Lecture 8. Reinforcement learning

Introduction to reinforcement learning

Related to animal learning literature.

Thorndike's law, according to Barto (1992):

> *If an action taken by a learning system is followed by a satisfactory state of affairs, then the tendency of the system to produce that particular action is strengthened or reinforced. Otherwise, the tendency of the system to produce that action is weakened.*

i.e. measure how well each decision correlates with global reinforcement signal.

RL in a nutshell (Harmon and Harmon, 1986)

Steps to riding a bike:

- Bike angled at 45 deg to right; turn left $\rightarrow$ OUCH
- Bike angled at 45 deg to right; turn right $\rightarrow$ OUCH
- (Okay, being at 45 deg a bad idea)
- Bike angled at 40 deg to right; turn left. $\rightarrow$ 45 deg to right $\rightarrow$ OUCH
- Bike angled at 40 deg to right; turn right $\rightarrow$ 35 deg to right $\rightarrow \ldots$

Concerns about reinforcement learning

- Concerns in a nutshell: credit assignment problem.

Two problems with reinforcement learning:

1. No specific signal on what the desired outputs should be.
   "It is as if each person in the United States tried to decide whether he or she had done a useful day's work by observing the gross national product [GNP] on a day by day basis" (Hinton, p220).
2. Reinforcement signal can occur long time after action. (e.g. one bad move in a game of chess, causing eventual loss of game.)
   "If ... a person wants to how their behavior affects the GNP, they need to know whether to correlate today's GNP with what they did yesterday or with what they did five years ago." (Hinton, p220).

Need for intermediate reinforcement states.

Hinton (1989) Connectionist learning systems, *Artificial Intelligence*, 40:185-234.

Conditioning paradigms

1. Classical conditioning, e.g. Pavlov's dogs.

term	example
unconditioned stimulus	food
unconditioned response	salivate
conditioned stimulus	bell
conditioned response	salivate
extinction	no reward

2. Instrumental conditioning. Here, the agent is active and is rewarded/punished on the basis of actions:
   - immediate reward (bee foraging)
   - delayed reward (backgammon, chess)

Rescorla-Wagner rule

Paradigms to explain:

paradigm	pre-train	train	result
Pavlovian		$s \to r$	$s \to' r'$
extinction	$s \to r$	$s \to \cdot$	$s \to' \cdot'$
partial		$s \to r \quad s \to \cdot$	$s \to \alpha' r'$
blocking	$s_1 \to r$	$s_1 + s_2 \to r$	$s_1 \to' r' \quad s_2 \to' \cdot'$
overshadow		$s_1 + s_2 \to r$	$s_1 \to \alpha' r' \quad s_2 \to (1-\alpha)' r'$
secondary	$s_1 \to r$	$s_2 \to s_1$	$s_2 \to' r'$

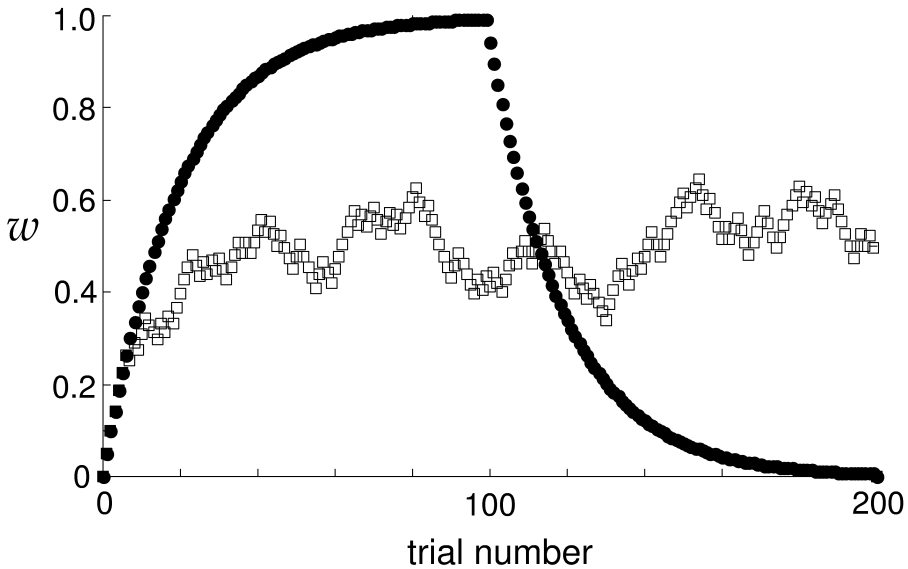
u denotes presence/absence [1/0] of stimulus; expected reward v:

$$v = wu$$

Set w to minimise the error $\langle (v - r)^2 \rangle$ between actual reward r and expected reward v. Gradient descent gets us *delta rule*:

$$w \to w + \epsilon \delta u \qquad \text{where} \quad \delta = r - v$$

Explaining Pavlovian/partial reinforcement/extinction



Solid circle: trials 1-100: $r = 1$ (Pavlovian); trials 101-200: $r = 0$ (extinction).
Open squares: $r = 1$ randomly on 50% of trials (partial reinforcement).

Blocking/inhibitory conditioning/overshadowing

paradigm	pre-train	train	result
blocking	$s_1 \to r$	$s_1 + s_2 \to r$	$s_1 \to' r' \quad s_2 \to' .'$
overshadow		$s_1 + s_2 \to r$	$s_1 \to \alpha' r' \quad s_2 \to (1-\alpha)' r'$
secondary	$s_1 \to r$	$s_2 \to s_1$	$s_2 \to' r'$

When multiple stimuli used we have $\mathbf{u}$ and $\mathbf{w}$ as vectors, $v = \mathbf{w} \cdot \mathbf{u}$ and

$$\mathbf{w} \to \mathbf{w} + \epsilon\delta\mathbf{u} \quad \delta = r - v$$

Blocking: In pre-training $w_1 \to 1$; during training $\delta = 0$ and hence w_2 remains at zero.

Overshadowing: simultaneous presentation of two stimuli means that reward is shared between them; once learnt $w_1 + w_2 = 1$.

Secondary conditioning: during pre-training $w_1 \to 1$; during training $r = 0$ and so $w_2 < 0$ (and w_1 can decrease). **Problem**?

Temporal difference (TD) learning

Consider time within a trial more carefully; t is an integer (0,T) within a trial; stimulus $u(t)$, prediction $v(t)$ and reward $r(t)$ vary during the trial. $v(t)$ predicts total future reward expected, i.e. from time t to end of trial:

$$\text{Value:} \qquad v(t) = \left\langle \sum_{\tau=0}^{T-t} r(t+\tau) \right\rangle$$

Linear-filter: estimate $v(t)$ based on stimuli just seen:

$$v(t) = \sum_{\tau=0}^{t} w(\tau) u(t-\tau)$$

Modify delta rule to update weights at any time t during trial:

$$w(\tau) \rightarrow w(\tau) + \epsilon \delta(t) u(t-\tau) \quad \delta(t) = \left(\sum_{\tau=0}^{T-t} r(t+\tau) \right) - v(t)$$

Problem: how to compute $\delta(t)$?

