Лекция 10

Продвинутое глубокое обучение: генеративные модели

Дополнительные главы машинного обучения

Андрей Фильченков

14.05.2021

План лекции

- Генерация и качество генерации
- GANы и их проблемы
- Задача оптимальной транспортировки и функции потерь
- Интересные идеи в GANах
- Автоэнкодеры
- Интересные идеи в AE

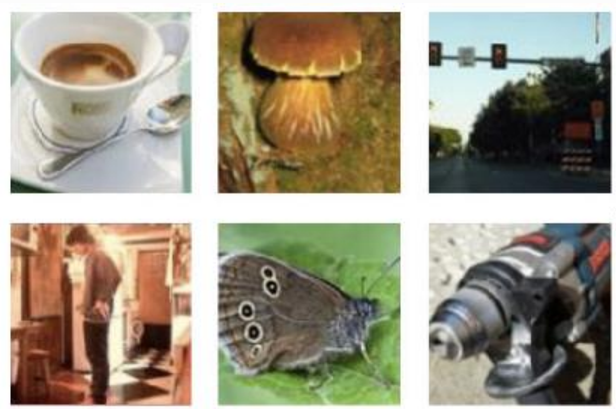
- В презентации используются материалы:
  - F.F. Li et al.’ курс “Convolutional Neural Networks for Visual Recognition”
  - А.С. Артамонов ”Image and video analysis”
- Слайды доступны: **shorturl.at/wGV59**
- Видео доступны: **shorturl.at/ovBTZ**

План лекции

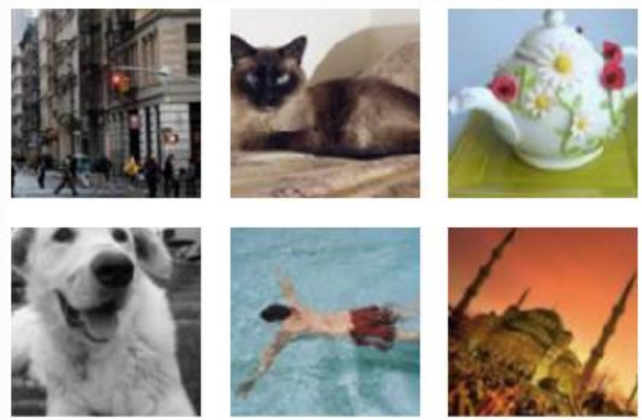
- Генерация и качество генерации
- GANы и их проблемы
- Задача оптимальной транспортировки и функции потерь
- Интересные идеи в GANах
- Автоэнкодеры
- Интересные идеи в AE

Задача генерации (напоминание)

По заданной выборке требуется сгенерировать новые образцы из того же распределения



Обучающая выборка $p_{\text{data}}(x)$



Сгенерированная выборка $p_{\text{model}}(x)$

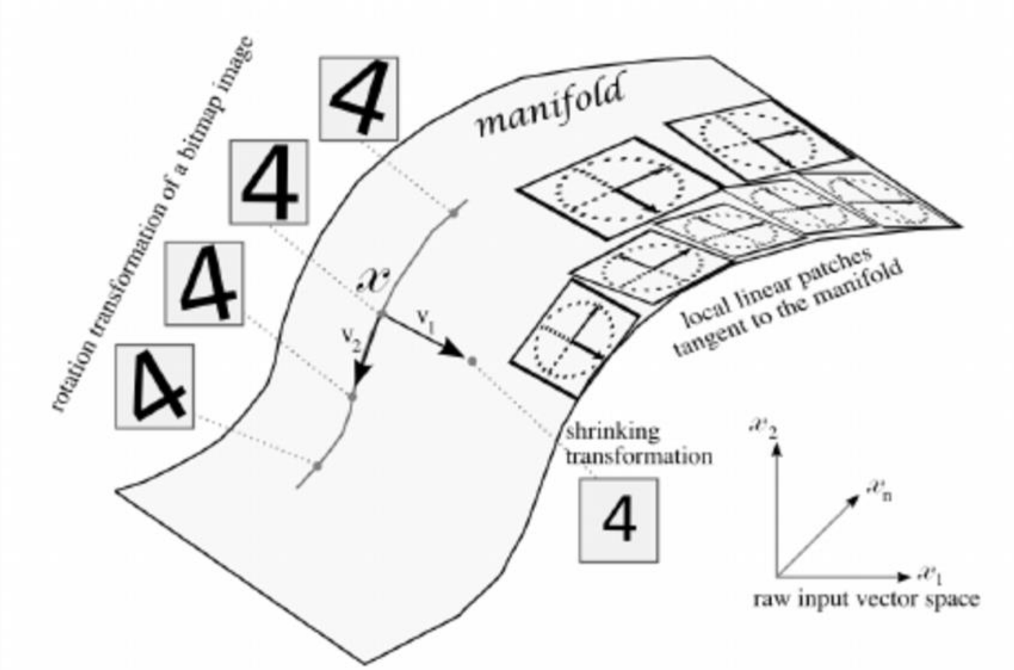
$p_{\text{model}}(x)$ должно быть похоже на $p_{\text{data}}(x)$

Сложности (напоминание)

- Оценка плотности распределения — основная проблема обучения без учителя (да и статистики в целом)
- Чем выше размерность пространства, тем сложнее восстанавливать многомерные распределения

Гипотеза многообразия

Многомерные данные из реального мира лежат в низкоразмерном многообразии внутри соответствующего многомерного пространства



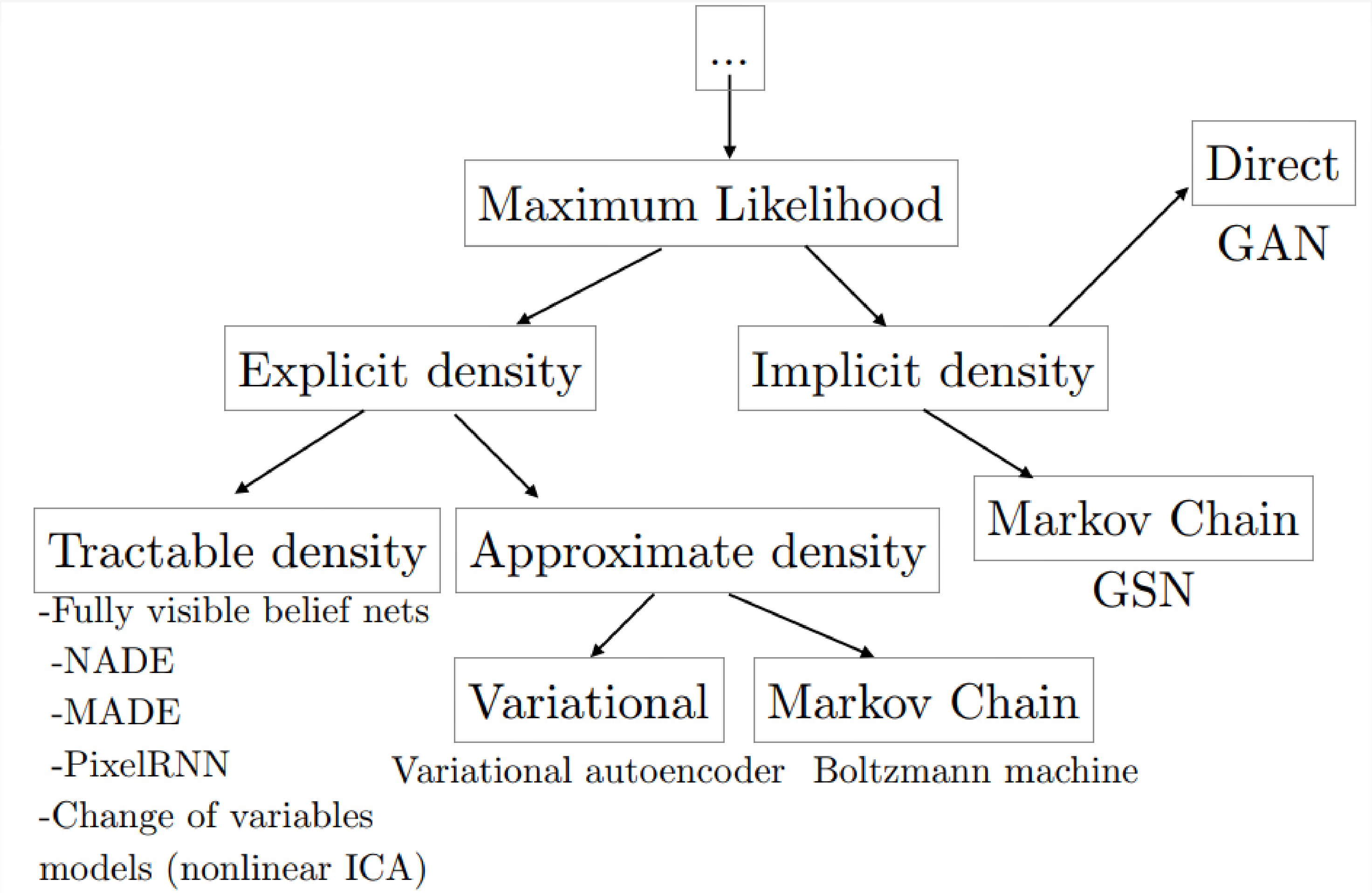
Два способа (напоминание)

Два основных способа:

- **Непосредственная оценка плотности**: непосредственное определение и решение для $p_{\text{model}}(x)$
- **Неявная оценка плотности**: модель, которая может выполнять выборку из $p_{\text{model}}(x)$ без ее непосредственного определения

Чем выше размерность пространства, тем сложнее восстанавливать многомерные распределения

Таксономия генеративных моделей (напоминание)



Проблема оценки качества моделей

Нужно

- сравнивать модели между собой
- сравнивать конфигурации моделей
- выбирать итоговые изображения

Как это сделать?

Inception score

- Максимизируем качество сгенерированных изображений: $p(y|x = G(z))$ должно быть хорошо предсказуемо.
- Максимизируем разнообразие:

$$\int_z p(y|x = G(z))dz$$

- Комбинируем:

$$\mathrm{IS}(G) = \exp\left(\mathrm{E}_{x\sim G} D_{KL}\big(p(y|x)||p(y)\big)\right)$$

Для оценки $p(y|x = G(z))$ используем обученную сеть Inception (отсюда название).

Fréchet Inception Distance (FID)

Измеряем расстояние между картами признаков у настоящих и сгенерированных изображений, предполагая, что они распределены по Гауссу:

