COMPSCI 688: Probabilistic Graphical Models

Last time: GPs
- GPML

Next time
- Flows?
- PPL?

Lecture 23: Normalizing Flows

- generative AI
- variational inference

Dan Sheldon

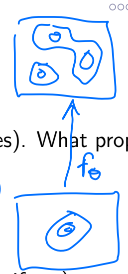
Manning College of Information and Computer Sciences
University of Massachusetts Amherst

HW 5 due Friday

Partially based on materials by Benjamin M. Marlin (marlin@cs.umass.edu) and Justin Domke (domke@cs.umass.edu)

Overview

Motivation: Transforming a Simple Distribution



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

$\max_\theta \sum \log p_\theta(x^{(n)})$

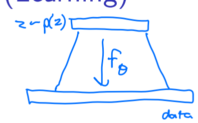
- Easy to sample (useful for generation)
- Easy to evaluate density (useful for learning)

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 $p_\theta(\mathbf{x})$ but with no noise

VAEs:
$\mu = f_\theta(z)$
$x \sim \mathcal{N}(\mu, \sigma^2 I)$

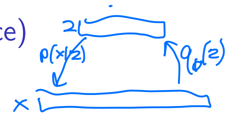
$$\begin{aligned} \mathbf{z} &\sim p(\mathbf{z}) \text{ simple, e.g. } \mathcal{N}(0, I) \\ \mathbf{x} &= f_\theta(\mathbf{z}) \end{aligned} \implies p_\theta(\mathbf{x})$$

Could we learn $p_\theta(\mathbf{x})$ "directly" by MLE?

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

Demo

Motivation: Transforming a Simple Distribution (Inference)



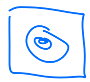
Also useful in variational inference, e.g. $q_\phi(\mathbf{z})$ in VAEs

Goal: $q_\phi(\mathbf{z}) \approx p(\mathbf{z}|\mathbf{x})$ where p, $\mathbf{x}$ are given. We used reparameterized Gaussians:

$$\begin{aligned} \epsilon &\sim \mathcal{N}(0, I) \\ \mathbf{z} = \mathcal{T}_\phi(\epsilon) &= L\epsilon + \mu \end{aligned} \implies \mathbf{z} \sim q_\phi(\mathbf{z}) = \mathcal{N}(\mathbf{z}; \mu, LL^T)$$

$\sigma \odot \epsilon + \mu$

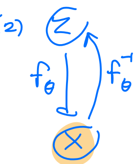
What if we used complex $\mathcal{T}_\phi(\epsilon)$ (e.g. neural net) instead?



- Would have a rich class of variational distributions.
- Could easily sample from $q_\phi(\mathbf{z})$
- For ELBO, need to compute density $q_\phi(\mathbf{z})$ under transformation $\mathcal{T}_\phi$. Can we do it?

Can we do it?

easy $p(z)$



Not in general. Consider the VAE model

$$\begin{aligned} \mathbf{z} &\sim p(\mathbf{z}) := \mathcal{N}(\mathbf{z}; 0, I) \quad (\text{“easy”}) \\ \mathbf{x} &\sim p(\mathbf{x}|\mathbf{z}) \end{aligned}$$

Even though $p(\mathbf{z})$ is “easy”, $p(\mathbf{x}) = \int p(\mathbf{z})p(\mathbf{x}|\mathbf{z})d\mathbf{z}$ is “hard”: need to enumerate all $\mathbf{z}$ that could have produced $\mathbf{x}$.

f_θ: many-to-one

Even if $\mathbf{x} = f_\theta(\mathbf{z})$ is deterministic, could be hard to reason about $\mathbf{z}$ that produced $\mathbf{x}$.

But if f_θ is **invertible**, we can do it!

