State Space Abstraction for Reinforcement Learning

Rowan McAllister & Thang Bui

MLG, Cambridge

Nov 6th, 2014
Rowan’s introduction
Types of abstraction [LWL06]

- Abstractions are partitioned based on state-action value functions $Q(s, a)$ and $Q^*(s, a)$
- Loosely categorised into five categories
Model-irrelevance abstraction $\phi_{\text{model}}$

**Definition**

\[ \phi_{\text{model}}(s_1) = \phi_{\text{model}}(s_2) \]
\[ \Rightarrow \begin{cases} R^a_{s_1} = R^a_{s_2} \\ \sum_{s' \in \phi^{-1}_{\text{model}}(x)} P^a_{s_1, s'} = \sum_{s' \in \phi^{-1}_{\text{model}}(x)} P^a_{s_2, s'} \quad \forall x, a \end{cases} \]

In words, for any action $a$, ground states in the same abstract class should have the same reward, and have the same transition probability into a new abstract class.
Model-irrelevance abstraction $\phi_{\text{model}}$

**Definition**

$\phi_{\text{model}}(s_1) = \phi_{\text{model}}(s_2)$

$\implies \left\{ \begin{array}{l}
R^a_{s_1} = R^a_{s_2} \\
\sum_{s' \in \phi^{-1}_{\text{model}}(x)} P^a_{s_1, s'} = \sum_{s' \in \phi^{-1}_{\text{model}}(x)} P^a_{s_2, s'} \quad \forall x, a
\end{array} \right. $

In words, for any action $a$, ground states in the same abstract class should

- have the same reward, and
- have the same transition probability into a new abstract class.
Model-irrelevance abstraction $\phi_{\text{model}}$

**Definition**

$$\phi_{\text{model}}(s_1) = \phi_{\text{model}}(s_2)$$

$$\Rightarrow \begin{cases} R^a_{s_1} = R^a_{s_2} \\ \sum_{s' \in \phi_{\text{model}}^{-1}(x)} P^a_{s_1,s'} = \sum_{s' \in \phi_{\text{model}}^{-1}(x)} P^a_{s_2,s'} \quad \forall x, a \end{cases}$$

In words, for any action $a$, ground states in the same abstract class should
- have the same **reward**, and
- have the same **transition probability** into a new abstract class.
Model-irrelevance abstraction $\phi_{\text{model}}$

**Definition**

$$\phi_{\text{model}}(s_1) = \phi_{\text{model}}(s_2)$$

$$\Rightarrow \begin{cases} R_{s_1}^a = R_{s_2}^a \\ \sum_{s' \in \phi^{-1}_{\text{model}}(x)} P_{s_1,s'}^a = \sum_{s' \in \phi^{-1}_{\text{model}}(x)} P_{s_2,s'}^a \quad \forall x, a \end{cases}$$

In words, for any action $a$, ground states in the same abstract class should

- have the same *reward*, and
- have the same *transition probability* into a new abstract class.

---

![Diagram](image-url)
Definition ($Q^\pi$-irrelevance abstraction)

$$\phi Q^\pi(s_1) = \phi Q^\pi(s_2) \implies Q^\pi(s_1, a) = Q^\pi(s_2, a), \forall \pi, a$$

In words, for any action $a$, ground states in the same abstract class should have the same state-action value function.
### Definition ($Q^\pi$-irrelevance abstraction)

| $\phi Q^\pi(s_1) = \phi Q^\pi(s_2)$ | $\implies$ | $Q^\pi(s_1, a) = Q^\pi(s_2, a), \forall \pi, a$ |

In words, for any action $a$, ground states in the same abstract class should

- have the same **state-action value function**.
Types of abstraction [cont’d]

Definition ($Q^\pi$-irrelevance abstraction)

$$\phi Q^\pi(s_1) = \phi Q^\pi(s_2) \implies Q^\pi(s_1, a) = Q^\pi(s_2, a), \forall \pi, a$$

In words, for any action $a$, ground states in the same abstract class should

- have the same state-action value function.
Types of abstraction [cont’d]

Definition ($Q^\pi$-irrelevance abstraction)

\[
\phi Q^\pi (s_1) = \phi Q^\pi (s_2) \implies Q^\pi (s_1, a) = Q^\pi (s_2, a), \forall \pi, a
\]

In words, for any action $a$, ground states in the same abstract class should
- have the same state-action value function.

