Reinforcement Learning Lecture 1
Reinforcement Learning and SARSA

Wulfram Gerstner
EPFL, Lausanne, Switzerland

Part 1: Examples of Reward-based Learning

Objectives for Lecture RL1 (Part 1-3)

- Reinforcement Learning (RL) is learning by rewards
- Agents and actions, states and rewards
- Convergence in expectation, online and batch.

Reading:
Sutton and Barto, Reinforcement Learning (MIT Press, 2nd edition 2018)
Chapters: 1.1-1.4; 2.1-2.6; 3.1-3.5; 6.4

Reading for this week:

Sutton and Barto, Reinforcement Learning (MIT Press, 2nd edition 2018, also online)

Chapters: 1.1-1.4; 2.1-2.6; 3.1-3.5; 6.4

Background reading:

Silver et al. 2017, Archive

Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm

REPETITION: Artificial Neural Networks for action learning

No labeled data?

Replaced by:

‘Value of action’

- ‘goodie’ for dog
- ‘success’
- ‘compliment’

BUT:

Reward is rare:

‘sparse feedback’ after a long action sequence

Previous slide. (already shown before the break)

How does a human learn to play table tennis: How does a child learn to play the piano? How does a dog learn to perform tricks?

In all these cases there is no supervisor. No master guides the hand of the players during the learning phase. Rather the player ‘discovers’ good movements by rather coarse feedback. For example, the ball in table tennis does not land on the table as it should. That is bad (negative feedback). The ball has a great spin so that the opponent does not get. This is good (positive feedback).

Similarly, it is hard to tell a dog what to do. But if you reinforce the dog’s behavior by giving a ‘goodie’ at the moment when it spontaneously performs a nice action, then it can learn quite amazing things.

In all these cases it is the ‘reward’ that guides the learning. Rewards can be the goodie for the dog, or just the feeling ‘now I did well’ for humans.

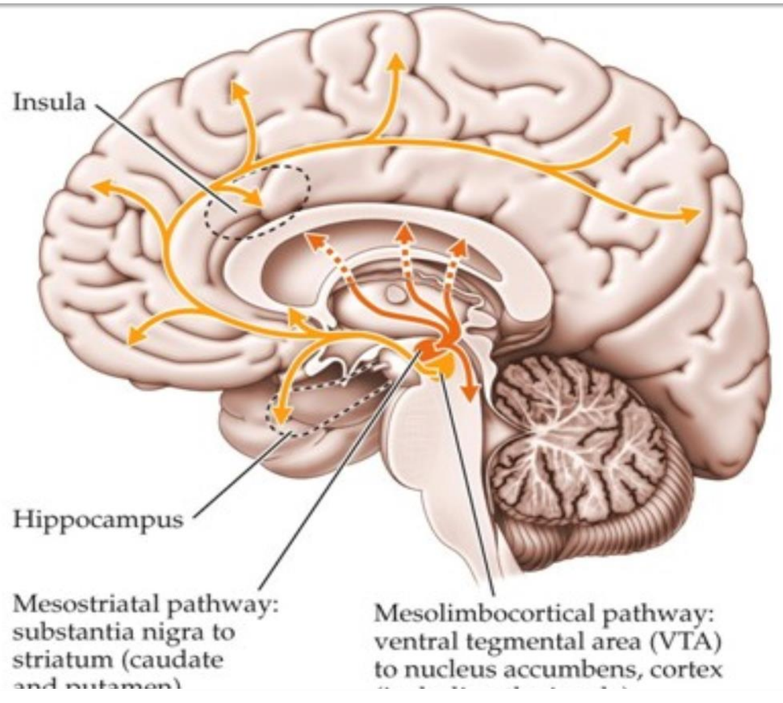
Reward information is available in the brain

Neuromodulator **dopamine**:
Signals “reward minus expected reward”

Schultz et al., 1997,
Waelti et al., 2001
Schultz, 2002

‘success signal’

Dopamine



Previous slide.

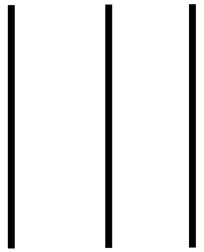
Inside the brain, reward information is transmitted by the neuromodulator dopamine. Neurons that use dopamine as their chemical transmission signal are situated in nuclei below the cortex and have cables (axons) that reach out to vast areas of the brain.

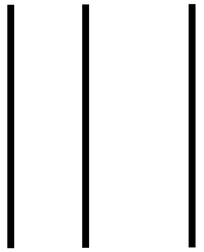
As we will see later, neurons that communicate with the neuromodulator dopamine transmit a generic success signal that is not just reward, but something like ‘reward minus expected reward’.

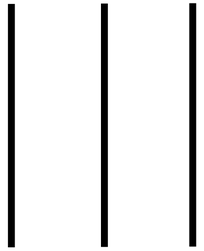
To conclude, reward information is available throughout the brain.

Examples of reinforcment learning

Middle bar: shifted left or shifted right?

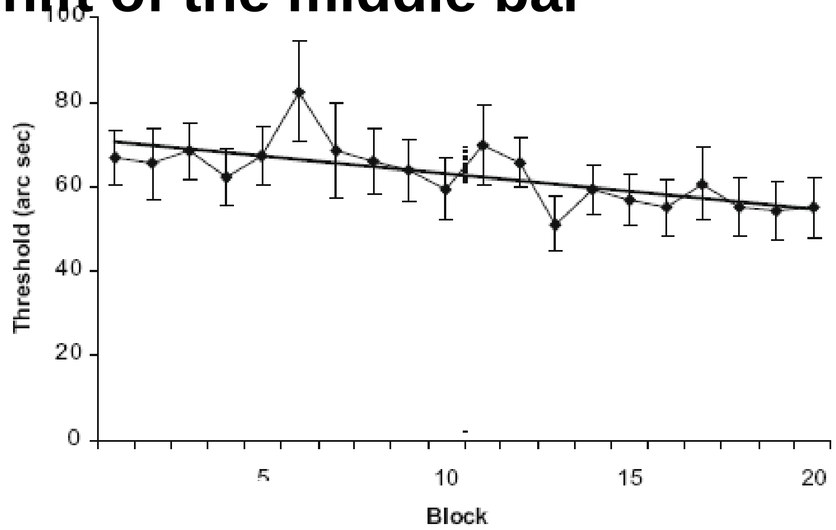






Observers get better at seeing the shift of the middle bar

Min. shift



Feedback:
tone for wrong response

Tartaglia,Aberg,Herzog 2009

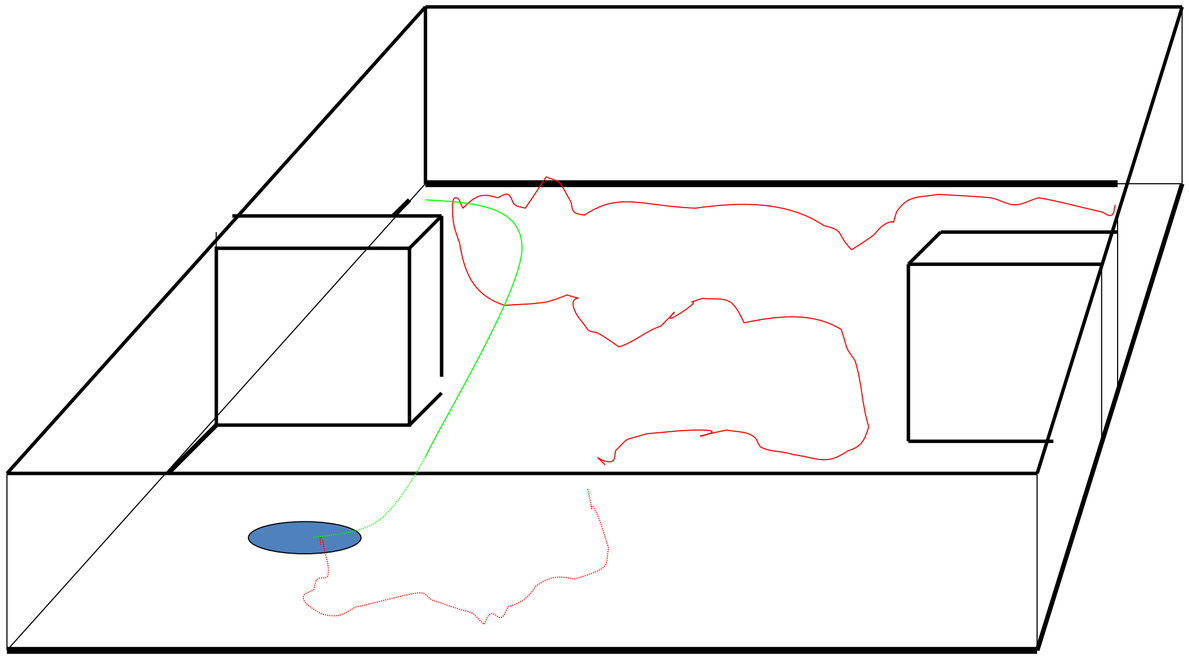
Previous slide (This example is not shown in class)

Let us look at a few additional examples, beyond table tennis.

Humans can get, by practice and feedback, better at recognizing a visual pattern with three bars. The task is to distinguish cases where the middle bar is shifted to the left from those where it is shifted to the right.

Bottom right:
The minimal shift that is just recognizable decreases over time (1 block = 1 practice session) indicating learning.
The feedback signal is just right or wrong.

