Theory and Applications of State Space Models for Time Series Data – a Personal View

Peter Tiňo

School of Computer Science
University of Birmingham, UK

Special thanks to ...

- Yuan Shen
- Barbara Hammer
- Alessio Micheli
- Michal Čerňanský
- Luba Beňúšková
- Jort van Mourik
- Igor Farkaš
- Ali Rodan
- Jochen Steil
- Nick Gianniotis
- Manfred Opper
- Alessandro Sperduti
- ...

Dynamical processes

Imagine a "box" inside which a dynamic process (i.e. things change in time) takes place. At time t, the "state of the system" is $\mathbf{x}(t) \in \mathcal{X}$.

The dynamics can be autonomous (e.g. "happens by itself"), or non-autonomous (e.g. driven by external input).

$$\mathbf{x}(t) = f(\mathbf{x}(t-1)), \quad \mathbf{x}(t) = f(\mathbf{u}(t), \mathbf{x}(t-1))$$

The dynamics can be continuous time (e.g. time is taken as a continuous entity - dynamics can be described e.g. by differential equations), or discrete time (e.g. changes happen in discrete time steps- can be described e.g. by iterative maps).

$$\frac{d\mathbf{x}(t)}{dt} = f(\mathbf{x}(t)), \quad \frac{d\mathbf{x}(t)}{dt} = f(\mathbf{u}(t), \mathbf{x}(t))$$

Dynamical processes - cont'd

The dynamics can be stationary (e.g. the law governing the dynamics is fixed and not allowed to change), or non-stationary (e.g. the manner in which things change is itself changing in time).

$$\mathbf{x}(t) = f_t(\mathbf{x}(t-1)), \quad \mathbf{x}(t) = f_t(\mathbf{u}(t), \mathbf{x}(t-1))$$

$$\frac{d\mathbf{x}(t)}{dt} = f_t(\mathbf{x}(t)), \quad \frac{d\mathbf{x}(t)}{dt} = f_t(\mathbf{u}(t), \mathbf{x}(t))$$

The dynamics can be deterministic (as above), or stochastic (e.g. corrupted by some form of dynamic noise). One now must work with full distributions over possible "states" of the system!

$$p(\mathbf{x}(t) \mid \mathbf{x}(t-1)), \quad p(\mathbf{x}(t) \mid \mathbf{u}(t), \mathbf{x}(t-1))$$

Observing dynamical processes

We can have a "telescope" with which to observe the "box" inside which a dynamic process is happening, e.g. we can have access to some coordinates of the dynamics, or some other readout function from them. We will call such observed entities observations.

$$\mathbf{y}(t) = h(\mathbf{x}(t)), \quad \mathbf{y}(t) = h(\mathbf{u}(t), \mathbf{x}(t))$$

Reading out of the observations can be corrupted by an observational noise. In this case we are uncertain about the value of observations and can only have distributions over possible observations.

$$p(\mathbf{y}(t) \mid \mathbf{x}(t)), \quad p(\mathbf{y}(t) \mid \mathbf{u}(t), \mathbf{x}(t))$$

State space model - state and state transition

We impose that the dynamic process we are observing is governed by a dynamic law prescribing how the state of the system evolves in time. The state captures all that we need to know about the past in order to describe future evolution of the system.

$$\mathbf{x}(t) = f(\mathbf{x}(t-1)), \quad \mathbf{x}(t) = f(\mathbf{u}(t), \mathbf{x}(t-1))$$

$$\frac{d\mathbf{x}(t)}{dt} = f(\mathbf{x}(t)), \quad \frac{d\mathbf{x}(t)}{dt} = f(\mathbf{u}(t), \mathbf{x}(t))$$

$$p(\mathbf{x}(t) \mid \mathbf{x}(t-1)), \quad p(\mathbf{x}(t) \mid \mathbf{u}(t), \mathbf{x}(t-1))$$

State space model - readout from states

We can have access to the dynamics through some function of the states producing observations.

$$\mathbf{y}(t) = h(\mathbf{x}(t)), \quad \mathbf{y}(t) = h(\mathbf{u}(t), \mathbf{x}(t))$$

$$p(\mathbf{y}(t) \mid \mathbf{x}(t)), \quad p(\mathbf{y}(t) \mid \mathbf{u}(t), \mathbf{x}(t))$$

It can be possible to "recover" the states (in some sense- e.g. up to topological conjugacy) from the observations - "observable states", or not - "unobservable states".

I only have observations...

In many cases we only have access to observations from the underlying dynamics and perhaps also to the driving input stream.

In oder to "generalize beyond the observations" we must somehow capture the law describing how the underlying dynamics evolves in time.

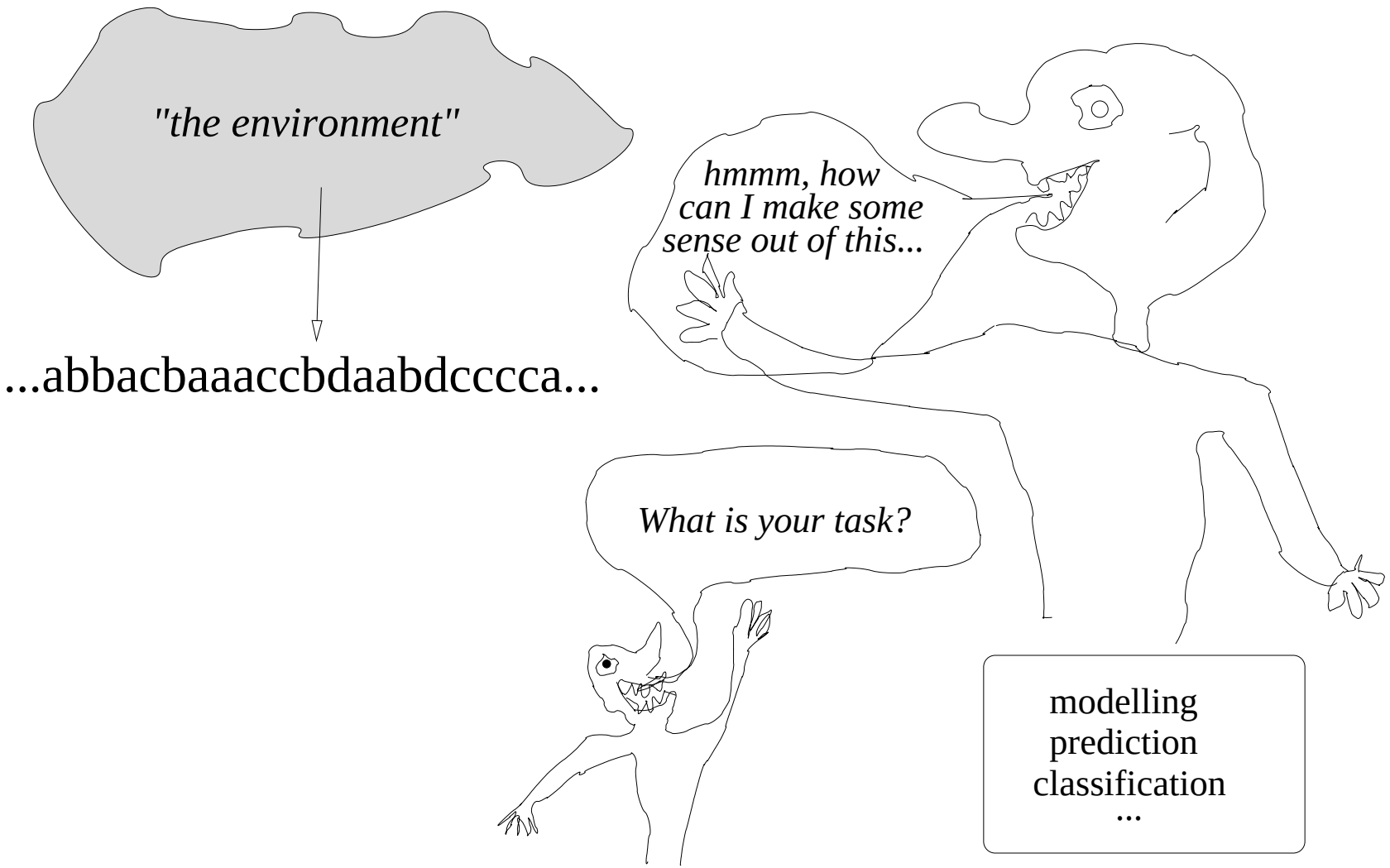
In some cases we have a deep theory underpinning such generalization processes. For example, for deterministic autonomous dynamics observed through a noiseless coordinate-projection readout, we have Taken's Theorem guaranteeing (under some rather mild conditions) recovery of the original dynamic law and its attractor (up to topological conjugacy).

Many "input time-lag window" (finite input memory) approaches take their inspiration from Taken's Theorem. The situation can get much more complicated once dynamic (and/or observational) noise is considered.

Focus of principles, discrete case

- a finite alphabet of abstract symbols $\mathcal{A} = \{1, 2, ..., A\}$
- possibly infinite strings over $\mathcal{A}$
- left-to-right processing of strings

Making sense of time series



Left-to-right processing of sequences

One can attempt to reduce complexity of the processing task

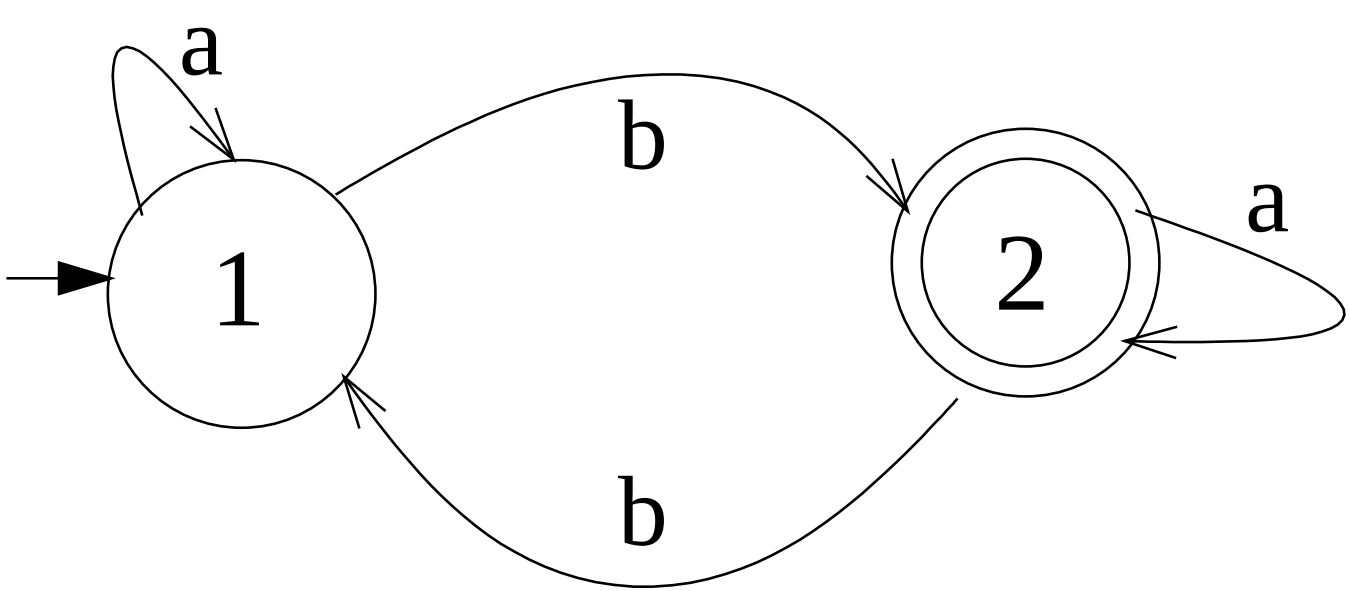
group together histories of symbols that have “the same functionality” w.r.t. the given task (e.g. next-symbol prediction)

Information processing states (IPS) are equivalence classes over sequences

IPS code what is important in everything we have seen in the past

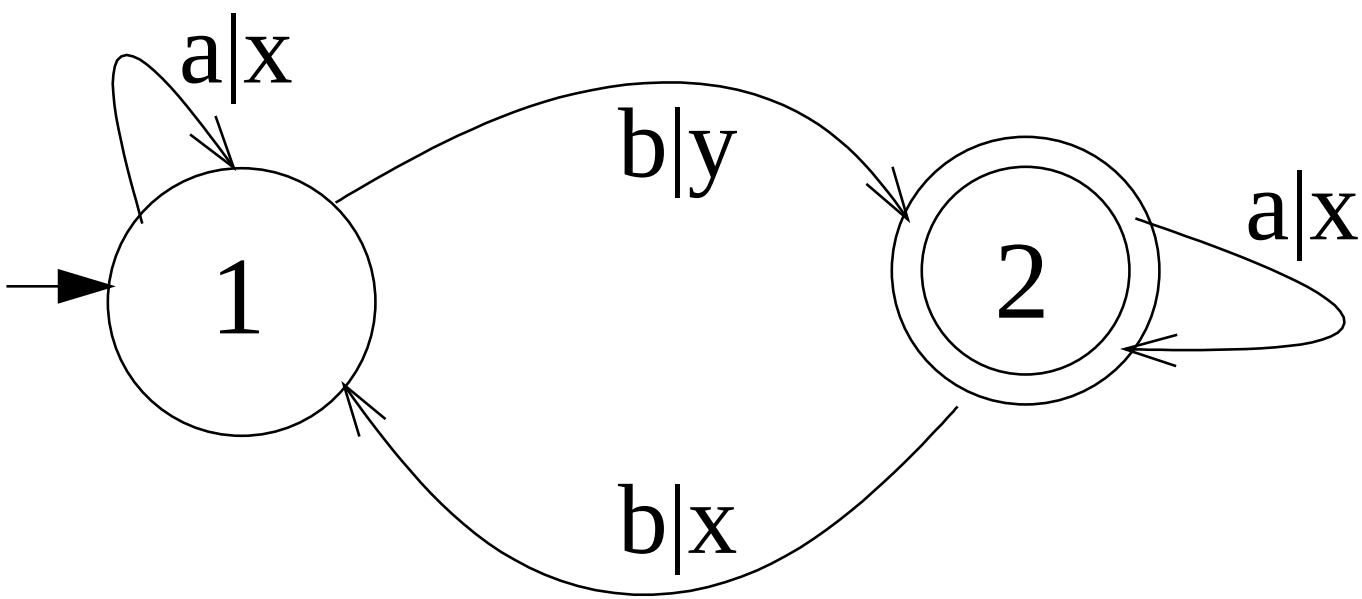
IPS - discrete state space, inputs, observations

A simple sequence classification example - FSA
IPS - $1, 2$
inputs - a, b
outputs - 2 "Grammatical": odd number of b's, 1 "Non-Grammatical": even number of b's (including none)



IPS - a simple example - transducer

translate input streams over $\{a, b\}$ to sequences over $\{x, y\}$
IPS - $1, 2$
inputs - a, b
outputs - x, y



IPS - the simplest case - finite time lag

The simplest construction of IPS is based on concentrating on the very recent (finite) past

e.g. definite memory machines, finite memory machines, Markov models

Example:

Only the last 5 input symbols matter when reasoning about what symbol comes next

... 1 2 1 1 2 3 2 1 2 1 2 4 3 2 1 1
... 2 1 1 1 1 3 4 4 4 4 1 4 3 2 1 1
... 3 3 2 2 1 2 1 2 2 1 3 4 3 2 1 1

All three sequences belong to the same IPS "43211"

Probabilistic framework - Markov model (MM)

What comes next?

... 1 2 1 1 2 3 2 1 2 1 2 4 3 2 1 1 | ?

... 2 1 1 1 1 3 4 4 4 4 1 4 3 2 1 1 | ?

... 3 3 2 2 1 2 1 2 2 1 3 4 3 2 1 1 | ?

finite context-conditional next-symbol distributions

$P(s \mid 11111)$

$P(s \mid 11112)$

$P(s \mid 11113)$

...

$P(s \mid 11121)$

...

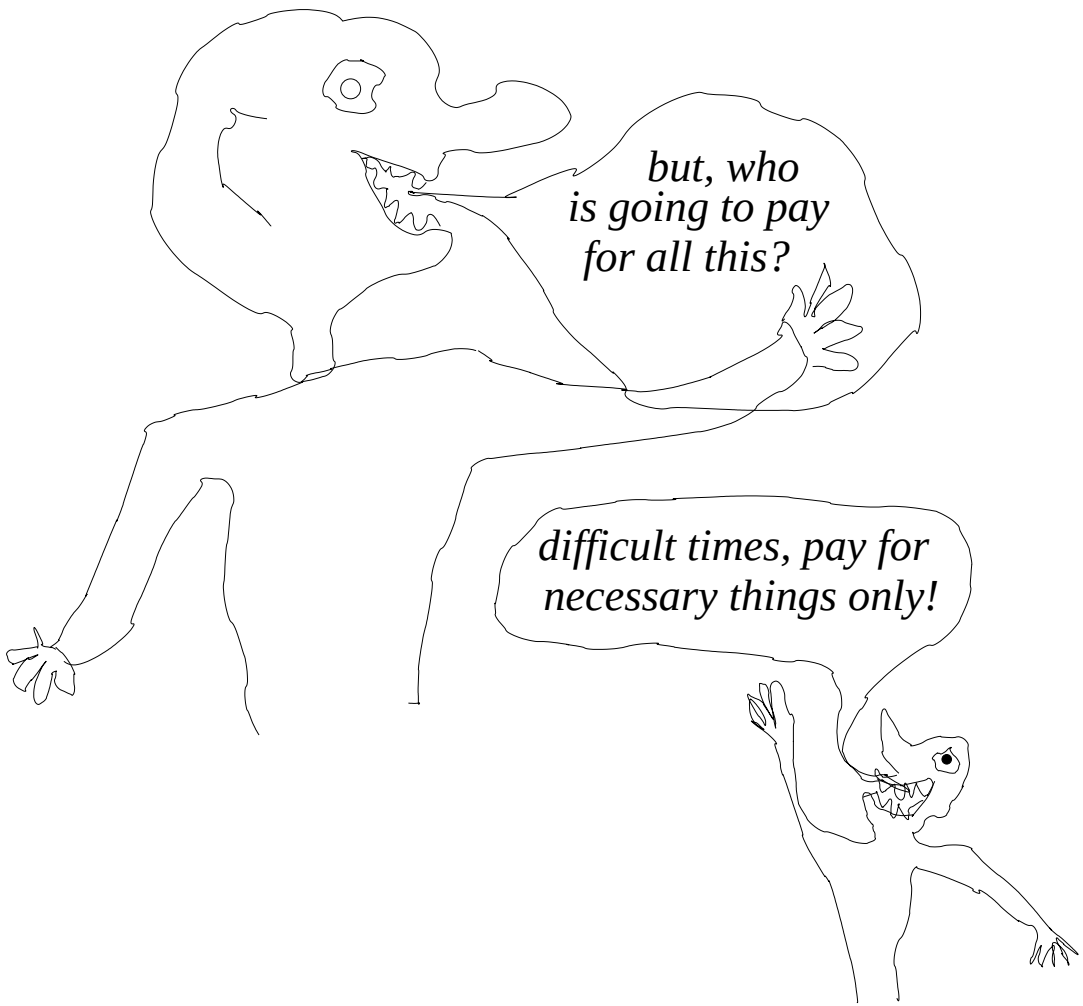
$P(s \mid 43211)$

...

