VOLGOGRAD STATE TECHNICAL UNIVERSITY

Машинное обучение и нейросетевые модели: нормализующие потоки

Лектор: Кравченя Павел Дмитриевич

Волгоград 2024

VOLGOGRAD STATE TECHNICAL UNIVERSITY

План лекции

1. Особенности моделей VAE и GAN. Идея нормализующих потоков.
2. Матрица Якоби и якобиан. Формула замены переменной.
3. Нормализующий поток. Архитектура нормализующего потока.
4. Плотность вероятности данных для комбинации отображений в потоке.
5. Поэлементно преобразующие нормализующие потоки.
6. Архитектура NICE. Понятие слоя связывания. Типы слоёв связывания.
7. Архитектура RealNVP. Организация слоёв связывания в RealNVP.
8. Архитектура Glow. Основные операции. Многослойная архитектура.
9. Модель авторегрессионных потоков. Обусловливающие функции.
10. Модель обратных авторегрессионных потоков. Особенности работы.

VOLGOGRAD STATE TECHNICAL UNIVERSITY

Особенности VAE и GAN

- Рассмотренные ранее модели – *генеративно-состязательные нейросети* и *вариационные автокодировщики* – обладают рядом *преимуществ*:
  - способность изучать *сложные паттерны* в данных;
  - создание *структурированного*, часто *хорошо интерпретируемого* латентного пространства;
  - генерация *высококачественных* изображений, в том числе, *условная*;
  - *контроль атрибутов* сгенерированных изображений, и т.д.
- Однако, данные модели имеют и существенные *недостатки*, которые *ограничивают* их практическое применение:
  - ✓ *не позволяют явно вычислить* плотность распределения *реальных данных* $p(\mathbf{x})$ из-за *вычислительной сложности*;

VOLGOGRAD STATE TECHNICAL UNIVERSITY

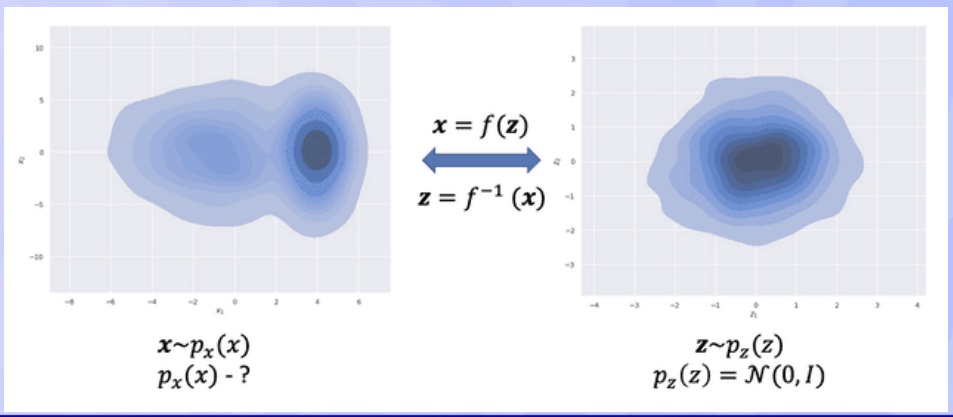
- ✓ процесс *обучения* может быть *сложным* из-за целого ряда явлений, включая *коллапс моды*, *коллапс апостериорного распределения*, *исчезающие градиенты* и *нестабильность обучения*.

- Следует заметить, что *явное вычисление* $p(\mathbf{x})$ позволяет *эффективно* решать многие задачи:
  - ➢ выборка *ненаблюдаемых*, но *реалистичных* новых данных (генерация);
  - ➢ предсказание *частоты* будущих событий (оценка плотности),
  - ➢ *вывод* латентных переменных,
  - ➢ *заполнение* неполных выборок данных, и т.д.
- *Явно* вычислять $p(\mathbf{x})$ способен другой *класс генеративных моделей* – *нормализующие потоки* (*normalizing flows*).

VOLGOGRAD STATE TECHNICAL UNIVERSITY

Идея нормализующих потоков

- Пусть $\mathbf{x} \sim p_x(\mathbf{x})$ – семпл из распределения *реальных данных*, а $\mathbf{z} \sim p_z(\mathbf{z})$ – семпл из распределения *латентной переменной* $\mathbf{z}$, причём $p_z(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbb{I})$.
- Пусть существует такое *отображение* $f\colon \mathbb{R}^n \to \mathbb{R}^n$, что $\mathbf{x} = f(\mathbf{z})$ и $\mathbf{z} = f^{-1}(\mathbf{x})$.
- Следовательно, с помощью функции f можно задать *преобразование* между *плотностями распределений* $p_x(\mathbf{x})$ и $p_z(\mathbf{z})$. А это означает, что можно *генерировать* объект $\mathbf{x}$, выполняя *семплирование* $\mathbf{z} \sim p_z(\mathbf{z})$ и выполняя *преобразование* $\mathbf{x} = f(\mathbf{z})$.
- *Обратное* отображение «*нормализует*» *сложное* распределение $p_x(\mathbf{x})$, приводя его к *простому* $p_z(\mathbf{z})$.



