Contents

Генеративные модели

Лекция 1: Генеративные модели

- Выход модели = Изображение
  - Увеличение четкости
  - Перенесение стиля
  - ...

  Евклидово расстояние

-
  - Сумма разниц между i-ми пикселями
  - Не измеряет расстояние содержательно
- **Perceptual loss**
  - Сравнивается контент изображений
  - Content loss

$$L^l{}_{\text{content}} = \sum_{ij} (A_{\text{ij}} - B_{\text{ij}})^2$$

- Берем i свертку на l слое и сравниваем отклики, j - отдельный элемент свертки
- Лучше брать последние слои
- Style loss

$$G^l{}_{\text{ij}}(A) \;=\; A^l{}_{\text{ik}} A^l{}_{\text{jk}}$$

- Сравниваем похожесть каналов

$$L^l{}_{\text{style}} \;=\; (G^l{}_{\text{ij}}(A) \;-\; G_{\text{ij}}{}^l(B))^2$$

$$L_{\text{style}} = \sum_l L^l{}_{\text{style}}$$

Perceptual loss

$$L(C,\ S,\ X)\ =\ \alpha L_{\text{content}}(C,\ X)\ +\ \beta L_{\text{style}}(S,\ X)\ \to \min_X$$

C - исходное изображение

S - стилевое изображение

X - выходное изображение

Обучаемые параметры = пиксели выходной картинки

Недостатки

- Очень долгая обработка

Ускорение

- Фиксируем стилевое изображение
- Обучим модель $\alpha(C)$, которая оптимизирует Perceptual loss
- Теперь подбираем не пиксели, а обучаем модель
- Оптимизируем по параметрам модели α
- ECCV16
- Image Transform Network похожа на U-Net

Superresolution

- Повысить разрешение картинки
- Подход в лоб
  - Расширяем картинку
  - Сетка выучивает дополнение к исходной картинке, чтобы MSE был минимальный между результатом и правильной картинкой
  - The deeper, the better

  Можно использовать perceptual loss
- – Но тогда появляются артефакты

Inpainting

- Восстановить кусок картинки

Лекция 2: Вариационные автокодировщики

- Denoising autoencoder

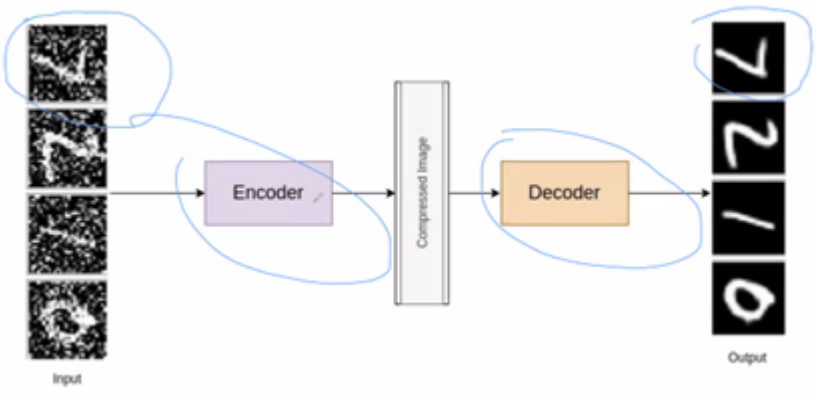


Figure 1: Denoising

Если берем случайную точку, на которой модель не обучалась → Получаем бред

Вероятностные методы

- Иногда проще описать в терминах вероятности
- Подбираем параметры распределений так, чтобы обучающая выборка имела высокую вероятность
- Тематическое моделирование
  - PLSA, LDA
  - Данные - набор текстов
  - Каждая тема = распределение N слов
  - Каждый текст = распределение на темах
  - Генерация
    - Выбираем тему
    - Выбираем слова и добавляем в текст

  Изображения

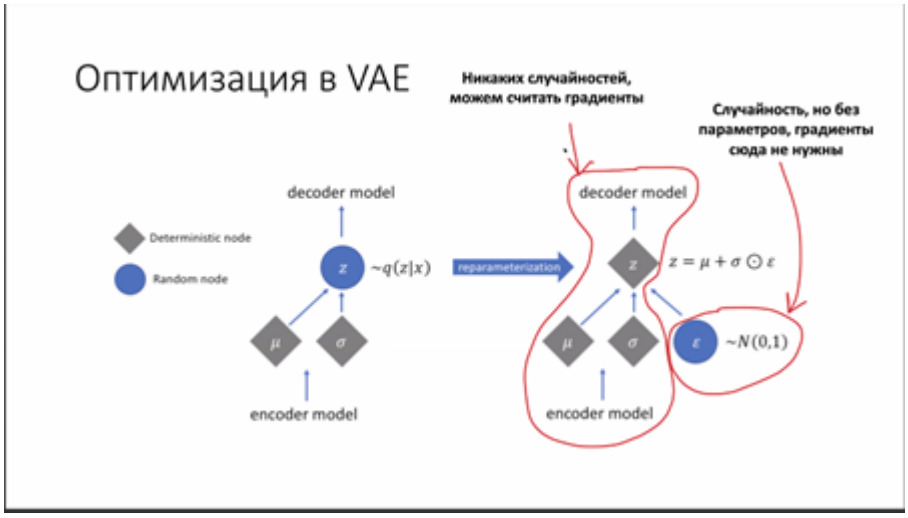
-
  - Хотим построить пространство представлений
  - Каждая картинка != точка, а **распределение**

  - Возьмем нормальное распределение

$$encoder(x) \ = \ (\mu(x), \ \sigma(x))$$

- МО и Дисперсия - вектор размера d
- Сэмплируем вектор Z из распределения сμи σ
- Декодировщик вектор разворачивает в картинку
- Раскодированная картинка должна быть как можно ближе к исходной
- Позволяет получить непрерывное пространство представлений
- $q(z|x)$- кодировщик
- $p(x|z)$- декодировщик $decoder(z) \ + \ \epsilon$
- $(E_{q(z|x_i)} log \ p(x_i|z) \ - \ KL(q(z|x) \ || \ N(0, \ 1))) \ \rightarrow \max$
- KL - Дивергенция Кульбака - Лейблера
  - Мера расстояния между распределениями
  - $$KL(q, \ p) \ = \ q(x) log \frac{q(x)}{p(x)}$$
  - Выполняет функцию регуляризации - чтобы не было узких распределений

Оптимизация - Reparametrization Trick

- 

Лекция 3: GAN

- Дано: выборка $x_i = p_x(x)$, распределение неизвестно
- Задача: генерировать новые объекты с таким же распределением
- Генерируем z из двумерного нормального распределения → пространство Z называется скрытым → Пропускаем через генератор G(Z) → получаем $\widehat{X} = G(Z)$
- Как посчитать сходство между X и$\widehat{X}$
  - Строим задачу бинарной классификации X и$\widehat{X}$ → Дискриминатор
  - Используем значения log-loss для обучения дискриминатора
  - $$L = -\frac{1}{n} y_i log D(x_i) + (1 - y_i) log(1 - D(x_i))$$
  - $$L(D,\ G) = -\frac{1}{n} log D(x_i) - \frac{1}{n} log(1 - D(G(z_i))$$
  - Обучение дискриминатора и генератора
    - $$\max_G \min_D L(D,\ G)$$

    Проблемы GAN

  - * Затухание градиентов
      - Дискриминатор идеально разделяет выборки → Loss = 0

    Схлопывание мод

    * - Несколько мод в распределении → получается выучить не все
      - Для некоторых мод дискриминатор возвращает 1 → в окрестности нет градиентов
  - **Wasserstein GAN**
    - Меняем функцию потерь
    - Расстояние Вассерштейна

· $EMD(P_r,\ P_\theta)\ =\ inf..$

· Дуальная форма:

$EMD(P_r,\ P_\theta)\ =\ sup_{||f||L<1}$

* Нет проблемы затухания градиентов
* Схлопывание мод

– **Conditional GAN**

* Подаем дополнительно класс

– **Cycle GAN**

* Есть дополнительная нейросеть, которая может переводить генерированные картинки обратно

– **Bidirectional GAN**

Лекция 4: Нормализационные потоки

Flow-based generative models

- Не 2 сети, а одна → Декодируем представление с помощью обратной функции
- Не используем x' для обучения
- VAE и GAN преобразуют скрытое пространство в данные необратимо
- Нормализационные потоки выучивают **обратимое** преобразование
- Позволяют рассчитать плотность вероятности для распределения
  – Можно обнаружить аномалии / редкость

Теорема о замене переменных

- Пространство признаков x
- Скрытое пространство z
  – Знаем функцию распределения
- $$(z\ =\ f(x)$$
- Находим p_x

- Знаем $p_z(z),\ z\ =\ f(x)$

$$p_x(x_i)\ =\ p_z(f(x_i))|det\frac{\partial f(x_i)}{\partial x_i}|$$

$$p_z(z_i)\ =\ p_x(f^{-1}(z_i))|det\frac{\partial f^{-1}(z_i)}{\partial z_i}|$$

Нормализационные потоки

- Есть признаки x - не знаем их распределение
- Надо найти $z\ =\ f(x)$, чтобы z имело известное распределение
- Используем метод градиентного спуска
  - Подбираем f с помощью логарифмической функции правдоподобия

$$L\ =\ -\frac{1}{n}\log\ p_x(x_i)$$

$$p_x(x_i)\ =\ p_z(f(x_i))|det\frac{\partial f(x_i)}{\partial x_i}|$$

$$L\ =\ -\frac{1}{n}log\ (p_z(f(x_i))|det\frac{\partial f(x_i)}{\partial x_i}|)$$

$$L\ =\ -\frac{1}{n}\log\ p_z(f(x_i)\ +\ log(|det\frac{\partial f(x_i)}{\partial x_i}|)\ \to \min$$

- Прямая функция используется для переведения входных данных в случайный шум → Обратная функция генерирует картинки из полученного шума
- Алгоритм обучения
  - Берем минибатч
  - Считаем лосс
    - $L\ =\ -\frac{1}{n}\log\ p_z(f(x_i)\ +\ log(|det\frac{\partial f(x_i)}{\partial x_i}|)$
  - Обновляем параметры функции f

- Алгоритм генерации
  - Сэмплим точки из распределения z
  - Генерируем объекты, используя обратную функцию $f^{-1}(z)$

  Можно сделать несколько слоев
-
  - Несколько промежуточных скрытых представлений
  - За счет этого можно научиться генерировать более сложные объекты
  - $z_i = f_2(y_i),\ y_i = f_1(x_i)$

$$p_x(x_i) = p_y(f_1(x_i))|det\frac{\partial f_1(x_i)}{\partial x_i}|$$

$$p_y(y_i) = p_z(f_2(y_i))|det\frac{\partial f_2(y_i)}{\partial y_i}|$$

$$p_x(x_i) = p_z(f_2(y_i))|det\frac{\partial f_2(y_i)}{\partial y_i}||det\frac{\partial f_1(x_i)}{\partial x_i}|$$

- Выбор функции
  - Дифференцируема?
  - Обратима?

Real-NVP

- X и Z - d-мерные вектора

$$z = f(x) = \begin{cases} z_{1:d} = x_{1:d} \\ z_{d+1:D} = x_{d+1:D} \odot \exp(s(x_{1:d})) + t(x_{1:d}) \end{cases}$$

где:

- $z_{1:d}$ - первые d компонент вектора z;
- $s(x_{1:d})$ и $t(x_{1:d})$ – нейронные сети с d входами и $D - d$ выходами;
- $\odot$ - поэлементное умножение.

- Матрица первых производных:

$$\frac{\partial \boldsymbol{f}(x)}{\partial x} = \begin{pmatrix} \mathbb{I}_d & 0 \\ \frac{\partial z_{1:d}}{\partial x_{1:d}} & diag\left(\exp(\boldsymbol{s}(x_{1:d}))\right) \end{pmatrix}$$

- Значение Якобиана:

$$\left|\det \frac{\partial \boldsymbol{f}(x)}{\partial x}\right| = \exp\left(\sum_{j=d+1}^{D} \boldsymbol{s}(x_{1:d})_j\right)$$

$$x = \boldsymbol{f}^{-1}(z) = \begin{cases} x_{1:d} = z_{1:d} \\ x_{d+1:D} = (z_{d+1:D} - \boldsymbol{t}(x_{1:d})) \odot \exp(-\boldsymbol{s}(x_{1:d})) \end{cases}$$

Masked Autoregressive Flow (MAF)

$$z = \boldsymbol{f}(x) = \begin{cases} z_1 = (x_1 - \mu_1)\exp(-s_1) \\ z_d = (x_d - \boldsymbol{\mu}_d(x_{1:d-1})) \odot \exp(-\boldsymbol{s}_d(x_{1:d-1})) \end{cases}$$

где:

- $z_{1:d}$ - первые d компонент вектора z;
- $\boldsymbol{\mu}_d(x_{1:d-1})$ и $\boldsymbol{s}_d(x_{1:d-1})$ – **нейронные сети** с $d-1$ входами и 1 выходом;
- $\odot$ - поэлементное умножение.

- Матрица первых производных **нижнетреугольная**:

$$\frac{\partial \boldsymbol{f}(x)}{\partial x} = \begin{pmatrix} \exp(-\boldsymbol{s}_1) & \cdots & 0 \\ \vdots & \ddots & \vdots \\ \frac{\partial z_D}{\partial x_1} & \cdots & \exp(-\boldsymbol{s}_D(x_{1:D-1})) \end{pmatrix}$$

- Значение Якобиана:

$$\left|\det \frac{\partial \boldsymbol{f}(x)}{\partial x}\right| = \exp\left(-\sum_{j=1}^{D} \boldsymbol{s}_d(x_{1:d-1})\right)$$

$$x = \boldsymbol{f}^{-1}(z) = \begin{cases} x_1 = z_1 \exp(s_1) + \mu_1 \\ x_d = z_d \exp(\boldsymbol{s}_d(x_{1:d-1})) + \boldsymbol{\mu}_d(x_{1:d-1}) \end{cases}$$

- Каждая компонента использует предыдущую → Дольше

Inverse Autoregressive Flow (IAF)

$$z = f(x) = \begin{cases} z_1 = (x_1 - \mu_1)\exp(-s_1) \\ z_d = (x_d - \mu_d(z_{1:d-1}))\exp(-s_d(z_{1:d-1})) \end{cases}$$

$$x = f^{-1}(z) = \begin{cases} x_1 = z_1 \exp(s_1) + \mu_1 \\ x_d = z_d \exp(s_d(z_{1:d-1})) + \mu_d(z_{1:d-1}) \end{cases}$$

- Генерация быстрее, так как используем компоненты вектора z, который мы и так знаем
- Обучение дольше, так как z обучается попиксельно

Обработка звука

Лекция 5: Диалоговые системы

- Прежде всего интересен текстовый формат
  - Большинство данных в диалоговых системах переводится в текст

  Чат-боты

-
  - Task-oriented → Узко специализированный
  - Голосовые помощники = большое количество task-oriented скиллов
- Типы ответов в диалоговых системах
  - Системы с готовыми ответами (Retrieval-based)
    - Работает с пулом реплик

    Generative-based

- Основные компатенты чат-бота
  - Intent Detection

    - Как обработать входную реплику, определить намерение пользователя
    - Задача классификации → Какому приложению отправить реплику
    - Выбор классификатора интента
        - В облаке → Можно использовать сложную модель
        - На устройстве → Поменьше
        - Bert, Distilled Bert
        - Можно даже не использовать машинное обучение → Ищем ключевые слова в предложениях

