UNIVERSITAT DE BARCELONA

Pràctica 3: Aprenentatge per reforç

Alejandro Guzman

20 novembre 2022

ÍNDEX

1 Introducció **3**
1 Aprenentatge per reforç . . . 3
2 Q-Learning . . . 3

2 Implementació **5**
1 Exercici 1 . . . 5
1.1 Taula Q-Learning . . . 5
1.2 Epsilon Greedy . . . 5
1.3 Convergència . . . 5
1.4 Proves . . . 6
2 Exercici 2 . . . 6
2.1 Qüestions . . . 6

3 Conclusions **8**

1 INTRODUCCIÓ

L'objectiu principal de la darrera pràctica de l'assignatura Intel·ligència Artificial del grau d'Enginyeria Informàtica impartit en la Universitat de Barcelona és implementar un nou algorisme pertanyent a la família d'algorismes d'entrenament per reforç. Aquest és més sofisticat per a resoldre els problemes plantejats en les altres dues pràcticas prèviament realitzades.

Primer describim què és l'entrenament per reforç i l'algorisme que implementarem en la pràctica de forma breu i clara amb l'objectiu de contextualitzar els objectius d'aquest treball. També servirà per a relacionar els continguts teòrics de l'assignatura amb la part pràctica d'aquesta mateixa.

1 Aprenentatge per reforç

Les dues grans àrees d'aprenentatge tradicional del Machine Learning són l'aprenentatge supervisat i l'aprenentatge no supervisat. Sembla difícil que aquí hi hagués espai per a altres opcions; tanmateix sí que n'hi ha i és l'aprenentatge per reforç. En aprenentatge per reforç no tenim una "etiqueta de sortida", per la qual cosa no és de tipus supervisat i si bé aquests algorismes aprenen per si mateixos, tampoc no són de tipus no supervisat, on s'intenta classificar grups tenint en compte alguna distància entre mostres.

El Reinforcement Learning llavors es troba en un punt intermig, on intentarà fer aprendre la màquina basant-se en un esquema de recompenses en un entorn on cal prendre accions i que està afectat per múltiples variables que canvien amb el temps. Dintre del RL nosaltres ens centrarem en un algorisme peculiar conegut com Q-Learning.

2 Q-Learning

Q-Learning és un dels models més utilitzat dintre del RL. Els elements que s'involucren en el desenvolupament del model són (d'igual manera que amb el RL)

- Polítiques: És una taula (encara que pot tenir n-dimensions) que indicarà al model "com actuar" en cada estat.
- Accions: les diverses eleccions que pot fer l'agent a cada estat.
- Recompenses: si sumem o restem puntuació amb l'acció presa.
- Comportament greedy de l'agent. És a dir, si es deixarà portar per grans recompenses immediates, o anirà explorant i valorant les riqueses a llarg termini.

L'objectiu principal en entrenar el nostre model a través de les simulacions serà anar "emplenant" la taula de Polítiques de manera que les decisions que vagi prenent el nostre agent obtinguin "la recompensa més gran" alhora que avancem i no ens quedem estancats, és a dir , podent complir l'objectiu global (o final) que volem assolir.

A la política l'anomenarem Q pel que $Q(state, action)$ ens indicarà el valor de la política per a un estat i una acció determinats. I per saber com anar completant la taula de polítiques ens valem de l'equació de Bellman expressada com

$$\hat{Q}(s,a) = Q(s,a) + \alpha \cdot [R + (\lambda \cdot max(Q(s',)) - Q(s,a)] \tag{1.1}$$

on $Q(s,a)$ és el valor actual, α és el rati d'aprenentatge, R és la recompensa, λ és la taxa de descompte i $maxQ(s',)$ és el valor òptim esperat. El que explica aquesta ecuació és com anar

actualitzant les polítiques $\hat{Q}(s,a)$ en base al valor actual més una futura recompensa que rebrem, en cas de prendre dita acció. El rati d'aprenentatge regula la velocitat en la que s'aprèn i la taxa de descompte tindrà en compte la recompensa en curt o llarg termini.

Un cop resumit els conceptes teòrics aplicats en l'actual pràctica, implememntarem els dos exercicis i documentarem la feina realitzada separant-los en dos apartats ben diferenciats dintre de l'informe, informant de l'implementació, possibles errors i resposta a les preguntes realitzades en l'enunciat de la pràctica.