![Diagram](image_url)
Types of abstraction [cont’d]

Definition ($Q^*$-irrelevance abstraction)

\[ \phi_{Q^*}(s_1) = \phi_{Q^*}(s_2) \iff Q^*(s_1, a) = Q^*(s_2, a), \forall a \]

In words, for any action \( a \), ground states in the same abstract class should have the same optimal state-action value function, or same optimal policy.
Types of abstraction [cont’d]

Definition ($Q^*$-irrelevance abstraction)

$$\phi_Q^*(s_1) = \phi_Q^*(s_2) \implies Q^*(s_1, a) = Q^*(s_2, a), \forall a$$

In words, for any action $a$, ground states in the same abstract class should

- have the same **optimal state-action value function**, or same **optimal policy**.
Types of abstraction [cont’d]

Definition ($Q^*$-irrelevance abstraction)

\[ \phi_{Q^*}(s_1) = \phi_{Q^*}(s_2) \implies Q^*(s_1, a) = Q^*(s_2, a), \forall a \]

In words, for any action $a$, ground states in the same abstract class should

- have the same optimal state-action value function, or same optimal policy.

```
<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>Y=0</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>Y=1</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

REWARDS

[Q: action-val.]
[π: Policy]
[V: state-val.]
```
Types of abstraction [cont’d]

Definition ($Q^*$-irrelevance abstraction)

\[ \phi_{Q^*}(s_1) = \phi_{Q^*}(s_2) \implies Q^*(s_1, a) = Q^*(s_2, a), \forall a \]

In words, for any action $a$, ground states in the same abstract class should

- have the same optimal state-action value function, or same optimal policy.

![Diagram showing rewards and policies]
Types of abstraction [cont’d]

Definition ($a^*$-irrelevance abstraction)

\[ \phi_{a^*}(s_1) = \phi_{a^*}(s_2) \]
\[ \implies \begin{cases} \exists a^* \mid a^* \text{ is optimal for } s_1 \text{ and } s_2 \\ Q^*(s_1, a^*) = \max_a Q^*(s_1, a) = \max_a Q^*(s_2, a) = Q^*(s_2, a^*) \end{cases} \]

In words, ground states in the same abstract class should have the same optimal action, and have the same optimal state-action value function upon taking optimal action.
Types of abstraction [cont’d]

Definition (\(a^*-\)irrelevance abstraction)

\[
\phi_{a^*}(s_1) = \phi_{a^*}(s_2)
\]

\[
\implies \begin{cases} 
\exists a^* | a^* \text{ is optimal for } s_1 \text{ and } s_2 \\
Q^*(s_1, a^*) = \max_a Q^*(s_1, a) = \max_a Q^*(s_2, a) = Q^*(s_2, a^*)
\end{cases}
\]

In words, ground states in the same abstract class should

- have the same optimal action, and
- have the same optimal state-action value function upon taking optimal action.
**Types of abstraction [cont’d]**

**Definition** \((a^{*-}\text{-irrelevance abstraction})\)

\[ \phi_{a^*}(s_1) = \phi_{a^*}(s_2) \]

\[ \implies \begin{cases} \exists a^* \mid a^* \text{ is optimal for } s_1 \text{ and } s_2 \\ Q^*(s_1, a^*) = \max_a Q^*(s_1, a) = \max_a Q^*(s_2, a) = Q^*(s_2, a^*) \end{cases} \]

In words, ground states in the same abstract class should
- have the same optimal action, and
- have the same optimal state-action value function upon taking optimal action.

---

**Diagram**

- **Rewards**
- **Q: action-val.**
- **π: Policy**
- **V: state-val.**
Types of abstraction [cont’d]

Definition ($a^*$-irrelevance abstraction)

\[
\phi_{a^*}(s_1) = \phi_{a^*}(s_2) \\
\implies \exists a^* \mid a^* \text{ is optimal for } s_1 \text{ and } s_2 \\
Q^*(s_1, a^*) = \max_a Q^*(s_1, a) = \max_a Q^*(s_2, a) = Q^*(s_2, a^*)
\]

In words, ground states in the same abstract class should
- have the same optimal action, and
- have the same optimal state-action value function upon taking optimal action.
And even more abstract ...

