Unifying task specification for reinforcement learning

Martha White
Assistant Professor
University of Alberta
Problem setting

Agent

Environment

state $S_t$

reward $R_t$

$R_{t+1}$

$S_{t+1}$

action $A_t$
Problem setting

Agent

Environment

state $S_t$

reward $R_t$

$R_{t+1}$

$S_{t+1}$

Infinite horizon

action $A_t$
Markov decision process: \((S, A, Pr, R, \gamma_c)\)
What is this talk about?

\[ \gamma_c \in [0, 1) \]
What is this talk about?

\[ \gamma_c \in [0, 1) \quad \rightarrow \quad \gamma : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1] \]
Main take-away
Main take-away

- Transition-based discounting is useful for you
Main take-away

• Transition-based discounting is useful for you
  • to simplify algorithm development
Main take-away

- **Transition-based discounting** is useful for you
  - to simplify algorithm development
  - to unify theoretical characterizations
Main take-away

- Transition-based discounting is useful for you
  - to simplify algorithm development
  - to unify theoretical characterizations
  - to simplify implementation
Outline

• Generalization to transition-based discounting
• The theoretical and algorithmic implications
  • generalized Bellman operators
• Utility of the generalized problem formalism
Returns (continuing)

\[ G_t = \sum_{i=0}^{\infty} \gamma_c^i R_{t+1+i} = R_{t+1} + \gamma_c G_{t+1} \]

\[ V(s) = \mathbb{E}[G_t \mid S_t = s] \quad \gamma_c \in [0, 1) \]
Returns (continuing)

\[ G_t = \sum_{i=0}^{\infty} \gamma_c^i R_{t+1+i} = R_{t+1} + \gamma_c G_{t+1} \]

\[ V(s) = \mathbb{E}[G_t \mid S_t = s] \quad \gamma_c \in [0, 1) \]

\( s_0, a_0, r_1, s_1, a_1, r_2, s_2, a_2, r_3, s_3, a_3, \ldots \)
Returns (continuing)

\[ G_t = \sum_{i=0}^{\infty} \gamma_c^i R_{t+1+i} = R_{t+1} + \gamma_c G_{t+1} \]

\[ V(s) = \mathbb{E}[G_t \mid S_t = s] \quad \gamma_c \in [0, 1) \]

\[ s_0, a_0, r_1, s_1, a_1, r_2, s_2, a_2, r_3, s_3, a_3, \ldots \]

\[ r_1 + \gamma_c r_2 + \gamma_c^2 r_3 + \ldots \]
Returns (continuing)

\[ G_t = \sum_{i=0}^{\infty} \gamma_c^i R_{t+1+i} = R_{t+1} + \gamma_c G_{t+1} \]

\[ V(s) = \mathbb{E}[G_t \mid S_t = s] \quad \gamma_c \in [0, 1) \]

\[ s_0, a_0, r_1, s_1, a_1, r_2, s_2, a_2, r_3, s_3, a_3, \ldots \]

\[ r_2 + \gamma_c r_3 + \gamma_c^2 r_4 + \ldots \]
$G_t = \sum_{i=0}^{\infty} \gamma_c^i R_{t+1+i} = R_{t+1} + \gamma_c G_{t+1}$

$V(s) = \mathbb{E}[G_t \mid S_t = s]$ \hspace{1cm} \gamma_c \in [0, 1)$

$s_0, a_0, r_1, s_1, a_1, r_2, s_2, a_2, r_3, s_3, a_3, \ldots$

$\downarrow$

$r_3 + \gamma_c r_4 + \gamma_c^2 r_5 + \ldots$
Selection of discount

\[ G_t = \sum_{i=0}^{\infty} \gamma_c^i R_{t+1+i} \quad \gamma_c \in [0, 1) \]

\( \gamma_c \) reflects horizon
Returns (episodic)
Returns (episodic)

$s_0 = (7, 3), a_0 = \text{Sth}, r_1 = -1, s_1 = (7, 2), a_1 = \text{Sth}, r_2 = -1, s_2 = (7, 1)$

$r_1 + r_2$

$r_2$

Null
How do we unify the two?

