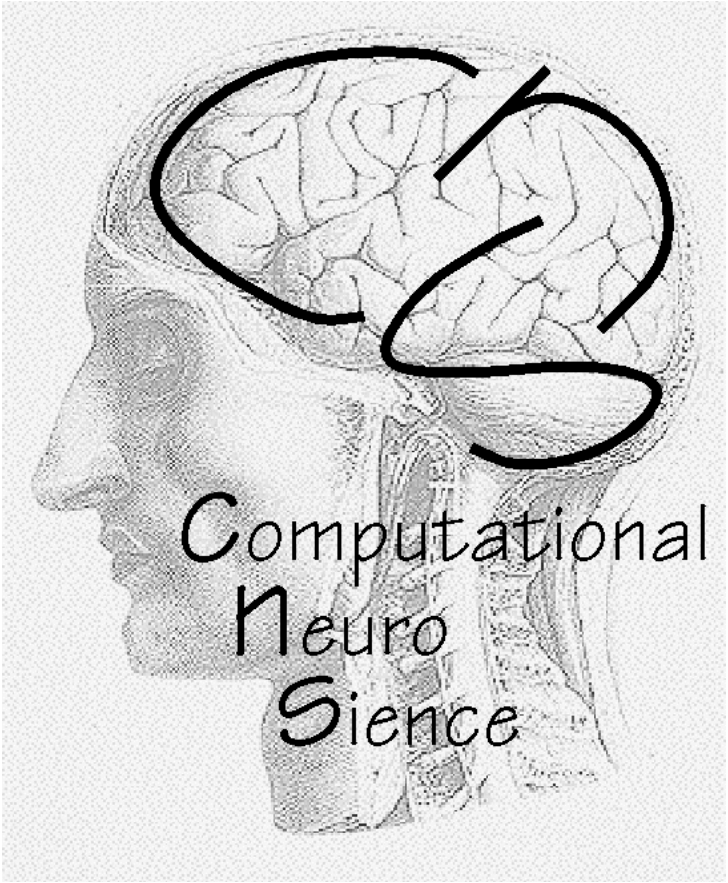
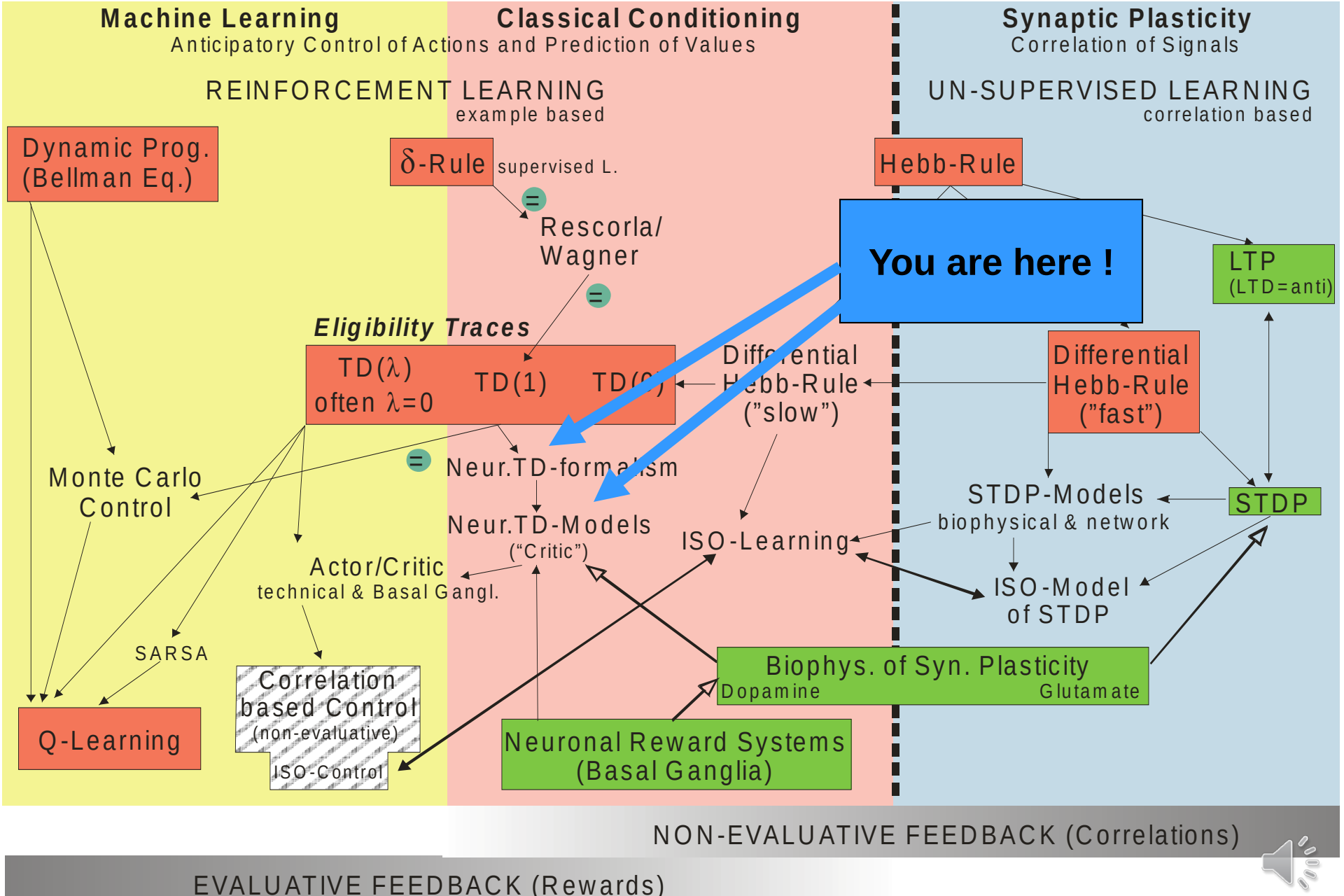
# *Advanced Computational Neuroscience*




Lecture 7:

Reinforcement Learning
Part B

# Reinforcement Learning – Relations to Brain Function I



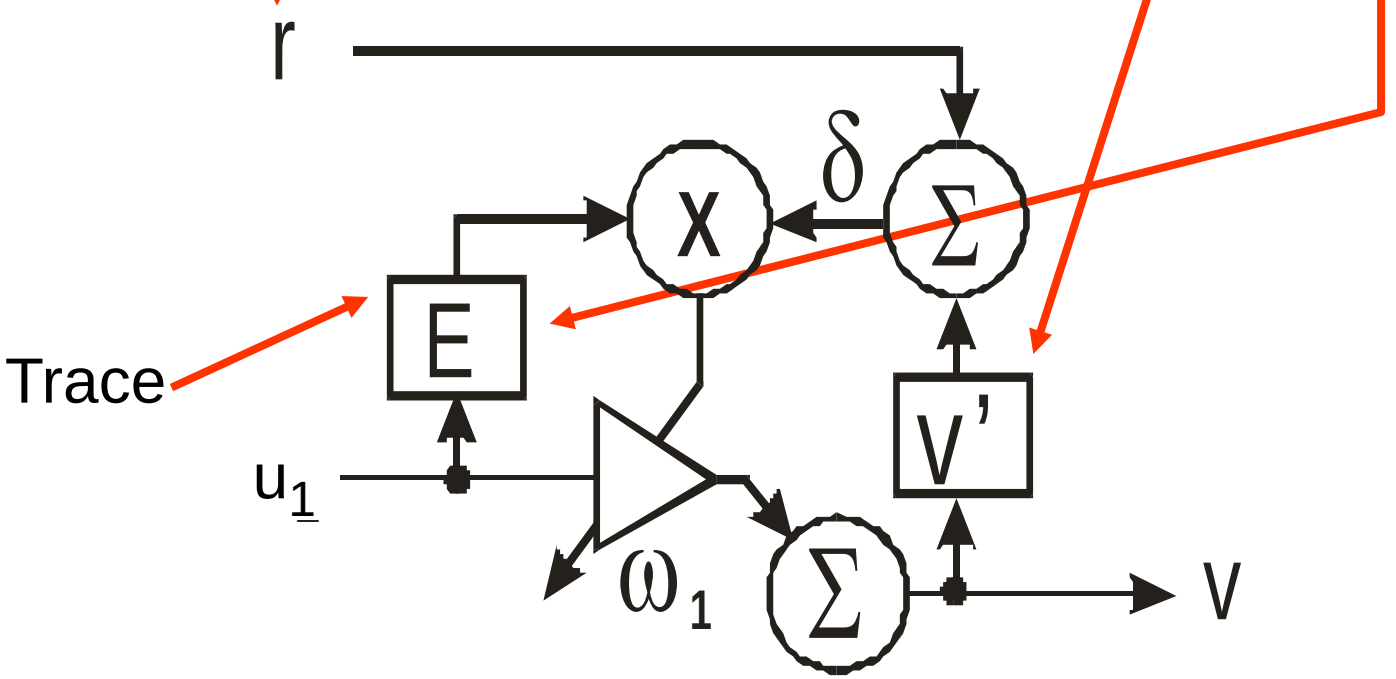
# How to implement TD in a Neuronal Way

Now we have: $V(s_t) \leftarrow V(s_t) + \alpha \delta_t \overline{x_t}(s)$

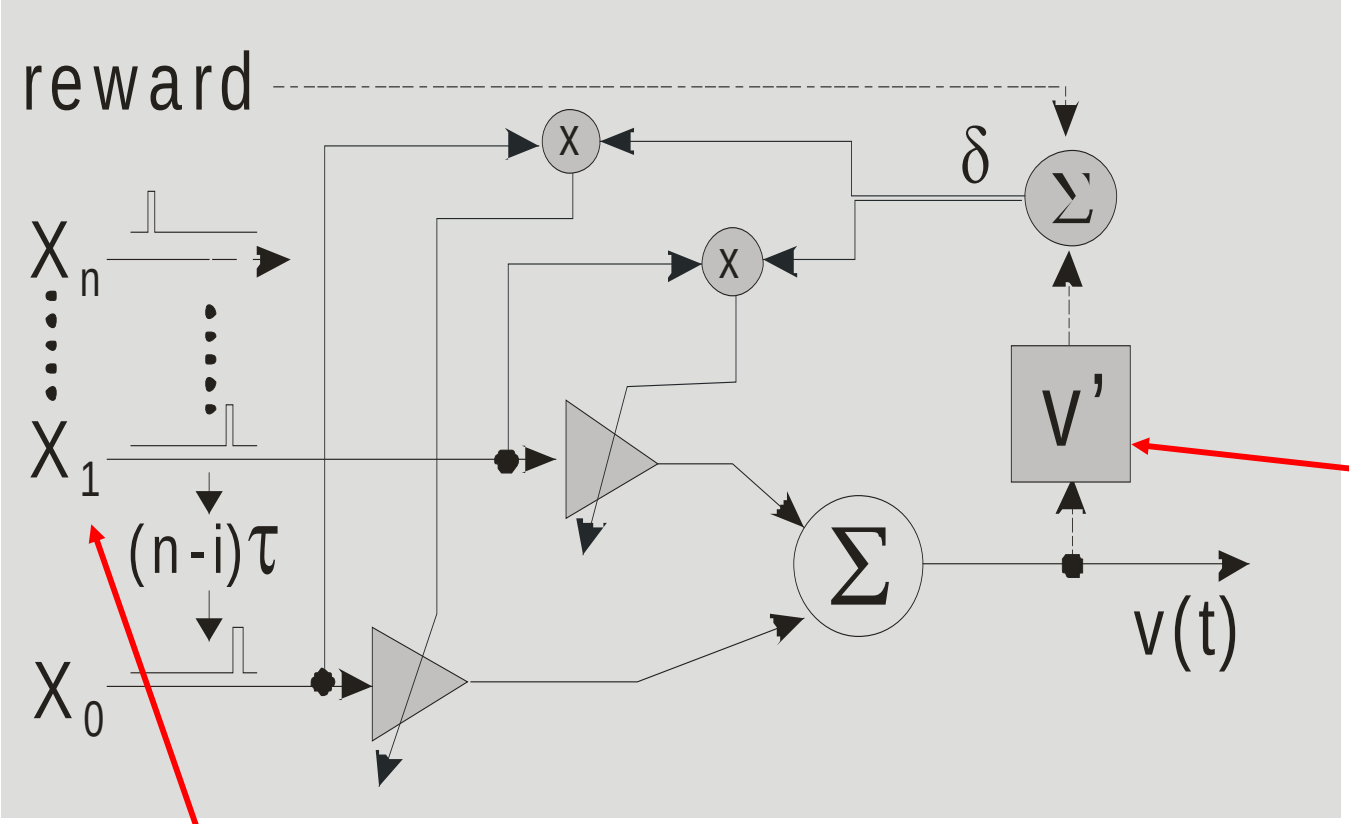
$\delta_t = r_{t+1} + \gamma V(s_{t+1}) - V(s_t)$

We had defined:
(first lecture!)

