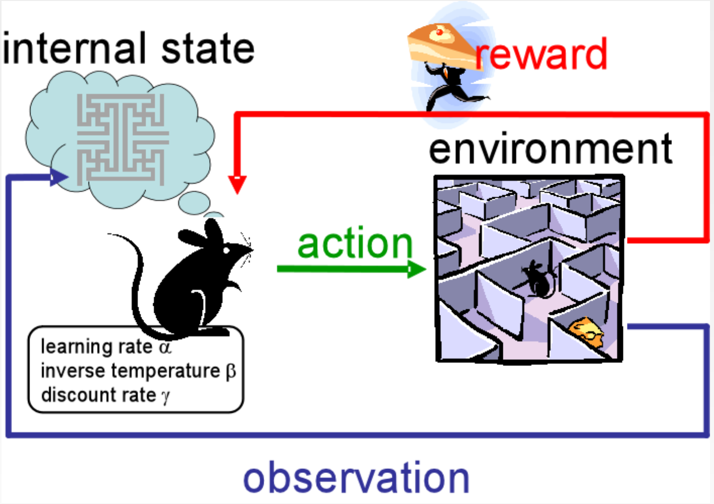
INFO 284 – Machine Learning

Reinforcement Learning

Bjørnar Tessem

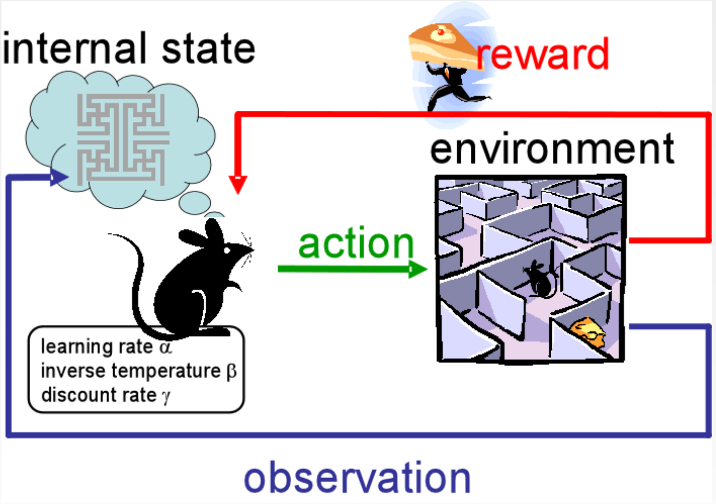
What is reinforcement learning

- There is no supervisor present.
- Importance of time/sequences.
- Concept of delayed rewards.
- The agents action effects its next input.



What is reinforcement learning

- Receive feedback in the form of **rewards**
- Agent's **utility** is defined by the reward function
- Must (learn to) act so as to **maximize expected rewards**
- All learning is based on observed samples of outcomes!



Example: Learning to Walk

Initial

[Kohl and Stone, ICRA 2004]

[Video: AIBO WALK – initial]

Example: Learning to Walk

Finished

[Kohl and Stone, ICRA 2004]

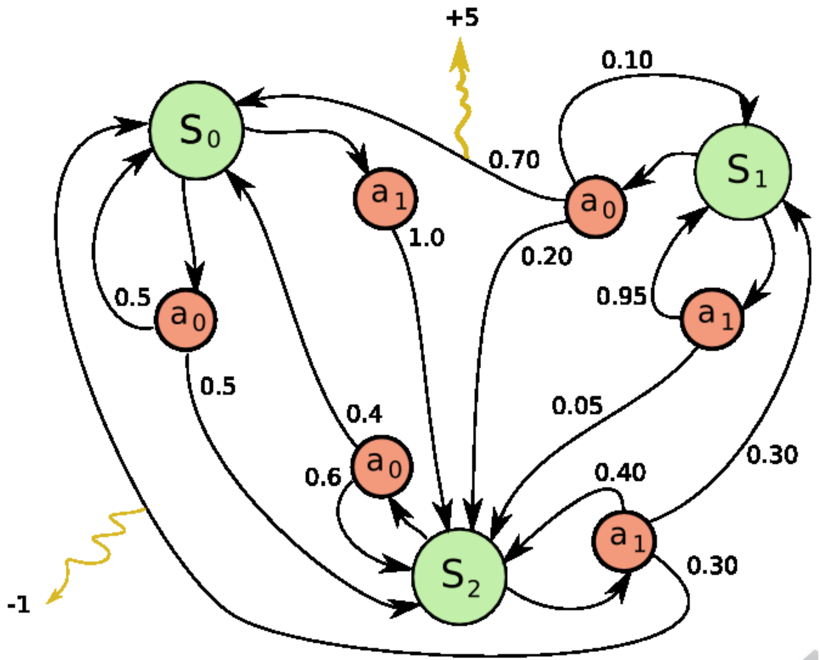
[Video: AIBO WALK – finished]