VOLGOGRAD STATE TECHNICAL UNIVERSITY

- Пусть задана функция $f: \mathbb{R}^n \to \mathbb{R}^m$. Тогда *матрица частных производных первого порядка* этой функции *по аргументам* называется *матрицей Якоби*:

$$J_{ij} = \frac{\partial f_i}{\partial x_j}, \qquad \mathbf{J}(\mathbf{x}) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}.$$

- *Определитель* – *функция*, определённая на множестве *квадратных матриц*. Если $\det \mathbf{M} = 0$, то матрица *не имеет обратной*.

$$\det \mathbf{M} = \det \begin{bmatrix} m_{11} & m_{12} & \cdots & m_{1n} \\ m_{21} & m_{22} & \cdots & m_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ m_{n1} & m_{n2} & \cdots & m_{nn} \end{bmatrix}.$$

- *Определитель* матрицы Якоби называется *якобианом*.

VOLGOGRAD STATE TECHNICAL UNIVERSITY

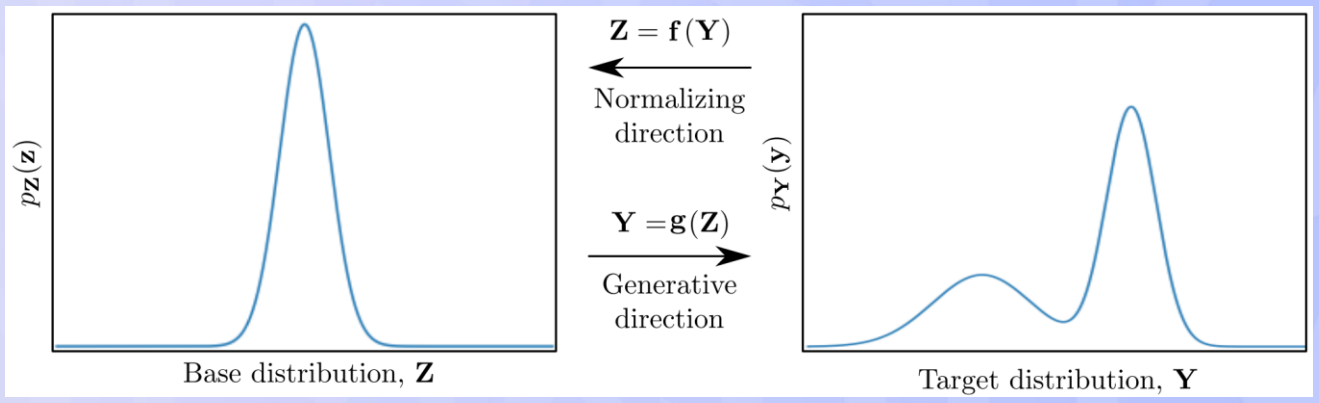
Формула замены переменной

- Пусть $\mathbf{y} \sim p_y(\mathbf{y})$, $\mathbf{z} \sim p_z(\mathbf{z})$, а отображение $f: \mathbb{R}^n \to \mathbb{R}^n$ *дифференцируемо, обратимо* и $\mathbf{y} = f(\mathbf{z})$. Тогда справедлива *формула замены переменной* (*change of variable theorem*):

$$p_y(\mathbf{y}) = p_z\left(f^{-1}(\mathbf{y})\right) \cdot \left|\det\left(\mathbf{J}_{f^{-1}}\right)\right|.$$

- В контексте генеративных моделей $g = f^{-1}(\mathbf{y})$ (*генератор*) «продвигает» плотность $p_z(\mathbf{z})$ (иногда называемую «*шумом*») к *сложной плотности* $p_y(\mathbf{y})$.

- Функция $f(\mathbf{z})$ «продвигает» $p_y(\mathbf{y})$ в *противоположном*, нормализующем направлении: от *сложного* распределения данных к более *простому* $p_z(\mathbf{z})$.



VOLGOGRAD STATE TECHNICAL UNIVERSITY

- *Нормализующим потоком* называют *обратимое дифференцируемое отображение* $f_{\boldsymbol{\theta}}: \mathbb{R}^n \to \mathbb{R}^n$, которое переводит реальные представления объектов в скрытые: $\mathbf{x} = f_{\boldsymbol{\theta}}(\mathbf{z})$ и $\mathbf{z} = f_{\boldsymbol{\theta}}^{-1}(\mathbf{x})$.
- При этом *плотность вероятности реальных данных* $p_x(\mathbf{x})$ вычисляется из плотности распределения $p_z(\mathbf{z})$ по *формуле замены переменной*.
- Обучение модели сводится к определению её *весов* с *помощью ММП*:

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}\in\Theta} \left[\log p_x(\mathcal{D}, \boldsymbol{\theta}) = \sum_{i=1}^{M} \log p_z\left(\left(f_{\boldsymbol{\theta}}^{-1}(\mathbf{x})\right)\right) + \log\left|\det\left(\mathbf{J}_{f^{-1}}(\mathbf{x})\right)\right| \right], \qquad \mathcal{D} = \{\mathbf{x}_i\}_{i=1}^{m}.$$

- Построение *произвольных сложных инвертируемых функций* является сложной задачей. Обычно используют функции, которые удобно *вычислять*, *инвертировать* и *рассчитывать* их якобиан.

