Quantification of Space-Time Structure with Dynamical Systems

Jose C. Principe

COMPUTATIONAL NEUROENGINEERING LABORATORY
UNIVERSITY OF FLORIDA

principe@cnel.ufl.edu

UF UNIVERSITY of FLORIDA
The Foundation for The Gator Nation

C N E L
Computational NeuroEngineering Lab
University of Florida

Acknowledgments

- My Students
  - Kan Li
  - Pingping Zhu
  - Goktug Cinar
  - Rakesh Chalasani
- My Collaborators
  - Badong Chen and Andreas Keil
- DARPA and NSF support

Overview

- Hierarchical Kalman Filters
- Cognitive Architectures for Sensory Processing
- KAARMA Algorithm
- Applications
  - Grammatical Inference (States)
  - Speech Recognition (States + Transitions)
- Conclusion

Time Dependency

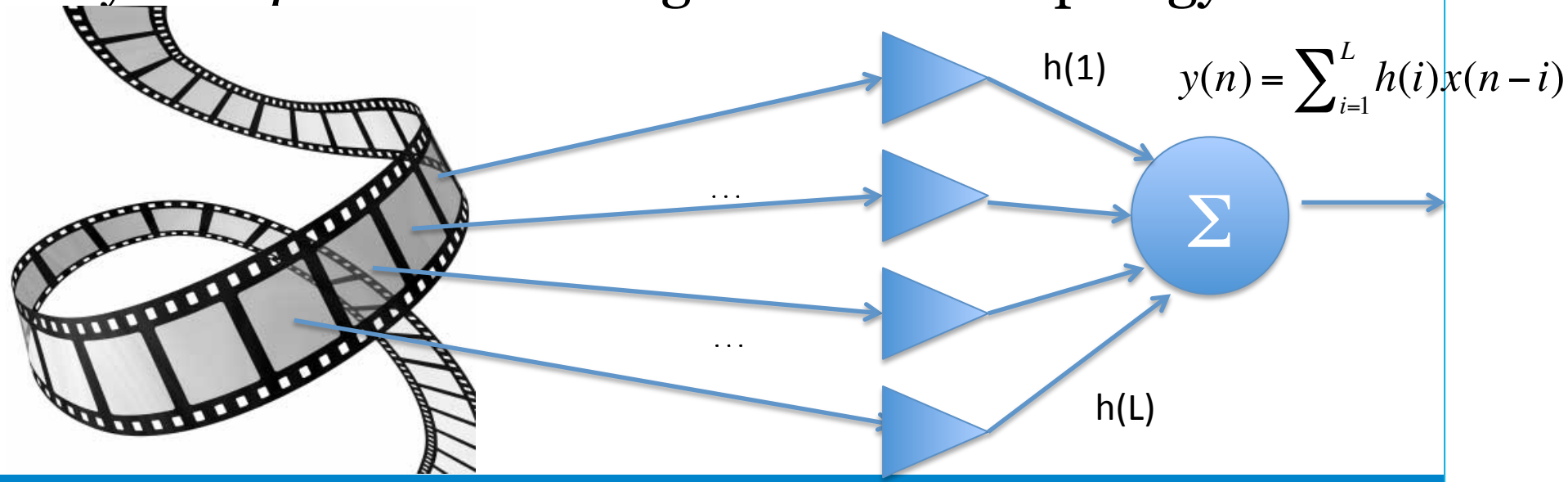
- The world and us are hugely complex dynamical systems
  - Cosmos

Seasons, Circadian cycles

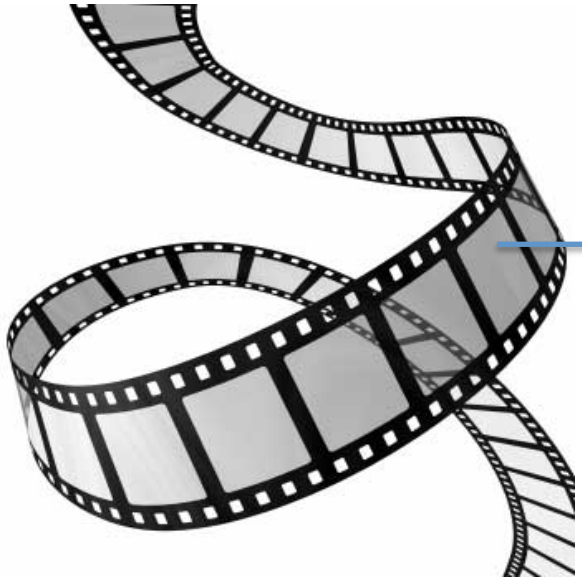
Heart sinus rhythm

Feedforward Topology

- But we keep using a finite unidirectional information flow created by finite impulse response (FIR) filter
- FIR filter, combinatorial model.
- No contex: static mapping.
- Rely on *a priori* knowledge of desired topology.

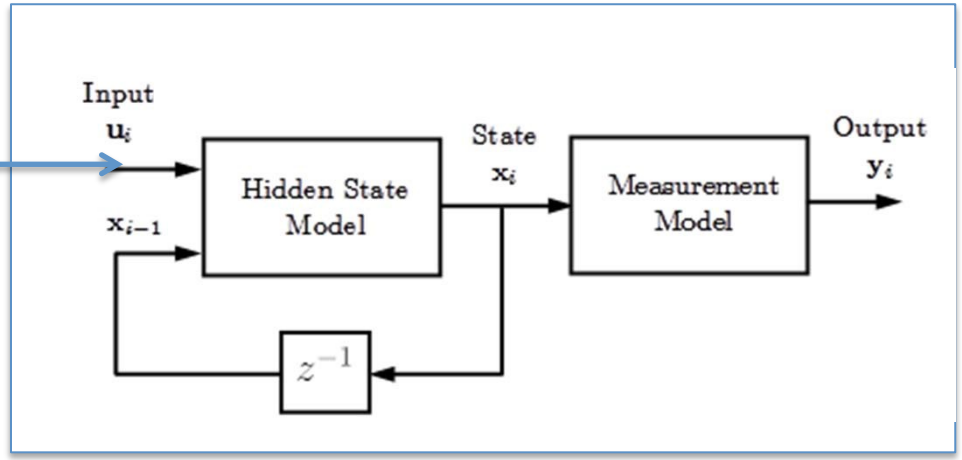


General Continuous Nonlinear State-Space Model



Current sample

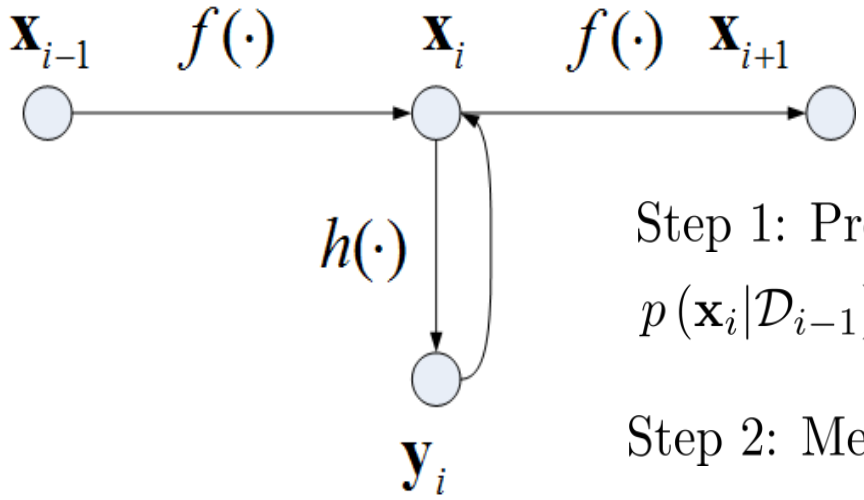
LEARNING MACHINE



$$\mathbf{x}_i = \mathbf{f}(\mathbf{x}_{i-1}, \mathbf{u}_i)$$
$$\mathbf{y}_i = \mathbf{h}(\mathbf{x}_i)$$

with input vector $\mathbf{u}_i \in \mathbb{R}^{n_u}$, state vector $\mathbf{x}_i \in \mathbb{R}^{n_x}$, output vector $\mathbf{y}_i \in \mathbb{R}^{n_y}$.

The Bayesian Filter



Step 1: Prediction

$$p(\mathbf{x}_i|\mathcal{D}_{i-1}) = \int p(\mathbf{x}_i|\mathbf{x}_{i-1})\, p(\mathbf{x}_{i-1}|\mathcal{D}_{i-1})\, d\mathbf{x}_{i-1}$$

Step 2: Measurement update

$$p(\mathbf{x}_i|\mathcal{D}_i) = \frac{p(\mathbf{y}_i|\mathbf{x}_i)\, p(\mathbf{x}_i|\mathcal{D}_{i-1})}{p(\mathbf{y}_i|\mathcal{D}_{i-1})}$$

where the denominator is calculated as

$$p(\mathbf{y}_i|\mathcal{D}_{i-1}) = \int p(\mathbf{y}_i|\mathbf{x}_i)\, p(\mathbf{x}_i|\mathcal{D}_{i-1})\, d\mathbf{x}_i$$

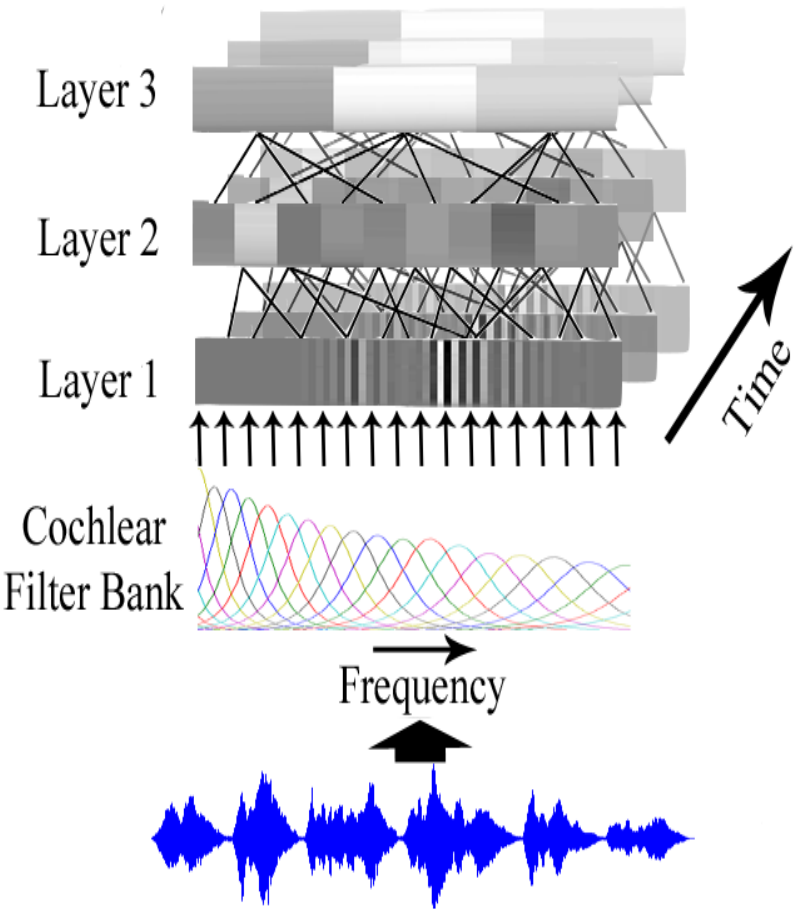
State model: Propagate distribution of hidden state

Hierarchical Linear Dynamical System

- The linear model consists of one measurement equation and multiple state transition equations.

$$
\begin{aligned}
z_t &= z_{t-1} + p_t \\
u_t &= G u_{t-1} + D z_{t-1} + r_t \\
x_t &= F x_{t-1} + B u_{t-1} + w_t \\
y_t &= H x_t + v_t
\end{aligned}
$$

- By design the top layer creates point attractors (Brownian state) to extract redundancies in the sound time structure by slowing down the top layer dynamics.
- The nested HLDS is driven bottom-up by the observations, and top-down by the states so indirectly it segments the input in spectral uniform regions.



