Generative Models - I

Energy Based Models, Variational Autoencoders

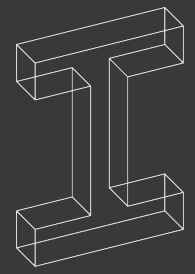
O. Deniz Akyildiz

StatML CDT Course

22 April 2024

IMPERIAL

Today's Plan



- Energy Based Models

  Introduction, Score Matching, Algorithms
- Variational Autoencoders

  Introduction, Basics, Theory
- Advanced Topics and Connections

Part I: Energy Based Models

Part I: Energy Based Models

- An Introduction to Sampling

Part I: Energy Based Models

- An Introduction to Sampling
- Energy-Based Models

Part I: Energy Based Models

- An Introduction to Sampling
- Energy-Based Models
- Maximum Likelihood, Score Matching etc.

Part I: Energy Based Models

- An Introduction to Sampling
- Energy-Based Models
- Maximum Likelihood, Score Matching etc.
- Applications and Examples

An Introduction to Sampling

What is sampling, what is generative modelling?

Sampling: Given a probability measure π, output (approximately) a sample $X \sim \pi$.

An Introduction to Sampling

What is sampling, what is generative modelling?

Sampling: Given a probability measure π, output (approximately) a sample $X \sim \pi$.

- Typically $\pi \propto \exp(-U(x))$ where $U(x)$ is a potential (or energy) function.

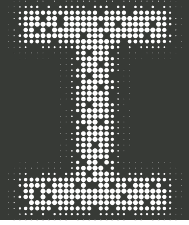
Sampling: Given a probability measure π, output (approximately) a sample $X \sim \pi$.

- Typically $\pi \propto \exp(-U(x))$ where $U(x)$ is a potential (or energy) function.

Generative modelling: Given a dataset $\{x_i\}_{i=1}^n$ or given an empirical measure

$$\hat{p}_{\text{data}} = \frac{1}{n} \sum_{i=1}^{n} \delta_{x_i},$$

output (approximately) a sample $X \sim p_{\text{data}}$.



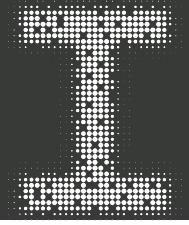
Sampling is a very general problem and a very old topic in statistics and computer science. One of the most popular methods is Markov chain Monte Carlo (MCMC).

Sampling is a very general problem and a very old topic in statistics and computer science. One of the most popular methods is Markov chain Monte Carlo (MCMC).

- Scales favourably with dimension

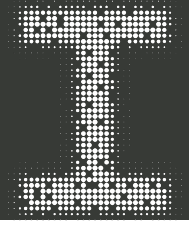
An Introduction to Sampling

Markov chain Monte Carlo



Sampling is a very general problem and a very old topic in statistics and computer science. One of the most popular methods is Markov chain Monte Carlo (MCMC).

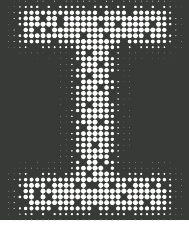
- Scales favourably with dimension
- Requires only the ability to evaluate the density up to a normalizing constant



Sampling is a very general problem and a very old topic in statistics and computer science. One of the most popular methods is Markov chain Monte Carlo (MCMC).

- Scales favourably with dimension
- Requires only the ability to evaluate the density up to a normalizing constant

The principle is to construct a Markov chain with stationary distribution π and sample from it.



The black-box way of doing it is to use Metropolis-Hastings.

The black-box way of doing it is to use Metropolis-Hastings.

- Sample $X' \sim q(x'|X_{n-1})$
- Set $X_n = X'$ with probability

$$\alpha(X'|X_{n-1}) = \min\left\{1, \frac{\Pi(X')q(X_{n-1}|X')}{\Pi(X_{n-1})q(X'|X_{n-1})}\right\}.$$

- Otherwise, set $X_n = X_{n-1}$.

where $\Pi(x) = \exp(-U(x))$.

The black-box way of doing it is to use Metropolis-Hastings.

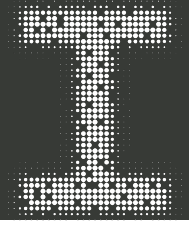
- Sample $X' \sim q(x'|X_{n-1})$
- Set $X_n = X'$ with probability

$$\alpha(X'|X_{n-1}) = \min\left\{1, \frac{\Pi(X')q(X_{n-1}|X')}{\Pi(X_{n-1})q(X'|X_{n-1})}\right\}.$$

- Otherwise, set $X_n = X_{n-1}$.

where $\Pi(x) = \exp(-U(x))$.

This leaves π invariant.

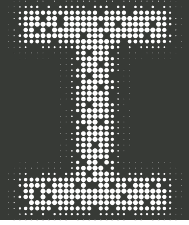


The choice of the proposal is crucial. One very popular choice is the Langevin proposal.

The choice of the proposal is crucial. One very popular choice is the Langevin proposal.

$$X' = X_{n-1} - \gamma \nabla U(X_{n-1}) + \sqrt{2\gamma} Z_n$$

where $Z_n \sim \mathcal{N}(0, I)$ and γ is the step size.

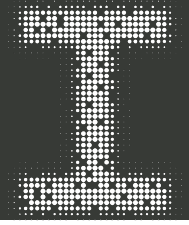


The choice of the proposal is crucial. One very popular choice is the Langevin proposal.

$$X' = X_{n-1} - \gamma \nabla U(X_{n-1}) + \sqrt{2\gamma} Z_n$$

where $Z_n \sim \mathcal{N}(0, I)$ and γ is the step size. This results in

$$q(x'|x) = \mathcal{N}(x - \gamma \nabla U(x), 2\gamma I).$$



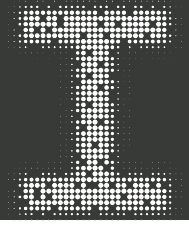
The choice of the proposal is crucial. One very popular choice is the Langevin proposal.

$$X' = X_{n-1} - \gamma \nabla U(X_{n-1}) + \sqrt{2\gamma} Z_n$$

where $Z_n \sim \mathcal{N}(0, I)$ and γ is the step size. This results in

$$q(x'|x) = \mathcal{N}(x - \gamma \nabla U(x), 2\gamma I).$$

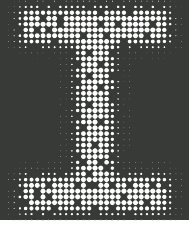
Why is this a good choice?



Consider the Langevin SDE for a generic drift ∇E:

$$\mathrm{d}X_t = -\nabla U(X_t)\mathrm{d}t + \sqrt{2}\mathrm{d}B_t,$$

where $(B_t)_{t\geq 0}$ is a Brownian motion.



Consider the Langevin SDE for a generic drift ∇E:

$$\mathrm{d}X_t = -\nabla U(X_t)\mathrm{d}t + \sqrt{2}\mathrm{d}B_t,$$

where $(B_t)_{t\geq 0}$ is a Brownian motion. This SDE has a (Gibbs-type) stationary measure

$$\pi \propto e^{-U(x)}.$$

Therefore, for a classical *sampling* problem for, say $\pi(x)$, we could set $U(x) = -\log \pi(x)$ (negative log-density).

Consider the Langevin SDE for a generic drift ∇E:

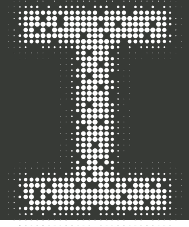
$$\mathrm{d}X_t = -\nabla U(X_t)\mathrm{d}t + \sqrt{2}\mathrm{d}B_t,$$

where $(B_t)_{t\geq 0}$ is a Brownian motion. This SDE has a (Gibbs-type) stationary measure

$$\pi \propto e^{-U(x)}.$$

Therefore, for a classical *sampling* problem for, say $\pi(x)$, we could set $U(x) = -\log \pi(x)$ (negative log-density).

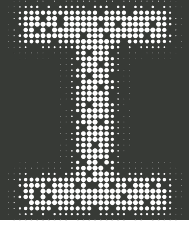
This diffusion converges to its stationary measure exponentially fast if E is μ-strongly-convex.



Consider the Langevin SDE for a generic drift ∇E:

$$\mathrm{d}X_t = -\nabla U(X_t)\mathrm{d}t + \sqrt{\frac{2}{\beta}}\mathrm{d}B_t,$$

where $(B_t)_{t\geq 0}$ is a Brownian motion.

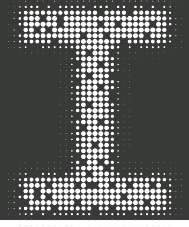


Consider the Langevin SDE for a generic drift ∇E:

$$\mathrm{d}X_t = -\nabla U(X_t)\mathrm{d}t + \sqrt{\frac{2}{\beta}}\mathrm{d}B_t,$$

where $(B_t)_{t\geq 0}$ is a Brownian motion. This SDE has a stationary measure

$$\pi \propto e^{-\beta U(x)}.$$



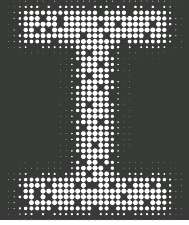
Consider the Langevin SDE for a generic drift ∇E:

$$\mathrm{d}X_t = -\nabla U(X_t)\mathrm{d}t + \sqrt{\frac{2}{\beta}}\mathrm{d}B_t,$$

where $(B_t)_{t\geq 0}$ is a Brownian motion. This SDE has a stationary measure

$$\pi \propto e^{-\beta U(x)}.$$

This stationary measure concentrates on the minima of E as $\beta \to \infty$ (Hwang, 1980).



Consider the Langevin SDE for a generic drift ∇E:

$$\mathrm{d}X_t = -\nabla U(X_t)\mathrm{d}t + \sqrt{\frac{2}{\beta}}\mathrm{d}B_t,$$

where $(B_t)_{t\geq 0}$ is a Brownian motion. This SDE has a stationary measure

$$\pi \propto e^{-\beta U(x)}.$$

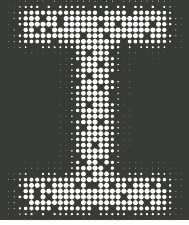
This stationary measure concentrates on the minima of E as $\beta \to \infty$ (Hwang, 1980).

Langevin diffusion is a global optimiser.

The Euler discretisation is the *unadjusted Langevin algorithm* (ULA):

$$X_{n+1}^{\gamma} = X_n^{\gamma} - \gamma \nabla U(X_n^{\gamma}) + \sqrt{2\gamma} W_{n+1}$$

where $(W_n)_{n \geq 0}$ are i.i.d standard Normal random variables.



The Euler discretisation is the *unadjusted Langevin algorithm* (ULA):

$$X_{n+1}^{\gamma} = X_n^{\gamma} - \gamma \nabla U(X_n^{\gamma}) + \sqrt{2\gamma} W_{n+1}$$

where $(W_n)_{n \geq 0}$ are i.i.d standard Normal random variables.

This chain has a *different* stationary measure π_γ but a number of guarantees can be derived for its convergence.

Theorem 1 (Durmus and Moulines, 2019)

Let $\mathcal{L}(X_t)$ be the law of the iterates of ULA, then

$$W_2^2(\mathcal{L}(X_n^{\gamma}), \pi) \lesssim \left(1 - \frac{\gamma\kappa}{2}\right)^{n+1} (d/m + \|x_0 - x^\star\|^2) + \gamma,$$

under suitable regularity conditions for V, restriction on γ where $\kappa := \kappa(m, L)$.

As we discussed initially, in generative modelling we do not know the density π but we have access to samples:

$$\hat{p}_{\text{data}} = \frac{1}{n} \sum_{i=1}^{n} \delta_{x_i}.$$

As we discussed initially, in generative modelling we do not know the density π but we have access to samples:

$$\hat{p}_{\text{data}} = \frac{1}{n} \sum_{i=1}^{n} \delta_{x_i}.$$

To sample from $\pi := p_{\text{data}}$, we need to learn a model to be able to use our favourite sampling schemes.

As we discussed initially, in generative modelling we do not know the density π but we have access to samples:

$$\hat{p}_{\text{data}} = \frac{1}{n} \sum_{i=1}^{n} \delta_{x_i}.$$

To sample from $\pi := p_{\text{data}}$, we need to learn a model to be able to use our favourite sampling schemes.

Next: Energy Based Models

Energy Based Models

Introduction

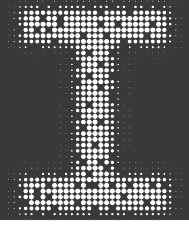
Given $\hat{p}_{\text{data}}$, we would like to estimate the density p_{data}.

Energy Based Models

Introduction

Given $\hat{p}_{\text{data}}$, we would like to estimate the density p_{data}.

- This is a classical density estimation problem.



Given $\hat{p}_{\text{data}}$, we would like to estimate the density p_{data}.

- This is a classical density estimation problem.

Discussion: Can we use kernel density estimation for this problem?

Energy Based Models

Introduction

Given $\hat{p}_{\text{data}}$, we would like to estimate the density p_{data}.

- This is a classical density estimation problem.

Discussion: Can we use kernel density estimation for this problem?

What is a sensible parametric model?

Consider the Gibbs-type probability measure

$$\pi_\theta \propto \exp(-U_\theta(x)),$$

where $U_\theta(x)$ is a potential function.

Energy Based Models

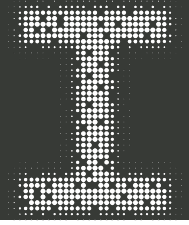
Introduction

Consider the Gibbs-type probability measure

$$\pi_\theta \propto \exp(-U_\theta(x)),$$

where $U_\theta(x)$ is a potential function.

When is this a flexible model for p_{data}?



Recall that if $U_\theta(x)$ is a neural network, then $U_\theta(x)$ is a universal approximator.

Recall that if $U_\theta(x)$ is a neural network, then $U_\theta(x)$ is a universal approximator.

For any continuous function $U(x)$, there exists a neural network $U_\theta(x)$ such that

$$\|U(x) - U_\theta(x)\|_\infty < \epsilon.$$

Recall that if $U_\theta(x)$ is a neural network, then $U_\theta(x)$ is a universal approximator.

For any continuous function $U(x)$, there exists a neural network $U_\theta(x)$ such that

$$\|U(x) - U_\theta(x)\|_\infty < \epsilon.$$

Let $U : \mathrm{X} \to \mathbb{R}$ and $U_\theta : \mathrm{X} \to \mathbb{R}$ be a neural network. Let $\pi \propto \exp(-U(x))$ and $\pi_\theta \propto \exp(-U_\theta(x))$.

Let $U : \mathrm{X} \to \mathbb{R}$ and $U_\theta : \mathrm{X} \to \mathbb{R}$ be a neural network. Let $\pi \propto \exp(-U(x))$ and $\pi_\theta \propto \exp(-U_\theta(x))$. Then (Atchadé et al., 2023)

$$\|\pi - \pi_\theta\|_{\mathrm{TV}} \leq \frac{m(\mathrm{X})}{2} \|U - U_\theta\|_\infty$$

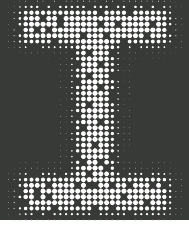
where m is the Lebesgue measure on X (with finite measure).

Let $U : \mathrm{X} \to \mathbb{R}$ and $U_\theta : \mathrm{X} \to \mathbb{R}$ be a neural network. Let $\pi \propto \exp(-U(x))$ and $\pi_\theta \propto \exp(-U_\theta(x))$. Then (Atchadé et al., 2023)

$$\|\pi - \pi_\theta\|_{\mathrm{TV}} \leq \frac{m(\mathrm{X})}{2} \|U - U_\theta\|_\infty$$

where m is the Lebesgue measure on X (with finite measure).

Therefore, an EBM is a universal approximator on bounded spaces.



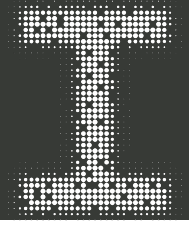
Training EBMs is a hard problem and there is a VAST literature on this topic.

Training EBMs is a hard problem and there is a VAST literature on this topic.

We will review (expanding):

- Maximum Likelihood Estimation
- Score Matching

and variations of these.

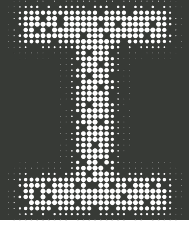


The idea is to maximise the expected likelihood of the data under the model. The *expected* log-likelihood of the data is given by

$$\ell(\theta) := \mathbb{E}_{p_{\text{data}}}[\log \pi_\theta(X)], \tag{1}$$

where p_{data} is the data distribution.

Mini-quiz: Prove that maximising this is equivalent to minimising the KL divergence between p_{data} and π_θ.



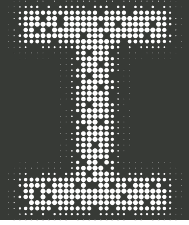
The idea is to maximise the expected likelihood of the data under the model. The *expected* log-likelihood of the data is given by

$$\ell(\theta) := \mathbb{E}_{p_{\text{data}}}[\log \pi_\theta(X)], \tag{1}$$

where p_{data} is the data distribution.

Mini-quiz: Prove that maximising this is equivalent to minimising the KL divergence between p_{data} and π_θ.

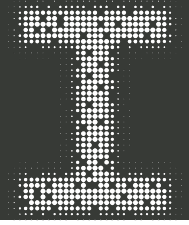
Proof on the board.



Let us look at the objective a bit more closely.

$$\ell(\theta) = \mathbb{E}_{p_{\text{data}}}[\log \pi_\theta(X)] = \int \log \pi_\theta(x) p_{\text{data}}(x) \mathrm{d}x,$$
$$= -\int U_\theta(x) p_{\text{data}}(x) \mathrm{d}x - \log Z_\theta,$$

where $Z_\theta = \int \exp(-U_\theta(x)) \mathrm{d}x$ is the normalising constant.



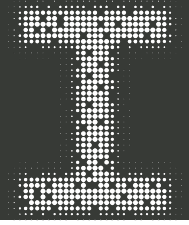
Let us look at the objective a bit more closely.

$$\ell(\theta) = \mathbb{E}_{p_{\text{data}}}[\log \pi_\theta(X)] = \int \log \pi_\theta(x) p_{\text{data}}(x) \mathrm{d}x,$$
$$= -\int U_\theta(x) p_{\text{data}}(x) \mathrm{d}x - \log Z_\theta,$$

where $Z_\theta = \int \exp(-U_\theta(x)) \mathrm{d}x$ is the normalising constant. For gradient based optimisation, we can consider

$$\nabla_\theta \ell(\theta) = -\mathbb{E}_{p_{\text{data}}}\left[\nabla_\theta U_\theta(X)\right] - \nabla_\theta \log Z_\theta.$$

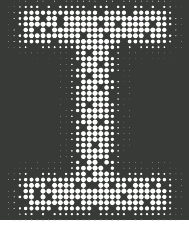
The last term is intractable.



Assuming $\int e^{-U_\theta(x)} \mathrm{d}x < \infty$, we have

$$\nabla_\theta \log Z_\theta = -\mathbb{E}_{\pi_\theta}[\nabla_\theta U_\theta(X)].$$

Mini-quiz: Prove this.

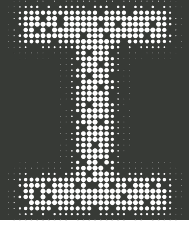


Assuming $\int e^{-U_\theta(x)} \mathrm{d}x < \infty$, we have

$$\nabla_\theta \log Z_\theta = -\mathbb{E}_{\pi_\theta}[\nabla_\theta U_\theta(X)].$$

Mini-quiz: Prove this.

Proof on the board.



Therefore, we have the full gradient that is of the form

$$\nabla_\theta \ell(\theta) = -\mathbb{E}_{p_{\text{data}}}\left[\nabla_\theta U_\theta(X)\right] + \mathbb{E}_{\pi_\theta}\left[\nabla_\theta U_\theta(X)\right].$$

Therefore, we have the full gradient that is of the form

$$\nabla_\theta \ell(\theta) = -\mathbb{E}_{p_{\text{data}}}\left[\nabla_\theta U_\theta(X)\right] + \mathbb{E}_{\pi_\theta}\left[\nabla_\theta U_\theta(X)\right].$$

We get then its empirical approximation

$$\nabla_\theta \ell_n(\theta) = -\frac{1}{n}\sum_{i=1}^{n} \nabla_\theta U_\theta(x_i) + \mathbb{E}_{\pi_\theta}\left[\nabla_\theta U_\theta(X)\right].$$

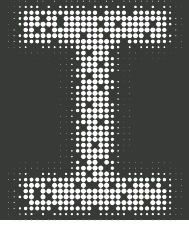
Therefore, we have the full gradient that is of the form

$$\nabla_\theta \ell(\theta) = -\mathbb{E}_{p_{\text{data}}}\left[\nabla_\theta U_\theta(X)\right] + \mathbb{E}_{\pi_\theta}\left[\nabla_\theta U_\theta(X)\right].$$

We get then its empirical approximation

$$\nabla_\theta \ell_n(\theta) = -\frac{1}{n}\sum_{i=1}^{n} \nabla_\theta U_\theta(x_i) + \mathbb{E}_{\pi_\theta}\left[\nabla_\theta U_\theta(X)\right].$$

The second term is still problematic!



$$\nabla_\theta \ell_n(\theta) = -\frac{1}{n}\sum_{i=1}^{n} \nabla_\theta U_\theta(x_i) + \mathbb{E}_{\pi_\theta}[\nabla_\theta U_\theta(X)].$$

The idea here is to use MCMC to sample from π_θ to approximate the expectation.

Consider the following optimization (gradient ascent) procedure

$$\begin{aligned}\theta_{k+1} &= \theta_k + \delta \nabla_\theta \ell_n(\theta_k), \\ &= \theta_k - \frac{\delta}{n} \sum_{i=1}^{n} \nabla U_{\theta_k}(x_i) + \delta \mathbb{E}_{\pi_{\theta_k}}[\nabla U_{\theta_k}(X)]\end{aligned}$$

Consider the following optimization (gradient ascent) procedure

$$\begin{aligned}\theta_{k+1} &= \theta_k + \delta \nabla_\theta \ell_n(\theta_k), \\ &= \theta_k - \frac{\delta}{n} \sum_{i=1}^{n} \nabla U_{\theta_k}(x_i) + \delta \mathbb{E}_{\pi_{\theta_k}}[\nabla U_{\theta_k}(X)]\end{aligned}$$

At iteration k, the expectation needs to be approximated, For this, we run a separate ULA chain at each k:

$$X_k^{(m)} = X_k^{(m-1)} - \gamma \nabla U_{\theta_k}(X_k^{(m-1)}) + \sqrt{2\gamma} W_k^{(m)},$$

where $(W_k^{(m)})_{m \geq 0}$ are i.i.d standard Normal random variables.

