Laut Text ist Environement Reward 0 wenn es geschafft -> Reward ist im Code aber immer -1 (Standartwert), Wert von 200 an max Schritten steht auch nicht im Codekm

Done wird aber nach 200 Versuchen trotzdem auf 0 gesetzt, der reward aber nie auf 0

Das macht aber keinen Unterschied, weil andere Werte bis dahin eh kleiner sind als -1 -> -1 maximum

Numstate/Q[0]=position, numstates/Q[1]=geschwindigkeit,Q[2]=möglicher Schritt(vor, zurück,stehen bleiben)

* Für jede Aktion an bestimmter Position mit bestimmter Geschwindigkeit wird reward gespeichert

Am Anfang alle Werte für Q zwischen 0 und 1 zufällig

Größeres Epsilon -> Höhere Wahrscheinlichkeit für zufälligen State, da kleinerer Wert von 1 abgezogen wird

Np.random.random: Wert zwischen 0 und 1

Epsilon zwischen 1-0, min Eps kleiner als Epsilon

Explorationzeit dauert länger bei größerer Episodenanzahl, weil Episoden in der Berechnung der epsilon reduktion mit einfließt

Discount faktor-> wie stark wird neuer Wert über alten gewertet: 1 bedeutet immer neue, 0 immer alte

Q Wert steigt immer kleiner -> wenn richtig -1

Bricht immer erst nach 200 ab, außer ziel wurde gefunden

* Sobald ziel gefunden wird, geht es für diesen Punkt nach oben, umso früher, umso schneller geht es hoch

Fragen:

Was ist Unterschied zwischen discount und learning?

Lerning rate:wie viel ist neue erkenntnis wert,discount: belohnung für schnelles finden

Discount legt fest, wie wertvoll neues Wissen ist im Gegensatz zu altem-> sorgt dafür das es terminiert, indem es nicht unendlich hoch steigt

Geht der q Wert bei mehr epochen kleiner als -10? Nein. Nicht bei 10000

Warum knickt es ab? Wenn Max zufällig (GREEDY ALGORITHMUS) „verbaut“, steigt der Wert des Weges wieder an, da jetzt diese Aktion nicht mehr unbedingt mehr Maximum bringt in dem Schritt

Warum ist der Wert nie kleiner als -10? ->

Antwort: Berechnung Q Wert (Hier aktueller Wert gleich max, zum Verdeutlichen des Prinzips):

Für Wert -1:-> -1.18 -> -0.18

Für Wert -5 -> -5.1 -> -0.1 learningrate=0.2,discount=0.9

Für -9 ->-9.02 -> -0.02

Für -10 -> -10 -> -0

Lange gerade, weil erst das erste finden einen Unterschied macht, nicht wenn man „nah“ dran ist,

Wert steigt für alle relativ gleichmäßig, da zunächst alle gleichen Reward haben und auch relativ gleich initialisiert

Wie findet man es schneller (kürzere Gerade)?

Anderer Ansatz für Berechnung von Epsilon? Damit schnelleres Erreichen des Ziels ?

Statt Epsilon sarsa?

* Versuchsdurchläufe

Unterschied von Q Learning Algorithmus bei Frochte gegenüber verwendet:

ˆ*Q* (*s*,*a*) = *r* +*γ*·max*a*′ˆ*Q*(*s*′,*a*′) vs. Q’(s1, s2, a) = (1 — w)\*Q(s1, s2, a) + w\*(r+d\*Q(s1’, s2’, argmax a’ Q(s1’, s2’, a’)))

Keine Learningrate

Seperates setzen des Q Werts bei finden auf reward, wahrscheinlich weil er sich ja nie ändert

Reward für einen Schritt ohne es zu finden ist -1. Deswegen ändert es sich auch nicht. Wenn es gefunden wurde bricht es ab-> ein letztes Setzen auf -1, aber für restliche Schritte 0. -> besserer reward. Man müsste q Wert nicht auf 1 setzen, da der neue ja kleiner ist und die großen gegen -10 konvergieren, aber so hat ein Erfolg viel stärkeren Einfluss und kann sich auch leichter gegen andere Rewards behaupten.

Ergebnisse Parameteranalyse:

Meisten Parameter waren schon vorher Best mögliche Ergebnisse, nur Epsilon zu verändern hat etwas geändert: Ergebnisse sind schon früher besser, aber dafür ist der Graph mit größeren Schwankungen (siehe passender Graph)

Mögliche Erklärung: Noch nicht ganz explored-> Bei zufällig nicht erkundetem kann noch kein Weg gefunden werden.

Warum wird früher eins gefunden, wenn davor es doch random ist?

Die Werte werden gleichmäßig größer, deswegen sind die Werte außen immer besser -> man versucht immer nach außen zu kommen, durch Schwung erreicht man noch mehr neue werte, so wird man schnell immer höher und erreicht das ziel

Wenn noch nicht alles erkundet und man auf noch nicht besuchte werte stößt, werden die erst einmal genommen, weil sie erstmal kleiner sind-> erst wenn diese oft genug erkundet wurden oder man zufällig wieder auf den besten weg kommt wird das ziel wieder häufiger erreicht, außerdem bekommt richtiger weg erstmal einen schlechteren Wert