TD learning

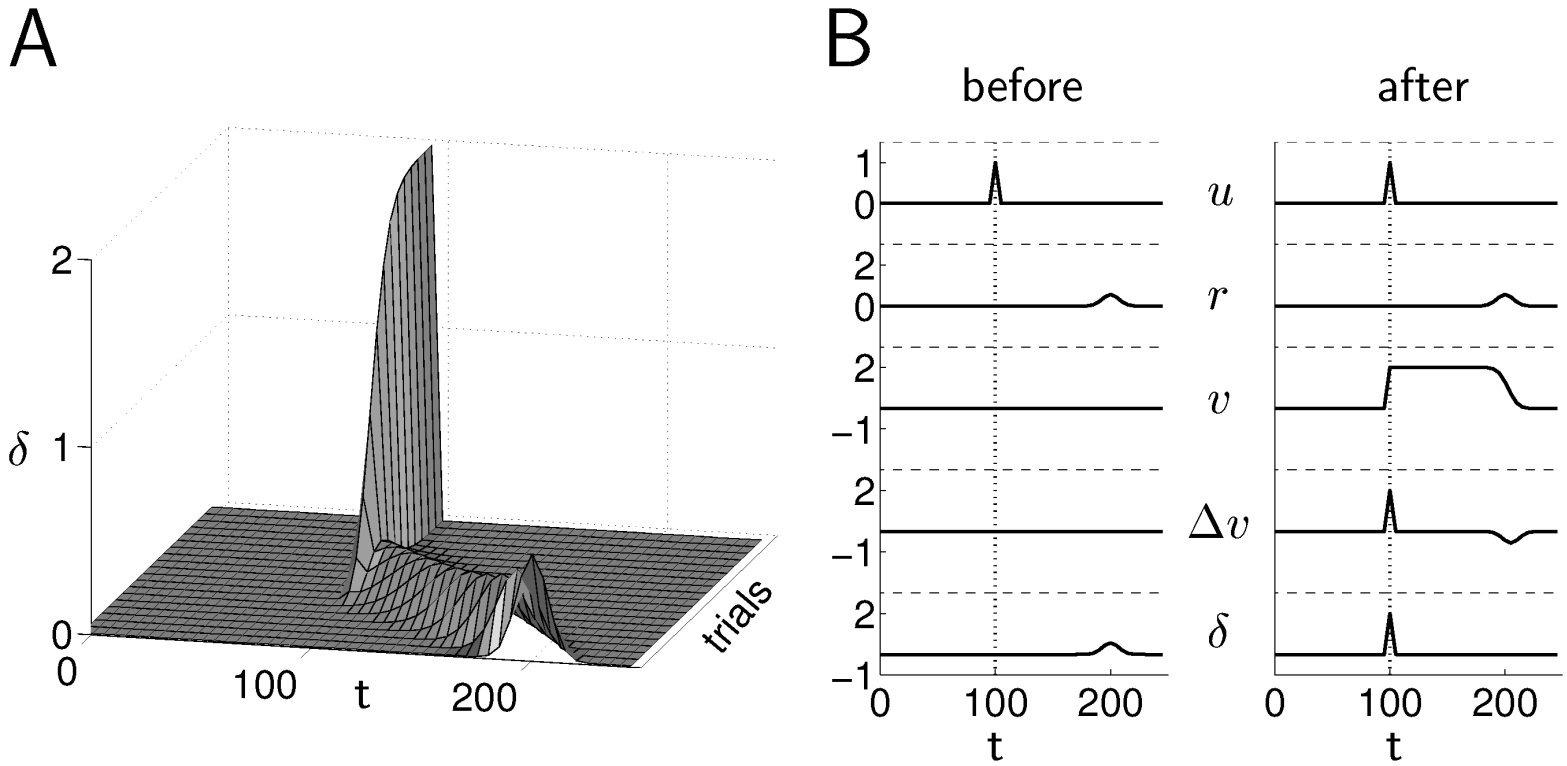
$$\delta(t) = \sum_{\tau=0}^{T-t} r(t+\tau) - v(t)$$
$$= r(t) + \sum_{\tau=0}^{T-t-1} r(t+1+\tau) - v(t)$$

The summation here can be approximated by $v(t+1)$ in a working model, so:

$$\delta(t) \approx r(t) + v(t+1) - v(t)$$
$$w(\tau) \to w(\tau) + \epsilon\delta(t)u(t-\tau)$$

This rule is named temporal difference rule because of the term $v(t+1) - v(t)$ which measures the difference in predictions (rather than difference in output and target).

Learning to predict a reward



Part II: Instrumental conditioning

Classical conditioning is artificial in that agent is passive receiving stimuli and rewards.
Natural situation: rewards delivered (after some delay) as a result of an agent's actions.

We will consider two cases:

1. immediate reward (static action choice).
2. delayed reward (delayed action choice).

Static action choice

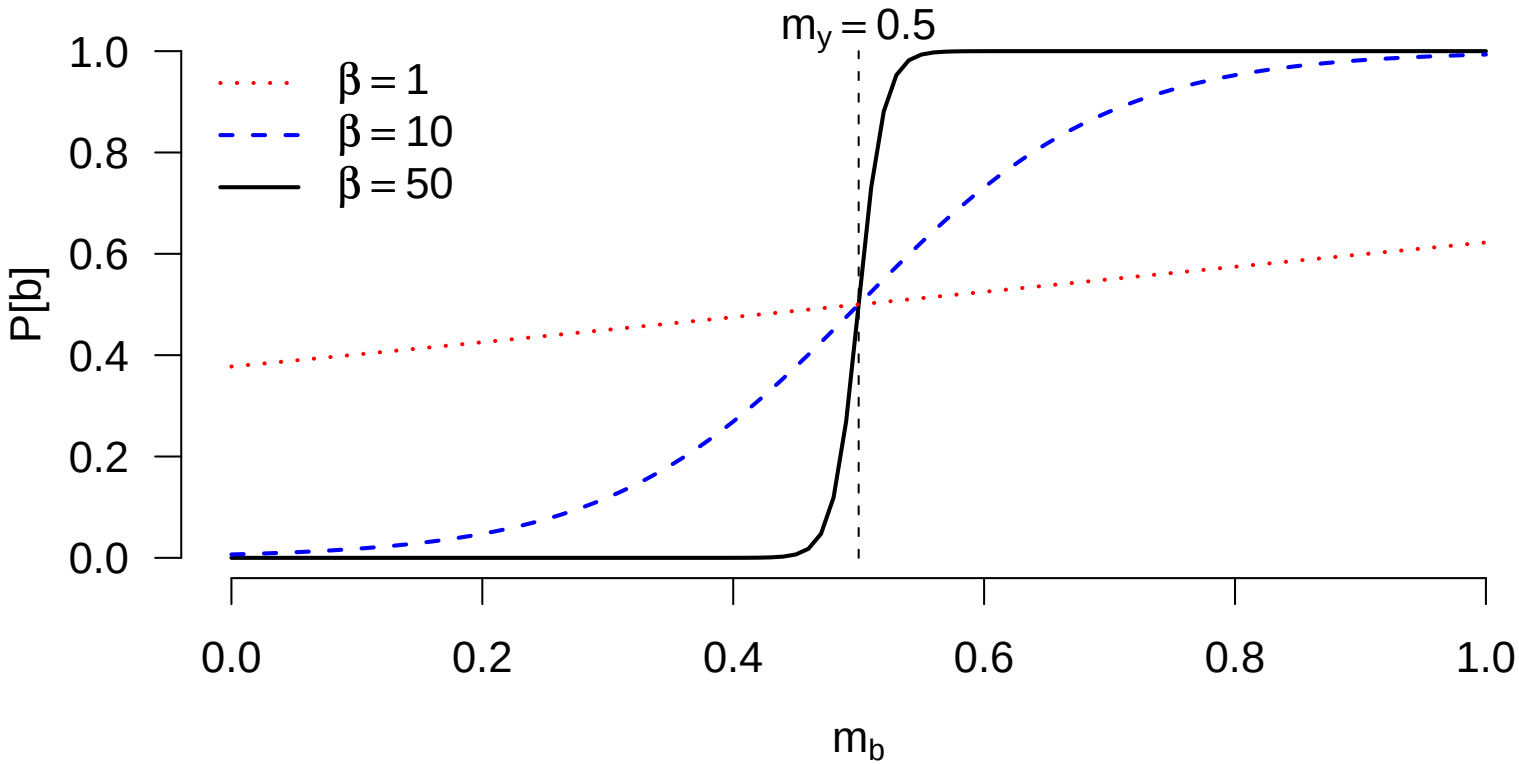
Consider the case of the model bee, deciding whether to visit a blue (b) flower or yellow (y) flower. The volume of nectar delivered at each trial varies so several samples are required.
Here we ignore any spatial structure to the problem of finding and sampling flowers.
Policy: scheme for choosing an action at every state.
Stochastic policy: probability of selecting blue flower (P[b]) or yellow flower (P[y] $\equiv$ 1-P[b]) given by softmax activation:

$$P[b] = \frac{\exp(\beta m_b)}{\exp(\beta m_b) + \exp(\beta m_y)} \quad P[y] = \frac{\exp(\beta m_y)}{\exp(\beta m_b) + \exp(\beta m_y)}$$

$m_b, m_y \in [-\infty, \infty]$ are the action values, which adapt during learning . β (fixed) controls exploration vs exploitation.
Two methods to solve task: indirect actor and direct actor.

Exploitation vs Exploration trade off

Assume here that $m_y = 0.5$ is fixed; how does that probability of choosing a blue flower, $P[b]$, vary with as m_b varies?



Small β allows for exploration of the sample space.

Large β allows for exploitation of the (current) higher action.
This task also known as the n-armed bandit (slot machine) problem.

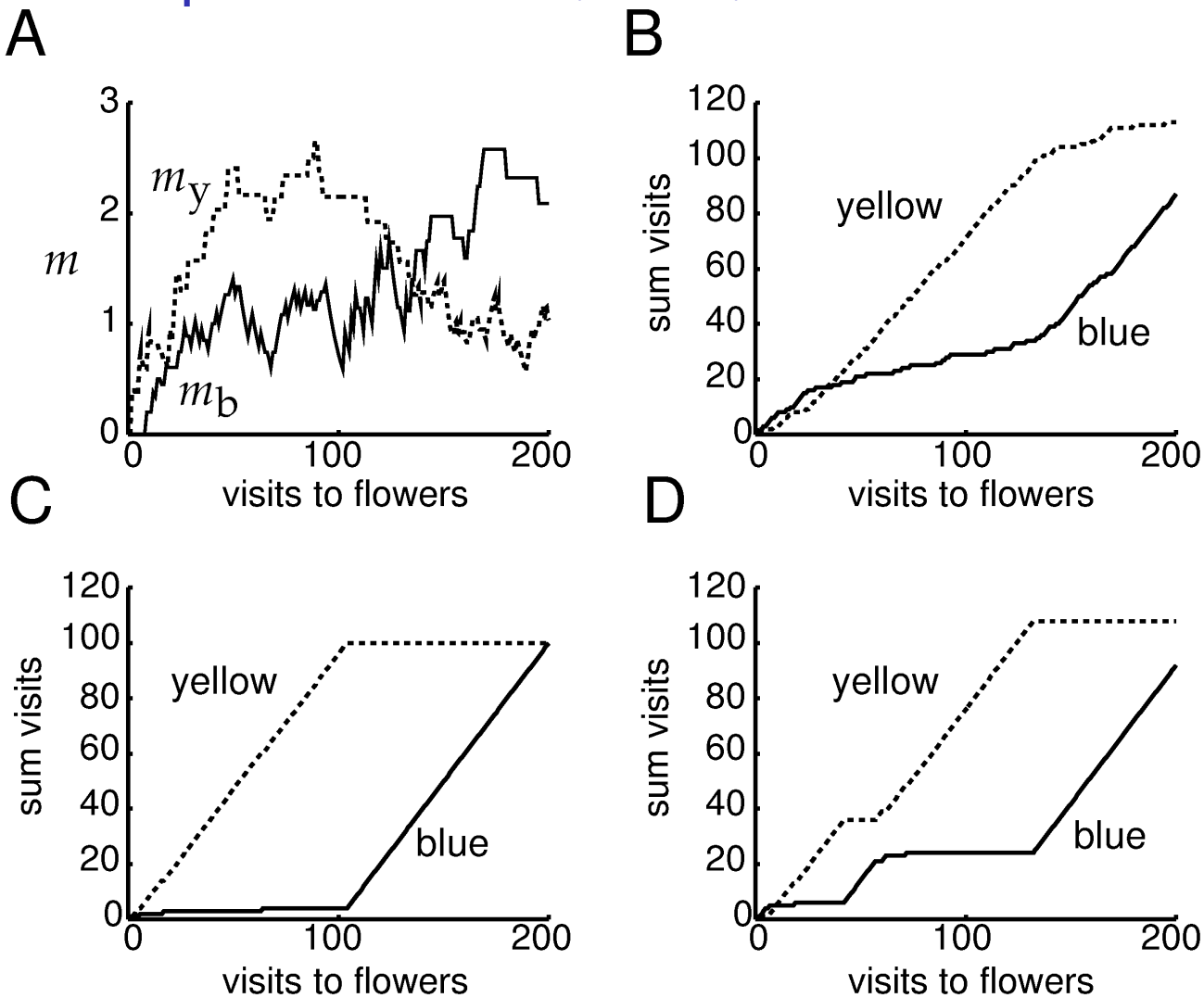
Indirect actor: approach

The indirect approach is for the bee to repeatedly visit the two flowers and determine the mean nectar volumes from each flower. This can be done using the delta rule, e.g. after visiting blue flower and receiving reward r_b:

$$m_b \to m_b + \epsilon\delta \qquad \delta = r_b - m_b$$

and leaves m_y unchanged. This leads to $m_b = \langle r_b \rangle$, with a similar rule for yellow flower.

