Generative Models with Latent Diffusions

Xinda Wu

October 2020

Content

1. Generative Models and Variational Inference
   - designs of variational distribution q_ϕ
2. Deep Generative Models
   - deep latent Gaussian model
   - neural SDEs (inference, exact sampling, expressivity)
3. Scalable Gradients for Neural SDEs
   - algorithm and experiment

Generative Models

- A *generative model*: probabilistic, estimates $p(y)$, the probability of observing the observation y.
  (describes how data sets are generated, and by sampling from this model we can generate new data)

Generative Models

- A *generative model*: probabilistic, estimates $p(y)$, the probability of observing the observation y.
  (describes how data sets are generated, and by sampling from this model we can generate new data)
- *Discriminative models* estimate $p(l|y)$, the probability of a label l given observation y, used for classification and regression.

Generative Models

- A *generative model*: probabilistic, estimates $p(y)$, the probability of observing the observation y.
  (describes how data sets are generated, and by sampling from this model we can generate new data)
- *Discriminative models* estimate $p(l|y)$, the probability of a label l given observation y, used for classification and regression.
- assumption: $\exists\, p_{data}$ s.t. $y \sim p_{data}$
  want: a generative model $p_{model} \approx p_{data}$

Generative Models

- A *generative model*: probabilistic, estimates $p(y)$, the probability of observing the observation y.
  (describes how data sets are generated, and by sampling from this model we can generate new data)
- *Discriminative models* estimate $p(l|y)$, the probability of a label l given observation y, used for classification and regression.
- assumption: $\exists\, p_{data}$ s.t. $y \sim p_{data}$
  want: a generative model $p_{model} \approx p_{data}$
- defining a parametric family of densities $\{p_\theta\}_\theta$, we to solve the problem

$$\max_\theta \frac{1}{n} \sum_{i=1}^{n} \log p_\theta(y_i).$$

  (finding the one that maximize the likelihood on the given data $\{y_i\}_{i=1}^n$ where $p_\theta(y)$ is interpreted as the plausibility of θ given the data y)

Generative Models

- A *generative model*: probabilistic, estimates $p(y)$, the probability of observing the observation y.
  (describes how data sets are generated, and by sampling from this model we can generate new data)
- *Discriminative models* estimate $p(l|y)$, the probability of a label l given observation y, used for classification and regression.
- assumption: $\exists\, p_{data}$ s.t. $y \sim p_{data}$
  want: a generative model $p_{model} \approx p_{data}$
- defining a parametric family of densities $\{p_\theta\}_\theta$, we to solve the problem

$$\max_\theta \frac{1}{n} \sum_{i=1}^{n} \log p_\theta(y_i).$$

  (finding the one that maximize the likelihood on the given data $\{y_i\}_{i=1}^n$ where $p_\theta(y)$ is interpreted as the plausibility of θ given the data y)
- Bayesian Inference: $p_\theta(x|y) = p_\theta(y|x)p_\theta(x)/p_\theta(y)$
- Approximate Inference
  - MCMC
  - Variational Inference

Variational Inference

- Idea: use a variational distribution $q_\phi(x|y)$ to approximate the posterior, and solve the problem:

$$\min_\phi D(q_\phi(x|y)||p_\theta(x|y))$$

Variational Inference

- Idea: use a variational distribution $q_\phi(x|y)$ to approximate the posterior, and solve the problem:

$$\min_\phi D(q_\phi(x|y)||p_\theta(x|y))$$

- By the definition of KL divergence

$$\begin{aligned}
D(q_\phi(x|y)||p_\theta(x|y)) &= \mathbf{E}_{q_\phi(x|y)} \log \frac{q_\phi(x|y)p_\theta(y)}{p_\theta(x|y)p_\theta(y)} \\
&= \mathbf{E}_{q_\phi(x|y)} \log \frac{q_\phi(x|y)}{p_\theta(x,y)} + \mathbf{E}_{q_\phi(x|y)}[\log p_\theta(y)] \\
&= \mathbf{F}_{\phi,\theta}(y) + \log p_\theta(y)
\end{aligned}$$

Variational Inference

- Idea: use a variational distribution $q_\phi(x|y)$ to approximate the posterior, and solve the problem:
$$\min_\phi D(q_\phi(x|y)||p_\theta(x|y))$$
- By the definition of KL divergence

$$\begin{aligned} D(q_\phi(x|y)||p_\theta(x|y)) &= \mathbf{E}_{q_\phi(x|y)} \log \frac{q_\phi(x|y)p_\theta(y)}{p_\theta(x|y)p_\theta(y)} \\ &= \mathbf{E}_{q_\phi(x|y)} \log \frac{q_\phi(x|y)}{p_\theta(x,y)} + \mathbf{E}_{q_\phi(x|y)}[\log p_\theta(y)] \\ &= \mathbf{F}_{\phi,\theta}(y) + \log p_\theta(y) \end{aligned}$$

with the *variational free energy*

$$\mathbf{F}_{\phi,\theta}(y) := D(q_\phi(x|y)||p_\theta(x)) - \mathbf{E}_{q_\phi(x|y)}[\log p_\theta(y|x)]$$

which is computable.

Variational Inference

- Idea: use a variational distribution $q_\phi(x|y)$ to approximate the posterior, and solve the problem:

$$\min_\phi D(q_\phi(x|y)||p_\theta(x|y))$$

- By the definition of KL divergence

$$\begin{aligned}
D(q_\phi(x|y)||p_\theta(x|y)) &= \mathbf{E}_{q_\phi(x|y)} \log \frac{q_\phi(x|y)p_\theta(y)}{p_\theta(x|y)p_\theta(y)} \\
&= \mathbf{E}_{q_\phi(x|y)} \log \frac{q_\phi(x|y)}{p_\theta(x,y)} + \mathbf{E}_{q_\phi(x|y)}[\log p_\theta(y)] \\
&= \mathbf{F}_{\phi,\theta}(y) + \log p_\theta(y)
\end{aligned}$$

  with the *variational free energy*

$$\mathbf{F}_{\phi,\theta}(y) := D(q_\phi(x|y)||p_\theta(x)) - \mathbf{E}_{q_\phi(x|y)}[\log p_\theta(y|x)]$$

  which is computable.

- The problem becomes:

$$\min_\phi \mathbf{F}_{\phi,\theta}(y)$$

Variational Inference

- Idea: use a variational distribution $q_\phi(x|y)$ to approximate the posterior, and solve the problem:

$$\min_\phi D(q_\phi(x|y)||p_\theta(x|y))$$

- By the definition of KL divergence

$$\begin{aligned} D(q_\phi(x|y)||p_\theta(x|y)) &= \mathbf{E}_{q_\phi(x|y)} \log \frac{q_\phi(x|y)p_\theta(y)}{p_\theta(x|y)p_\theta(y)} \\ &= \mathbf{E}_{q_\phi(x|y)} \log \frac{q_\phi(x|y)}{p_\theta(x,y)} + \mathbf{E}_{q_\phi(x|y)}[\log p_\theta(y)] \\ &= \mathbf{F}_{\phi,\theta}(y) + \log p_\theta(y) \end{aligned}$$

with the *variational free energy*

$$\mathbf{F}_{\phi,\theta}(y) := D(q_\phi(x|y)||p_\theta(x)) - \mathbf{E}_{q_\phi(x|y)}[\log p_\theta(y|x)]$$

which is computable.

