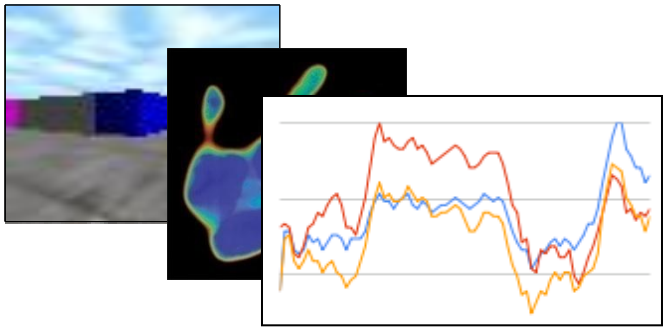
Hierarchical Bayesian Methods for Reinforcement Learning

David Wingate
wingated@mit.edu

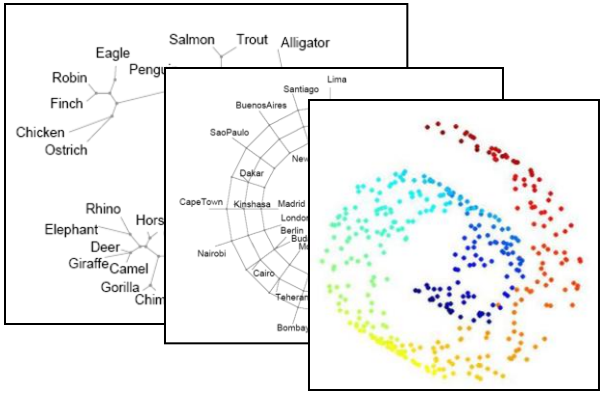
Joint work with Noah Goodman, Dan Roy,
Leslie Kaelbling and Joshua Tenenbaum

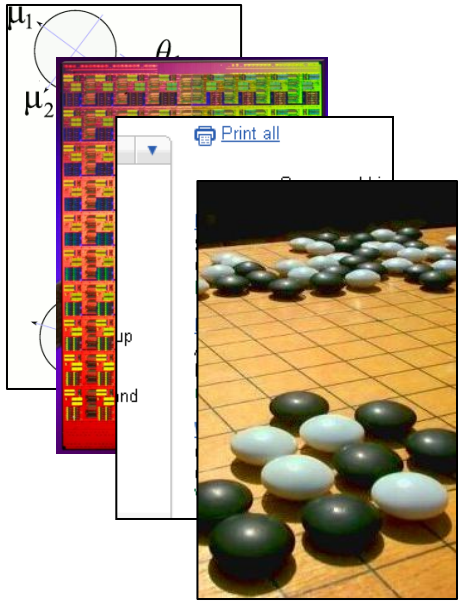
My Research: Agents

Rich sensory data



Structured prior knowledge



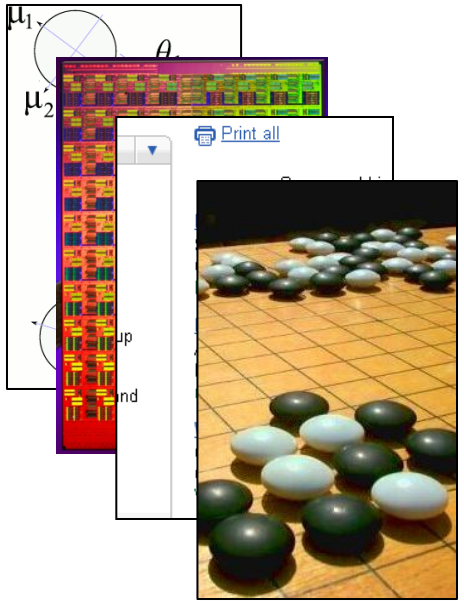


Reasonable abstract behavior

Problems an Agent Faces

Problems:

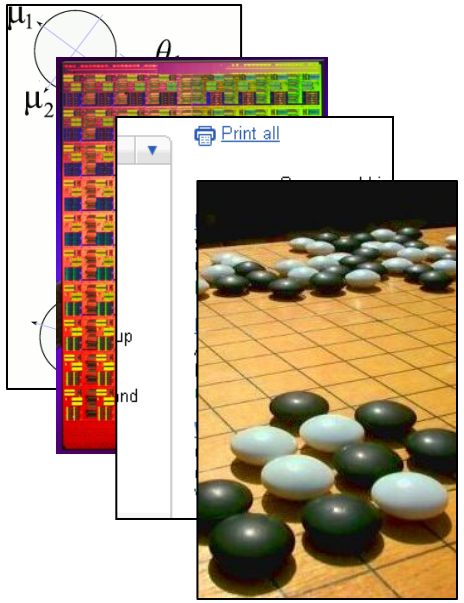
State estimation
Perception
Generalization
Planning
Model building
Knowledge representation
Improving with experience
…



My Research Focus

Problems:

State estimation
Perception
Generalization
Planning
Model building
Knowledge representation
Improving with experience
…



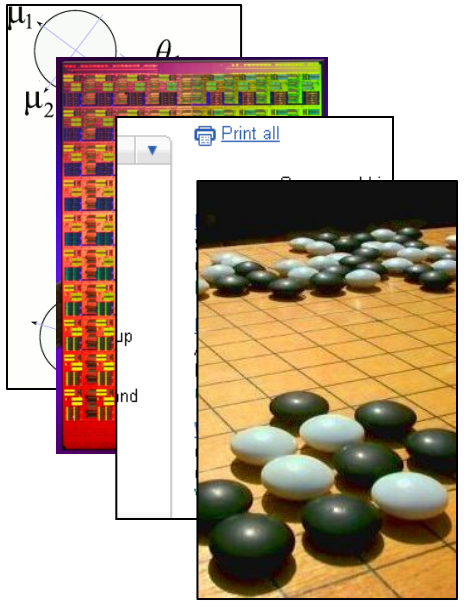
Tools:

Hierarchical Bayesian Models
Reinforcement Learning

Today's Talk

Problems:

State estimation
Perception
Generalization
Planning
Model building
Knowledge representation
Improving with experience
…



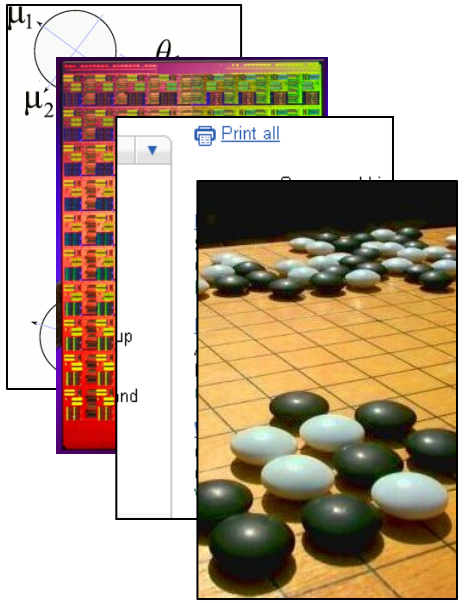
Tools:

Hierarchical Bayesian Models
Reinforcement Learning

Today's Talk

Problems:

State estimation
Perception
Generalization
Planning
Model building
Knowledge representation
Improving with experience
…



Tools:

Hierarchical Bayesian Models
Reinforcement Learning

Outline

- Intro: Bayesian Reinforcement Learning
- Planning: Policy Priors for Policy Search
- Model building: The Infinite Latent Events Model
- Conclusions

Bayesian Reinforcement Learning

What is Bayesian Modeling?

Find **structure in data**

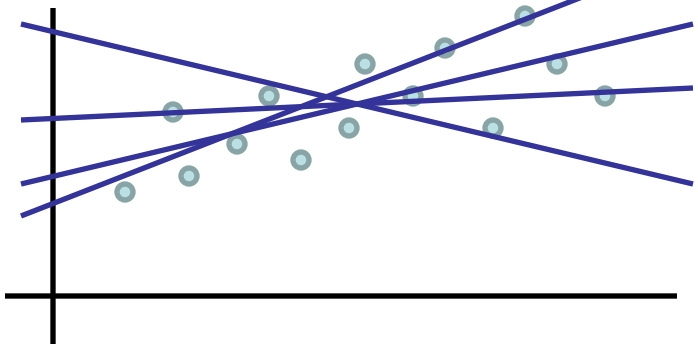
while **dealing explicitly with uncertainty**

The goal of a Bayesian is to reason about
the distribution of structure in data

$$p(\text{something interesting}|\text{data})$$

Example

What line **generated** this data?



That one?

This one?

What about this one?

Probably not this one

$$p(\text{something interesting}|\text{data})$$

What About the “Bayes” Part?

Bayes Law is a mathematical fact that helps us

$$p(\text{something interesting}|\text{data}) \propto$$

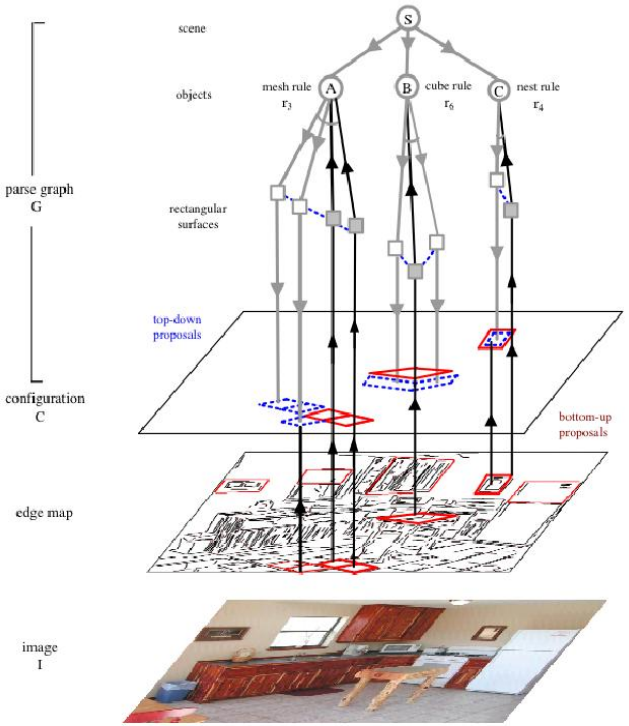
$$p(\text{data}|\text{something interesting}) \quad p(\text{something interesting})$$

Likelihood **Prior**

Distributions Over Structure

Visual perception
Natural language
Speech recognition
Topic understanding
Word learning
Causal relationships
Modeling relationships
Intuitive theories
…

Distributions Over Structure



Han, F and Zhu, S.C. Bottom-up/Top-down Image Parsing with Attribute Graph Grammar. IEEE Trans. on Pattern Analysis and Machine Intelligence (To appear).

Visual perception

Natural language

Speech recognition

Topic understanding

Word learning

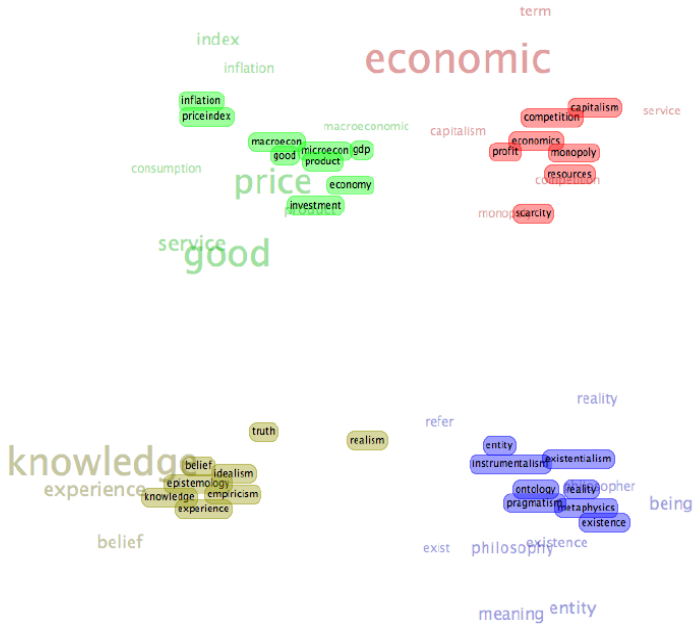
Causal relationships

Modeling relationships

Intuitive theories

…

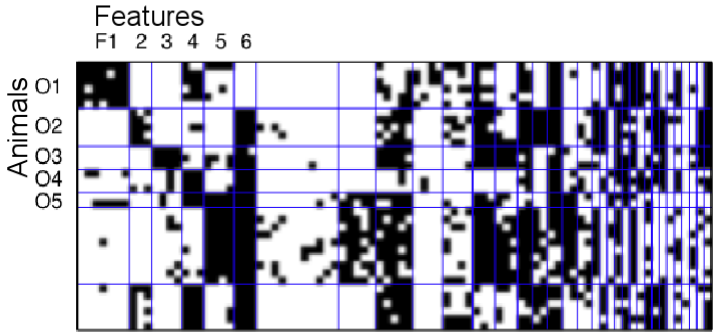
Distributions Over Structure



Model: D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent Dirichlet allocation. Journal of Machine Learning Research, 3, March 2003.
Implementation: Beau Cronin (MIT BCS, Navia Systems, Inc)

Visual perception
Natural language
Speech recognition
Topic understanding
Word learning
Causal relationships
Modeling relationships
Intuitive theories
…

Distributions Over Structure



O1 killer whale, blue whale, humpback, seal, walrus, dolphin
O2 antelope, horse, giraffe, zebra, deer
O3 monkey, gorilla, chimp
O4 hippo, elephant, rhino
O5 grizzly bear, polar bear

F1 flippers, strain teeth, swims, arctic, coastal, ocean, water
F2 hooves, long neck, horns
F3 hands, bipedal, jungle, tree
F4 bulbous body shape, slow, inactive
F5 meat teeth, eats meat, hunter, fierce
F6 walks, quadrapedal, ground

Model: Charles Kemp, Joshua B. Tenenbaum, Thomas L. Griffiths, Takeshi Yamada, and Naonori Ueda. Learning systems of concepts with an infinite relational model. AAAI, 2006.

Visual perception
Natural language
Speech recognition
Topic understanding
Word learning
Causal relationships
Modeling relationships
Intuitive theories
…

Inference

So, we've defined these distributions mathematically.
What can we do with them?

- Some questions we can ask:
  - Compute an expected value
  - Find the MAP value
  - Compute the marginal likelihood
  - Draw a sample from the distribution

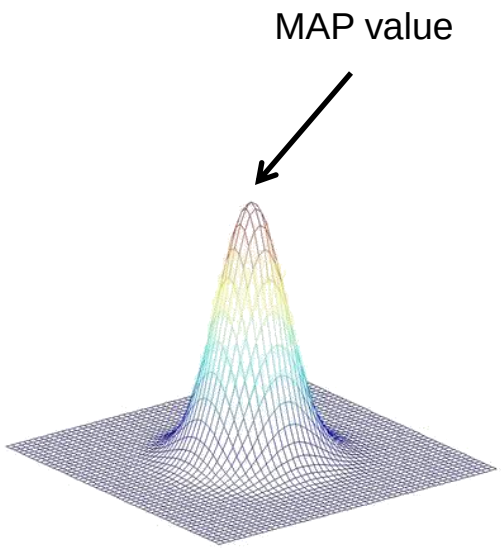
- All of these are computationally hard

Inference

So, we've defined these distributions mathematically.

What can we do with them?

- Some questions we can ask:
  - Compute an expected value
  - **Find the MAP value**
  - Compute the marginal likelihood
  - Draw a sample from the distribution
- All of these are computationally hard



Reinforcement Learning

RL = **learning meets planning**

Reinforcement Learning

RL = **learning meets planning**

Logistics and scheduling
Acrobatic helicopters
Load balancing
Robot soccer
Bipedal locomotion
Dialogue systems
Game playing
Power grid control
…

Reinforcement Learning

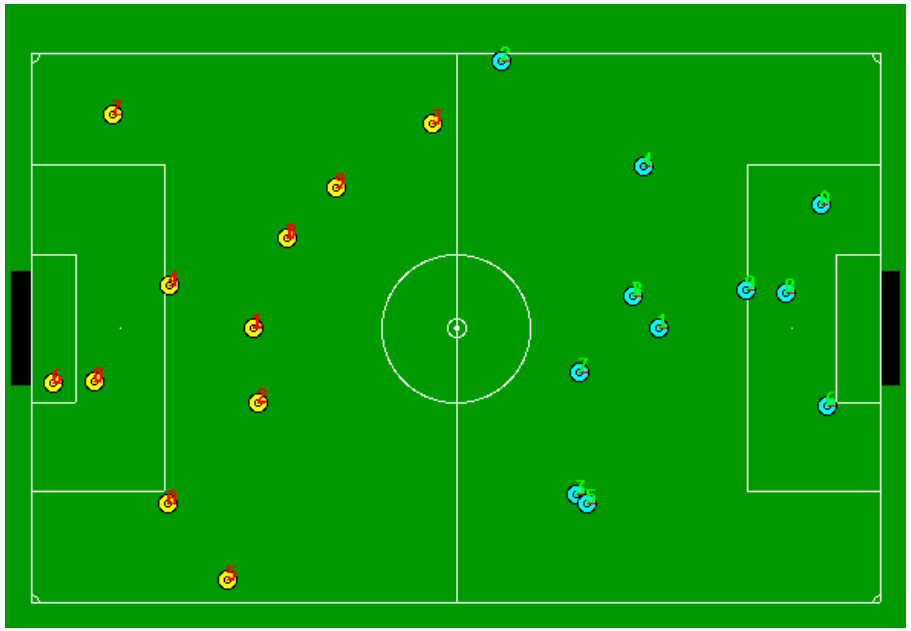
RL = **learning meets planning**

Logistics and scheduling
Acrobatic helicopters
Load balancing
Robot soccer
Bipedal locomotion
Dialogue systems
Game playing
Power grid control
…

Model: Pieter Abbeel. Apprenticeship Learning and Reinforcement Learning with Application to Robotic Control. PhD Thesis, 2008.

