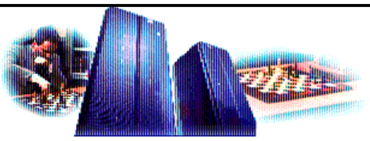
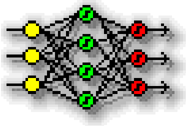
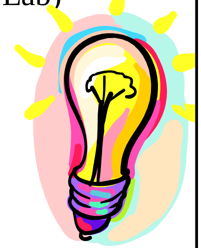




Sistemi Intelligenti Reinforcement Learning: Eligibility traces - Examples

Alberto Borghese

Università degli Studi di Milano
Laboratorio di Sistemi Intelligenti Applicati (AIS-Lab)
Dipartimento di Informatica
borghese@di.unimi.it

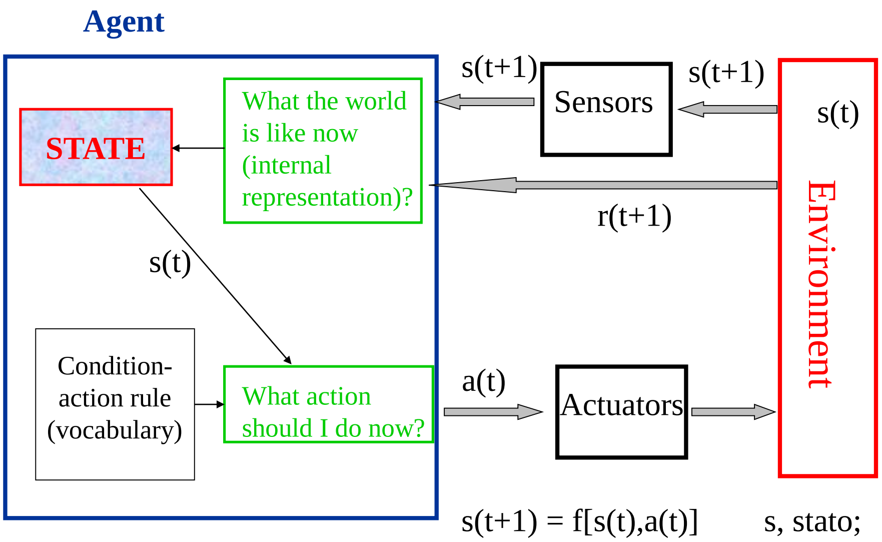


Sommario

Traccia

Esempi

Schematic diagram of an agent



Agent
STATE
What the world is like now (internal representation)?
s(t)
Condition-action rule (vocabulary)
What action should I do now?
s(t+1)
Sensors
s(t+1)
s(t)
r(t+1)
a(t)
Actuators
Environment

s, stato; a = uscita

s(t+1) = f[s(t),a(t)]
a(t) = g[s(t)]

s, stato;
a, ingresso

Formulazione di TD(0)

Correggo la stima corrente valutando l'"errore" ad un passo.

$$Q^{\pi}{}_{k+1}(s_t,a_t)=Q^{\pi}{}_{k}(s_t,a_t)+ \alpha\left[r_{t+1}+\gamma Q^{\pi}{}_{k}(s_{t+1},a_{t+1})-Q^{\pi}{}_{k}(s_t,a_t)\right]$$

$$\Delta Q^{\pi}(s_t,a_t)=+\alpha\,\delta_k$$

$$\delta_k=\left[r_{t+1}+\gamma Q^{\pi}{}_{k}(s_{t+1},a_{t+1})-Q^{\pi}{}_{k}(s_t,a_t)\right]$$

Come estendere l'orizzonte temporale dell'apprendimento?

Cosa rappresenta la Eligibility trace

Buffer di memoria: contiene traccia di eventi passati (stati visitati, azioni…); la traccia evapora nel tempo.

Quando viene calcolato un errore usando metodi basati su TD, la eligibility trace suggerisce quali variabili aggiornare (credit assignement).

Amplia l'orizzonte temporale sul quale fare l'aggiornamento a più di 1 passo.

Definisce se uno stato è eleggibile e "quanto" sia eleggibile, cioè che percentuale di aggiornamento meriti.