Cinar G., Príncipe J., "Clustering of Time Series Using a Hierarchical Linear Dynamical System", in Proc. ICASSP 2014, Florence, Italy

State Estimation in Joint Space

- We can re-write the nested dynamics as follows:

$$\begin{aligned} \tilde{X}_t &= \tilde{F}\tilde{X}_{t-1} + \tilde{W}_t \\ y_t &= \tilde{H}\tilde{X}_t + v_t \end{aligned}$$

Equivalent to a single layer linear model!

$$where \quad \tilde{X}_t = \begin{bmatrix} z_t \\ u_t \\ x_t \end{bmatrix}, \tilde{F} = \begin{bmatrix} I & \mathbf{0} & \mathbf{0} \\ D & G & \mathbf{0} \\ \mathbf{0} & B & F \end{bmatrix}$$

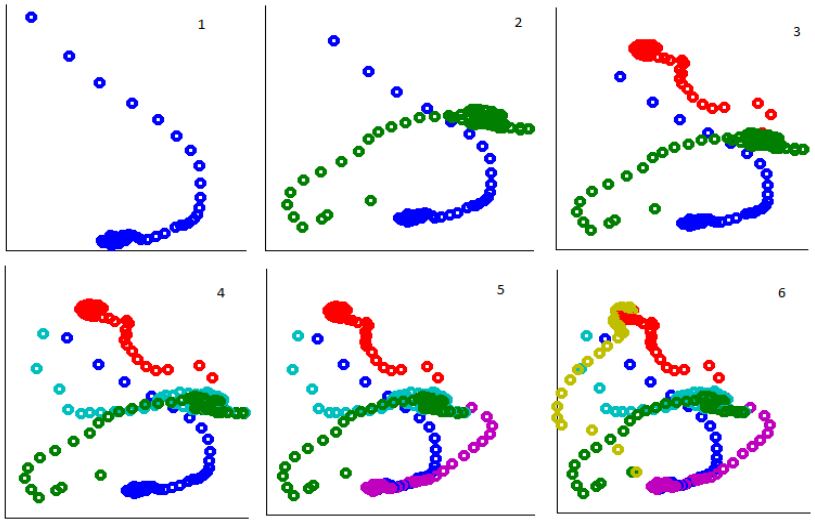
Constraints naturally enforced by design

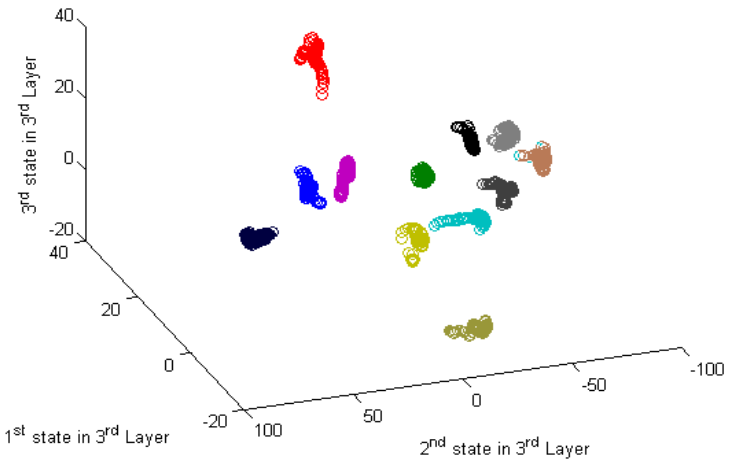
$$and \quad \tilde{H} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & H \end{bmatrix}, \tilde{W}_k = \begin{bmatrix} p_k \\ r_k \\ w_k \end{bmatrix}$$

- These equations define a joint state-space where we can do the estimation of all the hidden states in all the layers simultaneously.
- Therefore we can use the unconstrained cost function for inference and exploit the computational efficiency of the Kalman Filter.

Point attractors for Trumpet Notes

- Train with audio samples from Univ. of Iowa Musical Instrument notes (2 sec sustained notes) in the range E3-D6 for the nonvibrato Trumpet.
- The algorithm organizes in a self organizing fashion the different time structure of notes into point attractors in the state space of the highest layer (Hopfield network).





Monophonic/Chord Note Classification

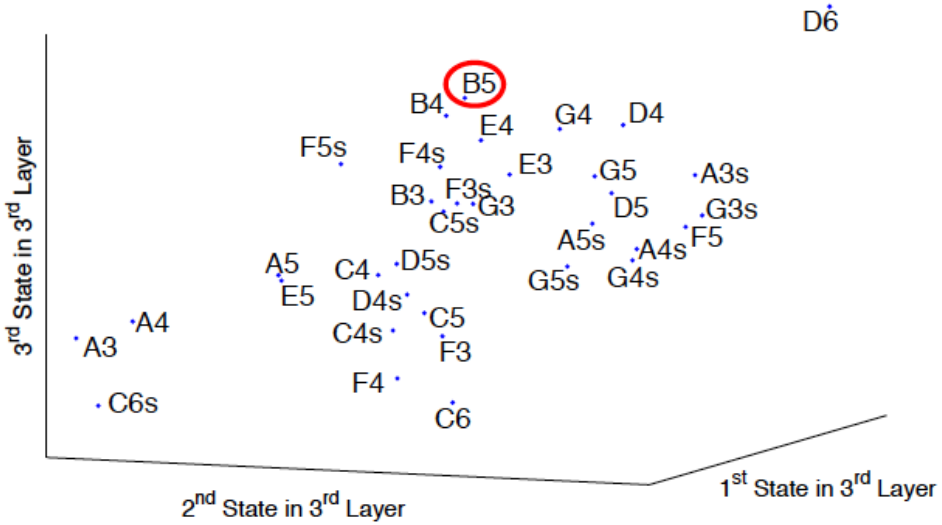
THE TABLE THE PERFORMANCE OF HLDS, SWIPE AND YIN ON MONOPHONIC NOTE CLASSIFICATION TESTED WITH MULTIPLE MUSICAL INSTRUMENTS.

Instrument	Violin G-string	Bass Trombone	Tenor Trombone	Bass Clarinet	Flute	Trumpet
HLDS	93.88%	90.41%	92.70%	86.94%	82.66%	96.61%
SWIPE'	90.63%	89.11%	92.86%	84.60%	77.91%	95.45%
YIN	51.20%	80.64%	90.40%	82.19%	78.74%	93.71%
Instrument	Bb Clarinet	Tuba	Eb Clarinet	Oboe	Horn	
HLDS	90.57%	87.86%	91.53%	94.22%	97.10%	
SWIPE'	82.94%	86.23%	83.03%	88.48%	96.16%	
YIN	82.52%	75.45%	81.20%	77.24%	90.06%	

THE TABLE SHOWS THE COMPARISON OF THE CLASSIFACTION ACCURACY FOR ISOLATED CHORDS USING DIFFERENT METHODS.

	HLDS	NMF
Isolated Chord Classification Accuracy	93.45%	91.45%

Advantage of Continuous State Space

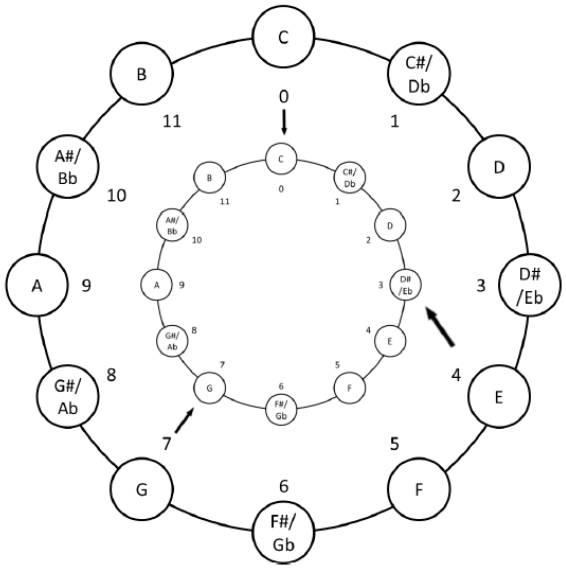


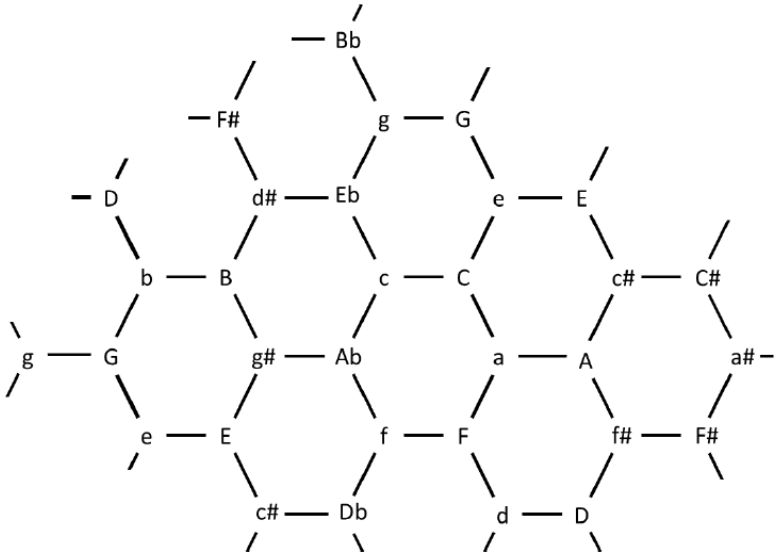
- Discovery of Notes

We train 7 models (s = 3, k = 10) **leaving out one note** (B5). How would the model classify the missing note?

- The model chooses notes that are musically close to B5, i.e. the model assigns either other octaves of B, or notes that are related as perfect fifths to B
- **The model also generalizes from the trumpet to the saxofone**.
- **We conclude that HLDS learned the metric of the music space**.

