Reinforcement Learning for Systems Neuroscientists

Emerson Harkin

University of Ottawa

August 9, 2019

Mathematical Background

Expected Values

The expected value of a discrete random variable X is

$$\mathbb{E}_X[X] = \sum_{x \in X} x p(X = x).$$

For a function of a random variable $g(\cdot)$, the expected value is

$$\mathbb{E}_X[g(X)] = \sum_{x \in X} g(x) p(X = x). \tag{1}$$

Example

A head-fixed mouse is presented with two lick ports. Define

$$X = \begin{cases} \text{no lick} & \text{with probability } 0.5 \\ \text{lick left} & \text{with probability } 0.4 \\ \text{lick right} & \text{with probability } 0.1 \end{cases} \qquad R(X) = \begin{cases} 0 & \text{if } X = \text{no lick} \\ 1 & \text{if } X = \text{lick left} \\ 2 & \text{if } X = \text{lick right}. \end{cases}$$

The expected reward is

$$\begin{aligned} \mathbb{E}[R(X)] &= R(\text{no lick})p(\text{no lick}) + R(\text{lick left})p(\text{lick left}) + R(\text{lick right})p(\text{lick right}) \\ &= 0 \times 0.5 + 1 \times 0.4 + 2 \times 0.1 \\ &= 0.6. \end{aligned}$$

Conditional Probability

Let a person's binned height H and weight W be potentially correlated random variables. The probability that a person's height is $h = 150\text{cm}$, given that their weight is $w = 60\text{kg}$ is

$$p(H = h \mid W = w) = p(h \mid w) = p(H = 150\text{cm} \mid W = 60\text{kg}).$$

Conditional Probability

Let a person's binned height H and weight W be potentially correlated random variables. The probability that a person's height is $h = 150\text{cm}$, given that their weight is $w = 60\text{kg}$ is

$$p(H = h \mid W = w) = p(h \mid w) = p(H = 150\text{cm} \mid W = 60\text{kg}).$$

The *overall* probability that a person's height is $h = 150\text{cm}$ ignoring their weight is

$$\begin{aligned} p(H = h) &= \mathbb{E}_W[p(H = h \mid W = w)] \\ &= \sum_{w \in W} p(H = h \mid W = w)p(W = w) \\ &= \sum_{w \in W} p(h \mid w)p(w) \\ &= p(H = 150\text{cm} \mid W = 55\text{kg})p(W = 55\text{kg}) + p(H = 150\text{cm} \mid W = 60\text{kg})p(W = 60\text{kg}) + \ldots \end{aligned}$$

Markov Chains

Consider a mouse presented with two ports that can be licked at any time and in any order.

Markov Chains

Consider a mouse presented with two ports that can be licked at any time and in any order. Model the behaviour of the mouse as a Markov chain with states $\mathcal{S} = \{\text{no lick}, \text{lick left}, \text{lick right}\}$. The state of the mouse at time t is a discrete random variable S_t over $\mathcal{S}$ from which a specific state s_t is sampled at each timestep with probability $p(S_t = s_t \mid S_{t-1} = s_{t-1})$.

Consider a mouse presented with two ports that can be licked at any time and in any order. Model the behaviour of the mouse as a Markov chain with states $\mathcal{S} = \{\text{no lick}, \text{lick left}, \text{lick right}\}$. The state of the mouse at time t is a discrete random variable S_t over $\mathcal{S}$ from which a specific state s_t is sampled at each timestep with probability $p(S_t = s_t \mid S_{t-1} = s_{t-1})$.

A *trajectory* $\tau = s_t, s_{t+1}, ..., s_{t+N}$ is an observed sequence of states which occur with joint probability $p(s_t, s_{t+1}, ..., s_{t+N})$. By the Markov property,

$$\begin{aligned} p(s_t, s_{t+1}, ..., s_{t+N}) &= p(s_t)p(s_{t+1} \mid s_t)...p(s_{t+N} \mid s_{t+N-1}) \\ &= p(s_t) \prod_{k=t+1}^{N} p(s_k \mid s_{k-1}). \end{aligned}$$

Markov Chains

Consider a mouse presented with two ports that can be licked at any time and in any order. Model the behaviour of the mouse as a Markov chain with states $\mathcal{S} = \{\text{no lick}, \text{lick left}, \text{lick right}\}$. The state of the mouse at time t is a discrete random variable S_t over $\mathcal{S}$ from which a specific state s_t is sampled at each timestep with probability $p(S_t = s_t \mid S_{t-1} = s_{t-1})$.

A *trajectory* $\tau = s_t, s_{t+1}, ..., s_{t+N}$ is an observed sequence of states which occur with joint probability $p(s_t, s_{t+1}, ..., s_{t+N})$. By the Markov property,

$$
\begin{aligned}
p(s_t, s_{t+1}, ..., s_{t+N}) &= p(s_t)p(s_{t+1} \mid s_t)...p(s_{t+N} \mid s_{t+N-1}) \\
&= p(s_t) \prod_{k=t+1}^{N} p(s_k \mid s_{k-1}).
\end{aligned}
$$

Take home point

The probability that the mouse's actions are [lick left, lick left, no lick, lick right] over four timesteps is trivially easy to compute *under the Markov assumption*. This is the main reason Markov processes are so widely used in reinforcement learning.

Pause to consider

Assume the mouse prefers the most recently licked port, even if it has not licked in several time steps. Can this be modelled with a Markov chain?

Pause to consider

Assume the mouse prefers the most recently licked port, even if it has not licked in several time steps. Can this be modelled with a Markov chain?

Answer: Yes, but the Markov chain must contain several more states.

$\mathcal{S} = \{(\text{no lick, last lick left}), (\text{no lick, last lick right}), \ldots\}$

Reinforcement Learning Basics

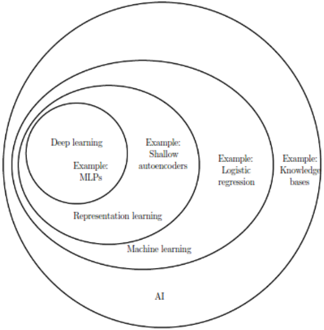

Figure 1.4: A Venn diagram showing how deep learning is a kind of representation learning, which is in turn a kind of machine learning, which is used for many but not all approaches to AI. Each section of the Venn diagram includes an example of an AI technology.

Figure: From Goodfellow et al. (2016?).

- Machine learning involves algorithms that can estimate model parameters from data.

Reinforcement learning in context

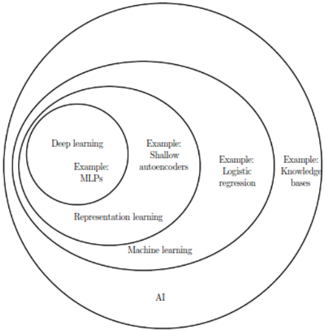

Figure 1.4: A Venn diagram showing how deep learning is a kind of representation learning, which is in turn a kind of machine learning, which is used for many but not all approaches to AI. Each section of the Venn diagram includes an example of an AI technology.

Figure: From Goodfellow et al. (2016?).

