Learning in Neural Networks (week 9)
Reinforcement Learning and the Brain:
Three-factor learning rules and 'brain-style' computing

Wulfram Gerstner
EPFL, Lausanne, Switzerland

Objectives for today:

- three-factor learning rules can be implemented by the brain
- three-factor rules are consistent with RL
- eligibility traces link correlations with delayed reward
- the dopamine signal has signature of the TD error
- local learning rules: 2-factor and three-factor

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) *Gerstner et al. (2018)* Eligibility traces and plasticiy on behavioral time scales: experimental support for neoHebbian three-factor learning rules, *Frontiers in neural circuits* https://doi.org/10.3389/fncir.2018.00053

(3) Wolfram Schultz et al., (1997) A neural substrate of prediction and reward, SCIENCE, https://www.science.org/doi/full/10.1126/science.275.5306.1593

Review: Biological Motivation of RL

Reinforcement Learning (RL)
→ **Learning by reward**

Field has two roots:
→ Optimization/Markov
Decision Process (MDP)
→ Biology

Questions for today:
- What elements of RL are ‘bio-plausible’?
- Can the brain implement RL?

Previous slide.
animals and humans are able to learn from rewards. This observation has been one of the major drives of RL.

(the other major drive is the theory of Markov Decision Models)

The question then is:

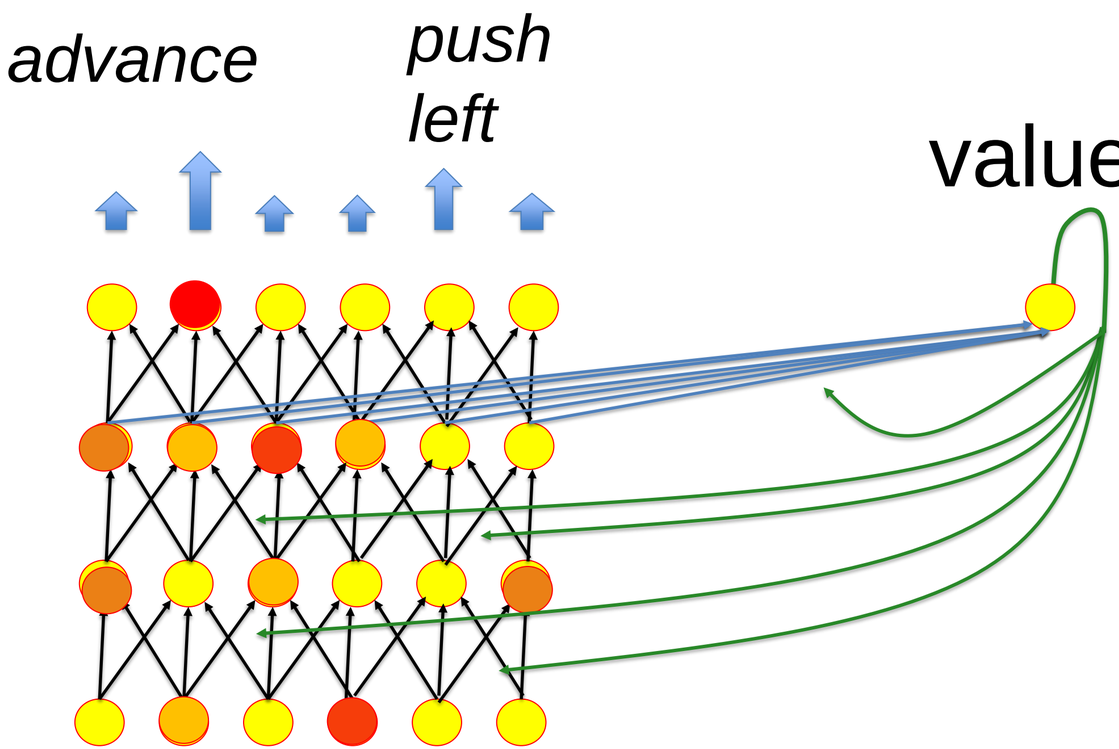
1. can we make the relation to biology more precise?

2. Can we exploit biological insights for unconvential computer hardware?

To answer these questions let us focus on the ‘Learning Rule’.

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

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



The update of parameters depends on the TD error!
The algo for the update is called a ‘learning rule’.

Previous slide.
Let us focus on the ‘Learning Rule’ or ‘update algorithm’ in the actor-critic setup.

There are weights w leading to the actor and other parameters θ leading to the critic.

Learning rule means that we analyze how these parameters change. Thus ‘learning rule’ in biology is a term that refers to the ‘parameter update algorithm’ in the corresponding mathematical learning model.

Review: Advantage Actor-Critic with Eligibility traces

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

Input: a differentiable policy parameterization $\pi(a|s,\boldsymbol{\theta})$
Input: a differentiable state-value function parameterization $\hat{v}(s,\mathbf{w})$
Algorithm parameters: $\lambda^{\mathbf{w}} \in [0,1]$, $\lambda^{\boldsymbol{\theta}} \in [0,1]$, $\alpha^{\mathbf{w}} > 0$, $\alpha^{\boldsymbol{\theta}} > 0$

Initialize state-value weights $\mathbf{w} \in \mathbb{R}^d$ and policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ (e.g., to $\mathbf{0}$)
Initialize $S \in \mathcal{S}$ (e.g., to s_0)

$\mathbf{z}^{\mathbf{w}} \leftarrow \mathbf{0}$ (d-component eligibility trace vector)
$\mathbf{z}^{\boldsymbol{\theta}} \leftarrow \mathbf{0}$ (d'-component eligibility trace vector)
Loop forever (for each time step):
  $A \sim \pi(\cdot|S,\boldsymbol{\theta})$
  Take action A, observe S', r
  $\delta \leftarrow r + \gamma\, \hat{v}(S',\mathbf{w}) - \hat{v}(S,\mathbf{w})$

  $\mathbf{z}^{\mathbf{w}} \leftarrow \lambda^{\mathbf{w}} \mathbf{z}^{\mathbf{w}} + \nabla \hat{v}(S,\mathbf{w})$
  $\mathbf{z}^{\boldsymbol{\theta}} \leftarrow \lambda^{\boldsymbol{\theta}} \mathbf{z}^{\boldsymbol{\theta}} + \nabla \ln \pi(A|S,\boldsymbol{\theta})$
  $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}} \delta \mathbf{z}^{\mathbf{w}}$
  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} \delta \mathbf{z}^{\boldsymbol{\theta}}$
  $S \leftarrow S'$

The algo for the update is the ‘learning rule’.

Adapted from Sutton and Barto

Previous slide. Review from DeepRL1

Red box:
Parameters in the advantage actor critic change proportional to
- The TD error delta
- The derivative of the value function for the critic
- The derivative of the log policy for the actor

In this version of the algo we also have eligibility traces. Set $\lambda=0$ to get a version without eligibility traces.

In the example on the next page eligibility traces are important.

Quiz: Relation of Advantage-Actor-Critic to other Policy Gradient Algos

Assume the transition to state x^{t+1} with a reward of r^{t+1} after taking action a^t at state x^t. The learning rule for the Advantage Actor-Critic with Eligibility traces is

'learning rule' of Advantage Actor-Critic with eligibility trace

$$\delta \leftarrow r^{t+1} + \gamma \hat{v}_w(x^{t+1}) - \hat{v}_w(x^t)$$
$$z^w \leftarrow \lambda^w z^w + \nabla_w \hat{v}_w(x^t)$$
$$z^\theta \leftarrow \lambda^\theta z^\theta + \nabla_\theta \ln[\pi(a_t|s_t,\theta)]$$
$$w \leftarrow w + \alpha^w z^w \delta$$
$$\theta \leftarrow \theta + \alpha^\theta z^\theta \delta$$

[ ] We get the Advantage Actor-Critic **without eligibility trace** if we set $\lambda^w = \lambda^\theta = 0$.

[ ] We get REINFORCE **without baseline** (with eligibility trace) if set $\delta \leftarrow r^{t+1}$

[ ] We get REINFORCE **without baseline** and **without eligibility trace**

if set $\delta \leftarrow R = r^{t+1} + \gamma r^{t+2} +$

[ ] REINFORCE **without baseline and without eligibility trace** has many terms

propto $\nabla_\theta \ln(\pi_\theta(a^t, x^t))$, $\nabla_\theta \ln(\pi_\theta(a^{t+1}, x^{t+1}))$, … and is therefore not an online algorithm

Previous slide. Your notes. All these algorithms have been covered in the lecture on policy gradient methods.

R denotes the return (sum of discounted rewards, starting from state t)

Relation of Advantage-Actor-Critic to other Policy Gradient Algos

Assume the transition to state x^{t+1} with a reward of r^{t+1} after taking action a^t at state x^t. The learning rule for the Advantage Actor-Critic with Eligibility traces is

'learning rule'
of Advantage
Actor-Critic
with eligibility trace

$$
\begin{aligned}
\delta &\leftarrow r^{t+1} + \gamma \hat{v}_w(x^{t+1}) - \hat{v}_w(x^t) \\
z^w &\leftarrow \lambda^w z^w + \nabla_w \hat{v}_w(x^t) \\
z^\theta &\leftarrow \lambda^\theta z^\theta + \nabla_\theta \ln[\pi(a_t|s_t,\theta)] \\
w &\leftarrow w + \alpha^w z^w \delta \\
\theta &\leftarrow \theta + \alpha^\theta z^\theta \delta
\end{aligned}
\qquad (1)
$$

→ Learning rules of other ONLINE RL policy gradient models are special cases of (1).

→ We take (1) as a starting point to discuss the relation with the brain

Can such a **learning rule** be implemented in the brain?

Previous slide. Review from DeepRL1

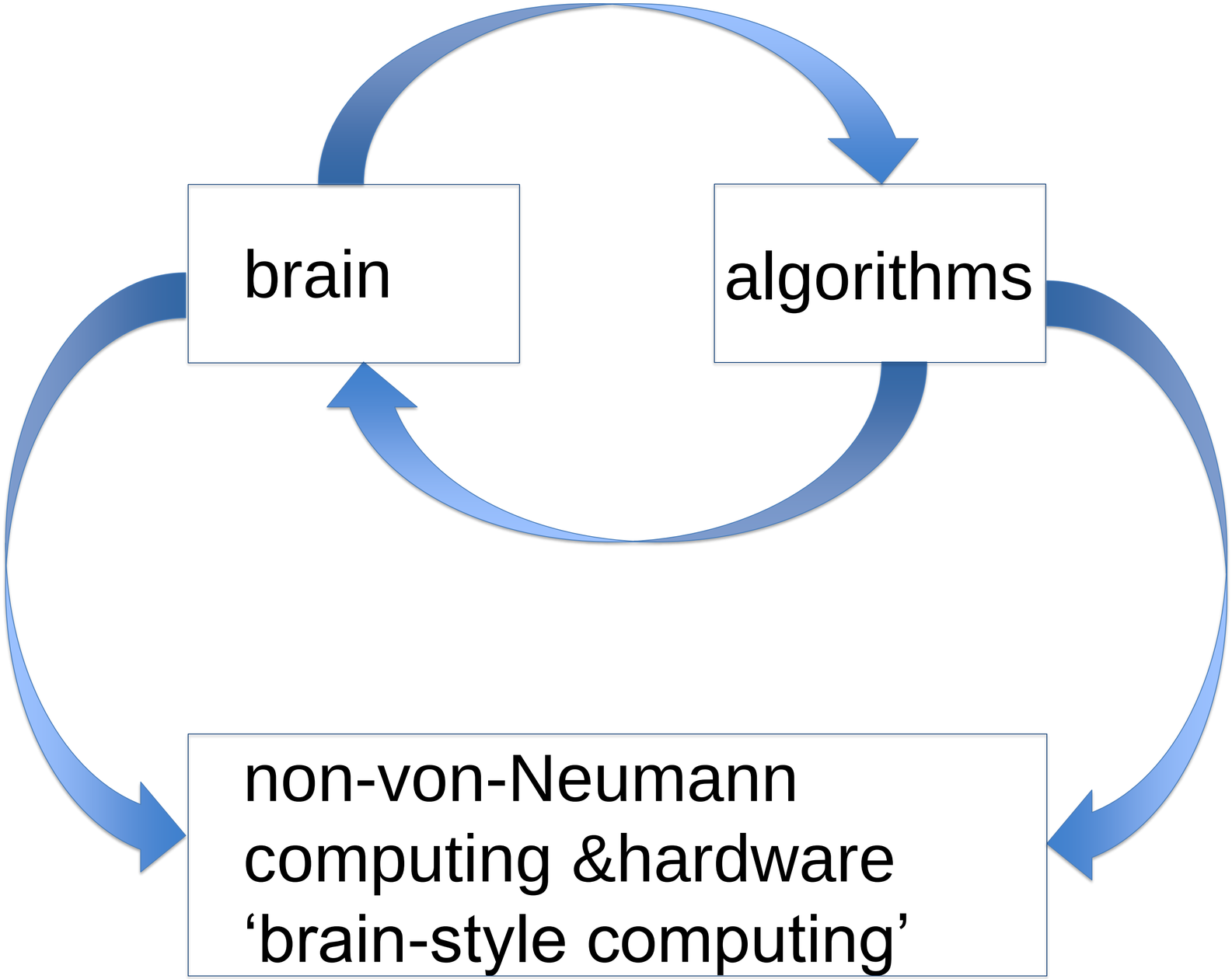
In the following we take the Advantage Actor Critic as our Reference Model.

Other Algorithms in the Family of Policy Gradients can be identified as special cases.

The first big question of this lecture is:
Can such a learning rule (update algorithm) be implemented in the brain?

The second big question of this lecture is (next slide):
Can the elements of an actor-critic architecture be implemented in the brain?

Learning Rules



Previous slide.

Our aim is to connect formal RL algorithms (right-hand side) with elements and structures in the brain (left-hand side).

This comparison will lead us to a non-von-Neumann computing paradigm that is fully distributed without central control, central memory, or central processing units.

This computing paradigm has sometimes been called ‘brain-style computing’.

Aside: similar ‘non-von-Neumann’ computing architectures are also seriously considered today in the hardware community for next-generation chip design or next generation materials! (→ separate lecture)

Questions for this Lecture

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

- What can we learn without Backprop?
- Applications of ‘brain-style computing’?

→ Properties of learning rules: ‘local’, ‘Hebbian’, ‘Three-factor’

There are big research fields interested in these questions:

→ Computational neuroscience

→ Cognitive neuroscience

→ Neuro-economics

→ Clinical Neuralscience of Addiction

Previous slide.

Program for this week.

In this introduction, we have reviewed some aspects of RL in an actor-critic structure, in particular the online ‘learning rule’, i.e., the algorithm for the parameter update after each step of the agent. In the following we focus on the learning rule and go back and forth between algorithms and the brain.

Having identified the basic aspects of the learning rule in RL, we now turn to the biology.

Reinforcement Learning and the Brain:

Three-factor learning rules and 'brain-style' computing

Wulfram Gerstner

EPFL, Lausanne, Switzerland

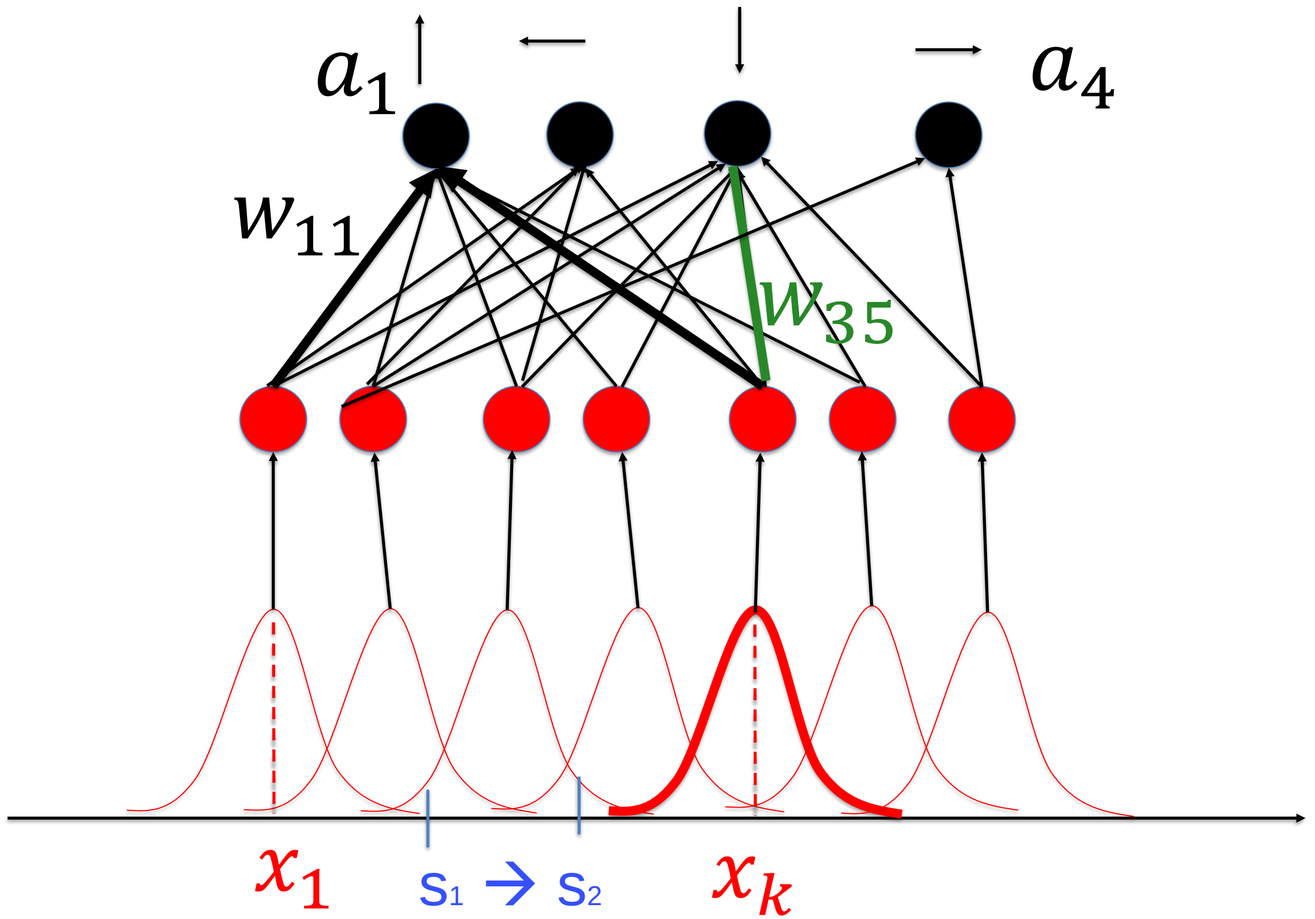
Previous slide.

Many of the slides contain material that we have already seen.
The aim of this lecture is to connect many different aspects together with a focus on the framework of three-factor rules.

Policy gradient rule – can we interpret this as local rule?

$North$: $\boldsymbol{a_1}$=**1**; a_2=a_3 =a_4 = 0

$East$: a_1 =a_2=a_3 =0; $\boldsymbol{a_4}$ = **1**



Discrete actions with
1 hot coding

If at time t, the action
$a_i^t = 1$ is chosen then
$a_j^t = 0$ for all other
output neurons $j \neq i$

Action choice:
Softmax

(previous slide)

1. The policy is softmax:

this implies that output neurons interact interact such that the policy $\pi\left(a_i^t = 1|\vec{x}\right)$ is normalized to

$$\sum_i \pi\left(a_i^t = 1|\vec{x}\right) = 1$$

2. The coding is 1-hot:

This implies that if at time *t*, the action $a_i^t = 1$ is chosen then neuron i sends immediately an output signal to all other neurons to inhibit their activity so that $a_j^t = 0$ for all other output neurons $j \neq i$.

Exercise: Continuous input representation

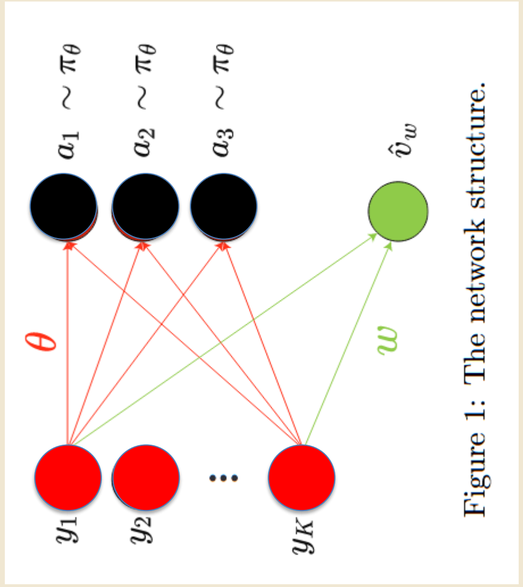


Figure 1: The network structure.

actor-critic update rule

In this exercise you will show how applying Advantage Actor-Critic with eligibity traces to a softmax policy in combination with a linear read-out function leads to a biologically plausible learning rule.

