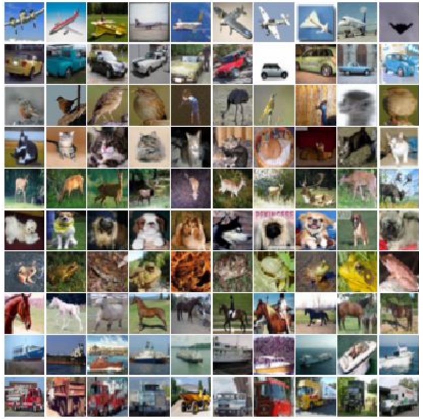
Introduction to flow-based models

Michał Stypułkowski, Maciej Zięba,
Maciej Zamorski

Generative models

Goal given unsupervised data the **model** should sample data from the same distribution.



training data from
true distribution

data sampled from
model distribution

Why bother with Generative Models?

Generating realistic images

Compressing, inpainting and reconstructing

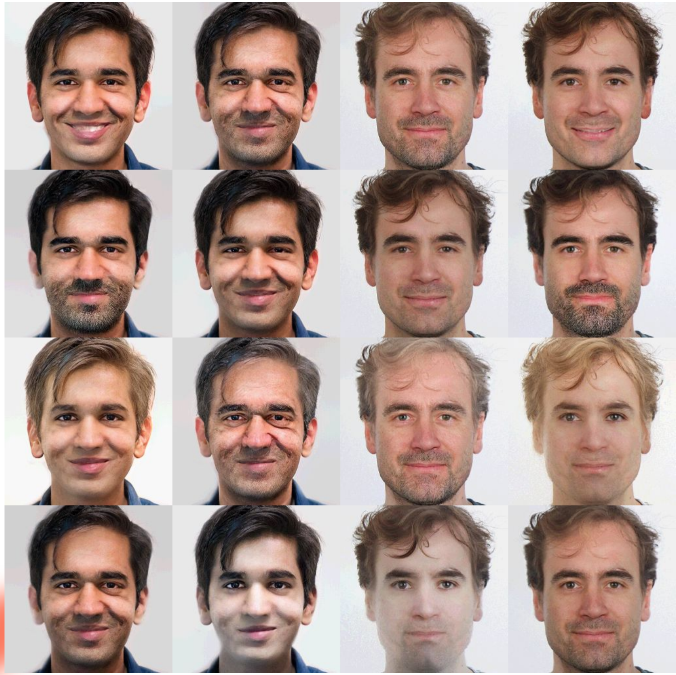
Why bother with Generative Models?

Conditional image generation of realistic samples for e.g. image search

long sleeve shirt black. spread collar. zip closure vertical zippered welt pockets

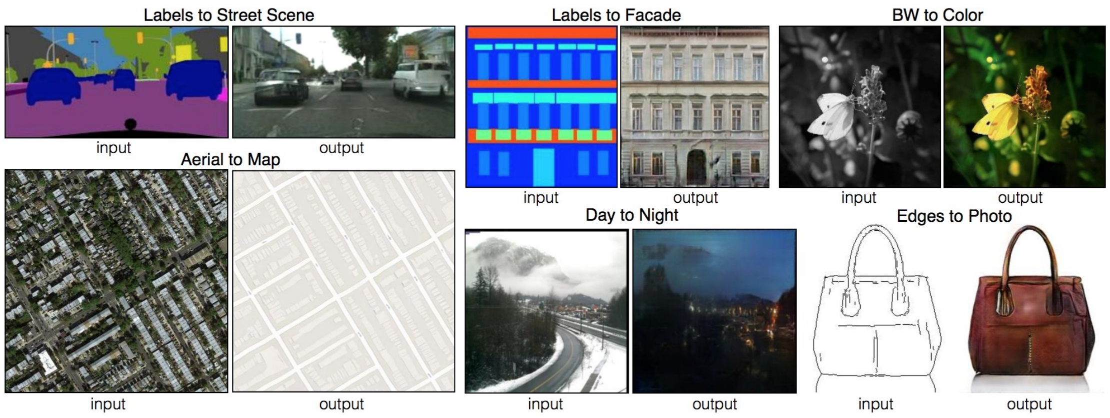
Why bother with Generative Models?

Various face manipulation tricks



Why bother with Generative Models?

image-to-image translation



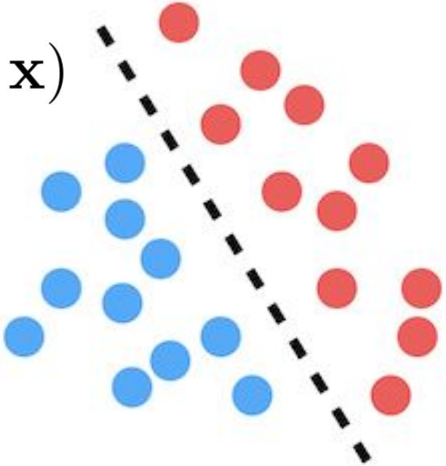
generating 3D points

Why bother with Generative Models?

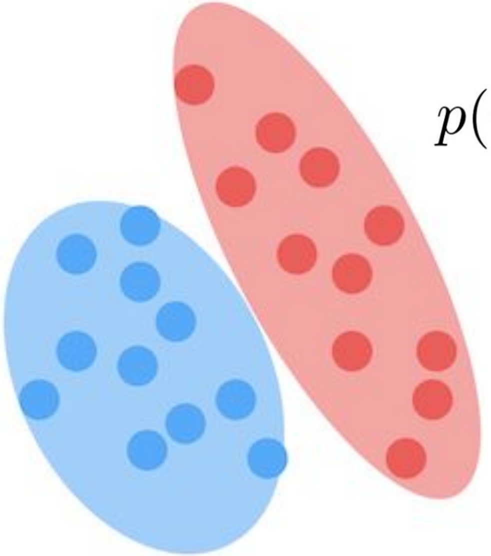
Inference using Bayesian rule

Discriminative

$$p(y|\mathbf{x}) = f(y, \mathbf{x})$$



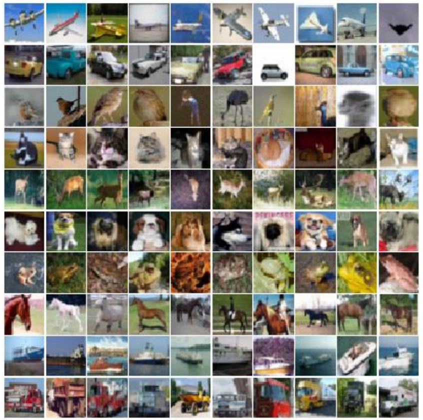
Generative



$$p(y|\mathbf{x}) = \frac{p(\mathbf{x}|y)p(y)}{p(\mathbf{x})}$$

Generative models - preliminaries

We have access to true samples from data distribution



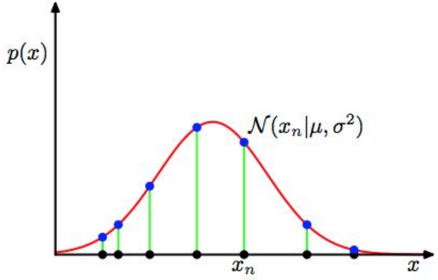
Our goal is to find some approximation of true data distribution

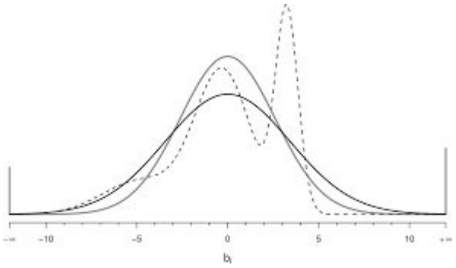
$$p(\mathbf{x})$$

Generative models - preliminaries

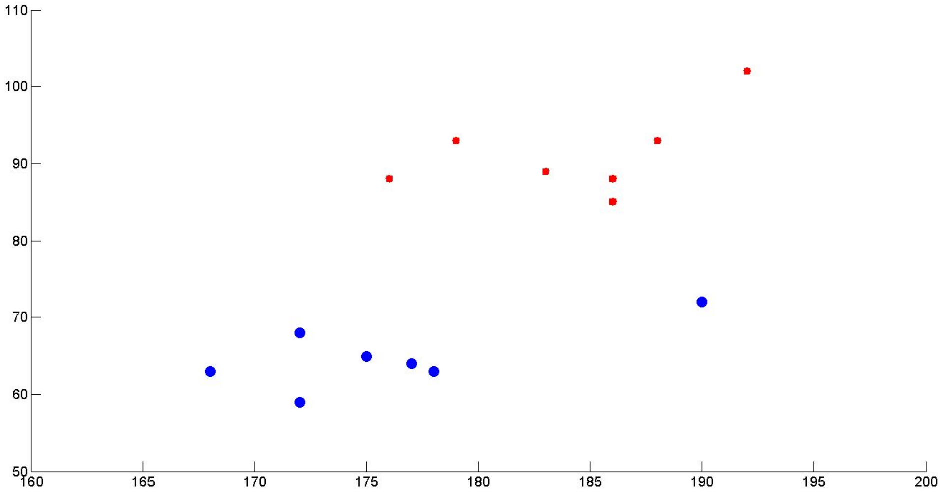
Standard approach assumes

- Select some well known distribution as true data approximation.
- Get the parameters by ML/MAP estimation.
- Sample examples from approximation





