Extended Elman Network for Bayesian Data Assimilation

Sixin Zhang
Université de Toulouse, INP, IRIT

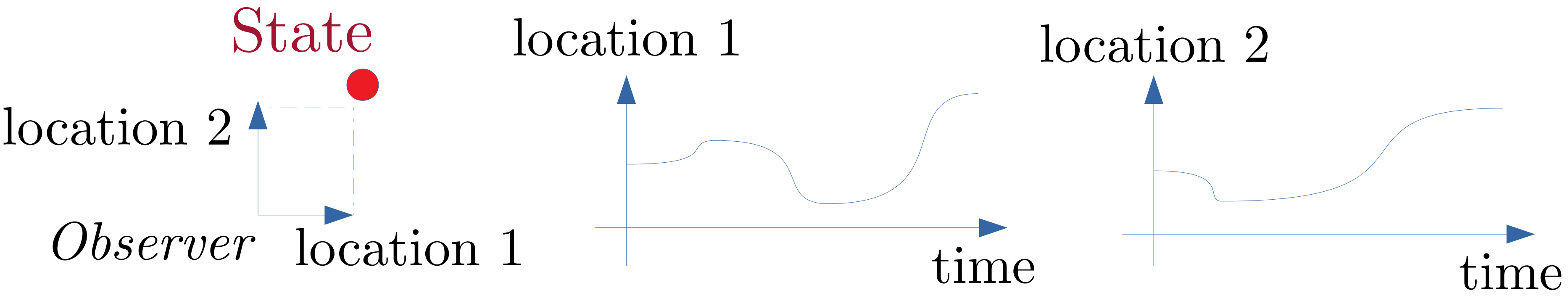
Joint work with P. Boudier, A. Fillion, S. Gratton, S. Gurol

Outline

- Data assimilation of chaotic dynamical systems
- State-of-the art methods
- Main contribution: Data assimilation networks (DAN)
- Conclusion and perspectives

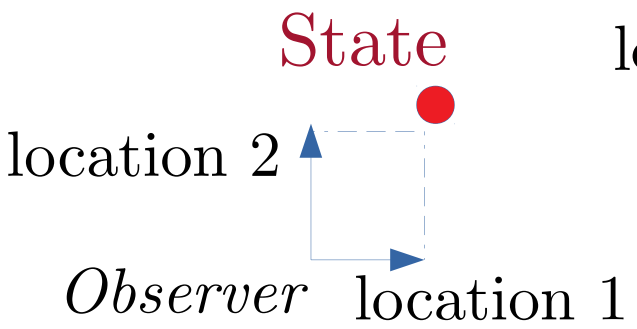
Chaotic dynamical system

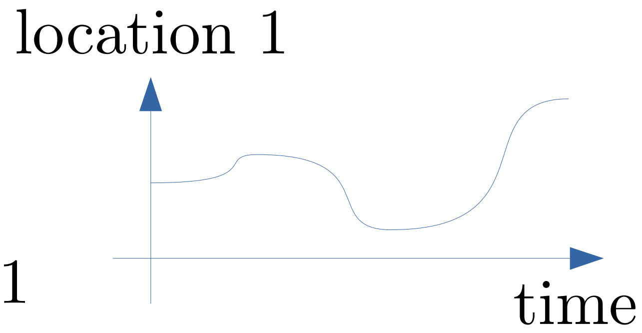
- A (dynamical) system describes the change of state variables

