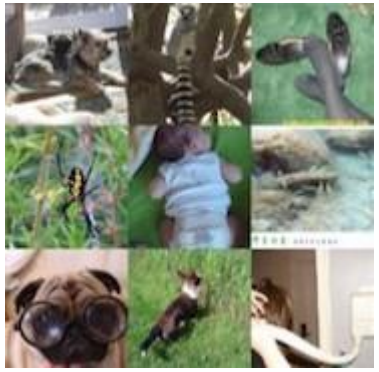
Введение в генеративные модели: VAE, PixelCNN/PixelRNN, GAN

Градобоев Дмитрий, БПМИ171

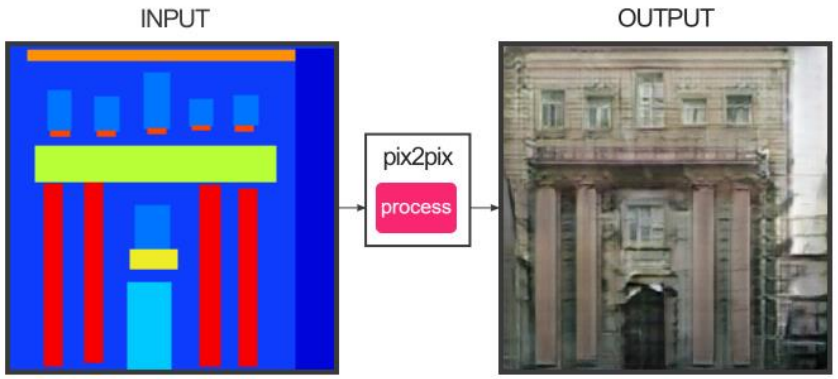
Генеративные модели

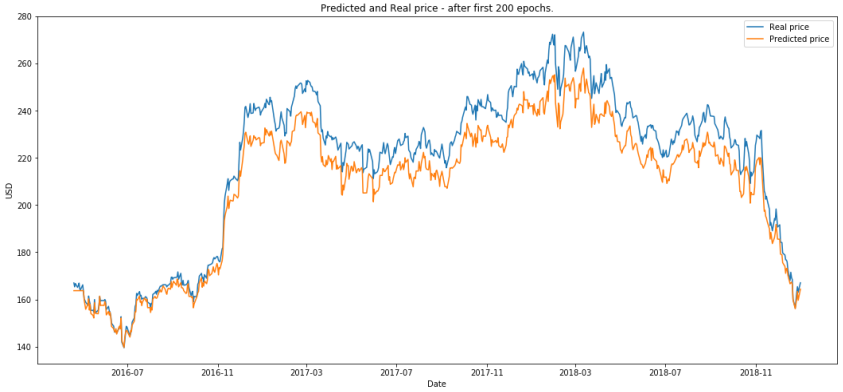


Обучающая выборка $\sim p_{data}(x)$

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

Цель: научиться обучать $p_{model}(x)$ похожим на $p_{data}(x)$

