Reinforcement Learning

CS786

27th August 2024

MDP→RL

- In MDP, {S,A,R,P} are known
- In RL, R and P are not known to begin with
- They are *learned* from experience
- Optimal policy is updated sequentially to account for increased information about rewards and transition probabilities
- Model-based RL
  - Learns transition probabilities P as well as optimal policy
- Model-free RL
  - Learns only optimal policy, not the transition probabilities P

Q-learning

- Derived from the Bush-Mosteller update rule
- Agent sees a set of states S
- Possesses a set of A actions applicable to these states
- Does <u>not</u> try to learn p(s, a, s')
- Tries to learn a quality belief about a state-action combination Q: S X A → Real

Q-learning update rule

- Start with random Q
- Update using

$$Q_{new}(s,a) = (1-\alpha)Q_{old}(s,a) + \alpha(r + \lambda \max_{a'} Q(s',a'))$$

- Parameter α controls the learning rate
- Parameter λ controls the time-discounting of future reward

Q-learning

- Agent sees a set of states S
- Possesses a set of A actions applicable to these states
- Does <u>not</u> try to learn p(s, a, s')
- Tries to learn a quality belief about a state-action combination Q: S X A → Real

Q-learning update rule

- Start with random Q
- Update using

$$Q_{new}(s,a) = (1-\alpha)Q_{old}(s,a) + \alpha(r + \lambda \max_{a'} Q(s',a'))$$

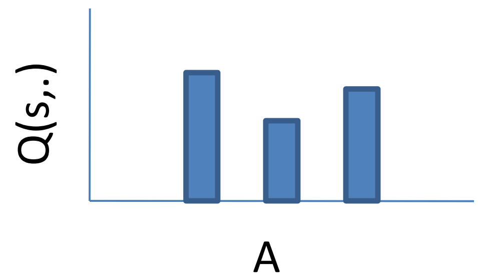
- Parameter α controls the learning rate
- Parameter λ controls the time-discounting of future reward
- s’ is the state accessed from s
- a’ are actions available in s’

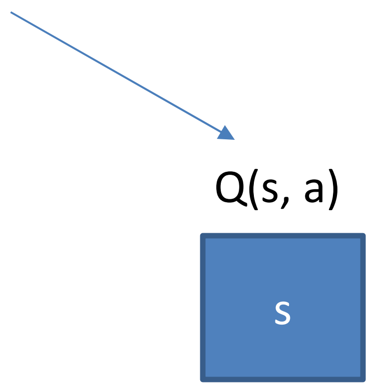
Q-learning algorithm

- Initialize Q(s,a) for all s and a
- For each episode
  - Initialize s
  - For each move
    - Choose a from s using Q (softmax/e-greedy)
    - Perform action a, observe R and s’
    - Update Q(s,a)
    - Move to s’
  - Until s’ is terminal/moves run out

Q-learning update

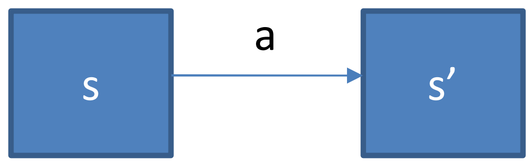
The value of taking action a in state s





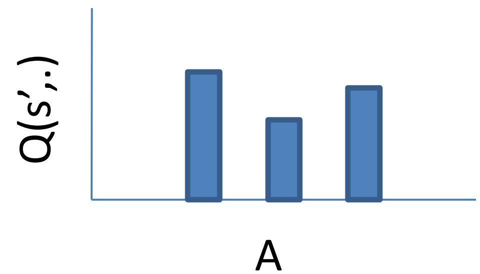
1. Select a using choice rule on Q

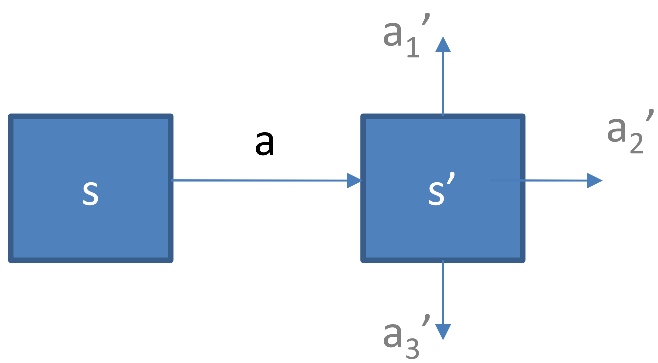
Q-learning update



1. Select a using choice rule on Q
2. Take action a from state s
3. Observe r and s'

Q-learning update

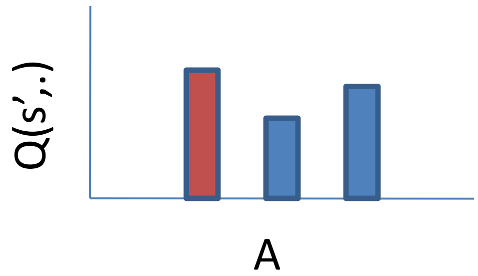


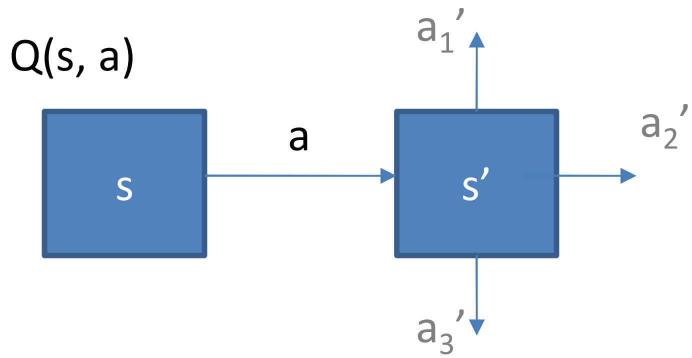


There are many possible a’ from the state you reach

1. Select a using choice rule on Q
2. Take action a from state s
3. Observe r and s’
4. Recall Q(s’,a’) for all a’ available from s’