**Definition ($\pi^*$-irrelevance abstraction)**

$$\phi_{\pi^*}(s_1) = \phi_{\pi^*}(s_2) \implies \exists a^* \mid a^* \text{ is optimal for } s_1 \text{ and } s_2$$
And even more abstract ...

**Definition ($\pi^*$-irrelevance abstraction)**

\[
\phi_{\pi^*}(s_1) = \phi_{\pi^*}(s_2) \implies \exists a^* \mid a^* \text{ is optimal for } s_1 \text{ and } s_2
\]

In words, ground states in the same abstract class should
- have the same **optimal action**.
And even more abstract ...

**Definition ($\pi^*$-irrelevance abstraction)**

\[ \phi_{\pi^*}(s_1) = \phi_{\pi^*}(s_2) \implies \exists a^* \mid a^* \text{ is optimal for } s_1 \text{ and } s_2 \]

In words, ground states in the same abstract class should
- have the same **optimal action**.

![Diagram](https://example.com/diagram.png)

[Rewards]
[Q: action-val.]
[$\pi$: Policy]
[V: state-val.]
And even more abstract ...

**Definition** \((\pi^*-\text{irrelevance abstraction})\)

\[
\phi_{\pi^*}(s_1) = \phi_{\pi^*}(s_2) \implies \exists a^* \mid a^* \text{ is optimal for } s_1 \text{ and } s_2
\]

In words, ground states in the same abstract class should
- have the same **optimal action**.
And even more abstract ...

**Definition ($\pi^*$-irrelevance abstraction)**

$$\phi_{\pi^*}(s_1) = \phi_{\pi^*}(s_2) \implies \exists a^* \mid a^* \text{ is optimal for } s_1 \text{ and } s_2$$

In words, ground states in the same abstract class should
- have the same **optimal action**.

![Diagram with rewards and states](image-url)
Types of abstraction - An example

(a)

\( s_0 \)

\( \phi_{a^*} \)
Types of abstraction - An example

(a) $\phi_{\alpha^*}$

(b) $\phi_{\pi^*}$
Types of abstraction - A comparison

Questions:

- Same optimal policy in both abstract space and ground space?
- Same optimal state-action value functions?
Types of abstraction - A comparison

Questions:
- Same optimal policy in both abstract space and ground space?
- Same optimal state-action value functions?

<table>
<thead>
<tr>
<th>Type</th>
<th>Same optimal policy</th>
<th>Q-learning optimality</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\phi_{\text{model}}$</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>$\phi Q^\pi$</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>$\phi Q^*$</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>$\phi a^*$</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>$\phi \pi^*$</td>
<td>✗</td>
<td>✗</td>
</tr>
</tbody>
</table>
Types of abstraction - A comparison

Questions:
- Same optimal policy in both abstract space and ground space?
- Same optimal state-action value functions?

<table>
<thead>
<tr>
<th>Type</th>
<th>Same optimal policy</th>
<th>Q-learning optimality</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\phi_{model}$</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>$\phi Q^\pi$</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>$\phi Q^*$</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>$\phi a^*$</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>$\phi \pi^*$</td>
<td>✗</td>
<td>✓</td>
</tr>
</tbody>
</table>

Ground optimal policy: 0 + 2
Abstract optimal policy: 0.5 + 1
Predictive State Representation: An introduction
A Zoubin cube for HMM/POMDP/PSR ...
A Zoubin cube for HMM/POMDP/PSR ...
Predictive State Representation - Overview

- A model for controlled dynamical systems
Predictive State Representation - Overview

- A model for controlled dynamical systems
- Why useful?
Predictive State Representation - Overview

- A model for controlled dynamical systems
- Why useful?
  - A compact representation compared to \( k \)-Markov or POMDP models
Predictive State Representation - Overview

- A model for controlled dynamical systems
- Why useful?
  - A compact representation compared to $k$-Markov or POMDP models

Goal: X
Environmental states: 17 squares x 4 orientations = 68
Actions: 3 [advance, turn left/right]
Observations: 2 [wall/no-wall]
Predictive State Representation - Overview

- A model for controlled dynamical systems
- Why useful?
  - A compact representation compared to $k$-Markov or POMDP models

Goal: X
Environmental states: 17 squares x 4 orientations = 68
Actions: 3 [advance, turn left/right]
Observations: 2 [wall/no-wall]

Predictive representation: 17
A model for controlled dynamical systems

Why useful?

