Efficient and Tractable System Identification through Supervised Learning

AHMED HEFNY
MACHINE LEARNING DEPARTMENT
CARNEGIE MELLON UNIVERSITY

JOINT WORK WITH
GEOFFREY GORDON, CARLTON DOWNEY,
BYRON BOOTS (GEORGIA TECH),
ZITA MARINHO, WEN SUN

Outline

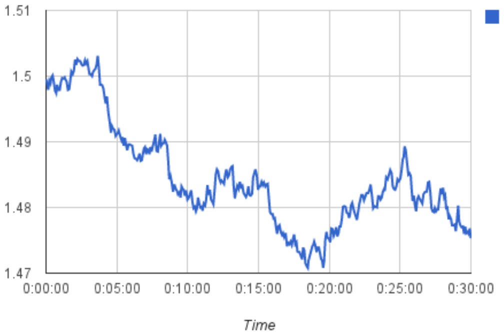
- Problem Statement:
  - Learning Dynamical Systems
  - Solution Properties
- Formulation:
  - A Taxonomy of Dynamical System Models
  - Predictive State Models: Formulation and Learning
  - Connection to Recurrent Networks
- Extensions:
  - Controlled Systems
  - Reinforcement Learning

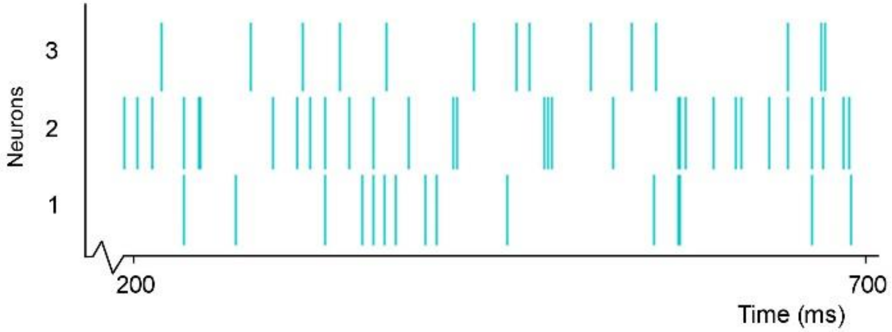
Outline

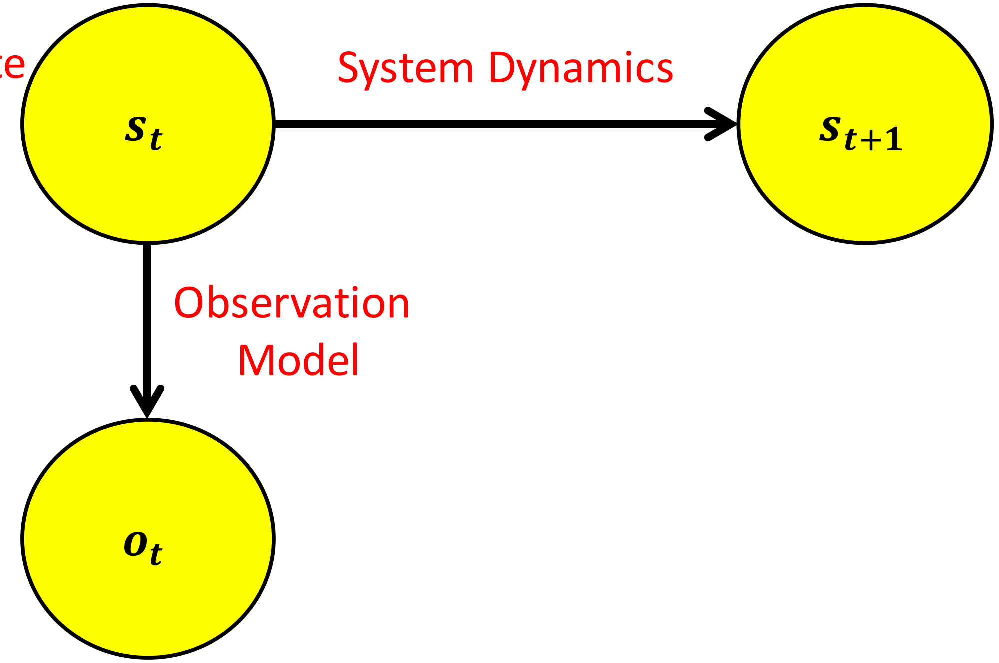
- Problem Statement:
  - Learning Dynamical Systems
  - Solution Properties
- Formulation:
  - A Taxonomy of Dynamical System Models
  - Predictive State Models: Formulation and Learning
  - Connection to Recurrent Networks
- Extensions:
  - Controlled Systems
  - Reinforcement Learning

Learning Dynamical Systems

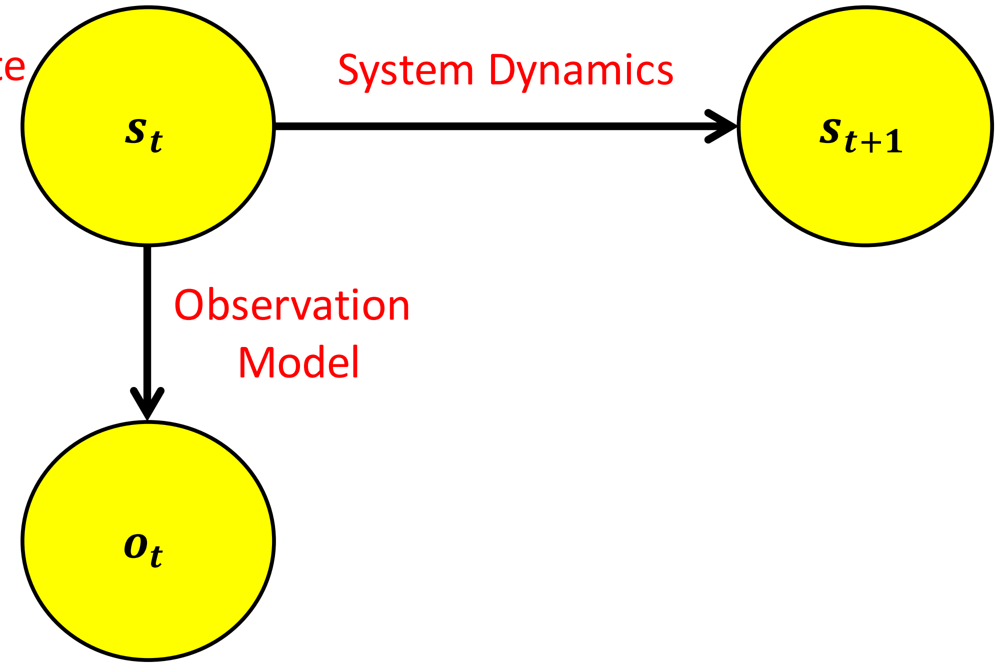
Learning Dynamical Systems





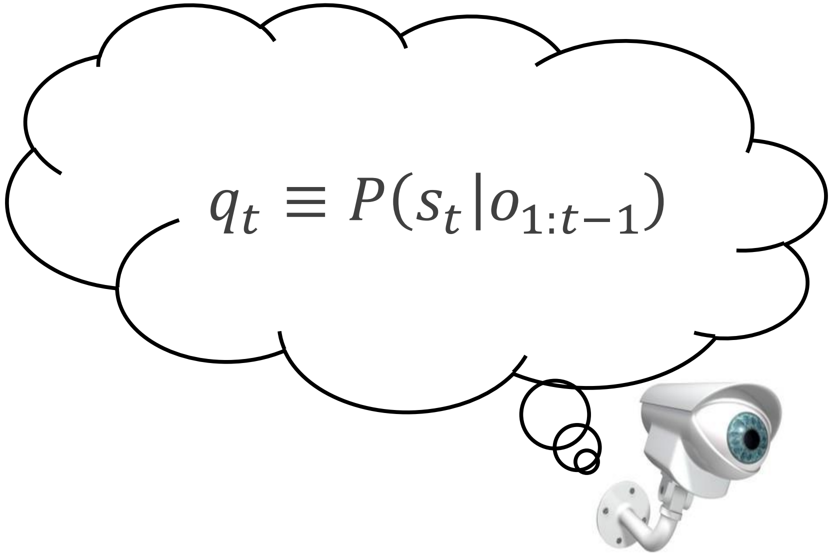


Learning Dynamical Systems

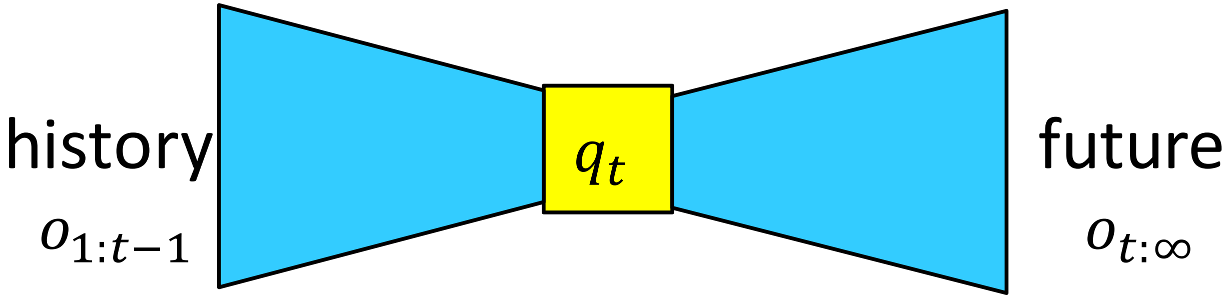


$$q_t \equiv P(s_t | o_{1:t-1})$$

Learning Dynamical Systems



$$P(\,o_{t+\tau}\mid o_{1:t=1}\,) \equiv P(o_{t+\tau}|q_t)$$



Learning a Recursive Filter

Given:

Training Sequences:

$(o_1, o_2, \ldots, o_T)$

Output:

- Initial belief q_1
- Filtering function f $\qquad q_{t+1} = f(q_t, o_t)$
- Observation function g $\qquad E[\, o_t \mid o_{1:t-1} \,] = g(q_t)$

Learning a Recursive Filter

Given:

Training Sequences:

$(o_1, o_2, \dots, o_T)$

Output:

- Initial belief q_1
- Filtering function f $\quad q_{t+1} = f(q_t, o_t)$
- Observation function g $\quad E[\, o_t \mid o_{1:t-1} \,] = g(q_t)$

System
- Non-linear
- Partially observable
- Controlled

Algorithm
- Theoretical Guarantees
- Scalability

Outline

- Problem Statement:
  - Learning Dynamical Systems
  - Solution Properties
- Formulation:
  - A Taxonomy of Dynamical System Models
  - Predictive State Models: Formulation and Learning
  - Connection to Recurrent Networks
- Extensions:
  - Controlled Systems
  - Reinforcement Learning

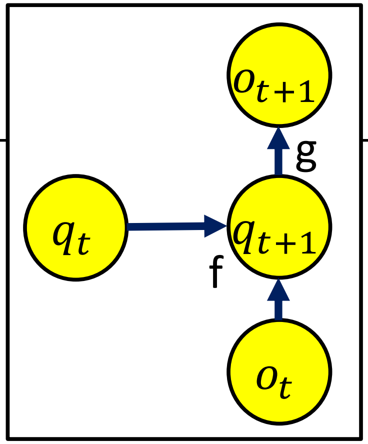
Learning a Recursive Filter

Given:

Training Sequences:

$(o_1, o_2, \dots, o_T)$

		RNN [BPTT]



Output:

- Initial belief q_1
- Filtering function f — $q_{t+1} = f(q_t, o_t)$
- Observation function g — $E[\,o_t \mid o_{1:t-1}\,] = g(q_t)$

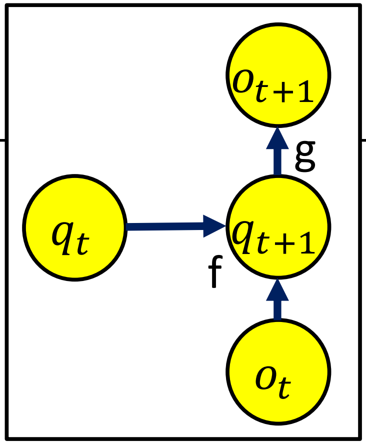
Learning a Recursive Filter

Given:

Training Sequences:

$(o_1, o_2, \ldots, o_T)$

Learn a model then derive f		HMM [EM, Tensor Decomp.]
Directly Learn f		RNN [BPTT]



Output:

- Initial belief q_1
- Filtering function f
- Observation function g

$$q_{t+1} = f(q_t, o_t)$$

$$E[\, o_t \mid o_{1:t-1} \,] = g(q_t)$$

Learning a Recursive Filter

Given:

Training Sequences:

$(o_1, o_2, \dots, o_T)$

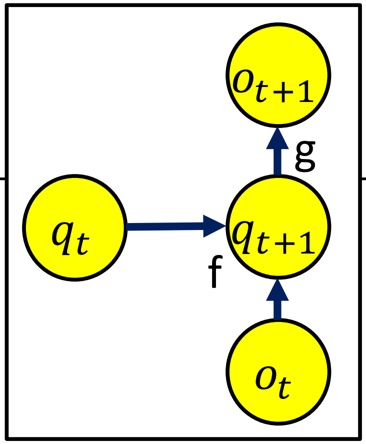
	Fix g (Predictive State)	Learn g (Latent State)
Learn a model then derive f		HMM [EM, Tensor Decomp.]
Directly Learn f	$q_t \equiv E[\psi(o_{t:\infty}) \mid o_{1:t-1}]$ state = E[sufficient future stats] PSIM [DAgger]	RNN [BPTT]

Output:

- Initial belief q_1
- Filtering function f
- Observation function g

$$q_{t+1} = f(q_t, o_t)$$

$$E[\, o_t \mid o_{1:t-1} \,] = g(q_t)$$



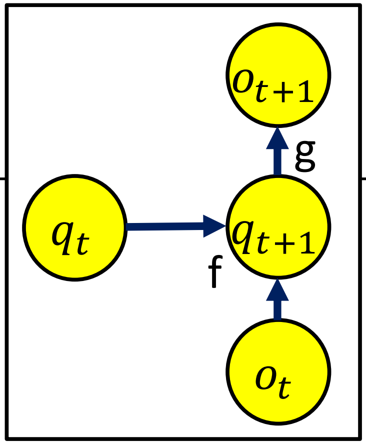
Learning a Recursive Filter

Given:

Training Sequences:

$(o_1, o_2, \ldots, o_T)$

	Fix *g* (Predictive State)	Learn *g* (Latent State)
Learn a model then derive *f*	Predictive State Models [Method of moments: Two-stage regression]	HMM [EM, Tensor Decomp.]
Directly Learn *f*	$q_t \equiv E[\psi(o_{t:\infty}) \mid o_{1:t-1}]$ state = E[sufficient future stats] PSIM [DAgger]	RNN [BPTT]



Output:

- Initial belief q_1
- Filtering function f
- Observation function g

$$q_{t+1} = f(q_t, o_t)$$

$$E[\, o_t \mid o_{1:t-1} \,] = g(q_t)$$

Learning a Recursive Filter

Why restrict f and g ?

* Predictive State (a.k.a Observable Representation):

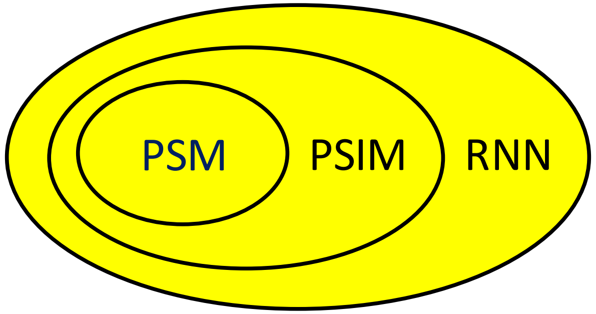
State is a prediction of future observation statistics

→ Future statistics are noisy estimates of the state.

→ Reduction to supervised learning.

* Additional assumptions on dynamics facilitate the development of an efficient algorithm with provable guarantees.

* Local improvement is still possible.

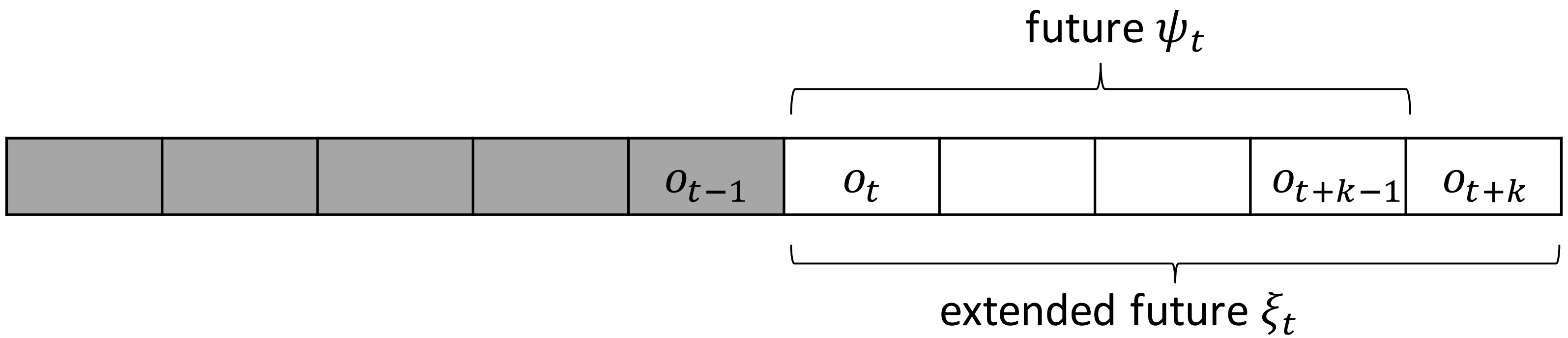


Predictive State Model (Formulation)

Predictive State $q_t \equiv P(\, o_{t:t+k-1} \mid o_{1:t-1}\,) \equiv E[\, \psi_t \mid o_{1:t-1}\,]$

