Artificial Neural Networks: Lecture 10

Policy Gradient Methods

Wulfram Gerstner

EPFL, Lausanne, Switzerland

1. Review Policy gradient
2. Review Subtracting the mean via the value function
3. Actor-Critic
4. Eligibility traces for policy gradient
5. Actor-Critic in the Brain
6. Application: Rat navigation
7. Model-based versus Model-free

Artificial Neural Networks: Lecture 12

Reinforcement Learning and the Brain

Wulfram Gerstner

EPFL, Lausanne, Switzerland

Objectives for today:

- Actor-critic

- three-factor learning rules can be implemented by the brain

- eligibility traces as ‘candidate parameter updates’

- the dopamine signal has signature of the TD error

- model-based versus model-free

Today we finish at around 12:45

Reading for this week:

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

Chapter: 15

Background reading:

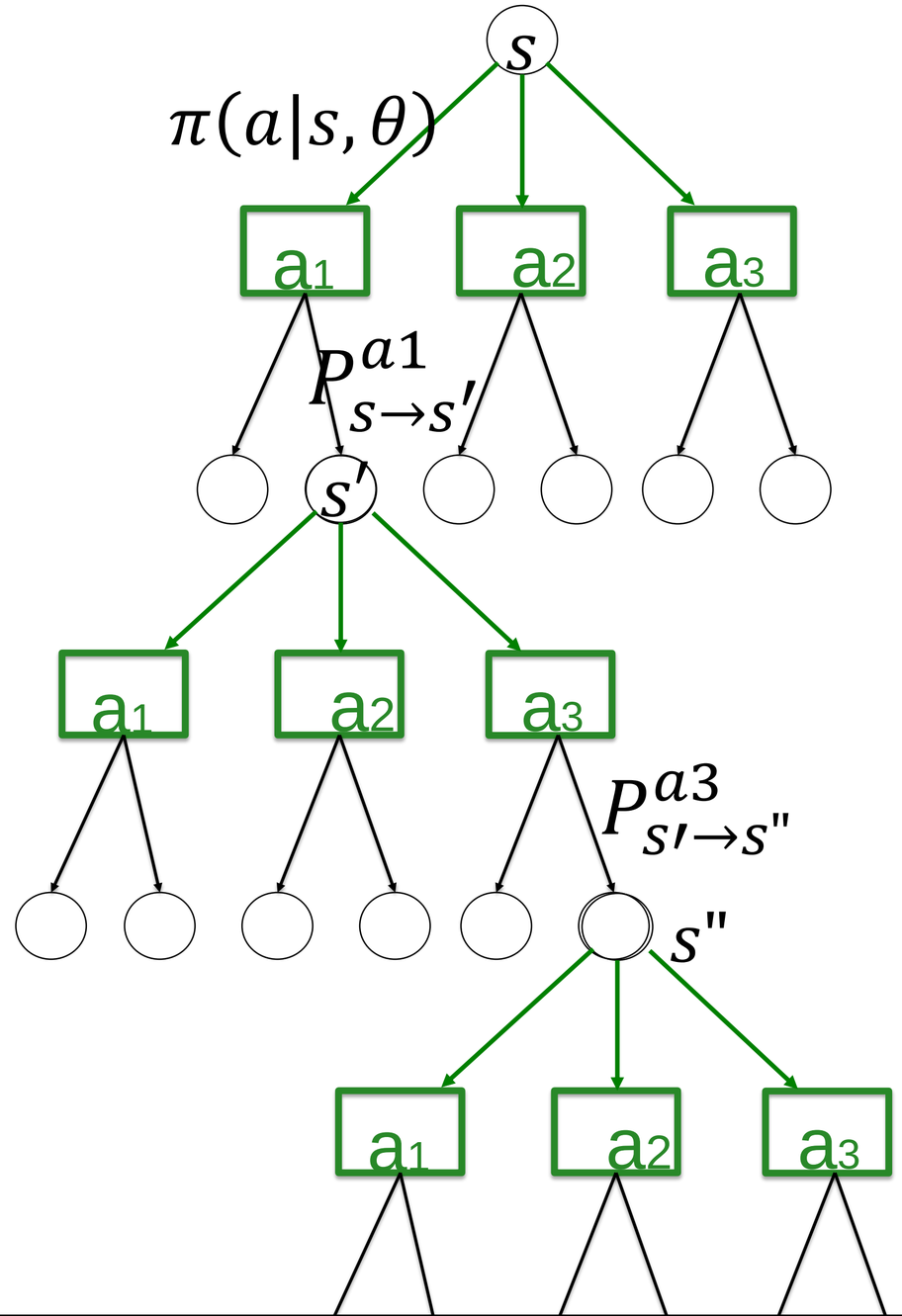
(1) Fremaux, Sprekeler, Gerstner (2013) Reinforcement learning using a continuous-time actor-critic framework with spiking neurons *PLOS Computational Biol. doi:10.1371/journal.pcbi.1003024*

(2) Fremaux, Gerstner (2016) Neuromodulated spike-timing-dependent plasticity, and theory of three-factor learning rules *Frontiers in neural circuits 9, 85*

(3) *J Gläscher, N Daw, P Dayan, JP O'Doherty (2010)* States versus rewards: dissociable neural prediction error signals underlying model-based and model-free reinforcement learning*, Neuron 66 (4), 585-595*

1. Review Policy Gradient

Aim:
update the parameters θ
of the policy $\pi(a|s,\theta)$



1. Review Policy Gradient

Calculation yields several terms of the form

Total accumulated discounted reward collected in one episode starting at s_t, a_t

$$\Delta\theta_j \propto [R^{a_t}_{s_t \to s_{end}}] \frac{d}{d\theta_j} \ln[\pi(a_t|s_t, \theta)]$$

$$+\gamma[R^{a_{t+1}}_{s_{t+1} \to s_{end}}] \frac{d}{d\theta_j} \ln[\pi(a_{t+1}|s_{t+1}, \theta)]$$

$$+ \ldots$$

(previous slide)
We consider a single episode that started in state s_t with action a_t and ends after several steps in the terminal state s_{end}
The result of the calculation gives an update rule for each of the parameters.
The update of the parameter θ_j contains several terms.
(i) the first term is proportional to the total accumulated (discounted) reward, also called return $R^{a_t}_{s_t \to s_{end}}$
(ii) the second term is proportional to gamma times the total accumulated (discounted) reward but starting in state s_{t+1}
(iii) the third term is proportional to gamma-squared times the total accumulated (discounted) reward but starting in state s_{t+2}
(iv)

We can think of this update as one update step for one episode. Analogous to the terminology last week, Sutton and Barto call this the Monte-Carlo update for one episode.
The log-likelihood trick was explained last week. Since this is a sampling based approach (1 episode=1 sample) each of the terms is proportional to ln π,

Artificial Neural Networks: Lecture 10

Policy Gradient Methods

Wulfram Gerstner

EPFL, Lausanne, Switzerland

1. Review Policy gradient
2. Review Subtracting the mean via the value function

(previous slide)

As discussed last week and in the exercise session, subtracting the mean of a variable helps to stabilize the algorithm.

There are two different ways to do this.

(i) Subtract the mean return (=value V) in a multistep-horizon algorithm (or the mean reward in a one step-horizon algorithm).
This is what we consider here in this section

(ii) Subtract mean expected reward PER TIME STEP (related to the delta-error) in a multi-step horizon algorithm.
This is what we will consider in section 3 under the term Actor-Critic.

2. Subtract a baseline

we derived this online gradient rule for multi-step horizon

$$\Delta\theta_j \propto \left[R^{a_t}_{s_t \to s_{end}}\right] \frac{d}{d\theta_j} \ln[\pi(a_t|s_t,\theta)]$$

But then this rule is also an online gradient rule

$$\Delta\theta_j \propto \left[R^{a_t}_{s_t \to s_{end}} - \boldsymbol{b}(\boldsymbol{s_t})\right] \frac{d}{d\theta_j} \ln[\pi(a_t|s_t,\theta)]$$

with the same expectation

(because a baseline shift drops out if we take the gradient)

(previous slide)
Please remember that the full update rule for the parameter θ_j
in a multi-step episode contains several terms of this form; here only the first of these terms is shown.

Similar to the case of the one-step horizon, we can subtract a bias b from the return $R^{a_t}_{s_t \to s_{end}}$ without changing the location of the maximum of the total expected return.

Moreover, this bias $b(s_t)$ can itself depend on the state s_t.
Thus the update rule now has terms

$$\Delta\theta_j \propto [R^{a_t}_{s_t \to s_{end}} - b(s_t)] \frac{d}{d\theta_j} \ln[\pi(a_t|s_t, \theta)]$$
$$+\gamma[R^{a_{t+1}}_{s_{t+1} \to s_{end}} - b(s_{t+1})] \frac{d}{d\theta_j} \ln[\pi(a_{t+1}|s_{t+1}, \theta)]$$
$$+\gamma^2[R^{a_{t+2}}_{s_{t+2} \to s_{end}} - b(s_{t+2})] \frac{d}{d\theta_j} \ln[\pi(a_{t+2}|s_{t+2}, \theta)]$$
$$+ \ldots$$

2. subtract a reward baseline

Total accumulated discounted reward collected in one episode starting at s_t, a_t

$$\Delta\theta_j \propto \left[R^{a_t}_{s_t \to s_{end}} - b(s_t)\right] \frac{d}{d\theta_j} \ln[\pi(a_t|s_t,\theta)] + \ldots$$

- The bias b can depend on state s
- Good choice is b ='mean of $[R^{a_t}_{s_t \to s_{end}}]$'

→ take $b(s_t) = V(s_t)$

→ learn value function $V(s)$

(previous slide
Is there a choice of the bias $b(s_t)$ that is particularly good?

One attractive choice is to take the bias equal to the expectation (or empirical mean). The logic is that if you take an action that gives more accumulated discounted reward than your empirical mean in the past, then this action was good and should be reinforced.
If you take an action that gives less accumulated discounted reward than your empirical mean in the past, then this action was not good and should be weakened.

But what is the expected discounted accumulated reward? This is, by definition, exactly the value of the state. Hence a good choice is to subtract the V-value.

And here is where finally the idea of Bellman equation and TD learning comes in through the backdoor: we can learn the V-value, and then use it as a bias in policy gradient.

2. Review: Deep reinforcement learning: alpha-zero

Network for choosing action

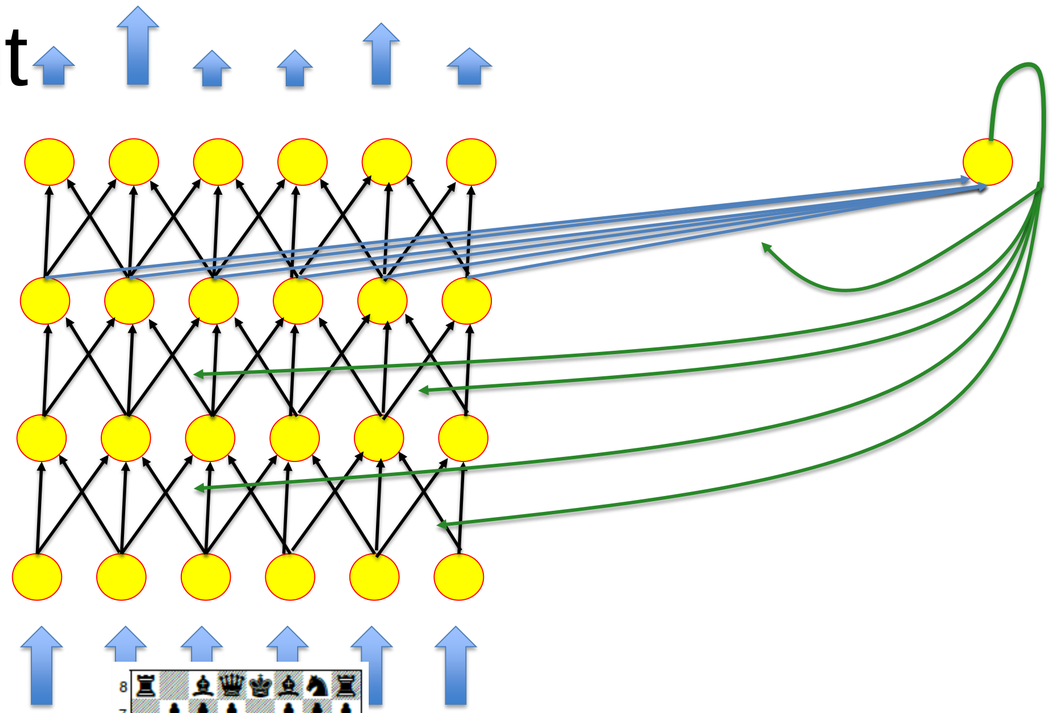
action:

Advance king

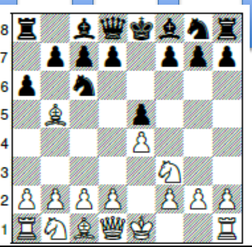
2e output for **value** of state:

$V(s)$

output



input



learning:

→ change connections

aims:

- learn value $V(s)$ of position
- learn action policy to win

Learning signal:

- η[actual Return - $V(s)$]

(previous slide)
Very schematically is this one of the ideas of deep reinforcement learning.

1) We construct a deep network with multiple layers. We use the output units for action choice and optimize the parameters via policy gradient.

2) We have a further output unit to estimate the V-value, and use it as a bias.

3) Update with

$$[R^{a_t}_{s_t \to s_{end}} - V(s_t)] \frac{d}{d\theta_j} \ln[\pi(a_t|s_t, \theta)] + \ldots$$

The model of the V-value can share some units with the model of the actions …

2. Review: Learning two Neural Networks: actor and value

Actions:

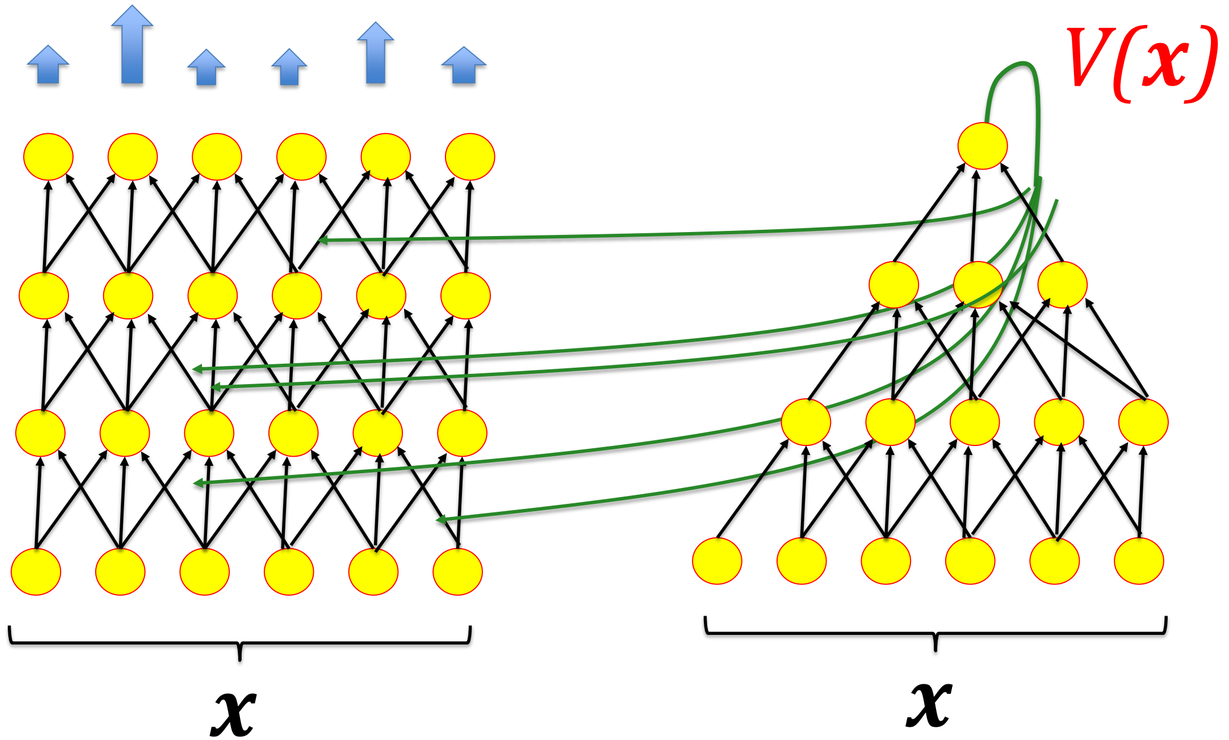
-Learned by
Policy gradient

- Uses $V(x)$ as baseline

Value function:

- Estimated by Monte-Carlo

-provides baseline $b=V(x)$
for action learning



x = states from episode:

$s_t, s_{t+1}, s_{t+2},$

(previous slide)
In the latter case we have two networks:

The actor network learns a first set of parameters, called θ in the algorithm of Sutton and Barto.
The value network learns a second set of parameters, with the label w .

The value $b(x = s_{t+n}) = V(x)$ is the **estimated** total accumulated discounted reward of an episode starting at $x = s_{t+n}$

The weights of the network implementing V(x) can be learned by Monte-Carlo sampling the return: you go from state s until the end, accumulate rewards, and calculate the average over all episodes that have started from (or transited through) the same state s. (See Backup-diagrams and Monte-Carlo of week 9).

The total accumulated discounted ACTUAL reward in ONE episode is $R^{a_{t+n}}_{s_{t+n} \to s_{end}}$
What matters is the difference $[R^{a_t}_{s_t \to s_{end}} - V(s_t)]$

Artificial Neural Networks: Lecture 11

Policy Gradient Methods

Wulfram Gerstner

EPFL, Lausanne, Switzerland

1. Review Policy gradient
2. Review Subtracting the mean via the value function
3. Actor-Critic

(previous slide)
We continue with the idea of two different type of outputs:
A set of actions a_k , and a value V.

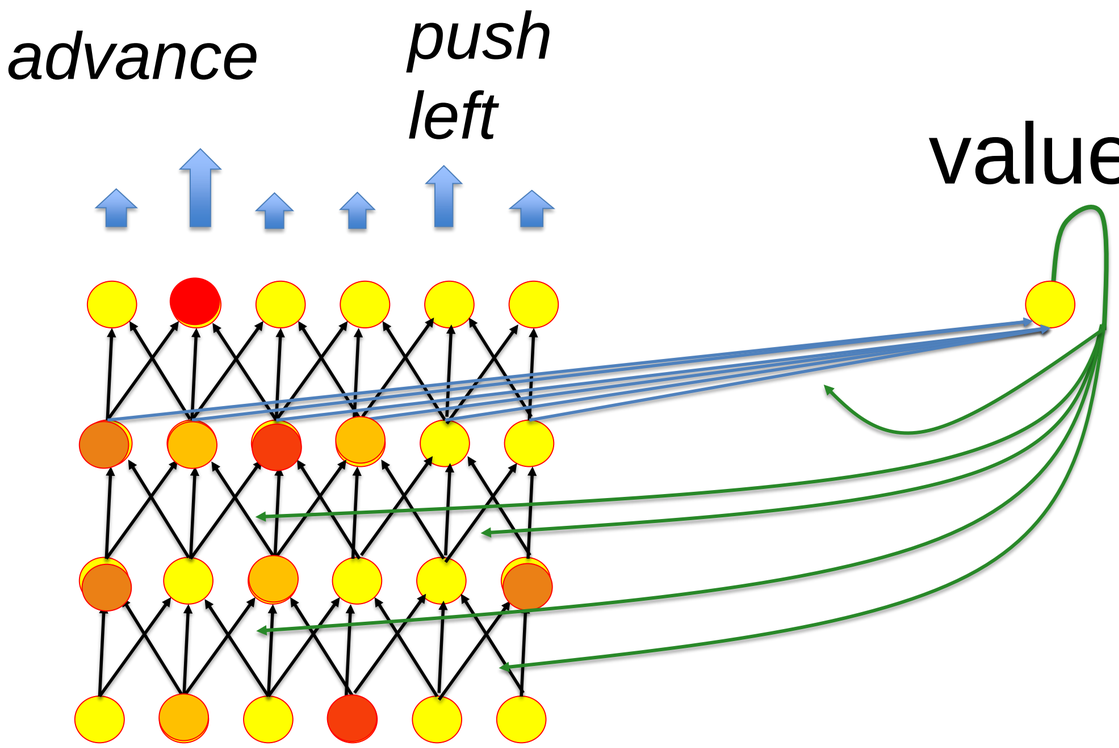
However, for the estimation of the V-value we now use the ‘bootstrapping’ provided by TD algorithm (see last week) rather than the simple Monte-Carlo estimation of the (discounted) accumulated rewards in a single episode.