  Slot Filling

- – Выделить из сообщения параметры
    - “Поставить будильник на 7 утра”
    - Слоты определяются на основе задачи
    - Извлечение именованных сущностей
        - Относится ли текущее слово к одному из классов, задающих слоты
        - Рекуррентные нейронные сети → Извлекаем скрытое состояние для каждого отдельного слова
    - Можно делать последовательно → Сначала определяем интент → Выделяем слоты
    - Можно решать задачи одновременно
  - Граф сценария
    - Какие вещи нужно доспросить, какие слоты заполнить

Вопросно-ответные системы, задача SQuAD

- Нейросетевая модель ищет ответ на вопрос
- Должна:
  - Понять вопрос на естественном языке
  - Найти ответ в массиве данных

- Выдать ответ на естественном языке

Типы вопросов

- Фактологические вопросы
  - Легкие вопросы для нейронных сетей

  Вопросы на понимание здравого смысла
  - Сложны для нейросетей
- Сравнительные вопросы
  - Тоже сложно

Подходы к построению QA систем

- Информационный поиск
  - Текстовая коллекция, разбитая на фрагменты

  Поиск по базе знаний
  - Формируем запрос к базе знаний как функцию → Она выдает ответ
  - Более сложный подход
- Задача SQuAD
  - Ищем ответ на вопрос в тексте
  - Stanford Question Answering Dataset
    - Текст вопроса
    - Текстовый фрагмент, содержащий ответ
    - Текст ответа

    Решение задачи SQuAD
  - Отбор параграфов, которые могут содержать ответ на вопрос
    - Обычно используют грубые и быстрые методы
    - Косинусное расстояние между TF-IDF / FastText вопроса и параграфа

Поиск подстрок фрагмента с ответом

* · Трансформеры

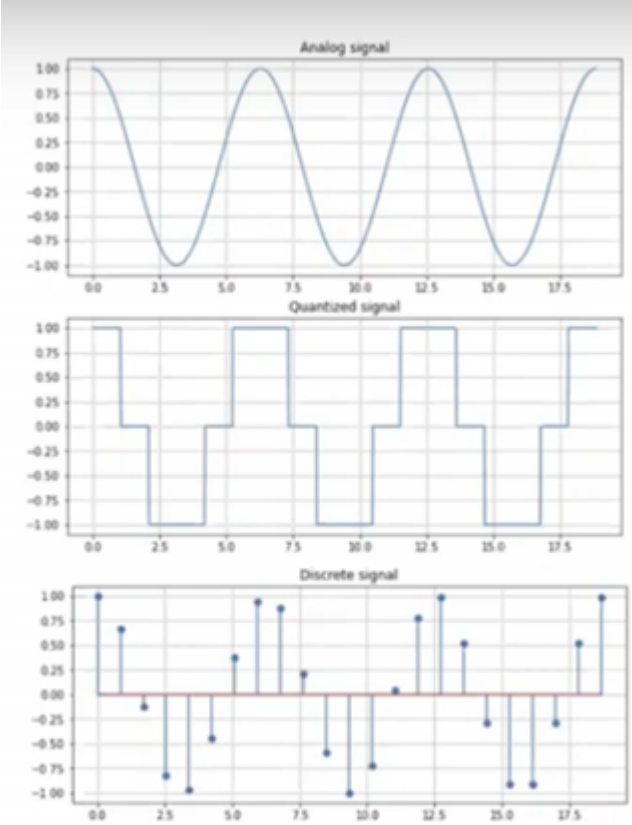
  · Ищем максимальное произведение вероятностей, что первый токен последовательности - начало ответа, а последний токен - конец ответа

Лекция 6: Обработка звука

- Звук - колебания воздуха
- Аналоговый сигнал подвергается дискретизации, квантованию, кодированию
  - Квантование
    * Сигнал разбивается на N уровней
    * Для каждой точки берется ближайший уровень

  Дискретизация

  – * Сигнал представляется в виде последовательных значений, взятых в дискретные моменты времени t с шагом d
    * В виде точек, а не функции



- Характеристики
  - Частота дискретизации - количество отсчетов амплитуды в секунду
  - Количество каналов
- Почему плохо работать со звуком в таком формате
  - Один звук состоит из 2000-4000 амплитуд → дорого хранить и обрабатывать
  - Нет инварианта относительно шума и трансформаций → лучше использовать спектрограмму

Дискретное преобразование Фурье

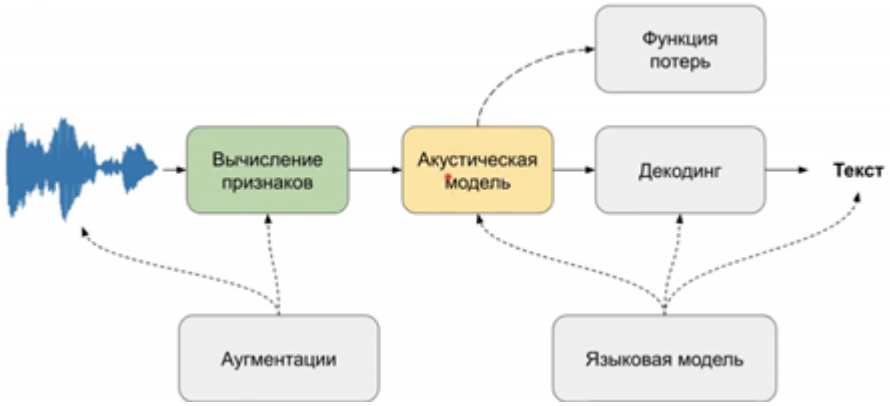
$$X = Mx$$
$$Mmn = exp(-2\pi i \frac{(m-1)(n-1)}{N})$$

- Спектрограмма
  - Нарезаем сигнал на окна с пересечением
  - Применяем оконную функцию к вырезанному окну
  - Применяем дискретное преобразование Фурье
  - Считаем квадрат комплексной нормы
  - Берем половину вектора + 1 в силу его симметричности
    - Свойство преобразования Фурье

  Мелспектрограмма

$$m = 2595 \log_{10}\left(1 + \frac{f}{700}\right) = 1127 \ln\left(1 + \frac{f}{700}\right)$$
$$f = 700\left(10^{\frac{m}{2595}} - 1\right) = 700\left(e^{\frac{m}{1127}} - 1\right)$$

- В конце берем **логарифм** от спектрограммы
- Из нее хуже звук восстанавливается
- Обратимость операции не точная



- Метрики

$$Word\ Error\ Rate\ =\ \frac{S\ +\ D\ +\ I}{S\ +\ D\ +\ C}$$

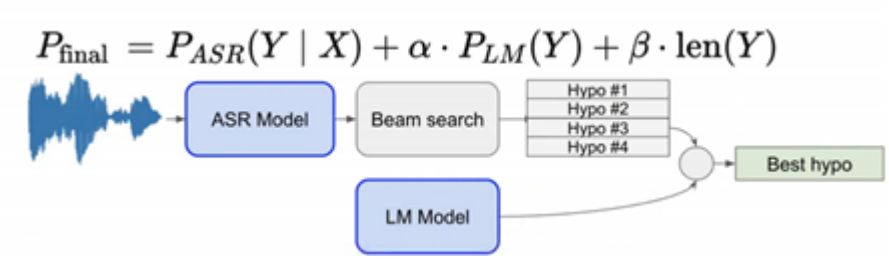
- S - кол-во замен
- D - кол-во удалений
- I - кол-во вставок