Extended Predictive State $p_t \equiv P(\, o_{t:t+k} \mid o_{1:t-1}\,) \equiv E[\, \zeta_t \mid o_{1:t-1}\,]$

Linear Dynamics $p_t = \boldsymbol{W}\, q_t$



$\boldsymbol{\psi_t}$	$\boldsymbol{q_t}$
Indicator Vector	Joint Probability Table
1st and 2nd moments	Gaussian Distribution

Predictive State Model (Formulation)

Predictive State $q_t \equiv P(\, o_{t:t+k-1} \mid o_{1:t-1}\,) \equiv E[\,\psi_t \mid o_{1:t-1}\,]$

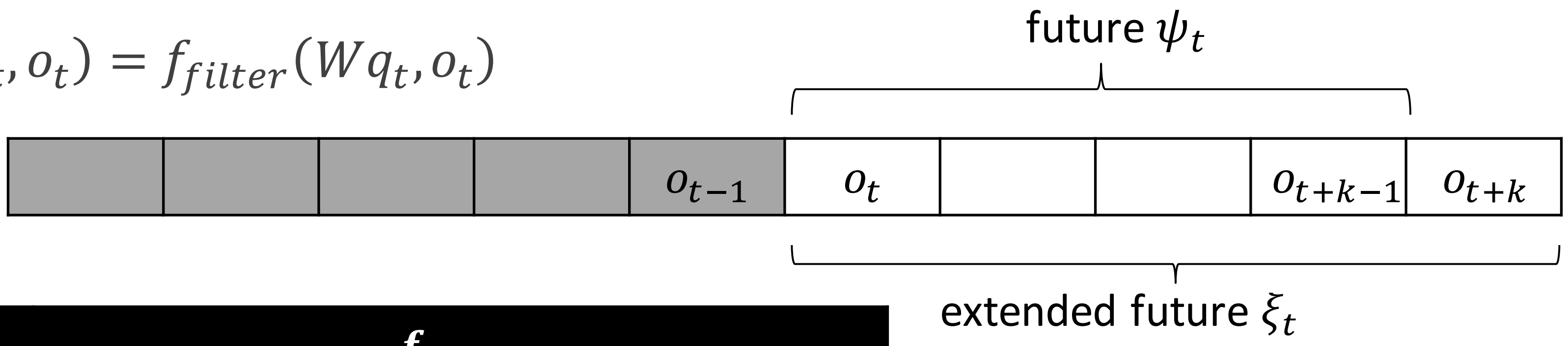
Extended Predictive State $p_t \equiv P(\, o_{t:t+k} \mid o_{1:t-1}\,) \equiv E[\,\zeta_t \mid o_{1:t-1}\,]$

Linear Dynamics $p_t = \boldsymbol{W}\, q_t$

Filtering: $f(q_t, o_t) = \boldsymbol{f}_{\boldsymbol{filter}}(p_t, o_t) = f_{filter}(W q_t, o_t)$

learned

fixed



$\boldsymbol{\psi_t}$	$\boldsymbol{q_t}$	$\boldsymbol{f_{filter}}$
Indicator Vector	Joint Probability Table	Bayes Rule: $P(\, o_{t+1:t+k} \mid o_{1:t}\,) \propto P(o_{t:t+k} \mid o_{1:t-1})$
1st and 2nd moments	Gaussian Distribution	Gaussian conditional mean and covariance.

Predictive State Model (Formulation)

Predictive State $q_t \equiv P(\, o_{t:t+k-1} \mid o_{1:t-1}\,) \equiv E[\, \psi_t \mid o_{1:t-1}\,]$

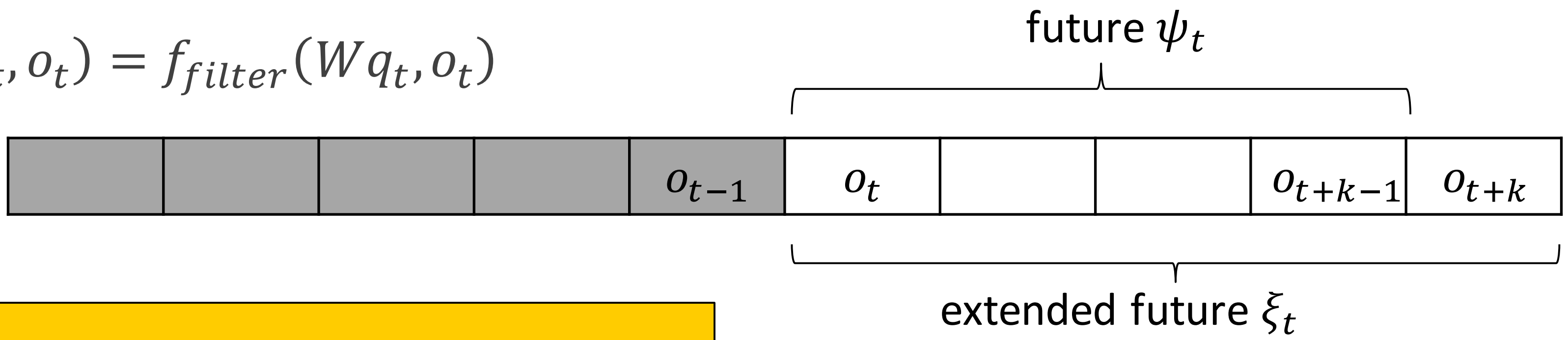
Extended Predictive State $p_t \equiv P(\, o_{t:t+k} \mid o_{1:t-1}\,) \equiv E[\, \zeta_t \mid o_{1:t-1}\,]$

Linear Dynamics $p_t = \boldsymbol{W}\, q_t$

Filtering: $f(q_t, o_t) = \boldsymbol{f}_{\boldsymbol{filter}}(p_t, o_t) = f_{filter}(W q_t, o_t)$

learned

fixed



Why linear W ?
Crucial to the consistency of learning algorithm.

Why this particular filtering formulation ?
Matches existing models (HMM, Kalman filter, PSR)

Predictive State Model (Learning)

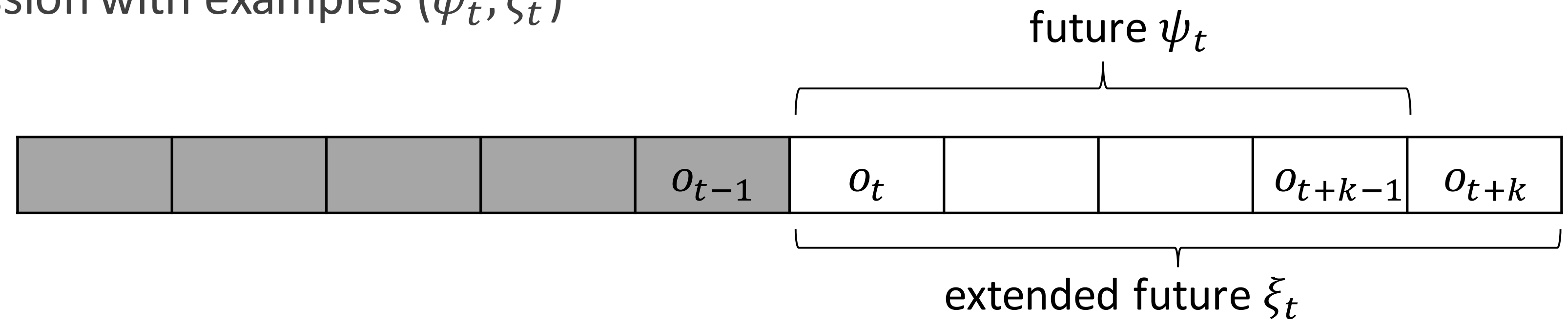
ϵ_t and ν_t are correlated
$Cov(q_t, p_t) \neq Cov(\psi_t, \zeta_t)$

ψ_t and ζ_t are unbiased estimates of q_t and p_t:

- $\psi_t = q_t + \epsilon_t$
- $\zeta_t = p_t + \nu_t$

Learning Procedure:

- $\hat{q}_0 = \frac{1}{N} \sum_i \psi_i$
- ~~Learn W using linear regression with examples (ψ_t, ζ_t)~~



Predictive State (Learning)

ψ_t and ζ_t are unbiased estimates of q_t and p_t:

- $\psi_t = q_t + \epsilon_t$
- $\zeta_t = p_t + \nu_t$

Learning Procedure:

- $\tilde{q}_0 = \frac{1}{N}\sum_i \psi_i$
- Denoise examples (ψ_t, ζ_t) to obtain $(\hat{q}_t, \hat{p}_t)$???
- Learn W using linear regression with examples $(\hat{q}_t, \hat{p}_t)$

