**Q-LEARNİNG ILE LABİRETTE YOL BULMA**

*Bahadır Yıldız 130202037 Fatih Arslancan 120202056*

Bilgisayar Mühendisliği Bölümü

Kocaeli Üniversitesi

[bahad41@gmail.com](mailto:bahad41@gmail.com) , [arslancanfatih@gmail.com](mailto:arslancanfatih@gmail.com)

**Özet**

*Bu projede bizden istenilen bir tür yapay zeka algoritması olan Q-Learning Algoritması ile labirentte yol bulunması istenmiştir. Proje için ayrı bir klasörde labirentler için txt formatında input ve output dosyaları verilmiştir. Bunların yanında labirentin gösterimi için txtlerin içindeki inputa göre pdf verilmiştir. İnputlara göre kazanç matrisi adı verilen R matrisi oluşturulmaktadır. Bu matrisin satır ve sütünları ortamları temsil etmektedir, R[i][j] değeri ise i durumundan j durumuna geçildiğinde elde edilen anlık kazanç değeridir. Kazanç değerleri -1, 0 ve 100 olabilmektedir. Daha sonra detaylı bir şekilde anlatılacaktır. Çıktıyı ise q matrisi göstermektedir. Algoritma her iterasyonda rastgele bir durumdan öğrenmeye başlar. A'ya göre durum değiştirir ve Q matrisini günceller. A'ya göre hedef duruma ulaşıldığında iterasyon sona ermektedir. Böylece Q learning algoritması kullanılarak makine öğrenilmesi amaçlanmıştır.*

**1. Giriş**

Bu projede bizden istenilen bir tür yapay zeka algoritması olan Q-Learning Algoritması ile labirentte yol bulunması istenmiştir. Bu algoritma makine öğrenmesine dayanmaktadır. Makine öğrenimi, günümüzde birçok uygulama alanı olan bir bilim dalıdır. Adından da anlaşılacağı gibi bilgisayara bir şeyleri öğretmeyi ve bu öğrendiklerini insan yararına kullandırtmayı amaçlayan çalışmalar yapmaktadır. Makine öğrenmesi yapay zekanın alt dallarından birisidir. Makinenin kendi kendine öğrenmesi, sahip olduğu sensör yada veritabanı gibi veri kaynaklarından elde edeceği verileri analiz ederek karşılaşabileceği yeni

durumlar karşısında en uygun kararı verebilmesidir. Makinenin öğrenmesinde kullanılan 3 çeşit yöntem vardır. Bunlar eğiticili, eğiticisiz ve takviyeli öğrenme yöntemleridir.

Takviyeli öğrenme, kendi ortamında algılama ve hareket yapan bir özerk etmenin amacını gerçekleştirmek için en uygun hareketleri yapmayı nasıl öğrenebileceği sorusuna cevap verir. Bu çok genel problem, bir hareketli robotun kontrolü öğrenme, fabrikalardaki işlemleri optimize etmeyi öğrenme ve oyun oynamayı öğrenme gibi işleri kapsar. Etmen ortamında bir hareket yaptığı zaman, bir eğitici ortaya çıkan durumun istenilebilirliğini göstermek için bir ödül veya ceza sağlar. Mesela bir oyunu oynama etmeni eğitildiği zaman, eğitici oyun kazanıldığında pozitif bir ödül, kaybedildiğinde negatif bir ödül ve diğer durumlarda sıfır ödül sağlayabilir. Etmenin amacı, en büyük toplam ödülü üreten hareketlerin sırasını öğrenmektir. Q learning algoritması yukarıda da anlatıldığı gibi ödül ve cezaya dayanan bir öğrenme algoritmasıdır.

Q-learning algoritmasını projemizdeki gibi labirentte dolaşan bir robot üzerinde anlatalım. Başlangıçta robot, labirent hakkında sadece yollar arasında geçiş varmı yokmu onu bilmektedir. Ancak çıkışa ulaştıran yolu bilmemektedir. Robot, rastgele yollara girerek girişten çıkışa ulaşmaya çalışacaktır. Bu gezinti esnasında çıkış bulunduğu takdirde çıkıştan bir önceki yola belirli bir kesinti yapılarak (çünkü en yüksek puan çıkışındır.) puan verilir. Bilgilerin tutulduğu Rve Q matrisleri vardır. Başlangıçta sadece çıkışlar ve olası yollar bellidir ancak en kısa yol yada yollar belli değildir.

