Announcements:

Exam (counts 70%): 1. July 2019, 12h20-15h00

- paper and pencil,
- no textbook/no slides/no calculator
- **1 sheet A5, double-sided, handwritten notes.**
- sample exam from last year online
- similar to exercises (and a few quiz questions)

Miniprojects (count 30%).

- miniproject validated after ‘fraud detection interview’
- miniproject 2a/2b (Sequences or RL)
- handout before Easter
- teams of 2 students, individual submission

Artificial Neural Networks: Lecture 8
Reinforcement Learning and SARSA

Wulfram Gerstner
EPFL, Lausanne, Switzerland

Objectives for today:

- Reinforcement Learning is learning by rewards
- Agents and actions
- Exploration vs Exploitation
- Bellman equation
- SARSA algorithm

Reading for this week:

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

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

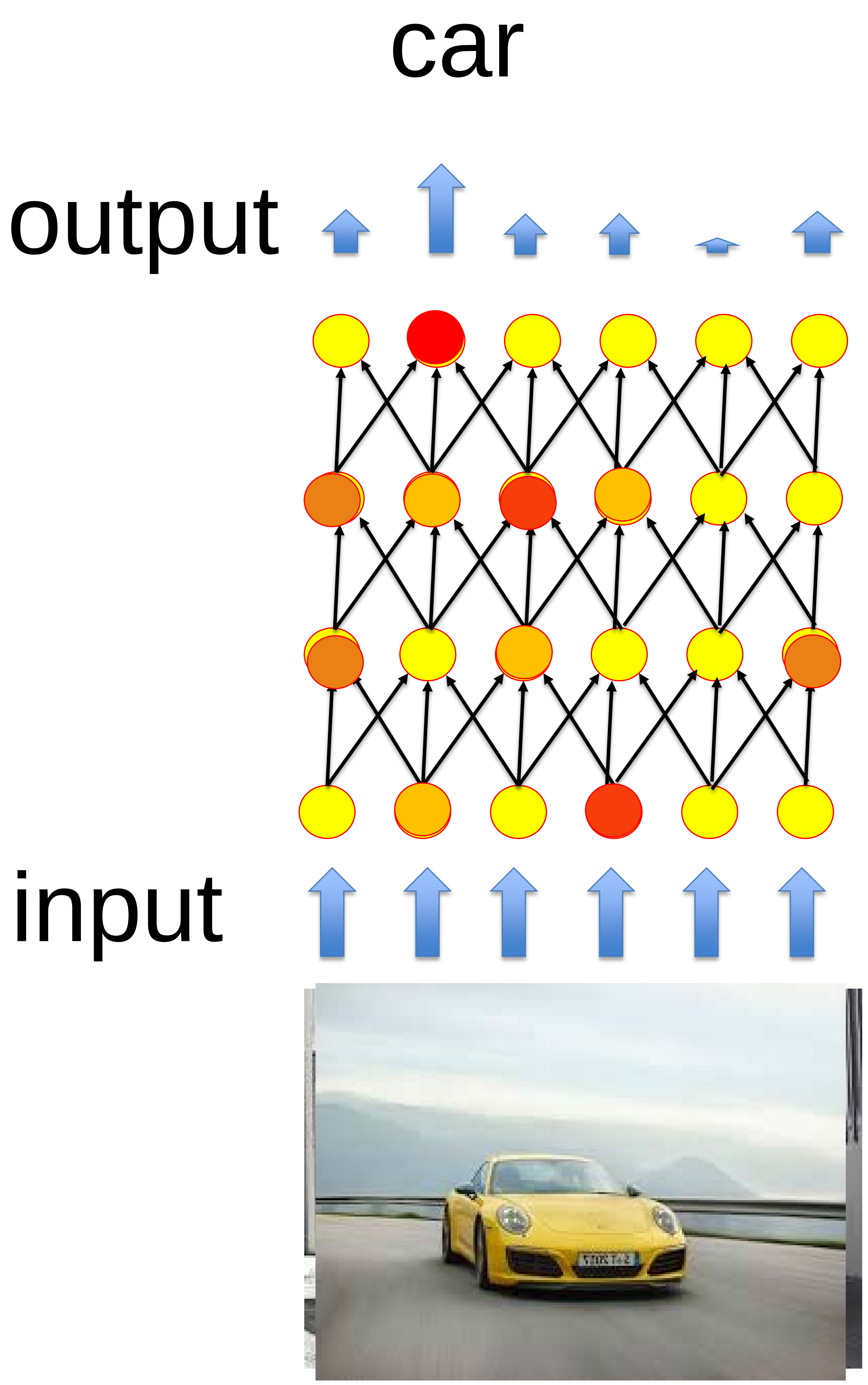
Background reading:

Silver et al. 2017, Archive

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

Review: Artificial Neural Networks for classification

feedforward network



review: Artificial Neural Networks for classification

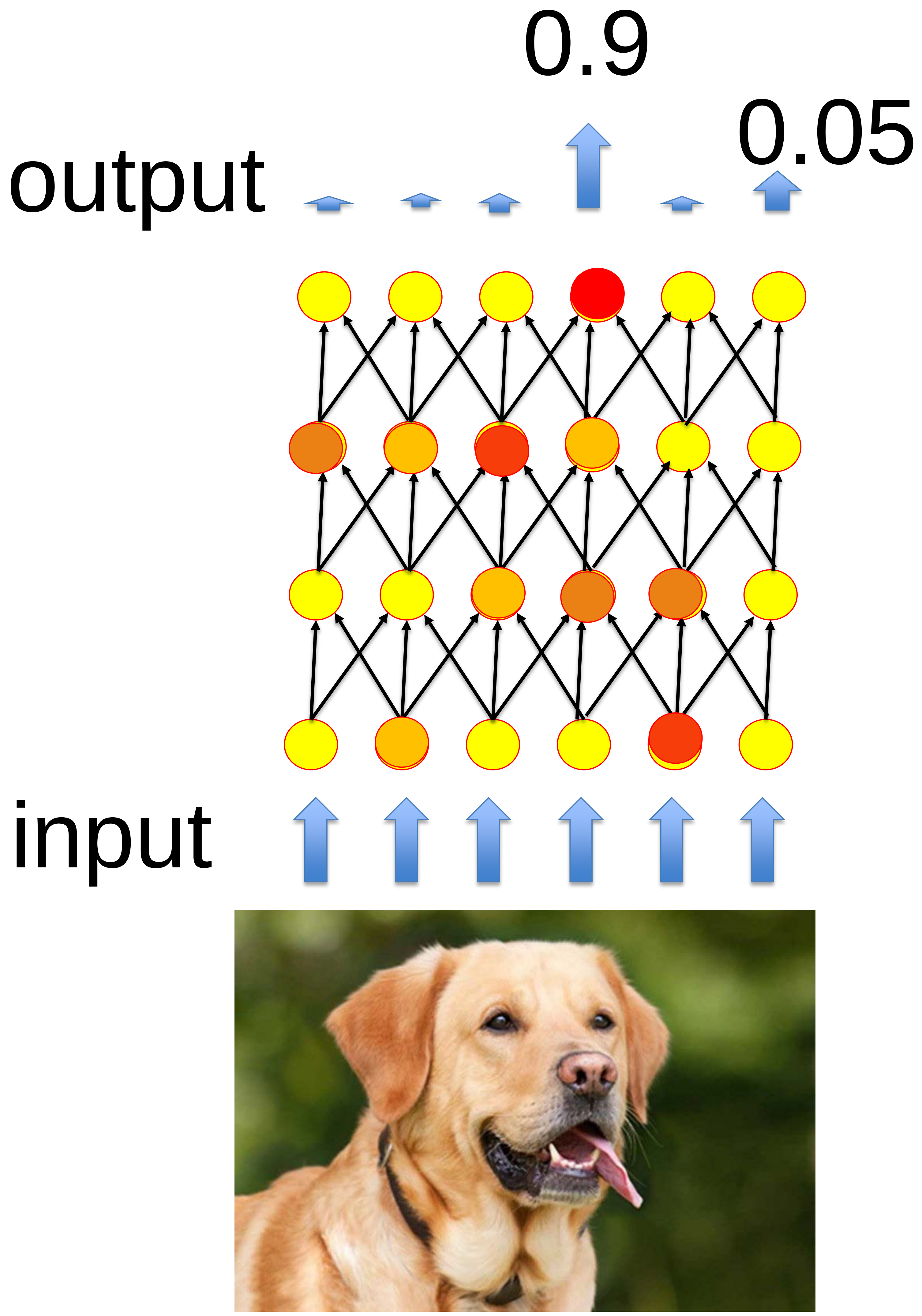
Prerequisite for learning:
labeled data base

$$\{ \quad (\boldsymbol{x}^{\mu}, \boldsymbol{t}^{\mu}) \quad , \qquad 1 \le \mu \le P \quad \};$$

Aim of learning:
Adjust connections such that output $\boldsymbol{y}^{\mu}$ is correct

$$\boldsymbol{y}^{\mu} = \boldsymbol{t}^{\mu}$$

(for each static input image, $\boldsymbol{x}^{\mu}$)



review: Artificial Neural Networks for classification

Prerequisite for learning:

labeled data base

$$\{ \quad (\boldsymbol{x}^{\mu}, \boldsymbol{t}^{\mu}) \quad , \qquad 1 \leq \mu \leq P \quad \};$$

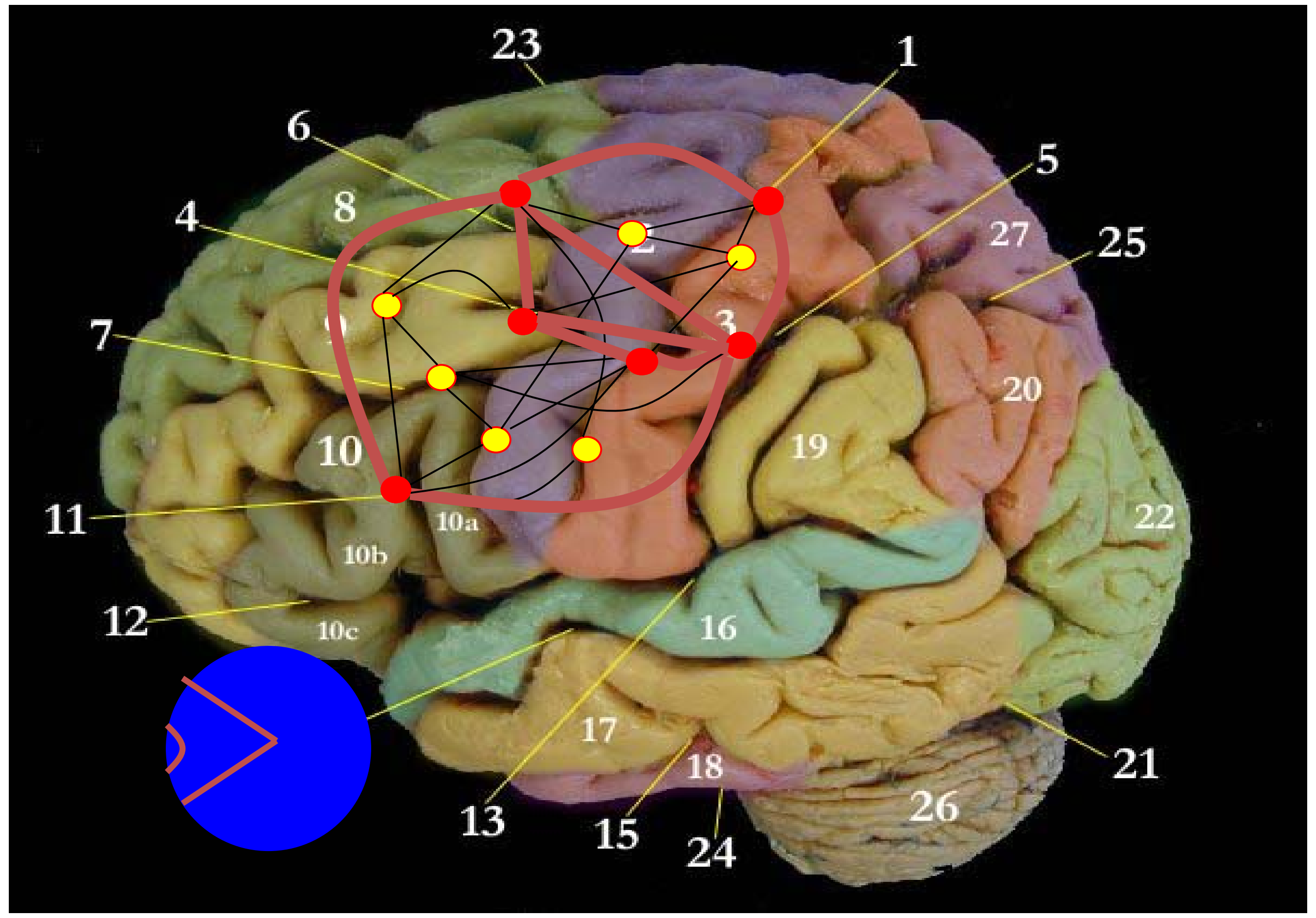
Question: Is this realistic?

1. Artificial Neural Networks for action learning

Where is the supervisor?
Where is the labeled data?

Replaced by:
'Value of action'
- 'goodie' for dog
- 'success'
- 'compliment'

BUT:
Reward is rare:
'sparse feedback' after
a long action sequence



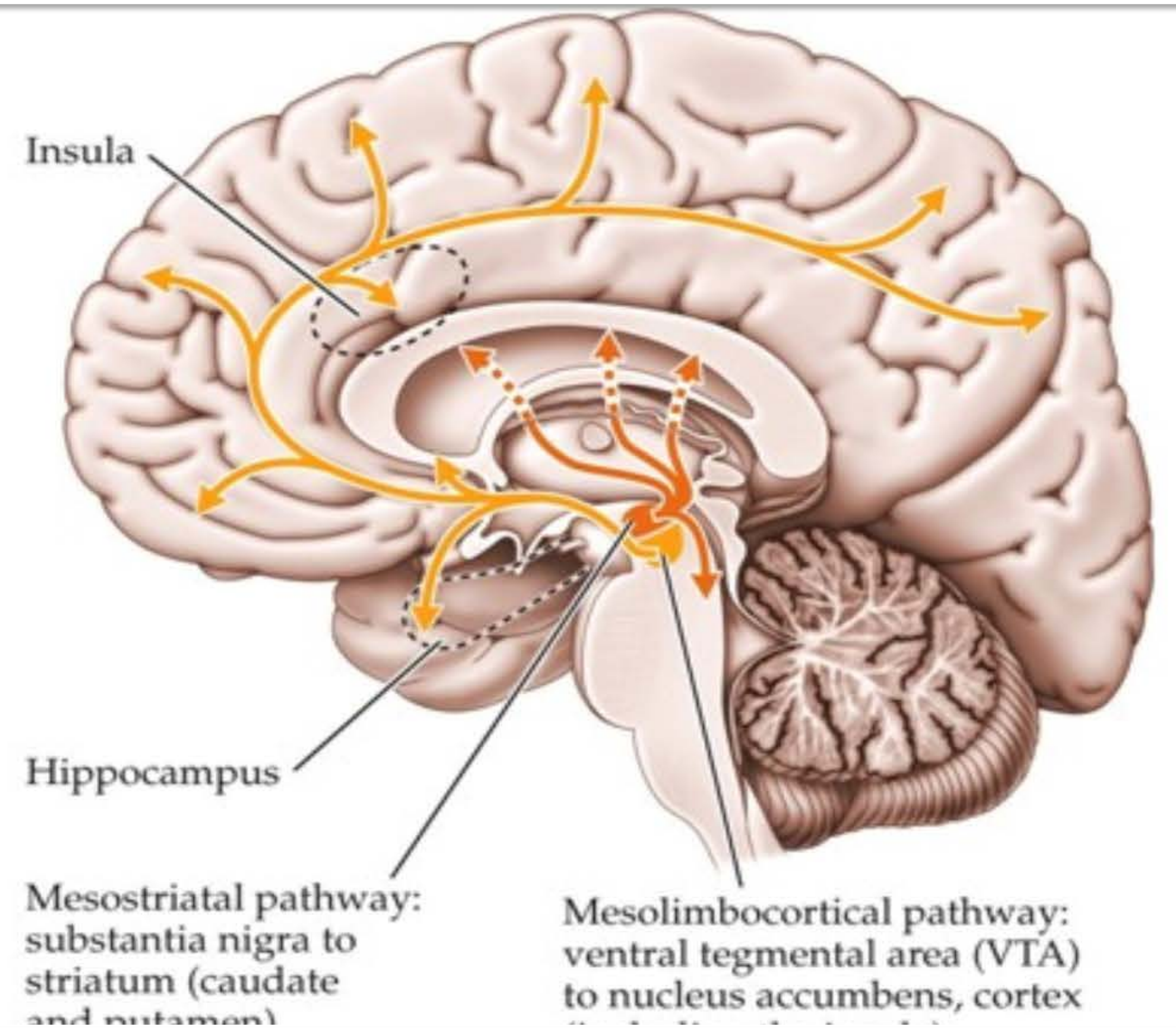
Reward information is available in the brain

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

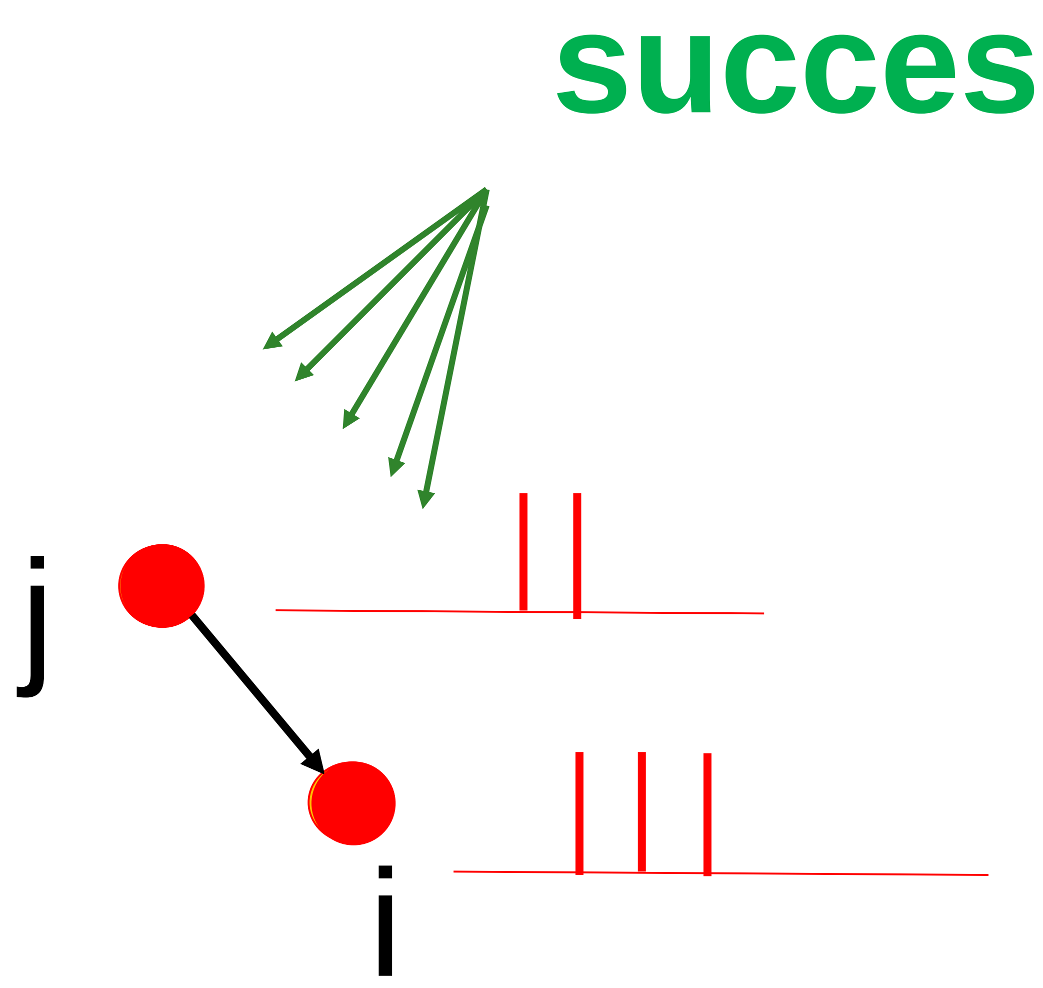
Schultz et al., 1997,
Waelti et al., 2001
Schultz, 2002

‘success signal’

Dopamine



Review: Modeling – the role of reward



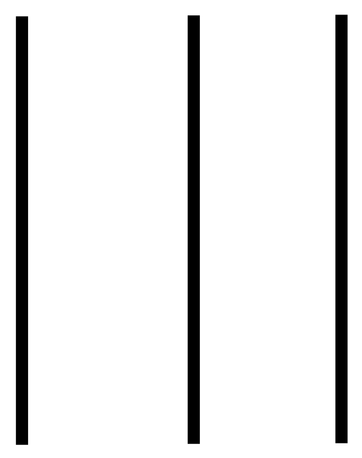
Three factors for changing a connection
- activity of neuron j
- activity of neuron i
- success

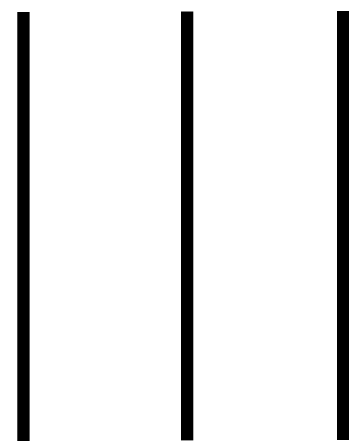
Barto 1985, Schultz et al. 1997; Waelti et al., 2001; Reynolds and Wickens 2002; Lisman et al. 2011

Reinforcement learning = learning based on reward

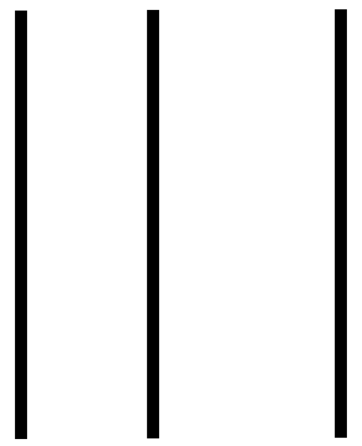
1. Examples of reinforcment learning

Middle bar: shifted left or shifted right?