Q-learning update





Assume maximally rewarding action will be selected at s'

1. Select a using choice rule on Q
2. Take action a from state s
3. Observe r and s'
4. Recall Q(s',a') for all a' available from s'
5. Update Q(s,a)

$$Q_{new}(s,a) = (1-\alpha)Q_{old}(s,a) + \alpha(r + \lambda \max_{a'} Q(s',a'))$$

Q-learning example

- Open AI gym's frozen lake
- Setup: agent is a character that has to walk from a start point (S) across a frozen lake (F) with holes (H) in some locations to reach G
- Specific instantiation

S	F	F	F
F	H	F	H
F	F	F	H
H	F	F	G

Q-learning example

- Agent starts with an empty Q-matrix
- Action possibilities = {left, right, up, down}
- Reward settings
  - H = -100
  - G = +100
  - F = 0

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

Q-learning example

- Learning occurs via exploration episodes
- One episode is a sequence of moves
- Let's work through one episode

S $\xrightarrow{0}$	F	F	F
F	H	F	H
F	F	F	H
H	F	F	G

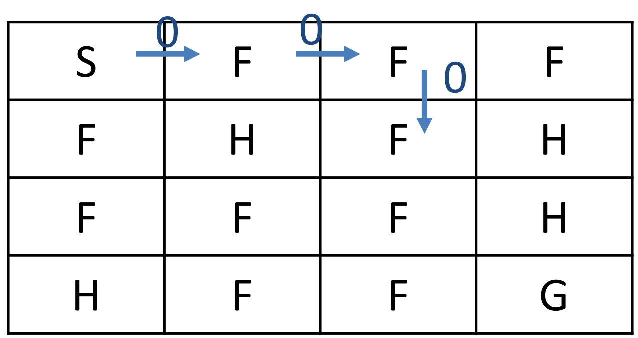
Q-learning example

- Learning occurs via exploration episodes
- One episode is a sequence of moves
- Let’s work through one episode

S $\xrightarrow{0}$	F $\xrightarrow{0}$	F	F
F	H	F	H
F	F	F	H
H	F	F	G

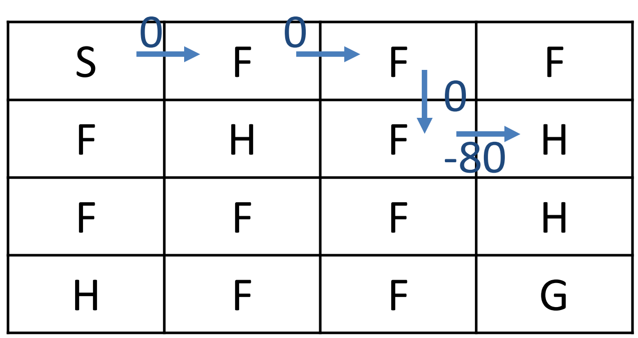
Q-learning example

- Learning occurs via exploration episodes
- One episode is a sequence of moves
- Let's work through one episode



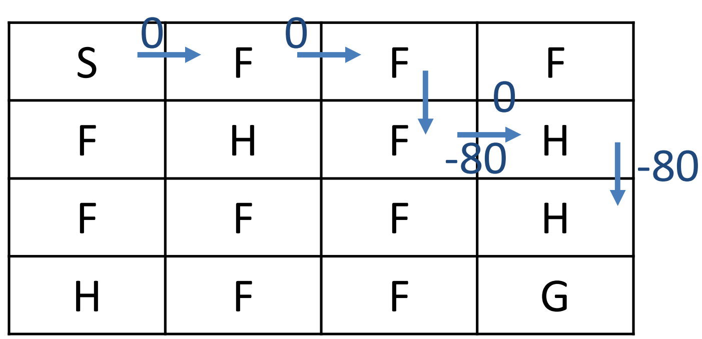
Q-learning example

- Learning occurs via exploration episodes
- One episode is a sequence of moves
- Let's work through one episode



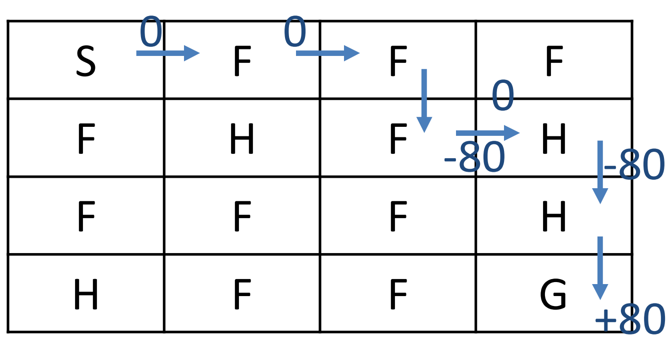
Q-learning example

- Learning occurs via exploration episodes
- One episode is a sequence of moves
- Let’s work through one episode



Q-learning example

- Learning occurs via exploration episodes
- One episode is a sequence of moves
- Let’s work through one episode