**1**

En kısa yolun belirlenebilmesi için tüm yolların denenip çıkışa yakınlığına göre derecelendirilmesi gerekir .Bu derecelendirme işlemini q learning algoritması aşağıdaki basit formüle göre yapmaktadır.

Q(state,action)=R(state,action)+ λMAX(ns,na) (1)

Q(state,action):Çıkışa ulaşan derecelendirilmiş geçişleri gösterecek olan matristir. R(state,action): Yolları ve çıkışları gösteren matristir. Lambda: Kesinti oranı denilebilir. Çıkıştan girişe doğru geçişlerin dereceleri düşmesi gerektiğinden 0 < Lambda < 1 olacak şekilde bir Lambda katsayısı seçilir. MAX(nextstate,nextaction):Yukarıda anlatıldığı şekilde yeni satırlara geçerken, geçilecek olan satırda birden fazla yol olması durumunda derecesi en yüksek olan yolun seçilmesi anlamına gelir. Q matrisi incelendiği zaman, artık makinemiz dereceleri takip ederek en kısa yolu bulabilir.

Bunlardan sonra biraz da makine öğrenmesi kavramından bahsetmek gerekir. Makine Öğrenmesi (Machine Learning), matematiksel ve istatistiksel yöntemler kullanarak mevcut verilerden çıkarımlar yapan, bu çıkarımlarla bilinmeyene dair tahminlerde bulunan yöntem paradigmasıdır. Makine öğrenmesine güncel hayatımızdan bazı örnekler: yüz tanıma, belge sınıflandırma, spam tespiti. Bir benzetme yapacak olursak öğrenci konu çalışıp ders kitabındaki örnek çözümlü soruları çözer ve öğrenir. Ardından kitapta bulunmayan ama aynı bilgiye dayanan farklı bir test önüne konulur. Öğrenci cevapları bilmeden testi çözer. Sonra değerlendirmeye alınır, ne kadar başarılı olduğu görülür. Eğer ezberci bir öğrenciyse benzeri kitapta olmayan soruları muhtemelen yanlış yanıtlayacaktır. Eğer işin temelini anlayan bir öğrenciyse farklı tarzda soru gelse bile doğru çözebilecektir. Öğrenci test sonucunu ve nerelerde hata yaptığını inceler. "Hmm şu soruda şöyle bir genelleme yapmışım ama aslında iş bu kadar basit değilmiş", "Hmm şu etkeni hesaba katmayı düşünemedim" gibi dersler çıkarır kendisine. Ardından derslerini almış bir gözle kitabını tekrar çalışır ve tekrar testi çözer. Yeterince iyi sonuç

alana kadar hatalarını keşfedip konunun püf noktalarını öğrenmeye çalışır. Makine öğrenmesi de bunun çok benzeridir. Süreç aynıdır. Terminolojisine bakacak olursak:

Gözlemler (Observations): öğrenmek ya da değerlendirmek için kullanılan her bir veri parçasıdır. Örn: her bir e-posta bir gözlemdir. Özellikler (Features): Bir gözlemi temsil eden (genelde sayısal) verilerdir. Örn: e-posta'nın uzunluğu, tarihi, bazı kelimelerin varlığı. Etiketler (Labels): Gözlemlere atfedilen kategorilerdi. Örn: spam, spam-değil. Eğitim Verisi (Training Data): Algoritmanın öğrenmesi için sunulan gözlemler dizisidir. Algoritma bu veriye bakarak çıkarımlarda bulunur, kafasında model kurar. Örn: çok sayıda spam/spam-değil diye etiketlenmiş e-posta gözlemi. Test Verisi (Test Data): Algoritmanın kafasında şekillendirdiği modelin ne kadar gerçeğe yakın olduğunu test etmek için kullanılan veri seti. Eğitim esnasında saklanır, eğitim bittikten sonra etiketsiz olarak algoritmaya verilerek algoritmanın (vermediğimiz etiketler hakkında) tahminlerde bulunması beklenir. Örn: spam olup olmadığı bilinen (ama gizlenen), eğitim verisindekilerden farklı çok sayıda e-posta gözlemi[5].

