Lecture 10
Temporal-difference learning

Jochen Braun

Otto-von-Guericke-Universität Magdeburg,
Cognitive Biology Group

Theoretical Neuroscience II, SS 2020

Credits:

Dayan & Abbott (2001) "Theoretical Neuroscience", Chapter 9

Sutton & Barto (1998) "Reinforcement Learning", Chapters 3-7

10. Temporal-difference learning (TDL)

Extend reinforcement from rewards to changing expectations ('temporal difference'), learning to expect **delayed** *rewards and to actively procure them ($\rightarrow$ next lecture).*

TDL for episodic tasks: *Learn to associate events* $\boldsymbol{u}$ *with delayed reward expectations* $\boldsymbol{w}(\tau)$*. Focus on total reward (future and present), not just present reward. TDL explains 'trace' and 'secondary' conditioning. Raised and lowered expectations (promises & threats) become equivalent to rewards and punishments.*

TDL for continuous tasks: *Illustrate generalized framework with 'gridworld', where 'agent' seeks exit. Learn to associate expected total future reward* $V_\pi(s)$ *with states* s *by accumulating value function (VF). We compute theoretical VF and compare VF obtained by TDL. For active agent, VF depends on action policy* $\pi(a)$*. Learning this action policy is subject of next lecture.*

Organization of lecture

- 1 Rescorla-Wagner recap
- 2 TD learning for episodic tasks
- 3 Trace conditioning
- 4 Secondary conditioning
- 5 General formulation of reinforcement learning
- 6 Gridworld
- 7 TD learning for continuous tasks

1 Rescorla-Wagner recap

In the last lecture, we modeled reinforcement learning in terms of **external** stimuli and rewards ...

$$\textit{Stimuli} \qquad \boldsymbol{u}(t) = \begin{cases} 1 & \text{if present} \\ 0 & \text{if absent} \end{cases}$$

$$\textit{Rewards} \qquad r(t) \quad \textit{scalar value}$$

... and in terms of **internal** weights (reward expectations) associated with each stimulus

$$\textit{Conditional expectation} \qquad \boldsymbol{w}(t)$$

$$\textit{Total expected reward} \qquad v = \boldsymbol{w}(t) \cdot \boldsymbol{u}(t)$$

The experience gained by trial and error was captured by the **Rescorla-Wagner rule**:

$$\boldsymbol{w} \rightarrow \boldsymbol{w} + \epsilon\, \delta\, \boldsymbol{u}, \qquad\qquad \delta = r - v$$

where δ was the *prediction error* (surprise) and ϵ was a *learning rate*. Over time, weights converge to the conditional expected reward:

$$w_k \quad \rightarrow \quad \langle r | u_k \rangle$$

Intuitively, a *positive* surprise δ attaches the *merit* to all stimuli present, *increasing* their weights.

A *negative* surprise δ attaches the *blame* to all stimuli present, *decreasing* their weights.

RW summary

While the RW accounts for reinforcement learning when stimuli and rewards are in close temporal proximity ...

- Pavlovian conditioning.
- Extinction
- Partial conditioning
- Blocking
- Overshadowing

... it does not account for the ability of animals to anticipate delayed rewards:

- ~~Trace conditioning~~
- ~~Secondary conditioning~~

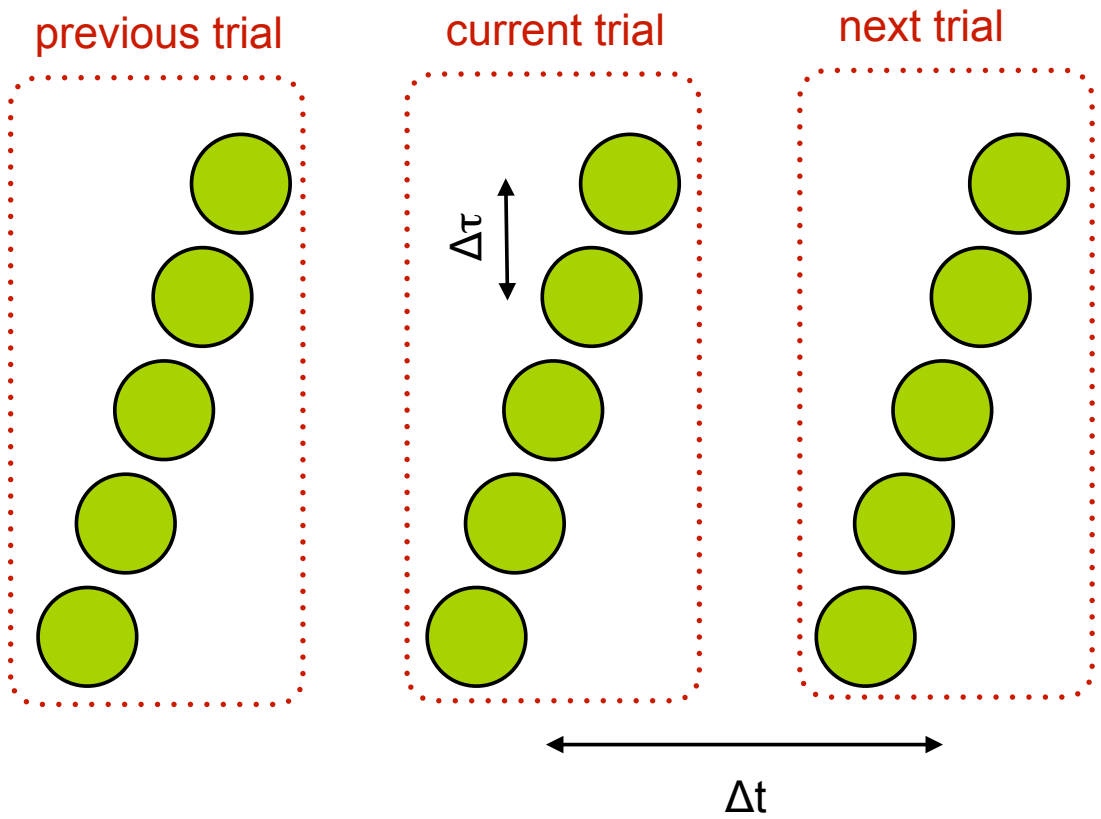
2 TD learning for episodic tasks

We now formulate a scheme for learning to expect delayed rewards ('temporal-difference learning' or 'TDL'). The crucial modifications will be:

- Learn to predict *total reward* (present and future), not just *reward* (present). *Total reward* is an expectation of reinforcement, rather than reinforcement itself.
- Form prediction error δ from *temporal difference* of expectation of reinforcement.
- Rising expectations (promises) become equivalent to rewards, falling expectations (threats) to punishments.
- To associate observed stimuli $\mathbf{u}$ with *delayed* rewards ...
- ... learn weights $\mathbf{w}(\tau)$ that are functions of delay τ.

Episodic task: coarse t vs. fine τ

From trial to trial, 'coarse' time t increases in steps of Δt. Within each trial, 'fine' time τ grows in steps of $\Delta\tau$, up to the trial duration τ_{max}. The 'foresight' of our scheme will end at τ_{max}.



ctd

Both **external** and **internal** state variables depend on within-trial time τ and will be updated between trials i

Stimuli	$\boldsymbol{u}_i(\tau)$,	τ *time*
Rewards	$r_i(\tau)$	
Conditional expect.	$\boldsymbol{w}_i(\Delta\tau)$,	$\Delta\tau$ *delay*
Total expect.	$v_i(\tau) = \sum_{\Delta\tau=0}^{\tau} \boldsymbol{w}_i(\Delta\tau) \cdot \boldsymbol{u}_i(\tau - \Delta\tau)$	

where i is the trial index. v_i sums up all promises from the past, for now and for the future.

TD learning rule

Define $v_i(\tau)$ as *total* future reward, expected from τ to τ_{max}:

$$v_i(\tau) = \left\langle \sum_{\tau'=\tau}^{\tau_{max}} r(t, \tau') \right\rangle$$

Our goal is to compute this from *past* events and *current* rewards in the usual manner:

total expectation

$$v_i(\tau) = \sum_{\Delta\tau=0}^{\tau} \boldsymbol{w}_i(\Delta\tau) \cdot \boldsymbol{u}_i(\tau - \Delta\tau)$$

specific expecations

$$\boldsymbol{w}_{i+1}(\tau) = \boldsymbol{w}_i(\tau) + \epsilon\, \delta(\tau)\, \boldsymbol{u}_i(\tau)$$

prediction error

$$\delta(\tau) = -v_i(\tau) + \sum_{\tau'=\tau}^{\tau_{max}} r_i(\tau')$$

ctd

At time τ, we know only present $r_i(\tau)$, not future $r_i(\tau' > \tau)$.

$$\delta(\tau) = -v_i(\tau) + r_i(\tau) + \sum_{\tau' > \tau} r_i(\tau')$$

We cannot use this directly! Instead, we 'bootstrap' and assume

$$\sum_{\tau' > \tau} r_i(\tau') \approx v_i(\tau + \Delta\tau)$$

basing prediction error on 'changing expectations'

$$\delta(\tau) \approx -v_i(\tau) + r_i(\tau) + v_i(\tau + \Delta\tau) = r_i(\tau) + \underbrace{\Delta v_i(\tau)}_{\text{changing expectations}}$$

ctd

The TD learning scheme thus involves *summing expectations* $\boldsymbol{w}(\tau)$

$$v_i(\tau) \quad = \quad \sum_{\Delta\tau=0}^{\tau} \boldsymbol{w}_i(\Delta\tau) \cdot \boldsymbol{u}_i(\tau - \Delta\tau)'$$

updating expectations

$$\boldsymbol{w}_i(\Delta t, \tau) \quad = \quad \boldsymbol{w}_i(\tau) + \epsilon\, \delta(\tau)\, \boldsymbol{u}_i(\tau)'$$

and the *temporal difference prediction error*

$$\delta(\tau) \quad = \quad r_i(\tau) + \Delta v_i(\tau), \qquad\qquad \Delta v_i(\tau) = v_i(\tau + \Delta\tau) - v_i(\tau).$$

Positive (negative) prediction errors at time τ arise either from actual rewards (punishments) $r_i(\tau)$ or, equivalently, from rising (lowering) expectations $\Delta v_i(\tau)$!