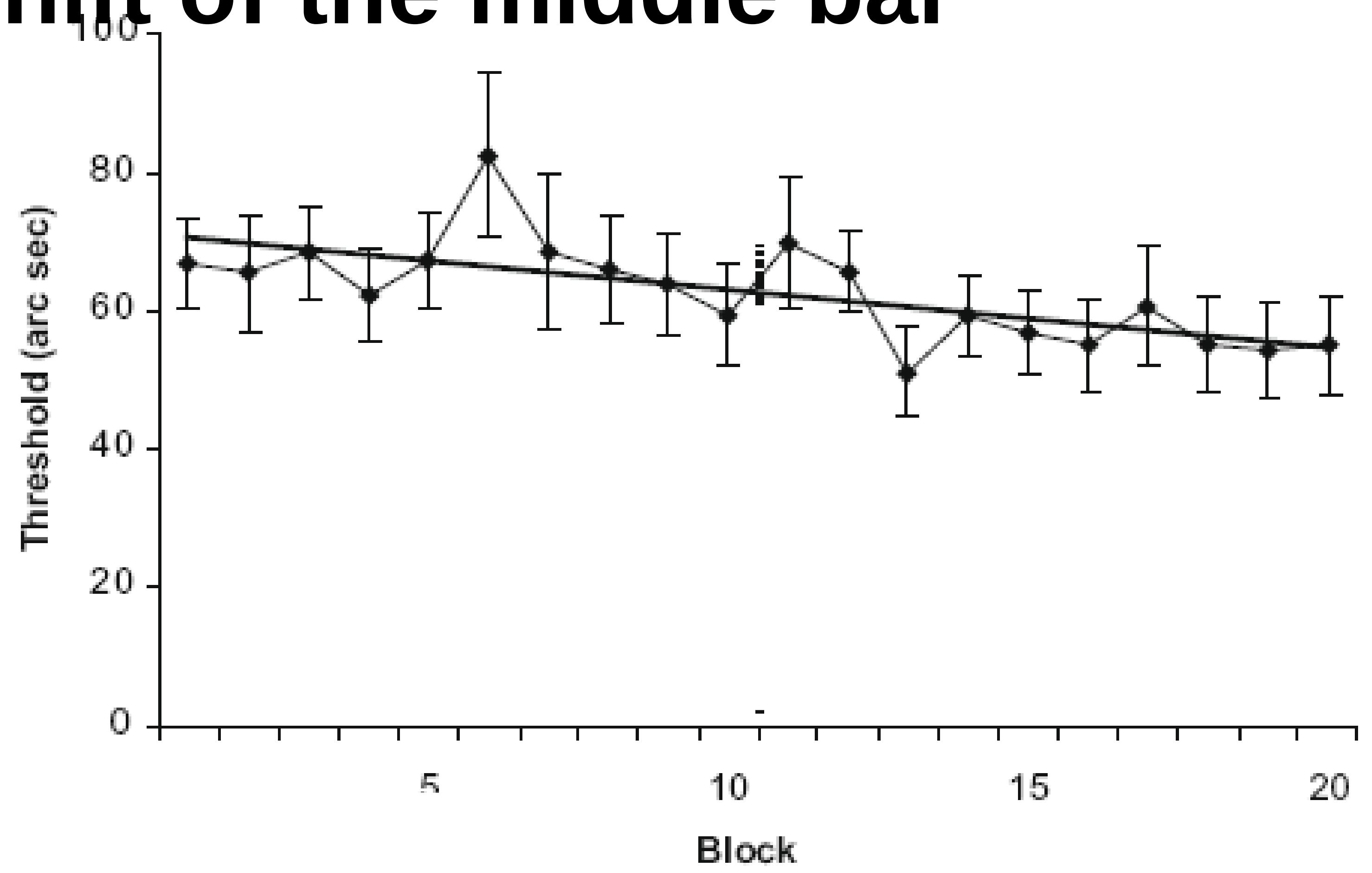




Observers get better at seeing the shift of the middle bar



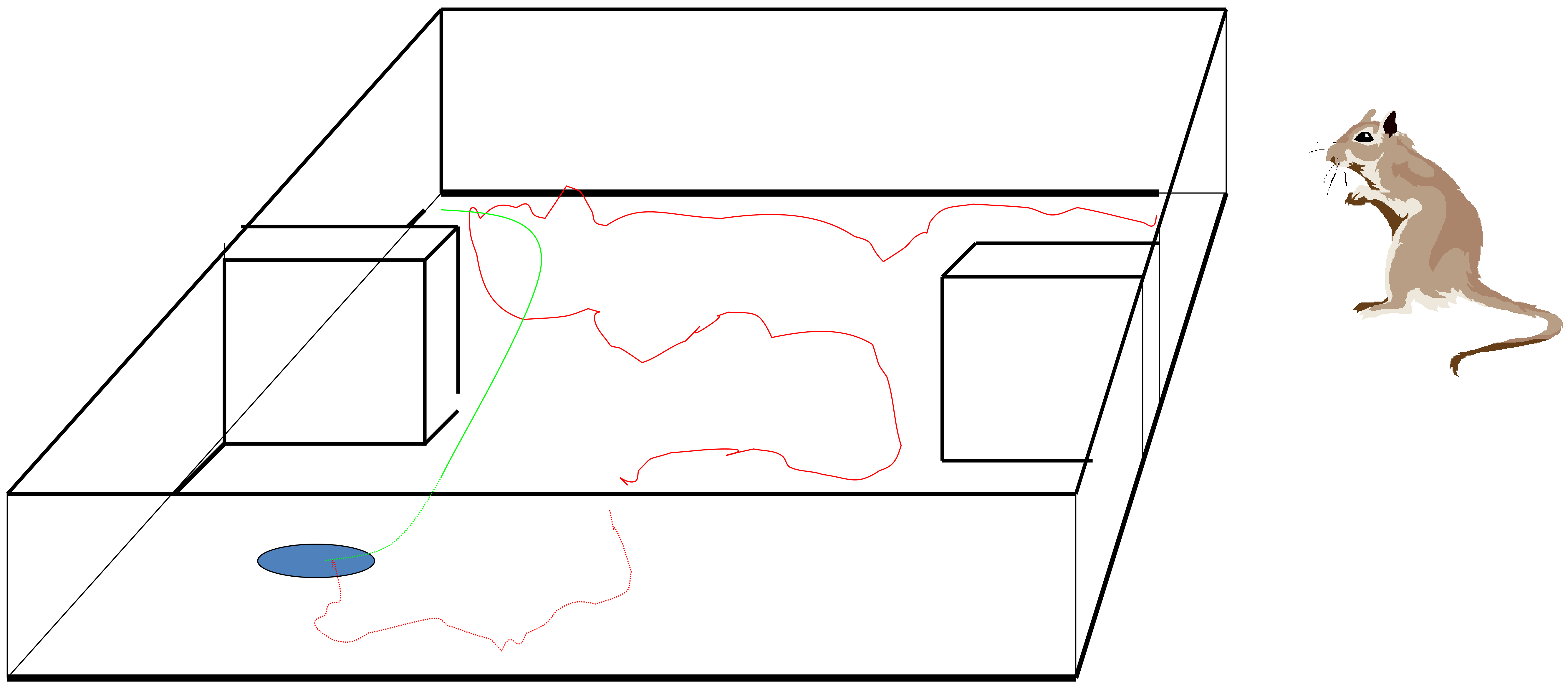
Min. shift



Feedback:
tone for wrong response

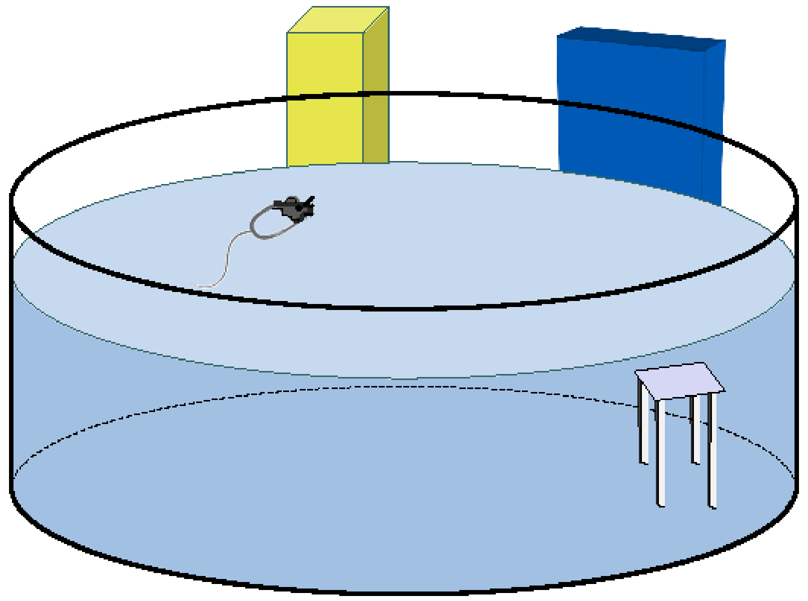
Tartaglia,Aberg,Herzog 2009

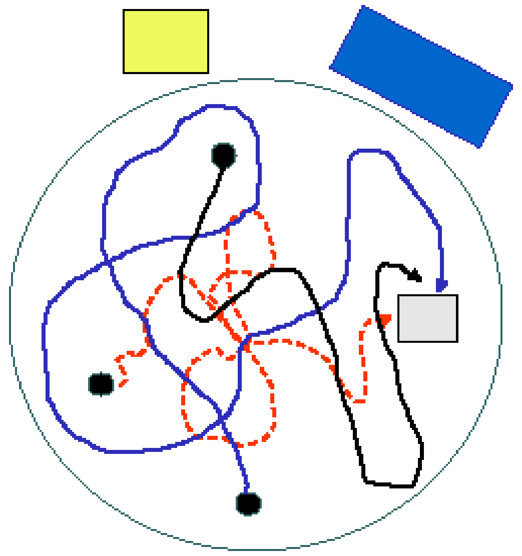
1. Examples of reinforcement learning: animal conditioning

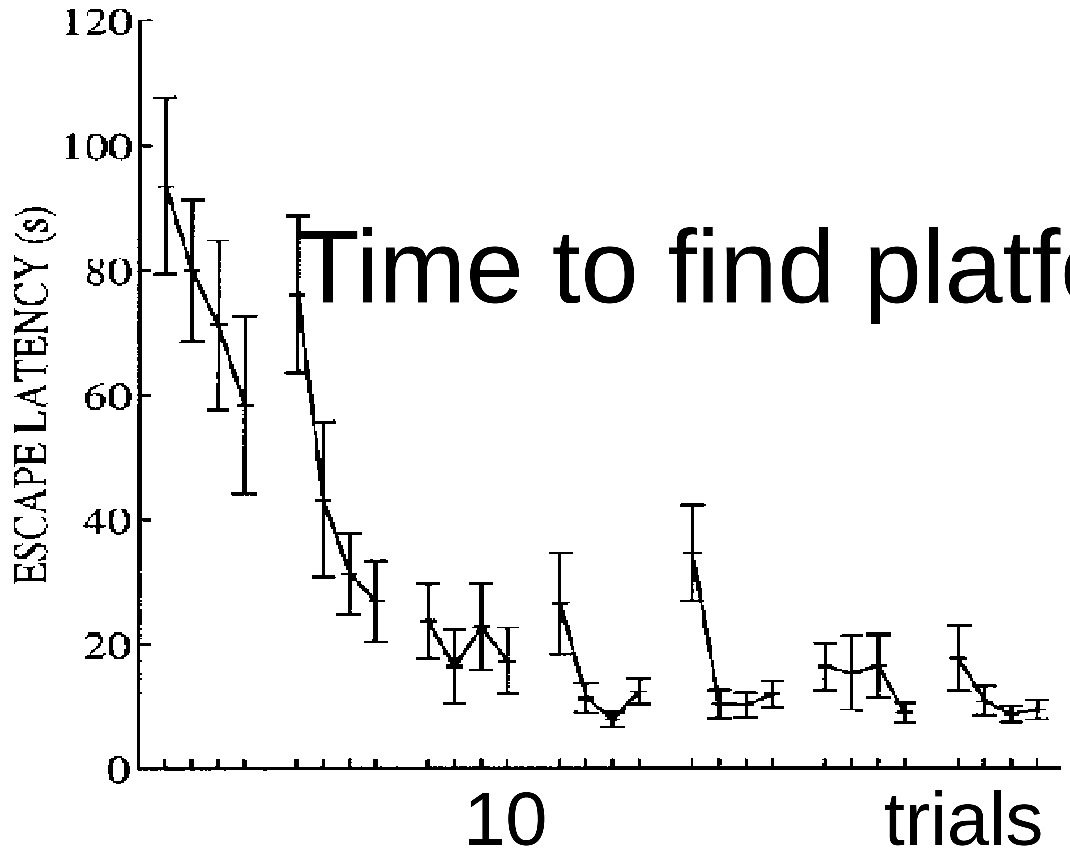


1. Examples of reinforcement learning: animal conditioning

Morris Water Maze







Time to find platform

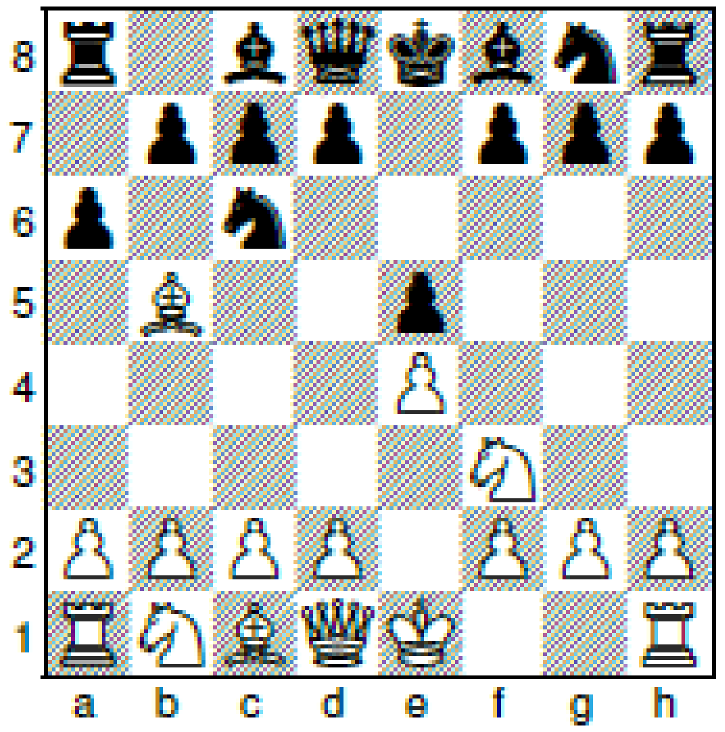
Rats learn to find the hidden platform

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

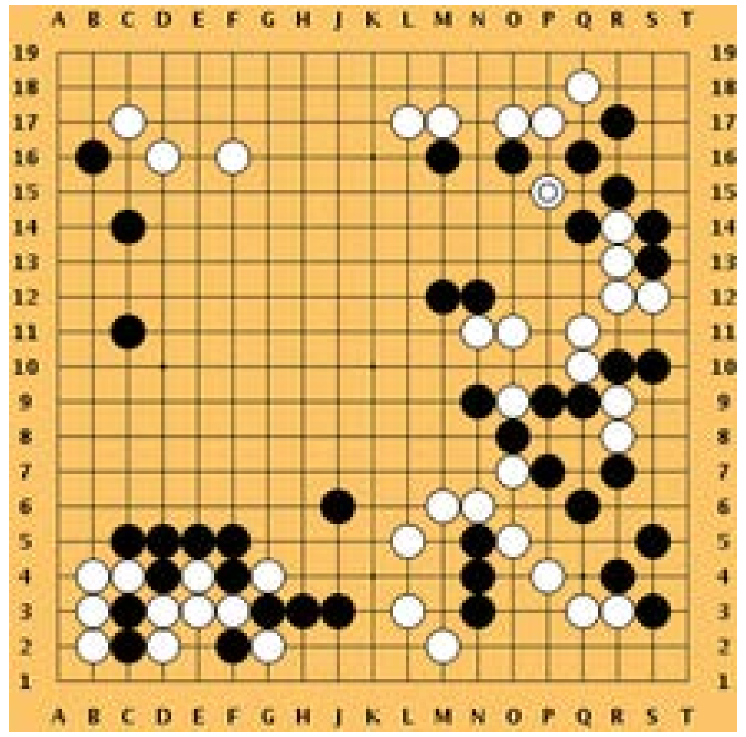
Foster, Morris, Dayan 2000

1. Deep reinforcement learning (in 3 weeks)

Chess



Go



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

In Go, it beats Lee Sedol

1. Deep reinforcement learning

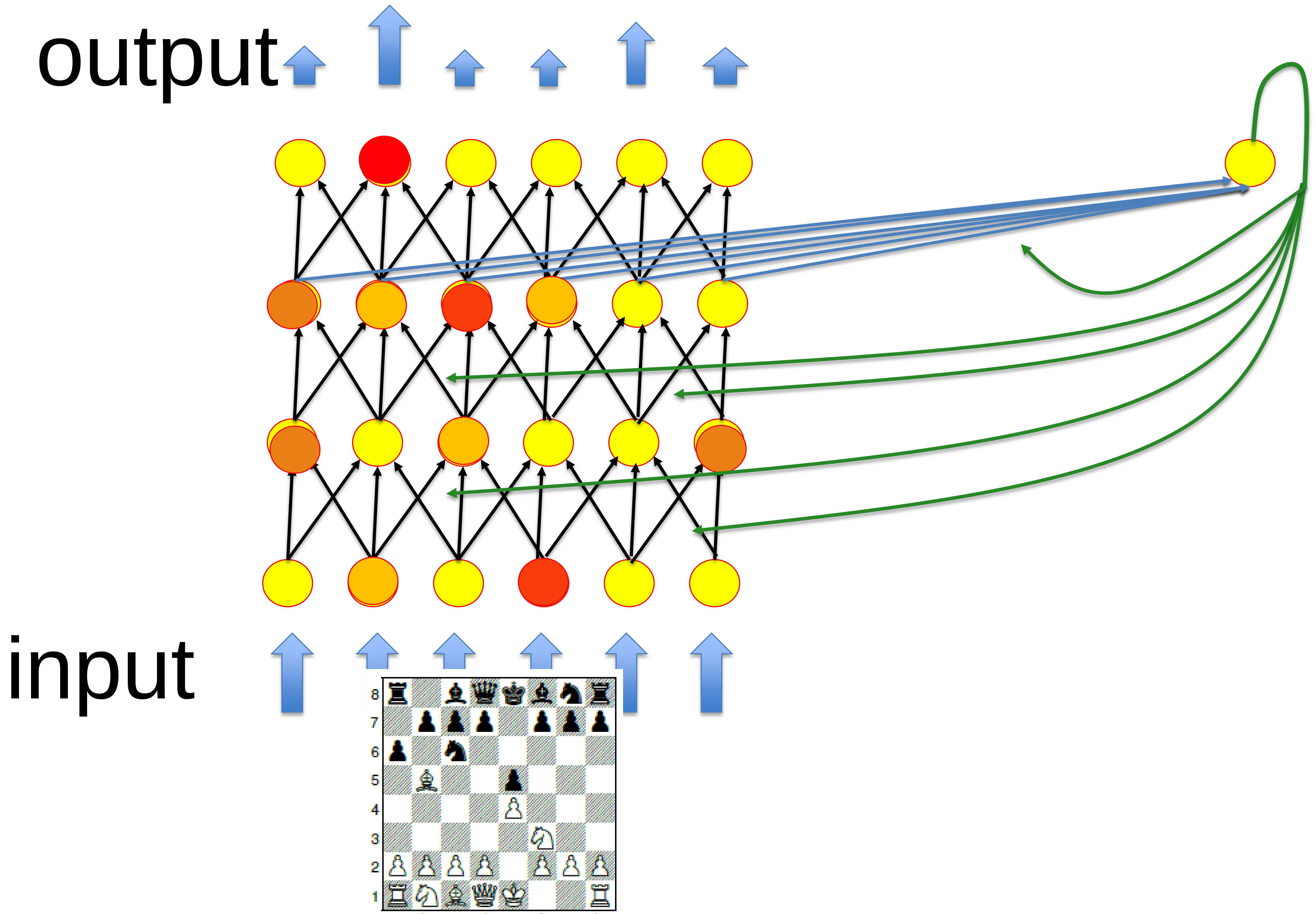
Network for choosing action

action:

Advance king

2^{e} output for **value** of state:

probability to win



learning:

- change connections

aim:

- Choose next action to win

aim for value unit:

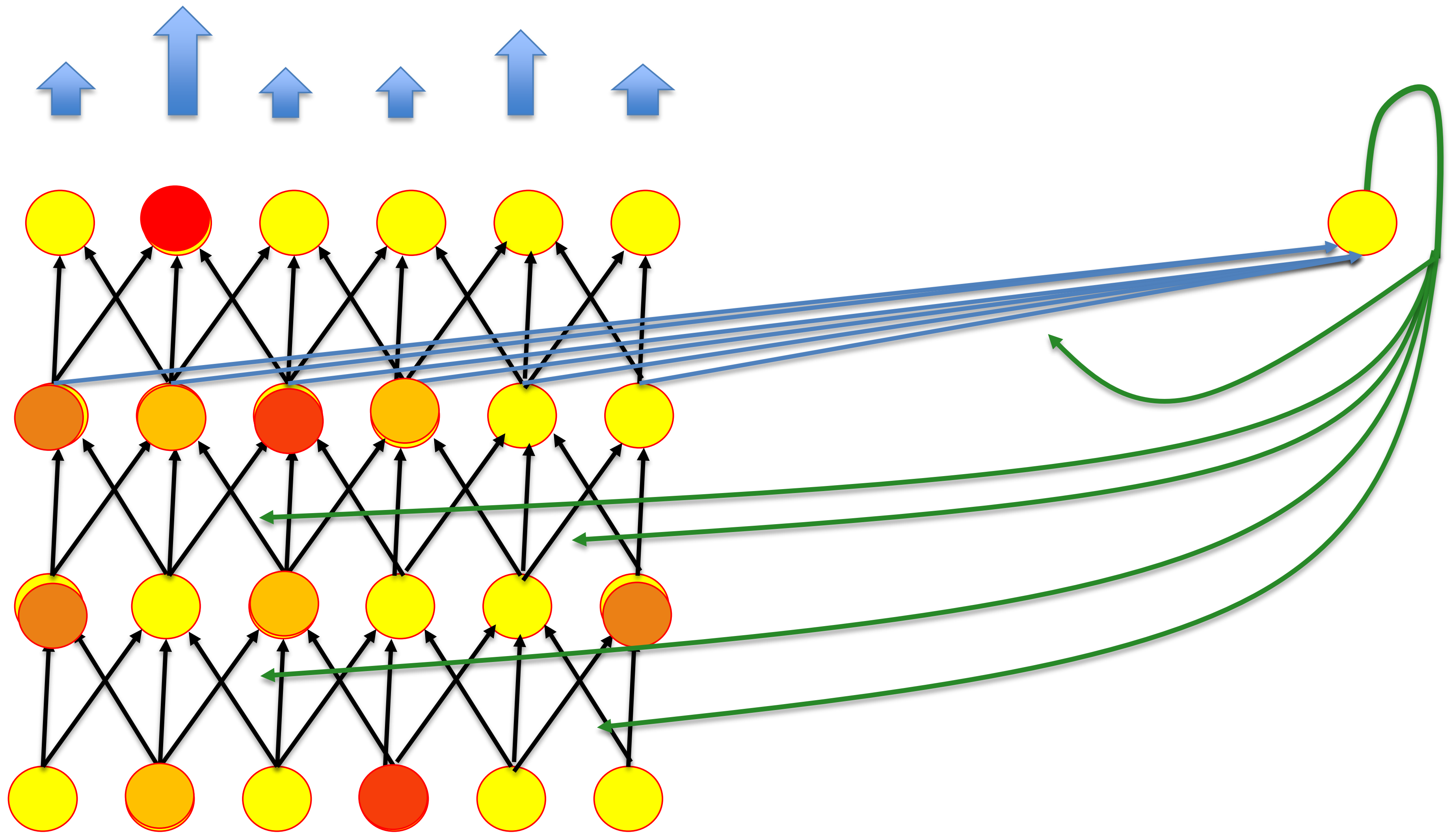
- Predict value of current position

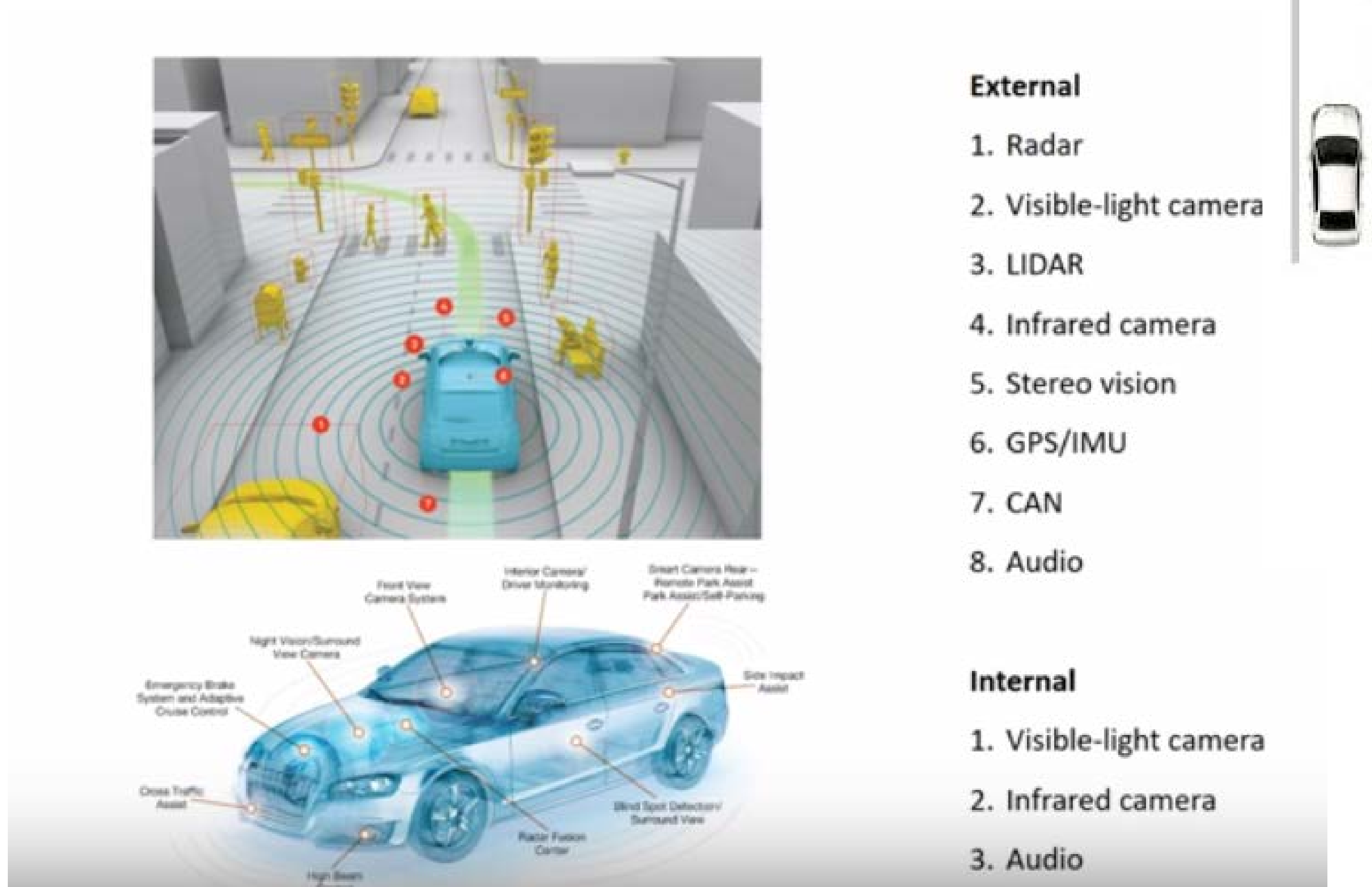
1. Deep Reinforcement Learning: Self-driving cars

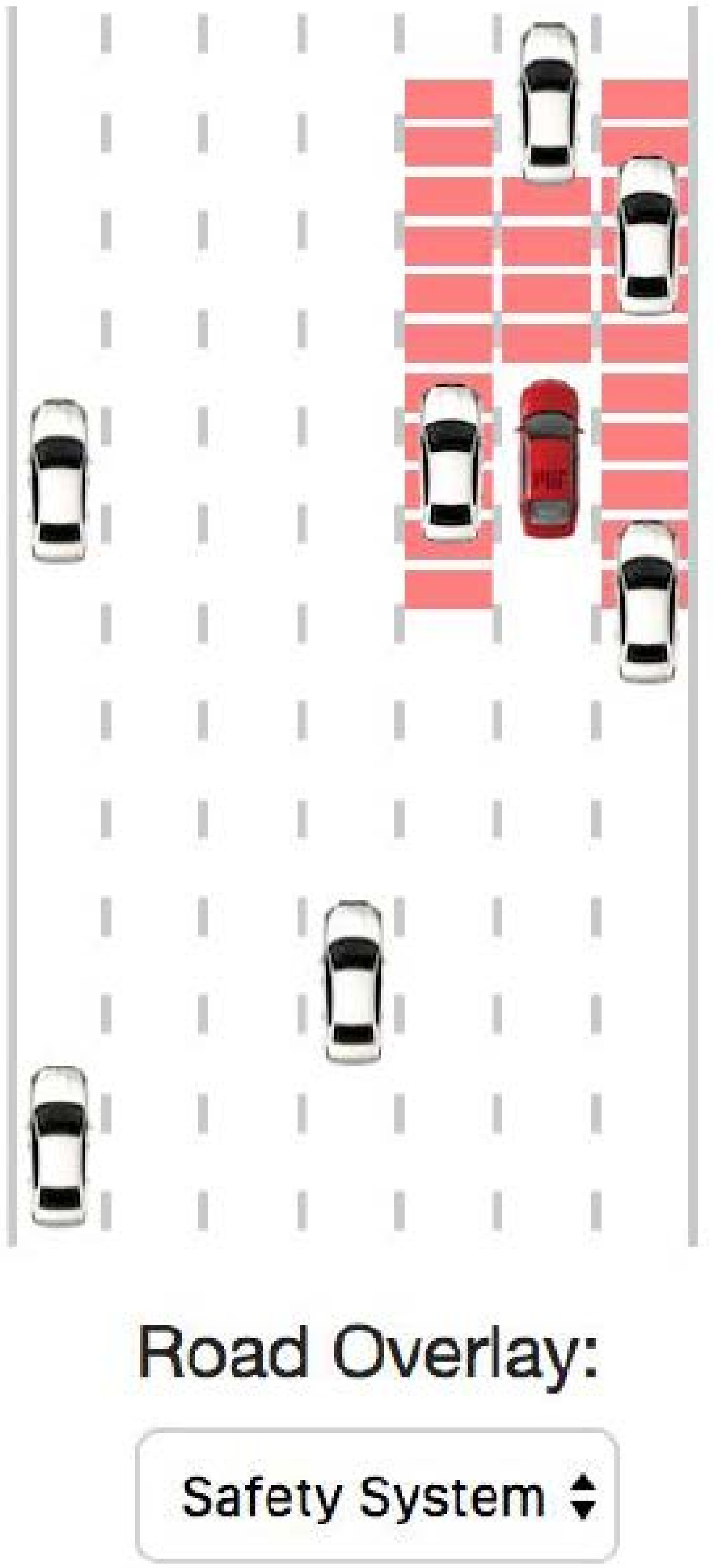
https://selfdrivingcars.mit.edu/ *Lex Friedman, MIT*

