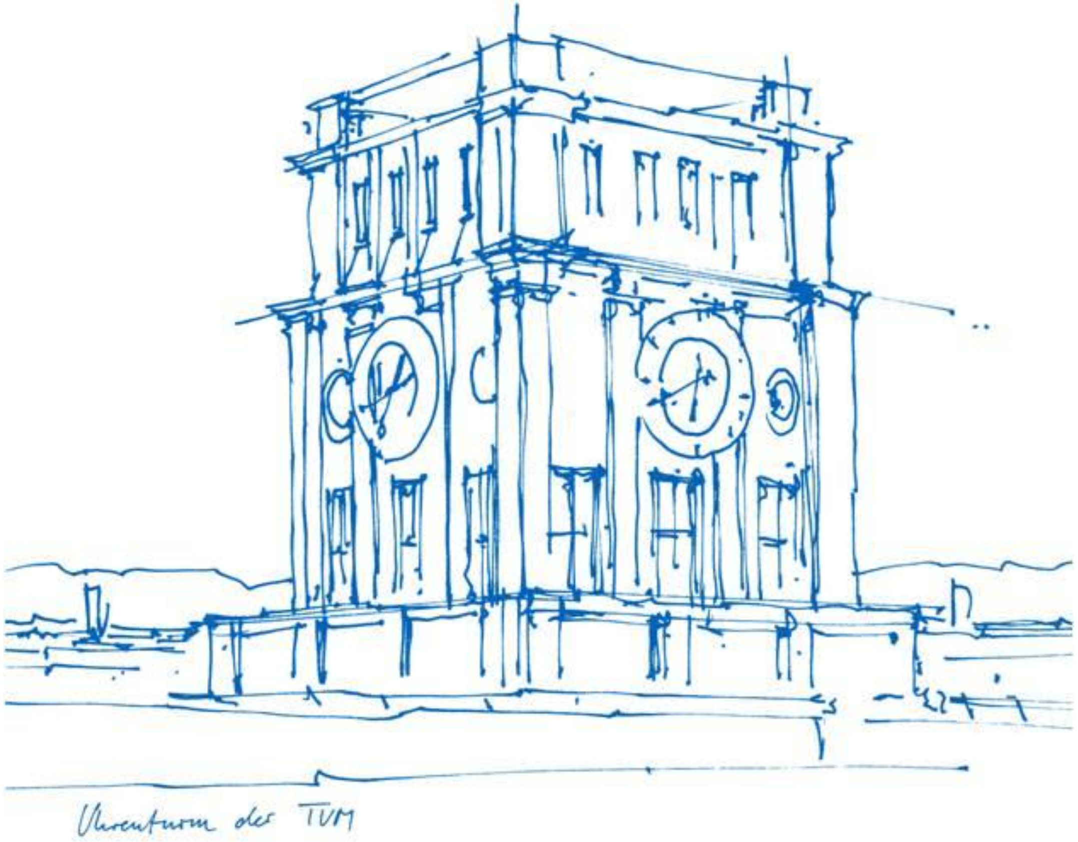
Bayesian Flow Networks

Data Analytics and Machine Learning

Jed Guzelkabaagac

Supervised by Marcel Kollovieh



Diffusion Models

Diffusion Models

Diffusion models typically operate directly on data samples x

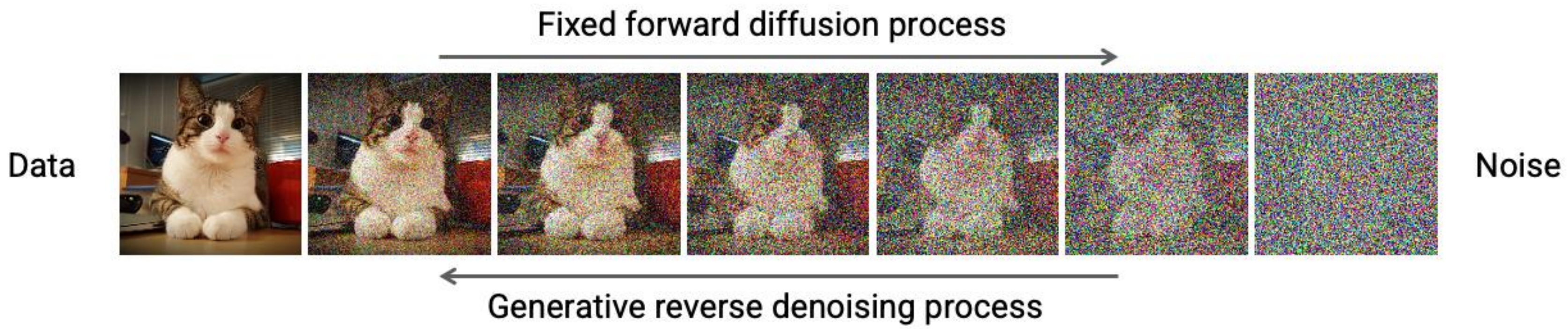
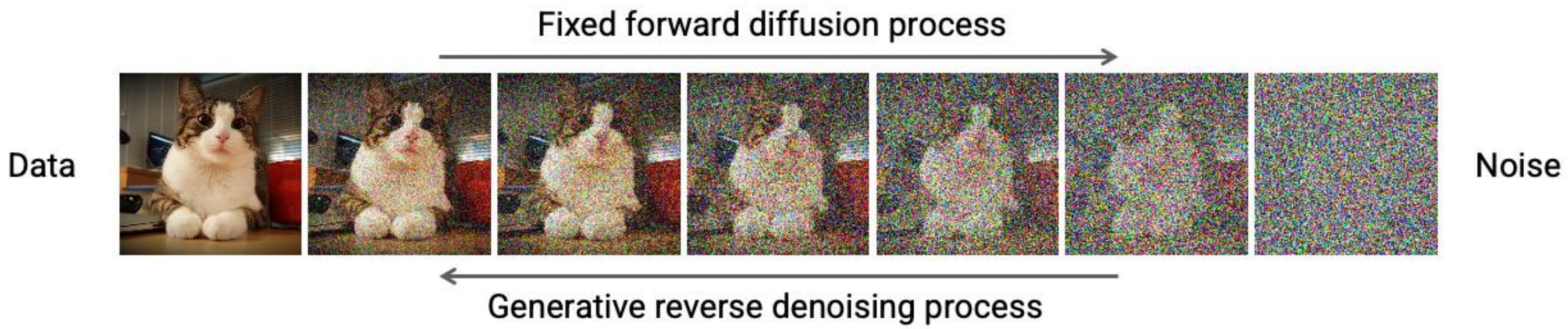


Image: *CVPR 2022 Tutorial Denoising Diffusion-based Generative Modeling: Foundations and Applications, Speakers: Arash Vahdat, Karsten Kreis, & Ruiqi Gao*

Diffusion Models

Diffusion models typically operate directly on data samples x



For discrete data, this leads to an **uninformative loss function**, impeding optimization.

$$\mathcal{L}_{\text{diffusion}} = \mathbb{E}_{x,\,\epsilon}\left[\|\epsilon - \epsilon_\theta(x_t, t)\|\right]$$

Image: *CVPR 2022 Tutorial Denoising Diffusion-based Generative Modeling: Foundations and Applications, Speakers: Arash Vahdat, Karsten Kreis, & Ruiqi Gao*

Bayesian Flow Networks

A unified differentiable generative framework

Bayesian Flow Networks

A unified differentiable generative framework

✓ Natively handles discrete data

Bayesian Flow Networks

A unified differentiable generative framework

✓ Natively handles discrete data

✓ Great performance on ‘hybrid data’ tasks

Bayesian Flow Networks

A unified differentiable generative framework

✓ Natively handles discrete data

✓ Great performance on ‘hybrid data’ tasks

✓ No corruption process needed – start with a simple prior

Bayesian Flow Networks

A unified differentiable generative framework

- ✓ Natively handles discrete data
- ✓ Great performance on ‘hybrid data’ tasks
- ✓ No corruption process needed – start with a simple prior
- ✓ ‘Bayesian perspective’ on diffusion model

Operating in Parameter Space

Goal: Operate on the parameter space to maintain continuity

Operating in Parameter Space

Goal: Operate on the parameter space to maintain continuity

	Data Space	**Parameter Space**
Continuous	$\mathbf{x} \in \mathbb{R}^D$	$\mu \in \mathbb{R}^D$
Discrete	$\mathbf{x} \in \{0, 1\}^{KD}$	$\theta \in [0, 1]^{KD}$

Operating in Parameter Space

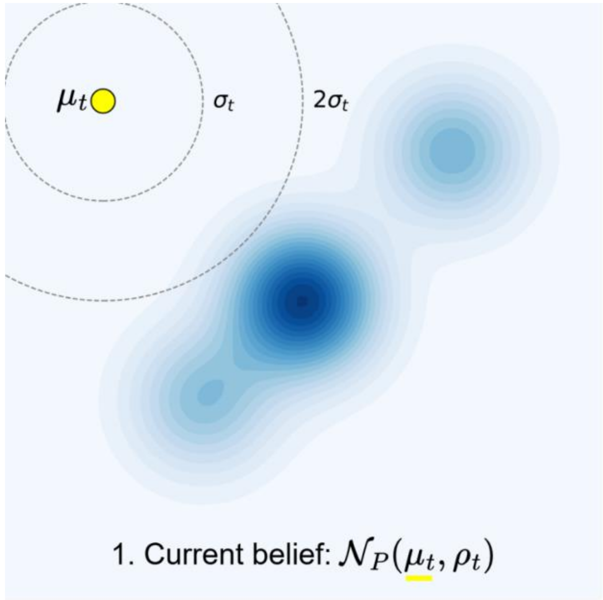
Goal: Operate on the parameter space to maintain continuity

	Data Space	**Parameter Space**
Continuous	$\mathbf{x} \in \mathbb{R}^D$	$\mu \in \mathbb{R}^D$
Discrete	$\mathbf{x} \in \{0,1\}^{KD}$	$\theta \in [0,1]^{KD}$

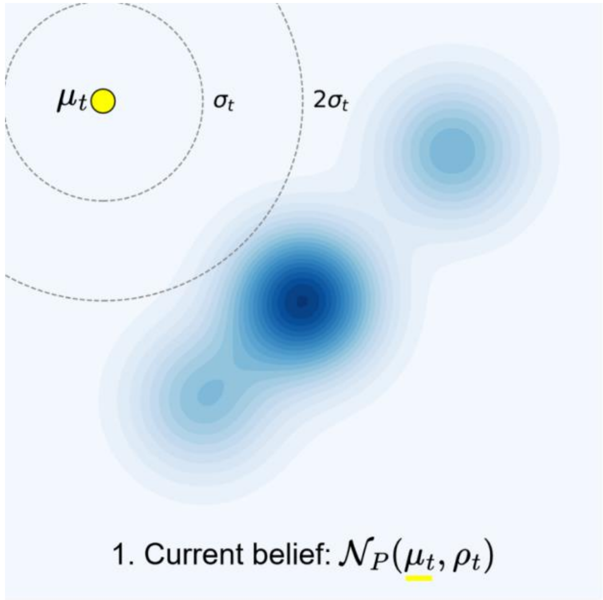
Discrete data: Relax each dimension's one-hot vector as a categorical distribution, which lives in a continuous simplex.

