Q-LEARNİNG İLE YOL PLANLAMASI

KOCAELİ ÜNİVERSİTESİ BİLGİSAYAR MÜHENDİSLİĞİ
YAZILIM LABORATUVARI PROJESİ
YASİN ÖMER KARA - FEYZA DEMİREL
180201077 – 190201110
y.omerkara1136@gmail.com feyzahae@gmail.com

ÖZET

Bizden Q-Learning algoritması kullanarak bir uygulama yapmamız istendi. Bu uygulamada robotun engellenen yolları seçmeyerek istenilen noktaya ulaşması beklenmektedir.

Ara yüz olarak 50x50 bir matris oluşturarak kullanıcıdan engelleri, başlangıç ve bitiş noktalarını belirlemesi istenecekti. Daha sonra bu matristeki bilgilerden yararlanarak Q-Learning algoritmasına gerekli bilgiler aktarılarak robotun istenilen noktadan hedef noktaya yasaklı bölgelere basmadan gitmesi beklenmektedir.

Robot, herhangi bir beyaz kareden başlayarak sağa, sola, aşağı, yukarı ve çapraz hareket edebilir. Atılan adımlar belirleyici olmalı ve engele çarpışmadıkça başarılı olur. Robot en son duvara geldiğinde robot sadece aşağı hareket ederek istenilen noktaya gelecektir. Sonuç olarak robot başlangıç noktasından istenilen hedefe gelinceye kadar hiçbir engele çarpmadan ve en kısa yolu bularak ödülü alır.

Eğer robot yasaklı bölgelerden birine basarsa oyun biter.

Q-Learning Algoritması nedir?

Q-Learning algoritması, pekiştirmeli öğrenmenin(reinforcement learning) en çok bilinen algoritmalarındandır.

Algoritmadaki temel amaç bir sonraki hareketleri inceleyip yapacağı hareketlere göre kazanacağı ödülü görmek ve bu ödülü **en çoklayıp (maximize)** buna göre hareket etmektir.

Ajanın ödül haritası, gitmesini istediğimiz ya da istemediğimiz yerler daha önceden bizim tarafımızdan belirlenir ve bu değerler **ödül tablosuna (Reward Table)** yazılır.

Ajanın tecrübeleri bu ödül tablosuna göre şekillenecektir. Ajan ödüle giderken her iterasyonda edindiği tecrübeleri gidebildiği yerleri seçerken en çoklamak için kullanır. Bu tecrübeleri ise **Q-Tablosu (Q-Table)** adı verdiğimiz bir tabloda tutar.

1. GİRİŞ

Projeyi C# programlama dili ile yazdık. Öncelikle C# ile ara yüzü nasıl oluşturacağımıza dair araştırmalar yaptık.

Daha sonra Q-Learningi C#'da nasıl kullanabileceğimize dair araştırmalar yaptık. Q-Tablosunu nasıl oluşturacağımızı, ödül tablosunu nasıl oluşturacağımızı araştırdık.

Daha sonra ise kullanıcının oluşturduğumuz ara yüzdeki belirlediği nokta bilgilerini nasıl alacağımıza dair araştırmalar yaptık.

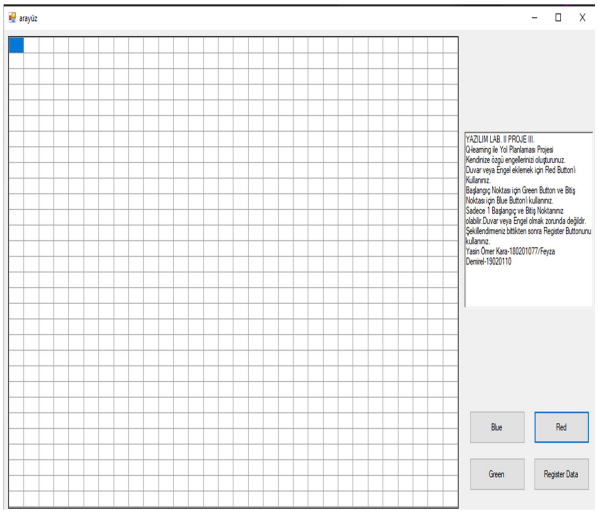
Gerekli araştırmaları ve yöntemlerimizi belirledikten sonra ise kodu yazmaya başladık.

