Генеративные модели

Bio

- ***Карагодин Никита***
- Бакалавриат физики 2023
- Магистратура МОВС\ИИ ВШЭ 2025
- *SberDevices MLE* (*TTS*)

План

- Применение генеративных моделей
- Главное из статистики
- Энтропия и информация
- Задачи генеративного моделирования
- Авторегрессионные модели

Применение генеративных моделей

Применение генеративных моделей

Современные генеративные модели могут работать с различными модальностями

Natural Language Processing: *GPT*

$GPT-1$ (2018 г) заложила основу текстовых генеративных моделей:

- Требовала дообучения под конкретную задачу

$GPT-2$ в 10 раз больше (1,5 млрд параметров):

- Текст стал более осмысленным
- Научилась выполнять задачи без дополнительного обучения ***(zero shot)***

$GPT-3$ уже до 175 млрд параметров:

- Гораздо больше обучающих данных
- Научилась справляться с задачами по примерам ***(few shot)***

Natural Language Processing: *GPT*

$GPT - 4$ (2023 г) имеет $\approx$ 1,76 трлн параметров:

- Увеличилось окно контекста
- Мультимодальность
- Режим размышления ***(thinking mode)***

$GPT - 5$ (2025 г) имеет до 0,635 трлн параметров:

- Увеличилось окно контекста $(128k \to 272k)$
- Увеличилась производительность

Natural Language Processing: *Gemini*

- Разработана как конкурент $GPT-4$
- Сразу создана как мультимодальная модель

Есть *3* версии:

- ***Ultra***: самая мощная
- ***Pro***: универсальная
- ***Nano***: для мобильных устройств

- ***Gemini* 1,5 (2024)**: огромное окно контекста до *1* млн токенов
- ***Gemini* 2,5 (2025)**: фокус на скорости ***(Flash*** версия***)*** и режим мышления ***(Deep Research)***

Natural Language Processing: *Claude*

- Разработан компанией ***Anthropic***, основанной бывшими исследователями ***OpenAI***
- Фокус – на безопасность и управляемость

Принципы конституционного ИИ:

- Модель следует набору правил, которые заложены в процесс обучения
- Безопасность встроена в архитектуру и не полагается на дообучение

Natural Language Processing: *Perplexity*

- Объединяет *LLM* с поиском в реальном времени
- Ищет информацию в интернете, генерирует ответ и предоставляет источники

Преимущества*:*

- ***Актуальность:*** предоставляет самые свежие данные
- ***Надёжность:*** снижает галлюцинации за счет ссылок на источники
- ***Прозрачность:*** показывает как был получен ответ

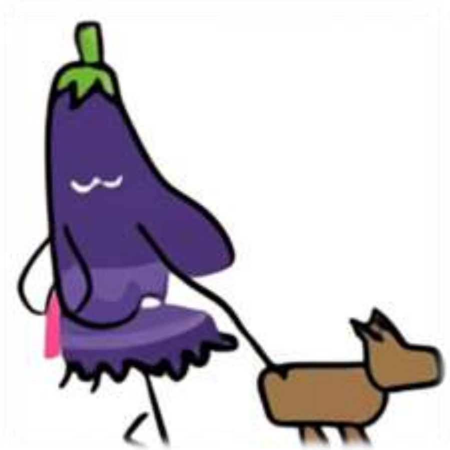
Natural Language Processing: *DeepSeek*

- Разработана китайской исследовательской группой ***DeepSeek***
- Является ***open-source*** моделью
- Модель очень большая, но при этом эффективная благодаря архитектуре ***Mixture-of-Experts*** (***MOE***)

Image Generation: *DALL – E*

- Модель от ***OpenAI***, доступная в продуктах ***OpenAI*** и ***Microsoft***
- Названа в честь Сальвадора Дали и робота ВАЛЛ-И

Promt: *An illustration of an eggplant in a tutu walking a dog*



DALL- E 1

DALL- E 2

Promt: *A hand – drawn sailboat circled by birds on the sea at sunrise*

DALL- E 2

DALL- E 3

Image Generation: Midjourney

Midjourney

- Коммерческий проект, запущенный в 2022 году
- Взаимодействие происходит через чат-бота в ***Discord***

Promt: *Pixel art scene, the eiffel tower at midnigh, city lights, romantic*

V1

V2

V3

V4

V5

Audio: WaveNet

- Разработана ***DeepMind*** (***Google***)
- Используется авторегрессионный подход *(1D Conv)*
- Впервые получили речь, очень похожую на человеческую
- Очень медленная генерация