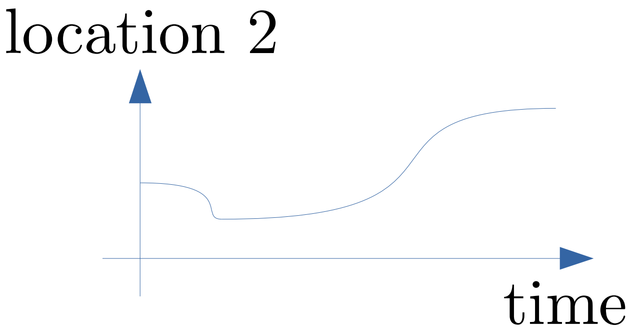


Chaotic dynamical system

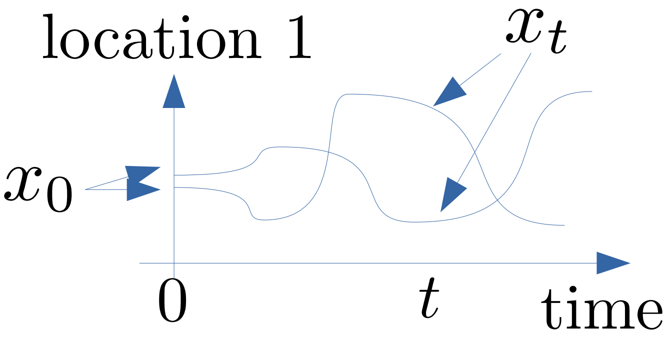
- A (dynamical) system describes the change of state variables







- A chaotic system is highly sensitive to initial conditions



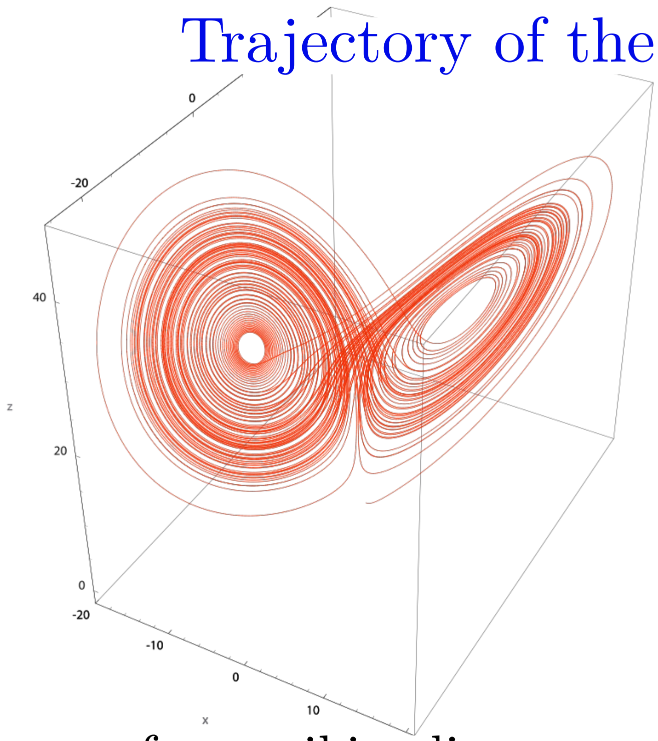
A small change of state x_0

$\Rightarrow$ large change of state x_t

Lorenz chaotic system

- Lorenz proposes in 1963 a chaotic system model in 3d (3 state variables)

Trajectory of the states over time

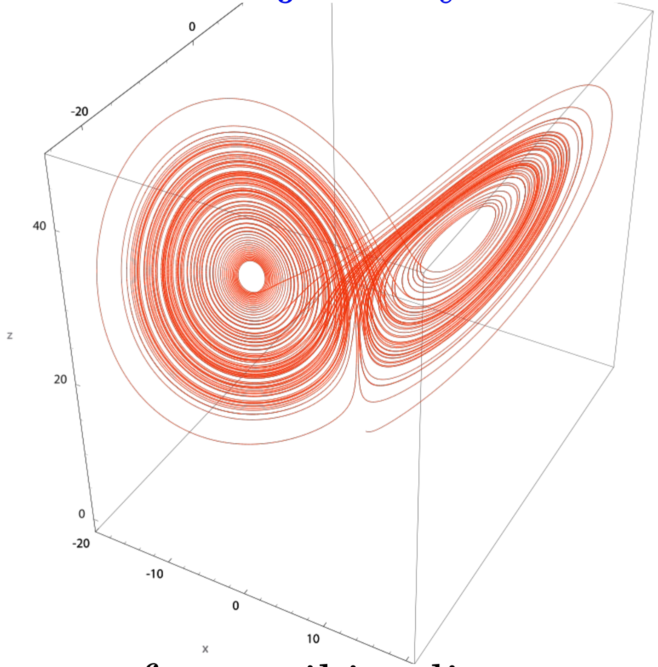


- from wikipedia

Lorenz chaotic system

- Lorenz proposes in 1963 a chaotic system model in 3d (3 state variables)