Points to note

TD learning for episodic tasks involves:

- Redefining the expected reward $v_i(\tau)$ as the *total future reward* expected from τ to τ_{max}.
- Weights $\boldsymbol{w}(\Delta\tau)$ predict when and how much expectations change!
- Computing the prediction error $\delta(\tau)$ as the sum of any reinforcement $r_i(\tau)$ and of changing expectations $\Delta v_i(\tau)$.
- Thus, in TD learning, rising hopes are equivalent to rewards, and falling hopes are equivalent to punishments.

- Psychologically plausible: motivating effect of promises, demotivating effect of threats!

3 Trace conditioning

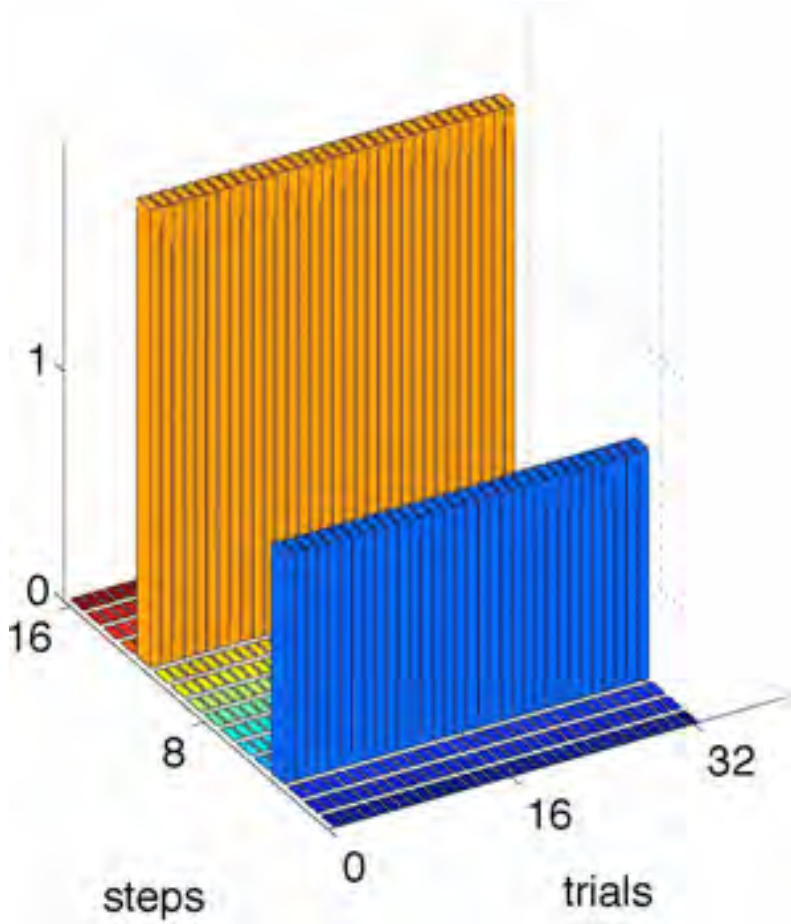
We apply TD learning to "trace conditioning", or learning to anticipate rewards that are delayed relative to the associated stimulus.

For simplicity, we consider identical trials in which stimuli and rewards always occur at the same within-trial time τ.

Note that, as long as the delay $\Delta\tau$ between stimulus and reward remains the same, we would obtain equivalent results for variable within-trial times.

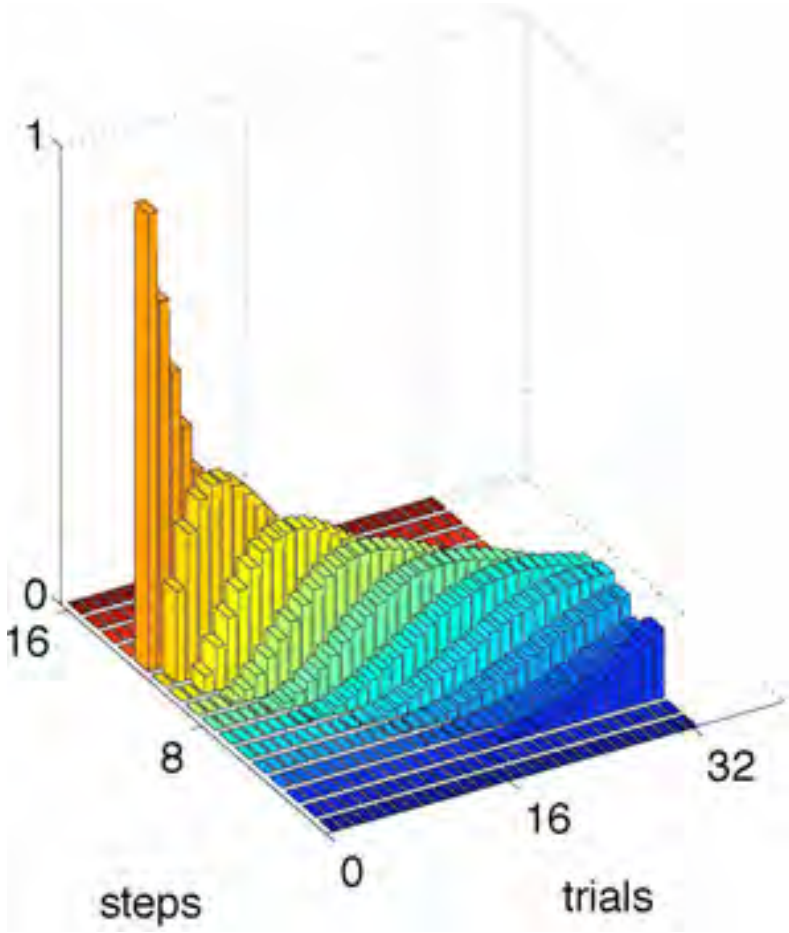
32 trials trace conditioning

Stimuli u (blue) and **rewards** r (orange)



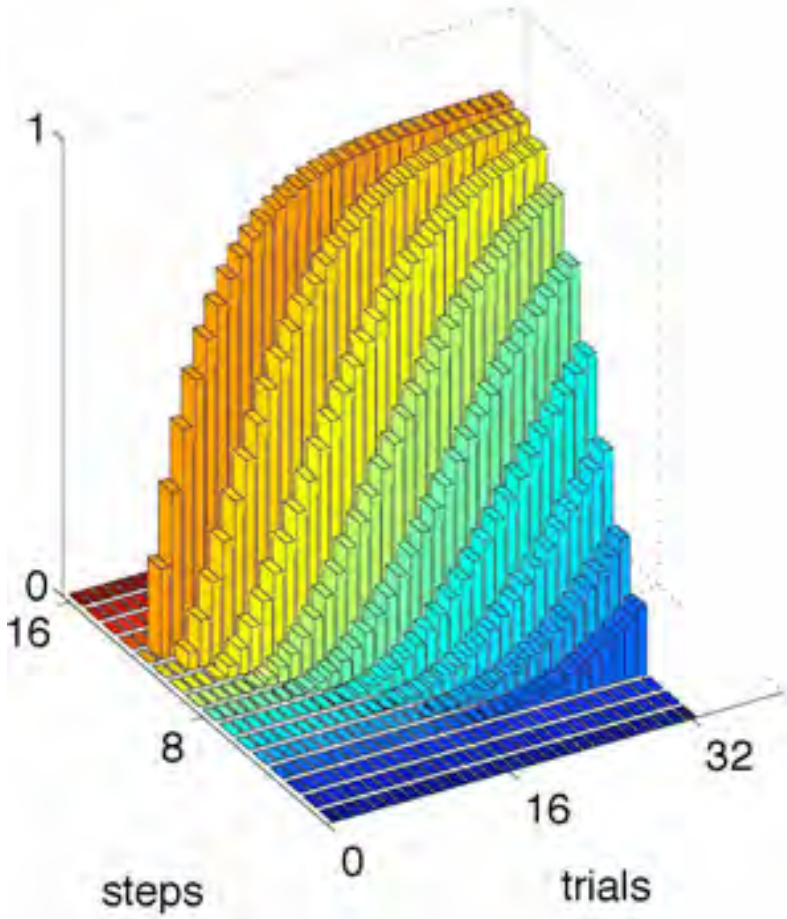
Every trial has a stimulus u (blue) at $\tau = 4$ and a reward r (orange) at $\tau = 12$.

