Generative Models

Claudius Krause
@ fkphys.uni-heidelberg.de

based on "Modern Machine Learning in LHC Physics"
by T. Plehn, A. Butter, B. Dillon, C. Krause
arXiv: soon

Rutgers Physics 694 physics.rutgers.edu/~dshih/694

Generative Models

problem: have a distribution $p_{truth}(x)$

& want to sample ("generate") new

$$x \sim p_{truth}(x)$$

p can be given - explicitly as function $\frac{dG}{G}$

- implicitly via a set of data points

$\{x\}$ defining $p_{data}(x)$ $(\simeq p_{truth}(x))$

applications in HEP

- event generation (parton, end-to-end)
- phase space
- calorimeter showers
- ...

this is a stochastic (random) process (RNG) ⇒ need random input

$z \sim p_{latent}(z) \quad \Rightarrow \quad x = f_\theta(z) \sim p_{model}(x) \approx p_{data}(x) \approx p_{truth}(x)$

↳ easy: Gaussian, uniform

these lectures: how to construct & train $f_\theta(z)$

VAEs GANs Normalizing Flows

How do we know, we have a good $f_\theta(z)$?

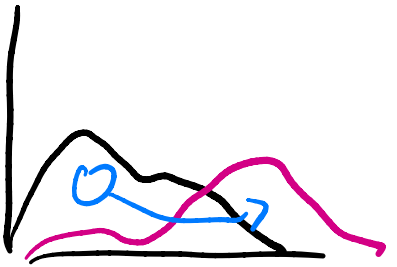
if we have $p_{model} \approx p_{data/truth}$ explicitly : Kullback-Leibler (KL) divergence

1) $D_{KL}[p_{data}, p_{model}] = \int dx\, p_{data}(x) \log \frac{p_{data}}{p_{model}}$

$= \left\langle \log \frac{p_{data}}{p_{model}} \right\rangle_{data}$

2) Earth Mover's Distance (Wasserstein metric)

works for continuum (p known)
or discrete (only samples available)

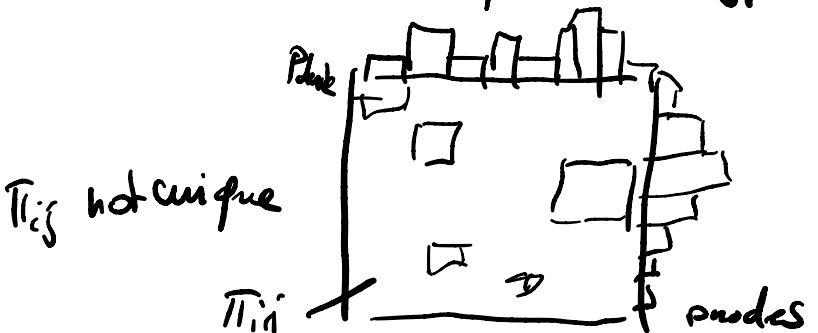


discrete p

$p_{data\,i}$ $\quad i \in [1, N_1]$

$p_{model\,j}$ $\quad j \in [1, N_2]$

define transport strategy π_{ij} as a matrix relating the two sets



$$p_{data\,i} = \frac{1}{N_2} \sum_j \pi_{ij}$$

$$p_{model\,j} = \frac{1}{N_1} \sum_i \pi_{ij}$$

$$W[p_{data}, p_{model}] = \min_{\pi} \frac{1}{N_1 N_2} \sum \pi_{ij} |x_i - y_j|$$

"amount of dirt" × distance

⇒ scales horribly w/ dimensionality

& is expensive to compute

3) generate events & look at single events (does a face look realistic? ↳ dataset dep.)

look at distributions ↳ (always a projection)