$w_i \rightarrow w_i + \mu[r(t+1) + \gamma\ v(t+1) - v(t)]\tilde{u}(t)$



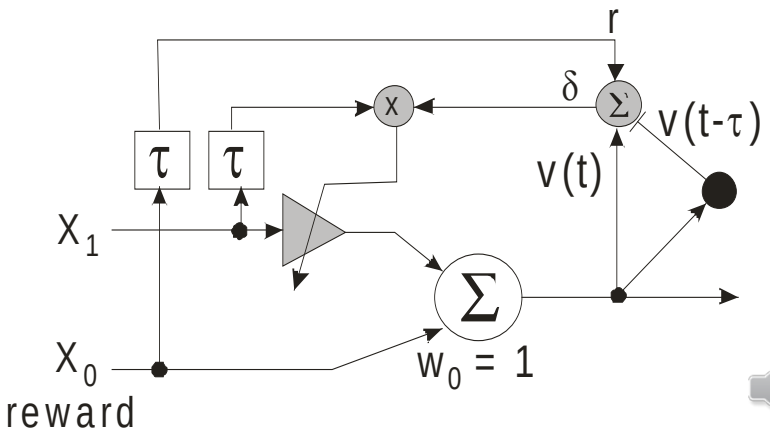
# How to implement TD in a Neuronal Way



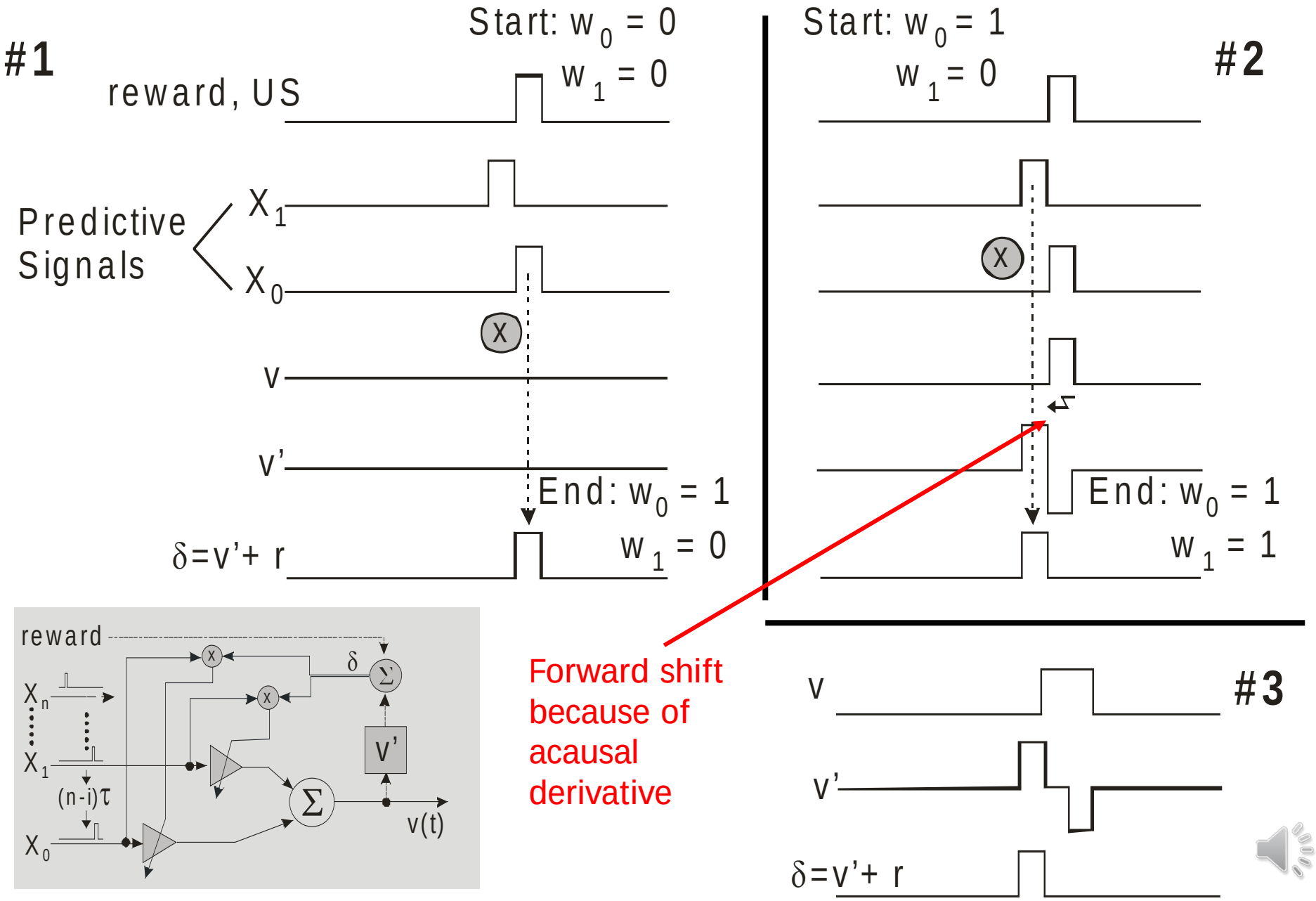

v(t+1)-v(t)

Serial-Compound representations $X_1,...X_n$ for defining an eligibility trace.

Note: v(t+1)-v(t) is acausal (future!). Make it "causal" by using delays.



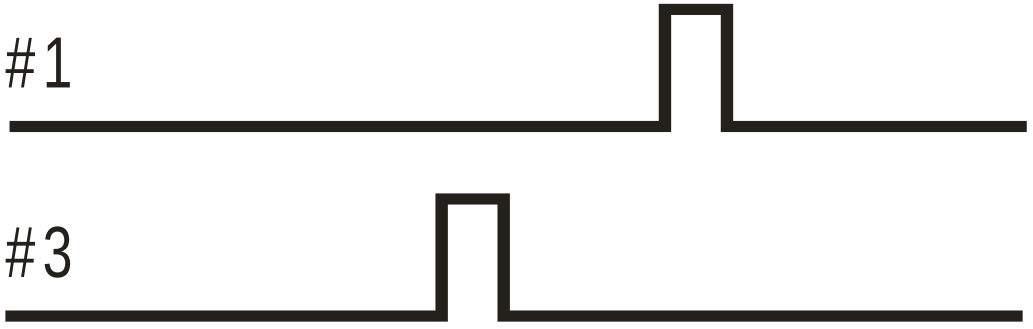
# How does this implementation behave: $w_i \leftarrow w_i + \mu\delta x_i$



# Observations

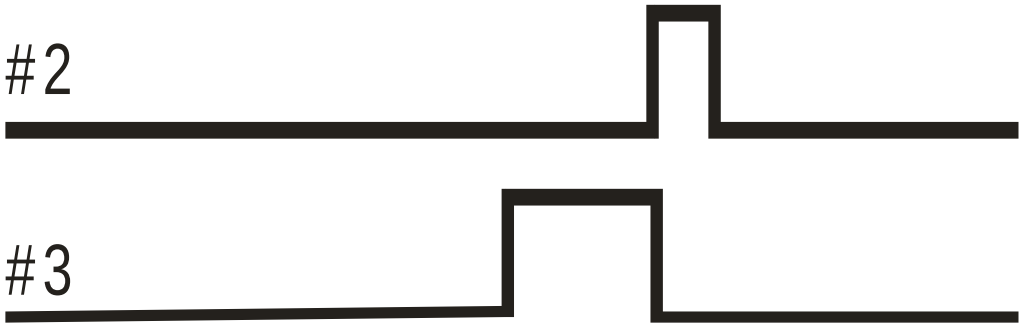
$\delta = v' + r$

$\delta$-error moves forward from the US to the CS.

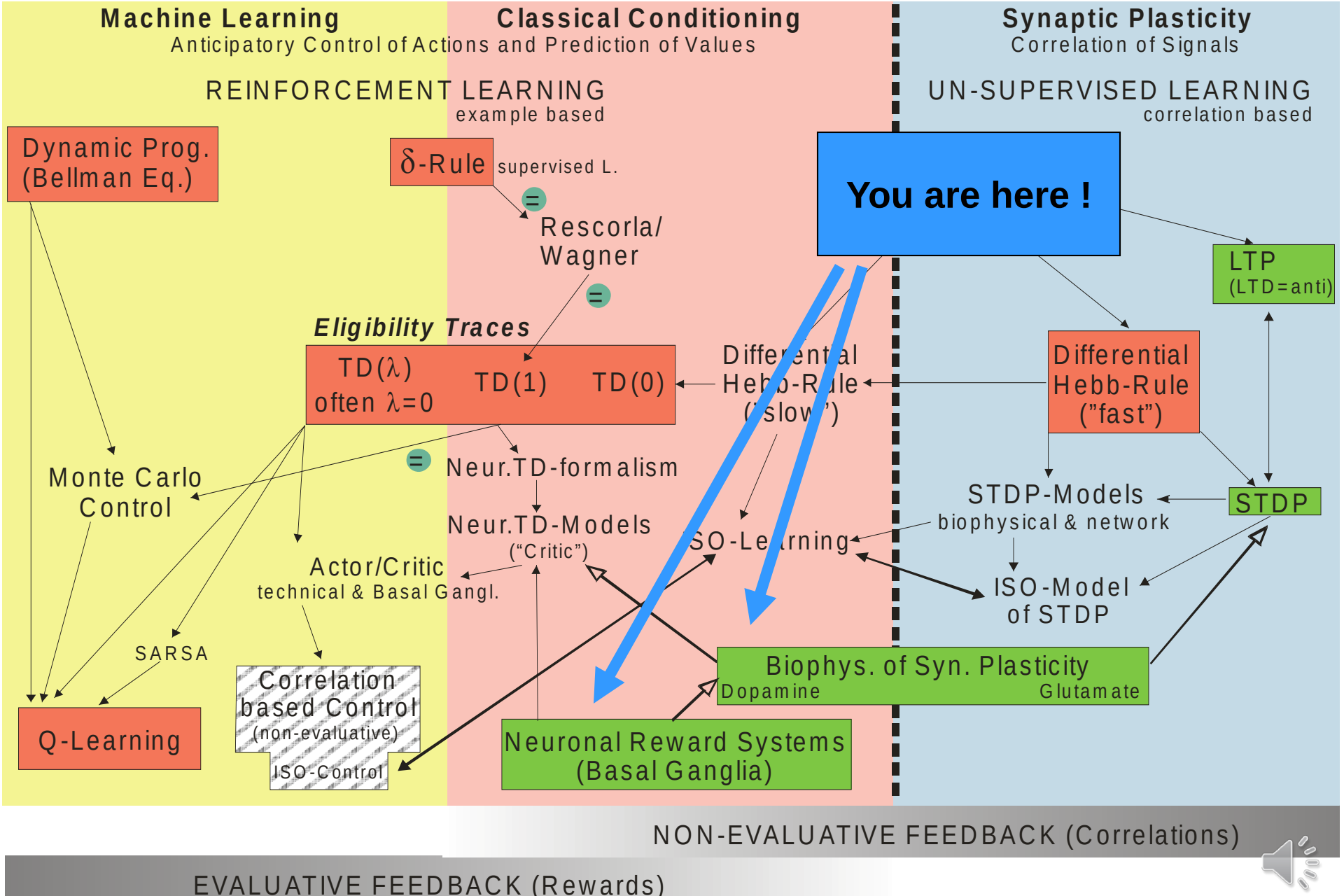



The reward expectation signal extends forward to the CS.

