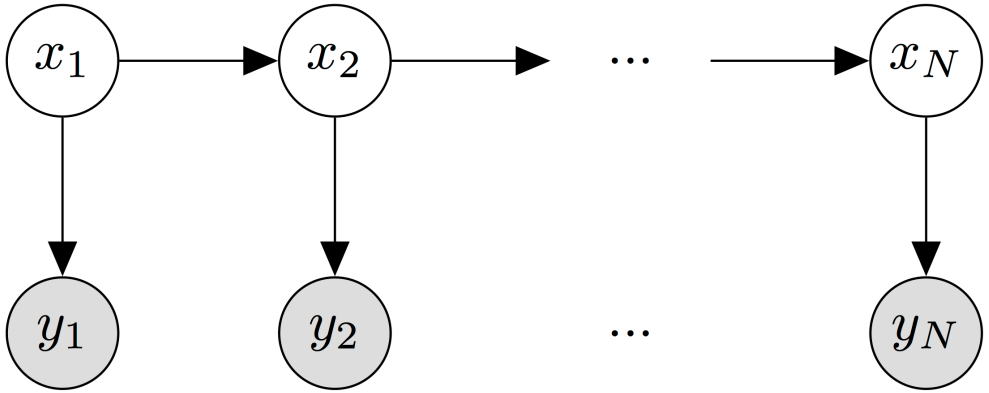
Adversarial Sequential Monte Carlo

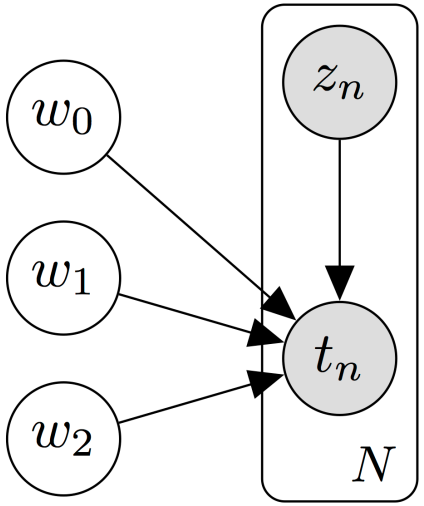
Kira Kempinska, John Shawe-Taylor

NIPS 2017 Advances in Approximate Bayesian Inference

UCL SECReT
SECURITY SCIENCE DOCTORAL RESEARCH TRAINING CENTRE

Directed graphical models

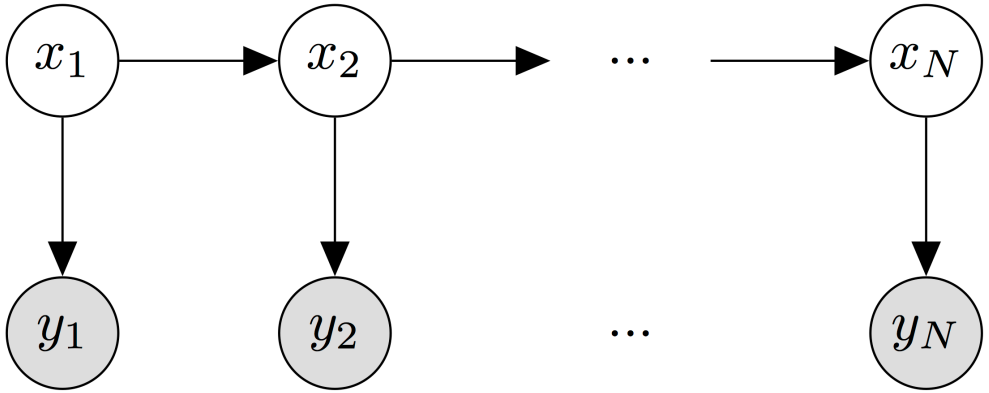


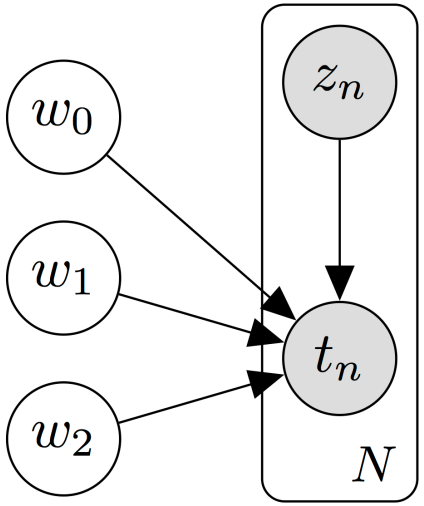


$$p(x, y) = p(x)p(y|x)$$

Inference $\boldsymbol{p(x|y)}$

Directed graphical models





$$p(x, y) = p(x)p(y|x)$$

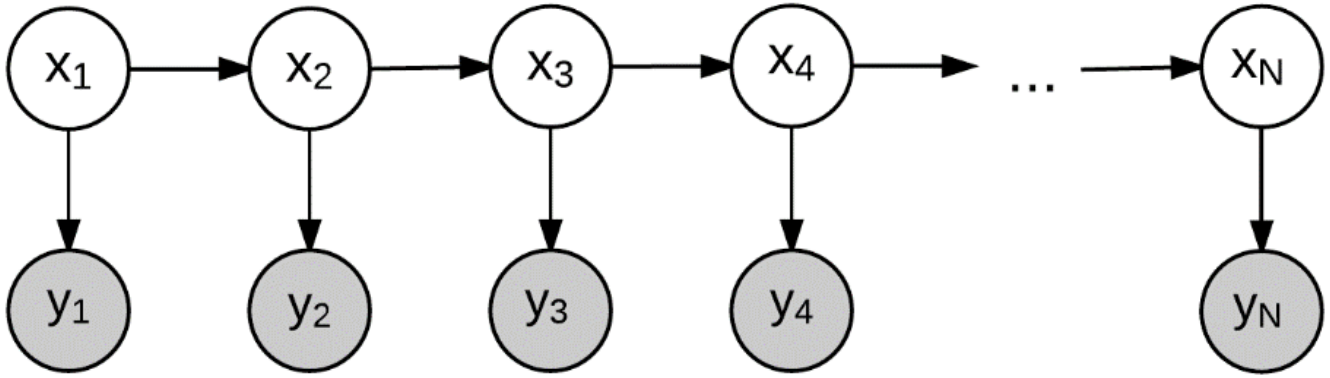
Inference $\boldsymbol{p(x|y)}$

Intractable for most models!

Sequential Monte Carlo methods