Trajectory of the states over time



- from wikipedia

- Extensible to more than 3d

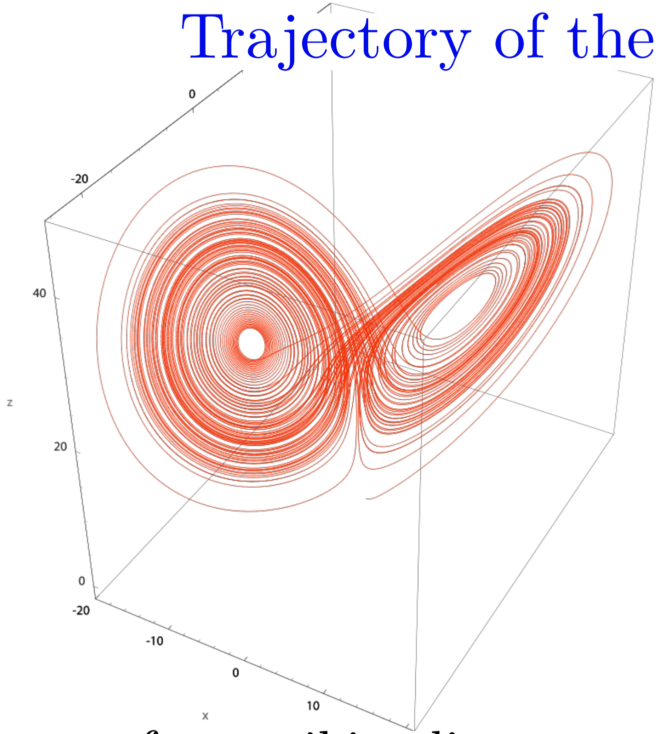
"Predictability: A problem partly solved"

$-Lorenz, 1995$

Lorenz chaotic system

- Lorenz proposes in 1963 a chaotic system model in 3d (3 state variables)

Trajectory of the states over time



- from wikipedia

- Extensible to more than 3d

"Predictability: A problem partly solved"

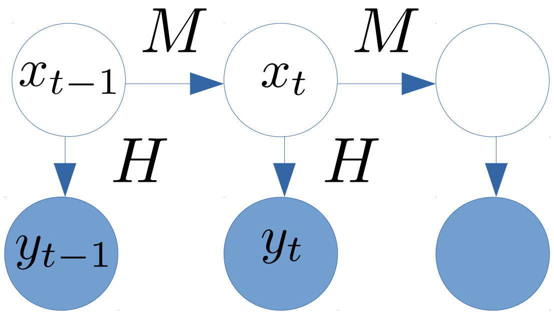
–*Lorenz*, 1995

- A toy model in Atmospheric/Ocean science

- In practice, consider high dimension systems (beyond Lorenz with more than 10^7 variables)

Data assimilation of chaotic system

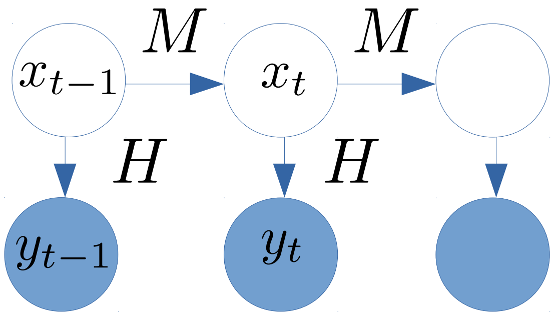
- Forcast the state x_{t+1} from noisy observations $y_1, \cdots, y_t$



$$\mathit{Assumptions:}\quad \begin{aligned} x_{t+1} &= M(x_t) + \text{ noise} \\ y_t &= H(x_t) + \text{ noise} \end{aligned}$$