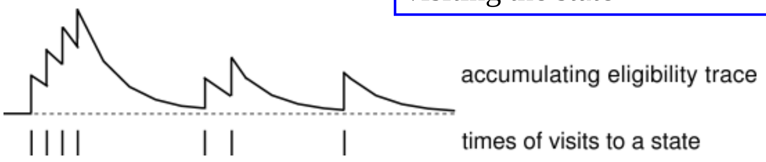
NB L'errore modifica il valore di Q(s,a) ma questo modifica a sua volta il valore di $Q(s_{t-1}, a_{t-1})$....

Eligibility trace per la funzione Q

$$e_t(s,a) = \begin{cases} \gamma\lambda e_{t-1}(s,a) + 1 & \text{if } s = s_t \text{ and } a = a_t; \\ \gamma\lambda e_{t-1}(s,a) & \text{otherwise.} \end{cases} \qquad \text{for all } s, a$$

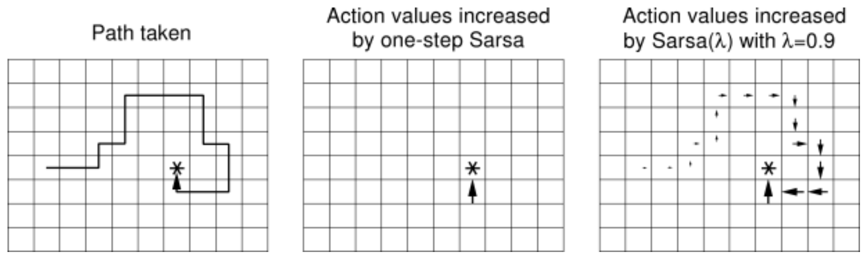
decay

Increases: depends only on visiting the state



e(s,a) = 0 at start, e(s,a) ≥ 0.

Esempio


Path taken
Action values increased by one-step Sarsa
Action values increased by Sarsa(λ) with λ=0.9

Con il semplice costo di una variabile per ogni coppia stato-azione, ho un aggiornamento graduale della funzione valore di più stati.

$Q^\pi(s,a)$ inizializzati ad un valore leggermente negativo.
r = 0 per ogni stato prossimo, tranne lo stato finale, per il quale r = +1.

Osservazioni

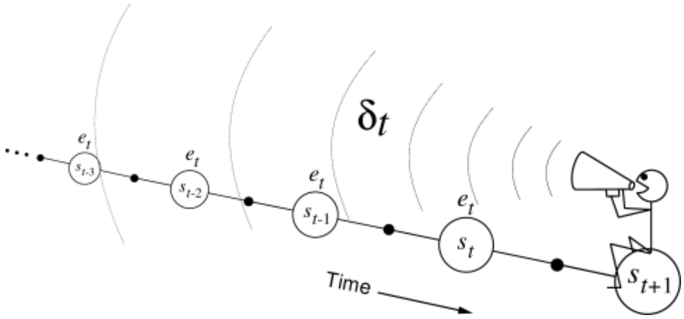


Figure 7.8: The backward or mechanistic view. Each update depends on the current TD error combined with traces of past events.

Earlier states are given less credit for the TD(0) error

Come utilizzare la eligibility trace

TD(0) Learning:

$$Q^{\pi}{}_{k+1}(s_t,a_t) = Q^{\pi}{}_{k}(s_t,a_t) + \alpha\left[r_{t+1} + \gamma Q^{\pi}{}_{k}\left(s_{t+1},a_{t+1}\right) - Q^{\pi}{}_{k}\left(s_t,a_t\right)\right]$$

Errore: δ_t

$$Q^{\pi}{}_{k+1}(s_t,a_t) = Q^{\pi}{}_{k}(s_t,a_t) + \alpha\delta_t \cdot 1$$

Per 1 coppia (s,a)