- Consider a graphical model with *N*-dimensional latent space that factorises according to:

$$p(x_{1:N}, y_{1:N}) = p(x_1)p(y_1|x_1)\prod_{n=2}^{N} p(x_n|x_{1:n-1})p(y_n|x_{1:n}, y_{1:n-1})$$

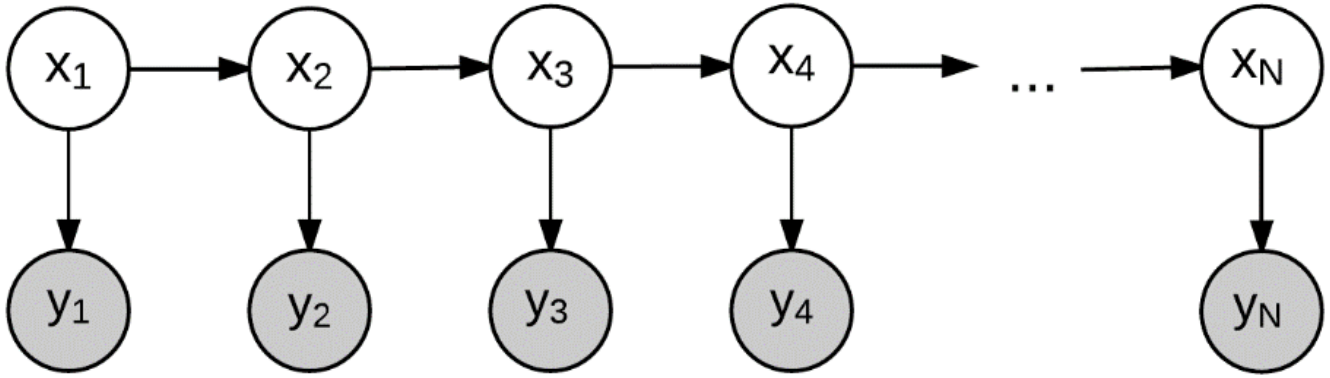


$\boldsymbol{p(x_{1:N}|y_{1:N})}$ **?**

Sequential Monte Carlo methods

- Consider a graphical model with *N*-dimensional latent space that factorises according to:

$$p(x_{1:N}, y_{1:N}) = p(x_1)p(y_1|x_1)\prod_{n=2}^{N} p(x_n|x_{1:n-1})p(y_n|x_{1:n}, y_{1:n-1})$$



$\boldsymbol{p(x_{1:N}|y_{1:N})}$ **?**

$\sim q(x_{1:N}|y_{1:N})$ proposal distribution

Sequential Monte Carlo

Idea: approximate $p(x_{1:N}|y_{1:N})$ with a weighted set of samples.

