Лаборатория адаптивных методов обработки данных

Учебный курс

«Машинное обучение в физике»

Занятие №1 (лекция).

Введение в машинное обучение. Специфика задач обработки данных в физике.

Авторы курса:

С.А. Доленко, И.М. Гаджиев, А.О. Ефиторов, И.В. Исаев, В.Р. Широкий

Предмет курса: что такое машинное обучение

Машинное обучение (Machine Learning) = обучение машины

- Построение модели исследуемого объекта (моделирование)
- Модель строится на основании примеров, описывающих объект
  - Обучение на примерах, анализ данных
- Адаптивные методы анализа данных (нейронные сети и не только)
  - Адаптивный = самоприспосабливающийся (к данным)
  - Другими словами, это методы, управляемые данными (data-driven)
- Общая методология машинного обучения (МО)
  - Что, зачем и в каком порядке надо делать
- Акцент на специфику задач обработки данных в физике
  - Примеры применения методов МО в физике (много общего с другими естественными науками, но есть и отличия)

Методы описания состояния объекта в физике

- **Содержательное физическое моделирование**
  - Теоретическое описание объекта с учётом известных физических закономерностей
  - Получение и использование аналитического выражения для интересующей исследователя физической величины
- **Численное моделирование**
  - Описание объекта физически содержательной системой уравнений
  - Численное решение системы с получением значения интересующей исследователя физической величины
- **Аппроксимационное моделирование**
  - Получение информации об объекте в виде массива примеров, описывающих его состояние в различных ситуациях
  - Аппроксимация описывающих объект закономерностей с получением оценки значения интересующей исследователя величины

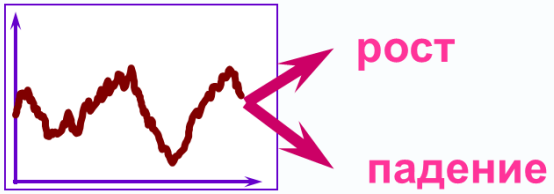
Аппроксимационное моделирование – построение функционального отображения

- **Получение информации об объекте в виде массива примеров, описывающих его состояние в различных ситуациях**
  - Для каждого примера должно быть известно желаемое (истинное) значение интересующей физической величины
- **Выбор семейства аппроксимирующих функций, желательно из физических соображений**, например:
  - релаксационный процесс с несколькими постоянными времени
  - разложение спектра на составляющие спектральные контуры
- **Если это невозможно – выбор универсального семейства аппроксимирующих функций**, например:
  - разложение сигнала во временном представлении по базису Фурье
  - вейвлет-разложение сигнала
  - нейросетевая аппроксимация

(нейронная сеть – универсальный аппроксиматор)

Место аппроксимационных методов

- **Плохо формализуемые задачи**, например:
  - Распознавание текста, речи, изображений
  - Медицинская, техническая диагностика
  - Прогнозирование значений или наступления событий



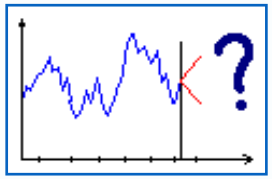
- **Аппроксимационная модель = разложение искомого отображения**
  - Обучение на примерах = построение «чёрного ящика»
  - Если есть возможность – ящик лучше покрасить,

    т.е. использовать имеющуюся априорную информацию

Типология задач, решаемых методами МО

- **Регрессия** (оценка значения величины)
  - Количественная оценка
  - Прогнозирование временных рядов
  - Аппроксимация зависимостей

$Y = 378.14 \pm 0.21$



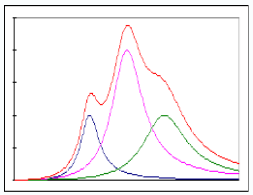
$Y = F(X_1, X_2, \ldots, X_n)$

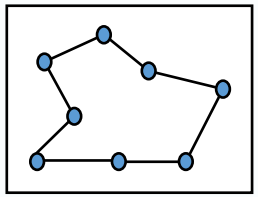
- **Классификация** (распознавание образов)
  - Бинарная
  - Многоклассовая
  - Многометочная

А Б В Г Д

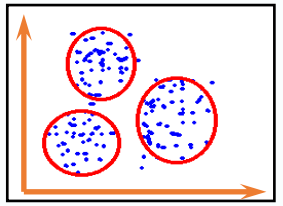
1 2 3 4 5 6

- **Оптимизация** (поиск оптимальных значений / комбинаций)
  - Поиск оптимальных значений
  - Комбинаторная оптимизация





- **Кластеризация** (разбиение данных на группы)



Границы между типами задач могут быть размыты,

одну и ту же задачу можно ставить как задачи разных типов

Виды задач классификации

- Бинарная
  - Обнаружение аномалии *(установка неисправна)*
  - Прогнозирование факта наступления события *(распад произойдёт)*
  - Диагностика принадлежности объекта множеству *(вы беременны!)*
- Многоклассовая
  - Распознавание символов *(это Ъ)*
  - Распознавание вида объекта *(это вода)*
  - Выбор одного из множеств, которым может принадлежать объект *(агрегатное состояние – жидкость)*
- Многометочная
  - Рубрикация *(статья по физике и методам МО)*
  - Диагностика принадлежности объекта одновременно нескольким множествам *(в растворе присутствуют ионы меди, натрия и хлора)* *(животное умеет плавать, летать и имеет 2 ноги)*

Уровни взаимодействия интеллекта с окружающим миром

1. Персептивный *(лат. perceptio – восприятие)*
   - Восприятие окружающего мира и получение информации о нем
   - Осуществление простейших ответных действий – реакция на внешние раздражители
2. Когнитивный *(лат. cognitio – познание)*
   - Познание окружающего мира
   - Установление закономерностей
   - Обучение действиям в ситуациях, схожих с известными
3. Креативный *(лат. creatio – творчество)*
   - Творческое осмысление и освоение окружающего мира
   - Изобретение нового
   - Умение адекватно действовать в незнакомых ситуациях

На повестке дня переход искусственных систем на креативный уровень.

Мостики через пропасть уже перекинуты, и они всё шире…

Добыча данных, машинное обучение и искусственный интеллект

- Добыча данных (Data Mining, DM)
  - Сбор данных, их очистка и первичная обработка
  - Простые виды анализа данных без применения обучаемых алгоритмов
- Машинное обучение (МО) (Machine Learning, ML)
  - Применение адаптивных методов, управляемых данными
  - Установление закономерностей и построение моделей
  - Применение полученных моделей в ситуациях, схожих с известными
- Искусственный интеллект (ИИ) (Artificial Intelligence, AI)
  - Комплексное применение различных методов, основанных на разных данных, разных подходах, различных областях знания
  - Получение универсальных комплексных моделей, пригодных для применения в широком круге разных ситуаций, в том числе новых

Слабый ИИ (Narrow AI) – эффективно решает конкретный круг задач

Сильный ИИ (General AI, AGI) – решает разные задачи

Добыча данных, машинное обучение и искусственный интеллект

- Добыча данных (Data Mining, DM)
  - Сбор данных, их очистка и первичная обработка
  - Простые виды анализа данных без применения обучаемых алгоритмов
- Машинное обучение (МО) (Machine Learning, ML)
  - Применение статистических методов, управляемых данными
  - Установление закономерностей и построение моделей
  - Применение полученных моделей в ситуациях, схожих с известными
- Искусственный интеллект (ИИ) (Artificial Intelligence, AI)
  - Комплексное применение различных методов, основанных на различных областях знания
  - Получение универсальных комплексных моделей, пригодных для применения в широком круге разнообразных задач

Персептивный уровень

Когнитивный уровень

Креативный уровень

Соответствие – условное!!!

Слабый ИИ (Narrow AI) – эффективно решает конкретный круг задач

Сильный ИИ (General AI, AGI) – решает разные задачи

Три волны интереса к нейронным сетям

Краткая историческая справка

- Первая волна (1957 – 1969)
  - Теорема Колмогорова о представлении функции *n* переменных (1957)
  - Персептрон Розенблатта (1958)
  - Минский, Пейперт, «Персептроны» (1969)
- Вторая волна (1986 – середина 90-х)
  - Алгоритм обратного распространения ошибки (1986)
  - Теорема об универсальной аппроксимации персептроном (1989)
- Третья волна (середина 2000-х – наши дни)
  - Глубокие НС
  - Свёрточные НС (с 1990-х!)
  - Рекуррентные сети с долгой краткосрочной памятью (LSTM)
  - Генеративные сети
  - Обучение с подкреплением (с 1980-х!)

Некоторые успехи третьей волны

- Решена задача дикторонезависимого распознавания речи
- Решена задача идентификации (классификации) объектов на изображениях и в видеопотоке
- Решена задача самообучения сложным играм на основе знания лишь правил игры (шахматы, сёги, го)
- Достигнуты существенные успехи в машинном анализе текстов
- Достигнуты существенные успехи в машинном переводе текстов
- Достигнуты существенные успехи в генерации новых данных

Полученные результаты основаны на доступности

- Существенно возросших вычислительных мощностей («закон Мура»)
- Практически неограниченного объёма данных некоторых типов
- Серьёзных инвестиций, связанных с уже достигнутыми успехами

Специфика данных в физике

- Высокая размерность пространства входных признаков
- Сравнительно небольшое количество примеров (исключение – проекты класса “megascience”)
- Противоречивость и неполнота данных – плохая представительность
- Нелинейность аппроксимируемой зависимости, иногда сильная
- Некорректность и/или плохая обусловленность (обратные задачи)
- Мультиколлинеарность (взаимозависимость) признаков
- Дискретность значений данных
- Наличие шумов
- Существует также специфика конкретных типов физических задач

Всё это накладывает существенные ограничения на арсенал доступных методов МО и требует разработки специальных методик их применения к данным физического эксперимента.

Стандартная структура данных для МО

- Прямоугольная матрица

		Входные признаки					Выходные признаки			
	ID_прим	ВхПр1	ВхПр2	ВхПр3	…	ВхПрN	ВыхПр1	ВыхПр2	…	ВыхПрL
Примеры	Прим1	X11	X12	X13	…	X1N	Y11	Y12	…	Y1L
	Прим2	X21	X22	X23	…	X2N	Y21	Y22	…	Y2L
	Прим3	X31	X32	X33	…	X3N	Y31	Y32	…	Y3L
	Прим4	X41	X42	X43	…	X4N	Y41	Y42	…	Y4L
	…	…	…	…	…	…	…	…	…	…
	ПримP	XP1	XP2	XP3	…	XPN	YP1	YP2	…	YPL

- Столбцы – признаки (features) (N входных и L выходных)
- Первая строка может содержать названия признаков
- Строки – примеры (patterns) (P примеров)

  Термины «образцы», “examples”, “samples” имеют побочные смыслы.
- Первый столбец может содержать идентификаторы примеров
- Матрица не должна содержать пропусков!

Общая схема машинного обучения

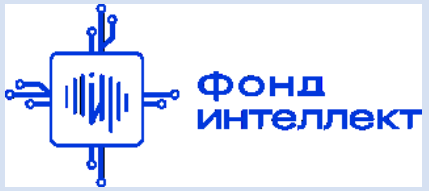
Общая информация о курсе

- Составные части курса:
  - 11 лекций
  - 5 практических занятий, 5 домашних заданий
  - Соревнование по решению сложной задачи реального мира
  - Зачёт (вопрос о возможности внесения в диплом прорабатывается)
- Одно занятие в неделю по вторникам в 17-30 или в 19-00
  плюс два дополнительных занятия, чтобы уложиться в семестр
- Очные занятия с трансляцией в Интернете
- Выкладывание видеозаписей занятий на платформе Teach-In
- Для допуска к зачёту необходимо сдать не менее 4 домашних заданий, не менее 3 из них должно быть зачтено
- 5 победителей соревнования получат зачёт «автоматом»

На зачёте можно пользоваться любыми источниками

Спасибо за внимание!

Лаборатория адаптивных методов обработки данных



Учебный курс

«**Машинное обучение в физике**»

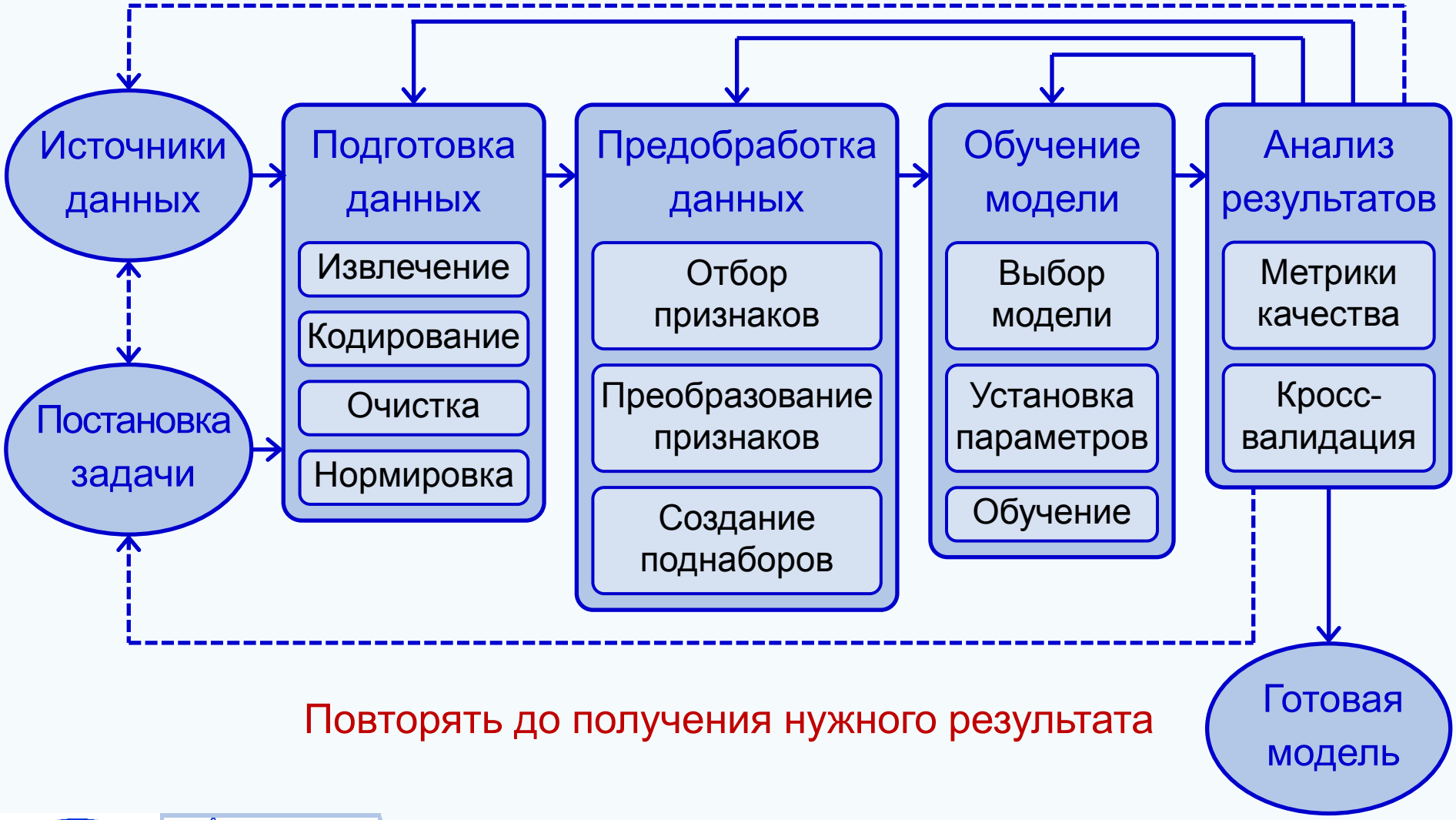
Занятие №3 (лекция).

Подготовка данных. Оценка качества моделей.

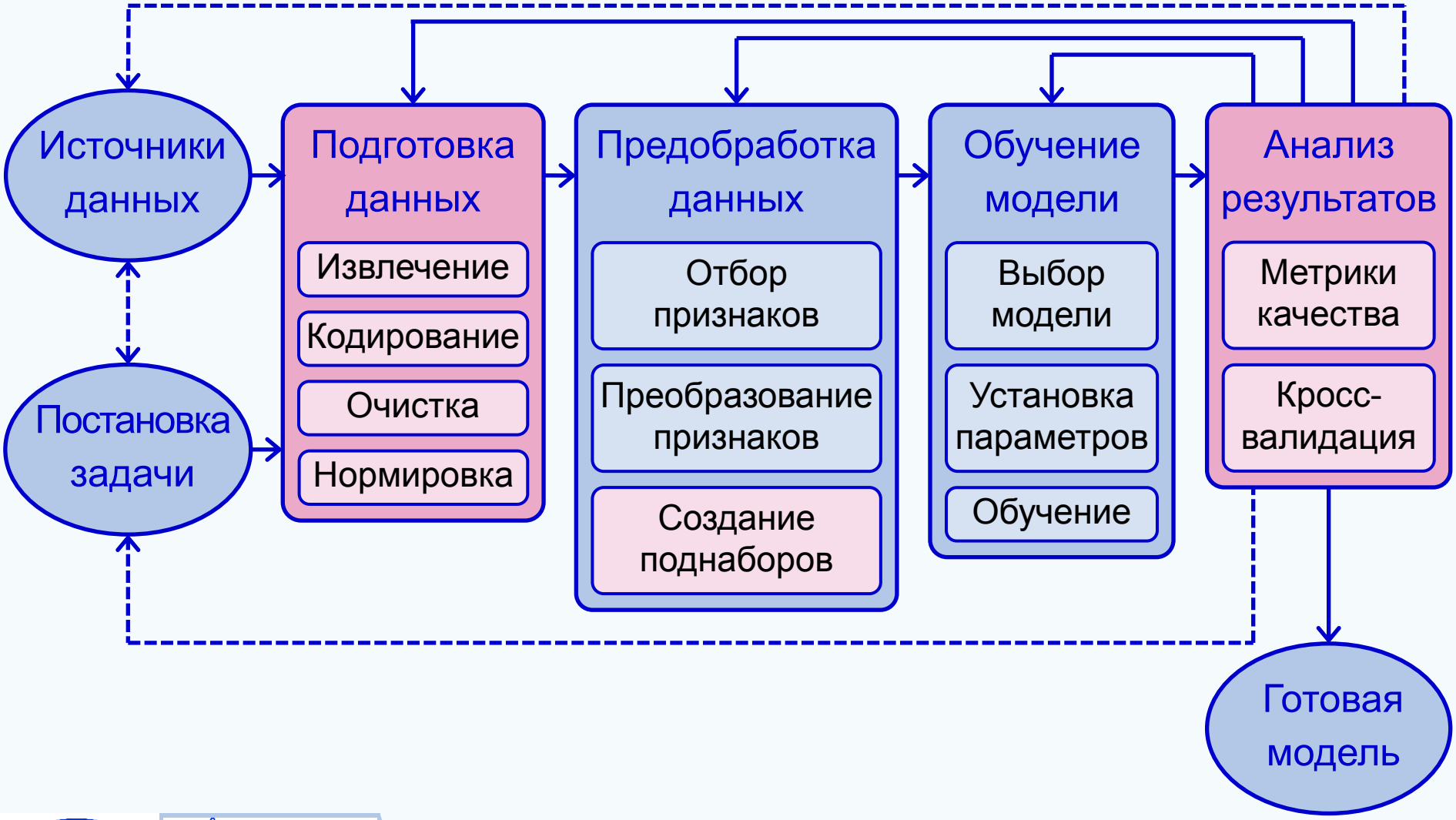
Авторы курса:

С.А. Доленко, И.М. Гаджиев, А.О. Ефиторов, И.В. Исаев, В.Р. Широкий

Общая схема машинного обучения



Содержание занятия



Данные

Определения

Информация (information): знания или сведения,
относящиеся к таким объектам, как факты, события, предметы,
процессы или идеи, включая концепции,
которые в соответствующих контекстах имеют конкретное значение.

