Deep Learning-based approaches to
(1) Density est. (2) sampling from dstns

Setting: $X_1, \dots X_n \overset{iid}{\sim} P_X$

We could like to (1) estimate the density P_X
(2) sample from P_X

Approaches:

deep learning based

(1) Parametric (mixture models)
(2) Energy-based models (markov random fields)
(3) VAE (variational autoencoders) Kingma, Welling 2013
(4) GAN (generative adversarial networks Goodfellow et al. 2014
(5) Normalizing Flows
(6) Diffusion

	(1)	(2)	(3)	(4)	(5)	(6)
(1) ability to sample	mcmc	✓	✓	✓	✓	✓
(2) evaluate density	unnorm'd density	✓	X	X	✓	✓
(3) expressiveness dimensionality scaling	m	X	m	M	✓	
(4) training/ estimate/ learn	MCMC	ML	SGD ELBO	SGD game theory	SGD	

Normalizing flows

Approach:

(1) Select a r.v. Z, that has a density that is easy to evaluate and whose distn is easy to sample from.

(2) Assume there exists an invertible (bijection) differentiable function f_θ s.t. $Z = f_\theta(X)$ $f: \mathcal{X} \to \mathcal{Z}$

(3) Find $\hat{\theta} = \underset{\theta \in \Theta}{\operatorname{argmax}} \log(L(D, \theta))$ (where $X \sim P_X$, $Z \sim P_Z$)

$= \underset{\theta \in \Theta}{\operatorname{argmax}} \log\left(\prod_{i=1}^{n} p_X(x_i)\right)$

$= \underset{\theta \in \Theta}{\operatorname{argmax}} \sum_{i=1}^{n} \log(p_X(x_i))$

discussed on page 13

Q how is this a fnc of θ?

Recall: Change of variable formula

Given $X \sim P_X$, $Z \sim P_Z$, and $f: \mathcal{X}^p \to \mathcal{Z}^p$ where f is a bijection & differentiable, then

$$p_X(x) = p_Z(f(x)) \left|\det(J_f(x))\right|$$

where $J_f(x)$ is the Jacobian of f evaluated at x
e.g. matrix of partial derivatives $f: \mathcal{X}^p \to \mathcal{Z}^q$

$$\begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_p} \\ \vdots & & \\ \frac{\partial f_q}{\partial x_1} & \cdots & \frac{\partial f_q}{\partial x_p} \end{bmatrix}$$

Note: $\det(X)$ is only defined for square matrices so $\mathcal{X} \cong \mathcal{Z}$ must have the same dim.

(cont.) $\hat{\theta} = \underset{\theta \in \Theta}{\operatorname{argmax}} \sum_{i=1}^{n} \log(p_{\mathcal{X}}(x_i))$

$\overset{\text{change of var}}{=} \underset{\theta}{\operatorname{argmax}} \sum_{i=1}^{n} \log\left[p_{\mathcal{Z}}(f_\theta(x_i)) \left| \det(J_{f_\theta}(x_i)) \right| \right]$

$= \underset{\theta}{\operatorname{argmax}} \sum_{i=1}^{n} \log\left[p_{\mathcal{Z}}(f_\theta(x_i)) \right] + \sum_{i=1}^{n} \log\left(\left| \det J_{f_\theta}(x_i) \right| \right)$

↑ fit with gradient descent

$\sum_{k=1}^{K} \log\left[\det(J_{f_{\theta,k}}(x_i)) \right]$

(4) (A) To evaluate the density of a new point:

$$p_{\mathcal{X}}(x_j) = p_{\mathcal{Z}}(f_{\hat{\theta}}(x_j)) \left| \det(J_{f_{\hat{\theta}}}(x_j)) \right|$$

(B) To sample from $P_{\mathcal{X}}$,

(1) sample from $P_{\mathcal{Z}}$ to obtain z_j

(2) $x_j = f_{\hat{\theta}}^{-1}(z_j)$ ← why invertibility is (practically) needed

Terminology: f is called a "flow" (as it flows P_X to P_Z) and since Z is usually chosen to be $\sim N$, $\rightarrow$ normalizing flows

---

Takeaways:

- unusual application of C.o.V. ("backwards")
- don't directly estimate the quantity/function of interest. Instead learn a map from it to a quantity/function you know.