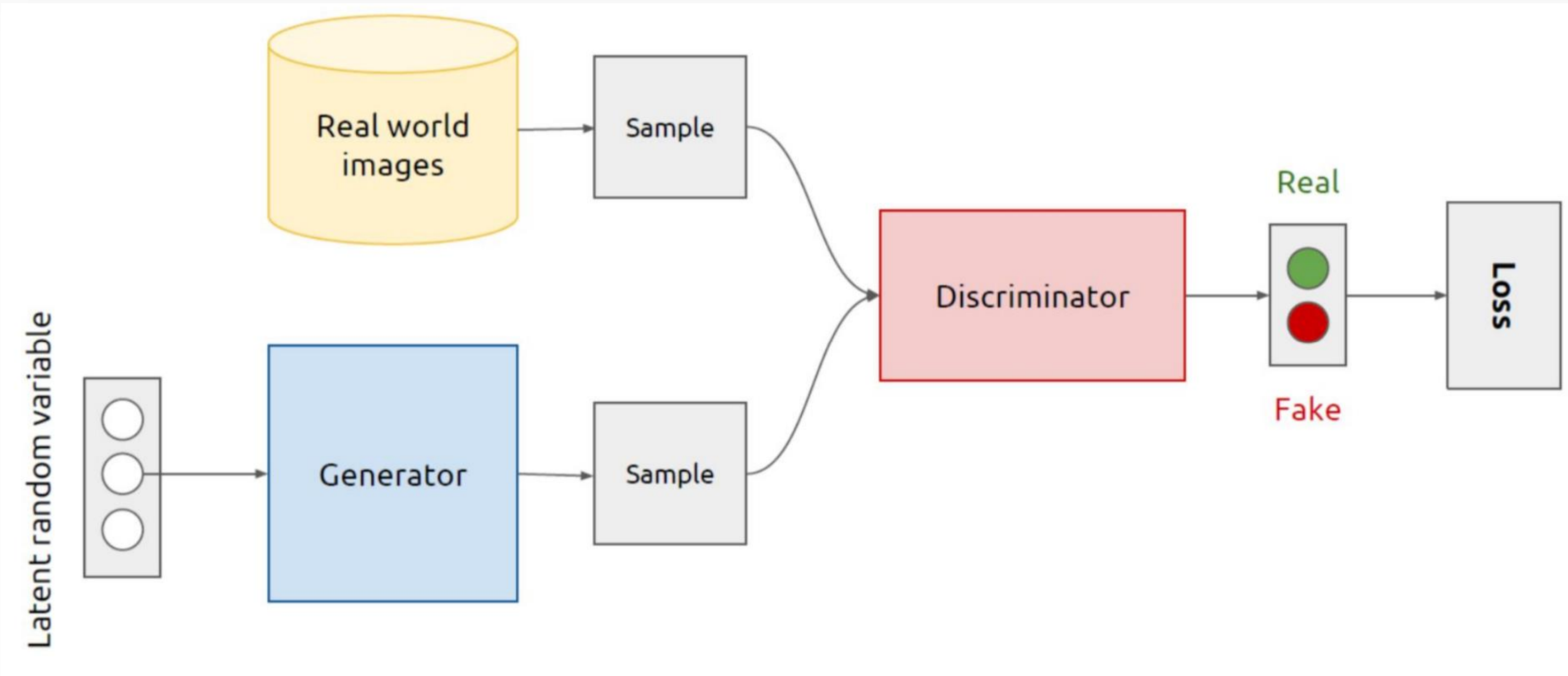
$$\mathrm{FID}(x, g) = \left\|\mu_x - \mu_g\right\|_2^2 + \mathrm{Tr}\left(\Sigma_x + \Sigma_g - 2\left(\Sigma_x\Sigma_g\right)^{\frac{1}{2}}\right),$$

где $A_x \sim N(\mu_x, \Sigma_x)$ и $A_g \sim N\left(\mu_g, \Sigma_g\right)$ — 2048-мерные активационные функции Inception-v3 pool3, а Tr — след матрицы.

План лекции

- Генерация и качество генерации
- GANы и их проблемы
- Задача оптимальной транспортировки и функции потерь
- Интересные идеи в GANах
- Автоэнкодеры
- Интересные идеи в AE

GANы (напоминание)



Минимаксная игра (напоминание)

Можно обучать совместно в постановке минимаксной игры

Минимаксная целевая функция:

$$\min_{\theta_g} \max_{\theta_d} \left[\mathrm{E}_{x \sim p_{\text{data}}} \log D_{\theta_d}(x) + \mathrm{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

где D_{θ_d} — дискриминатор с параметрами θ_d
(пытается максимизировать целевую функцию так, и сделать так, чтобы $D(x)$ был близок к 1 (настоящий) и $D(G(z))$ был близок к 0 (сгенерированный))

и G_{θ_g} — генератор с параметрами θ_g
(пытается минимизировать целевую функцию и сделать D(G(z)) is близким к 1)

Поочередное обучение (напоминание)

Поочередно:

1. **Градиентный подъем** по дискриминатору

$$\max_{\theta_d}\left[\mathrm{E}_{x\sim p_{\text{data}}}\log D_{\theta_d}(x)+\mathrm{E}_{z\sim p(z)}\log(1-D_{\theta_d}(G_{\theta_g}(z)))\right]$$

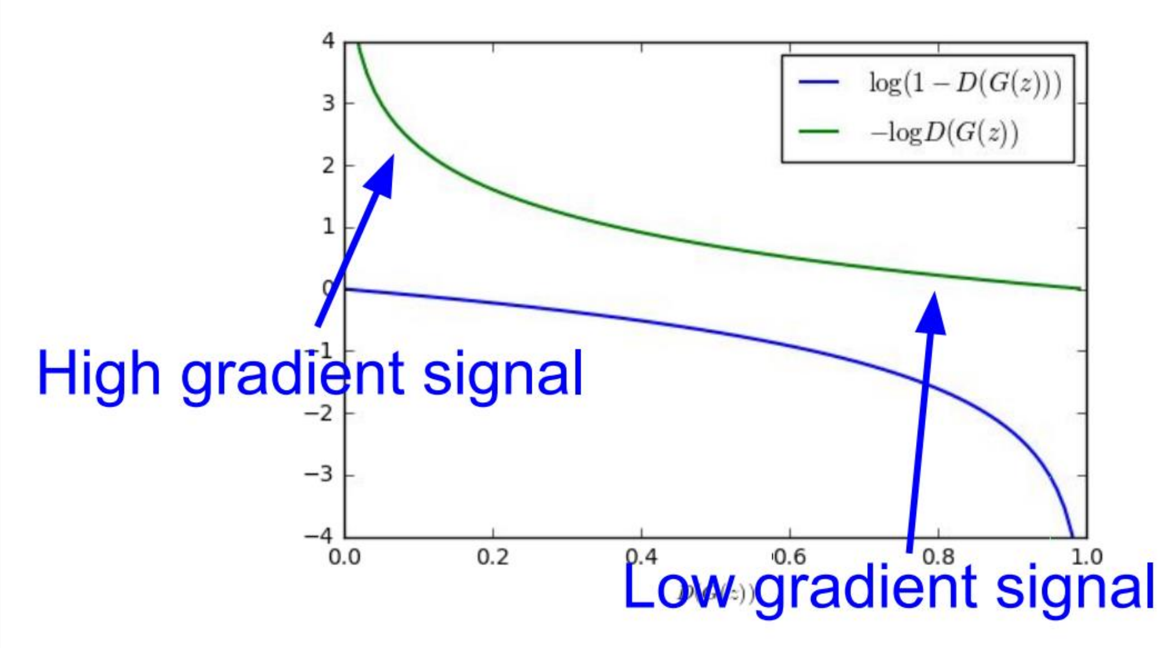
2. **Градиентный спуск** по генератору

$$\min_{\theta_g}\mathrm{E}_{z\sim p(z)}\log(1-D_{\theta_d}(G_{\theta_g}(z)))$$

Изменение обучения генератора (напоминание)

Вместо этого попробуем **градиентный подъем** на генераторе

$$\max_{\theta_g} \mathrm{E}_{z\sim p(z)} \log(D_{\theta_d}(G_{\theta_g}(z)))$$



Conditional GANs (напоминание)

Идея: добавить несколько меток, чтобы дискриминатор мог работать как классификатор по отношению к некоторым меткам.

Тогда

- разное распределение для каждого класса
- целевая функция выглядит так:

$$\min_{\theta_g} \max_{\theta_d} \left[\mathrm{E}_{x \sim p_{\text{data}}} \log D_{\theta_d}(x, y) + \mathrm{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z, y), y)) \right]$$

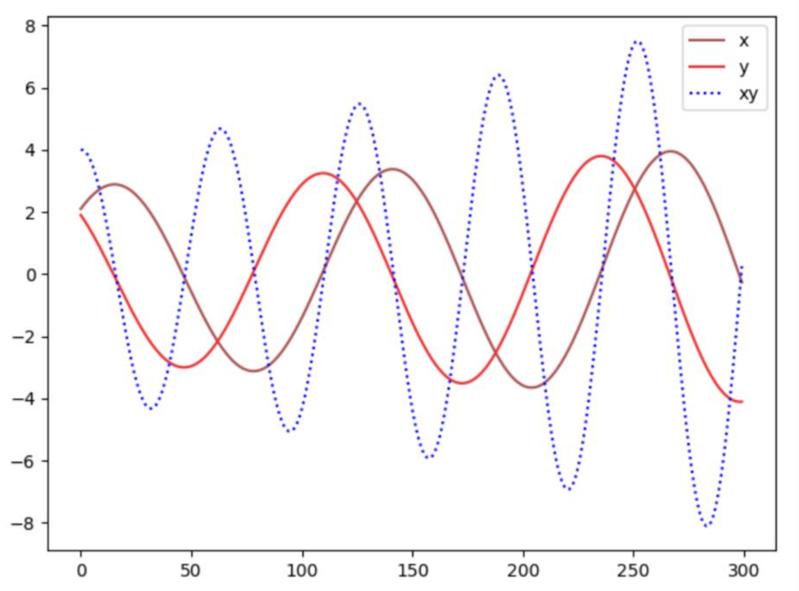
Проблемы GANов

Проблемы GANов

- Нет гарантий сходимости
- Осцилляция
- Слишком сильный дискриминатор (исчезающий градиент)
- Схлопывание мод распределения (mode collapse)
- KL-дивергенция это не лучшая функция для оптимизации

Гарантии сходимости

Пусть игрок А контролирует значение x и хочет максимизировать xy, а игрок B контролирует значение y и хочет минимизировать xy



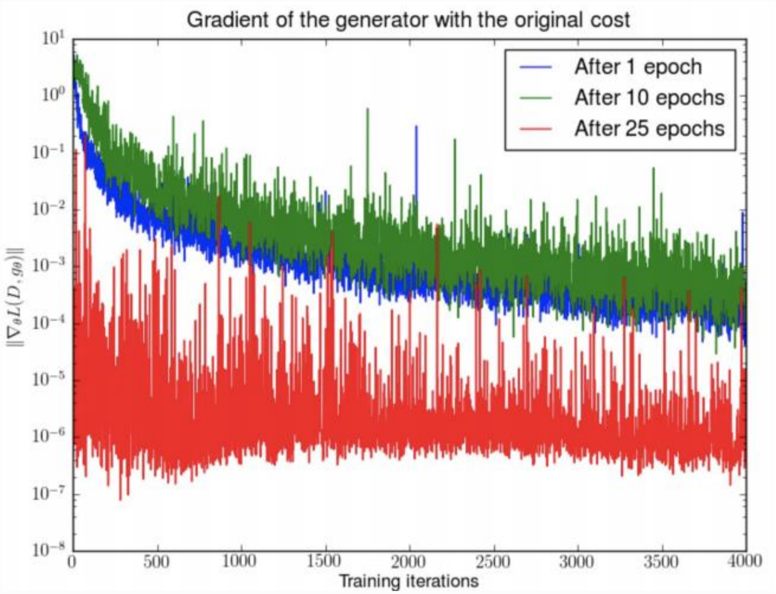
Градиентный спуск не гарантирует сходимости

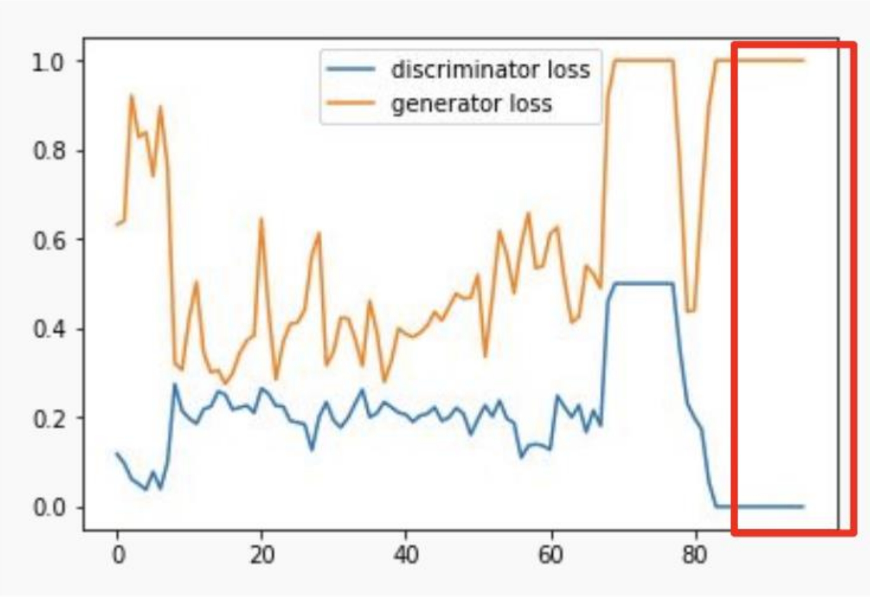
Борьба с осцилляцией

- Подбор гиперпараметров
- Уход от постановки минимаксной игры
- Сопоставление карт признаков (feature matching)
- Историческое усреднение (historical averaging)

