Deep Learning for Smoothing in Dynamical Systems

Hannes Larsson, Songchen Li

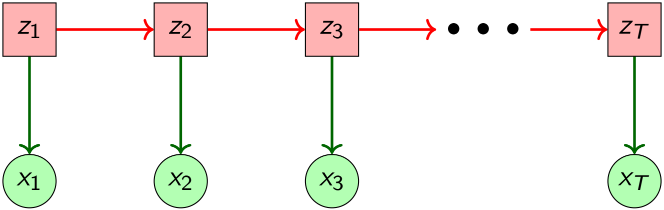
UPPSALA UNIVERSITET

2018-01-09

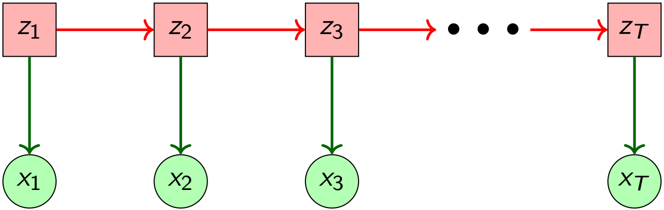
Outline

- Many dynamical systems can be described by a *state space model*:

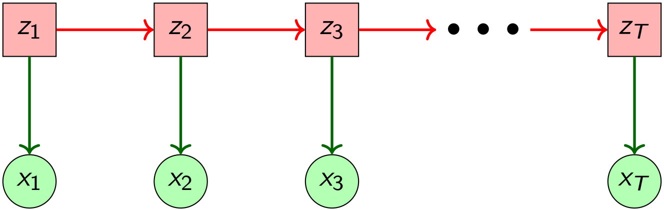
- Many dynamical systems can be described by a *state space model*:



Dynamical Systems and State Space Models



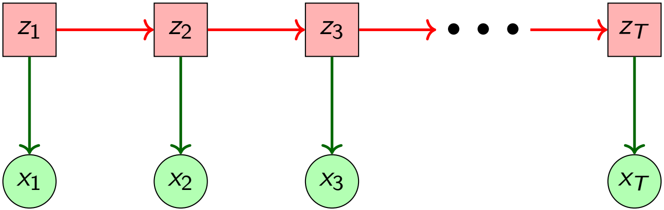
$$\begin{cases} z_{t+1} = f(z_t) + v_{1_t} \\ x_t = g(z_t) + v_{2_t}, \end{cases} \iff \begin{cases} Z_{t+1}|z_t \sim p(z_{t+1}|z_t) \\ X_t|z_t \sim p(x_t|z_t) \end{cases}$$



$$\begin{cases} z_{t+1} = f(z_t) + v_{1_t} \\ x_t = g(z_t) + v_{2_t}, \end{cases} \iff \begin{cases} Z_{t+1} | z_t \sim p(z_{t+1} | z_t) \\ X_t | z_t \sim p(x_t | z_t) \end{cases}$$

- $z_1, z_2, ..., z_T$ are *hidden states*.
- $x_1, x_2, ..., x_T$ are the *measured* quantities.

Dynamical Systems and State Space Models



$$\begin{cases} z_{t+1} = f(z_t) + v_{1_t} \\ x_t = g(z_t) + v_{2_t}, \end{cases} \iff \begin{cases} Z_{t+1}|z_t \sim p(z_{t+1}|z_t) \\ X_t|z_t \sim p(x_t|z_t) \end{cases}$$

- $z_1, z_2, ..., z_T$ are *hidden states*.
- $x_1, x_2, ..., x_T$ are the *measured* quantities.
- v_1 and v_2 are noise terms.