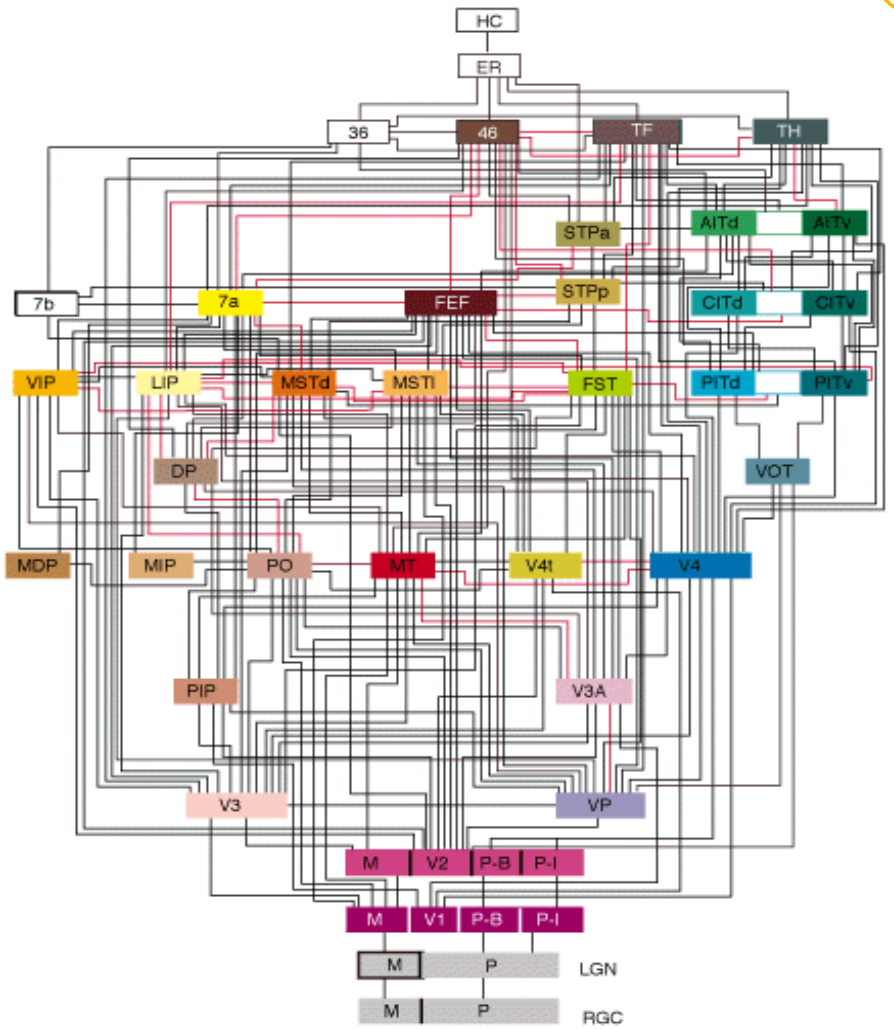
Testing Musical Distances with HLDS





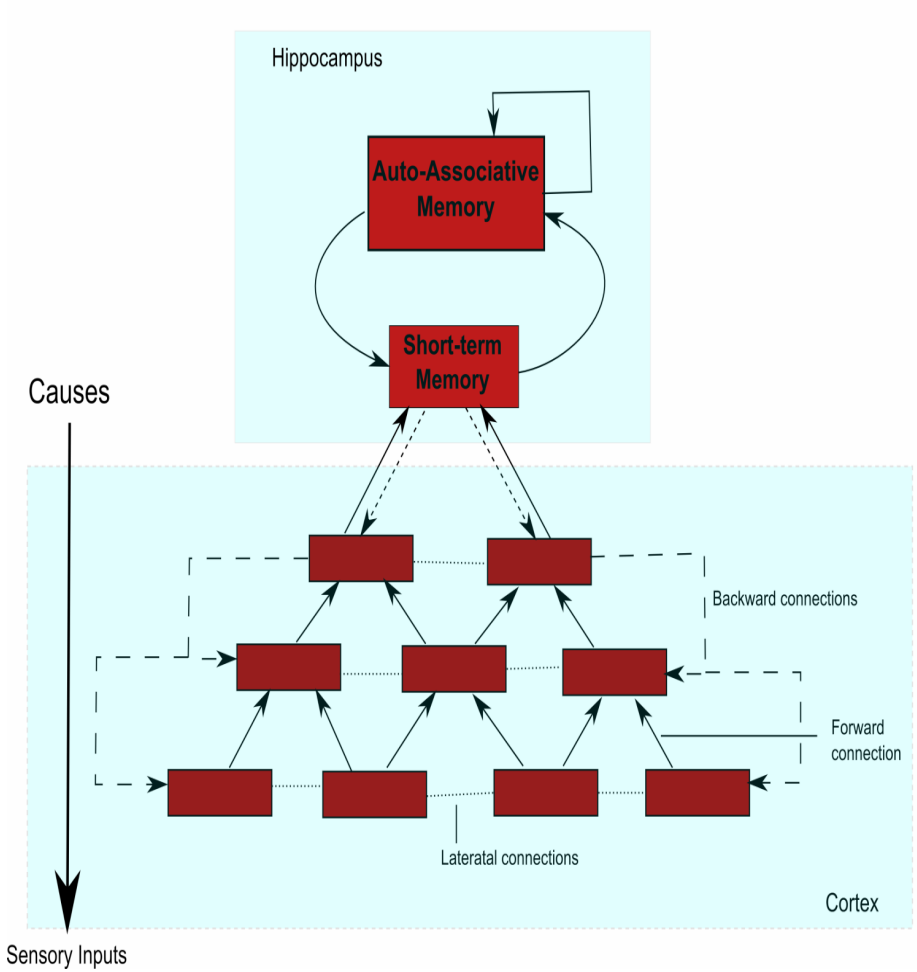
- **Voice-leading space** where pitches are represented by the logarithms of their fundamental frequencies (pitches are close if they are neighbors on the piano keyboard). Hence the distance is measured according to the usual metric on $\mathfrak{R}$.
- **Tonnetz space** is based on acoustics (fundamental and harmonics) with notes places in hexagons (tiling of 2 D space).
- **They do not always agree**: Based on the Riemannian Tonnetz, C major is closer to F major, whereas it is closer to F minor based on the voice-leading distance.
- **Model agrees most often with Tonnetz** (10 from 15 models)

Neural Anatomy of the Visual System



- We share Helmholtz' view that cortical function evolved to explain sensory inputs. As such we seek to understand the role of processing and stored experience in a machine learning framework for the decoding of sensory input.

Cognitive Architecture for Object Recognition in Video



Goal:

Develop a bidirectional, dynamical, adaptive, self-organizing, distributed and hierarchical model for sensory cortex processing using approximate Bayesian inference.

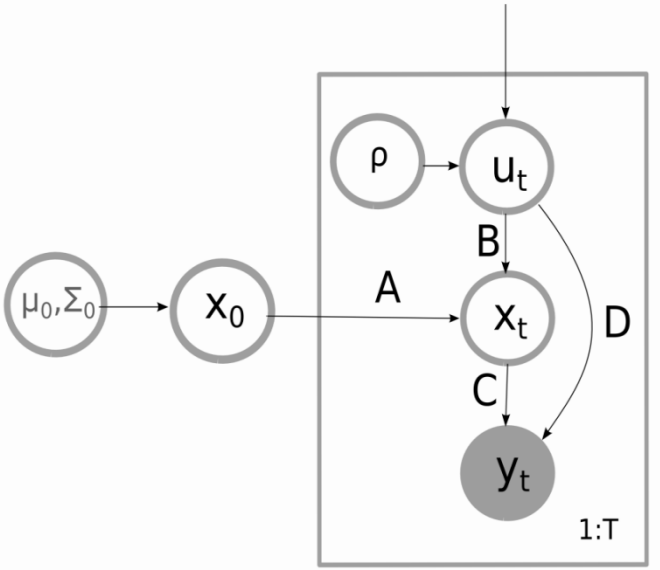
Principe J. Chalasani R., "Cognitive Architecture for Sensory Processing", Proceedings of the IEEE, vol 102, #4, 514-525, 2014

Sensory Processing Functional Principles

Generalized state space model with additive noise:

y_t – Observations
x_t – Hidden states
u_t – Causal states

$$y_t = Cx_t + Du_t + n_t$$
$$x_t = Ax_{t-1} + Bu_t + v_t$$

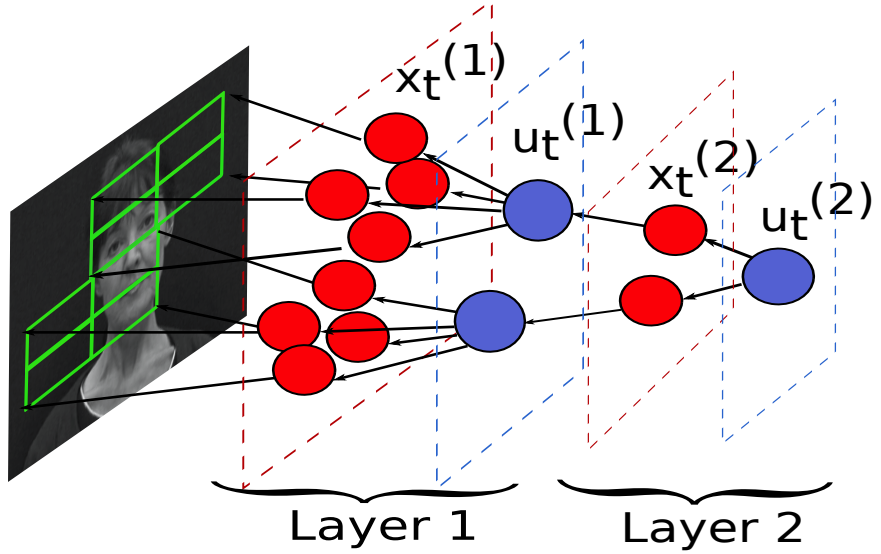


Hidden states model the history and the internal state.

Causes model the "inputs" driving the system .

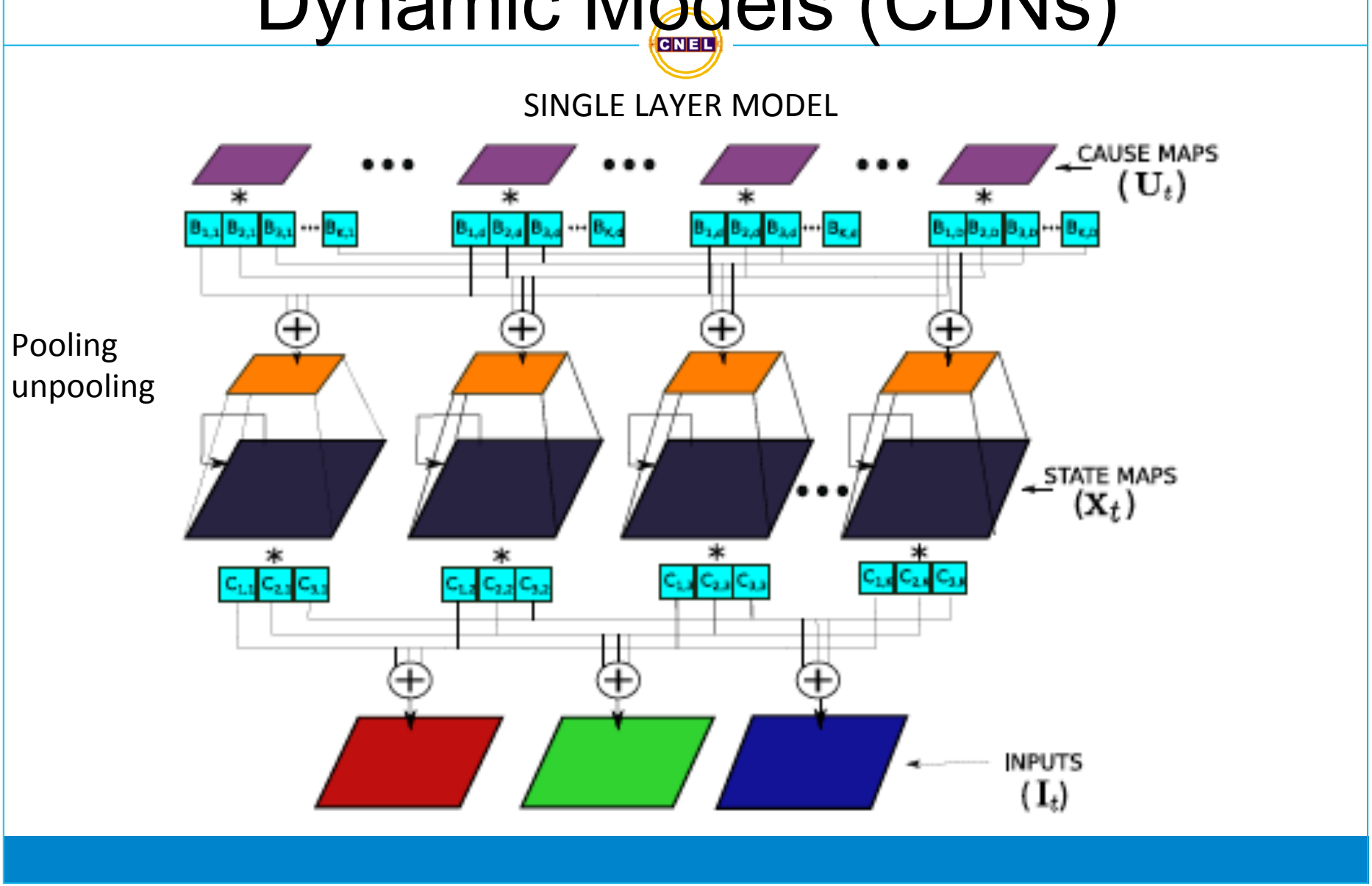
Empirical Bayesian priors create a hierarchical model, the layer on the top tries to predict the causes for the layer below.

Multi-Layered Architecture



- Tree structure with tiling of scene at bottom
- Computational model is uniform within layer and across
- Different spatial scales due to pooling which also slows the time scale in upper layers
- Learning is greedy (one layer at a time)
- This creates a Markov chain across layers

Scalable Architecture with Convolutional Dynamic Models (CDNs)



Chalasani, R., and Principe, J.C, “Context Dependent Encoding with Convolutional Dynamic Network", accepted in IEEE Neural Networks and Learning Systems, 2015

Convolution Dynamic Models

- Each channel I^m_t is modeled as a linear combination of K matrices convolved with filters $C_{m,k}$

$$I_t^m = \sum_{k=1}^{K} C_{m,k} * X_t^k + N_t^m \qquad m \in \{1,2,..M\}$$

$$X_t^k(i,j) = \sum_{k'=1}^{K} a_{k,k'} X_{t-1}^{k'}(i,j) + V_t^k(i,j)$$

- $a_{k,k'}$ are the lateral connections and here we only consider self-recurrent connections ($a_{k,k'}$=1 for k=k', zero otherwise) because the application is object recognition

Convolutional Dynamic Models

- Energy function for state maps (x is a matrix):

$$E_1(\mathbf{x}_t, \mathbf{I}_t, \mathbf{C}) = \sum_{m=1}^{M} \|\mathbf{I}_{t,m} - \sum_{k=1}^{K} C_{k,m} * x_{t,k}\|_2^2 + \lambda \|\mathbf{x}_t - \mathbf{x}_{t-1}\|_1 + \boldsymbol{\gamma} \|\mathbf{x}_t\|_1$$

- Energy function for cause maps (x is pooled):

$$E_2(\mathbf{u}_t, \mathbf{x}_t, \mathbf{B}) = \Big(\sum_{k=1}^{K} |\gamma_k \cdot x_{t,k}| \Big) + \beta \|\mathbf{u}_t\|_1$$

$$\gamma_k = \gamma_0 \left[\frac{1 + exp(-[\sum_{d=1}^{D} B_{d,k} * u_{t,d}])}{2} \right]$$

Convolution Dynamic Models

- Learning is done layer by layer starting from the bottom
- To simplify learning, we do not consider any top down connections for inference
- Filters are normalized to unit norm after learning
- The gradients are