- Machine learning involves algorithms that can estimate model parameters from data.
  - Supervised learning.
    - Models that learn to predict the true value of a known variable.
    - Example: logistic regression.
  - Unsupervised learning.
    - Models that uncover hidden structure in data.
    - Examples: K-means clustering, PCA.
  - **Reinforcement learning (RL).**
    - Models that learn through interaction with their environment.
    - Examples: YouTube recommendation algorithm, control systems.

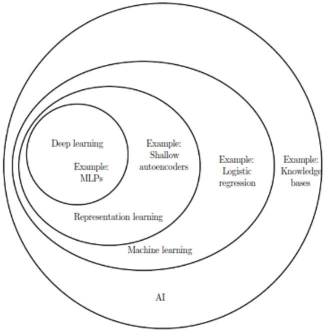

Figure 1.4: A Venn diagram showing how deep learning is a kind of representation learning, which is in turn a kind of machine learning, which is used for many but not all approaches to AI. Each section of the Venn diagram includes an example of an AI technology.

Figure: From Goodfellow et al. (2016?).

- Machine learning involves algorithms that can estimate model parameters from data.
  - Supervised learning.
    - Models that learn to predict the true value of a known variable.
    - Example: logistic regression.
  - Unsupervised learning.
    - Models that uncover hidden structure in data.
    - Examples: K-means clustering, PCA.
  - **Reinforcement learning (RL).**
    - Models that learn through interaction with their environment.
    - Examples: YouTube recommendation algorithm, control systems.
- Deep learning is now a common technique for nonlinear function approximation.
  - Can be used for any type of machine learning.
  - Example: value function approximation in RL.

Background

What does the world look like to a RL algorithm?

- **Agent:** Entity controlled by the algorithm.
  - Defined in terms of **actions** and their associated **values**.
- **Environment:** Anything not directly controlled by the algorithm.
  - Defined in terms of **states** and their associated transitions.

Notation

A_t: Random variable for action taken at time t.

a_t: Action actually taken at time t (i.e., sample drawn from A_t).

S_t: Random variable for state occupied at time t.

Background

What does the world look like to a RL algorithm?

- **Agent:** Entity controlled by the algorithm.
  - Defined in terms of **actions** and their associated **values**.
- **Environment:** Anything not directly controlled by the algorithm.
  - Defined in terms of **states** and their associated transitions.

Notation

A_t: Random variable for action taken at time t.

a_t: Action actually taken at time t (i.e., sample drawn from A_t).

S_t: Random variable for state occupied at time t.

Example

Situation	**Agent**	**Environment**
After pressing a lever, food reward is delivered		x
Hungry mouse chooses to eat food	x	
Mouse is no longer hungry after eating		x

Essential components of RL algorithms

Essential components of RL algorithms

- Reward signal.
  - **Quantity to be maximized.**
  - Usually a scalar $R_t \in \mathbb{R}$ at time t.

Essential components of RL algorithms

- Reward signal.
  - **Quantity to be maximized.**
  - Usually a scalar $R_t \in \mathbb{R}$ at time t.
- Value function.
  - Expected future rewards under a given behavioural policy.
  - Usually a function $V_\pi(S_t)$.

Essential components of RL algorithms

- Reward signal.
  - **Quantity to be maximized.**
  - Usually a scalar $R_t \in \mathbb{R}$ at time t.
- Value function.
  - Expected future rewards under a given behavioural policy.
  - Usually a function $V_\pi(S_t)$.
- Behavioural policy.
  - Probability distribution over available actions.
  - Written $\pi(a_t \mid s_t) \equiv p(A_t = a_t \mid S_t = s_t)$.
    - Probability of selecting action a_t out of the set of available actions in the current state.
  - Usually high-value actions are chosen with high probability.

Essential components of RL algorithms

- Reward signal.
  - **Quantity to be maximized.**
  - Usually a scalar $R_t \in \mathbb{R}$ at time t.
- Value function.
  - Expected future rewards under a given behavioural policy.
  - Usually a function $V_\pi(S_t)$.
- Behavioural policy.
  - Probability distribution over available actions.
  - Written $\pi(a_t \mid s_t) \equiv p(A_t = a_t \mid S_t = s_t)$.
    - Probability of selecting action a_t out of the set of available actions in the current state.
  - Usually high-value actions are chosen with high probability.

Note

The value function and behavioural policy are inextricably linked.

- Usually we choose actions based on estimated value.
- Value depends on future actions set by the policy.

Evaluating the Value Function

Expected Total Reward

Let $\tau_{t:T} = [s_t, s_{t+1}, ..., s_T]$ be a trajectory of states the mouse passes through from time t to T. Define $G(\tau_{t+1:T}) = \sum_{i=t+1}^{T} R(s_i)$ to be the total reward obtained by the mouse starting from s_t. The simplest value function of s_t we might define is the expected total future reward

$$V_\pi(s_t) \equiv \mathbb{E}_{\mathcal{T}}\left[G(\tau_{t+1:T}) \mid S_t = s_t; \pi\right].$$

Expected Total Reward

Let $\tau_{t:T} = [s_t, s_{t+1}, ..., s_T]$ be a trajectory of states the mouse passes through from time t to T. Define $G(\tau_{t+1:T}) = \sum_{i=t+1}^{T} R(s_i)$ to be the total reward obtained by the mouse starting from s_t. The simplest value function of s_t we might define is the expected total future reward

$$V_\pi(s_t) \equiv \mathbb{E}_{\mathcal{T}}\left[G(\tau_{t+1:T}) \mid S_t = s_t; \pi\right].$$

Recalling that $\mathbb{E}_X[g(X)] = \sum_{x \in X} g(x)p(x)$ (1), we can equivalently write

$$\begin{aligned} V_\pi(s_t) &= \sum_{\tau_{t+1:T} \in \mathcal{T}} G(\tau_{t+1:T}) p(\tau_{t+1:T} \mid s_t; \pi) \\ &= \sum_{\tau_{t+1:T} \in \mathcal{T}} [R(s_{t+1}) + R(s_{t+2}) + ... + R(s_{t+N})] p(s_{t+1}, s_{t+2}, ..., s_T \mid s_t; \pi), \end{aligned}$$

summing over all possible trajectories $\tau \in \mathcal{T}$ that begin with s_t.