- Usually we want to estimate z given some observations x

Filtering and Smoothing

- Usually we want to estimate z given some observations x
- Filtering - online problem

Filtering and Smoothing

- Usually we want to estimate z given some observations x
- Filtering - online problem
- Smoothing - offline problem

Filtering and Smoothing

- Usually we want to estimate z given some observations x
- Filtering - online problem
- Smoothing - offline problem
- if f and g linear and v_1, v_2 Gaussian white noise the optimal solution is the *Kalman filter/smoother*

Smoothing

- Offline problem - finding all z as a function of x

$$p(\mathbf{z}|\mathbf{x}) = \frac{p(\mathbf{x}, \mathbf{z})}{p(\mathbf{x})}$$

Smoothing

- Offline problem - finding all z as a function of x

$$p(\mathbf{z}|\mathbf{x}) = \frac{p(\mathbf{x}, \mathbf{z})}{p(\mathbf{x})}$$

- $\mathbf{x} = (x_1, x_2, ..., x_T)$
- $\mathbf{z} = (z_1, z_2, ..., z_T)$

$$p(\mathbf{x}, \mathbf{z}) = p(z_0)\Big[\prod_{t=1}^{T} p(z_t|z_{t-1})\Big]\Big[\prod_{t=1}^{T} p(x_t|z_t)\Big]$$

Variational Inference

- Variational inference

- Variational inference

$$p(\mathbf{z}|\mathbf{x}) = \frac{p(\mathbf{x}, \mathbf{z})}{p(\mathbf{x})}$$
$$p(\mathbf{x}) = \int p(\mathbf{x}, \mathbf{z}) d\mathbf{z}$$

- Variational inference

$$p(\mathbf{z}|\mathbf{x}) = \frac{p(\mathbf{x}, \mathbf{z})}{p(\mathbf{x})}$$
$$p(\mathbf{x}) = \int p(\mathbf{x}, \mathbf{z}) d\mathbf{z}$$

- $p(\mathbf{x})$ is hard to calculate

Variational Inference

- Variational inference

$$p(\mathbf{z}|\mathbf{x}) = \frac{p(\mathbf{x}, \mathbf{z})}{p(\mathbf{x})}$$
$$p(\mathbf{x}) = \int p(\mathbf{x}, \mathbf{z}) d\mathbf{z}$$

- $p(\mathbf{x})$ is hard to calculate
- Idea: use approximate posterior $q_\theta(\mathbf{z}|\mathbf{x}) \approx p(\mathbf{z}|\mathbf{x})$

Variational Inference

- Chose $q_\theta(\mathbf{x}|\mathbf{z})$ s.t. it maximises the Evidence Lower Bound (ELBO)

$$\mathcal{L} = \mathbb{E}_{q_\theta}[\log p(\mathbf{x}, \mathbf{z})] - \mathbb{E}_{q_\theta}[\log q_\theta(\mathbf{z}|\mathbf{x})]$$

- Chose $q_\theta(\mathbf{x}|\mathbf{z})$ s.t. it maximises the Evidence Lower Bound (ELBO)

$$\mathcal{L} = \mathbb{E}_{q_\theta}[\log p(\mathbf{x}, \mathbf{z})] - \mathbb{E}_{q_\theta}[\log q_\theta(\mathbf{z}|\mathbf{x})]$$

- This is equivalent to minimising the Kullback–Leibler divergence

$$D_{KL}(q||p) = \int_\Omega p(\omega) \log \frac{p(\omega)}{q(\omega)} d\omega$$

- Still hard to optimise the ELBO

- Still hard to optimise the ELBO
- We need the gradient which involves a derivative of an expected value w.r.t. the approximate posterior

$$\mathcal{L} = \mathbb{E}_{q_\theta}[\log p(\mathbf{x}, \mathbf{z}) - \log q_\theta(\mathbf{z}|\mathbf{x})] =: \mathbb{E}_{q_\theta}[f(\mathbf{x}, \mathbf{z})]$$

Black Box Variational Inference

- Still hard to optimise the ELBO
- We need the gradient which involves a derivative of an expected value w.r.t. the approximate posterior

$$\mathcal{L} = \mathbb{E}_{q_\theta}[\log p(\mathbf{x}, \mathbf{z}) - \log q_\theta(\mathbf{z}|\mathbf{x})] =: \mathbb{E}_{q_\theta}[f(\mathbf{x}, \mathbf{z})]$$