- C - кол-во совпадений
- **CER** - посимвольное совпадение

- **Listen, Attend, Spell**
  - Listener = пирамидальный LSTM-encoder

Выходы для каждого слоя конкатенируем и подаем в следующую лстм

- Speller = LSTM декодер
- Attention
- Минимизируем кросс-энтропию с правильными буквами
- Учим с помощью teach-forcing'a
- Декодируем с помощью beam search
- Добавляем языковую модель для улучшения результата
- Second pass fusing



- Shallow fusing
  - Добавляем языковую модель в beam search

Лекция 7: Обработка звука 2

Connectionist Temporal Classification

- Вход и выход разной длины и они не выровнены
- Алгоритм
  - Для каждого фрейма делаем предсказание буквы
  - Склеиваем соседние одинаковые предсказания
  - Для поиска пробелов используют эпсилон-токены

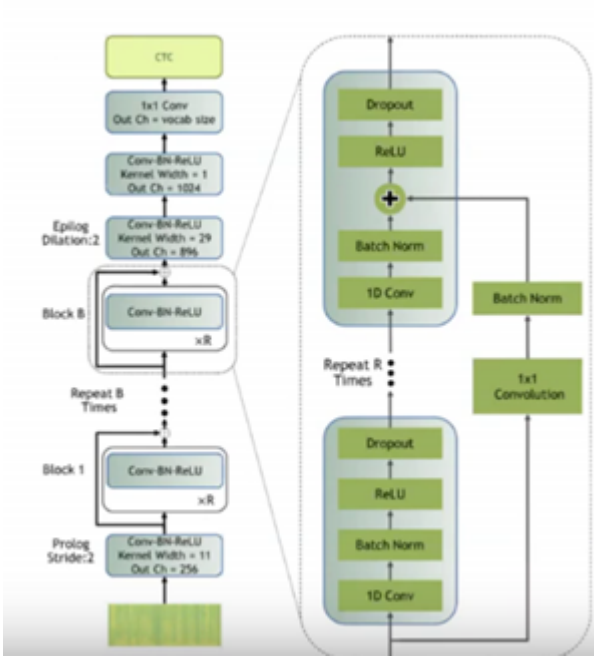
* Тишина

Удаляем эпсилон-токены и склеиваем предложение

Как считать и пробрасывать градиенты

- – Импортировать из торча
  – Динамическое программирование

Jasper

- Сверточная модель
- 

- Из-за skip-connections быстро сходится
- Функция потерь CTC
- 1-D convolutions
  – Одномерный сигнал с большим количеством каналов
  – Канал = частота
  – Размерность 1 = время

Аугментации

- Данных не очень много
- Помогают с нехваткой данных и улучшают робастность
- Либо по времени, либо по частотам вырезаем кусок спектрограммы, заменяя нулевыми амплитудами

Синтезация голоса

Пайплайн TTS

- Hello
- Text Frontend
  - Нормализация текста
    - Убираем специальные символы

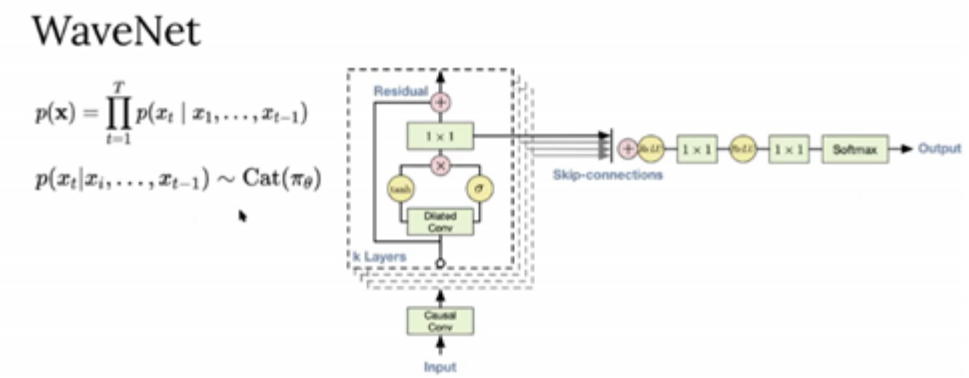
    Транскрибируем фонемы
  - – ∗ Переводит буквы в звукы (написание не совпадает с произношением)
- Mel Synthesis
  - Синтезируем частоты

  WAV Synthesis

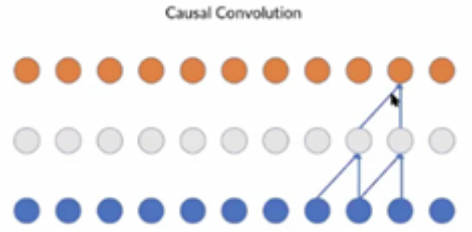
Как измерять качество

- Сложно оценить качество
- Нет правильного ответа
- Субъективность правильности
- Синтез оценивается с помощью краудсорса

Wave-Net



- Causal Conv



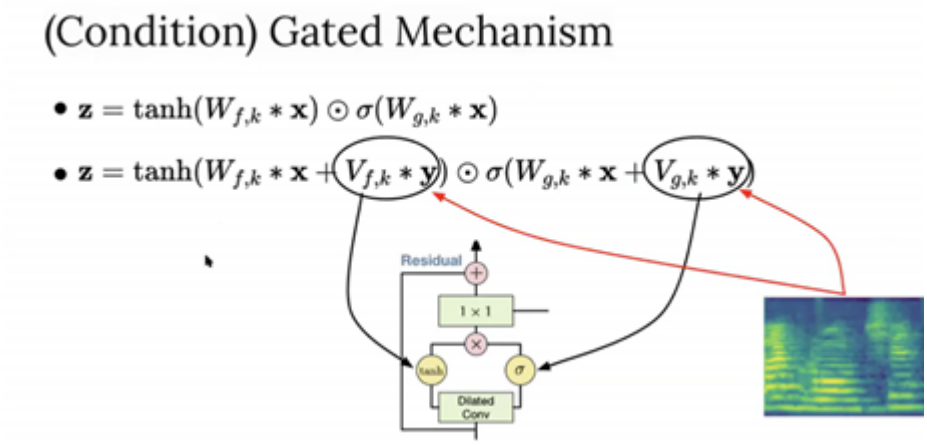
  –

  – Не используем паддинг

  Mu law encoding

- – $f(x_t) = sign(x_t)\frac{ln(1+\mu|x_t|)}{ln(1+\mu)}$

  – В высоком разрешении храним низкие амплитуды, в низком высокие



  –

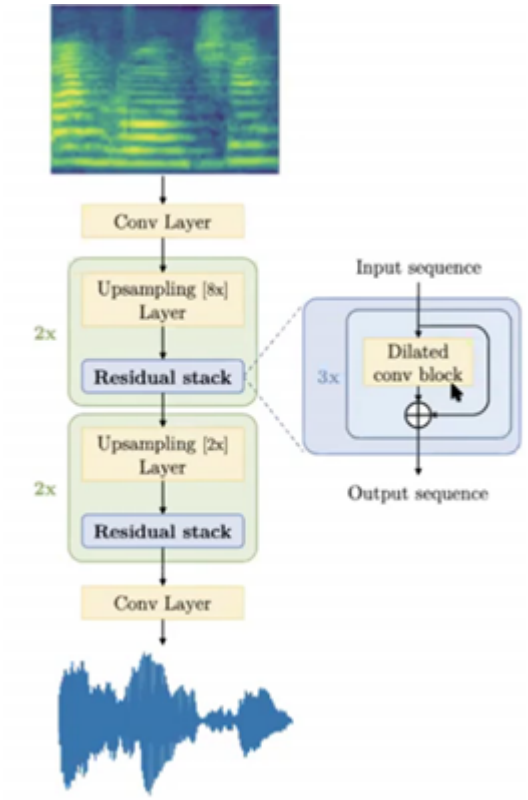
  - $W_{f,k}$, $W_{j,k}$ - некоторые свертки
  - Gated Mechanism - выучивает на какие куски аудио смотреть (как LSTM)
  - $V_{f,\ k} * y$, $V_{g,\ k} * y$- генерируем с помощью сверток, y - уже сгенерированная Мел-спектрограмма
  - x - Все предыдущие предсказанные сэмплы
- Функция потерь
  - Категориальное распределение
  - Нормальное распределение
  - Логистическое распределение