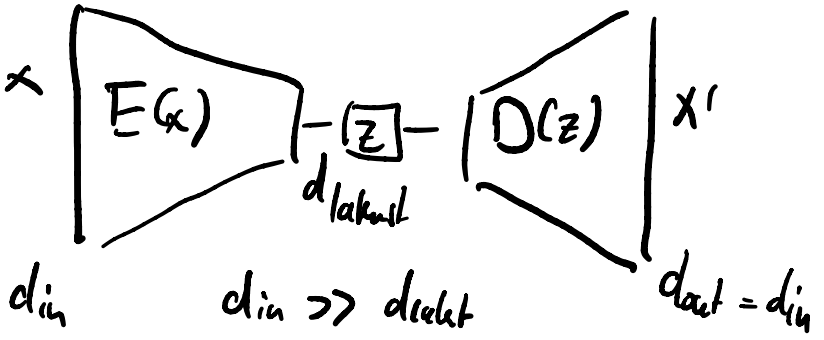
4) Classifier test [1610.06545] ⇒ train NN to distinguish data vs gen. samples

if $p_{correct} = 0.5$ (classifier is confused)

↳ $p_{data}(x) = p_{model}(x)$

1/ Variational Autoencoder VAE 13.12.6114

start w/ AE part first



loss: MSE

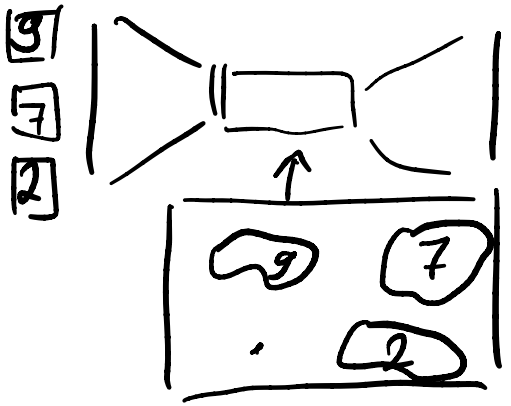
$$\mathcal{L}_{MSE} = \frac{1}{N} \sum_{i=1}^{N} |x_i - x_i'|^2$$

$x_i' = D(E(x))$

- compress information ("lossy") → will learn most useful latent representation of data
- generative model: sampling from z & use D(z)

Bad: latent space not regularized

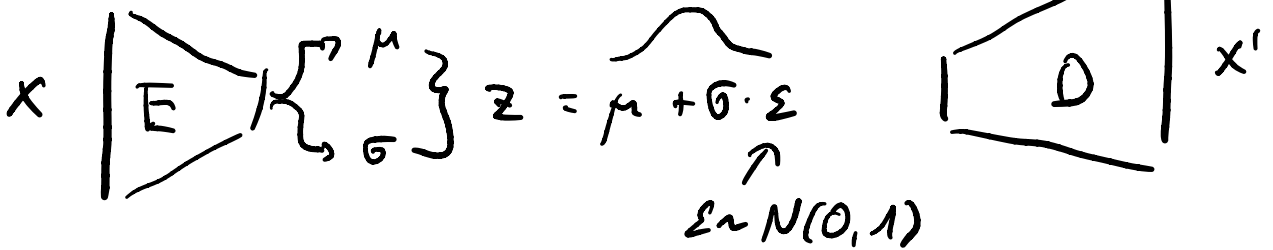
- what distribution to sample from ?
- interpolation -> gaps



$z \sim E_\theta(z|x)$

$x' \sim D_\theta(x'|z)$

VAE give shape to latent space



given data x, we want to know / infer z of the gen. model

$$D(z|x) = \frac{D(x|z) \cdot P_z(z)}{P_{data}(x)}$$

VAE: approximate $D(z|x)$ by $E(z|x)$ (variational inference)

$$D_{KL}[E(z|x), D(z|x)] = \left\langle \log \frac{E(z|x)}{D(z|x)} \right\rangle_{E(z|x)} =$$

$$= \left\langle \log E(z|x) - \log \frac{D(x|z)\, P_z(z)}{P_{data}(x)} \right\rangle_{E(z|x)}$$

$$= \underbrace{\left\langle -\log D(x|z) \right\rangle_{E(z|x)} + D_{KL}[E(z|x), P_z(z)]}_{-ELBO \rightarrow \text{bounds LL}} + \log P_{data}$$

$$\equiv \mathcal{L}_{MSE} + \beta_{KL} D_{KL}[E(z|x), N(0,1)]$$

if Gaussian dist: MSE reconstruction

regularizes the latent space to be Gaussian

latent space 

1 1 7 7 3 8

Sampling: from $N(0,1)$ & then passing thru $D(z)$

For practical applications : we need a conditional VAE

- condition (discrete or continuous) needs to be given to E & D

$$D(z, c) \text{ generates } x \text{ for a given } c$$

pros:
- easy to setup / train
- explicit map to the latent space

cons
- rarely state of the art
- no optimality guarantee

[2] Generative Adversarial Networks (GAN) 1406.2661

train 2 adversarial NN : generator $G : z \to x$

z ↓ unif.-/gauss

discriminator / critic $D(\ ) = \begin{cases} 0 & \text{generated data} \\ 1 & \text{true data} \end{cases}$