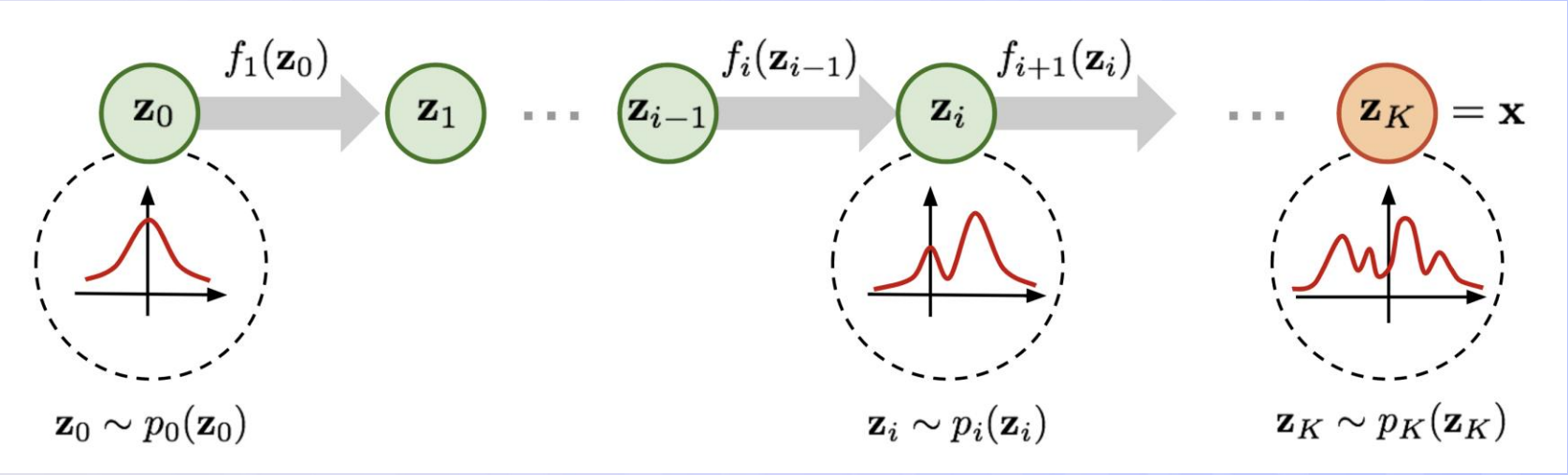
VOLGOGRAD STATE TECHNICAL UNIVERSITY

Комбинация отображений

- Как правило, для получения *достаточной выразительности* модель *нормализующего потока* составляют из *комбинации* K отображений:

$$f = f_1 \circ f_2 \circ \cdots \circ f_K$$

$$\mathbf{z}_{i-1} \sim p_{i-1}(\mathbf{z}_{i-1}) \qquad \mathbf{z}_i \sim f_i(\mathbf{z}_{i-1}) \Leftrightarrow \mathbf{z}_{i-1} = f_i^{-1}(\mathbf{z}_i) \qquad p_i(\mathbf{z}_i) = p_{i-1}\left(f_i^{-1}(\mathbf{z}_i)\right)\left|\det \frac{\partial f_i^{-1}}{\partial \mathbf{z}_i}\right|$$



VOLGOGRAD STATE TECHNICAL UNIVERSITY

Вывод плотности вероятности данных для комбинации отображений

- Воспользуемся *теоремой об обращении функции* (*inverse function theorem*), согласно которой, если $\mathbf{y} = f(\mathbf{x})$ и $\mathbf{x} = f^{-1}(\mathbf{y})$, то:

$$\frac{df^{-1}(\mathbf{y})}{d\mathbf{y}} = \frac{d\mathbf{x}}{d\mathbf{y}} = \left(\frac{d\mathbf{y}}{d\mathbf{x}}\right)^{-1} = \left(\frac{df(\mathbf{x})}{d\mathbf{x}}\right)^{-1}.$$

- Тогда:

$$p_i(\mathbf{z}_i) = p_{i-1}\left(f_i^{-1}(\mathbf{z}_i)\right)\left|\det\frac{\partial f_i^{-1}}{\partial \mathbf{z}_i}\right| = p_{i-1}\left(f_i^{-1}(\mathbf{z}_i)\right)\left|\det\left(\frac{\partial f_i}{\partial \mathbf{z}_{i-1}}\right)^{-1}\right|.$$

- Воспользуемся *свойством детерминанта инвертируемой функции*:

  так как $\det(\mathbf{M}) \cdot \det(\mathbf{M}^{-1}) = \det(\mathbf{M} \cdot \mathbf{M}^{-1}) = \det(\mathbf{E}) = 1$, то $\det(\mathbf{M}^{-1}) = (\det(\mathbf{M}))^{-1}$

- Тогда:

$$p_i(\mathbf{z}_i) = p_{i-1}\left(f_i^{-1}(\mathbf{z}_i)\right)\left|\det\left(\frac{\partial f_i}{\partial \mathbf{z}_{i-1}}\right)^{-1}\right| = p_{i-1}\left(f_i^{-1}(\mathbf{z}_i)\right)\left|\det\left(\frac{\partial f_i}{\partial \mathbf{z}_{i-1}}\right)\right|^{-1}.$$

VOLGOGRAD STATE TECHNICAL UNIVERSITY

Вывод плотности вероятности данных для комбинации отображений

- *Логарифмируя* полученное выражение, приходим к виду:

$$\log p_i(\mathbf{z}_i) = \log p_{i-1}\left(f_i^{-1}(\mathbf{z}_i)\right) - \log\left|\det\left(\frac{\partial f_i}{\partial \mathbf{z}_{i-1}}\right)\right|.$$

- Представим последовательность преобразования данных таким образом:

$$\mathbf{x} = \mathbf{z}_K = f_K \circ f_{K-1} \circ \cdots \circ f_1(\mathbf{z}_0), \qquad \mathbf{z}_i \sim \pi_i(\mathbf{z}_i).$$

$$\log p(\mathbf{x}) = \log \pi_K(\mathbf{z}_K) = \log \pi_{K-1}(\mathbf{z}_{K-1}) - \log\left|\det \frac{\partial f_K}{\partial \mathbf{z}_{K-1}}\right| =$$

$$= \log \pi_{K-2}(\mathbf{z}_{K-2}) - \log\left|\det \frac{\partial f_K}{\partial \mathbf{z}_{K-1}}\right| - \log\left|\det \frac{\partial f_{K-1}}{\partial \mathbf{z}_{K-2}}\right| = \cdots = \log \pi_0(\mathbf{z}_0) - \sum_{i=1}^{K} \log\left|\det \frac{\partial f_i}{\partial \mathbf{z}_{i-1}}\right|.$$

- Вычисление *якобиана* является *вычислительно затратной операцией*!
- Функцию f выбирают так, чтобы было *легко посчитать* f^{-1} и *якобиан*, но при этом обеспечить *достаточную выразительность* модели.