advance and accerate

Value: security, duration of travel



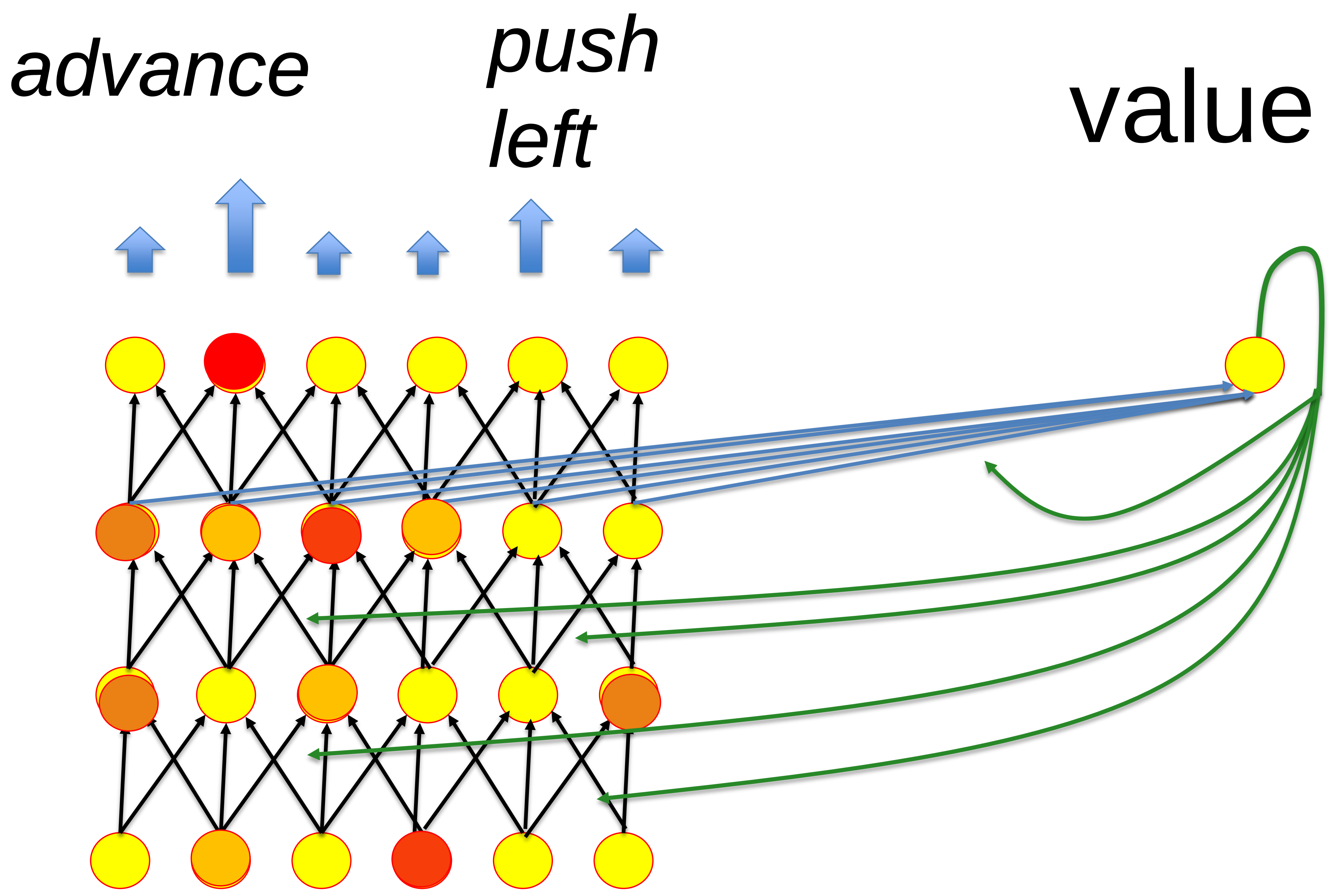


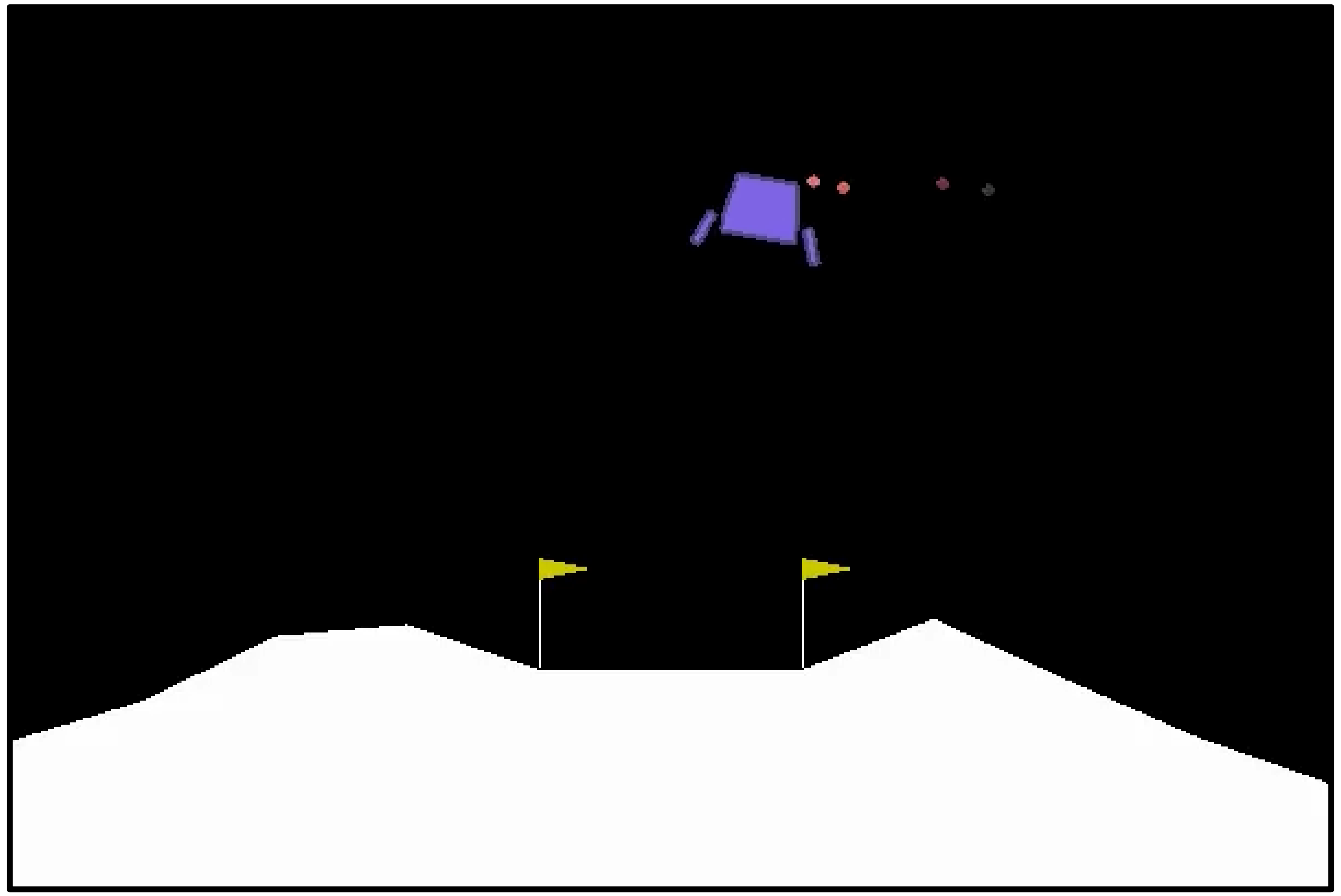


1. Deep Reinforcement Learning: Lunar Lander (miniproject)

actions

Aim: land between poles





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

Artificial Neural Networks: Lecture 8
Reinforcement Learning and SARSA

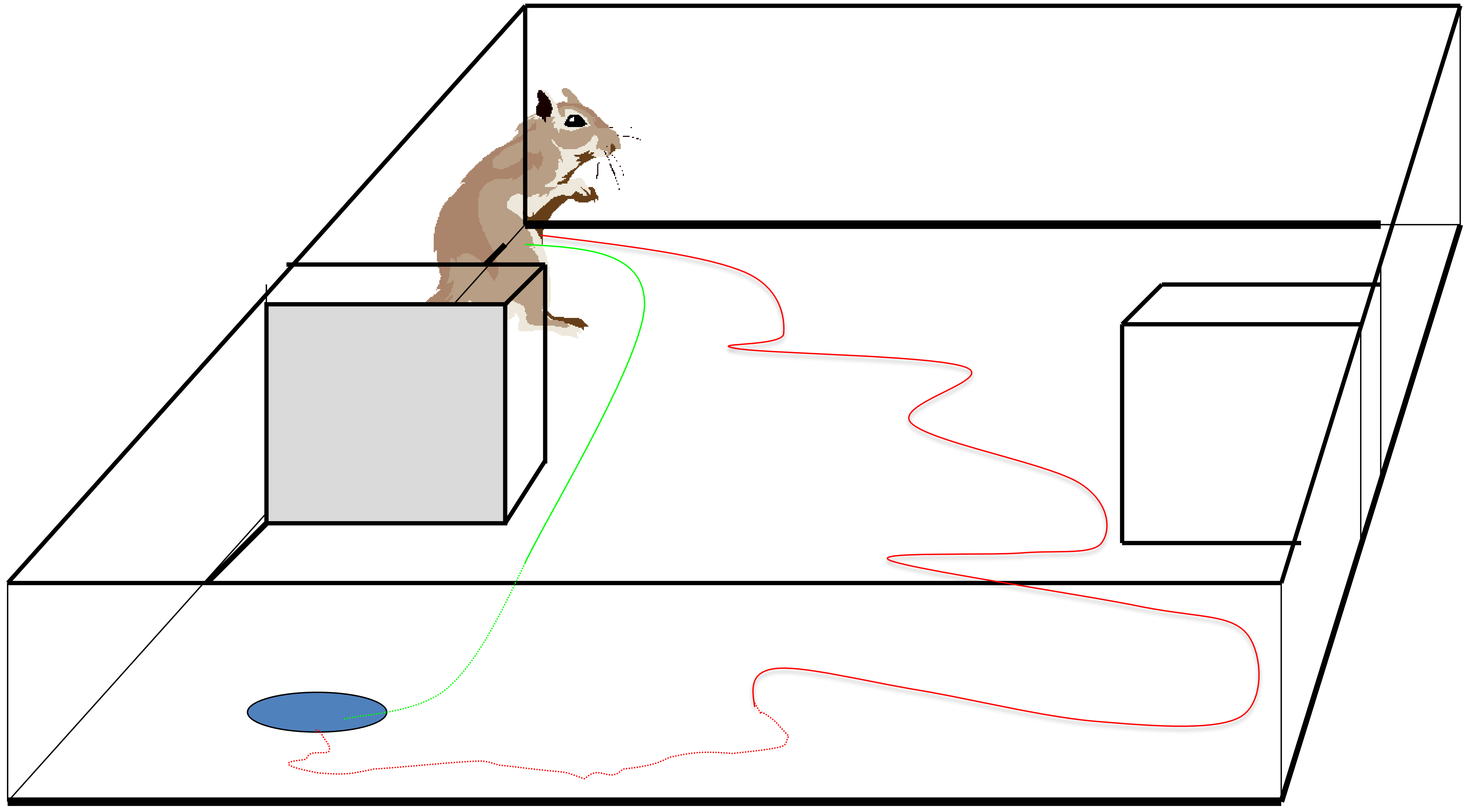
1. Learning by Reward: Reinforcement Learning
2. Elements of Reinforcement Learning

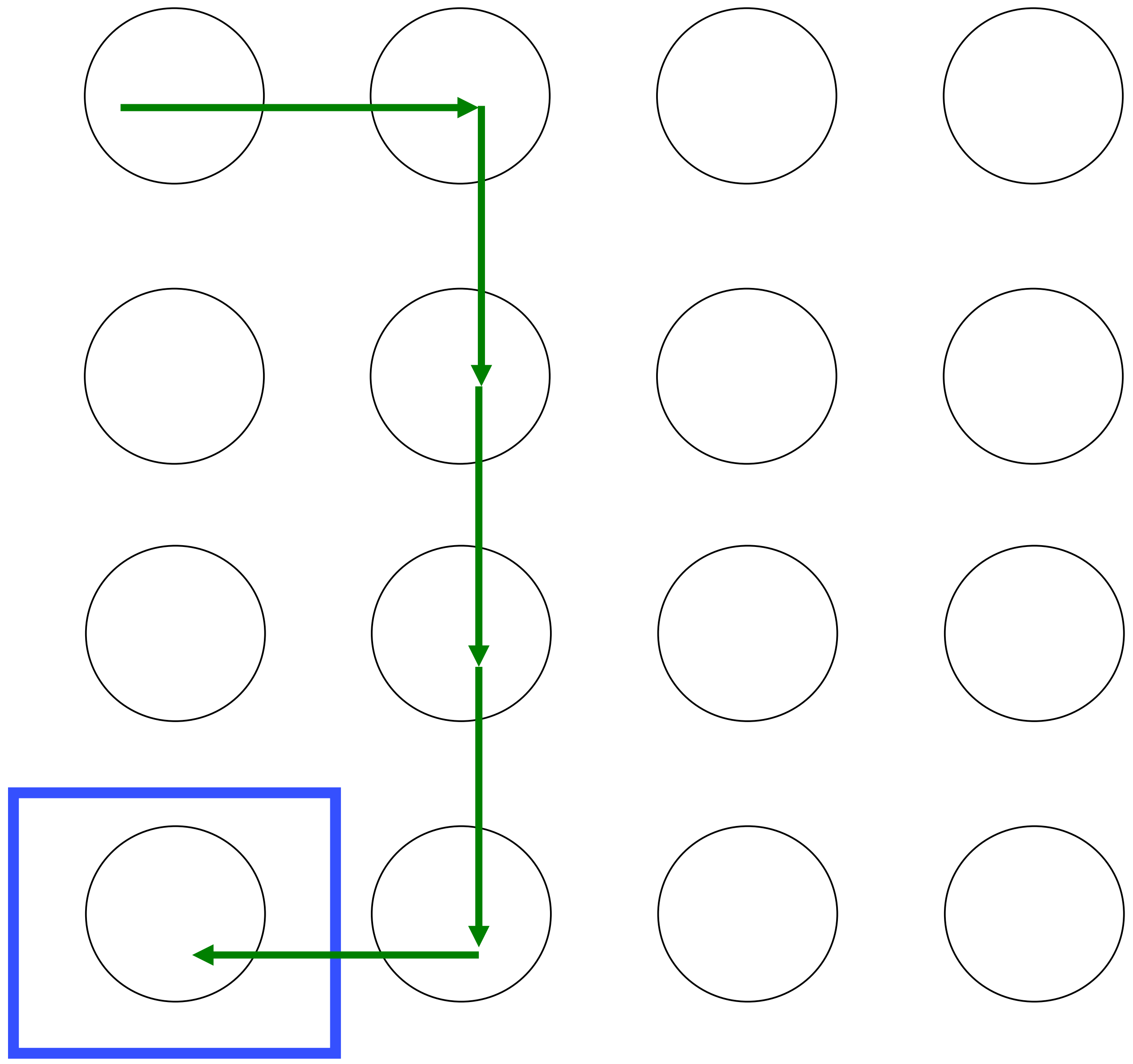
2. Elements of Reinforcement Learning:

-states
-actions
-rewards

2. Elements of Reinforcement Learning:

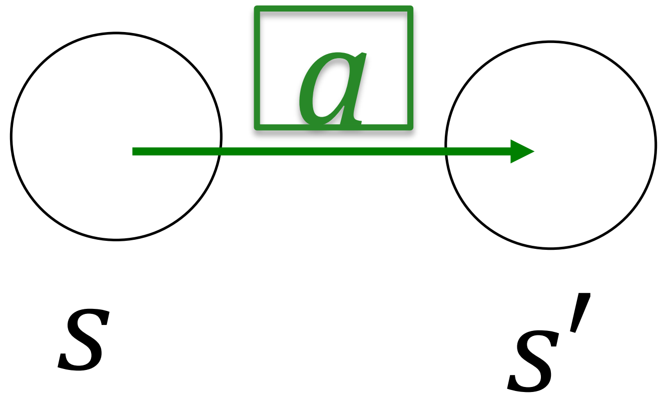
- discrete states
- discrete actions
- sparse rewards





2. Elements of Reinforcement Learning:

- discrete states:
  - old state s
  - new state s'



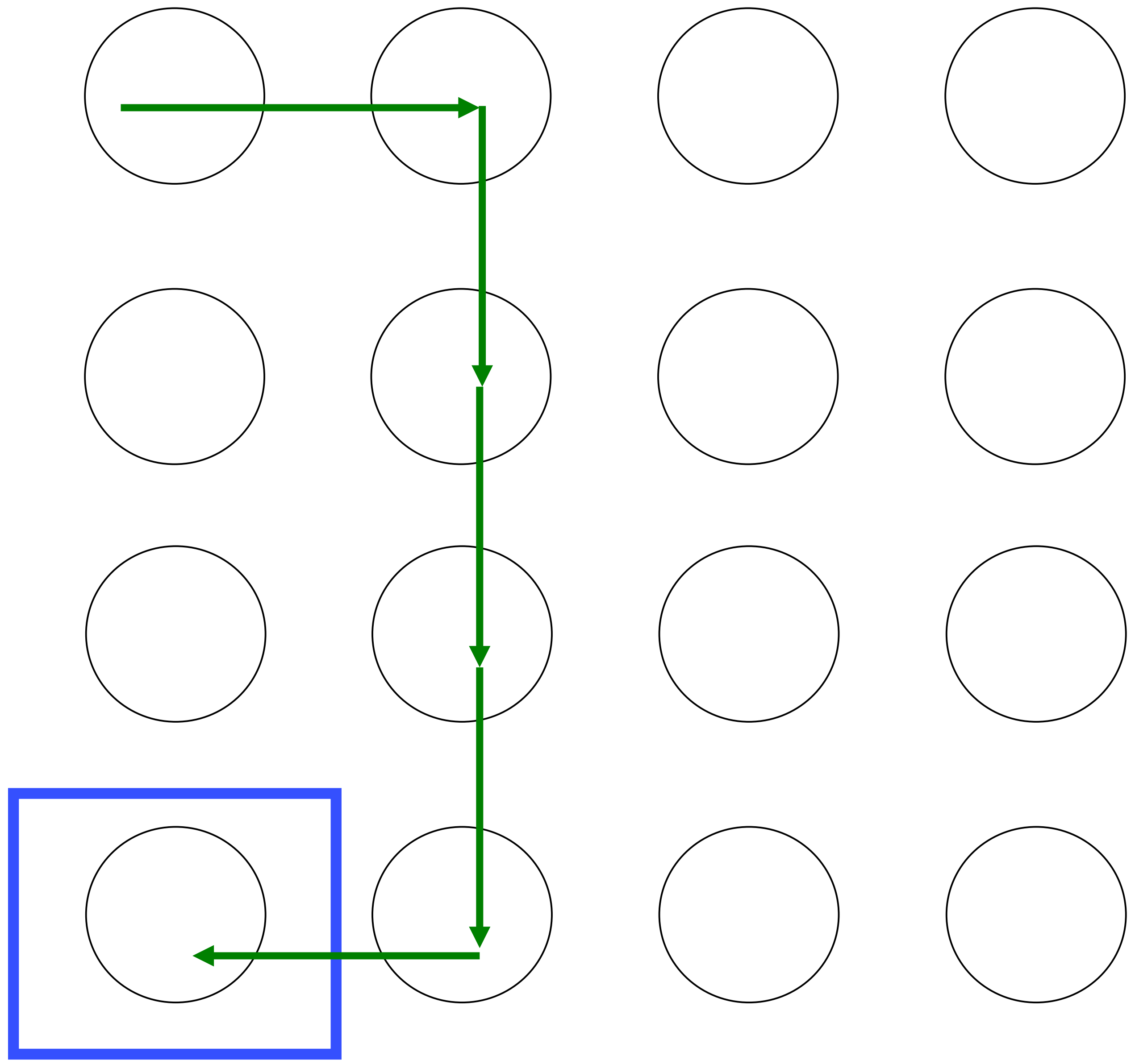
- current state: s_t

- discrete actions: a

- Mean rewards for transitions:

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

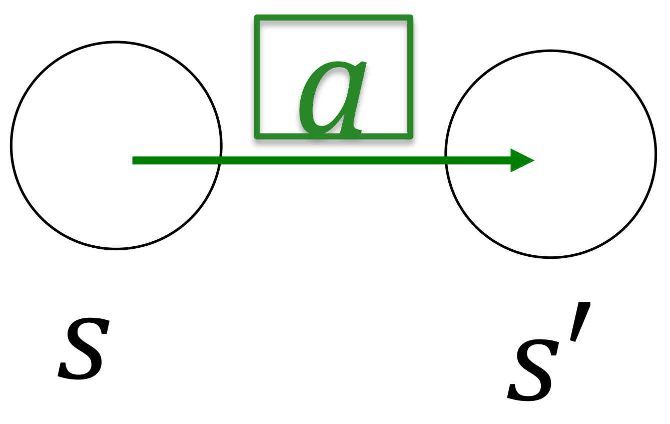
- current reward: r_t



often most transitions have zero reward

2. States Reinforcement Learning:

- discrete states:
  old state s
  new state s'



- current state: s_t

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

2. Reinforcement Learning: Example Acrobot

3 actions: $\boxed{a1}$ = no torque,

$\boxed{a2}$ = torque +1 at elbow,

States? $\boxed{a3}$ = torque -1 at elbow

reward if tip above line

→ discretize!

Suppose 5 states per dimension,
How many states in total?
[ ] 5
[ ] 25
[ ] 125
[ ] 625

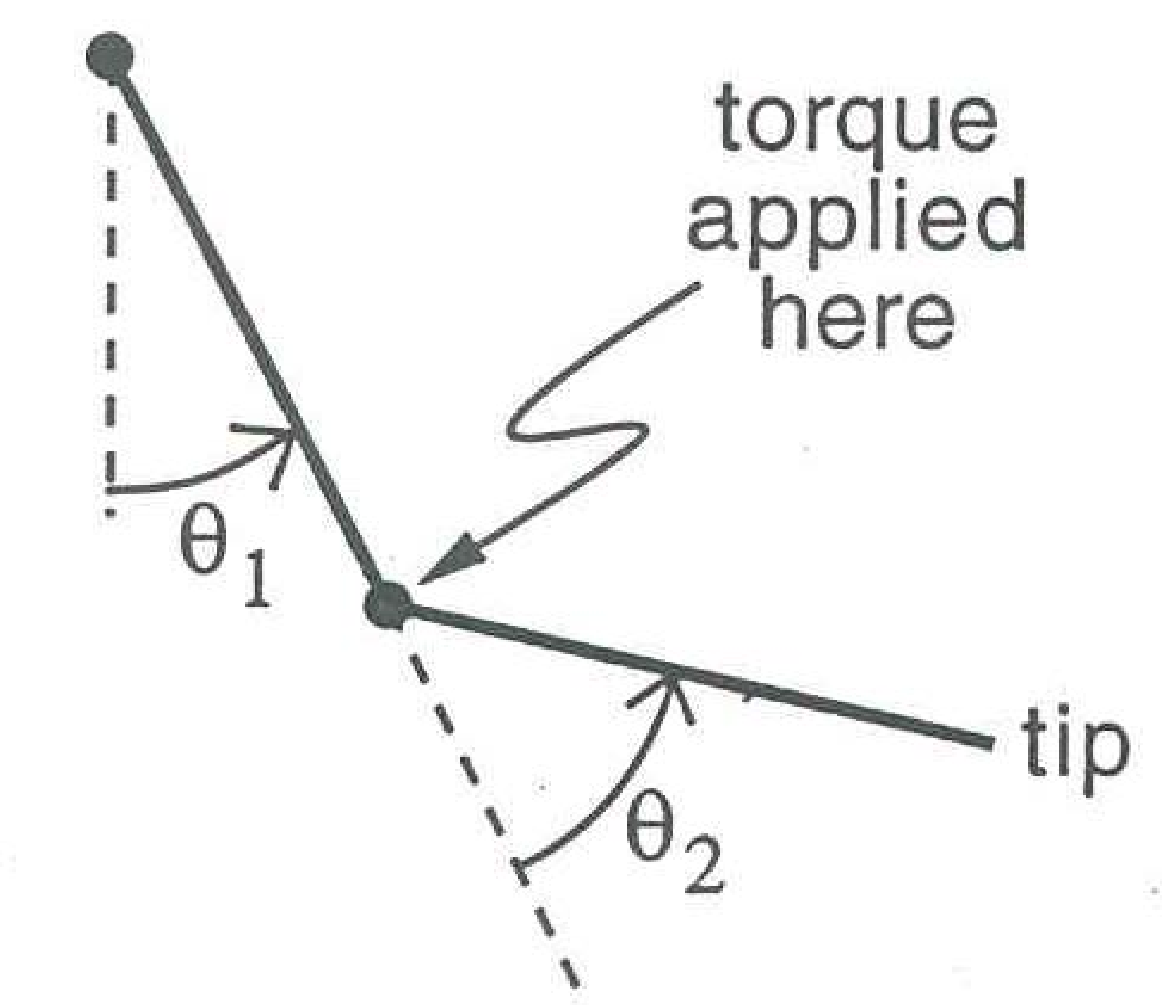


Figure 11.4 The acrobot.

From Book: Sutton and Barto

2. Reinforcement Learning: Example Acrobot

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

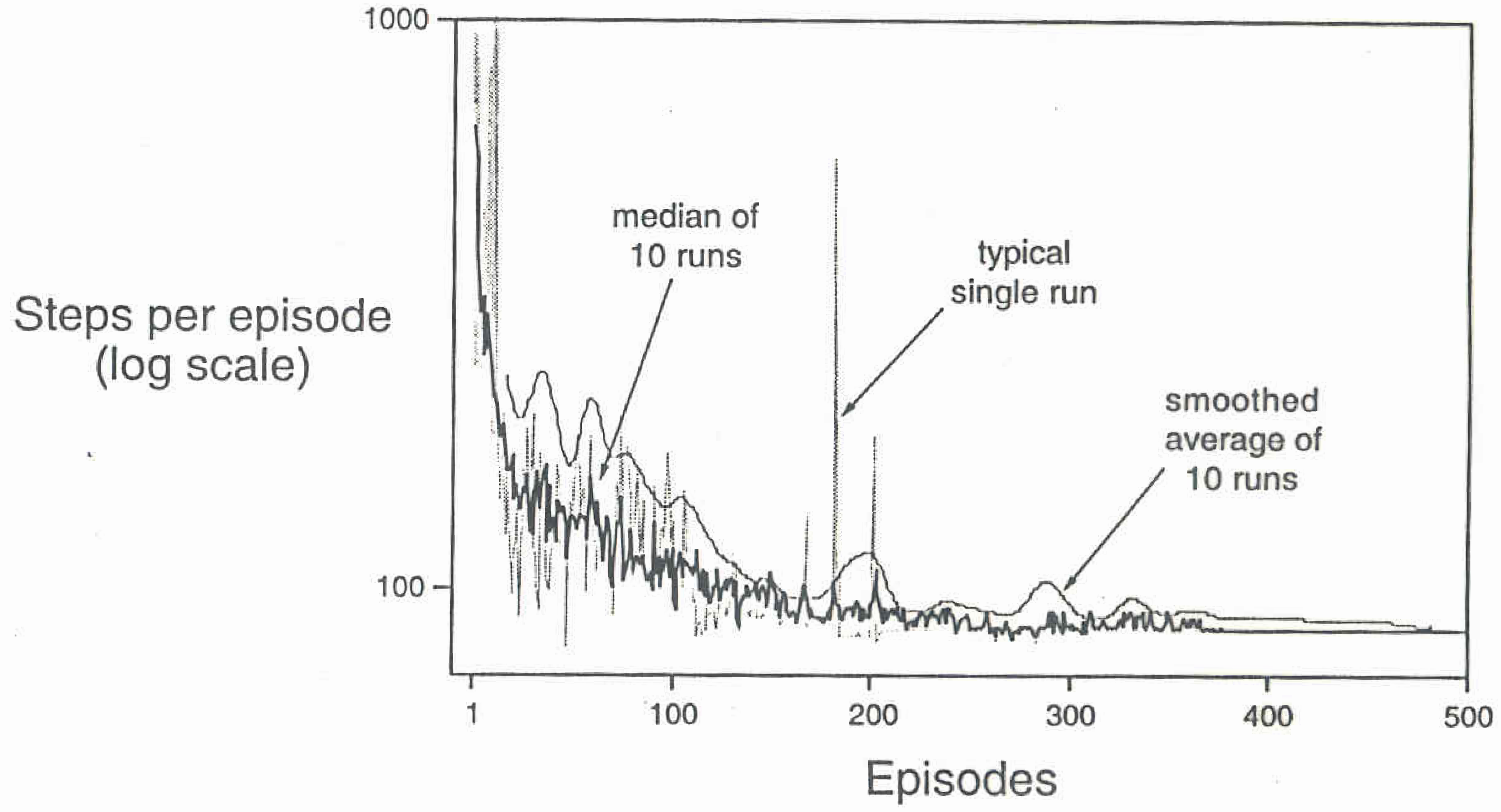


Figure 11.6 Learning curves for Sarsa(λ) on the acrobot task.