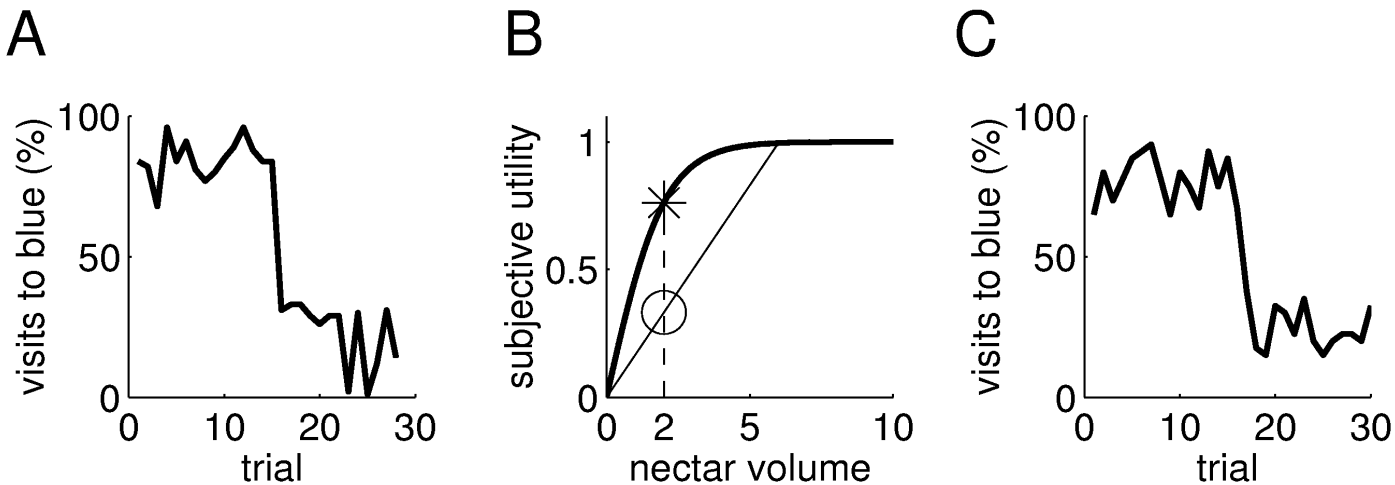
Indirect actor: performance (TN Fig 9.4)



Rewards: $\langle r_b \rangle = 1, \langle r_y \rangle = 2$ with switch after 100 trials. A,B: $\beta = 1$, slow to adapt, but ultimately good. C,D: $\beta = 50$: depending on initial conditions, exploitation strategy is successful.

Indirect actor: bumble bee risk aversion (TN Fig 9.5)

Blue flowers (reliable): always provide 2 μl nectar.
Yellow flowers (unreliable): 1/3 visits provide 6 μl, and 2/3 visits provide no nectar (same mean).
Bees (A) prefer the reliable flower and when characteristics switch (trial 15), will quickly switch preference.



B: Subjective utility of nectar for variable flowers (circle) and constant flowers (star). (This is empirically derived.)
C: model bee performance ($\epsilon = .3, \beta = 23/8$) on single trial.
NB: bees can be persuaded to sample variable flower if the expected mean is raised to trade-off against the variability.

Direct actor: methods

Choose action values (m) directly to maximise the average expected reward. i.e. gradient ascent of:

$$\langle r \rangle = P[b]\langle r_b \rangle + P[y]\langle r_y \rangle$$
$$\frac{\partial r}{\partial m_b} = \beta P[b](1 - P[b])(\langle r_b \rangle - \bar{r}) - \beta P[y](P[b])(\langle r_y \rangle - \bar{r})$$

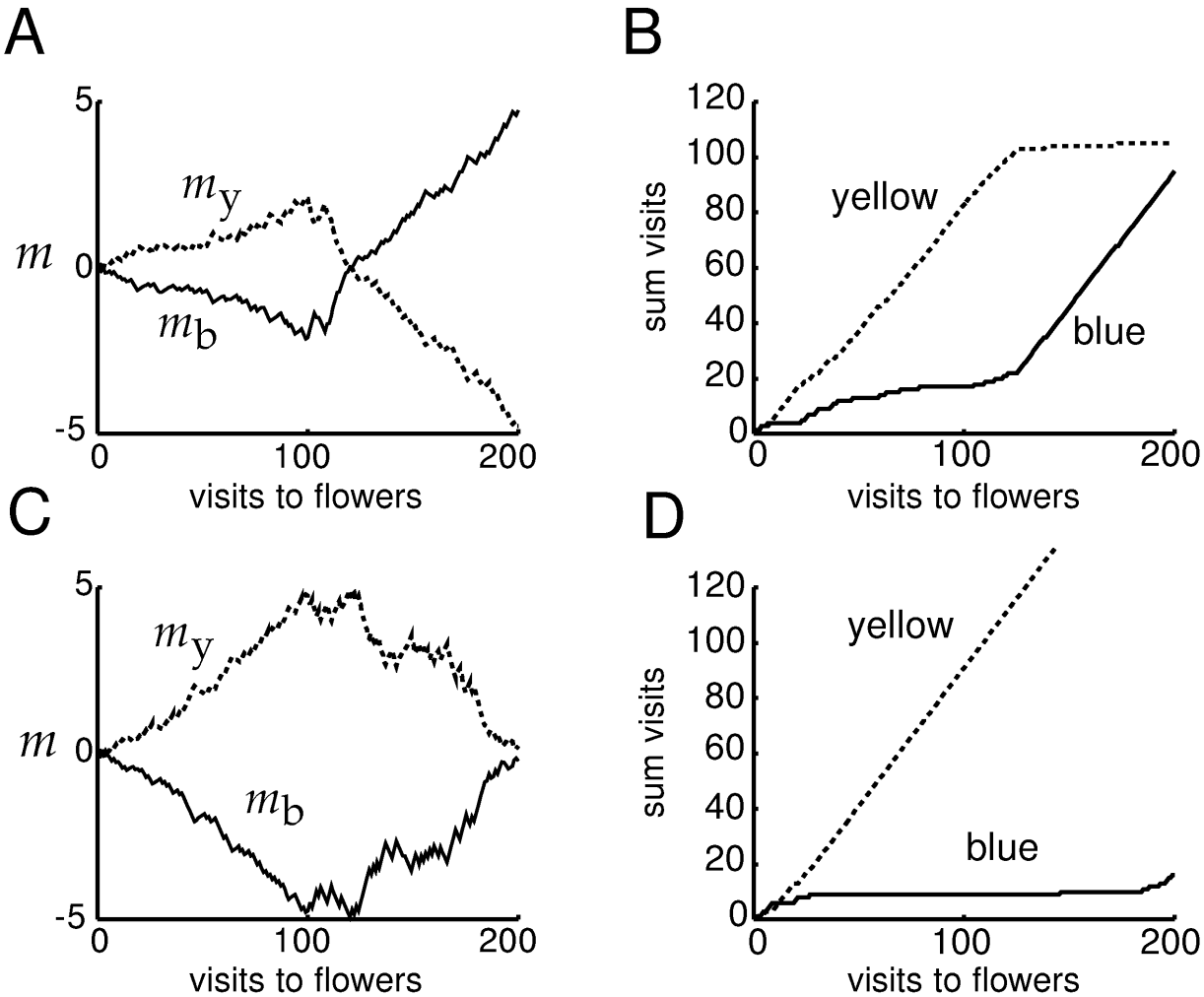
which leads to learning rule for action a [using Kronecker delta]:

$$m_b \to m_b + \epsilon(\delta_{ab} - P[b])(r_a - \bar{r})$$
$$m_y \to m_y + \epsilon(\delta_{ay} - P[y])(r_a - \bar{r})$$

$\bar{r}$ controls the speed of learning, and typically is set to mean reward under the policy.