▶ A compact representation compared to $k$-Markov or POMDP models

Goal: X
Environmental states: 17 squares x 4 orientations = 68
Actions: 3 [advance, turn left/right]
Observations: 2 [wall/no-wall]

Predictive representation: 17

▶ Speedy learning
A review of POMDPs

A POMDP is defined by a tuple \( \langle S, A, O, T, O, b_0 \rangle \):

- \( S \): set of hidden states, \( \dim(S) = n \)
- \( A \): set of actions
- \( O \): set of observations
- \( T \): set of transition matrices, one for each action \( a \in A \)
- \( O_a \): set of diagonal matrices, one for each action-observation pair
- \( b_0 \): initial belief state, \( \{ p(S_i | \emptyset) \} \)

POMDPs use a belief state to encode historical information:

Belief state at history \( h \):

\[
b(S | h) = \{ p(S_i | h) \}
\]

After taking an action \( a \) and observing \( o \), the belief state can be updated:

\[
b(S | hao) = \sum_{i} p(S_i | h) T_a O_a b(S | h) T_a O_a 1 \]

PSRs remove hidden states and use probabilities of particular future actions/outcomes to encode historical information.
A review of POMDPs

A POMDP is defined by a tuple $\langle S, A, O, T, O, b_0 \rangle$:

- $S, A, O$: sets of hidden states, actions and observations, $\dim(S) = n$
- $T$: set of $n \times n$ transition matrices, one for each action $a \in A$
- $O$: set of $n \times n$ diagonal matrices, one for each action-observation pair
- $b_0$: initial belief state, $\{p(S_i|\emptyset)\}_{i=1}^n$

POMDPs use a belief state to encode historical information:

Belief state at history $h$:

$$b(S|h) = \{p(S_i|h)\}_{i=1}^n$$

After taking an action $a$ and observing $o$, the belief state can be updated:

$$b(S|ha_0) = b(S|h) T_a O_a b(S|h) T_a O_a 1_n$$

PSRs remove hidden states and use probabilities of particular future actions/outcomes to encode historical information.
A review of POMDPs

- A POMDP is defined by a tuple \( \langle S, A, O, T, O, b_0 \rangle \):
  \( S, A, O \): sets of hidden states, actions and observations, \( \dim(S) = n \)
  \( T \): set of \( n \times n \) transition matrices, one for each action \( a \in A \)
  \( O \): set of \( n \times n \) diagonal matrices, one for each action-observation pair
  \( b_0 \): initial belief state, \( \{ p(S_i|\emptyset) \}_{i=1}^n \)

- POMDPs use a belief state to encode historical information
  Belief state at history \( h \): \( b(S|h) = \{ p(S_i|h) \}_{i=1}^n \)
A review of POMDPs

- A POMDP is defined by a tuple $⟨S, A, O, T, O, b₀⟩$:
  - $S, A, O$: sets of hidden states, actions and observations, $\dim(S) = n$
  - $T$: set of $n \times n$ transition matrices, one for each action $a ∈ A$
  - $O$: set of $n \times n$ diagonal matrices, one for each action-observation pair
  - $b₀$: initial belief state, $\{p(S_i|∅)\}_{i=1}^n$

- POMDPs use a belief state to encode historical information
  - Belief state at history $h$: $b(S|h) = \{p(S_i|h)\}_{i=1}^n$

- After taking an action $a$ and observing $o$, the belief state can be updated:

$$b(S|hao) = \frac{b(S|h)T^aO^{a,o}}{b(S|h)T^aO^{a,o}1_n}$$
A review of POMDPs

- A POMDP is defined by a tuple \( \langle S, A, O, T, O, b_0 \rangle \):
  - \( S, A, O \): sets of hidden states, actions and observations, \( \dim(S) = n \)
  - \( T \): set of \( n \times n \) transition matrices, one for each action \( a \in A \)
  - \( O \): set of \( n \times n \) diagonal matrices, one for each action-observation pair
  - \( b_0 \): initial belief state, \( \{p(S_i|\emptyset)\}_{i=1}^{n} \)

- POMDPs use a belief state to encode historical information
  - Belief state at history \( h \): \( b(S|h) = \{p(S_i|h)\}_{i=1}^{n} \)