Затухание градиента

Слишком сильный дискриминатор тормозит обучение генератора



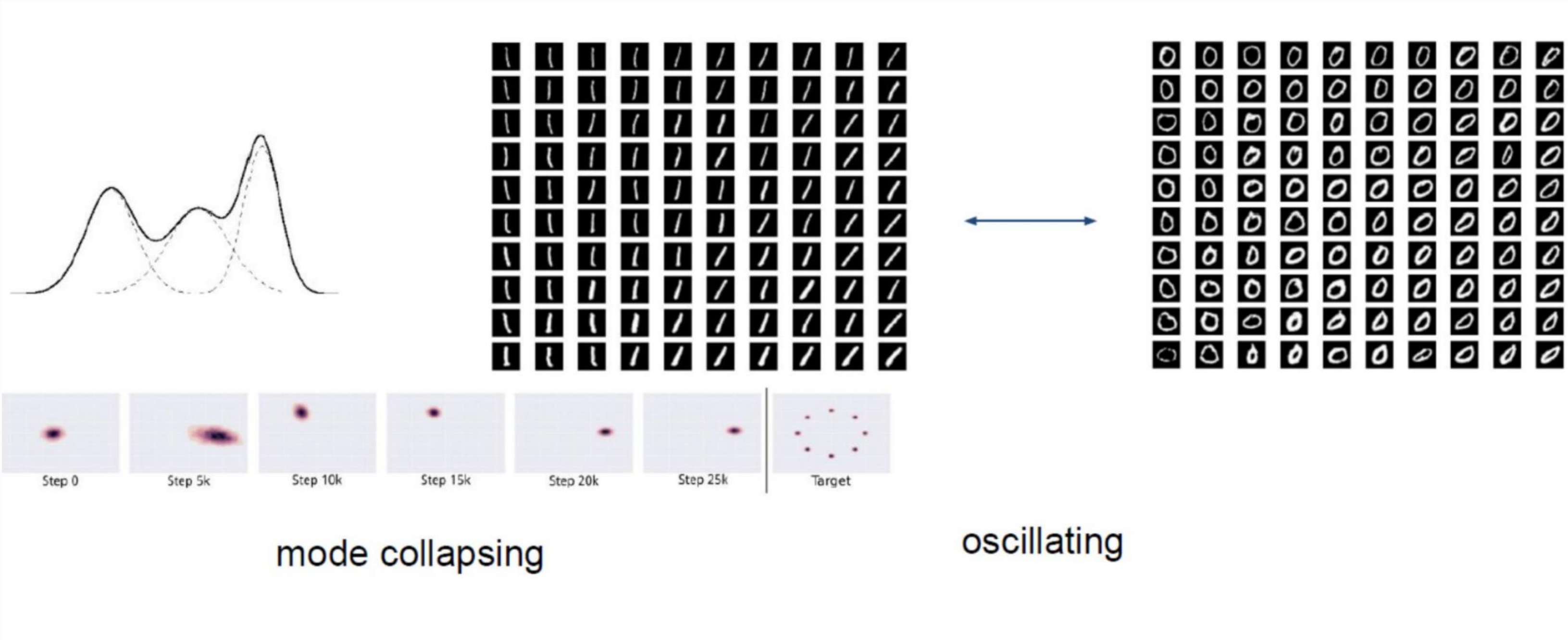


Борьба с затуханием градиента

- Добавление шума дискриминатору
- Торможение обучения дискриминатора

Mode collapse

Генератор воспроизводит только некоторые из мод распределения



Борьба с mode collapse

- Множественные генераторы (AdaGAN, MAD-GAN)
- Сопоставление карт признаков
- Минибатчевая дискриминация
- CGANы

Другие полезные трюки

- Одностороннее сглаживание меток (one-side label smoothing)
- Experience reply
- Батчевая нормализация
- Спектральная нормализация

Дивергенция Йенсена-Шеннона

Задача минимизации

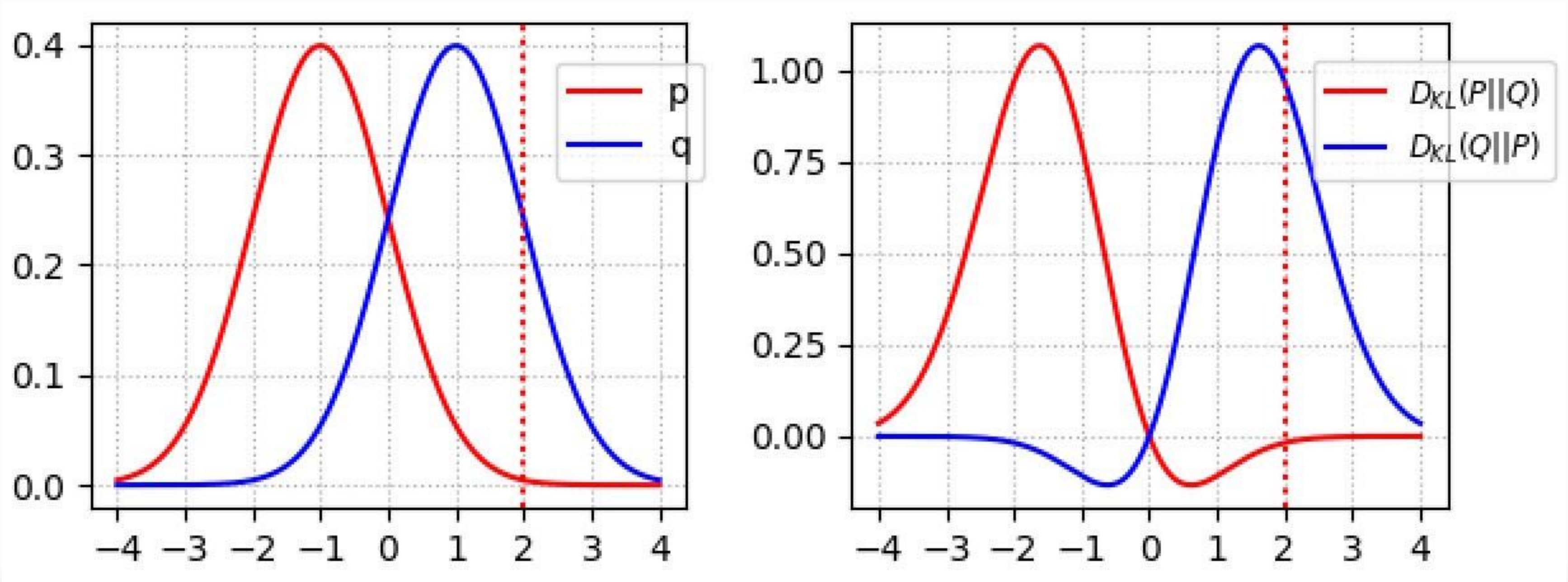
$$\min_{\theta_g} \max_{\theta_d} \left[\mathrm{E}_{x \sim p_{\text{data}}} \log D_{\theta_d}(x) + \mathrm{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

эквивалентна минимизации дивергенции Йенсена-Шеннона

$$D_{KL}\left(p_{data} || \frac{p_{data} + p_G}{2}\right) + D_{KL}\left(p_G || \frac{p_{data} + p_G}{2}\right)$$

Что не так с KL-дивергенцией

- Иногда бывает равна 0
- Не совсем точно отображает наши ожидания



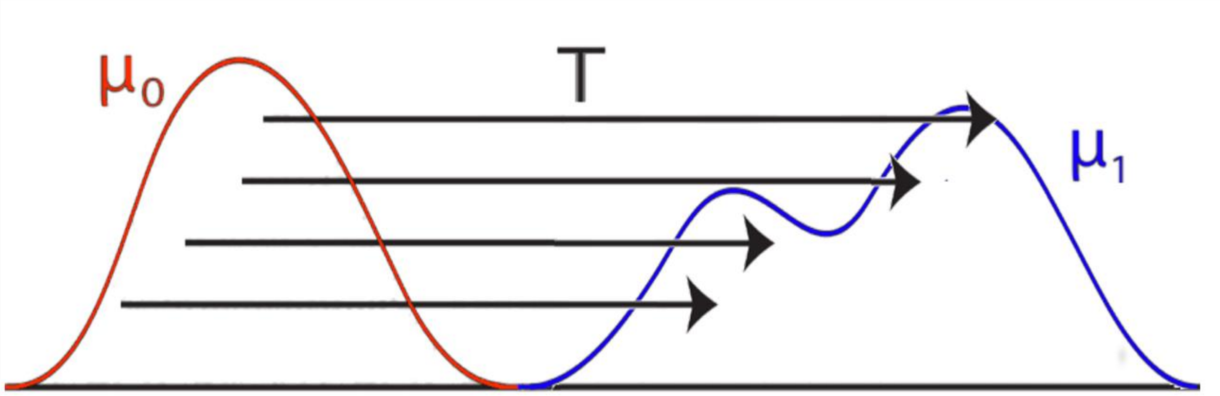
План лекции

- Генерация и качество генерации
- GANы и их проблемы
- Задача оптимальной транспортировки и функции потерь
- Автоэнкодеры
- Интересные идеи в AE

Задача оптимального перемещения

Известна как задача Монжа – Канторовича

Задача Монжа: переместить оптимальным способом одну кучу в земли в другую

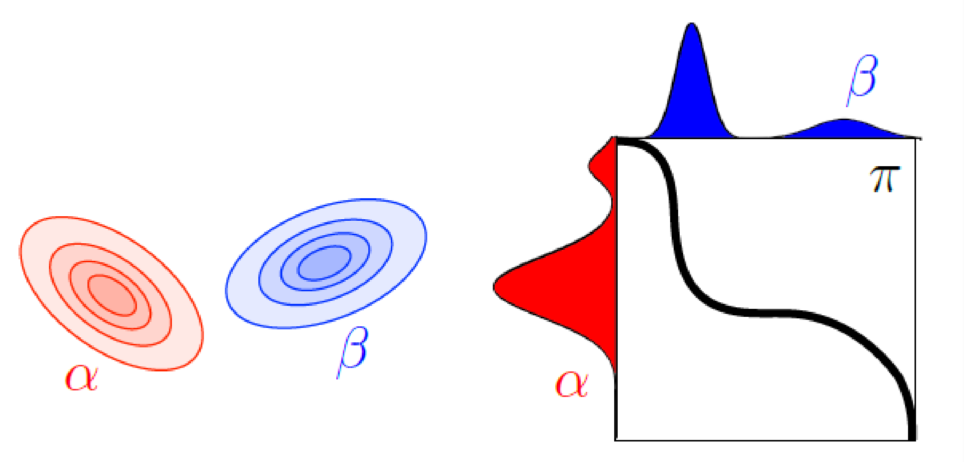


В непрерывном случае:

$$\min_{T}\left\{\int c(x,T(x))d\alpha(x) : T_{\#}(\alpha)=\beta\right\}$$

Задача Монжа-Канторовича

Найти оптимальный план перемещения между двумя мерами



Расстояние Вассерштейна

Расстояние Вассерштейна:

$$W(p_{\text{data}}, p_{\text{model}}) = \inf_{\gamma \in \Pi(p_{\text{data}}, p_{\text{model}})} \mathrm{E}_{(x,y)\sim\gamma}\left[||x-y||\right]$$

где $\Pi(p_{\text{data}}, p_{\text{gen}})$ это множество совместных распределений над p_{data} и p_{model}.