Data assimilation of chaotic system

- Forcast the state x_{t+1} from noisy observations $y_1, \cdots, y_t$



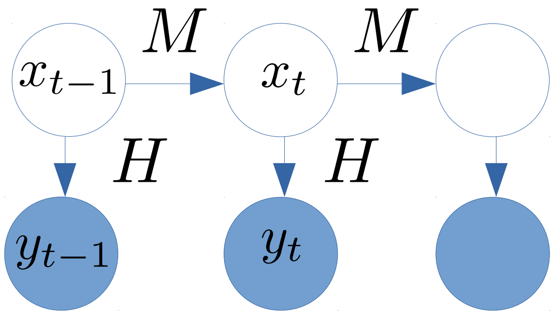
$$\mathit{Assumptions:} \quad x_{t+1} = M(x_t) + \text{ noise}$$
$$y_t = H(x_t) + \text{ noise}$$

$\mathit{Examples:}$

x_t	location	temperature
y_t	GPS	thermometer

Data assimilation of chaotic system

- Forcast the state x_{t+1} from noisy observations $y_1, \cdots, y_t$



Assumptions : $x_{t+1} = M(x_t) +$ noise

$y_t = H(x_t) +$ noise

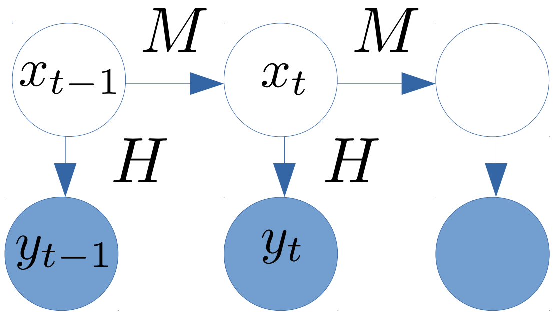
Examples :

x_t	location	temperature
y_t	GPS	thermometer

- In a chaotic system M, we estimate the probability of states

Data assimilation of chaotic system

- Forcast the state x_{t+1} from noisy observations $y_1, \cdots, y_t$



Assumptions : $x_{t+1} = M(x_t) +$ noise

$y_t = H(x_t) +$ noise

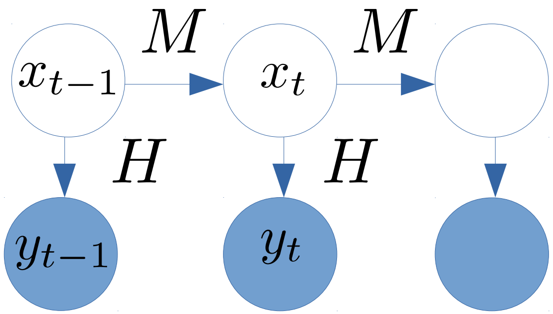
Examples :

x_t	location	temperature
y_t	GPS	thermometer

- In a chaotic system M, we estimate the probability of states
- An inference problem of latent models in Machine learning

Inference in latent models

- Predict the state variable $\mathbf{x}$ given the observed variable $\mathbf{y}$

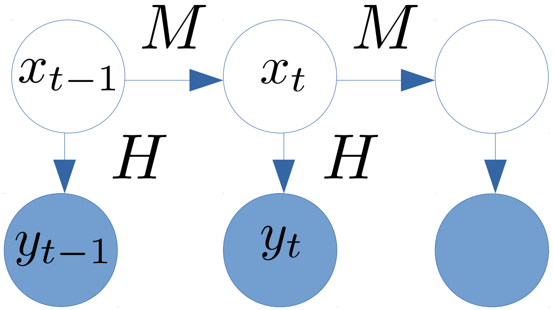


p_t^b: probability of $\mathbf{x} = x_t$ given $\mathbf{y} = (y_1, \cdots y_{t-1})$

p_t^a: probability of $\mathbf{x} = x_t$ given $\mathbf{y} = (y_1, \cdots y_t)$

Inference in latent models

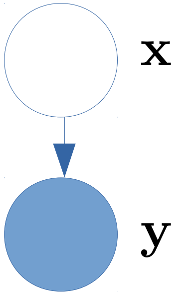
- Predict the state variable $\mathbf{x}$ given the observed variable $\mathbf{y}$



p_t^b: probability of $\mathbf{x} = x_t$ given $\mathbf{y} = (y_1, \cdots y_{t-1})$