Consider a policy and a value network as in Figure 1 with K input neurons $\{y_k = f(x - x_k)\}_{k=1}^K$. The policy network is parameterized by θ and has three output neurons corresponding to actions a_1, a_2 and a_3 with 1-hot coding. If $a_k = 1$, action a_k is taken. The output neurons are sampled from a softmax policy: The probability of taking action a_i is given by

$$\pi_\theta(a_i = 1|x) = \frac{\exp[\sum_k \theta_{ik} y_k]}{\sum_j \exp[\sum_k \theta_{jk} y_k]}. \tag{1}$$

In addition, consider the exponential value network $\hat{v}_w(x) = \exp\left[\sum_k w_k y_k\right]$.

Assume the transition to state x^{t+1} with a reward of r^{t+1} after taking action a^t at state x^t. The learning rule for the Advantage Actor-Critic with Eligibility traces is

$$\begin{aligned}
\delta &\leftarrow r^{t+1} + \gamma \hat{v}_w(x^{t+1}) - \hat{v}_w(x^t) \\
z^w &\leftarrow \lambda^w z^w + \nabla_w \hat{v}_w(x^t) \\
z^\theta &\leftarrow \lambda^\theta z^\theta + \nabla \ \ln[\pi(a_t|s_t, \theta)] \\
w &\leftarrow w + \alpha^w z^w \delta \\
\theta &\leftarrow \theta + \alpha^\theta z^\theta \delta
\end{aligned}$$

Ex 1 NOW!

Your goal is to show that this learning rule applied to the network of Figure 1 has a biological interpretation.

a. Show that

$$\frac{d}{dw_5} \hat{v}_w(x^t) = y_5^t \hat{v}_w(x^t)\,. \tag{2}$$

b. Interpret the update of the eligibity trace z_5^w in terms of a 'presynaptic factor' and a 'postsynaptic factor'. Can the rule be implemented in biology?

c. Show that

$$\frac{d}{d\theta_{35}} \ln[\pi_\theta(a_i^t = 1|x^t)] = [a_3^t - \pi_\theta(a_3 = 1|x^t)] y_5^t\,. \tag{3}$$

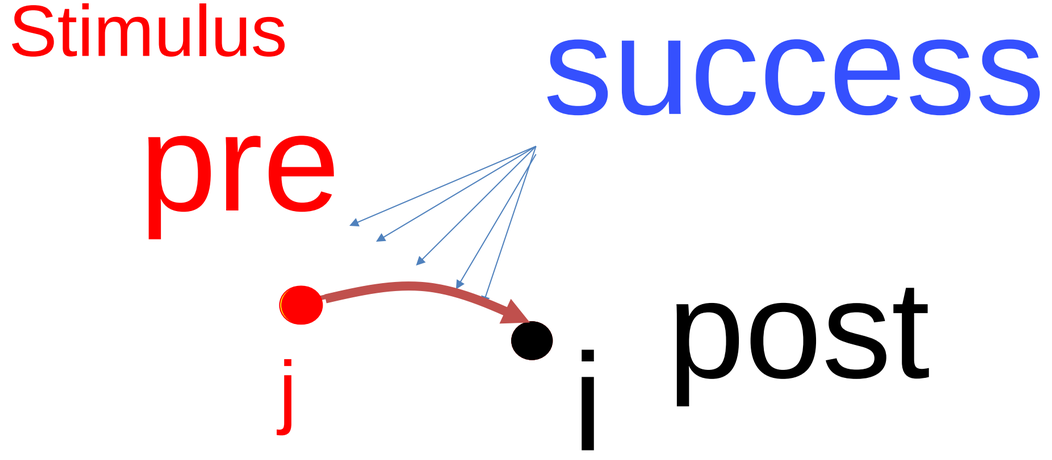
Hint: simply insert the softmax and then take the derivative and exploit 1-hot coding of actions.

d. Interpret the update of the eligibity trace z_{35}^θ in terms of a 'presynaptic factor' and a 'postsynaptic factor'. Can the rule be implemented in biology?

(previous slide)
Your notes.

Discussion of Exercise: Comparison with Biology (no eligibility trace)

parameter = weight w_{ij}



Change depends on pre and post

Three factors: **success** **post** **pre**

$$\Delta w_{ij} = \eta \; S(a_i^t, \vec{x}) [\, a_i^t - \langle a_i(\vec{x}) \rangle] x_j$$

postsynaptic factor is

'activity – expected activity'

Previous slide.

Reinforcement Learning includes a set of very powerful algorithm – as we have seen in previous lectures. Here S denotes the success, which is reward (in REINFORCE) or reward minus baseline (in REINFORCE with baseline),
or TD error (in the advantage actor-critic)

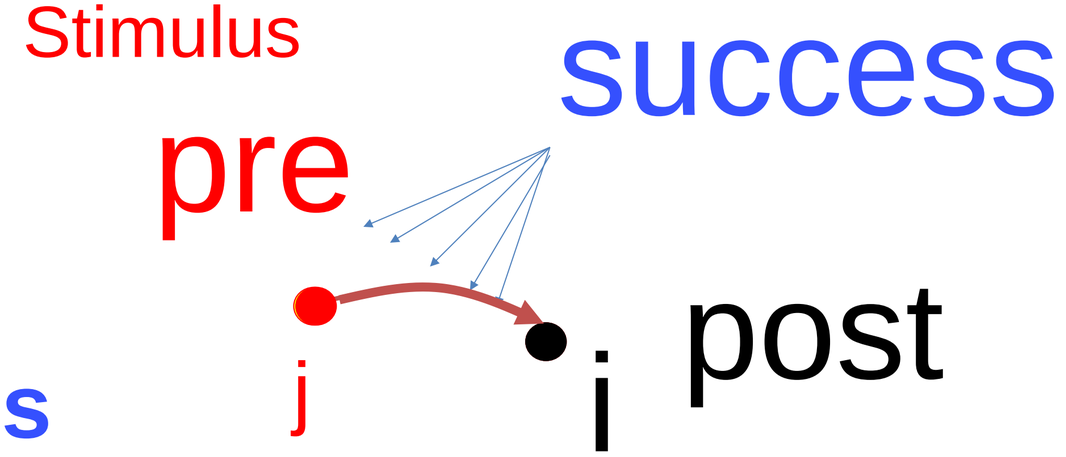
For today the big question is:
Is the structure of the brain suited to implement reinforcement learning algorithms?
If so which one? Q-learning or SARSA? How about Policy gradient?
Is the brain architecture compatible with an actor-critic structure?

These are the questions we will address in the following sections.
A key element is the algorithmic structure of a 'Three-factor Rule'.
The specific rule here is instantaneous (no eligibility trace).
The exercise discusses a version with eligibility trace. If you have calculated the solution with eligibility trace, you can set $\lambda=0$ to remove the eligibility trace.

Three-factor rule (no eligibility trace)

Change depends

- Local factor **pre**
- Local factor **post**
- Global broadcast factor **success**
- **Success could be reward or TD error**



Three factors: **success** **post** **pre**

$$\Delta w_{ij} = \eta \; S(a_i^t, \vec{x}) [\, a_i^t - \langle a_i(\vec{x}) \rangle] x_j$$

postsynaptic factor is

'activity – expected activity'

Previous slide.

The result of Reinforcement Learning with an actor-critic leads to a three-factor rule:

- A presynaptic factor, activity of the sending neuron, such as spike arrival at the synapse.
- A postsynaptic factor: its activity (output spikes, a=1 or inactive a=0) minus the ‘mean drive’ for this state $\langle y_i(\vec{x})\rangle = \pi(a_i|\vec{x}\,)$
- In addition to the above two local factor (similar to a Hebb rule) there is one global broadcasting factor. The success.
- The success could be the reward itself (REINFORCE algorithm), or the TD signal (advantage actor critic).
- The specific version here is the one without eligibility traces. We will come back to eligibility traces later.

Reinforcement Learning and the Brain:

Three-factor learning rules and 'brain-style' computing

Wulfram Gerstner

EPFL, Lausanne, Switzerland

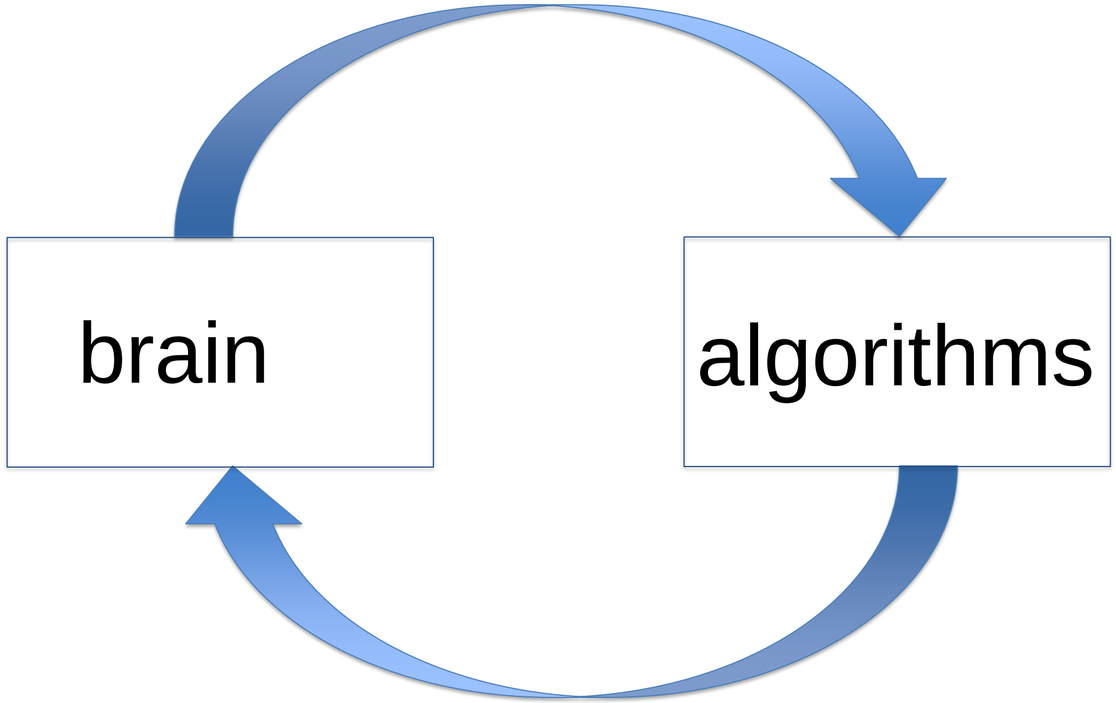
1. Coarse Brain Anatomy
2. Synaptic Plasticity
3. Three-factor Learning Rules
4. **Policy Gradient with Eligibility Traces Revisited**

Previous slide.

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

Learning Rules

3-factor learning rules



Advantage Actor-Critic with eligibility traces

Previous slide.
We will now compare the learning rule of the advantage actor critic with eligibility traces to the three-factor rules of the brain.

We bring together the actor-critic with eligibility traces and the results of exercise 1 today.

4. Eligibility traces from Policy Gradient (Exercise today)

Run episode.

At each time step, observe state s_t, action a_t, reward r_t

1) Update eligibility trace

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

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

2) update parameters: Two variants

Variant A $\Delta w_k = \eta \ r_t \ z_k$ → REINFORCE (w. elig. trace)

Variant B $\Delta w_k = \eta \ \delta_t \ z_k$ → Actor-Critic (w. elig. trace)

Previous slide. repetition of the exercises from week 10 and Exercise of Today
Leads to the algo on slide 7

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

Input: a differentiable policy parameterization $\pi(a|s,\boldsymbol{\theta})$
Input: a differentiable state-value function parameterization $\hat{v}(s,\mathbf{w})$
Algorithm parameters: $\lambda^{\mathbf{w}} \in [0,1]$, $\lambda^{\boldsymbol{\theta}} \in [0,1]$, $\alpha^{\mathbf{w}} > 0$, $\alpha^{\boldsymbol{\theta}} > 0$

Initialize state-value weights $\mathbf{w} \in \mathbb{R}^d$ and policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ (e.g., to $\mathbf{0}$)
Initialize $S \in \mathcal{S}$ (e.g., to s_0)

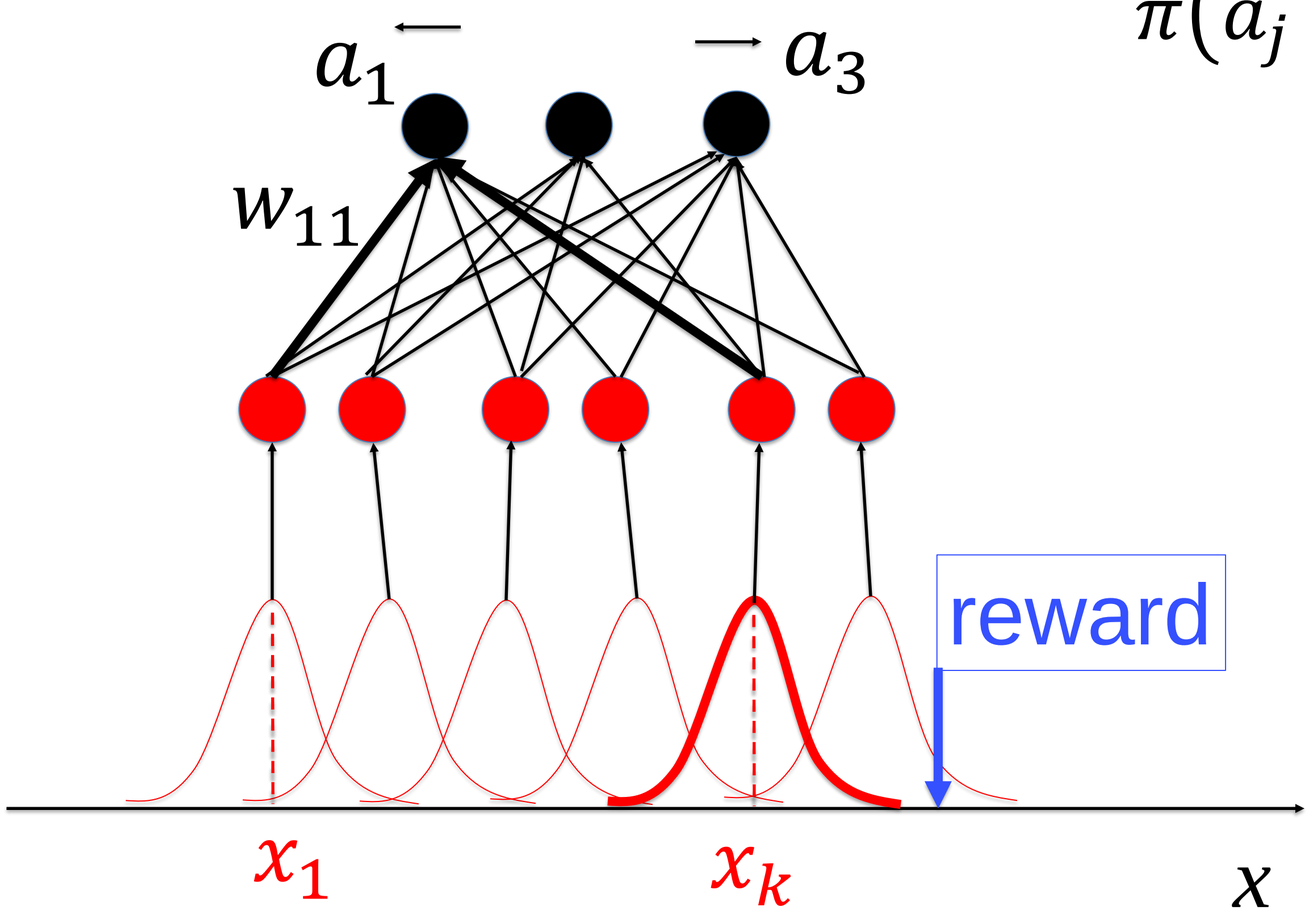
$\mathbf{z}^{\mathbf{w}} \leftarrow \mathbf{0}$ (d-component eligibility trace vector)
$\mathbf{z}^{\boldsymbol{\theta}} \leftarrow \mathbf{0}$ (d'-component eligibility trace vector)
Loop forever (for each time step):
  $A \sim \pi(\cdot|S,\boldsymbol{\theta})$
  Take action A, observe S', r
  $\delta \leftarrow r + \gamma\, \hat{v}(S',\mathbf{w}) - \hat{v}(S,\mathbf{w})$

  $\mathbf{z}^{\mathbf{w}} \leftarrow \lambda^{\mathbf{w}} \mathbf{z}^{\mathbf{w}} + \nabla \hat{v}(S,\mathbf{w})$
  $\mathbf{z}^{\boldsymbol{\theta}} \leftarrow \lambda^{\boldsymbol{\theta}} \mathbf{z}^{\boldsymbol{\theta}} + \nabla \ln \pi(A|S,\boldsymbol{\theta})$
  $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}} \delta \mathbf{z}^{\mathbf{w}}$
  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} \delta \mathbf{z}^{\boldsymbol{\theta}}$
  $S \leftarrow S'$

Adapted from Sutton and Barto

4. Example: Linear activation model with softmax policy

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



parameters

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

$$y_k = f(\vec{x} - x_k)$$

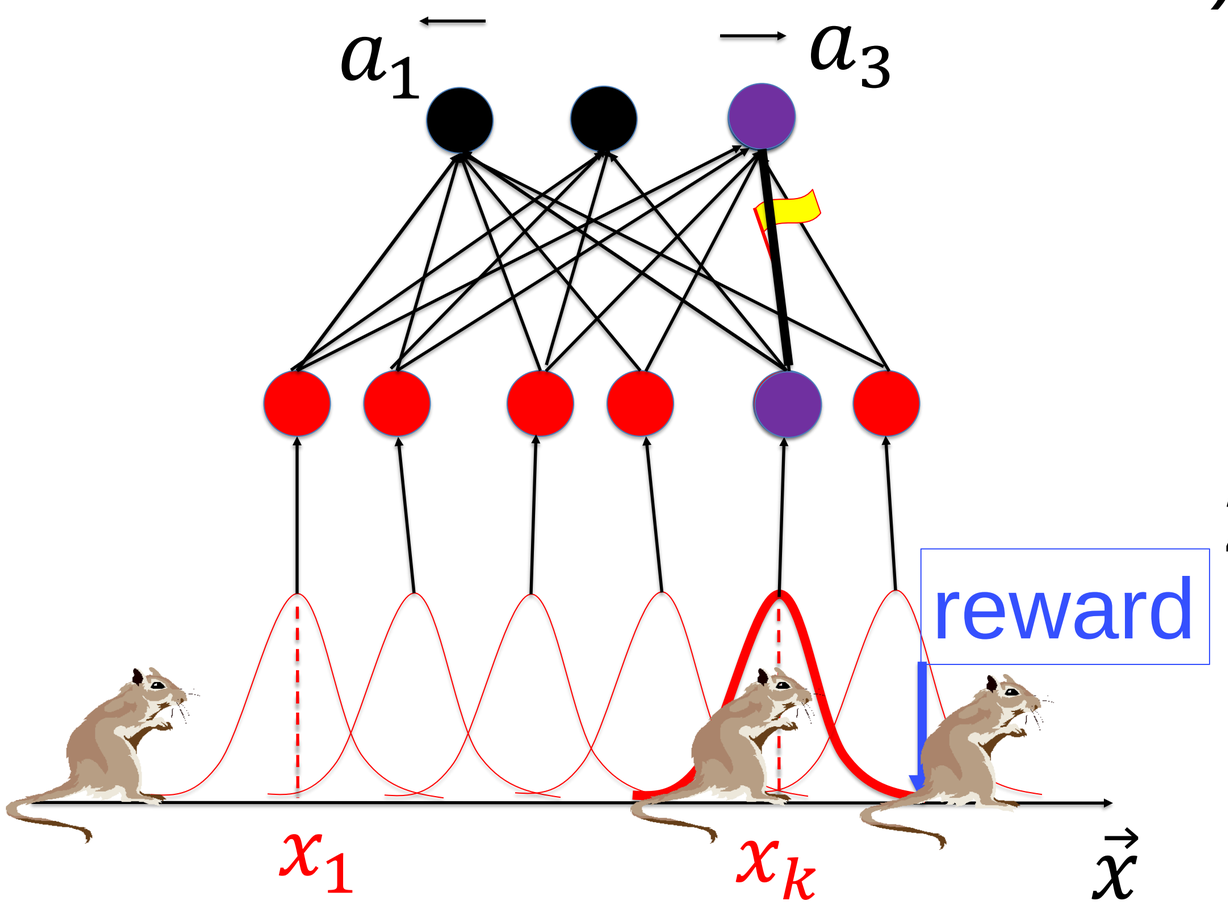
f =basis function

Previous slide.
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.