Его нельзя посчитать напрямую, но можно найти приблизительное решение. Оно эквивалентно

$$W(p_{\text{data}}, p_{\text{gen}}) = \sup_{|f_L|<1} \left(\mathrm{E}_{x\sim p_{\text{data}}}[f(x)] - \mathrm{E}_{x\sim p_{\text{model}}}[f(x)]\right)$$

Супремум берется во всем функциям с константой Липшица, не большей 1

Wasserstein loss

Будем пользоваться этим функционалом для обучения

	Discriminator/Critic	Generator
GAN	$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)\right]$	$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)$
WGAN	$\nabla_w \frac{1}{m} \sum_{i=1}^{m} \left[f\left(x^{(i)}\right) - f\left(G\left(z^{(i)}\right)\right)\right]$	$\nabla_\theta \frac{1}{m} \sum_{i=1}^{m} f\left(G\left(z^{(i)}\right)\right)$

Для поддержания константы обрежем градиенты (clipping)

WGAN-GP

Вместо обрезания градиента, добавим регуляризацию, зависящую от градиента:

$$L = \underbrace{\mathop{\mathbb{E}}_{\tilde{\boldsymbol{x}} \sim \mathbb{P}_g} [D(\tilde{\boldsymbol{x}})] - \mathop{\mathbb{E}}_{\boldsymbol{x} \sim \mathbb{P}_r} [D(\boldsymbol{x})]}_{\text{Original critic loss}} + \underbrace{\lambda \mathop{\mathbb{E}}_{\hat{\boldsymbol{x}} \sim \mathbb{P}_{\hat{\boldsymbol{x}}}} \left[(\|\nabla_{\hat{\boldsymbol{x}}} D(\hat{\boldsymbol{x}})\|_2 - 1)^2 \right]}_{\text{Our gradient penalty}}.$$

Другие функции потерь

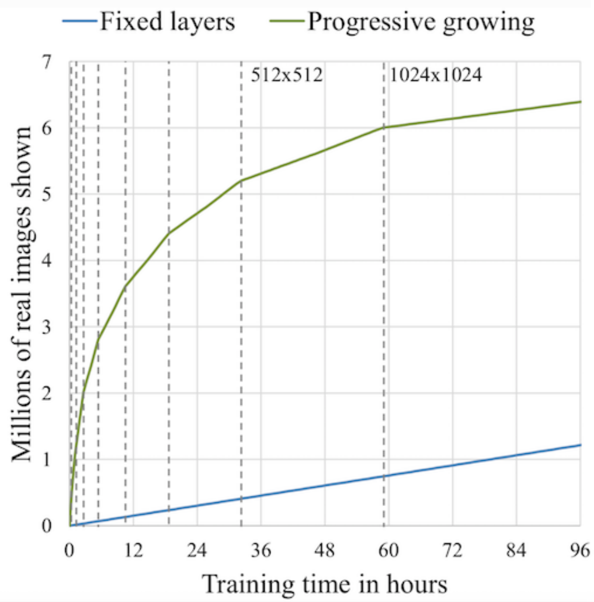
DRAGAN	$L_D^{DRAGAN} = L_D^{GAN} + \lambda E[(\lvert\nabla D(\alpha x - (1 - \alpha x_p))\rvert - 1)^2]$ $L_G^{DRAGAN} = L_G^{GAN}$
CGAN	$L_D^{CGAN} = E[\log(D(x,c))] + E[\log(1 - D(G(z),c))]$ $L_G^{CGAN} = E[\log(D(G(z),c))]$
infoGAN	$L_{D,Q}^{infoGAN} = L_D^{GAN} - \lambda L_I(c,c')$ $L_G^{infoGAN} = L_G^{GAN} - \lambda L_I(c,c')$
ACGAN	$L_{D,Q}^{ACGAN} = L_D^{GAN} + E[P(class = c\lvert x)] + E[P(class = c\lvert G(z))]$ $L_G^{ACGAN} = L_G^{GAN} + E[P(class = c\lvert G(z))]$
EBGAN	$L_D^{EBGAN} = D_{AE}(x) + \max(0, m - D_{AE}(G(z)))$ $L_G^{EBGAN} = D_{AE}(G(z)) + \lambda \cdot PT$
BEGAN	$L_D^{BEGAN} = D_{AE}(x) - k_t D_{AE}(G(z))$ $L_G^{BEGAN} = D_{AE}(G(z))$ $k_{t+1} = k_t + \lambda(\gamma D_{AE}(x) - D_{AE}(G(z)))$

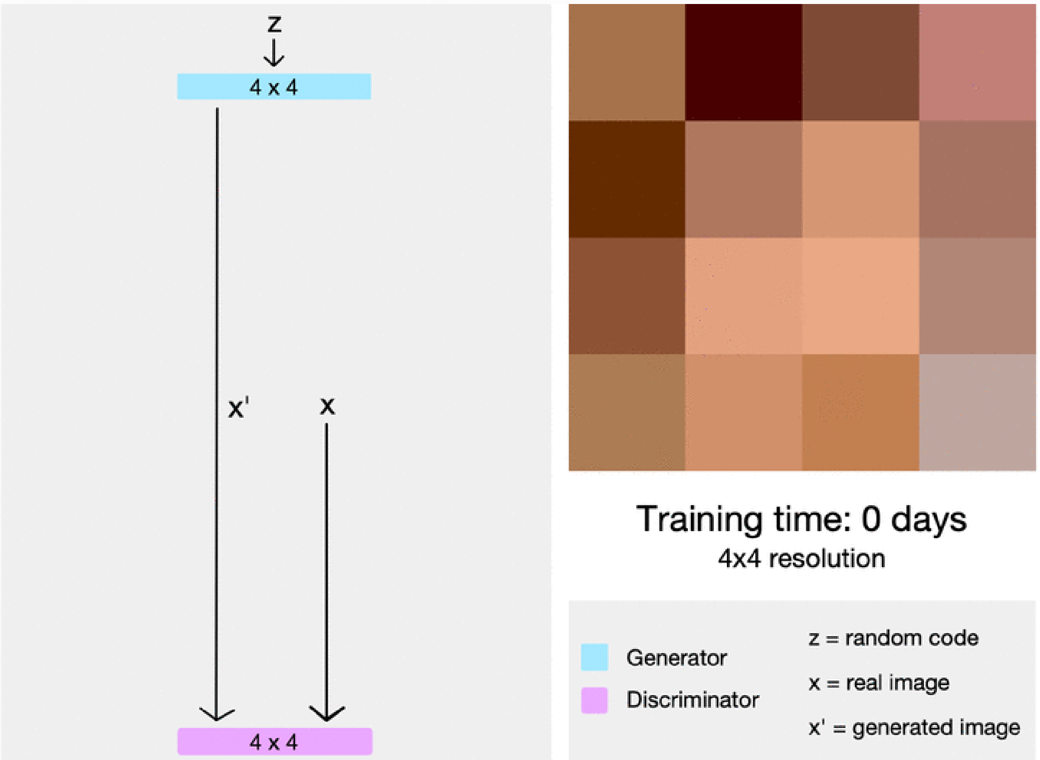
План лекции

- Генерация и качество генерации
- GANы и их проблемы
- Задача оптимальной транспортировки и функции потерь
- Интересные идеи в GANах
- Автоэнкодеры
- Интересные идеи в AE

ProGAN

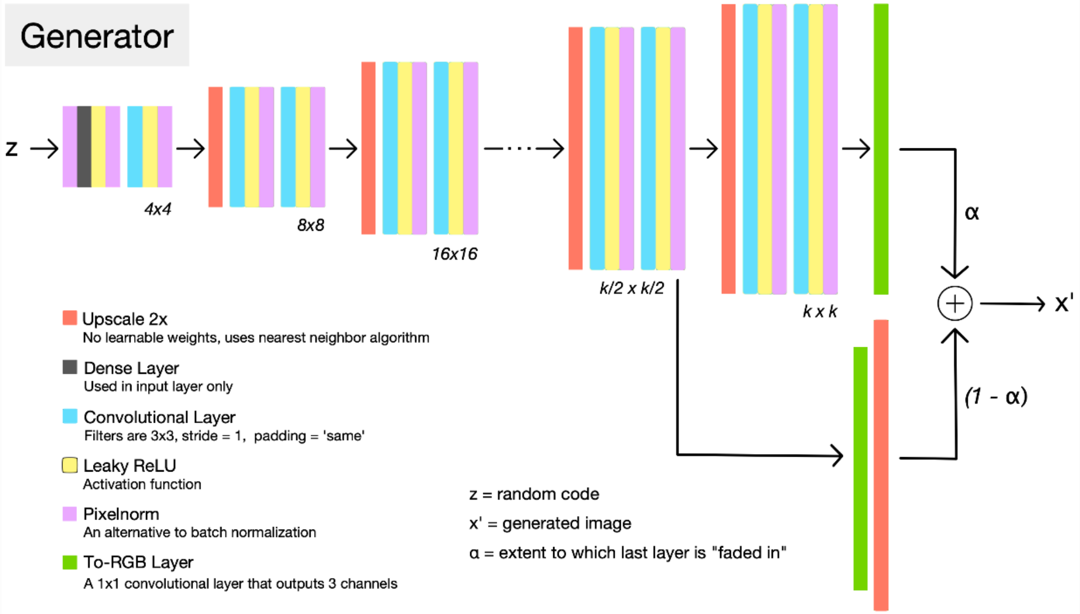
Последовательное добавление слоев большего размера после полного обучения предыдущих слоев





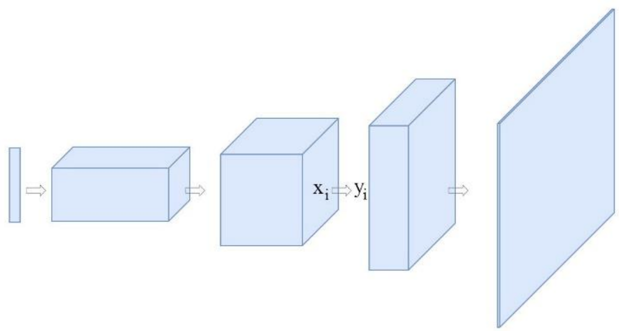
Архитектура и обучение ProGAN

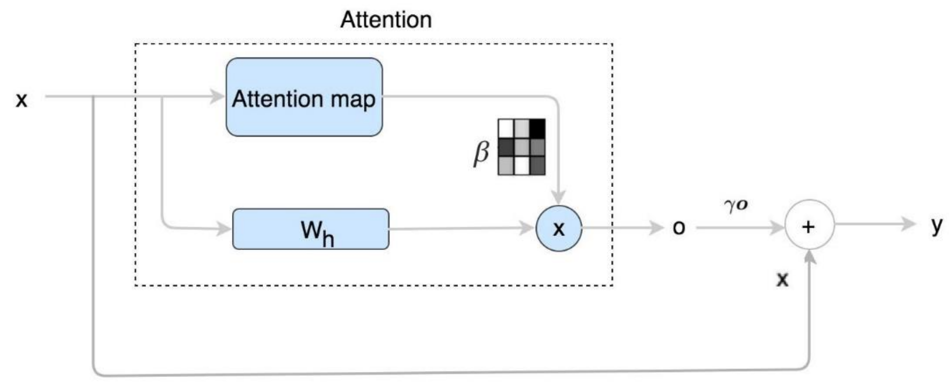
- Новые слои добавляются одновременно в генератор и дискриминатор
- Чтобы смягчить добавление, сигнал с нового слоя идет с постепенно увеличивающейся со временем константой α
- Вместо батчевой нормализации пиксельная нормализация

