SAKARYA
UYGULAMALI BİLİMLER
ÜNİVERSİTESİ

Ders : Yapay Zeka İlkeleri

Öğretim Görevlisi : Ekin EKİNCİ

Grup Üyeleri : *Barış BARTİN

*Cihangir İNCAZ

*Hasan Ifreitekh

Proje Konusu : Tic Tac Toe

Kullanılan Library :

Bu projemizde Python diline ait Numphy kütüphanesini kullandık. NumPykütüphanesi , Python programlama dili için geliştirilmiş açık kaynaklı bir bilimsel hesaplama kütüphanesidir. Temel olarak büyük, çok boyutlu diziler ve matrislerle çalışmak için yüksek performanslı araçlar sunar. NumPy, matematiksel ve mantıksal işlemler, Fourier dönüşümleri, doğrusal cebir işlemleri ve rastgele sayı üretimi gibi geniş bir yelpazede işlevsellik sağlar.

NumPy, yüksek performanslı hesaplamalar gerektiren bilimsel ve mühendislik uygulamaları için ideal bir kütüphanedir. Çok yönlülüğü ve kullanım kolaylığı sayesinde, veri analizi, makine öğrenimi, finans ve daha birçok alanda yaygın olarak kullanılmaktadır.

```python
import numpy as np

class TicTacToe:
    """
    guncel_durum: [0 X 0 0 0 0 0 0 0] gibi mevcut oyun durumunu tutar
    """

    def __init__(self):
        self.guncel_durum = np.zeros(9, dtype=np.int8)
        self.kazanan = None
        self.oyuncu = 1

    def guncel_oyunu_ciz(self):
        guncel_durum = ['X' if x == 1 else 'O' if x == -1 else '--' for x in self.guncel_durum]
        print(f'{guncel_durum[0]:^5} {guncel_durum[1]:^5} {guncel_durum[2]:^5}')
        print(f'{guncel_durum[3]:^5} {guncel_durum[4]:^5} {guncel_durum[5]:^5}')
        print(f'{guncel_durum[6]:^5} {guncel_durum[7]:^5} {guncel_durum[8]:^5}')
        print('_' * 15)
```

İlk başta oyunun başlangıç durumu __init__ (self) ile tanımlanır oyun boş bir oyun tahtası ve “x” i oynacak kişi ile başlar.Daha sonra güncel_oyunu_ciz tanımlaması ile oyun her hamle yapıldıkça güncel_oyunu_al(self) bloğu çalışır ve güncel durumu döndürür.Güncel duruma bakılırak terminalde tekrar çizilir.

```python
def guncel_oyunu_al(self):
    return self.guncel_durum

def guncel_oyun_tupu_al(self):
    return tuple(self.guncel_durum)

def musait_pozisyonlari_al(self):
    return np.argwhere(self.guncel_durum == 0).ravel()

def oyunu_sifirla(self):
    self.guncel_durum = np.zeros(9, dtype=np.int8)
    self.oyuncu = 1

def oyuncuyu_al(self):
    return self.oyuncu
```

Güncel oyunu tüple ve default olarak döndürme,müsait pozisyonlar kısmında boş olan “0” ların indeklersini döndürme oyunu sıfırlama işlemi yani başa alma işlemi ve en sonunda oyuncu sırasını döndürme işlemi tanımlanmıştır

```python
def hareket_yap(self, hareket):  # oyuncu 1 ise X, oyuncu -1 ise O
    if hareket in self.musait_pozisyonlari_al():
        self.guncel_durum[hareket] = self.oyuncu
        self.oyuncu *= -1
    else:
        print('Bu pozisyon uygun değil')
```

Yapay zeka kısmında yani self kısmında müsait pozisyonlara bakılıp hamle yapılır ve sıradaki oyuncuya sıra gelir

```python
def _hareket_yap(self, _guncel_durum, hareket):
    _guncel_durum[hareket] = self.oyuncu
    return _guncel_durum
```