The networks structure remains the same as before:
An actor (action network) and a critic (value function).

Sutton and Barto reserve the term ‘actor-critic’ to the network where V-values are learned with a TD algorithm.
However, other people would also call the network that we saw in section 6 as an actor-critic network and the one that we study now is then called ‘advantage actor critic’.

3. Actor-Critic = ‘REINFORCE’ with TD bootstrapping

- Estimate $V(s)$
- learn via TD error



(previous slide)
Bottom right: Recall from the TD algorithms that the updates of the weights are proportional to the TD error δ

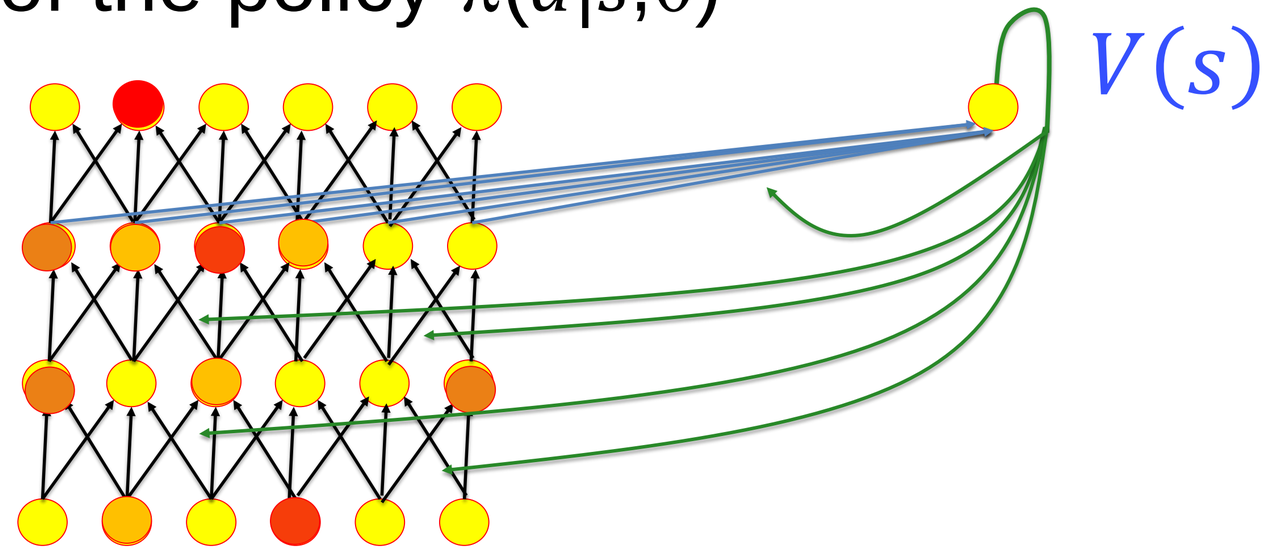
In the actor-critic algorithm the TD error is now also used as the learning signal for the policy gradient:

TD error: The current reward r_t at time step t
is compared with the expected reward for this time step [$V(s_t) - \gamma V(\quad)s_t$]

[Note the difference to the algorithm in section 6:
There the total accumulated discounted reward $R^{a_t}_{s_t \to s_{end}}$
was compared with $V(s_t)$]

3. ACTOR-CRITIC (versus REINFORCE with baseline)

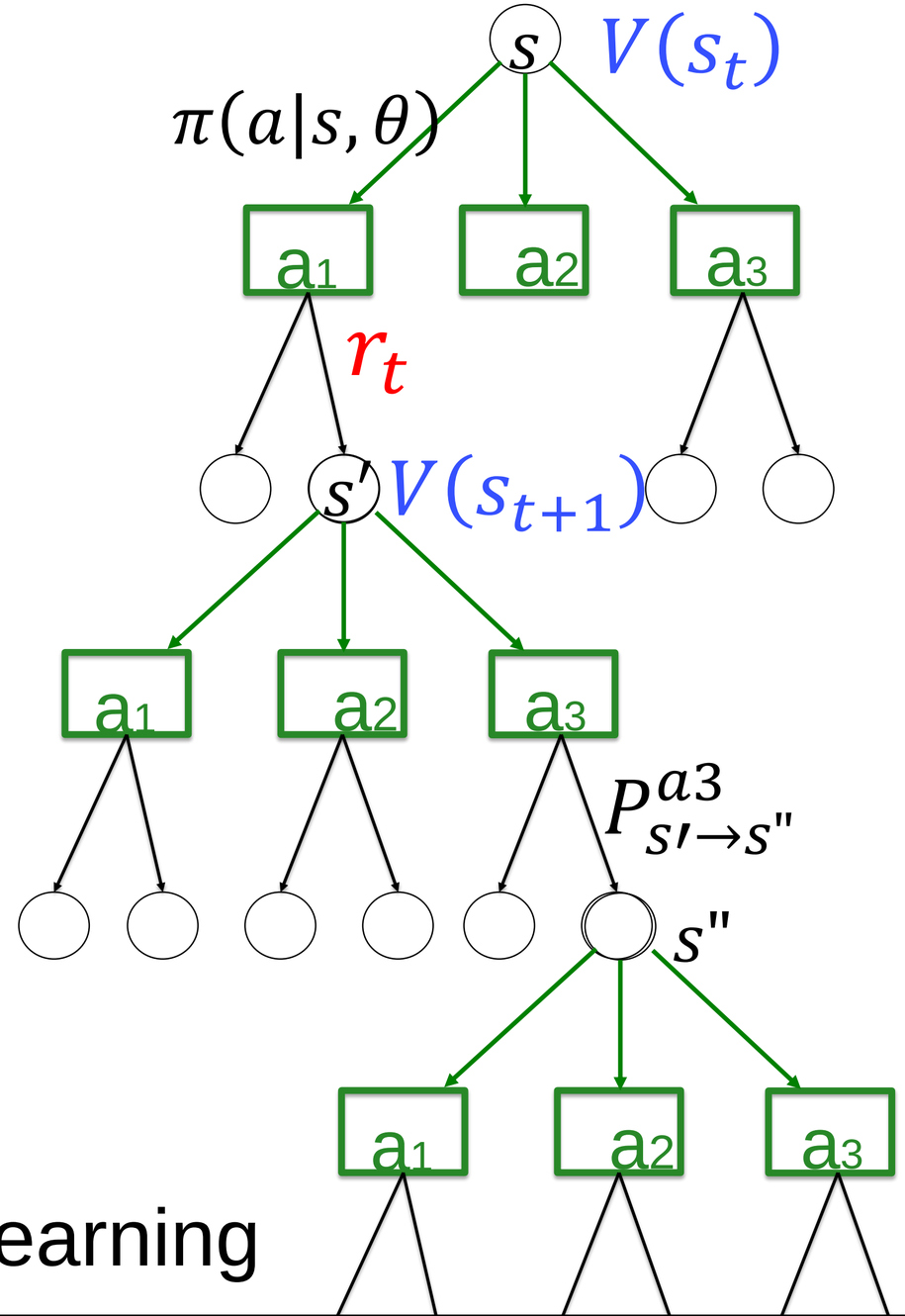
Aim of actor:
update the parameters θ
of the policy $\pi(a|s,\theta)$



update proportional toTD-error

$$\delta = \eta\,[r_t + \gamma V(s_{t+1}) - V(s_t)]$$

Aim of critic: estimate V using TD learning



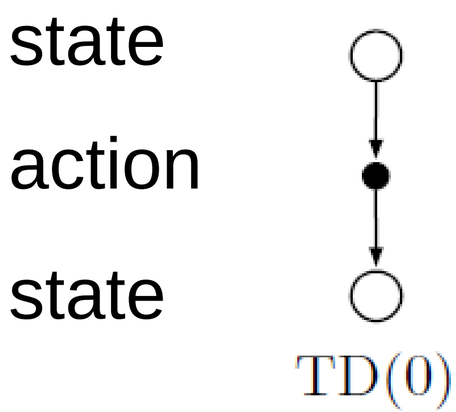
(previous slide)
For a comparison with start with actor-critic:

In both algorithms (actor critic and REINFORCE with baseline), the actor learns actions via policy gradient.

In the actor-critic algorithm the critic learns the V-value via bootstrap TD-learning (see week 9).

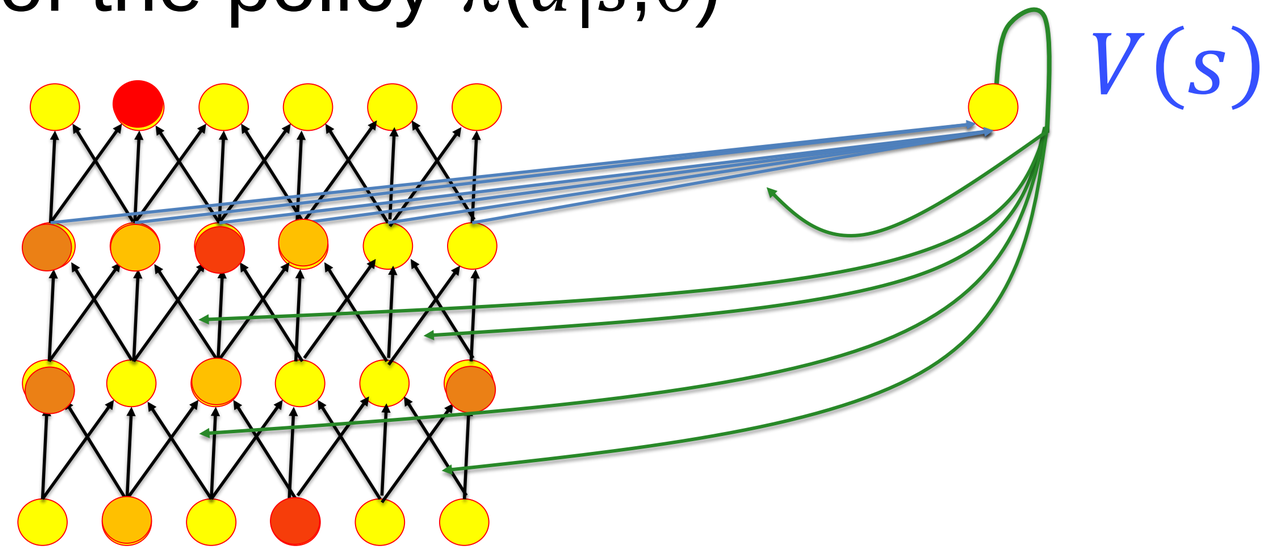
In the actor-critic algorithm the TD error is used as the learning signal for the policy gradient.

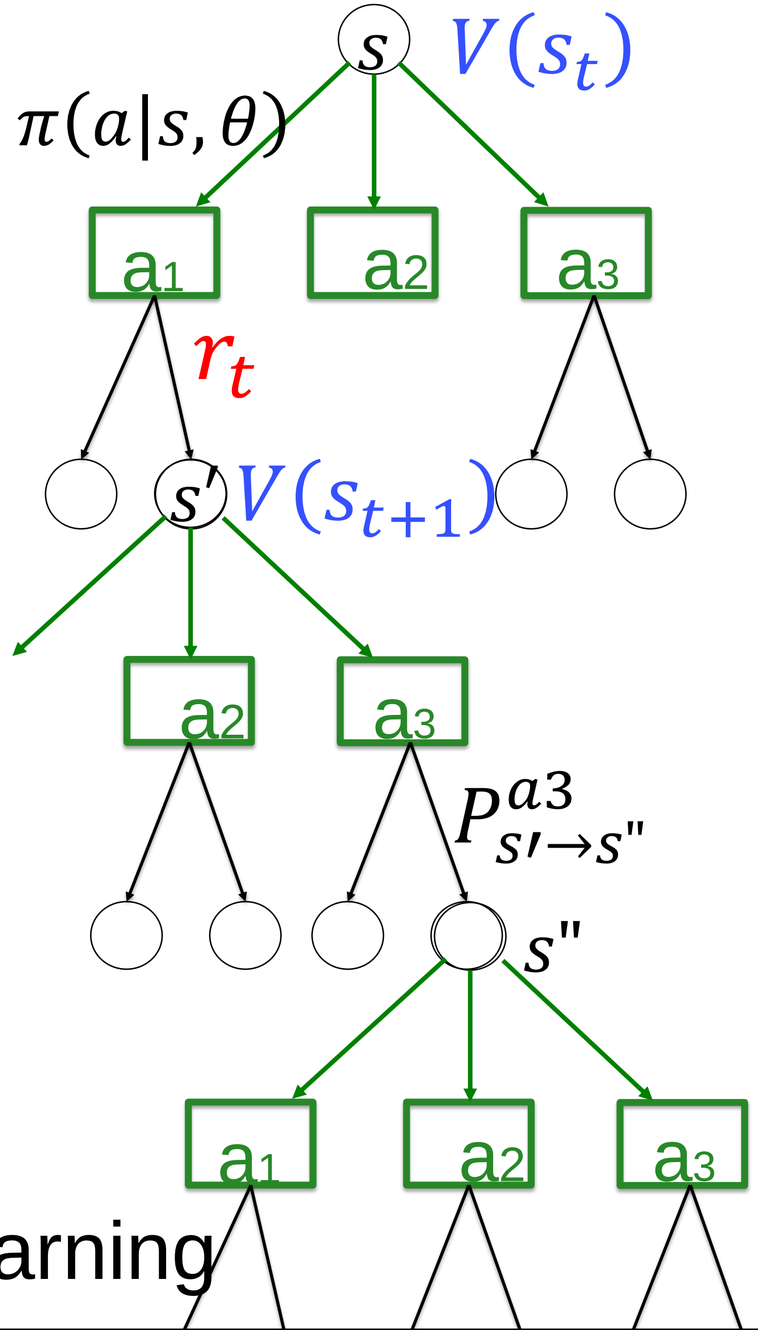
The backup diagram is short:



3. REINFORCE with baseline (vs actor-critic)

1. Aim of actor in REINFORCE
update the parameters θ
of the policy $\pi(a|s,\theta)$





2. update proportional to
RETURN-error: $[R^{a_t}_{s_t \to s_{end}} - V(s_t)]$

3. Aim of critic: estimate V using TD learning

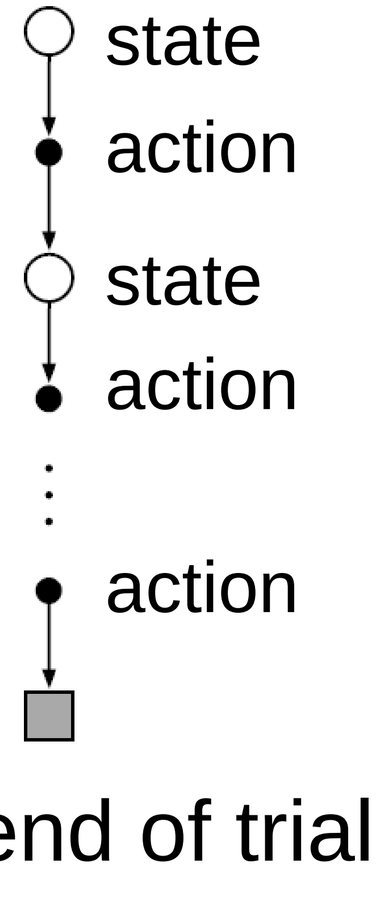
(previous slide)
We continue the comparison with REINFORCE with baseline.

In both algorithms (actor critic and REINFORCE with baseline), the actor learns actions via policy gradient.

In the REINFORCE algorithm the baseline estimator learns the V-value via Monte-Carlo sampling of full episodes (see week 9).

In the REINFOCE algorithm the mismatch between actual return and estimated V-value (‘RETURN error’) is used as the learning signal for the policy gradient.

The Backup diagram is long:



Artificial Neural Networks: Lecture 11

Deep Reinforcement Learning

Wulfram Gerstner

EPFL, Lausanne, Switzerland

1. Review Policy gradient
2. Review Subtracting the mean via the value function
3. Actor-Critic
4. Eligibility traces for policy gradient

(previous slide)

Two weeks ago we discussed eligibility traces.

It turns out that policy gradient algorithm have an intimate link with eligibility traces. In fact, eligibility traces arise naturally for policy gradient algorithms.

In standard policy gradient (e.g., REINFORCE with baseline) we need to run each time until the end of the episode before we have the information necessary to update the weights.

The advantage of eligibility traces is that updates can be done truly online (similar to the actor critic with bootstrapping).

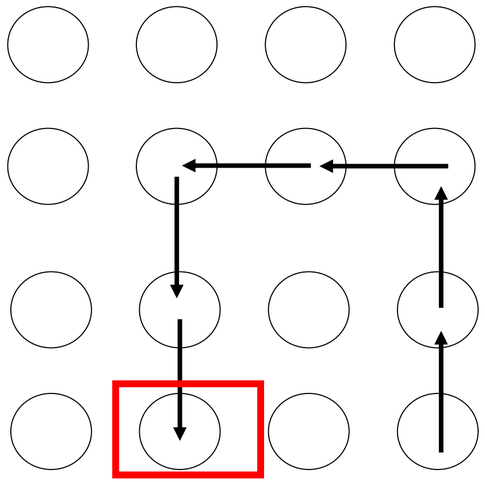
4. Actor-critic with eligibility traces

- Online algorithm
- Actor learns by policy gradient
- Critic learns by TD-learning
- Update eligibility traces while moving
- For each parameter, one eligibility trace
- Update weights proportional to TD-delta

(previous slide)
The idea of policy gradient is combined with the notion of eligibility traces that we had seen two weeks ago.

The result is an algorithm that is truly online: you do not have to wait until the end of an episode to start with the updates.

4. Review: Eligibility Traces

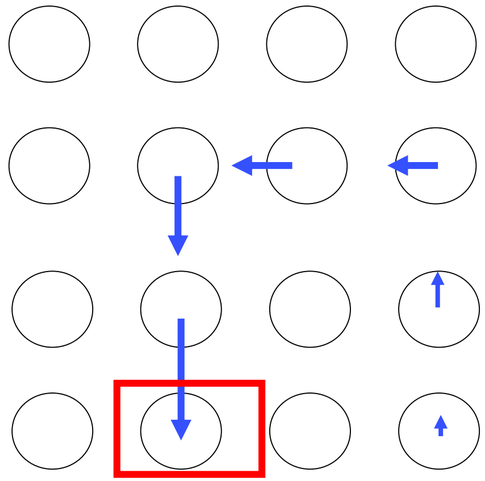


Idea:

- keep memory of previous state-action pairs
- memory decays over time
- Update an eligibility trace for state-action pair

$$e(s,a) \leftarrow \lambda\, e(s,a)$$ decay of **all** traces

$$e(s,a) \leftarrow e(s,a) + 1$$ if action a chosen in state s



- update **all** Q-values:

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

TD-delta

Here: SARSA with eligibility trace

(previous slide)
This the SARSA algorithm with eligibility traces that we had seen two weeks ago. We had derived this algo for a tabular Q-learning model as well as for a network with basis functions and linear read-out units for the Q-values Q(s,a).