**2. Temel Bilgiler**

Programlama dili olarak C++ kullanılmıştır. C++ bilindiği gibi programlama dünyasında en çok ilgi gören C dilinden türemiştir. C++’ı klasik C dilinden farklı yapan yanı; Nesne Yönelimli Programlamayı da ( Object Oriented Programming) C’nin sözdizimi kurallarıyla birlikte desteklemesidir. Normalde C ile sadece yapısal programlama yapılırken C++ dili ile hem yapısal hem de nesne yönelimli programlar yazabilir. C++ diline başlarken önce dilin C’den gelen ve temel olan özelliklerini öğrenmek gerekir. Örneğin, değişken tanımlama, atama ve diğer basit aritmetik işlemler, kontrol yapıları ve döngüler vs.dir [2]. Biz bu projede Visual C++ kullandık. Bu dil Microsoft’un sağladığı bir dildir. Visual C++, C++ ile program yazmak üzere donatılmış, görsel yönelimli bir yazılım geliştirme ortamıdır.

**2**

C++ ya da C dilinde yazılması olası olan her türlü program Visual C++ ile yazabilir.

Derleyici olarak ilk başta Visual Studio denenmiştir. Versiyon ve kütüphane sorunları ortaya çıkmıştır bu yüzden Dev C ve CodeBlocks derleyicileri de denenmiştir fakat bu derleyicilerde hataya rastlanmıştır. En son olarak soruna bütün olarak bakılarak problem giderilmiştir. Problemin düzeltilmesi için kullanılan diziler yerine dinamik dizi görevi gören “Vektör” denilen yapı kullanılmıştır. Böylece kod Visual Studio üzerinden çalıştırılmıştır. Microsoft Visual Studio, Microsoft tarafından geliştirilen bir tümleşik geliştirme ortamıdır (IDE). Visual Studio, değişik programlama dillerini de destekler, bu da kod editörü ve hata ayıklayıcısının neredeyse tüm programlama dillerini desteklemesini sağlamaktadır. Bu derleyicinin seçilmesinin nedeni çoğu projenin bu ortamda yazılmış olmasının verdiği tecrübedir. Ayrıca projenin arayüz kısmı bu derleyicide daha kolay ve pratik bir şekilde yapılabilmektedir.

**3.Yazılım Aşaması**

Bize verilen projeye bakarsak bu ödevde Q-learning algoritması kullanılarak labirentte yol bulma probleminin çözülmesi istenmektedir. Ödevde verilen bilgiye göre labirent bir cizge olarak düşünülebilir labirentteki odacıklar düğüm olarak temsil edilirse, aralarında geçiş olan odalar bu çizgede bağlı düğümler olarak düşünülebilir. Ödevden istenilenler ise, input.txt isimli dosyadan komşuluk listesi verilen kare labirent okunacaktır. Bu dosyada başka bilgi olmayacaktır. İkinci olarak Kullanıcıdan hedef odacık, başlangıç odacık numaralar ve iterasyon sayisi istenecektir. Eğer geliştirilecek uygulama konsoldan çalışacaksa bunlar konsol parametreleri olarak grafiksel arayüz destekleyen bir uygulama geliştirilecekse arayüzden bu değerler alınabilir. Ayrıyeten projede öğrenme katsayısının 0.8 olarak kullanılacağı verilmiştir. Hedef düğüme bağlamayan geçişler için anlık kazanç 0 hedef durumuna bağlayan geçişlerin kazancı 100 olacaktır. Ayrıca her hedef durumdan

kendine kazanç 100 olan geçiş tanımlanacaktır. Son olarak ta girdi ve çıktı isterleri belirtildiği şekilde karşılanmalıdır. Labirent boyutu konusunda herhangi bir kısıtlama yoktur, ancak değerlendirme de kullanılacak labirentlerin kesinlikle kare labirent olması gerektiği söylenmiştir.

*Şekil 1*: Labirent Gösterimi

Şekil 1 e göre her bölgeyi numaralandıralım. 0 bölge için bakacak olursak bu düğümün komşuları 1 ve 2 dir. Diğer bölgeleri de sırasıyla numaralandırarak komşularına bakacak olursak 1.düğümün komşusu 0 ve 3 tür. 2.düğümün komşusu yoktur. 3.düğümün komşusu da 1 numaralı bölgedir.