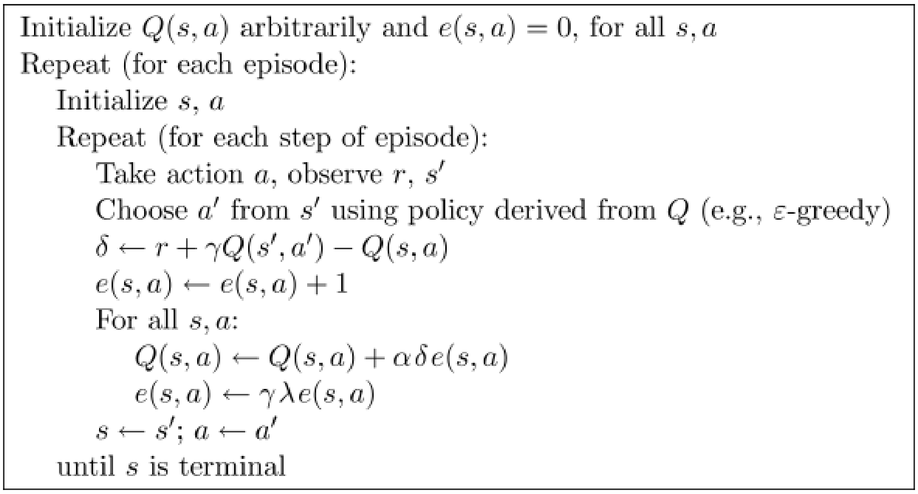
$$Q^{\pi}{}_{k+1}(s,a) = Q^{\pi}{}_{k}(s,a) + \alpha\delta_t e(s,a)$$

Per tutte le coppie (s,a)

Propagate error at time t to all (s,a)

Eleggibilità: e_t(s,a)

SARSA(λ)



```
Initialize $Q(s,a)$ arbitrarily and $e(s,a) = 0$, for all $s, a$
Repeat (for each episode):
    Initialize $s$, $a$
    Repeat (for each step of episode):
        Take action $a$, observe $r$, $s'$
        Choose $a'$ from $s'$ using policy derived from $Q$ (e.g., $\varepsilon$-greedy)
        $\delta \leftarrow r + \gamma Q(s',a') - Q(s,a)$
        $e(s,a) \leftarrow e(s,a) + 1$
        For all $s, a$:
            $Q(s,a) \leftarrow Q(s,a) + \alpha\delta e(s,a)$
            $e(s,a) \leftarrow \gamma\lambda e(s,a)$
        $s \leftarrow s'$; $a \leftarrow a'$
    until $s$ is terminal
```

Trace and Q(λ)

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha\left[r_{t+1} + \gamma \max_{a_{t+1}} Q(s_{t+1},a_{t+1}) - Q(s_t,a_t)\right]$$

Quanto posso propagare all’indietro l’”errore”?

NB L’azione che scelgo può non essere la migliore, la policy viene modificata run-time e la funzione Q viene associata a cammini diversi nel grafo di transizione di stato.

Watkin’s version

Watkin's Q(λ)

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha\left[r_{t+1} + \gamma \max_{a_{t+1}} Q(s_{t+1},a_{t+1}) - Q(s_t,a_t)\right]$$

Suppongo di scegliere a’ = a_{t+1} azione prescritta dalla policy π.

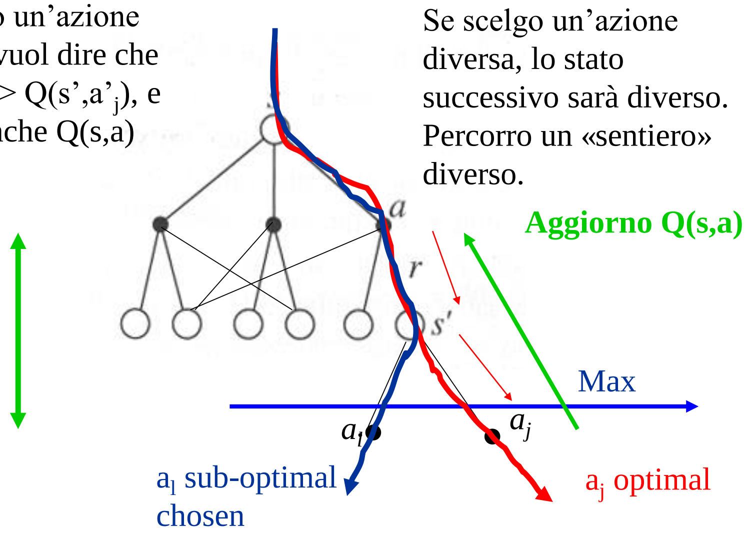
Posso sempre calcolare Q(s_t, a_t), scegliendo il max(Q(s_{t+1}, a_{t+1})). Questo vuole dire ipotizzare di scegliere a_{max} = argmax(max(Q(s_{t+1}, a_{t+1}) ≠ π(s’)); in questo caso: $a_{max} \neq$ a’.

Ma poi devo ripartire da capo perchè da lì in poi seleziono una sequenza diversa di transizioni di stato. Visito il grafo in modo diverso.