In the latter case it was not the Q value itself that had an eligibility trace, but the weights (parameters) that contributed to that Q-value.

We now use the same idea.

4. Eligibility Traces

Idea:

- keep memory of previous ‘candidate updates’
- memory decays over time
- Update an **eligibility trace for each parameter**

$$z_k \leftarrow z_k \ \lambda$$ decay of **all** traces

$$z_k \leftarrow z_k + \frac{d}{dw_k}\ln[\pi(a|s, w_k)]$$ increase of **all** traces

- update **all** parameters of ‘actor’ network:

$$\Delta w_k = \eta \ [r - (V(s_t) - \gamma V(s_{t+1}))] \ z_k$$

TD-delta

Here: policy gradient with eligibility trace

(previous slide)
Eligibility traces can be generalized to deep networks.
Here we focus on the actor network.
For each parameter w_k of the network we have a shadow parameter z_k : the eligibility trace.
Eligibility traces decay at each time step ($\lambda < 1$) and are updated proportional to the derivative of the log-policy. Interpretation:
The update of the eligibility trace can be seen as a 'candidate parameter update' – but it is not yet the 'real' update of the actual parameters.

The update of the actual parameters w_k of the actor network are proportional to the eligibility trace z_k and the TD-error

$$\delta = [r_t + \gamma V(s_{t+1}) - V(s_t)]$$

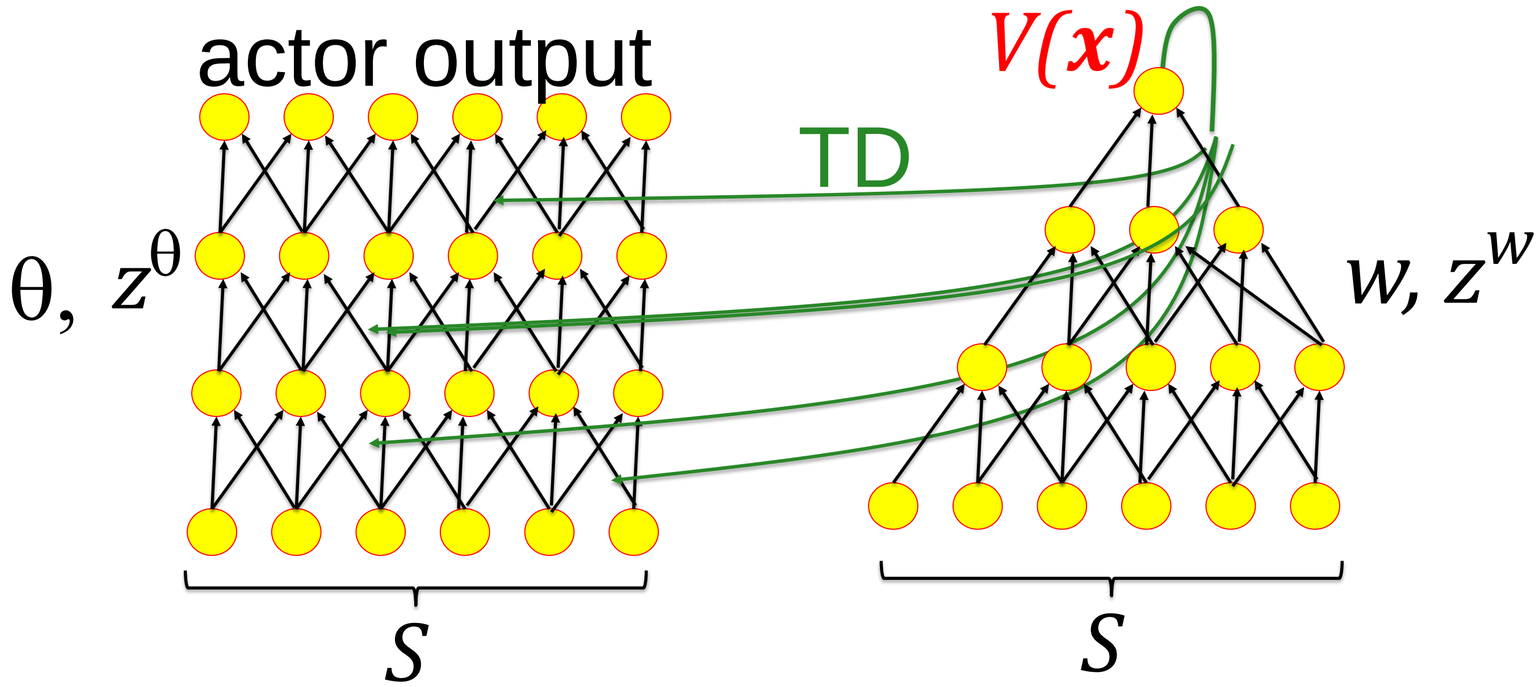
$$= [r_t - [V(s_t) - \gamma V(s_{t+1})]]$$

Parameters are updated at each time step of the episode (as opposed to Monte-Carlo where one has to wait for the end of the episode). Hence 'true online'.

(previous slide) NETWORK for **Algorithm in Pseudo-code by Sutton and Barto.**

The actor network has parameters θ
Eligibility traces of actor have parameters z.
The critic network has parameters *w.*
Eligibility traces of critic have parameters z.

Actor chooses actions with policy π



(previous slide) **Algorithm in Pseudo-code by Sutton and Barto.**

The actor network has parameters θ
While the critic network has parameters w.

The actor network is learned by policy gradient with eligibility traces.
The critic network by TD learning with eligibility traces.

Candidate updates are implemented as eligibility traces z.

4. Actor-Critic with Eligibility traces bootstrapping

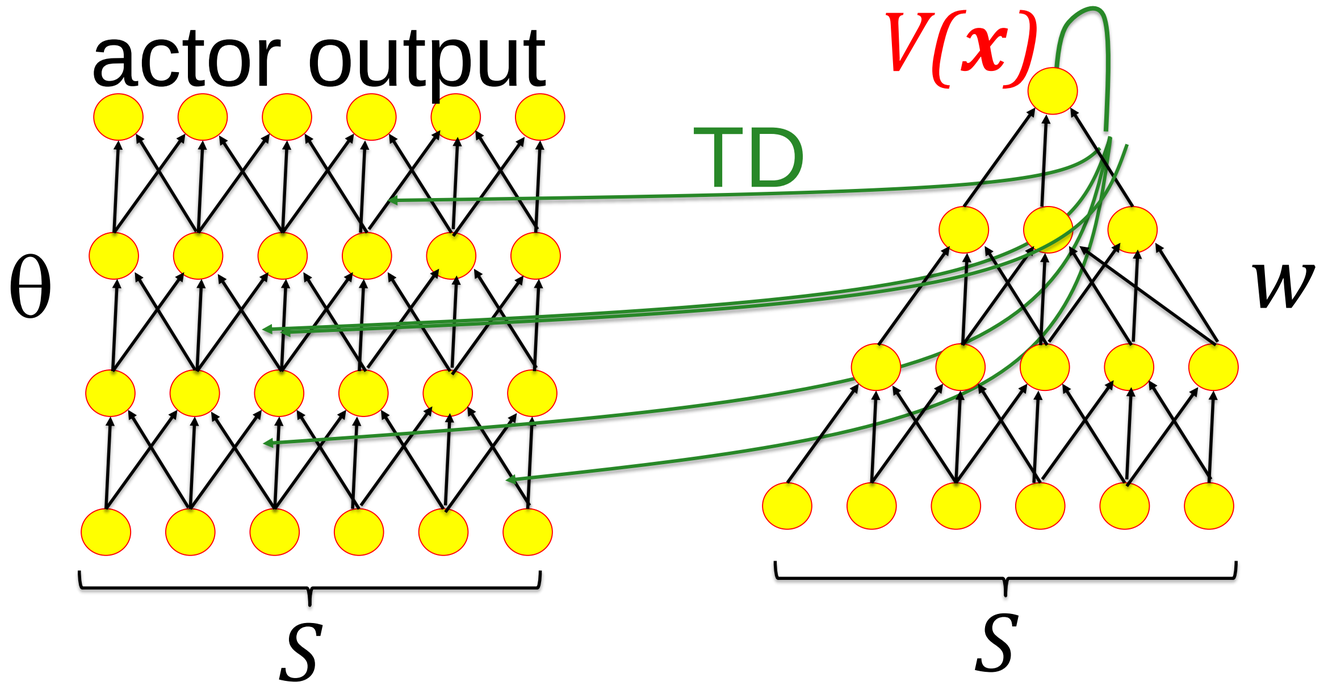
Actor–Critic with Eligibility Traces (episodic), for estimating $\pi_{\boldsymbol{\theta}} \approx \pi_*$

Input: a differentiable policy parameterization $\pi(a|s,\boldsymbol{\theta})$
Input: a differentiable state-value function parameterization $\hat{v}(s,\mathbf{w})$
Parameters: trace-decay rates $\lambda^{\boldsymbol{\theta}} \in [0,1]$, $\lambda^{\mathbf{w}} \in [0,1]$; step sizes $\alpha^{\boldsymbol{\theta}} > 0$, $\alpha^{\mathbf{w}} > 0$
Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ and state-value weights $\mathbf{w} \in \mathbb{R}^{d}$ (e.g., to $\mathbf{0}$)
Loop forever (for each episode):
    Initialize S (first state of episode)
    $\mathbf{z}^{\boldsymbol{\theta}} \leftarrow \mathbf{0}$ (d'-component eligibility trace vector)
    $\mathbf{z}^{\mathbf{w}} \leftarrow \mathbf{0}$ (d-component eligibility trace vector)
    $I \leftarrow 1$
    Loop while S is not terminal (for each time step):
        $A \sim \pi(\cdot|S,\boldsymbol{\theta})$
        Take action A, observe S', R
        $\delta \leftarrow R + \gamma\hat{v}(S',\mathbf{w}) - \hat{v}(S,\mathbf{w})$ (if S' is terminal, then $\hat{v}(S',\mathbf{w}) \doteq 0$)
        $\mathbf{z}^{\mathbf{w}} \leftarrow \gamma\lambda^{\mathbf{w}}\mathbf{z}^{\mathbf{w}} + I\nabla\hat{v}(S,\mathbf{w})$
        $\mathbf{z}^{\boldsymbol{\theta}} \leftarrow \gamma\lambda^{\boldsymbol{\theta}}\mathbf{z}^{\boldsymbol{\theta}} + I\nabla \ln\pi(A|S,\boldsymbol{\theta})$
        $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}}\delta\mathbf{z}^{\mathbf{w}}$
        $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}}\delta\mathbf{z}^{\boldsymbol{\theta}}$
        $I \leftarrow \gamma I$
        $S \leftarrow S'$

(previous slide) **Algorithm in Pseudo-code by Sutton and Barto.**

The actor network has parameters θ
While the critic network has parameters w

The actor network is learned by policy gradient with eligibility traces.
The critic network by TD learning with eligibility traces.
Note that Sutton and Barto include a discount factor γ but in the exercises we will see that the discount factor can (to an excellent approximation) be absorbed into λ



4. Quiz: Policy Gradient and Reinforcement learning

Your friend has followed over the weekend a tutorial in reinforcement learning and claims the following. Is he right?

[ ] Even some policy gradient algorithms use V-values

[ ] V-values for policy gradient are calculated in a separate network (but some parameters can be shared with the actor network)

[ ] The actor-critic network has basically the same architecture as REINFORCE with baseline

[ ] While actor-critic uses ideas from TD learning, REINFORCE with baseline uses Markov estimates of V-values

[ ] Eligibility traces are ‘shadow’ variables for each parameter

[ ] Eligibility traces appear naturally in policy gradient algos.

(your comments)

The proof of the last item is what we will sketch now – proof in the exercises.

4. Review from week 10 .Policy Gradient over multiple time steps

Calculation yields several terms of the form

Total accumulated discounted reward collected in one episode starting at s_t, a_t

$$\Delta\theta_j \propto [R^{a_t}_{s_t \to s_{end}}] \frac{d}{d\theta_j} \ln[\pi(a_t|s_t,\theta)]$$

$$+\gamma[R^{a_{t+1}}_{s_{t+1} \to s_{end}}] \frac{d}{d\theta_j} \ln[\pi(a_{t+1}|s_{t+1},\theta)]$$

+ …

Previous slide.
This is a repetition of an earlier slide.

4. Policy Gradient over multiple time steps (Exercis2)

Step 1: Rewrite $R^{a_t}_{s_t \to s_{end}} = r_{t+n} + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 r_{t+3}$

Step 2: Use same update formula, but for state s_{t+1}

Step 3: Reorder terms according to r_{t+n}

$$\Delta\theta_j \propto [R^{a_t}_{s_t \to s_{end}}] \frac{d}{d\theta_j} \ln[\pi(a_t|s_t, \theta)]$$

$$+\gamma [R^{a_{t+1}}_{s_{t+1} \to s_{end}}] \frac{d}{d\theta_j} \ln[\pi(a_{t+1}|s_{t+1}, \theta)]$$

$$+ \ldots$$

Previous slide.
This is a repetition of the ideas for the exercise.

4. Policy Gradient for eligibility traces (Exercise)

Step 4: Introduce ‘shadow variables’ for eligibility trace

$$z_k \leftarrow z_k \ \lambda$$ decay of **all** traces

$$z_k \leftarrow z_k + \frac{d}{dw_k}\ln[\pi(a|s, w_k)]$$ increase of **all** traces

Step 5: Rewrite update rule for parameters with eligibility trace

$$\Delta w_k = \eta \ r_t \ z_k$$

Previous slide.
This is a repetition of the exercise

4. Eligibility traces from Policy Gradient (Exercise)

Run trial. At each time step, observe state, action, reward

1) Update eligibility trace

$$z_k \leftarrow z_k \ \lambda$$ decay of **all** traces

$$z_k \leftarrow z_k + \frac{d}{dw_k}\ln[\pi(a|s, w_k)]$$ increase of **all** traces

2) update parameters

$$\Delta w_k = \eta \ r_t \ z_k$$

Previous slide.

And these two updates can now be mapped to the algorithm of Sutton and Barto that we saw a few slides before.

Conclusion: eligibility traces are a compact form for rewriting a policy gradient algorithm.

[There are minor differences at the ‘bounderies’ that is, the beginning and end of each episode – but these do not matter].

Artificial Neural Networks: Lecture 11

Policy Gradient Methods

Wulfram Gerstner

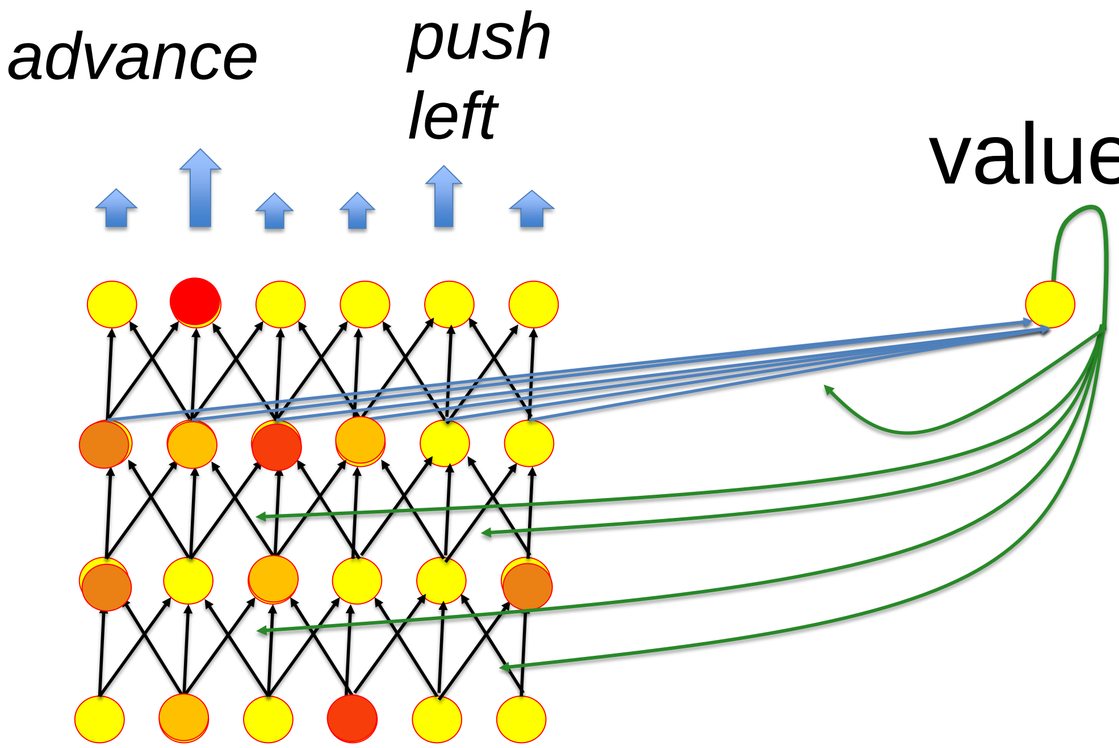
EPFL, Lausanne, Switzerland

1. Review Policy gradient
2. Review Subtracting the mean via the value function
3. Actor-Critic
4. Eligibility traces for policy gradient
5. Actor-Critic in the Brain

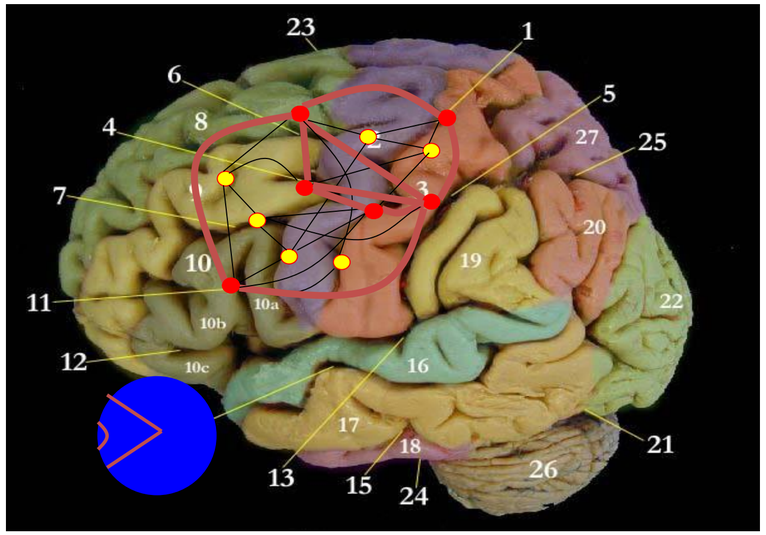
(your comments)
Reinforcement learning algorithms are a class of ‘bio-inspired’ algorithms. But have they something to do with the brain?