- The problem becomes:

$$\min_\phi \mathbf{F}_{\phi,\theta}(y)$$

- Also obtain the following inequality

$$-\log p_\theta(y) \leq \mathbf{F}_{\phi,\theta}(y)$$

'=' holds when $q_\phi(x|y) = p_\theta(x|y)$. The problem of MLE becomes:

$$\min_{\phi,\theta} \mathbf{F}_{\phi,\theta}(y)$$

Variational Inference

$$\mathbf{F}_{\phi,\theta}(y) = \underbrace{D(q_\phi(x|y)||p_\theta(x))}_{\text{regulariser}} \underbrace{-\mathbf{E}_{q_\phi}[\log p_\theta(y|x_k)]}_{\text{reconstruction error}}$$

Variational Inference

$$\mathbf{F}_{\phi,\theta}(y) = \underbrace{D(q_\phi(x|y)||p_\theta(x))}_{\text{regulariser}} \underbrace{-\mathbf{E}_{q_\phi}[\log p_\theta(y|x_k)]}_{\text{reconstruction error}}$$

Algorithm 2: Learning with Variational Inference

Input: parameters ϕ for variational distributions, θ for the generative model (both initialized randomly).

while *the free energy* $F_{\phi,\theta}$ *not converged* **do**

- $y \leftarrow \{\text{Get mini-batch}\}$
- compute the variational distribution q_ϕ
- sample $x \sim q_\phi(\cdot)$
- compute the free energy $F_{\phi,\theta}(y) \approx F_{\phi,\theta}(x, y)$
- $\Delta\theta \propto -\nabla_\theta F_{\phi,\theta}$
- $\Delta\phi \propto -\nabla_\phi F_{\phi,\theta}$

end

Variational Inference

$$\mathbf{F}_{\phi,\theta}(y) = \underbrace{D(q_\phi(x|y)||p_\theta(x))}_{\text{regulariser}} \underbrace{-\mathbf{E}_{q_\phi}[\log p_\theta(y|x_k)]}_{\text{reconstruction error}}$$

Algorithm 3: Learning with Variational Inference

Input: parameters ϕ for variational distributions, θ for the generative model (both initialized randomly).

while *the free energy* $F_{\phi,\theta}$ *not converged* **do**

  $y \leftarrow$ {Get mini-batch}

  compute the variational distribution q_ϕ

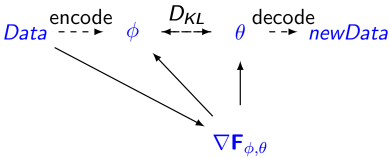
  sample $x \sim q_\phi(\cdot)$

  compute the free energy $F_{\phi,\theta}(y) \approx F_{\phi,\theta}(x, y)$

  $\Delta\theta \propto -\nabla_\theta F_{\phi,\theta}$

  $\Delta\phi \propto -\nabla_\phi F_{\phi,\theta}$

end



Variational Inference

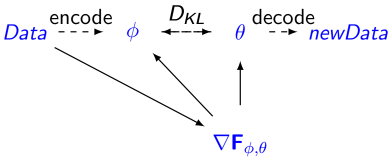
$$\mathbf{F}_{\phi,\theta}(y) = \underbrace{D(q_\phi(x|y)||p_\theta(x))}_{\text{regulariser}} \underbrace{-\mathbf{E}_{q_\phi}[\log p_\theta(y|x_k)]}_{\text{reconstruction error}}$$

Algorithm 4: Learning with Variational Inference

Input: parameters ϕ for variational distributions, θ for the generative model (both initialized randomly).

while *the free energy* $F_{\phi,\theta}$ *not converged* **do**
- $y \leftarrow$ {Get mini-batch}
- compute the variational distribution q_ϕ
- sample $x \sim q_\phi(\cdot)$
- compute the free energy $F_{\phi,\theta}(y) \approx F_{\phi,\theta}(x, y)$
- $\Delta\theta \propto -\nabla_\theta F_{\phi,\theta}$
- $\Delta\phi \propto -\nabla_\phi F_{\phi,\theta}$

end



Two problems in implementation:

- compute the gradients of free energy
- construct q_ϕ, balancing between richness and scalability

Designs of the Variational Distribution q_ϕ

Mean-field Approximation (Parisi, 1988)

- assume the latent variables to be mutually independent and the distribution q_ϕ factorizes as follows

$$q_\phi(x_0, ..., x_k) = \prod_{i=0}^{k} q_{\phi_i}(x_i)$$

Designs of the Variational Distribution q_ϕ

Mean-field Approximation (Parisi, 1988)

- assume the latent variables to be mutually independent and the distribution q_ϕ factorizes as follows

$$q_\phi(x_0, ..., x_k) = \prod_{i=0}^{k} q_{\phi_i}(x_i)$$

- cannot capture the correlation between latent variables (fig.1)
- fail to fit a multimodal posterior (fig.2)
- Improvements: parameterize the correlation; use mixture model

$$q_{mix} = \sum_i \alpha_i q_{mf}^{(i)}$$

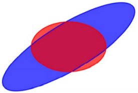


Figure 1: uncorrelated

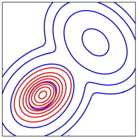


Figure 2: not fitting multi-modal

Designs of the Variational Distribution q_ϕ

Normalizing Flows (Rezende 2015)

- Idea: use a sequence of 'simple', differentiable, invertible transformations to construct an arbitrarily complex distribution

$$x_K = f_K \circ \ldots \circ f_1(x_0); \quad x_0 \sim q(x_0)$$

Designs of the Variational Distribution q_ϕ

Normalizing Flows (Rezende 2015)

- Idea: use a sequence of 'simple', differentiable, invertible transformations to construct an arbitrarily complex distribution

$$x_K = f_K \circ \ldots \circ f_1(x_0); \quad x_0 \sim q(x_0)$$

- For each mapping $f : \mathbb{R}^d \to \mathbb{R}^d$, random variable x with distribution $q(x)$, the resulting variable $x' = f(x)$ has a distribution (chage of variable)

$$q(x') = q(x) \left| \det \frac{\partial f^{-1}}{\partial x'} \right| = q(x) \left| \det \frac{\partial f}{\partial x} \right|^{-1}.$$

Designs of the Variational Distribution q_ϕ

Normalizing Flows (Rezende 2015)

- Idea: use a sequence of 'simple', differentiable, invertible transformations to construct an arbitrarily complex distribution

$$x_K = f_K \circ \ldots \circ f_1(x_0); \quad x_0 \sim q(x_0)$$

- For each mapping $f : \mathbb{R}^d \to \mathbb{R}^d$, random variable x with distribution $q(x)$, the resulting variable $x' = f(x)$ has a distribution (chage of variable)

$$q(x') = q(x) \left| \det \frac{\partial f^{-1}}{\partial x'} \right| = q(x) \left| \det \frac{\partial f}{\partial x} \right|^{-1}.$$

after K transformations

$$\ln q_K(x_K) = \ln q_0(x_0) - \sum_{i=1}^{K} \ln \left| \det \frac{\partial f_i}{\partial x_{i-1}} \right|$$

Designs of the Variational Distribution q_ϕ

Normalizing Flows (Rezende 2015)

- Idea: use a sequence of 'simple', differentiable, invertible transformations to construct an arbitrarily complex distribution

$$x_K = f_K \circ \ldots \circ f_1(x_0); \quad x_0 \sim q(x_0)$$

- For each mapping $f : \mathbb{R}^d \to \mathbb{R}^d$, random variable x with distribution $q(x)$, the resulting variable $x' = f(x)$ has a distribution (chage of variable)

$$q(x') = q(x) \left| \det \frac{\partial f^{-1}}{\partial x'} \right| = q(x) \left| \det \frac{\partial f}{\partial x} \right|^{-1}.$$

after K transformations

$$\ln q_K(x_K) = \ln q_0(x_0) - \sum_{i=1}^{K} \ln \left| \det \frac{\partial f_i}{\partial x_{i-1}} \right|$$

- *Law of the unconscious statistician* (LOTUS):
expectation w.r.t. q_K can be computed without explicitly knowing q_K

$$\mathbf{E}_{q_K}[g(x_K)] = \mathbf{E}_{q_0}[g(f_K \circ \ldots \circ f_1(x_0))]$$

Designs of the Variational Distribution q_ϕ

- Planar flow:

$$f(x) = x + \mathbf{u}h(\mathbf{w}^\intercal x + b)$$

  with $\mathbf{w} \in \mathbb{R}^d$, $\mathbf{u} \in \mathbb{R}^d$, $b \in \mathbb{R}$ parameters and $h : \mathbb{R} \to \mathbb{R}$ a smooth, element-wise nonlinear function
- Radial flow: $f(x) = x + \frac{\beta}{\alpha + r}(x - x_0)$ with $r = |x - x_0|$, $\alpha \in \mathbb{R}^+$, $\beta \in \mathbb{R}$

Designs of the Variational Distribution q_ϕ

- Planar flow:

$$f(x) = x + \mathbf{u}h(\mathbf{w}^\intercal x + b)$$

with $\mathbf{w} \in \mathbb{R}^d$, $\mathbf{u} \in \mathbb{R}^d$, $b \in \mathbb{R}$ parameters and $h : \mathbb{R} \to \mathbb{R}$ a smooth, element-wise nonlinear function

