Spectral Learning for Structured Partially Observable Environments

Lucas Langer

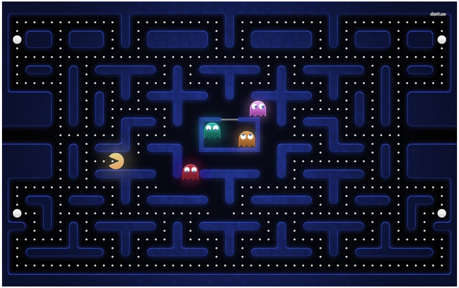
Supervisors: Borja Balle, Doina Precup

August 27, 2015

Structured Partially Observable Environments

Structured Environments

1. Goal: Predict observations
2. Plan: Exploit structure
3. Example: Pacman



PSR: The Timing Case

- Model environment with a Predictive state representation
- For timing we have one observation: 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 $1xn$
  A_σ: Transition matrix nxn
  α_∞: Normalizer $nx1$

- PSRs compute probabilities of observations
  $f(\sigma^k) = \alpha_0 \cdot A_\sigma^k \cdot \alpha_\infty$

Spectral Learning of PSRs

Step 1: Represent data as a Hankel Matrix

Step 2: Singular Value Decomposition

Step 3: Pick Model Size

Step 4: Learn PSR: $\langle \alpha_0, \{A_\sigma\}, \alpha_\infty \rangle$

The Base System

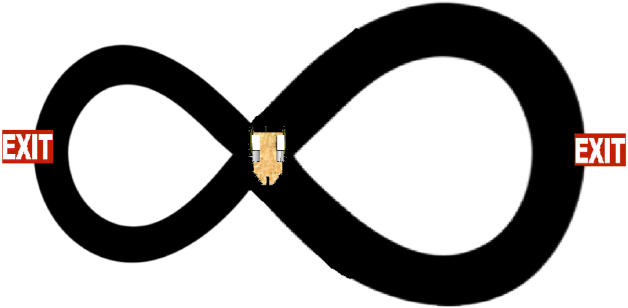
- Idea: Add $\{A_\sigma, A_{\sigma^2}, A_{\sigma^4}, A_{\sigma^8}, ...A_{\sigma^N}\}$ as extra transition operators
- Timing queries: $f(\sigma^{11}) = \alpha_0 \cdot A_{\sigma^8} \cdot A_{\sigma^2} \cdot A_{\sigma^1} \cdot \alpha_\infty$
- Motivation:

  1) Express transitions directly

  2) Faster queries

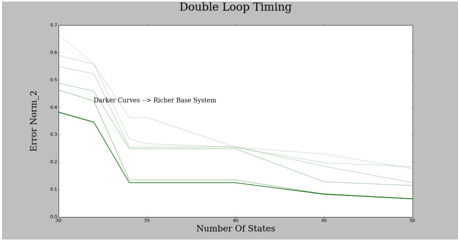
Timing with the Base

Agent goes through loops until leaving through an exit state. Loop lengths are 64 and 16 time units (not to scale).

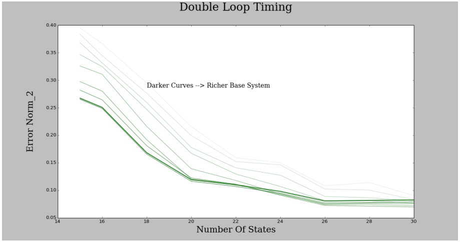


Base System Performance for Loops

No noise in durations



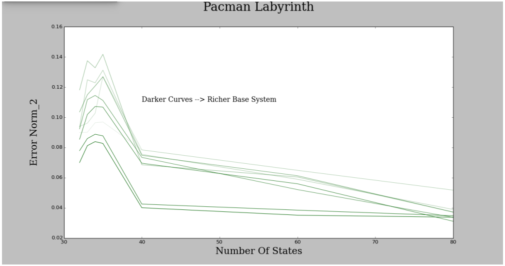
Noise in durations

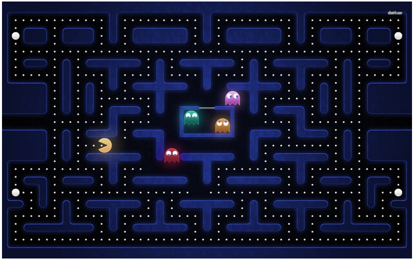


- Base System dominates for smaller models
- Noise allows for smaller models

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

Timing Predictions

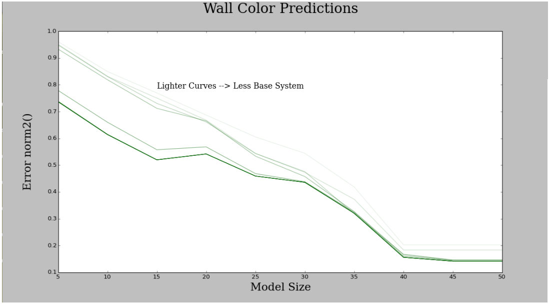


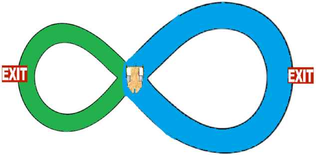


$$||f - \hat{f}|| = \sqrt{\sum_{x \in observations} (f(x) - \hat{f(x)})^2}$$

Wall Color Predictions

We paint the first loop green and the second loop blue





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

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
- Solution: iterative greedy heuristic
- Substring properties: **long**, **frequent**, **diverse**
- Entropy view of structure

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"

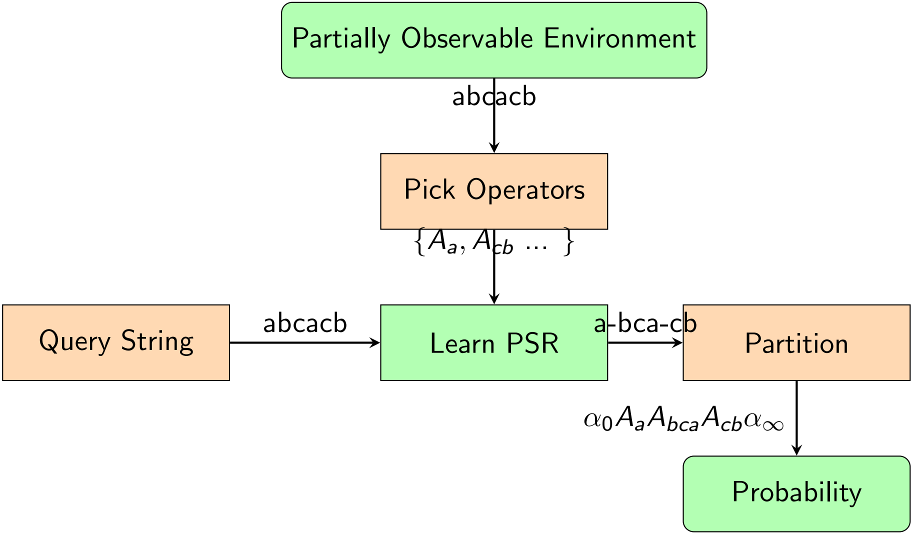
  Computation: $f(abcacb) = \alpha_0 \cdot A_a \cdot A_{bca} \cdot A_{cb} \cdot \alpha_\infty$

- Solution: dynamic programming

Conclusion and Future Work

- What's left for the Base System?
  1) Theoretical analysis
  2) Test heuristics on labyrinths
  3) Further optimize heuristics

The Big Picture



Questions? Comments?