Prediction error $\delta(\tau)$



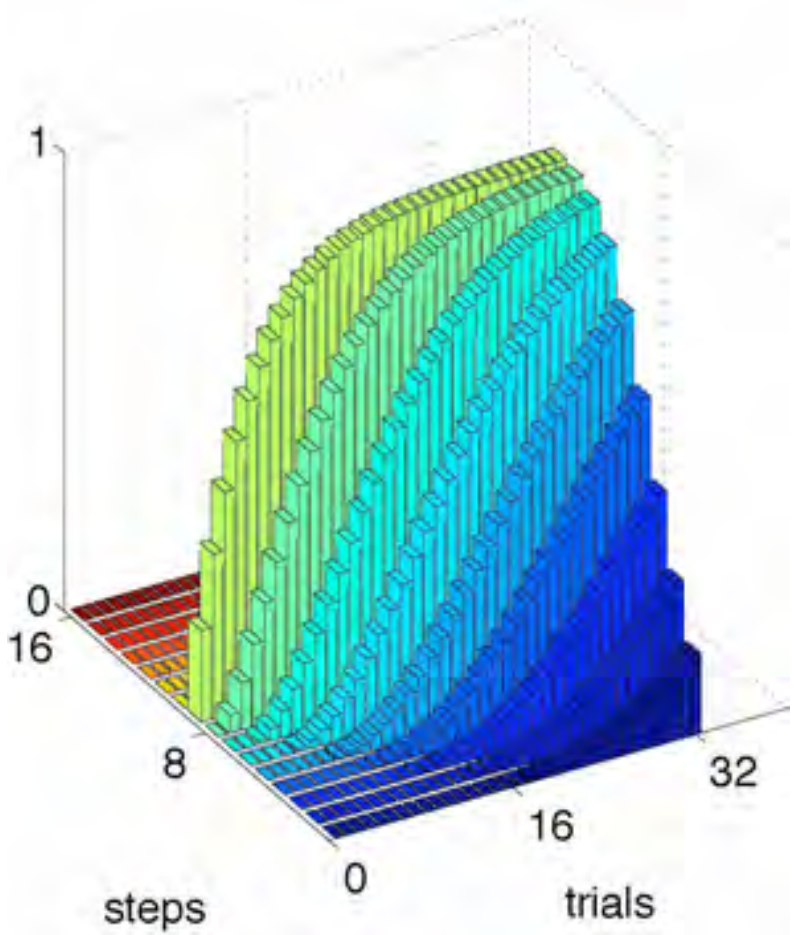
is reward $r(\tau)$ plus changed expectation $\Delta v(\tau)$. Every $\delta(\tau)$ increments delayed expectation $w(\tau - \tau')$ for earlier stimuli $u(\tau')$. This gradually transfers prediction error from $\tau = 12$ to $\tau = 4$.

Expected total future reward $v(\tau)$



is obtained by summing up delayed expectation weights $w(\tau)$ for earlier stimuli $u(\tau')$. Total future reward gradually rises between $\tau = 4$ and $\tau = 12$.

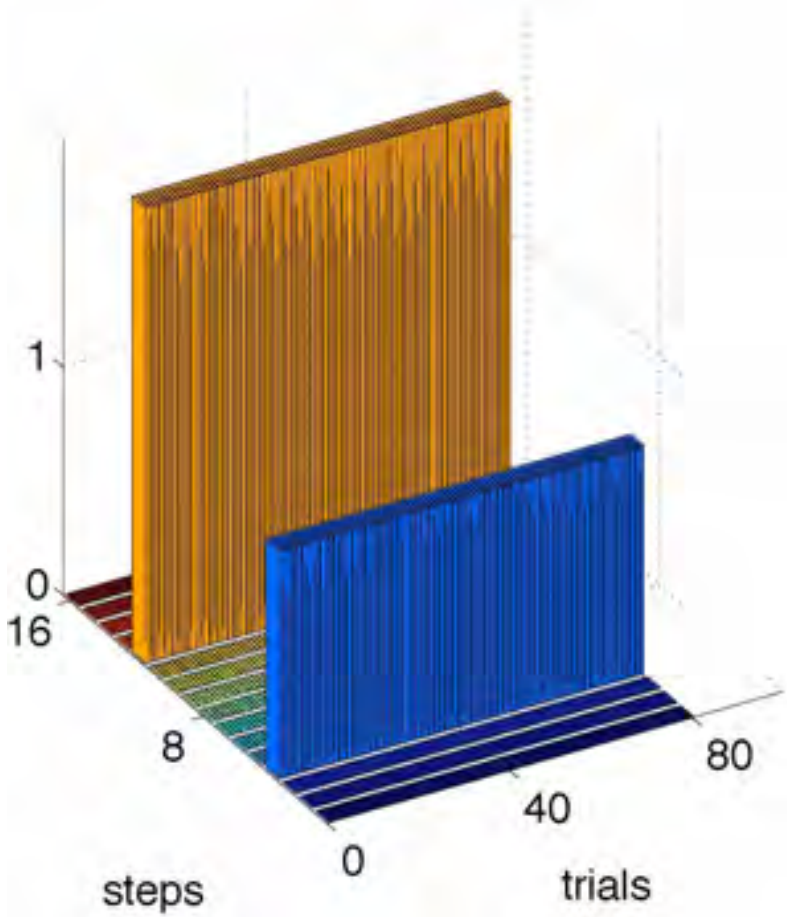
Delayed expectation **weights** $w(\Delta\tau)$:



are incremented for earlier stimuli $u(\tau')$, whenever there is a prediction error $\delta(\tau)$ (as already mentioned). Weights gradually rise between $\Delta\tau = 0$ and $\Delta\tau = 8$.

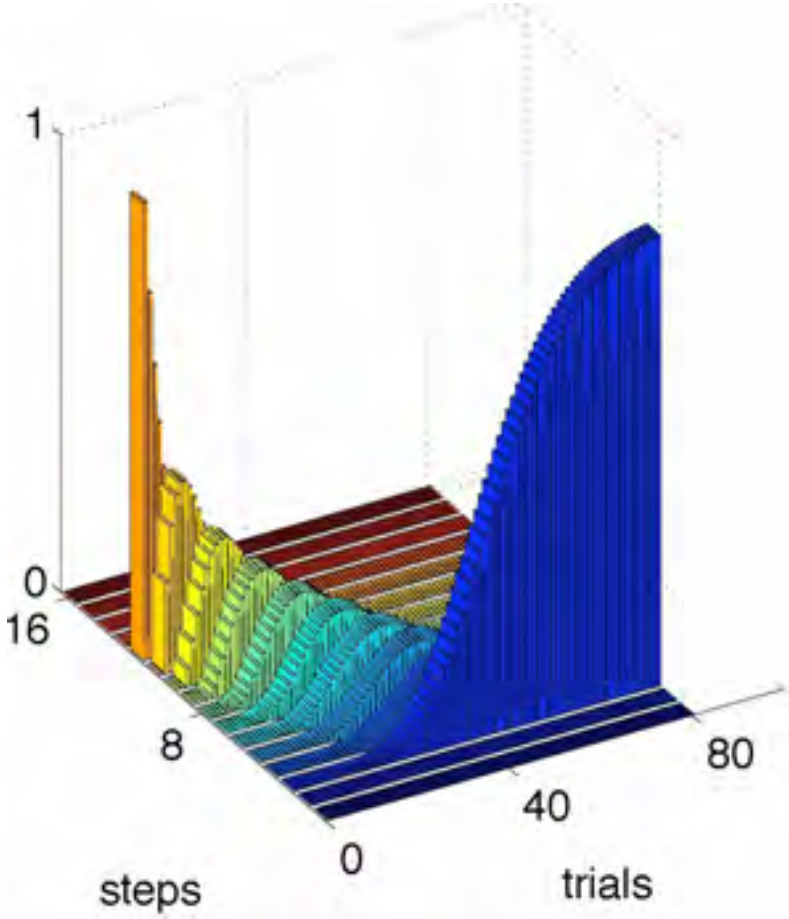
80 trials trace conditioning

Stimuli u (blue) and **rewards** r (orange):



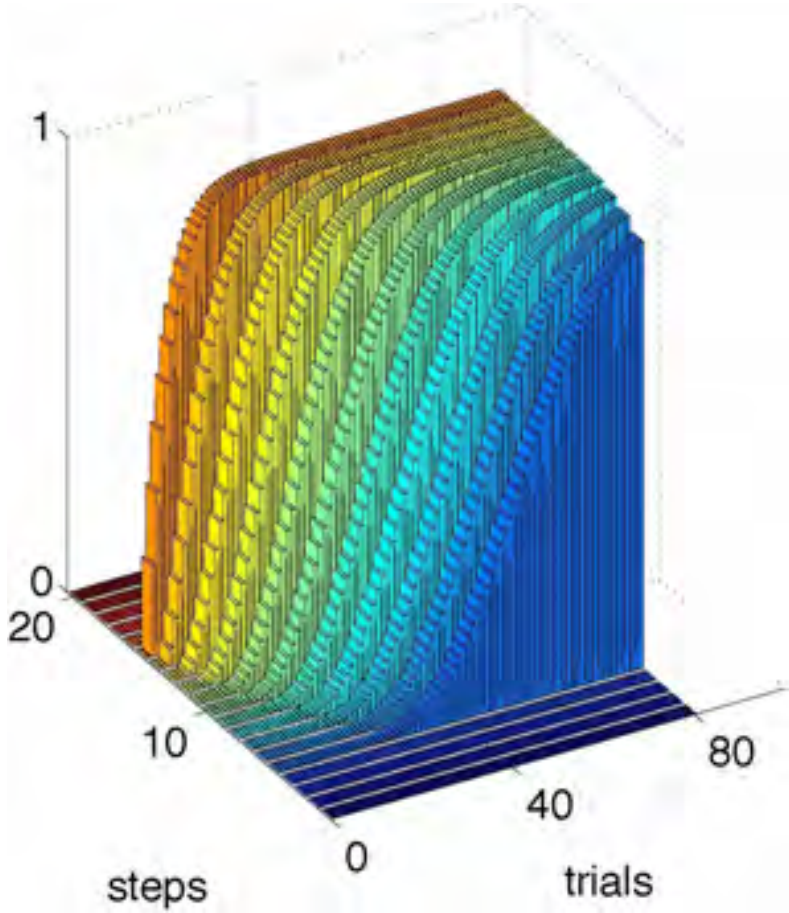
Every trial brings a stimulus at $\tau = 4$ and a reward at $\tau = 12$.