Since our agent and environment obey the Markov property, we can find an equivalent recursive definition of the value function.

$$
\begin{aligned}
V_\pi(s_t) &= \sum_{\tau_{t+1} \in \mathcal{T}} \left[G(\tau_{t+1}) + \sum_{\tau_{t+2:T} \in \mathcal{T}} G(\tau_{t+2:T}) p(s_{t+2}, s_{t+3}, \dots, s_T \mid s_{t+1}; \pi) \right] p(s_{t+1} \mid s_t; \pi) \\
&= \sum_{\tau_{t+1} \in \mathcal{T}} [G(\tau_{t+1}) + V_\pi(s_{t+1})] p(s_{t+1} \mid s_t; \pi)
\end{aligned}
$$

Since our agent and environment obey the Markov property, we can find an equivalent recursive definition of the value function.

$$
\begin{aligned}
V_\pi(s_t) &= \sum_{\tau_{t+1} \in \mathcal{T}} \left[G(\tau_{t+1}) + \sum_{\tau_{t+2:T} \in \mathcal{T}} G(\tau_{t+2:T}) p(s_{t+2}, s_{t+3}, \ldots, s_T \mid s_{t+1}; \pi) \right] p(s_{t+1} \mid s_t; \pi) \\
&= \sum_{\tau_{t+1} \in \mathcal{T}} [G(\tau_{t+1}) + V_\pi(s_{t+1})] p(s_{t+1} \mid s_t; \pi)
\end{aligned}
$$

Because $G(\tau_{t+1}) = R(s_{t+1})$, we can substitute and rearrange to obtain an intuitive definition of the value function

$$
V_\pi(s_t) = \mathbb{E}_{s_{t+1}}[R(s_{t+1}) \mid s_t; \pi] + \mathbb{E}_{s_{t+1}}[V_\pi(s_{t+1}) \mid s_t; \pi].
$$

Since our agent and environment obey the Markov property, we can find an equivalent recursive definition of the value function.

$$\begin{aligned} V_\pi(s_t) &= \sum_{\tau_{t+1} \in \mathcal{T}} \left[G(\tau_{t+1}) + \sum_{\tau_{t+2:T} \in \mathcal{T}} G(\tau_{t+2:T}) p(s_{t+2}, s_{t+3}, \ldots, s_T \mid s_{t+1}; \pi) \right] p(s_{t+1} \mid s_t; \pi) \\ &= \sum_{\tau_{t+1} \in \mathcal{T}} [G(\tau_{t+1}) + V_\pi(s_{t+1})] p(s_{t+1} \mid s_t; \pi) \end{aligned}$$

Because $G(\tau_{t+1}) = R(s_{t+1})$, we can substitute and rearrange to obtain an intuitive definition of the value function

$$V_\pi(s_t) = \mathbb{E}_{s_{t+1}}[R(s_{t+1}) \mid s_t; \pi] + \mathbb{E}_{s_{t+1}}[V_\pi(s_{t+1}) \mid s_t; \pi].$$

Think ahead

This recursive definition allows us to *bootstrap*. If computing the true value of $V_\pi(s_{t+1})$ is difficult or impossible, we can use an estimate $\hat{V}_\pi(s_{t+1})$ in its place.

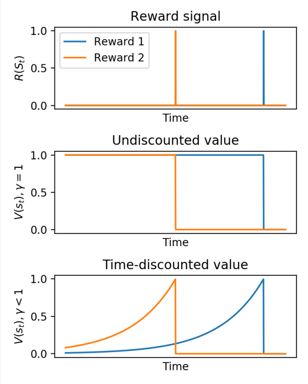

Figure: Comparison of $V_\pi(s_t)$ with discounting ($\gamma < 1$) and without ($\gamma = 1$).

Our previous definition of $V_\pi(s_t)$ has a problem: immediate and delayed rewards of equal magnitude have the same value.

Discounting

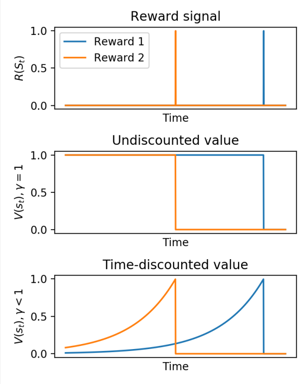

Figure: Comparison of $V_\pi(s_t)$ with discounting ($\gamma < 1$) and without ($\gamma = 1$).

Our previous definition of $V_\pi(s_t)$ has a problem: immediate and delayed rewards of equal magnitude have the same value.

But, intuitively, closer rewards are better.

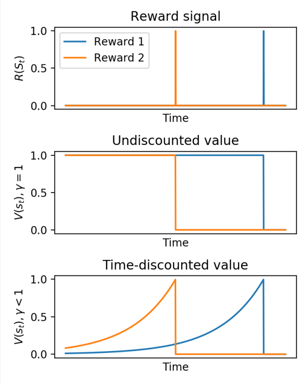

Figure: Comparison of $V_\pi(s_t)$ with discounting ($\gamma < 1$) and without ($\gamma = 1$).

Our previous definition of $V_\pi(s_t)$ has a problem: immediate and delayed rewards of equal magnitude have the same value.

But, intuitively, closer rewards are better.
To fix this, we introduce a time-discounting parameter γ to obtain a new definition

$$V_\pi(s_t) \equiv \mathbb{E}_{s_{t+1}}[R(s_{t+1}) \mid s_t; \pi] + \gamma \mathbb{E}_{s_{t+1}}[V_\pi(s_{t+1}) \mid s_t; \pi],$$

where $0 < \gamma \leq 1$.

Pause to consider

We introduced time discounting using a scaling factor γ applied at each time step. This way, a reward of size 2 that is 100 timesteps away is currently valued at $2\gamma^{100} \leq 2$. How else could we express time discounting?

Pause to consider

We introduced time discounting using a scaling factor γ applied at each time step. This way, a reward of size 2 that is 100 timesteps away is currently valued at $2\gamma^{100} \leq 2$. How else could we express time discounting?

Answer:

$$\gamma^{\frac{\tau_{discount}}{dt}} \approx \frac{1}{e}$$

$$\tau_{discount} \approx \frac{dt}{e \log \gamma},$$

where $\tau_{discount}$ is the *time-constant* of temporal discounting.

Pause to consider

Often the value function cannot be evaluated exactly. What are some reasons this might happen?

Pause to consider

Often the value function cannot be evaluated exactly. What are some reasons this might happen?

- We don't know exactly how our environment and/or behavioural policy work.
  - Mouse is uncertain about layout of maze, time to delayed reward, which lick port is rewarded, level of hunger, etc.
  - Formally, $p(s_{t+1} \mid s_t; \pi)$ is not known.

Pause to consider

Often the value function cannot be evaluated exactly. What are some reasons this might happen?

- We don't know exactly how our environment and/or behavioural policy work.
  - Mouse is uncertain about layout of maze, time to delayed reward, which lick port is rewarded, level of hunger, etc.
  - Formally, $p(s_{t+1} \mid s_t; \pi)$ is not known.
- Long time horizon $T \approx \infty$ makes $G(\tau_{t+1:T}) = \sum_{i=t+1}^{T} R(s_i)$ intractible.
  - This is ubiquitous in animal learning.
  - *Continual learning* is a topic of ongoing research in RL.

Pause to consider

Often the value function cannot be evaluated exactly. What are some reasons this might happen?

- We don't know exactly how our environment and/or behavioural policy work.
  - Mouse is uncertain about layout of maze, time to delayed reward, which lick port is rewarded, level of hunger, etc.
  - Formally, $p(s_{t+1} \mid s_t; \pi)$ is not known.
- Long time horizon $T \approx \infty$ makes $G(\tau_{t+1:T}) = \sum_{i=t+1}^{T} R(s_i)$ intractible.
  - This is ubiquitous in animal learning.
  - *Continual learning* is a topic of ongoing research in RL.
- Large state space $\implies$ *very* large set of trajectories.
  - Maze with many turns $\implies$ *very* large set of possible paths through maze.