Change of Variable

Change of Variable in 1D (False Start)

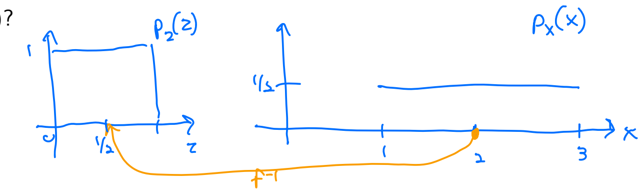
1D: density = $\frac{\text{prob}}{\text{length}}$

prob = $\int p_x(x)\,dx$

Example (false start). Suppose

$Z \sim$ Unif$(0, 1)$
$X = 2Z + 1 := f(Z)$

What is $p_X(2)$?



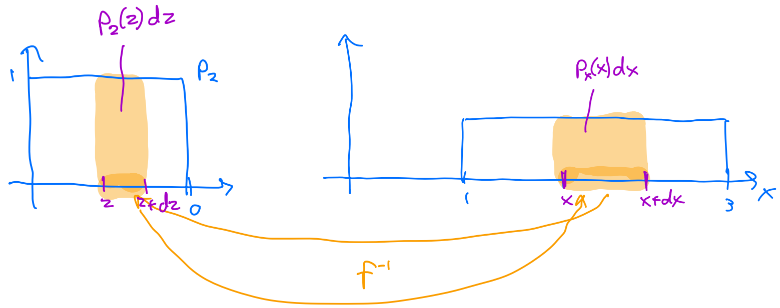
Easy to guess $p_X(2) = p_Z(f^{-1}(2)) = p_Z(\frac{1}{2}) = 1$. **Wrong**.

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

Volume Change

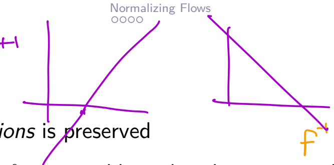
Density **at points** is not preserved under transformations.

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



Change of Variable in 1D

$x = 2z+1$



Correct approach: probability of *regions* is preserved

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

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

$$p_X(x) = p_Z(z)\left|\frac{dz}{dx}\right| \qquad z = g(x), \quad \frac{dz}{dx} = g'(x)$$

$$p_X(x) = p_Z(g(x))|g'(x)| \qquad (g = f^{-1})$$

(Also assume f differentiable.)

Change of Variable in 1D

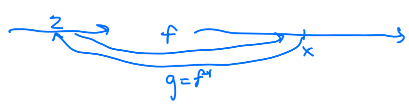
Formal statement: suppose $X = f(Z)$ for invertible, differentiable f with inverse g. Then

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

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

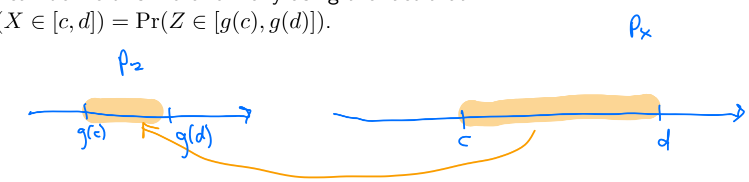
$$p_X(x) = p_Z(z)\left|\frac{1}{f'(z)}\right|$$

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



Change of Variable in 1D Proof

We can derive this more formally using the fact that $\Pr(X \in [c,d]) = \Pr(Z \in [g(c), g(d)])$.



For $c < d$ we have:

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

$$= \Pr(g(c) \le Z \le g(d))$$

$$= \int_{g(c)}^{g(d)} p_Z(z)dz$$

$$= \int_{c}^{d} p_Z(g(x))g'(x)dx$$

$z = g(x)$

$dz = g'(x)\,dx$

The last line uses the calculus change of variable formula with the substitution $z = g(x)$. (So this is really the same change of variable formula.)

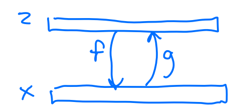
Since c and d are arbitrary, by comparing the integrands we see that $p_X(x) = p_Z(g(x))g'(x)$.

