Statistical Models & Computing Methods

Lecture 10: Advanced VI

Cheng Zhang

School of Mathematical Sciences, Peking University

December 10, 2020

Introduction

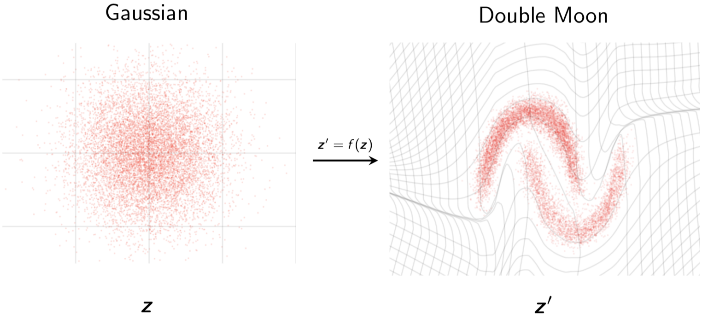
- The approximation accuracy of VI depends on the expressive power of the approximating distributions.
- Ideally, we want a rich variational family of distributions that provide accurate approximation while maintaining the compuational efficiency and scalability.
- In this lecture, we will discuss some recent techniques for improving the flexibility of variational approximations.
- We will also talk about methods that combine MCMC and VI for the best of both worlds, and some non-parameteric VI methods.

Simple Distributions is Not Enough

- VI requires the approximating distributions to have the following properties
  - Analytic density
  - Easy to sample
- Many simple distributions satisfy the above properties, e.g., Gaussian, general exponential family distributions. Therefore, they are commonly used in VI.
- Unfortunately, the posterior distribution could be much more complex (highly skewed, multi-modal, etc).
- How can we improve the complexity of our variational approximations while maintaining the desired properties?

Improve flexibility via Transforms

- Idea: Map simple distributions to complex distributions via learnable transforms.



Change of Variable Formula

Change of Variables

Assume that the mapping between z and x, given by $f : \mathbb{R}^n \to \mathbb{R}^n$, is invertible such that $x = f(z)$ and $z = f^{-1}(x)$

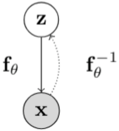
$$p_x(x) = p_z(f^{-1}(x)) \left| \det \left(\frac{\partial f^{-1}(x)}{\partial x} \right) \right|$$

- x, z need to be continuous and have the same dimension. For example, if $x \in \mathbb{R}^n$ then $z \in \mathbb{R}^n$
- For any invertible matrix A, $\det(A^{-1}) = \det(A)^{-1}$

$$p_x(x) = p_z(z) \left| \det \left(\frac{\partial f(z)}{\partial z} \right) \right|^{-1}$$

Normalizing Flow Models

- Consider a directed, latent-variable model over observed variables x and latent variables z.
- In a normalizing flow model, the mapping between z and x, given by $f_\theta : \mathbb{R}^n \mapsto \mathbb{R}^n$, is deterministic and invertible such that $x = f_\theta(z)$ and $z = f_\theta^{-1}(x)$



- Using change of variables, the probability $p(x)$ is given by

$$p_x(x|\theta) = p_z(z) \left| \det \left(\frac{\partial f_\theta(z)}{\partial z} \right) \right|^{-1}$$

Normalizing Flow Models

- **Normalizing Transforms**: Change of variables gives a normalized density after applying an invertible transformation
- **Flow**: Invertible transformations can be composed with each other

$$z_k = f_k(z_{k-1}), \quad k = 1, \ldots, K$$

- The log-likelihood of z_K

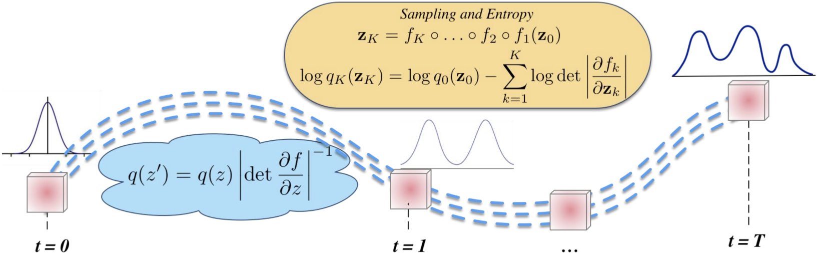
$$\log p_K(z_K) = \log p_0(z_0) - \sum_{k=1}^{K} \log \left| \det \left(\frac{\partial f_k(z_{k-1})}{z_{k-1}} \right) \right|$$

Remark: for simplicity, we omit the parameters for each of these transformations $f_1, f_2, \ldots, f_K$.

Normalizing Flows

Exploit the rule for change of variables

- Start with a simple distribution for z_0 (e.g., Gaussian).
- Apply a sequence of K invertible transformations.


