Generative Adversarial Networks

Рубачёв Иван

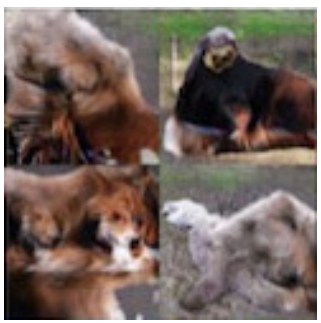
НИУ ВШЭ

Содержание

Порождающие модели

Порождающие модели

Обучающая выборка $\sim \mathrm{p}_{\mathrm{data}}(\mathrm{x})$



Порожденные примеры $\sim \mathrm{p}_{\mathrm{model}}(\mathrm{x})$

Хотим, чтобы $\mathrm{p}_{\mathrm{model}}(\mathrm{x})$ была похожа на $\mathrm{p}_{\mathrm{data}}(\mathrm{x})$

Для чего нужны порождающие модели?

- Генерировать реалистичные фотографии, super resolution, image impainting, ...

Для чего нужны порождающие модели?

- Генерировать реалистичные фотографии, super resolution, image impainting, ...
- В результате обучения порождающих моделей можно получить полезные высокоуровневые признаки

Для чего нужны порождающие модели?

- Генерировать реалистичные фотографии, super resolution, image impainting, ...
- В результате обучения порождающих моделей можно получить полезные высокоуровневые признаки
- Задачи, в которых более одного правильного ответа (например сгенерировать следующий кадр видео)

Различные варианты порождающих моделей

- Явно выраженная плотность

Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017

Различные варианты порождающих моделей

- Явно выраженная плотность
  - Простая факторизующаяся плотность
    - PixelRNN
    - PixelCNN

Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017

Различные варианты порождающих моделей

- Явно выраженная плотность
  - Простая факторизующаяся плотность
    - PixelRNN
    - PixelCNN
  - Приближения к плотности
    - Вариационный автокодировщик (VAE)

Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017

Различные варианты порождающих моделей

- Явно выраженная плотность
  - Простая факторизующаяся плотность
    - PixelRNN
    - PixelCNN
  - Приближения к плотности
    - Вариационный автокодировщик (VAE)
- Неявно выраженная плотность
  - Модель напрямую семплирует объекты
    - Порождающие состязательные сети (GAN)

Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017

Простая факторизующаяся плотность

Можно разложить $\mathrm{p}(\mathrm{x})$ в произведение:

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

Хотим максимизировать правдоподобие

Простая факторизующаяся плотность

Можно разложить $p(x)$ в произведение:

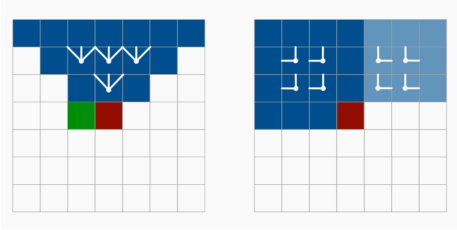
$$p(x) = \prod_{i=1}^{n} p(x_i | x_1, \ldots, x_{i-1})$$

Моделируем нейросетью

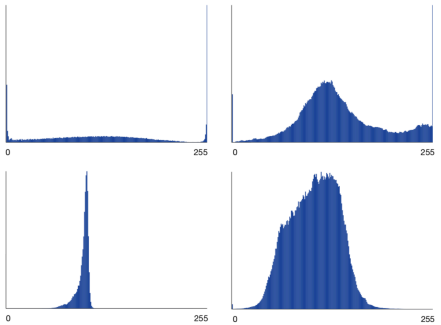
Хотим максимизировать правдоподобие

PixelRNN

- Последовательно генерируем пиксели
- Максимизируем правдоподобие на обучающей выборке
- Обучение и построение изображения работают медленно



Последовательное построение изображения

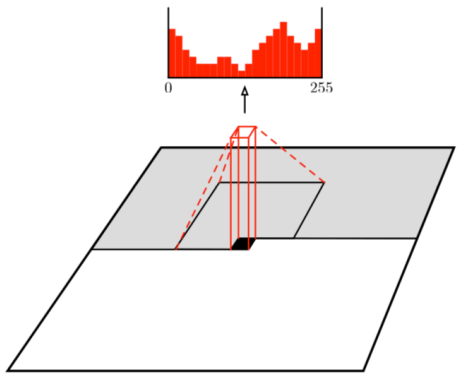


Пример выхода для различных пикселей

van den Oord, Pixel Recurrent Neural Networks, 2016

PixelCNN

- Вместо RNN используем CNN
- Максимизируем правдоподобие на обучающей выборке
- Обучение быстрее чем в PixelRNN
- Построение нового изображения по прежнему медленное