5. Review: Actor-Critic = ‘REINFORCE’ with TD signal

- Estimate $V(s)$
- learn via TD error



Review: 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

5. Actor-Critic in the brain?

Questions for this section:

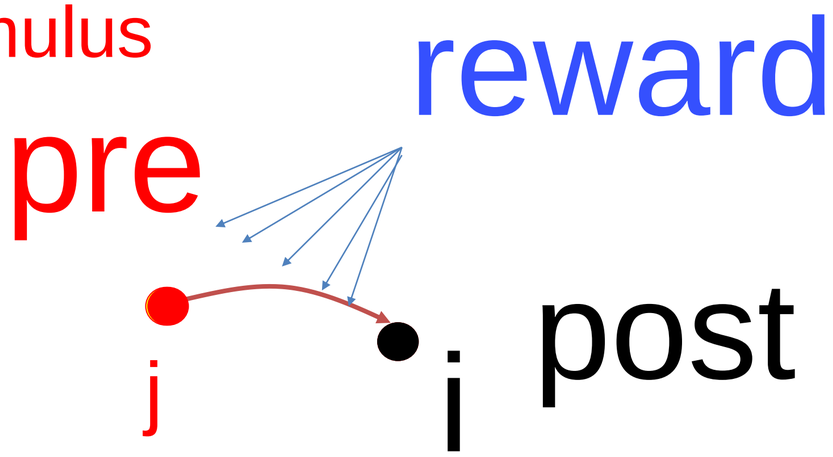
- does the brain implement reinforcement learning algorithms?
- Can the brain implement an actor-critic structure?

(your comments)

5. Review: Policy Gradient: Comparison with Biology

parameter = weight w_j

$$\Delta w_j \propto R(y, \vec{x})[\, y - \langle y \rangle] x_j$$



Weight vector turns in direction of input

Three factors: **reward** **post** **pre**

$$\Delta w_{ij} = \eta \; R(\vec{y}, \vec{x})[\, y_i - \langle y_i \rangle] x_j$$

postsynaptic factor is

'activity – expected activity'

Previous slide.

Reinforcement Learning includes a set of very powerful algorithm – as we have seen in previous lecture.

For today the big question is:
Is the structure of the brain suited to implement reinforcement learning algorithms?

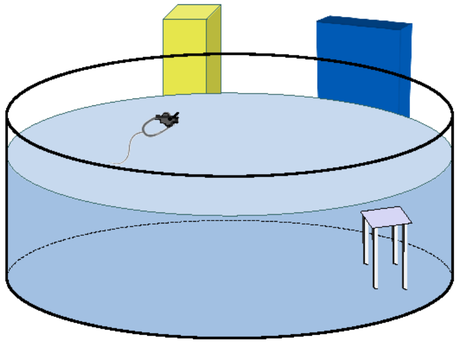
If so which one? Q-learning or SARSA? Policy gradient?

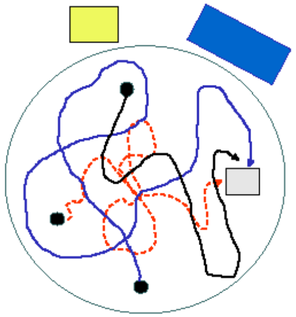
Is the brain architecture compatible with an actor-critic structure?

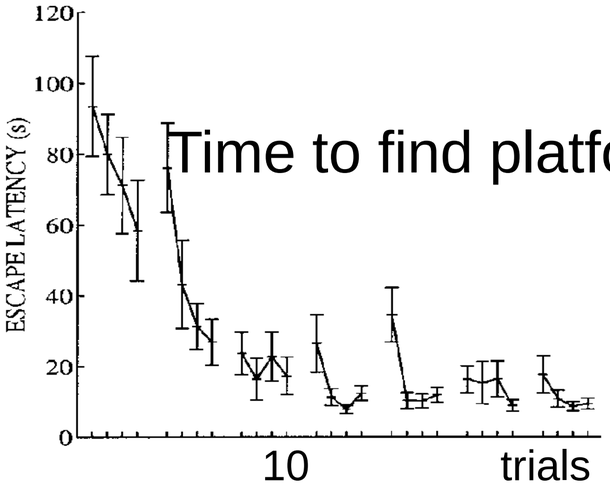
These are the questions we will address in the following.
And to do so, we have to first get a big of background information on brain anatomy.

5. Review from week8: 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)

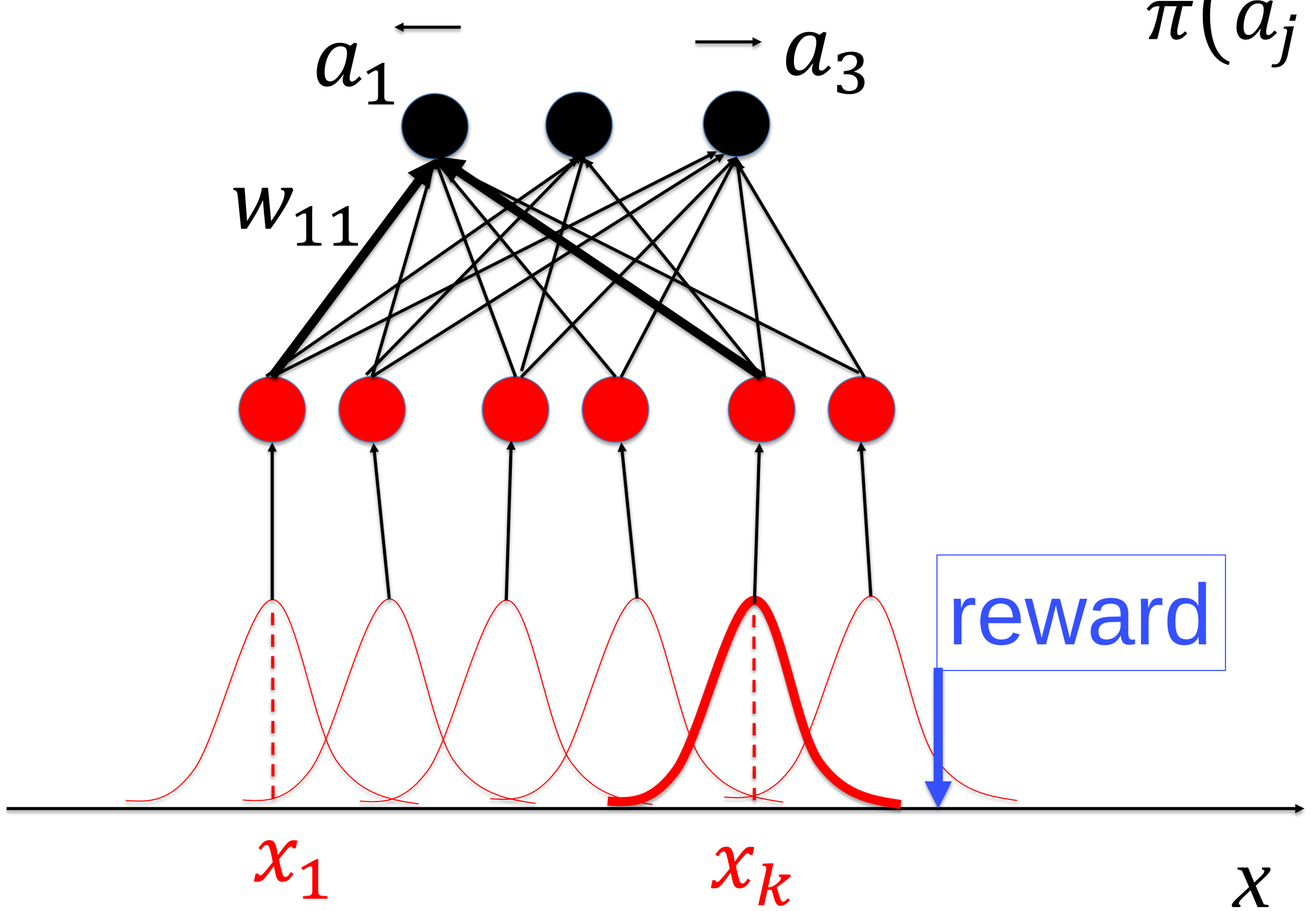
Foster, Morris, Dayan 2000

Previous slide.
Behvioral experiment in the Morris Water Maze.
The water is milky so that the platform is not visible.

After a few trials the rat swims directly to the platform.
Platform is like a reward because the rat gets out of the cold water.

5. Example: Linear activation model with softmax policy

left: $a_1 = 1$ *stay:* $a_2 = 1$ *right:* $a_3 = 1$



parameters

$$\pi(a_j = 1 | x, \theta) = softmax[\sum_k w_{jk}\, y_k]$$

$$y_k = f(x - x_k)$$

f=basis function

Previous slide.

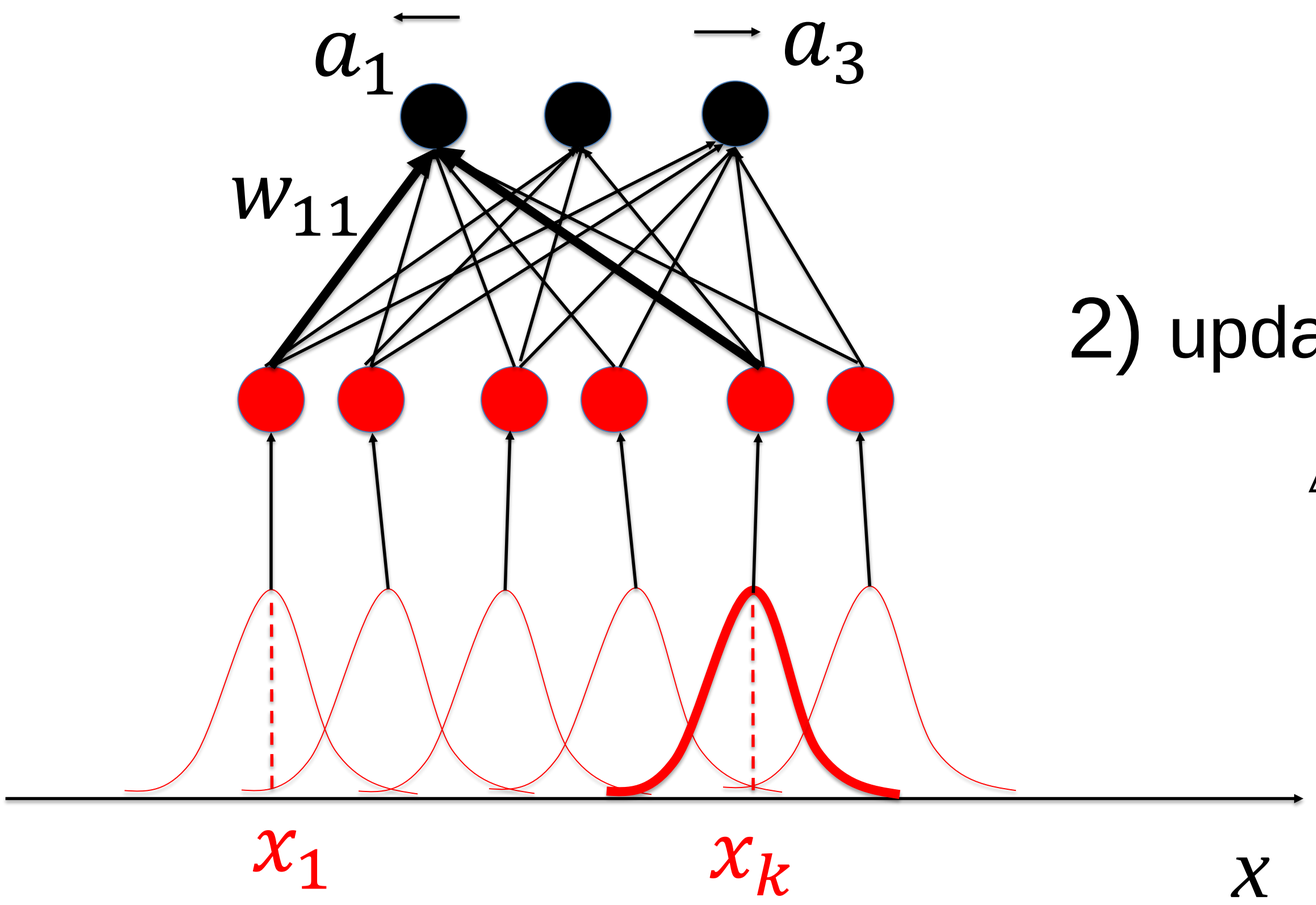
I now want to show that reinforcement learning with policy gradient gives rise to three-factor learning rules.

Suppose the agent moves on a linear track.
There are three possible actions: left, right, or stay.

The policy is given by the softmax function. The total drive of the action neurons is a linear function of the activity y of the hidden neurons which in turn depends on the input x. The activity of hidden neuron k is f(x-x_k). The basis function f could for example be a Gaussian function with center at x_k.

5. Example: Linear activation model with softmax policy

left: *stay:* *right:*



1) Update eligibility trace (for each weight)

$$z_{ik} \leftarrow z_{ik}\,\lambda$$

$$z_{ik} \leftarrow z_{ik} + \frac{d}{dw_k}\ln[\pi(a_i|x)]$$

2) update weights

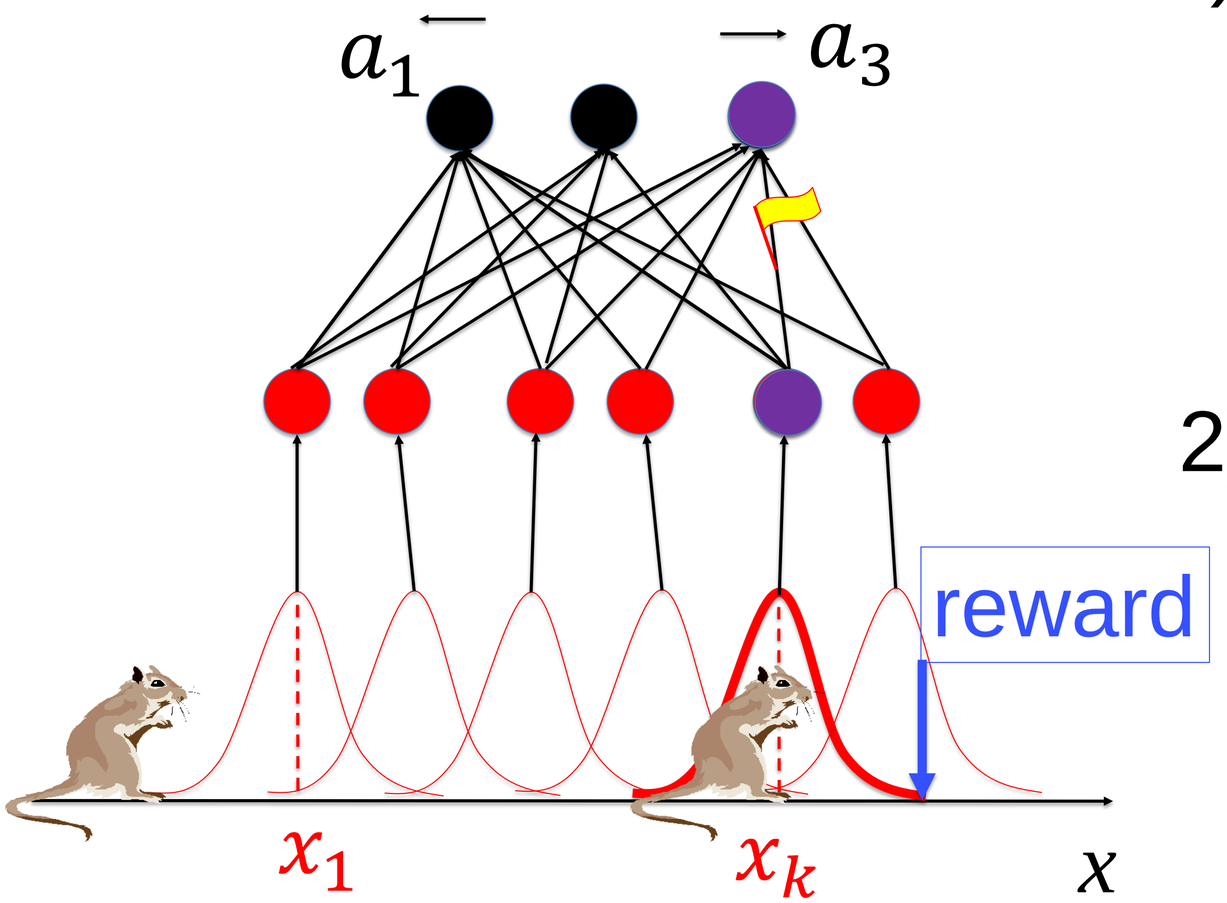
$$\Delta w_{lk} = \eta\; r_t\; z_{lk}$$

Exercise 1 now
8 minutes

Previous slide.
Now we apply the update rule resulting from policy gradient with eligibility traces descent (copy from earlier slide).
This is the in-class exercise (Exercise 1 of this week).

5. Example: Linear activation model with softmax policy

left: $a_1 = 1$ *stay:* $a_2 = 1$ *right:* $a_3 = 1$



0) Choose action $a_i \in \{0,1\}$

1) Update eligibility trace

$$z_{ik} \leftarrow z_{ik}\ \lambda$$

$$z_{ik} \leftarrow z_{ik} + y_k(x)[a_i - \pi(a_i|x)]$$

2) update weights

$$\Delta w_{lk} = \eta\ r_t\ z_{lk}$$

Previous slide.
This is the result of the in-class exercise (Exercise 1 of this week).
Importantly, the update of the eligibility trace is a local learning rule that depends on a presynaptic factor and a postsynaptic factor.

5. Summary: 3-factor rules from Policy gradient

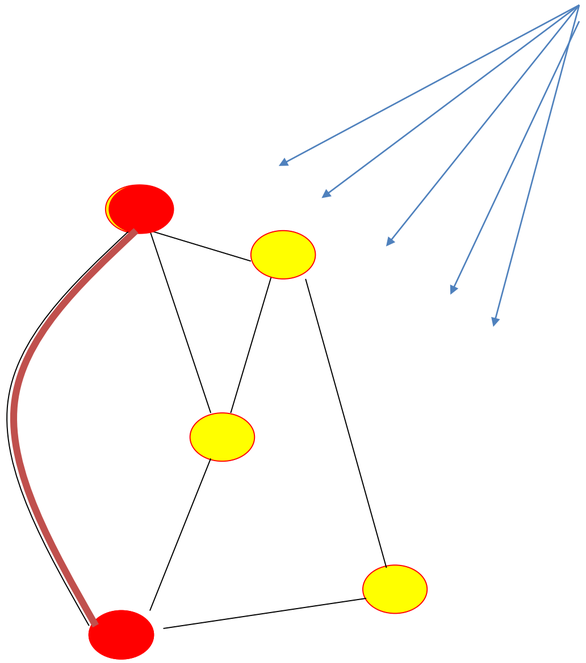
- Policy gradient with one hidden layer and linear softmax readout yields a 3-factor rule
- Eligibility trace is set by joint activity of presynaptic and postsynaptic neuron
- Update happens proportional to reward and eligibility trace
- The presynaptic neuron represents the state
- The postsynaptic neuron the action
- True online rule
  → could be implemented in biology
  → can also be implemented in parallel asynchr. hardware

Previous slide.
Summary: A policy gradient algorithm in a network where the output layer has a linear drive with softmax output leads to a three-factor learning rule for the connections between neurons in the hidden layer and the output.

These three factor learning rules are important because they are completely asynchronous, local, and online and could therefore be implemented in biology or parallel hardware.