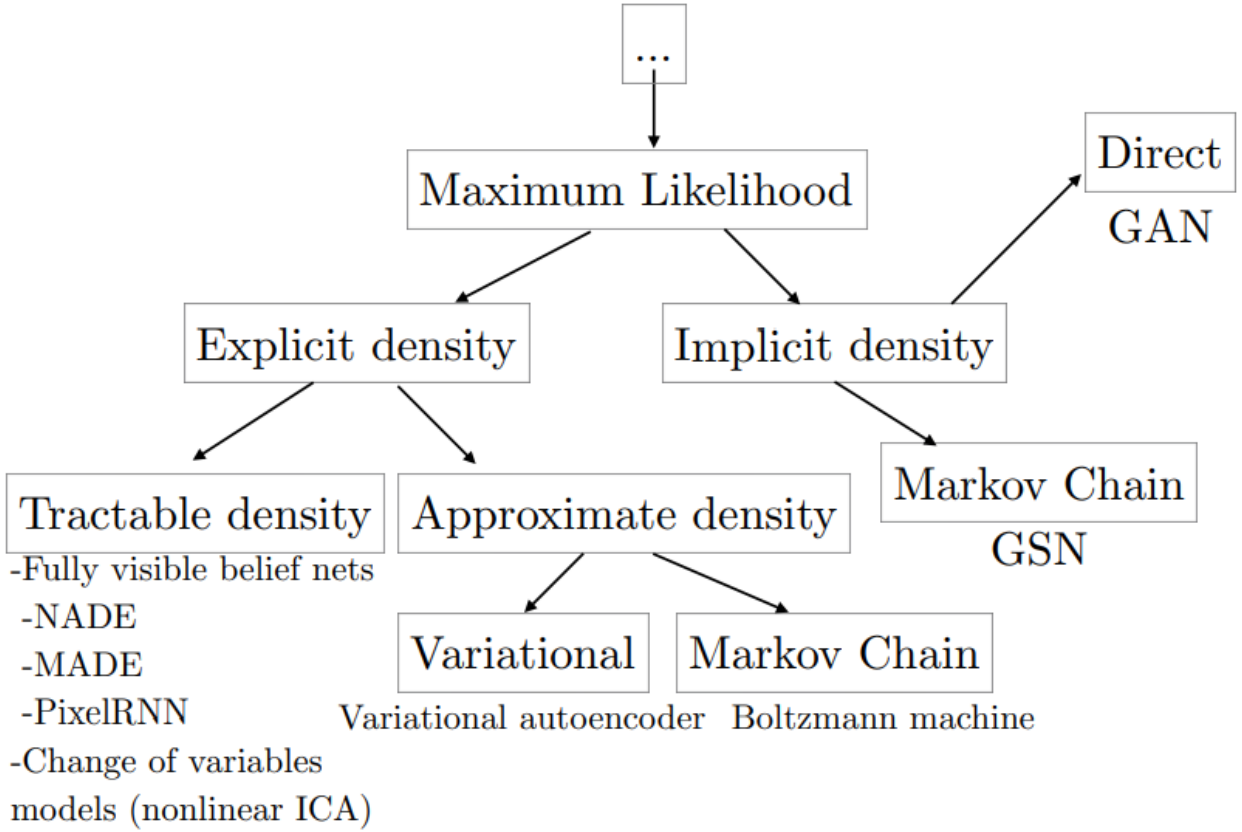




Зачем?

- Сгенерированные данные
- Симуляция временных рядов
- Выделение основных признаков

Подход к генерации



- Явно выраженная плотность
  - Простая плотность
    - PixelRNN/PixelCNN
  - Приближение плотности
    - VAE
- Неявная работа с плотностью
  - Прямое сэмплирование
    - GAN

Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.