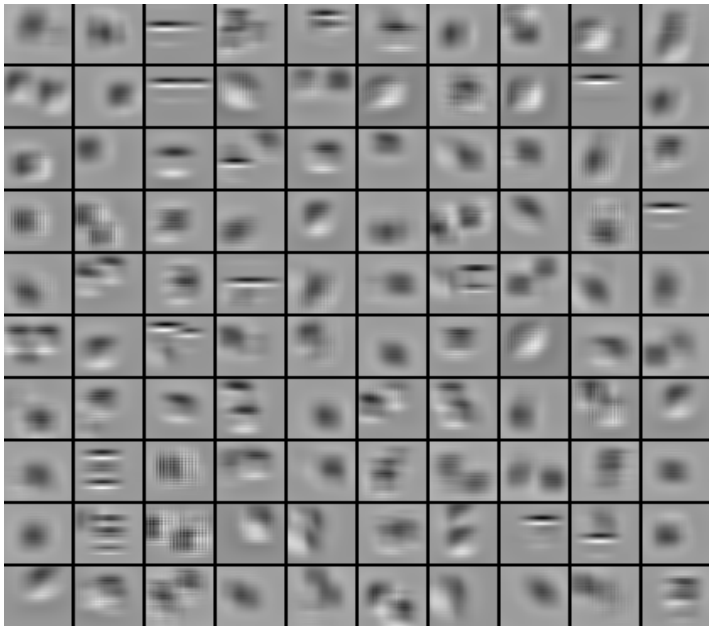
$$\nabla_{C^I_{m,k'}} E_I = -2X_t^{k',I} * (I_t^m - \sum_{k=1}^{K} C_{k,m} * X_t^{k,I})$$

$$\nabla_{B^I_{m,d'}} E_I = -U_t^{d',I} * \left[\left(\exp\{-\sum_{d=1}^{D} B_{k',m} * U_t^{d,I}\} \right) . \left| down(X_t^{k',I}) \right| \right]$$

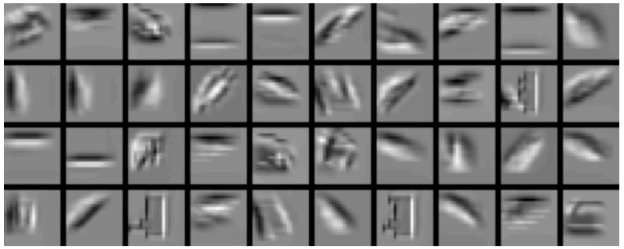
Object Recognition- Training

- Learning on Van Hateren natural video database (128x128).
- Architecture:
  - Layer 1: 16 states of 7x7 filters and 32 causes of 6x6 filters.
  - Layer 2: 64 states of 7x7 filters and 128 causes.
  - Pooling: 2 x 2 between states and causes.

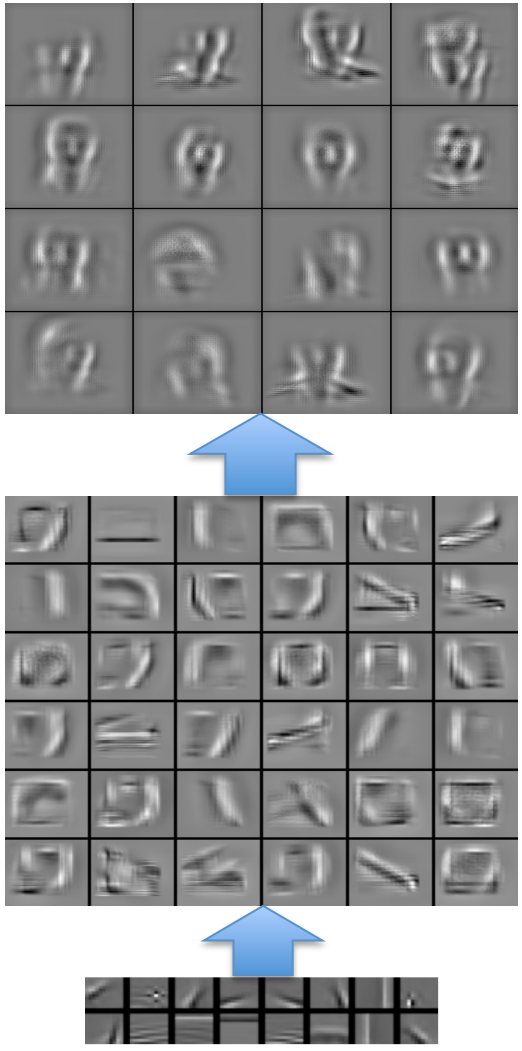
Layer 1 - Causes



Layer 1 -States



Improving Discriminability in Occlusion

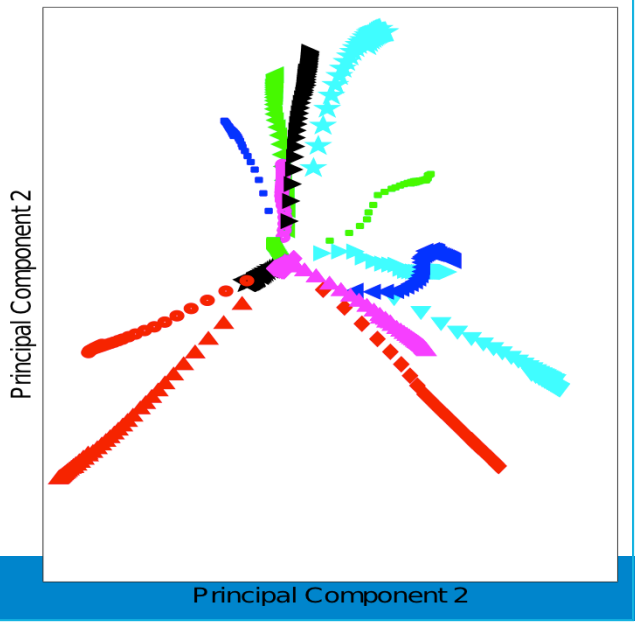


Layer -2 Causes

Example Video frames
[VidTIMIT]

Layer -1 Causes

Layer -1 States



Object Recognition with Time Context

Contextual information during inference can lead to a consistent representation of objects

- COIL-100 dataset:
  - 72 frames per object.
- Top-down inference is run over each sequence
  - We assume that the test data is partially available (4 frames) during training.
  - So called “transductive” learning.
- Four frames per object for training a linear SVM.

  (0º, 90º, 180º, 270º)

Object Recognition Results

Methods	Accuracy (%)
View-tuned network (VTU) [Wersing & Korner, 2003]	79.10 %
Convolutional Nets with temporal coherence [Mobahi et al, 2009]	92.25 %
Stacked ISA with temporal coherence [Zou et al, 2012]	87.00 %
Our method; without temporal coherence	79.45 %
Our method; with temporal coherence	94.41 %
Our method; with temporal coherence + Top-down	**98.34 %**

Testing Discriminability in Sequence Labeling

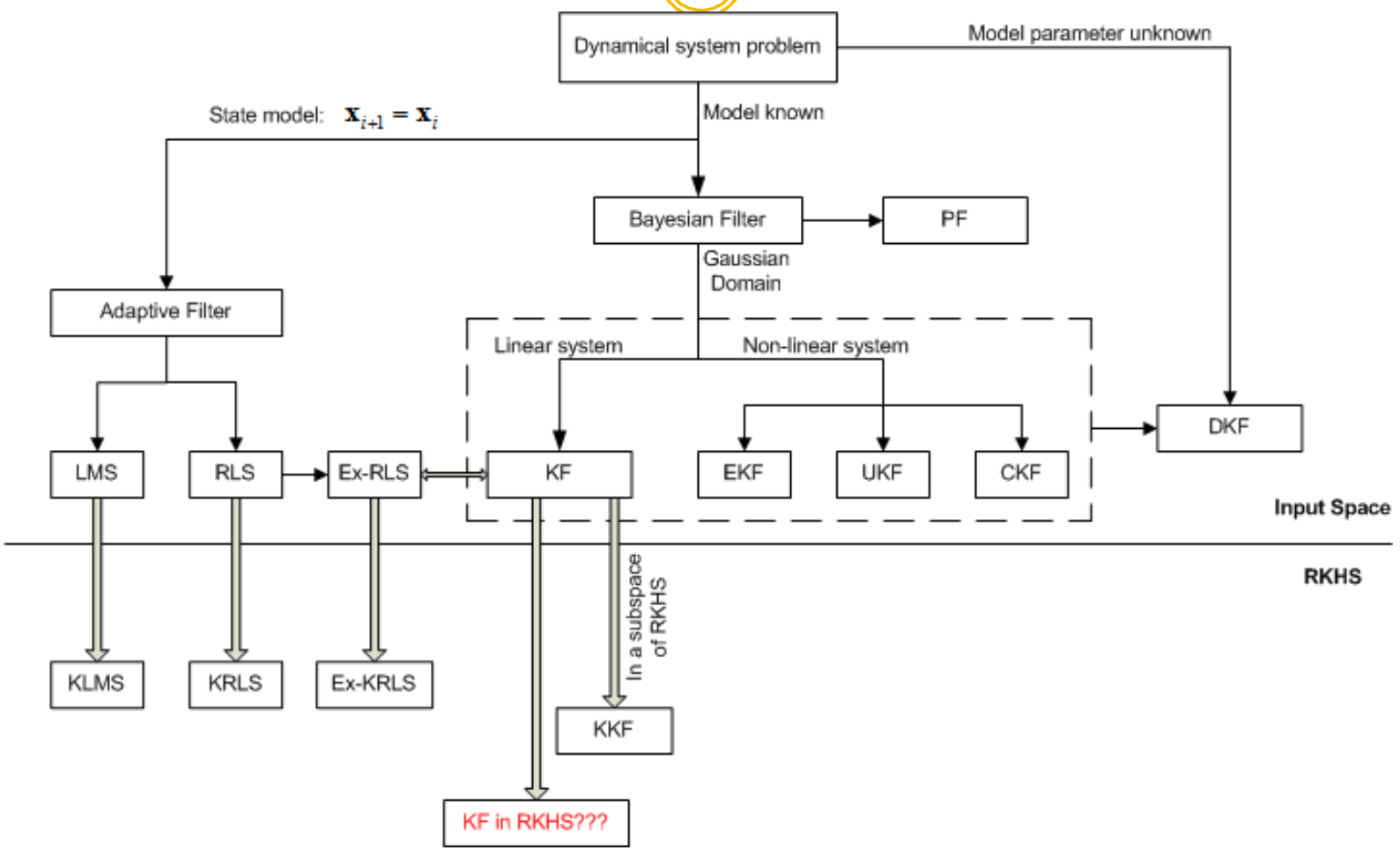
- Honda/UCSD face data set (20 for training, 39 for testing) using Viola Jones face finding algorithm (on 20x20 patches).Histogram equalization is done. 2 layer model $(16,48)_1$ $(64,100)_2$, 5x5 filters, causes concatenated as features

Sequence Lengths / Methods	50 frames	100 frames	Full length	Average
MDA [Wang and Chen, 2009]	74.36	94.87	97.44	88.89
AHISD [Cevikalp and Triggs, 2010]	87.18	84.74	89.74	87.18
CHSID [Cevikalp and Triggs, 2010]	82.05	84.62	92.31	86.33
SANP [Hu et al., 2011]	84.62	92.31	**100**	92.31
DFRV [Chen et al., 2012b]	89.74	97.44	97.44	94.87
CDN w/o context	89.74	97.44	97.44	94.87
CDN with context	**92.31**	**100**	**100**	**97.43**

Remarks

- The HLDS is easy to compute in real time but it is restricted to linear inference in the hierarchical structure
- The DCN is computationally demanding but it is quite general and results are very good.
- Hence the goal is to investigate better compromises of performance versus computational complexity

Dynamical System Modeling



Foundations of RKHS

Theorem(Moore-Aronszajn): Given any nonnegative definite function $k(x, y)$, there exists a uniquely determined (possibly infinite dimensional) Hilbert space $\mathcal{H}$ consisting of functions on $\mathcal{X}$ such that

(I) $\forall x \in \mathcal{X},\ k(x, \cdot) \in \mathcal{H}$

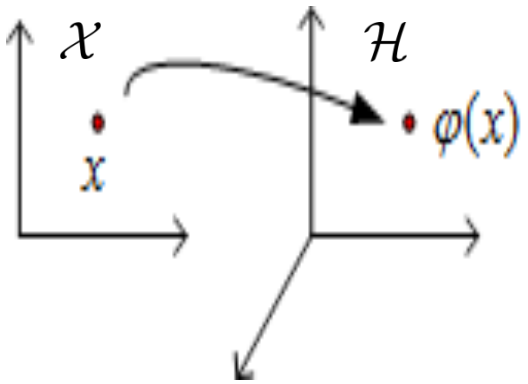
(II) $\forall x \in \mathcal{X},\ \forall f \in \mathcal{H},\ f(x) = \langle f, k(x, \cdot) \rangle_{\mathcal{H}}$. (reproducing property)

Mercer's Theorem: there are countable many nonnegative eigenvalues $\{\lambda_i : i \in \mathbb{N}\}$ and corresponding orthonormal eigenfunctions $\{\psi_i : i \in \mathbb{N}\} \in \mathcal{H}_k$ such that

$$k(x, y) := \sum_{i \in \mathbb{N}} \lambda_i \psi_i(x) \psi_i(y)^T,\ (x, y) \in \mathcal{X} \times \mathcal{X}$$

Feature map $\varphi : \mathcal{X} \to \mathcal{H}$ $\quad k(x, \cdot) := \varphi(x)$

$$\begin{aligned} k(x, y) &= \langle k(x, \cdot), k(y, \cdot) \rangle_{\mathcal{H}} \\ &= \langle \varphi(x), \varphi(y) \rangle_{\mathcal{H}} \\ \varphi(x)(j) &:= \sqrt{\lambda_j} \psi_j(x) \end{aligned}$$



Foundations of RKHS

Representer Theorem: Functions have the following representation

$$f(\cdot) = \sum_{j \in \mathbb{N}_n} \alpha_j k(x_j, \cdot), \qquad x_j \in \mathbf{X}.$$

where $\mathbb{N}_n := \{1, 2, \ldots, n\}$ and $\{\alpha_j : j \in \mathbb{N}_n\} \subset \mathbb{R}$ are parameters typically obtained from training data, $\{x_j : j \in \mathbb{N}_n\}$.

with an universal kernel, we can approximates any real-valued target function defined on a compact space arbitrarily well as the number of summands increases without bound.

