Introduction to Normalizing Flows

by

Emilio Jorge

CHALMERS

October 10, 2019

Contents

Generative models

Want to represent (generate from) distribution $p(x)$

Generative models

Want to represent (generate from) distribution $p(x)$

Some obstacles arise:

- Limiting to simple parametric distributions might not be good enough

Generative models

Want to represent (generate from) distribution $p(x)$

Some obstacles arise:

- Limiting to simple parametric distributions might not be good enough
- Nice if we have tractable likelihood

Generative models

Want to represent (generate from) distribution $p(x)$

Some obstacles arise:

- Limiting to simple parametric distributions might not be good enough
- Nice if we have tractable likelihood
  - Evaluate likelihood of new data
  - Find conditional relationships
  - Can be used to evaluate models

Generative models

Want to represent (generate from) distribution $p(x)$

Some obstacles arise:

- Limiting to simple parametric distributions might not be good enough
- Nice if we have tractable likelihood
  - Evaluate likelihood of new data
  - Find conditional relationships
  - Can be used to evaluate models

Generative models

- Generative adversarial networks
- Likelihood based methods
  - Autoregressive models
  - Variational autoencoders
  - Flow based models

Why Flows?

- Exact likelihood (no lower bound)

Why Flows?

- Exact likelihood (no lower bound)
- Efficient, both inference and synthesis (?)

Why Flows?

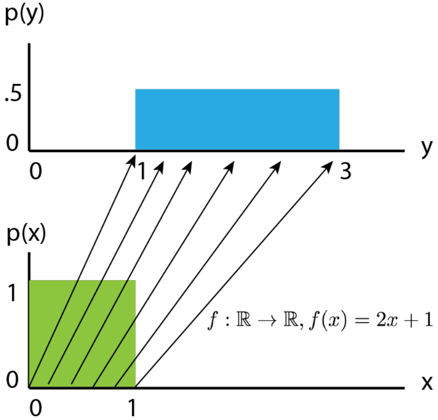
- Exact likelihood (no lower bound)
- Efficient, both inference and synthesis (?)
- Useful latent space
- Low memory usage

Variable transformation

- $X \sim \text{Uniform}(0,1)$
- $Y = f(X) = 2X + 1$

Variable transformation

- $X \sim \text{Uniform}(0, 1)$
- $Y = f(X) = 2X + 1$



(By Eric Jang)

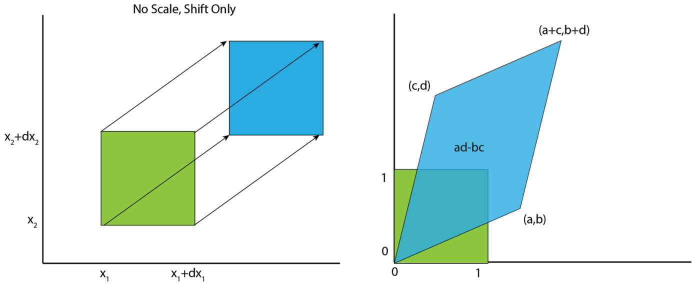
Variable transformation

- $X \sim \text{Uniform}([0, 1]x[0, 1])$

Variable transformation

- $X \sim \text{Uniform}([0,1]x[0,1])$
- Transform with

$$M = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$



Figure(By Eric Jang)

Preserve mass

First case:

$$p(x)dx = p(y)dy$$
$$p(y) = p(x)|dx/dy|$$

Second case:

Scale with absolut value of determinant of M since transformation $\mathbf{v} = \phi(\mathbf{u})$ gives

$$\int f(\mathbf{v})d\mathbf{v} = \int f(\phi(\mathbf{u}))|det\phi'(\mathbf{u})|d\mathbf{u}$$