Reinforcement Learning

RL = **learning meets planning**



Logistics and scheduling
Acrobatic helicopters
Load balancing
Robot soccer
Bipedal locomotion
Dialogue systems
Game playing
Power grid control
…

Model: Peter Stone, Richard Sutton, Gregory Kuhlmann. Reinforcement Learning for RoboCup Soccer Keepaway. Adaptive Behavior, Vol. 13, No. 3, 2005

Reinforcement Learning

RL = **learning meets planning**

Logistics and scheduling
Acrobatic helicopters
Load balancing
Robot soccer
Bipedal locomotion
Dialogue systems
Game playing
Power grid control
…

Model: David Silver, Richard Sutton and Martin Muller. Sample-based learning and search with permanent and transient memories. ICML 2008

Bayesian RL

Use Hierarchical Bayesian methods to

learn a rich model of the world

while using planning to

figure out what to do with it

Outline

- Intro: Bayesian Reinforcement Learning
- Planning: Policy Priors for Policy Search
- Model building: The Infinite Latent Events Model
- Conclusions

Bayesian Policy Search

Joint work with Noah Goodman, Dan Roy
Leslie Kaelbling and Joshua Tenenbaum

Search

Search is important for AI / ML (and CS!) in general

Combinatorial optimization, path planning, probabilistic inference…

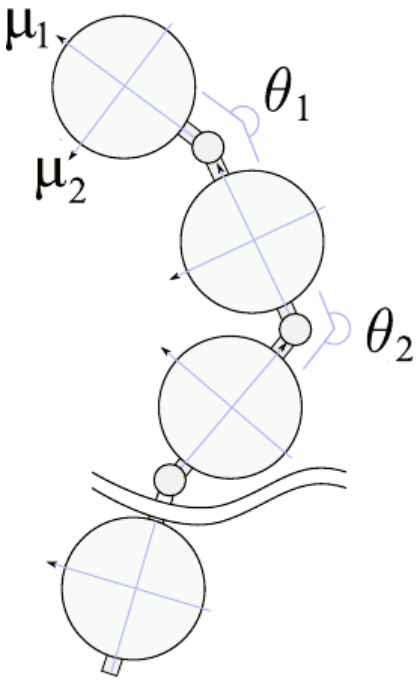
Often, it's important to have the **right search bias**

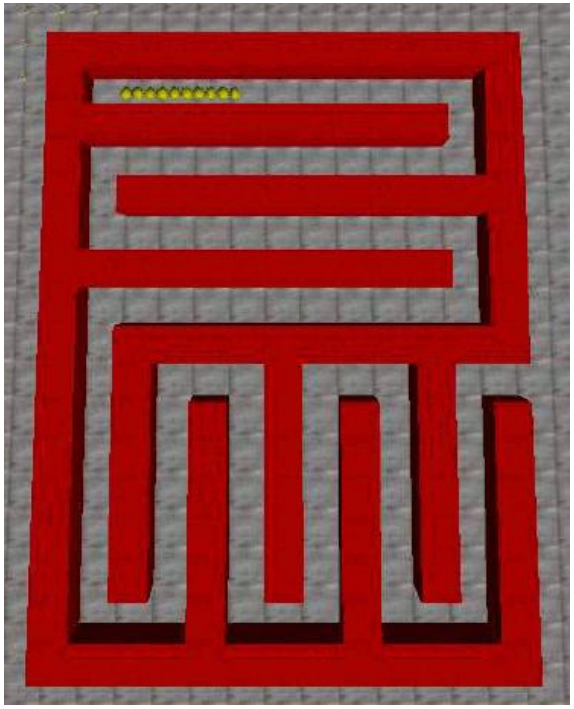
Examples: **heuristics, compositionality, parameter tying, …**

But what if we
don't know the search bias?

Let's **learn it.**

Snake in a (planar) Maze





10 segments
9D continuous action
Anisotropic friction
State: ~40D
Deterministic

Observations:
walls around head

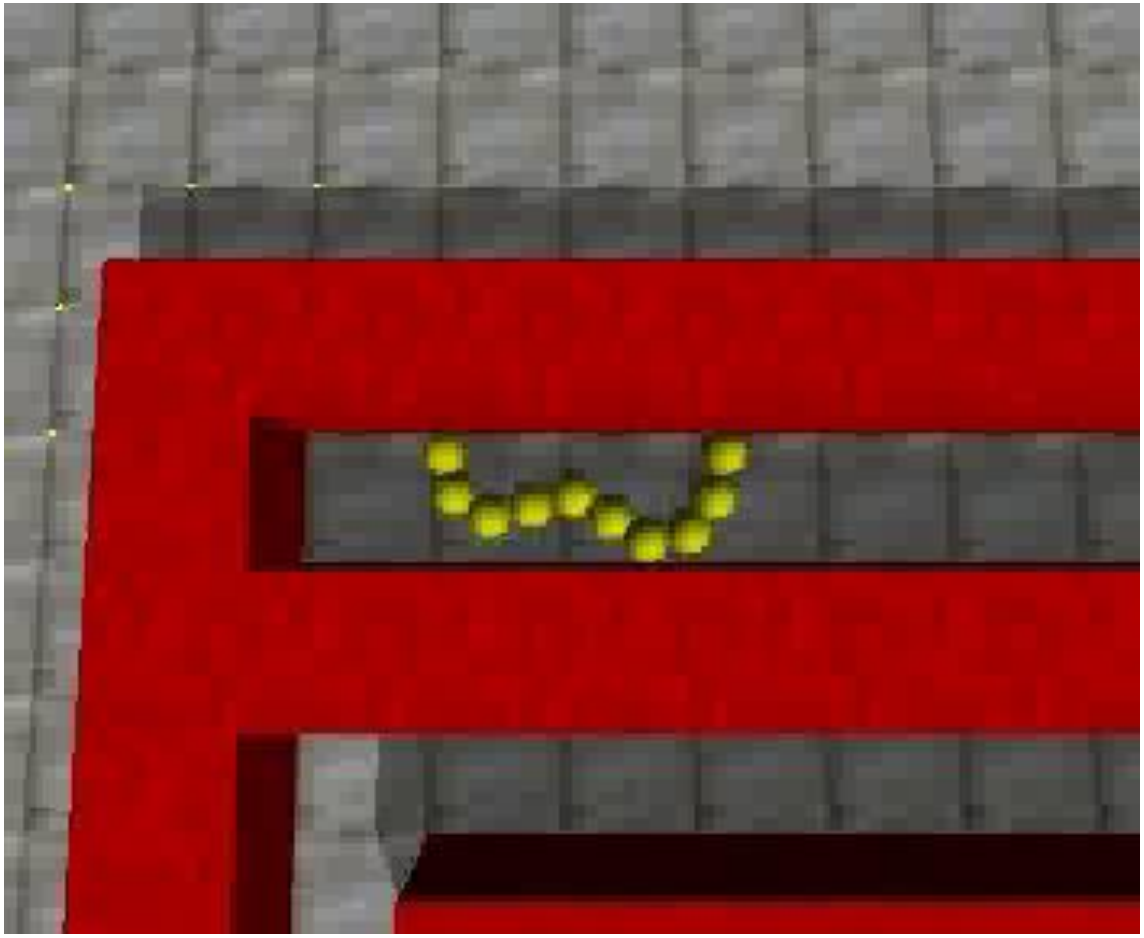
Goal: find a **trajectory**
(sequence of 500 actions) through the track

Snake in a (planar) Maze

This is a search problem.

But it's a **hard space to search.**

Human* in a Maze



* Yes, it's me.

Domain Adaptive Search

How do you **find good trajectories** in hard-to-search spaces?

One answer:
As you search, **learn more than just the trajectory.**

Spend some time navel gazing.
Look for **patterns in the trajectory**, and use those patterns to improve your overall search.

Bayesian Trajectory Optimization

$$p(\text{actions}|\text{goal}) \propto p(\text{goal}|\text{actions})\ p(\text{actions})$$

Posterior

This is what we want to optimize!

Likelihood

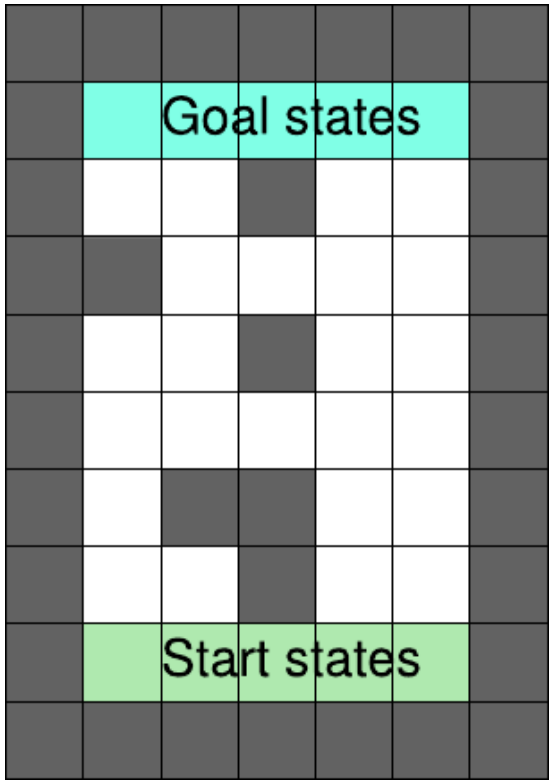
We'll use "distance along the maze"

Prior

Allows us to incorporate knowledge

This is a **MAP inference problem.**

Example: Grid World



Objective:
for each state,
determine
the optimal action

(one of **N, S, E, W**)

The mapping from state
to action is called a
policy

Key Insight

In a stochastic hill climbing inference algorithm,
the action prior can structure the proposal kernels, which structures the search

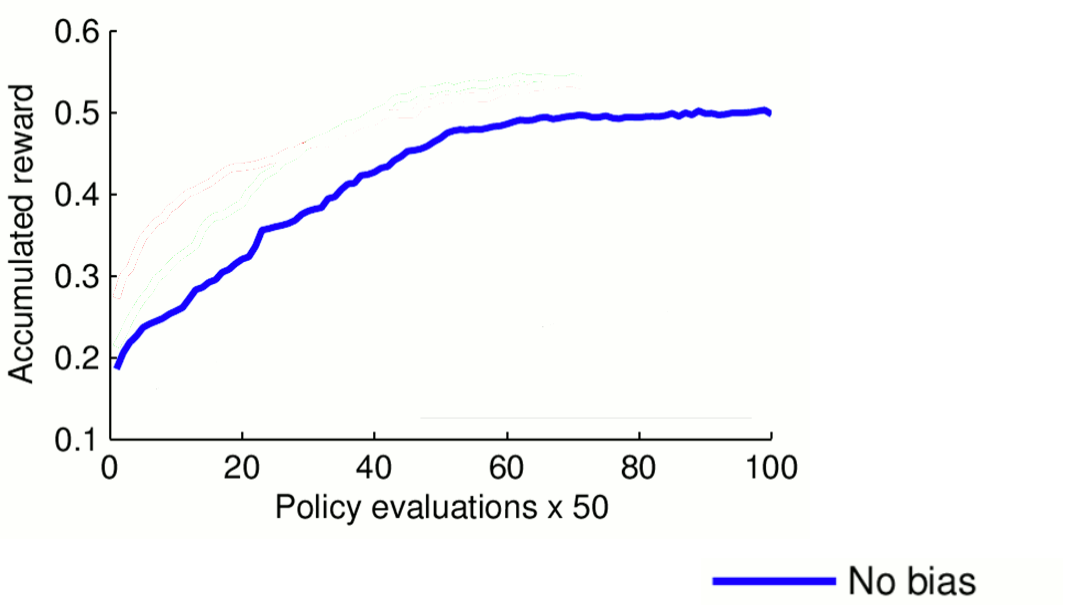
Algorithm: Stochastic Hill-Climbing Search

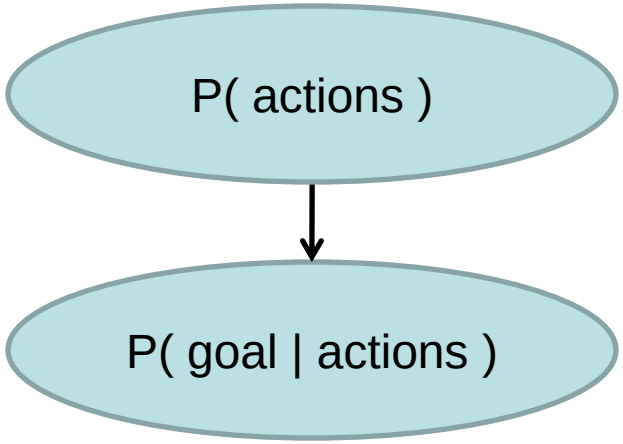
```
Policy = initialize_policy()

Repeat forever

    new policy = propose_change( policy | prior )

    new_prior = find_patterns_in_policy()

    noisy-if ( value(new_policy) > value(policy) )
        policy = new_policy
End;
```

1. Compute value of policy
2. Select a state
3. Propose new action
   from the learned prior
4. **Inference about structure in the policy itself**
5. Compute value of new policy
6. Accept / reject

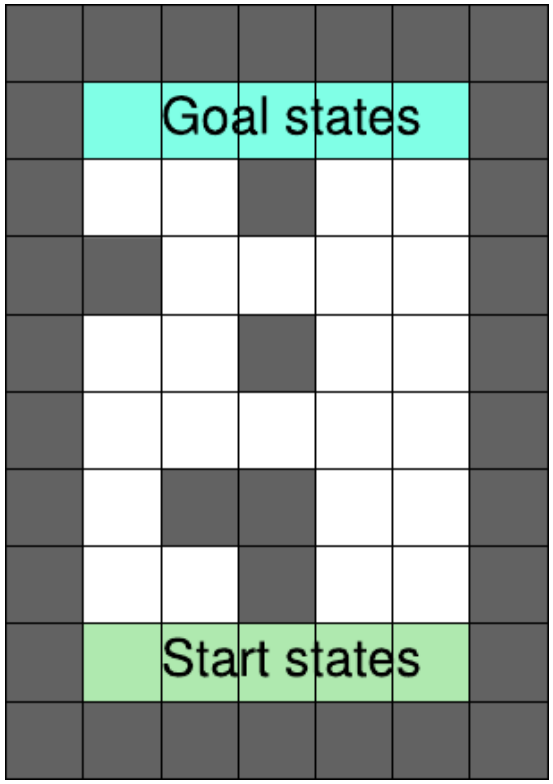
Example: Grid World

Totally uniform prior





Example: Grid World

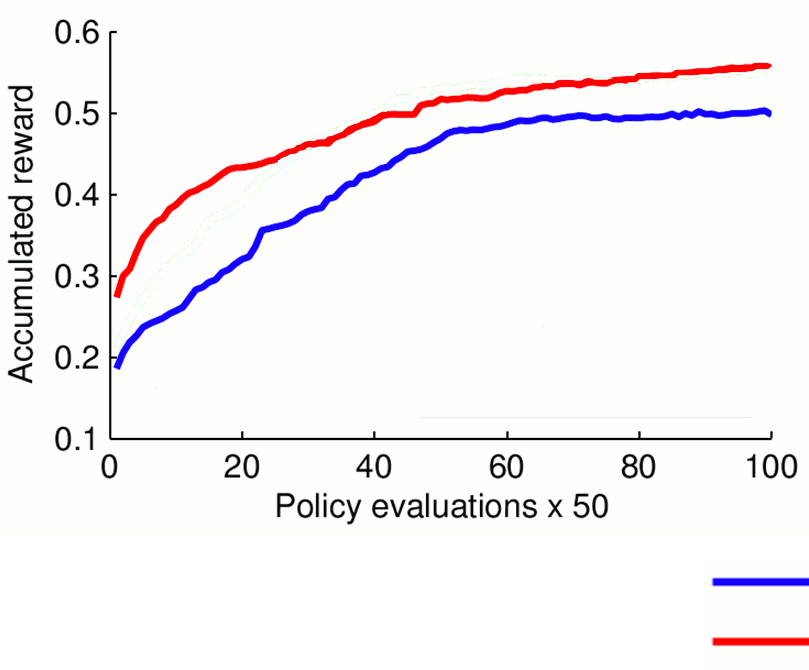


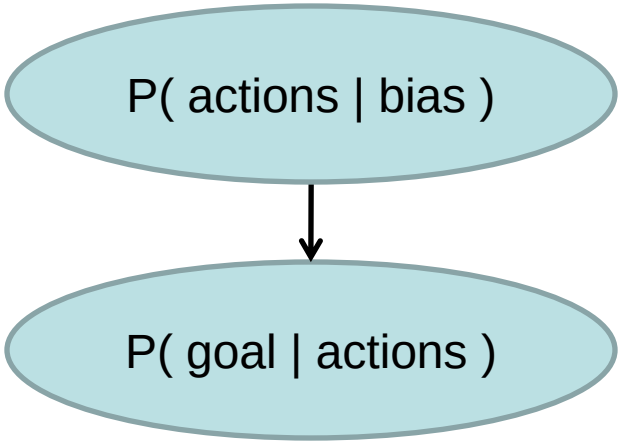
Note:
The optimal action
in most states is
North