4. Review Exercise: 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

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

$$z_{ik} \leftarrow z_{ik} + \frac{d}{dw_k}\ln[\pi(a_i^t = 1|\vec{x})]$$

$$\Delta z_{ij} = \eta[\,a_i^t - \langle a_i(\vec{x})\rangle]x_j$$

2) update weights

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

Already done in Exercise
→ Three-factor rule with eligibility traces

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.
The reward is the third factor and has no indices (since it acts as a global factor, broadcasted to all neurons and synapses).

$$\Delta z_{ij} = \eta [\, a_i^t - \langle a_i(\vec{x}) \rangle] x_j$$

4. Summary: 3-factor rules derived 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 the eligibility trace and to either reward r (REINFORCE) or TD error (Adv. Actor-Critic)
- **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
  - → non-von-Neumann compute paradigm

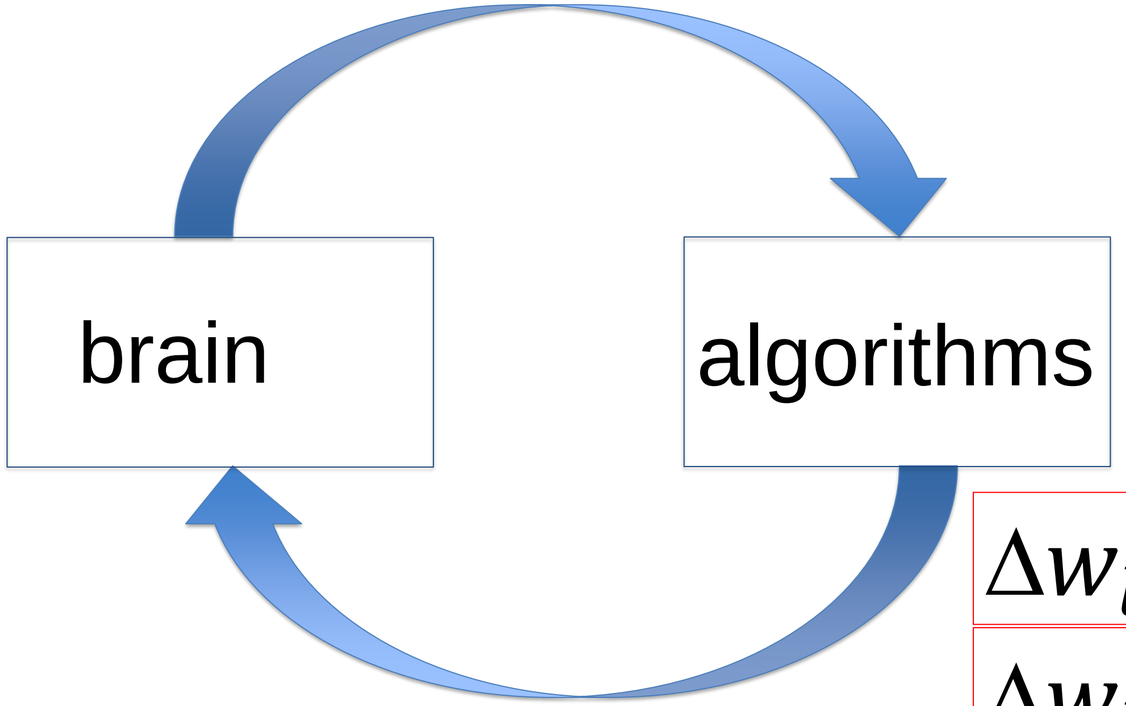
Previous slide.
Summary: A policy gradient algorithm in a network where the output layer has a linear drive with softmax output leads to a three-factor learning rule for the connections between neurons in the hidden layer and the output.

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.

The global modulator could present either the reward r directly (in the style of the REINFORCE algorithm); or it could present the TD error (which yields an interpretation as advantage actor-critic.

Which one of the two possibilities would fit the dopamine signal?
This is the next question

Learning Rules



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

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

The learning rule of the (advantage) actor-critic or REINFORCE with eligibility traces are both compatible with three-factor rules

Updates proportional to the reward r or TD error δ_t

Review: Advantage Actor-Critic with Eligibility traces

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

Input: a differentiable policy parameterization $\pi(a|s,\boldsymbol{\theta})$
Input: a differentiable state-value function parameterization $\hat{v}(s,\mathbf{w})$
Algorithm parameters: $\lambda^{\mathbf{w}} \in [0,1]$, $\lambda^{\boldsymbol{\theta}} \in [0,1]$, $\alpha^{\mathbf{w}} > 0$, $\alpha^{\boldsymbol{\theta}} > 0$

Initialize state-value weights $\mathbf{w} \in \mathbb{R}^d$ and policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ (e.g., to $\mathbf{0}$)
Initialize $S \in \mathcal{S}$ (e.g., to s_0)

$\mathbf{z}^{\mathbf{w}} \leftarrow \mathbf{0}$ (d-component eligibility trace vector)
$\mathbf{z}^{\boldsymbol{\theta}} \leftarrow \mathbf{0}$ (d'-component eligibility trace vector)
Loop forever (for each time step):
  $A \sim \pi(\cdot|S,\boldsymbol{\theta})$
  Take action A, observe S', r
  $\delta \leftarrow r + \gamma\, \hat{v}(S',\mathbf{w}) - \hat{v}(S,\mathbf{w})$

  $\mathbf{z}^{\mathbf{w}} \leftarrow \lambda^{\mathbf{w}} \mathbf{z}^{\mathbf{w}} + \nabla \hat{v}(S,\mathbf{w})$
  $\mathbf{z}^{\boldsymbol{\theta}} \leftarrow \lambda^{\boldsymbol{\theta}} \mathbf{z}^{\boldsymbol{\theta}} + \nabla \ln \pi(A|S,\boldsymbol{\theta})$
  $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}} \delta \mathbf{z}^{\mathbf{w}}$
  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} \delta \mathbf{z}^{\boldsymbol{\theta}}$
  $S \leftarrow S'$

← TD signal

Adapted from Sutton and Barto

5. Combine Eligibility Traces with TD in Advantage 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:

$$\Delta w_k = \eta \ [r - (V(s) - \gamma V(s'))] \ z_k$$

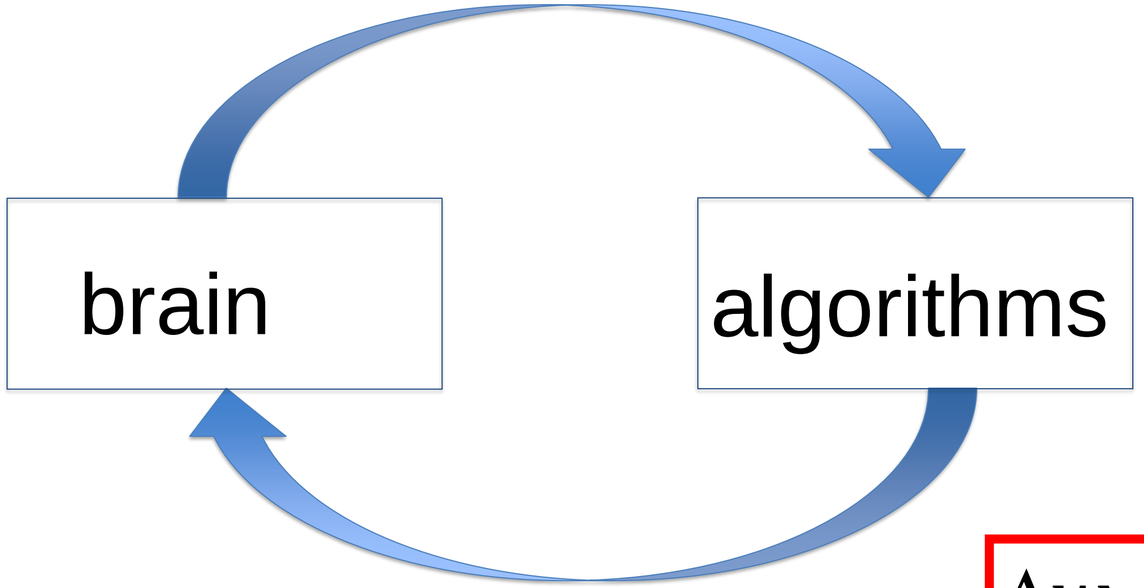
TD-delta

→ policy gradient with eligibility trace and TD error

Previous slides.

As a reminder (not shown in class). Review of algorithm with actor-critic architecture and policy gradient with eligibility traces and TD.

Learning Rule for Advantage Actor Critic



$[r+\gamma V(s')-V(s)]$

TD signal

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

The learning rule of the **advantage** actor-critic with eligibility traces is consistent with a brain-like three-factor rule

Conditions: 1) the brain can broad-cast a TD signal!
2) state representation is good in the layer before action selection

Previous slide.
The main difference between standard REINFORCE with eligibility traces and the Advantage Actor Critic is that
in the advantage actor-critic the global modulator represents the TD error
whereas it represents the immediate reward for REINFORCE.

Both would be compatible with brain-like learning rules.

We now show that the TD signal is consistent with the dopamine signal!

Reinforcement Learning and the Brain:

Three-factor learning rules and 'brain-style' computing

Wulfram Gerstner

EPFL, Lausanne, Switzerland

1. Coarse Brain Anatomy
2. Synaptic Plasticity
3. Three-factor Learning Rules
4. Policy Gradient with Eligibility Traces Revisited
5. **Dopamine as a Third Factor is a TD-like signal**

Previous slide.
So far the third factor remained rather abstract. We mentioned that a neuromodulator such as dopamine could be involved. Let us make this idea more precise and show experimental data.

5. Neuromodulators as Third factor

Three factors are needed for synaptic changes:

- Presynaptic factor = spikes of presynaptic neuron
- Postsynaptic factor = spikes of postsynaptic neuron or increased voltage
- 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.
Before we start let us review the basics of a three-factor learning rule. We said that the third factor could be a neuromodulator such as dopamine.

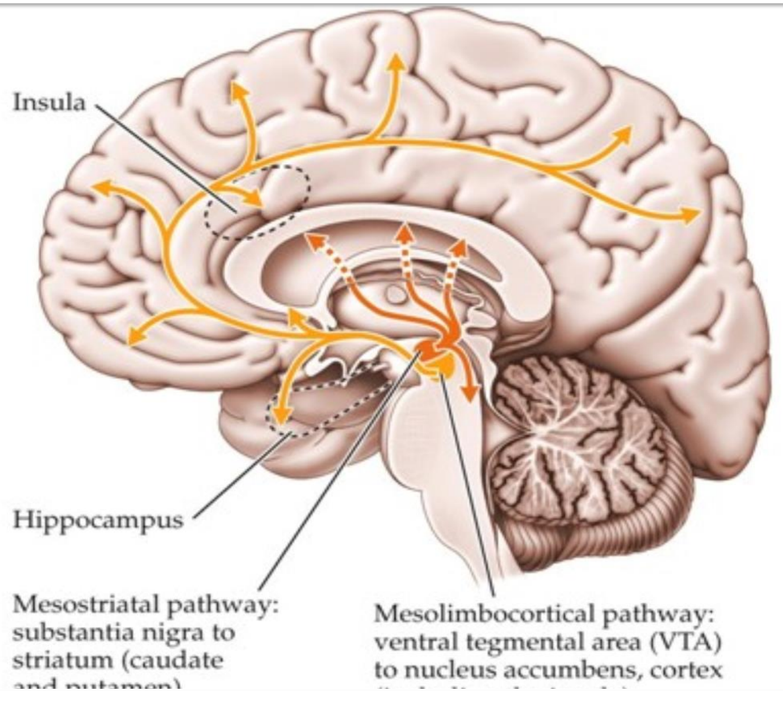
Review: Reward information

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 →touch lever →1s→reward

action

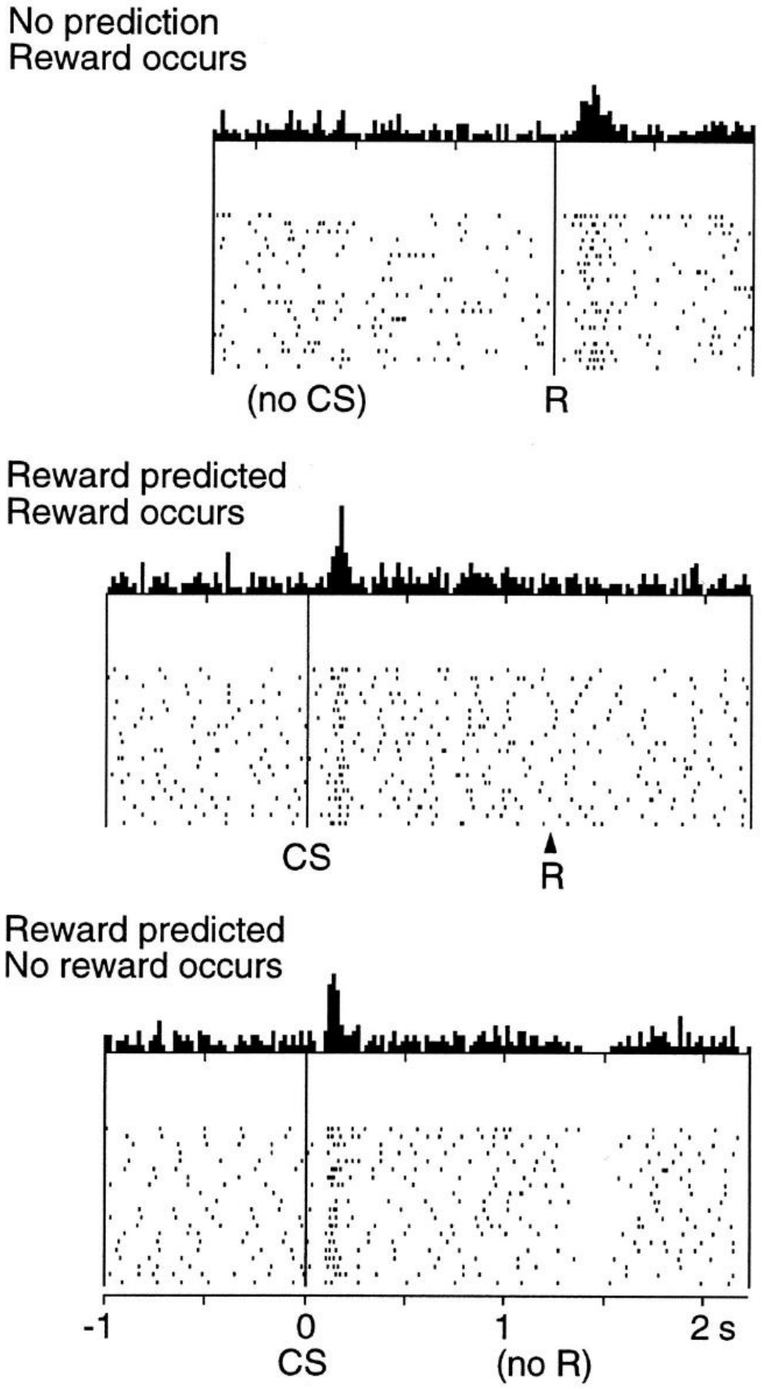
$r(t)$

CS:
Conditioning
Stimulus

$$DA(t) = [r(t) + \gamma V(s') - V(s)]$$

TD-delta

Sutton book, reprinted from W. Schultz (1997, Science)



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

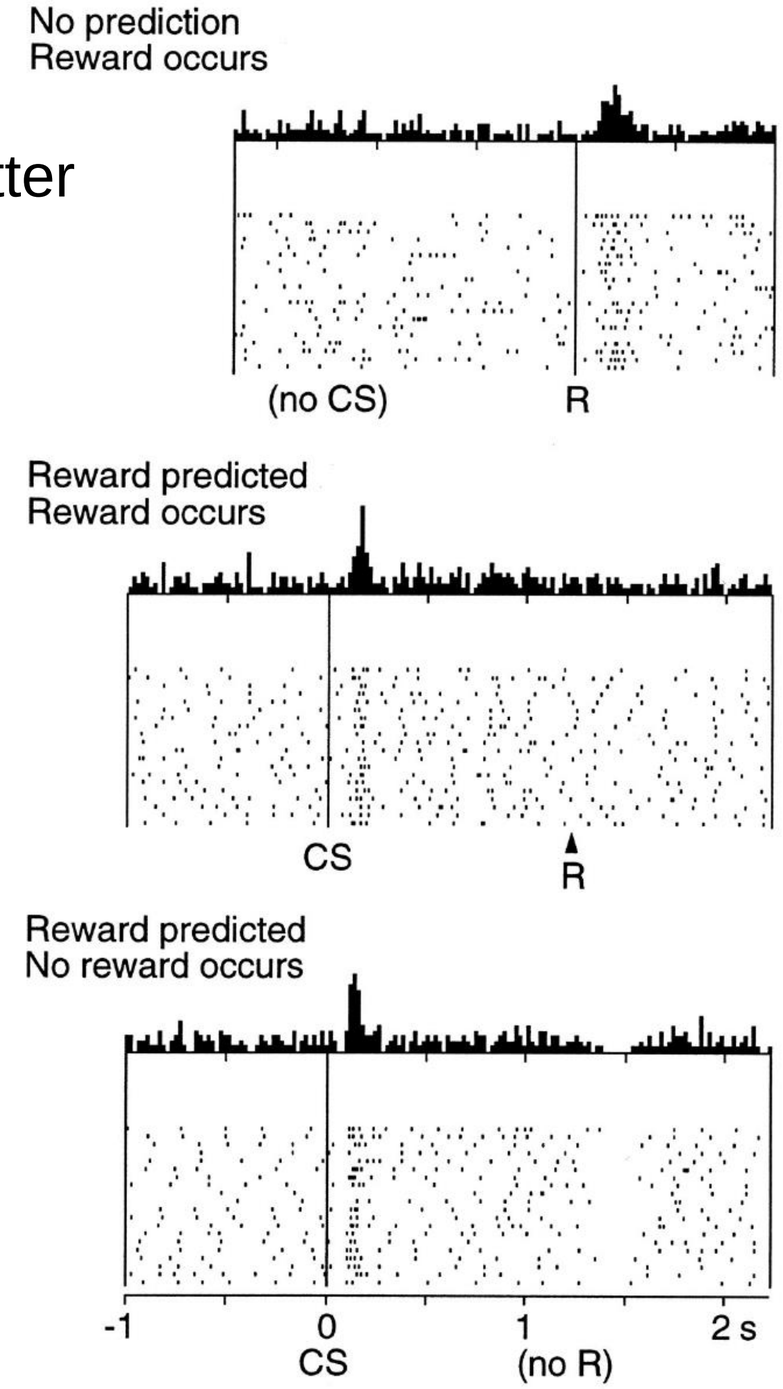


Figure 15.3: The response of dopamine neurons drops below baseline shortly after the time when an expected reward fails to occur. Top: dopamine neurons are activated by the unpredicted delivery of a drop of apple juice. Middle: dopamine neurons respond to a conditioned stimulus (CS) that predicts reward 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

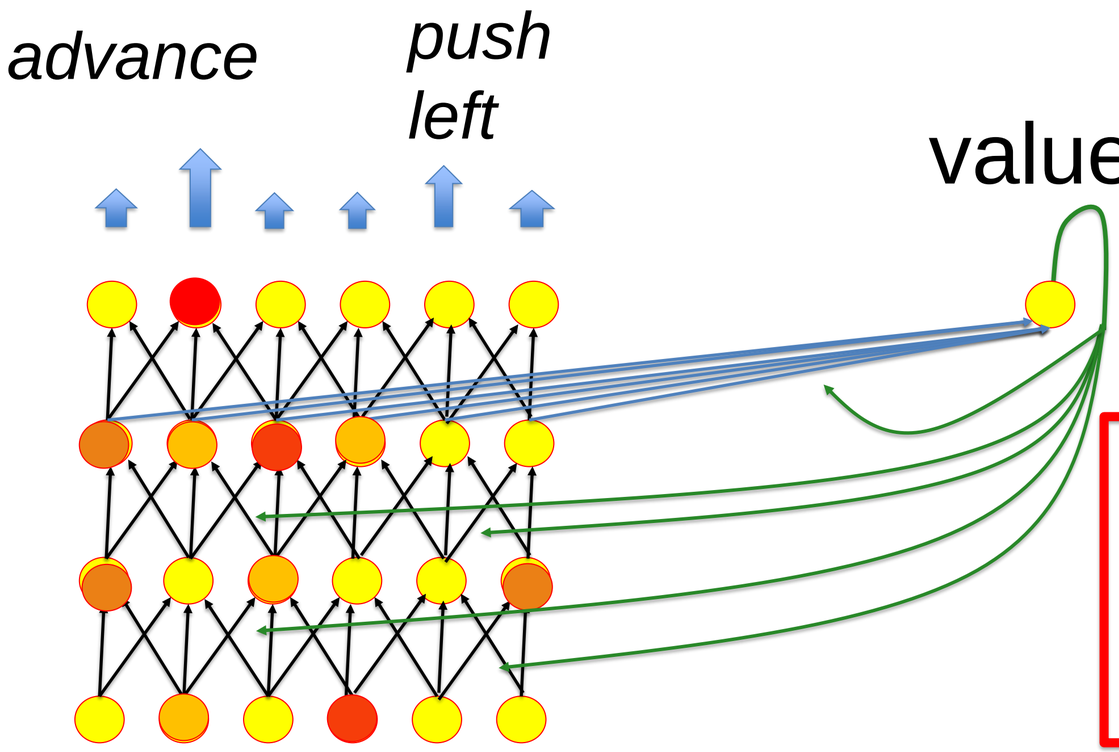
$$DA(t) = [\underbrace{r(t) + \gamma V(s') - V(s)}_{\text{TD-delta}}]$$

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.

5. Application: Advantage Actor-Critic = update with TD signal

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



Dopamine = TD-error

$$\delta = [r_t + \gamma V(s') - V(s)]$$

Previous slide.