From Book:
Sutton and Barto

2. Reinforcement Learning: Example Acrobot

after 400 episodes

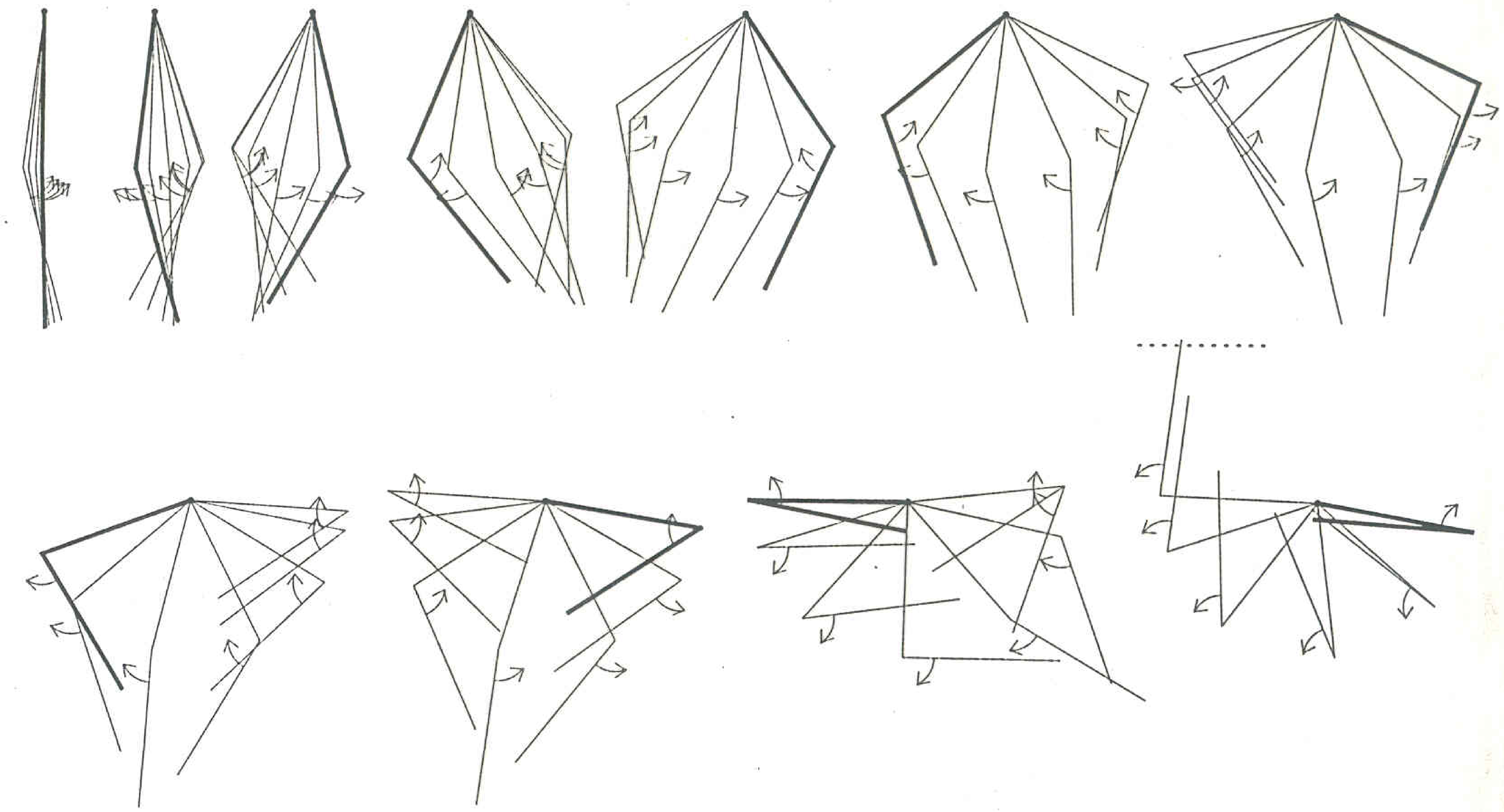


Figure 11.7 A typical learned behavior of the acrobot. Each group is a series of consecutive positions, the thicker line being the first. The arrow indicates the torque applied at the second joint.

From Book: Sutton and Barto

2. Reinforcement Learning: Example backgammon

Case Studies

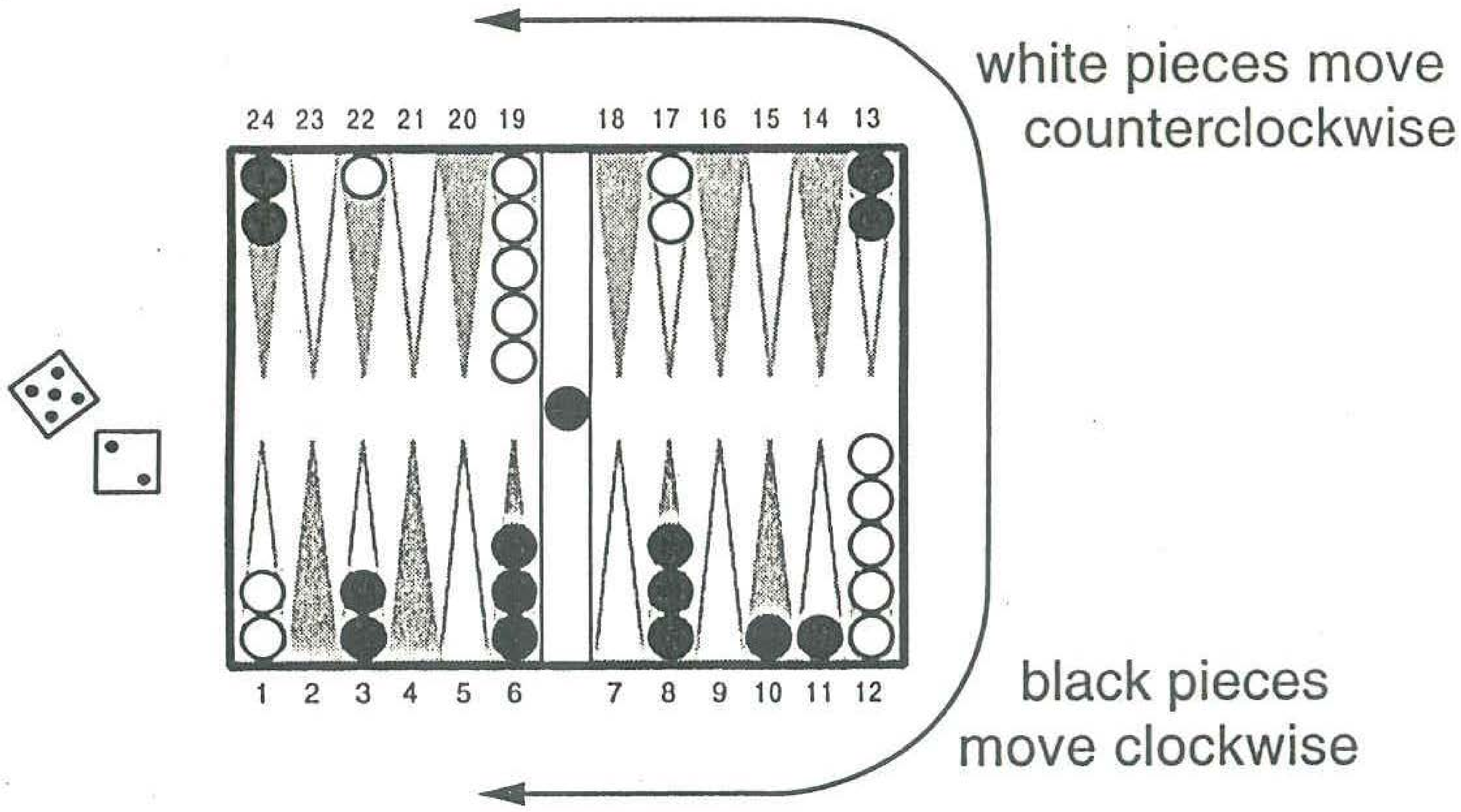


Figure 11.1 A backgammon position.

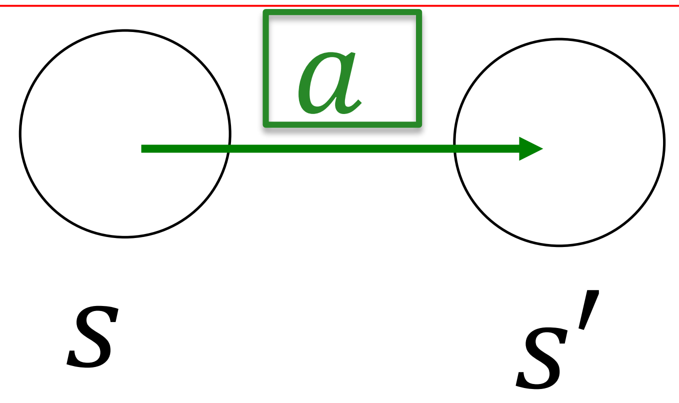
From Book:
Sutton and Barto

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

2. Elements of Reinforcement Learning: Summary

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



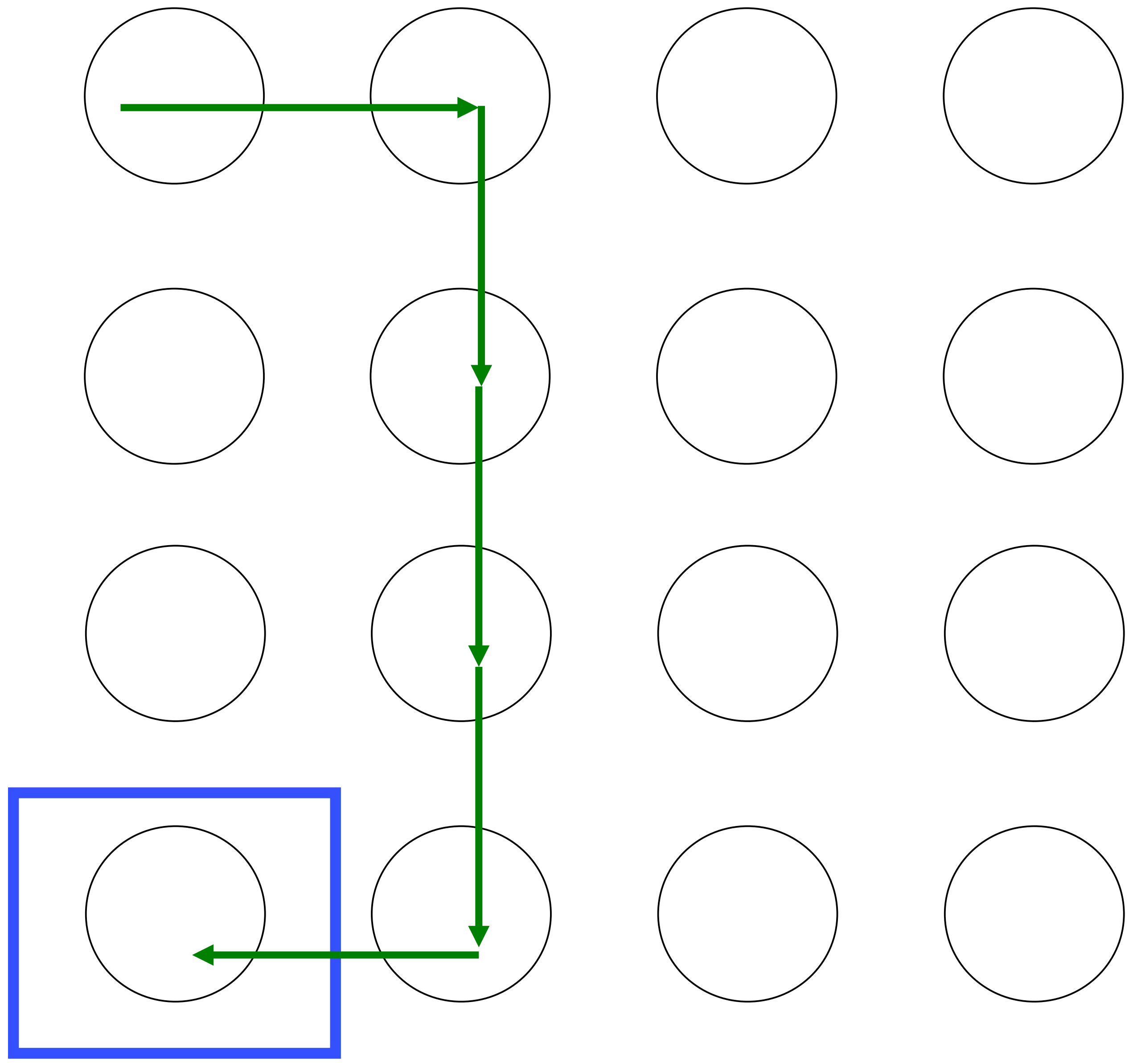
- discrete actions: a

- Mean reward for transition:

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

- current actual reward: r_t

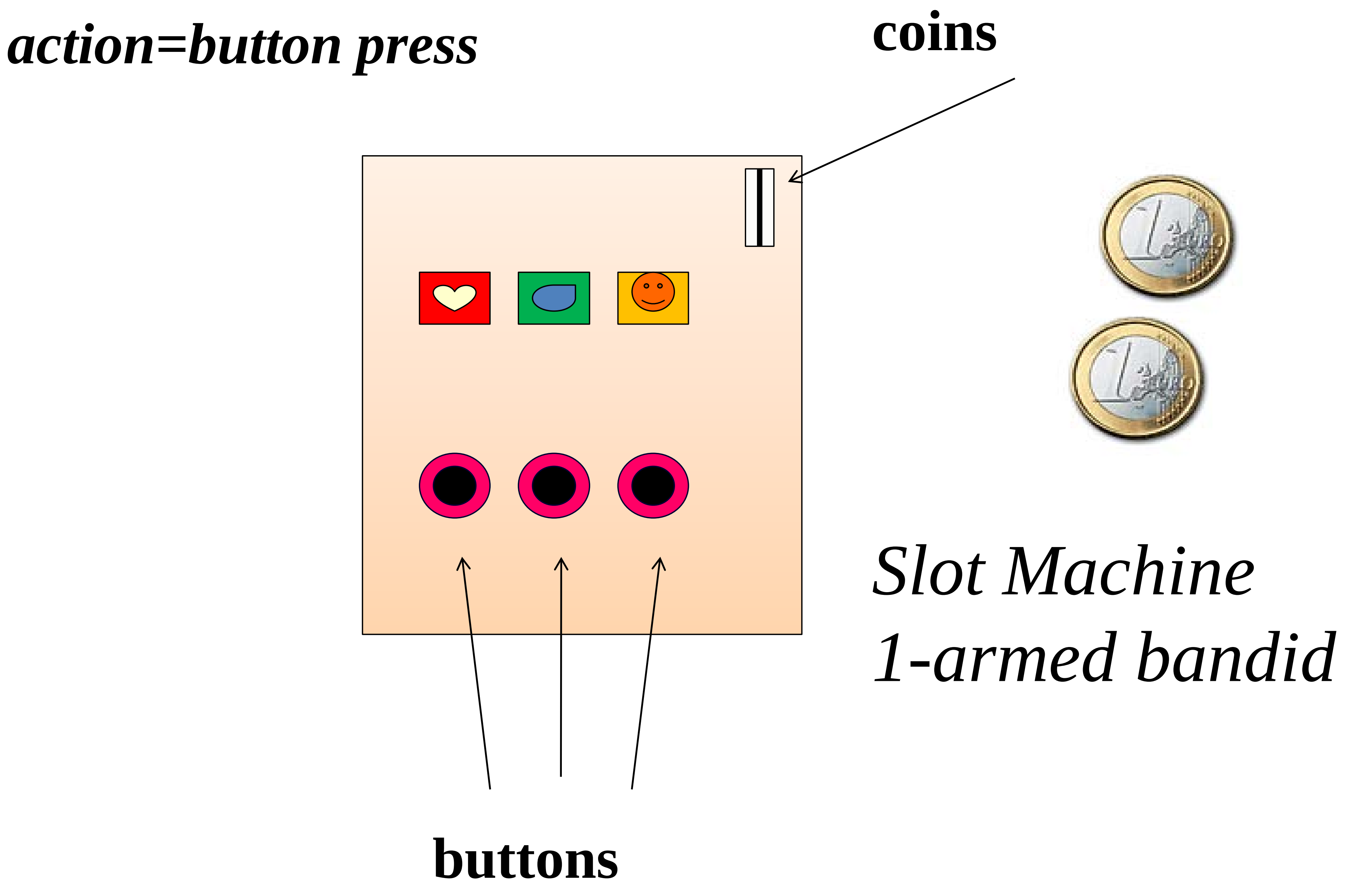
often most transitions have zero reward



Artificial Neural Networks: Lecture 8
Reinforcement Learning and SARSA

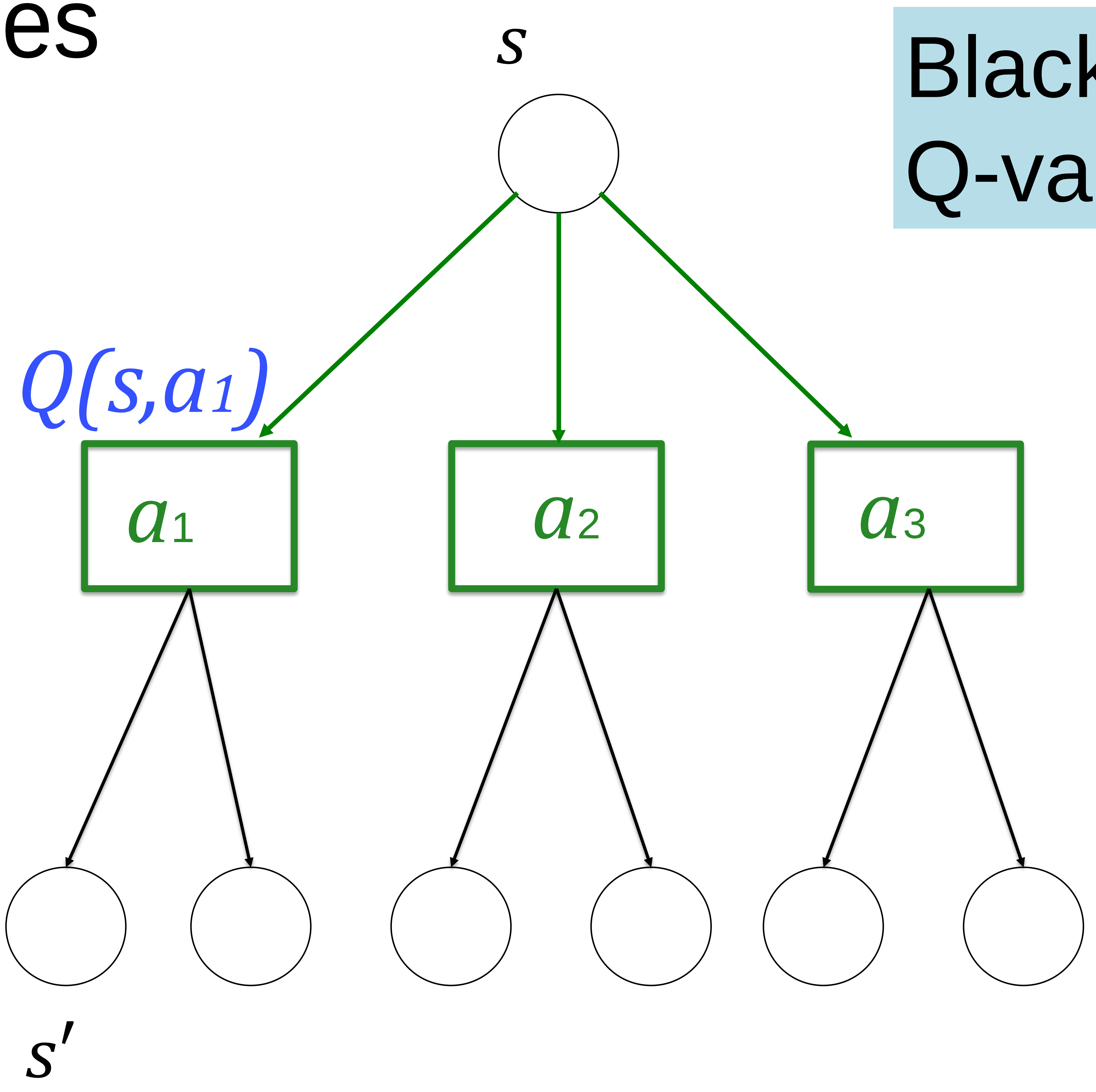
1. Learning by Reward: Reinforcement Learning
2. Elements of Reinforcement Learning
3. One-step horizon (bandit problems)

2. One-step horizon games (bandit)



2. One-step horizon games

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



Blackboard1: Q-values

2. One-step horizon games

Blackboard1:
Q-values

2. One-step horizon games: Q-value

Q-value $Q(s,a)$

Expected reward for action a starting from s

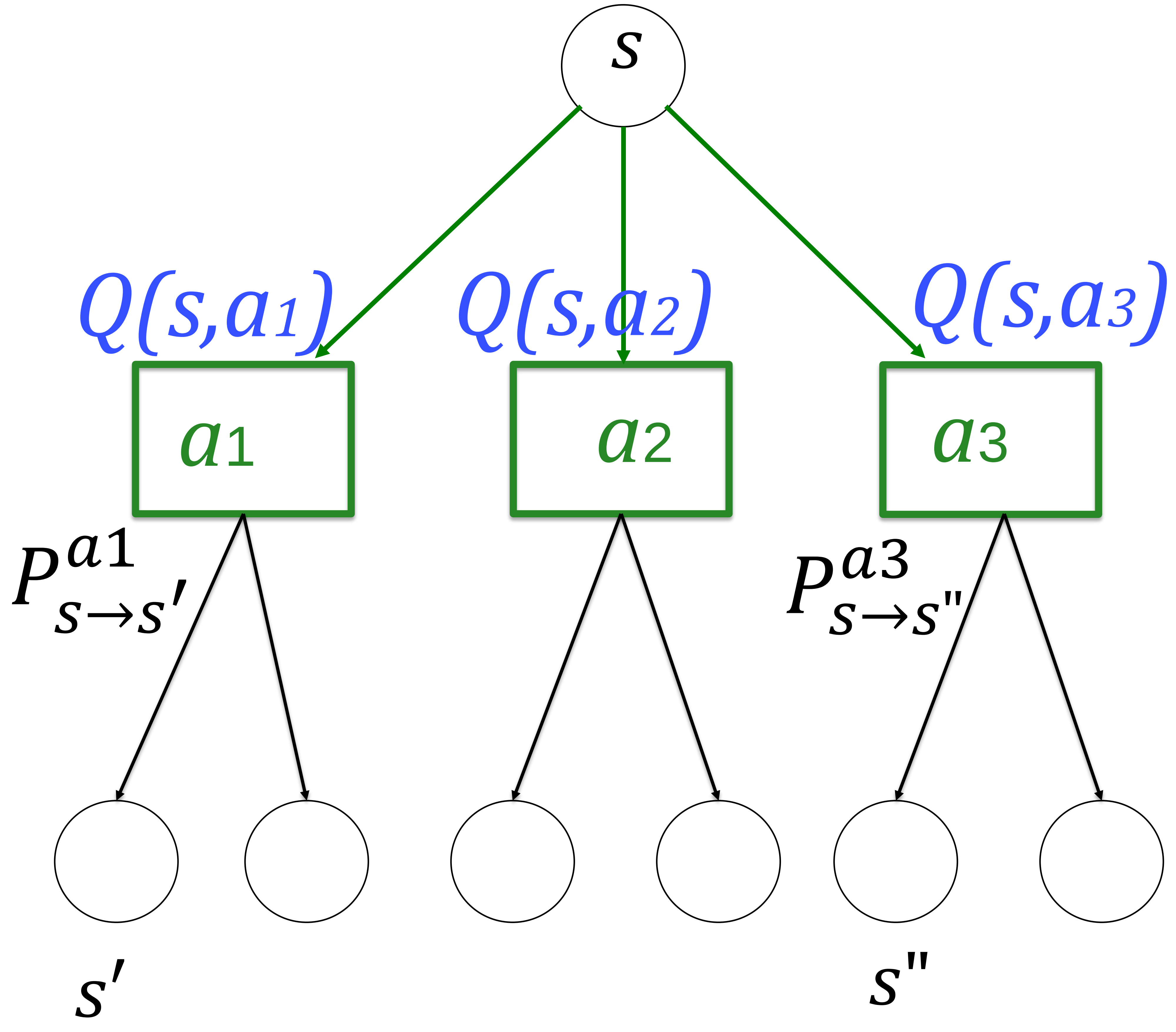
$$Q(s,a) = \sum_{s'} P^a_{s\to s'} R^a_{s\to s'}$$

Reminder:

$$R^a_{s\to s'} = E(r|s', a, s)$$

Similarly:

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



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

2. Optimal policy (greedy)

Suppose all Q-values are known:

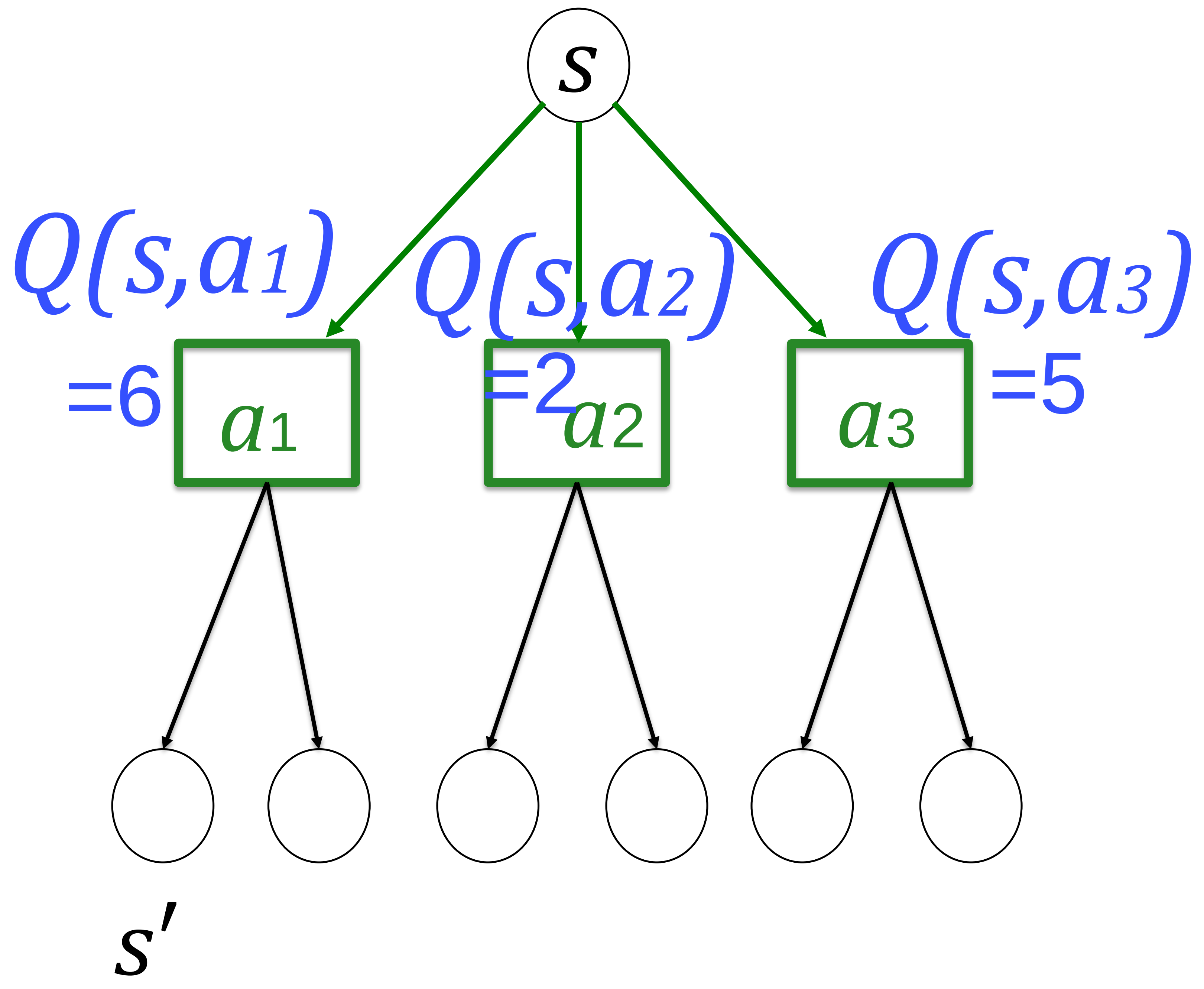
take action a with*

$Q(s,a^*) > Q(s,a_j)$

other actions

optimal action:

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



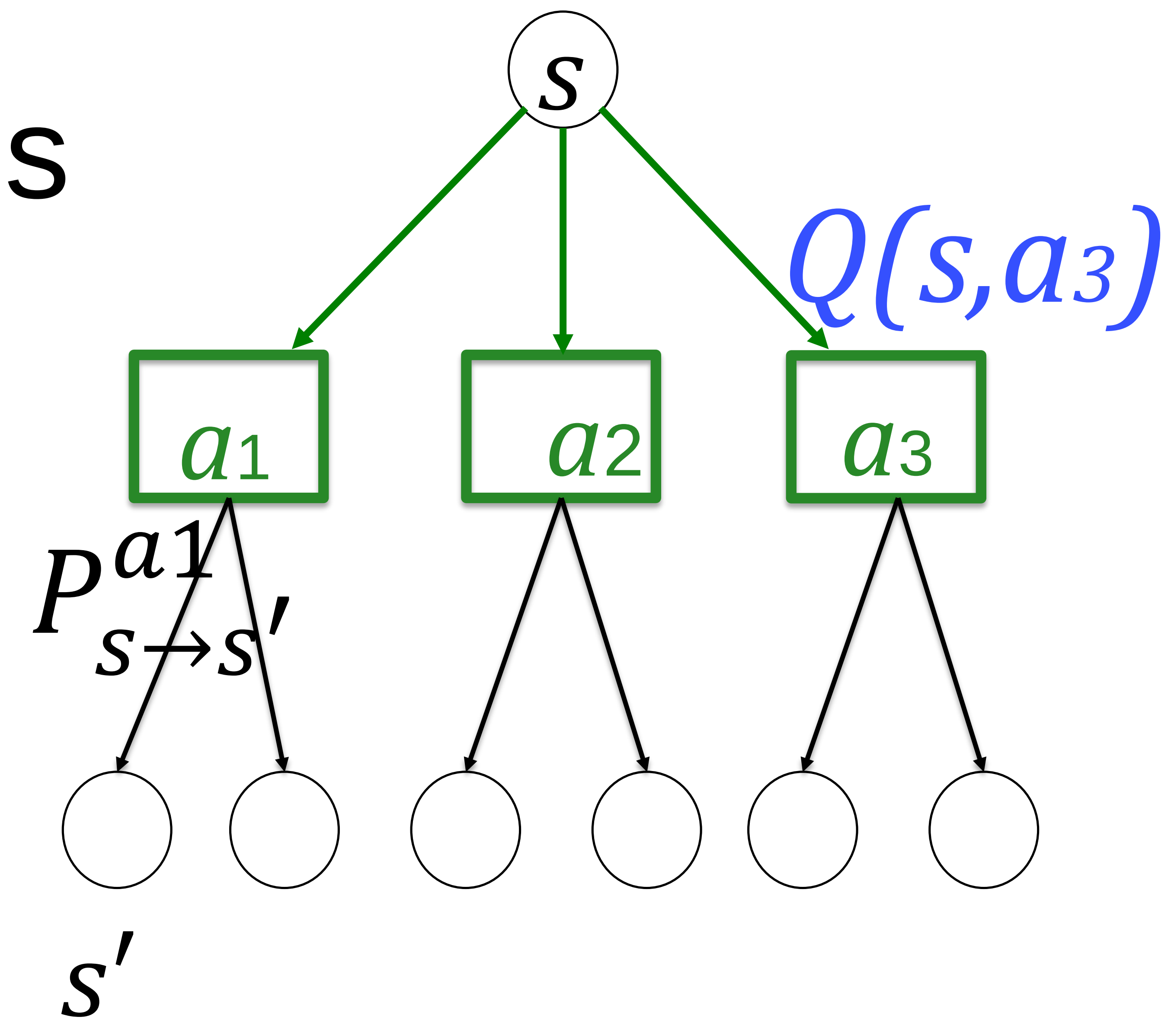
Optimal policy is also called ‘greedy policy’

2. 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

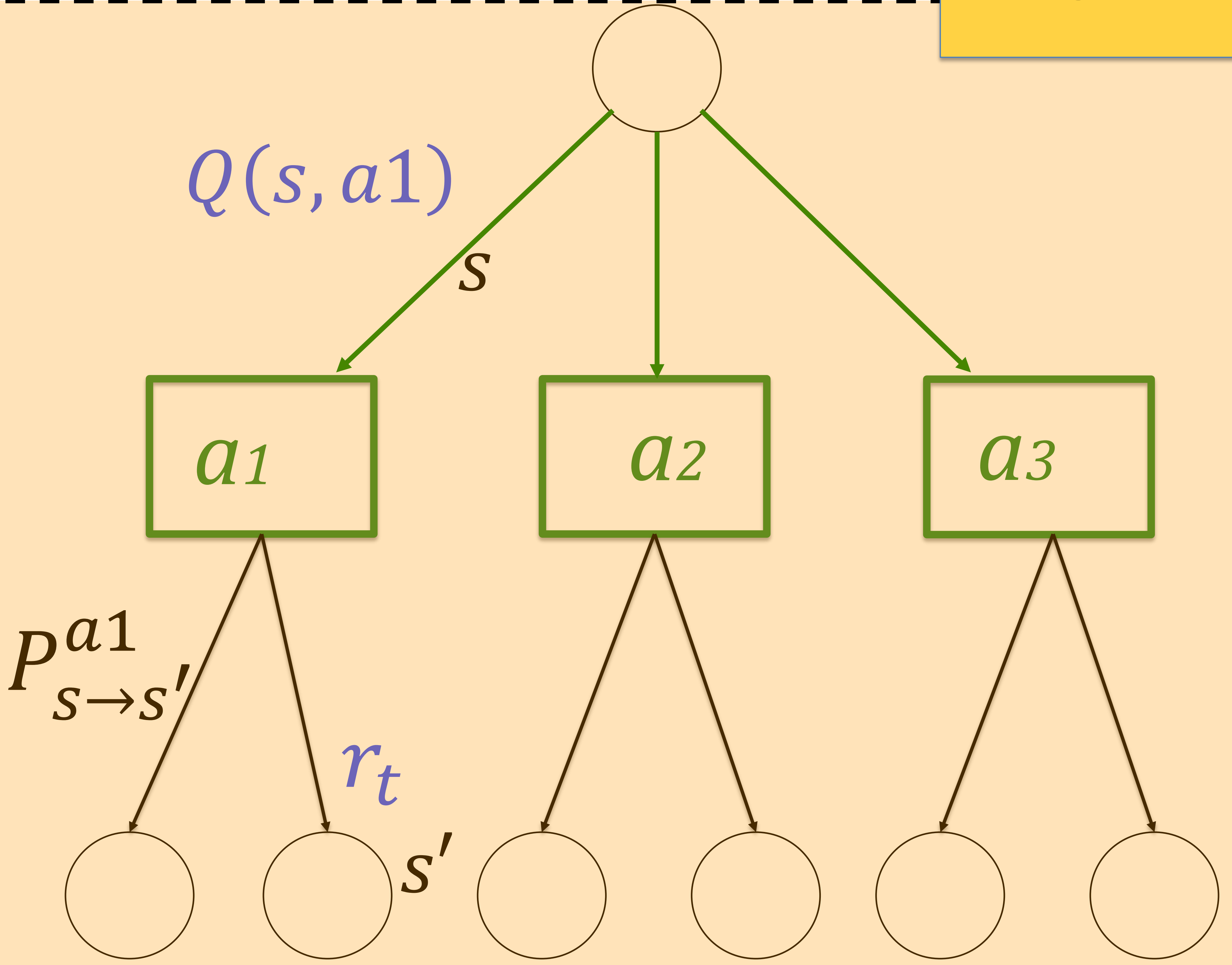


Exercise 1 now (in class)

5 min, now!

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

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

Blackboard2:
Exercise 1

2. 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

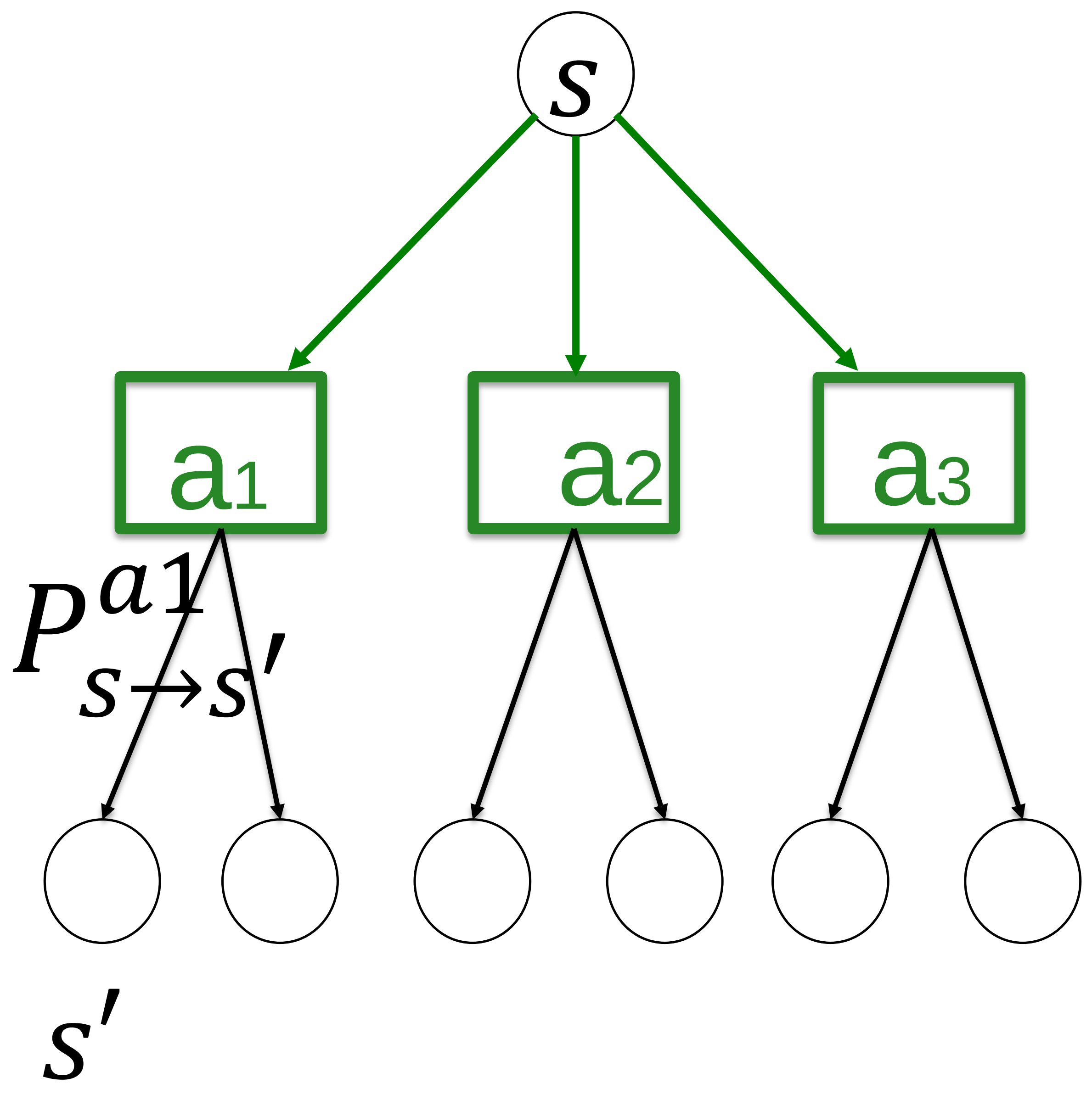
If Q-value not known:

→ estimate by trial and error

→ update with rule

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

Let learning rate η decrease over time



Convergence in Expectation

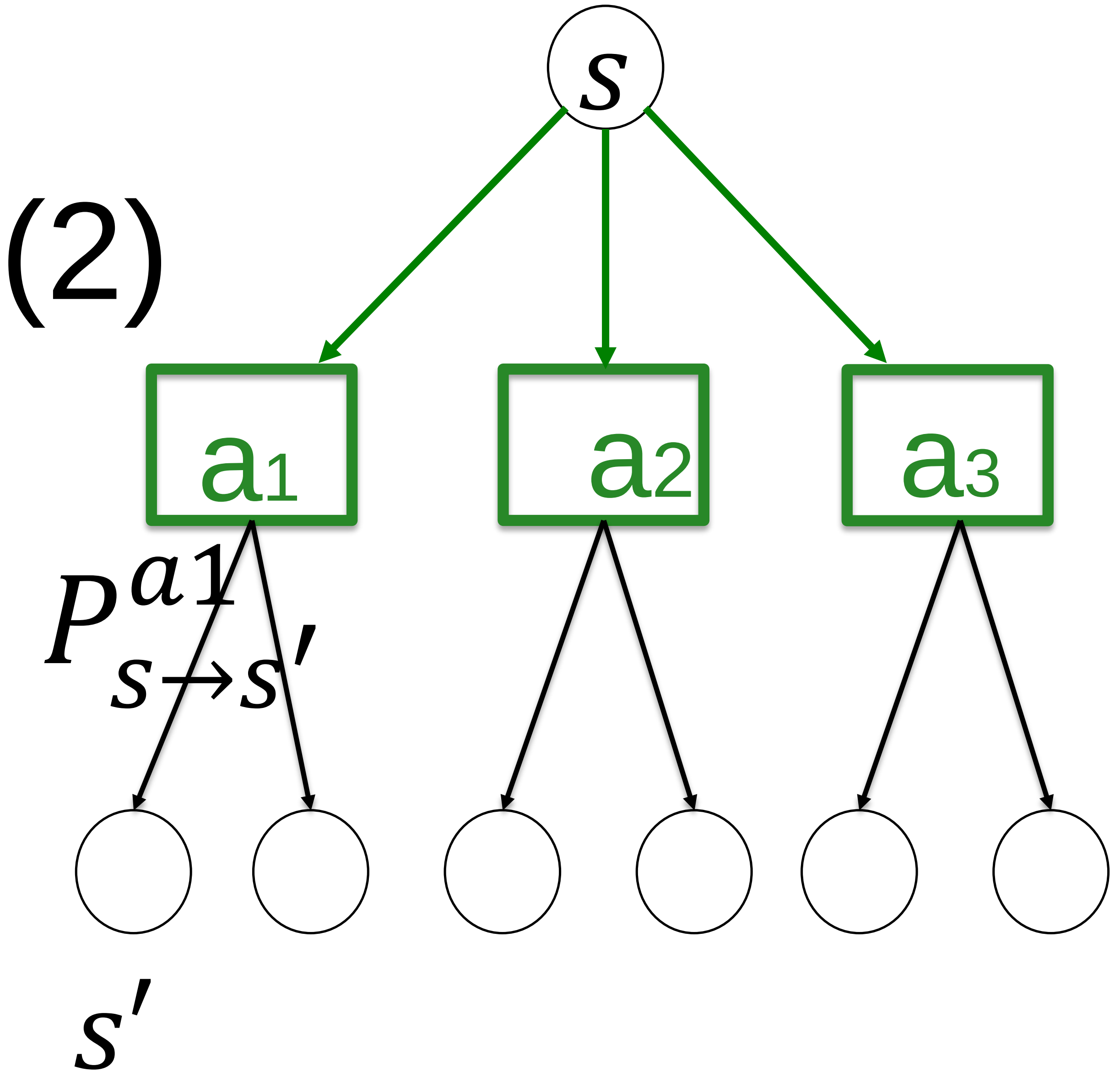
Blackboard3: Proof of (i).

After taking action a in state s, we update with

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

(i) If (1) converges in expectation, then Q fluctuates around,

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

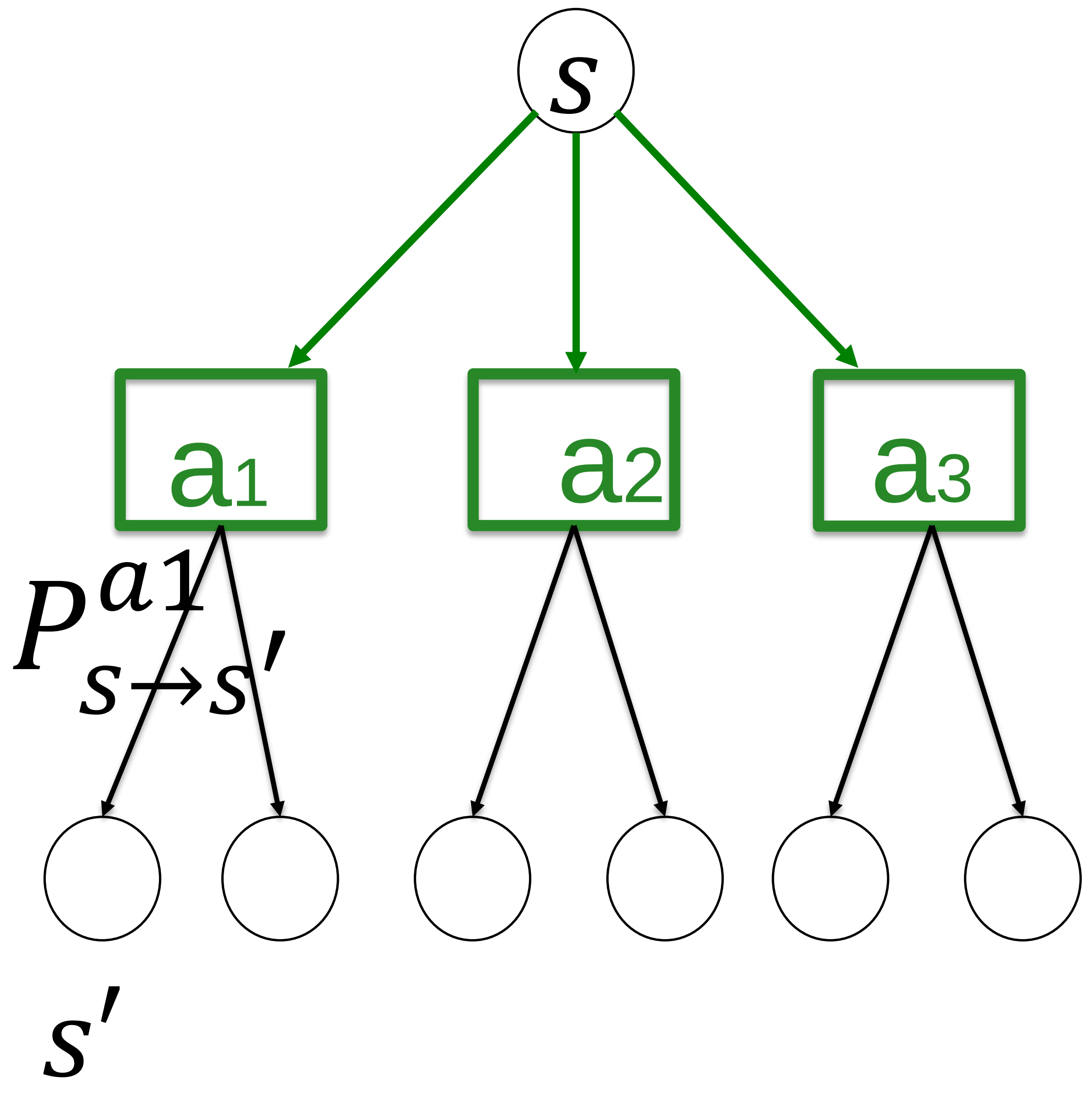


(ii) If the learning rate η decreases, fluctuations around $E[Q(s,a)]$ decrease.

Artificial Neural Networks: Lecture 8
Reinforcement Learning and SARSA

1. Learning by Reward: Reinforcement Learning
2. Elements of Reinforcement Learning
3. Exploration vs Exploitation

3. Exploration – Exploitation dilemma



Ideal: take action with maximal $Q(s, a)$

Problem: correct Q values not known
(since reward probabilities and branching probabilities unknown)

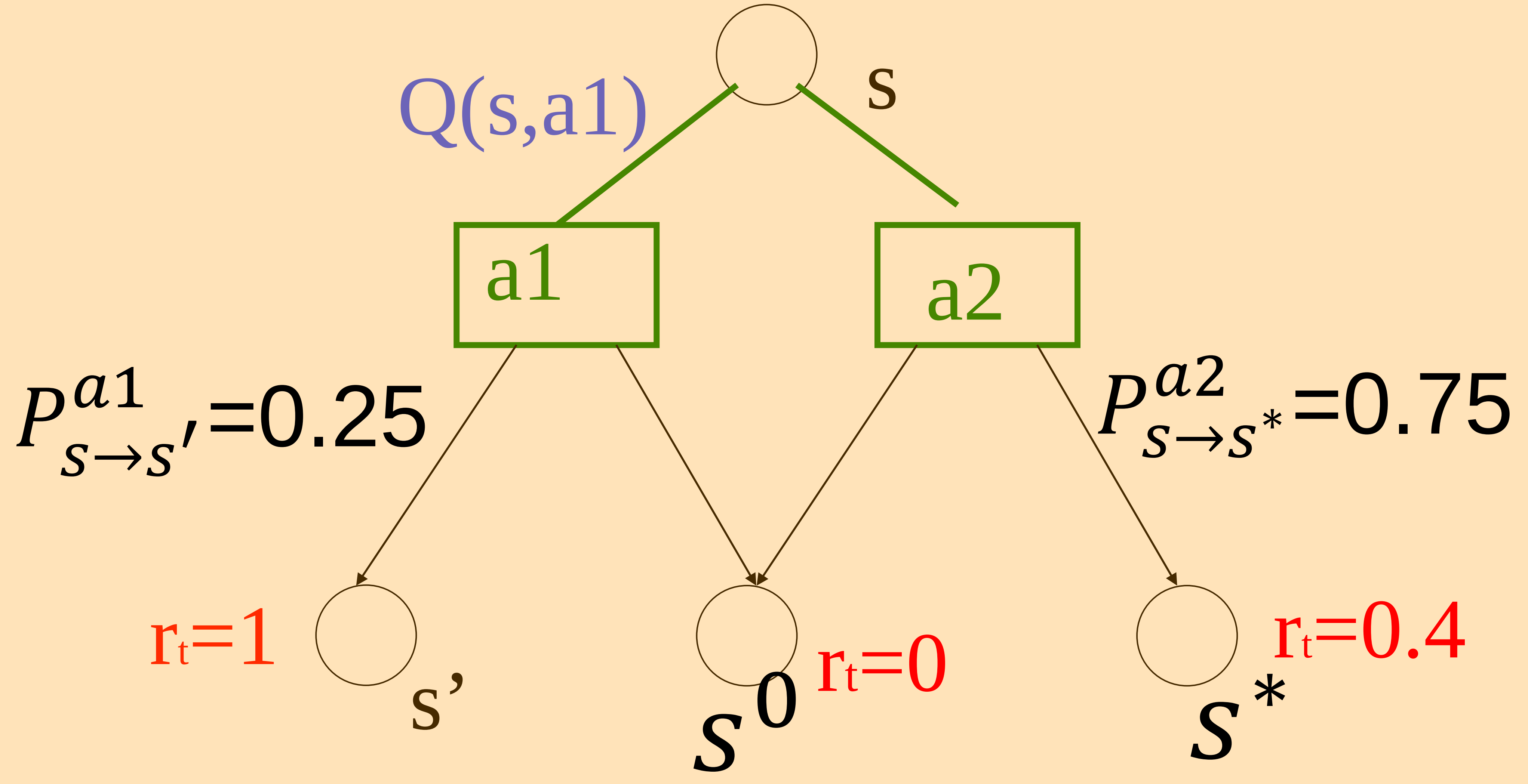
Exploration versus exploitation

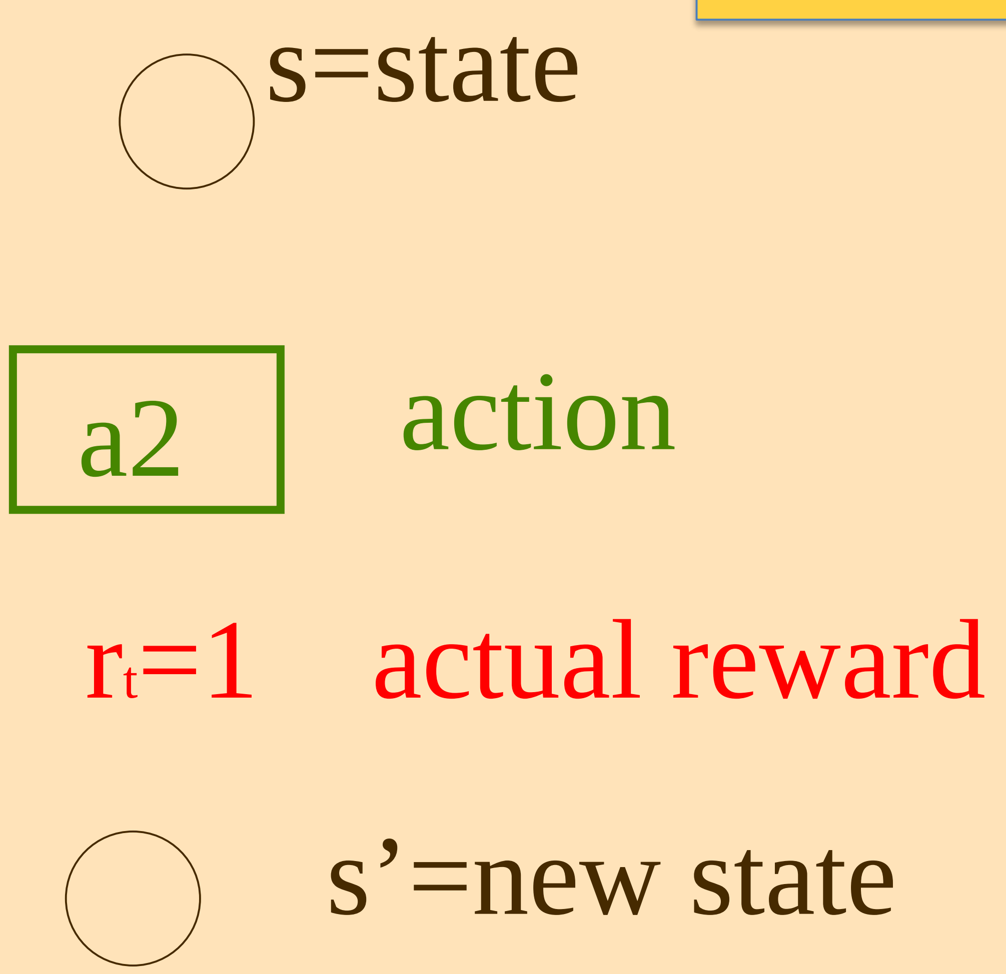
Explore so as to estimate reward probababities

Take action which looks optimal, so as to maximize reward

Exercise 2.a - 2.c now: Exploration-Exploitation

5 min, now!