Let's put that
in the prior

Example: Grid World

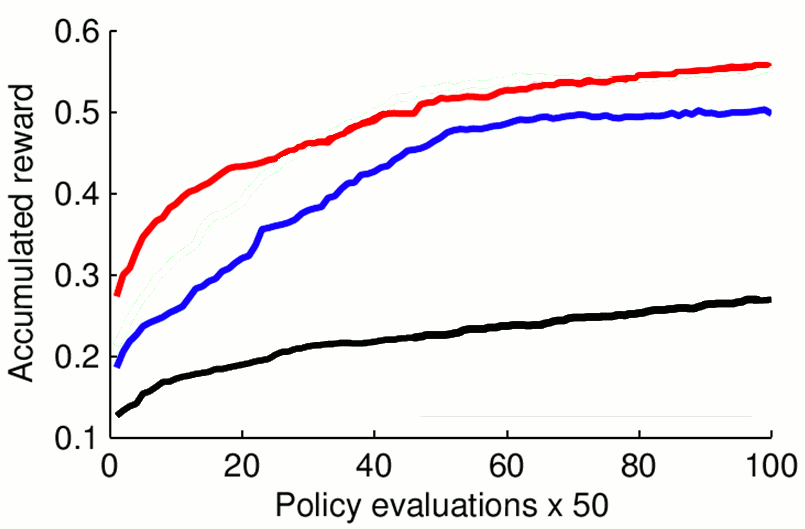
North-biased prior

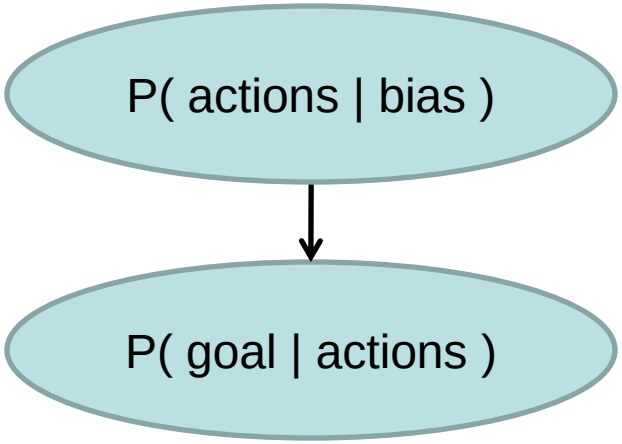




Example: Grid World

South-biased prior

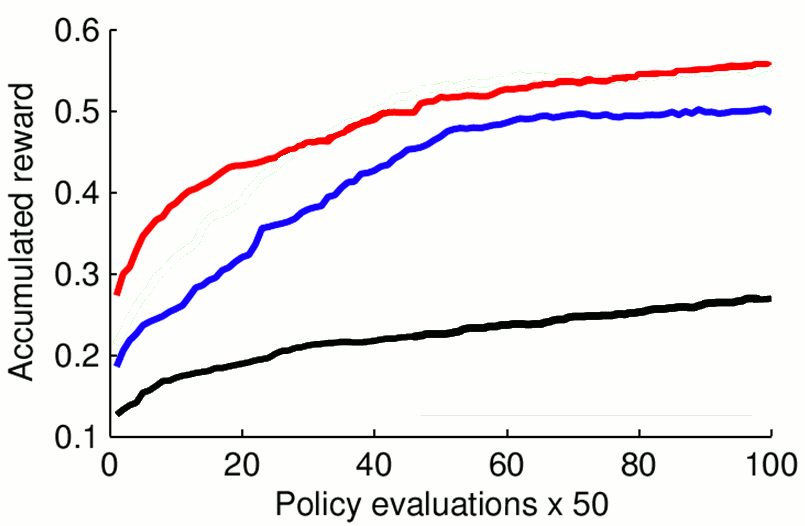


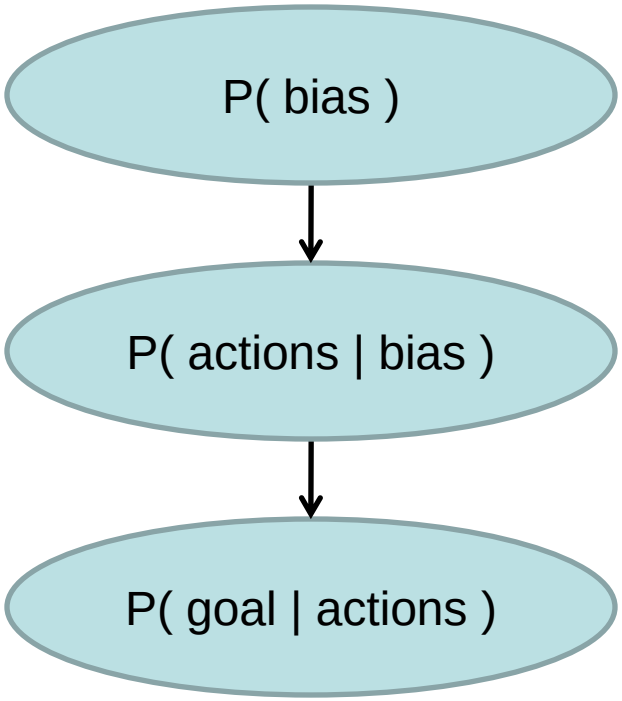


No bias
North bias
South bias

Example: Grid World

Hierarchical (learned) prior

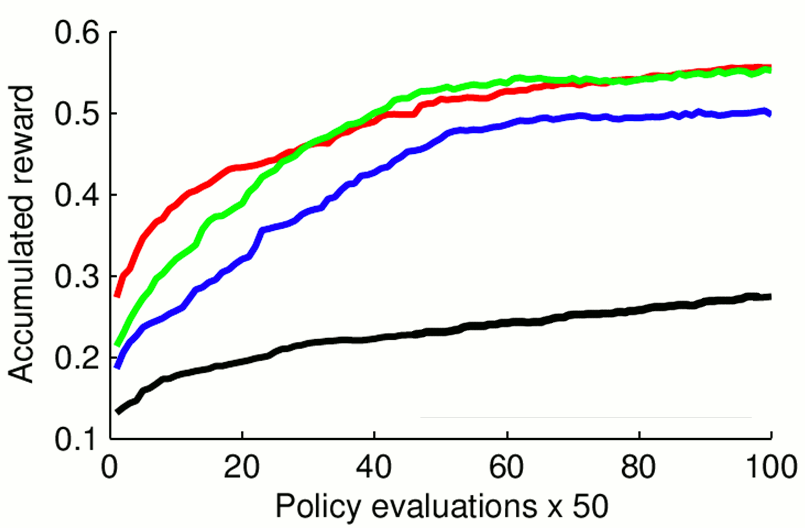


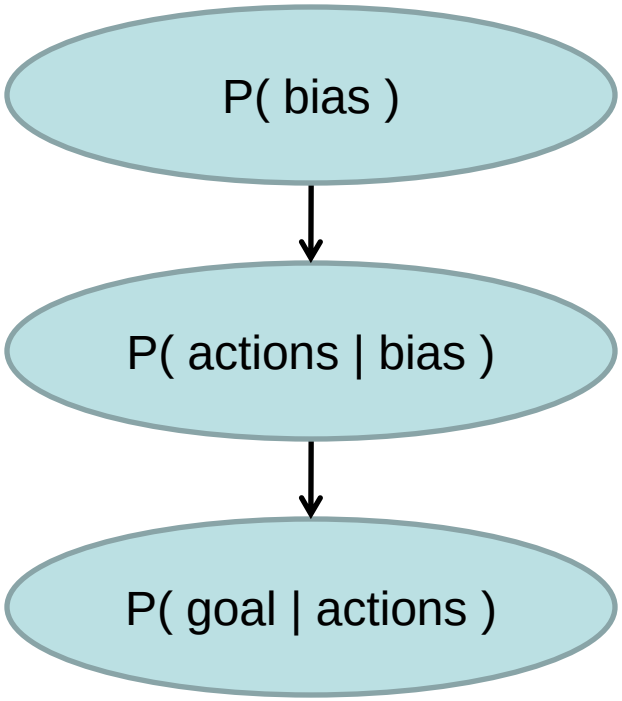


No bias
North bias
Learned bias
South bias

Example: Grid World

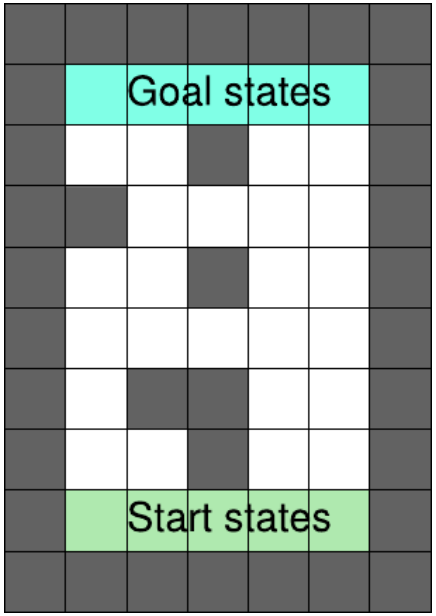
Hierarchical (learned) prior

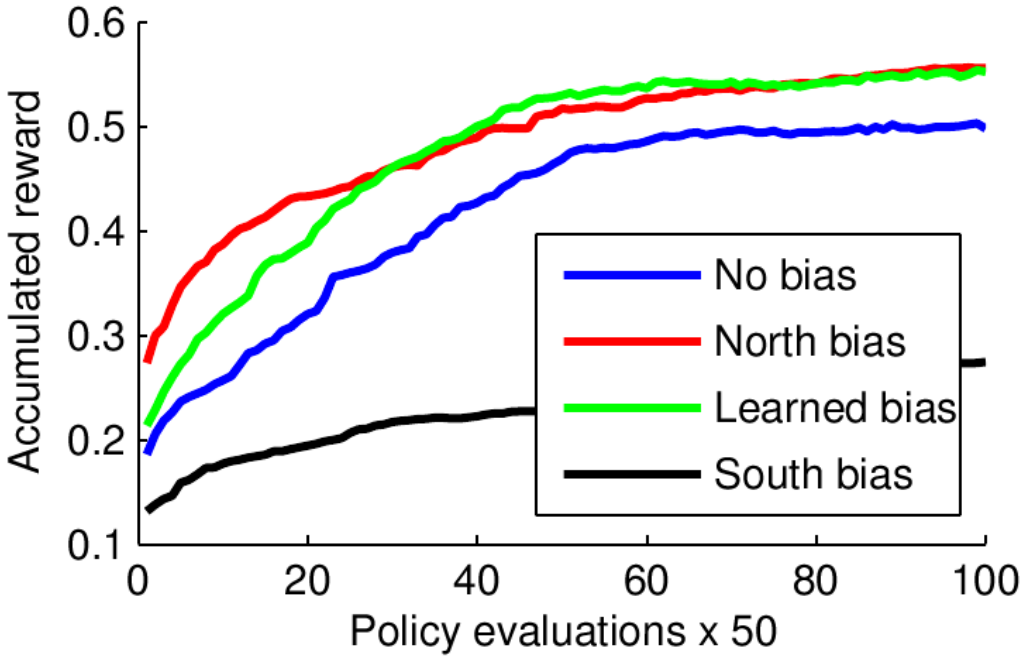




No bias
North bias
Learned bias
South bias

Grid World Conclusions



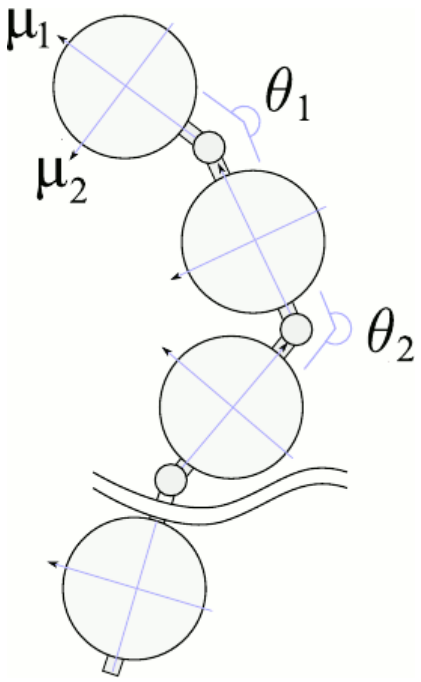


Learning the prior **alters the policy search space!**

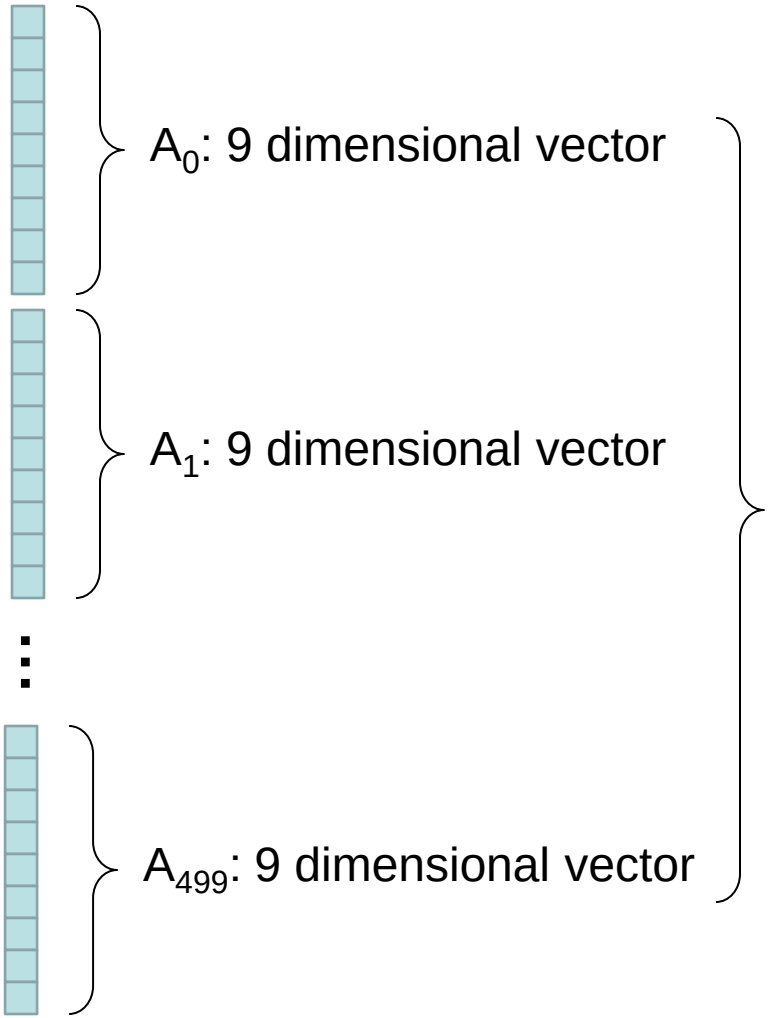
This is the introspection I was talking about!

Some call this **the blessing of abstraction**

Back to Snakes



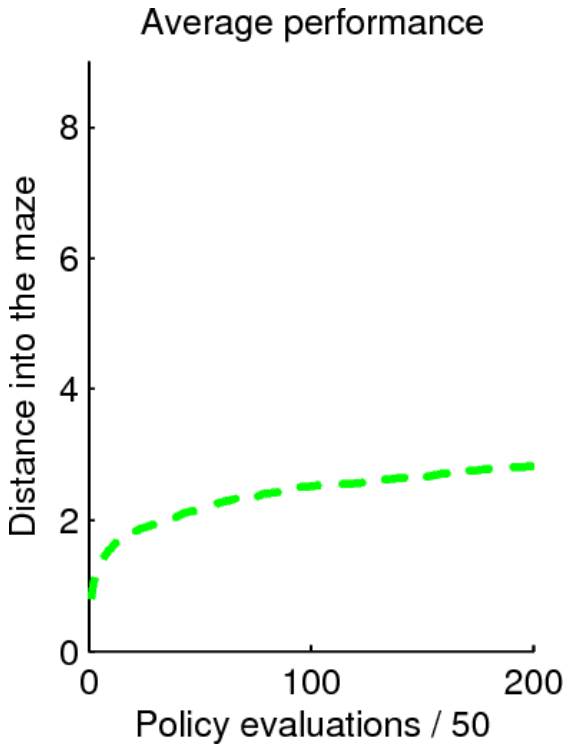
Finding a Good Trajectory

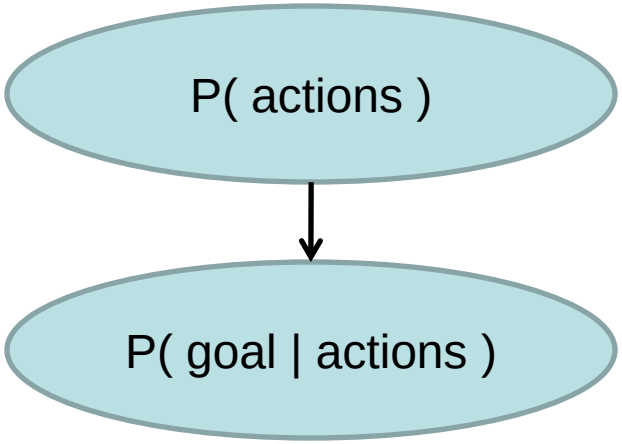


Simplest approach:
direct optimization

…of a **4,500 dimensional function!**

Direct Optimization Results



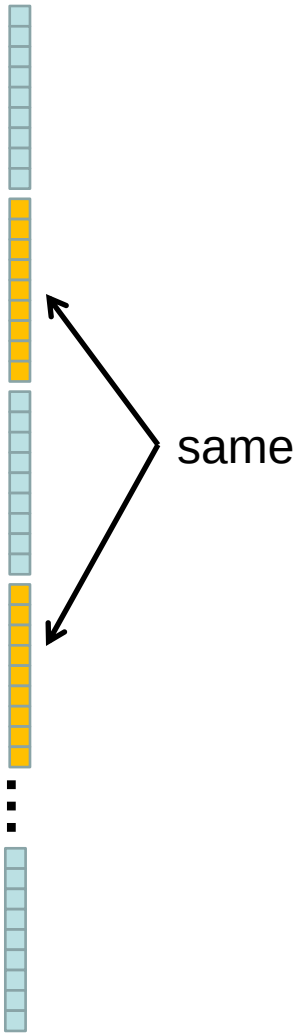


Direct optimization

Repeated Action Structure

Suppose we encode some prior knowledge: **some actions are likely to be repeated**

Repeated Action Structure



Suppose we encode some prior knowledge: **some actions are likely to be repeated**

If we can tie them together, this would **reduce the dimensionality of the problem**

Of course, we don't know which ones should be tied. **So we'll put a distribution over all possible ways of sharing.**

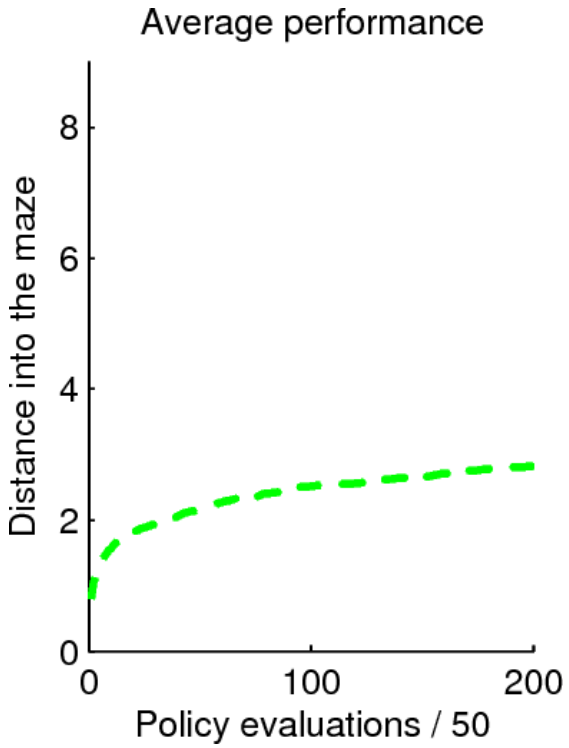
Whoa!