p_t^a: probability of $\mathbf{x} = x_t$ given $\mathbf{y} = (y_1, \cdots y_t)$

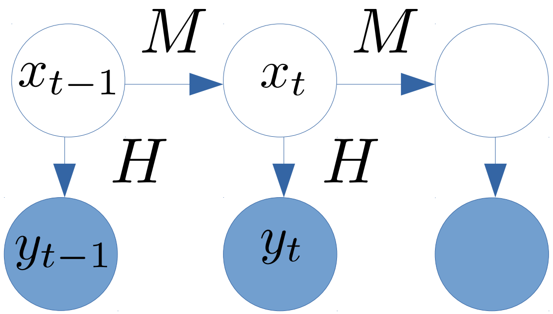
- Two common frameworks in Machine learning



How to estimate the probability of $\mathbf{x}$ *given* $\mathbf{y}$*?*

Inference in latent models

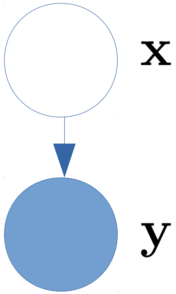
- Predict the state variable $\mathbf{x}$ given the observed variable $\mathbf{y}$



p_t^b: probability of $\mathbf{x} = x_t$ given $\mathbf{y} = (y_1, \cdots y_{t-1})$

p_t^a: probability of $\mathbf{x} = x_t$ given $\mathbf{y} = (y_1, \cdots y_t)$

- Two common frameworks in Machine learning



How to estimate the probability of $\mathbf{x}$ *given* $\mathbf{y}$*?*

1. Maximum **likelihood** estimation: Variational Auto-encorder
2. **Likelihood-free** estimation: Generative adversarial network

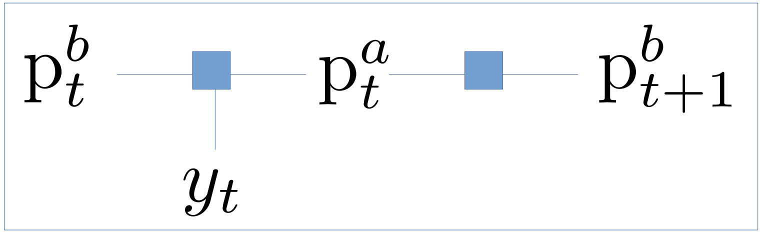
Key problem of Data assimilation

- How can we compute these probabilities for $t = 1, 2, \cdots$?

$$\mathrm{p}_t^b\text{: probability of } x_t \text{ given } (y_1, \cdots y_{t-1})$$

$$\mathrm{p}_t^a\text{: probability of } x_t \text{ given } (y_1, \cdots y_t)$$

- Ideal solution based on a recurrent **Bayesian rule**



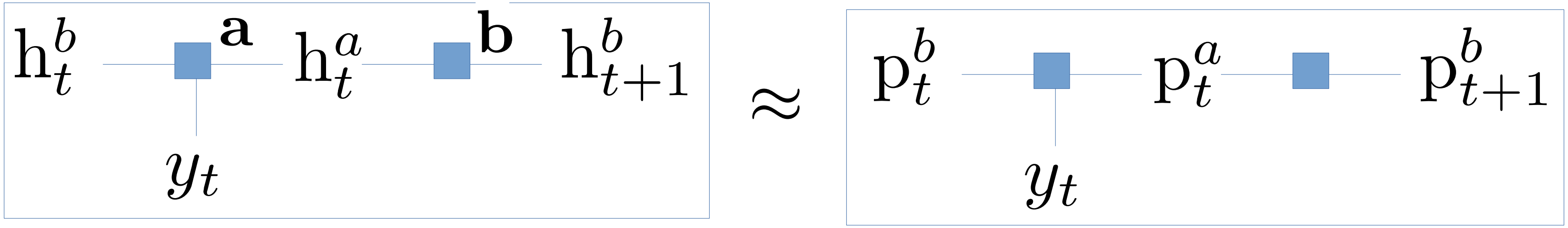
$$\mathrm{p}_t^a(x_t|y_t) = \frac{p(y_t|x_t)p_t^b(x_t)}{\int p(y_t|x)p_t^b(x)dx}$$

$$\mathrm{p}_{t+1}^b(x_{t+1}) = \int p(x_{t+1}|x_t)p_t^a(x_t|y_t)dx_t$$

Time-invariant, but *intractable* transform

State-of-the art methods

- Recurrence in Ensemble Kalman Filter methods
  - Use h_t^a, h_t^b to estimate the moments of p_t^a, p_t^b (mean/covariance)