PixelRNN/PixelCNN

x – картинка размера $n \times n$

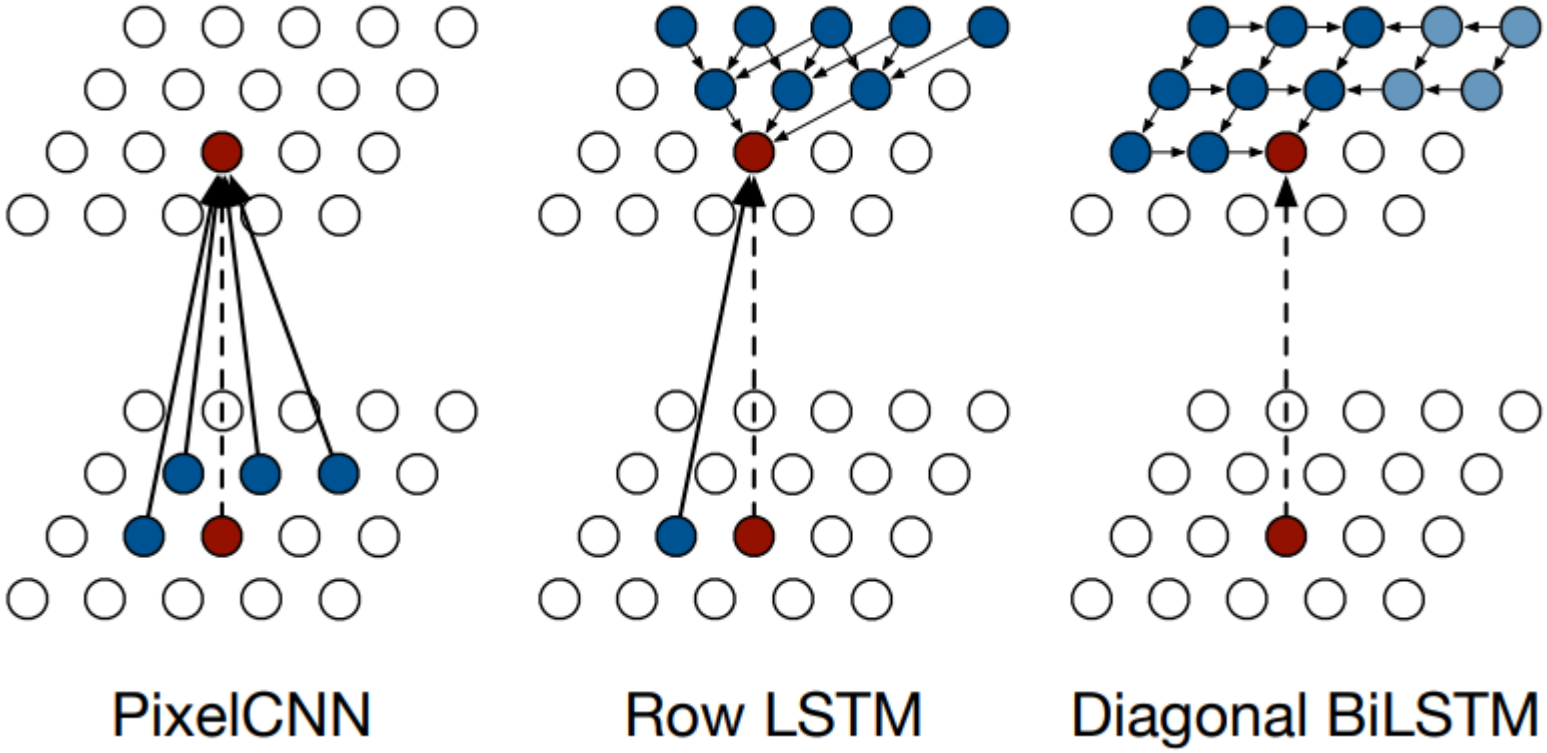
$$p(x) = \prod_{i=1}^{n^2} p(x_i | x_1, ..., x_{i-1})$$

Упорядочиваем пиксели – $x_1, \ldots, x_{n^2}$

Максимизируем правдоподобие.

Для подсчёта отдельных вероятностей используем нейросети.

