Normalising Flows

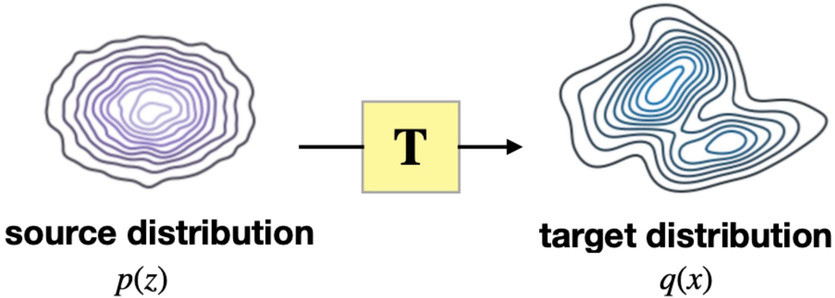
Denis Derkach

CS HSE faculty, Generative Models, spring 2020

Contents

Motivation

Quite often we want to sample from some distribution (but we do not know how to do this). What we know is how to sample from a simple pdf: Gaussian, uniform or something.



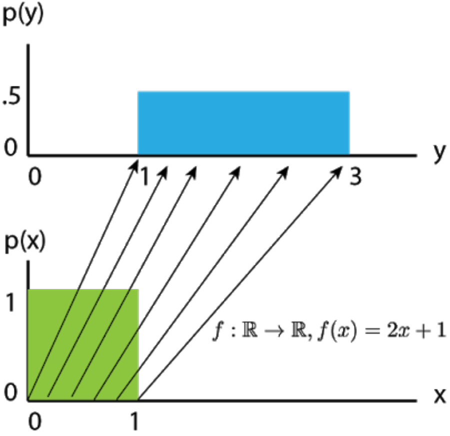
We want a deterministic map T from source to target density.

Conversation of Probability Mass

- Let's build a simple example:

$$X \sim U[0; 1]$$

- Take $f(X) = 2X + 1$. If we define $y = f(x)$. What is the probability of $p(y)$?
- Seems evident, since $f(X)$ is a simple affine transformation, we will have uniform distribution in y.
- Since probability mass integrates to 1, blue plot should be two times lower than green.



from E. Jang blog

RV Simple Transformation

› Let's consider a more complicated example:

$$p(x) = 2x, x \in [0;1];$$
$$Y : f(X) = x^2.$$

› We can compute pdf of Y, $p(y)$, using its CDF:

$$\begin{aligned} F_Y(y) &= \mathbb{P}(Y < y) = \mathbb{P}(X^2 < y) = \mathbb{P}(X < \sqrt{y}) = \\ &= F_X(\sqrt{y}) = \int\limits_0^{\sqrt{y}} p(x)dx = y. \end{aligned}$$

› Which means:

$$p(y) = F_Y'(y) = \frac{dF_X(\sqrt{y})}{dy} = 1$$

› Thus, $Y \sim U[0;1]$. We derived the pdf of Y.

1D variable transformation

We can generalize the above examples for 1D:

$$
\begin{aligned}
X &= f(Z); \\
f &: Z \mapsto X; \\
h(X) &= f^{-1}(X).
\end{aligned}
$$

Than

$$p_X(x) = p_Z(h(x))|h'(x)|.$$

Indeed, the above examples can be solved simply producing the derivative of inverse transformation.

Multidimensional transformations

We can generalise to a multidimensional case:

$$\begin{aligned} \mathbf{z} \sim q(\mathbf{z}) \in \mathbb{R}^d; \\ f: \mathbb{R}^d \to \mathbb{R}^d; \\ \mathbf{y} = f(\mathbf{z}). \end{aligned}$$

than pdf of y is

$$p(\mathbf{y}) = q(f^{-1}(\mathbf{y})) \left| \det \frac{\partial f^{-1}(\mathbf{y})}{\partial \mathbf{y}} \right| = q(\mathbf{z}) \left| \det \frac{\partial f(\mathbf{z})}{\partial \mathbf{z}} \right|^{-1},$$

NB1: x and z are continuous and both have dimension d.

NB2: the second equality comes from the inverse-function theorem.

Normalizing Flow Models

- If we take Z to be latent parameter and X to be observed variables.
- In a normalizing flow model, the mapping between Z and X is given by $f_\theta : \mathbb{R}^d \to \mathbb{R}^d$, which is deterministic and invertible such that $X = f_\theta(Z)$ and $Z = f_\theta^{-1}(X)$. Note that the density will be normalised
- Using change of variables, the marginal likelihood $p(x)$ is given is known.
- We than can apply a "flow" of transformations:

$$z_m = f_\theta^m \circ f_\theta^{m-1} \cdots \circ f_\theta^1(z_0) = f_\theta^m(f_\theta^{m-1}(\ldots(f_\theta^1(z_0)))) = f_\theta(z_0),$$

where z_0 is a simple distribution (like Gaussian).

Properties of definition

For definition:

$$z_m = f_\theta^m \circ f_\theta^{m-1} \dots \circ f_\theta^1(z_0) = f_\theta^m(f_\theta^{m-1}(\dots(f_\theta^1(z_0)))) = f_\theta(z_0),$$

it's evident that:

$$p(\mathbf{x};\theta) = q(f_\theta^{-1}(\mathbf{z})) \prod_{i=1}^{m} \left| \det \frac{\partial (f_\theta^i)^{-1}(\mathbf{z^m})}{\partial \mathbf{z^m}} \right|.$$

Note that we have invertible transformations.

Finding Best Transformation

Since we calculate the pdf of the observable. We can easily produce a good estimate of the parameter. For a dataset D:

$$\theta^* = \arg\max_\theta \log p(\mathbf{x} = D; \theta).$$

Simple prior $p_Z(z)$ allows for efficient sampling and tractable likelihood evaluation.

Planar flows

Let's choose the transformation:

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

parameterized by $\theta = (w, u, b)$ where h(·) is an element-wise non-linearity, $u, w \in \mathbb{R}^d$, $b \in \mathbb{R}$

Jacobian for a single transformation will look like

$$\frac{\partial f}{\partial z} = \mathbb{I} + uh(w^T z + b)w^T$$

and the determinant is straightforward to calculate.

From D. Rezende Variational Inference with Normalizing Flows

Planar flows: example

Let's look at the specific example: $z \in \mathbb{R}^2$.

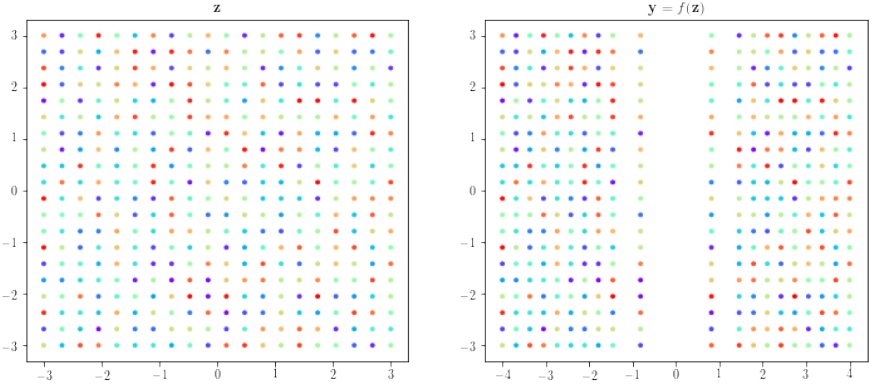
$$\begin{aligned} q(z) &= \mathcal{N}(0; \mathbb{I}); \\ w &= [5; 0]^T; \\ u &= [1; 0]^T; \\ b &= 0; \\ h(x) &= \tanh(x). \end{aligned}$$

The determinant of the Jacobian can be computed using previous slides' formula and the analytic pdf can then be computed.

From A. Fatir's blog

Planar flows: example

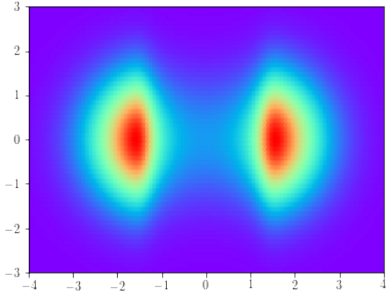
To illustrate $f(x)$, we generate two sets of dots:

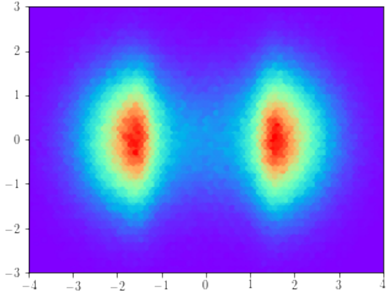


This is how the grid will behave under the f transformation.

Planar flows: example

We than produce a dataset, train the likelihood and obtain the empirical dataset:

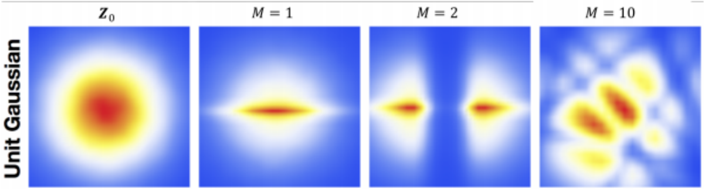




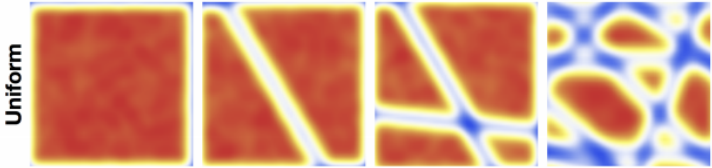
left: target pdf, right: empirical pdf.

Multiple Planar Flows

- Base distribution: Gaussian



- Base distribution: Uniform



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

Jacobian Problems

Computing likelihoods also requires the evaluation of determinants of $d \times d$ Jacobian matrices, where d is the data dimensionality.

- Computing the determinant for a $d \times d$ matrix is $\mathcal{O}(n^3)$: prohibitively expensive within a learning loop!
- Key idea: Choose transformations so that the resulting Jacobian matrix has special structure. For example, the determinant of a triangular matrix is the product of the diagonal entries, i.e., an $\mathcal{O}(n)$ operation.

Triangular Jacobian

- There always exists a unique (up to ordering) increasing triangular map that transforms a source density to a target density (see Bogachev et al. for detail).
- Let's consider the triangular Jacobian:

$$J = \begin{pmatrix} \frac{\partial f_1}{\partial z_1} & \dots & 0 \\ \dots & \dots & \dots \\ \frac{\partial f_1}{\partial z_n} & \dots & \frac{\partial f_n}{\partial z_n} \end{pmatrix}$$

  It describes the flow with transformation $x_i = f_i(z)$ that only depends on $z \leq i$.

NICE: Additive Coupling layers

- For n-dimensional transformation, choose d such that we have two disjoint subsets $z_{i:d}$ and $z_{d+1:n}$.
- Forward mapping: $z \mapsto x$:
  - $x_{1:d} = z_{1:d}$ (identity transformation).
  - $x_{d+1:n} = z_{d+1:n} + m_\theta(z_{1:d})$, m_θ is a neural network with parameters θ, d input units, and $n-d$ output units.
- inverse mapping $x \mapsto z$:
  - $z_{1:d} = x_{1:d}$ (identity transformation).
  - $x_{d+1:n} = z_{d+1:n} - m_\theta(z_{1:d})$.
- Jacobian of forward mapping:

$$J = \begin{pmatrix} I_d & 0 \\ \frac{\partial x_{d+1:n}}{\partial z_{1:d}} & I_{n-d} \end{pmatrix}$$

- $\det J = 1$, thus volume preserving.

From L. Dinh Non-linear Independent Components Estimation

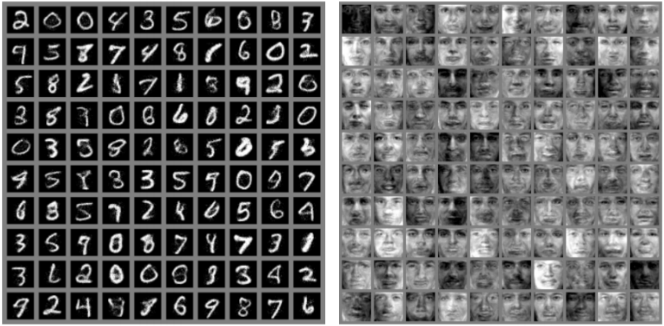
NICE: Rescaling layers

- Additive coupling layers are computed together (with arbitrary order of variables in each layer).
- Final layer rescaling:
- Forward mapping $z \mapsto x$:
  - $x_i = s_i z_i$, with $s_i > 0$, scaling factor.
- Inverse mapping $x \mapsto z$:
  - $z_i = x_i / s_i$.
- Jacobian of forward mapping:

$$J = diag(s)$$

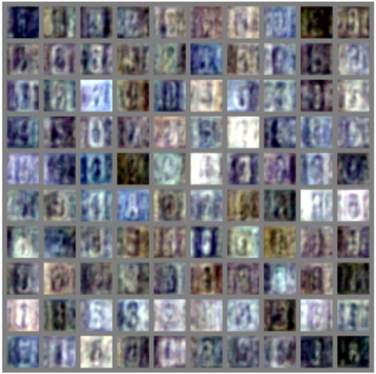
- $\det J = \prod_{i=1}^{n} s_i$.

NICE: Results

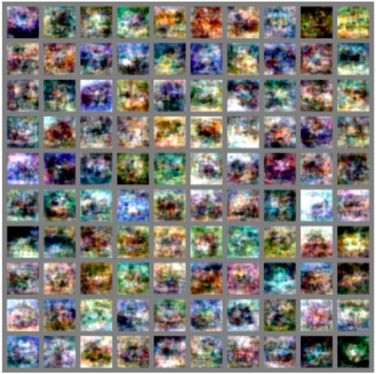


(a) Model trained on MNIST (b) Model trained on TFD

NICE: Results



(c) Model trained on SVHN



(d) Model trained on CIFAR-10

Real-NVP: Non-volume preserving NICE

- Forward mapping: $z \mapsto x$:
  - $x_{1:d} = z_{1:d}$ (identity transformation).
  - $x_{d+1:n} = z_{d+1:n} \odot \exp(\alpha_\theta(z_{1:d})) + \mu_\phi(z_{1:d}))$, μ_ϕ and α_θ are neural networks d input units, and $n-d$ output units.
- inverse mapping $x \mapsto z$:
  - $z_{1:d} = x_{1:d}$ (identity transformation).
  - $x_{d+1:n} = (x_{d+1:n} - \mu_\phi(x_{1:d})) \odot \exp(-\alpha_\theta(x_{1:d}))$.
- Jacobian of forward mapping:

$$J = \begin{pmatrix} I_d & 0 \\ \frac{\partial x_{d+1:n}}{\partial z_{1:d}} & diag(\exp(\alpha_\theta(z_{1:d}))) \end{pmatrix}$$

- $\det J = \exp\left(\sum_{i=d+1}^{n} (\alpha_\theta(z_{1:d}))_i\right)$.

From L. Dinh et al. Density estimation using Real NVP

r-NVP:results

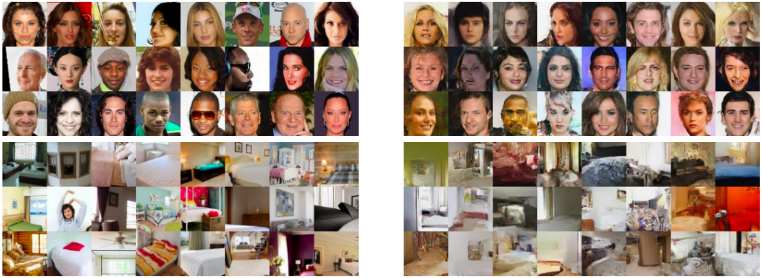


Figure 5: On the left column, examples from the dataset. On the right column, samples from the model trained on the dataset. The datasets shown in this figure are in order: CIFAR-10, Imagenet (32×32), Imagenet (64×64), CelebA, LSUN (bedroom).

Latent space interpolations via Real-NVP

Using with four validation examples $\mathbf{z}^{(1)}, \mathbf{z}^{(2)}, \mathbf{z}^{(3)}, \mathbf{z}^{(4)}$, define interpolated $\mathbf{z}$ as:

$$\mathbf{z} = cos\phi(\mathbf{z}^{(1)} cos\phi' + \mathbf{z}^{(2)} sin\phi') + sin\phi(\mathbf{z}^{(3)} cos\phi' + \mathbf{z}^{(4)} sin\phi')$$

with manifold parameterized by ϕ and ϕ'.

Masked Autoregressive Flows

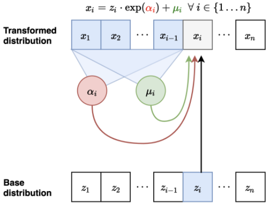
Reminder: Autoregressive models

- Take Autoregressive Model:

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

  such that
  $p(x_i|x_{i<1}) = \mathcal{N}(\mu_i(x_1, \dots, x_{i-1}), \exp\left(\alpha_i(x_1, \dots, x_{i-1})\right)^2)$,
  with μ and α are Neural network outputs.
- We have a direct estimation of likelihood in this model.
- To sample, we need to go through consecutive steps:
  - $z_i \sim \mathcal{N}(0; 1)$;
  - $x_1 = \exp(\alpha_1) z_1 + \mu_1$
  - $x_2 = \exp(\alpha_2(x_1)) z_1 + \mu_2(x_1)$
  - and so on.
- Might be stacked as Flow from Gaussians to observable space.

Masked and Inverse Autoregressive Flow (MAF/IAF)

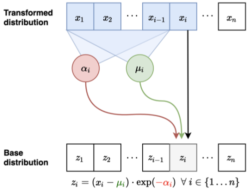


- looks similar to MADE;
- Forward mapping $z \mapsto x$:
  - $z_i \sim \mathcal{N}(0; 1)$;
  - $x_1 = \exp(\alpha_1) z_1 + \mu_1$
  - $x_2 = \exp(\alpha_2(x_1)) z_1 + \mu_2(x_1)$
  - and so on.
- sampling is sequential and slow.

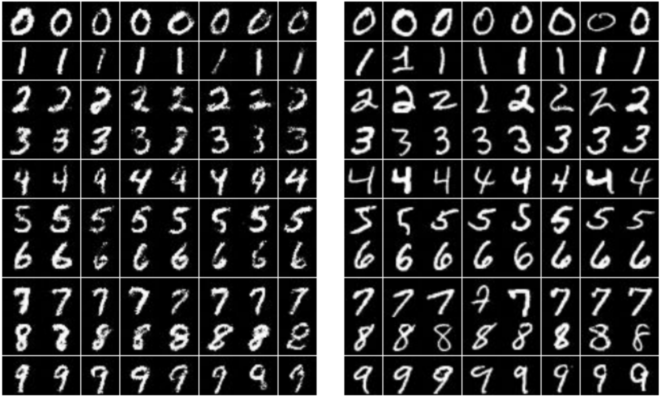
From Stanford GM lectures by S. Ermano G. Papamakarios et al. Masked Autoregressive Flow for Density Estimation

MAF: Inverse

- Inverse mapping $x \mapsto z$:
  - Compute all μ_i and α_i
  - $z = \exp(-\alpha_1) \odot (x - \mu)$
- Jacobian is lower diagonal, hence determinant can be computed efficiently.
- Likelihood evaluation is easy and parallelizable.
- Thus, the training is relatively fast.



MAF:results



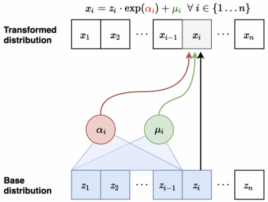
(a) Generated images

(b) Real images

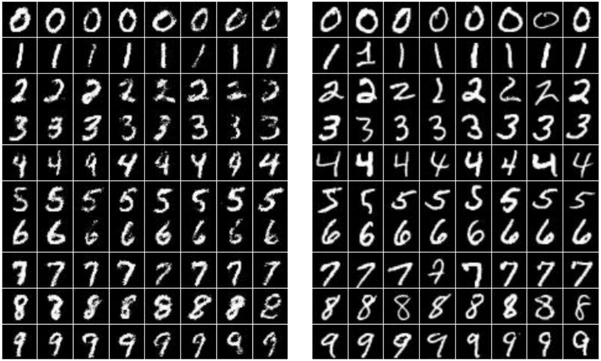
Figure 3: Class-conditional generated and real images from MNIST. Rows are different classes. Generated images are sorted by decreasing log likelihood from left to right.

Inverse Autoregressive Flow (IAF)

- Forward mapping $z \mapsto x$:
  - Sample all z_i;
  - Compute all μ_i and α_i.
  - $x = \exp(-\alpha) \odot (z - \mu)$.
- Inverse mapping $x \mapsto z$:
  - Sequential calculation.
  - $z_i = \exp(-\alpha_i(z_{<i}))(x - \mu_i(z_{<i}))$.
- Fast to sample from, slow to evaluate likelihoods of data points (train).



IAF:results



(a) Generated images (b) Real images

Figure 3: Class-conditional generated and real images from MNIST. Rows are different classes. Generated images are sorted by decreasing log likelihood from left to right.

MAF and IAF

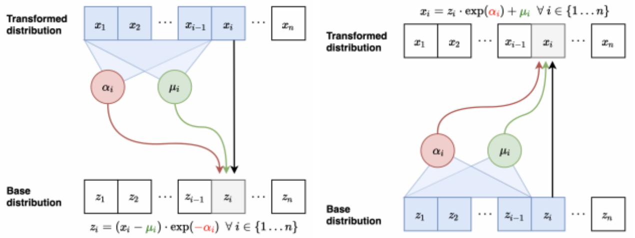


Figure: Inverse pass of MAF (**left**) vs. Forward pass of IAF (**right**)

- MAF and IAF use autoregressive transformations based on MADE building block.
- One can see that IAF forward mapping and MAF Inverse mapping are connected up to parameterisation.
- In fact, they are inverse of each other.

MAF and IAF

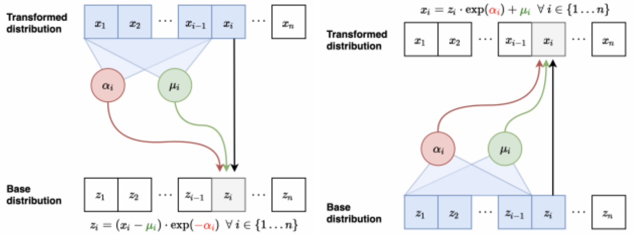
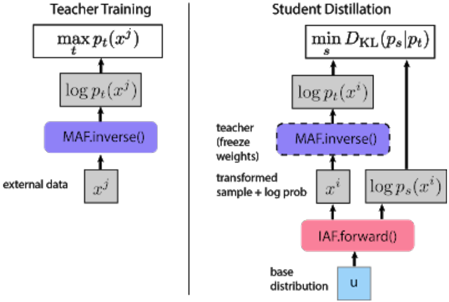


Figure: Inverse pass of MAF (**left**) vs. Forward pass of IAF (**right**)

- MAF: Fast likelihood evaluation, slow sampling - best for training based on MLE, density estimation.
- IAF: Fast sampling, slow likelihood evaluation - best for for real-time generation.

Teacher-Student Model



- Two part training with a teacher and student model.
- Teacher (MAF) trained first, than student (IAF) initialised.
- Student model cannot efficiently evaluate density for external datapoints but allows for efficient sampling.

Probability Density Distillation

- Student, s, is trained to match the teachers' distribution t using KL divergence:

$$KL(s,t) = \mathbb{E}_{x \sim s}[\log s(x) \log t(x)]$$

- Training:
  - Train teacher via MLE and obtain likelihood.
  - Train student to minimize KL divergence.
  - Use student to sample.
- Improves sampling efficiencies by a factor 100 for Wavenet.

Future developments

More results are produced this year. In general, they can be separated into following categories.

1. Det Identities	2. Coupling Blocks	3. Autoregressive	4. Unbiased Estimation
Planar NF Sylvester NF ...	NICE Real NVP Glow ...	Inverse AF Neural AF Masked AF ...	FFJORD **Residual Flows**

Jacobian

(Low rank)	(Lower triangular + structured)	(Lower triangular)	(Arbitrary)

Conclusion

- 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 tradeoffs in evaluating forward and inverse transformations