Markov Decision Processes
and
Reinforcement Learning

Readings:

- Mitchell, chapter 13
- Kaelbling, et al., *Reinforcement Learning: A Survey*, JAIR, 1996
- for much more: *Reinforcement Learning, an Introduction*, Sutton & Barto

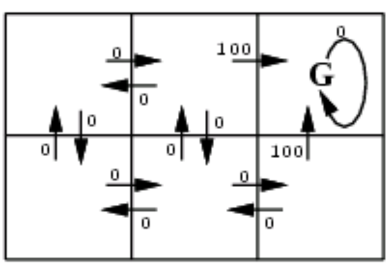
Machine Learning 10-701

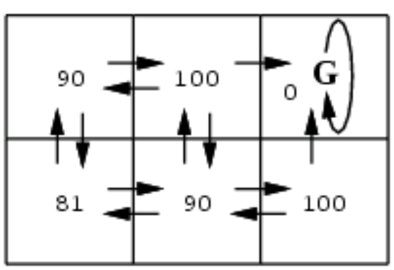
April 26, 2010

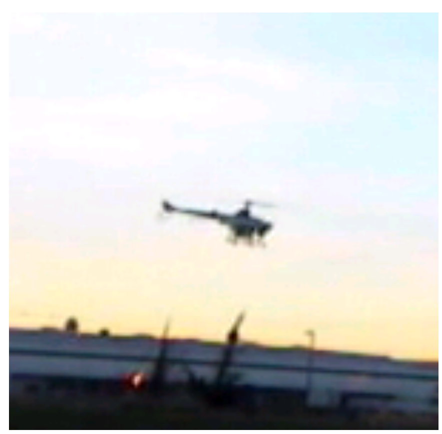
Tom M. Mitchell
Machine Learning Department
Carnegie Mellon University

Reinforcement Learning

[Sutton and Barto 1981; Samuel 1957; ...]







$$V^*(s) = E[r_t + \gamma\, r_{t+1} + \gamma^2 r_{t+2} + \ldots]$$

Reinforcement Learning: Backgammon

[Tessauro, 1995]

Learning task:

- chose move at arbitrary board states

Training signal:

- final win or loss

Training:

- played 300,000 games against itself

Algorithm:

- reinforcement learning + neural network

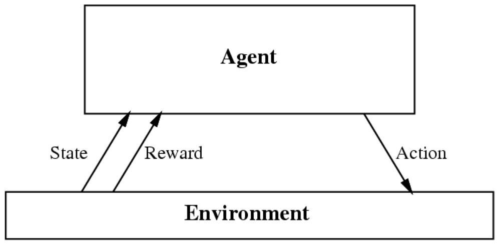
Result:

- World-class Backgammon player

Outline

- Learning control strategies
  - Credit assignment and delayed reward
  - Discounted rewards
- Markov Decision Processes
  - Solving a known MDP
- Online learning of control strategies
  - When next-state function is known: value function $V^*(s)$
  - When next-state function unknown: learning $Q^*(s,a)$
- Role in modeling reward learning in animals

Reinforcement Learning Problem



$$s_0 \xrightarrow[r_0]{a_0} s_1 \xrightarrow[r_1]{a_1} s_2 \xrightarrow[r_2]{a_2} \ldots$$

Goal: Learn to choose actions that maximize

$$r_0 + \gamma r_1 + \gamma^2 r_2 + \ldots \text{ , where } 0 \leq \gamma < 1$$

Markov Decision Process = Reinforcement Learning Setting

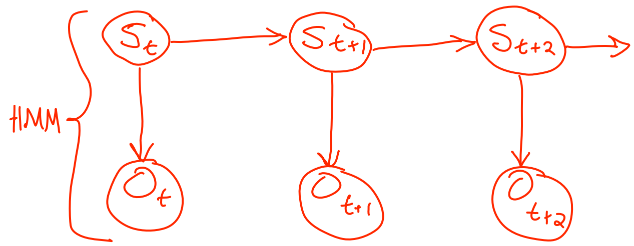
- Set of states S
- Set of actions A
- At each time, agent observes state $s_t \in S$, then chooses action $a_t \in A$
- Then receives reward r_t , and state changes to s_{t+1}
- Markov assumption: $P(s_{t+1} | s_t, a_t, s_{t-1}, a_{t-1}, ...) = P(s_{t+1} | s_t, a_t)$
- Also assume reward Markov: $P(r_t | s_t, a_t, s_{t-1}, a_{t-1},...) = P(r_t | s_t, a_t)$