3. Recent experiments for Three-factor rules

Neuromodulators for reward, interestingness, surprise; attention; novelty



Step 1: co-activation sets eligibility trace

Step 2: eligibility trace decays over time

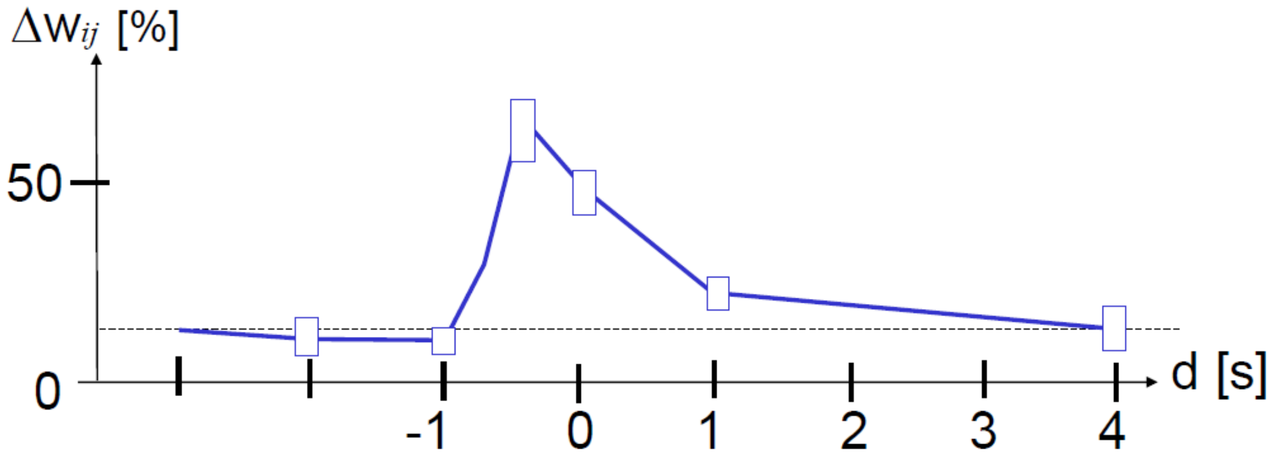
Step 3: delayed neuro-Modulator: eligibility trace translated into weight change

Previous slide.

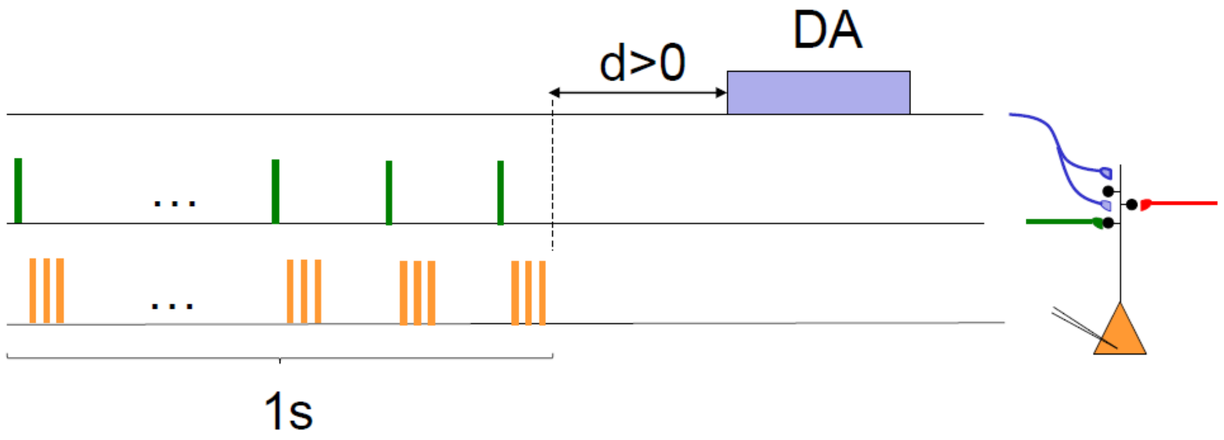
three-factor learning rules are a theoretical concept.

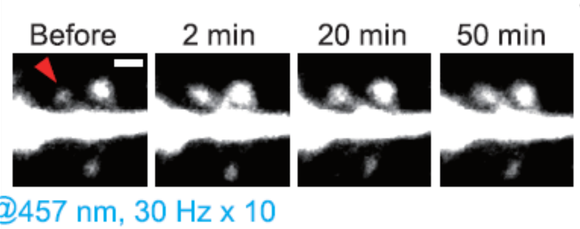
But are there any experiments? Only quite recently, a few experimental results were published that directly address this question.

5. Three-factor rules in striatum: eligibility trace and delayed Da



Yagishita et al. 2014

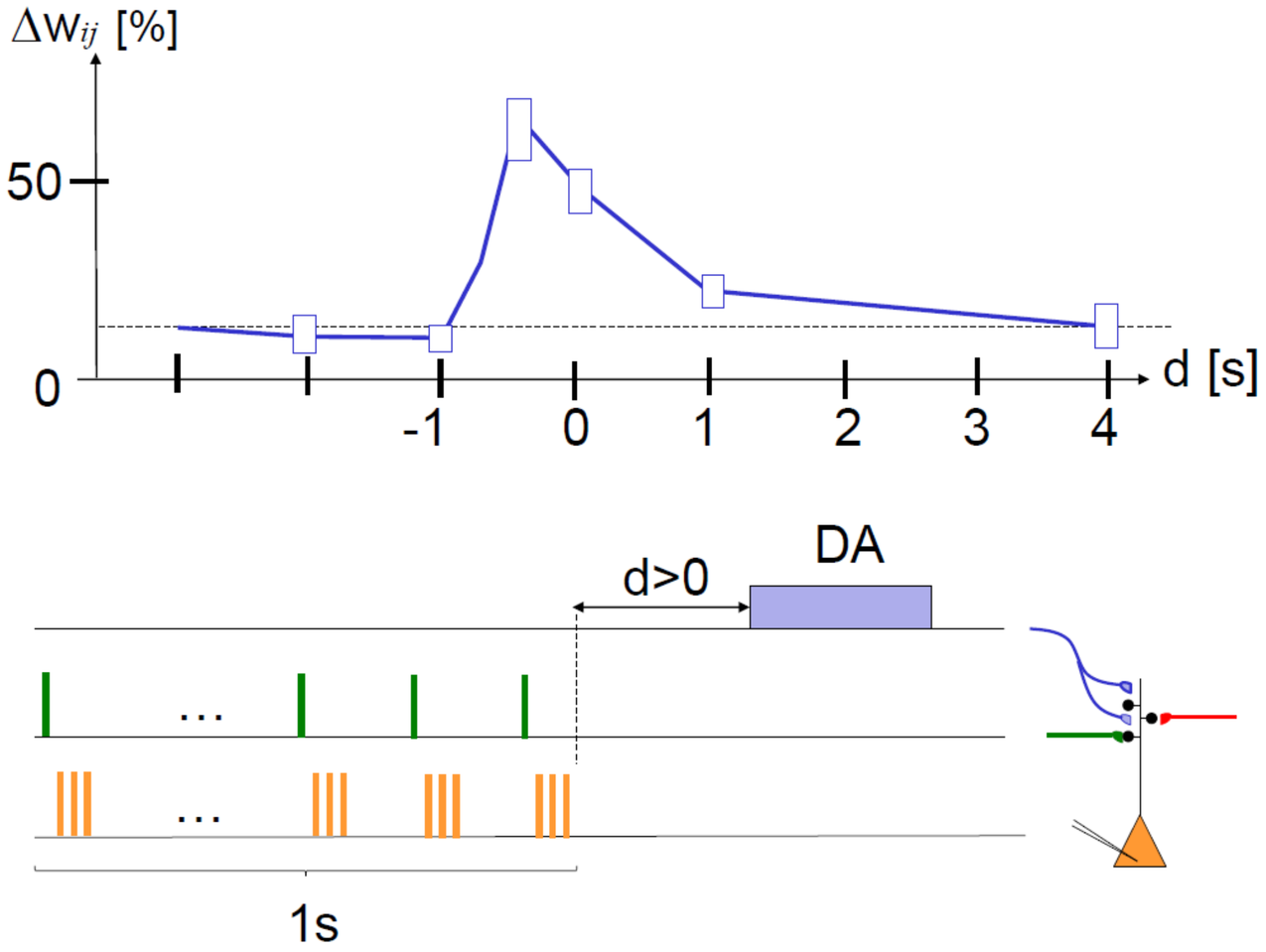




-Dopamine can come with a delay of 0 -1s

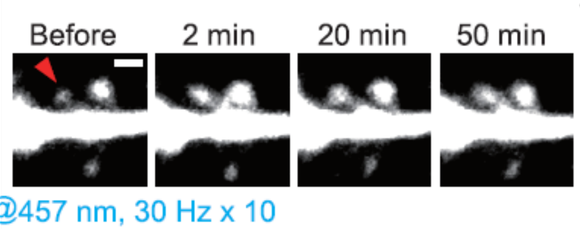
-Long-Term stability over at least 50 min.

5. Three-factor rules in striatum: eligibility trace and delayed Da



Yagishita et al. 2014

In striatum medial spiny cells, stimulation of presynaptic glutamatergic fibers (green) followed by three postsynaptic action potentials (STDP with pre-post-post-post at +10ms) repeated 10 times at 10Hz yields LTP if dopamine (DA) fibers are stimulated during the presentation (d < 0) or shortly afterward (d = 0s or d = 1s) but not if dopamine is given with a delay d = 4s; redrawn after Fig. 1 of (Yagishita et al., 2014), with delay d defined as time since end of STDP protocol



-Dopamine can come with a delay of 0-1s

-Long-Term stability over at least 50 min.

5. Neuromodulators as Third factor

Three factors are needed for synaptic changes:

- Presynaptic factor = activity of presynaptic neuron
- Postsynaptic factor = activity of postsynaptic neuron
- Third factor = Neuromodulator such as dopamine

Presynaptic and postsynaptic factor ‘select’ the synapse.

→ a small subset of synapses becomes ‘eligible’ for change.

The ‘Third factor’ is a nearly global signal

→ broadcast signal, potentially received by all synapses.

Synapses need all three factors for change

Previous slide.
The third factor in a three-factor learning rule should be the global factor signaling success or reward. We said that the third factor could be a neuromodulator such as dopamine.

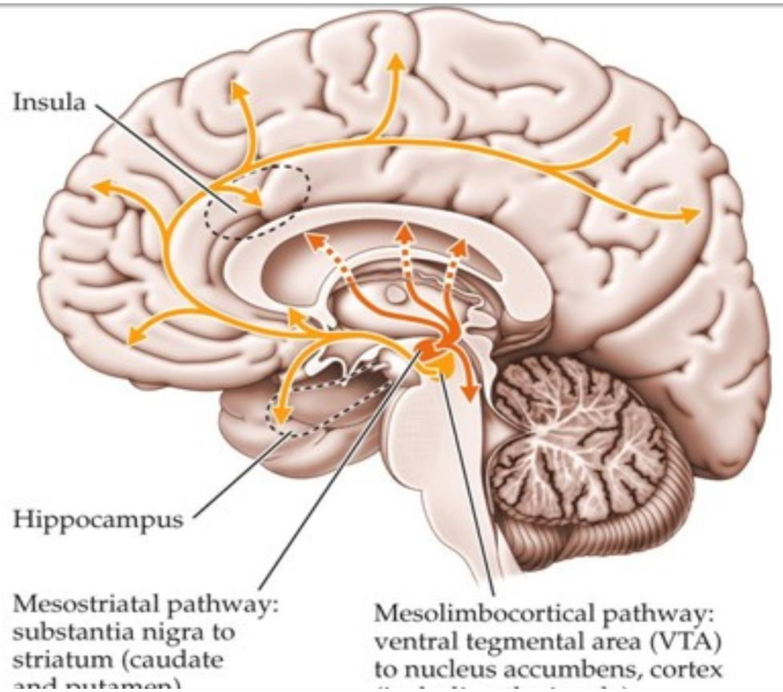
Review from week 8: Reward information

Neuromodulator **dopamine**: - is nearly globally broadcasted
- signals reward minus expected reward

‘success signal’

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

Dopamine



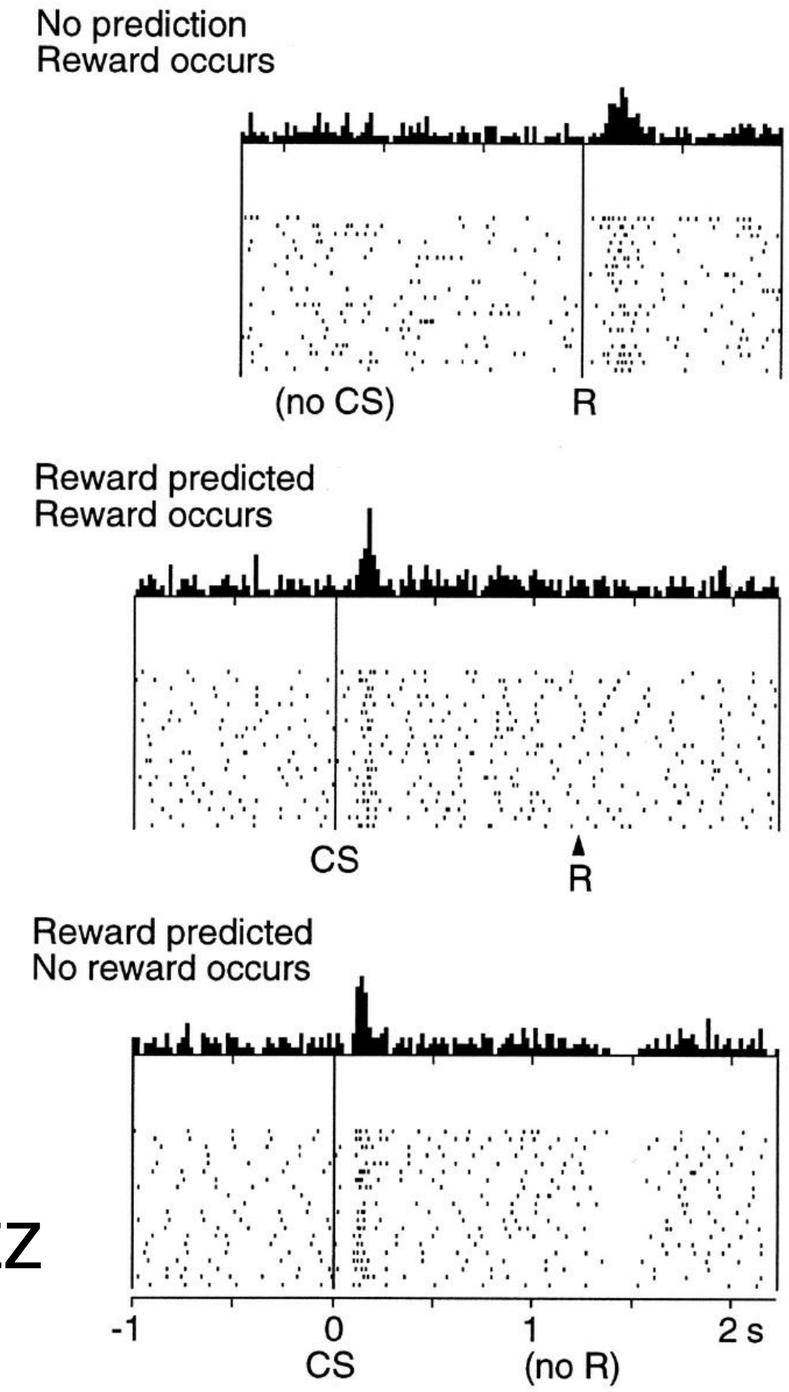
Previous slide. Dopamine neurons send dopamine signals to many neurons and synapses in parallel in a broadcast like fashion.

5. Dopamine as Third factor

Conditioning:

red light →1s→reward

CS:
Conditioning
Stimulus



Sutton book, reprinted from W. Schultz

5. Dopamine as Third factor

This is now the famous experiment of W. Schultz.
In reality the CS was not a red light, but that does not matter

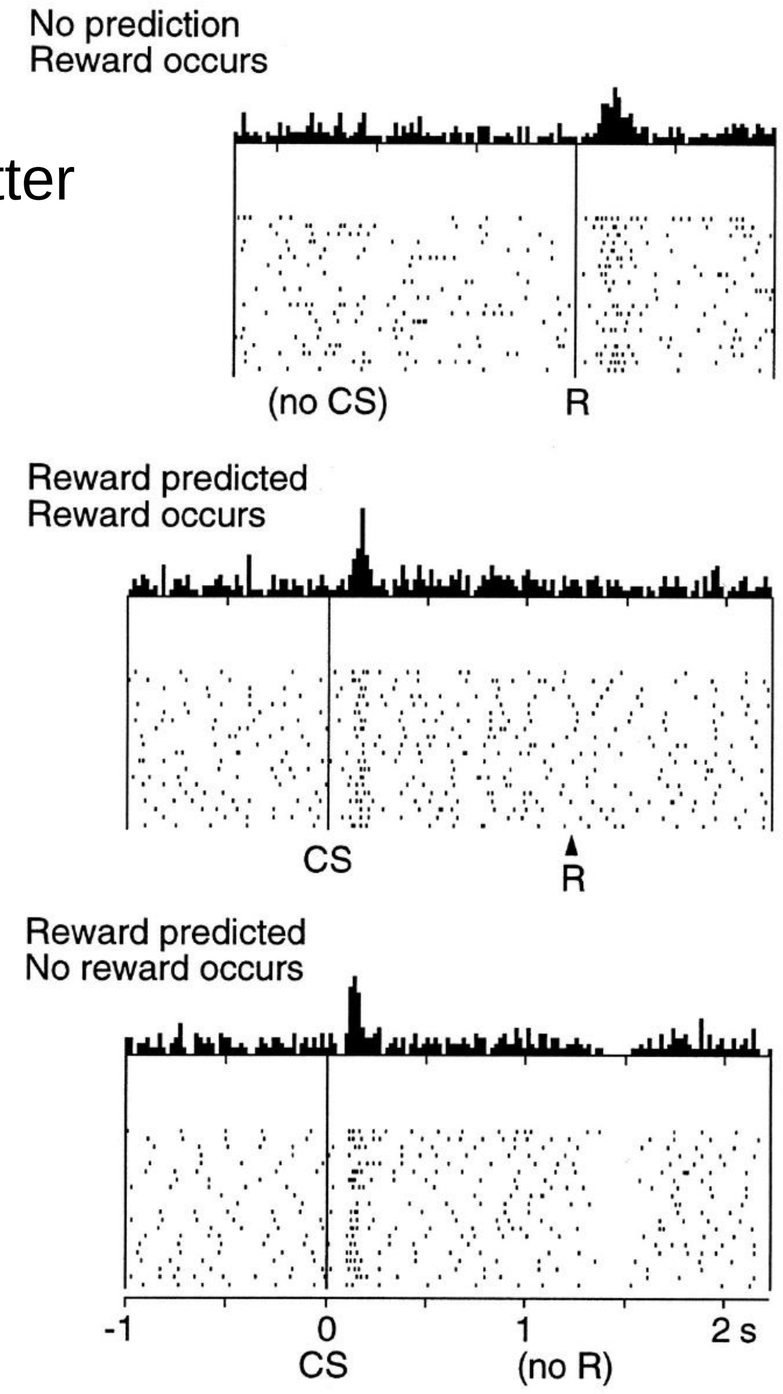


Figure 15.3: The response of dopamine neurons drops below baseline shortly after the time when an expected reward fails to occur. Top: dopamine neurons are activated by the unpredicted delivery of a drop of apple juice. Middle: dopamine neurons respond to a conditioned stimulus (CS) that predicts reward and do not respond to the reward itself. Bottom: when the reward predicted by the CS fails to occur, the activity of dopamine neurons drops below baseline shortly after the time the reward is expected to occur. At the top of each of these panels is shown the average number of action potentials produced by monitored dopamine neurons within small time intervals around the indicated times. The raster plots below show the activity patterns of the individual dopamine neurons that were monitored; each dot represents an action potential. From Schultz, Dayan, and Montague, A Neural Substrate of Prediction and Reward, *Science*, vol. 275, issue 5306, pages 1593-1598, March 14, 1997. Reprinted with permission from AAAS.