Bu kısım yapay zeka kısmı içindir burda kendi kendine oynaması için gerekli bir fonksiyondur sıra usteki gibi self.oyuncu *= -1 yazılmadığından oyuncuyu ilgilendiren bir kısım değildir

```python
def sonraki_durumlari_al(self):
    durumlar = []
    _guncel_durum = self.guncel_durum
    _musait_hareketler = self.musait_pozisyonlari_al()
    for hareket in _musait_hareketler:
        durumlar.append(self._hareket_yap(_guncel_durum=_guncel_durum, hareket=hareket))
    return durumlar

def kazanan_var_mi(self, oyun_devam=False):
    kazanan_koordinatlari = 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_koordinatlari:
        toplam = sum(self.guncel_durum[koordinat])
        if toplam == 3:  # X kazandı
            if oyun_devam:
                print('X Kazandı!')
            self.kazanan = 1
            self.oyunu_sifirla()
            return 1
        elif toplam == -3:  # O kazandı
            if oyun_devam:
                print('O Kazandı!')
            self.kazanan = -1
            self.oyunu_sifirla()
            return -1
        elif sum(self.guncel_durum == 1) == 5:  # Beraberlik
            if oyun_devam:
                print('Beraberlik')
            self.kazanan = -2
            self.oyunu_sifirla()
            return -2
    return False
```

İlk başta durumlar dizisi oluşturulup boş boşluktaki hareketler hamle sayısı bitene kadar döngüye sokulup olabilecek hamleler hesaplanıp durum dizisine atanır ve döndürülür

Kazanan_var_mi sorgusu ile kazanmak için gereken kombinasyonlar tanımlanıp her hamle sonrası sorgulanıp kazanan 3 olursa(yanı kombinasyonlardan biri ile eşleşirse) oyun sıfırlanır.

Kazanan -3 olursa (yani “0” kullanıcısı kazanırsa) yine oyun sıfırlanır diğer durumda ise beraberlik ihtimali kalmıştır ordada oyun sonrasında sıfırlanır ve çıktı ekranına “beraberlik” yazılır burda 3 ve -3 sayıları TicTacToe oyununda 3 tane x ya da 3 tane belli kombinasyonda berabere gelmesinden kaynaklanır burda yapılan sorguda eşleşen her x için +1 eşleşen her o için -1 kullanılmıştır 3 tane eşleşme sonucu kazanan açığa çıkar.

TicTacToe tanımlamasını yaptığımıza göre Yapay zeka ajanımızın tanımlamasına geçelim.

```python
import numpy as np
import random
import pickle

class Ajan:
    def __init__(self, oyun, oyuncu='X', bolum=100000, epsilon=0.9, indirim_faktoru=0.6, epsilon_azaltma_faktoru=0.01):
```

Burda __init metodu ile alınacak parametreler belirlenmiştir.

```python
self.oyun = oyun
self.oyuncu = oyuncu
self.beyin = dict()
self.bolum = bolum
self.epsilon = epsilon
self.indirim_faktoru = indirim_faktoru
self.sonuclar = {'X': 0, 'O': 0, 'B': 0}
self.epsilon_azaltma_faktoru = epsilon_azaltma_faktoru
```

Burda oyun TicTacToe olarak belirlenmiştir oyuncu kısmında ise ajanın “X” ya da “O” olacağına karar verilir , beyin kısmı ise Q learning formatının değerlerinin tutulacağı sözlük olarak düşünebiliriz parametreleri (durum,hareket) = Q-learning değeri olarak düşünebiliriz. Bölüm kısmı ise yapay zekanın eğitim için oynayacağı oyun sayısını ifade eder. Epsilon değeri ise 0-1 arası değerler alıp ajanın kıyaslama sonrası rastgele ya da q-learning değerine göremi hareket edeceğini öğrenmek içindir. İndirim faktörü ise her bölüm sonrası Q-learning algoritmasında alacağımız değeri bölüm sayısı artıkça ne kadar dikkate alacağımızın değeridir. Sonuçlar ise ajanın eğitim sonucu kazandığı ,kaybettiği ve berabere kaldığı oyun sayısını tutar. Epsilon azaltma faktörü ise eğitim sırasında epsilon değerinin ne kadar azaltılacağını belirler.

```python
def beyin_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('Henüz bir beyin yok. Önce ajanı eğitmelisiniz. Bu nedenle bu oyun ajanı rastgele oynayacak')
```