Potential Trouble

(1) dimensionality of f (must be $\mathbb{R}^p \to \mathbb{R}^p$) may cause issues for:

(1A) f_θ^{-1} and (1B) $\det(J_{f_\theta}(x_i))$

(2) need f to be very (arbitrarily) "flexible"

Next: how to construct such f's

Starting with "programatic" approach to achieving (2)

It would be nice to break f into simple building blocks. (so that we can build arbitrarily complex flows from simple building blocks)

(the deep learning approach)

Recall: invertible, differentiable functions are closed under composition.

If f_j are invertible & diff. then

$$f(x) = f_k \circ f_{k-1} \circ \cdots \circ f_1(x) \quad (\star)$$

is also, in particular

$$f^{-1}(z) = f_1^{-1} \circ f_2^{-1} \cdots \circ f_k^{-1}(z)$$

Recall: $\det(J_f) = \det\left(\prod_{k=1}^{K} J_{f_k}\right) = \prod_{k=1}^{K} \det(J_{f_k})$

(for f of the type defined in $\star$)

$\Rightarrow$ so it is enough to have an easy-to-calc $\det(J_{f_k})$

Now, we consider how to construct f that address (1A) & (1B) starting with simplest option:

Linear flow

$$z = f_k(x) = A_k x + b_k$$

inverse: $f_k^{-1}(z) = A_k^{-1}(z - b_k)$

$$\det(J_{f_k}) = \det(A_k)$$

issues: (1) linear fns are closed under composition.

(2) inv and det can be computationally hard depending on structure of A_k

First we will address (2) by restricting f

What kinds of f have easy f_k^{-1} and $\det(J_{f_k})$?

A_k	inverse (f_k^{-1}) (A_k^{-1})	det (J_{f_k}) $(\det(A_k))$
full	$O(p^3)$	$O(p^3)$
diag	$O(p)$	$O(p)$
☆ triangular	$O(p^2)$	$O(p)$
block diag	$O(c^3 p)$	$O(c^3 p)$

Recall:

(A) The determinant of a triangular matrix is the product of the diagonals (can see via Laplace expansion)

$$J = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & & 0 \\ & \ddots & \\ \frac{\partial f_p}{\partial x_1} & \cdots & \frac{\partial f_p}{\partial x_p} \end{bmatrix} \qquad \det(J) = \prod_{j=1}^{p} \frac{\partial f_j}{\partial x_j}$$

(B) for a block triangular matrix,

$$J = \begin{bmatrix} A & 0 \\ C & B \end{bmatrix} \qquad \det(J) = \det(A)\det(B)$$

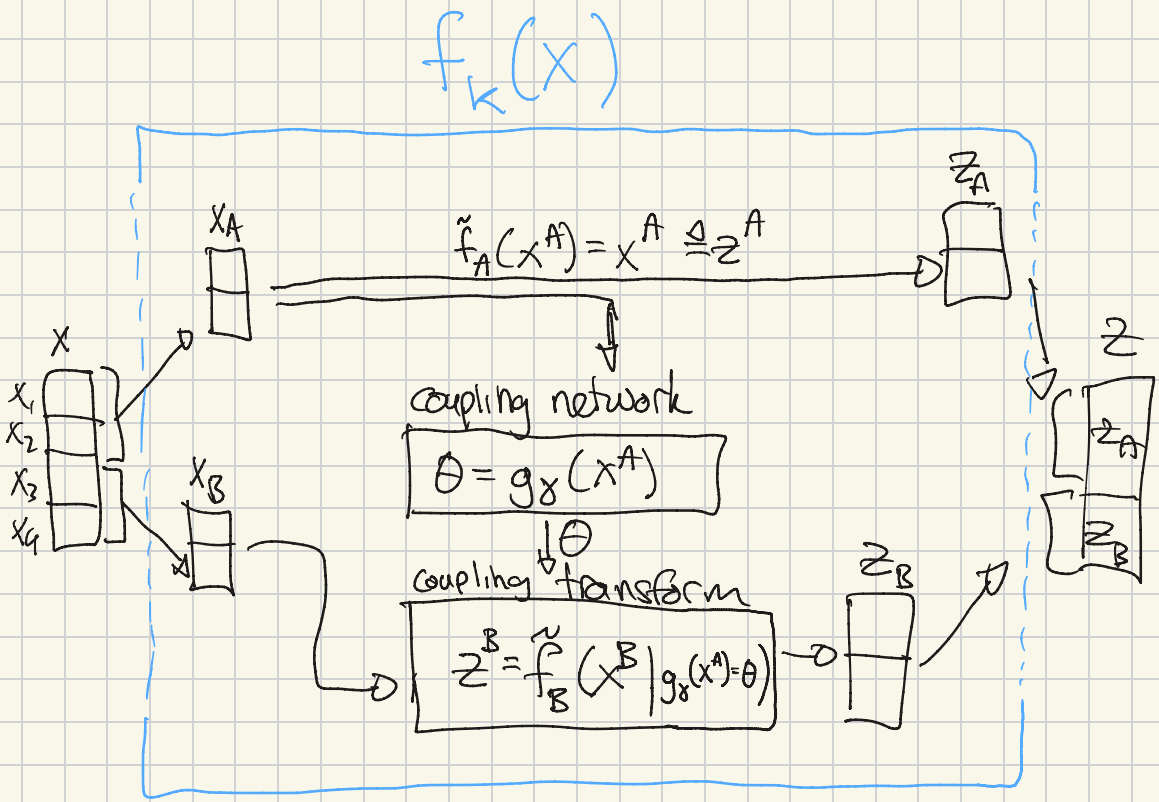
Now, how to build a flow (f_k) that is more expressive than the linear flow while also having a "simple" Jacobian?

Next, we look at an example of such a flow (whose Jacobian has a block triangular structure)

coupling flow

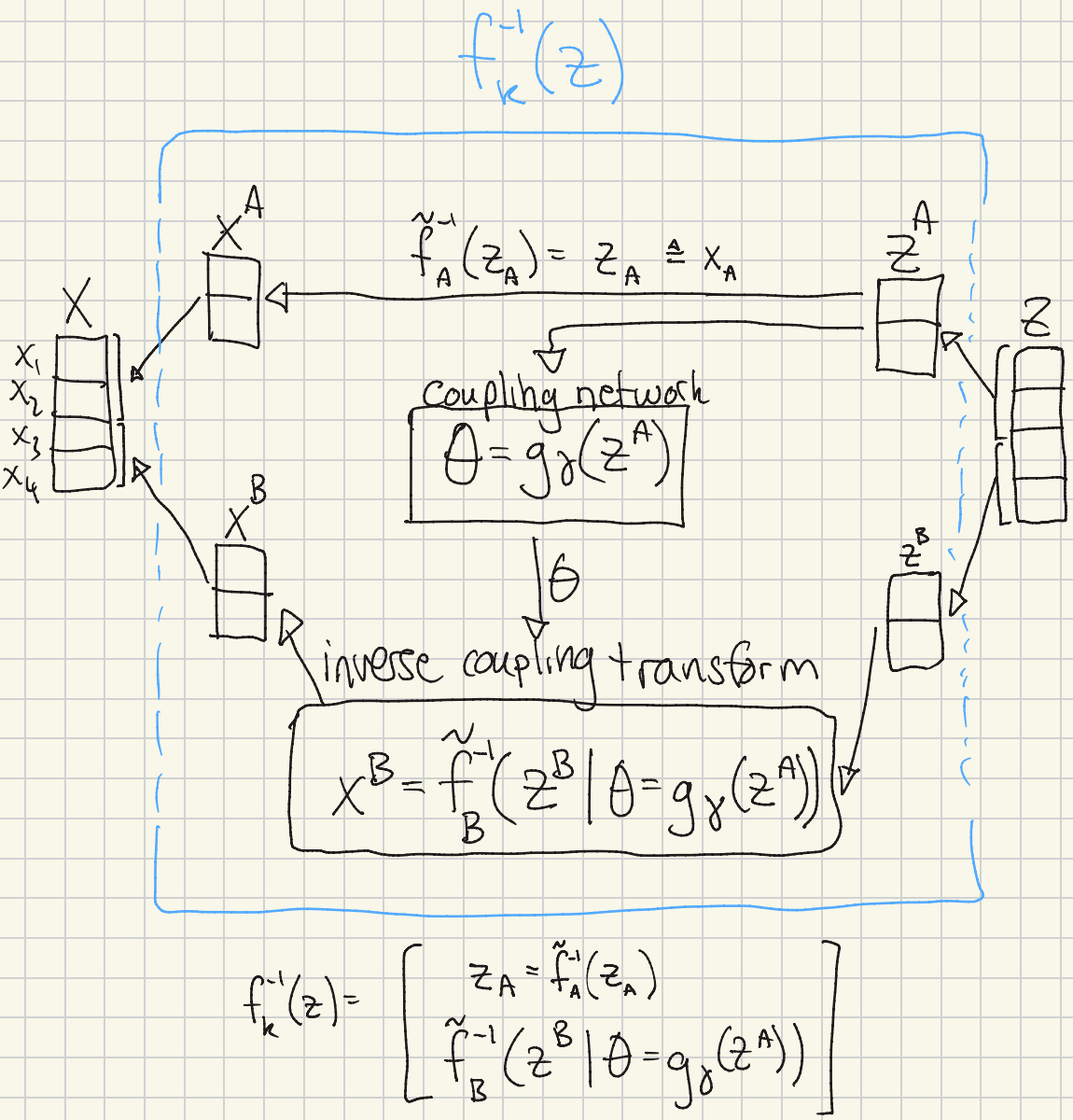
- This flow (f_k) has a general structure that results in a "simple" $\det(J_{f_k})$ while also allowing f_k to capture complex dependencies (via g_γ introduced below)
- Next we will define f_k, f_k^{-1}, and J_{f_k}

- First, we define the "forward" flow $(f_k(x))$



$$f_k(x) = \begin{bmatrix} x^A = \tilde{f}_A(x_A) \\ \tilde{f}_B(x^B \mid \theta = g_\gamma(x^A)) \end{bmatrix} \triangleq \begin{bmatrix} f_k^A(x) \\ f_k^B(x) \end{bmatrix}$$

- Next we define the inverse $f_k^{-1}(z)$:



$$f_k^{-1}(z) = \begin{bmatrix} z_A = \tilde{f}_A^{-1}(z_A) \\ \tilde{f}_B^{-1}(z^B \mid \theta = g_\gamma(z^A)) \end{bmatrix}$$

- Finally, we define the Jacobian (J_{f_k}):

$$J_{f_k}(x) \simeq \begin{bmatrix} \frac{\partial f_k^A}{\partial x_A} & \frac{\partial f_k^A}{\partial x_B} \\ \frac{\partial f_k^B}{\partial x_A} & \frac{\partial f_k^B}{\partial x_B} \end{bmatrix} = \begin{bmatrix} \text{I} & \text{II} \\ \text{III} & \text{IV} \end{bmatrix}$$

$$= \begin{bmatrix} \text{I} & \text{O} \\ \frac{\partial}{\partial x_A}\left[\tilde{f}_B(x^B \mid g(x^A)=\theta)\right] & \frac{\partial}{\partial x_B}\left[\tilde{f}_B(x^B \mid g(x^A)=\theta)\right] \end{bmatrix}$$

In the IV partial derivative, θ is constant (wrt x_B) $\Rightarrow 0$

$$\begin{bmatrix} \text{I} & \text{O} \\ \frac{\partial}{\partial x_A}\left[\tilde{f}_B(x^B \mid g(x^A)=\theta)\right] & \frac{\partial}{\partial x_B}\left[\tilde{f}_B(x^B \mid \theta)\right] \end{bmatrix}$$

III might be complex, but is ignored in the determinant $\downarrow_0$

$$= \begin{bmatrix} \text{I} & \text{O} \\ \frac{\partial}{\partial x_A}\left[\tilde{f}_B(x^B \mid g(x^A)=\theta)\right] & J_{\tilde{f}_B}(x^B \mid \theta) \end{bmatrix}$$

Some important properties of f_k:

(1) Invertibility: For f_k^{-1} to be simple, it is sufficient for $\tilde{f}_B$ to be easy to invert (w.r.t x_B)

(2) The determinant of the Jacobian (J_{f_k}):

J_{f_k} is block (lower) triangular, so

$$\det(J_{f_k}) = \det(I) \cdot \det(J_{\tilde{f}_B}(x^B|\theta)) = \det(J_{\tilde{f}_B}(x^B|\theta))$$

Thus, for f_k to have a tractable $\det(J_{f_k})$, it is sufficient for $\det(J_{\tilde{f}_B}(x^B|\theta))$ to be tractable.

This will be the case if $\tilde{f}_B$ is a simple function wrt x_B (not necessarily wrt x_A).

(3) Expressivity of f_k: Notice that no restrictions are needed on the complexity/form of $g_\gamma(x^A)$ (to produce easy f_k^{-1} and $\det(J_{f_k})$).