Examples of reinforcement learning: animal conditioning

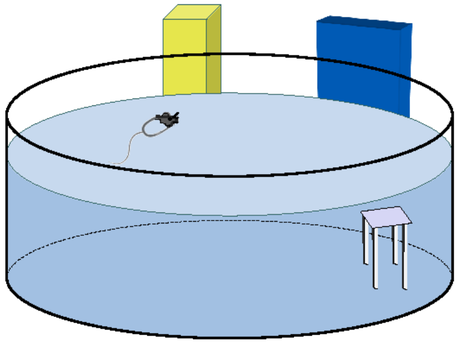


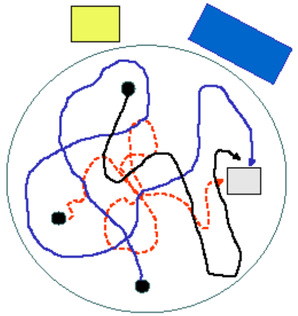
Previous slide. (already shown before the break)

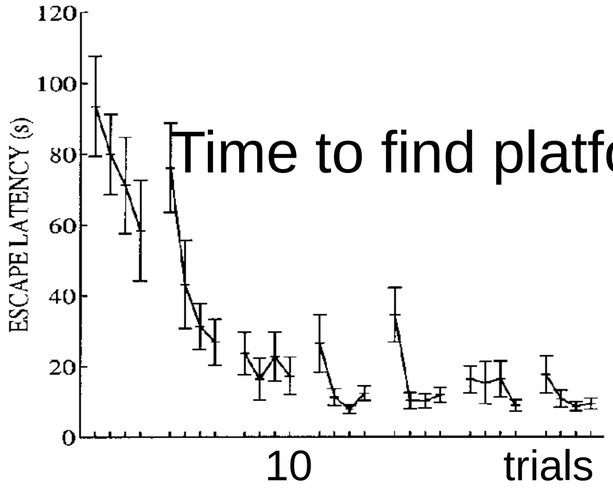
If you put a rat into an environment it will wander around. Suppose that, at some place, it discovers a food source hidden below the sand of the surface.
After a couple of trials it will go straight to the location of the food source which implies that it has learned the appropriate sequence of actions in the environment to find the food source.

Examples of reinforcement learning: animal conditioning

Morris Water Maze







Time to find platform

Rats learn to find
the hidden platform

(Because they like to
get out of the cold water)

Foster, Morris, Dayan 2000

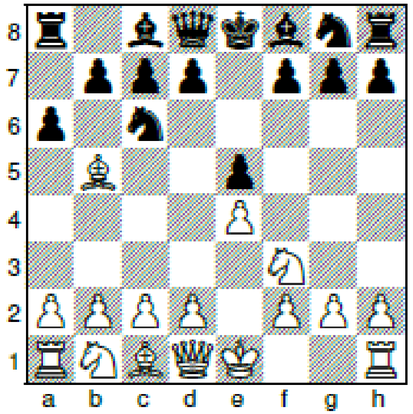
Previous slide. (This example is not shown in class)

Actual experiments for location learning are often performed in a Morris water maze. In the maze, there are 4 starting points and one target location which is a platform hidden (in milky water) just below the water surface. The rat does not like to swim in cold water and therefore tries to find the platform.

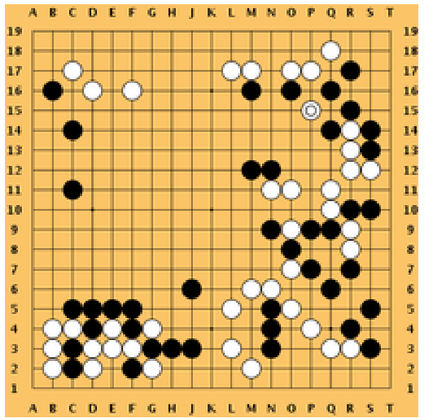
After a few trials it swims straight to the platform.
Bottom right: the time to reach the platform decreases over trials, indicating learning.

REPETITION: Deep reinforcement learning

Chess



Go



Artificial neural network (*AlphaZero*) discovers different strategies by playing against itself.

In Go, it beats Lee Sedol

Previous slide.

In chess a neural network trained by reinforcement learning discovers winning strategies by playing against itself. Similarly, a neural network playing Go against itself learns to play at a level so as to beat one of the world champions.

The aim of the class is to arrive at Deep Reinforcement Learning (Deep RL): Today we start with (standard) RL, in a few weeks we turn to deep networks, and in May we will turn to Deep RL.

Deep reinforcement learning

Network for choosing action

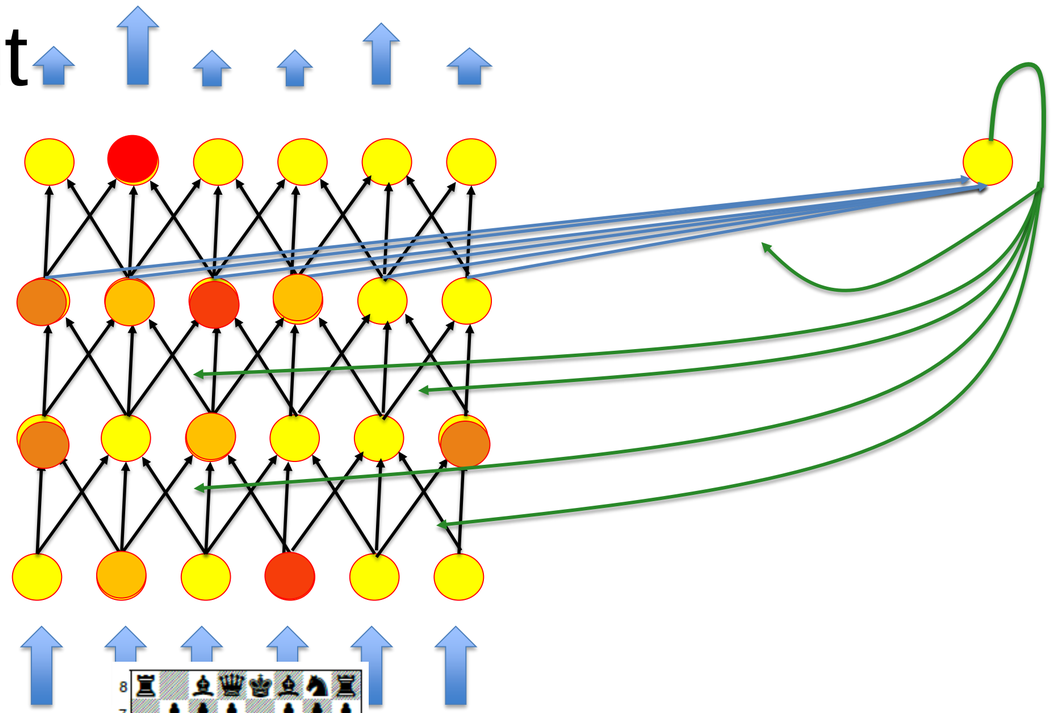
action:

Advance king

2nd output for **value** of state:

probability to win

output



input

Learning by success signal

- change connections

aim:

- choose next action to win

aim for value unit:

- predict value of current position

Previous slide. (already shown before the break)

At the end of this semester, you will be able to understand the algorithms and network structure used to achieve these astonishing performances. Important are two types of outputs.

Left: different output neurons represent different actions.
Right: an additional output neuron represents the value of the present state; we can loosely define the value as the probability to win, or the ‘average reward’ that you can get starting from this state.

The input is a representation of the present state of the game.

Details will become clear toward the end of the semester; at the moment the aim is just to give you a flavor of the high-level concepts.

Deep Reinforcement Learning:

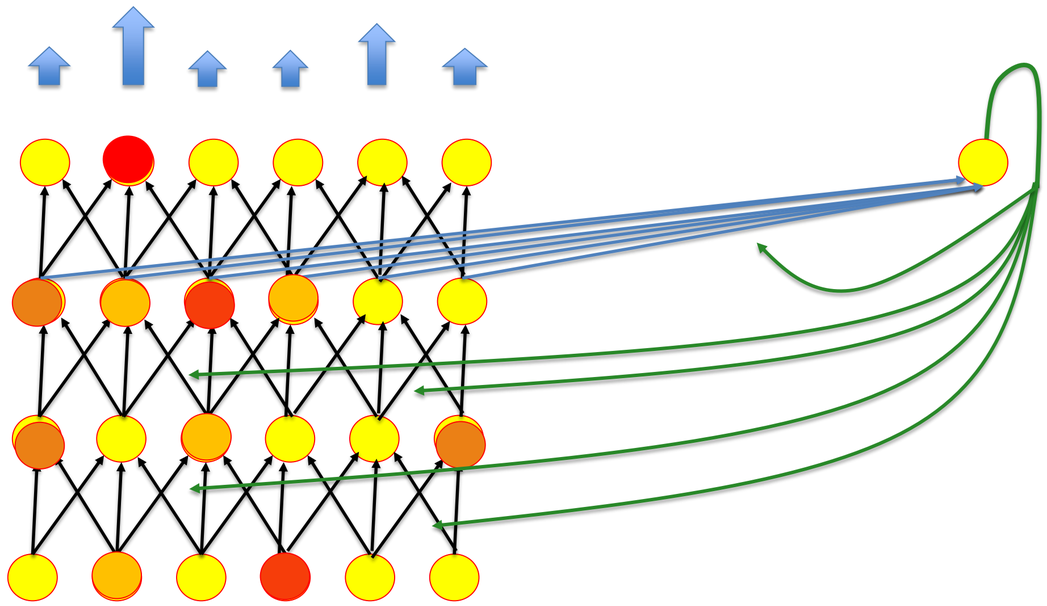
Control a dynamic system (example of past minproject)