Flows

$$z \sim q(z)$$
$$y = f(z)$$

Flows

$$z \sim q(z)$$
$$y = f(z)$$

Substitute previous equation

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

Flows

$$z \sim q(z)$$
$$y = f(z)$$

Substitute previous equation

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

$$\mathbf{z}_K = f_K \circ \cdots \circ f_1(\mathbf{z}_0), \quad \mathbf{z}_0 \sim q_0(\mathbf{z}_0)$$
$$\mathbf{z}_K \sim q_K(\mathbf{z}_K) = q_0(\mathbf{z}_0) \prod_{k=1}^{K} \left| \det \frac{\partial f_k}{\partial \mathbf{z}_{k-1}} \right|^{-1}$$

Stacking

- Want to do repeated transformations

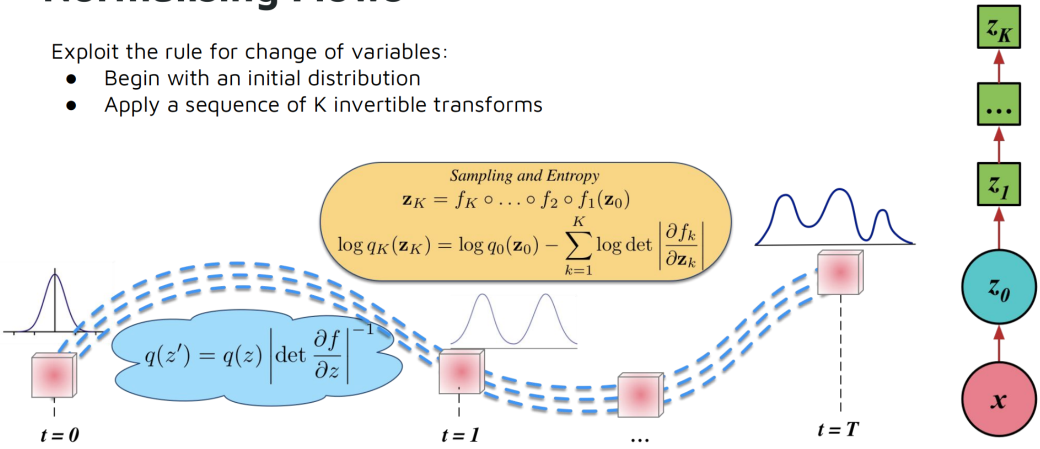
Stacking

- Want to do repeated transformations
- Add (invertible) non-linearity
- Good choice:
  - Leaky ReLU
- Bad choices:
  - ReLU
  - Sigmoid

Normalising Flows

Exploit the rule for change of variables:

- Begin with an initial distribution
- Apply a sequence of K invertible transforms



Distribution flows through a sequence of invertible transforms

Rezende and Mohamed, 2015

From Shakir Mohamed and Danilo Rezende's UAI 2017 Tutorial

How do we learn the parameters?

- Tractable likelihood function
  - Just maximize likelihood of dataset

MLE

$$z \sim q(z)$$
$$y = f(z) = \begin{bmatrix} a & b \\ c & d \end{bmatrix} z$$

MLE

$$z \sim q(z)$$

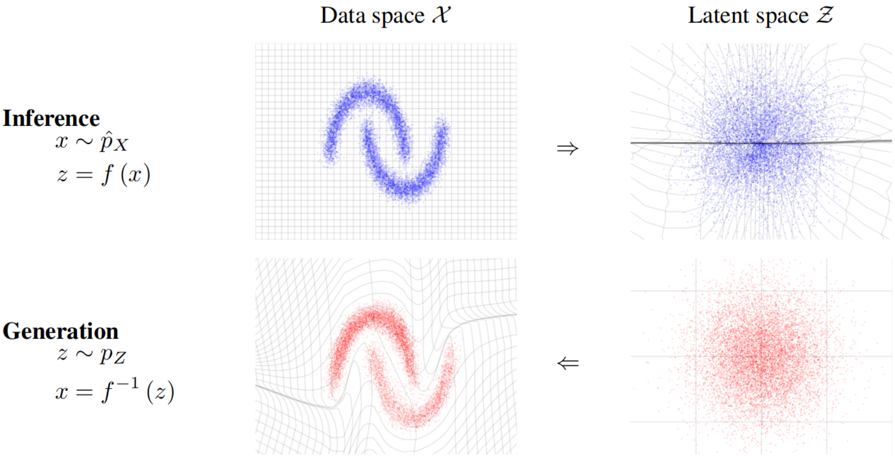
$$y = f(z) = \begin{bmatrix} a & b \\ c & d \end{bmatrix} z$$