Sampling and Entropy
$\mathbf{z}_K = f_K \circ \ldots \circ f_2 \circ f_1(\mathbf{z}_0)$
$\log q_K(\mathbf{z}_K) = \log q_0(\mathbf{z}_0) - \sum_{k=1}^{K} \log \det \left| \frac{\partial f_k}{\partial \mathbf{z}_k} \right|$
$q(z') = q(z) \left| \det \frac{\partial f}{\partial z} \right|^{-1}$
$t = 0$
$t = 1$
...
$t = T$

Distribution flows through a sequence of invertible transforms

Adapted from Mohamed and Rezenda, 2017

Planar Flows

- Planar flow (Rezende and Mohamed, 2015).

  $$x = f_\theta(z) = z + uh(w^\top z + b)$$

  parameterized by $\theta = (w, u, b)$ where h is a non-linear function
- Absolute value of the determinant of the Jacobian

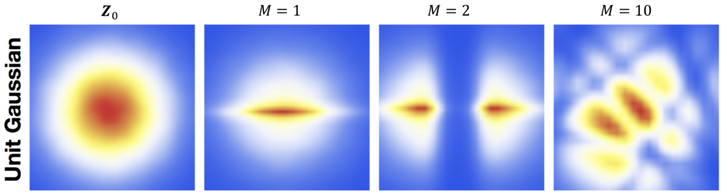
  $$\begin{aligned}\left|\det \frac{\partial f_\theta(z)}{\partial z}\right| &= \left|\det(I + h'(w^\top z + b)uw^\top)\right| \\ &= \left|1 + h'(w^\top z + b)u^\top w\right|\end{aligned}$$

- Need to restrict parameters and non-linearity for the mapping to be invertible. For example,

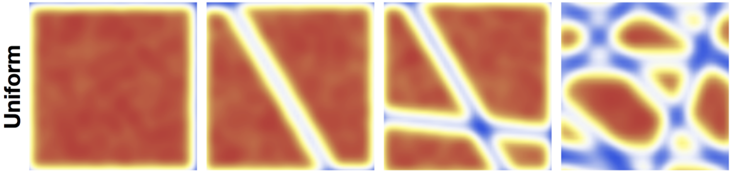
  $$h(\cdot) = \tanh(\cdot), \quad h'(w^\top z + b)u^\top w \geq -1$$

Planar Flows

- Base distribution: Gaussian



- Base distribution: Uniform



- 10 planar transformations can transform simple distributions into a more complicated one.

VI with Normalizing Flows

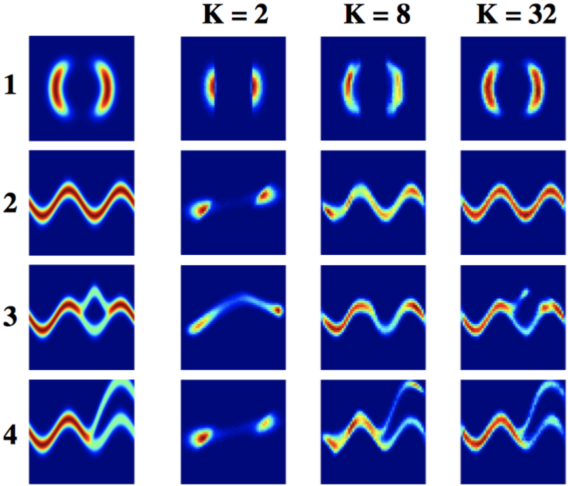
- Learning via maximizing the ELBO

$$
\begin{aligned}
L &= \mathbb{E}_{q_K(z_K)} \log \frac{p(x, z_K)}{q_K(z_K)} \\
&= \mathbb{E}_{q_0(z_0)} \log p(x, z_K) - \mathbb{E}_{q_0(z_0)} \log q_0(z_0) \\
&\qquad - \sum_{k=1}^{K} \mathbb{E}_{q_0(z_0)} \log \left| \det \left(\frac{\partial f_k(z_{k-1})}{\partial z_{k-1}} \right) \right|
\end{aligned}
$$

- Exact likelihood evaluation via inverse transformation and change of variable formula
- Sampling via forward transformation

$$z_0 \sim q_0(z_0), \quad z_K = f_K \circ f_{K-1} \circ \cdots \circ f_1(z_0)$$

VI with Normalizing Flows



Adapted from Rezenda and Mohamed, 2015

Requirements for Normalizing Flows

- Simple initial distribution $q_0(z_0)$ that allows for efficient samping and tractable likelihood evaluation, e.g., Gaussian
- Sampling requires efficient evaluation of

$$z_k = f_k(z_{k-1}), \quad k = 1, \ldots, K$$

- Likelihood computation also requires the evaluation of determinants of $n \times n$ Jacobian matrices $\sim \mathcal{O}(n^3)$, prohibitively expensive within a learning loop!
- Design transformations so that the resulting Jacobian matrix has special structure. For example
  - lower rank update to identity as in planar flows.
  - triangular matrix whose determinant is just the product of the diagonal entries, i.e., an $\mathcal{O}(n)$ operation.

Designing Invertible Transformations

- NICE or Nonlinear Independent Components Estimation (Dinh et al., 2014) composes two kinds of invertible transformations: additive coupling layers and rescaling layers
- Real-NVP (Dinh et al., 2017)
- Inverse Autoregressive Flow (Kingma et al., 2016)
- Masked Autoregressive Flow (Papamakarios et al., 2017)

NICE: Additive Coupling Layers

- Partition the variable z into two disjoint subsets

$$z = z_{1:d} \cup z_{d+1:n}$$

- Forward mapping $z \mapsto x$:

$$x_{1:d} = z_{1:d}, \quad x_{d+1:n} = z_{d+1:n} + m_\theta(z_{1:d})$$

  where $m_\theta : \mathbb{R}^d \mapsto \mathbb{R}^{n-d}$ is a neural network with parameters θ

- Backward mapping $x \mapsto z$:

$$z_{1:d} = x_{1:d}, \quad z_{d+1:n} = x_{d+1:n} - m_\theta(x_{1:d})$$

- Forward/Backward mapping is volume preserving: the determinant of the Jacobian is 1.