actions

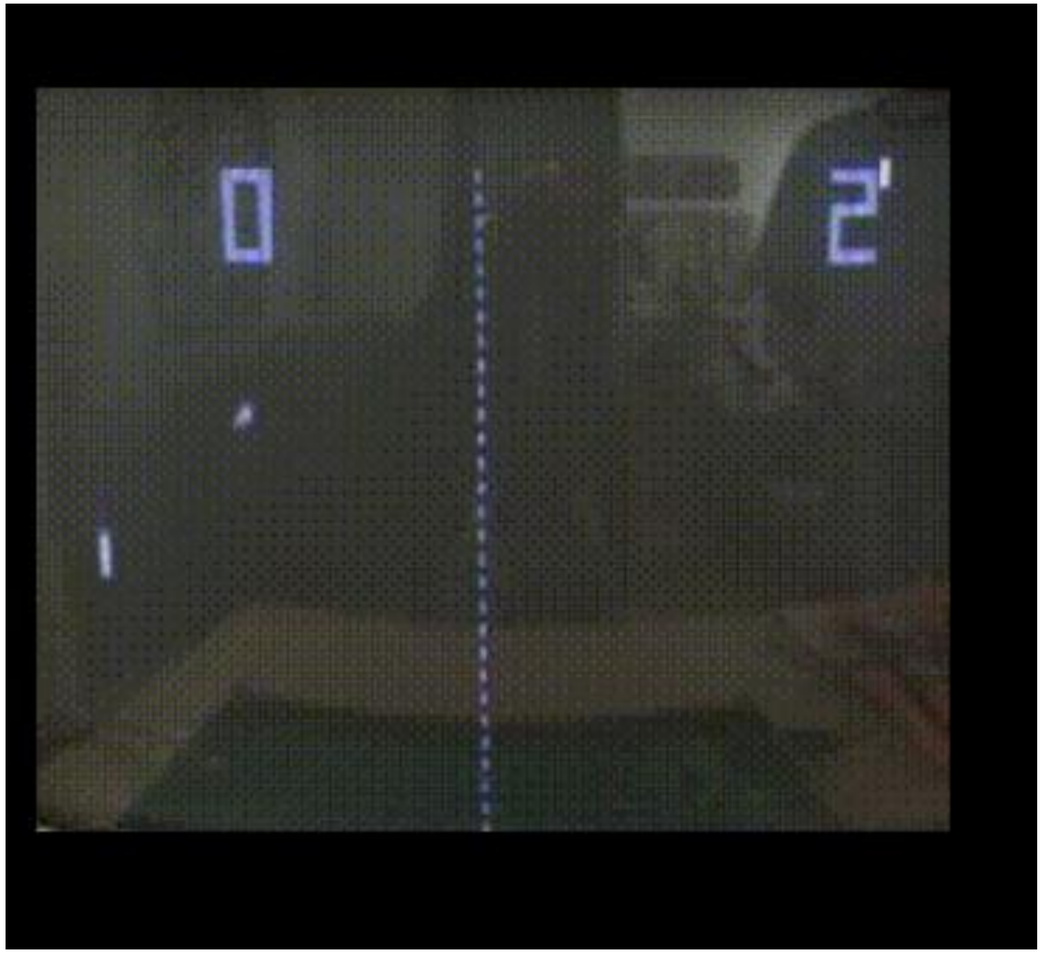
advance

push left

value



Example: Play Pong (Atari game)



Previous slide.

In the miniproject training will be based on reward: successful behavior of the simulated agent will give positive rewards.

Quiz: Rewards in Reinforcement Learning

[ ] Reinforcement learning is based on rewards
[ ] Reinforcement learning aims at optimal action choices
[ ] In chess, the player gets an external reward after every move
[ ] In table tennis, the player gets a reward when he makes a point
[ ] A dog can learn to do tricks if you give it rewards at appropriate moments

Previous slide. Your notes (already shown before the break)

.

Reinforcement Learning Lecture 1
Reinforcement Learning and SARSA

Wulfram Gerstner
EPFL, Lausanne, Switzerland

Part 2: Elements of Reinforcement Learning

- Examples of Reward-based Learning
- **Elements of Reinforcement Learning**

Previous slide.
We now start with the formalization of reinforcement learning

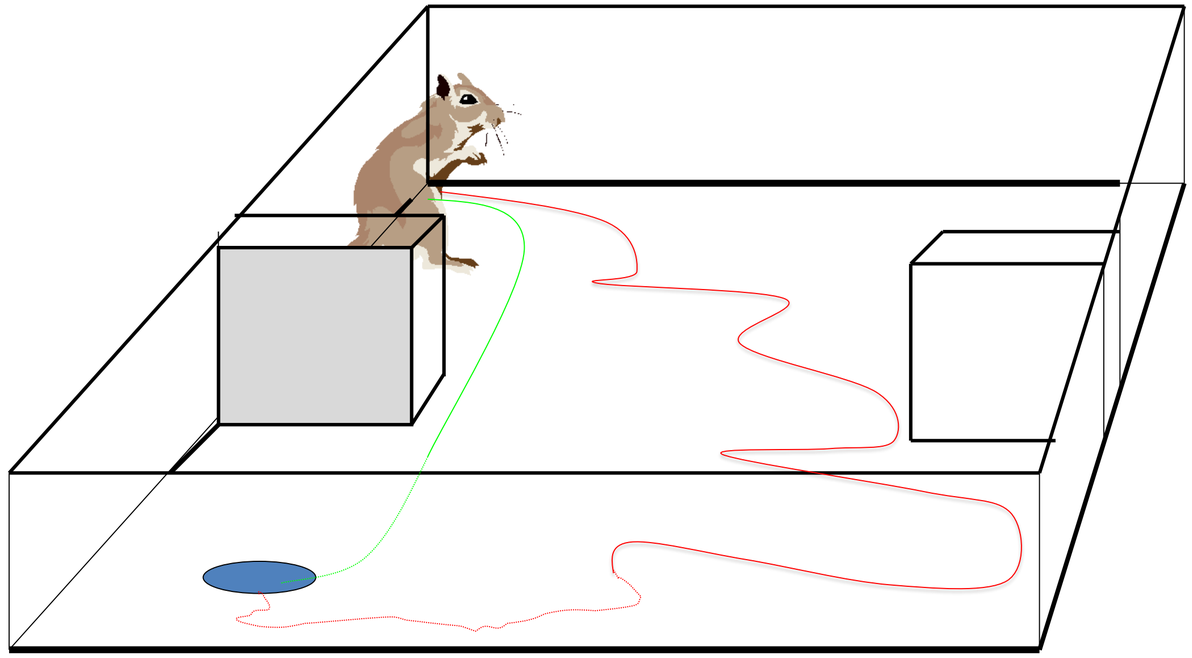
REPETITION: Elements of Reinforcement Learning:

-states
-actions
-rewards

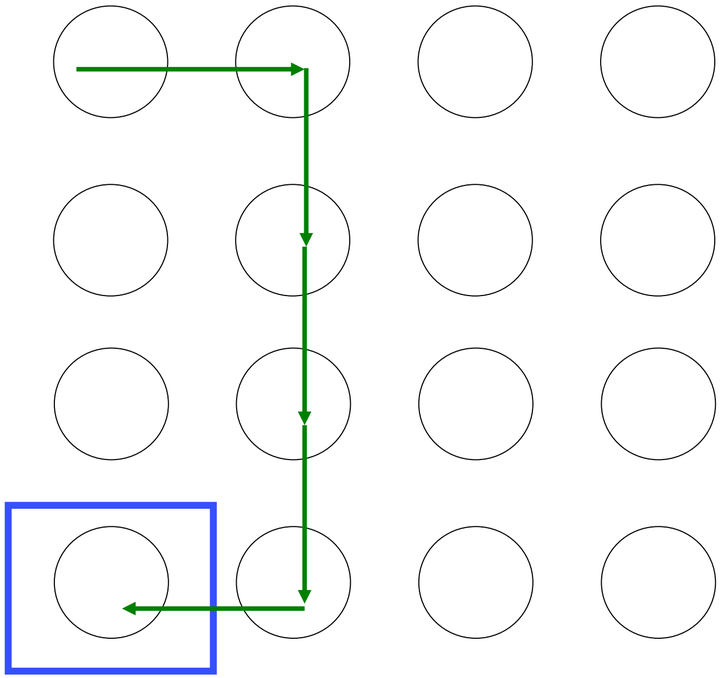
Previous slide.

Reinforcement learning needs states, actions, and rewards.

Elements of Reinforcement Learning:



- discrete states
- discrete actions
- sparse rewards



Previous slide (already shown before the break)

.

Note that, for standard formulations of Reinforcement Learning Theories this (normally) implies discretizing space and actions.

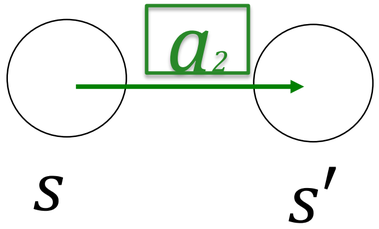
We will study continuous-space formulations only next week.

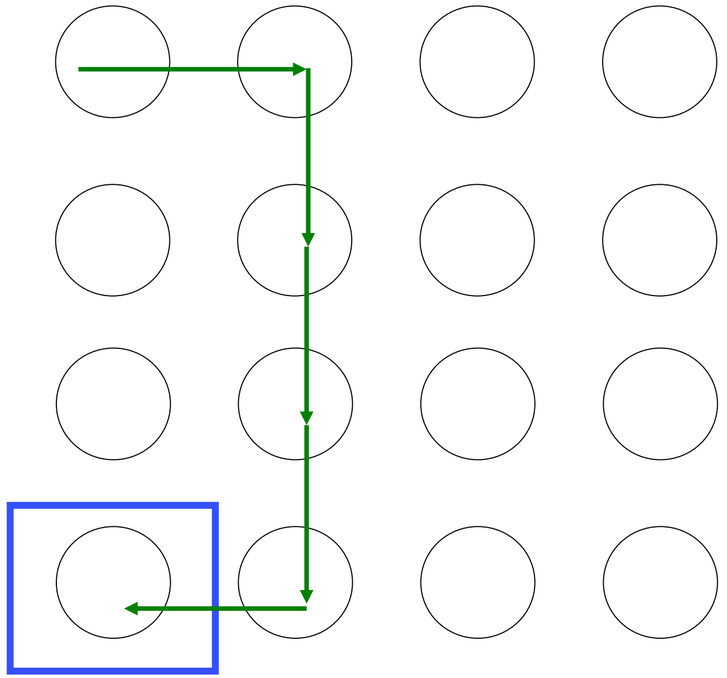
REPETITION: Elements of Reinforcement Learning:

- discrete states:
  - old state s
  - new state s'
- current state: s_t
- discrete actions: $a_1,\ a_2\ \ldots\ a_A$
- current action: a_t
- current reward: r_t
- Mean rewards for transitions:

$$R^{a}_{s \to s'}$$

often most transitions have zero reward





Previous slide.

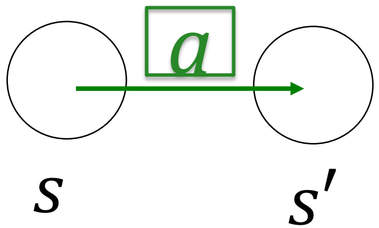
The elementary step is:
The agent starts in state s.
It takes action a
It arrives in a new state s’
Potentially receiving reward r (during the transition or upon arrival at s’).

Since rewards are stochastic we have to distinguish the mean reward at the transition (capital R with indices identifying the transition) from the actual reward (lower-case r with index t) that is received at time t on a transition.

Note that in many practical situations most transitions or states have zero rewards, except a single ‘goal’ state at the end.