$V$

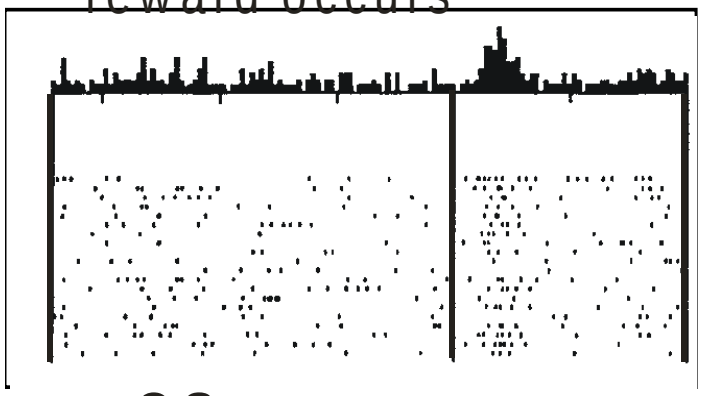


# Reinforcement Learning – Relations to Brain Function II

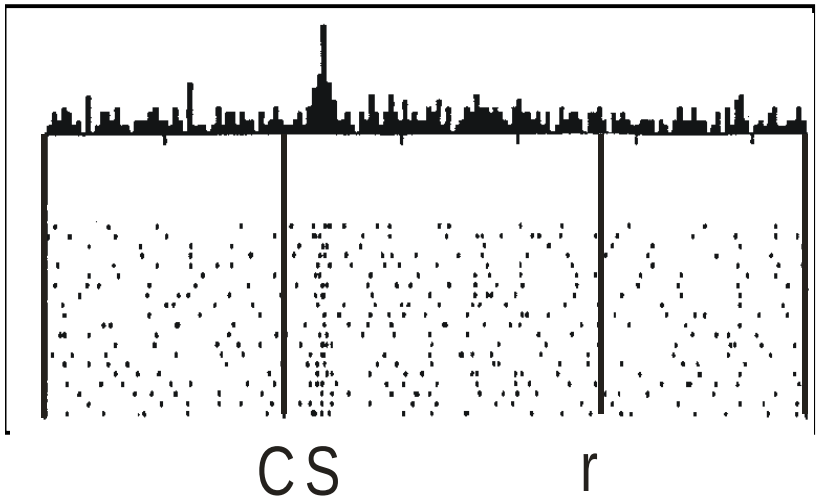


# TD-learning & Brain Function

Novelty Response:
no prediction,
reward occurs



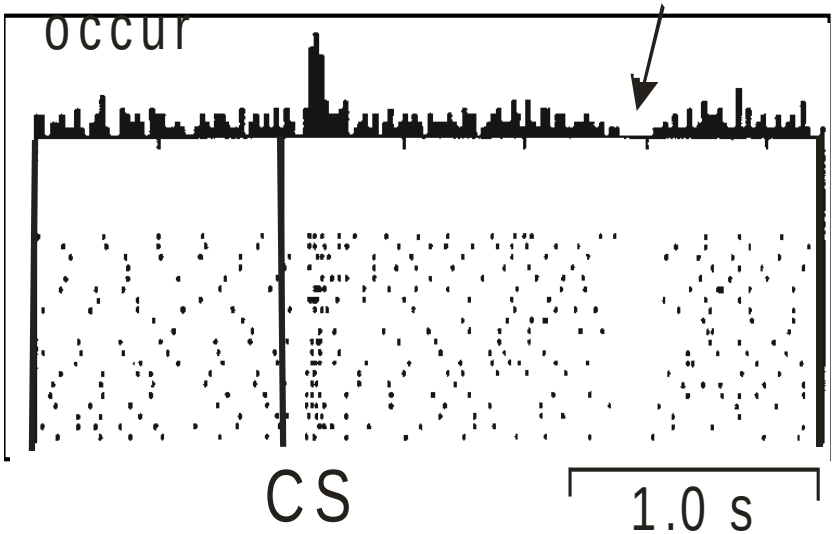

After learning:
predicted reward occurs




This neuron is supposed to represent the $\delta$-error of TD-learning, which has moved forward as expected.

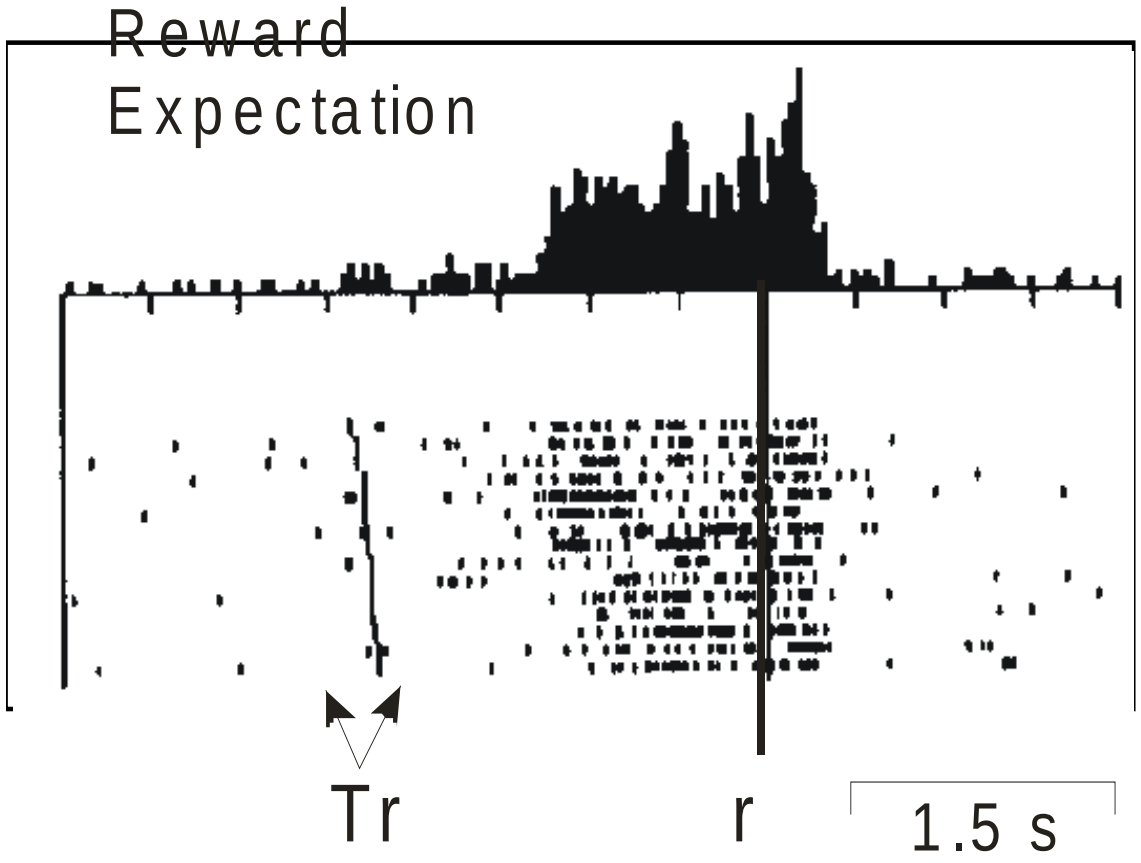
DA-responses in the basal ganglia pars compacta of the substantia nigra and the medially adjoining ventral tegmental area (VTA).

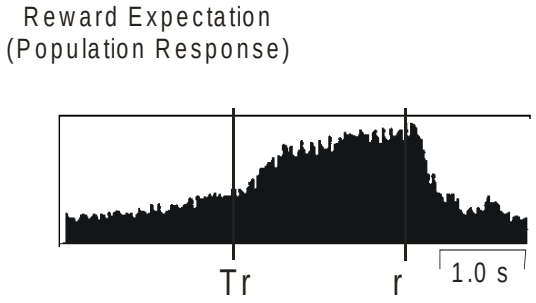
After learning:
predicted reward does not occur




Omission of reward leads to inhibition as also predicted by the TD-rule.

# TD-learning & Brain Function



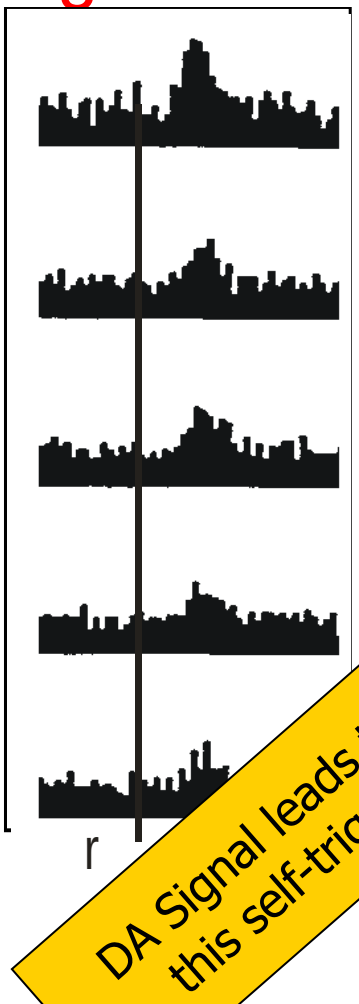




This is even better visible from the population response of 68 striatal neurons

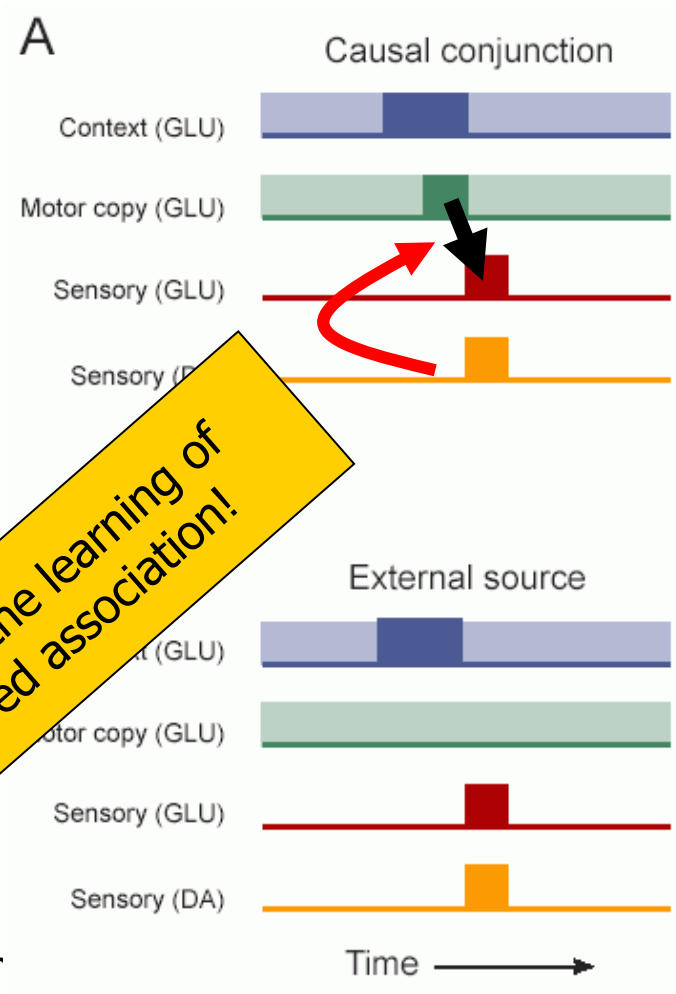
This neuron is supposed to represent the reward expectation signal v. It has extended forward (almost) to the CS (here called Tr) as expected from the TD-rule. Such neurons are found in the striatum, orbitofrontal cortex and amygdala.

# TD-learning & Brain Function Deficiencies

Continuous decrease of novelty response during learning




DA Signal leads to the learning of this self-triggered association!




Incompatible to [illegible]al compour representation of the stimulus as the δ-error should move step by step forward, which is not found. Rather it shrinks at r and grows at the CS.