Pause to consider

Often the value function cannot be evaluated exactly. What are some reasons this might happen?

- We don't know exactly how our environment and/or behavioural policy work.
  - Mouse is uncertain about layout of maze, time to delayed reward, which lick port is rewarded, level of hunger, etc.
  - Formally, $p(s_{t+1} \mid s_t; \pi)$ is not known.
- Long time horizon $T \approx \infty$ makes $G(\tau_{t+1:T}) = \sum_{i=t+1}^{T} R(s_i)$ intractible.
  - This is ubiquitous in animal learning.
  - *Continual learning* is a topic of ongoing research in RL.
- Large state space $\implies$ *very* large set of trajectories.
  - Maze with many turns $\implies$ *very* large set of possible paths through maze.

Key point

$V_\pi(s_t)$ is easy to define but impossible to evaluate under normal circumstances. Methods to approximate the value function are at the core of RL.

So far we have only seen the *state value function* $V_{\pi}(s_t)$, but other types are possible.

Types of Value Functions

So far we have only seen the *state value function* $V_\pi(s_t)$, but other types are possible. Recall that $V_\pi(s_t)$ only depends on the policy $\pi(a_t \mid s_t)$ because of the term

$$\begin{aligned} p(s_{t+1} \mid s_t; \pi) &= \mathbb{E}_{A_t}[p(s_{t+1} \mid a_t, s_t) \mid s_t] \\ &= \sum_{a_t \in A_t} p(s_{t+1} \mid a_t, s_t)\pi(a_t \mid s_t). \end{aligned}$$

Types of Value Functions

So far we have only seen the *state value function* $V_\pi(s_t)$, but other types are possible. Recall that $V_\pi(s_t)$ only depends on the policy $\pi(a_t \mid s_t)$ because of the term

$$p(s_{t+1} \mid s_t; \pi) = \mathbb{E}_{A_t}[p(s_{t+1} \mid a_t, s_t) \mid s_t]$$
$$= \sum_{a_t \in A_t} p(s_{t+1} \mid a_t, s_t)\pi(a_t \mid s_t).$$

By factoring out the behavioural policy $\pi(a_t \mid s_t)$ from $V_\pi(s_t)$ we can obtain a discounted *action value function*

$$Q(s_t, a_t) \equiv \mathbb{E}_{S_{t+1}}[R(s_{t+1}) \mid s_t, a_t] + \gamma \mathbb{E}_{S_{t+1}, A_{t+1}}[Q_\pi(s_{t+1}, a_{t+1}) \mid s_t, a_t; \pi]$$

which returns the value of taking a specific action a_t in state s_t and following π thereafter.

Key point

- The state value function $V_\pi(s_t)$ gives the expected discounted future rewards following π from state s_t.
  - Easy to understand.
- The action value function $Q_\pi(s_t, a_t)$ gives the expected discounted future rewards by taking action a_t in state s_t and following π thereafter.
  - Meaningful for evaluating the value of particular choices or behaviours.

Example

Consider again our head-fixed mouse presented with two lick ports. Assume only the left lick port is rewarded. What type of value function should we use to model this experiment?

Example

Consider again our head-fixed mouse presented with two lick ports. Assume only the left lick port is rewarded. What type of value function should we use to model this experiment?

Answer: Because the reward obtained depends strongly on the action a_t, an action value function $Q_\pi(s_t, a_t)$ is best.

Example

Consider again our head-fixed mouse presented with two lick ports. Assume only the left lick port is rewarded. What type of value function should we use to model this experiment?
Answer: Because the reward obtained depends strongly on the action a_t, an action value function $Q_\pi(s_t, a_t)$ is best.

If both lick ports are rewarded with probability 0.5, what value function should we use?

Example

Consider again our head-fixed mouse presented with two lick ports. Assume only the left lick port is rewarded. What type of value function should we use to model this experiment?
Answer: Because the reward obtained depends strongly on the action a_t, an action value function $Q_\pi(s_t, a_t)$ is best.

If both lick ports are rewarded with probability 0.5, what value function should we use?
Answer: We should use $V_\pi(s_t)$. While the Q value is still valid, using it here would be needlessly complicated since Q is independent of π.

$$p(s_{t+1} \mid s_t, a_t) = p(s_{t+1} \mid s_t) \implies Q_\pi(S_t = s_t, A_t = x) = Q_\pi(S_t = s_t, A_t = y) \forall x, y.$$

Behavioural Policies

Behavioural Policies

- Probability distribution over available actions.
  - $\pi(a_t \mid s_t) \equiv p(A_t = a_t \mid S_t = S_t)$

Behavioural Policies

- Probability distribution over available actions.
  - $\pi(a_t \mid s_t) \equiv p(A_t = a_t \mid S_t = S_t)$
- If we always choose the best action, then

$$\pi(a_t \mid s_t) \equiv \begin{cases} 1 & A_t = \text{argmax}_{a_t} Q_\pi(s_t, a_t) \\ 0 & \text{otherwise.} \end{cases}$$

Behavioural Policies

- Probability distribution over available actions.
  - $\pi(a_t \mid s_t) \equiv p(A_t = a_t \mid S_t = S_t)$
- If we always choose the best action, then

$$\pi(a_t \mid s_t) \equiv \begin{cases} 1 & A_t = \text{argmax}_{a_t} Q_\pi(s_t, a_t) \\ 0 & \text{otherwise.} \end{cases}$$

  - This leads to optimal behaviour as long as our value function is correct.

Behavioural Policies

- Probability distribution over available actions.
  - $\pi(a_t \mid s_t) \equiv p(A_t = a_t \mid S_t = s_t)$
- If we always choose the best action, then

$$\pi(a_t \mid s_t) \equiv \begin{cases} 1 & A_t = \text{argmax}_{a_t} Q_\pi(s_t, a_t) \\ 0 & \text{otherwise.} \end{cases}$$

  - This leads to optimal behaviour as long as our value function is correct.

Key point

Picking a policy π is easy if our value function is correct. **But** our value function is **almost never correct!**

Exploitation vs. Exploration

Figure: Greed is good.

Figure: Greed is good.

- A policy that always chooses the highest valued action is called *greedy*.

Exploitation vs. Exploration

Figure: Greed is good.

- A policy that always chooses the highest valued action is called *greedy*.
- The policy that is greedy with respect to the *true* value function is optimal.

Figure: Greed is good.

- A policy that always chooses the highest valued action is called *greedy*.
- The policy that is greedy with respect to the *true* value function is optimal.
- In practice, we do not have access to the true value function, and we have to make a tradeoff.

Exploitation vs. Exploration

Figure: Greed is good.

- A policy that always chooses the highest valued action is called *greedy*.
- The policy that is greedy with respect to the *true* value function is optimal.
- In practice, we do not have access to the true value function, and we have to make a tradeoff.
  - Exploitation: actions that are greedy with respect to the current policy.
    - Obtain rewards.

Exploitation vs. Exploration

Figure: Greed is good.