Generalized model-free RL

- Bush Mosteller style models simply update value based on a discounted average of received rewards
  - Useless in trying to predict the value of sequential events, e.g. A →B → reward
- A more generalized notion of reward learning was needed
  - Q-learning is one instance of temporal difference learning
  - Other flavors of model-free reinforcement learning also exist, e.g. policy gradient methods

SARSA update rule

- Start with random Q
- Update using

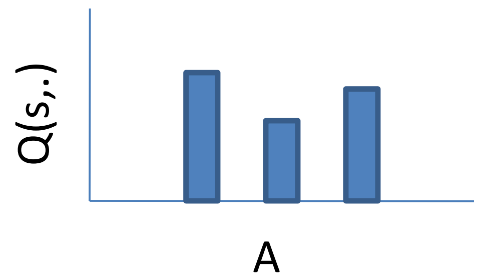
$$Q_{new}(s,a) = (1-\alpha)Q_{old}(s,a) + \alpha(r + \lambda Q_{old}(s',a'))$$

- Parameter α controls the learning rate
- Parameter λ controls the time-discounting of future reward
- s' is the state accessed from s
- **a' is the action selected in s'**
  – Different from q-learning

SARSA algorithm

- Start with random Q(s, a) for all s and a
- For each episode
  - Initialize s
  - Choose a using Q (softmax/greedy)
  - For each move
    - Take action a, observe r, s’
    - Choose a’ from s’ by comparing Q(s’, .)
    - Update Q(s, a)
    - Move to s’, remember a’
  - Until s’ is terminal/moves run out

SARSA update

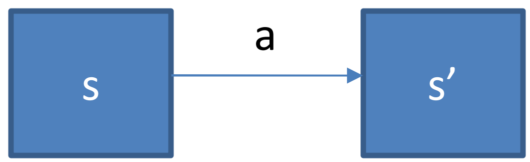


The value of taking action a in state s

Q(s, a)

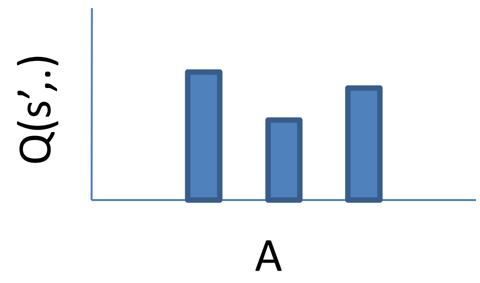
1. Start with a selected in the previous iteration

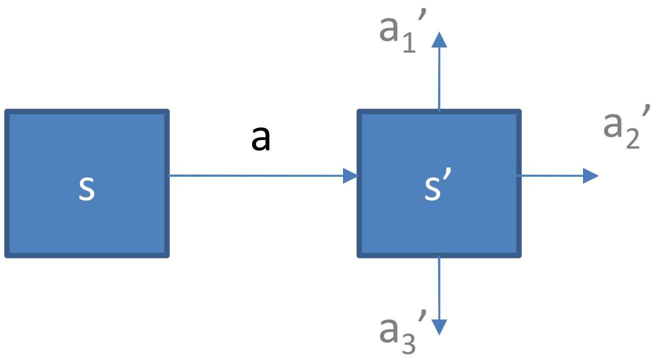
SARSA update



1. Start with a selected in the previous iteration
2. Take action a from state s
3. Observe r and s’

SARSA update

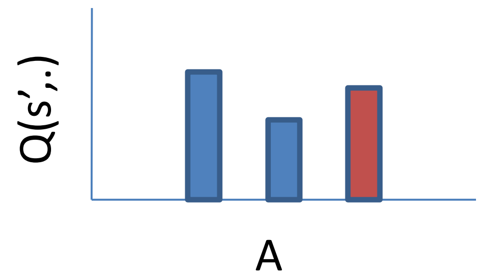


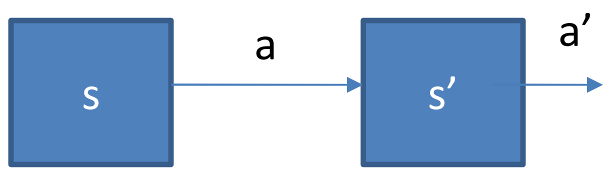


There are many possible a' from the state you reach

1. Start with a from the previous iteration
2. Take action a from state s
3. Observe r and s'
4. Recall Q(s',a') for all a' available from s'

SARSA update





a’ is selected using the choice rule

1. Start with a from the previous iteration
2. Take action a from state s
3. Observe r and s’
4. Recall Q(s’,a’) for all a’ available from s’
5. Select a’ using choice rule on Q
6. Update Q(s,a)

$$Q_{new}(s,a) = (1-\alpha)Q_{old}(s,a) + \alpha(r + \lambda Q_{old}(s',a'))$$

RL in the brain

Temporal difference learning

- Consider the Q-learning update

$$Q_{new} = Q_{old} + \alpha(r + \lambda \max_{a'} Q(s', a') - Q(s, a))$$

- Or the SARSA update

$$Q_{new} = Q_{old} + \alpha(r + Q(s', a') - Q(s, a))$$

- A generic temporal difference principle can be discerned for behavioral reinforcement

$$V(t+1) = V(t) + \alpha(r + V(s(t+1)) - V(s(t))$$

The TD learning principle

- Bush Mosteller algorithm, with V as learned reinforcement value

$$V(t+1) = V(t) + \alpha(R(t) - V(t))$$

- Temporal difference learning

$$V(t+1) = V(t) + \alpha(F(t) - V(t))$$

- Discounted future rewards not available instantaneously
  - Use Bellman optimality principle

$$F(t) = r(t+1) + \lambda F(t+1)$$

Reinterpreting the learning gradient

- In Bush Mosteller, the reward prediction error is driven by the difference between
  - A discounted average of received rewards
  - The current reward
- In TD learning, RPE is the difference between
  - Expected value of discounted future rewards F

    $$F(t) = r(t+1) + \lambda r(t+2) + \lambda R(t+3) + \cdots$$
  - Information suggesting the expectation is mistaken