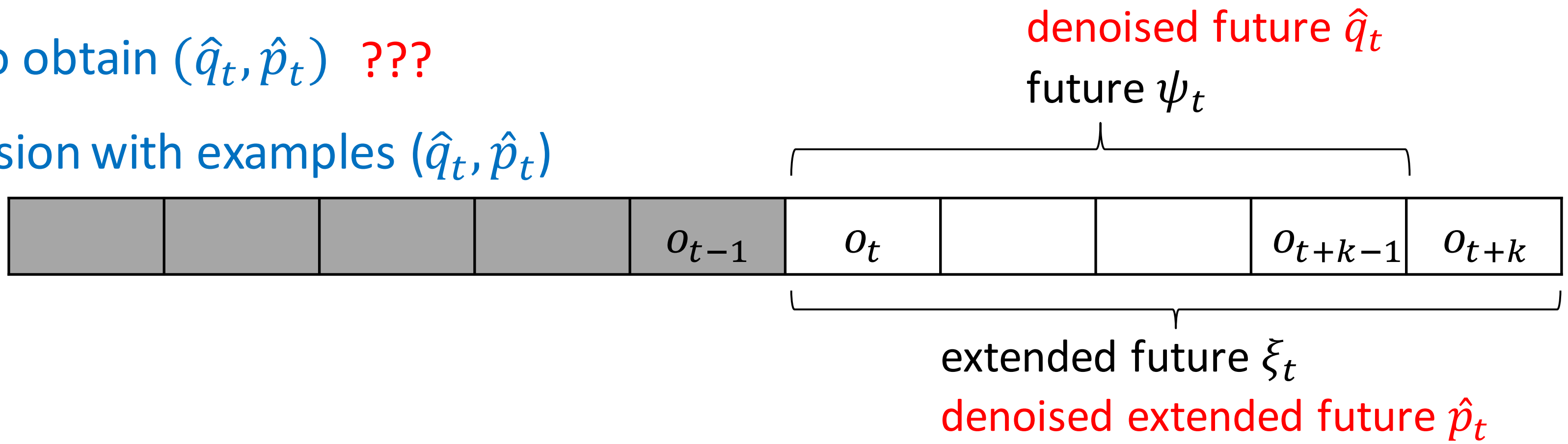
ϵ_t and ν_t are correlated
$Cov(q_t, p_t) \neq Cov(\psi_t, \zeta_t)$

$p_t = W q_t$

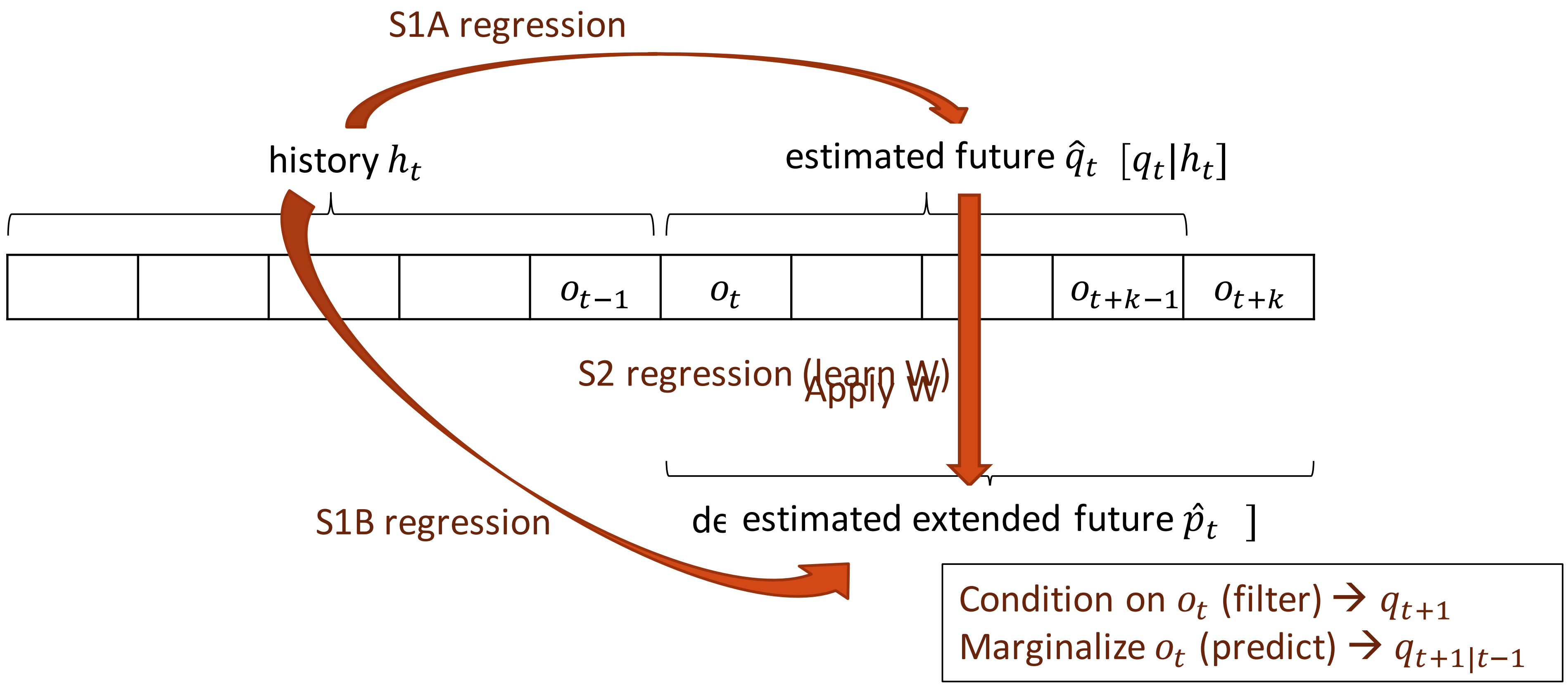
$E[\zeta_t \mid o_{1:t-1}] = W E[\psi_t \mid o_{1:t-1}]$

Use regression $E[\zeta_t \mid o_{t-k:t-1}] = W E[\psi_t \mid o_{t-k:t-1}]$

$\hat{q}_t = W \hat{p}_t$



Learning Dynamical Systems Using Instrument Regression



In a nutshell

* Predictive State:
  - State is a prediction of future observations
  - Future observations are noisy estimates of the state

* Two stage regression:
  - Use history features to “denoise” states (S1 Regression)
  - Use denoised states to learn dynamics (S2 Regression)

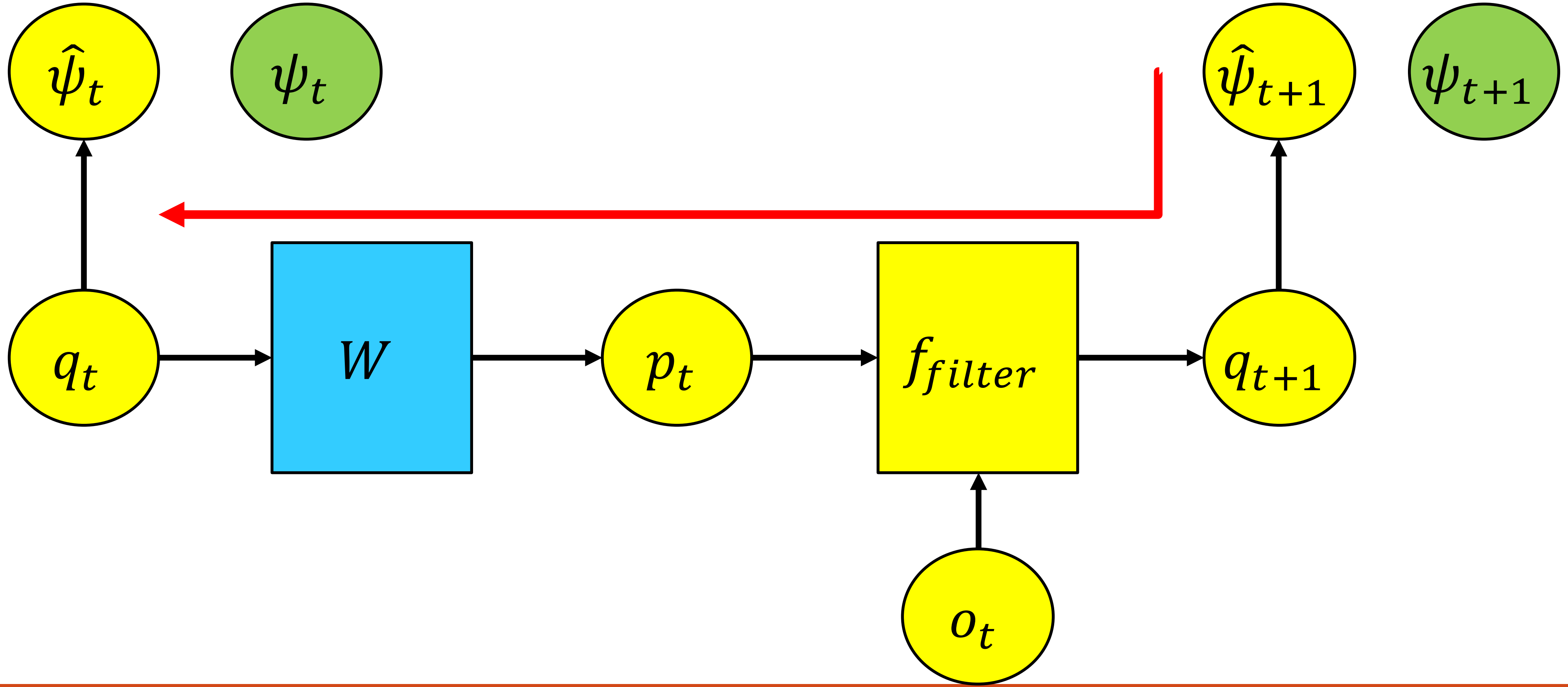
What do we gain ?