VOLGOGRAD STATE TECHNICAL UNIVERSITY

Поэлементно преобразующие нормализующие потоки

- Простой нормализующий поток может быть построен на основе любой биективной скалярной функции (*elementwise flow*). Пусть $h: \mathbb{R} \to \mathbb{R}$ – скалярная биективная функция, а $\mathbf{x} = [x_1, x_2, \dots, x_D]^T$. Тогда:

$$f^{-1}(\mathbf{x}) = [h(x_1), h(x_2), \dots, h(x_D)]^T.$$

- Функция $f^{-1}(\mathbf{x})$ также *биективна*, а её *якобиан*:

$$\det\left(J_{f^{-1}}\right) = \prod_{i=1}^{D} \det\left(\frac{\partial h(x_i)}{\partial x_i}\right).$$

- Подход может быть *обобщен* путем добавления *своей функции* $h_i(x_i)$ для *каждого* элемента x_i.

- *Недостатком* данного типа нормализующего потока является отсутствие *смеси компонент* переменной.

VOLGOGRAD STATE TECHNICAL UNIVERSITY

NICE: Non-linear Independent Component Estimation

- Метод основан на *идее*, что в *хорошем представлении* данные имеют *распределение*, которое легко моделировать.
- В *NICE* латентные переменные разделяются на два блока: $\mathbf{z} = (\mathbf{z}_{1..d}, \mathbf{z}_{d+1..K})$, и к ним применяется *преобразование* следующего вида:

$$\mathbf{x}_{1..d} = \mathbf{z}_{1..d},$$
$$\mathbf{x}_{d+1..K} = \mathbf{z}_{d+1..K} + m(\mathbf{z}_{1..d}).$$

Здесь m – произвольная *сложная функция* (например, *нейронная сеть*).

- Для данного преобразования легко вычисляется обратное ему и якобиан:

$$\mathbf{z}_{1..d} = \mathbf{x}_{1..d},$$
$$\mathbf{z}_{d+1..K} = \mathbf{x}_{d+1..K} - m(\mathbf{x}_{1..d}).$$

$$\det\left(J_{f^{-1}}\right) = \det\left(\frac{\partial f^{-1}}{\partial \mathbf{x}}\right) = \prod_{i=1}^{d} \frac{\partial \mathbf{z}_i}{\partial \mathbf{x}_i} \cdot \prod_{i=d+1}^{K} \frac{\partial \mathbf{z}_i}{\partial \mathbf{x}_i} = 1.$$

- Из-за того, что первые d каналов вектора *совпадают* с координатами *нормального шума*, моделирования этих каналов *не происходит*.

- С целью *повышения выразительности модели* взаимосвязь между блоками латентной переменной можно *расширить*.
- Введем преобразование $\mathbf{x} = f(\mathbf{z})$ следующим образом:

$$\mathbf{x}_{1..d} = \mathbf{z}_{1..d},$$
$$\mathbf{x}_{d+1..K} = g\left(\mathbf{z}_{d+1..K}; m(\mathbf{z}_{1..d})\right).$$

- И тогда *обратное преобразование* и *якобиан*:

$$\mathbf{z}_{1..d} = \mathbf{x}_{1..d},$$
$$\mathbf{z}_{d+1..K} = g^{-1}\left(\mathbf{x}_{d+1..K}; m(\mathbf{x}_{1..d})\right). \quad \det\left(J_{f^{-1}}\right) = \det\left(\frac{\partial f^{-1}}{\partial \mathbf{x}}\right) = \det\begin{bmatrix} \mathbf{E}_d & \mathbf{0} \\ \frac{\partial \mathbf{z}_{d+1..K}}{\partial \mathbf{x}_{1..d}} & \frac{\partial \mathbf{z}_{d+1..K}}{\partial \mathbf{x}_{d+1..K}} \end{bmatrix} = \det\left[\frac{\partial \mathbf{z}_{d+1..K}}{\partial \mathbf{x}_{d+1..K}}\right].$$

- Подобное преобразование называется *слоем связывания* (*coupling layer*) со *связывающей функцией* m. Представленный выше слой связывания представлен в *обобщенном виде* (*general coupling layer*).