- Algorithms and theory treat the two cases separately
- Absorbing state not a complete solution
How do we unify the two?

- Algorithms and theory treat the two cases separately
- Absorbing state not a complete solution
- Recent generalizations to state-based discount almost the complete solution
Unification using transition-based discounting

- Discount generalized to a function on \((s,a,s')\)
  \[
  \gamma : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]
  \]
- Can smoothly encode continuing or episodic
  - …and specify a whole new set of returns
Generalized return

\[ \gamma : S \times A \times S \rightarrow [0, 1] \]

\[ G_t = \sum_{i=0}^{\infty} \left( \prod_{j=0}^{i-1} \gamma(S_{t+j}, A_{t+j}, S_{t+j+1}) \right) R_{t+1+i} \]

\[ = R_{t+1} + \gamma_{t+1} G_{t+1} \]

\[ \gamma_{t+1} = \gamma(S_t, A_t, S_{t+1}) \]
Generalized return

\[ \gamma : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1] \]

\[ G_t = \sum_{i=0}^{\infty} \left( \prod_{j=0}^{i-1} \gamma(S_{t+j}, A_{t+j}, S_{t+j+1}) \right) R_{t+1+i} \]

\[ = R_{t+1} + \gamma_{t+1} G_{t+1} \quad \gamma_{t+1} = \gamma(S_t, A_t, S_{t+1}) \]

If \( \gamma(s, a, s') = \gamma_c \)

\[ \prod_{j=0}^{i-1} \gamma(S_{t+j}, A_{t+j}, S_{t+j+1}) = \gamma_c^{i-1} \]
Encoding episodic tasks

- $\gamma(s,a,s') = 0$ for a terminal transition
- $s'$ is the start state for the next episode
- Return is truncated at termination by the product of discounts, with $\gamma(s,a,s') = 0$

$$G_t = R_{t+1} + \gamma_{t+1} G_{t+1}$$
Example: taxi domain

Actions:
N, E, S, W, Pickup, Drop-off

States:
(x, y, passenger location)

State: (2, 1, 3)
Location can be (0,1,2,3) or 4 for in taxi

Passenger in black square
Example: taxi domain

\[ G_t = R_{t+1} + \gamma_{t+1} G_{t+1} \]
Example: taxi domain

• What are the transition probabilities at (4,4,3)?

\[ G_t = R_{t+1} + \gamma_{t+1} G_{t+1} \]
Example: taxi domain

• What are the transition probabilities at (4,4,3)?

• $P((4,4,3), \text{Pick-up}, (4,4,4)) = 1.0$

\[
G_t = R_{t+1} + \gamma_{t+1} G_{t+1}
\]
Example: taxi domain

- What are the transition probabilities at (4,4,3)?
  - $P((4,4,3), \text{Pick-up}, (4,4,4)) = 1.0$
- What is the discount function?

$$G_t = R_{t+1} + \gamma_{t+1} G_{t+1}$$
Example: taxi domain

• What are the transition probabilities at (4,4,3)?

  • \( P((4,4,3), \text{Pick-up}, (4,4,4)) = 1.0 \)

• What is the discount function?

  • \( \gamma((4,4,3), \text{Pick-up}, (4,4,4)) = 0.0, \text{else } 1.0 \)

\[
G_t = R_{t+1} + \gamma_{t+1} G_{t+1}
\]
Example: taxi domain

- What are the transition probabilities at (4,4,3)?
  - \( P((4,4,3), \text{Pick-up}, (4,4,4)) = 1.0 \)

- What is the discount function?
  - \( \gamma((4,4,3), \text{Pick-up}, (4,4,4)) = 0.0, \text{ else } 1.0 \)

- Why not \( \gamma_s((4,4,4)) = 0.0 ? \)

\[ G_t = R_{t+1} + \gamma_{t+1} G_{t+1} \]
Example: taxi domain

• What are the transition probabilities at (4,4,3)?
  • \( P((4,4,3), \text{Pick-up}, (4,4,4)) = 1.0 \)

• What is the discount function?
  • \( \gamma((4,4,3), \text{Pick-up}, (4,4,4)) = 0.0, \text{else 1.0} \)

• Why not \( \gamma_s((4,4,4)) = 0.0 \)?