5. Summary: Dopamine as Third factor

- Dopamine signals ‘reward minus expected reward’
- Dopamine signals an ‘event that predicts a reward’
- Dopamine signals approximately the TD-error

$$DA(t) = [\underbrace{r(t) - (V(s) - \gamma V(s'))}_{\text{TD-delta}}]$$

Previous slide.

The paper of W. Schultz has related the dopamine signal to some basic aspects of Temporal difference Learning. The Dopamine signal is similar to the TD error.

6. Eligibility Traces with TD in Actor-Critic

Idea:

- keep memory of previous ‘candidate updates’
- memory decays over time
- Update an **eligibility trace for each parameter**

$z_k \leftarrow z_k \ \lambda$ decay of **all** traces

$z_k \leftarrow z_k + \frac{d}{dw_k} \ln[\pi(a|s, w_k)]$ increase of **all** traces

- update **all** parameters:

$$\Delta w_k = \eta \ \underbrace{[r - (V(s) - \gamma V(s'))]}_{\text{TD-delta}} \ z_k$$

→ policy gradient with eligibility trace and TD error

Previous slide.

Review of algorithm with actor-critic architecture and policy gradient with eligibility traces and TD.

6. Summary: Eligibility Traces with TD in Actor-Critic

Three-factor rules:

Presynaptic and postsynaptic factor ‘select’ the synapse.

→ a small subset of synapses becomes ‘eligible’ for change.

The ‘Third factor’ is a nearly global broadcast signal

→ potentially received by all synapses.

Synapses need all three factors for change

The ‘Third factor’ can be the Dopamine-like TD signal

→ Need actor-critic architecture to calculate $\gamma V(s') - V(s)$

→ Dopamine signals $[r_t + \gamma V(s') - V(s)\ ]$

Previous slide.

The three factor rule, dopamine, TD signals, value functions now all fit together.

Artificial Neural Networks: Lecture 11

Policy Gradient Methods

Wulfram Gerstner

EPFL, Lausanne, Switzerland

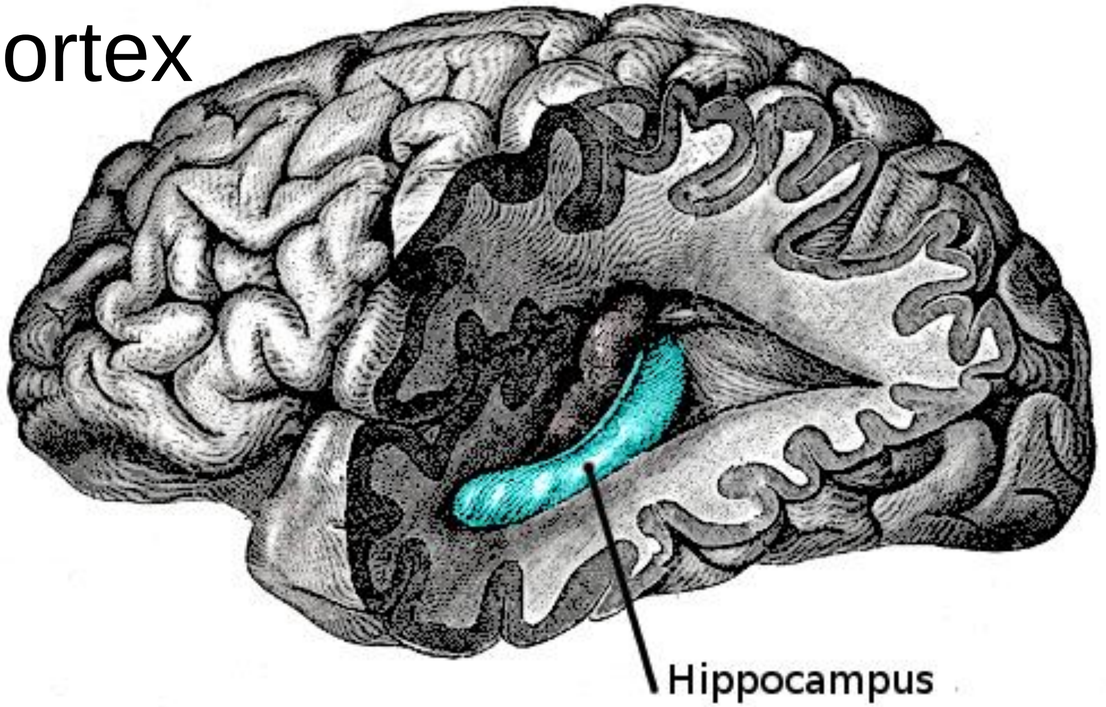
1. Review Policy gradient
2. Review Subtracting the mean via the value function
3. Actor-Critic
4. Eligibility traces for policy gradient
5. Actor-Critic in the Brain
6. Application: Rat navigation

Previous slide.

We said that the three factor rule, dopamine, TD signals, value functions now all fit together. Let’s apply this to the problem of navigation.

7. Coarse Brain Anatomy: hippocampus

Hippocampus
- Sits below/part of temporal cortex
- Involved in memory
- Involved in spatial memory



Spatial memory:
knowing where you are,
knowing how to navigate in an environment

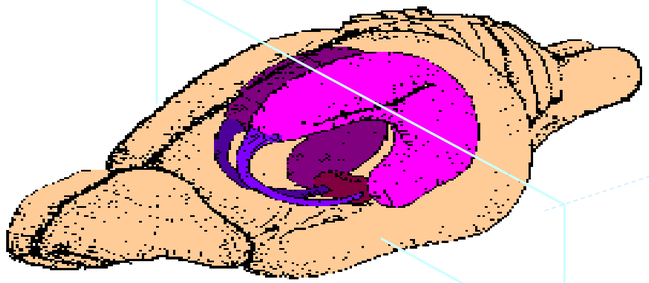
fig: Wikipedia

Henry Gray (1918) *Anatomy of the Human Body*

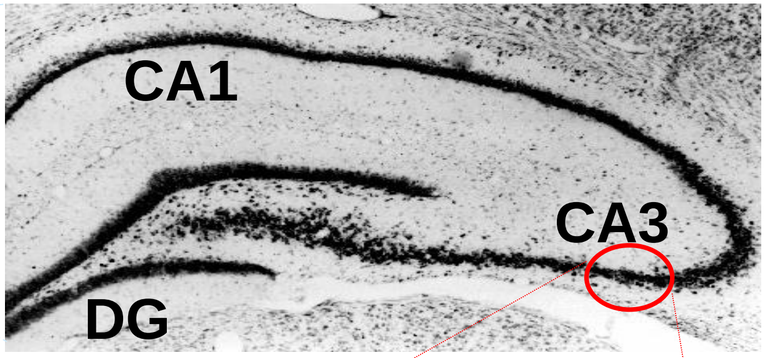
Previous slide.

the problem of navigation needs the spatical representation of the hippocampus.

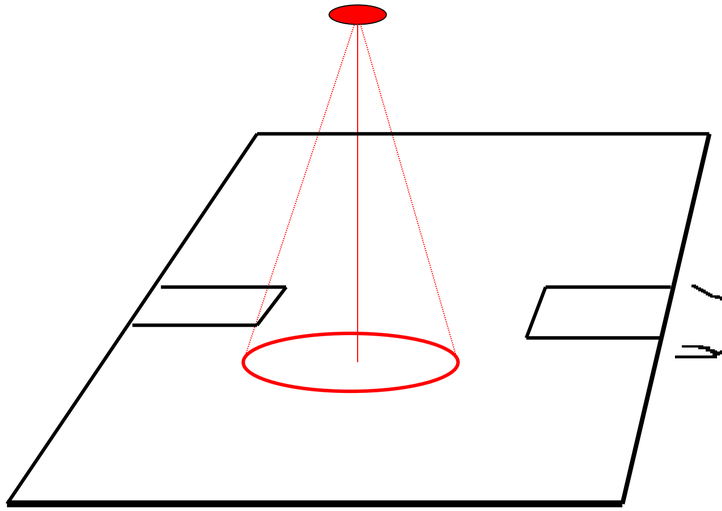
7. Place cells in rat hippocampus

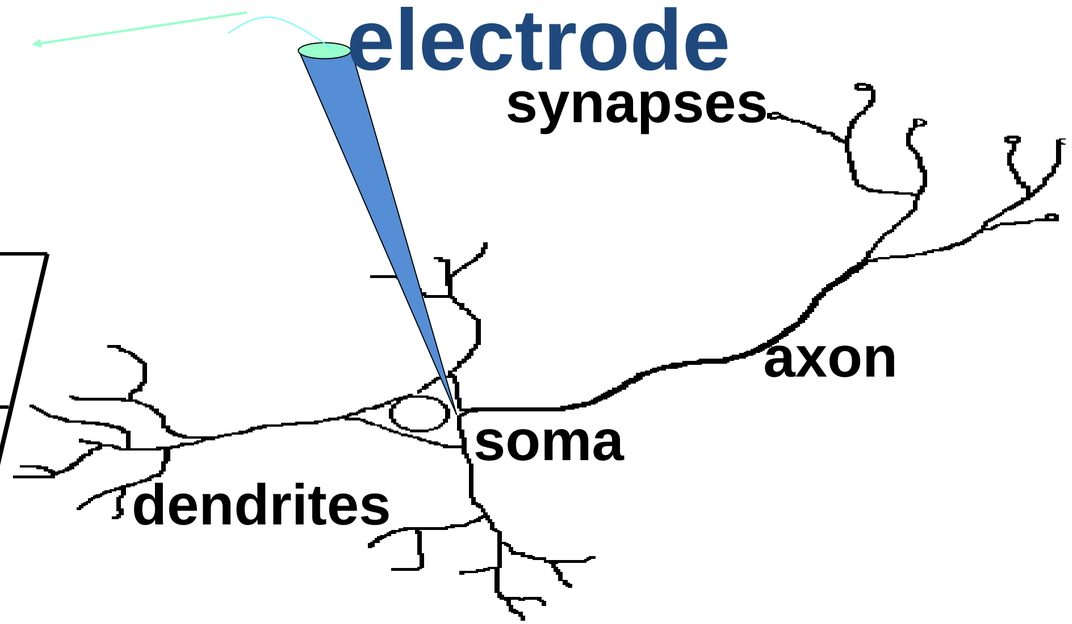


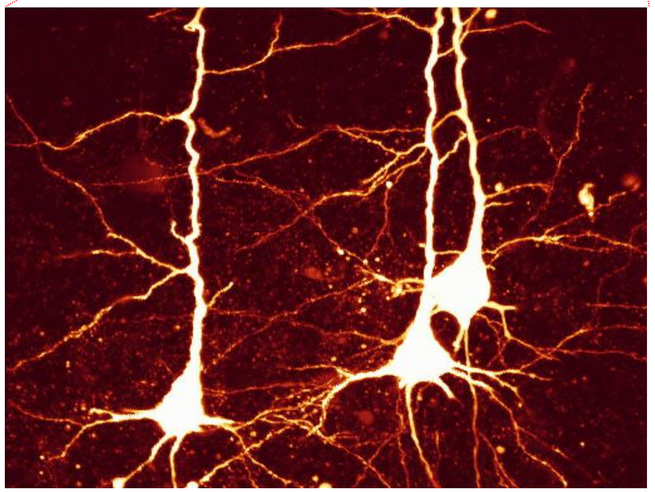
rat brain



Place fields







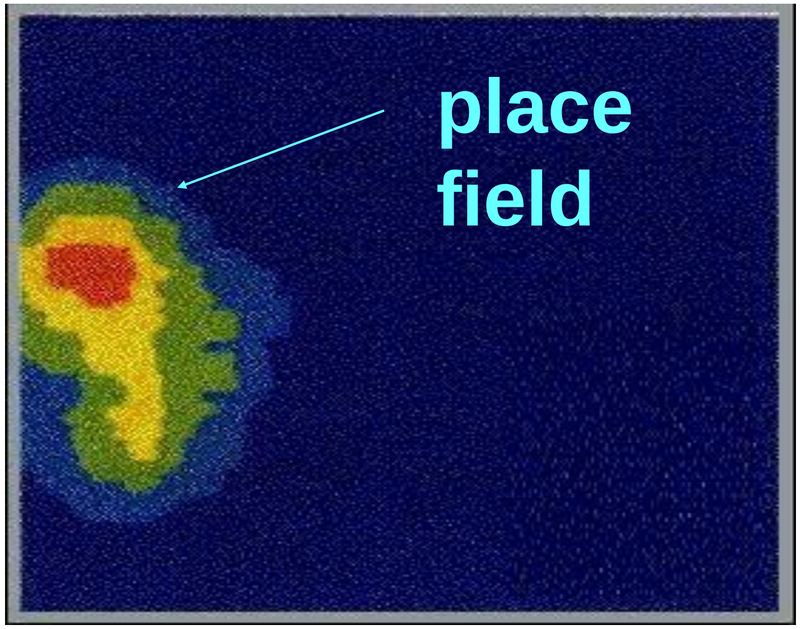
pyramidal cells

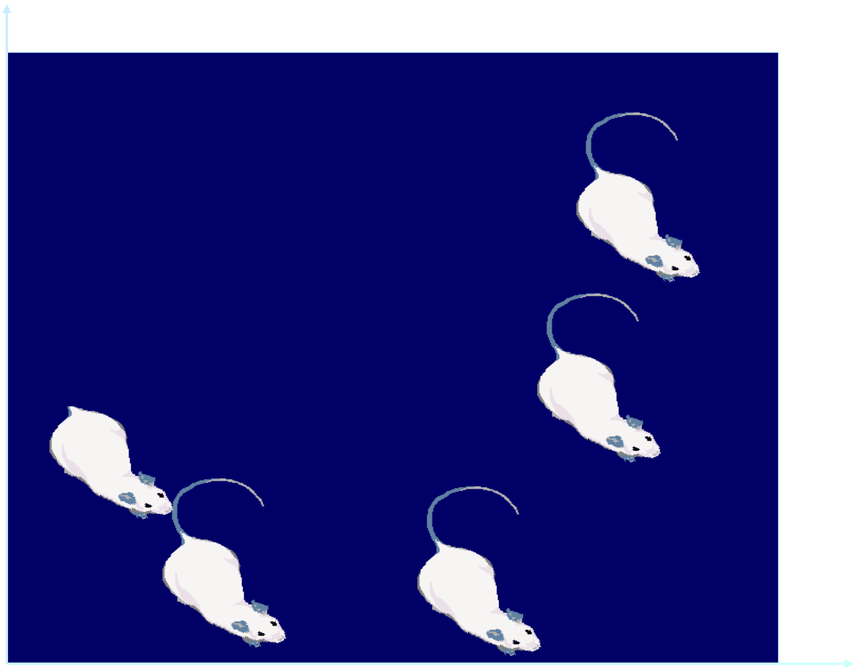
Previous slide.

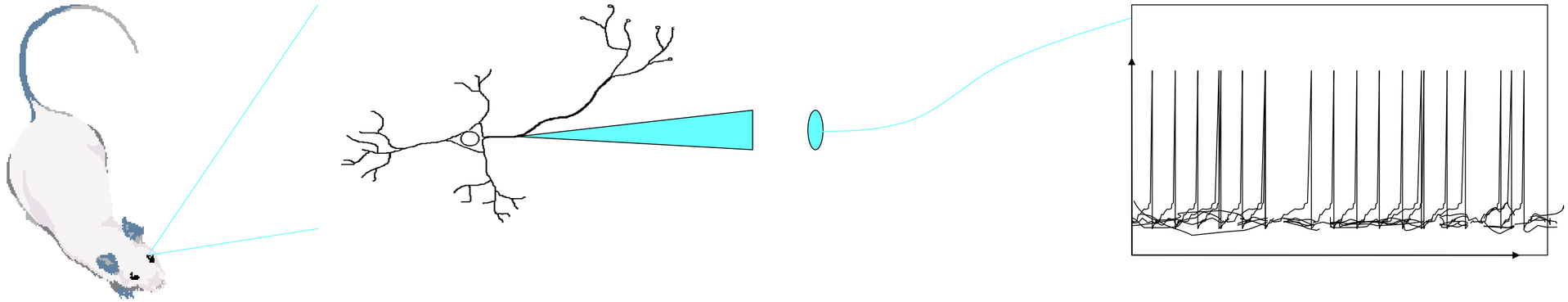
the hippocampus of rodents (rats or mice) looks somewhat different to that of humans. Importantly, cells in hippocampus of rodents respond only in a small region of the environment. For this reason they are called place cells. The small region is called the place field of the cell.

6. Hippocampal place cells

Main property: encoding the animal's location



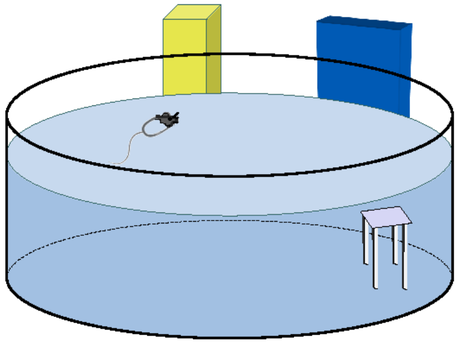


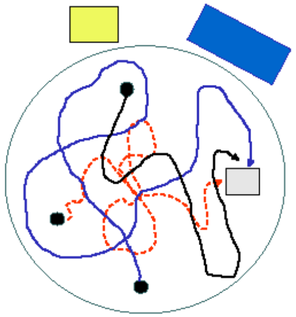


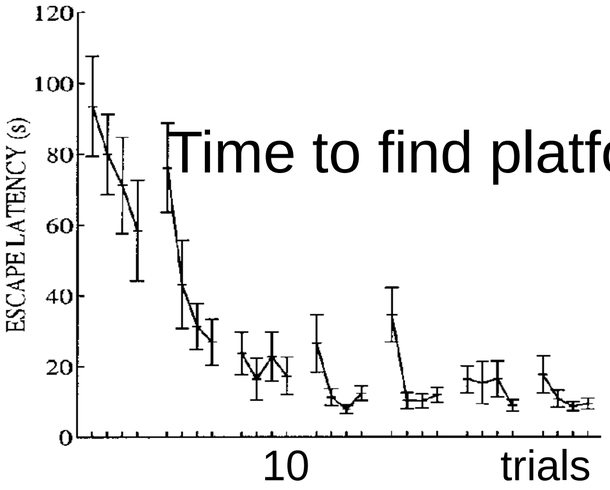
Previous slide.
Left: experimentally measured place field of a single cell in hippocampus.
Right: computer animation of place field

6. Review of Morris Water maze

Morris Water Maze







Time to find platform

Rats learn to find
the hidden platform

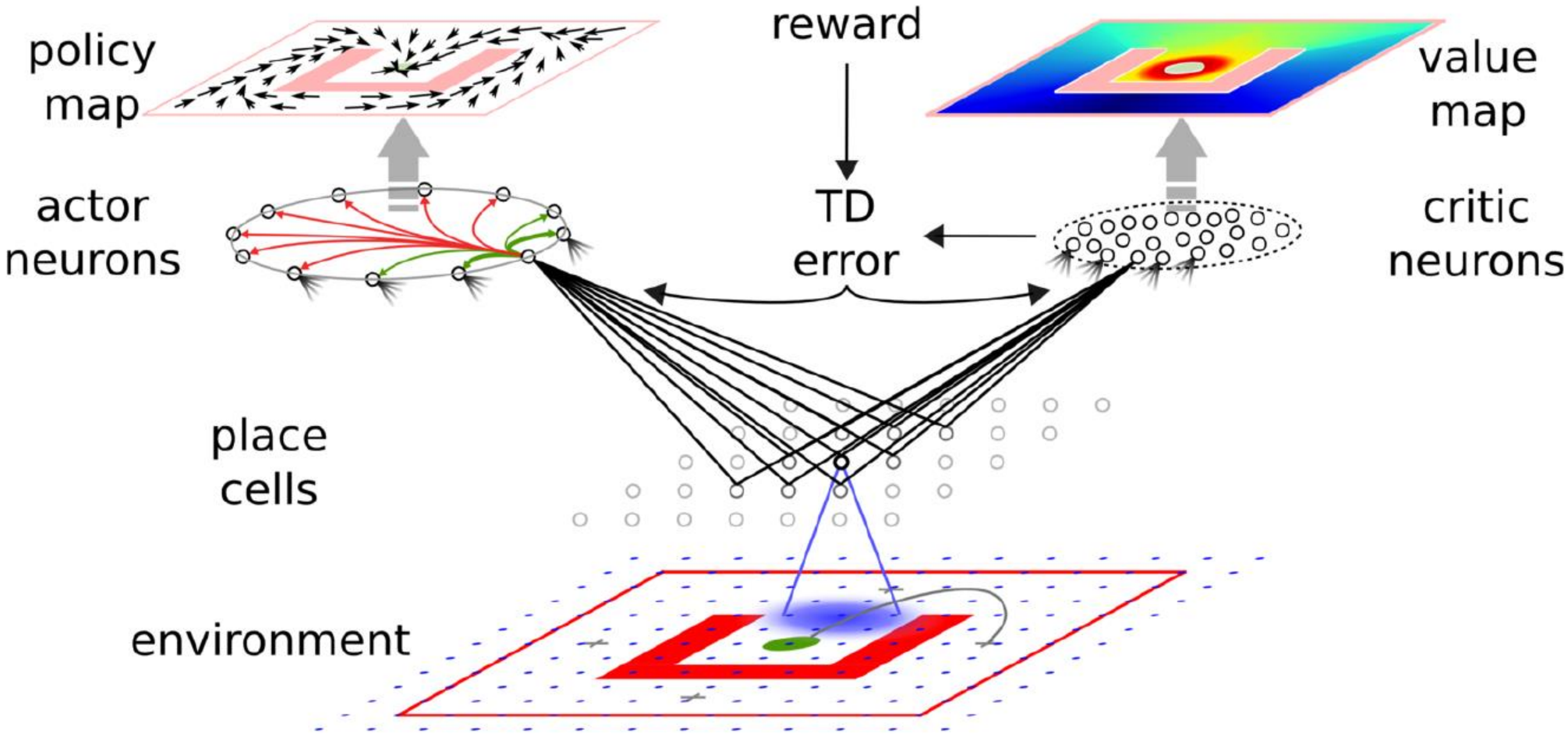
(Because they like to
get out of the cold water)

Foster, Morris, Dayan 2000

Previous slide.
Behvioral experiment in the Morris Water Maze.
The water is milky so that the platform is not visible.