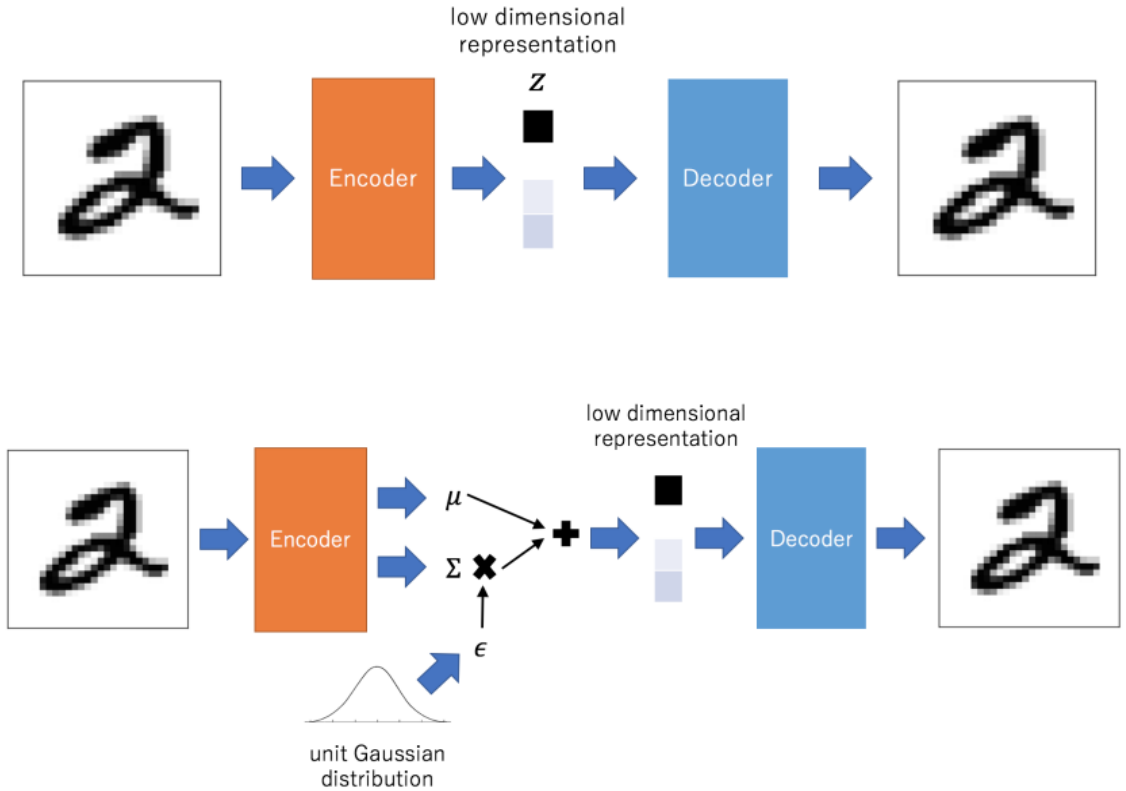
PixelRNN/PixelCNN



PixelRNN/PixelCNN

- Плюсы:
  - Явный подсчет правдоподобия
  - Хорошие результаты метрики
  - Высокое качество генерации
- Минусы:
  - Низкая скорость

VAE



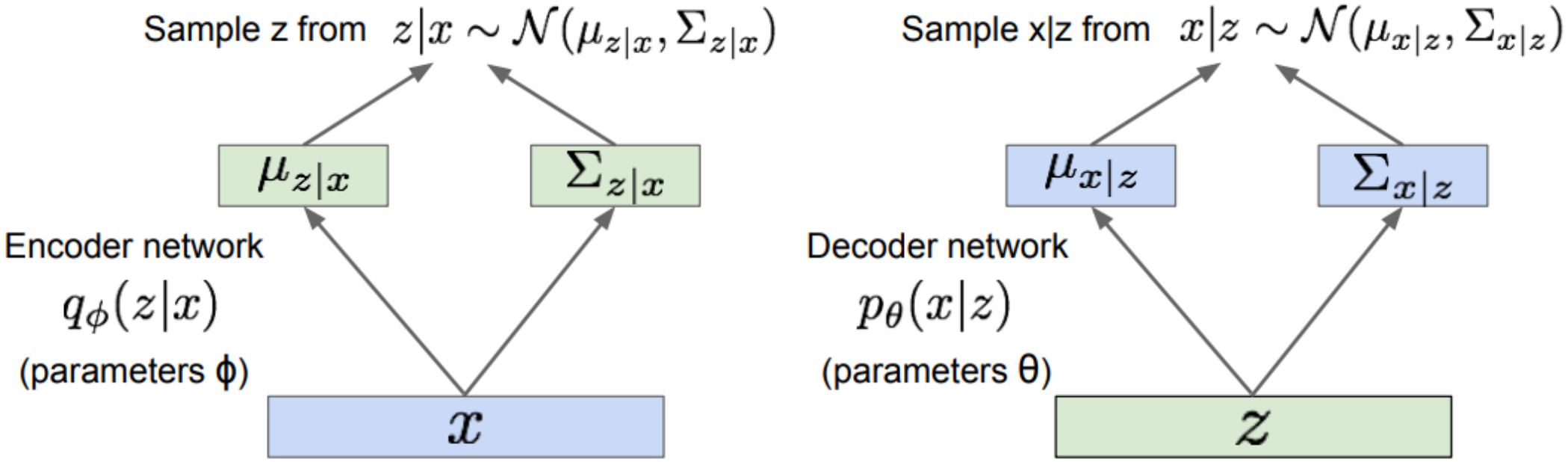
$$p_\theta(x) = \int p_\theta(z) p_\theta(x|z) dz$$