2.a Assume that you initialize all Q values with zero; set η =0.2 (constant)

Trial 1: you choose action a1, you get r_t=1

Trial 2: you choose action a2, you get r_t=0.4

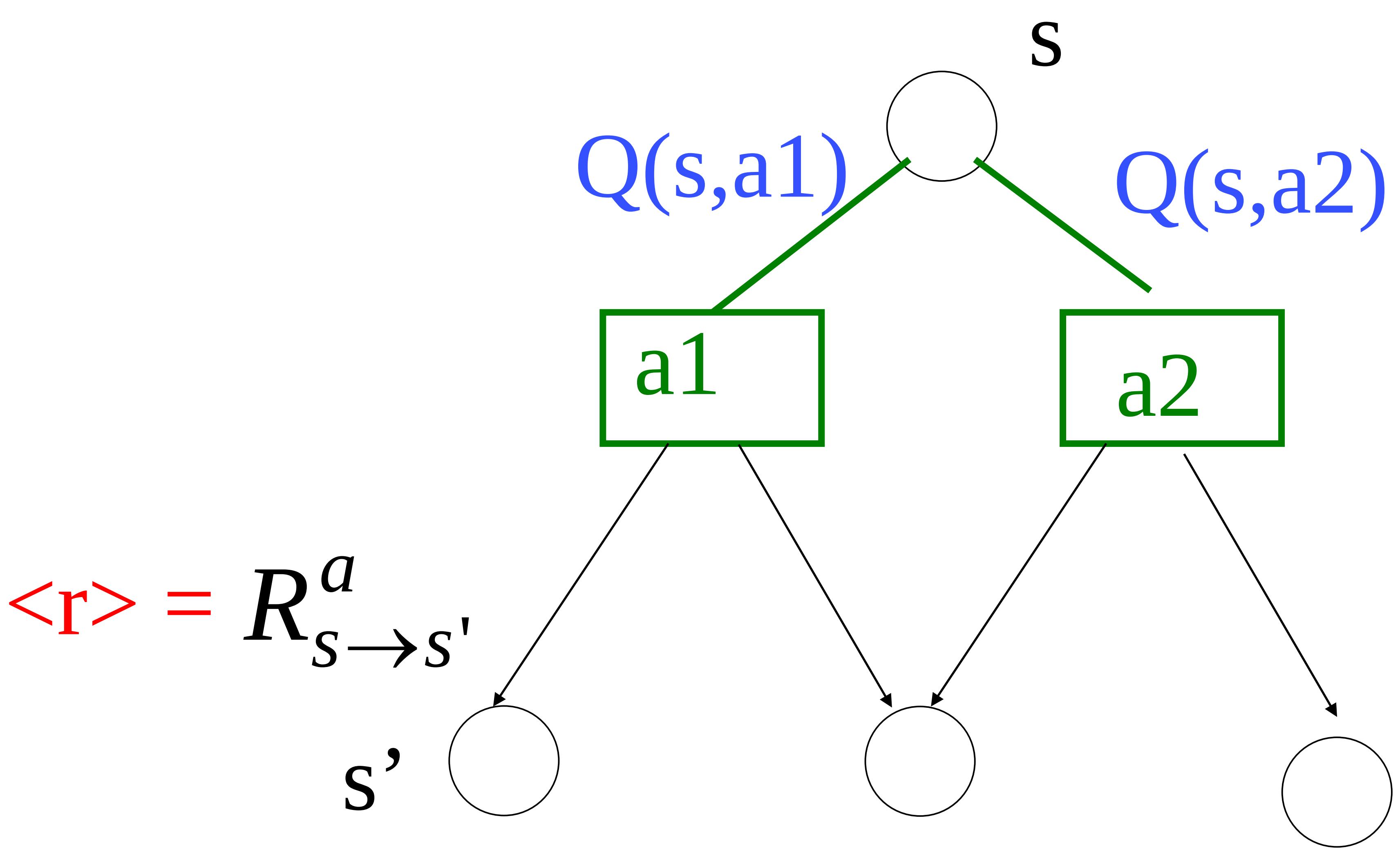
$$\Delta Q(s,a) = \eta\,[r_t - Q(s,a)]$$

2.b Trial 3 – 5: continue 'greedy' (assume that you do not get rewards), update Q

2.c Calculate for both actions the expected reward $Q(s,a)=\sum_{s'} P^{a}_{s\to s'} R^{a}_{s\to s'}$

Blackboard4:
Exercise 2a-2c

3. Exploration and Exploitation



Problem: correct Q values not known

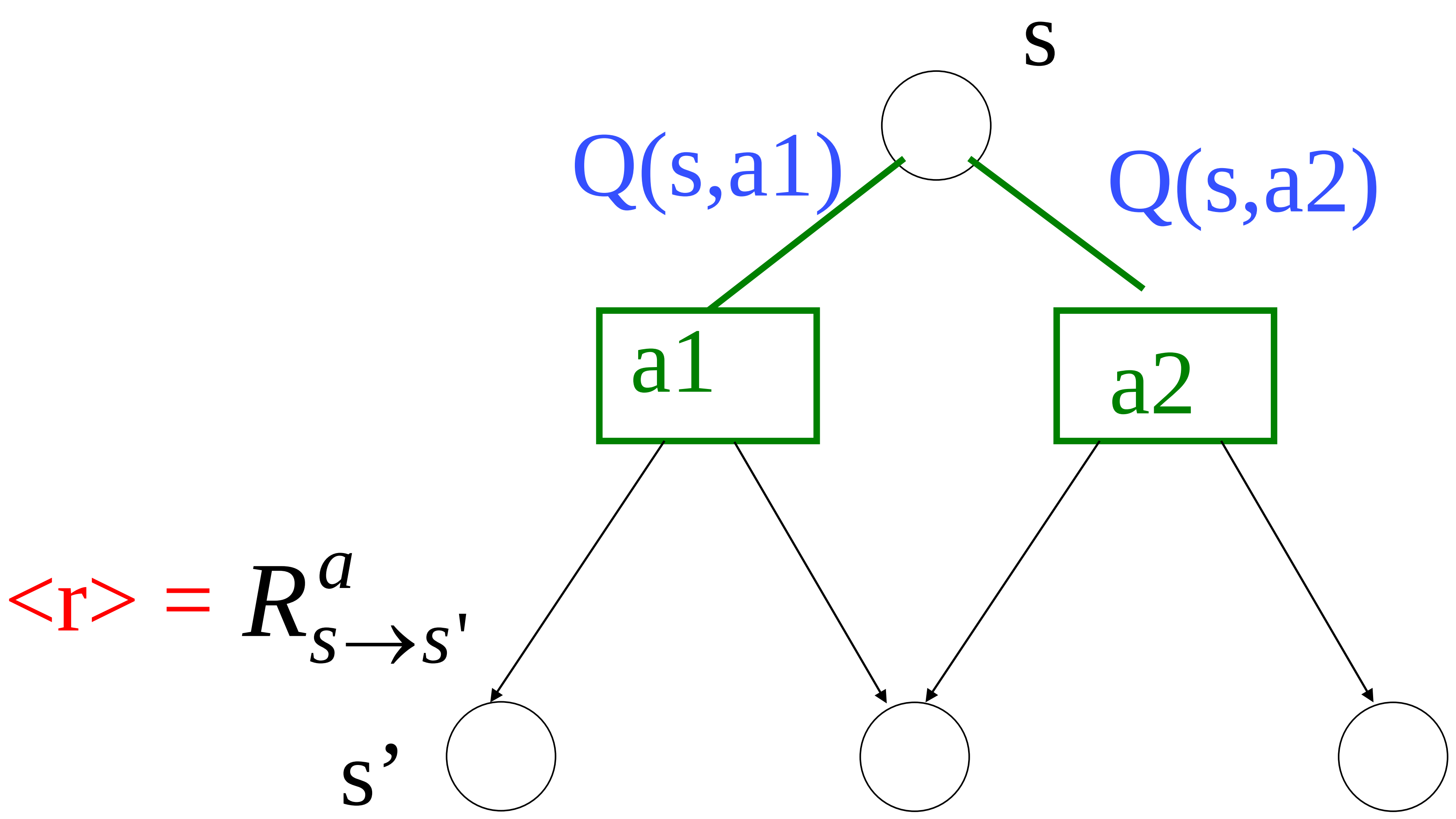
greedy strategy:

- take action a* which looks best

$$Q(s,a^*)>Q(s,a_j)$$

ATTENTION:
with 'greedy' you may get stuck with a sub-optimal strategy

3. Exploration and Exploitation: practical approach



Problem: correct Q values not known

greedy strategy:

- take action a* which looks best

$$Q(s,a^*)>Q(s,a_j)$$

ε-greedy strategy:

- take action a* which looks best with prob $P=1-\varepsilon$

Softmax strategy: take action a' with prob

$$P(a')=\frac{\exp[\beta Q(a')]}{\sum_a \exp[\beta Q(a)]}$$

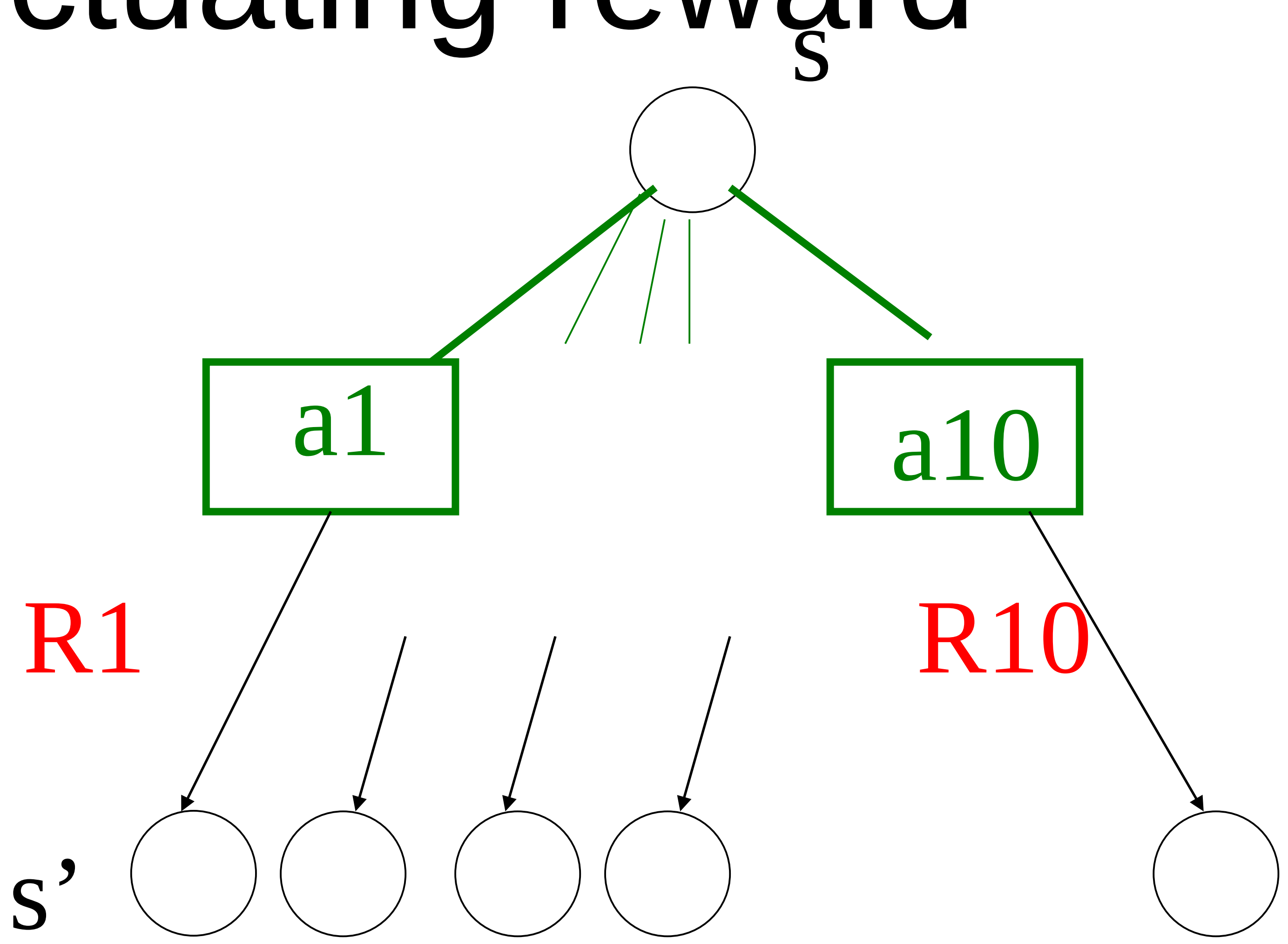
Optimistic greedy:

initialize with Q values that are too big

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

3. Exploration and Exploitation: practical approach

Example: 10-armed bandit with fluctuating reward



in each action, actual rewards fluctuate around a mean

Rk= $R^{ak}_{s \to s'}$

Epsilon-greedy: simulation

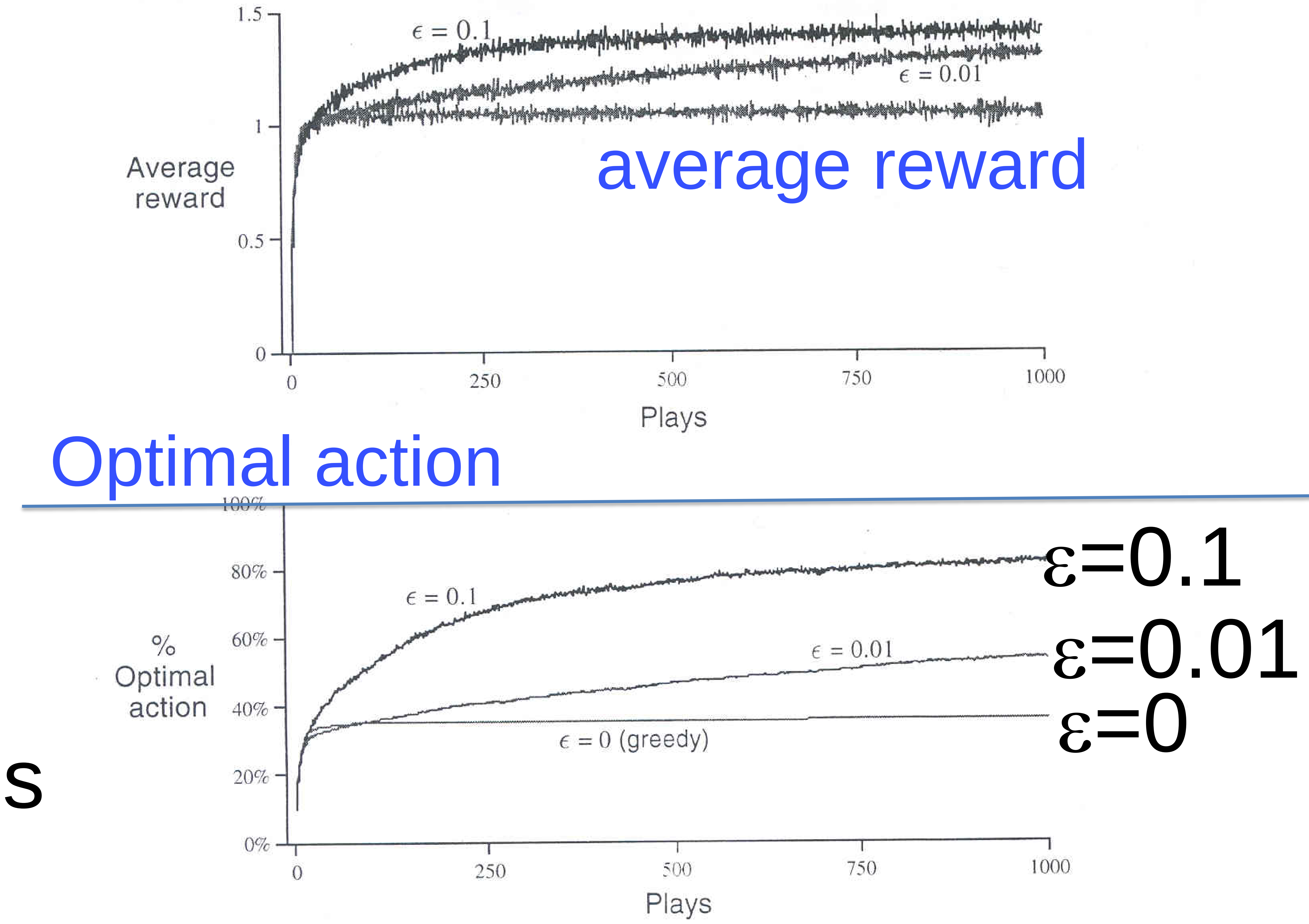


Figure 2.1 Average performance of ϵ-greedy action-value methods on the 10-armed testbed. These data are averages over 2000 tasks. All methods used sample averages as their action-value estimates.

book: Sutton and Barto

3. Exploration and Exploitation: practical approach

Epsilon-greedy, combined with iterative update of Q-values

A simple bandit algorithm

Initialize, for $a = 1$ to k:

$Q(a) \leftarrow 0$

$N(a) \leftarrow 0$

Repeat forever:

$$A \leftarrow \begin{cases} \arg\max_a Q(a) & \text{with probability } 1-\varepsilon \quad \text{(breaking ties randomly)} \\ \text{a random action} & \text{with probability } \varepsilon \end{cases}$$

$R \leftarrow bandit(A)$

$N(A) \leftarrow N(A) + 1$

$Q(A) \leftarrow Q(A) + \frac{1}{N(A)}\left[R - Q(A)\right]$

Sutton and Barto, ch. 2

3. Quiz: Exploration – Exploitation dilemma

We use an iterative method and update Q-values with **eta=0.1**

[ ] With a greedy policy the agent uses the best possible action

[ ] Using an epsilon-greedy method with epsilon = 0.1
means that, even after convergence of Q-values,
in at least 10 percent of cases a suboptimal action is chosen.

[ ] If the rewards in the system are between 0 and 1 and Q-values
are initialized with Q=2, then each action is played at least
5 times before exploitation starts.

3. Quiz: Exploration – Exploitation dilemma

All Q values are initialized with the same value Q=0.1

Rewards in the system are r =0.5 for action 1 (always)
and r=1.0 for action 2 (always)

We use an iterative method and update Q-values with **eta=0.1**

[ ] if we use softmax with beta = 10, then, after 100 steps, action 2 is chosen almost always

[ ] if we use softmax with beta = 0.1, then action 2 is taken about twice as often as action 1.

Softmax strategy: take action a' with prob

$$P(a') = \frac{\exp[\beta Q(a')]}{\sum_a \exp[\beta Q(a)]}$$

Artificial Neural Networks: Lecture 8
Reinforcement Learning and SARSA

1. Learning by Reward: Reinforcement Learning
2. Elements of Reinforcement Learning
3. Exploration vs Exploitation
4. Bellman equation

4. Multistep horizon

Policy $\pi(s, a)$

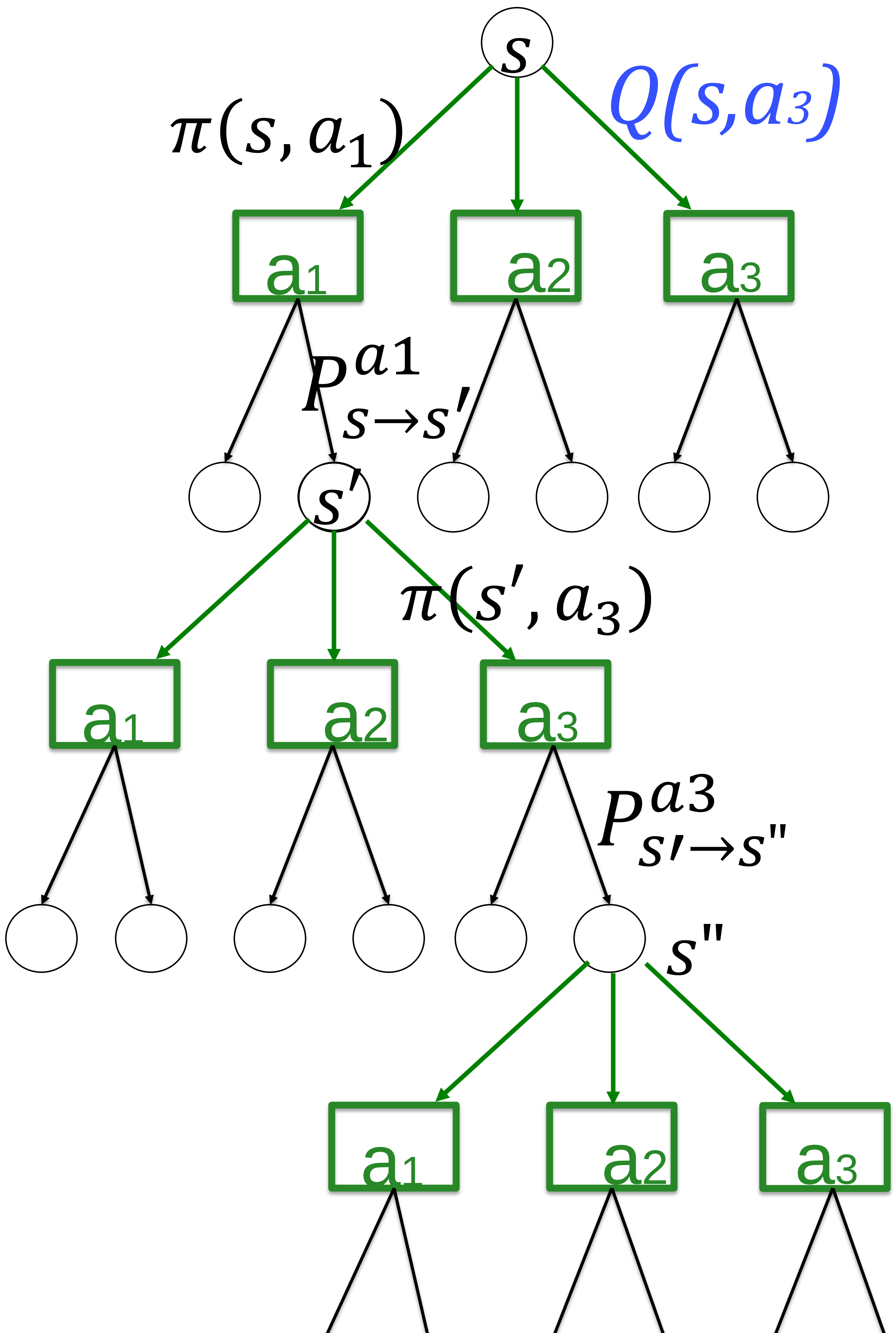
probability to choose action a in state s

$1 = \sum_{a'} \pi(s, a')$

Examples of policy:
-epsilon-greedy
-softmax

Stochasticity $P^{a}_{s \to s'}$

probability to end in state s' taking action a in state s



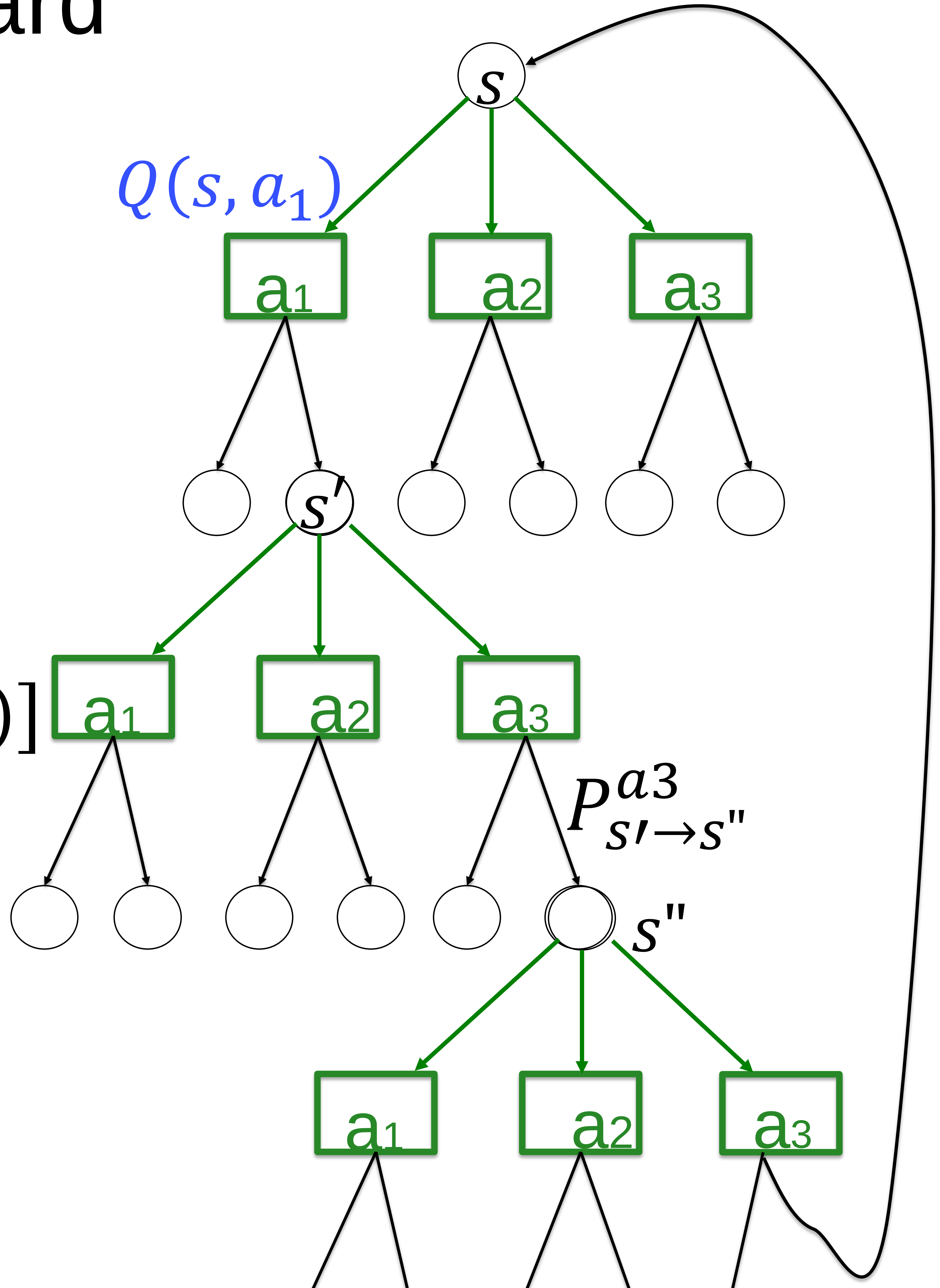
4. Total expected (discounted) reward

Starting in state s with action a

$Q(s,a)$ =

$$= \left\langle r_t \quad + \gamma\, r_{t+1} + \gamma^2 r_{t+2} + \gamma^3\, r_{t+3} + \cdots \right\rangle$$

$$= E[r_t + \gamma\, r_{t+1} + \gamma^2 r_{t+2} + \gamma^3\, r_{t+3} + \ \ldots |s, a)]$$



Discount factor: γ <1

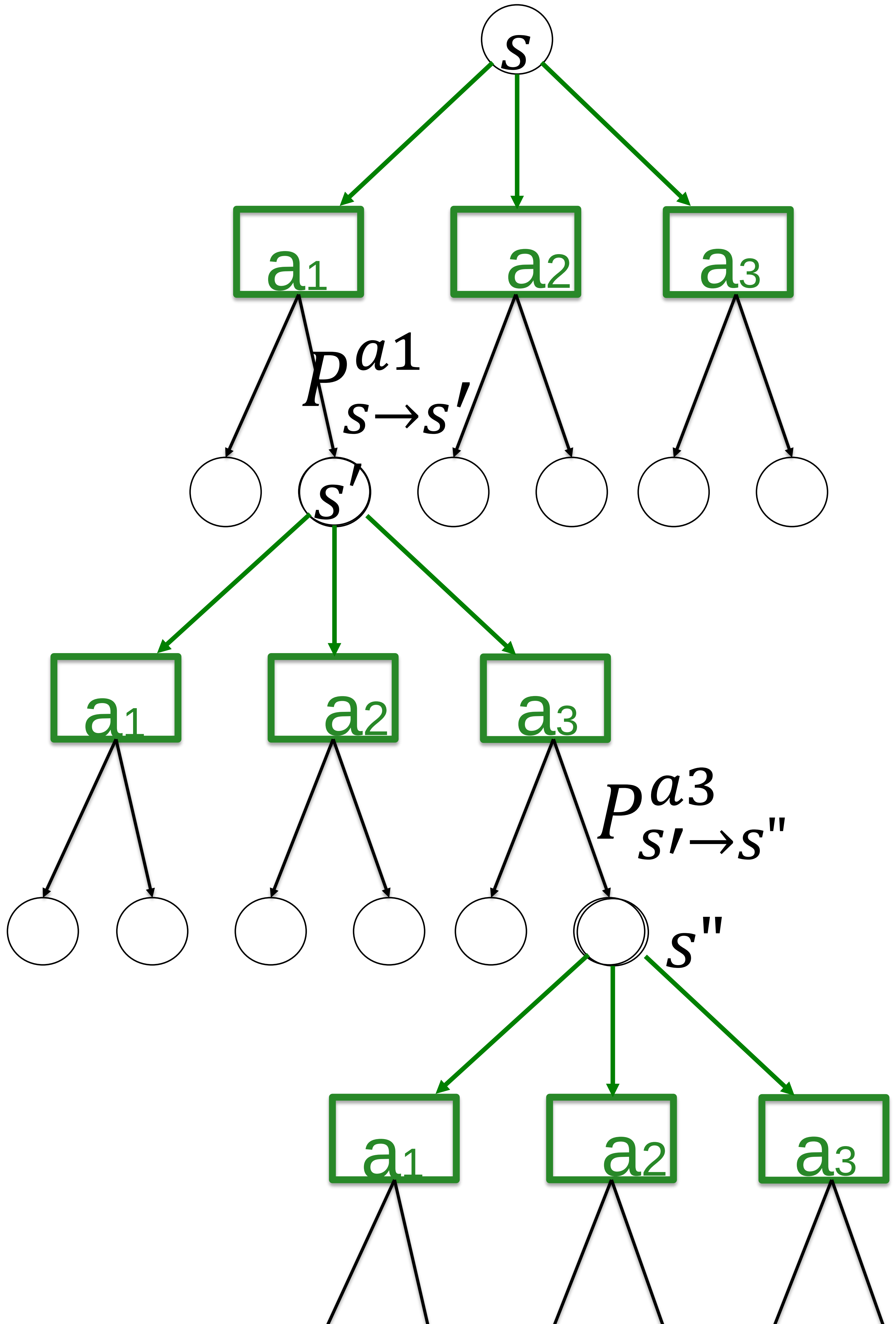
-important for recurrent networks!