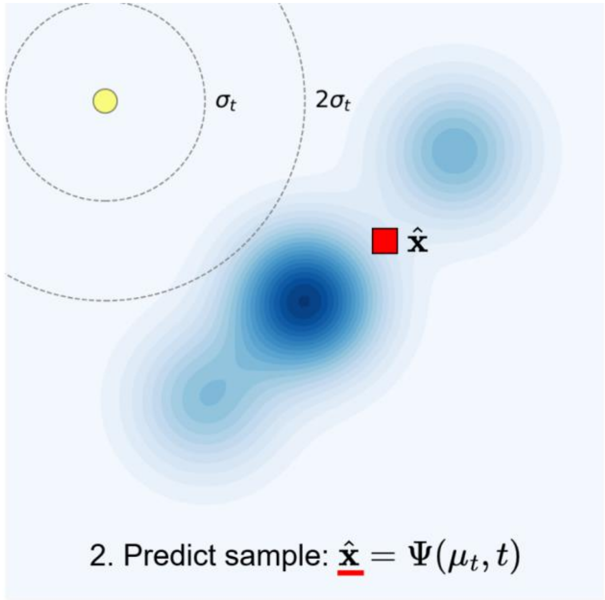
⇒ Maintain a **fully differentiable pipeline** even for discrete data

Example: Continuous Generation

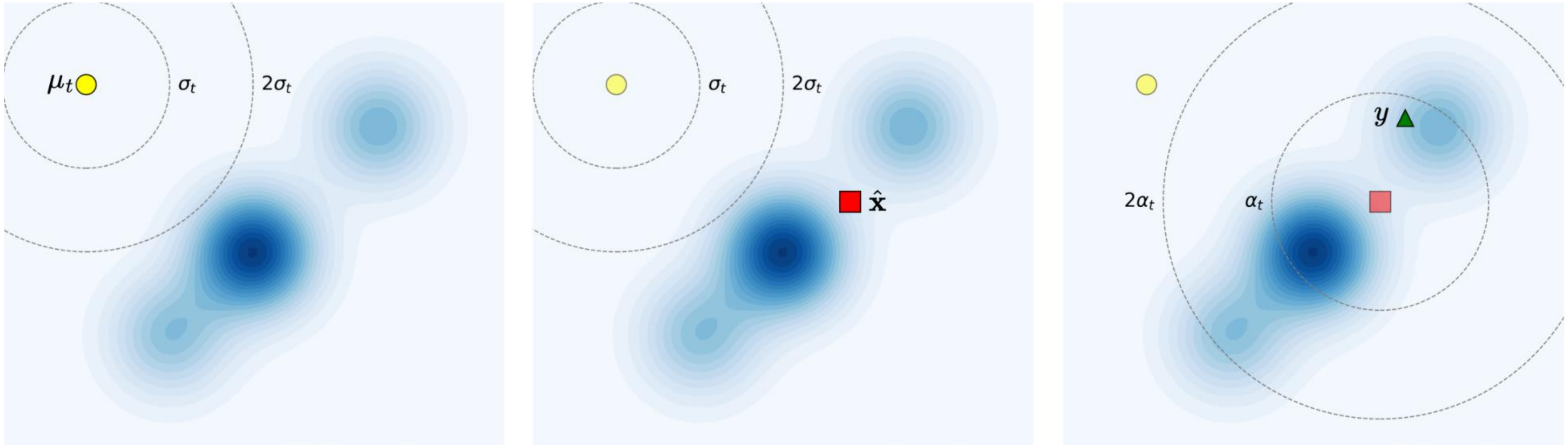


Example: Continuous Generation





Example: Continuous Generation

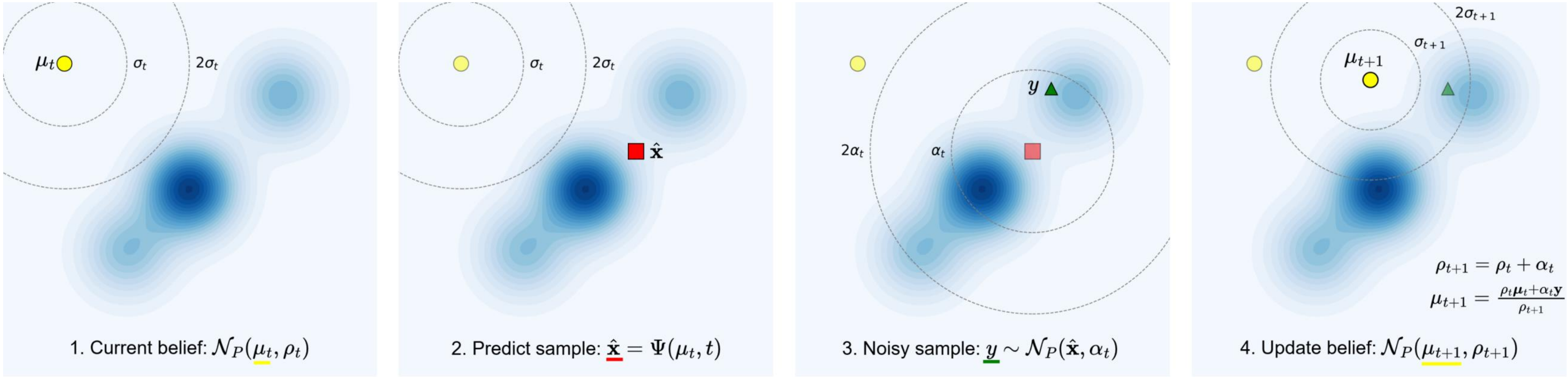


1. Current belief: $\mathcal{N}_P(\mu_t, \rho_t)$

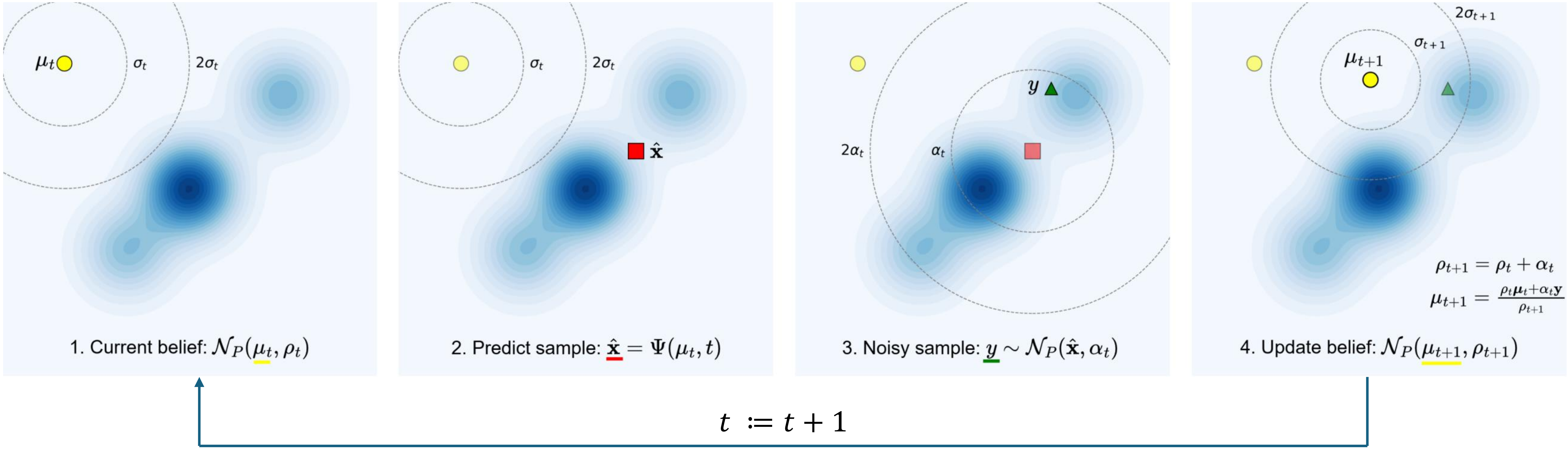
2. Predict sample: $\hat{\mathbf{x}} = \Psi(\mu_t, t)$