- The task: learn a policy π: S → A for choosing actions that maximizes

$$E[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + ...] \quad 0 < \gamma \leq 1$$

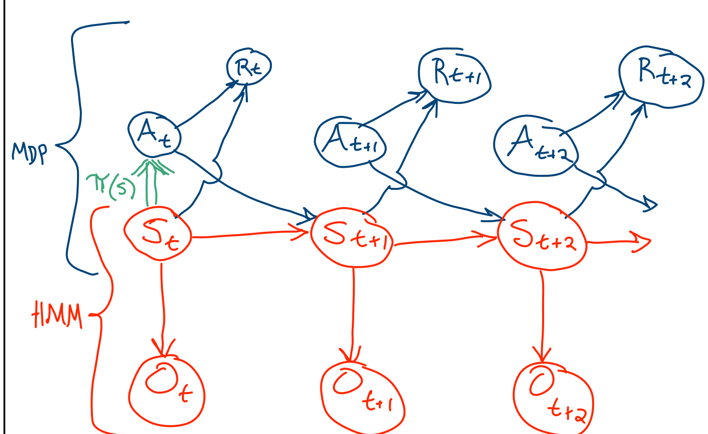
for every possible starting state s_0

HMM, Markov Process, Markov Decision Process



Tom Mitchell, April 2010

HMM, Markov Process, Markov Decision Process



Tom Mitchell, April 2010

Reinforcement Learning Task for Autonomous Agent

Execute actions in environment, observe results, and

- Learn control policy π: S→A that maximizes $\sum_{t=0}^{\infty} \gamma^t E[r_t]$ from every state s ∈ S

Note:

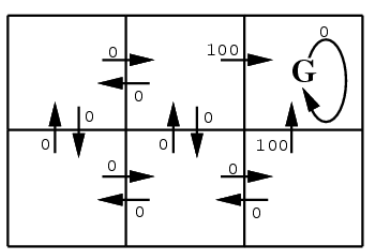
- Function to be learned is π: S→A
- But training examples are not of the form <s, a>
- They are instead of the form < <s,a>, r >

Reinforcement Learning Task for Autonomous Agent

Execute actions in environment, observe results, and

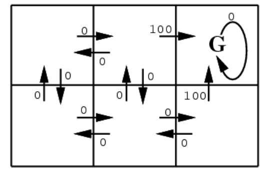
- Learn control policy π: S→A that maximizes $\sum_{t=0}^{\infty} \gamma^t E[r_t]$ from every state s ∈ S

Example: Robot grid world, deterministic reward *r(s,a)*



$r(s, a)$ (immediate reward)

Value Function for each Policy



- Given a policy π : S → A, define

$$V^{\pi}(s) = E[\sum_{t=0}^{\infty} \gamma^t r_t]$$ assuming action sequence chosen according to π, starting at state *s*

- Then we want the policy π^* where

$$\pi^* = \arg\max_{\pi} V^{\pi}(s), \quad (\forall s)$$

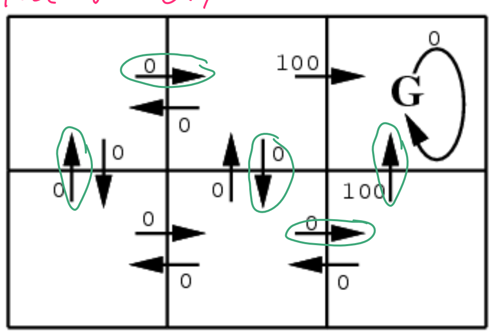
- For any MDP, such a policy exists!
- We'll abbreviate $V^{\pi^*}(s)$ as $V^*(s)$
- Note if we have $V^*(s)$ and $P(s_{t+1}|s_t,a)$, we can compute $\pi^*(s)$