- After taking an action \( a \) and observing \( o \), the belief state can be updated:

  \[
  b(S|hao) = \frac{b(S|h)T^aO^{a,o}}{b(S|h)T^aO^{a,o}1_n}
  \]

PSRs remove hidden states and use probabilities of particular future actions/outcomes to encode historical information.
A test or experiment \( t \) is a sequence of actions and observations:

\[
t = a_1 o_1 a_2 o_2 \cdots a_k o_k
\]

Observed history \( h \) of length \( n \):

\[
h = a_1 o_1 a_2 o_2 \cdots a_n o_n
\]

Test prediction given history:

\[
p(t | h) = \text{prob}(o_{n+1} = o_1, \ldots, o_{n+k} = o_k | h, a_{n+1} = a_1, \ldots, a_{n+k} = a_k)
\]

Set of \( m \) core tests \( T \):

\[
T = \{ t^*_1, t^*_2, \ldots, t^*_m \}
\]

Prediction given history:

\[
p(T | h) = \{ p(t^*_1 | h), p(t^*_2 | h), \ldots, p(t^*_m | h) \}
\]
A test or experiment $t$ is a sequence of actions and observations

$$t = a^1 o^1 a^2 o^2 \cdots a^k o^k$$
Predictive State Representation - Notations

- A test or experiment $t$ is a sequence of actions and observations
  \[ t = a^1 o^1 a^2 o^2 \cdots a^k o^k \]

- Observed history $h$ of length $n$
  \[ h = a_1 o_1 a_2 o_2 \cdots a_n o_n \]
Predictive State Representation - Notations

- A test or experiment $t$ is a sequence of actions and observations
  
  \[ t = a^1 o^1 a^2 o^2 \cdots a^k o^k \]

- Observed history $h$ of length $n$

  \[ h = a_1 o_1 a_2 o_2 \cdots a_n o_n \]

- Test prediction given history

  \[ p(t|h) = \text{prob}(o_{n+1} = o^1, \cdots, o_{n+k} = o^k|h, a_{n+1} = a^1, \cdots, a_{n+k} = a^k) \]
Predictive State Representation - Notations

- A test or experiment $t$ is a sequence of actions and observations
  \[ t = a^1 o^1 a^2 o^2 \cdots a^k o^k \]

- Observed history $h$ of length $n$
  \[ h = a_1 o_1 a_2 o_2 \cdots a_n o_n \]

- Test prediction given history
  \[ p(t|h) = \text{prob}(o_{n+1} = o^1, \ldots, o_{n+k} = o^k|h, a_{n+1} = a^1, \ldots, a_{n+k} = a^k) \]

- Set of $m$ core tests $\mathcal{T}$
  \[ \mathcal{T} = \{ t^*_1, t^*_2, \cdots, t^*_m \} \]
  \[ p(\mathcal{T}|h) = \{ p(t^*_1|h), p(t^*_2|h), \cdots, p(t^*_m|h) \} \]
Predictive State Representation - Definition

Definition ([LSS02])

A set $T$ is a PSR iff its prediction vector forms a sufficient statistic for the dynamical system, or

$$p(t|h) = f_t(p(T|h)),$$

for any test $t$ and history $h$.

Mapping $f_t(.)$ could be linear or non-linear

Linear $f_t(.)$ means for all tests $t$,

$$\exists w_t . t . p(t|h) = p^{\top}(T|h)w_t$$

Updating $p(T|h)$ upon taking action $a$ and observing $o$,

$$p(a_t^*i|hao) = p(ao_t^*i|h) = f_{aot^*i}(p(T|h)) f_{ao}(p(T|h)) = p^{\top}(T|h)w_{aot^*i} p^{\top}(T|h)w_{ao}$$
Definition ([LSS02])

A set $\mathcal{T}$ is a PSR iff its prediction vector forms a sufficient statistic for the dynamical system, or

$$p(t|h) = f_t(p(\mathcal{T}|h)),$$

for any test $t$ and history $h$. 
Predictive State Representation - Definition

Definition ([LSS02])

A set $\mathcal{T}$ is a PSR iff its prediction vector forms a sufficient statistic for the dynamical system, or

$$p(t|h) = f_t(p(\mathcal{T}|h)),$$

for any test $t$ and history $h$.

