[Paper review 27]

Dennsity Estimation using Real NVP

(Laurent Dinh, et al., 2017)

[Contents]

1. Abstract

real NVP = real-valued Non-Volumne Preserving transformation

- powerful, stably invertible, learnable transformations
- resulting in
    - exact log-likelihood computation
    - exact and efficient sampling
    - exact and efficient inference of latent variables
    - interpretable latent space

2. Introduction

representation learning

- have recently advanced thanks to supervised learning techniques
- unsupervised learning can also help!

one principle approach to unsupervised learning : "generative probabilistic modeling"

$\rightarrow$ but problem in high dimension real NVP : tractable yet expressive approach to model high-dimensional data!

3. Related Works

lots of works (based on "generative probabilistic modeling") have focused on maximum likelihood

- probabilistic undirected graphs

  (ex. RBM, DBM $\rightarrow$ due to intractability, used approximation like Mean Field Inference and MCMC)
- directed graphical models

4. Model definition

in this paper...

- learn highly nonlinear models in high-dimensional continuous spaces throguh ML
- use more flexible class of architectures (using the change of variable formula)

4.1 change of variable formula

$$\begin{aligned} p_X(x) &= p_Z(f(x)) \left|\det\left(\frac{\partial f(x)}{\partial x^T}\right)\right| \\ \log(p_X(x)) &= \log(p_Z(f(x))) + \log\left(\left|\det\left(\frac{\partial f(x)}{\partial x^T}\right)\right|\right) \end{aligned}$$

4.2 Coupling layers

computing the Jacobian with high-dimensional domain & codomain : very expensive!

$\rightarrow$ "triangular matrix" (both tractable and flexible)

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

4.3 Properties

Jacobian of transformation :

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

we can efficiently compute its determinant as ...$\exp\left[\sum_j s\left(x_{1:d}\right)_j\right]$.

computing the inverse is no more complex than forward propagation!

(= meaning that sampling is as efficient as inference)

$$\begin{cases} y_{1:d} & = x_{1:d} \\ y_{d+1:D} & = x_{d+1:D} \odot \exp(s\left(x_{1:d}\right)) + t\left(x_{1:d}\right) \end{cases}$$
$$\Leftrightarrow \begin{cases} x_{1:d} = y_{1:d} \\ x_{d+1:D} = \left(y_{d+1:D} - t\left(y_{1:d}\right)\right) \odot \exp(-s\left(y_{1:d}\right)) \end{cases}$$

4.4 Masked Convolution

$y = b \odot x + (1-b) \odot \left(x \odot \exp(s(b \odot x)) + t(b \odot x)\right)$

- partitioning using "binary mask b "
  - 1 for the first half
  - 0 for the second half
- $s(\cdot)$ and $t(\cdot)$ are rectified CNN

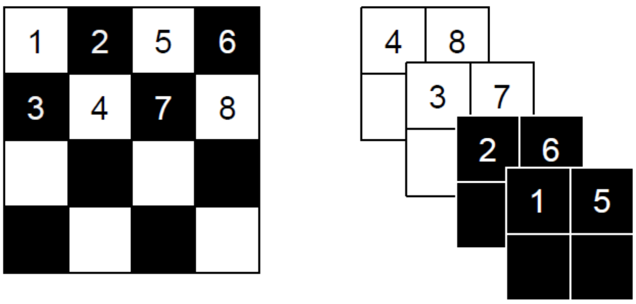


Figure 3: Masking schemes for affine coupling layers. On the left, a spatial checkerboard pattern mask. On the right, a channel-wise masking. The squeezing operation reduces the $4 \times 4 \times 1$ tensor (on the left) into a $2 \times 2 \times 4$ tensor (on the right). Before the squeezing operation, a checkerboard pattern is used for coupling layers while a channel-wise masking pattern is used afterward.

4.5 Combining coupling layers

forward transformation leaves some components unchanged...

$\rightarrow$ can be overcome by "composing coupling layers"! still tractable

$$\frac{\partial\left(f_b \circ f_a\right)}{\partial x_a^T}\left(x_a\right) = \frac{\partial f_a}{\partial x_a^T}\left(x_a\right) \cdot \frac{\partial f_b}{\partial x_b^T}\left(x_b = f_a\left(x_a\right)\right)$$
$$\det(A \cdot B) = \det(A)\det(B)$$

inverse : $\left(f_b \circ f_a\right)^{-1} = f_a^{-1} \circ f_b^{-1}$

4.6 Multi-scale Architecture

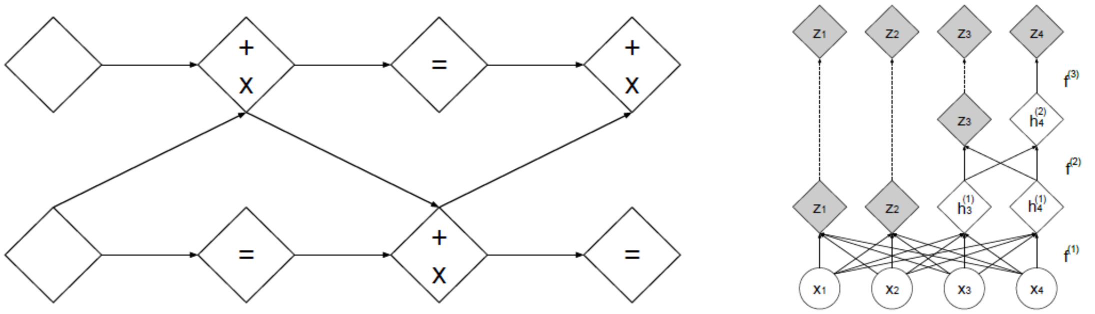
by using "squeezing operation"

for each channel...

- $s \times s \times c \rightarrow \frac{s}{2} \times \frac{s}{2} \times 4c$
- effectively trading "spatial size" for "number of channels"

Sequence of "coupling-squeezing-coupling"

$$
\begin{aligned}
h^{(0)} &= x \\
\left(z^{(i+1)}, h^{(i+1)}\right) &= f^{(i+1)}\left(h^{(i)}\right) \\
z^{(L)} &= f^{(L)}\left(h^{(L-1)}\right) \\
z &= \left(z^{(1)}, \ldots, z^{(L)}\right)
\end{aligned}
$$



(a) In this alternating pattern, units which remain identical in one transformation are modified in the next.

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

4.7 Batch Normalization

use deep Resnet & BN & WN

$x \mapsto \frac{x-\tilde{\mu}}{\sqrt{\tilde{\sigma}^2+\epsilon}}$, has a Jacobian matrix $\left(\prod_i \left(\tilde{\sigma}_i^2+\epsilon\right)\right)^{-\frac{1}{2}}$