Prioritized memory access explains planning and hippocampal replay

Mattar and Daw (2017)

UCL CompHip. Journal Club

December 13, 2017

Intro: Reinforcement learning recap

s : State
a : Action
R : Reward

$\pi(a,s)$: Policy (action given state)
γ : Discount factor
q_π : Expected return

Goal: To select actions that maximize the expected cumulative discounted reward

$$(1) \quad G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ \dots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

Reward at time t+1

Cumulative discounted return

State at time t = s

$$(2) \quad q_\pi(s,a) = \ \mathrm{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s, A_t = a\right]$$

Action at time t = a

Expected discounted return from following policy q_π until end

$$(3) \quad Q(S_t,A_t) \leftarrow Q(S_t,A_t) + \alpha\left[R_t + \gamma \max_{a \in A} Q(S_{t+1},a) - Q(S_t,A_t)\right]$$

Learned action-value function

New value

Old value

Update policy towards action that produces max. return

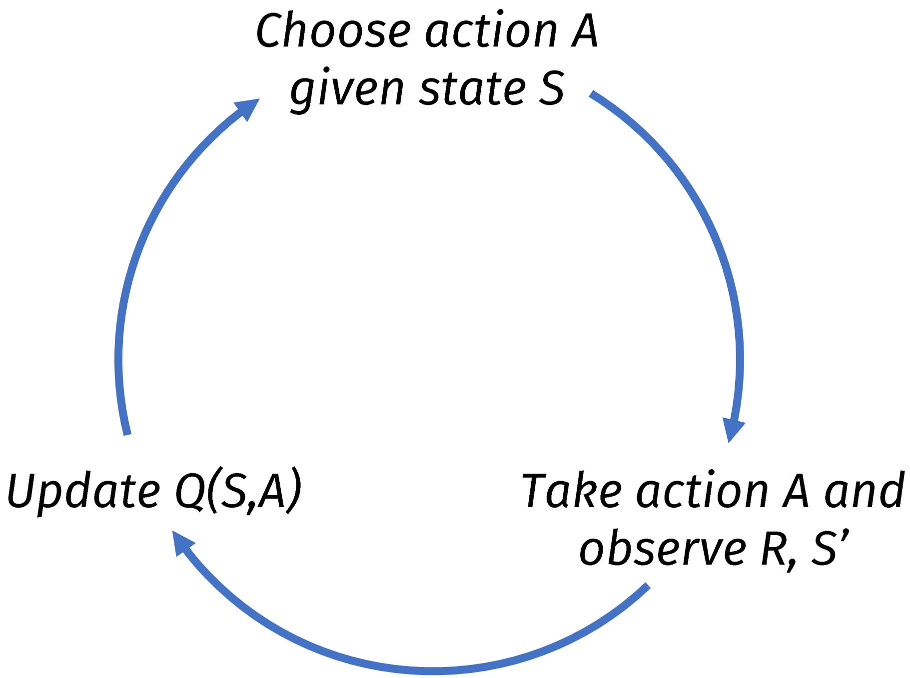
Intro: Reinforcement learning recap

s : State
a : Action
R : Reward

$\pi(a,s)$: Policy (action given state)
γ : Discount factor
q_π : Expected return

Goal: To select actions that maximize the expected cumulative discounted reward

Repeat until end:



Intro: DYNA

s : State
a : Action
R : Reward

$\pi(a, s)$: Policy (action given state)
γ : Discount factor
q_π : Expected return

Goal: To select actions that maximize the expected cumulative discounted reward

Model-Free RL:
- No model
- **Learn** value function (and/or policy) from experience
- Fast
- Can't update policy to reflect change in environment

Model-Based RL:
- Learn a model from experience
- **Plan** value function (and/or policy) from model
- Slower than model-free
- Changes in the environment reflected in planning

Dyna:
- Learn a model from real experience
- **Learn and plan** value function (and/or policy) from real and simulated experience
- Fusion of model-based / model-free methods