Algorithm MLE for EBMs via ULA

Input: The dataset $\{x_i\}_{i=1}^n$, the step size δ, the number of MCMC steps M, the burn-in B, the step size γ, the number of iterations N.
for $k = 1, \ldots, K$ **do**
    Fix $X_k^{(0)}$
    for $m = 1, \ldots, M$ **do**
        $X_k^{(m)} = X_k^{(m-1)} - \gamma \nabla U_{\theta_k}(X_k^{(m-1)}) + \sqrt{2\gamma} W_k^{(m)}$.
    end for
    $\nabla \ell_n(\theta_k) = -\frac{1}{n} \sum_{i=1}^n \nabla U_{\theta_k}(x_i) + \frac{1}{M-B} \sum_{m=B+1}^M \nabla U_{\theta_k}(X_k^{(m)})$.
    $\theta_{k+1} = \theta_k + \delta \nabla \ell_n(\theta_k)$.
end for

Algorithm Pseudocode for EBM training via CD-1

Input: The dataset $\{x_i\}_{i=1}^n$, the step size δ, the number of MCMC steps M, the burn-in B, the step size γ, the number of iterations N.
for $k = 1, \ldots, K$ **do**
    Randomly choose i.
    $X_k^{(0)} = x_i$.
    $X_k^{(1)} = X_k^{(0)} - \gamma \nabla U_{\theta_k}(X_k^{(0)}) + \sqrt{2\gamma} W_k^{(1)}$.
    $\nabla \ell_n(\theta_k) = -\frac{1}{n} \sum_{i=1}^n \nabla U_{\theta_k}(x_i) + \nabla U_{\theta_k}(X_k^{(1)})$.
    $\theta_{k+1} = \theta_k + \delta \nabla \ell_n(\theta_k)$.
end for

Often with the stochastic gradients:

$$n^{-1} \sum_{i=1}^{n} \nabla U_{\theta_k}(x_i) \approx J^{-1} \sum_{j=1}^{J} \nabla U_{\theta_k}(x_{i_j}).$$

Algorithm Pseudocode for EBM training via PCD

Input: The dataset $\{x_i\}_{i=1}^n$, the step size δ, the number of MCMC steps M, the burn-in B, the step size γ, the number of iterations N.
for $k = 1, \ldots, K$ **do**
    for $i = 1, \ldots, N$ **do**
        $X_k^{(i)} = X_k^{(i)} - \gamma \nabla U_{\theta_k}(X_k^{(i)}) + \sqrt{2\gamma} W_k^{(i)}$.
    end for
    $\nabla \ell_n(\theta_k) = -n^{-1} \sum_{i=1}^n \nabla U_{\theta_k}(x_i) + N^{-1} \sum_{i=1}^N \nabla U_{\theta_k}(X_k^{(i)})$.
    $\theta_{k+1} = \theta_k + \delta \nabla \ell_n(\theta_k)$.
end for

Algorithm "Short-Run MCMC" (Nijkamp et al., 2019)

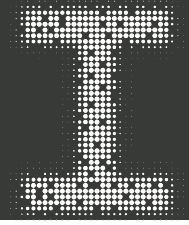
Input: The dataset $\{x_i\}_{i=1}^n$, the step size δ, the number of MCMC steps M, the burn-in B, the step size γ, the number of iterations N.
for $k = 1, \ldots, K$ **do**
    Fix $X_k^{(0)} \sim p_0$
    for $m = 1, \ldots, M$ **do**
        $X_k^{(m)} = X_k^{(m-1)} - \gamma \nabla U_{\theta_k}(X_k^{(m-1)}) + \sqrt{2\gamma} W_k^{(m)}$.
    end for
    $\nabla \ell_n(\theta_k) = -\frac{1}{n} \sum_{i=1}^n \nabla U_{\theta_k}(x_i) + \frac{1}{M-B} \sum_{m=B+1}^M \nabla U_{\theta_k}(X_k^{(m)})$.
    $\theta_{k+1} = \theta_k + \delta \nabla \ell_n(\theta_k)$.
end for

Note this method also noises the data with a single step every iteration.

We covered so far maximum likelihood estimation for EBMs with no latent variables.

We covered so far maximum likelihood estimation for EBMs with no latent variables.

Next: Latent variable EBMs and MLE.

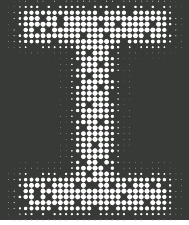


Let $p_\theta(x, z)$ be a probabilistic model with parameters θ, data x, and latent variables z.

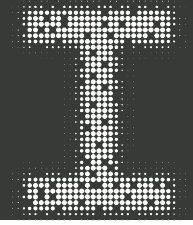
Given the data x, the typical goal is to build maximum likelihood estimate of the parameters θ by maximising the marginal likelihood

$$\theta \in \arg\max_\theta \log p_\theta(x), \tag{2}$$

where $p_\theta(x) = \int p_\theta(x, z)\mathrm{d}z$ is the marginal likelihood.



In classical literature for LVMs, the standard method to fit parameters is the EM algorithm.



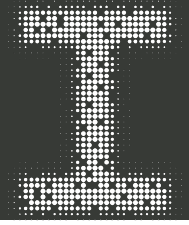
In classical literature for LVMs, the standard method to fit parameters is the EM algorithm.

$$\theta_{n+1} \in \arg\max_{\theta} Q(\theta, \theta_n), \tag{3}$$

where

$$Q(\theta, \theta_n) = \mathbb{E}_{p_{\theta_n}(z|x)}[\log p_\theta(x, z)].$$

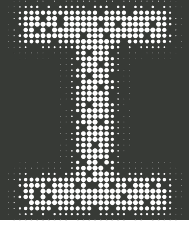
There is a long history of implementing this method. In most situations (which will certainly be the case in the case of EBMs), the posterior $p_\theta(\cdot|y)$ will not be tractable.



We below summarise one estimation strategy for the latent variable models. Assume that we are interested in running the gradient ascent scheme

$$\theta_{n+1} = \theta_n + \delta \nabla_\theta \log p_{\theta_n}(x), \tag{4}$$

where δ is the step size.

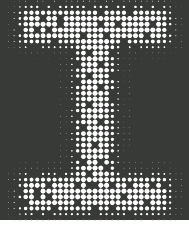


We below summarise one estimation strategy for the latent variable models. Assume that we are interested in running the gradient ascent scheme

$$\theta_{n+1} = \theta_n + \delta \nabla_\theta \log p_{\theta_n}(x), \tag{4}$$

where δ is the step size.

Mini-quiz: What can we do?



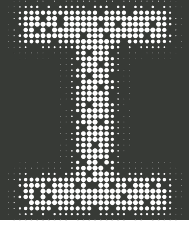
We can rely on the following proposition.

Proposition 1 (Fisher's identity)

Assume that $\int p_\theta(x, z)\mathrm{d}z < \infty$ *for all* θ. *Then the gradient of the log marginal likelihood can be written as*

$$\nabla_\theta \log p_\theta(x) = \mathbb{E}_{p_\theta(z|x)}[\nabla_\theta \log p_\theta(x, z)]. \tag{5}$$

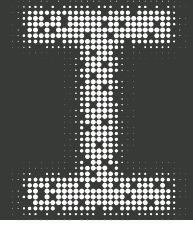
Proof on the board.



The gradient is given by

$$\nabla_\theta \log p_\theta(x) = \mathbb{E}_{p_\theta(z|x)}[\nabla_\theta \log p_\theta(x, z)],$$

however the posterior $p_\theta(\cdot|x)$ is intractable.



The SOUL method is based on the following idea. We can approximate the posterior $p_\theta(z|x)$ by running a short Langevin algorithm on the joint $p_\theta(x, z)$.

The SOUL method is based on the following idea. We can approximate the posterior $p_\theta(z|x)$ by running a short Langevin algorithm on the joint $p_\theta(x, z)$.

At iteration θ_k, one can sample from the posterior $p_{\theta_k}(z|x)$, running an MCMC sampler for M steps:

$$Z_k^{(m)} = Z_k^{(m-1)} + \gamma \nabla_\theta \log p_{\theta_k}(x, Z_k^{(m-1)}) + \sqrt{2\gamma} W_k^{(m)},$$

where $W_k^{(m)} \sim \mathcal{N}(0, I_d)$ is a standard Gaussian random variable, $Z_k^{(0)} = Z_{k-1}^{(M)}$, and $m = 1, \ldots, M$. After B burn-in samples, the expectation in the gradient can be approximated as

$$\mathbb{E}_{p_{\theta_k}(z|x)}[\nabla_\theta \log p_{\theta_k}(x, z)] \approx \frac{1}{M - B} \sum_{m=B+1}^{M} \nabla_\theta \log p_{\theta_k}(x, Z_k^{(m)}).$$

Set $U_\theta(z) = -\log p_\theta(x, z)$.

Algorithm SOUL for LVMs (De Bortoli et al., 2021)

Input: The dataset x, the step size δ, the number of MCMC steps M, the burn-in B, the step size γ, the number of iterations K.
for $k = 1, \dots, K$ **do**
  Fix $Z_k^{(0)} = Z_{k-1}^{(M)}$.
  for $m = 1, \dots, M$ **do**
    $Z_k^{(m)} = Z_k^{(m-1)} - \gamma \nabla U_{\theta_k}(Z_k^{(m-1)}) + \sqrt{2\gamma} W_k^{(m)}$.
  end for
  $\nabla \ell_n(\theta_k) = -\frac{1}{M-B} \sum_{m=B+1}^{M} \nabla U_{\theta_k}(X_k^{(m)})$.
  $\theta_{k+1} = \theta_k + \delta \nabla \ell_n(\theta_k)$.
end for

In-class exercise: Consider the latent variable EBM model:

$$p_\theta(x, z) = p_\beta(x|z)p_\alpha(z),$$

where

$$p_\alpha(z) = \frac{e^{-U_\alpha(z)}p_0(z)}{Z_\alpha}.$$

Can you derive the MLE training algorithm?

We covered so far maximum likelihood estimation for EBMs with no latent variables. We then discussed latent variable EBMs and MLE.

We covered so far maximum likelihood estimation for EBMs with no latent variables. We then discussed latent variable EBMs and MLE.

Next: Energy Based Models and Score Matching.

Recall the Euler discretisation is the *unadjusted Langevin algorithm* (ULA):

$$X_{t+1}^{\gamma} = X_t^{\gamma} + \gamma \nabla \log \Pi(X_t^{\gamma}) + \sqrt{2\gamma} W_{t+1}$$

where $(W_t)_{t \geq 0}$ are i.i.d standard Normal random variables.

Recall the Euler discretisation is the *unadjusted Langevin algorithm* (ULA):

$$X_{t+1}^{\gamma} = X_t^{\gamma} + \gamma \nabla \log \Pi(X_t^{\gamma}) + \sqrt{2\gamma} W_{t+1}$$

where $(W_t)_{t \geq 0}$ are i.i.d standard Normal random variables.

Notice that we only need the gradient of the (unnormalised) log-density for sampling.

Recall the Euler discretisation is the *unadjusted Langevin algorithm* (ULA):

$$X_{t+1}^{\gamma} = X_t^{\gamma} + \gamma \nabla \log \Pi(X_t^{\gamma}) + \sqrt{2\gamma} W_{t+1}$$

where $(W_t)_{t \geq 0}$ are i.i.d standard Normal random variables.

Notice that we only need the gradient of the (unnormalised) log-density for sampling.

The idea of score matching is to directly estimate the gradient of the log-density.

Score matching methods are based on Fisher divergence, which is defined as

$$\mathrm{F}(\pi_1||\pi_2) = \frac{1}{2}\mathbb{E}_{\pi_1}[\|\nabla \log \pi_1(X) - \nabla \log \pi_2(X)\|^2]. \tag{6}$$

In the case of generative modelling, we are interested in computing θ where

$$\theta \in \operatorname*{argmin}_{\theta} \mathrm{F}(p_{\text{data}}||\pi_\theta),$$

where

$$\mathrm{F}(p_{\text{data}}||\pi_\theta) = \frac{1}{2}\mathbb{E}_{p_{\text{data}}}[\|\nabla \log p_{\text{data}}(X) - \nabla \log \pi_\theta(X)\|^2]. \tag{7}$$

Proposition 2 (Hyvärinen, 2005)

The loss in (7) *can be written as*

$$\mathrm{F}(p_{data}||\pi_\theta) = \mathbb{E}_{p_{data}}\left[\mathrm{Tr}\nabla^2 \log \pi_\theta(X) + \frac{1}{2}\|\nabla \log \pi_\theta(X)\|^2\right]. \quad (8)$$

Proof on the board for 1D case.

Energy Based Models

Score Matching

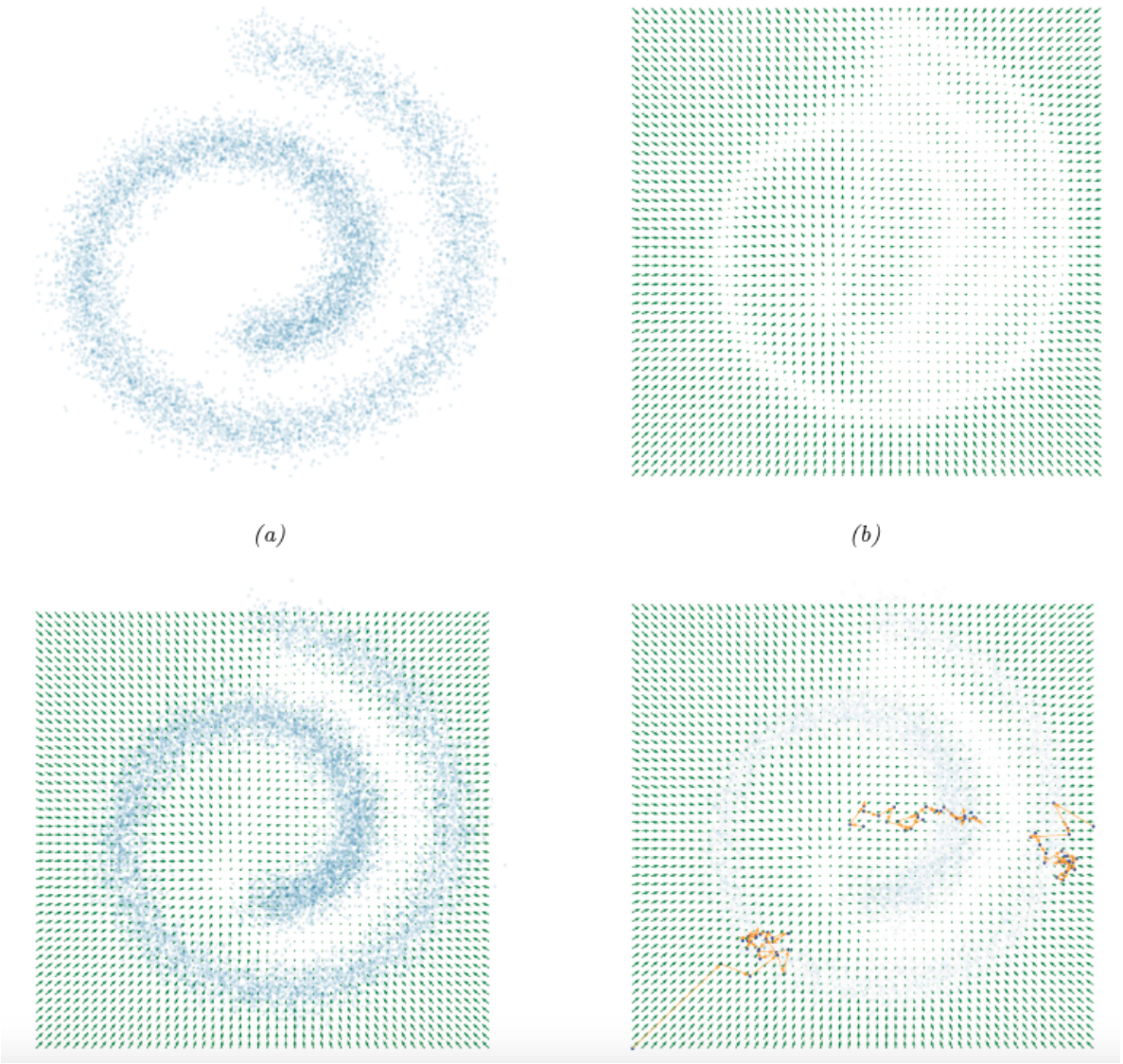


Figure: From Murphy (2023)

Unfortunately, this idea doesn't quite work in practice.

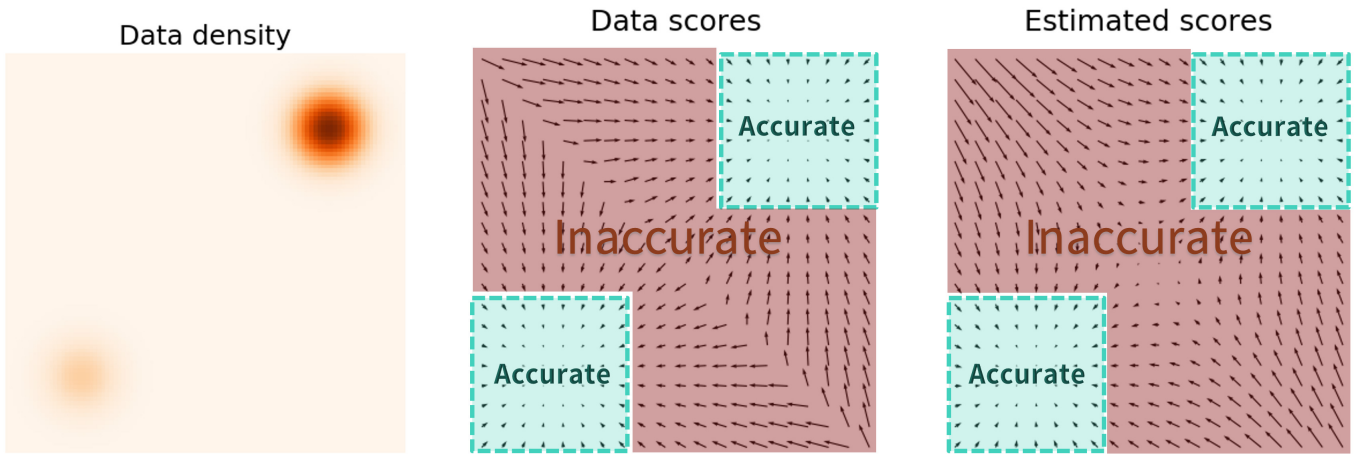


Image credit: `https://yang-song.net/blog/2021/score/`

Score matching can be inefficient and is expensive due to the Hessian term. Instead, we can leverage the following idea.

Let us define the noisy version of the data distribution as

$$p^{\sigma}_{\text{data}}(x) = \int p_{\text{data}}(x')K(x|x')\mathrm{d}x',$$

where $K(x|x')$ is a kernel.

Proposition 3 (Vincent, 2011)

Given the Fisher divergence between the noisy data distribution and the model distribution,

$$\mathrm{F}(p^{\sigma}_{data}||\pi_\theta) = \mathbb{E}_{p^{\sigma}_{data}}\left[\frac{1}{2}\|\nabla \log p^{\sigma}_{data}(X) - \nabla \log \pi_\theta(X)\|^2\right], \quad (9)$$

we have

$$\mathrm{F}(p^{\sigma}_{data}||\pi_\theta) = \mathbb{E}_{p(x,x')}\left[\frac{1}{2}\|\nabla_x \log K(X|X') - \nabla \log \pi_\theta(X)\|^2\right]. \quad (10)$$

where $p(x, x') = p_{data}(x')K(x|x')$.

Proposition 3 (Vincent, 2011)

Given the Fisher divergence between the noisy data distribution and the model distribution,

$$\mathrm{F}(p^{\sigma}_{data}||\pi_\theta) = \mathbb{E}_{p^{\sigma}_{data}}\left[\frac{1}{2}\|\nabla \log p^{\sigma}_{data}(X) - \nabla \log \pi_\theta(X)\|^2\right], \quad (9)$$

we have

$$\mathrm{F}(p^{\sigma}_{data}||\pi_\theta) = \mathbb{E}_{p(x,x')}\left[\frac{1}{2}\|\nabla_x \log K(X|X') - \nabla \log \pi_\theta(X)\|^2\right]. \quad (10)$$

where $p(x, x') = p_{data}(x')K(x|x')$.

Proof on the board.

For example, one can simply choose

$$K(x|x') = \mathcal{N}(x; x', \sigma^2 I_d),$$

where σ is a hyperparameter. In this situation, the estimate will take the form (Song and Kingma, 2021)

$$\mathrm{F}(p^{\sigma}_{\text{data}}||\pi_\theta) = \mathbb{E}_{p_{\text{data}}(x')}\mathbb{E}_{z\sim\mathcal{N}(0,\sigma^2 I_d)}\left[\frac{1}{2}\left\|\frac{z}{\sigma} + \nabla_x \log \pi_\theta(x' + \sigma z)\right\|^2\right].$$

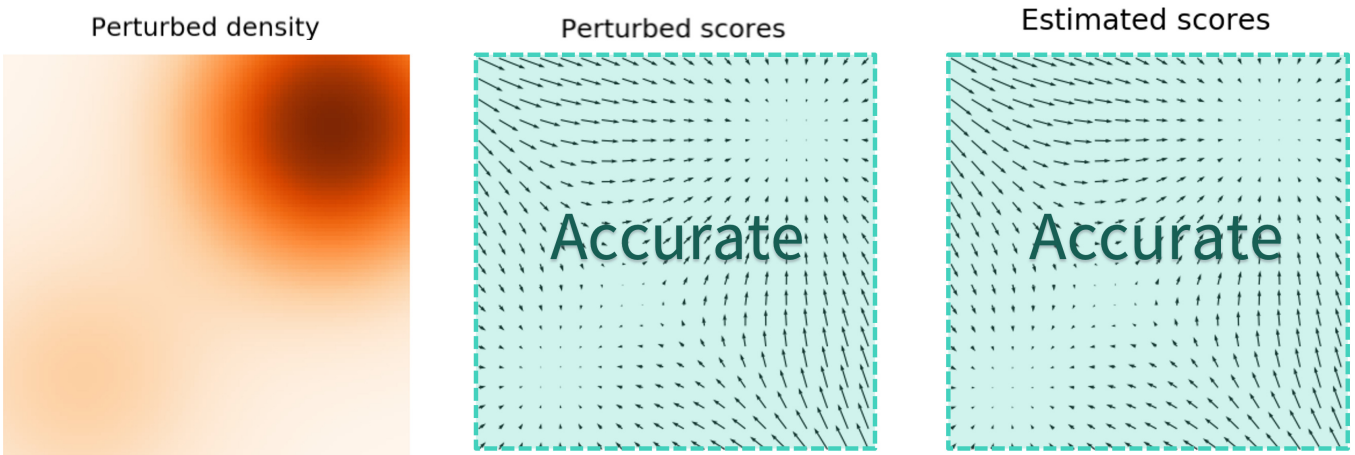
One can estimate this score, by plugging the empirical data distribution in using

$$\hat{p}_{\text{data}} = \frac{1}{n} \sum_{i=1}^{n} \delta_{x_i}.$$

and using samples from the Gaussian distribution, which results in the unbiased estimate of the loss

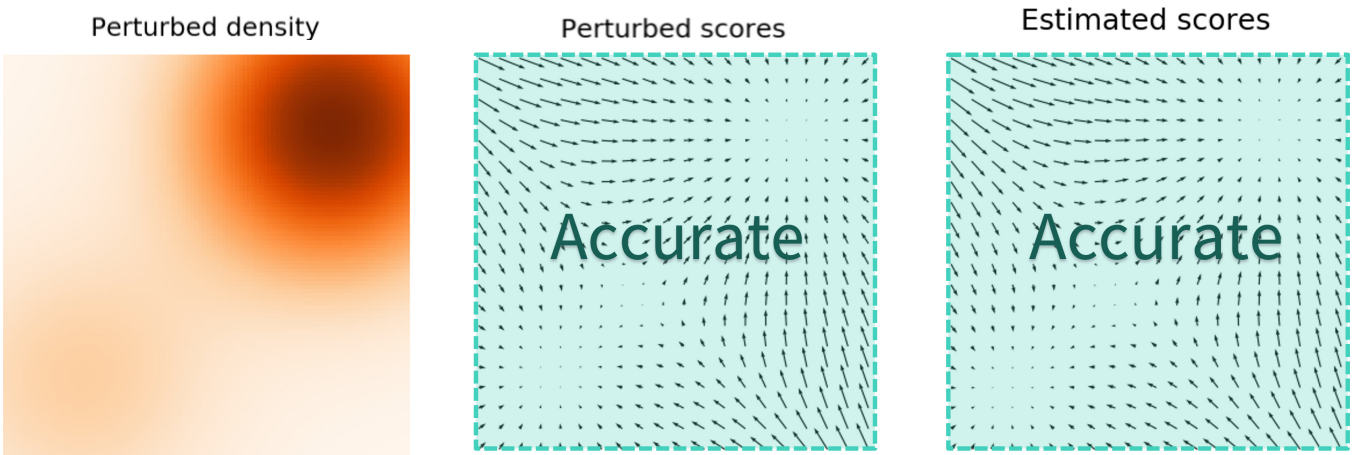
$$\hat{\mathrm{F}}(p^{\sigma}_{\text{data}} || \pi_\theta) = \frac{1}{2n} \sum_{i=1}^{n} \left\| \frac{z^{(i)}}{\sigma} + \nabla_x \log \pi_\theta(x_i + \sigma z^{(i)}) \right\|^2.$$

Perturbed scores are better behaved:



But there is a bias as we approximate the noisy data distribution.