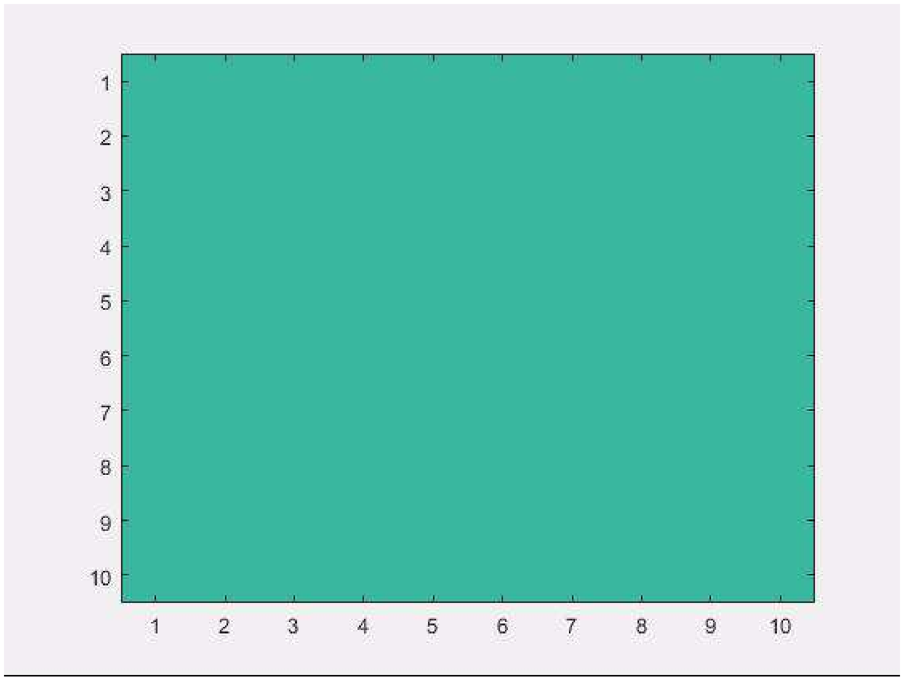
http://www.scholarpedia.org/article/Temporal_difference_learning

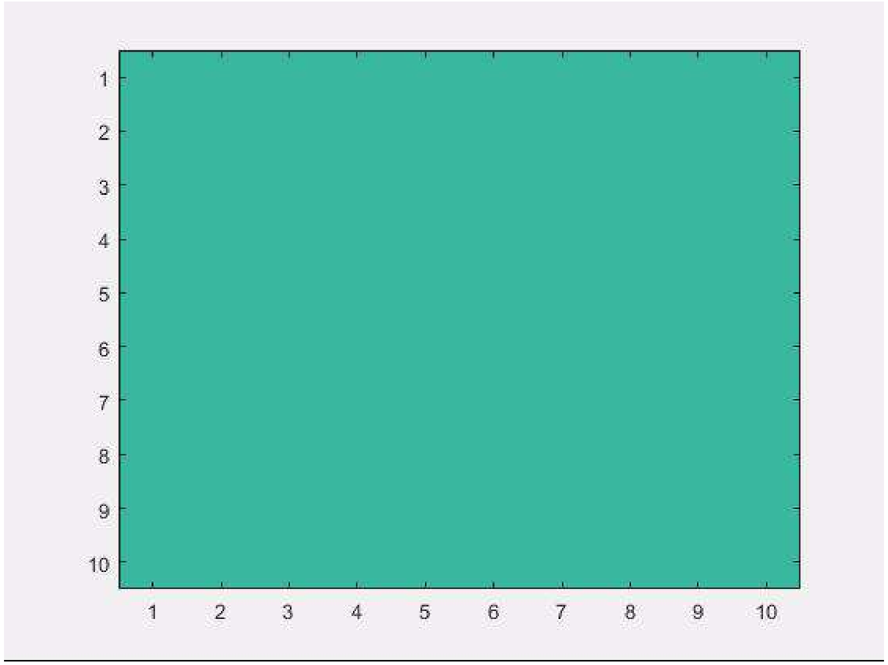
The TD reward prediction error

$$\delta(t) = R + \lambda V(s(t+1)) - V(s(t))$$

Learning continues until reward expectations are perfectly aligned with received reward

The role of the future



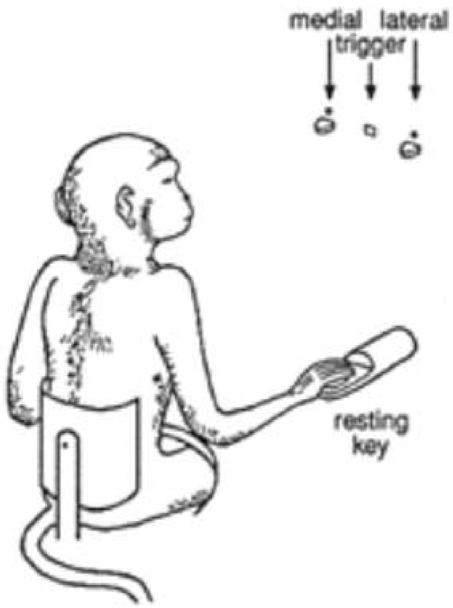


Myopic learning (λ = 0)