NICE: Rescaling Layers

- Additive coupling layers are composed together (with arbitrary partitions of variables in each layer)
- Final layer of NICE uses a rescaling transformation
- Forward mapping $z \mapsto x$:

$$x_i = s_i z_i, \quad i = 1, \ldots, n$$

  where $s_i > 0$ is the scaling factor for the i-th dimension.
- Backward mapping $x \mapsto z$:

$$z_i = \frac{x_i}{s_i}, \quad i = 1, \ldots, n$$

- Jacobian of forward mapping:

$$J = \mathrm{diag}(s), \quad \det(J) = \prod_{i=1}^{n} s_i.$$

RealNVP: Non-volume Preserving NICE

- Forward mapping $z \mapsto x$:

$$x_{1:d} = z_{1:d}, \quad x_{d+1:n} = z_{d+1:n} \odot \exp(\alpha_\theta(z_{1:d})) + \mu_\theta(z_{1:d})$$

  where α_θ and μ_θ are both neural networks.

- Backward mapping $x \mapsto z$:

$$z_{1:d} = x_{1:d}, \quad z_{d+1:n} = \exp(-\alpha_\theta(x_{1:d})) \odot (x_{d+1:n} - \mu_\theta(x_{1:d}))$$

- The determinant of the Jacobian of forward mapping

$$\det\left(\frac{\partial x}{\partial z}\right) = \exp\left(\sum \alpha_\theta(z_{1:d})\right)$$

- Non-volume preserving transformation in general since determinant can be less than or greater than 1.

Autoregressive Models as Normalizing Flows

- Consider a Gaussian autoregressive model

$$p(x) = \prod_{i=1}^{n} p(x_i|x_{<i})$$

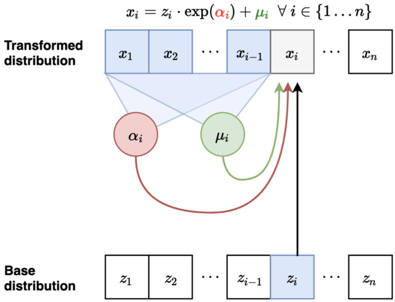
where $p(x_i|x_{<i}) = \mathcal{N}(\mu_i(x_{1:i-1}), \exp(\alpha_i(x_{1:i-1}))^2)$. μ_i and α_i are neural networks for $i > 1$ and constants for $i = 1$.

- Sequential sampling:

$$z_i \sim \mathcal{N}(0,1), \quad x_i = \exp(\alpha_i(x_{1:i-1}))z_i + \mu_i(x_{1:i-1}), \quad i = 1, \ldots, n$$

- Flow interpretation: transforms samples from the standard Gaussian to those generated from the model via invertible transformations (parameterized by μ_i, α_i)

Masked Autoregressive Flow (MAF)

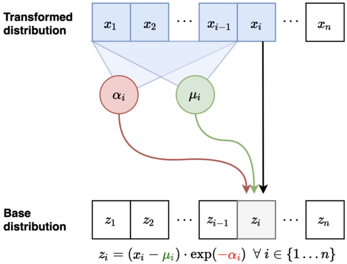


- Forward mapping from $z \mapsto x$:

$$x_i = \exp(\alpha_i(x_{1:i-1}))z_i + \mu_i(x_{1:i-1}), \quad i = 1, \dots, n$$

- Like autoregressive models, sampling is sequential and slow ($\mathcal{O}(n)$)

Masked Autoregressive Flow (MAF)



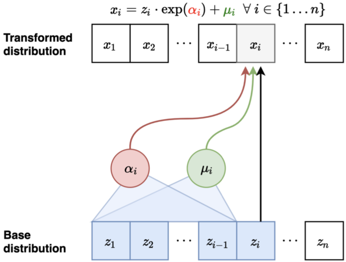
- Inverse mapping from $x \mapsto z$: shift and scale

$$z_i = (x_i - \mu_i(x_{1:i-1})) / \exp(\alpha_i(x_{1:i-1})), \quad i = 1, \ldots, n$$

  Note that this can be done in parallel.

- Jacobian is lower diagonal, hence determinant can be computed efficiently.
- Likelihood evaluation is easy and parallelizable.

Inverse Autoregressive Flow (IAF)



- Forward mapping from $z \mapsto x$ (parallel):

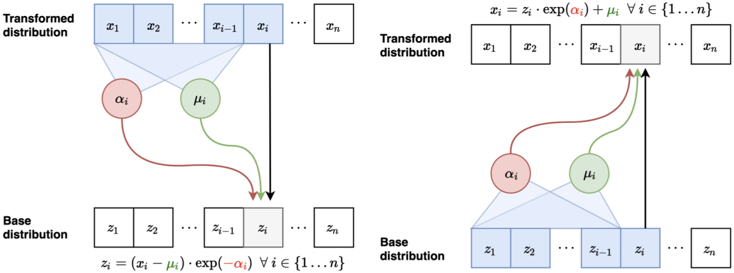
$$x_i = \exp(\alpha_i(z_{1:i-1}))z_i + \mu_i(z_{1:i-1}), \quad i = 1, \ldots, n$$

- Backward mapping from $x \mapsto z$ (sequential):

$$z_i = (x_i - \mu_i(z_{1:i-1}))/\exp(\alpha_i(z_{1:i-1}))$$

- Fast to sample from, slow to evaluate likelihoods of data points. However, likelihood evaluation for a sampled point is fast.