Analisi grafica delle mosse ε-eplorative

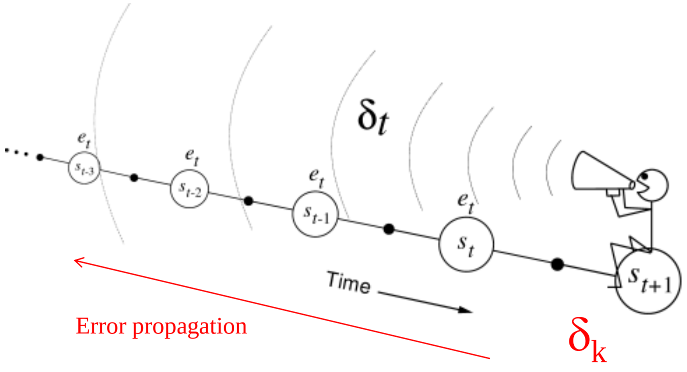
Se scelgo un'azione diversa, vuol dire che $Q(s',a'_l) > Q(s',a'_j)$, e quindi anche Q(s,a) aumenta.

Se scelgo un'azione diversa, lo stato successivo sarà diverso. Percorro un «sentiero» diverso.



Osservazioni

When to stop propagation?



When $s_{t-k} = s_{t+1}$

New strategy for updating eligibility trace

Eligibility set to 0 for the states in which the non-greedy action was chosen.

First decay or set to 0 the eligibility.
Then increment by 1 the eligibility of the current state.

$$e_t(s,a) = \mathcal{I}_{ss_t} \cdot \mathcal{I}_{aa_t} + \begin{cases} \gamma\lambda e_{t-1}(s,a) & \text{if } Q_{t-1}(s_t,a_t) = \max_a Q_{t-1}(s_t,a); \\ 0 & \text{otherwise,} \end{cases}$$

Q-learning

Aggiorno Q:

$$Q_{k+1}(s,a) = Q_{k+1}(s.a) + \alpha\delta_t e_k(s,a)$$

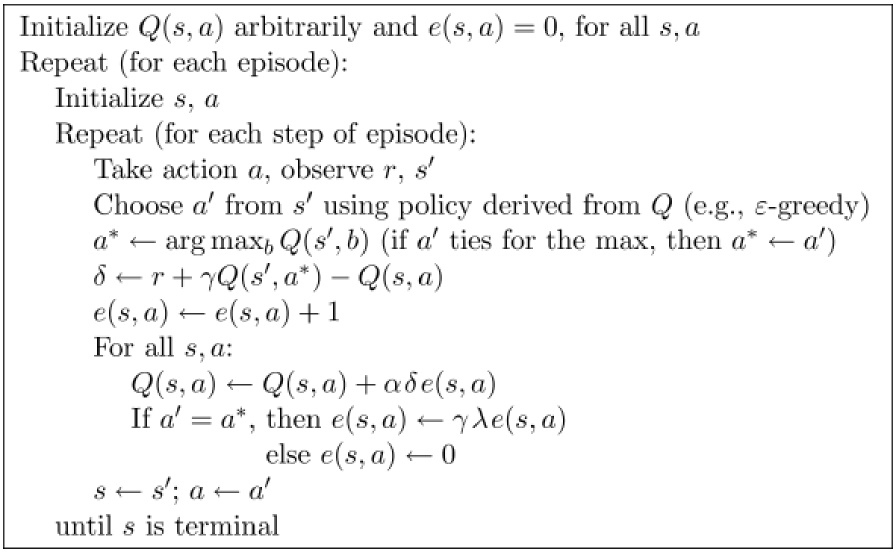
$$\delta_t = r_{t+1} + \gamma \max_{a'} Q_t(s_{t+1}, a') - Q_t(s_t, a_t).$$

Scelta di a:
Se scelgo a_{max}, continuo come SARSA, altrimenti e(s,a) = 0.

