Генеративные модели

Лекция 4: Вариационные автокодировщики

Модели со скрытыми переменными

Предположим, что помимо наблюдаемой переменной $\mathbf{x}$ в нашей задаче есть некоторая скрытая переменная $\boldsymbol{z}$, которая влияет на $\mathbf{x}$

Мы не можем измерить ее напрямую, но знаем, что она есть

Мы можем записать совместную плотность и разложить ее на условную и априорную:

$$p(\mathbf{x}, \mathbf{z}|\boldsymbol{\theta}) = p(\mathbf{x}|\boldsymbol{z}, \boldsymbol{\theta}) \cdot p(\boldsymbol{z}|\boldsymbol{\theta})$$

$p(\boldsymbol{z}|\boldsymbol{\theta})$ – априорное распределение на скрытых переменных
$p(\mathbf{x}|\boldsymbol{z}, \boldsymbol{\theta})$ – условное распределение наблюдаемых данных

Проинтегрируем по всем $\mathbf{z}$, чтобы найти правдоподобие:

$$p(\mathbf{x}|\boldsymbol{\theta}) = \int p(\mathbf{x}, \mathbf{z}|\boldsymbol{\theta})\, d\boldsymbol{z} = \int p(\mathbf{x}|\boldsymbol{z}, \boldsymbol{\theta}) \cdot p(\boldsymbol{z}|\boldsymbol{\theta})\, d\boldsymbol{z}$$

Модели со скрытыми переменными

Наша цель – по-прежнему максимизация правдоподобия:

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{n} \log p_{\boldsymbol{\theta}}(\mathbf{x}_i) \quad \rightarrow \quad \boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{n} \log \int p(\mathbf{x}_i|\boldsymbol{z}_i, \boldsymbol{\theta}) \cdot p(\boldsymbol{z}_i|\boldsymbol{\theta})\, d\boldsymbol{z}_i$$

Мы заменили одно сложное распределение $p(\mathbf{x}|\boldsymbol{\theta})$ на интеграл от двух более простых:

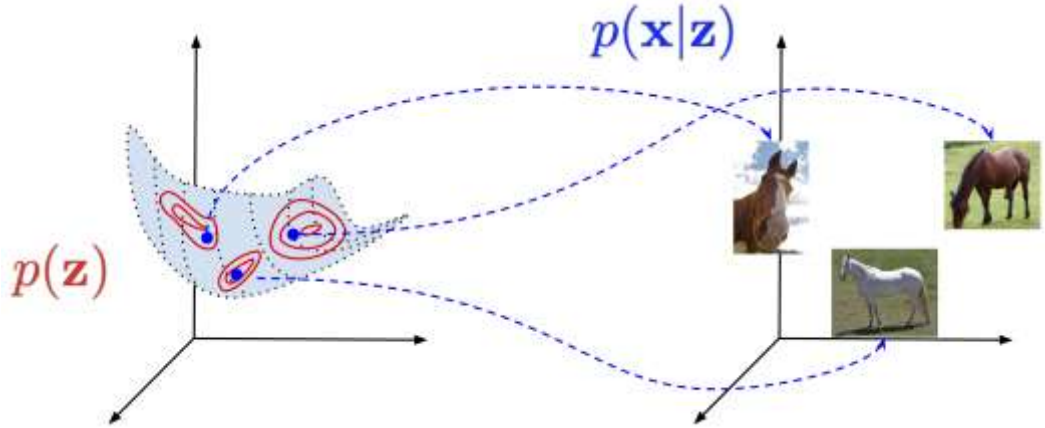
$$\log p(\mathbf{x}|\boldsymbol{\theta}) = \log \int p(\mathbf{x}|\boldsymbol{z}, \boldsymbol{\theta}) \cdot p(\boldsymbol{z}|\boldsymbol{\theta})\, d\boldsymbol{z} \quad \rightarrow \quad \max_{\boldsymbol{\theta}}$$

Выбирая подходящие распределения для $p(\mathbf{x}|\boldsymbol{z}, \boldsymbol{\theta})$ и $p(\boldsymbol{z}|\boldsymbol{\theta})$, мы можем либо получить аналитическое решение для $\boldsymbol{\theta}^*$, либо использовать методы оптимизации

MLE для моделей со скрытыми переменными

Нужно вычислить сложный интеграл:

$$\sum_{i=1}^{n} \log p(\mathbf{x}_i|\boldsymbol{\theta}) = \sum_{i=1}^{n} \log \int p(\mathbf{x}_i|\boldsymbol{z}_i, \boldsymbol{\theta}) \cdot p(\boldsymbol{z}_i)\, d\boldsymbol{z}_i$$

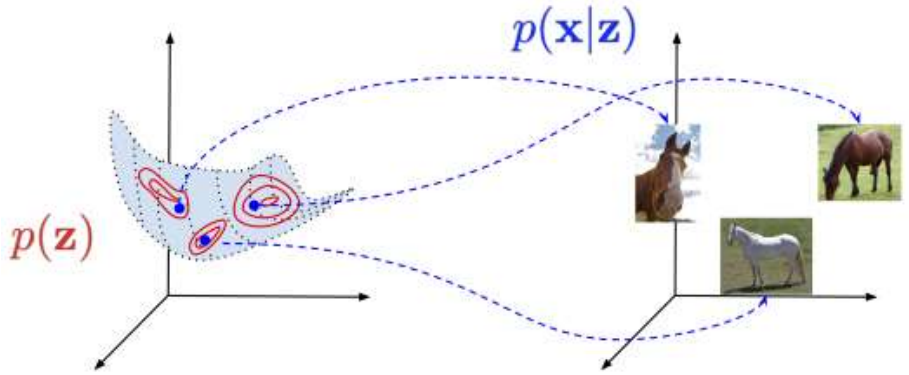


Попробуем оценить этот интеграл методом Монте-Карло:

$$p(\mathbf{x}|\boldsymbol{\theta}) = \int p(\mathbf{x}|\boldsymbol{z}, \boldsymbol{\theta}) \cdot p(\boldsymbol{z})\, d\boldsymbol{z} = \mathbb{E}_{p(\boldsymbol{z})}[p(\mathbf{x}|\boldsymbol{z}, \boldsymbol{\theta})] \approx \frac{1}{K} \sum_{k=1}^{K} p(\mathbf{x}|\boldsymbol{z}_k, \boldsymbol{\theta})$$

MLE для моделей со скрытыми переменными

- Большинство сэмплов $\boldsymbol{z}_k$ попадут в «бесполезные» области и декодер сгенерирует шум



Наше априорное распределение $p(\boldsymbol{z})$ ничего не знает об изображениях $\mathbf{x}$

