Învățare Automata: **Tema 2**
Problema lifturilor

Tudor Berariu & Mihai Trăscău
Laboratorul AI-MAS
Facultatea de Automatică și Calculatoare

22 martie 2012

1 Motivație

Scopul acestei teme îl reprezintă familiarizarea cu un alt algoritm de învățare prin recompensă - SARSA - și implementarea acestuia pentru rezolvarea unei probleme reale. Secțiunea 2 descrie problema care trebuie modelată și rezolvată folosind învățarea prin recompensă. Problema în discuție este una intens studiată în literatură [2, 5]. În secțiunea 3 este prezentat algoritmul SARSA care trebuie implementat pentru rezolvarea temei. Citiți aceaste două secțiuni cu atenție (eventual și materiale suplimentare) înainte de a trece la rezolvarea cerințelor (secțiunea 4).

Succes!

2 Problema lifturilor

Se consideră următorul scenariu. O clădire cu 5 niveluri (numărând aici și parterul) este dotată cu 2 lifturi cu o capacitate de 10 persoane fiecare. La fiecare etaj se găsește un panou cu două butoane de unde o persoană poate chema liftul anunțându-și intenția de a urca sau de a coborî. În interiorul liftului există butoane ce corespund tuturor etajelor. Odată ajuns în cabina unuia dintre cele două lifturi, fiecare pasager apasă butonul corespunzător etajului la care dorește să ajungă.

Fiecare lift parcurge distanța dintre două etaje într-o unitate de timp. Deschiderea ușilor, urcarea și coborârea pasagerilor, urmate de închiderea ușilor se petrec în 2 unități de timp.

Vom numi *timp de întârziere* timpul scurs între apariția unei persoane (care coincide cu anunțarea intenției de a coborî sau urca) și coborârea acestuia din cabina liftului la destinație din care se scade perioada minimă de timp necesară parcurgerii directe (fără opriri sau schimbări de direcție ale liftului) a distanței dintre etajul inițial și etajul destinație.

$$TimpIntarziere = T_{destinatie} - T_{aparitie} - |Etaj_{aparitie} - Etaj_{destinatie}| \quad (1)$$

Spunem că lifturile funcționează mai eficient dacă timpul de întârziere total pe o perioadă mai lungă este mai mic.

3 Algoritmul SARSA

Algoritmul SARSA (State-Action-Reward-State-Action) [3, 4] este similar algoritmului Q-learning (făcut la curs și la laborator).

Algoritmul 1 Algoritmul SARSA

pentru toate stările s **execută**
  pentru toate acțiunile a **execută**
    Q(s,a) $\longleftarrow 0$
  termină ciclu
termină ciclu
pentru toate episoadele **execută**
  $s \longleftarrow$ stare inițială
  $a \longleftarrow \epsilon$-greedy$(Q, s)$
  repetă
    execută a
    observă s', r
    $a' \longleftarrow \epsilon$-greedy$(Q, s')$
    $Q(s,a) \longleftarrow Q(s,a) + \alpha\Big(r + \gamma Q(s', a') - Q(s,a)\Big)$
    $s \longleftarrow s'$
    $a \longleftarrow a'$
  până când s este stare finală
termină ciclu

Regula de actualizare a funcției Q este diferită față de cea a algoritmului Q-Learning (vezi formula 2) prin faptul că pentru *recompensele viitoare* nu se alege întotdeauna valoarea maximă pentru $Q(s', *)$, ci se alege o acțiune a' conform politicii (care include și componenta de explorare) și se utilizează valoarea $Q(s', a')$. Din puncte de vedere conceptual, această diferență se traduce prin faptul că SARSA este un algoritm *on-policy* spre deosebire de Q-Learning care este un algoritm *off-policy*.

Formula de actualizare a funcției Q în algoritmul SARSA:

$$Q_{t+1}(s_t, a_t) \longleftarrow Q_t(s_t, a_t) + \alpha_t\Big(R(s_t, a_t) + \gamma Q_t(s_{t+1}, a_{t+1}) - Q_t(s_t, a_t)\Big) \quad (2)$$

În cazul unui proces de învățare continuu este de așteptat ca algoritmii on-policy să se comporte mai bine decât algoritmii off-policy întrucât politica

învățată ține cont și de componenta de explorare. Acest lucru este util mai ales atunci când mediul este unul dinamic și nu se poate renunța la explorare.