Review of actor-critic architecture

5. 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

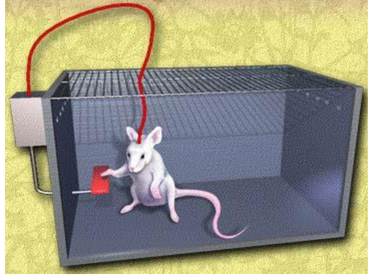
→ Need actor-critic architecture to calculate $\gamma V(s') - V(s)$

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

Previous slide.

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

Action of dopamine
(from paper by W. Schultz et al. 1997)



- drugs like amphetamine and cocaine exert their **addictive** actions in part by prolonging the influence of dopamine on target neurons. Heroin and Cannabis increase dopamine concentration in nucleus accumbens.

- rats press bars to excite dopamine neurons via an implanted electrode. The rats often choose these ‘apparently rewarding’ stimuli over food and sex.

- animals treated with dopamine receptor blockers learn less rapidly to press a bar for a reward pellet.

Previous slide.
Dopamine is an important molecule, one of the neuromodulators. It is related to reward-based learning and closely linked to addiction.

Tanda, Pontieri, DiChiara
https://www.science.org/doi/full/10.1126/science.276.5321.2048
Dichiara et al, 2003
https://www.sciencedirect.com/science/article/pii/S0028390804002199

Self-stimulation:
Phillips and Fiebiger,
https://psycnet.apa.org/record/1980-00546-001

Dopamine and Motor learning
R.J. Beninger (1983)
https://www.sciencedirect.com/science/article/pii/0165017383900383

5. Summary: Dopamine as a Reinforcement Signal

Dopamine is a **brain-internal broadcast signal**
sometimes called **'intrinsic reward system'**, triggered by

→ (extrinsic) reward: chocolate, sweet food, drugs,
…. for humans also: 'praise', money

→ not just reward: also 'surprise', 'novelty'

→ more than reward: 'reward minus expected reward'

→ involved in drug addiction

Games and social networks try to make participants/users addicted by stimulating the 'intrinsic reward system'.

Example: - decrease the expected reward for some time, and return then back to 'normal' reward back.
- add a stochastic component
- bonus points / reach next level

$$\delta = [r_t + \gamma V(s') - V(s)]$$

Previous slide.

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

5. Summary

Learning outcome: RL learning rules and the brain

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

→ synaptic 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

→ decay 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

- addiction hijacks the dopamine reward system

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) *Gerstner et al. (2018)* Eligibility traces and plasticiy on behavioral time scales: experimental support for neoHebbian three-factor learning rules, *Frontiers in neural circuits* https://doi.org/10.3389/fncir.2018.00053

(3) Wolfram Schultz et al., (1997) A neural substrate of prediction and reward, SCIENCE, https://www.science.org/doi/full/10.1126/science.275.5306.1593

(4) Wolfram Schultz (2002) Getting formal with dopamine and reward Neuron 36 (2), 241-263 https://www.sciencedirect.com/science/article/pii/S089662730200967 4

[ ] At least 60 percent of the material was new to me

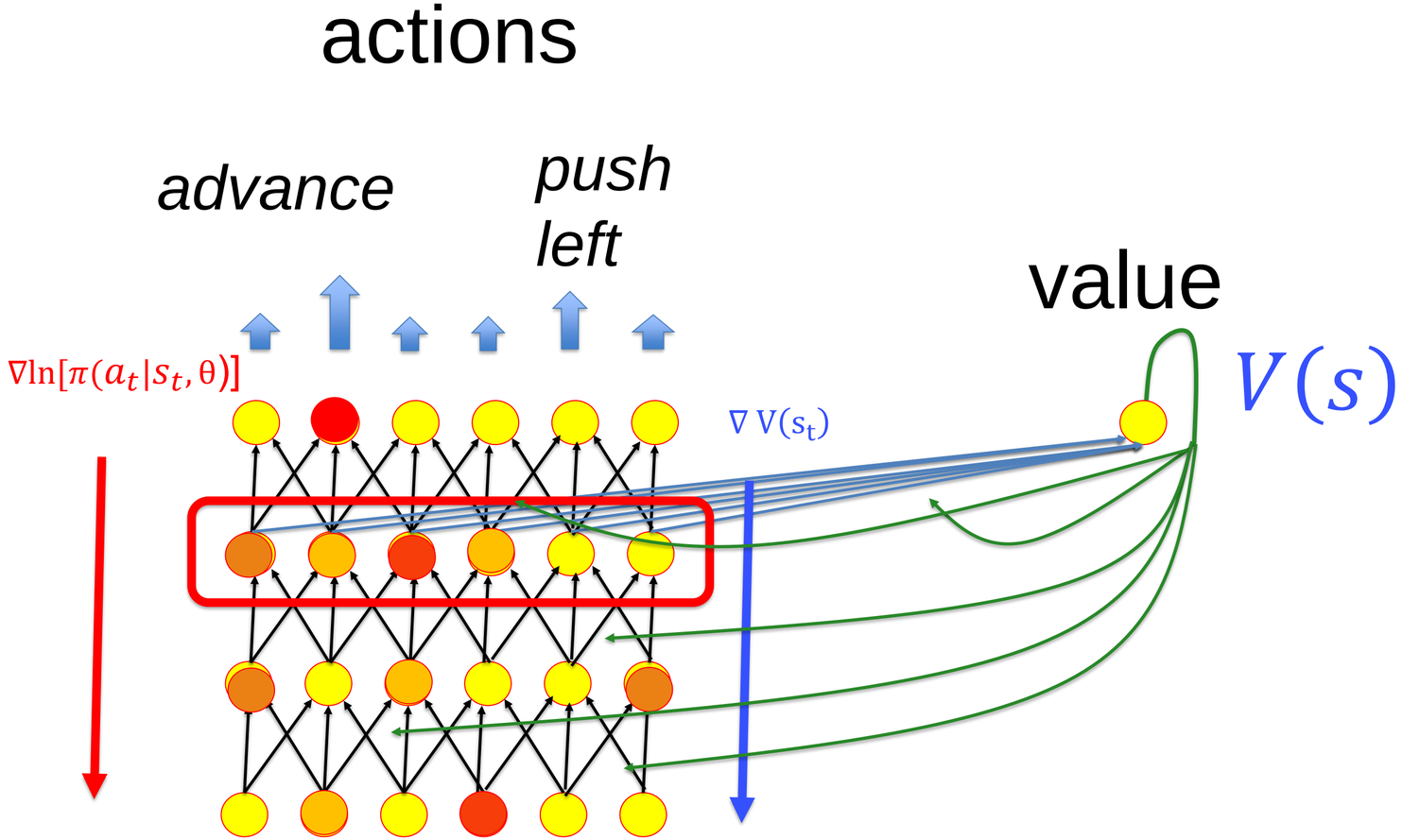
[ ] I have the feeling that I understood 80 percent or more

[ ] ‘Even though’ I study CS/Math/Physics/EE, I found the links to learning in biology interesting

THE END

Previous slide.

5. Application: Advantage Actor-Critic = update with TD signal



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

Dopamine = TD-error

$$\delta = [r_t + \gamma V(s') - V(s)]$$

Deep RL: parameters of AAC optimized with BackProp

Biology: ‘need good representation of state in final layer

→ no BackProp please!

Previous slide.

Review of actor-critic architecture

Learning in Neural Networks:
Three-factor rules for navigation

Wulfram Gerstner
EPFL, Lausanne, Switzerland

Navigation in a Maze (Model Study)

- What is the task?
- How are ‘states’ represented?
- How are ‘actions’ represented?
- How is the ‘learning rule’ represented?
- How are ‘eligibility traces’ implemented?

→Use three-factor rule as a framework for learning

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 in a maze.

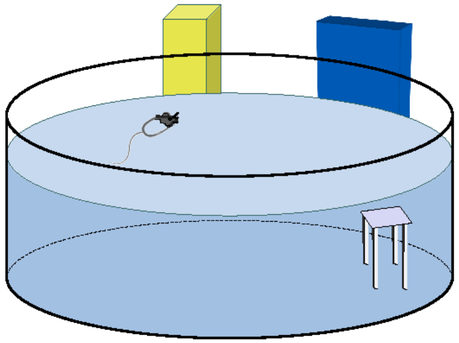
The navigation task is relevant for animals: find a place with food (or shelter) and later return to it.

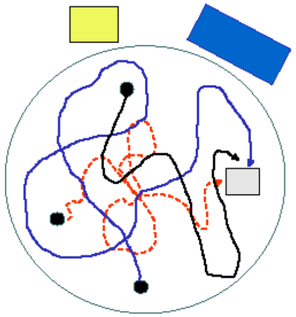
For biological plausibility we have to consider:

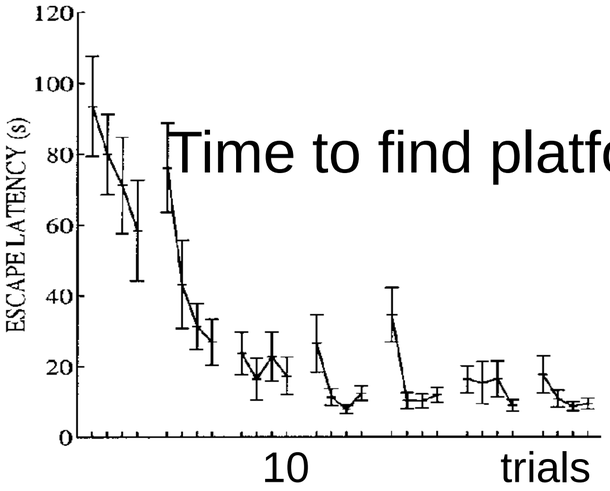
- Representation of states
- Representation of actions
- Representation of TD signal and learning rule

Review: TASK = conditioning in the Morris Water Maze

Morris Water Maze







Time to find platform

Rats learn to find
the hidden platform

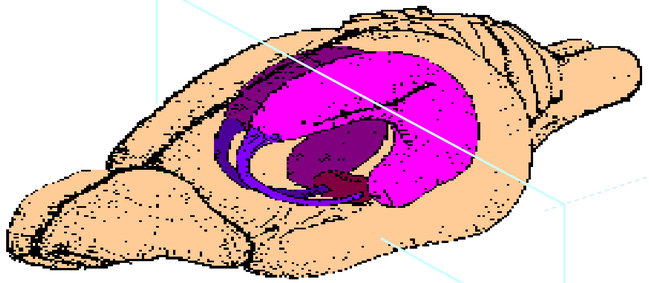
(Because they like to
get out of the cold water)

Foster, Morris, Dayan 2000

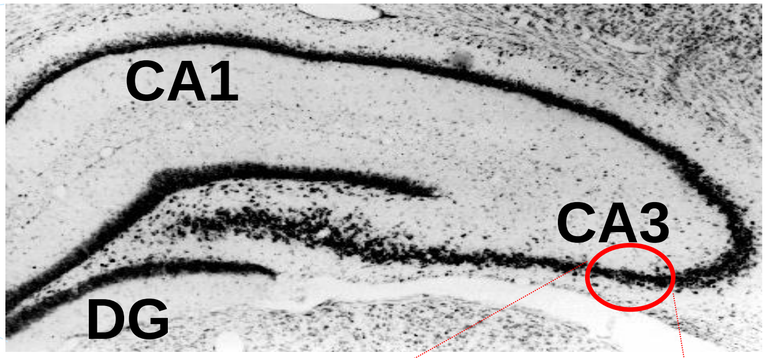
Previous slide.
Behvioral experiment in the Morris Water Maze.
The water is milky so that the platform is visible.

After a few trials the rat swims directly to the platform

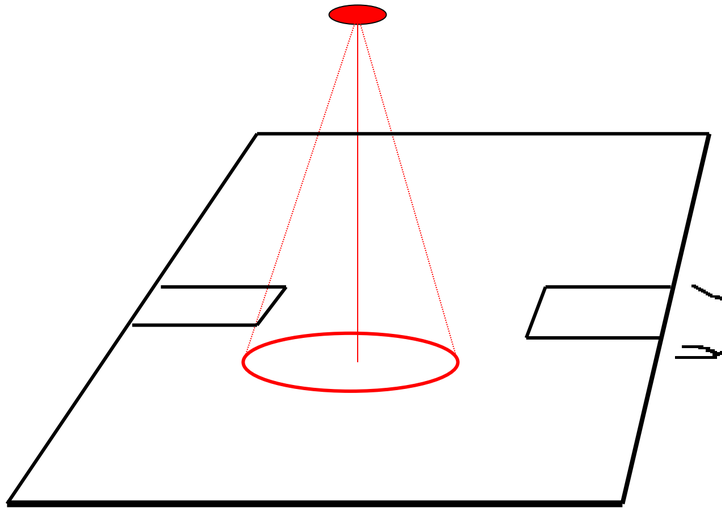
Recall: Representation of states: Place cells in rat hippocampus

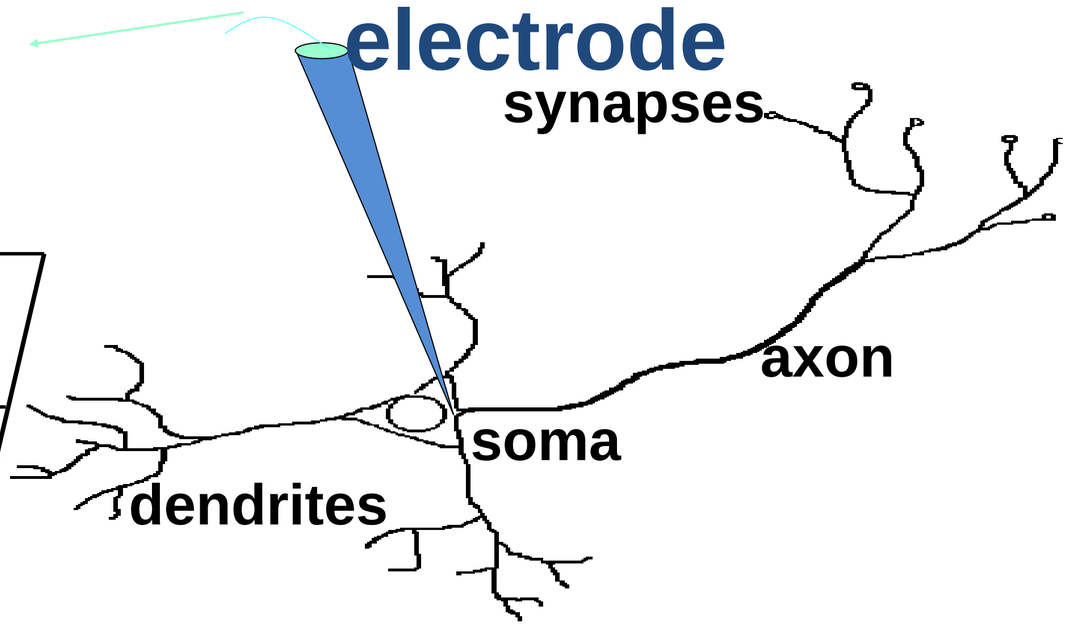


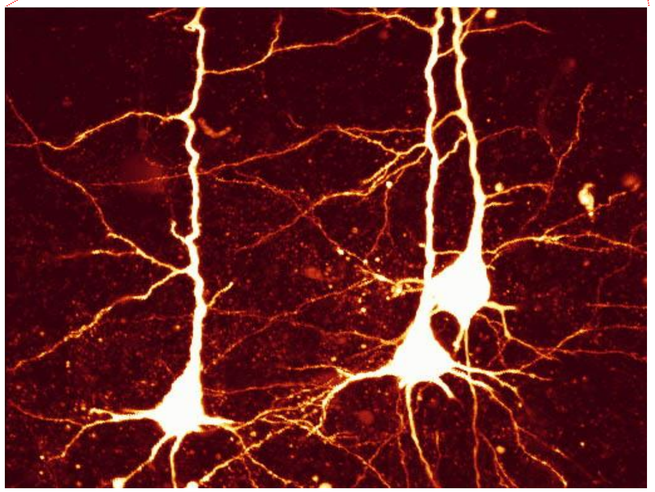
rat brain



Place fields







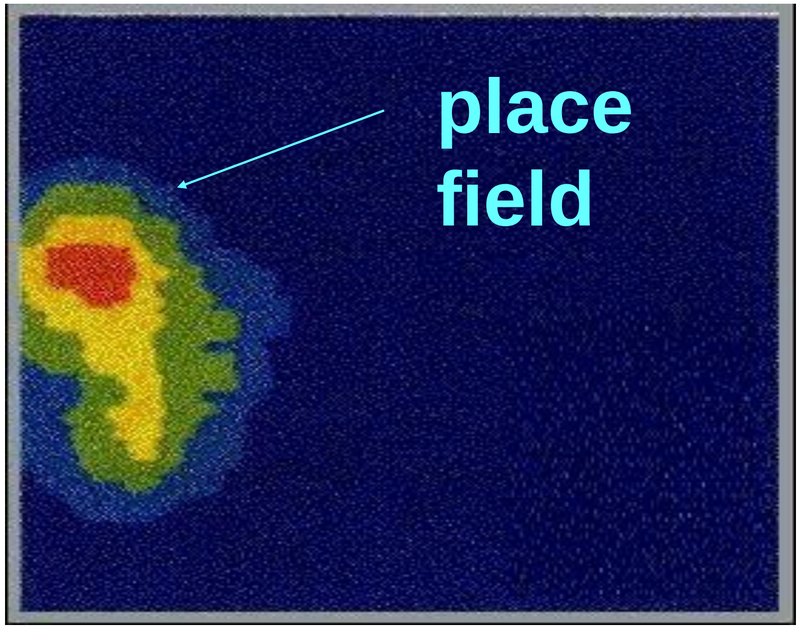
pyramidal cells

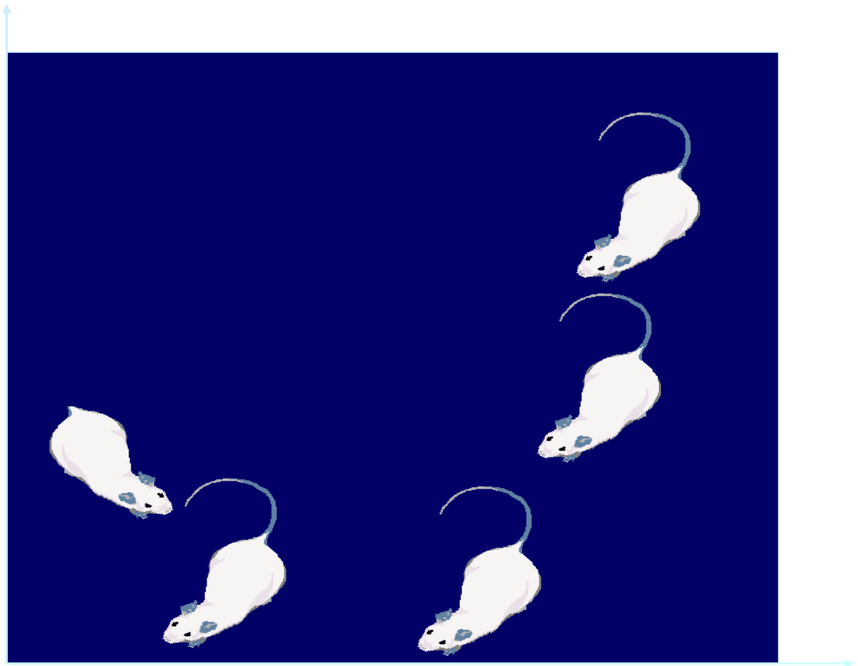
Previous slide.

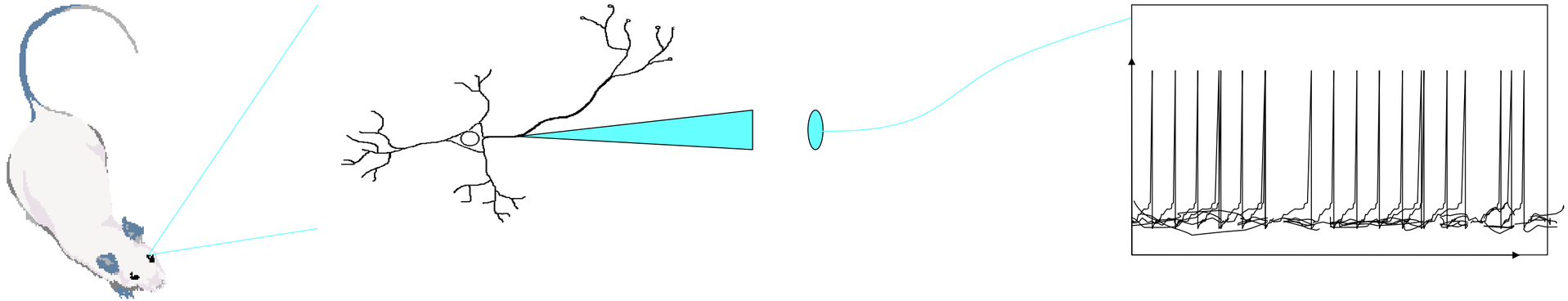
the hippocampus of rodents (rats or mice) looks somewhat different to that of humans. Importantly, cells in hippocampus of rodents respond only in a small region of the environment. For this reason they are called place cells. The small region is called the place field of the cell.