-avoids blow-up of summation

-gives less weight to reward in **far** future

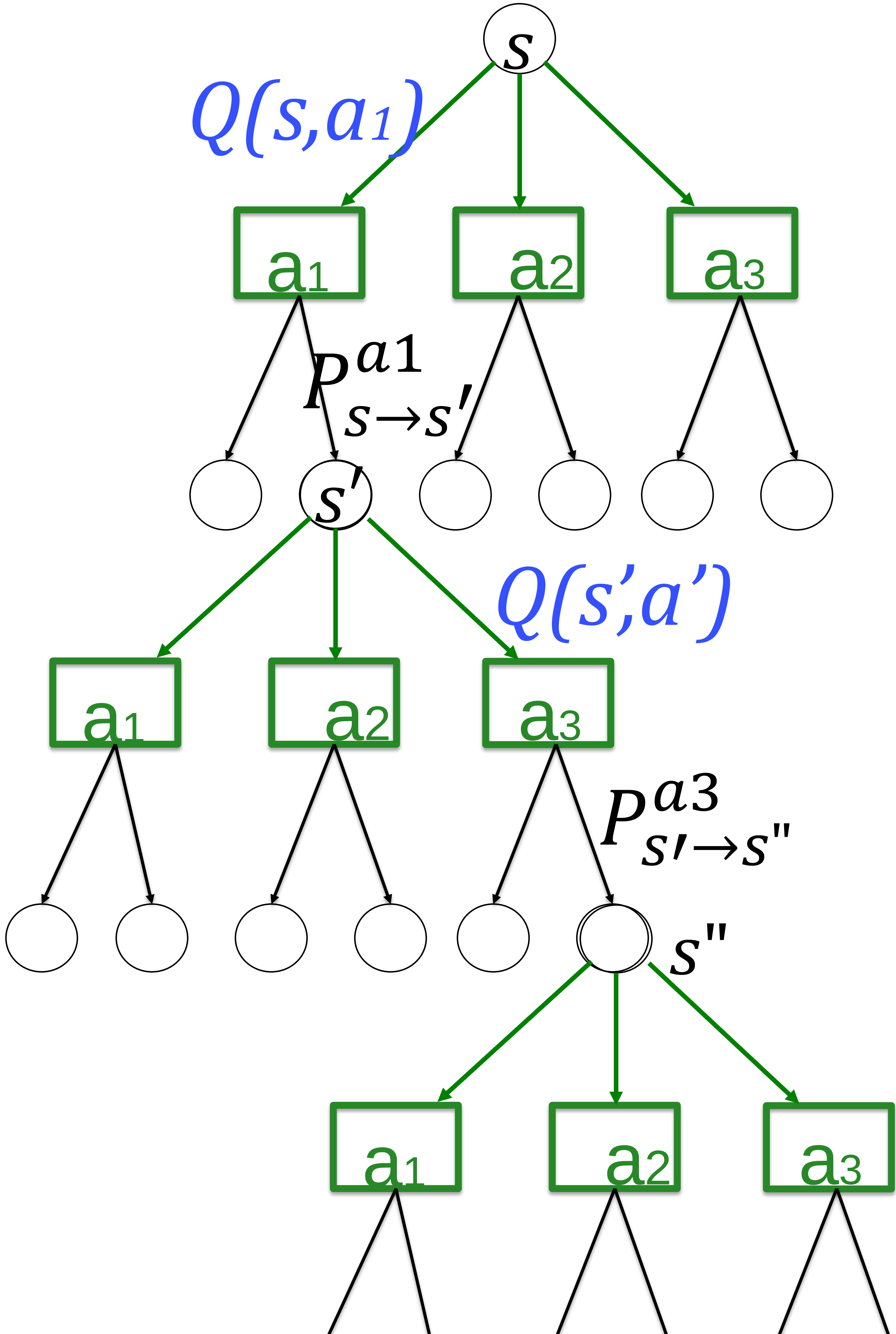
4. Bellman equation

Blackboard5: Bellman eq.



4. Bellman equation with policy π

$$Q(s,a) = \sum_{s'} P^{a}_{s\to s'}\left[R^{a}_{s\to s'} + \gamma \sum_{a'} \pi(s',a') Q(s',a') \right]$$



Bellman equation =
value consistency of
neighboring states

Remark:
Sometimes Bellman equation is written for greedy policy: $\pi(s,a) = \delta_{a,a*}$
with action $a^* = \underset{a'}{\operatorname{argmax}}\, Q(s,a')$

4. Bellman equation (for optimal actions)

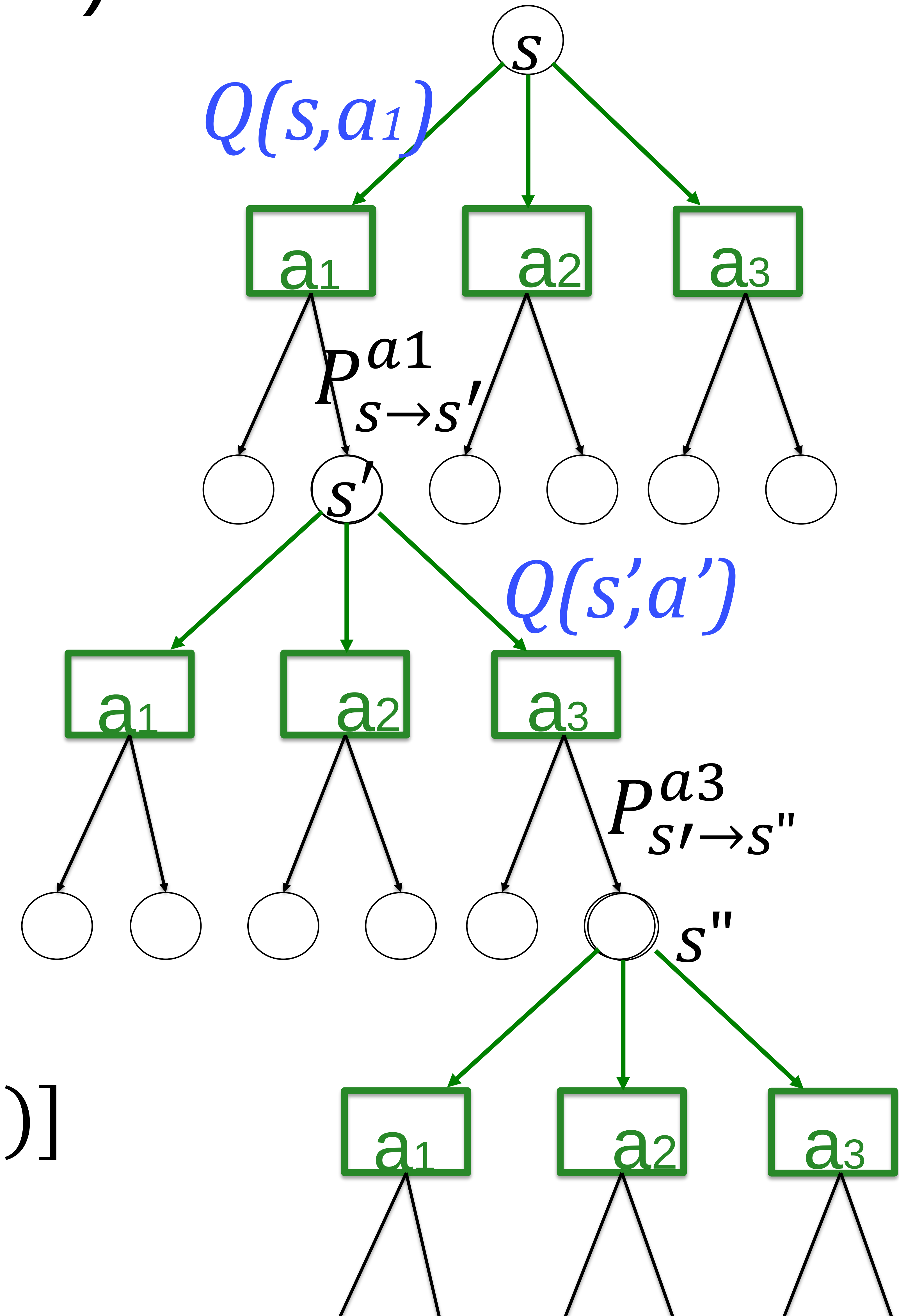
$$Q(s,a) = \sum_{s'} P^{a}_{s\to s'} \left[R^{a}_{s\to s'} + \gamma \underbrace{\sum_{a'} \pi(s',a') Q(s',a')} \right]$$

for greedy policy:

$$\pi(s,a) = \delta_{a,a*}$$

with action $a^* = \underset{a'}{\operatorname{argmax}}\, Q(s,a')$

$$Q(s,a) = \sum_{s'} P^{a}_{s'\to s''} \; [\, R^{a}_{s'\to s''} + \gamma \max_{a'} Q(s'a')]$$

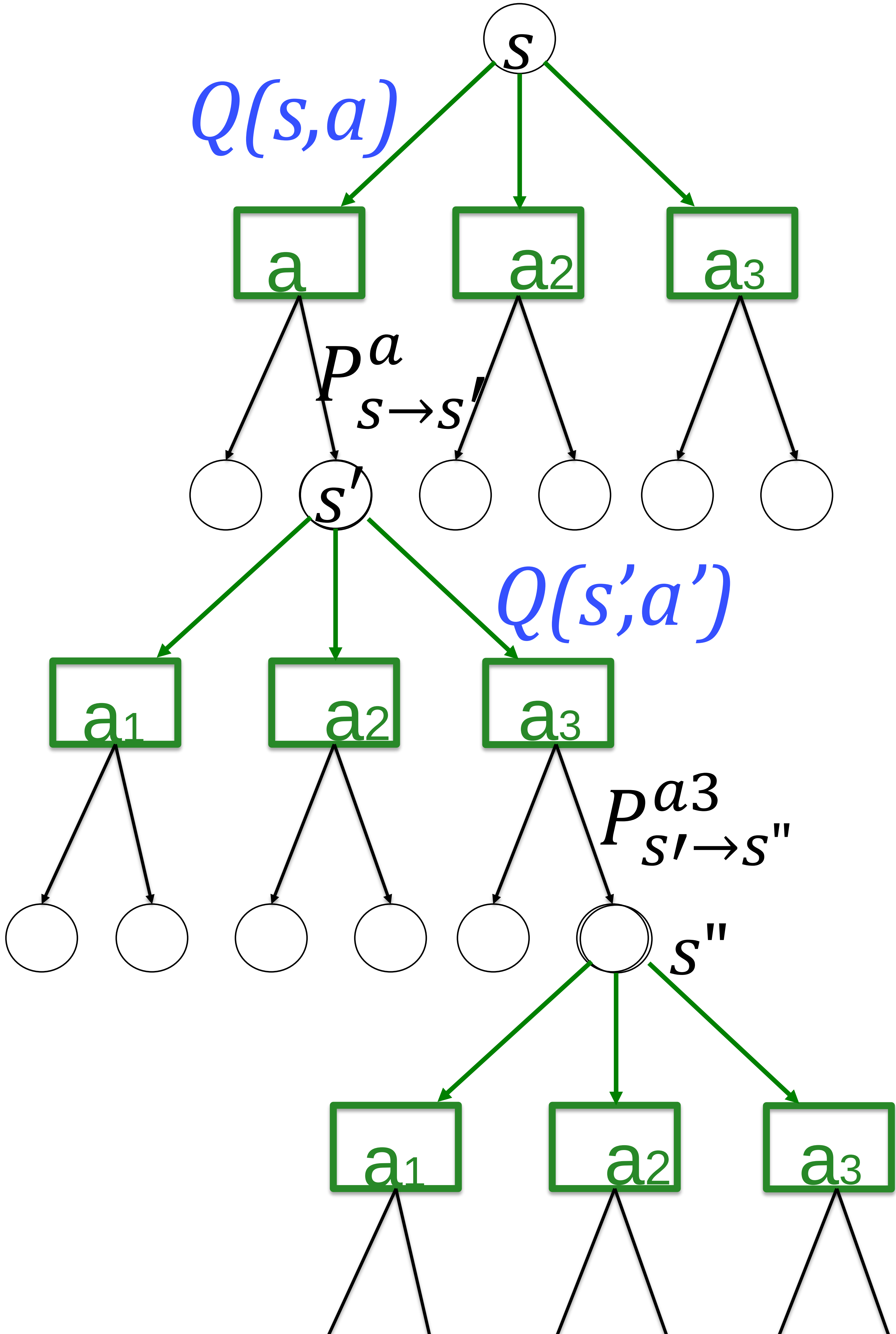


4. Quiz: Bellman equation with policy π

$$Q(s,a) = \sum_{s'} P^{a}_{s\to s'} \left[R^{a}_{s\to s'} + \gamma \sum_{a'} \pi(s',a') Q(s',a') \right]$$

[ ] The Bellman equation is linear in the variables $Q(s'a')$

[ ] The set of variables $Q(s',a')$ that solve the Bellman equation is unique and does not depend on the policy

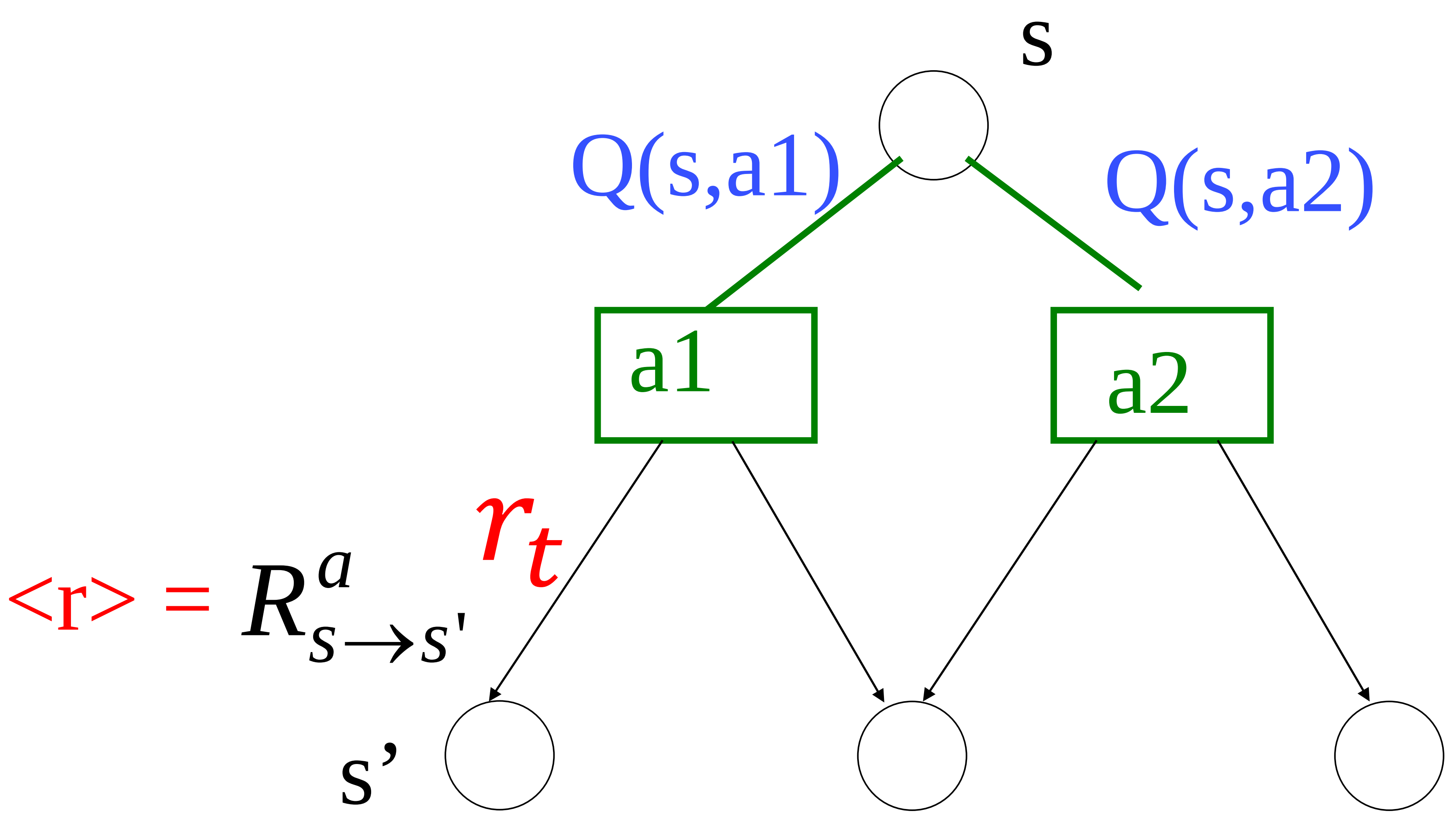


Artificial Neural Networks: Lecture 8
Reinforcement Learning and SARSA

1. Learning by Reward: Reinforcement Learning
2. Elements of Reinforcement Learning
3. Exploration vs Exploitation
4. Bellman equation
5. SARSA algorithm

3. Iterative update of Q-values

Problem: Q-values not given



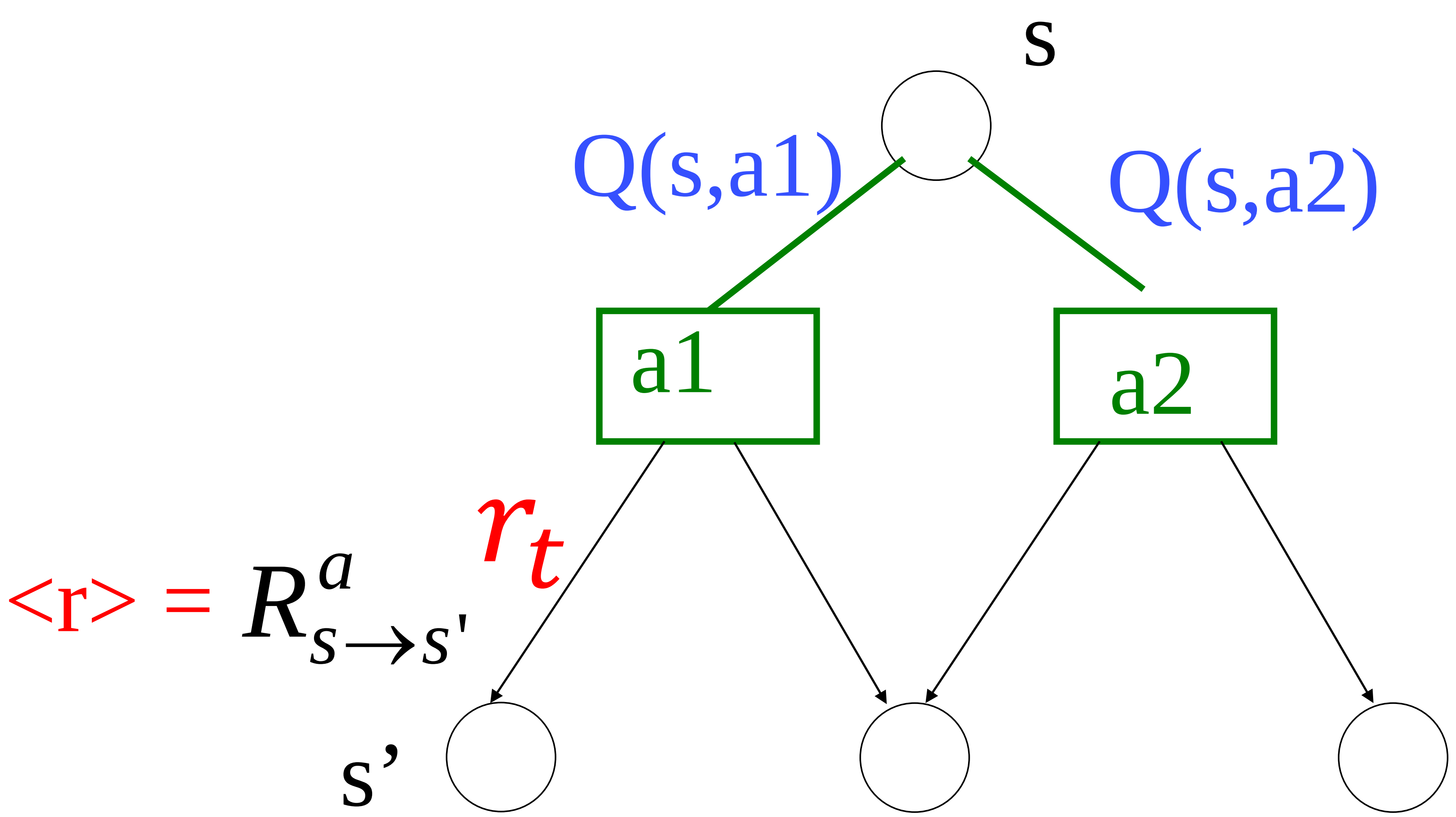
Solution: iterative update

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

while playing with policy $\pi(s,a)$

5. Iterative update of Q-values for multistep environments

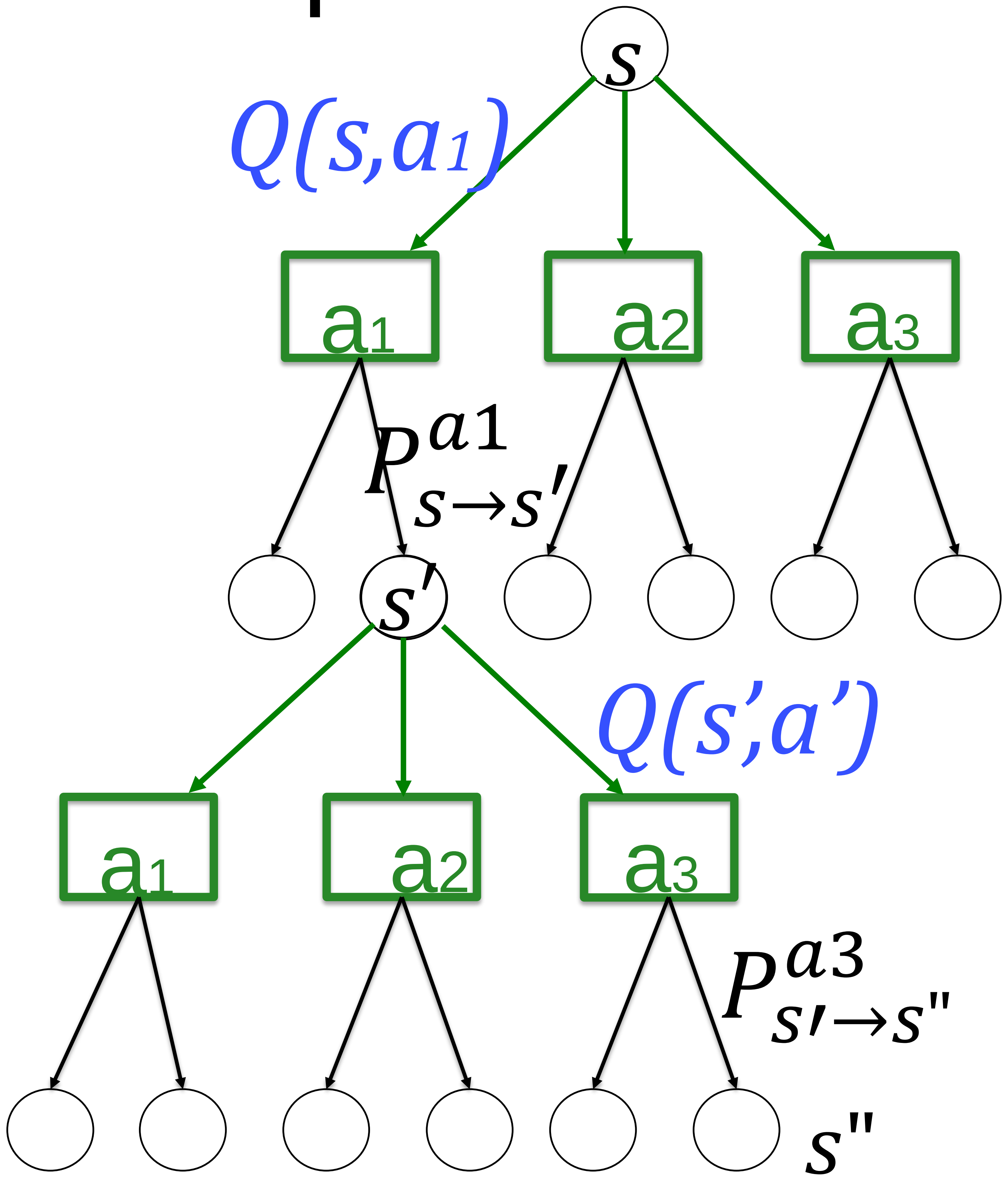
Problem: Q-values not given



Solution: iterative update

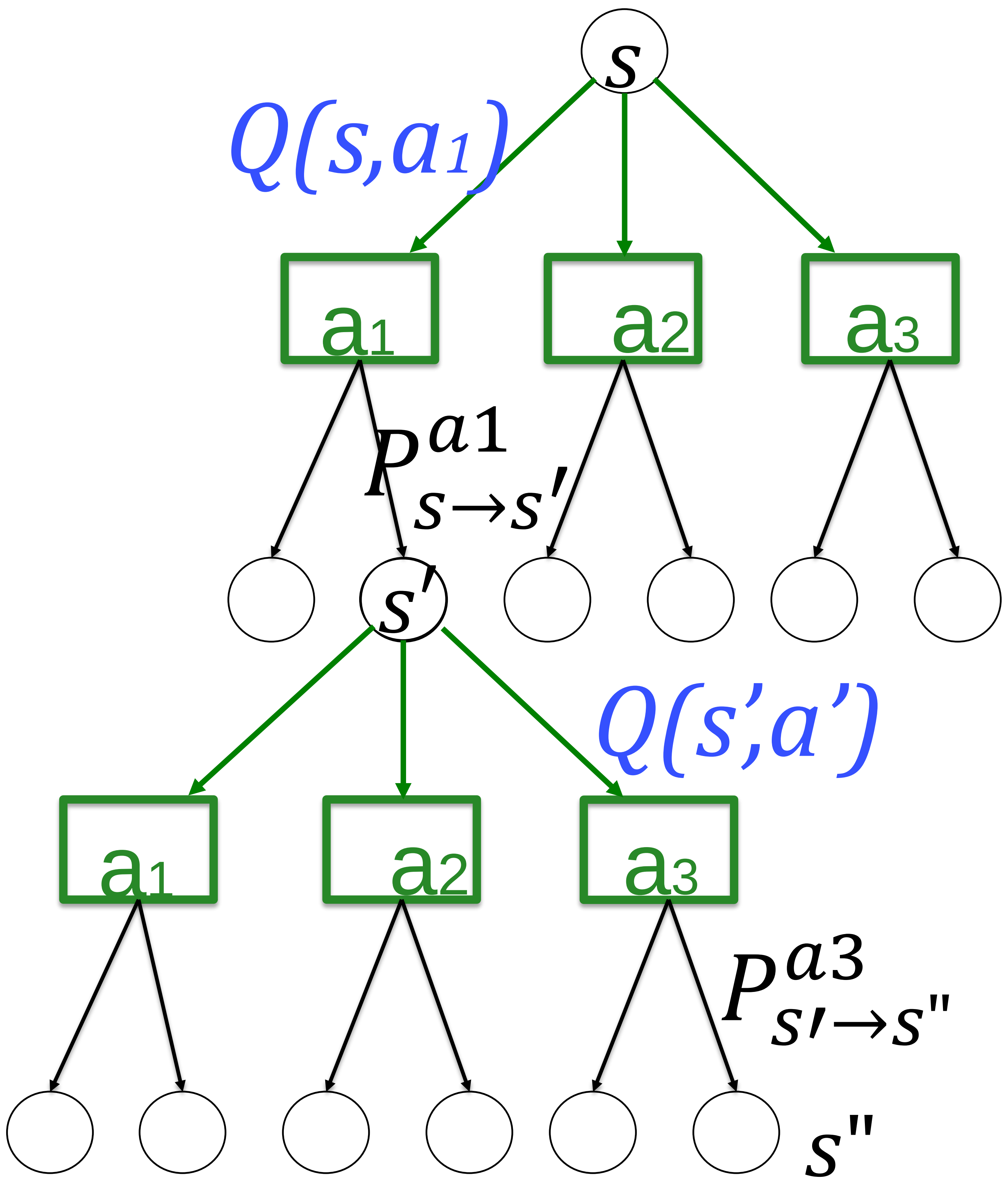
$$\Delta Q(s,a) = \eta\ [r_t - Q(s,a)]$$

