NORMALIZING FLOWS

Oliver Schulte

Simon Fraser University

CMPT 728

Introduction to Deep Learning

OVERVIEW AND MAIN IDEA

EXAMPLES: CONDITIONAL GENERATION

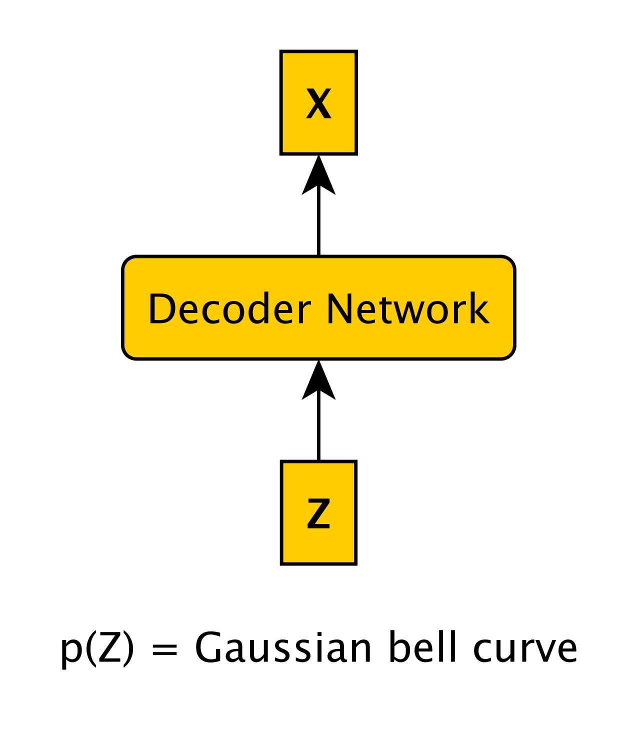
conditional generative models

courtesy: wimbledon

courtesy: 9GAG

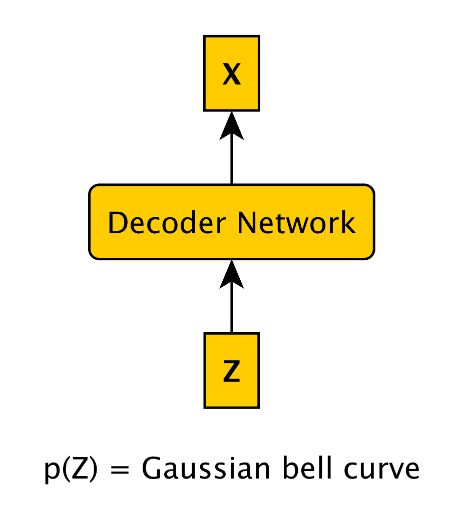
RECAP GENERATIVE MODELLING: NEURAL NETWORK APPROACH

- Input random vector **z** to neural net
  - Typically from a Gaussian bell curve distribution
- Network maps **z** to output vector **x**
- Intuitively, the output should be like the observations **x**