Данные (data): интерпретируемое представление информации
в соответствующей форме, **удобной** для передачи, интерпретации и обработки.

ГОСТ Р ИСО 8000-2–2019 “Качество данных. Часть 2. Словарь.”

Данные

Виды данных

- Признаковые описания объектов или матрицы объект-признак
  - Каждый объект описывается набором признаков
- Временные ряды, сигналы
  - Последовательность измерений во времени, которое может представляться числом, вектором, а в общем случае - признаковым описанием в данный момент времени.
- Изображения
- Видео
- Текст
- Данные в специальных форматах
  - графы, файлы, ссылки, веб-страницы и т. д.
- И т. д.

Подготовка данных

Этапы подготовки данных

- Кодирование
  - Перевод всех данных в числовую форму
- Предобработка данных
  - приведение всех входных и выходных данных к форме, удобной для дальнейшей обработки, анализа и пригодной для использования в методах машинного обучения
- Нормировка
  - приведение всех входных и выходных данных к единому масштабу

Подготовка данных

Предобработка данных

Предобработка данных: приведение всех входных и выходных данных к форме, удобной для дальнейшей обработки, анализа и пригодной для использования в методах машинного обучения

Виды признаков:

- Вещественные (в т.ч. временные):
  - Дискретные и непрерывные
  - Интервальные и относительные
- Категориальные:
  - Упорядоченные (ординальные)
  - Неупорядоченные (номинальные)

Подготовка данных

Предобработка данных

Преобразование вещественных признаков:

- Бинаризация
  - Признакам ниже порогового значения присваивается одно значение, а выше - другое.
- Бининг (дискретизация, квантование)
  - Преобразование вещественного признака в порядковый, за счет замены значения из интервала на одно значение
- Агрегация
  - Сумма, среднее, а в общем случае - любая функция над несколькими признаками
- Округление
- Логарифмирование
- Сглаживание и т. д.

Подготовка данных

Предобработка данных

Преобразование категориальных признаков

- Label encoding
  - Каждая категория заменяется на некоторое число
- One-hot encoding
  - Каждая категория заменяется на отдельный бинарный признак, где заданной категории соответствует значение 1, остальным - 0
- Эмбеддинги (Занятие 8)
  - Исходное пространство категорий кодируются в пространство меньшей размерности, при помощи методов машинного обучения

Подготовка данных

Нормировка

Нормировка данных: приведение всех входных и выходных данных к единому масштабу, с целью повышения устойчивости и качества решения

- Методы, требующие нормировку данных:
  - Нейронные сети, анализ главных компонент
  - Методы, основанные на подсчёте расстояний (KNN, K-means)
  - Методы, использующие градиентный спуск
- Методы, не требующие нормировку данных:
  - Методы, основанные на деревьях решений

Подготовка данных

Нормировка

Способы нормировки данных:

- На отрезок (MinMax Scaling):
  - все точки переносятся на заданный отрезок, обычно [0,1]

$$x^* = \frac{x - x_{min}}{x_{max} - x_{min}}$$

- Нормировка по максимуму:
  - не сдвигает/центрирует данные, не убирает разреженность

$$x^* = \frac{x}{Max(x)}$$

- Стандартизация (Standart Scaling, Z-score normalization)
  - Признак центрируется и масштабируется

$$x^* = \frac{x - \bar{x}}{\sigma_x}$$

- L1, L2-норма
  - Каждый элемент вектора делится на его длину

$$x^* = \frac{x}{\|x\|_{1,2}}$$

- Десятичное масштабирование (decimal scaling)
  - перемещение десятичной точки на число разрядов, соответствующее порядку числа

$$x^* = \frac{x}{10^{min\{i:10^i \geq x\}}}$$

https://scikit-learn.org/stable/modules/classes.html#module-sklearn.preprocessing

Подготовка данных

Качество данных

Качество данных (data quality): степень, с которой набор характеристик, присущих данным, отвечает требованиям.

ГОСТ Р ИСО 8000-2–2019 “Качество данных. Часть 2. Словарь.”

Требования, предъявляемые к данным:

- Релевантность – соответствие данных целям и задачам анализа, степень адекватности данных рассматриваемому объекту
- Корректность (точность) - отсутствие выбросов, аномалий, шумов
- Полнота - отсутствие пропусков, достаточное количество примеров, равномерность распределения данных, сбалансированность классов.
- Непротиворечивость, согласованность, безызбыточность, ясность, доступность, актуальность и т. д.

Подготовка данных

Методы оценки качества данных

Мотивация: повышение качества решения

Иначе: “Мусор на входе – мусор на выходе”

Методы оценки качества данных:

- ❑ Релевантность, безызбыточность
  - Отбор и преобразование признаков (занятие 4)
- ❑ Корректность (точность)
  - Поиск выбросов и аномалий
- ❑ Полнота
  - Поиск пропусков в данных
  - Анализ распределений определяемых параметров (для задачи регрессии)
  - Анализ распределения примеров по классам (для задачи классификации)

Подготовка данных

Поиск выбросов и аномалий

- ❑ С помощью здравого смысла или априорных знаний
  - Когда известен нормальный диапазон значений признака
- ❑ Статистические тесты
  - Удаление значений, расположенных вдали от среднего, медианы и т.п.
- ❑ Модельные тесты
  - Построение модели, которая описывает данные
  - Удаление значений, на которых модель сильно ошибается
- ❑ Метрические методы (LOF и др.)
  - Основаны на вычислении расстояний между примерами
- ❑ Кластеризация
  - Удаление маленьких кластеров

https://dyakonov.org/2017/04/19/поиск-аномалий-anomaly-detection/

Подготовка данных

Пропуски в данных

- Оставить
  - Но не все модели могут работать с пропусками
- Анализ природы пропуска
  - В ряде случаев отсутствие информации – тоже информация
  - Добавление признака с меткой пропуска
- Удаление признаков с пропусками,
  если они малоинформативные или пропусков слишком много
- Удаление примеров с пропусками, если примеров достаточно
- Замена на фиксированное или легковычисляемое значение
  - минимум, максимум, среднее, медиана, мода и т.п.
- Восстановление при помощи модели, обученной на других признаках

Подготовка данных

Понятие несбалансированности данных

- Задача классификации
  - Доли объектов разных классов существенно различаются
- Задача регрессии
  - Искаженное / не нормальное распределение значений определяемого параметра
  - Любая задача прогнозирования событий:
    фоновых значений много, значений для событий - мало

Подготовка данных

Способы работы с несбалансированными данными

- Удаление примеров (Undersampling)
  - Случайное (Random Undersampling)
  - С учетом расстояний между примерами (Nearmiss, Tomek links и др.)
  - С учетом ошибок классификатора (Edited nearest neighbors – ENN и др.)
- Дублирование примеров (Oversampling)
  - Случайное (Random Oversampling): с возвратами / без возвратов
  - По заданным условиям: балансированное, стратифицированное и др.
- Генерация искусственных примеров
  - Линейная комбинация примеров (SMOTE, ADASYN и др.)
  - Вариационные автоэнкодеры (занятие 13)
- Комбинирование подходов

https://dyakonov.org/2021/05/27/imbalance/ https://loginom.ru/blog/imbalance-class

Анализ результатов

Методы оценки качества для задачи регрессии

Постановка задачи регрессии: минимизация отклонений между реальными значениями, и значениями, предсказанными моделью.

Оценка качества для задачи регрессии – **анализ отклонений**

Особенности отклонений:

- Выбросы - высокие значения отклонений для некоторых примеров
- Смещение (bias) и разброс (variance)
- Гетероскедастичность - неоднородность отклонений, зависимость величины отклонений от какого-либо фактора

Анализ результатов

Методы оценки качества для задачи регрессии

Методы оценки качества:

- ❑ Визуальный
  - Диаграмма рассеяния
  - Гистограмма ошибок
- ❑ Метрики качества
  - Среднее абсолютное отклонение (CAO, MAE - Mean Absolute Error)
  - Среднеквадратичная ошибка (СКО, RMSE - Root Mean Squared Error)
  - Коэффициент детерминации R^2
  - Коэффициент корреляции r
  - Взвешенная ошибка – где требуется разная цена ошибки в различных диапазонах

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|a(x_i) - y_i|$$

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(a(x_i) - y_i)^2}$$

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(a(x_i) - y_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y})^2}$$

$$r = \frac{\sum_{i=1}^{N}(a(x_i) - \bar{a})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{N}(a(x_i) - \bar{a})^2} \cdot \sqrt{\sum_{i=1}^{N}(y_i - \bar{y})^2}}$$

$$weighted\ MAE = \frac{1}{N}\sum_{i=1}^{N} w_i|a(x_i) - y_i|$$

Анализ результатов

Методы оценки качества для задачи классификации

Постановка задачи классификации: отнесение примеров к одному из заданных классов.

Оценка качества для задачи классификации – **анализ попаданий примеров** в тот или иной класс.

Виды задач классификации:

- Бинарная
  - каждый пример может принадлежать к одному из двух классов
- Многоклассовая
  - каждый пример может принадлежать к одному из нескольких классов
- Многометочная
  - каждый пример может принадлежать одновременно к нескольким классам

Анализ результатов

Методы оценки качества для задачи классификации

Особенности ошибок бинарной классификации:

- Наличие 2-х типов ошибки:
  - Ошибка 1-го рода: ложноположительная, “ложная тревога”, false positive
  - Ошибка 2-го рода: ложноотрицательная, “пропуск цели”, false negative
- Ошибки взаимносимметричны
  - Можно поменять между собой положительный и отрицательный исход
  - За положительный исход обычно берется естественное положение вещей
- Ошибки связаны между собой:
  - Уменьшение ошибки одного типа
    может приводить к увеличению ошибки другого типа
- Для ряда задач цена ошибок 1-го и 2-го рода может быть разной
  - Диагностирование болезней – минимизация пропусков цели,
    за счет некоторого увеличения ложных тревог

Анализ результатов

Методы оценки качества для задачи классификации

Минимизация ошибок 2-го рода в живой природе:

Парейдолия - зрительная иллюзия, заключающаяся в формировании иллюзорных образов на основе реальных объектов.

Тот, кто убежал от пустого куста, передал свои гены потомкам, а тот, кто не убежал от куста, где спрятался тигр, – нет.

Анализ результатов

Методы оценки качества для задачи классификации

Методы оценки качества бинарной классификации:

- Таблица сопряженности
  - Количество примеров для каждого исхода
- Общая точность (Accuracy)
  - Доля правильных ответов
  - Плохо работает при дисбалансе
- Точность, полнота (Precision, Recall)
  - Оценка качества на каждом классе по отдельности
- F1-мера (F1-score)
  - Среднее гармоническое между точностью и полнотой

		Ответ модели	
		Positive	Negative
Истинное значение	Positive	True positive (TP)	False negative (FN)
	Negative	False positive (FP)	True negative (TN)

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

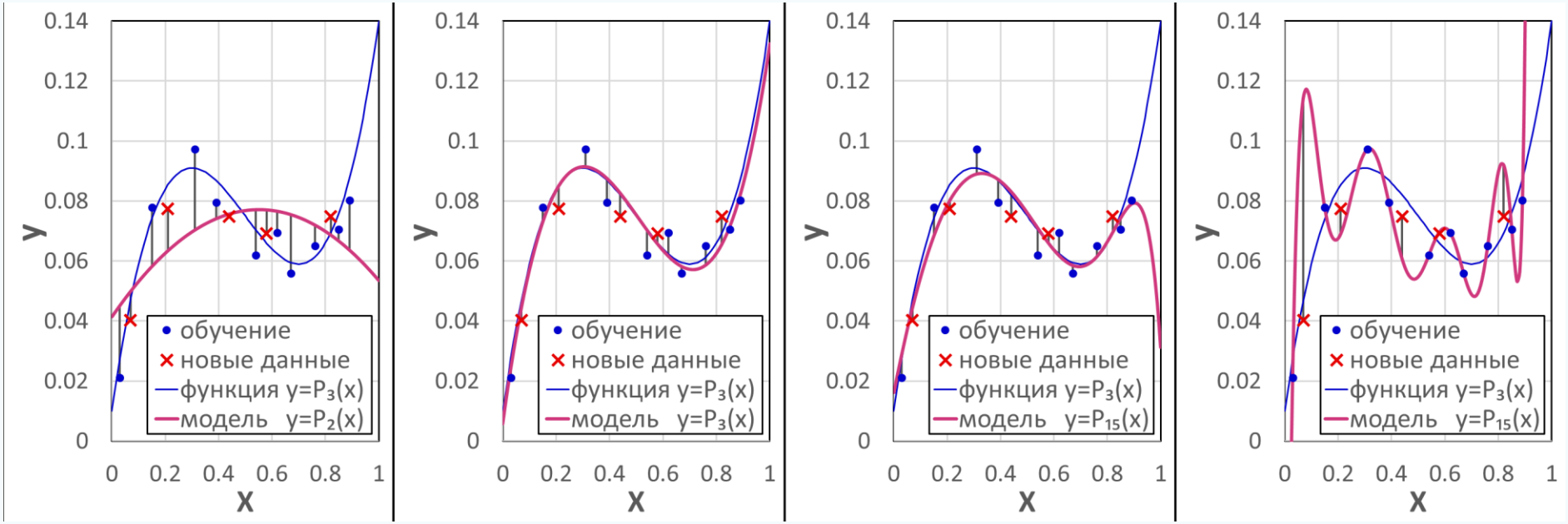
$$F_1\ score = 2\frac{Precision \cdot Recall}{Precision + Recall}$$

- ROC – кривая, AUC

Анализ результатов

Переобучение и недообучение

Качество работы моделей на новых данных:



Недообучение Хорошая модель Переобучение

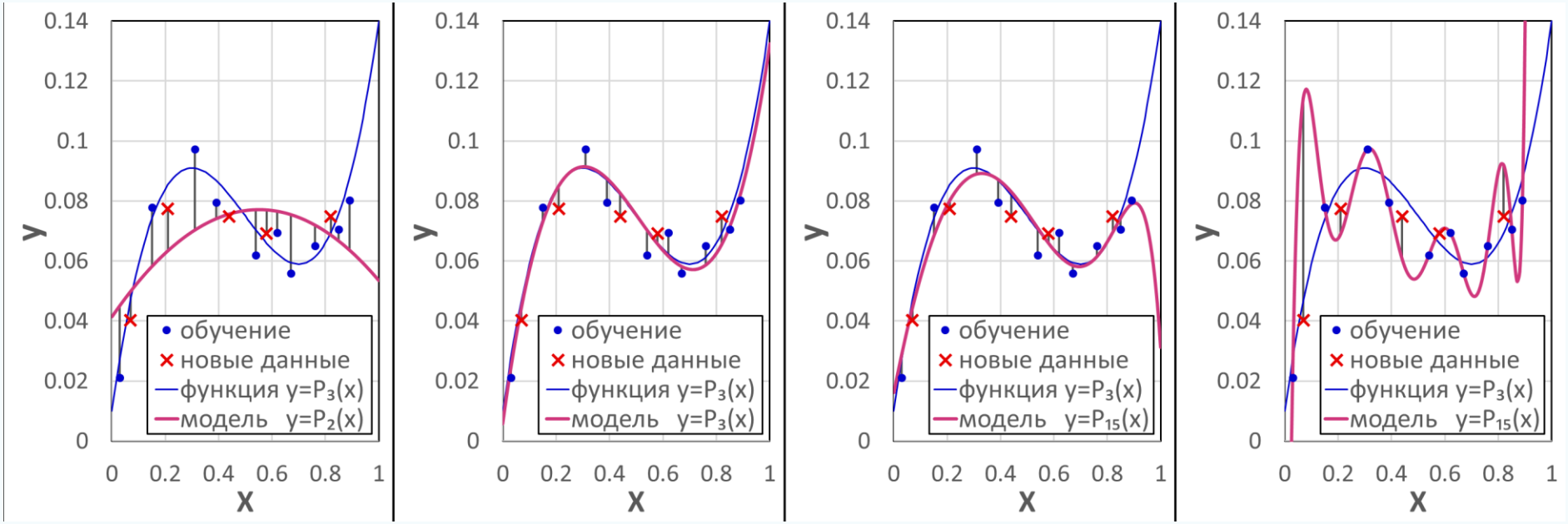
Недообучение – большая ошибка и на данных обучения, и на новых данных

Переобучение – малая ошибка на данных обучения,
большая ошибка на новых данных

Анализ результатов

Переобучение и недообучение

Качество работы моделей на новых данных:



Недообучение Хорошая модель Переобучение

Для выявления случаев переобучения моделей

оценка качества их работы должна производиться на независимых данных,

которые в обучении не участвовали.