- Radial flow: $f(x) = x + \frac{\beta}{\alpha + r}(x - x_0)$ with $r = |x - x_0|$, $\alpha \in \mathbb{R}^+$, $\beta \in \mathbb{R}$
- Free energy with planar flow

$$\begin{aligned}
\mathbf{F}_{\phi,\theta}(y) :&= D(q_\phi(x|y)||p_\theta(x)) - \mathbf{E}_{q_\phi(x|y)}[\log p_\theta(y|x)] \\
&= \mathbf{E}_{q_\phi(x|y)}[\log q_\phi(x|y) - \log p_\theta(x, y)] \\
&= \mathbf{E}_{q_0(x_0)}[\ln q_K(x_K) - \log p_\theta(x_K, y)] \\
&= \mathbf{E}_{q_0(x_0)}[\ln q_0(x_0)] - \mathbf{E}_{q_0(x_0)}[\log p_\theta(x_K, y)] \\
&\quad - \mathbf{E}_{q_0(x_0)}\left[\sum_{i=1}^{K} \ln |1 + \mathbf{u}_i^\intercal \psi_i(x_{i-1})|\right]
\end{aligned}$$

with $\psi(x) = h'(\mathbf{w}^\intercal x + b)\mathbf{w}$

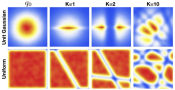


Figure 3: planarFlow

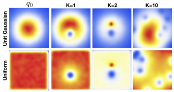


Figure 4: radialFlow

Designs of the Variational Distribution q_ϕ

- Planar flow:

$$f(x) = x + \mathbf{u}h(\mathbf{w}^\intercal x + b)$$

with $\mathbf{w} \in \mathbb{R}^d$, $\mathbf{u} \in \mathbb{R}^d$, $b \in \mathbb{R}$ parameters and $h : \mathbb{R} \to \mathbb{R}$ a smooth, element-wise nonlinear function

- Radial flow: $f(x) = x + \frac{\beta}{\alpha + r}(x - x_0)$ with $r = |x - x_0|$, $\alpha \in \mathbb{R}^+$, $\beta \in \mathbb{R}$
- Free energy with planar flow

$$\begin{aligned}
\mathbf{F}_{\phi,\theta}(y) :&= D(q_\phi(x|y)||p_\theta(x)) - \mathbf{E}_{q_\phi(x|y)}[\log p_\theta(y|x)] \\
&= \mathbf{E}_{q_\phi(x|y)}[\log q_\phi(x|y) - \log p_\theta(x, y)] \\
&= \mathbf{E}_{q_0(x_0)}[\ln q_K(x_K) - \log p_\theta(x_K, y)] \\
&= \mathbf{E}_{q_0(x_0)}[\ln q_0(x_0)] - \mathbf{E}_{q_0(x_0)}[\log p_\theta(x_K, y)] \\
&- \mathbf{E}_{q_0(x_0)}\left[\sum_{i=1}^{K} \ln |1 + \mathbf{u}_i^\intercal \psi_i(x_{i-1})|\right]
\end{aligned}$$

with $\psi(x) = h'(\mathbf{w}^\intercal x + b)\mathbf{w}$

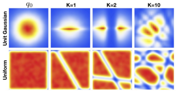


Figure 3: planarFlow

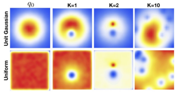


Figure 4: radialFlow

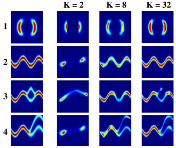


Figure 5: approx posterior using planarFlow

Deep Generative Models

Deep Latent Gaussian Model (Rezende, Kingma, 2014)

In this model, the latent variables $X_0, ..., X_k$ and the observed variable Y are generated recursively according to

$$\begin{aligned} X_0 &= Z_0 \\ X_i &= X_{i-1} + b_i(X_{i-1}) + \sigma_i Z_i,\ i = 1, ..., k \\ Y &\sim p(\cdot|X_k) \end{aligned}$$

where $Z_i \overset{i.i.d}{\sim} \mathcal{N}(0, I_d)$ in $\mathbb{R}^d$ and $b_i : \mathbb{R}^d \to \mathbb{R}^d$ parametric nonlinear transformations, $\sigma_i \in \mathbb{R}^d \times \mathbb{R}^d$ sequence of matrices, $p(\cdot|\cdot)$ observation likelihood.

- b_i, implemented by *multilayer perceptrons* (MLPs). In practice, it may select other type of neural nets, up to the purpose of usage

Deep Generative Models

Deep Latent Gaussian Model (Rezende, Kingma, 2014)

In this model, the latent variables $X_0, ..., X_k$ and the observed variable Y are generated recursively according to

$$\begin{aligned} X_0 &= Z_0 \\ X_i &= X_{i-1} + b_i(X_{i-1}) + \sigma_i Z_i, \; i = 1, ..., k \\ Y &\sim p(\cdot|X_k) \end{aligned}$$

where $Z_i \overset{i.i.d}{\sim} \mathcal{N}(0, I_d)$ in $\mathbb{R}^d$ and $b_i : \mathbb{R}^d \to \mathbb{R}^d$ parametric nonlinear transformations, $\sigma_i \in \mathbb{R}^d \times \mathbb{R}^d$ sequence of matrices, $p(\cdot|\cdot)$ observation likelihood.

- b_i, implemented by *multilayer perceptrons* (MLPs). In practice, it may select other type of neural nets, up to the purpose of usage
- designed based on the idea of *representation learning*: $\dim(X) < \dim(Y)$

Deep Generative Models

Deep Latent Gaussian Model (Rezende, Kingma, 2014)

In this model, the latent variables $X_0, ..., X_k$ and the observed variable Y are generated recursively according to

$$\begin{aligned} X_0 &= Z_0 \\ X_i &= X_{i-1} + b_i(X_{i-1}) + \sigma_i Z_i, \ i = 1, ..., k \\ Y &\sim p(\cdot|X_k) \end{aligned}$$

where $Z_i \overset{i.i.d}{\sim} \mathcal{N}(0, I_d)$ in $\mathbb{R}^d$ and $b_i : \mathbb{R}^d \to \mathbb{R}^d$ parametric nonlinear transformations, $\sigma_i \in \mathbb{R}^d \times \mathbb{R}^d$ sequence of matrices, $p(\cdot|\cdot)$ observation likelihood.

- b_i, implemented by *multilayer perceptrons* (MLPs). In practice, it may select other type of neural nets, up to the purpose of usage
- designed based on the idea of *representation learning*: $\dim(X) < \dim(Y)$
- when $k = 1$, b uses linear activation, without primitive random variables Z, it performs the *principal component analysis* (PCA)

Deep Generative Models

Deep Latent Gaussian Model (Rezende, Kingma, 2014)

In this model, the latent variables $X_0, \ldots, X_k$ and the observed variable Y are generated recursively according to

$$\begin{aligned} X_0 &= Z_0 \\ X_i &= X_{i-1} + b_i(X_{i-1}) + \sigma_i Z_i,\ i = 1, \ldots, k \\ Y &\sim p(\cdot|X_k) \end{aligned}$$

where $Z_i \overset{i.i.d}{\sim} \mathcal{N}(0, I_d)$ in $\mathbb{R}^d$ and $b_i : \mathbb{R}^d \to \mathbb{R}^d$ parametric nonlinear transformations, $\sigma_i \in \mathbb{R}^d \times \mathbb{R}^d$ sequence of matrices, $p(\cdot|\cdot)$ observation likelihood.

- b_i, implemented by *multilayer perceptrons* (MLPs). In practice, it may select other type of neural nets, up to the purpose of usage
- designed based on the idea of *representation learning*: $\dim(X) < \dim(Y)$
- when $k = 1$, b uses linear activation, without primitive random variables Z, it performs the *principal component analysis* (PCA)
- Stochastic Backpropagation (Rezende, 2014)

Deep Generative Models

Deep Latent Gaussian Model (Rezende, Kingma, 2014)

In this model, the latent variables $X_0, ..., X_k$ and the observed variable Y are generated recursively according to

$$
\begin{aligned}
X_0 &= Z_0 \\
X_i &= X_{i-1} + b_i(X_{i-1}) + \sigma_i Z_i,\ i = 1, ..., k \\
Y &\sim p(\cdot | X_k)
\end{aligned}
$$

where $Z_i \overset{i.i.d}{\sim} \mathcal{N}(0, I_d)$ in $\mathbb{R}^d$ and $b_i : \mathbb{R}^d \to \mathbb{R}^d$ parametric nonlinear transformations, $\sigma_i \in \mathbb{R}^d \times \mathbb{R}^d$ sequence of matrices, $p(\cdot|\cdot)$ observation likelihood.