Intro: DYNA

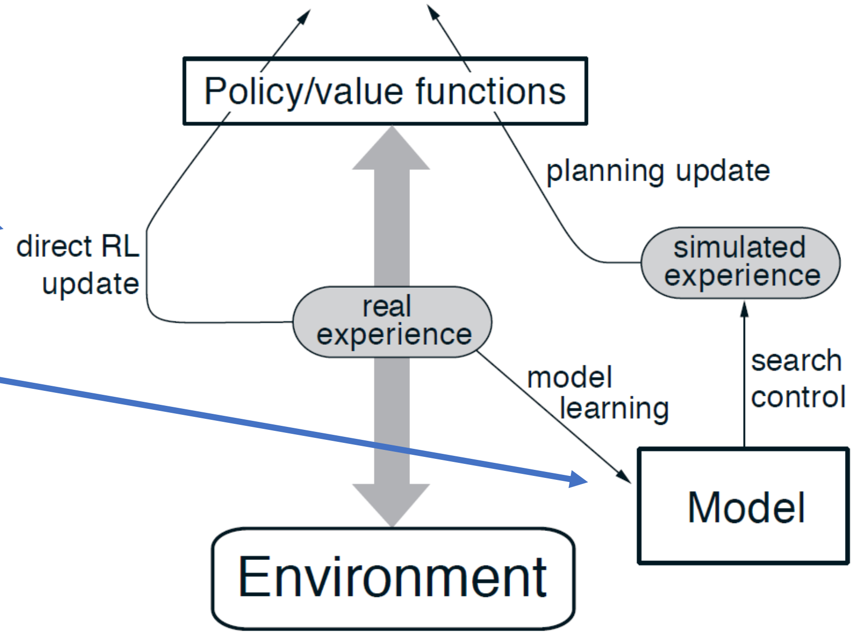
s : State
a : Action
R : Reward

$\pi(a, s)$: Policy (action given state)
γ : Discount factor
q_π : Expected return

Goal: To select actions that maximize the expected cumulative discounted reward

Learn from two sources:

1. Real experience (animal movement)
2. Simulate experience (replay) from learned model



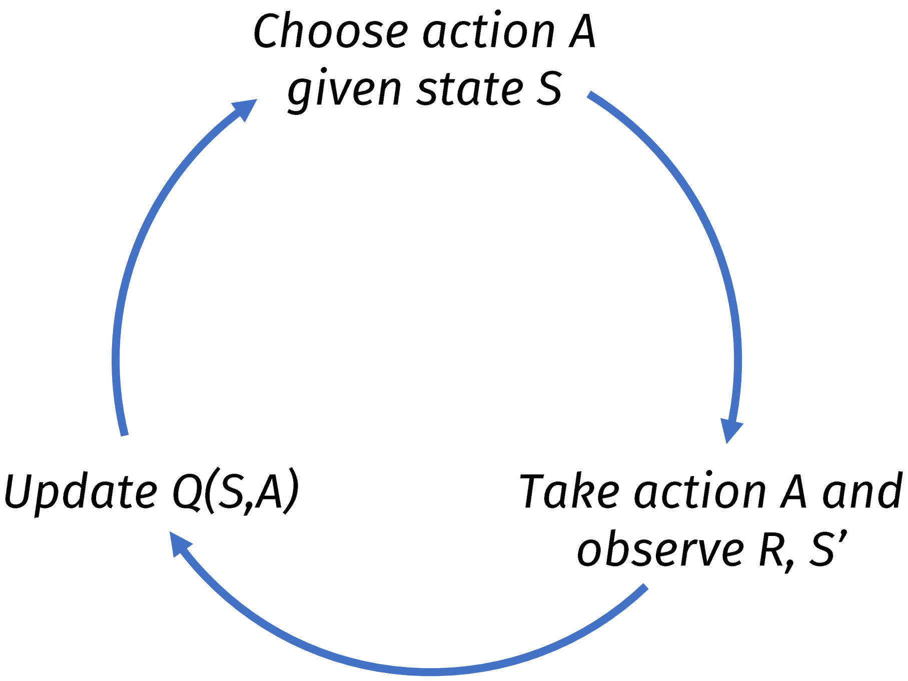
Model = transition structure of environment