Perturbed scores are better behaved:



But there is a bias as we approximate the noisy data distribution.

This observation led to the idea of *adding* progressively more noise to data and learn associated scores (a bit later).

Image credit: `https://yang-song.net/blog/2021/score/`

Sliced score matching defines a new *sliced* Fisher divergence (Song et al., 2020)

$$\mathrm{SF}(p_{\text{data}}||\pi_\theta) = \mathbb{E}_{p(v)}\mathbb{E}_{p_{\text{data}}}\left[\frac{1}{2}\left(v^\top \nabla_x \log \pi_{\text{data}}(X) - v^\top \nabla_x \log \pi_\theta(X)\right)^2\right],$$

where $p(v)$ is a distribution where we sample projections v from.

Sliced score matching defines a new *sliced* Fisher divergence (Song et al., 2020)

$$\text{SF}(p_{\text{data}}||\pi_\theta) = \mathbb{E}_{p(v)}\mathbb{E}_{p_{\text{data}}}\left[\frac{1}{2}\left(v^\top \nabla_x \log \pi_{\text{data}}(X) - v^\top \nabla_x \log \pi_\theta(X)\right)^2\right],$$

where $p(v)$ is a distribution where we sample projections v from. The sliced Fisher divergence can be written as

$$\text{SF}(p_{\text{data}}||\pi_\theta) = \mathbb{E}_{p_v}\mathbb{E}_{p_{\text{data}}}\left[v^\top \nabla_x^2 \log \pi_\theta(X) v + \frac{1}{2}(v^\top \nabla_x \log \pi_\theta(X))^2\right].$$

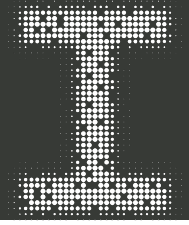
Sliced score matching defines a new *sliced* Fisher divergence (Song et al., 2020)

$$\mathrm{SF}(p_{\text{data}}||\pi_\theta) = \mathbb{E}_{p(v)}\mathbb{E}_{p_{\text{data}}}\left[\frac{1}{2}\left(v^\top \nabla_x \log \pi_{\text{data}}(X) - v^\top \nabla_x \log \pi_\theta(X)\right)^2\right],$$

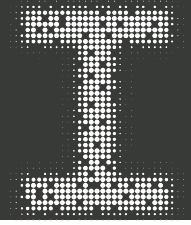
where $p(v)$ is a distribution where we sample projections v from. The sliced Fisher divergence can be written as

$$\mathrm{SF}(p_{\text{data}}||\pi_\theta) = \mathbb{E}_{p_v}\mathbb{E}_{p_{\text{data}}}\left[v^\top \nabla_x^2 \log \pi_\theta(X) v + \frac{1}{2}(v^\top \nabla_x \log \pi_\theta(X))^2\right].$$

The term $v^\top \nabla_x^2 \log \pi_\theta(X) v$ can be computed without storing the Hessian but still expensive compared to the denoising score matching.

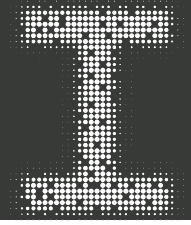


Denoising score matching appeared as the cheapest option to approximate scores, but has an inherent bias as it estimates the noisy scores.



Denoising score matching appeared as the cheapest option to approximate scores, but has an inherent bias as it estimates the noisy scores.

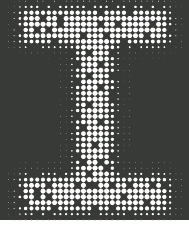
For $\sigma \approx 0$, the bias is small, but training is unstable and mixing can be arbitrarily slow, especially for p_{data} with multiple modes or which concentrates on low-dimensional manifolds.



Denoising score matching appeared as the cheapest option to approximate scores, but has an inherent bias as it estimates the noisy scores.

For $\sigma \approx 0$, the bias is small, but training is unstable and mixing can be arbitrarily slow, especially for p_{data} with multiple modes or which concentrates on low-dimensional manifolds.

Can we "bridge" easy to sample distributions and the data distribution, via the use of noise scales?



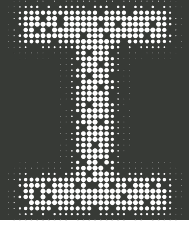
This is the idea in Song and Ermon (2019) that unlocked the path to diffusion models.

This is the idea in Song and Ermon (2019) that unlocked the path to diffusion models.

Let $\{\sigma_i\}_{i=1}^{L}$ denote the sequence of L noise levels that satisfies $\frac{\sigma_1}{\sigma_2} = \ldots = \frac{\sigma_{L-1}}{\sigma_L} > 1$. For each noise level, we define

$$p_{\text{data}}^{\sigma_i}(x) = \int p_{\text{data}}(x') K_{\sigma_i}(x|x') \mathrm{d}x',$$

The pedestrian approach is to train a model for each noise level, but this is computationally expensive.



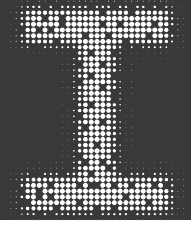
This is the idea in Song and Ermon (2019) that unlocked the path to diffusion models.

Let $\{\sigma_i\}_{i=1}^{L}$ denote the sequence of L noise levels that satisfies $\frac{\sigma_1}{\sigma_2} = \ldots = \frac{\sigma_{L-1}}{\sigma_L} > 1$. For each noise level, we define

$$p_{\text{data}}^{\sigma_i}(x) = \int p_{\text{data}}(x') K_{\sigma_i}(x|x') \mathrm{d}x',$$

The pedestrian approach is to train a model for each noise level, but this is computationally expensive.

We will instead choose a single network for all noise levels: Noise conditional score networks (NCSN).



Let us denote the DSM loss for a single noise level as

$$\ell(\theta, \sigma) = \mathrm{F}(p^{\sigma}_{\text{data}} || \pi_\theta) = \mathbb{E}_{p^{\sigma}_{\text{data}}} \left[\frac{1}{2} \| \nabla \log p^{\sigma}_{\text{data}}(X) - \mathrm{s}_\theta(X) \|^2 \right].$$

Let us denote the DSM loss for a single noise level as

$$\ell(\theta, \sigma) = \mathrm{F}(p^{\sigma}_{\text{data}} || \pi_\theta) = \mathbb{E}_{p^{\sigma}_{\text{data}}} \left[\frac{1}{2} \| \nabla \log p^{\sigma}_{\text{data}}(X) - \mathrm{s}_\theta(X) \|^2 \right].$$

Let us clarify this for the specific kernel K. Recall that we can write

$$\ell(\theta, \sigma) = \mathbb{E}_{p(x,x')} \left[\frac{1}{2} \| \nabla_x \log K_\sigma(X|X') - \mathrm{s}_\theta(X, \sigma) \|^2 \right].$$

where $p(x, x') = p_{\text{data}}(x') K_\sigma(x|x')$.

The final loss is then

$$\ell(\theta, \sigma) = \mathbb{E}_{p(x,x')} \left[\frac{1}{2} \left\| \frac{X - X'}{\sigma^2} + \mathrm{s}_\theta(X, \sigma) \right\|^2 \right].$$

It is typical to build the final loss as

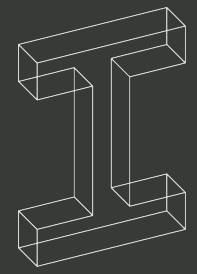
$$\ell(\theta) = \frac{1}{L} \sum_{i=1}^{L} \lambda(\sigma_i) \ell(\theta, \sigma_i),$$

where $\lambda(\sigma_i)$ is a weighting function. Song and Ermon (2019) propose $\lambda(\sigma) = \sigma^2$. Therefore, the training phase of the algorithm is to find

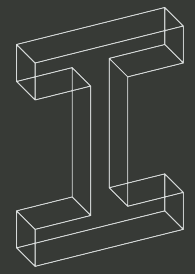
$$\theta_\star \in \operatorname*{argmin}_\theta \ell(\theta).$$

Algorithm Pseudocode for sampling from EBMs using annealed Langevin dynamics

Input: The trained NCSN $s_{\theta_\star}(x, \sigma)$, the number of noise levels L, the step size ε, the number of iterations T
$X_0 =$ random initialisation.
for $i = 1, \ldots, L$ **do**
    $\gamma_i = \varepsilon \frac{\sigma_i^2}{\sigma_L^2}$.
    for $t = 1, \ldots, T$ **do**
        $X_{t+1} = X_t + \gamma_i s_{\theta_\star}(X_t, \sigma_i) + \sqrt{2\gamma_i} W_{t+1}$.
    end for
    $X_0 = X_T$.
end for

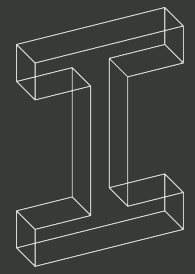


We covered the basics of EBMs and latent variable EBMs.



We covered the basics of EBMs and latent variable EBMs.

The training algorithms have been predominantly based on Markov chain Monte Carlo methods.



We covered the basics of EBMs and latent variable EBMs.

The training algorithms have been predominantly based on Markov chain Monte Carlo methods.

Next: Training and inference using variational inference.

Part II: Variational Autoencoders

Part II: Variational Autoencoders

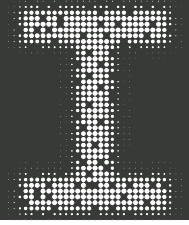
- An introduction to variational inference (on the board)

Part II: Variational Autoencoders

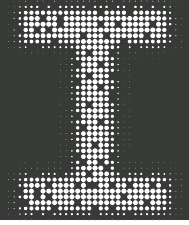
- An introduction to variational inference (on the board)
- Variational Autoencoders

Part II: Variational Autoencoders

- An introduction to variational inference (on the board)
- Variational Autoencoders
- Extensions of VAEs

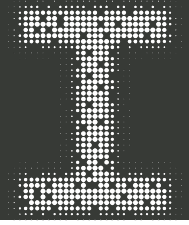


VAEs are highly related to latent variable EBMs, in fact, they share the same model structure for $p_\theta(x, z)$.



VAEs are highly related to latent variable EBMs, in fact, they share the same model structure for $p_\theta(x, z)$.

Instead of training the model via MCMC as in the case of EBMs, VAEs build a variational family of the form $q_\phi(z|x)$.



VAEs are highly related to latent variable EBMs, in fact, they share the same model structure for $p_\theta(x, z)$.

Instead of training the model via MCMC as in the case of EBMs, VAEs build a variational family of the form $q_\phi(z|x)$.

We then derive the ELBO in the almost identical way.

In this case, we have a latent variable model, $p_\theta(x) = \int p_\theta(x, z)\mathrm{d}z$ and we can write

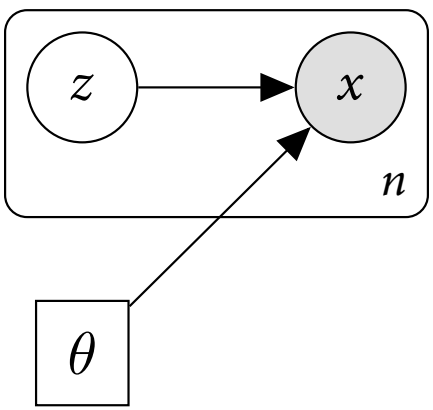
$$\begin{aligned}
\mathbb{E}_{p_{\text{data}}(x)}[\log p_\theta(x)] &= \mathbb{E}_{p_{\text{data}}(x)}\left[\log \int p_\theta(x, z)\mathrm{d}z\right], \\
&= \mathbb{E}_{p_{\text{data}}(x)}\left[\log \int p_\theta(x, z)\frac{q_\phi(z|x)}{q_\phi(z|x)}\mathrm{d}z\right], \\
&\geq \mathbb{E}_{p_{\text{data}}(x)}\left[\int q_\phi(z|x)\log\frac{p_\theta(x, z)}{q_\phi(z|x)}\mathrm{d}z\right], \\
&= \mathbb{E}_{p_{\text{data}}(x)}\left[\mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x, z)] - \mathbb{E}_{q_\phi(z|x)}[\log q_\phi(z|x)]\right],
\end{aligned}$$

where we have used Jensen's inequality in the third line. An instantiation of the last expression with $\hat{p}_{\text{data}}$:

$$\text{ELBO}(\theta, \phi) = \frac{1}{n}\sum_{i=1}^{n}\left[\mathbb{E}_{q_\phi(z|x_i)}[\log p_\theta(x_i, z)] - \mathbb{E}_{q_\phi(z|x_i)}[\log q_\phi(z|x_i)]\right]. \tag{11}$$

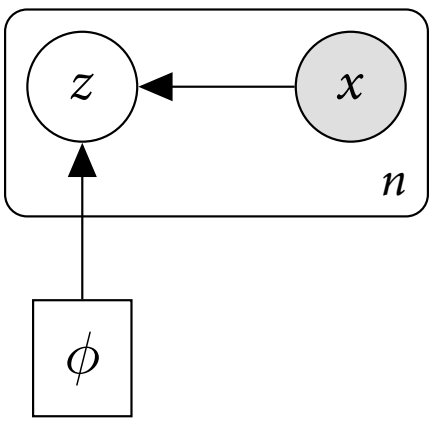
An alternative derivation assumes the following decoder model:

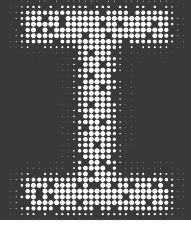
$$p_\theta(x, z) = \prod_{i=1}^{n} p_\theta(x_i|z_i)p(z_i).$$



An alternative derivation assumes the following encoder model:

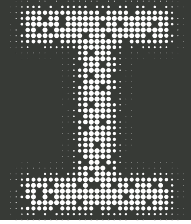
$$q_\phi(z|x) = \prod_{i=1}^{n} q_\phi(z_i|x_i).$$





Then classical VI derivation gives the ELBO:

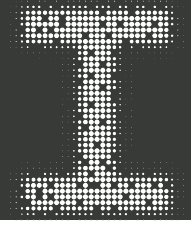
$$\text{ELBO}(\theta, \phi) = \sum_{i=1}^{n} \left[\mathbb{E}_{q_\phi(z_i|x_i)}[\log p_\theta(x_i, z_i)] - \mathbb{E}_{q_\phi(z_i|x_i)}[\log q_\phi(z_i|x_i)] \right].$$



Then classical VI derivation gives the ELBO:

$$\text{ELBO}(\theta, \phi) = \sum_{i=1}^{n} \left[\mathbb{E}_{q_\phi(z_i|x_i)}[\log p_\theta(x_i, z_i)] - \mathbb{E}_{q_\phi(z_i|x_i)}[\log q_\phi(z_i|x_i)] \right].$$

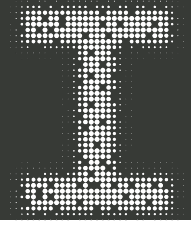
In the case of VAEs, $p_\theta(x_i|z_i)$ is called the decoder and $q_\phi(z_i|x_i)$ is called the encoder.



The decoder, as we will consider here, is just defined as

$$p_\theta(x|z) = \mathcal{N}(x; \mu_\theta(z), \sigma^2 I_d),$$

where $\mu_\theta(z) : \mathrm{Z} \to \mathrm{X}$ is a neural network. The decoder is a standard Gaussian distribution with mean $\mu_\theta(z)$ and variance $\sigma^2 I_d$.



The decoder, as we will consider here, is just defined as

$$p_\theta(x|z) = \mathcal{N}(x; \mu_\theta(z), \sigma^2 I_d),$$

where $\mu_\theta(z) : \mathrm{Z} \to \mathrm{X}$ is a neural network. The decoder is a standard Gaussian distribution with mean $\mu_\theta(z)$ and variance $\sigma^2 I_d$.

Sometimes, following the deterministic AE literature, the neural network $\mu_\theta(z)$ is also called a decoder

The encoder is defined as

$$q_\phi(z|x) = \mathcal{N}(z; \mu_\phi(x), \mathrm{diag}(\sigma^2_\phi(x))),$$

where $\mu_\phi(x) : \mathrm{X} \to \mathrm{Z}$ and $\sigma^2_\phi(x) : \mathrm{X} \to \mathrm{Z}$ are neural networks. The encoder is a Gaussian distribution with mean $\mu_\phi(x)$ and variance $\sigma^2_\phi(x)$. The encoder is trained to encode the data x into the latent space. The usual parameterisation is that a neural network for the mean and *log-variance* is trained. In fact, it is usual to train a single network $h_\phi(x) : \mathrm{X} \to \mathrm{Z} \times \mathrm{Z}$ and then split the output into the mean and the log-variance. The log-variance is used to ensure that the variance is positive.

Variational Autoencoders

VAE graph

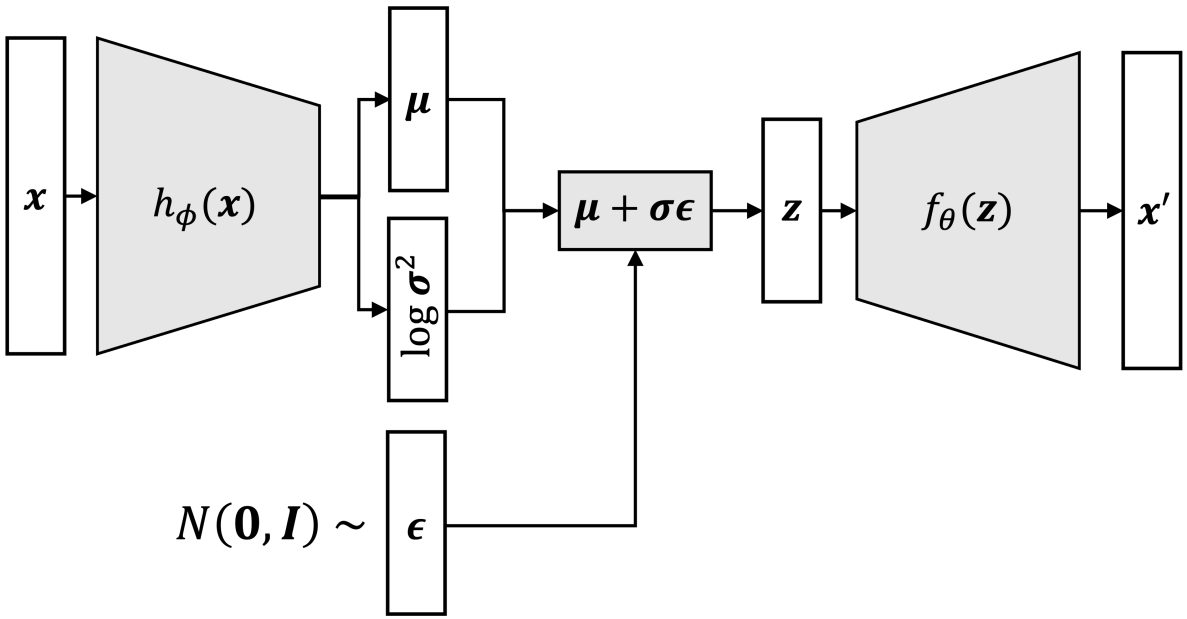


Figure: From (Bernstein, 2023)