After a few trials the rat swims directly to the platform.
Platform is like a reward because the rat gets out of the cold water.

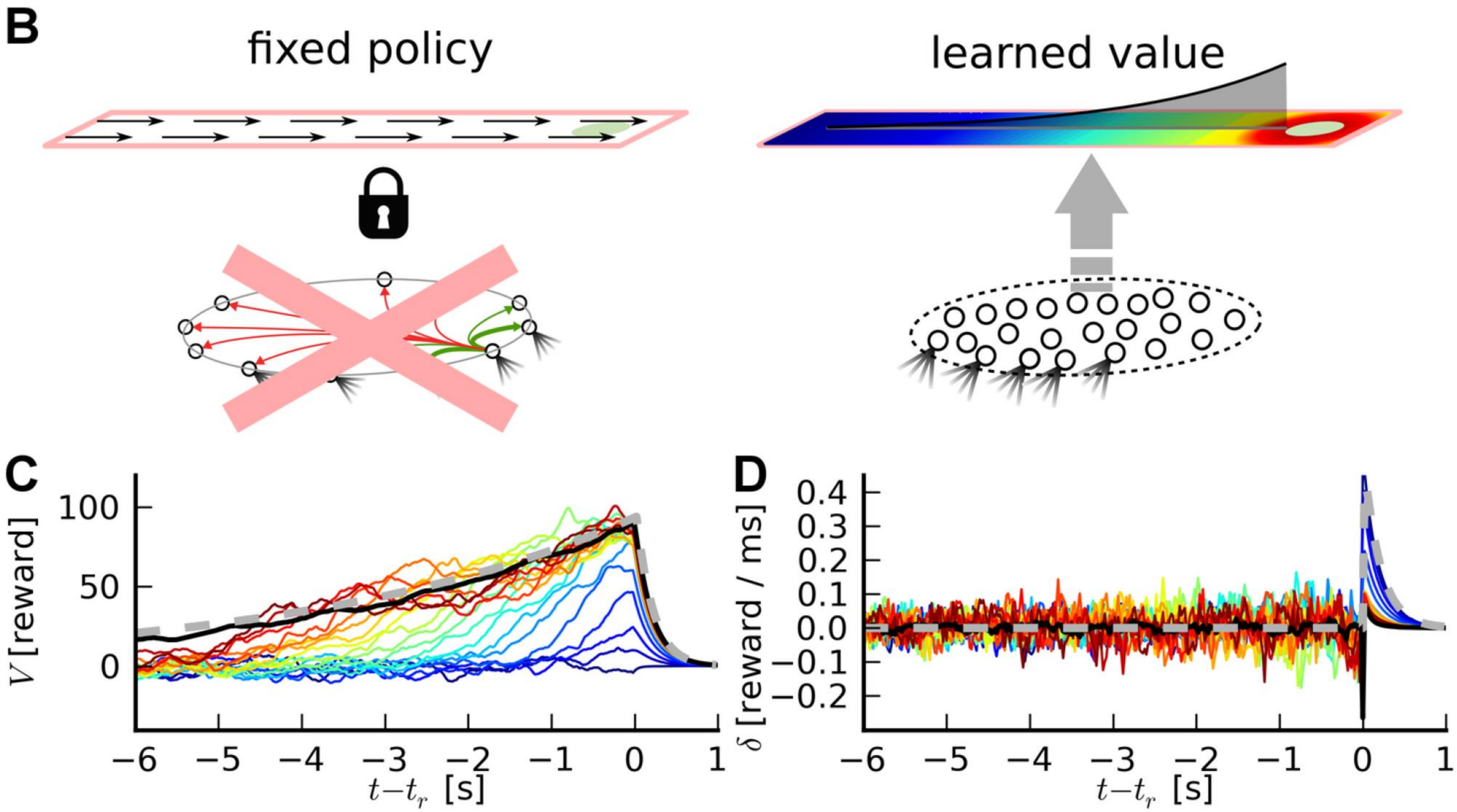
6. Maze Navigation with TD in Actor-Critic



Fremaux et al. (2013)

Figure 1. Navigation task and actor-critic network. From bottom to top: the simulated agent evolves in a maze environment, until it finds the reward area (green disk), avoiding obstacles (red). Place cells maintain a representation of the position of the agent through their tuning curves. Blue shadow: example tuning curve of one place cell (black); blue dots: tuning curves centers of other place cells. Right: a pool of critic neurons encode the expected future reward (value map, top right) at the agent's current position. The change in the predicted value is compared to the actual reward, leading to the temporal difference (TD) error. The TD error signal is broadcast to the synapses as part of the learning rule. Left: a ring of actor neurons with global inhibition and local excitation code for the direction taken by the agent. Their choices depending on the agent's position embody a policy map (top left).
doi:10.1371/journal.pcbi.1003024.g001

6. Critic implements Value



Fremaux et al. (2013)

7. Critic

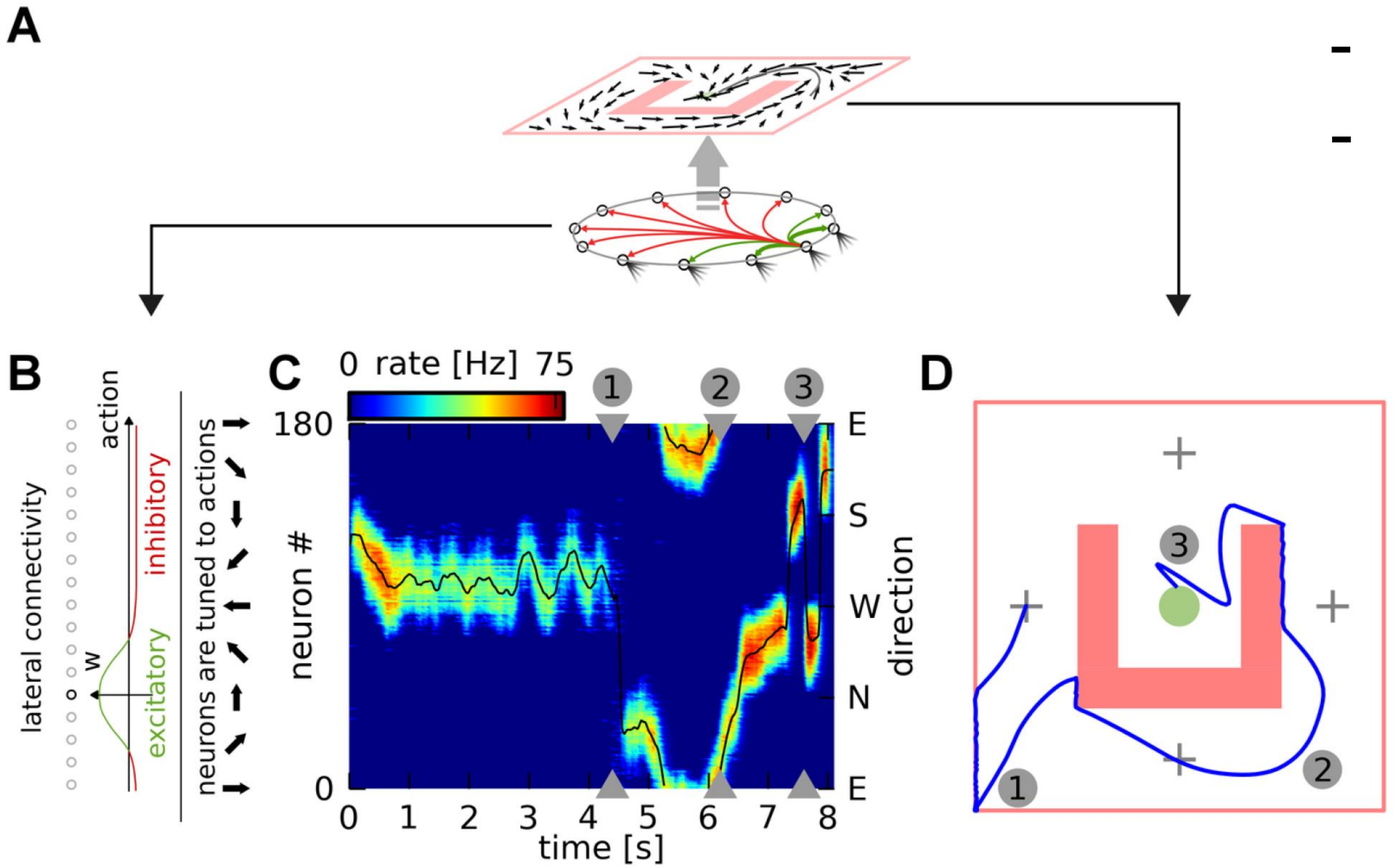
The task of the critic (previous slide)
B: Linear track task. The linear track experiment is a simplified version of the standard maze task. The actor's choice is forced to
the correct direction with constant velocity (left), while the critic learns to represent value (right).
C: Value function learning by the critic. Each colored
trace shows the value function represented by the critic neurons activity against time in the N~20 first simulation trials (from dark blue in trial 1 to dark red in trial 20), with t~tr corresponding to the time of the reward delivery. The black line shows an average over trials 30 to 50, after learning converged. The gray dashed line shows the theoretical value function.
D: TD signal d(t) corresponding to the simulation in C. The gray dashed line shows the reward time course r(t).

Fremaux et al. (2013)

6. Ring of Actor neurons implements policy

Note: no need to formally define a softmax function

- Local excitation
- Long-range inhibition
- Not a formal softmax



Fremaux et al. (2013)

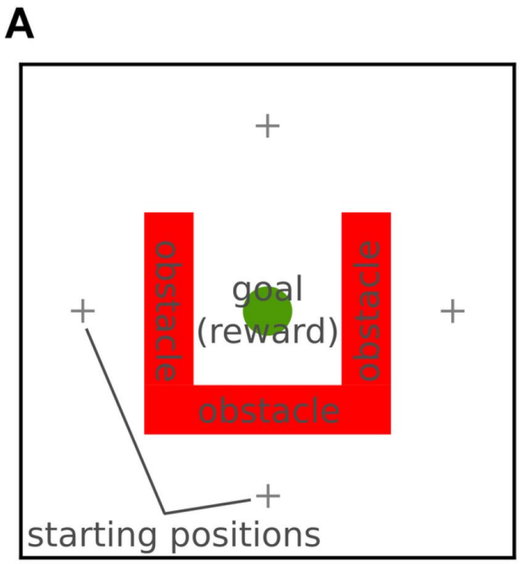
Ring of actor neurons

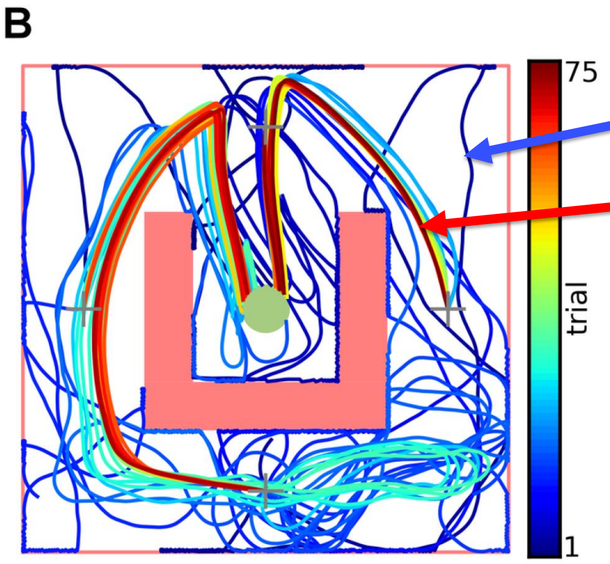
Actor neurons (previous slide).
A: A ring of actor neurons with lateral connectivity (bottom, green: excitatory, red: inhibitory) embodies the agent's policy (top).
B: Lateral connectivity. Each neuron codes for a distinct motion direction. Neurons form excitatory synapses to similarly tuned neurons and inhibitory synapses to other neurons.
C: Activity of actor neurons during an example trial. The activity of the neurons (vertical axis) is shown as a color map against time (horizontal axis). The lateral connectivity ensures that there is a single bump of activity at every moment in time. The black line shows the direction of motion (right axis; arrows in panel B) chosen as a result of the neural activity.
D: Maze trajectory corresponding to the trial
shown in C. The numbered position markers match the times marked in C.

.

Fremaux et al. (2013)

6. Maze Navigation with TD in Actor-Critic with spiking neurons

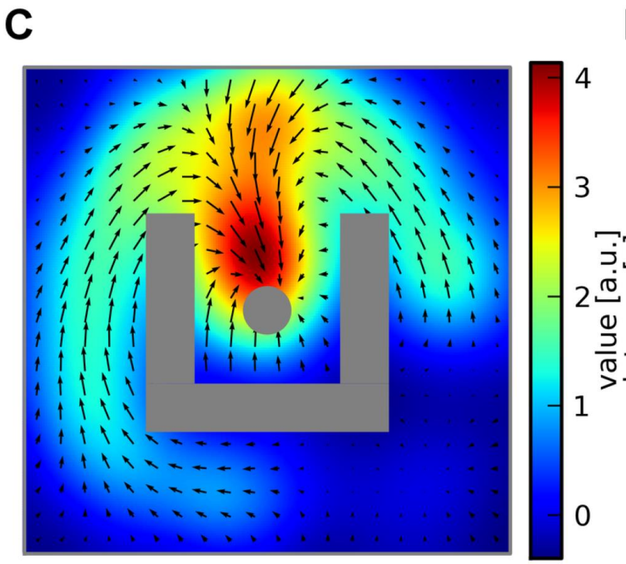


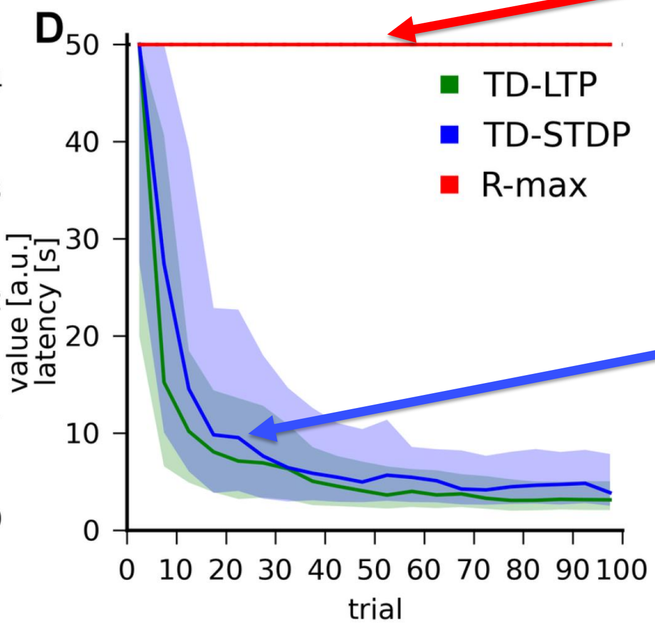


early trial

Late trial

value map





R-max:

Policy gradient without the critic. The goal was never found within 50s.

TD-STDP:

After 25 trials, the goal was found within 20s.

7. Maze Navigation with TD in Actor-Critic with spiking neurons

Maze navigation learning task.
A: The maze consists of a square enclosure, with a circular goal area (green) in the center. A U-shaped obstacle (red) makes the task harder by forcing turns on trajectories from three out of the four possible starting locations (crosses).
B: Color-coded trajectories of an example TD-LTP agent during the first 75 simulated trials. Early trials (blue) are spent exploring the maze and the obstacles, while later trials (green to red) exploit stereotypical behavior.
C: Value map (color map) and policy (vector field) represented by the synaptic weights of the agent of panel B after 2000s simulated seconds.
D: Goal reaching latency of agents using different learning rules. Latencies of N~100 simulated agents per learning rule. The solid lines shows the median shaded area represents the 25th to 75th percentiles. The R-max agent were simulated without a critic and enters times-out after 50 seconds.