* More understanding of existing algorithms:
  - Spectral algorithms for learning HMMs, Kalman filters, PSRs are two stage regression algorithms with linear regression in all stages.

* Theoretical Results (Asymptotic and finite sample):
  - Error in estimating W is $\tilde{O}\left(1/\sqrt{N}\right)$ [Under mild assumptions]
  - Exact rate depends on S1 regression error

* New flavors of dynamical systems learning algorithms:
  - HMM with logistic regression.
  - Online learning of linear dynamical systems (Sun et al. 2015).
  - Linear dynamical systems with sparse dynamics (Hefny et al 2015, Gus Xia 2016).

Predictive State Models as RNNs



Predictive State Models as RNNs

Predictive state models define RNNs that are easy to initialize !!

Back to Special Case: Modeling Discrete Systems with Indicator Vectors

Assume the discrete case: o_t is an indicator vector.

Let $\psi_t = o_t$ and $\zeta_t = o_t \otimes o_{t+1}$

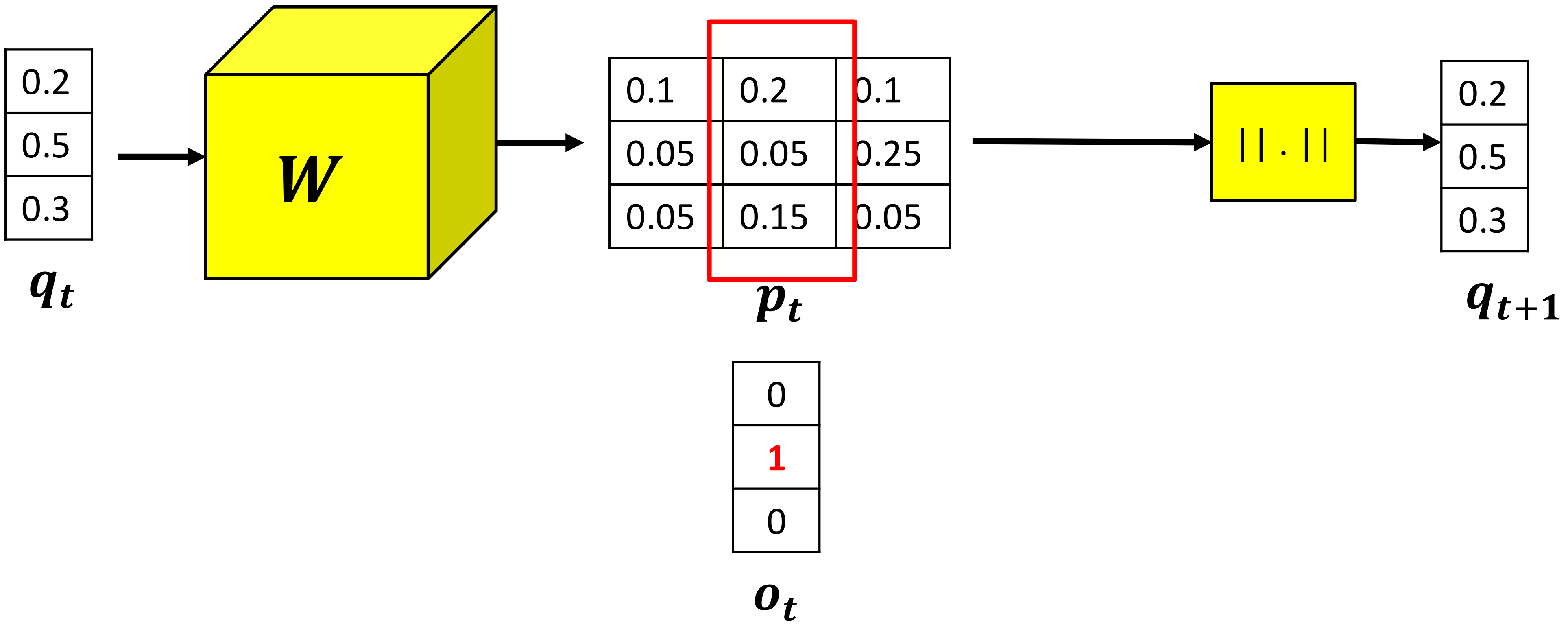
Then:

$q_t \rightarrow$ Probability Vector

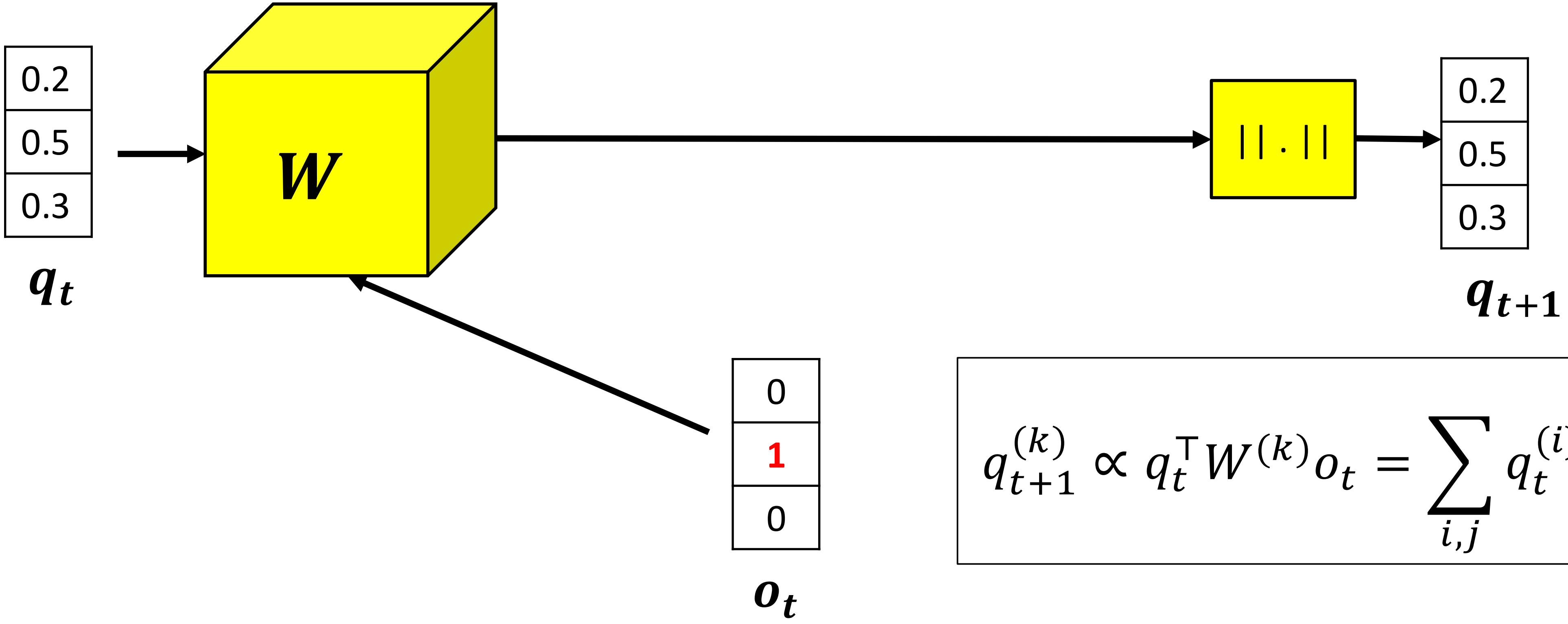
$p_t \rightarrow$ Joint Probability Table

$f(p_t, o_t) \rightarrow$ Choose column from p_t corresponding to o_t then renormalize