Intro: DYNA

These two modes of learning can operate in parallel, or sequentially:

One run is equivalent to a two-state 'replay event'

Direct Experience



Simulated experience using model

Repeat:

1. *Choose random previous state S*
2. *Choose random action A previously taken at S*
3. *Transition to S' given model*
4. *Update Q(S,A)*

Intro: DYNA

s : State
a : Action
R : Reward

$\pi(a, s)$: Policy (action given state)
γ : Discount factor
q_π : Expected return

One run is equivalent to
a two-state 'replay event'

Simulated experience using model

Repeat:

1. Choose random previous state S

2. Choose random action A previously taken at S

3. Transition to S' given model

4. Update Q(S,A)

- Mattar and Daw propose a better way to select which states are updated

- Instead, prioritize states based on gain and need

Intro: Prioritized memory access (PME)

s : State
a : Action
R : Reward

$\pi(a, s)$: Policy (action given state)
γ : Discount factor
q_π : Expected return

Goal: To select actions that maximize the expected cumulative discounted reward

$$e_k = (s_k, a_k, r_k, s_{k+1})$$

A single experience (replay event)

$$Utility(s_k, a_k) = Gain \times Need$$

Choose the experience e_k with the highest expected utility

$$\mathrm{Gain}(\mathrm{s_k}, \mathrm{a_k}) = \begin{matrix} \textit{Expected returns} \\ \textit{under updated policy} \end{matrix} - \begin{matrix} \textit{Expected returns} \\ \textit{under old policy} \end{matrix}$$

i.e. the increase in expected returns resulting from an update to state-action value $Q(s_k, a_k)$

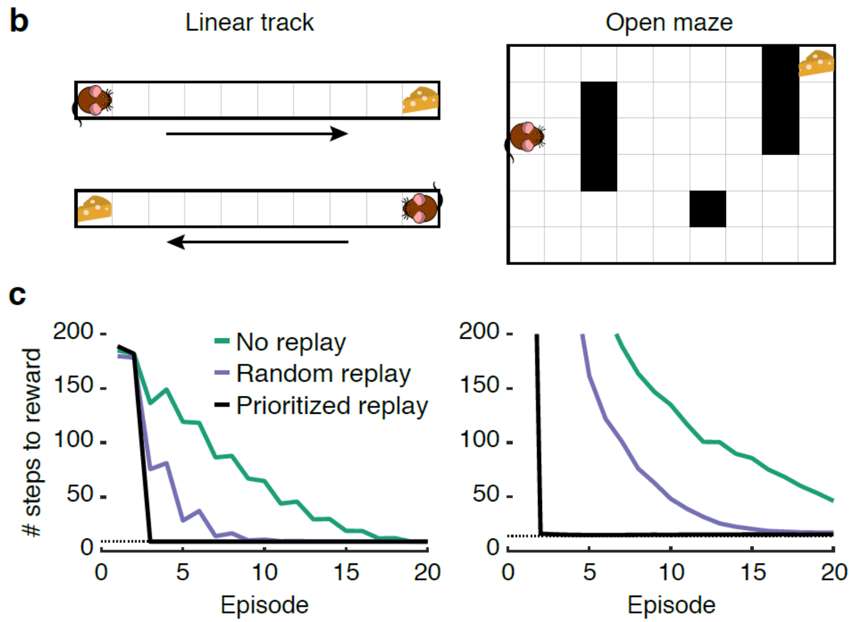
$$\mathrm{Need}(\mathrm{s_k}) = \textit{Discounted number of future visits to state } s_k$$

(This can be approximated by the successor representation)