  Модель выдает распределение

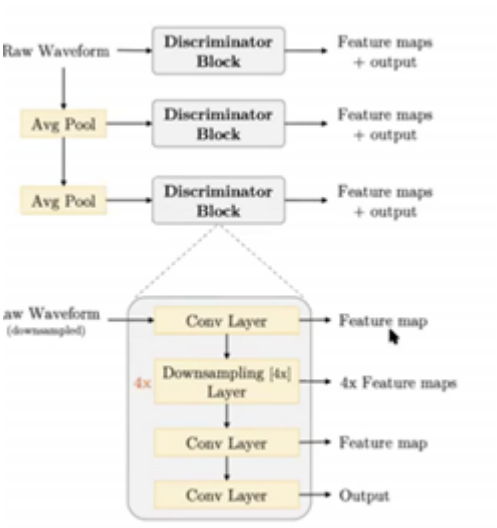
- Skip-connections идут к выводу

Mel-GAN

- Получаем Mel из предыдущей системы
- Генератор:



- Дискриминатор:



- Multiscale discriminator
- Feature Matching

$$\mathcal{L}_{FM}(G, D_k) = \mathbb{E}_{x, s \sim p_{data}} \left[\sum_{i=1}^{T} \frac{1}{N_i} \left\| D_k^{(i)}(x) - D_k^{(i)}(G(s)) \right\|_1 \right]$$

- Hinge Loss

$$\min_{D_k} \mathbb{E}_x[\min(0, 1 - D_k(x))] + \mathbb{E}_{s,z}[\min(0, 1 + D_k(G(s, z)))], \forall k = 1, 2, 3$$

$$\min_G \mathbb{E}_{s,z}\left[\sum_{k=1,2,3} -D_k(G(s, z))\right]$$

-

Рекомендательные системы

Лекция 8: Рекомендательные системы

...

User-based рекомендации

Оцениваем сходство с другими пользователями → Рекомендуем пользователю то, что нравится похожим на него, но он не видел

Проблемы Memory-based

- Долго
- Нет обучения, алгоритм не подкручивается
- Проблема холодного старта → Новые объекты и пользователи = проблема

Item-based рекомендации

- Считаем сходства по тому, как объекты понравились разным пользователям → находим ближайшие

Модели со скрытыми переменными

(Latent factor model)

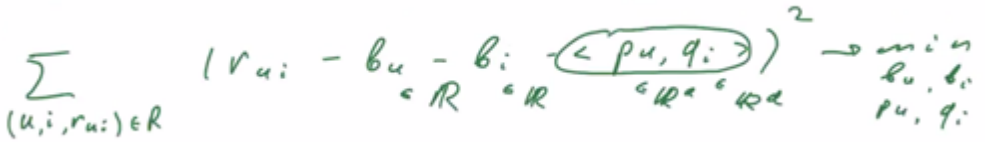
Строим векторы p, q → <p, q> ≈ рейтинг

d - число жанров видео

p_u- как каждому пользователю нравятся жанры

q- распределение жанров в видео

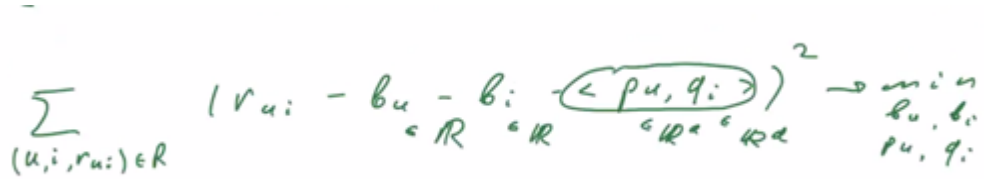
<p, q> отражает совпадение интересов



- $b_u,\ b_i$ - сдвиги
- Можно добавлять рекомендацию
- $P\ =\ (p_1|\ p_2|\ ...\ |p_n)$
- $Q\ = (q_1|q_2|\ ...\ |q_m)$
- $(P^TQ)_{\text{ui}} = <p_u,\ q_i>$
- Приближение матрицы матрицей меньшего ранга

Как обучать?

- Просто стохастическим спуском → выбираем u, i → работает не очень
- ALS (alternating least squares)
  - a)
    - Матрицы P, Q инициализируем
    - Фиксируем матрицу Q



    - q_i теперь фиксированное, а p нужно найти
    - Получается задача обучения линейной модели

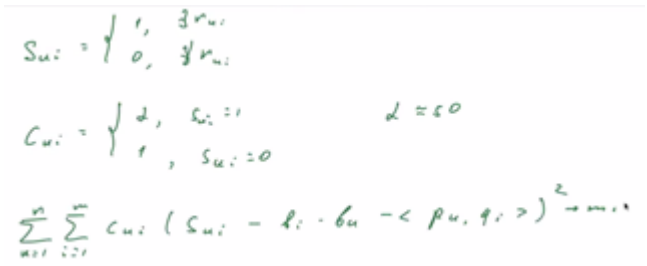
    b)
  - – * Фиксируем матрицу P
    * ...
  - Практический нюанс:
    - Делаем ALS

* Храним только Q
* Приходит новый пользователь
* Делаем один шаг ALS при фиксированной матрице Q
* Делаем <p, q>

Модификация для учета неявной информации

- * Ситуация, когда рейтинги очень специфичные
  * Для (u, i) мы либо знаем, что взаимодействие было, либо не знаем ничего
    · Не знаем понравилось ли пользователю

    IALS



* ·
  · Все объекты, с которыми пользователь взаимодействовал = положительный пример
  · Пользователь не взаимодействовал → Ставим маленький вес при помощи Cui

Лекция 9: Рекомендательные системы 2

Метрики качества рекомендаций

1. Насколько рекомендации подходят
   a. На чем измерять качество
      i. Оффлайн - измеряем на исторических данных
         1. Режем по событиям юзера
         2. Режем по времени

      Онлайн - A/B тестирование

ii. 1. Берем две группы пользователей, рандомно, чтобы они не отличались

2. В одной группе делаем рекомендации старым методом, а в другой новым методом

3. Сравниваем метрики

Регрессия

b. i. MSE, MAE, ...

c. Классификация

i. F-мера, AUC-ROC, ...

Качество ранжирования

d. i. Система выдает ранжированный список

ii. Показываем пользователю top-k айтемов

iii. $R_u(K)$ - top-k рекомендаций

iv. L_u- айтемы, которые пользователю понравились

v. hitrate@k = $[R_u(K) \cap L_u \neq pustomu\ mnozhestvu]$

vi. precision@k = $|R_u(K) \cap L_u|$ / K- не учитывает ранжирование

vii. recall@k = $|R_u(K) \cap L_u|$ / $|L_u|$

viii. DCG

1. $a_{\text{ui}} = a(u,\ i)$

2. Сортируем айтемы по невозрастанию a_{ui}

3. r_{ui1}, ..., r_{uip}- истинные рейтинги

4. DCG@k = $g(r_{\text{uip}}) \times d(p)$

5. d - штраф за позицию

6. g(r) = $2^r - 1$ или g(r) = r

7. d(p) = $\frac{1}{log(p\ +\ 1)}$

nDCG@k - нормализованный

ix. 1. DCG(u) / maxDCG(u)

Другие метрики

2. a. Новизна (novelty)
   i. Число айтемов, которые раньше не рекомендовались
   ii. Опросы (никто не отвечает)