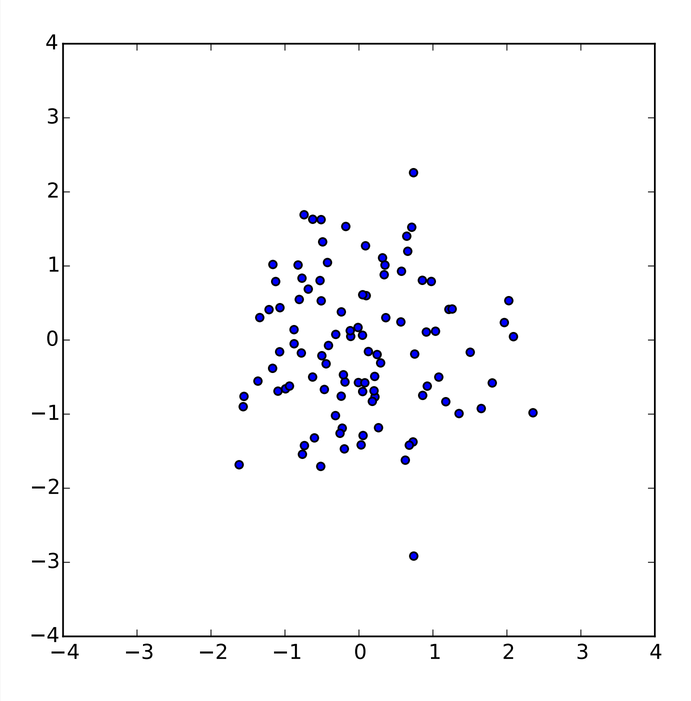


EXACT LOG-LIKELIHOOD

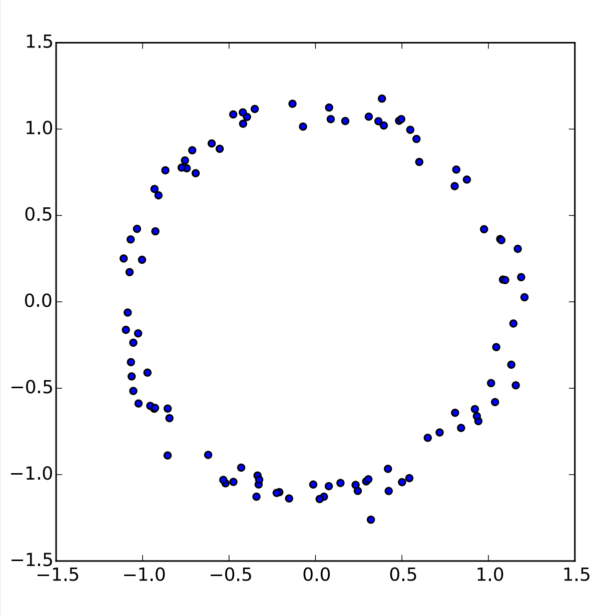
- Loss function for input $\mathbf{x}$ = - ln(P($\mathbf{x}$))
  - The negative log-likelihood is the standard loss for generative models
- In the decoder architecture
  $P(\mathbf{x}) \approx \int_{\mathbf{z}} I(f(\mathbf{z}) = \mathbf{x})\, p(\mathbf{z})\, d\mathbf{z}$
  - $I(f(\mathbf{z}) = \mathbf{x})$ returns 1 if the decoder maps random $\mathbf{z}$ to $\mathbf{x}$, 0 o.w.
- Integral is intractable
- VAE approach: approximate integral
- Normalizing flow: solve integral <u>exactly</u> assuming that f is <u>invertible</u>



GAUSSIAN EXAMPLE



$f(z) = z/10 + z/||z||$



Samples from Gaussian distribution of 2D P(**Z**)

Samples from output distribution P(**X**)

- Since f is invertible, for each output **x**, there is a unique $f^{-1}(z)$ that generates it.

PROS AND CONS

+ We can compute the log-likelihood of a data point exactly
  + See below for more details
- Source space Z and target space X must have some dimensionality → no dimensionality reduction/encoding
  - Can be addressed to some extent

COMPUTING THE LIKELIHOOD FOR INVERTIBLE TRANSFORMATIONS

The change of variable formula

INVERTING PROBABILITY DISTRIBUTIONS

- What is the right way to compute the probability of a data point $x=f(z)$? I.e.,
- Want $q(x)=q(f(z))$ given $p(z)$
- First try: $q^*(x) = p(z) = p(f^{-1}(x))$
- Does not quite work because the $q^*(x)$ numbers don't add up to 1.
  - Need $\int_x q(x)\, dx = 1$, can fix by setting $q(x) = p(f^{-1}(x))/ C$ for suitable constant C
- But how to compute C?