Prediction error $\delta(\tau)$



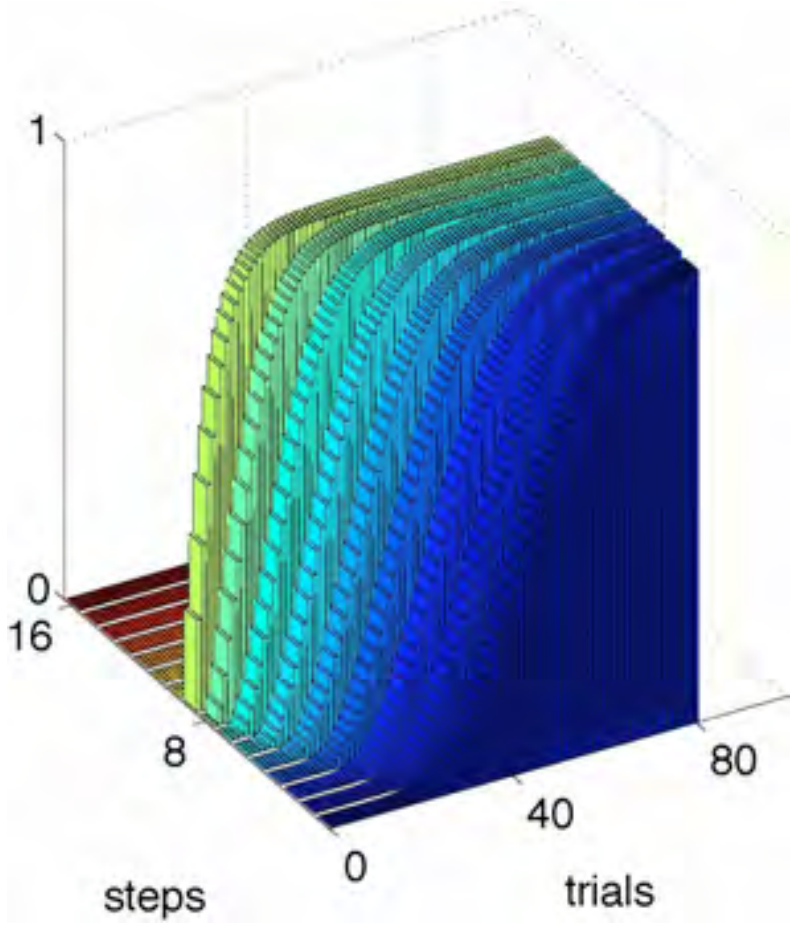
The prediction error is transferred from $\tau = 12$ to $\tau = 4$.

Expected total future reward $v(\tau)$



Total future reward rises between $\tau = 4$ and $\tau = 12$.

Delayed expectation **weights** $w(\Delta\tau)$:



Weights rise between $\Delta\tau = 0$ and $\Delta\tau = 8$.

Points to note

Trace conditioning involves:

- Rewards increase delayed weights, creating rising expectation next time round.
- Rising expectation increase less delayed weights.
- Repeats lead iteratively to an increase in immediate weights.
- Eventually, prediction error shifts from reward to predictive stimulus.

- Consistent with experimental observations on dopamine neurons!

4 Secondary conditioning

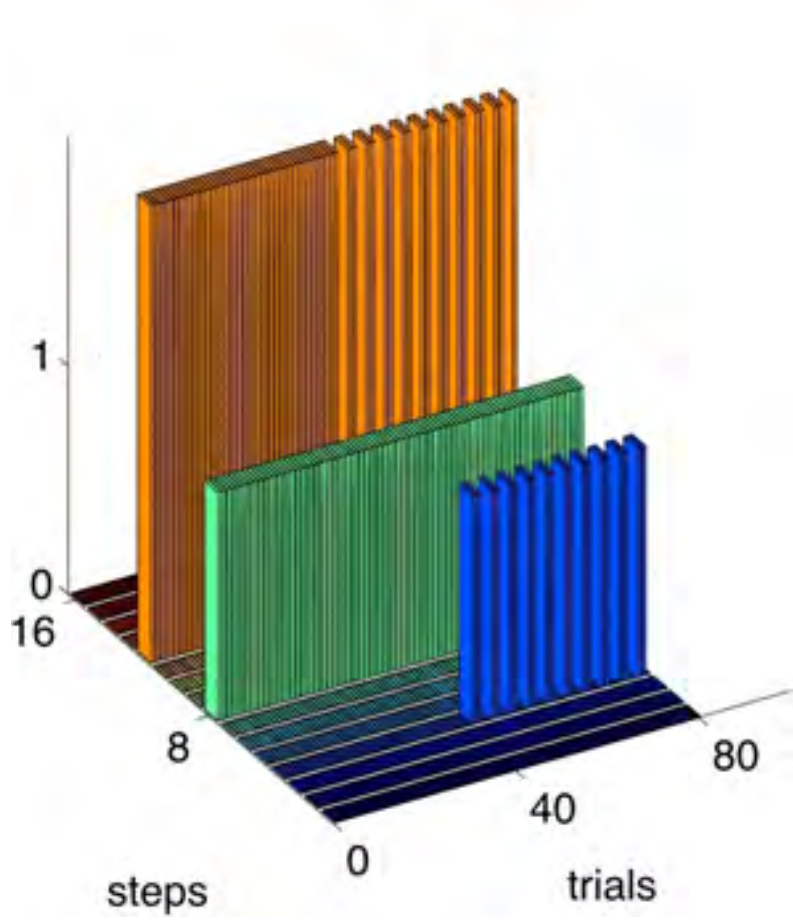
Secondary conditioning involves associating one stimulus with a reward, followed by associating a second stimulus with the first stimulus (but not with a reward!). In animals, the second stimulus evokes an expectation of reward (with which it has never been paired).

Thus, rewards are used to condition promises (first stimuli), which in turn condition promises of promises (second stimuli).

Note that this works only as long as there are not too many trials without reward (so that not all expectations are extinguished).

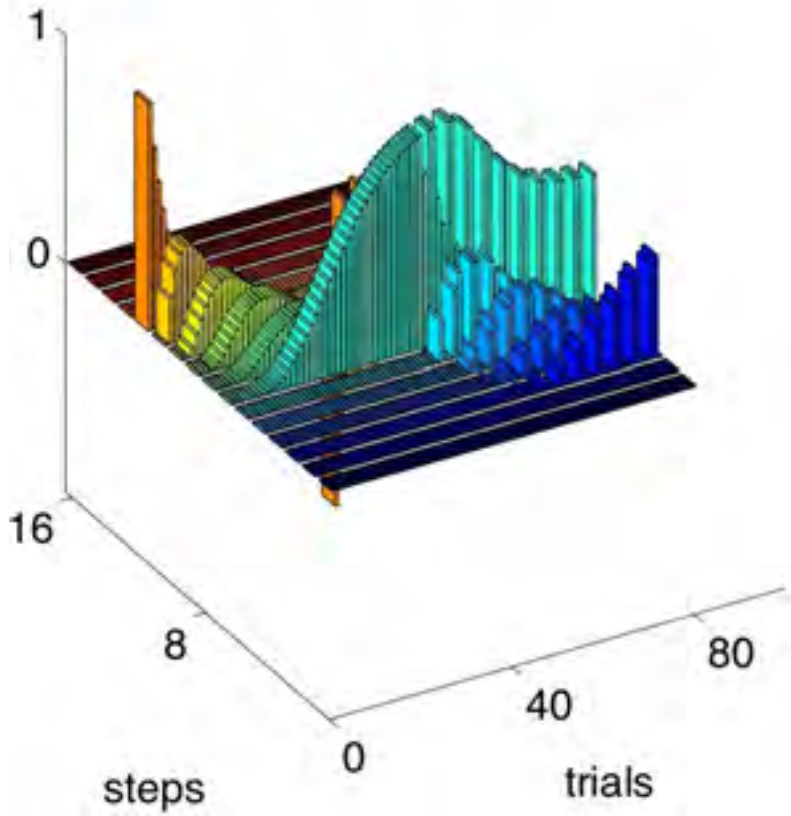
Trace and secondary conditioning (40 trials each)

First stimuli u_1 (green), **second stimuli** u_2 (blue) and **rewards** r (orange):