$P(s \mid 44444)$

$s \in \{1, 2, 3, 4\}$

Difficult times...



Markov model

MM are intuitive and simple, but ...

Number of IPS (prediction contexts) grows exponentially fast with memory length

Large alphabets and long memories are infeasible:

- computationally demanding and
- very long sequences are needed to obtain sound statistical estimates of free parameters

Variable memory length MM (VLMM)

Sophisticated implementation of potentially high-order MM

Takes advantage of subsequence structure in data

Saves resources by making memory depth context dependent

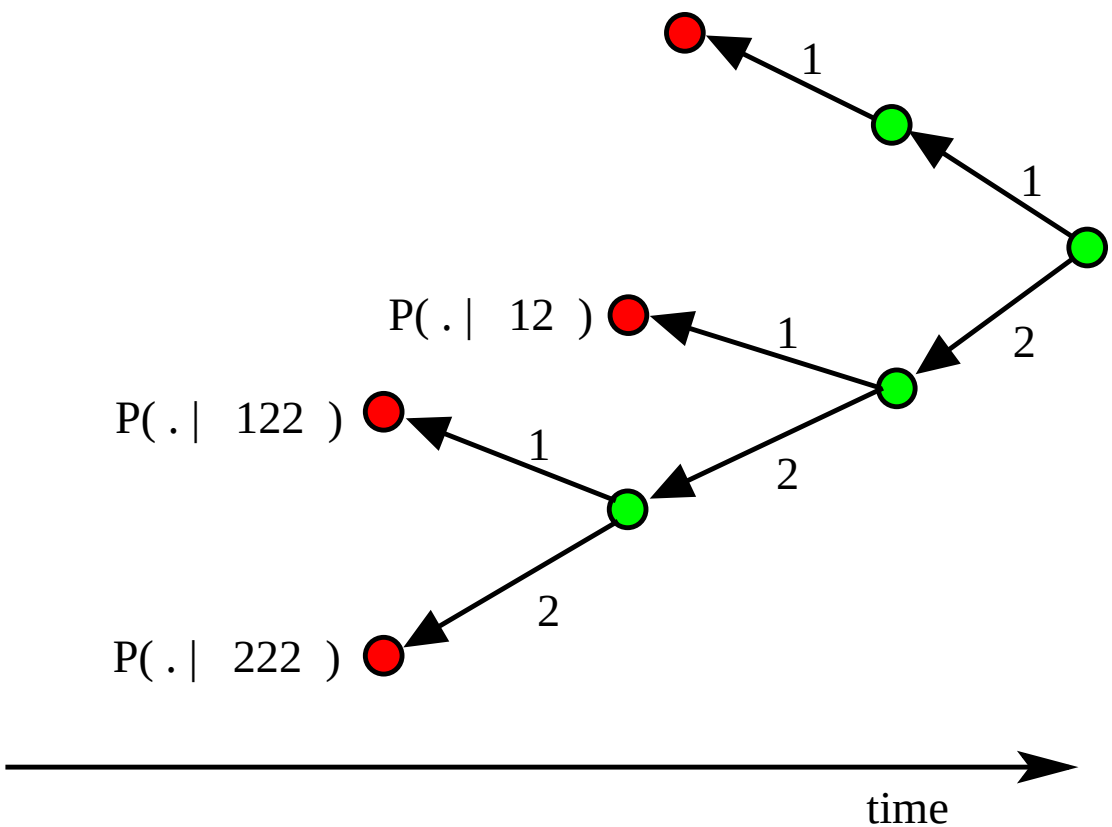
Use deep memory only when it is needed

Natural representation of IPS in form of Prediction Suffix Trees

Closely linked to the idea of universal simulation of information sources.

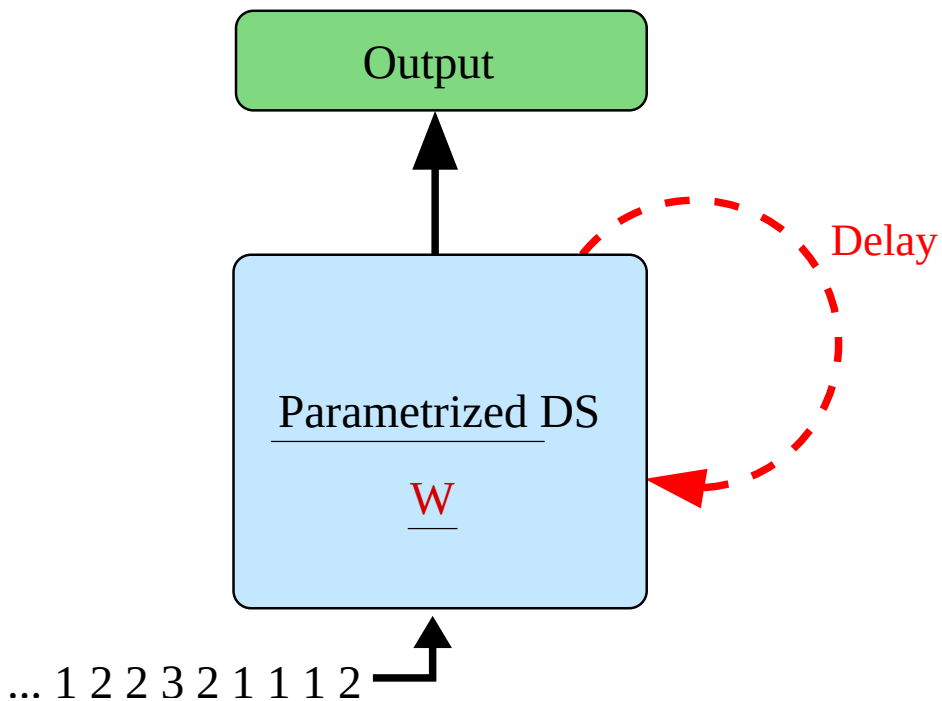
Prediction suffix tree (PST)

IFS are organized on nodes of PST
Traverse the tree from root towards leaves in reversed order
Complex and potentially time-consuming training



Each node i
has associated next−symbol
probability distribution
P(. | i)

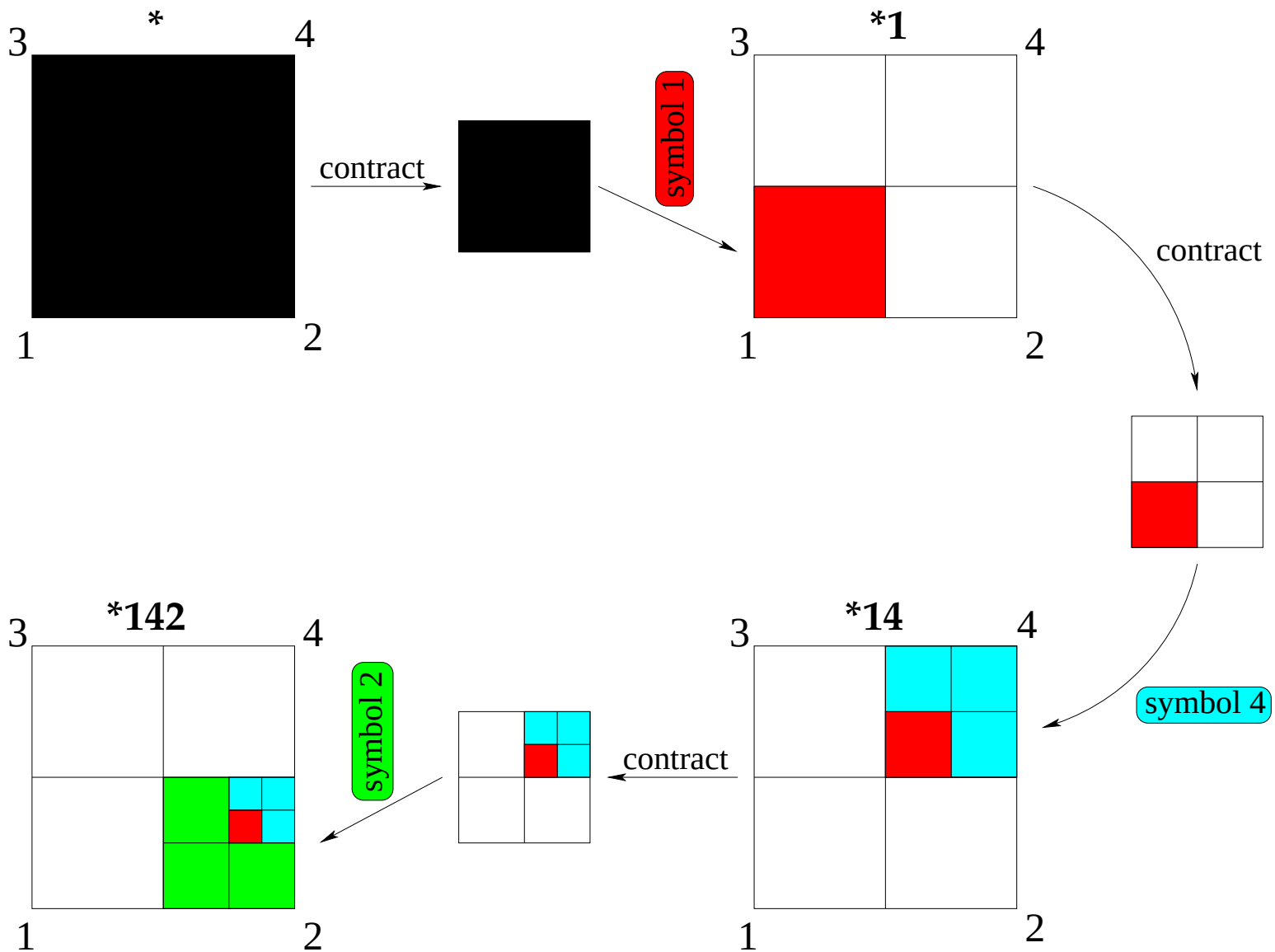
Continuous state space, discrete inputs/outputs



States of the DS form IPS: $\mathbf{x}(t) = f(\mathbf{u}(t), \mathbf{x}(t-1);\ \mathbf{w})$

These states code the entire history of symbols we have seen so far

The power of contractive DS



Affine input-driven dynamics

$$\begin{aligned} \mathbf{x}(t) &= f(\mathbf{u}(t), \mathbf{x}(t-1)) \\ &= \rho \cdot \mathbf{x}(t-1) + (1-\rho) \cdot \mathbf{u}(t). \end{aligned}$$

or, more generally

$$\mathbf{x}(t) = \frac{\rho}{L}\mathbf{x}(t-1) + \mathbf{u}(t), \;\; \mathbf{u}(t) \in \left\{0, \frac{1}{L}, \frac{2}{L}, ..., \frac{L-1}{L}\right\}^d,$$

with a scale parameter $\rho \in (0, 1]$.

Mapping sequences with contractions

Mapping symbolic sequences into Euclidean space

Coding strategy is Markovian

Sequences with shared histories of recently observed symbols have close images

The longer is the common suffix of two sequences, the closer are they mapped to each other

Cluster structure of IPS - reduced IPS set

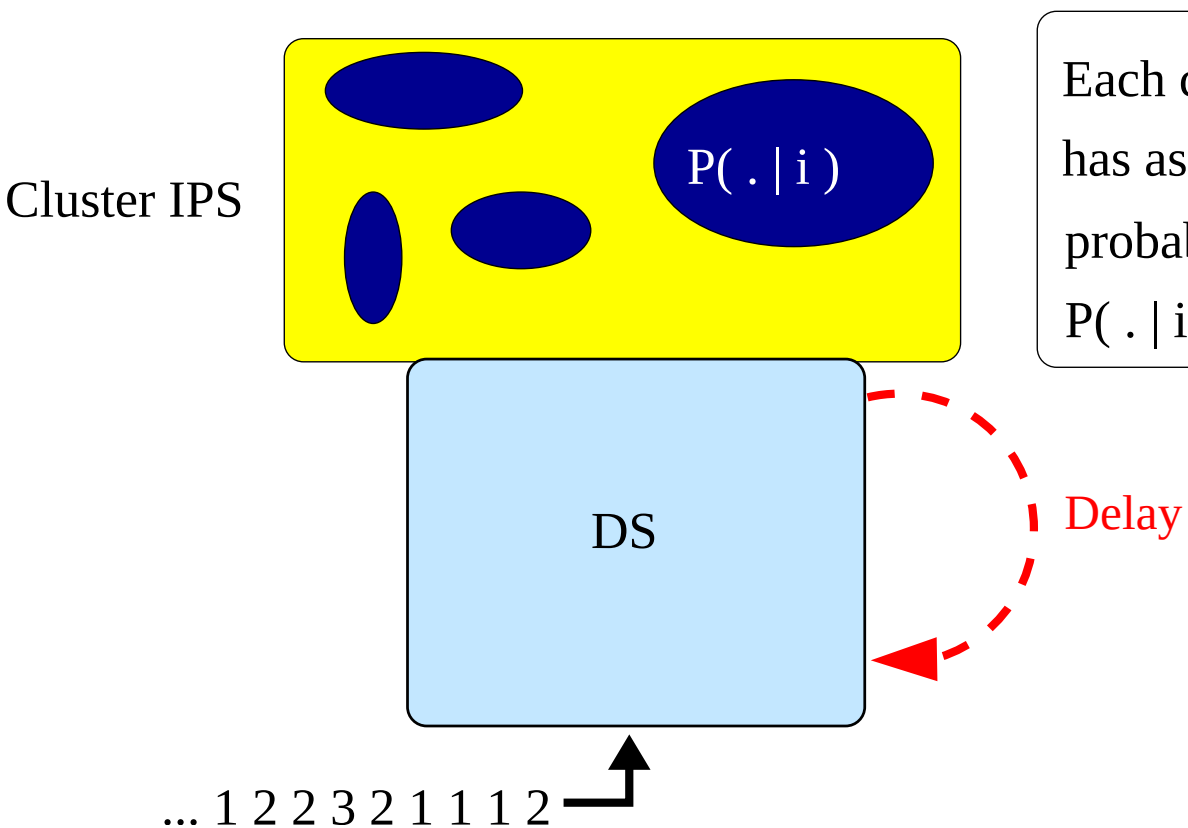
The output readout map $\mathbf{y}(t) = h(\mathbf{x}(t))$ is often smooth and/or highly constrained. This imposes the requirement that IPS leading to the same/similar output should lie "close" to each other.
Hence, it is natural to look at the cluster structure of IPS

Each cluster represents an abstract IPS

If we wish, we can estimate the topology/statistics of state transitions/symbol emissions on such new IPS on the available data and test them on a hold-out set

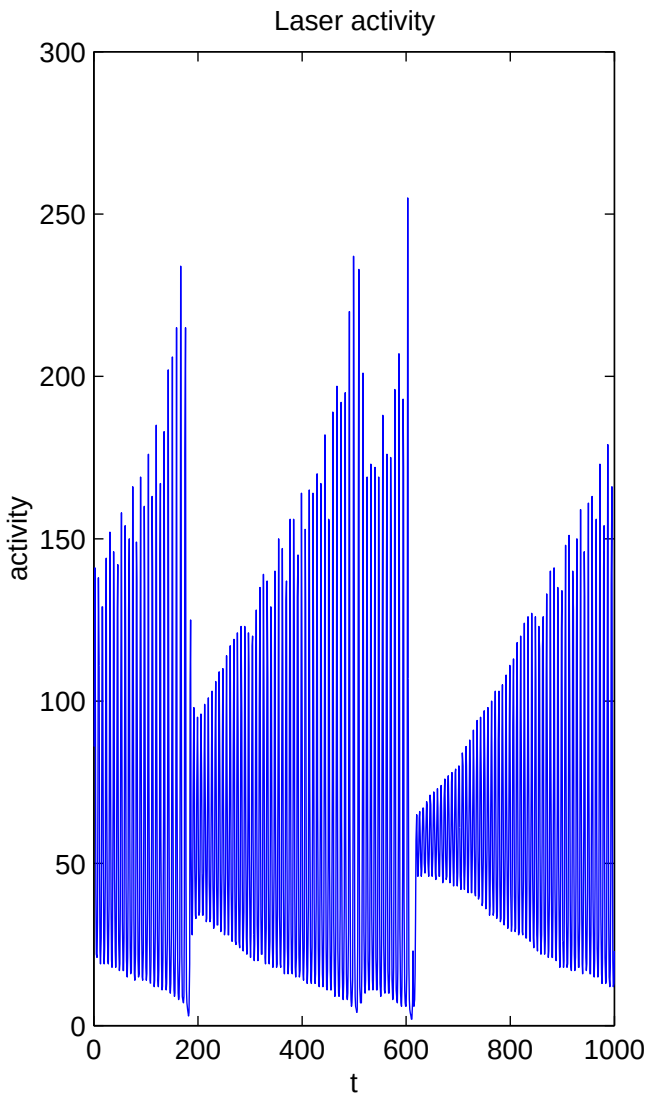
Readout from IPS - Prediction Machines (PM)