   Разнообразие (diversity)

   b. i. Измеряем попарное расстояние между эмбеддингами айтемов
   ii. Смотреть на дисперсию метаданных
   iii. Предсказания модели для каждого айтема независимы, но в реальности это не так → Разнообразие повышает пользовательские метрики

   c. Serendipity - умение рекомендовать очень редкие айтемы, которые понравятся пользователю
   i. b - новая
   ii. B - мн-во книг, которые пользователь оценил
   iii. C_{Bw}- число книг автора w в множестве B
   iv. S_B- максимальное число книг одного автора в B
   v. $d(b,\ B)\ =\ \frac{1\ +\ C_B - C_{B,w}}{1\ +\ C_B}$

   Что с этим всем делать?

   d. i. Подход 1
      1. Есть бизнес-метрика M, есть остальные метрики f, . . . , f
      2. Ищем веса при метриках, чтобы они приближали M

      Подход 2 (Ухудшающие эксперименты)

   ii. 1. Рандом, предлагать только популярное, . . .
      2. Подбираем веса так, чтобы во всех ухудшающих экспериментах взвешенная сумма уменьшалась

О разном

- Как делать отбор кандидатов

- Сократить всю базу до более маленькой выборки
- Простые методы
  - Те же жанры, исполнители, ...
  - Самое популярное сейчас

  На основе матричных разложений
- – Пользователь-исполнитель
- На основе сходства
  - Уже есть эмбеддинги для пользователей и айтемов

Холодный старт

- Новый пользователь
  - Самое популярное?
  - Опрос об интересах

  Новый айтем
- – Показать фанатам этого же исполнителя
  - Exploration - случайным пользователям показывать

- Контентные рекомендации
  - Можно делать факторы из эмбеддингов
  - Факторы:
    - Расстояние между контентным эмбеддингом этого айтемы и айтемов, которые нравились до этого
    - DSSM
      - Deep semantic similarity model
      - Сеть с двумя половинами (user / item)
      - Item
      - Строим эмбеддинг по контенту
      - User
      - Строим эмбеддинг по контенту всех айтемов, которые раньше понравились пользователю

- Выучиваем эмбеддинги так, чтобы эмбеддинги были близки для близких объектов → Триплетный лосс

Нейросетевая коллаборативная фильтрация (NCF)

- Есть эмбеддинг пользователя, есть айтема
- Конкатенируем
- Сверху полносвязные слои
- Результаты лучше, чем обычное матричное разложение (LFM)
- Steffen Rendle - критиковал эту технологию → Нейросети очень сложно аппроксимировать просто скалярное произведение

Продуктовая аналитика

Лекция 10: A/B-тестирование

Двойное слепое рандомизированное плацебо-контролируемое исследование

- Разделение на две случайные группы
- Пациент не знает и врач не знает

Тройное слепое ...

- Исследователи должны тоже не знать

Для чего нужно?

- Оценка полезности нового явления
- Исследование зависимостей
  - Ухудшающие эксперименты → подбор новых метрик

Мониторинги

- Вечный обратный эксперимент - возвращаем части пользователей то как было раньше
- Построение целей и KPI

Из чего состоит эксперимент

- Разбиение пользователей

  - Может быть сложно получить случайные группы, в программировании рандомизация работает неплохо
  - Как разбивать новых пользователей?
  - Несколько экспериментов одновременно? → по 5% можно проводить 20 экспериментов за раз → Усложняет задачу разбиения
  - Как можно детектировать качество разбиения?
  - Виды
    * По пользователям - cookies
    * По визитам
    * По действиям

Конфигурация эксперимента

- – Длительность - trade-off
  - Размер выборок
  - Срез
- Метрики
- Интерпретация результата
- Корректность эксперимента

Виды экспериментов

- Одномерный / Многомерный
- Прямой / Обратный
- Временный / Вечный
- AA / AB

Конфигурация

- Размер выборок
  - Trade-off - при больших выборках можно продержать пользователей на плохом продукте + невозможность тестировать большое количество экспериментов vs Состоятельность эксперимента

  - При большой выборке можно использовать нормальные распределения + доверительный интервал меньше
  - Чтобы доверительный интервал сократился в 2 раза надо в 4 раза увеличить размер выборки

Длительность

- – Нужно учитывать сезонность
- Срез

Метрика

- Чувствительность
- Шум
  - Большая дисперсия на среднее
  - Работает только на большом количестве пользователей

Интерпретация

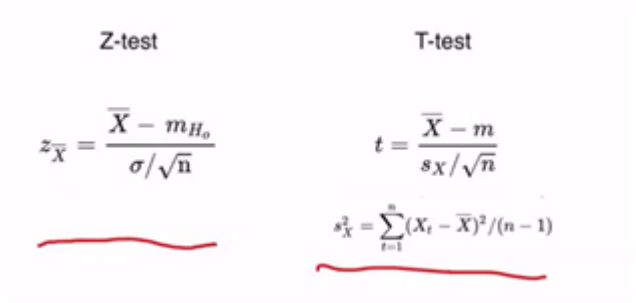
- Иерархия
  - Связь с бизнес показателями
  - Интерпретируемость
  - Можно найти прокси-метрики для больших метрик и доказать влияние

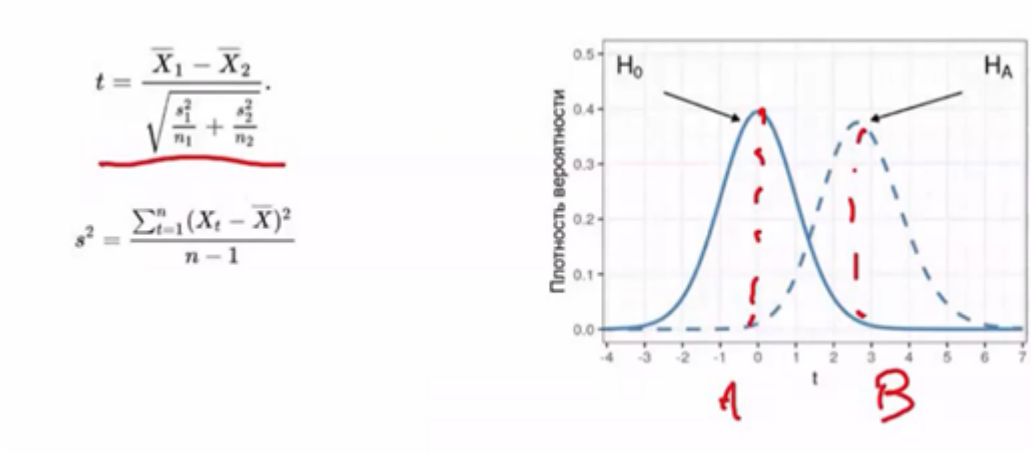
Статистические тесты

- T-test
- Mann-Whitney

Лекция 11: A/B тестирование (2)

- Статистический критерий - правило, которое позволяет делать вывод о том, стоит отвергать гипотезу или нет
- P-value - вероятность ошибки, при условии, что нулевая гипотеза верна
- Доверительный интервал → Если один лежит правее или левее другого, можно сделать вывод о значимости изменения

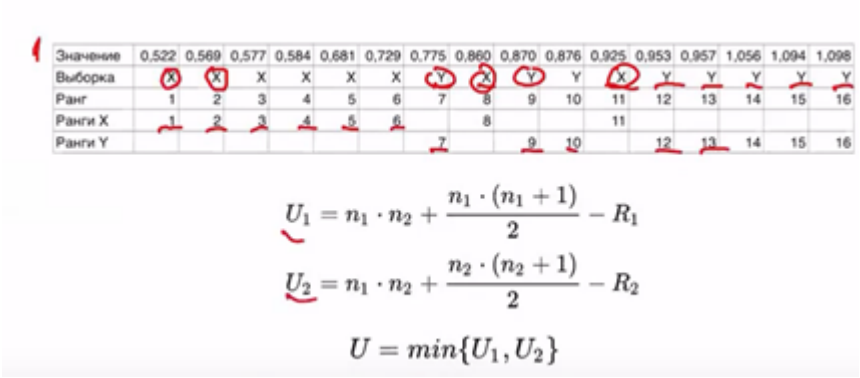
- **Z-тест / T-тест**
  - 

  - Для гипотезы о средних
  - Можем понять доверительные интервалы из этих тестов
  - 