2. YÖNTEM

Projeyi yapmaya öncelikle ara yüzü oluşturarak başladık. Ara yüzü oluştururken Datagrid Table kullandık.

Ara yüzde kullanıcının gerekli renkleri girmesi için 3 renk butonu ekledik. Bu butonlar Blue, Green ve Red isimli butonlardı. (Şekil A.1)

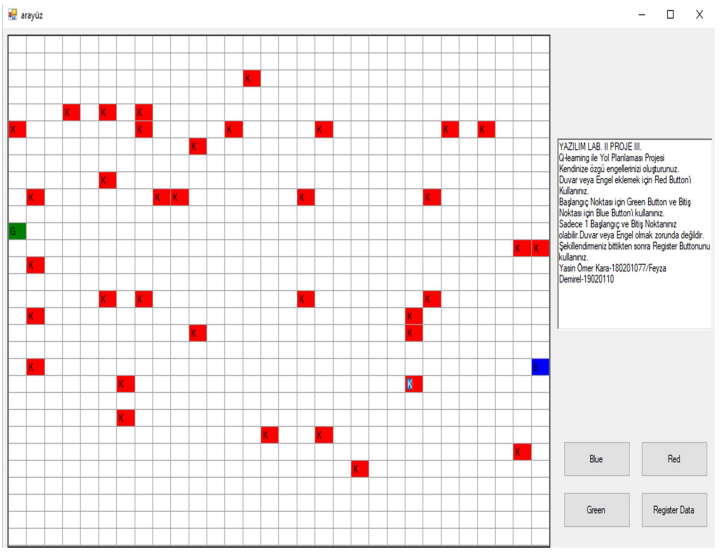
Şekil A.1:



Bu şekilde bulunan Register butonu ise girilen verilerin txtye eklenmesini sağlıyor.

Kullanıcıdan engelleri oluşturması için öncelikle Red butonuna basmasını ve sonra engel eklemek istediği karelere basmasını istedik. Başlangıç için Green ve bitiş noktası için de Blue butonuna basarak eklemek istediği noktaları seçmesini istedik.(Şekil A.2)

Şekil A.2:



Kullanıcı verileri girdikten sonraki görüntü yukarıdaki şekildeki gibi görünüyor.

Bu veriler girildikten sonra Q-Learning algoritmasını projemize ekledik. Q-Learning algoritmasını yazarken öncelikle birden fazla kaynak araştırdık. Ve daha sonra elde ettiğimiz bilgiler sonucunda kodumuza en uygun şekilde algoritmayı oluşturduk.

Bu durumda en zorlandığımız konu matrisleri oluşturmaktı. Çünkü sabit değil kullanıcıya göre sürekli değişen bir matris oluşturmamız gerekiyordu.

Diğer istediğimiz şeylerden biri de algoritmanın olabildiğince hızlı çalışmasıydı çünkü robotun yolu öğrenmesi için sürekli deneme yanılma yolu ile çalışması gerekiyordu.

Gerekli araştırmalar sonucunda algoritmayı koda ekledik.

Aşağıda Q-Learning algoritmamızın bir kısmını kod olarak paylaştık. (Şekil A.3)

Şekil A.3:

```
static double[][] CreateQuality(int ns)
{
    double[][] Q = new double[ns][];
    for (int i = 0; i < ns; ++i)
        Q[i] = new double[ns];
    return Q;
}
static List<int> GetPossNextStates(int s,int[][] FT)
{
    List<int> result = new List<int>();
    for (int j = 0; j < FT.Length; ++j)
        if (FT[s][j] == 1) result.Add(j);
    return result;
}
static int GetRandNextState(int s, int[][] FT)
{
    List<int> possNextStates = GetPossNextStates(s, FT);
    int ct = possNextStates.Count;
    int idx = rnd.Next(0, ct);
    return possNextStates[idx];
}
static void Train(int[][] FT, double[][] R, double[][] Q, int goal, double gamma, double lrnRate, int maxEpochs
{
    for (int epoch = 0; epoch < maxEpochs; ++epoch)
    {
        int currState = rnd.Next(0, R.Length);
        while (true)
        {
            int nextState = GetRandNextState(currState, FT);
            List<int> possNextNextStates = GetPossNextStates(nextState, FT);
            double maxQ = double.MinValue;
            for (int j = 0; j < possNextNextStates.Count; ++j)
            {
                int nns = possNextNextStates[j];  // short alias
                double q = Q[nextState][nns];
                if (q > maxQ) maxQ = q;
```