3. Noisy sample: $y \sim \mathcal{N}_P(\hat{\mathbf{x}}, \alpha_t)$

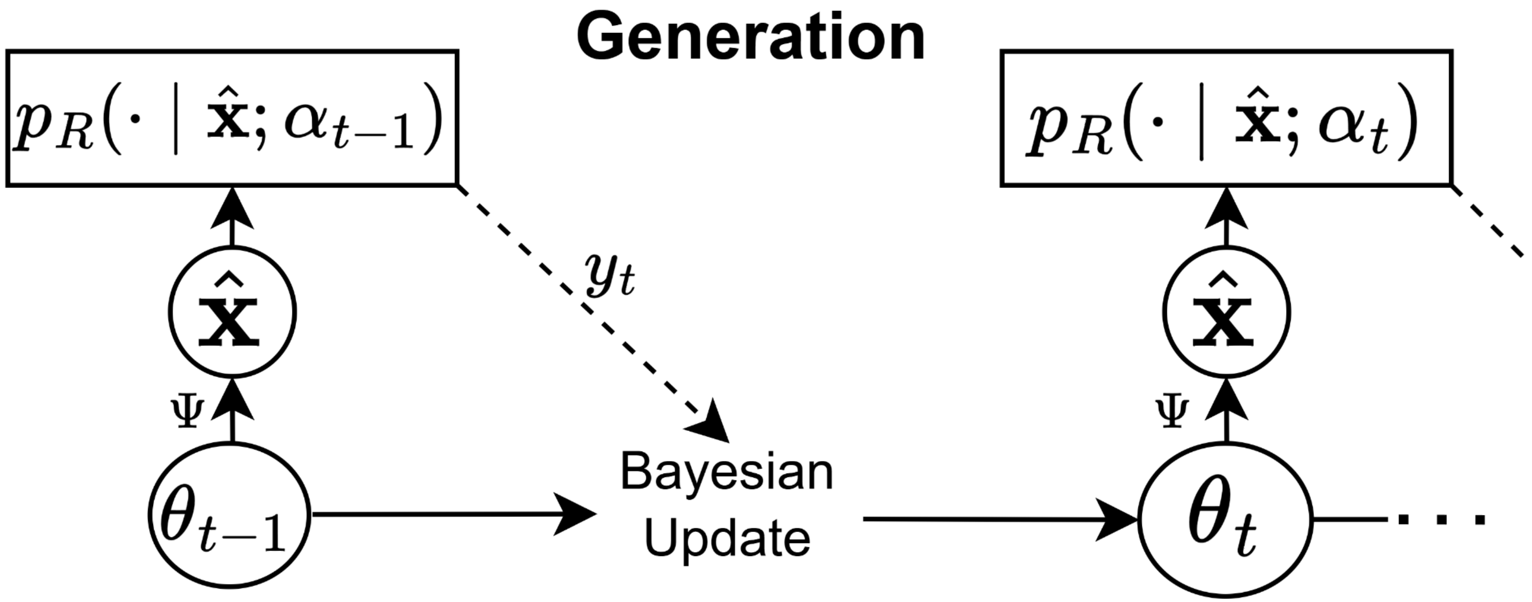
Example: Continuous Generation



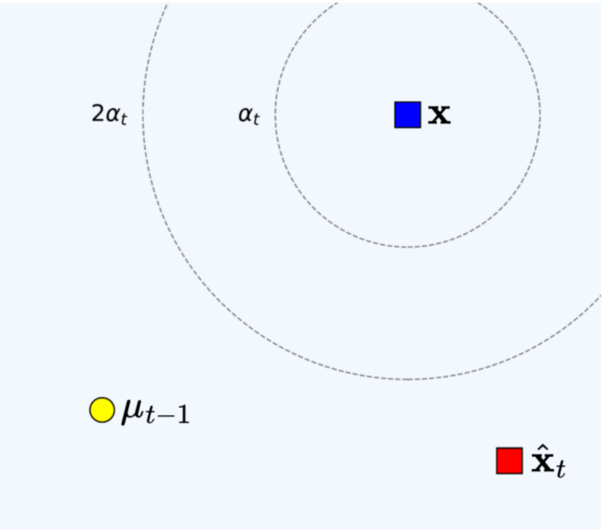
Example: Continuous Generation



Generative Process

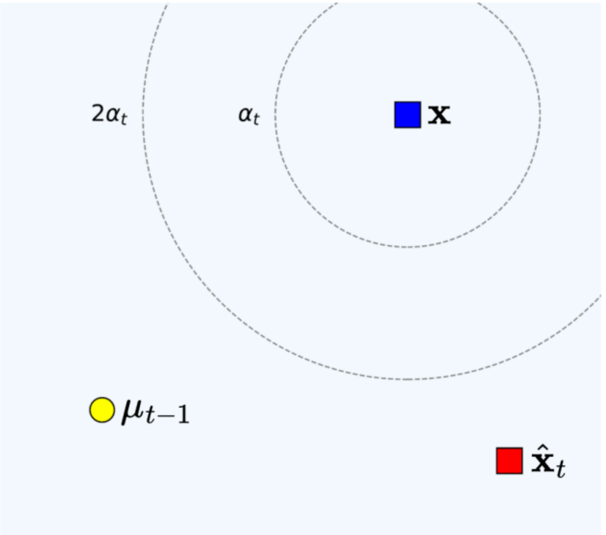


Example: Continuous Training

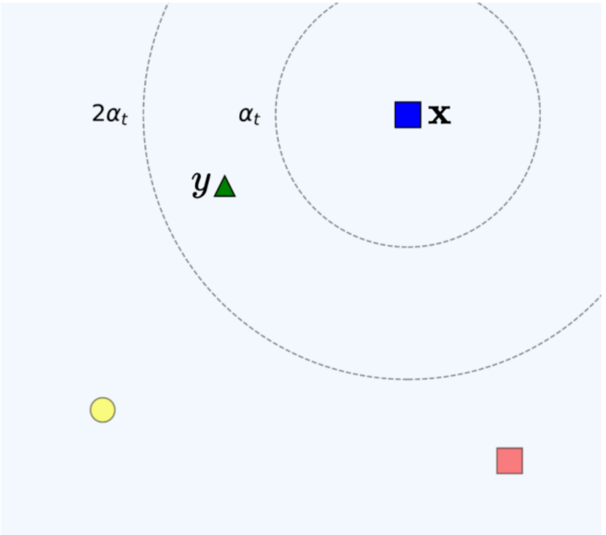


1. Prediction: $\hat{\mathbf{x}}_t = \Psi(\mu_{t-1}, t-1)$

Example: Continuous Training

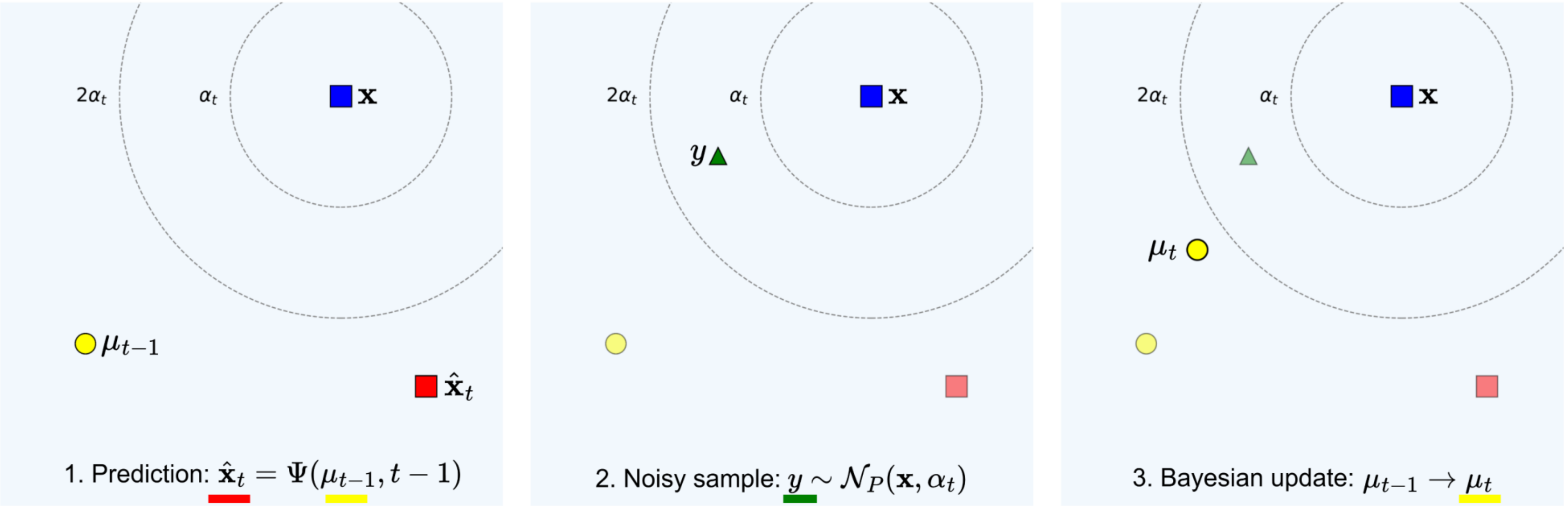


1. Prediction: $\hat{\mathbf{x}}_t = \Psi(\mu_{t-1}, t-1)$

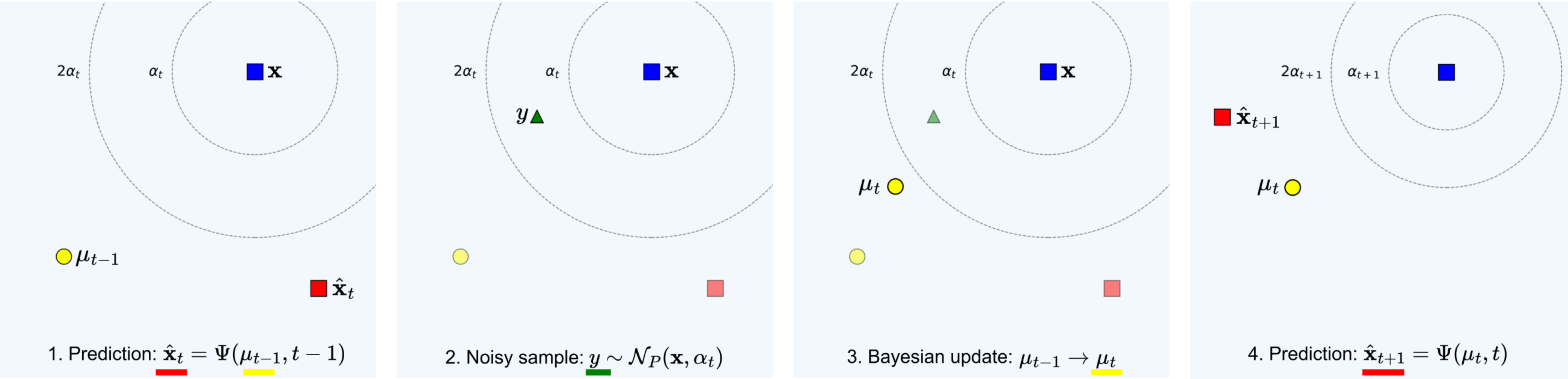


2. Noisy sample: $y \sim \mathcal{N}_P(\mathbf{x}, \alpha_t)$

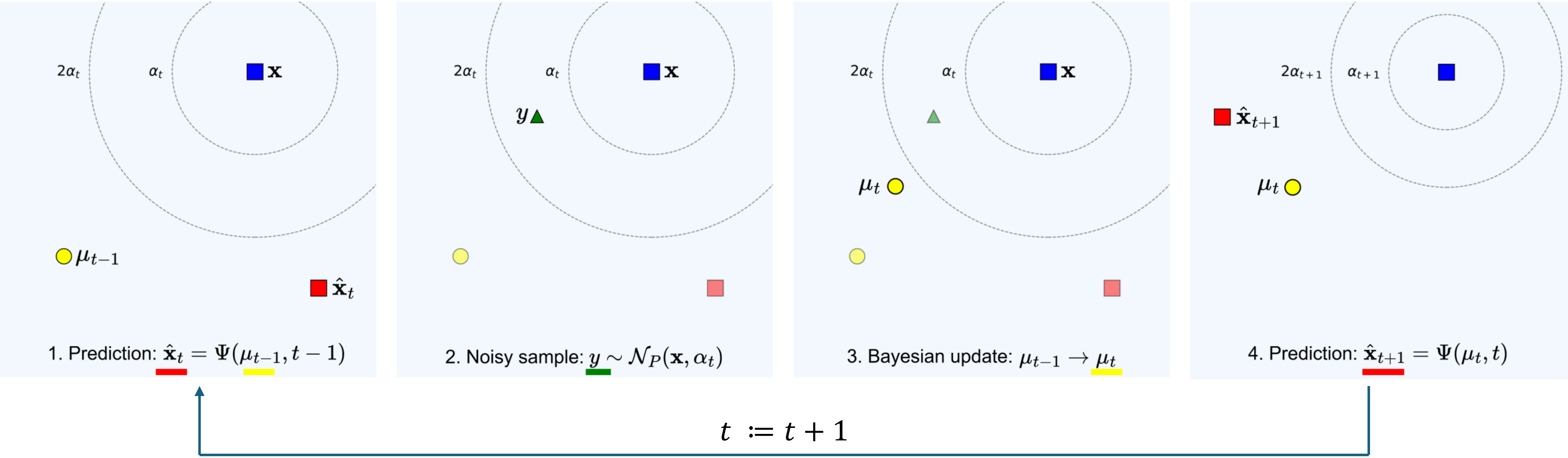
Example: Continuous Training



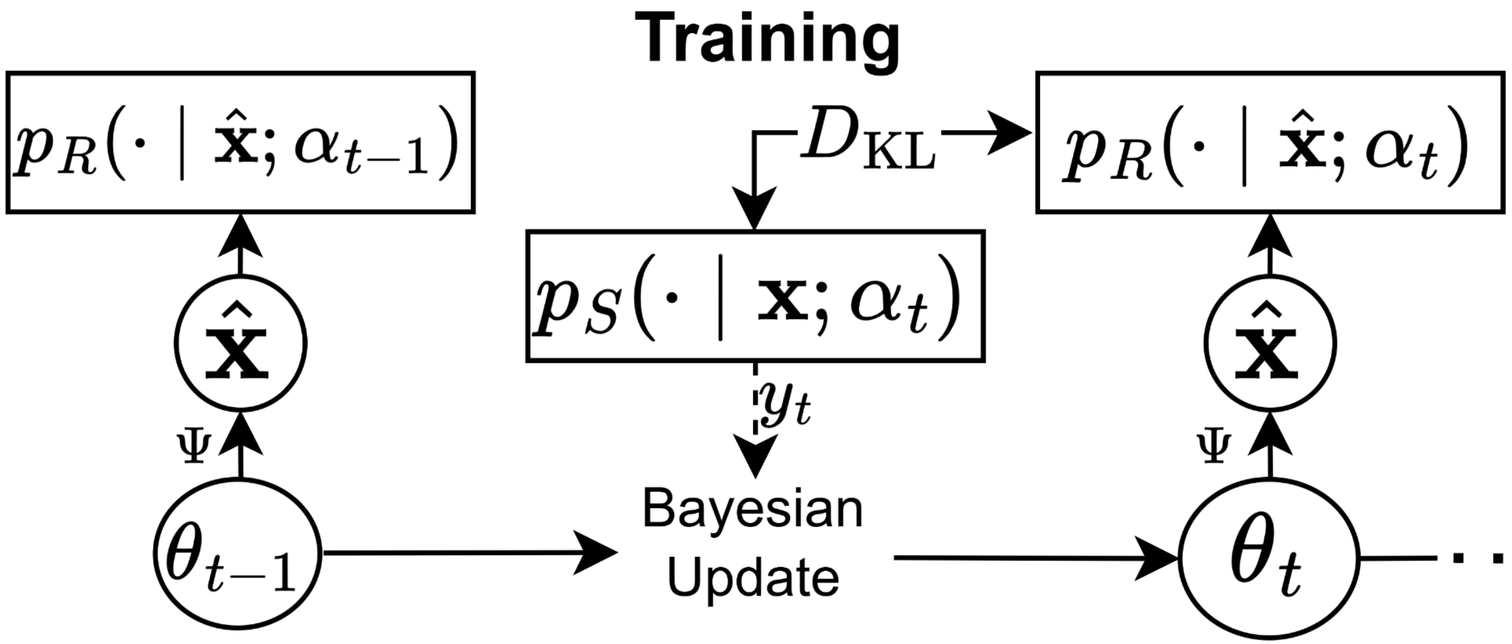
Example: Continuous Training



Example: Continuous Training



Training Process



Summary: Parameters θ

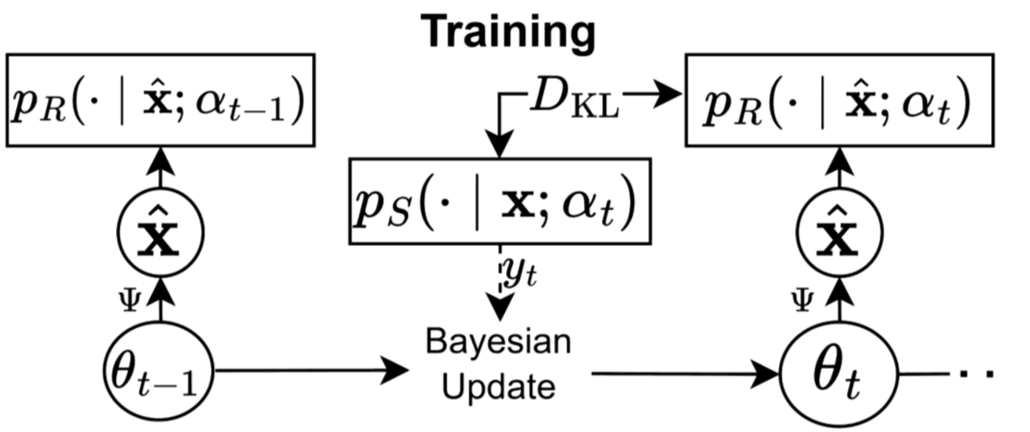
Purpose:

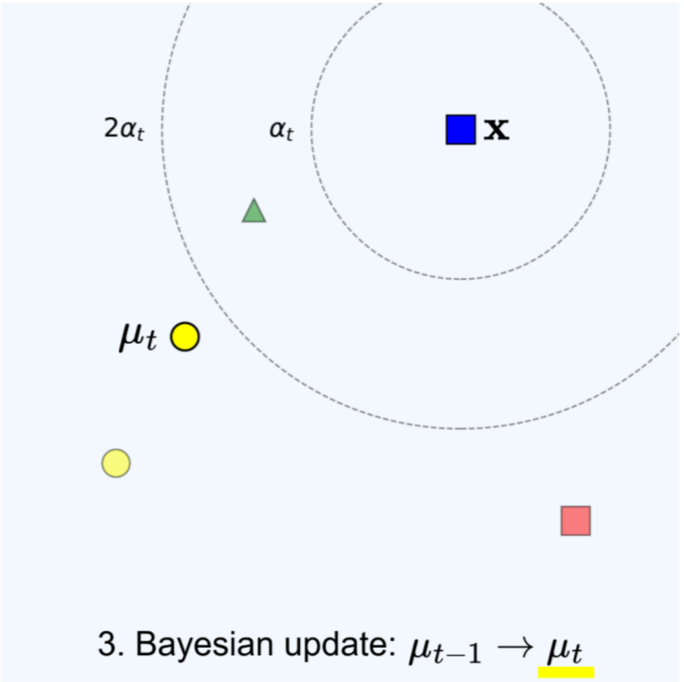
Parameters θ are inputs to the neural network Ψ

$$\hat{\mathbf{x}}_t \triangleq \Psi(\boldsymbol{\theta}_{t-1}, t-1)$$

Evolution:

During training, we perform Bayesian updates on θ_t, using samples from the sender distribution (around $\mathbf{x}$)





Sender and Receiver Distributions

Sender and Receiver Distributions

Fix n steps of the network, choose an accuracy schedule $\alpha_1 < \ldots < \alpha_n$

$$p_S(\mathbf{y} \mid \mathbf{x}; \alpha) = \mathcal{N}(\mathbf{y} \mid \mathbf{x}, \alpha^{-1}\mathbf{I})$$

Continuous case

Sender and Receiver Distributions

Fix n steps of the network, choose an accuracy schedule $\alpha_1 < \ldots < \alpha_n$

$$p_S(\mathbf{y} \mid \mathbf{x}; \alpha) = \mathcal{N}(\mathbf{y} \mid \mathbf{x}, \alpha^{-1}\mathbf{I})$$

Continuous case

- Depends on data type
- Parameterised by some point x and predetermined noise α
- Required to be factorizable (independent across dimensions

Sender and Receiver Distributions

Fix n steps of the network, choose an accuracy schedule $\alpha_1 < \ldots < \alpha_n$

$$p_S(\mathbf{y} \mid \mathbf{x}; \alpha) = \mathcal{N}(\mathbf{y} \mid \mathbf{x}, \alpha^{-1}\mathbf{I})$$

Continuous case

- Depends on data type
- Parameterised by some point x and predetermined noise α
- Required to be factorizable (independent across dimensions

$\sim \quad p_S(\mathbf{y} \mid$  $; \alpha_t)$

Sender and **Receiver** Distributions

Similarly, receiver distribution is parameterized by ‘current understanding’ $\hat{\mathbf{x}}$ and noise α

Sender and **Receiver** Distributions

Similarly, receiver distribution is parameterized by 'current understanding' $\hat{\mathbf{x}}$ and noise α

$$p_R(\mathbf{y} \mid \hat{\mathbf{x}}; \alpha) \triangleq \mathop{\mathbb{E}}_{p_O(\mathbf{x}' \mid \hat{\mathbf{x}})} p_S(\mathbf{y} \mid \mathbf{x}'; \alpha)$$

The receiver **is a mixture of sender distributions** weighted by its belief about $\mathbf{x}$

Sender and **Receiver** Distributions

Similarly, receiver distribution is parameterized by ‘current understanding’ $\hat{\mathbf{x}}$ and noise α

$$p_R(\mathbf{y} \mid \hat{\mathbf{x}}; \alpha) \triangleq \underset{p_O(\mathbf{x}' \mid \hat{\mathbf{x}})}{\mathbb{E}} p_S(\mathbf{y} \mid \mathbf{x}'; \alpha)$$

The receiver **is a mixture of sender distributions** weighted by its belief about $\mathbf{x}$

Now consider a loss function defined by the KL divergence

$$\mathrm{KL}\left[p_S(\mathbf{y} \mid \mathbf{x}, \alpha) \,\|\, p_R(\mathbf{y} \mid \hat{\mathbf{x}}, \alpha)\right]$$

Sender and **Receiver** Distributions

Similarly, receiver distribution is parameterized by 'current understanding' $\hat{\mathbf{x}}$ and noise α

$$p_R(\mathbf{y} \mid \hat{\mathbf{x}}; \alpha) \triangleq \mathop{\mathbb{E}}_{p_O(\mathbf{x}' \mid \hat{\mathbf{x}})} p_S(\mathbf{y} \mid \mathbf{x}'; \alpha)$$

The receiver **is a mixture of sender distributions** weighted by its belief about $\mathbf{x}$

Now consider a loss function defined by the KL divergence

$$\mathrm{KL}\left[p_S(\mathbf{y} \mid \mathbf{x}, \alpha) \,\|\, p_R(\mathbf{y} \mid \hat{\mathbf{x}}, \alpha)\right]$$

- ✓ Minimizing the loss pushes $\hat{\mathbf{x}}$ towards $\mathbf{x}$
- ✓ KL gives a 'smooth training signal' with respect to parameters of NN Ψ

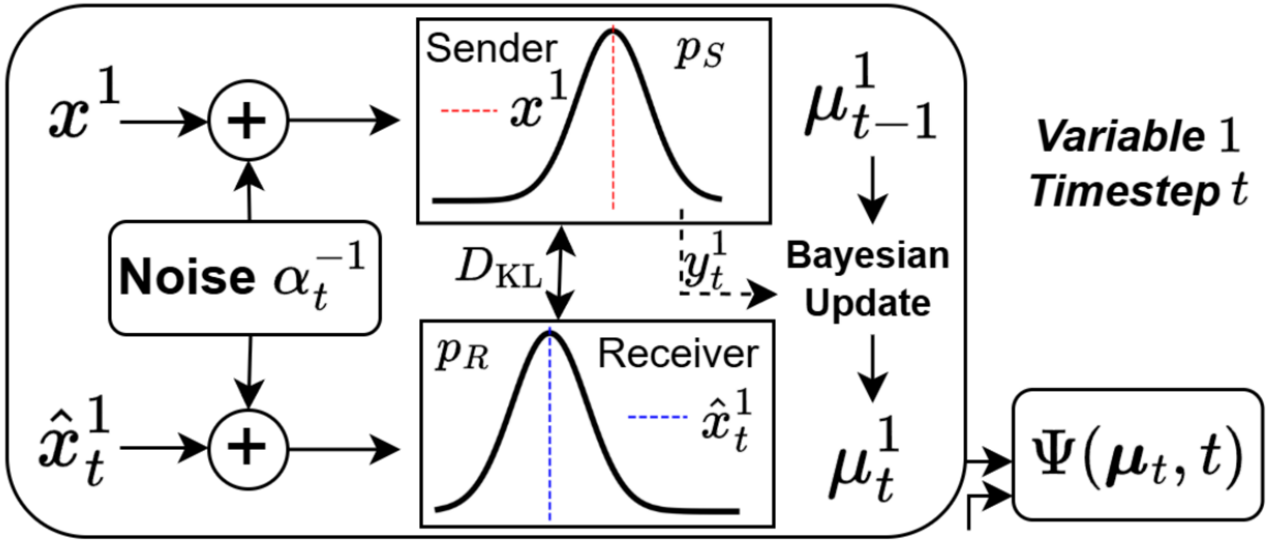
Internal Distributions: Continuous

Internal Distributions of BFN

$$p_S(\cdot \mid \mathbf{x}; \alpha_t) = \mathcal{N}(\mathbf{x}, \alpha_t^{-1}\mathbf{I}).$$

$$p_O(\mathbf{x}' \mid \hat{\mathbf{x}}_t) = \delta(\mathbf{x}' - \hat{\mathbf{x}}_t).$$

$$\begin{aligned} p_R(\cdot \mid \hat{\mathbf{x}}_t; \alpha_t) &= \mathop{\mathbb{E}}_{p_O(\mathbf{x}' \mid \hat{\mathbf{x}}_t)} p_S(\cdot \mid \mathbf{x}'; \alpha_t) \\ &= \mathcal{N}(\hat{\mathbf{x}}_t, \alpha_t^{-1}\mathbf{I}) \end{aligned}$$



Variable 1
Timestep t

Loss Function

Equivalent to a VAE loss where we have a **sequence** of latent samples $\mathbf{y}_1, \ldots, \mathbf{y}_n$

$$L^n(\mathbf{x}) = \mathbb{E}_{p(\boldsymbol{\theta}_1,\ldots,\boldsymbol{\theta}_{n-1})} \sum_{i=1}^{n} D_{\mathrm{KL}}\big(p_S(\cdot \mid \mathbf{x};\, \alpha_i) \,\|\, p_R(\cdot \mid \hat{\mathbf{x}}_i; \alpha_i)\big)$$

Loss Function

Equivalent to a VAE loss where we have a **sequence** of latent samples $\mathbf{y}_1, \ldots, \mathbf{y}_n$

$$L^n(\mathbf{x}) = \mathbb{E}_{p(\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_{n-1})} \sum_{i=1}^{n} D_{\mathrm{KL}}\big(p_S(\,\cdot \mid \mathbf{x};\, \alpha_i)\,\big\|\, p_R(\,\cdot \mid \hat{\mathbf{x}}_i; \alpha_i)\big)$$

Transmission interpretation:

What is the cost of transmitting sample $\mathbf{y}_i \sim p_S\big(\cdot \mid \mathbf{x}; \alpha_i\big)$ *from the sender to a receiver that believes* $\mathbf{y}_i \sim p_R\big(\cdot \mid \hat{\mathbf{x}}_i; \alpha_i\big)$

Loss Function

Equivalent to a VAE loss where we have a **sequence** of latent samples $\mathbf{y}_1, \ldots, \mathbf{y}_n$

$$L^n(\mathbf{x}) = \mathbb{E}_{p(\boldsymbol{\theta}_1,\ldots,\boldsymbol{\theta}_{n-1})} \sum_{i=1}^{n} D_{\mathrm{KL}}\big(p_S(\cdot \mid \mathbf{x};\, \alpha_i) \,\big\|\, p_R(\cdot \mid \hat{\mathbf{x}}_i; \alpha_i)\big)$$