Value Function – what are the $V^{\pi}(s)$ values?

$$V^{\pi}(s) = E[\sum_{t=0}^{\infty} \gamma^t r_t]$$

Suppose π is shown by circled action from each state

Suppose γ = 0.9



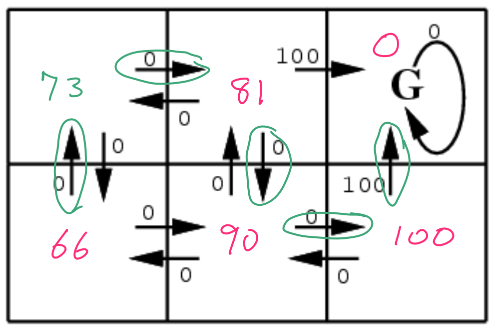
$r(s, a)$ (immediate reward)

Value Function – what are the $V^\pi(s)$ values?

$$V^\pi(s) = E[\sum_{t=0}^{\infty} \gamma^t r_t]$$

Suppose π is shown by circled action from each state

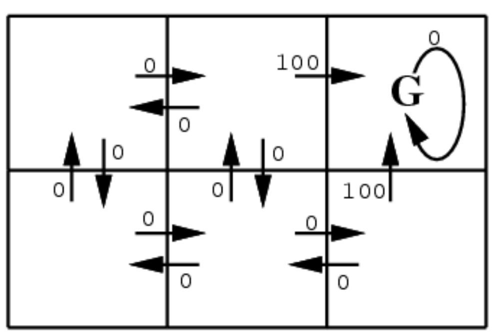
Suppose $\gamma = 0.9$



$r(s, a)$ (immediate reward)

Value Function – what are the $V^*(s)$ values?

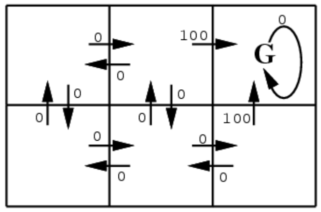
$$V^\pi(s) = E[\sum_{t=0}^{\infty} \gamma^t r_t]$$



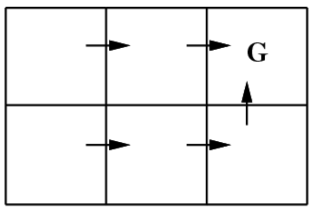
$r(s, a)$ (immediate reward)

Immediate rewards r(s,a)

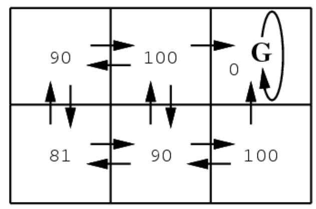
State values V*(s)



$r(s, a)$ (immediate reward) values



One optimal policy



$V^*(s)$ values

Recursive definition for V*(S)

$$V^*(s) = E[\sum_{t=0}^{\infty} \gamma^t r_t]$$

assuming actions are chosen according to the optimal policy, π^*

$$V^*(s_1) = E[r(s_1, a_1)] + E[\gamma r(s_2, a_2)] + E[\gamma^2 r(s_3, a_3)] + \ldots]$$

$$V^*(s_1) = E[r(s_1, a_1)] + \gamma E_{s_2|s_1,a_1}[V^*(s_2)]$$

$$V^*(s) = E[r(s, \pi^*(s))] + \ \gamma E_{s'|s,\pi^*(s)}[V^*(s')]$$

Value Iteration for learning V* : assumes $P(S_{t+1}|S_t, A)$ known

Initialize V(s) arbitrarily

Loop until policy good enough

Loop for s in S

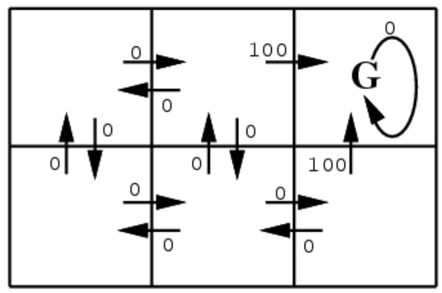
Loop for a in A

- $Q(s,a) \leftarrow r(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) V(s')$