- Idea: Sample z from q_θ and calculate noisy gradient

Black Box Variational Inference: Noisy Gradient

UPPSALA UNIVERSITET

- Let $q_\theta(z_t|x_t, ..., x_T) = \mathcal{N}(\mu_t, \sigma_t^2)$

$$z_t = \mu_t + \sigma_t \epsilon =: g_\theta(\epsilon)$$
$$\epsilon \sim \mathcal{N}(0, 1) =: p(\epsilon)$$

Black Box Variational Inference: Noisy Gradient

- Let $q_\theta(z_t | x_t, ..., x_T) = \mathcal{N}(\mu_t, \sigma_t^2)$

$$z_t = \mu_t + \sigma_t \epsilon =: g_\theta(\epsilon)$$
$$\epsilon \sim \mathcal{N}(0, 1) =: p(\epsilon)$$

- Approximate
$\nabla_\theta \mathcal{L} = \mathbb{E}_{p(\epsilon)}[\nabla f_\theta(x, g_\theta(\epsilon))] \approx \frac{1}{N} \sum_{i=1}^{N} \nabla f_\theta(x, g_\theta(\epsilon))$

Black Box Variational Inference: Noisy Gradient

UPPSALA UNIVERSITET

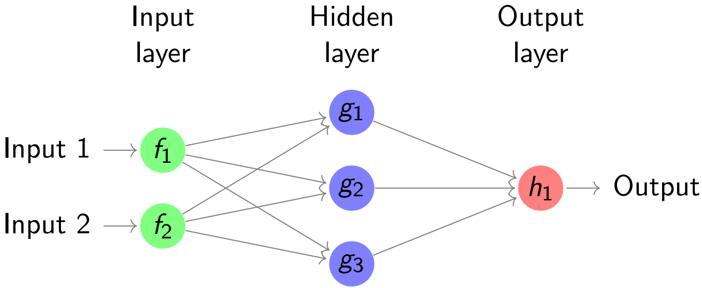
- Let $q_\theta(z_t | x_t, ..., x_T) = \mathcal{N}(\mu_t, \sigma_t^2)$

$$z_t = \mu_t + \sigma_t \epsilon =: g_\theta(\epsilon)$$
$$\epsilon \sim \mathcal{N}(0, 1) =: p(\epsilon)$$

- Approximate $\nabla_\theta \mathcal{L} = \mathbb{E}_{p(\epsilon)}[\nabla f_\theta(x, g_\theta(\epsilon))] \approx \frac{1}{N} \sum_{i=1}^{N} \nabla f_\theta(x, g_\theta(\epsilon))$
- This is called the *reparametrisation trick*

- A Neural Network is a function of some input variables.

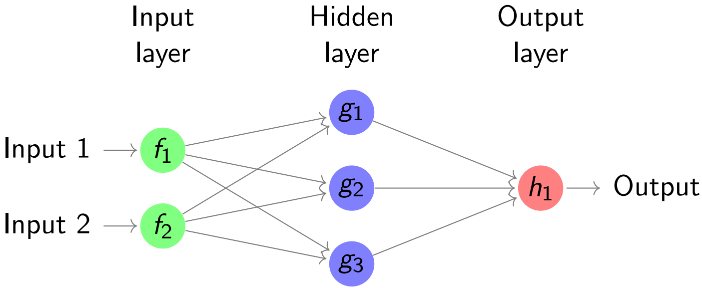
- A Neural Network is a function of some input variables.