Change of Variable: General Case

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

$\in \mathbb{R}^d$ $\in \mathbb{R}^d$

$$p_x(\mathbf{x}) = p_{\mathbf{z}}(g(\mathbf{x})) \cdot \left|\det \frac{\partial g(\mathbf{x})}{\partial \mathbf{x}}\right|$$



- The matrix $\frac{\partial g(\mathbf{x})}{\partial \mathbf{x}} \in \mathbb{R}^{D \times D}$ is the *Jacobian* of g. It's (i, j)th entry is $\frac{\partial g_i(x)}{\partial x_j}$
- It's also true that $\frac{\partial g(\mathbf{x})}{\partial \mathbf{x}} = \left(\frac{\partial f(\mathbf{z})}{\partial \mathbf{z}}\right)^{-1}$ for $\mathbf{z} = g(\mathbf{x})$. So we often call $\frac{\partial g(\mathbf{x})}{\partial \mathbf{x}}$ the *inverse Jacobian* of f

- Another version, often convenient. Let $\mathbf{z} = g(\mathbf{x})$. Then

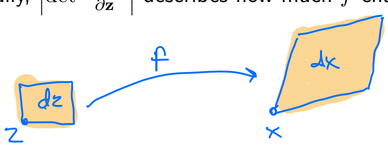
$\det(A^{-1}) = \frac{1}{\det(A)}$

$= \det(A)^{-1}$

$$p_{\mathbf{x}}(\mathbf{x}) = p_{\mathbf{z}}(\mathbf{z}) \cdot \left|\det \frac{\partial f(\mathbf{z})}{\partial \mathbf{z}}\right|^{-1}$$

$\frac{\mathrm{Vol}(dx)}{\mathrm{Vol}(dz)}$

- Geometrically, $\left|\det \frac{\partial f(\mathbf{z})}{\partial \mathbf{z}}\right|$ describes how much f changes the volume of a small hypercube.



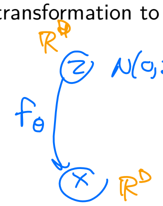
Normalizing Flows

Normalizing Flow

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

$\mathcal{N}(z; 0, I) \in \mathbb{R}^D$

$\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})$ simple (e.g., Gaussian)
$\mathbf{x} = f_\theta(\mathbf{z})$ invertible



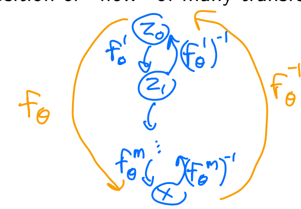
By the change-of-variable formula, the density is

$$p_{\mathbf{x}}(\mathbf{x}; \theta) = p_{\mathbf{z}}(f_\theta^{-1}(\mathbf{x})) \cdot \left| \det \frac{\partial f_\theta^{-1}(\mathbf{x})}{\partial \mathbf{x}} \right|$$

Normalizing Flow

Most often $f_\theta = f_\theta^m \circ \cdots \circ f_\theta^1$ is a composition or "flow" of many transformations:

$\mathbf{z}_0 \sim p_{\mathbf{z}_0}(\mathbf{z}_0)$ simple
$\mathbf{z}_1 = f_\theta^1(\mathbf{z}_0)$
$\mathbf{z}_2 = f_\theta^2(\mathbf{z}_1)$
$\vdots$
$\mathbf{x} = \mathbf{z}_m = f_\theta^m(\mathbf{z}_{m-1})$

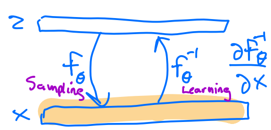


The density is

$$p_{\mathbf{x}}(\mathbf{x}; \theta) = p_{\mathbf{z}_0}(f_\theta^{-1}(\mathbf{x})) \cdot \prod_{j=1}^{m} \left| \det \frac{\partial (f_\theta^j)^{-1}(\mathbf{z}_j)}{\partial \mathbf{z}_j} \right|$$

(Uses rules for Jacobian of composition and product of determinants.)