Recall: Representation of states: 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

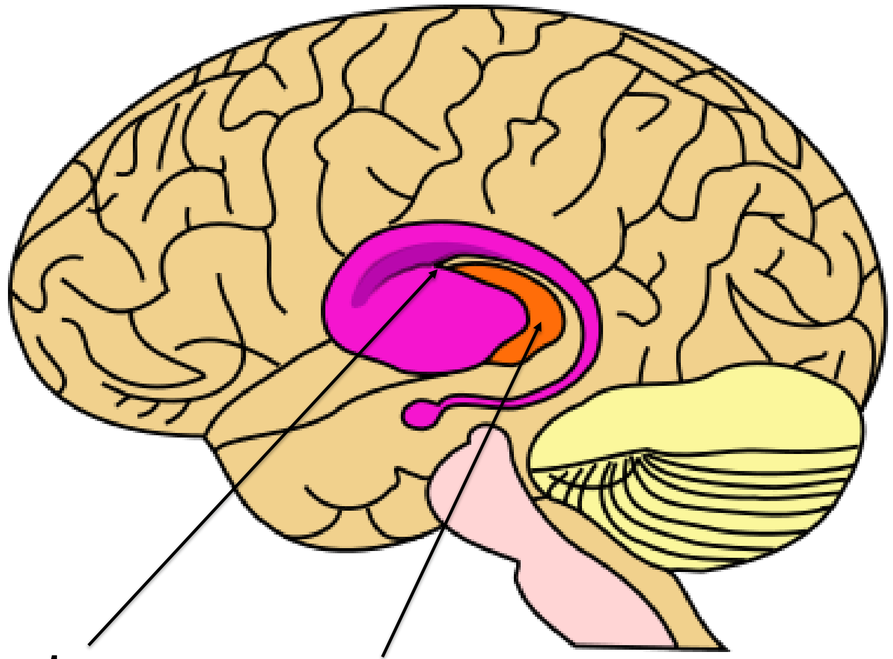
Recall: Striatum

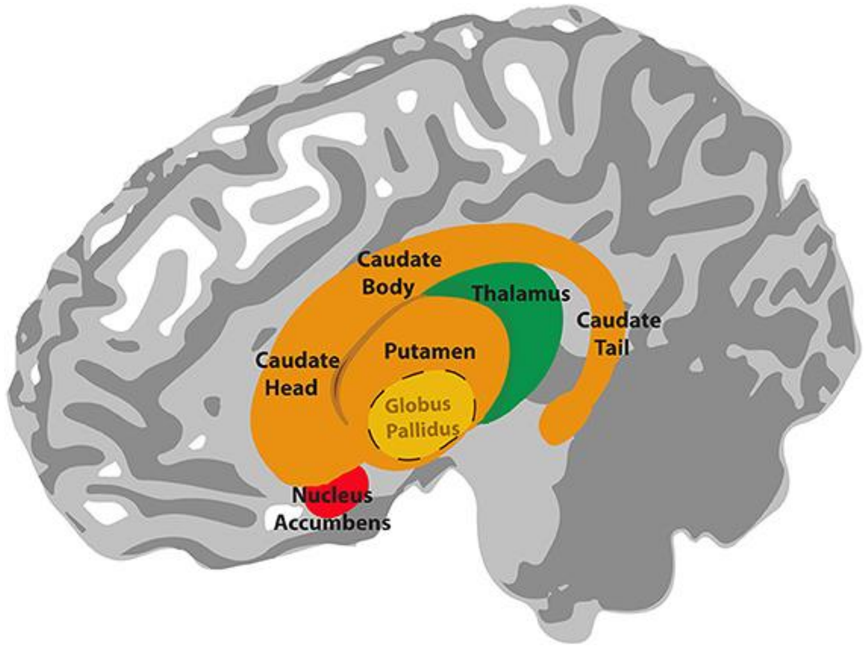
- Striatum sits below cortex
- Part of the ‘basal ganglia’
- **Dorsal striatum involved in action selection, decisions**

Striatum consists of

- Caudate (dorsal striatum)
- Putamen (dorsal striatum)

https://en.wikipedia.org/wiki/Striatum





Nucleus Accumbens is part of **ventral striatum**

fig: Wikipedia

Previous slide.
Left: Sketch of the Anatomical location of striatum and thalamus.
Right: the striatum lies also below the cortex. Since the striatum is involved in action selection it will play an important role in this lecture.
From Wikipedia:
The **striatum** is a nucleus (a cluster of neurons) in the subcortical basal ganglia of the forebrain. The striatum is a critical component of the motor and reward systems; receives glutamatergic and dopaminergic inputs from different sources; and serves as the primary input to the rest of the basal ganglia.
Functionally, the striatum coordinates multiple aspects of cognition, including both motor and action planning, decision-making, motivation, reinforcement, and reward perception.The striatum is made up of the caudate nucleus and the lentiform nucleus. The lentiform nucleus is made up of the larger putamen, and the smaller globus pallidus.
In primates, the striatum is divided into a **ventral striatum**, and a **dorsal striatum**, subdivisions that are based upon function and connections. The ventral striatum consists of the nucleus accumbens and the olfactory tubercle. The dorsal striatum consists of the caudate nucleus and the putamen. A white matter, nerve tract (the internal capsule) in the dorsal striatum separates the caudate nucleus and the putamen.[4] Anatomically, the term *striatum* describes its striped (striated) appearance of grey-and-white matter

Review: Coarse Brain Anatomy and R

Reinforcement learning needs:

- representation of states / sensory input / ‘where’
  → hippocampus? / sensory cortex?

- action selection → striatum?, motor cortex?

- reward signals → dopamine?

→ Candidate brain areas and brain signals!

Previous slide.

In reinforcement learning, the essential variables are the states (defined by sensory representation), a policy for action selection, the actions themselves, and the rewards given by the environment.

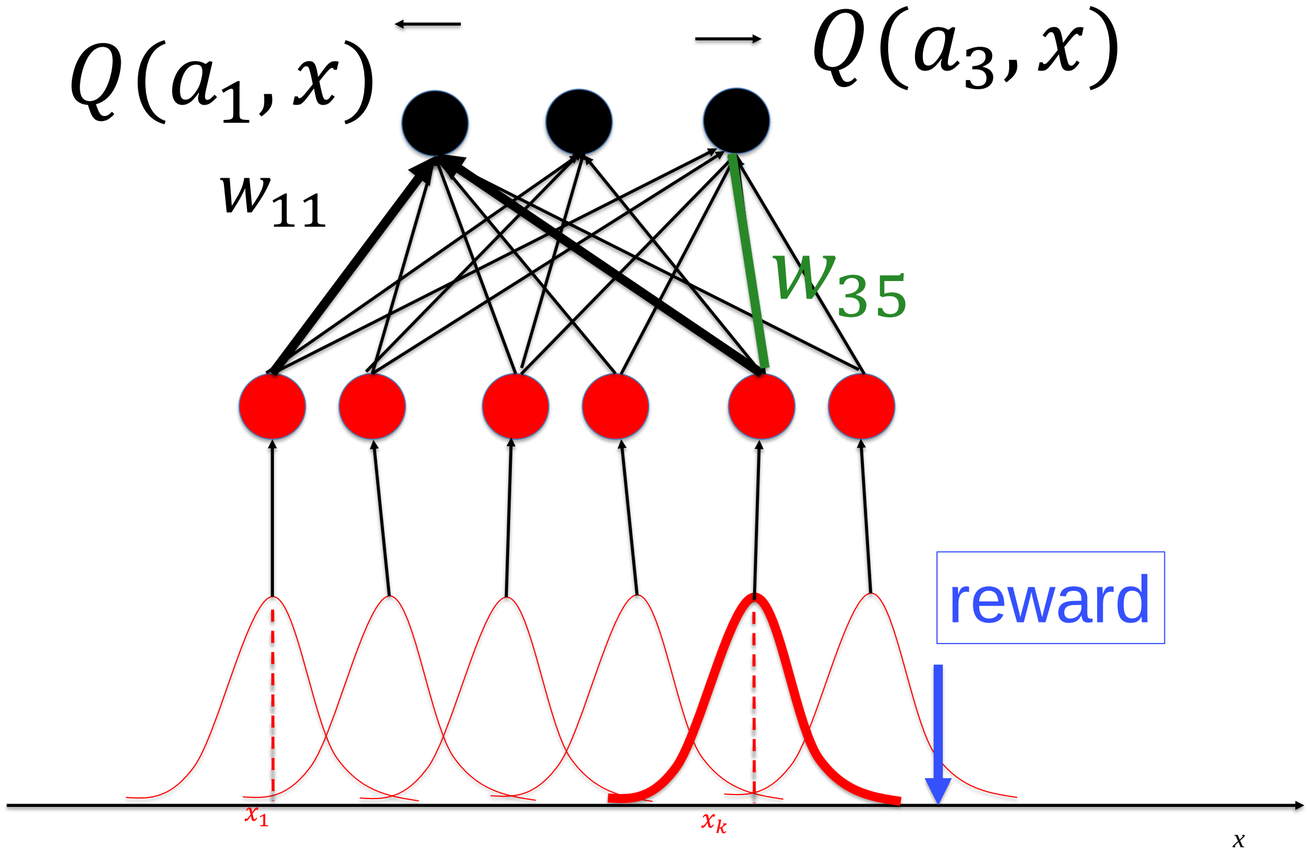
If we want to link reinforcement learning to the brain, we will have to search for corresponding substrates and functions in the brain.

The potential relations show candidate brain region for a mapping to state, actions, and reward. The above rough ideas need to be defined during the rest of this lecture.

Example: Linear activation model with softmax policy

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

$$\pi(a_j = 1 | x, \theta) = softmax[Q_1, Q_2, Q_3]$$
$$= softmax[\sum_k w_{jk}\, y_k]$$



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

Previous slide.

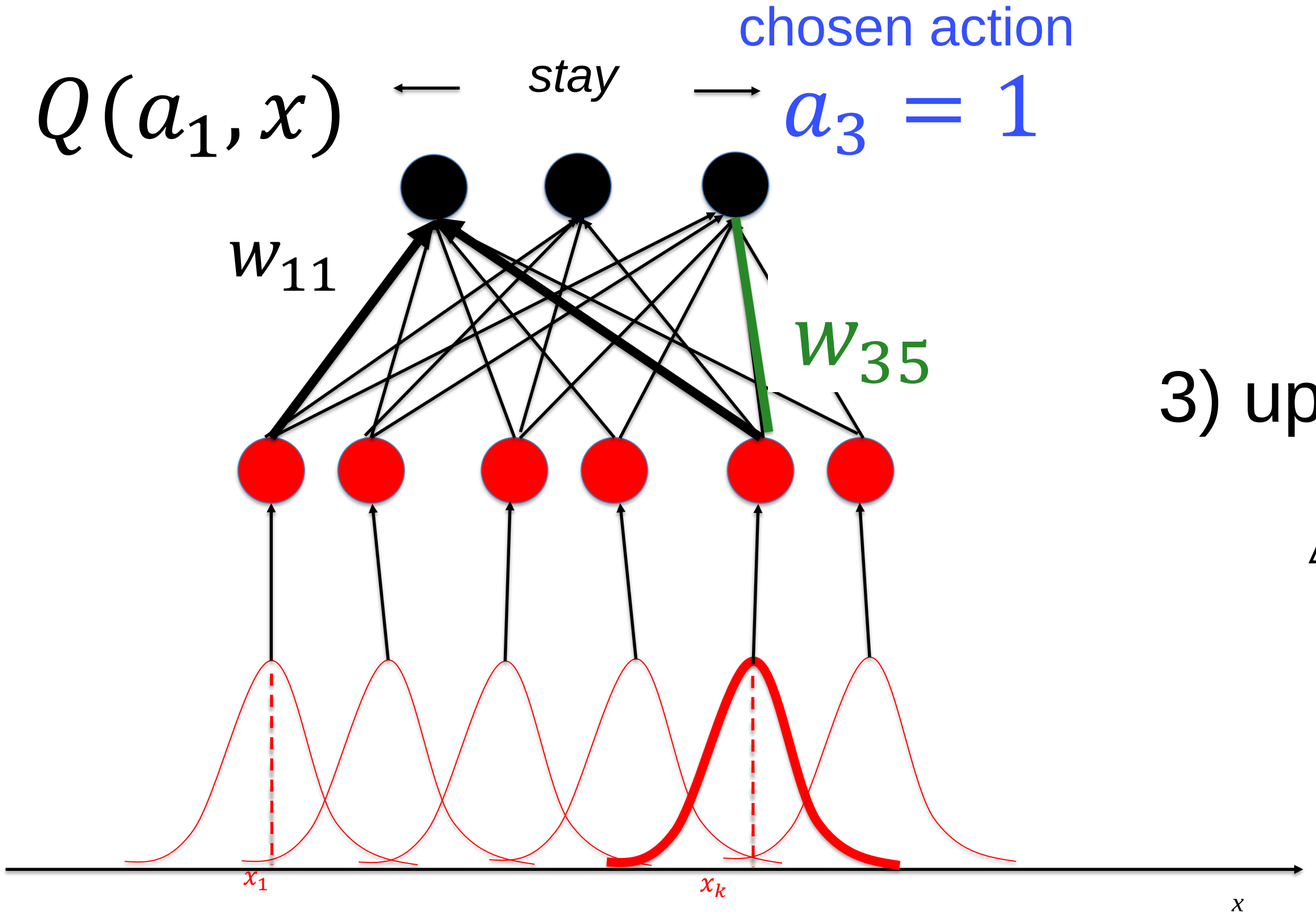
I now want to show that reinforcement learning with SARSA gives rise to three-factor learning rules.

Suppose the agent moves on a linear track.
There are three possible actions: left, right, or stay.
We interpret the action neurons as the Q-values!

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.

Review: Three-factor rules for SARSA(λ)

1) Choose action



2) Update eligibility trace (for each weight)

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

$$z_{ik} \leftarrow z_{ik} + pre \cdot post$$

chosen action=1
other action = 0

3) update weights

$$\Delta w_{lk} = \eta\ \mathrm{TD} \cdot z_{lk}$$

TD error is positive
if reward > expected reward

Previous slide.
Now we apply the update rule resulting from SARSA with eligibility traces.

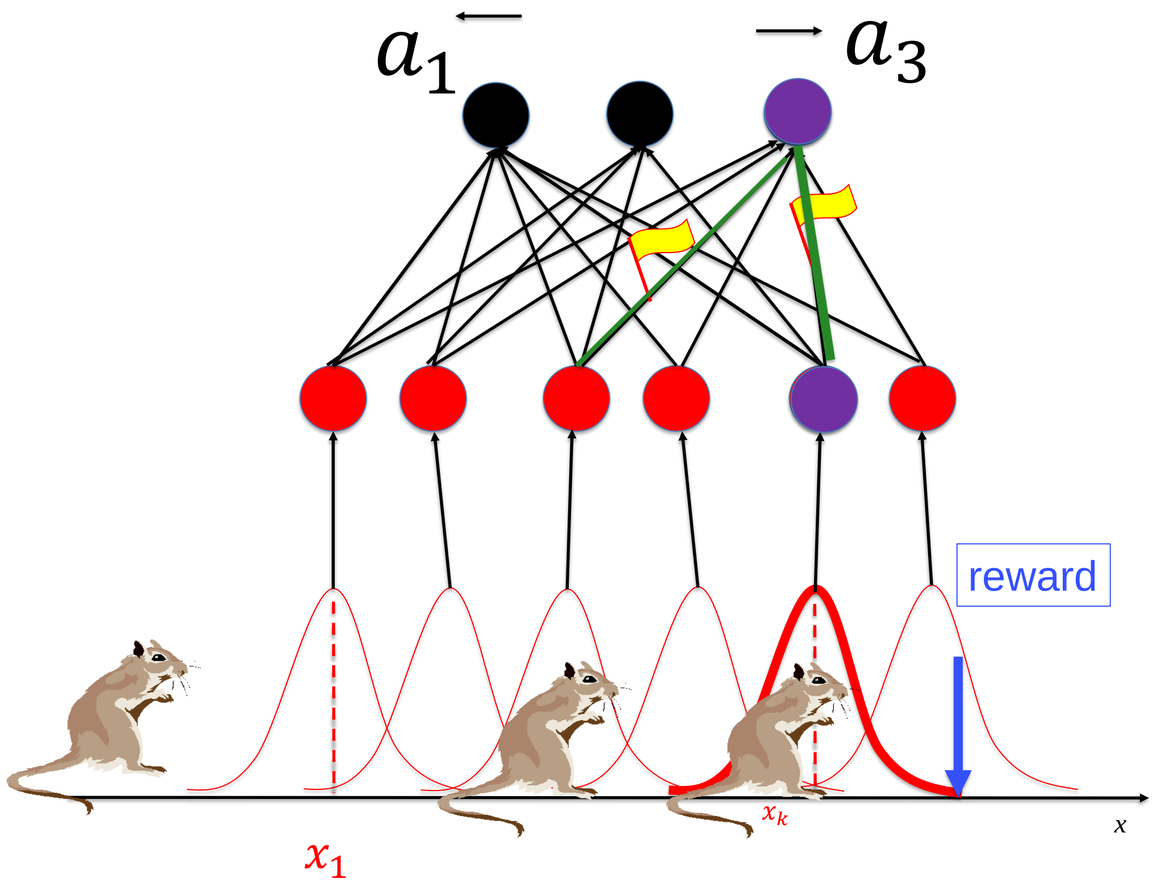
The calculation on the blackboard shows that the derivative of the loss function yields a term that can be interpreted as the multiplication of pre- and postsynaptic activity.

Without an eligibility trace, this multiplication would directly lead to a weight change. However, we want to use an eligibility trace. In this case, the term pre times post leads to an increase of the corresponding eligibility trace. The further multiplication with the TD error then causes the final weight update.

It is important to make the distinction between the Q-value (represented by the output neurons) and the binary action value (zero or one) that is actually chosen. The postsynaptic activity is the binary action value.

This requires some lateral competition between output neurons.

Review: Three-factor rules for SARSA(λ) - illustration



- place fields provide a compact encoding of the environment
- With a good encoding goal-oriented navigation is possible with a SINGLE layer.
- Hence three-factor rules can be used:

1) pre = representation of state
2) post = action

eligibility trace to bridge time scale of 1s

3) third factor = TD error

Previous slide.
Importantly, the update of the eligibility trace is a local learning rule that depends on a presynaptic factor and a postsynaptic factor. The update of the weight then depends on the eligibility trace and the TD error. The eligibility trace can be envisaged as a marker attached to the synapse (‘flag’). The TD error can be envisaged as being provided by a dopamine signal.

Problems in a biological interpretation could potentially arise from the facts that

- The action neurons are also representing Q-values – but in the end a SINGLE binary action is chosen
- This choice is implemented by the softmax policy – which implicitly would require a lateral interaction between neurons

→ A potential solution could be to implement winner-take-all dynamics but with a stochastic component at the moment of action choice.

An alternative solution can be by separating values (such as Q-values or V-values) from the action. This is the choice that we follow in the next week.

Summary: Three-factor rules, SARSA and eligibility

- Three-factor rules with eligibility traces enable to learn rapidly (in a few trials) despite delayed rewards
- Three-factor rules work with a single hidden layer (not compatible with BackProp)
- Three-factor rules therefore need a ‘good representation’ in the layer just before action choice

- Biological interpretation of SARSA with eligibility traces as three-factor rule is possible
- lateral inhibitory interactions necessary to impose a unique action choice
- Double role of action neurons as actors and Q-values is potentially problematic

Previous slide.
Summary

Reinforcement Learning Lecture 5

Policy Gradient and Actor-Critic 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. Math: Elibility traces arise naturally
6. **Application of Actor-Critic to navigation task**

Previous slide.

Previous section: SARSA applied to a navigation task

NEXT section: actor-critic applied to a navigation task.

We use the fact that place cells exist → good representation is available. Deep learning is not necessary!

Actor Critic with Eligibility Traces (continuing setting)

Actor–Critic with Eligibility Traces (continuing)

Input: a differentiable policy parameterization $\pi(a|s,\boldsymbol{\theta})$
Input: a differentiable state-value parameterization $\hat{v}(s,\mathbf{w})$
Parameters: trace-decay rates $\lambda^{\boldsymbol{\theta}} \in [0,1]$, $\lambda^{\mathbf{w}} \in [0,1]$; step sizes $\alpha^{\boldsymbol{\theta}} > 0$, $\alpha^{\mathbf{w}} > 0$, $\eta > 0$

$\mathbf{z}^{\boldsymbol{\theta}} \leftarrow \mathbf{0}$ (d'-component eligibility trace vector)
$\mathbf{z}^{\mathbf{w}} \leftarrow \mathbf{0}$ (d-component eligibility trace vector)

Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ and state-value weights $\mathbf{w} \in \mathbb{R}^{d}$ (e.g., to $\mathbf{0}$)
Initialize $S \in \mathcal{S}$ (e.g., to s_0)
Repeat forever:
$A \sim \pi(\cdot|S,\boldsymbol{\theta})$
Take action A, observe S', r
$\delta \leftarrow r + \hat{v}(S',\mathbf{w}) - \hat{v}(S,\mathbf{w})$ (if S' is terminal, then $\hat{v}(S',\mathbf{w}) \doteq 0$)
$R \leftarrow R + \eta\delta$
$\mathbf{z}^{\mathbf{w}} \leftarrow \lambda^{\mathbf{w}}\mathbf{z}^{\mathbf{w}} + \nabla_{\mathbf{w}}\hat{v}(S,\mathbf{w})$
$\mathbf{z}^{\boldsymbol{\theta}} \leftarrow \lambda^{\boldsymbol{\theta}}\mathbf{z}^{\boldsymbol{\theta}} + \nabla_{\boldsymbol{\theta}} \ln \pi(A|S,\boldsymbol{\theta})$
$\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}}\delta\mathbf{z}^{\mathbf{w}}$
$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}}\delta\mathbf{z}^{\boldsymbol{\theta}}$
$S \leftarrow S'$

TD error (for γ=1) but set γ=λ!