Generative models - preliminaries



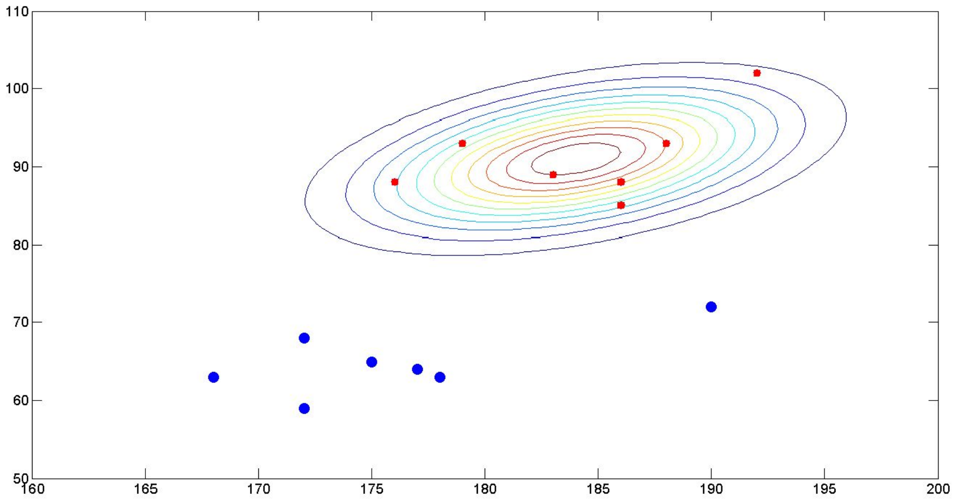
$$\boldsymbol{\mu}_0 = [176.00, \ 64.86]$$

$$\boldsymbol{\Sigma}_0 = \begin{bmatrix} 49.67 & 17.29 \\ 17.29 & 17.13 \end{bmatrix}$$

$$\boldsymbol{\mu}_1 = [184.29, \ 91.14]$$

$$\boldsymbol{\Sigma}_1 = \begin{bmatrix} 29.57 & 13.39 \\ 13.39 & 31.14 \end{bmatrix}$$

Generative models - preliminaries



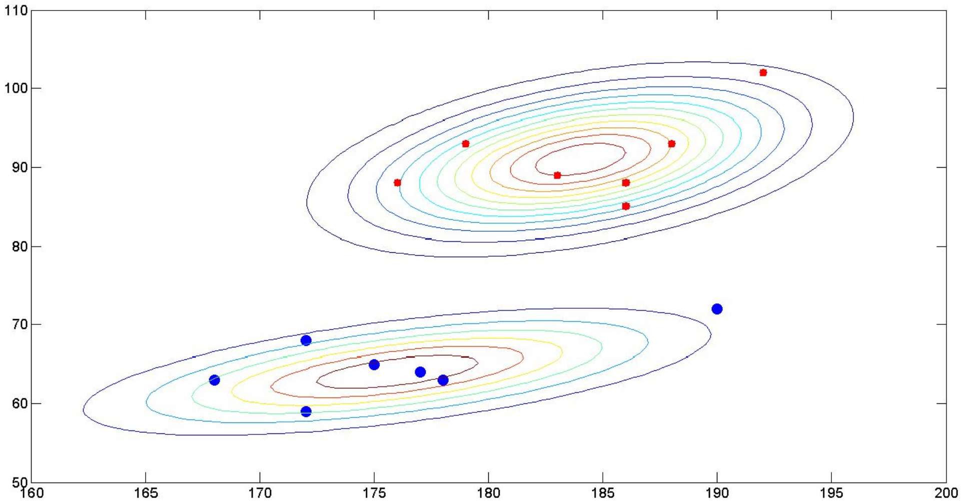
$$\boldsymbol{\mu}_0 = [176.00,\ 64.86]$$

$$\boldsymbol{\Sigma}_0 = \begin{bmatrix} 49.67 & 17.29 \\ 17.29 & 17.13 \end{bmatrix}$$

$$\boldsymbol{\mu}_1 = [184.29,\ 91.14]$$

$$\boldsymbol{\Sigma}_1 = \begin{bmatrix} 29.57 & 13.39 \\ 13.39 & 31.14 \end{bmatrix}$$

Generative models - preliminaries



$$\boldsymbol{\mu}_0 = [176.00,\ 64.86]$$

$$\boldsymbol{\Sigma}_0 = \begin{bmatrix} 49.67 & 17.29 \\ 17.29 & 17.13 \end{bmatrix}$$

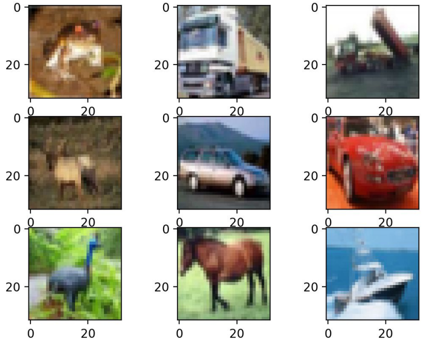
$$\boldsymbol{\mu}_1 = [184.29,\ 91.14]$$

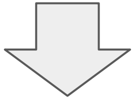
$$\boldsymbol{\Sigma}_1 = \begin{bmatrix} 29.57 & 13.39 \\ 13.39 & 31.14 \end{bmatrix}$$

Generative models - preliminaries

Standard approach assumes

- We need to choose particular rather simple distribution for approximation
- Usually, there are too many parameters to be estimated





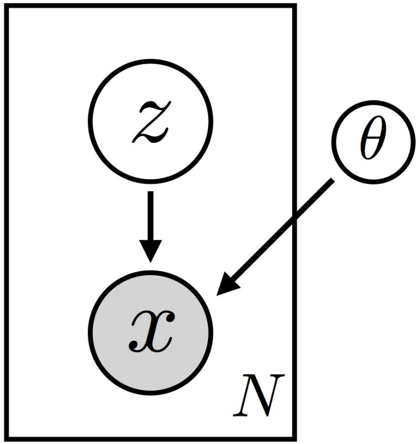
for 32x32x3 image space we need 4 720 128 parameters for covariance matrix !!!!

Generative models - preliminaries

Solution - assume some low dimensional latent representation z with simple prior

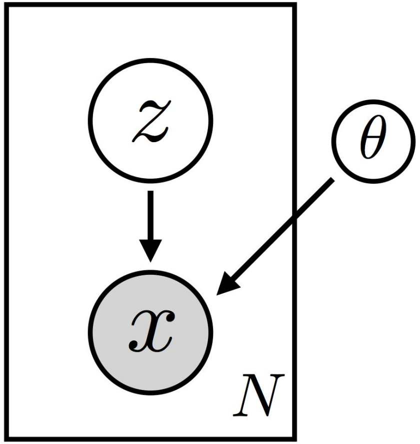
but ....

$$p(x) = \int p_\theta(x|\, z)p(z)dz$$



Generative models - preliminaries

Solution - assume some low dimensional latent representation z with simple prior

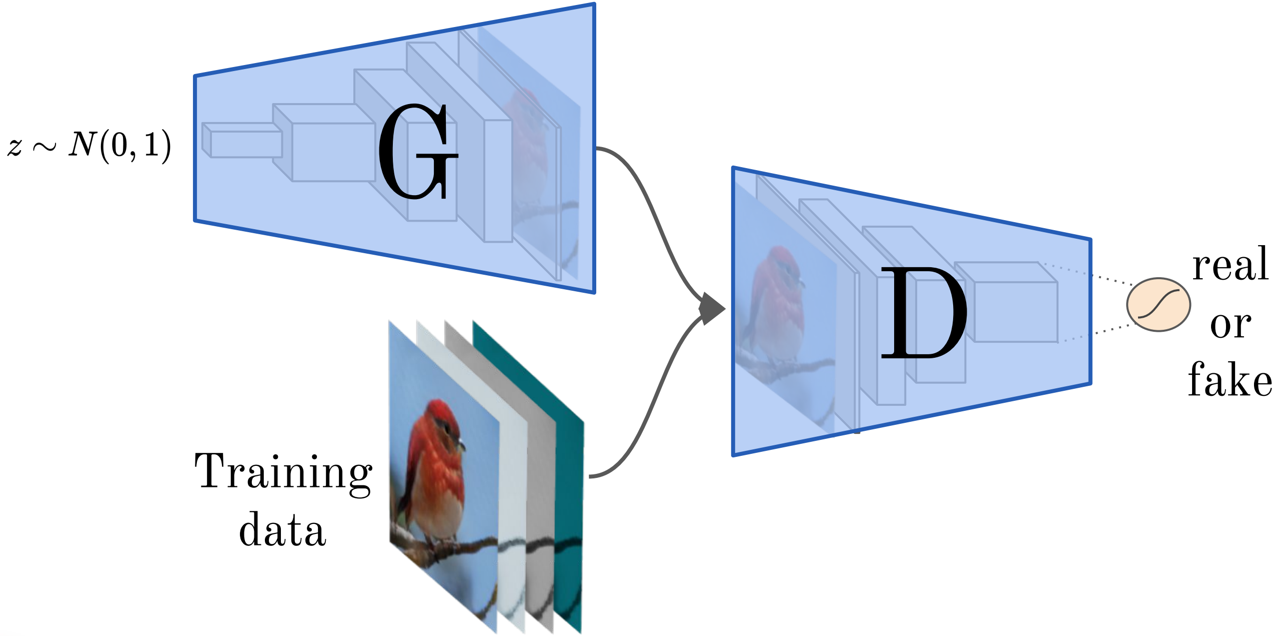


but ....

$$p(x) = \int p_\theta(x|\, z)p(z)dz$$

we can try to sample from p(x) without knowing the explicit form - GAN is some solution

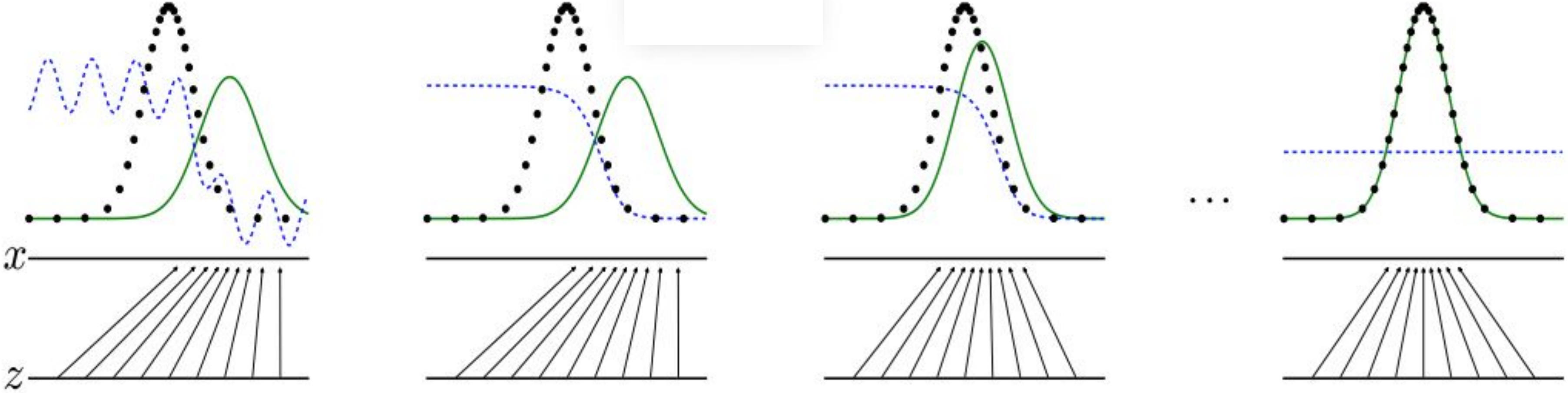
Generative Adversarial Networks (GANs)



Generator transforms random noise into a sample from an artificial distribution

Discriminator outputs likelihood in (0, 1) that image comes from the real distribution

Generative Adversarial Networks (GANs)



Blue dashed line denotes discriminator (D) scores, green solid represents true distribution, and black dots are generated samples (G); z denotes the domain of latent variable, x the domain of training data.

Source: Goodfellow, Ian, et. al. **"Generative Adversarial Nets "** arXiv preprint arXiv:1406.2661 (2014).

Generative Adversarial Networks (GANs)
Training objective

$$\min_{\theta_g} \max_{\theta_d} V(D_{\theta_d}, G_{\theta_g}) = \mathbb{E}_x \big[\log D_{\theta_d}(x) \big] + \mathbb{E}_z \big[\log(1 - D_{\theta_d}(G_{\theta_g}(z))) \big]$$

Source: Goodfellow, Ian, et. al. **"Generative Adversarial Nets "** arXiv preprint arXiv:1406.2661 (2014).

Generative Adversarial Networks (GANs)
Training objective

$$\min_{\theta_g} \max_{\theta_d} V(D_{\theta_d}, G_{\theta_g}) = \mathbb{E}_x \left[\log \boxed{D_{\theta_d}(x)} \right] + \mathbb{E}_z \left[\log(1 - \boxed{D_{\theta_d}(G_{\theta_g}(z))} \right]$$

Discriminator output for real data

Discriminator output for generated data

Source: Goodfellow, Ian, et. al. **"Generative Adversarial Nets "** arXiv preprint arXiv:1406.2661 (2014).

Generative Adversarial Networks (GANs)
Training objective

$$\min_{\theta_g} \max_{\theta_d} V(D_{\theta_d}, G_{\theta_g}) = \mathbb{E}_x \left[\log \boxed{D_{\theta_d}(x)} \right] + \mathbb{E}_z \left[\log(1 - \boxed{D_{\theta_d}(G_{\theta_g}(z))} \right]$$

Discriminator output for real data

Discriminator output for generated data

Discriminator wants to **maximize** it's values for real data and minimize its values for the generated data

Generator wants to **minimize** 1 minus the discriminators output for generated data

Source: Goodfellow, Ian, et. al. **"Generative Adversarial Nets "** arXiv preprint arXiv:1406.2661 (2014).

Generative Adversarial Networks (GANs)
Training objective

$$\min_{\theta_g} \max_{\theta_d} V(D_{\theta_d}, G_{\theta_g}) = \mathbb{E}_x \left[\log D_{\theta_d}(x) \right] + \mathbb{E}_z \left[\log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

1. Update **D** by **ascending** its gradient

$$\max_{\theta_d} \mathbb{E}_x \left[\log D_{\theta_d}(x) \right] + \mathbb{E}_z \left[\log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

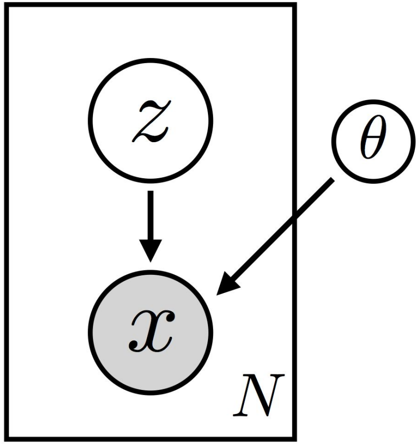
2. Update **G** by **descending** its gradient:

$$\min_{\theta_g} \mathbb{E}_z \left[\log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

Source: Goodfellow, Ian, et. al. **"Generative Adversarial Nets "** arXiv preprint arXiv:1406.2661 (2014).

Generative models - preliminaries

Solution - assume some low dimensional latent representation z with simple prior



but ....

$$p(x) = \int p_\theta(x|\, z)p(z)dz$$

other solution - try to approximate the true posterior $p_\theta(z|x)$

with the inference network $q_\phi(z\,|\,x)$

Variational Autoencoder (VAE)
Training objective

$$\log p_\theta(x) = \mathbb{E}_{z \sim q_\phi(z|x)}\Big[\log p_\theta(x)\Big] \quad (p_\theta(x) \text{ is independent of } z)$$

Variational Autoencoder (VAE)
Training objective

$$
\begin{aligned}
\log p_\theta(x) &= \mathbb{E}_{z \sim q_\phi(z|x)}\Big[\log p_\theta(x)\Big] \quad (p_\theta(x) \text{ is independent of } z) \\
&= \mathbb{E}_z\left[\log \frac{p_\theta(x \mid z)p(z)}{p(z \mid x)}\right] \quad (\text{Bayes' Rule})
\end{aligned}
$$

Variational Autoencoder (VAE)
Training objective

$$\log p_\theta(x) = \mathbb{E}_{z\sim q_\phi(z|x)}\Big[\log p_\theta(x)\Big] \quad (p_\theta(x) \text{ is independent of } z)$$

$$= \mathbb{E}_z\left[\log \frac{p_\theta(x \mid z)p(z)}{p(z \mid x)}\right] \quad (\text{Bayes' Rule})$$

$$= \mathbb{E}_z\left[\log \frac{p_\theta(x \mid z)p(z)}{p(z \mid x)} \frac{q_\phi(z \mid x)}{q_\phi(z \mid x)}\right] \quad (\text{Multiply by } 1)$$

Variational Autoencoder (VAE)
Training objective

$$
\begin{aligned}
\log p_\theta(x) &= \mathbb{E}_{z \sim q_\phi(z|x)}\Big[\log p_\theta(x)\Big] \quad (p_\theta(x) \text{ is independent of } z) \\
&= \mathbb{E}_z\left[\log \frac{p_\theta(x \mid z)p(z)}{p(z \mid x)}\right] \quad \text{(Bayes' Rule)} \\
&= \mathbb{E}_z\left[\log \frac{p_\theta(x \mid z)p(z)}{p(z \mid x)} \frac{q_\phi(z \mid x)}{q_\phi(z \mid x)}\right] \quad \text{(Multiply by 1)} \\
&= \mathbb{E}_z\Big[\log p_{\theta(x \mid z)}\Big] - \mathbb{E}_z\left[\log \frac{q_\phi(z \mid x)}{p(z)}\right] + \mathbb{E}_z\left[\log \frac{q_\phi(z \mid x)}{p(z \mid x)}\right] \quad \text{(Logarithms)}
\end{aligned}
$$

Variational Autoencoder (VAE)
Training objective

$$
\begin{aligned}
\log p_\theta(x) &= \mathbb{E}_{z\sim q_\phi(z|x)}\Big[\log p_\theta(x)\Big] \quad (p_\theta(x) \text{ is independent of } z) \\
&= \mathbb{E}_z\left[\log \frac{p_\theta(x \mid z)p(z)}{p(z \mid x)}\right] \quad \text{(Bayes' Rule)} \\
&= \mathbb{E}_z\left[\log \frac{p_\theta(x \mid z)p(z)}{p(z \mid x)} \frac{q_\phi(z \mid x)}{q_\phi(z \mid x)}\right] \quad \text{(Multiply by 1)} \\
&= \mathbb{E}_z\left[\log p_{\theta(x \mid z)}\right] - \mathbb{E}_z\left[\log \frac{q_\phi(z \mid x)}{p(z)}\right] + \mathbb{E}_z\left[\log \frac{q_\phi(z \mid x)}{p(z \mid x)}\right] \quad \text{(Logarithms)} \\
&= \mathbb{E}_z\left[\log p_{\theta(x \mid z)}\right] - D_{KL}(q_\phi(z \mid x) \,||\, p(z)) + D_{KL}(q_\phi(z \mid x) \,||\, p(z|x))
\end{aligned}
$$

Variational Autoencoder (VAE)
Training objective

$$
\begin{aligned}
\log p_\theta(x) &= \mathbb{E}_{z\sim q_\phi(z|x)}\Big[\log p_\theta(x)\Big] \quad (p_\theta(x) \text{ is independent of } z) \\
&= \mathbb{E}_z\left[\log \frac{p_\theta(x\,|\,z)p(z)}{p(z\,|\,x)}\right] \quad \text{(Bayes' Rule)} \\
&= \mathbb{E}_z\left[\log \frac{p_\theta(x\,|\,z)p(z)}{p(z\,|\,x)} \frac{q_\phi(z\,|\,x)}{q_\phi(z\,|\,x)}\right] \quad \text{(Multiply by 1)} \\
&= \mathbb{E}_z\Big[\log p_{\theta(x\,|\,z)}\Big] - \mathbb{E}_z\left[\log \frac{q_\phi(z\,|\,x)}{p(z)}\right] + \mathbb{E}_z\left[\log \frac{q_\phi(z\,|\,x)}{p(z\,|\,x)}\right] \quad \text{(Logarithms)} \\
&= \underbrace{\mathbb{E}_z\Big[\log p_{\theta(x\,|\,z)}\Big] - D_{KL}(q_\phi(z\,|\,x)\,||\,p(z))}_{\mathcal{L}(x,\theta,\phi)} + \underbrace{D_{KL}(q_\phi(z\,|\,x)\,||\,p(z|x))}_{\geq 0}
\end{aligned}
$$

Variational Autoencoder (VAE)
Training objective

$$
\begin{aligned}
\log p_\theta(x) &= \mathbb{E}_{z\sim q_\phi(z|x)}\Big[\log p_\theta(x)\Big] \quad (p_\theta(x) \text{ is independent of } z) \\
&= \mathbb{E}_z\left[\log \frac{p_\theta(x \mid z)p(z)}{p(z \mid x)}\right] \quad (\text{Bayes' Rule}) \\
&= \mathbb{E}_z\left[\log \frac{p_\theta(x \mid z)p(z)}{p(z \mid x)} \frac{q_\phi(z \mid x)}{q_\phi(z \mid x)}\right] \quad (\text{Multiply by } 1) \\
&= \mathbb{E}_z\left[\log p_{\theta(x \mid z)}\right] - \mathbb{E}_z\left[\log \frac{q_\phi(z \mid x)}{p(z)}\right] + \mathbb{E}_z\left[\log \frac{q_\phi(z \mid x)}{p(z \mid x)}\right] \quad (\text{Logarithms}) \\
&= \underbrace{\mathbb{E}_z\left[\log p_{\theta(x \mid z)}\right] - D_{KL}(q_\phi(z \mid x) \,||\, p(z))}_{\mathcal{L}(x,\theta,\phi)} + \underbrace{D_{KL}(q_\phi(z \mid x) \,||\, p(z|x))}_{\geq 0}
\end{aligned}
$$

Evidence Lower Bound (**ELBO**)

$$\log p_\theta(x) \geq \mathcal{L}(x, \theta, \phi)$$

MLE training objective

$$\theta^*, \phi* = \arg\max_{\theta,\phi} \sum_{i=1}^{N} \mathcal{L}(x_i, \theta, \phi)$$

Variational Autoencoder (VAE)
Training objective

$$\log p_\theta(x) = \underbrace{\mathbb{E}_z\big[\log p_{\theta(x\,|\,z)}\big] - D_{KL}(q_\phi(z\,|\,x)\,||\,p(z))}_{\mathcal{L}(x,\theta,\phi)} + \boxed{\underbrace{D_{KL}(q_\phi(z\,|\,x)\,||\,p(z|x))}_{\geq 0}}$$

The KL divergence of the approximation and true posterior distribution,
Is greater or equal to zero. We can't optimize directly.

$$\log p_\theta(x) \geq \mathcal{L}(x_i, \theta, \phi)$$

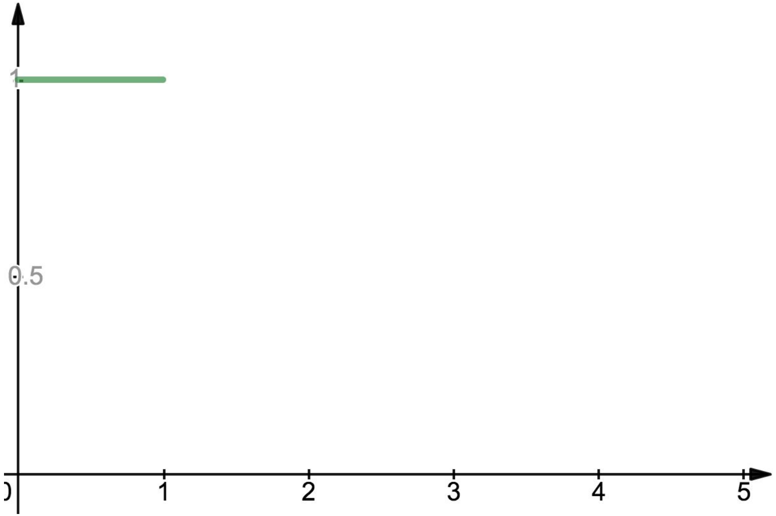
$$\log p_\theta(x) \geq \mathbb{E}_z\big[\log p_{\theta(x\,|\,z)}\big] - \ D_{KL}(q_\phi(z\,|\,x)\,||\,p(z))$$

Change of Variables rule

Example 1

Consider density function for uniform distribution:

$$p_X(x) = \begin{cases} 1 & 0 \leq x \leq 1 \\ 0 & otherwise \end{cases}$$



We create a new random variable using the following transformation:

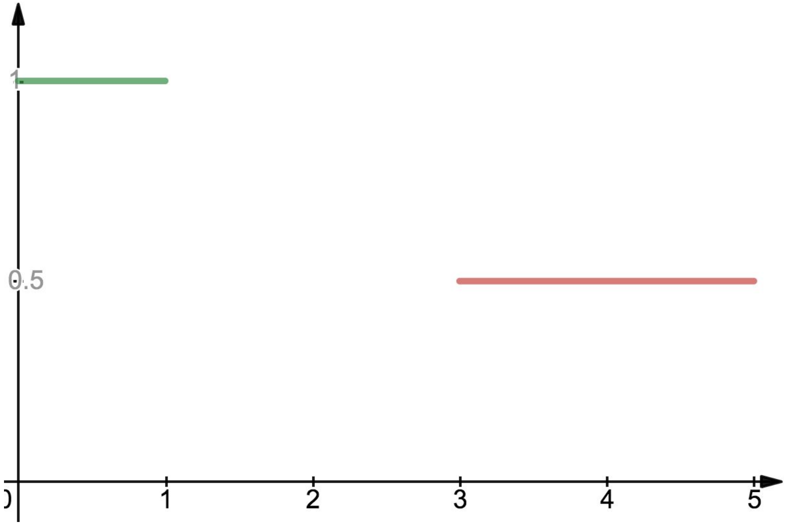
$$Y = f(X) = 2 \cdot X + 3$$

What is density function for a new variable Y?

Example 1

The new density function can be defined as:

$$p_Y(y) = \begin{cases} 0.5 & 3 \leq y \leq 5 \\ 0 & otherwise \end{cases}$$



Thanks to change of variable formula:

$$p_Y(y) = p_X(f^{-1}(y)) \left| \frac{df^{-1}(y)}{dy} \right|$$

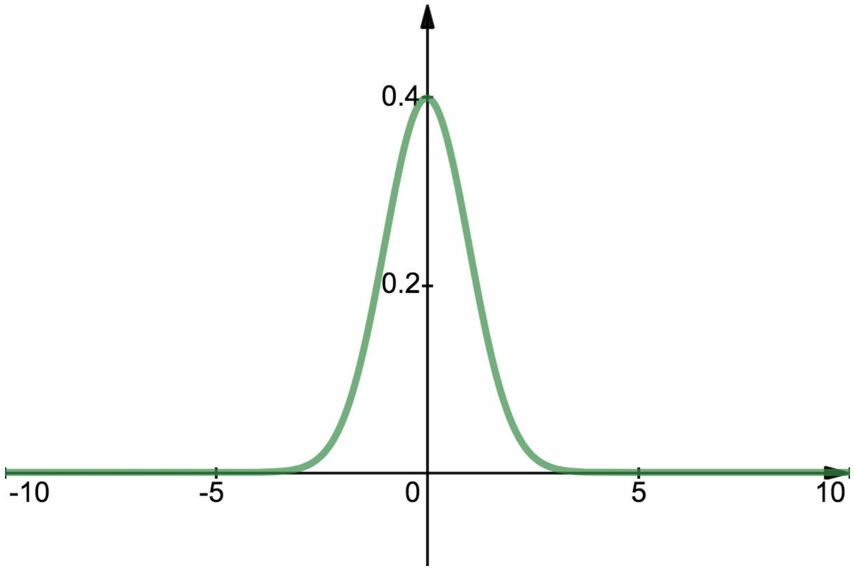
where:

$$f^{-1}(Y) = \frac{Y-3}{2}$$

Example 2

Let's consider N(0,1)

$$p_X(x) = \frac{1}{\sqrt{2\pi}} \exp\{-\frac{1}{2}x^2\}$$



and the same transformation:

$$Y = f(X) = 2 \cdot X + 3$$

$$f^{-1}(Y) = \frac{Y-3}{2}$$

Example 2

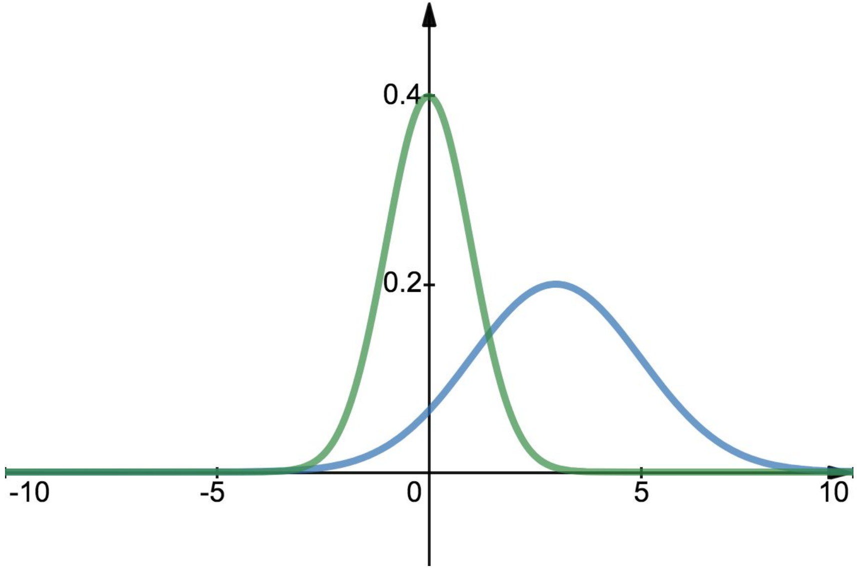
Thanks to change of variable formula:

$$p_Y(y) = p_X(f^{-1}(y)) \left| \frac{df^{-1}(y)}{dy} \right|$$

we have:

$$p_Y(y) = p_X(\frac{y-3}{2}) \cdot \frac{1}{2}$$

$$p_Y(y) = \frac{1}{\sqrt{2\pi \cdot 2^2}} \exp\{-\frac{1}{2}\frac{(y-3)^2}{2^2}\} = \mathcal{N}(3, 2^2)$$



Example 2

Assuming the normal distribution:

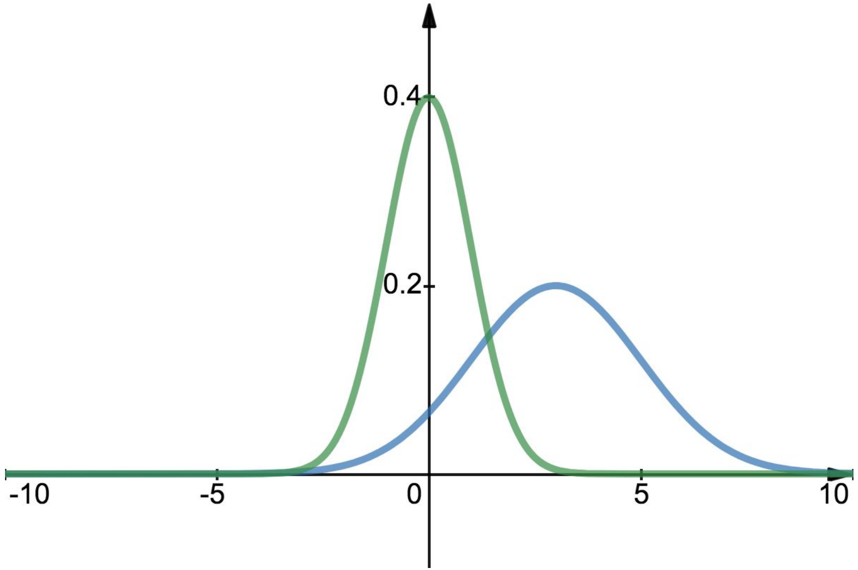
$$p_X(x) = \mathcal{N}(\mu, \sigma^2)$$

and the variable transformation:

$$Y = a \cdot X + b$$

we have:

$$p_Y(y) = \mathcal{N}(a \cdot \mu + b, (a \cdot \sigma)^2)$$



Example 2

Assuming the normal distribution:

$$p_X(x) = \mathcal{N}(\mu, \sigma^2)$$

and the variable transformation:

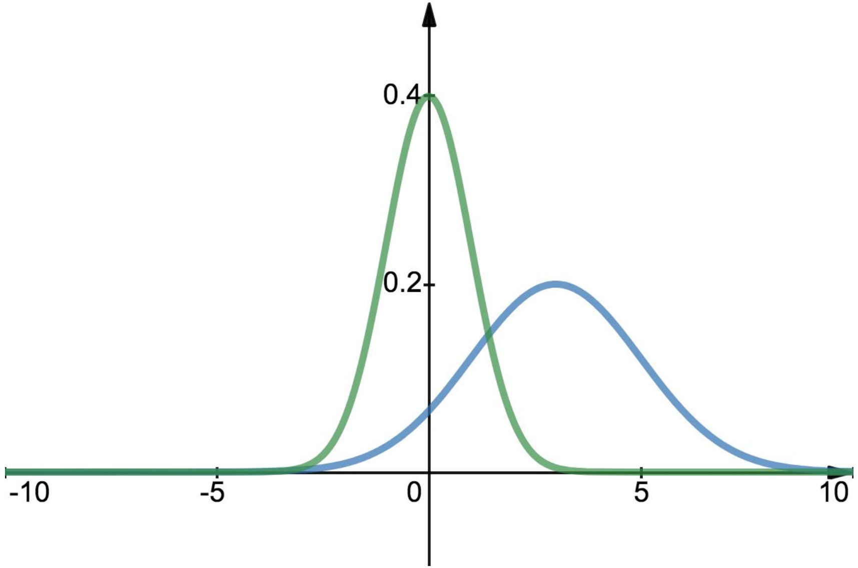
$$Y = a \cdot X + b$$

we have:

shifting

$$p_Y(y) = \mathcal{N}(\boxed{a \cdot \mu} + \boxed{b}, \boxed{(a \cdot \sigma)^2})$$

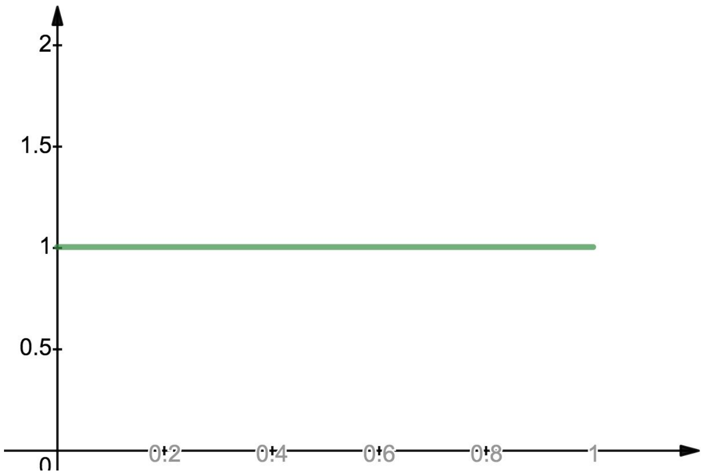
scaling scaling



Example 3

we consider again:

$$p_X(x) = \begin{cases} 1 & 0 \leq x \leq 1 \\ 0 & otherwise \end{cases}$$



but more complex transformation

$$Y = f(X) = \sqrt{X}$$

$$f^{-1}(Y) = Y^2$$

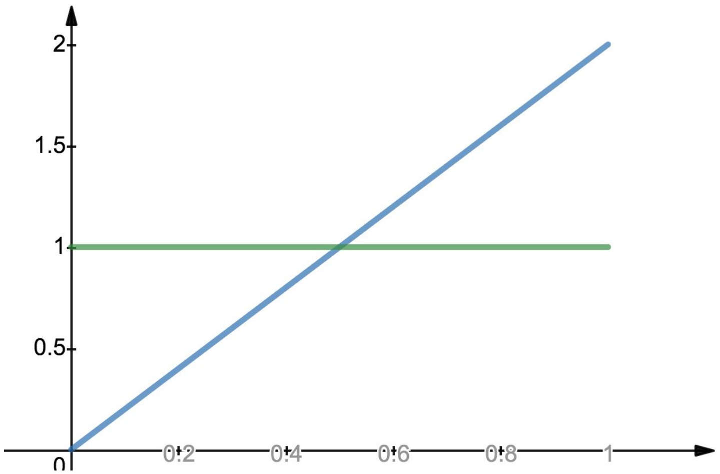
Example 3

Recalling change of variable formula:

$$p_Y(y) = p_X(f^{-1}(y)) \left| \frac{df^{-1}(y)}{dy} \right|$$

we have:

$$p(y) = \begin{cases} 2y & 0 <= y <= 1 \\ 0 & otherwise \end{cases}$$



Multidimensional case

for multidimensional case:

$$p_Y(y) = p_X(f^{-1}(y)) \left| \frac{df^{-1}(y)}{dy} \right|$$

becomes:

$$p_{\mathbf{Y}}(\mathbf{y}) = p_{\mathbf{X}}(\mathbf{f}^{-1}(\mathbf{y})) | \det \mathbf{J}_{\mathbf{f}^{-1}} |$$

where:

$$\mathbf{J}_{\mathbf{f}^{-1}} = \begin{bmatrix} \frac{\partial f_1^{-1}}{\partial y_1} & \cdots & \frac{\partial f_1^{-1}}{\partial y_D} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_D^{-1}}{\partial y_1} & \cdots & \frac{\partial f_D^{-1}}{\partial y_D} \end{bmatrix}$$

Multidimensional case

It also works for:

$$p_X(x) = p_Y(f(x))\left|\frac{df(x)}{dx}\right|$$

where:

$$p_{\mathbf{X}}(\mathbf{x}) = p_{\mathbf{Y}}(\mathbf{f}(\mathbf{x}))\left|\det \mathbf{J_f}\right|$$

and:

$$\mathbf{J_f} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_D} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_D}{\partial x_1} & \cdots & \frac{\partial f_D}{\partial x_D} \end{bmatrix}$$

Multidimensional case -example

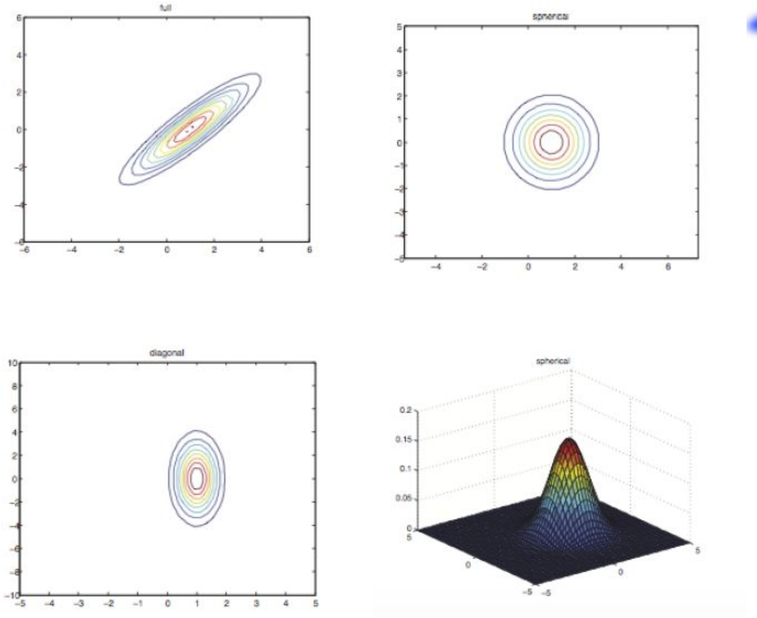
we consider the multivariate Gaussian:

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{\frac{D}{2}}(\det \boldsymbol{\Sigma})^{\frac{1}{2}}} \exp\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\}$$

and the following transformation:

$$\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{b}$$

$$\mathbf{x} = \mathbf{A}^{-1}(\mathbf{y} - \mathbf{b})$$



Multidimensional case -example

taking into account:

$$p_{\mathbf{Y}}(\mathbf{y}) = p_{\mathbf{X}}(\boxed{\mathbf{f}^{-1}(\mathbf{y})})\boxed{|\det \mathbf{J}_{\mathbf{f}^{-1}}|}$$

we have:

$$p(\mathbf{y}) = \frac{|\det \mathbf{A}^{-1}|}{(2\pi)^{\frac{D}{2}}(\det \mathbf{\Sigma})^{\frac{1}{2}}} \exp\{-\frac{1}{2}(\mathbf{A}^{-1}(\mathbf{y}-\mathbf{b})-\boldsymbol{\mu})^{\mathrm{T}}\mathbf{\Sigma}^{-1}(\mathbf{A}^{-1}(\mathbf{y}-\mathbf{b})-\boldsymbol{\mu})\}$$

after transformations we have:

$$p(\mathbf{y}) = \frac{1}{(2\pi)^{\frac{D}{2}}(\det(\mathbf{A}\mathbf{\Sigma}\mathbf{A}^{\mathrm{T}}))^{\frac{1}{2}}} \exp\{-\frac{1}{2}(\mathbf{y}-\mathbf{A}\boldsymbol{\mu})^{\mathrm{T}}(\mathbf{A}\mathbf{\Sigma}\mathbf{A}^{\mathrm{T}})^{-1}(\mathbf{y}-\mathbf{A}\boldsymbol{\mu})\}$$

$$\mathcal{N}(\mathbf{A}\boldsymbol{\mu}, \mathbf{A}\mathbf{\Sigma}\mathbf{A}^{\mathrm{T}})$$

Multidimensional case

$$p_{\mathbf{X}}(\mathbf{x}) = p_{\mathbf{Y}}(\mathbf{f}(\mathbf{x}))\,|\det \mathbf{J_f}|$$

Challange for large dimensions

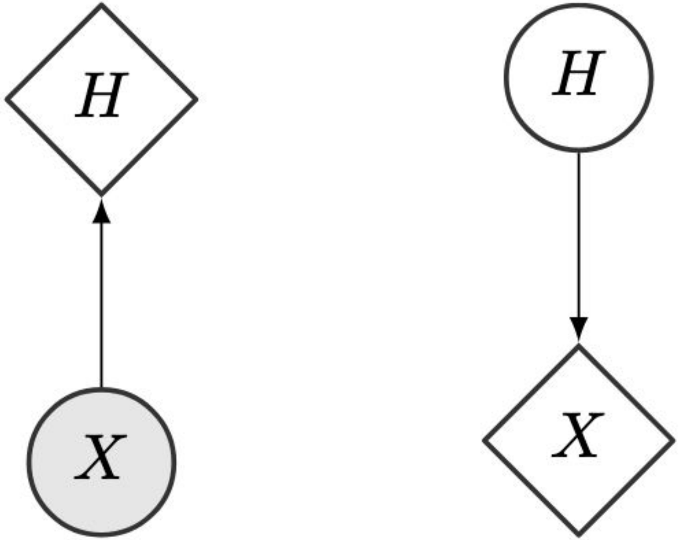
need of invertible transformation

$$\mathbf{J_f} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_D} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_D}{\partial x_1} & \cdots & \frac{\partial f_D}{\partial x_D} \end{bmatrix}$$

NICE:
Non-linear Independent Components Estimation

NICE: Non-linear Independent Components Estimation

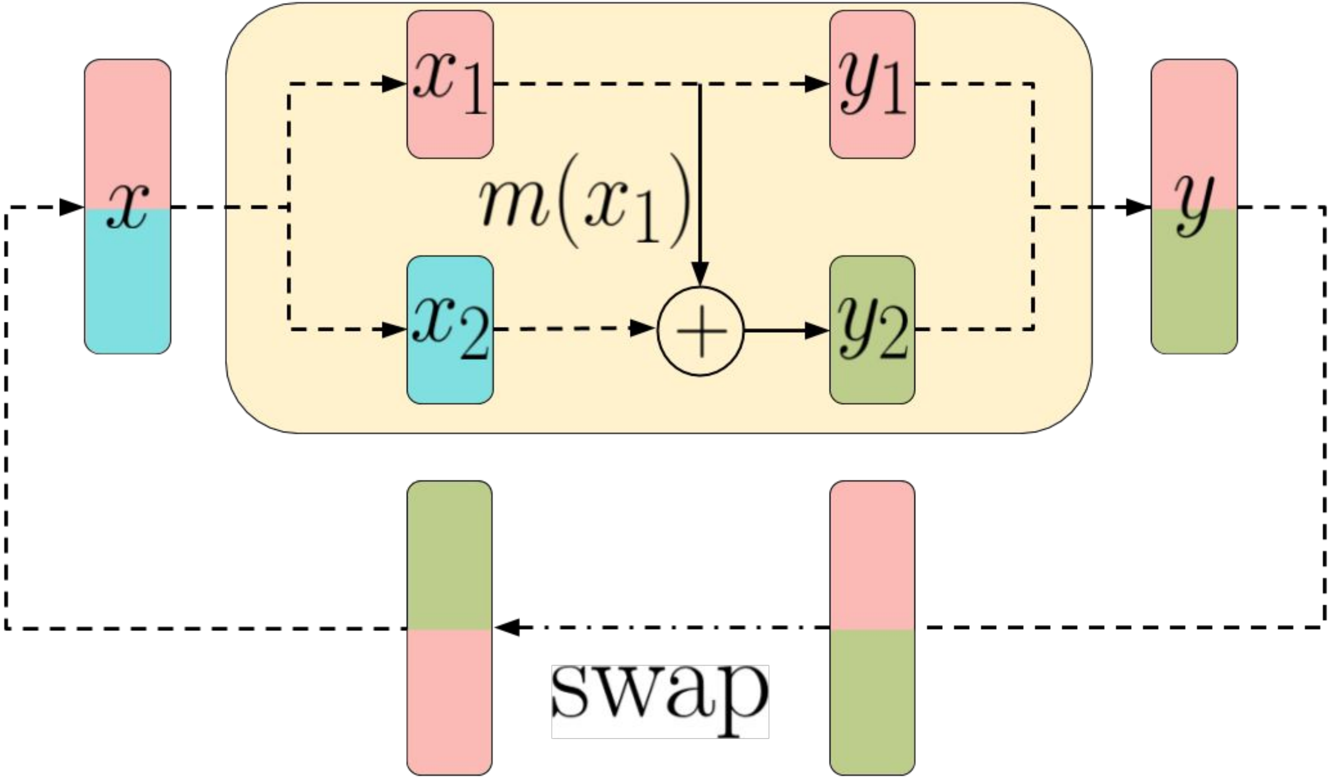
- Probabilistic model using the change of variables theorem
- Find **invertible** transformation between easy and complex probabilistic distributions



(a) Inference (b) Sampling

Source: Dinh, Laurent, et. al. **"NICE: Non-linear Independent Components Estimation "** ICLR 2015 (Workshop)

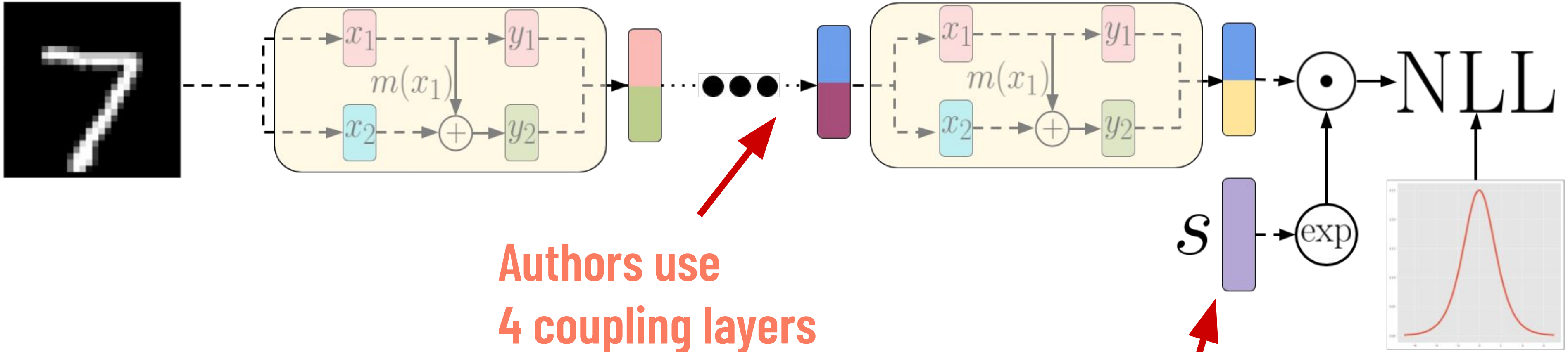
NICE - main idea



Inference transformation

NICE - main idea

Inference transformation



Authors use
4 coupling layers

Scaling vector

NICE: Non-linear Independent Components Estimation

In other words, we can define the mapping as follows:

$$\begin{cases} y_1 = x_1 \\ y_2 = x_2 + m(x_1) \end{cases}$$

NICE: Non-linear Independent Components Estimation

Arbitrary partition,
not necessarily splitting in half

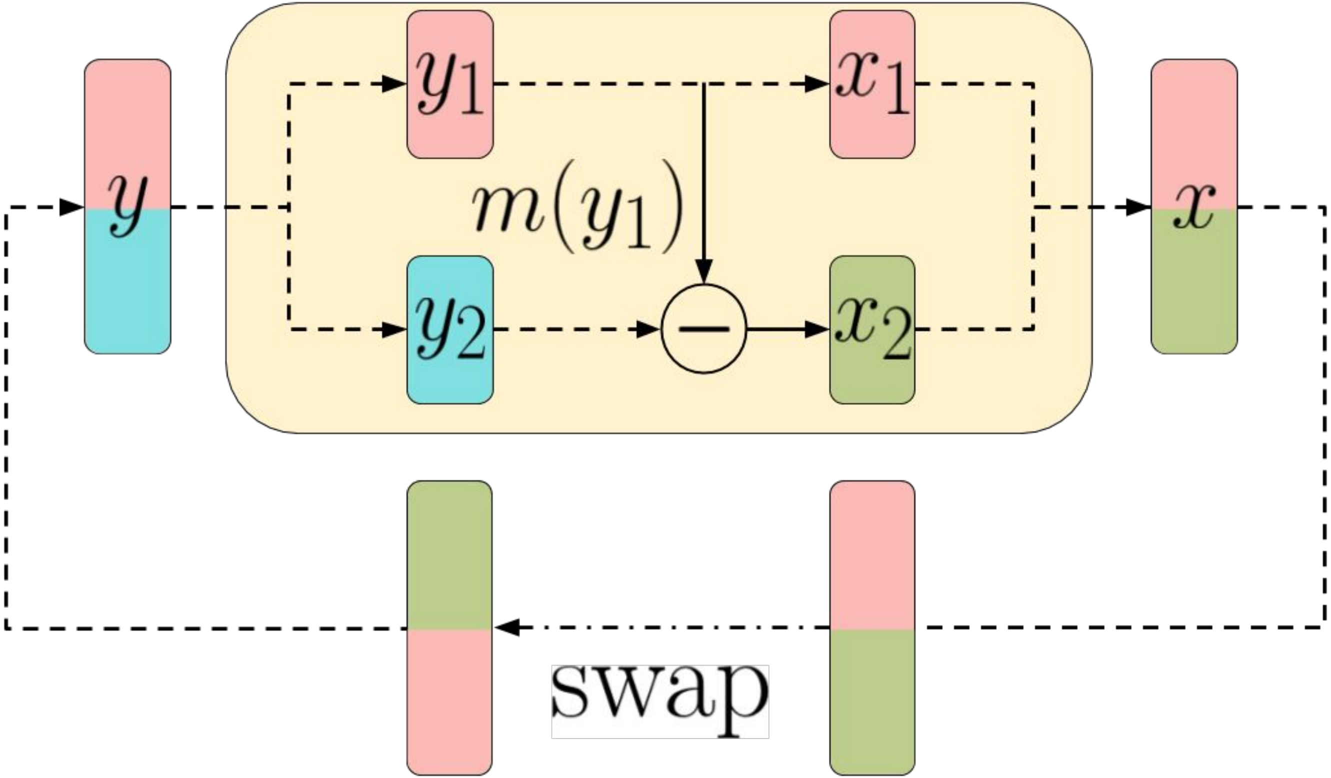
$$\begin{cases} y_1 = x_1 \\ y_2 = x_2 + m(x_1) \end{cases}$$