- A neuron: $g_i(\mathbf{x}) = K(\sum_t \omega_t f_t(\mathbf{x}) + b)$

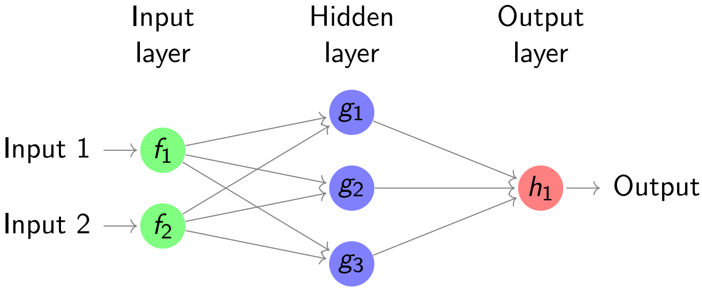
Neural Networks

- A Neural Network is a function of some input variables.



- A neuron: $g_i(\mathbf{x}) = K(\sum_t \omega_t f_t(\mathbf{x}) + b)$
- b is called a bias and ω is called a weight, K is some non-linear function, called an activation function.

- A Neural Network is a function of some input variables.



- A neuron: $g_i(\mathbf{x}) = K(\sum_t \omega_t f_t(\mathbf{x}) + b)$
- b is called a bias and ω is called a weight, K is some non-linear function, called an activation function.
- Typically some loss function is minimised w.r.t. the weights and biases.

Summary so far

Summary so far

UPPSALA UNIVERSITET

- State Space Models

Summary so far

- State Space Models
- Smoothing

Summary so far

UPPSALA UNIVERSITET

- State Space Models
- Smoothing
- Variational Inference

Summary so far

- State Space Models
- Smoothing
- Variational Inference
- Black Box Variational Inference

Summary so far

UPPSALA UNIVERSITET

- State Space Models
- Smoothing
- Variational Inference
- Black Box Variational Inference
- Neural Networks

Implementation

UPPSALA UNIVERSITET

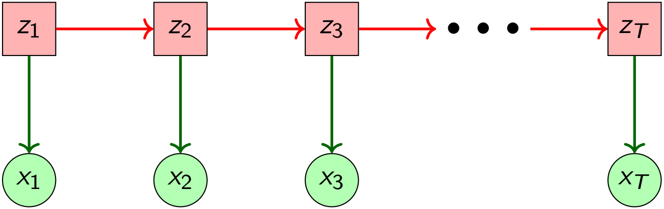
- All of the methods here were implemented with Python and TensorFlow.
- TensorFlow is an open source software library commonly used in deep learning.

BBVI for Smoothing Using RNN

- RNN: Recurrent neural network, a kind of NN that uses the information from different time steps

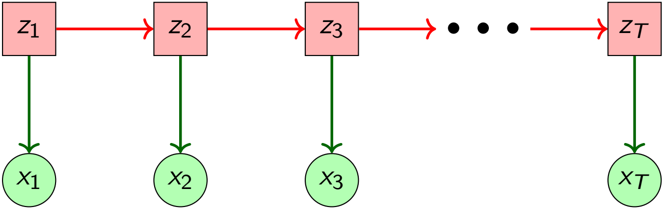
BBVI for Smoothing Using RNN

- RNN: Recurrent neural network, a kind of NN that uses the information from different time steps
- Smoothing: Knowing all the measurements $\mathbf{x}$, try to find $p(\mathbf{z}|\mathbf{x})$



BBVI for Smoothing Using RNN

- RNN: Recurrent neural network, a kind of NN that uses the information from different time steps
- Smoothing: Knowing all the measurements $\mathbf{x}$, try to find $p(\mathbf{z}|\mathbf{x})$

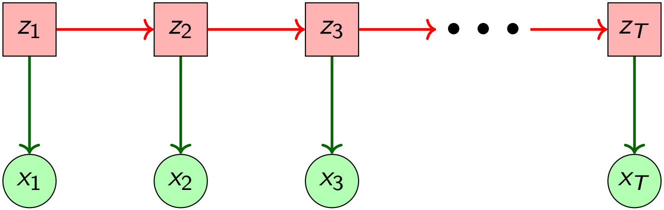


- BBVI: Black box variational inference

$$q_\theta(\mathbf{z}|\mathbf{x}) \xrightarrow{\text{Optimize } \theta} \text{Maximize ELBO} \longrightarrow q_\theta(\mathbf{z}|\mathbf{x}) \approx p(\mathbf{z}|\mathbf{x})$$