- Mapping $f_t(.)$ could be linear or non-linear
- Linear $f_t(.)$ means for all tests $t$,

$$\exists w_t \text{ s.t. } p(t|h) = p^T(\mathcal{T}|h)w_t$$
Predictive State Representation - Definition

**Definition ([LSS02])**

A set $\mathcal{T}$ is a PSR iff its prediction vector forms a sufficient statistic for the dynamical system, or

$$p(t|h) = f_t(p(\mathcal{T}|h)),$$

for any test $t$ and history $h$.

- Mapping $f_t(.)$ could be linear or non-linear
- Linear $f_t(.)$ means for all tests $t$,

$$\exists w_t \text{ s.t. } p(t|h) = p^\top(\mathcal{T}|h)w_t$$

- Updating $p(\mathcal{T}|h)$ upon taking action $a$ and observing $o$

$$p(t_i^*|hao) = \frac{p(aot_i^*|h)}{p(ao|h)} = \frac{f_{aot_i^*}(p(\mathcal{T}|h))}{f_{ao}(p(\mathcal{T}|h))} = \frac{p^\top(\mathcal{T}|h)w_{aot_i^*}}{p^\top(\mathcal{T}|h)w_{ao}}$$
Theorem ([LSS02])
For any environment that can be represented by a finite POMDP model, there exists a linear PSR with number of tests $m$ no larger than the number of states in the minimal POMDP model.

POMDP: 5 states

Linear PSR:
$T = \{ r_1, f_0 r_1, f_0 f_0 r_1, f_0 f_0 f_0 r_1 \}$

Nonlinear PSR:
$T = \{ r_1, f_0 r_1 \}$
Predictive State Representation - Why use PSR?

**Theorem ([LSS02])**

For any environment that can be represented by a finite POMDP model, there exists a linear PSR with number of tests $m$ no larger than the number of states in the minimal POMDP model.
Predictive State Representation - Why use PSR?

**Theorem ([LSS02])**

For any environment that can be represented by a finite POMDP model, there exists a linear PSR with number of tests \( m \) no larger than the number of states in the minimal POMDP model.
Theorem ([LSS02])

For any environment that can be represented by a finite POMDP model, there exists a linear PSR with number of tests $m$ no larger than the number of states in the minimal POMDP model.

POMDP: 5 states
Linear PSR: $\mathcal{T} = \{r1, f0r1, f0f0r1, f0f0f0r1, f0f0f0f0r1\}$
Nonlinear PSR: $\mathcal{T} = \{r1, f0r1\}$
System dynamics matrix $\mathcal{D}$ specifies predictions of any test given any history.

$$
\begin{array}{c|cc}
\mathcal{D} & t_1 & \cdots & t_i & \cdots \\
\hline
\emptyset_h & p(t_1|\emptyset_h) & \cdots & p(t_i|\emptyset_h) \\
h_1 & p(t_1|h_1) & \cdots & p(t_i|h_1) \\
\vdots & \vdots & \vphantom{p(t_1|h_1)} & \vdots \\
h_j & p(t_1|h_j) & \cdots & p(t_i|h_j) \\
\vdots & \vdots & \vphantom{p(t_1|h_1)} & \vdots \\
\end{array}
$$
System dynamics matrix $\mathcal{D}$ specifies predictions of any test given any history.

<table>
<thead>
<tr>
<th>$\mathcal{D}$</th>
<th>$t_1$</th>
<th>$\cdots$</th>
<th>$t_i$</th>
<th>$\cdots$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\emptyset_h$</td>
<td>$p(t_1</td>
<td>\emptyset_h)$</td>
<td>$p(t_i</td>
<td>\emptyset_h)$</td>
</tr>
<tr>
<td>$h_1$</td>
<td>$p(t_1</td>
<td>h_1)$</td>
<td>$p(t_i</td>
<td>h_1)$</td>
</tr>
<tr>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$h_j$</td>
<td>$p(t_1</td>
<td>h_j)$</td>
<td>$p(t_i</td>
<td>h_j)$</td>
</tr>
<tr>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$\mathcal{D}$ has an infinite number of columns and an infinite number of rows!
Predictive State Representation - A system dynamics matrix

System dynamics matrix $\mathcal{D}$ specifies predictions of any test given any history.