VOLGOGRAD STATE TECHNICAL UNIVERSITY

Типы слоёв связывания

- Слои связывания могут быть различными, различаясь выбором вида функции $g(\mathbf{a}, \mathbf{b})$:
  - *Аддитивный* слой связывания (как в NICE):

$$g(\mathbf{a}, \mathbf{b}) = \mathbf{a} + \mathbf{b}.$$

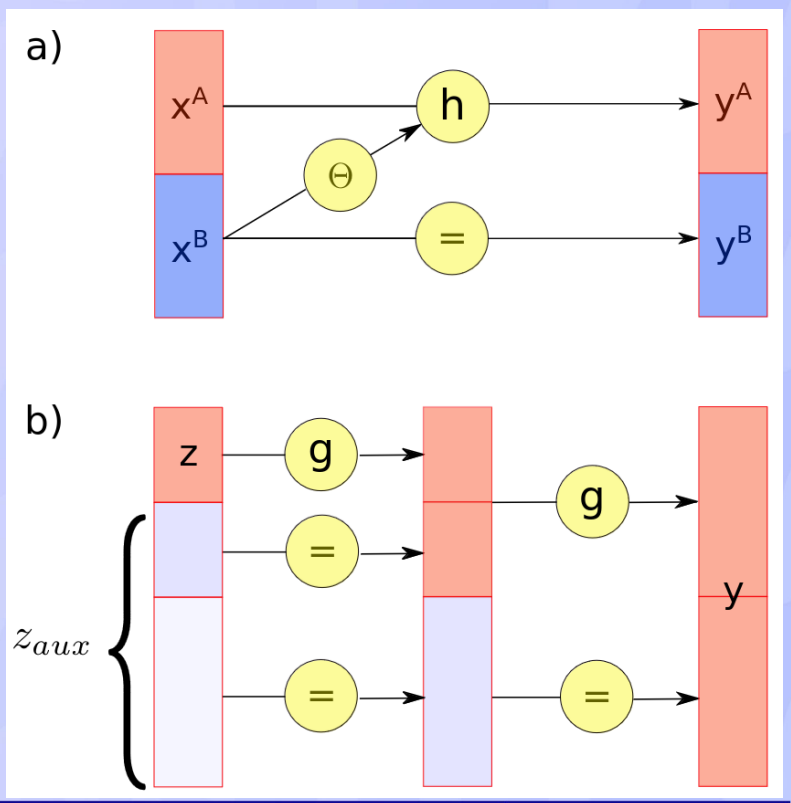
  - *Мультипликативный* слой связывания:

$$g(\mathbf{a}, \mathbf{b}) = \mathbf{a} \odot \mathbf{b}, \qquad \mathbf{b} \neq \mathbf{0}.$$

  - *Афинный* слой связывания:

$$g(\mathbf{a}, \mathbf{b}) = \mathbf{a} \odot \mathbf{b}^A + \mathbf{b}^B, \qquad \mathbf{b}^A \neq \mathbf{0},$$
$$\text{если } m\colon \mathbb{R}^d \to \mathbb{R}^{K-d} \times \mathbb{R}^{K-d}$$

- Слои могут *объединяться*, образуя более *сложную структуру*.



VOLGOGRAD STATE TECHNICAL UNIVERSITY

RealNVP: Real-valued Non-Volume Preserving

- *Афинное связывание* было применено в *архитектуре RealNVP*, в которой реализовано преобразование вида:

$$\mathbf{x}_{1..d} = \mathbf{z}_{1..d},$$
$$\mathbf{x}_{d+1..K} = \mathbf{z}_{d+1..K} \odot \exp\big(s(\mathbf{z}_{1..d})\big) + t(\mathbf{z}_{1..d}).$$

  Здесь $s\colon \mathbb{R}^d \to \mathbb{R}^{K-d}$ и $t\colon \mathbb{R}^d \to \mathbb{R}^{K-d}$ – функции *масштабирования* и *сдвига*.

- *Обратное преобразование* примет вид:

$$\mathbf{z}_{1..d} = \mathbf{x}_{1..d},$$
$$\mathbf{z}_{d+1..K} = \big(\mathbf{x}_{d+1..K} - t(\mathbf{z}_{1..d})\big) \odot \exp\big(-s(\mathbf{z}_{1..d})\big).$$

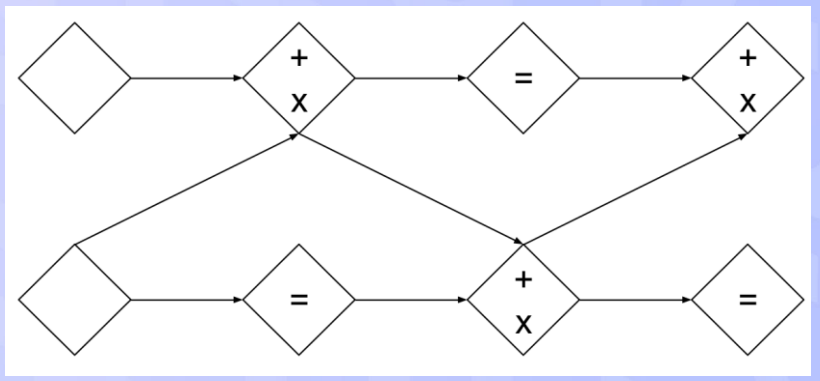
- А *якобиан*:

$$\det\big(J_{f^{-1}}\big) = \det\begin{bmatrix} \mathbf{E}_d & \mathbf{0}_{d\times(K-d)} \\ \dfrac{\partial \mathbf{z}_{d+1..K}}{\partial \mathbf{x}_{1..d}} & \operatorname{diag}\big(\exp\big(-s(\mathbf{z}_{1..d})\big)\big) \end{bmatrix} = \prod_{j=1}^{K-d} \exp\big(-s(\mathbf{z}_{1..d})\big)_j = \exp\left(-\sum_{j=1}^{K-d} s(\mathbf{z}_{1..d})_j\right).$$

VOLGOGRAD STATE TECHNICAL UNIVERSITY

Организация слоёв связывания в архитектуре RealNVP

- В *афинном связывании* большая часть компонент остается *неизменной*.
- Чтобы преобразование было способным моделировать распределение *во всех компонентах*, в разных слоях связывания неизменными оставляют *разные подмножества* компонент.
- Слои связывания организуются по *чередующемуся* шаблону, так что компоненты, которые остаются *неизменными* в одном слое связи, *обновляются* в следующем.

