

***AD:*** *SEDANUR*

***SOYAD:*** *PEKER*

***ÖĞRENCİ NO:*** *22010903060*

***PROJE KONUSU:*** *Q Learning Algoritması ile Tic Tac Toe Oyunu*

**Q Learning Algoritması Nedir?**

Q learning algoritması pekiştirmeli öğrenmenin en çok bilinen algoritmalarındandır. Pekiştirmeli öğrenme bir ajanın kendi eylem ve deneyimlerinden aldığı geri dönütler ile öğrenme işlemini gerçekleştirerek ödüle ulaşmasına denir. Q learning ise ortamdaki her olası durum ve eylemlerin tutulduğu bir değer tablosudur. Mevcut duruma göre bir sonraki hamleler ifade edilir ve bu hamlelerden hangisinin gerçekleşmesinin daha iyi olacağına dair bir değer öğrenilir.

Q learning algoritması ile eğiteceğimiz X oyuncusu ilk başta rastgele seçimler yapacak çünkü hamle yapmak için neyi baz alacağını bilmemektedir. Rastgele yaptığı hamleler sonucunda değerler alan ajan bu değerleri kayıt edecek ve oyunun sonuçlarından bazı çıkarımlar yaparak bir sonraki oyunda tecrübelerini kullanıp kazanmaya daha yatkın hamleler yapan bir ajan olacaktır.

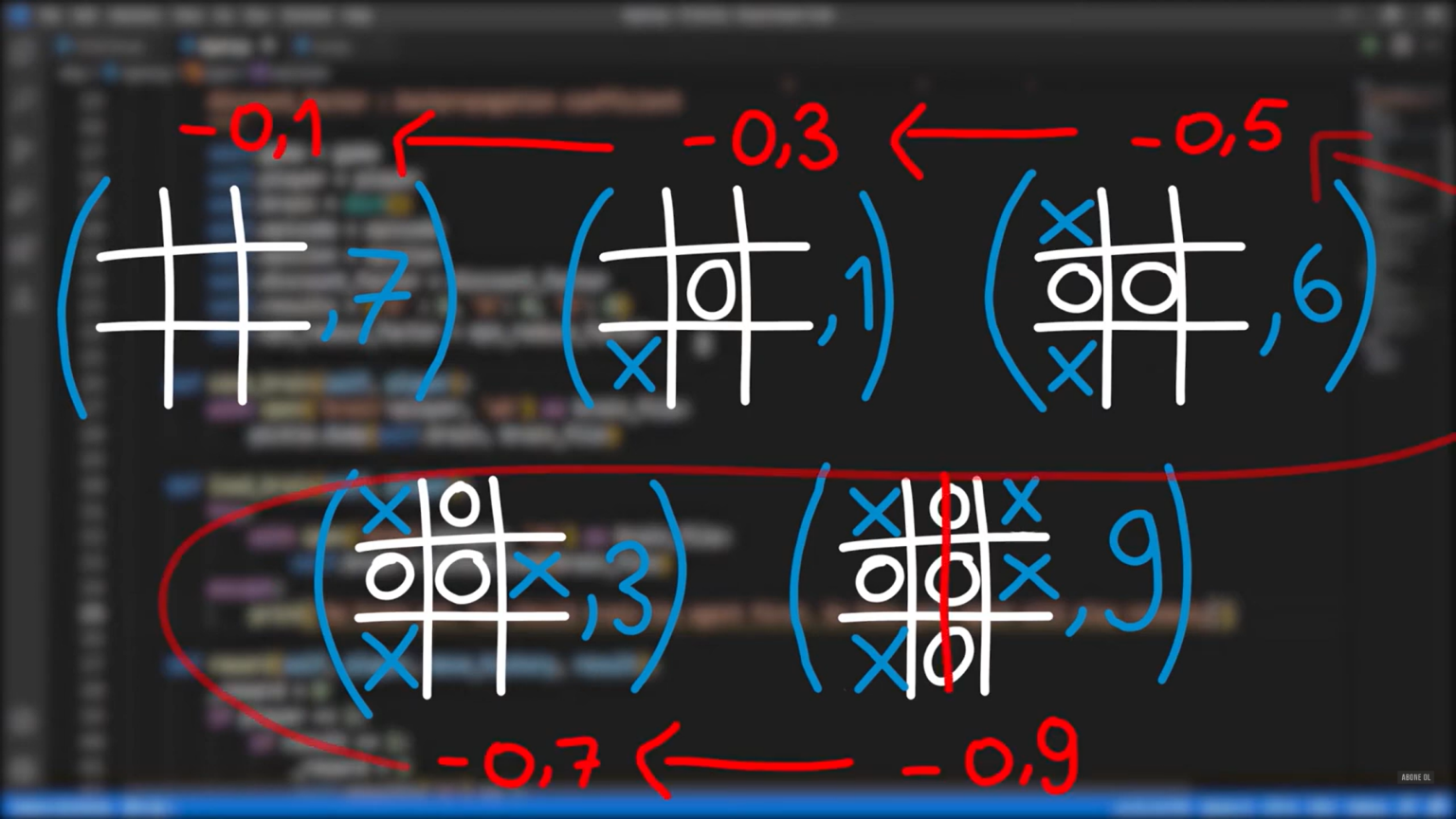
***oyun:*** Tic Tac Toe class’ındaki oyun nesnesidir.