We can interpret $(r_a - \bar{r}) \equiv \delta$ so that above-average rewards are reinforced if $\delta_{ab} = 1$.

Direct actor: performance



Rewards: $\langle r_b \rangle = 1, \langle r_y \rangle = 2$ with switch after 100 trials.

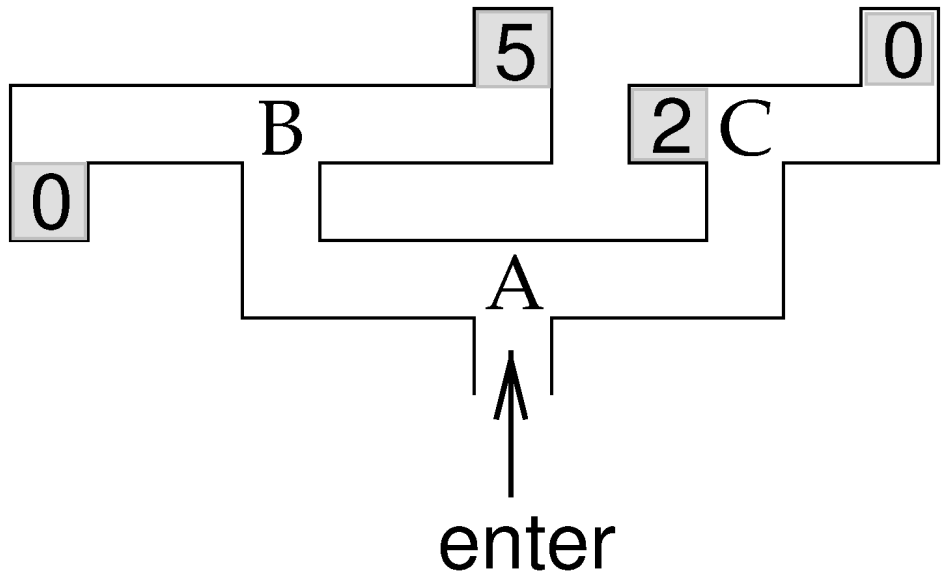
A,B: good performance, tracking switch in performance.

C,D, poor performance after switch.

Despite poor performance (compared to indirect actor), useful in actor–critic models . . .

Sequential action choice

Rewards are often delayed, so how do we know which actions to reinforce? e.g. in maze task below, decision to turn left or right at A is not immediately rewarded:



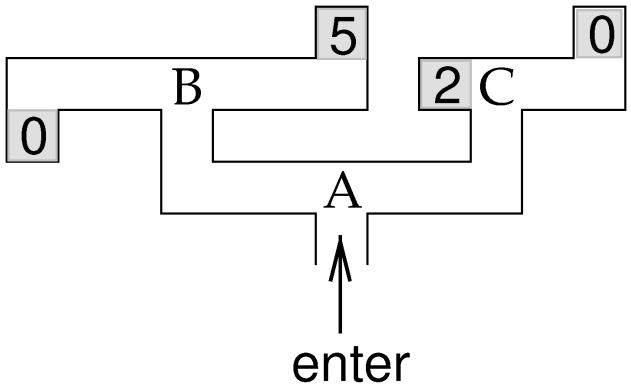
actor: at each state ($u \in A, B, C$), decide whether to go L/R, based on softmax activation. Actor learns, e.g. at loc A, based on critic providing estimates of future rewards, v(B), v(C).

critic: evaluates total future reward at any location. Critic uses TD learning to predict v(u).

We use static action choice, with critic providing intermediate rewards rather than animal receiving immediate (external) reward.

Critic learning

policy evaluation: for a fixed policy (i.e. **m** fixed), use TD learning to modify weights such that $v(u)$ are reliable.



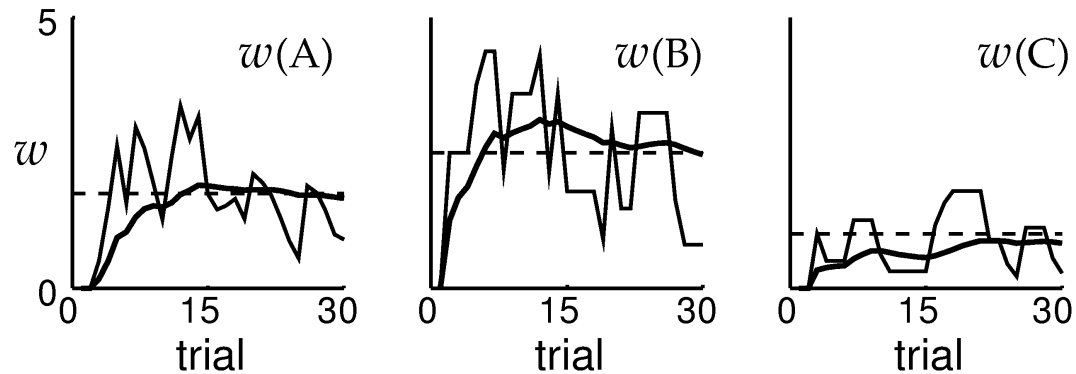
Here $v(u) = w(u)$ due to coding of location.

Consider initial unbiased policy $\mathbf{m}(u) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, u = A, B, C$. Ideal $v(u)$:

$v(B) = 0.5(0 + 5) = 2.5 \quad v(C) = 0.5(2 + 0) = 1 \quad v(A) = 0.5(2.5 + 1) = 1.75$

Learn to predict $v(u)$; assume rat takes action a at locn u to move to u':

$$w(u) \to w(u) + \epsilon\delta \qquad \text{where} \quad \delta = r_a(u) + v(u') - v(u)$$



Actor learning/1

policy improvement: critic provides $v(u)$ as rewards of intermediate states so that actor learns how to improve. (i.e. update weights **m**).
Update actor using direct-actor method:

$$m_b \rightarrow m_b + \epsilon(\delta_{ab} - P[b])(r_a - \bar{r})$$

Let $\delta = r_a - \bar{r}$

r_a is a sample of worth of current loc; $\bar{r}$ is the average value of all actions that can be taken. We can likewise define an error term for the sequential action choice, taking action a to move from u to u':

$$\delta = \text{estimate of worth of current action} -$$
$$\text{average value of all actions that can be taken at loc}$$
$$\delta = [r_a(u) + v(u')] - v(u)$$
$$m_{a'}(u) \rightarrow m_{a'}(u) + \epsilon(\delta_{aa'} - P[a'; u])\delta$$