**3.1 Kullanılan Sınıflar**

Projede ilk başta dizi daha sonra derleyici hatalarından dolayı verktör kullanıldığını söylenmişti. Vektör (vector) standart isim uzayının (std::) bir parçasıdır ve aslında bir sınıf şablonudur. STL (Standart Şablon Kütüphanesi)’in bir üyesidir ve bir ardışık topluluk tur. Vektör bir dinamik dizi gerçeklemesidir, programlarda hemen hemen her tür veri tipi ya da nesneyi tutan bir dizi oluşturmak için kullanılabilir. Vektör eski C stili diziler yerine, verileri dinamik olarak bir dizide tutmamız gereken ve bu dizilerin otomatik olarak genişleyip küçülmesi gerektiği durumlarda ideal bir seçim olacaktır.[3].Vektörlerin bazı fonksiyonları şunlardır:

-push\_back() fonksiyonu veriyi dizinin sonuna ekler. pop\_back() vektörün sahip olduğu son elemanı çıkarır. back() vektörün son elemanını döndürür. -swap() tüm vektör içeriğini başka bir vektöre aktarır.

**3**

-insert() vektör içinde belirtilen bir konuma eleman ekler veya çıkarır. -erase() fonksiyonu vektör içinde belirtilen bir konumda bulunan elemanı çıkarır. size() vektör boyutunu döndürür. -begin() vektör başlangıcını gösterir. -end() vektör sonunu gösterir -clear() fonksiyonu ise vektör içeriğini siler.

Dizilerden de bahsetmemiz gerekecektir. Çünkü kullandığımız matrisler de çok boyutlu dizi olarak tanımlanmaktadır. Çok boyutlu diziler genelde döngüler (loop) ile birlikte kullanılırlar. Bilindiği üzere aslında dizi (array) kavramının varlık sebebi birden fazla değişkeni hafızada bir arada tutmak ve kolayca ulaşmaktır. Dolayısıyla birden fazla değişkene erişirken indis numaralarından (satır ve sütün numaralarından) erişmek çoğu zaman avantajlı bir durumdur.

Kullanılan diğer bir yöntem ise rastgele sayı üretmek için kullanılan rand() fonksiyonudur. C Dilinde Rastgele Sayı üretmek için rand(); fonksiyonu kullanılır bunun içinde <stdlib.h> kütüphanesi tanımlamak gerekir. Srand() ile başlangıç değeri verilebilir ama sonuçlar bu seferde başlangıç değerine göre hep aynı çıkacaktır.

**3.2 Yazılım Mimarisi**

İlk olarak gerekli kütüphaneler tanımlanmıştır. Projeyi Visual Studio’da derlediğimiz için "stdafx.h" kütüphanesi de ekledik. Sonra lambda değişkenini 0,8 olarak tanımladık. Main kısmında kullanıcıdan inputları alan bir konsol menüsü tanımlanmıştır. İnputlar matrislerin boyutuna göre değişebilmektedir fakat kare matris olma zorunluluğu vardır. Dosya okuma işlemi de main içindedir. Kullanıcı işlemlerinden sonra ilk iş R matrisini oluşturmaktır.

\*\*\* std::vector<std::vector<int>> R(iliskiler.size(), std::vector<int>(iliskiler.size(), -1));

R matrisi düğüm sayısı x düğüm sayısı boyutunda başlangıçta bütün değerlerini -1 yapıyoruz.

for(int i = 0; i < iliskiler.size(); i++){

for(int j = 0; j < iliskiler.size(); j++){

R[i][j] = -1;

} }

İkinci olarak

\*\*\* 2 Boyutlu ilişkiler vektöründe gezinerek R matrisini güncelliyoruz.

i = 0 =>

j = 0 => R[0][ iliskiler[0][0] ] = 0,

İliskiler[0][0] = 1 olduğu icin R[0][1] = 0 yaparak komşu olduğunu belirtiyoruz. Bitis düğümü kendisi ile komşu olması gerektiği için en son onu da güncelliyoruz.