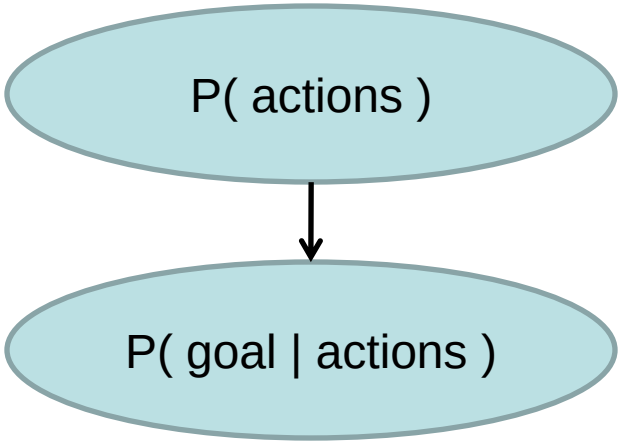
Wait, wait, wait.

Are you seriously suggesting taking a hard problem, and making it **harder** by increasing the number of things you have to learn?

Doesn't conventional machine learning wisdom say that as you **increase model complexity** you run the **risk of overfitting**?

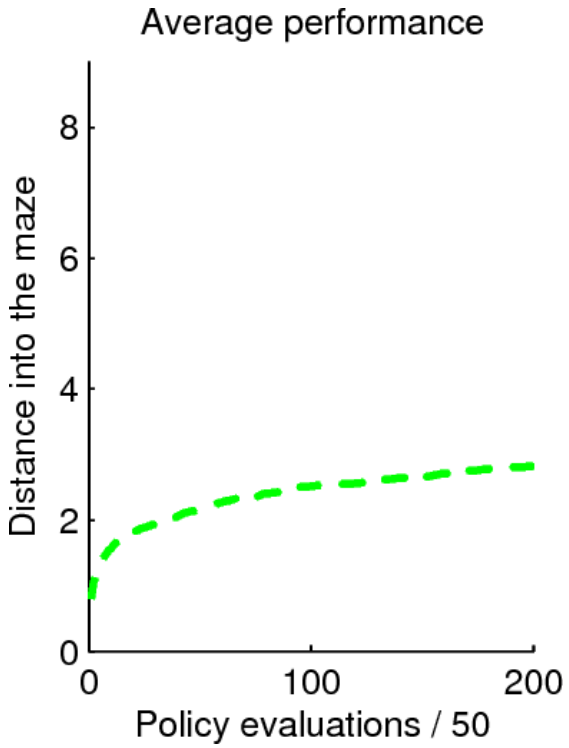
Direct Optimization

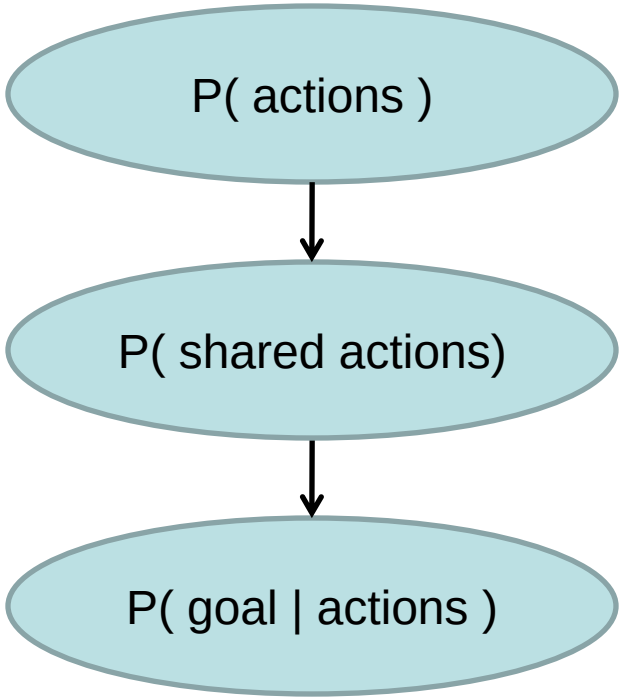




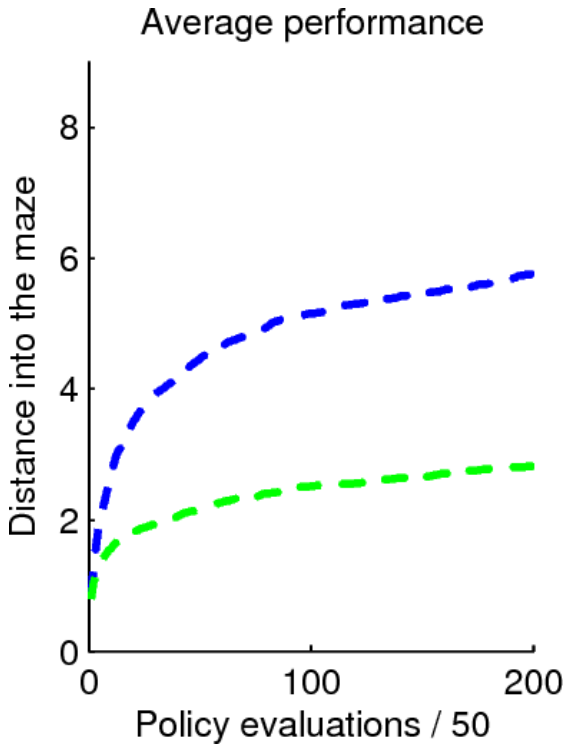
Direct optimization

Shared Actions



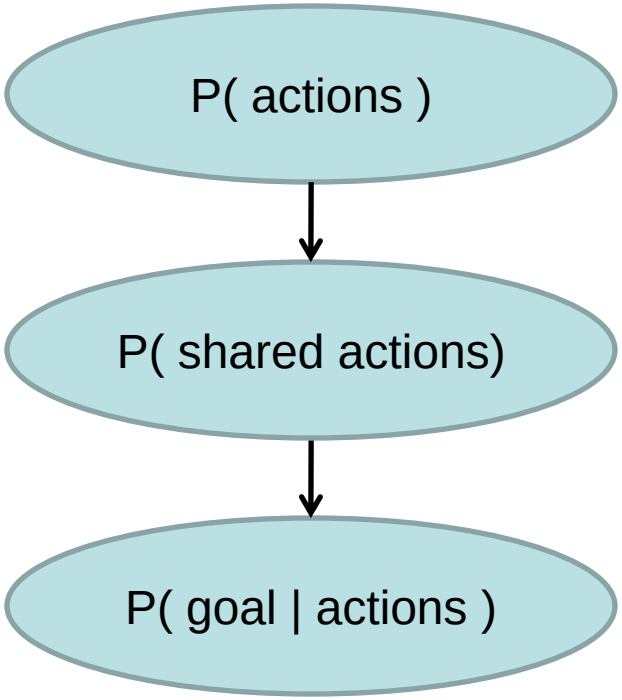


Shared Actions

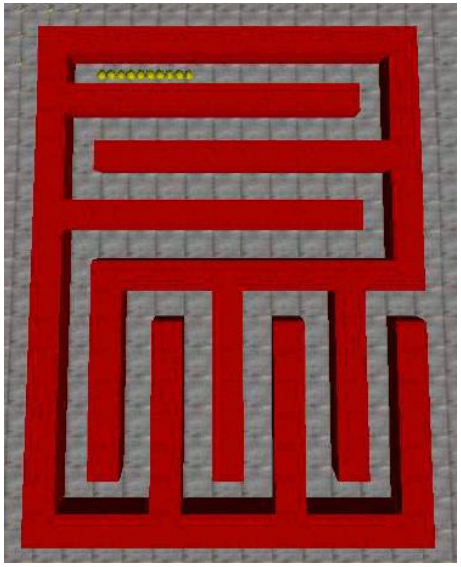


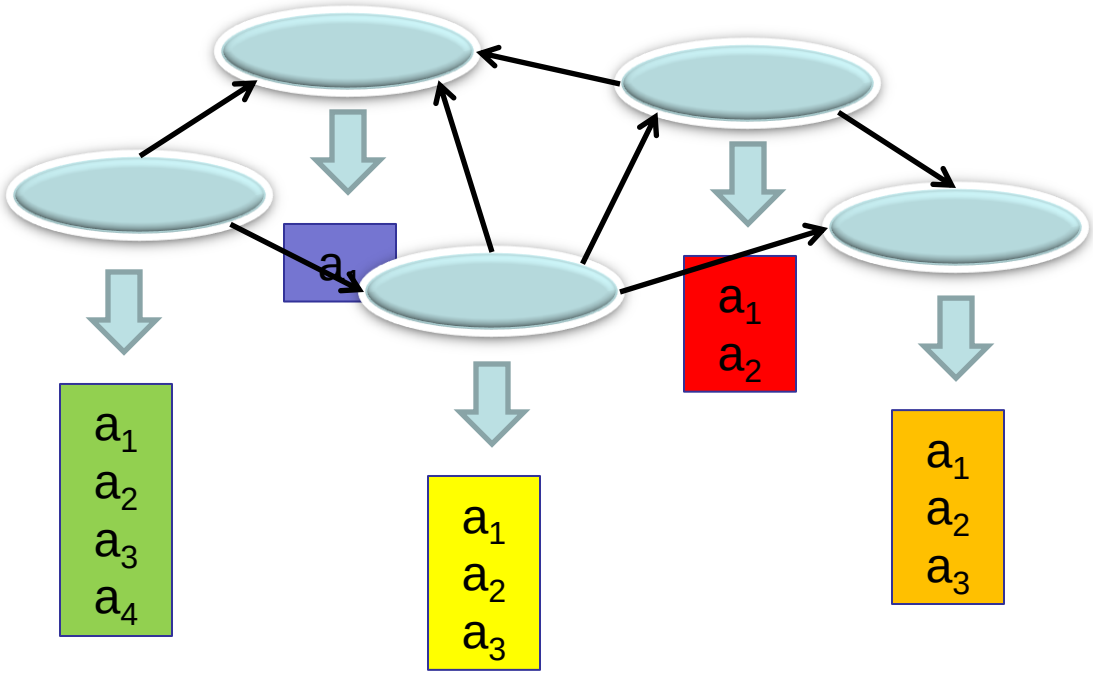
Reusable actions

Direct optimization



States of Behavior in the Maze





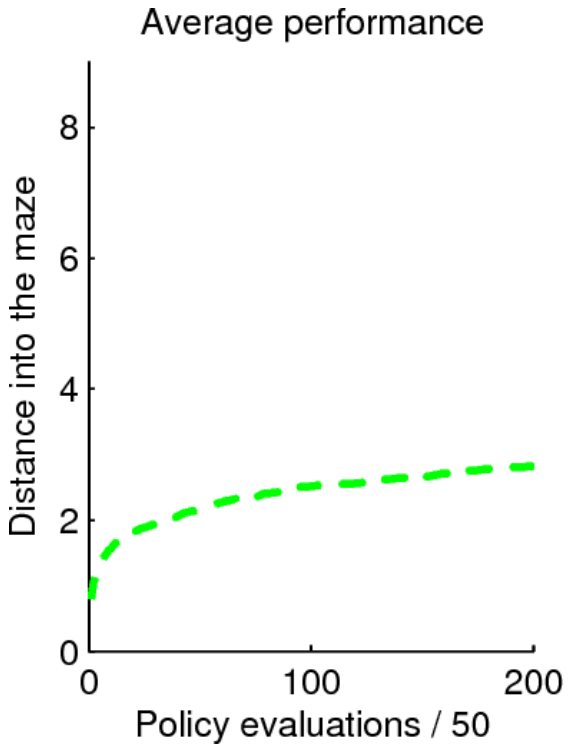
Favor
state reuse

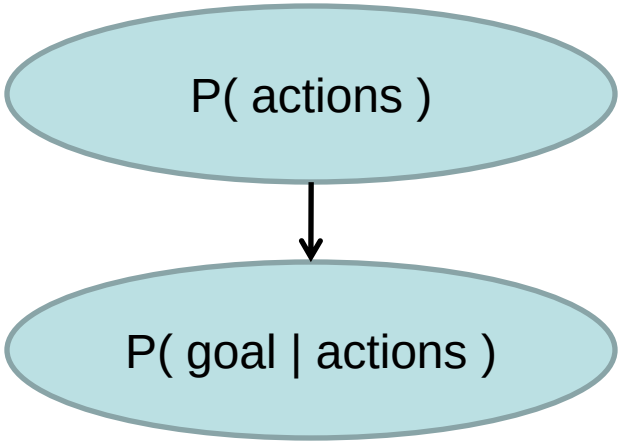
Favor
transition reuse

Each state picks its
own action

Potentially **unbounded number** of states and primitives

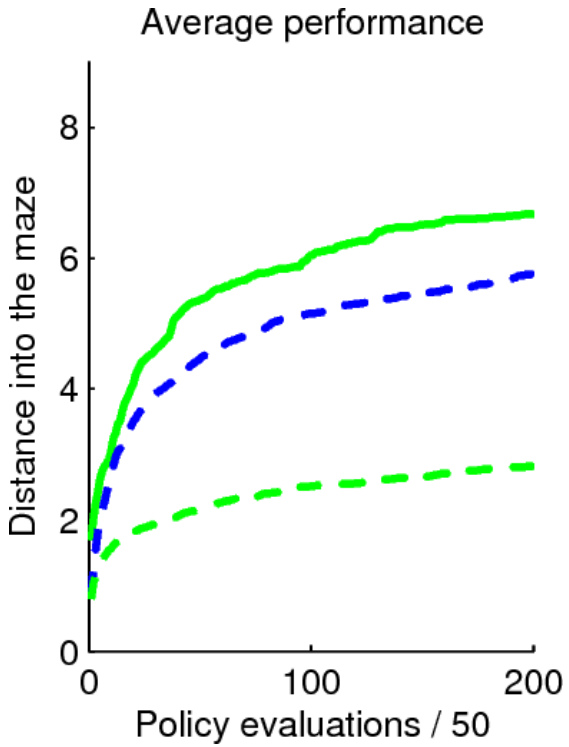
Direct Optimization

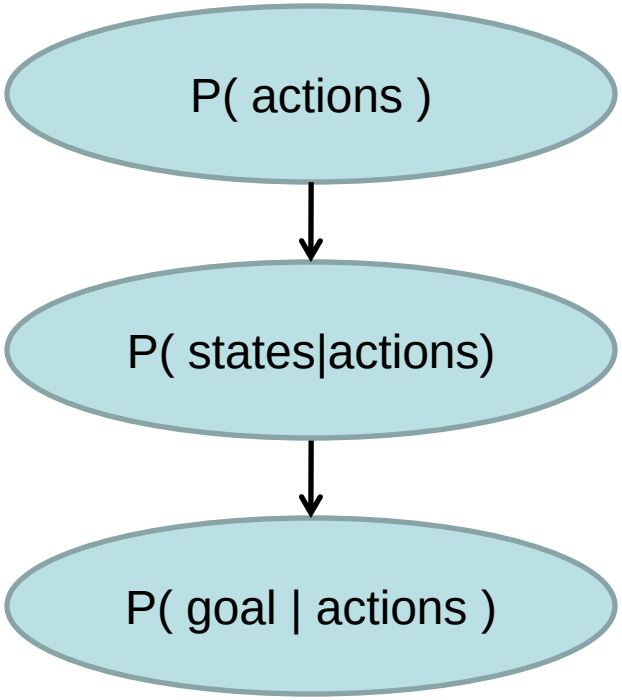




Direct optimization

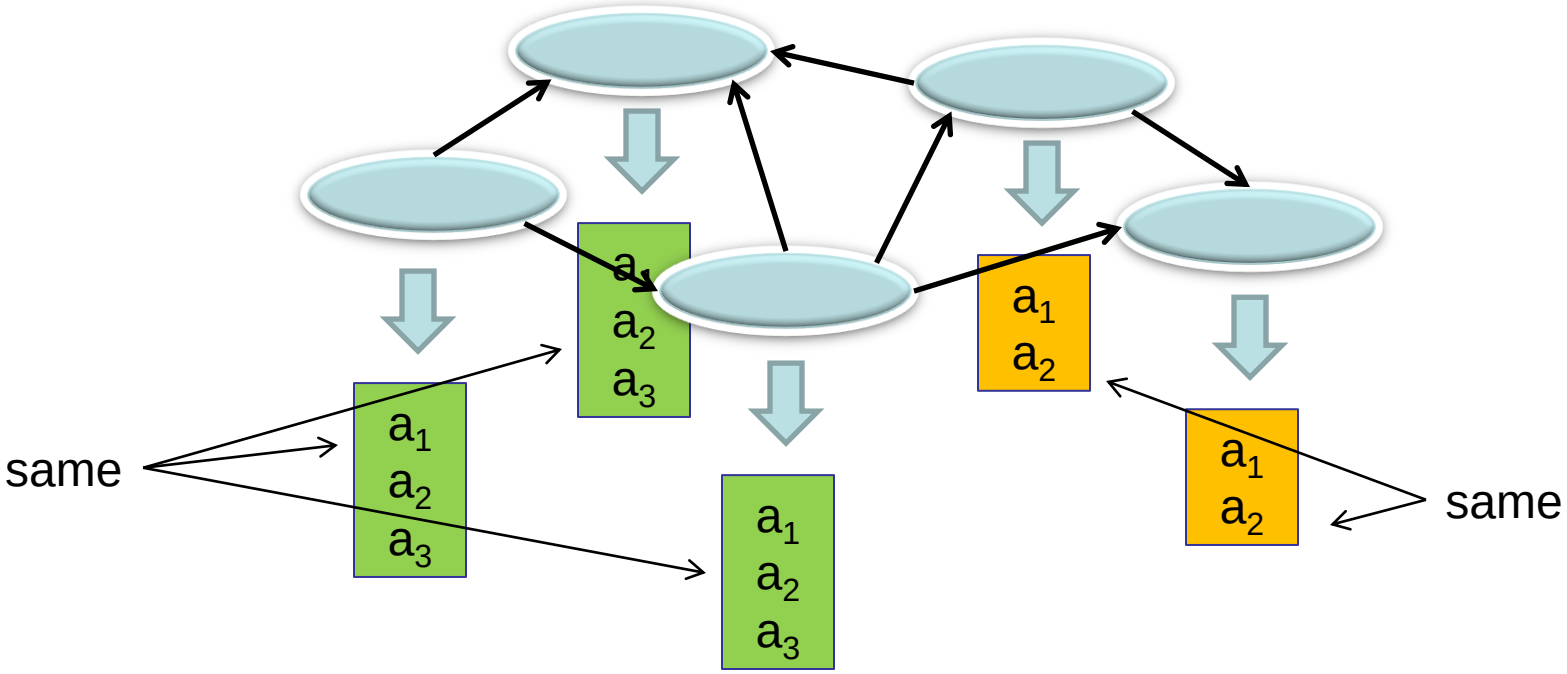
Finite State Automaton



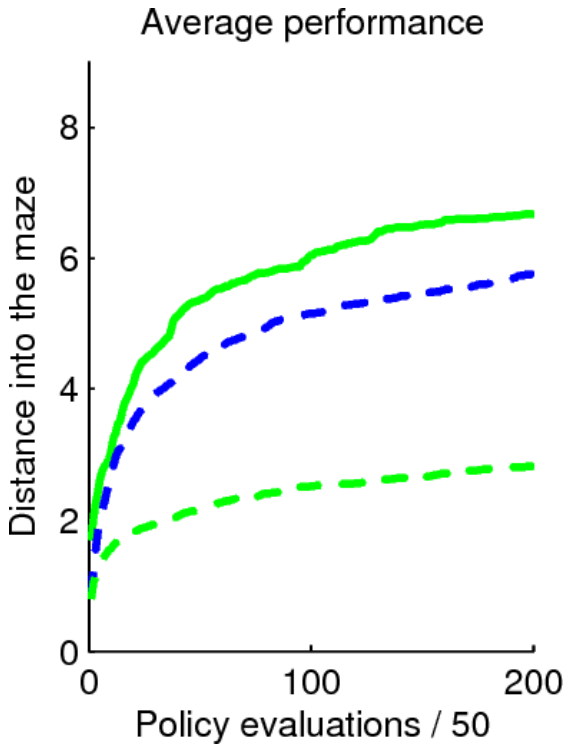