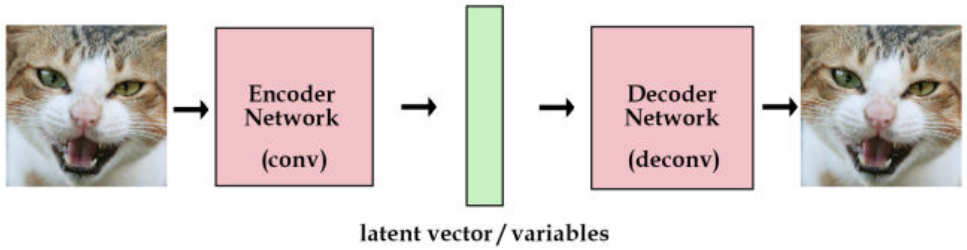


Последовательное построение изображения

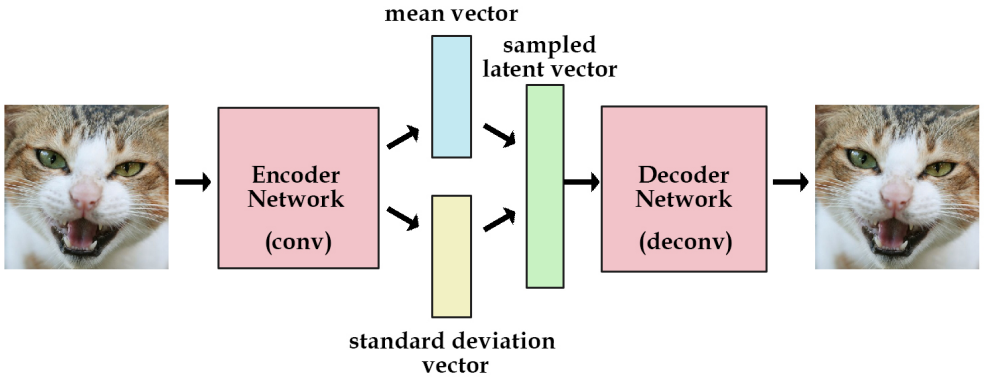
van den Oord et. al., Conditional Image Generation with PixelCNN Decoders, 2016

Variational autoencoder

- Автокодировщик



- Вариационный автокодировщик



Variational autoencoder

$$\mathrm{p}_\theta(\mathrm{x}) = \int \mathrm{p}_\theta(\mathrm{z})\mathrm{p}_\theta(\mathrm{x}|\mathrm{z})\mathrm{dz}$$

- Не можем оптимизировать напрямую
- Можно найти нижнюю границу и оптимизировать её

$$\mathcal{L}(\theta, \phi, \mathrm{x}, \mathrm{z}) = \mathbb{E}_{\mathrm{q}_\phi(\mathrm{z}|\mathrm{x})}\left[\log \mathrm{p}_\theta(\mathrm{x}|\mathrm{z})\right] - \mathrm{D}_{\mathrm{KL}}\left(\mathrm{q}_\phi(\mathrm{z}|\mathrm{x})||\mathrm{p}(\mathrm{z})\right)$$

- Reparametrization trick: $\mathrm{z} = \mu + \sigma \odot \varepsilon, \quad$ where $\varepsilon \sim \mathcal{N}(0, 1)$

GAN

Generative Adversarial Networks

- GAN: моделируем семплирование объектов из распределения;

Generative Adversarial Networks

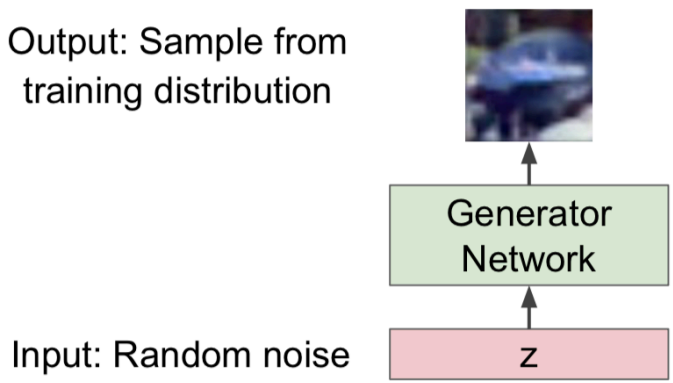
- GAN: моделируем семплирование объектов из распределения;
- Как моделировать сэмплирование из сложного распределения?