REPETITION: States in Reinforcement Learning:

- discrete states:
  - starting state s
  - arrival state s'
- current state: s_t



state = current configuration/well-defined situation
= generalized ‘location’ of actor in environment

Previous slide.

What are these discrete states?
Loosely speaking a state is the current configuration that **uniquely** describes the momentary situation. We can think of the generalized ‘location’ of the actor in the environment

To get acquainted with this, let us look at an example.

Reinforcement Learning: Example Acrobot

3 actions: a_1 = no torque,

a_2 = torque +1 at elbow,

a_3 = torque -1 at elbow

States?

→ discretize!

reward if tip above line

Suppose 5 states per dimension, How many states in total?

[ ] 5

[ ] 25

[ ] 125

[ ] 625

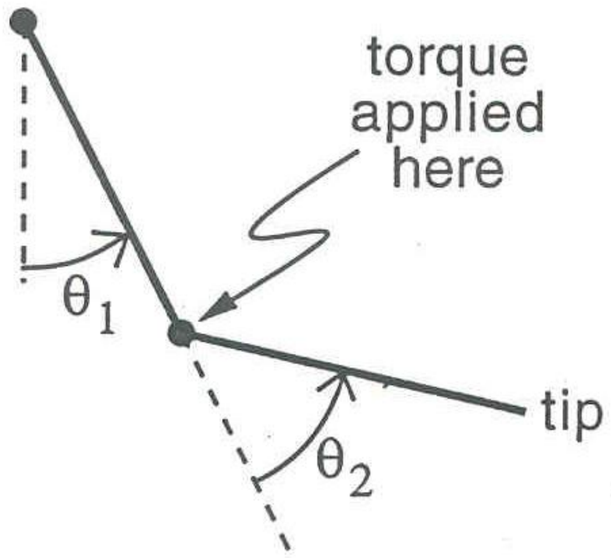


Figure 11.4 The acrobot.

From Book: Sutton and Barto

Previous slide.
The aim of the acrobat is to move the tip above the blue line. To achieve this torque can be applied at the ‘elbow’ link. The second link is the ‘shoulder’.

There are three possible actions.
But what are the states? How many states do we have?

Reinforcement Learning: Example Acrobot

1st episode: long sequence of random actions
400th episode: short sequence of ‘smart’ actions

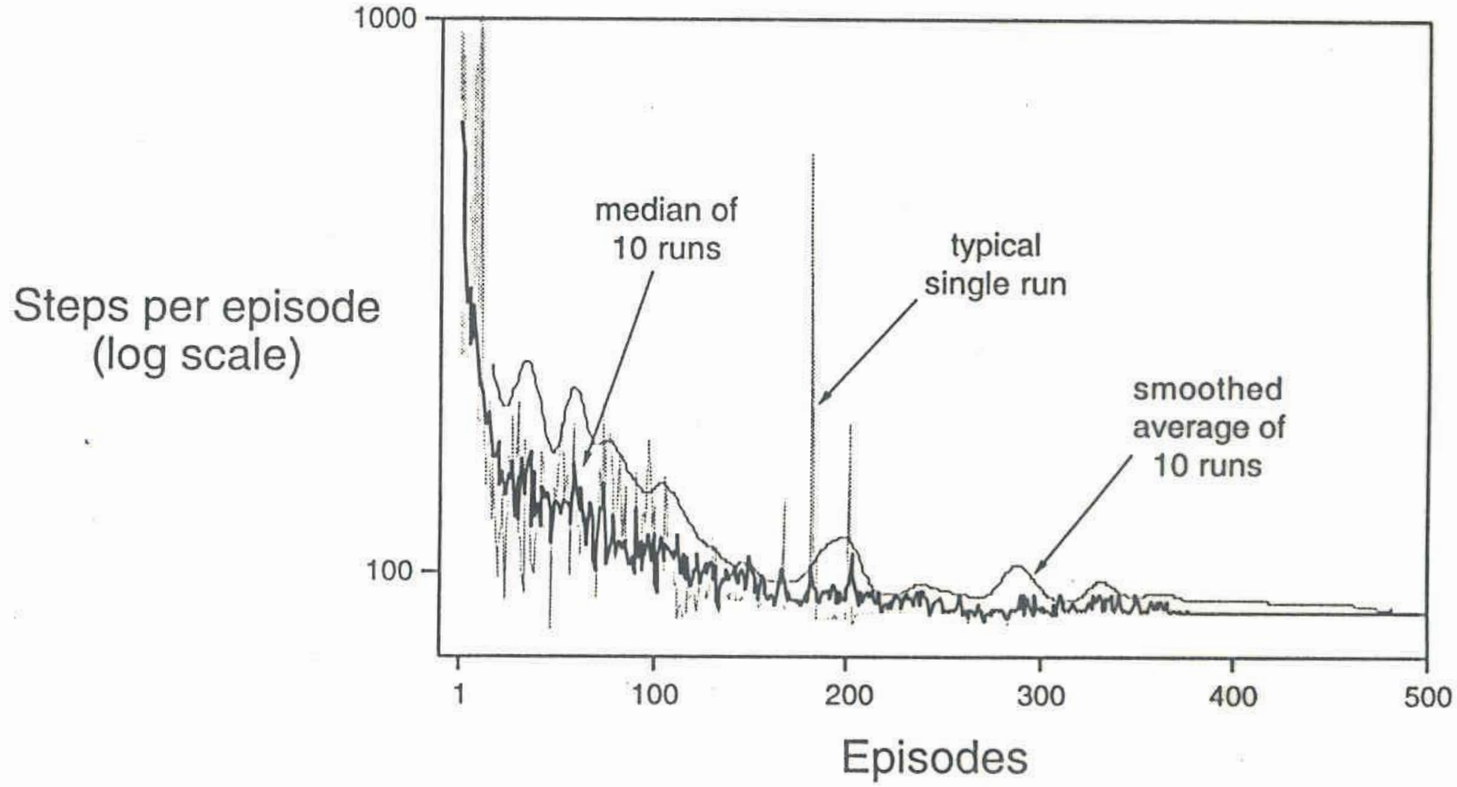


Figure 11.6 Learning curves for Sarsa(λ) on the acrobot task.

From Book:
Sutton and Barto

Previous slide.

An episode finishes if the target is reached. Over time episodes get shorter and shorter indicating that the acrobat has discovered (via reinforcement learning) a smart sequence of actions so as to reach the target (i.e., move the tip above the reference line)

Reinforcement Learning: Example Acrobot

after 400 episodes

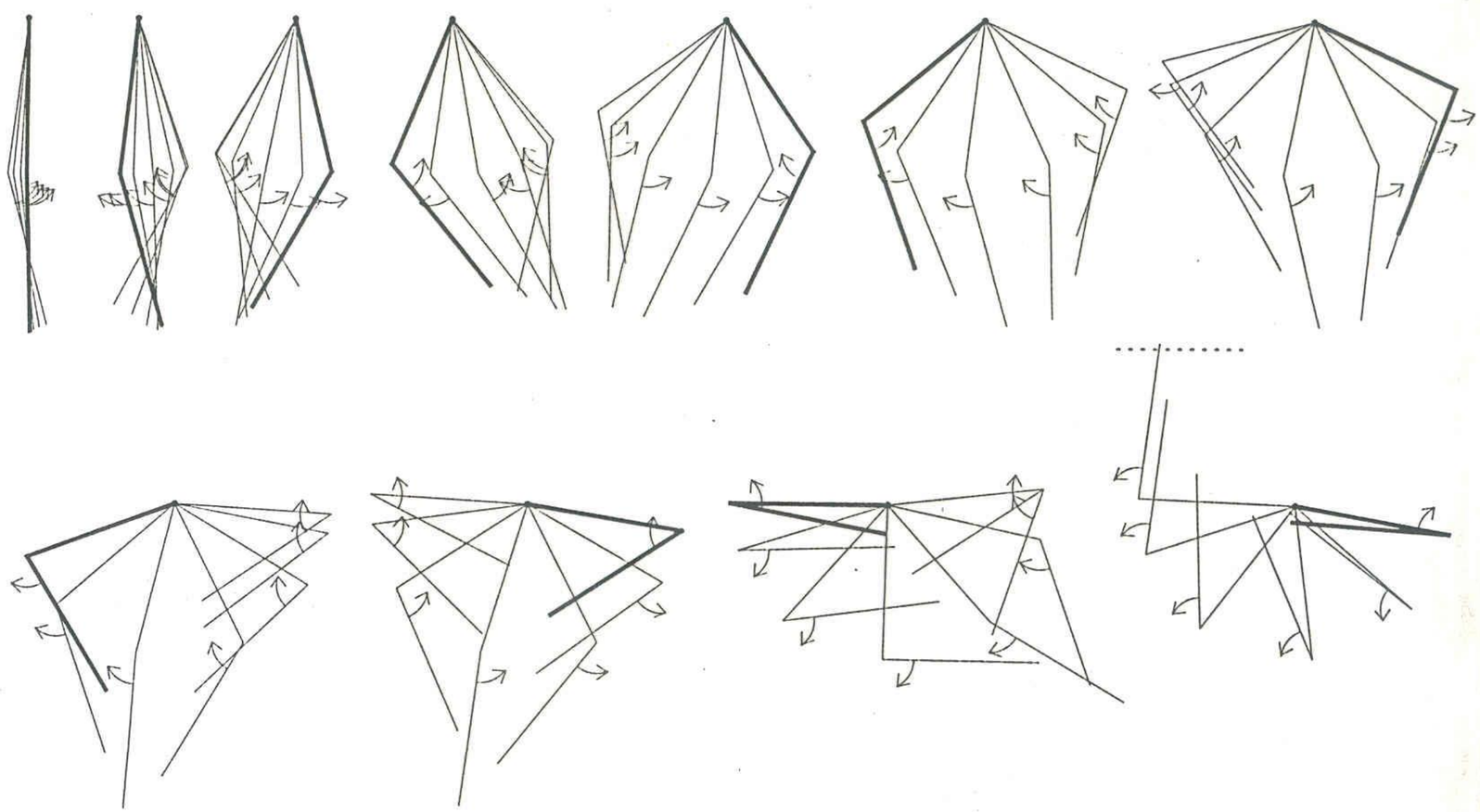


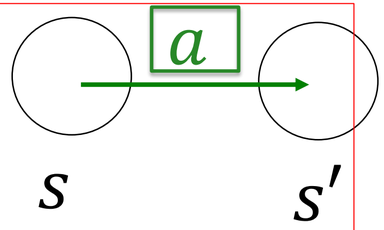
Figure 11.7 A typical learned behavior of the acrobot. Each group is a series of consecutive positions, the thicker line being the first. The arrow indicates the torque applied at the second joint.

