Closing the Learning-Planning Loop with Predictive State Representations

Byron Boots
Sajid M. Siddiqi
Geoffrey J. Gordon

Select Lab

Carnegie Mellon University

Overview

Learning models of dynamical systems with actions

Overview

Learning models of dynamical systems with actions

Bringing system identification and reinforcement learning closer together

Overview

Learning models of dynamical systems with actions

Bringing system identification and reinforcement learning closer together

Predictive State Representations (PSRs)

Overview

Learning models of dynamical systems with actions

Bringing system identification and reinforcement learning closer together

Predictive State Representations (PSRs)

- more general than finite-dimensional POMDPs

Overview

Learning models of dynamical systems with actions

Bringing system identification and reinforcement learning closer together

Predictive State Representations (PSRs)

- more general than finite-dimensional POMDPs
- today: learning is closed form, statistically consistent

Overview

Learning models of dynamical systems with actions

Bringing system identification and reinforcement learning closer together

Predictive State Representations (PSRs)

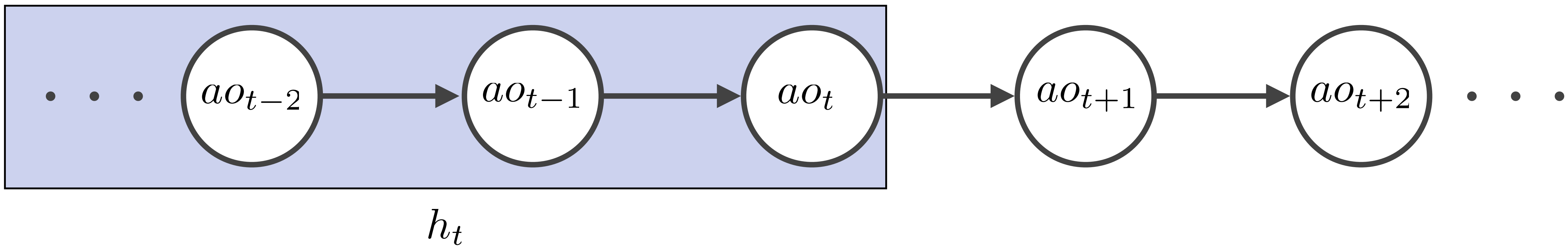
- more general than finite-dimensional POMDPs
- today: learning is closed form, statistically consistent
- evaluate learning by planning in the learned model.

Modeling a Dynamical System

$$\cdots \quad ao_{t-2} \rightarrow ao_{t-1} \rightarrow ao_t \rightarrow ao_{t+1} \rightarrow ao_{t+2} \quad \cdots$$

Given a sequence of actions and observations
from a partially observable system

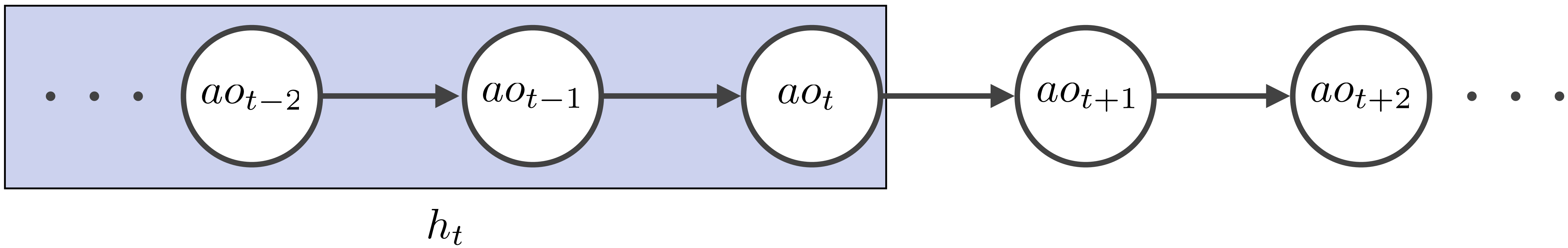
Modeling a Dynamical System



Naive Model: History

- updating is trivial
- harder: storage, prediction, ...

Modeling a Dynamical System



Naive Model: History

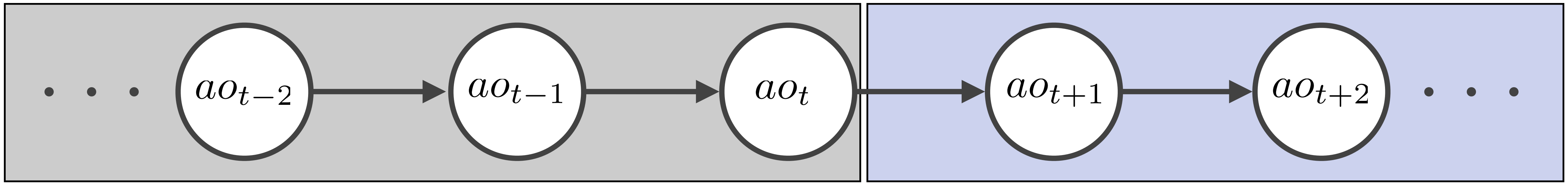
- updating is trivial
- harder: storage, prediction, ...

Want to learn something that is less naive

Insight:

The purpose of a dynamical system model is to predict the future

Modeling a Dynamical System



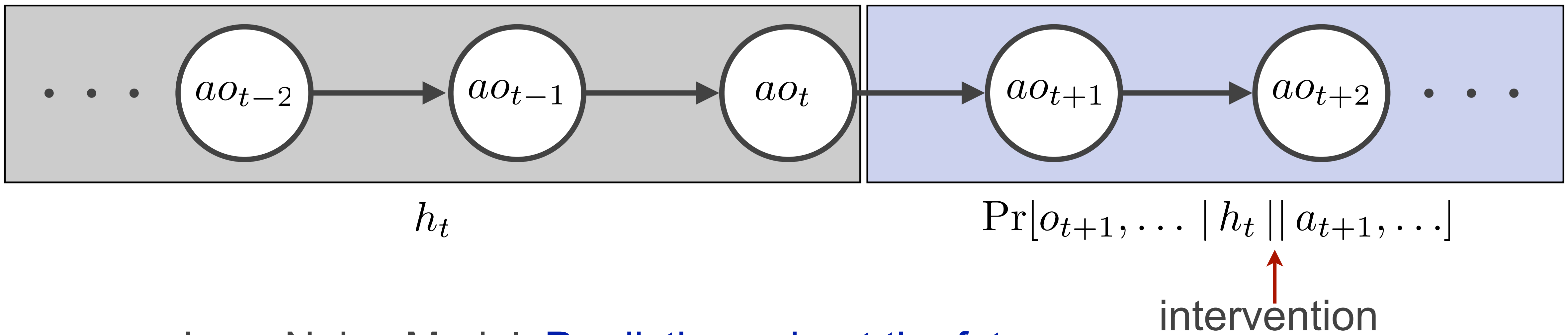
h_t

$\Pr[o_{t+1}, \ldots \mid h_t \mid\mid a_{t+1}, \ldots]$

intervention

Less Naive Model: Predictions about the future

Modeling a Dynamical System

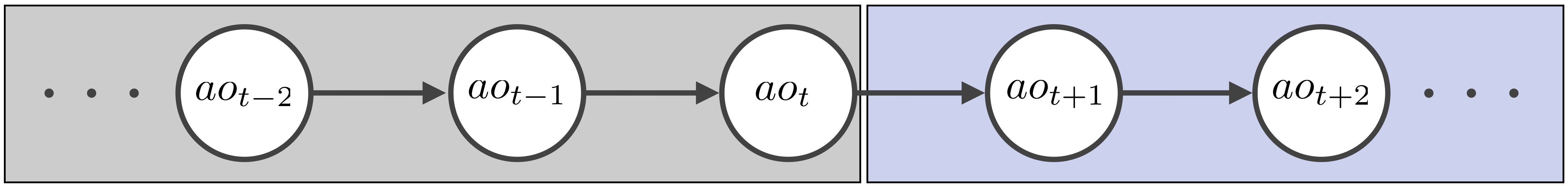


Less Naive Model: Predictions about the future

Test: ordered sequence of action observation pairs $\tau = a_1 o_1 \ldots a_k o_k$

Prediction of a test: $\Pr[\tau^O \mid h \,||\, \tau^A]$

Modeling a Dynamical System



h_t

$\Pr[o_{t+1}, \ldots \mid h_t \,||\, a_{t+1}, \ldots]$

intervention

Less Naive Model: Predictions about the future

Test: ordered sequence of action observation pairs $\tau = a_1 o_1 \ldots a_k o_k$

Prediction of a test: $\Pr[\tau^O \mid h \,||\, \tau^A]$

A Predictive State Representation (PSR) consists of the probabilities of all possible tests

- predicting is trivial
- harder: storage, updating, ...

PSRs: Storage

Q: Do we have to store all possible tests?

PSRs: Storage

Q: Do we have to store all possible tests?

A: No.

E.g., HMM: transition matrix T, observation probability matrix O

Intuition 1:
probabilities of tests can be computed as a linear function of state

$$\Pr[o_{t+k} \mid h_t] = \underline{O T^k} \, \underline{s(h_t)}$$

$$\begin{bmatrix} \Pr[o_{t+1} \mid h_t] \\ \vdots \\ \Pr[o_{t+k} \mid h_t] \end{bmatrix} = As(h_t)$$

Intuition 2:
HMM state can be determined exactly as a linear function of a finite set of test predictions

$$A^\dagger \begin{bmatrix} \Pr[o_{t+1} \mid h_t] \\ \vdots \\ \Pr[o_{t+k} \mid h_t] \end{bmatrix} = s(h_t)$$

PSRs: Storage

Q: Do we have to store all possible tests?

A: No.