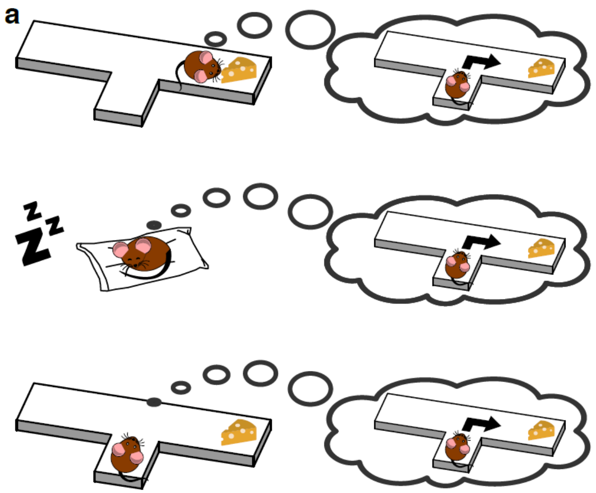
Results 1: PME access improves learning speed

- ‘Grid-world’ environments – move UP, DOWN, LEFT, RIGHT
  - Environment 1: Linear track with end reward
  - Environment 2: 2D room with objects and reward

- Softmax action decisions at each state (balance of exploration vs. exploitation)



Three ways to learn:

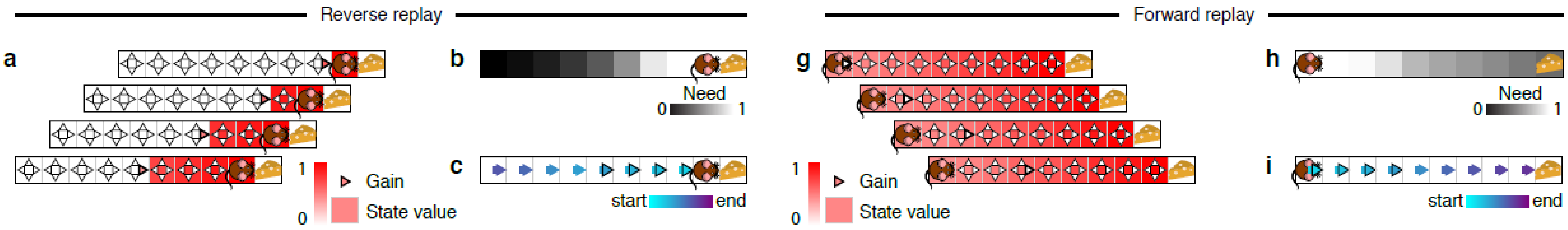


Reverse replay after reward

‘Offline’ reactivation

Prospective activation before choice

Results 2: Forward and reverse replay



Prediction errors (unexpected / changed reward) cause large gain term directly behind the animal

Need term larger in states ahead of the animal

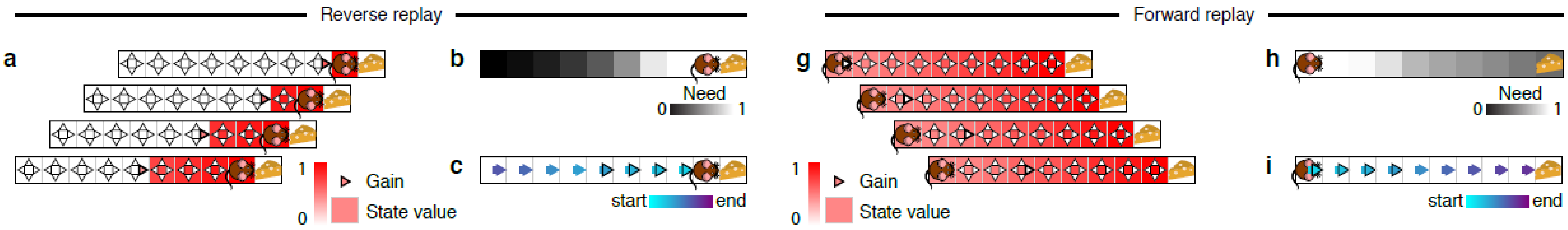
Changes to this state directly affect the probability of collecting the reward in the future

