Lecture 12: Predictive maps

Samuel Gershman

Harvard University

Roadmap

- This lecture covers the flexibility-efficiency trade-off entailed by different predictive representations and the algorithms that operate on them.

Roadmap

- This lecture covers the flexibility-efficiency trade-off entailed by different predictive representations and the algorithms that operate on them.
- **Successor representation**: generalization of the value function concept that predicts a multiplicity of different targets (future states or features). This enables more flexibility than the value function alone, with only a modest efficiency loss.

Roadmap

- This lecture covers the flexibility-efficiency trade-off entailed by different predictive representations and the algorithms that operate on them.
- **Successor representation**: generalization of the value function concept that predicts a multiplicity of different targets (future states or features). This enables more flexibility than the value function alone, with only a modest efficiency loss.
- This "predictive map" may be represented in the hippocampus and updated by dopaminergic temporal difference errors or a temporally asymmetric learning rule.

Roadmap

- This lecture covers the flexibility-efficiency trade-off entailed by different predictive representations and the algorithms that operate on them.
- **Successor representation**: generalization of the value function concept that predicts a multiplicity of different targets (future states or features). This enables more flexibility than the value function alone, with only a modest efficiency loss.
- This "predictive map" may be represented in the hippocampus and updated by dopaminergic temporal difference errors or a temporally asymmetric learning rule.
- A low-dimensional representation in entorhinal cortex may regularize learning toward predictions that are smooth across the state space.

Flexibility-efficiency trade-off

- The models described in the last two lectures focus on learning representations (value functions and policies) that facilitate optimal prediction and behavior with relatively little computation.

Flexibility-efficiency trade-off

- The models described in the last two lectures focus on learning representations (value functions and policies) that facilitate optimal prediction and behavior with relatively little computation.
- Simplest case: value functions and policies are stored in look-up tables, requiring no computation beyond finding the right entry for a given state-action pair.

Flexibility-efficiency trade-off

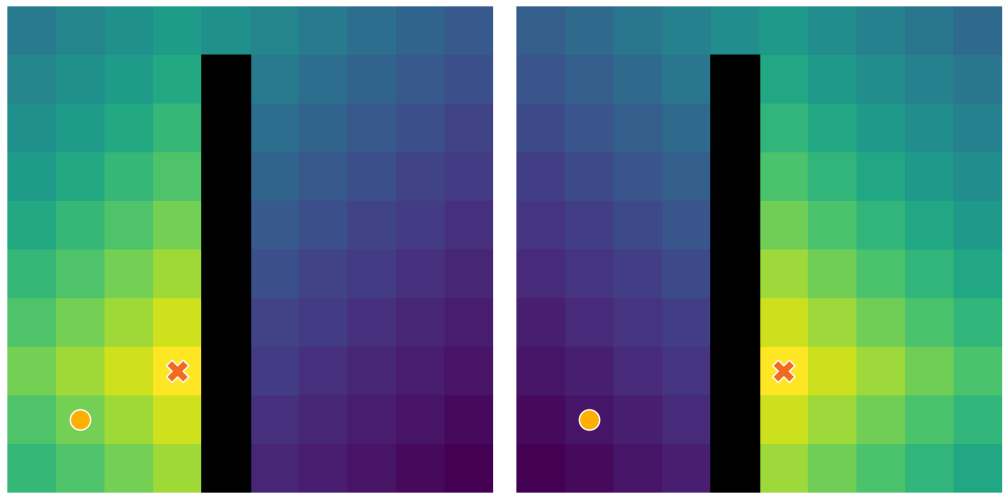
- The models described in the last two lectures focus on learning representations (value functions and policies) that facilitate optimal prediction and behavior with relatively little computation.
- Simplest case: value functions and policies are stored in look-up tables, requiring no computation beyond finding the right entry for a given state-action pair.
- More complex case: functions are neural networks, still relatively cheap.

Flexibility-efficiency trade-off

- The models described in the last two lectures focus on learning representations (value functions and policies) that facilitate optimal prediction and behavior with relatively little computation.
- Simplest case: value functions and policies are stored in look-up tables, requiring no computation beyond finding the right entry for a given state-action pair.
- More complex case: functions are neural networks, still relatively cheap.
- The trade-off is that these representations are relatively inflexible: if the world changes, the representations may need to be relearned.

Local changes have non-local effects on the value function

The goal is shown as an X and the agent is shown as a circle. Black squares are walls. The shading of the other squares shows the value function. The two gridworlds differ only by the goal placement.



Local changes have non-local effects on the value function

- A local change has non-local changes on the value function because the value of each state depends on the values of other states.

Local changes have non-local effects on the value function

- A local change has non-local changes on the value function because the value of each state depends on the values of other states.
- States may need to be resampled in order to update the value function.

Local changes have non-local effects on the value function

- A local change has non-local changes on the value function because the value of each state depends on the values of other states.
- States may need to be resampled in order to update the value function.
- When experience is scarce, relearning may be prohibitively costly.

Local changes have non-local effects on the value function

- A local change has non-local changes on the value function because the value of each state depends on the values of other states.
- States may need to be resampled in order to update the value function.
- When experience is scarce, relearning may be prohibitively costly.
- But animals don't always need to relearn. They're sometimes capable of rapid adaptation. How?

Latent learning

- Animals can sometimes learn environment structure in the absence of reinforcement.

Latent learning

- Animals can sometimes learn environment structure in the absence of reinforcement.
- Rats allowed to freely explore a maze learn more quickly when subsequently reinforced [Tolman 1948].

Latent learning

- Animals can sometimes learn environment structure in the absence of reinforcement.
- Rats allowed to freely explore a maze learn more quickly when subsequently reinforced [Tolman 1948].
- Contextual fear conditioning: shock paired with a context. If the shock is delivered immediately, conditioning is *weaker* compared to the same procedure with a delayed shock [Fanselow 1990].

