UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA

Project 3: Reinforcement Learning

Inteligenta Artificiala

Autori: Pinciuc Darius
Grupa: 30236

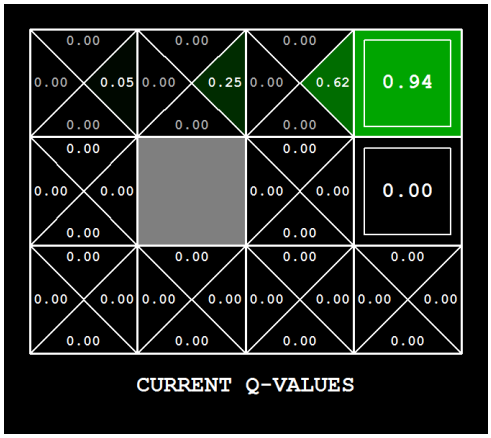
Facultatea de Automatica
si Calculatoare

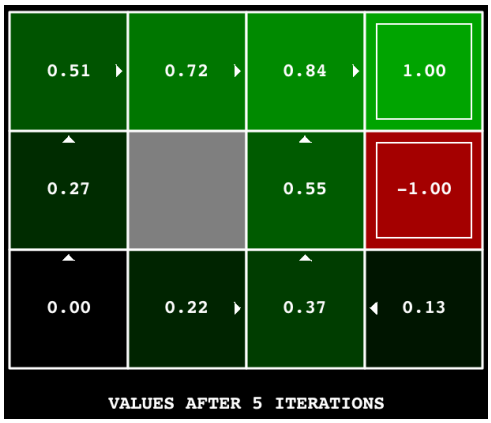
17 Ianuarie 2023

Cuprins

1 Introducere

În cadrul acestui proiect, am implementat Value Iteration, Q-Learning și Policies.





2 Value Iteration

Ecuția de actualizare a stării:

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} T(s, a, s') \left[R(s, a, s') + \gamma V_k(s')\right]$$

Value Iteration este o tehnică utilizată în învățarea prin recompensă pentru a găsi politici optime în cadrul unui MDP. Aceasta se bazează pe iterații succesive asupra valorilor asociate stărilor pentru a găsi valorile optime și apoi a deduce politicile corespunzătoare.

Această ecuație exprimă actualizarea valorilor stărilor $V_{k+1}(s)$, în funcție de valorile stărilor succesoare $V_k(s')$. Factorul γ reprezintă rata de reducere, iar (s, a, s') sunt tranzițiile dintre stări.

Metoda runValueIteration(self) Implementează algoritmul de Iterație de Valori. Actualizează valorile stărilor în funcție de ecuația de actualizare a valorilor din algoritmul Value Iteration, pe parcursul numărului de iterații specificat.

Metoda getValue(self, state) Returnează valoarea asociată unei stări specificate. Această valoare este calculată în cadrul constructorului, folosind algoritmul de Iterație de Valori.

Metoda computeQValueFromValues(self, state, action) Calculează valoarea Q pentru o pereche de stare-acțiune dată, bazată pe valorile stărilor calculate în algoritmul de Iterație de Valori.

Metoda computeActionFromValues(self, state) Calculează acțiunea optimă pentru o stare dată, bazată pe valorile stărilor calculate în algoritmul de Iterație de Valori. În cazul de egalitate, se poate alege oricare acțiune.

Metoda getPolicy(self, state) Returnează politica (acțiunea optimă) pentru o stare dată.

Metoda getAction(self, state) Returnează acțiunea optimă pentru o stare dată. Nu se face explorare în acest caz.

Metoda getQValue(self, state, action) Returnează valoarea Q pentru o pereche de stare-acțiune dată. Este o interfață simplificată pentru a accesa metoda computeQValueFromValues.

3 Policies

Termenul "Policies" se referă la strategii sau seturi de reguli pe care un agent le urmează pentru a lua decizii într-un Proces Decizional Markovian (MDP). factorul de reducere - answerDiscount; recompensa - answerLivingReward; zgomot - answerNoise;

3a: Are un factor de reducere scăzut (0.1), niciun zgomot (0.0) și o recompensă de -1.0 . Factorul de reducere scăzut sugerează o concentrare asupra recompenselor imediate, iar absența zgomotului indică acțiuni deterministe. Recompensa negativă încurajează agentul să își termine sarcina rapid.

3b: Are un factor de reducere scăzut (0.1), un nivel moderat de zgomot (0.1) și o recompensă de -1.0 . Factorul de reducere scăzut sugerează o concentrare asupra recompenselor imediate, iar nivelul moderat de zgomot introduce o anumită incertitudine în acțiuni. Recompensa negativă încurajează agentul să își termine sarcina rapid.

3c: Factorul de reducere este mai mare (0.9), zgomotul este la un nivel moderat (0.1), iar recompensa este de -1.0 . Factorul de reducere mai mare sugerează că agentul se concentrează asupra recompenselor viitoare, iar nivelul moderat de zgomot introduce o anumită incertitudine în acțiuni.

3d: Are un factor de reducere mai mare (0.9), un nivel mai ridicat de zgomot (0,2) și o recompensă de -1.0 . Factorul de reducere mai mare sugerează că agentul se concentrează asupra recompenselor viitoare, iar nivelul mai ridicat de zgomot adaugă mai multă incertitudine în acțiuni.

3e: Are un factor de reducere scăzut (0.1), un nivel mai ridicat de zgomot (0.2) și o recompensă de -1.0. Factorul de reducere scăzut sugerează o concentrare asupra recompenselor imediate, iar nivelul mai ridicat de zgomot adaugă mai multă incertitudine în acțiuni.

4 Q-Learning

getQValue(self, state, action) Returnează valoarea Q asociată perechii de stare-acțiune specificate. Dacă această pereche nu a fost întâlnită în trecut, returnează 0.0.

computeValueFromQValues(self, state) Returnează valoarea maximă a funcției Q pentru starea dată, unde maximul este luat peste acțiunile legale. În cazul în care nu există acțiuni legale (starea terminală), se returnează 0.0.

computeActionFromQValues(self, state) Calculează cea mai bună acțiune de luat într-o anumită stare, ținând cont de valorile Q. În caz de egalitate, alege aleator între acțiunile la fel de bune.

getAction(self, state) Determină acțiunea pe care agentul o va lua în starea curentă. Cu o probabilitate dată de epsilon, alege o acțiune aleatoare (explorare), altfel, alege acțiunea cu cea mai mare valoare Q (exploatare).

update(self, state, action, nextState, reward) Actualizează valoarea Q pentru perechea de stare-acțiune dată, bazată pe tranziția observată. Folosește formula Q-learning pentru a actualiza valoarea Q în funcție de recompensa observată și valoarea Q a stării următoare.