- A policy that always chooses the highest valued action is called *greedy*.
- The policy that is greedy with respect to the *true* value function is optimal.
- In practice, we do not have access to the true value function, and we have to make a tradeoff.
  - Exploitation: actions that are greedy with respect to the current policy.
    - Obtain rewards.
  - Exploration: actions that are non-greedy with respect to the current policy.
    - Search for a better policy.

Strategies for Balancing Exploration and Exploitation

- Off-policy control.
  - Use an explorative policy to control the agent while refining a separate policy. When exploitation is needed, switch to the policy being refined.
- ϵ-softness (aka ϵ-greediness).
  - Use a greedy policy to control behaviour, but take a random action a small percentage of the time.

Definition of ϵ-softness

$$\pi(a_t \mid s_t) \equiv \begin{cases} 1-\epsilon & A_t = \text{argmax}_{a_t} Q_\pi(s_t, a_t) \\ \frac{\epsilon}{N-1} & \text{otherwise} \end{cases}$$

for a policy with N possible actions in state s_t.

Example

Consider a freely-moving mouse presented with two lick ports at opposite ends of a box, which are rewarded with separate, non-constant probabilities.

Example

Consider a freely-moving mouse presented with two lick ports at opposite ends of a box, which are rewarded with separate, non-constant probabilities.

At each timestep, the mouse chooses from a set of available actions $\mathcal{A} = \{\text{lick current port}, \text{switch to other port}, \text{do nothing}\}$ that cause it to transition deterministically between states in $\mathcal{S} = \{\text{sit at port 1}, \text{sit at port 2}, \text{lick port 1}, \text{lick port 2}\}$.

Example

Consider a freely-moving mouse presented with two lick ports at opposite ends of a box, which are rewarded with separate, non-constant probabilities.

At each timestep, the mouse chooses from a set of available actions $\mathcal{A} = \{\text{lick current port}, \text{switch to other port}, \text{do nothing}\}$ that cause it to transition deterministically between states in $\mathcal{S} = \{\text{sit at port 1}, \text{sit at port 2}, \text{lick port 1}, \text{lick port 2}\}$.

- A neural population containing three ensembles could encode $Q_\pi(s_t, a_t)$ for each available action in the current state.

Example

Consider a freely-moving mouse presented with two lick ports at opposite ends of a box, which are rewarded with separate, non-constant probabilities.

At each timestep, the mouse chooses from a set of available actions $\mathcal{A} = \{\text{lick current port}, \text{switch to other port}, \text{do nothing}\}$ that cause it to transition deterministically between states in $\mathcal{S} = \{\text{sit at port 1}, \text{sit at port 2}, \text{lick port 1}, \text{lick port 2}\}$.

- A neural population containing three ensembles could encode $Q_\pi(s_t, a_t)$ for each available action in the current state.
- Lateral inhibition in this population could implement a winner-take-all argmax function over available actions, leading to greedy action selection.

Example

Consider a freely-moving mouse presented with two lick ports at opposite ends of a box, which are rewarded with separate, non-constant probabilities.

At each timestep, the mouse chooses from a set of available actions $\mathcal{A} = \{\text{lick current port}, \text{switch to other port}, \text{do nothing}\}$ that cause it to transition deterministically between states in $\mathcal{S} = \{\text{sit at port 1}, \text{sit at port 2}, \text{lick port 1}, \text{lick port 2}\}$.

- A neural population containing three ensembles could encode $Q_\pi(s_t, a_t)$ for each available action in the current state.
- Lateral inhibition in this population could implement a winner-take-all argmax function over available actions, leading to greedy action selection.
- Moderate lateral inhibition could implement an ϵ-soft policy via a soft argmax function.

Food for thought

Consider the set of three ensembles encoding $Q_\pi(s_t, a_t) \forall a_t \in \mathcal{A}$ from the previous example. If we activate all three ensembles optogenetically or chemogenetically, how does the representation of Q_π change, and how would this affect behaviour?

Food for thought

Consider the set of three ensembles encoding $Q_\pi(s_t, a_t) \forall a_t \in \mathcal{A}$ from the previous example. If we activate all three ensembles optogenetically or chemogenetically, how does the representation of Q_π change, and how would this affect behaviour?

- All $Q_\pi(s_t, a_t)$ set to the same high value.
  - "Greedy" behaviour becomes random.

Food for thought

Consider the set of three ensembles encoding $Q_\pi(s_t, a_t) \forall a_t \in \mathcal{A}$ from the previous example. If we activate all three ensembles optogenetically or chemogenetically, how does the representation of Q_π change, and how would this affect behaviour?

- All $Q_\pi(s_t, a_t)$ set to the same high value.
  - "Greedy" behaviour becomes random.
- All $Q_\pi(s_t, a_t)$ increased by a fixed amount.
  - Behaviour becomes less greedy?

Food for thought

Consider the set of three ensembles encoding $Q_\pi(s_t, a_t) \forall a_t \in \mathcal{A}$ from the previous example. If we activate all three ensembles optogenetically or chemogenetically, how does the representation of Q_π change, and how would this affect behaviour?

- All $Q_\pi(s_t, a_t)$ set to the same high value.
  - "Greedy" behaviour becomes random.
- All $Q_\pi(s_t, a_t)$ increased by a fixed amount.
  - Behaviour becomes less greedy?
- All $Q_\pi(s_t, a_t)$ increased multiplicatively.
  - Behaviour does not change at all?

Value Function Optimization

Multiple Techniques for Optimizing Q_π or V_π

Most are based on biologically unrealistic assumptions.

Multiple Techniques for Optimizing Q_π or V_π

Most are based on biologically unrealistic assumptions.

- Dynamic programming.
  - Requires a perfect model of the environment.

Multiple Techniques for Optimizing Q_π or V_π

Most are based on biologically unrealistic assumptions.

- Dynamic programming.
  - Requires a perfect model of the environment.
- Monte-Carlo methods.
  - Requires a rigid trial structure.
  - All value function adjustments are perfomed at trial end.

Multiple Techniques for Optimizing Q_π or V_π

Most are based on biologically unrealistic assumptions.

- Dynamic programming.
  - Requires a perfect model of the environment.
- Monte-Carlo methods.
  - Requires a rigid trial structure.
  - All value function adjustments are perfomed at trial end.

Most biologically plausible optimization algorithm is *temporal difference* (TD) learning.

- Perfect environmental model is not required.
  - In fact, no environmental model is needed at all.
- Value functions are updated in real time.

Pause to consider

Recall that state and action value functions include a term $p(s_{t+1} \mid s_t; \pi)$ or $p(s_{t+1} \mid s_t, a_t)$ to model environmental state transitions.

Pause to consider

Recall that state and action value functions include a term $p(s_{t+1} \mid s_t; \pi)$ or $p(s_{t+1} \mid s_t, a_t)$ to model environmental state transitions.
"Model-free" methods (e.g. TD and Monte-Carlo) avoid specifying full distributions for these terms by sampling from them directly.

Pause to consider

Recall that state and action value functions include a term $p(s_{t+1} \mid s_t; \pi)$ or $p(s_{t+1} \mid s_t, a_t)$ to model environmental state transitions.
“Model-free” methods (e.g. TD and Monte-Carlo) avoid specifying full distributions for these terms by sampling from them directly.
Model-free and model-based methods are thought to reflect habitual and goal-directed behaviours.