IAF is Inverse of MAF


Transformed distribution
x_1 x_2 $\cdots$ x_{i-1} x_i $\cdots$ x_n
α_i
μ_i
Base distribution
z_1 z_2 $\cdots$ z_{i-1} z_i $\cdots$ z_n
$z_i = (x_i - \mu_i) \cdot \exp(-\alpha_i) \;\; \forall\, i \in \{1 \ldots n\}$
$x_i = z_i \cdot \exp(\alpha_i) + \mu_i \;\; \forall\, i \in \{1 \ldots n\}$
Transformed distribution
x_1 x_2 $\cdots$ x_{i-1} x_i $\cdots$ x_n
α_i
μ_i
Base distribution
z_1 z_2 $\cdots$ z_{i-1} z_i $\cdots$ z_n

Inverse pass of MAF (left) vs. Forward pass of IAF (right)

- Interchanging z and x in the inverse transformation of MAF gives the forward transformation of IAF.
- Similarly, forward transformation of MAF is inverse transformation of IAF.

Summary of Nomalizing Flows

- Transform simple distributions into more complex distributions via change of variables
- Jacobian of transformations should have tractable determinant for efficient learning and density estimation
- Computational tradeoff in evaluating forward and inverse transformations
  - **MAF**: Fast likelihood evaluation, slow sampling, more suited for MLE based training, density estimation.
  - **IAF**: Fast sampling, slow likelihood evaluation, more suited for variational inference, real time generation.
  - **NICE** and **RealNVP**: Fast on both side, but generally less flexible than the others.

MCMC Recap

- MCMC approximates the posterior through a sequence of transitions

  $$z_0 \sim q(z_0), \quad z_t \sim q(z_t|z_{t-1}, x), \quad t = 1, 2, \ldots$$

  where the transition kernel satisfies the detailed balance condition

  $$p(x, z_{t-1})q(z_t|z_{t-1}, x) = p(x, z_t)q(z_{t-1}|z_t, x)$$

- Pros
  - automatically adapts to true posterior
  - asymptotically unbiased
- Cons
  - slow convergence, hard to assess quality
  - tuning headaches

MCMC as Flows

- Each iteration in MCMC can be viewed as a mapping $z_{t-1} \mapsto z_t$, and the marginal likelihood of z_T is

$$q(z_T|x) = \int q(z_0|x) \prod_{t=1}^{T} q(z_t|z_{t-1}, x)\; dz_0, \ldots, dz_{T-1}$$

- Variational lower bound

$$L = \mathbb{E}_{q(z_T|x)} \log \frac{p(x, z_T)}{q(z_T|x)} \leq \log p(x)$$

- The stochastic Markov chain, therefore, can be viewed as a nonparametric variational approximation.
- Can we combine MCMC and VI to get the best of both worlds?

Auxiliary Variational Lower Bound

- Use auxiliary random variables $y = (z_0, \ldots, z_{T-1})$ to construct a tractable lower bound

$$L_{\text{aux}} = \mathbb{E}_{q(y,z_T|x)} \log \frac{p(x,z_T)r(y|z_T,x)}{q(y,z_T|x)} \leq \log p(x)$$

  - $r(y|z_T,x)$ is an arbitrary auxiliary distribution, e.g.

$$r(y|z_T,x) = \prod_{t=1}^{T} r_t(z_{t-1}|z_t,x)$$

- This is a looser lower bound

$$\begin{aligned} L_{\text{aux}} &= \mathbb{E}_{q(y,z_T|x)} \left(\log p(x,z_T) + \log r(y|z_T,x) - \log q(y,z_T|x) \right) \\ &= L - \mathbb{E}_{q(z_T|x)} \left(D_{KL}(q(y|z_T,x) \| r(y|z_T,x)) \right) \\ &\leq L \leq \log p(x) \end{aligned}$$

Monte Carlo Estimate of MCMC Lower Bound

- Suppose $z_0, z_1, \ldots, z_T$ is a sampled trajectory

$$\begin{aligned} z_0 &\sim q(z_0|x) \\ z_t &\sim q_t(z_t|z_{t-1}, x), \quad t = 1, \ldots, T \end{aligned}$$

- Unbiased stochastic estimate of L_{aux}

$$\begin{aligned} \hat{L}_{\text{aux}} &= \log p(x, z_T) - \log q(z_0|x) + \sum_{t=1}^{T} \left(\log \frac{r_t(z_{t-1}|z_t, x)}{q_t(z_t|z_{t-1}, x)} \right) \\ &= \log p(x, z_0) - \log q(z_0|x) + \sum_{t=1}^{T} \log \alpha_t \end{aligned}$$

where

$$\alpha_t = \frac{p(x, z_t) r_t(z_{t-1}|z_t, x)}{p(x, z_{t-1}) q_t(z_t|z_{t-1}, x)}$$