R[bitis][bitis] = 0;

i' den bitis' e geçiş varsa 100, bitiş' den i' ye geçiş varsa 0 bitiş kendisi ile komşu olduğu için bitiş hariç diğerleri -1 almaktadır.

std::vector<std::vector<double>> Q(iliskiler.size(), std::vector<double>(iliskiler.size(), 0));

Başlangıçta bütün değerleri 0 olacak, daha sonra öğrendikçe güncelleyeceğiz. Daha sonra Random besleyici, random sayı oluştururken bunlardan gelen verilere bakacak srand(time(NULL)); ' deki time(NULL) kısmı bu. Bir tane random durum için bir tane de random eylem için ikisini de aynı random besleyiciyle beslersek aynı sayıyı üretirler.

std::mt19937 rng; //random besleyici

std::mt19937 rng2; //random besleyici

rng.seed(std::random\_device()());

rng2.seed(std::random\_device()());

**4**

Bu işlemden sonra Q-Learning formülü uygulanmıştır. Ek kısa yolun indislerini tutmak için dinamik bir dizi oluşturulup baslangiç düğümü yol vektorüne atanmıştır.

int idx = baslangic;

yol.push\_back(baslangic);

Daha sonra bitişe ulaşana kadar başlangıçtan başlayarak bitişe ulaşana kadar Q vektöründen idx.inci satırdaki en büyük elemanın indisini bulup aynı işlemi tekrarlıyoruz.

while (idx != bitis) {

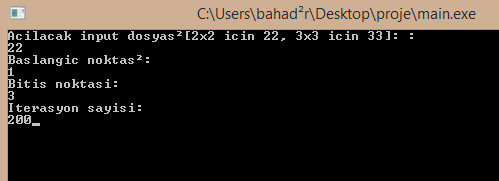
idx = (int)std::distance(Q[idx].begin(), std::max\_element(Q[idx].begin(), Q[idx].end()));

yol.push\_back(idx);

Son olarakta Sıralanmamış dizide en büyük eleman buluyor. Daha hızlı bulunabilir fakat dizinin kopyasını alıp sıralayıp büyük elemanı bulup döndürmek gerekmektedir.

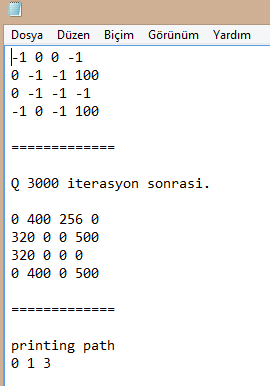
**3.3 Projenin Derlenmesi**

Projenin derlenmesini Dev C üzerinde gösterelim. Derle butonuna bastıktan sonra kullanıcının karşısına konsol ekranı gelmektedir. Bu ekranda kullanıcıya açılacak input dosyası, başlangıç-bitiş noktaları ve iterasyon sayısı sorulmaktadır.



*Şekil 2*: Konsol

Değerler girildikten sonra girdiğimiz değerlere göre yapılan işlemler output.txt dosyasına kaydedilmektedir. Bu txt te r matrisi q matrisi kazançlar ve labirentte gidilen yol görülebilmektedir.



*Şekil 3*:22Output.txt

**4.Sonuçlar**

Projede istenilen çoğu özellik çalışmaktadır. Kullanıcıdan input bilgileri alınabilmektedir. Başlangıçta varsayılan değeri -1 olan R matrisi ve bu matristen elde edilen Q matrisi verilen formüle göre oluşturulmaktadır. Doğru yol en altta gösterilmektedir. Labirent gösterilmemektedir ve arayüz kullanılmadan sadece konsol uygulaması yapılmıştır.

**5**

**5.Kaynakça**

[1] Nicolai M. Josuttis, The C++ Standard Library: A Tutorial and Reference 1st Edition

[2]https://baybarsyazilim.wordpress.com/2014/12/15/c-nedir-ne-ise-yarar/#more-194 C++

[3] <http://omerozcan.net/q-learning/>

[4] by Kevin P. Murphy, Machine Learning: A Probabilistic Perspective (Adaptive Computation and Machine Learning series) 1st Edition

[5] <http://www.emrealadag.com/makine-ogrenmesi-nedir.html>

**6**