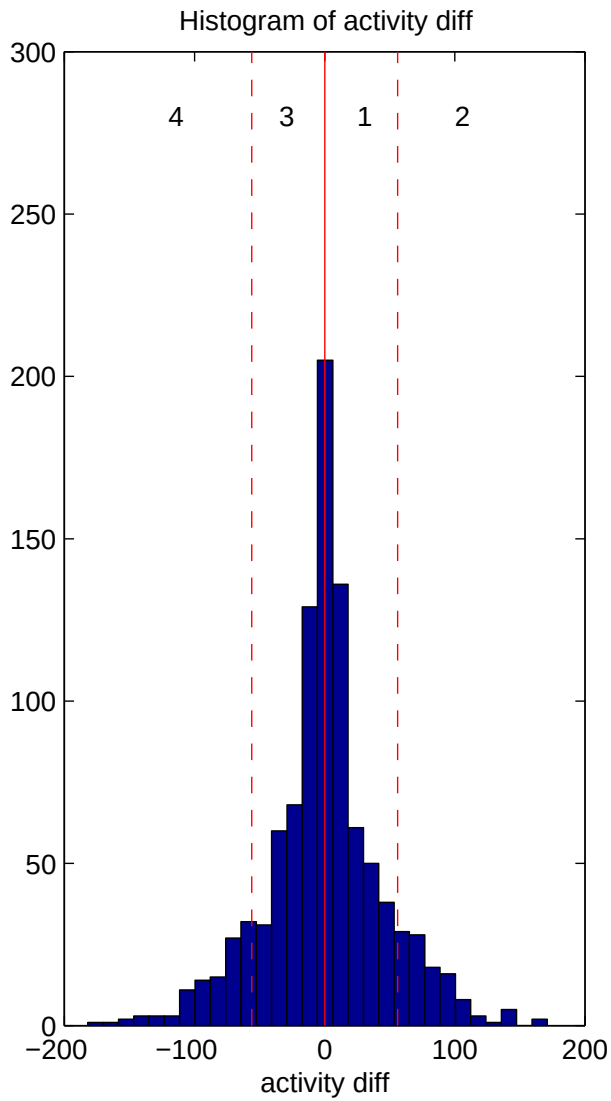
Reduce complexity of IPS by identifying "higher-level" IPS with clusters (indexed by i) of IPS $\mathbf{x}(t)$



Each cluster i=1,2,...,M
has associated next−symbol
probability distribution
P(. | i)

Chaotic laser

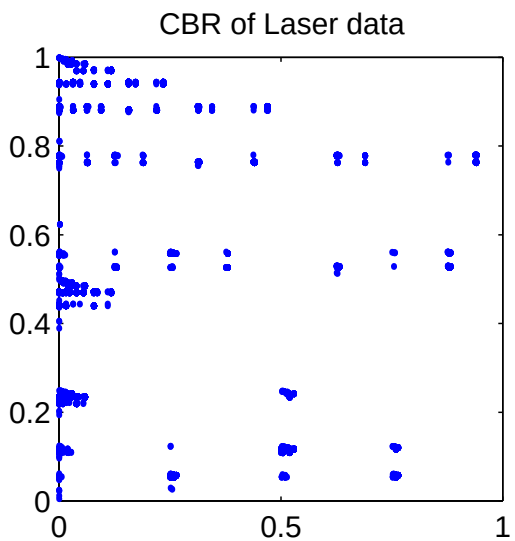


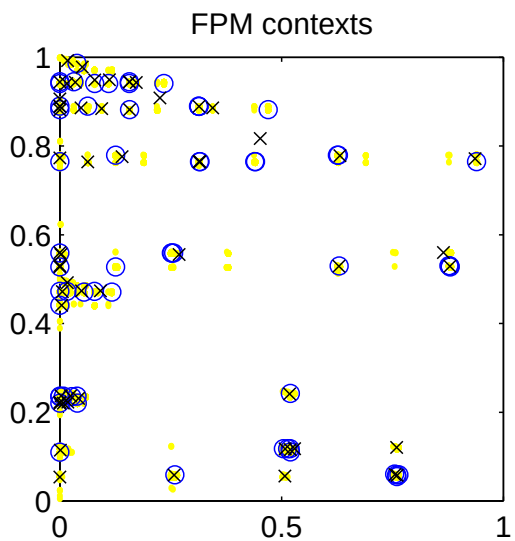


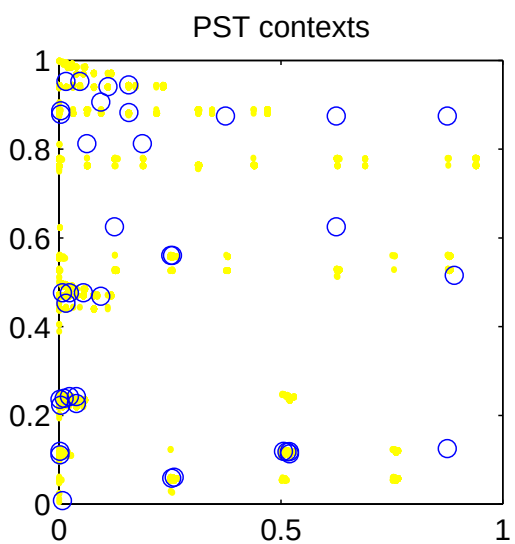
Training sequence: 8000 symbols

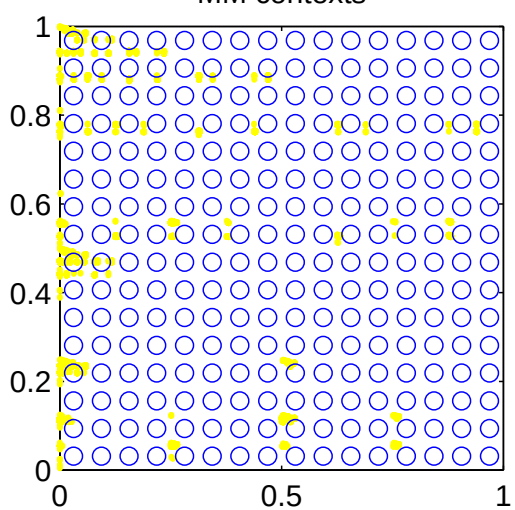
Test sequence: 2000 symbols

Fractal representation through affine dynamics

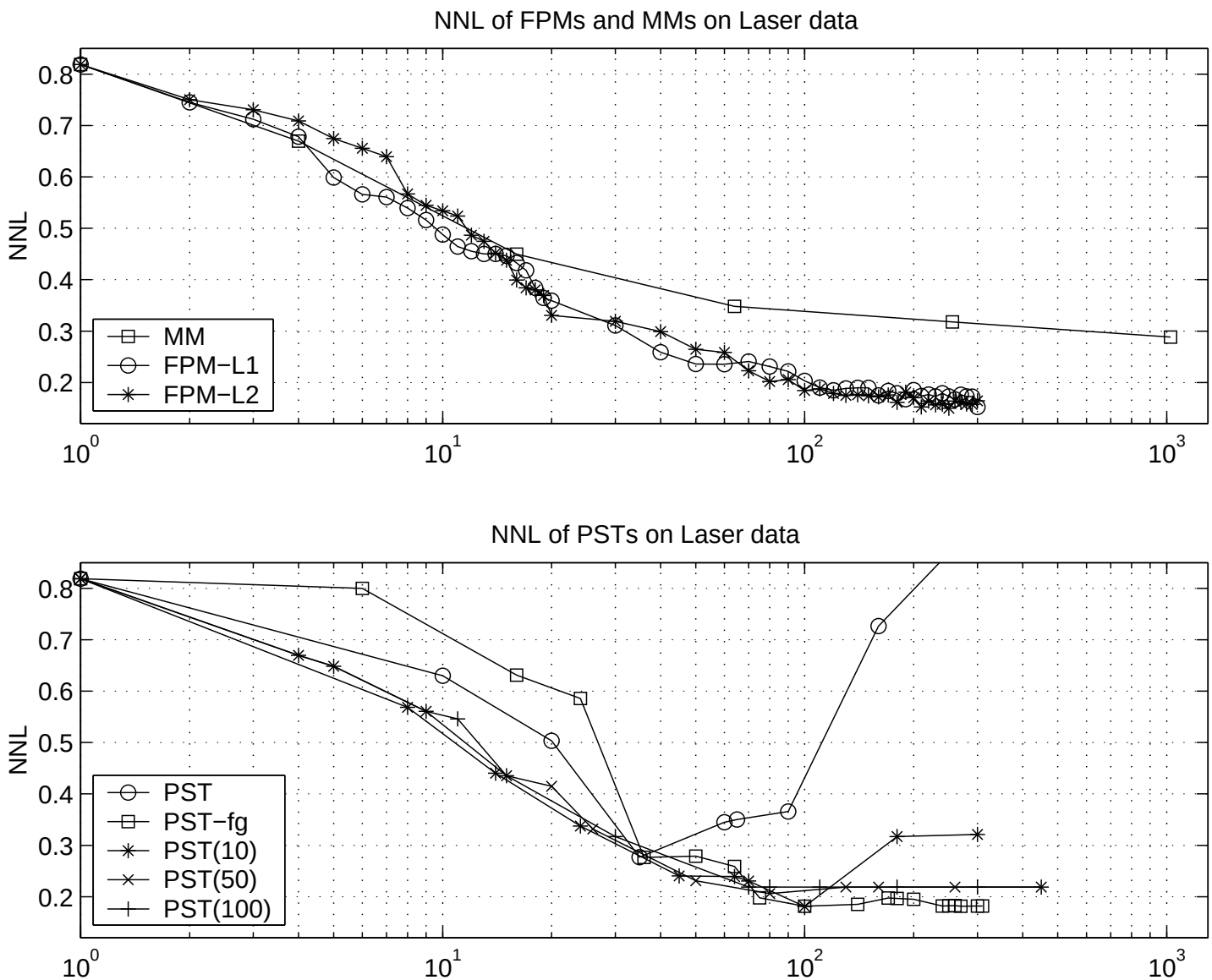




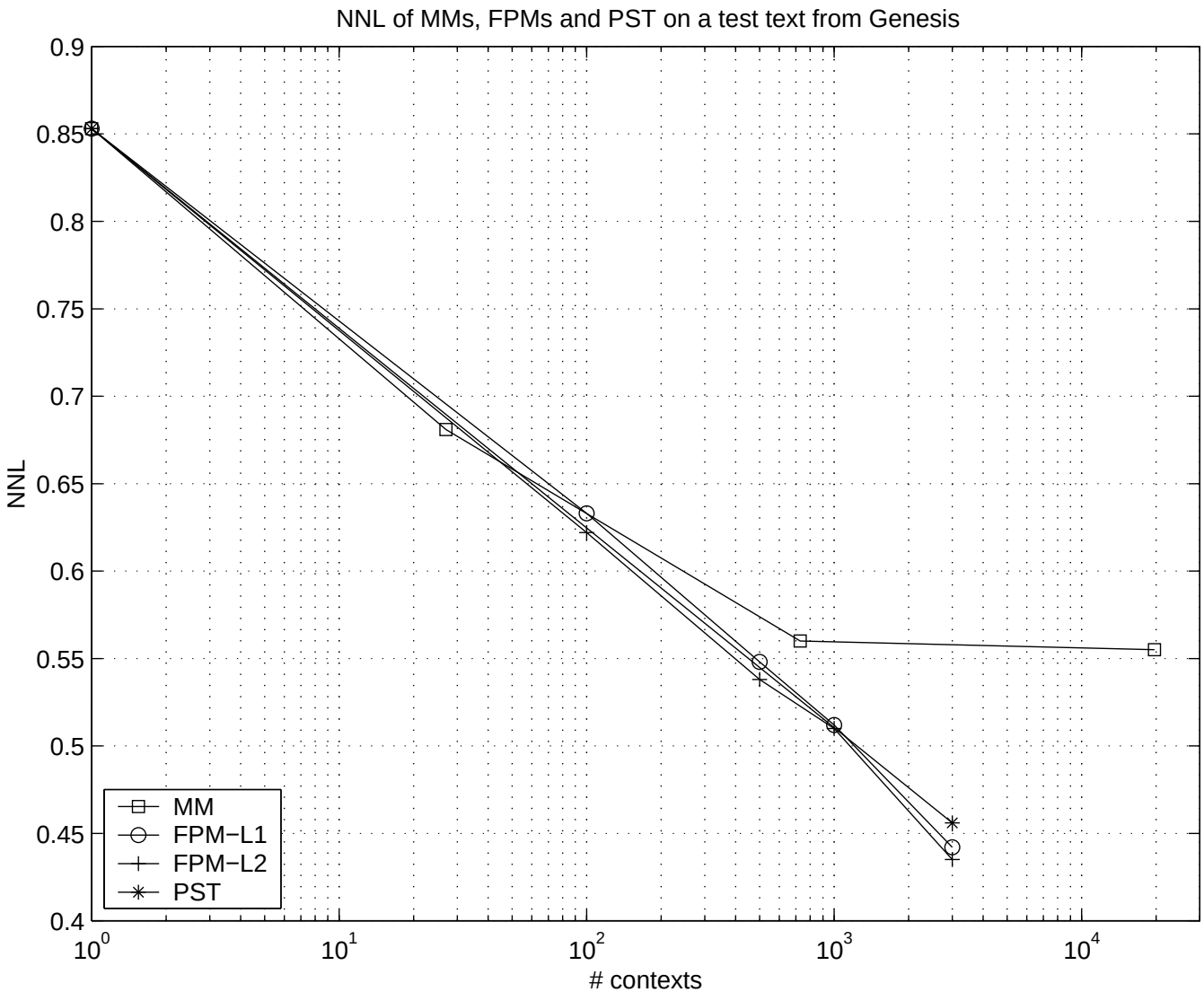




Laser - results (NNL on test set)



Bible

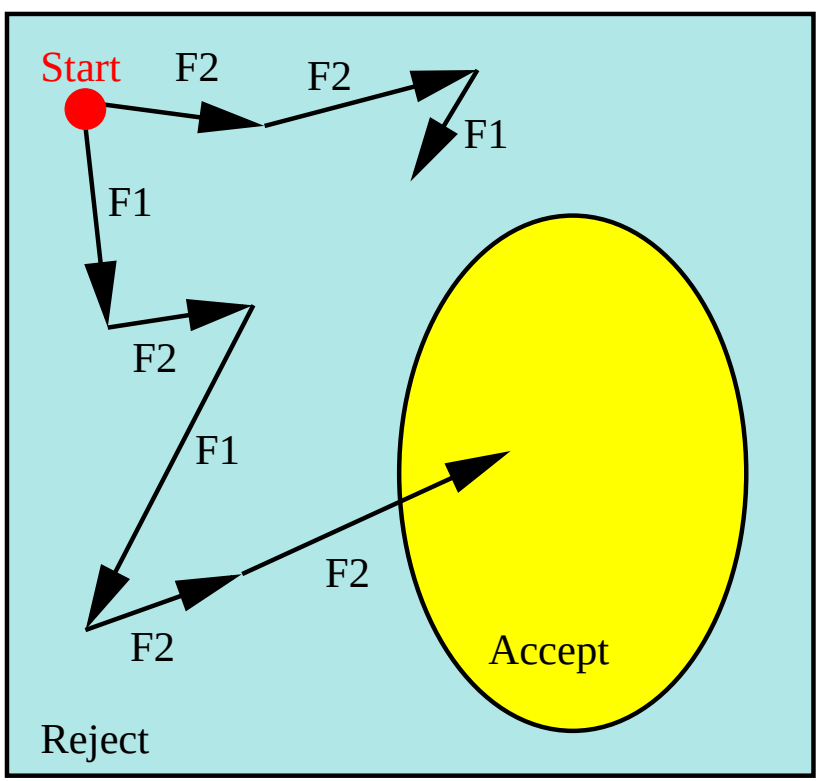


Training sequence: Old Testament - Genesis
Test sequence: Genesis

‘Pure’ state transition formulation

Another possibility is to keep the state transition formulation without additional memory mechanism, but allow infinite number of IFS (countable, or even uncountable)

E.g. Dynamical recognizers studied by Pollack, Moore, ...



State transitions:

On symbol 1: x −> F1(x)

On symbol 2: x −> F2(x)

12122 − grammatical

221 − non−grammatical

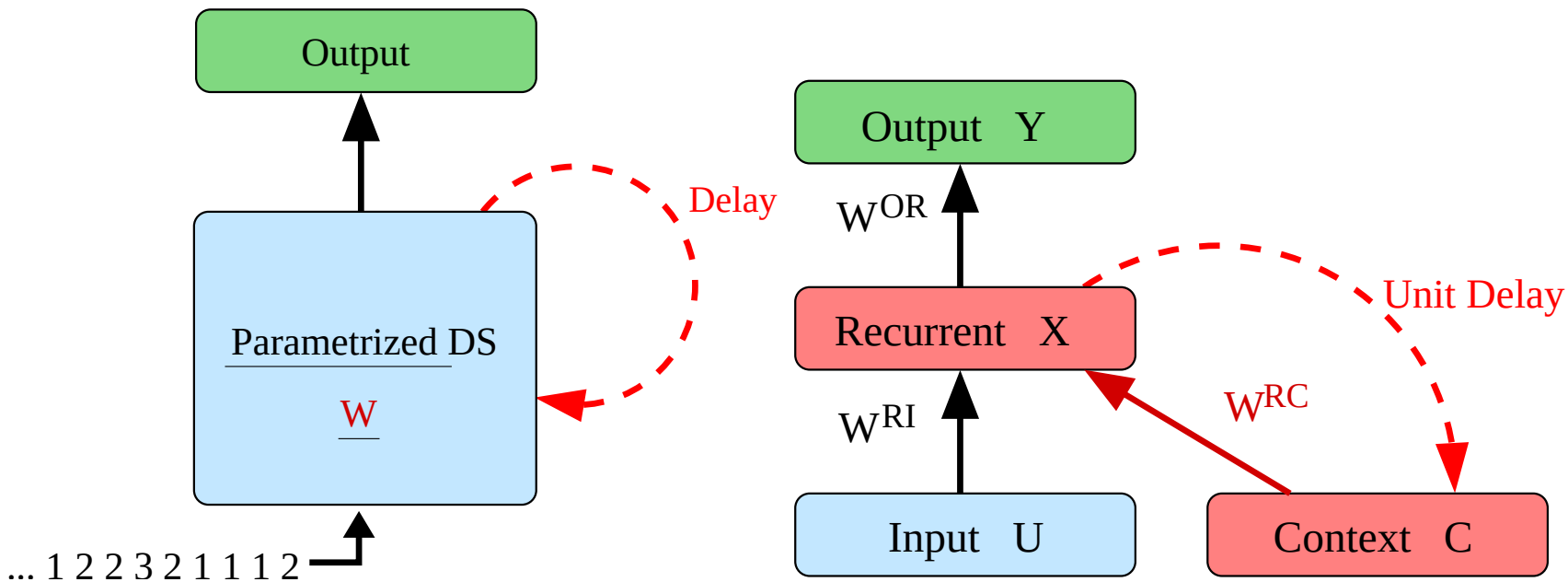
Continuous state space

State transitions over a continuous state space $\mathcal{X}$ can be considered iterations of a non-autonomous dynamical system on $\mathcal{X}$.

Each map $F_i : \mathcal{X} \to \mathcal{X}$ corresponds to distinct input symbol $i \in \mathcal{A}$.

Having a state space with uncountable number of states can lead to Turing (super-Turing) capabilities, but we need infinite precision of computations (Siegelmann, Moore,...).