Actor learning/2

policy improvement: critic provides $v(u)$ as rewards of intermediate states so that actor learns how to improve. (i.e. update weights **m**).

e.g. for rat in location u
about to take action a to move to location u'
(and receive potentially any external reward $r_a(u)$.
Update action values at u using direct-actor method:

$$m_{a'}(u) \rightarrow m_{a'}(u) + \epsilon(\delta_{aa'} - P[a'; u])\delta$$

where $\delta = r_a(u) + v(u') - v(u)$

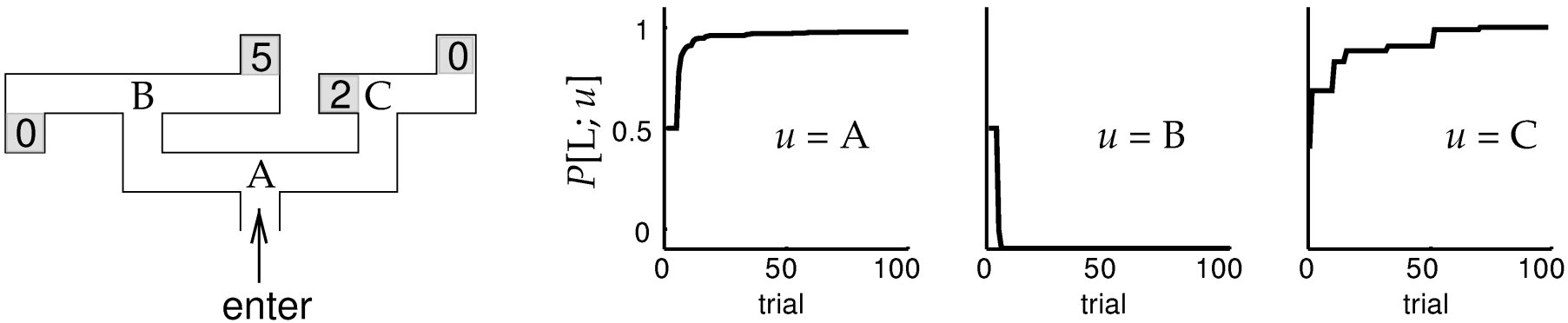
$P[a'; u] =$ prob. of taking action a' at u, determined by **m**

N.B. Delta function: $\delta_{aa'} = \begin{cases} 1 & \text{if } a = a' \\ 0 & \text{otherwise} \end{cases}$

Actor-critic learning

As policy changes, TD learning in critic changes. However, hope is that since we are improving actions at each location, critic will perform better and lead to even better actions.
Monotonic improvement can be proved formally for Markov decision problems (see Appendix to TN chapter 9).

actor-critic algorithm: interleave policy evaluation and policy improvement steps.



$P[L; u]$: probability of turning left at each location. Learning at C is relatively slow as it normally learns to turn left at A, so ignoring point C.

Generalizations of actor-critic learning

1. Use state-vector **u** at each location u; now have action matrix **M**

$$\text{critic update:} \quad \mathbf{w} \rightarrow \mathbf{w} + \epsilon\delta\mathbf{u}(u)$$

$$\text{actor:} \quad m_a = \sum_b M_{ab} u_b(u), \qquad P[a] = \text{softmax}(m_a)$$

$$M_{a'b} \rightarrow M_{a'b} + \epsilon(\delta_{aa'} - P[a'; u]) \times \delta \times u_b(u)$$

2. Rewards soon after an action should be more important than rewards received later. Use **discounting factor**, γ [0,1] so that error signal becomes:

$$\delta = r_a(u) + \gamma v(u') - v(u)$$

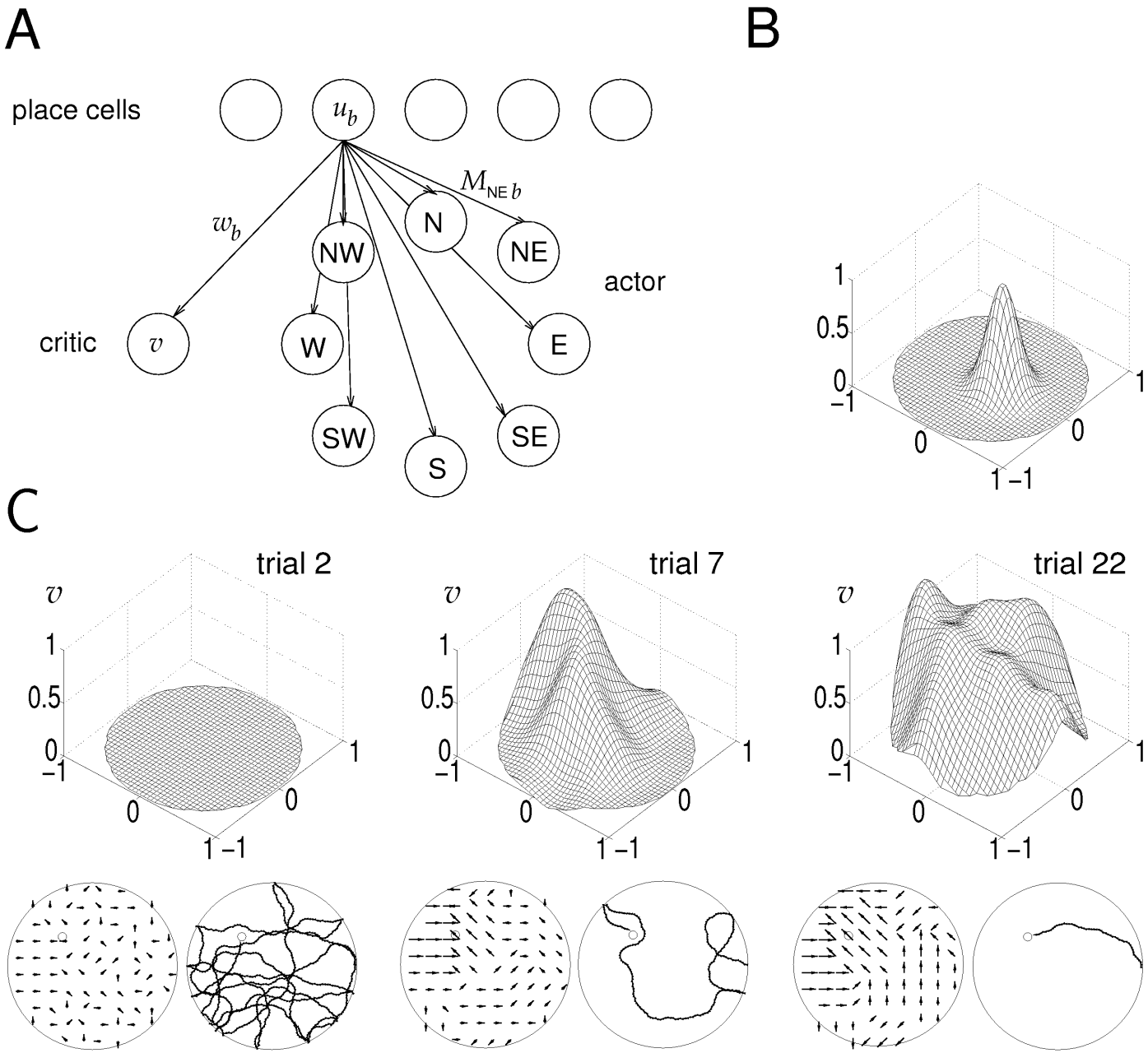
3. [TD(λ) learning] Use a factor λ to weight how strongly rewards from temporally distant points affects learning; create **stimulus traces** to speed up learning.