- b_i, implemented by *multilayer perceptrons* (MLPs). In practice, it may select other type of neural nets, up to the purpose of usage
- designed based on the idea of *representation learning*: $\dim(X) < \dim(Y)$
- when $k = 1$, b uses linear activation, without primitive random variables Z, it performs the *principal component analysis* (PCA)
- Stochastic Backpropagation (Rezende, 2014)
- representative power increases as $k \to \infty$. To avoid $n_\theta \to \infty$, consider its diffusion limit

Generative Models with Latent Diffusions

- Consider the continuous-time limit of DLGM, the latent object becomes a d-dimensional diffusion process
$$dX_t = b(X_t, t)dt + \sigma(X_t, t)dW_t, \;\; t \in [0, 1]$$
and the observed variable $Y \sim p(\cdot|X_1)$, latent space $\mathbb{W} = C([0, 1]; \mathbb{R}^d)$ and joint density
$$p_\theta(dy, dw) = p_\theta(y|X_1(w))\mu_w(dw)dy$$

Generative Models with Latent Diffusions

- Consider the continuous-time limit of DLGM, the latent object becomes a d-dimensional diffusion process

$$dX_t = b(X_t, t)dt + \sigma(X_t, t)dW_t, \;\; t \in [0, 1]$$

  and the observed variable $Y \sim p(\cdot|X_1)$, latent space $\mathbb{W} = C([0, 1]; \mathbb{R}^d)$ and joint density

$$p_\theta(dy, dw) = p_\theta(y|X_1(w))\mu_w(dw)dy$$

- We work on the *special case* $\sigma \equiv I_d$,

$$dX_t = b(X_t, t; \theta)dt + dW_t, \;\; t \in [0, 1]$$

Generative Models with Latent Diffusions

- Consider the continuous-time limit of DLGM, the latent object becomes a d-dimensional diffusion process

$$dX_t = b(X_t, t)dt + \sigma(X_t, t)dW_t, \;\; t \in [0, 1]$$

  and the observed variable $Y \sim p(\cdot|X_1)$, latent space $\mathbb{W} = C([0, 1]; \mathbb{R}^d)$ and joint density

$$p_\theta(dy, dw) = p_\theta(y|X_1(w))\mu_w(dw)dy$$

- We work on the *special case* $\sigma \equiv I_d$,

$$dX_t = b(X_t, t; \theta)dt + dW_t, \;\; t \in [0, 1]$$

  and $b : \mathbb{R}^d \times [0, 1] \to \mathbb{R}^d$
  - implemented by neural nets, θ weight parameters
  - sufficiently well behaves (bounded, Lipschitz), admits a unique strong solution and a transition kernel

A Stochastic Control Problem

Consider the following stochastic control problem:

- **controlled diffusion process** $X^u = \{X^u_t\}_{t\in[0,1]}$ defined by

$$dX^u_t = (b(X^u_t, t; \theta) + u(X^u_t, t; \phi))dt + dW_t, \ t \in [0,1]$$

for $u : \mathbb{R}^d \times [0,1] \to \mathbb{R}^d$ measurable with $\mathbf{E}[\int_0^1 \|u_s\|^2 ds] < \infty$

A Stochastic Control Problem

Consider the following stochastic control problem:

- **controlled diffusion process** $X^u = \{X_t^u\}_{t\in[0,1]}$ defined by

$$dX_t^u = (b(X_t^u, t; \theta) + u(X_t^u, t; \phi))dt + dW_t, \ t \in [0,1]$$

  for $u : \mathbb{R}^d \times [0,1] \to \mathbb{R}^d$ measurable with $\mathbf{E}[\int_0^1 \|u_s\|^2 ds] < \infty$
- **cost-to-go** function defined in the form of variational free energy:

$$J^u(x,t) := D(\mathbf{P}^u || \mathbf{P}^0) - \mathbf{E}[\log g(X_1^u) | X_t^u = x]$$

  for each u and $g : \mathbb{R} \to (0, \infty)$ be given, where $\mathbf{P}^0 := \mathrm{Law}(X_{[t,1]})$ and $\mathbf{P}^u := \mathrm{Law}(X_{[t,1]}^u)$

A Stochastic Control Problem

Consider the following stochastic control problem:

- **controlled diffusion process** $X^u = \{X_t^u\}_{t\in[0,1]}$ defined by

$$dX_t^u = (b(X_t^u, t; \theta) + u(X_t^u, t; \phi))dt + dW_t,\ t \in [0,1]$$

  for $u : \mathbb{R}^d \times [0,1] \to \mathbb{R}^d$ measurable with $\mathbf{E}[\int_0^1 \|u_s\|^2 ds] < \infty$
- **cost-to-go** function defined in the form of variational free energy:

$$J^u(x,t) := D(\mathbf{P}^u || \mathbf{P}^0) - \mathbf{E}[\log g(X_1^u) | X_t^u = x]$$

  for each u and $g : \mathbb{R} \to (0, \infty)$ be given, where $\mathbf{P}^0 := \text{Law}(X_{[t,1]})$ and $\mathbf{P}^u := \text{Law}(X_{[t,1]}^u)$
- **Girsanov representation**
  X and X^u differ by a change of drift, by Girsanov formula

$$\frac{d\mathbf{P}^u}{d\mathbf{P}^0} = \exp\left(\int_t^1 u_s^T dW_s + \frac{1}{2}\int_t^1 \|u_s\|^2 ds\right),$$

$$D(\mathbf{P}^u || \mathbf{P}^0) = \mathbf{E}_{\mathbf{P}^u}\left[\log \frac{d\mathbf{P}^u}{d\mathbf{P}^0}\right] = \mathbf{E}\left[\frac{1}{2}\int_t^1 \|u_s\|^2 ds\right]$$

$$\Rightarrow J^u(x,t) := \mathbf{E}\left[\frac{1}{2}\int_t^1 \|u_s\|^2 ds - \log g(X_1^u) \middle| X_t^u = x\right]$$

A Stochastic Control Problem

- **Goal**: to find the value function $v : \mathbb{R}^d \times [0, 1] \to \mathbb{R}_+$

$$v(x, t) := \inf_u J^u(x, t); \; v(\cdot, 1) = -\log g(\cdot)$$

and the optimal control u^* s.t. $v(x, t) = J^{u^*}(x, t)$

A Stochastic Control Problem

- **Goal**: to find the value function $v : \mathbb{R}^d \times [0,1] \to \mathbb{R}_+$

$$v(x,t) := \inf_u J^u(x,t);\ v(\cdot,1) = -\log g(\cdot)$$

  and the optimal control u^* s.t. $v(x,t) = J^{u^*}(x,t)$
- **Bellman equation**
  By the *principle of optimality*(Bellman, 1957), from t to $t+dt$,

$$v(x,t) = \min_u \left\{ v(x,t+dt) + \mathbf{E}\left[\frac{1}{2}\int_t^{t+dt} \|u_s\|^2 ds\right]\right\}$$

A Stochastic Control Problem

- **Goal**: to find the value function $v : \mathbb{R}^d \times [0, 1] \to \mathbb{R}_+$

$$v(x, t) := \inf_u J^u(x, t);\ v(\cdot, 1) = -\log g(\cdot)$$

  and the optimal control u^* s.t. $v(x, t) = J^{u^*}(x, t)$

- **Bellman equation**
  By the *principle of optimality* (Bellman, 1957), from t to $t + dt$,

$$v(x, t) = \min_u \left\{ v(x, t + dt) + \mathbf{E}\left[\frac{1}{2} \int_t^{t+dt} \|u_s\|^2 ds \right] \right\}$$

  use Itô formula to expand $v(x, t + dt)$ and let $dt \to 0$, we have

$$\frac{\partial v}{\partial t} + \mathcal{L}_t v = -\min_{\alpha \in \mathbb{R}^d} \left\{ \alpha^\intercal \nabla v + \frac{1}{2} \|\alpha\|^2 \right\};\ v(\cdot, 1) = -\log g(\cdot)$$

A Stochastic Control Problem

- **Goal**: to find the value function $v : \mathbb{R}^d \times [0,1] \to \mathbb{R}_+$

$$v(x,t) := \inf_u J^u(x,t);\; v(\cdot,1) = -\log g(\cdot)$$

  and the optimal control u^* s.t. $v(x,t) = J^{u^*}(x,t)$

- **Bellman equation**
  By the *principle of optimality*(Bellman, 1957), from t to $t+dt$,

$$v(x,t) = \min_u \left\{ v(x,t+dt) + \mathbf{E}\left[\frac{1}{2}\int_t^{t+dt} \|u_s\|^2 ds\right]\right\}$$

  use Itô formula to expand $v(x,t+dt)$ and let $dt \to 0$, we have

$$\frac{\partial v}{\partial t} + \mathcal{L}_t v = -\min_{\alpha \in \mathbb{R}^d} \left\{ \alpha^\intercal \nabla v + \frac{1}{2}\|\alpha\|^2 \right\};\, v(\cdot,1) = -\log g(\cdot)$$

  reduced to the following Cauchy problem

$$\frac{\partial v}{\partial t} + \mathcal{L}_t v = \frac{1}{2}\|\nabla v\|^2 \text{ on } \mathbb{R}^d \times [0,1];\, v(\cdot,1) = -\log g(\cdot)$$

  which can be solved by Feynman-Kac formula.