• Why not add a termination state?

\[
G_t = R_{t+1} + \gamma_{t+1} G_{t+1}
\]
What are the implications?
What are the implications?

- Unified analysis for episodic and continuing problems —> can extend previous results
What are the implications?

- Unified analysis for episodic and continuing problems — can extend previous results
- How does this change the algorithms?
What are the implications?

• Unified analysis for episodic and continuing problems —> can extend previous results

• How does this change the algorithms?
  • very little

• avoids two versions of an algorithm
Do all algorithms extend?

• Can define a generalized Bellman operator

• recursive form for return, that is Markov

\[ G_t = \sum_{i=0}^{\infty} \left( \prod_{j=1}^{i} \gamma_{t+j} \right) R_{t+1+i} = R_{t+1} + \gamma_{t+1} G_{t+1} \]
Do all algorithms extend?

- Can define a generalized Bellman operator
  - recursive form for return, that is Markov

\[ G_t = \sum_{i=0}^{\infty} \left( \prod_{j=1}^{i} \gamma_{t+j} \right) R_{t+1+i} = R_{t+1} + \gamma_{t+1} G_{t+1} \]

- Replace \( \gamma_c \) with \( \gamma_{t+1} \)
Definitions for the operator

\[ v_\pi(s) = \mathbb{E}[G_t | S_t = s] \]
\[ = \mathbb{E}[R_{t+1} + \gamma_{t+1}G_{t+1} | S_t = s] \]
\[ = \mathbb{E}[R_{t+1} | S_t = s] + \mathbb{E}[\gamma_{t+1}v_\pi(S_{t+1}) | S_t = s] \]
Definitions for the operator

\[ v_{\pi}(s) = \mathbb{E}[G_t | S_t = s] \]
\[ = \mathbb{E}[R_{t+1} + \gamma_{t+1} G_{t+1} | S_t = s] \]
\[ = \mathbb{E}[R_{t+1} | S_t = s] + \mathbb{E}[\gamma_{t+1} v_{\pi}(S_{t+1}) | S_t = s] \]
\[ = r_{\pi}(s) + \sum_{s'} P_{\pi,\gamma}(s, s') v_{\pi}(s') \]

\[ P_{\pi,\gamma}(s, s') = \sum_a \pi(s, a) \Pr(s, a, s') \gamma(s, a, s') \]

\[ r_{\pi}(s) = \sum_a \pi(s, a) \sum_{s'} \Pr(s, a, s') r(s, a, s') \]
Definitions for the operator

\[ v_\pi(s) = \mathbb{E}[G_t | S_t = s] \]
\[ = \mathbb{E}[R_{t+1} + \gamma_{t+1} G_{t+1} | S_t = s] \]
\[ = \mathbb{E}[R_{t+1} | S_t = s] + \mathbb{E}[\gamma_{t+1} v_\pi(S_{t+1}) | S_t = s] \]
\[ = r_\pi(s) + \sum_{s'} P_{\pi,\gamma}(s, s') v_\pi(s') \]

\[ P_{\pi,\gamma}(s, s') = \sum_a \pi(s, a) \Pr(s, a, s') \gamma(s, a, s') \]
\[ r_\pi(s) = \sum_a \pi(s, a) \sum_{s'} \Pr(s, a, s') r(s, a, s') \]
Bellman operator

\[ v_\pi = r_\pi + \sum_{s'} P_{\pi,\gamma}(\cdot, s') v_\pi(s') = r_\pi + P_{\pi,\gamma} v_\pi \]
Bellman operator

\[ v_\pi = r_\pi + \sum_{s'} P_{\pi, \gamma}(\cdot, s') v_\pi(s') = r_\pi + P_{\pi, \gamma} v_\pi \]
e.g., \[ r_\pi + \gamma_c P_\pi v_\pi \]
Bellman operator

\[ v_\pi = r_\pi + \sum_{s'} P_{\pi, \gamma}(s, s') v_\pi(s') = r_\pi + P_{\pi, \gamma} v_\pi \]

e.g., \[ r_\pi + \gamma c P_{\pi} v_\pi \]