Parametric

Concatenative

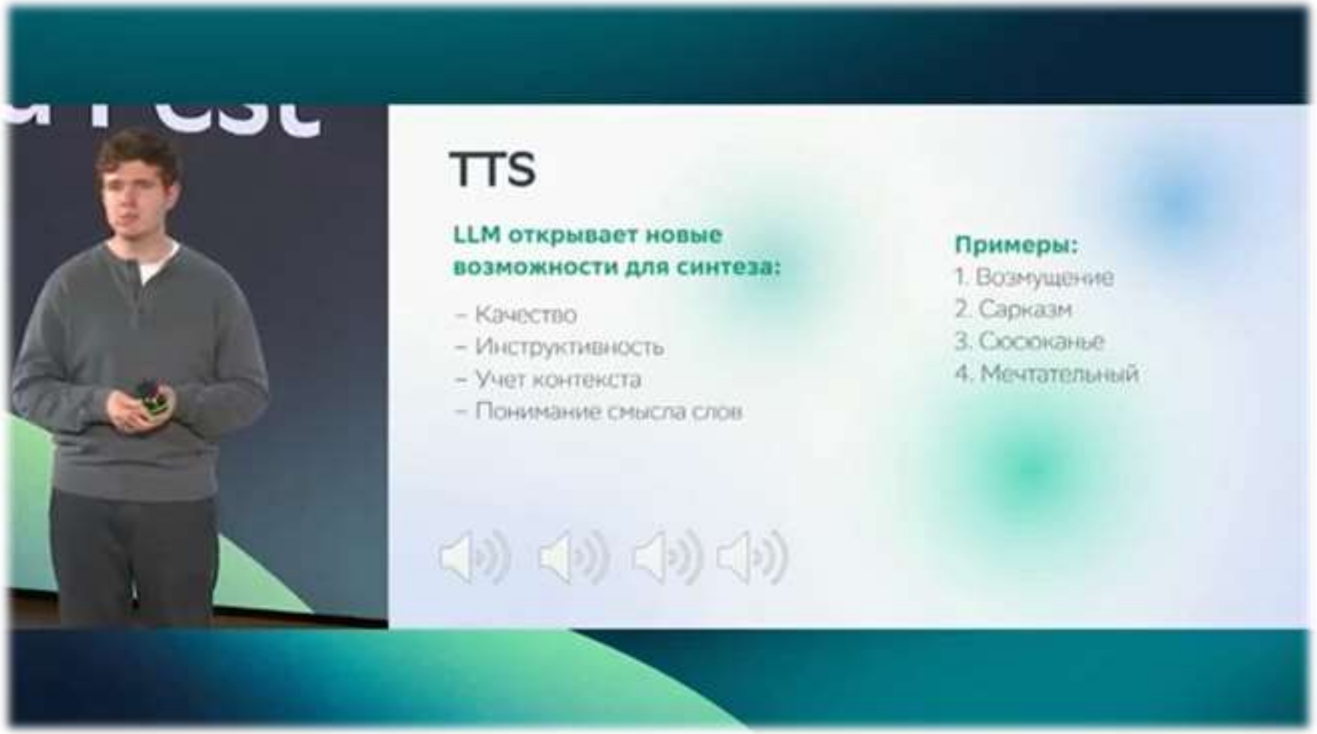
WaveNet

$Audio\!: VITS$

- Разработана в 2021 году
- ***End-to- End*** модель, сразу и текста получает аудио
- Использует зоопарк генеративных моделей: $\boldsymbol{AR, Flow, VAE, GAN}$

Audio: *LLM Speech Synthesis*

- Модели управляются богатыми текстовыми промтами, которые описывают не только что сказать, но и как это сказать
- Модели используют связку ***Tokenizer*** + ***LLM*** + ***Vocoder***



Video: *Sora*

- Разработана компанией ***OpenAI***
- Создает реалистичные видеоролики *(*без звука*)*
- Хорошо понимает окружающий мир и физику

Video: *Veo*

- Разработана в ***Google DeepMind***
- Генерирует качественные видеоролики со звуком

Главное из статистики

Метод максимального правдоподобия

Основная идея ММП – выбрать такие параметры модели $\boldsymbol{\theta}$, при которых наблюдаемые данные становятся наиболее вероятными

Пусть у нас есть выборка $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$ из n независимых и одинаково распределнных наблюдений *(i.i.d)* из некоторого распределения $p_{\boldsymbol{\theta}}(\mathbf{X})$

Правдоподобие для нашей выборки – это произведение плотностей каждого отдельного объекта:

$$f_{\boldsymbol{\theta}}(\mathbf{X}) = f(\boldsymbol{X}|\theta) = \prod_{i=1}^{n} p_{\boldsymbol{\theta}}(\mathbf{x}_i)$$

$f(\theta|\boldsymbol{X})$ – функция правдоподобия

Цель – найти такие значения $\boldsymbol{\theta}$, которые максимизируют $f(\theta|\boldsymbol{X})$

Метод максимального правдоподобия

Вместо функции правдоподобия обычно используют её логарифм – логарифмическую функцию правдоподобия $L_{\boldsymbol{\theta}}(\mathbf{X})$:

$$L_{\boldsymbol{\theta}}(\mathbf{X}) = \log f_{\boldsymbol{\theta}}(\mathbf{X}) = \log \prod_{i=1}^{n} p_{\boldsymbol{\theta}}(\mathbf{x}_i) = \sum_{i=1}^{n} \log p_{\boldsymbol{\theta}}(\mathbf{x}_i)$$

Оценкой максимального правдоподобия (ОМП) параметров $\boldsymbol{\theta}$ называют такие значение параметров, которые максимизируют функцию правдоподобия:

$$\boldsymbol{\theta}^*_{\text{ОМП}} = \underset{\boldsymbol{\theta}}{argmax}\ L_{\boldsymbol{\theta}}(\mathbf{X})$$

Метод максимального правдоподобия

1) Записать функцию правдоподобия $L_{\boldsymbol{\theta}}(\mathbf{X})$

2) Найти градиент $L_{\boldsymbol{\theta}}(\mathbf{X})$ по параметру $\boldsymbol{\theta}$

3) Приравнять градиент к нулю и решить уравнение относительно $\boldsymbol{\theta}$

Метод максимального правдоподобия распространён благодаря ряду свойств оценок:

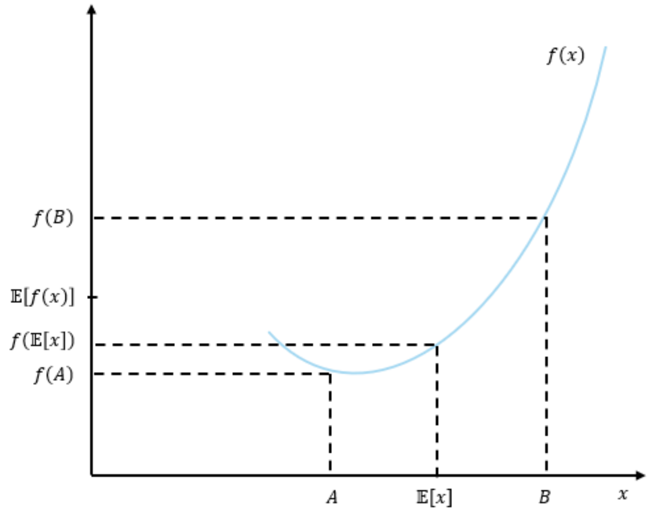
- ✓ *Состоятельность*
- ✓ *Асимптотическая эффективность*
- ✓ *Асимптотическая нормальность*