Gaussian kernel: is universal defined as

$$k(x, y) = \exp\left(-\frac{\|x-y\|^2}{2\sigma^2}\right) = \exp\left(-\varrho\|x - y\|^2\right)$$

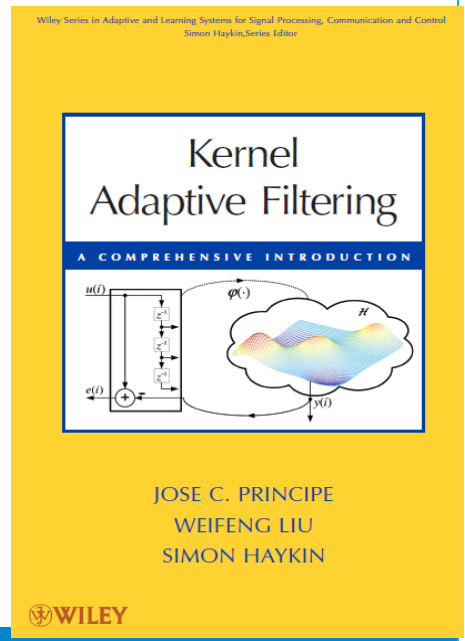
where σ is the kernel size and ϱ is the kernel parameter.

Dimensionality of $\mathcal{H}$ associated with Gaussian kernel n_k is infinite.

Conventional Kernel Approach

- Feedforward network (KLMS): partitions the input into segments of equal length and learn the nonlinear mapping between the *exemplars* and their corresponding labels.

- Inadequate generalization for modeling dynamical systems:
  - Only learns the ***static*** mapping between input-output pairs.
  - Infinite number of exemplars leads to an infinite number of weights.
  - Solution is never compact or exact.



Conventional Kernel Approach

- The simplest of the recurrent structures is the Extended Recursive Least Square (Ex-RLS) algorithm.
- We proved that its kernelized version does not allow for general modeling in RKHS using the Representer Theorem.
- We implemented a Kernel Kalman filter using statistical embedding operators, which still has high computational complexity

Zhu P., Chen B., Principe J., "Learning Nonlinear Generative Models of Time Series with a Kalman Filter in RKHS", IEEE Trans. Signal Proc, Vol 62, # 1, 141-155, 2014

General Continuous Nonlinear State-Space Model

- For simplicity we can rewrite the state-space model in terms of a new augmented hidden state vector, via concatenation

$$\mathbf{s}_i \triangleq \begin{bmatrix} \mathbf{x}_i \\ \mathbf{y}_i \end{bmatrix} = \begin{bmatrix} \mathbf{f}(\mathbf{x}_{i-1}, \mathbf{u}_i) \\ \mathbf{h} \circ \mathbf{f}(\mathbf{x}_{i-1}, \mathbf{u}_i) \end{bmatrix}$$

$$\mathbf{y}_i = \mathbf{s}_i^{(n_s - n_y + 1 : n_s)} = \begin{bmatrix} \mathbf{0} & \mathbf{I}_{n_y} \end{bmatrix} \begin{bmatrix} \mathbf{x}_i \\ \mathbf{y}_i \end{bmatrix}$$

with new state vector $\mathbf{s}_i \in \mathbb{R}^{n_x + n_y}$, $\mathbf{I}_{n_y}$ is an $n_y \times n_y$ identity matrix, and $\circ$ is the function composition operator.

Theory of KAARMA

- To learn the general continuous nonlinear transition and observation functions, we map the augmented state vector and input vector into two separate RKHSs

$$\varphi(\mathbf{s}_i) \in \mathcal{H}_s \text{ and } \phi(\mathbf{u}_i) \in \mathcal{H}_u$$

- By the Representer Theorem, the new state-space model in the coupled RKHS is defined as the following set of weights (functions in the input space)

$$\Omega \triangleq \Omega_{\mathcal{H}_{su}} = \begin{bmatrix} \mathbf{g}(\cdot,\cdot) \\ \mathbf{h} \circ \mathbf{g}(\cdot,\cdot) \end{bmatrix}$$

where $\mathcal{H}_{su} \triangleq \mathcal{H}_s \otimes \mathcal{H}_u$ and $\otimes$ is the tensor-product operator.

Li Kan, Principe J., "Kernel Adaptive Auto Regressive Moving Average Algorithm", accepted in IEEE Trans. Neural Networks and Learning Systems, 2015

Theory of KAARMA

- The features in the tensor-product RKHS are

$$\psi(\mathbf{s}_{i-1}, \mathbf{u}_i) \stackrel{\Delta}{=} \varphi(\mathbf{s}_{i-1}) \otimes \phi(\mathbf{u}_i) \in \mathcal{H}_{su}$$

- The tensor product kernel is defined by

$$\begin{aligned}\langle \psi(\mathbf{s},\mathbf{u}), \psi(\mathbf{s}',\mathbf{u}') \rangle_{\mathcal{H}_{su}} &= \mathcal{K}_{su}(\mathbf{s},\mathbf{u},\mathbf{s}',\mathbf{u}') = (\mathcal{K}_s \otimes \mathcal{K}_u)(\mathbf{s},\mathbf{u},\mathbf{s}',\mathbf{u}') \\ &= \mathcal{K}_u(\mathbf{u},\mathbf{u}')\,\mathcal{K}_s(\mathbf{s},\mathbf{s}'),\end{aligned}$$

$$\text{Gaussian Kernel: } \mathcal{K}(\mathbf{u},\mathbf{u}') = \exp(-a\,\|\mathbf{u}-\mathbf{u}'\|^2)$$

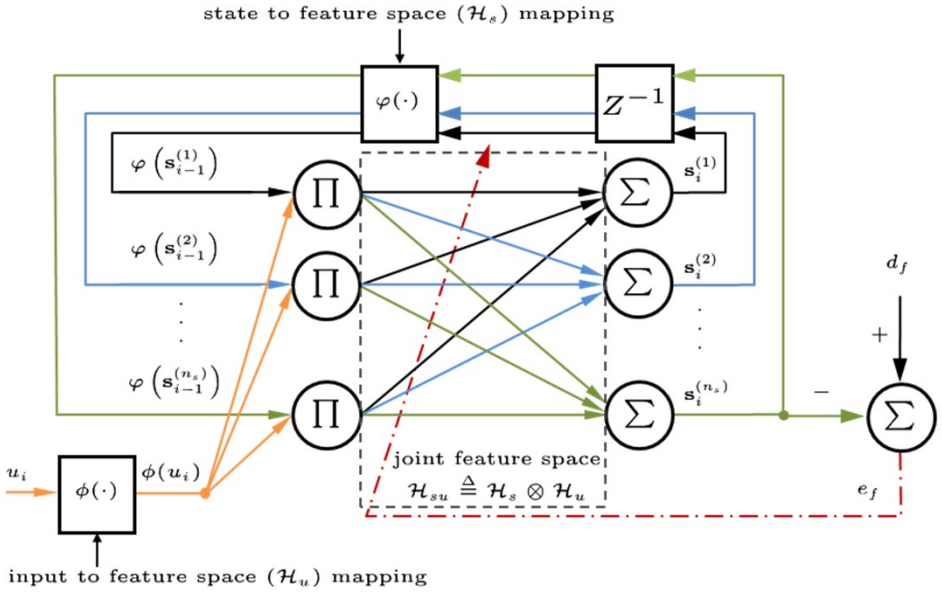
- And the kernel state-space model is expressed as

$$\begin{aligned}\mathbf{s}_i &= \Omega^T \psi(\mathbf{s}_{i-1}, \mathbf{u}_i) \\ \mathbf{y}_i &= \mathbb{I}\mathbf{s}_i\end{aligned}$$

the measurement equation $\mathbb{I} \stackrel{\Delta}{=} \begin{bmatrix} \mathbf{0} & \mathbf{I}_{n_y} \end{bmatrix}$ simplifies to a selector matrix.

Theory of KAARMA

- The general state-space model for dynamical systems is equivalent to performing linear filtering in the RKHS with a recurrent RBF network



Simple recurrent network in the RKHS with scalar input and output.

Real Time Recurrent Learning

- We evaluate the error gradient at time i with respect to the weight in the RKHS

$$\frac{\partial \varepsilon_i}{\partial \mathbf{\Omega}^{(k)}} = \frac{\partial \mathbf{e}_i^T \mathbf{e}_i}{2\partial \mathbf{\Omega}^{(k)}} = -\mathbf{e}_i^T \frac{\partial \mathbf{y}_i}{\partial \mathbf{\Omega}^{(k)}} = -\mathbf{e}_i^T \frac{\partial \mathbf{y}_i}{\partial \mathbf{s}_i} \frac{\partial \mathbf{s}_i}{\partial \mathbf{\Omega}^{(k)}} \qquad \text{where } \frac{\partial \mathbf{y}_i}{\partial \mathbf{s}_i} = \mathbb{I}.$$

- We expand the state gradient using the product rule

$$\frac{\partial \mathbf{s}_i}{\partial \mathbf{\Omega}^{(k)}} = \frac{\partial \mathbf{\Omega}^T \psi(\mathbf{s}_{i-1}, \mathbf{u}_i)}{\partial \mathbf{\Omega}^{(k)}} = \mathbf{\Omega}^T \frac{\partial \psi(\mathbf{s}_{i-1}, \mathbf{u}_i)}{\partial \mathbf{\Omega}^{(k)}} + \mathbf{I}_{n_s}^{(k)} \psi(\mathbf{s}_{i-1}, \mathbf{u}_i)^T$$

where $\mathbf{I}_{n_s}^{(k)}$ is the k-th column of the $n_s \times n_s$ identity matrix.

Real Time Recurrent Learning

- Using the Representer Theorem, the weight at time *i* is a linear combination of the prior mappings

$$\mathbf{\Omega}_i = \mathbf{\Psi}_i \mathbf{A}_i$$

where $\mathbf{\Psi}_i \triangleq [\psi(\mathbf{s}_{-1}, \mathbf{u}_0), \cdots, \psi(\mathbf{s}_{m_i-2}, \mathbf{u}_{m_i-1})]$.