=cause-effect

There are short-latency Dopamine responses! These signals could promote the discovery of agency (i.e. those initially unpredicted events that are caused by the agent) and subsequent identification of critical causative actions to reselect components of behavior and context that immediately precede unpredicted sensory events. When the animal/agent is the cause of an event, repeated trials should enable the basal ganglia to converge on behavioral and contextual components that are critical for eliciting it, leading to the emergence of a novel action.

# Reinforcement Learning – The Control Problem

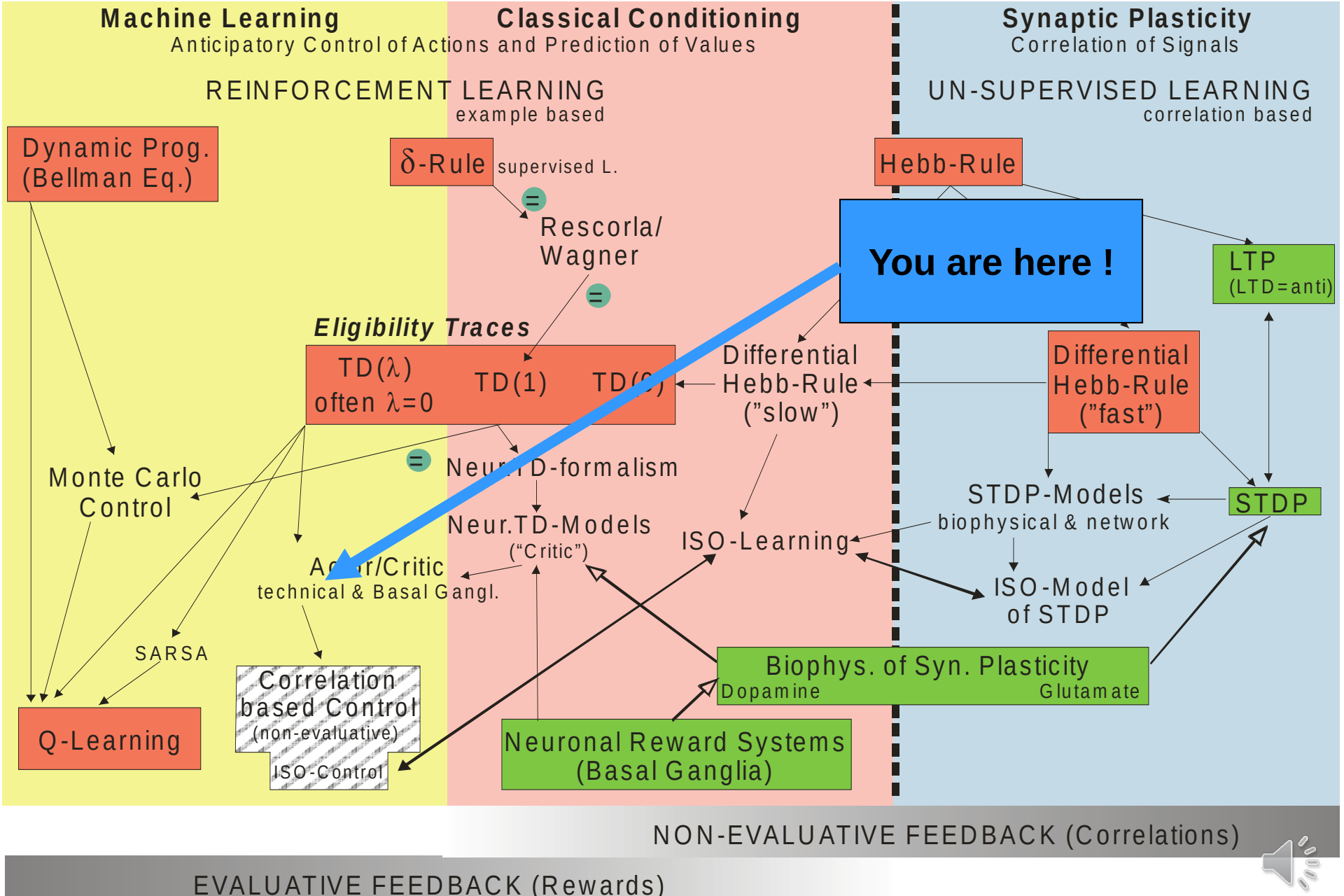
So far we have concentrated on evaluating an unchanging policy. Now comes the question of how to actually improve a policy $\pi$ trying to find the optimal policy.

We will discuss:

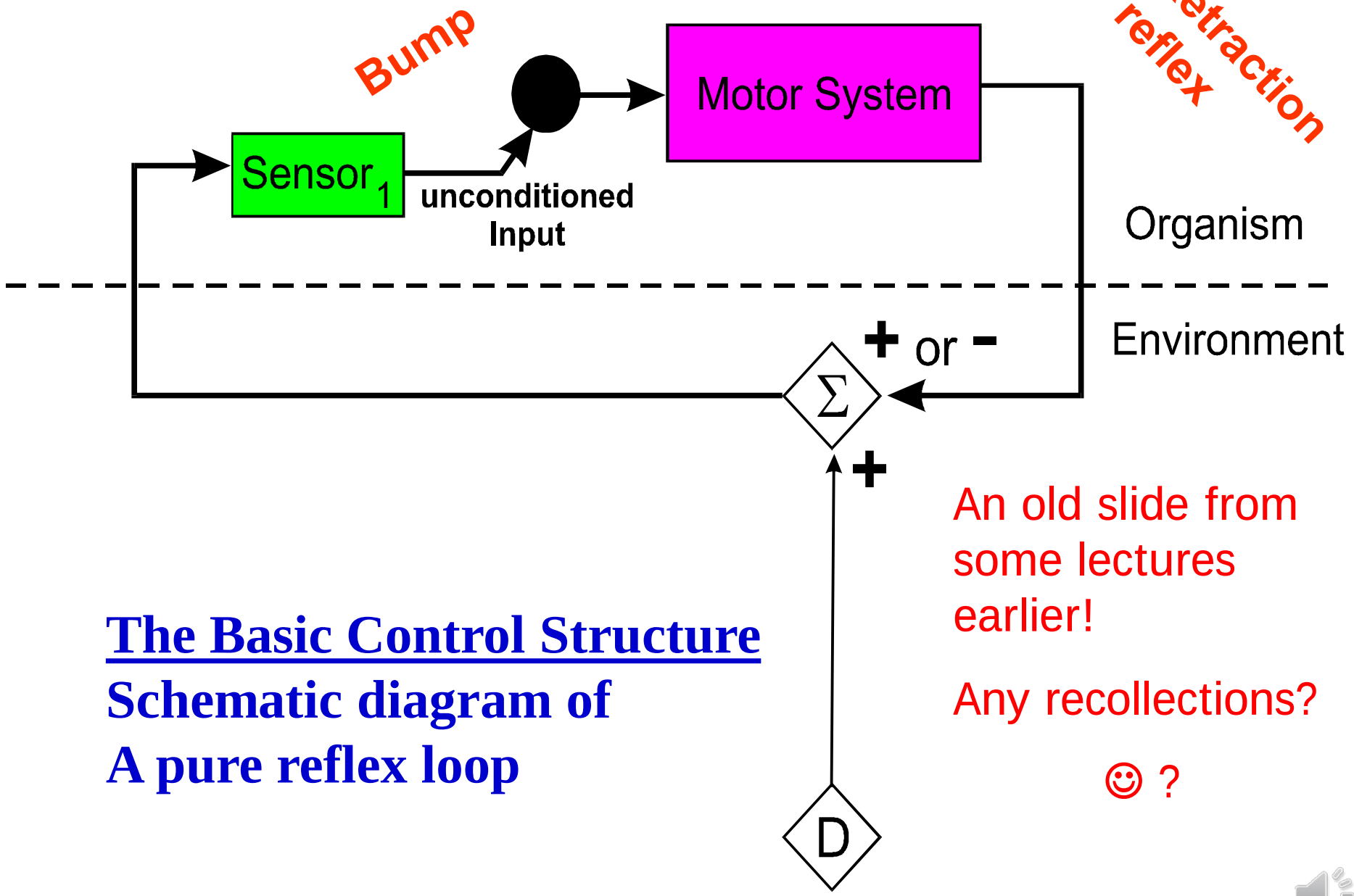
1) Actor-Critic Architectures
2) SARSA Learning
3) Q-Learning

Abbreviation for policy: $\pi$

# Reinforcement Learning – Control Problem I



# Control Loops




## The Basic Control Structure

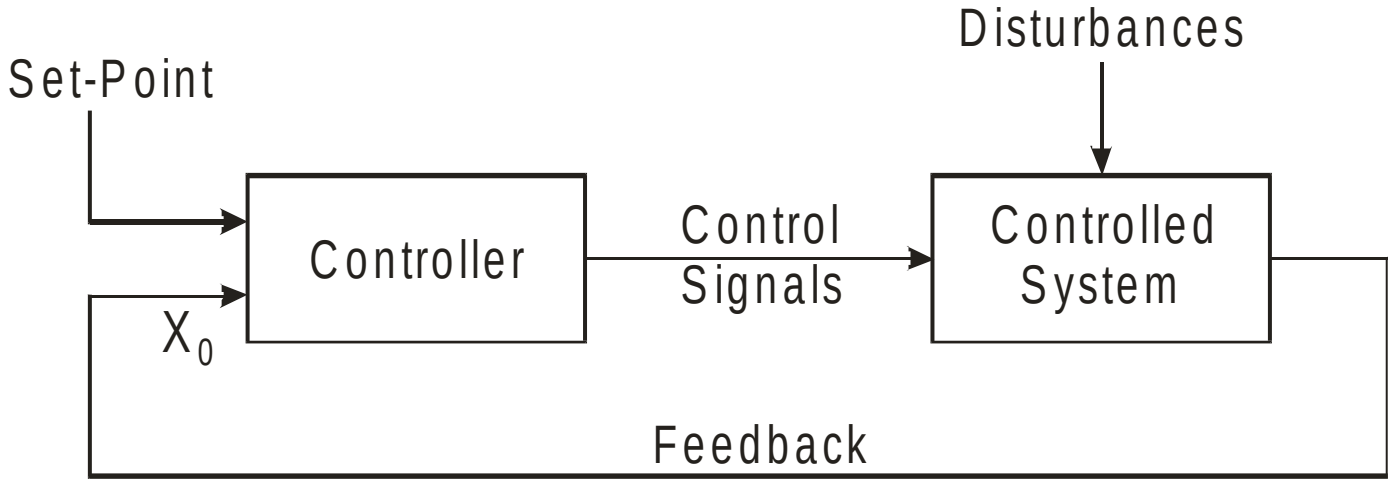
**Schematic diagram of**
**A pure reflex loop**

An old slide from some lectures earlier!

Any recollections?

