There and Back Again: A Tale of Expectations and Swamps of Monte Carlo integration numerical

Neurips 2020 Tutorial by Marc Deisenroth and Heights

Monte Carlo

Swamps of

b

A
There and Back Again: A Tale of Expectations and Slopes. A Neurips 2020 Tutorial by Marc Deisenroth and Heights Monte Carlo Normalization Flow integration numerical Swamps of ABCA
Change of Variables and Normalizing Flows

Cheng Soon Ong
Marc Peter Deisenroth

December 2020
Normalizing flows for density estimation

Key idea
(Tabak & Turner, 2013; Rippel & Adams, 2013; Rezende & Mohamed, 2015)

Build complex distributions from simple distributions via a flow of successive (invertible) transformations

Figure: Generated with PyMC3 (Salvatier et al., 2016)
Normalizing flows for density estimation

Key idea
(Tabak & Turner, 2013; Rippel & Adams, 2013; Rezende & Mohamed, 2015)
Build complex distributions from simple distributions via a flow of successive (invertible) transformations

Key ingredient: \textit{Change-of-variables trick}
Change of Variables
Change of variables: Key idea

Key idea

Transform random variable $X$ into random variable $Z$ using an invertible transformation $\phi$, while keeping track of the change in distribution.
Change of variables: Key idea

Transform random variable $X$ into random variable $Z$ using an invertible transformation $\phi$, while keeping track of the change in distribution.

- Distribution $p_X$ induces distribution $p_Z$ via transformation $\phi$.
- Distribution $p_Z$ induces distribution $p_X$ via transformation $\phi^{-1}$. 
Determinant of Jacobian

\[
\left| \det \left( \frac{dz}{dx} \right) \right| = \left| \det \left( \frac{d\phi(x)}{dx} \right) \right|
\]

tells us how much the domain \( dx \) is stretched to \( dz \)
How it works

▶ Constraint: volume preservation

\[
\int_{\mathcal{X}} p_X(x)dx = 1 = \int_{\mathcal{Z}} p_Z(z)dz
\]
How it works

- Constraint: volume preservation

\[ \int_\mathcal{X} p_X(x) \, dx = 1 = \int_\mathcal{Z} p_Z(z) \, dz \]

- Volume preservation: Rescale \( p_Z \) by the inverse of Jacobian determinant

\[ p_Z(z) = p_X(x) \left| \det \left( \frac{d\phi(x)}{dx} \right) \right|^{-1} \]
Considerations

Express target distribution $p_Z$ in terms of known distribution $p_X$ and the Jacobian determinant of an invertible mapping $\phi$

- No need to invert $\phi$ explicitly
- Generate expressive distributions $p_Z$ by simple $p_X$ and flexible transformation $\phi$
Applications

- Numerical integration (turn indefinite integrals into definite ones)
- Neural ODEs (E 2017, Chen et al., 2018)
- Learning in implicit generative models (e.g., GANs) and likelihood-free inference (e.g., ABC)
  (e.g., Mohamed & Lakshminarayanan, 2016; Sisson et al., 2007)
- **Normalizing flows** (Rezende & Mohamed, 2015)
Normalizing Flows
Normalizing flows for density estimation

**Key idea**

(Tabak & Turner, 2013; Rippel & Adams, 2013; Rezende & Mohamed, 2015)

Build complex distributions from simple distributions via a flow of successive (invertible) transformations

*Figure: Generated with PyMC3 (Salvatier et al., 2016)*
How it works

Random variable $z_0 \sim p_0$

Simple base distribution $p_0$, e.g. $p_0 = \mathcal{N}(0, I)$
How it works

- Random variable $z_0 \sim p_0$
- Simple base distribution $p_0$, e.g. $p_0 = \mathcal{N}(0, I)$
- Successive transformation of $z_k$ via invertible transformations $f_k$:
  \[ z_k = f_k(z_{k-1}) \]
- Observed data $x = z_K$ at the end of the chain
  \[ x = z_K = f_K \circ f_{K-1} \circ \cdots \circ f_1(z_0) \]
Repeated application of change-of-variables trick

\[
p(x) = p(z_K) = p(z_0) \prod_{k=1}^{K} \left| \det \frac{df_k(z_{k-1})}{dz_{k-1}} \right|^{-1}
\]