3. DENEYSEL SONUÇLAR

Öncelikle farklı programlama dilleri kullanarak projeyi oluşturmayı denedik. Ama bizim için projeyi ara yüzü, algoritmayı en kolay şekilde sürdürdüğümüz dil C# oldu.

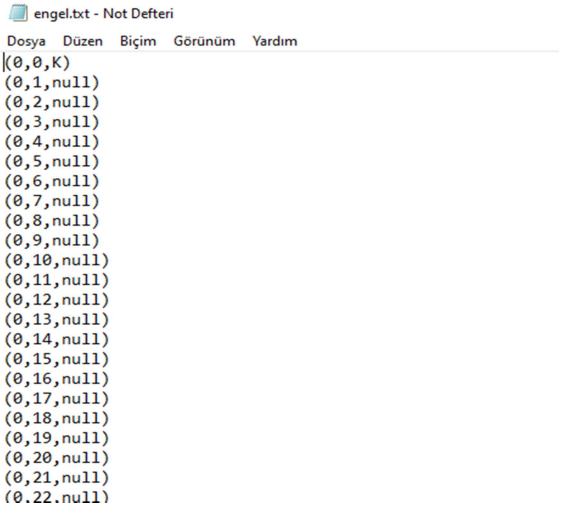
Öncelikle ara yüzü oluştururken farklı şeyler denedik. Ama denediklerimiz arasında kullanıcının işaretlediği renk verilerini en kolay alabildiğimiz yapı Datagrid Table oldu. Bu yüzden ara yüz tercihimizi bu taraftan yana kullandık.

Kullanıcının istediği renkte bloklar eklemesi için renk butonları ekledik. Ve bu renk butonlarından başlangıç ve bitiş butonlarına tıklandığında sadece bir tane bloğu boyamalarına izin verdik.

Çünkü bir tane başlangıç ve bir tane bitiş noktası olmalıydı.

Butonlardan Register butonuna tıklandığında ise matristeki her bir kare için konumlarını ve bir renge boyandıysa boyandığı rengi, boyanmadıysa da null değerini yazdırdık. (Şekil A.4)

Şekil A.4:



Yukarıdaki resimde bu değerleri yazdırdığımız engel.txt dosyası içerisindeki değerlerden bazıları yer alıyor.

Robotun en kısa yolu bulması için istendiği gibi Q-Learning algoritmasını kodumuza ekledik.

Q-Learning algoritmasını koda eklerken öncelikle matrisleri nasıl oluşturacağımızı araştırdık ve araştırmalarımız sonucunda Ödül Tablosunu ve Q Tablosunu oluşturduk.

4. KULLANDIĞIMIZ KÜTÜPHANELER

Projede kullandığımız kütüphaneler;

- System.Windows.Forms
- System.Drawing
- System.Text
- System.IO

5. KAYNAKÇA

[1].Web Site

https://medium.com/deep-learning-turkiye/q-learninge-giriş-6742b3c5ed2b

[2].Web Site

https://code-ai.mk/how-to-implement-q-learning-algorithm-in-c/

[3].Web Site

https://www.srdrylmz.com/tag/c-arayuz-olusturma/

[4].Web Site

https://docs.microsoft.com/tr-tr/dotnet/api/system.windows.forms.datagrid?view=netframework-4.8

[5].Web Site

https://docs.microsoft.com/en-us/archive/msdn-magazine/2018/august/test-run-introduction-to-q-learning-using-csharp

6. AKIŞ DİYAGRAMI