BBVI for Smoothing Using RNN

- Factorize $q_\theta(\mathbf{z}|\mathbf{x})$



$$\begin{cases} q_\theta(\mathbf{z}|\mathbf{x}) = q_\theta(z_1|x_1, \dots, x_T) \prod_{t=2}^{T} q_\theta(z_t|z_{t-1}, x_t, \dots, x_T) \\ q_\theta(z_t|z_{t-1}, x_t, \dots, x_T) \sim \mathcal{N}(\mu_t(z_{t-1}, x_t, \dots, x_T), \sigma_t^2(z_{t-1}, x_t, \dots, x_T)) \end{cases}$$

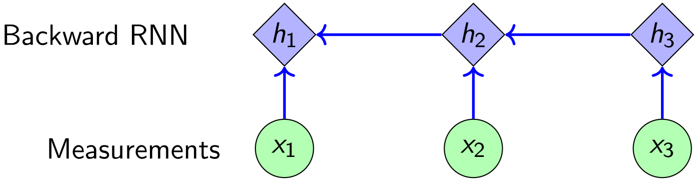
BBVI for Smoothing Using RNN

UPPSALA UNIVERSITET

$$q_\theta(\mathbf{z}|\mathbf{x}) = q_\theta(z_1|x_1, \ldots, x_T) \prod_{t=2}^{T} q_\theta(z_t|z_{t-1}, x_t, \ldots, x_T)$$

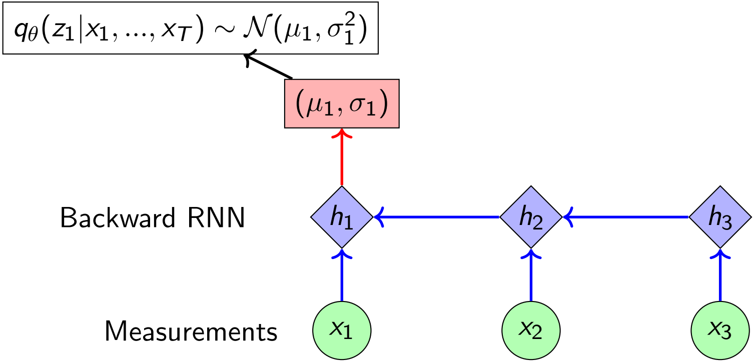
BBVI for Smoothing Using RNN

$$q_\theta(\mathbf{z}|\mathbf{x}) = q_\theta(z_1|x_1, ..., x_T) \prod_{t=2}^{T} q_\theta(z_t|z_{t-1}, x_t, ..., x_T)$$



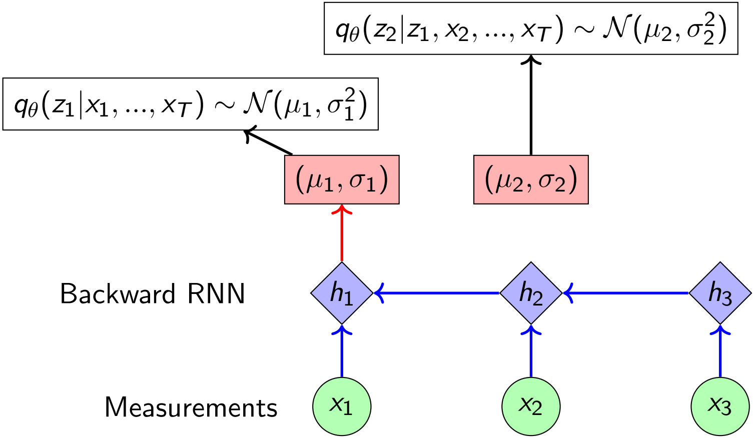
BBVI for Smoothing Using RNN

$$q_\theta(\mathbf{z}|\mathbf{x}) = q_\theta(z_1|x_1, \dots, x_T) \prod_{t=2}^{T} q_\theta(z_t|z_{t-1}, x_t, \dots, x_T)$$