1. Sample *K* particles from (presumably simpler) proposal distribution:

   $p(x_{1:N}|y_{1:N}) \sim q(x_{1:N}|y_{1:N})$

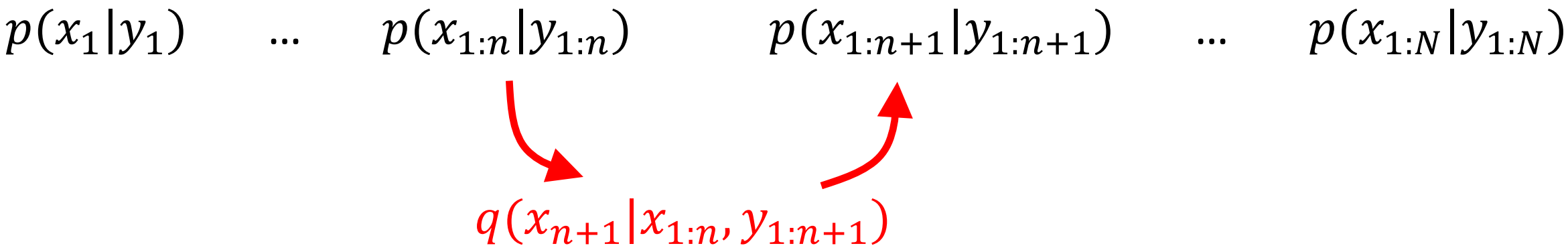
2. Weigh the particles according to the *unnormalised* weight function:

   $$w(x_{1:N}) = \frac{p(x_{1:N},y_{1:N})}{q(x_{1:N}|y_{1:N})}$$

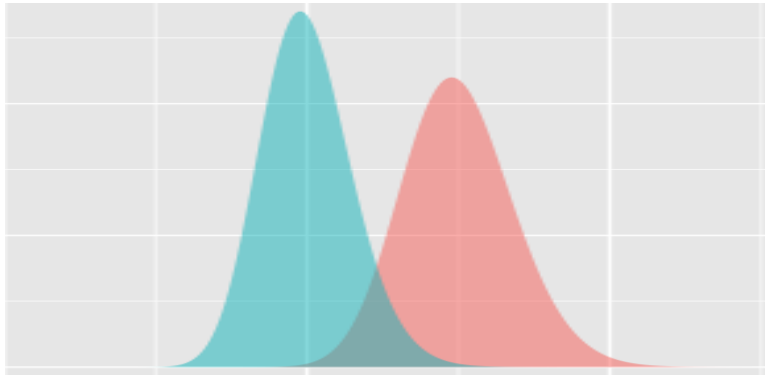
3. Resample the particles (with replacement) according to their weights.

Sequential Monte Carlo

Trick: break down the approximation of $p(x_{1:N}|y_{1:N})$ into a series of intermediate target distributions.



Sequential Monte Carlo

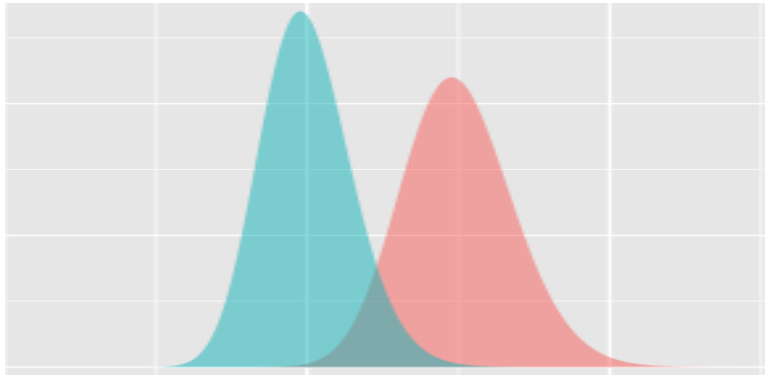


- The optimal proposal distribution is given by

$$q_{opt}(x_n|x_{1:n-1}, y_{1:n}) \propto p(x_n|x_{1:n-1})p(y_n|x_{1:n}, y_{1:n-1})$$

- In practice, the optimal proposal is almost always intractable.
- **The performance of particle filters crucially depends on the quality of the proposal!**

Sequential Monte Carlo



- The optimal proposal distribution is given by

$$q_{opt}(x_n|x_{1:n-1}, y_{1:n}) \propto p(x_n|x_{1:n-1})p(y_n|x_{1:n}, y_{1:n-1})$$

- In practice, the optimal proposal is almost always intractable.
- **The performance of particle filters crucially depends on the quality of the proposal!**