Linear PSRs:

It is possible to predict all tests as linear combinations of predictions of a set of core tests
(e.g. HMMs, POMDPs)

PSRs: Storage

Q: Do we have to store all possible tests?

A: No.

Linear PSRs:

It is possible to predict all tests as linear combinations of predictions of a set of core tests (e.g. HMMs, POMDPs)

Let $Q = \{q_i\}$ be a set of tests

PSRs: Storage

Q: Do we have to store all possible tests?

A: No.

Linear PSRs:

It is possible to predict all tests as linear combinations of predictions of a set of core tests (e.g. HMMs, POMDPs)

Let $Q = \{q_i\}$ be a set of tests

Then $Q(h) = [\Pr[q_1^O \mid h \,||\, q_1^A], ..., \Pr[q_{|Q|}^O \mid h \,||\, q_{|Q|}^A]]$ is a prediction vector for these tests

PSRs: Storage

Q: Do we have to store all possible tests?

A: No.

Linear PSRs:

It is possible to predict all tests as linear combinations of predictions of a set of core tests (e.g. HMMs, POMDPs)

Let $Q = \{q_i\}$ be a set of tests

Then $Q(h) = [\Pr[q_1^O \mid h \,||\, q_1^A], \ldots, \Pr[q_{|Q|}^O \mid h \,||\, q_{|Q|}^A]]$ is a prediction vector for these tests

Q is a core set of tests, *iff* for any test τ: $\Pr[\tau^O \mid h \,||\, \tau^A] = r_\tau^\intercal Q(h)$

PSRs: Storage

Q: Do we have to store all possible tests?

A: No.

Linear PSRs:

It is possible to predict all tests as linear combinations of predictions of a set of core tests (e.g. HMMs, POMDPs)

Let $Q = \{q_i\}$ be a set of tests

Then $Q(h) = [\Pr[q_1^O \mid h \,||\, q_1^A], ..., \Pr[q_{|Q|}^O \mid h \,||\, q_{|Q|}^A]]$ is a prediction vector for these tests

Q is a core set of tests, *iff* for any test τ: $\Pr[\tau^O \mid h \,||\, \tau^A] = r_\tau^\intercal Q(h)$

PSR state is a prediction vector of core tests

PSRs: Updating State

After taking action a and observing o we can update $Q(h_t)$ recursively:

PSRs: Updating State

After taking action a and observing o we can update $Q(h_t)$ recursively:

recall: $\Pr[\tau^O \,|\, h \,||\, \tau^A] = r_\tau^\mathsf{T} Q(h)$

PSRs: Updating State

After taking action a and observing o we can update $Q(h_t)$ recursively:

recall: $\Pr[\tau^O \mid h \,||\, \tau^A] = r_\tau^\mathsf{T} Q(h)$

$$\Pr[\underline{o,}\, o_1, \ \ldots, o_k \mid h_t \,||\, \underline{a,}\, a_1, \ldots, a_k] \quad = \quad r_{aoq_i}^\mathsf{T} \cdot Q(h_t)$$

prediction of core tests

prob. of test $\underline{ao}q_i$

PSRs: Updating State

After taking action a and observing o we can update $Q(h_t)$ recursively:

recall: $\Pr[\tau^O \mid h \,||\, \tau^A] = r_\tau^\top Q(h)$

prediction of core tests

$$\Pr[\underline{o,} o_1, \; \ldots, o_k \mid h_t \,||\, \underline{a,} a_1, \ldots, a_k] \quad = \quad r_{aoq_i}^\top \cdot Q(h_t)$$

prob. of test $\underline{a}oq_i$

Let M_{ao} be the matrix with rows $r_{aoq_i}^\top$

PSRs: Updating State

After taking action a and observing o we can update $Q(h_t)$ recursively:

recall: $\Pr[\tau^O \mid h \,||\, \tau^A] = r_\tau^\top Q(h)$

$$\Pr[\underline{o,}\, o_1, \;\ldots, o_k \mid h_t \,||\, \underline{a,}\, a_1, \ldots, a_k] \quad = \quad r_{aoq_i}^\top \cdot Q(h_t)$$

prediction of core tests

prob. of test $\underline{ao}q_i$

Let M_{ao} be the matrix with rows $r_{aoq_i}^\top$

Then we can use Bayes' Rule to update state recursively:

$$Q(hao) = \frac{M_{ao}Q(h)}{\Pr[o \mid h \,||\, a]}$$

M_{ao} is a linear transition matrix (one for each action-observation pair)

PSRs: Updating State

After taking action a and observing o we can update $Q(h_t)$ recursively:

recall: $\Pr[\tau^O \,|\, h \,||\, \tau^A] = r_\tau^\top Q(h)$

$$\Pr[\underline{o,}\, o_1, \ \ldots, o_k \,|\, h_t \,||\, \underline{a,}\, a_1, \ldots, a_k] \quad = \quad r_{aoq_i}^\top \cdot Q(h_t)$$

prediction of core tests

prob. of test $\underline{ao}q_i$

Let M_{ao} be the matrix with rows $r_{aoq_i}^\top$

Then we can use Bayes' Rule to update state recursively:

$$Q(hao) = \frac{M_{ao}Q(h)}{\Pr[o \,|\, h \,||\, a]} = \frac{M_{ao}Q(h)}{m_\infty^\top M_{ao} Q(h)}$$

M_{ao} is a linear transition matrix (one for each action-observation pair)

$m_\infty^\top$ is a normalizing vector

PSRs

In summary:

- PSR state is vector of predictions over a small set of core tests
- PSRs can predict any test as a linear function of state
- PSRs update state by applying a matrix M_{ao} and then renormalizing

Previous Work

Would like to learn a PSR from sequences of observations and actions

Previous Work

Would like to learn a PSR from sequences of observations and actions

Discovery problem: find minimal set of core tests

Learning problem: find PSR parameters

Previous Work

Would like to learn a PSR from sequences of observations and actions

Discovery problem: find minimal set of core tests

Learning problem: find PSR parameters

In practice, finding a large set of tests capturing elements of the system that we want to model is easy

but, finding a minimal set of core tests is hard

Previous Work

Previous Solutions: perform an incremental combinatorial search
to try to grow a minimal core set

learn M_{ao} etc. by regression

[Wolfe, James, Singh, 2005], [Wiewiora, 2005], [Bowling et. al, 2006]

in practice require a huge amount of data

Previous Work

An alternative approach? Subspace Identification

- Spectral algorithms for identifying
  - Linear Dynamical Systems
    [Van Overschee, De Moor, 1996], [Soatto,Chiuso, 2001], [Katayama, 2005], ...
  - Hidden Markov Models
    [Hsu, Kakade, Zhang, 2008]
  - Reduced-Rank Hidden Markov Models
    [Siddiqi, Boots, Gordon, 2010]
- Closed-form, no local optima, statistically consistent

Previous Work

An alternative approach? Subspace Identification

- Spectral algorithms for identifying
  - Linear Dynamical Systems
    [Van Overschee, De Moor, 1996], [Soatto,Chiuso, 2001], [Katayama, 2005], ...
  - Hidden Markov Models
    [Hsu, Kakade, Zhang, 2008]
  - Reduced-Rank Hidden Markov Models
    [Siddiqi, Boots, Gordon, 2010]
- Closed-form, no local optima, statistically consistent

matrix factorization instead of combinatorial search
to solve the discovery problem

Today

This work:

- Specify a spectral learning (subspace identification) algorithm for PSRs

Today

This work:

- Specify a spectral learning (subspace identification) algorithm for PSRs
- Extend algorithm to use features of tests and histories

Today

This work:

- Specify a spectral learning (subspace identification) algorithm for PSRs
- Extend algorithm to use features of tests and histories
- Apply algorithm to high dimensional data

Today

This work:

- Specify a spectral learning (subspace identification) algorithm for PSRs
- Extend algorithm to use features of tests and histories
- Apply algorithm to high dimensional data
- Plan in learned model

Outline

1. Preliminaries & PSRs
2. Subspace Identification
3. Learning PSRs by Subspace ID
4. Extending Learning to use Features
5. Experimental Results

Subspace Identification (SSID)