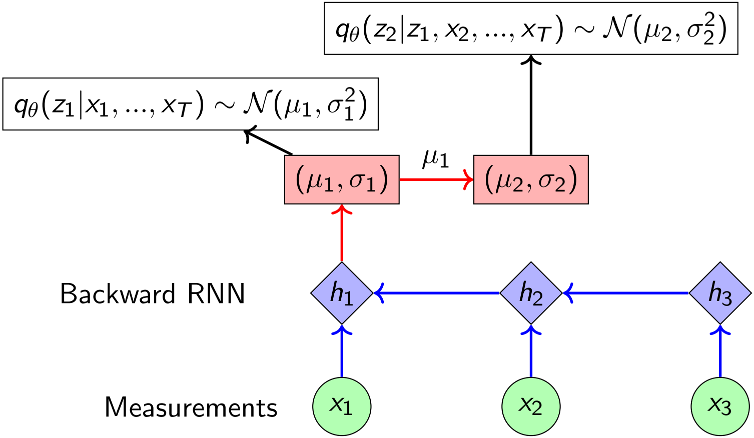
BBVI for Smoothing Using RNN

$$q_\theta(\mathbf{z}|\mathbf{x}) = q_\theta(z_1|x_1, ..., x_T) \prod_{t=2}^{T} q_\theta(z_t|z_{t-1}, x_t, ..., x_T)$$



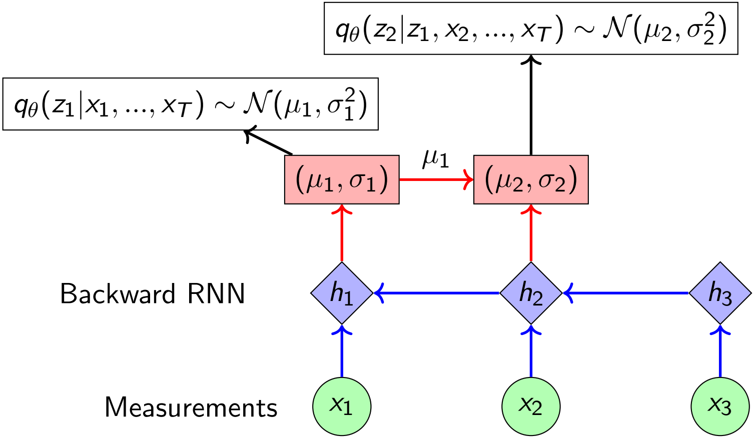
BBVI for Smoothing Using RNN

$$q_\theta(\mathbf{z}|\mathbf{x}) = q_\theta(z_1|x_1, \ldots, x_T) \prod_{t=2}^{T} q_\theta(z_t|z_{t-1}, x_t, \ldots, x_T)$$



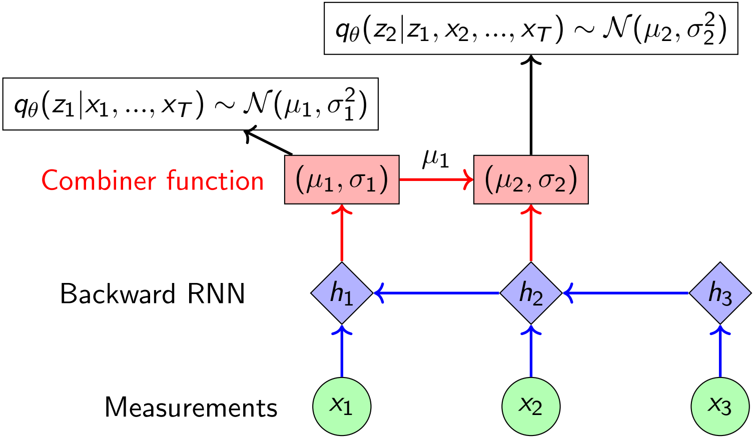
BBVI for Smoothing Using RNN

$$q_\theta(\mathbf{z}|\mathbf{x}) = q_\theta(z_1|x_1, \dots, x_T) \prod_{t=2}^{T} q_\theta(z_t|z_{t-1}, x_t, \dots, x_T)$$