Concepts

- Policy - defines the learning agent's way of behaving at a given time. It is a mapping from perceived states of the environment to actions to be taken when in those states
  - Reflex agents – agents that always behave the same way
- Reward - defines the goal of a reinforcement learning problem. It defines what are good and what are bad events for the agent in the immediate sense.
- Value function (Utilty function)- specifies what is a good outcome for the agent long term. The value of a state is a total amount of reward an agent can expect to accumulate over the future starting from that state.
- Environment - something that mimics the behaviour of the environment, or allows inferences to be made about how the environment will behave.

Markov decision process

- A sequential decision problem for a fully observable, stochastic environment with a Markovian transition model and additive rewards.
- Solution should specify what agent should do for any state that the agent might reach (**policy**)
- Stochastic nature means that the a policy starting from the same initial state may lead to different environment.
- The goal is to find an **optimal policy** (a policy that yields the highest expected utility)



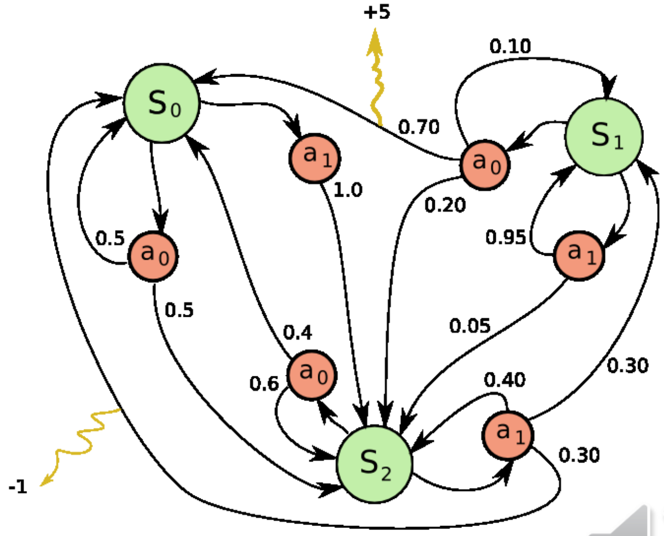
Markov decision process

- A Markov decision process (MDP) is a structure that consists of a set of states (with an initial state s0); a set A(s) of actions in each state; a transition model **P(s'|s, a)**; and a reward function R(s)
- We want to win as highest reward as possible
- What the agent should do for any state that the agent might reach is described in a **policy**
- **π(s)** - what the agent should do in state s.
- optimal policy **π*(s)**
- Additive rewards: The utility of a state sequence is

$U_h([s_0, s_1, s_2, \ldots]) = R(s_0) + R(s_1) + R(s_2) + \ldots$

- Discounted rewards: The utility of a state sequence is

$U_h([s_0, s_1, s_2, \ldots]) = R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \ldots$



How much is a policy worth?