Recurrent Neural Network (Elman)



Neurons organized in 4 groups: Input (U), Context (C), Recurrent (X) and Output (Y)

X+C can be considered an "extended" input to the network

C is an exact copy of X from the previous time step

RNN - parametrization

$$\mathbf{x}(t) = f(\mathbf{u}(t), \mathbf{c}(t)) = f(\mathbf{u}(t), \mathbf{x}(t-1))$$

$$\mathbf{y}(t) = h(\mathbf{x}(t))$$

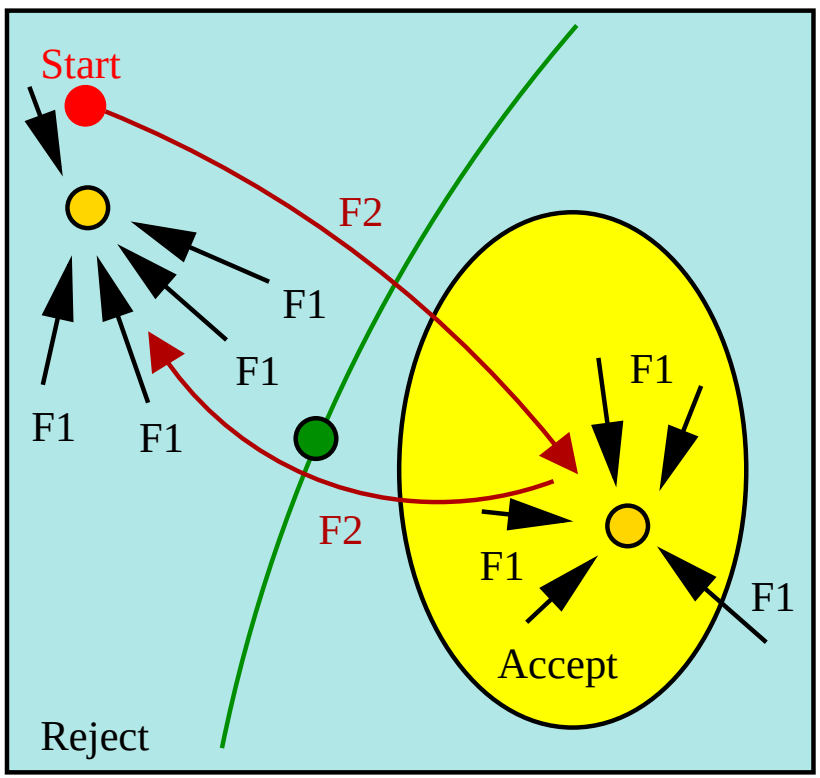
$$\begin{aligned} x_i(t) &= \sigma\left(\sum_j W_{ij}^{RI} \cdot u_j(t) + \sum_j W_{ij}^{RC} \cdot x_j(t-1) + T_i^R\right) \\ y_i(t) &= \sigma\left(\sum_j W_{ij}^{OR} \cdot x_j(t) + T_i^O\right) \end{aligned}$$

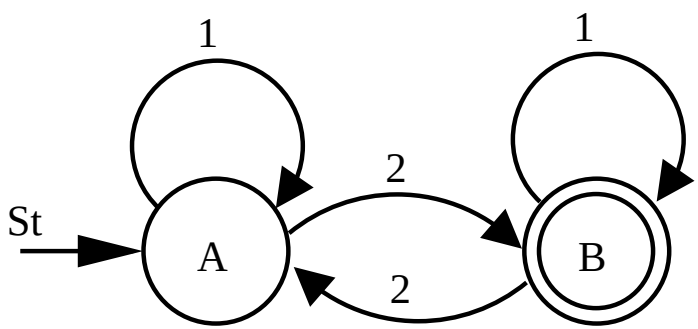
$\mathbf{W}^{RI}$, $\mathbf{W}^{RC}$, $\mathbf{W}^{OR}$ and $\mathbf{T}^R$, $\mathbf{T}^O$ are real-valued connection weight matrices and threshold vectors, respectively

σ is a sigmoid activation function, e.g. $\sigma(u) = (1 + \exp(-u))^{-1}$

Attractive sets

To latch a piece of information for a potentially unbounded number of time steps we need attractive sets





Grammatical:

all strings containg odd number of 2's

Information latching problem

1. To latch an important piece of information for future reference we need to create an attractive set

2. But derivatives of the state-transition map are small in the neighborhood of such an attractive set

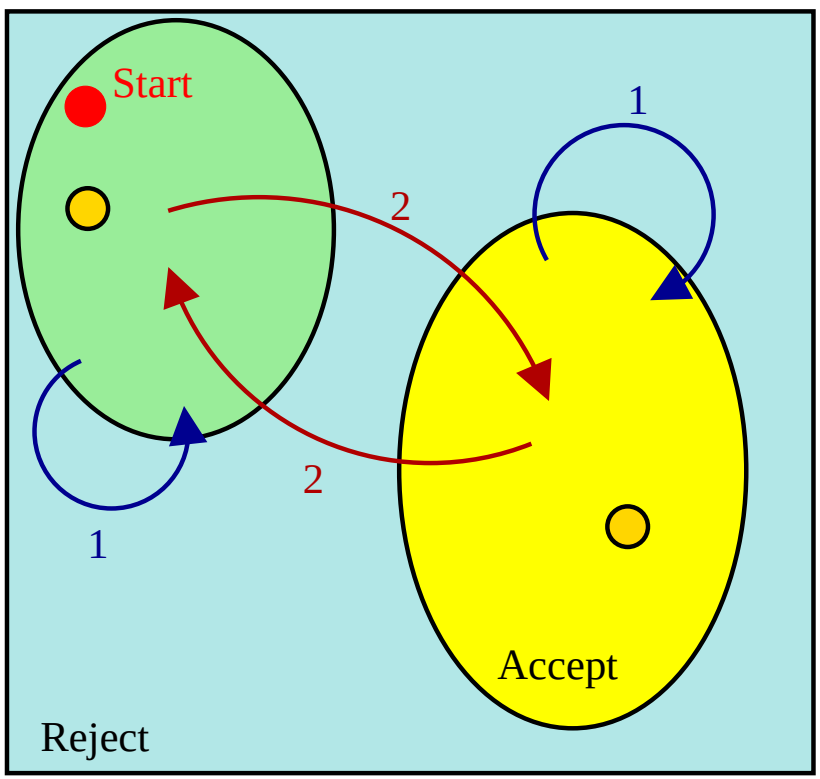
3. We cannot propagate error information when training RNN via a gradient-based procedure – derivatives decrease exponentially fast through time

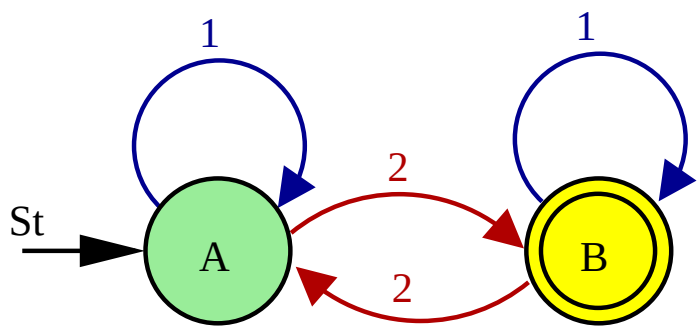
Curse of long time spans

As soon as we try to create a mechanism for keeping important bits of information from the past, we lose the ability to set the parameters to the appropriate values since the error information gets lost very fast

It is actually quite non-trivial (although not impossible) to train a parametrized state space model (e.g. RNN) beyond finite memory on non-toy problems

Informative models form RNN by clustering IPS





Extracted FSM:

all strings containg odd number of 2's

Bit of a shock!

There are good reasons for initializing RNN with small weights
(indeed it is a common practice)

State-transition maps of RNN
(with commonly used sigmoid activation functions)
prior to training (small weights) are contractions

Clustering IPS naturally forms Markov states:
we group together neighboring points that are images of sequences with shared suffixes

RNN without any training is already great!

Markovian architectural bias of RNN

It's all about contractions...

Given an input symbol $s \in \mathcal{A}$ at time t, and activations of recurrent units from the previous step, $\mathbf{x}(t-1)$,

$$\mathbf{x}(t) = f_s(\mathbf{x}(t-1)).$$

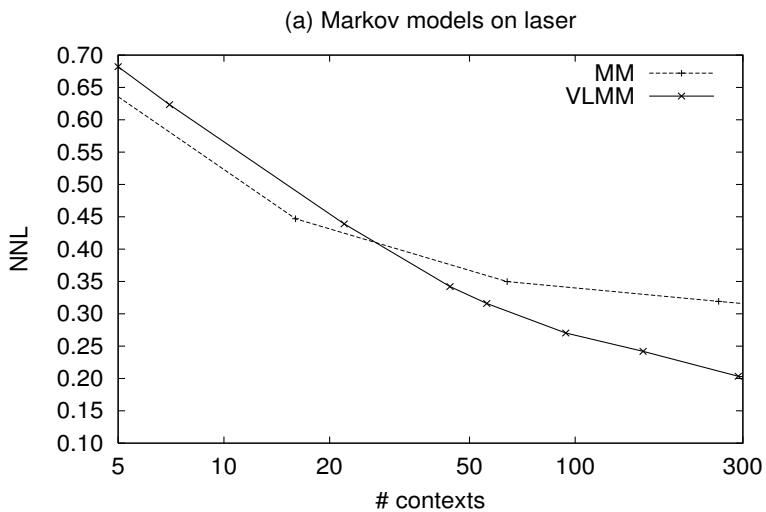
This notation can be extended to sequences over $\mathcal{A}$ the recurrent activations after t time steps are

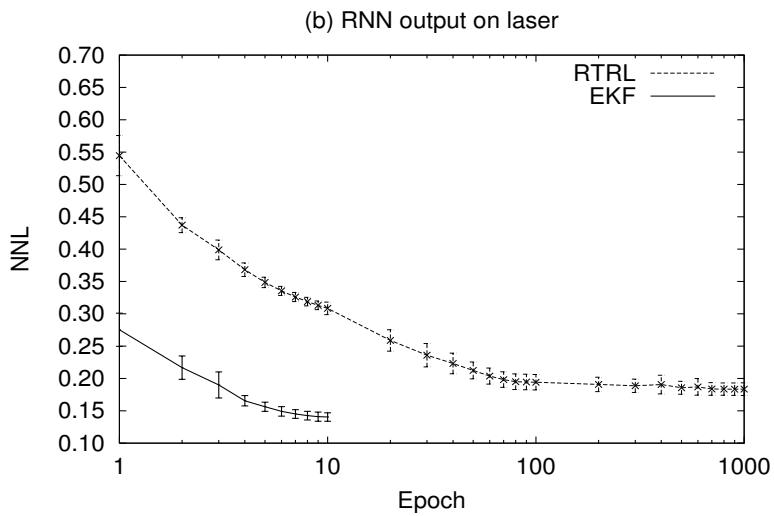
$$\mathbf{x}(t) = f_{s_1 \ldots s_t}(\mathbf{x}(0)).$$

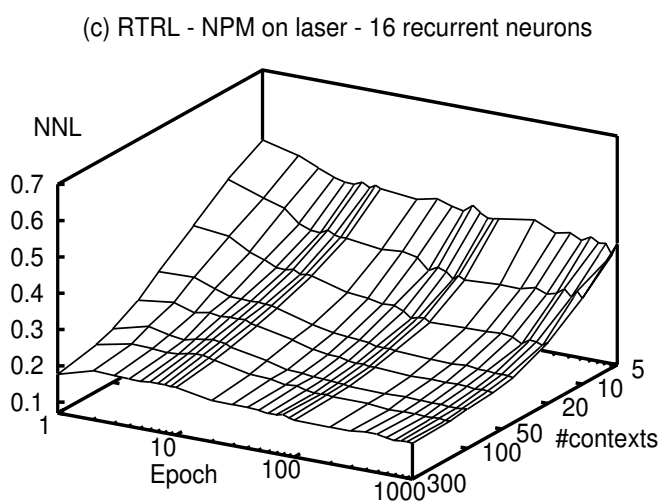
If maps f_s are contractions with Lip. constant $0 < C < 1$,

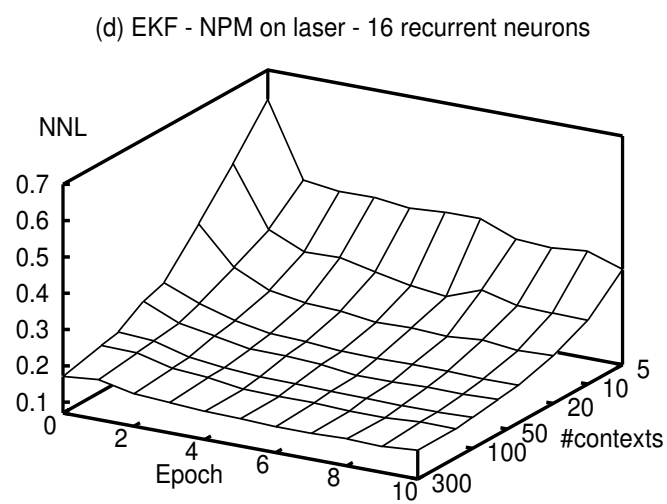
$$\begin{aligned} \|f_{vq}(\mathbf{x}) - f_{wq}(\mathbf{x})\| &\leq C^{|q|} \cdot \|f_v(\mathbf{x}) - f_w(\mathbf{x})\| \\ &\leq C^{|q|} \cdot diam(\mathcal{X}). \end{aligned}$$

Chaotic laser revisited









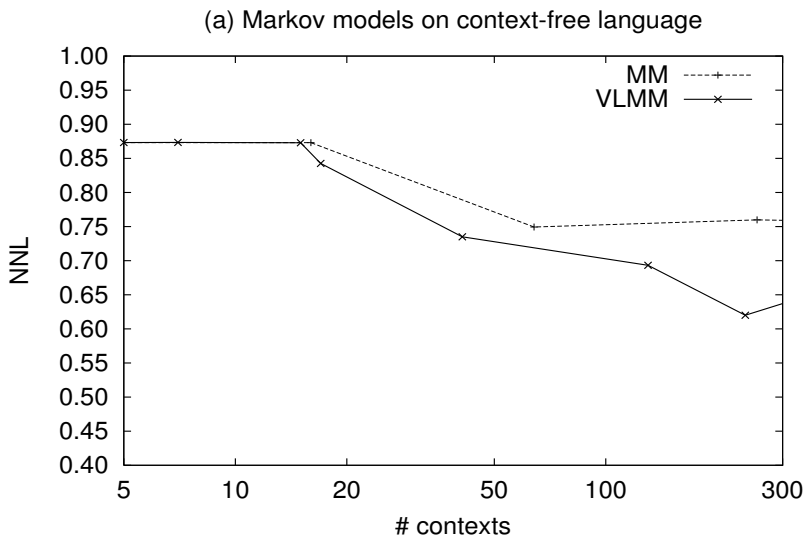
Deep Recursion

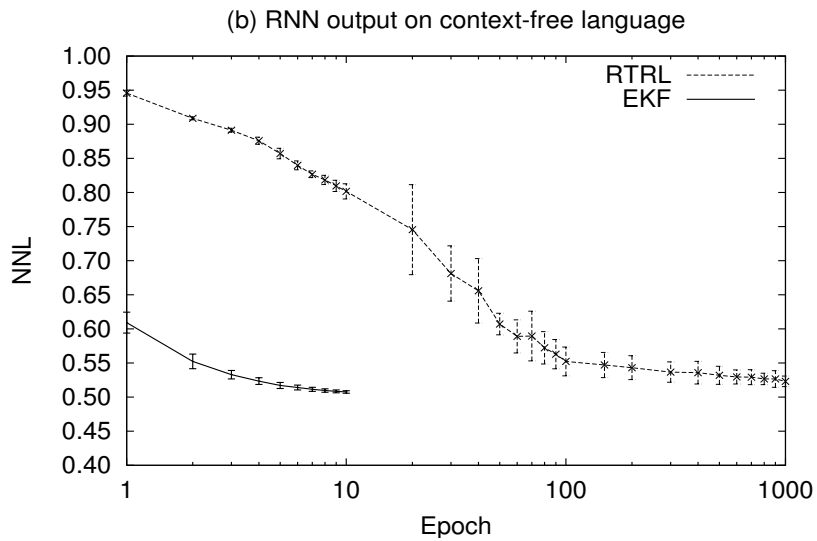
Production rules:
$R \to 1R3$, $R \to 2R4$, $R \to e$.

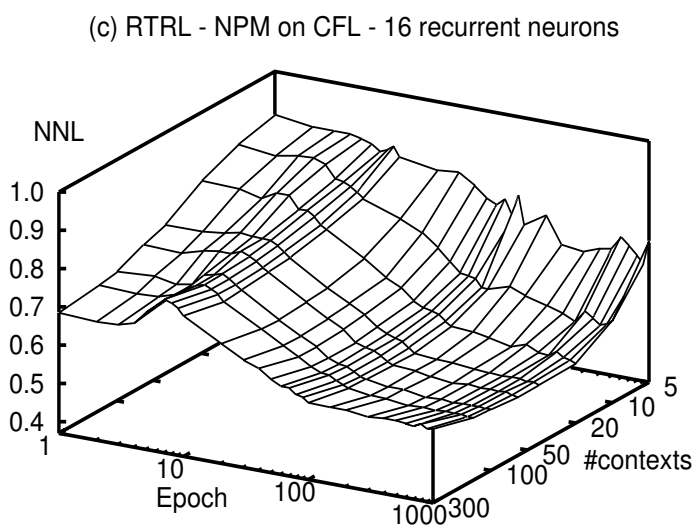
Length	Percent in the sequence	Examples
2	34.5 %	$13, 24$
4	20.6%	$1243, 1133$
6	13.6 %	$122443, 211344$
8	10.3%	22213444
10,12,14,...20	each length 3.5 %	12121211333434343

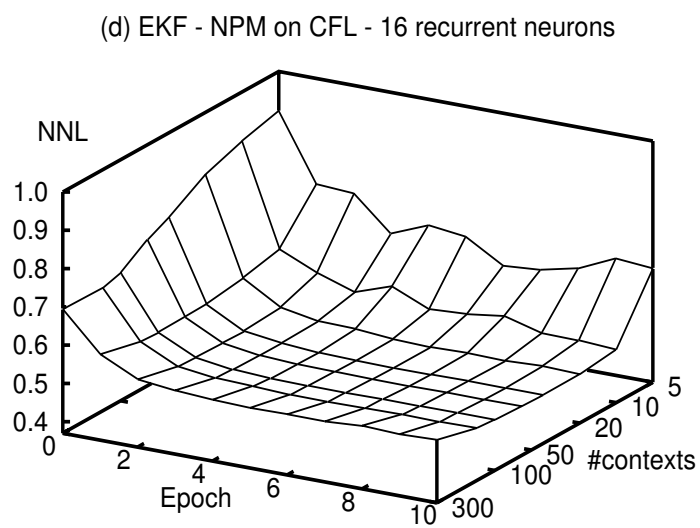
Table 1: Distribution of string lengths. Training sequence – 6156 symbols, test sequence – 6192 symbols

Deep recursion - results









Theoretical grounding

Theorem (Hammer & Tiňo):

- Recurrent networks with contractive transition function can be approximated arbitrarily well on input sequences of unbounded length by a definite memory machine. Conversely,

- every definite memory machine can be simulated by a recurrent network with contractive transition function.