while playing with policy $\pi(s,a)$



$$\Delta Q(s,a) = \ ?$$

Blackboard6: SARSA update



5. Iterative update of Q-values for multistep environments

Bellman equation:

$$Q(s,a) = \sum_{s'} P^a_{s\to s'} \left[R^a_{s\to s'} + \gamma \sum_{a'} \pi(s',a') Q(s',a') \right]$$

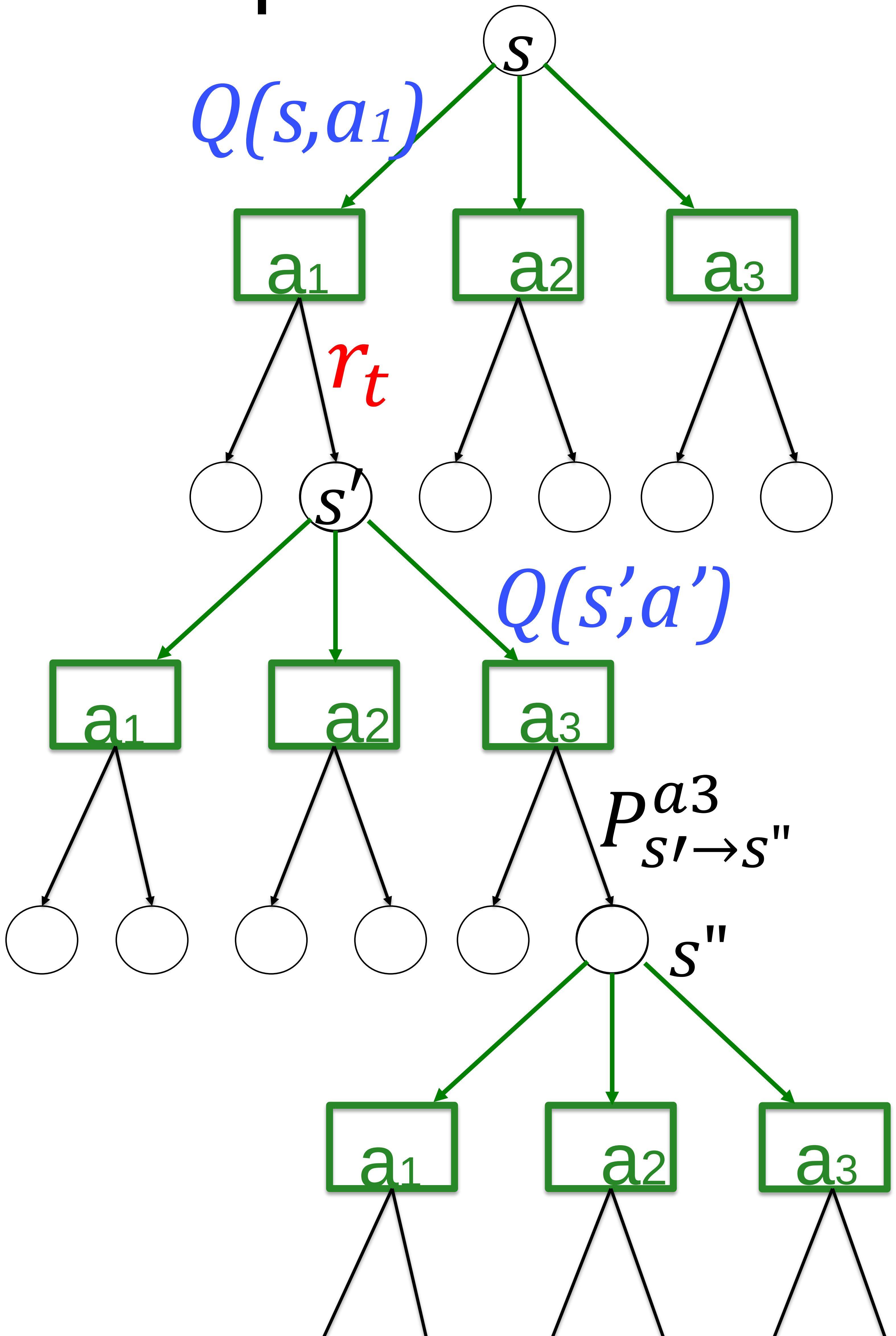
Problem:

- Q-values not given
- branching probabilities not given
- reward probabilities not given

Solution: iterative update

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

while playing with policy $\pi(s,a)$



5. SARSA vs. Bellman equation

$$Q(s,a) = \sum_{s'} P^{a}_{s\to s'}\left[R^{a}_{s\to s'} + \gamma \sum_{a'} \pi(s',a') Q(s',a') \right]$$

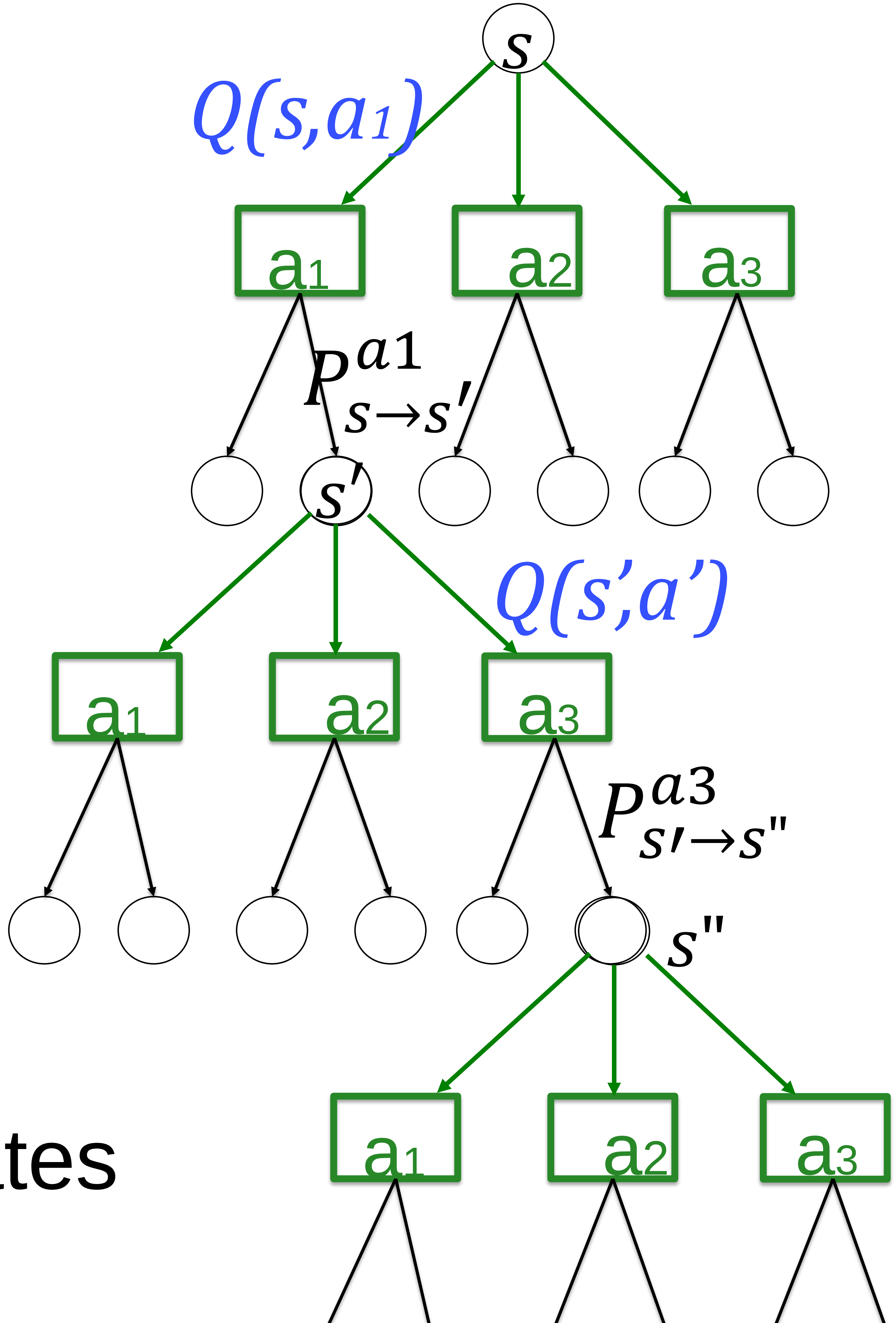
Bellman equation

= consistency of Q-values across neighboring states

SARSA update rule

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

= make Q-values of neighboring states more consistent



5. SARSA algorithm

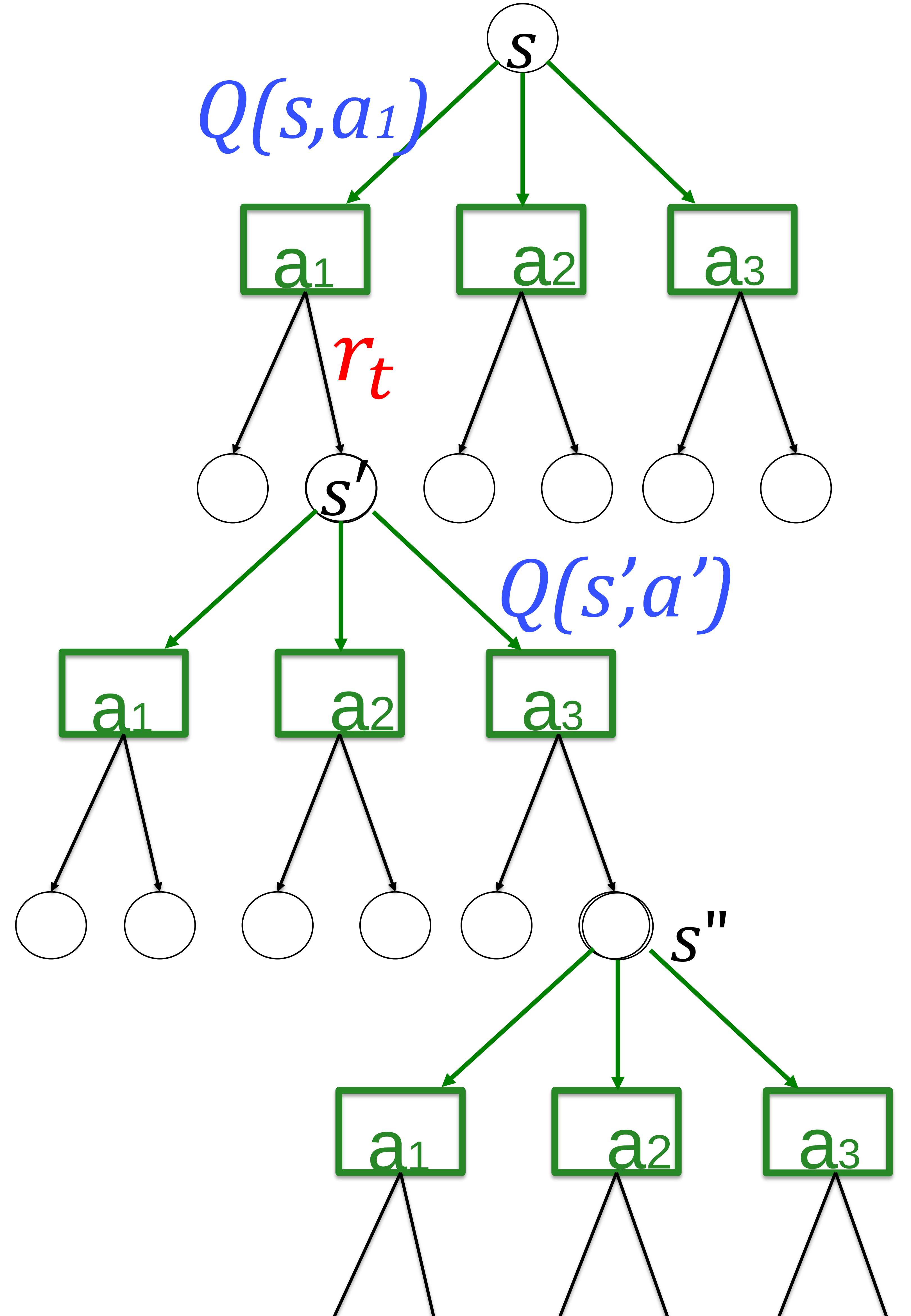
Initialise Q values
Start from initial state s

1) being in state **s**
   choose action **a**
   [according to policy $\pi(s,a)$]
2) Observe reward **r**
   and next state **s'**
3) Choose action **a'** in state s'
   [according to policy $\pi(s,a)$]
4) Update with SARSA update rule

$$\Delta Q(s,a) = [r_t + \gamma Q(s',a') - Q(s,a)]$$

5) set: s ← s'; a ← a'
6) Goto 1)

Stop when all Q-values have converged



Exercise now, 8 min (at home)

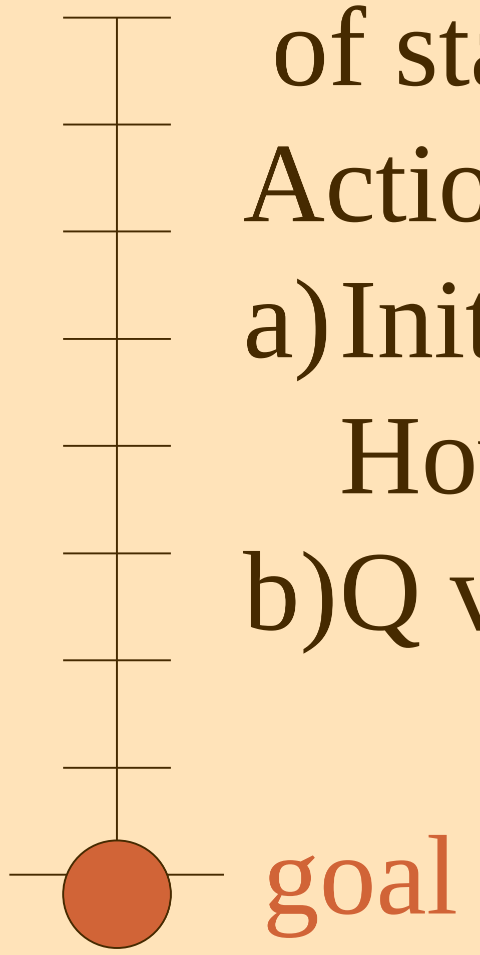
- **Update of Q values in SARSA**

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

- **policy for action choice:**

  Pick most often action

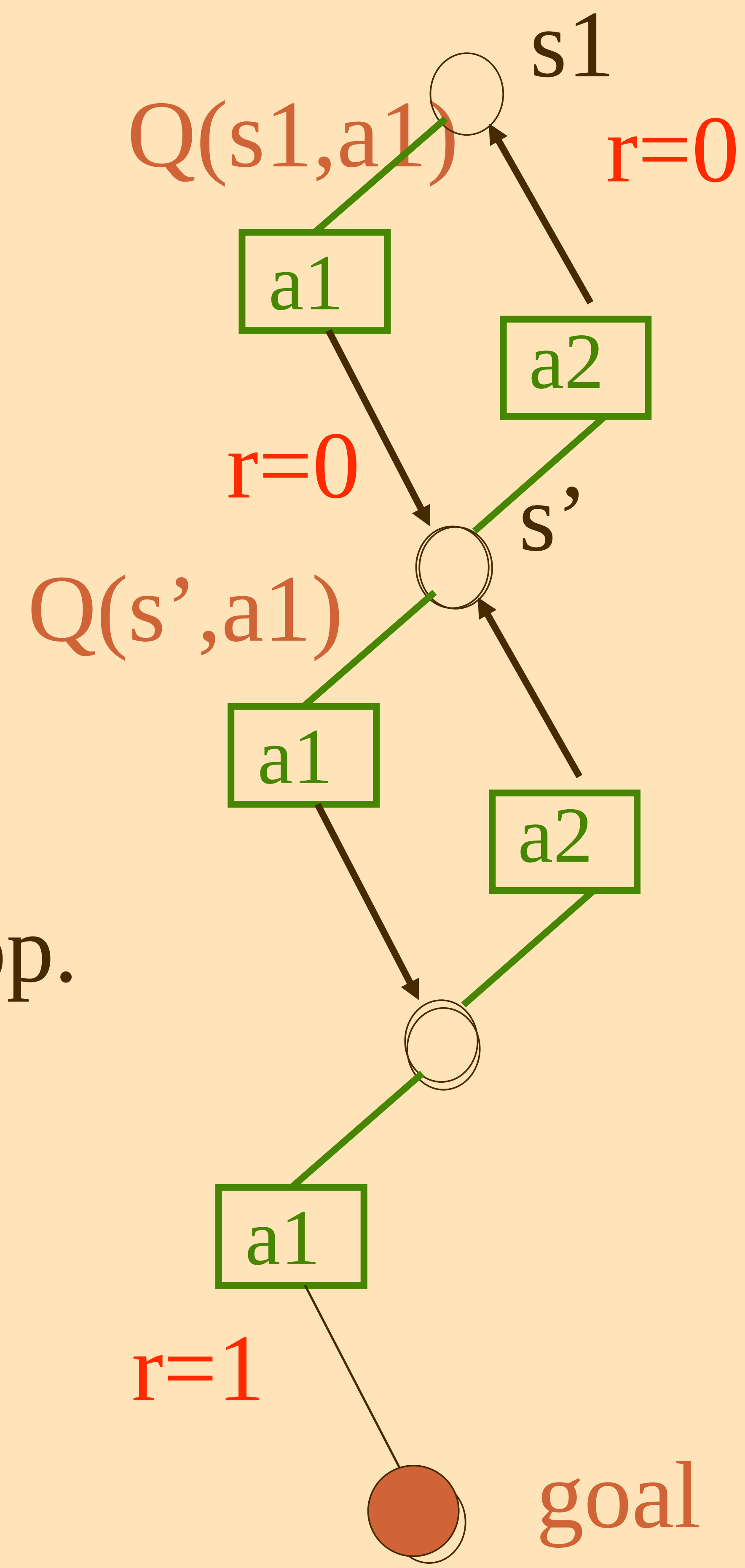
$$a_t^* = \arg\max_a Q_a(s,a)$$



Consider a linear sequence of states. Reward only at goal. Actions are up or down.

a) Initialise Q values at 0. Start at top. How do Q values develop?

b) Q values after 2 complete trials?



5. Convergence in expectation of SARSA: theorem

Assumption:

The SARSA algo has been applied for a very long time, using updates

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

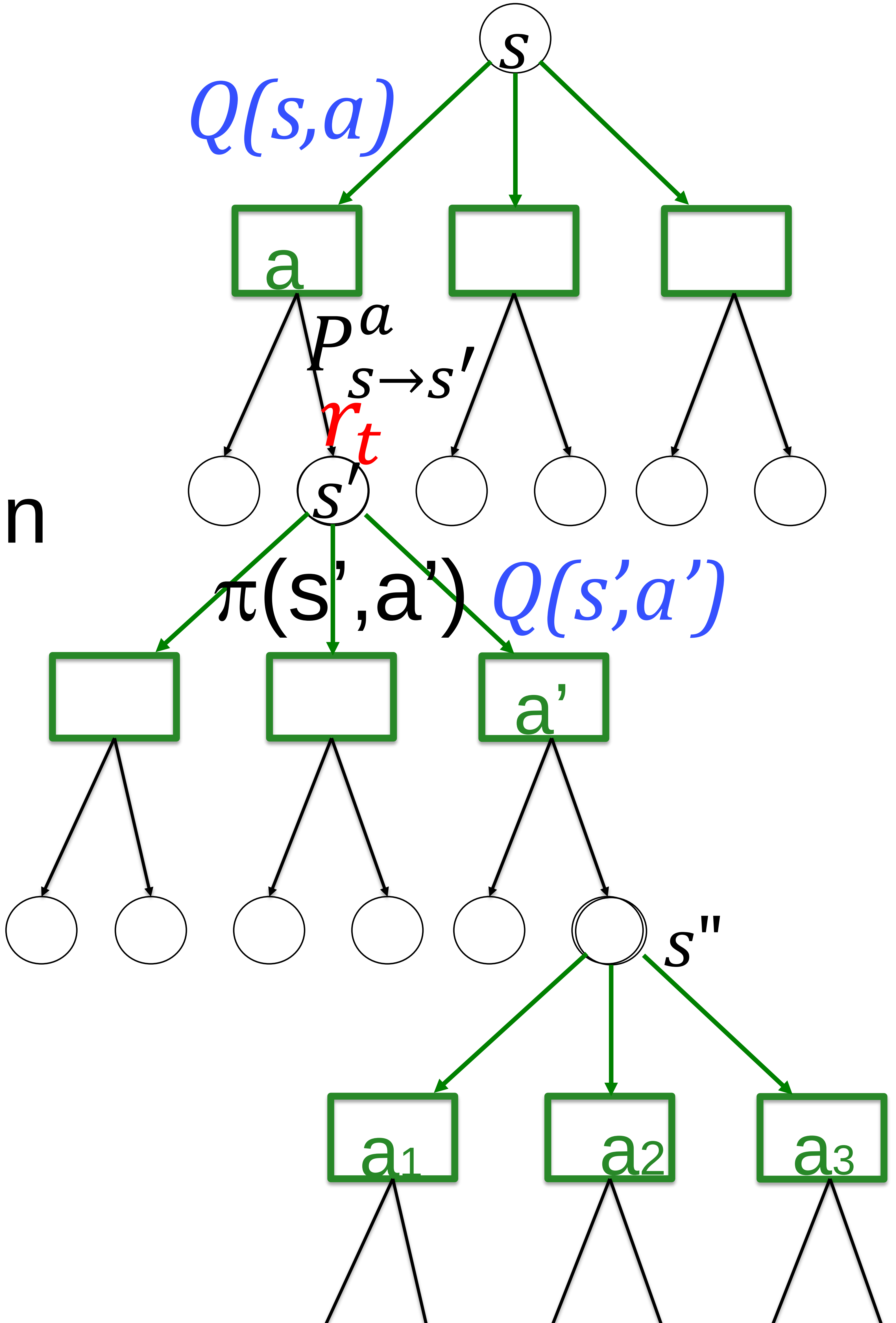
IF all Q-values have converged in expectation

$$\langle \Delta Q(s,a) \rangle = 0$$

THEN

The set of Q-values solves the Bellman eq.

$$Q(s,a) = \sum_{s'} P^a_{s\to s'}\left[R^a_{s\to s'} + \gamma \sum_{a'} \pi(s',a') Q(s',a')\right]$$



5. Convergence in expectation of SARSA: theorem

Blackboard7:
SARSA convergence

Look at graph to take expectations:
- if algo in state s, all remaining expectations are “given s”
- if algo on a branch (s,a), all remaining exp. are “given s and a”

5. SARSA algorithm

We have initialized SARSA and played for n>2 steps.
Is the following true?

[ ] in SARSA, updates are applied after each move.

[ ] in SARSA, the agent updates the Q-value $Q(s(t), a(t))$ related to the **current** state $s(t)$

[ ] in SARSA, the agent updates the Q-value $Q(s(t-1), a(t-1))$ related to the **previous** state, when it is in state $s(t)$

[ ] in SARSA, the agent moves in the environment using the policy $\pi(s, a)$

[ ] SARSA is an online algorithm

Reinforcement Learning and SARSA

Learning outcome and conclusions:

- **Reinforcement Learning is learning by rewards**
  - → world is full of rewards (but not full of labels)
- **Agents and actions**
  - → agent learns by interacting with the environment
  - → state s, action a, reward r
- **Exploration vs Exploitation**
  - → optimal actions are easy if we know reward probabilities
  - → since we don’t know the probabilities we need to explore
- **Bellman equation**
  - → self-consistency condition for Q-values
- **SARSA algorithm:** state-action-reward-state-action
  - → update while exploring environment with current policy