A Stochastic Control Problem

Theorem (Jamison, 1975; Dai Pra, 1991)

Consider the above control problem, then the value function is given by

$$v(x,t) = -\log \mathbf{E}[g(X_1)|X_t = x]$$

the optimal control is given by

$$u^*(x,t) = -\nabla v(x,t) = \nabla \log \mathbf{E}[g(X_1)|X_t = x]$$

the corresponding controlled diffusion X^{u^*} has the transition density

$$\kappa^*_{s,t}(x,y) = \kappa_{s,t}(x,y)\exp(v(x,s) - v(y,t))$$

where $\kappa_{s,t}(\cdot)$ is the transition density of uncontrolled process.

A Stochastic Control Problem

Theorem (Jamison, 1975; Dai Pra, 1991)

Consider the above control problem, then the value function is given by

$$v(x,t) = -\log \mathbf{E}[g(X_1)|X_t = x]$$

the optimal control is given by

$$u^*(x,t) = -\nabla v(x,t) = \nabla \log \mathbf{E}[g(X_1)|X_t = x]$$

the corresponding controlled diffusion X^{u^*} has the transition density

$$\kappa^*_{s,t}(x,y) = \kappa_{s,t}(x,y)\exp(v(x,s) - v(y,t))$$

where $\kappa_{s,t}(\cdot)$ is the transition density of uncontrolled process.

- **Entropy Inequality**:

$$-\log \mathbf{E}[g(X_1)|X_0 = x] \le D(\mathbf{P^u}||\mathbf{P^0}) - \mathbf{E}[\log g(X_1^u)|X_0^u = x]$$

A Stochastic Control Problem

Theorem (Jamison, 1975; Dai Pra, 1991)

Consider the above control problem, then the value function is given by

$$v(x,t) = -\log \mathbf{E}[g(X_1)|X_t = x]$$

the optimal control is given by

$$u^*(x,t) = -\nabla v(x,t) = \nabla \log \mathbf{E}[g(X_1)|X_t = x]$$

the corresponding controlled diffusion X^{u^*} has the transition density

$$\kappa^*_{s,t}(x,y) = \kappa_{s,t}(x,y)\exp(v(x,s) - v(y,t))$$

where $\kappa_{s,t}(\cdot)$ is the transition density of uncontrolled process.

- **Entropy Inequality**:

$$-\log \mathbf{E}[g(X_1)|X_0 = x] \leq D(\mathbf{P^u}||\mathbf{P^0}) - \mathbf{E}[\log g(X_1^u)|X_0^u = x]$$

- **Variational upper-bound**: (let $g(x) = p(y|x)$ observation likelihood)

$$\begin{aligned} -\log \mathbf{E}[p(y|X_1)|X_0 = x] &\leq D(\mathbf{P^u}||\mathbf{P^0}) - \mathbf{E}[\log p(y|X_1^u)|X_0^u = x] \\ &= \underbrace{\mathbf{E}\left[\frac{1}{2}\int_0^1 \|u_s\|^2 ds - \log p(y|X_1^u)\middle| X_0^u = x\right]}_{\mathbf{F}^u(y;\phi,\theta) :=} \end{aligned}$$

Generative Models with Latent Diffusions

Sampling problem: Given target distribution μ, and the prior process with $X_1 \sim \nu$

- $\exists u$ s.t. $X_1^u \sim \mu$?

Generative Models with Latent Diffusions

Sampling problem: Given target distribution μ, and the prior process with $X_1 \sim \nu$

- $\exists u$ s.t. $X_1^u \sim \mu$?
- among these controls, which has the minimal control cost?

$$D(\mathbf{P}^u || \mathbf{P}^0) \to \text{min?}$$

Generative Models with Latent Diffusions

Sampling problem: Given target distribution μ, and the prior process with $X_1 \sim \nu$

- $\exists u$ s.t. $X_1^u \sim \mu$?
- among these controls, which has the minimal control cost?

$$D(\mathbf{P}^u || \mathbf{P}^0) \to \min?$$

- an optimal control s.t. encoder and decoder communicate most efficiently?

Generative Models with Latent Diffusions

Sampling problem: Given target distribution μ, and the prior process with $X_1 \sim \nu$

- $\exists u$ s.t. $X_1^u \sim \mu$?
- among these controls, which has the minimal control cost?

$$D(\mathbf{P}^u || \mathbf{P}^0) \to \min?$$

- an optimal control s.t. encoder and decoder communicate most efficiently?

Generative Models with Latent Diffusions

Sampling problem: Given target distribution μ, and the prior process with $X_1 \sim \nu$

- $\exists u$ s.t. $X_1^u \sim \mu$?
- among these controls, which has the minimal control cost?

$$D(\mathbf{P}^u || \mathbf{P}^0) \to \min?$$

- an optimal control s.t. encoder and decoder communicate most efficiently?

Theorem

Given a target μ at $t = 1$, $X_0 = 0$, $X_1 \sim \nu$, with $\mu \ll \nu$. Let $g = f = d\mu/d\nu$, then

$$X_1^* \sim \mu$$

and the optimal control u^* has the minimal energy

$$D(\mathbf{P^u} || \mathbf{P^0}) \geq D(d\mu || d\nu)$$

among all admissible controls u that induce the target distribution μ at $t = 1$

Generative Models with Latent Diffusions

Proof:

- $X_1^* \sim \mu$

$$\begin{aligned}\mathbb{P}[X_1^* \in A] &= \int_A \kappa^*_{0,1}(0,y)dy \\ &= \int_A \kappa_{0,1}(0,y)\exp(v(0,0) - v(y,1))dy \\ &= \int_A f\, d\nu \\ &= \mu(A)\end{aligned}$$

Generative Models with Latent Diffusions

Proof:

- $X_1^* \sim \mu$

$$\begin{aligned}\mathbb{P}[X_1^* \in A] &= \int_A \kappa^*_{0,1}(0,y)dy \\ &= \int_A \kappa_{0,1}(0,y)\exp(v(0,0) - v(y,1))dy \\ &= \int_A f\, d\nu \\ &= \mu(A)\end{aligned}$$

- Recall the entropy inequality

$$-\log \underbrace{\mathbf{E}[f(X_1)|X_0 = 0]}_{=\mathbf{E}_\nu[\frac{d\mu}{d\nu}]=1} \leq D(\mathbf{P^u}||\mathbf{P^0}) - \underbrace{\mathbf{E}[\log f(X_1^u)|X_0^u = 0]}_{=\mathbf{E}_\mu[\log \frac{d\mu}{d\nu}] = D(d\mu||d\nu)}$$

$$\implies D(\mathbf{P^u}||\mathbf{P^0}) \geq D(d\mu||d\nu)$$

$$\implies \underbrace{D(d\mu||d\nu)}_{\text{minimal energy}} = \min_u\{\tfrac{1}{2}\mathbf{E}[\int_0^1 \|u_s\|^2 ds]\}$$

Generative Models with Latent Diffusions

Fix $b(x,t) \equiv 0$, $X_1 \sim \gamma_d$,

- compute the value function

$$\begin{aligned} v(x,t) &= -\log \mathbf{E}[g(X_1)|X_t = x] = -\log \mathbf{E}[f(W_1)|X_t = x] \\ &= -\log \left((2\pi(1-t))^{-d/2} \int_t^1 f(y) \exp(-\frac{\|y-x\|^2}{2(1-t)}) dy \right) \\ &= -\log Q_{1-t} f(x) \end{aligned}$$

- conpute the optimal control

$$u^*(x,t) = -\nabla v(x,t) = \underbrace{\nabla \log Q_{1-t} f(x)}_{\text{Föllmer drift}}$$

Expressivity: could we approximate Föllmer drift by neural nets?

Expressivity: could we approximate Föllmer drift by neural nets?