Latent learning

- Animals can sometimes learn environment structure in the absence of reinforcement.
- Rats allowed to freely explore a maze learn more quickly when subsequently reinforced [Tolman 1948].
- Contextual fear conditioning: shock paired with a context. If the shock is delivered immediately, conditioning is *weaker* compared to the same procedure with a delayed shock [Fanselow 1990].
- Additional time is required for the animal to explore the box and form a predictive map, which then allows the shock delivery to propagate across the whole state space. Our goal is to formalize this idea.

The successor representation

- The value function admits the following decomposition:

$$V^{\pi}(s) = \sum_{\tilde{s}} M^{\pi}(s, \tilde{s}) R(\tilde{s})$$

where

$$M^{\pi}(s, \tilde{s}) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^{t} \mathbb{I}[s_t = \tilde{s}] | s_0 = s, \pi\right]$$

is the SR (expected discounted number of times an agent visits state s' on a trajectory starting in state s).

The successor representation

- The value function admits the following decomposition:

$$V^{\pi}(s) = \sum_{\tilde{s}} M^{\pi}(s, \tilde{s}) R(\tilde{s})$$

  where

$$M^{\pi}(s, \tilde{s}) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \mathbb{I}[s_t = \tilde{s}] | s_0 = s, \pi\right]$$

  is the SR (expected discounted number of times an agent visits state s' on a trajectory starting in state s).
- The SR formalizes our intuitions about a predictive map: it specifies where the agent will be in the near future (with a predictive horizon set by the discount factor γ).

The successor representation

- The value of state s is a linear combination of immediate expected rewards in other states (s') weighted by the frequency of visiting each state in the near future starting from s.

The successor representation

- The value of state s is a linear combination of immediate expected rewards in other states (s') weighted by the frequency of visiting each state in the near future starting from s.
- Given the SR, values can be computed by a linear operation on the reward function.

The successor representation

- The value of state s is a linear combination of immediate expected rewards in other states (s') weighted by the frequency of visiting each state in the near future starting from s.
- Given the SR, values can be computed by a linear operation on the reward function.
- Key feature: separation of state prediction and reward prediction. This allows the model to flexibly reuse state predictions when the reward changes.

Bellman equation for the successor representation

- Analogous to the value function:

$$M^{\pi}(s, \tilde{s}) = \mathbb{I}[s = \tilde{s}] + \gamma \sum_{s'} T^{\pi}(s, s') M^{\pi}(s', \tilde{s})$$

Bellman equation for the successor representation

- Analogous to the value function:

$$M^{\pi}(s, \tilde{s}) = \mathbb{I}[s = \tilde{s}] + \gamma \sum_{s'} T^{\pi}(s, s') M^{\pi}(s', \tilde{s})$$

- TD learning can be used to update an estimate of the SR, $\hat{M}^{\pi}$, from observed state transitions ($s \to s'$):

$$\Delta \hat{M}^{\pi}(s, \tilde{s}) \propto \mathbb{I}[s = s'] + \gamma \hat{M}^{\pi}(s', \tilde{s}) - \hat{M}^{\pi}(s, \tilde{s})$$

  The right-hand side is the TD error applied to long-range state prediction.

Bellman equation for the successor representation

- Analogous to the value function:

$$M^{\pi}(s, \tilde{s}) = \mathbb{I}[s = \tilde{s}] + \gamma \sum_{s'} T^{\pi}(s, s') M^{\pi}(s', \tilde{s})$$

- TD learning can be used to update an estimate of the SR, $\hat{M}^{\pi}$, from observed state transitions ($s \to s'$):

$$\Delta \hat{M}^{\pi}(s, \tilde{s}) \propto \mathbb{I}[s = s'] + \gamma \hat{M}^{\pi}(s', \tilde{s}) - \hat{M}^{\pi}(s, \tilde{s})$$

  The right-hand side is the TD error applied to long-range state prediction.

- The immediate rewards can also be learned by a simple error-driven update after observing reward r in state s:

$$\Delta \hat{R}(s) \propto r - \hat{R}(s)$$

Illustration with contextual fear conditioning

- Context pre-exposure in contextual fear conditioning provides the SR with data to form a predictive map: delivery of shock (change to the reward function coincident with the agent's current location) is propagated to other states in the environment.

Illustration with contextual fear conditioning

- Context pre-exposure in contextual fear conditioning provides the SR with data to form a predictive map: delivery of shock (change to the reward function coincident with the agent's current location) is propagated to other states in the environment.
- Eliminating pre-exposure attenuates conditioned fear to the context: upon return to the context, the animal is likely placed in a location different from the one that coincided with the shock, so it will only produce a fear response if the shock expectation has been propagated across the state space.

Flexibility-efficiency trade-off revisited

- SR lies in between pure model-free algorithms (like value learning with TD) and pure model-based algorithms (planning, discussed in the next lecture).

Flexibility-efficiency trade-off revisited

- SR lies in between pure model-free algorithms (like value learning with TD) and pure model-based algorithms (planning, discussed in the next lecture).
- Compared to model-free algorithms, SR can more flexibly adapt to changes in the reward function.

Flexibility-efficiency trade-off revisited

- SR lies in between pure model-free algorithms (like value learning with TD) and pure model-based algorithms (planning, discussed in the next lecture).
- Compared to model-free algorithms, SR can more flexibly adapt to changes in the reward function.
- Model-based algorithms are maximally flexible, since they can in principle adapt to any change in the reward and transition functions, but at the cost of higher computational complexity.

Transition and reward revaluation

- SR is differentially sensitive to changes in the transition function compared to the reward function.

Transition and reward revaluation

- SR is differentially sensitive to changes in the transition function compared to the reward function.
- The SR achieves its efficiency by compiling transition information in long-range state predictions. A change in the transition function induces non-local changes in the SR, analogous to the non-local changes induced in the value function.