The expected utility obtained by executing π starting in s is given by

$$U^\pi(s) = E\left[\sum_{t=0}^{\infty} \gamma^t R(S_t)\right],$$

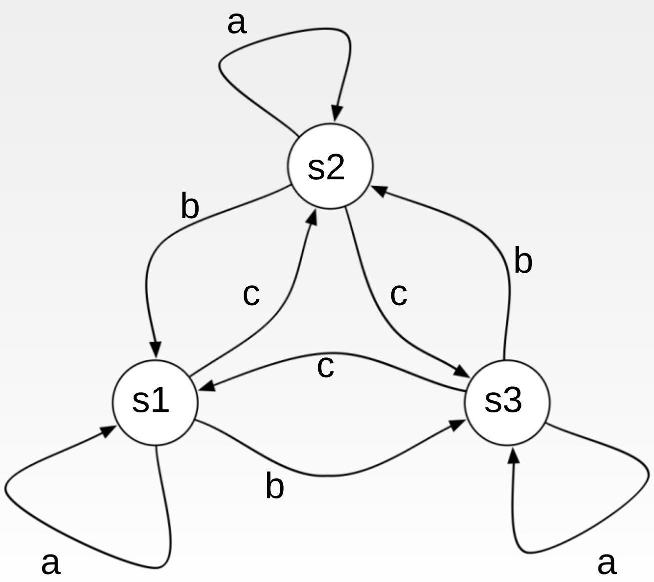
$$\pi_s^* = \underset{\pi}{\operatorname{argmax}}\, U^\pi(s).$$

- Learn an optimal **policy**:
  - Utility-based agent - policy maps states to utility
  - must have a model of the environment
  - Learns by optimization

Q-learning

- Learning by doing
- Learn the expected utility of taking a given action

state	action	Q-value
s1	a	?
s1	b	?
s1	c	?
s2	a	?
s2	b	?
s2	c	?
s3	a	?
s3	b	?
s3	c	?



Q-learning (learning an action-utility function)

- Initialise an array Q(**s**tate,**a**ction) arbitrarily (choose utilities).
- At each time t the agent selects an action, observes a reward, enters a new state (that may depend on both the previous state and the selected action), and updates Q.
- Update = we change the utility for the state we just came out of

$$Q(s_t, a_t) \leftarrow (1-\alpha) \cdot \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \overbrace{\left(\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_{a} Q(s_{t+1}, a)}_{\text{estimate of optimal future value}} \right)}^{\text{learned value}}$$

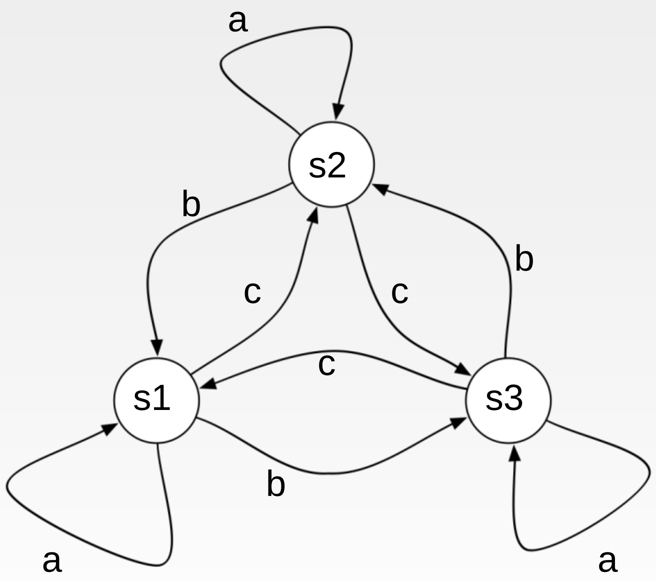
best action in
next state

HTTPS://TOWARDSDATASCIENCE.COM/INTRODUCTION-TO-Q-LEARNING-88D1C4F2B49C

Q-learning

In state s2 select a, receive -10, $\gamma = 0.5$, $\alpha = 0.1$

state	action	Q-value
s1	a	0
s1	b	0
s1	c	0
s2	a	0
s2	b	0
s2	c	0
s3	a	0
s3	b	0
s3	c	0