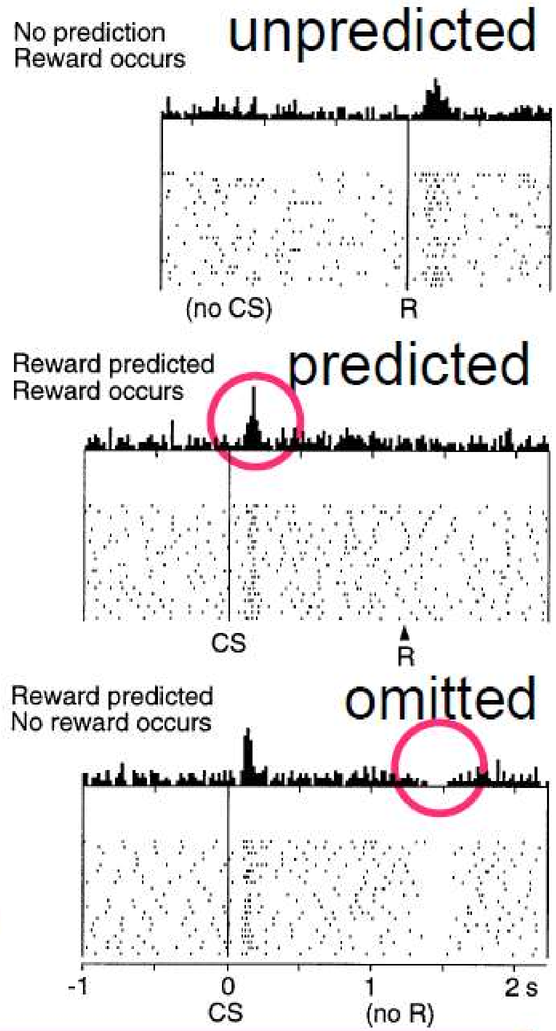
Future-sensitive learning (λ > 0)