***oyuncu:*** Ajanın X’i mi yoksa O’yu mu oynayacağını gösteren parametredir.

***beyin:*** Oyundaki hamlelerin Q değerlerini tutar.

***bolum:*** Ajanın kaç oyun ile eğitileceğini gösterir. Kodda 100.000 oyunda eğitilmiştir.

***epsilon:*** Ne sıklıkla rastgele hamleler yapacağını gösterir. Kodda bu değer 0.9’dur. Bu da ajanın %90 ihtimalle rastgele hamleler yapacağını ve %10 ihtimalle de kaydettiği tablosuna göre hamleler yapacağını gösterir. Ajanın eğitimi ilerledikçe epsilon değeri sıfıra yaklaşır ve eğitim bitiminde ajan artık tablosuna göre hamleler yapar.

***indirim\_fak:*** Ajan yaptığı hamleler sonucu ödüllendirilir. En son hamleden ilk başta yapılan hamleye doğru ödül puanı düşürülür. Çünkü gelecekteki ödüller anlık ödüllere göre daha belirsizdir ve ajanın daha kısa vadeli ödüllere odaklanması daha sağlam kararlar vermesini sağlar.

***eps\_azaltma\_fak:*** Epsilon değerinin her bin oyunda bir düşmesini istediğimiz miktardır. Kodda bu miktar 0.01’dir bu da her bin oyunda epsilon değeri 0.01 miktarında azaltılacak demektir. Yani 90.000 oyunun sonunda epsilon değerinin sıfır olacak ve son 10.000 oyunda ise ajan sadece tablodaki verileri kullanacağını göstermektedir.

Eğitim bittiğinde 76.173 kere X oyuncusu, 9.922 kere de O oyuncusunun kazandığı ve 13.905 kere de berabere kaldıkları gözlemlenmiştir.

**KODLAR**

**Agent.py**

import numpy as np  
import random  
import pickle  
  
class Ajan:  
 def \_\_init\_\_(self, oyun, oyuncu='X', bolum=100000, epsilon=0.9, indirim\_fak=0.6, eps\_azaltma\_fak=0.01):  
 self.oyun = oyun  
 self.oyuncu = oyuncu  
 self.beyin = dict()  
 self.bolum = bolum  
 self.epsilon = epsilon  
 self.indirim\_fak = indirim\_fak  
 self.sonuclar = {'X': 0, 'O': 0, 'D': 0}  
 self.eps\_azaltma\_fak = eps\_azaltma\_fak  
  
 def beyni\_kaydet(self, oyuncu):  
 with open('beyin' + oyuncu, 'wb') as beyin\_dosyasi:  
 pickle.dump(self.beyin, beyin\_dosyasi)  
  
 def beyin\_yukle(self, oyuncu):  
 try:  
 with open('beyin' + oyuncu, 'rb') as beyin\_dosyasi:  
 self.beyin = pickle.load(beyin\_dosyasi)  
 except:  
 print('Kayitli gecmis bulunmamaktadir. Bu yuzden once ajani egitmelisiniz. Bundan dolayi ajan rastgele oynayacaktir.')  
  
 def odul(self, oyuncu, move\_history, sonuc):  
 \_odul = 0  
 if oyuncu == 1:  
 if sonuc == 1:  
 \_odul = 1  
 self.sonuclar['X'] += 1  
 elif sonuc == -1:  
 \_odul = -1  
 self.sonuclar['O'] += 1  
 elif oyuncu == -1:  
 if sonuc == 1:  
 \_odul = -1  
 self.sonuclar['X'] += 1  
 elif sonuc == -1:  
 \_odul = 1  
 self.sonuclar['O'] += 1  
  
 if sonuc == -2:  
 self.sonuclar['D'] += 1  
 move\_history.reverse()  
 for durum, aksiyon in move\_history:  
 self.beyin[durum, aksiyon] = self.beyin.get((durum, aksiyon), 0.0) + \_odul  
 \_odul \*= self.indirim\_fak  
  