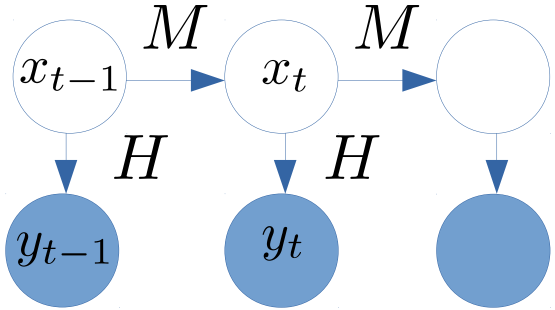
  - **Key**: derive update rules for **a** and **b** under limited ensemble size

$$\textit{ensemble size} = \textit{dimension of ensemble } h_t^a,\ h_t^b$$

  - Performance sensitive to the choice of the ensemble size
  - Introduce localization and inflation regularisation techniques (reduce the sampling noise)

State-of-the art methods

- Machine learning methods fall roughly into 3 settings



  - Supervised: Use (x_t, y_t) to estimate p_t^a, p_t^b

    Fablet et al. (2021): state information (mode)

    Revach et al. (2022): mean/covariance

  - Self-supervised: Use y_t and M to estimate p_t^a, p_t^b (moments)

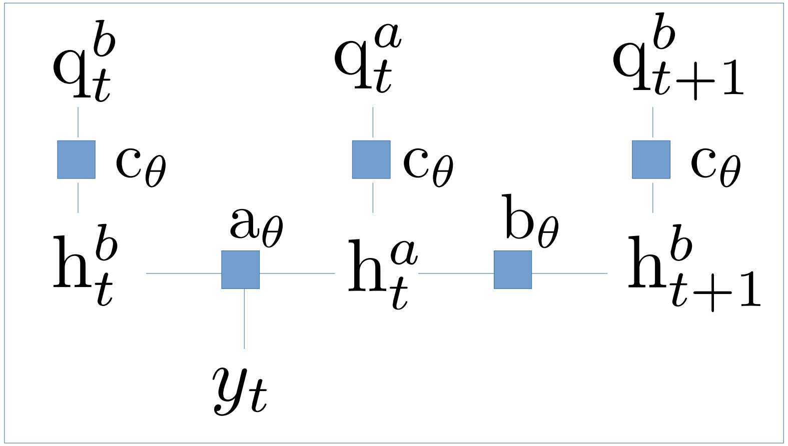
    Harter and de Campos Velho (2012), McCabe and Brown (2021)

  - Unsupervised: Use y_t to learn M (and estimate modes of p_t^a, p_t^b)

    Bocquet et al. (2019,2020), Brajard et al. (2020)

Data Assimilation Networks

Idea: learn $(a_\theta, b_\theta, c_\theta)$ to generate $(q^b_{t,\theta}, q^a_{t,\theta}) \approx (p^b_t, p^a_t)$

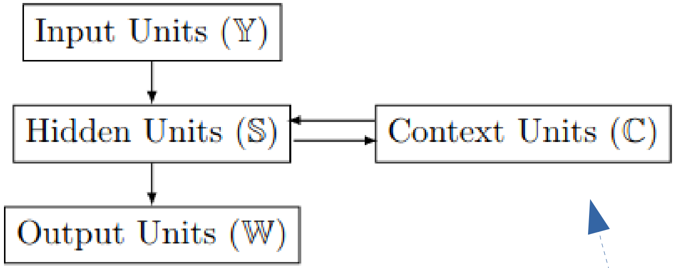


memory size $=$ *dimension of* h^a_t, h^b_t (ensemble size)

