

Spectral Learning for Structured Partially Observable Environments

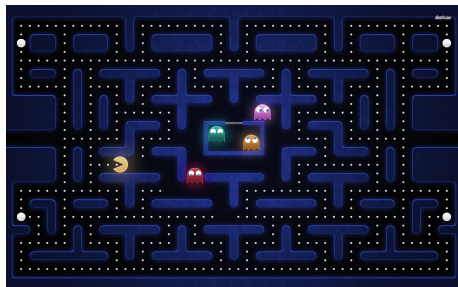
Lucas Langer

Supervisors: Borja Balle, Doina Precup

August 30, 2015

Structured Environments

- ① Goal: Predict observations
- ② Plan: Exploit patterns
- ③ Example: Pacman



PSR: The Timing Case

- Model: Predictive State Representation (PSR)

Timing: one observation symbol: σ

Notation: σ : one time unit, σ^k : k time units

- PSR defined by: $\langle \alpha_0, \{A_\sigma\}, \alpha_\infty \rangle$

α_0 : Initial weighting on states $1 \times n$

A_σ : Transition matrix $n \times n$

α_∞ : Normalizer $n \times 1$

- PSRs compute probabilities of observations

$$f(\sigma^k) = \alpha_0 \cdot A_\sigma^k \cdot \alpha_\infty$$

Ex: HMMs

Spectral Learning of PSRs

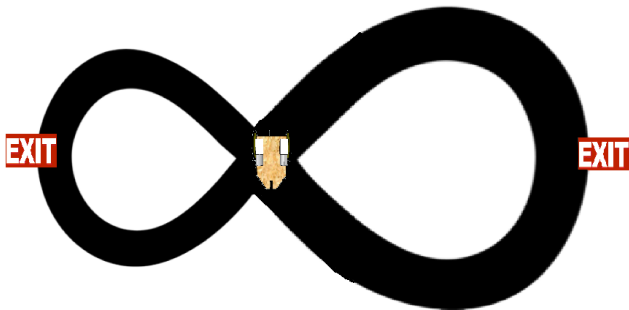
- Step 1: Represent Data as a Hankel Matrix
- Step 2: Spectral: Singular Value Decomposition
- Step 3: Pick Model Size
- Step 4: Learn PSR: $\langle \alpha_0, \{A_\sigma\}, \alpha_\infty \rangle$

The Base System

- Idea: Learn $\{A_\sigma, A_{\sigma^2}, A_{\sigma^4}, A_{\sigma^8}, \dots A_{\sigma^N}\}$ as extra transition operators
- Timing queries: $f(\sigma^{11}) = \alpha_0 \cdot A_{\sigma^8} \cdot A_{\sigma^2} \cdot A_{\sigma^1} \cdot \alpha_\infty$
- Motivation:
 - 1) Express transitions directly
 - 2) Faster queries

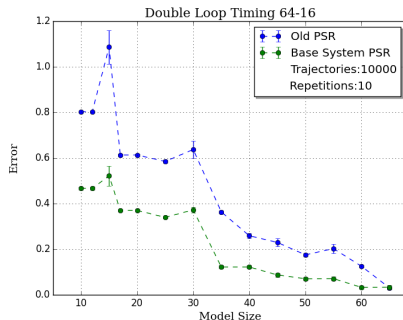
Timing with the Base

Agent drives around loops until leaving through an exit state.

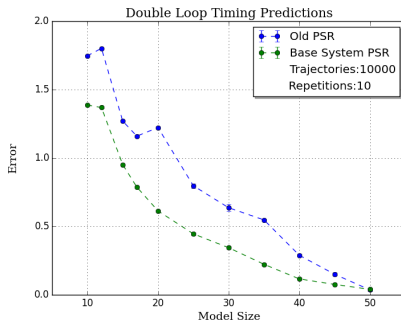


Base System Performance for Loops

64-16 Loop Lengths



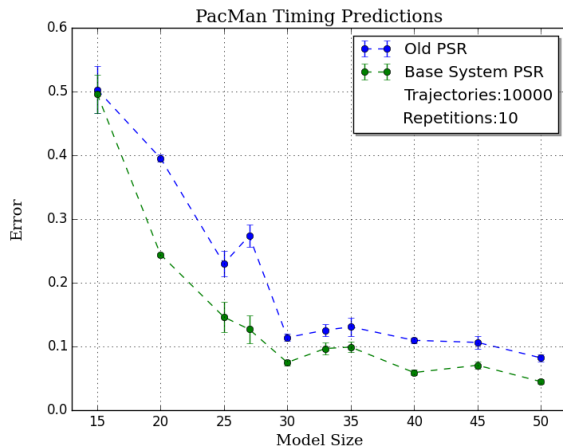
47-27 Loop Lengths



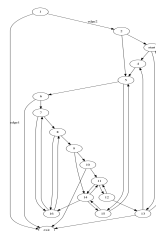
- Base System dominates for smaller models

$$\|f - \hat{f}\| = \sqrt{\sum_{x \in \text{observations}} (f(x) - \hat{f}(x))^2}$$