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

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

$$= q(\begin{bmatrix} d & -b \\ -c & a \end{bmatrix} \frac{\mathbf{y}}{ad - bc}) \left| \frac{1}{ab - cd} \right|$$

Normalizing flows



Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. “Density estimation using Real NVP”. In: *ICLR 2017* (2016)

Normalizing Flows

Effient computations

Normalizing Flows

Effient computations

Need:

1. Easily invertible
2. Fast determinant of Jacobian

Normalizing Flows

Planar flow

$$
\begin{aligned}
f(\mathbf{z}) &= \mathbf{z} + \mathbf{u}h(\mathbf{w}^T\mathbf{z} + b) \\
\psi(\mathbf{z}) &= h'(\mathbf{w}^T\mathbf{z} + b)\mathbf{w} \\
\left|\det \frac{\partial f}{\partial \mathbf{z}}\right| &= \left|1 + \mathbf{u}^T\psi(\mathbf{z})\right|
\end{aligned}
$$

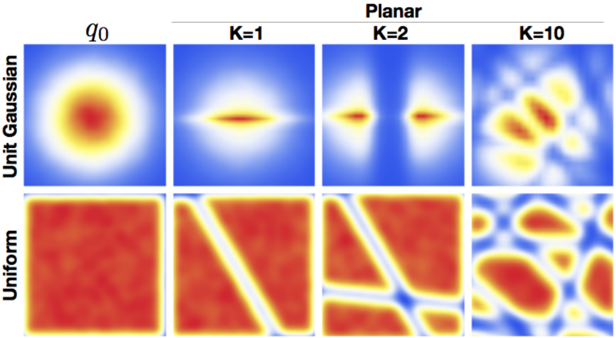
Normalizing Flows

Planar flow

$$f(\mathbf{z}) = \mathbf{z} + \mathbf{u}h(\mathbf{w}^T\mathbf{z} + b)$$

$$\psi(\mathbf{z}) = h'(\mathbf{w}^T\mathbf{z} + b)\mathbf{w}$$

$$\left|\det\frac{\partial f}{\partial \mathbf{z}}\right| = |1 + \mathbf{u}^T\psi(\mathbf{z})|$$



Danilo Jimenez Rezende and Shakir Mohamed. "Variational inference with normalizing flows". In: *ICML 2015* (2015)

Autoregressive flows

- Planar flow is not good enough
- Still want invertablility

Autoregressive flows

- Planar flow is not good enough
- Still want invertablility
- Can make dependence ”triangular”

$$y_i = f(\mathbf{z}_{1:i}), \qquad J = \frac{\partial \mathbf{y}}{\partial \mathbf{z}}$$

$$\det J = \prod_{i=1}^{d} J_{ii}$$

(Add some reordering between layers)

Autoregressive flows

Masked Autoregressive Flow (MAF)

$$
\begin{aligned}
y_1 &= \mu_1 + \sigma_1 z_1 \\
y_i &= \mu(\mathbf{y}_{1:i-1}) + \sigma(\mathbf{y}_{1:i-1}) z_i
\end{aligned}
$$

Autoregressive flows

Masked Autoregressive Flow (MAF)

$$
\begin{aligned}
y_1 &= \mu_1 + \sigma_1 z_1 \\
y_i &= \mu(\mathbf{y}_{1:i-1}) + \sigma(\mathbf{y}_{1:i-1}) z_i
\end{aligned}
$$

- Potential issues?