Back to Special Case: Modeling Discrete Systems with Indicator Vectors

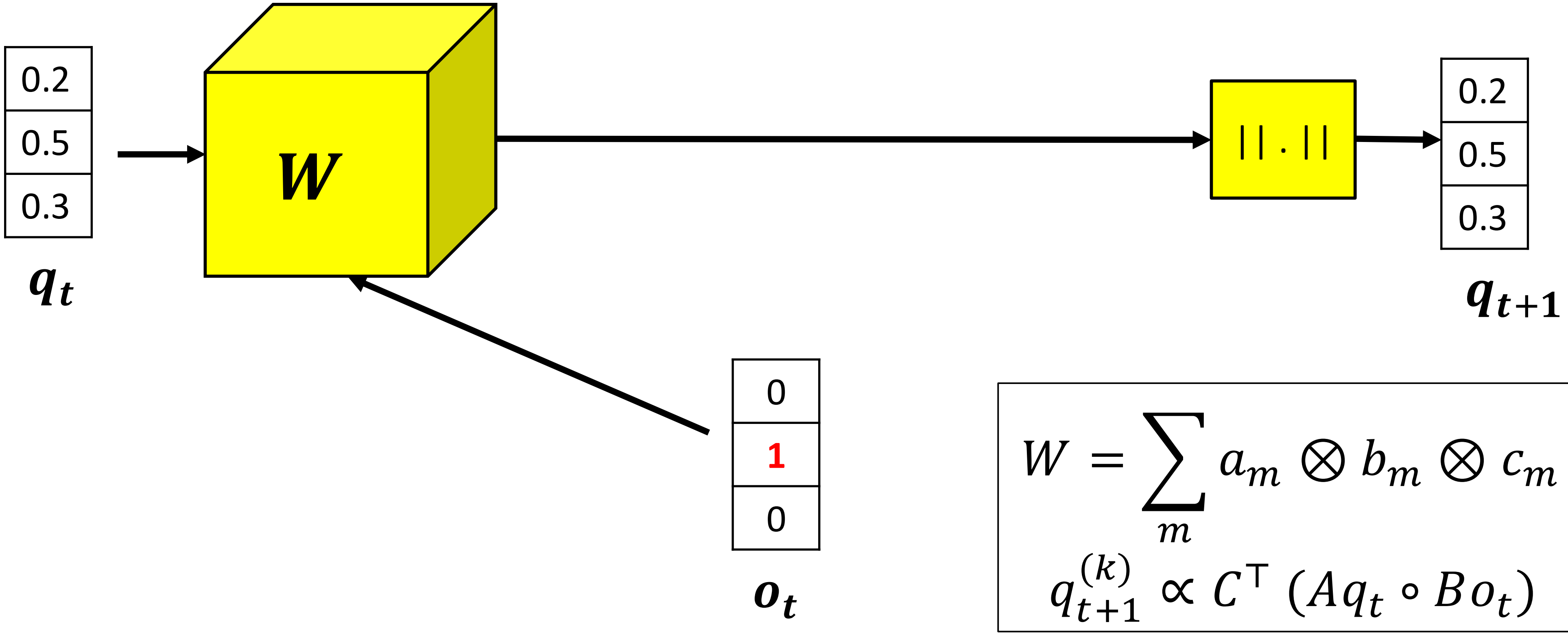


Back to Special Case: Modeling Discrete Systems with Indicator Vectors



$$q_{t+1}^{(k)} \propto q_t^\top W^{(k)} o_t = \sum_{i,j} q_t^{(i)} W^{(ijk)} o_t^{(j)}$$

Back to Special Case: Modeling Discrete Systems with Indicator Vectors



$$W = \sum_m a_m \otimes b_m \otimes c_m$$

$$q_{t+1}^{(k)} \propto C^\top \left(A q_t \circ B o_t \right)$$

Multiplicative unit

Predictive State Models as RNNs

Predictive units have a multiplicative structure, similar to LSTMs and GRUs.

What about the continuous case ?

Mean-maps for Continuous Observations

Mean-maps provide a powerful tool to model non-parametric distribution using the feature map of a universal kernel.

A discrete distribution is a special case that uses the delta kernel and indicator feature map. Continuous distributions can be modeled using e.g. RBF kernel.

Discrete Case	General Case	
Indicator Vector	Kernel feature map $\phi(x)$	
Joint Probability Table P(X,Y)	Covariance Operator C_{XY}	
Conditional Probability Table	Conditional Operator $C_{X\|Y}$	
Normalization P(X,Y) → P(X|Y)	Kernel Bayes Rule $C_{X\|Y} = C_{XY}C_{YY}^{-1}$	

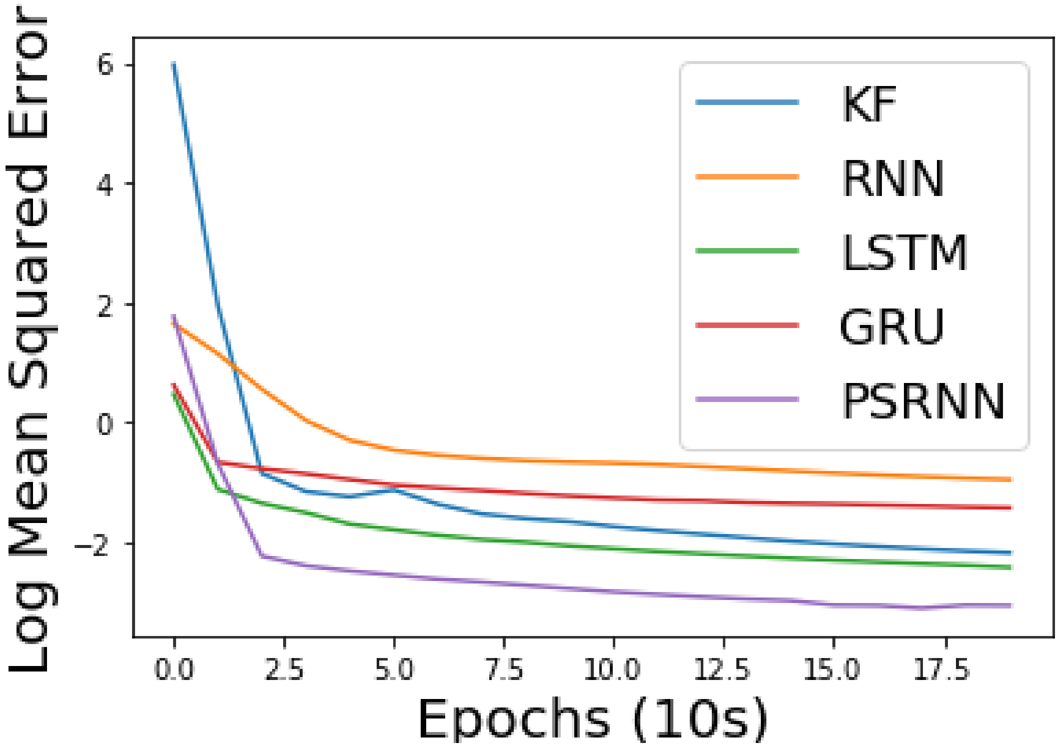
Mean-maps for Continuous Observations

Mean-maps provide a powerful tool to model non-parametric distribution using the feature map of a universal kernel.

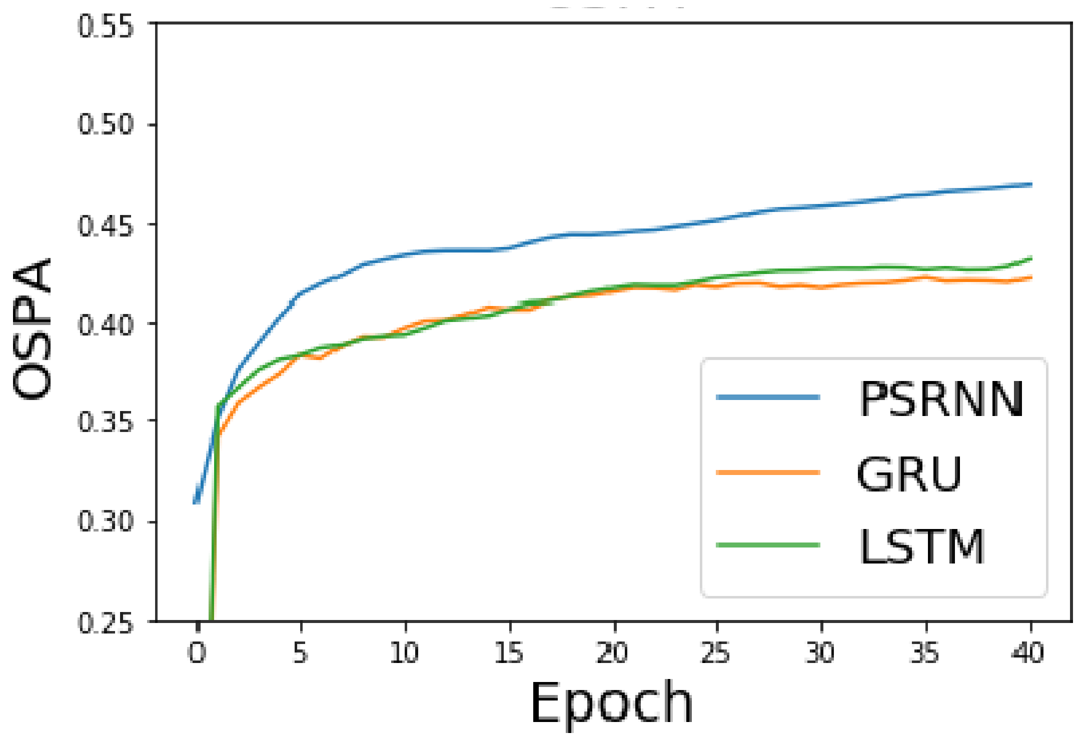
A discrete distribution is a special case that uses the delta kernel and indicator feature map. Continuous distributions can be modeled using e.g. RBF kernel.