CHANGE OF VARIABLES FORMULA

Theorem Let

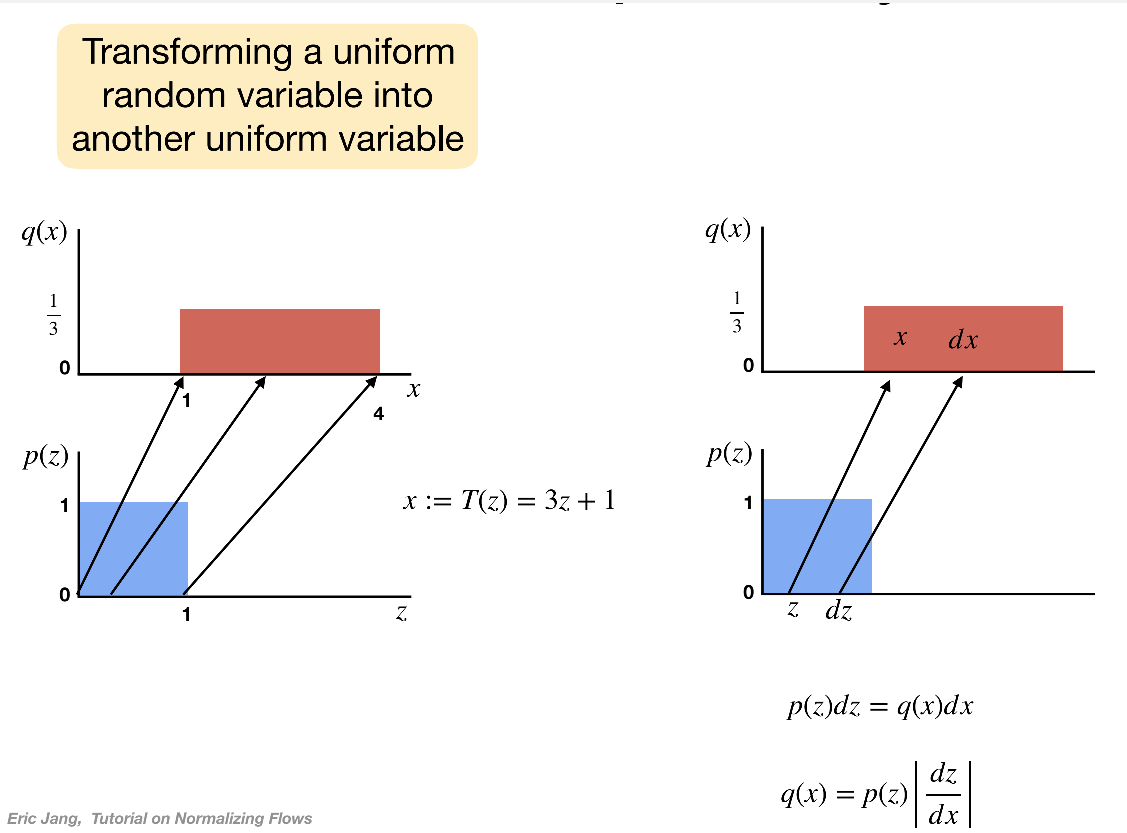
- Z be a source univariate variable with density function p(z)
- $T: Z \rightarrow X$ be an invertible transformation of z-values to target x-values
- Then $q(x) = p(T^{-1}(x)) \, |dx/dz|^{-1}$

CHANGE OF MULTIPLE VARIABLES FORMULA

Theorem Let

- $\mathbf{Z}$ be a source vector variable with density function $p(\mathbf{Z})$
- $T: \mathbf{Z} \rightarrow \mathbf{X}$ be an invertible transformation of **z**-values to target **x**-values
- Then $q(\mathbf{x}) = p(T^{-1}(\mathbf{x})) \, |\det(\nabla_{\mathbf{Z}} \mathbf{T}(\mathbf{z}))|^{-1}$
  where $\det(\nabla_{\mathbf{Z}} \mathbf{T}(\mathbf{z}))$ is the Jacobian of the transformation

ILLUSTRATION



video

LEARNING ISSUES

LEARNING INVERTIBLE TRANSFORMATIONS

- Maximize the exact log-likelihood
- Must be able to compute
  - Jacobian T
  - Determinant of Jacobian $\nabla_{\mathbf{z}}\mathbf{T}$
  - Inverse of **T**
    - Since $\mathbf{z}_i = T^{-1}(\mathbf{x}_i)$