В многомерном пространстве объем «пустого» пространства огромен по сравнению с объемом полезной области

Чтобы получить надежную оценку интеграла, нам понадобиться экспоненциально большое число сэмплов K

Вариационный вывод

Наша цель – оценить $\log p(\mathbf{x}|\boldsymbol{\theta})$:

$$\log p(\mathbf{x}|\boldsymbol{\theta}) = \log \int p(\mathbf{x}, \boldsymbol{z}|\boldsymbol{\theta}) d\boldsymbol{z}$$

Введем вспомогательное распределение $q(\boldsymbol{z})$ – произвольную плотность над скрытыми переменными $\boldsymbol{z}$

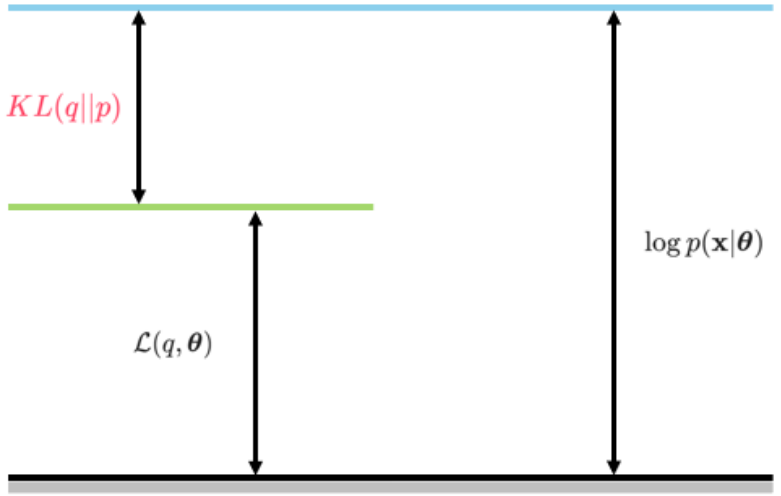
Идея:

Мы не знаем $\log p(\mathbf{x}|\boldsymbol{\theta})$, но можем попытаться построить для него некую ***нижнюю границу*** (***lower bound***), которую будем оптимизировать

Evidence Lower Bound

Зазор между $\log p(\mathbf{x}|\boldsymbol{\theta})$ и $\mathcal{L}_{q,\boldsymbol{\theta}}(\mathbf{x})$ равен KL-дивергенции между вариационным распределением $q(\boldsymbol{z})$ и истинным апостериорным распределением $p(\boldsymbol{z}|\mathbf{x},\boldsymbol{\theta})$:

$$\log p(\mathbf{x}|\boldsymbol{\theta}) - \mathcal{L}_{q,\boldsymbol{\theta}}(\mathbf{x}) = KL(q(\boldsymbol{z})||p(\boldsymbol{z}|\mathbf{x},\boldsymbol{\theta}))$$



Выводы

Вместо прямой максимизации правдоподобия $\log p(\mathbf{x}|\boldsymbol{\theta})$ будем максимизировать его нижнюю границу $\mathcal{L}_{q,\boldsymbol{\theta}}(\mathbf{x})$ по параметрам вариационного распределения q и параметрам модели $\boldsymbol{\theta}$:

$$\max_{q,\boldsymbol{\theta}} \mathcal{L}_{q,\boldsymbol{\theta}}(\mathbf{x})$$

Максимизируя $\mathcal{L}_{q,\boldsymbol{\theta}}(\mathbf{x})$, мы одновременно стараемся сделать 2 вещи:

1. Увеличить правдоподобие наблюдаемых данных

2. Уменьшить зазор $KL(q(\boldsymbol{z})||p(\boldsymbol{z}|\mathbf{x},\boldsymbol{\theta}))$, аппроксимируя $p(\boldsymbol{z}|\mathbf{x},\boldsymbol{\theta})$ с помощью $q(\boldsymbol{z})$

План

- Amortized Variational Inference
- Variational Autoencoder
- ELBO Surgery

Amortized Variational Inference

Проблемы вариационного вывода

Цель:

Хотим аппроксимировать сложное апостериорное распределение $p(\boldsymbol{z}|\mathbf{x}, \boldsymbol{\theta})$ с помощью более простого $q(\boldsymbol{z})$

***Проблема* 1**:

Пространство всех возможных функций $q(\boldsymbol{z})$ бесконечно

***Проблема* 2**:

Для датасета из N объектов $\{\mathbf{x}_1, .., \mathbf{x}_n\}$, нам нужно решать N отдельных задач оптимизации, чтобы найти $\{q_1(\mathbf{z}_1), .., q_n(\mathbf{z}_n)\}$

Амортизированный вариационный вывод

Решение 1, параметризация:

Ограничим поиск $q(\mathbf{z})$ по конкретным семействам распределений:

$$q(\mathbf{z}) \to q_{\boldsymbol{\phi}}(\mathbf{z})$$

Решение 2, амортизация (***Amortized Variational Inference***):

Вместо отдельных параметров $\boldsymbol{\phi}$ для каждого $\mathbf{x}$ обучим единую ***нейросеть−кодировщик,*** которая по $\mathbf{x}$ будет предсказывать параметры для q:

$$q_{\boldsymbol{\phi}}(\mathbf{z}) \to q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$$

Амортизированный вариационный вывод

Введя эти 2 ограничения мы:

1. Ушли от необходимости считать вариационное распределение для каждого объекта $\mathbf{x}$
2. Введя параметры $\boldsymbol{\phi}$, мы признаем, что наше решение может быть не точным и KL-дивергенция между $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\mathbf{x})$ и $p(\boldsymbol{z}|\mathbf{x},\boldsymbol{\theta})$ может быть больше нуля

Теперь наша задача сводится к нахождению к нахождению оптимальных $\boldsymbol{\phi}$ и $\boldsymbol{\theta}$, которые максимизируют $ELBO$

- E-шаг:

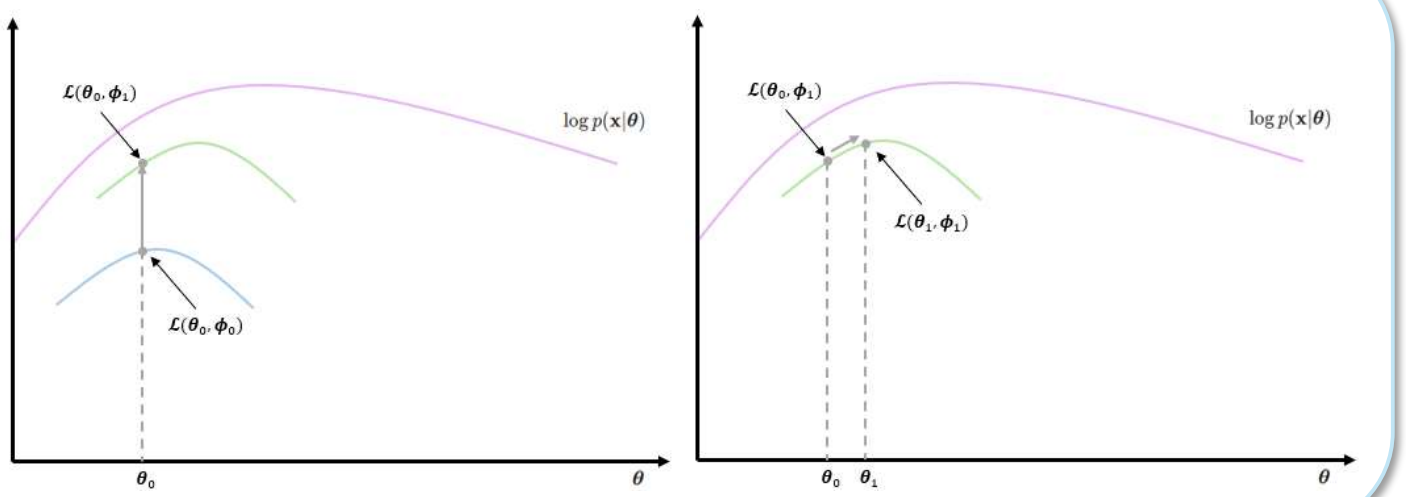
$$\boldsymbol{\phi}_k = \boldsymbol{\phi}_{k-1} + \eta \nabla_{\boldsymbol{\phi}} \mathcal{L}_{\boldsymbol{\phi},\boldsymbol{\theta}_{k-1}}(\mathbf{x}) \Big|_{\boldsymbol{\phi}=\boldsymbol{\phi}_{k-1}}$$

- M-шаг:

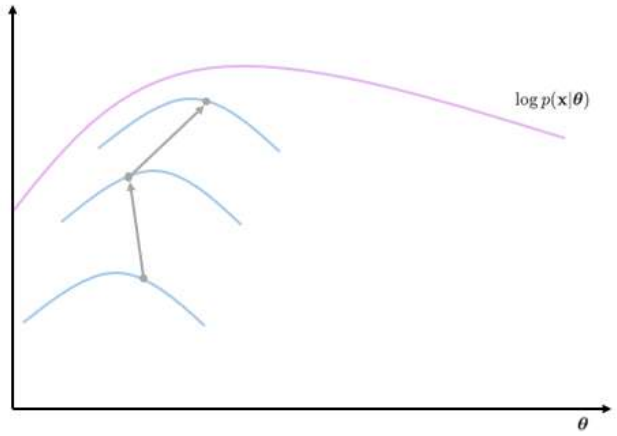
$$\boldsymbol{\theta}_k = \boldsymbol{\theta}_{k-1} + \eta \nabla_{\boldsymbol{\theta}} \mathcal{L}_{\boldsymbol{\phi}_k,\boldsymbol{\theta}}(\mathbf{x}) \Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_{k-1}}$$

EM алгоритм

- У нас по-прежнему чередование E- и M-шагов, но теперь с помощью градиентного спуска
- Отличие от обычного EM алгоритма в том, что теперь $KL(q(\boldsymbol{z})||p(\boldsymbol{z}|\mathbf{x},\boldsymbol{\theta})) \neq \mathbf{0}$ из-за ограничений кодировщика



- На практике мы будем делать эти шаги ***одновременно*** (***совместно***)



EM алгоритм

Как вычислить градиенты $\nabla_{\boldsymbol{\phi}}\mathcal{L}_{\boldsymbol{\phi},\boldsymbol{\theta}_{k-1}}$ и $\nabla_{\boldsymbol{\theta}}\mathcal{L}_{\boldsymbol{\phi}_k,\boldsymbol{\theta}}$?

$$\boldsymbol{\phi}_k = \boldsymbol{\phi}_{k-1} + \eta\nabla_{\boldsymbol{\phi}}\mathcal{L}_{\boldsymbol{\phi},\boldsymbol{\theta}_{k-1}}(\mathbf{x})\Big|_{\boldsymbol{\phi}=\boldsymbol{\phi}_{k-1}}$$

$$\boldsymbol{\theta}_k = \boldsymbol{\theta}_{k-1} + \eta\nabla_{\boldsymbol{\theta}}\mathcal{L}_{\boldsymbol{\phi}_k,\boldsymbol{\theta}}(\mathbf{x})\Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_{k-1}}$$

Формула $\boldsymbol{ELBO}$ содержит в себе матожидание по вариационному распределению $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\mathbf{x})$:

$$\mathcal{L}_{\boldsymbol{\phi},\boldsymbol{\theta}}(\mathbf{x}) = \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\mathbf{x})}[\log p(\mathbf{x}|\boldsymbol{z},\boldsymbol{\theta})] - KL(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\mathbf{x})||p(\boldsymbol{z}))$$

Градиент на M-шаге

Считаем градиент по параметрам $\boldsymbol{\theta}$:

$$\mathcal{L}_{\boldsymbol{\phi},\boldsymbol{\theta}}(\mathbf{x}) = \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\mathbf{x})}[\log p(\mathbf{x}|\boldsymbol{z},\boldsymbol{\theta})] - KL(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\mathbf{x})||p(\boldsymbol{z}))$$

KL-дивергенция не зависит от $\boldsymbol{\theta}$, так как q параметризуется $\boldsymbol{\phi}$, а априорное распределение $p(\boldsymbol{z})$ обычно фиксировано:

$$\nabla_{\boldsymbol{\theta}}\mathcal{L}_{\boldsymbol{\phi},\boldsymbol{\theta}}(\mathbf{x}) = \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\mathbf{x})}[\log p(\mathbf{x}|\boldsymbol{z},\boldsymbol{\theta})] = \nabla_{\boldsymbol{\theta}} \int q_{\boldsymbol{\phi}}(\boldsymbol{z}|\mathbf{x}) \log p(\mathbf{x}|\boldsymbol{z},\boldsymbol{\theta}) d\boldsymbol{z} = \int q_{\boldsymbol{\phi}}(\boldsymbol{z}|\mathbf{x}) \nabla_{\boldsymbol{\theta}} \log p(\mathbf{x}|\boldsymbol{z},\boldsymbol{\theta}) d\boldsymbol{z}$$