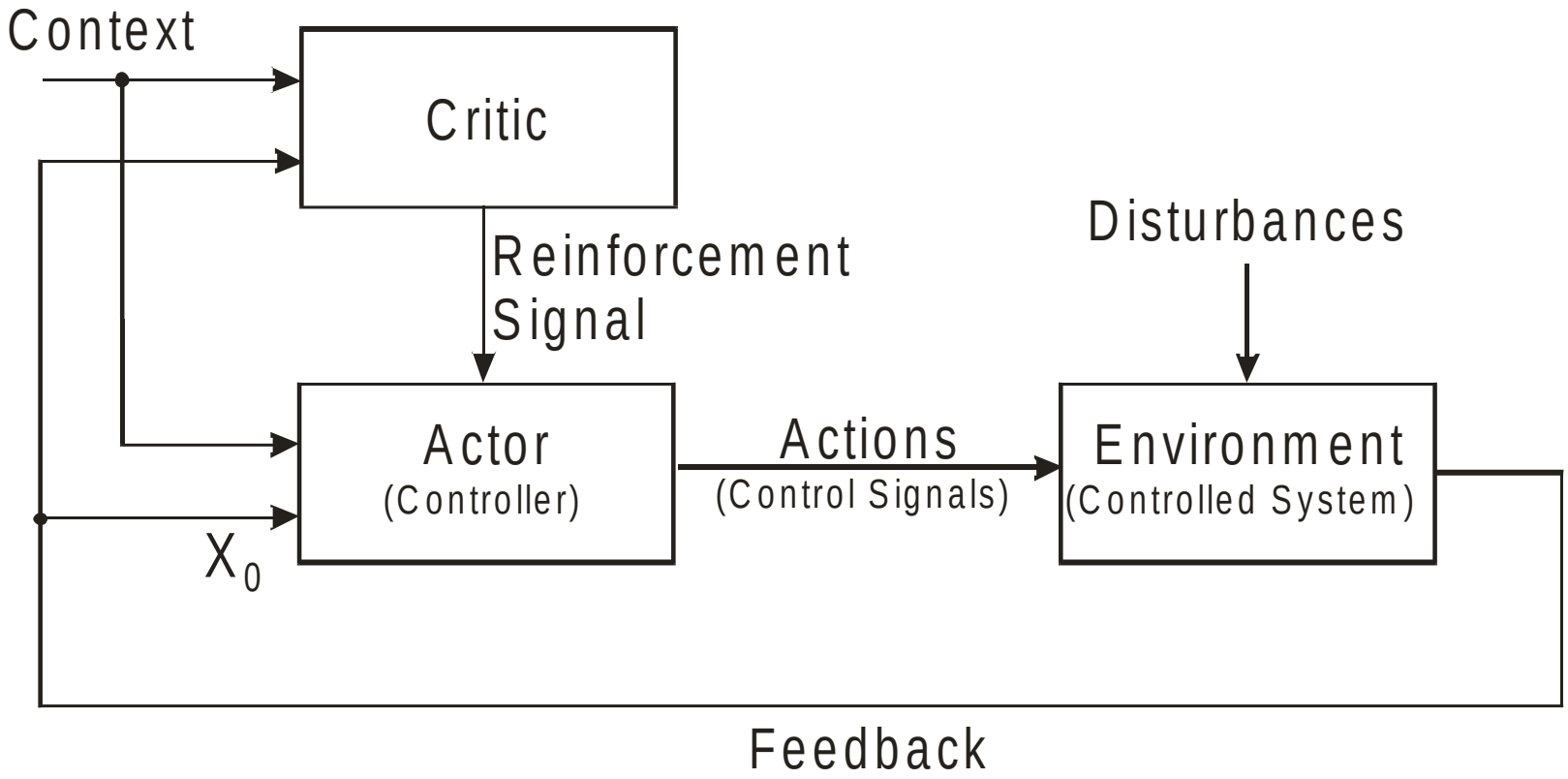
☺ ?

# Control Loops




A basic feedback–loop controller (Reflex) as in the slide before.

# Control Loops




An Actor-Critic Architecture: The Critic produces *evaluative*, reinforcement feedback for the Actor by observing the consequences of its actions. The Critic takes the form of a TD-error which gives an indication if things have gone better or worse than expected with the preceding action. Thus, this TD-error can be used to evaluate the preceding action: If the error is positive the tendency to select this action should be strengthened or else, lessened.

# Example of an Actor-Critic Procedure

Action selection here follows the Gibb's Softmax method:

$$\pi(s,a) = \frac{e^{p(s,a)}}{\sum_b e^{p(s,b)}}$$

where p(s,a) are the values of the modifiable (by the Critic!) policy parameters of the actor, indicating the tendency to select action a when being in state s.

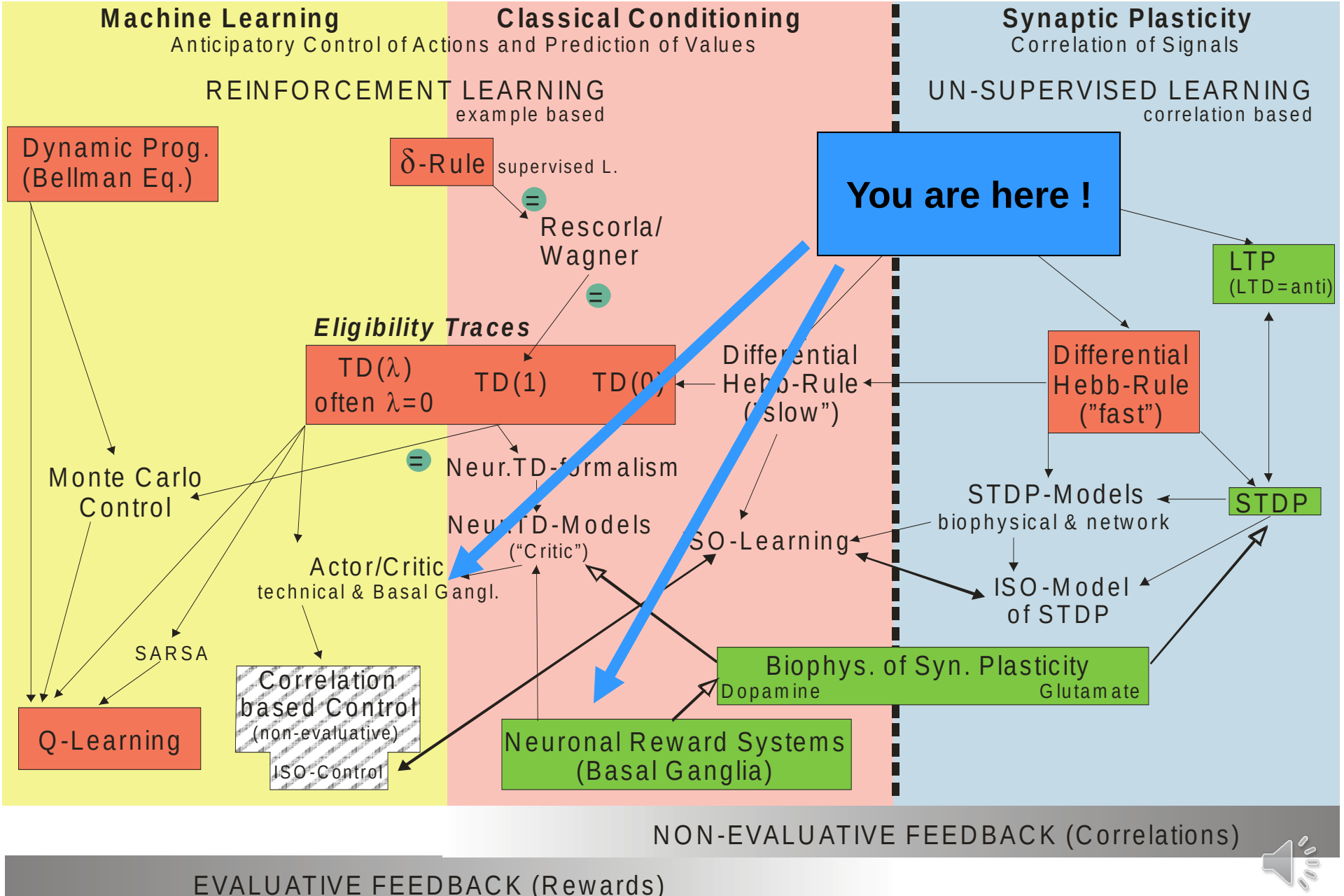
We can now modify p for a given state action pair at time t with:

$$p(s_t,a_t) \leftarrow p(s_t,a_t) + \beta\delta_t$$

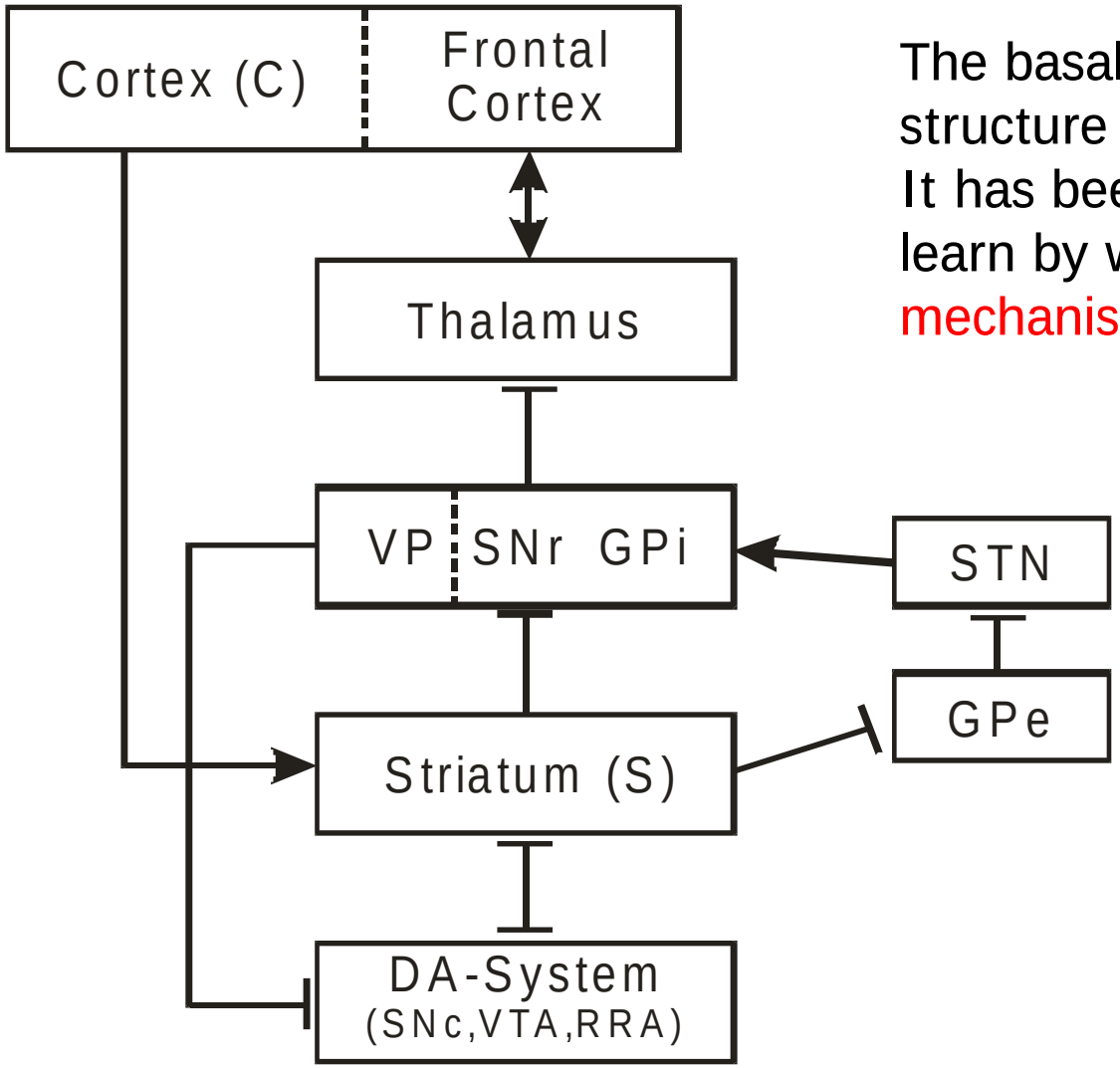
where $\delta_t$ is the $\delta$-error of the TD-Critic.