Aggiorno e:

$$e_t(s,a) = \mathcal{I}_{ss_t} \cdot \mathcal{I}_{aa_t} + \begin{cases} \gamma\lambda e_{t-1}(s,a) & \text{if } Q_{t-1}(s_t,a_t) = \max_a Q_{t-1}(s_t,a); \\ 0 & \text{otherwise,} \end{cases}$$

Algorithm for Watkin's Q(λ)



```
Initialize $Q(s,a)$ arbitrarily and $e(s,a) = 0$, for all $s, a$
Repeat (for each episode):
    Initialize $s$, $a$
    Repeat (for each step of episode):
        Take action $a$, observe $r$, $s'$
        Choose $a'$ from $s'$ using policy derived from $Q$ (e.g., $\varepsilon$-greedy)
        $a^* \leftarrow \arg\max_b Q(s', b)$ (if $a'$ ties for the max, then $a^* \leftarrow a'$)
        $\delta \leftarrow r + \gamma Q(s', a^*) - Q(s,a)$
        $e(s,a) \leftarrow e(s,a) + 1$
        For all $s, a$:
            $Q(s,a) \leftarrow Q(s,a) + \alpha \delta e(s,a)$
            If $a' = a^*$, then $e(s,a) \leftarrow \gamma \lambda e(s,a)$
                              else $e(s,a) \leftarrow 0$
        $s \leftarrow s'$; $a \leftarrow a'$
    until $s$ is terminal
```

Sommario

Traccia

Esempi

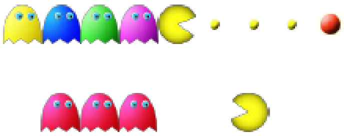
Clever Pac-man

Tohru Iwatani, formato arcade da sala, 1980.

N.A.Borghese, A.Rossini and C.Quadri (2012) Clever Pac-man, Proceedings of the 21st Italian Workshop on Neural Nets, WIRN2011, Frontiers in Artificial Intelligence and Applications, IOS Press (Apolloni, Bassis, Esposito, Morabito eds.), pp.11-19.

Applied Intelligent Systems Laboratory
Computer Science Department
University of Milano
http://ais-lab.dsi.unimi.it

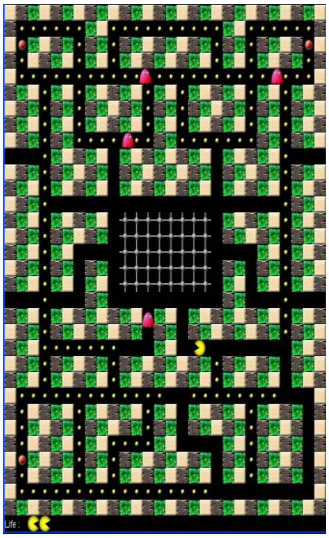
Motivation



How can we make a computer agent play Pac-man?

The Pac-man game

Arcade computer game
- An agent that moves in a maze. The agent is a stilyzed yellow mouth that opens/closes.
- The maze is constituted of corridors paved with (yellow) pills.
- When all pills are eaten the agent can move to the next game level.
- Some enemies, with the shape of pink ghosts, are present, that go after the pacman.
- Special pills, called power pills (pink spheres) are present among the pills. They allow the pacman to eat the ghosts but their effect lasts for a limited amount of time.
- Each eaten pill is worth one point, while each eaten ghost is worth 200, 400, 800, 1600 points (first, second, third ghost).



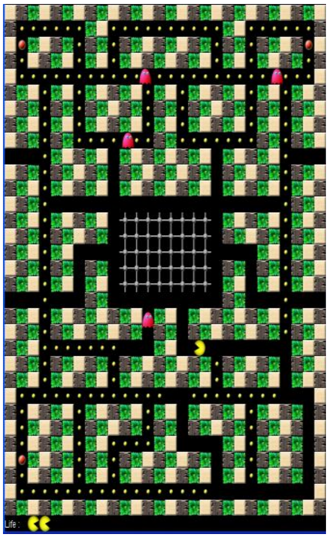
Pac-man as a learning agent

No a-priori information is available to the pac-man.

Enviroment
The environment (maze structure, ghosts and pills position) is not known to the pac-man → **environment identification.** Large number of cells (≅ 30 x 32 = 960) and situations.

Reward is not known.

Ghosts behavior has also to be specified.



Agent:
•Elements: State, Actions, Rewards, Value function.
•Policy: Action = f(State).
•Learning machinery.

Pac-man learning

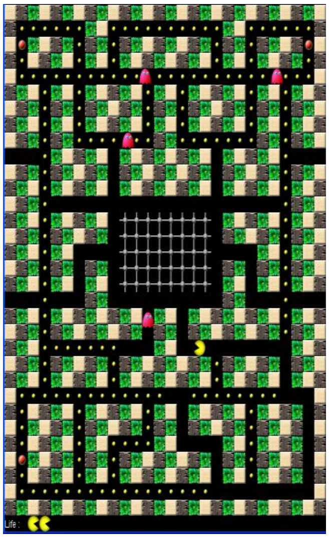
Reinforcement learning is explored here. Fuzzy state definition allows managing the number of cells

Agent:

- •Elements: State, Actions, Rewards, Value function.
- •Policy: Action = f(State).
- •Learning machinery.