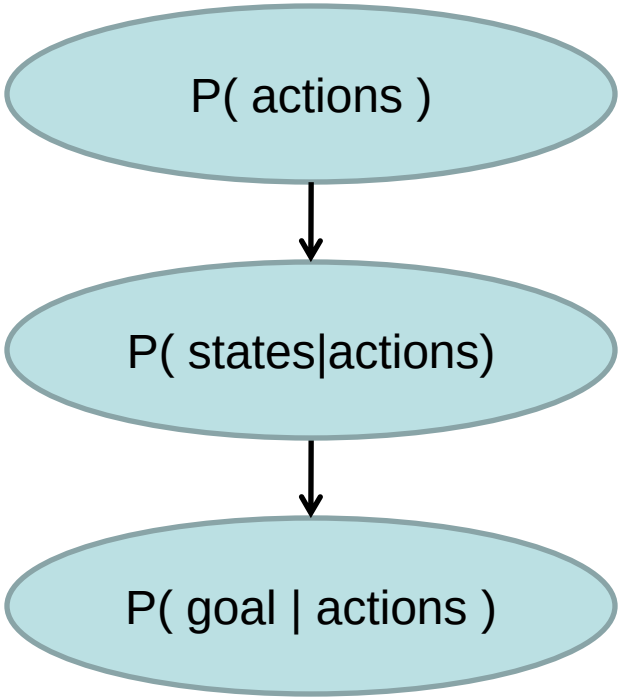
Sharing Action Sequences

Add the ability to
reuse actions across states

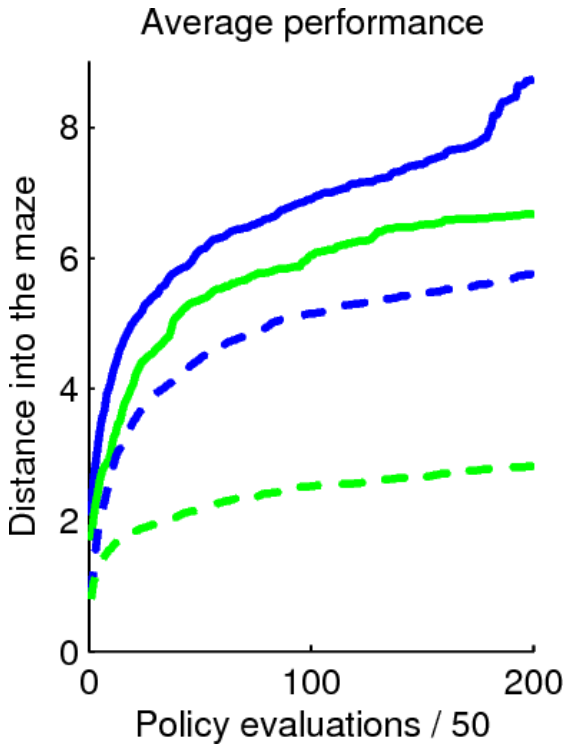


Finite State Automaton

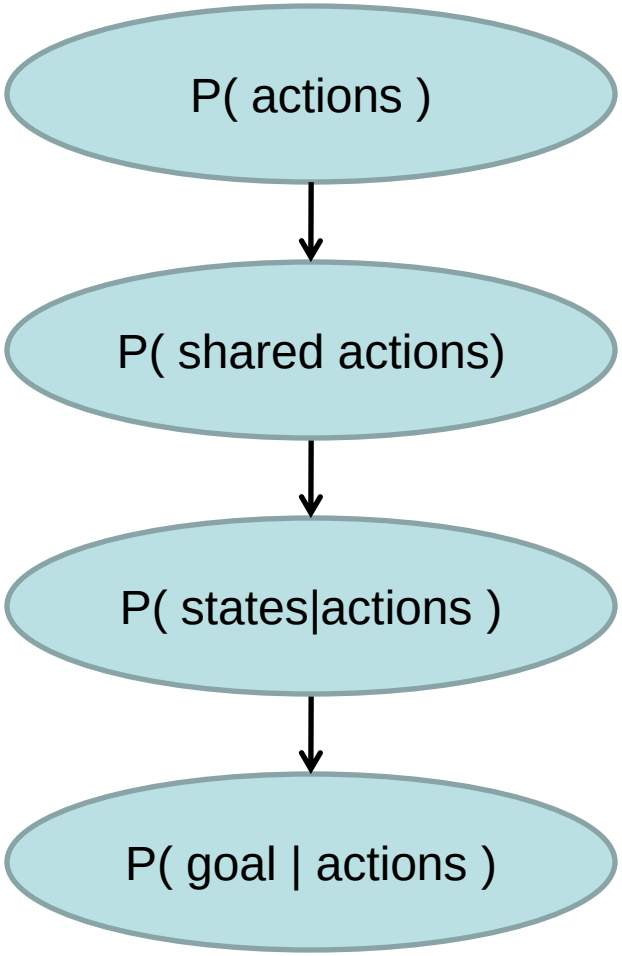




Final Model



Reusable states + reusable actions
Reusable states
Reusable actions
Direct optimization



Snake's Policy Prior

State prior:

Nonparametric finite state controller

$$G_0 \sim \mathrm{GEM}(\alpha)$$
$$G_{s,o} \sim \mathrm{DP}(G_0, \alpha)$$

$$\mathrm{state}_0 \sim G_0$$
$$\mathrm{state}_{t+1} \sim G_{\mathrm{state}_t, \mathrm{obs}_t}$$

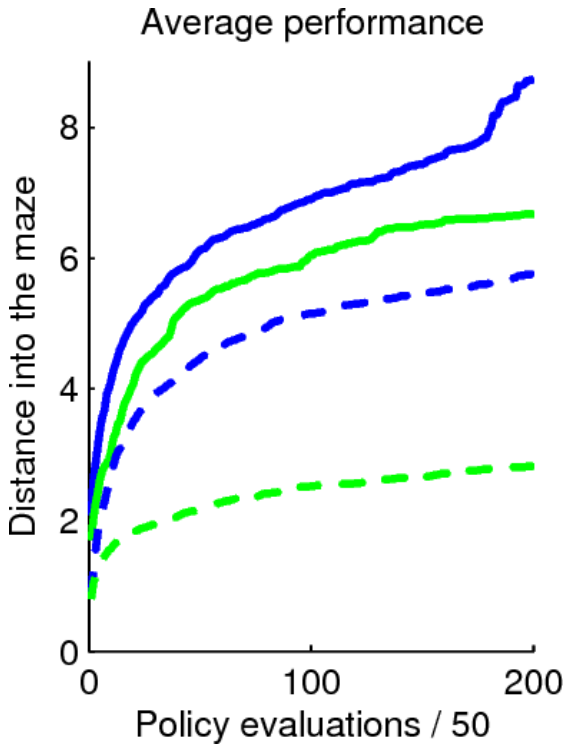
Hierarchical action prior:

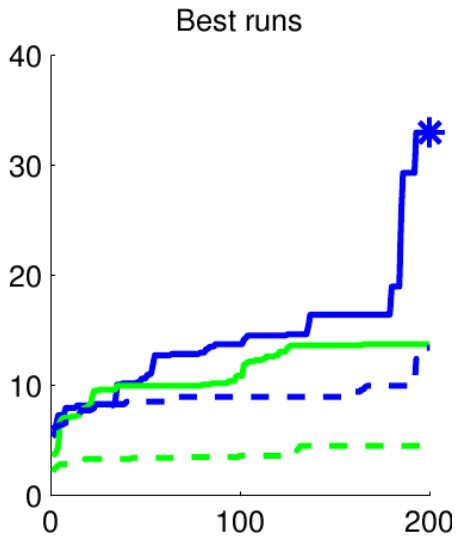
Open-loop motor primitives

$$n \sim \mathrm{Poisson}(\lambda_n)$$
$$k_n \sim \mathrm{Poisson}(\lambda_k)$$
$$a_{kt} \sim 9 \text{ dim. } \mathcal{N}(0, \sigma^2)$$
$$\mathrm{action} \sim \mathrm{DP}(a_k, \alpha)$$

Note: this is like an **HDP-HMM**

This Gets All the Way Through!



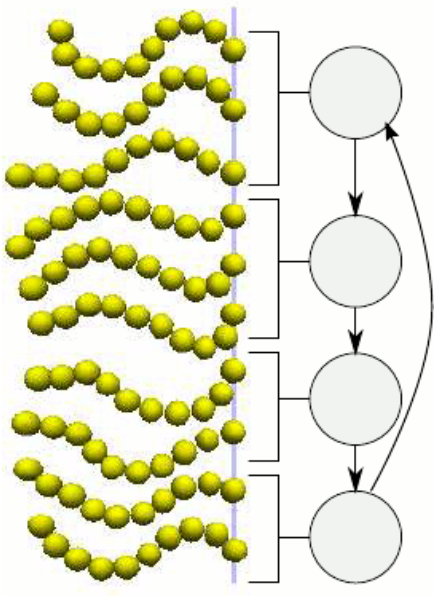


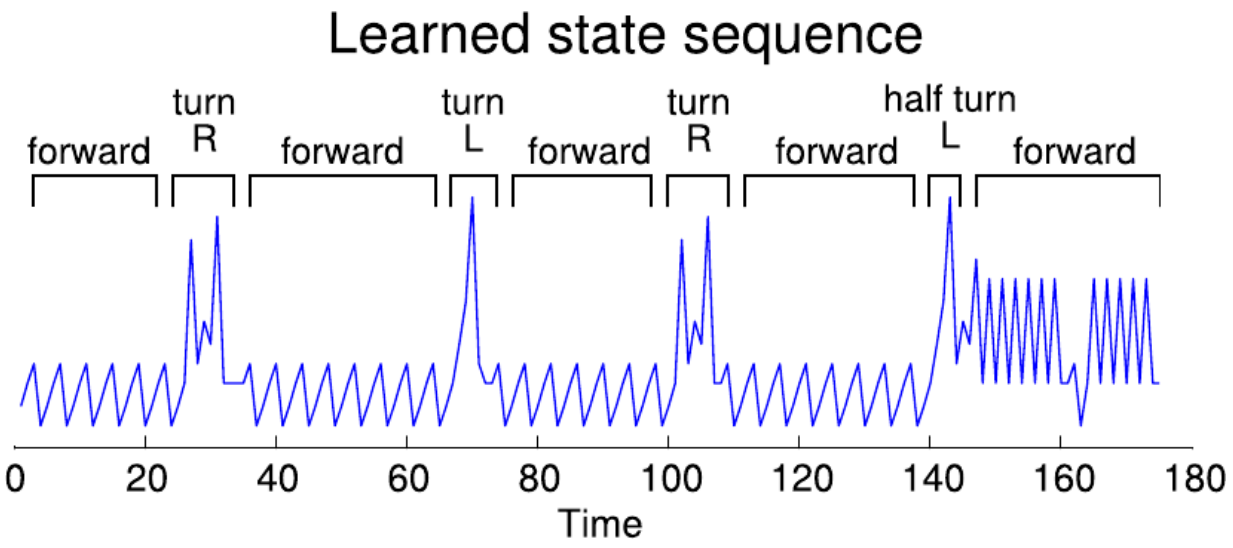
Reusable states + reusable actions
Reusable states
Reusable actions
Direct optimization