ММП был введен и подробно описан Рональдом Фишером в 1920-ые годы

Неравенство Йенсена

Для выпуклой функции f и случайной величины X выполняется следующее неравенство:

$$f(\mathbb{E}[X]) \leq \mathbb{E}[f(X)]$$



Для выпуклой функции $f(x)$ среднее значение функции всегда больше или равно значению функции от среднего

Неравенство пригодится при вариационном выводе

Монте-Карло оценка

- Метод Монте-Карло полезен для вычисления многомерных интегралов
- Мы аппроксимируем интеграл средним значением функции

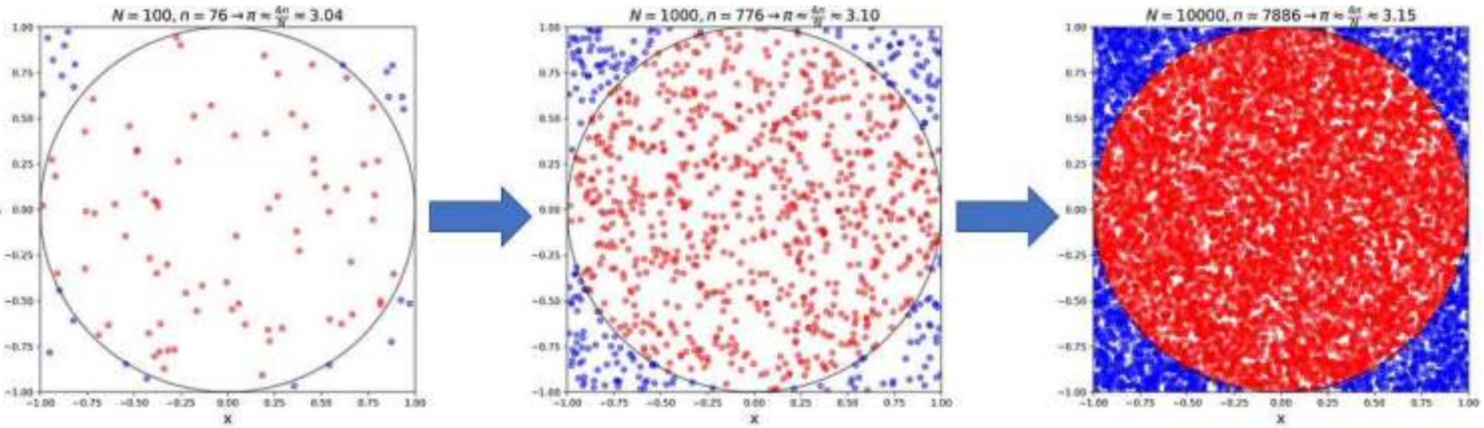
Пусть:

$\mathbf{x} \in \mathbb{R}^m$ – случайная величина с плотностью $p(\mathbf{x})$

$\boldsymbol{f}: \mathbb{R}^m \to \mathbb{R}^d$ – вектор–функция

Тогда оценка математического ожидания:

$$\mathbb{E}_{p(\mathbf{x})}[\boldsymbol{f}(\mathbf{x})] = \int p(\mathbf{x})\boldsymbol{f}(\mathbf{x})d\mathbf{x} \approx \frac{1}{n}\sum_{i=1}^{n}\boldsymbol{f}(\mathbf{x}_i), \qquad \mathbf{x}_i \sim p(\mathbf{x})$$



$$\frac{S_{circle}}{S_{rect}} \approx \frac{n}{N} \approx \frac{\pi}{4}$$

Энтропия и информация

Количество информации

Количество информации должно коррелировать со степенью нашего удивления

- Неожиданное событие → много информации
- Ожидаемое событие → мало информации
- Гарантированное событие → 0 информации

Интуиция:

- Информация $h(x)$ должна быть функцией вероятности $p(x)$
- Для независимых событий информация аддитивна: $h(x, y) = h(x) + h(y)$

Этим двум свойствам удовлетворяет логарифм:

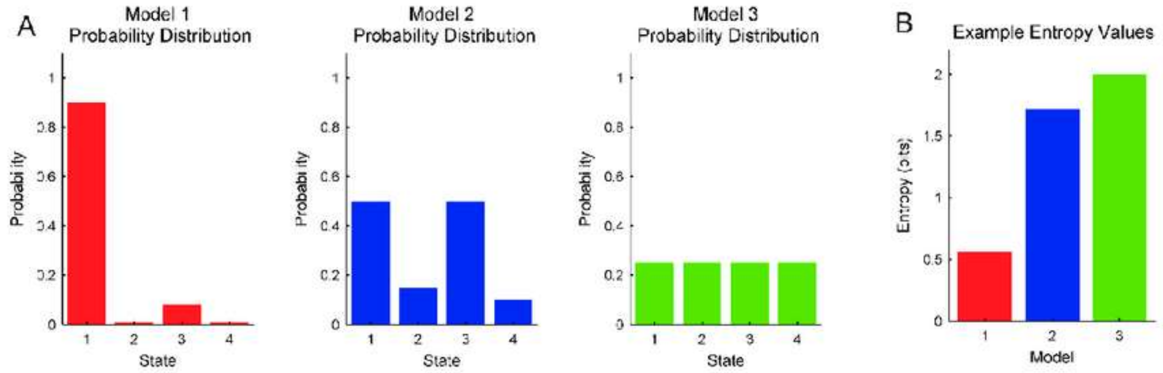
$$h(x) = -\log_2 p(x)$$

Энтропия и мера неопределенности

Среднее количество информации, которое мы получаем наблюдая за исходами случайной величины:

$$H(X) = \mathbb{E}[h(x)] = -\sum_{x} p(x) \log_2 p(x)$$

Эта величина называется ***энтропией***



- Равномерное распределение имеет максимальную энтропию
- Неравномерное распределение (один исход сильно доминирует) имеет низкую энтропию
- Чем выше энтропия, тем менее предсказуемо распределение