Pacman Labyrinth



(a) Pacman

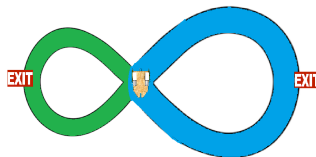
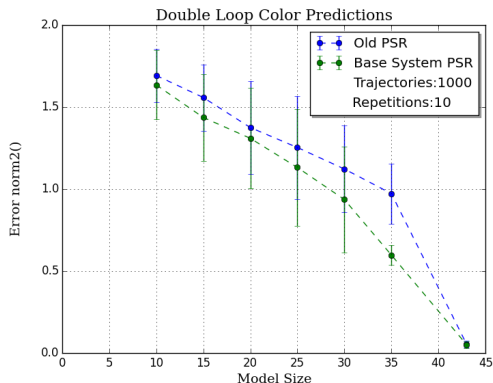


(b) Graph

$$\|f - \hat{f}\| = \sqrt{\sum_{x \in \text{observations}} (f(x) - \hat{f}(x))^2}$$

Wall Color Predictions

We paint the first loop green and the second loop blue



$$\|f - \hat{f}\| = \sqrt{\sum_{x \in \text{observations}} (f(x) - \hat{f}(x))^2}$$

Picking the Base System

- Observations: $\{ "a^{30}":10, "a^{60}":5, "b^{18}":15 \}$

Desired Base System: $A_{a^{30}}, A_{b^{18}}, A_a, A_b$

- Substring properties: **long, frequent, diverse**

Low entropy view of structure

- Solution: iterative greedy heuristic

Computing with the Base System

- How should we execute queries?

Goal: **minimize number of matrices**

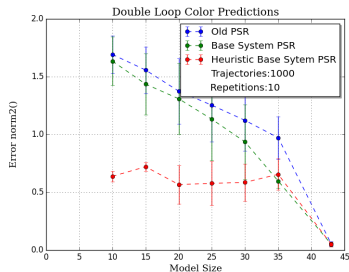
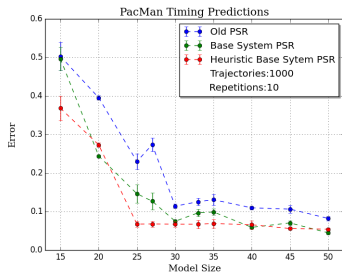
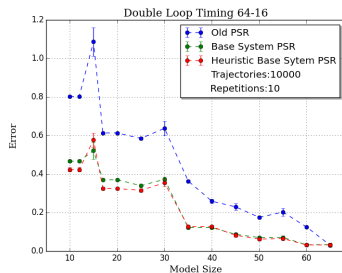
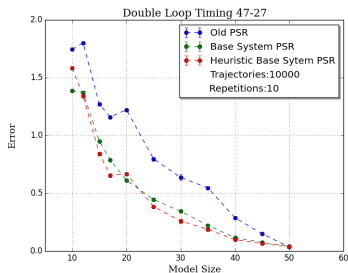
Query string: "abcacb", Base System = $\{A_{ab}, A_{bca}, A_{cb}, A_a, A_b\}$

Desired partition: "a—bca—cb"

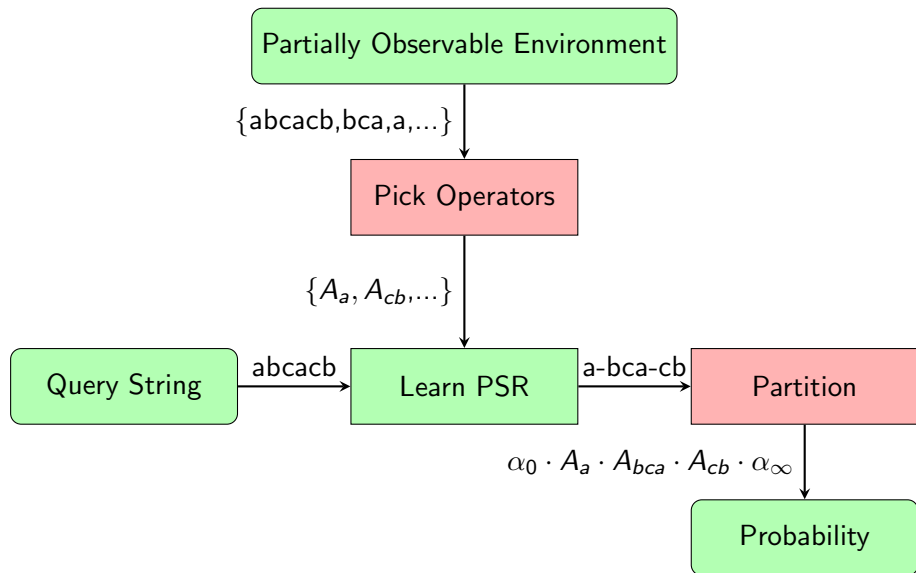
Computation: $f(abcacb) = \alpha_0 \cdot A_a \cdot A_{bca} \cdot A_{cb} \cdot \alpha_\infty$

- Solution: dynamic programming

Performance of Heuristics



The Big Picture



Questions?