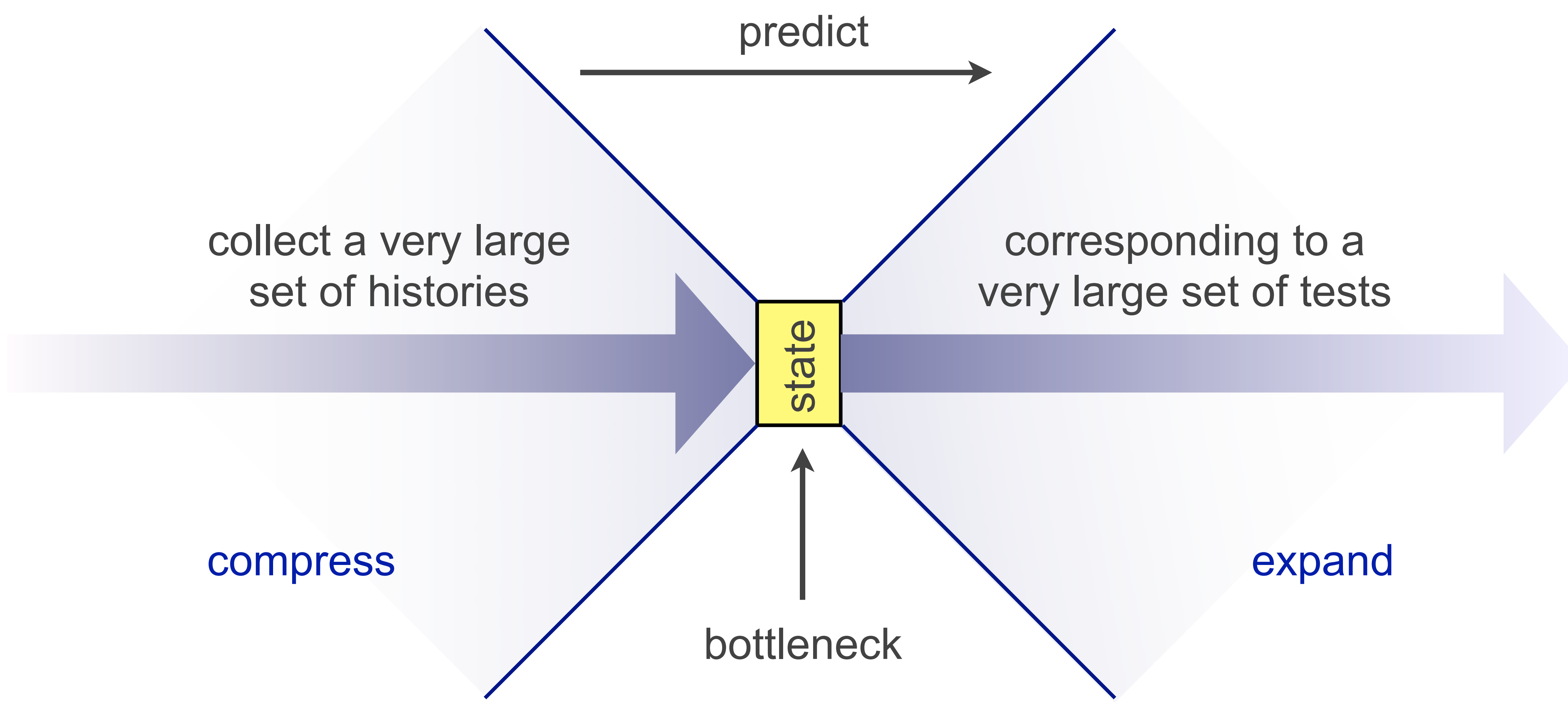
strategy for system identification:

collect a very large set of histories

corresponding to a very large set of tests

Subspace Identification (SSID)

strategy for system identification:



Subspace Identification (SSID)

Insights:

- All necessary information for predicting future from past is in the covariance of past & future
- Bottleneck = rank constraint (SVD)

Subspace Identification (SSID)

Insights:

- All necessary information for predicting future from past is in the covariance of past & future
- Bottleneck = rank constraint (SVD)

Benefits:

- Easy to estimate covariance
- SVD robust, closed form
- Statistically consistent, computationally efficient

Outline

1. Preliminaries & PSRs

2. Subspace Identification

3. Learning PSRs by Subspace ID

4. Extending Learning to use Features

5. Experimental Results

PSR Parameters

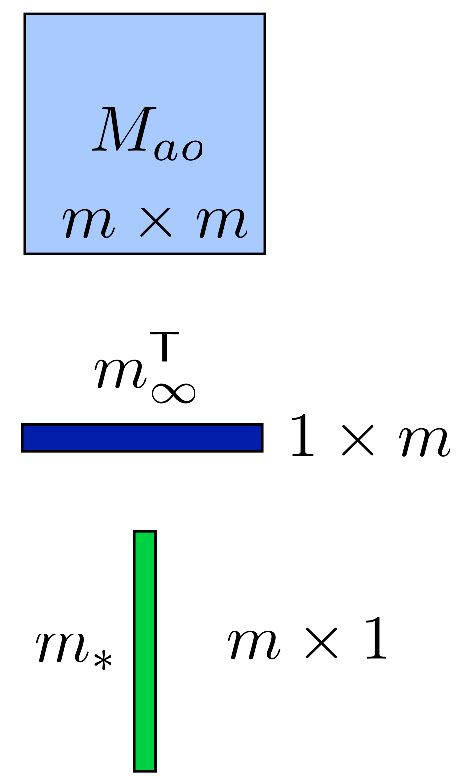
n : cardinality of large set of tests and histories

m : cardinality of minimal core tests

$M_{ao}: m \times m$ transition matrix

$m_\infty^\mathsf{T}: 1 \times m$ normalization vector

$m_*: m \times 1$ vector of prior probabilities of tests



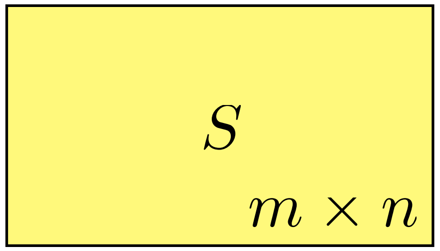
PSR Parameters

n : cardinality of large set of tests and histories

m : cardinality of minimal core tests

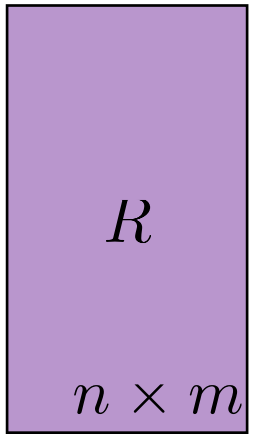
$S : m \times n$ matrix of minimal core test probabilities

$S_{i,j} = \Pr[q_i^O \mid H_j \mid\mid q_i^A]$



$R : n \times m$ matrix of linear prediction functions

$R_{i,j} = \Pr[\tau_i^O \mid q_j^O \mid\mid q_j^A, \tau_i^A]$



Spectral Learning for PSR Parameters

Idea: Recover PSR parameters from observable joint probabilities of tests and histories

Spectral Learning for PSR Parameters

Idea: Recover PSR parameters from observable joint probabilities of tests and histories

Let $\mathcal{T}$ be some large core set of tests and $\mathcal{H}$ be a set of histories

1. Define

$$[P_{\mathcal{T},\mathcal{H}}]_{i,j} \equiv \Pr[\tau_i^O, H_j \,||\, \tau_i^A]$$

$$[P_{\mathcal{T},ao,\mathcal{H}}]_{i,j} \equiv \Pr[\tau_i^O, o, H_j \,||\, a, \tau_i^A]$$

Spectral Learning for PSR Parameters

Idea: Recover PSR parameters from observable joint probabilities of tests and histories

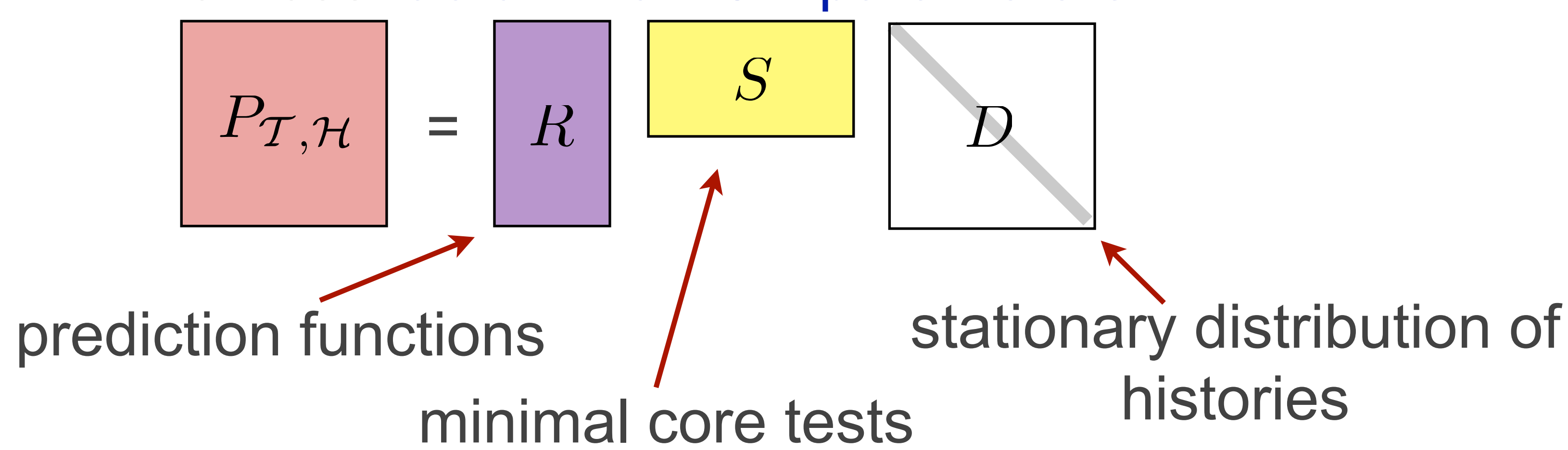
Let $\mathcal{T}$ be some large core set of tests and $\mathcal{H}$ be a set of histories

1. Define

$$[P_{\mathcal{T},\mathcal{H}}]_{i,j} \equiv \Pr[\tau_i^O, H_j \,||\, \tau_i^A]$$

$$[P_{\mathcal{T},ao,\mathcal{H}}]_{i,j} \equiv \Pr[\tau_i^O, o, H_j \,||\, a, \tau_i^A]$$

2. Matrices factor into PSR parameters



sense
learn
act

Spectral Learning for PSR Parameters

Idea: Recover PSR parameters from observable joint probabilities of tests and histories

Let $\mathcal{T}$ be some large core set of tests and $\mathcal{H}$ be a set of histories

1. Define

$$[P_{\mathcal{T},\mathcal{H}}]_{i,j} \equiv \Pr[\tau_i^O, H_j \,||\, \tau_i^A]$$

$$[P_{\mathcal{T},ao,\mathcal{H}}]_{i,j} \equiv \Pr[\tau_i^O, o, H_j \,||\, a, \tau_i^A]$$