Transition and reward revaluation

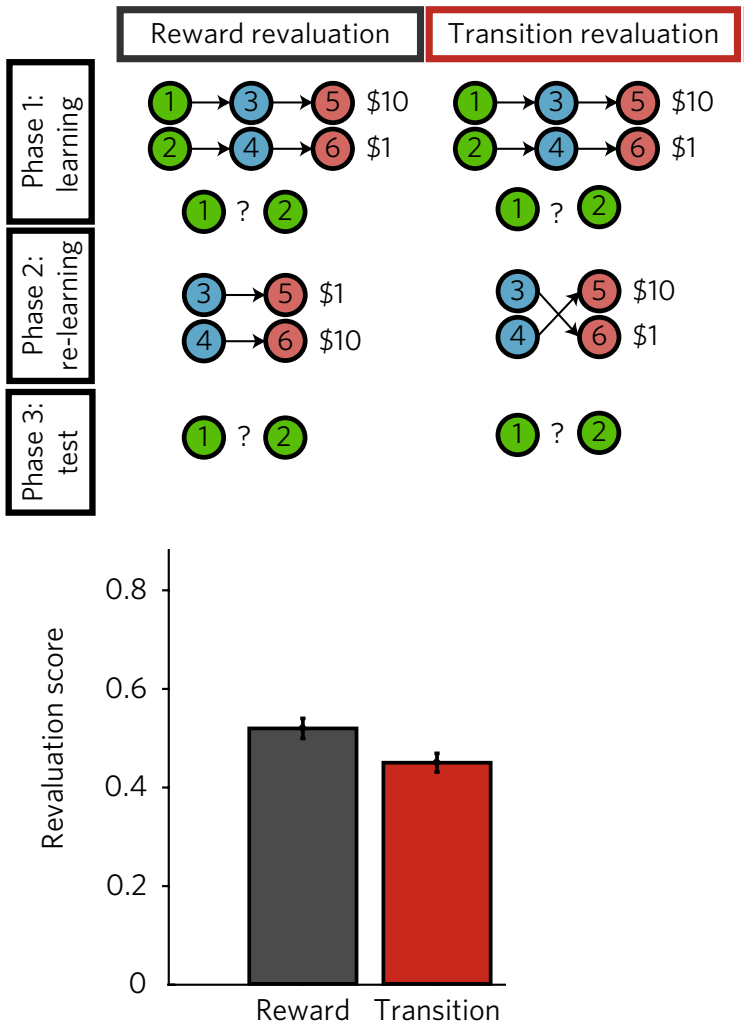
- SR is differentially sensitive to changes in the transition function compared to the reward function.
- The SR achieves its efficiency by compiling transition information in long-range state predictions. A change in the transition function induces non-local changes in the SR, analogous to the non-local changes induced in the value function.
- Changes to the reward function induce local changes to the reward function estimate.

Transition and reward revaluation

- SR is differentially sensitive to changes in the transition function compared to the reward function.
- The SR achieves its efficiency by compiling transition information in long-range state predictions. A change in the transition function induces non-local changes in the SR, analogous to the non-local changes induced in the value function.
- Changes to the reward function induce local changes to the reward function estimate.
- Model-based algorithms are equally sensitive to both transition and reward function changes, whereas model-free algorithms are equally insensitive.

Transition and reward revaluation

(Top) Experimental design. Circles denote states and arrows denote transitions. (Bottom) Behavioral results. Revaluation score denotes the change in preference rating after versus before the relearning phase.



[Momennejad et al 2017]

A predictive map in the hippocampus

- Long tradition of thinking about the hippocampus as representing a "cognitive map" [O'Keefe & Nadel 1978], motivated by place cells in the hippocampus tuned to an animal's location. But it later became evident that place cells were not simply registering an animal's location.

A predictive map in the hippocampus

- Long tradition of thinking about the hippocampus as representing a "cognitive map" [O'Keefe & Nadel 1978], motivated by place cells in the hippocampus tuned to an animal's location. But it later became evident that place cells were not simply registering an animal's location.
- When rats repeatedly traverse a linear track, place cells expand their tuning in the direction opposite of travel [Mehta et al 2000]. A cell that initially responds when the rat is in a particular location begin responding to earlier locations.

A predictive map in the hippocampus

- Long tradition of thinking about the hippocampus as representing a "cognitive map" [O'Keefe & Nadel 1978], motivated by place cells in the hippocampus tuned to an animal's location. But it later became evident that place cells were not simply registering an animal's location.
- When rats repeatedly traverse a linear track, place cells expand their tuning in the direction opposite of travel [Mehta et al 2000]. A cell that initially responds when the rat is in a particular location begin responding to earlier locations.
- Place fields concentrate near rewards [Hollup et al 2001].

A predictive map in the hippocampus

- Long tradition of thinking about the hippocampus as representing a "cognitive map" [O'Keefe & Nadel 1978], motivated by place cells in the hippocampus tuned to an animal's location. But it later became evident that place cells were not simply registering an animal's location.
- When rats repeatedly traverse a linear track, place cells expand their tuning in the direction opposite of travel [Mehta et al 2000]. A cell that initially responds when the rat is in a particular location begin responding to earlier locations.
- Place fields concentrate near rewards [Hollup et al 2001].
- Lesions of the hippocampus impair several forms of latent learning, and stimulus-stimulus predictions are encoded in hippocampal activity during latent learning.

A predictive map in the hippocampus

- SR offers a different account of these phenomena. Assume that each state corresponds to a location, and the place field for a single cell corresponds to one column of the SR estimate, $\hat{M}^{\pi}(\cdot, \tilde{s})$; we will refer to this as the *SR field* tuned to location $\tilde{s}$.