From Book:
Sutton and Barto

Previous slide.
One example of an action sequence, after learning, is shown.

Summary: Elements of Reinforcement Learning

There can be MANY states
Often need to discretize first
(→ later we try to model in continuum)



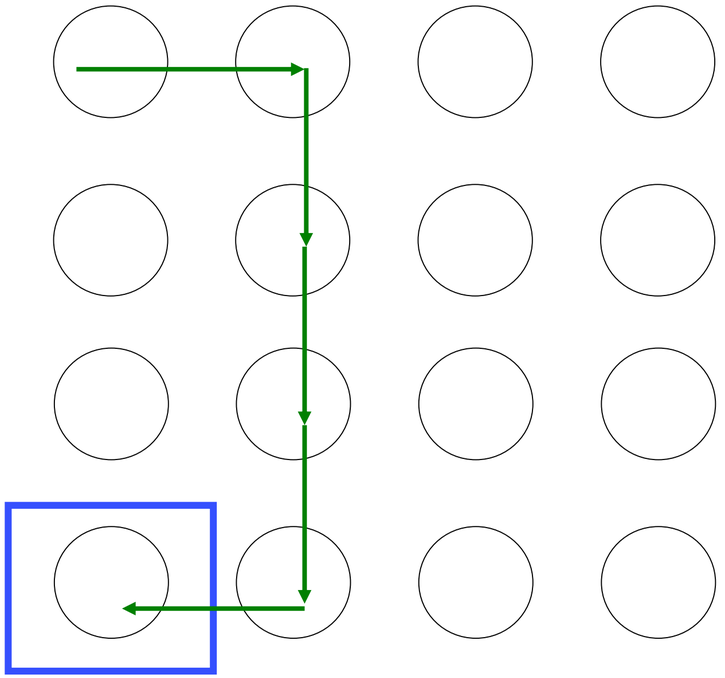
- discrete actions: a

- Mean reward for transition:

$$R^{a}_{s\to s'} = E\left(r\middle|s,a,s'\right)$$

- current actual reward: r_t

often most transitions have zero reward



Previous slide.
Conclusion: In all practical situations, there is an enormous number of states.

In many situations we can think of the actions as discrete. For the moment we also think of the states as discrete (but next week we will go to continuous state space)

Quiz: Reinforcement Learning for backgammon

Case Studies

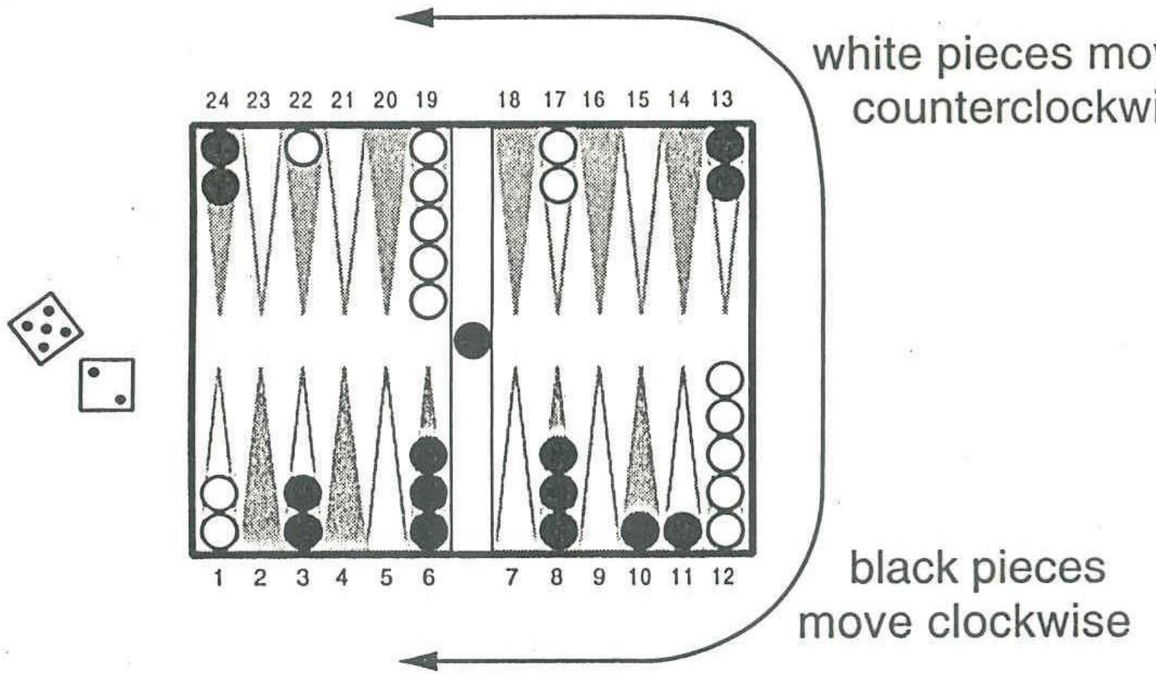


Figure 11.1 A backgammon position.

Game position = discrete states!

Suppose 2 pieces per player, How many states in total?

[ ] 100<n<500

[ ] 500<n<5000

[ ] 5 000<n<50 000

[ ] n>50 000

From Book:
Sutton and Barto

Previous slide.

Backgammon game. There are 24 fields on the board. Players have several pieces. Pieces are protected if there are two of the same color on the same field.

To make it simply, we now consider that both players have two pieces each left. How many different states are there in total?

Reinforcement Learning Lecture 1
Reinforcement Learning and SARSA

Wulfram Gerstner
EPFL, Lausanne, Switzerland

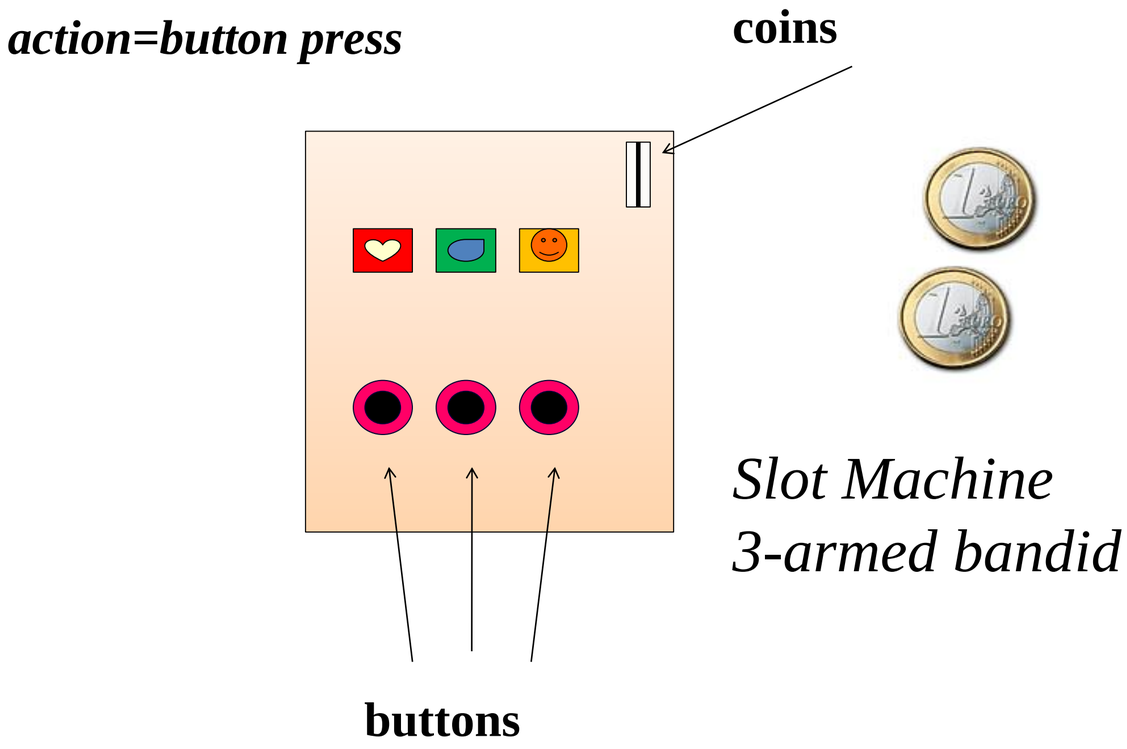
Part 3: One-step horizon (bandit problems)

- Examples of Reward-based Learning
- Elements of Reinforcement Learning
- **One-step horizon (bandit problems)**

Previous slide.

We start with the simplest discrete example: the game is over and reward is given after a single step.

One-step horizon games (bandit)

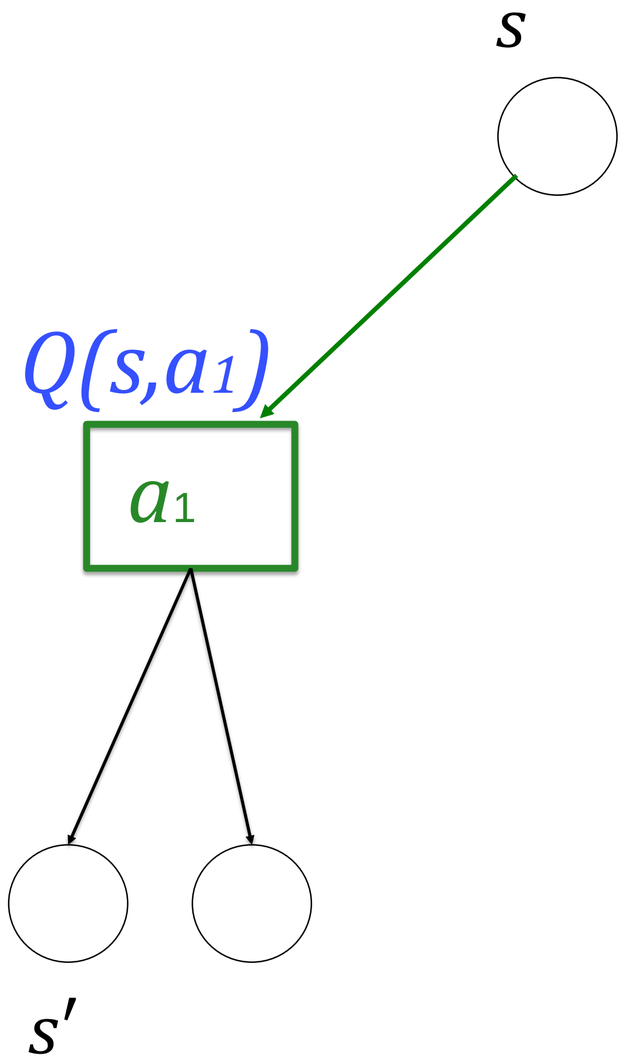


Slot Machine
3-armed bandid

Previous slide.
The standard example is a multi-armed bandit, or slot machine: you have to choose between a few actions, and once you have pressed the button you can just wait and see whether you get reward or not.

