Overview
Change of Variable
Normalizing Flows
Building Flow Models

COMPSCI 688: Probabilistic Graphical Models
Last time: GPs
- GPM
- GPM
Next time:
- Flows?
- PPL?

HW 5 due Friday

Overview
Change of Variable
Normalizing Flows
Building Flow Models

Motivation: Transforming a Simple Distribution

Suppose we want to learn a model $p_\theta(x)$ for a complex $x$ (like images). What properties do we want from $p_\theta(x)$?

- Easy to sample (useful for generation)
- Easy to evaluate density (useful for learning)

Many simple distributions satisfy these properties (e.g., Gaussian, uniform).

But data distributions are complex! E.g. multi-modal.

**Key idea behind flow models:** map simple distributions to complex ones through deterministic invertible transformations

Motivation: Transforming a Simple Distribution (Learning)

Consider our VAE model $p_\theta(x)$ but with no noise

\[
\begin{align*}
\mathbf{z} & \sim p(z) \text{ simple, e.g. } \mathcal{N}(0, I) \\
x & = f_\theta(z)
\end{align*}
\]

Could we learn $p_\theta(x)$ “directly” by MLE?

- Can easily generate samples $x \sim p_\theta(x)$
- To learn, need to compute the density $p_\theta(x)$ under transformation $f_\theta$. Can we do it?

Demo
Motivation: Transforming a Simple Distribution (Inference)

Also useful in variational inference, e.g. \( q_\theta(z) \) in VAEs

**Goal**: \( q_\theta(z) \approx p(z|x) \) where \( p, x \) are given. We used reparameterized Gaussians:

\[
\epsilon \sim \mathcal{N}(0, I) \\
z = T_\theta(\epsilon) = L \epsilon + \mu \implies z \sim q_\theta(z) = \mathcal{N}(z; \mu, LL^T)
\]

What if we used complex \( T_\theta(\epsilon) \) (e.g. neural net) instead?

- Would have a rich class of variational distributions.
- Could easily sample from \( q_\theta(z) \)
- For ELBO, need to compute density \( q_\theta(z) \) under transformation \( T_\theta \). Can we do it?

Can we do it?

Not in general. Consider the VAE model

\[
z \sim p(z) := \mathcal{N}(z; 0, I) \quad ("easy") \\
x \sim p(x|z)
\]

Even though \( p(z) \) is “easy”, \( p(z)p(x|z)dz \) is “hard”: need to enumerate all \( z \) that could have produced \( x \).

Even if \( x = f_\phi(z) \) is deterministic, could be hard to reason about \( z \) that produced \( x \).

But if \( f_\phi \) is invertible, we can do it!

Change of Variable in 1D (False Start)

**Example** (false start). Suppose

\[
Z \sim \text{Unif}(0, 1) \\
X = 2Z + 1 := f(Z)
\]

What is \( p_X(2) \)?

Easy to guess \( p_X(2) = p_Z(f^{-1}(2)) = p_Z(\frac{1}{2}) = 1 \). **Wrong**.

Correct answer is \( p_X(2) = \frac{1}{2} \). Easy to see \( X \sim \text{Unif}(1, 3) \).
Volume Change

Density at points is not preserved under transformations.

**Issue**: transformations also “stretch” or “compress” space (change volume)

Change of Variable in 1D

**Correct approach**: probability of regions is preserved

Informal derivation: if $X = f(Z)$ and $f$ is invertible and with inverse $g$ then

$$p_X(x) dx = p_Z(z) dz$$

$$p_X(x) = p_Z(z) \left| \frac{dz}{dx} \right|$$

$$p_X(x) = p_Z(g(x)) |g'(x)|$$

(Also assume $f$ differentiable.)

Change of Variable in 1D Proof

We can derive this more formally using the fact that

$$\Pr(X \in [c, d]) = \Pr(Z \in [g(c), g(d)])$$

Let $z = g(x)$. We can also write

$$p_X(x) = p_Z(g(x)) |g'(x)|$$

since $g'(x) = 1/f'(z)$ (calculus fact).

For $c < d$ we have:

$$\int_c^d p_X(x) dx = \Pr(c \leq X \leq d)$$

$$= \Pr(g(c) \leq Z \leq g(d))$$
Change of Variable: General Case

Suppose \( z \sim p_z(z) \) and \( x = f(z) \) for invertible, differentiable \( f : \mathbb{R}^D \rightarrow \mathbb{R}^D \) with inverse \( g \). Then

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

The matrix \( \frac{\partial f(z)}{\partial x} \in \mathbb{R}^{D \times D} \) is the Jacobian of \( g \). Its \((i,j)\)th entry is \( \frac{\partial f_i(z)}{\partial x_j} \).

