Spectral learning for structured partially observable environments

Lucas Langer

McGill University

lucas.langer@mail.mcgill.ca

August 17, 2015

Overview

Structure partially observable environments

- Goal: Improve predictive performance
- Example: Pacman
- Plan: Represent structure in model

PSRs: The Timing Case

- PSRs defined by $= < \alpha_0, \{A_\sigma\}, \alpha_\infty\} >$

  α_0: initial weighting on n states

  A_σ nxn transition matrix

  α_∞: normalizer n states

  σ represents one time unit, σ^k represents k time units
- PSRs compute probabilities of observations

  $f(\sigma^k) = \alpha_0 * A_\sigma^k * \alpha_\infty$
- PSRs can be used to represent the states of a system

  E.g: $\alpha_0 * A_\sigma^k$

Overview of Learning

- Spectral algorithms can learn PSRs from Data
- Flavour of Algorithm:

  Step 1: Represent Data as a matrix

  Step 2: Singular value decomposition

  Step 3: Pick Number of States for PSR

  Step 4: Learn the PSR with matrix computations

The Base System

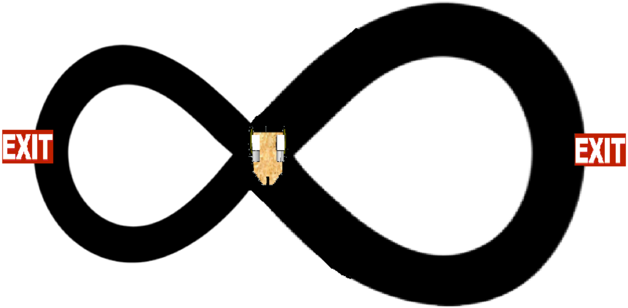
- Number representations: $11 = 2^3 + 2^1 + 2^0$
- Timing queries $f(a^11) = \alpha * A_(a^8) * A_(a^2) * A_(a^1)$
- Motivation:

  1) Express transitions directly to avoid error build up

  2) Faster queries. Discussion $\alpha_0 * (A_\sigma)^k$

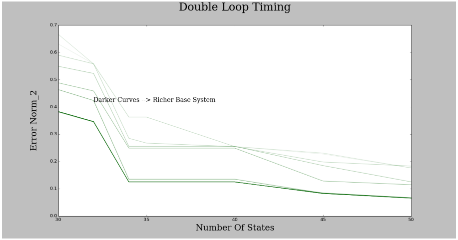
Timing with the Base

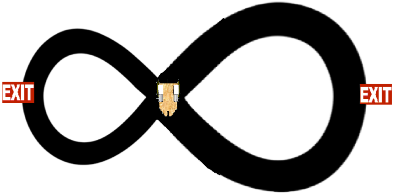
Agent goes through loops until leaving through an exit state. Exit states have transition probabilities of 0.4 and 0.6. Loop lengths are 64 and 16.



Double Loop Results

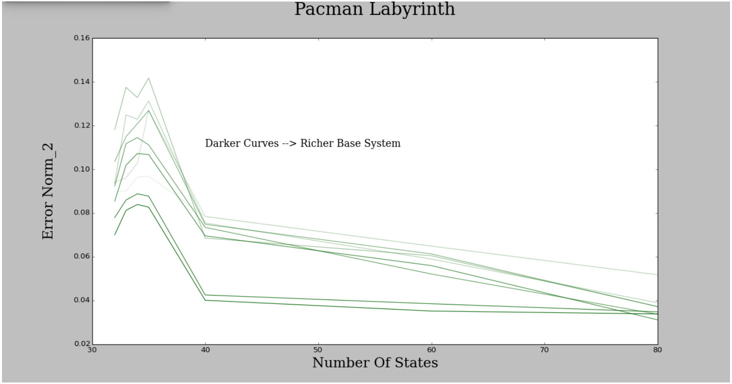
$$||f_A - f_A Bar|| = (\sum(f_A(x) - f_A Bar)^2)^(0.5)$$





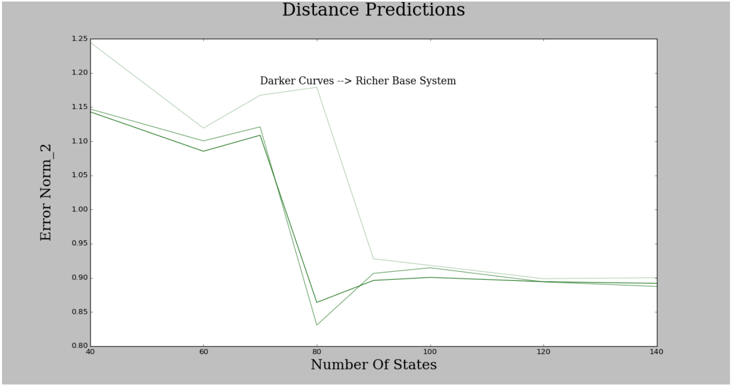
PacMan Labyrinth Results

$$||f_A - f_A Bar|| = (\sum(f_A(x) - f_A Bar)^2)^(0.5)$$

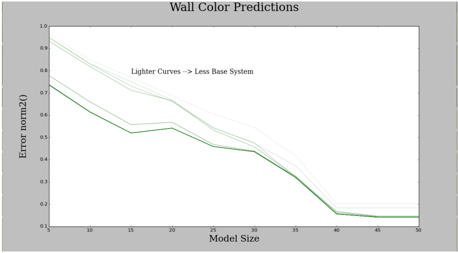


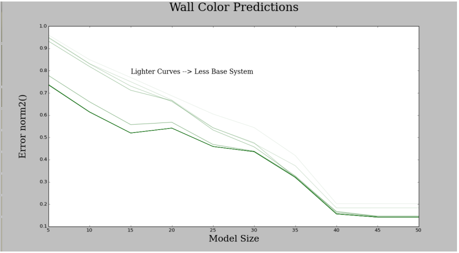
Distance Predictions

We use $\alpha_0 * A_\sigma^k$ as a representation of state. Linear regression gives us a distance weighting on states.



We paint the first loop green and the second loop red.





Picking the Base System

-
- One wants **long** and frequent sub-strings
- Want to make sure Base System is **diverse**

  I.e it needs to cover different types of common sub-strings
- Solution: iterative greedy heuristic

Computing with the Base System

- Input String: "abcacb", Base System $= \{A_a b, A_b ca, A_c b\}$
- Computing with Base is a string partitioning problem
- Desired Partition: "a—bca—cb""
- General Goal: Minimize number of partitions
- Solution: Dynamic programming

Conclusion and Future Work

- There is still work to be done on PSR learning!
- What's left for the Base System?

  1) Theoretical analysis

  2) Further optimize construction

Questions? Comments?