- Using substitution and applying the chain rule, we obtain

$$\mathbf{\Omega}^T \frac{\partial \psi(\mathbf{s}_{i-1}, \mathbf{u}_i)}{\partial \mathbf{\Omega}^{(k)}} = \mathbf{A}_i^T \frac{\partial \mathbf{\Psi}_i^T \psi(\mathbf{s}_{i-1}, \mathbf{u}_i)}{\partial \mathbf{s}_{i-1}} \frac{\partial \mathbf{s}_{i-1}}{\partial \mathbf{\Omega}_i^{(k)}}$$

$$= 2a_s \mathbf{A}_i^T \mathbf{K}_i \mathbf{D}_i^T \frac{\partial \mathbf{s}_{i-1}}{\partial \mathbf{\Omega}_i^{(k)}}$$

$$\triangleq \mathbf{\Lambda}_i \frac{\partial \mathbf{s}_{i-1}}{\partial \mathbf{\Omega}_i^{(k)}}$$

$\mathbf{K}_i = \mathrm{diag}(\mathbf{\Psi}_i^T \psi(\mathbf{s}_{i-1}, \mathbf{u}_i))$ is the diagonal matrix of kernel-evaluation products

$\mathbf{D}_i = [(\mathbf{s}_{-1} - \mathbf{s}_{i-1}), \cdots, (\mathbf{s}_{m_i-2} - \mathbf{s}_{i-1})]$ is the state difference matrix

Real Time Recurrent Learning

- Finally, we obtain the following recursion

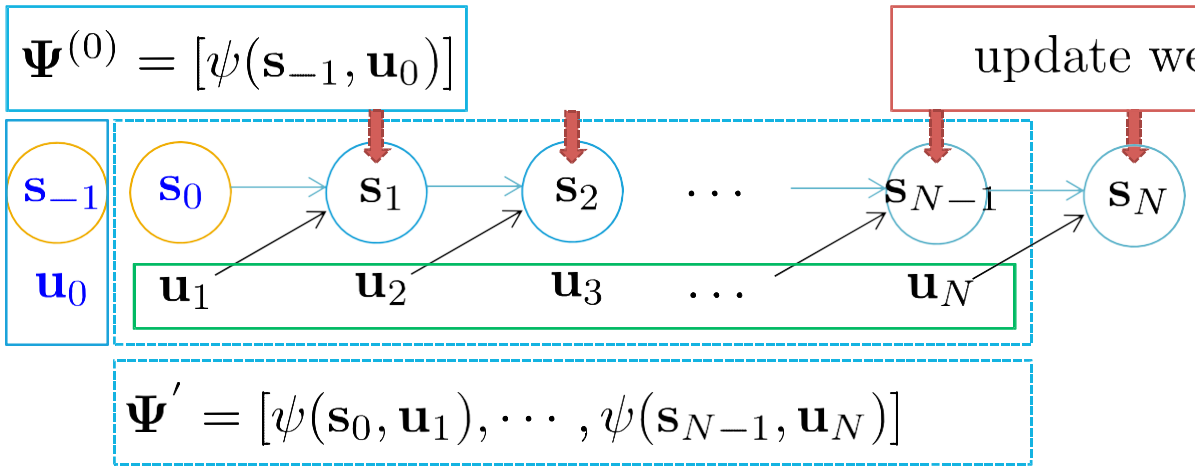
$$\frac{\partial \mathbf{s}_i}{\partial \mathbf{\Omega}_i^{(k)}} = \boldsymbol{\Lambda}_i \frac{\partial \mathbf{s}_{i-1}}{\partial \mathbf{\Omega}_i^{(k)}} + \mathbf{I}_{n_s}^{(k)} \psi(\mathbf{s}_{i-1}, \mathbf{u}_i)^T.$$

- Since the state gradient is independent of the error (future), we can forward propagate it using the initialization

$$\frac{\partial \mathbf{s}_0}{\partial \mathbf{\Omega}_i^{(k)}} = \mathbf{0}$$

since the initial state is functionally independent of the filter weights.

Complexity: Regression



$\mathbf{\Psi}^{(i+1)} = [\mathbf{\Psi}^{(i)}, \mathbf{\Psi}']$ Updated Dictionary

Regression (update every new sample/symbol)

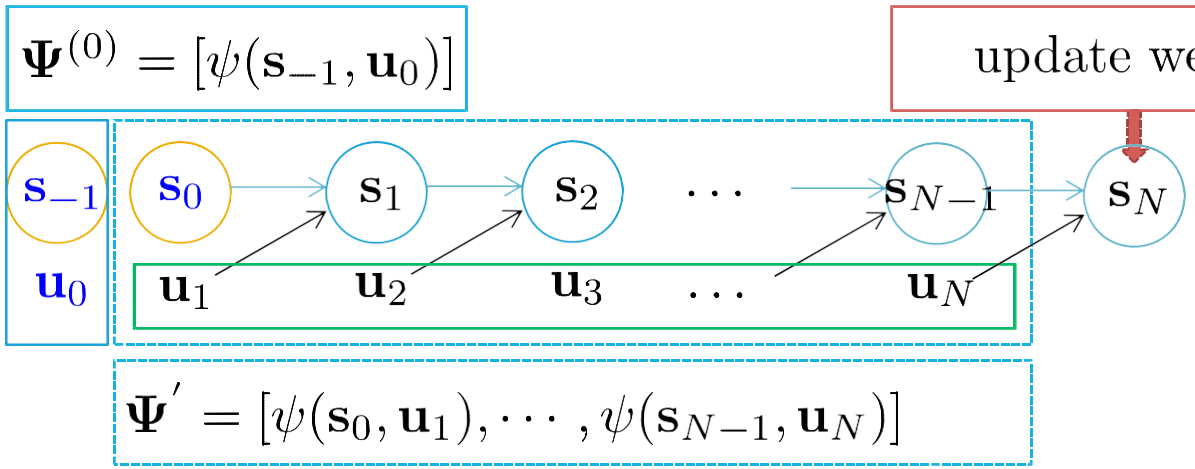
Sum of bases is a triangular number $\sum_{k=1}^{N} k = \frac{N(N+1)}{2}$

Memory (Training) $O(N^2)$

Computation (Training) $O(N^3)$

$\mathbf{freq} \triangleq \frac{\#update}{\#symbol} = 1$

Complexity: Classification



Hidden State

Input

New Centers

$\Psi^{(1)} = [\Psi^{(0)}, \Psi']$ Updated Dictionary

Classification (update every new sequence/string)

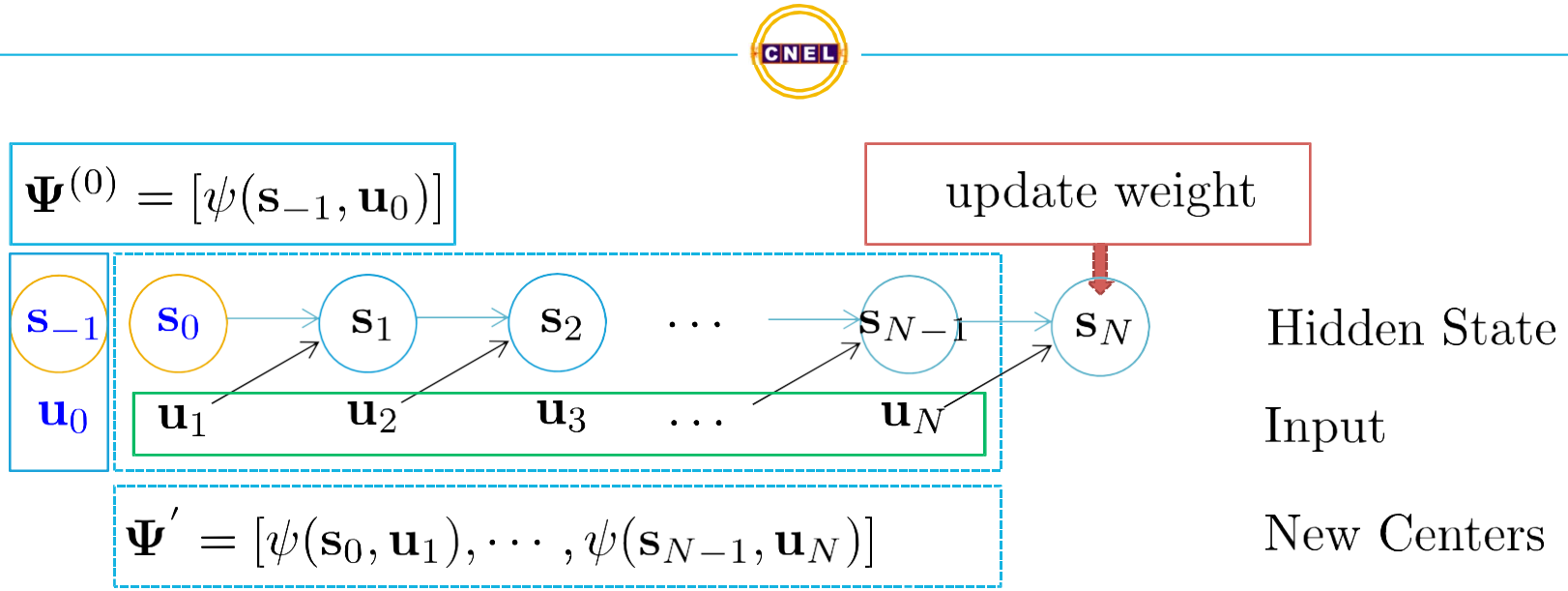
Sum of bases is linear N

Memory (Training) $O(N)$

Computation (Training) $O(N^2)$

$$\textbf{freq} \triangleq \frac{\#update}{\#symbol} = \frac{1}{N}$$

Vector Quantization on the Centers



$$\mathrm{dis}(\Psi'^{(i)}, \Psi) \triangleq \min_{1 \le j \le m_s} \sqrt{a_s \left\| \mathbf{s}'^{(i)} - \mathbf{s}^{(j)} \right\| + a_u \left\| \mathbf{u}'^{(i)} - \mathbf{u}^{(j)} \right\|}$$

$\Psi^{(1)} = [\Psi^{(0)}, \Psi'']$ Updated Dictionary

Chen B., Zhao P., Zhu P., Príncipe J., Quantized Kernel Least Mean Square Algorithm. IEEE Trans. Neural Netw. Learning Syst. 23(1): 22-32 (2012)

Remarks on KAARMA

- Learns the general state transition and measurement functions completely from data.
- Takes scalar input.
- Forces the state vector space into well-separated partitions
- We can use simple clustering techniques (QKAARMA) to achieve compact solutions without performance sacrifice.
- Distinct regions in the hidden state space correspond to state nodes in some finite state machine (FSM), with accepting states indicated by nonnegative response values.

DFA Synthesis from KAARMA

- Once the KAARMA correctly identifies the system, we can perform a binarization of its continuous state space to obtain a discrete finite automaton (DFA):
  - Start from the initial state, form root node.
  - For each distinct state node defined by the quantization partition, alternate the symbols of the alphabet, e.g., {0,1}, at the network input to generate the corresponding children states.
  - Repeat until no distinct sate is visited.
  - Using Moore's algorithm to eliminate non-distinguishable states, forming the minimal DFA.

KAARMA becomes a syntactic pattern recognizer

Grammatical Inference

- Identification and Reconstruction of DFA on Tomita Grammars
- Comparisons with Recurrent Neural Networks (RNN)
- Comparisons

Syntactic Pattern Recognition

- Problem: Given a set of positive and negative training sequences, describe the discriminating property of the two.

Positive Samples	**Negative** Samples
1	10
11	01
111	00
1111	011
11111	110
111111	1111110

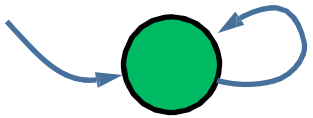
(Tomita regular grammar # 1)

Solution:

English: Accept any binary string that does not contain '0'.

Regular Expression: 1*

or Deterministic Finite Automaton (*DFA*):



1

Tomita Grammars

- Evaluate the performance of KARF using the Tomita grammars as benchmark.