- Bootstrap Particle Filters (Gordon et al., 1993):

$$q_{BPF}(x_n|x_{1:n-1}, y_{1:n}) = \mathrm{p}(x_n|x_{1:n-1})$$

Adversarial Sequential Monte Carlo

- Optimal proposal (reminder):

$$q(x_n|x_{1:n-1}, y_{1:n}) \propto p(x_n|x_{1:n-1})p(y_n|x_{1:n}, y_{1:n-1})$$

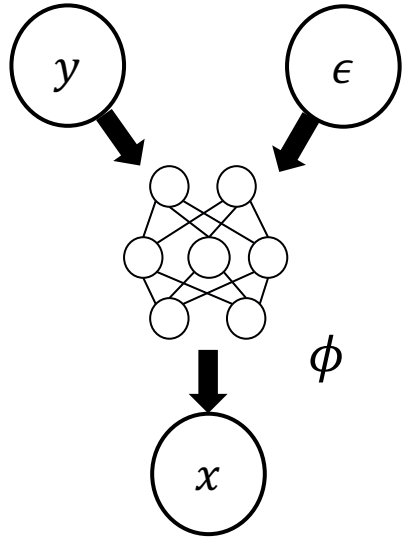
Adversarial Sequential Monte Carlo

- Optimal proposal (reminder):

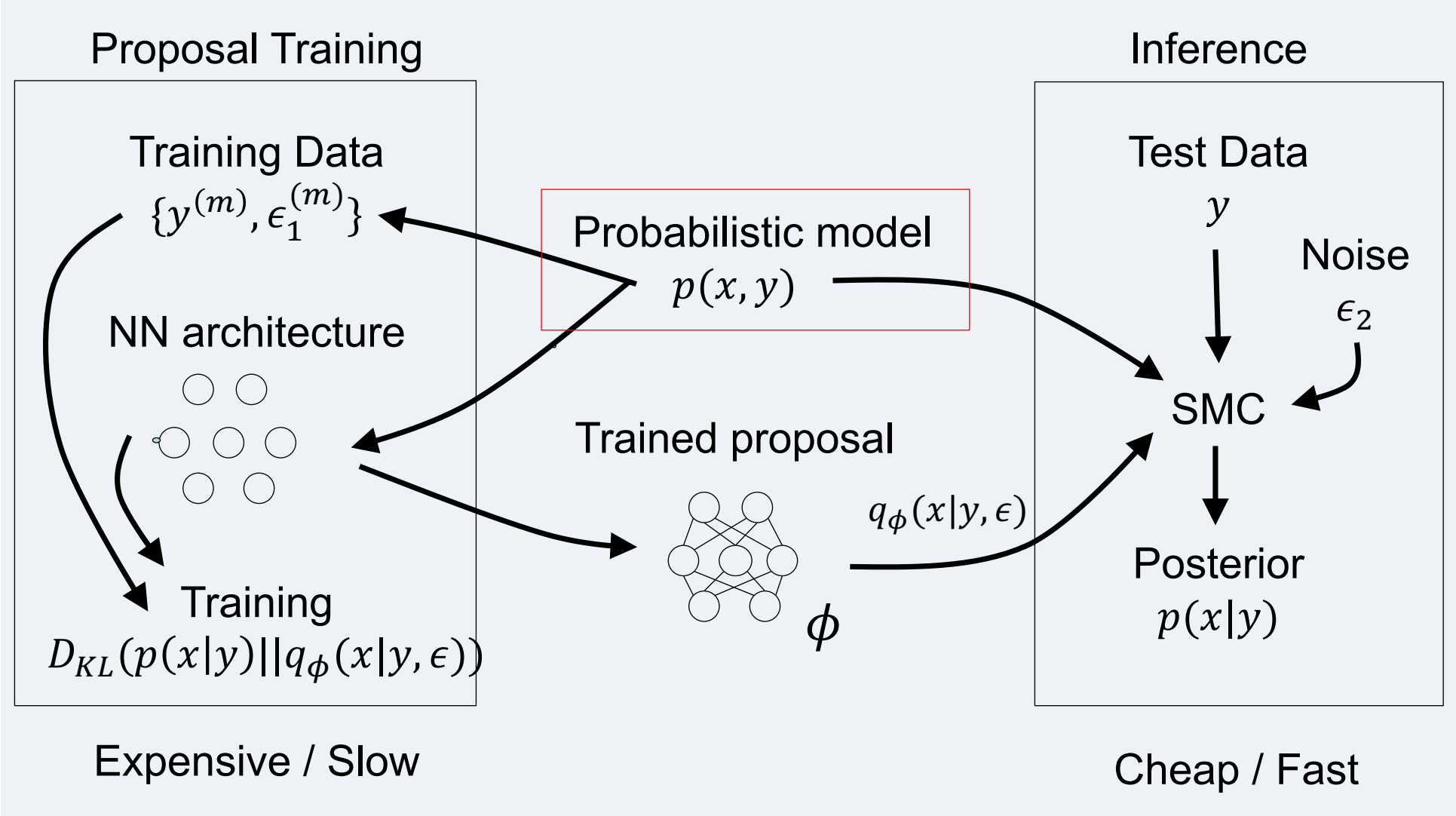
$$q(x_n|x_{1:n-1}, y_{1:n}) \propto p(x_n|x_{1:n-1})p(y_n|x_{1:n}, y_{1:n-1})$$

- **We train highly flexible proposals by representing them as implicit models.**

- Implicit models use an input noise ϵ and a deep network architecture to flexibly characterise a wide range of proposal distributions.