A predictive map in the hippocampus

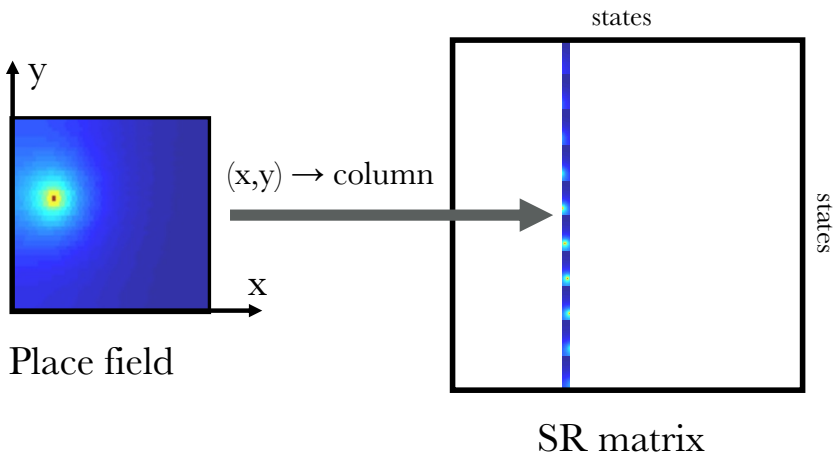
- SR offers a different account of these phenomena. Assume that each state corresponds to a location, and the place field for a single cell corresponds to one column of the SR estimate, $\hat{M}^{\pi}(\cdot, \tilde{s})$; we will refer to this as the *SR field* tuned to location $\tilde{s}$.
- An SR field is a *retrospective representation* in the sense that activity is maximal for states leading to $\tilde{s}$.

A predictive map in the hippocampus

- SR offers a different account of these phenomena. Assume that each state corresponds to a location, and the place field for a single cell corresponds to one column of the SR estimate, $\hat{M}^{\pi}(\cdot, \tilde{s})$; we will refer to this as the *SR field* tuned to location $\tilde{s}$.
- An SR field is a *retrospective representation* in the sense that activity is maximal for states leading to $\tilde{s}$.
- A row of the SR, $\hat{M}^{\pi}(s, \cdot)$, corresponds to the population code for state s. This is a *predictive representation* in the sense that activity is maximal for cells tuned to states leading from s.

From place fields to SR fields

Place field corresponding to a single column of the SR



[Gershman 2018]

From place fields to SR fields

- Place cells are traditionally identified using a random foraging paradigm in which rats are placed in a box and food pellets are randomly distributed, which induces the rat to explore the whole box.

From place fields to SR fields

- Place cells are traditionally identified using a random foraging paradigm in which rats are placed in a box and food pellets are randomly distributed, which induces the rat to explore the whole box.
- In this setting, with no systematic asymmetries in the direction of travel, SR fields will resemble classical place fields—approximately symmetric around the preferred location.

From place fields to SR fields

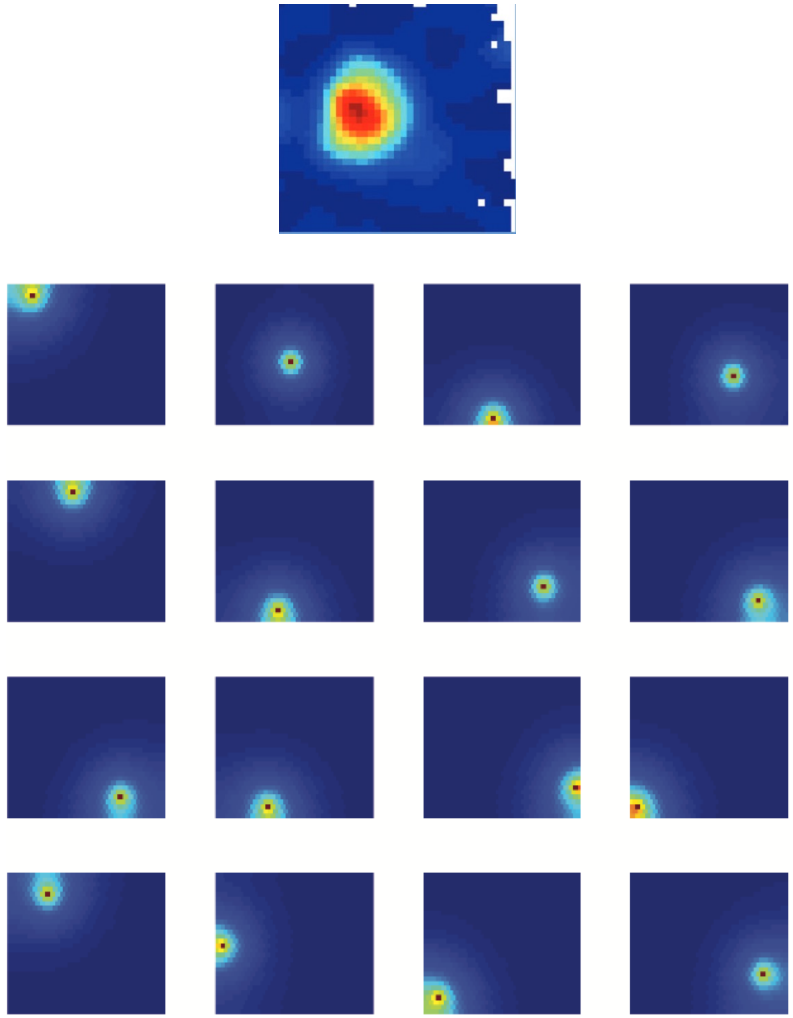
- Place cells are traditionally identified using a random foraging paradigm in which rats are placed in a box and food pellets are randomly distributed, which induces the rat to explore the whole box.
- In this setting, with no systematic asymmetries in the direction of travel, SR fields will resemble classical place fields—approximately symmetric around the preferred location.
- If rats are instead trained on the linear track or circular water maze, systematic asymmetries in the direction of travel will bias the SR fields backwards in the direction of travel (reflecting the retrospective nature of the representation) and concentrate them near the platform (reflecting the shift in tuning toward frequently visited locations).