- replace $Q_t f(x)$ by Monte Carlo estimate
- replace $f(\cdot)$ by a neural net approximation $\hat{f}(\cdot;\theta)$
- elementary operations(i.e. gradients,multiplications, reciprocals) can be computed by neural nets

$$\nabla \log Q_{1-t} f(x) \approx \nabla \log \left\{ \frac{1}{N} \sum_{n=1}^{N} \hat{f}(x + \sqrt{1-t} z_n; \theta) \right\}$$
$$= \frac{\sum_{n=1}^{N} \nabla \hat{f}(x + \sqrt{1-t} z_n; \theta)}{\sum_{n=1}^{N} \hat{f}(x + \sqrt{1-t} z_n; \theta)}$$

Generative Models with Latent Diffusions

- **(A1)** f differentiable, both f and ∇f are Lipschitz, and exits a constant $c \in (0, 1]$ such that $f > c$ everywhere

Generative Models with Latent Diffusions

- **(A1)** f differentiable, both f and ∇f are Lipschitz, and exits a constant $c \in (0, 1]$ such that $f > c$ everywhere
- (*regularity of the Föllmer drift*) With assumption $(A1)$, the Föllmer drift $b(x, t) = \nabla \log Q_{1-t} f(x)$ is bounded in the Euclidean norm,

$$\|\nabla \log Q_{1-t} f(x)\| \leq \frac{L}{c}$$

for $x \in \mathbb{R}^d$, $t \in [0, 1]$, and L the maximum of the Lipschitz constants of f and ∇f. And also, it is Lipschitz with Lipschitz constant $L/c + L^2/c^2$,

$$\|b(x, t) - b(x', t)\| \leq \left(\frac{L}{c} + \frac{L^2}{c^2}\right) \|x - x'\|.$$

Generative Models with Latent Diffusions

- **(A1)** f differentiable, both f and ∇f are Lipschitz, and exits a constant $c \in (0, 1]$ such that $f > c$ everywhere
- (*regularity of the Föllmer drift*) With assumption $(A1)$, the Föllmer drift $b(x, t) = \nabla \log Q_{1-t} f(x)$ is bounded in the Euclidean norm,

$$\|\nabla \log Q_{1-t} f(x)\| \leq \frac{L}{c}$$

  for $x \in \mathbb{R}^d$, $t \in [0, 1]$, and L the maximum of the Lipschitz constants of f and ∇f. And also, it is Lipschitz with Lipschitz constant $L/c + L^2/c^2$,

$$\|b(x, t) - b(x', t)\| \leq \left(\frac{L}{c} + \frac{L^2}{c^2} \right) \|x - x'\|.$$

- **(A2)** activation function σ is differentiable and *universal* (any univariate Lipschitz function on a bounded interval can approximated well by a 2-layer MLP)

Generative Models with Latent Diffusions

- **(A1)** f differentiable, both f and ∇f are Lipschitz, and exits a constant $c \in (0, 1]$ such that $f > c$ everywhere
- (*regularity of the Föllmer drift*) With assumption $(A1)$, the Föllmer drift $b(x, t) = \nabla \log Q_{1-t} f(x)$ is bounded in the Euclidean norm,

$$\|\nabla \log Q_{1-t} f(x)\| \leq \frac{L}{c}$$

for $x \in \mathbb{R}^d$, $t \in [0, 1]$, and L the maximum of the Lipschitz constants of f and ∇f. And also, it is Lipschitz with Lipschitz constant $L/c + L^2/c^2$,

$$\|b(x, t) - b(x', t)\| \leq \left(\frac{L}{c} + \frac{L^2}{c^2}\right) \|x - x'\|.$$

- **(A2)** activation function σ is differentiable and *universal* (any univariate Lipschitz function on a bounded interval can approximated well by a 2-layer MLP)
- **(A3)** both f and ∇f can be efficiently approximated by a neural net on any compact subset of $\mathbb{R}^d$

Generative Models with Latent Diffusions

Theorem (Tzen 2019)

Suppose assumptions $A1$-$A3$ are in force. Let L denote the maximum of Lipschitz constants of f and ∇f. Then for any $0 < \epsilon < 16L^2/c^2$, there exists a neural net $\hat{v} : \mathbb{R}^d \times [0,1] \to \mathbb{R}^d$ with size polynomial in $1/\epsilon, d, L, c, 1/c$, and the following holds:
If $\{\hat{X}_t\}_{t\in[0,1]}$ is the diffusion process governed by the Itô SDE

$$d\hat{X}_t = \hat{b}(\hat{X}_t, t)dt + dW_t,\ \hat{X}_0 = 0$$

with the drift $\hat{b}(x,t) = \hat{v}(x, \sqrt{1-t})$, then $\hat{\mu} := \mathrm{Law}(\hat{X}_1)$ satisfies $D(\mu||\hat{\mu}) \leq \epsilon$.

Generative Models with Latent Diffusions

Theorem (Tzen 2019)

Suppose assumptions $A1$-$A3$ are in force. Let L denote the maximum of Lipschitz constants of f and ∇f. Then for any $0 < \epsilon < 16L^2/c^2$, there exists a neural net $\hat{v} : \mathbb{R}^d \times [0,1] \to \mathbb{R}^d$ with size polynomial in $1/\epsilon, d, L, c, 1/c$, and the following holds:
If $\{\hat{X}_t\}_{t\in[0,1]}$ is the diffusion process governed by the Itô SDE

$$d\hat{X}_t = \hat{b}(\hat{X}_t, t)dt + dW_t,\ \hat{X}_0 = 0$$

with the drift $\hat{b}(x,t) = \hat{v}(x, \sqrt{1-t})$, then $\hat{\mu} := \mathrm{Law}(\hat{X}_1)$ satisfies $D(\mu||\hat{\mu}) \le \epsilon$.

Proof:
Using probabilistic method and results from theory of empirical process to control the error incurred in each of the following steps:

- replace $Q_t f(x)$ by Monte Carlo estimate
- replace $f(\cdot)$ by a neural net approximation $\hat{f}(\cdot;\theta)$
- approximate the elementary operations by neural nets

Trainability

Consider the following problem of sensitivity analysis: (Gobet and Munos, 2005)

Given a d-dimensional Itô process,

$$dX_t^\alpha = b(X_t^\alpha, t; \alpha)dt + \sigma(X_t^\alpha, t; \alpha)dW_t;\ t \in [0, T]$$

where α is a n_α-dimensional parameter, and a function $\mathcal{L} : \mathbb{R}^d \to \mathbb{R}$ is given. We want to compute the gradient of the expectation

$$J(\alpha) := \mathbf{E}[\mathcal{L}(X_T^\alpha)]$$

w.r.t α.
Note: for Neural SDE, $\alpha = (\phi, \theta)$, $\mathcal{L}(\cdot) = \mathbf{F}^u(y; \phi, \theta)(\cdot)$

Trainability

Consider the following problem of sensitivity analysis: (Gobet and Munos, 2005)

Given a d-dimensional Itô process,

$$dX_t^\alpha = b(X_t^\alpha, t; \alpha)dt + \sigma(X_t^\alpha, t; \alpha)dW_t;\ t \in [0, T]$$

where α is a n_α-dimensional parameter, and a function $\mathcal{L} : \mathbb{R}^d \to \mathbb{R}$ is given. We want to compute the gradient of the expectation

$$J(\alpha) := \mathbf{E}[\mathcal{L}(X_T^\alpha)]$$

w.r.t α.
Note: for Neural SDE, $\alpha = (\phi, \theta)$, $\mathcal{L}(\cdot) = \mathbf{F}^u(y; \phi, \theta)(\cdot)$

- **Path-wise forward**

$$\nabla_\alpha J = \nabla_\alpha \mathbf{E}[\mathcal{L}(X_T^\alpha)] = \mathbf{E}[\nabla \mathcal{L}(X_T^\alpha)\nabla_\alpha X_T^\alpha]$$

$$\frac{\partial X_t}{\partial \alpha_i} = \int_0^t \left(\frac{\partial b_s}{\partial \alpha_i} + \frac{\partial b_s}{\partial x}\frac{\partial X_s}{\partial \alpha_i} \right) ds + \sum_{l=1}^d \int_0^t \left(\frac{\partial \sigma_{s,l}}{\partial \alpha_i} + \frac{\partial \sigma_{s,l}}{\partial x}\frac{\partial X_s}{\partial \alpha_i} \right) dW_s^l$$

state sensitivities $\nabla_\alpha X_t$ will be computationally prohibitive as n_α increases largely