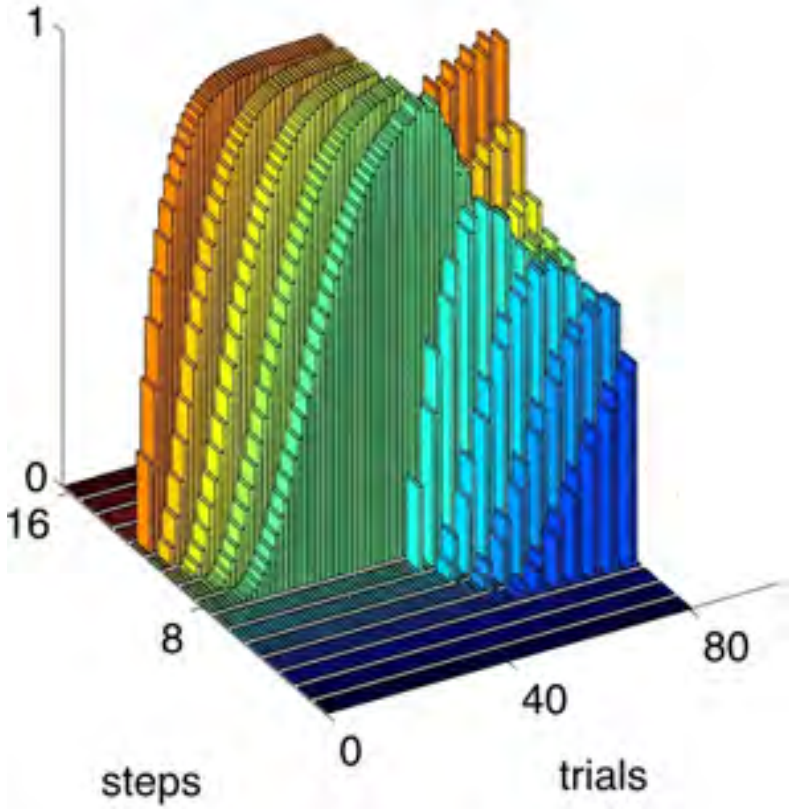
u_1 at $\tau = 8$, u_2 at $\tau = 4$, r at $\tau = 12$.

Prediction error $\delta(\tau)$



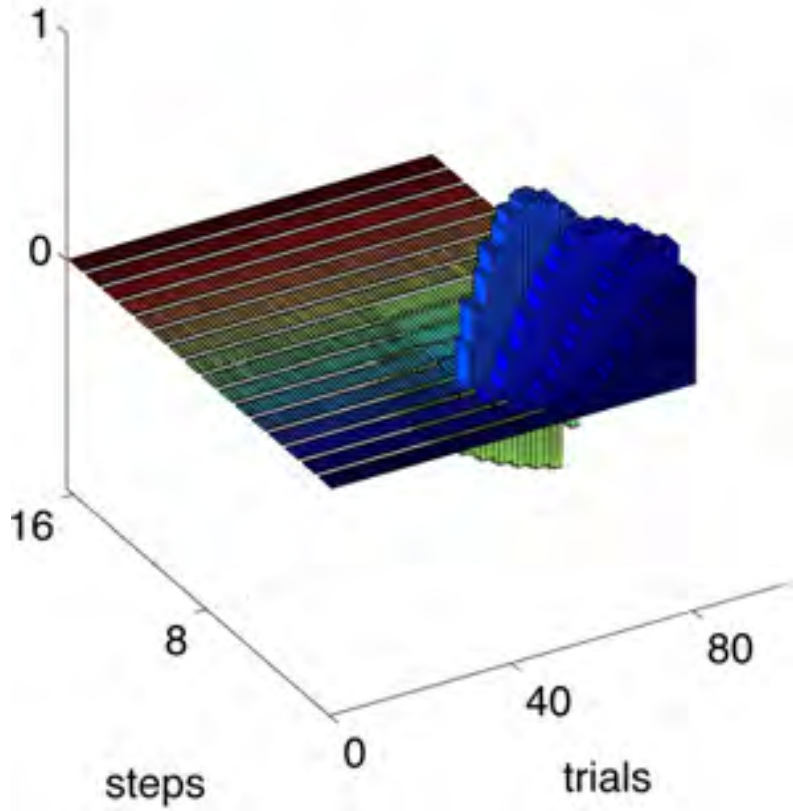
$\delta(\tau)$ transfers from $\tau = 12$ to $\tau = 8$ and then from $\tau = 8$ to $\tau = 4$.

Expected total future reward $v(\tau)$



The expected future reward rises first at $\tau = 8$ and later also at $\tau = 4$.

Delayed expectation **weights** $w(\Delta\tau)$:



The weights of the second stimulus are positive for $0 \leq \Delta\tau \leq 3$ and negative for $4 \leq \Delta\tau \leq 8$.

TD learning explains secondary conditioning because:

- The first stimulus evokes a rising expectation Δv lasting until the time of the reward.
- In terms of the prediction error $\delta = r + \Delta v$, this rising expectation is equivalent to an actual reward.
- If a second stimulus predicts this positive surprise, the second stimulus also also evokes a rising expectation lasting until the time of the first stimulus.
- Thereafter, the second stimulus evokes a falling expectation (due to the subsequent failure of reward).

Points to note

Main points of TD learning:

- From events $u(\tau)$, a future time course $w(\Delta\tau)$ is learned.
- It predicts total future rewards, from $\tau + \Delta\tau$ onwards.
- Rising reward expectations also act as rewards.
- Prediction error transferred from reward $\rightarrow$ primary predictor $\rightarrow$ secondary predictor.
- Endows agent with foresight about delayed rewards.

Appears psychologically plausible (e.g., promises & politics)

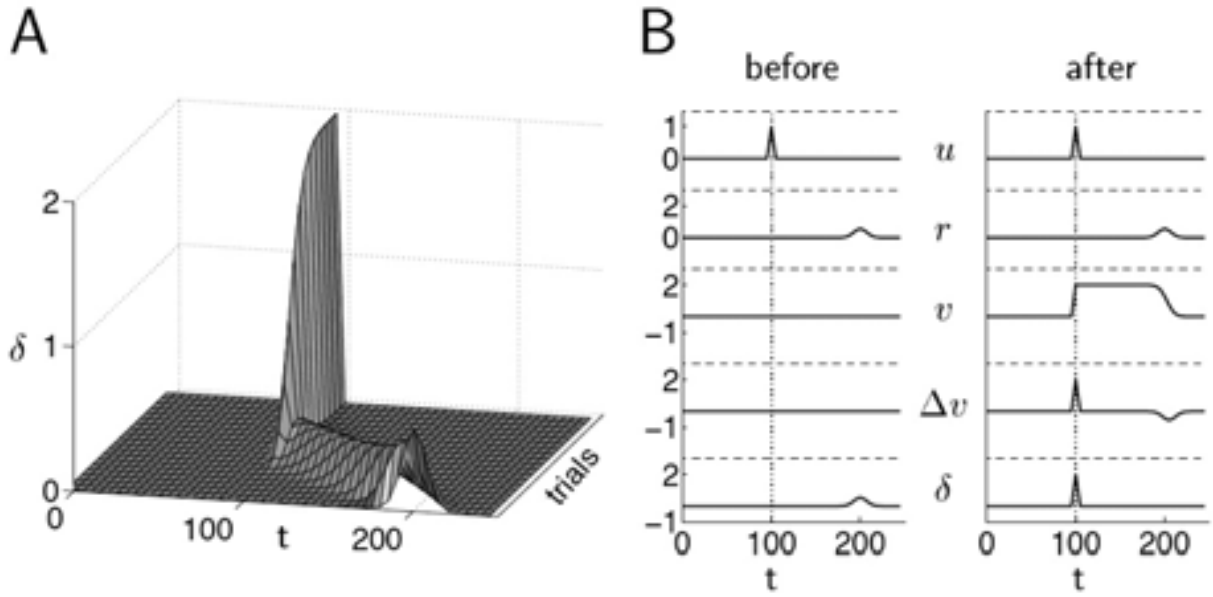
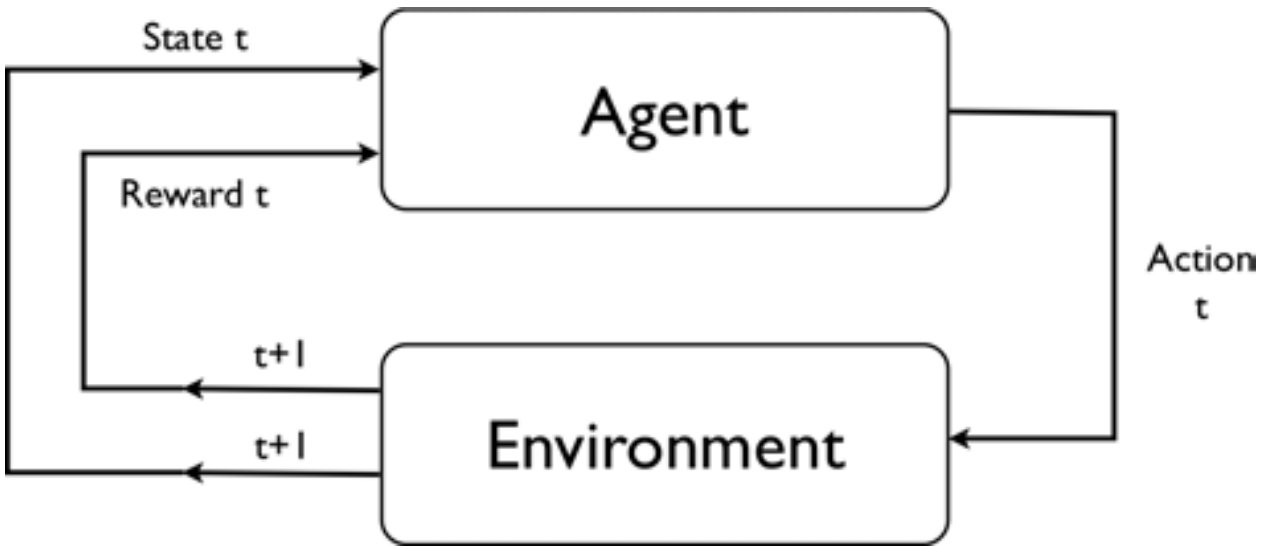


Figure 9.2: Learning to predict a reward. A) The surface plot shows the prediction error $\delta(t)$ as a function of time within a trial, across trials. In the early trials, the peak error occurs at the time of the reward ($t = 200$), while in later trials it occurs at the time of the stimulus ($t = 100$). (B) The rows show the stimulus $u(t)$, the reward $r(t)$, the prediction $v(t)$, the temporal difference between predictions $\Delta v(t-1) = v(t) - v(t-1)$, and the full temporal difference error $\delta(t-1) = r(t-1) + \Delta v(t-1)$. The reward is presented over a short interval, and the prediction v sums the total reward. The left column shows the behavior before training, and the right column after training. $\Delta v(t-1)$ and $\delta(t-1)$ are plotted instead of $\Delta v(t)$ and $\delta(t)$ because the latter quantities cannot be computed until time $t+1$ when $v(t+1)$ is available.