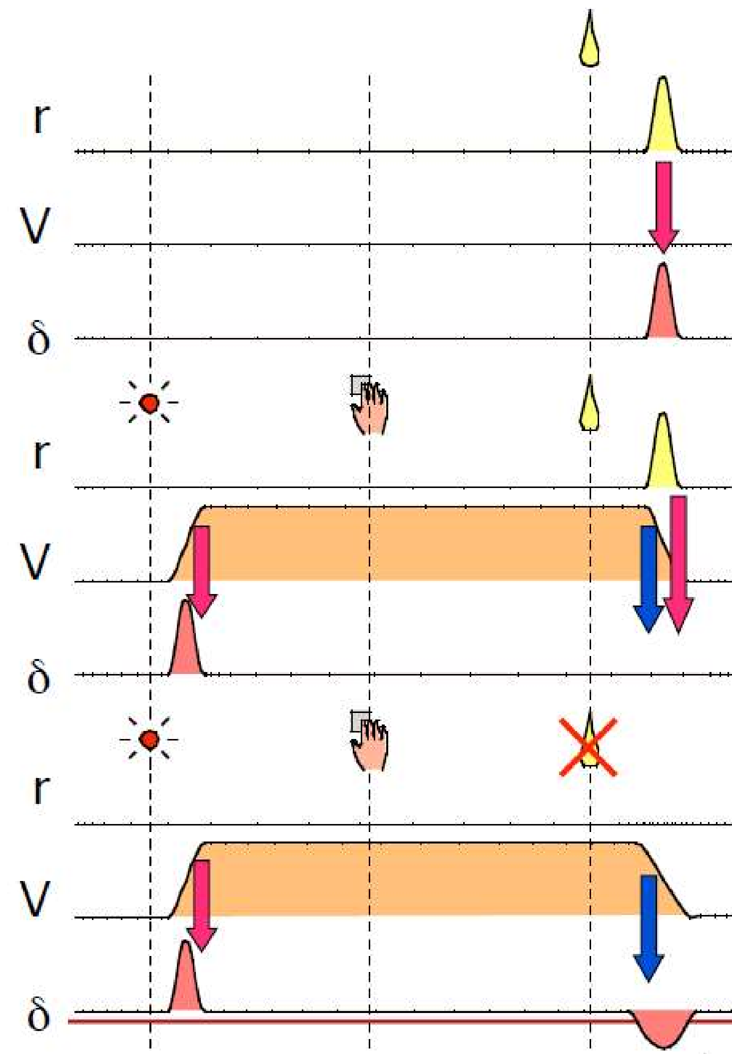
Dopamine Neurons Code TD Error

$$\delta(t) = r(t) + \gamma V(s(t+1)) - V(s(t))$$



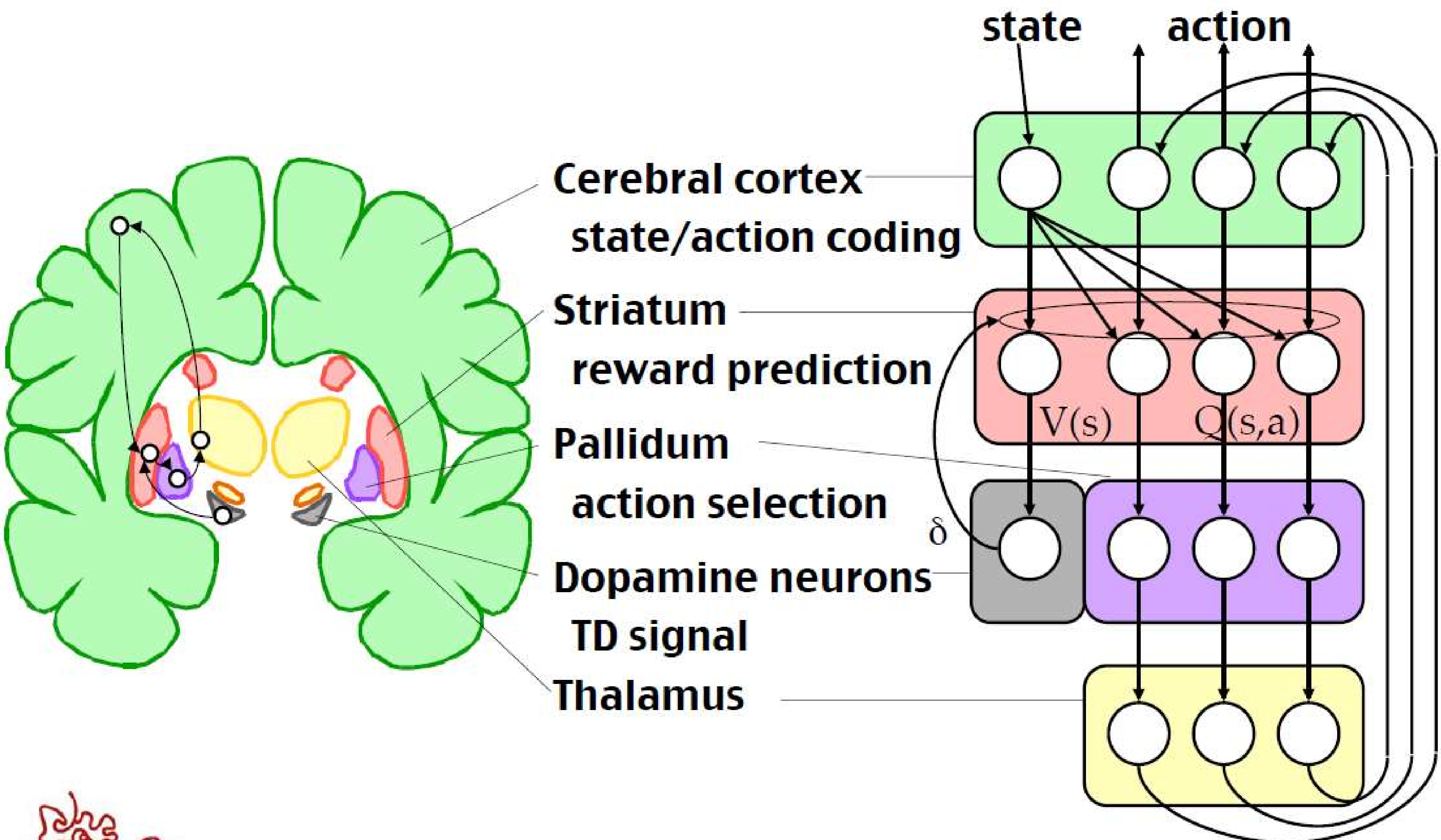
(Schultz et al. 1997)





Basal Ganglia for Reinforcement Learning?

(Doya 2000, 2007)



Cocaine addiction (a success story)

- Cocaine pharmacodynamics
  - Is a dopamine reuptake inhibitor
- Under normal circumstances the TD signal is

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

- When you take cocaine

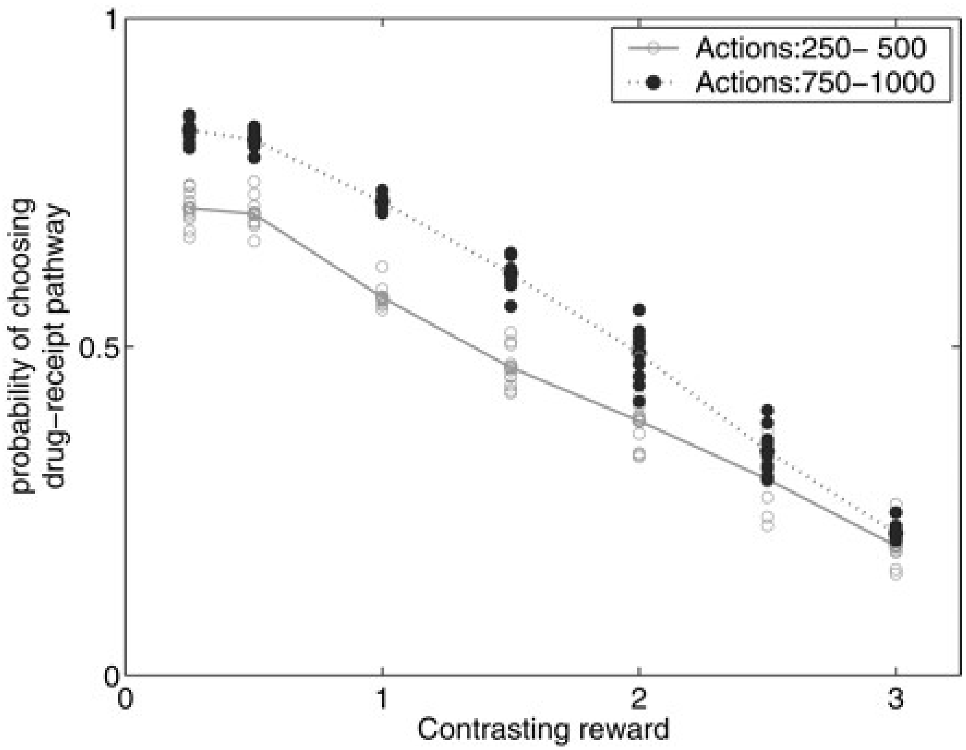
$$\delta_t = \max\left\{r_{t+1} + \gamma V\left(s_{t+1}\right) - V\left(s_t\right) + D_t, D_t\right\}$$

The mechanics of physical addiction

- In the beginning, taking cocaine is associated with positive TD signal
  - So taking cocaine is learned
- But presence of cocaine in the system prevents the TD signal from becoming negative
  - No matter what you do
  - Behavior cannot be unlearned!