Bellman operator

\[ T v = r_\pi + P_{\pi, \gamma} v \]

Reach solution (fixed point) when \( T v = v \)

Given models, can use dynamic programming
Otherwise, stochastic approximations (e.g., TD)
Key property: contraction

• The operator $T$ has to be a contraction

• If $T$ is an expansion, then repeated application of $T$ to $v$ could expand to infinity
Key property: contraction

- The operator $T$ has to be a contraction

- If $T$ is an expansion, then repeated application of $T$ to $v$ could expand to infinity

$$\|Tv_1 - Tv_2\|_D = \|P_{\pi,\gamma}(v_1 - v_2)\|_D \leq \|P_{\pi,\gamma}\|_D \|v_1 - v_2\|_D$$
Contraction properties

\[ s_D = \| P_{\pi, \gamma} \|_D \]

- Smaller \( s_D \) corresponds to faster contraction
- Example: constant discount

\[ s_D = \| P_{\pi, \gamma} \|_D \]
\[ = \gamma_c \| P_{\pi} \|_D \]
\[ = \gamma_c \]
Extending previous results
Extending previous results

• LSTD convergence rates specifically derived for continuing case
  • extends to episodic with this generalization

• Unify seminal bias bounds for TD
  • with more explicit episodic bounds
Extending previous results

• LSTD convergence rates specifically derived for continuing case
  • extends to episodic with this generalization

• Unify seminal bias bounds for TD
  • with more explicit episodic bounds

• **New result**: convergence of ETD for transition-based trace, but not under on-policy weighting
Bias bounds

Continuing:

$$\|Tv_1 - Tv_2\|_D \leq \frac{\gamma_c(1 - \lambda)}{1 - \gamma_c \lambda} \|v_1 - v_2\|_D$$

SSP (episodic):

Exists contraction constant $s < 1$
Bias bounds

$$\|Tv_1 - Tv_2\|_D \leq s_D \|v_1 - v_2\|_D$$

$$s_D = \|P_{\pi,\gamma}\|_D$$

If policy reaches a transition where discount less than 1 guaranteed to have $$s_D < 1$$
Bias bounds

\[ \|Tv_1 - Tv_2\|_D \leq s_D \|v_1 - v_2\|_D \]

\[ s_D = \|P_{\pi,\gamma}\|_D \]

If policy reaches a transition where discount less than 1 guaranteed to have \( s_D < 1 \)

<table>
<thead>
<tr>
<th></th>
<th>0.989</th>
<th>0.990</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Episodic taxi</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \gamma_c = 0.99 )</td>
<td>0.989</td>
<td>0.990</td>
</tr>
<tr>
<td><strong>1% single path</strong></td>
<td>0.989</td>
<td></td>
</tr>
<tr>
<td><strong>10% single path</strong></td>
<td>0.987</td>
<td></td>
</tr>
<tr>
<td><strong>1% all paths</strong></td>
<td>0.978</td>
<td></td>
</tr>
<tr>
<td><strong>10% all paths</strong></td>
<td>0.898</td>
<td></td>
</tr>
</tbody>
</table>
Bias bounds

\[ \| T v_1 - T v_2 \|_D \leq s_D \| v_1 - v_2 \|_D \]

\[ s_D = \| P_{\pi, \gamma} \|_D \]

If policy reaches a transition where discount less than 1 guaranteed to have \( s_D < 1 \)