Hence initialization with small weights induces an architectural bias into learning with recurrent neural networks.

Learnability (PAC)

Valid also for continuous input/output spaces.

The architectural bias emphasizes one possible region of the weight space where generalization ability can be formally proved.

Standard RNN are not distribution independent learnable in the PAC sense if arbitrary precision and inputs are considered.

Theorem (Hammer & Tiňo):

- Recurrent networks with contractive transition function with a fixed contraction parameter fulfill the distribution independent UCED property and so
- unlike general recurrent networks, are distribution independent PAC-learnable.

Fractal analysis

Complexity of the driving input stream (topological entropy) is directly reflected in the complexity of the state space activations (fractal dimension).

Theorem (Tiňo & Hammer):

Recurrent activations inside contractive RNN form fractal clusters the dimension of which can be bounded by the scaled entropy of the underlying driving source. The scaling factors are fixed and are given by the RNN parameters.

So do we need adaptive hidden state structure?

Hidden states add flexibility, but make training more difficult.

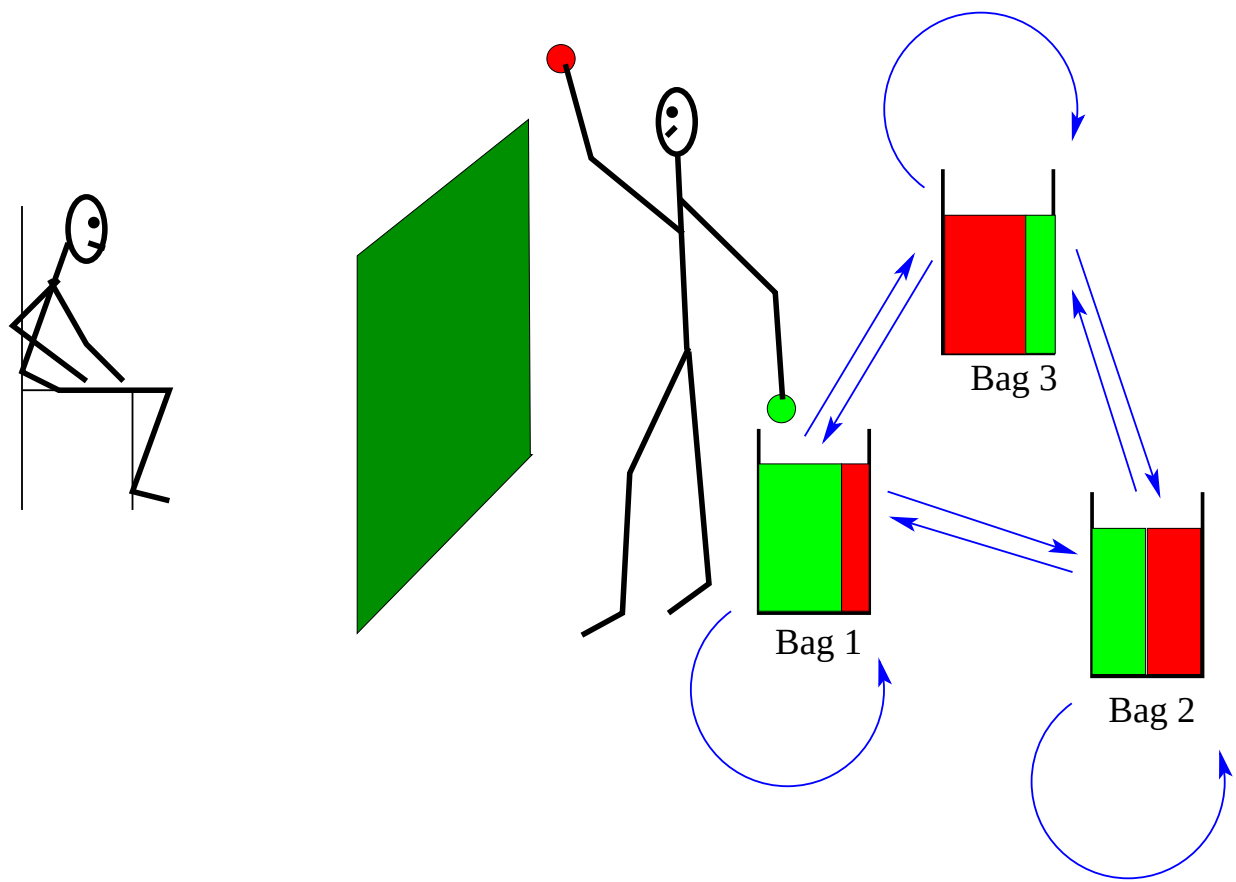
Can represent and track underlying changes in the generative process.

If the number of states is finite
- hidden Markov models
- switching models
- ...

Finite state set - Hidden Markov Model

Stationary emissions conditional on hidden (unobservable) states.

Hidden states represent basic operating "regimes" of the process.



Temporal structure - Hidden Markov Model

We have M bags of balls of different colors (Red - R, Green - G).

We are standing behind a curtain and at each point in time we select a bag j, draw (with replacement) a ball from it and show the ball to an observer. Color of the ball shown at time t is $C(t) \in \{R, G\}$. We do this for T time steps.

The observer can only see the balls, it has no access to the information about how we select the bags.

Assume: we select bag at time t based only on our selection at the previous time step $t - 1$ (1st-order Markov assumption).

HMM - probabilistic model

Probabilistic, finite state and observation spaces version of

$$\mathbf{x}(t) = f(\mathbf{x}(t-1)), \quad \mathbf{y}(t) = h(\mathbf{x}(t)), \text{ hence,}$$

$$p(\mathbf{x}(t) \mid \mathbf{x}(t-1)), \quad p(\mathbf{y}(t) \mid \mathbf{x}(t))$$

- alphabet of A symbols, $\mathcal{A} = \{1, 2, ..., A\}$.
- Consider a set of symbolic sequences, $\mathbf{s}^{(n)} = (s_t^{(n)})_{t=1:T_n}$, $n = 1 : N$
- Assuming independently generated sequences, the likelihood is

$$\mathcal{L} = \prod_{n=1}^{N} p(\mathbf{s}^{(n)}),$$

$$p(\mathbf{s}^{(n)}) = \sum_{\mathbf{x} \in K^{T_n}} p(x_1) \prod_{t=2}^{T_n} p(x_t|x_{t-1}) \prod_{t=1}^{T_n} p(s_t^{(n)}|x_t)$$

Model estimation - if only we knew ...

If we knew which bags were used at which time steps, things would be very easy! ... just counting

Hidden variables z_t^j:

$z_t^j = 1$, iff bag j was used at time t;

$z_t^j = 0$, otherwise.

$$P(bag_j \to bag_k) = \frac{\sum_{t=1}^{T-1} z_t^j \cdot z_{t+1}^k}{\sum_{q=1}^{M} \sum_{t=1}^{T-1} z_t^j \cdot z_{t+1}^q} \qquad \text{[state transitions]}$$

$$P(color = c \mid bag_j) = \frac{\sum_{t=1}^{T} z_t^j \cdot \delta(c = C(t))}{\sum_{g \in \{R,G\}} \sum_{t=1}^{T} z_t^j \cdot \delta(g = C(t))} \qquad \text{[emissions]}$$

But we don't ...

We need to come up with reasonable estimates of probabilities for hidden events such as:

- $z_t^j \cdot z_{t+1}^k = 1$
  at time t we selected bag j and at the next time step we selected bag k

- $z_t^j \cdot \delta(c = C(t)) = 1$
  at time t we selected bag j and drew ball of color c

Again, the probability estimates need to be based on observed data $\mathcal{D}$ and our current model of state transition and emission probabilities.

Re-estimate the model

$$P(z_t^j \cdot z_{t+1}^k = 1 \mid \mathcal{D}, \text{ current model}) = R_t^{j \to k}$$

$$P(z_t^j \cdot \delta(c = C(t)) = 1 \mid \mathcal{D}, \text{ current model}) = R_t^{j,c}$$

Then we can re-estimate the model parameters as follows:

$$P(bag_j \to bag_k) = \frac{\sum_{t=1}^{T-1} R_t^{j \to k}}{\sum_{q=1}^{M} \sum_{t=1}^{T-1} R_t^{j \to q}} \qquad \text{[state transitions]}$$

$$P(color = c \mid bag_j) = \frac{\sum_{t=1}^{T} R_t^{j,c}}{\sum_{g \in \{R,G\}} \sum_{t=1}^{T} R_t^{j,g}} \qquad \text{[emissions]}$$

Estimating values of the hidden variables

In this case we haven't dealt with the crucial question of how to compute the estimation for possible values of hidden variables, given the observed data and current model parameters.

This can be done efficiently - Forward-Backward algorithm.

There is a consistent framework for training any form of latent-variable model via maximum likelihood: Estimation–Maximization (E-M) algorithm.

Learning - can we do better than hand-waving?

Observed data: $\mathcal{D}$

Parameterized model of data items **t**: $p(\mathbf{t}|\mathbf{w})$

Log-likelihood of **w**: $\log p(\mathcal{D}|\mathbf{w})$

Train via Maximum Likelihood:

$$\mathbf{w}_{ML} = \underset{\mathbf{w}}{\operatorname{argmax}} \ \log p(\mathcal{D}|\mathbf{w}).$$

Complete data log-likelihood

Observed data: $\mathcal{D}$

Unobserved data: $\mathcal{Z}$ (realization of compound hidden variable Z)

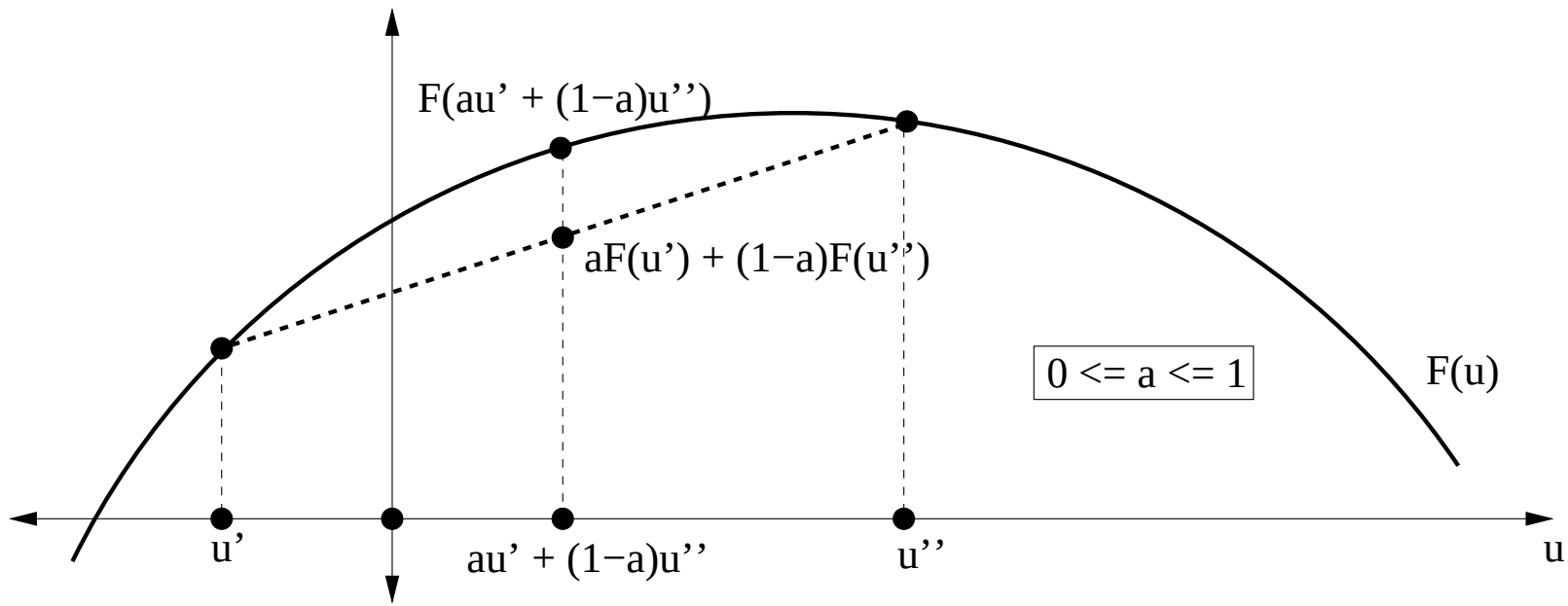
Log-likelihood of $\mathbf{w}$: $\log p(\mathcal{D}|\mathbf{w})$

Complete data log-likelihood: $\log p(\mathcal{D}, \mathcal{Z}|\mathbf{w})$

By marginalization:

$$p(\mathcal{D}|\mathbf{w}) = \sum_{\mathcal{Z}} p(\mathcal{D}, \mathcal{Z}|\mathbf{w})$$

Concave function



For any concave function $F : \mathcal{R} \to \mathcal{R}$ and any $u', u'' \in \mathcal{R}$, $a \in [0, 1]$:

$$F(au' + (1 - a)u'') \geq aF(u') + (1 - a)F(u'').$$

$$F\left(\sum_i a_i u_i\right) \geq \sum_{i=1} a_i F(u_i), \quad a_i \geq 0, \quad \sum_i a_i = 1$$

A lower bound on log-likelihood

Pick 'any' distribution Q for hidden variable Z.
As usual, $Q(Z = \mathcal{Z})$ will be denoted simply by $Q(\mathcal{Z})$.

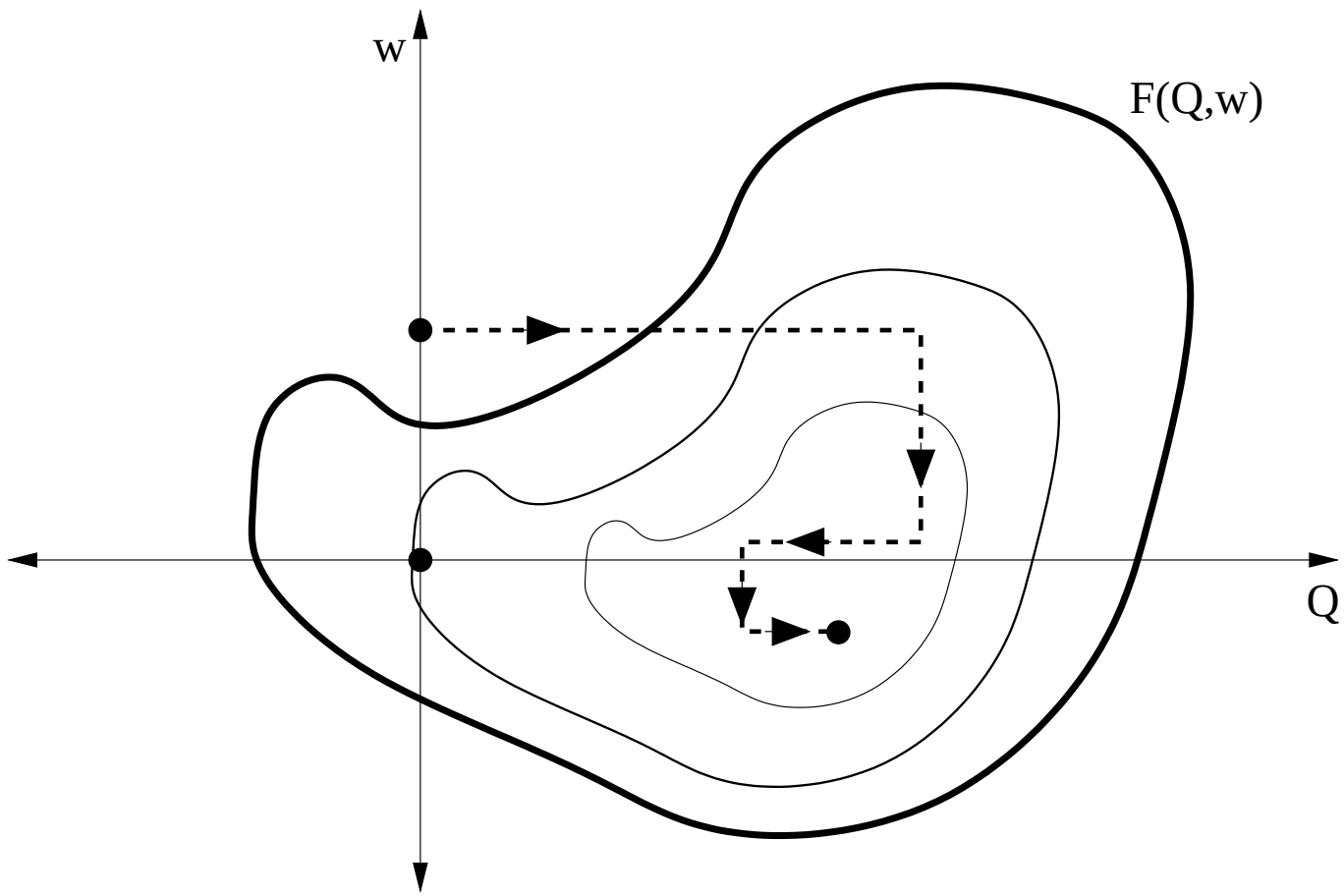
$\log(\cdot)$ is a concave function and $\sum_{\mathcal{Z}} Q(\mathcal{Z}) = 1$, $Q(\mathcal{Z}) > 0$.

Log-likelihood of $\mathbf{w}$: $\log p(\mathcal{D}|\mathbf{w})$

$$\begin{aligned}
\log p(\mathcal{D}|\mathbf{w}) &= \log \left(\sum_{\mathcal{Z}} p(\mathcal{D}, \mathcal{Z}|\mathbf{w}) \right) \\
&= \log \left(\sum_{\mathcal{Z}} Q(\mathcal{Z}) \, \frac{p(\mathcal{D}, \mathcal{Z}|\mathbf{w})}{Q(\mathcal{Z})} \right) \\
&\geq \sum_{\mathcal{Z}} Q(\mathcal{Z}) \log \frac{p(\mathcal{D}, \mathcal{Z}|\mathbf{w})}{Q(\mathcal{Z})} = \mathcal{F}(Q, \mathbf{w})
\end{aligned}$$

Maximize the lower bound on log-likelihood

Do ‘coordinate-wise’ ascent on $\mathcal{F}(Q, \mathbf{w})$.



