TD learning

Biology: dompanine

TD(λ)

Using a longer trajectory rather than single step:

For a single step:
The expected total Return from S on is:

$R(S) = r + \gamma V(S')$

For two steps:

$$R_t^{(2)} = r_{t+1} + \gamma r_{t+2} + \gamma^2 V_t(s_{t+2})$$

TD(λ)

n-step return at time t: Using a trajectory of length n

$$R_t^{(n)} = r_{t+1} + \gamma r_{t+2} + \gamma^2 + \cdots + \gamma^{n-1} r_{t+n} + \gamma^n V_t(s_{t+n}).$$

Estimation of the total return based on n steps

The value V(S) can be updated following n steps from S by:

$$\Delta V_t(s_t) = \alpha \Big[R_t^{(n)} - V_t(s_t) \Big],$$

Summary: generalizes the 1-step update

n-step return at time t:

$$R_t^{(n)} = r_{t+1} + \gamma r_{t+2} + \gamma^2 + \cdots + \gamma^{n-1} r_{t+n} + \gamma^n V_t(s_{t+n}).$$

The value V(S) can be updated following n steps from S by:

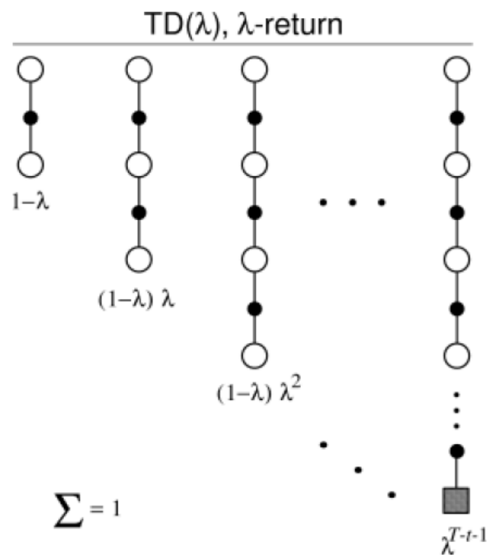
$$\Delta V_t(s_t) = \alpha \left[R_t^{(n)} - V_t(s_t) \right],$$

Generalizes the 1-step learning :

$\Delta V_t(S_t) = \alpha[r_{t+1} + \gamma V_t(S_{t+1}) - V_t(S_t)]$

Averaging trajectories:

- It is also possible to average trajectories; we can use the sub-trajectories of the full length-n trajectory to update V(S).
- A particular averaging (particular weights) is the TD(λ) weights:
- The weights are 1, λ, λ^2,… with all this multiplied by (1-λ) since a weighted average needs the sum of weights to be 1.
-



λ – Return

Using the single long trajectory we had:

$$R_t^{(n)} = r_{t+1} + \gamma r_{t+2} + \gamma^2 + \cdots + \gamma^{n-1} r_{t+n} + \gamma^n V_t(s_{t+n}).$$

The λ –return is the weighted average of all lengths:

$$R_t^\lambda \quad = \quad (1-\lambda) \sum_{n=1}^{T-t-1} \lambda^{n-1} R_t^{(n)} \quad + \quad \lambda^{T-t-1} R_t.$$

TD(λ)

And the learning rules:

Singe long trajectory:

$$\Delta V_t(s_t) = \alpha\Big[R_t^{(n)} - V_t(s_t)\Big],$$

TD(λ) learning:

$$\Delta V_t(s_t) = \alpha\Big[R_t^{\lambda} - V_t(s_t)\Big].$$

Eligibility traces

TD(λ) learning:

$$\Delta V_t(s_t) = \alpha \Big[R_t^{\lambda} - V_t(s_t) \Big].$$

To compute this at time t, we need the n next steps which we still do not have.
We want at time t to update back, the previous n visited states.
This can be done with '*eligibility trace*

Each visited state becomes 'eligible' for update, updates take place later:

Implementing TD(λ) with Eligibility Traces

A memory called 'eligibility trace' is added to each state $e_t(S)$
It is updated by:

$$e_t(s) = \begin{cases} \gamma\lambda e_{t-1}(s) & \text{if } s \neq s_t; \\ \gamma\lambda e_{t-1}(s) + 1 & \text{if } s = s_t, \end{cases} \tag{7.5}$$

The trace of S is incremented by 1 when S is visited, and decays by γλ at each step. Here γ is the discount factor and λ is the decay parameter.