  - Нужно знать, что распределение среднего близко к нормальному

Критерий Mann-Whitney

- Берем массив и сортируем
  - Смотрим какой выборке принадлежит каждое значение
  - Ранговый критерий
    - Выписываем ранги X и Y



Значение	0,522	0,569	0,577	0,584	0,681	0,729	0,775	0,860	0,870	0,876	0,925	0,953	0,957	1,056	1,094	1,098
Выборка	X	X	X	X	X	X	Y	X	Y	Y	X	Y	Y	Y	Y	Y
Ранг	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Ранги X	1	2	3	4	5	6		8			11					
Ранги Y							7		9	10		12	13	14	15	16

$$U_1 = n_1 \cdot n_2 + \frac{n_1 \cdot (n_1 + 1)}{2} - R_1$$

$$U_2 = n_1 \cdot n_2 + \frac{n_2 \cdot (n_2 + 1)}{2} - R_2$$

$$U = min\{U_1, U_2\}$$

  - *
  - * Робастный способ + не зависит от распределения

Примеры

- Эксперимент 1
  - Ухудшение метрики через время (пользователи привыкли?)
    - * Может быть просто сезонность - продлить эксперимент
    - * Взять срез для пользователей, которые пользовались с первого дня и посмотреть метрику для них
    - * Подождать 21 день и сделать обратный эксперимент
    - * Убедится, что эксперимент работает корректно

Эксперимент 2

- – По первым 4м дням видим улучшение метрик
    - * Сезонность
    - * Проверяем метрику каждый час = фактически множественная проверка гипотез

Эксперимент 3

- – По первым 4м дням ухудшение метрик
  - Поменять порог в процессе эксперимента?
    - * Нужно продлить эксперимент + переразбить пользователей

Эксперимент 4

- – Метрика растет, значимость всего 0.9
  - Можно ли считать, что метрика растет?

    - Выбрать более мощный критерий
    - Продлить эксперимент
    - Взять менее шумные прокси-метрики

Аналитика ML-продукта

- Рекомендации
  - Сеть
    - DSSM - двухголовая нейросеть, которая выводит вектора для товаров
    - Ищем соседей с помощью KNN для юзера
    - Делаем фичи
    - Ранжируем по фичам

    Иерархия метрик
  - Ранжируем метрики с помощью модели, оценивающей вес некоторой наиболее важной бизнес метрики

  Применение A/B тестирования
- – Эксперимент перед внедрением новой формулы
  - Вечный эксперимент с отклонением
  - Вечный эксперимент со старой формулой
  - Эксперимент со случайными рекомендациями
  - Ухудшающие эксперименты

Временные ряды

Модели экспоненциального сглаживания

- **Holt, Winters, 1957**
  - Алгоритм прогнозирования, а не модель в статистическом смысле этого слова
  - $\widehat{y}_1 = y_1,\ \widehat{y}_{t\ +\ 1} =\ \alpha y_t + (1\ -\ \alpha)\widehat{y}_{t\ -\ 1}$
  - Строить предиктивные интервалы нельзя

  - Неплохо прогнозирует месячные / сезонные данные
  - Альфа подбирается минимизацией ошибок прогнозов
- **Rob Hyndman, 2002**
  - Формулы прогнозирования для моделей экспоненциального сглаживания следуют из статистической модели
  - Добавляются возможности построения интервалов и модифицирования модель
  - Классический подход - модель оценивается с помощью ММП
  - STAN - вероятностный язык программирования для байесовского вывода
    * Пользователь описывает модель, описывает предпосылки на неизвестные параметры
- **Slavek Smyl, 2015**
  - Оценка более сложных моделей с помощью STAN
- **Sean Taylor, 2017**
  - Prophet
  - Модификация экспоненциального сглаживания
- **Orbit, 2020**
  - Модификация
- **ETS - Error, Trend, Seasonality**
  - Каждая из компонент может отсутствовать (N), может входить аддитивно (A), может входить мультипликативно (H)
  - Выразить текущие показатели через прошлые
  - Нужны тренд и сезонность
  - Наклон линии тренда
$$b_t = b_{t-1}$$
  - Сезонность
$$s_t = s_{t-12}$$

– Уровень очищенный от сезонности

$$\ell_t = \ell_{t-1} + b_{t-1}$$

– Наблюдаемый показатель y_t

$$y_t = \ell_{t-1} + b_{t-1} + s_{t-12}$$

– *Существуют начальные условия*

$$b_0, \ell_0, s_0, s_{-1}, s_{-2}, s_{-11},$$

– *Условие идентификации*

$$s_{-11} = 0$$

- **ETS с учетом ошибки**
  – Добавляется ошибка

$$u_t \sim \mathcal{N}(0, \sigma^2)$$

  – Наклон линии тренда

$$b_t = b_{t\ -\ 1} + \beta u_t$$

  – Сезонность

$$s_t = s_{t-12} + \gamma u_t$$

  – Уровень очищенный от сезонности

$$\ell_t = \ell_{t-1} + b_{t-1} + \alpha u_t$$

  – Наблюдаемый показатель y_t

$$y_t = \ell_{t-1} + b_{t-1} + s_{t-12} + u_t$$

  – Оцениваются параметры:

$$sigma^2, \alpha, \beta, \gamma, \ell_0, s_0, s_{-1}, s_{-2}, s_{-3}, \ldots$$

  – y разбивается на часть, которая может быть предсказуема и часть со случайными ошибками

Теоретический байесовский подход

Три типа величин:

1. Вообще не наблюдаемые случайные величины - параметры модели
2. Наблюдаемые при оценивании модели - прошлые y
3. Наблюдаемые после оценивания модели - будущие y

На вход:

1. Изначальное априорное мнение о параметре, $f(\theta)$
2. Модель для данных, функция правдоподобия, $f(y_1, \dots, y_T|\theta)$

Применяем формулу условной вероятности:

$$f(\theta \mid y_t, \dots, y_T) = \frac{f(\theta) f(y_1, \dots, y_T \mid \theta)}{f(y_1, \dots, y_T)}$$

Получаем апостериорное мнение о параметре:

$$f(\theta \mid y_1, \dots, y_T)$$

С помощью θ получаем апостериорное мнение о будущих величинах:

$$f(y_{T+h} \mid y_1, \dots, y_T)$$

Цепи Маркова в байесовском подходе

Описываем:

1. Изначальное априорное мнение о параметре $f(\theta)$
2. Модель для данных, функция правдоподобия $f(y_1, \dots, y_T \mid \theta)$

На выходе черный ящик выдает последовательность из θ

Причем:

$$\theta^k \xleftarrow{distr} f(\theta \mid y)$$

Реккурентные уравнения

1. Хотим выразить все ненаблюдаемые величины через параметры модели и наблюдения y_t

   (a) Выражаем u_t

   $$u_t = y_t - \ell_{t-1} - \theta b_{t-1} - s_{t-12} - r_t$$

   (b) Записываем уравнения на b_t, s_t, ℓ_t в виде средневзвешенного всех значений

   $$b_t = (1-\beta)\theta b_{t-1} + \beta(y_t + \ell_{t-1} - s_{t-12} - r_t)$$

   $$s_t = (1-\gamma)s_{t-12} + \gamma(y_t - \ell_{t-1} - \theta b_{t-1} - r_t)$$

   $$\ell_t = (1-\alpha)(\ell_{t-1} + r_t + \theta b_{t-1}) + \alpha(y_t - s_{t-12})$$