Для оценки матожидания будем использовать метод Монте-Карло:

$$\nabla_{\boldsymbol{\theta}}\mathcal{L}_{\boldsymbol{\phi},\boldsymbol{\theta}}(\mathbf{x}) \approx \nabla_{\boldsymbol{\theta}} \log p(\mathbf{x}|\boldsymbol{z}^*,\boldsymbol{\theta}), \qquad \boldsymbol{z}^* \sim q_{\boldsymbol{\phi}}(\boldsymbol{z}|\mathbf{x})$$

Поскольку $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\mathbf{x})$ обусловлено на конкретный $\mathbf{x}$, то его вероятностная масса будет сосредоточена в области пространства $\boldsymbol{z}$, которая наиболее вероятна для этого $\mathbf{x}$, а значит дисперсия будет сильно ниже, чем при наивном подходе

Градиент на E-шаге

Считаем градиент по параметрам $\boldsymbol{\phi}$ при фиксированных $\boldsymbol{\theta}$:

$$\mathcal{L}_{\boldsymbol{\phi},\boldsymbol{\theta}}(\mathbf{x}) = \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\mathbf{x})}[\log p(\mathbf{x}|\boldsymbol{z},\boldsymbol{\theta})] - KL(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\mathbf{x})||p(\boldsymbol{z}))$$

Распределение $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\mathbf{x})$ зависит от $\boldsymbol{\phi}$ и мы не можем занести градиент внутрь интеграла:

$$\nabla_{\boldsymbol{\theta}}\mathcal{L}_{\boldsymbol{\phi},\boldsymbol{\theta}}(\mathbf{x}) = \nabla_{\boldsymbol{\phi}} \int q_{\boldsymbol{\phi}}(\boldsymbol{z}|\mathbf{x}) \log p(\mathbf{x}|\boldsymbol{z},\boldsymbol{\theta}) d\boldsymbol{z} - \nabla_{\boldsymbol{\phi}} KL(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\mathbf{x})||p(\boldsymbol{z}))$$

$$\neq \int q_{\boldsymbol{\phi}}(\boldsymbol{z}|\mathbf{x}) \nabla_{\boldsymbol{\phi}} \log p(\mathbf{x}|\boldsymbol{z},\boldsymbol{\theta}) d\boldsymbol{z} - \nabla_{\boldsymbol{\phi}} KL(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\mathbf{x})||p(\boldsymbol{z}))$$

Law of the Unconscious Statistician

- Пусть у нас есть случайная величина $\mathbf{x}$ с распределением $p(\mathbf{x})$
- Создадим новую случайную величину $\boldsymbol{y} = \boldsymbol{f}(\mathbf{x})$
- Хотим посчитать матожидание некоторой функции $\boldsymbol{g}$ от нашей новой переменной $\boldsymbol{y}$

Сложный путь:

- Находим распределение $p(\boldsymbol{y})$
- Вычисляем матожидание:

$$\mathbb{E}_{p(\boldsymbol{y})}[\boldsymbol{g}(\mathbf{y})] = \int p(\boldsymbol{y}) \cdot \boldsymbol{g}(\boldsymbol{y}) d\boldsymbol{y}$$

Простой путь (***LOTUS trick***):

$$\mathbb{E}_{p(\boldsymbol{y})}[\boldsymbol{g}(\mathbf{y})] = \mathbb{E}_{p(\mathbf{x})}[\boldsymbol{g}(\boldsymbol{f}(\mathbf{x}))] = \int p(\mathbf{x}) \cdot \boldsymbol{g}(\boldsymbol{f}(\mathbf{x})) d\mathbf{x}$$

Reparameterization trick

Считаем градиент по параметрам $\boldsymbol{\phi}$ при фиксированных $\boldsymbol{\theta}$:

$$\mathcal{L}_{\boldsymbol{\phi},\boldsymbol{\theta}}(\mathbf{x}) = \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\mathbf{x})}[\log p(\mathbf{x}|\boldsymbol{z},\boldsymbol{\theta})] - KL(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\mathbf{x})||p(\boldsymbol{z}))$$

Будем использовать ***трюк репараметризации*** (***reparameterization trick***) – считать матожидание по распределению, которое не зависит от $\boldsymbol{\phi}$

Предположим, что мы можем выразить $\boldsymbol{z}$ из распределения $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\mathbf{x})$ как детерминированную функцию $\boldsymbol{g}$ от некоторой независимой случайной величины $\boldsymbol{\epsilon}$ из распределения $p(\boldsymbol{\epsilon})$:

$$\boldsymbol{z} = \boldsymbol{g}_{\boldsymbol{\phi}}(\mathbf{x},\boldsymbol{\epsilon}) \qquad\qquad \boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon})$$

Пусть $p(\boldsymbol{\epsilon}) = \mathcal{N}(\mathbf{0},\boldsymbol{I})$ и функция репараметризации имеет вид:

$$\boldsymbol{z} = \boldsymbol{g}_{\boldsymbol{\phi}}(\mathbf{x},\boldsymbol{\epsilon}) = \boldsymbol{\mu}_{\boldsymbol{\phi}}(\mathbf{x}) + \boldsymbol{\sigma}_{\boldsymbol{\phi}}(\mathbf{x}) \cdot \boldsymbol{\epsilon}$$

$$q_{\boldsymbol{\phi}}(\boldsymbol{z}|\mathbf{x}) = \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{\phi}}(\mathbf{x}), \boldsymbol{\sigma}_{\boldsymbol{\phi}}(\mathbf{x}))$$

Градиент на E-шаге

Считаем градиент по параметрам $\boldsymbol{\phi}$ при фиксированных $\boldsymbol{\theta}$:

$$\mathcal{L}_{\boldsymbol{\phi},\boldsymbol{\theta}}(\mathbf{x}) = \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\mathbf{x})}[\log p(\mathbf{x}|\boldsymbol{z},\boldsymbol{\theta})] - KL(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\mathbf{x})||p(\boldsymbol{z}))$$

Используем ***reparameterization trick,*** чтобы внести оператор градиента внутрь интеграла*:*

$$\nabla_{\boldsymbol{\phi}} \int q_{\boldsymbol{\phi}}(\boldsymbol{z}|\mathbf{x}) \log p(\mathbf{x}|\boldsymbol{z},\boldsymbol{\theta}) d\boldsymbol{z} = \int p(\boldsymbol{\epsilon})\, \nabla_{\boldsymbol{\phi}} \log p\big(\mathbf{x}\big|\boldsymbol{g}_{\boldsymbol{\phi}}(\mathbf{x},\boldsymbol{\epsilon}),\boldsymbol{\theta}\big)\, d\boldsymbol{\epsilon}$$

Оценим этот интеграл с помощью метода Монте-Карло:

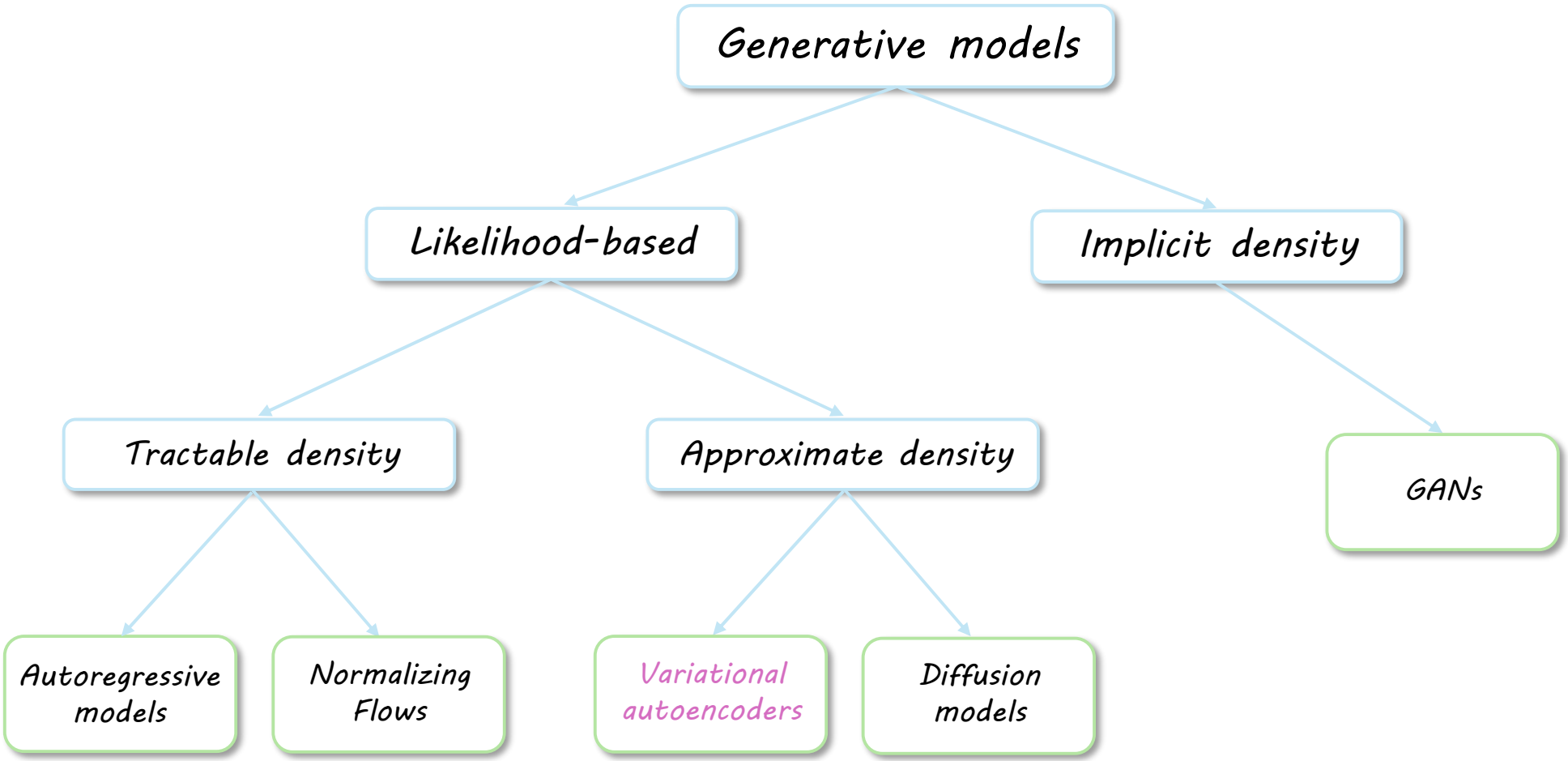
$$\int p(\boldsymbol{\epsilon})\, \nabla_{\boldsymbol{\phi}} \log p\big(\mathbf{x}\big|\boldsymbol{g}_{\boldsymbol{\phi}}(\mathbf{x},\boldsymbol{\epsilon}),\boldsymbol{\theta}\big)\, d\boldsymbol{\epsilon} \approx \nabla_{\boldsymbol{\phi}} \log p\big(\mathbf{x}\big|\boldsymbol{\mu}_{\boldsymbol{\phi}}(\mathbf{x}) + \boldsymbol{\sigma}_{\boldsymbol{\phi}}(\mathbf{x}) \cdot \boldsymbol{\epsilon}^*,\boldsymbol{\theta}\big), \qquad \boldsymbol{\epsilon}^* \sim \mathcal{N}(\mathbf{0},\boldsymbol{I})$$

Для KL –дивергенции между двумя нормальными распределениями существует аналитическая формула*:*

$$\nabla_{\boldsymbol{\phi}}\, KL(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\mathbf{x})||p(\boldsymbol{z})) = \nabla_{\boldsymbol{\phi}}\, KL(\mathcal{N}\left(\boldsymbol{\mu}_{\boldsymbol{\phi}}(\mathbf{x}),\boldsymbol{\sigma}_{\boldsymbol{\phi}}(\mathbf{x})\right)||\mathcal{N}(\mathbf{0},\boldsymbol{I}))$$

Variational Autoencoder

Зоопарк генеративных моделей

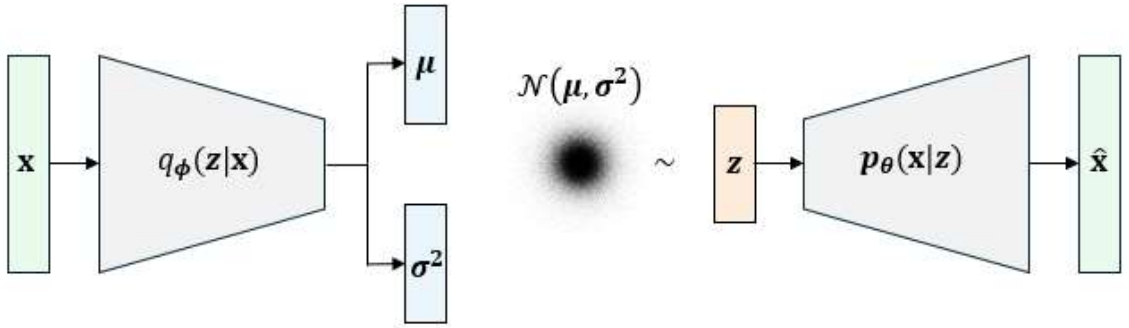


Variational Autoencoder

- ***VAE*** состоит из двух нейросетей*:* ***кодировщика*** и ***декодировщика***