Learning with eligibility traces

Take a step, compute a singel-step TD error:

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

Update V(S):

$$\Delta V_t(s) = \alpha \delta_t e_t(s), \qquad \text{for all } s \in \mathcal{S}.$$

V(S) is updated at each step, although the current step is different. If S was visited, then S1, S2, S3, then V(S) will be updated with the error of each of them. δ

The full TD(λ) Algorithm:

Initialize $V(s)$ arbitrarily and $e(s) = 0$, for all $s \in \mathcal{S}$
Repeat (for each episode):
    Initialize s
    Repeat (for each step of episode):
        $a \leftarrow$ action given by π for s
        Take action a, observe reward, r, and next state, s'
        $\delta \leftarrow r + \gamma V(s') - V(s)$
        $e(s) \leftarrow e(s) + 1$
        For all s:
            $V(s) \leftarrow V(s) + \alpha\delta e(s)$
            $e(s) \leftarrow \gamma\lambda e(s)$
        $s \leftarrow s'$
    until s is terminal

V(S) is updated at each step, although the current step is different from S. If S was visited, then S1, S2, S3, then V(S) will be updated with the error of each of them.

Eligibility traces

Updating state values V(S) by eligibility traces is mathematically identical to the 'forward' TD(λ) learning:

$$\Delta V_t(s_t) = \alpha\Big[R_t^{\lambda} - V_t(s_t)\Big].$$

The update does not rely on future values, and has plausible biological models.

SARSA (λ)

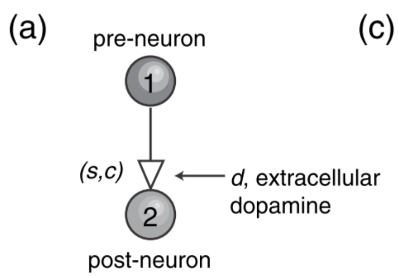
Initialize $Q(s,a)$ arbitrarily and $e(s,a) = 0$, for all s, a
Repeat (for each episode):
    Initialize s, a
    Repeat (for each step of episode):
        Take action a, observe r, s'
        Choose a' from s' using policy derived from Q (e.g., ε-greedy)
        $\delta \leftarrow r + \gamma Q(s',a') - Q(s,a)$
        $e(s,a) \leftarrow e(s,a) + 1$
        For all s, a:
            $Q(s,a) \leftarrow Q(s,a) + \alpha\delta e(s,a)$
            $e(s,a) \leftarrow \gamma\lambda e(s,a)$
        $s \leftarrow s'$; $a \leftarrow a'$
    until s is terminal

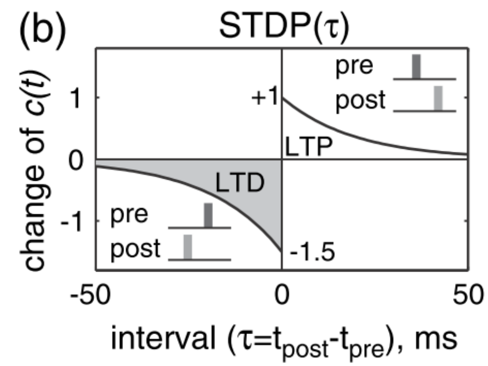
Eligibility traces – biology

Cerebral Cortex October 2007;17:2443-2452
doi:10.1093/cercor/bhl152
Advance Access publication January 13, 2007