Код

```
!pip install arviz #Visualize
!pip install pystan #STAN
!pip install sktime

import pystan
import arviz as vz
import pandas as pd
import numpy as np

from sktime.datasets import load_airline
from sktime.utils.plotting import plot_series
from sktime.forecasting.exp_smoothing import

  ExponentialSmoothing

y = load_airline()
plotseries(y)

ln_y = np.log(y)
plotseries(ln_y)

model_code = """
data {
  int<lower=0> n; //number of observations
  vector[n] y; // time series data
```

```
  vector[n] x; //predictor
}

parameters {
  // equation parameters - aprior knowledge
  real<lower=0, upper=1> alpha;
  real<lower=0, upper=1> beta;
  real<lower=0, upper=1> gamma;
  real<lower=0, upper=1> theta;
  real<lower=0, upper=1> alpha;
  real k;
  real<lower=0> sigma;

  // initial values
  real linit;
  real binit;
  vector[12] sinit;
}

transformed parameters {
  vector[n + 1] l;
  vector[n + 1] b;
  vector[n + 12] s;
  vector[n] r;
  vector[n] y_hat; // E(y_t | F_{t - 1})

  // initial observations
  l[1] = linit;
  b[1] = binit;
  for (t in 1:12) {
    s[t] = sinit[t];
  }


  //update equations
  for (t in 1:n) {

    r[t] = k * x[t];
    b[t] = (1 - beta) * theta * b[t - 1] + beta *
      (y[t] - l[t - 1] - s[t - 12] - r[t]);
    s[t] = s[t - 12] + gamma *
```

```
      (y[t] - l[t - 1] + theta b[t - 1] - s[t - 12] - r[t]);
    l[t] = (1 - alpha)(l[t - 1] + r[t] + theta * b[t - 1]) +
      alpha * (y[t] - s[t - 12]);
    yhat[t] = l[t - 1] + theta * b[t - 1] + s[t - 12] + r[t];
  }
}

model {
  // prior
  alpha ~ uniform(0, 1);
  beta ~ uniform(0, 1);
  gamma ~ uniform(0, 1);
  theta ~ theta(0, 1);
  sigma ~ normal(0, 10) T[0]; // T means truncated
  k ~ normal(0, 10);

  // prior for initial values
  linit ~ normal(0, 10);
  binit ~ normal(0, 10);
  sinit ~ normal(0, 10);
}"""

model = pystan.StanModel(model_code = model_code)

x = np.concatenate([np.zeros(24), np.ones(120)])

air_data = {'n': 144, 'y': ln_y_values, 'x': x}
post = model.sampling(air_data, chains = 2,
  iter = 5000, warmup = 1000)

post
post['k'] # look at parameters

az.plot_trace(post['alpha']) # check convergence

ln_y_145_hat = post['l[145]'] + post['theta']
  + post['b[145]'] + post['k']
  + 1 * np.random.normal(loc = 0, scale = post['sigma'])
np.median(ln_y_145_hat) #prediction
np.quantile(ln_y_145_hat, q = [0.025, 0.975]) #interval prediction
```

```
## ETS(AAA)
ets_aaa = ExponentialSmoothing(trend = 'add',
                               seasonal = 'add',
                               sp = 12)
ets_aaa.fit(ln_y)
ets_aaa.predict(1)
```

MLOps

Bash

1. Терминал - графическая оболочка
2. Bash - стандартный язык Линукса
3. Модель исполнения Bash
   (a) Для него каждая программа = черный ящик
   (b) Запускает черный ящик и и вводит аргументы командной строки
   (c) У каждой программы есть два потока вывода - *вывод и ошибки*
   (d) stdin → stdout, stderr
   (e) stdin, stdout, stderr лежат в /dev
4. Операторы
   (a) > - меняет файл вывода с stdout на произвольный
   (b) echo - повторяет текст в stdout
   (c) cat - склеивает содержимое файлов вместе, выводим в stdout
   (d) >> - как > но не затирает файл, а добавляет в конец
   (e) < - заменяем stdin на файл
   (f) python3 -c - можно вводить команду в виде строки, а не .py
   (g) << - ввод данных из терминала, а не из файла (надо указывать END, END)
   (h) <<< - ввод одной строки, без маркеров END

(i) `<()` - вывод команды в скобках будет использоваться как вход для следующей команды

(j) Комбинируя `<` и `<()` можно перенаправлять вывод одной программы в другую

(k) yes - команда бесконечно генерирующая y

(l) mkdir - создает директории

(m) touch - создает пустые файлы

(n) diff - сравнивает файлы

(o) `|` - вывод левой программы попадает в ввод правой

(p) wc - word count, с модификатором -l считает строки

(q) head - первые 10 строк файла

(r) `$()` - перенаправляем вывод не в файл, а прямо в bash

5. Полезные программы

   (a) head - прочитать начало файла (-n можно указать количество строк, -c можно указать количество байт)

   (b) tail - последние строки файла (-n +t - пропустить t строк сверху)

   (c) sort - сортировка (по умолчанию строки, -n - числа, можно сортировать по части строки -k - выбор колонки, -r - в обратном порядке)

   (d) shuf - перемешивает входящие данные

   (e) uniq - уникальные значения в файле (ожидает sort, -c - подсчитывает сколько было одинаковых элементов)

   (f) cut - разбивает строки по разделителю (-f берет отдельные колонки)

   (g) uname -a - информация о системе

   (h) grep - позволяет фильтровать строки по регулярному выражению

   (i) sed - обработка данных в специальном формате

   (j) jq - для обработки формата JSON

(k) tar - архивирование и разархивирование

(l) wget, curl - выгрузка данных из интернета

Docker

1. В питоне можно поднимать сервер

   `nohup python3 -m http.server --bind 0.0.0.0 9091 & echo $1 > run.pid`

   (a) `nohup` - запускает в бэкграунд

   (b) 0.0.0.0 = запуск на всех доступных айпи

   (c) 9091 - порт

   (d) В файле run.pid появляется идентификатор процесса

2. `expose_port_on_colab` - создаем страничку на ngrok
3. ngrok - открывает сервер для интернета
4. `joblib - dump, load` - позволяет сохранять и грузить любые питоновские объекты
5. Flask - строим веб сервера
6. С помощью `requests` можно отправлять post-запросы на сервер
7. print(r.json) возвращается результат запроса
8. `torchserve, torch-model-archiver, captum` - позволяет напрямую из торча выводить модели на сервер

   (a) `mkdir model_store`

   (b) `создаем index_to_name.json` - чтобы переводить классы в названия

   (c) `torch-model-archiver` - архивирует модель в нужную папку

      i. –model-name - имя модели

      ii. –version - версия

      iii. –model-file - файл модели .py

      iv. –serialized-file - веса

      v. –export-model-path - в какую папку класть модель

      vi. –extra-files - . . .

vii. –handler - функция внутри model.py которую надо вызывать

(d) `torchserve --start --ncm --model-store model-store --models --ts-config`

i. tsconfig.prop - добавляем порты для разных функций

9. **Docker**

(a) Позволяет запускать процессы изолированно от OS

(b) Вместо VM, но легче

(c) Container - экземпляр процесса

(d) Image - запускает много процессов под контейнер

(e) Docker имеет способы прокидывать директории с исходной машины в контейнер

i. Контейнер будет их воспринимать как внутренние, а не внешние

(f) Dockerfile - образ

(g) `docker build -t name:latest -f Dockerfile .`

(h) `docker run -v /path_from:/path_to command` - пробрасываем папку, выполняем команду

(i) `docker run -p PORTS name:latest` - пробрасывает порты