Autoregressive flows

Masked Autoregressive Flow (MAF)

$$\begin{aligned} y_1 &= \mu_1 + \sigma_1 z_1 \\ y_i &= \mu(\mathbf{y}_{1:i-1}) + \sigma(\mathbf{y}_{1:i-1}) z_i \end{aligned}$$

- Potential issues?
  - Not parallelisable in forward pass (generation)

Autoregressive flows

Masked Autoregressive Flow (MAF)

$$y_1 = \mu_1 + \sigma_1 z_1$$
$$y_i = \mu(\mathbf{y}_{1:i-1}) + \sigma(\mathbf{y}_{1:i-1}) z_i$$

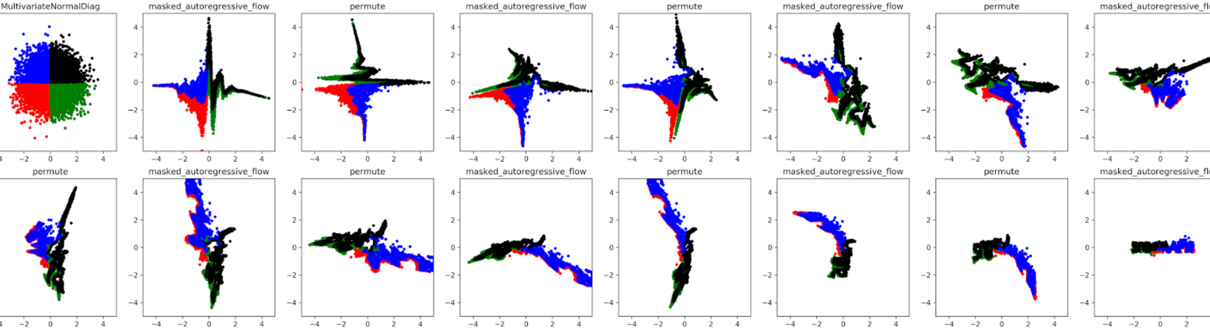
- Potential issues?
  - Not parallelisable in forward pass (generation)
  - Inverse Autoregressive Flows

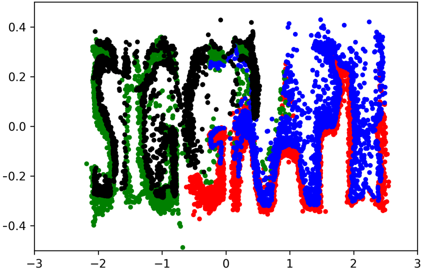
Autoregressive flows

Masked Autoregressive Flow (MAF)

$$\begin{aligned} y_1 &= \mu_1 + \sigma_1 z_1 \\ y_i &= \mu(\mathbf{y}_{1:i-1}) + \sigma(\mathbf{y}_{1:i-1}) z_i \end{aligned}$$

- Potential issues?
  - Not parallelisable in forward pass (generation)
  - Inverse Autoregressive Flows
    - Not parallelisable in backward pass (density estimation)





(By Eric Jang https://blog.evjang.com/2018/01/nf2.html)

Autoregressive models

- Pixel RNN
- WaveNet

Real NVP

Real-valued non-volume preserving transformations

$$
\begin{aligned}
\mathbf{y}_{1:k} &= \mathbf{z}_{1:k}, \\
\mathbf{y}_{k+1:d} &= \mathbf{z}_{k+1:d} \circ \sigma(\mathbf{z}_{1:k}) + \mu(\mathbf{z}_{1:k})
\end{aligned}
$$

Real NVP

Real-valued non-volume preserving transformations

$$\begin{aligned} \mathbf{y}_{1:k} &= \mathbf{z}_{1:k}, \\ \mathbf{y}_{k+1:d} &= \mathbf{z}_{k+1:d} \circ \sigma(\mathbf{z}_{1:k}) + \mu(\mathbf{z}_{1:k}) \end{aligned}$$

Keep first *k* dimensions and use them to transform other dimensions

Real NVP

Real-valued non-volume preserving transformations

$$\begin{aligned} \mathbf{y}_{1:k} &= \mathbf{z}_{1:k}, \\ \mathbf{y}_{k+1:d} &= \mathbf{z}_{k+1:d} \circ \sigma(\mathbf{z}_{1:k}) + \mu(\mathbf{z}_{1:k}) \end{aligned}$$