2 IMPLEMENTACIÓ

1 Exercici 1

Els algorismes RL poden funcionar tant en entorns estocàstics com deterministes. Aquesta primera part de la pràctica consisteix a implementar un algorisme Q-Leaning que aprengui a resoldre el mateix problema proposat a la pràctica 1, on només es poden moure les peces blanques. Això és un exemple del cas més simple dels entorns deterministes. L'objectiu llavors és implementar un algorisme Q-Learning, actualitzat mitjançant diferència temporal, que troba el camí cap al checkmate en el menor nombre de moviments possible. Utilitzarem també una taula per emmagatzemar els Q-values corresponents.

1.1 Taula Q-Learning

Un dels elements més importants, per no dir el que més, és la taula on s'emmagatzemaran per a cada estat la recompensa i els seus respectius estats futurs. La taula s'anirà actualitzant a cada iteració amb la idea de que convergeixi en algun punt per tal de verificar que l'algorisme ha aprés.

A l'hora de programar, interpretarem aquesta taula com un diccionari. Per tant crearem un diccionari per al Q-Learning on cada estat tindrà un diccionari dels estats futurs als quals pot arribar amb la seva puntuació, la recompensa del propi estat i la key, que serà el mateix CurrentState. Per a cada nou estat es cridarà a una funció encarregada d'afegir una nova entrada a la taula de Q-Learning que representi aquest nou estat i, per a la següent iteració on es trobi aquest estat, l'entrada ja estarà creada i només s'haurà d'actualitzar el valor.

1.2 Epsilon Greedy

Per tal de realitzar l'exploració dels diferents estats ens basarem en l'exploració voraç al voltant d'un valor epsilon. Aquest valor l'inicialitzem en el codi amb $\epsilon = 0.9$ i significa que el 10% dels casos on ens trobem en un nou estat de l'actual iteració, agafarem la millor opció que ens dona el màxim Q-Value, en canvi el 90% dels casos explorarem un nou estat que no hagi estat explorat. D'aquesta manera ens assegurem que explotem els valors òptims de la taula de Q-Values sense oblidar-nos d'explorar nous estats que poden tenir bons valors també.

En acabar les iteracions, degut al gran nombre d'iteracions realitzades durant le diferents execucions, ens donem compte que és pràcticament impossible haber deixat algun estat important durant l'exploració degut a la altra probabilitat d'explorar enlloc d'explotar. Si l'epsilon tingués un valor més petit la probabilitat de deixar algun estat important sense explorar augmentaria.

1.3 Convergència

Una dels aspectes més importants en l'execució del codi es saber quan parar i quan s'ha de continuar aprenent. Això ho valorarem en funció de la convergència de les actualitzacions dels Q-Values. Ho hem implementat de forma que, proposant un marge de 0.15, si obtenim en una iteració un error dintre del rang (-0.15, 0.15), llavors aquella iteració estarà convergint a la possible solució del problema.

L'error el calculem de forma molt simple. En cada iteració l'error és el sumatori de la diferència entre el sample i el qValue de cada estat explorat.

```
sample = reward + gamma*self.maxQValue(nextString, self.
    qTable)
#Anem sumant la diferencia entre la reward real i la predita.
#Si es prou petit, vol dir que estem convergint
error += sample - qValue
```

I finalment calculem la mitjana de l'error segons el nombre de moviments fins al checkmate per tal de normalitzar el valor.

```
#Calculem la mitjana de la delta
if numMovimentsCheckMate != 0:
    mitjanaDelta = delta/numMovimentsCheckMate
#Si esta en l'interval (-error, error), vol dir que aquest
    camí ha convergit.
if mitjanaDelta < error and mitjanaDelta > -error:
    numConvergingPaths += 1
#Si no, reiniciem el comptador.
else:
    numConvergingPaths = 0
```

Per tant si arribem a 10 camins convergents seguits, suposem que la solució convergeix i retornem la solució.

1.4 Proves

En les diferents proves que hem realitzat, s'arriba a la convergència amb un nombre d'iteracions considerables, potser sigui degut al marge de l'error per considerar o no una iteració com a convergent. Igualment l'important és assegurar-se que el camí realment convergeix assegurant-nos que és la solució òptima al problema.

A continuació hem realitzat una prova indicant cada 500 iteracions quin error es troba actualment. Com es pot comprovar en l'output de l'execució mostrat a continuació (una part, sense mostrar el camí òptim en el taulell), s'arriba a la solució òptima en un total de 22150 iteracions. Depenent de l'execució aquest nombre disminueix o augmenta, però podem comprovar uqe ha trobat el camí òptim.

```
Path sequence:  [[[7,0,2], [7,4,6]], [[6,3,6], [7,0,2]],
    [[5,4,6], [7,0,2]], [[5,4,6], [0,0,2]], [[4, 3,6],
    [0,0,2]], [[3,3,6], [0,0,2]], [[2,4,6], [0,0,2]]]
Total of iterations: 22155
```

2 Exercici 2

Utilitzant el codi de la pràctica 2 com a punt de partida, programarem dos agents RL (blancs i negres), i fer que aprenguin a competir entre ells fins a fer escac i mat. Utilitzarem Q-Learning en qualsevol dels dos agents. El codi del segon exercici és molt similar al del primer, únicament que ara són dos agents i ens trobem en un escenari dinàmic, on les recompenses estan en constant canvi degut a que les peces contràries es troben en constant moviment.