Pause to consider

Recall that state and action value functions include a term $p(s_{t+1} \mid s_t; \pi)$ or $p(s_{t+1} \mid s_t, a_t)$ to model environmental state transitions.
"Model-free" methods (e.g. TD and Monte-Carlo) avoid specifying full distributions for these terms by sampling from them directly.
Model-free and model-based methods are thought to reflect habitual and goal-directed behaviours.

Example: model-free learning

Consider a fledgeling electrophysiologist attempting to seal onto a cell.

$$\mathcal{S} = \{\text{not sealed}, \text{sealed}\}$$
$$\mathcal{A} = \{\text{amount of suction} \in \{0, 1, ..., 10\}\}$$

The experimenter does not know eg
$p(S_{t+1} = \text{sealed} \mid S_t = \text{not sealed}, A_t = 3) > p(S_{t+1} = \text{sealed} \mid S_t = \text{not sealed}, A_t = 7)$, but will eventually learn to apply the correct amount of suction by sampling from this distribution.

Recall

Earlier we obtained recursive definitions of V and Q value functions of the form

$$V_\pi(s_t) \equiv \mathbb{E}_{s_{t+1}}[R(s_{t+1}) \mid s_t; \pi] + \gamma \mathbb{E}_{s_{t+1}}[V_\pi(s_{t+1}) \mid s_t; \pi].$$

Recall

Earlier we obtained recursive definitions of V and Q value functions of the form

$$V_\pi(s_t) \equiv \mathbb{E}_{s_{t+1}}[R(s_{t+1}) \mid s_t; \pi] + \gamma \mathbb{E}_{s_{t+1}}[V_\pi(s_{t+1}) \mid s_t; \pi].$$

Recall that these allow us to use a bootstrapped *estimate* $\hat{V}_\pi(s_{t+1})$ in place of $V_\pi(s_{t+1})$ when the latter is not available or difficult to compute.

Recall

Earlier we obtained recursive definitions of V and Q value functions of the form

$$V_\pi(s_t) \equiv \mathbb{E}_{s_{t+1}}[R(s_{t+1}) \mid s_t; \pi] + \gamma \mathbb{E}_{s_{t+1}}[V_\pi(s_{t+1}) \mid s_t; \pi].$$

Recall that these allow us to use a bootstrapped *estimate* $\hat{V}_\pi(s_{t+1})$ in place of $V_\pi(s_{t+1})$ when the latter is not available or difficult to compute.

Key point

Core idea of TD learning is to compare three quantities across short time intervals.

Recall

Earlier we obtained recursive definitions of V and Q value functions of the form

$$V_\pi(s_t) \equiv \mathbb{E}_{s_{t+1}}[R(s_{t+1}) \mid s_t; \pi] + \gamma \mathbb{E}_{s_{t+1}}[V_\pi(s_{t+1}) \mid s_t; \pi].$$

Recall that these allow us to use a bootstrapped *estimate* $\hat{V}_\pi(s_{t+1})$ in place of $V_\pi(s_{t+1})$ when the latter is not available or difficult to compute.

Key point

Core idea of TD learning is to compare three quantities across short time intervals.

1. Observed reward ($R(s_{t+1})$).

Recall

Earlier we obtained recursive definitions of V and Q value functions of the form

$$V_\pi(s_t) \equiv \mathbb{E}_{s_{t+1}}[R(s_{t+1}) \mid s_t; \pi] + \gamma \mathbb{E}_{s_{t+1}}[V_\pi(s_{t+1}) \mid s_t; \pi].$$

Recall that these allow us to use a bootstrapped *estimate* $\hat{V}_\pi(s_{t+1})$ in place of $V_\pi(s_{t+1})$ when the latter is not available or difficult to compute.

Key point

Core idea of TD learning is to compare three quantities across short time intervals.

1. Observed reward ($R(s_{t+1})$).
2. Bootstrapped future value estimate ($\hat{V}_\pi(s_{t+1})$).

Recall

Earlier we obtained recursive definitions of V and Q value functions of the form

$$V_\pi(s_t) \equiv \mathbb{E}_{s_{t+1}}[R(s_{t+1}) \mid s_t; \pi] + \gamma \mathbb{E}_{s_{t+1}}[V_\pi(s_{t+1}) \mid s_t; \pi].$$

Recall that these allow us to use a bootstrapped *estimate* $\hat{V}_\pi(s_{t+1})$ in place of $V_\pi(s_{t+1})$ when the latter is not available or difficult to compute.

Key point

Core idea of TD learning is to compare three quantities across short time intervals.

1. Observed reward ($R(s_{t+1})$).
2. Bootstrapped future value estimate ($\hat{V}_\pi(s_{t+1})$).
3. Current value estimate ($V_\pi(s_t)$).

Temporal Difference Learning

For an agent exploring its environment, the following TD update is performed at each timestep

$$Q_\pi(s_t, a_t) \leftarrow Q_\pi(s_t, a_t) + \alpha \left[R(s_{t+1}) + \gamma \hat{Q}_\pi(s_{t+1}, a_{t+1}) - Q_\pi(s_t, a_t) \right]$$

where $0 < \alpha \leq 1$ is an effective learning rate, γ is the temporal discounting parameter, and $\hat{Q}_\pi(s_{t+1}, a_{t+1})$ is the estimated value of the next state action pair from a lookup table.

Temporal Difference Learning

For an agent exploring its environment, the following TD update is performed at each timestep

$$Q_\pi(s_t, a_t) \leftarrow Q_\pi(s_t, a_t) + \alpha \left[R(s_{t+1}) + \gamma \hat{Q}_\pi(s_{t+1}, a_{t+1}) - Q_\pi(s_t, a_t) \right]$$

where $0 < \alpha \leq 1$ is an effective learning rate, γ is the temporal discounting parameter, and $\hat{Q}_\pi(s_{t+1}, a_{t+1})$ is the estimated value of the next state action pair from a lookup table.

If we define a TD error term

$$\delta_t \equiv R(s_{t+1}) + \gamma \hat{Q}_\pi(s_{t+1}, a_{t+1}) - Q_\pi(s_t, a_t),$$

we can write the TD update more succintly

$$Q_\pi(s_t, a_t) \leftarrow Q_\pi(s_t, a_t) + \alpha \delta_t.$$

Properties of TD Reward Prediction Errors

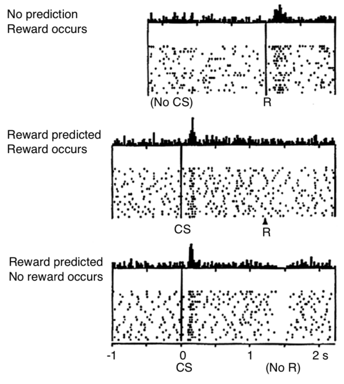

Figure: Dopaminergic reward prediction errors. Schultz et al. (1997)

Yes, *those* RPEs.

