Spectral learning for structured partially observable environments

Lucas Langer

McGill University

lucas.langer@mail.mcgill.ca

August 17, 2015

Overview

The Timing Case

- For the timing case $\sum = \{\sigma\}$
- An observation of duration k is denoted by σ^k
- WFA will be $= < \alpha_0, \{A_\sigma\}, \alpha_\infty\} >$
- $f_A(\sigma^k) = \alpha_0 * A_\sigma^k * \alpha_\infty$
- Blackboard: A spectral learning algorithm for WFA

The Base System

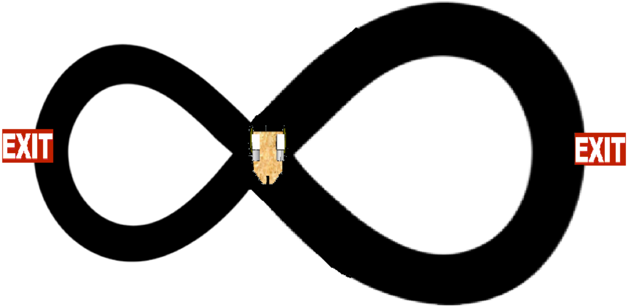
- Number representations:
  $39 = 1 * 2^5 + 0 * 2^4 + 0 * 2^3 + 1 * 2^2 + 1 * 2^1 + 1 * 2^0$
- Timing queries $f(a^3 9) = \alpha * A_a^3 2 * A_a^4 * A_a^2 * A_a^1$
- Motivation:
- 1) Express transitions directly to avoid error build up
- 2) Faster queries. Discussion $\alpha_0 * (A_\sigma)^k$

The Base System Cont.

- When taking a reduced model compounding errors are a threat
- Analogy to rounding: Round(51.63*34.12) v.s Round(51.63) * Round(34.12)
- Let π: n states –¿ k states be the projection operator from a system with n states to the k-best states
- $f_Base(\sigma^128) = (\pi * \alpha_0) * (\pi * A_\sigma^128) * (\pi * \alpha_\infty)$
  $f_Naive(x) = (\pi * \alpha_0) * (\pi * A_\sigma^128) * (\pi * \alpha_\infty)$

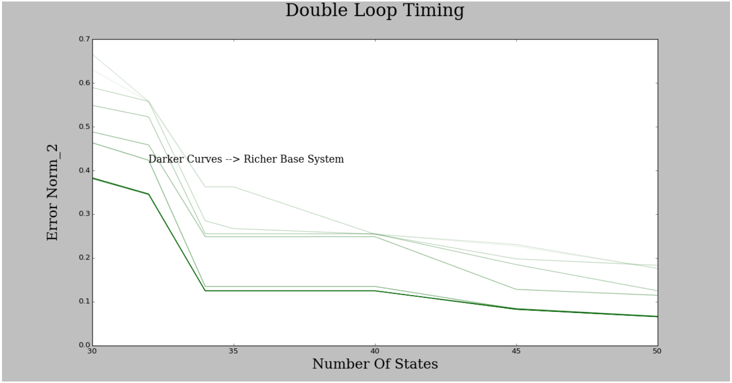
Timing with the Base

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.



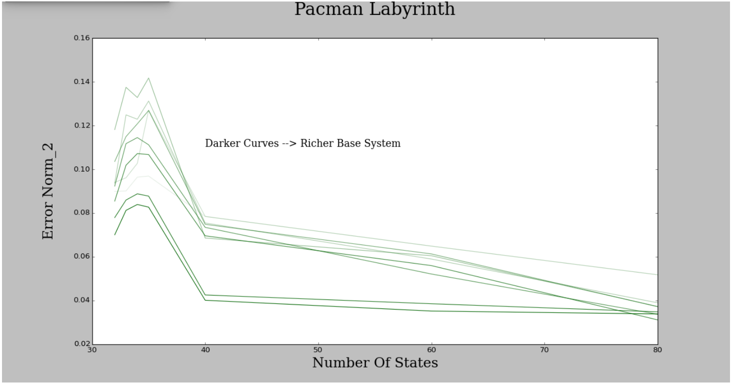
Double Loop Results

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



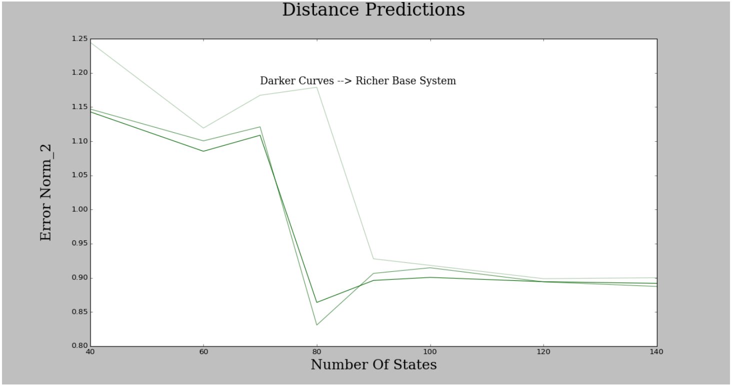
PacMan Labyrinth Results

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



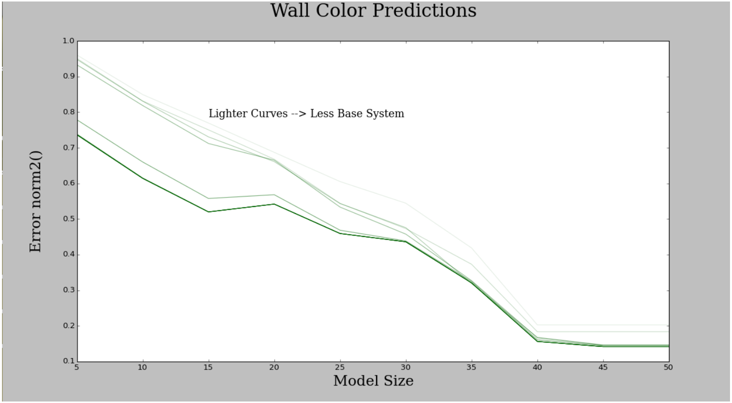
Distance Predictions

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



Wall Color Predictions

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



Picking the Base System

- In general, one wants **long** and frequent sub-strings
- Want to make sure Base System is **diverse**
- Solution: Greedy heuristic

Picking the Base System Cont.

- **Algorithm:**
- Pick sequence $x = x_1 x_2 \ldots x_n$ whose operator reduces matrix products the most. This step depends on the current Base System!
- Update Base System by learning A_x
- Example:
- Input Data: $\{a^20, a^30, a^50, b^5, b^10\}$
- Initial Base: $\{A_a, A_b\}$
- Iteration 1: A_a^10 Added
- Iteration 2: A_b^10 Added, NOT A_a^5

Computing with the Base System

- Goal of Heuristic: minimize number of matrices in query
- Solution: Dynamic programming
- Example:

Questions? Comments?