Environment:

- •Ghosts behavior.
- •Rewards



The ghosts original behavior

In the original game design (*Susan Lammers: "Interview with Toru Iwatani, the designer of Pac-Man", Programmers at Work 1986*), the four ghosts had different personalities:
Ghost #1, chases directly after Pac- man.
Ghost #2, positions himself a few dots in front of Pac-man mouth (if these two ghosts and the Pac-man are inside the same corridor a sandwich movement occurs).
Ghost #3 and #4, move randomly.

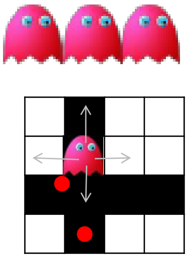
In the present implementation all the four ghosts can assume all three possible behaviors depending on the situation of the game (the state).

Ghosts have to escape the Pac-man when the power pill is active.

The more the game progresses the more the ghosts have to aim to the Pac-man.

The ghosts behavior

At each step each ghost has to decide if moving north, south, east, west.



Shy behavior. The ghost moves away from the closest ghost This allows distributing the ghosts inside the maze. When the power pill is active, the ghosts tend to move as far as possible from the Pac-man. ***The direction the maximize the increment of distance is chosen.*** When ties are present, the Pac-man makes a randomized choice to avoid stereotyped behavior.

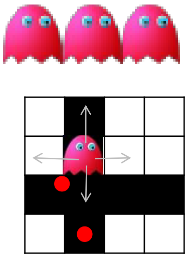
Random behavior. It chooses an admissible direction randomly.

Hunting behavior. The ghost chooses the direction of the minimum path to the Pac-man. Minimum path has to be updated at each step as the Pac-man moves. The Floyd-Warshall algorithm is used to pre-compute the minimum path, distance between pairs of cells, for each cell of the maze, at game loading time.

Defence behavior. The ghosts go in the area in which the pills density is maximum. To this aim the maze is subdivided into nine partially overlapped areas: {0 - ½; ¼ - ¾; ½ - 1} and the ghost aims to the center of the area waiting for the Pac-man.

The Fuzzy behavior implementation

At each step each ghost chooses among the four possible behaviors: shy, random, hunting and defence, according to a **fuzzy policy**.



Input fuzzy variables are:

- distance between the ghost and the Pac-man
- distance with the nearest ghost.
- frequency of the Pac-man eating pills.
- life time of the Pac-man (that is associated to its ability, the more the game progresses, the more aggressive become the ghosts).
- Power pill active

A set of rules have been designed like for instance:

- *If pacman_near AND skill_good, Then hunting_behavior*
- *If pacman_near AND skill_med AND pill_med, Then hunting_behavior*
- *If pacman_near AND skill_med AND pill_far, Then hunting_behavior*
- *If pacman_med AND skill_good AND pill_far, Then hunting_behavior*
- *If pacman_med AND skill_med AND pill_far, Then hunting_behavior*
- *If pacman_far AND skill_good AND pill_far, Then hunting_behavior*

Input class boundaries are chosen so that ghosts have hunting as preferred action (four times the other actions) in real game situations.

At start all ghosts are grouped in the center.

The Pac-man and fuzzy Q-learning

Fuzzy description of the state is mandatory to avoid combinatorial explosion of the number of the states.

The state of the game is described by three (fuzzy) variables:
• minimum distance from the closest pill.
• minium distance from the closest power pill.
• minimum distance from a ghost.

Three fuzzy classes for each variable -> 27 fuzzy states.