$$\delta_t = r_{t+1} + \gamma V(s_{t+1}) - V(s_t)$$

# Reinforcement Learning – Control I & Brain Function III



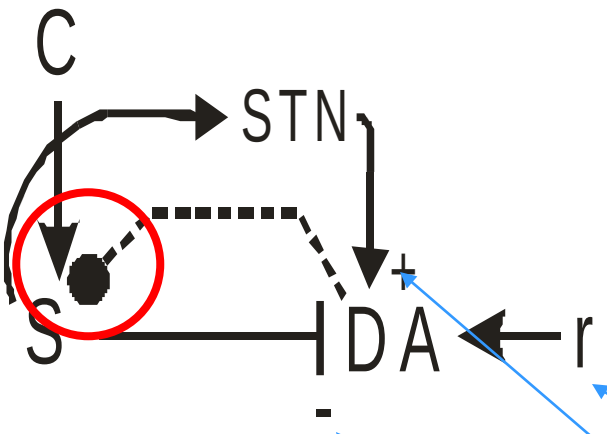
# Actor-Critics and the Basal Ganglia



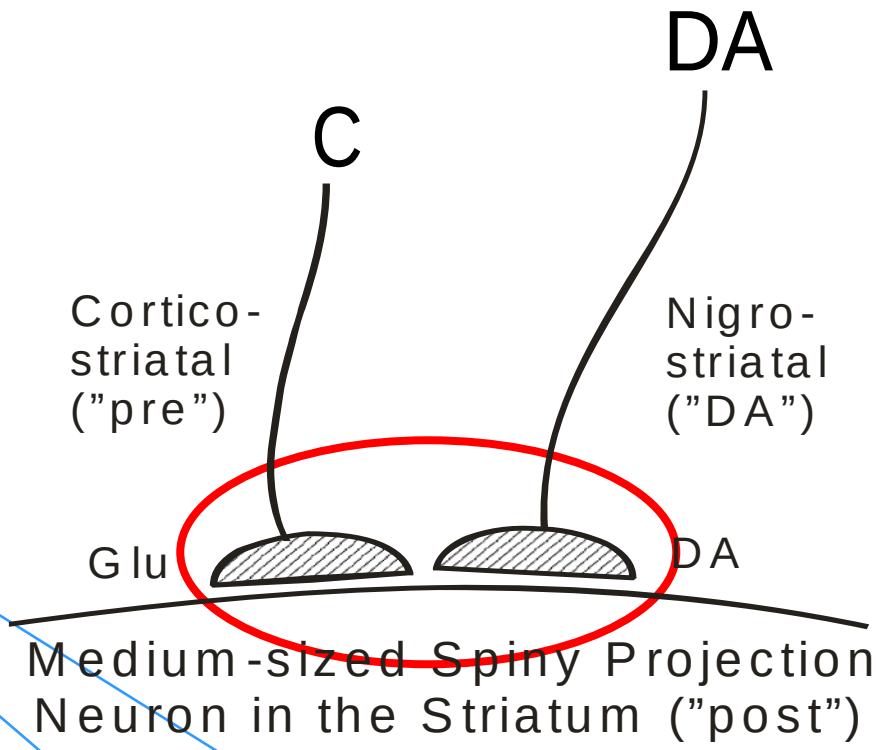

The basal ganglia are a brain structure involved in motor control. It has been suggested that they learn by ways of an Actor-Critic mechanism.

VP=ventral pallidum,
SNr=substantia nigra pars reticulata,
SNc=substantia nigra pars compacta,
GPi=globus pallidus pars interna,
GPe=globus pallidus pars externa,
VTA=ventral tegmental area,
RRA=retrorubral area,
STN=subthalamic nucleus.

# Actor-Critics and the Basal Ganglia: The Critic

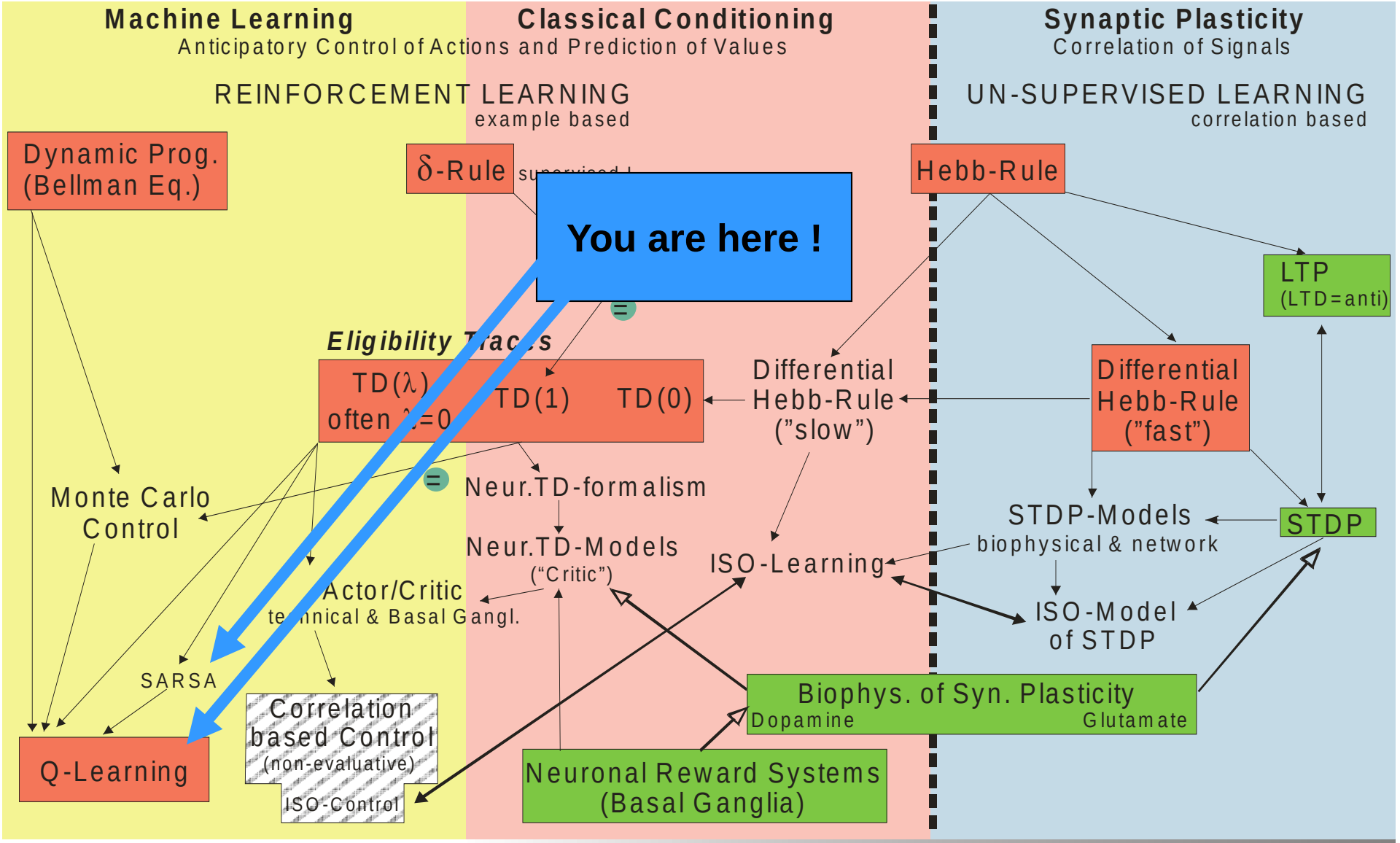



Cortex=C, striatum=S, STN=subthalamic Nucleus, DA=dopamine system, r=reward.




So called striosomal modules fulfil the functions of the adaptive Critic. The prediction-error ($\delta$) characteristics of the DA-neurons of the Critic are generated by: 1) Equating the reward r with excitatory input from the lateral hypothalamus. 2) Equating the term v(t) with indirect excitation at the DA-neurons which is initiated from striatal striosomes and channelled through the subthalamic nucleus onto the DA neurons. 3) Equating the term v(t−1) with direct, long-lasting inhibition from striatal striosomes onto the DA-neurons. There are many problems with this simplistic view though: timing, mismatch to anatomy, etc.

# Reinforcement Learning – Control Problem II

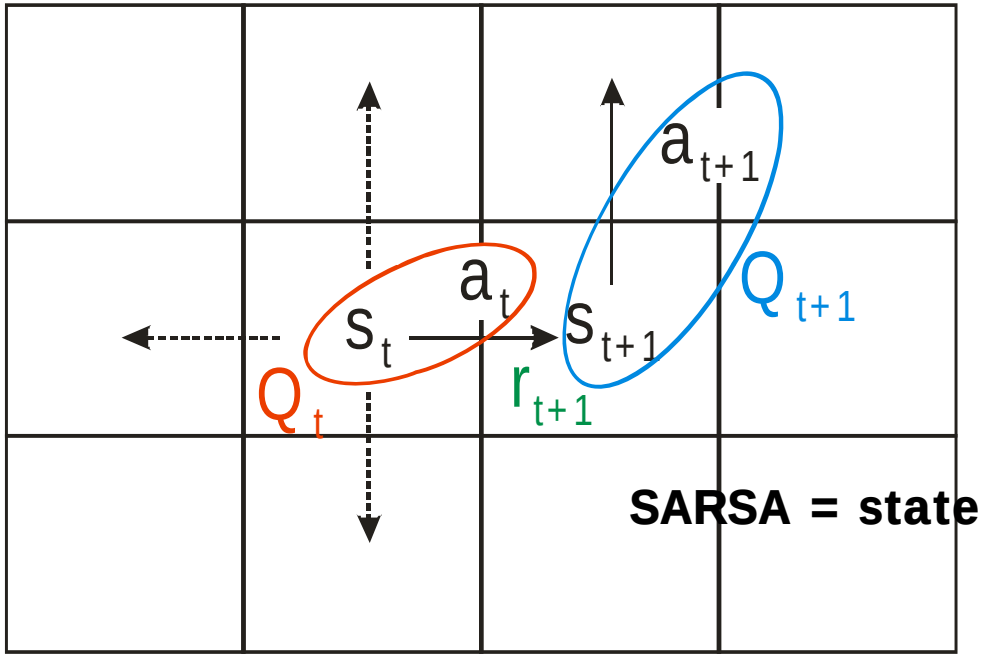


# SARSA-Learning

It is also possible to directly evaluate actions by assigning "Value" (Q-values and not V-values!) to state-action pairs and not just to states.

Interestingly one can use exactly the same mathematical formalism and write:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$$




The Q-value of state-action pair $s_t$,$a_t$ will be updated using the reward at the next state and the Q-value of the next *used* state-action pair $s_{t+1}$,$a_{t+1}$.

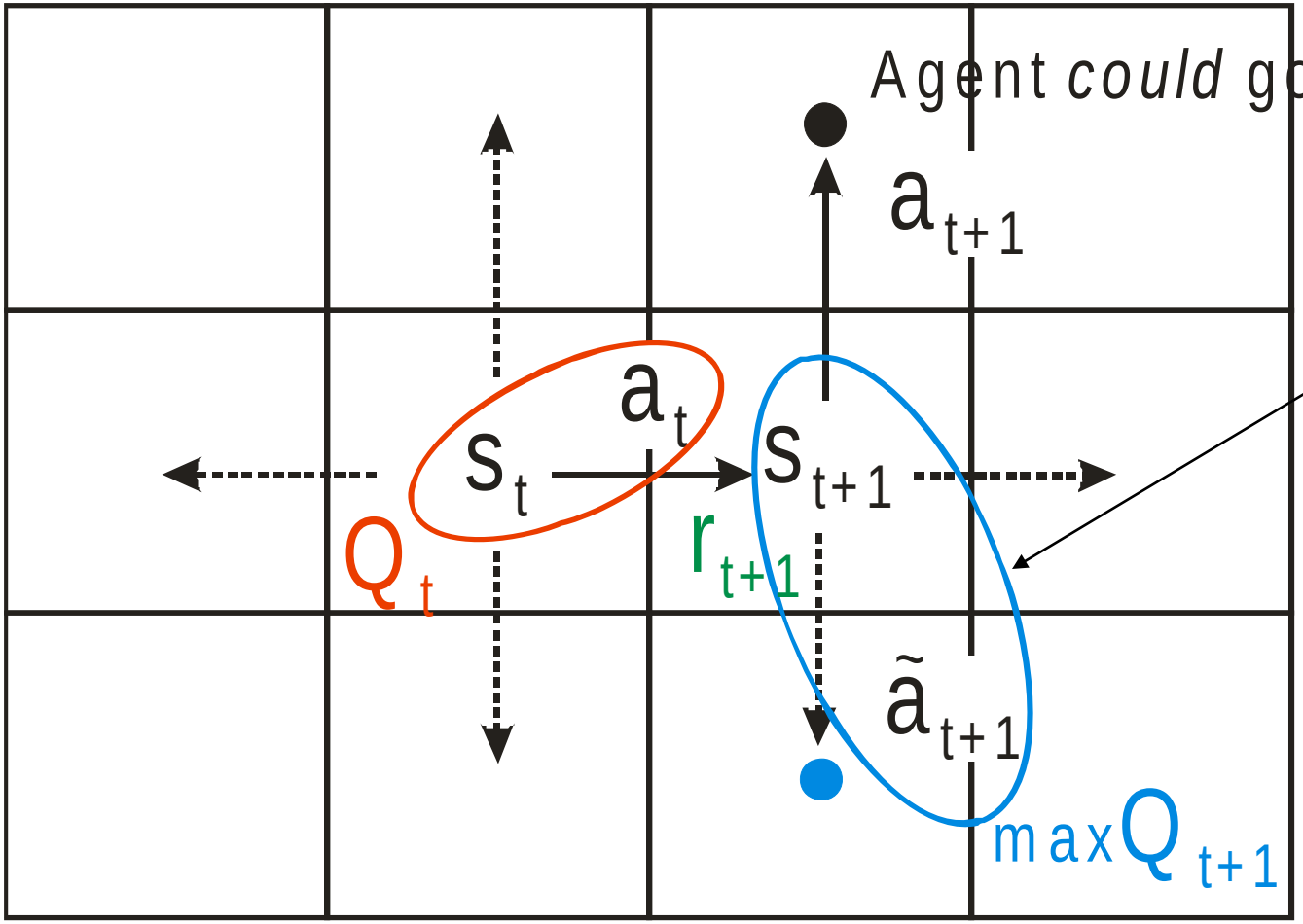
**SARSA = state-action-reward-state-action**

**On-policy update!**

# Q-Learning

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)]$$

**Note the difference! Called off-policy update.**




Even if the agent will not go to the 'blue' state but to the 'black' one, it will nonethe-less use the 'blue' Q-value for update of the 'red' state-action pair.