Adversarial Sequential Monte Carlo



Adversarial Proposal Training

- We want to learn parameters ϕ of an implicit proposal distribution q_ϕ that make it as close as possible to the true posterior:

$$\min_\phi KL[q_\phi(x_{1:N}|y_{1:N})||p(x_{1:N}|y_{1:N})] = \min_\phi E_{q_\phi} \log \frac{q_\phi(x_{1:N}|y_{1:N})}{p(x_{1:N}|y_{1:N})}$$

$$= \min_\phi E_{q_\phi}[\log \frac{q_\phi(x_{1:N}|y_{1:N})}{p(x_{1:N})} - \log p(y_{1:N}|x_{1:N})] + \log p(y_{1:N}))$$

Proposal q_ϕ is implicit, so we need adversarial training to minimise KL divergence.

Adversarial Proposal Training

- We want to learn parameters ϕ of an implicit proposal distribution q_ϕ that make it as close as possible to the true posterior:

$$\min_\phi KL[q_\phi(x_{1:N}|y_{1:N})||p(x_{1:N}|y_{1:N})] = \min_\phi E_{q_\phi} \log \frac{q_\phi(x_{1:N}|y_{1:N})}{p(x_{1:N}|y_{1:N})}$$
$$= \min_\phi E_{q_\phi}[\log \frac{q_\phi(x_{1:N}|y_{1:N})}{p(x_{1:N})} - \log p(y_{1:N}|x_{1:N})] + \log p(y_{1:N}))$$

Proposal q_ϕ is implicit, so we need adversarial training to minimise KL divergence.

- We can ignore the marginal likelihood $\log p(y_{1:N})$ and factorise the optimisation:

$$\min_{\phi_n} E_{q_{\phi_n}} \left[\log \frac{q_{\phi_n}(x_n|x_{1:n-1}, y_{1:n})}{p(x_n|x_{1:n-1})} - \log p(y_n|x_{1:n}, y_{1:n-1}) \right] \quad \text{for } n = 1, \dots, N$$

Adversarial Proposal Training

- Our training objective:

$$\min_{\phi_n} E_{q_{\phi_n}}\left[\log \frac{q_{\phi_n}(x_n|x_{1:n-1}, y_{1:n})}{p(x_n|x_{1:n-1})} - \log p(y_n|x_{1:n}, y_{1:n-1})\right] \qquad \text{for } n = 1, \ldots, N$$

Adversarial Proposal Training

- Our training objective:

$$\min_{\phi_n} E_{q_{\phi_n}}\left[\log \frac{q_{\phi_n}(x_n|x_{1:n-1}, y_{1:n})}{p(x_n|x_{1:n-1})} - \log p(y_n|x_{1:n}, y_{1:n-1})\right] \qquad \text{for } n = 1, \ldots, N$$

- The above formulation suggests the following adversarial training objectives:

Adversarial Proposal Training

- Our training objective:

$$\min_{\phi_n} E_{q_{\phi_n}} \left[\log \frac{q_{\phi_n}(x_n | x_{1:n-1}, y_{1:n})}{p(x_n | x_{1:n-1})} - \log p(y_n | x_{1:n}, y_{1:n-1}) \right] \qquad \text{for } n = 1, \ldots, N$$

- The above formulation suggests the following adversarial training objectives:

Discriminator:

$$\max_T E_{q_{\phi_n}(x_n | x_{1:n-1}, y_{1:n})} \log \sigma(T(x_{1:n}, y_{1:n})) + E_{p(x_n | x_{1:n-1})} \log(1 - \sigma(T(x_{1:n}, y_{1:n})))$$

Adversarial Proposal Training

- Our training objective:

$$\min_{\phi_n} E_{q_{\phi_n}}\left[\log \frac{q_{\phi_n}(x_n|x_{1:n-1}, y_{1:n})}{p(x_n|x_{1:n-1})} - \log p(y_n|x_{1:n}, y_{1:n-1})\right] \qquad \text{for } n = 1, \ldots, N$$

discriminator

- The above formulation suggests the following adversarial training objectives:

Discriminator:

$$\max_T E_{q_{\phi_n}(x_n|x_{1:n-1},y_{1:n})} \log \sigma(T(x_{1:n}, y_{1:n})) + E_{p(x_n|x_{1:n-1})} \log(1 - \sigma(T(x_{1:n}, y_{1:n})))$$

Optimal discriminator: $T^*(x_{1:n}, y_{1:n}) = \log \frac{q_{\phi_n}(x_n|x_{1:n-1}, y_{1:n})}{p(x_n|x_{1:n-1})}$

Adversarial Proposal Training

- Our training objective:

$$\min_{\phi_n} E_{q_{\phi_n}}\left[\log \frac{q_{\phi_n}(x_n|x_{1:n-1}, y_{1:n})}{p(x_n|x_{1:n-1})} - \log p(y_n|x_{1:n}, y_{1:n-1})\right] \qquad \text{for } n = 1, \ldots, N$$

discriminator **generator**

- The above formulation suggests the following adversarial training objectives:

Discriminator:

$$\max_T E_{q_{\phi_n}(x_n|x_{1:n-1},y_{1:n})} \log \sigma(T(x_{1:n}, y_{1:n})) + E_{p(x_n|x_{1:n-1})} \log(1 - \sigma(T(x_{1:n}, y_{1:n})))$$

Optimal discriminator: $T^*(x_{1:n}, y_{1:n}) = \log \frac{q_{\phi_n}(x_n|x_{1:n-1}, y_{1:n})}{p(x_n|x_{1:n-1})}$

Generator:

$$\min_{\phi_n} E_\epsilon(T^*(x_{1:n-1}, x_{\phi_n}, y_{1:n}) - \log p(y_n|x_{1:n-1}, x_{\phi_n}, y_{1:n-1}))$$

Generator network representing the proposal distribution q_{ϕ_n}

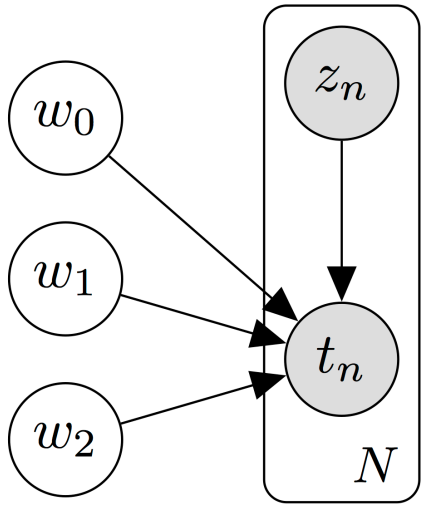
Case Study 1: polynomial regression

Consider a non-conjugate polynomial regression model with global-only latent variables proposed by [5]:

$$p(w_d) = Laplace\left(0,10^{1-d}\right) \quad d = 0,1,2$$

$$p(t_n|w_0, w_1, w_2, z_n) = t_v(w_0 + w_1 z_n + w_2 z_n^2, \epsilon^2)$$

for $n = 1, \ldots, N$, fixed $v = 4$, $\epsilon = 1$ and $z_n \in (-10,10)$ uniformly.



Our goal is to infer latent variables $\boldsymbol{x} \equiv \{\boldsymbol{w_0}, \boldsymbol{w_1}, \boldsymbol{w_2}\}$ given observed variables $\boldsymbol{y} \equiv \{\boldsymbol{z_n}, \boldsymbol{y_n}\}_{\boldsymbol{n=1}}^{\boldsymbol{N}}$.