Burda kaydet metodu ajanın Q-learning değerlerini dosyaya kaydeder yükle metodu ise daha önce kaydedilmiş beyni dosyadan yükler eğer beyin dosyası

yoksa çıktı olarak “Henüz bir beyin yok. Önce ajanı eğitmelisiniz. Bu nedenle bu oyun ajanı rastgele oynayacak” metin bloğunu gösterir.

```python
def odullendir(self, oyuncu, hareket_gecmisi, 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['B'] += 1
    hareket_gecmisi.reverse()
    for durum, hareket in hareket_gecmisi:
        self.beyin[durum, hareket] = self.beyin.get((durum, hareket), 0.0) + _odul
        _odul *= self.indirim_faktoru
```

Burda oyuncunun hareket geçmişine göre ödül değeri belirlenir ve hareket sayısı kadar her hareket geçmişi için beyin güncellenir. Ödül tablomuzun yanı Q-learning algortimasının değerleri burda belirlenir. Ve Q değerleri indirim faktörü dikkate alınarak hesaplanır.

```python
def beyin_kullan(self):
    olasi_hareketler = self.oyun.musait_pozisyonlari_al()
    max_qdegeri = -1000
    en_iyi_hareket = olasi_hareketler[0]
    for hareket in olasi_hareketler:
        qdegeri = self.beyin.get((self.oyun.guncel_oyun_tupu_al(), hareket), 0.0)
        if qdegeri > max_qdegeri:
            en_iyi_hareket = hareket
            max_qdegeri = qdegeri
        elif qdegeri == max_qdegeri and random.random() < 0.5:
            en_iyi_hareket = hareket
            max_qdegeri = qdegeri
        elif len(olasi_hareketler) == 9:
            en_iyi_hareket = random.choice(olasi_hareketler)
            break
    return en_iyi_hareket
```

Q-değerlerine göre en iyi hareketi seçer. Aynı Q-değerine sahip hareketler arasında rastgele seçim yapar. Eğer bütün hareketler eşit ise, rastgele bir hareket seçer.

```python
def beyin_egit_x_rastgele(self):
    for _ in range(self.bolum):
        if _ % 1000 == 0:
            print('Bölüm: ' + str(_))
            self.epsilon -= self.epsilon_azaltma_faktoru
        hareket_gecmisi = []
        while True:
            if sum(self.oyun.guncel_oyunu_al() == 1) == 0 or random.random() < self.epsilon:
                musait_hareketler = self.oyun.musait_pozisyonlari_al()
                hareket_x = random.choice(musait_hareketler)
                hareket_gecmisi.append([self.oyun.guncel_oyun_tupu_al(), hareket_x])
                self.oyun.hareket_yap(hareket_x)
            else:
                hareket_x = self.beyin_kullan()
                hareket_gecmisi.append([self.oyun.guncel_oyun_tupu_al(), hareket_x])
                self.oyun.hareket_yap(hareket_x)
            if self.oyun.kazanan_var_mi():
                self.odullendir(1, hareket_gecmisi, self.oyun.kazanan)
                break
            musait_hareketler = self.oyun.musait_pozisyonlari_al()
            hareket_o = random.choice(musait_hareketler)
            self.oyun.hareket_yap(hareket_o)
            if self.oyun.kazanan_var_mi():
                self.odullendir(1, hareket_gecmisi, self.oyun.kazanan)
                break
    self.beyin_kaydet('X')
    print('EĞİTİM TAMAMLANDI!')
    print('SONUÇLAR:')
    print(self.sonuclar)
```

X oyuncusu için Q-öğrenme eğitimini gerçekleştirir. O oyuncusunun hareketlerini rastgele yapar. Eğitim sonunda beyin dosyasını kaydeder ve sonuçları ekrana yazar. Her bölümde epsilon değeri belirli bir oranda azaltılır.

```python
def beyin_egit_o_rastgele(self):
    for _ in range(self.bolum):
        if _ % 1000 == 0:
            print('Bölüm: ' + str(_))
            self.epsilon -= self.epsilon_azaltma_faktoru
        hareket_gecmisi = []
        while True:
            musait_hareketler = self.oyun.musait_pozisyonlari_al()
            hareket_x = random.choice(musait_hareketler)
            self.oyun.hareket_yap(hareket_x)
            if self.oyun.kazanan_var_mi():
                self.odullendir(-1, hareket_gecmisi, self.oyun.kazanan)
                break
            if random.random() < self.epsilon:
                musait_hareketler = self.oyun.musait_pozisyonlari_al()
                hareket_o = random.choice(musait_hareketler)
                hareket_gecmisi.append([self.oyun.guncel_oyun_tupu_al(), hareket_o])
                self.oyun.hareket_yap(hareket_o)
            else:
                hareket_o = self.beyin_kullan()
                hareket_gecmisi.append([self.oyun.guncel_oyun_tupu_al(), hareket_o])
                self.oyun.hareket_yap(hareket_o)
            if self.oyun.kazanan_var_mi():
                self.odullendir(-1, hareket_gecmisi, self.oyun.kazanan)
                break
    self.beyin_kaydet('O')
    print('EĞİTİM TAMAMLANDI!')
    print('SONUÇLAR:')
    print(self.sonuclar)
```