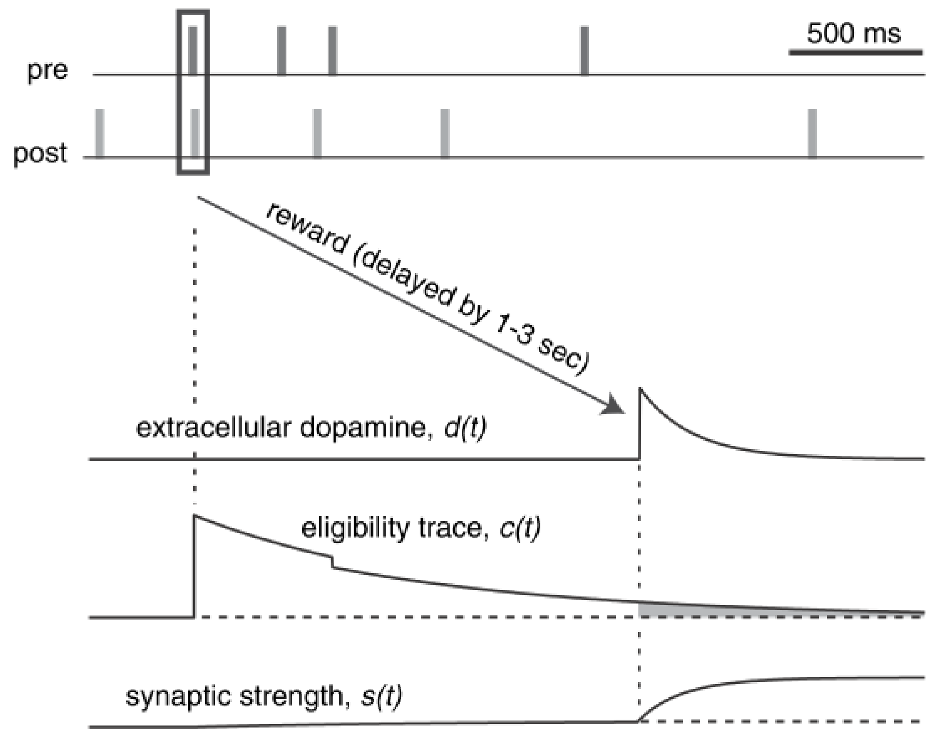
Solving the Distal Reward Problem through Linkage of STDP and Dopamine Signaling

SDTP

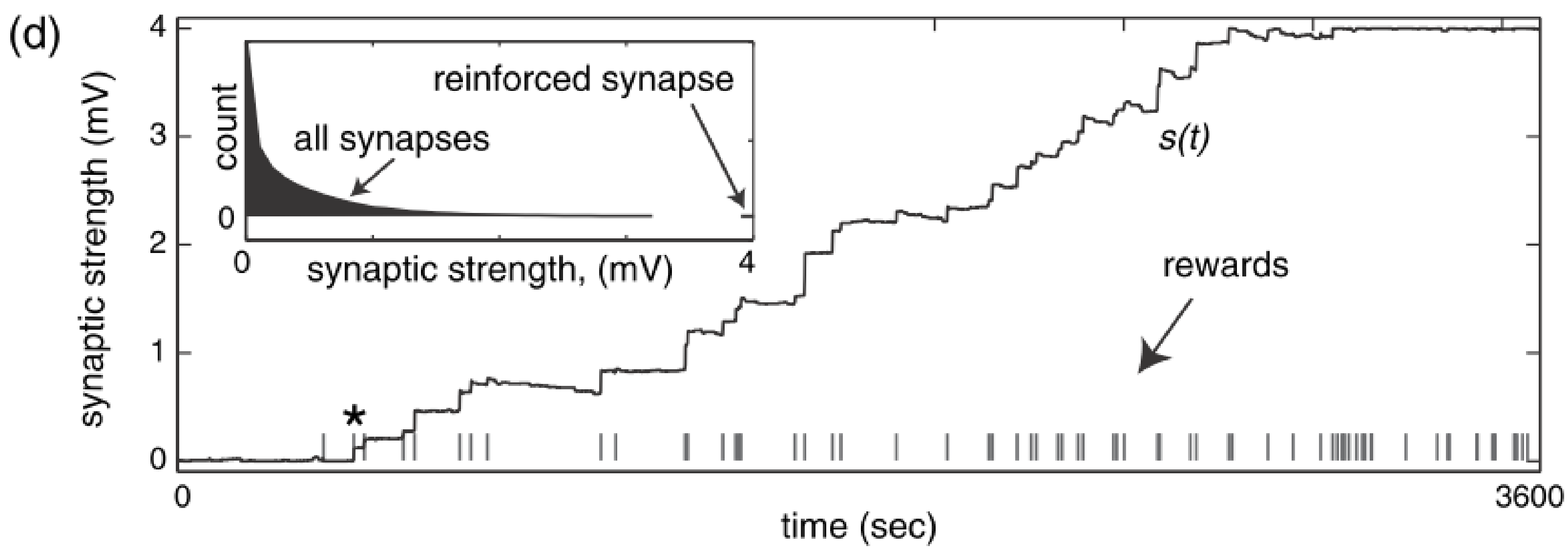

(a)
pre-neuron
1
(s,c)
d, extracellular dopamine
2
post-neuron
(c)