Update eligibility traces

Update parameters

Adapted from Sutton&Barton 2018

Notes:

1) Here Sutton and Barto have suppressed the factor γ. (by setting $\gamma=1$). They do not identify $\gamma=\lambda$.

2) The starting point of the derivation of Sutton and Barto is to optimize the average reward (I optimized average return with discount factor γ.

3) However, I would set $\gamma=\lambda$ and add the λ in the update for the TD error:

$$\delta \leftarrow \eta\,[\, r_t + \lambda\, V(S', w) - V(S, w)]$$

The reason is
(i) that the critic learns V-values with TD-learning and should use the same discounting $\gamma=\lambda$ as the actor.
(ii) The baseline subtraction for an actor that uses discounted cumulative rewards for returns should match the consistency condition of the Bellman equation.

Actor Critic with Eligibility Traces and three-factor rules

In a neural network that is NOT DEEP this algorithm is

- Online
- Causal
- 3-factor rule

→ biologically plausible

3-factor rule:

$\delta_{TD}\ pre \cdot post$

Repeat forever:

$A \sim \pi(\cdot|S,\boldsymbol{\theta})$

Take action A, observe S', r

$\delta \leftarrow r + \gamma\ \hat{v}(S',\mathbf{w}) - \hat{v}(S,\mathbf{w})$ ← TD error γ=λ!

$R \leftarrow R + \eta\delta$

$\mathbf{z}^{\mathbf{w}} \leftarrow \lambda^{\mathbf{w}}\mathbf{z}^{\mathbf{w}} + \nabla_{\mathbf{w}}\hat{v}(S,\mathbf{w})$

$\mathbf{z}^{\boldsymbol{\theta}} \leftarrow \lambda^{\boldsymbol{\theta}}\mathbf{z}^{\boldsymbol{\theta}} + \nabla_{\boldsymbol{\theta}} \ln \pi(A|S,\boldsymbol{\theta})$ } Update eligibility traces

$\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}}\delta\mathbf{z}^{\mathbf{w}}$

$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}}\delta\mathbf{z}^{\boldsymbol{\theta}}$ } Update parameters

$S \leftarrow S'$

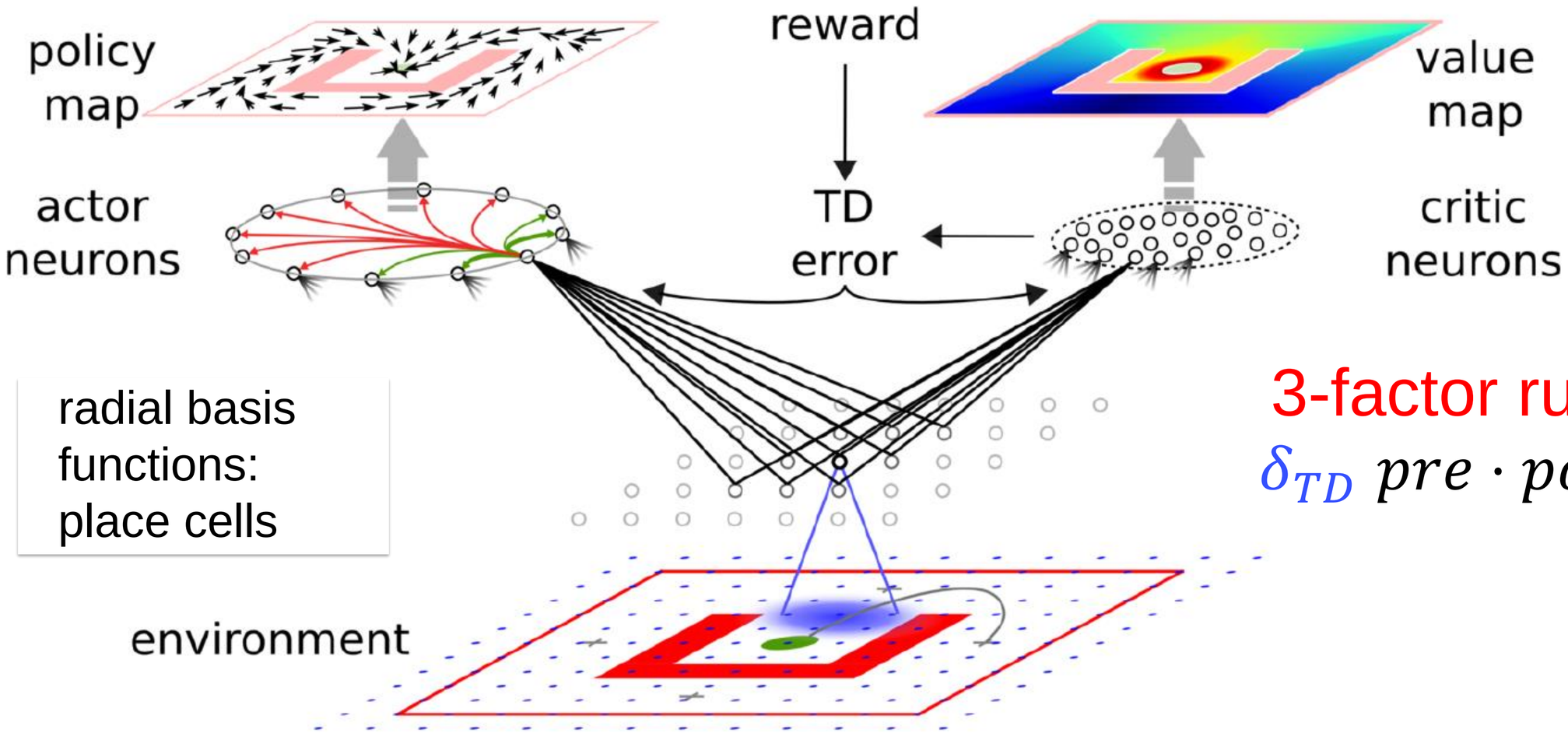
Adapted from Sutton&Barton 2018

Previous slide.

Not deep means:
Not a deep network:
We have a good representation one layer away from the output, and the output (final layer) is the action layer (or value layer)

Application to a navigation task (see computer exercise)

6. Maze Navigation with TD in Actor-Critic



3-factor rule:

$\delta_{TD}\ pre \cdot post$

Fremaux et al. (2013)

The environment contains an obstacle (red) and a invisible reward location (green).
The overall structure of the neural network is that of an actor-critic architecture.
Both actor and critic are driven by radial basis functions that represent the environment.

The actor neurons are organized in a ring. Neighboring neurons share information (details not necessary to know at this stage, see next slide).
Critic neurons are drawn here as a population of neurons, but this could be just as well a SINGLE critic neuron.

Critic neurons learn to represent the value of the current state using TD learning derived from the representation of V-values. The same TD signal is also used the update the connections to the actor neurons.

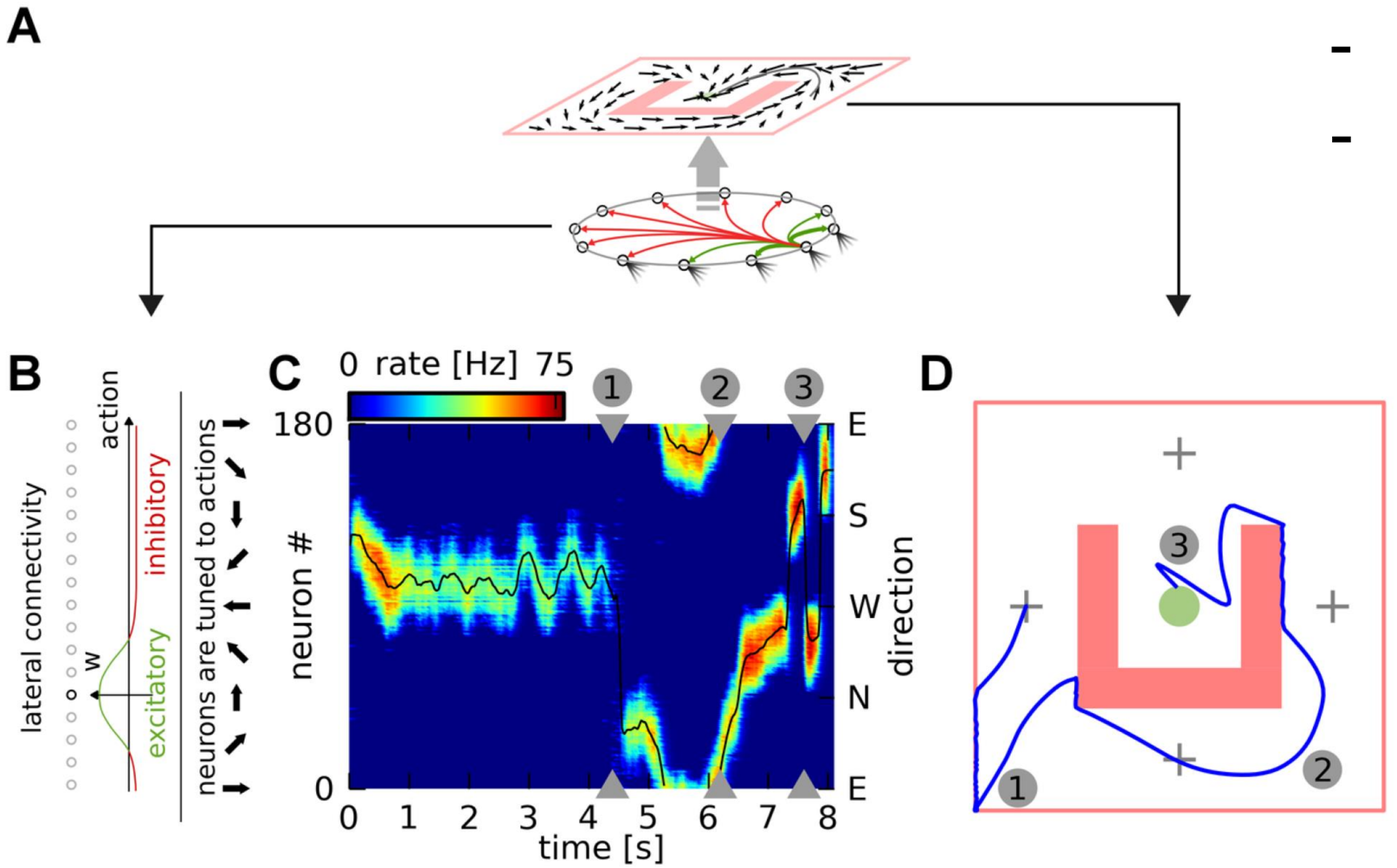
We do not need a deep network to learn the task, since radial basis functions (place cells) provide a compact representation of the environment. The actor-critic learning is then equivalent to a three-factor rule.

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

  ‘soft competition’ of action neurons



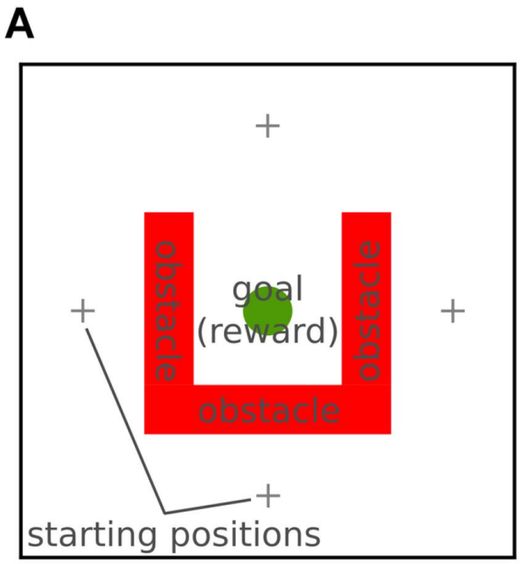
Fremaux et al. (2013)

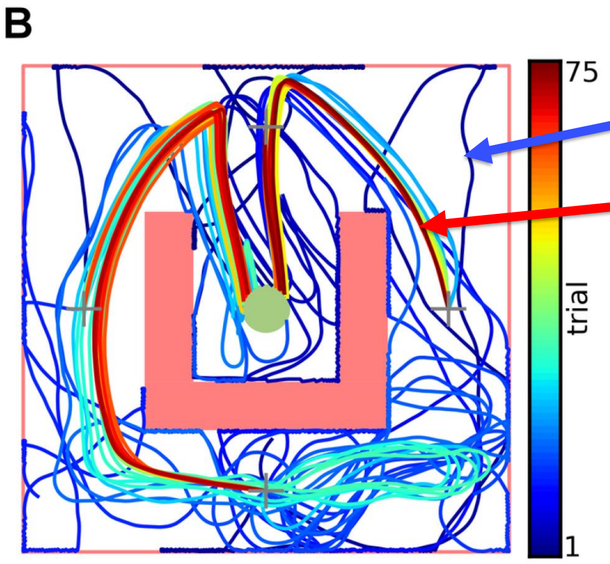
Ring of actor neurons (similar to soft competitive learning)

Actor neurons (previous slide).
A: A ring of actor neurons with lateral connectivity (bottom, green: excitatory, red: inhibitory) embodies the agent's policy (top). Each neuron represents one of the 360 possible directions.
B: Lateral connectivity. Each neuron codes for a distinct motion direction. Neurons form excitatory connections to similarly tuned neurons and inhibitory synapses to other neurons. As a result, neighboring neurons learn from each other of how to represent actions.
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

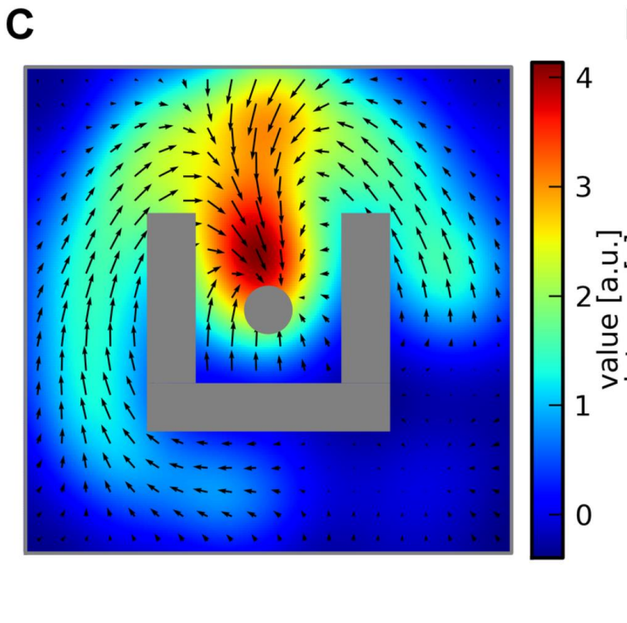


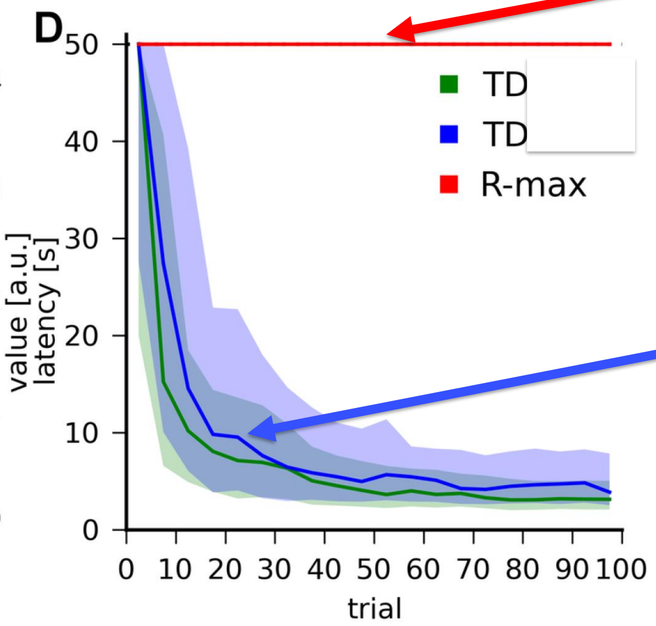


early trial

Late trial

value map





R-max: (REINFORCE)

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

TD: Actor-Critic

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

6. 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 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 2000 simulated seconds.
D: Goal reaching latency of agents using three slightly different learning rules for the actors . 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 (REINFORCE without baseline) agent was simulated without a critic and enters times-out after 50 seconds.

Fremaux et al. (2013)

6. TD in Actor-Critic for navigation task

- Learns in a few trials (assuming good representation)
- Works in continuous time (thanks to eligibility traces).
- Works in continuous space and for continuous actions
- Critic implements value function
- TD signal calculated by critic
- Three-factor rules for learning both actor and critic
- TD signal used for both actor and critic weights

Fremaux et al. (2013)

Previous slide.
Summary of findings.

An additional trick is that actor neurons interact with each other so that neighboring neurons share information and learn ‘similar’ policies.
This was not shown in class.

Actor-critic with TD is better than standard policy gradient, because over many trials information about state-values can diffuse back from the reward location into distant parts of the environment. Note that the discount factor γ of the value function can have a different value than that of the eligibility trace of the actor.

Standard policy gradient cannot learn beyond the forgetting scale of the eligibility trace (here 1 second).

6. Summary

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

Actor-Critic works much better than REINFORCE without baseline.

Actor-Critic with Eligibility Traces is a powerful and stable algorithm and highly recommended.

Probably the best model-free algorithm.

Previous slide.

In summary, we get fast reinforcement learning with biologically plausible components (3-factor rule with TD learning/dopamine).

However, since we cannot do backprop in biology, these results rely on the fact that the network has a ‘good’ representation of states, just below the layer of actions selection.

Place cells are such a good representation.

But how do we get place cells from visual input?

Reinforcement Learning Lecture 3

Wulfram Gerstner
EPFL, Lausanne, Switzerland

Continuous input space: representation of 'states'

Unsupervised learning of a 'good' state representation

Arleo and Gerstner (2000), ***Spatial cognition and neuro-mimetic navigation: A model of hippocampal place cell activity.*** *Biol. Cybern. 83:287–299.*

Strösslin et al. (2005), ***Robust self-localisation and navigation based on hippocampal place cells.*** *Neural Networks 18:1125–1140* doi:10.1016/j.neunet.2005.08.012

Sheynikhovich et al. (2009), Is There a Geometric Module for Spatial Orientation? Insights From a Rodent Navigation Model, Psychol. Review 116:540

Previous slide.

Using a specific example we want to illustrate why function approximation yields an inductive bias for generalization.

Inductive bias in Reinforcement Learning

Before you code an RL problem, try to answer the following questions:

1) Is the problem such that in similar (neighboring) input states the best action is (likely to be) the same?

2) Is the problem such that if I find the reward with action a^* from state s, then a^* is probably good in other states as well?

3) Do I expect rewards in many states or rather only in a few 'goal states'?

4) Moreover, are rewards given for states or state-action transitions?

5) Is there a topology/neighborhood relation that would enable us to talk about two actions as being 'similar'?

Previous slide.

If you know the answer to one of the questions you can use this knowledge to choose your coding scheme for inputs and for the action space.