<table>
<thead>
<tr>
<th>$\mathcal{D}$</th>
<th>$t_1$</th>
<th>$\cdots$</th>
<th>$t_i$</th>
<th>$\cdots$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\emptyset_h$</td>
<td>$p(t_1</td>
<td>\emptyset_h)$</td>
<td>$p(t_i</td>
<td>\emptyset_h)$</td>
</tr>
<tr>
<td>$h_1$</td>
<td>$p(t_1</td>
<td>h_1)$</td>
<td>$p(t_i</td>
<td>h_1)$</td>
</tr>
<tr>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$h_j$</td>
<td>$p(t_1</td>
<td>h_j)$</td>
<td>$p(t_i</td>
<td>h_j)$</td>
</tr>
<tr>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$\mathcal{D}$ has an infinite number of columns and an infinite number of rows! Let $\mathcal{T}$ be the set of linearity independent columns of $\mathcal{D}$, then $\mathcal{T}$ is a linear PSR, as

$$D(:, t) = D(:, \mathcal{T}) w_t \quad \text{or} \quad p(t | h) = p(\mathcal{T} | h) w_t$$
Predictive State Representation - A system dynamics matrix

System dynamics matrix $\mathcal{D}$ specifies predictions of any test given any history.

<table>
<thead>
<tr>
<th>$\mathcal{D}$</th>
<th>$t_1$</th>
<th>$\cdots$</th>
<th>$t_i$</th>
<th>$\cdots$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\emptyset_h$</td>
<td>$p(t_1</td>
<td>\emptyset_h)$</td>
<td>$p(t_i</td>
<td>\emptyset_h)$</td>
</tr>
<tr>
<td>$h_1$</td>
<td>$p(t_1</td>
<td>h_1)$</td>
<td>$p(t_i</td>
<td>h_1)$</td>
</tr>
<tr>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$h_j$</td>
<td>$p(t_1</td>
<td>h_j)$</td>
<td>$p(t_i</td>
<td>h_j)$</td>
</tr>
<tr>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$\mathcal{D}$ has an infinite number of columns and an infinite number of rows!

Let $\mathcal{T}$ be the set of linearly independent columns of $\mathcal{D}$, then $\mathcal{T}$ is a linear PSR, as

$$D(:, t) = D(:, \mathcal{T})w_t \quad \text{or} \quad p(t|h) = p(\mathcal{T}|h)w_t$$

Define $\mathcal{H}$ be the set of linearly independent rows of $\mathcal{D}$, then $\mathcal{H}$ forms a set of core histories
Predictive State Representation - How to discover $\mathcal{T}$ [JS04]

Data: Matrix $D$

Result: $T = \{t^∗_1, t^∗_2, \ldots, t^∗_m\}$

Initialisation:

$v_0 = 0; T = \emptyset; H = \emptyset$

while true do

Form one step extension matrix of $DH, T^1 = \begin{bmatrix} DH, T^1 + 1, T^1 + 1 \end{bmatrix}$

$T \leftarrow$ linearly independent columns of $D^1, T^1$

$H \leftarrow$ linearly independent rows of $D^1, T^1$

$v_i \leftarrow$ rank of $D^1, T^1$

if Stopping condition: $v_i = v_{i-1}$ then stop

end

end

Caveats:

$D$ cannot be computed exactly in general, hence sampling is needed.

$T$ may not be complete, as $v_i = v_{i-1}$ is too strict.
Predictive State Representation - How to discover $T$ [JS04]

**Data:** Matrix $D$

**Result:** $T = \{t_1^*, t_2^*, \cdots, t_m^*\}$

Initialisation: $v_0 = 0; T = \emptyset; H = \emptyset;$

**while** true **do**

Form one step extension matrix of $D_{H,T}$

$$D_{H_1,T_1} = \begin{bmatrix}
D_{H,T} & D_{H,T+1} \\
D_{H+1,T} & D_{H+1,T+1}
\end{bmatrix}$$

$T \leftarrow$ linearly independent columns of $D_{H_1,T_1}$

$H \leftarrow$ linearly independent rows of $D_{H_1,T_1}$

$v_i \leftarrow$ rank of $D_{H_1,T_1}$

**if** Stopping condition: $v_i == v_{i-1}$ **then**

**stop**

**end**

**end**

Caveats:

$D$ cannot be computed exactly in general, hence sampling is needed.