5 General formulation of reinforcement learning

Theories of reinforcement learning have developed a number concepts that apply also to biological learning.



In general, an agent in an environment experiences certain states s and reinforcements r and may respond with certain actions a.

Episodic and continuous tasks

"Expected return" is the total reward received in the long run. *Episodic* definition is modified for *continuous* tasks:

Episodic: sum until episode ends at τ_{max}

$$R_\tau = r_\tau + r_{\tau+1} + \ldots + r_{\tau_{max}}$$

Continuous: sum of discounted returns to infinity

$$R_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} \ldots = \sum_{k=0}^{\infty} \gamma^k \, r_{t+k}$$

where $\gamma < 1$ is a discounting factor. (Sooner rewards are more valuable than later rewards.)

Markov property

The "state" is the information available to the agent. Typically, "state" represents immediate stimuli, reward, and action. If required, the state may be extended to include memories of past states. Ideally, "state" is a compact representation of all relevant information, past or present.

General environment (depends on all states, actions ...):

$$P\{s_{t+1}, r_{t+1} | s_t, a_t, r_t, \ldots, r_0, s_0, a_0\}$$

Environment with *Markov property* (only current states, actions ...)

$$P\{s_{t+1}, r_{t+1} | s_t, a_t, r_t\}$$

'What matters is where I am now, not how I got here'.

Policy

A policy π is a mapping from states s to actions a:

$$\pi(a, s) \quad \text{probability of transition} \quad s \xrightarrow{a} s'$$

For now, we consider only the 'dumb' policy of acting randomly:

$$\pi(a, s) = \frac{1}{N_a(s)}$$

where $N_a(s)$ is the number of possible actions a at state s.

[What's *your* policy when you find yourself at Hasselbach Platz at 11 pm ? M2? Coco Bar? Flower Power? The Hyde?]

Value function

The value function $V_\pi(s)$ maps each state s to the associated expected return R_t:

$$V_\pi(s) = \langle R_t | s \rangle_\pi$$

Value function encapsulates expectations. How promising is state s with current policy π?

Value function summarizes all past experience, good or bad. The objective of reinforcement learning is to incrementally acquire this function.

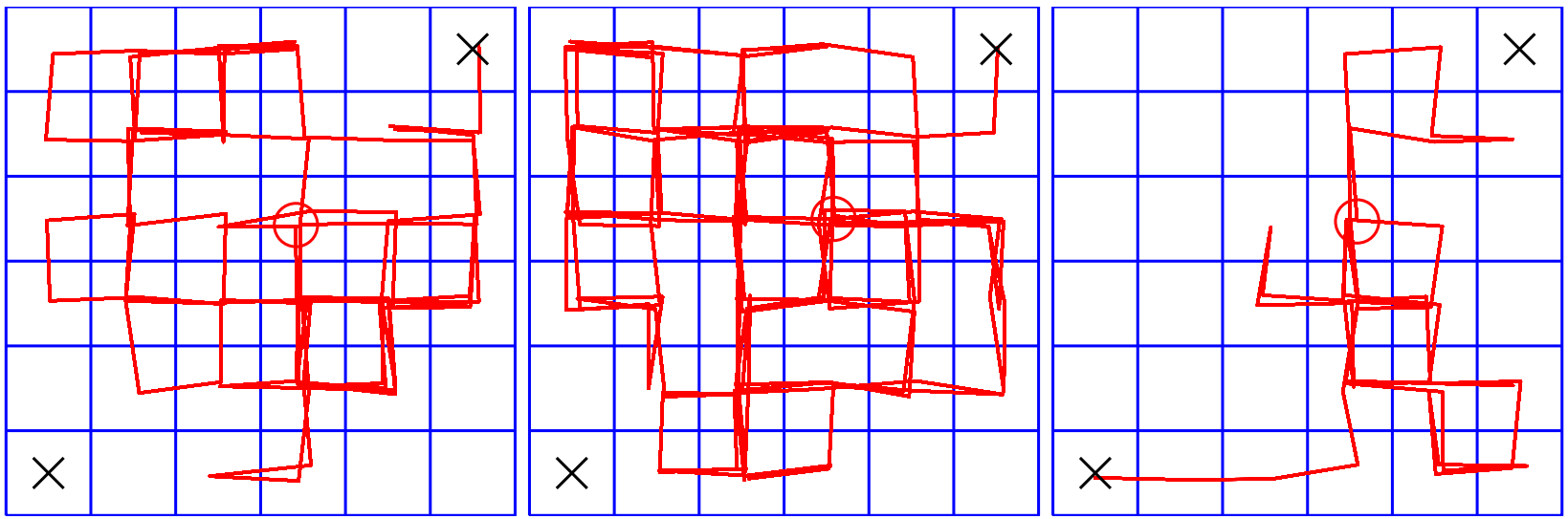
Value function depends strongly on policy, because expected reward depends strongly on possible future actions.

Points to note

- Reinforcement learning applies to *episodic* and *continuous* tasks.
- The *expected return* R_t is computed differently in each case.
- The available information at time t includes the current reward r_t, the current environment s_t, and the action a_t (about to be undertaken).
- If this information is sufficient to determine the next reward and state (r_{t+1}, s_{t+1}), the environment is said to have the *Markov property*.
- The objective of reinforcement learning is to acquire a *value function* $V_\pi(s)$, which specifies the expected return associated with each state.
- For an active agent, the value function depends on a *policy* $\pi(a, s)$, which specifies the probabilities of alternative actions a.

6 Gridworld

We illustrate these ideas with a "gridworld" of $N \times N$ states. In this world, an agent moves left, right, up, or down until it accidentally finds one of two exits.

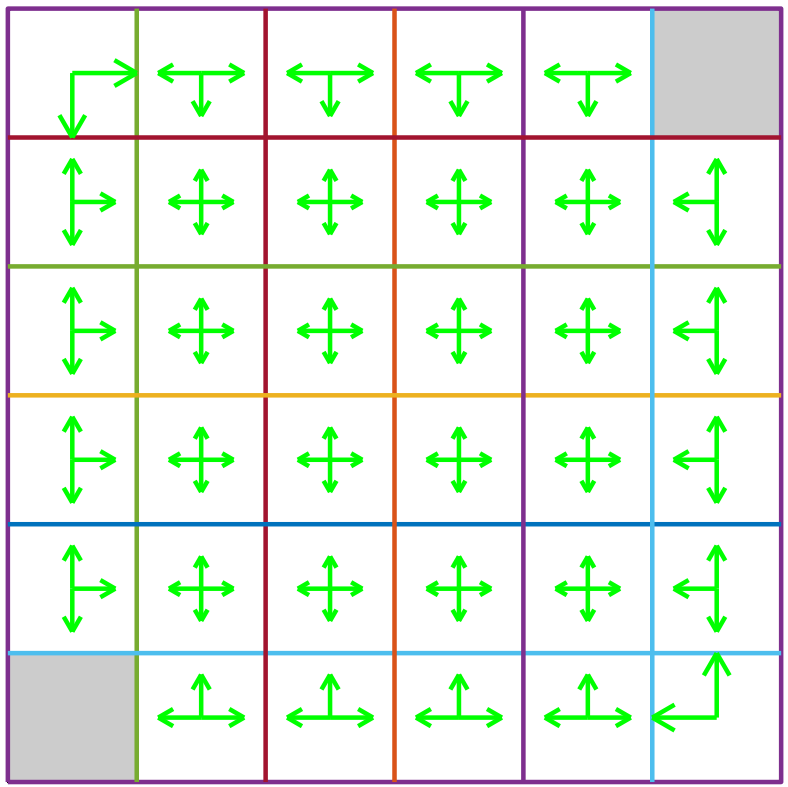


Policy

A policy π determines the probability of an action a, given a state s. Our agent is dumb and follows a random policy:

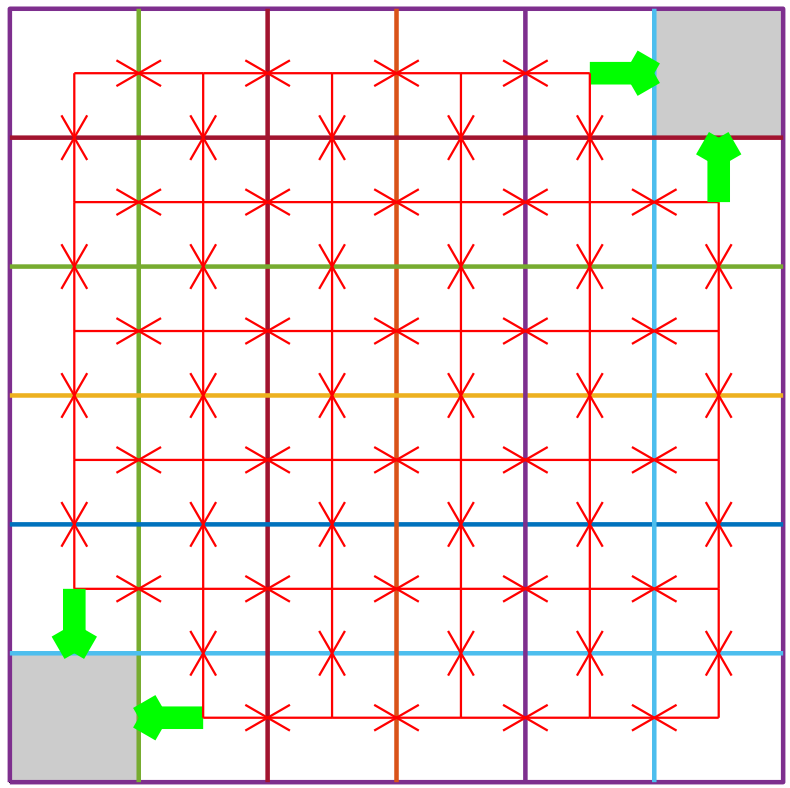
$$\pi(a|s) = {}^{1}/_{N_a(s)}$$

where $N_a(s)$ is the number of available actions from state s.



Reinforcement

As it moves, our agent receives 'punishments' and 'rewards'. All steps to *non-exit* states are punished with -1. All steps to *exit* states are rewarded with $+10$.



Value function

The value function V can be obtained only from experience.
Intrinsically, all states have zero value.

0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

Value function improves with experience

The value function is defined as the total reward expected after a state s, discounted over *all* future steps:

$$V_\pi(s) = \sum_a \pi(a,s) \left[r(a,s) + \gamma V_\pi(s') \right], \qquad s \xrightarrow{a} s'$$

Once the entire reinforcement structure (all states and actions) has been experienced, we can compute a first approximation over *one* future step:

–1	–1	–1	–1	2.7	0
–1	–1	–1	–1	–1	2.7
–1	–1	–1	–1	–1	–1
–1	–1	–1	–1	–1	–1
2.7	–1	–1	–1	–1	–1
0	2.7	–1	–1	–1	–1

ctd

Recursive application yields the value function for multiple future steps (n=5, n=10, n=20):

–3.4	–3.3	–3.1	–1.6	1.9	0
–3.3	–3.4	–3.2	–2.8	–0.88	1.9
–3.1	–3.2	–3.4	–3.2	–2.8	–1.6
–1.6	–2.8	–3.2	–3.4	–3.2	–3.1
1.9	–0.88	–2.8	–3.2	–3.4	–3.3
0	1.9	–1.6	–3.1	–3.3	–3.4

–5.7	–5.7	–5	–3.3	1.2	0
–5.7	–5.6	–5.4	–4.3	–2.3	1.2
–5	–5.4	–5.4	–5.2	–4.3	–3.3
–3.3	–4.3	–5.2	–5.4	–5.4	–5
1.2	–2.3	–4.3	–5.4	–5.6	–5.7
0	1.2	–3.3	–5	–5.7	–5.7

–7.5	–7.4	–6.6	–4.6	0.32	0
–7.4	–7.4	–7	–5.8	–3.4	0.32
–6.6	–7	–7.1	–6.7	–5.8	–4.6
–4.6	–5.8	–6.7	–7.1	–7	–6.6
0.32	–3.4	–5.8	–7	–7.4	–7.4
0	0.32	–4.6	–6.6	–7.4	–7.5

Points to note

- We consider an agent moving with a random policy π in a 'gridworld'.
- Initially, the agent does not know how different actions a from each state s are rewarded or punished.
- By trial and error, the reinforcement structure $r(a, s)$ is experienced and learned.
- The value function V_π summarizes this experience more succinctly:

$$V_\pi(s) = \sum_a \pi(a, s) \left[r(a, s) + \gamma V_\pi(s') \right], \qquad s \xrightarrow{a} s'$$

- To compute $V_\pi(s)$, we need to know the entire reinforcement structure $r(s, a)$ and perform a recursive computation! Might there not be a better way?

7 TD learning for continuous tasks

Temporal-difference learning is characterized by:

- Updating values at every time-step (instead of waiting until the entire reinforcement structure is known).
- Basing estimates on other estimates (bootstrapping).
- Deterministic convergence (difficult to prove).
- Accommodating separate value and policy functions ("actor-critic models").

We exemplify this for gridworld, but postpone "policy learning" to next lecture.

TD(0)

The simplest TD method is based on the *TD prediction error*

$$\delta(a) = r(s') + \gamma V_\pi(s') - V_\pi(s), \qquad s \xrightarrow{a} s'$$

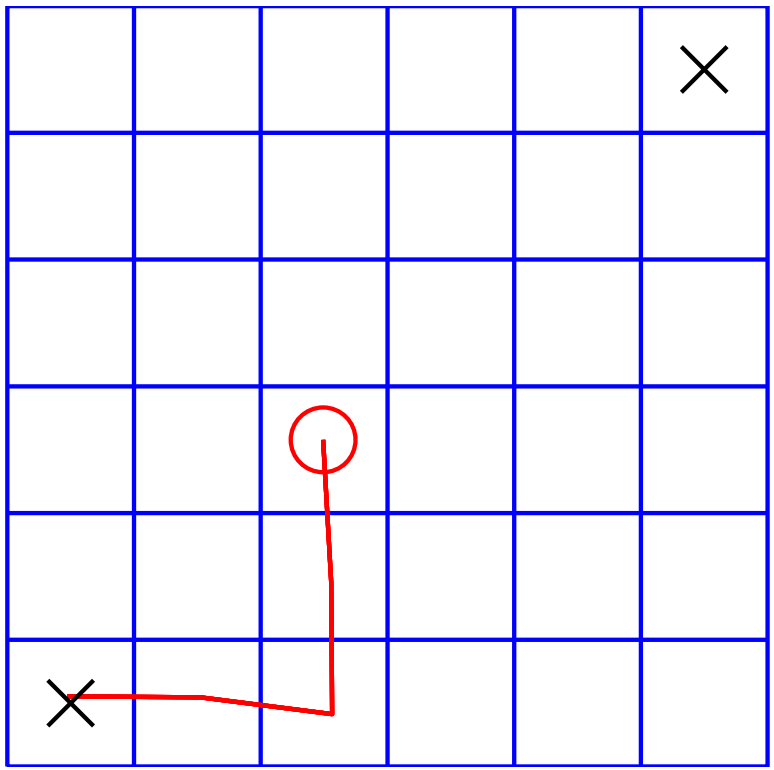
where $r(s')$ is the reward received after action a and $\gamma < 1$ is a discounting parameter.

The update rule is retrospective (applies to the previous state):

$$V_\pi(s) \rightarrow V_\pi(s) + \alpha\, \delta(a)$$

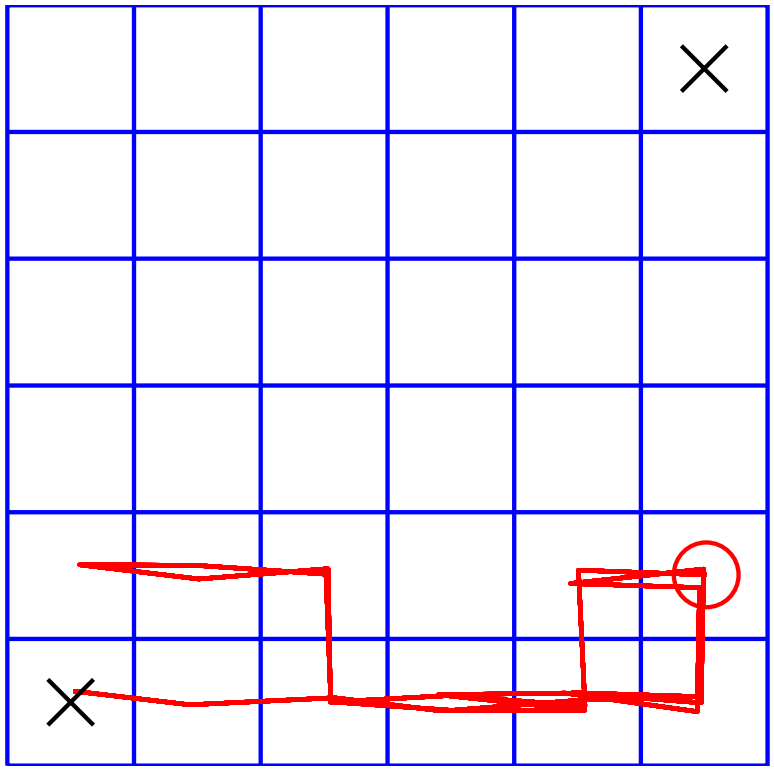
where $\alpha << 1$ is a learning rate. The method converges if α is sufficiently small. (If it is not, values change erratically with accidental actions).