One-step horizon games

Q-value: $Q(s,a)$
Expected reward for action a starting from s



Blackboard1:
Q-values

Previous slide.

One of the most central notion in reinforcement learning is the Q-value.
Q(s,a) has two indices: you start in state s and take action a.
The Q-value Q(s,a) is (an estimate of) the mean expected reward that you will get if you take action a starting from state s.

One-step horizon games

Blackboard1:
Q-values

Your notes.

One-step horizon games: Q-value

Q-value $Q(s,a)$

Expected reward for action a starting from s

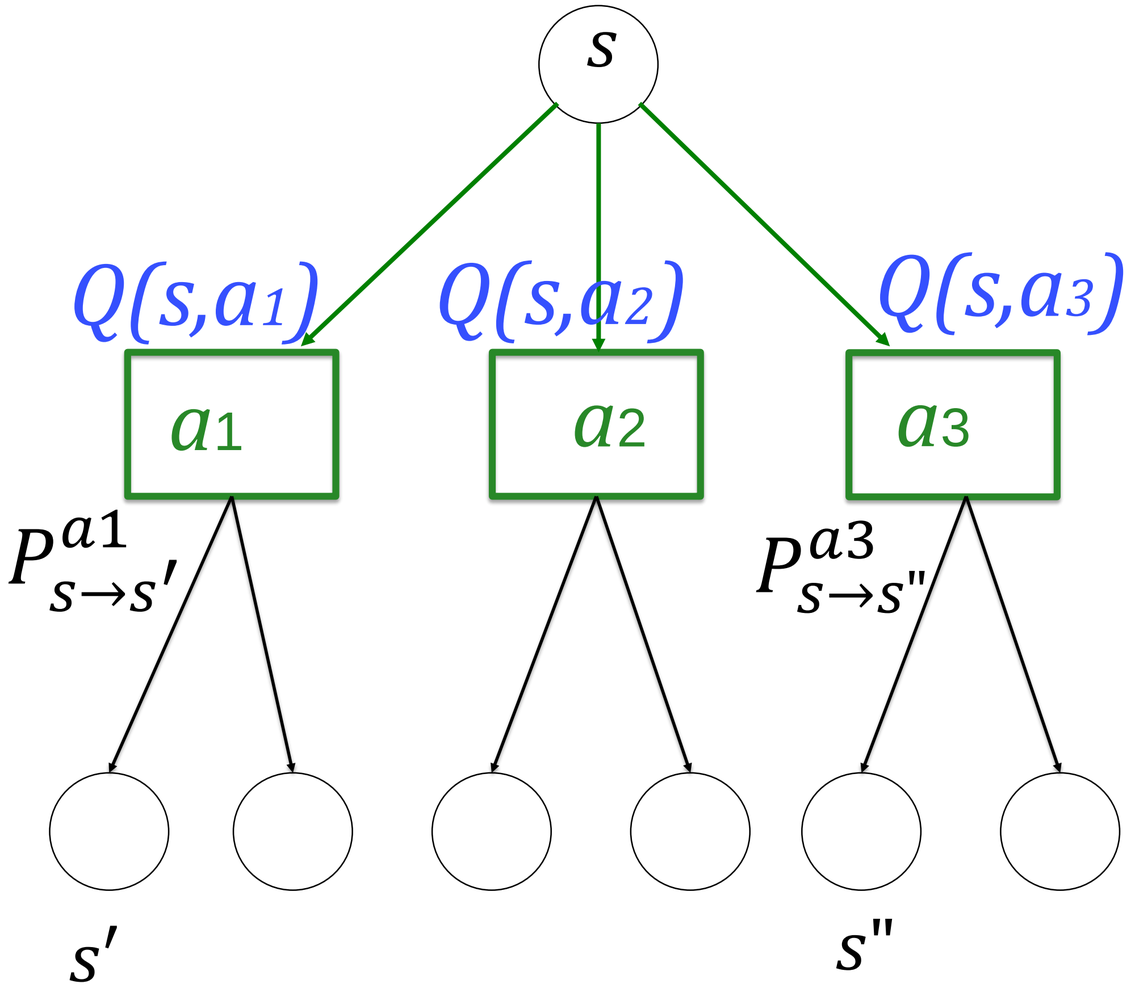
$$Q(s,a) = \sum_{s'} P^{a}_{s\to s'} R^{a}_{s\to s'}$$

Reminder:

$$R^{a}_{s\to s'} = E\left(r \middle| s', a, s\right)$$

Similarly:

$$Q(s,a) = E(r|s,a)$$



Now we know the Q-values: which action should you choose?

Previous slide.

$P_{s \to s'}^{a1}$ is the probability that you end up in a specific state s' if you take action a1 in state s.
We refer to this sometimes as the 'branching ratio' below the 'actions'.

Q(s,a) is attached to the branches linking the state s with the actions.

actions are indicated by green boxes; states are indicated by black circles.

The mean reward $R_{s \to s'}^{a}$ is defined as the expected reward given that you start in state s with action a and end up in state s' (see Blackboard 1).

Given the branching ratio and the mean rewards, it is easy to calculate the Q-values (Blackboard 1).

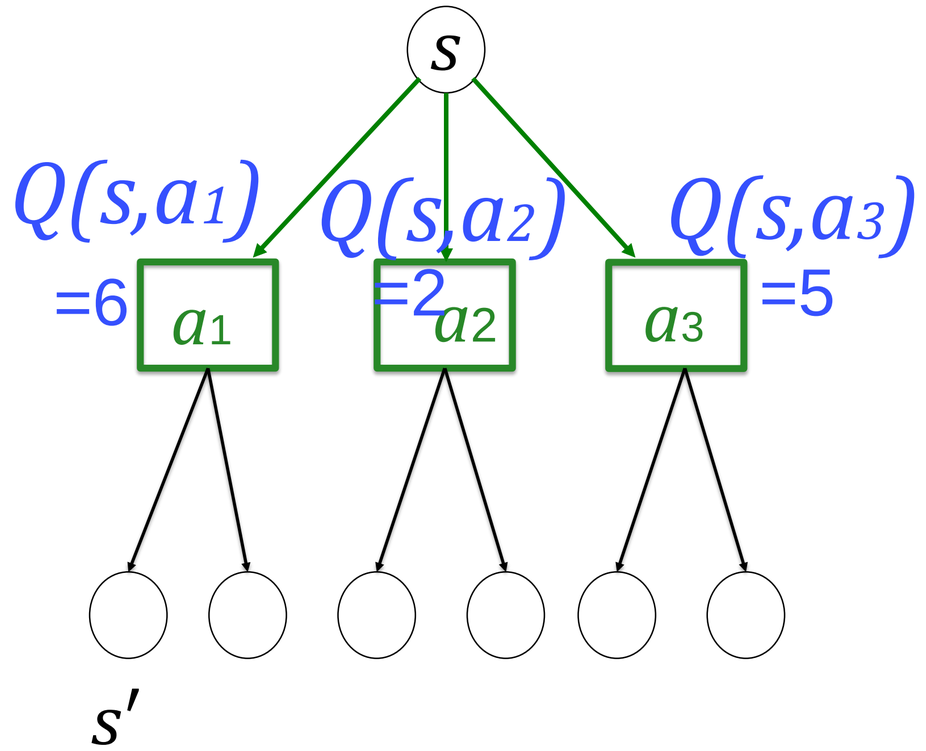
Optimal policy (greedy)

Suppose all Q-values are known:

take action a with*

$$Q(s,a^*) \geq Q(s,a_j)$$

other actions



optimal action:

$a^* = \text{argmax}_a\, [Q(s,a)]$

Optimal policy is also called ‘greedy policy’

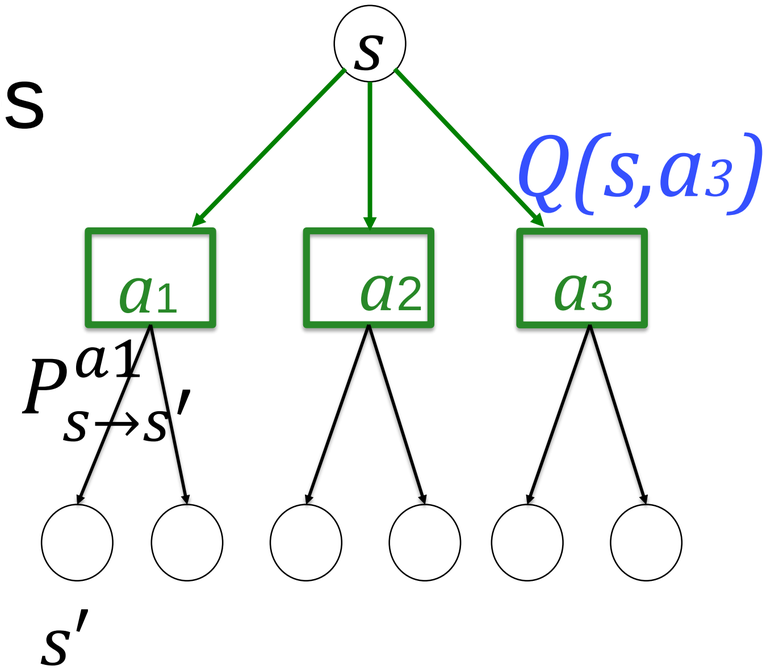
Previous slide.
And once you have the Q-values it is easy to choose the optimal action:
Just take the one with maximal Q-value.