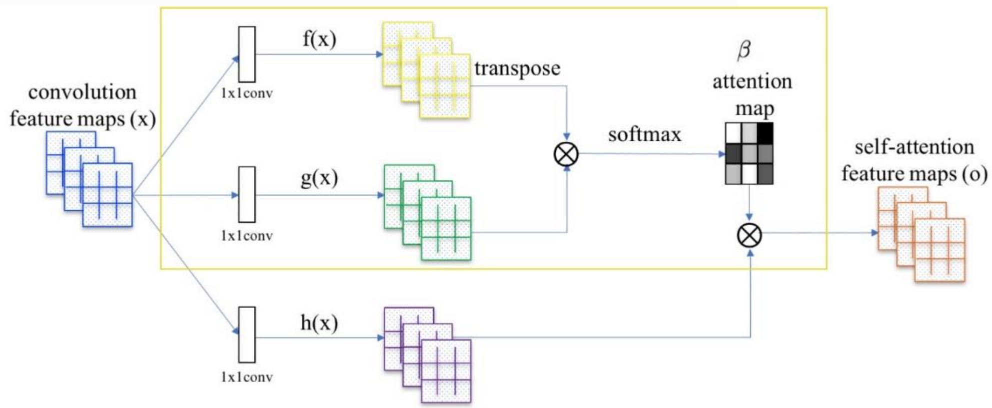


SAGAN

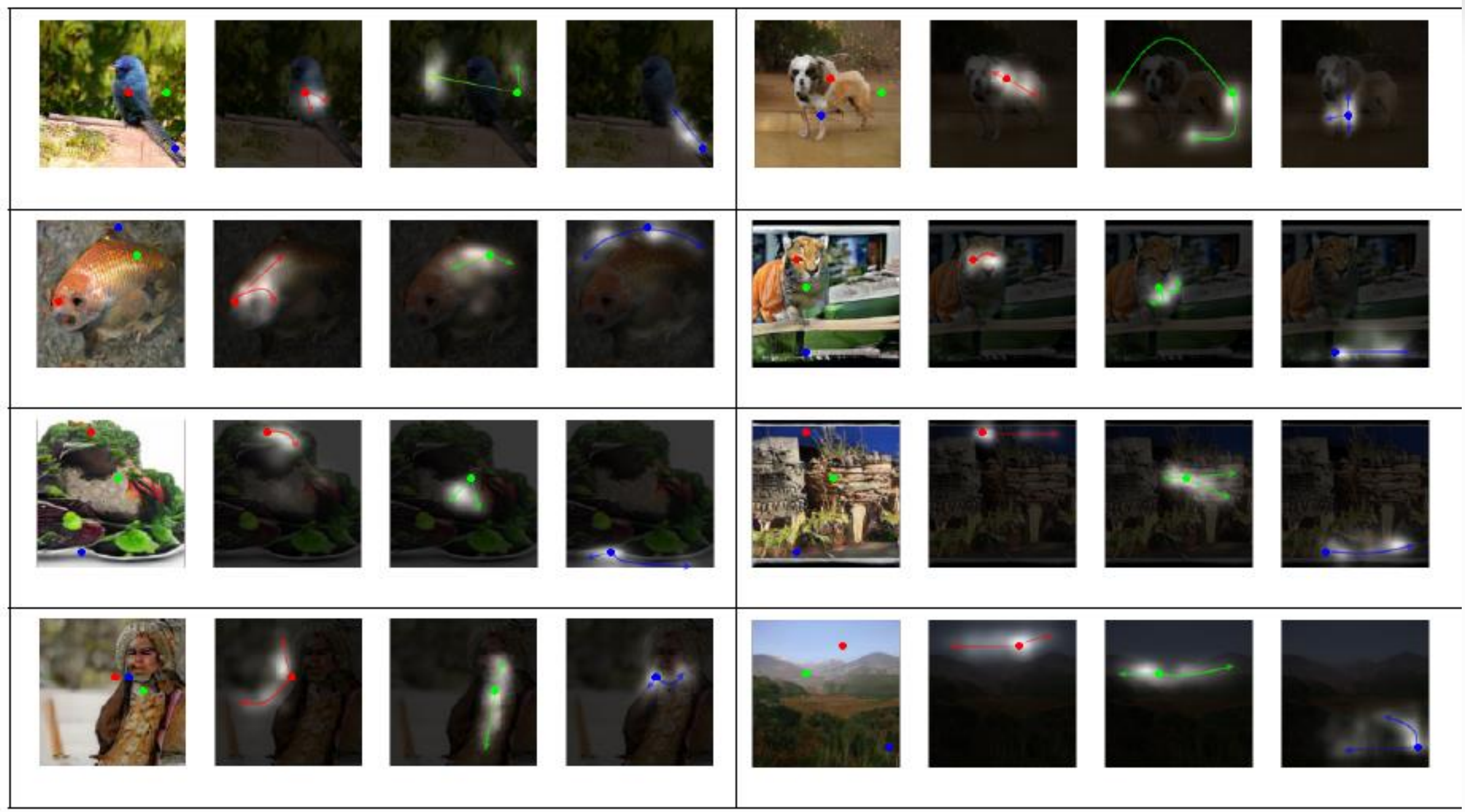
Идея: добавим self-attention к каждому сверточному слою генератора







Работа SAGAN



Распутывание признаков

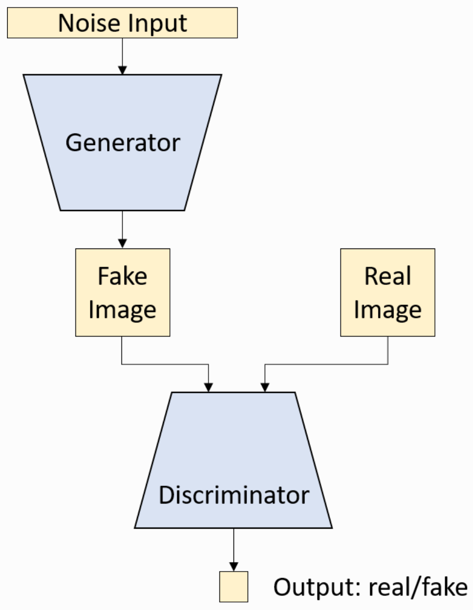
Задача распутывания (disentanglement) – наделение признаков семантическим смыслом.

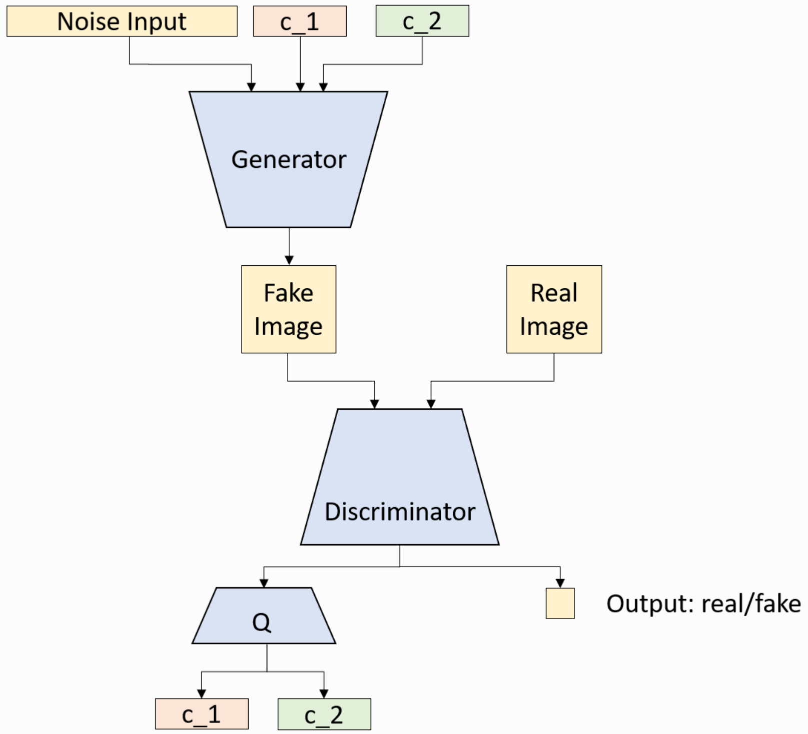
Это можно делать:

- за счет манипуляций с обучающими данными
- за счет скрытого представления
- за счет самообучения

InfoGAN

Добавим сеть Q, которая будет предсказывать интересующие нас параметры



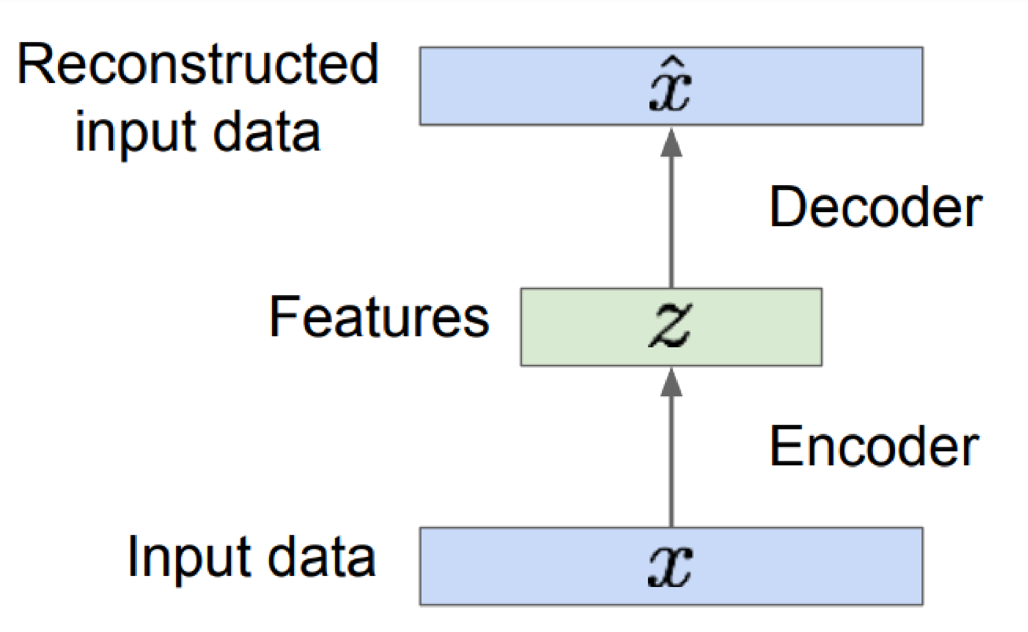


План лекции

- Генерация и качество генерации
- GANы и их проблемы
- Задача оптимальной транспортировки и функции потерь
- Интересные идеи в GANах
- Автоэнкодеры
- Интересные идеи в AE

Автокодировщики (напоминание)

Автокодировщик (autoencoder) – глубокая нейронная сеть, способная строить низкоразмерные представления данных за счет нелинейной трансформации.



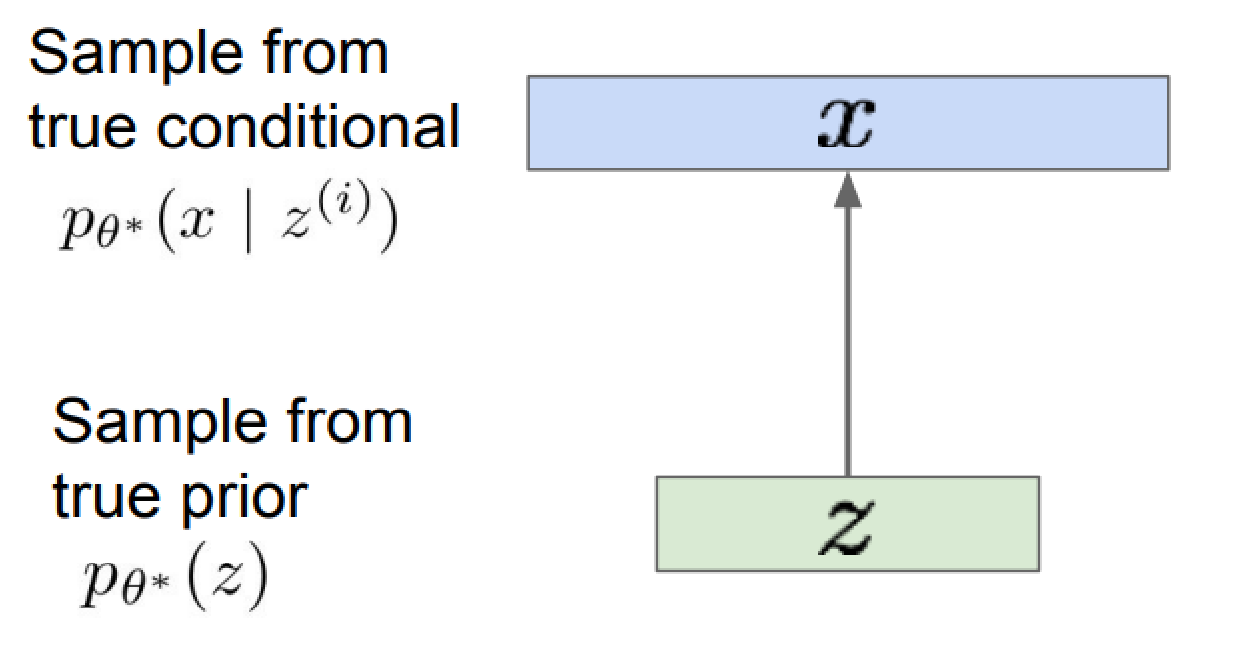
Основная идея

Вместо того, чтобы использовать некоторые предположения о том, как должна выглядеть структура вероятностной модели, мы определяем неразрешимую функцию плотности с некоторой скрытой переменной z:

$$p_\theta(x) = \int p_\theta(x|z)p_\theta(z)dz$$

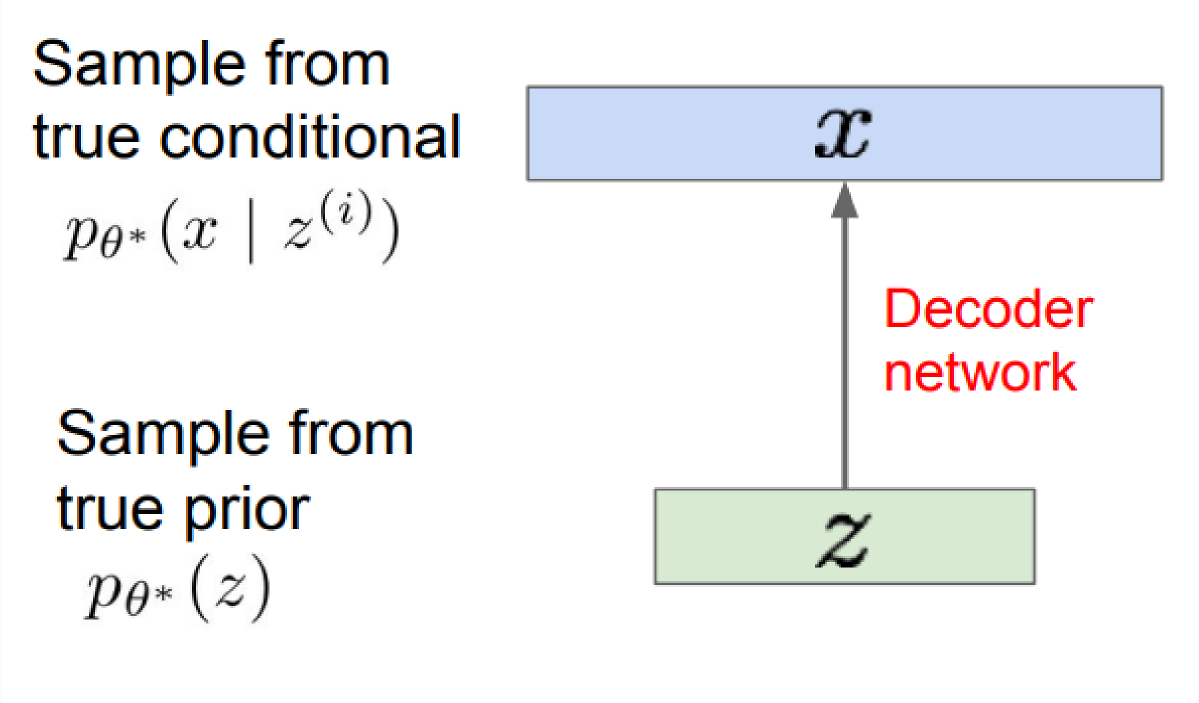
Скрытые переменные

Предположим, что данные обучения генерируются в зависимости от некоторого латентного *z*, устроенного достаточно простым образом



Представление $p(x|z)$

Условное $p(x \mid z)$ является сложным, восстановим его при помощи нейронной сети



Параметры обучения

$$p_\theta(x) = \int p_\theta(x|z)p_\theta(z)dz$$

$$\theta^* = \arg\min_\theta -\log \int p_\theta(x|z)p_\theta(z)dz$$

В чём состоит проблема?

Параметры обучения

$$p_\theta(x) = \int p_\theta(x|z)p_\theta(z)dz$$

$$\theta^* = \arg\min_\theta - \log \int p_\theta(x|z)p_\theta(z)dz$$

$\int p_\theta(x|z)p_\theta(z)dz$ трудноразрешим!

Параметры обучения

$$p_\theta(x) = \int p_\theta(x|z) p_\theta(z) dz$$

$$\theta^* = \arg\max_\theta \log \int p_\theta(x|z) p_\theta(z) dz$$

$\int p_\theta(x|z) p_\theta(z) dz$ трудноразрешим!

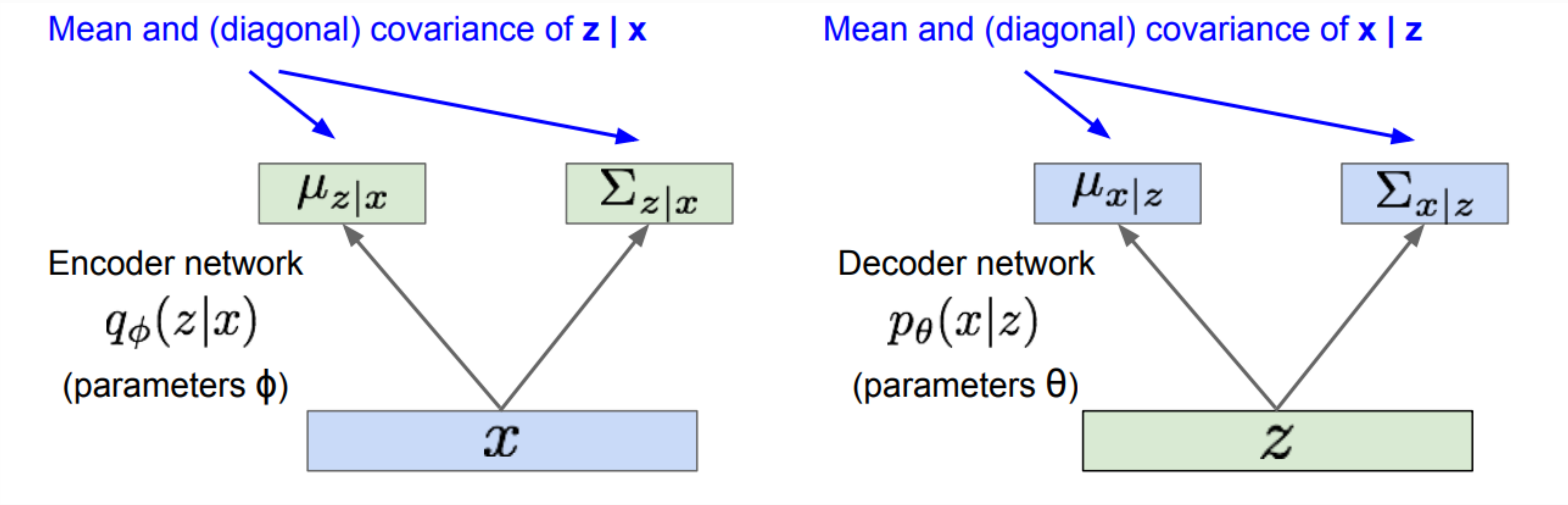
Вероятность апостериорных данных также нельзя выразить!

Сеть кодировщика

Идея: добавить сеть кодировщика $q_\phi(z|x)$, апроксимирующую $p_\phi(z|x)$

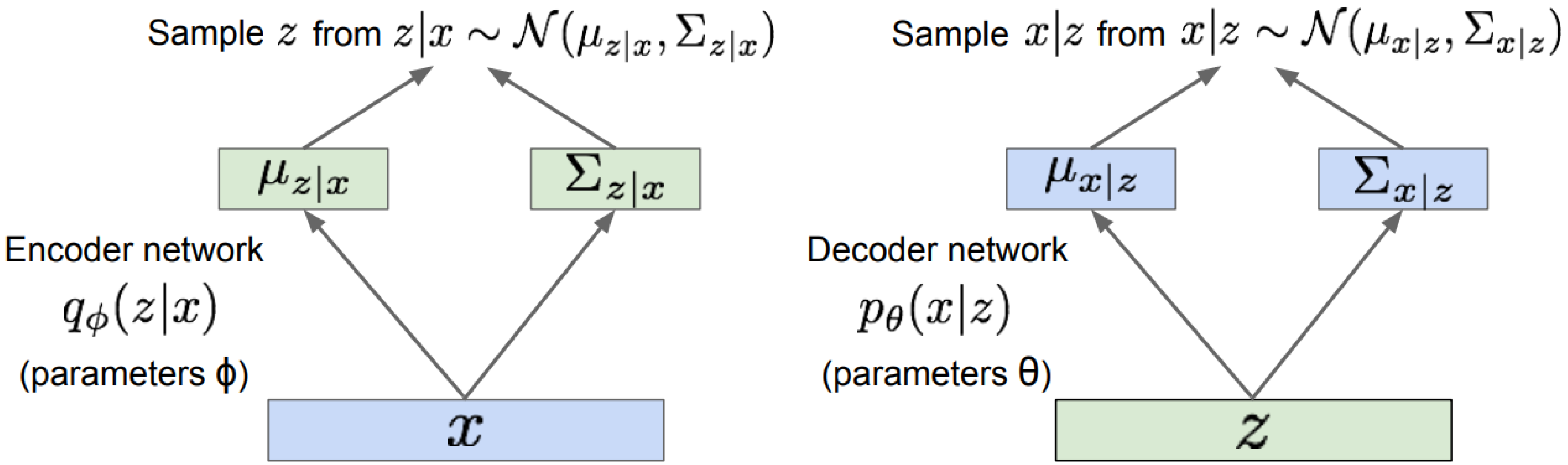
Кодировщик и декодировщик

Кодировщик и декодировщик являются вероятностными, оба предполагают гиперпараметры распределения (скажем, по Гауссу)



Сэмплирование с кодировщиком и декодировщиком

Мы можем выбрать z с помощью кодировщика и $x \mid z$ с помощью декодировщика.



Возвращаясь к правдоподобию

$$\theta^* = \arg\max_\theta \log p_\theta(x)$$

$$\log p_\theta(x^{(i)}) = \mathrm{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta\left(x^{(i)}\right)\right] =$$

$$= \mathrm{E}_z\left[\log \frac{p_\theta\left(x^{(i)}|z\right)p_\theta(z)}{p_\theta\left(z|x^{(i)}\right)}\right] =$$

$$= \mathrm{E}_z\left[\log \frac{p_\theta\left(x^{(i)}|z\right)p_\theta(z)}{p_\theta\left(z|x^{(i)}\right)} \frac{q_\phi\left(z|x^{(i)}\right)}{q_\phi\left(z|x^{(i)}\right)}\right] =$$

$$= \mathrm{E}_z\left[\log p_\theta\left(x^{(i)}|z\right)\right] - \mathrm{E}_z\left[\log \frac{q_\phi\left(z|x^{(i)}\right)}{p_\theta(z)}\right] + \mathrm{E}_z\left[\log \frac{q_\phi\left(z|x^{(i)}\right)}{p_\theta\left(z|x^{(i)}\right)}\right] =$$

$$= \mathrm{E}_z\left[\log p_\theta\left(x^{(i)}|z\right)\right] - D_{KL}\left(q_\phi\left(z|x^{(i)}\right)||p_\theta(z)\right)$$
$$+ D_{KL}\left(q_\phi\left(z|x^{(i)}\right)||p_\theta\left(z|x^{(i)}\right)\right)$$

Нижние границы

$$\mathrm{E}_z\left[\log p_\theta\left(x^{(i)}|z\right)\right] - D_{KL}\left(q_\phi\left(z|x^{(i)}\right)||p_\theta(z)\right) + D_{KL}\left(q_\phi\left(z|x^{(i)}\right)||p_\theta\left(z|x^{(i)}\right)\right)$$

Decoder network gives $p_\theta(x|z)$, can compute estimate of this term through sampling.