At this point, we have essentially **learned everything about the domain!**

Snakes in a Maze

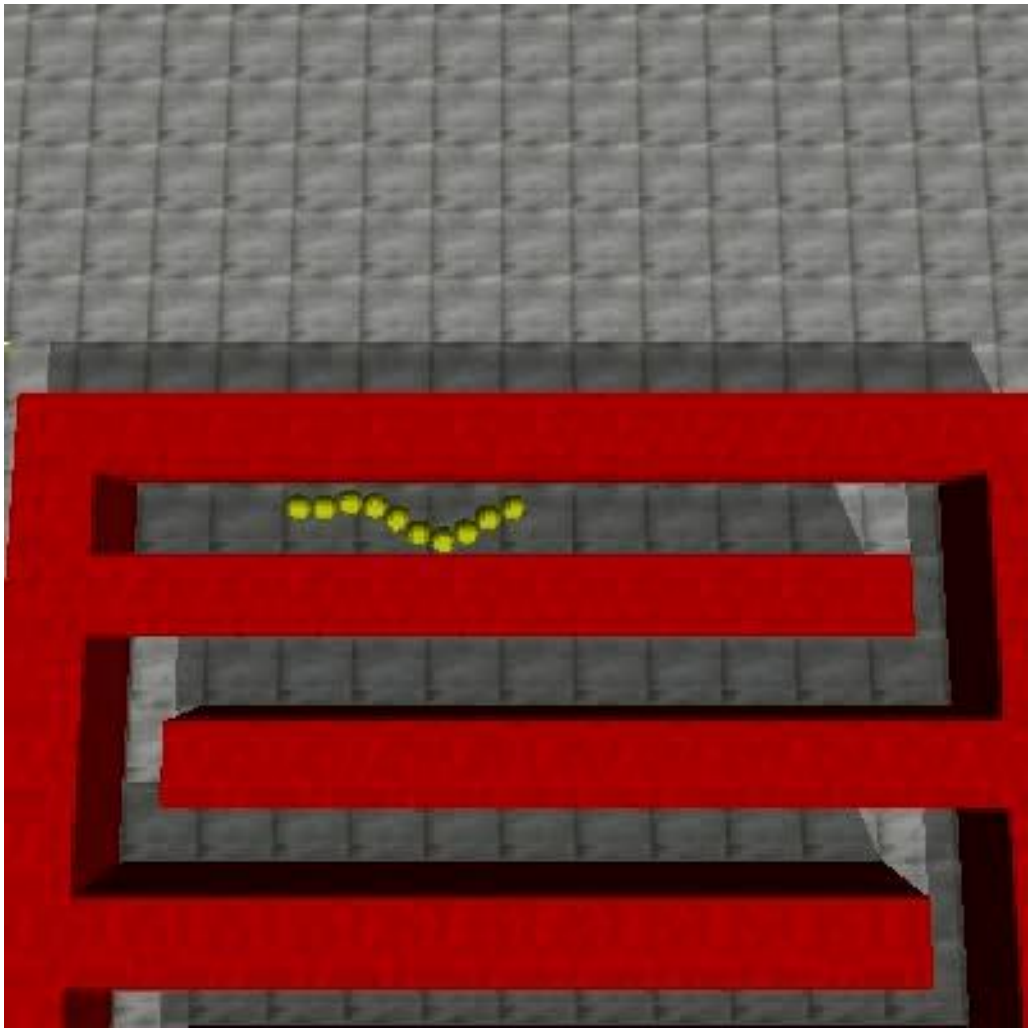
Let's examine **what was learned**



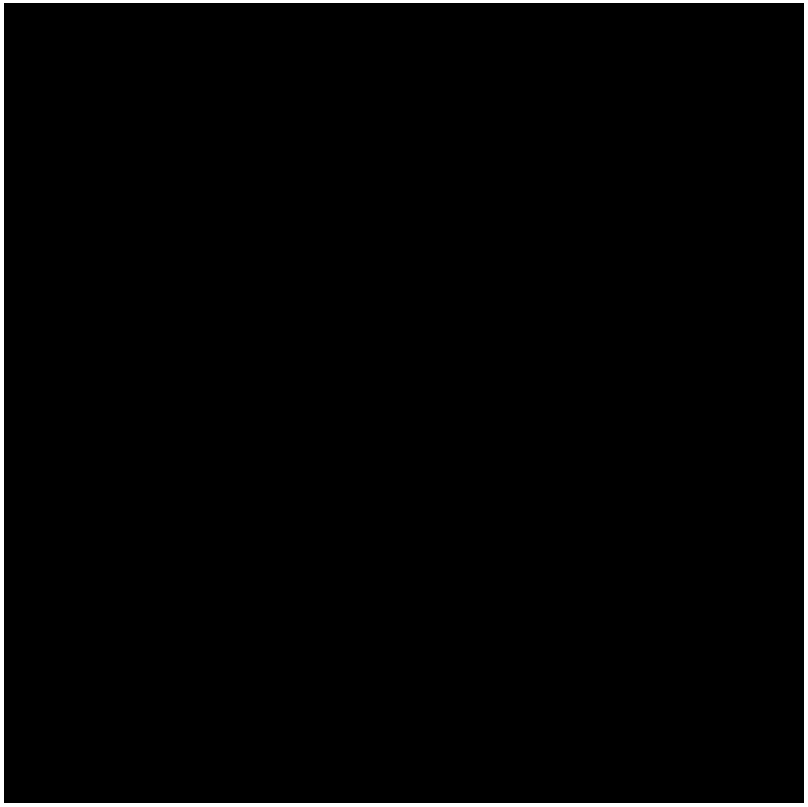


Four states **wiggle forward**

Snakes in a Maze



Bonus: Spider in a Maze



Key Point

Increasing the richness of our model **decreased the complexity of solving the problem**

Summary

- **Search is important for AI / ML in general**
  - Combinatorial optimization, path planning, probabilistic inference…
- **Adaptive search** can be useful for many problems
  - Transferring useful information within or between tasks
  - Learned parameter tying simplifies the search space
- Contribution: a novel application of Bayes
  - Modeling side: finding and leveraging **structure in actions**
  - Computational side: priors can **structure a search space**
- **Many future possibilities here!**

Outline

- Intro: Bayesian Reinforcement Learning
- Planning: Policy Priors for Policy Search
- Model building: The Infinite Latent Events Model
- Conclusions

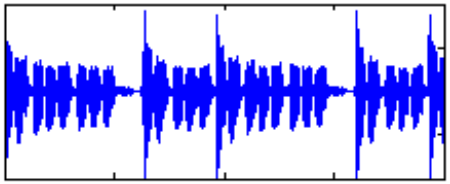
The Infinite Latent Events Model

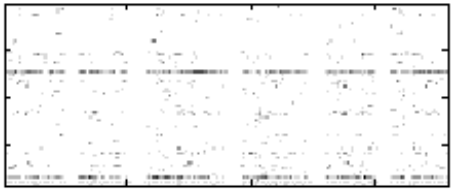
Joint work with Noah Goodman, Dan Roy and Joshua Tenenbaum

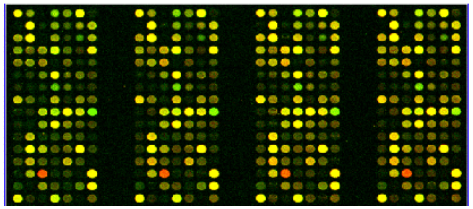
Learning Factored Causal Models

Suppose I hand you…

- **Temporal gene expression data**
- **Neural spike train data**
- **Audio data**
- **Video game data**







…and I ask you to build a predictive model

What do these problems have in common?

- Must find **explanatory variables**
  - Clusters of genes / neurons; individual sounds; sprite objects
  - Could be **latent or observed**
- Must identify **causal relationships between them**

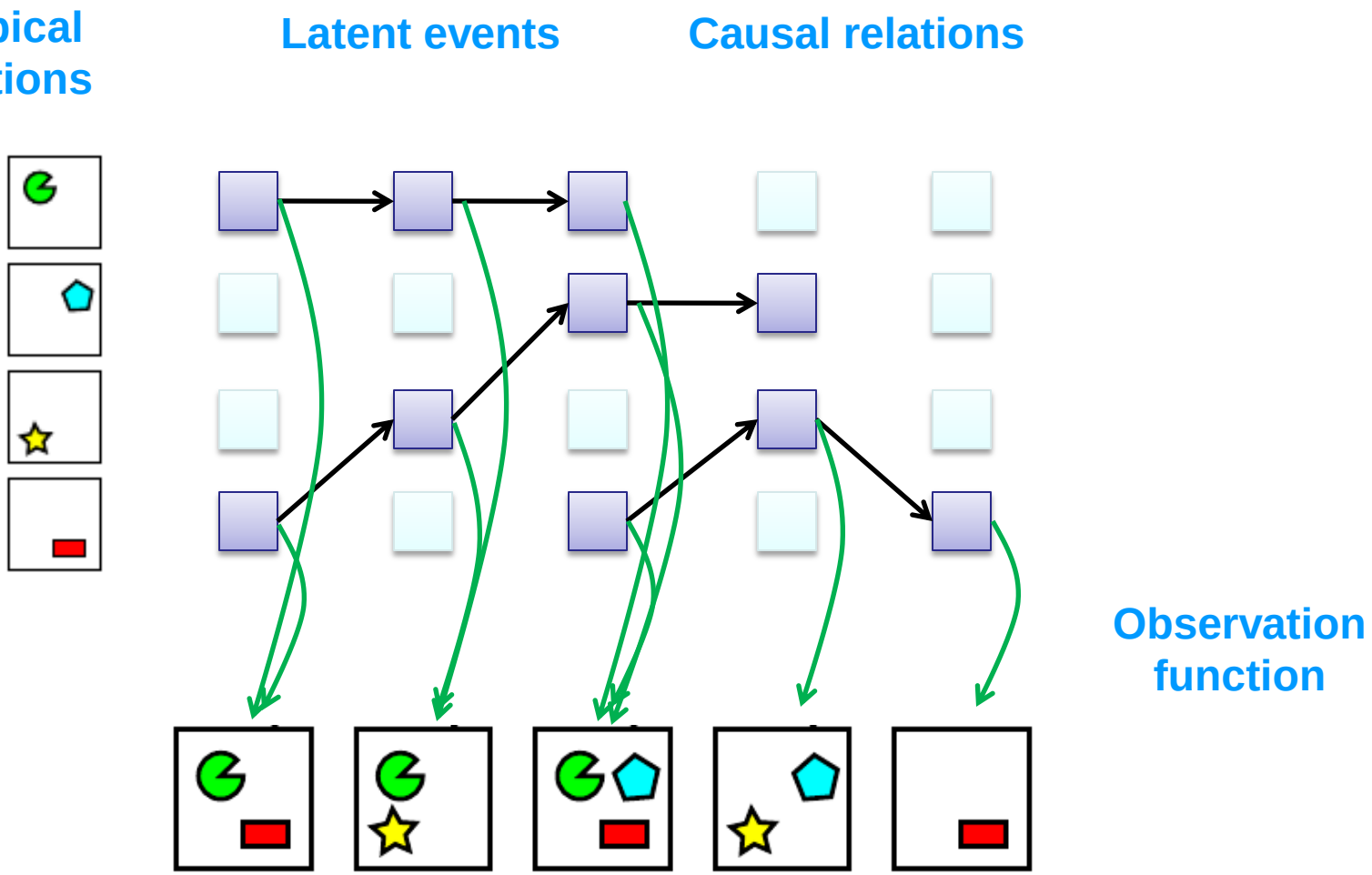
Problem Statement

Given a **sequence of observations**

Simultaneously discover

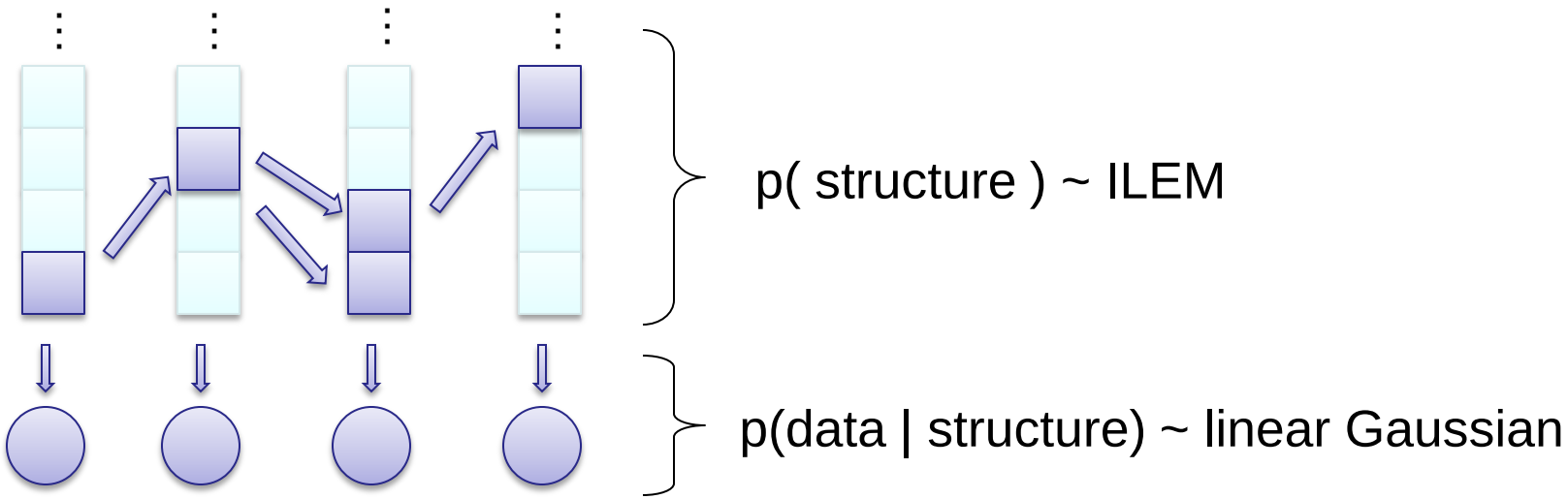
- **Number** of latent factors (events)
- **Which events** are active at which times
- **The causal structure** relating successive events
- How **events combine to form observations**