$$\tilde{\mathbf{u}}(t) = \tilde{\mathbf{u}}(t-1) + (1-\lambda)(\mathbf{u}(t) - \tilde{\mathbf{u}}(t-1))$$

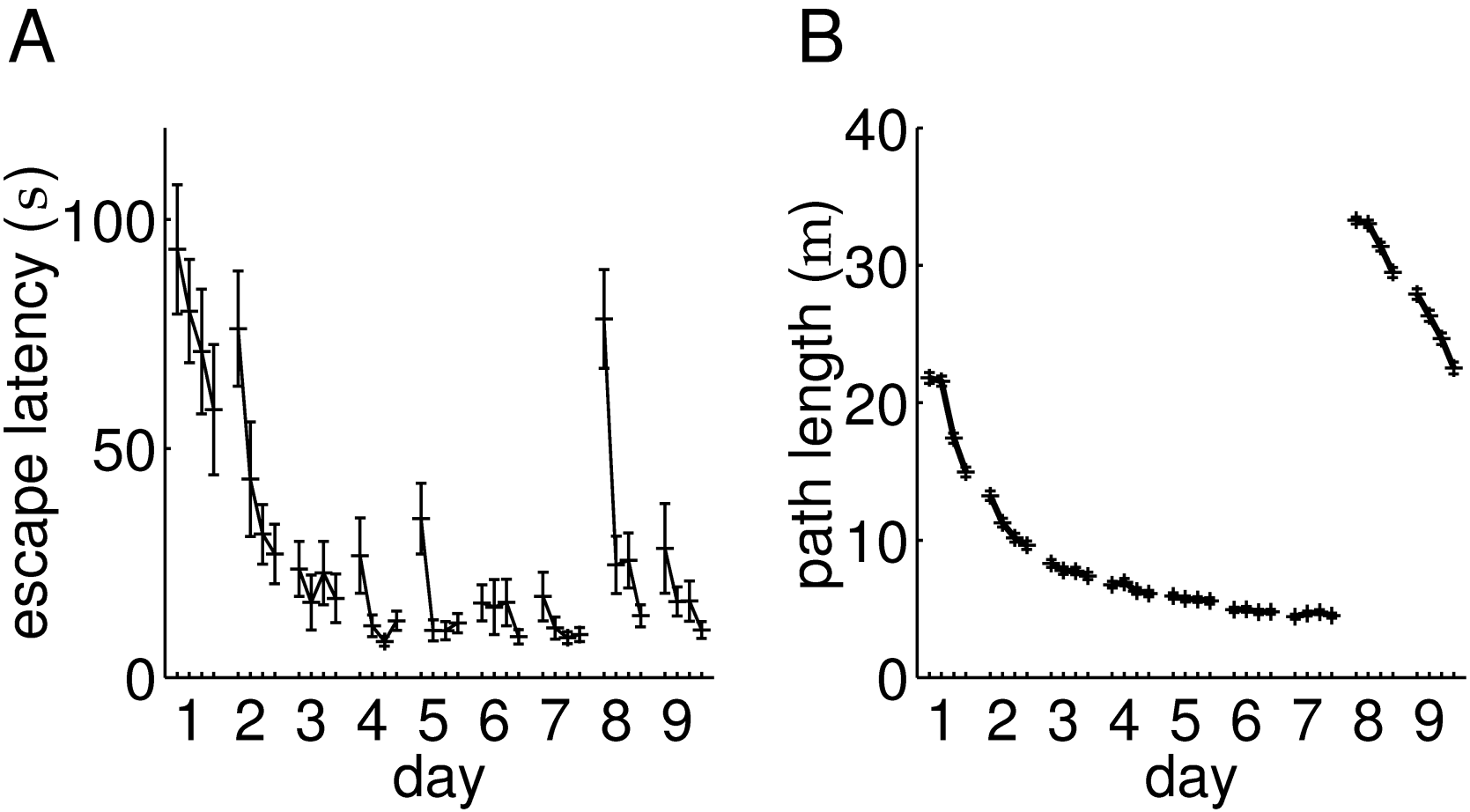
Learning the water maze (Foster et al. (2000))

- Morris water maze: 2m diameter pool with milky water and hidden platform.
- Rat is averse to water and hence wants to find the platform to stop swimming.
- Once found, navigation to platform on subsequent trials is fairly rapid.
- Lesions to hippocampus impairs navigation performance; place cells are thought to be involved.
- But place cells only say what portion of space you are in, rather than providing a navigation cue (e.g. go left).