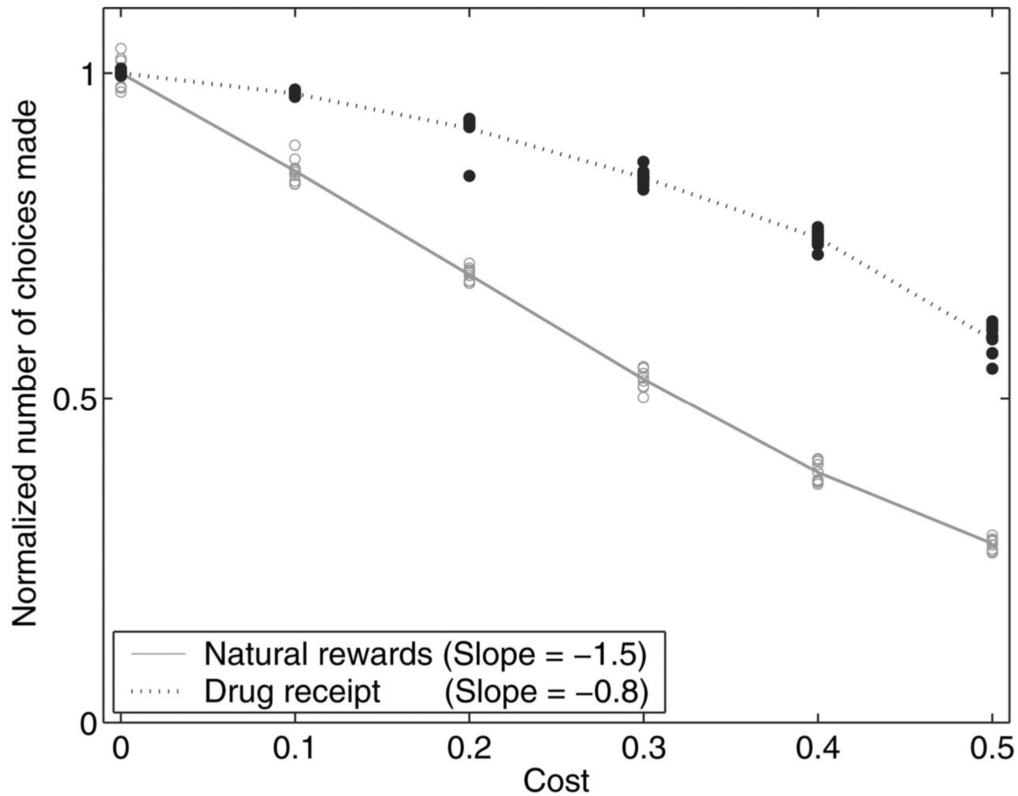
Reward insensitivity

- Observer will become unable to tradeoff drug consumption with other rewards



Cost insensitivity

- Observe is unable to reduce preference with increasing cost



Cocaine addiction (a success story)

- Cocaine pharmacodynamics
  - Is a dopamine reuptake inhibitor
- Under normal circumstances the TD signal is