Анализ результатов

Кросс-валидация

Проблема ограниченного количества примеров:

- Для выявления случаев переобучения моделей
  оценка качества их работы должна производиться
  на независимых данных, которые в обучении не участвовали.
- Выделение из исходной выборки
  набора данных для независимой оценки качества работы моделей
  ведет к уменьшению количества данных для обучения,
  что может являться причиной недообучения или переобучения.

Решение проблемы – кросс-валидация (cross-validation, CV):

- Обучение группы моделей при разных разбиениях исходного набора
  на данные для обучения и на данные для оценки
  с последующей агрегацией (усреднением) результатов оценивания.

Анализ результатов

Кросс-валидация

Исходный набор данных, состоящая из L примеров,
разбивается N раз на непересекающиеся поднаборы:
тестовый, состоящий из k примеров, и тренировочный из $(L-k)$ примеров.

Способы разделения выборки:

- Контроль на отложенных данных (hold-out CV): $N=1$
- Полный скользящий контроль (complete CV)
  - Оценка строится по всем $N=C^k_L$ разбиениям
- Контроль по отдельным объектам (leave-one-out CV):
  - Частный случай полного скользящего контроля при $k=1$, $N=L$
- Случайные разбиения:
  - Выбирается небольшой число случайных разбиений
    из всего множества $N=C^k_L$ разбиений

Анализ результатов

Кросс-валидация

Исходный набор данных, состоящая из *L* примеров,
разбивается *N* раз на непересекающиеся поднаборы:
тестовый, состоящий из *k* примеров, и тренировочный из *(L-k)* примеров.

Способы разделения выборки:

- Контроль по q блокам (q-fold CV)
  - Выборка случайным образом разбивается на q непересекающихся блоков одинаковой длины k=N/q
  - Каждый блок по очереди становится тестовым поднабором, при этом обучение производится по остальным q-1 блокам
- Контроль по r×q блокам (r×q-fold CV)
  - Контроль по q блокам (q-fold CV) повторяется r раз

Анализ результатов

Кросс-валидация

Стратификация:

- Для более адекватной оценки результатов (меньшей дисперсии оценок), необходимо, чтобы после разбиения каждый поднабор сохранял свойства исходного набора данных.
- Задача классификации
  - Соотношение размеров классов в поднаборах должно быть близко к соотношению размеров классов в исходном наборе
- Задача регрессии
  - Распределения признаков и определяемых величин поднаборов должны быть похожи на их распределения в исходном наборе

Анализ результатов

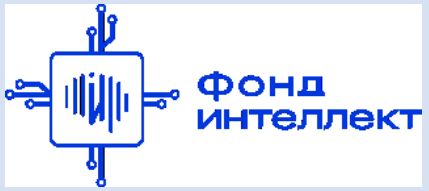
Кросс-валидация

Стратификация:

- Стратификация заключается в разделении исходного набора на части (страты) по какому-либо критерию, например:
  - В задаче классификации – сами классы
  - В задачи регрессии – интервалы по возрастанию какого-либо признака или определяемого параметра
- При разбиении исходного набора на тестовый набор длины k и тренировочный длины $(L-k)$ каждая страта делится между тестовым и тренировочным наборами в том же соотношении $k:(L-k)$

Спасибо за внимание!

Лаборатория адаптивных методов обработки данных



Учебный курс

«Машинное обучение в физике»

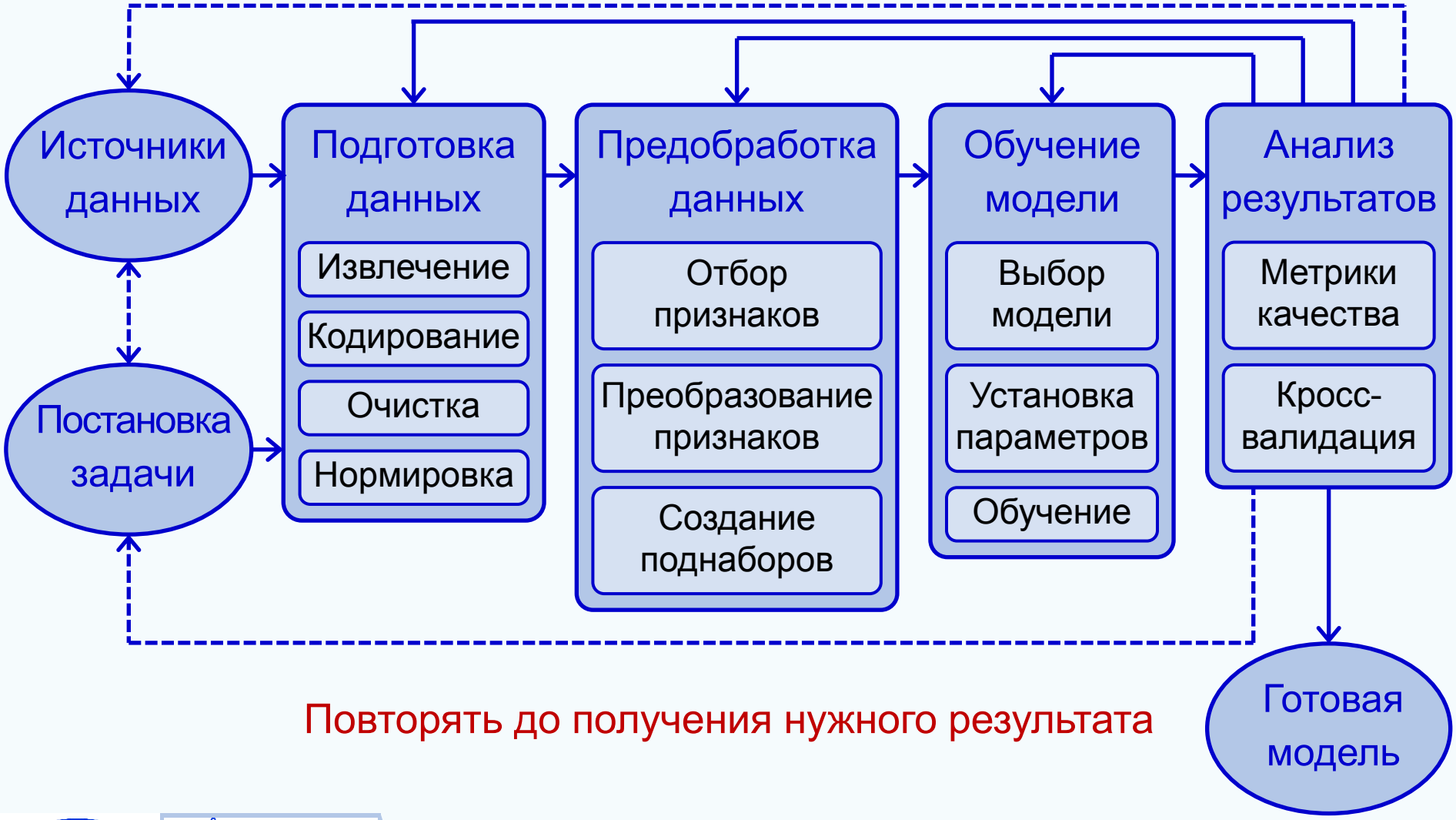
Занятие №4 (лекция).

Отбор и преобразование входных признаков. Оценка значимости входов.

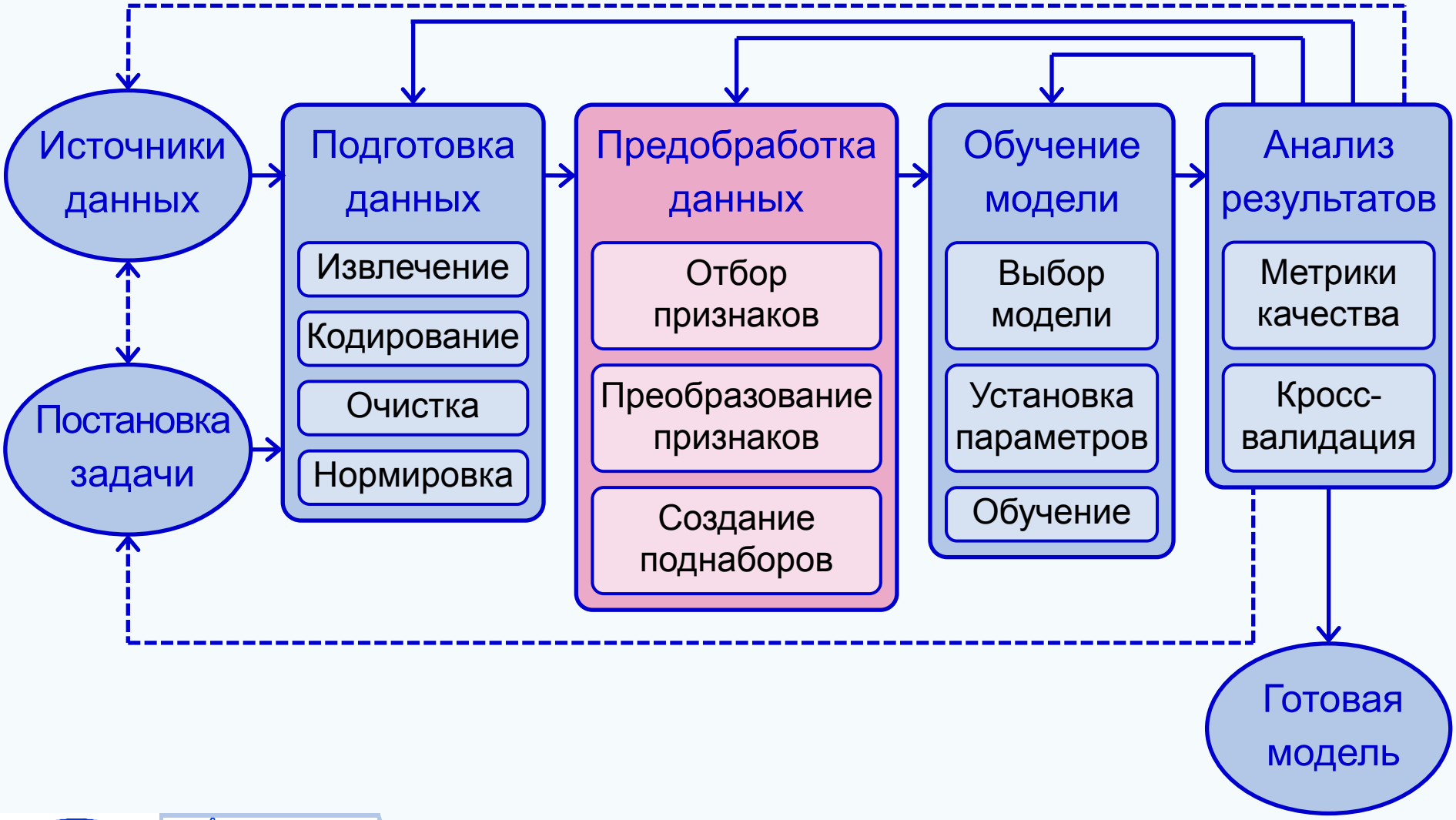
Авторы курса:

С.А. Доленко, И.М. Гаджиев, А.О. Ефиторов, И.В. Исаев, В.Р. Широкий

Общая схема машинного обучения



Содержание занятия



Высокая размерность данных

Проблематика

- Высокая размерность задачи
  - “Проклятие размерности” – количество точек, необходимое для оценки поведения функции многих переменных в *N*-мерном пространстве, зависит от размерности пространства *N* экспоненциально.
  - При большом количестве признаков метрические и линейные методы становятся неэффективными
- Шумовые признаки – признаки, не относящиеся к решаемой задаче
  - Могут влиять в негативную сторону на качество решения задачи
- Мультиколлинеарность – корреляционная зависимость между признаками
  - Приводит к неустойчивой работе некоторых алгоритмов, из-за чего результаты обучения сильно отличаются от запуска к запуску, а также повышается чувствительность решения к шумам в данных

Высокая размерность данных

Проблематика

Мотивация к понижению размерности данных:

- Повышение качества решения
- Улучшение устойчивости работы алгоритмов
- Снижение вычислительных затрат,
  уменьшение размера требуемой оперативной памяти
- Увеличение скорости работы
- Уменьшение размеров хранилищ данных
- Повышение интерпретируемости результатов

Высокая размерность данных

Проблематика

Возможные постановки задачи понижения размерности данных:

- Наилучшее качество решения основной задачи
  - Размерность данных и время получения результата второстепенны
- Минимальная размерность задачи
  - При сохранении приемлемого качества решения основной задачи
- Приемлемый результат
  - Приемлемое качество решения, приемлемая размерность данных, полученные за приемлемое время
- Точно заданная размерность данных
  - При исследовании эффективности алгоритмов
- Интерпретируемый набор данных
- И т. д.

Высокая размерность данных

Способы понижения размерности данных

Способы понижения размерности задачи по выходу:

- Задача регрессии:
  - Автономное определение – построение отдельных моделей с одним выходом для каждого определяемого параметра
  - Групповое определение (multi-task learning) – построение отдельных моделей для взаимосвязанных групп определяемых параметров
- Задача многометочной (multi-label) классификации:
  - Разбиение на отдельные задачи бинарной классификации
  - Построение независимых моделей для каждой метки
- Задача многоклассовой (multiclass) классификации:
  - Разбиение на отдельные задачи бинарной классификации
  - Подходы “one-vs-one”, “one-vs-all”, error-correcting output code (ECOC) (Занятие 15)

Высокая размерность данных

Способы понижения размерности данных

Подходы “one-vs-one”, “one-vs-all”, ECOC

- “One-vs-one”
  - Решается задача о выборе принадлежности к одному из двух классов
  - Объекты *i*-го класса имеют метку 1, *j*-го — метку 0, остальные классы не участвуют
  - Решается *N(N-1)/2* задач бинарной классификации
  - Ответ - тот класс, за который проголосовало большинство

1	0	-	-
1	-	0	-
1	-	-	0
-	1	0	-
-	1	-	0
-	-	1	0

- “One-vs-all” (“One-vs-The-Rest”)
  - Решается задача о принадлежности / не принадлежности к данному классу
  - Объекты *i*-го класса имеют метку 1, остальные - метку 0
  - Решается *N* задач бинарной классификации

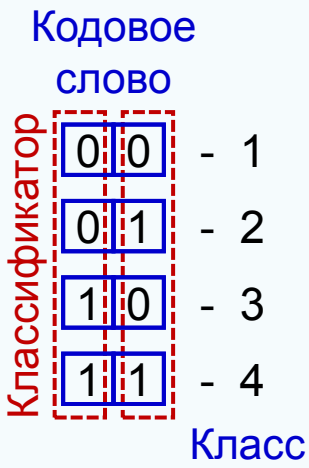
1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

Высокая размерность данных

Способы понижения размерности данных

Подходы “one-vs-one”, “one-vs-all”, ECOC

- “Error-Correcting Output Code (ECOC)
  - Метки классов кодируются бинарными векторами - кодовыми словами длины $L \geq log_2 N$
  - Задача нахождения истинного класса для объекта сводится к определению L неизвестных бит кодового слова
  - Для каждого бита кодового слова строится бинарный классификатор отделяющий группу классов
  - Вычисляется кодовое слово и выбирается класс
  - Если полученного кодового слова нет в словаре, то выбирается класс, ближайший по расстоянию Хэмминга, т.е. где число позиций с отличающимися символами минимально



https://dyakonov.org/2016/01/15/ecoc/

Dietterich T.G., Bakiri G.: Solving multiclass learning problems via error-correcting output codes. Journal of Artificial Intelligence Research, 2, 263–286 (1995).

Высокая размерность данных

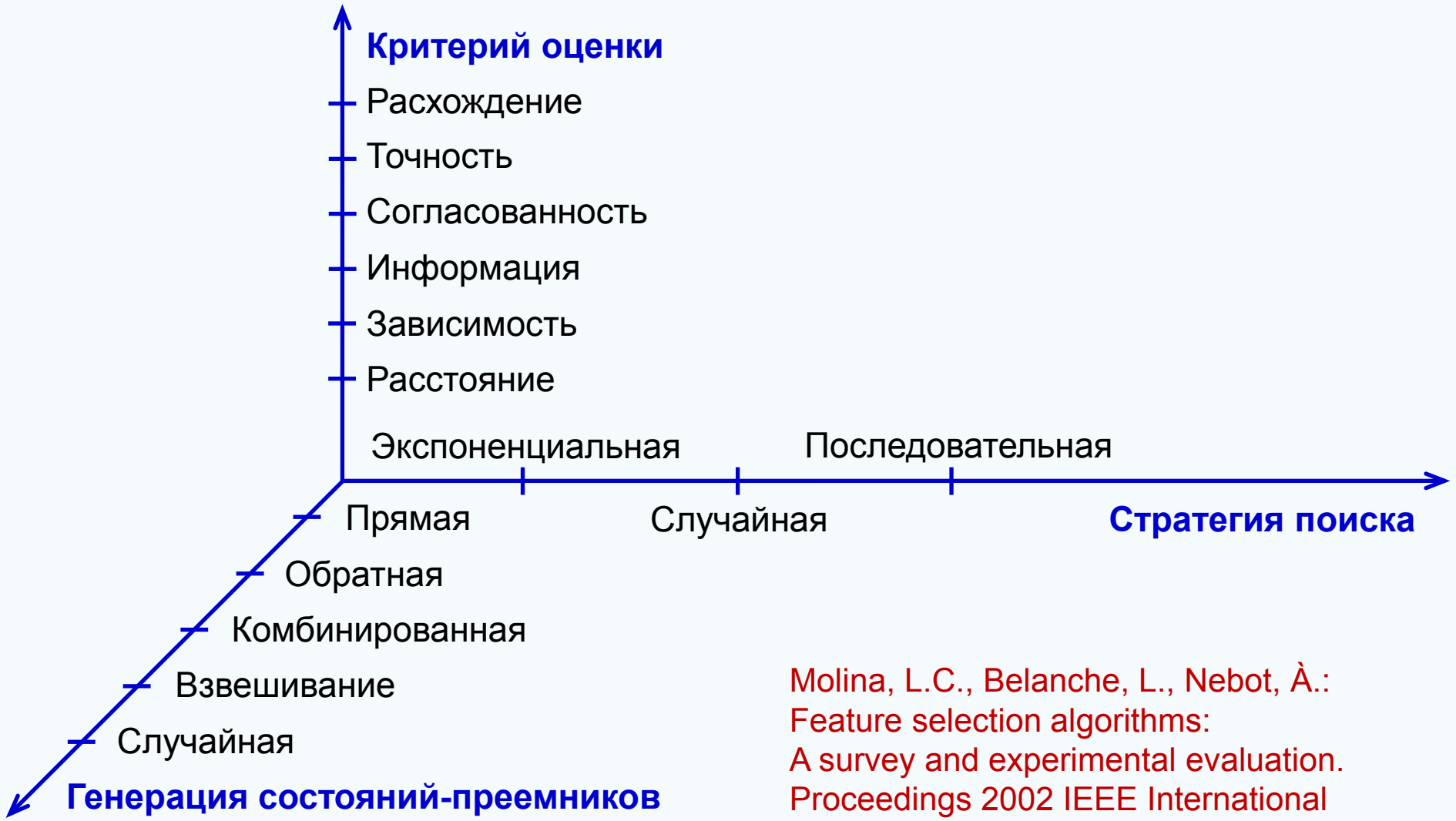
Способы понижения размерности данных

Способы понижения размерности задачи по входу:

- Отбор признаков
  - Выбор некоторого подмножества признаков из исходного множества признаков
- Преобразование признаков
  - Формирование новых признаков на основе существующих

Отбор признаков

Классификация методов отбора



Molina, L.C., Belanche, L., Nebot, À.:
Feature selection algorithms:
A survey and experimental evaluation.
Proceedings 2002 IEEE International
Conference on Data Mining, 306-313 (2002)

Отбор признаков

Классификация методов отбора

- Фильтры (Filter Methods)
  - Для оценки важности признаков используются только характеристики самих данных
- Встроенные методы (Embedded Methods)
  - Методы отбора признаков естественным образом встроены в сами алгоритмы обучения
- Обертки (Wrapper Methods)
  - Основаны на обучении целевого алгоритма решения задачи на различных подмножествах признаков и выборе подмножества, удовлетворяющего заданному критерию

Guyon I., Elisseeff A.: An introduction to variable and feature selection. Journal of machine learning research, 3, 1157-1182 (2003).