$V(s) \leftarrow \max_a Q(s,a)$

End loop

End loop



V(s) converges to V*(s)

Dynamic programming

Value Iteration

Interestingly, value iteration works even if we randomly traverse the environment instead of looping through each state and action methodically

- but we must still visit each state infinitely often on an infinite run
- For details: [Bertsekas 1989]
- Implications: online learning as agent randomly roams

If max (over states) difference between two successive value function estimates is less than ε, then the value of the greedy policy differs from the optimal policy by no more than $2\epsilon\gamma/(1-\gamma)$

Tom Mitchell, April 2010

So far: learning optimal policy when we know $P(s_t \mid s_{t-1}, a_{t-1})$

What if we don't?

Q learning

Define new function, closely related to V*

$$V^*(s) = E[r(s, \pi^*(s))] + \gamma E_{s'|s,\pi^*(s)}[V^*(s')]$$

$$Q(s,a) = E[r(s,a)] + \gamma E_{s'|s,a}[V^*(s')]$$

If agent knows Q(s,a), it can choose optimal action without knowing $P(s_{t+1}|s_t,a)$!

$$\pi^*(s) = \arg\max_a Q(s,a) \qquad V^*(s) = \max_a Q(s,a)$$

And, it can learn Q without knowing $P(s_{t+1}|s_t,a)$

Q Function

Consider first the deterministic case. P(s'| s,a) deterministic, denoted δ(s,a)

Define new function very similar to V^*

$$Q(s,a) \equiv r(s,a) + \gamma V^*(\delta(s,a))$$

If agent learns Q, it can choose optimal action even without knowing δ!

$$\pi^*(s) = \underset{a}{\operatorname{argmax}}[r(s,a) + \gamma V^*(\delta(s,a))]$$

$$\pi^*(s) = \underset{a}{\operatorname{argmax}}\, Q(s,a)$$

Q is the evaluation function the agent will learn

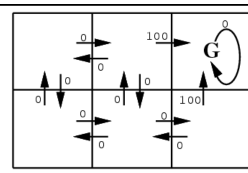
Immediate rewards r(s,a)

State values V*(s)

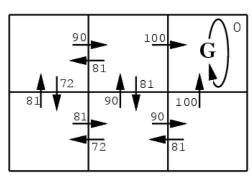
State-action values Q*(s,a)

$$V^*(s) = E[r(s,\pi^*(s))] + \gamma E_{s'|s,\pi^*(s)}[V^*(s')]$$

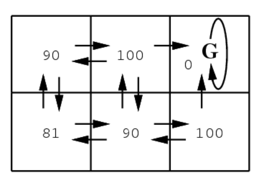
Bellman equation.



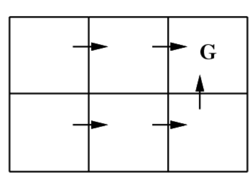
$r(s,a)$ (immediate reward) values



$Q(s,a)$ values



$V^*(s)$ values



One optimal policy

Training Rule to Learn Q

Note Q and V^* closely related:

$$V^*(s) = \max_{a'} Q(s, a')$$

Which allows us to write Q recursively as

$$\begin{aligned} Q(s_t, a_t) &= r(s_t, a_t) + \gamma V^*(\delta(s_t, a_t))) \\ &= r(s_t, a_t) + \gamma \max_{a'} Q(s_{t+1}, a') \end{aligned}$$

Nice! Let $\hat{Q}$ denote learner's current approximation to Q. Consider training rule

$$\hat{Q}(s, a) \leftarrow r + \gamma \max_{a'} \hat{Q}(s', a')$$

where s' is the state resulting from applying action a in state s

Q Learning for Deterministic Worlds

For each s, a initialize table entry $\hat{Q}(s, a) \leftarrow 0$

Observe current state s

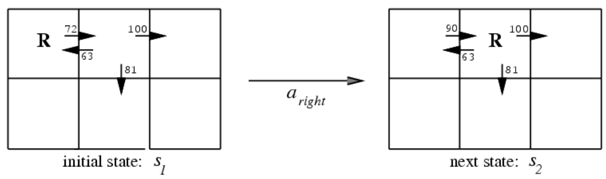
Do forever:

- Select an action a and execute it
- Receive immediate reward r
- Observe the new state s'
- Update the table entry for $\hat{Q}(s, a)$ as follows:

$$\hat{Q}(s, a) \leftarrow r + \gamma \max_{a'} \hat{Q}(s', a')$$

- $s \leftarrow s'$

Updating $\hat{Q}$



$$
\begin{aligned}
\hat{Q}(s_1, a_{right}) &\leftarrow r + \gamma \max_{a'} \hat{Q}(s_2, a') \\
&\leftarrow 0 + 0.9 \ \max\{63, 81, 100\} \\
&\leftarrow 90
\end{aligned}
$$

notice if rewards non-negative, then

$$(\forall s, a, n) \quad \hat{Q}_{n+1}(s, a) \geq \hat{Q}_n(s, a)$$

and

$$(\forall s, a, n) \quad 0 \leq \hat{Q}_n(s, a) \leq Q(s, a)$$

$\hat{Q}$ converges to Q. Consider case of deterministic world where see each $\langle s, a \rangle$ visited infinitely often.

Proof: Define a full interval to be an interval during which each $\langle s, a \rangle$ is visited. During each full interval the largest error in $\hat{Q}$ table is reduced by factor of γ

Let $\hat{Q}_n$ be table after n updates, and Δ_n be the maximum error in $\hat{Q}_n$; that is

$$\Delta_n = \max_{s,a} |\hat{Q}_n(s, a) - Q(s, a)|$$

For any table entry $\hat{Q}_n(s, a)$ updated on iteration $n + 1$, the error in the revised estimate $\hat{Q}_{n+1}(s, a)$ is

$$
\begin{aligned}
|\hat{Q}_{n+1}(s, a) - Q(s, a)| &= |(r + \gamma \max_{a'} \hat{Q}_n(s', a')) \\
&\quad -(r + \gamma \max_{a'} Q(s', a'))| \\
&= \gamma |\max_{a'} \hat{Q}_n(s', a') - \max_{a'} Q(s', a')| \\
&\leq \gamma \max_{a'} |\hat{Q}_n(s', a') - Q(s', a')| \\
&\leq \gamma \max_{s'', a'} |\hat{Q}_n(s'', a') - Q(s'', a')| \\
|\hat{Q}_{n+1}(s, a) - Q(s, a)| &\leq \gamma \Delta_n
\end{aligned}
$$

Use general fact:

$$|\max_a f_1(a) - \max_a f_2(a)| \leq \max_a |f_1(a) - f_2(a)|$$

Nondeterministic Case

Q learning generalizes to nondeterministic worlds

Alter training rule to

$$\hat{Q}_n(s,a) \leftarrow (1-\alpha_n)\hat{Q}_{n-1}(s,a)+\alpha_n[r+\max_{a'}\hat{Q}_{n-1}(s',a')]$$

where

$$\alpha_n = \frac{1}{1+visits_n(s,a)}$$

Can still prove convergence of $\hat{Q}$ to Q [Watkins and Dayan, 1992]

Tom Mitchell, April 2010

Temporal Difference Learning

Q learning: reduce discrepancy between successive Q estimates

One step time difference:

$$Q^{(1)}(s_t,a_t) \equiv r_t + \gamma \max_a \hat{Q}(s_{t+1},a)$$

Why not two steps?

$$Q^{(2)}(s_t,a_t) \equiv r_t + \gamma r_{t+1} + \gamma^2 \max_a \hat{Q}(s_{t+2},a)$$

Or n?

$$Q^{(n)}(s_t,a_t) \equiv r_t+\gamma r_{t+1}+\cdots+\gamma^{(n-1)} r_{t+n-1}+\gamma^n \max_a \hat{Q}(s_{t+n},a)$$

Blend all of these:

$$Q^{\lambda}(s_t,a_t) \equiv (1-\lambda)\left[Q^{(1)}(s_t,a_t) + \lambda Q^{(2)}(s_t,a_t) + \lambda^2 Q^{(3)}(s_t,a_t)\right.$$