2.1 Qüestions

a. Quines diferències podeu descriure del comportament d'aquests agents respecte al del punt 1?

En el punt 1 teniem un escenari estàtic, on el món en el que es trobava l'agent era invariable, és a dir, les recompenses que otorgaven els diversos estats no cambiaven segons factors condicionats. En cambi en el punt 2 ens trobem a dos agents que juguen en el mateix món, per lo tant les decisions d'un agent afecten al món que envolta a l'altre agent i viceversa.

Per exemple, la recompensa de moure una peça blanca d'un lloc a un altre no serà la mateixa en un torn que en el següent, degut a que afectarà la nova posició de les negres que prendràn en el torn intermig entre aquests dos (de les blanques), per lo tant és molt més impredictible aquest món dinàmic on hi participen dos agents, els quals tenen el mateix objectiu i, a la vegada, aquest objectiu perjudica a l'altre (o guanya un, o un altre, o empaten).

b. Quant de temps es triga a aprendre?

Depenent de l'error de convergència, de l'heurística utilitzada i de com estigui implementat el codi. Normalment triga un temps major al punt 1 degut a que, com hem comentat anteriorment, ens trobem en un escenari que incorpora un món dinàmic que es troba en constant moviment. Això implica que a cada iteració les recompenses poden variar segons el recorregut dels dos agents i la proporció de terreny que han explorat.

c. Què passa si varieu les seves taxes d'aprenentatge relatives?

Depenent de quina taxa d'aprenentatge relativa es variï, cambiarà el nombre d'iteracions o la convergència de la solució òptima.

Si variem el valor de l'epsilon augmentant-lo (sempre ha de trobar-se entre 0 i 1) llavors es donarà més importància a l'exploració que no a l'explotació, per tant a cada iteració els agents exploraràn més estats assegurant que descobreixen el major nombre d'aquests. En canvi si es disminueix el seu nombre es donarà més importància a l'explotació, per tant a cada iteració els agents es centraràn més en els bons Q-Values ja obtinguts que no a explorar nous estats en busca de valors encara millors.

Si variem la gamma es varia la taxa d'aprenentatge amb la qual l'algorisme aprèn, el que vol dir que s'arribarà en més o en menys iteracions a la solució òptima, ja que els Q-Values variaràn més ràpidament si la gamma és elevada. Però en aquest cas passa com en l'algorisme del descens estocàstic del gradient o qualsevol algorisme d'optimització, si inicialitzem el valor de la taxa d'aprenentatge molt elevat, el canvi serà lo suficientment gran entre una iteració i una altra que no convergirà, en canvi si és molt petit, es més fàcil la seva convergència però es necessitaràn moltes iteracions per a que succeeixi, per tant s'ha de trobar un valor equilibrat d'aquesta taxa d'aprenentatge.

3 CONCLUSIONS

La pràctica s'ha sentit com un desenllaç del treball pràctic de tota l'assignatura, millorant la resolució dels problemes plantejats en les dues primeres pràctiques utilitzant algorismes més sofisticats.

Per la part de la dedicació en hores, ha sigut una pràctica molt llarga on la majoria del temps ha sigut per poder implementar correctament el pseudocodi amb el codi que ofereix el professorat, de forma que primer s'ha d'analitzar el codi per posteriorment implementar els mètodes a sobre d'aquest mateix codi.

El resultat obtingut a sigut satisfactori, ja que s'ha pogut adquirir la majoria dels objectius que oferia la pràctica, tot i que ha sigut difícil sobretot implementar el segon exercici.

Personalment, també potser perquè he fet la pràctica sol, al principi he trobat diversos problemes que m'han costat continuar amb el desenvolupament del primer exercici, però després de diversos intents he aconseguit solucionar-ho. Tot i que el segon exercici és d'una complexitat més elevada i també he trobat problemes.

Finalment, ha sigut una pràctica interessant que ha servit com a introducció a lo que realment significa intel·ligència artificial, utilitzant algorismes d'aprenentatge per reforç, a part de continuar aprenent programació en Python.