Given: dataset $\mathscr{D} := \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \ldots, \mathbf{x}_n\} \sim q(\mathbf{x})$

learn the density $q(\mathbf{x})$

choose a simple source density $p(\mathbf{z})$

use maximum likelihood

$$\prod_{i=1}^{n} q(\mathbf{x}_i) = \prod_{i=1}^{n} p(\mathbf{z}_i) \left| det\left(\nabla_{\mathbf{z}} \mathbf{T}(\mathbf{z}_i) \right) \right|^{-1}$$

$$\hat{\mathbf{T}} := \arg\max_{\mathbf{T}} \prod_{i=1}^{n} p(\mathbf{z}_i) \left| det\left(\nabla_{\mathbf{z}} \mathbf{T}(\mathbf{z}_i) \right) \right|^{-1}$$

$$\hat{\mathbf{T}} := \arg\max_{\mathbf{T}} \sum_{i=1}^{n} \log p(\mathbf{z}_i) - \log \left| det\left(\nabla_{\mathbf{z}} \mathbf{T}(\mathbf{z}_i) \right) \right|$$

TRIANGULAR INCREASING MAPS

- Typically learn a sequence of invertible transformations $T_1, .., T_k$
  - Increases expressive power
- Triangular increasing maps learn a sequence where each transformation uses one more input variable z_i

➢ Makes the Jacobian and determinants easy to compute

ILLUSTRATION OF INCREASING TRIANGULAR MAPS

$$\mathbf{T} : \mathbb{R}^d \rightarrow \mathbb{R}^d$$

$$x_1 = T_1(z_1)$$
$$x_2 = T_2(z_1, z_2)$$
$$x_3 = T_3(z_1, z_2, z_3)$$
$$\vdots$$
$$x_d = T_d(z_1, z_2, z_3, \ \ldots, z_d)$$

$$\nabla_{\mathbf{z}} \mathbf{T} = \begin{bmatrix} \frac{\partial T_1}{\partial z_1} & 0 & \cdots & 0 \\ \frac{\partial T_2}{\partial z_1} & \frac{\partial T_2}{\partial z_2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial T_d}{\partial z_1} & \frac{\partial T_d}{\partial z_2} & \cdots & \frac{\partial T_d}{\partial z_d} \end{bmatrix}$$

triangular : T_j **is a function of** $z_1, z_2, \ldots, z_j$

increasing : T_j **is increasing w.r.t** z_j → $\frac{\partial T_j}{\partial z_j} > 0$