Need term (expected discounted visits) related to previous trajectories, which tend to lead to the goal as policies improve

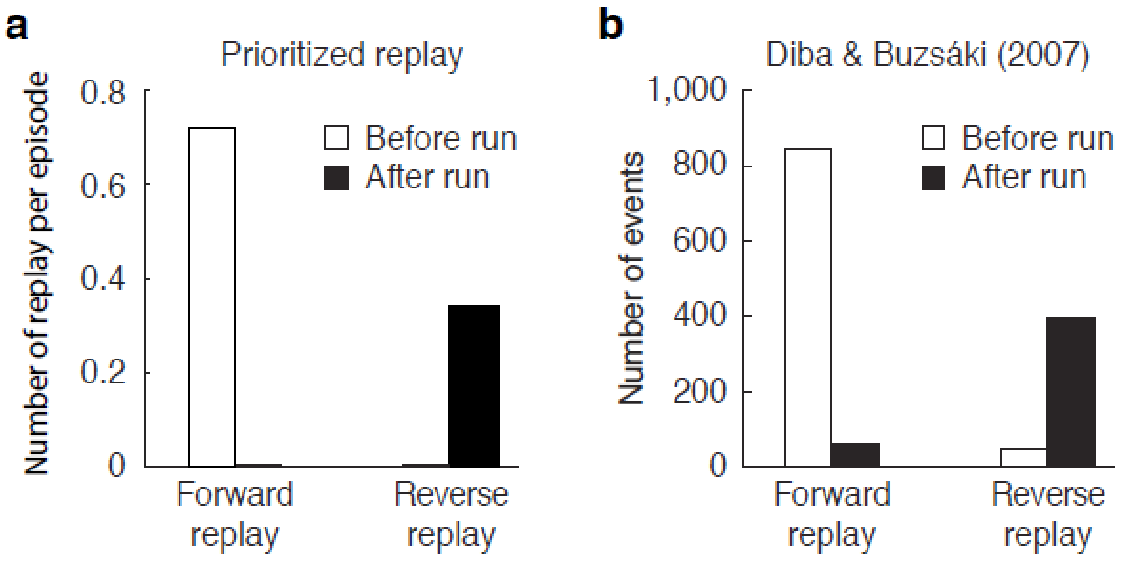
After first state is updated, choose two steps back, three steps back… = reverse replay

When need > gain, greatest utility in states ahead producing forward sweeps

Results 2: Forward and reverse replay



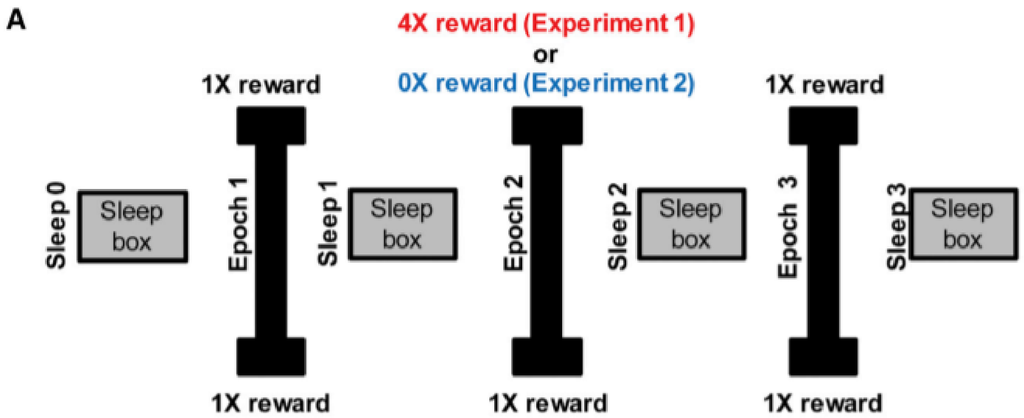
Compare to Diba and Buzsaki (2007):