Li J. et al.: Feature selection: A data perspective. ACM computing surveys, 50(6), 1-45 (2017). arXiv preprint arXiv:1601.07996v4 (2016).

Отбор признаков

Методы отбора типа “Фильтр”

- Принцип работы
  - Вычисление критерия оценки важности для каждого признака, на основе свойств самого признакового пространства
  - Отбор тех признаков, для которых данный критерий преодолевает заданный порог

Отбор признаков

Методы отбора типа “Фильтр”

- Особенности:
  - Обладают самой **высокой вычислительной эффективностью**
  - **Могут применяться к данным высокой размерности**.
  - Выбирается **универсальный набор** признаков, не зависящий от конкретного целевого алгоритма решения задачи
  - Выбранные при помощи них наборы признаков могут оказаться **неоптимальными** для целевого алгоритма решения задачи
  - **Плохо работают с мультиколлинеарными признаками**, выбирая или отбрасывая сразу группу таких признаков

Отбор признаков

Методы отбора типа “Фильтр”

Отбор по дисперсии (Variance Threshold)

- Идея метода
  - Как правило, признаки с низкой дисперсией (в пределе константные) не являются значимыми
- Особенности
  - Простой в реализации, быстрый, не содержит внутренних параметров
  - Не учитывает взаимосвязи между разными признаками
  - Высокая дисперсия не означает релевантность признака

$$R_i = D[x_i]$$

Отбор признаков

Методы отбора типа “Фильтр”

Отбор по корреляции

- Идея метода
  - Значимые признаки имеют высокую корреляцию (антикорреляцию) с определяемым параметром
- Особенности
  - Простой в реализации, быстрый, не содержит внутренних параметров
  - Вследствие того, что используется коэффициент линейной корреляции, алгоритм способен обнаруживать только линейные зависимости
  - Не учитывает взаимосвязи между разными признаками

$$R_i = |\rho(x_i, y)|$$

$$\rho(x, y) = \frac{cov(x, y)}{\sigma_x \sigma_y} = \frac{E[(x - \bar{x})(y - \bar{y})]}{\sigma_x \sigma_y}$$

Отбор признаков

Встроенные методы отбора

- Принцип работы
  - Обучение целевого алгоритма решения задачи на полном наборе признаков
  - Вычисление критерия оценки важности для каждого признака, основываясь на характеристиках обученной модели
  - Отбор тех признаков, для которых данный критерий преодолевает заданный порог

Отбор признаков

Встроенные методы отбора

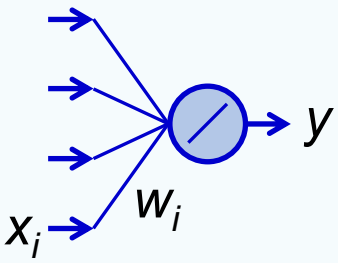
- Особенности:
  - По **вычислительной эффективности** занимают **промежуточное положение** между фильтрами и обертками
  - **Плохо применимы к данным высокой размерности** в случае неэффективной работы самих алгоритмов на таких данных
  - **Наличие мультиколлинеарных признаков** может приводить к **неустойчивости результатов**, при которой множество отбираемых признаков будет отличаться от запуска к запуску процедуры отбора

Отбор признаков

Встроенные методы отбора

Анализ весов линейной регрессии

- Идея метода
  - Важные признаки имеют более высокие значения весов
- Особенности
  - При наличии мультиколлинеарных признаков
    возможно возникновение больших значений весов,
    что приводит в таком случае к неприменимости данного метода отбора,
    а также к неустойчивости результатов работы самого алгоритма



$$R_i = |w_i|$$

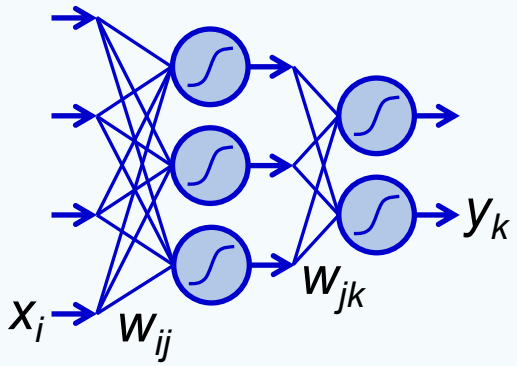
Отбор признаков

Встроенные методы отбора

Анализ весов нейронной сети

- Идея метода
  - Влияние признака на целевую переменную,
    может осуществляться по множеству траекторий,
    ведущих от соответствующего входа к соответствующему выходу
  - Вклад одной траектории определяется как произведение весов вдоль нее
  - Важность признака определяется как сумма вкладов
    всех возможных траекторий между исследуемыми входом и выходом.

$$R_{ik} = \frac{\sum_{j=1}^{M} |w_{ij} w_{jk}|}{\sum_{i=1}^{N} \sum_{j=1}^{m} |w_{ij} w_{jk}|}$$



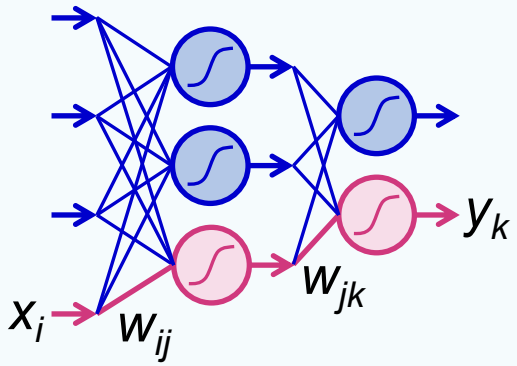
Отбор признаков

Встроенные методы отбора

Анализ весов нейронной сети

- Идея метода
  - Влияние признака на целевую переменную,
    может осуществляться по множеству траекторий,
    ведущих от соответствующего входа к соответствующему выходу
  - Вклад одной траектории определяется как произведение весов вдоль нее
  - Важность признака определяется как сумма вкладов
    всех возможных траекторий между исследуемыми входом и выходом.

$$R_{ik} = \frac{\sum_{j=1}^{M} |w_{ij} w_{jk}|}{\sum_{i=1}^{N} \sum_{j=1}^{m} |w_{ij} w_{jk}|}$$



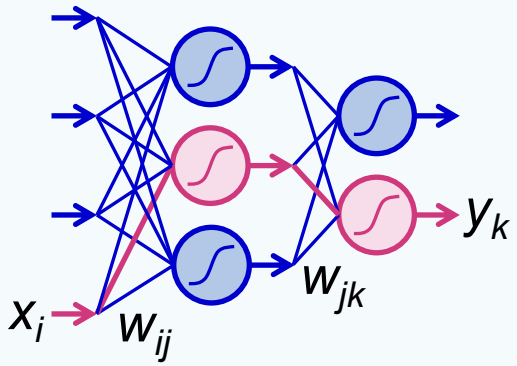
Отбор признаков

Встроенные методы отбора

Анализ весов нейронной сети

- Идея метода
  - Влияние признака на целевую переменную,
    может осуществляться по множеству траекторий,
    ведущих от соответствующего входа к соответствующему выходу
  - Вклад одной траектории определяется как произведение весов вдоль нее
  - Важность признака определяется как сумма вкладов
    всех возможных траекторий между исследуемыми входом и выходом.

$$R_{ik} = \frac{\sum_{j=1}^{M} |w_{ij} w_{jk}|}{\sum_{i=1}^{N} \sum_{j=1}^{m} |w_{ij} w_{jk}|}$$



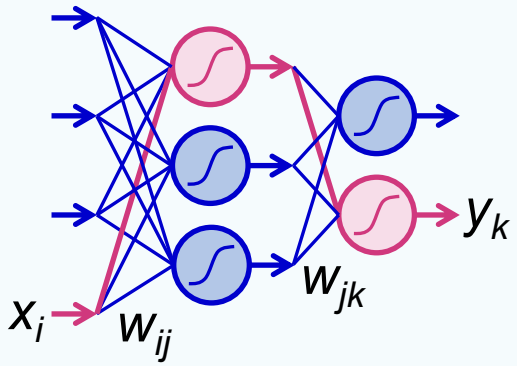
Отбор признаков

Встроенные методы отбора

Анализ весов нейронной сети

- Идея метода
  - Влияние признака на целевую переменную,
    может осуществляться по множеству траекторий,
    ведущих от соответствующего входа к соответствующему выходу
  - Вклад одной траектории определяется как произведение весов вдоль нее
  - Важность признака определяется как сумма вкладов
    всех возможных траекторий между исследуемыми входом и выходом.

$$R_{ik} = \frac{\sum_{j=1}^{M} |w_{ij} w_{jk}|}{\sum_{i=1}^{N} \sum_{j=1}^{m} |w_{ij} w_{jk}|}$$



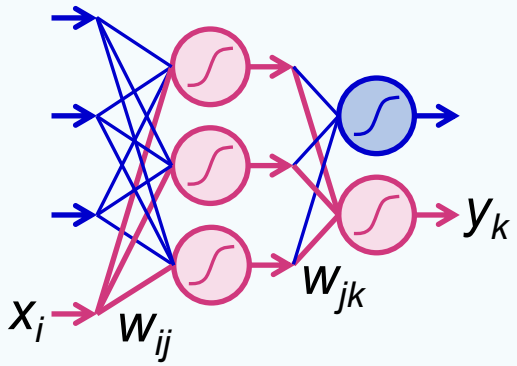
Отбор признаков

Встроенные методы отбора

Анализ весов нейронной сети

- Идея метода
  - Влияние признака на целевую переменную,
    может осуществляться по множеству траекторий,
    ведущих от соответствующего входа к соответствующему выходу
  - Вклад одной траектории определяется как произведение весов вдоль нее
  - Важность признака определяется как сумма вкладов
    всех возможных траекторий между исследуемыми входом и выходом.

$$R_{ik} = \frac{\sum_{j=1}^{M} |w_{ij} w_{jk}|}{\sum_{i=1}^{N} \sum_{j=1}^{m} |w_{ij} w_{jk}|}$$



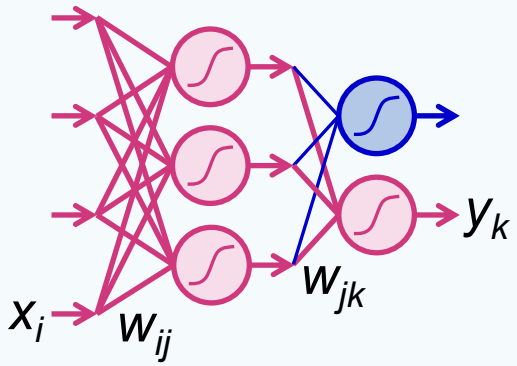
Отбор признаков

Встроенные методы отбора

Анализ весов нейронной сети

- Идея метода
  - Влияние признака на целевую переменную,
    может осуществляться по множеству траекторий,
    ведущих от соответствующего входа к соответствующему выходу
  - Вклад одной траектории определяется как произведение весов вдоль нее
  - Важность признака определяется как сумма вкладов
    всех возможных траекторий между исследуемыми входом и выходом.

$$R_{ik} = \frac{\sum_{j=1}^{M} |w_{ij} w_{jk}|}{\sum_{i=1}^{N} \sum_{j=1}^{m} |w_{ij} w_{jk}|}$$



Отбор признаков

Встроенные методы отбора

Анализ весов нейронной сети

- Особенности
  - Алгоритм является оценочным, поэтому возможно использование быстрых алгоритмов обучения нейронных сетей.
  - Наиболее часто используется для нейронных сетей с 1 скрытым слоем
  - Легко масштабируется на случай сетей с большим числом скрытых слоев
  - Контрастность результатов падает с повышением сложности сети, а также с увеличением количества признаков
  - Мультиколлинеарность также приводит к снижению контрастности метода, вследствие “размазывания” важности по группе таких признаков.

Gevrey M. *et al.:* Review and comparison of methods to study the contribution of variables in artificial neural network models. Ecological modelling, 160(3), 249-264 (2003).

Satizábal M. H.F., Pérez-Uribe A.: Relevance metrics to reduce input dimensions in artificial neural networks. Lecture Notes in Computer Science, 4668, 39-48 (2007).

Отбор признаков

Встроенные методы отбора

Способы повышения контрастности анализа весов

- Агрегация результатов нескольких обученных моделей
  - Например, усреднение важности признака по нескольким моделям
- Регуляризация (Sparse Learning based Methods) (Занятие 7)
  - Введение в процедуру обучения ограничений на большие веса модели
  - Может достигаться за счет внесения в функцию потерь дополнительного штрафного члена:

$$Loss = Loss(X, Y, W) + \alpha \cdot penalty(W)$$

- ✓ L1 – регуляризация (Lasso) $penalty(W) = \|W\|_1 = \sum_{i=1}^{d} |w_i|$
- ✓ L2 – регуляризация (Регуляризация Тихонова, Ridge) $penalty(W) = \|W\|_2 = \sum_{i=1}^{d} w_i^2$
- ✓ Регуляризация ElasticNet $penalty(W) = \alpha_1 \|W\|_1 + \alpha_2 \|W\|_2$

Отбор признаков

Методы отбора типа “Обертки”

- Принцип работы
  - Обучение целевого алгоритма решения задачи на различных подмножествах признаков
  - Оценка качества решения для каждой модели
  - Выбор того подмножества признаков, на котором достигается выполнение критерия остановки процедуры отбора

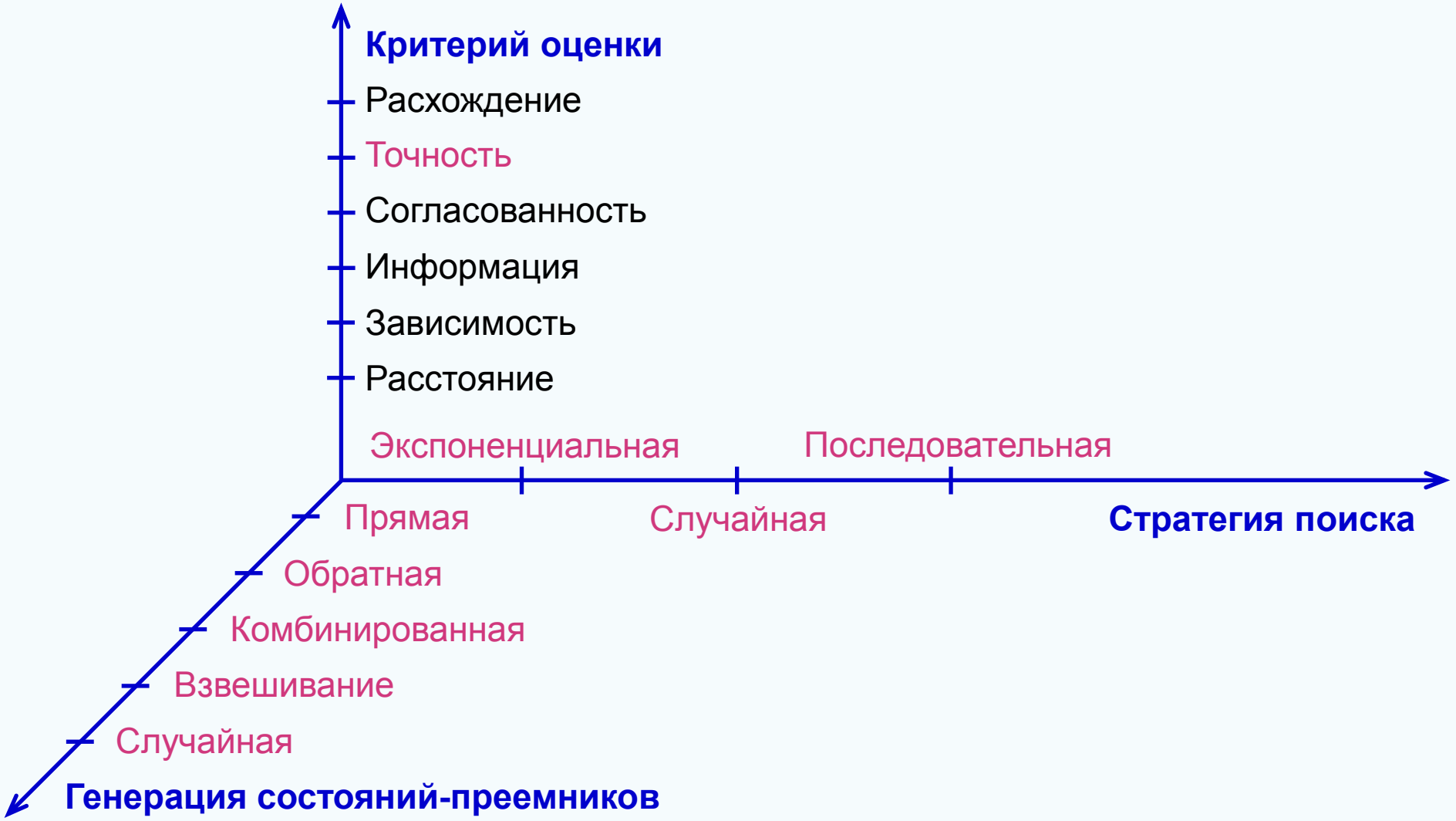
Отбор признаков

Методы отбора типа “Обертки”

- Особенности:
  - Обладают самой **низкой вычислительной эффективностью**
  - **Применение к данным высокой размерности затруднено,** как вследствие высоких вычислительных затрат, так и вследствие неэффективной работы целевого алгоритмов на таких данных
  - Возможен подбор **оптимального набора** признаков для конкретного целевого метода решения задачи
  - **Могут работать с мультиколлинеарными признаками**, выбирая из группы таких признаков те, которые несут наибольшую информацию об объекте

Отбор признаков

Методы отбора типа “Обертки”



Отбор признаков

Методы отбора типа “Обертки”

- Реализации
  - **Экспоненциальный поиск** (Exponential search, Complete search)
    - ✓ Исчерпывающий перебор
    - ✓ FOCUS, Метод ветвей и границ (Branch-and-bound) и др.
  - **Последовательный поиск** (Sequential search, Greedy Hill Climbing)
    - ✓ Последовательное добавление и/или исключение признаков
    - ✓ Рекурсивное исключение признаков и др.
  - **Случайный поиск** (Random search, Heuristic search)
    - ✓ Метод группового учета аргументов
    - ✓ Эволюционные методы: генетические алгоритмы, метод роя частиц (particle swarm optimization) и др.
    - ✓ Симуляция отжига и др.