Keep first k dimensions and use them to transform other dimensions

Gives nice paralellization

$$\begin{aligned} \mathbf{z}_{1:k} &= \mathbf{y}_{1:k}, \\ \mathbf{z}_{k+1:d} &= (\mathbf{y}_{k+1:d} - \mu(\mathbf{y}_{1:k})) \, / \, \sigma(\mathbf{y}_{1:k}). \end{aligned}$$

σ, μ no longer need to be invertible. Can be any neural network etc.

Glow

Glow: Generative Flow with Invertible 1×1 Convolutions
Multiscale to handle images.
Use 1x1 convolution with same dimension of input and output channels instead of random permutations.

Description	Function	Reverse Function	Log-determinant
Actnorm. See Section 3.1.	$\forall i,j : \mathbf{y}_{i,j} = \mathbf{s} \odot \mathbf{x}_{i,j} + \mathbf{b}$	$\forall i,j : \mathbf{x}_{i,j} = (\mathbf{y}_{i,j} - \mathbf{b})/\mathbf{s}$	$h \cdot w \cdot \mathtt{sum}(\log \lvert\mathbf{s}\rvert)$
Invertible 1×1 convolution. $\mathbf{W} : [c \times c]$. See Section 3.2	$\forall i,j : \mathbf{y}_{i,j} = \mathbf{W}\mathbf{x}_{i,j}$	$\forall i,j : \mathbf{x}_{i,j} = \mathbf{W}^{-1}\mathbf{y}_{i,j}$	$h \cdot w \cdot \log \lvert\det(\mathbf{W})\rvert$ or $h \cdot w \cdot \mathtt{sum}(\log \lvert\mathbf{s}\rvert)$ (see eq. (10))
Affine coupling layer. See Section 3.3 and (Dinh et al., 2014)	$\mathbf{x}_a, \mathbf{x}_b = \mathtt{split}(\mathbf{x})$ $(\log \mathbf{s}, \mathbf{t}) = \mathtt{NN}(\mathbf{x}_b)$ $\mathbf{s} = \exp(\log \mathbf{s})$ $\mathbf{y}_a = \mathbf{s} \odot \mathbf{x}_a + \mathbf{t}$ $\mathbf{y}_b = \mathbf{x}_b$ $\mathbf{y} = \mathtt{concat}(\mathbf{y}_a, \mathbf{y}_b)$	$\mathbf{y}_a, \mathbf{y}_b = \mathtt{split}(\mathbf{y})$ $(\log \mathbf{s}, \mathbf{t}) = \mathtt{NN}(\mathbf{y}_b)$ $\mathbf{s} = \exp(\log \mathbf{s})$ $\mathbf{x}_a = (\mathbf{y}_a - \mathbf{t})/\mathbf{s}$ $\mathbf{x}_b = \mathbf{y}_b$ $\mathbf{x} = \mathtt{concat}(\mathbf{x}_a, \mathbf{x}_b)$	$\mathtt{sum}(\log(\lvert\mathbf{s}\rvert))$

Glow

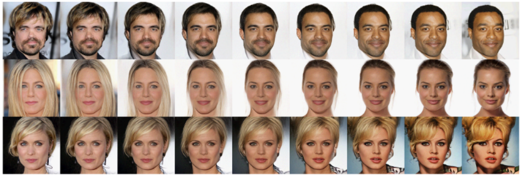
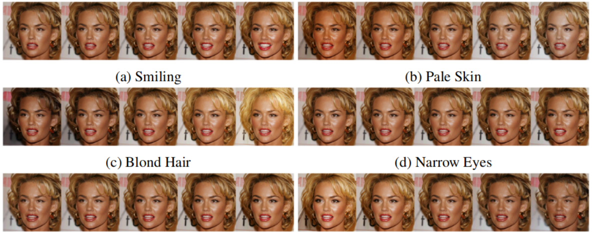