<table>
<thead>
<tr>
<th>( \lambda_c )</th>
<th>0.0</th>
<th>0.5</th>
<th>0.9</th>
<th>0.99</th>
<th>0.999</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Episodic taxi</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \gamma_c = 0.99 )</td>
<td>0.989</td>
<td>0.979</td>
<td>0.903</td>
<td>0.483</td>
<td>0.086</td>
</tr>
<tr>
<td>0.990</td>
<td>0.980</td>
<td>0.908</td>
<td>0.497</td>
<td>0.090</td>
<td></td>
</tr>
<tr>
<td><strong>1% single path</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.989</td>
<td>0.978</td>
<td>0.898</td>
<td>0.467</td>
<td>0.086</td>
<td></td>
</tr>
<tr>
<td>0.987</td>
<td>0.975</td>
<td>0.887</td>
<td>0.439</td>
<td>0.086</td>
<td></td>
</tr>
<tr>
<td><strong>1% all paths</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.978</td>
<td>0.956</td>
<td>0.813</td>
<td>0.304</td>
<td>0.042</td>
<td></td>
</tr>
<tr>
<td>0.898</td>
<td>0.815</td>
<td>0.468</td>
<td>0.081</td>
<td>0.009</td>
<td></td>
</tr>
</tbody>
</table>
Generalizing to probabilistic discounts

\[ \Pr(r, \gamma|s, a, s') \]
Generalizing to probabilistic discounts

\[ v_\pi(s) = \sum_{a,s'} \pi(s, a) \Pr(s, a, s') E[r + \gamma v_\pi(s') | s, a, s'] \]

\[ = \sum_{a,s'} \pi(s, a) \Pr(s, a, s') E[r | s, a, s'] \]

\[ + \sum_{a,s'} \pi(s, a) \Pr(s, a, s') E[\gamma | s, a, s'] v_\pi(s') \]

\[ = r_\pi(s) + \sum_{s'} P_{\pi,\gamma}(s, s') v_\pi(s') \]

for \( \gamma(s, a, s') = E[\gamma | s, a, s'] \).
How are all these conclusions affected by function approximation?

• Assumed states; but likely partially observable

• May no longer be able to solve $TV = V$ but can
  • minimize Bellman residual $||TV - V||$
  • get projected fixed point (MSPBE): $\Pi TV = V$

• …
Example: TD

initialize $\theta$ arbitrarily

**loop** over episodes

initialize $e = 0$

initialize $S_0$

**repeat** for each step in the episode

generate $R_{t+1}, S_{t+1}$ for $S_t$

if terminal: $\delta \leftarrow R_{t+1} - \theta^\top \phi(S_t)$

else: $\delta \leftarrow R_{t+1} + \gamma \theta^\top \phi(S_{t+1}) - \theta^\top \phi(S_t)$

$e \leftarrow e + \phi(S_t)$

$\theta \leftarrow \theta + \alpha \delta e$

$e \leftarrow \gamma \lambda e$
Example: TD

initialize $\theta$ arbitrarily

loop over episodes

initialize $e = 0$

initialize $S_0$

repeat for each step in the episode

generate $R_{t+1}, S_{t+1}$ for $S_t$

if terminal: $\delta \leftarrow R_{t+1} - \theta^T \phi(S_t)$

else: $\delta \leftarrow R_{t+1} + \gamma \theta^T \phi(S_{t+1}) - \theta^T \phi(S_t)$

$e \leftarrow e + \phi(S_t)$

$\theta \leftarrow \theta + \alpha \delta e$

$e \leftarrow \gamma \lambda e$
Example: TD

initialize $\theta$ arbitrarily
initialize $e = 0$
initialize $S_0$

repeat until agent done interaction
  generate $R_{t+1}, S_{t+1}$ for $S_t$
  $\delta \leftarrow R_{t+1} + \gamma_{t+1} \theta^\top \phi(S_{t+1}) - \theta^\top \phi(S_t)$
  $e \leftarrow e + \phi(S_t)$
  $\theta \leftarrow \theta + \alpha \delta e$
  $e \leftarrow \gamma_{t+1} \lambda e$

initialize $\theta$ arbitrarily
loop over episodes
  initialize $e = 0$
  initialize $S_0$
  repeat for each step in the episode
    generate $R_{t+1}, S_{t+1}$ for $S_t$
    if terminal: $\delta \leftarrow R_{t+1} - \theta^\top \phi(S_t)$
    else: $\delta \leftarrow R_{t+1} + \gamma \theta^\top \phi(S_{t+1}) - e$
      $e \leftarrow e + \phi(S_t)$
      $\theta \leftarrow \theta + \alpha \delta e$
      $e \leftarrow \gamma \lambda e$
Outline