Физический смысл энтропии

Энтропия - это теоретический предел для сжатия данных

По теореме Шеннона энтропия $H(X)$ — это минимальное среднее количество бит, необходимого для кодирования одного значения случайной величины

- Низкая энтропия → высокая предсказуемость → данные можно сжать эффективно
- Высокая энтропия → низкая предсказуемость → данные сжимаются плохо

Пример:

- $\{101, 101, 101{,}101{,}101\}$
- $\{010, 001, 101{,}100{,}111\}$

KL –дивергенция

Часто нам приходится заменять истинное и неизвестное распределение $p(x)$ на более простое $q(x)$

KL –дивергенция показывает количество информации, которое мы теряем, когда аппроксимируем $p(x)$ с помощью $q(x)$:

$$D_{KL}(p|q) = \sum_x p(x) \cdot [\,\log p(x) - \log q(x)\,] = \sum_x p(x) \cdot \log \frac{p(x)}{q(x)}$$

$$= \mathbb{E}_{p(x)}\left[\log \frac{p(x)}{q(x)}\right]$$

Свойства:

1. $D_{KL}(p|q) \geq 0$
2. $D_{KL}(p|q) = 0$, если $p = q$
3. Несимметричность: $D_{KL}(p|q) \neq D_{KL}(q|p)$

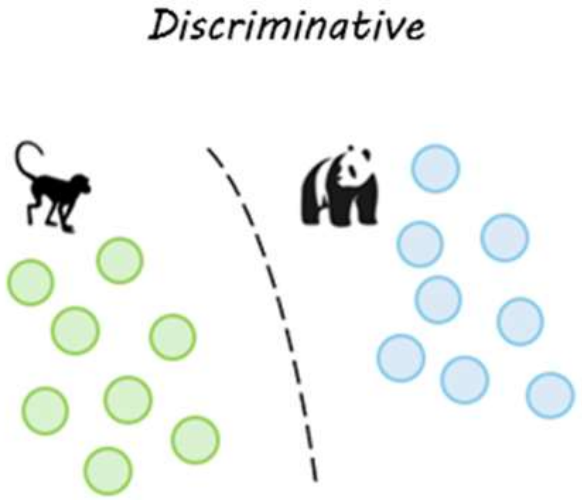
Задачи генеративного моделирования

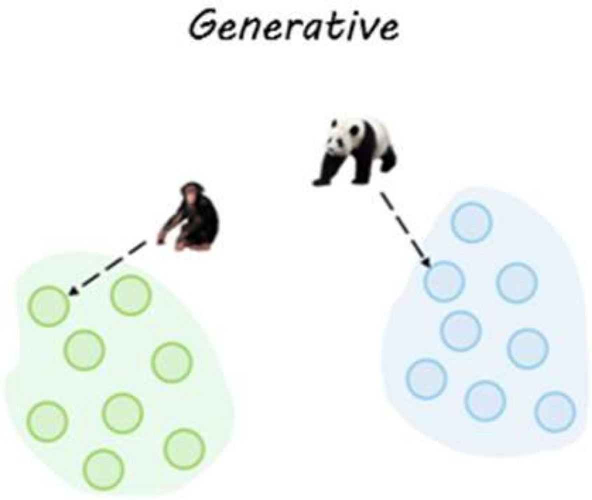
Задачи генеративного моделирования

Модели машинного обучения

Дискриминативные

Генеративные



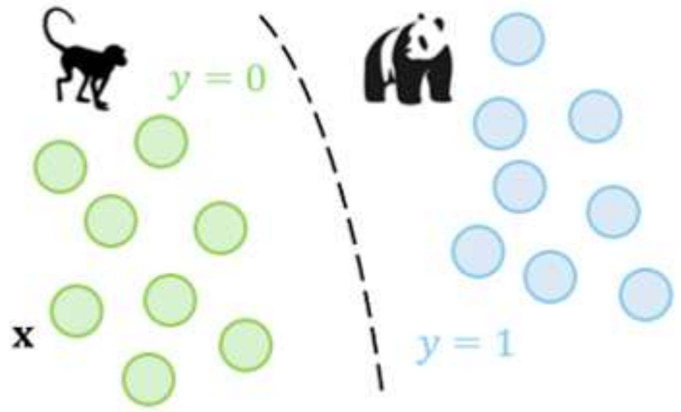


Дискриминативные модели

Предсказывают свойства объекта*:*

- Классификация *(*какой класс*)*
- Регрессия *(*число*)*

Цель: найти распределение $p(y|\mathbf{x})$.