From place fields to SR fields

- Place cells are traditionally identified using a random foraging paradigm in which rats are placed in a box and food pellets are randomly distributed, which induces the rat to explore the whole box.
- In this setting, with no systematic asymmetries in the direction of travel, SR fields will resemble classical place fields—approximately symmetric around the preferred location.
- If rats are instead trained on the linear track or circular water maze, systematic asymmetries in the direction of travel will bias the SR fields backwards in the direction of travel (reflecting the retrospective nature of the representation) and concentrate them near the platform (reflecting the shift in tuning toward frequently visited locations).
- Thus, the SR can explain both classical and non-classical aspects of place fields.

From place fields to SR fields

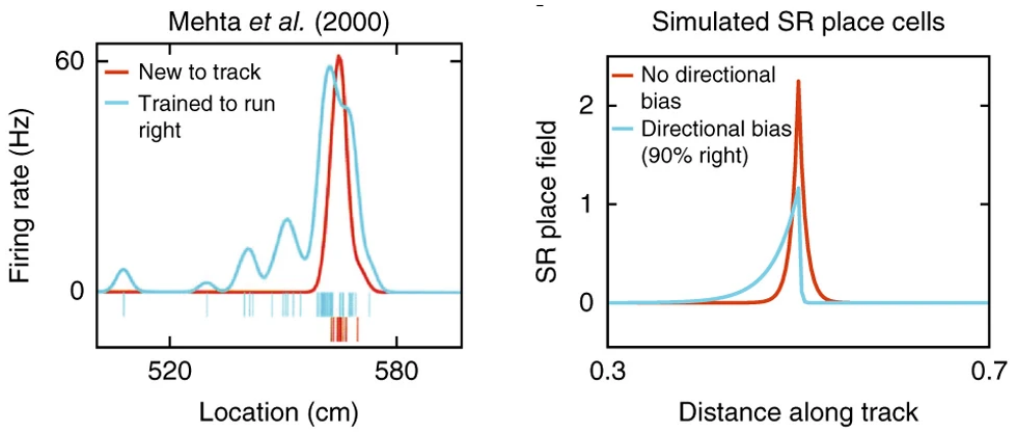
Real and simulated place fields in a random foraging paradigm



[Moser et al 2015; Stachenfeld et al 2017]

From place fields to SR fields

(Left) Real neurons. (Right) Successor representation.



[Mehta et al 2000; Stachenfeld et al 2017]

Learning the SR with dopamine

- Since we've already shown that the SR can be estimated using TD learning, a natural hypothesis is that dopamine provides the TD errors needed for this learning mechanism.

Learning the SR with dopamine

- Since we've already shown that the SR can be estimated using TD learning, a natural hypothesis is that dopamine provides the TD errors needed for this learning mechanism.
- This **generalized prediction error** hypothesis (generalizing from scalar rewards to feature vectors) can resolve a number of puzzles concerning dopamine.

Learning the SR with dopamine

- Picture of phasic dopamine developed in the last two lectures: reward prediction error.

Learning the SR with dopamine

- Picture of phasic dopamine developed in the last two lectures: reward prediction error.
- However, some dopamine neurons respond to prediction errors not only in the *amount* of reward but also in the *identity* of reward, such as different flavors [Takahashi et al 2017].

Learning the SR with dopamine

- Picture of phasic dopamine developed in the last two lectures: reward prediction error.
- However, some dopamine neurons respond to prediction errors not only in the *amount* of reward but also in the *identity* of reward, such as different flavors [Takahashi et al 2017].
- Dopamine neuron populations encode information about reward identity that diminishes over learning, consistent with a generalized prediction error signal [Stalnaker et al 2019].

Learning the SR with dopamine

- Picture of phasic dopamine developed in the last two lectures: reward prediction error.
- However, some dopamine neurons respond to prediction errors not only in the *amount* of reward but also in the *identity* of reward, such as different flavors [Takahashi et al 2017].
- Dopamine neuron populations encode information about reward identity that diminishes over learning, consistent with a generalized prediction error signal [Stalnaker et al 2019].
- If states correspond to reward identities, then generalized prediction errors can capture the sensitivity of dopamine neurons to sensory surprise.

Learning the SR with dopamine

- Picture of phasic dopamine developed in the last two lectures: reward prediction error.
- However, some dopamine neurons respond to prediction errors not only in the *amount* of reward but also in the *identity* of reward, such as different flavors [Takahashi et al 2017].
- Dopamine neuron populations encode information about reward identity that diminishes over learning, consistent with a generalized prediction error signal [Stalnaker et al 2019].
- If states correspond to reward identities, then generalized prediction errors can capture the sensitivity of dopamine neurons to sensory surprise.
- If dopamine neurons are tuned to different states, then reward identity will be encoded at the population level as long as the generalized prediction errors are non-zero.

Learning the SR with dopamine

- If the hippocampus encodes the SR and dopamine is used to update the SR, are dopamine inputs to the hippocampus critical?

Learning the SR with dopamine

- If the hippocampus encodes the SR and dopamine is used to update the SR, are dopamine inputs to the hippocampus critical?
- Evidence is suggestive but indirect. Over-representation of high-reward locations in the hippocampus can be attenuated by inhibition of dopamine inputs to the hippocampus [Mamad et al 2017; Krishnan et al 2022].

Learning the SR with dopamine

- If the hippocampus encodes the SR and dopamine is used to update the SR, are dopamine inputs to the hippocampus critical?
- Evidence is suggestive but indirect. Over-representation of high-reward locations in the hippocampus can be attenuated by inhibition of dopamine inputs to the hippocampus [Mamad et al 2017; Krishnan et al 2022].
- Hippocampal plasticity depends on dopamine signaling [Tsetsenis et al 2023].

Recurrent neural network implementation

- Another possibility: mechanisms endogenous to the hippocampus could compute and learn the SR [Fang et al 2023].

Recurrent neural network implementation

- Another possibility: mechanisms endogenous to the hippocampus could compute and learn the SR [Fang et al 2023].
- Let x_t denote an input to a linear recurrent neural network (RNN) with recurrent synaptic connectivity matrix Ω, where Ω_{ij} denotes the strength of the synapse connecting neuron i to neuron j.