Variational Autoencoders

VAE graph

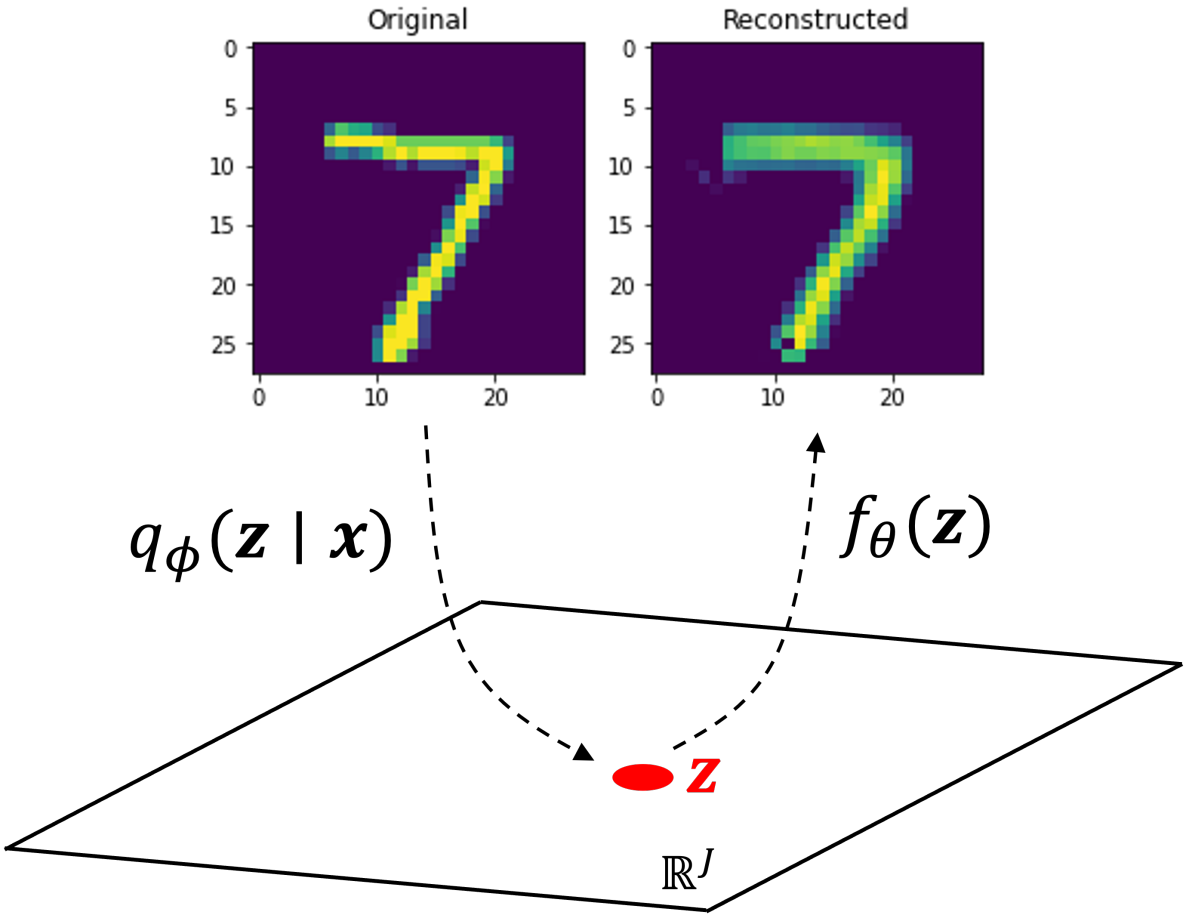


Figure: From (Bernstein, 2023)

Variational Autoencoders

VAE graph

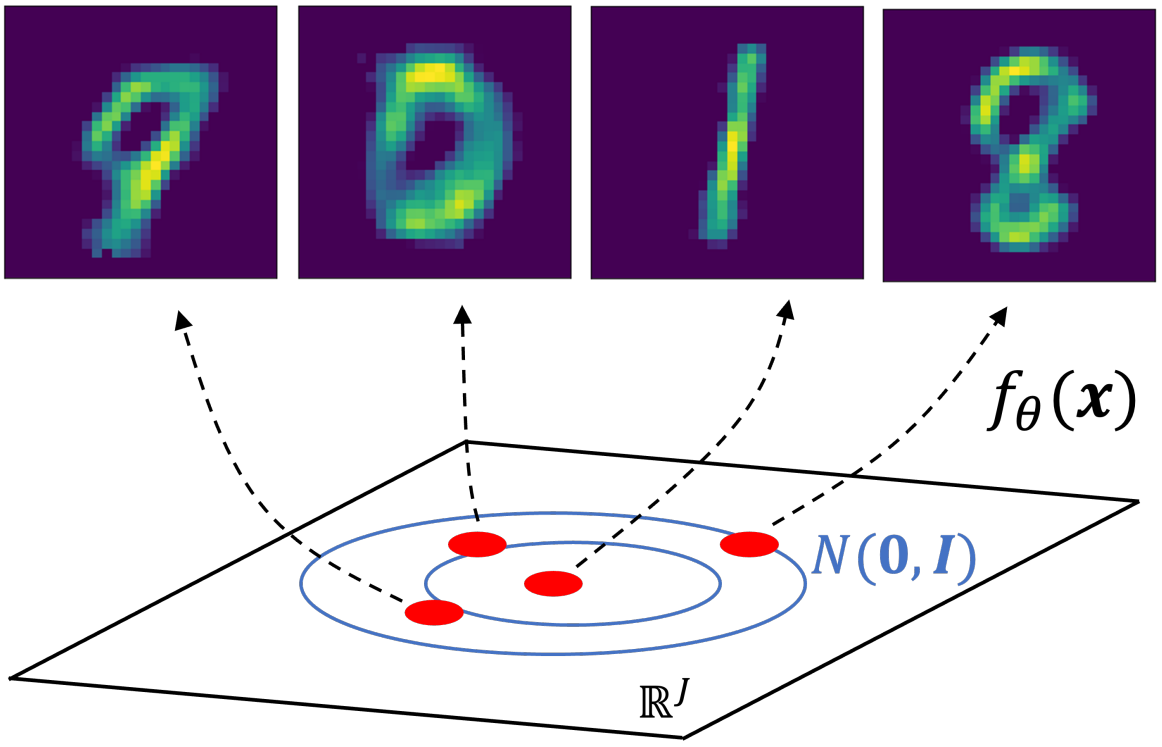


Figure: From (Bernstein, 2023)

Variational Autoencoders

VAE graph

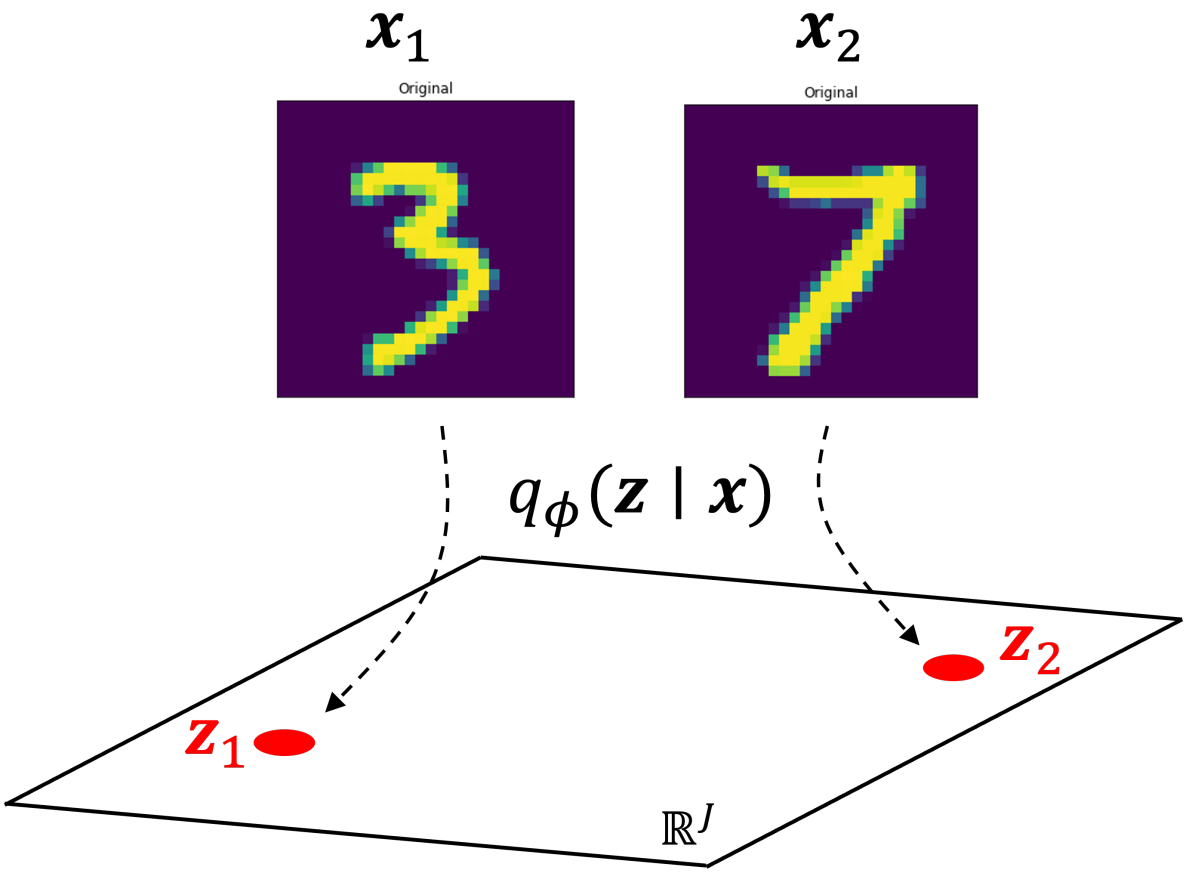


Figure: From (Bernstein, 2023)

Variational Autoencoders

VAE graph

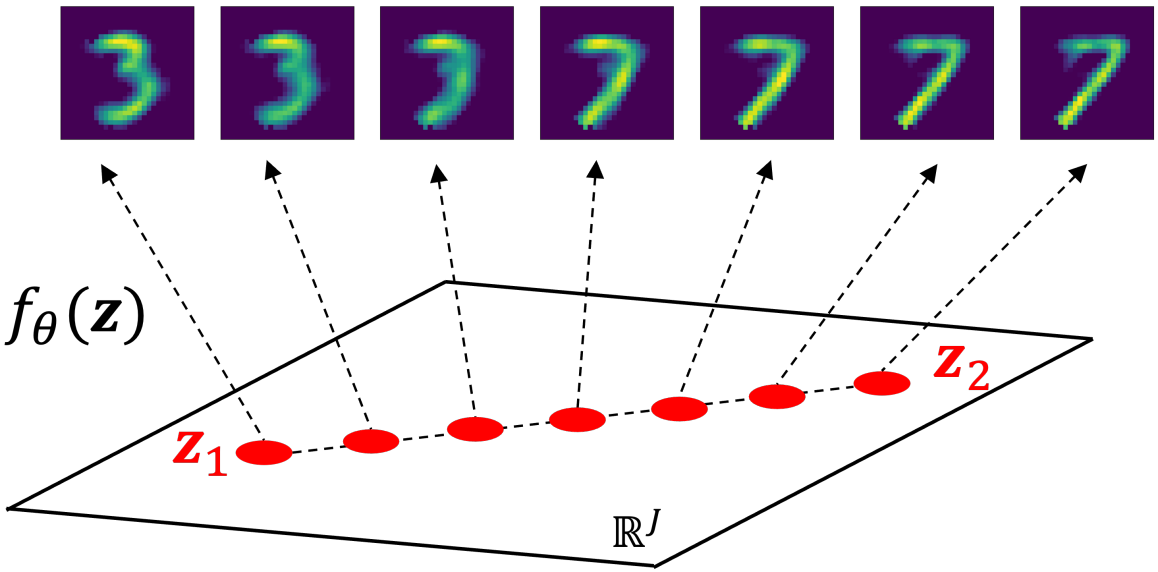


Figure: From (Bernstein, 2023)

Hierarchical Variational Autoencoders

Deep VAEs

Hierarchical VAEs are a generalisation of VAEs where the latent space is split into multiple levels. Let us define the model (Murphy, 2023)

$$p_\theta(x, z_{1:L}) = p_\theta(z_L) \left[\prod_{l=L-1}^{1} p_\theta(z_l|z_{l+1}) \right] p_\theta(x|z_1),$$

and the variational family

$$q_\phi(z|x) = q_\phi(z_L|x) \prod_{l=L-1}^{1} q_\phi(z_l|z_{l+1}, x).$$

Hierarchical Variational Autoencoders

Deep VAEs

Figure 21.13: Samples from a VDVAE model (trained on FFHQ dataset) from different levels of the hierarchy. From Figure 1 of [Chi21a]. Used with kind permission of Rewon Child.

Figure: From (Murphy, 2023)

All generative models we have looked at so far considered the data as i.i.i.d samples.

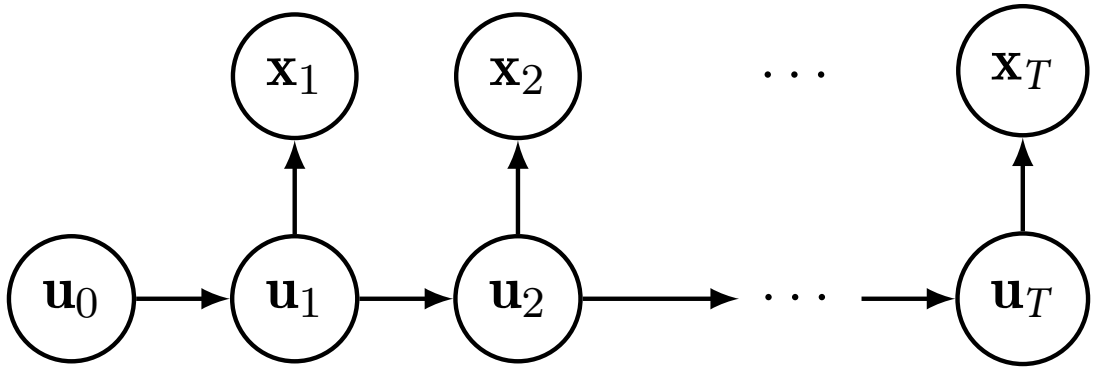
All generative models we have looked at so far considered the data as i.i.i.d samples.

Next: Dynamical VAEs.

The filtering problem

Introduction

Consider a standard state-space model:



The generic probabilistic description of such a system is given by

$$\mathbf{u}_0 \sim \mu_0(\mathbf{u}_0) \tag{12}$$

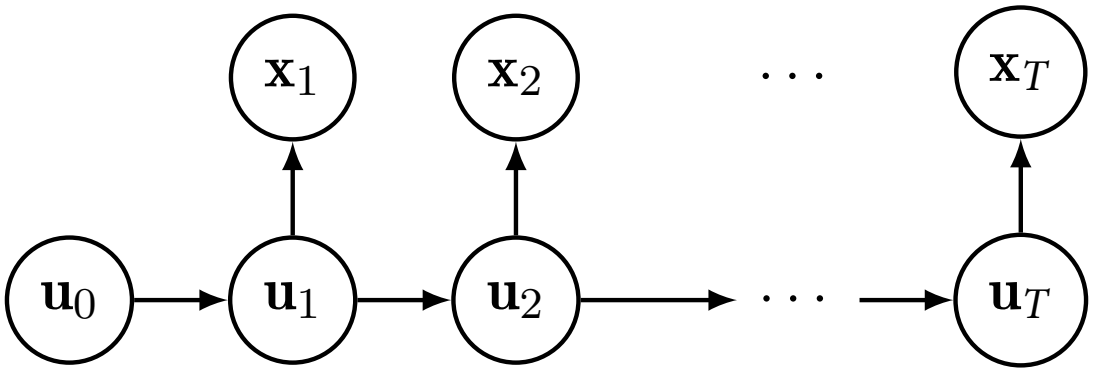
$$\mathbf{u}_k|\mathbf{u}_{k-1} \sim p(\mathbf{u}_k|\mathbf{u}_{k-1}) \tag{13}$$

$$\mathbf{x}_k|\mathbf{u}_{k-1} \sim p(\mathbf{x}_k|\mathbf{u}_{k-1}) \tag{14}$$

where $p(\mathbf{u}_k|\mathbf{u}_{k-1})$ is called a *transition* model and $p(\mathbf{x}_k|\mathbf{u}_k)$ is called *the likelihood*.

The filtering problem

Introduction

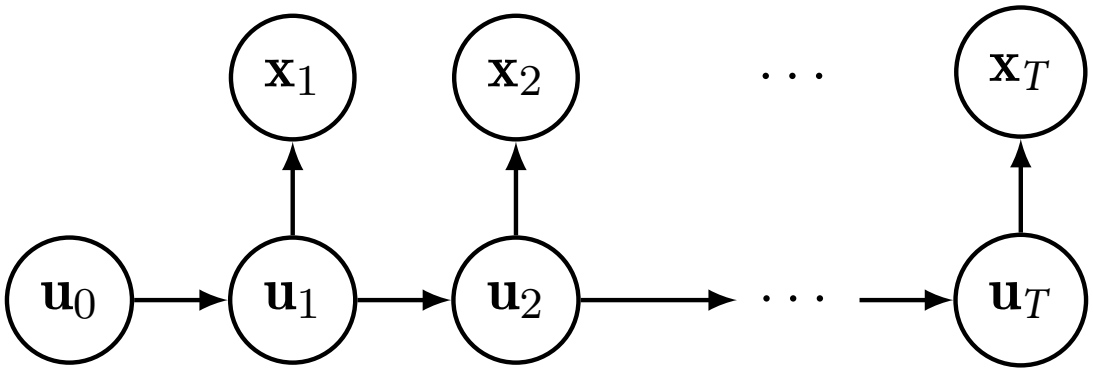


A standard problem within this setting is to estimate latent states $(\mathbf{u}_k)_{k\geq 0}$ given observations $(\mathbf{x}_k)_{k\geq 1}$.

- The *filtering problem* is to estimate $\mathbf{u}_k$ given $\mathbf{x}_1, \ldots, \mathbf{x}_k$.
- The *smoothing problem* is to estimate $\mathbf{u}_k$ given $\mathbf{x}_1, \ldots, \mathbf{x}_T$.

The filtering problem

Introduction



A standard problem within this setting is to estimate latent states $(\mathbf{u}_k)_{k\geq 0}$ given observations $(\mathbf{x}_k)_{k\geq 1}$.

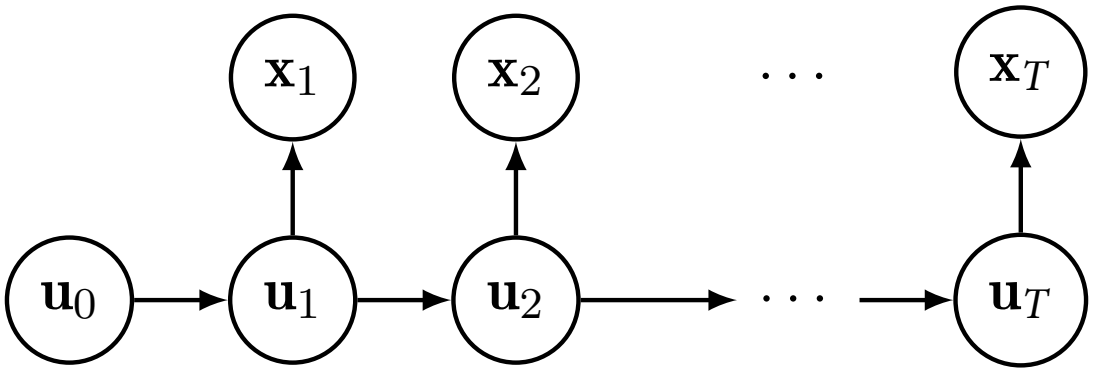
- The *filtering problem* is to estimate $\mathbf{u}_k$ given $\mathbf{x}_1, \ldots, \mathbf{x}_k$.
- The *smoothing problem* is to estimate $\mathbf{u}_k$ given $\mathbf{x}_1, \ldots, \mathbf{x}_T$.

A number of applications in data assimilation

- Object tracking
- Weather forecasting
- Volatility estimation

The filtering problem

Introduction



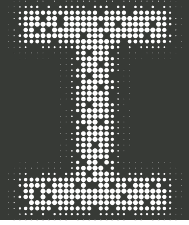
A standard problem within this setting is to estimate latent states $(\mathbf{u}_k)_{k\geq 0}$ given observations $(\mathbf{x}_k)_{k\geq 1}$.

- The *filtering problem* is to estimate $\mathbf{u}_k$ given $\mathbf{x}_1, \ldots, \mathbf{x}_k$.
- The *smoothing problem* is to estimate $\mathbf{u}_k$ given $\mathbf{x}_1, \ldots, \mathbf{x}_T$.

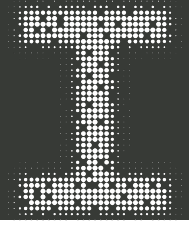
A number of applications in data assimilation

- Object tracking
- Weather forecasting
- Volatility estimation

60 years of research: A massive data assimilation and stochastic filtering literature.



The standard methods estimate the filtering distribution $p(\mathbf{u}_k|\mathbf{x}_{1:k})$ and the smoothing distribution $p(\mathbf{u}_k|\mathbf{x}_{1:T})$.



The standard methods estimate the filtering distribution $p(\mathbf{u}_k|\mathbf{x}_{1:k})$ and the smoothing distribution $p(\mathbf{u}_k|\mathbf{x}_{1:T})$.

- For linear Gaussian models, the Kalman filter/smoother provides the optimal solution.

The standard methods estimate the filtering distribution $p(\mathbf{u}_k|\mathbf{x}_{1:k})$ and the smoothing distribution $p(\mathbf{u}_k|\mathbf{x}_{1:T})$.

- For linear Gaussian models, the Kalman filter/smoother provides the optimal solution.
- For nonlinear-Gaussian models, Kalmanesque filters: Extended Kalman filter, unscented Kalman filter.

The standard methods estimate the filtering distribution $p(\mathbf{u}_k|\mathbf{x}_{1:k})$ and the smoothing distribution $p(\mathbf{u}_k|\mathbf{x}_{1:T})$.

- For linear Gaussian models, the Kalman filter/smoother provides the optimal solution.
- For nonlinear-Gaussian models, Kalmanesque filters: Extended Kalman filter, unscented Kalman filter.
- For nonlinear and non-Gaussian models: particle filters.