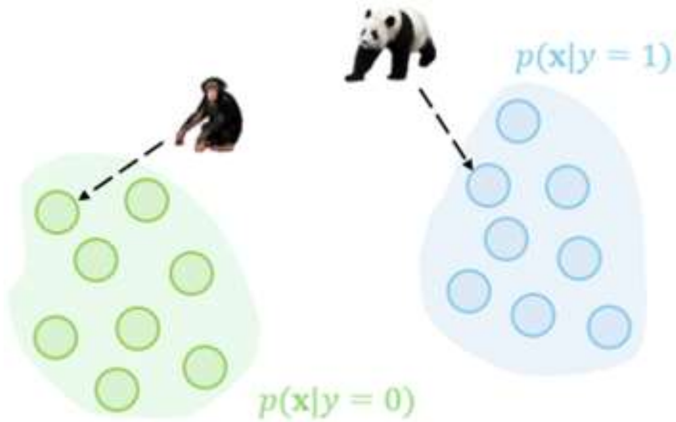


Генеративные модели

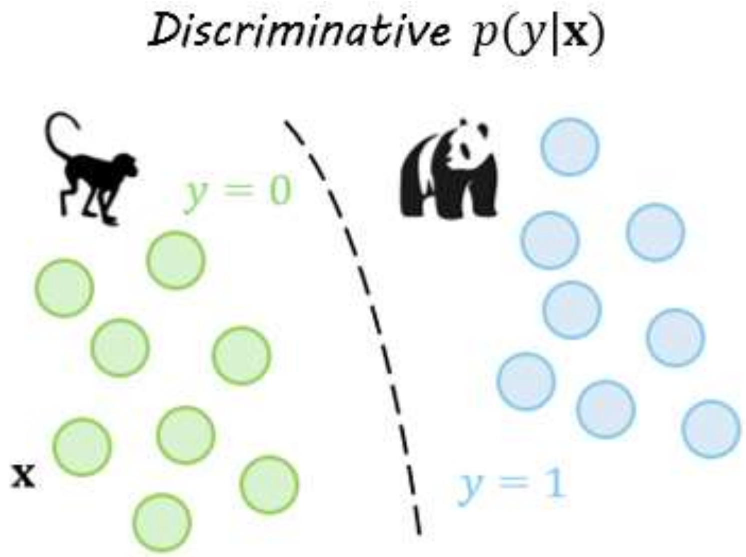
По известным свойствам объекта создают новый объект

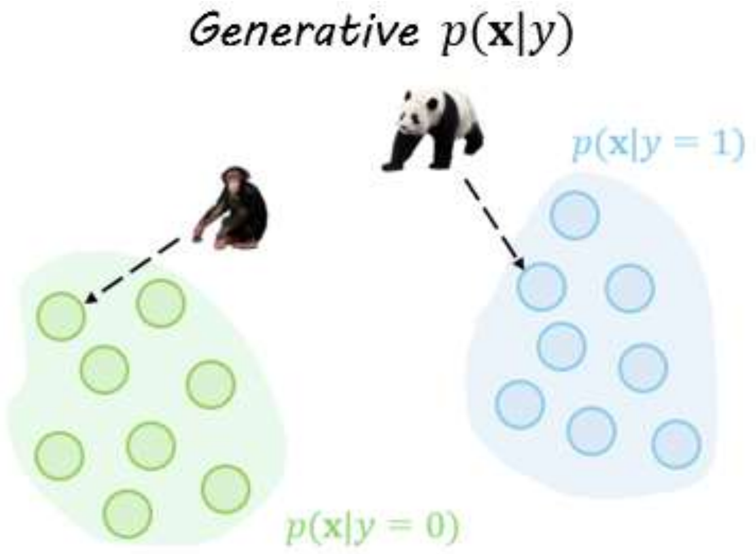
Цель: смоделировать распределение $p(\mathbf{x})$ или $p(\mathbf{x}|\mathrm{y})$.

На одну метку y может приходиться бесконечное количество объектов $\mathbf{x}$.



Задачи генеративного моделирования





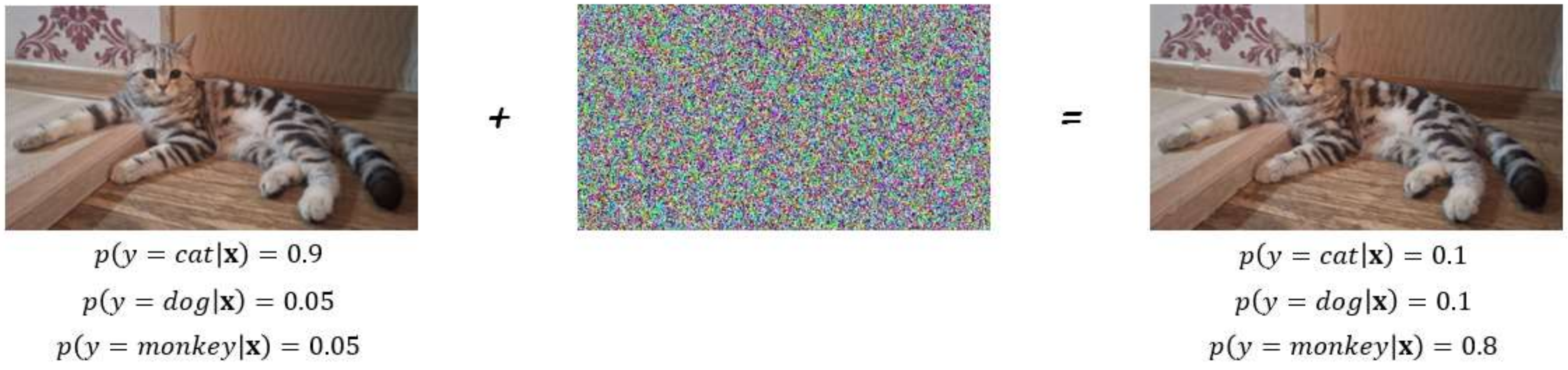
Определяют вероятность характеристики y при известном объекте $\mathbf{x}$

Создают объект $\mathbf{x}$ по заданной характеристике y

Ограничения дискриминативных моделей

Допустим, мы обучили нейросеть, которая классифицирует изображения животных $y \in \mathcal{Y}$

$\mathcal{Y} = \{\text{кошка}, \text{собака}, \text{обезьяна}\}$



Нейросетям, которые выучивают $p(y|\mathbf{x})$ иногда не хватает ***семантического*** понимания изображений