Recurrent neural network implementation

- Another possibility: mechanisms endogenous to the hippocampus could compute and learn the SR [Fang et al 2023].
- Let x_t denote an input to a linear recurrent neural network (RNN) with recurrent synaptic connectivity matrix Ω, where Ω_{ij} denotes the strength of the synapse connecting neuron i to neuron j.
- Firing rate dynamics of the recurrent units (z) can be written in discrete time as:

$$z_{t+1} = x_t + \gamma z_t \Omega$$

where the discount factor γ plays the role of a synaptic gain.

Recurrent neural network implementation

- If $x_t = x$ is a constant one-hot encoding of the state (i.e., $x(s) = 1$ whenever the system is in state s, otherwise $x(s) = 0$), and the connectivity matrix corresponds to the state transition matrix, $\Omega = T^\pi$, then at steady state the recurrent units will converge to the row of the SR corresponding to x:

$$\lim_{t \to \infty} z_t = xM^\pi = M^\pi(s, \cdot)$$

Recurrent neural network implementation

- If $x_t = x$ is a constant one-hot encoding of the state (i.e., $x(s) = 1$ whenever the system is in state s, otherwise $x(s) = 0$), and the connectivity matrix corresponds to the state transition matrix, $\Omega = T^\pi$, then at steady state the recurrent units will converge to the row of the SR corresponding to x:

$$\lim_{t \to \infty} z_t = xM^\pi = M^\pi(s, \cdot)$$

- Intuitively, the RNN is progressing the predictions forward at each iteration and adding up the results.

Recurrent neural network implementation

- How can Ω be learned such that it corresponds to T^{π}?

Recurrent neural network implementation

- How can Ω be learned such that it corresponds to T^{π}?
- This requires that a learning rule associate consecutive states together and normalize the synaptic strengths so that they define a valid probability distribution. A learning rule satisfying both requirements [Fang et al 2023]:

$$\Delta\Omega \propto z_{t-1}^{\top} z_t - z_{t-1}^{\top} z_{t-1} \Omega$$

Recurrent neural network implementation

- How can Ω be learned such that it corresponds to T^{π}?
- This requires that a learning rule associate consecutive states together and normalize the synaptic strengths so that they define a valid probability distribution. A learning rule satisfying both requirements [Fang et al 2023]:

$$\Delta\Omega \propto z_{t-1}^{\top} z_t - z_{t-1}^{\top} z_{t-1} \Omega$$

- First term implements temporally asymmetric potentiation, similar to spike-timing dependent plasticity.

Recurrent neural network implementation

- How can Ω be learned such that it corresponds to T^{π}?
- This requires that a learning rule associate consecutive states together and normalize the synaptic strengths so that they define a valid probability distribution. A learning rule satisfying both requirements [Fang et al 2023]:

$$\Delta\Omega \propto z_{t-1}^{\top} z_t - z_{t-1}^{\top} z_{t-1} \Omega$$

- First term implements temporally asymmetric potentiation, similar to spike-timing dependent plasticity.
- Second term is anti-Hebbian depotentiation that ensures normalization.

Recurrent neural network implementation

- How can Ω be learned such that it corresponds to T^{π}?
- This requires that a learning rule associate consecutive states together and normalize the synaptic strengths so that they define a valid probability distribution. A learning rule satisfying both requirements [Fang et al 2023]:

$$\Delta\Omega \propto z_{t-1}^{\top} z_t - z_{t-1}^{\top} z_{t-1} \Omega$$

- First term implements temporally asymmetric potentiation, similar to spike-timing dependent plasticity.
- Second term is anti-Hebbian depotentiation that ensures normalization.
- Update rule uses only information local to a synapse, and converges to the transition matrix asymptotically.

Successor features

- The SR is tabular—it assumes that the state space can be discretely enumerated. Learning a tabular representation requires experiencing each state many times. This is clearly prohibitive in large or continuous state spaces.

Successor features

- The SR is tabular—it assumes that the state space can be discretely enumerated. Learning a tabular representation requires experiencing each state many times. This is clearly prohibitive in large or continuous state spaces.
- If expected rewards can be represented as a linear function of the features, $R(s) = \sum_d w_d x_d$, then the value function can be represented as a linear function of **successor features** (SFs), $\psi^\pi(s)$:

$$V^\pi(s) = \sum_d w_d \psi^\pi_d(s), \qquad \psi^\pi(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t f(s_t) | s_0 = s, \pi\right]$$

where $x = f(s)$.

Successor features

- SFs are analogous to value functions, but accumulating general features rather than just reward. This allows an agent to generalize across states, and even across tasks.

Successor features

- SFs are analogous to value functions, but accumulating general features rather than just reward. This allows an agent to generalize across states, and even across tasks.
- SFs also obey a Bellman equation:

$$\psi^{\pi}(s) = f(s) + \gamma \sum_{s'} T^{\pi}(s, s')\psi^{\pi}(s')$$

and hence can be estimated using TD learning.

Successor features

- SFs are analogous to value functions, but accumulating general features rather than just reward. This allows an agent to generalize across states, and even across tasks.
- SFs also obey a Bellman equation:

$$\psi^{\pi}(s) = f(s) + \gamma \sum_{s'} T^{\pi}(s, s')\psi^{\pi}(s')$$

  and hence can be estimated using TD learning.
- If different subpopulations of dopamine neurons signal TD errors for different features, this model may explain how a single canonical computation (the TD update) is compatible with the diversity of feature sensitivity across dopamine neurons [Engelhard et al 2019].

Prioritized replay for offline updating

- The hippocampus is active "offline" (during quiet rest or sleep), sometimes replaying past events. Can we formalize the function of such replay?