Avantajul algoritmului SARSA față de Q-Learning este foarte bine surprinsă în următorul paragraf din [1]: [...] *perhaps it would be better to learn the policy that is optimal, given that you will explore ϵ of the time. It would be like a person who, when walking, always takes a random step every 100 paces or so. Such a person would avoid walking along the top of a cliff, even when that is the optimal policy for a person who doesn't explore randomly.*

Pseudocodul SARSA este prezentat în Algoritmul 1. Pentru alegerea acțiunii dintr-o stare s folosiți tehnica ϵ-greedy, descrisă de Algoritmul 2.

Algoritmul 2 Algoritmul ϵ-greedy

Intrări: Q, starea s,ϵ, A
Ieșire: acțiunea a

1: $A' \longleftarrow \{a \in A | a$ acțiune validă în $s\}$
2: $p \longleftarrow random(0, 1)$
3: **dacă** $p < \epsilon$ **atunci**
4: $\quad a \longleftarrow random(A')$
5: **altfel**
6: $\quad a \longleftarrow \underset{act \in A'}{\text{argmax}}\ Q(s, act)$
7: **termină dacă**

4 Cerințe

Pentru a rezolva tema trebuie să tratați următoarele cerințe:

1. Construiți un simulator al problemei lifturilor. Se definește un scenariu ca fiind o listă de tupluri: $\langle timp\ apariție, etaj\ inițial, etaj\ destinație \rangle$, unde fiecare tuplu reprezintă un pasager. Generați scenarii care să acopere câteva sute/mii de unități de timp.

2. Folosiți algoritmul SARSA (descris în secțiunea 3) pentru a determina o politică cât mai eficientă pentru cele două lifturi. Reprezentarea stărilor, acțiunile posibile și schema de recompensare sunt la alegerea dumneavoastră. Ceea ce se dorește este scăderea timpului de întârziere al tuturor pasagerilor.

3. Faceți un grafic al evoluției timpului de întârziere total într-o fereastră de 50 de unități de timp pentru întreaga perioadă simulată. Un punct al graficului reprezintă valoarea medie a timpului de întârziere din 50 de unități de timp (primul punct: 1-50, al doilea punct: 2-51, etc.)

Pentru punctaj bonus tratați acest caz:

1. Există o cameră video la fiecare etaj în fața ușilor liftului care (împreună cu alți senzori) poate da pentru fiecare unitate de timp următoarele informații:

- câte persoane așteaptă pentru a urca,
- câte persoane așteaptă pentru a coborî la etajul respectiv.

5 Indicații și notare

În arhiva temei includeți:

- toate fișierele sursă (eventual cu Makefile / script de compilare și rulare)
- fișier pdf cu descrierea detaliilor de implementare: cum sunt reprezentate stările, care sunt acțunile, care este sistemul de recompensare, grafic (grafice dacă ați încercat mai multe variante) cu timpul de întârziere, etc.

Punctajul se va acorda conform tabelei 1.

Tip	Descriere	Punctaj	
		Implementare	Explicații
Obligatoriu	1. Simulator	2p	-
	2. Algoritmul SARSA	5p	2p
	3. Evoluția timpului de întârziere	0.5p	0.5p
Bonus	1. Camere	1.5p	0.5p
Total		12p	

Tabela 1: Împărțirea punctelor între task-uri

Bibliografie

[1] L.C. Baird III. *Reinforcement learning through gradient descent*. PhD thesis, Citeseer, 1999.

[2] R. Crites and A. Barto. Improving elevator performance using reinforcement learning. In *Advances in Neural Information Processing Systems 8*. Citeseer, 1996.

[3] G.A. Rummery and M. Niranjan. *On-line Q-learning using connectionist systems*. University of Cambridge, Department of Engineering, 1994.

[4] S.P. Singh and R.S. Sutton. Reinforcement learning with replacing eligibility traces. *Recent Advances in Reinforcement Learning*, pages 123–158, 1996.

[5] X. Yuan, L. Busoniu, and R. Babuška. Reinforcement learning for elevator control. In *Proceedings 17th IFAC World Congress (IFAC-08)*, pages 2212–2217, 2008.