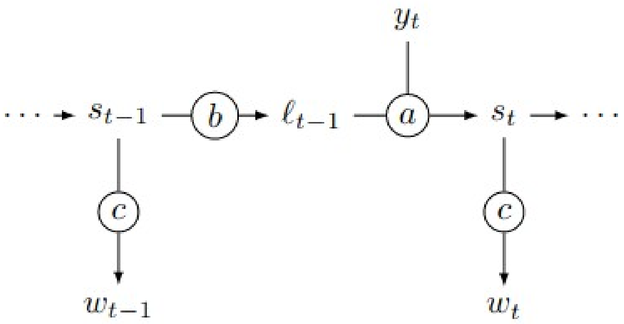
- Parameterize $(a_\theta, b_\theta, c_\theta)$ with θ using Recurrent Neural networks
- Objective function: $\min_\theta L(\theta) = \mathbb{E}\left(\frac{1}{T}\sum_{t=1}^{T} KL(q^a_{t,\theta}|p^a_t) + KL(q^b_{t,\theta}|p^b_t)\right)$
- A general framework with no Gaussian assumptions on $q^a_{t,\theta}, q^b_{t,\theta}$

DAN as Extended Elman Network

Elman Network



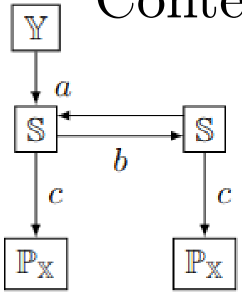
(a) Scheme of a Elman Network



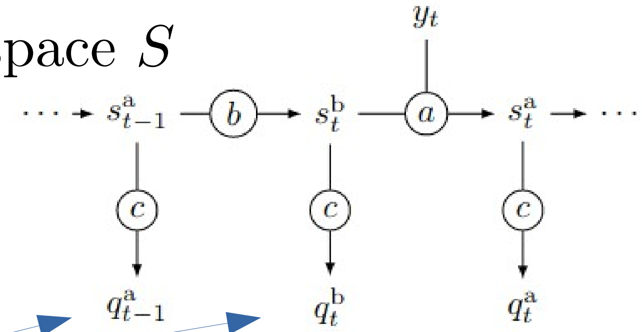
(b) Scheme of a recursive Elman application

DAN

Context C = Memory space S



(a) Scheme of a DAN



(b) Scheme of a recursive DAN application

Two outputs instead of one at each time

Main results

- Question: What happens to the case $q^b_{t,\theta}, q^a_{t,\theta}$ are Gaussian distributions?

Informal Theorem: Assume $q^a_{t,\theta} := \mathcal{N}(\mu^a_{t,\theta}, \Sigma^a_{t,\theta})$ and $q^b_{t,\theta} := \mathcal{N}(\mu^b_{t,\theta}, \Sigma^b_{t,\theta})$, then with infinite memory size the optimal θ^* of the objective $L(\theta)$ satisfies

$$\mu^a_{t,\theta^*} \text{ is the mean of } p^a_t, \quad \mu^b_{t,\theta^*} \text{ is the mean of } p^b_t$$

$$\Sigma^a_{t,\theta^*} \text{ is the covariance of } p^a_t, \quad \Sigma^b_{t,\theta^*} \text{ is the covariance of } p^b_t$$