Case Study 1: polynomial regression

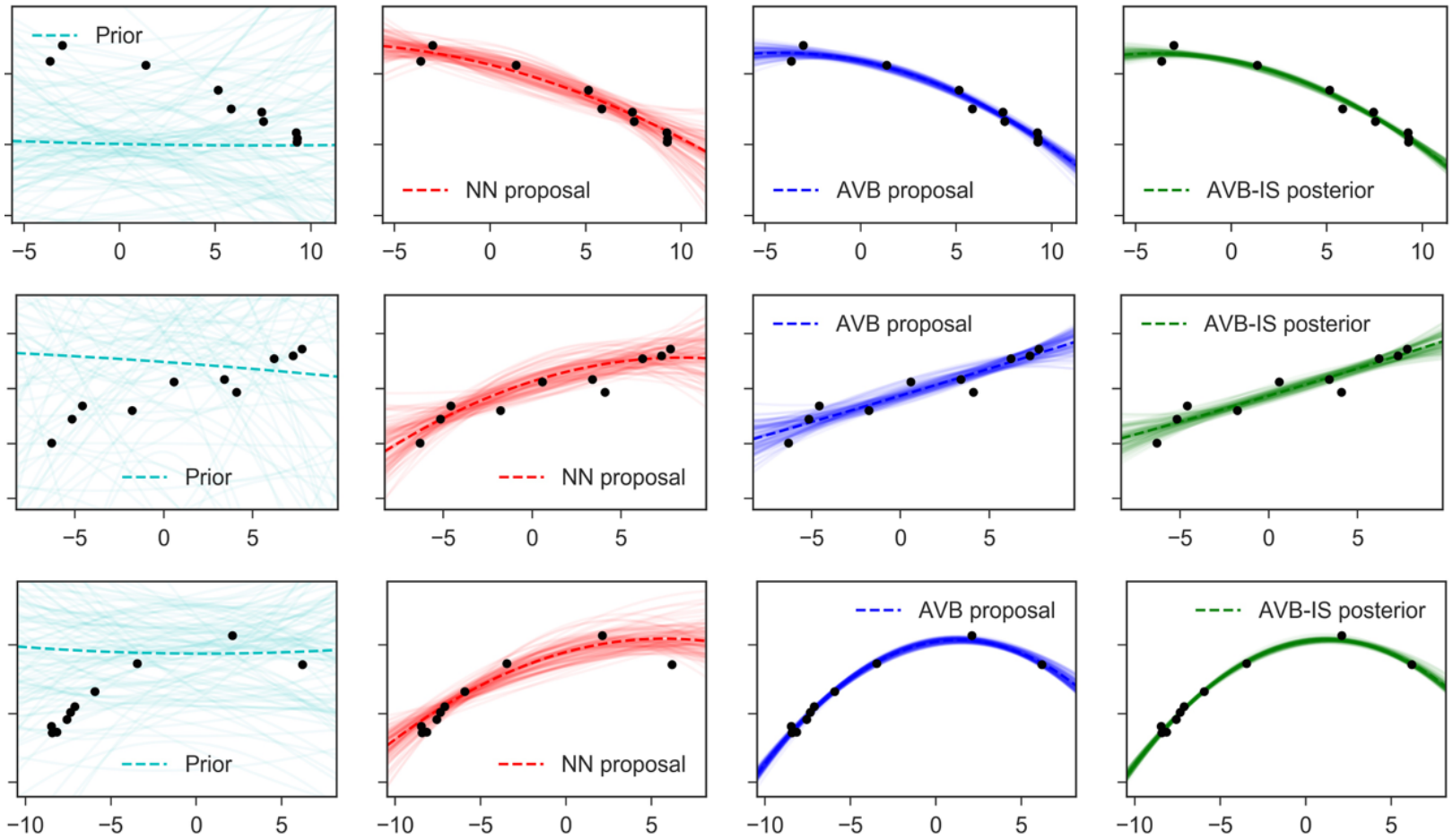


Figure: Exemplary output in the polynomial regression case study.

Plots show 100 regression curves, each curve estimated from one sample of weights, and the mean curve marked as a dashed line.

Black dots represent true observations.

Case Study 2: benchmark non-linear state-space model

We now consider a more complex temporal model given by the following dynamics:

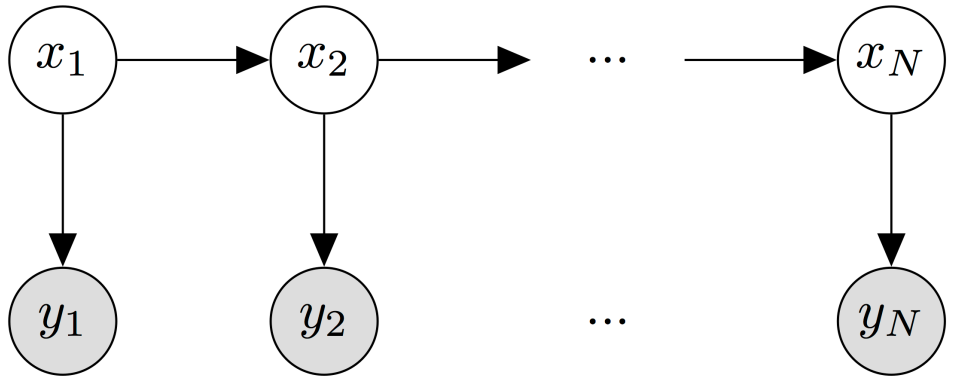
$$p(x_n|x_{n-1}) = \mathbb{N}(x_n; f(x_{n-1}, \sigma_v^2), \qquad p(x_1) = \mathbb{N}(\mathrm{x}_1; 0{,}5)$$