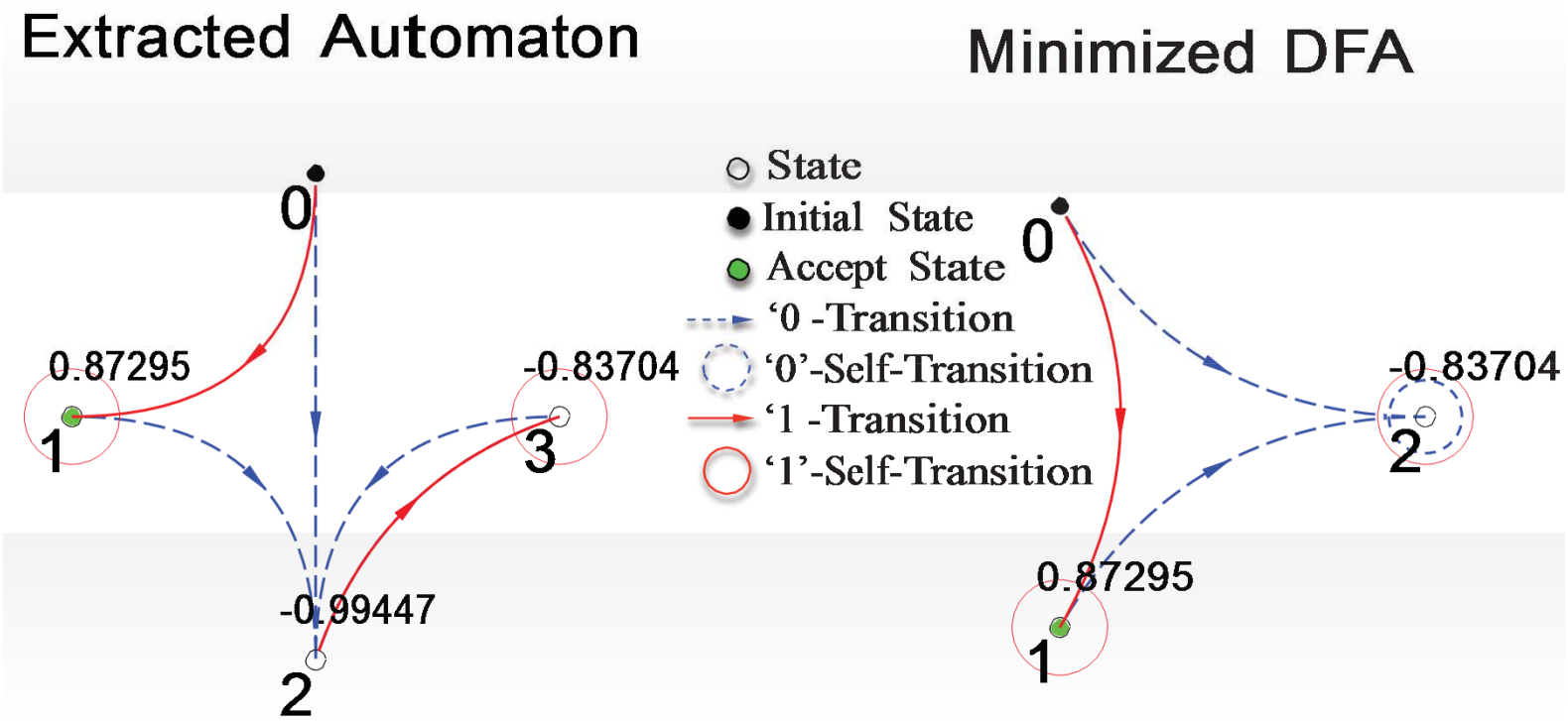
Grammar No.	Description
1	1*
2	(10)*
3	An odd number of consecutive 1's is always followed by an even number of consecutive 0's.
4	Any string with fewer than three consecutive 0's.
5	Any even length string with zero modulo-2 sum of all the bits.
6	Difference between the number of 1's and 0's is a multiple of 3.
7	0*1*0*1*

Tomita Grammars

- Training set consists of 1000 randomly generated binary strings, with lengths of 1-15 symbols (mean length is 7.758), and labeled according to grammar.
- The stimulus-response pairs are presented to the network sequentially: one bit at a time.
- At the conclusion of each string, the network weights are updated.

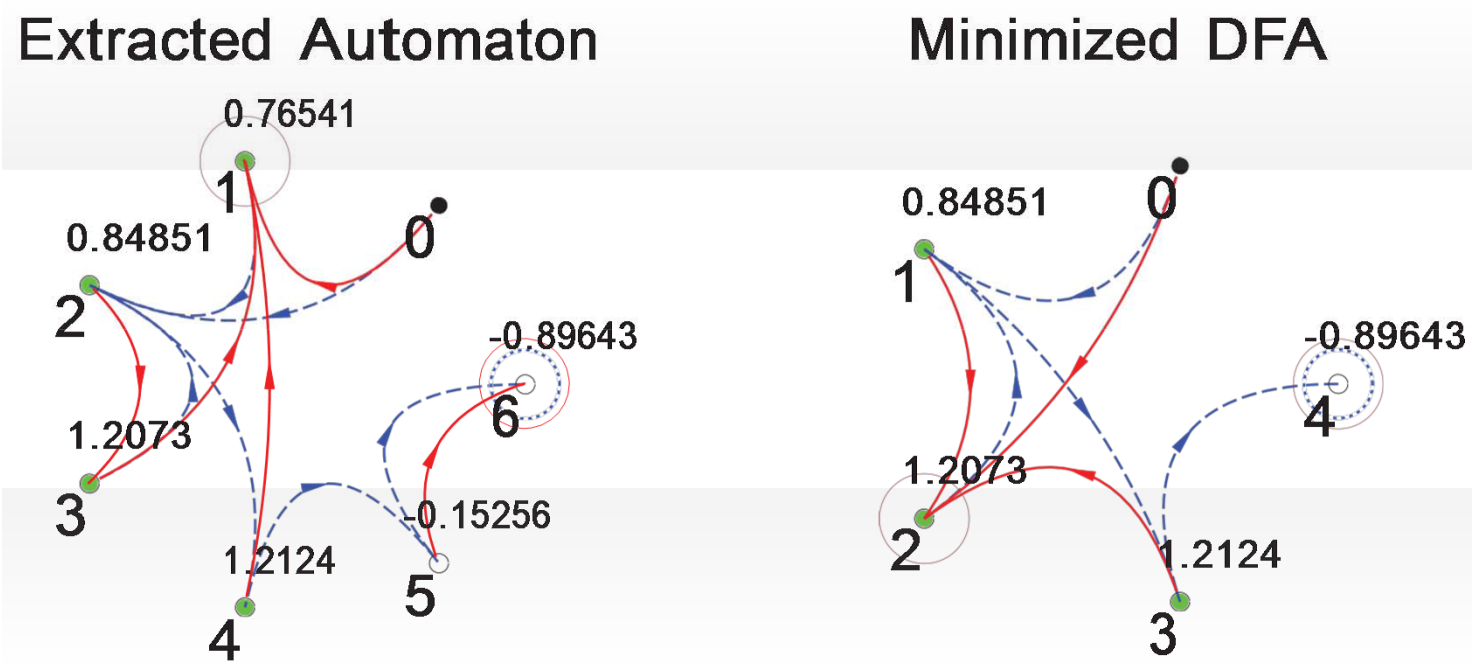
Tomita Grammars

- QKAARMA generated DFA for Tomita grammar #1.



Tomita Grammars

- QKAARMA generated DFA for Tomita grammar #4.



Tomita Grammars

- Summary of the result

Grammar	QKAARMA size	Extract. DFA	Min. DFA
#1	20	4	3
#2	22	6	4
#3	46	8	6
#4	28	7	5
#5	34	5	5
#6	28	5	4
#7	36	8	6

Hidden states $\mathbf{s} \in \mathbb{R}^3$, kernel parameters $a_s = a_u = 1$, learning rate $\eta = 0.3$, quantization threshold $q = 0.45$

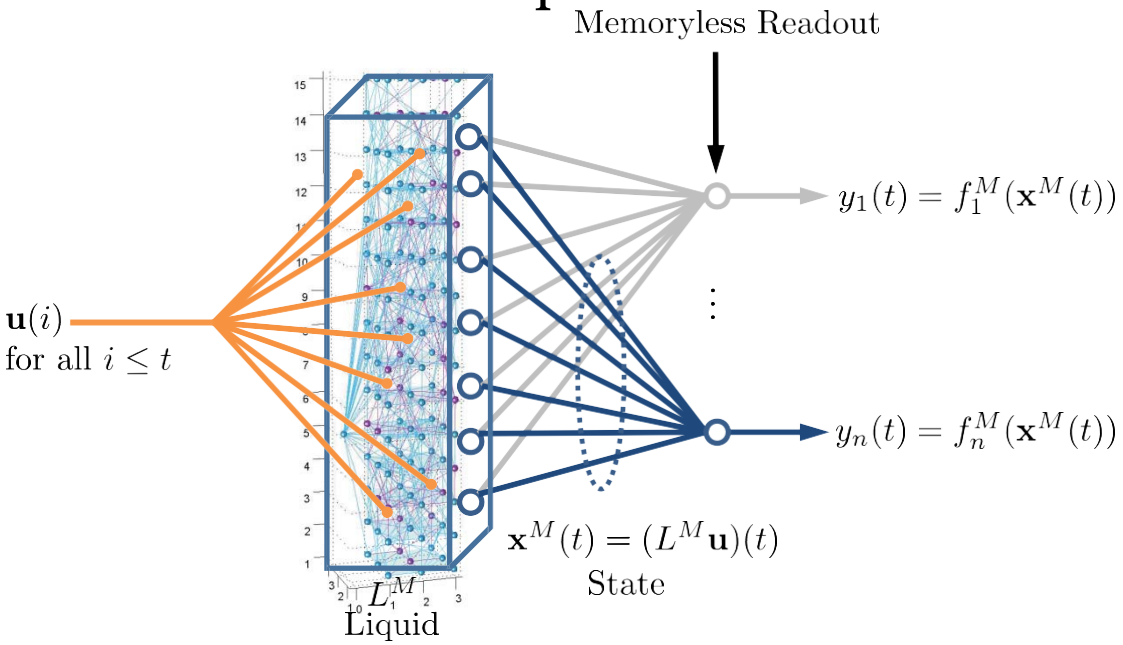
Comparison to RNN

- Performance averaged over 10 random initializations.
- RNNs are epoch trained on all binary strings of length 0-9, in alphabetical order.
- Test set consists of all strings of length 10-15 (64512 total).

	Inference Engine	train size	test error	accuracy	network size	extraction rate	DFA size
Grammar 1	QKAARMA	170	4	99.994	43.3	1.00	4.5
	RNN (Miller & Giles '93)	23000	1	99.999	9 (1st)	1.00	9.2
	RG (Schmidhuber & Hochreiter '96)	182	-	-	1 (A1)	-	-
Grammar 2	QKAARMA	700	3	99.995	29.8	1.00	6
	RNN	77000	5	99.992	9 (2nd)	1.00	9.9
	RG	1511	-	-	3 (A1)	-	-
Grammar 4	QKAARMA	900	1343	97.919	25	1.00	8.2
	RNN	46000	1240	98.078	9 (2nd)	0.81	12.3
	RG	13833	-	-	2 (A1)	-	-
Grammar 6	QKAARMA	1160	2944	95.437	36.6	1.00	5.5
	RNN	49000	8725	86.475	9 (2nd)	0.67	10.5
Grammar 7	QKAARMA	4400	4623	92.834	30.2	1.00	10.8
	RNN	121000	889	98.622	9 (2nd)	0.86	10.7

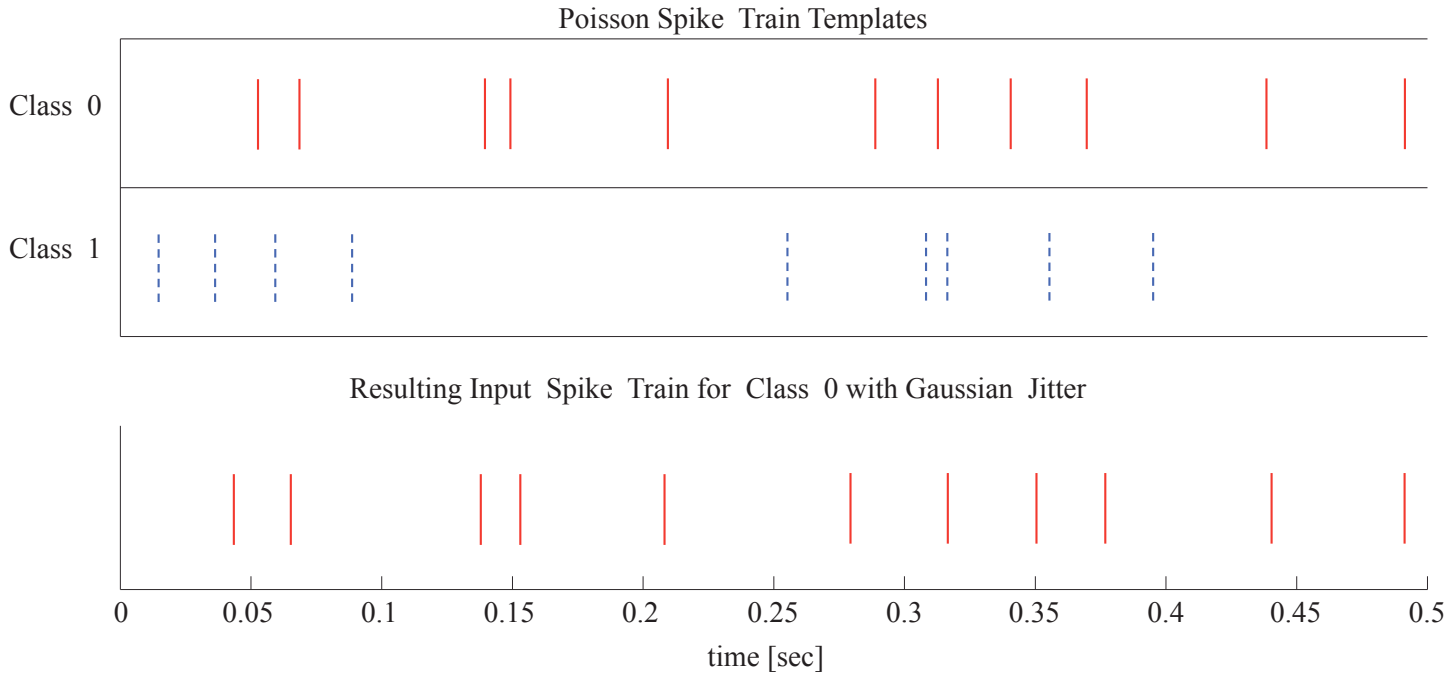
Liquid State Machine (LSM)

- RNN require significantly more data and training epochs.
- LSMs rely on fixed, randomly initialized recurrent network: dynamic reservoir.
- No stable state like an attractor: “liquid state”.