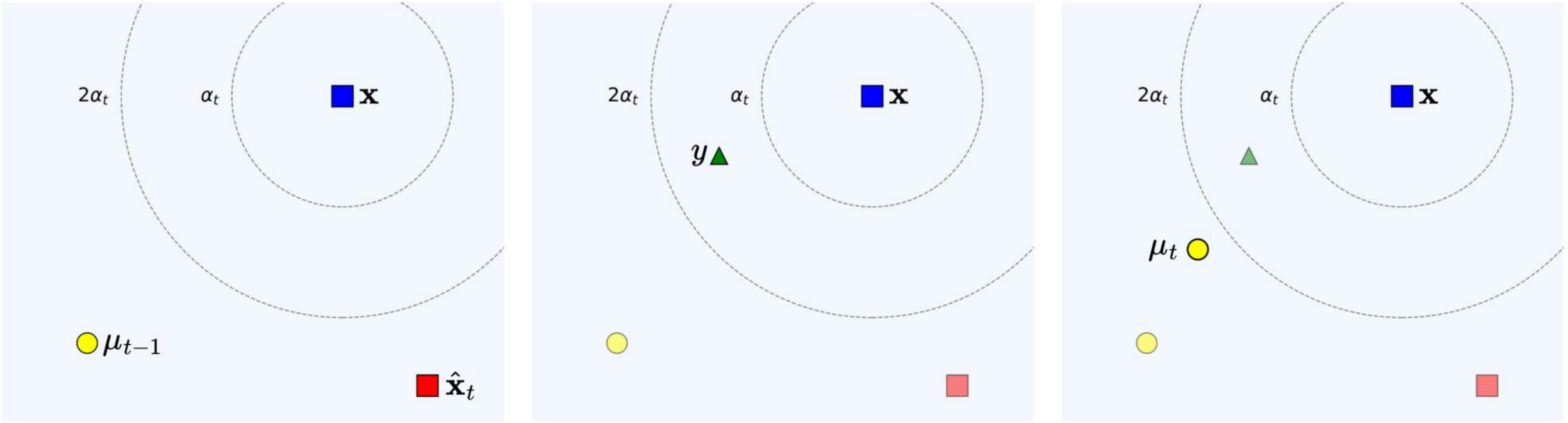
Transmission interpretation:

What is the cost of transmitting sample $\mathbf{y}_i \sim p_S\big(\cdot \mid \mathbf{x}; \alpha_i\big)$ *from the sender to a receiver that believes* $\mathbf{y}_i \sim p_R\big(\cdot \mid \hat{\mathbf{x}}_i; \alpha_i\big)$

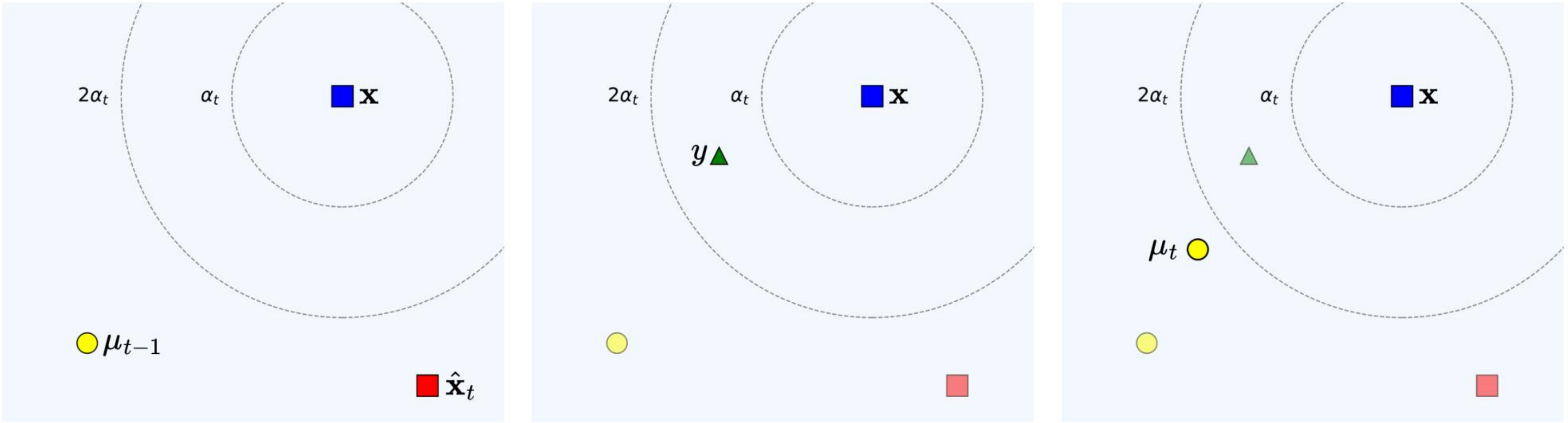
Sampling Issue: $\boldsymbol{\theta}_1 \rightarrow \boldsymbol{\theta}_2 \rightarrow \ldots \rightarrow \boldsymbol{\theta}_{n-1}$ from $p(\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_{n-1}) = \prod_{i=1}^{n-1} p_U(\boldsymbol{\theta}_i \mid \boldsymbol{\theta}_{i-1}, \mathbf{x}; \alpha_i)$

✗ Computationally expensive ✗ Not parallelizable

Update Distribution

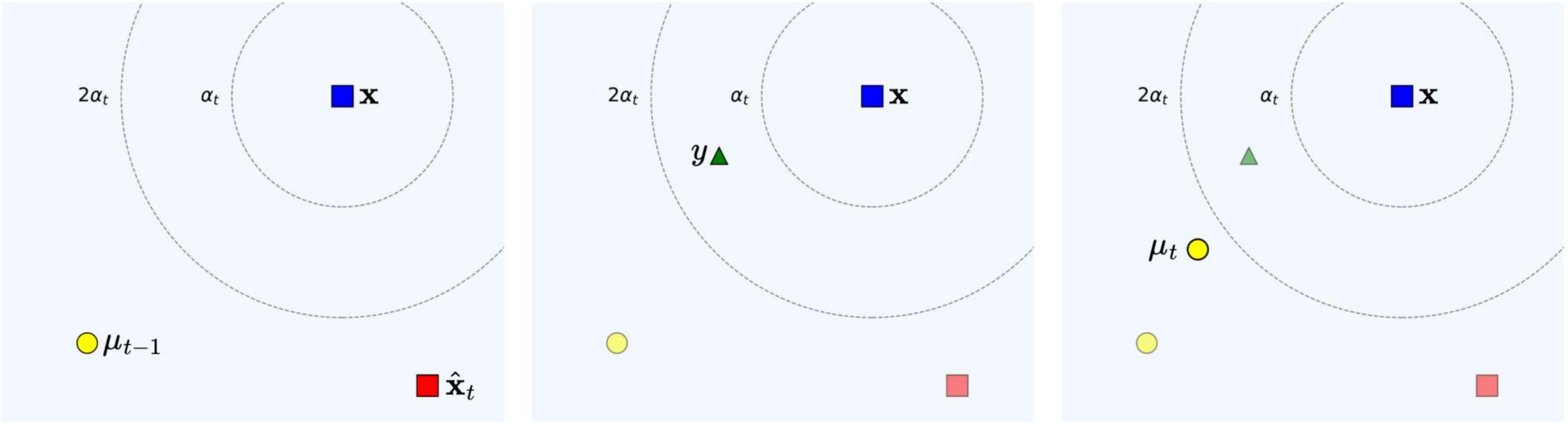


Update Distribution



$$\boldsymbol{\mu}_t = h(\boldsymbol{\mu}_{t-1}, \mathbf{y}_t, \alpha_t) = \frac{\alpha_t \mathbf{y}_t + \rho_{t-1} \boldsymbol{\mu}_{t-1}}{\rho_t}$$

Update Distribution

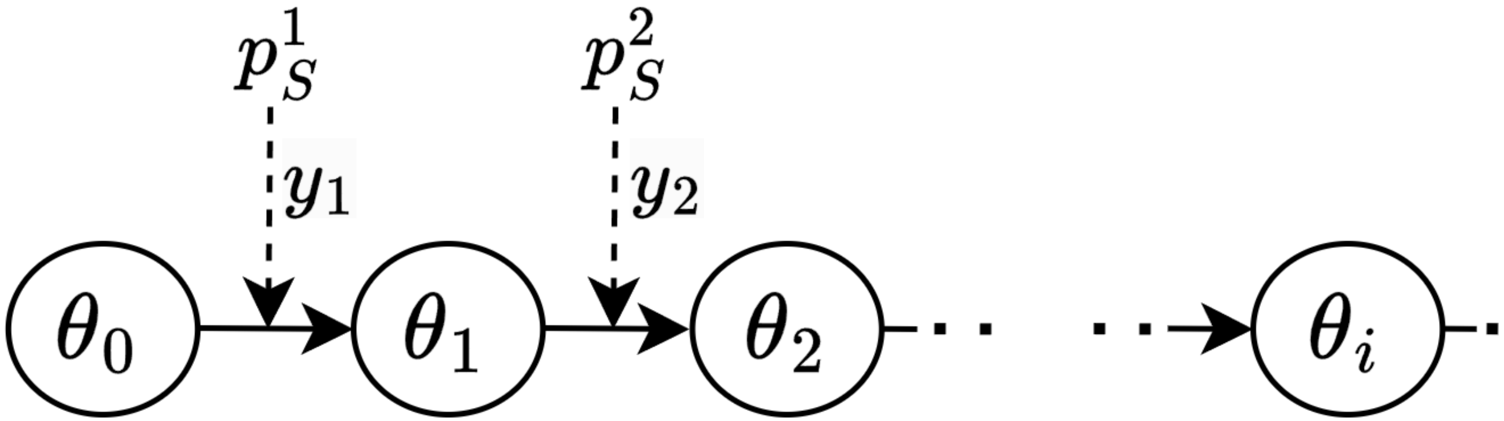


$$\boldsymbol{\mu}_t = h(\boldsymbol{\mu}_{t-1}, \mathbf{y}_t, \alpha_t) = \frac{\alpha_t \mathbf{y}_t + \rho_{t-1}\boldsymbol{\mu}_{t-1}}{\rho_t}$$

$$p_U(\boldsymbol{\mu}_t \mid \boldsymbol{\mu}_{t-1}, \mathbf{x}; \alpha_t) = \mathcal{N}\left(\boldsymbol{\mu}_t \mid \frac{\alpha_t \mathbf{x} + \boldsymbol{\mu}_{t-1}\rho_{t-1}}{\rho_t}, \frac{\alpha_t}{\rho_t^2}\mathbf{I}\right)$$

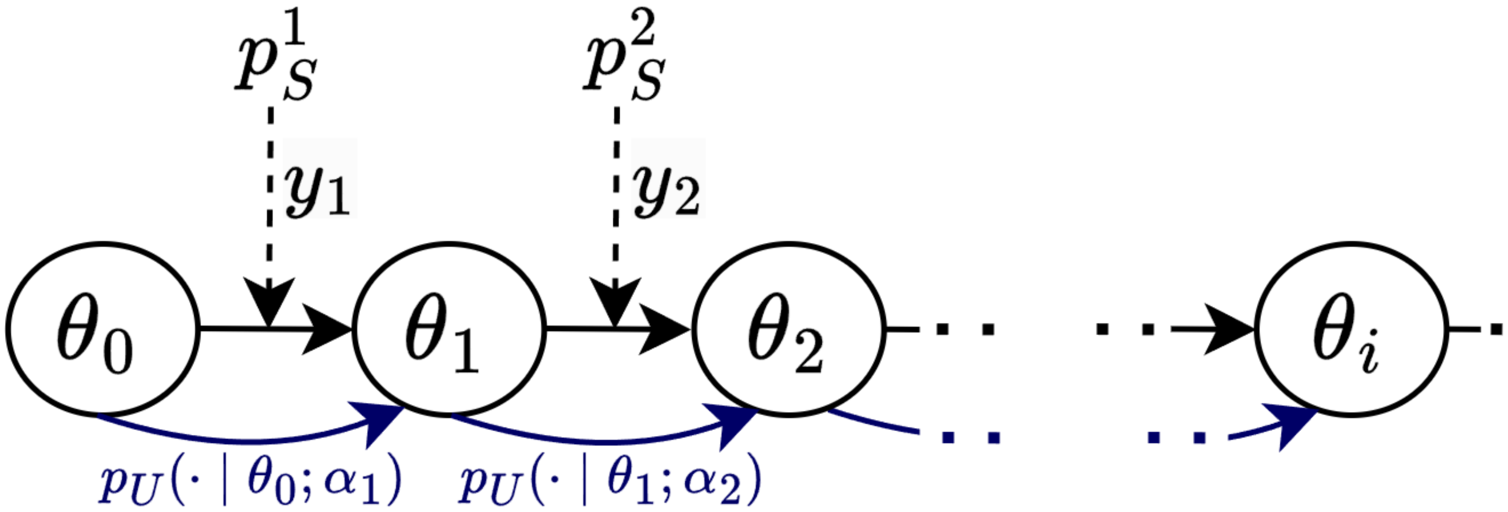
Update Distribution

Goal: Sample $\boldsymbol{\theta}_i$ directly (without sampling $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_{i-1}$) to simplify $p(\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_{n-1})$