Autoencoding unstructured data

Kalman VAEs (Fraccaro et al., 2017)

Let us denote the data with $\mathbf{y}_{1:T}$. We look at a three layer model:

$$\begin{aligned}
\mathbf{y}_t|\mathbf{x}_t &\sim p_\theta(\mathbf{y}_t|\mathbf{x}_t) \\
\mathbf{x}_t|\mathbf{u}_t &\sim p_\nu(\mathbf{x}_t|\mathbf{u}_t) \\
\mathbf{u}_t|\mathbf{u}_{t-1} &\sim p_\Lambda(\mathbf{u}_t|\mathbf{u}_{t-1}) \\
\Lambda &\sim p(\Lambda)
\end{aligned}$$

This model includes:

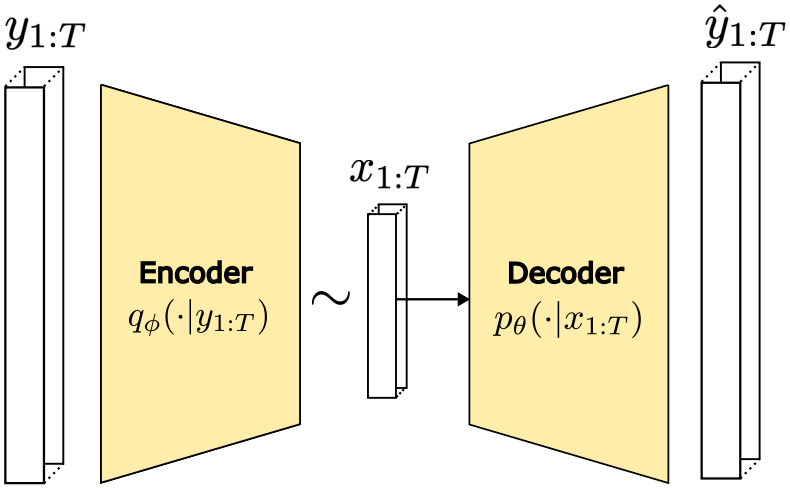
- $\mathbf{y}_{1:T}$: the unstructured, high-dimensional data
- $\mathbf{x}_{1:T}$: pseudo-observations, embedding of high-dimensional data to a low-dimensional space
- $\mathbf{u}_{1:T}$: latent states, the hidden variables driven by a prescribed dynamics

Autoencoding unstructured data

Kalman VAEs (Fraccaro et al., 2017)

Autoencoder Structure

- Data encoded to latent space
- Latents decoded

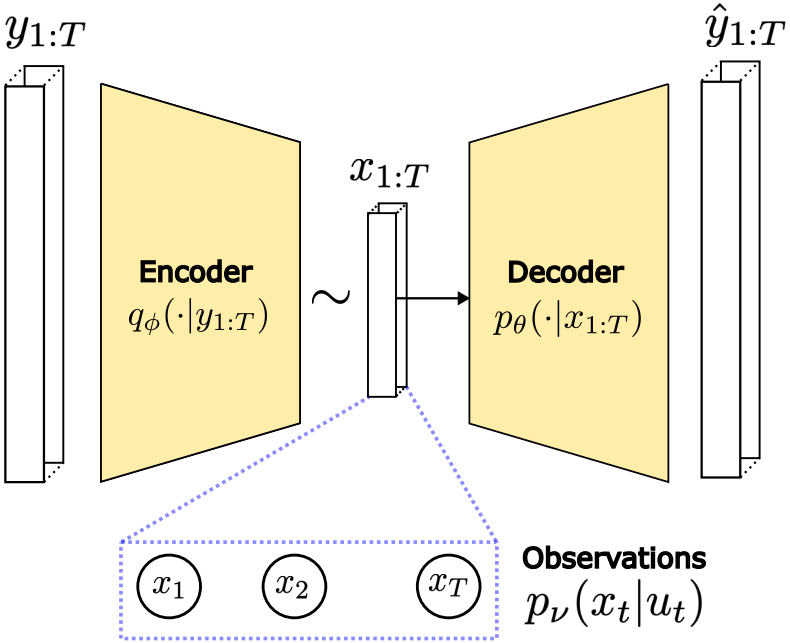


Autoencoding unstructured data

Kalman VAEs (Fraccaro et al., 2017)

Autoencoder Structure

- Data encoded to latent space
- Latents decoded



Latent Dynamics

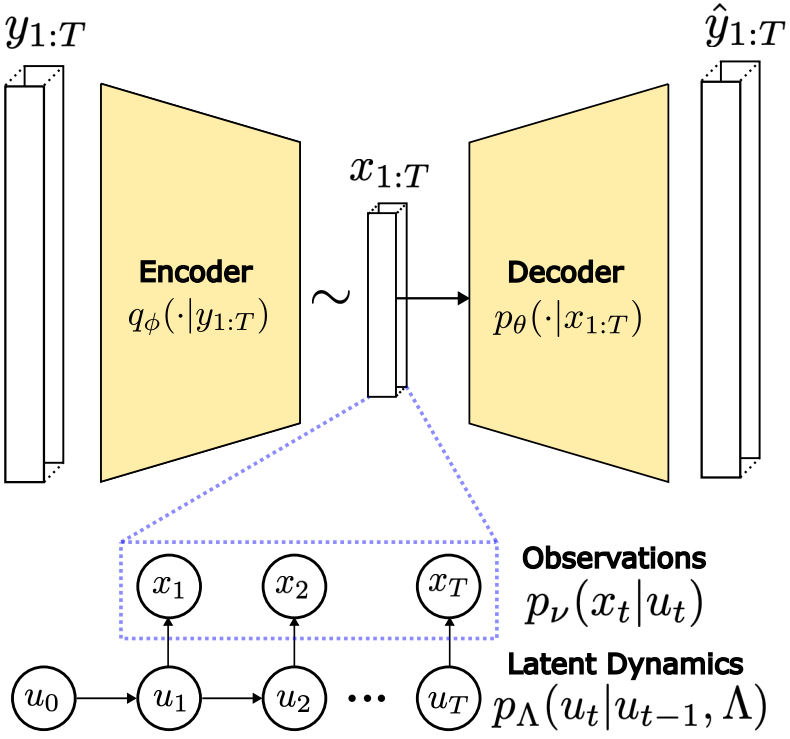
- Latent encoding represents 'pseudo-obervations' of underlying dynamic system

Autoencoding unstructured data

Kalman VAEs (Fraccaro et al., 2017)

Autoencoder Structure

- Data encoded to latent space
- Latents decoded



Latent Dynamics

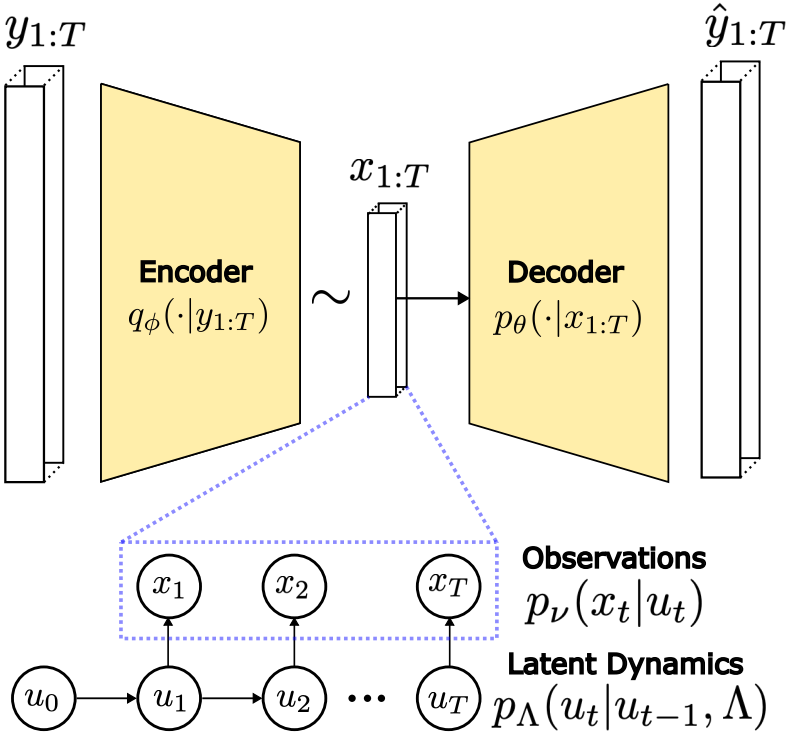
- Latent encoding represents 'pseudo-obervations' of underlying dynamic system
- Filtering used to infer latent states

Autoencoding unstructured data

Kalman VAEs (Fraccaro et al., 2017)

Autoencoder Structure

- Data encoded to latent space
- Latents decoded



Latent Dynamics

- Latent encoding represents 'pseudo-obervations' of underlying dynamic system
- Filtering used to infer latent states

Inference

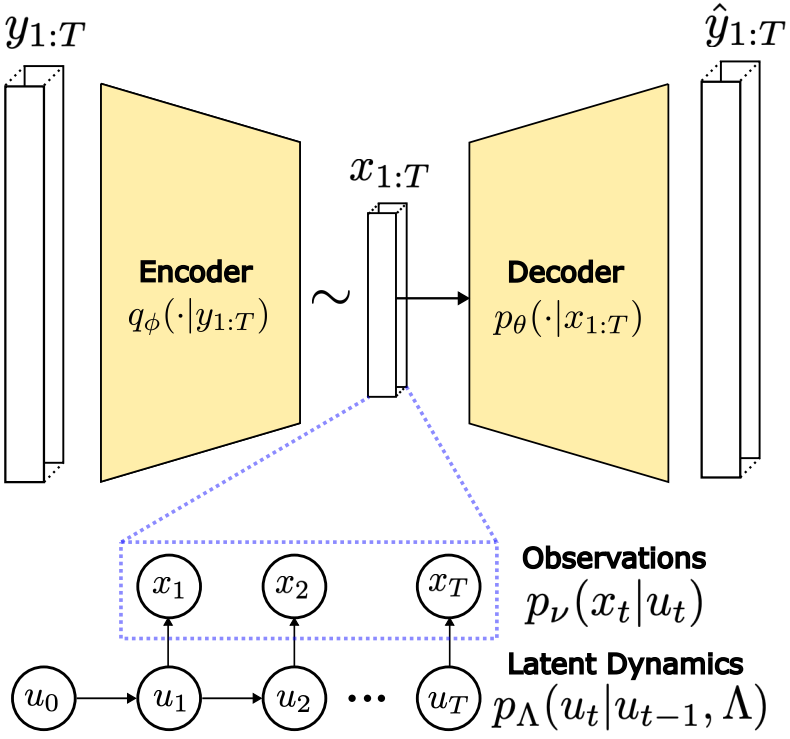
- Prior over latent space defined by assumed dynamic model

Autoencoding unstructured data

Kalman VAEs (Fraccaro et al., 2017)

Autoencoder Structure

- Data encoded to latent space
- Latents decoded



Latent Dynamics

- Latent encoding represents 'pseudo-obervations' of underlying dynamic system
- Filtering used to infer latent states

Inference

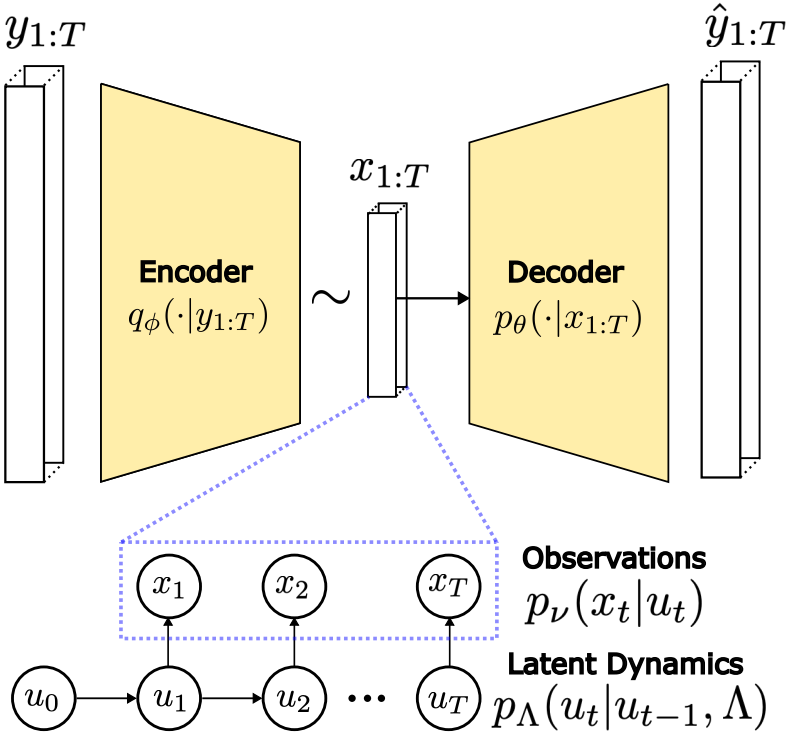
- Prior over latent space defined by assumed dynamic model
- Variational approximation made to latent posterior

Autoencoding unstructured data

Kalman VAEs (Fraccaro et al., 2017)

Autoencoder Structure

- Data encoded to latent space
- Latents decoded



Latent Dynamics

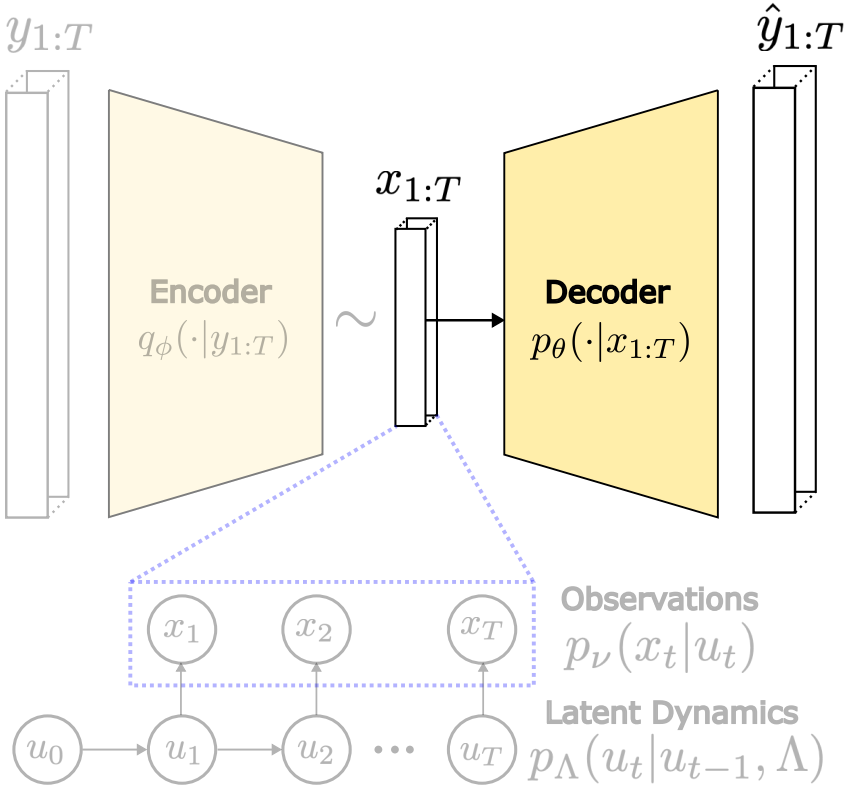
- Latent encoding represents 'pseudo-obervations' of underlying dynamic system
- Filtering used to infer latent states

Inference

- Prior over latent space defined by assumed dynamic model
- Variational approximation made to latent posterior
- Evidence lower bound of $\log p(\mathbf{y}_{1:T})$ maximised to learn encoding and decoding

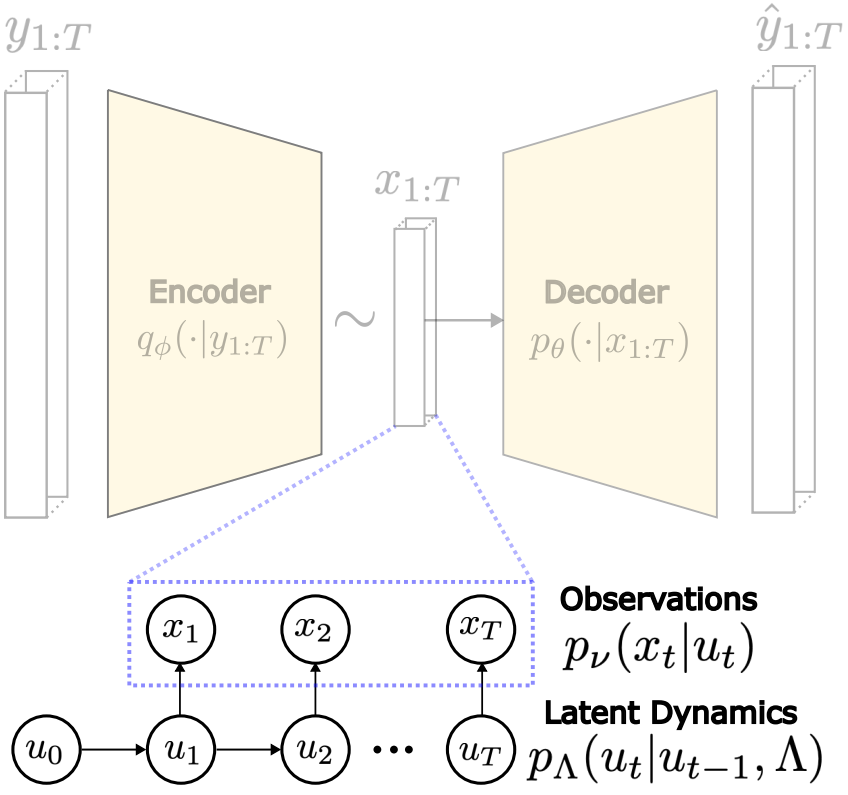
Factorize joint distribution:

$$p(\mathbf{u}_{1:T}, \mathbf{x}_{1:T}, \mathbf{y}_{1:T}) = p(\mathbf{y}_{1:T}|\mathbf{x}_{1:T}, \mathbf{u}_{1:T})p(\mathbf{x}_{1:T}|\mathbf{u}_{1:T})p(\mathbf{u}_{1:T})$$
$$= p_\theta(\mathbf{y}_{1:T}|\mathbf{x}_{1:T})p_\nu(\mathbf{x}_{1:T}|\mathbf{u}_{1:T})p_\Lambda(\mathbf{u}_{1:T}).$$



Factorize joint distribution:

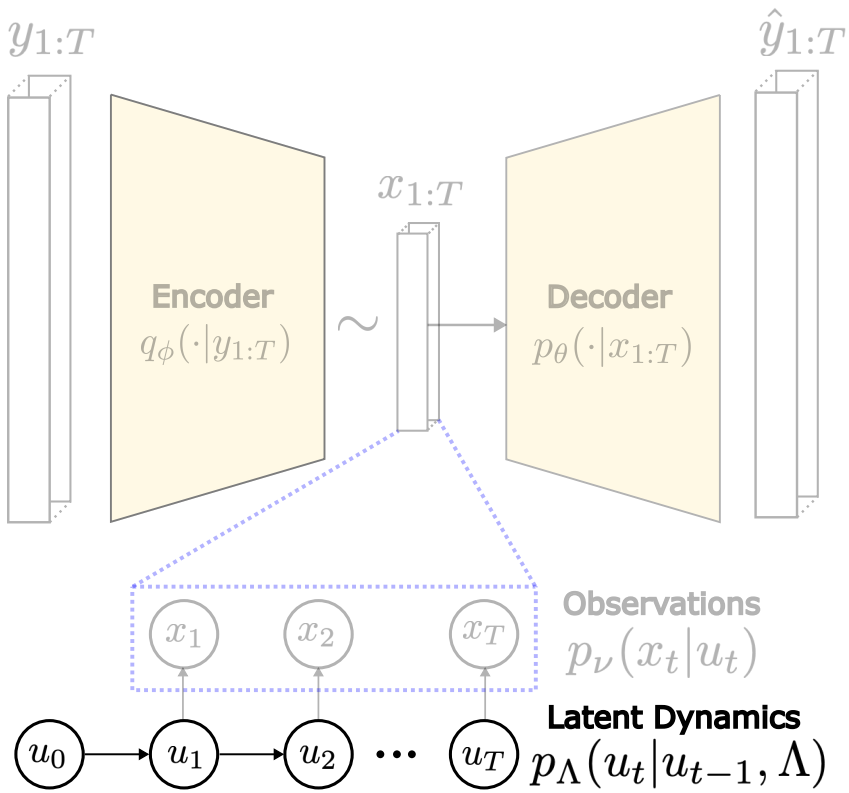
$$p(\mathbf{u}_{1:T}, \mathbf{x}_{1:T}, \mathbf{y}_{1:T}) = p(\mathbf{y}_{1:T}|\mathbf{x}_{1:T}, \mathbf{u}_{1:T})p(\mathbf{x}_{1:T}|\mathbf{u}_{1:T})p(\mathbf{u}_{1:T})$$
$$= p_\theta(\mathbf{y}_{1:T}|\mathbf{x}_{1:T})p(\mathbf{x}_{1:T}|\mathbf{u}_{1:T})p_\Lambda(\mathbf{u}_{1:T})$$



Model

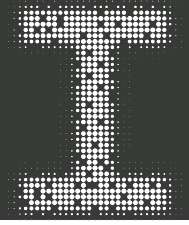
Factorize joint distribution:

$$p(\mathbf{u}_{1:T}, \mathbf{x}_{1:T}, \mathbf{y}_{1:T}) = p(\mathbf{y}_{1:T}|\mathbf{x}_{1:T}, \mathbf{u}_{1:T})p(\mathbf{x}_{1:T}|\mathbf{u}_{1:T})p(\mathbf{u}_{1:T})$$
$$= p_\theta(\mathbf{y}_{1:T}|\mathbf{x}_{1:T})p(\mathbf{x}_{1:T}|\mathbf{u}_{1:T})p_{\mathbf{\Lambda}}(\mathbf{u}_{1:T})$$



Variational family

Model derivation



We have described the model $p(\mathbf{u}_{1:T}, \mathbf{x}_{1:T}, \mathbf{y}_{1:T})$.

Variational family

Model derivation

We have described the model $p(\mathbf{u}_{1:T}, \mathbf{x}_{1:T}, \mathbf{y}_{1:T})$.

Next, we specify a variational family $q(\mathbf{u}_{1:T}, \mathbf{x}_{1:T} | \mathbf{y}_{1:T})$:

$$\begin{aligned} q(\mathbf{u}_{1:T}, \mathbf{x}_{1:T} | \mathbf{y}_{1:T}) &= q_{\Lambda,\nu}(\mathbf{u}_{1:T} | \mathbf{x}_{1:T}) q_\phi(\mathbf{x}_{1:T} | \mathbf{y}_{1:T}) \\ &= q_{\Lambda,\nu}(\mathbf{u}_{1:T} | \mathbf{x}_{1:T}) \prod_{t=1}^{T} q_\phi(\mathbf{x}_t | \mathbf{y}_t). \end{aligned}$$

Variational family

Model derivation

We have described the model $p(\mathbf{u}_{1:T}, \mathbf{x}_{1:T}, \mathbf{y}_{1:T})$.

Next, we specify a variational family $q(\mathbf{u}_{1:T}, \mathbf{x}_{1:T}|\mathbf{y}_{1:T})$:

$$\begin{aligned} q(\mathbf{u}_{1:T}, \mathbf{x}_{1:T}|\mathbf{y}_{1:T}) &= q_{\Lambda,\nu}(\mathbf{u}_{1:T}|\mathbf{x}_{1:T}) q_\phi(\mathbf{x}_{1:T}|\mathbf{y}_{1:T}) \\ &= q_{\Lambda,\nu}(\mathbf{u}_{1:T}|\mathbf{x}_{1:T}) \prod_{t=1}^{T} q_\phi(\mathbf{x}_t|\mathbf{y}_t). \end{aligned}$$

- $q_{\Lambda,\nu}(\mathbf{u}_{1:T}|\mathbf{x}_{1:T})$ is an approximate (or exact) filter

Variational family

Model derivation

We have described the model $p(\mathbf{u}_{1:T}, \mathbf{x}_{1:T}, \mathbf{y}_{1:T})$.