Generative Adversarial Networks

- GAN: моделируем семплирование объектов из распределения;
- Как моделировать сэмплирование из сложного распределения?
- Можно сэмплировать из простого распределения и выучить преобразование;

Generative Adversarial Networks

- GAN: моделируем семплирование объектов из распределения;
- Как моделировать сэмплирование из сложного распределения?
- Можно сэмплировать из простого распределения и выучить преобразование;
- Нейронная сеть может приблизить это преобразование;



Обучение GAN: игра с 2 игроками

- **Генератор**: «Обмануть» дискриминатор, порождая примеры похожие на настоящие
  Дискриминатор: Отличать примеры порожденные генератором от настоящих

Обучение GAN: игра с 2 игроками

- **Генератор**: «Обмануть» дискриминатор, порождая примеры похожие на настоящие
  Дискриминатор: Отличать примеры порожденные генератором от настоящих
- Целевая функция:

$$\min_{\theta_{\mathrm{g}}} \max_{\theta_{\mathrm{d}}} \left[\underbrace{\mathbb{E}_{\mathrm{x}\sim \mathrm{p}_{\mathrm{data}}} \log \mathrm{D}_{\theta_{\mathrm{d}}}(\mathrm{x})}_{\text{реальные данные}} + \underbrace{\mathbb{E}_{\mathrm{z}\sim \mathrm{p}(\mathrm{z})} \log(1 - \mathrm{D}_{\mathrm{theta}_{\mathrm{d}}}(\mathrm{G}_{\mathrm{theta}_{\mathrm{g}}}(\mathrm{z})))}_{\text{примеры генератора}} \right]$$

Обучение GAN: игра с 2 игроками

- **Генератор**: «Обмануть» дискриминатор, порождая примеры похожие на настоящие
  Дискриминатор: Отличать примеры порожденные генератором от настоящих
- Целевая функция:

$$\min_{\theta_{\mathrm{g}}} \max_{\theta_{\mathrm{d}}} \left[\underbrace{\mathbb{E}_{\mathrm{x} \sim \mathrm{p}_{\mathrm{data}}} \log \mathrm{D}_{\theta_{\mathrm{d}}}(\mathrm{x})}_{\text{реальные данные}} + \underbrace{\mathbb{E}_{\mathrm{z} \sim \mathrm{p}(\mathrm{z})} \log(1 - \mathrm{D}_{\mathrm{theta}_{\mathrm{d}}}(\mathrm{G}_{\mathrm{theta}_{\mathrm{g}}}(\mathrm{z})))}_{\text{примеры генератора}} \right]$$

- Дискриминатор: $\mathrm{D}(\mathrm{x}) \to 1$ и $\mathrm{D}(\mathrm{G}(\mathrm{x})) \to 0$.
  Генератор: $\mathrm{D}(\mathrm{G}(\mathrm{x})) \to 1$

Алгоритм обучения

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:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D_{\theta_d}(x^{(i)}) + \log(1 - D_{\theta_d}(G_{\theta_g}(z^{(i)}))) \right]$$

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 ascending its stochastic gradient (improved objective):

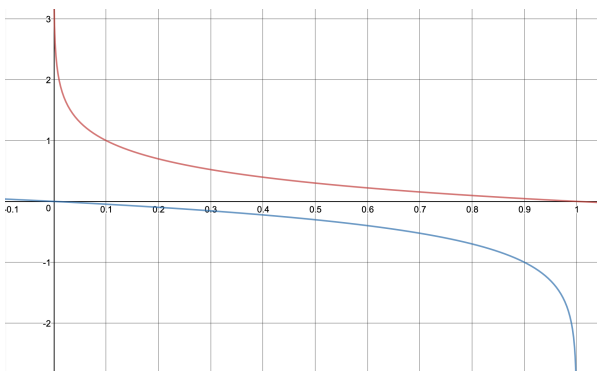
$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log(D_{\theta_d}(G_{\theta_g}(z^{(i)})))$$

end for

Проблемы с градиентом

- Если порожденные примеры плохие, то градиент будет маленьким, поэтому обычно максимизируют:

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



Графики $\log(1 - D(G(z)))$ и $-\log(D(G(z)))$

Сравнения и выводы

Сравнения

- PixelCNN и PixelRNN: можно вычислить $\mathrm{p}(\mathrm{x})$, хорошие примеры, но медленно;
- VAE: позволяют извлекать полезные признаки $\mathrm{q}(\mathrm{z}|\mathrm{x})$, порожденные примеры хуже чем с GAN
- GAN: покоазывают лучшие результаты, но менее стабильны и не дают дополнительной информации $\mathrm{p}(\mathrm{x}), \mathrm{p}(\mathrm{z}|\mathrm{x})$;

Материалы

- Stanford, cs231n, Lecture 13. Generative Models, 2017
- Глубокое обучение Погружение в мир нейронных сетей. 8.2 Порождающие модели, 8.3 Состязательные сети