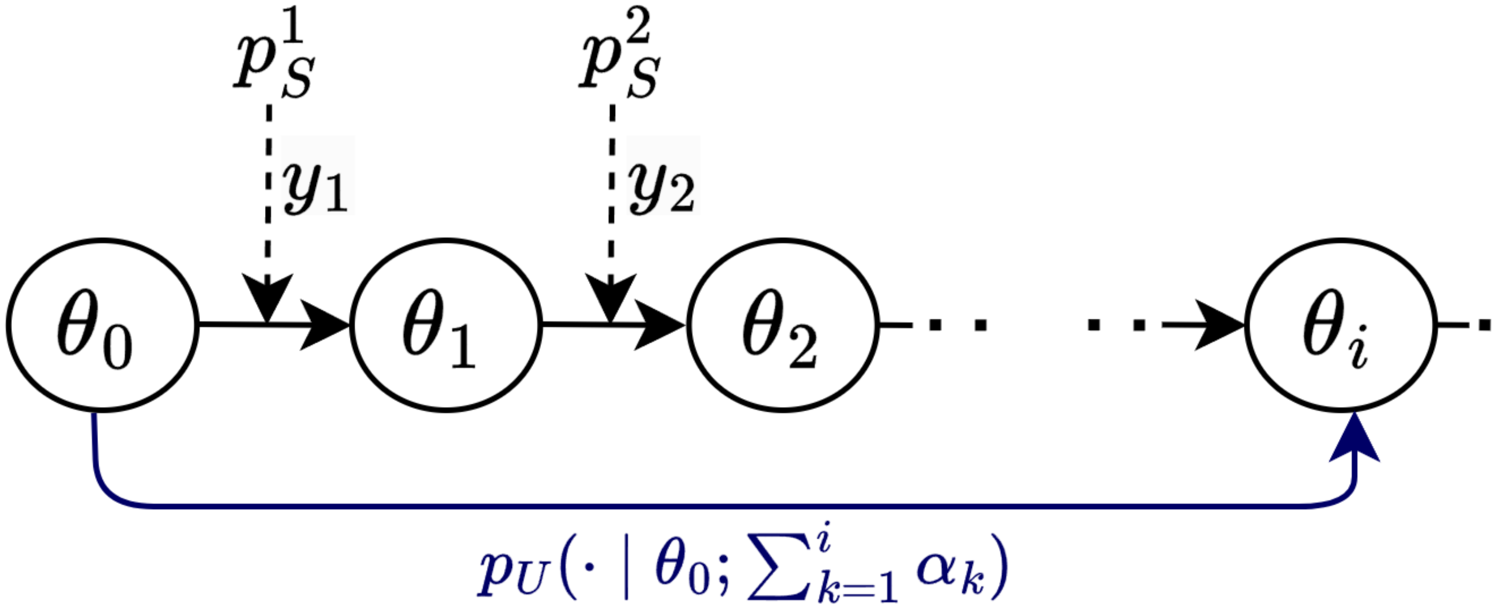
Update Distribution

Goal: Sample $\boldsymbol{\theta}_i$ directly (without sampling $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_{i-1}$) to simplify $p(\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_{n-1})$



Update Distribution

Goal: Sample $\boldsymbol{\theta}_i$ directly (without sampling $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_{i-1}$) to simplify $p(\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_{n-1})$



Additive Accuracies: $\underset{p_U(\boldsymbol{\theta}' \mid \boldsymbol{\theta}, \mathbf{x}; \alpha_a)}{\mathbb{E}} p_U(\boldsymbol{\theta}'' \mid \boldsymbol{\theta}', \mathbf{x}; \alpha_b) \overset{!}{=} p_U(\boldsymbol{\theta}'' \mid \boldsymbol{\theta}, \mathbf{x}; \alpha_a + \alpha_b)$

Loss Function

$$L^n(\mathbf{x}) = \mathbb{E}_{p(\boldsymbol{\theta}_1,\ldots,\boldsymbol{\theta}_{n-1})} \sum_{i=1}^{n} D_{\mathrm{KL}}\Big(p_S(\cdot \mid \mathbf{x};\alpha_i) \,\|\, p_R(\cdot \mid \hat{\mathbf{x}}_i;\alpha_i)\Big)$$

Loss Function

$$L^n(\mathbf{x}) = \mathbb{E}_{p(\boldsymbol{\theta}_1,\ldots,\boldsymbol{\theta}_{n-1})} \sum_{i=1}^{n} D_{\mathrm{KL}}\Big(p_S(\cdot \mid \mathbf{x}; \alpha_i) \,\|\, p_R(\cdot \mid \hat{\mathbf{x}}_i; \alpha_i)\Big)$$

$$= n\,\mathbb{E}_{i \sim U\{1,n\}}\, \mathbb{E}_{p(\boldsymbol{\theta}_1,\ldots,\boldsymbol{\theta}_{n-1})} \underbrace{D_{\mathrm{KL}}\Big(p_S(\cdot \mid \mathbf{x}; \alpha_i) \,\|\, p_R(\cdot \mid \hat{\mathbf{x}}_i; \alpha_i)\Big)}_{\boxed{\text{Depends on } \boldsymbol{\theta}_{i-1} \text{ only}}}$$

Loss Function

$$
\begin{aligned}
L^n(\mathbf{x}) &= \mathbb{E}_{p(\boldsymbol{\theta}_1,\dots,\boldsymbol{\theta}_{n-1})} \sum_{i=1}^{n} D_{\mathrm{KL}}\Big(p_S(\cdot \mid \mathbf{x}; \alpha_i) \,\|\, p_R(\cdot \mid \hat{\mathbf{x}}_i; \alpha_i)\Big) \\
&= n\,\mathbb{E}_{i\sim U\{1,n\}}\,\mathbb{E}_{p(\boldsymbol{\theta}_1,\dots,\boldsymbol{\theta}_{n-1})} \underbrace{D_{\mathrm{KL}}\Big(p_S(\cdot \mid \mathbf{x}; \alpha_i) \,\|\, p_R(\cdot \mid \hat{\mathbf{x}}_i; \alpha_i)\Big)}_{\boxed{\text{Depends on } \boldsymbol{\theta}_{i-1} \text{ only}}} \\
&= n\,\mathbb{E}_{i\sim U\{1,\dots,n\}}\,\mathbb{E}_{p_U(\boldsymbol{\theta}_{i-1}\mid\boldsymbol{\theta}_0,\mathbf{x};\sum_{j=1}^{i-1}\alpha_j)} D_{\mathrm{KL}}\Big(p_S(\cdot \mid \mathbf{x}; \alpha_i) \,\|\, p_R(\cdot \mid \hat{\mathbf{x}}_i; \alpha_i)\Big)
\end{aligned}
$$

Loss Function

$$L^n(\mathbf{x}) = \mathbb{E}_{p(\boldsymbol{\theta}_1,\ldots,\boldsymbol{\theta}_{n-1})} \sum_{i=1}^{n} D_{\mathrm{KL}}\Big(p_S(\cdot \mid \mathbf{x}; \alpha_i) \,\|\, p_R(\cdot \mid \hat{\mathbf{x}}_i; \alpha_i)\Big)$$

$$= n\,\mathbb{E}_{i\sim U\{1,n\}}\,\mathbb{E}_{p(\boldsymbol{\theta}_1,\ldots,\boldsymbol{\theta}_{n-1})} \underbrace{D_{\mathrm{KL}}\Big(p_S(\cdot \mid \mathbf{x}; \alpha_i) \,\|\, p_R(\cdot \mid \hat{\mathbf{x}}_i; \alpha_i)\Big)}_{\boxed{\text{Depends on } \boldsymbol{\theta}_{i-1} \text{ only}}}$$

$$= n\,\mathbb{E}_{i\sim U\{1,\ldots,n\}}\,\mathbb{E}_{p_U(\boldsymbol{\theta}_{i-1} \mid \boldsymbol{\theta}_0, \mathbf{x}; \sum_{j=1}^{i-1} \alpha_j)} D_{\mathrm{KL}}\Big(p_S(\cdot \mid \mathbf{x}; \alpha_i) \,\|\, p_R(\cdot \mid \hat{\mathbf{x}}_i; \alpha_i)\Big)$$

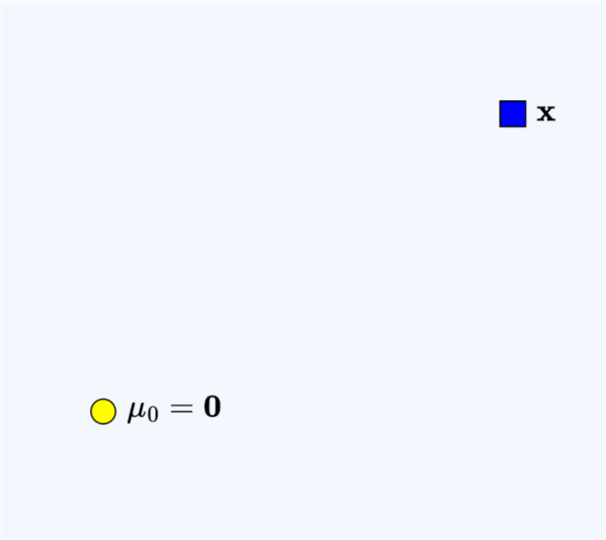
- ✓ Compute Monte Carlo without the n-step evolution of parameters
- ✓ Natural extension to continuous time loss

Loss Function

$$L^n(\mathbf{x}) = n\,\mathbb{E}_{i\sim U\{1,\ldots,n\}}\,\mathbb{E}_{p_U(\boldsymbol{\theta}_{i-1}|\boldsymbol{\theta}_0,\mathbf{x};\sum_{j=1}^{i-1}\alpha_j)}\,D_{\mathrm{KL}}\Big(p_S(\cdot \mid \mathbf{x};\alpha_i)\,\|\,p_R(\cdot \mid \hat{\mathbf{x}}_i;\alpha_i)\Big)$$

Loss Function

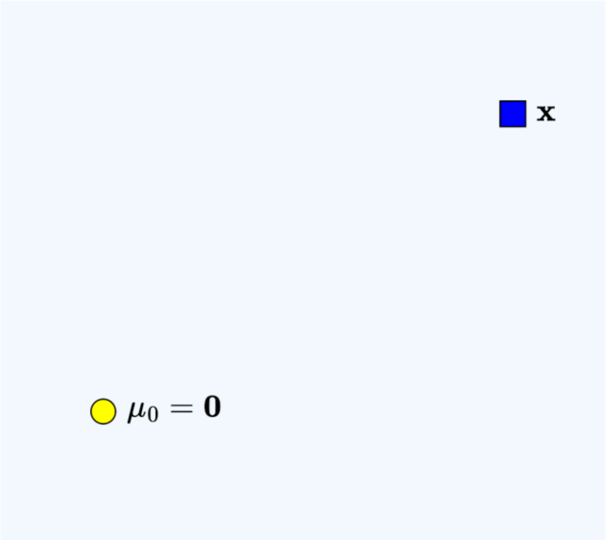
$$L^n(\mathbf{x}) = n\,\mathbb{E}_{i\sim U\{1,\ldots,n\}}\,\mathbb{E}_{p_U(\boldsymbol{\theta}_{i-1}|\boldsymbol{\theta}_0,\mathbf{x};\sum_{j=1}^{i-1}\alpha_j)}\,D_{\mathrm{KL}}\Big(p_S(\cdot\mid\mathbf{x};\alpha_i)\,\|\,p_R(\cdot\mid\hat{\mathbf{x}}_i;\alpha_i)\Big)$$