$D(x)$ / $D(G(z))$

train D by minimizing $\langle 1 - D(x) \rangle_{x \sim P_{true}}$ & $\langle D(x) \rangle_{x \sim P_{gen}}$

enhance sensitivity $1 - 0 \to -\log D$

$D \to -\log(1-D)$

$$\mathcal{L}_D = \langle -\log D \rangle_{P_{true}} + \langle -\log(1-D) \rangle_{P_{gen}}$$ BCE

train G by making it generate samples that look like real data

$$L_G = \langle -\log D \rangle_{p_{gen}} \sim \langle \log(1-D) \rangle_{p_{gen}}$$

↓ same minimum, better gradients

together:

$$\min_G \max_D \langle \log D(x) \rangle_{p_{true}} + \log\left(1 - D(G(z))\right)_{p_z}$$

G ↘ fools D

D ↘ is a good classifier

Optimal point: Nash equilibrium

$$D_{opt} = \frac{P_{true/data}}{P_{true/data} + P_{gen}}$$

$$\mathcal{L}_D \rightarrow - \left\langle \log \frac{P_{data}}{P_{data} + P_{gen}} \right\rangle_{P_{data}} - \left\langle \log \frac{P_{gen}}{P_{data} + P_{gen}} \right\rangle_{P_{gen}}$$

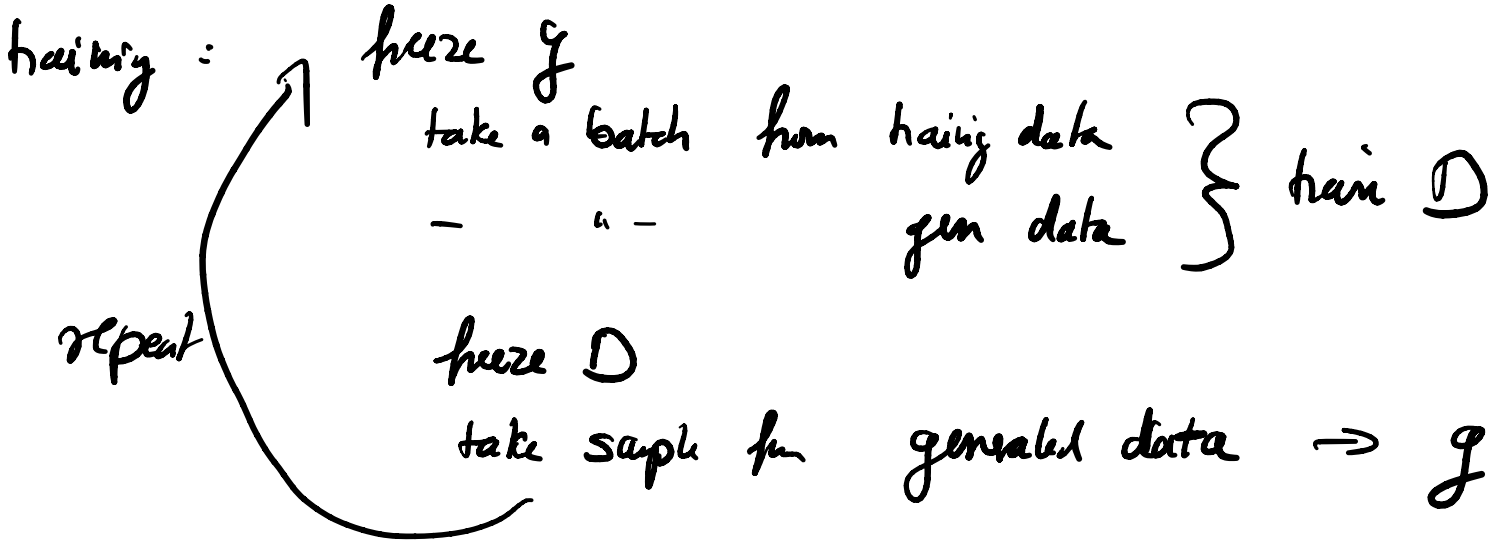
$$= - D_{KL}\left(P_{data}, P_{data} + P_{gen}\right) - D_{KL}\left(P_{gen}, P_{data} + P_{gen}\right)$$