Water maze: architecture

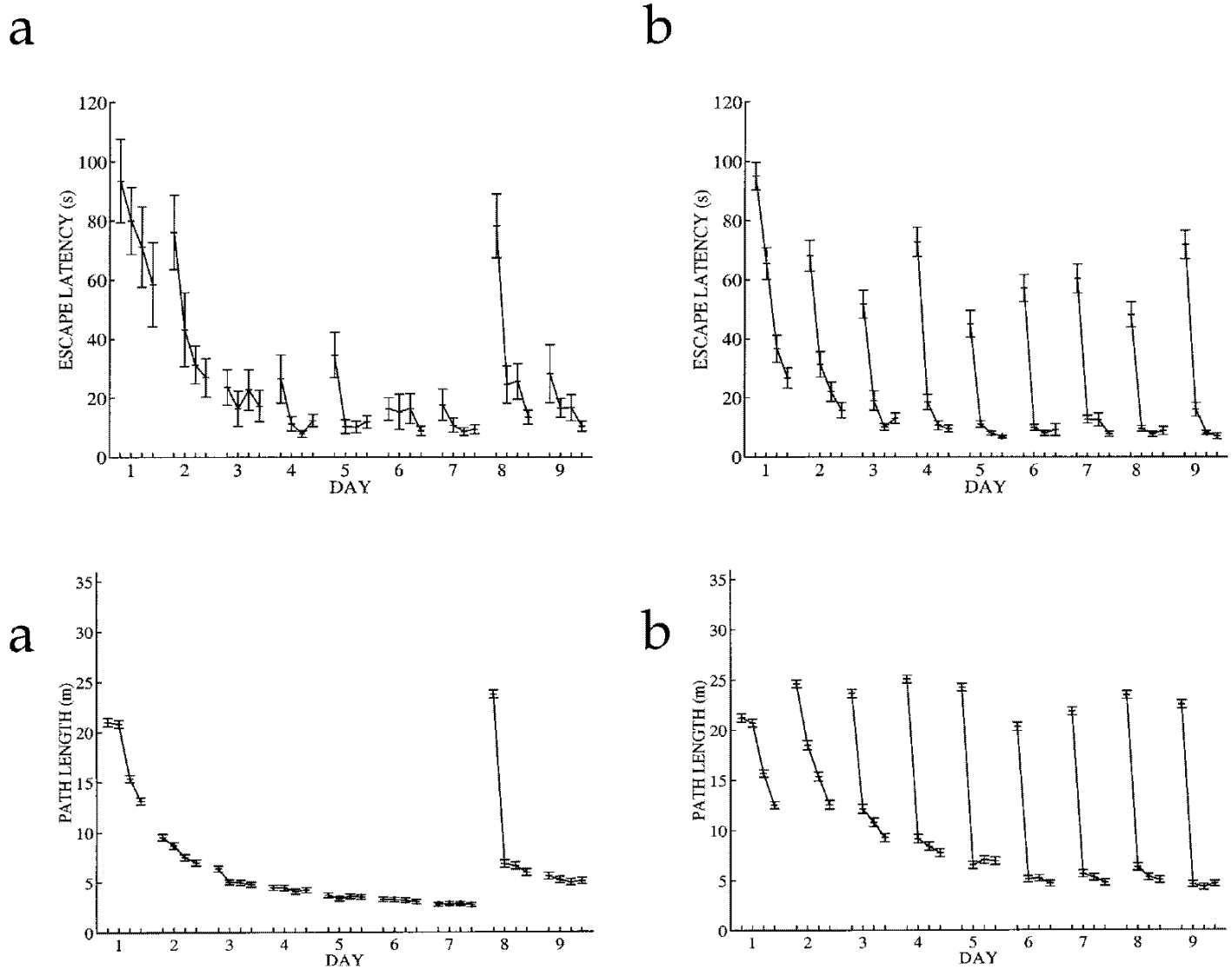


Water maze: performance



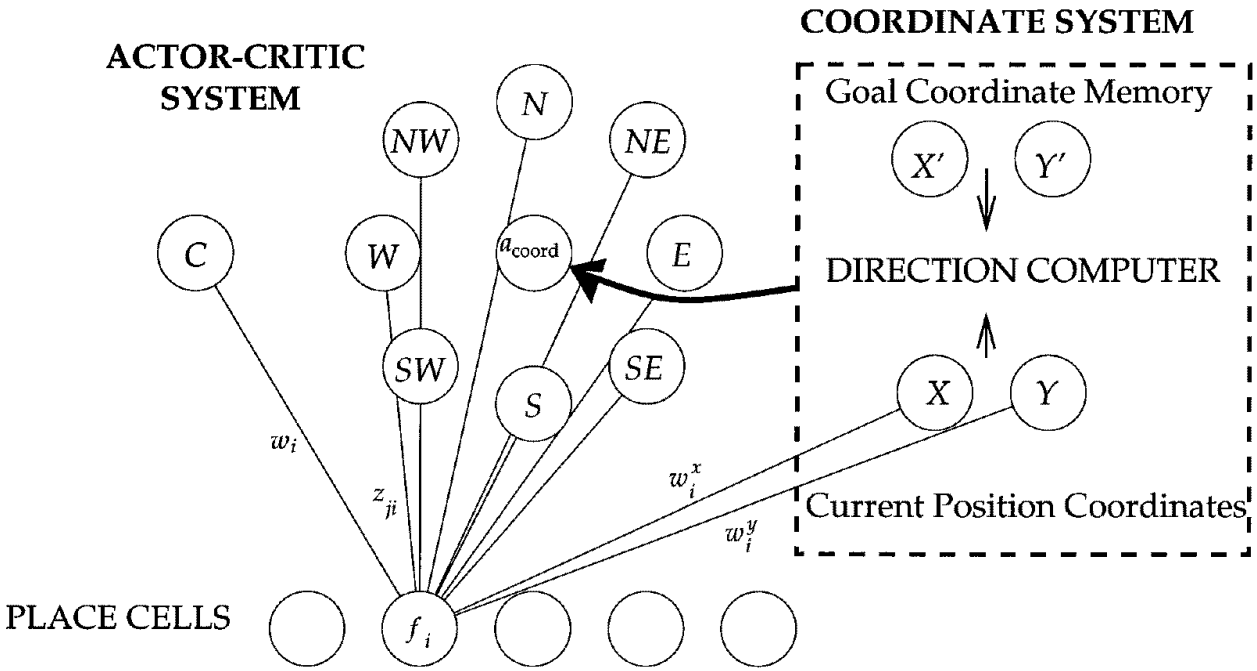
Left: experimental data — time to reach platform. Right: path length in model. Platform moved at day 8. Model broadly okay, although problem when platform switched.

Delayed matching to place (DMP) task



Top: exptl data; bottom: model with coordinate system.
Left: reference task; right: delayed matching to place.

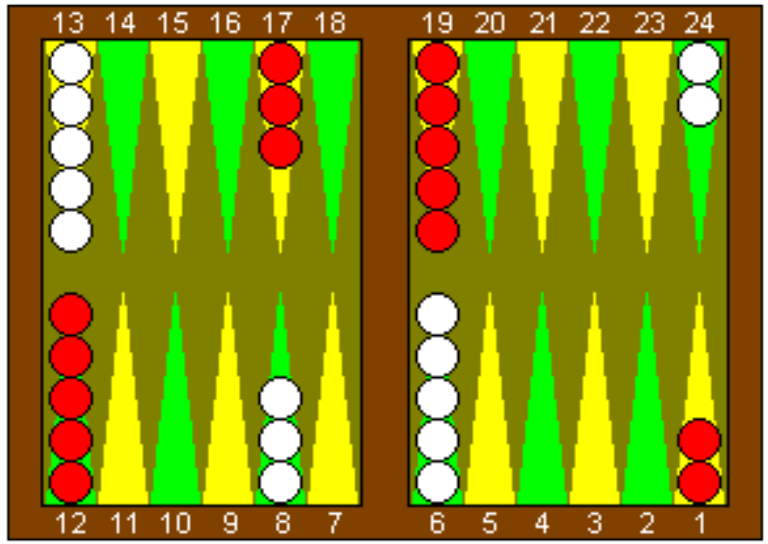
Learning the DMP task: coordinate system



Weights w^X and w^Y adjusted also using TD learning; $\mathrm{a}_{\mathrm{coord}}$ indicates the preferred direction that the coordinate system would take, which is considered in competition with the actor's preferred direction.

Real-world application of TD learning (Tesauro 1995)

`http://www.research.ibm.com/massive/tdl.html`



Backgammon is perfect since it a noisy system (through dice-rolling); neural network takes state representation (198 units) and outputs 1 value, $V(t)$, i.e. chance of winning in that state.

After dice-rolled, each of ≈ 20 next states evaluated; pick state with maximal $V(t)$. Network trains against itself, with ultimate reward of $r = 1$ if player wins.

As of 1995, extremely close to equaling the world's best human players. Has taught experts new opening moves (e.g. splitting (24-23) rather than slotting (6-5)).

Summary

- Rescorla-Wagner rule for simple tasks.
- Temporal difference learning.
- Actor-critic models.
- Application to navigation: place cells provide substrate for TD learning to provide goal-directed navigation.
- Reading: TN Chapter 9; Foster et al. (2000).