Results 3: Asymmetric modulation of replay events

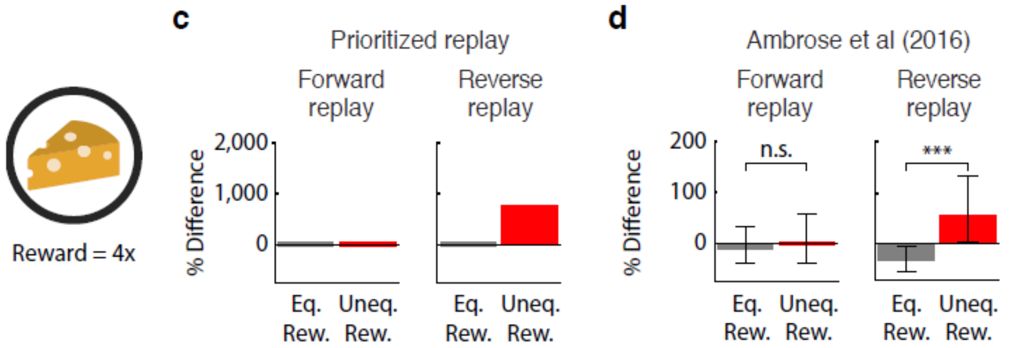
Ambrose et al. (2016): Repeated trials with either 4x or 0x reward on linear track

Forward events not affected by changing reward

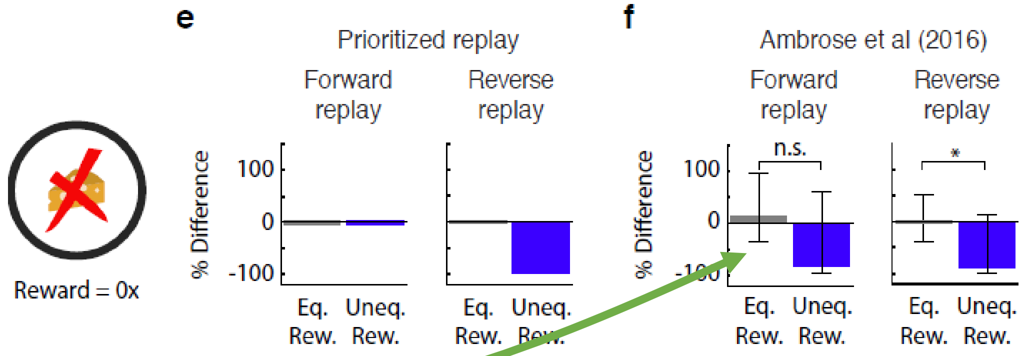


Note small increase in reverse replay events in response to conventional 1x rewards in 0x reward trials

Rate of reverse replay increases in response to unexpectedly large rewards (positive prediction errors; Ambrose et al., 2016)



Rate of reverse replay decreases for unexpectedly small reward (negative prediction error; Ambrose et al., 2016)