$$= - 2 D_{JS}\left[P_{data}, P_{gen}\right] + 2 \log 2$$

$$\mathcal{L}_g \rightarrow \left\langle \log (1 - D_{opt}) \right\rangle_{P_{gen}}$$

$$= D_{KL}\left[P_{gen}, \frac{P_{data} + P_{gen}}{2}\right] - \log 2$$

↳ minimal for $P_{gen} = P_{data}$


training :
freeze G
take a batch from training data
— " — gen data
} train D
repeat
freeze D
take sample from generated data → G

Problems w/ GANs

- convergence — min-max — unstable
  - vanishing gradients from D
    - ↳ by regularization — 1705.09367, 1801.04406
- metrics for success
- mode collapse — GAN gets stuck in vicious cycle

G learns to produce realistic 1
D will be fooled
then D will learn 2's are real
then G will focus on 2's

Ways to improve GANs

→ replace discriminator by critic

D learn Wasserstein distance between P_{data} & P_{gen}

WGAN 1701.07875

GAN

pro: great, realistic single images

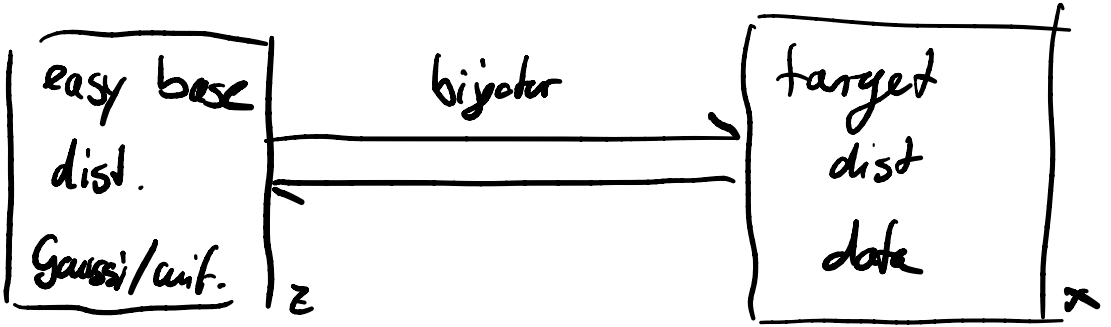
con: distributions only @ 𝒪(10%)

3| Normalizing Flows

VAEs & GANs learn implicitly (only access to samples, not $p(x)$)

Flows learn $p(x)$ explicitly

key idea: learn complicated coordinate transformation



$$\dim z = \dim x$$

for coordinate transform $\vec{x} = f(\vec{z})$

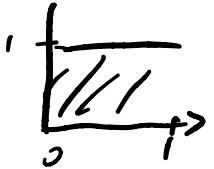
$$p(x) = \pi(z) \left| \det \frac{df(z)}{dz} \right|^{-1}$$

$$= \pi(f^{-1}(x)) \left| \det \frac{\partial f^{-1}(x)}{\partial x} \right|$$

→ need Jacobian determinant

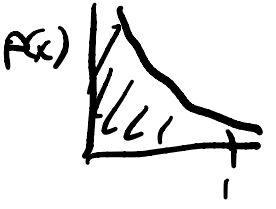
→ need inverse $\vec{z} = f^{-1}(\vec{x})$