Learning and Prediction



- Learning by maximum likelihood. Find θ to maximize

$$\frac{1}{N}\sum_{n=1}^{N} \log p(\mathbf{x}^{(n)}; \theta) = \frac{1}{N}\sum_{n=1}^{N} \left(\log p_{\mathbf{z}}(f_\theta^{-1}(\mathbf{x}^{(n)})) + \log \left| \det \frac{\partial f_\theta^{-1}(\mathbf{x}^{(n)})}{\partial \mathbf{x}^{(n)}} \right| \right)$$

- Learning uses inverse mapping $\mathbf{x} \mapsto \mathbf{z}$ and change of variables formula
- Prediction (sampling) uses simple distribution for $\mathbf{z}$ and forward mapping $\mathbf{z} \mapsto \mathbf{x}$

Building Flow Models

Building Flow Models

To build a flow model we need

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

Triangular Jacobian

$$(z_1, z_2, \dots, z_i, \dots, z_D) \xmapsto{f} (x_1, x_2, \dots, x_i, \dots, x_D)$$

$$J = \frac{\partial f}{\partial \mathbf{z}} = \begin{bmatrix} \frac{\partial f_1}{\partial z_1} & \cdots & \frac{\partial f_1}{\partial z_D} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_D}{\partial z_1} & \cdots & \frac{\partial f_D}{\partial z_D} \end{bmatrix}$$

Suppose $x_i = f_i(\mathbf{z})$ only depends on $z_1, \dots, z_i$. Then

$$J = \frac{\partial f}{\partial \mathbf{z}} = \begin{bmatrix} \frac{\partial f_1}{\partial z_1} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ \frac{\partial f_D}{\partial z_1} & \cdots & \frac{\partial f_D}{\partial z_D} \end{bmatrix}$$

is lower triangular $\implies$ the determinant is the product of the diagonal entries of J, can be computed in **linear time**.

Real-NVP

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

$$\mathbf{z} = \begin{bmatrix} \mathbf{z}_1 \\ \mathbf{z}_2 \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix}.$$

The forward mapping $\mathbf{z} \mapsto \mathbf{x}$ is

$$\begin{aligned} \mathbf{x}_1 &= \mathbf{z}_1 && \text{(identity)} \\ \mathbf{x}_2 &= \mu_\theta(\mathbf{z}_1) + \mathbf{z}_2 \odot \exp(\alpha_\theta(\mathbf{z}_1)) && \text{(shift and scale } \mathbf{z}_2 \text{ based on } \mathbf{z}_1) \end{aligned}$$

where $\mu_\theta(\cdot)$ and $\alpha_\theta(\cdot)$ are neural networks from $\mathbb{R}^d \to \mathbb{R}^d$.

The inverse mapping is $\mathbf{x} \mapsto \mathbf{z}$ is therefore

$$\begin{aligned} \mathbf{z}_1 &= \mathbf{x}_1 && \text{(identity)} \\ \mathbf{z}_2 &= (\mathbf{x}_2 - \mu_\theta(\mathbf{x}_1)) \odot \exp(-\alpha_\theta(\mathbf{x}_1)) && \text{(unshift and unscale } \mathbf{x}_2 \text{ based on } \mathbf{x}_1) \end{aligned}$$

The Jacobian of the forward mapping and its determinant are

$$J = \frac{\partial \mathbf{x}}{\partial \mathbf{z}} = \begin{bmatrix} I & 0 \\ \frac{\partial \mathbf{x}_2}{\partial \mathbf{z}_1} & \text{diag}(\exp(\alpha_\theta(\mathbf{z}_1))) \end{bmatrix}$$

$$\det(J) = \prod_{i=1}^{d} \exp\left(\alpha_\theta(\mathbf{z}_1)_i\right) = \exp\left(\sum_{i=1}^{d} \alpha_\theta(\mathbf{z}_1)_i\right)$$

Change order of dimensions in different layers, so sometimes $\mathbf{z}_2 \mapsto \mathbf{x}_2$ 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_θ too much. Some more competitive current models descend from normalizing flows.