Отбор признаков

Методы отбора типа “Обертки”

Исчерпывающий перебор

(Exhaustive Feature Selection)

- Идея метода:
  - Полный перебор всех возможных поднаборов признаков
- Особенности
  - Требуется перебор (2^N-1) возможных вариантов
  - Эффективен при очень малом количестве признаков (~20) и малоресурсозатратном целевом алгоритме обучения
  - Гарантированно будет найден лучший поднабор

Отбор признаков

Методы отбора типа “Обертки”

Последовательное добавление признаков

(Add, Greedy Forward Selection, Forward Sequential Feature Selection)

- Идея метода:
  - Итеративное добавление “лучшего” признака из имеющихся.
  - На каждой итерации добавляется один признак, который в сочетании с полученным на предыдущей итерации поднабором обеспечивает наилучшее качество решения
  - Процедура отбора стартует с одного признака
- Особенности
  - Сложность алгоритма $O(N^2)$ или $O(nN)$ при остановке на n-й итерации
  - Алгоритм склонен включать в поднабор лишние признаки.
  - Если признак был единожды включён, он остаётся в поднаборе навсегда

Отбор признаков

Методы отбора типа “Обертки”

Последовательное исключение признаков

(Del, Greedy Backward Elimination, Backward Sequential Feature Selection)

- Идея метода:
  - Итеративное исключение “худшего” признака из имеющихся
  - На каждой итерации удаляется один признак, исключение которого из полученного на предыдущей итерации поднабора обеспечивает наилучшее качество решения
  - Процедура отбора стартует с полного набора признаков
- Особенности
  - Сложность алгоритма $O(N^2)$ или $O(nN)$ при остановке на *n-й* итерации
  - Медленнее, чем Add, т.к. в начале процедуры отбора обучение моделей происходит на больших поднаборах
  - Применяется когда заранее известно, что информативных признаков гораздо больше, чем шумовых.

Отбор признаков

Методы отбора типа “Обертки”

Последовательное добавление-исключение признаков (Add-Del, Greedy Hill Climbing)

- Идея метода:
  - “Два шага вперёд – один назад”
  - Итеративное чередование k операций Add и m операций Del (k>m)
  - Процедура отбора стартует с одного признака
- Особенности
  - Работает дольше, чем Add и Del по-отдельности
  - На практике гораздо чаще удаётся найти лучшее решение, чем методам Add или Del
  - Не гарантирует оптимальность

Отбор признаков

Методы отбора типа “Обертки”

Рекурсивное исключение признаков

(Recursive feature elimination)

- Идея метода:
  - Итеративное исключение “худших” признаков из имеющихся
  - На каждой итерации используется встроенный метод отбора признаков
  - Удаляется один или несколько признаков,
    обладающие наихудшим значением критерия важности,
    затем обучается новая модель на оставшемся подмножестве
  - Процедура отбора стартует с полного набора признаков
- Особенности
  - Сложность алгоритма *O(N)* или *O(n)* при остановке на *n-й* итерации

Отбор признаков

Комплексные алгоритмы отбора признаков

- Идея подхода:
  - Последовательное применение нескольких методов отбора в случае очень высокой размерности исходного множества признаков
- Примеры реализации:
  - Отбор по корреляции + анализ весов нейронной сети
    - ✓ Исключение неинформативных признаков посредством отбора по корреляции (r~0)
    - ✓ Обучение нейронной сети на получившимся поднаборе признаков и последующее применение метода анализа весов
  - Двукратный анализ весов нейронной сети
    - ✓ Обучение нескольких нейронных сетей на непересекающихся подмножествах признаков и отбор посредством анализа весов
    - ✓ Слияние полученных поднаборов признаков, обучение нейронной сети, повторный отбор посредством анализа весов

Dolenko S. et al.: Comparison of Adaptive Algorithms for Significant Feature Selection ... Lecture Notes in Computer Science, 5769, 397-405 (2009)

Преобразование признаков

Методы преобразования признаков

- Мотивация к применению:
  - Снижение размерности задачи
  - Устранение мультиколлинеарности
  - Добавление нелинейности / преобразование в линейную задачу
  - Оценка реальной размерности данных
- Примеры преобразований:
  - Создание новых признаков как функций от исходных
  - Фурье и вейвлет-преобразование (Занятие 9)
  - Анализ главных компонент (разложение по сингулярным значениям), метод проекций на латентные структуры (Занятие 6, 9)
  - Kernel-trick (Занятие 7)
  - Автоэнкодеры, эмбеддинги (Занятие 8)
  - t-SNE (Занятие 6), UMAP и т.д.

Оценка размерности задачи

Методы оценки реальной размерности задачи

Размерность задачи - количество независимых параметров, необходимых для описания исследуемого объекта

- ❑ Мотивация:
  - Оценка предела, до которого следует снижать размерность задачи путем отбора и преобразования признаков
- ❑ Способы оценки:
  - Анализ главных компонент
    - ✓ Количество главных компонент, описывающих 95 % дисперсии
  - Автоэнкодеры
    - ✓ Минимальный размер узкого горла, при котором возможно восстановление данных
  - Вычисление фрактальной размерности

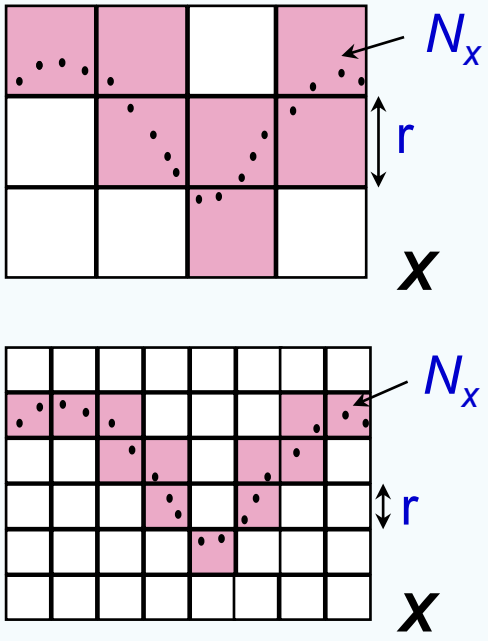
Оценка размерности задачи

Оценка фрактальной размерности данных

Метод Box-counting (размерность Минковского)

- Идея метода:
  - Разобьем пространство переменных X размерности n на ячейки - гиперкубы со стороной r и будем уменьшать r
  - Общее количество гиперкубов N ~ $(1/r)^n$
  - Количество гиперкубов с данными Nx ~ $(1/r)^d$
  - фрактальная размерность d:

$$d = \lim_{r \to 0} \left[\frac{\log N_x(r)}{\log(1/r)} \right]$$



  - При численной реализации алгоритма, параметр r подбирается таким образом, чтобы в каждом заполненном гиперкубе находилось в среднем по 2 точки

Оценка размерности задачи

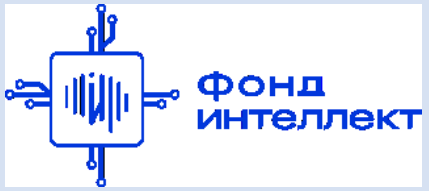
Оценка фрактальной размерности данных

Метод Box-counting (размерность Минковского)

- Особенности
  - Метод может давать завышенную оценку размерности
- Области применения
  - Оценка реальной размерности задачи
  - Фрактальная размерность может использоваться как композитный признак для других методов машинного обучения в следующих областях:
    - ✓ Обработки изображений
    - ✓ Анализа временных рядов

Спасибо за внимание!

Лаборатория адаптивных методов обработки данных



Учебный курс

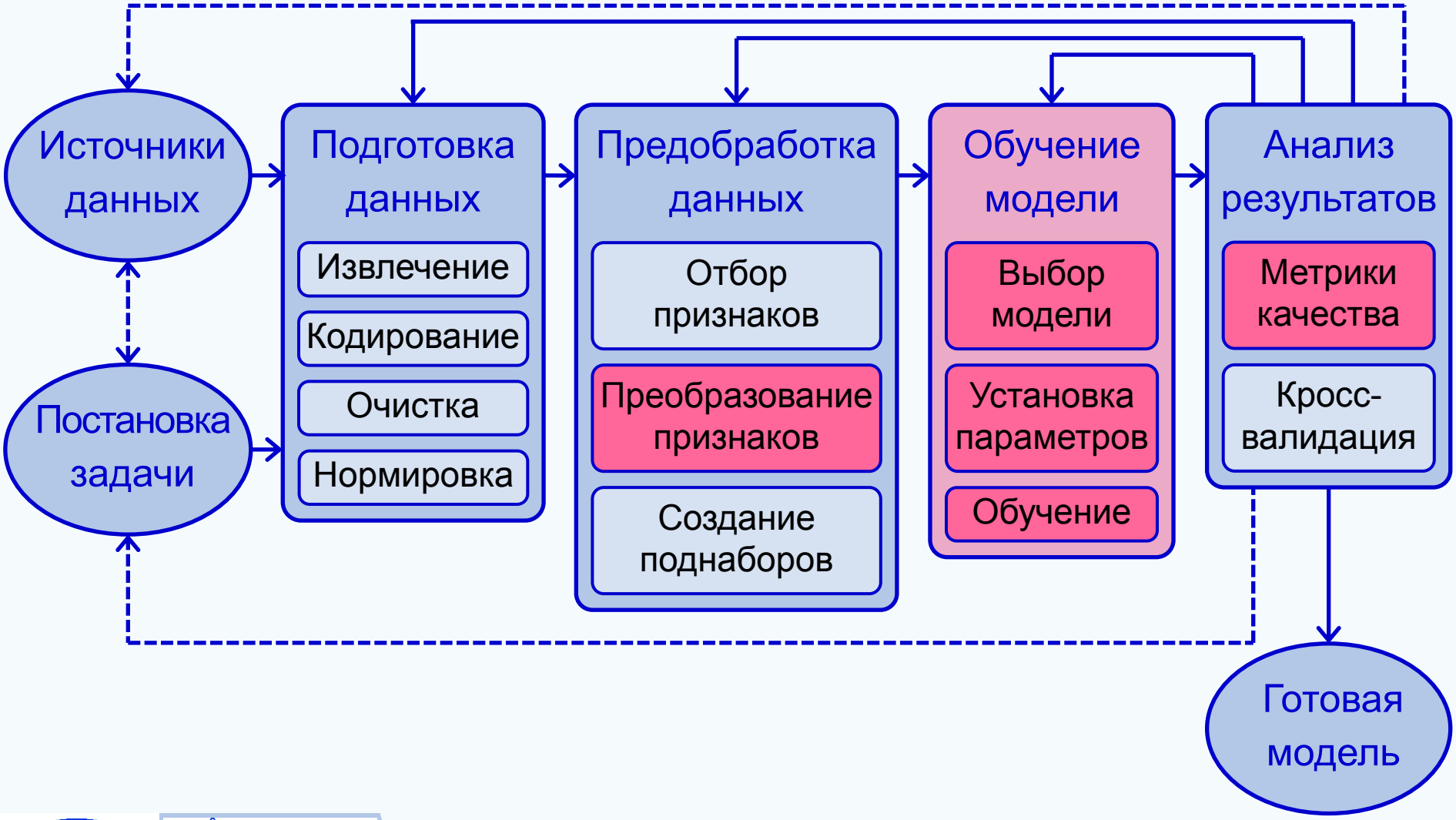
«**Машинное обучение в физике**»

Занятие №6 (лекция).
Анализ главных компонент и методы на его основе.
Кластер-анализ.
Нейронные сети и самоорганизующиеся карты Кохонена.

Авторы курса:

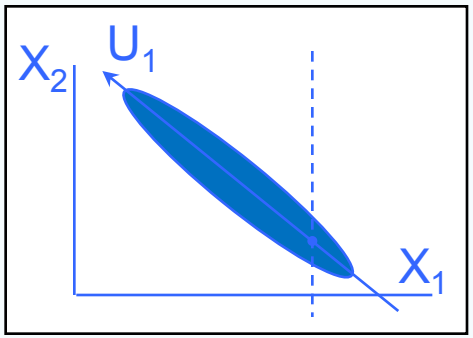
С.А. Доленко, И.М. Гаджиев, А.О. Ефиторов, И.В. Исаев, В.Р. Широкий

Содержание занятия



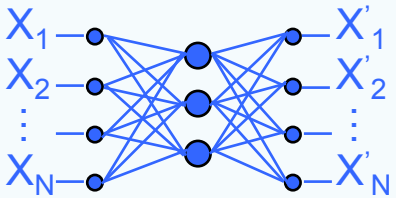
Разложение по сингулярным значениям. Линейный анализ главных компонент (АГК)

Рассчитаем для данных ковариационную матрицу C_{ij}.



Её собственные вектора U_n: $CU_n = \lambda_n U_n$ указывают направления её главных компонент, а собственные значения λ_n являются значениями дисперсии вдоль этих направлений.
Это разложение по сингулярным значениям (SVD – Singular Value Decomposition).

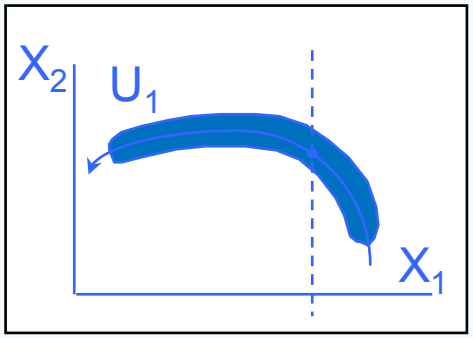
- Ограничиваясь несколькими старшими главными компонентами, можно понизить размерность данных.
- Наиболее вероятным значением для пропущенного входа является значение, соответствующее положению точки на первой главной компоненте: $U_2 = c_1x_1 + c_2x_2 = 0 \Rightarrow x_2 = -(c_1/c_2)\cdot x_1$



Аналог понижения размерности с помощью линейного АГК даёт автоассоциативная нейронная сеть с “узким горлом” и линейными нейронами.

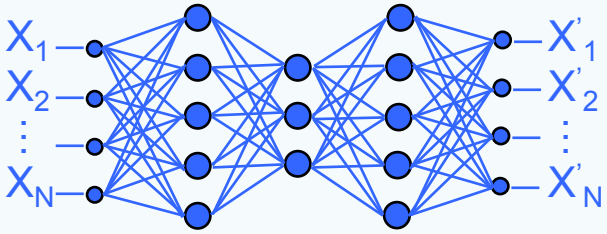
Преимущества такой реализации:
1) адаптивность; 2) простота введения нелинейности

Нелинейный анализ главных компонент



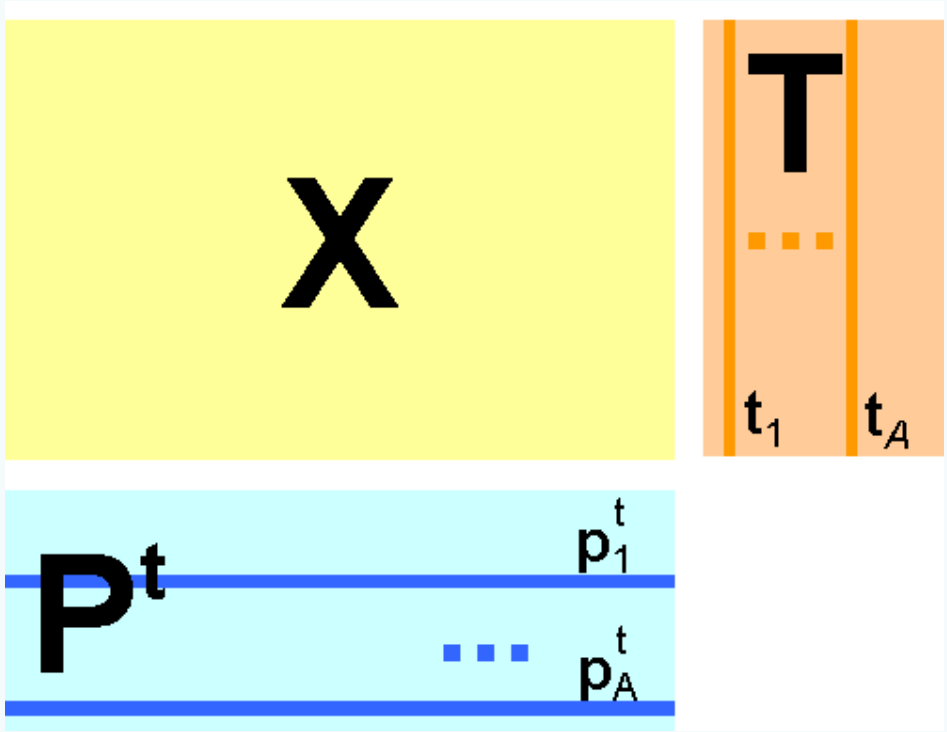
Более нетривиальные результаты
дает нелинейный АГК, в котором
главные компоненты являются не прямыми,
а непрерывными кривыми произвольного вида.