- Generalization to transition-based discounting
- The theoretical and algorithmic implications
  - Utility of the generalized problem formalism
Additional utility

- Control
  - simplifies specification of subgoals
  - enables soft termination

- Policy evaluation
  - GVF$s and predictive knowledge
Example: taxi domain

**Actions:**
N, E, S, W, Pickup, Drop-off

**States:**
(x, y, passenger location)

Passenger in black square

State: (2, 1, 3)
Location can be (0,1,2,3) or 4 for in taxi
Optimal policy

Can use average reward or continuing formulation

\[ d_\pi v_\pi = d_\pi (r_\pi + P_\pi, \gamma v_\pi) \]
\[ = d_\pi r_\pi + \gamma_c d_\pi P_\pi v_\pi \]
\[ = d_\pi r_\pi + \gamma_c d_\pi v_\pi \]
\[ \Rightarrow d_\pi v_\pi = \frac{1}{1 - \gamma_c} d_\pi r_\pi. \]
Easy specification of subgoals

• Each pick-up and drop-off can be a subtask
  • numerically more stable than a constant discount
  • options easily encoded with this generalization
Easy specification of subgoals

- Each pick-up and drop-off can be a subtask
- Numerically more stable than a constant discount
- Options easily encoded with this generalization

<table>
<thead>
<tr>
<th>Trans-Soft</th>
<th>Total Pickup and Dropoff</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>7.74 ± 0.03</td>
</tr>
<tr>
<td>$\gamma_c = 0.1$</td>
<td>0.00 ± 0.00</td>
</tr>
<tr>
<td>$\gamma_c = 0.3$</td>
<td>0.02 ± 0.01</td>
</tr>
<tr>
<td>$\gamma_c = 0.5$</td>
<td>0.04 ± 0.01</td>
</tr>
<tr>
<td>$\gamma_c = 0.6$</td>
<td>0.03 ± 0.01</td>
</tr>
<tr>
<td>$\gamma_c = 0.7$</td>
<td>7.12 ± 0.03</td>
</tr>
<tr>
<td>$\gamma_c = 0.8$</td>
<td>7.34 ± 0.03</td>
</tr>
<tr>
<td>$\gamma_c = 0.9$</td>
<td>3.52 ± 0.06</td>
</tr>
<tr>
<td>$\gamma_c = 0.99$</td>
<td>0.01 ± 0.01</td>
</tr>
</tbody>
</table>
Benefits of soft termination

- **Soft termination:**
  \[ \gamma(s, a, s') = 0.1 \]

- Some amount of the value after subgoal should be considered important

<table>
<thead>
<tr>
<th>Soft termination</th>
<th>Hard termination</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1.1</td>
<td>-1.4</td>
</tr>
<tr>
<td>-1.2</td>
<td>-1.2</td>
</tr>
<tr>
<td>-1.2</td>
<td>-1.2</td>
</tr>
<tr>
<td>-1.2</td>
<td>-1.2</td>
</tr>
<tr>
<td>-7.7</td>
<td>-8</td>
</tr>
</tbody>
</table>

The taxi domain, where the pickup/drop-off platforms are at (0,0), (0,4), (3,0) and (4,4). The Passenger P is at the source (1) with Pickup action, with maximum eigenvalue less than 1, satisfies this condition and so for some step-size \[ c \].

To see how this generalized Bellman operator modifies the policies learned using hard and soft termination algorithms. Many algorithms can be easily generalized to the transition-based trace parameters in the appendix. We define the expected approximate value function, given by a vector \[ \langle v \rangle \].