Next, we specify a variational family $q(\mathbf{u}_{1:T}, \mathbf{x}_{1:T} | \mathbf{y}_{1:T})$:

$$\begin{aligned} q(\mathbf{u}_{1:T}, \mathbf{x}_{1:T} | \mathbf{y}_{1:T}) &= q_{\Lambda,\nu}(\mathbf{u}_{1:T} | \mathbf{x}_{1:T}) q_\phi(\mathbf{x}_{1:T} | \mathbf{y}_{1:T}) \\ &= q_{\Lambda,\nu}(\mathbf{u}_{1:T} | \mathbf{x}_{1:T}) \prod_{t=1}^{T} q_\phi(\mathbf{x}_t | \mathbf{y}_t). \end{aligned}$$

- $q_{\Lambda,\nu}(\mathbf{u}_{1:T} | \mathbf{x}_{1:T})$ is an approximate (or exact) filter
- $q_\phi(\mathbf{x}_t | \mathbf{y}_t)$ is the encoder for each $\mathbf{y}_t$.

Evidence lower bound (ELBO)

Model derivation

$$\text{ELBO} = \int \log \left[\frac{p_\theta(\mathbf{y}_{1:T}|\mathbf{x}_{1:T}) p(\mathbf{x}_{1:T}|\mathbf{u}_{1:T}) p_\Lambda(\mathbf{u}_{1:T})}{q(\mathbf{u}_{1:T}, \mathbf{x}_{1:T}|\mathbf{y}_{1:T})} \right] q(\mathbf{u}_{1:T}, \mathbf{x}_{1:T}|\mathbf{y}_{1:T}) \mathrm{d}\mathbf{x}_{1:T} \mathrm{d}\mathbf{u}_{1:T}$$

Evidence lower bound (ELBO)

Model derivation

$$\text{ELBO} = \int \log \left[\frac{p_\theta(\mathbf{y}_{1:T}|\mathbf{x}_{1:T}) p(\mathbf{x}_{1:T}|\mathbf{u}_{1:T}) p_\Lambda(\mathbf{u}_{1:T})}{q(\mathbf{u}_{1:T}, \mathbf{x}_{1:T}|\mathbf{y}_{1:T})} \right] q(\mathbf{u}_{1:T}, \mathbf{x}_{1:T}|\mathbf{y}_{1:T}) \mathrm{d}\mathbf{x}_{1:T} \mathrm{d}\mathbf{u}_{1:T}$$

Variational family:

$$q(\mathbf{u}_{1:T}, \mathbf{x}_{1:T}|\mathbf{y}_{1:T})$$

Evidence lower bound (ELBO)

Model derivation

$$\text{ELBO} = \int \log \left[\frac{p_\theta(\mathbf{y}_{1:T}|\mathbf{x}_{1:T}) p(\mathbf{x}_{1:T}|\mathbf{u}_{1:T}) p_\Lambda(\mathbf{u}_{1:T})}{q(\mathbf{u}_{1:T}, \mathbf{x}_{1:T}|\mathbf{y}_{1:T})} \right] q(\mathbf{u}_{1:T}, \mathbf{x}_{1:T}|\mathbf{y}_{1:T}) \mathrm{d}\mathbf{x}_{1:T} \mathrm{d}\mathbf{u}_{1:T}$$

Variational family:

$$q(\mathbf{u}_{1:T}, \mathbf{x}_{1:T}|\mathbf{y}_{1:T}) = q_{\Lambda,\nu}(\mathbf{u}_{1:T}|\mathbf{x}_{1:T}) q_\phi(\mathbf{x}_{1:T}|\mathbf{y}_{1:T})$$

Evidence lower bound (ELBO)

Model derivation

$$\text{ELBO} = \int \log \left[\frac{p_\theta(\mathbf{y}_{1:T}|\mathbf{x}_{1:T}) p(\mathbf{x}_{1:T}|\mathbf{u}_{1:T}) p_\Lambda(\mathbf{u}_{1:T})}{q(\mathbf{u}_{1:T}, \mathbf{x}_{1:T}|\mathbf{y}_{1:T})} \right] q(\mathbf{u}_{1:T}, \mathbf{x}_{1:T}|\mathbf{y}_{1:T}) \mathrm{d}\mathbf{x}_{1:T} \mathrm{d}\mathbf{u}_{1:T}$$

Variational family:

$$\begin{aligned} q(\mathbf{u}_{1:T}, \mathbf{x}_{1:T}|\mathbf{y}_{1:T}) &= q_{\Lambda,\nu}(\mathbf{u}_{1:T}|\mathbf{x}_{1:T}) q_\phi(\mathbf{x}_{1:T}|\mathbf{y}_{1:T}) \\ &= q_{\Lambda,\nu}(\mathbf{u}_{1:T}|\mathbf{x}_{1:T}) \prod_{t=1}^{T} q_\phi(\mathbf{x}_t|\mathbf{y}_t) \end{aligned}$$

Evidence lower bound (ELBO)

Model derivation

$$\text{ELBO} = \int \log \left[\frac{p_\theta(\mathbf{y}_{1:T}|\mathbf{x}_{1:T}) p(\mathbf{x}_{1:T}|\mathbf{u}_{1:T}) p_\Lambda(\mathbf{u}_{1:T})}{q(\mathbf{u}_{1:T}, \mathbf{x}_{1:T}|\mathbf{y}_{1:T})} \right] q(\mathbf{u}_{1:T}, \mathbf{x}_{1:T}|\mathbf{y}_{1:T}) \mathrm{d}\mathbf{x}_{1:T} \mathrm{d}\mathbf{u}_{1:T}$$

Variational family:

$$\begin{aligned} q(\mathbf{u}_{1:T}, \mathbf{x}_{1:T}|\mathbf{y}_{1:T}) &= q_{\Lambda,\nu}(\mathbf{u}_{1:T}|\mathbf{x}_{1:T}) q_\phi(\mathbf{x}_{1:T}|\mathbf{y}_{1:T}) \\ &= q_{\Lambda,\nu}(\mathbf{u}_{1:T}|\mathbf{x}_{1:T}) \prod_{t=1}^{T} q_\phi(\mathbf{x}_t|\mathbf{y}_t) \end{aligned}$$

If we let the variational distribution $q_{\Lambda,\nu}(\mathbf{u}_{1:T}|\mathbf{x}_{1:T})$ be the full latent posterior $p(\mathbf{u}_{1:T}|\mathbf{x}_{1:T})$, the following drops out:

Evidence lower bound (ELBO)

Model derivation

$$\text{ELBO} = \int \log \left[\frac{p_\theta(\mathbf{y}_{1:T}|\mathbf{x}_{1:T}) p(\mathbf{x}_{1:T}|\mathbf{u}_{1:T}) p_\Lambda(\mathbf{u}_{1:T})}{q(\mathbf{u}_{1:T}, \mathbf{x}_{1:T}|\mathbf{y}_{1:T})} \right] q(\mathbf{u}_{1:T}, \mathbf{x}_{1:T}|\mathbf{y}_{1:T}) \mathrm{d}\mathbf{x}_{1:T} \mathrm{d}\mathbf{u}_{1:T}$$

Variational family:

$$\begin{aligned} q(\mathbf{u}_{1:T}, \mathbf{x}_{1:T}|\mathbf{y}_{1:T}) &= q_{\Lambda,\nu}(\mathbf{u}_{1:T}|\mathbf{x}_{1:T}) q_\phi(\mathbf{x}_{1:T}|\mathbf{y}_{1:T}) \\ &= q_{\Lambda,\nu}(\mathbf{u}_{1:T}|\mathbf{x}_{1:T}) \prod_{t=1}^{T} q_\phi(\mathbf{x}_t|\mathbf{y}_t) \end{aligned}$$

If we let the variational distribution $q_{\Lambda,\nu}(\mathbf{u}_{1:T}|\mathbf{x}_{1:T})$ be the full latent posterior $p(\mathbf{u}_{1:T}|\mathbf{x}_{1:T})$, the following drops out:

$$\begin{aligned} \text{ELBO} &= \int \log \left[\frac{p_\theta(\mathbf{y}_{1:T}|\mathbf{x}_{1:T}) p(\mathbf{x}_{1:T})}{q_\phi(\mathbf{x}_{1:T}|\mathbf{y}_{1:T})} \right] q_\phi(\mathbf{x}_{1:T}|\mathbf{y}_{1:T}) \mathrm{d}\mathbf{x}_{1:T} \\ &= \mathbb{E}_{q_\phi(\mathbf{x}_{1:T}|\mathbf{y}_{1:T})} \left[\log p_\theta(\mathbf{y}_{1:T}|\mathbf{x}_{1:T}) - \log q_\phi(\mathbf{x}_{1:T}|\mathbf{y}_{1:T}) + \log p(\mathbf{x}_{1:T}) \right] \end{aligned}$$

Evidence Lower Bound

Model derivation

$$\mathbb{E}_{q_\phi(\mathbf{x}_{1:T}|\mathbf{y}_{1:T})}\left[\log p_\theta(\mathbf{y}_{1:T}|\mathbf{x}_{1:T}) - \log q_\phi(\mathbf{x}_{1:T}|\mathbf{y}_{1:T}) + \log p(\mathbf{x}_{1:T})\right]$$

Evidence Lower Bound

Model derivation

$$\mathbb{E}_{q_\phi(\mathbf{x}_{1:T}|\mathbf{y}_{1:T})}\left[\log p_\theta(\mathbf{y}_{1:T}|\mathbf{x}_{1:T}) - \log q_\phi(\mathbf{x}_{1:T}|\mathbf{y}_{1:T}) + \log p(\mathbf{x}_{1:T})\right]$$

- In general, not analytically tractable.

Evidence Lower Bound

Model derivation

$$\mathbb{E}_{q_\phi(\mathbf{x}_{1:T}|\mathbf{y}_{1:T})}\left[\log p_\theta(\mathbf{y}_{1:T}|\mathbf{x}_{1:T}) - \log q_\phi(\mathbf{x}_{1:T}|\mathbf{y}_{1:T}) + \log p(\mathbf{x}_{1:T})\right]$$

- In general, not analytically tractable.
- Monte Carlo sample taken $\mathbf{x}_{1:T}^{(i)} \sim q_\phi(\cdot|\mathbf{y}_{1:T})$, for $i = 1, \ldots, N$.

Evidence Lower Bound

Model derivation

$$\mathbb{E}_{q_\phi(\mathbf{x}_{1:T}|\mathbf{y}_{1:T})}\left[\log p_\theta(\mathbf{y}_{1:T}|\mathbf{x}_{1:T}) - \log q_\phi(\mathbf{x}_{1:T}|\mathbf{y}_{1:T}) + \log p(\mathbf{x}_{1:T})\right]$$

- In general, not analytically tractable.
- Monte Carlo sample taken $\mathbf{x}_{1:T}^{(i)} \sim q_\phi(\cdot|\mathbf{y}_{1:T})$, for $i = 1, \ldots, N$.
- $\log p(\mathbf{x}_{1:T})$ is approximated w.r.t. underlying filter (exact or approximate)

This is our objective we wish to maximise to learn the autoencoding after sampling $\mathbf{x}_{1:T}$.

$$\mathcal{F}^N(\phi, \theta) = \frac{1}{N}\sum_{i=1}^{N} \log p_\theta(\mathbf{y}_{1:T}|\mathbf{x}_{1:T}^{(i)}) - \log q_\phi(\mathbf{x}_{1:T}^{(i)}|\mathbf{y}_{1:T}) + \log p(\mathbf{x}_{1:T}^{(i)})$$

Decoder - Reconstruction Likelihood Let us look at a single term:

$$\log p_\theta(\mathbf{y}_{1:T}|\mathbf{x}_{1:T}) - \log q_\phi(\mathbf{x}_{1:T}|\mathbf{y}_{1:T}) + \log p(\mathbf{x}_{1:T})$$

- Decoder: The map from latent $\mathbf{x}_t \to \hat{\mathbf{y}}_t$ (reconstruction).

Decoder - Reconstruction Likelihood Let us look at a single term:

$$\log p_\theta(\mathbf{y}_{1:T}|\mathbf{x}_{1:T}) - \log q_\phi(\mathbf{x}_{1:T}|\mathbf{y}_{1:T}) + \log p(\mathbf{x}_{1:T})$$

- Decoder: The map from latent $\mathbf{x}_t \to \hat{\mathbf{y}}_t$ (reconstruction).
- This should encode as much prior information about the data generating process as possible:

Decoder - Reconstruction Likelihood Let us look at a single term:

$$\log p_\theta(\mathbf{y}_{1:T}|\mathbf{x}_{1:T}) - \log q_\phi(\mathbf{x}_{1:T}|\mathbf{y}_{1:T}) + \log p(\mathbf{x}_{1:T})$$

- Decoder: The map from latent $\mathbf{x}_t \to \hat{\mathbf{y}}_t$ (reconstruction).
- This should encode as much prior information about the data generating process as possible:
  - Deep nets if completely unknown

Decoder - Reconstruction Likelihood Let us look at a single term:

$$\log p_\theta(\mathbf{y}_{1:T}|\mathbf{x}_{1:T}) - \log q_\phi(\mathbf{x}_{1:T}|\mathbf{y}_{1:T}) + \log p(\mathbf{x}_{1:T})$$

- Decoder: The map from latent $\mathbf{x}_t \to \hat{\mathbf{y}}_t$ (reconstruction).
- This should encode as much prior information about the data generating process as possible:
  - Deep nets if completely unknown
  - Linear (or generalised linear) models

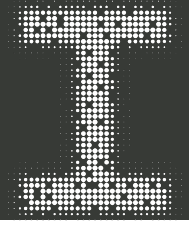
Decoder - Reconstruction Likelihood Let us look at a single term:

$$\log p_\theta(\mathbf{y}_{1:T}|\mathbf{x}_{1:T}) - \log q_\phi(\mathbf{x}_{1:T}|\mathbf{y}_{1:T}) + \log p(\mathbf{x}_{1:T})$$

- Decoder: The map from latent $\mathbf{x}_t \to \hat{\mathbf{y}}_t$ (reconstruction).
- This should encode as much prior information about the data generating process as possible:
  - Deep nets if completely unknown
  - Linear (or generalised linear) models
- If some structure is known this can be used to inform the variational encoder.

Encoder - Variational Posterior

$$\mathcal{F}(\phi, \theta) = \log p_\theta(\mathbf{y}_{1:T}|\mathbf{x}_{1:T}) - \log q_\phi(\mathbf{x}_{1:T}|\mathbf{y}_{1:T}) + \log p(\mathbf{x}_{1:T})$$



Encoder - Variational Posterior

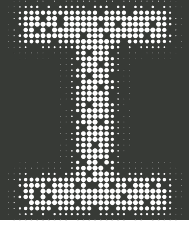
$$\mathcal{F}(\phi, \theta) = \log p_\theta(\mathbf{y}_{1:T}|\mathbf{x}_{1:T}) - \log q_\phi(\mathbf{x}_{1:T}|\mathbf{y}_{1:T}) + \log p(\mathbf{x}_{1:T})$$

- This is the variational distribution, from which $\mathbf{x}_{1:T}$ is sampled.

Encoder - Variational Posterior

$$\mathcal{F}(\phi, \theta) = \log p_\theta(\mathbf{y}_{1:T}|\mathbf{x}_{1:T}) - \log q_\phi(\mathbf{x}_{1:T}|\mathbf{y}_{1:T}) + \log p(\mathbf{x}_{1:T})$$

- This is the variational distribution, from which $\mathbf{x}_{1:T}$ is sampled.
- This is 'inverting' the data generating process, and so should be chosen sensibly to match the decoder.



Encoder - Variational Posterior

$$\mathcal{F}(\phi, \theta) = \log p_\theta(\mathbf{y}_{1:T}|\mathbf{x}_{1:T}) - \log q_\phi(\mathbf{x}_{1:T}|\mathbf{y}_{1:T}) + \log p(\mathbf{x}_{1:T})$$

- This is the variational distribution, from which $\mathbf{x}_{1:T}$ is sampled.
- This is 'inverting' the data generating process, and so should be chosen sensibly to match the decoder.
- Again, in the case of no prior knowledge, we use neural networks.

Encoder - Variational Posterior

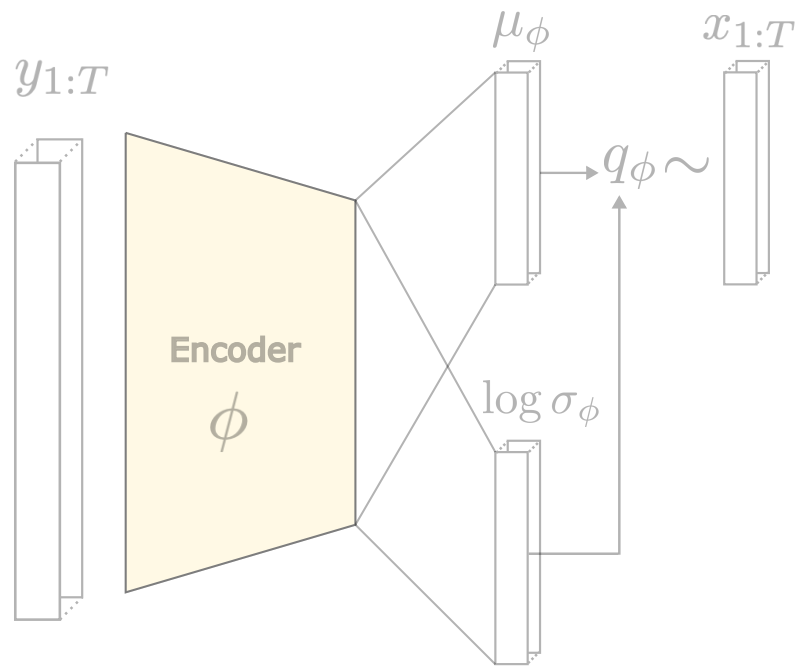
$$\mathcal{F}(\phi, \theta) = \log p_\theta(\mathbf{y}_{1:T}|\mathbf{x}_{1:T}) - \log q_\phi(\mathbf{x}_{1:T}|\mathbf{y}_{1:T}) + \log p(\mathbf{x}_{1:T})$$

With no prior knowledge about data generating process, we use deep nets.

$$p_\theta(\mathbf{y}|\mathbf{x}) = \mathcal{N}(\mu_\theta(\mathbf{x}), \nu^2 I)$$

$$q_\phi(\mathbf{x}|\mathbf{y}) = \mathcal{N}\left(\mu_\phi(\mathbf{y}), \mathrm{diag}(\sigma_\phi(\mathbf{y}))\right)$$

$$\mathbf{x}_{1:T} \sim q_\phi(\mathbf{x}_{1:T}|\mathbf{y}_{1:T})$$



Encoder - Variational Posterior

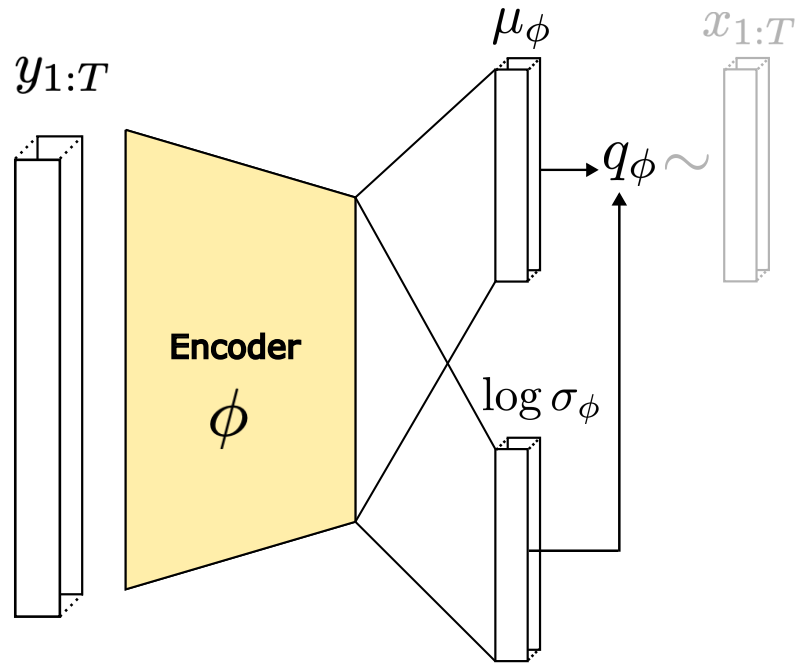
$$\mathcal{F}(\phi, \theta) = \log p_\theta(\mathbf{y}_{1:T}|\mathbf{x}_{1:T}) - \log q_\phi(\mathbf{x}_{1:T}|\mathbf{y}_{1:T}) + \log p(\mathbf{x}_{1:T})$$

With no prior knowledge about data generating process, we use deep nets.

$$p_\theta(\mathbf{y}|\mathbf{x}) = \mathcal{N}(\mu_\theta(\mathbf{x}), \nu^2 I)$$

$$q_\phi(\mathbf{x}|\mathbf{y}) = \mathcal{N}\left(\mu_\phi(\mathbf{y}), \mathrm{diag}(\sigma_\phi(\mathbf{y}))\right)$$

$$\mathbf{x}_{1:T} \sim q_\phi(\mathbf{x}_{1:T}|\mathbf{y}_{1:T})$$



Encoder - Variational Posterior

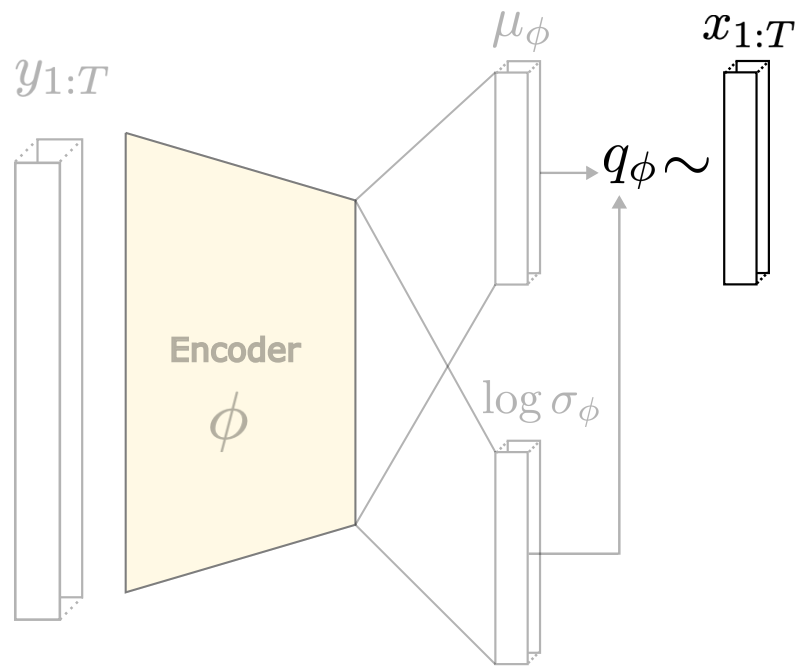
$$\mathcal{F}(\phi, \theta) = \log p_\theta(\mathbf{y}_{1:T}|\mathbf{x}_{1:T}) - \log q_\phi(\mathbf{x}_{1:T}|\mathbf{y}_{1:T}) + \log p(\mathbf{x}_{1:T})$$

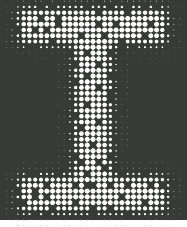
With no prior knowledge about data generating process, we use deep nets.

$$p_\theta(\mathbf{y}|\mathbf{x}) = \mathcal{N}(\mu_\theta(\mathbf{x}), \nu^2 I)$$

$$q_\phi(\mathbf{x}|\mathbf{y}) = \mathcal{N}\left(\mu_\phi(\mathbf{y}), \mathrm{diag}(\sigma_\phi(\mathbf{y}))\right)$$

$$\mathbf{x}_{1:T} \sim q_\phi(\mathbf{x}_{1:T}|\mathbf{y}_{1:T})$$





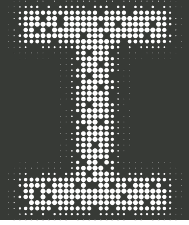
Prior - Log-marginal-likelihood

$$\mathcal{F}(\phi, \theta) = \log p_\theta(\mathbf{y}_{1:T}|\mathbf{x}_{1:T}) - \log q_\phi(\mathbf{x}_{1:T}|\mathbf{y}_{1:T}) + \log p(\mathbf{x}_{1:T})$$

Prior - Log-marginal-likelihood

$$\mathcal{F}(\phi, \theta) = \log p_\theta(\mathbf{y}_{1:T}|\mathbf{x}_{1:T}) - \log q_\phi(\mathbf{x}_{1:T}|\mathbf{y}_{1:T}) + \log p(\mathbf{x}_{1:T})$$

- Physics-informed prior placed on the latent space.