Example Factorization

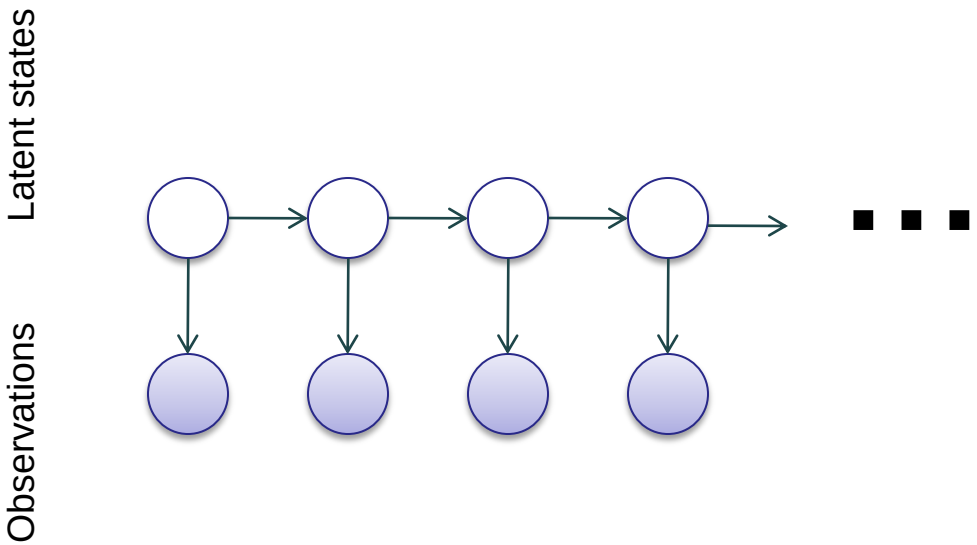


Our Model: The ILEM

The ILEM is a **distribution over factored causal structures**

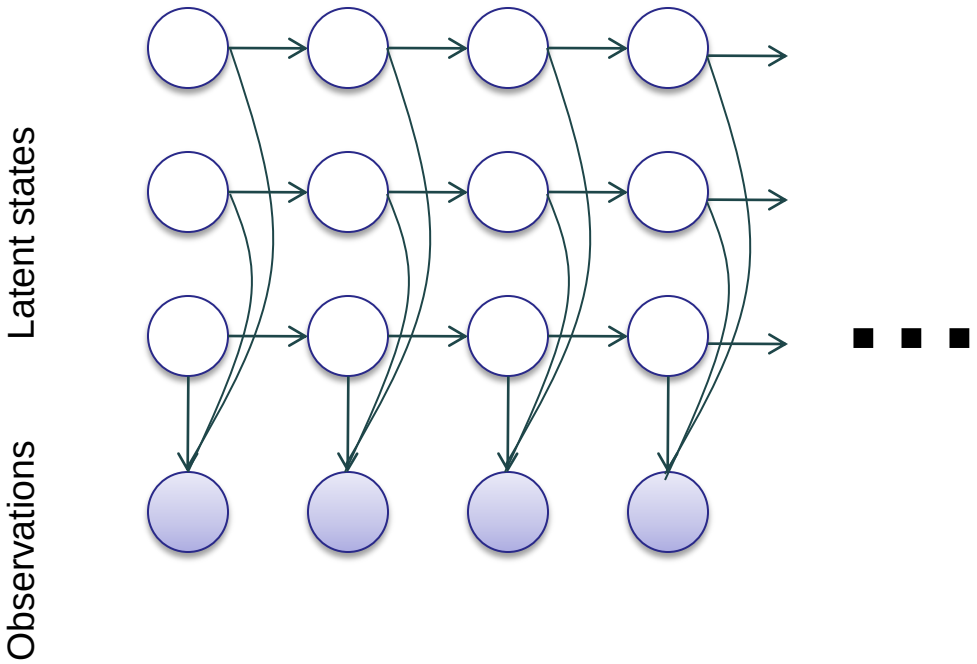


Relationship to Other Models



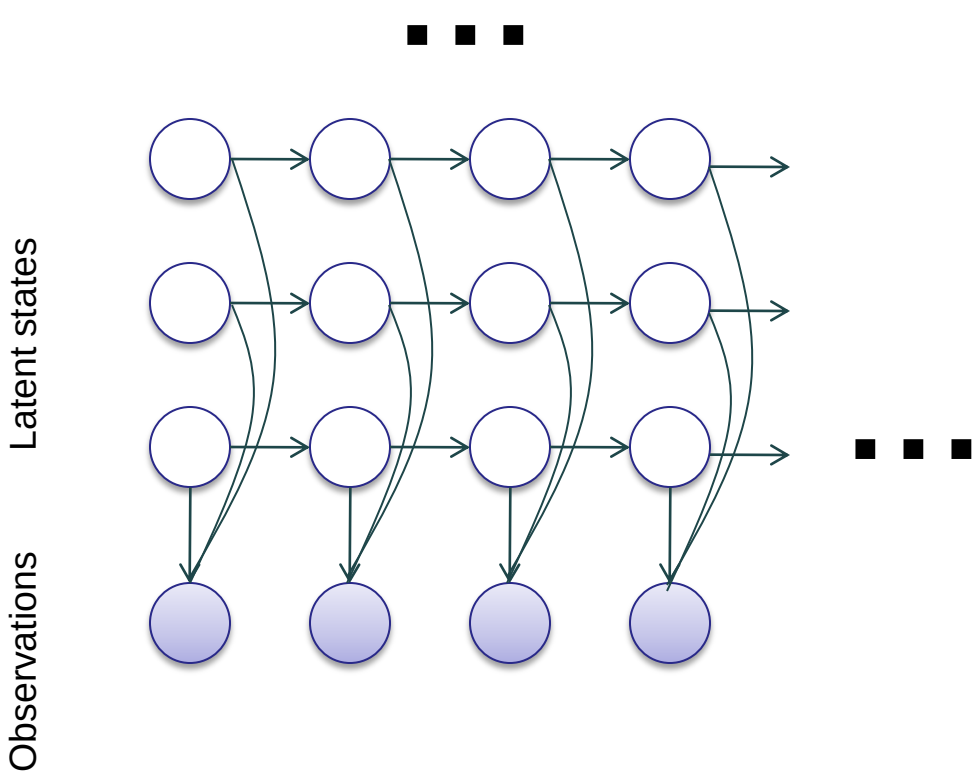
HMM

Relationship to Other Models



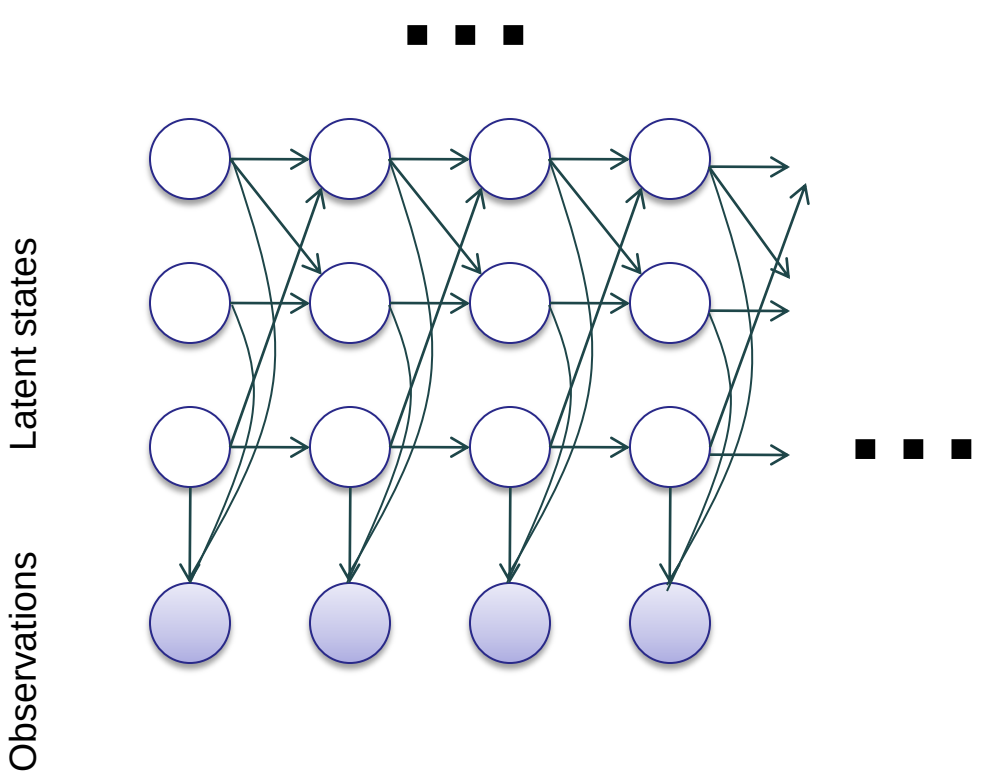
Factorial HMM

Relationship to Other Models



Infinite Factorial HMM

Relationship to Other Models

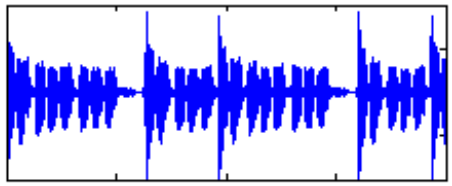


Infinite Latent Events Model

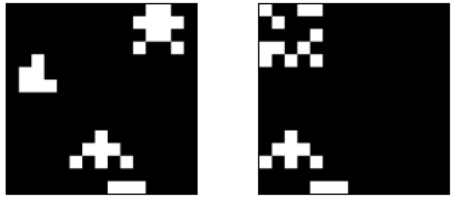
Applications of the ILEM

Experiments in four domains:

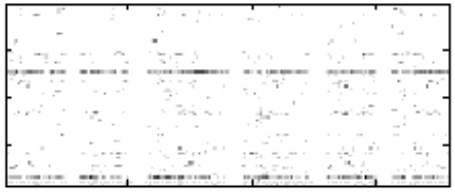
- Causal source separation



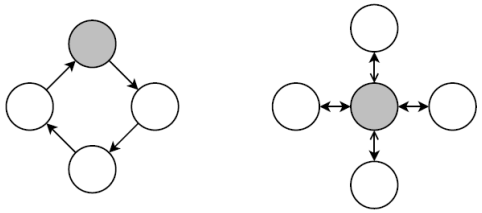
- Simple video game



- Neural spike train data



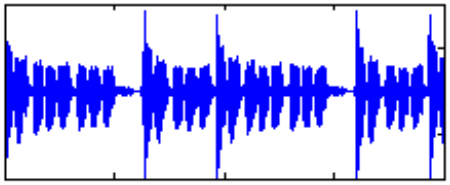
- Network intruder detection



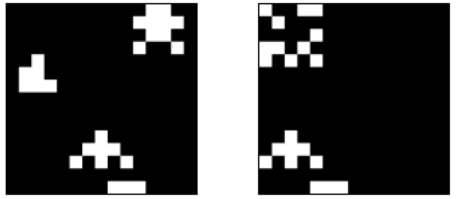
Applications of the ILEM

Experiments in four domains:

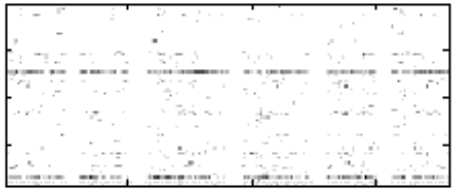
Causal source separation



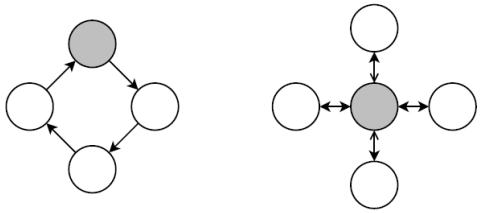
Simple video game



Neural spike train data



Network intruder detection



Neural Spike-Train Data

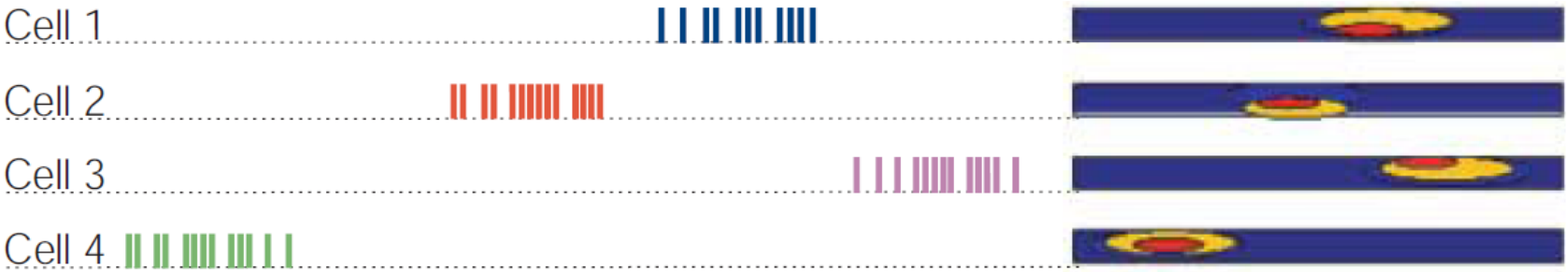
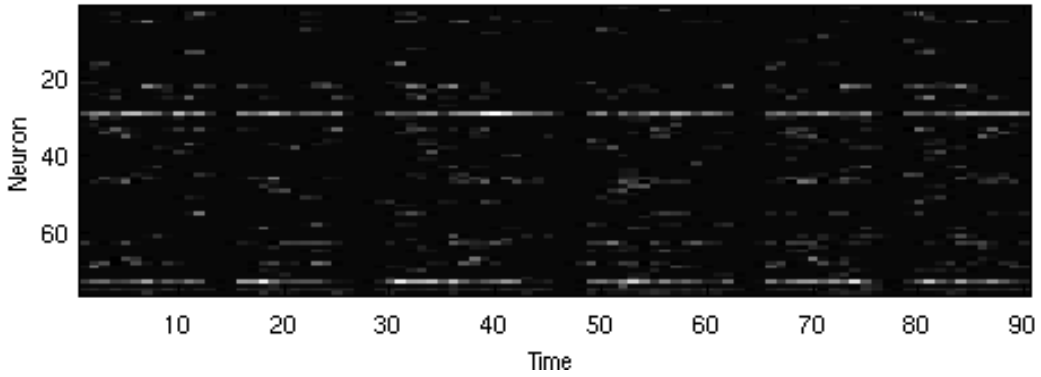


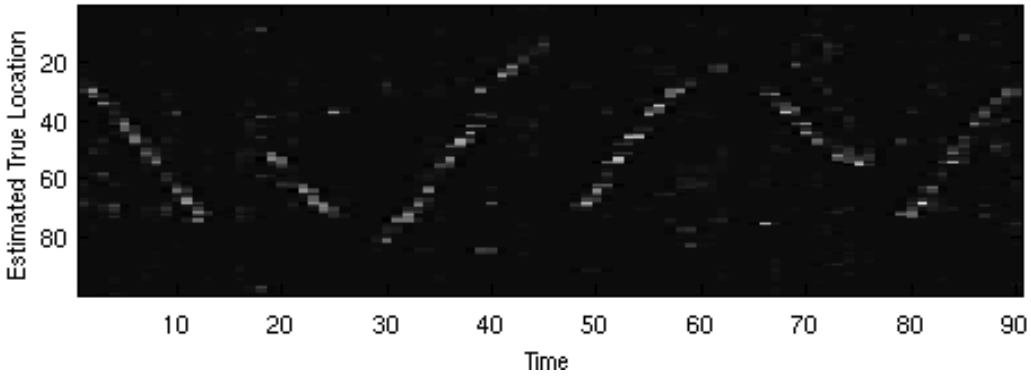
Image from **NMDA receptors, place cells and hippocampal spatial memory**
Kazu Nakazawa, Thomas J. McHugh, Matthew A. Wilson & Susumu Tonegawa
Nature Reviews Neuroscience 5, 361-372 (May 2004)

Setup

Original data



Place cell tuning curves

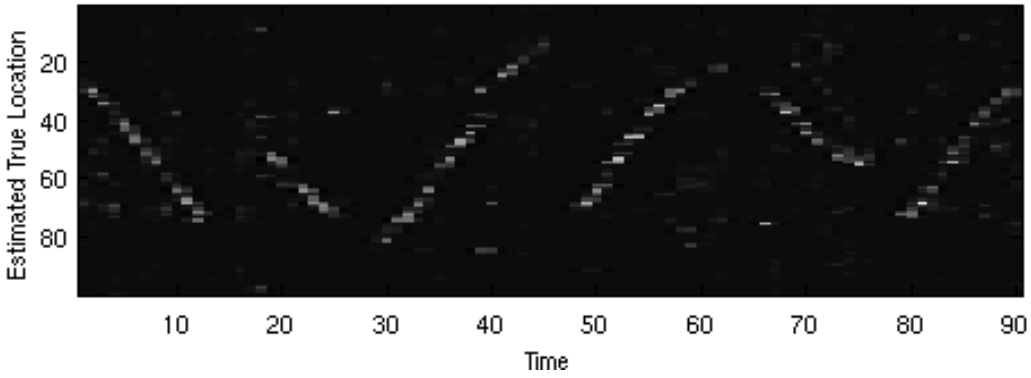


Important note:

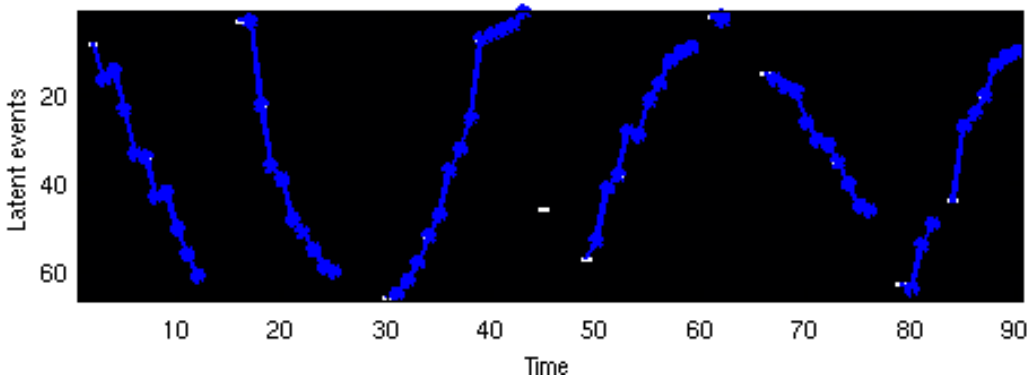
Tuning curves were generated from supervised data!

Results

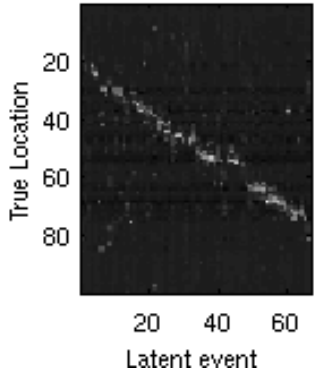
Estimated ground truth (supervised)



ILEM Results (unsupervised)



Learns latent prototypical neural activations which **code for location**



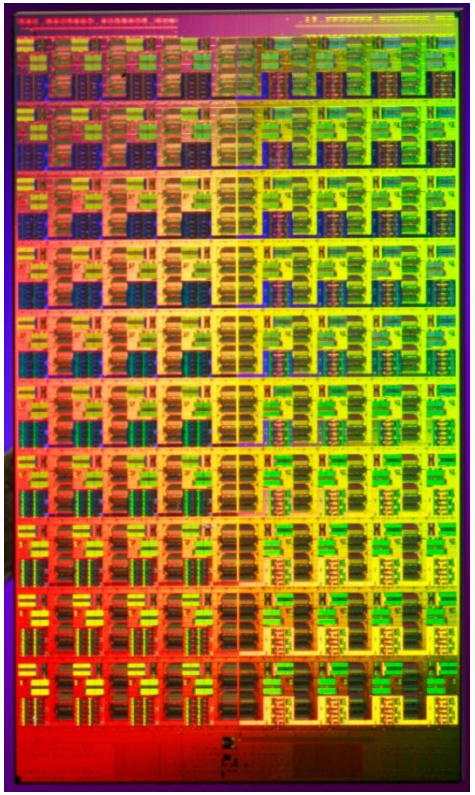
The Future

A future multicore scenario

- It's the year **2018**
- Intel is running a **15nm process**
- CPUs have **hundreds of cores**

There are many sources of asymmetry

- Cores **regularly overheat**
- Manufacturing defects result in **different frequencies**
- Nonuniform access to **memory controllers**



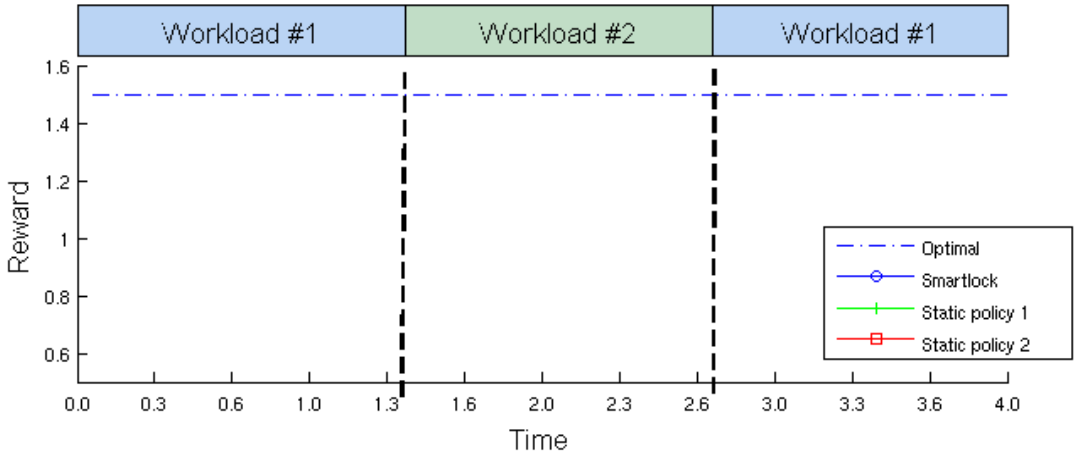
How can a programmer take full advantage of this hardware?