$T$ may not be complete, as $v_i == v_{i-1}$ is too strict.
Predictive State Representation - How to discover $\mathcal{T}$ [JS04]

**Data:** Matrix $\mathcal{D}$

**Result:** $\mathcal{T} = \{t_1^*, t_2^*, \cdots, t_m^*\}$

**Initialisation:** $v_0 = 0; \mathcal{T} = \emptyset; \mathcal{H} = \emptyset;

**while** true **do**

- Form one step extension matrix of $\mathcal{D}_{\mathcal{H},\mathcal{T}}$

\[
\mathcal{D}_{\mathcal{H}_1,\mathcal{T}_1} = \begin{bmatrix}
\mathcal{D}_{\mathcal{H},\mathcal{T}} & \mathcal{D}_{\mathcal{H},\mathcal{T}+1} \\
\mathcal{D}_{\mathcal{H}+1,\mathcal{T}} & \mathcal{D}_{\mathcal{H}+1,\mathcal{T}+1}
\end{bmatrix}
\]

- $\mathcal{T} \leftarrow$ linearly independent columns of $\mathcal{D}_{\mathcal{H}_1,\mathcal{T}_1}$
- $\mathcal{H} \leftarrow$ linearly independent rows of $\mathcal{D}_{\mathcal{H}_1,\mathcal{T}_1}$
- $v_i \leftarrow$ rank of $\mathcal{D}_{\mathcal{H}_1,\mathcal{T}_1}$

**if** Stopping condition: $v_i == v_{i-1}$ **then**

- **stop**

**end**

**end**

**Caveats:**

- $\mathcal{D}$ cannot be computed exactly in general, hence sampling is needed.
- $\mathcal{T}$ may not be complete, as $v_i == v_{i-1}$ is too strict.
Predictive State Representation - How to learn $w_t$ in linear PSRs [JS04]
Predictive State Representation - How to learn $w_t$ in linear PSRs [JS04]

Recall:

$$p(aot^*_i|h) = f_{aot_i^*}(p(T|h)) = p(T|h)w_{aot_i^*}$$

$$p(ao|h) = f_{ao}(p(T|h)) = p(T|h)w_{ao}$$
Predictive State Representation - How to learn $w_t$ in linear PSRs [JS04]

Recall:

$$p(a_{ot}^*|h) = f_{a_{ot}^*}(p(T|h)) = p(T|h)w_{a_{ot}^*}$$
$$p(ao|h) = f_{ao}(p(T|h)) = p(T|h)w_{ao}$$

Note that $w_t$ does not depend on $h$, hence

$$p(a_{ot}^*|H) = p(T|H)w_{a_{ot}^*}$$
$$p(ao|H) = p(T|H)w_{ao}$$
Recall:

\[ p(aot_i^*|h) = f_{aot_i^*}(p(T|h)) = p(T|h)w_{aot_i^*} \]
\[ p(ao|h) = f_{ao}(p(T|h)) = p(T|h)w_{ao} \]

Note that \( w_t \) does not depend on \( h \), hence

\[ p(aot_i^*|H) = p(T|H)w_{aot_i^*} \]
\[ p(ao|H) = p(T|H)w_{ao} \]

As \( \dim(T) = \dim(H) \) and rows in \( p(T|H) \) are linearly independent,

\[ w_{aot_i^*} = p^{-1}(T|H)p(aot_i^*|H) \]
\[ w_{ao} = p^{-1}(T|H)p(ao|H) \]
Predictive State Representation - Planning

PSRs can

- capture the regularities of the environment,
- represent the environment is a more compact way,
- speed up learning process.
Predictive State Representation - Planning

PSRs can
- capture the regularities of the environment,
- represent the environment in a more compact way,
- speed up learning process.

An example from [RRST05]:

1696 states: locations + orientations
840 possible start states
2 observations: wall/no-wall
3 actions: advance, turn left/right
Target G
Average optimal path has 42.2 steps
Predictive State Representation - Learning rate [RRST05]
Predictive State Representation - Learning rate [RRST05]
Predictive State Representation - Learning rate [RRST05]
For continuous observations:

- HMM → LDS
- PSR → Predictive Linear-Gaussian [RS06]

iOOM/iPSR: set of core tests $T$ grows while exploring the environment?
Summary

State space abstraction

- can provide state space generalisations,
- can speed up learning/planning in reinforcement learning,
- is an alternative to value function approximations.

Predictive state representations as an alternative to POMDPs

- model systems using only observable quantities and no hidden states
- are compact, but hard to learn
References


Thanks!