MCMC Always Improves The ELBO

- Using the detailed balance condition

$$\alpha_t = \frac{p(x, z_t) r_t(z_{t-1}|z_t, x)}{p(x, z_{t-1}) q_t(z_t|z_{t-1}, x)} = \frac{r_t(z_{t-1}|z_t, x)}{q_t(z_{t-1}|z_t, x)}$$

- Therefore,

$$L_{\text{aux}} = \mathbb{E}_{q(z_0|x)} \log \frac{p(x, z_0)}{q(z_0|x)} + \sum_{t=1}^{T} \mathbb{E}_{q(y, z_T|x)} \log \frac{r_t(z_{t-1}|z_t, x)}{q_t(z_{t-1}|z_t, x)}$$

- For optimal $r_t(z_{t-1}|z_t, x) = q(z_{t-1}|z_t, x)$

$$\mathbb{E}_q \log \frac{r_t(z_{t-1}|z_t, x)}{q_t(z_{t-1}|z_t, x)} = \mathbb{E}_q \log \frac{q(z_{t-1}|z_t, x)}{q_t(z_{t-1}|z_t, x)} \geq 0$$

- MCMC iterations always improve approximation unless already perfect! In practice, we need

$$r_t(z_{t-1}|z_t, x) \approx q(z_{t-1}|z_t, x)$$

Optimizing The Markov Chain

- Specify a parameterized Markov chain

$$q_\theta(z) = q_\theta(z_0|x) \prod_{t=1}^{T} q_\theta(z_t|z_{t-1}, x)$$

- Specify a parameterized auxiliary distribution $r_\theta(y|z_T, x)$
- Sample MCMC trajectories for the variational lower bound

$$\hat{L}(\theta) = \log p(x, z_T) - \log q(z_0|x) + \sum_{t=1}^{T} \left(\log \frac{r_t(z_{t-1}|z_t, x)}{q_t(z_t|z_{t-1}, x)} \right)$$

- Run SGD using $\nabla_\theta \hat{L}(\theta)$ (reparameterization trick)

Example: Bivariate Gaussian

- A bivariate Gaussian target distribution

$$p(z^1, z^2) \propto \exp\left(-\frac{1}{2\tau_1^2}(z^1 - z^2)^2 - \frac{1}{2\tau_2^2}(z^1 + z^2)^2\right)$$

- **Gibbs sampling**

$$q(z_t^i | z_{t-1}) = p(z^i | z^{-i}) = \mathcal{N}(\mu_i, \sigma_i^2)$$

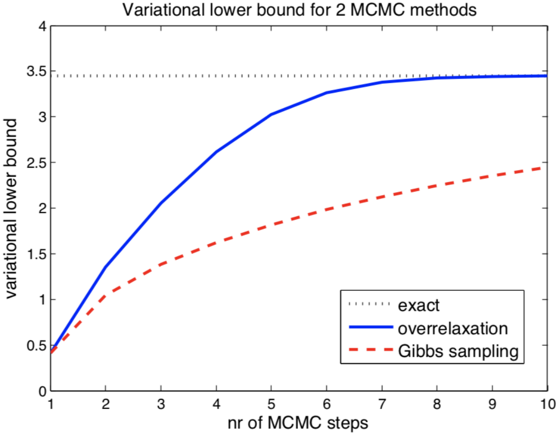
- **Over-relaxation** (Adler, 1981)

$$q(z_t^i | z_{t-1}) = \mathcal{N}(\mu_i + \alpha(z_{t-1}^i - \mu_i), \sigma_i^2(1 - \alpha^2))$$

- Gaussian reverse model $r_t(z_{t-1}|z_t)$, linear dependence on z_t. Find the best α via variational lower bound maximization.

Example: Bivariate Gaussian

Gibbs sampling versus over-relaxation for a bivariate Gaussian



The improved mixing of over-relaxation results in an improved variational lower bound.

Hamiltonian Variational Inference

- We can use Hamiltonian dynamics for more efficient transition distributions

$$v_t' \sim q(v_t'|z_{t-1}, x), \quad (v_t, z_t) = \Phi(v_t', z_{t-1})$$

  where $\Phi : \mathbb{R}^{2n} \mapsto \mathbb{R}^{2n}$ is the Hamiltonian flow.

- Φ is deterministic, invertible and volume preserving

$$q(v_t, z_t|z_{t-1}, x) = q(v_t'|z_{t-1}, x), \quad r(v_t', z_{t-1}|z_t, x) = r(v_t|z_t, x)$$

- Note that we would use *leapfrog* integrator to discretize the Hamiltonian flow. However, the resulting map $\hat{\Phi}$ is also invertible and volume preserving, and the above equations still hold.