It’s also true that \( \frac{\partial g(x)}{\partial x} = \left( \frac{\partial f(z)}{\partial x} \right)^{-1} \) for \( z = g(x) \). So we often call \( \frac{\partial g(x)}{\partial x} \) the inverse Jacobian of \( f \).

Another version, often convenient. Let \( z = g(x) \). Then

\[
\det(A^{-1}) = \frac{1}{\det(A)}
\]

\[
\det \left( \frac{\partial f(z)}{\partial x} \right)^{-1} = \det(A)^{-1}
\]

Geometrically, \( \det \frac{\partial f(z)}{\partial x} \) describes how much \( f \) changes the volume of a small hypercube.
Normalizing Flow

A normalizing flow uses a simple prior and learned transformation to model data

\[ \mathcal{N}(z; 0, I) \in \mathbb{R}^D \]
\[ z \sim p_z(z) \text{ simple (e.g., Gaussian)} \]
\[ x = f_\theta(z) \text{ invertible} \]

By the change-of-variable formula, the density is

\[ p_x(x; \theta) = p_z(f_\theta^{-1}(x)) \cdot \left| \det \frac{\partial f_\theta^{-1}(x)}{\partial x} \right| \]

Learning and Prediction

- Learning by maximum likelihood. Find \( \theta \) to maximize

\[ \frac{1}{N} \sum_{n=1}^{N} \log p(x^{(n)}; \theta) = \frac{1}{N} \sum_{n=1}^{N} \left( \log p_z(x^{(n)}; \theta) + \log \left| \det \frac{\partial f_\theta^{-1}(x^{(n)})}{\partial x^{(n)}} \right| \right) \]

- Learning uses inverse mapping \( x \mapsto z \) and change of variables formula

- Prediction (sampling) uses simple distribution for \( z \) and forward mapping \( z \mapsto x \)

Building Flow Models

Most often \( f_\theta = f_\theta^m \circ \cdots \circ f_\theta^1 \) is a composition or “flow” of many transformations:

\[ z_0 \sim p_{z_0}(z_0) \text{ simple} \]
\[ z_1 = f_\theta^1(z_0) \]
\[ z_2 = f_\theta^2(z_1) \]
\[ \vdots \]
\[ x = z_m = f_\theta^m(z_{m-1}) \]

The density is

\[ p_x(x; \theta) = p_{z_0}(f_\theta^{-1}(x)) \cdot \prod_{j=1}^{m} \left| \det \frac{\partial f_\theta^{-1}(z_j)}{\partial z_j} \right| \]

(Uses rules for Jacobian of composition and product of determinants.)
### Building Flow Models

To build a flow model we need

- A distribution $p(z)$ that is "easy". Can sample and compute density. $\sqrt{N(0,I)}$
- Transformations $f_0$ that are
  - Always invertible
  - Allow us to compute the determinant easily. In general, it is $O(D^3)$ — too expensive!
  - Key idea: choose transformations with special structure
  - **Sufficiently complex**

### Real-NVP

There are many constructions that ensure a triangular Jacobian. We’ll look at one: “Real-NVP”. We split $z$ and $x$ into two equal-sized parts of size $d = D/2$:

$$z = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}, \quad x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}.$$  

The forward mapping $z \mapsto x$ is

- $x_1 = z_1$ (identity)
- $x_2 = \mu_0(z_1) + z_2 \odot \exp(\alpha_0(z_1))$ (shift and scale $z_2$ based on $z_1$)

where $\mu_0(\cdot)$ and $\alpha_0(\cdot)$ are neural networks from $\mathbb{R}^d \to \mathbb{R}^d$.

### Triangular Jacobian

The inverse mapping is $x \mapsto z$ is therefore

$$z_1 = x_1$$  (identity)

$$z_2 = (x_2 - \mu_0(x_1)) \odot \exp(-\alpha_0(x_1))$$  (unshift and unscale $x_2$ based on $x_1$)

The Jacobian of the forward mapping and its determinant are

$$J = \frac{\partial x}{\partial z} = \begin{bmatrix} I_d & 0 \\ \frac{\partial \mu_0}{\partial z_1} & \frac{\partial \alpha_0}{\partial z_1} \end{bmatrix} \text{diag}(\exp(\alpha_0(z_1)))$$

$$\det(J) = \prod_{i=1}^d \exp(\alpha_0(z_1)_i) = \exp\left(\sum_{i=1}^d \alpha_0(z_1)_i\right)$$

Change order of dimensions in different layers, so sometimes $z_2 \mapsto x_2$ is identity instead.
Demo

- Demo: implementation and 2d density estimation with Real-NVP
- There are tons of examples on the internet of images generated by flows. Take a look.
- Flows have been used for tons of applications
  - They can be extremely good for VI.
  - They are good at generating images, but not the most competitive models right now (if you care). One reason is they restrict $f_\theta$ too much. Some more competitive current models descend from normalizing flows.