Постановка задачи

Пусть у нас есть набор данных $\{\mathbf{x}_i\}_{i=1}^{n}$,

$\mathbf{x}_i \in \mathbb{R}^m$ – вектор, представляющий объект

$p_{data}(\mathbf{x})$ описывает как объекты распределены в пространстве $\mathbb{R}^m$

Цель: найти неизвестное распределение $p_{data}(\mathbf{x})$

Зная $p_{data}(\mathbf{x})$ мы сможем оценивать вероятность новых объектов и генерировать новые

Проблема высокой размерности

Объекты находятся в пространстве высокой размерности

Прямая оценка $p_{data}(\mathbf{x})$ в таких пространствах затруднена

Найти истинное распределение $p_{data}(\mathbf{x})$ почти невозможно

Будем искать аппроксимацию внутри некоторого заранее заданного ***параметрического семейства распределений*** $p_{\boldsymbol{\theta}}(\mathbf{x})$

$\boldsymbol{\theta}$ – параметры модели

Ищем $\boldsymbol{\theta}$, которые максимизируют правдоподобие:

$$\boldsymbol{\theta}^* = \arg \max_{\boldsymbol{\theta}} \sum_{i=1}^{n} \log p_{\boldsymbol{\theta}}(\mathbf{x}_i)$$

Связь правдоподобия и KL-дивергенции

Наша задача – найти такие параметры $\boldsymbol{\theta}$, при которых $p_{\boldsymbol{\theta}}(\mathbf{x})$ будет максимально близко к $p_{data}(\mathbf{x})$:

$$p_{\boldsymbol{\theta}}(\mathbf{x}) \approx p_{data}(\mathbf{x})$$

Будем искать такие параметры $\boldsymbol{\theta}$, которые минимизируют дивергенцию между истинным и модельным распределениями:

$$\min_{\boldsymbol{\theta}} D(p_{data}(\mathbf{x}) \mid p_{\boldsymbol{\theta}}(\mathbf{x}))$$

Минимизация прямой KL –дивергенции эквивалента максимизации правдоподобия

Связь правдоподобия и KL-дивергенции

Разложим формулу для прямой KL-дивергенции:

$$KL(p_{data}(\mathbf{x})|p_{\boldsymbol{\theta}}(\mathbf{x})) = \int p_{data}(\mathbf{x}) \log \frac{p_{data}(\mathbf{x})}{p_{\boldsymbol{\theta}}(\mathbf{x})} d\mathbf{x} = \int p_{data}(\mathbf{x}) \log p_{data}(\mathbf{x})\, d\mathbf{x} - \int p_{data}(\mathbf{x}) \log p_{\boldsymbol{\theta}}(\mathbf{x})\, d\mathbf{x}$$

$\int p_{data}(\mathbf{x}) \log p_{data}(\mathbf{x})\, d\mathbf{x}$ – энтропия истинного распределения (с минусом)

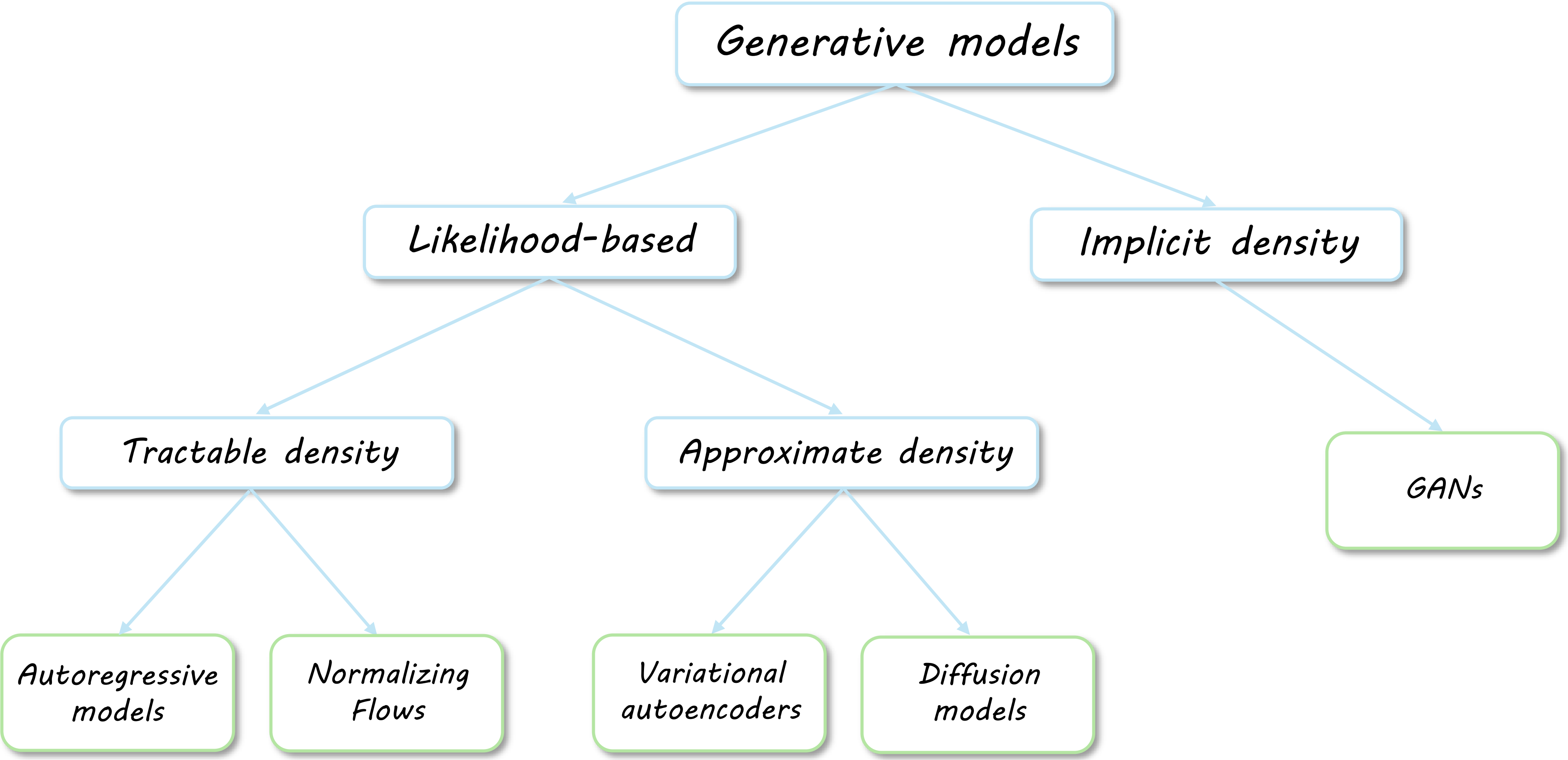
$\int p_{data}(\mathbf{x}) \log p_{\boldsymbol{\theta}}(\mathbf{x})\, d\mathbf{x}$ – матожидание логарифма правдоподобия по истинному распределению