Fuzzy aggregated state	Closest ghost	Closest pill	Closest power pill
1	Low	Low	Low
2	Low	Low	Medium
3	Low	Low	High
4	Low	Medium	Low
5	Low	Medium	Medium
6	Low	Medium	High
7	Low	High	Low
8	Low	High	Medium
9	Low	High	High
10	Medium	Low	Low
11	Medium	Low	Medium
12	Medium	Low	High
13	Medium	Medium	Low
14	Medium	Medium	Medium
15	Medium	Medium	High
16	Medium	High	Low
17	Medium	High	Medium
18	Medium	High	High
19	High	Low	Low
20	High	Low	Medium
21	High	Low	High
22	High	Medium	Low
23	High	Medium	Medium
24	High	Medium	High
25	High	High	Low
26	High	High	Medium
27	High	High	High

Q-learning

Agent – the pacman

- State (fuzzy states) – {s}
- Actions (Go to Pill, Go to Power Pill, Avoid Ghost, Go after Ghost) – {a}

Environment

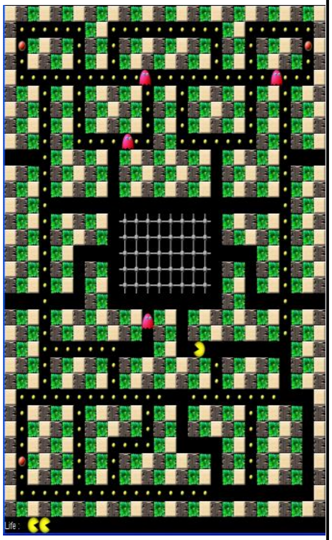
Related to enviroment, not known to the agent:

- Environment evolution: $s_{t+1} = g(s_t, a_t)$.
- Reward: points gained $r_{t+1} = r(s_t, a_t, s_{t+1})$ in particular situations, e.g. Pill eaten, death)

The pacman optimizes through learning:

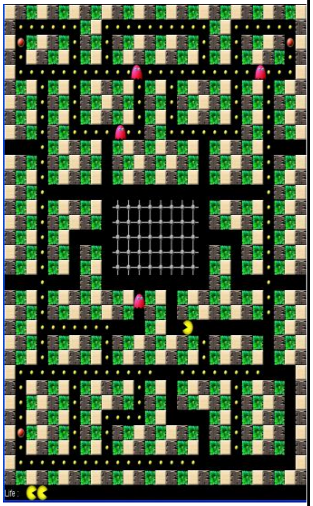
- Policy: $a_t = f(s_t)$
- Value function: $Q = Q(s_t, a_t)$

$Q(s_t,a_t) = Q(s_t,a_t) + \alpha[r_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t,a_t)]$



Fuzzy State of the Pac-man

We measure the state:
-The distance from the closest ghost, c1.
- The distance from the closest pill, c2.
- The distance from the closest power pill, c3.
Each element can fall in more than one state at each time step



We compute the membership to each fuzzy state s_j as:

$$\mu(s_j) = \frac{\sum_{i=1}^{3} m(c_i)}{3}$$

Membership of each of the 3 components of the state. We update Variables taking into account fuzzyness of states.

With m(.) degree of membership of the measurement c_i to one of the fuzzy classes(small, medium, large) associated to each state variable (distance from closest ghost, closest pill, closest power pill).

More than one state can be active at each time step and the degrees of activity, $\mu(s_j)$ add to one.

Fuzzy Q-learning

The value function for the state s^*, constituted of all the fuzzy states, s_i, with their membership value, from which the Pac-man moves, with action a, receives contribution from all the next state s_{t+1}* of the Pac-man inside the maze:

$$Q(s_t*, a_t) = \frac{1}{n}\sum_{i=1}^{n} \mu(s_{t,i})\; q\left(s_{t,i}, a_t\right)$$

where q(.) is updated using Q-learning strategy as:

$$q(s_{t,i}, a_t) = q(s_{t,i}, a_t) + \alpha_{s,a}\left[r + \gamma \cdot \frac{1}{N}\max_{a'} Q(s_{t+1}, a') - q(s_{t,i}, a_t)\right]$$

α is chosen as:

$$\alpha_{s,a} = \frac{1}{\sum_{0}^{\tau=t-1} \mu\left(s_{\tau,i}\right)}$$

That is a natural extension of running average computation and it is inversely proportional to the cumulative membership of all the states active at that time step.ù

For each fuzzy state, a different optimal action for the next state s', is identified according to Q(s',a'). The action implemented in the one associated to the maximum fitness of the associated fuzzy state.

Implementation issues of Pac-man policy

Policy: $a_t = f(s_t)$

a_t = {Go to Pill, Go to Power Pill, Avoid Ghost, Go to Ghost}

Go to Pill. The Pac-man always goes to the closest pill, independently on the position of the ghosts. If ties occur the choice is randomized to avoid stereotyped behavior.