(b)
STDP(τ)
change of c(t)
pre
post
+1
LTP
LTD
-1.5
1
0
-1
-50
0
50
interval (τ=t_post-t_pre), ms

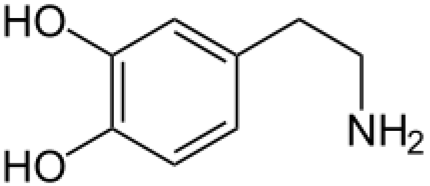
Eligibility



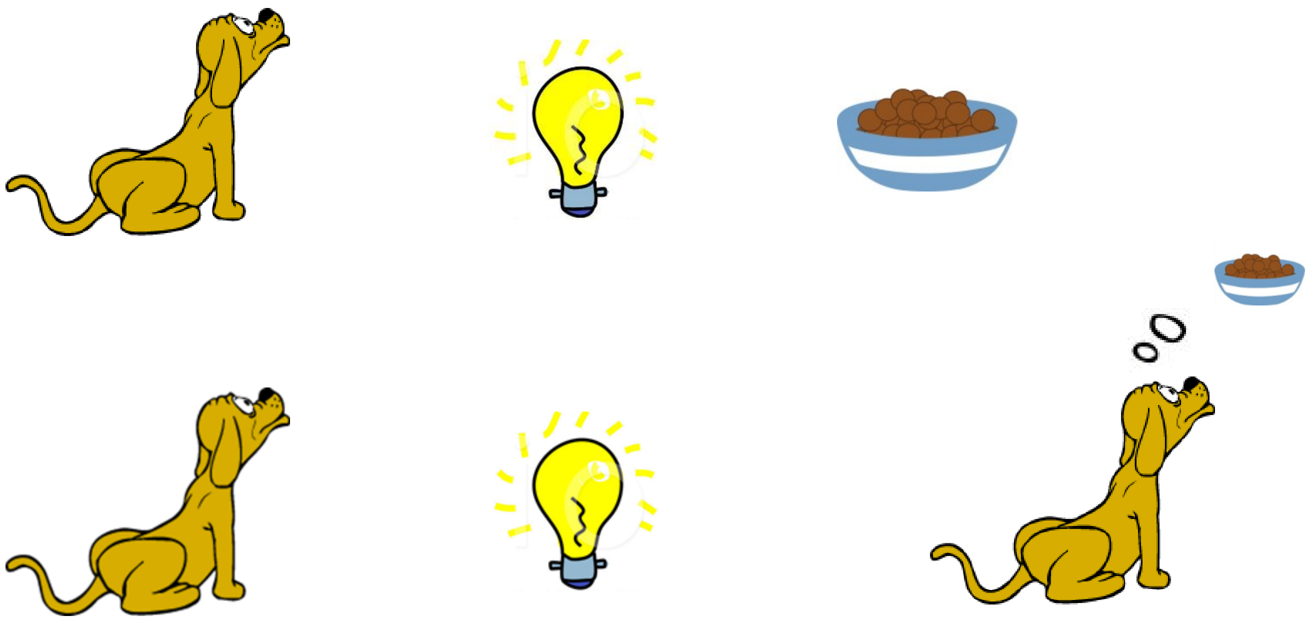
Synaptic Reinforcement



Dopamine story

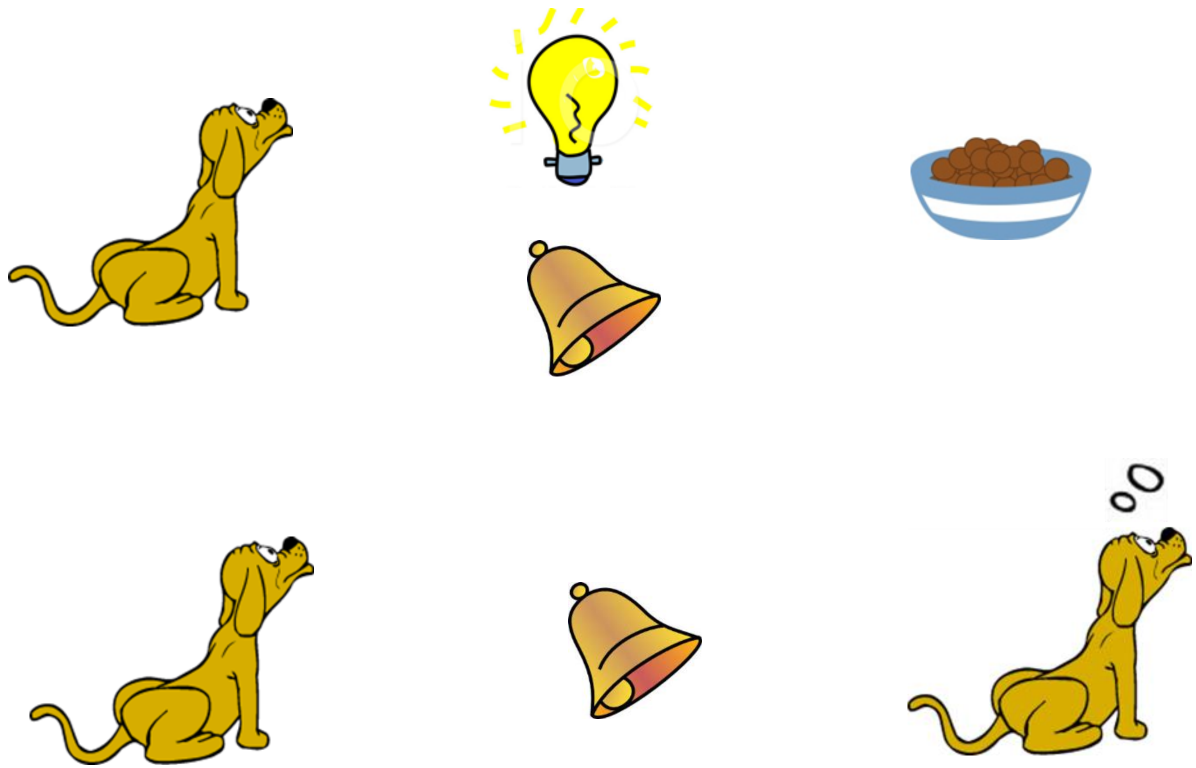


Behavioral support for 'prediction error'



Associating light cue with food

‘Blocking’



No response to the bell
The bell and food were consistently associated
There was no prediction error,
prediction error, not association, drives learning

Rescola - Wagner

Associative learning occurs not because two events co-occur but because that co-occurrence is unanticipated on the basis of current associative strength.

$$\Delta V_X^{n+1} = \alpha_X \beta(\lambda - V_{tot})$$

and

$$V_{tot} = V_X^n + \Delta V_X^{n+1}$$

A, β are rate parameters. V_{tot} is the total association from all cues on this trial. λ is the currently expected value. Learning occurs if the current value V_{tot} is different from expectation.

Still no action selection, policy for behavior, long sequences

Iterative solution for V(S)

$$V_\pi(S) = < r_1 + \gamma\, V_\pi(S') >$$

$$V(S) \leftarrow V(S) + \alpha\, [\, (r + \gamma V(S')) - V(S)\,]$$

Error

$$\delta(t) = r(t) + \gamma \hat{V}(t+1) - \hat{V}(t)$$

Prediction error, TD error

- Learning is driven by the prediction error:
- $\delta(t) = r + \gamma V(S')) - V(S)$