 def beyni\_kullan(self):  
 olasi\_aksiyonlar = self.oyun.mevcut\_pozisyon\_al()  
 max\_qdegeri = -1000  
 iyi\_aksiyon = olasi\_aksiyonlar[0]  
 for aksiyon in olasi\_aksiyonlar:  
 qdegeri = self.beyin.get((self.oyun.mevcut\_tuple\_al(), aksiyon), 0.0)  
 if qdegeri > max\_qdegeri:  
 iyi\_aksiyon = aksiyon  
 max\_qdegeri = qdegeri  
 elif qdegeri == max\_qdegeri and random.random() < 0.5:  
 iyi\_aksiyon = aksiyon  
 max\_qdegeri = qdegeri  
 elif len(olasi\_aksiyonlar) == 9:  
 iyi\_aksiyon = random.choice(olasi\_aksiyonlar)  
 break  
 return iyi\_aksiyon  
  
 def x\_rastgele\_egit(self):  
 for \_ in range(self.bolum):  
 if \_ % 1000 == 0:  
 print('Bolum: ' + str(\_))  
 self.epsilon -= self.eps\_azaltma\_fak  
 move\_history = []  
 while True:  
 if sum(self.oyun.mevcut\_oyunu\_al() == 1) == 0 or random.random() < self.epsilon:  
 mevcut\_aksiyonlar = self.oyun.mevcut\_pozisyon\_al()  
 eylem\_x = random.choice(mevcut\_aksiyonlar)  
 move\_history.append([self.oyun.mevcut\_tuple\_al(), eylem\_x])  
 self.oyun.hareket\_et(eylem\_x)  
 else:  
 eylem\_x = self.beyni\_kullan()  
 move\_history.append([self.oyun.mevcut\_tuple\_al(), eylem\_x])  
 self.oyun.hareket\_et(eylem\_x)  
 if self.oyun.kazandi\_mi():  
 self.odul(1, move\_history, self.oyun.kazanan)  
 break  
 mevcut\_aksiyonlar = self.oyun.mevcut\_pozisyon\_al()  
 eylem\_o = random.choice(mevcut\_aksiyonlar)  
 self.oyun.hareket\_et(eylem\_o)  
 if self.oyun.kazandi\_mi():  
 self.odul(1, move\_history, self.oyun.kazanan)  
 break  
 self.beyni\_kaydet('X')  
 print('Egitim Tamamlandi.')  
 print('Sonuclar:')  
 print(self.sonuclar)  
  
 def o\_rastgele\_egit(self):  
 for \_ in range(self.bolum):  
 if \_ % 1000 == 0:  
 print('Bolum: ' + str(\_))  
 self.epsilon -= self.eps\_azaltma\_fak  
 move\_history = []  
 while True:  
 mevcut\_aksiyonlar = self.oyun.mevcut\_pozisyon\_al()  
 eylem\_x = random.choice(mevcut\_aksiyonlar)  
 self.oyun.hareket\_et(eylem\_x)  
 if self.oyun.kazandi\_mi():  
 self.odul(-1, move\_history, self.oyun.kazanan)  
 break  
 if random.random() < self.epsilon:  
 mevcut\_aksiyonlar = self.oyun.mevcut\_pozisyon\_al()  
 eylem\_o = random.choice(mevcut\_aksiyonlar)  
 move\_history.append([self.oyun.mevcut\_tuple\_al(), eylem\_o])  
 self.oyun.hareket\_et(eylem\_o)  
 else:  
 eylem\_o = self.beyni\_kullan()  
 move\_history.append([self.oyun.mevcut\_tuple\_al(), eylem\_o])  
 self.oyun.hareket\_et(eylem\_o)  
 if self.oyun.kazandi\_mi():  
 self.odul(-1, move\_history, self.oyun.kazanan)  
 break  
 self.beyni\_kaydet('O')  
 print('Egitim tamamlandi.')  
 print('Sonuclar:')  
 print(self.sonuclar)  
  
 def insan\_ile\_oyna(self):  
 self.beyin\_yukle(self.oyuncu)  
 emir = 1 if self.oyuncu == 'X' else -1  
 while True:  
 if emir == 1:  
 self.oyun.hareket\_et(self.beyni\_kullan())  
 self.oyun.mevcut\_oyunu\_ciz()  
 emir \*= -1  
 if self.oyun.kazandi\_mi(isgame=True):  
 break  
 else:  
 eylem\_o = int(input('Hangi kareye oynayacaksiniz?'))  
 self.oyun.hareket\_et(eylem\_o - 1)  
 self.oyun.mevcut\_oyunu\_ciz()  
 emir \*= -1  
 if self.oyun.kazandi\_mi(isgame=True):  
 break  
 def bilgisayar\_ile\_oyna(self):  
 self.beyin\_yukle('X')  
 self.beyin\_yukle('O')  
 emir = 1  
 while True:  
 if emir == 1:  
 self.oyun.hareket\_et(self.beyni\_kullan())  
 self.oyun.mevcut\_oyunu\_ciz()  
 if self.oyun.kazandi\_mi(isgame=True):  
 break  
 else:  
 self.oyun.hareket\_et(self.beyni\_kullan())  
 self.oyun.mevcut\_oyunu\_ciz()  
 if self.oyun.kazandi\_mi(isgame=True):  
 break  
 emir \*=-1