Hamiltonian Variational Inference

- HMC trajectory

$$z_0 \sim q(z_0|x)$$
$$v'_t \sim q_t(v'_t|z_{t-1}, x), \quad v_t, z_t = \hat{\Phi}(v'_t, z_{t-1}), \quad t = 1, \ldots, T$$

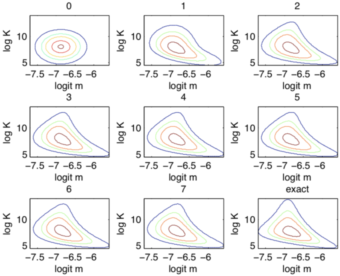
- Lower bound estimate

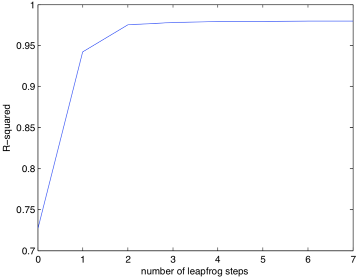
$$\hat{L}(\theta) = \log p(x, z_0) - \log q(z_0|x) + \sum_{t=1}^{T} \log \frac{p(x, z_t) r_t(v_t|z_t, x)}{p(x, z_{t-1}) q_t(v'_t|x, z_{t-1})}$$

- Stochastic optimization using $\nabla_\theta \hat{L}(\theta)$
  - No rejection step, to keep everything differentiable.
  - θ includes all parameters in q and r, and may include some HMC hyperparameters (stepsize and mass matrix) as well.
  - Differentiate through the leapfrog integrator.

Examples: Overdispersed Counts

A simple 2-dimensional beta-binomial model for overdispersion. One step of Hamiltonian dynamics with varying number of leapfrog steps.





Examples: Generative Model for MNIST

Variational autoencoder for binarized MNIST, Gaussian prior $p(z) = \mathcal{N}(0, I)$, MLP conditional likelihood $p_\theta(x|z)$

Model	$-L$	$-\log p(x)$
Results with $q(z_0|x) = \mathcal{N}(\mu, \sigma^2 \mathbf{I})$*:*		
5 leapfrog steps	90.86	87.16
10 leapfrog steps	87.60	85.56
With $q(z_0|x) =$ *inference network:*		
No leapfrog steps	94.18	88.95
1 leapfrog step	91.70	88.08
4 leapfrog steps	89.82	86.40
8 leapfrog steps	88.30	85.51

- MCMC makes bound tighter, give better marginal likelihood.
- MCMC also works with simple initialization.

Examples: Generative Model for MNIST

Variational autoencoder for binarized MNIST, Gaussian prior $p(z) = \mathcal{N}(0, I)$, MLP conditional likelihood $p_\theta(x|z)$

Model	$-L$	$-\log p(x)$
Results with $q(z_0|x) = \mathcal{N}(\mu, \sigma^2\mathbf{I})$*:*		
5 leapfrog steps	90.86	87.16
10 leapfrog steps	87.60	85.56
With $q(z_0|x) =$ *inference network:*		
No leapfrog steps	$\boxed{94.18}$	$\boxed{88.95}$
1 leapfrog step	91.70	88.08
4 leapfrog steps	89.82	86.40
8 leapfrog steps	$\boxed{88.30}$	$\boxed{85.51}$

- MCMC makes bound tighter, give better marginal likelihood.
- MCMC also works with simple initialization.

Examples: Generative Model for MNIST

Variational autoencoder for binarized MNIST, Gaussian prior $p(z) = \mathcal{N}(0, I)$, MLP conditional likelihood $p_\theta(x|z)$

Model	$-L$	$-\log p(x)$
Results with $q(z_0\|x) = \mathcal{N}(\mu, \sigma^2\mathbf{I})$*:*		
5 leapfrog steps	90.86	87.16
10 leapfrog steps	87.60	85.56
With $q(z_0\|x) =$ *inference network:*		
No leapfrog steps	94.18	88.95
1 leapfrog step	91.70	88.08
4 leapfrog steps	89.82	86.40
8 leapfrog steps	88.30	85.51

- MCMC makes bound tighter, give better marginal likelihood.
- MCMC also works with simple initialization.

Combining MCMC and VI

- MCMC improves variational approximation
  - MCMC kernels automatically adapt to target $p(z|x)$.
  - More flexible approximations in addition to standard exponential family distributions.
  - More MCMC steps ⇒ slower iterations, but few iterations needed for convergence.
- Optimizing variational bound improves MCMC
  - Automatic tuning, convergence assessment, independent sampling, no rejections.
  - Learning MCMC transitions $q_t(z_t|z_{t-1}, x)$.
  - Optimize initialization $q(z_0|x)$.
- Many possibilities left to explore.

Particle Based Variational Inference

- So far, most of the approximating distributions used in VI take a parametric form, that is $q_\theta(x)$ with parameter θ.
- This parametric form often limits the power of the approximating distributions.
- In what follows, we will introduce a particle based VI introduced by Liu et al. that uses non-parameteric approximating distributions.

Stein's Method

- A general theoretical tool for bounding differences between distributions, introduced by Charles Stein.
- The key idea is to characterize a distribution p with a Stein operator $\mathcal{A}_p$, such that

$$p = q \quad \Longleftrightarrow \quad \mathbb{E}_{x \sim q}[\mathcal{A}_p f(x)] = 0, \quad \forall f \in \mathcal{F}$$

For continuous distributions with smooth density $p(x)$,

$$\mathcal{A}_p f(x) := s_p(x)^T f(x) + \nabla_x \cdot f(x)$$

where $s_p(x) = \nabla_x \log p(x)$ is the score function.

Note that $s_p(x)$ does not dependent on the normalizing constant of $p(x)$, so $p(x)$ can be unnormalized.