- Ограничиваясь несколькими старшими главными компонентами, можно понизить размерность данных.
- Наиболее вероятным значением для пропущенного входа является значение, соответствующее положению точки на первой главной компоненте



Аналог понижения размерности
с помощью нелинейного АГК даёт пятислойная
автоассоциативная нейронная сеть
с “узким горлом” и нелинейными нейронами.

Линейный анализ главных компонент: матричный подход



Матрица счетов (scores)

Размерность – (*I*×*A*)

A новых координат

Матрица **X**

(исходные данные)

Размерность – (*I*×*J*)

(**I** примеров,

J признаков)

Матрица нагрузок (loadings)

Размерность – (*A*×*J*)

Матрица перехода

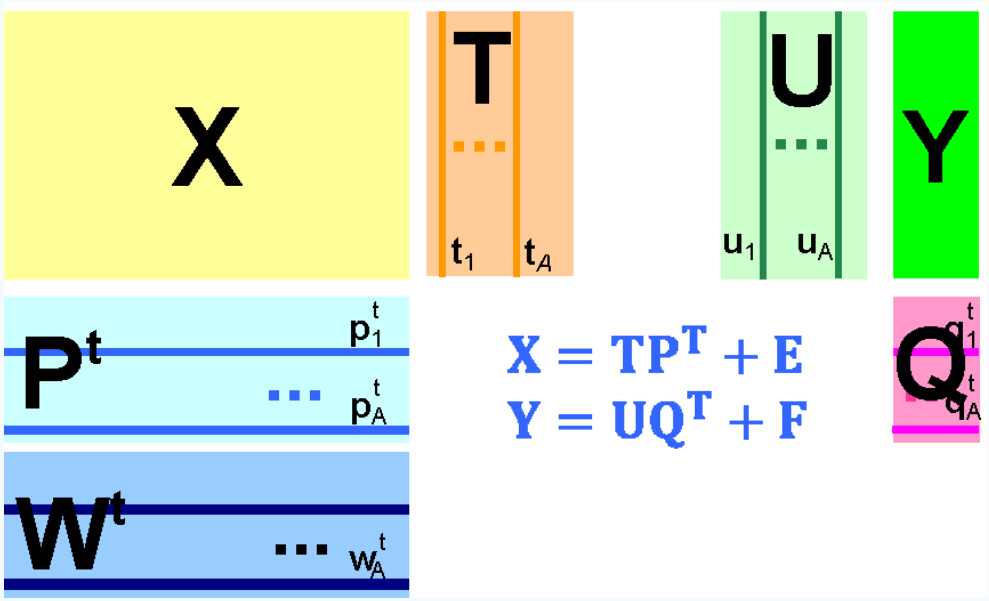
в пространство ГК (**A**-мерное)

$$\mathbf{X} = \mathbf{T}\mathbf{P}^{\mathbf{T}} + \mathbf{E} = \sum_{a=1}^{A} t_a (p_a)^T + \mathbf{E}$$

Метод проекций на латентные структуры (ПЛС) (Projection to Latent Structures, PLS)

В основе линейная регрессия: $Y = BX$

Проводим декомпозицию обеих матриц **X** и **Y** так, чтобы максимизировать корреляцию между векторами счетов t_a и u_a



U – матрица счетов **Y**
Q – матрица нагрузок **Y**

T – матрица счетов **X**
P – матрица нагрузок **X**
W – матрица взвешенных нагрузок **X**

Матрица регрессионных коэффициентов:

$$B = W(P^T W)^{-1} Q^T$$

Многомерное разрешение кривых (МРК)

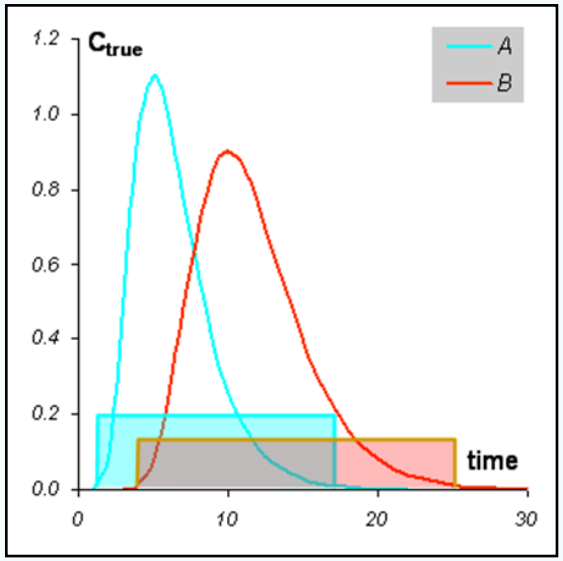
Многомерное разрешение кривых (Multivariate Curve Resolution) – группа основанных на линейном АГК методов восстановления спектральных профилей компонент **S(λ)** и их концентрационных профилей **C(t)** из смешанных спектров **X(λ,t)**

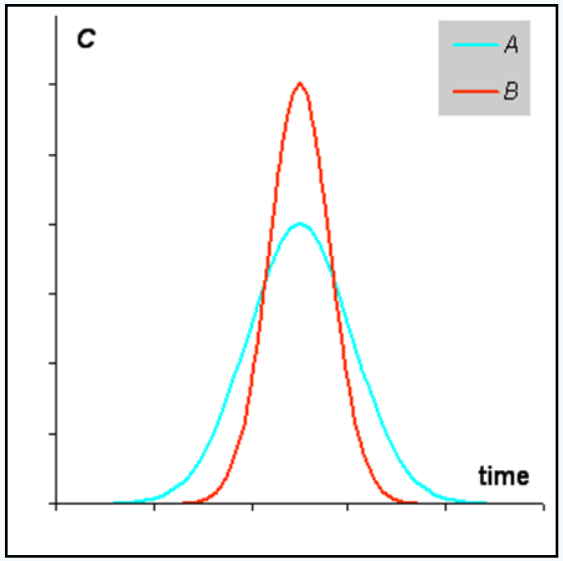
$$X = CS^T + E$$

Неоднозначность (неопределённость) разрешения:

1) Вращательная: $CS^T = CRR^TS^T = (CR)(SR)^T = \check{C}\check{S}^T$

2) Масштабная: $CS^T = CRR^{-1}S^T = (CR)(SR^{-1})^T = \check{C}\check{S}^T$





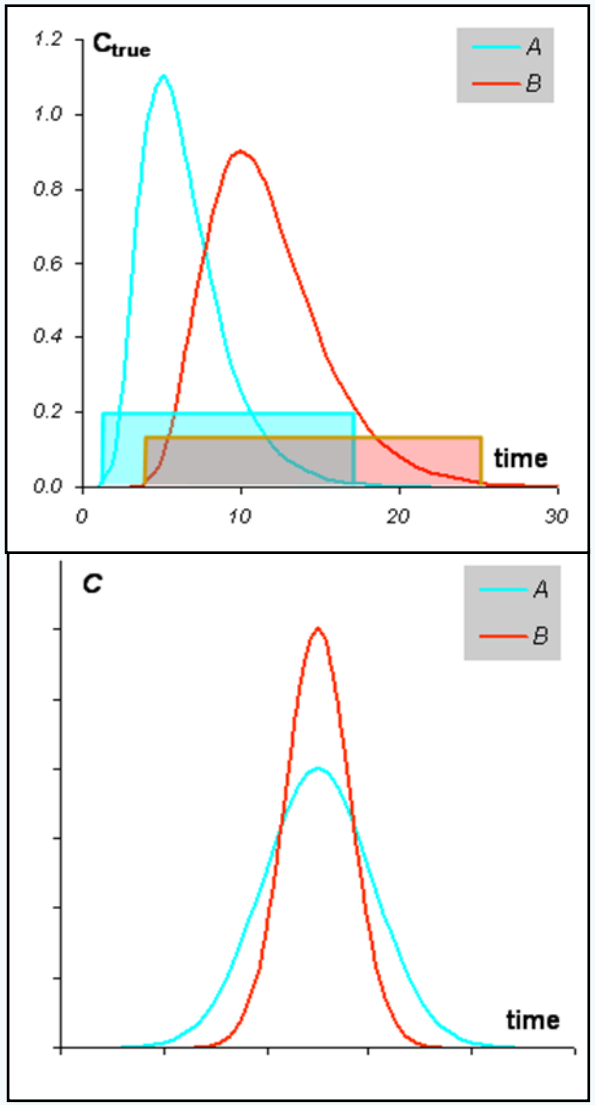
Многомерное разрешение кривых

$$X=CS^T+E$$

Необходимые условия разрешимости задачи МРК (теоремы Манне):

1) Концентрационный профиль компонента может быть восстановлен, если все остальные концентрационные окна, пересекающиеся с его окном, имеют участки вне этого окна.

2) Спектральный профиль компонента может быть восстановлен, если его концентрационное окно не лежит целиком внутри окна другого компонента.



Решение задач кластеризации

- **Подготовка и нормировка данных**
  - Существенную роль играет метрика расстояния между примерами
  - Шумы в данных существенно влияют на результат
- **Выбор алгоритма кластеризации**
  - Основанные на расстоянии до центроида кластера
  - Основанные на расстоянии до других примеров кластера
- **Поиск оптимального числа кластеров**
  - Дивизимный (делятся самые большие кластеры)
  - Агломеративный (объединяются самые близкие кластеры)
- **Критерии оценки качества кластеризации**
  - Кластеры должны быть компактными и изолированными (примеры одного кластера близко, кластеры друг от друга далеко)

Алгоритм кластеризации K средних (k-means)

Шаг 1. Задается полное число кластеров **K**.

Выбираются центры кластеров

(напр., случайным образом, или самые далекие точки)

Шаг 2. Определяется принадлежность примеров к кластерам

по расстоянию до центра ближайшего кластера

(в соответствии с выбранной метрикой расстояния)

Шаг 3. Счетчик итераций **i=i+1**

Шаг 4. Находится центр каждого кластера

(среднее всех точек кластера)

Шаг 5. Определяется принадлежность примеров к кластерам

Шаг 6. Если принадлежность примеров к кластерам

изменилась по сравнению с предыдущей итерацией,

то переход на Шаг 3

Дивизимный иерархический поиск числа кластеров

Шаг 1. Изначально число кластеров **K=2**

Шаг 2. Запускается алгоритм K средних.

Шаг 3. Ищется кластер C_m с максимальным внутрикластерным расстоянием.

Шаг 4. Найденный кластер разбивается на два подкластера:

ищутся индексы k_1, k_2 такие, что $\delta(x_{k_1}, x_{k_2}) \geq \delta(x_{k_3}, x_{k_4})$

для любых k_3, k_4 из C_m

Шаг 5. $X(k_1)$ становится новым центром C_m,

а $X(k_2)$ - центром нового кластера $(K=K+1)$.

Центры остальных кластеров остаются прежними.

Шаг 6. Если $K<N$ (максимального числа кластеров), то идти на Шаг 2.

Шаг 7. Поиск оптимального числа кластеров по порогу.

Агломеративный иерархический поиск числа кластеров

Шаг 1. Изначально число кластеров **K=N**

Шаг 2. Запускается алгоритм K средних.

Шаг 3. Ищутся два кластера с минимальным межкластерным расстоянием $d = \frac{1}{N_i N_j} \sum_{x' \in C_i} \sum_{x'' \in C_j} \delta(x' - x'')$

Шаг 4. Найденные кластеры объединяются **(K=K-1)**.

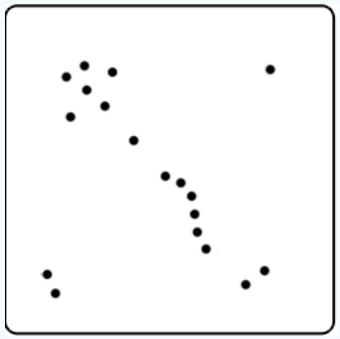
Центры остальных кластеров остаются прежними.

Шаг 5. Если **K>1**, то идти на Шаг 2.

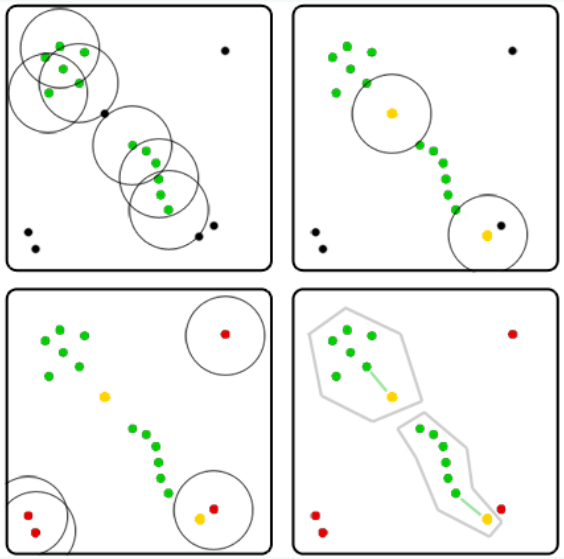
Шаг 6. Поиск оптимального числа кластеров по порогу.

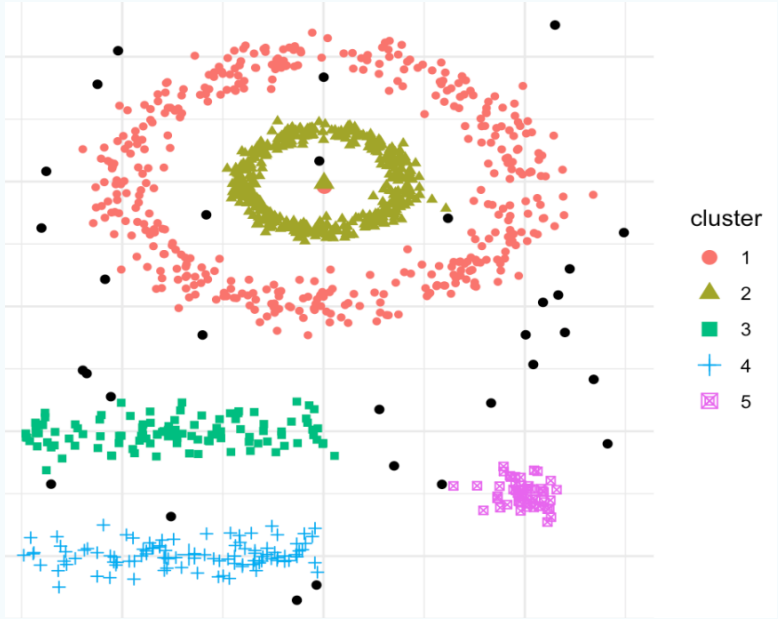
Алгоритм кластеризации DBSCAN
(Density-Based Spatial Clustering of Applications with Noise)

- Назовем область $E(x)$, для которой $y: \rho(x,y) \le \varepsilon$, ε-окрестностью объекта x.
- Корневым объектом или ядерным объектом степени m называется объект, ε-окрестность которого содержит не менее m объектов: $|E(x)| \ge m$.
- Объект p непосредственно плотно-достижим из объекта q, если $p \in E(q)$ и q – корневой объект.
- Объект p плотно-достижим из объекта q, если $\exists p_1, p_2, \ldots, p_n$, $p_1 = q$, $p_n = p$ такие, что $\forall i \in 1 \ldots n-1$ p_{i+1} непосредственно плотно-достижим из p_i