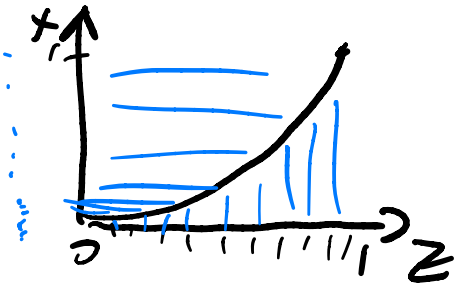
$\pi(z)$ uniform



$$x = f(z) = z^2$$

$$p(x) = 1 \cdot |2z|^{-1} = \frac{1}{2\sqrt{x}}$$





bijector works in both ways

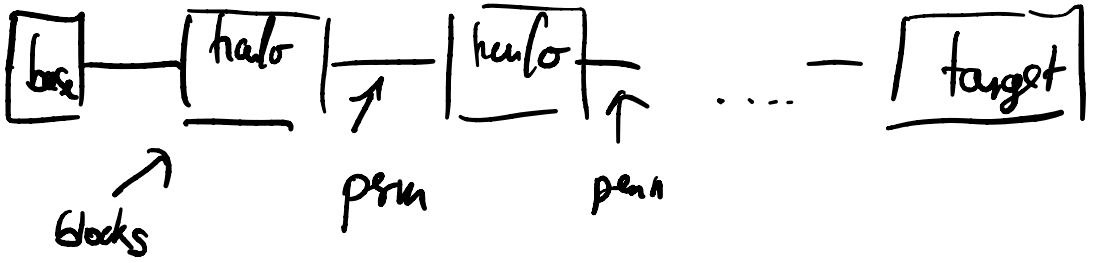
$\pi(z)$ —— sampling ——→ $p(x)$

←—— density estimator ——

first idea: $f(z) = NN(z)$

does not work well
- invert a NN ?
- Jacobian ~ $O(n^3)$

Flows avoid these problems by learning parameters of a series of invertible, pre-defined transformations (w/ tractable Jacobian)



chain of bijectors is a bijector

1505.05770

$$x = C_K \circ C_{K-1} \circ C_{K-2} \cdots C_1(z_0) \qquad z_0 \sim \pi_0(z_0)$$

$$p(x) = \pi_0(z_0) \prod_{k=1}^{K} \left| \det \frac{\partial C_k}{\partial z_{k-1}} \right|^{-1}$$

training - max $\log p$ (or min. NLL)

- use D_{KL} to fit to (normalized) target fractions

$\Rightarrow$ more stable than GANs, better performance than VAEs

in more detail

inside each block, assume the halo fractions

$$\vec{c}(\vec{z}, \vec{\mu}) = \left(c_1(z_1, \vec{\mu}_1), c_2(z_2, \vec{\mu}_2), \dots c_n(z_n, \vec{\mu}_n)\right)^T$$

$\vec{\mu}_i$ still depend on z_j $(i \neq j)$ to capture correlations

there are 2 main architectures to tame Jacobians

↳ $\mathcal{O}(n^3) \Rightarrow \mathcal{O}(n)$

matrix is triangular

① autoregressive models

$\vec{\mu}_1 = \text{const.}$ $\quad \vec{\mu}_2(z_1)$ $\quad \vec{\mu}_3 = \vec{\mu}_3(z_1, z_2)$ $\quad \vec{\mu}_i(z_1 \dots z_{i-1})$