2. Matrices factor into PSR parameters

$$P_{\mathcal{T},\mathcal{H}} = R \quad S \quad D$$

prediction functions

minimal core tests

stationary distribution of histories

$$P_{\mathcal{T},ao,\mathcal{H}} = R \quad M_{ao} \quad S \quad D$$

transition matrix

Spectral Learning for PSR Parameters

Pick a U s.t. $U^\top R$ is invertible

Spectral Learning for PSR Parameters

Pick a U s.t. $U^\top R$ is invertible

and given (from previous slide):

$$P_{\mathcal{T},\mathcal{H}} = RSD \qquad P_{\mathcal{T},ao,\mathcal{H}} = RM_{ao}SD$$

Spectral Learning for PSR Parameters

Pick a U s.t. $U^\top R$ is invertible

and given (from previous slide):

$$P_{\mathcal{T},\mathcal{H}} = RSD \qquad P_{\mathcal{T},ao,\mathcal{H}} = RM_{ao}SD$$

$$B_{ao} \equiv (U^\top P_{\mathcal{T},ao,\mathcal{H}})(U^\top P_{\mathcal{T},\mathcal{H}})^\dagger = (U^\top R)M_{ao}(U^\top R)^{-1}$$

Spectral Learning for PSR Parameters

Pick a U s.t. $U^\top R$ is invertible

and given (from previous slide):

$$P_{\mathcal{T},\mathcal{H}} = RSD \qquad P_{\mathcal{T},ao,\mathcal{H}} = RM_{ao}SD$$

$$B_{ao} \equiv (U^\top P_{\mathcal{T},ao,\mathcal{H}})(U^\top P_{\mathcal{T},\mathcal{H}})^\dagger = (U^\top R)M_{ao}(U^\top R)^{-1}$$

similarity transform
of the minimal PSR
transition matrix

Spectral Learning for PSR Parameters

Pick a U s.t. $U^\top R$ is invertible

and given (from previous slide):

$$P_{\mathcal{T},\mathcal{H}} = RSD \qquad P_{\mathcal{T},ao,\mathcal{H}} = RM_{ao}SD$$

$$B_{ao} \equiv (U^\top P_{\mathcal{T},ao,\mathcal{H}})(U^\top P_{\mathcal{T},\mathcal{H}})^\dagger = (U^\top R) M_{ao} (U^\top R)^{-1}$$

similarity transform
of the minimal PSR
transition matrix

other parameters can be recovered up to a linear transform from observable matrices as well

$b_\infty^\top = m_\infty^\top (U^\top R)^{-1}$ normalizing vector

$b_* = (U^\top R) m_*$ initial test predictions

Spectral Learning for PSR Parameters

Q: Why is this important?

Spectral Learning for PSR Parameters

Q: Why is this important?

We can perform inference with transformed parameters!

$$\Pr[o_1, \ldots, o_k \,||\, a_1, \ldots a_k]$$

Spectral Learning for PSR Parameters

Q: Why is this important?

We can perform inference with transformed parameters!

$$
\begin{aligned}
&\Pr[o_1, \dots, o_k \,||\, a_1, \dots a_k] \\
=&m_\infty^\intercal M_{a_k o_k} \dots M_{a_1 o_1} m_*
\end{aligned}
$$

Spectral Learning for PSR Parameters

Q: Why is this important?

We can perform inference with transformed parameters!

$$
\begin{aligned}
&\Pr[o_1, \ldots, o_k \,||\, a_1, \ldots a_k] \\
=&m_\infty^\top M_{a_k o_k} \ldots M_{a_1 o_1} m_* \\
=&m_\infty^\top \underline{(U^\top R)^{-1}(U^\top R)} M_{a_k o_k} \underline{(U^\top R)^{-1}} \ldots \underline{(U^\top R)} M_{a_1 o_1} \underline{(U^\top R)^{-1}(U^\top R)} m_*
\end{aligned}
$$

Spectral Learning for PSR Parameters

Q: Why is this important?

We can perform inference with transformed parameters!

$$
\begin{aligned}
&\Pr[o_1, \ldots, o_k \,||\, a_1, \ldots a_k] \\
=& m_\infty^\top M_{a_k o_k} \ldots M_{a_1 o_1} m_* \\
=& \underline{m_\infty^\top (U^\top R)^{-1}} \, \underline{(U^\top R) M_{a_k o_k} (U^\top R)^{-1}} \ldots \underline{(U^\top R) M_{a_1 o_1} (U^\top R)^{-1}} \, \underline{(U^\top R) m_*} \\
=& b_\infty^\top B_{a_k o_k} \ldots B_{a_1 o_1} b_*
\end{aligned}
$$

Spectral Learning for PSR Parameters

Q: Why is this important?

We can perform inference with transformed parameters!

$$
\begin{aligned}
&\Pr[o_1, \ldots, o_k \,||\, a_1, \ldots a_k] \\
=& m_\infty^\top M_{a_k o_k} \ldots M_{a_1 o_1} m_* \\
=& m_\infty^\top (U^\top R)^{-1} (U^\top R) M_{a_k o_k} (U^\top R)^{-1} \ldots (U^\top R) M_{a_1 o_1} (U^\top R)^{-1} (U^\top R) m_* \\
=& b_\infty^\top B_{a_k o_k} \ldots B_{a_1 o_1} b_*
\end{aligned}
$$

We can also filter, predict, and plan with these parameters

(the linear transforms cancel)

we can parameterize PSRs in terms of observable quantities

Spectral Learning for PSR Parameters

The algorithm:

1. Look at triples $\langle \mathcal{T}, ao, \mathcal{H} \rangle$ in the data and estimate joint probabilities: $\widehat{P}_{\mathcal{T},\mathcal{H}}$ and $\widehat{P}_{\mathcal{T},ao,\mathcal{H}}$

2. Compute SVD of $\widehat{P}_{\mathcal{T},\mathcal{H}}$ and take the left singular vectors as $\widehat{U}$

3. Find transformed PSR parameters e.g. $\widehat{B}_{ao} \equiv (\widehat{U}^{\intercal}\widehat{P}_{\mathcal{T},ao,\mathcal{H}})(\widehat{U}^{\intercal}\widehat{P}_{\mathcal{T},\mathcal{H}})^{\dagger}$

Spectral Learning for PSR Parameters

The algorithm:

1. Look at triples $\langle \mathcal{T}, ao, \mathcal{H} \rangle$ in the data and estimate joint probabilities: $\widehat{P}_{\mathcal{T},\mathcal{H}}$ and $\widehat{P}_{\mathcal{T},ao,\mathcal{H}}$

2. Compute SVD of $\widehat{P}_{\mathcal{T},\mathcal{H}}$ and take the left singular vectors as $\widehat{U}$

3. Find transformed PSR parameters e.g. $\widehat{B}_{ao} \equiv (\widehat{U}^{\intercal}\underline{\widehat{P}_{\mathcal{T},ao,\mathcal{H}}})(\widehat{U}^{\intercal}\underline{\widehat{P}_{\mathcal{T},\mathcal{H}}})^{\dagger}$

as data increases,
estimates converge to true joint probs.

transformed parameter estimates are consistent

Spectral Learning for PSR Parameters

Transformed parameters allow accurate
PSR inference, filtering, prediction, planning
(other terms cancel)

Spectral Learning for PSR Parameters

Transformed parameters allow accurate
PSR inference, filtering, prediction, planning
(other terms cancel)

Learning is closed form, statistically consistent

Spectral Learning for PSR Parameters

Transformed parameters allow accurate
PSR inference, filtering, prediction, planning
(other terms cancel)

Learning is closed form, statistically consistent

A k-dimensional PSR is considerably more expressive than a k-state POMDP

Two recent HMM learning algorithms that outperform previous methods are special cases of this PSR algorithm

[Hsu et al., 2009], [Siddiqi et al., 2010]

Outline

1. Preliminaries & PSRs
2. Subspace Identification
3. Learning PSRs by Subspace ID
4. Extending Learning to use Features
5. Experimental Results

Spectral Learning with Features

In the real-world, PSR learning algorithms often need to see a lot of tests and histories to recover parameters

Spectral Learning with Features

In the real-world, PSR learning algorithms often need to see a lot of tests and histories to recover parameters

Solution: Use features of tests and features of histories

- can use a small set of features in place of a larger set of tests and a larger set of histories
- selection of features allows us to incorporate expert knowledge

Spectral Learning with Features

In the real-world, PSR learning algorithms often need to see a lot of tests and histories to recover parameters

Solution: Use features of tests and features of histories

- can use a small set of features in place of a larger set of tests and a larger set of histories
- selection of features allows us to incorporate expert knowledge

We redefine:

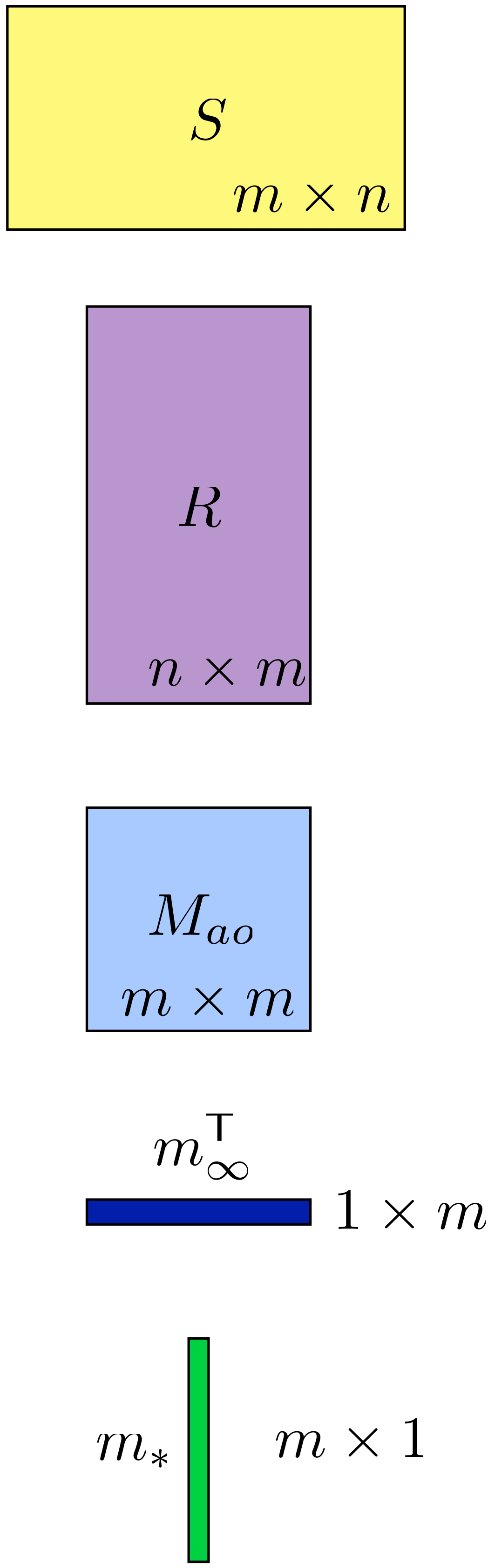
$$[P_{\mathcal{T},\mathcal{H}}]_{i,j} = \mathbb{E}(\phi_i^{\mathcal{T}}(\tau^O) \cdot \phi_j^{\mathcal{H}}(h) \,||\, \tau^A)$$

$$[P_{\mathcal{T},ao,\mathcal{H}}]_{i,j} = \mathbb{E}(\phi_i^{\mathcal{T}}(\tau^O) \cdot \phi_j^{\mathcal{H}}(h) \cdot \delta(o) \,||\, a, \tau^A)$$

PSR Parameters

n : cardinality of large set of tests and histories

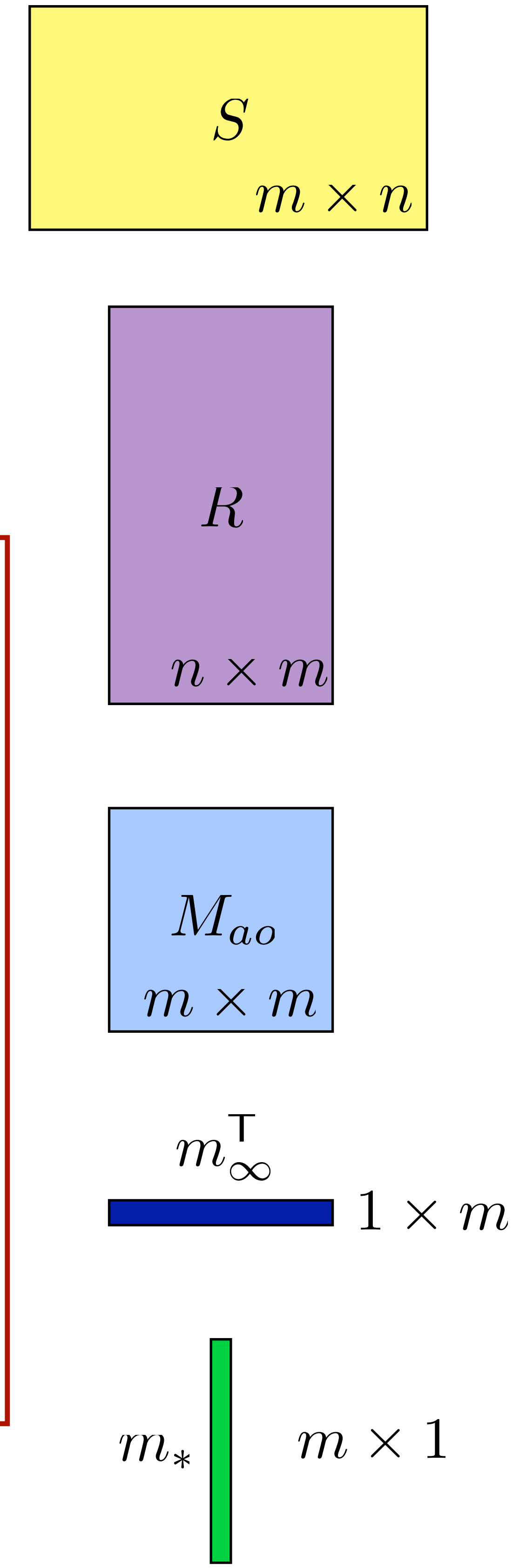
m : cardinality of minimal core tests



PSR Parameters

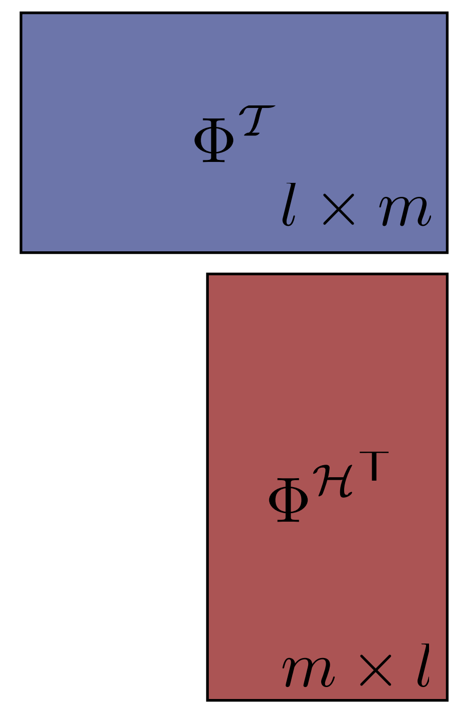
n : cardinality of large set of tests and histories