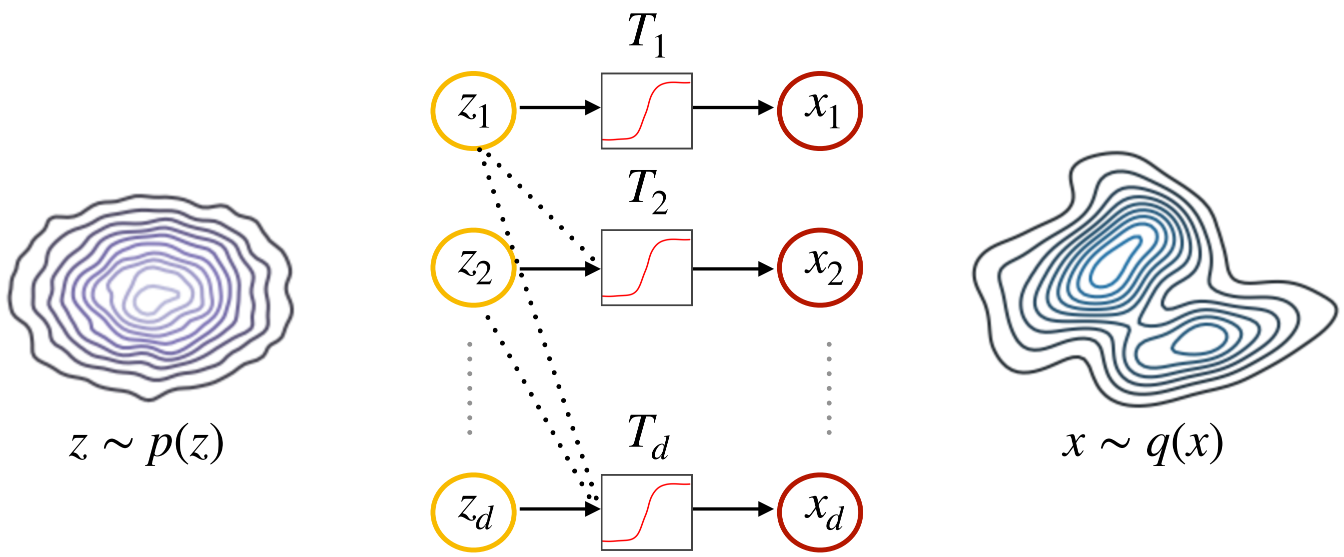
triangular maps

inverse and Jacobian are *easy* to compute

TRIANGULAR MAPS ARE UNIVERSAL

- **Theorem** (Paraphrase) For a fixed variable ordering $z_1, z_2, \ldots, z_d$ there always exists a unique increasing triangular map T: $\mathbf{Z} \rightarrow \mathbf{X}$ that transforms a source density p($\mathbf{z}$) to a target density q($\mathbf{x}$)
- Bogachev, V. et. al. Triangular Transformation of Measures, Sbornik: Mathematics, 2005

BIG PICTURE



learn T by maximizing likelihood

$$\min_{T} \sum_{i=1}^{n} \left[-\log p\left(\mathbf{T}^{-1}(\mathbf{x}_i)\right) + \sum_{j} \log \partial_j T_j\left(\mathbf{T}^{-1}(\mathbf{x}_i)\right) \right]$$

Marzouk, Y. et.al. Sampling via Measure Transport: An Introduction, Springer, 2016

AUTO-REGRESSIVE FLOW MODELS

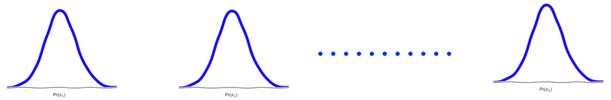
- Can always rewrite a joint density $q(\mathbf{x})$ as $q(\mathbf{x})=q(x_1) \times q(x_2|x_1) \times \ldots \times q(x_d|x_{<d})$
- Think about a "sequence" $x_1,x_2,\ldots,x_d$

➢ Learn a sequence of transformations $T_1,..,T_d$ s.t. Each T_i models the conditional density $q(x_i|x_{<i})$

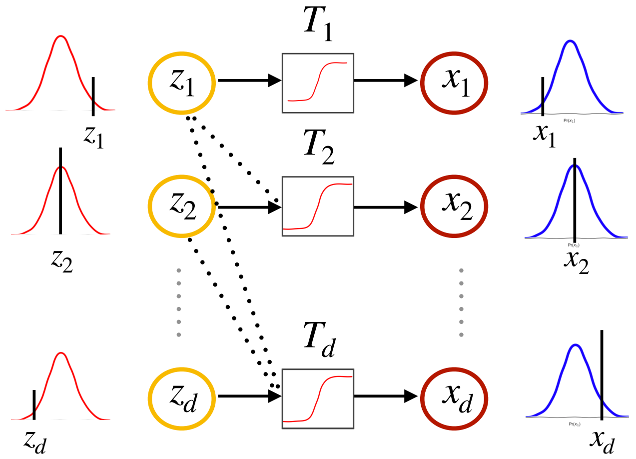
- E.g. with Gaussians like in a VAE

AUTO-REGRESSIVE FLOW WITH GAUSSIANS

$$q(x) = q_1(x_1) \cdot q_2(x_2 \,|\, x_1) \cdot \ldots \cdot q_d(x_d \,|\, x_{<d})$$



$\mathcal{N}(\mu_1, \sigma_1^2)$ $\mathcal{N}(\mu_2, \sigma_2^2)$ $\mathcal{N}(\mu_d, \sigma_d^2)$



$$x_1 = \sigma_1 \cdot z_1 + \mu_1 =: T_1(z_1)$$

$$x_2 = \sigma_2(z_1) \cdot z_2 + \mu_2(z_1) := T_2(z_2\,;z_1)$$

$$\vdots$$

$$x_d = \sigma_d(z_{<d}) \cdot z_d + \mu_d(z_{<d}) := T_d(z_d\,;z_{<d})$$

SUMMARY

- Normalizing flow key idea: map random inputs to generated outputs via an invertible function
- Can compute likelihood of observed inputs **x** exactly using change-of-variables theorem
- But need to compute for learned mapping 1) inverse 2) Jacobian 3) determinant of Jacobian
- This is possible if we use increasing triangular maps (without loss of expressive power)
- Demos