NICE: Non-linear Independent Components Estimation

Inverting the mapping is trivial

$$\begin{cases} x_1 = y_1 \\ x_2 = y_2 - m(x_1) \end{cases}$$

NICE: Non-linear Independent Components Estimation



Inverse mapping (sampling)

NICE: Non-linear Independent Components Estimation

Model is trained by maximizing following likelihood

$$\log p_X(x) = \log p_H(f(x)) + \log \left| \det \frac{\partial f(x)}{\partial x} \right|$$

Assuming isotropic probability distribution we obtain

$$\log p_X(x) = \sum_{d=1}^{D} \log p_{H_d}(f_d(x)) + \log \left| \det \frac{\partial f(x)}{\partial x} \right|$$

NICE - main idea

Transformation Jacobian

$$\frac{\partial y}{\partial x} = \begin{bmatrix} I & 0 \\ \frac{\partial y_2}{\partial x_1} & \frac{\partial y_2}{\partial x_2} \end{bmatrix}$$

NICE - main idea

Transformation Jacobian

$$\frac{\partial y}{\partial x} = \begin{bmatrix} I & 0 \\ \frac{\partial y_2}{\partial x_1} & \frac{\partial y_2}{\partial x_2} \end{bmatrix}$$

Matrices

NICE: Non-linear Independent Components Estimation

Linear algebra review - computing 2x2 determinant

$$\det \begin{bmatrix} a & b \\ c & d \end{bmatrix} = ad - bc$$

NICE: Non-linear Independent Components Estimation

Transformation of Jacobian

$$\det \frac{\partial y}{\partial x} = \det \begin{bmatrix} I & 0 \\ \frac{\partial y_2}{\partial x_1} & \frac{\partial y_2}{\partial x_2} \end{bmatrix}$$

NICE: Non-linear Independent Components Estimation

Transformation of Jacobian

$$\det \begin{bmatrix} I & 0 \\ \frac{\partial y_2}{\partial x_1} & \frac{\partial y_2}{\partial x_2} \end{bmatrix} = \det \frac{\partial y_2}{\partial x_2}$$

NICE: Non-linear Independent Components Estimation

$$\det \frac{\partial y}{\partial x} = \det \frac{\partial y_2}{\partial x_2}$$

NICE: Non-linear Independent Components Estimation

$$\det \frac{\partial y_2}{\partial x_2} = 1 \Rightarrow \det \frac{\partial y}{\partial x} = 1$$

NICE: Non-linear Independent Components Estimation

$$\det \frac{\partial y_2}{\partial x_2} = 1 \Rightarrow \det \frac{\partial y}{\partial x} = \boxed{1}$$

- Transformation determinant of the Jacobian **does not** depend on m(·)

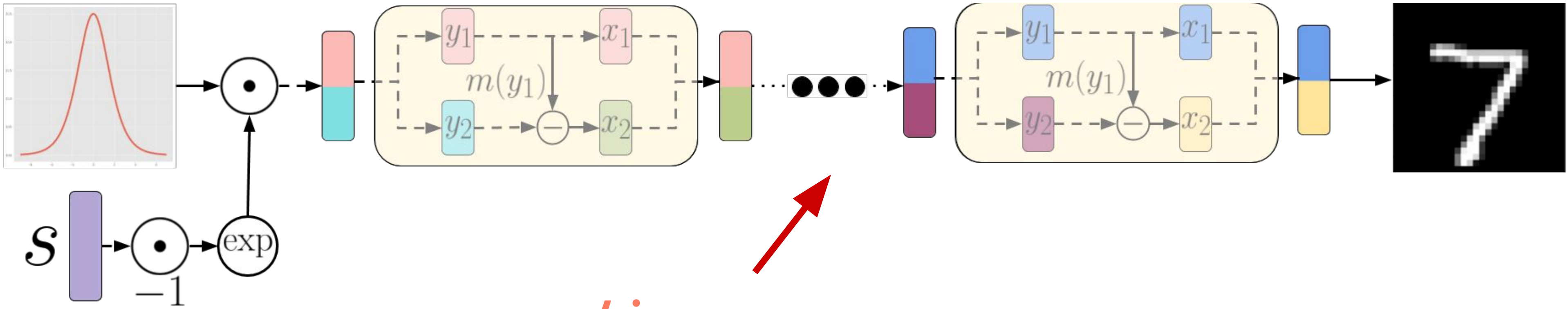
NICE: Non-linear Independent Components Estimation

$$\det \frac{\partial y_2}{\partial x_2} = 1 \Rightarrow \det \frac{\partial y}{\partial x} = \boxed{1}$$

- Transformation determinant of the Jacobian **does not** depend on m(·)
- Function m(·) can be arbitrarily complex, e.g. neural network

NICE - sampling

Inverse transformation



4 inverse coupling layers

Scaling vector

Source: Dinh, Laurent, et. al. **"NICE: Non-linear Independent Components Estimation "** ICLR 2015 (Workshop)

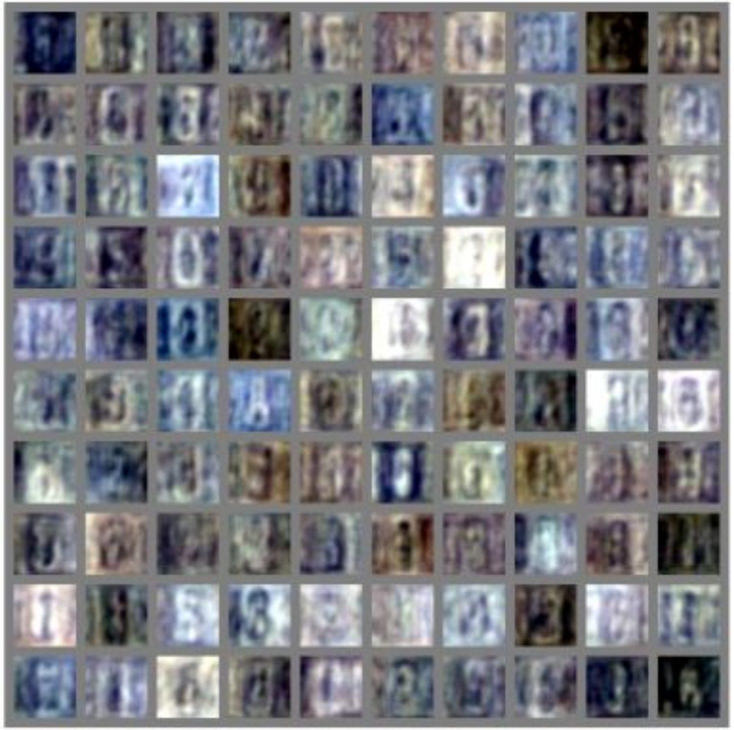
NICE - samples

(a) Model trained on MNIST

(b) Model trained on TFD

Source: Dinh, Laurent, et. al. **"NICE: Non-linear Independent Components Estimation "** ICLR 2015 (Workshop)

NICE - samples



(c) Model trained on SVHN

(d) Model trained on CIFAR-10

Source: Dinh, Laurent, et. al. **"NICE: Non-linear Independent Components Estimation "** ICLR 2015 (Workshop)

TOOPLOOX

www.tooploox.com