در ابتدا درباره کلاس agent و فانکشن هایی که در آن پیاده سازی کردیم صحبت میکنم.

برای یادگیری agent از qlearning استفاده می کنیم.که فرمول آ مطابق زیر است

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + (\alpha)\,[sample]$$

پس در کانستراکتور کلاس agent چندین متغیر داریم که به صورت زیر است

1.discount:که میزان ارزشی است که به state اینده می دهیم.

2.epsilon:برای این هست اکشنی که انتخاب می کنیم چه میزان بر اساس اطلاعات قبلی باشد و چه میزان به صورت رندوم باشد مخصوصا در ابتدا که هنوز تجربه ای نداریم مقدار epsilon باید بیشتر باشد و به مرور زمان کم بشود.

3.learning rate:همان الفا در فرمول بالا است که نشان دهنده ان است که چه میزان باید به تجربه ای که اکنون کردیم اهمیت بدیم و چه میزان به تجربه ای که بعدا می کنیم و learning rate هم مانند epsilon باید به مرور زمان کم بشود چون تجربه ما نیز بیشتر میشود.

4.numdiscrete: که برابر تعداد نقاطی است که در ان فضای پیوسته انتخاب می کنیم.

5.state: که در ابتدا برابر با none است و در هر مرحله اپدیت می شود.

6.action: که در ابتدا برابر با none است و در هر مرحله اپدیت می شود.

7.numberaction:که سه حرکت می توانیم انجام دهیم چپ و حرکتی نکنیم و راست که به ترتیب به ان ها اعداد 0و1و2 را نسبت می دهیم.

8.numberstates : که برابر با تعداد state های ما است در بالا اشاره تعدادی نقطه را از فضای پیوسته خود انتخاب می کنیم در واقع طبق observationspace ما دو المان داریم یکی پوزیشن ماشین در محور x و دیگری سرعت ماشین است ما numdiscrete نقطه را در محور x انتخاب کردیم و هر کدوم از این نقاط می توانند numdiscrete سرعت داشته باشند پس ما در مجموع numdiscrete به توان دو تا state داریم.

9.qtable : جدول qtable ما یک ارایه دو بعدی است که ابعاد ان برابر با تعداد state ها در تعداد اکشن ها است که در ابتدا تمام خانه های ان 0 است و در هر مرحله اپدیت می شود.

10.discretestates:گفتیم numdisrete نقطه در محور x و که هر نقطه در نقطه محور x میتونیم numdiscrete سرعت داشته باشیم حالا مشخص کنیم این state های ما هر کدوم در محور x و هم چنین سرعتشون چه عددی است این کار را با تابع linspace استفاده می کنیم با توجه به اینکه حد بالا و پایین را هم رو محور x و هم در سرعت داریم میتوانیم این کار بکنیم یعنی مثلا روی محور x حد پایین ما برابر با 1.2- و حد بالا ما برابر با 0.6 است ما باید به تعداد numdiscrete نقطه بین این دو مقدار داشته باشیم که فواصل بینشون با هم برابر است برای سرعت هم به همین صورت است که ان به صورت زیر است:

```
self.discretestates = [np.linspace(-1.2, 0.6,numdiscrete+2)[1:-1]
                       ,np.linspace(-0.07, 0.07,numdiscrete+2)[1:-1]]
```

توابع:

1.findstateinqtable:طبق observationشده یک ارایه است که یک المت ان پوزیشن ان روی محور x و یک عدد دیگر آن سرعت ماشین است ما باید با استفاده از این اطلاعات یک عدد به این state نسبت بدهیم که بین 0 تا numdiscrete به توان دو باشد تا بتوانیم ان را در جدول qtable خو ذخیره کنیم observation[0] برابر با پوزیشن ماشین روی محور x است با استفاده از discretestates می ایم نزدیکترین عددی که در المان اول این ارایه یعنی همون محور x است را پیدا میکنیم برای سرعت نیز همین کار انجام میدیم این کار می توانیم با استفاده از تابع digitize انجام بدهیم که index مربوط به ان نزدیکترین عدد را در discrestate بر می گرداند برای سرعت نیز همین کار میکنیم و بعد برای اینکه باید این عددی که به state میدهیم یکتا باشد اندیسی که برای سرعت به دست می اوریم را در numdiscrete ضرب میکنیم.کد ان به صورت زیر است:

```
def findstateinqtable(self, observation):
    s1=np.digitize(observation[0],self.discretestates[0])
    s2=np.digitize(observation[1],self.discretestates[1])*(self.numdiscret
    s3=s1+s2
    return s3
```

2.startaction: هر اپیزود محیطمان را ریست میکنیم و هم چنین باید مقادیر learningrate وepsilon را نیز اپدیت کنیم یعنی باید در هر اپیزود مقادیرشون رو کم بکنیم طبق توضیحاتی که همون اول دادم و در مرحله شروع طبق جدول qtable اون اکشنی انتخاب می کنیم که مقدار بیشتری داشته باشد کد آن به صورت زیر است:

```
def startactionep(self,observation):
    self.epsilon=self.epsilon*(0.995)
    self.learningrate = self.learningrate*(0.99995)
    if self.learningrate<pow(10,-5):
        self.learningrate=pow(10,-5)
    self.state = self.findstateinqtable(observation)
    nextact=0
    maxq=np.max(self.qtable[self.state,:])
    if maxq==self.qtable[self.state,0]:
        nextact=0
    elif maxq==self.qtable[self.state,1]:
        nextact=1
    else:
        nextact=2
    return nextact
```

chooseaction.3:در این مرحله باید اکشن مناسب را انتخاب بکنیم که این با استفاده از فرمولی هست که در ابتدای داک نوشتم کد ان به صورت زیر است

```
def chooseaction(self, observation, reward):
    nextstate = self.findstateinqtable(observation)
    nextact=0
    if (1 - self.epsilon) <= np.random.uniform():
        nextact = np.random.randint(0,3)
    else:
        maxq=np.max(self.qtable[nextstate,:])
        if maxq==self.qtable[nextstate,0]:
            nextact=0
        elif maxq==self.qtable[nextstate,1]:
            nextact=1
        else:
            nextact=2
    sample=reward+self.discount*(np.max(self.qtable[nextstate,:]))
    self.qtable[self.state, self.action] +=self.learningrate*(sample-s
```