Maximize $\mathcal{F}(Q, \mathbf{w})$ w.r.t. Q (w fixed)

$$\begin{aligned}\mathcal{F}(Q, \mathbf{w}) &= -\sum_{\mathcal{Z}} Q(\mathcal{Z}) \log \frac{Q(\mathcal{Z})}{p(\mathcal{Z}|\mathcal{D}, \mathbf{w})} + \sum_{\mathcal{Z}} Q(\mathcal{Z}) \log p(\mathcal{D}|\mathbf{w}) \\ &= -D_{KL}[Q(Z)||P(Z|\mathcal{D}, \mathbf{w})] + \log p(\mathcal{D}|\mathbf{w}).\end{aligned}$$

Since $\log p(\mathcal{D}|\mathbf{w})$ is constant w.r.t. Q, we only need to maximize $-D_{KL}[Q(Z)||P(Z|\mathcal{D}, \mathbf{w})]$, which is equivalent to minimizing

$$D_{KL}[Q(Z)||P(Z|\mathcal{D}, \mathbf{w})] \geq 0.$$

This is achieved when $D_{KL}[Q(Z)||P(Z|\mathcal{D}, \mathbf{w})] = 0$, i.e. when

$$Q_*(Z) = P(Z|\mathcal{D}, \mathbf{w}) \quad \text{and} \quad \mathcal{F}(Q, \mathbf{w}) = \log p(\mathcal{D}|\mathbf{w}).$$

Exact click, no longer a lose lower bound!

E-Step

We have:

$$Q_*(Z) = P(Z|\mathcal{D}, \mathbf{w}).$$

The optimal way for guessing the values of hidden variables Z is to set the distribution of Z to the posterior over $\mathcal{Z}$, given the observed data $\mathcal{D}$ and current parameter settings $\mathbf{w}$.

Estimation of the posterior $P(Z|\mathcal{D}, \mathbf{w})$ is done in the E-Step of the E-M algorithm.

Maximize $\mathcal{F}(Q, \mathbf{w})$ w.r.t. w (Q is fixed to Q_*)

$$\begin{aligned}\mathcal{F}(Q_*, \mathbf{w}) &= -\sum_{\mathcal{Z}} Q_*(\mathcal{Z}) \log p(\mathcal{D}, \mathcal{Z}|\mathbf{w}) - \sum_{\mathcal{Z}} Q_*(\mathcal{Z}) \log Q_*(\mathcal{Z}) \\ &= E_{Q_*(Z)}[\log p(\mathcal{D}, \mathcal{Z}|\mathbf{w})] + H(Q_*).\end{aligned}$$

Since the entropy of Q_*, $H(Q_*)$, is constant (Q_* is fixed), we only need to maximize $E_{Q_*(Z)}[\log p(\mathcal{D}, Z|\mathbf{w})]$:

$$\mathbf{w}_* = \underset{\mathbf{w}}{\operatorname{argmax}}\, E_{Q_*(Z)}[\log p(\mathcal{D}, Z|\mathbf{w})].$$

M-Step

We have:

$$\mathbf{w}_* = \underset{\mathbf{w}}{\operatorname{argmax}}\, E_{Q_*(Z)}[\log p(\mathcal{D}, Z|\mathbf{w})].$$

The optimal way for estimating the parameters $\mathbf{w}$ is to select the parameter values $\mathbf{w}_*$ that maximize the expected value of the complete data log-likelihood $p(\mathcal{D}, \mathcal{Z}|\mathbf{w})$, where the expectation is taken w.r.t. the posterior distribution over the hidden data $\mathcal{Z}$, $P(\mathcal{Z}|\mathcal{D}, \mathbf{w})$.

Find a single parameter vector $\mathbf{w}_*$ for all hidden variable settings $\mathcal{Z}$ (since we don't know the true values of Z), but while doing this, weight the importance of each particular setting $\mathcal{Z}$ by the posterior probability $P(\mathcal{Z}|\mathcal{D}, \mathbf{w})$.

E-M Algorithm

Given the current parameter setting $\mathbf{w}^{old}$ do:

- E-Step:
  Estimate $P(Z|\mathcal{D}, \mathbf{w}^{old})$, the posterior distribution over $\mathcal{Z}$, given the observed data $\mathcal{D}$ and current parameter settings $\mathbf{w}^{old}$.

- M-Step:
  Obtain new parameter values $\mathbf{w}^{new}$ by maximizing

$$E_{P(Z|\mathcal{D},\mathbf{w}^{old})}[\log p(\mathcal{D}, Z|\mathbf{w})].$$

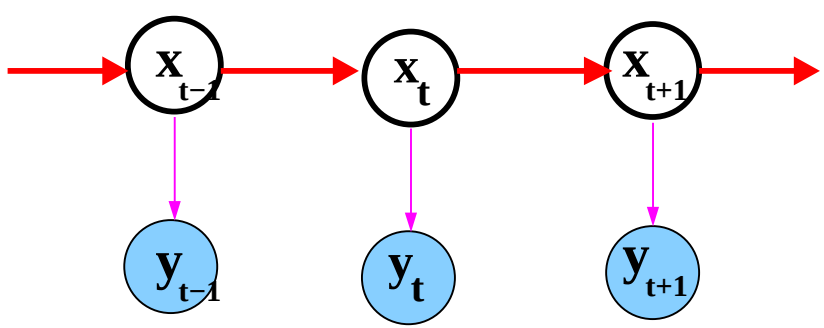
- Set $\mathbf{w}^{old} := \mathbf{w}^{new}$ and go to E-Step.

General Probabilistic State Space Modeling

- Introduction
- Autoregressive models
- Linear state space models
- Dynamical systems
- Non-linear state-space models

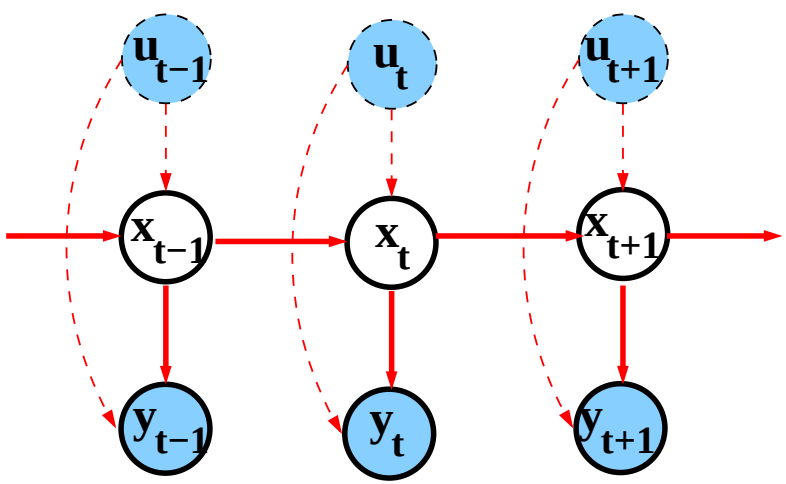
Definition

- A state space model consists of an unobserved state sequence $\mathbf{X}_t \in \mathcal{X}$, with time index $t = 0, 1, 2, ...$, and an obserevation sequence $\mathbf{Y}_t \in \mathcal{Y}$, with $t = 1, 2, ...$;
- State spaces $\mathcal{X}$ and $\mathcal{Y}$ are either open or compact subsets of an Euclidean space;
- The state sequence $\mathbf{X}_0, \mathbf{X}_1, \mathbf{X}_2, ...$ is a Markov chain;
- Conditionally on $\{\mathbf{X}_0, \mathbf{X}_1, ...\}$, the $\mathbf{Y}_t$'s are independent and $\mathbf{Y}_t$ depends on $\mathbf{X}_t$ only;
- Graphical representation:



Definition (cont.)

- Conditional independence. For example, given $\mathbf{X}_t$, $\mathbf{X}_{t+1}$ and $\mathbf{Y}_t$ are independent with each other;
- The observations could include both input variables, $\mathbf{U}_t \in \mathcal{U}$, and output variables, $\mathbf{Y}_t \in \mathcal{Y}$;
- As input variables, $\mathbf{U}_t$ are usually observed without noise corruption;
- Graphical representation:



Probabilistic Modelling

- Initial state density $\mathbf{X}_0 \sim a_0(\mathbf{x})$;
- State transition densities

$$\mathbf{X}_t | (\mathbf{x}_{t-1}, \mathbf{u}_t) \sim a_t(\mathbf{x}, \mathbf{x}_{t-1}, \mathbf{u}_t);$$

- Observation densities

$$\mathbf{Y}_t | (\mathbf{x}_t, \mathbf{u}_t) \sim b_t(\mathbf{y}, \mathbf{x}_t, \mathbf{u}_t).$$

The joint density of $(\mathbf{X}_{0:t}, \mathbf{U}_{1:t}, \mathbf{Y}_{1:t})$ is given by

$$p(\mathbf{x}_{0:t}, \mathbf{u}_{1:t}, \mathbf{y}_{1:t}) = a_0(\mathbf{x}_0) \prod_{s=1}^{t} a_s(\mathbf{x}_s, \mathbf{u}_s, \mathbf{x}_{s-1}) b_s(\mathbf{y}_s, \mathbf{x}_s, \mathbf{u}_s).$$

Probabilistic Modelling (cont.)

- The joint density factorizes into a product of terms containing only pairs of variables $\longrightarrow$
  The joint process $(\mathbf{X}_t, \mathbf{Y}_t)$ is also Markovian;
- The observations $\mathbf{Y}_t$ alone are not Markovian $\longrightarrow$
  The dependence structure of $\mathbf{Y}_t$ could be more complicated than for a Markov process;
- The conditional density $p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})$ is proportional to the joint density $p(\mathbf{x}_{0:t}, \mathbf{y}_{1:t}) \longrightarrow$
  Conditionally on $Y_{1:t} = \mathbf{y}_{1:t}$, the state variables are still Markovian (so-called conditional Markov process);
- Densities $p(\mathbf{x}_t|\mathbf{y}_{1:t})$ store all information about the conditional Markov process up to time t.
- The IPS are densities over the state space $\mathcal{X}$!!!

Statistical Inference

The main tasks we need to solve are

1. Inference about states $\mathbf{x}_s$ based on a stretch of observed values $\mathbf{y}_{q:t}$ for a given model (i.e. a_t and b_t known);
2. Inference about unknown parameters in a_t and b_t.

Inference about $\mathbf{X}_s$ given $\mathbf{y}_{1:t}$ is called

1. prediction if $s > t$,
2. filtering if $s = t$,
3. smoothing if $s < t$.

Autoregressive Models

Autoregressive models for univariate time series $y_1, y_2, \ldots$:

- Autoregressive (AR) models;
- Autoregressive-moving average (ARMA) models
- Autoregressive-moving average with exogenous terms (ARMAX) models

AR Models

- The model: $y_t = \phi_1 y_{t-1} + \phi_2 y_{t-2} + ... + \phi_p y_{t-p} + w_t$ where ϕ_1, ϕ_2, ..., ϕ_p are constants and w_t is a Gaussian noise series with mean zero and variance σ_w^2;
- Define $\mathbf{x}_t = (x_t, x_{t-1}, ..., x_{t-p+1})^\intercal$;
- Define $\mathbf{w}_t = (w_t, 0, ..., 0)^\intercal$ and $\mathbf{H} = (1, 0, ..., 0)$;
- The state equation: $\mathbf{x}_{t+1} = \mathbf{F}\mathbf{x}_t + \mathbf{w}_t$ with

$$\mathbf{F} = \begin{pmatrix} \phi_1 & \phi_2 & \cdots & \phi_{p-1} & \phi_p \\ 1 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 0 & 1 & 0 \end{pmatrix} \in \mathcal{R}^{p \times p}$$

- The observation equation: $y_t = \mathbf{H}\mathbf{x}_t$
- The state evolution is 'contaminated' by noise!

ARMA Models

- Consider the univariate ARMA(2, 1) models:

$$y_t = \phi_1 y_{t-1} + \phi_2 y_{t-2} + \theta w_{t-1} + w_t.$$

  where ϕ_1, ϕ_2, and θ are constant, and w_ts are Gaussian noise with zero mean and variance σ_w^2

- Define $\mathbf{x}_t = (x_t, x_{t-1})^\intercal$;
- State equation: $\mathbf{x}_t = \begin{pmatrix} \phi_1 & 1 \\ \phi_2 & 0 \end{pmatrix} \mathbf{x}_{t-1} + w_t \begin{pmatrix} 1 \\ \theta \end{pmatrix}$;
- Define $\mathbf{H} = (1, 0) \rightarrow$ Observation equation: $y_t = \mathbf{H}\mathbf{x}_t$;
- Verification

$$\begin{aligned} x_t &= \phi_1 x_{t-1} + x_{t-2} + w_t \\ x_{t-1} &= \phi_2 x_{t-1} + \theta w_t \\ x_{t-2} &= \phi_2 x_{t-2} + \theta w_{t-1} \\ x_t &= \phi_1 x_{t-1} + \phi_2 x_{t-2} + \theta w_{t-1} + w_t \end{aligned}$$

ARMAX model

An univariate ARMAX(p, q) model ($p > q$):

$$y_t = \Gamma u_t + \sum_{j=1}^{p} \phi_j y_{t-j} + \sum_{j=1}^{q} \theta_j w_{t-j} + w_t$$

State equation:

$$\mathbf{x}_{t+1} = \begin{pmatrix} \phi_1 & 1 & 0 & \cdots & 0 \\ \phi_2 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \phi_{p-1} & 0 & 0 & \cdots & 1 \\ \phi_p & 0 & 0 & \cdots & 0 \end{pmatrix} \mathbf{x}_t + \begin{pmatrix} 1 \\ \theta_1 \\ \vdots \\ \theta_q \\ 0 \\ \vdots \end{pmatrix} w_t + \begin{pmatrix} \Gamma \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix} u_t$$

Observation equation: $y_t = (1, 0, 0, ..., 0)\mathbf{x}_t$

Linear State Space Model

- Definition
- Kalman Filtering
- Kalman Smoothing
- Parameter Estimation
- Examples

Definition

Linear-Gaussian assumption:

$$\begin{aligned} a_0(\mathbf{x}) &= \mathcal{N}(\mathbf{x}; \mu_0, \Sigma_0) \\ a_t(\mathbf{x}, \mathbf{u}_t, \mathbf{x}_{t-1}) &= \mathcal{N}\left(\mathbf{x}; \mathbf{F}\mathbf{x}_{t-1} + \Upsilon\mathbf{u}_t, \Sigma\right) \\ b_t(\mathbf{y}, \mathbf{u}_t, \mathbf{x}_t) &= \mathcal{N}\left(\mathbf{y}; \mathbf{H}\mathbf{x}_t + \Gamma\mathbf{u}_t, \mathbf{R}\right) \end{aligned}$$

where $\mathbf{x}_t$ and μ_0 are $p \times 1$ vector, $\mathbf{y}_t$ are $q \times 1$ vector, $\mathbf{u}_t$ are $r \times 1$ vector, $\mathbf{F}$, Σ_0 and Σ are $p \times p$ matrix, $\mathbf{R}$ are $q \times q$ matrix, $\mathbf{H}$ are $q \times p$ matrix,, Υ_0 and Γ are $p \times r$ matrix.

$$\begin{aligned} \mathbf{x}_0 &= \mu_0 + \mathbf{w}_0; \quad \mathbf{w}_0 \sim \mathcal{N}(\mathbf{w}_0; 0, \Sigma_0) \\ \mathbf{x}_t &= \mathbf{F}\mathbf{x}_{t-1} + \Upsilon\mathbf{u}_t + \mathbf{w}_t; \quad \mathbf{w}_t \sim \mathcal{N}(\mathbf{w}_t; 0, \Sigma) \\ \mathbf{y}_t &= \mathbf{H}\mathbf{x}_t + \Gamma\mathbf{u}_t + \mathbf{v}_t; \quad \mathbf{v}_t \sim \mathcal{N}(\mathbf{v}_t; 0, \mathbf{R}) \end{aligned}$$

Kalman Filtering

- Recall that the conditional densities $p(\mathbf{x}_t|\mathbf{y}_{1:t})$ do store all information about the state space model up to time $t \longrightarrow$

$$p(\mathbf{x}_t|\mathbf{y}_{1:t}) = \mathcal{N}(\mathbf{x}_t; \mathbf{m}_{t|t}, \mathbf{P}_{t|t});$$

- At $t = 0$, $\mathbf{m}_{t|t} = \mu_0$ and $\mathbf{P}_{t|t} = \Sigma_0$;
- At time t, predictive mean and covariance matrix:

$$\begin{aligned}
\mathbf{m}_{t|t-1} &= \mathbb{E}\left[\mathbf{x}_t|\mathbf{y}_{1:t-1}\right] \\
&= \mathbb{E}\left[\mathbf{F}\mathbf{x}_{t-1} + \Upsilon\mathbf{u}_t + \mathbf{w}_t|\mathbf{y}_{1:t-1}\right] \\
&= \mathbf{F}\mathbf{m}_{t-1|t-1} + \Upsilon\mathbf{u}_t \\
\mathbf{P}_{t|t-1} &= \mathbb{E}\left[(\mathbf{x}_t - \mathbf{m}_{t|t-1})(\mathbf{x}_t - \mathbf{m}_{t|t-1})^\intercal\right] \\
&= \mathbb{E}\left[[\mathbf{F}(\mathbf{x}_{t-1} - \mathbf{m}_{t|t-1}) + \mathbf{w}_t][\mathbf{F}(\mathbf{x}_{t-1} - \mathbf{m}_{t|t-1}) + \mathbf{w}_t]^\intercal\right] \\
&= \mathbf{F}\mathbf{P}_{t-1|t-1}\mathbf{F}^\intercal + \Sigma
\end{aligned}$$

Conditional Gaussians

If $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are partitioned as follows:

$$\boldsymbol{\mu} = \begin{bmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \end{bmatrix}$$

and

$$\boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12} \\ \boldsymbol{\Sigma}_{21} & \boldsymbol{\Sigma}_{22} \end{bmatrix}$$

The distribution of $\boldsymbol{x}_1$ conditional on $\boldsymbol{x}_2 = a$:
mean vector

$$\bar{\boldsymbol{\mu}} = \boldsymbol{\mu}_1 + \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}\left(\mathbf{a} - \boldsymbol{\mu}_2\right)$$

and covariance matrix

$$\overline{\boldsymbol{\Sigma}} = \boldsymbol{\Sigma}_{11} - \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}\boldsymbol{\Sigma}_{21}.$$

Kalman Filtering (cont.)