$p(x_1)$ $\quad p(x_2|x_1)$ $\quad p(x_3|x_2,x_1)$ $\dots$

$$p(x) = \prod_{i=1}^{n} p(x_i|x_1 \dots x_{i-1})$$

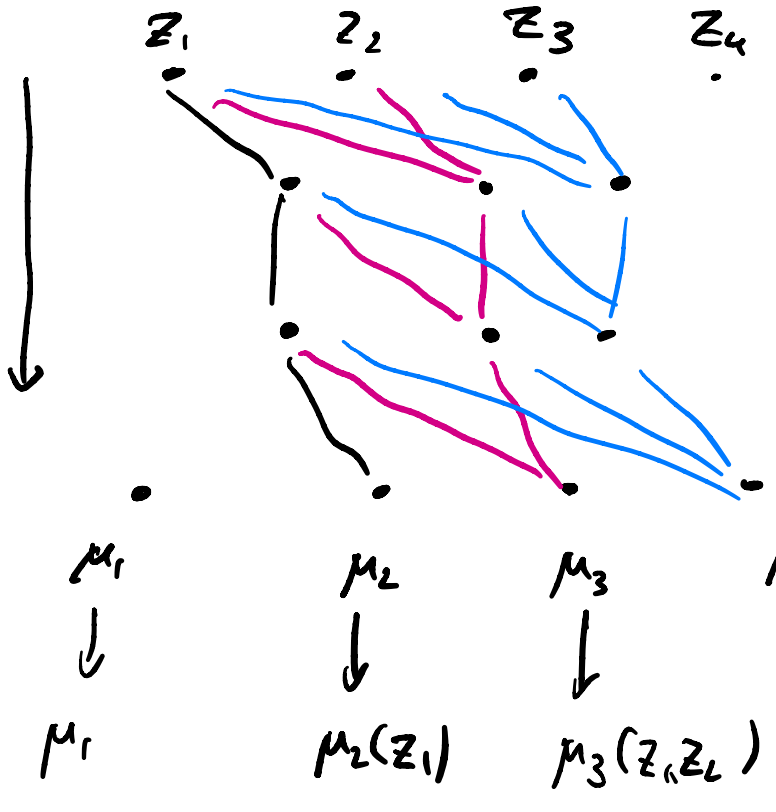
$$x_i = c_i(z_{1:i})$$

$$J = \begin{pmatrix} \diagdown & 0 \\ * & \diagdown \end{pmatrix} \quad \mathcal{O}(n)$$

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

NN realization : MADE (masked Autoencoder for Density Estimation)

1502.03509



forward: single call gives all μ_s fast

inverse loop n-times slow to get all z_s

fast	inference $p(x)$	Sampling
slow	Sampling	inference $p(x)$

MAF

1705.07057

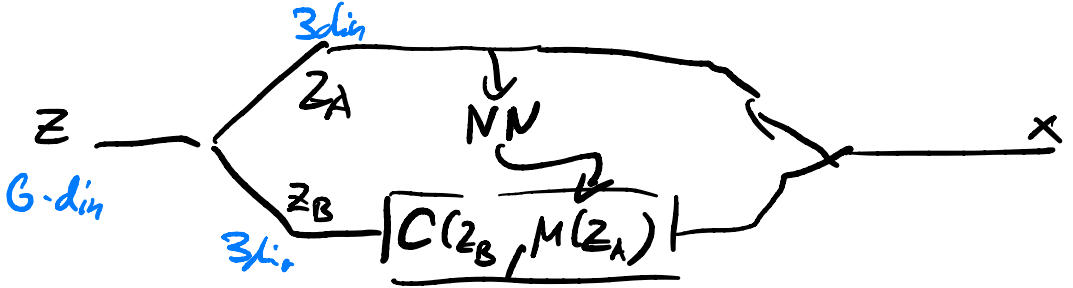
IAF

1606.04934

② Coupling Layer based flows (bipartit) INN

Real NVP 1605.08803

Split z in 2 sets z_A z_B



forward $x_A = z_A$

equally fast in both directions

$x_B = C_B(z_B, \mu(z_A))$

inverse $z_A = x_A$ ($= x_A$)

$z_B = C^{-1}(x_B, \mu(z_A))$

$$|\det J| = \begin{vmatrix} 1 & 0 \\ \frac{\partial C_B}{\partial z_A} & \frac{\partial C_B}{\partial z_B} \end{vmatrix}$$

diagonal

how about $C(z, \vec{\mu})$

↳ detailed choice depends on domain

base gaussian $(-\infty, \infty)$ → logit

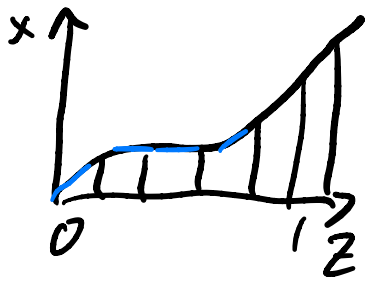
uniform $(0, 1)$ → sigmoid

affine coupling layer $C(z, \underbrace{\vec{\mu}}_{s, t}) = z \odot \exp(s) + t = x$

inverse : $(x-t) \odot \exp(-s) = z$

$|\det J| = \exp(\Sigma s)$

piecewise splines (monotonic)



- linear, quadric 1808.03856
- Rational Quadratic Splines 1906.04032

$$C = \frac{a_2 \alpha^2 + a_1 \alpha + a_0}{b_2 \alpha^2 + b_1 \alpha + b_0}$$

α ... location inside bin

a_i, b_i given by NN