m : cardinality of minimal core tests

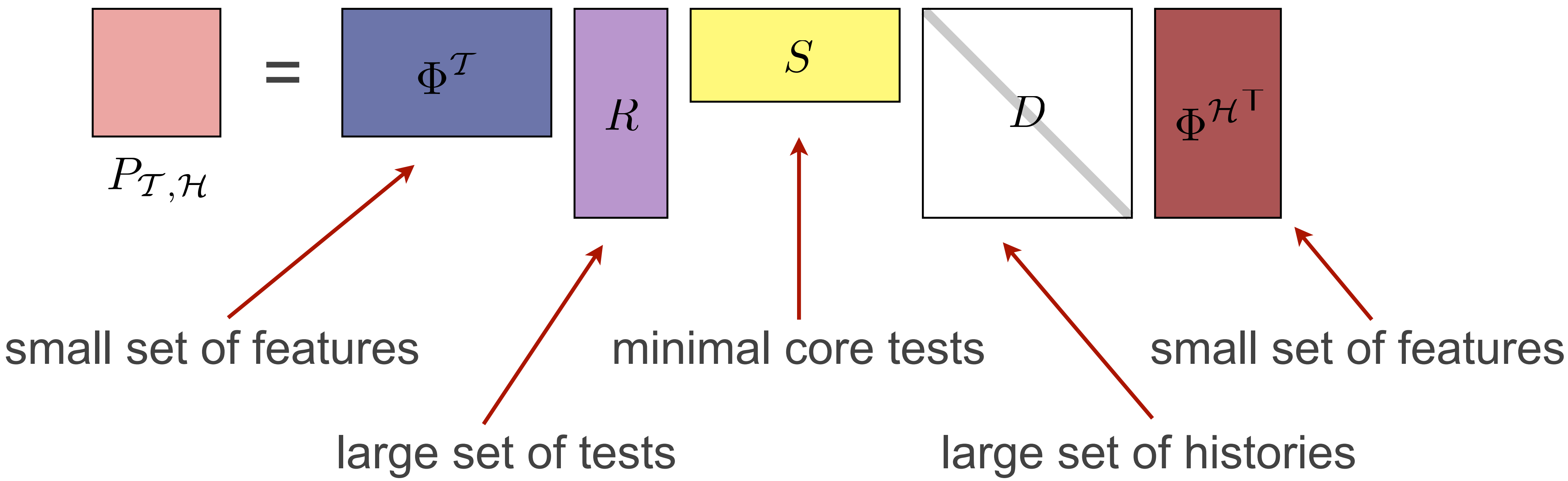


$\Phi^{\mathcal{T}}$: $l \times m$ features of tests

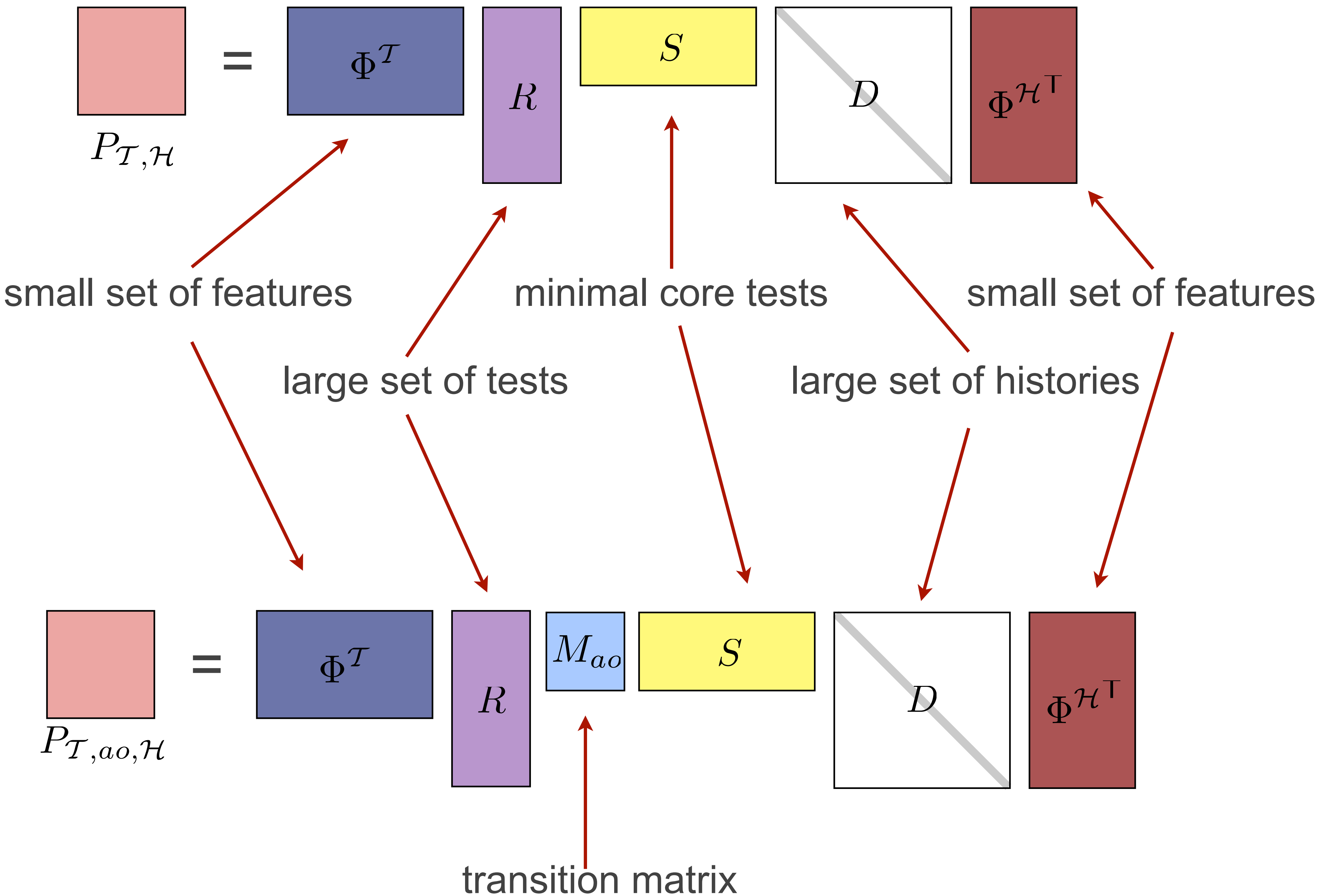
$\Phi^{\mathcal{H}^\top}$: $m \times l$ features of histories



Spectral Learning with Features



Spectral Learning with Features



Spectral Learning with Features

$P_{\mathcal{T},\mathcal{H}} = \Phi^{\mathcal{T}} R S D \Phi^{\mathcal{H}}$ $\quad$ $P_{\mathcal{T},ao,\mathcal{H}} = \Phi^{\mathcal{T}} R M_{ao} S D \Phi^{\mathcal{H}}$ $\quad$ $U^{\top}\Phi^{\mathcal{T}} R$ is invertible

$$B_{ao} \equiv (U^{\top} P_{\mathcal{T},ao,\mathcal{H}})(U^{\top} P_{\mathcal{T},\mathcal{H}})^{\dagger} = (U^{\top}\Phi^{\mathcal{T}} R) M_{ao} (U^{\top}\Phi^{\mathcal{T}} R)^{-1}$$

similarity transform
of the minimal PSR
transition matrix

other parameters can be recovered up to a linear transform as well

Spectral Learning with Features

$$P_{\mathcal{T},\mathcal{H}} = \Phi^{\mathcal{T}} R S D \Phi^{\mathcal{H}} \qquad P_{\mathcal{T},ao,\mathcal{H}} = \Phi^{\mathcal{T}} R M_{ao} S D \Phi^{\mathcal{H}} \qquad U^{\top}\Phi^{\mathcal{T}} R \text{ is invertible}$$

$$B_{ao} \equiv (U^{\top} P_{\mathcal{T},ao,\mathcal{H}})(U^{\top} P_{\mathcal{T},\mathcal{H}})^{\dagger} = (U^{\top}\Phi^{\mathcal{T}} R) M_{ao} (U^{\top}\Phi^{\mathcal{T}} R)^{-1}$$

similarity transform
of the minimal PSR
transition matrix

other parameters can be recovered up to a linear transform as well

can use the exact same algorithm to
recover PSR parameters

Spectral Learning with Features

Possible to learn PSRs for continuous observation spaces

Use kernel density estimation to model distributions of observations: see paper for details.

Outline

1. Preliminaries & PSRs

2. Subspace Identification

3. Learning PSRs by Subspace ID

4. Extending Learning to use Features

5. Experimental Results

Experiments

Future of Robotics: Engineering + Learning

State of the art: lots of engineering, comparatively little learning

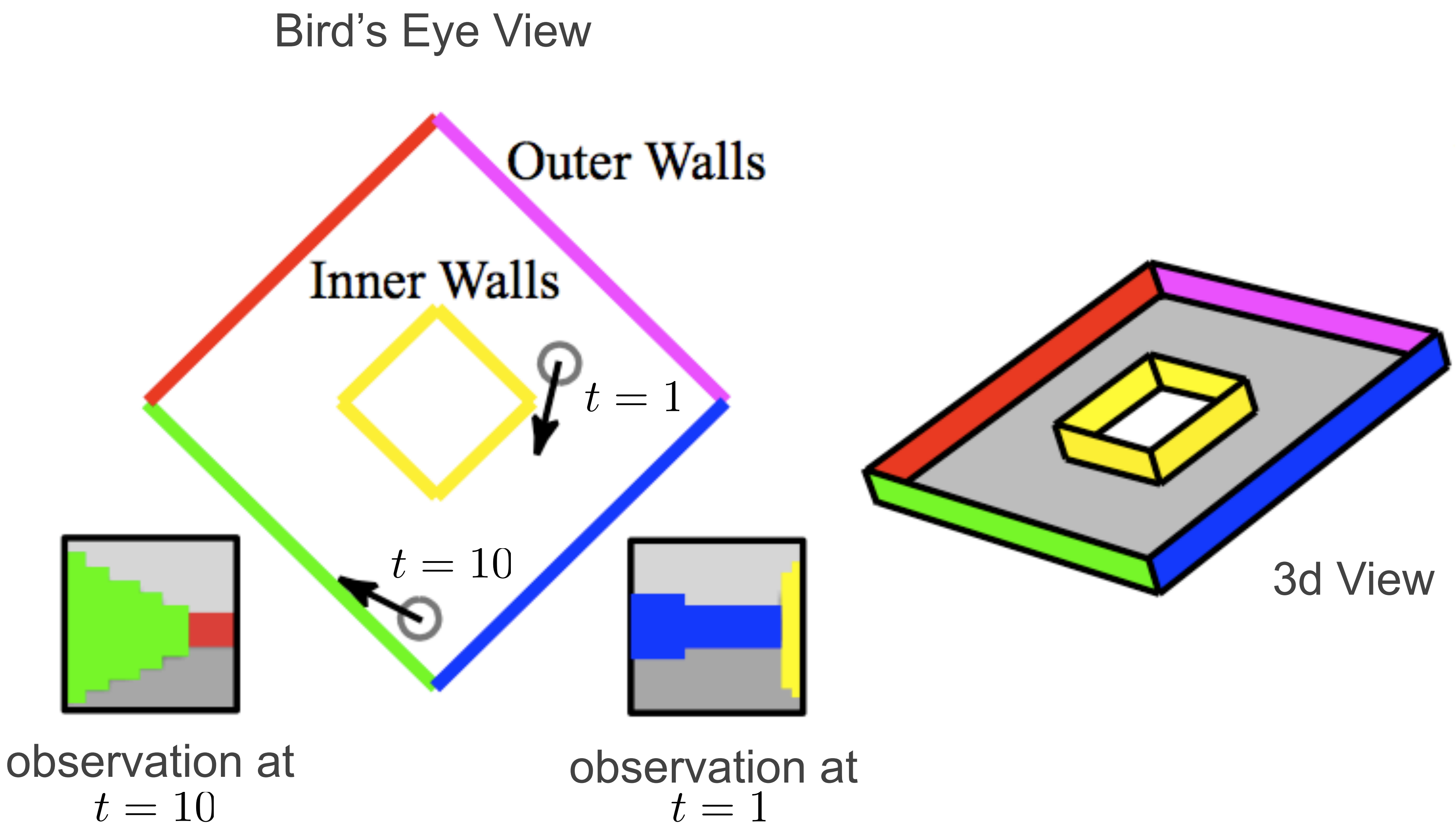
- learning algorithms are not capable of converting a huge amount of raw data to model of environment
- we believe our learning algorithm is much better than previous methods

Experiments

We test the capabilities of algorithm by dropping all engineering support:

The algorithm itself does all of the heavy lifting

Experimental Domain



agents can execute discrete but noisy translations and rotations

Experiments

Goal:

1. Learn a PSR model of how an agent's observations change as it takes actions in the environment

2. Test learned model by planning

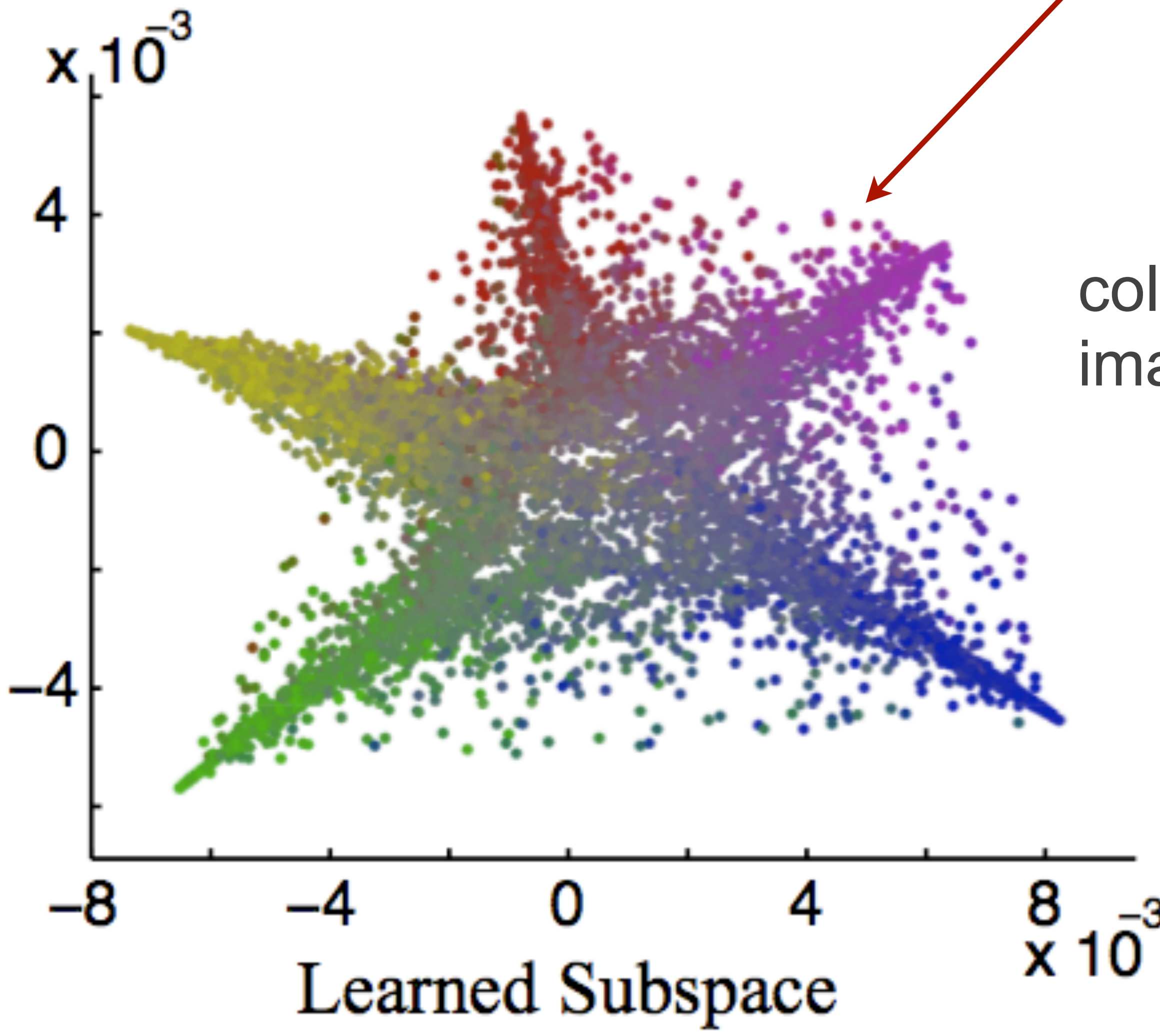
Learning

Fix a random policy execute it many times from many different starting positions.

- sample 10,000 start positions
- collect short execution traces of observations (16x16 images) and actions (1-6)
- use all the methods from this talk to learn a PSR model

Experimental Results

project histories into learned subspace

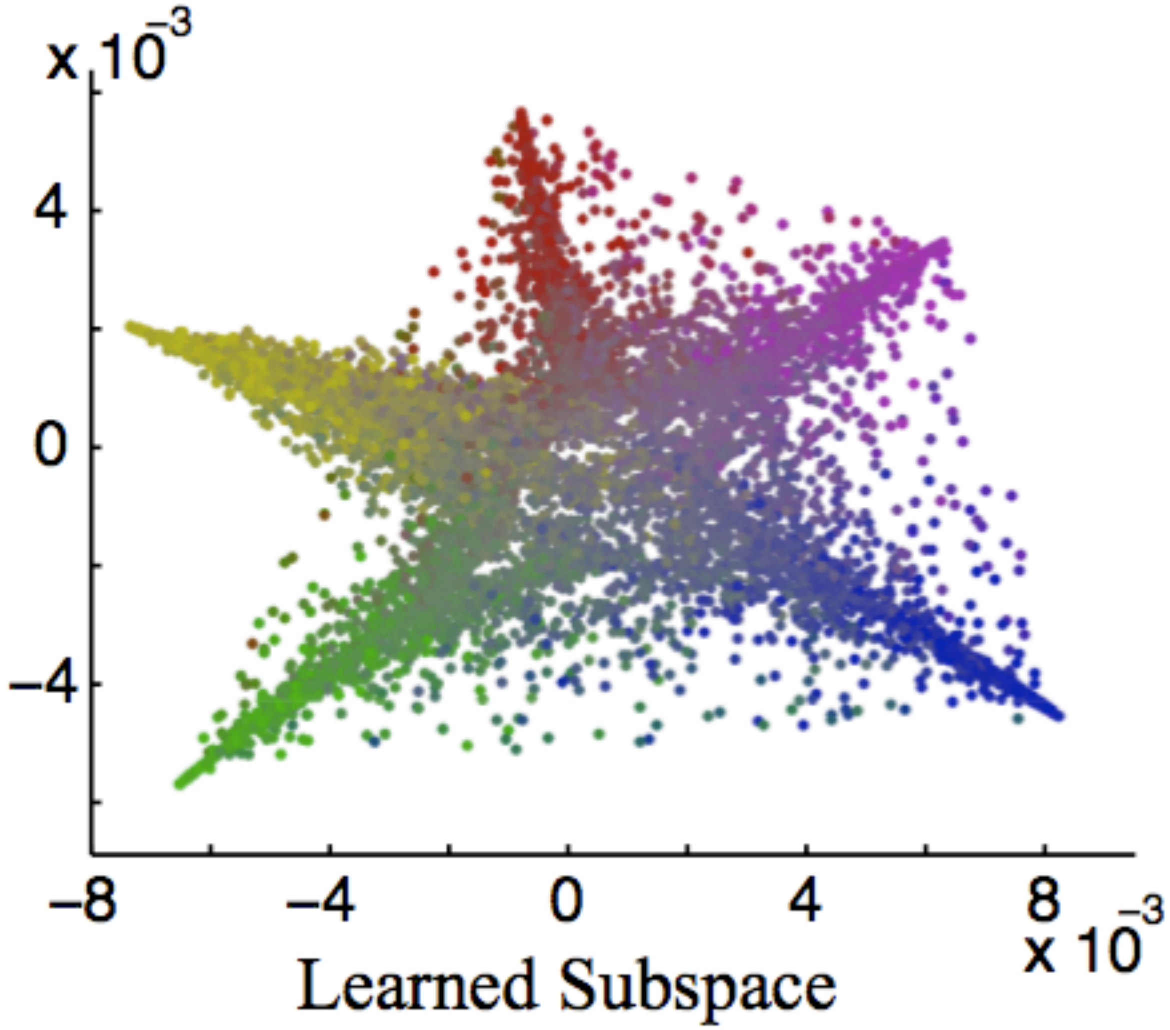


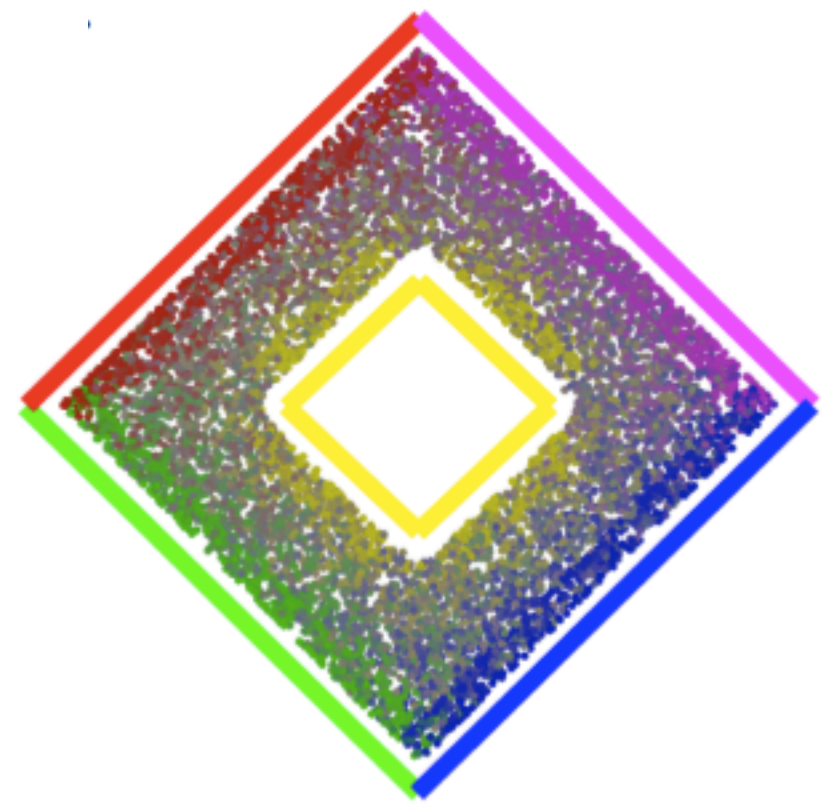
colors are the predominant color in image following history

2 dimensions of a learned 5 dimensional subspace

Experimental Results

map points back to geometric space





Planning

learned a reasonable PSR subspace

the best test of the learned model is planning

To plan, need a reward function:

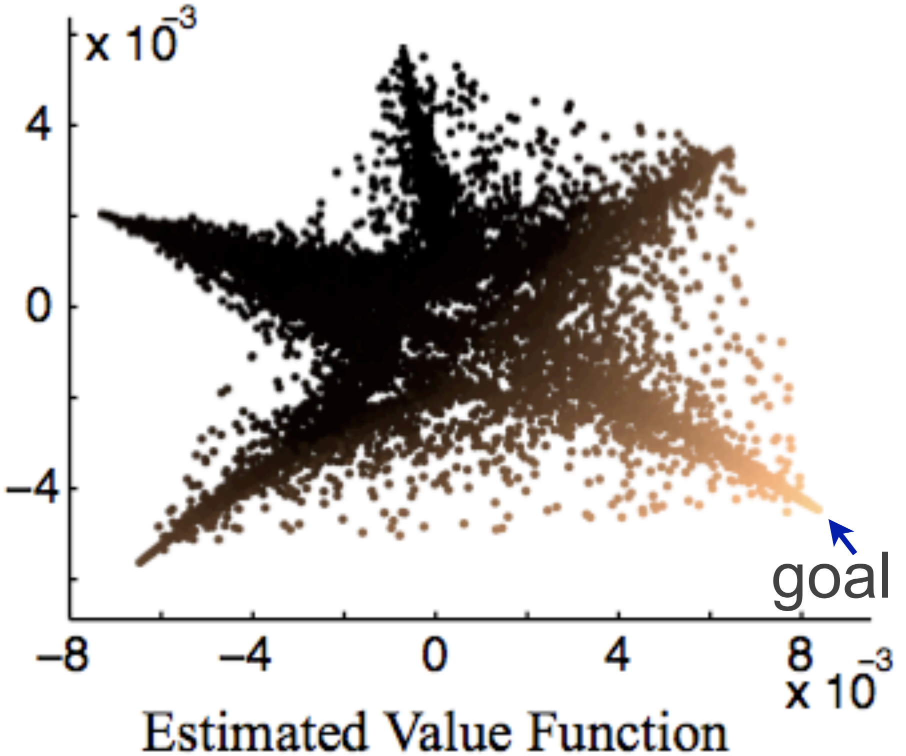
- regression from state to reward
- include reward as observation

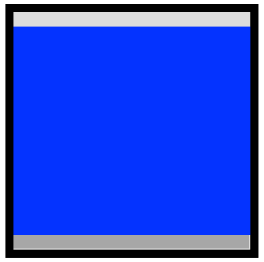
Planning

Planning in PSRs is just like planning in POMDPS:

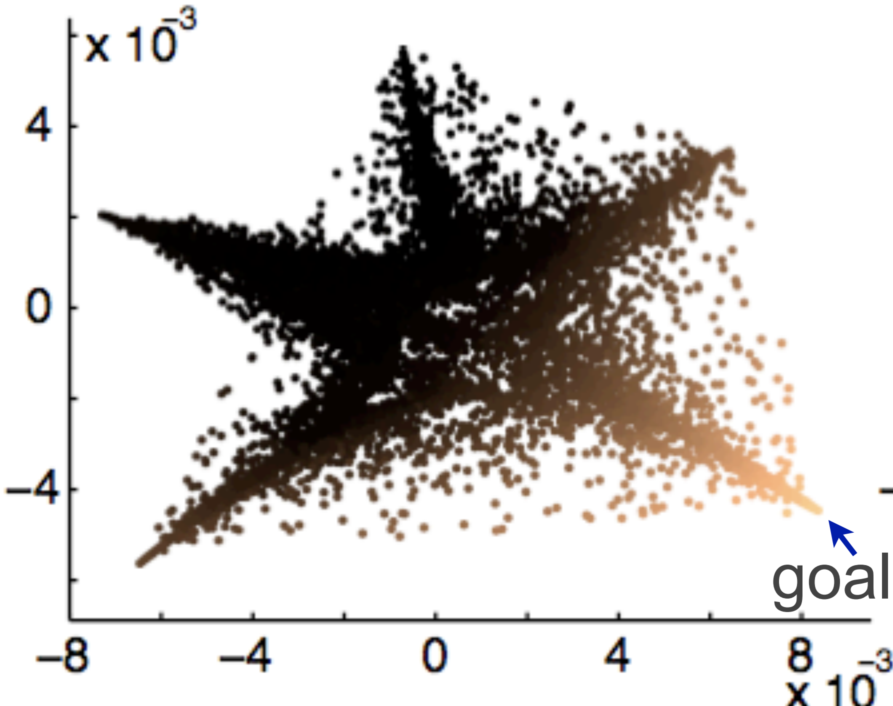
- i.e. Hard! (in the worst case exponential in horizon)
- exponential depends on dim. of PSR
  - Thus, PSRs can be exponentially better than POMDPs due to lower dimensionality of state space.
- we can use approximate techniques (PBVI)

Planning

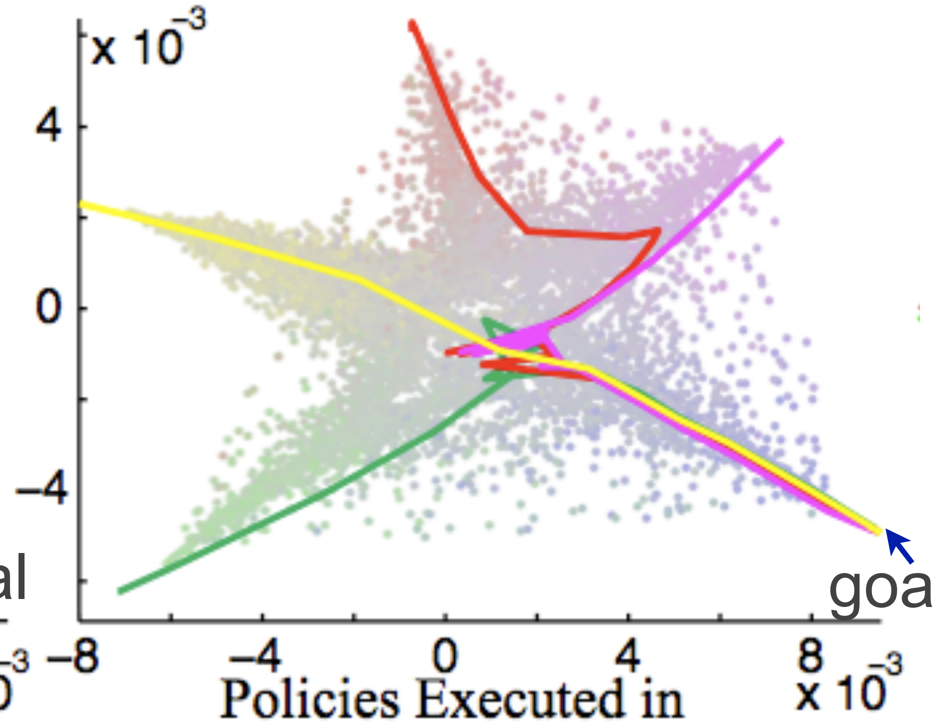




Planning

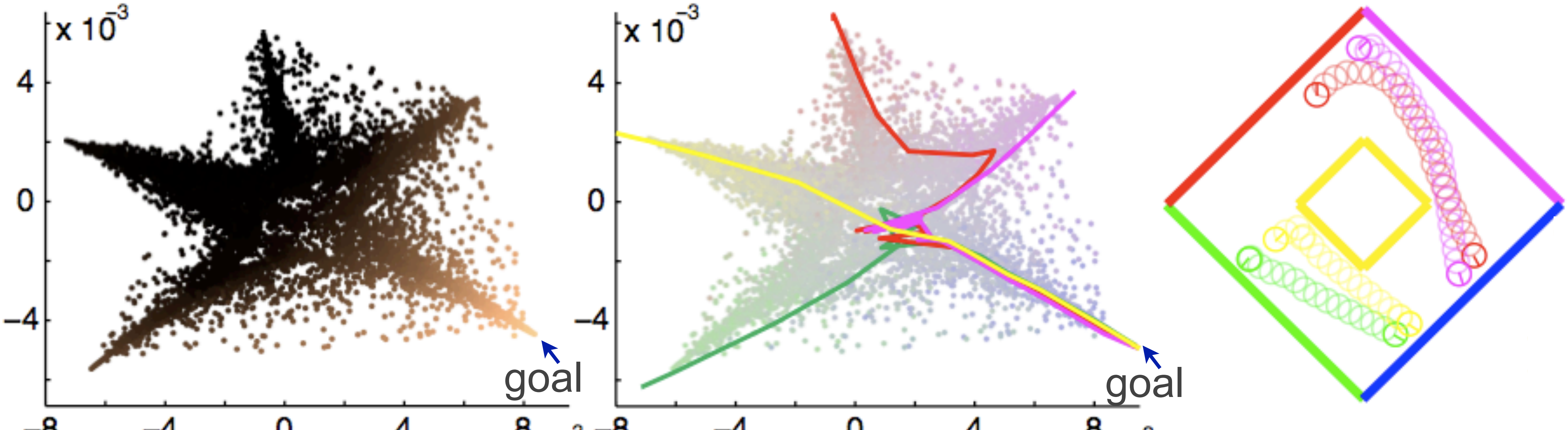


Estimated Value Function



Policies Executed in Learned Subspace

Planning



Estimated Value Function

Policies Executed in Learned Subspace

Paths Taken in Geometric Space

Conclusion

Summary:

- Introduced a consistent, spectral learning algorithm for PSRs
- Extended learning algorithm to use features
- Successfully applied PSR learning to high dimensional data
- Planned in learned model
- Should be viewed in the context of merging powerful subspace ID algorithms together with planning and RL

Thank you!