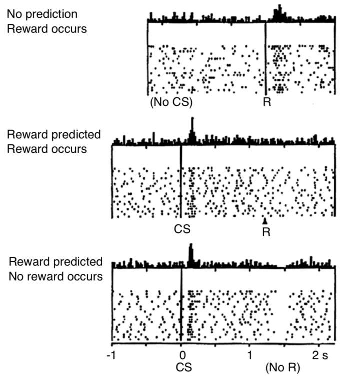

Figure: Dopaminergic reward prediction errors. Schultz et al. (1997)

Yes, *those* RPEs.

Not quite “experienced minus expected reward”.

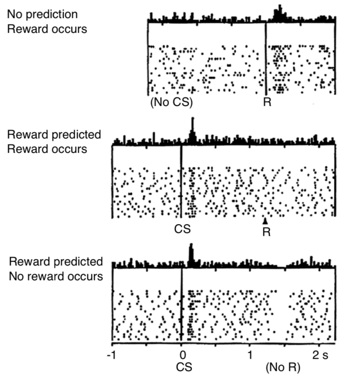

Figure: Dopaminergic reward prediction errors. Schultz et al. (1997)

Yes, *those* RPEs.

Not quite "experienced minus expected reward".

$$\delta_t \equiv R(s_{t+1}) + \gamma \hat{Q}_\pi(s_{t+1}, a_{t+1}) - Q_\pi(s_t, a_t)$$

Properties of TD Reward Prediction Errors

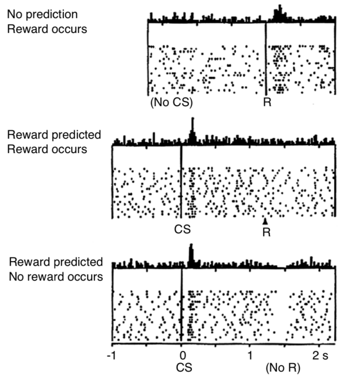

Figure: Dopaminergic reward prediction errors. Schultz et al. (1997)

Yes, *those* RPEs.

Not quite "experienced minus expected reward".

$$\delta_t \equiv R(s_{t+1}) + \gamma \hat{Q}_\pi(s_{t+1}, a_{t+1}) - Q_\pi(s_t, a_t)$$

Key point

- TD errors are partly due to temporal differences in the value function.
- TD errors only *asymptotically* approach zero.
- Therefore, the shape of TD RPEs partly reflect the shape of the value function.

TD Errors Come in Many Shapes and Sizes

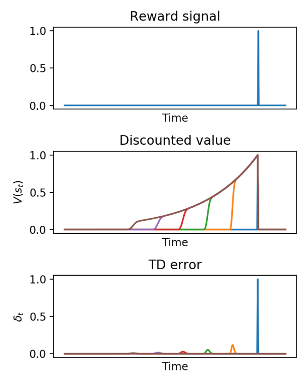

Figure: TD errors in a classical conditioning task (cue not shown).

TD Errors Come in Many Shapes and Sizes

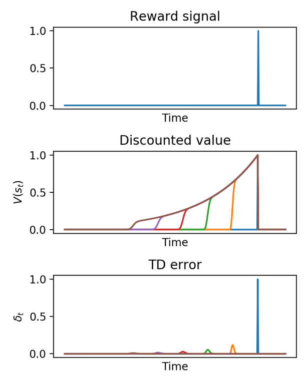

Figure: TD errors in a classical conditioning task (cue not shown).

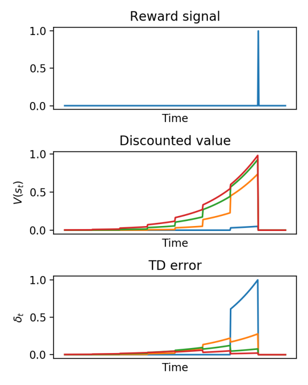

Figure: *n*-step TD errors in the same task.

TD Errors Come in Many Shapes and Sizes

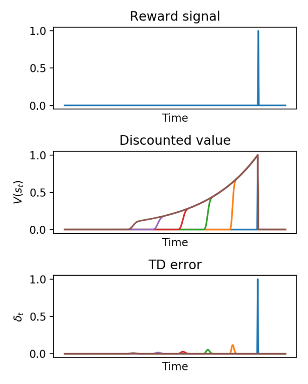

Figure: TD errors in a classical conditioning task (cue not shown).

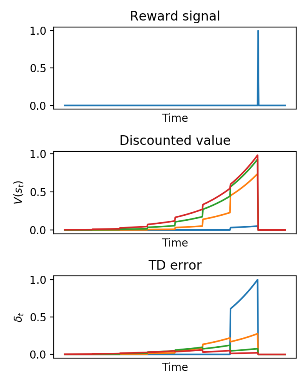

Figure: *n*-step TD errors in the same task.

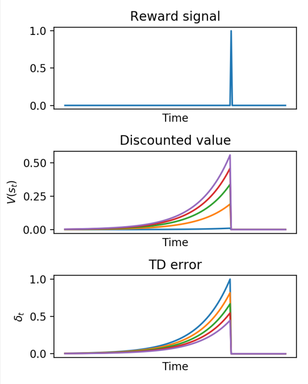

Figure: TD errors based on eligibility traces.

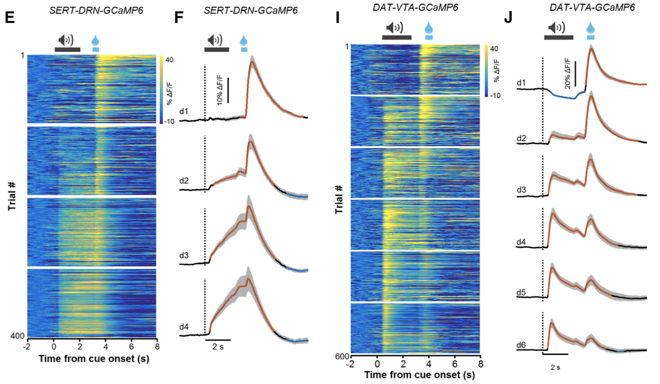

Figure: Population responses of VTA DA and DRN 5HT neurons over the course of learning in a classical conditioning task. Zhong et al. (2017)

Food for thought

Does the DRN encode $R(s_{t+1}) + \hat{Q}_\pi(s_{t+1}, a_{t+1})$?

Food for thought

Does the DRN encode $R(s_{t+1}) + \hat{Q}_\pi(s_{t+1}, a_{t+1})$?

Reward $+Q$ **value**	**DRN**
Responds to rewards	Population responds to rewards
Not directly used for learning	Not reinforcing
Related to action choices	Regulates behaviour
Closely related to δ_t	Strong connection with DA system
Heterogenous responses to anticipated rewards	Maybe?

Food for thought

Does the DRN encode $R(s_{t+1}) + \hat{Q}_\pi(s_{t+1}, a_{t+1})$?

Reward $+Q$ **value**	**DRN**
Responds to rewards	Population responds to rewards
Not directly used for learning	Not reinforcing
Related to action choices	Regulates behaviour
Closely related to δ_t	Strong connection with DA system
Heterogenous responses to anticipated rewards	Maybe?