Temporal Difference Learning

$$Q^\lambda(s_t, a_t) \equiv (1-\lambda)\left[Q^{(1)}(s_t, a_t) + \lambda Q^{(2)}(s_t, a_t) + \lambda^2 Q^{(3)}(s_t, a_t)\right.$$

Equivalent expression:

$$\begin{aligned} Q^\lambda(s_t, a_t) \;&= r_t + \gamma[\;(1-\lambda)\max_a \hat{Q}(s_t, a_t) \\ &\quad +\lambda\; Q^\lambda(s_{t+1}, a_{t+1})] \end{aligned}$$

TD(λ) algorithm uses above training rule

- Sometimes converges faster than Q learning
- converges for learning V^* for any $0 \le \lambda \le 1$ (Dayan, 1992)
- Tesauro's TD-Gammon uses this algorithm

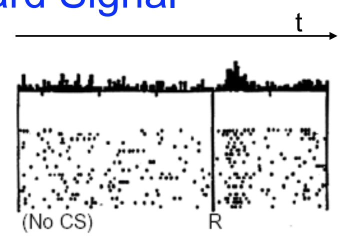
MDPs and Reinforcement Learning: Further Issues

- What strategy for choosing actions will optimize
  - learning rate? (*explore* uninvestigated states)
  - obtained reward? (*exploit* what you know so far)
- Can we bound sample complexity?
  - R-Max learns with δ, ε bounds in polynomial number of actions
- *Partially observable* Markov Decision Processes
  - state is not fully observable
  - must maintain probability distribution over possible state you're in
- Convergence guarantee with function approximators?
- Correspondence to human learning?

Dopamine As Reward Signal

[Schultz et al., *Science*, 1997]

No prediction
Reward occurs

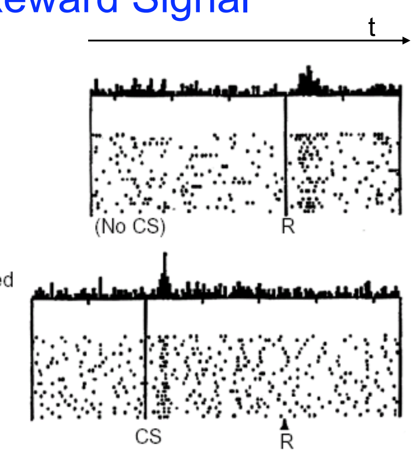


Dopamine As Reward Signal

[Schultz et al., *Science*, 1997]

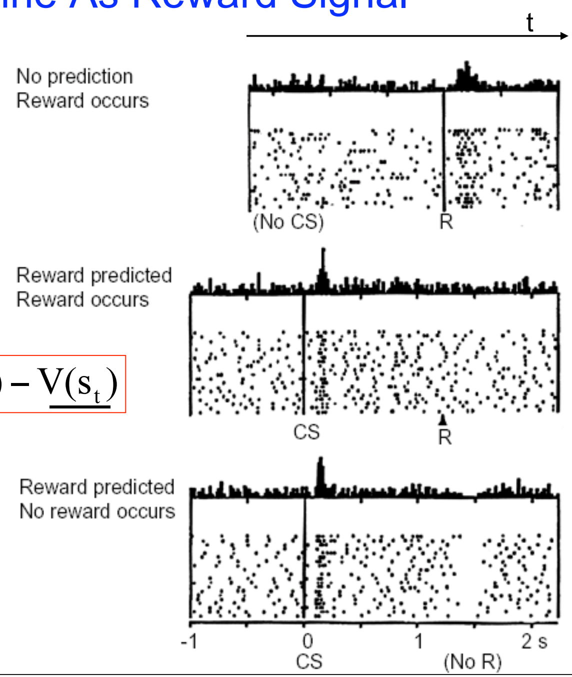
No prediction
Reward occurs

Reward predicted
Reward occurs



Dopamine As Reward Signal

[Schultz et al., *Science*, 1997]



$$\text{error} = \underline{r_t + \gamma V(s_{t+1})} - \underline{V(s_t)}$$

RL Models for Human Learning

[Seymore et al., Nature 2004]