Prioritized replay for offline updating

- The hippocampus is active "offline" (during quiet rest or sleep), sometimes replaying past events. Can we formalize the function of such replay?
- Suppose we have an estimate of the state-action value function, $\hat{Q}^{\pi}$, which we just updated after a state-action-reward sequence.

Prioritized replay for offline updating

- The hippocampus is active "offline" (during quiet rest or sleep), sometimes replaying past events. Can we formalize the function of such replay?
- Suppose we have an estimate of the state-action value function, $\hat{Q}^{\pi}$, which we just updated after a state-action-reward sequence.
- The simplest approach to propagating this value information across all states and actions is to sample random state-action pairs (s, a) and use the Bellman equation to determine the updated value (Bellman backup):

$$\hat{Q}^{\pi}(s, a) \leftarrow R(s) + \gamma \sum_{s'} T(s'|s, a) \max_{a'} \hat{Q}^{\pi}(s', a')$$

Prioritized replay for offline updating

- The hippocampus is active "offline" (during quiet rest or sleep), sometimes replaying past events. Can we formalize the function of such replay?
- Suppose we have an estimate of the state-action value function, $\hat{Q}^{\pi}$, which we just updated after a state-action-reward sequence.
- The simplest approach to propagating this value information across all states and actions is to sample random state-action pairs (s, a) and use the Bellman equation to determine the updated value (Bellman backup):

$$\hat{Q}^{\pi}(s, a) \leftarrow R(s) + \gamma \sum_{s'} T(s'|s, a) \max_{a'} \hat{Q}^{\pi}(s', a')$$

- Repeating this update for different state-action pairs is guaranteed to converge to the correct values under the optimal policy π^*.

Prioritized replay for offline updating

- Two drawbacks: (1) requires a model (T and R), which might not be available; (2) might be very inefficient or intractable if the state/action space is vast.

Prioritized replay for offline updating

- Two drawbacks: (1) requires a model (T and R), which might not be available; (2) might be very inefficient or intractable if the state/action space is vast.
- Instead of using a model, use a stochastic approximation based on samples $e = (s, a, s', r)$ drawn from memory:

$$\Delta \hat{Q}^{\pi}(s, a) \propto r + \gamma \max_{a'} \hat{Q}^{\pi}(s', a') - \hat{Q}^{\pi}(s, a)$$

Prioritized replay for offline updating

- Two drawbacks: (1) requires a model (T and R), which might not be available; (2) might be very inefficient or intractable if the state/action space is vast.
- Instead of using a model, use a stochastic approximation based on samples $e = (s, a, s', r)$ drawn from memory:

$$\Delta \hat{Q}^{\pi}(s, a) \propto r + \gamma \max_{a'} \hat{Q}^{\pi}(s', a') - \hat{Q}^{\pi}(s, a)$$

- Sampling from memory also has the advantage that it focuses updating on parts of the state-action space that the agent is likely to visit (assuming the past is representative of the future).

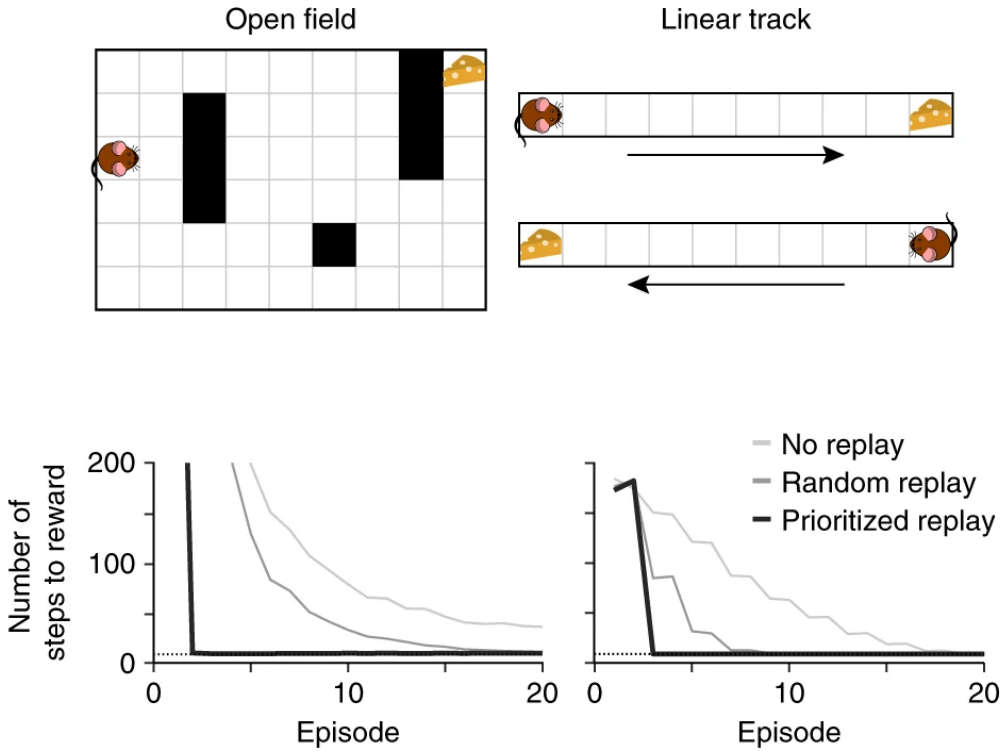
Prioritized replay for offline updating

- The way memory is sampled can have a big impact on the speed of learning.

Prioritized replay for offline updating

- The way memory is sampled can have a big impact on the speed of learning.
- Optimal prioritization strategy can do much better than uniformly sampling memory.

Optimal vs. uniform memory sampling



[Mattar & Daw 2018]

Prioritized replay for offline updating

- Optimal prioritization strategy is defined by the *expected value of backup* (EVB):

$$\text{EVB}(k) = \hat{V}^{\pi_k}(s) - \hat{V}^{\pi}(s)$$

where s is the agent's current state, k indexes memories $\{e_k\}$, and π_k is the new policy after applying a backup to the retrieved memory.