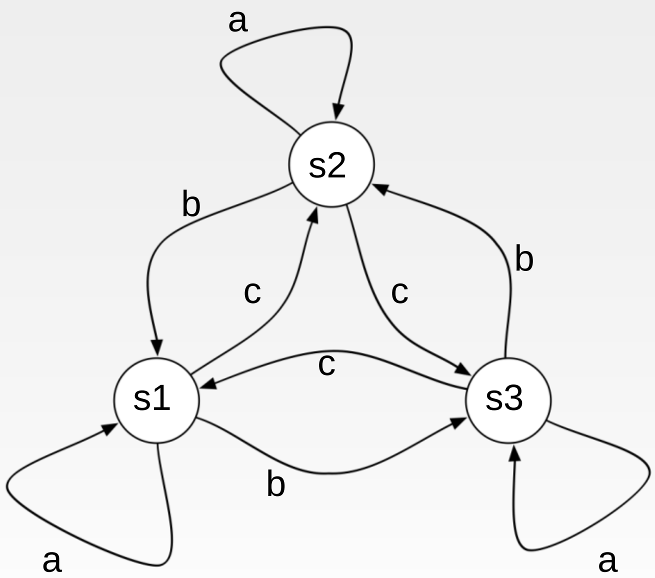


state	action	Q-value
s1	a	0
s1	b	0
s1	c	0
s2	a	-1
s2	b	0
s2	c	0
s3	a	0
s3	b	0
s3	c	0

Q-learning

- In state s2 select b, receive 20, γ = 0.5, α = 0.1

state	action	Q-value
s1	a	0
s1	b	0
s1	c	0
s2	a	-1
s2	b	0
s2	c	0
s3	a	0
s3	b	0
s3	c	0

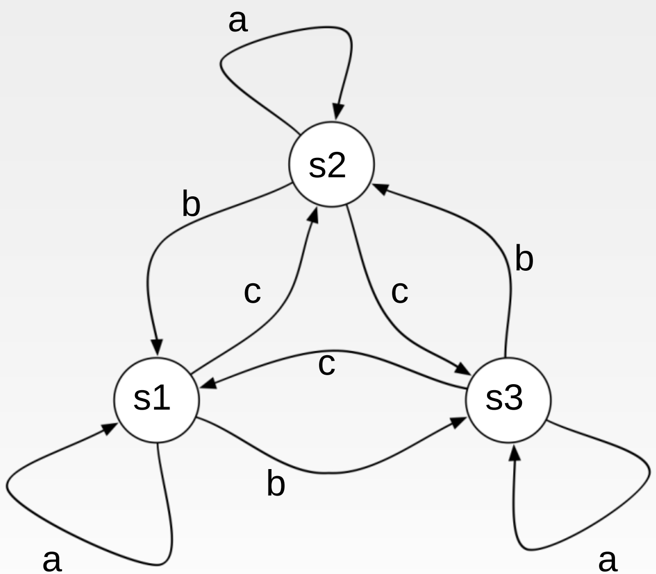


state	action	Q-value
s1	a	0
s1	b	0
s1	c	0
s2	a	-1
s2	b	2
s2	c	0
s3	a	0
s3	b	0
s3	c	0

Q-learning

- In state s1 select c, receive 10, $\gamma = 0.5$, $\alpha = 0.1$

state	action	Q-value
s1	a	0
s1	b	0
s1	c	0
s2	a	-1
s2	b	2
s2	c	0
s3	a	0
s3	b	0
s3	c	0



state	action	Q-value
s1	a	0
s1	b	0
s1	c	1.1
s2	a	-1
s2	b	2
s2	c	0
s3	a	0
s3	b	0
s3	c	0

Q-learning properties

- Fully observable states and rewards
- Do not need a complete environment model initially
  - Add states to model as encountered
- Converges if learning rate decreases slowly
- Based on random exploration of actions

Exploration vs. exploitation

- Learning when doing!
- Exploit learned Q-values
  - Choose action with best Q-value
  - Will not learn if only exploiting
- Explore
  - Choose random action
- Strategies
  - ε-exploration – explore randomly part of the time
    - Reduce ε as time goes
  - Optimism in face of uncertainty – assign high initial q-values
    - Need to start with realistic Q-values
- SARSA
  - Include risk of exploration in updates
  - Look one more step ahead

Additional approaches

- Random outcome of actions
  - Learn probabilities for transitions
  - Build environment model
- Learn supervised models for value of state
  - Environments are complex
  - Use state properties as features and Q-values as targets
- Learning strategies in multiagent environments
  - Game playing (Alpha Zero)

uib.no