Discrete Case	General Case	RFF Approximation
Indicator Vector	Kernel feature map $\phi(x)$	RFF Feature Vector
Joint Probability Table P(X,Y)	Covariance Operator C_{XY}	Covariance Matrix
Conditional Probability Table	Conditional Operator $C_{X\|Y}$	Conditional Matrix
Normalization P(X,Y) → P(X|Y)	Kernel Bayes Rule $C_{X\|Y} = C_{XY}C_{YY}^{-1}$	Solve Linear System

Results



Error predicting handwriting trajectories



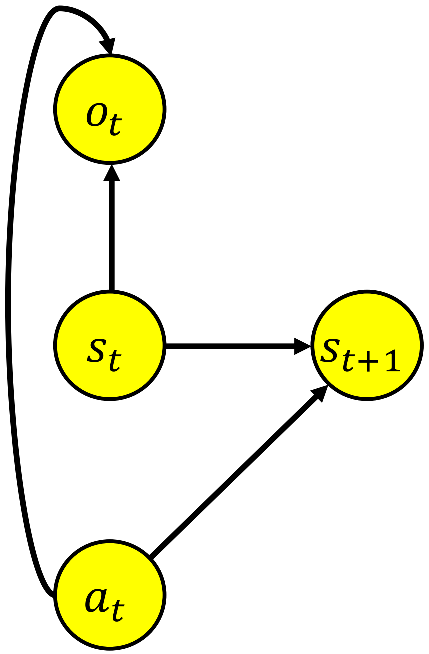
Character prediction accuracy of English text

Outline

- Problem Statement:
  - Learning Dynamical Systems
  - Solution Properties
- Formulation:
  - A Taxonomy of Dynamical System Models
  - Predictive State Models: Formulation and Learning
  - Connection to Recurrent Networks
- Extensions:
  - Controlled Systems
  - Reinforcement Learning

Extension to Controlled Systems

In controlled systems we have observations and actions. (e.g. car velocity and pressure on pedals)



Recursive Filter:

$$E[\, o_t \mid q_t, a_t \,] = g(q_t, a_t)$$

$$E[q_{t+1} | q_t, o_t, a_t] = f(q_t, a_t, o_t)$$

Extension to Controlled Systems

Same principle

$$P_t = W(Q_t)$$

This time, the predictive state is a linear operator encoding conditional distribution of future observations given future actions.

Example: Think of Q_t and P_t as conditional probability tables.

Requires appropriate modifications to S1 regression. S2 regression remains the same.

Extension to Controlled Systems

Two stage regression: It is all about finding

$$E[Q_t | o_{t-k:t-1}, a_{t-k:t-1}]$$

0.1	0.8	0.5	0.2
0.3	0.1	0.2	0.1
0.6	0.1	0.3	0.7

Observation

Action

Extension to Controlled Systems

Two stage regression: It is all about finding

$$E[Q_t | o_{t-k:t-1}, a_{t-k:t-1}]$$

Problem:

At each time step we observe a noisy version of

a *slice* of Q_t

Observation

?	?	1	?
?	?	0	?
?	?	0	?

Action

Extension to Controlled Systems

Two stage regression: It is all about finding

$$E[Q_t | o_{t-k:t-1}, a_{t-k:t-1}] = \hat{Q}(o_{t-k:t-1}, a_{t-k:t-1})$$

Problem:

At each time step we observe a noisy version of

a *slice* of Q_t

Solution 1 (Joint Modeling):

Predict the joint distribution of observation and actions.

Manually convert to conditional table (e.g. normalize columns).

Observation

0	0	1	0
0	0	0	0
0	0	0	0

Action

Extension to Controlled Systems

Two stage regression: It is all about finding

$$E[Q_t | o_{t-k:t-1}, a_{t-k:t-1}] = \hat{Q}(o_{t-k:t-1}, a_{t-k:t-1})$$

Problem:

At each time step we observe a noisy version of a *slice* of Q_t

Solution 2 (Conditional Modeling):

Train regression model to fit the observed slice of Q_t.

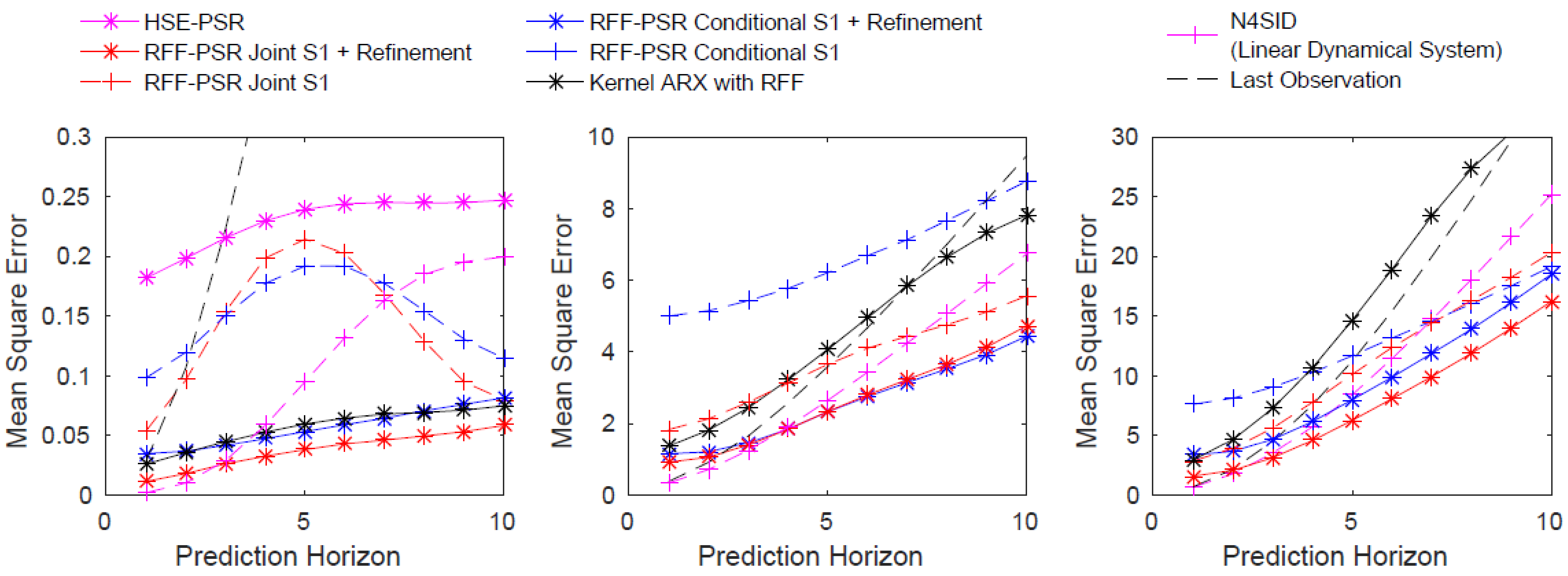
$$\min_{\hat{Q}} \sum_t \left\| \hat{Q}(o_{t-k:t-1}, a_{t-k:t-1})\, \psi_t^a - \psi_t^o \right\|^2$$

Observation

D/C	D/C	1	D/C
D/C	D/C	0	D/C
D/C	D/C	0	D/C

Action

Results

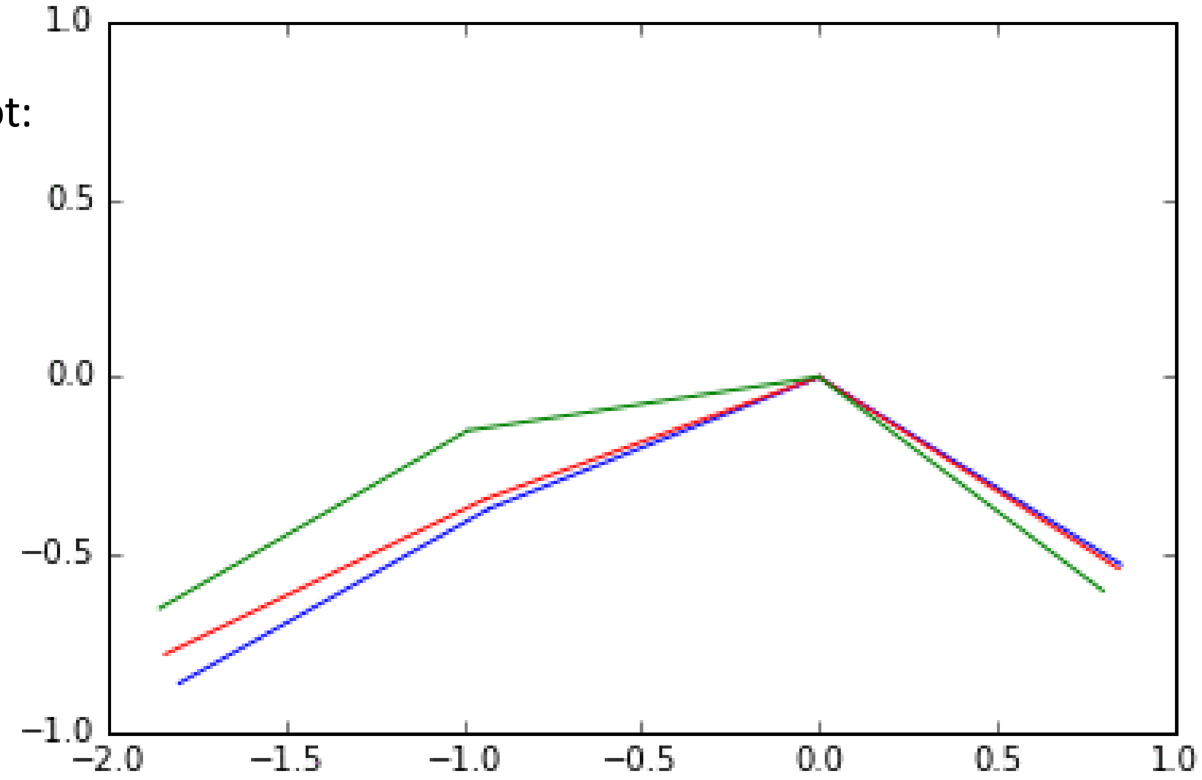


HSE-PSR
RFF-PSR Joint S1 + Refinement
RFF-PSR Joint S1
RFF-PSR Conditional S1 + Refinement
RFF-PSR Conditional S1
Kernel ARX with RFF
N4SID
(Linear Dynamical System)
Last Observation
Mean Square Error
Prediction Horizon
Mean Square Error
Prediction Horizon
Mean Square Error
Prediction Horizon