Общая идея:

- Одному объекту соответствует не одна точка, а целое распределение $q_{\phi}(\boldsymbol{z}|\mathbf{x})$
- Кодировщик $q_{\phi}(\boldsymbol{z}|\mathbf{x})$ выдает не один вектор, а два – вектор средних $\boldsymbol{\mu}$ и дисперсий $\boldsymbol{\sigma^2}$
- Из этого распределения берется случайная точка $\boldsymbol{z}$, которая затем подается в декодер для восстановления

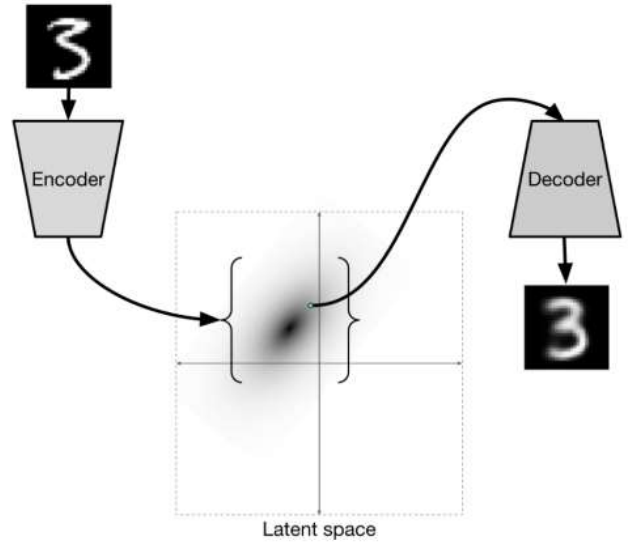


Функция потерь

Обучение VAE – поиск компромисса между двумя членами:

$$\text{Loss} = -\underbrace{\mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\mathbf{x})}[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\boldsymbol{z})]}_{Reconstruction\ term} + \underbrace{KL(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\mathbf{x})||p(\boldsymbol{z}))}_{Regularization\ term}$$

- ***Reconstruction term*** – насколько хорошо декодер восстанавливает $\mathbf{x}$ из $\boldsymbol{z}$
- ***Regularization term*** – насколько распределение $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\mathbf{x})$ похоже на априорное $p(\boldsymbol{z})$, заставляет скрытое пространство быть гладким и хорошо организованным



Функция потерь: Reconstruction term

Пусть функция потерь состоит только из ***Reconstruction term***:

$$\text{Loss} = -\mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\mathbf{x})}[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\boldsymbol{z})]$$

- У модели одна цель – восстановить $\mathbf{x}$ из $\boldsymbol{z}$ как можно точнее
- Чтобы помочь декодеру, кодировщик $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\mathbf{x})$ будет делать $\boldsymbol{z}$ уникальным для каждого $\mathbf{x}$
- Самый простой способ сделать это – схлопнуть $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\mathbf{x})$ в одну точку $\rightarrow \boldsymbol{\sigma^2 = 0}$

Итог:

- ***Восстановление***: наилучшее
- ***Скрытое пространство***: разрушено, поскольку $\boldsymbol{z}$ будут далеко разбросаны по всему пространству
- ***Генерация***: плохая

Функция потерь: Regularization term

Пусть функция потерь состоит только из ***Regularization term***:

$$\text{Loss} = KL(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\mathbf{x})||p(\boldsymbol{z}))$$

- У модели одна цель – сделать $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\mathbf{x})$ неотличимым от $p(\boldsymbol{z})$
- Самый простой способ сделать это – игнорировать $\mathbf{x}$ и всегда выдавать $\boldsymbol{\mu} = \mathbf{0}$ и $\boldsymbol{\sigma}^2 = \mathbf{1}$

Итог:

- ***Восстановление***: плохое, $\boldsymbol{z}$ не будет содержать никакой информации об $\mathbf{x}$
- ***Скрытое пространство***: идельное, но бесполезное
- ***Генерация***: плохая

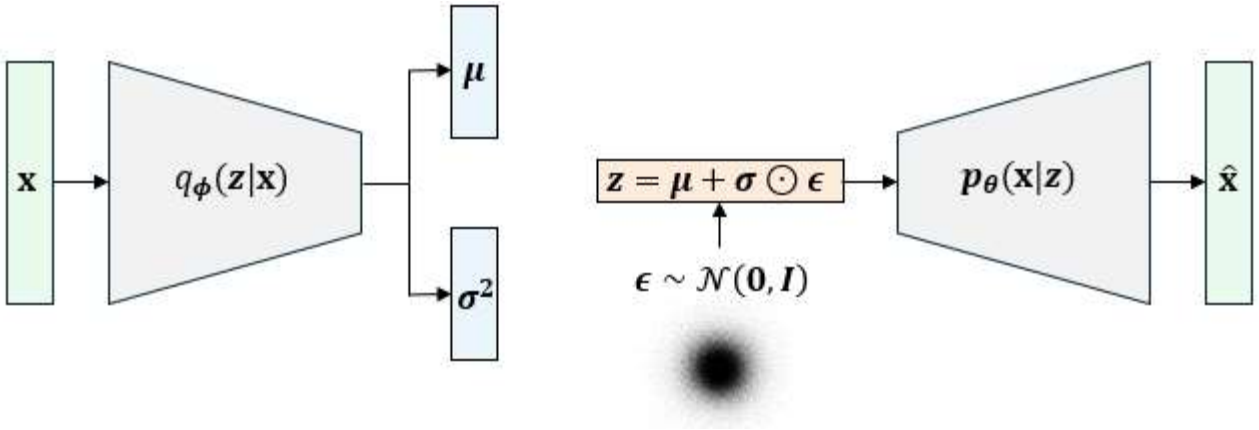
Обучение VAE

Алгоритм:

- Берем случайный объект $\mathbf{x}$
- Пропускаем его через кодировщик $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\mathbf{x})$, получаем $\boldsymbol{\mu}_{\boldsymbol{\phi}}(\mathbf{x})$ и $\boldsymbol{\sigma}_{\boldsymbol{\phi}}(\mathbf{x})$
- Вычисляем $\boldsymbol{z}^*$ с помощью репараметризации:

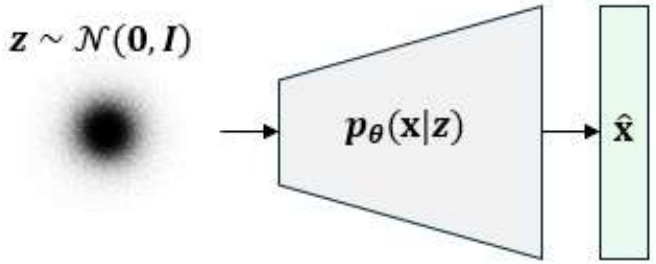
  Берем $\boldsymbol{\epsilon}^* \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
  Считаем $\boldsymbol{z}^* = \boldsymbol{\mu}_{\boldsymbol{\phi}}(\mathbf{x}) + \boldsymbol{\sigma}_{\boldsymbol{\phi}}(\mathbf{x}) \cdot \boldsymbol{\epsilon}^*$

- Пропускаем $\boldsymbol{z}^*$ через декодер и получаем восстановленный $\hat{\mathbf{x}}$
- Считаем лосс, обновляем параметры $\boldsymbol{\phi}$ и $\boldsymbol{\theta}$



Генерация VAE

- Кодировщик $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\mathbf{x})$ больше не нужен
- Берем случайный вектор $\boldsymbol{z}^*$ из априорного распределения $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$
- Подаем $\boldsymbol{z}^*$ на вход обученному декодеру



VAE и нормализующие потоки

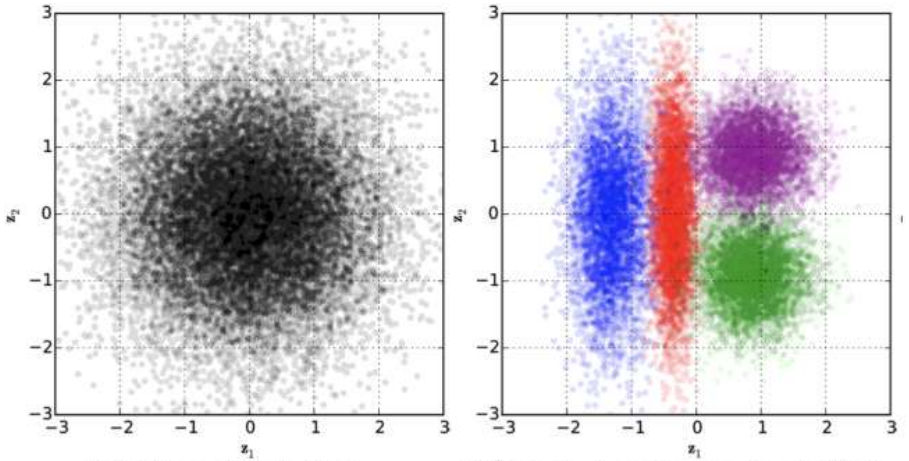
	VAE	Flow
Loss	$ELBO\ \mathcal{L}$	$MLE\ /\ Forward\ KL$
Encoder	$Stochastic\ \mathbf{z} \sim q_{\boldsymbol{\phi}}(\mathbf{z}\|\mathbf{x})$	$Deterministic\ \mathbf{z} = \boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x})$
Decoder	$Stochastic\ \mathbf{x} \sim p_{\boldsymbol{\theta}}(\mathbf{x}\|\mathbf{z})$	$Deterministic\ \mathbf{x} = \boldsymbol{g}_{\boldsymbol{\theta}}(\mathbf{z})$
Parameters	$\boldsymbol{\theta}, \boldsymbol{\phi}$	$\boldsymbol{\theta}$

ELBO Surgery

Проблемы VAE

- VAE часто генерирует размытые, усредненные изображения

- Основная причина – несоответствие между априорным распределением $p(\boldsymbol{z})$ и распределением, которое выучивает кодировщик
- Мы предполагаем, что скрытое пространство должно быть простым $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$
- Кодировщик на сложных данных будет выучивать сложное, мультимодальное распределение



(a) Prior distribution (b) Posteriors in standard VAE

ELBO Surgery

- Попробуем найти ***среднюю* KL *−дивергенцию*** по всей выборке
- Введем среднее вариационное апостериорное распределение $q_{avg}(\boldsymbol{z}|\boldsymbol{\phi}) = \frac{1}{n}\sum_{i=1}^{n} q_{\boldsymbol{\phi}}(\boldsymbol{z}_i|\mathbf{x})$

$$\frac{1}{n}\sum_{i=1}^{n} KL(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\mathbf{x}_\mathrm{i})||p(\boldsymbol{z})) = \frac{1}{n}\sum_{i=1}^{n}\int q_{\boldsymbol{\phi}}(\boldsymbol{z}|\mathbf{x}_i)\log\frac{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\mathbf{x}_\mathrm{i})}{p(\boldsymbol{z})}d\boldsymbol{z} =$$

$$\frac{1}{n}\sum_{i=1}^{n}\int q_{\boldsymbol{\phi}}(\boldsymbol{z}|\mathbf{x}_\mathrm{i})\log\frac{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\mathbf{x}_\mathrm{i})\cdot q_{avg}(\boldsymbol{z}|\boldsymbol{\phi})}{p(\boldsymbol{z})\cdot q_{avg}(\boldsymbol{z}|\boldsymbol{\phi})}d\boldsymbol{z} = \int\frac{1}{n}\sum_{i=1}^{n} q_{\boldsymbol{\phi}}(\boldsymbol{z}|\mathbf{x}_\mathrm{i})\log\frac{q_{avg}(\boldsymbol{z}|\boldsymbol{\phi})}{p(\boldsymbol{z})}d\boldsymbol{z} +$$

$$\frac{1}{n}\sum_{i=1}^{n}\int q_{\boldsymbol{\phi}}(\boldsymbol{z}|\mathbf{x}_\mathrm{i})\log\frac{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\mathbf{x}_\mathrm{i})}{q_{avg}(\boldsymbol{z}|\boldsymbol{\phi})}d\boldsymbol{z} = KL(q_{avg}(\boldsymbol{z}|\boldsymbol{\phi})||p(\boldsymbol{z})) + \frac{1}{n}\sum_{i=1}^{n} KL(q_{\boldsymbol{\phi}}(\boldsymbol{z}_i|\mathbf{x})||q_{avg}(\boldsymbol{z}|\boldsymbol{\phi}))$$

$$KL(q_{avg}(\boldsymbol{z}|\boldsymbol{\phi})||p(\boldsymbol{z})) + \mathbb{I}_q[\mathbf{x},\boldsymbol{z}]$$

ELBO Surgery

- ***Средняя* KL *−дивергенция*** по всей выборке раскладывается на 2 члена:

$$\frac{1}{n}\sum_{i=1}^{n} KL(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\mathbf{x}_\mathrm{i})||p(\boldsymbol{z})) = KL(q_{avg}(\boldsymbol{z}|\boldsymbol{\phi})||p(\boldsymbol{z})) + \mathbb{I}_q[\mathbf{x}, \boldsymbol{z}]$$