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

- When you take cocaine

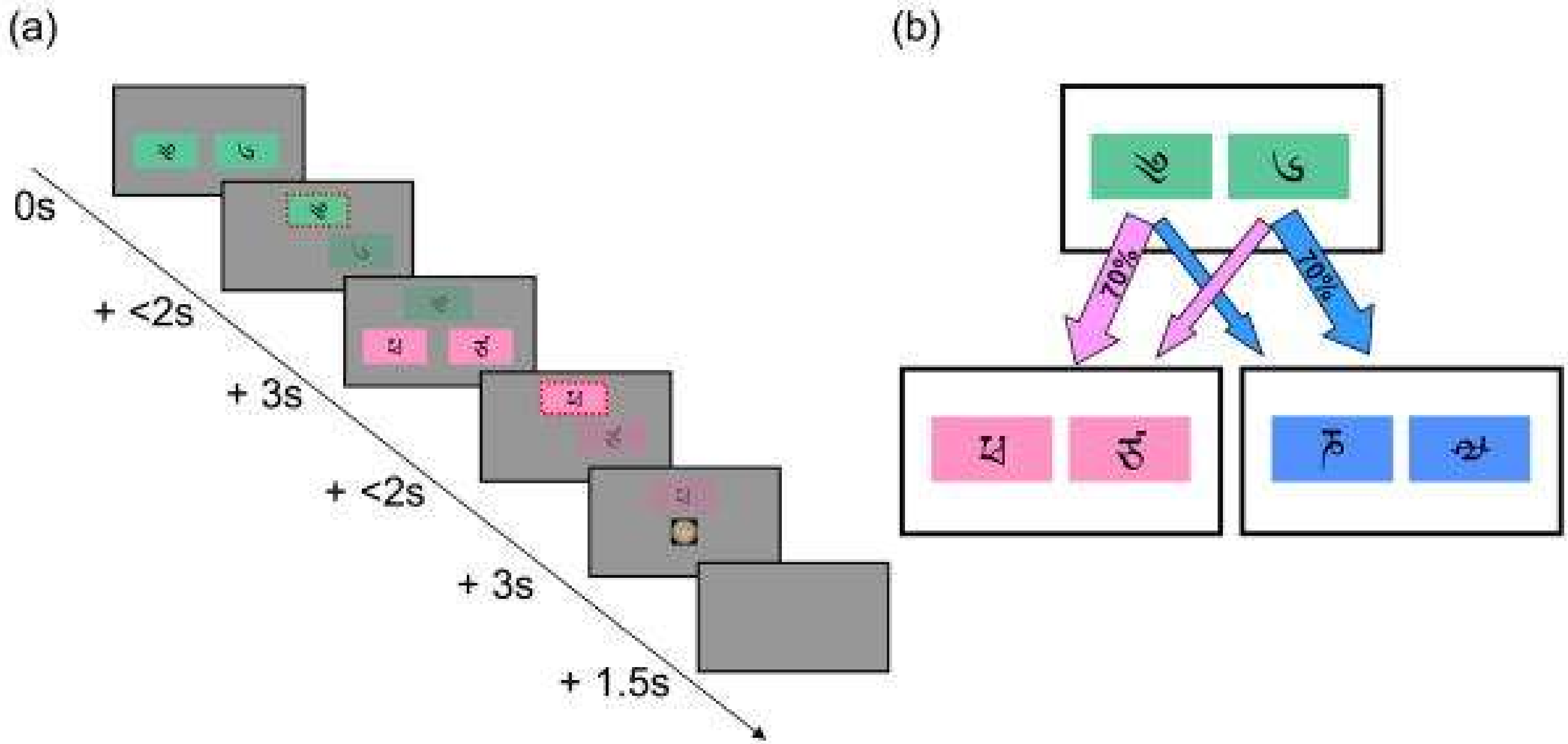
$$\delta_t = \max\left\{r_{t+1} + \gamma V\left(s_{t+1}\right) - V\left(s_t\right) + D_t, D_t\right\}$$

Addiction: a computational process gone awry *(Redish, 2004)*

The model free vs model-based debate

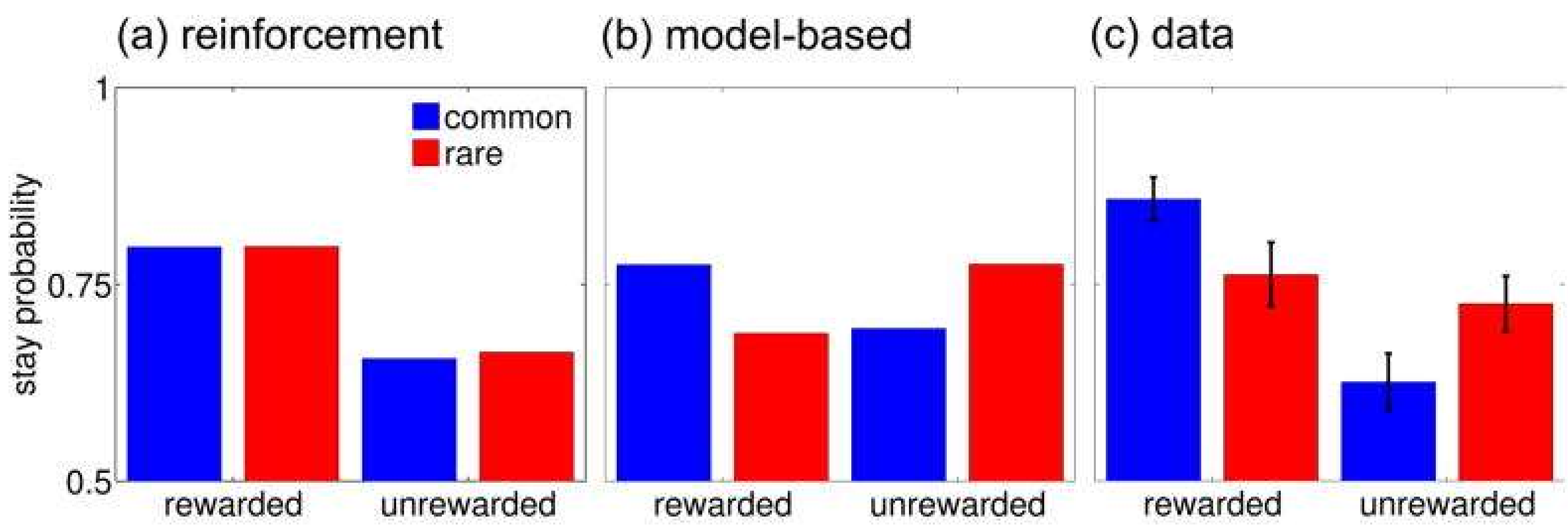
- Model free learning → actions that lead to rewards become more preferable
- What about goal-based decision-making?
  - Do animals not learn the physics of the world in making decisions?
- Model-based learning
- People have argued for two systems
  - Thinking fast and slow (Balleine & O'Doherty, 2010)

A clever experiment



- The Daw task (Daw et al, 2011) is a two-stage Markov decision task
- Differentiates model-based and model-free accounts empirically

Predictions meet data



- Behavior appears to be a mix of both strategies
- What does this mean?
- Active area of research