Prioritized replay for offline updating

- Optimal prioritization strategy is defined by the *expected value of backup* (EVB):

$$\text{EVB}(k) = \hat{V}^{\pi_k}(s) - \hat{V}^{\pi}(s)$$

  where s is the agent's current state, k indexes memories $\{e_k\}$, and π_k is the new policy after applying a backup to the retrieved memory.
- EVB quantifies the improvement in the policy that could be attained by applying a backup to a particular memory.

Prioritized replay for offline updating

- Optimal prioritization strategy is defined by the *expected value of backup* (EVB):

$$\text{EVB}(k) = \hat{V}^{\pi_k}(s) - \hat{V}^{\pi}(s)$$

  where s is the agent's current state, k indexes memories $\{e_k\}$, and π_k is the new policy after applying a backup to the retrieved memory.
- EVB quantifies the improvement in the policy that could be attained by applying a backup to a particular memory.
- Optimal prioritization strategy schedules these updates based on decreasing EVB.

Prioritized replay for offline updating

- EVB can be expressed in the following way:

$$\text{EVB}(s, k) = \text{Gain}(k) \cdot \text{Need}(s, k),$$

where $\text{Gain}(k) = V^{\pi_k}(s_k) - V^{\pi}(s_k)$ is the improvement local to the retrieved state, and $\text{Need}(s, k) = M^{\pi}(s, s_k)$ is how often the memory is needed (how often the retrieved state will be revisited in the future), quantified by the SR.

Prioritized replay for offline updating

- EVB can be expressed in the following way:

$$\text{EVB}(s, k) = \text{Gain}(k) \cdot \text{Need}(s, k),$$

  where $\text{Gain}(k) = V^{\pi_k}(s_k) - V^{\pi}(s_k)$ is the improvement local to the retrieved state, and $\text{Need}(s, k) = M^{\pi}(s, s_k)$ is how often the memory is needed (how often the retrieved state will be revisited in the future), quantified by the SR.
- Thus, in this setting the predictive map serves to guide memory retrieval.

Prioritized replay for offline updating

- Key idea: place cell reactivation reflects prioritized retrieval of past memories.

Prioritized replay for offline updating

- Key idea: place cell reactivation reflects prioritized retrieval of past memories.
- Model explains why place cells tend to be reactivated sequentially based on the sequence of experienced states, but critically makes different predictions about whether the sequence will be in forward or reverse order depending on the situation.

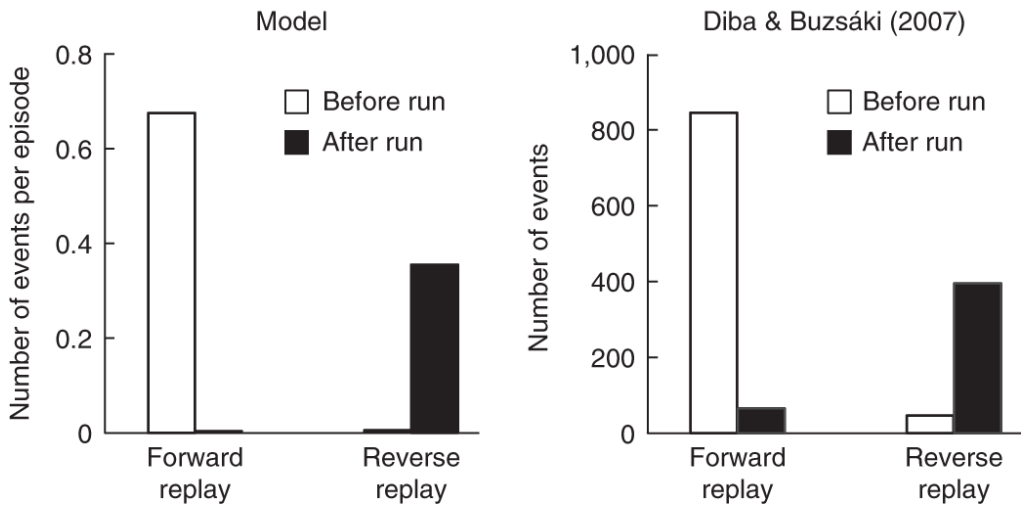
Prioritized replay for offline updating

- Key idea: place cell reactivation reflects prioritized retrieval of past memories.
- Model explains why place cells tend to be reactivated sequentially based on the sequence of experienced states, but critically makes different predictions about whether the sequence will be in forward or reverse order depending on the situation.
- If the agent encounters an unexpected reward, the gain term will be greatest for the current state's predecessors, leading to backward replay.

Prioritized replay for offline updating

- Key idea: place cell reactivation reflects prioritized retrieval of past memories.
- Model explains why place cells tend to be reactivated sequentially based on the sequence of experienced states, but critically makes different predictions about whether the sequence will be in forward or reverse order depending on the situation.
- If the agent encounters an unexpected reward, the gain term will be greatest for the current state's predecessors, leading to backward replay.
- In the absence of a prediction error, the need term dominates, leading to forward replay (states expected in the near future have higher need). This is why forward replay tends to be observed before an animal starts a run on the linear track.

Frequency of forward vs. reverse replays



[Mattar & Daw 2018]

Study question

How might you incorporate predictive maps into the kind of policy gradient algorithms described in the last lecture?

Summary

- Predictive maps enable the brain to (partially) escape the inflexibility that curses model-free algorithms, while achieving comparable efficiency.

Summary

- Predictive maps enable the brain to (partially) escape the inflexibility that curses model-free algorithms, while achieving comparable efficiency.
- A canonical RL computation (TD learning) can be generalized to construct predictive maps.

Summary

- Predictive maps enable the brain to (partially) escape the inflexibility that curses model-free algorithms, while achieving comparable efficiency.
- A canonical RL computation (TD learning) can be generalized to construct predictive maps.
- Hippocampus appears to represent a predictive map, possibly updated by vector-valued dopamine signals encoding generalized prediction errors.