“O” oyuncusu için Q-öğrenme eğitimini gerçekleştirir. X oyuncusunun hareketlerini rastgele yapar. Eğitim sonunda beyin dosyasını kaydeder ve sonuçları ekrana yazar.

```python
def insanla_oyna(self):
    self.beyin_yukle(self.oyuncu)
    sirasi = 1 if self.oyuncu == 'X' else -1
    while True:
        if sirasi == 1:
            self.oyun.hareket_yap(self.beyin_kullan())
            self.oyun.guncel_oyunu_ciz()
            sirasi *= -1
            if self.oyun.kazanan_var_mi(oyun_devam=True):
                break
        else:
            hareket_o = int(input('Hangi kare?'))
            self.oyun.hareket_yap(hareket_o - 1)
            self.oyun.guncel_oyunu_ciz()
            sirasi *= -1
            if self.oyun.kazanan_var_mi(oyun_devam=True):
                break
```

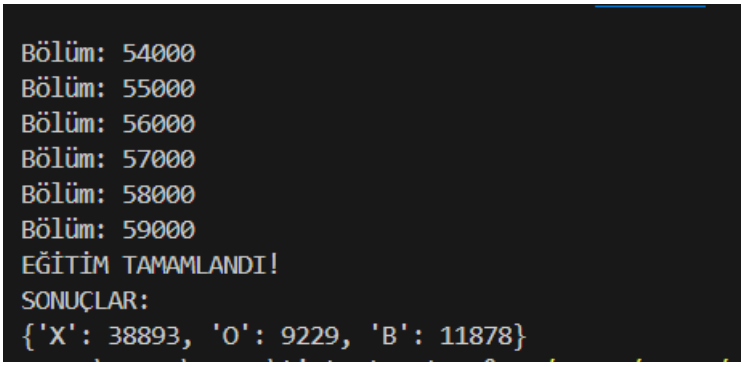
Ajanın oyuncuya karşı oynamasını sağlar. Beyin dosyasını yükler ve oyuncu sırayla hamle yapar. Oyun durumunu ekrana çizer ve oyunun sonucunu kontrol eder.

Son olarakta main dosyamızı inceleyelim

```python
from TicTacToe import TicTacToe
from Agent import Ajan

oyun = TicTacToe()

# X oyuncusu için  = indirim_faktoru = 0.6
# O oyuncusu için  = indirim_faktoru = 0.5 tavsiye edilir.
ajan = Ajan(oyun, 'X', indirim_faktoru = 0.6, bolum = 60000)

# ajan.insanla_oyna() --->burda beyinX dosyasının olmasına dikkat edin eğer dosya yoksa
# Q-learning tablosu null olacağından yapay zeka rastegele hamle yapacaktır

# ajan.beyin_egit_x_rastgele()  -----> beyinX dosyamızın oluşacağı kısım Q learning tablosunun dolacağı kısım
# olarakta nitelenebilir.
```

Burda TicTacToe dosyasında tanımladığımız oyunumuzu oyun değişkeni içine atıyoruz daha sonra Ajan sınıfından ajan isimli x ile oynayayan oynayacağı oyun sayısı ve indirim faktörü belli olan nesnemizi oluşturuyoruz daha sonra bu nesneyi eğitmek istersek ajan.beyin_egit_x_rastgele() komutu ile ajanımıza 60000 oyun oynatıp beyinX adlı dosyamızı oluşturmuş ve ödül tablosu değerlerimizi bulmuş oluyoruz daha sonra ajan ile oynamak için ajan.insanla_oyna() komutu ile oyunumuzu test edebiliriz.

60000 oyun sonrası yapay zeka eğitim sonucumuz:



Yapay zeka ile oynanan bir oyunun sonucu: