Normalizing flows - part 2

Overview

Wednesday:

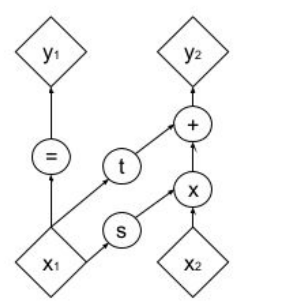
1) **Various N-d non-linear generalizations**

2) **Conditional PDFs: Amortization and the connection to deep learning**
3) **A NF tool: jammy_flows**
4) **Hands-on 1**

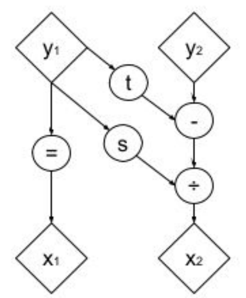
5) **Probabilistic deep learning**
6) **Coverage / Systematics / Goodness-of-Fit**
7) **Hands-on 2**

Certain N-d non-linear normalizing-flow functions

1) “Affine Coupling layer” (Dinh et al. 2014 (NICE) / Dinh et al. 2017 (real-NVP))



(a) Forward propagation



(b) Inverse propagation

$$
\begin{aligned}
y_{1:d} &= x_{1:d} \\
y_{d+1:D} &= x_{d+1:D} \odot \exp\big(s(x_{1:d})\big) + t(x_{1:d}),
\end{aligned}
$$

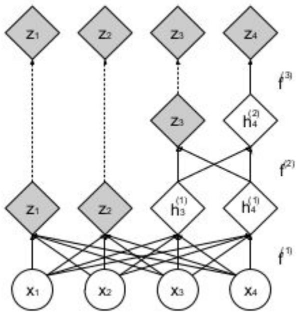
$$
\frac{\partial y}{\partial x^T} = \begin{bmatrix} \mathbb{I}_d & 0 \\ \frac{\partial y_{d+1:D}}{\partial x^T_{1:d}} & \operatorname{diag}\big(\exp\left[s\left(x_{1:d}\right)\right]\big) \end{bmatrix},
$$

Induces coupling between different layers

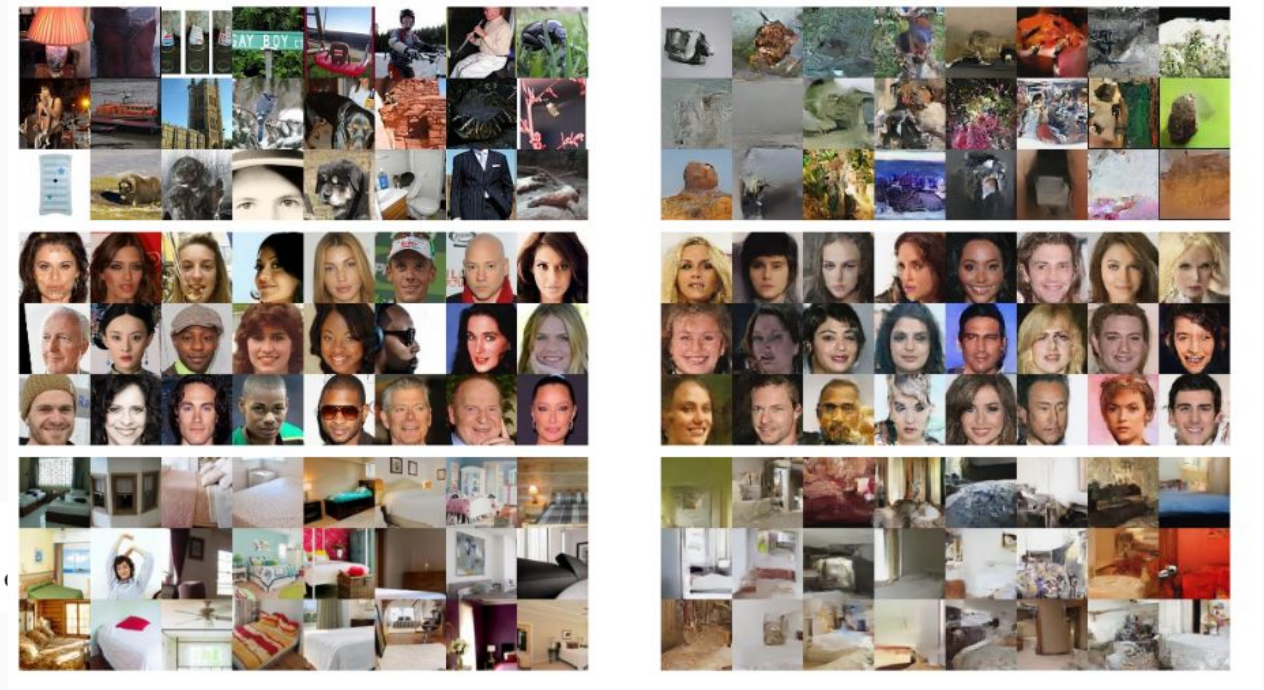
Functions ***s*** and ***t*** can be arbitrarily complex (neural networks)

These functions are also called “conditioners”

1) “Affine Coupling layer” (Dinh et al. 2014 (NICE) / Dinh et al. 2017 (real-NVP))



(b) Factoring out variables. At each step, half the variables are directly modeled as Gaussians, while the other half undergo further transformation.



Can be scaled to very high dimensions (here images)

Certain N-d non-linear normalizing-flow functions

2) "Autoregressive" structure

$$p(x_0) \cdot p(x_1; x_0) \cdot \ldots p(x_i; x_{i-1}, x_{i-2}, \ldots, x_0)$$

Flow function for "affine" example:

$$y_0 = \mu_0 + \sigma_0 \odot \epsilon_0, \qquad y_i = \mu_i(\mathbf{y}_{1:i-1}) + \sigma_i(\mathbf{y}_{1:i-1}) \cdot \epsilon_i$$

Unlike Coupling layers, no dimensions remain untransformed, and Conditioning on previous transformed variables

Certain N-d non-linear normalizing-flow functions

2) "Autoregressive" structure

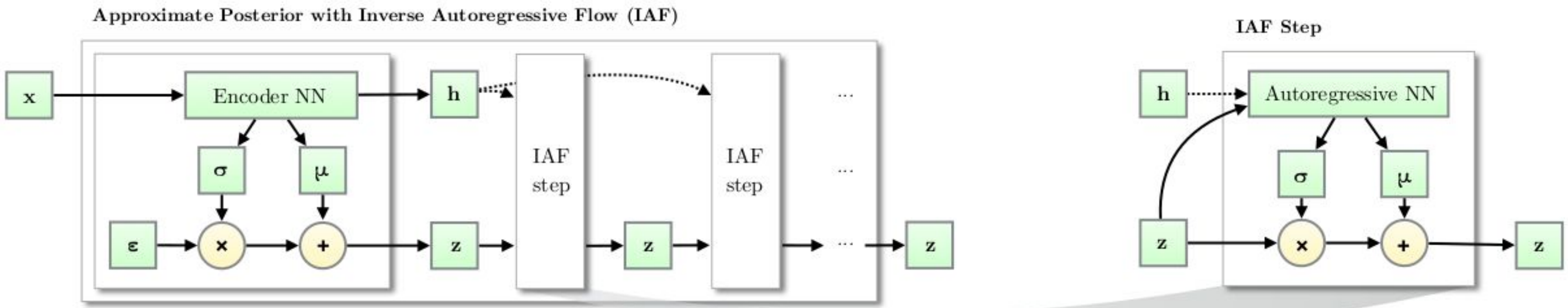
$$p(x_0) \cdot p(x_1; x_0) \cdot \ldots p(x_i; x_{i-1}, x_{i-2}, \ldots, x_0)$$

Flow function for "affine" example:

$$y_0 = \mu_0 + \sigma_0 \odot \epsilon_0, \qquad y_i = \mu_i(\mathbf{y}_{1:i-1}) + \sigma_i(\mathbf{y}_{1:i-1}) \cdot \epsilon_i$$

Unlike Coupling layers, no dimensions remain untransformed, and Conditioning on previous transformed variables

One can also interpret the autoregressive transformations as acting in time Inverse Autoregressive flows (IAFs, Kingma et al. 2016)



Certain N-d non-linear normalizing-flow functions

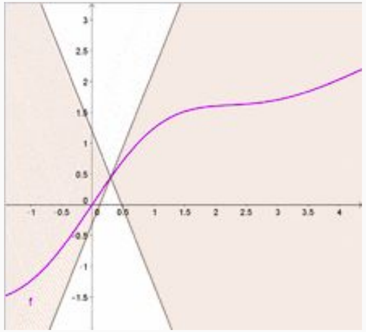
3) Use Invertible residual networks:
(Behrman et al. 2019)

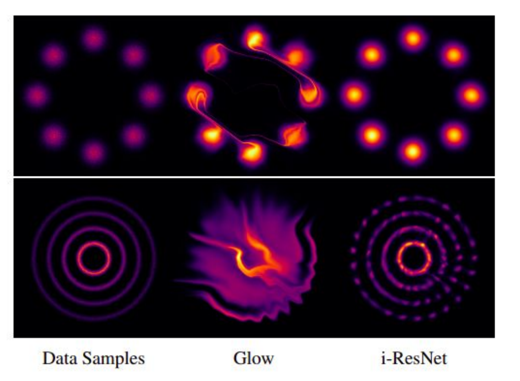
$$x_{t+1} \leftarrow x_t + g_{\theta_t}(x_t)$$

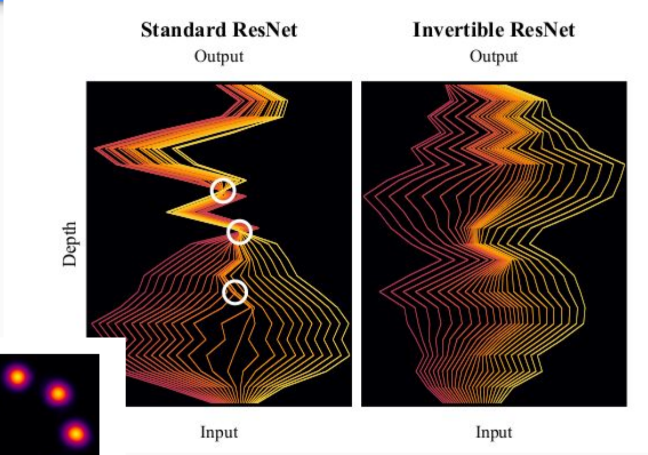
invertible if

$$\mathrm{Lip}(g_{\theta_t}) < 1, \textit{ for all } t = 1, \ldots, T,$$

where $\mathrm{Lip}(g_{\theta_t})$ *is the Lipschitz-constant of* g_{θ_t}.







Certain N-d non-linear normalizing-flow functions

3) Use Invertible residual networks:
(Behrman et al. 2019)

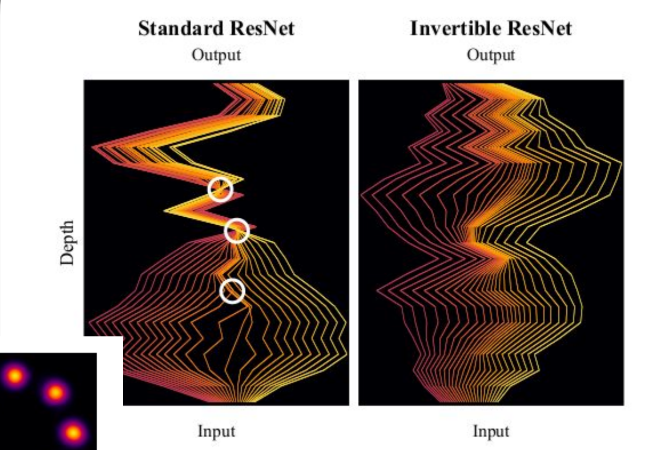
Flow parameters are NN network parameters

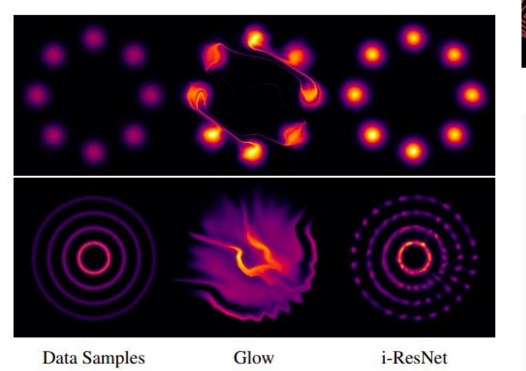
$$x_{t+1} \leftarrow x_t + g_{\theta_t}(x_t)$$

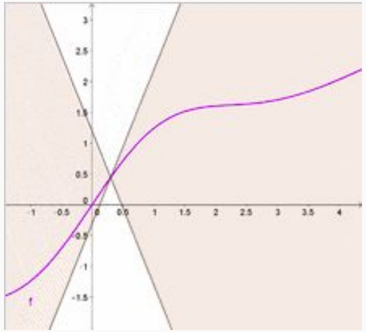
invertible if

$$\mathrm{Lip}(g_{\theta_t}) < 1, \textit{ for all } t = 1, \ldots, T,$$

where $\mathrm{Lip}(g_{\theta_t})$ *is the Lipschitz-constant of* g_{θ_t}.







Lipschitz constraint similar to critic constraint in WGAN training

Certain N-d non-linear normalizing-flow functions

4) Gaussianization Flow(arXiv:2003.01941)

$$D_{\mathrm{KL}}\left(p(\mathbf{x}) \,\|\, \mathcal{N}(\mathbf{0},\mathbf{I})\right) \triangleq J(\mathbf{x}) = I(\mathbf{x}) + J_m(\mathbf{x})$$

"Multi information / total correlation"
-> correlation between variables

$$I(\mathbf{x}) = D_{\mathrm{KL}}\left(p(\mathbf{x}) \,\middle\|\, \prod_i^D p_i(x^{(i)})\right)$$

$$J_m(\mathbf{x}) = \sum_{i=1}^D D_{\mathrm{KL}}\left(p_i(x^{(i)}) \,\middle\|\, \mathcal{N}(0,1)\right).$$

"Procedure of **Gaussianization**"
-> Transform a random variable into Gaussian
By: 1) reducing correlations
2) making marginals = standard normal

Iterative Gaussianization (Chen and Gopinath, 2001)
Achieve 1) + 2) by iterated application of a Rotation (e.g. defined by PCA matrix) and nonlinear transformation of individual dimensions

New idea: **Learn** Rotations + nonlinear transformations in a normalizing flow $T_{\boldsymbol{\theta}}(\mathbf{x}) = \Psi_{\boldsymbol{\theta}_L} \circ R_L \circ \Psi_{\boldsymbol{\theta}_{L-1}} \circ \cdots \circ \Psi_{\boldsymbol{\theta}_1} \circ R_1 \mathbf{x}$

Certain N-d non-linear normalizing-flow functions

4) Gaussianization Flow(arXiv:2003.01941)

"Multi information / total correlation"
-> correlation between variables

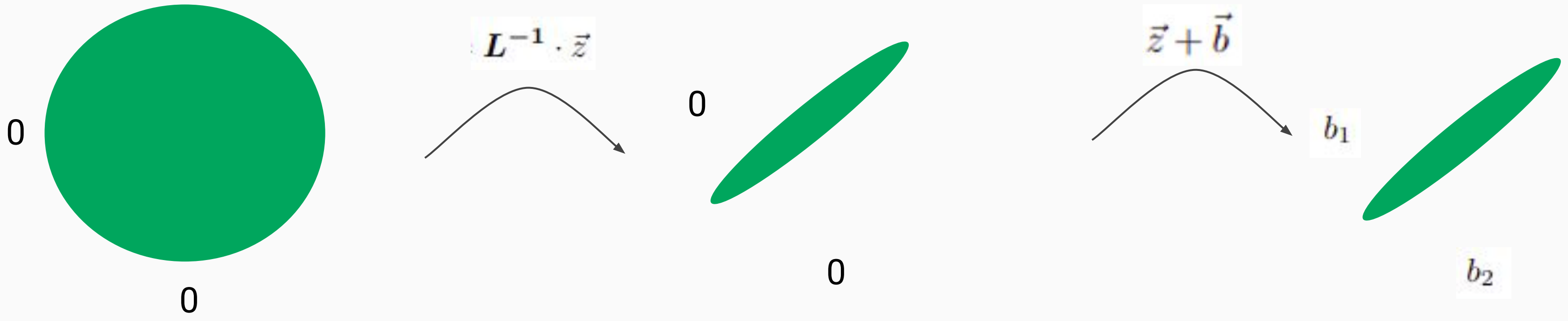
A better intuitive understanding:

Start with Affine flow (Gaussian):

$$p(x) = \frac{1}{\sqrt{2 \cdot \pi}} \cdot |\det(\boldsymbol{L})| \cdot \exp\left(-0.5 \cdot (\vec{x} - \vec{b})^T \cdot \boldsymbol{L}^T \cdot \boldsymbol{L} \cdot (\vec{x} - \vec{b})\right)$$

Inv. Flow function

$$\vec{z} = f^{-1}(\vec{x}) = \boldsymbol{L}(\vec{x} - \vec{b})$$



Certain N-d non-linear normalizing-flow functions

4) Gaussianization Flow(arXiv:2003.01941)

“Multi information / total correlation”
-> correlation between variables

A better intuitive understanding:

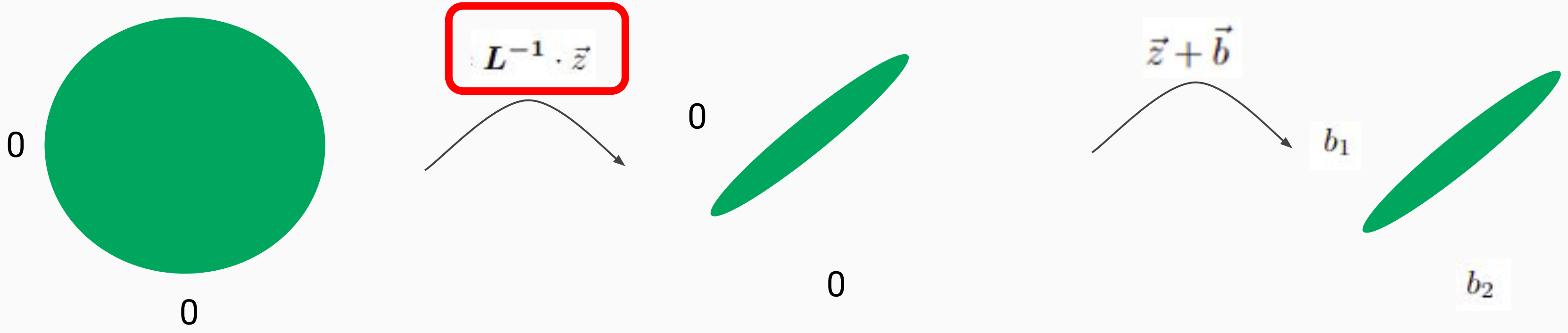
Start with Affine flow (Gaussian):

$$p(x) = \frac{1}{\sqrt{2\cdot\pi}} \cdot |\det(\boldsymbol{L})| \cdot \exp\left(-0.5 \cdot (\vec{x}-\vec{b})^T \cdot \boldsymbol{L}^T \cdot \boldsymbol{L} \cdot (\vec{x}-\vec{b})\right)$$

Inv. Flow function

$$\vec{z} = f^{-1}(\vec{x}) = \boldsymbol{L}(\vec{x}-\vec{b})$$

Let us decompose this transformation further



Certain N-d non-linear normalizing-flow functions

4) Gaussianization Flow(arXiv:2003.01941)

"Multi information / total correlation"
-> correlation between variables

A better intuitive understanding:

Start with Affine flow (Gaussian):

Inv. Flow function

$$\vec{z} = f^{-1}(\vec{x}) = \boldsymbol{L}(\vec{x} - \vec{b})$$

$$p(x) = \frac{1}{\sqrt{2 \cdot \pi}} \cdot |\det(\boldsymbol{L})| \cdot \exp\left(-0.5 \cdot (\vec{x} - \vec{b})^T \cdot \boldsymbol{L}^T \cdot \boldsymbol{L} \cdot (\vec{x} - \vec{b})\right)$$

$$p(x) = \frac{1}{\sqrt{2 \cdot \pi}} \cdot |\det(\boldsymbol{D})| \cdot \exp\left(-0.5 \cdot (\vec{x} - \vec{b})^T \cdot \boldsymbol{V} \cdot \boldsymbol{D} \cdot \boldsymbol{U}^T \cdot \boldsymbol{U} \cdot \boldsymbol{D} \cdot \boldsymbol{V}^T \cdot (\vec{x} - \vec{b})\right)$$

Certain N-d non-linear normalizing-flow functions

4) Gaussianization Flow(arXiv:2003.01941)

"Multi information / total correlation"
-> correlation between variables

A better intuitive understanding:

Start with Affine flow (Gaussian):

Inv. Flow function

$$p(x) = \frac{1}{\sqrt{2\cdot\pi}} \cdot |\det(\boldsymbol{L})| \cdot \exp\left(-0.5 \cdot (\vec{x}-\vec{b})^T \cdot \boldsymbol{L}^T \cdot \boldsymbol{L} \cdot (\vec{x}-\vec{b})\right)$$

$$\vec{z} = f^{-1}(\vec{x}) = \boldsymbol{L}(\vec{x}-\vec{b})$$

$$p(x) = \frac{1}{\sqrt{2\cdot\pi}} \cdot |\det(\boldsymbol{D})| \cdot \exp\left(-0.5 \cdot (\vec{x}-\vec{b})^T \cdot \boldsymbol{V} \cdot \boldsymbol{D} \cdot \boldsymbol{U}^T \cdot \boldsymbol{U} \cdot \boldsymbol{D} \cdot \boldsymbol{V}^T \cdot (\vec{x}-\vec{b})\right)$$

$$p(x) = \frac{1}{\sqrt{2\cdot\pi}} \cdot |\det(\boldsymbol{D})| \cdot \exp\left(-0.5 \cdot (\boldsymbol{D} \cdot \boldsymbol{O} \cdot (\vec{x}-\vec{b}))^T \cdot \boldsymbol{D} \cdot \boldsymbol{O} \cdot (\vec{x}-\vec{b})\right)$$

$$f^{-1}(\vec{x}) = \boldsymbol{D} \cdot \boldsymbol{O}(\vec{x}-\vec{b})$$

Flow function
Is actually
Scaling + Rotation $\longrightarrow$ $f(\vec{z}) = \boldsymbol{O} \cdot \boldsymbol{D}(\vec{x}) + \vec{b}$

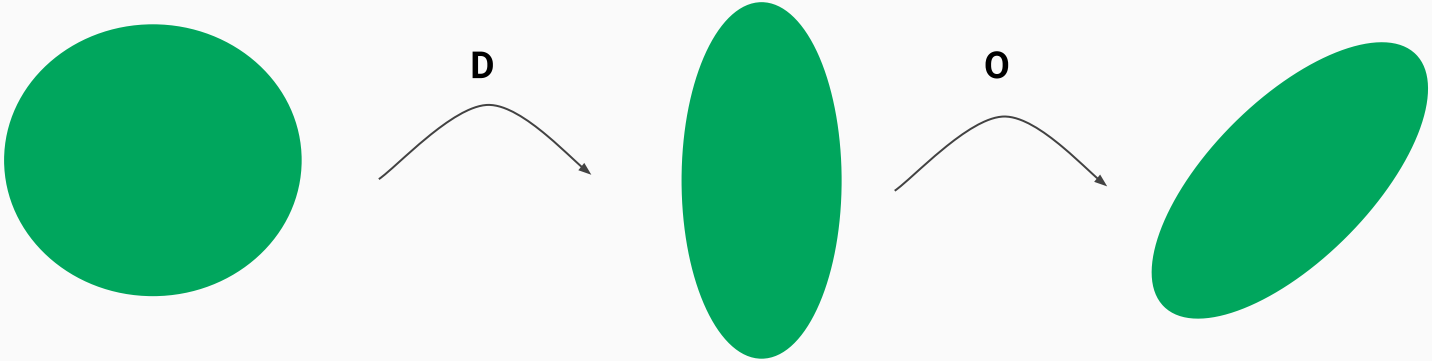
Certain N-d non-linear normalizing-flow functions

4) Gaussianization Flow(arXiv:2003.01941)

“Multi information / total correlation”
-> correlation between variables

Flow function
Is actually
Scaling + Rotation

$$f(\vec{z}) = \boldsymbol{O} \cdot \boldsymbol{D}(\vec{x}) + \vec{b}$$



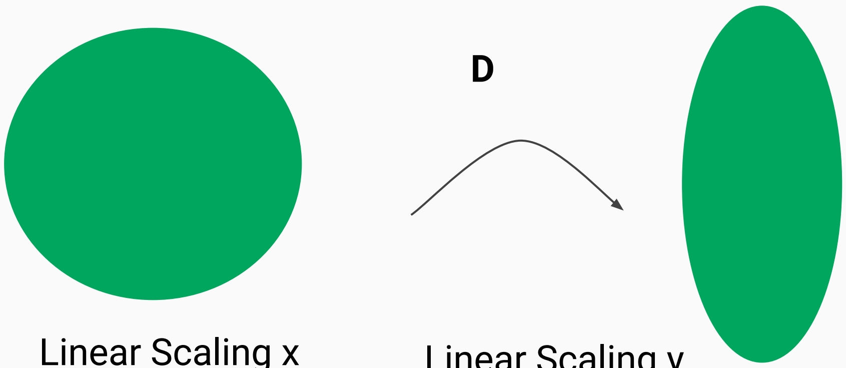
Certain N-d non-linear normalizing-flow functions

4) Gaussianization Flow(arXiv:2003.01941)

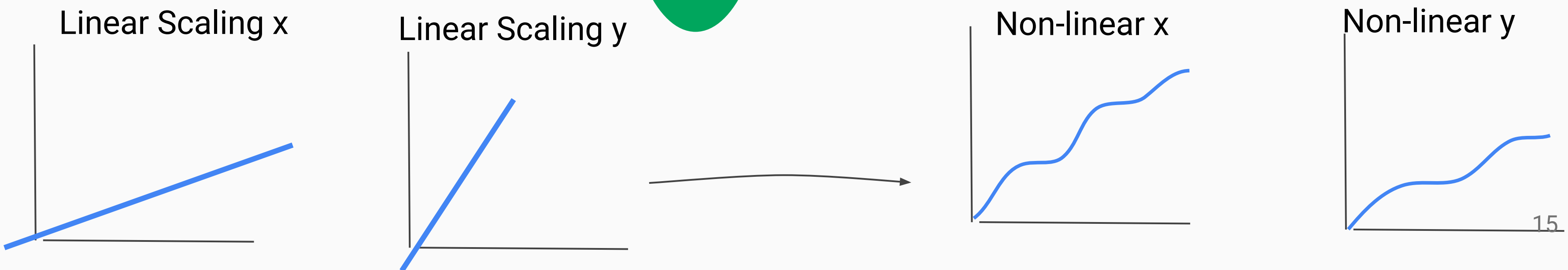
“Multi information / total correlation”
-> correlation between variables

Flow function
Is actually
Scaling + Rotation

$$f(\vec{z}) = \boldsymbol{O} \cdot \boldsymbol{D}(\vec{x}) + \vec{b}$$



$$\boldsymbol{D} = \begin{pmatrix} a_x & 0 \\ 0 & a_y \end{pmatrix}$$



Certain N-d non-linear normalizing-flow functions

4) Gaussianization Flow(arXiv:2003.01941)

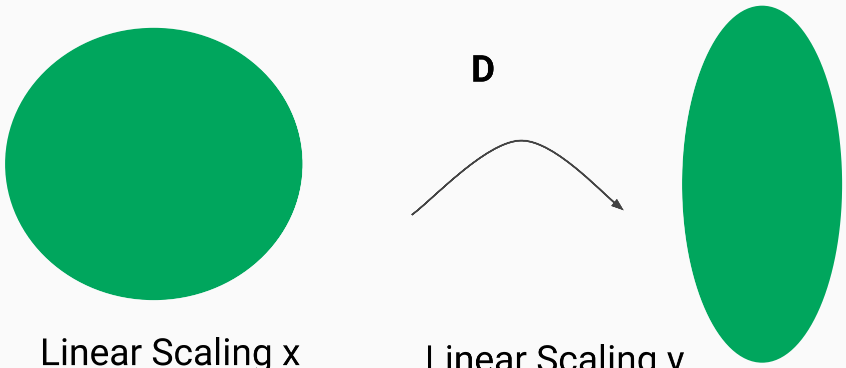
Flow function
Is actually
Scaling + Rotation → $f(\vec{z}) = \boldsymbol{O} \cdot \boldsymbol{D}(\vec{x}) + \vec{b}$

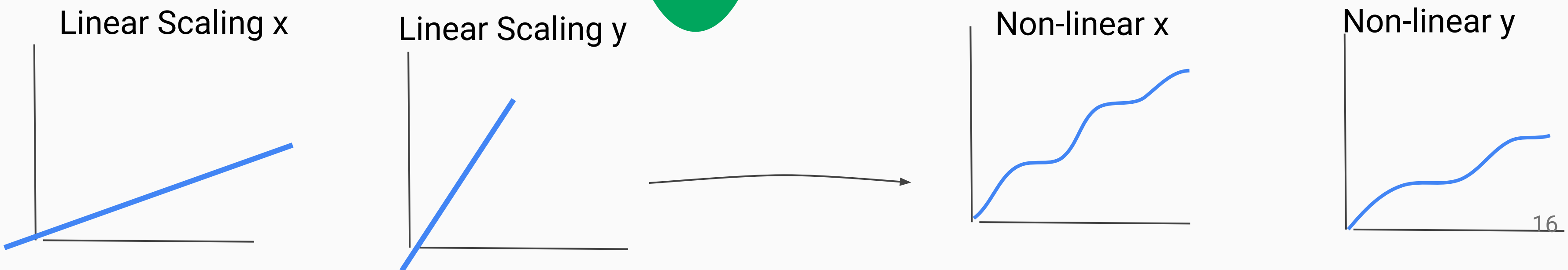
"Multi information / total correlation"
-> correlation between variables

Apply many iterations of "rotation" and non-linear
Dimension-wise scaling

$$T_{\boldsymbol{\theta}}(\mathbf{x}) = \Psi_{\boldsymbol{\theta}_L} \circ R_L \circ \Psi_{\boldsymbol{\theta}_{L-1}} \circ \cdots \circ \Psi_{\boldsymbol{\theta}_1} \circ R_1 \mathbf{x}$$

$$\boldsymbol{D} = \begin{pmatrix} a_x & 0 \\ 0 & a_y \end{pmatrix}$$

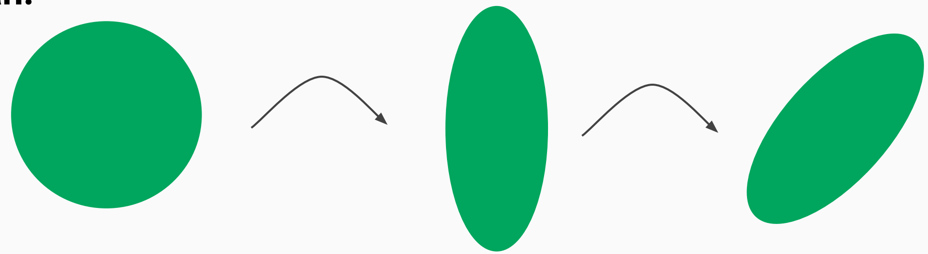




Certain N-d non-linear normalizing-flow functions

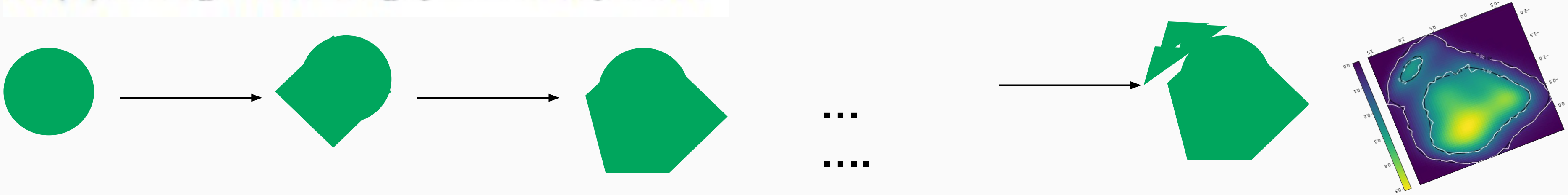
4) Gaussianization Flow(arXiv:2003.01941)

Gaussian:



Gaussianization flow generalized by non-linear scalings instead of “linear scalings”

$$T_{\boldsymbol{\theta}}(\mathbf{x}) = \Psi_{\boldsymbol{\theta}_L} \circ R_L \circ \Psi_{\boldsymbol{\theta}_{L-1}} \circ \cdots \circ \Psi_{\boldsymbol{\theta}_1} \circ R_1 \mathbf{x}$$

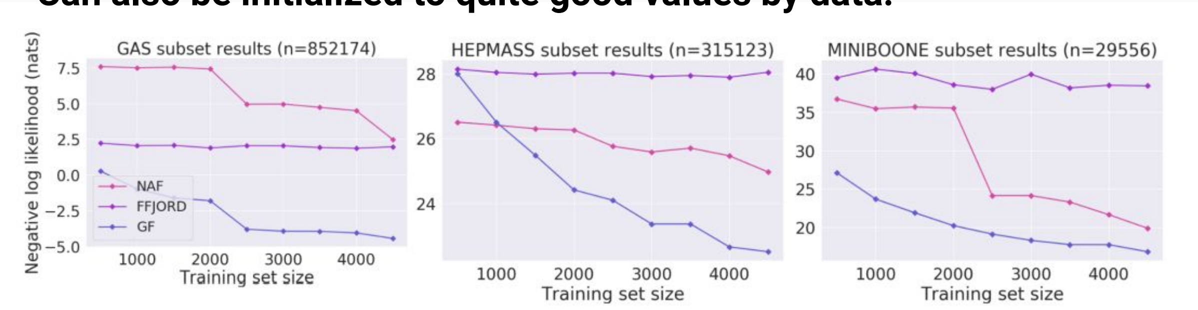


Step by step refinement of the PDF - provably approximates any distribution

Certain N-d non-linear normalizing-flow functions

4) Gaussianization Flow(arXiv:2003.01941)

Can also be initialized to quite good values by data!



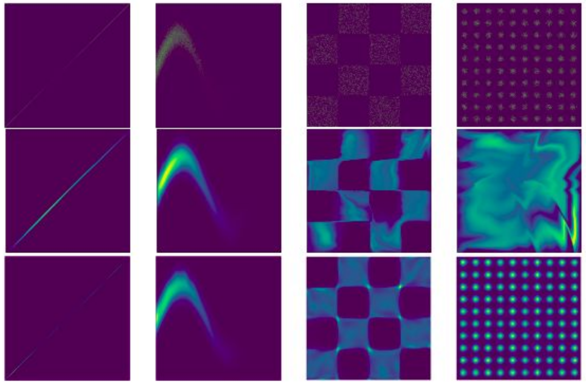


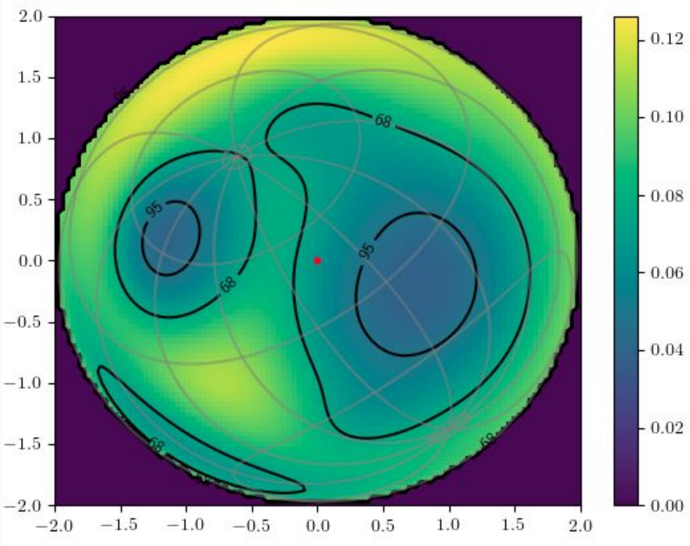
Figure 2: 2D density estimation results. **Top:** Ground truth samples. **Middle:** Glow. **Bottom:** GF.

Method	POWER	GAS	HEPMASS	MINIBOONE	BSDS300	MNIST	FMNIST
Real NVP	-0.17	-8.33	18.71	13.55	-153.28	1.06	**2.85**
Glow	-0.17	-8.15	18.92	11.35	-155.07	1.05	2.95
FFJORD	-0.46	-8.59	**14.92**	10.43	**-157.40**	**0.99**	-
RBIG	1.02	0.05	24.59	25.41	-115.96	1.71	4.46
GF(ours)	**-0.57**	**-10.13**	17.59	**10.32**	-152.82	1.29	3.35
MADE	3.08	-3.56	20.98	15.59	-148.85	2.04	4.18
MAF	-0.24	-10.08	17.70	11.75	-155.69	1.89	-
TAN	-0.48	-11.19	15.12	11.01	-157.03	-	-
MAF-DDSF	-0.62	-11.96	15.09	8.86	-157.73	-	-

Other interesting current research

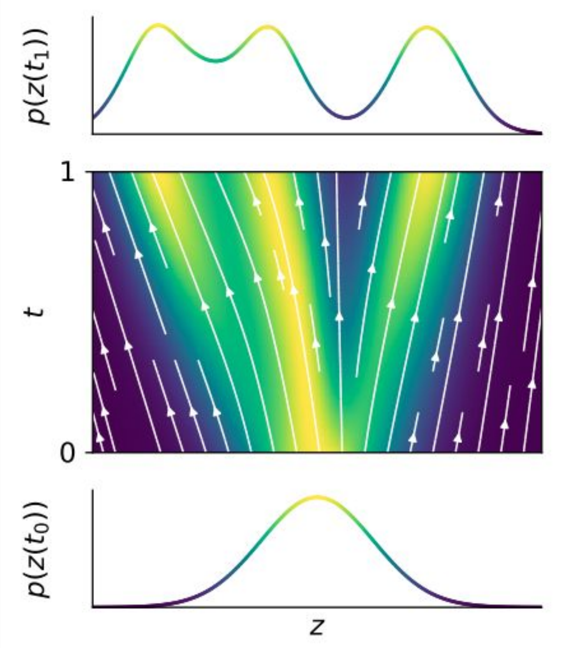
“Manifold” normalizing flows

$$p(T(x)) = \frac{\pi(x)}{\sqrt{\det(E^\top J^\top J E)}}$$



(2002.02428)

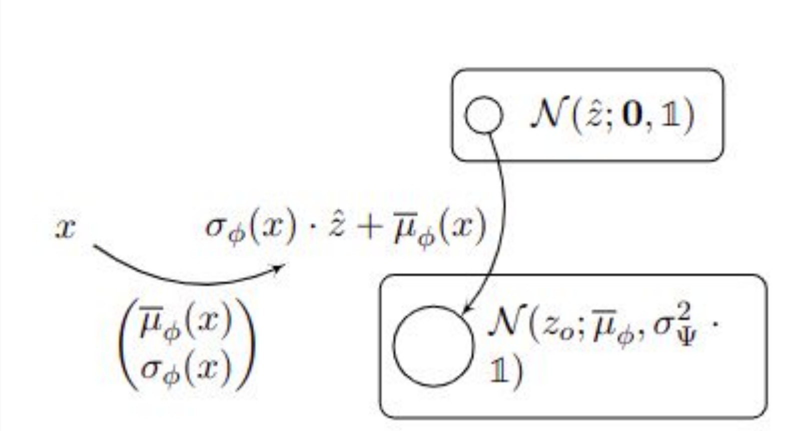
“Continuous” normalizing flows



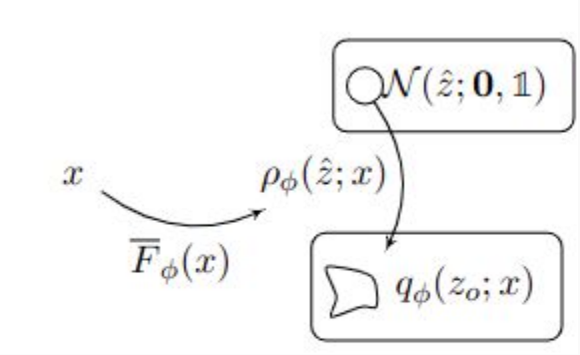
(FFJORD, 1810.01367)

Conditional PDFs .. parameters of flow are output of a neural network

Conditional affine flow



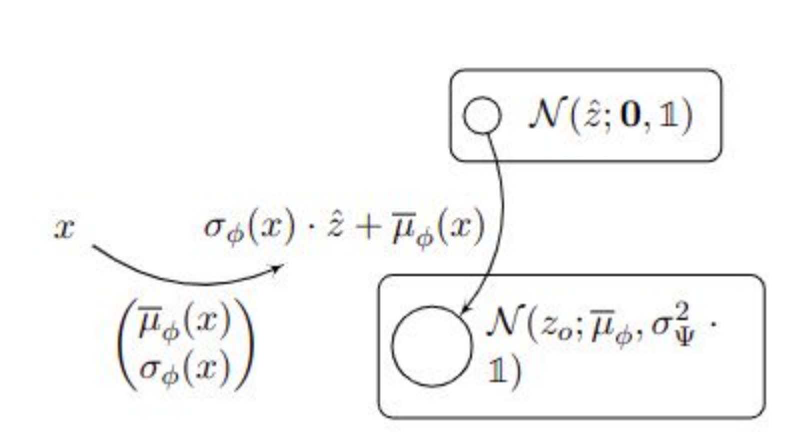
Conditional general flow



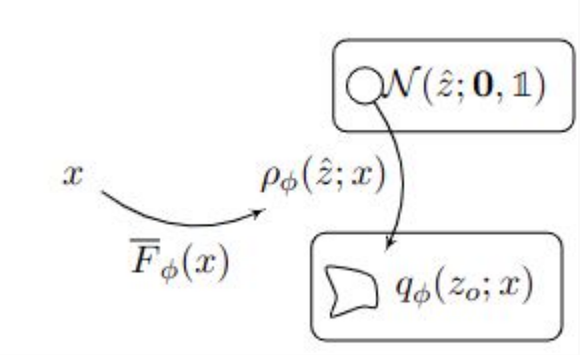
Instead of flow parameters, one optimizes NN parameters

Conditional PDFs .. parameters of flow are output of a neural network

Conditional affine flow



Conditional general flow

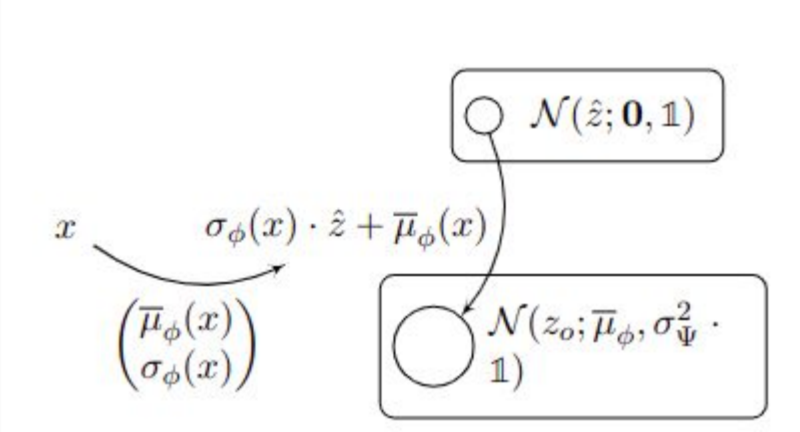


→ Instead of flow parameters, one optimizes NN parameters

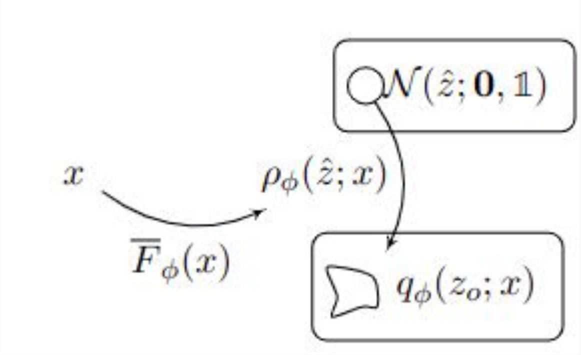
→ Conditional normalizing flow shows that **MSE loss comes from conditional Flow that only consists of a shift** (and unit scaling) $0.5\cdot(x-\mu)^2 = \ln(p(x))$

Conditional PDFs .. parameters of flow are output of a neural network

Conditional affine flow



Conditional general flow



- Instead of flow parameters, one optimizes NN parameters
- Conditional normalizing flow shows that **MSE loss comes from conditional Flow that only consists of a shift** (and unit scaling) $0.5\cdot(x-\mu)^2=\ln(p(x))$
- The process of **predicting parameters by a neural network** is also called "**amortization**"

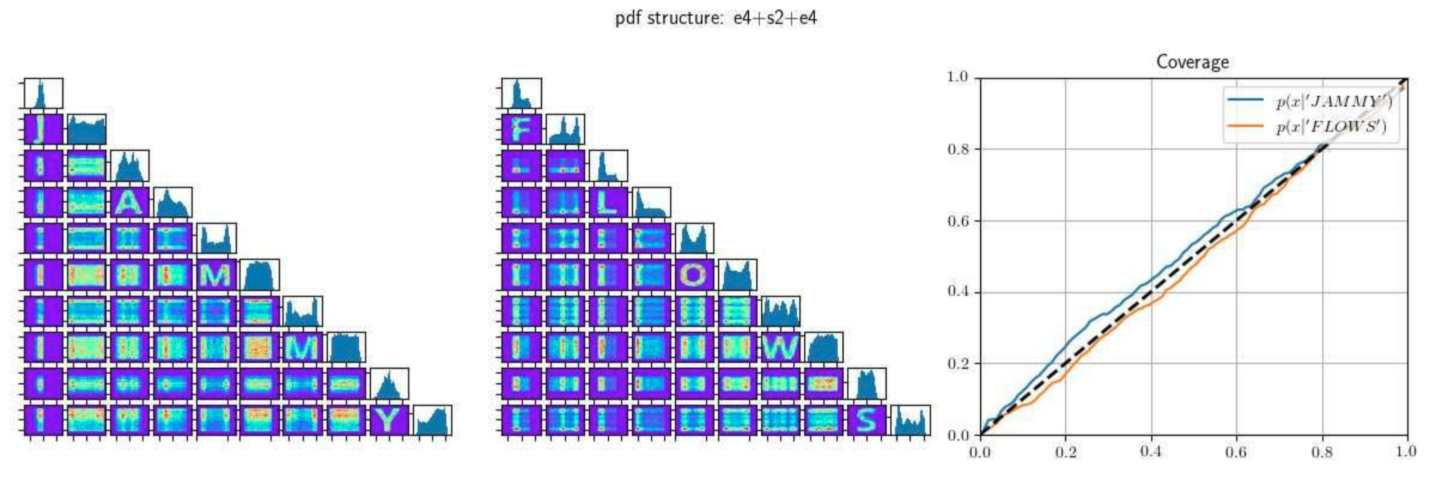
A package for flows on manifolds: jammy_flows (https://github.com/thoglu/jammy_flows)

import jammy_flows

pdf=jammy_flows.pdf("e4+s2+e4", "gggg+n+gggg")

pdf.sample(nsamples=1000)

A package to describe amortized (conditional) normalizing-flow PDFs defined jointly on tensor products of manifolds with coverage control. The connection between different manifolds is fixed via an autoregressive structure.

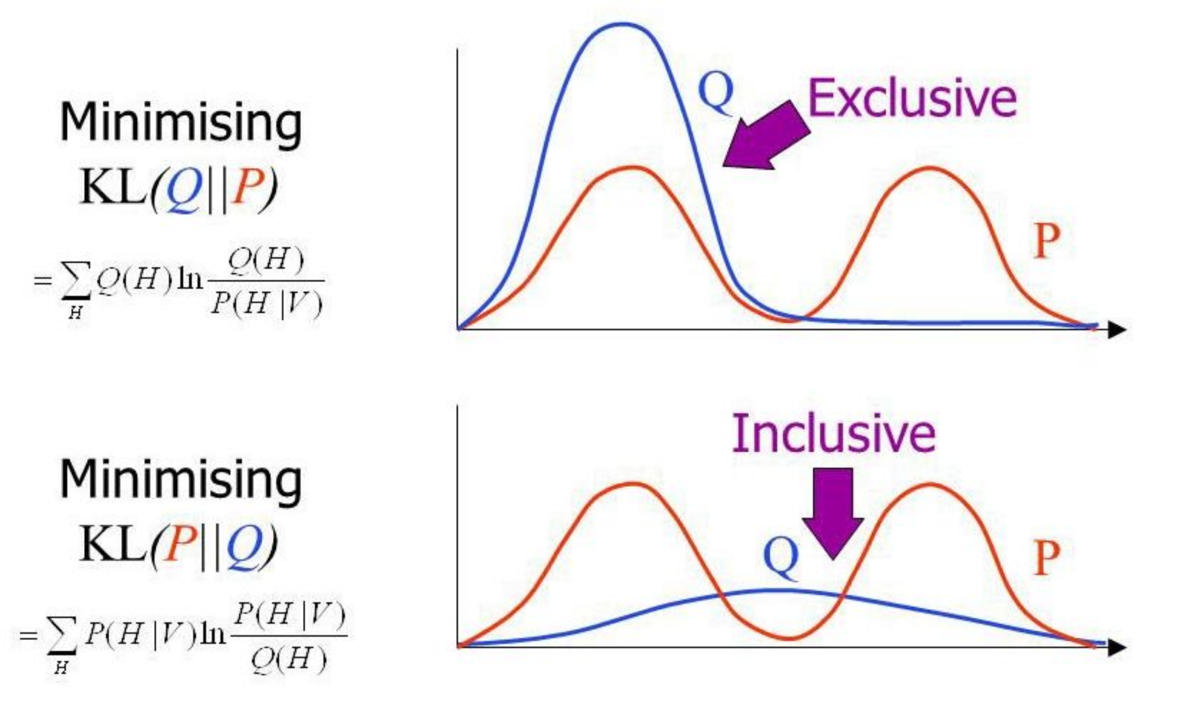


Hands on 1

https://colab.research.google.com/drive/1ySGCjEeVdKhCodwPxe_sw0bBoAymRe4o?usp=sharing

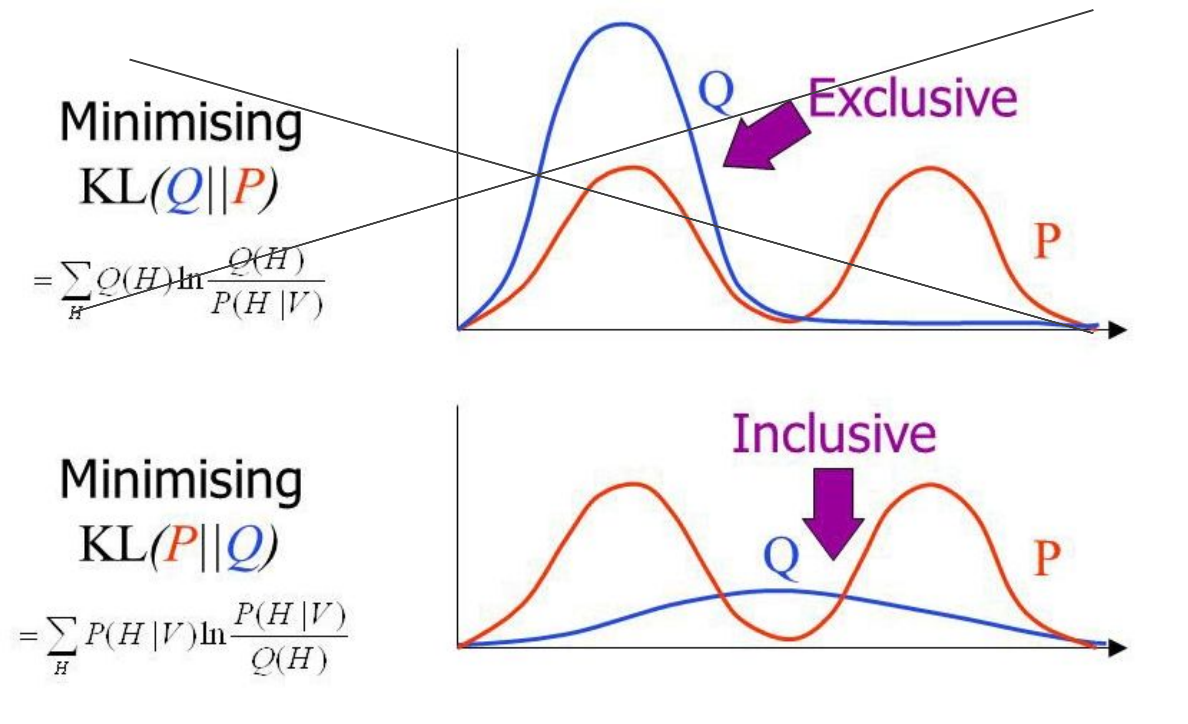
KL-divergence introduction

“Inclusive” KL-diverengce vs “exclusive” KL-divergence



KL-divergence introduction

"Inclusive" KL-diverengce vs "exclusive" KL-divergence



In most settings:

P = "True" PDF
(not accessible,
"Nature",
Samples from MC simulation
Draw from P)

Q = "Approximating PDF",
Parametrized by us,
"Surrogate model"

Supervised learning as (inclusive) KL-divergence minimization

What is a Monte Carlo simulation?
Samples from some “true” distribution

Computer scientists call this
“conditional ML objective”
Of supervised learning

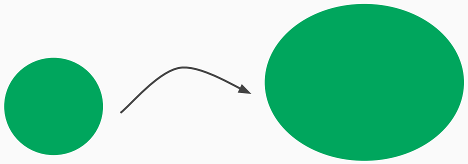
Physicists should call it
“variational inference objective”
For the **variational Posterior**

Supervised learning as (inclusive) KL-divergence minimization

What is a Monte Carlo simulation?
Samples from some “true” distribution

Sample from “systematics distribution” during MC generation

Computer scientists call this
“conditional ML objective”
Of supervised learning



Including systematics is trivial!

Physicists should call it
“variational inference objective”
For the **variational Posterior**

Deep learning generalizes classical statistical approaches

"Bayesian"

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}$$

MCMC

"Frequentist"

Maximum likelihood principle

Deep learning generalizes classical statistical approaches

KL-Div posterior

"Bayesian"

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}$$

(90's)

MCMC

Variational inference

KL-Div likelihood

"Frequentist"

(Aikake 70's)

Maximum likelihood principle

Deep learning generalizes classical statistical approaches

KL-Div posterior

Joint KL
(deep learning)

KL-Div likelihood

"Bayesian"

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}$$

(90's)

"Frequentist"

(Aikake 70's)

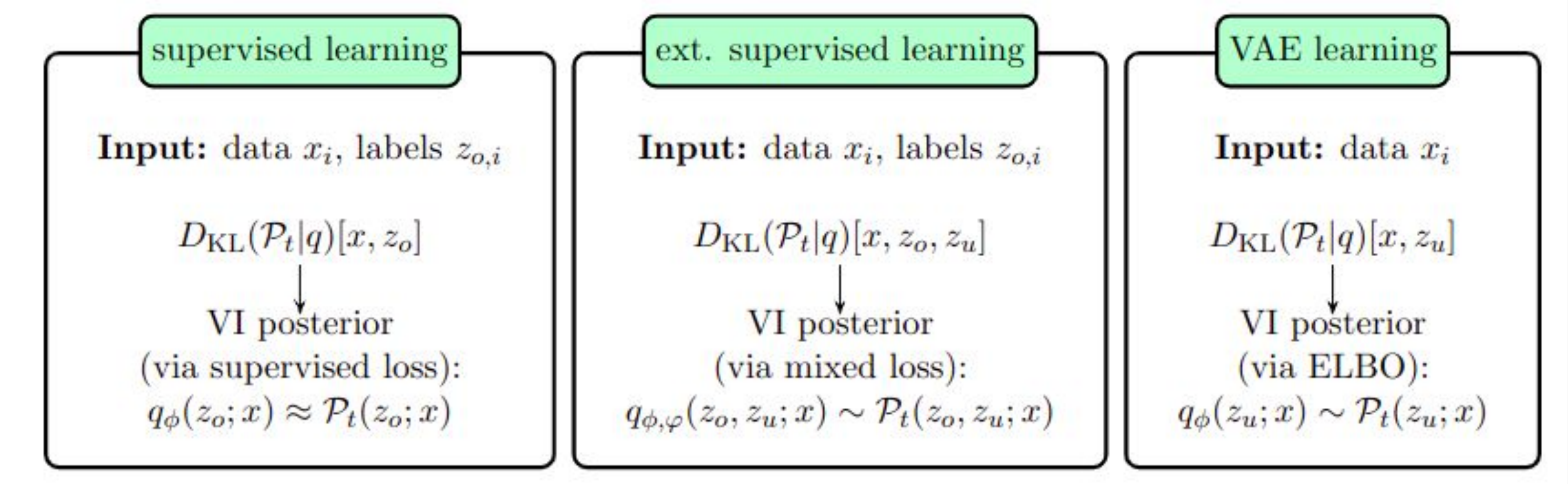
MCMC

Variational inference

Maximum likelihood principle

Probabilistic deep learning

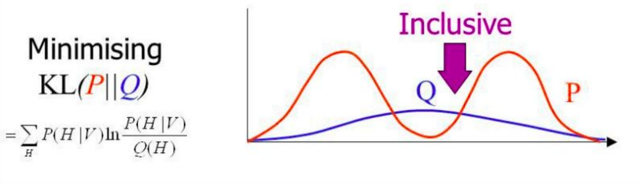
“All of deep learning is probability distribution matching”



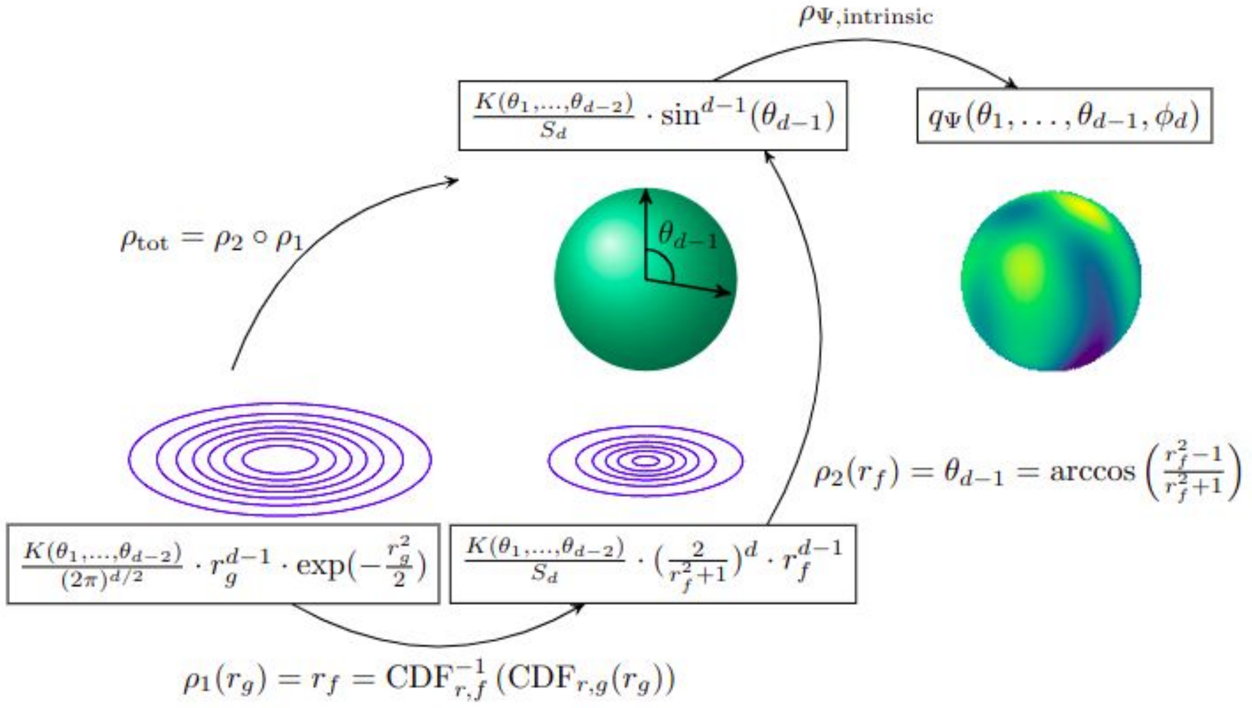
“Inklusive KL divergence”

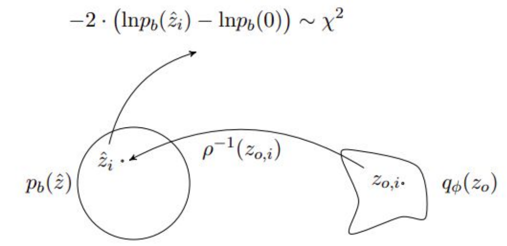
“exclusive KL divergence”

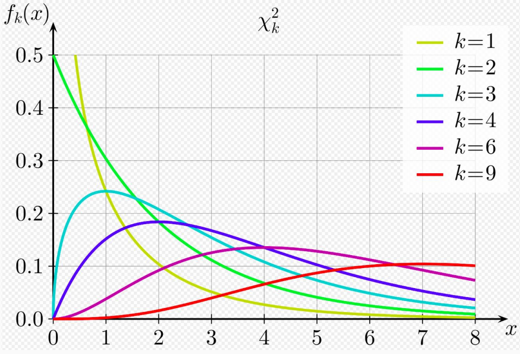
(2008.05825)



Coverage for NFs, including NFs on manifolds







(2008.05825)

Goodness of Fit

ext. supervised learning

Input: data x_i, labels $z_{o,i}$

$$D_{\mathrm{KL}}(\mathcal{P}_t|q)[x, z_o, z_u]$$

$$\downarrow$$

VI posterior
(via mixed loss):
$$q_{\phi,\varphi}(z_o, z_u; x) \sim \mathcal{P}_t(z_o, z_u; x)$$

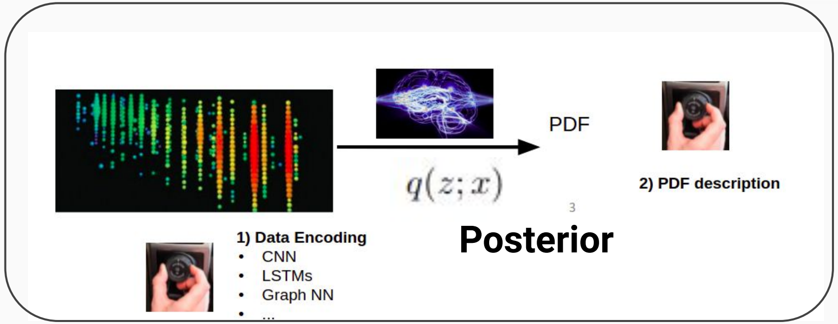
VAE learning

Input: data x_i

$$D_{\mathrm{KL}}(\mathcal{P}_t|q)[x, z_u]$$

$$\downarrow$$

VI posterior
(via ELBO):
$$q_{\phi}(z_u; x) \sim \mathcal{P}_t(z_u; x)$$



Likelihood $p(x; z)$

(2008.05825)

Goodness of Fit

ext. supervised learning

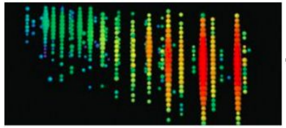
Input: data x_i, labels $z_{o,i}$

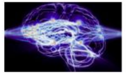
$$D_{\mathrm{KL}}(\mathcal{P}_t|q)[x, z_o, z_u]$$
$$\downarrow$$
VI posterior
(via mixed loss):
$$q_{\phi,\varphi}(z_o, z_u; x) \sim \mathcal{P}_t(z_o, z_u; x)$$

VAE learning

Input: data x_i

$$D_{\mathrm{KL}}(\mathcal{P}_t|q)[x, z_u]$$
$$\downarrow$$
VI posterior
(via ELBO):
$$q_\phi(z_u; x) \sim \mathcal{P}_t(z_u; x)$$





PDF

$q(z; x)$

2) PDF description

3

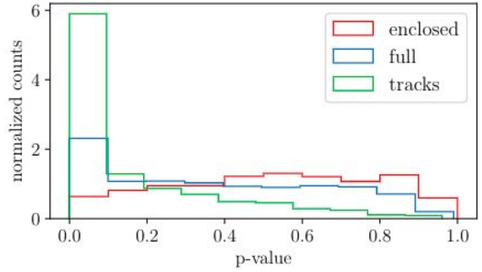
Posterior

1) Data Encoding
- CNN
- LSTMs
- Graph NN
- ...

Likelihood $p(x; z)$

$$p_{\mathrm{val}} = \int_{x,z} I_{T(x,z)>T(x_{\mathrm{obs}},z)} p_\theta(x; z) q_\phi(z; x_{\mathrm{obs}}) dx dz$$

$$T(x, z) = \ln p_\theta(x; z)/N_d$$



(2008.05825)

Hands on 2

1) Check out https://github.com/thoglu/jammy_flows.git
2) Go to /examples/ subdirectory
3) Check out the notebook with examples
4) Call script jammy_flows.py with suitable parameters