Moving randomly, our actor updates its value function upon each time-step (run 1):



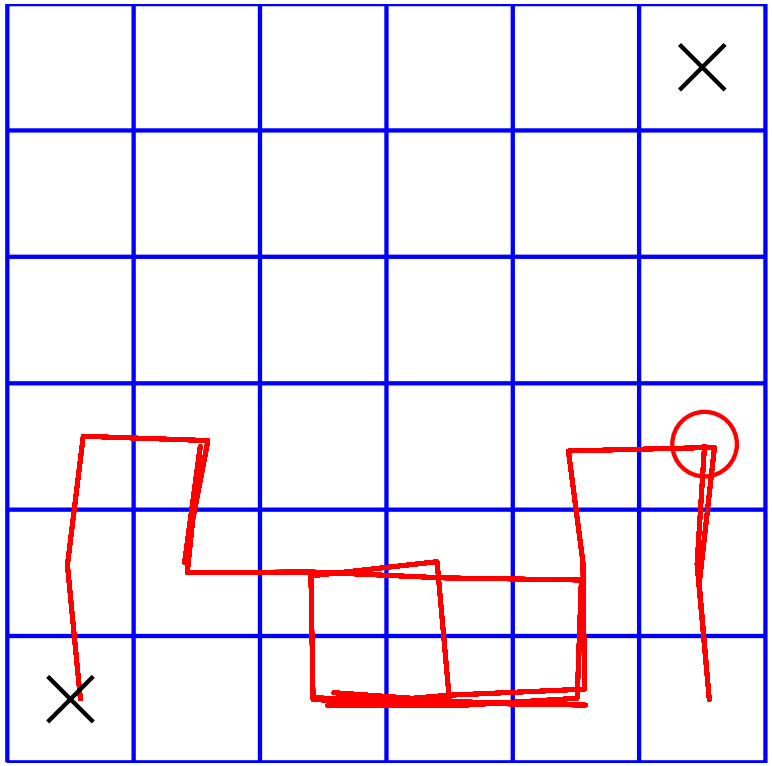
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	–0.05	0	0	0
0	0	–0.05	0	0	0
0	0.5	–0.05	0	0	0

Run 5:



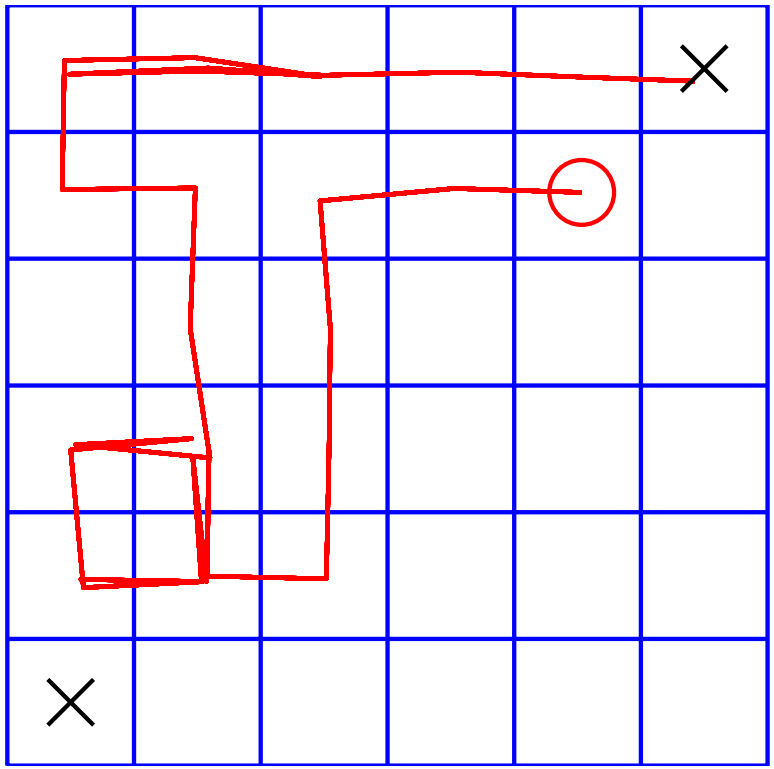
–0.05	–0.05	–0.05	–0.05	0.93	0
–0.098	–0.1	–0.05	–0.054	–0.12	0.5
–0.15	–0.1	–0.052	–0.098	–0.14	–0.2
–0.15	–0.15	–0.1	–0.1	–0.15	–0.16
–0.052	–0.1	–0.1	0	–0.1	–0.15
0	0.5	–0.14	–0.2	–0.24	–0.16

Run 10:



–0.84	–0.76	–0.53	–0.35	0.63	0
–0.87	–0.87	–0.5	–0.28	–0.2	0.5
–0.37	–0.54	–0.45	–0.5	–0.38	–0.2
–0.17	–0.33	–0.34	–0.26	–0.29	–0.3
1.4	–0.25	–0.4	–0.18	–0.31	–0.3
0	1.4	–0.58	–0.64	–0.52	–0.26

Run 20:



–1.4	–1.8	–1.5	–0.76	1.3	0
–1.4	–1.8	–1.6	–1.4	–0.67	0.55
–0.95	–1.4	–1.4	–1.5	–1.2	–0.36
–0.58	–1.1	–1.1	–1.1	–1.1	–0.45
2.4	–0.48	–1.2	–1.5	–1.3	–0.63
0	1.4	–0.77	–1.2	–1.2	–0.51

TD ctd

After multiple runs, the value function $V(s)$ begins to describe the entire reinforcement structure $r(s, a)$:

–7.5	–7.4	–6.6	–4.6	0.32	0
–7.4	–7.4	–7	–5.8	–3.4	0.32
–6.6	–7	–7.1	–6.7	–5.8	–4.6
–4.6	–5.8	–6.7	–7.1	–7	–6.6
0.32	–3.4	–5.8	–7	–7.4	–7.4
0	0.32	–4.6	–6.6	–7.4	–7.5

Correct values

–1.4	–1.8	–1.5	–0.76	1.3	0
–1.4	–1.8	–1.6	–1.4	–0.67	0.55
–0.95	–1.4	–1.4	–1.5	–1.2	–0.36
–0.58	–1.1	–1.1	–1.1	–1.1	–0.45
2.4	–0.48	–1.2	–1.5	–1.3	–0.63
0	1.4	–0.77	–1.2	–1.2	–0.51

TD approximation

Points to note

- In TD learning, each action outcome immediately contributes to the value function $V(s)$.
- Its basis is the temporal prediction error

$$\delta(a) = r(s') + \gamma V_\pi(s') - V_\pi(s), \qquad s \xrightarrow{a} s'$$

where $r(s')$ is the reward received after action a and $\gamma < 1$ is a discounting parameter.

- The update rule is retrospective (applies to previous state) after each action a

$$V_\pi(s) \rightarrow V_\pi(s) + \alpha\, \delta(a)$$

where $\alpha << 1$ is a learning rate.

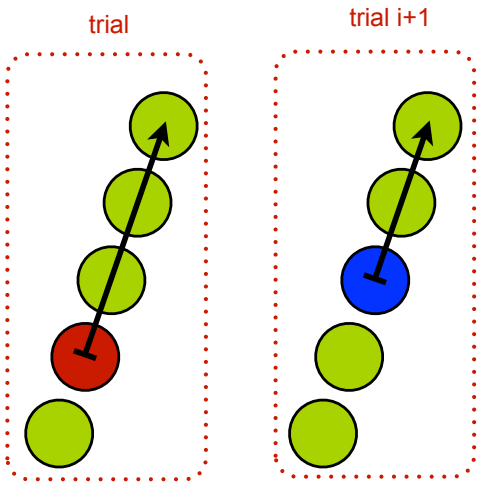
- The value function $V_\pi(s)$ provides a succinct summary of the reinforcement structure $r(a, s)$.

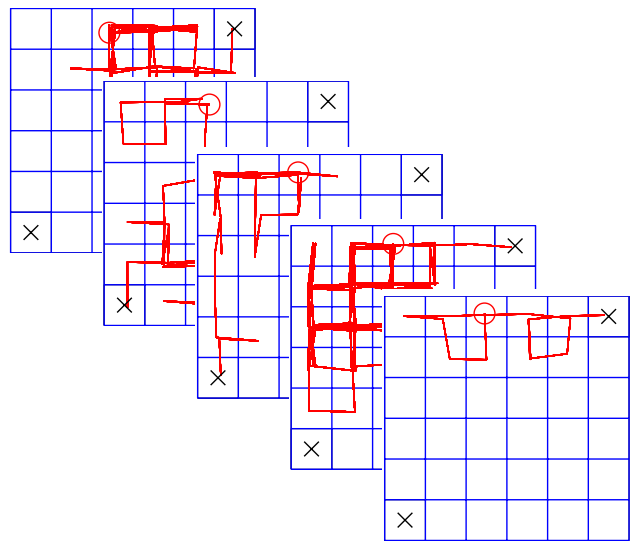
TD learning summary

- Agent estimates *total expected reward*, from stimulus to end of trial (episodic), or from current position to exit (continuous).
- Prediction error sums *rewards* and changes in *expectations*.

$$\delta(\tau) = r(\tau, s) + \Delta v(\tau, s)$$

- Bases estimates on earlier estimates (bootstrapping).





Next: Policy learning