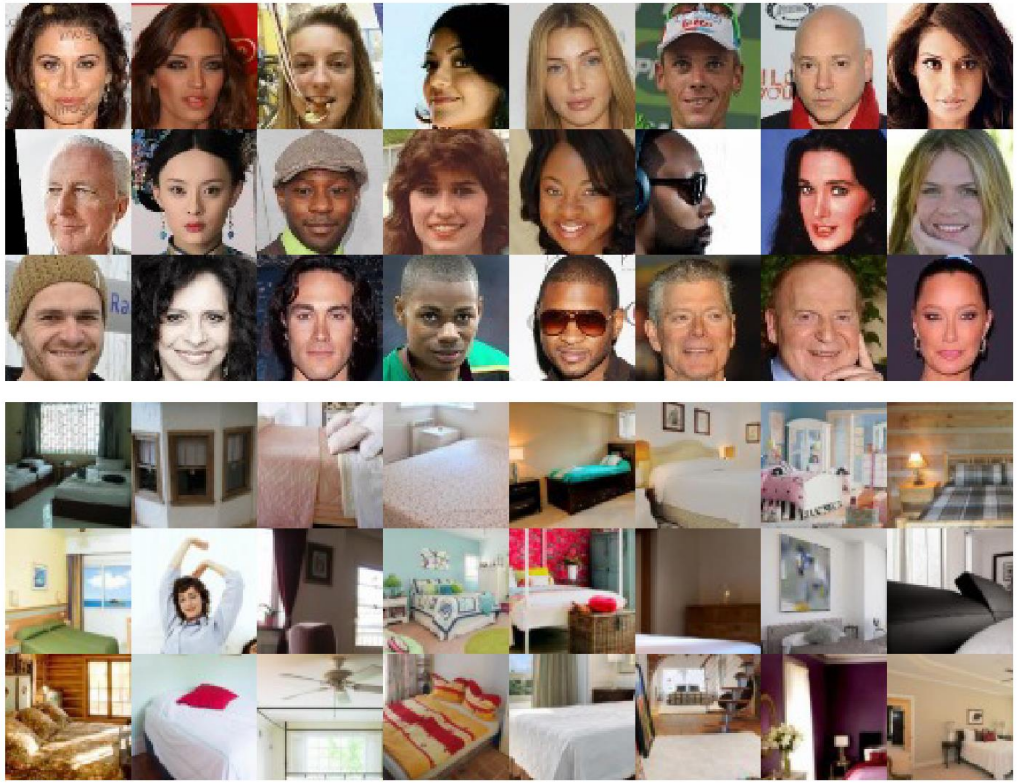


- Также, чтобы улучшить *сходимость* глубоких нормализующих потоков, авторы используют *Batch Normalization*. Данное преобразование тоже является обратимым, а его *якобиан* вычисляется просто.

VOLGOGRAD STATE TECHNICAL UNIVERSITY

Результаты работы RealNVP

Примеры изображений из датасетов



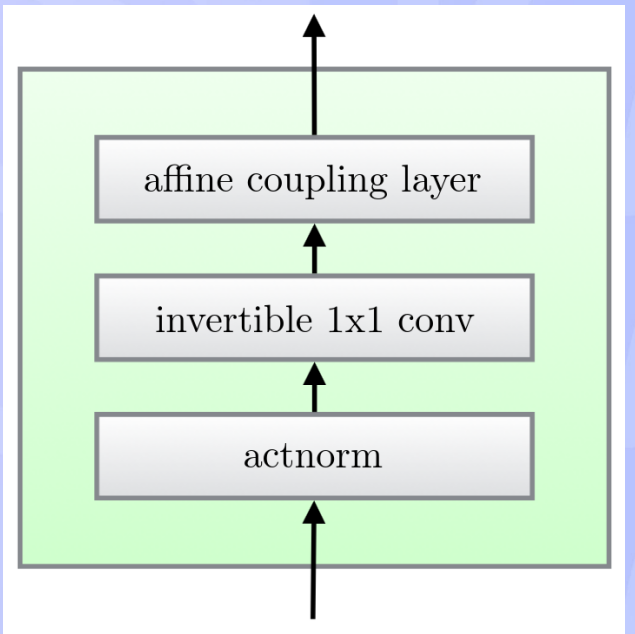
Примеры семплов из RealNVP

VOLGOGRAD STATE TECHNICAL UNIVERSITY

Glow: Generative Flow with Invertible 1×1 Convolutions

- Модель *Glow* расширяет модель *RealNVP*, заменяя *чередование* слоёв связывания *инвертируемой свёрткой* 1 × 1.
- *Преобразование* данных в модели *Glow* состоит из *трех шагов*:

1. Activation normalization («*actnorm*»). Выполняет *афинное преобразование* для *каждого канала* с *обучаемыми* параметрами.
2. Invertible 1 × 1 convolution. *Обобщает* операцию *перестановки порядка* каналов. Имеет *одинаковое* количество *входных* и *выходных* каналов.
3. Affine coupling layer. Слой *связывания*, имеет ту же *архитектуру*, что и в *RealNVP*.