One answer: let **machine learning help manage complexity**

Smartlocks

A **mutex** combined with a **reinforcement learning agent**

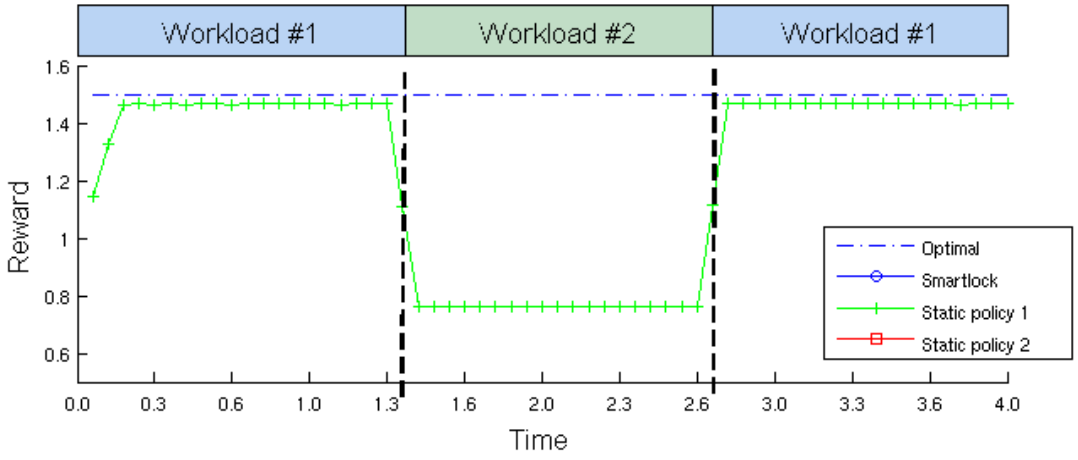
Learns to resolve contention by **adaptively prioritizing lock acquisition**



Smartlocks

A **mutex** combined with a **reinforcement learning agent**

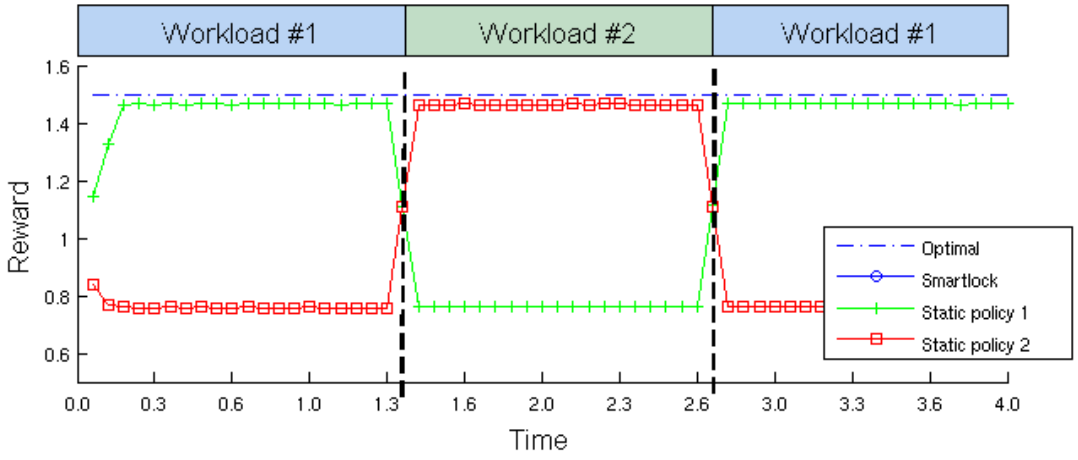
Learns to resolve contention by **adaptively prioritizing lock acquisition**



Smartlocks

A **mutex** combined with a **reinforcement learning agent**

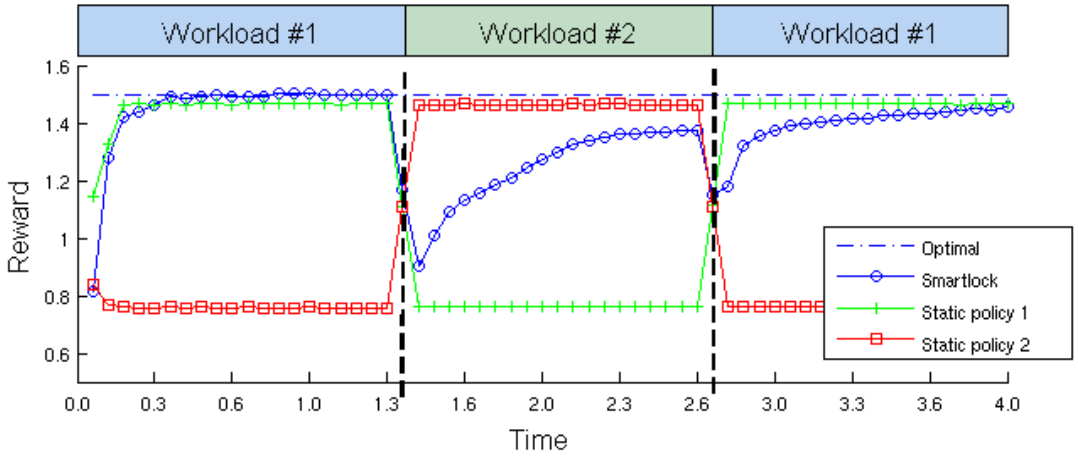
Learns to resolve contention by **adaptively prioritizing lock acquisition**



Smartlocks

A **mutex** combined with a **reinforcement learning agent**

Learns to resolve contention by **adaptively prioritizing lock acquisition**



Could be applied to resolve contention for different resources: **scheduler, disk, network, memory…**

ILEM + RL + Multicore

Smartlocks are currently a **model-free method**

Better: **learn a factored causal model of the current workload!**

Future work: **scale up to meet this challenge**

More generally: **RL + ML for managing complex systems**

Conclusions

Conclusions

- **Creating compelling agents touches many different problems**
  - Perception, sys id, state estimation, planning, representations…
- **Finding factored, causal structure in timeseries data is a general problem that is widely applicable**
  - Many possibilities for extended ILEM-type models
  - Structure might exist in data, states, or actions
  - Useful in routing, scheduling, optimization, inference…
  - A Bayesian view of domain-adaptive search is potentially powerful
- **Hierarchical Bayes is a useful lingua franca**
  - Can reason about uncertainty at many levels
  - Learning at multiple levels of abstraction can simplify problems
  - **A unified language for talking about policies, models, and state representations and uncertainty at every level**

Thank you!

The ILEM

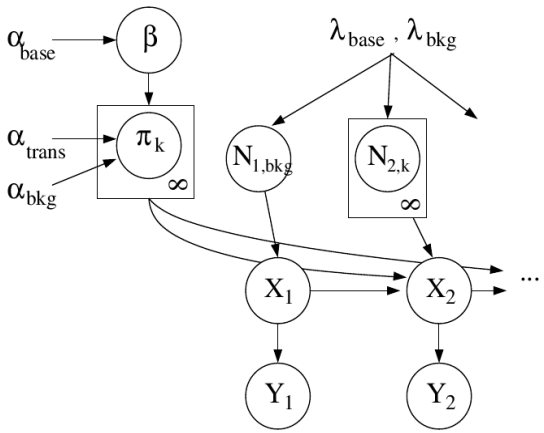
Assume there is a distribution over **infinite-by-infinite binary DBN**

Integrate them all out: results in a **nonparametric distribution**

Generative process

$$
\begin{aligned}
\beta &\sim \mathrm{GEM}(\alpha_{\mathrm{base}}) \\
\pi_k &\sim \mathrm{DP}(\alpha_{\mathrm{trans}}, \beta) \\
\pi_{\mathrm{bkg}} &\sim \mathrm{DP}(\alpha_{\mathrm{bkg}}, \beta) \\
N_{t,\mathrm{bkg}} &\sim \mathrm{Poisson}(\lambda_{\mathrm{bkg}}) \\
N_{t,k} &\sim \mathrm{Poisson}(\lambda_{\mathrm{base}}) \\
D_{t,k,i} &\sim \pi_k \\
C_{t,k} &= \{D_{t,k,i} : i \leq N_{t,k}\} \\
X_1 &= C_{1,\mathrm{bkg}} \\
X_t &= \bigcup_{k \in X_{t-1} \cup \{\mathrm{bkg}\}} C_{t,k}
\end{aligned}
$$

Graphical model



Favors
determinism and reuse

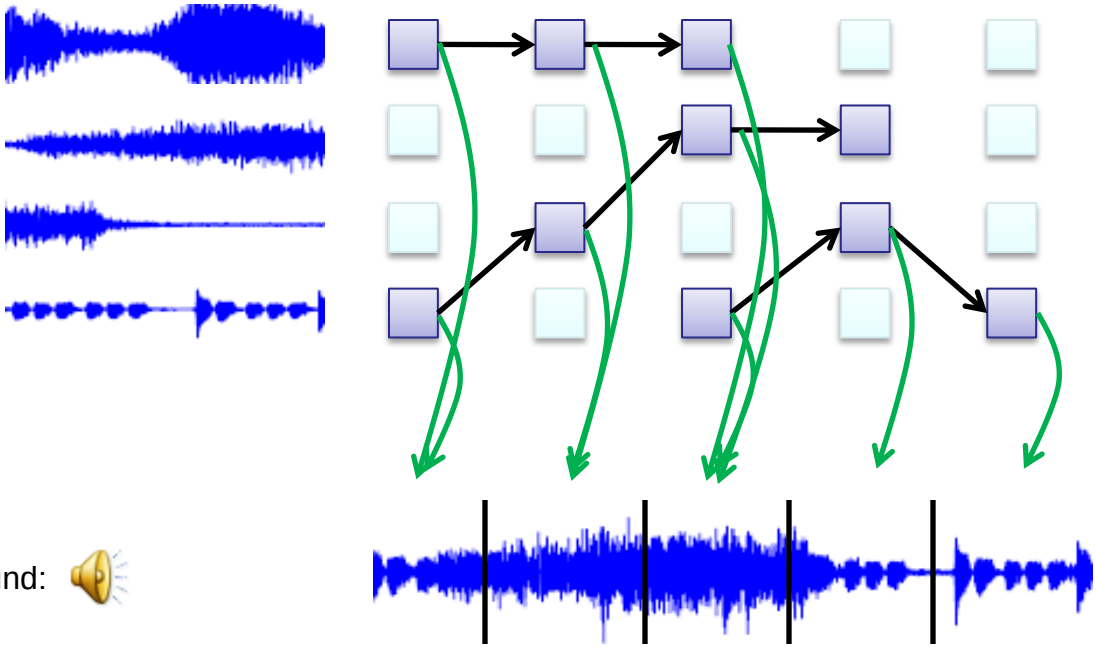
Can be informally thought of as

- **a factored Infinite HMM**
- **an infinite binary DBN**
- **the causal version of the IBP**

Theorems: related to the
HDP-HMM and **Noisy-OR DBNs**

Causal Factorization of Soundscapes

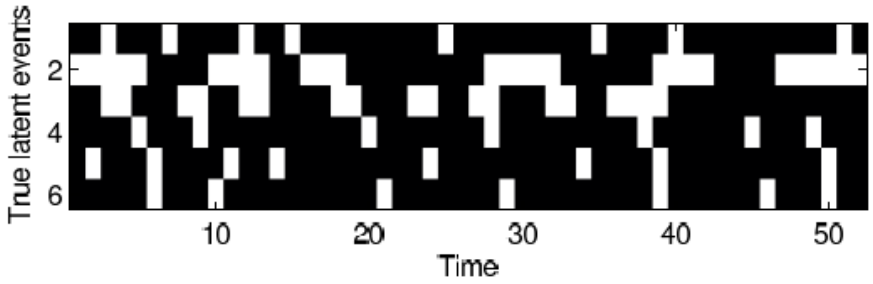
- **Causal** version of a blind-source separation problem
- **Linear-Gaussian** observation function
- Observations confounded in time and frequency domains



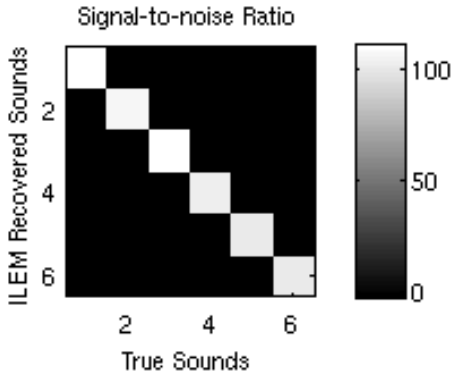
Original sound:

Causal Factorization of Soundscapes: Results

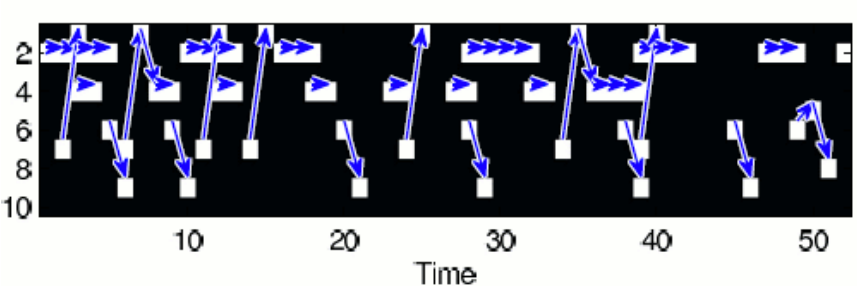
True events



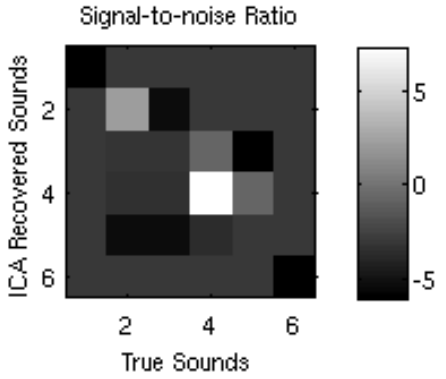
ILEM



Inferred events

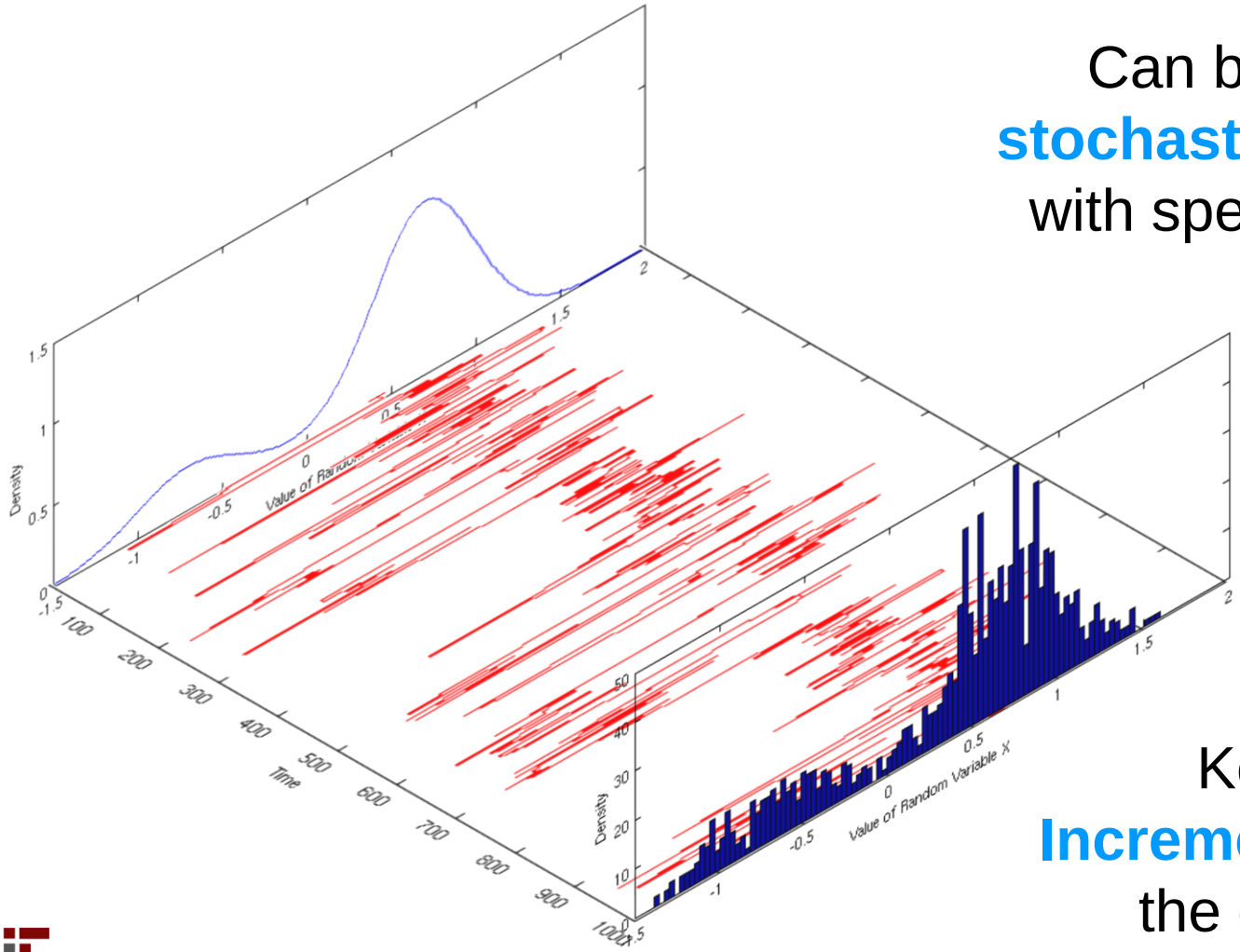


ICA



Recovered prototypical observations:

Generic MCMC Inference



Can be viewed as **stochastic local search** with special properties

Key concept: **Incremental changes** to the **current state**