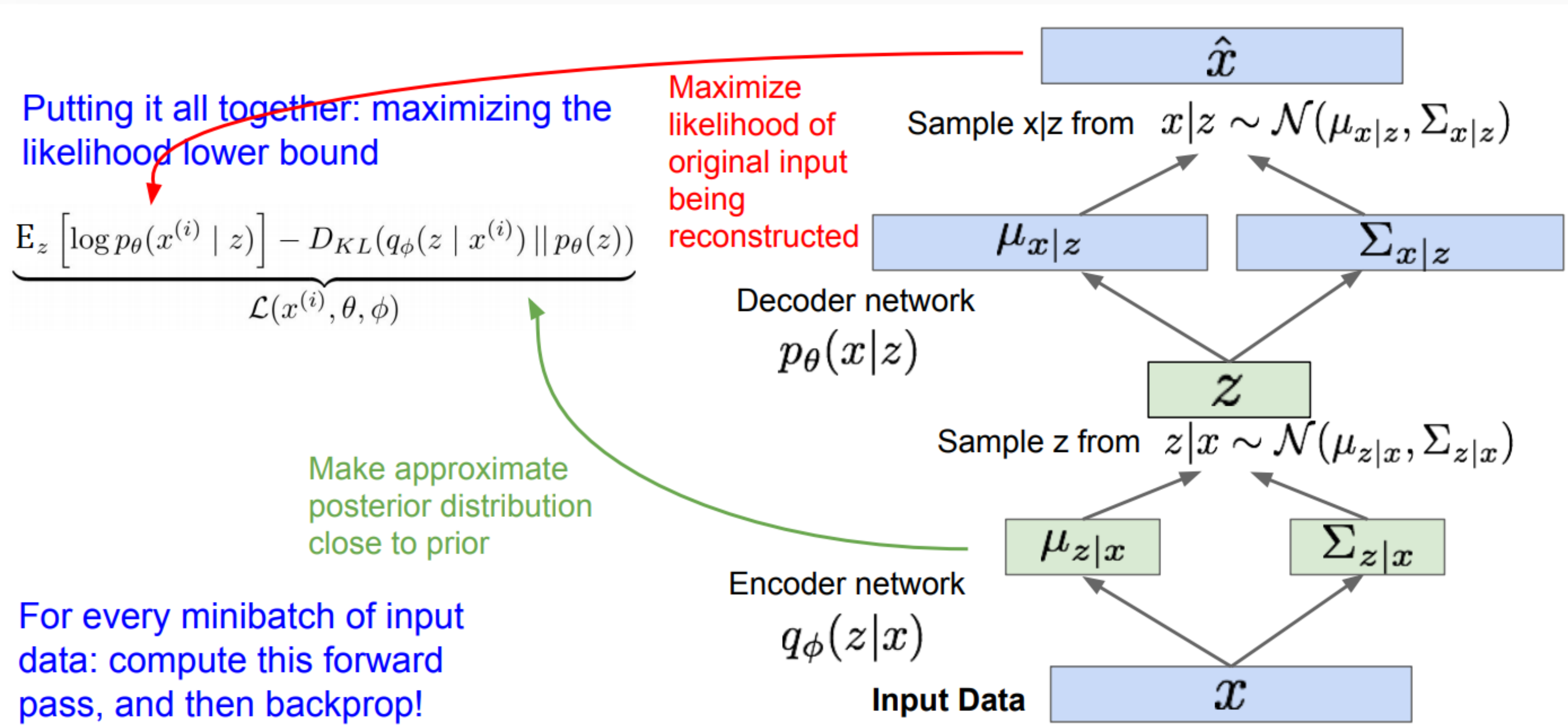
This KL term (between Gaussians for encoder and z prior) has nice closed-form solution!

$p_\theta(z|x)$ intractable (saw earlier), can't compute this KL term :(But we know KL divergence always >= 0.

$$\boxed{\underbrace{\mathrm{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z))}_{\mathcal{L}(x^{(i)},\theta,\phi)}} + \underbrace{D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z \mid x^{(i)}))}_{\boxed{\geq 0}}$$

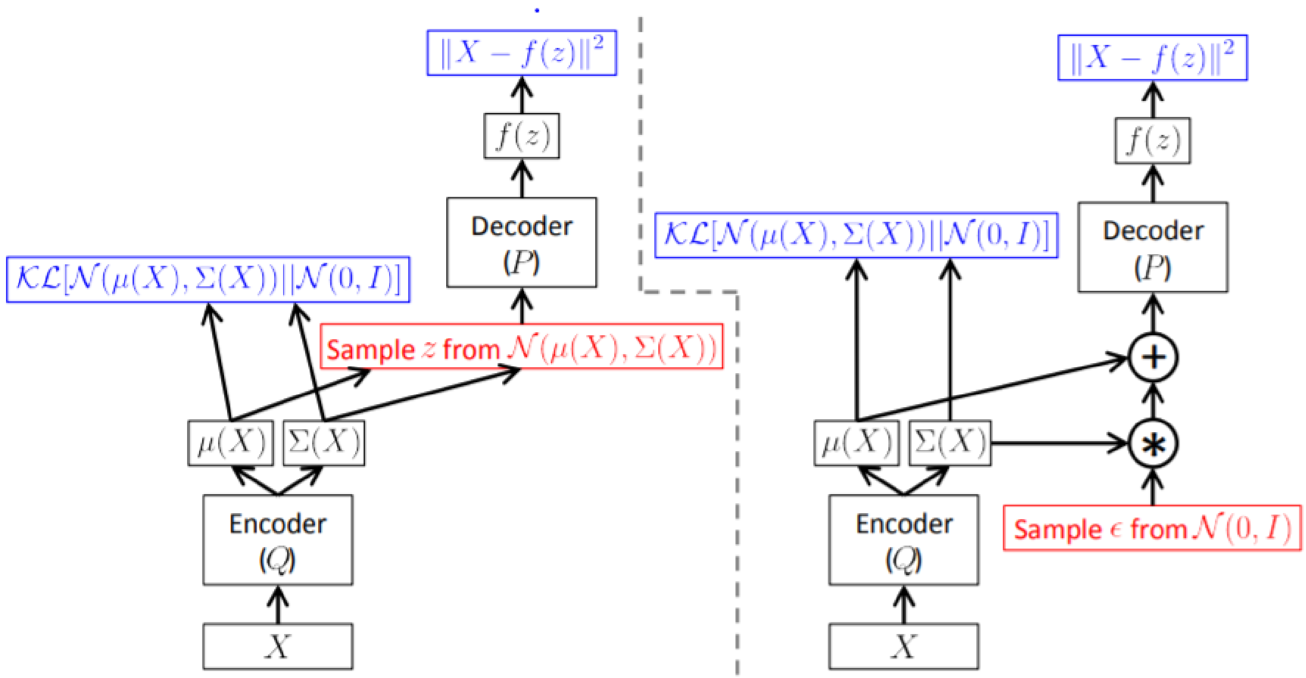
Tractable lower bound which we can take gradient of and optimize! ($p_\theta(x|z)$ differentiable, KL term differentiable)