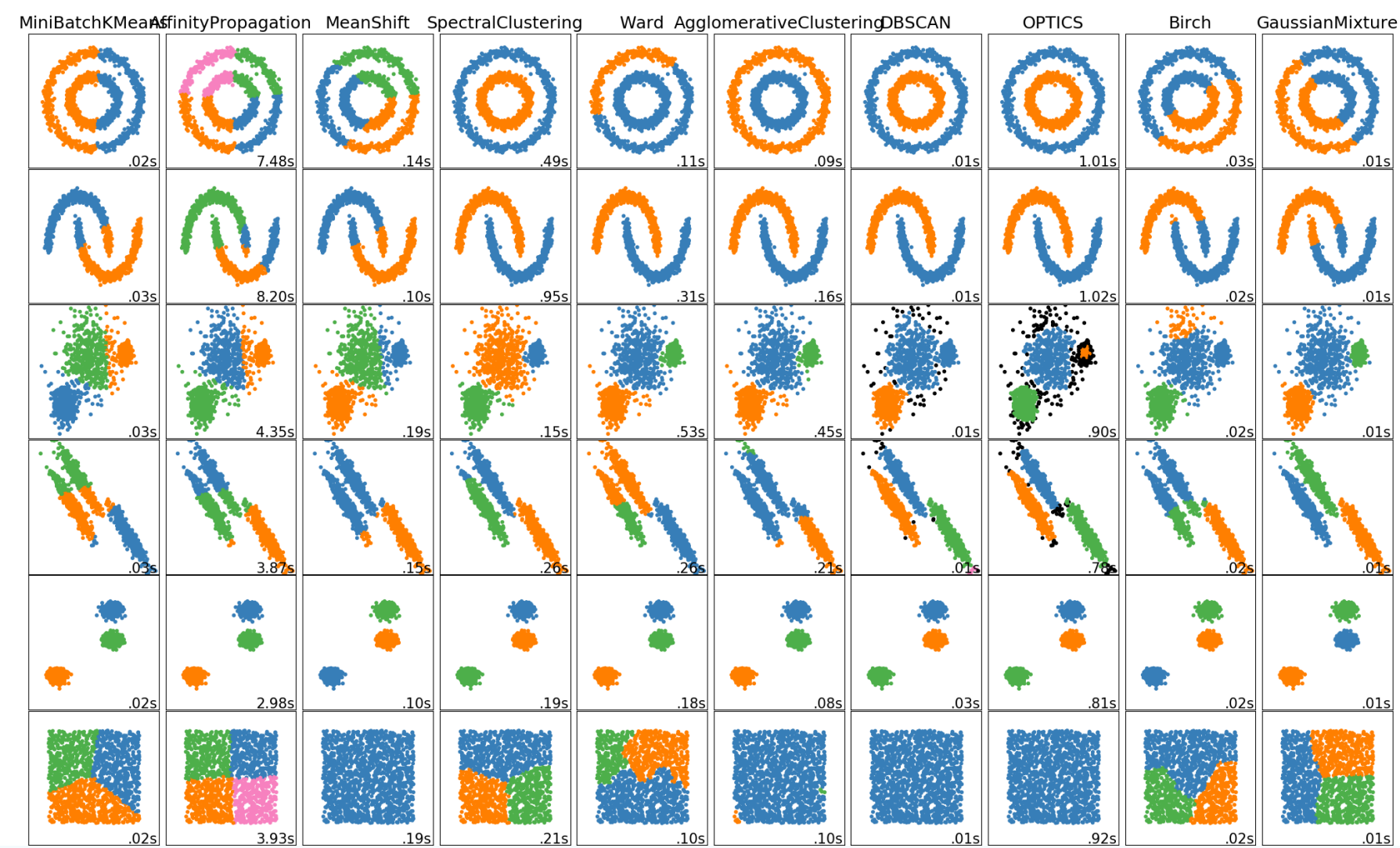


m – число соседей
ε – радиус окрестности





Сравнение алгоритмов кластеризации на модельных данных



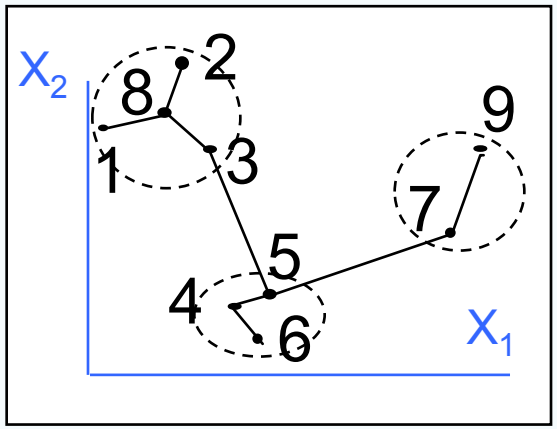
https://scikit-learn.org/stable/auto_examples/cluster/plot_cluster_comparison.html

Проблемы алгоритмов кластеризации

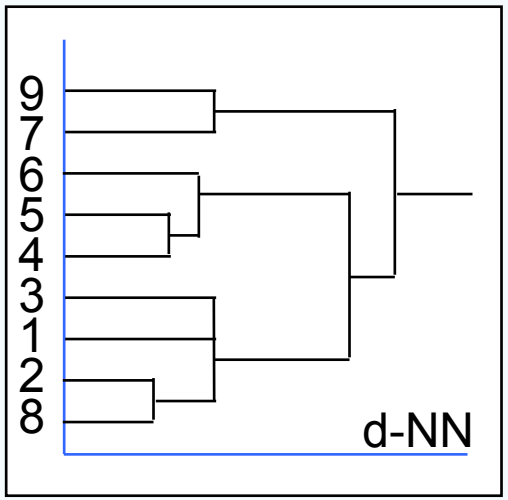
- **Начальный выбор центров кластеров может существенно влиять на окончательный результат**
  - Повтор кластер-анализа для различной инициализации
  - Инициализация с использованием других алгоритмов (например, сеть Кохонена)
- **Количество кластеров задается из априорных соображений**
  - Перебор по числу кластеров
  - Процедуры адаптивного поиска оптимального числа кластеров
    - ✓ дивизимные (разделяются самые большие кластеры)
    - ✓ агломеративные (объединяются самые близкие кластеры)
- **Шум может существенно влиять на результат**
  - Специальные процедуры отсева выбросов

Поиск оптимального числа кластеров по порогу

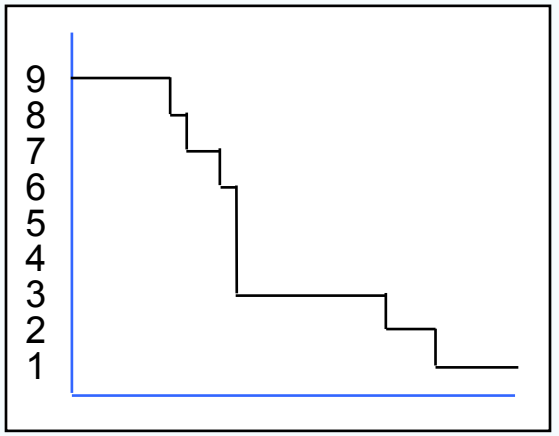
- **Порог можно искать для межкластерных расстояний, средних дисперсий и других оценок качества кластеризации**
  - Ниже оптимального порога - частое (естественное) укрупнение кластеров
  - Выше оптимального порога - редкое (вынужденное) укрупнение кластеров



Граф ближ.соседства



Дендрограмма



Изменение порога

Метрики оценки качества кластеризации

- Среднее дисперсий кластеров, взвешенных по объемам

$$J(K) = \frac{1}{K}\sum_{i=1}^{K}\frac{1}{N_i}\sum_{x\in C_i}\left\|\vec{x}-\vec{z}_i\right\|^2 \qquad z_i = \frac{1}{N_i}\sum_{x\in C_i}\vec{x}$$

- Индекс Davies-Bouldin (для ***K>1***)

$$DB(K) = 1/K\cdot\left(\sum_{h=1}^{K}R_h\right), \text{ где } R_h = \max_{j\neq h}\left(e_h + e_j\right)/d_{hj}$$

e_h – дисперсия расстояний внутри кластера ***h***

d_{hj} – Евклидово расстояние между кластерами ***h*** и ***j***

- Индекс Данна (Dunn index)

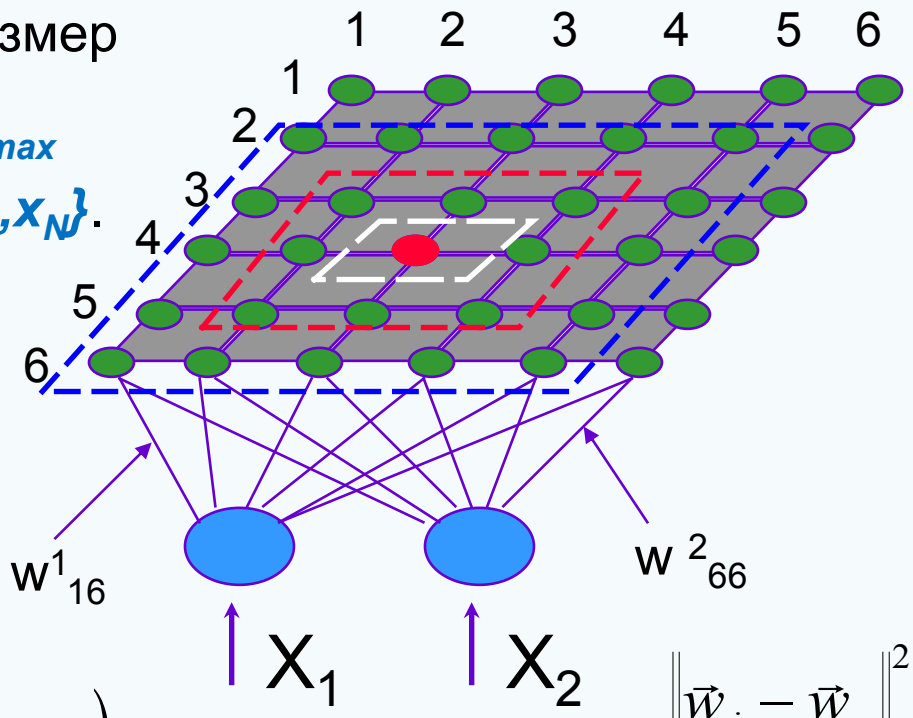
$$DI_m = \min_{1\leqslant i\leqslant m}\left\{\min_{1\leqslant j\leqslant m, j\neq i}\left\{\frac{\delta(C_i,C_j)}{\max\limits_{1\leqslant k\leqslant m}\Delta_k}\right\}\right\}$$

$\delta(C_i, C_j)$ – расстояние между кластерами ***i*** и ***j***

Δ_k – диаметр кластера ***k***

Нейронная сеть Кохонена

Teuvo Kohonen, 1982



Шаг 1. Инициализировать веса малыми случайными значениями. Задать размер окрестности $\sigma(0)$, скорость $\eta(0)$ и t_{max}

Шаг 2. Задать значения входов $\{x_1,\ldots,x_N\}$.

Шаг 3. Вычислить расстояния до всех нейронов $d_k = \sum_{i=1}^{N} \left(x_i - w_{ki}\right)^2$

Шаг 4. Найти нейрон-победитель $p: d_p < d_k \; \forall k \neq p$

Шаг 5. Подстроить веса победителя и его соседей $\Delta w_{ji} = \eta(t) R(t) \left(x_i - w_{ji}\right)$

$$R(t) = \exp\left(-\frac{\left\|\vec{w}_j - \vec{w}_p\right\|^2}{2\sigma^2(t)}\right)$$

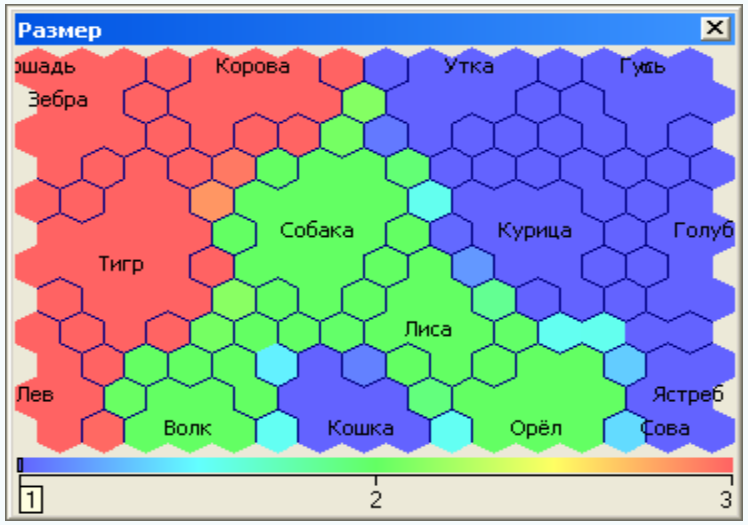
Шаг 6. Обновить размер окрестности $R(t)$ и скорость $\eta(t)$

$$\sigma(t) = \sigma(0)(1 - t / t_{max}) \qquad \eta(t) = \eta(0)(1 - t / t_{max})$$

Шаг 7. Если $(t < t_{max})$, то Шаг 2, иначе СТОП

Как правило, **K << P**

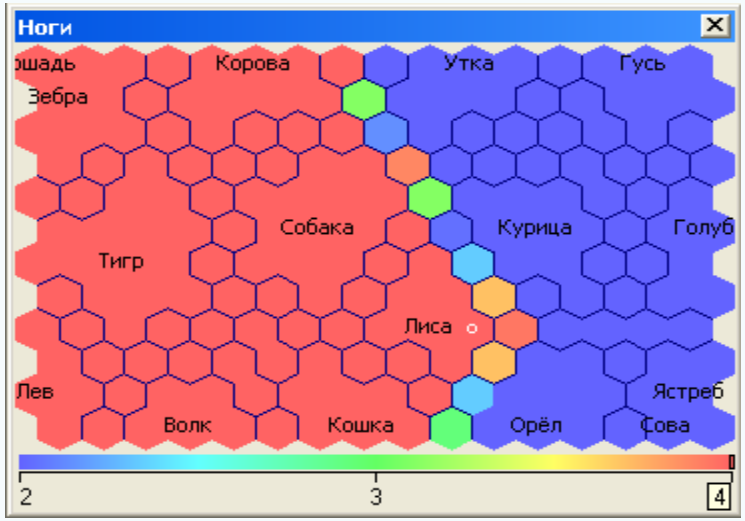
Самоорганизующаяся карта Кохонена

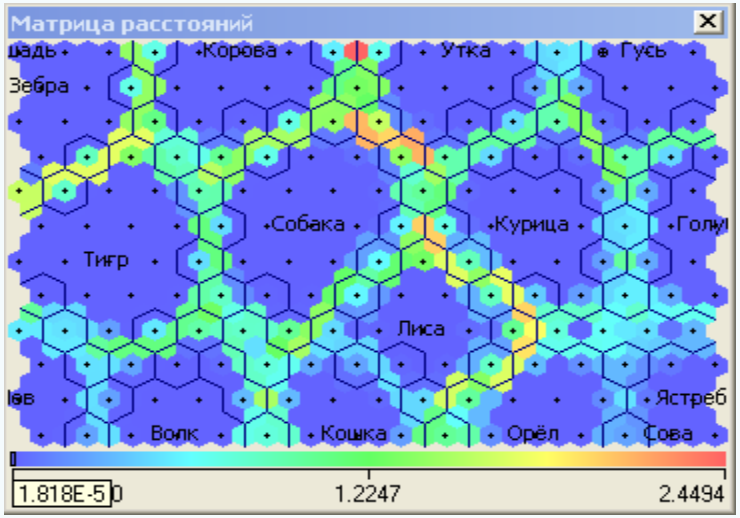


Используется для наглядной двумерной визуализации многомерных данных

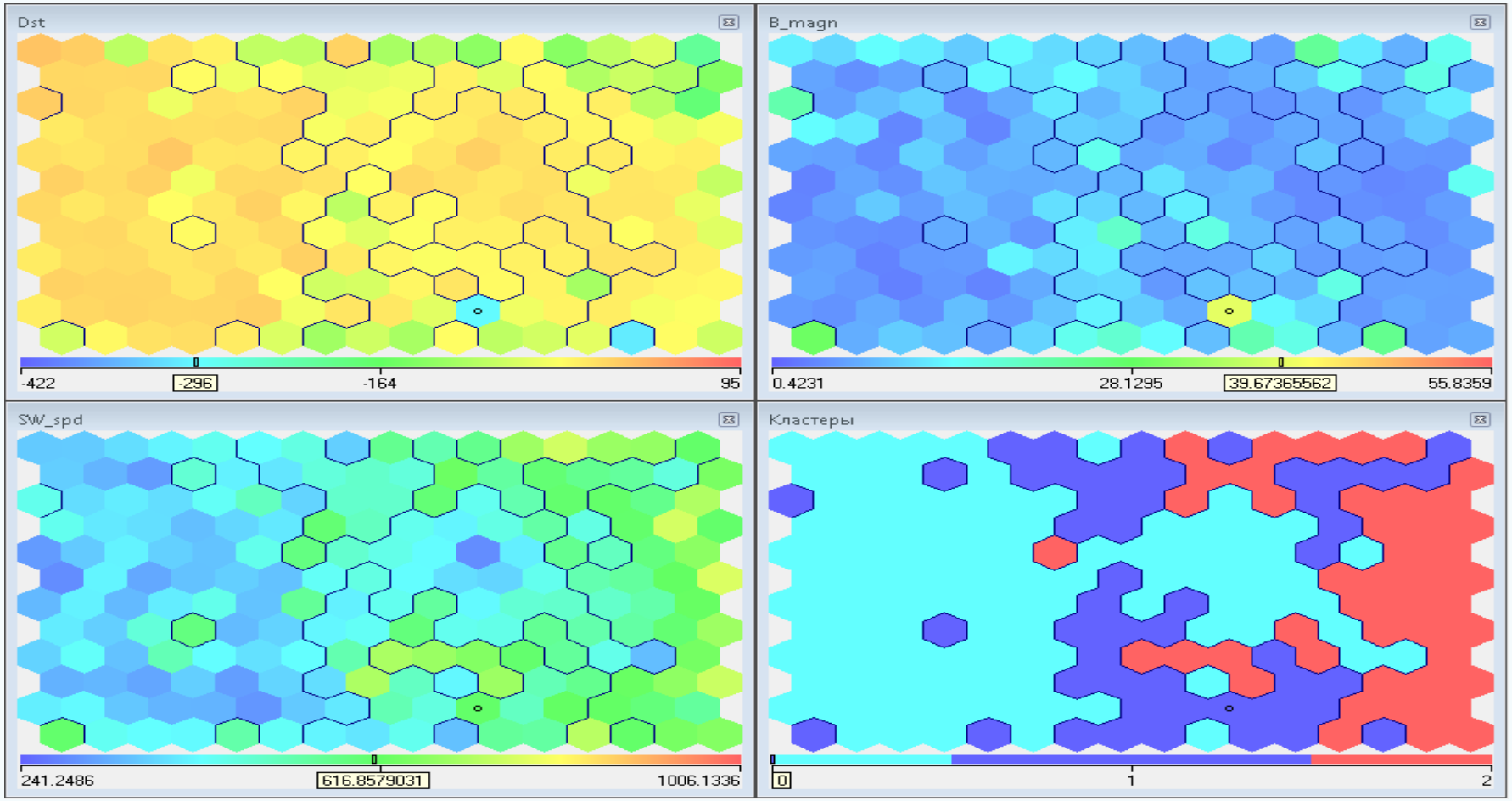
K >> P
K ~ P
K << P

Пример «Животные»