$$z = \mu + \sigma \cdot \epsilon, \text{ где } \epsilon \sim N(0, 1)$$

$$p_\theta(z|x) = \frac{p_\theta(x|z) p_\theta(z)}{p_\theta(x)}$$

Найдём оценку на апостериорное распределение

VAE


Sample z from $z|x \sim \mathcal{N}(\mu_{z|x}, \Sigma_{z|x})$
$\mu_{z|x}$
$\Sigma_{z|x}$
Encoder network
$q_\phi(z|x)$
(parameters ϕ)
x
Sample x|z from $x|z \sim \mathcal{N}(\mu_{x|z}, \Sigma_{x|z})$
$\mu_{x|z}$
$\Sigma_{x|z}$
Decoder network
$p_\theta(x|z)$
(parameters θ)
z

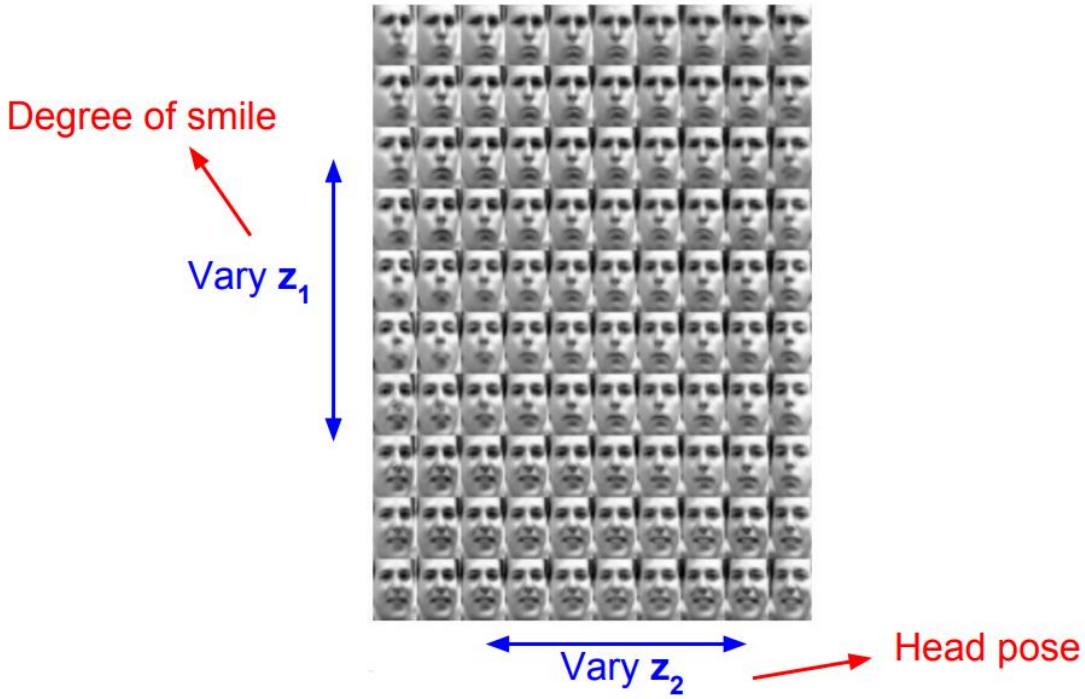
VAE

$$log\, p_\theta(x^{(i)}) = E_z\Big[log\, p_\theta(x^{(i)}|z)\Big] - D_{KL}(q_\phi(z|x^{(i)})||p_\theta(z)) + \underbrace{D_{KL}(q_\phi(z|x^{(i)})||p_\theta(z|x^{(i)}))}_{\geq 0}$$

$$\mathcal{L}(x^{(i)}, \theta, \phi) = E_z\Big[log\ p_\theta(x^{(i)}|z)\Big] - D_{KL}(q_\phi(z|x^{(i)})||p_\theta(z))$$