Continuing the recursion, we obtain\[ \langle V \rangle = (I - \gamma T)^{-1} \langle r \rangle \].

Successful pickup and drop-off with total reward \[ 8 \].

For state-based return will again be cutoff by the subgoal, not accounting for orientation after picking up the passenger. Consequently, it takes more left turns after pickup, resulting...
How do we use this generality?

• Do not need to use full generality
  • …we know at least two useful settings
• Particularly useful for policy evaluation and predictive knowledge
  • GVFds (Horde), Predictron
  • Predictive representations
Predictive knowledge

observations as cumulants, persistent policies, …
Suggestions for automatically setting the discount

- Parametrize the discount
  - similarly to option-critic

- Variety of constant discounts for different horizons
  - myopic gamma = 0 for one-step predictions

- Decrease discount based on stimuli
  - e.g., sudden drop in stimulus (light)
Learning in compass world
## Learning in compass world

<table>
<thead>
<tr>
<th>True Leap</th>
<th>Agent Observation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Leap Optimal Generation</td>
<td>2746112</td>
</tr>
<tr>
<td></td>
<td>1.000000</td>
</tr>
<tr>
<td>Leap Random Generation</td>
<td>2050002</td>
</tr>
<tr>
<td></td>
<td>0.825600</td>
</tr>
<tr>
<td>Leap without predictions</td>
<td>1511575</td>
</tr>
<tr>
<td></td>
<td>0.533000</td>
</tr>
</tbody>
</table>

**Step:** 2789838  
**Speed:** 0
Learning in compass world

<table>
<thead>
<tr>
<th>True Leap</th>
<th>Agent Observation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Leap Optimal Generation</td>
<td>2746112</td>
</tr>
<tr>
<td></td>
<td>1.000000</td>
</tr>
<tr>
<td>Leap Random Generation</td>
<td>2050002</td>
</tr>
<tr>
<td></td>
<td>0.825600</td>
</tr>
<tr>
<td>Leap without predictions</td>
<td>1511575</td>
</tr>
<tr>
<td></td>
<td>0.533000</td>
</tr>
</tbody>
</table>

Step: 2789838
Speed: 0
Is it really worth it?
Is it really worth it?

- If you’re considering state-based, may as well do transition-based (and avoid the addition of hypothetical states)
Is it really worth it?

- If you’re considering state-based, may as well do transition-based (and avoid the addition of hypothetical states)

- Algorithm implementation simpler for non-expert
  - **modular**: only need if statements in the discount function
  - **abstraction**: our A.I. algorithms should be as agnostic as possible to the problem settings
  - **simplicity**: e.g., computing stationary distributions from P
Is it really worth it?

• If you’re considering state-based, may as well do transition-based (and avoid the addition of hypothetical states)

• Algorithm implementation simpler for non-expert
  • modular: only need if statements in the discount function
  • abstraction: our A.I. algorithms should be as agnostic as possible to the problem settings
  • simplicity: e.g., computing stationary distributions from P

• Our theoretical results should apply to both episodic and continuing problems
Is it really worth it?

• If you’re considering state-based, may as well do transition-based (and avoid the addition of hypothetical states)

• Algorithm implementation simpler for non-expert
  • modular: only need if statements in the discount function
  • abstraction: our A.I. algorithms should be as agnostic as possible to the problem settings
  • simplicity: e.g., computing stationary distributions from $P$

• Our theoretical results should apply to both episodic and continuing problems

• Beneficial mindset shift to lifelong learning
Is it really worth it?

- If you’re considering state-based, may as well do transition-based (and avoid the addition of hypothetical states)

- Algorithm implementation simpler for non-expert
  - **modular**: only need if statements in the discount function
  - **abstraction**: our A.I. algorithms should be as agnostic as possible to the problem settings
  - **simplicity**: e.g., computing stationary distributions from $P$

- Our theoretical results should apply to both episodic and continuing problems

- Beneficial mindset shift to lifelong learning

Thank you!