Go to Power Pill. Similar as above.

Go to Ghost. Similar as above.

Avoid Ghost. If only the closest ghost is considered, the Pac-man would easily run into a second ghost. The move the minimizes the weighted distance with all the ghost could be considered, but this would move the Pac-man in a small area close to the corners of the maze. We have implemented a weighted distance computed only inside a small area around the actual position of the Pac-man (that changes at each time step). Moreover, in case of ties, the Pac-man choses the direction the leads to the closest power pill (if still present in the maze).

Additional implementation issues

Few heuristics have been introduced:

Persistence *(cf. DeLooze, L.L.; Viner, W.R.; "Fuzzy Q-learning in a nondeterministic environment: developing an intelligent Ms. Pac-Management", Computational Intelligence and Games, 2009. CIG 2009. pp.162-169, 7-10 Sept. 2009).* Forcing the same action for n steps (n=5 here).

Persistence removal. When power pill effect ends. A brisk change of behavior is often observed.

Taboo. Inhibits the Pac-man to return in the previous state.

Parameters role

Rewards. The death of the Pac-man receives instant reward of -1000. A less negative reward was not enough to compensate all the positive points earned during a typical game. A more negative reward made the Pac-man “depressed” and little inclined to look for pills.

Fuzzy classes boundary: d=5, d=12 and d = 25 were assumed as maximum distance for the classes: low, medium and large. These values have been experimentally set analyzing the game results.

Pills reward: no particular effect was observed when the value was in the range [0.1 ÷ 1].

Greediness of the policy

Greediness of the policy: ε-greedy policy is fundamental to obtain very good results.

With random policy (blue) little points are gained.

Some more points can be gained if the Pac-man always chooses “avoid ghosts” unless he has eaten the Power Pill (orange).

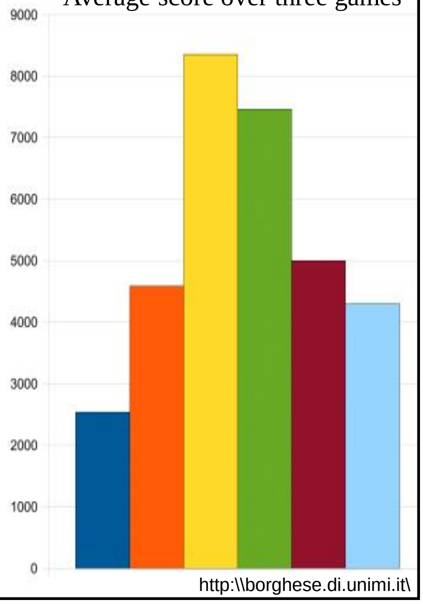
Maximum reward is obtained when Q-learning with ε-greedy policy with ε=0.1) choice is adopted and r = 0.1 per pill (yellow).

A high reward is obtained when Q-learning with ε-greedy policy with ε=0.1) choice is adopted and r = 1 per pill (green).

Less reward is obtained with Q-learning with greedy policy (brown).

An even small reward is obtained with Q-learning with greedy policy, when fuzzy classes boundaries are different: d = {6, 18, 30} (cyan).

Average score over three games



Conclusion and further developments

- Highest score was around 4,500 and reported in *DeLooze, L.L.; Viner, W.R.; "Fuzzy Q-learning in a nondeterministic environment: developing an intelligent Ms. Pac-Man agent", Computational Intelligence and Games, 2009. CIG 2009. pp.162-169, 7-10 Sept. 2009.* We obtain here a large improvement in the score.
- Fuzzy approach has made RL approach feasible.

- We have only considered the bonus represented by power pills.
- A single scheme was used.
- Fuzzy classes boundaries were not optimized.
- A human player elaborates **strategies** both in chasing and escaping that are based on a global “view” of the game. This would require a much elaborate learning machinery than “simple” RL.

- Here is the Pac-man learning live....

Launch Fuzzy Pac-man

- Spostarsi nella cartella *bin dell'applicazione.*
- Lanciare il file main:

```
java pacman.PacmanMAIN
```

Interacting with an artificial partner: modeling the role of emotional aspects

I. Cattinelli, M. Goldwurm and N.A. Borghese (2008) Biological Cybernetics, pp.254-259.

Applied Intelligent Systems Laboratory
Computer Science Department
University of Milano
http://ais-lab.dsi.unimi.it

Sommario

Traccia

Esempi