

# Learning Multi-Step Predictive State Representations

Lucas Langer (♣), Borja Balle (♦), Doina Precup (♣)

McGill University (♣), Lancaster University (♦)

July 12, 2016

# Predictive State Representations (PSR)

## Motivation

- 1 Make predictions in partially observable environments
- 2 Learn a representation of hidden states

## Advantages over HMMs

- 1 Statistically consistent and well understood learning efficiency
- 2 Smaller representations of state

[Boots et al., 2011; Hsu et al., 2009; Bailly et al., 2010]

## Why Multi-Step PSRs?

- 1 Leverage structure of underlying environment
- 2 Directly apply transitions which occur at longer time-scales

# PSR: The single observation case

- 1 PSR defined by:  $\langle \alpha_0, \{A_\sigma\}, \alpha_\infty \rangle$ , where  
 $\alpha_0$  is an initial weighting on states ( $1 \times n$ )  
 $A_\sigma$  is a transition matrix ( $n \times n$ )  
 $\alpha_\infty$  is a normalizer ( $n \times 1$ )  
 $n$ : number of latent states
- 2 PSRs compute probabilities of observations  
$$f(\sigma^k) = \alpha_0 \cdot A_\sigma^k \cdot \alpha_\infty$$

[Littman et al., 2001; Singh et al., 2004; Rosencrantz et al., 2004]

# Spectral Learning of PSRs

- 1 Represent data as a Hankel Matrix
- 2 Singular Value Decomposition
- 3 Pick model size
- 4 Linear Algebra
- 5 Result:  $\langle \alpha_0, \{A_\sigma\}, \alpha_\infty \rangle$

[Boots et al., 2011]

# Adding multi-step operators: The Base System

- ① Learn  $\{A_{\sigma^2}, A_{\sigma^4}, A_{\sigma^8}, \dots A_{\sigma^{2N}}\}$  as extra transition operators

Operators learned separately so  $A_{\sigma} \cdot A_{\sigma} \neq A_{\sigma^2}$

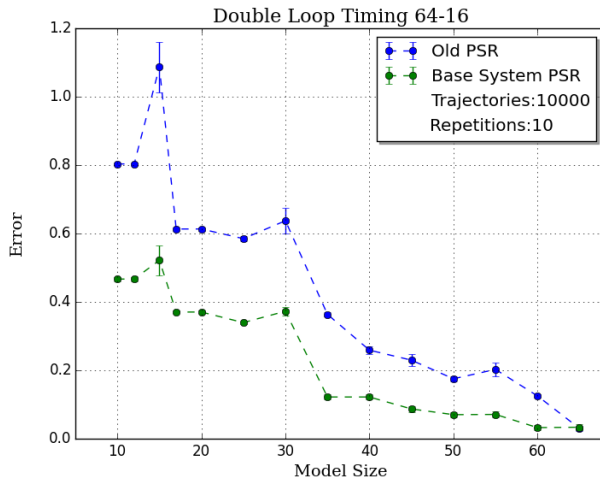
- ②  $f(\sigma^{11}) = \alpha_0 \cdot A_{\sigma^8} \cdot A_{\sigma^2} \cdot A_{\sigma^1} \cdot \alpha_{\infty}$

- ③ **Why might this help?**

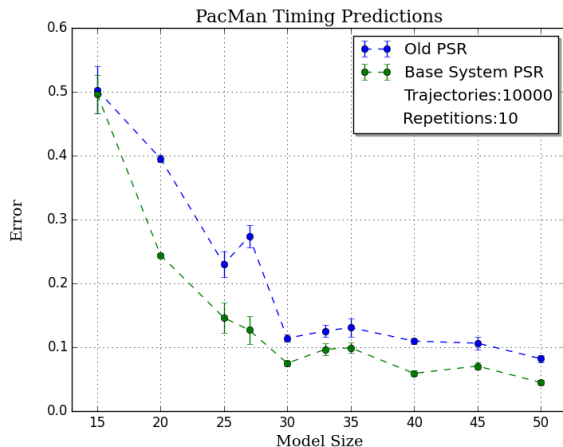
Reduce build up of error

Faster computations

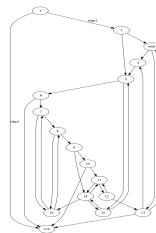
# Base System Performance for Loops



# Pacman-like Labyrinth



(a) Pacman



(b) Graph

# In general, which operators to learn?

- 1 Observations:  $\{a^{30}b^{15}, a^{60}, b^{15}\}$

Desired operators:  $A_{a^{30}b^{15}}, A_{a^{60}}, A_{a^{30}}, A_{b^{15}}, A_a, A_b$

- 2 Sub-string properties: **long, frequent, diverse**

Structured environments should be easier

- 3 Trade-off between number of operators and learning time

Iterative greedy heuristic works well

Could also try an entropy based approach



# How should we execute queries

## 1 Minimize number of matrices

Represent transitions as compactly as possible

## 2 Query string: "abcacb", Operators = $\{A_{ab}, A_{bca}, A_{cb}, A_a, A_b\}$

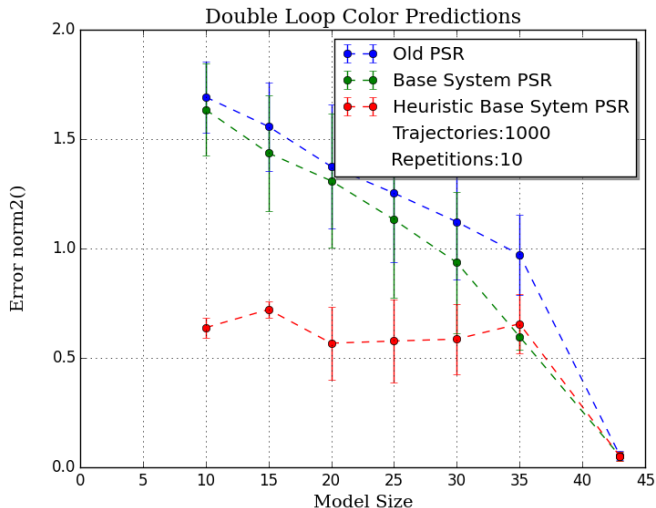
Desired partition: "a—bca—cb"

Computation:  $f(abcacb) = \alpha_0 \cdot A_a \cdot A_{bca} \cdot A_{cb} \cdot \alpha_\infty$

## 3 Solution: dynamic programming

State update for online applications

# Performance of Heuristics



# Questions?