Învățare Automată - Laboratorul 5
Q-Learning

Tudor Berariu
tudor.berariu@gmail.com
Laboratorul AIMAS
Facultatea de Automatică și Calculatoare
Universitatea Politehnica București

21 martie 2016

1 Scopul laboratorului

Scopul laboratorului îl reprezintă înțelegerea și implementarea algoritmului Q-Learning.

2 Algoritmul Q-Learning

Algoritmul 1 Algoritmul *Q-Learning*

pentru toate stările s:
  pentru toate acțiunile a valide în s:
    Q(s,a) $\longleftarrow$ 0

pentru toate episoadele:
  $s \longleftarrow$ stare inițială
  cât timp s este stare finală:
    $a \longleftarrow \epsilon$-greedy$(Q, s)$
    execută a
    observă s', r
    $Q(s,a) \longleftarrow Q(s,a) + \alpha\Big(r + \gamma \max_{a'} Q(s', a') - Q(s,a)\Big)$
    $s \longleftarrow s'$

3 Greuceanu și Balaurul

Pe o hartă bidimensională se înfruntă Greuceanu și-un balaur. Greuceanu trebuie să găsească mărul fermecat înainte de a fi prins de balaur și înainte ca balaurul să calce pe măr. Balaurul simte direcția în care se află Greuceanu și se îndreaptă către el.

Concret, Greuceanu câștigă jocul și 10 puncte dacă ajunge primul la mărul fermecat. Greuceanu pierde jocul dacă este prins de balaur sau dacă balaurul calcă pe măr. De-asemenea, la fiecare moment de timp Greuceanu pierde câte 0.1 puncte. Dacă ajunge la -20 de puncte, Greuceanu pierde jocul.

3.1 Funcțiile care abstractizează jocul

- `get_initial_state(input_file)` primește calea unui fișier ce conține harta jocului și întoarce starea inițială a jocului (un șir de caractere).
- `get_valid_actions(state)` primește o stare și întoarce o listă de acțiuni ce pot fi executate în acea stare
- `apply_action(state, action)` primește o stare și o acțiune și întoarce starea în care se ajunge prin aplicarea acțiunii, recompensa primită și un șir de caractere cu informații despre desfășurarea jocului.
- `is_final_state(state, score)` primește starea și scorul curente și răspunde dacă jocul s-a terminat.
- `display_state(state)` afișează starea curentă.

4 Cerințe

[**6p**] Implementați algoritmul *Q-learning* (completați funcția `q_learning`).

[**2p**] Implementați strategia ϵ-greedy de selecție a unei acțiuni. Funcția primește toate acțiunile valide dintr-o stare dată. Atât timp cât există acțiuni ce nu au fost explorate, se va alege aleator una dintre acestea. Altfel, cu o probabilitate ϵ se va alege o acțiune aleatoare, iar cu o probabilitate $1 - \epsilon$ se va alege cea mai bună acțiune din starea respectivă.

[**2p**] Implementați rutina de evaluare a politicii *lacome* (care alege întotdeauna cea mai bună acțiune).

[**2p**] Găsiți metaparametrii potriviți pentru o învățare cât mai rapidă pe toate cele trei hărți (rata de învățare, valoarea lui ϵ). Încercați să modificați ϵ pe parcursul învățării.