Figure 5: Linear interpolation in latent space between real images



(a) Smiling (b) Pale Skin

(c) Blond Hair (d) Narrow Eyes

(e) Young (f) Male

Figure 6: Manipulation of attributes of a face. Each row is made by interpolating the latent code of an image along a vector corresponding to the attribute, with the middle image being the original image

Comparisons

- For same parameter budget GANs have sharper images

Comparisons

- For same parameter budget GANs have sharper images
  - GANs can set large subspace of manifold to have zero probability

Comparisons

- For same parameter budget GANs have sharper images
  - GANs can set large subspace of manifold to have zero probability
- In NF, all data will have positive likelihood
  - ”The likelihood objective doesn’t allow trading off diversity for realism, so models need to be much larger to achieve realism”

Comparisons

- For same parameter budget GANs have sharper images
  - GANs can set large subspace of manifold to have zero probability
- In NF, all data will have positive likelihood
  - ”The likelihood objective doesn’t allow trading off diversity for realism, so models need to be much larger to achieve realism”
- Automatically obtain latent space

Comparisons

- Training NF model can take a long time

Comparisons

- Training NF model can take a long time
  - 40 GPUs for days/week to train GLOW for 256px images (about 2000$ AWS cost)
    - GLOW has 1200 convolutional layers

Comparisons

- Training NF model can take a long time
  - 40 GPUs for days/week to train GLOW for 256px images (about 2000$ AWS cost)
    - GLOW has 1200 convolutional layers
  - Much faster at generating data than AR models (can be orders of magnitude)
    - NF generating a 256×256 image at batch size 1 takes about 130ms on a single 1080 Ti

Comparisons

- Training NF model can take a long time
  - 40 GPUs for days/week to train GLOW for 256px images (about 2000$ AWS cost)
    - GLOW has 1200 convolutional layers
  - Much faster at generating data than AR models (can be orders of magnitude)
    - NF generating a 256×256 image at batch size 1 takes about 130ms on a single 1080 Ti
    - AR models can give better likelihoods

Comparisons

- Training NF model can take a long time
  - 40 GPUs for days/week to train GLOW for 256px images (about 2000$ AWS cost)
    - GLOW has 1200 convolutional layers
  - Much faster at generating data than AR models (can be orders of magnitude)
    - NF generating a 256×256 image at batch size 1 takes about 130ms on a single 1080 Ti
    - AR models can give better likelihoods
  - GANs hard to optimize and have difficulty assessing overfitting and generalization

Usage

- Value functions in RL
- Anomaly detection
- Generate text and music
- Variational inference

Conclusion

Normalizing flows:

- Exact likelihood
- Fast inference and sampling?
- Useful latent space
- Tensorflow has Bijector API with built in tools

- Danilo Jimenez Rezende and Shakir Mohamed. "Variational inference with normalizing flows". In: *ICML 2015* (2015).
- George Papamakarios, Theo Pavlakou, and Iain Murray. *Masked Autoregressive Flow for Density Estimation*. 2017. arXiv: 1705.07057.
- Durk P Kingma et al. "Improved variational inference with inverse autoregressive flow". In: *NIPS 2016*. 2016.
- Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. "Density estimation using Real NVP". In: *ICLR 2017* (2016).
- Durk P Kingma and Prafulla Dhariwal. "Glow: Generative flow with invertible 1x1 convolutions". In: *NIPS 2018*. 2018.
- Adam Kosiorek. *Normalizing Flows*. Apr. 2018. URL: http://akosiorek.github.io/ml/2018/04/03/norm_flows.html.
- Eric Jang. *Normalizing Flows Tutorial, Part 1: Distributions and Determinants*. Jan. 2018. URL: https://blog.evjang.com/2018/01/nf1.htmll.
- Eric Jang. *Normalizing Flows Tutorial, Part 2: Modern Normalizing Flows*. Jan. 2018. URL: https://blog.evjang.com/2018/01/nf2.html.