Entropy

\[
\log p(x) = \log p(z_K) = \log p(z_0) - \sum_{k=1}^{K} \log \left| \det \left( \frac{df_k(z_{k-1})}{dz_{k-1}} \right) \right|
\]
Repeated application of a planar flow $z_k = f_k(z_{k-1}) = z_{k-1} + u\sigma(w^T z_{k-1} + b)$
Repeated application of a planar flow \( z_k = f_k(z_{k-1}) = z_{k-1} + u\sigma(w^Tz_{k-1} + b) \)
Repeated application of a planar flow

\[ z_k = f_k(z_{k-1}) = z_{k-1} + u \sigma(w^T z_{k-1} + b) \]
Repeated application of a planar flow $z_k = f_k(z_{k-1}) = z_{k-1} + u \sigma(w^\top z_{k-1} + b)$
Illustration with PyMC3 (Salvatier et al., 2016)

Figure: Generated using a PyMC3 tutorial (Salvatier et al., 2016)

Repeated application of a planar flow $z_k = f_k(z_{k-1}) = z_{k-1} + u\sigma(w^\top z_{k-1} + b)$
Computing expectations

\[ E_{p_X}[l(x)] = E_{p_K}[l(z_K)] = E_{p_0}[l(f_K \circ \cdots \circ f_1(z_0))] \]
Computing expectations

\[ \mathbb{E}_{p_X}[l(x)] = \mathbb{E}_{p_K}[l(z_K)] = \mathbb{E}_{p_0}[l(f_K \circ \cdots \circ f_1(z_0))] \]

- Expectations w.r.t. \( p_K \) can be computed without explicitly knowing \( p_K \) or \( p_X \)
  - Sample \( z_0^{(s)} \sim p_0 \)
  - Push sample forward through sequence of deterministic transformations
    - Valid sample \( x^{(s)} \sim p_X(x) \)
- Monte-Carlo estimation to get expected value
Computational considerations

- Compute log-determinant of Jacobian
- Cheap (linear) if Jacobian is (block-)diagonal or triangular
Computational considerations

- Compute log-determinant of Jacobian
- Cheap (linear) if Jacobian is (block-)diagonal or triangular
- Require partial derivatives

\[
\frac{\partial z_k^{(d)}}{\partial z_{k-1}^{(d)}} = 0 \quad \Rightarrow \quad \frac{dz_k}{dz_{k-1}} = \begin{bmatrix}
\frac{\partial z_k^{(1)}}{\partial z_{k-1}^{(1)}} & 0 & \cdots & 0 \\
\vdots & \ddots & \ddots & \vdots \\
\frac{\partial z_k^{(D)}}{\partial z_{k-1}^{(1)}} & \cdots & \cdots & \frac{\partial z_k^{(D)}}{\partial z_{k-1}^{(D)}}
\end{bmatrix} \in \mathbb{R}^{D \times D}.
\]
Autoregressive flows

- High-level idea:

\[ z_k^{(d)} = \phi(z_{k-1}^{(\leq d)}) \]
Autoregressive flows

High-level idea:

\[ z^{(d)}_k = \phi(z^{(\leq d)}_{k-1}) \]

- NICE (Dinh et al., 2014)
- Inverse autoregressive flow (Kingma et al., 2016)
- Real NVP (Dinh et al., 2017)
- Masked autoregressive flow (Papamakarios et al., 2017)
- Glow (Kingma & Dhariwal, 2018)
- (Block) neural autoregressive flows, spline flows, ... (e.g., Huang et al., 2018; de Cao et al., 2019; Durkan et al., 2019)
Application areas

▶ Variational inference in deep generative models (e.g., Rezende & Mohamed, 2015)
▶ Graph neural networks (Liu et al., 2019)
▶ Parallel WaveNet (van den Oord et al., 2018)
Application areas

▶ Variational inference in deep generative models (e.g., Rezende & Mohamed, 2015)
▶ Graph neural networks (Liu et al., 2019)
▶ Parallel WaveNet (van den Oord et al., 2018)
▶ Continuous flows
  ▶ Neural ODEs (e.g, E, 2017; Chen et al., 2018)
  ▶ Flows on manifolds (e.g., Gemici et al., 2016; Rezende et al., 2020; Mathieu & Nickel, 2020)
Normalizing flows provide a constructive way to generate rich distributions

Key idea: Transform a simple distribution using a flow of successive (invertible) transformations

Key ingredient: Change-of-variables trick

Jacobians can be computed efficiently, if the transformations are defined appropriately

Can be used as a generator and inference mechanism


