Deep Generative Models

Lecture 5

Roman Isachenko

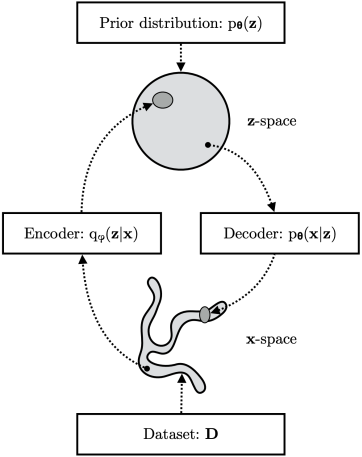
Moscow Institute of Physics and Technology

2022

Recap of previous lecture

Variational autoencoder (VAE)

- VAE learns stochastic mapping between $\mathbf{x}$-space, from $\pi(\mathbf{x})$, and a latent $\mathbf{z}$-space, with simple distribution.
- The generative model learns distribution $p(\mathbf{x}, \mathbf{z}|\boldsymbol{\theta}) = p(\mathbf{z})p(\mathbf{x}|\mathbf{z}, \boldsymbol{\theta})$, with a prior distribution $p(\mathbf{z})$, and a stochastic decoder $p(\mathbf{x}|\mathbf{z}, \boldsymbol{\theta})$.
- The stochastic encoder $q(\mathbf{z}|\mathbf{x}, \phi)$ (inference model), approximates the true but intractable posterior $p(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta})$.



Kingma D. P., Welling M. An introduction to variational autoencoders, 2019

Recap of previous lecture

LVM

$$p(\mathbf{x}|\boldsymbol{\theta}) = \int p(\mathbf{x}, \mathbf{z}|\boldsymbol{\theta}) d\mathbf{z} = \int p(\mathbf{x}|\mathbf{z}, \boldsymbol{\theta}) p(\mathbf{z}) d\mathbf{z}$$

- More powerful $p(\mathbf{x}|\mathbf{z}, \boldsymbol{\theta})$ leads to more powerful generative model $p(\mathbf{x}|\boldsymbol{\theta})$.
- Too powerful $p(\mathbf{x}|\mathbf{z}, \boldsymbol{\theta})$ could lead to posterior collapse: $q(\mathbf{z}|\mathbf{x})$ will not carry any information about $\mathbf{x}$ and close to prior $p(\mathbf{z})$.

Autoregressive decoder

$$p(\mathbf{x}|\mathbf{z}, \boldsymbol{\theta}) = \prod_{j=1}^{m} p(x_j|\mathbf{x}_{1:j-1}, \mathbf{z}, \boldsymbol{\theta})$$

- Global structure is captured by latent variables.
- Local statistics are captured by limited receptive field autoregressive model.

Gulrajani I. et al. PixelVAE: A Latent Variable Model for Natural Images, 2016

Recap of previous lecture

Decoder weakening

- Powerful decoder $p(\mathbf{x}|\mathbf{z}, \boldsymbol{\theta})$ makes the model expressive, but posterior collapse is possible.
- PixelVAE model uses the autoregressive PixelCNN model with small number of layers to limit receptive field.

KL annealing

$$\mathcal{L}(q, \boldsymbol{\theta}, \beta) = \mathbb{E}_{q(\mathbf{z}|\mathbf{x}, \phi)} \log p(\mathbf{x}|\mathbf{z}, \boldsymbol{\theta}) - \beta \cdot KL(q(\mathbf{z}|\mathbf{x}, \phi)||p(\mathbf{z}))$$

Start training with $\beta = 0$, increase it until $\beta = 1$ during training.

Free bits

Ensure the use of less than λ bits of information:

$$\mathcal{L}(q, \boldsymbol{\theta}, \lambda) = \mathbb{E}_{q(\mathbf{z}|\mathbf{x}, \phi)} \log p(\mathbf{x}|\mathbf{z}, \boldsymbol{\theta}) - \max(\lambda, KL(q(\mathbf{z}|\mathbf{x}, \phi)||p(\mathbf{z}))).$$

This results in $KL(q(\mathbf{z}|\mathbf{x}, \phi)||p(\mathbf{z})) \geq \lambda$.

Recap of previous lecture

VAE objective

$$\log p(\mathbf{x}|\boldsymbol{\theta}) \geq \mathcal{L}(q, \boldsymbol{\theta}) = \mathbb{E}_{q(\mathbf{z}|\mathbf{x}, \phi)} \log \frac{p(\mathbf{x}, \mathbf{z}|\boldsymbol{\theta})}{q(\mathbf{z}|\mathbf{x}, \phi)} \rightarrow \max_{q, \boldsymbol{\theta}}$$

IWAE objective

$$\mathcal{L}_K(q, \boldsymbol{\theta}) = \mathbb{E}_{\mathbf{z}_1, \dots, \mathbf{z}_K \sim q(\mathbf{z}|\mathbf{x}, \phi)} \log \left(\frac{1}{K} \sum_{k=1}^{K} \frac{p(\mathbf{x}, \mathbf{z}_k|\boldsymbol{\theta})}{q(\mathbf{z}_k|\mathbf{x}, \phi)} \right) \rightarrow \max_{\phi, \boldsymbol{\theta}}.$$

Theorem

1. $\log p(\mathbf{x}|\boldsymbol{\theta}) \geq \mathcal{L}_K(q, \boldsymbol{\theta}) \geq \mathcal{L}_M(q, \boldsymbol{\theta}) \geq \mathcal{L}(q, \boldsymbol{\theta}), \quad \text{for } K \geq M;$
2. $\log p(\mathbf{x}|\boldsymbol{\theta}) = \lim_{K \to \infty} \mathcal{L}_K(q, \boldsymbol{\theta})$ if $\frac{p(\mathbf{x}, \mathbf{z}|\boldsymbol{\theta})}{q(\mathbf{z}|\mathbf{x}, \phi)}$ is bounded.

- IWAE makes the variational bound tighter and extends the class of variational distributions.
- Gradient signal becomes really small, training is complicated.
- IWAE is a standard quality measure for VAE models.

Burda Y., Grosse R., Salakhutdinov R. Importance Weighted Autoencoders, 2015

Outline

1. Normalizing flows

2. Forward and Reverse KL for Normalizing flows

3. Residual and Linear flows

Outline

1. Normalizing flows
2. Forward and Reverse KL for Normalizing flows
3. Residual and Linear flows

Likelihood-based models so far...

Autoregressive models

$$p(\mathbf{x}|\boldsymbol{\theta}) = \prod_{j=1}^{m} p(x_j|\mathbf{x}_{1:j-1}, \boldsymbol{\theta})$$

- tractable likelihood,
- no inferred latent factors.

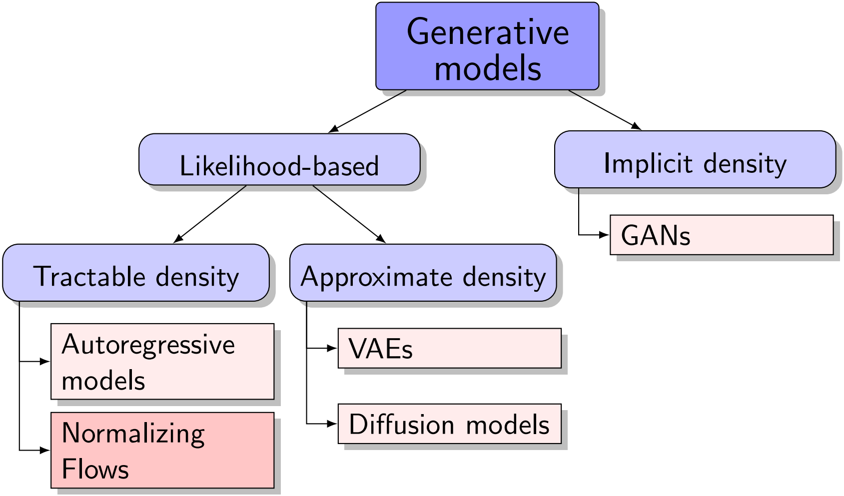
Latent variable models

$$p(\mathbf{x}|\boldsymbol{\theta}) = \int p(\mathbf{x}, \mathbf{z}|\boldsymbol{\theta}) d\mathbf{z}$$

- latent feature representation,
- intractable likelihood.

How to build model with latent variables and tractable likelihood?

Generative models zoo



Normalizing flows prerequisites

Jacobian matrix

$$\mathbf{z} = f(\mathbf{x}), \quad \mathbf{J} = \frac{\partial \mathbf{z}}{\partial \mathbf{x}} = \begin{pmatrix} \frac{\partial z_1}{\partial x_1} & \cdots & \frac{\partial z_1}{\partial x_m} \\ \cdots & \cdots & \cdots \\ \frac{\partial z_m}{\partial x_1} & \cdots & \frac{\partial z_m}{\partial x_m} \end{pmatrix} \in \mathbb{R}^{m \times m}$$

Change of variable theorem (CoV)

Let $\mathbf{x}$ be a random variable with density function $p(\mathbf{x})$ and $f : \mathbb{R}^m \to \mathbb{R}^m$ is a differentiable, invertible function (diffeomorphism). If $\mathbf{z} = f(\mathbf{x})$, $\mathbf{x} = f^{-1}(\mathbf{z}) = g(\mathbf{z})$, then

$$p(\mathbf{x}) = p(\mathbf{z})|\det(\mathbf{J}_f)| = p(\mathbf{z}) \left| \det \left(\frac{\partial \mathbf{z}}{\partial \mathbf{x}} \right) \right| = p(f(\mathbf{x})) \left| \det \left(\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} \right) \right|$$

$$p(\mathbf{z}) = p(\mathbf{x})|\det(\mathbf{J}_g)| = p(\mathbf{x}) \left| \det \left(\frac{\partial \mathbf{x}}{\partial \mathbf{z}} \right) \right| = p(g(\mathbf{z})) \left| \det \left(\frac{\partial g(\mathbf{z})}{\partial \mathbf{z}} \right) \right| .$$

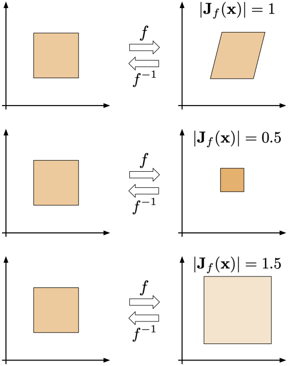
Jacobian determinant

Inverse function theorem

If function f is invertible and Jacobian matrix is continuous and non-singular, then

$$\mathbf{J}_f = \mathbf{J}_{g^{-1}} = \mathbf{J}_g^{-1}, \quad |\det(\mathbf{J}_f)| = \frac{1}{|\det(\mathbf{J}_g)|}$$

- $\mathbf{x}$ and $\mathbf{z}$ have the same dimensionality ($\mathbb{R}^m$).
- $f(\mathbf{x}, \boldsymbol{\theta})$ could be parametric function.
- Determinant of Jacobian matrix $\mathbf{J} = \frac{\partial f(\mathbf{x}, \boldsymbol{\theta})}{\partial \mathbf{x}}$ shows how the volume changes under the transorfmation.



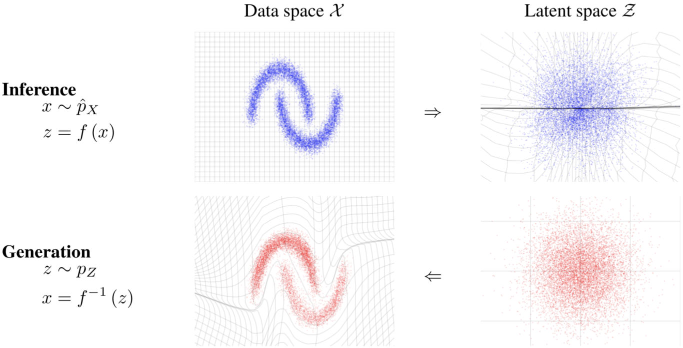
https://jmtomczak.github.io/blog/3/3_flows.html

Fitting flows

MLE problem

$$p(\mathbf{x}|\boldsymbol{\theta}) = p(\mathbf{z}) \left| \det \left(\frac{\partial \mathbf{z}}{\partial \mathbf{x}} \right) \right| = p(f(\mathbf{x}, \boldsymbol{\theta})) \left| \det \left(\frac{\partial f(\mathbf{x}, \boldsymbol{\theta})}{\partial \mathbf{x}} \right) \right|$$

$$\log p(\mathbf{x}|\boldsymbol{\theta}) = \log p(f(\mathbf{x}, \boldsymbol{\theta})) + \log |\det(\mathbf{J}_f)| \to \max_{\boldsymbol{\theta}}$$



Dinh L., Sohl-Dickstein J., Bengio S. Density estimation using Real NVP, 2016

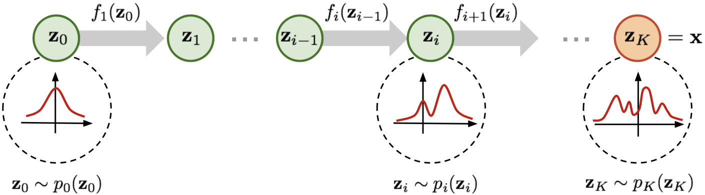
Composition of flows

Theorem

Diffeomorphisms are **composable** (If $\{f_k\}_{k=1}^K$ satisfy conditions of the change of variable theorem, then $\mathbf{z} = f(\mathbf{x}) = f_K \circ \cdots \circ f_1(\mathbf{x})$ also satisfies it).

$$p(\mathbf{x}) = p(f(\mathbf{x})) \left| \det \left(\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} \right) \right| = p(f(\mathbf{x})) \left| \det \left(\frac{\partial \mathbf{f}_K}{\partial \mathbf{f}_{K-1}} \cdots \frac{\partial \mathbf{f}_1}{\partial \mathbf{x}} \right) \right| =$$

$$= p(f(\mathbf{x})) \prod_{k=1}^{K} \left| \det \left(\frac{\partial \mathbf{f}_k}{\partial \mathbf{f}_{k-1}} \right) \right| = p(f(\mathbf{x})) \prod_{k=1}^{K} |\det(\mathbf{J}_{f_k})|$$



https://lilianweng.github.io/lil-log/2018/10/13/flow-based-deep-generative-models.html

Flows

$$\log p(\mathbf{x}|\boldsymbol{\theta}) = \log p(f(\mathbf{x}, \boldsymbol{\theta})) + \log |\det(\mathbf{J}_f)|$$

Definition

Normalizing flow is a *differentiable, invertible* mapping from data $\mathbf{x}$ to the noise $\mathbf{z}$.

- **Normalizing** means that the inverse flow takes samples from $\pi(\mathbf{x})$ and normalizes them into samples from the density $p(\mathbf{z})$.
- **Flow** refers to the trajectory followed by samples from $p(\mathbf{z})$ as they are transformed by the sequence of transformations

$$\mathbf{z} = f_K \circ \cdots \circ f_1(\mathbf{x}); \quad \mathbf{x} = f_1^{-1} \circ \cdots \circ f_K^{-1}(\mathbf{z}) = g_1 \circ \cdots \circ g_K(\mathbf{z})$$

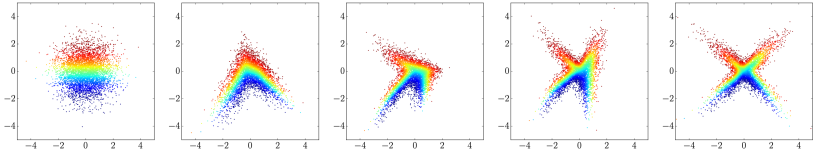
Log likelihood

$$\log p(\mathbf{x}|\boldsymbol{\theta}) = \log p(f_K \circ \cdots \circ f_1(\mathbf{x})) + \sum_{k=1}^{K} \log |\det(\mathbf{J}_{f_k})|,$$

where $\mathbf{J}_{f_k} = \frac{\partial \mathbf{f}_k}{\partial \mathbf{f}_{k-1}}$.

Flows

Example of a 4-step flow



Flow log likelihood

$$\log p(\mathbf{x}|\boldsymbol{\theta}) = \log p(f(\mathbf{x}, \boldsymbol{\theta})) + \log |\det(\mathbf{J}_f)|$$

What is the complexity of the determinant computation?

What we want

- Efficient computation of the Jacobian matrix $\mathbf{J}_f = \frac{\partial f(\mathbf{x}, \boldsymbol{\theta})}{\partial \mathbf{x}}$;
- Efficient sampling from the base distribution $p(\mathbf{z})$;
- Efficient inversion of $f(\mathbf{x}, \boldsymbol{\theta})$.

Papamakarios G. et al. Normalizing flows for probabilistic modeling and inference, 2019

Outline

Forward KL vs Reverse KL

Forward KL

$$
\begin{aligned}
KL(\pi || p) &= \int \pi(\mathbf{x}) \log \frac{\pi(\mathbf{x})}{p(\mathbf{x}|\boldsymbol{\theta})} d\mathbf{x} \\
&= -\mathbb{E}_{\pi(\mathbf{x})} \log p(\mathbf{x}|\boldsymbol{\theta}) + \text{const} \rightarrow \min_{\boldsymbol{\theta}}
\end{aligned}
$$

Maximum likelihood estimation is equivalent to minimization of the Monte-Carlo estimation of forward KL.

Forward KL for flow model

$$
\log p(\mathbf{x}|\boldsymbol{\theta}) = \log p(f(\mathbf{x}, \boldsymbol{\theta})) + \log |\det(\mathbf{J}_f)|
$$

- We need to be able to compute $f(\mathbf{x}, \boldsymbol{\theta})$ and its Jacobian.
- We need to be able to compute the density $p(\mathbf{z})$.
- We don't need to think about computing the function $g(\mathbf{z}, \boldsymbol{\theta}) = f^{-1}(\mathbf{z}, \boldsymbol{\theta})$ until we want to sample from the flow.

Forward KL vs Reverse KL

Reverse KL

$$
\begin{aligned}
KL(p||\pi) &= \int p(\mathbf{x}|\boldsymbol{\theta}) \log \frac{p(\mathbf{x}|\boldsymbol{\theta})}{\pi(\mathbf{x})} d\mathbf{x} \\
&= \mathbb{E}_{p(\mathbf{x}|\boldsymbol{\theta})} \left[\log p(\mathbf{x}|\boldsymbol{\theta}) - \log \pi(\mathbf{x})\right] \rightarrow \min_{\boldsymbol{\theta}}
\end{aligned}
$$

Reverse KL for flow model

$$
\log p(\mathbf{x}|\boldsymbol{\theta}) = \log p(\mathbf{z}) + \log |\det(\mathbf{J}_f)| = \log p(\mathbf{z}) - \log |\det(\mathbf{J}_g)|
$$

$$
KL(p||\pi) = \mathbb{E}_{p(\mathbf{z})} \left[\log p(\mathbf{z}) - \log |\det(\mathbf{J}_g)| - \log \pi(g(\mathbf{z}, \boldsymbol{\theta}))\right]
$$

- We need to be able to compute $g(\mathbf{z}, \boldsymbol{\theta})$ and its Jacobian.
- We need to be able to sample from the density $p(\mathbf{z})$ (do not need to evaluate it) and to evaluate(!) $\pi(\mathbf{x})$.
- We don't need to think about computing the function $f(\mathbf{x}, \boldsymbol{\theta})$.

Flow KL duality

Theorem

Fitting flow model $p(\mathbf{x}|\boldsymbol{\theta})$ to the target distribution $\pi(\mathbf{x})$ using forward KL (MLE) is equivalent to fitting the induced distribution $p(\mathbf{z}|\boldsymbol{\theta})$ to the base $p(\mathbf{z})$ using reverse KL:

$$\arg\min_{\boldsymbol{\theta}} KL(\pi(\mathbf{x})||p(\mathbf{x}|\boldsymbol{\theta})) = \arg\min_{\boldsymbol{\theta}} KL(p(\mathbf{z}|\boldsymbol{\theta})||p(\mathbf{z})).$$

- $p(\mathbf{z})$ is a base distribution; $\pi(\mathbf{x})$ is a data distribution;
- $\mathbf{z} \sim p(\mathbf{z})$, $\mathbf{x} = g(\mathbf{z}, \boldsymbol{\theta})$, $\mathbf{x} \sim p(\mathbf{x}|\boldsymbol{\theta})$;
- $\mathbf{x} \sim \pi(\mathbf{x})$, $\mathbf{z} = f(\mathbf{x}, \boldsymbol{\theta})$, $\mathbf{z} \sim p(\mathbf{z}|\boldsymbol{\theta})$;

$$\log p(\mathbf{z}|\boldsymbol{\theta}) = \log \pi(g(\mathbf{z}, \boldsymbol{\theta})) + \log|\det(\mathbf{J}_g)|;$$
$$\log p(\mathbf{x}|\boldsymbol{\theta}) = \log p(f(\mathbf{x}, \boldsymbol{\theta})) + \log|\det(\mathbf{J}_f)|.$$

Papamakarios G. et al. Normalizing flows for probabilistic modeling and inference, 2019

Flow KL duality

Theorem

Fitting flow model $p(\mathbf{x}|\boldsymbol{\theta})$ to the target distribution $\pi(\mathbf{x})$ using forward KL (MLE) is equivalent to fitting the induced distribution $p(\mathbf{z}|\boldsymbol{\theta})$ to the base $p(\mathbf{z})$ using reverse KL:

$$\arg\min_{\boldsymbol{\theta}} KL(\pi(\mathbf{x})||p(\mathbf{x}|\boldsymbol{\theta})) = \arg\min_{\boldsymbol{\theta}} KL(p(\mathbf{z}|\boldsymbol{\theta})||p(\mathbf{z})).$$

Proof

$$\begin{aligned}
KL\left(p(\mathbf{z}|\boldsymbol{\theta})||p(\mathbf{z})\right) &= \mathbb{E}_{p(\mathbf{z}|\boldsymbol{\theta})}\big[\log p(\mathbf{z}|\boldsymbol{\theta}) - \log p(\mathbf{z})\big] = \\
&= \mathbb{E}_{p(\mathbf{z}|\boldsymbol{\theta})}\left[\log \pi(g(\mathbf{z}, \boldsymbol{\theta})) + \log|\det(\mathbf{J}_g)| - \log p(\mathbf{z})\right] = \\
&= \mathbb{E}_{\pi(\mathbf{x})}\left[\log \pi(\mathbf{x}) - \log|\det(\mathbf{J}_f)| - \log p(f(\mathbf{x}, \boldsymbol{\theta}))\right] = \\
&= \mathbb{E}_{\pi(\mathbf{x})}\big[\log \pi(\mathbf{x}) - \log p(\mathbf{x}|\boldsymbol{\theta})\big] = KL(\pi(\mathbf{x})||p(\mathbf{x}|\boldsymbol{\theta})).
\end{aligned}$$

Papamakarios G., Pavlakou T., Murray I. Masked Autoregressive Flow for Density Estimation, 2017

Outline

Jacobian structure

Flow log-likelihood

$$\log p(\mathbf{x}|\boldsymbol{\theta}) = \log p(f(\mathbf{x}, \boldsymbol{\theta})) + \log \left| \det \left(\frac{\partial f(\mathbf{x}, \boldsymbol{\theta})}{\partial \mathbf{x}} \right) \right|$$

The main challenge is a determinant of the Jacobian matrix.

What is the $det(\mathbf{J})$ in the following cases?

1. Consider a linear layer $\mathbf{z} = \mathbf{W}\mathbf{x}$.
2. Let $\mathbf{z}$ be a permutation of $\mathbf{x}$.
3. Let z_j depend only on $\mathbf{x}_j$.

$$\log \left| \det \left(\frac{\partial f(\mathbf{x}, \boldsymbol{\theta})}{\partial \mathbf{x}} \right) \right| = \log \left| \prod_{j=1}^{m} f_j'(x_j, \boldsymbol{\theta}) \right| = \sum_{j=1}^{m} \log \left| f_j'(x_j, \boldsymbol{\theta}) \right| .$$

4. Let z_j depend only on $\mathbf{x}_{1:j}$ (autoregressive dependency).

Linear flows

$$\mathbf{z} = f(\mathbf{x}, \boldsymbol{\theta}) = \mathbf{W}\mathbf{x}, \quad \mathbf{W} \in \mathbb{R}^{m \times m}, \quad \boldsymbol{\theta} = \mathbf{W}, \quad \mathbf{J}_f = \mathbf{W}$$

In general, we need $O(m^3)$ to invert matrix.

Invertibility

- Diagonal matrix $O(m)$.
- Triangular matrix $O(m^2)$.
- It is impossible to parametrize all invertible matrices.

Invertible 1x1 conv

$\mathbf{W} \in \mathbb{R}^{c \times c}$ - kernel of 1x1 convolution with c input and c output channels. The computational complexity of computing or differentiating $\det(\mathbf{W})$ is $O(c^3)$. Cost to compute $\det(\mathbf{W})$ is $O(c^3)$. It should be invertible.

Kingma D. P., Dhariwal P. Glow: Generative Flow with Invertible 1x1 Convolutions, 2018

Linear flows

$$\mathbf{z} = f(\mathbf{x}, \boldsymbol{\theta}) = \mathbf{W}\mathbf{x}, \quad \mathbf{W} \in \mathbb{R}^{m \times m}, \quad \boldsymbol{\theta} = \mathbf{W}, \quad \mathbf{J}_f = \mathbf{W}$$

Matrix decompositions

- LU-decomposition
$$\mathbf{W} = \mathbf{PLU},$$
where $\mathbf{P}$ is a permutation matrix, $\mathbf{L}$ is lower triangular with positive diagonal, $\mathbf{U}$ is upper triangular with positive diagonal.
- QR-decomposition
$$\mathbf{W} = \mathbf{QR},$$
where $\mathbf{Q}$ is an orthogonal matrix, $\mathbf{R}$ is an upper triangular matrix with positive diagonal.

Kingma D. P., Dhariwal P. Glow: Generative Flow with Invertible 1x1 Convolutions, 2018

Hoogeboom E., Van Den Berg R., and Welling M. Emerging convolutions for generative normalizing flows, 2019

Residual Flows

Matrix determinant lemma

$$\det\left(\mathbf{I}_m + \mathbf{V}\mathbf{W}^T\right) = \det(\mathbf{I}_d + \mathbf{W}^T\mathbf{V}), \quad \text{where } \mathbf{V}, \mathbf{W} \in \mathbb{R}^{m \times d}.$$

Planar flow

$$\mathbf{x} = g(\mathbf{z}, \boldsymbol{\theta}) = \mathbf{z} + \mathbf{v}\,\sigma(\mathbf{w}^T\mathbf{z} + b).$$

Here $\boldsymbol{\theta} = \{\mathbf{v}, \mathbf{w}, b\}$, $\sigma(\cdot)$ is a smooth element-wise non-linearity.

$$\left|\det\left(\frac{\partial g(\mathbf{z}, \boldsymbol{\theta})}{\partial \mathbf{z}}\right)\right| = \left|\det\left(\mathbf{I} + \sigma'(\mathbf{w}^T\mathbf{z} + b)\mathbf{v}\mathbf{w}^T\right)\right| = \left|1 + \sigma'(\mathbf{w}^T\mathbf{z} + b)\mathbf{w}^T\mathbf{v}\right|$$

The transformation is invertible, for example, if

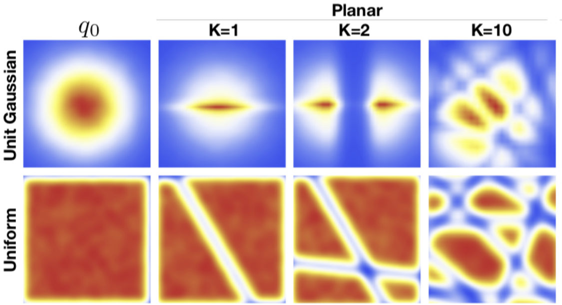
$$\sigma = \tanh; \quad \sigma'(\mathbf{w}^T\mathbf{z} + b)\mathbf{w}^T\mathbf{v} \geq -1.$$

Rezende D. J., Mohamed S. Variational Inference with Normalizing Flows, 2015

Residual Flows

Expressiveness of planar flows

$$\mathbf{z}_K = g_1 \circ \dots \circ g_K(\mathbf{z}); \quad g_k = g(\mathbf{z}_k, \boldsymbol{\theta}_k) = \mathbf{z}_k + \mathbf{v}_k \, \sigma(\mathbf{w}_k^T \mathbf{z}_k + b_k).$$



Sylvester flow: planar flow extension

$$g(\mathbf{z}, \boldsymbol{\theta}) = \mathbf{z} + \mathbf{V} \, \sigma(\mathbf{W}^T \mathbf{z} + \mathbf{b}).$$

Rezende D. J., Mohamed S. Variational Inference with Normalizing Flows, 2015
Berg R. et al. Sylvester normalizing flows for variational inference, 2018

Summary

- Flow models transform a simple base distribution to a complex one via a sequence of invertible transformations with tractable Jacobian.
- Flow models have a tractable likelihood that is given by the change of variable theorem.
- Flows could be fitted using forward and reverse KL minimization. We will consider each of the scenarios later in the course.
- Linear flows try to parametrize set of invertible matrices via matrix decompositions.
- Planar and Sylvester flows are residual flows which use matrix determinant lemma.