Adjoint State Sensitivity

- **Idea:**
  - mimic the *standard back-propagation* of neural net

$$\frac{d\mathcal{L}}{dh_t} = \frac{d\mathcal{L}}{dh_{t+1}} \frac{dh_{t+1}}{dh_t}$$

  - define the *adjoint state* $a_t = d\mathcal{L}/dx_t$
  - derive a *backward dynamic* for the adjoint state which shows how the gradient propagate backward
  - treating the parameters as additional part of the augmented state

Adjoint State Sensitivity

- **Idea:**
  - mimic the *standard back-propagation* of neural net

$$\frac{d\mathcal{L}}{dh_t} = \frac{d\mathcal{L}}{dh_{t+1}} \frac{dh_{t+1}}{dh_t}$$

  - define the *adjoint state* $a_t = d\mathcal{L}/dx_t$
  - derive a *backward dynamic* for the adjoint state which shows how the gradient propagate backward
  - treating the parameters as additional part of the augmented state
- **Advantages**:
  - adjoint state does **not** depend on parameters
  - the gradients w.r.t parameters can be computed using adjoint state sensitivity
  - "$n_\alpha + 1$ systems " $\rightarrow$ "**2** systems"

Adjoint State Sensitivity

Some backward calculus: (Kunita, 2019)

- **Two-sided filtration** $\{\mathcal{F}_{s,t}\}_{s\le t;\, s,t\in\mathbb{T}}$,

$$\mathcal{F}_{s,t} := \sigma(W_u - W_v : s \le u < v \le t),\ s,t \in \mathbb{T}(:= [0,T])$$

- **Backward Wiener process** $\{\check{W}_t\}_{t\in\mathbb{T}}$,

$$\check{W}_t = W_t - W_T,\ t \in \mathbb{T}$$

  which is adapted to the backward filtration $\{\mathcal{F}_{t,T}\}_{t\in\mathbb{T}}$

- **Backward Stratonovich integrals**
  for continuous semimartingale $\check{Y}_t$ adapted to the backward filtration, define(in the L^2 sense)

$$\int_s^T \check{Y}_t \circ d\check{W}_t = \lim_{|\Pi|\to 0} \sum_{k=1}^{N} \frac{1}{2}\left(\check{Y}_{t_k} + \check{Y}_{t_{k-1}}\right)\left(\check{W}_{t_{k-1}} - \check{W}_{t_k}\right)$$

- **Stratonovich SDE**

$$Z_T = z_0 + \int_0^T b(Z_t,t)dt + \sum_{i=1}^{m} \int_0^T \sigma_i(Z_t,t) \circ dW_t^{(i)}$$

  with $b,\ \sigma \in C_b^{\infty,1}$ so that the SDE has unique strong solution.

Adjoint State Sensitivity

- $\Phi_{s,t}(z) := Z_t^{s,z}$, the solution at time t when the process started at z at time s. Given a realization of the Wiener process, this defines

$$\mathcal{S} = \{\Phi_{s,t}\}_{s \leq t; s,t \in \mathbb{T}}$$

a collection of continuous maps from $\mathbb{R}^d$ to itself, satisfying

$$\Phi_{s,u} = \Phi_{t,u} \circ \Phi_{s,t}\,, \;\; s < t < u$$

Adjoint State Sensitivity

- $\Phi_{s,t}(z) := Z_t^{s,z}$, the solution at time t when the process started at z at time s. Given a realization of the Wiener process, this defines

$$\mathcal{S} = \{\Phi_{s,t}\}_{s \le t; s,t \in \mathbb{T}}$$

a collection of continuous maps from $\mathbb{R}^d$ to itself, satisfying

$$\Phi_{s,u} = \Phi_{t,u} \circ \Phi_{s,t}\ ,\ \ s < t < u$$

- **Stochastic flow of diffeomorphisms** [Thm 3.7.1 (Kunita, 2019)]
  - each $\Phi_{s,t}$ is a diffeomorphism $\mathbb{R}^d \to \mathbb{R}^d$
  - $\check{\Psi}_{s,t} := \Phi_{s,t}^{-1}$, the *inverse flow* satisfies the backward SDE:

$$\check{\Psi}_{s,t}(z) = z - \int_s^t b(\check{\Psi}_{s,t}(z), u)du - \sum_{i=1}^m \int_s^t \sigma_i(\check{\Psi}_{s,t}(z), u) \circ d\check{W}_u^{(i)}$$

  for $z \in \mathbb{R}^d$, $s, t \in \mathbb{T}$, $s \le t$.

Adjoint State Sensitivity

Recall from $\frac{d\mathcal{L}}{dh_t} = \frac{d\mathcal{L}}{dh_{t+1}} \frac{dh_{t+1}}{dh_t}$

- $A_{s,t}(z) := \nabla(\mathcal{L}(\Phi_{s,t}(z))) = \nabla\mathcal{L}(\Phi_{s,t}(z))\nabla\Phi_{s,t}(z)$
- $\check{A}_{s,T}(z) := A_{s,T}(\check{\Psi}_{s,T}(z)) = \underbrace{\nabla\mathcal{L}(z)}_{\check{A}_T}\underbrace{\nabla\Phi_{s,T}(\check{\Psi}_{s,T}(z))}_{\partial Z_T/\partial Z_s}$
- $J_{s,t}(z) := \nabla\check{\Psi}_{s,t}(z) = \partial Z_s/\partial Z_t$ satisfies,

$$J_{s,t}(z) = I_d - \int_s^t \nabla b(\check{\Psi}_{r,t}(z), r)J_{r,t}(z)dr \\ - \sum_{i=1}^{m}\int_s^t \nabla\sigma_i(\check{\Psi}_{r,t}(z), r)J_{r,t}(z) \circ d\check{W}_r^{(i)}$$

- $K_{s,t}(z) := J_{s,t}(z)^{-1} = \nabla\Phi_{s,t}(\check{\Psi}_{s,t}(z)) = \partial Z_t/\partial Z_s$ satisfies,

$$K_{s,t}(z) = I_d + \int_s^t K_{r,t}(z)\nabla b(\check{\Psi}_{r,t}(z), r)dr \\ + \sum_{i=1}^{m}\int_s^t K_{r,t}(z)\nabla\sigma_i(\check{\Psi}_{r,t}(z), r) \circ d\check{W}_r^{(i)}$$

Adjoint State Sensitivity

- $\check{A}_{s,T}(z) = \check{A}_T K_{s,T}(z) = \nabla\mathcal{L}(z)\nabla\Phi_{s,T}(\check{\Psi}_{s,T}(z))$

$$\begin{aligned}\check{A}_{s,T}(z) = \nabla\mathcal{L}(z) &+ \int_s^T \check{A}_{r,T}(z)^\intercal \nabla b(\check{\Psi}_{r,T}(z), r)dr \\ &+ \sum_{i=1}^m \int_s^T \check{A}_{r,T}(z)^\intercal \nabla\sigma_i(\check{\Psi}_{r,T}(z), r) \circ d\check{W}_r^{(i)}\end{aligned}$$

- consider augmented state $Y_t := (Z_t, \alpha)$, which satisfies a Stratonovich SDE with the drift function $\tilde{b}(y,t) = (b(z,t), \mathbf{0}_{n_\alpha})$ and the diffusion function $\tilde{\sigma}_i(y,t) = (\sigma_i(y,t), \mathbf{0}_{n_\alpha})$
- write the backward SDE for augmented adjoint $\check{A}^y := (\check{A}^z, \check{A}^\alpha)$ separately, we get the total gradient w.r.t parameters

$$\begin{aligned}\check{A}^\alpha_{s,T}(z) = \nabla_\alpha\mathcal{L}(z) &+ \int_s^T \check{A}^z_{r,T}(z)^\intercal \nabla_\alpha b(\check{\Psi}_{r,T}(z), r)dr \\ &+ \sum_{i=1}^m \int_s^T \check{A}^z_{r,T}(z)^\intercal \nabla_\alpha\sigma_i(\check{\Psi}_{r,T}(z), r) \circ d\check{W}_r^{(i)}\end{aligned}$$

Adjoint State Sensitivity

Algorithm 5: Stochastic Ajoint Sensitivity (Stratonovich)

Input: parameters α, start time t_0, stop time t_1, final state z_{t_1}, observation gradient $\partial\mathcal{L}/z_{t_1}$, drift function $b(z,t,\alpha)$, diffusion function $\sigma(z,t,\alpha)$, Wiener process sample path $w(t)$.