$$p(y_n|x_n) = \mathbb{N}(y_n; g(x_n), \sigma_w^2)$$

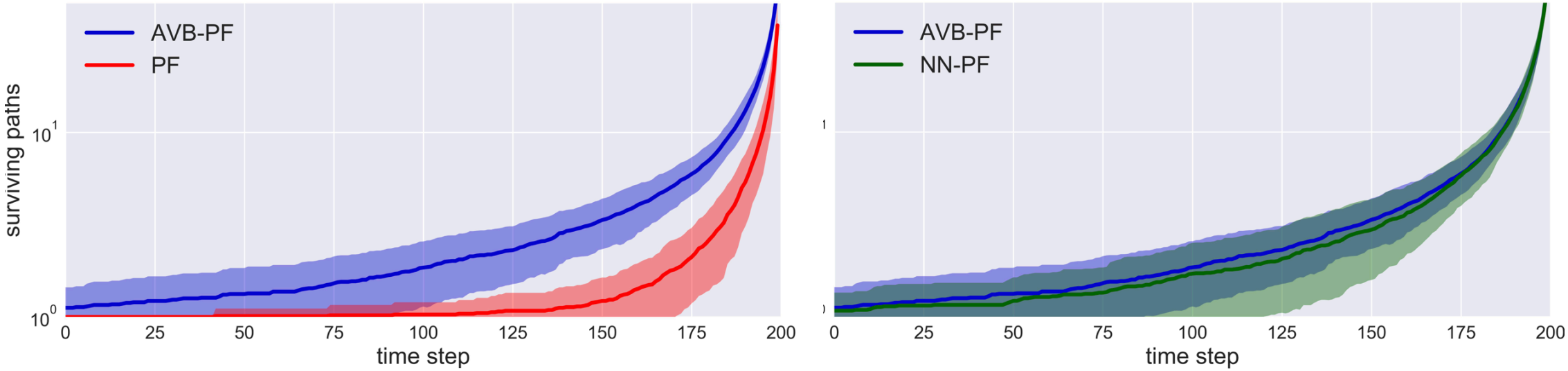
$$f(x_{n-1}) = x_{n-1}/2 + 25x_{n-1}/(1 + x_{n-1}^2), \qquad g(x_n) = x_n^2/20$$

for fixed $\theta = (\sigma_v, \sigma_w) = (\sqrt{10}, 1)$.

Our goal is to infer a sequence of true states $x_{1:N}$ from a sequence of observations $y_{1:N}$.



Case Study 2: benchmark non-linear state-space model



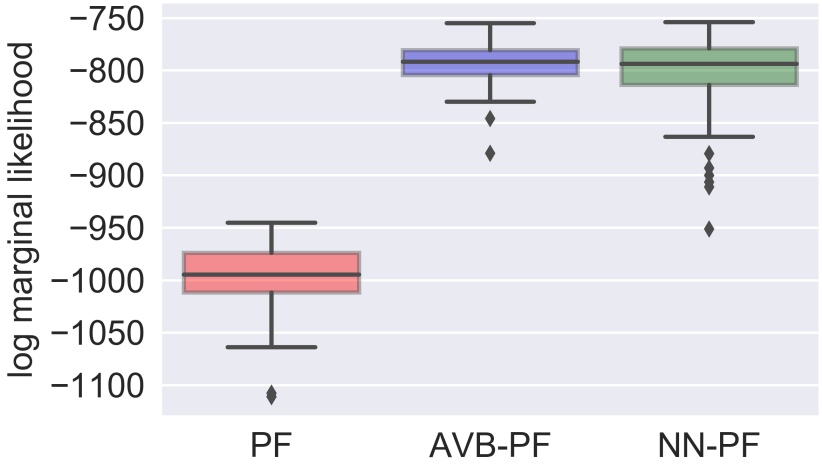


Figure: Adversarially learnt proposals reduce particle degeneracy and explain data better. Plots show mean and standard deviation of the number of unique particles (top) and log-marginal likelihood (bottom) over 100 sampled sequences of length $N = 200$, using 100 particles.

Key references

[1] Christophe Andrieu, Arnaud Doucet, and Roman Holenstein. Particle markov chain monte carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(3):269–342, 2010.

[2] Arnaud Doucet, Nando De Freitas, and NJ Gordon. Sequential monte carlo methods in practice. series statistics for engineering and information science, 2001.

[3] Arnaud Doucet and Adam M Johansen. A tutorial on particle filtering and smoothing: Fifteen years later. *Handbook of nonlinear filtering*, 12(656-704):3, 2009.

[4] Lars Mescheder, Sebastian Nowozin, and Andreas Geiger. Adversarial variational bayes: Unifying variational autoencoders and generative adversarial networks. *arXiv preprint arXiv:1701.04722*, 2017.

[5] Brooks Paige and Frank Wood. Inference networks for sequential monte carlo in graphical models. In *International Conference on Machine Learning*, pages 3040–3049, 2016.

Thank You!

Any questions?