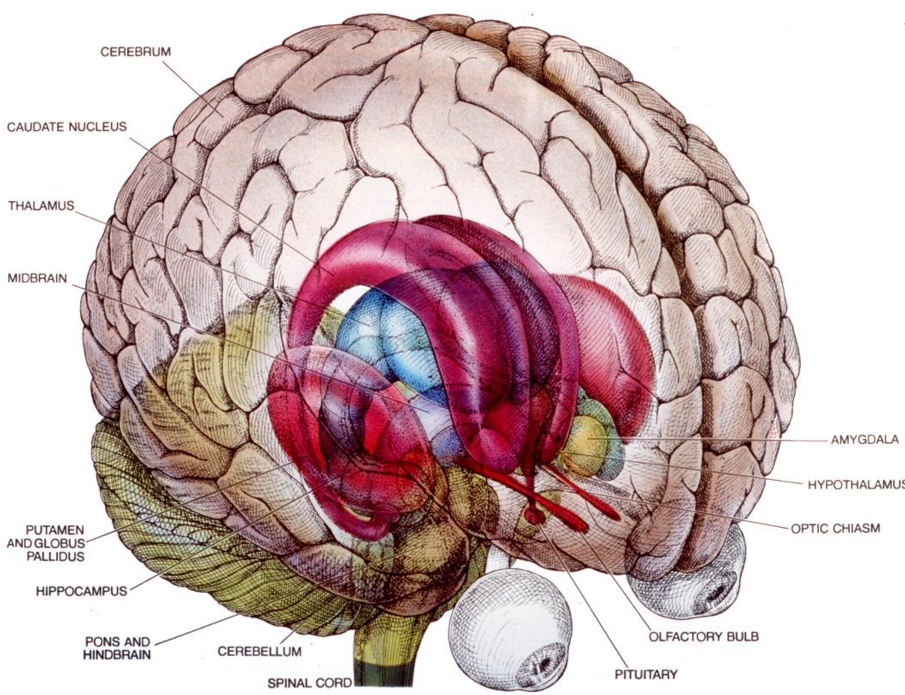
- Computed by the dopamine system

- (Here too, if there is no error, no learning will take place)

Domaminergic neurons

- Dopamine is a neuro-modulator
- In the:
- VTA (ventral tegmental area)
- Substantia Nigra
- These neurons send their axons to brain structures involved in motivation and goal-directed behavior, for example, the striatum, nucleus accumbens, and frontal cortex.

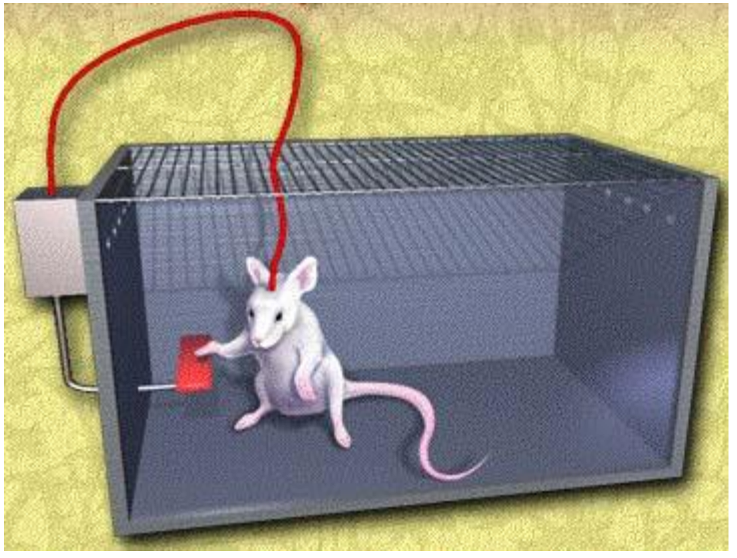
Major players in RL



Effects of dopamine, why it is associated with reward and reward related learning

- drugs like amphetamine and cocaine exert their addictive actions in part by prolonging the influence of dopamine on target neurons
- Second, neural pathways associated with dopamine neurons are among the best targets for electrical self-stimulation.
- animals treated with dopamine receptor blockers learn less rapidly to press a bar for a reward pellet

Self stimulation

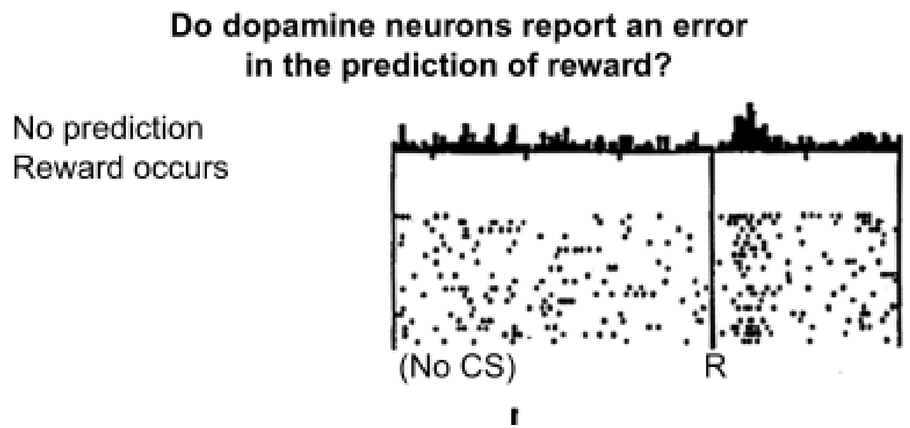


- You can put a stimulating electrode in various places. In the Dopamine system (e.g. VTA), the animal will continue stimulating.