Prior - Log-marginal-likelihood

$$\mathcal{F}(\phi, \theta) = \log p_\theta(\mathbf{y}_{1:T}|\mathbf{x}_{1:T}) - \log q_\phi(\mathbf{x}_{1:T}|\mathbf{y}_{1:T}) + \log p(\mathbf{x}_{1:T})$$

- Physics-informed prior placed on the latent space.
- Assume a particular dynamic model, and construct a SSM.

Prior - Log-marginal-likelihood

$$\mathcal{F}(\phi, \theta) = \log p_\theta(\mathbf{y}_{1:T}|\mathbf{x}_{1:T}) - \log q_\phi(\mathbf{x}_{1:T}|\mathbf{y}_{1:T}) + \log p(\mathbf{x}_{1:T})$$

- Physics-informed prior placed on the latent space.
- Assume a particular dynamic model, and construct a SSM.

Nonlinear State-Space Model

$$\begin{aligned} \text{Transition:} \quad & \mathbf{u}_t = f(\mathbf{u}_{t-1}) + \epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0, R) \\ \text{Observation:} \quad & \mathbf{x}_t = h(\mathbf{u}_t) + \nu_t, \quad \nu_t \sim \mathcal{N}(0, Q) \end{aligned}$$

Prior - Log-marginal-likelihood

$$\mathcal{F}(\phi, \theta) = \log p_\theta(\mathbf{y}_{1:T}|\mathbf{x}_{1:T}) - \log q_\phi(\mathbf{x}_{1:T}|\mathbf{y}_{1:T}) + \log p(\mathbf{x}_{1:T})$$

- Physics-informed prior placed on the latent space.
- Assume a particular dynamic model, and construct a SSM.

Nonlinear State-Space Model

$$\begin{aligned} \text{Transition:} \quad & \mathbf{u}_t = f(\mathbf{u}_{t-1}) + \epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0, R) \\ \text{Observation:} \quad & \mathbf{x}_t = h(\mathbf{u}_t) + \nu_t, \quad \nu_t \sim \mathcal{N}(0, Q) \end{aligned}$$

How do we set the state-space model?

Transition Model - $p(\mathbf{u}_n|\mathbf{u}_{n-1}, \Lambda)$

$$\text{PDE} \quad \partial_t u + L_\Lambda u + \mathcal{F}_\Lambda(u) = f + \epsilon, \quad \epsilon \sim \mathcal{GP}(\delta(t-t')k(x,x'))$$

$$\text{ODE/SDE} \quad \mathrm{d}\mathbf{u}_t = f(\mathbf{u}, t)\mathrm{d}t + L\mathrm{d}\mathbf{w}_t$$

- Statistical Finite Element Method (Girolami et al., 2021; Duffin et al., 2021) used for spatial discretization of PDEs.
- Time-discretization then applied to give a probabilistic state transition.

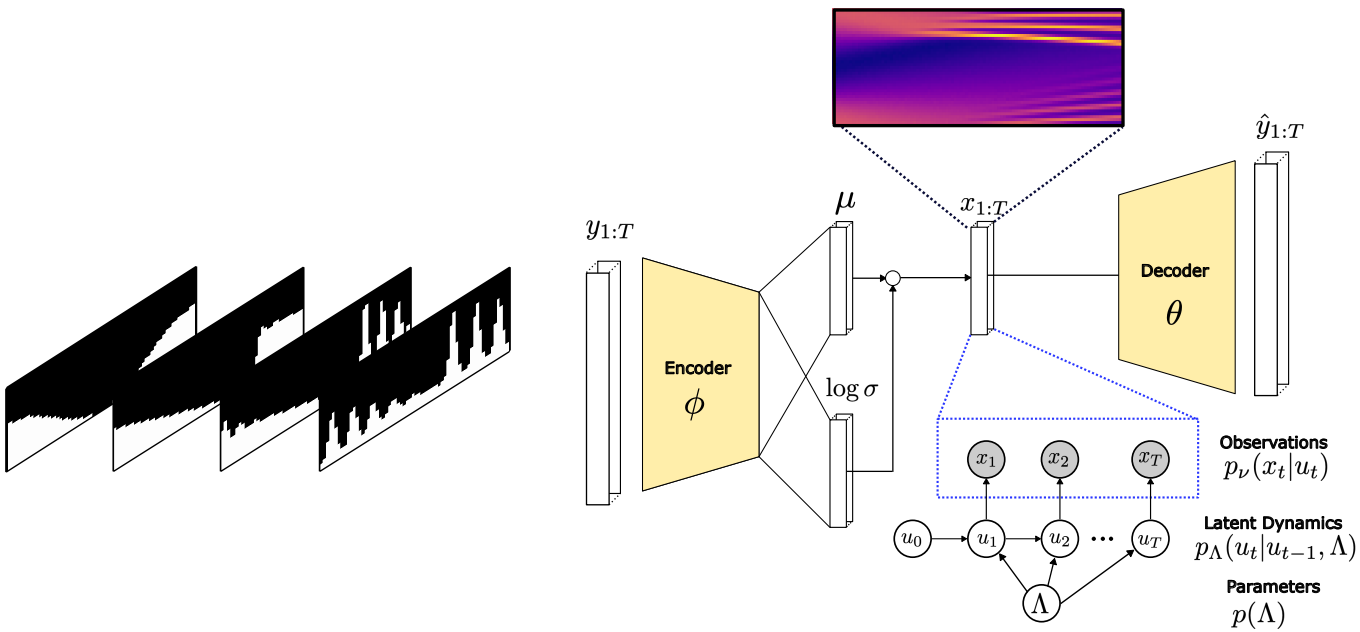
Observation Model - $p(\mathbf{x}_n|\mathbf{u}_n)$

$$\mathbf{x}_t = \mathbf{H}\mathbf{u}_t + \nu_t, \quad \nu_t \sim \mathcal{N}(0, Q)$$

- Extended Kalman Filter used to recursively calculate the marginal-likelihood $\log p(\mathbf{x}_{1:T}) = \log p(\mathbf{x}_1) + \sum_{n=2}^{T} \log p(\mathbf{x}_n|\mathbf{x}_{1:n-1})$.

Collecting all together

Big picture



Parameter Inference

Inferring the dynamics

- We can extend this methodology to deal with unknown parameters.

Parameter Inference

Inferring the dynamics

- We can extend this methodology to deal with unknown parameters.
- Scenario where we posit certain underlying process, but wish to infer parameters purely from video data.

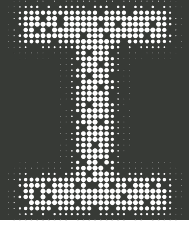
Parameter Inference

Inferring the dynamics

- We can extend this methodology to deal with unknown parameters.
- Scenario where we posit certain underlying process, but wish to infer parameters purely from video data.
- Prior over parameters $p(\Lambda)$, and latent transitions are conditioned on the parameters $p_\Lambda(\mathbf{u}_n|\mathbf{u}_{n-1}, \Lambda)$.

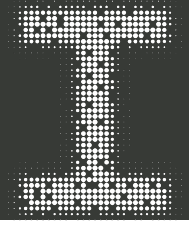
MAP Estimation

Inferring the dynamics



MAP Estimation

Inferring the dynamics



MAP Estimation:

$$\mathcal{F}(\theta, \phi, \Lambda) = \log \frac{p_\theta(\mathbf{y}_{1:T}|\mathbf{x}_{1:T})}{q_\phi(\mathbf{x}_{1:T}|\mathbf{y}_{1:T})} + \log p(\mathbf{x}_{1:T}|\Lambda) + \log p(\Lambda)$$

MAP Estimation

Inferring the dynamics

MAP Estimation:

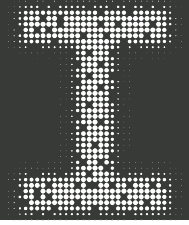
$$\mathcal{F}(\theta, \phi, \Lambda) = \log \frac{p_\theta(\mathbf{y}_{1:T}|\mathbf{x}_{1:T})}{q_\phi(\mathbf{x}_{1:T}|\mathbf{y}_{1:T})} + \log p(\mathbf{x}_{1:T}|\Lambda) + \log p(\Lambda)$$

Use gradient ascent to update the value of Λ at each epoch.

$$\Lambda \leftarrow \Lambda + \gamma \nabla_\Lambda \left\{\log p(\mathbf{x}_{1:T}|\Lambda) + \log p(\Lambda)\right\}$$

Variational Approach

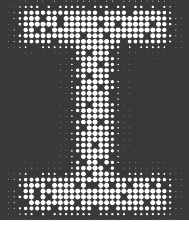
Inferring the dynamics



Variational Approximation, $\Lambda^{(i)} \sim q_\lambda(\Lambda)$, gives a new ELBO:

Variational Approach

Inferring the dynamics



Variational Approximation, $\Lambda^{(i)} \sim q_\lambda(\Lambda)$, gives a new ELBO:

$$\text{ELBO} = \mathbb{E}_{q_\phi(\mathbf{x}_{1:T}|\mathbf{y}_{1:T})}\left[\log \frac{p_\theta(\mathbf{y}_{1:T}|\mathbf{x}_{1:T})}{q_\phi(\mathbf{x}_{1:T}|\mathbf{y}_{1:T})} + \mathbb{E}_{q_\lambda(\Lambda)}\left[\log p(\mathbf{x}_{1:T}|\Lambda) + \log \frac{p(\Lambda)}{q_\lambda(\Lambda)}\right]\right]$$

Variational Approach

Inferring the dynamics

Variational Approximation, $\Lambda^{(i)} \sim q_\lambda(\Lambda)$, gives a new ELBO:

$$\text{ELBO} = \mathbb{E}_{q_\phi(\mathbf{x}_{1:T}|\mathbf{y}_{1:T})}\left[\log \frac{p_\theta(\mathbf{y}_{1:T}|\mathbf{x}_{1:T})}{q_\phi(\mathbf{x}_{1:T}|\mathbf{y}_{1:T})} + \mathbb{E}_{q_\lambda(\Lambda)}\left[\log p(\mathbf{x}_{1:T}|\Lambda) + \log \frac{p(\Lambda)}{q_\lambda(\Lambda)}\right]\right]$$

Expectations again approximated via MC to provide the new loss:

$$\mathcal{F}(\theta, \phi, \lambda) = \log \frac{p_\theta(\mathbf{y}_{1:T}|\mathbf{x}_{1:T})}{q_\phi(\mathbf{x}_{1:T}|\mathbf{y}_{1:T})} + \frac{1}{M}\sum_{i=1}^{M}\left[\log p(\mathbf{x}_{1:T}|\Lambda^{(i)})\right] - D_{KL}(q_\lambda(\Lambda), p(\Lambda))$$

Approximate posterior via marginalisation

One final step

- (Extended) Kalman Filter gives us $p(\mathbf{u}_t|\mathbf{x}_{1:t})$, conditioned on a sample from $q_\phi(\cdot|\mathbf{y}_{1:T})$.

Approximate posterior via marginalisation

One final step

- (Extended) Kalman Filter gives us $p(\mathbf{u}_t|\mathbf{x}_{1:t})$, conditioned on a sample from $q_\phi(\cdot|\mathbf{y}_{1:T})$.
- We can account for the uncertainty of the variational encoding through marginalisation:

Approximate posterior via marginalisation

One final step

- (Extended) Kalman Filter gives us $p(\mathbf{u}_t|\mathbf{x}_{1:t})$, conditioned on a sample from $q_\phi(\cdot|\mathbf{y}_{1:T})$.
- We can account for the uncertainty of the variational encoding through marginalisation:

$$
\begin{aligned}
p(\mathbf{u}_t|\mathbf{y}_{1:t}) &= \int p(\mathbf{u}_t, \mathbf{x}_{1:t}|\mathbf{y}_{1:t})\mathrm{d}\mathbf{x}_{1:t} = \int p(\mathbf{u}_t|\mathbf{x}_{1:t}, \mathbf{y}_{1:t})p(\mathbf{x}_{1:t}|\mathbf{y}_{1:t})\mathrm{d}\mathbf{x}_{1:t} \\
&\approx \int p(\mathbf{u}_t|\mathbf{x}_{1:t})q_\phi(\mathbf{x}_{1:t}|\mathbf{y}_{1:t})\mathrm{d}\mathbf{x}_{1:t} \\
&\approx \frac{1}{M}\sum_{i=1}^{M} p(\mathbf{u}_t|\mathbf{x}_{1:t}^{(i)}), \quad \mathbf{x}_{1:t}^{(i)} \sim q_\phi(\cdot|\mathbf{y}_{1:t})
\end{aligned}
$$

This is a mixture of Gaussians.

Lorenz 63 system

Examples from (Glyn-Davies et al., 2022)

Derived via a truncated decomposition of a PDE describing fluid convection between two plates held at a constant temperature difference.

The 2D stream function $\psi(x, z)$ describes the flow assumed to be parallel to the plates.

$$y = (-\partial_z \psi, 0, \partial_x \psi)$$

$$a(1 + a^2)^{-1} \kappa^{-1} \psi(x, z) = X\sqrt{2} \sin(\pi x / l) \sin(\pi z / d)$$

We observe the velocity field (y) over a 2D domain.

$$\begin{aligned} \dot{u}_1 &= -\sigma u_1 + \sigma u_2, \\ \dot{u}_2 &= -u_1 u_3 + r u_1 - u_2, \\ \dot{u}_3 &= u_1 u_2 - b u_3 \end{aligned}$$

with parameters $\Lambda = \{\sigma, r, b\}$.

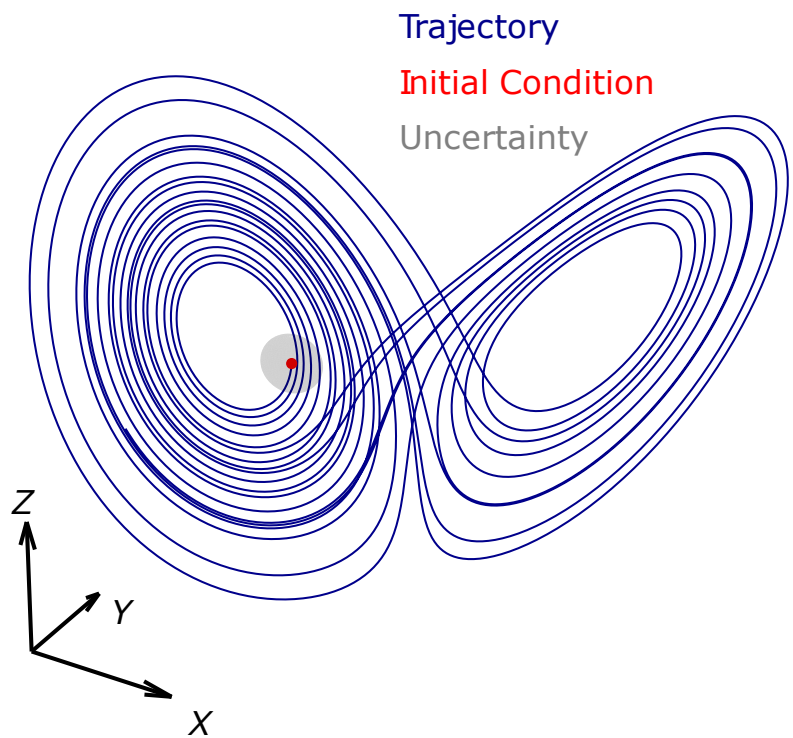
$$y = (-\partial_z \psi, 0, \partial_x \psi)$$

$$a(1 + a^2)^{-1} \kappa^{-1} \psi(x, z) = X\sqrt{2} \sin(\pi x / l) \sin(\pi z / d)$$

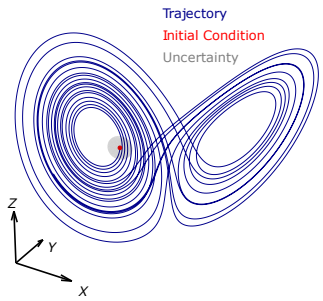
'***In these equations X is proportional to the intensity of the convective motion***'. Lorenz, 1963

We could encode this knowledge about the linear modulation of $\mathbf{y}_{1:T}$ with $\mathbf{x}_{1:T}$ by specifying a linear decoder, and learning this linear mapping.

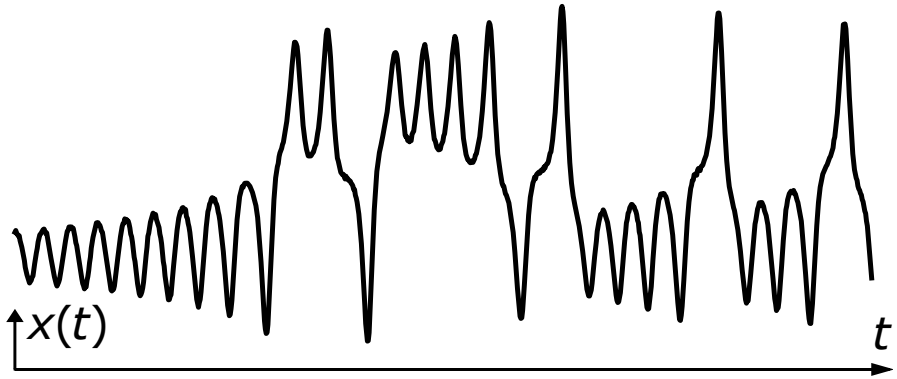
$$\mathbf{u} = \begin{bmatrix} \dot{u}_1 \\ \dot{u}_2 \\ \dot{u}_3 \end{bmatrix} = \begin{bmatrix} -\sigma u_1 + \sigma u_2 \\ -u_1 u_3 + r u_1 - u_2 \\ u_1 u_2 - b u_3 \end{bmatrix}$$



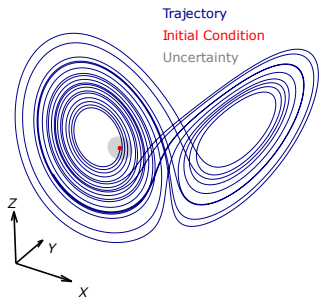
$$\begin{bmatrix} \dot{u}_1 \\ \dot{u}_2 \\ \dot{u}_3 \end{bmatrix} = \begin{bmatrix} -\sigma u_1 + \sigma u_2 \\ -u_1 u_3 + r u_1 - u_2 \\ u_1 u_2 - b u_3 \end{bmatrix}$$



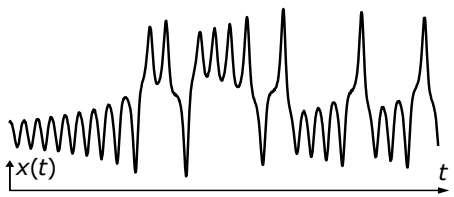
$$\mathbf{x} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma_{\text{obs}})$$



