.com/a-beginners-guide-to-q-learning-c3e2a30a653c
4. Q-Learning, USA, 15.05.2021, https://en.wikipedia.org/wiki/Q-learning