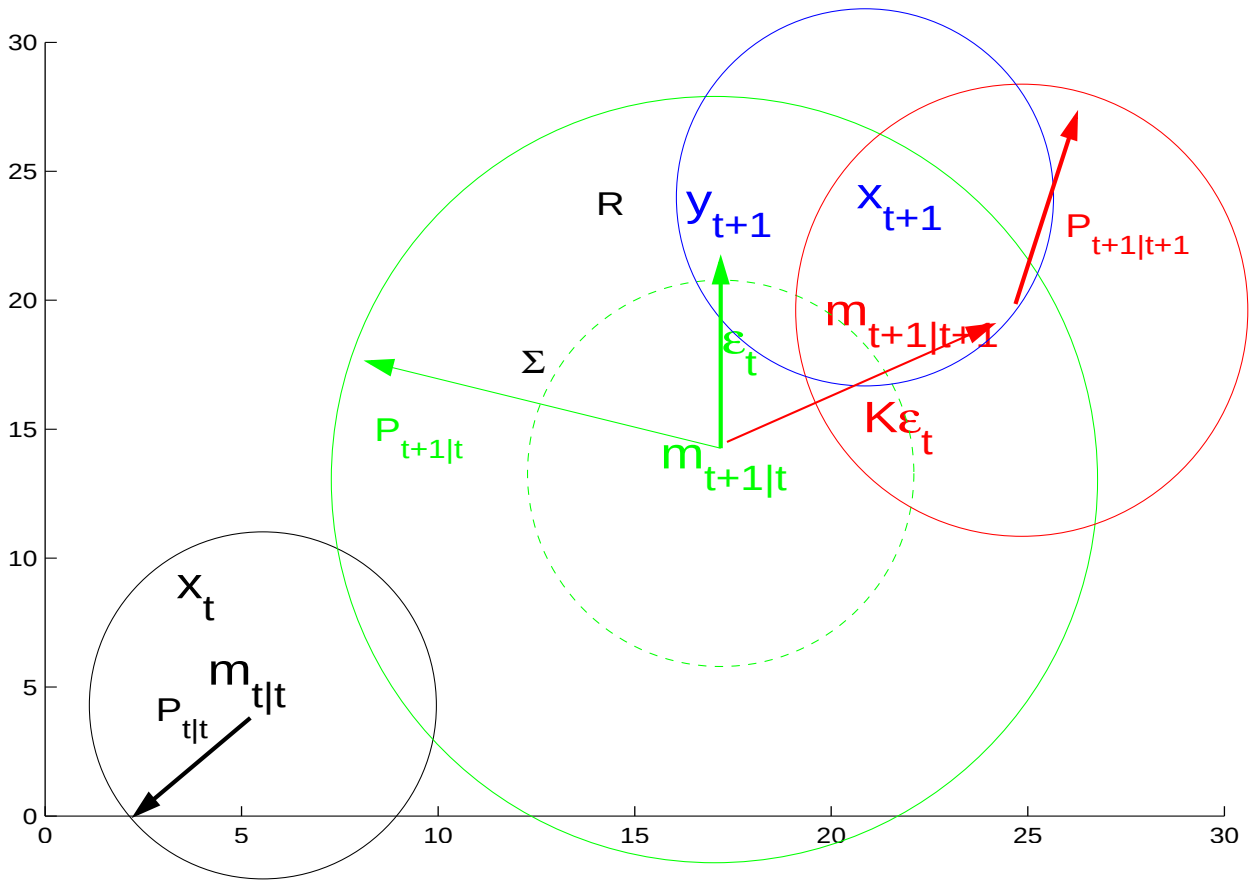
- The innovations:

$$\epsilon_t = \mathbf{y}_t - \mathbb{E}\left[\mathbf{y}_t | \mathbf{y}_{1:t-1}\right] = \mathbf{y}_t - \Gamma \mathbf{u}_t - \mathbf{H}\mathbf{m}_{t|t-1}$$

- Conditional on ϵ_t, the conditional Gaussian $\mathcal{N}(\mathbf{x}_t; \mathbf{m}_{t|t}, \mathbf{P}_{t|t})$ can be derived from the joint Gaussian $\mathcal{N}(\mathbf{x}_t, \epsilon_t; \mathbf{m}_{\mathbf{x}_t,\epsilon_t}, \mathbf{P}_{\mathbf{x}_t,\epsilon_t})$;
- $\mathbb{E}\left[\epsilon_t\right] = 0$;
- var(ϵ_t) = $\mathbf{H}\mathbf{P}_{t|t-1}\mathbf{H}^\intercal + \mathbf{R}$;
- cov($\mathbf{x}_t, \epsilon_t | \mathbf{y}_{1:t}$) = $\mathbf{P}_{t|t-1}\mathbf{H}^\intercal$;
- $m_{t|t} = m_{t|t-1} + \mathbf{K}_t \epsilon_t$
- $\mathbf{P}_{t|t} = \mathbf{P}_{t|t-1} - \mathbf{K}_t \mathbf{H} \mathbf{P}_{t|t-1}$
- The Kalman gain: $\mathbf{K}_t = (\text{cov}(\mathbf{x}_t, \epsilon_t | \mathbf{y}_{1:t})) \cdot (\text{var}(\epsilon_t))^{-1}$.

Kalman Filtering (cont.)

Interpretation of $\mathbf{K}_t$:



Kalman Smoothing

- Given the observation sequence $\{\mathbf{y}_1, ..., \mathbf{y}_T\}$:

$$p(\mathbf{x}_t|\mathbf{y}_{1:T}) = \mathcal{N}(\mathbf{x}_t; \mathbf{m}_{t|T}, \mathbf{P}_{t|T});$$

- The conditional means $\mathbf{m}_{t-1|T}$ and $\mathbf{m}_{t|T}$ are those values that maximize the joint Gaussian density $p(\mathbf{x}_{t-1}, \mathbf{x}_t|\mathbf{y}_{1:T})$;
- The conditional joint density $p(\mathbf{x}_{t-1}, \mathbf{x}_t|\mathbf{y}_{1:T})$ has the forms

$$p(\mathbf{x}_{t-1}, \mathbf{x}_t|\mathbf{y}_{1:T}) \propto p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1}) \cdot p(\mathbf{x}_t|\mathbf{x}_{t-1});$$

- Suppose we have $\mathbf{m}_{t|T}$ avaiable, the $\mathbf{m}_{t-1|T}$ is found by maximizing

$$\mathcal{N}(\mathbf{x}_{t-1}; \mathbf{m}_{t-1|t-1}, \mathbf{P}_{t-1|t-1}) \cdot \mathcal{N}(\mathbf{m}_{t|T}; \mathbf{F}\mathbf{x}_{t-1}, \Sigma);$$

Kalman Smoothing(cont.)

- $\mathbf{m}_{t-1|T} = \mathbf{m}_{t-1|t-1} + \mathbf{J}_{t-1}(\mathbf{m}_{t|T} - \mathbf{m}_{t|t-1})$;
- $\mathbf{P}_{t-1|T} = \mathbf{P}_{t-1|t-1} + \mathbf{J}_{t-1}(\mathbf{P}_{t|T} - \mathbf{P}_{t|t-1})\mathbf{J}_{t-1}^{\intercal}$;
- $\mathbf{J}_{t-1} = \mathbf{P}_{t-1|t-1}\mathbf{F}^{\intercal}\left(\mathbf{P}_{t|t-1}\right)^{-1}$;
- Interpretation of $\mathbf{J}_{t-1}$:
  - Recall that $\mathbf{P}_{t|t-1} = \mathbf{F}\mathbf{P}_{t-1|t-1}\mathbf{F}^{\intercal} + \Sigma$;
  - Note that $\mathbf{J}_{t-1}$ = 1, if $\mathbf{F}$ is a identity matrix and $\Sigma = 0$.

There are many equivalent forms of the smoother in the literature. They differ numerically with respect to speed and accuracy.

Parameter Estimation

- The parameter set to estimate is $\theta = \{\mathbf{F}, \Sigma, \mathbf{R}\}$;
- The incomplete data likelihood

$$-2 \log L_{\mathbf{Y}}(\Theta) = \sum_{t=1}^{T} \log |\text{var}(\epsilon_t(\Theta))| + \sum_{t=1}^{T} \epsilon_t(\Theta)^{\intercal} \text{var}(\epsilon_t(\Theta))^{-1} \epsilon_t(\Theta));$$

- The complete data likelihood

$$\begin{aligned} -2 \log L_{\mathbf{X},\mathbf{Y}}(\Theta) &= \log |\Sigma_0| + (\mathbf{x}_0 - \mu_0)^{\intercal} \Sigma_0^{-1} (\mathbf{x}_0 - \mu_0) \\ &+ \log |\Sigma| + \sum_{t=1}^{T} (\mathbf{x}_t - \mathbf{F}\mathbf{x}_{t-1})^{\intercal} \Sigma^{-1} (\mathbf{x}_t - \mathbf{F}\mathbf{x}_{t-1}) \\ &+ \log |\mathbf{R}| + \sum_{t=1}^{T} (\mathbf{y}_t - \mathbf{H}\mathbf{x}_t)^{\intercal} \Sigma^{-1} (\mathbf{y}_t - \mathbf{H}\mathbf{x}_t); \end{aligned}$$

Parameter Estimation (cont.)

- The Q function at iteration j

$$Q\left(\Theta|\Theta^{(j-1)}\right) = \mathbb{E}\left[-2\log L_{\mathbf{X},\mathbf{Y}}(\Theta)|\mathbf{y}_{1,\ldots,T}, \Theta^{(j-1)}\right],$$

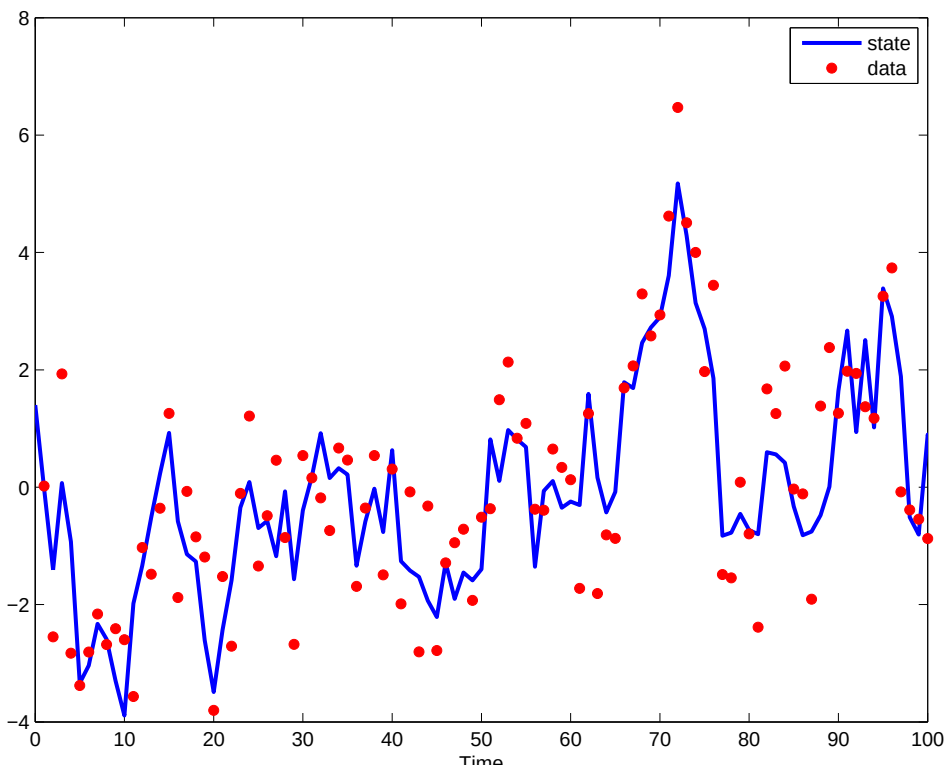
  which is an upper bound on $\log L_{\mathbf{Y}}(\Theta)$;

- The algorithm
  1. Initialize $\Theta^{(0)}$ and set j = 1;
  2. Compute $\log L_{\mathbf{Y}}(\Theta^{(j-1)})$;
  3. (E-Step) Perform Kalman filtering and smoothing using $\Theta^{(j-1)}$;
  4. (M-step) Use the smoothed estimates ($\mathbf{m}_{t|T}$, $\mathbf{P}_{t|T}$, ...) to compute $\Theta^{(j)}$ that maximizes $Q\left(\Theta|\Theta^{(j-1)}\right)$;
  5. Repeat Steps 2 - 4 for $\log L_{\mathbf{Y}}(\Theta^{(j-1)})$ to converge.

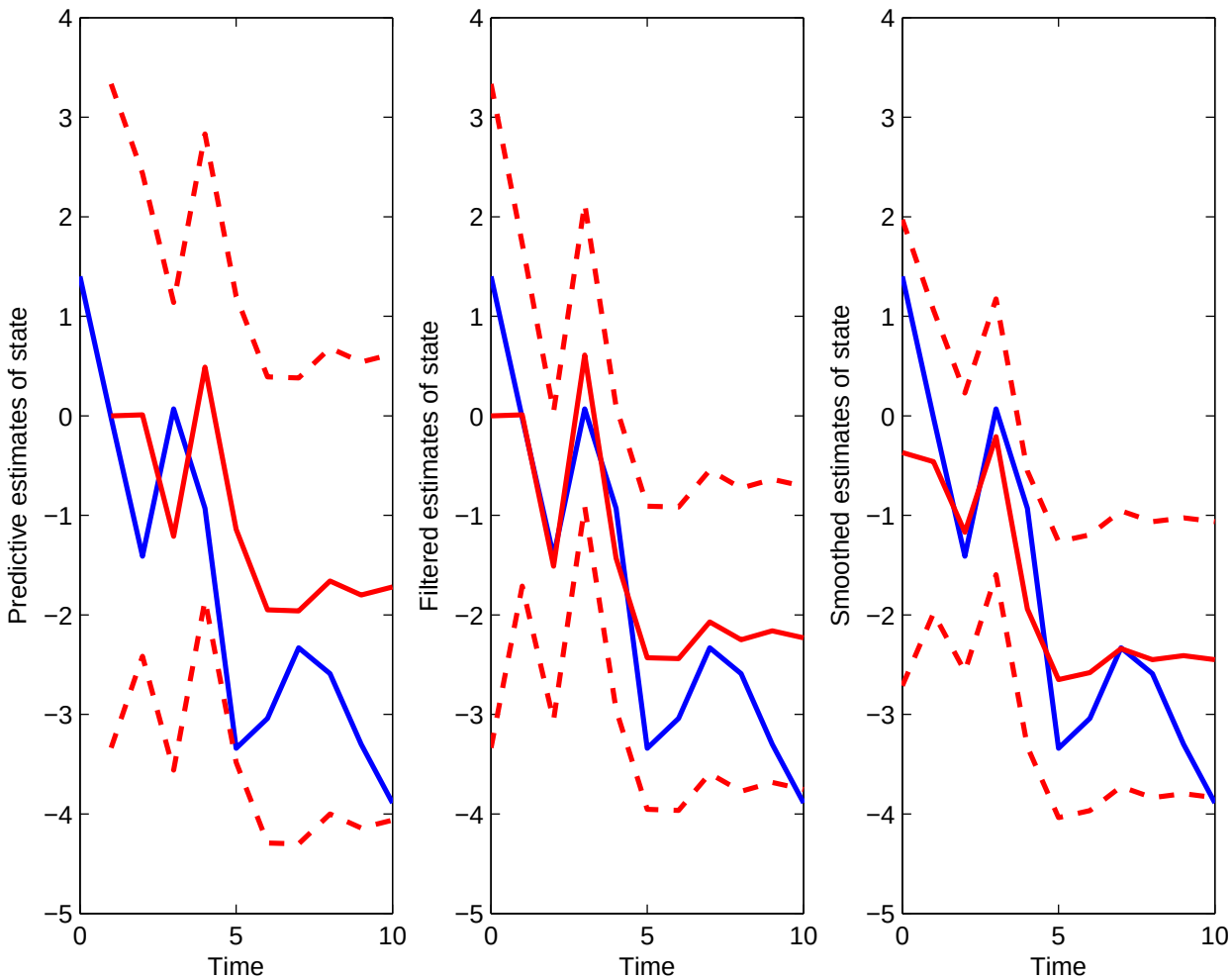
Example

$$\begin{aligned} x_t &= \phi x_{t-1} + w_t \\ y_t &= x_t + w_t, \end{aligned}$$

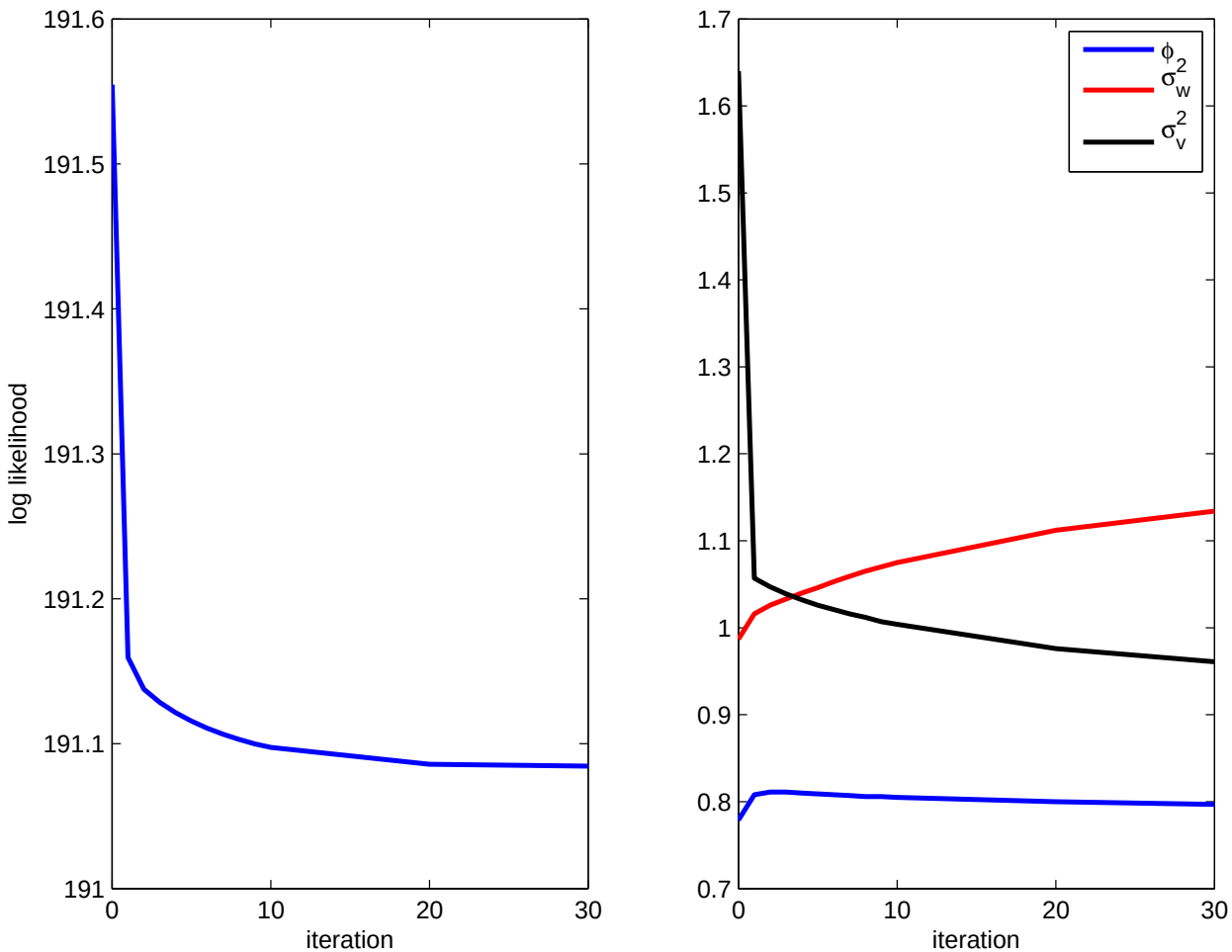
with $\phi = 0.8$, $\sigma_w^2 = 1.0$, and $\sigma_v^2 = 1.0$.