Оценим интеграл:

$$KL(p_{data}(\mathbf{x})|p_{\boldsymbol{\theta}}(\mathbf{x})) \approx const - \frac{1}{n}\sum_{i=1}^{n} \log p_{\boldsymbol{\theta}}(\mathbf{x}_i)$$

$$\min_{\boldsymbol{\theta}} KL(p_{data}(\mathbf{x})|p_{\boldsymbol{\theta}}(\mathbf{x})) <=> \max_{\boldsymbol{\theta}} \frac{1}{n}\sum_{i=1}^{n} \log p_{\boldsymbol{\theta}}(\mathbf{x}_i)$$

Зоопарк генеративных моделей



Авторегрессионные модели

Обучение авторегресионных моделей

Хотим найти параметры $\boldsymbol{\theta}$, которые максимизируют правдоподобие $p_{\boldsymbol{\theta}}(\mathbf{X})$ обучающего набора данных $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_{\mathrm{n}}\}$

Предполагаем, что объекты $\mathbf{x}_{\mathrm{i}}$ являются независимыми, тогда:

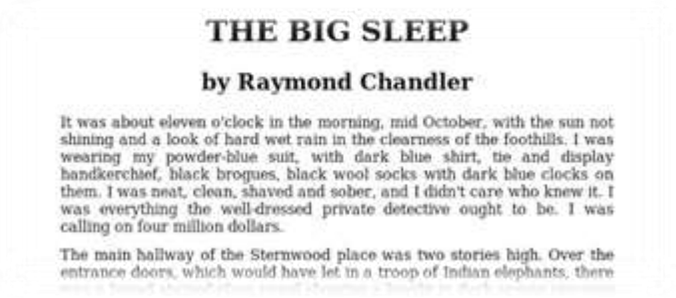
$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{argmax} \log p_{\boldsymbol{\theta}}(\mathbf{X}) = \underset{\boldsymbol{\theta}}{argmax} \sum_{i=1}^{n} \log p_{\boldsymbol{\theta}}(\mathbf{x}_i)$$

Для обучения нужно быстро уметь считать $\log p_{\boldsymbol{\theta}}(\mathbf{x})$ и его градиент $\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\mathbf{x})$

Обучение авторегресионных моделей

Объекты в реальном мире – многомерные случайные величины

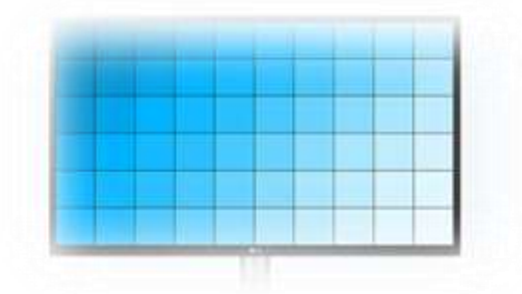
Прямое моделирование $p_{\boldsymbol{\theta}}(\mathbf{x}) = p_{\boldsymbol{\theta}}(\mathrm{x}_1, \mathrm{x}_2, \dots, \mathrm{x}_\mathrm{D})$ – сложная задача из-за проклятия размерности


THE BIG SLEEP

by Raymond Chandler

It was about eleven o'clock in the morning, mid October, with the sun not shining and a look of hard wet rain in the clearness of the foothills. I was wearing my powder-blue suit, with dark blue shirt, tie and display handkerchief, black brogues, black wool socks with dark blue clocks on them. I was neat, clean, shaved and sober, and I didn't care who knew it. I was everything the well-dressed private detective ought to be. I was calling on four million dollars.

The main hallway of the Sternwood place was two stories high. Over the entrance doors, which would have let in a troop of Indian elephants, there



Разложение многомерной плотности

Разложим совместную вероятность на произведение условных:

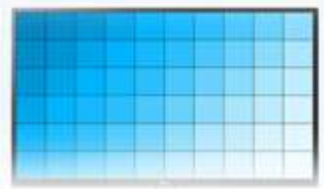
$$p_{\boldsymbol{\theta}}(\mathbf{x}) = p_{\boldsymbol{\theta}}(\mathrm{x}_1, \mathrm{x}_2, \dots, \mathrm{x}_\mathrm{D}) = p_{\boldsymbol{\theta}}(\mathrm{x}_1) \cdot p_{\boldsymbol{\theta}}(\mathrm{x}_2|\mathrm{x}_1) \cdot p_{\boldsymbol{\theta}}(\mathrm{x}_3|\mathrm{x}_1, \mathrm{x}_2) \cdot \dots \cdot p_{\boldsymbol{\theta}}(\mathrm{x}_\mathrm{D}|\mathrm{x}_1, \mathrm{x}_2, \dots, \mathrm{x}_{\mathrm{D}-1})$$

$$= \prod_{j}^{D} p_{\boldsymbol{\theta}}(\mathrm{x}_\mathrm{i}|\mathbf{x}_{1:j-1})$$