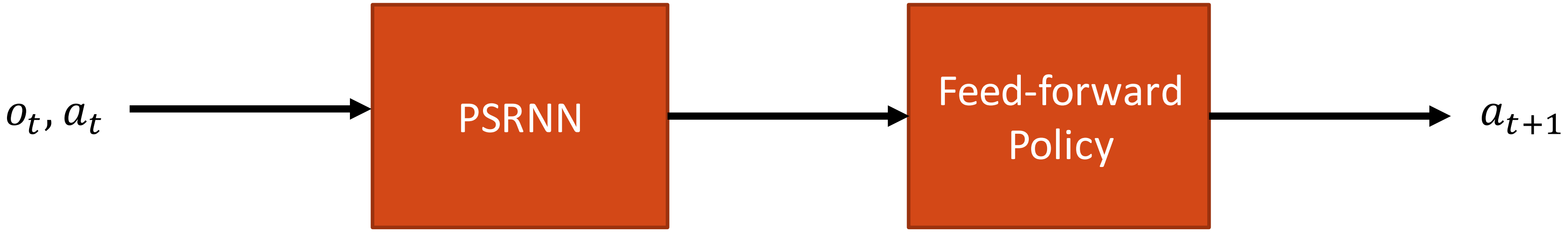
Results

Predicting the pose of a swimming robot:

- Ground Truth
- Linear ARX
- RFFPSR (our model)

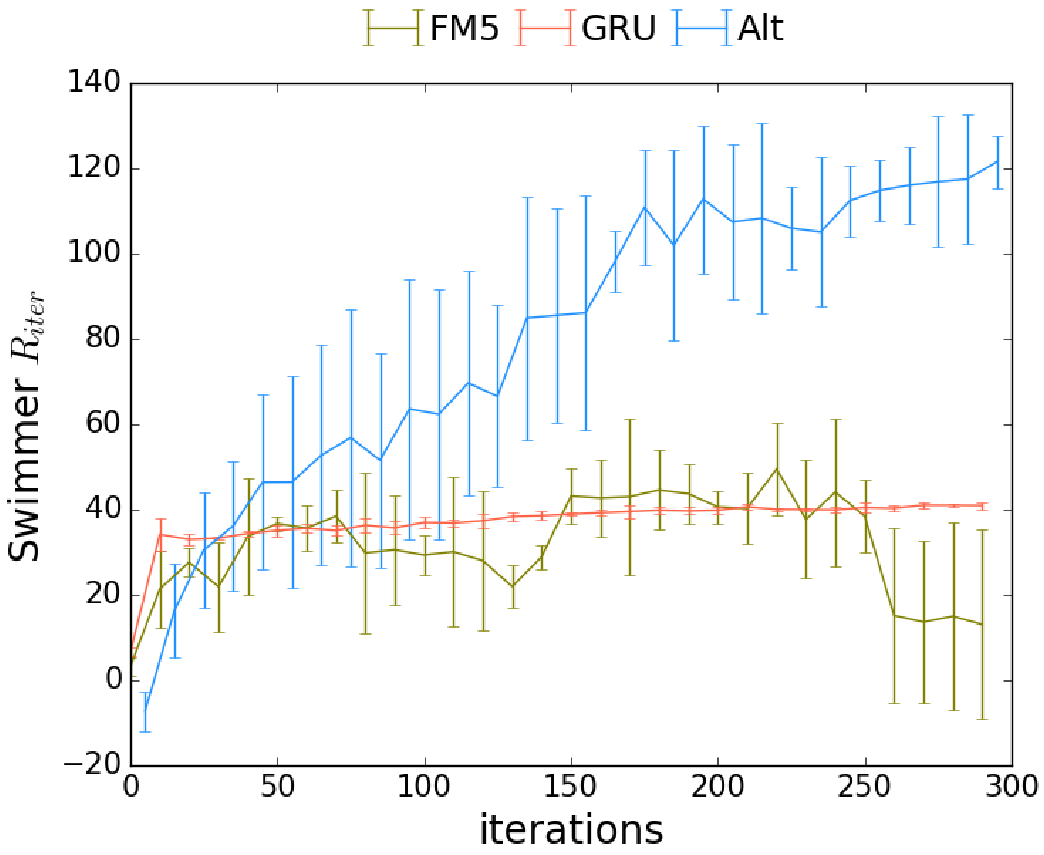


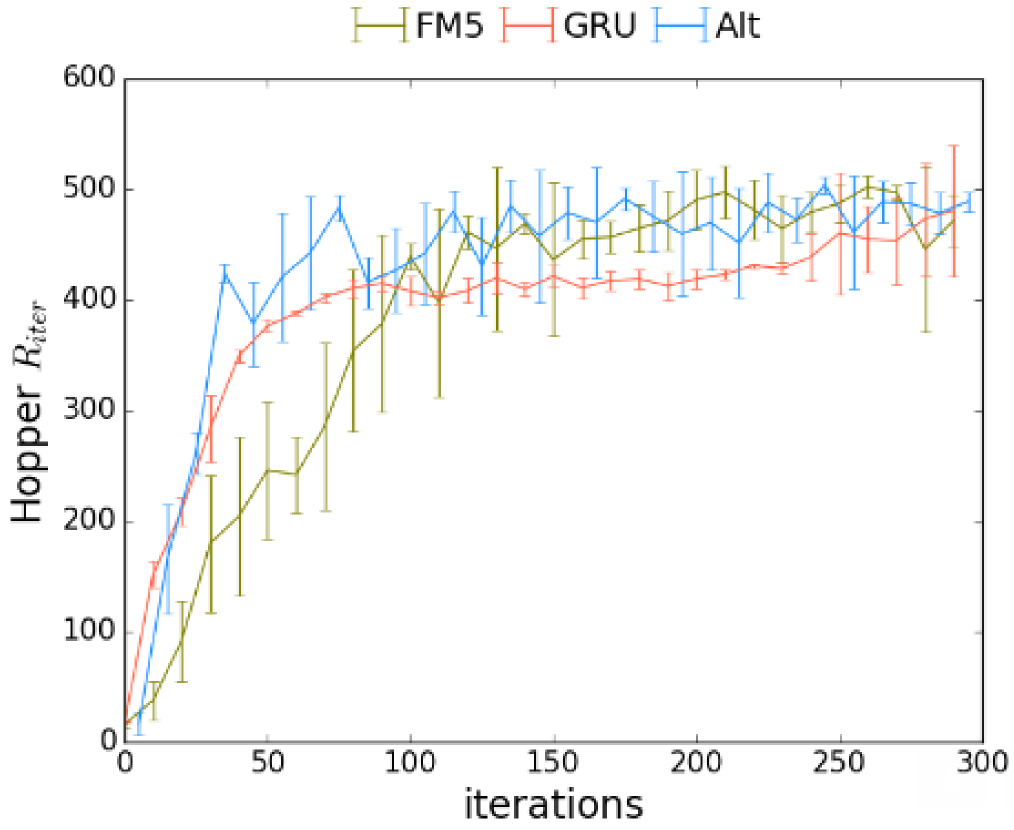
Reinforcement Learning with Predictive State Policy Networks



- Can be initialized
- States have a meaning: can be trained to match observations

Preliminary Results





Conclusions

- Predictive State Models for filtering in dynamical systems:
  - **Predictive State:** State is a prediction of future observations.
  - **Two Stage Regression**: Learning predictive state models can be reduced to supervised learning.

- Predictive State Models are a special type of recurrent networks.

- Can be extended to controlled systems and employed in reinforcement learning.

Thank you !

* Hefny, Downey and Gordon, “Supervised Learning for Dynamical System Learning” (NIPS15), https://arxiv.org/abs/1505.05310

* Hefny, Downey and Gordon, “Supervised Learning for Controlled Dynamical System Learning”, https://arxiv.org/abs/1702.03537

* Downey, Hefny, Li, Boots and Gordon, “Predictive State Recurrent Neural Networks”, https://arxiv.org/abs/1705.09353