Notes:

1) For SARSA and Q-learning rigorous proofs exist that they will always converge to the optimal policy.
2) Q-learning is the most widely used method for policy optimization.
3) For *regular* state-action spaces in a *fully Markovian* system Q-learning converges faster than SARSA.

*Regular state-action spaces:* States tile the state space in a non-overlapping way. System is fully deterministic (Hence rewards and values are associated to state-action pairs in a deterministic way.). Actions cover the space fully.

Note: In real world applications (e.g. robotics) there are many RL-systems, which are not regular and not fully Markovian.

# Problems of RL

**Curse of Dimensionality**
In real world problems ist difficult/impossible to define discrete state-action spaces.

**(Temporal) Credit Assignment Problem**
RL cannot handle large state action spaces as the reward gets too much dilited along the way.

**Partial Observability Problem**
In a real-world scenario an RL-agent will often not know exactly in what state it will end up after performing an action. Furthermore states must be history independent.

**State-Action Space Tiling**
Deciding about the actual state- and action-space tiling is difficult as it is often critical for the convergence of RL-methods. Alternatively one could employ a continuous version of RL, but these methods are equally difficult to handle.

**Non-Stationary Environments**
As for other learning methods, RL will only work quasi stationary environments.

# Problems of RL

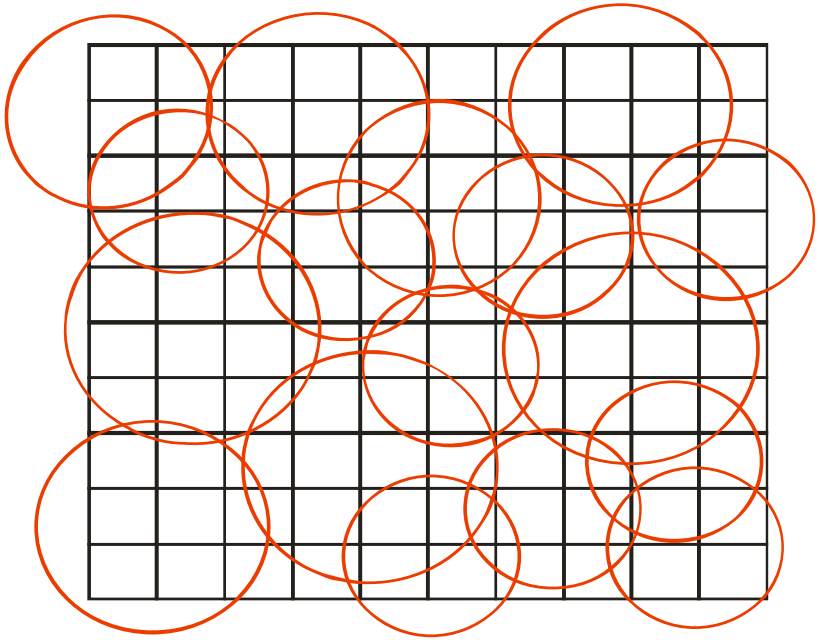
**Credit Structuring Problem**
One also needs to decide about the reward-structure, which will affect the learning. Several possible strategies exist:
**external evaluative feedback**: The designer of the RL-system places rewards and punishments *by hand*. This strategy generally works only in very limited scenarios because it essentially requires detailed knowledge about the RL-agent's world.
**internal evaluative feedback**: Here the RL-agent will be equipped with sensors that can measure physical aspects of the world (as opposed to 'measuring' numerical rewards). The designer then only decides, which of these physical influences are rewarding and which not.

**Exploration-Exploitation Dilemma**
RL-agents need to explore their environment in order to assess its reward structure. After some exploration the agent might have found a set of apparently rewarding actions. However, how can the agent be sure that the found actions where actually the best? Hence, when should an agent continue to explore or else, when should it just exploit its existing knowledge? Mostly heuristic strategies are employed for example *annealing-like* procedures, where the naive agent starts with exploration and its exploration-drive gradually diminishes over time, turning it more towards exploitation.

# (Action -)Value Function Approximation

In order to reduce the temporal credit assignment problem methods have been devised to approximate the value function using so-called features to define an augmented state-action space.

Most commonly one can use large, overlapping features (like “receptive fields”) and thereby coarse-grain the state space.



Black: Regular non-overlapping state space (here 100 states).

Red: Value function approximation using here 17 features, only.

Note: Rigorous convergence proof do *in general* not anymore exist for Function Approximation systems.

# An Example: Path-finding in simulated rats

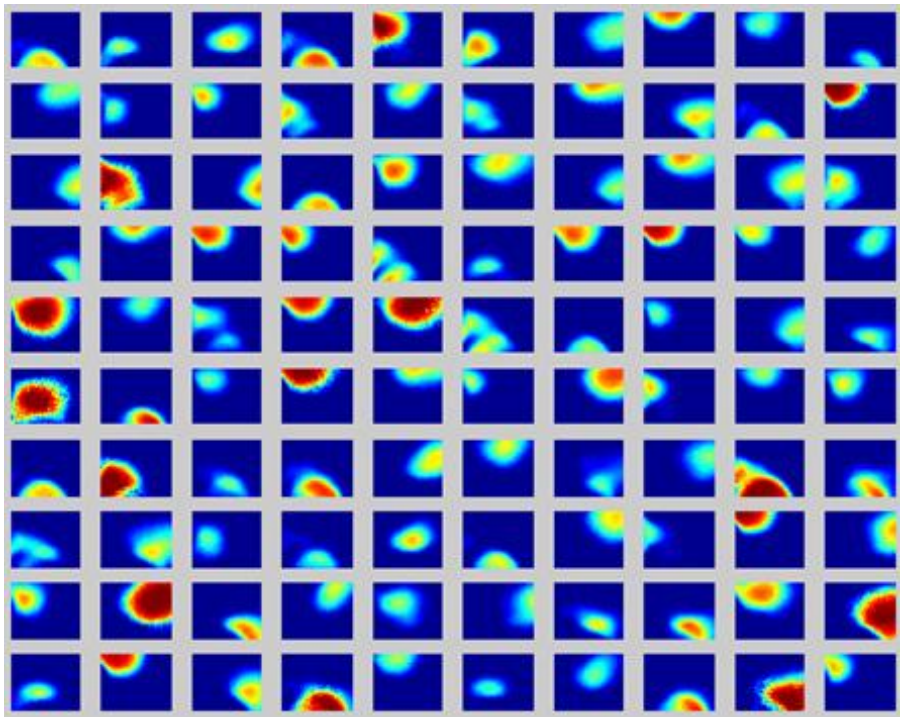
Goal: A simulated rat should find a reward in an arena.

This is a non-regular RL-system, because

1) Rats prefer straight runs (hence states are often “jumped-over” by the simulated rat). Actions do not cover the state space fully.

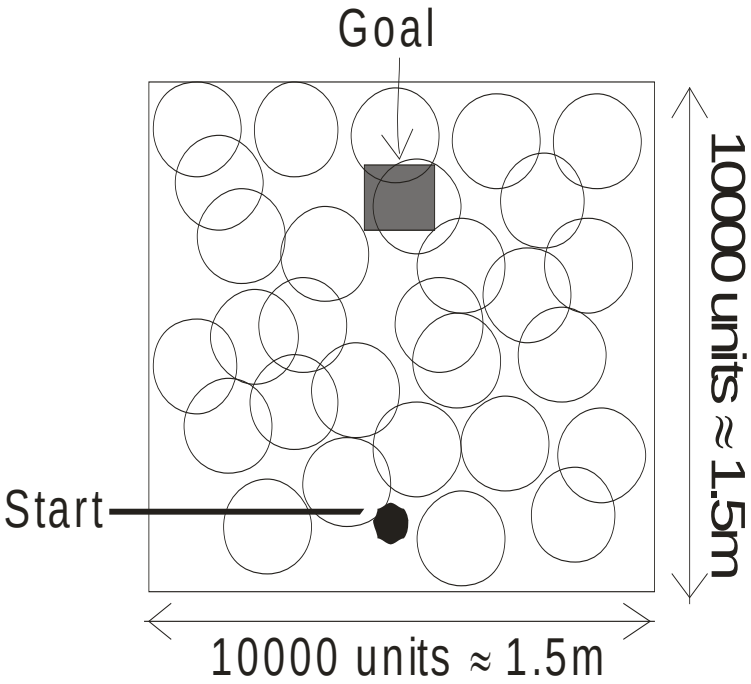
2) Rats (probably) use their hippocampal Place-Fields to learn such task. These place fields have different sizes and cover the space in an overlapping way. Furthermore, they fire to some degree stochastically.

   Hence they could represent an Action Value Function Approximation system.

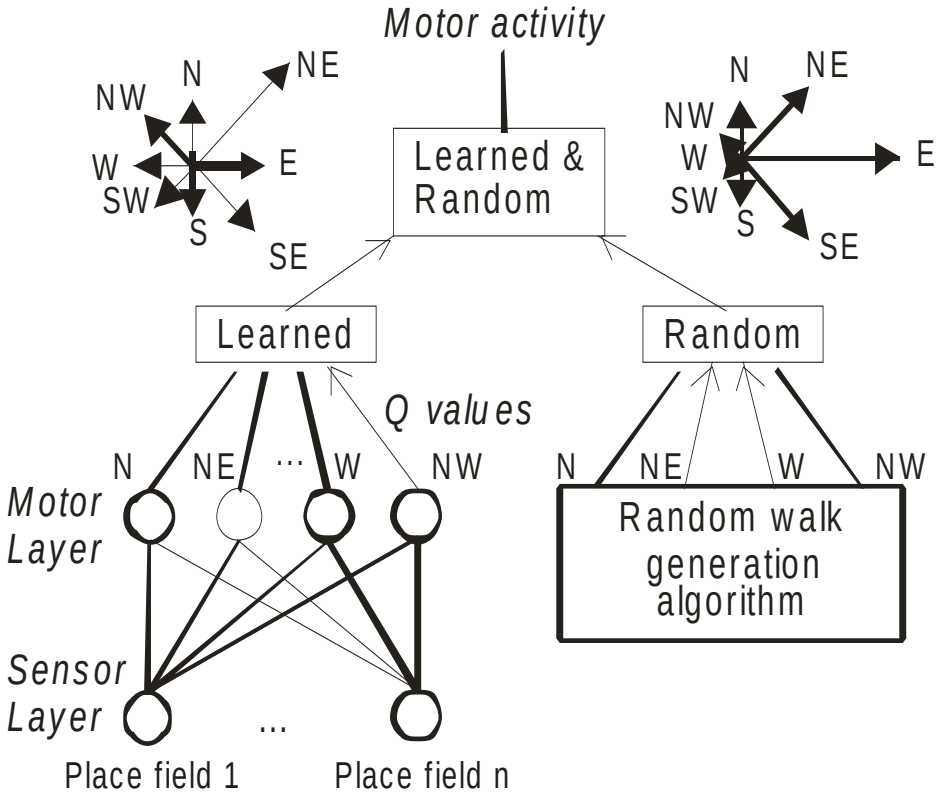


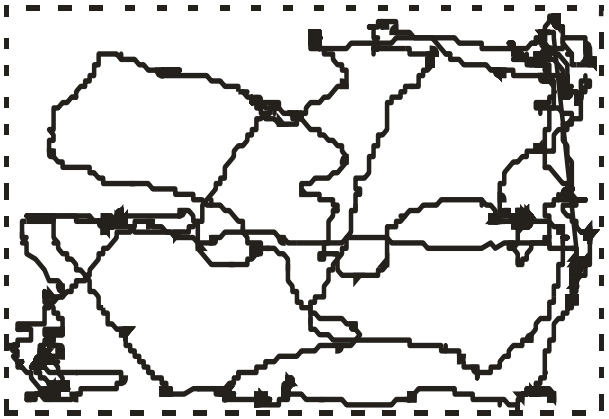
Place field activity in an areana

## Place field system




## Path generation and Learning



Real (left) and generated (right) path examples.