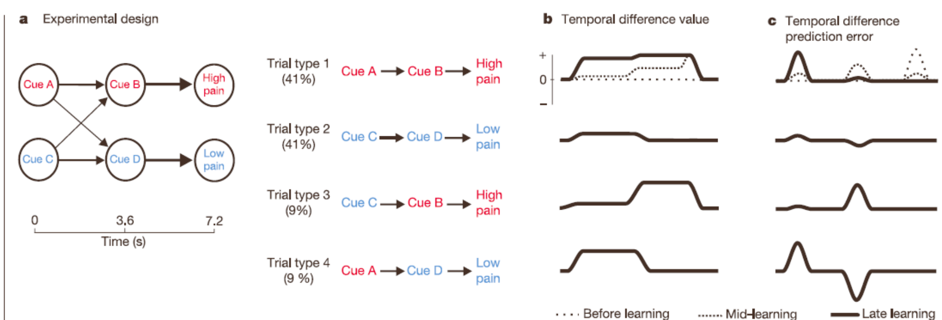
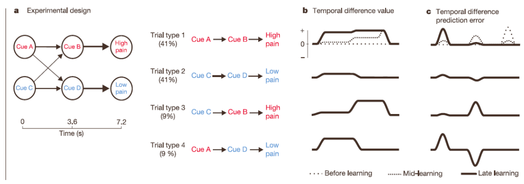


Figure 1 Experimental design and temporal difference model. **a**, The experimental design expressed as a Markov chain, giving four separate trial types. **b**, Temporal difference value. As learning proceeds, earlier cues learn to make accurate value predictions (that is, weighted averages of the final expected pain). **c**, Temporal difference prediction error; during learning the prediction error is transferred to earlier cues as they acquire the ability to make predictions. In trial types 3 and 4, the substantial change in prediction elicits a large positive or negative prediction error. (For clarity, before and mid-learning are shown only for trial type 1.)



[Seymore et al., Nature 2004]

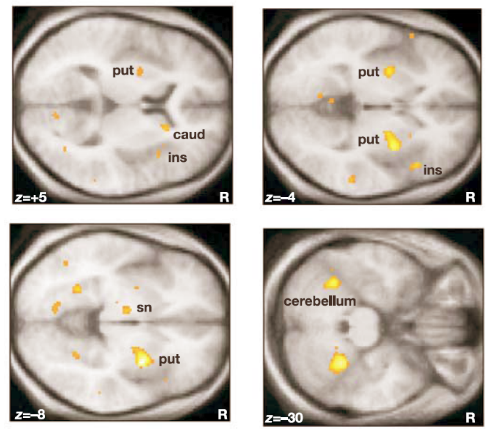
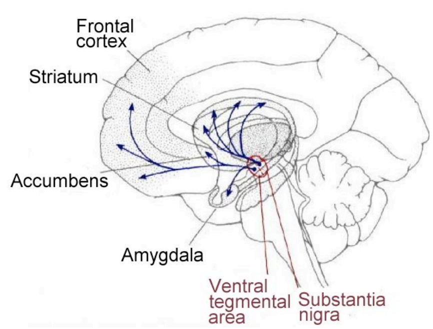


Figure 2 Temporal difference prediction error (statistical parametric maps). Areas coloured yellow/orange show significant correlation with the temporal difference

One Theory of RL in the Brain

from [Nieuwenhuis et al.]

- Basal ganglia monitor events, predict future rewards
- When prediction revised upward (downward), causes increase (decrease) in activity of midbrain dopaminergic neurons, influencing ACC
- This dopamine-based activation somehow results in revising the reward prediction function. Possibly through direct influence on Basal ganglia, and via prefrontal cortex



Summary: Temporal Difference ML Model Predicts Dopaminergic Neuron Acitivity during Learning

- Evidence now of neural reward signals from
  - Direct neural recordings in monkeys
  - fMRI in humans (1 mm spatial resolution)
  - EEG in humans (1-10 msec temporal resolution)

- Dopaminergic responses encode Bellman error

- Some differences, and efforts to refine the model
  - How/where is the value function encoded in the brain?
  - Study timing (e.g., basal ganglia learns faster than PFC ?)
  - Role of prior knowledge, rehearsal of experience, multi-task learning?