VAE

- Плюсы:
  - “Честная и принципиальная” генерация
  - Получаем признаки, которые можем использовать
- Минусы:
  - Мы используем нижнюю оценку, а не точный подсчёт
  - Сэмплы размыты, качество генерации хуже, чем у SOTA моделей



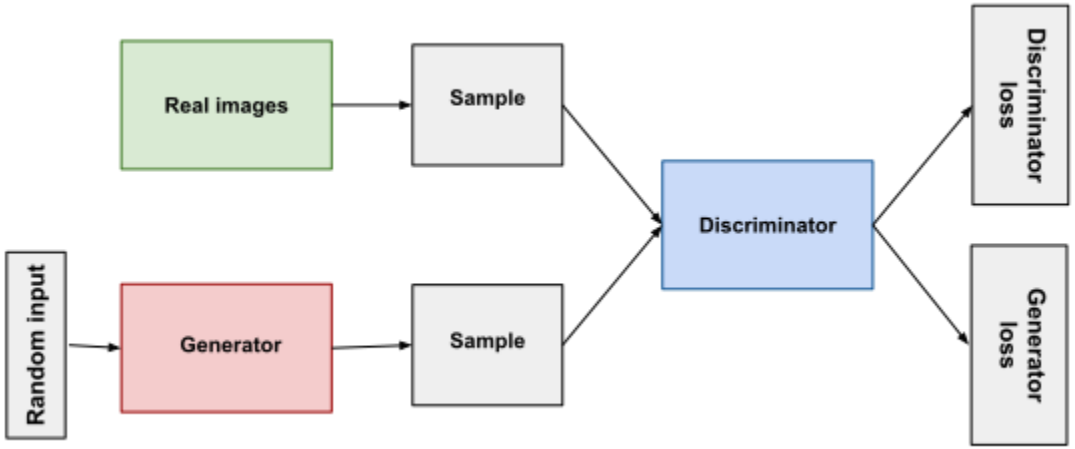
GAN

Не хотим явно смотреть на распределение, но хотим его смоделировать.

Идея: Игра с 2 игроками.

Генератор – генерирует сэмплы, пытаясь “обмануть” дискриминатор.

Дискриминатор – отличает настоящие сэмплы от сгенерированных.



GAN

Целевая функция:

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

Дискриминатор будет сводить D(x) к 1, D(G(z)) к 0.

Генератор – D(G(z)) к 1.

GAN

for number of training iterations **do**

for k steps **do**

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
- Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

end for

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

GAN

- Плюсы:
  - Хорошее качество генерации
- Минусы:
  - Не стабильны
  - Не дают дополнительных данных

Вопросы

1. В чём основная идея PixelRNN и PixelCNN, и какая из-за этого возникает проблема в этих моделях?
2. Выпишите формулу логарифмического правдоподобия для VAE и поясните её слагаемые.
3. Как происходит алгоритм обучения в GAN?

Источники

Stanford, cs231n, Lecture 13 | Generative Models. 2017 https://www.youtube.com/watch?v=5WoItGTWV54

Pixel Recurrent Neural Networks. 2016. https://arxiv.org/abs/1601.06759

Auto-Encoding Variational Bayes. 2014 https://arxiv.org/abs/1312.6114

Tutorial on Variational Autoencoders. 2016 https://arxiv.org/abs/1606.05908

Generative Adversarial Nets. 2014 https://arxiv.org/abs/1406.2661