AD: *SEDANUR*

SOYAD: *PEKER*

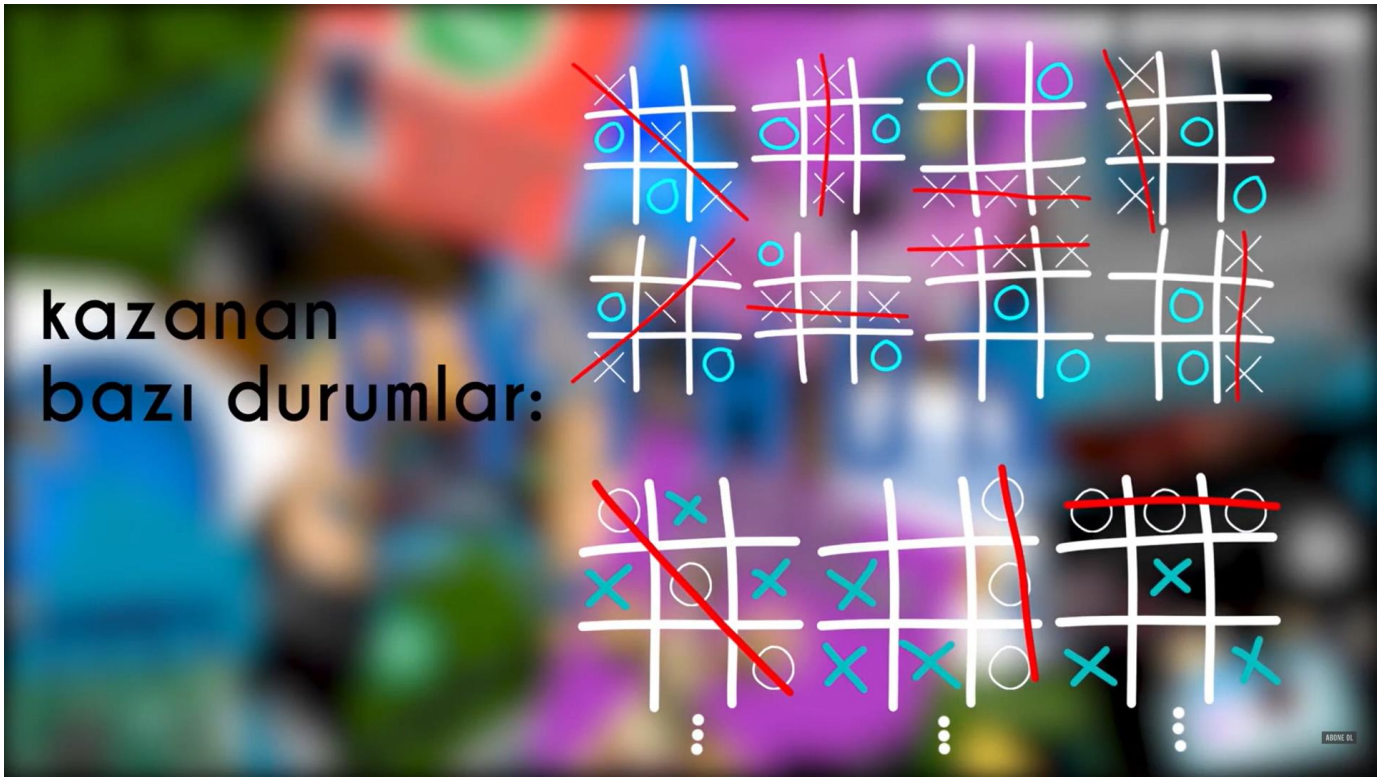
ÖĞRENCİ NO: *22010903060*

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.

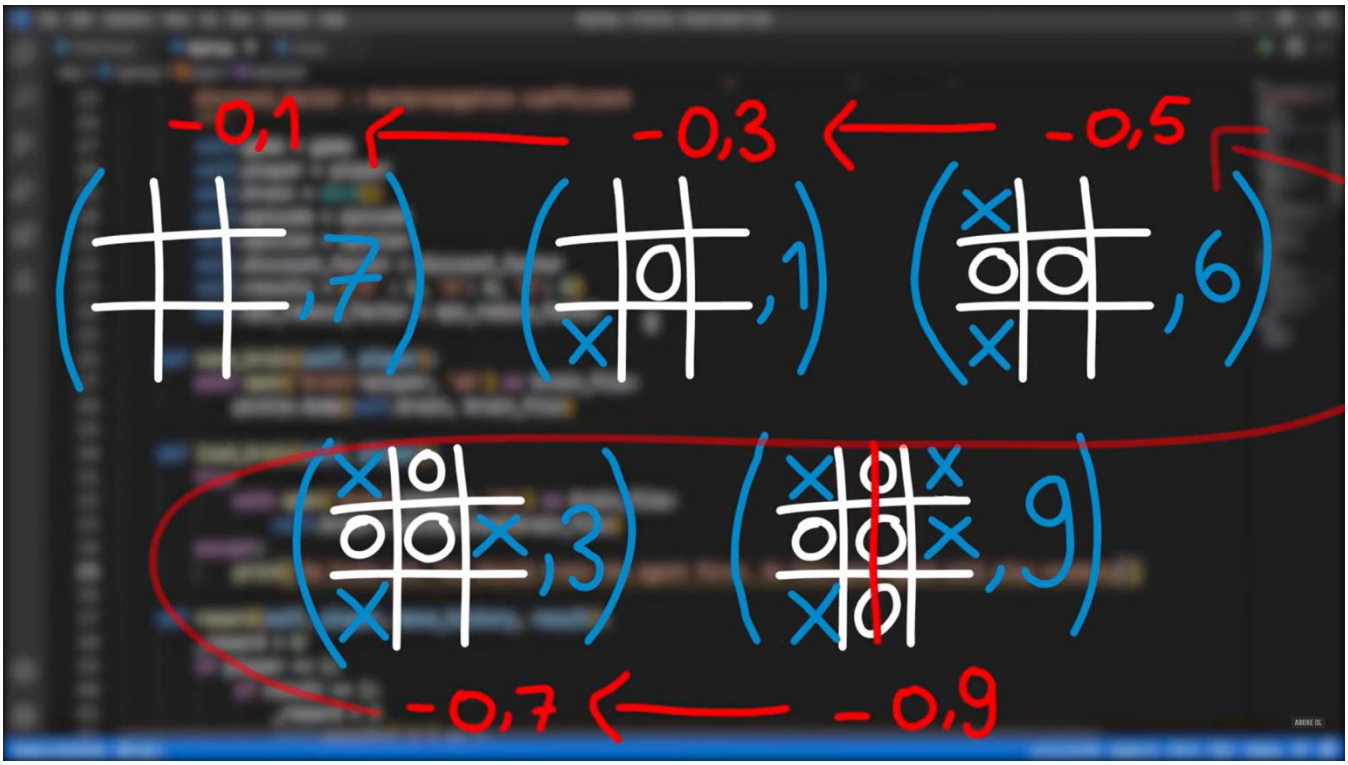
oyuncu: Ajanın X'i mi yoksa O'yu mu oynayacağını gösteren parametredir.

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