Thus, $\tilde{f}_B$ can be made arbitrarily complex (ie nonlinear) as a function of x_A through its dependence on $\theta = g_\gamma(x_A)$ via the choice of this g_γ (e.g. a DNN or similar)

What are some choices for $\tilde{f}_B$ that are easy to invert and have simple $\det(J_{\tilde{f}_B})$?

- Additive (NICE, Dinh et al. 2014)

$$\tilde{f}_B(x_B \mid t(x_A)) = x_B + t(x_A)$$

$$\tilde{f}_B^{-1}(z_B \mid t(z_A)) = z_B - t(z_A)$$

$$\det(J_{\tilde{f}_B}) = I$$

- Affine (RealNVP, Dinh et al. 2016)

$$\tilde{f}_B(x_B \mid t(x_A), s(x_A)) = \exp\{s(x_A)\} \odot x_B + t(x_A)$$

$$\tilde{f}_B^{-1}(z_B \mid t(z_A), s(z_A)) = \frac{(z_B - t(z_A))}{\exp\{s(z_A)\}}$$

$$\det(J_{\tilde{f}_B}) = \det\left(\begin{bmatrix} \exp\{s_1(x_A)\} & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & \exp\{s_q(x_A)\} \end{bmatrix}\right)$$

$$= \exp\left\{\sum_{j=1}^{q} s_j(x_A)\right\}$$

$q = \dim(x_B)$
$= \frac{p}{2}$ for the f_k we diagrammed

Now we have constructed an f_k that is expressive (via g_γ), with tractable f_k^{-1} and $\det(J_{f_k})$ (via $\tilde{f}_\beta$).

One outstanding issue remains, namely that directly composing f_k of the type we defined would leave x_A unchanged i.e.

$$\begin{bmatrix} x_A \\ z_B \end{bmatrix} = f(x) = (f_K \circ \cdots \circ f_1)(x)$$

One solution is to alternate f_k with random(⁉fixed) permutations (P_k), thus:

$$f(x) = (f_K \circ P_K \circ \cdots \circ f_1 \circ P_1)(x)$$

Note that permutations are invertible & differentiable, meeting the requirements for the change of variable formula, and they have simple inverse and $\det(J)$ (± 1).

Summary

Developed a general structure for flows that is expressive with simple f^{-1} & $\det(J_f)$

☆ One important outstanding issue. We began with the critical assumption that $\exists f_\theta \in F$ s.t. $Z \overset{d}{=} f_\theta(X)$ where $X \sim P_X, Z \sim P_Z$ but then we restricted our search space for f_θ (F) to a very specific function family (coupling flows). So,

(1) Is there any guarantee that f_θ exists such that $Z \overset{d}{=} f_\theta(X)$ for this particular (P_X, P_Z) pair?

(2) If yes, then is $f_\theta \in F$ also guaranteed? (i.e. can we be sure that one parametrization of the coupling flow is this f_θ?)

Key result for (1):

(1) Existence (Lemma 2.1, Bogachev et al 2005)

(roughly) For absolutely continuous P_X, P_Z on $\mathbb{R}^p$, $\exists!$ increasing, triangular mapping f_θ s.t. $Z \overset{d}{=} f_\theta(X)$. Where

- a mapping $f_\theta(x^{(1)}, \ldots, x^{(p)}) \triangleq (f_\theta^{(1)} \ldots f_\theta^{(p)})$ is triangular if $\forall j$ $f_\theta^{(j)}$ is a function of $(x^{(1)}, \ldots, x^{(j)})$ (and thus has triangular Jacobian)
- a mapping $f_\theta(x^{(1)}, \ldots, x^{(p)}) \triangleq (f_\theta^{(1)} \ldots f_\theta^{(p)})$ is increasing if $f_\theta^{(j)}$ is a (strictly) increasing function w.r.t $x^{(j)}$ $\forall j$

[see also Knöthe-Rosenblatt map/rearrangement]

Universality: Regarding (2) (ie whether $f_\theta \in \mathcal{F}$), this (unsurprisingly) depends on $\mathcal{F}$. Some classes of $\mathcal{F}$ for which this universality has been shown:

- neural autoregressive flows (Huang et al, 2018)
- augmented neural ODE (Dupont et al, 2019)
- sum of squares polynomial flow (Jaini et al 2019)

[see also a more detailed survey of the statistical properties of triangular flows: "Triangular Flows for Generative Modeling - Statistical Consistency, Smoothness Classes, and Fast Rates" Irons et al 2022]