VOLGOGRAD STATE TECHNICAL UNIVERSITY

Нормализация активации в Glow

- Поскольку *дисперсия активаций* после слоя *пакетной нормализации* обратно пропорциональна размеру *мини-батча*, то *производительность* модели *снижается* при его небольшом размере. А для *больших* изображений размер мини-батча *небольшой* из-за ограничения памяти.
- Для обработки изображений $\mathbf{x}, \mathbf{y} \in \mathbb{R}_{+}^{H \times W \times C}$ предлагается использовать слой *нормализации активации*:

$$\forall i, j : \mathbf{y}_{i,j} = \mathbf{s} \odot \mathbf{x}_{i,j} + \mathbf{b} \quad \Leftrightarrow \quad \mathbf{x}_{i,j} = \frac{(\mathbf{y}_{i,j} - \mathbf{b})}{\mathbf{s}},$$

$$\log\left|\det\left(J_{f^{-1}}\right)\right| = \log\left|\det\left(\frac{\partial f^{-1}}{\partial \mathbf{x}}\right)\right| = H \cdot W \cdot \sum_{j} \log|\mathbf{s}|_{j}\,.$$

- Параметры $\mathbf{s}$ и $\mathbf{b}$ *инициализируются* таким образом, что *на первом шаге* обучения *активации* имеют *нулевое среднее* и *единичную дисперсию*.

VOLGOGRAD STATE TECHNICAL UNIVERSITY

Инвертируемая свёртка в Glow

- Пусть на вход инвертируемой свёртки 1×1 подаётся $\mathbf{x} \in \mathbb{R}_{+}^{H \times W \times C}$. Свертка задана матрицей $\mathbf{W} \in \mathbb{R}^{C \times C}$. Тогда:

$$\forall i, j : \mathbf{y}_{i,j} = \mathbf{W}\mathbf{x}_{i,j} \quad \Leftrightarrow \quad \mathbf{x}_{i,j} = \mathbf{W}^{-1}\mathbf{y}_{i,j},$$

- Для расчёта *обратного* преобразования требуется *обратить* матрицу $\mathbf{W}$. Поскольку размеры матрицы сравнительно *небольшие*, данная операция выполняется за *вменяемое время*.
- Рассчитаем *якобиан*. Учтём, что каждый *вектор*, соответствующий одному «*пикселю*» изображения $\mathbf{x}_{i,j}$, $i \in [1..H]$, $j \in [1..W]$, *независимо от других* подаётся на вход *свёрточному* слою и участвует в расчёте *производной*.

$$\log\left|\det\left(J_{f^{-1}}\right)\right| = \log\left|\det\left(\frac{\partial \mathrm{conv2d}(\mathbf{x}, \mathbf{W})}{\partial \mathbf{x}}\right)\right| = \log\left(\left|\det \mathbf{W}\right|^{H \cdot W}\right) = H \cdot W \cdot \log\left|\det \mathbf{W}\right|.$$

VOLGOGRAD STATE TECHNICAL UNIVERSITY

Афинный слой связывания в Glow

- *Афинный слой связывания аналогичен слою в RealNVP:*

$$\mathbf{x}_a, \mathbf{x}_b = \text{split}(\mathbf{x}),$$
$$(\log \mathbf{s}, \mathbf{t}) = \text{NN}(\mathbf{x}_b),$$
$$\mathbf{s} = \exp \log \mathbf{s},$$
$$\mathbf{y}_a = \mathbf{s} \odot \mathbf{x}_a + \mathbf{t},$$
$$\mathbf{y}_b = \mathbf{x}_b,$$
$$\mathbf{y} = \text{concat}(\mathbf{y}_a, \mathbf{y}_b).$$

$$\mathbf{y}_a, \mathbf{y}_b = \text{split}(\mathbf{y}),$$
$$(\log \mathbf{s}, \mathbf{t}) = \text{NN}(\mathbf{y}_b),$$
$$\mathbf{s} = \exp \log \mathbf{s},$$
$$\mathbf{x}_a = \frac{\mathbf{y}_a - \mathbf{t}}{\mathbf{s}},$$
$$\mathbf{x}_b = \mathbf{y}_b,$$
$$\mathbf{x} = \text{concat}(\mathbf{x}_a, \mathbf{x}_b).$$

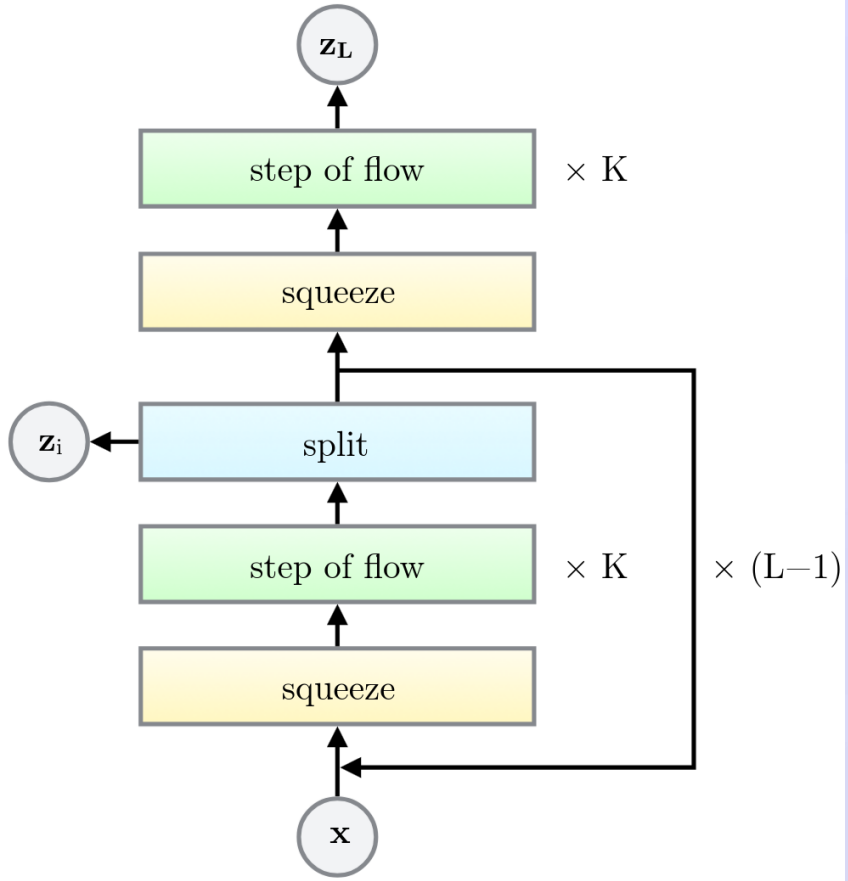
- Якобиан:

$$\log\left|\det\left(J_{f^{-1}}\right)\right| = \sum_j s(\mathbf{z}_{1..d})_j .$$

- Функция $\text{split}(\cdot)$ *разделяет* входной вектор на *две половины* вдоль размерности *каналов*, функция $\text{concat}(\cdot, \cdot)$ выполняет *обратную операцию*.

VOLGOGRAD STATE TECHNICAL UNIVERSITY

Многослойная архитектура Glow и результаты её работы



VOLGOGRAD STATE TECHNICAL UNIVERSITY

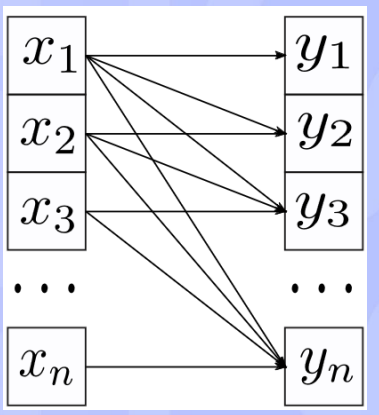
- Пусть $h_{\boldsymbol{\theta}}: \mathbb{R} \to \mathbb{R}$ – *биективная* функция. Тогда *отображение* $g: \mathbb{R}^D \to \mathbb{R}^D$ называется *авторегрессионной моделью*, если *текущее* выходное значение модели $\mathbf{y} = g(\mathbf{x})$ зависит от *предыдущих* значений на *входе*:

$$y_t = h_{\Theta_t(\mathbf{x}_{1:t-1})}(x_t),$$

где $\mathbf{x}_{1:t} = (x_1, x_2, \ldots, x_t)$. Для $t = 2, \ldots, D$ выбирается произвольная функция $\Theta_t(\cdot)$, отображающая входные значения на множество параметров $\boldsymbol{\theta}$, и $\Theta_1 = \text{const}$.

- Функции $\Theta_t(\cdot)$ называются *обусловливающими* (*conditioners*).
- Матрица Якоби *авторегрессионной модели* имеет *треугольный* вид. *Якобиан* может быть рассчитан *аналогично*:

$$\det(J) = \prod_{t=1}^{D} \frac{\partial y_t}{\partial x_t}.$$



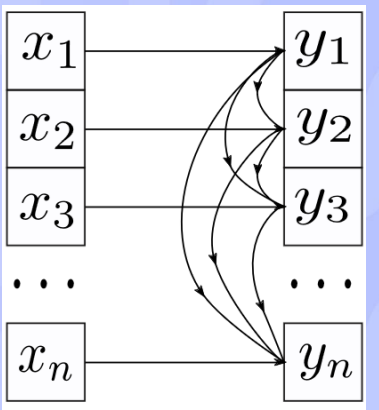
VOLGOGRAD STATE TECHNICAL UNIVERSITY

Обратные авторегрессионные потоки

- Модель *авторегрессионных потоков* трудно применить на практике к данным *высокой размерности*, поскольку *генерация нового объекта* осуществляется *авторегрессионно* по *входным* значениям, что становится слишком *вычислительно сложным*.
- Идея: введем *авторегрессионную* зависимость не по $\mathbf{x}$, а по $\mathbf{y}$:

$$y_t = h_{\Theta_t(\mathbf{y}_{1:t-1})}(x_t).$$

- Проблема долгого вычисления авторегрессии остаётся. Модель *обратного авторегрессионного потока* позволяет генерировать объекты *быстрее* (так как $|y| \ll |x|$), но процесс *обучения* уже не может быть *распараллелен*, и он занимает *больше времени*.



VOLGOGRAD STATE TECHNICAL UNIVERSITY

Демонстрация практических примеров

VOLGOGRAD STATE TECHNICAL UNIVERSITY

Заключение

1. Рассмотрели нормализующие потоки как класс глубоких генеративных моделей, разобрались с принципами их организации и работы.
2. Выяснили основные правила функционирования нормализующих потоков, их обучения и генерации новых данных с их помощью.
3. Рассмотрели организацию поэлементно преобразующих потоков.
4. Ввели понятие слоёв связывания в нормализующих потоках, рассмотрели модели NICE и RealNVP, их организацию и правила работы.
5. Познакомились с архитектурой Glow, в теории и на практическом примере разобрали процессы обучения и генерации модели.
6. Рассмотрели применение потоков к построению прямых и обратных авторегрессионных моделей, разобрали особенности их работы.

VOLGOGRAD STATE TECHNICAL UNIVERSITY

Спасибо за внимание!

Волгоград 2024