**TicTacToe.py**

import numpy as np  
  
class TicTacToe:  
 def \_\_init\_\_(self):  
 self.mevcut\_durum = np.zeros(9, dtype=np.int8)  
 self.kazanan = None  
 self.oyuncu = 1  
  
 def mevcut\_oyunu\_ciz(self):  
 mevcut\_durum = ['X' if x == 1 else 'O' if x == -1 else '--' for x in self.mevcut\_durum]  
 print(f'{mevcut\_durum[0]:^5} {mevcut\_durum[1]:^5} {mevcut\_durum[2]:^5}')  
 print(f'{mevcut\_durum[3]:^5} {mevcut\_durum[4]:^5} {mevcut\_durum[5]:^5}')  
 print(f'{mevcut\_durum[6]:^5} {mevcut\_durum[7]:^5} {mevcut\_durum[8]:^5}')  
 print('\_' \* 25)  
  
 def mevcut\_oyunu\_al(self):  
 return self.mevcut\_durum  
  
 def mevcut\_tuple\_al(self):  
 return tuple(self.mevcut\_durum)  
  
 def mevcut\_pozisyon\_al(self):  
 return (np.argwhere(self.mevcut\_durum == 0).ravel())  
  
 def oyunu\_sifirla(self):  
 self.mevcut\_durum = np.zeros(9, dtype=np.int8)  
 self.oyuncu = 1  
  
 def oyuncu\_al(self):  
 return self.oyuncu  
  
 def hareket\_et(self, aksiyon): *# player is 1 for X, player is -1 for O* if aksiyon in self.mevcut\_pozisyon\_al():  
 self.mevcut\_durum[aksiyon] = self.oyuncu  
 *# self.draw\_current\_game()* self.oyuncu \*= -1  
 else:  
 print('Mevcut degil.')  
  
 def \_hareket\_et(self, \_mevcut\_durum, aksiyon):  
 \_mevcut\_durum[aksiyon] = self.oyuncu  
 return \_mevcut\_durum  
  
 def sonraki\_durum(self):  
 durum = []  
 \_mevcut\_durum = self.mevcut\_durum  
 \_uygun\_hareketler = self.mevcut\_pozisyon\_al()  
 for move in \_uygun\_hareketler:  
 durum.append(self.\_hareket\_et(\_mevcut\_durum=\_mevcut\_durum, aksiyon=move))  
 return durum  
  
 def kazandi\_mi(self, isgame=False):  
 kazanan\_koordinatlar = np.array([[0, 1, 2], [3, 4, 5], [6, 7, 8],  
 [0, 3, 6], [1, 4, 7], [2, 5, 8],  
 [0, 4, 8], [2, 4, 6]])  
 for koordinat in kazanan\_koordinatlar:  
 toplam = sum(self.mevcut\_durum[koordinat])  
 if toplam == 3:  
 if isgame:  
 print('KAZANAN: X')  
 self.kazanan = 1  
 self.oyunu\_sifirla()  
 return 1  
 elif toplam == -3:  
 if isgame:  
 print('KAZANAN: O')  
 self.kazanan = -1  
 self.oyunu\_sifirla()  
 return -1  
 elif sum(self.mevcut\_durum == 1) == 5:  
 if isgame:  
 print('BERABERE')  
 self.kazanan = -2  
 self.oyunu\_sifirla()  
 return -2  
 return False

**run.py**

from TicTacToe import TicTacToe  
from Agent import Ajan  
  
oyun = TicTacToe()  
ajan = Ajan(oyun, 'X',indirim\_fak = 0.6, bolum = 100000)  
  
*#ajan.x\_rastgele\_egit()  
#ajan.bilgisayar\_ile\_oyna()*ajan.insan\_ile\_oyna()

**KAYNAKÇA**

<https://medium.com/deep-learning-turkiye/q-learninge-giri%C5%9F-6742b3c5ed2b#:~:text=Q%2DLearning%20algoritmas%C4%B1%2C%20peki%C5%9Ftirmeli%20%C3%B6%C4%9Frenmenin,maximize)%20buna%20g%C3%B6re%20hareket%20etmektir>.

<https://www.muhendisbeyinler.net/pekistirmeli-ogrenme-reinforcement-learning-nedir/>

<https://github.com/ardaakdere/Q-Learning-TicTacToe/tree/main>

<https://www.youtube.com/watch?v=Qy2B4Xvpf-U&list=LL>