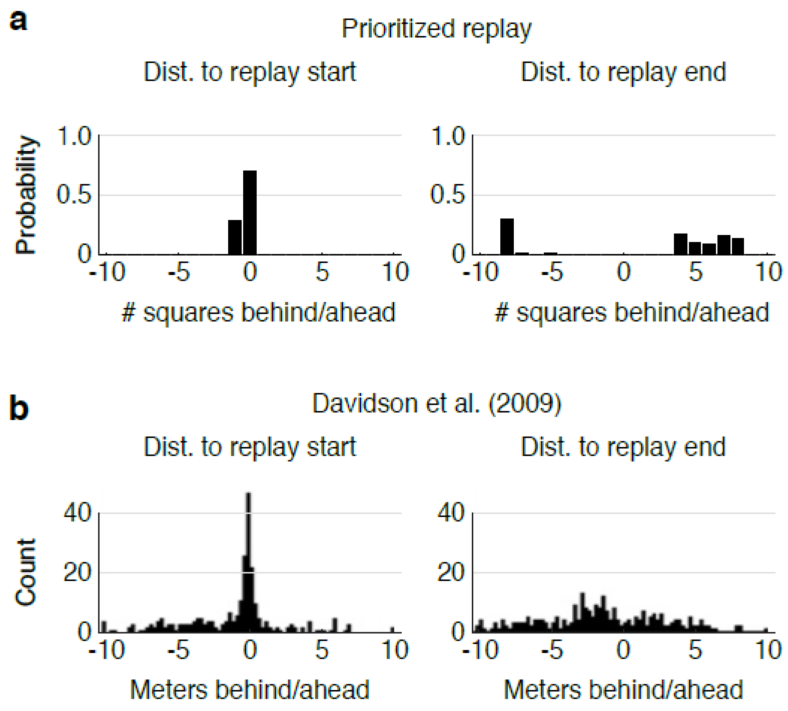


Results 4: Statistics of replay locations

Distribution of start and end locations

Beginning of replay trajectories biased towards animal and goal locations...

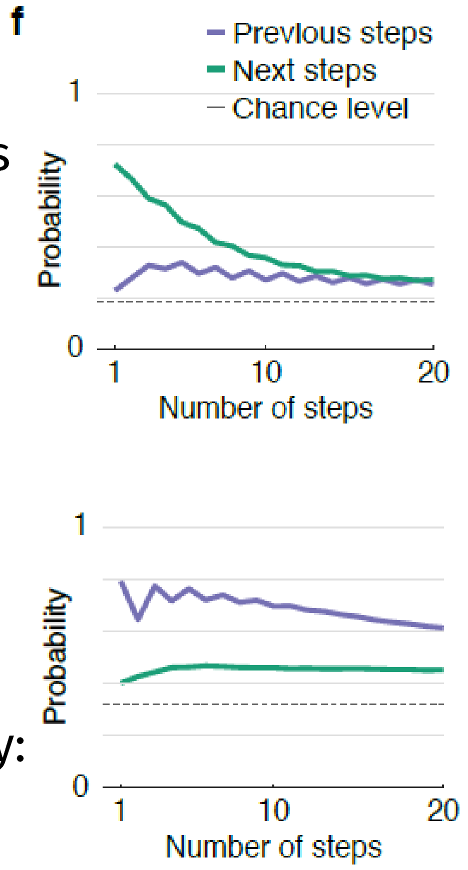
...no bias for the end of the trajectories (at least for animal location)



Effect on simulated behaviour

Forward events predict future occupancy:

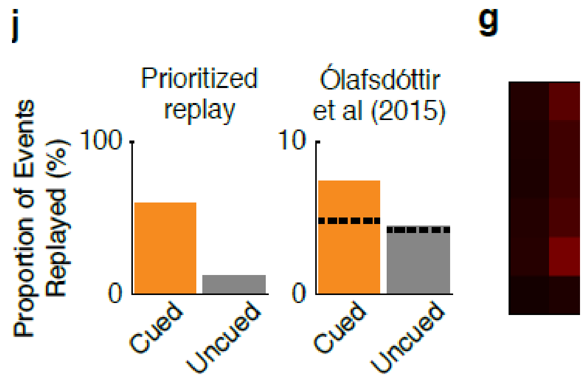
Backward events predict past occupancy:



Effect on offline reactivations

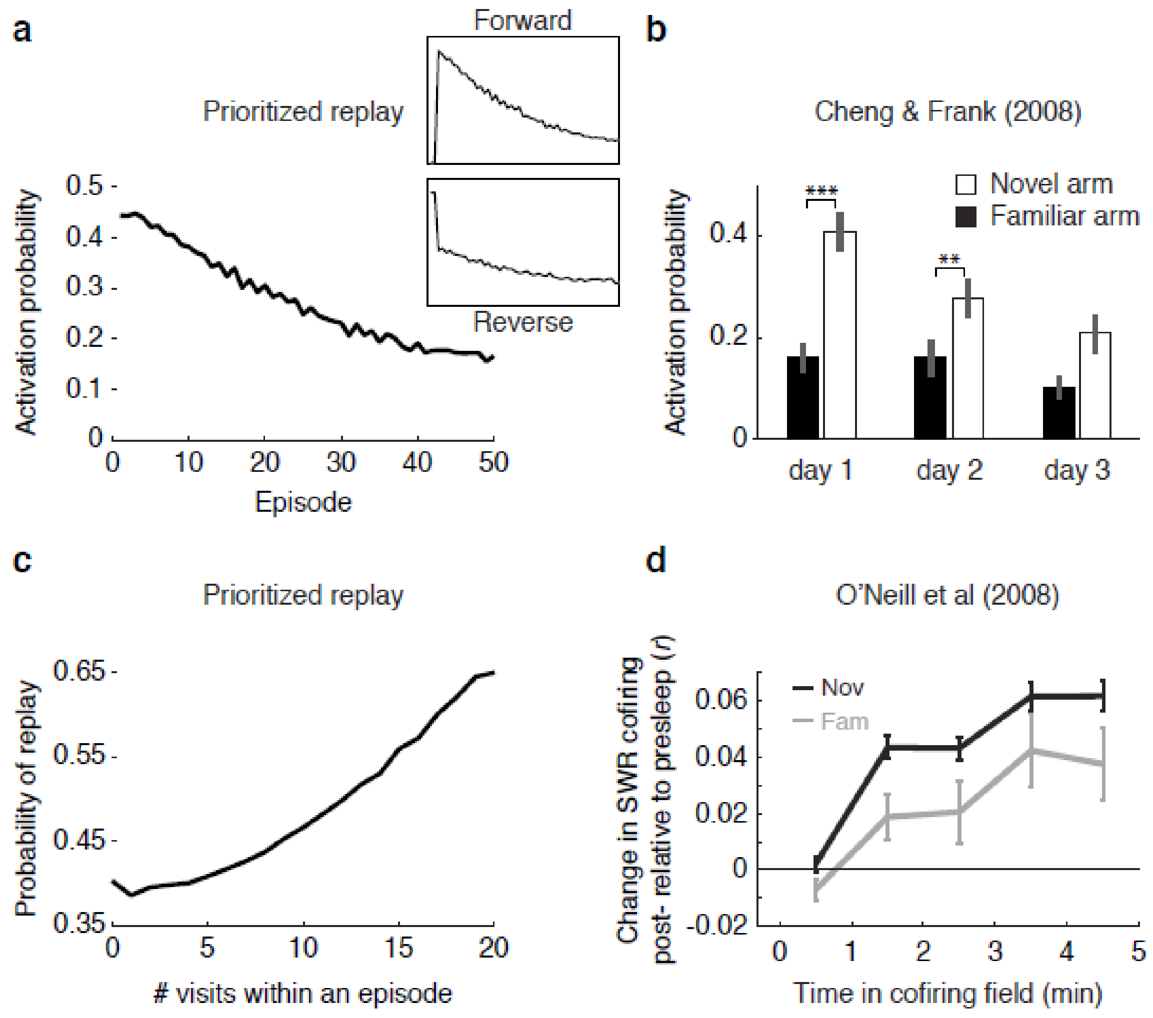
Simulated T-maze with reward on one arm (Olafsdottir et al., 2015)

Updated states over-represent reward location



Results 5: Replay frequency decreases with learning

- Gain term decreases with experience since its related to predictions errors, which also decrease due to improving policy
- Need term increases since trajectories become more stereotyped
- Decreasing gain leads to fewer overall replay events (i.e. there are still replayed states, but they are no longer contiguous in space)
- But, increasing need term increase probability that a given state is included in a significant replay event



End

Thanks!