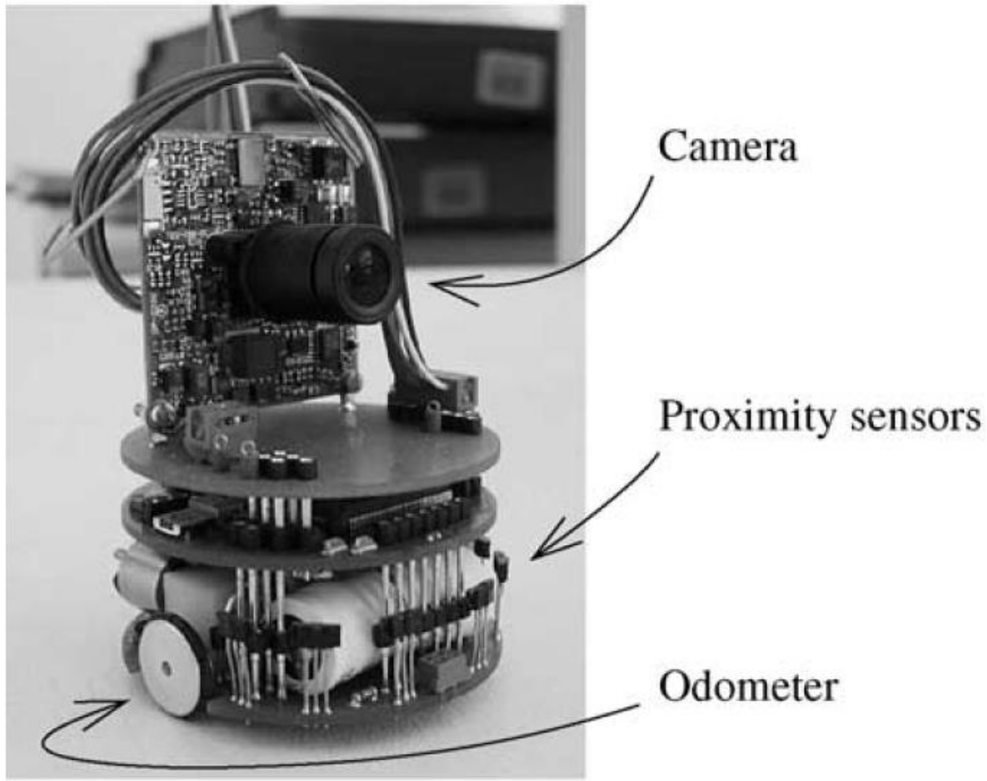
Inductive bias in Reinforcement Learning (Example 2): Self-localization and Navigation to Goal

- 2-dimensional arena 80cmx60cm
- single goal location: reach goal from arbitrary start location
- 120 actions (=directions of movement)

Agent:
Khepera Robot

Camera:
240^0 view
>240 000 pixel

Preprocessing:
Gabor filter bank



Strösslin et al. (2005), ***Robust self-localisation and navigation based on hippocampal place cells.*** *Neural Networks 18:1125–1140;* doi:10.1016/j.neunet.2005.08.012

Previous slide.

The camera of the Khepera robot makes snaptshots in 4 directions that are combined into a single ‘view’ covering a viewing field of 240 degree (total would be 360 degree). This corresponds to > 240 000 pixels per view.

The image serves as input that represents the present ‘state’.

The robot moves in a square (or circular) arena.

The task is to find an goal location (not marked!), from any possible start configuration.

Self-localization and Navigation to Goal

- 2-dimensional arena 80cmx60cm
- Task: reach goal location (5cmx5cm) from arbitrary start
- 120 actions (=directions of movement)
- state = camera input = >240’000 pixels

How many episodes do we need to train the agent to solve the task?

[ ] <20

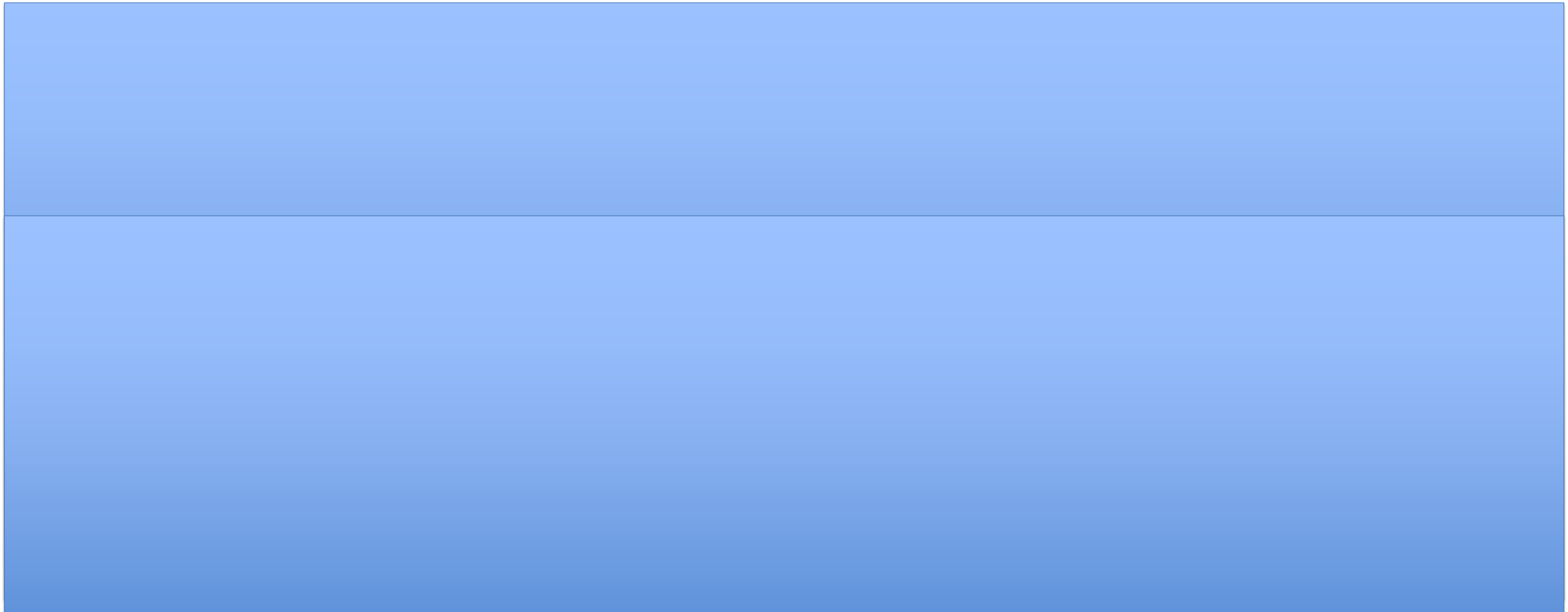
[ ] 20 – 200

[ ] 200 – 1000

[ ] > 1000

Self-localization and Navigation to Goal, starting from visual input

We start with images of > 240’000 pixels, but we may consider

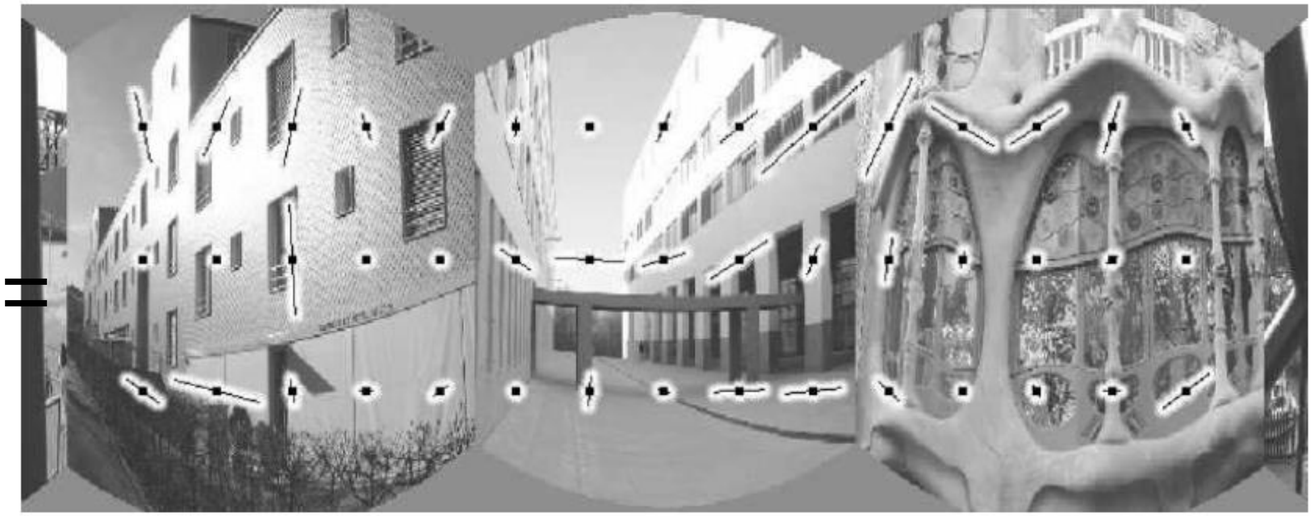


Inductive bias in Reinforcement Learning (Example 2)

- **Preprocessing Gabor filter bank:**

  Filters of several spatial frequency and orientation
  at 45 different locations,
  200 filters per location.

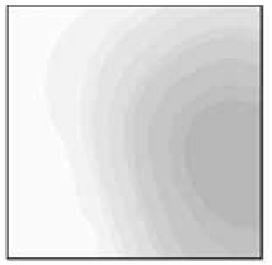
- **Snap-shot of environment =**



  store the vector F_j
  of 9000 filter responses

- **'Basis-function'** $\phi\left(F(t) - F_j\right)$

  similarity of current view $F(t)$ with stored view vector F_j
  after rotation to optimal matching angle



sample basis function

Strösslin et al. (2005), Neural Networks 18:1125–1140; doi:10.1016/j.neunet.2005.08.012

Previous slide.

The camera of the Khepera robot makes snaptshots in 4 directions that are combined into a single ‘view’ covering a viewing field of 240 degree (total would be 360 degree). This corresponds to > 240 000 pixels per view.

The sample image (caption below) shows the mean orientation of all Gabor filters with the lowest spatial frequency at the 45 sampling locations.

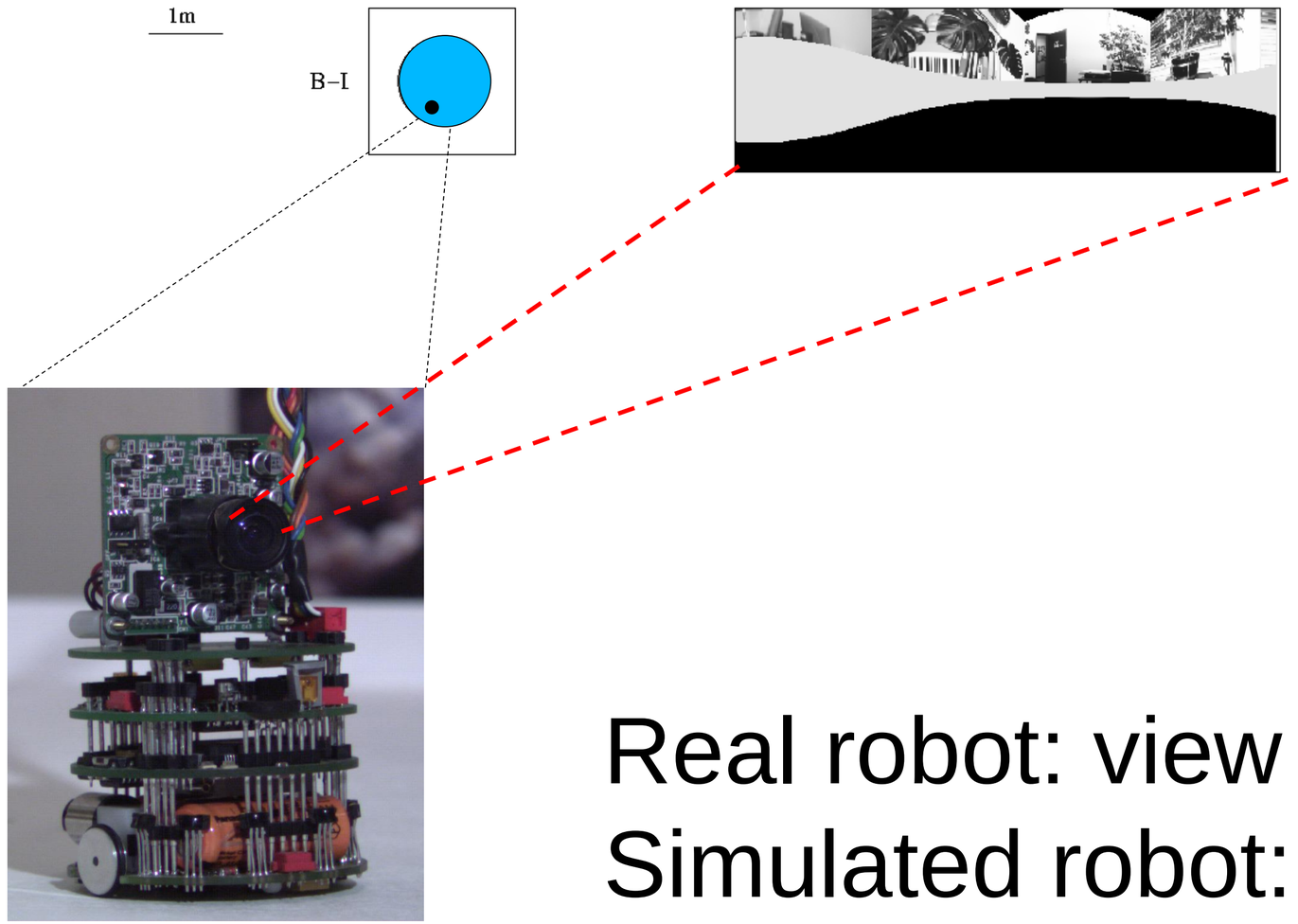
The Gabor filters come as pairs of sine and cosine filters (or complex filters) and only the total amplitude, but not the phase of the response of the filter pair is recorded.

The set of filter responses at time t of all 9000 filters is denoted by $F(t)$

Details of the processing steps are explained in the next few slides

Fig. 3. Response of the artificial retina applied to a view from the ‘Buildings’ environment. The black lines represent the weighted sum of the gabor filter responses for all orientations. They indicate the direction and ‘strength’ (line length) of edges near each retinal point.

Self-localization and Navigation to Goal: Details of visual processing and Extraction of Basis Function



Real robot: view field 4X60°

Simulated robot: view field 280°

Strösslin et al. (2005), Neural Networks 18:1125–1140; doi:10.1016/j.neunet.2005.08.012

Previous slide.

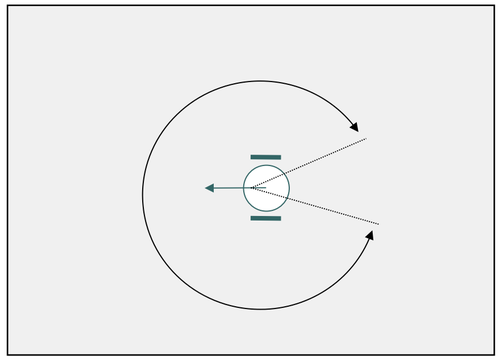
The robot takes a sample image.

With a real robot: we let the robot rotate around its own axis to take views in 4 directions, each view over 60 degree angle; the four views are considered as a single image of view angle 240 degrees.

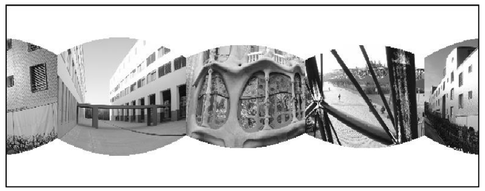
In simulated robots one case use directly 280 or 300 or even 360 degree as a viewing angle.

Model: stores views of visited places

Robot in an environment

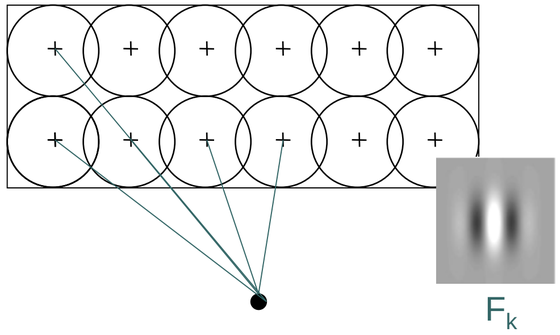


Visual input at each time step



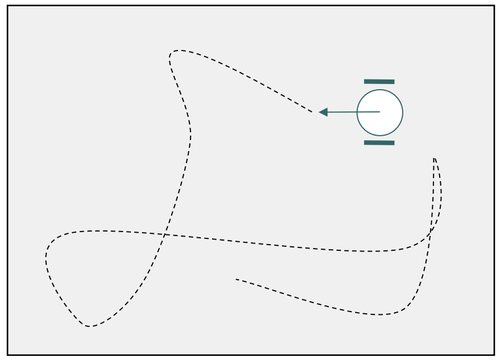
Local view : activation of set of 9000 Gabor wavelets

$$L(\vec{p},\Phi)=\{F_k\}$$

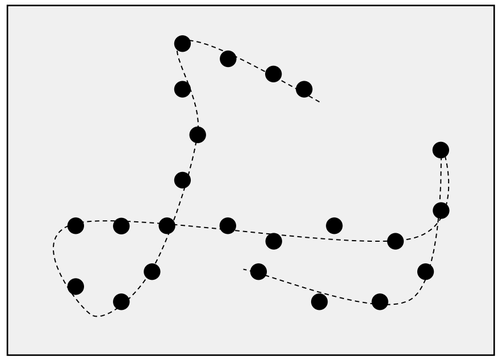


Single View Cell stores a local view

Environment exploration



Population of view cells



All local views are stored in an incrementally growing view cell population

Previous slide.

During exploration the robot takes a new sample image whenever it does not recognize the view. Recognition is defined that 10 or more cells strongly respond to the new image.

If not, the image is classified as novel. Novelty triggers learning (as a third factor!).

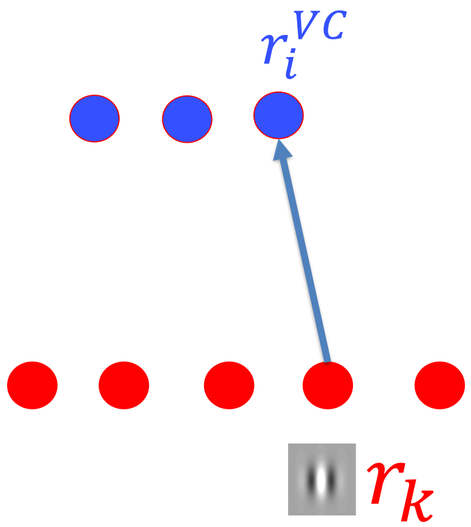
The novel sample image is memorized by storing the set of responses of the 9000 Gabor filters.

Model 'stores' views of visited places: 3-factor rule

synapses from Gabor filter k (Receptive Fields) to 'view cell' j

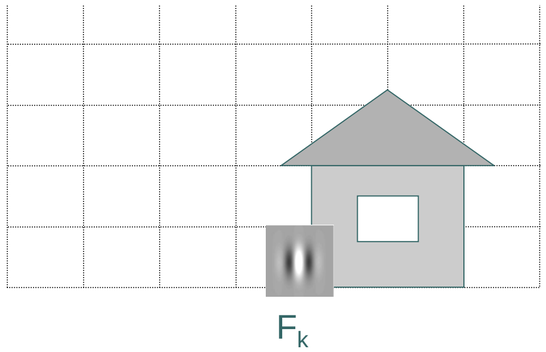
$$\Delta w_{ik} = \eta r_i^{VC} r_k \quad if \quad r_k > \theta_1 \quad \text{(else =0)}$$

$$\eta = 1 \; if \; 'novel \; view'$$



$$r_i^{VC} = 1 \; for \; newly \; recruited \; cell$$

$$r_k = \left| \int F_k(x') Image(x') dx' \right|^2$$



Strösslin et al. (2005), Neural Networks 18:1125–1140; doi:10.1016/j.neunet.2005.08.012

Previous slide.

What does it mean to ‘store the present view’?

The model uses a three-factor rule that yields ‘1-shot’ Hebbian learning.

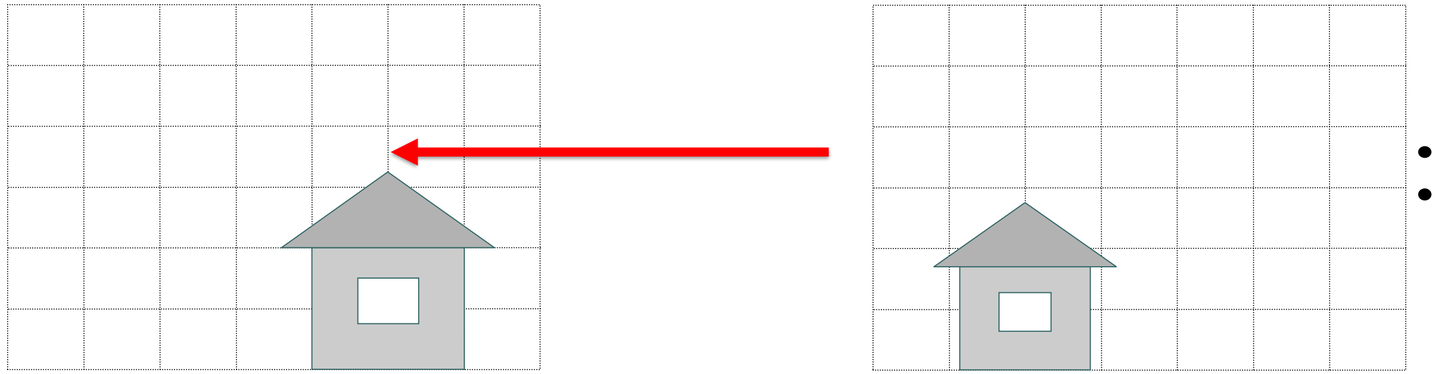
1. The image is applied to the set of thousands of filters. We think of filters as receptive fields. For each filter a presynaptic neuron is active proportional to the response of its filter.
2. We compare the total response of summed over all presynaptic neurons with a threshold. If the total sum is smaller than a novelty threshold, then the present view is novel.
3. If novelty is discovered all postsynaptic cells are turned off, except a single cell j whose activity is set to $r_j = 1$.
4. We apply Hebbian learning (Eq. 1) to this cell with learning rate $\eta = 1$

The result yields learning in a single trial (one-shot), controlled by novelty (third factor).