Example (cont.)



Example (cont.)



Non-linear state space models

- Extended Kalman filtering
- Non-linear filtering
- Non-linear smoothing
- Parameter estimation
- Example

Extended Kalman filtering

Nonlinear state equation:

$$\mathbf{X}_t = \mathbf{F}(\mathbf{X}_{t-1}) + \mathbf{W}_t \quad \text{and} \quad \mathbf{Y}_t = \mathbf{H}\mathbf{X}_t + \mathbf{V}_t$$

Linearization:

$$\mathbf{X}_t = \mathbf{F}_t(\mathbf{X}_{t-1}) \approx \mathbf{F}_t(\mathbf{m}_{t-1|t-1}) + \left[\frac{d\mathbf{F}(\mathbf{X}_t)}{d\mathbf{X}_t}\right]_{\mathbf{m}_{t-1|t-1}} (\mathbf{X}_{t-1} - \mathbf{m}_{t-1|t-1}).$$

Prediction:

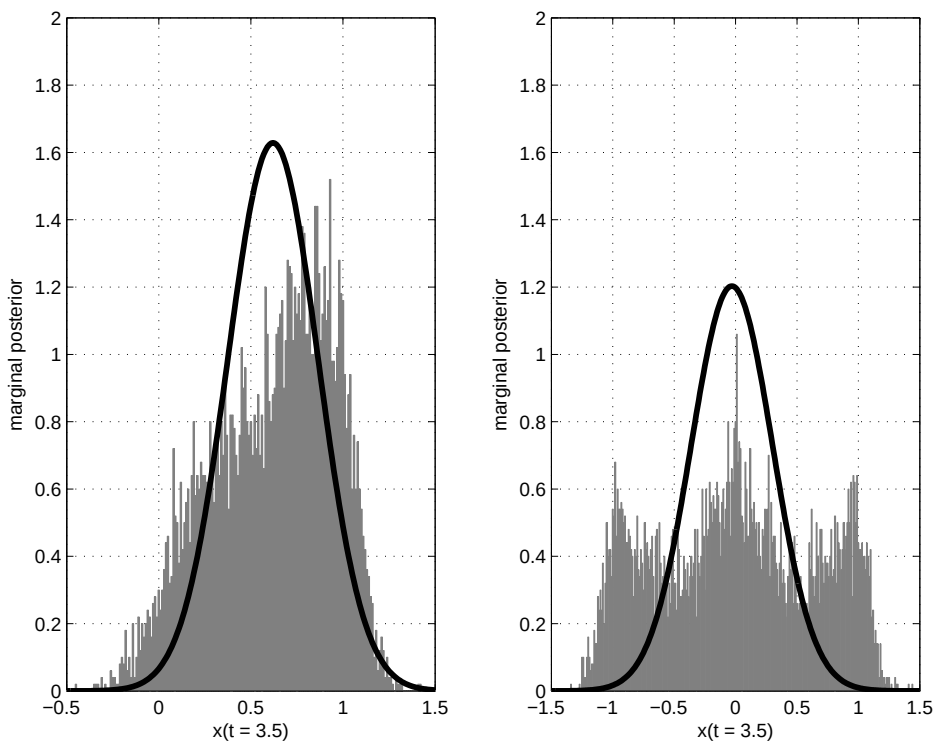
$$\mathbf{m}_{t|t-1} = \mathbf{F}(\mathbf{m}_{t-1|t-1})$$

$$\mathbf{P}_{t|t-1} = \left[\frac{d\mathbf{F}(\mathbf{X}_t)}{d\mathbf{X}_t}\right]_{\mathbf{m}_{t-1|t-1}} \mathbf{P}_{t-1|t-1} \left[\frac{d\mathbf{F}(\mathbf{X}_t)}{d\mathbf{X}_t}\right]^{\mathsf{T}}_{\mathbf{m}_{t-1|t-1}} + \Sigma$$

Extended Kalman filtering (cont.)

Problems:

- Conditional densities are assumed to be Gaussian;



- No information about approximation error

Extended Kalman filtering (cont.)

Particle approximation of $p(x)$

- Draw N samples from $p(x)$: $\{x_1, x_2, ..., x_N\}$;
- Approximate $p(x)$ by $\tilde{p}(x) \approx \frac{1}{N} \sum_{k=1}^{N} \delta(x - x_k)$;
- Capability of approximating multimodal distribution;
- When $N \to \infty$, $\tilde{p}(x) \to p(x)$.
- If a direct draw is impossible, Markov Chain Monte Carlo or Sequential Monte Carlo.

Non-linear filtering

- filtering densities: $f_{t|t}(\mathbf{x}_t) = p(\mathbf{x}_t|\mathbf{y}_{1:t})$;
- prediction densities for state: $f_{t+1|t}(\mathbf{x}_{t+1}) = p(\mathbf{x}_{t+1}|\mathbf{y}_{1:t})$;
- filtering: $f_{t|t} \longrightarrow f_{t+1|t} \longrightarrow f_{t+1|t+1}$;
- Markov transition: $f_{t+1|t}(\mathbf{x}_{t+1}) = \int a_t(\mathbf{x}_{t+1}, \mathbf{x}) f_{t|t}(\mathbf{x}) d\mathbf{x}$
- Bayes' formula: $p(\mathbf{x}|\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{x}) \cdot p(\mathbf{x})}{\int p(\mathbf{y}|\mathbf{x}) \cdot p(\mathbf{x}) d\mathbf{x}} \longrightarrow$

$$f_{t+1|t+1}(\mathbf{x}_{t+1}) = \frac{b_t(\mathbf{y}_{t+1}, \mathbf{x}_{t+1}) f_{t+1|t}(\mathbf{x}_{t+1})}{\int b_t(\mathbf{y}_{t+1}, \mathbf{x}) f_{t+1|t}(\mathbf{x}) d\mathbf{x}}.$$

- prediction densities for observation:

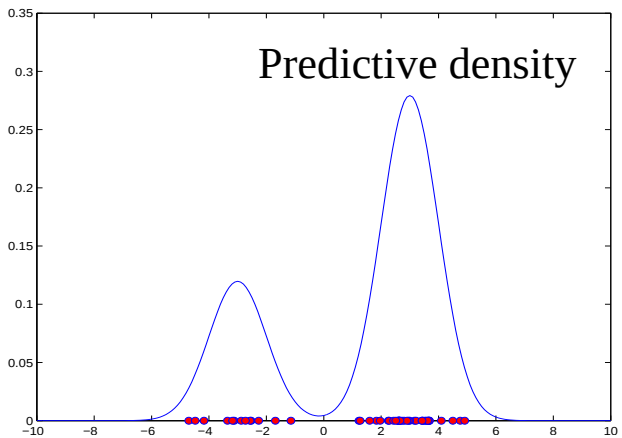
$$p_{t+1|t}(\mathbf{y}_{t+1}|\mathbf{y}_{1:t}) = \int b_t(\mathbf{y}_{t+1}, \mathbf{x}) f_{t+1|t}(\mathbf{x}) d\mathbf{x};$$

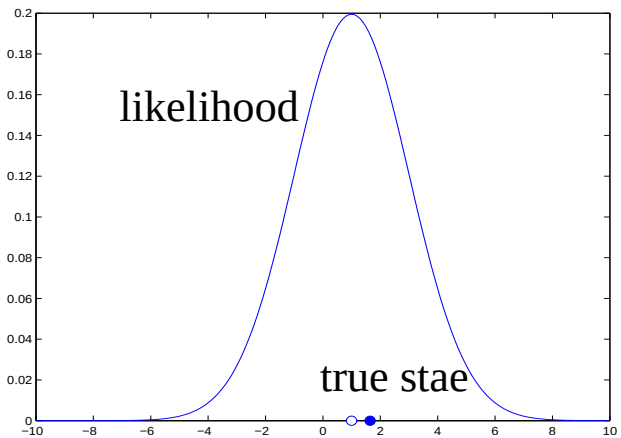
Non-linear filtering (cont.)

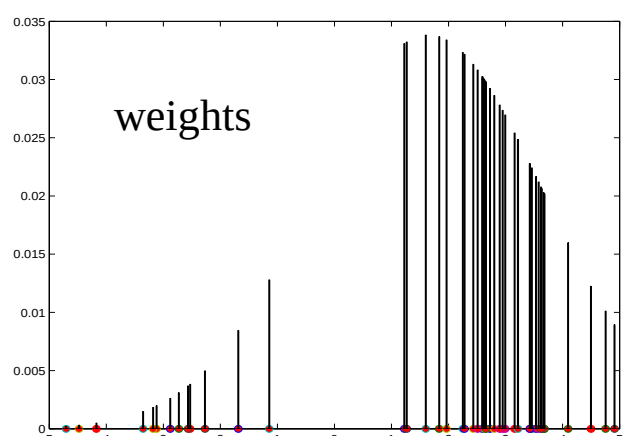
The standard particle filtering

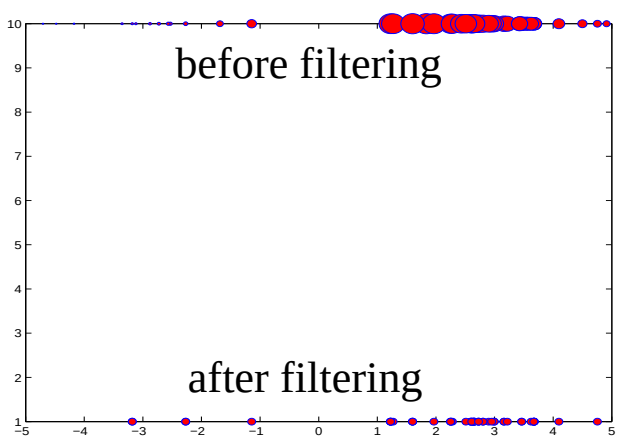
1. Generate $(x_0^1, ..., x_0^N)$ from $a_0(x)$ and set t = 0;
2. We have $(x_t^1, ..., x_t^N)$ for an particle approximation of $f_{t|t}(x_t)$;
3. Generate $\tilde{x}_{t+1}^j \sim a_{t+1}(x_t^j, x)$, j = 1, ..., N, to obtain an particle approximation of $f_{t+1|t}(x_{t+1})$;
4. Compute weights $\lambda_{t+1}^j = \frac{b_{t+1}(\tilde{x}_{t+1}^j, y_{t+1})}{\sum_k b_{t+1}(\tilde{x}_{t+1}^k, y_{t+1})}$;
5. Generate I_j with $p[I_j = k] = \lambda_{t+1}^j$ and set x_{t+1}^j to $\tilde{x}_{t+1}^{I_j}$;
6. Increase t by 1 and go back to Step 2.

Nonlinear Filtering (cont.)

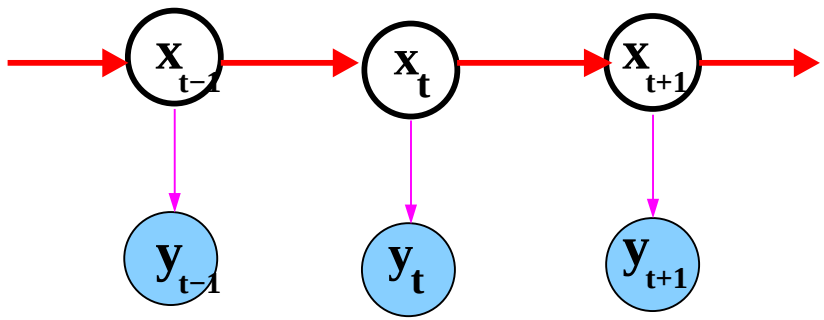








Nonlinear smoothing



- Bayes' formula

$$p(\mathbf{x}_t|\mathbf{x}_{t+1},\mathbf{y}_{1:T}) = p(\mathbf{x}_t|\mathbf{x}_{t+1},\mathbf{y}_{1:t}) = \frac{a_{t+1}(\mathbf{x}_t,\mathbf{x}_{t+1})f_{t|t}(\mathbf{x}_t|\mathbf{y}_{1:t})}{f_{t+1|t}(\mathbf{x}_{t+1}|\mathbf{y}_{1:t})}$$

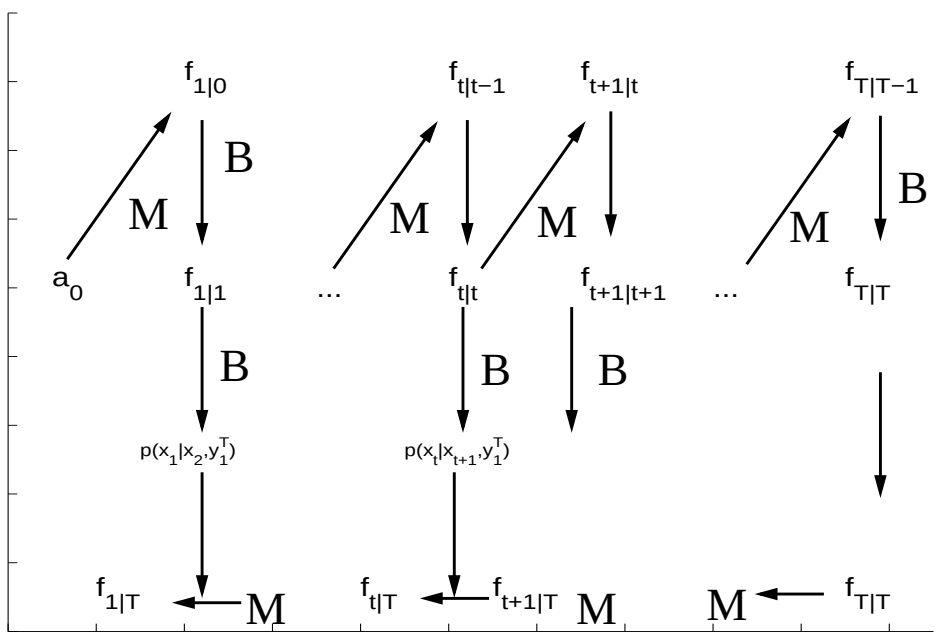
- Markov transition

$$p(\mathbf{x}_t|\mathbf{y}_{1:T}) = \int p(\mathbf{x}_t|\mathbf{x}_{t+1},\mathbf{y}_{1:T}) \cdot p(\mathbf{x}_{t+1}|\mathbf{y}_{1:T}) d\mathbf{x}_{t+1}.$$

Nonlinear smoothing (cont.)

Smoothing densities:

$$f_{t|T}(\mathbf{x}_t) = f_{t|t}(\mathbf{x}_t) \int \frac{a_{t+1}(\mathbf{x}_t, \mathbf{x})}{f_{t+1|t}(\mathbf{x})} \cdot f_{t+1|T}(\mathbf{x}) dx.$$



Non-linear smoothing (cont.)

We have $(x^1_{t|t}, ..., x^N_{t|t})$ for t = 1, ..., T representing filtering densities $f_{t|t}(x_t)$.To generate $x^j_{t|T}$ given $x^j_{t+1|T}$, we sample from

$$\begin{aligned} g_t(x) &\propto a_{t+1}(x, x^j_{t+1|T}) f_{t|t}(x|y_{1:t}) \\ &\propto a_{t+1}(x, x^j_{t+1|T}) \cdot \left\{ b_t(x, y_t) \sum_{i=1}^{N} a_t(x^i_{t|t}, x) \right\}, \end{aligned}$$

using accept reject method, with the index as auxiliary variable;

1. Generate I uniform on $\{1, ..., N\}$ and x from $a(x^I_{t-1|t-1}, x)$;
2. Generate U uniform on $[0, c^j_t]$. If $U < b_t(x, y_t) a_{t+1}(x, x^j_{t+1|T})$, put $x^j_{t|T} = x$.

Parameter estimation

Assume now that both a_t and b_t depende on a finite dimensional parameter Θ. The the likelihood of Θ given the observed series $y_{1:T}$

$$p(y_{1:T}|\theta) = \prod_{t=1}^{T} p(y_t|y_{1:t-1;\Theta}) = \prod_{t=1}^{T} \int b_t(x, y_t; \Theta) f_{t|t-1}(x|y_{1:t-1}; \Theta) dx.$$

Each factor on the right is obtained as a normalization during the filter recursion.

$$\int b_t(x, y_t; \Theta) f_{t|t-1}(x|y_{1:t-1}; \Theta) dx = \frac{1}{N} \sum_{j=1}^{N} b_t(x^j_{t|t-1}, y_t)$$

Maximization can be done by a general purpose optimization algorithm.

Parameter estimation(cont.)

Since the joint likelihood of the observations and the hidden states can be written explicitly, using the EM algorithm is natural.In the E-step, we have to compute

$$Q(\Theta, \Theta') = \mathbb{E}\left[\log p(x_{0:T}, y_{1:T}; \Theta | y_{1:T}, \Theta'\right]$$

where Θ' is the current approximation to the MLE. In the M-step, we maximize $Q(\Theta, \Theta')$ with respect to Θ. The function $Q(\Theta, \Theta')$ contains the terms

$$\mathbb{E}\left[\log a_t(x_{t-1}, x_t; \Theta) | y_{1:T}, \Theta'\right] \quad \text{and} \quad \mathbb{E}\left[\log b_t(x_t, y_t; \Theta) | y_{1:T}, \Theta'\right],$$

which can be computed by using smoothed particle $x^j_{t|T}$, j = 1, ..., N.