Обучение VAE



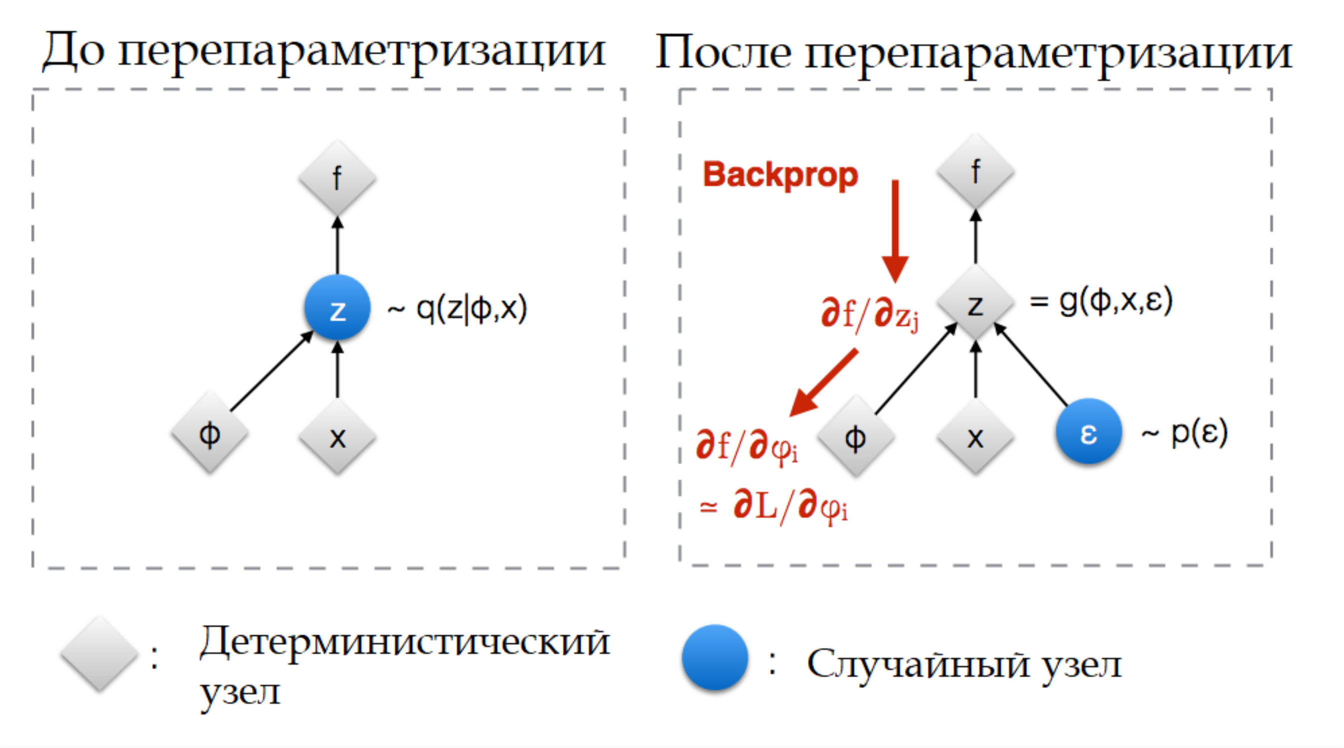
Репараметризационный трюк

Заменим переменные, добавив случайный коэффициент



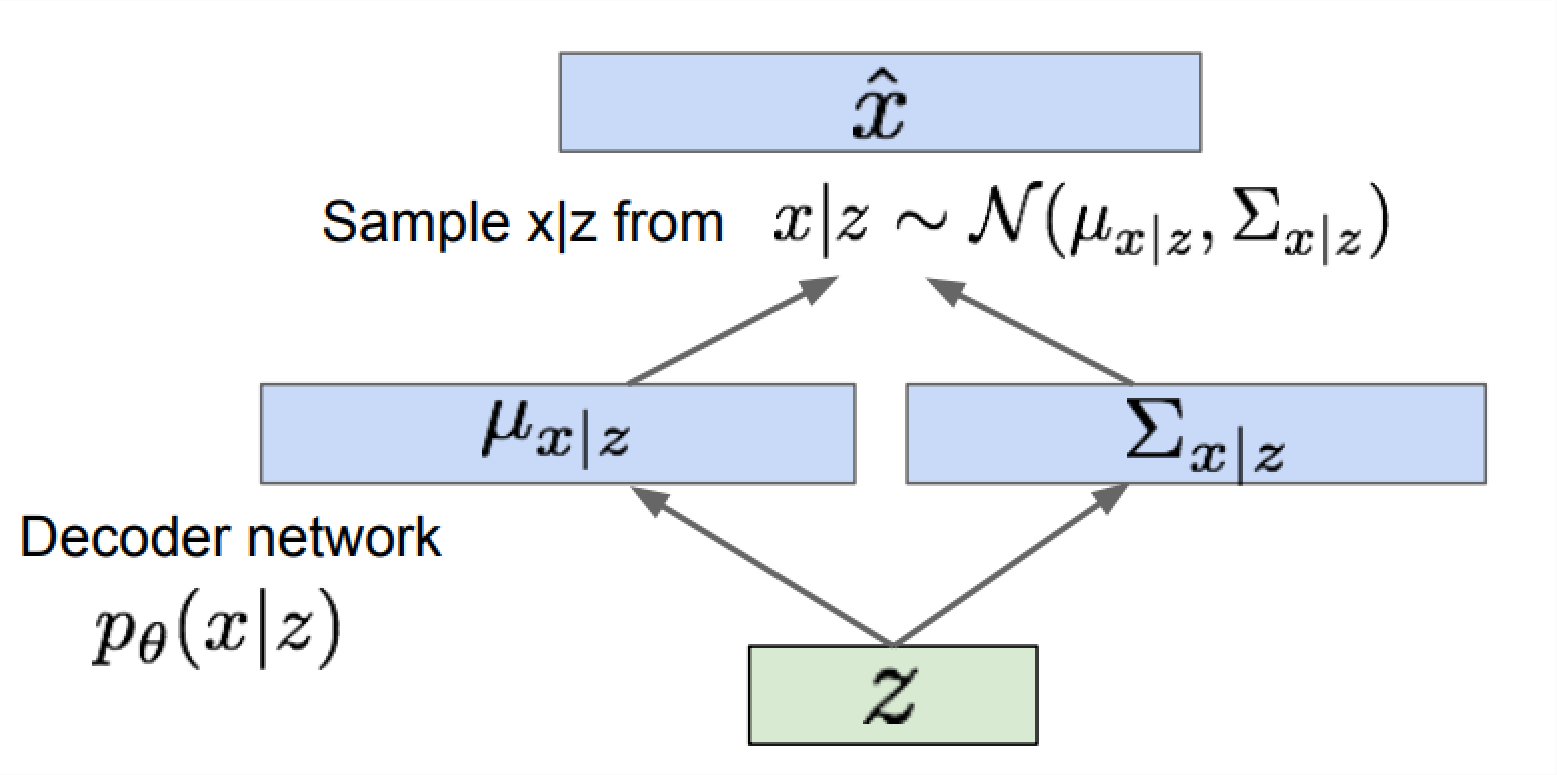
Репараметризационный трюк

Тем самым можно будет пустить через узел градиентный спуск



Создание данных с помощью VAE

Простой пример с декодером



Анализ VAE

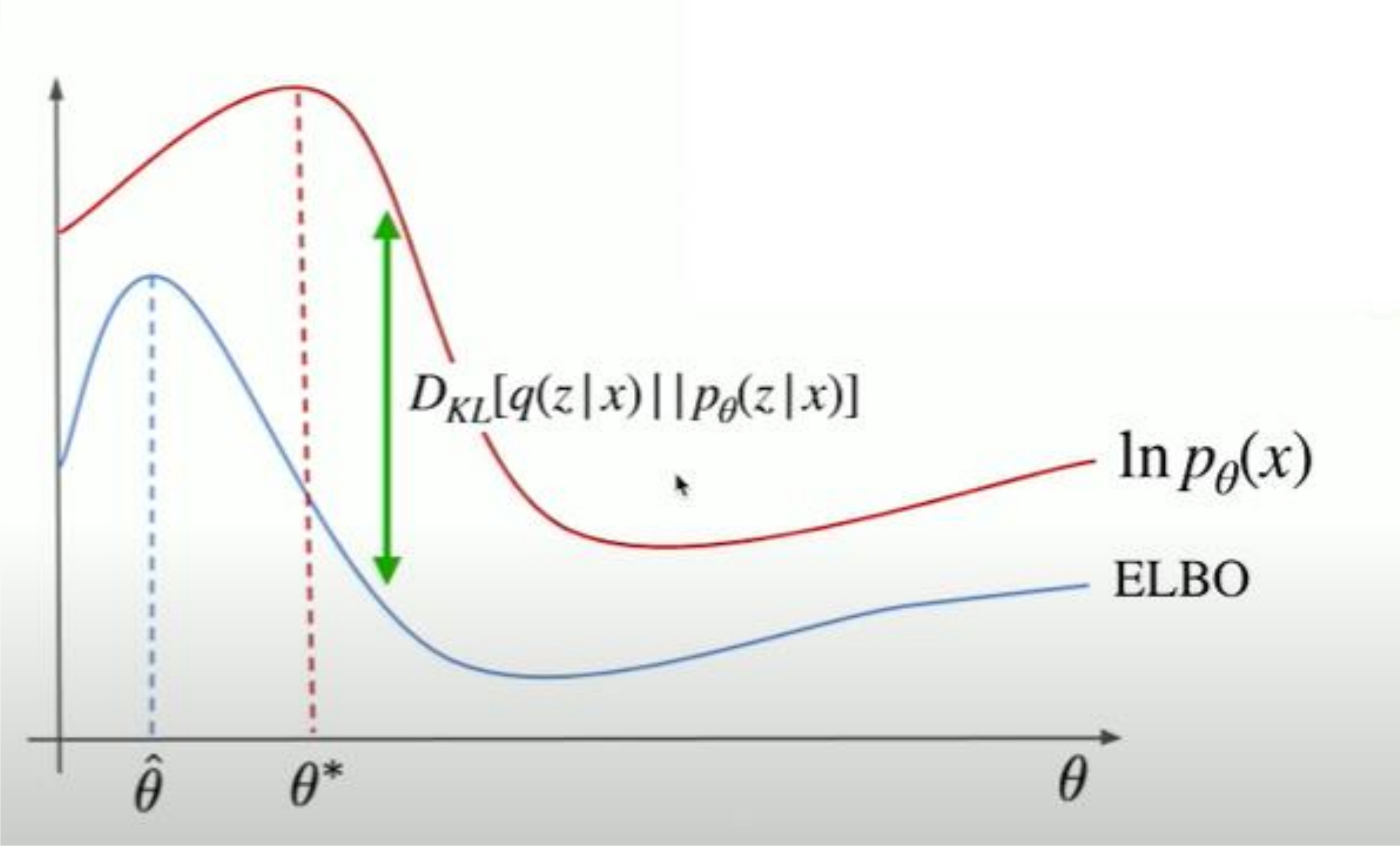
Преимущества:

- Принципиальный подход к генеративным моделям
- Позволяет сделать вывод $q\,(z \mid x)$, может быть полезным представлением функции для других задач

Недостатки:

- Максимизация нижней границы вероятности - не такая хорошая оценка, как PixelRNN / PixelCNN.
- Образцы более размытые и более низкого качества по сравнению с современными (GAN)

Некоторые проблемы

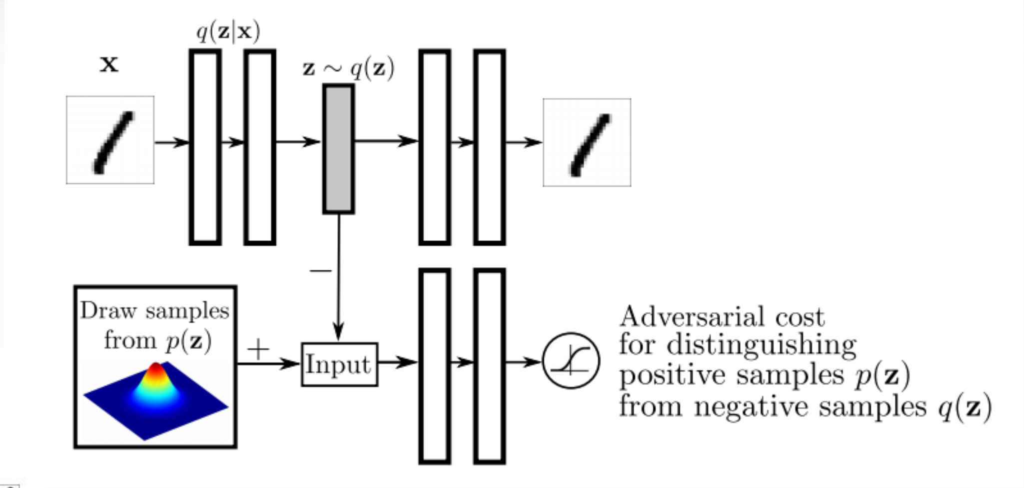


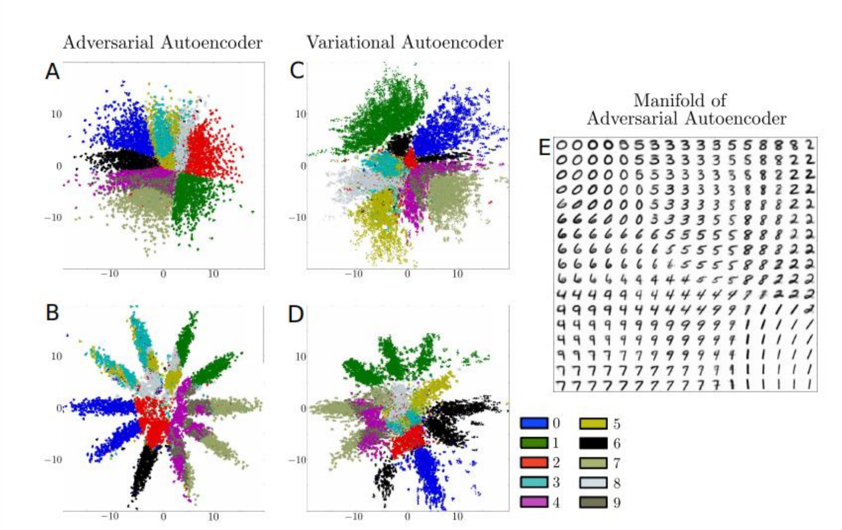
План лекции

- Генерация и качество генерации
- GANы и их проблемы
- Задача оптимальной транспортировки и функции потерь
- Интересные идеи в GANах
- Автоэнкодеры
- Интересные идеи в AE

Adversarial autoencoder

Будем добавлять лосс за то, что распределение скрытой переменной не похоже на то, которое мы закладываем



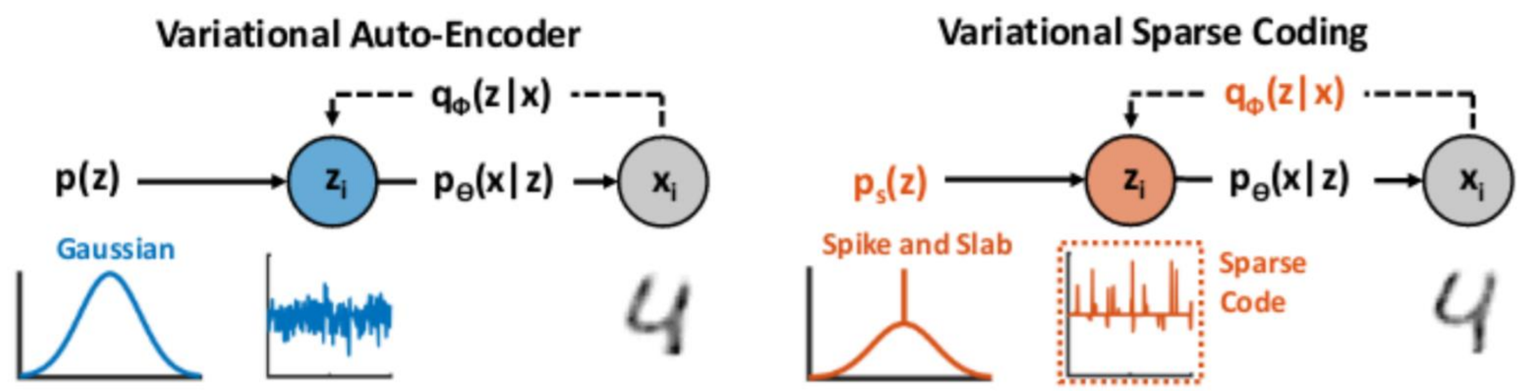


Spike and Slab prior

Если взять вот такую функцию априорной вероятности,

$$p_S(z) = \prod_{j=1}^{J} \left(\alpha \mathcal{N}(z_j; 0, 1) + (1-\alpha)\delta(z_j)\right)$$

то апостериорное распределение будет смесью



Wasserstein autoencoder

Вместо похожести с исходным распределением по KL, будем сравнивать то, насколько сильно различаются их представления