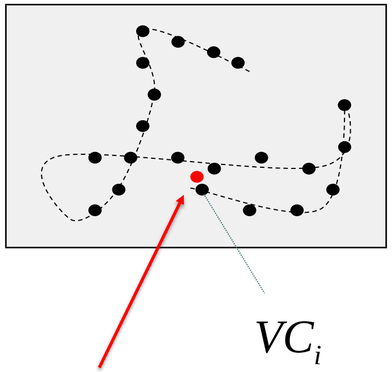
Model: extracting gaze orientation

Stored local view i

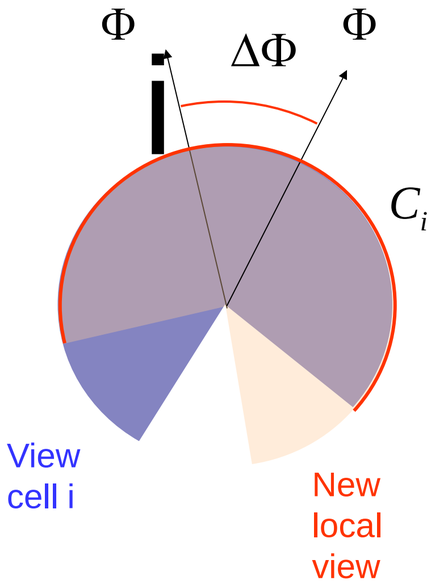
New local view



Population of view cells



position at new local view



Alignment of views → current gaze direction

Previous slide.

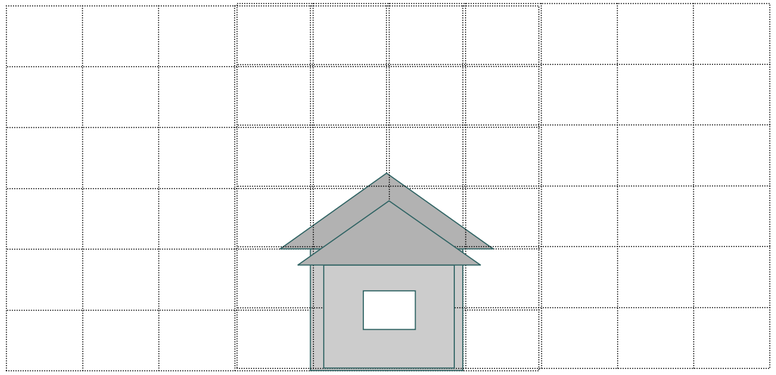
The filter responses at time t are compared to the stored filter responses.
To find the best match the new image is rotated.

The angle of rotation (necessary to yield the best match) tells us about the direction of gaze compared to the gaze direction at the moment when the original image was stored.

Model: extracting position via basis functions

Stored local view i

New local view



Small difference between local views – spatially close positions

stored/current filter responses

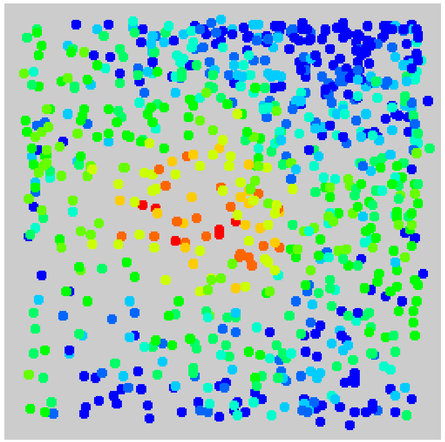
↓ ↓

Difference: $\Delta L_i = |F_i - F(t)|$

Similarity measure: $r_i^{VC} = \exp(-\frac{|\Delta L_i|^2}{\sigma_{VC}^2})$

Basis Function $\phi(F(t) - F_i) = r_i^{VC}$

population of view cells responding at red position



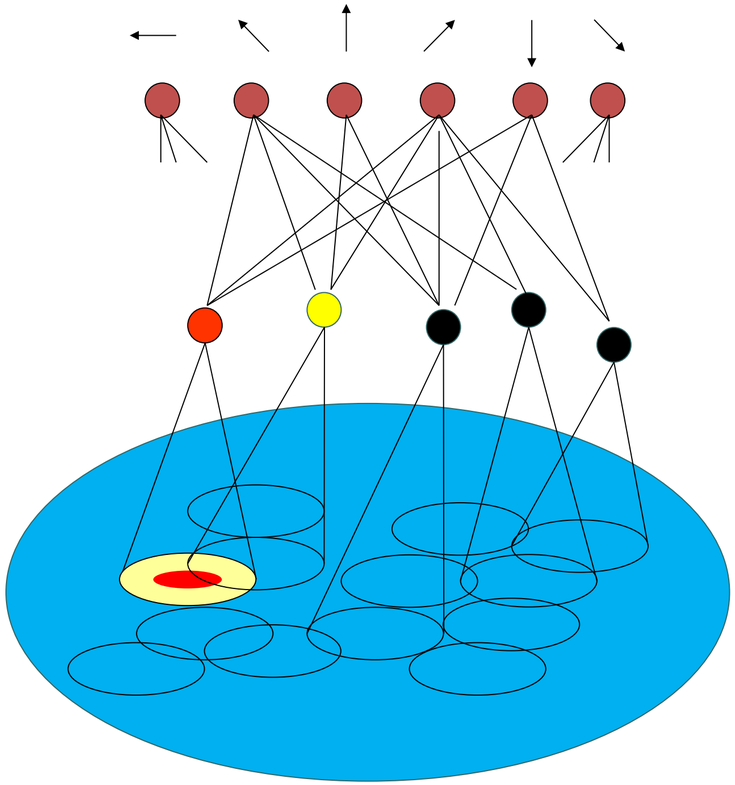
Previous slide.

After the rotation to best-match position. the filter responses at time t are compared to the stored filter responses. This yields the basis function.

The image on the right shows which basis functions respond when the robot is at a specific location. Red indicates strong response.

Note that basis functions do not know where they are located in space (i.e. they have no spatial position label, but just their response profile and an index for each basis function). For this image we have plotted a dot at a place that corresponds to the location of the maximal response of a given basis function. But this is **for visualization purpose only.**

Summary: From Pixel input to Basis function to Q-values



$$Q(s, a'|\boldsymbol{w}) = \sum_j w_j(a')\phi\big(F(t) - F_j\big)$$

a'

Activity = Q Value

$w_j(a')$

$\phi\big(F(t) - F_j\big)$

f_j

state s(t)

preprocesing: Gabor filters + rotation/alignment

Previous slide.

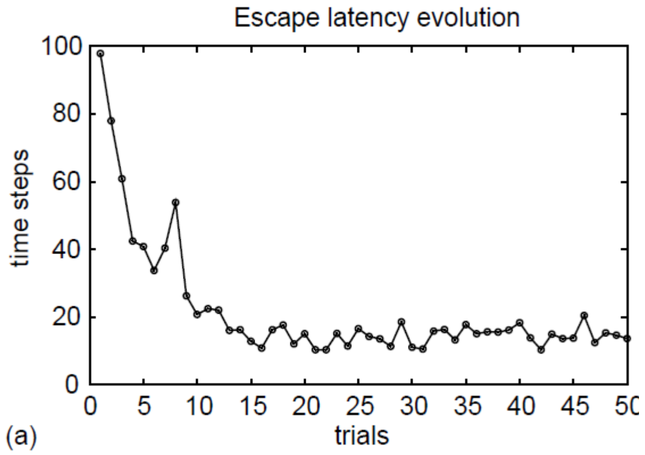
Action neurons represent the Q-values. In total there are 120 neurons. We may consider them to lie on a circle with a position on the circlue corresponding to the direction of movement triggered by the action.

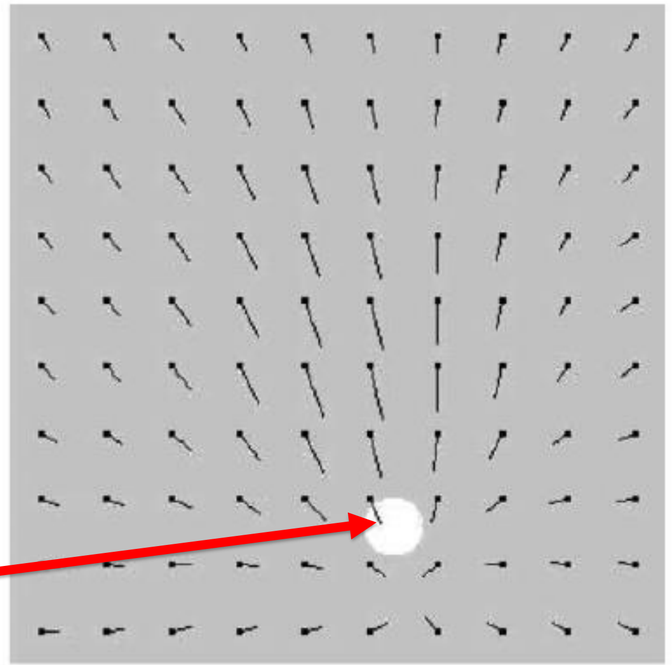
The center f_j of each basis function j corresponds to the (stored) response of thousands of Gabor filters recorded at some time t_j during exploration. The output of the basis function j measures the similarity with the current view, represent by the current response of the Gaborfilters, The vector of all Gabor filter responses at time t is f(t).

The figure on the left shows rather schematically the net result of the processing steps. The functions ϕ are visualized as local basis functions in the environment. Weights connect to actions that code for the different movement directions. The activiation of each action unit indicates its Q-value.

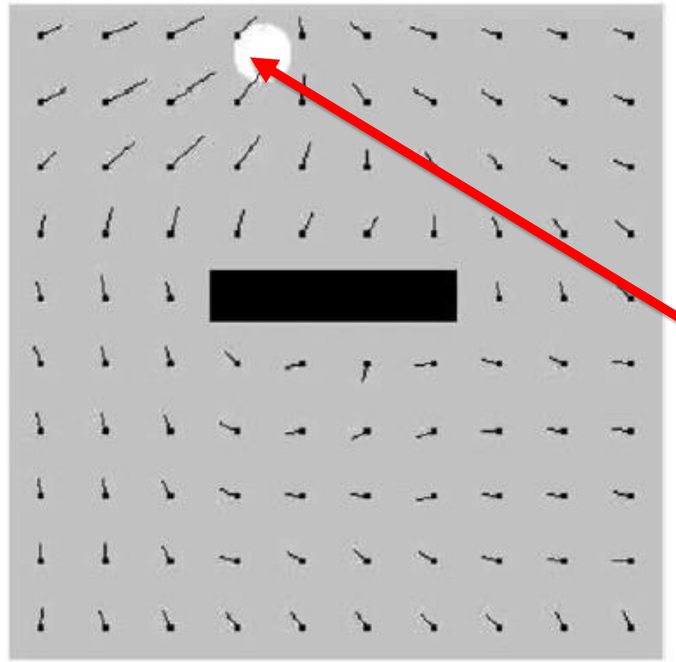
Self-localization and Navigation to Goal:

- While exploring: take new snapshot whenever less than 10 basis functions are active → creates new basis function
- Reinforcement Learning by Q-learning/SARSA
- Final action directions after 20 episodes (goal-findings)





goal



goal

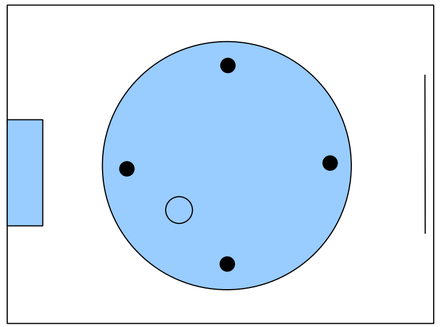
Previous slide.

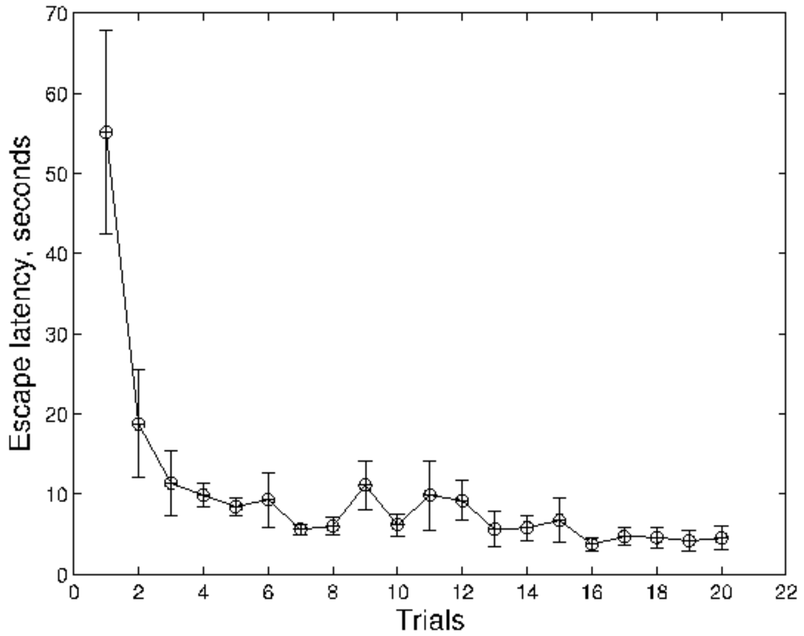
The left image shows the time it takes to find the goal, as a function of successful trials (episodes).

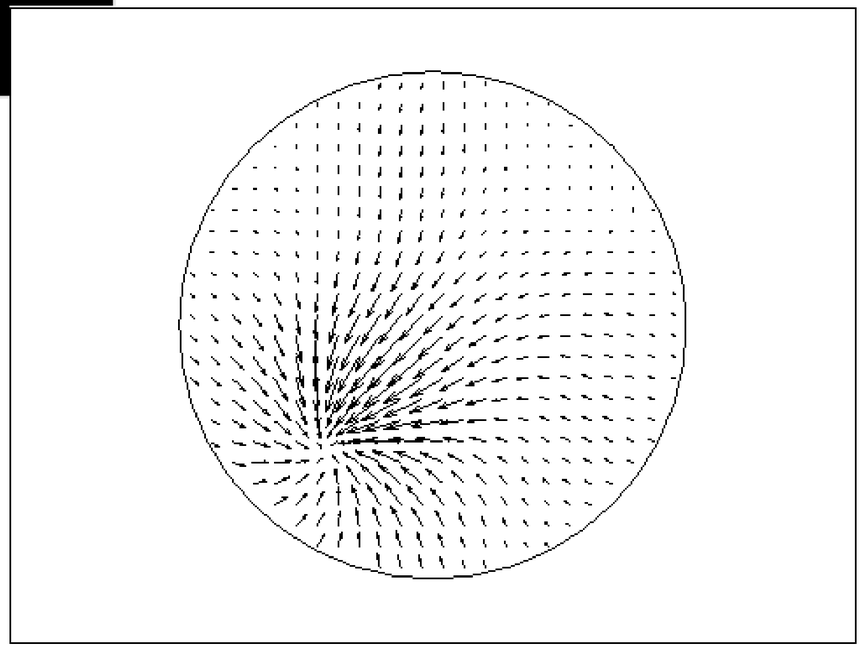
The right image shows needles that indicate the learned direction of movement after 20 trials.

Navigation Results: Office environment

Sheynikhovich et al. Psychological Review, 2009







- Map after 10 trials

- Learning = relate present view (location) to movement direction
- Needs alignment of the views to known orientation

Strösslin et al. (2005), Neural Networks 18:1125–1140; doi:10.1016/j.neunet.2005.08.012

Previous slides.

- Coding of input space: we sample vectors of feature responses in the high-dimensional space, but we know that in the end they encode only two dimensions, so that sampling is indeed possible.

- The space is further reduced from 3 to 2 dimensions by algorithmic rotation of images (= shift of feature vectors) to get rid of difference due to orientation.

- We can work with relatively long eligibility traces, since there is a single goal state.

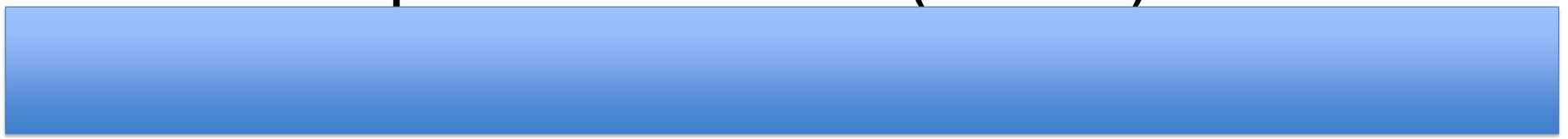
- We generalize across actions: we imagine actions forming a ring of possible directions. Neighboring actions should learn (in most states) similar behavior, hence if action a* is chosen with SARSA and learns (at rate eta), then all its neighbors learn as well (but with slightly reduced rate).

Reinforcement learning can be fast!

- input: 240 000 pixels (or values of 9 000 Gabor filters)
  → high-dimensional!
- output: 120 actions
  → high-dimensional!

Why does it work?

What is the ‘real’ input dimension? (states)



What is the ‘real’ output-dimension? (actions)

Inductive bias in Reinforcement Learning

Before you code an RL problem, try to answer the following questions:

1) Is the problem such that in similar (neighboring) input states the best action is (likely to be) the same?

Yes → broad overlapping representation of states is possible.

low intrinsic dimensionality of state space → sampling possible

2) Is the problem such that if I find the reward with action a^* from state s, then a^* is probably good in all states?

No, not in presence of obstacles or objects in the middle

→ global representation of states is not useful.

Inductive bias in Reinforcement Learning

3) Do I expect rewards in many states or rather only in a few ‘goal states’?

Single goal state → long eligibility traces possible.

4) Moreover, are rewards given for states or state-action transitions?

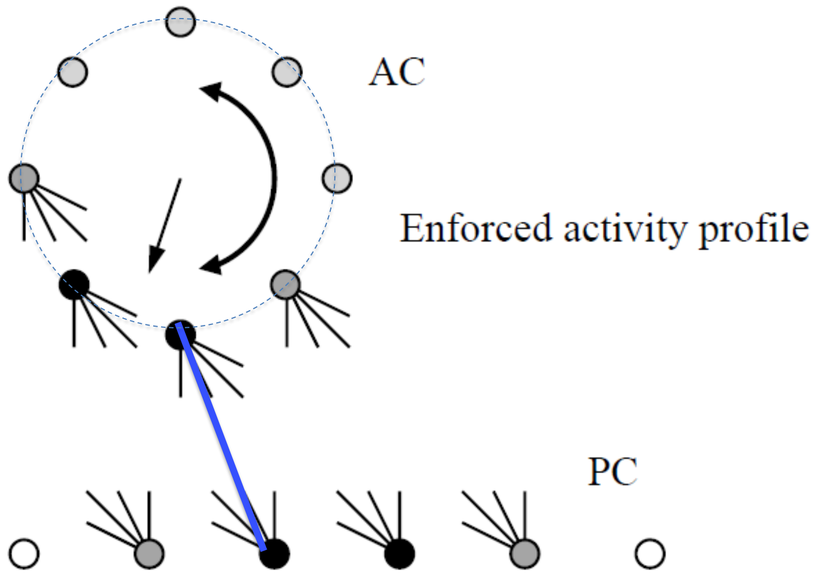
Rewards only in states → exploration easier: stop exploration if each state well represented

5) Is there a topology/neighborhood relation that would enable us to talk about two actions being ‘similar’?

Inductive bias in Reinforcement Learning

5) Is there a topology/neighborhood relation that would enable us to talk about two actions being ‘similar’?

Yes → Generalization across actions space possible.



Enforce activity profile
= spread Q-value activity from ‘winning action’ to neighbors
= learn neighbors at the same time
= learn as if all similar actions had been taken as well

Inductive bias in Reinforcement Learning

The EFFECTIVE number of parameters is much lower than the number of weights, since neighboring state neurons and neighboring action neurons learn similar things.

Additional inductive bias is also used in this example:

- Odometry (wheel turns) allows to give a noisy prediction of current location.
- This prediction can be combined with the filter response to give more localized filters
- The odometry in turn can be calibrated by the recognized filter responses.
- No stable compass, GPS, or knowledge of ‘where’ necessary

Previous slides.

- Coding of input space: we sample vectors of feature responses in the high-dimensional space, but we know that in the end they encode only two dimensions, so that sampling is indeed possible.

- The space is further reduced from 3 to 2 dimensions by algorithmic rotation of images (= shift of feature vectors) to get rid of difference due to orientation.

- We can work with relatively long eligibility traces, since there is a single goal state.

- We generalize across actions: we imagine actions forming a ring of possible directions. Neighboring actions should learn (in most states) similar behavior, hence if action a* is chosen with SARSA and learns (at rate eta), then all its neighbors learn as well (but with slightly reduced rate).

Inductive bias in Reinforcement Learning

Use all prior knowledge you have, before you start coding:

- No Free Lunch
- a generic neural network is rarely the best
- choose encoding and preprocessing so that generalization across ‘similar things’ becomes possible.

Reinforcement Learning can be extremely fast!!!

Reinforcement Learning needs a good representation of states!

Representation of states can be learned with three-factor rules: biologically plausible, no BackProp

Previous slide. Summary

.