$\mathbf{x}_{1:j-1}$ – вектор всех компонент объекта $\mathbf{x}$ до позиции j

Теперь задача максимизации правдоподобия принимает вид:

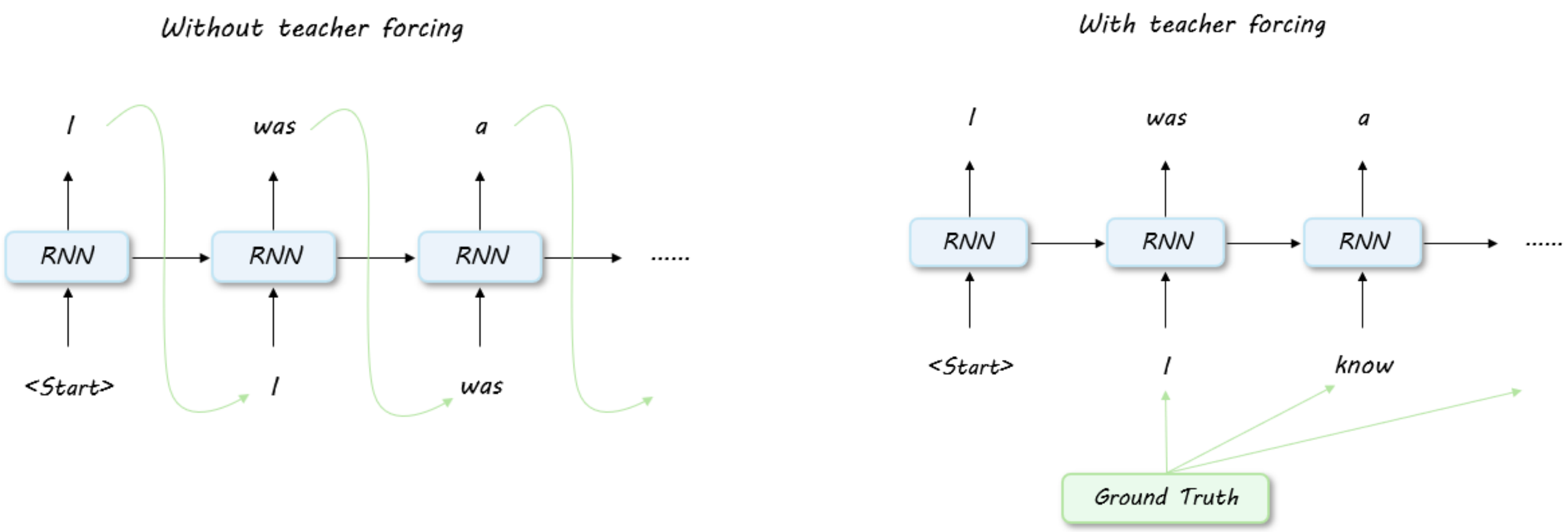
$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{argmax} \sum_{i=1}^{n} \sum_{j}^{D} \log p_{\boldsymbol{\theta}}(x_{ij}|\mathbf{x}_{i,1:j-1})$$



Обучение авторегресионных моделей

В начале обучения модель часто совершает ошибки, которые с каждым шагом накапливаются

Решение:
Принудительно подавать на вход модели правильные данные (***teacher forcing***)



При генерации модель использует собственные предсказания

Авторегрессия в RNN

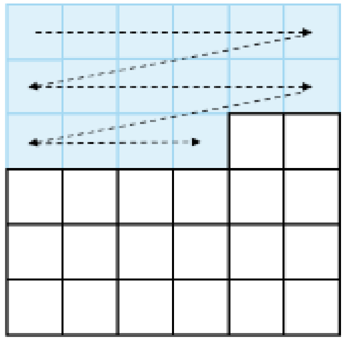
RNN по своей природе являются авторегрессионными:

$$\boldsymbol{h}_t = f(\boldsymbol{W}_{hh}\boldsymbol{h}_{t-1} + \boldsymbol{W}_{xh}\boldsymbol{x}_t + \boldsymbol{b}_h)$$

$\boldsymbol{W}_{hh}, \boldsymbol{W}_{xh}, \boldsymbol{b}_h -$ обучаемые параметры

Звук – тоже последовательность, и RNN могут предсказывать следующий отсчёт

- На изображениях тоже можно задать порядок
- Обычно используют растровый порядок ***(Raster Scan Order)***
- Одной из первых таких моделей была ***PixelRNN***

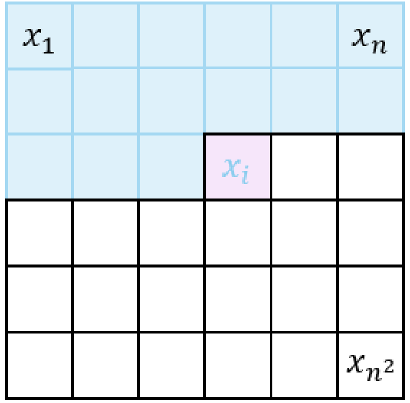


Авторегрессия в CNN

- RNN были очень медленными и при обучении, и при генерации
- Для ускорения обучения стали использовать свёрточные сети

PixelCNN:

- Предложили использовать маскированные свертки
- На ядра свёртки накладывается маска, которая блокирует доступ к будущим пикселям



1	1	1
1	0	0
0	0	0

Недостатки $PixelCNN$

Недостатки:

- Генерация все еще последовательная
- Есть слепое пятно
- Неэффективные предсказания

Последние две проблемы были решены в $\boldsymbol{Gated\ PixelCNN}$ и $\boldsymbol{PixelCNN++}$

Всем

О КАК

ЛЕКЦИЯ ЗАКОНЧИЛАСЬ