- $q_{avg}(\boldsymbol{z}|\boldsymbol{\phi})-$ среднее распределение, которое кодировщик выдает по всей выборке
- $KL(q_{avg}(\boldsymbol{z}|\boldsymbol{\phi})||p(\boldsymbol{z}))-$ маргинальная KL −дивергенция
- взаимная информация (***Mutual Information***)

- ***Marginal KL*** заставляет $q_{avg}(\boldsymbol{z}|\boldsymbol{\phi})$ быть похожим на $p(\boldsymbol{z})$

- ***Mutual Information*** отвечает за качество кодирования и побуждает $\boldsymbol{z}$ содержать как можно больше информации об $\mathbf{x}$

Optimal Prior

ELBO по всей выборке*:*

$$\frac{1}{n}\sum_{i=1}^{n}\mathcal{L}_{\boldsymbol{\phi},\boldsymbol{\theta}}(\mathbf{x}_i) = \frac{1}{n}\sum_{i=1}^{n}\left(\mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\mathbf{x}_i)}[\log p_{\boldsymbol{\theta}}(\mathbf{x}_i|\boldsymbol{z})] - KL(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\mathbf{x}_i)||p(\boldsymbol{z}))\right)$$

$$\frac{1}{n}\sum_{i=1}^{n}\mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\mathbf{x}_i)}[\log p_{\boldsymbol{\theta}}(\mathbf{x}_i|\boldsymbol{z})] - KL(q_{avg}(\boldsymbol{z}|\boldsymbol{\phi})||p(\boldsymbol{z})) - \mathbb{I}_q[\mathbf{x},\boldsymbol{z}]$$

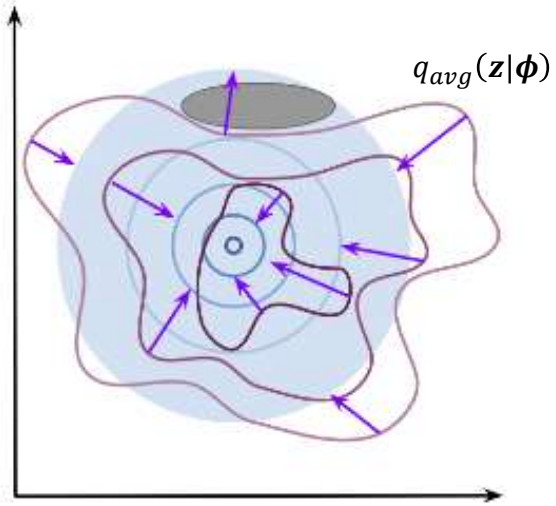
- $KL(q_{avg}(\boldsymbol{z}|\boldsymbol{\phi})||p(\boldsymbol{z})) = 0,$ когда $p(\boldsymbol{z}) = q_{avg}(\boldsymbol{z}|\boldsymbol{\phi})$

- Оптимальное априорное распределение $p(\boldsymbol{z})$ – это не $\mathcal{N}(\mathbf{0},\boldsymbol{I})$, а сложное, мультимодальное среднее апостериорное распределение, которое выучивает кодировщик*:*

$$p(\boldsymbol{z}) = q_{avg}(\boldsymbol{z}|\boldsymbol{\phi}) = \frac{1}{n}\sum_{i=1}^{n}q_{\boldsymbol{\phi}}(\boldsymbol{z}_i|\mathbf{x})$$

Проблемы обычного *Prior*

- В VAE мы заставляем $q_{avg}(\boldsymbol{z}|\boldsymbol{\phi})$ соответствовать простому распределению $p(\boldsymbol{z}) = \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
- Это приводит к ***чрезмерной регуляризации*** (***over –***



Обучаемый *Prior*

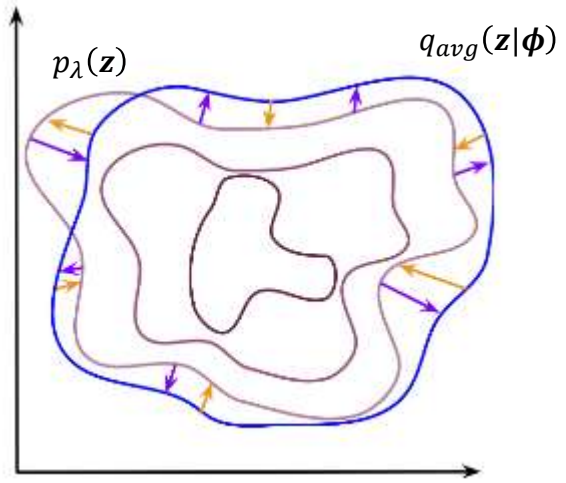
- Оптимальное априорное распределение – это среднее апостериорное распределение:

$$p(\boldsymbol{z}) = q_{avg}(\boldsymbol{z}|\boldsymbol{\phi}) = \frac{1}{n}\sum_{i=1}^{n} q_{\boldsymbol{\phi}}(\boldsymbol{z}_i|\mathbf{x})$$

- Такой подход ведёт к ***переобучению*** (***overfitting***) и вычислительно невозможен

- Будем использовать ***обучаемый prior*** $p_{\boldsymbol{\lambda}}(\boldsymbol{z})$

- Теперь не только кодировщик подстраивается под *prior*, но и *prior* под кодировщик



Prior на основе нормализующих потоков

Чтобы получить гибкий обучаемый *prior*, можем использовать нормализующие потоки:

$$\log p_{\boldsymbol{\lambda}}(\mathbf{x}) = \log p(\boldsymbol{z}^*) + \log \left| \det \left(\frac{\partial \boldsymbol{z}^*}{\partial \boldsymbol{z}} \right) \right| = \log p(\boldsymbol{f}_{\boldsymbol{\lambda}}(\boldsymbol{z})) + \log |\det(\boldsymbol{J}_{\boldsymbol{f}})|$$

В *ELBO* нужно заменить фиксированный *prior* на обучаемый:

$$\mathcal{L}_{\boldsymbol{\phi},\boldsymbol{\theta},\boldsymbol{\lambda}}(\mathbf{x}) = \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\mathbf{x})}\left[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\boldsymbol{z}) + \log p_{\lambda}(\mathbf{x}) - \log q_{\boldsymbol{\phi}}(\boldsymbol{z}|\mathbf{x})\right] =$$

$$\mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\mathbf{x})}\left[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\boldsymbol{z}) + \log p(\boldsymbol{f}_{\lambda}(\boldsymbol{z})) + \log |\det(\boldsymbol{J}_{\boldsymbol{f}})| - \log q_{\boldsymbol{\phi}}(\boldsymbol{z}|\mathbf{x})\right]$$

Спасибо за внимание!