One-step horizon games

Q-value = expected reward for state-action pair

If Q-value is known, choice of action is simple
→ take action with highest Q-value

BUT: we normally do not know the Q-values
→ estimate by trial and error



Previous slide.
The only remaining problem is that we do not know the Q-values, because the casino gives you neither the branching ratio nor the reward scheme.

Hence the only way to find out is by trial and error (that is, by playing many times – the casino will love this!).

Teaching monitoring – monitoring of understanding

[ ] today, up to here, at least 60% of material was new to me.

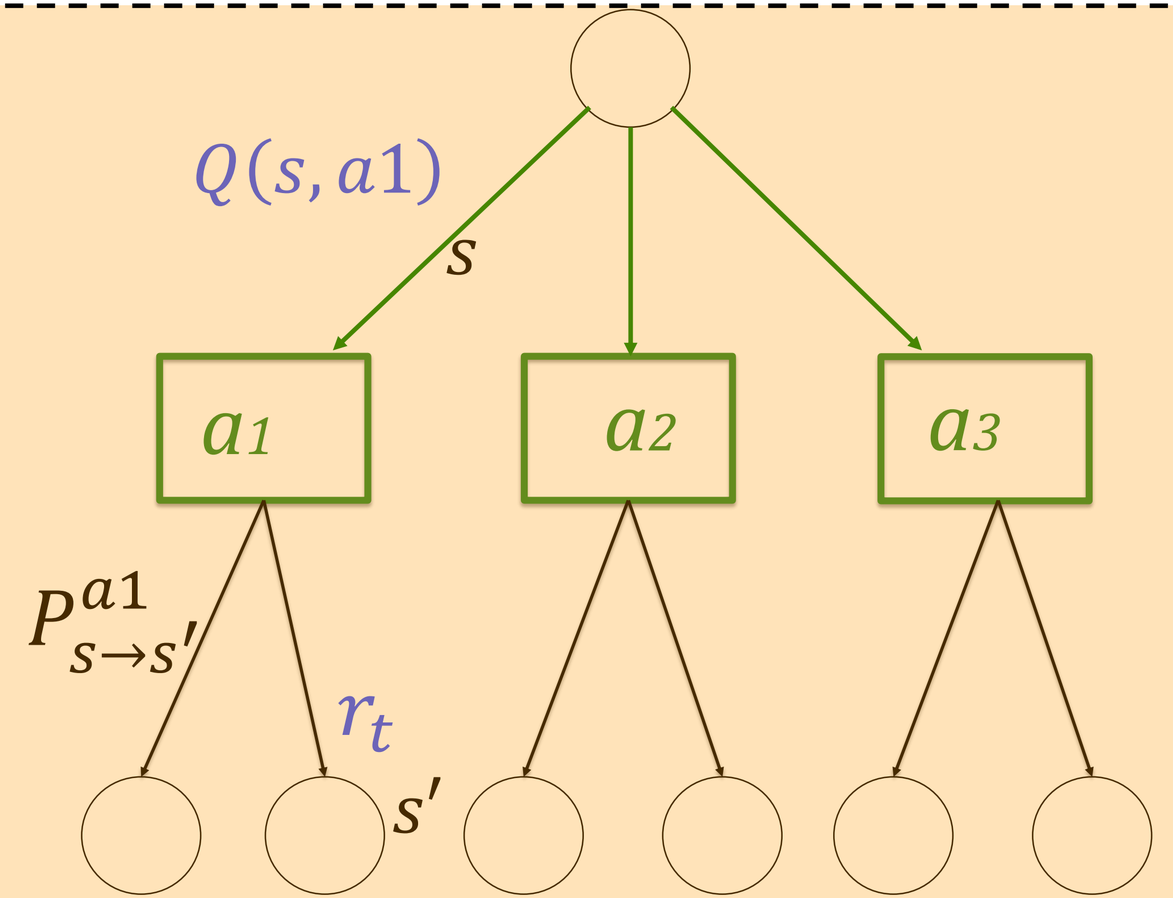
[ ] up to here, I have the feeling that I have been able to follow (at least) 80% of the lecture.

Previous slide.
Teaching monitoring – feedback for the teacher.

Exercise 1 (from earlier session today)

Expected reward

$$Q(s,a) = \sum_{s'} P^{a}_{s \to s'} R^{a}_{s \to s'}$$



Show that empirical averaging over k trials gives an update rule

$$\Delta Q(s,a) = \eta\,[r_t - Q(s,a)]$$

Exercise 1 (in class)

Exercise 1. Iterative update (in class)

We consider an empirical evaluation of $Q(s, a)$ by averaging the rewards for action a over the first k trials:

$$Q_k = \frac{1}{k} \sum_{i=1}^{k} r_i.$$

We now include an additional trial and average over all $k + 1$ trials.

a. Show that this procedure leads to an iterative update rule of the form

$$\Delta Q_k = \eta(r_k - Q_{k-1}),$$

(assuming $Q_0 = 0$).

b. What is the value of η?

c. Give an intuitive explanation of the update rule. *Hint: Think of the following: If the actual reward is larger then my estimate, then I should ...*

Blackboard2: Exercise 1

One-step horizon: Proposition

Q-value = expected reward for state-action pair

If Q-value is known, choice of action is simple

→ take action with highest Q-value

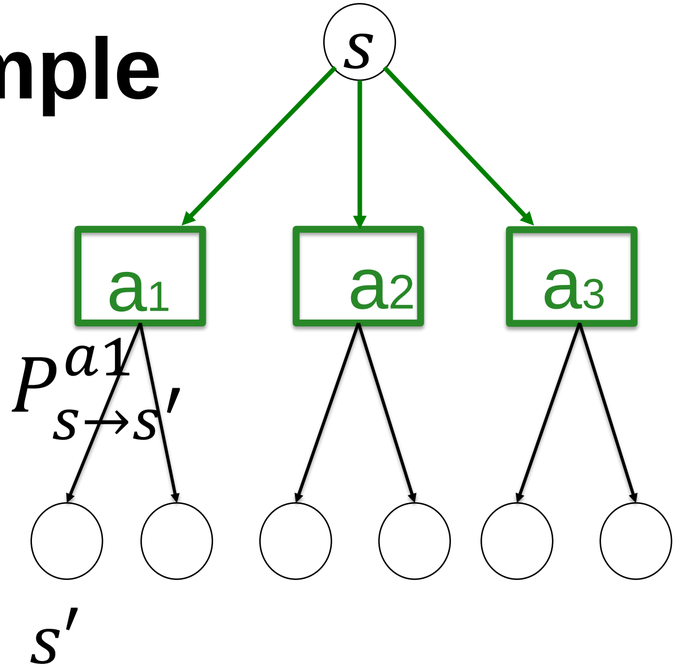
If Q-value not known:

→ estimate $\hat{Q}$ by trial and error

→ update with rule

$$\Delta\hat{Q}(s,a) = \eta\,[r_t - \hat{Q}(s,a)\,] \qquad \textbf{(1)}$$

→Let learning rate η decrease over time



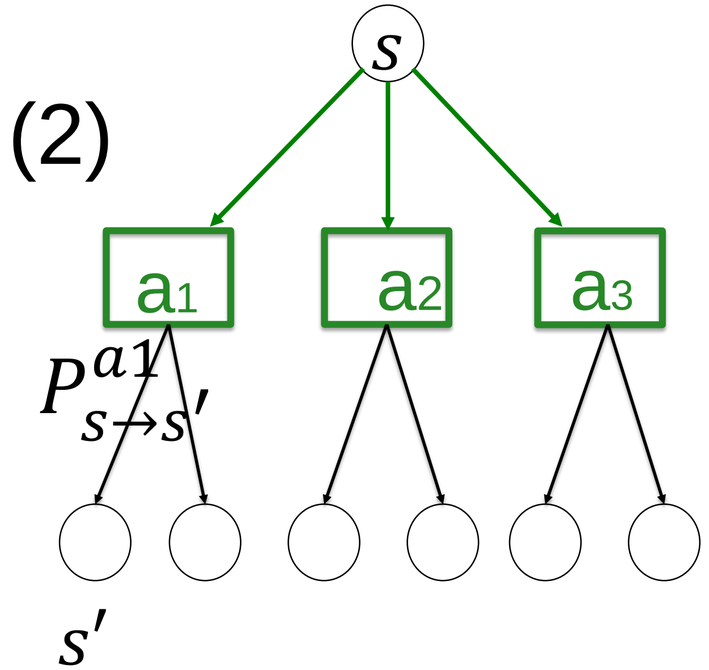
Convergence in Expectation

After taking action a in state s, we update with

$$\Delta\hat{Q}(s,a) = \eta\,[r_t - \hat{Q}(s,a)] \qquad (1)$$

(i) If (1) has **converged in expectation given (s,a),** then $\hat{Q}(s,a)$ has a value,

$$\widehat{Q}(s,a) = E\,[\widehat{Q}(s,a)|s,a] = Q(s,a) = \sum_{s'} P^{a}_{s\to s'}\, R^{a}_{s\to s'} \qquad (2)$$



(ii) If the learning rate η decreases, fluctuations around the **empirical mean** $\langle \widehat{\boldsymbol{Q}}(\boldsymbol{s},\boldsymbol{a})\rangle_{\boldsymbol{t}|\boldsymbol{s},\boldsymbol{a}}$ decrease. If $\langle \widehat{\boldsymbol{Q}}(\boldsymbol{s},\boldsymbol{a})\rangle_{\boldsymbol{t}|\boldsymbol{s},\boldsymbol{a}}$ converges for fixed η, then **the empirical mean approaches** $\boldsymbol{Q}(\boldsymbol{s},\boldsymbol{a})$.

Previous slide.

When evaluating the **expectation value given (s,a),** the learning rate drops out since we set the left-hand-side to zero. The exact value of η is not relevant, as discussed in the theorem. Part (i) of the theorem states that the expectation value of $\hat{Q}(s,a)$ is the correct Q-value. For a quick proof of $E\,[\hat{Q}(s,a)|s,a] = Q(s,a)$ see the video. On the blackboard a stronger statement was shown:

$$\hat{Q}(s,a) = Q(s,a).$$

Convergence in expectation is equivalent to imagining that you start millions of trials with the same value $\hat{Q}(s,a)$ without any intermediate update. So in that sense it is like an infinite 'batch' of examples. The stochastic variables are the **next** state s' and the received reward r_t. The value of $\hat{Q}(s,a)$ is not stochastic but 'frozen'. Therefore (trivially) $E\,[\hat{Q}(s,a)|s,a] = \hat{Q}(s,a)$.