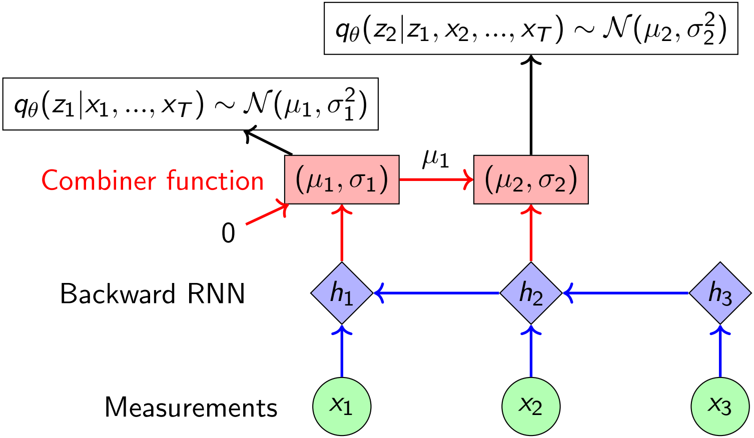
BBVI for Smoothing Using RNN

$$q_\theta(\mathbf{z}|\mathbf{x}) = q_\theta(z_1|x_1, \ldots, x_T) \prod_{t=2}^{T} q_\theta(z_t|z_{t-1}, x_t, \ldots, x_T)$$



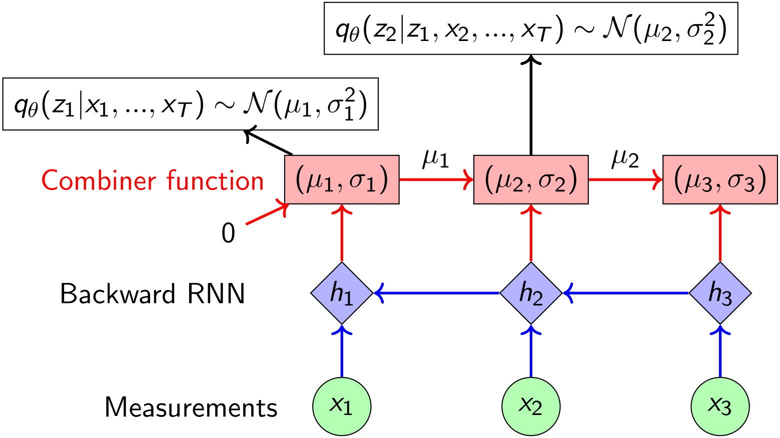
BBVI for Smoothing Using RNN

$$q_\theta(\mathbf{z}|\mathbf{x}) = q_\theta(z_1|x_1, ..., x_T) \prod_{t=2}^{T} q_\theta(z_t|z_{t-1}, x_t, ..., x_T)$$