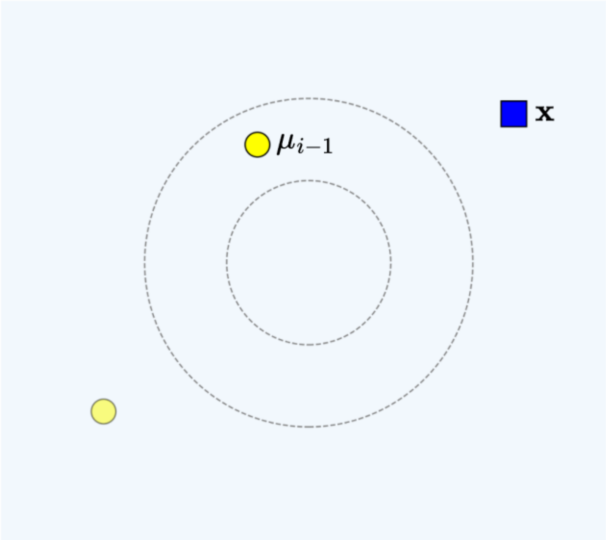
1. Sample: $i \sim U\{1,..,n\}$

Loss Function

$$L^n(\mathbf{x}) = n\,\mathbb{E}_{i\sim U\{1,\ldots,n\}}\,\mathbb{E}_{p_U(\boldsymbol{\theta}_{i-1}|\boldsymbol{\theta}_0,\mathbf{x};\sum_{j=1}^{i-1}\alpha_j)}\,D_{\mathrm{KL}}\Big(p_S(\cdot \mid \mathbf{x};\alpha_i)\,\Big\|\,p_R(\cdot \mid \hat{\mathbf{x}}_i;\alpha_i)\Big)$$



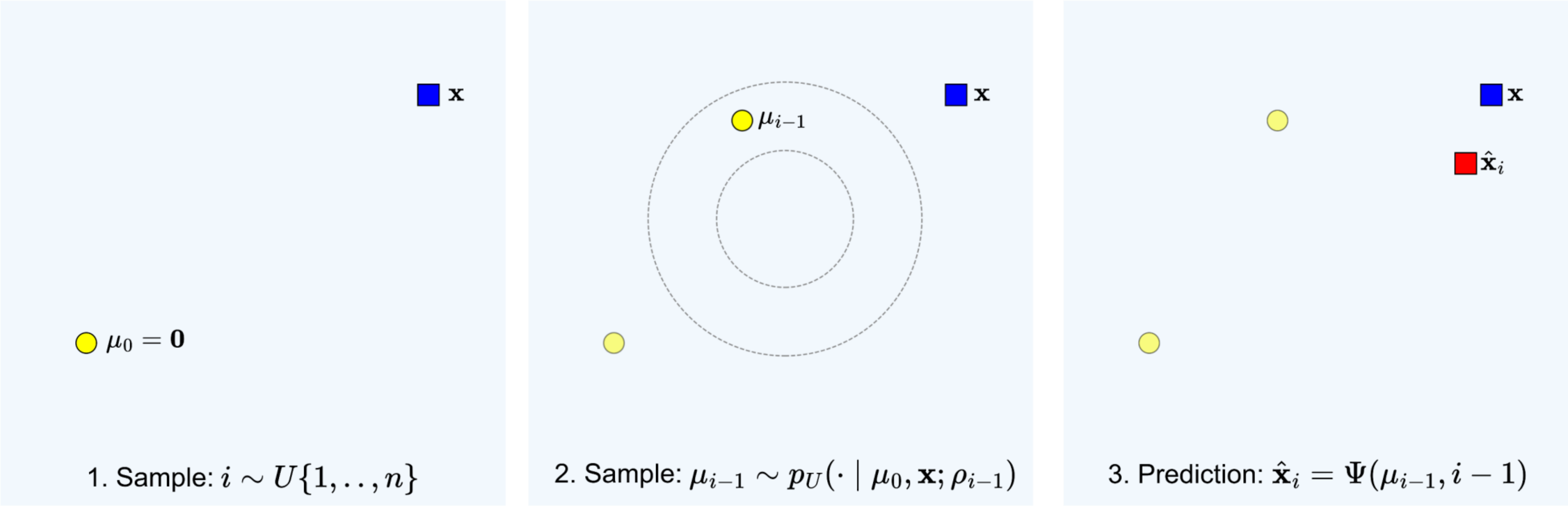
1. Sample: $i \sim U\{1,..,n\}$



2. Sample: $\mu_{i-1} \sim p_U(\cdot \mid \mu_0, \mathbf{x}; \rho_{i-1})$

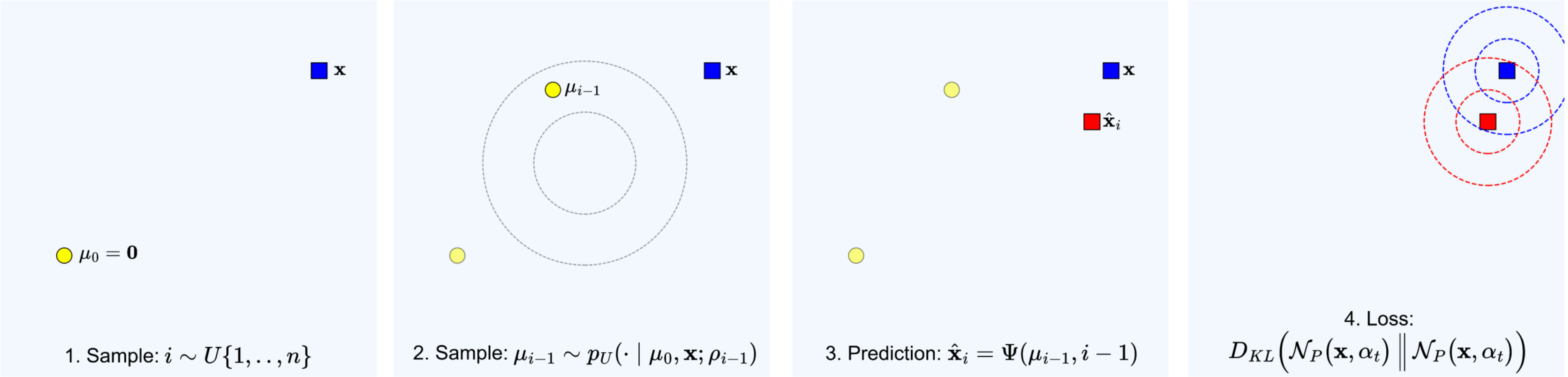
Loss Function

$$L^n(\mathbf{x}) = n\,\mathbb{E}_{i\sim U\{1,\ldots,n\}}\,\mathbb{E}_{p_U(\boldsymbol{\theta}_{i-1}|\boldsymbol{\theta}_0,\mathbf{x};\,\sum_{j=1}^{i-1}\alpha_j)}\,D_{\mathrm{KL}}\Big(p_S(\cdot \mid \mathbf{x};\alpha_i)\,\big\|\,p_R(\cdot \mid \hat{\mathbf{x}}_i;\alpha_i)\Big)$$



Loss Function

$$L^n(\mathbf{x}) = n\,\mathbb{E}_{i\sim U\{1,\ldots,n\}}\,\mathbb{E}_{p_U(\boldsymbol{\theta}_{i-1}\mid\boldsymbol{\theta}_0,\mathbf{x};\sum_{j=1}^{i-1}\alpha_j)}\,D_{\mathrm{KL}}\Big(p_S(\cdot\mid\mathbf{x};\alpha_i)\,\Big\|\,p_R(\cdot\mid\hat{\mathbf{x}}_i;\alpha_i)\Big)$$



Flow Distribution

Goal: Extend to continuous time. First define the accuracy schedule $\beta(t) = \int_0^t \alpha(t')\,dt'$

$$p_F(\boldsymbol{\theta} \mid \mathbf{x}; t) \triangleq p_U(\boldsymbol{\theta} \mid \boldsymbol{\theta}_0, \mathbf{x}; \beta(t))$$

Flow Distribution

Goal: Extend to continuous time. First define the accuracy schedule $\beta(t) = \int_0^t \alpha(t')\,dt'$

$$
\begin{aligned}
p_F(\boldsymbol{\theta} \mid \mathbf{x}; t) &\triangleq p_U(\boldsymbol{\theta} \mid \boldsymbol{\theta}_0, \mathbf{x}; \beta(t)) \\
&= \mathcal{N}\left(\boldsymbol{\mu} \mid \frac{\beta(t)}{1+\beta(t)}\mathbf{x}, \frac{\beta(t)}{(1+\beta(t))^2} I\right)
\end{aligned}
$$

Continuous case

Flow Distribution

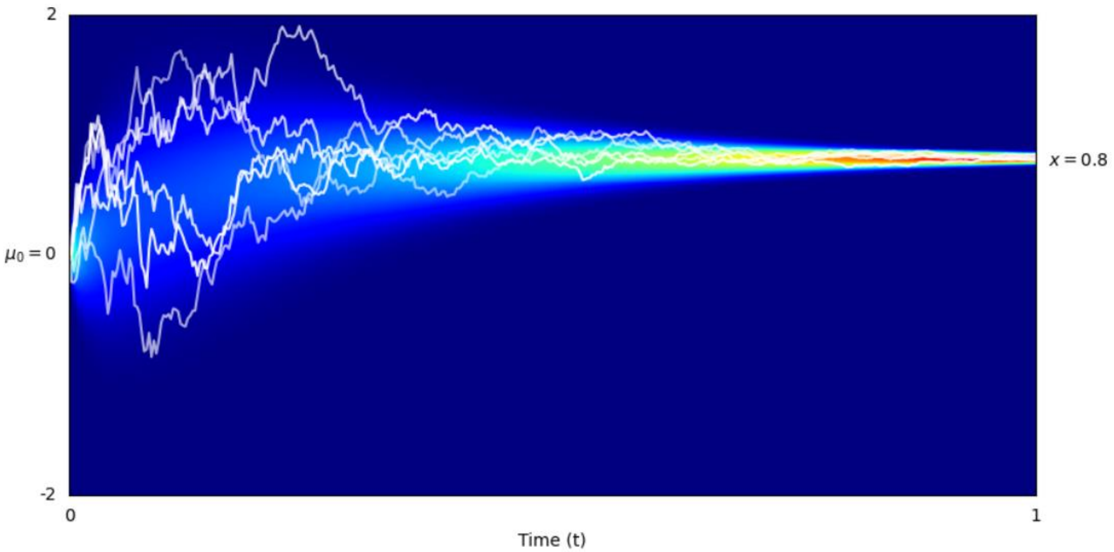
Goal: Extend to continuous time. First define the accuracy schedule $\beta(t) = \int_0^t \alpha(t')\,dt'$

$$p_F(\boldsymbol{\theta} \mid \mathbf{x}; t) \triangleq p_U(\boldsymbol{\theta} \mid \boldsymbol{\theta}_0, \mathbf{x}; \beta(t))$$
$$= \mathcal{N}\left(\boldsymbol{\mu} \mid \frac{\beta(t)}{1+\beta(t)}\mathbf{x}, \frac{\beta(t)}{(1+\beta(t))^2} I\right)$$ *Continuous case*

Benefits:

✓ Permits continuous time loss L^∞

✓ Decoupled number of samples from t



Differences to Diffusion:

Occurring in parameter space, not sample space

Start with an initialisation instead of noisy sample

Continuous Time Loss

Similarly to Variational Diffusion Models, we derive a continuous time loss

$$L^{\infty}(\mathbf{x}) \stackrel{\text{def}}{=} \lim_{n \to \infty} L^{n}(\mathbf{x})$$