Stein's Method

- When $p = q$, we have Stein's Identity

$$\mathbb{E}_{x\sim p}\left[s_p(x)^T f(x) + \nabla_x \cdot f(x)\right] = 0$$

- Stein's identity defines an infinite number of identities indexed by test function f, widely applied in learning probabilistic models, variance reduction, optimization and many more.
- When $p \neq q$, we have (also by Stein's Identity)

$$\mathbb{E}_{x\sim q}[\mathcal{A}_p f(x)] = \mathbb{E}_{x\sim q}[(s_p(x) - s_q(x))^T f(x)] \qquad (1)$$

  Easy to find test function $f(x)$ such that (1) is non-zero. For example:

$$f(x) = s_p(x) - s_q(x)$$

Stein Discrepancy

- We therefore, define Stein Discrepancy between p and q as follows
$$D(q\|p) := \max_{f \in \mathcal{F}} \mathbb{E}_{x \sim q}[\mathcal{A}_p f(x)] \tag{2}$$
where $\mathcal{F}$ is a rich enough set of functions.
- Traditionally, Stein's method takes $\mathcal{F}$ to be sets of functions with bounded Lipschitz norm, which is computationally difficult for practical use.
- We can use a kernel trick to construct a reproducing kernel Hilbert space (RKHS) where there is a closed form solution to (2).

Reproducing Kernel Hilbert Space

- Let $k(x, x')$ be a positive definite kernel, that is
$$\int_{\mathcal{X}} g(x)k(x,x')g(x')\,dxdx' > 0, \quad \forall\ 0 < \|g\|_2^2 < \infty.$$
By Mercer's theorem,
$$k(x, x') = \sum\nolimits_i \lambda_i e_i(x) e_i(x')$$

- We can define a RKHS $\mathcal{H}$ that contains linear combinations of these eigenfunctions
$$f(x) = \sum\nolimits_i f_i e_i(x), \quad \langle f, g\rangle_{\mathcal{H}} = \sum\nolimits_i \frac{f_i g_i}{\lambda_i}$$
with $\|f\|_{\mathcal{H}}^2 = \langle f, f\rangle_{\mathcal{H}} = \sum_i f_i^2/\lambda_i$.

- Reproducing Property
$$f(x) = \langle f, k(\cdot, x)\rangle_{\mathcal{H}}, \quad k(x, x') = \langle k(\cdot, x), k(\cdot, x')\rangle_{\mathcal{H}}.$$

Kernelized Stein Discrepancy

- Given a positive definite kernel $k(x, x')$, Liu et al. define a **kernelized Stein discrepancy** (KSD) $D(q\|p)$ as follows

$$D(q\|p) = \sqrt{\mathbb{E}_{x,x'\sim q}[\delta_{p,q}(x)^T k(x,x')\delta_{p,q}(x')]}$$

where $\delta_{p,q}(x) = s_p(x) - s_q(x)$. Obviously,

$$D(q\|p) \geq 0, \quad D(q\|p) = 0 \Leftrightarrow q = p.$$

- With the spectral decomposition, we can rewrite KSD as

$$D(q\|p) = \sqrt{\sum_i \lambda_i \|\mathbb{E}_{x\sim q}[\mathcal{A}_p e_i(x)]\|_2^2}$$

Kernelized Stein Discrepancy

- It turns out that KSD can be viewed as standard Stein discrepancy over a specific family of functions $\mathcal{F}$, i.e, the unit ball of $\mathcal{H}^d = \mathcal{H} \times \cdots \times \mathcal{H}$.
- Denote $\beta(x') = \mathbb{E}_{x\sim q}[\mathcal{A}_p k_{x'}(x)]$, then

$$D(q\|p) = \|\beta\|_{\mathcal{H}^d}$$

- Moreover, we have

$$\langle \beta, f \rangle_{\mathcal{H}^d} = \mathbb{E}_{x\sim q}[\mathcal{A}_p f(x)], \quad \forall f \in \mathcal{H}^d$$

- Therefore,

$$D(q\|p) = \max_{f\in\mathcal{F}} \mathbb{E}_{x\sim q}[\mathcal{A}_p f(x)]$$

where $\mathcal{F} = \{f \in \mathcal{H}^d : \|f\|_{\mathcal{H}^d} \leq 1\}$. The maximum is achieved at $f^* = \beta/\|\beta\|_{\mathcal{H}^d}$.

Stein Variational Gradient Descent

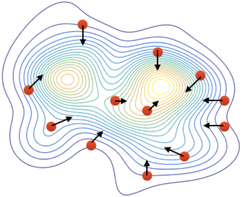
Proposed by Liu and Wang, 2016.

Idea: represent the distribution using a collection of particles $\{x_i\}_{i=1}^n$ and iteratively move these particles toward the target p by updates of form

$$x_i \leftarrow T(x_i), \quad T(x) = x + \epsilon\phi(x)$$

where ϕ is a perturbation direction chosen to maximumly decrease the KL divergence.

$$\phi = \arg\max_{\phi\in\mathcal{F}} \left\{ -\frac{\partial}{\partial\epsilon} D_{\mathrm{KL}}(q_T \| p) \Big|_{\epsilon=0} \right\}$$



where q_T is the density of $x' = T(x)$ when the current density of x is $q(x)$.