Fremaux et al. (2013)

6. TD in Actor-Critic with spiking neurons

- Learns in a few trials (assuming good representation)
- Works in continuous time.
- No artificial ‘events’ or ‘time steps’
- Works with spiking neurons
- Works in continuous space and for continuous actions
- Uses a biologically plausible 3-factor learning rule
- Critic implements value function
- TD signal calculated by critic
- Actor neurons interact via synaptic connections
- No need for algorithmic ‘softmax’

Fremaux et al. (2013)

Previous slide.
Summary of findings

6. Summary

Several aspects of TD learning in an actor-critic framework can be mapped to the brain:

Sensory representation: Cortex and Hippocampus
Actor : Dorsal Striatum
Critic : Ventral Striatum (nucleus accumbens)

TD-signal: Dopamine

Learning in a few trials (not millions!) possible, if the sensory presentation is well adapted to the task

Previous slide. Summary

Artificial Neural Networks: Lecture 10

Policy Gradient Methods

Wulfram Gerstner

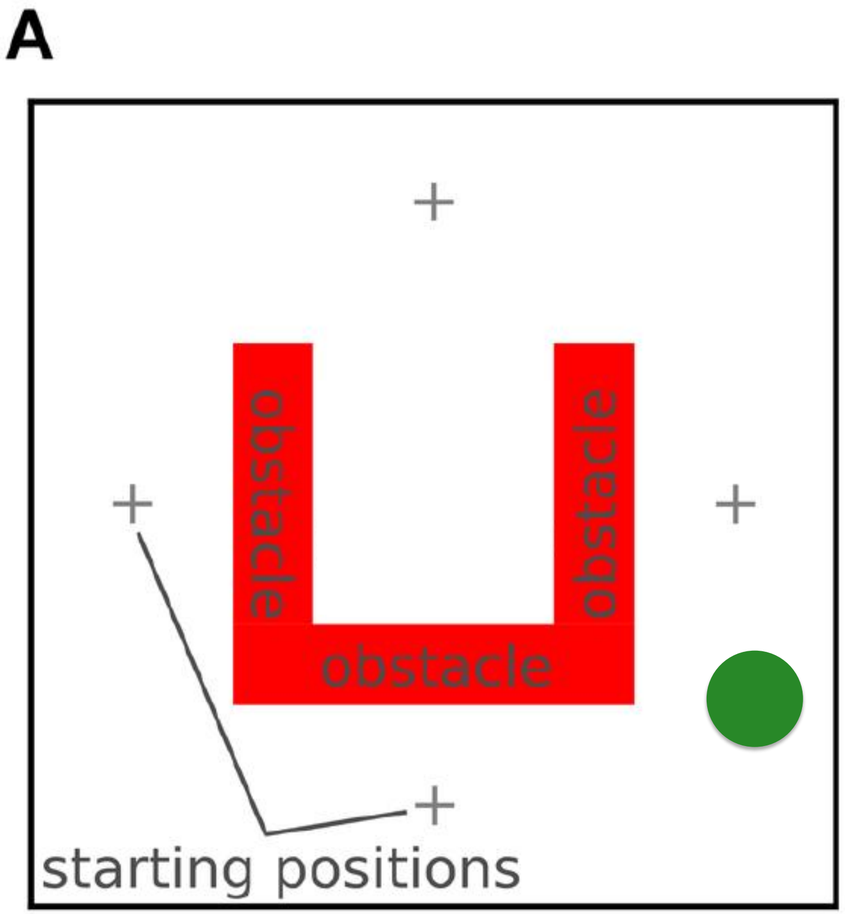
EPFL, Lausanne, Switzerland

1. Review Policy gradient
2. Review Subtracting the mean via the value function
3. Actor-Critic
4. Eligibility traces for policy gradient
5. Actor-Critic in the Brain
6. Application: Rat navigation
7. Model-based versus Model-free

Previous slide.
Final point: are we looking at the right type of RL algorithm?

7. Model-based versus Model-free

What happens in RL when you shift the goal after learning?

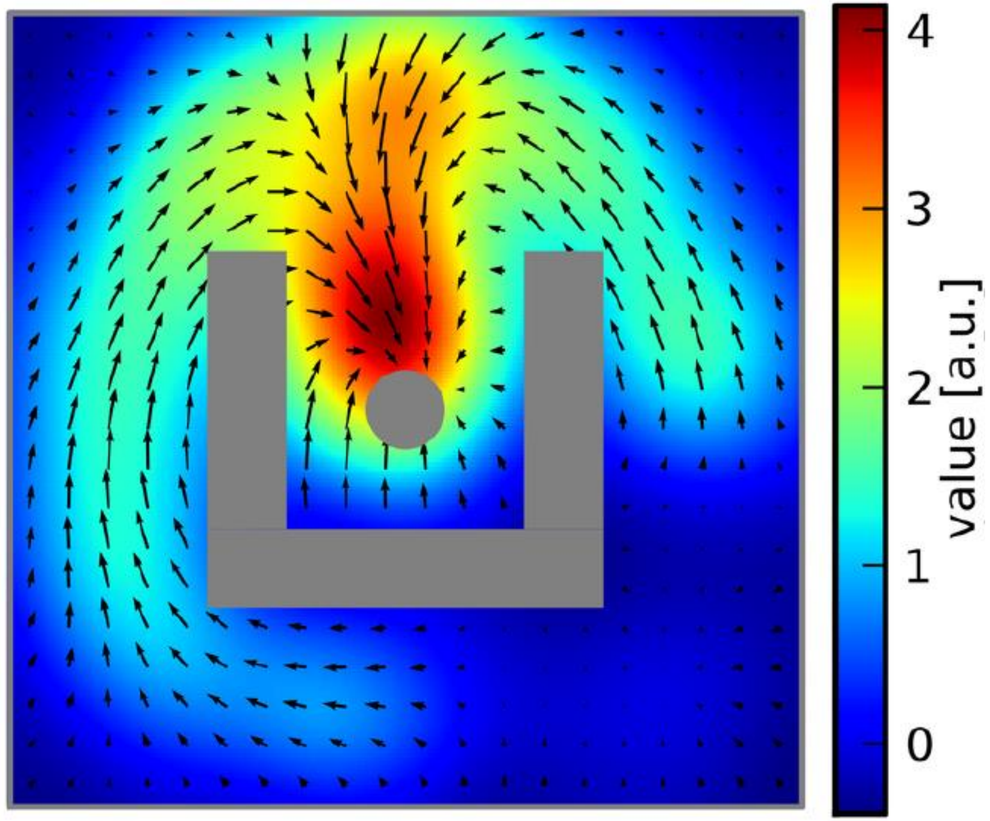


Previous slide.
Final point: are we looking at the right type of RL algorithm?
Imagine that the target location is shifted in the SAME environment.

7. Model-based versus Model-free Reinforcement Learning

What happens in RL when you shift the goal after learning?

→ The value function has to be re-learned from scratch.



agent learns ‘arrows’, but not the lay-out of the environment:
Standard RL is ‘model-free’

Previous slide.
After a shift, the value function has to be relearned from scratch, because the RL algorithm does not build a model of the world. We just learn ‘arrows’: what is the next step (optimal next action), given the current state?

7. Model-based versus Model-free Reinforcement Learning

Definition:
Reinforcement learning is model-free, if the agent does not learn a model of the environment.

Note: of course, the learned actions are always implemented by some model, e.g., actor-critic.
Nevertheless, the term model-free is standard in the field.

Previous slide.
All standard RL algorithms that we have seen so far are ‘model free’.

7. Model-based versus Model-free Reinforcement Learning

Definition:
Reinforcement learning is model-based, if the agent does also learn a model of the environment.

Examples: Model of the environment

- state s1 is a neighbor of state s17.
- if I take action a5 in state s7, I will go to s8.
- The distance from s5 to s15 is 10m.
- etc

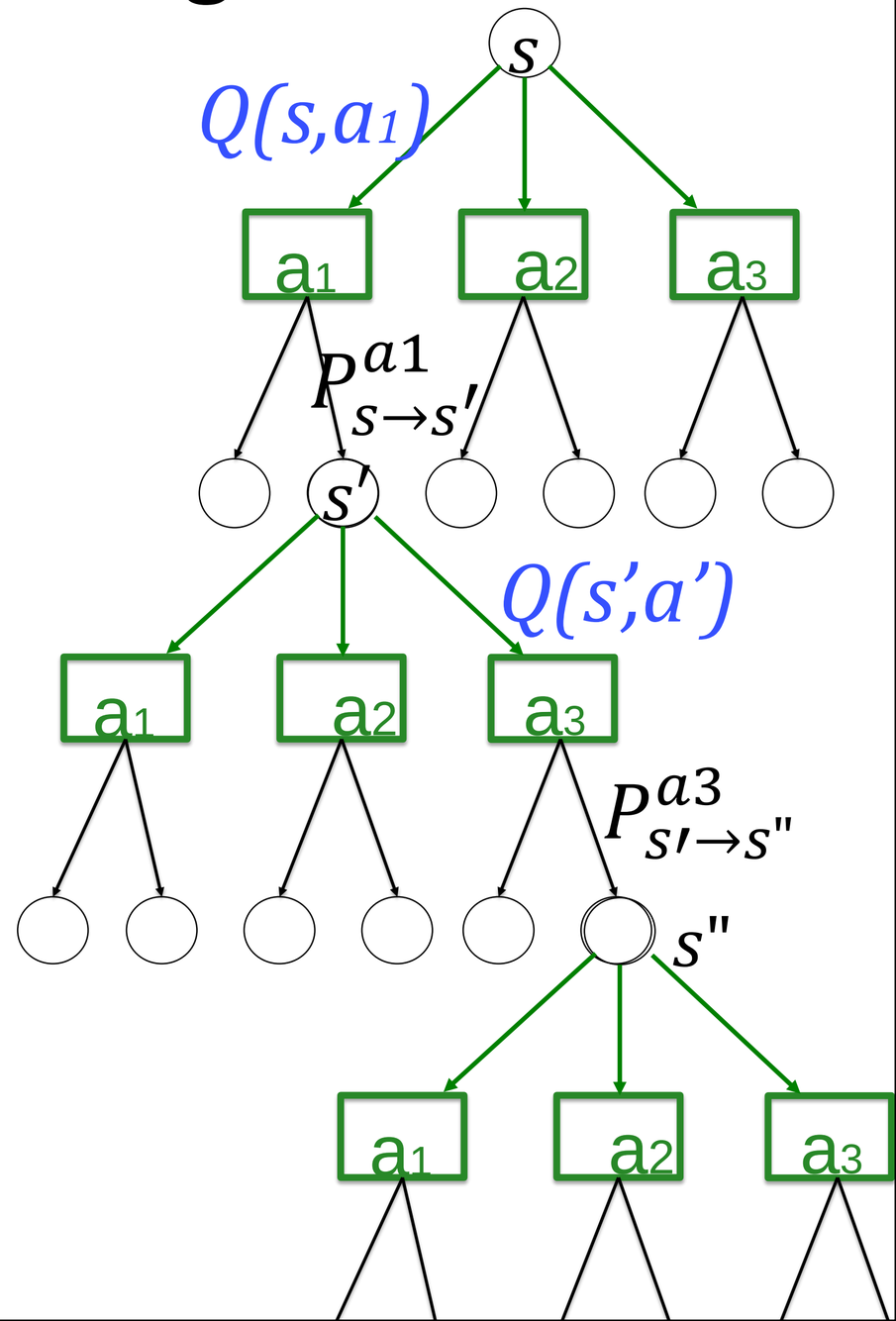
Previous slide.
Examples of knowledge of the environment, that would be typical for model based algorithm

7. Model-based versus Model-free Q-learning

Model-free:
the agent learns directly and only the Q-values

Model-based:
the agent learns the Q-values and also the transition probabilities

$$P^{a1}_{s \to s'}$$



Previous slide.
Let us go back to our ‘tree’. If the algorithm knows the transition probabilities, then this means that it is a model-based algorithm

7. Model-based versus Model-free Reinforcement Learning

Advantages of Model-based RL:

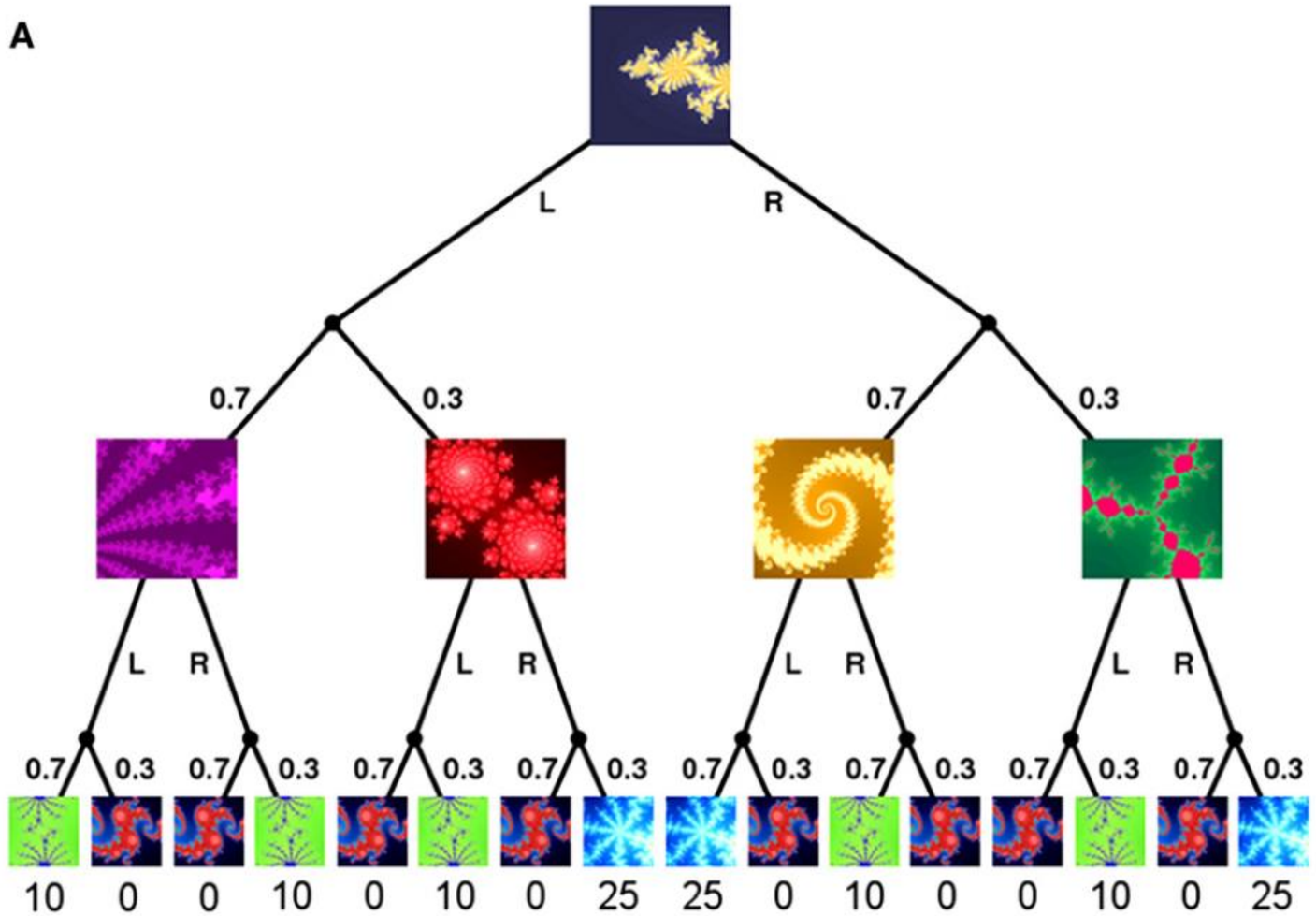
- the agent can readapt if the reward-scheme changes
- the agent can explore potential future paths in its ‘mind’
  → agent can plan an action path
- the agent can update Q-values in the background
  → dream about action sequences
  (run them in the model, not in reality)

Note: Implementations of Chess and Go are ‘model-based’, because the agent knows the rules of the game and can therefore plan an action path. It does not even have to learn the ‘model’.

next slide.

Many modern applications of RL have a model-based component, because you need to play a smaller number of ‘real’ action sequences …
And computer power for running things in the background is cheap.

7. Model-based learning



Model based:
You know what state to expect given current state and action choice.
‘state prediction’

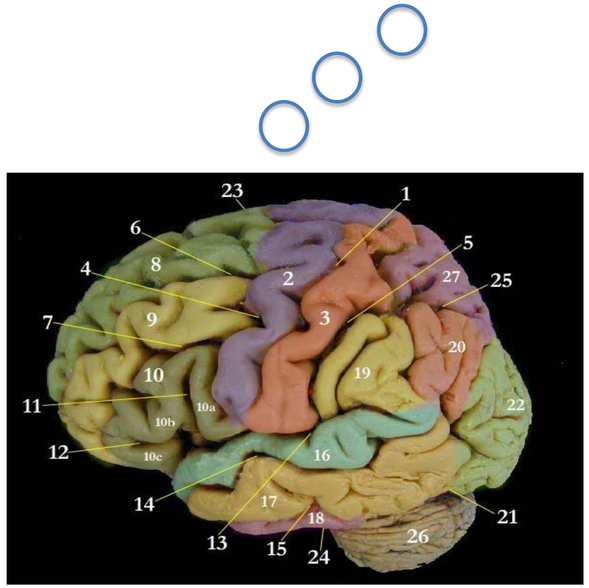
Gläscher et al. 2010

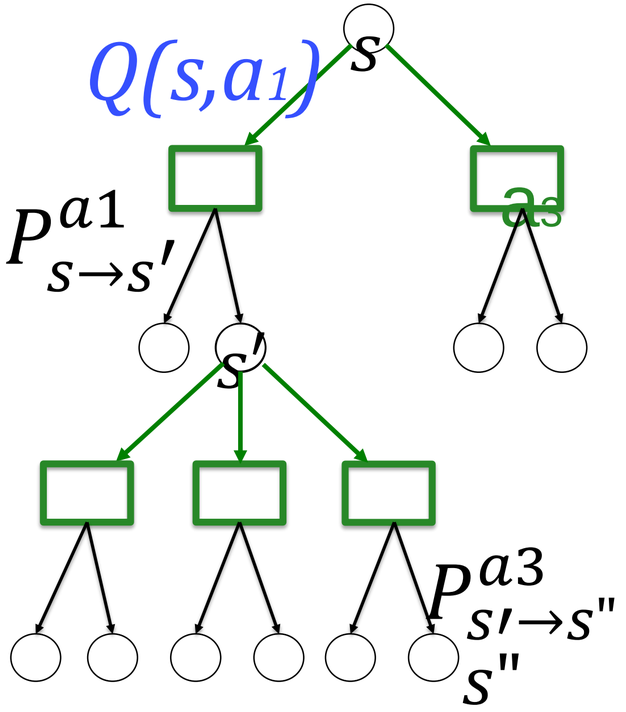
State and Reward Prediction Task (previous slide)
(A) A specific experimental task was a sequential two-choice Markov decision task in which all decision states are represented by fractal images. The task design follows that of a binary decision tree. Each trial begins in the same state. Subjects can choose between a left (L) or right (R) button press. With a certain probability (0.7/0.3) they reach one of two subsequent states in which they can choose again between a left or right action. Finally, they reach one of three outcome states associated with different monetary rewards (0, 10cent, and 25cent).

Gläscher et al. 2010

7. Model-based Reinforcement learning

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





Gläscher et al. 2010

Model based RL allows to think about consequences of actions:
Where will I get a reward?

You just need to play the probabilities forward over the model graph: you simulate an experience before taking the real actions.

Gläscher et al. 2010

Learning outcomes and Conclusions:

- **policy gradient algorithms**
  → updates of parameter propto
- **why subtract the mean reward?**
  → reduces noise of the online stochastic gradient
- **actor-critic framework** $[R \quad]\frac{d}{d\theta_j}\ln[\pi]$
  → combines TD with policy gradient
- **eligibility traces as ‘candidate parameter updates’**
  → true online algorithm, no need to wait for end of episode
- **Differences of model-based vs model-free RL**
  → play out consequences in your mind by running the state transition model wait

Learning outcome for Relation to Biology:

- three-factor learning rules can be implemented by the brain

→ weight changes need presynaptic factor,
postsynaptic factor and a neuromodulator (3rd factor)

→ actor-critic and other policy gradient methods
give rise to very similar three-factor rules

- eligibility traces as 'candidate parameter updates'

→ set by joint activation of pre- and postsynaptic factor

→ decays over time

→ transformed in weight update if dopamine signal comes

- the dopamine signal has signature of the TD error

→ responds to reward minus expected reward

→ responds to unexpected events that predict reward