- In the Orbital cortex for example you can put the electrode in a taste-related sub-region, activated by food. The animal will stimulate the electrode when it is hungry, but will stop activating when he is not.

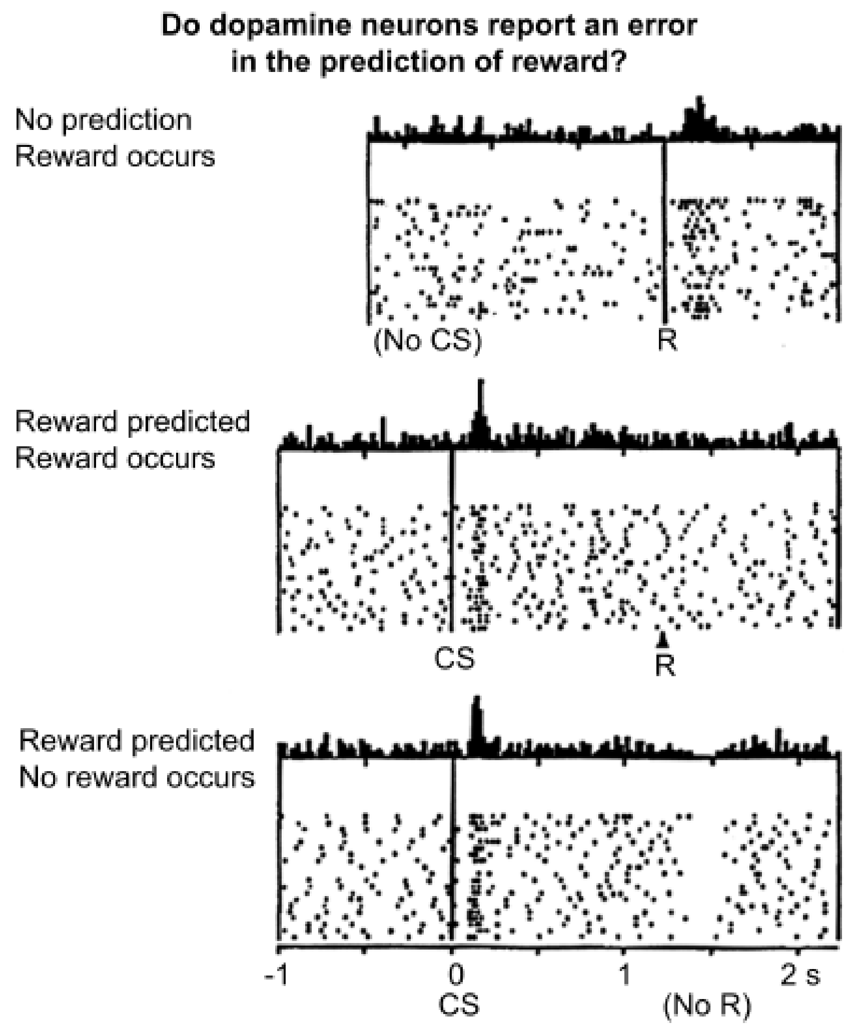
Dopamine and prediction error

The animal (rat, monkey) gets a cue (visual, or auditory).
A reward after a delay (1 sec below)



A Neural Substrate of Prediction and Reward
Wolfram Schultz *et al.*
Science **275**, 1593 (1997);

Dopamine and prediction error



TD, prediction error
Conclusion of the biological study

$$\delta(t) = r(t) + \gamma\hat{V}(t + 1) - \hat{V}(t)$$

This $\delta(t)$ is called the TD error and acts as a surrogate prediction error signal that is instantly available at time $t + 1$. As described below, $\delta(t)$ is used to improve the estimates of $V(t)$ and also to choose appropriate actions.

Computational TD learning is similar:

Take a step, compute a TD error:

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

Update V(S):

$$\Delta V_t(s) = \alpha \delta_t e_t(s), \qquad \text{for all } s \in \mathcal{S}. \tag{7.7}$$

V(S) is updated at each step, although the current step is different. If S was visited, then S1, S2, S3, then V(S) will be updated with the error of each of them. δ