def *augmented drift* $\bar{b}(z_t, a_t, t, \alpha)$**:**

  return $[-b(z_t, -t, \alpha),\ a_t^\intercal \partial b/\partial z,\ a_t^\intercal \partial b/\partial\alpha]$

def *augmented diffusion* $\bar{\sigma}(z_t, a_t, t, \alpha)$**:**

  return $[-\sigma_i(z_t, -t, \alpha),\ a_t^\intercal \partial\sigma_i/\partial z,\ a_t^\intercal \partial\sigma_i/\partial\alpha]$

def *replicated noise* $\bar{w}(t)$**:**

  return $[-w(-t),\ -w(-t),\ -w(-t)]$

$$\begin{bmatrix} z_{t_0} \\ \partial\mathcal{L}/\partial z_{t_0} \\ \partial\mathcal{L}/\partial\alpha \end{bmatrix} = \texttt{SDESolver}\left(\begin{bmatrix} z_{t_1} \\ \partial\mathcal{L}/\partial z_{t_1} \\ \mathbf{0}_{n_\alpha} \end{bmatrix}, \bar{b}, \bar{\sigma}, \bar{w}, -t_1, -t_0\right)$$

return $\partial\mathcal{L}/\partial z_{t_0}$, $\partial\mathcal{L}/\partial\alpha$

Adjoint State Sensitivity

Algorithm 6: Stochastic Ajoint Sensitivity (Stratonovich)

Input: parameters α, start time t_0, stop time t_1, final state z_{t_1}, observation gradient $\partial\mathcal{L}/z_{t_1}$, drift function $b(z,t,\alpha)$, diffusion function $\sigma(z,t,\alpha)$, Wiener process sample path $w(t)$.

def *augmented drift* $\bar{b}(z_t, a_t, t, \alpha)$**:**
  return $[-b(z_t, -t, \alpha),\ a_t^{\intercal}\partial b/\partial z,\ a_t^{\intercal}\partial b/\partial\alpha]$
def *augmented diffusion* $\bar{\sigma}(z_t, a_t, t, \alpha)$**:**
  return $[-\sigma_i(z_t, -t, \alpha),\ a_t^{\intercal}\partial\sigma_i/\partial z,\ a_t^{\intercal}\partial\sigma_i/\partial\alpha]$
def *replicated noise* $\bar{w}(t)$**:**
  return $[-w(-t),\ -w(-t),\ -w(-t)]$

$$\begin{bmatrix} z_{t_0} \\ \partial\mathcal{L}/\partial z_{t_0} \\ \partial\mathcal{L}/\partial\alpha \end{bmatrix} = \texttt{SDESolver}\left(\begin{bmatrix} z_{t_1} \\ \partial\mathcal{L}/\partial z_{t_1} \\ \mathbf{0}_{n_\alpha} \end{bmatrix}, \bar{b}, \bar{\sigma}, \bar{w}, -t_1, -t_0\right)$$

return $\partial\mathcal{L}/\partial z_{t_0},\ \partial\mathcal{L}/\partial\alpha$

Table 1: L denotes the numbers of steps in the SDE solving. n_α is the dimension of the parameter, and d is the dimension of the system state.

Method	Memory	Time
Path-wise Forward Sensitivity	$\mathcal{O}(1)$	$\mathcal{O}(L \cdot (n_\alpha + d))$
Adjoint Sensitivity	$\mathcal{O}(1)$	$\mathcal{O}(L)$

Implementation

- parameterize the prior and the approximate posterior using SDEs

$$dZ_t = b_\theta(Z_t, t)dt + \sigma(Z_t, t)dW_t \quad (prior)$$
$$d\tilde{Z}_t = \tilde{b}_\phi(\tilde{Z}_t, t)dt + \sigma(\tilde{Z}_t, t)dW_t \quad (approx.post.)$$

sharing the same diffusion function σ, with $Z_0 = \tilde{Z}_0 = z_0 \in \mathbb{R}^d$, and $u : \mathbb{R}^d \times [0, T] \to \mathbb{R}^m$ satisfies $b_\phi(z, t) - b_\theta(z, t) = \sigma(z, t)u(z, t)$.

Implementation

- parameterize the prior and the approximate posterior using SDEs

$$
\begin{aligned}
dZ_t &= b_\theta(Z_t, t)dt + \sigma(Z_t, t)dW_t \quad (prior) \\
d\tilde{Z}_t &= \tilde{b}_\phi(\tilde{Z}_t, t)dt + \sigma(\tilde{Z}_t, t)dW_t \quad (approx.post.)
\end{aligned}
$$

  sharing the same diffusion function σ, with $Z_0 = \tilde{Z}_0 = z_0 \in \mathbb{R}^d$, and $u : \mathbb{R}^d \times [0, T] \to \mathbb{R}^m$ satisfies $b_\phi(z, t) - b_\theta(z, t) = \sigma(z, t)u(z, t)$.

- Modelling time series: use a recurrent neural net (RNN) to encode the posterior process, MLP to implement the prior process and KL divergence penalty for parameter update

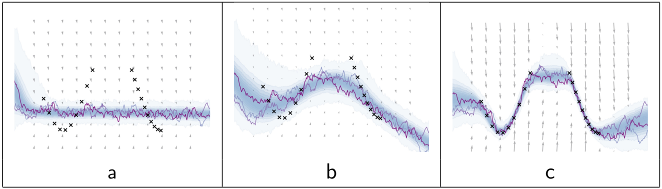
$$
\begin{tikzcd}
Data \arrow[r, dashed, "\text{RNN encoding}"] \arrow[drr, "\text{adj}"'] & \phi \arrow[r, dashed, leftrightarrow, "D_{KL}"] & \theta \arrow[r, dashed, "\text{decode}"] & newData \\
& & \nabla \mathbf{F}_{\phi,\theta} \arrow[ul] \arrow[u] &
\end{tikzcd}
$$

Experiments

- Presetting a stochastic process, and using simulation data to play neural SDEs

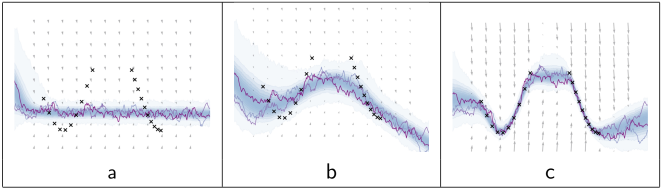
Experiments

- Presetting a stochastic process, and using simulation data to play neural SDEs
- fitting to a Ornstein-Uhlenbeck process



Experiments

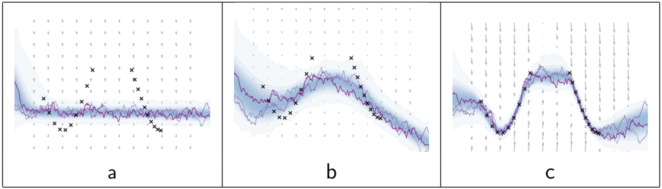
- Presetting a stochastic process, and using simulation data to play neural SDEs
- fitting to a Ornstein-Uhlenbeck process



- Possible issues in practice:
  - SDE solvers may not be working well
  - overfitting
  - KL divergence may not be well-defined or constantly 0, when we deal with high dimensional data supported in low dimensional sub-manifold

Experiments

- Presetting a stochastic process, and using simulation data to play neural SDEs
- fitting to a Ornstein-Uhlenbeck process



- Possible issues in practice:
  - SDE solvers may not be working well
  - overfitting
  - KL divergence may not be well-defined or constantly 0, when we deal with high dimensional data supported in low dimensional sub-manifold

- Unofficial package from Google Research:
  https://github.com/google-research/torchsde

References

- D. J. Rezende, S. Mohamed. *Variational Inference with Normalizing Flows*. Proceedings of the 32nd International Conference on Machine Learning, PMLR 37:1530-1538, 2015.
- D. J. Rezende, S. Mohamed, D. Wierstra. *Stochastic Backpropagation and Approximate Inference in Deep Generative Models*. Proceedings of the 31st International Conference on Machine Learning, PMLR 32(2):1278-1286, 2014.
- D. P. Kingma, M. Welling, *Auto-Encoding Variational Bayes*, International Conference on Learning Representations, 2014.
- B. Tzen, M. Raginsky, *Theoretical guarantees for sampling and inference in generative models with latent diffusions*, Proceedings of Machine Learning Research vol 99:1–31, 2019.
- Xuechen Li, Ting-Kam Leonard, Ricky T.Q. Chen, David Duvenaud, *Scalable Gradients and Variational Inference for Stochastic Differential Equation*, Proceedings of The 2nd Symposium on Advances in Approximate Bayesian Inference, PMLR 118:1-28, 2020.

Thank you!