Stein Variational Gradient Descent

- Perturbation direction is closely related to Stein operator

$$-\frac{\partial}{\partial \epsilon} D_{\mathrm{KL}}(q_T \| p)\Big|_{\epsilon=0} = \mathbb{E}_{x\sim q}[\mathcal{A}_p \phi(x)]$$

- This gives another interpretation of Stein discrepancy

$$D(q\|p) = \max_{\phi\in\mathcal{F}} \left\{ -\frac{\partial}{\partial \epsilon} D_{\mathrm{KL}}(q_T \| p)\Big|_{\epsilon=0} \right\}$$

- Most importantly, the optimum direction has a closed form when $\mathcal{F}$ is the unit ball of RKHS $\mathcal{H}^d$:

$$\begin{aligned} \phi^*(\cdot) &= \mathbb{E}_{x\sim q}[\mathcal{A}_p k(x,\cdot)] \\ &= \mathbb{E}_{x\sim q}[\nabla_x \log p(x) k(x,\cdot) + \nabla_x k(x,\cdot)] \end{aligned}$$

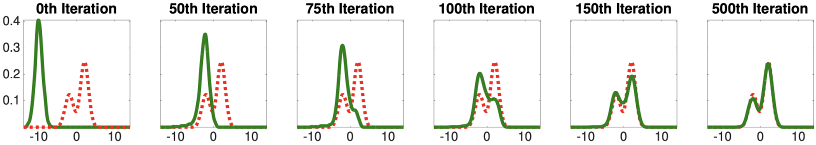
Stein Variational Gradient Descent

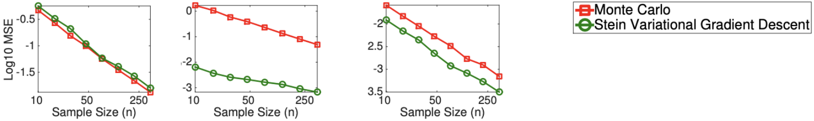
We can approximate the expectation $E_{x\sim q}$ with the empirical average over current particles

$$x_i \leftarrow x_i + \epsilon \frac{1}{n} \sum_{j=1}^{n} \left[\nabla_x \log p(x_j) k(x_j, x_i) + \nabla_{x_j} k(x_j, x_i) \right], \; 1 \leq i \leq n$$

- Deterministically transport probability mass from initial q_0 to target p.
- Reduces to standard gradient ascent for MAP when using a single particle ($n = 1$).
- $\nabla_x \log p(x_j)$: the gradient term moves the particles towards high probability domains of $p(x)$.
- $\nabla_x k(x_j, x_i)$: the repulsive force term enforces diversity in the particles and prevents them from collapsing to the modes of $p(x)$.

Examples: Mixture of Gaussian

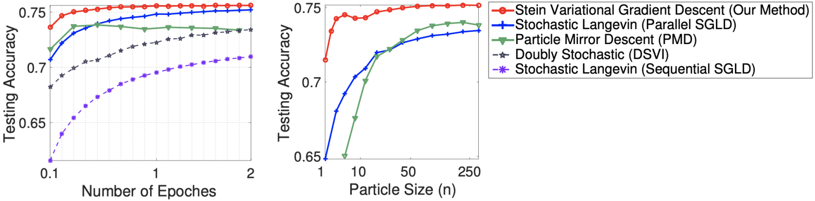




(a) Estimating $\mathbb{E}(x)$ *(b) Estimating* $\mathbb{E}(x^2)$ *(c) Estimating* $\mathbb{E}(\cos(\omega x + b))$

Liu et al., 2016

Examples: Bayesian Logistic Regression



(a) Particle size $n = 100$

(b) Results at 3000 iteration ($\approx$ 0.32 epoches)

Liu et al., 2016

References

- D. J. Rezende and S. Mohamed. Variational inference with normalizing flows. Proceedings of the 32nd International Conference on Machine Learning, pages 1530–1538, 2015.
- L. Dinh, D. Krueger, and Y. Bengio. NICE: Non-linear Independent Components Estimation. arXiv:1410.8516, 2014.
- L. Dinh, J. Sohl-Dickstein, and S. Bengio. Density estimation using Real NVP. Proceedings of the 5th International Conference on Learning Representations, 2017.
- Stephen L Adler. Over-relaxation method for the monte carlo evaluation of the partition function for multiquadratic actions. Physical Review D, 23(12):2901, 1981.

References

- D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling. Improved variational inference with Inverse Autoregressive Flow. Advances in Neural Information Processing Systems 29, pages 4743–4751, 2016.
- Papamakarios, G., Murray, I., and Pavlakou, T. (2017). Masked autoregressive flow for density estimation. In Advances in Neural Information Processing Systems, pages 2335–2344.
- T. Salimans,D. P. Kingma,and M. Welling. Markov chain monte carlo and variational inference: Bridging the gap. In ICML, 2015.

References

- Q. Liu, J. D. Lee, and M. I. Jordan. A kernelized Stein discrepancy for goodness-of-fit tests and model evaluation. In ICML, 2016.
- Q. Liu and D. Wang. Stein variational gradient descent: A general purpose bayesian inference algorithm. In Advances in Neural Information Processing Systems 29, pp. 2370–2378, 2016