Perhaps a better question is: Over *which* rewards and actions can the DRN be seen as encoding a reward signal and value function?

Alternative State Space Representations

State Vectors (1)

So far we have considered models that represent the environment as existing in a particular discrete state $s_t \in \mathcal{S}$.

- Difficult to optimize for any realistic environment.
  - Large state spaces $\mathcal{S}$ are required.
  - No generalization across similar states.

State Vectors (1)

So far we have considered models that represent the environment as existing in a particular discrete state $s_t \in \mathcal{S}$.

- Difficult to optimize for any realistic environment.
  - Large state spaces $\mathcal{S}$ are required.
  - No generalization across similar states.

Example

Consider a mouse in a classical conditioning task.

$$\mathcal{S} = \{\text{go cue on and smell of experimenter A}, \text{go cue on and smell of experimenter B}, \ldots\}$$

Under this model, knowledge about the go cue acquired under experimenter A cannot be applied under experimenter B.

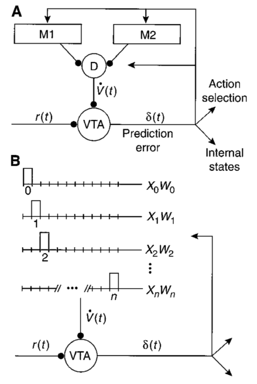

Figure: State vectors in the VTA. Schultz et al. (1997).

Instead, we can represent states as vector combinations of discrete *features*.

Notation

$\mathbf{s}_t$: Vector of state features.
- A list of all state features.

θ: Learnable parameters.
- Typically a vector of weights corresponding to each feature.

Example

Consider again the mouse in a classical conditioning task. Write the environmental state as a vector

$$\mathbf{s}_t = \begin{bmatrix}\text{go cue} \in \{0,1\} & \text{smell A} \in \{0,1\} & \text{smell B} \in \{0,1\} & \cdots\end{bmatrix}^\top$$

with a corresponding weight vector

$$\theta = \begin{bmatrix}w_{\text{go cue}} & w_{\text{smell A}} & w_{\text{smell B}} & \cdots\end{bmatrix}^\top.$$

Example

Consider again the mouse in a classical conditioning task. Write the environmental state as a vector

$$\mathbf{s}_t = \begin{bmatrix} \text{go cue} \in \{0,1\} & \text{smell A} \in \{0,1\} & \text{smell B} \in \{0,1\} & \cdots \end{bmatrix}^\top$$

with a corresponding weight vector

$$\theta = \begin{bmatrix} w_{\text{go cue}} & w_{\text{smell A}} & w_{\text{smell B}} & \cdots \end{bmatrix}^\top .$$

The corresponding state value function might be

$$V_\pi(\mathbf{s}_t; \theta) \equiv \mathbf{s}_t \cdot \theta = \sum s_i \theta_i$$

where θ is adjusted during learning.

Example (continued)

Suppose that the animal learns a strong association with the go cue, such that the weights in θ reach an equilibrium

$$\theta \to \begin{bmatrix} 1 & 0 & 0 & \cdots \end{bmatrix}^\top .$$

Example (continued)

Suppose that the animal learns a strong association with the go cue, such that the weights in θ reach an equilibrium

$$\theta \to \begin{bmatrix} 1 & 0 & 0 & \cdots \end{bmatrix}^\top.$$

If we introduce a new predictive stimulus after the go cue (e.g. smell of experimenter B), what will happen to θ?

Example (continued)

Suppose that the animal learns a strong association with the go cue, such that the weights in θ reach an equilibrium

$$\theta \to \begin{bmatrix} 1 & 0 & 0 & \cdots \end{bmatrix}^\top.$$

If we introduce a new predictive stimulus after the go cue (e.g. smell of experimenter B), what will happen to θ?

Answer: Remember that the TD update rule is defined as

$$V_\pi(\mathbf{s}_t; \theta) \leftarrow V_\pi(\mathbf{s}_t; \theta) + \alpha\delta_t.$$

Note that since the animal has already learned to fully predict the reward from the go cue, $\delta_t \approx 0$.

Example (continued)

Suppose that the animal learns a strong association with the go cue, such that the weights in θ reach an equilibrium

$$\theta \to \begin{bmatrix} 1 & 0 & 0 & \cdots \end{bmatrix}^\top.$$

If we introduce a new predictive stimulus after the go cue (e.g. smell of experimenter B), what will happen to θ?

Answer: Remember that the TD update rule is defined as

$$V_\pi(\mathbf{s}_t; \theta) \leftarrow V_\pi(\mathbf{s}_t; \theta) + \alpha\delta_t.$$

Note that since the animal has already learned to fully predict the reward from the go cue, $\delta_t \approx 0$.
Therefore, there is no basis for updating θ to reflect the new cue. The smell of experimenter B is said to be *blocked*.

Pause to consider

How else could we implement $V_\pi(\mathbf{s}_t; \theta)$?

Pause to consider

How else could we implement $V_\pi(\mathbf{s}_t; \theta)$?

- As a simple nonlinear combination of input features.
  - E.g. $\sum s_i^2\theta_i$

Pause to consider

How else could we implement $V_\pi(\mathbf{s}_t; \theta)$?

- As a simple nonlinear combination of input features.
  - E.g. $\sum s_i^2 \theta_i$
- As a deep neural network.
  - C.f. deep reinforcement learning.

Representing Time

- Time can be represented in a state vector in multiple ways.
  - Depends on choice of temporal basis functions.
- Shape of value function depends strongly on time representation.

Representing Time

- Time can be represented in a state vector in multiple ways.
  - Depends on choice of temporal basis functions.
- Shape of value function depends strongly on time representation.
- Limiting shape of RPEs depends strongly on time representation.
  - In some circumstances, ramping RPEs may be observed (see Gershman (2014) comment on Howe et al. (2013)).

Topics for Further Reading

Topics for Futher Reading

- Eligibility traces.
  - Fuzzy multi-step TD learning for state vectors.
- Off-policy control.
  - Separate policies for exploration and exploitation.
- Actor-critic algorithms.
  - Possibly implemented by basal ganglia.
- The deadly triad.
  1. Function approximation.
  2. Bootstrapping.
  3. Off-policy training.

Conclusion

Conclusion

Conclusion

- Reinforcement learning is a subfield of artificial intelligence and machine learning.

Conclusion

- Reinforcement learning is a subfield of artificial intelligence and machine learning.
- RL agents have two main components.
  1. Value function $V_\pi(s_t)$ or $Q_\pi(s_t, a_t)$.
     - Used for *prediction*.
  2. Behavioural policy $\pi(a_t \mid s_t)$.
     - Used for *control*.

Conclusion

- Reinforcement learning is a subfield of artificial intelligence and machine learning.
- RL agents have two main components.
  1. Value function $V_\pi(s_t)$ or $Q_\pi(s_t, a_t)$.
     - Used for *prediction*.
  2. Behavioural policy $\pi(a_t \mid s_t)$.
     - Used for *control*.
- Temporal difference (TD) learning is a biologically plausible optimization algorithm.
  - Implements learning of habit-like behaviours.

Thank You!