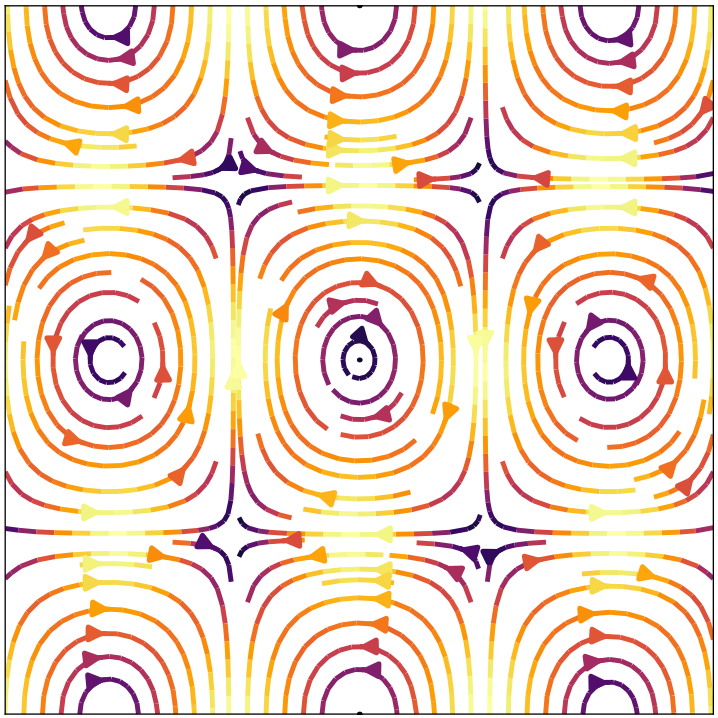
$$\begin{bmatrix} \dot{u}_1 \\ \dot{u}_2 \\ \dot{u}_3 \end{bmatrix} = \begin{bmatrix} -\sigma u_1 + \sigma u_2 \\ -u_1 u_3 + r u_1 - u_2 \\ u_1 u_2 - b u_3 \end{bmatrix}$$



$$\mathbf{x} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} + \epsilon$$

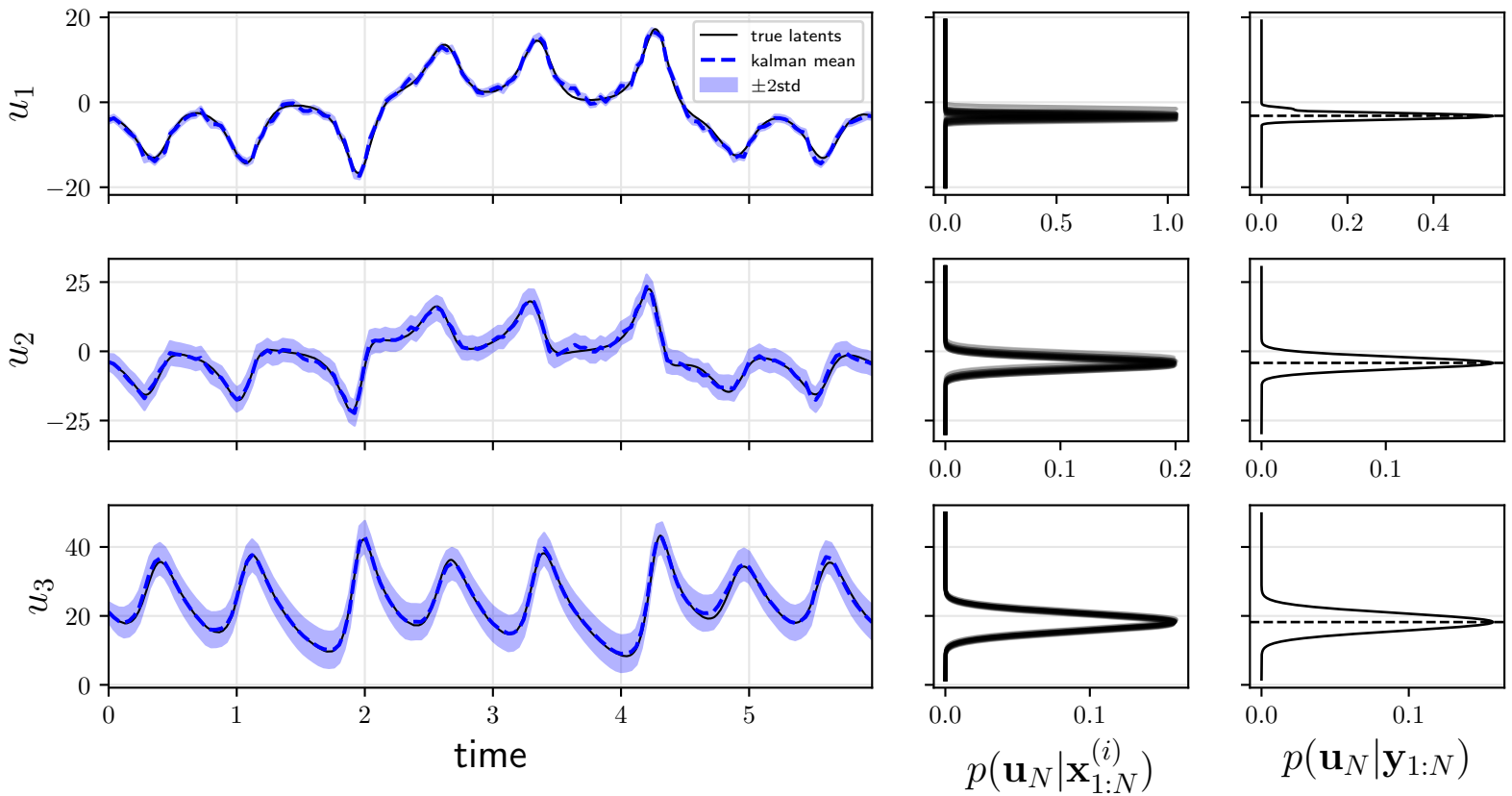


$$y = (-\partial_z \psi, 0, \partial_x \psi)$$



Lorenz 63 system

Examples from (Glyn-Davies et al., 2022)



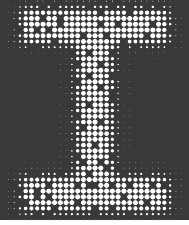
KdV Equation

Examples from (Glyn-Davies et al., 2022)

Korteweg–De Vries (KdV) equation with periodic boundary conditions is used as an example of a nonlinear PDE.

KdV Equation

Examples from (Glyn-Davies et al., 2022)



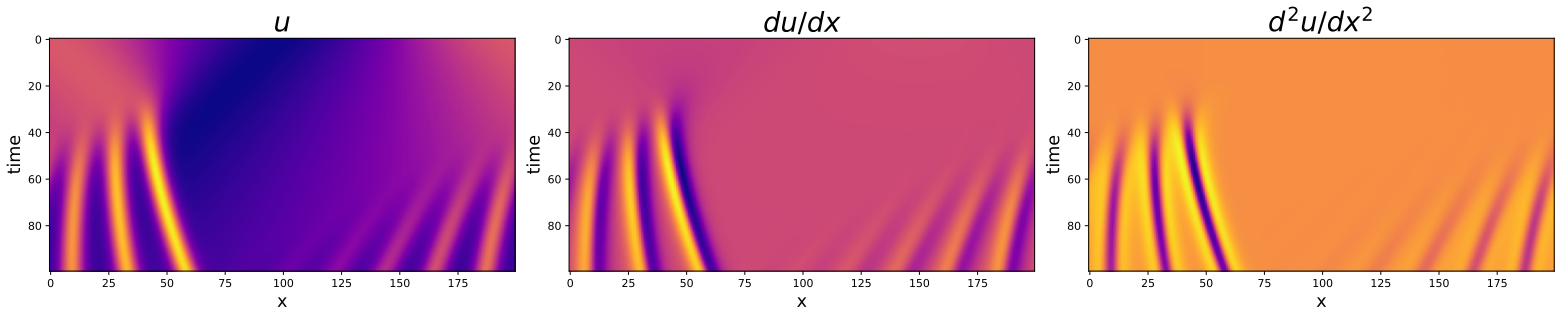
Korteweg–De Vries (KdV) equation with periodic boundary conditions is used as an example of a nonlinear PDE.

KdV Equation

$$\frac{\partial u}{\partial t} + \alpha u \frac{\partial u}{\partial x} + \beta \frac{\partial^3 u}{\partial x^3} = 0$$

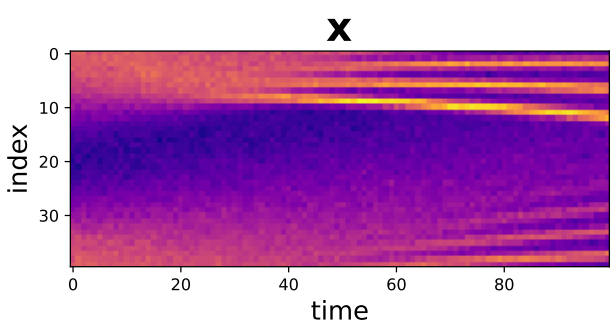
where $t \in [0, T], \quad x \in [0, L], \quad u(x, t) = u(x + L, t).$

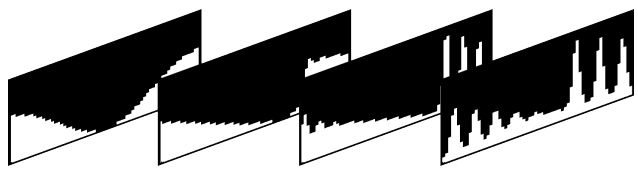
$$\frac{\partial u}{\partial t} + \alpha u \frac{\partial u}{\partial x} + \beta \frac{\partial^3 u}{\partial x^3} = 0$$



$$\mathbf{x} = \mathbf{H}\mathbf{u} + \epsilon$$

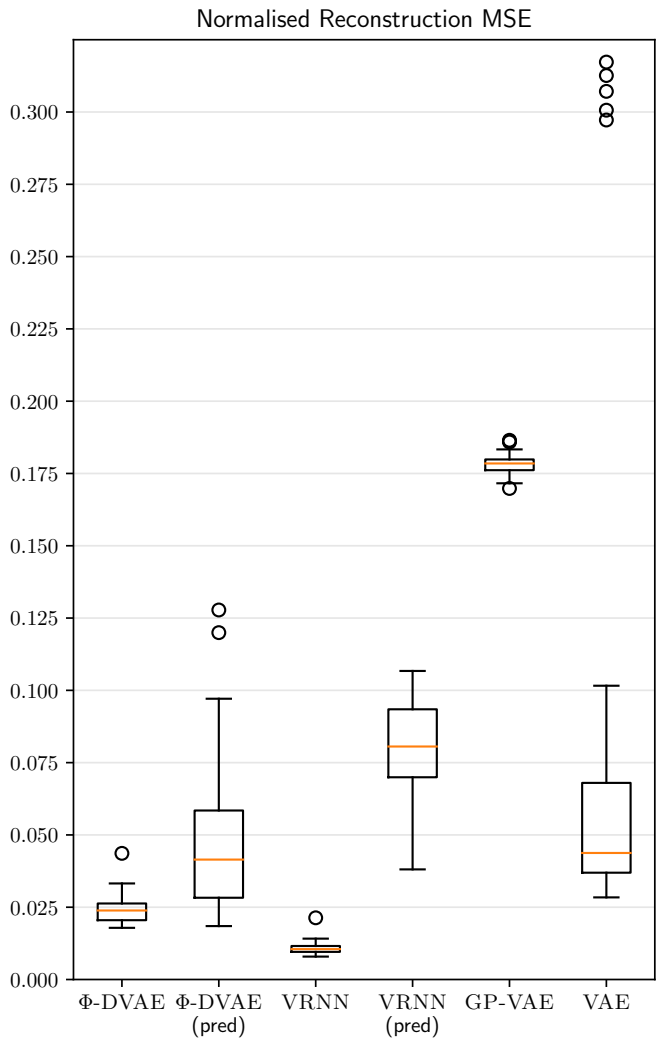
$\mathbf{y}$



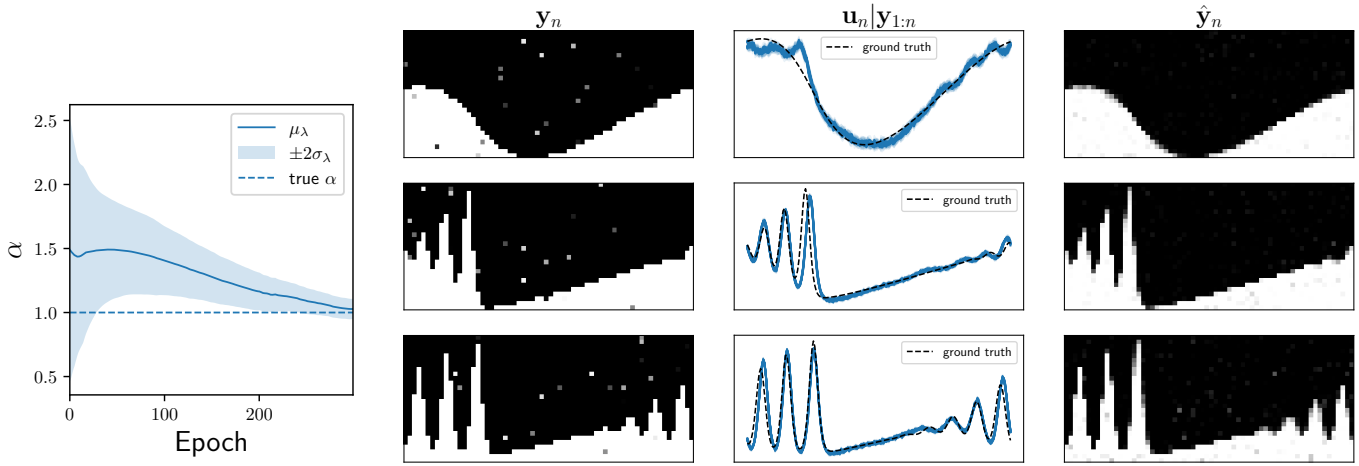


KdV equation

Examples from (Glyn-Davies et al., 2022)



KdV equation



(a) Parameter estimation. (b) Image data. (c) Latent inference. (d) Reconstructions.

Figure: KdV joint inference results; frames shown for $n = \{10, 50, 75\}$.

Parameter estimation

Wavespeed estimation

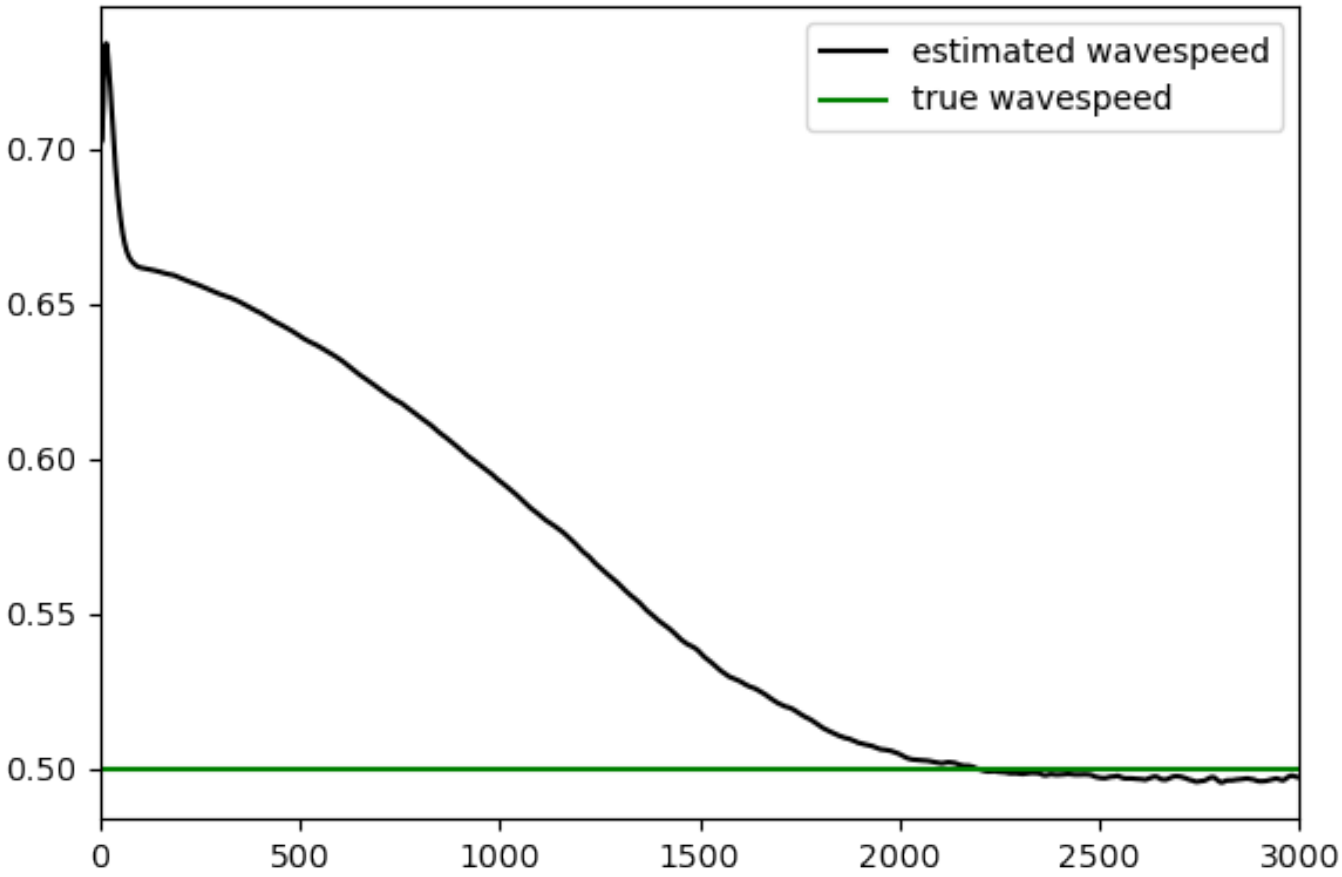


Figure: MAP inference of wavespeed parameter. Initialised at $c = 0.7, p(\Lambda) = \mathcal{N}(0.7, 0.2^2)$.

Parameter estimation

Lorenz 63 example

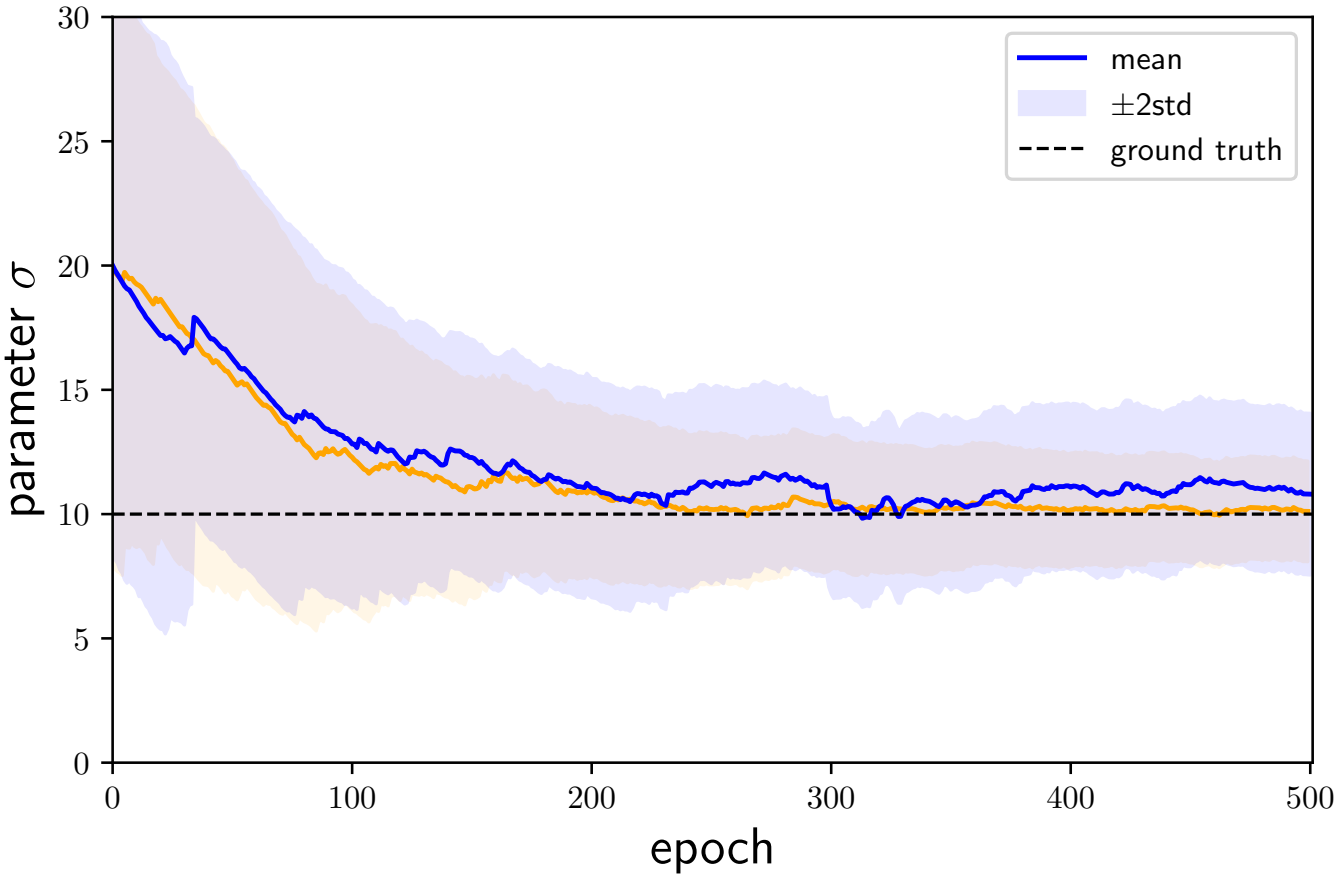


Figure: Variational inference for Lorenz-63 σ parameter. Blue is joint autoencoding and parameter estimation from $\mathbf{y}_{1:T}$, Orange estimates parameter using ground truth observations $\mathbf{x}_{1:T}$.

Questions and discussion?

References I

- Hwang, Chii-Ruey (1980). "Laplace's method revisited: weak convergence of probability measures". In: *The Annals of Probability*, pp. 1177–1182.
- Durmus, Alain and Eric Moulines (2019). "High-dimensional Bayesian inference via the unadjusted Langevin algorithm". In: *Bernoulli* 25.4A, pp. 2854–2882.
- Atchadé, Yves, Keer Jiang, and Yi Sun (2023). *On Generative Energy-Based Models*.
- Nijkamp, Erik, Mitch Hill, Song-Chun Zhu, and Ying Nian Wu (2019). "Learning non-convergent non-persistent short-run mcmc toward energy-based model". In: *Advances in Neural Information Processing Systems* 32.
- De Bortoli, Valentin, Alain Durmus, Marcelo Pereyra, and Ana F Vidal (2021). "Efficient stochastic optimisation by unadjusted Langevin Monte Carlo". In: *Statistics and Computing* 31.3, pp. 1–18.

References II

- Hyvärinen, Aapo (2005). "Estimation of non-normalized statistical models by score matching". In: *Journal of Machine Learning Research* 6.4.
- Murphy, Kevin P (2023). *Probabilistic machine learning: Advanced topics*. MIT press.
- Vincent, Pascal (2011). "A connection between score matching and denoising autoencoders". In: *Neural computation* 23.7, pp. 1661–1674.
- Song, Yang and Diederik P Kingma (2021). "How to train your energy-based models". In: *arXiv preprint arXiv:2101.03288*.
- Song, Yang, Sahaj Garg, Jiaxin Shi, and Stefano Ermon (2020). "Sliced score matching: A scalable approach to density and score estimation". In: *Uncertainty in Artificial Intelligence*. PMLR, pp. 574–584.
- Song, Yang and Stefano Ermon (2019). "Generative modeling by estimating gradients of the data distribution". In: *Advances in neural information processing systems* 32.

References III

- Bernstein, Matthew N. (Mar. 2023). *Variational Autoencoders*. https://mbernste.github.io/posts/vae/. Accessed: 2024-04-21.
- Fraccaro, Marco, Simon Kamronn, Ulrich Paquet, and Ole Winther (2017). "A disentangled recognition and nonlinear dynamics model for unsupervised learning". In: *Advances in Neural Information Processing Systems*, pp. 3601–3610.
- Girolami, Mark, Eky Febrianto, Ge Yin, and Fehmi Cirak (2021). "The statistical finite element method (statFEM) for coherent synthesis of observation data and model predictions". In: *Computer Methods in Applied Mechanics and Engineering* 375, p. 113533.
- Duffin, Connor, Edward Cripps, Thomas Stemler, and Mark Girolami (2021). "Statistical finite elements for misspecified models". In: *Proceedings of the National Academy of Sciences* 118.2.

References IV

- Glyn-Davies, Alex John, Connor Duffin, Omer Deniz Akyildiz, and Mark Girolami (2022). "Φ-DVAE: Physics-Informed Dynamical Variational Autoencoders for Unstructured Data Assimilation". In.
- Lorenz, Edward N (1963). "Deterministic nonperiodic flow". In: *Journal of atmospheric sciences* 20.2, pp. 130–141.