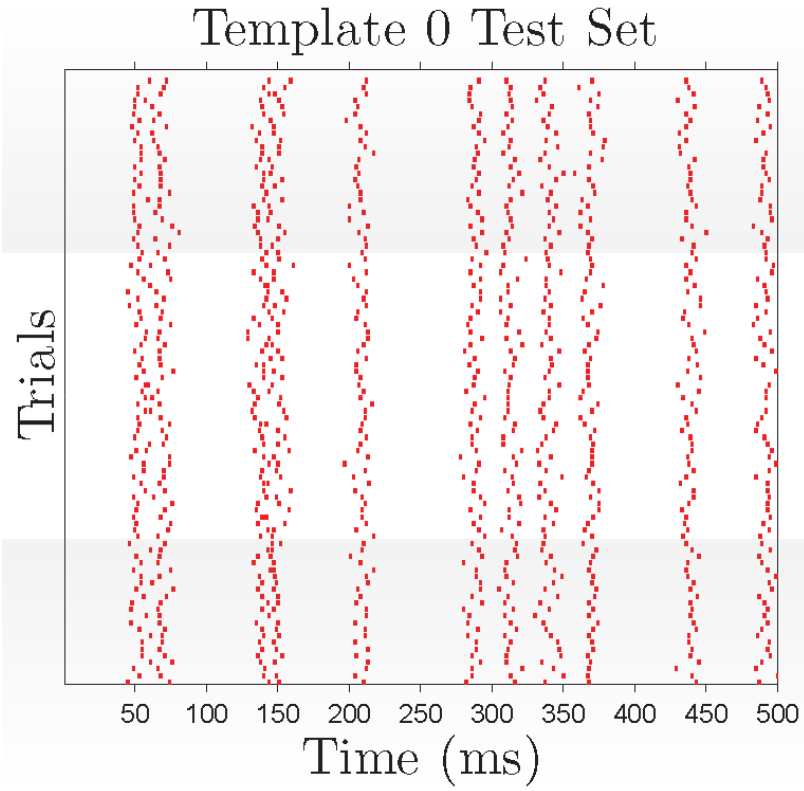
Temporal Processing

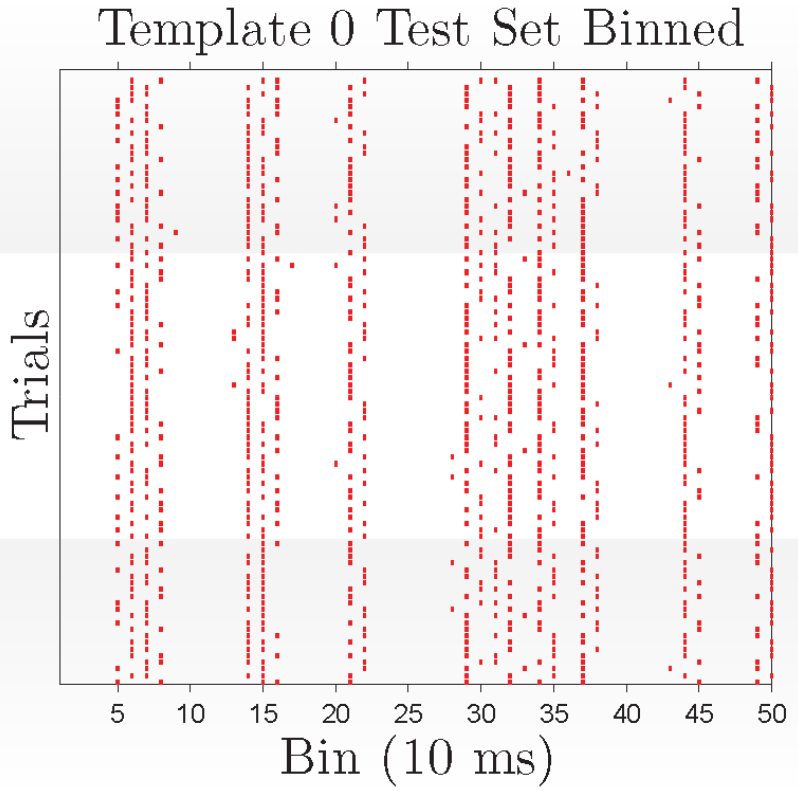
- Random displacement of spikes creates two templates: Gaussian or uniform jitter σ = 4 ms.
- As non-numeric data, there are no spatial cues to rely on.
- 500 realizations for training and 200 for testing



Temporal Processing

- Data format





LSM Performance

- Recurrent neural microcircuit comprised of 135 integrate and fire neurons (20% are inhibitory)
- State of microcircuit sampled every 25 ms by low-pass filtering the response.

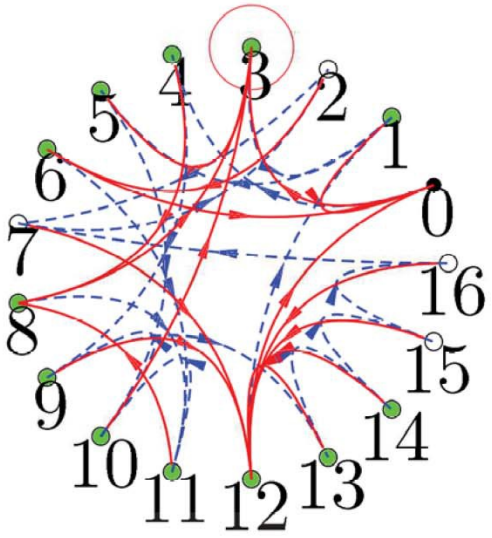
	Criteria	Linear Classification	p-Delta Rule	Linear Regression	Backpropagation
Train	CC	0.4568	0.6109	0.4773	0.7280
	MAE	0.2721	0.2533	0.4006	0.2327
	MSE	0.2721	0.1662	0.1928	0.1175
	score	0.7841	0.5773	0	0
Test	CC	0.4527	0.5652	0.3757	0.6772
	MAE	0.2710	0.2674	0.4086	0.2561
	MSE	0.2710	0.1846	0.2207	0.1353
	score	0.8052	0.6199	0	0

DFA Solution Using QKAARMA

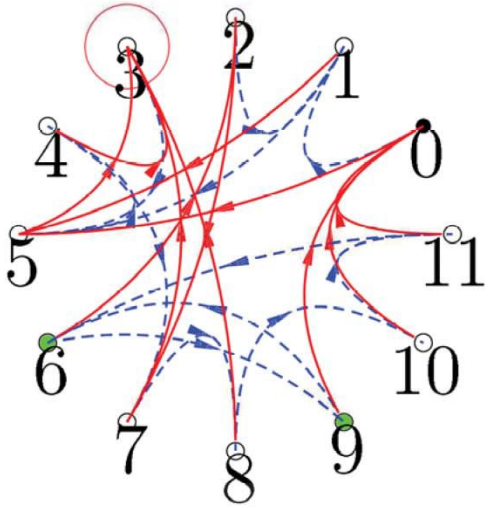
- DFA extracted from QKAARMA with 100% accuracy.

Hidden states $\mathbf{s} \in \mathbb{R}^3$, kernel parameters $a_s = a_u = 1$, learning rate $\eta = 0.1$, quantization threshold $q = 0.4$, DFA extraction quantization threshold $q_{DFA} = 0.11$

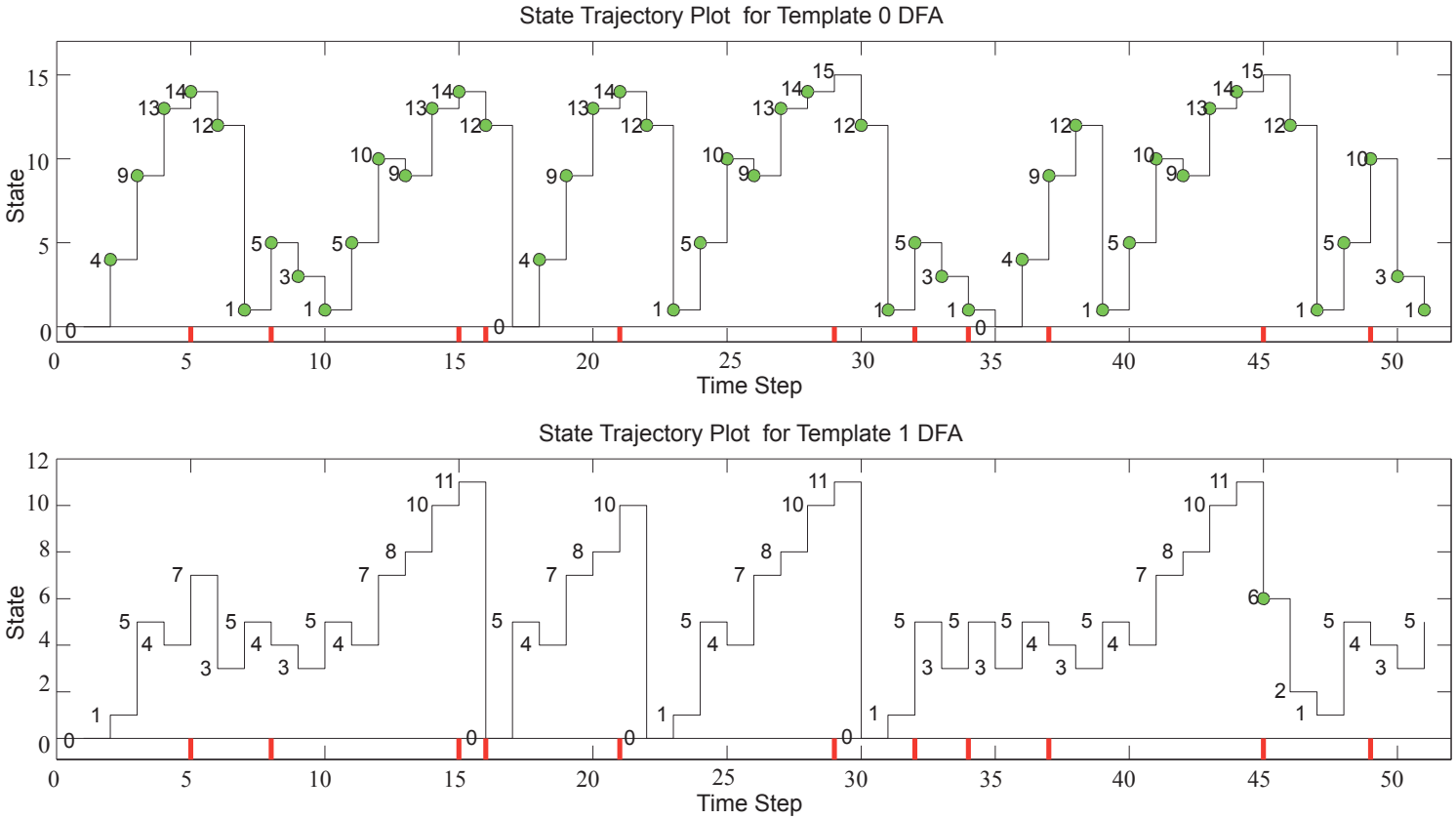
Spike Template 0



Spike Template 1



DFA Solution Using QKAARMA



Remarks

- RNN require significantly more data and training epochs than KAARMA.
- LSM has no stable states and random recurrent networks have no guarantee on performance.
- DFA provides efficient and exact solutions.
- Grammar-based solutions open the door to novel applications in neuroscience such as comparing long term firing rates of neurons associated with different behavior.

Future Work

- Feature spaces induced by Gaussian kernels are special Hilbert spaces where all evaluations are finite. However, this does not translate directly into convergent dynamics.
- For recurrent systems, this requires studies of stability that are beyond bounded-input bounded-output (BIBO) stability.
- Along with stability, a proper treatment of exploding gradients will also be pursued in the future.
- Evaluate the performances using distance measures in the RKHS, e.g., correntropy induced metric.