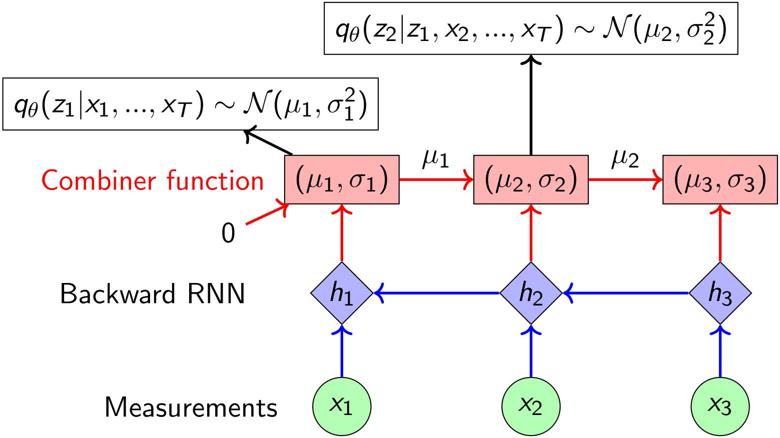
BBVI for Smoothing Using RNN

$$q_\theta(\mathbf{z}|\mathbf{x}) = q_\theta(z_1|x_1,\ldots,x_T)\prod_{t=2}^{T} q_\theta(z_t|z_{t-1},x_t,\ldots,x_T)$$



BBVI for Smoothing Using RNN

$$q_\theta(\mathbf{z}|\mathbf{x}) = q_\theta(z_1|x_1, \ldots, x_T) \prod_{t=2}^{T} q_\theta(z_t|z_{t-1}, x_t, \ldots, x_T)$$



- Output: $\boldsymbol{\mu} = \mu_1, \mu_2, \ldots, \mu_T \quad \boldsymbol{\sigma} = \sigma_1, \sigma_2, \ldots, \sigma_T$

BBVI for Smoothing Using RNN

Train the RNN

- Maximising the objective function: ELBO

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\mu}, \boldsymbol{\sigma}) = \mathcal{L}(\mathbf{x}, \theta)$$

UPPSALA UNIVERSITET

Train the RNN

- Maximising the objective function: ELBO

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\mu}, \boldsymbol{\sigma}) = \mathcal{L}(\mathbf{x}, \theta)$$

- Use the noisy gradients to update θ

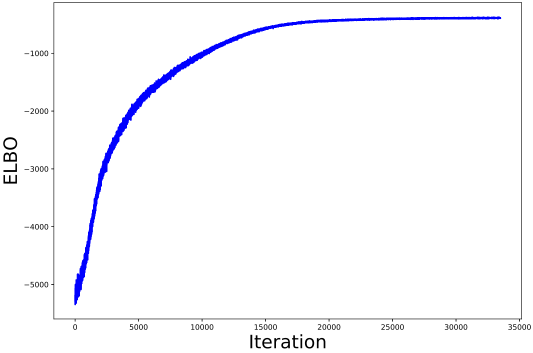
UPPSALA UNIVERSITET

Train the RNN

- Maximising the objective function: ELBO

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\mu}, \boldsymbol{\sigma}) = \mathcal{L}(\mathbf{x}, \theta)$$

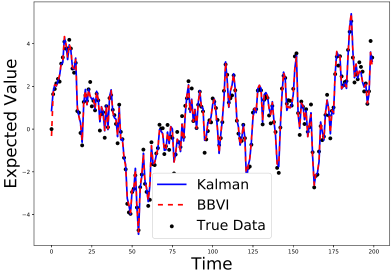
- Use the noisy gradients to update θ



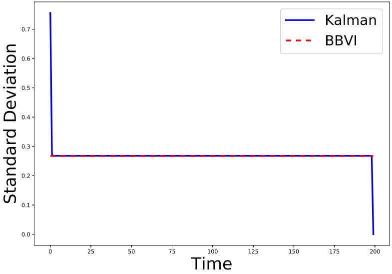
BBVI for Smoothing Using RNN

UPPSALA UNIVERSITET

Results



(a) Expected Value



(b) Standard Deviation

Conclusions

UPPSALA UNIVERSITET

- Our BBVI using a RNN recovers the full posterior probability density function obtained by the Kalman smoother.

Conclusions

- Our BBVI using a RNN recovers the full posterior probability density function obtained by the Kalman smoother.
- BBVI using RNN is more flexible than the Kalman smoother, because it can also be used for non-linear SSM with noise that is not Gaussian. All we need to do is to rewrite the ELBO according to the new model.

Thank you for listening!

UPPSALA
UNIVERSITET

Any questions?