Continuous Time Loss

Similarly to Variational Diffusion Models, we derive a continuous time loss

$$L^{\infty}(\mathbf{x}) \stackrel{\text{def}}{=} \lim_{n\to\infty} L^{n}(\mathbf{x})$$

$$= \lim_{\epsilon\to 0} \frac{1}{\epsilon} \mathop{\mathbb{E}}_{\substack{t\sim U(\epsilon,1)\\ p_F(\boldsymbol{\theta}|\mathbf{x},t-\epsilon)}} D_{KL}\Big(p_S(\cdot \mid \mathbf{x}; \alpha(t,\epsilon)) \,\|\, p_R(\cdot \mid \hat{\mathbf{x}}; \alpha(t,\epsilon))\Big)$$

Continuous Time Loss

Similarly to Variational Diffusion Models, we derive a continuous time loss

$$L^{\infty}(\mathbf{x}) \stackrel{\text{def}}{=} \lim_{n\to\infty} L^{n}(\mathbf{x})$$

$$= \lim_{\epsilon\to 0} \frac{1}{\epsilon} \mathop{\mathbb{E}}_{\substack{t\sim U(\epsilon,1)\\ p_F(\boldsymbol{\theta}|\mathbf{x},t-\epsilon)}} D_{KL}\Big(p_S(\cdot \mid \mathbf{x}; \alpha(t,\epsilon)) \,\|\, p_R(\cdot \mid \hat{\mathbf{x}}; \alpha(t,\epsilon))\Big)$$

$$= -\ln \sigma_1 \mathop{\mathbb{E}}_{t\sim U(0,1), p_F(\boldsymbol{\theta}|\mathbf{x};t)} \frac{\left\|\mathbf{x} - \hat{\mathbf{x}}(\boldsymbol{\theta},t)\right\|^2}{\sigma_1^{2t}}$$

Continuous case

Continuous Time Loss

Similarly to Variational Diffusion Models, we derive a continuous time loss

$$L^{\infty}(\mathbf{x}) \stackrel{\text{def}}{=} \lim_{n\to\infty} L^n(\mathbf{x})$$

$$= \lim_{\epsilon\to 0} \frac{1}{\epsilon} \mathbb{E}_{\substack{t\sim U(\epsilon,1) \\ p_F(\boldsymbol{\theta}|\mathbf{x},t-\epsilon)}} D_{KL}\Big(p_S(\cdot \mid \mathbf{x}; \alpha(t,\epsilon)) \,\|\, p_R(\cdot \mid \hat{\mathbf{x}}; \alpha(t,\epsilon))\Big)$$

$$= -\ln \sigma_1 \mathbb{E}_{t\sim U(0,1), p_F(\boldsymbol{\theta}|\mathbf{x};t)} \frac{\|\mathbf{x} - \hat{\mathbf{x}}(\boldsymbol{\theta}, t)\|^2}{\sigma_1^{2t}}$$

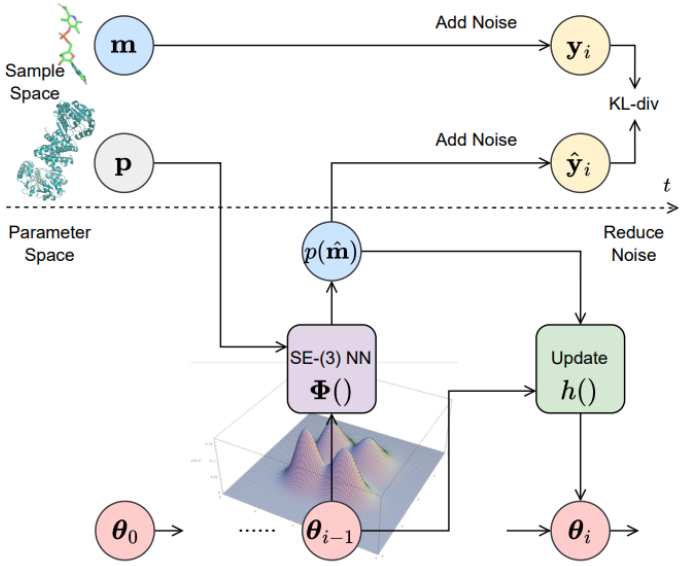
Continuous case

- ✓ No need to choose discrete number of steps
- ✓ Simplifies to a cleaner integral
- ✓ Flexible steps at inference time when trained on continuous loss

Application: Hybrid Data

BFNs excel at **hybrid data tasks** due to their **unified framework**. For example, with structure-based drug design (SBDD) where we model (continuous) atom coordinates and (discrete) atom types

- ✓ Robust against mode collapse and noise
- ✓ Low variance updates and faster convergence
- ✓ Fewer sampling errors (incomplete or distorted molecules)



MolCRAFT: Structure-Based Drug Design in Continuous Parameter Space (Qu et al., 2024, arXiv:2404.12141)

Thank you for listening

Supplementary Material

‘Full’ Loss

Define the n-step *discrete-time loss* $L^n(\mathbf{x})$ as the expected number of nats required to first transmit $\mathbf{y}_1, \ldots, \mathbf{y}_n$, and the *reconstruction loss* $L^r(\mathbf{x})$ as the expected number of nats required to then transmit $\mathbf{x}$. Since — using a bits-back coding scheme [7, 11] — it requires $D_{KL}(p_S \parallel p_R)$ nats to transmit a sample from p_S to a receiver with p_R,

$$L^n(\mathbf{x}) = \mathop{\mathbb{E}}_{p(\boldsymbol{\theta}_1,\ldots,\boldsymbol{\theta}_{n-1})} \sum_{i=1}^{n} D_{KL}\left(p_S(\cdot \mid \mathbf{x}; \alpha_i) \parallel p_R(\cdot \mid \boldsymbol{\theta}_{i-1}; t_{i-1}, \alpha_i)\right), \tag{13}$$

$$L^r(\mathbf{x}) = -\mathop{\mathbb{E}}_{p_F(\boldsymbol{\theta}|\mathbf{x},1)} \ln p_O(\mathbf{x} \mid \boldsymbol{\theta}; 1).$$

Note that $L^r(\mathbf{x})$ is not directly optimised in this paper; however it is indirectly trained by optimising $L^n(\mathbf{x})$ since both are minimised by matching the output distribution to the data. Furthermore, as long as $\beta(1)$ is high enough, the input distribution at $t = 1$ will be very close to $\mathbf{x}$, making it trivial for the network to fit $p_O(\mathbf{x} \mid \boldsymbol{\theta}; 1)$.

The loss function $L(\mathbf{x})$ is defined as the total number of nats required to transmit the data, which is the sum of the n-step and reconstruction losses:

$$L(\mathbf{x}) = L^n(\mathbf{x}) + L^r(\mathbf{x}) \tag{16}$$

Loss VAE Perspective

Alternatively $L(\mathbf{x})$ can be derived as the loss function of a variational autoencoder (VAE; [18]). Consider the sequence $\mathbf{y}_1, \ldots, \mathbf{y}_n$ as a latent code with posterior probability given by

$$q(\mathbf{y}_1, \ldots, \mathbf{y}_n) = \prod_{i=1}^{n} p_S\left(\mathbf{y}_i \mid \mathbf{x}; \alpha_i\right), \tag{17}$$

and autoregressive prior probability given by

$$p(\mathbf{y}_1, \ldots, \mathbf{y}_n) = \prod_{i=1}^{n} p_R(\mathbf{y}_i \mid \boldsymbol{\theta}_{i-1}; t_{i-1}, \alpha_i). \tag{18}$$

Then, noting that the decoder probability $p(\mathbf{x} \mid \mathbf{y}_1, \ldots, \mathbf{y}_n) = p_O(\mathbf{x} \mid \boldsymbol{\theta}_n; 1)$, the complete transmission process defines a VAE with loss function given by the negative variational lower bound (VLB)

$$\begin{aligned} L(\mathbf{x}) = -\mathrm{VLB}(\mathbf{x}) &= D_{KL}\left(q \,\|\, p\right) - \mathop{\mathbb{E}}_{\mathbf{y}_1,\ldots,\mathbf{y}_n \sim q} \ln p(\mathbf{x} \mid \mathbf{y}_1, \ldots, \mathbf{y}_n) &(19)\\ &= L^n(\mathbf{x}) + L^r(\mathbf{x}). &(20)\end{aligned}$$

Bayesian Updates

Given parameters $\boldsymbol{\theta}$ and sender sample $\mathbf{y}$ drawn with accuracy α the *Bayesian update function* h is derived by applying the rules of Bayesian inference to compute the updated parameters $\boldsymbol{\theta}'$:

$$\boldsymbol{\theta}' \leftarrow h(\boldsymbol{\theta}, \mathbf{y}, \alpha). \tag{5}$$

The *Bayesian update distribution* $p_{_U}(\cdot \mid \boldsymbol{\theta}, \mathbf{x}; \alpha)$ is then defined by marginalizing out $\mathbf{y}$:

$$p_{_U}(\boldsymbol{\theta}' \mid \boldsymbol{\theta}, \mathbf{x}; \alpha) = \mathop{\mathbb{E}}_{p_{_S}(\mathbf{y} \mid \mathbf{x}; \alpha)} \delta\left(\boldsymbol{\theta}' - h(\boldsymbol{\theta}, \mathbf{y}, \alpha)\right), \tag{6}$$

where $\delta\left(\cdot - \boldsymbol{a}\right)$ is the multivariate Dirac delta distribution centred on the vector $\boldsymbol{a}$. In Sections 4.4 and 6.7 we will prove that both forms of $p_{_U}(\cdot \mid \boldsymbol{\theta}, \mathbf{x}; \alpha)$ considered in this paper have the following property: the accuracies are additive in the sense that if $\alpha = \alpha_a + \alpha_b$ then

$$p_{_U}(\boldsymbol{\theta}'' \mid \boldsymbol{\theta}, \mathbf{x}; \alpha) = \mathop{\mathbb{E}}_{p_{_U}(\boldsymbol{\theta}' \mid \boldsymbol{\theta}, \mathbf{x}; \alpha_a)} p_{_U}(\boldsymbol{\theta}'' \mid \boldsymbol{\theta}', \mathbf{x}; \alpha_b). \tag{7}$$

Bayesian Updates: Continuous

Given a univariate Gaussian prior $\mathcal{N}\left(\mu_a, \rho_a^{-1}\right)$ over some unknown data x it can be shown [27] that the Bayesian posterior after observing a noisy sample y from a normal distribution $\mathcal{N}\left(x, \alpha^{-1}\right)$ with known precision α is $\mathcal{N}\left(\mu_b, \rho_b^{-1}\right)$, where

$$\rho_b = \rho_a + \alpha, \tag{46}$$

$$\mu_b = \frac{\mu_a \rho_a + y\alpha}{\rho_b}. \tag{47}$$

Since both $p_{_I}(\mathbf{x} \mid \boldsymbol{\theta})$ and $p_{_S}(\mathbf{y} \mid \mathbf{x}; \alpha)$ distributions are normal with diagonal covariance, Eqs. 46 and 47 can be applied to obtain the following Bayesian update function for parameters $\boldsymbol{\theta}_{i-1} = \{\boldsymbol{\mu}_{i-1}, \rho_{i-1}\}$ and sender sample $\mathbf{y}$ drawn from $p_{_S}(\cdot \mid \mathbf{x}; \alpha \boldsymbol{I}) = \mathcal{N}\left(\mathbf{x}, \alpha^{-1}\boldsymbol{I}\right)$:

$$h(\{\boldsymbol{\mu}_{i-1}, \rho_{i-1}\}, \mathbf{y}, \alpha) = \{\boldsymbol{\mu}_i, \rho_i\}, \tag{48}$$

with

$$\rho_i = \rho_{i-1} + \alpha, \tag{49}$$

$$\boldsymbol{\mu}_i = \frac{\boldsymbol{\mu}_{i-1}\rho_{i-1} + \mathbf{y}\,\alpha}{\rho_i}. \tag{50}$$