In practice, we do not have expectations but online updates with fluctuations. It is important that η is small at the end of learning so as to limit the amount of fluctuations. Part (ii) states that **online mean** for small learning rate also goes to the correct Q-value.

Indeed, since the equations are linear (for the bandit problem = 1-step horizon), the calculation of part (i) apply analogously to the long-term empirical temporal average (denoted by angular brackets). The average is across all those time steps where action a was chosen in state s, denoted as $\langle \hat{\boldsymbol{Q}}(\boldsymbol{s},\boldsymbol{a})\rangle_{\boldsymbol{t}|\boldsymbol{s},\boldsymbol{a}}$. We assume convergence, hence our hypothesis reads

$$\langle \Delta\hat{\boldsymbol{Q}}(\boldsymbol{s},\boldsymbol{a})\rangle_{\boldsymbol{t}|\boldsymbol{s},\boldsymbol{a}} = \boldsymbol{\eta}\,\langle r_t - \hat{\boldsymbol{Q}}(\boldsymbol{s},\boldsymbol{a})\rangle_{\boldsymbol{t}|\boldsymbol{s},\boldsymbol{a}} = 0\ .$$

The specific result $\langle \hat{\boldsymbol{Q}}(\boldsymbol{s},\boldsymbol{a})\rangle_{\boldsymbol{t}|\boldsymbol{s},\boldsymbol{a}} = \boldsymbol{Q}(\boldsymbol{s},\boldsymbol{a})$ is based on linearity and is not true for the multi-step horizon that we discuss later.

Proof: Convergence in Expectation

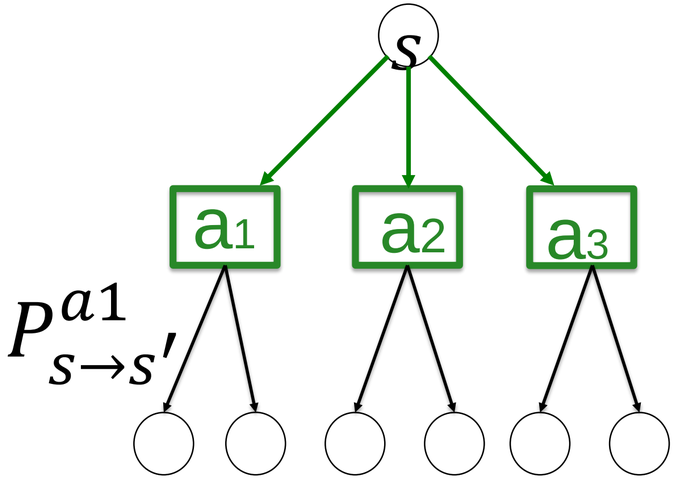
Blackboard3: Proof of (i)

After taking action a in state s, we update with

$$\Delta\hat{Q}(s,a) = \eta\,[r_t - \hat{Q}(s,a)\,] \qquad (1)$$

(i) If (1) has converged in expectation, then $\hat{Q}(s,a)$ has an expectation value,

$$E\left[\widehat{Q}(s,a)\right] = \widehat{Q}(s,a) = \sum_{s'} P^{a}_{s\to s'}\, R^{a}_{s\to s'} = Q(s,a) \qquad (2)$$



Note: the expectation is over all possible ‘futures’. For the bandit problem the future is defined by the possible next states and possible rewards.

Your notes.

Blackboard3

Part (i) of Theorem
converged in expectation → $E(\Delta\hat{Q}(s,a)|s,a)=0$

expectation of all possible futures with correct statistical weight

we always start in (s,a) while the system is frozen

Perspective similar to a batch mode:
update only **after** (infinitely) many trials that all start in (s,a) with the same value $\hat{Q}(s,a)$
=
update the expectation over all possibilities that may occur in the next time step.

Previous slide:
$\hat{Q}(s,a)$ denotes the current estimate of the Q-value. Claim: If Q no longer changes (in expectation) then it must be the correct Q-value.

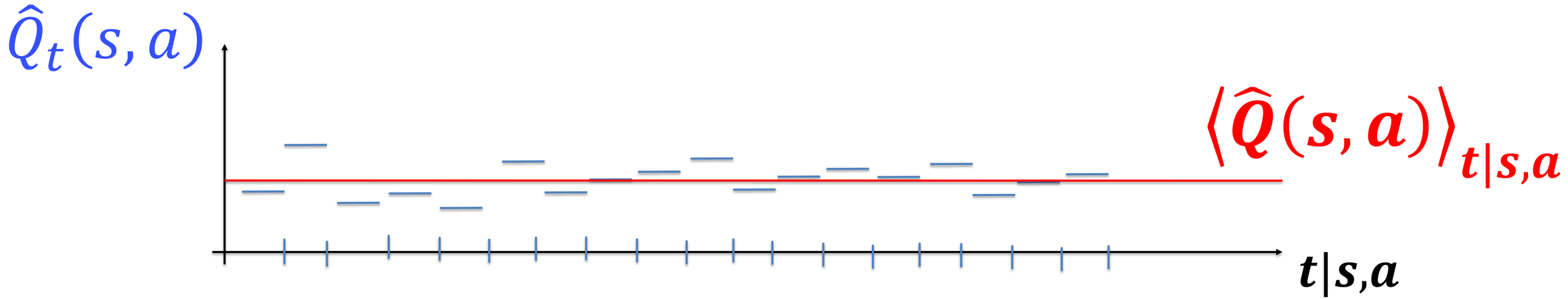
There are different views on how to interpret the ‘expectation;:
- Formally from a mathematical point of view: average over all possible outcomes of the next time step given (s,a).
- In a simulation this would correspond to the following sampling procedure:
You freeze the value of $\hat{Q}(s,a)$ and run MANY times (N to infinity) a test with the state-action pair (s,a) as a starting condition. Then you evaluate the resulting ‘batch update’ averaged across all these examples. If the batch update with millions of examples is zero, that implies that you have converged to the correct value.
In the copies of the blackboard notes, there are two versions of the proof:

First, on page 2, top half of page a SIMPLE proof. $E\left[\widehat{Q}(s,a)|s,a\right] = Q(s,a) = \sum_{s'} P^{a}_{s\to s'}, R^{a}_{s\to s'}$
Second, on page 4 (final page), the stronger proof with more in-between steps showing $\widehat{Q}(s,a) = E\left[\widehat{Q}(s,a)|s,a\right] = Q(s,a) = \sum_{s'} P^{a}_{s\to s'}, R^{a}_{s\to s'}$

Blackboard3

Part (ii) of Theorem:

We work with the online update $\Delta\hat{Q}(s,a)$. With finite learning rate, the value of $\hat{Q}_t(s,a)$ fluctuates around a mean $\langle \hat{\boldsymbol{Q}}(\boldsymbol{s},\boldsymbol{a}) \rangle_{\boldsymbol{t}|\boldsymbol{s},\boldsymbol{a}}$



Under the hypothesis of the theorem (i.e., the mean converges), then the mean is equal to the ‘correct’ Q-value.

Notes.
Proof of part (ii) of the theorem is in the Blackboard notes on page 3 – think about it. The proof works because of linearity.

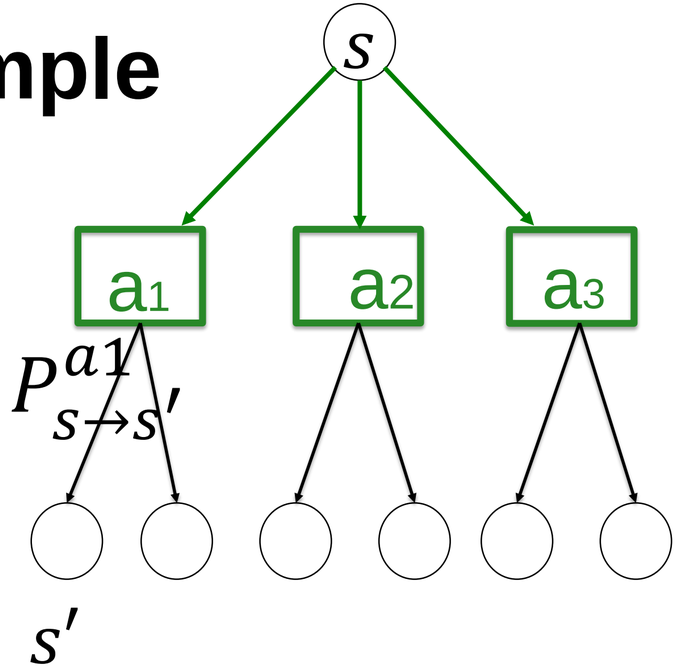
More information regarding the philosophy of different averaging procedures also in Exercise 3 this week and beginning of the lecture of next week.

One-step horizon: summary

Q-value = expected reward for state-action pair

If Q-value is known, choice of action is simple

→ take action with highest Q-value



If Q-value not known:

→ estimate $\hat{Q}$ by trial and error

→ update with rule

$$\Delta\hat{Q}(s,a) = \eta\,[r_t - \hat{Q}(s,a)\,] \qquad \textbf{(1)}$$

→Let learning rate η decrease over time

Iterative algorithm (1) converges in expectation

Previous slide.
Let us distinguish the ESTIMATE $\hat{Q}(s,a)$ from the real Q-value $Q(s,a)$

The update rule can be interpreted as follows:
if the actual reward is larger than (my estimate of) the expected reward, then I should increase (a little bit) my expectations.

The learning rate η :

In exercise 1, we found a rather specific scheme for how to reduce the learning rate over time. But many other schemes also work in practice. For example you keep η constant for a block of time, and then you decrease it for the next block.

Note: in later lectures I will often use the symbol α instead of η

Both symbols indicate what is called the ‘learning rate’ in Deep Learning.

Teaching monitoring – monitoring of understanding

[ ] today, at least 60% of material was new to me.

[ ] I have the feeling that I have been able to follow (at least) 80% of the lecture.

Previous slide.
Teaching monitoring – feedback for the teacher.