- Optimal matching of mean and covariance matrices

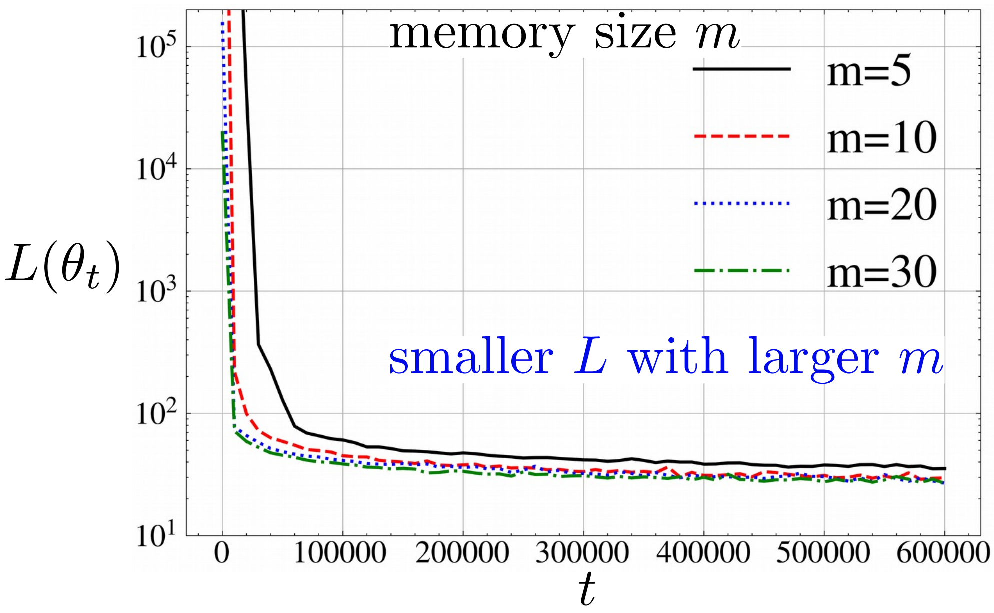
$\Rightarrow$ Potential to capture complex chaotic dynamics

- This includes Kalman filter as a special linear case

- What happens with finite memory size ?

Performance of DAN on Lorenz 95

- Train DAN on 1000 trajectories $\{x_t, y_t\}_{t \leq T}$ with 40 state variables
- Minimization of $L(\theta_t)$ by *truncated gradients* for $T = 6 \times 10^5$



Performance of DAN on Lorenz 95

- Given one trajectory $\{x_t, y_t\}_{t\leq T}$, compute the mean μ_t^a of q_{t,θ_T}^a and the mean μ_t^b of q_{t,θ_T}^b from $y_1, \cdots, y_t$, for $t \leq T$

Accuracy of the mean w.r.t memory size m

$$\frac{1}{T}\sum_{t=1}^{T}\|\mu_t^a - x_t\|$$

m	5	10	20	30
DAN	**0.401**	0.388	0.376	0.376
IEnKF-Q	3.939	2.798	0.413	0.355
LETKF	0.4647	0.3629	0.3460	0.3424
LETKF*	0.4092	**0.3610**	**0.3460**	**0.3418**

$$\frac{1}{T}\sum_{t=1}^{T}\|\mu_t^b - x_t\|$$

m	5	10	20	30
DAN	**0.453**	0.436	0.423	0.423
IEnKF-Q	4.021	2.920	0.460	0.399
LETKF	0.5171	0.4075	0.3890	0.3851
LETKF*	0.4565	**0.4047**	**0.3890**	**0.3846**

- DAN, IEnKF-Q, LETKF: tuned on $m = 20$. LETKF*: tuned on each m.

When $m < d = 40$, accuracy of DAN is comparable to ETKF methods

$\Rightarrow$ Replace tuning inflation/localization across m by learning

Stability analysis

- How sensitive is the performance w.r.t to the range of t?

$$\frac{1}{T}\sum_{t=1}^{T}\|\mu_t^a - x_t\|$$

m	5	10	20	30
DAN	**0.401**	**0.388**	**0.376**	0.376
IEnKF-Q	3.939	2.798	0.413	0.355

$$\frac{1}{T}\sum_{t=T+1}^{2T}\|\mu_t^a - x_t\|$$

m	5	10	20	30
DAN	0.400	0.388	0.377	0.376
IEnKF-Q	3.941	2.785	0.412	0.356

Similar accuracy

- How sensitive is the performance w.r.t to x_0?

$$\frac{1}{T}\sum_{t=1}^{T}\|\mu_t^a - x_t\|$$

burning time	10^1	10^3	10^5	10^7
DAN	0.376	0.376	0.377	0.377
IEnKF-Q	0.414	0.413	0.414	0.413

Similar accuracy

Conclusions and perspectives

- ETKF methods with Gaussian assumps are sub-optimal.

- DAN can in theory achieve optimal estimations of state probabilites by optimizing likelihood-based objective function.

- Numerically DAN, when trained with Gaussian pdfs, can achieve comparable performance to ETKFs on Lorenz-95, without inflation/localization tuning across memory size.

- Future: extension of DAN to non-Gaussian pdfs, and to self-supervised/unsupervised learning setups.