# Equations used for Function Approximation

We use SARSA as Q-learning is know to be more divergent in systems with function approximation:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t))$$

For function approximation, we define normalized Q-values by:

$$Q(s_t, a_t) = \sum_i \theta_{i,a_t} \Phi_i(s_t) / \sum_i \Phi_i(s_t)$$

where $\Phi_i(s_t)$ are the features over the state space, and $\theta_{i,a_t}$ are the adaptable weights binding features to actions.

We assume that a place cell i produces spikes with a scaled Gaussian-shaped probabilitv distribution:

$$p(\delta_i) = Ae^{-(\delta_i^2/\sigma^2)}$$

where $\delta_i$ is the distance from the i-th place field centre to the sample point (x,y) on the rat's trajectory, $\sigma$ defines the width of the place field, and A is a scaling factor.

We then use the actual place field spiking to determine the values for features $\Phi_i$, i = 1, .., n, which take the value of 1, if place cell i spikes at the given moment on the given point of the trajectory of the model animal, otherwise it is zero:

$$\Phi_i(s_t) = \begin{cases} 1 & \text{if place cell i spikes at } s_t \\ 0 & \text{else.} \end{cases}$$

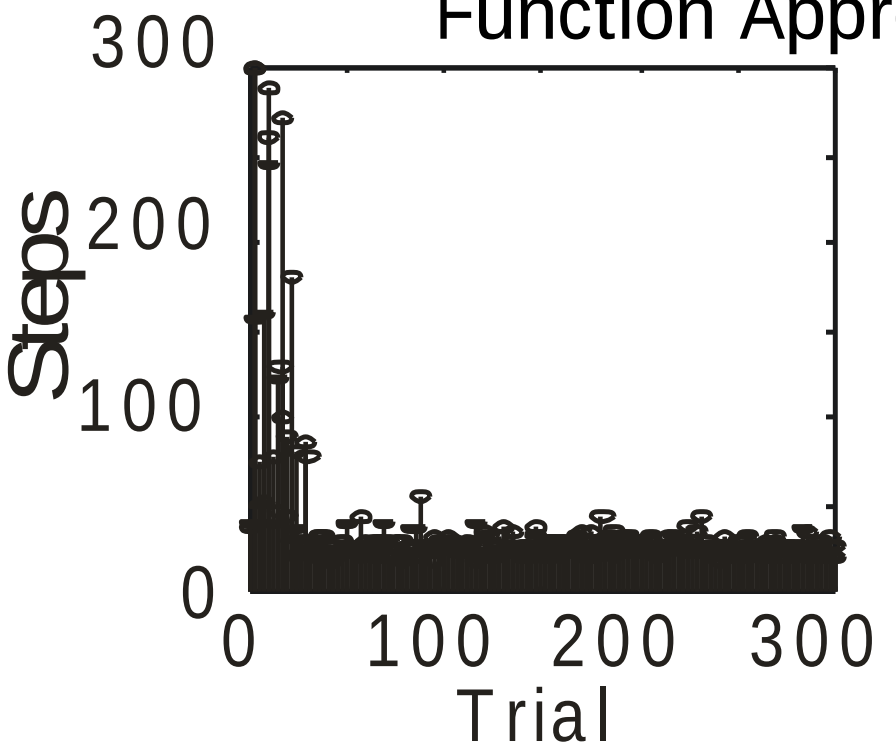
SARSA learning then can be described by:

$$\theta_{i,a_t} \leftarrow \theta_{i,a_t} + \alpha\big(r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - \theta_{i,a_t}\ \Phi_i(s_t)\big)$$

Where $\theta_{i,a_t}$ is the weight from the *i*-th place cell to action(-cell) *a*, and state $s_t$ is defined by $(x_t,y_t)$, which are the actual coordinates of the model animal in the field.

# Results

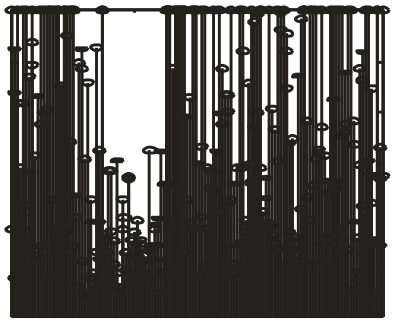
With Function Approximation




Without Function Approximation

With function approximation one obtains much faster convergence.

However, this system does not always converge anymore.



Divergent run

# RL versus CL

Reinforcement learning and correlation based (hebbian) learning in comparison:

RL:

1) Evaluative Feedback (rewards)
2) Network emulation (TD-rule, basal ganglia)
3) Goal directed action learning possible.

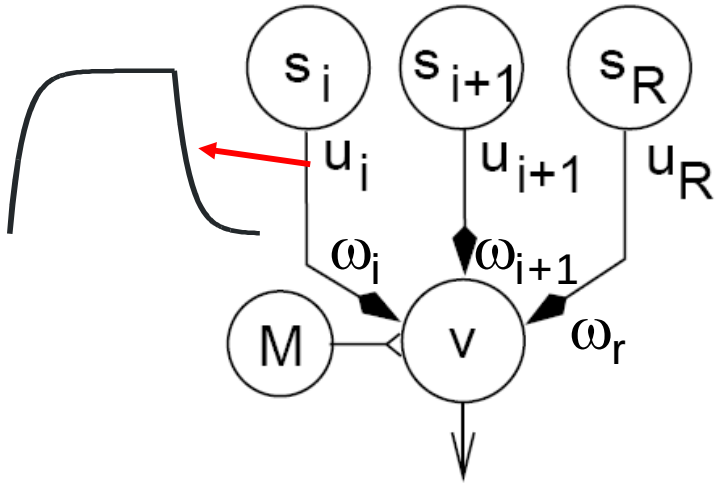
CL:

1) Non-evaluative Feedback (correlations only)
2) Single cell emulation ([diff.] Hebb rule, STDP)
3) Only homestasis action learning possible (?)

It can be proved that Hebbian learning which uses a third factor (Dopamine, e.g ISO3-rule) can be used to emulate RL (more specifically: the TD-rule) in a fully equivalent way.

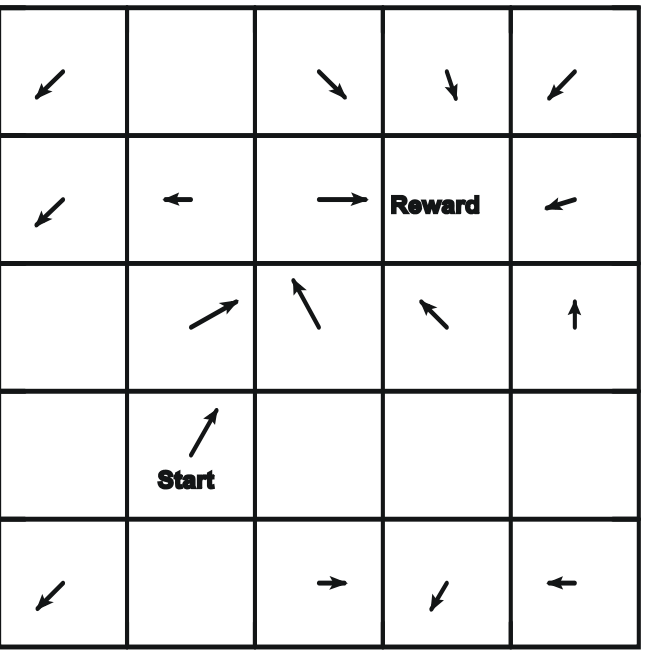
# Neural-SARSA (n-SARSA)

$$\frac{d\omega_i(t)}{dt} = \mu u_i(t)\, v_i'(t) M(t)$$



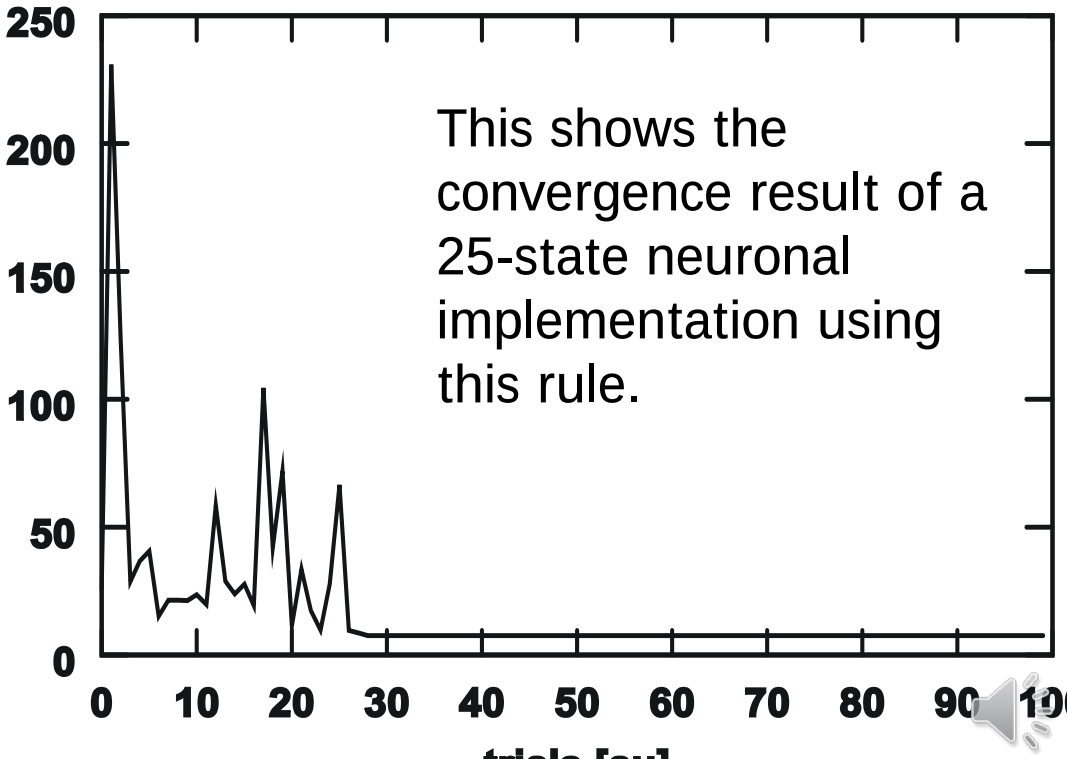

When using an appropriate timing of the third factor M and “humps” for the u-functions one gets exactly the TD values at $\omega_i$

**A)** **Action field**




x

**B)** **Number of steps to reach the goal**




This shows the convergence result of a 25-state neuronal implementation using this rule.

**Next:**

**Maps in the brain**