Исследование геомагнитного индекса Dst

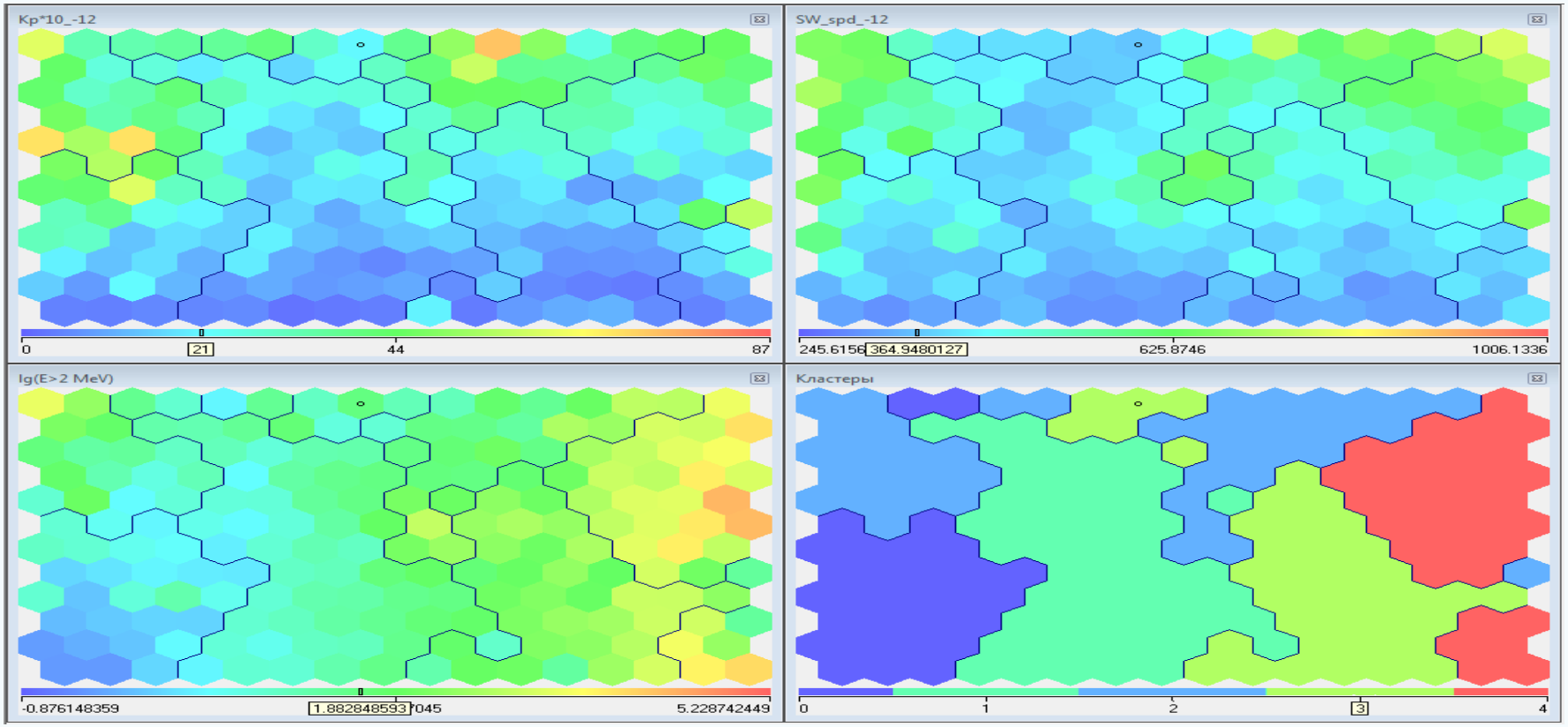


0 – Голубой – спокойный период

1 – Синий – главная фаза магнитной бури

2 – Красный – фаза восстановления

Исследование потока электронов внешнего радиационного пояса Земли



0 – минимум солнечной активности
1 – главная фаза магнитной бури
2 – состояние внешнего РПЗ в отсутствие бури
3 – окончание фазы восстановления
4 – максимальный рост потока электронов РПЗ

Нейронная сеть и самоорганизующиеся карты Кохонена: некоторые итоги

- **Преимущества**
  - Построение компактной модели данных, задаваемой статистически
  - Быстрое обучение (по сравнению с многими видами нейронных сетей)
  - Возможность подстройки обученной сети при изменении статистики
- **Недостатки**
  - Заданность топологии и некоторая зависимость от неё
  - Зависимость от инициализации весов
  - Желателен большой тренировочный набор
- **В каких случаях используется**
  - Задачи кластеризации, компрессии
  - Визуализация многомерных данных
  - Высокая размерность данных
  - Большой объём данных

Акад. Teuvo Kohonen
(1934 – 2021)

Спасибо за внимание!

Лаборатория адаптивных методов обработки данных

Учебный курс

«Машинное обучение в физике»

Занятие №7 (лекция).

Базовые методы машинного обучения.

Авторы курса:

С.А. Доленко, И.М. Гаджиев, А.О. Ефиторов, И.В. Исаев, В.Р. Широкий

Линейная регрессия

Один из самых простых алгоритмов машинного обучения. Моделируем зависимость целевой переменной от p признаков в виде:

$$\hat{y} = \sum_{j=1}^{p} w_i x_i + w_0$$

Пусть X – тренировочный набор (в строках находятся примеры X_i.

$$X = \begin{pmatrix} 1 & X_0 \\ \dots & \dots \\ 1 & X_N \end{pmatrix}$$

Формулу можно записать в матричном виде:

$$\hat{Y} = Xw$$

Линейная регрессия (получение весов)

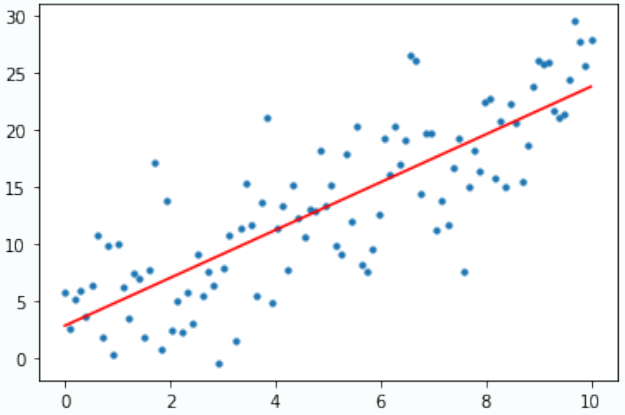
Обозначим за Y ответы примеров тренировочного набора

Для получения весов можно воспользоваться методом наименьших квадратов:

$$L(w) = (Y - Xw)^r (Y - Xw)$$

Условие минимума L(w) – равенство нулю производной, откуда можно получить выражение для весов в задаче линейной регрессии:

$$w = (X^T X)^{-1} X^T Y$$



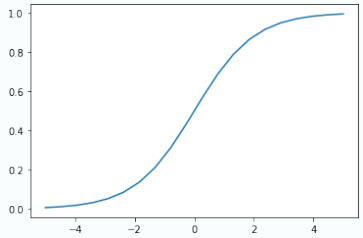
Линейная регрессия. Замечания

- Метод наименьших квадратов дает оптимальную оценку параметров линейной регрессии согласно критерию максимального правдоподобия;
- Перед использование линейной регрессии лучше провести масштабирование данных, т.к. признаки могут иметь разную размерность;
- Для подбора параметров можно использовать и MAE – для этого нужно применить численные методы;
- Линейная регрессия легко превращается в полиномиальную – достаточно добавить столбцы с соответствующими степен признаков.

Логистическая регрессия

Линейная регрессия плохо подходит для решения задачи классификации, т.к. получаемые значения неограничены. Для получения значений в диапазоне от [0,1] и моделирования вероятности принадлежности к классу 0 или 1 используется логистическая регрессия:

$$\hat{y} = \sigma(z) = \frac{1}{1 + e^{-z}} \qquad z = \sum_{j=1}^{p} w_i x_i + w_0$$



Для выбора оптимальных весов w обычно максимизируют (например, методом Ньютона) логарифм правдоподобия

$$L = \sum_{i=1}^{N} y_i \log(\hat{y}_i)$$

Логистическая регрессия. Замечания

- Перед использование логистической лучше провести масштабирование данных, т.к. признаки могут иметь разную размерность;
- Линейная логистическая регрессия легко превращается в полиномиальную – достаточно добавить столбцы с соответствующими степен признаков;
- Многомерный вариант логистической регрессии – использование функции softmax

$$softmax(x) = \frac{e^{x_i}}{\sum e^{x_i}}$$

Регуляризация Lp

Проблема переобучения – модель имеет слишком большое число параметров – в результате получаем решения с «зашумленными» значениями коэффициентов

Регуляризация Lp

Проблема переобучения – модель имеет слишком большое число параметров – в результате получаем решения с «зашумленными» значениями коэффициентов

Решение – использование регуляризации – подавления весов. Заставляем модель выбирать веса максимально «экономным» способом, штрафуя за большие величины.

Для этого модифицируем функцию потерь L

Общий вид Lp регуляризации:

$$L_1 = L(w) + \lambda \sum_{j=1}^{s} |w_j|^p$$

Регуляризация L2 (Ridge)

P=2, в качестве добавки к функции потерь используем сумму квадратов весов

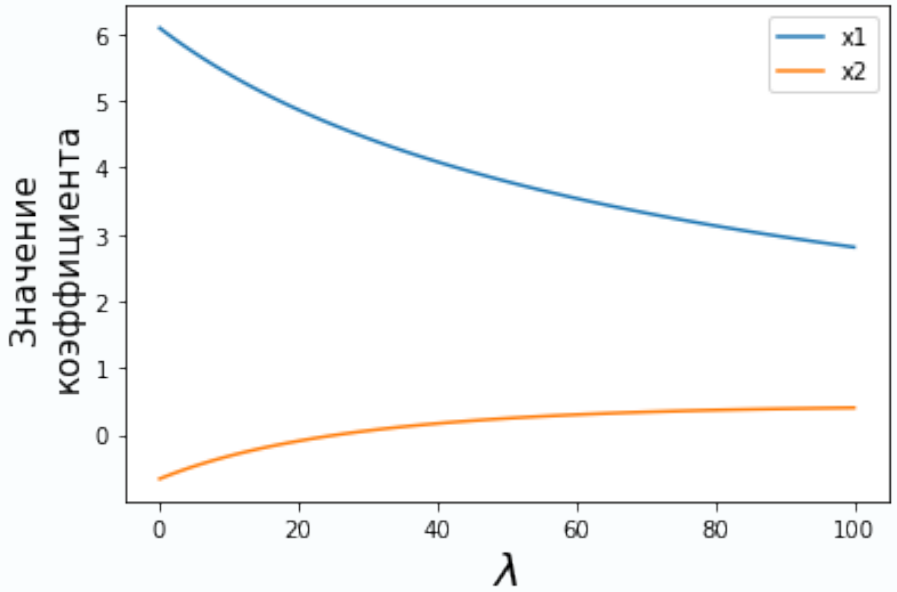
$$L_1 = L(w) + \lambda \sum_{j=0}^{s} |w_j|^2$$

В случае, если в качестве исходной модели используем линейную регрессию (в таком случае модель обычно называют гребневой регрессией), а в качестве функционала ошибки – сумму квадратов остатков, выражение для весов принимает вид:

$$w = (X^T X + \lambda Y)^{-1} X^T Y$$

Регуляризация L2 (Ridge)

Задача линейной регрессии с двумя скоррелированными признаками – смотрим на значения коэффициентов при увеличении параметра регуляризации. Характерное поведение – коэффициенты плавно либо убывают, либо возрастают. Чем больше параметр – тем больший вес имеет их сумма квадратов

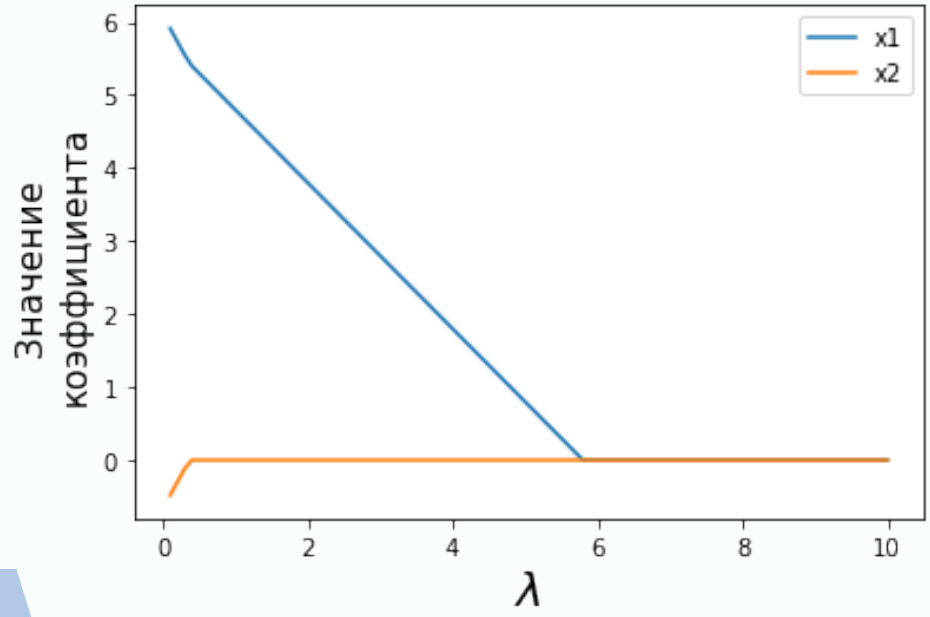


Регуляризация L1 (Лассо)

P=2, в качестве добавки к функции потерь используем сумму квадратов весов. Решаем численными методами

$$L_1 = L(w) + \lambda \sum_{j=0}^{s} |w_j|$$

Пример из предыдущего слайда

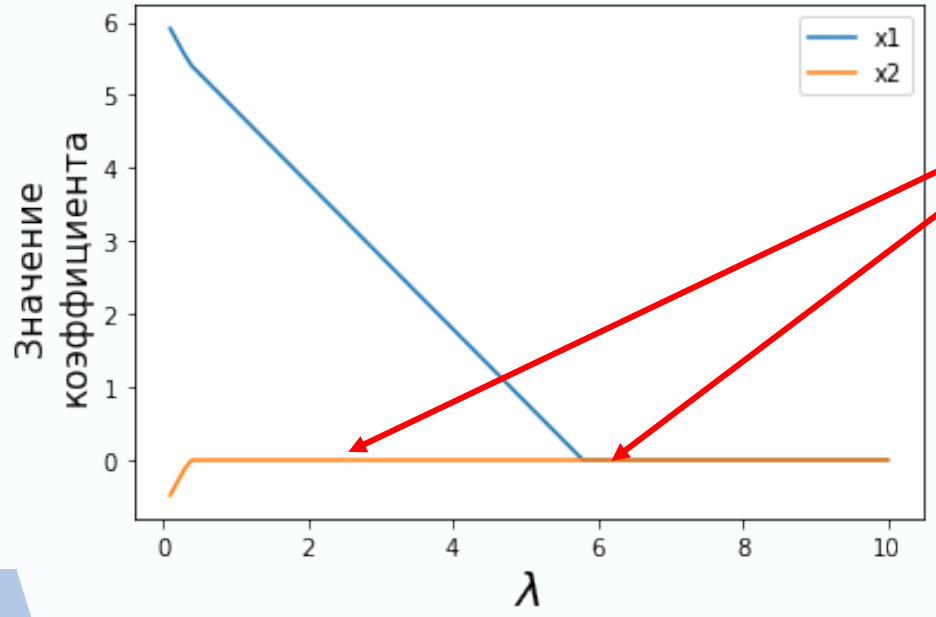


Регуляризация L1 (Лассо)

P=1, в качестве добавки к функции потерь используем сумму квадратов весов. Решаем численными методами

$$L_1 = L(w) + \lambda \sum_{j=1}^{s} |w_j|$$

Пример из предыдущего слайда



Начиная с некоторого значения параметра коэффициенты обращаются в 0

Регуляризация L1 (Лассо)

Альтернативная формулировка для L1 регуляризации:

$$w = argmin\big(L(w)\big)$$

$$\sum_{j=0}^{s} |w_j| \quad \leq C$$

Ищем первое пересечение красных линий с синим ромбом. Часто оказывается, что оно находится на угле ромба, где один из коэффициентов в точности равен 0!

Регуляризация: Замечания

- L2 регуляризация позволяет уменьшить переобучение модели и производит уменьшения весов «плохих» признаков;
- L2 регуляризация позволяет уменьшить веса «плохих» признаков или даже полностью занулить их. Таким образом осуществляется мягкий отбор признаков;
- Иногда используют комбинацию L1 и L2 регуляризации – Elastic Net (сумма соответствующих добавок). Позволяет сочетать плавное уменьшение и полное зануление «плохих» признаков;
- Параметры регуляризации следует подбирать индивидуально для каждой задачи!

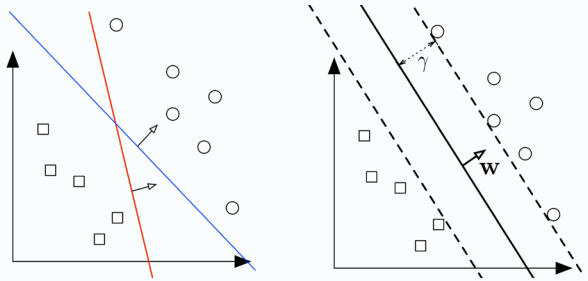
Метод опорных векторов

Главная идея – провести такую разделяющую гиперповерхность в пространстве признаков, которая максимально удалена от соседних примеров из противоположных классов (имеет максимальный отступ).

Такую задачу можно формализовать как поиск такой гиперплоскости, что расстояние до ближайшего примера должно быть максимально и при этом гиперплоскость должна правильно разделять примеры на классы.

Выражение для отступа, который должен быть максимизирован:

$$M = \min_{i=1..n} \frac{y_i(wx + w_0)}{|w|}$$



https://www.cs.cornell.edu/courses/cs4780/2018fa/lectures/lecturenote09.html

Метод опорных векторов: Оптимизация

Лагранжева формулировка для нахождения разделяющей гиперплоскости ({y}={-1,1}):

$$\begin{cases} \min\limits_{w,w_0,\zeta}\left(\frac{1}{2}|w|^2 + C\sum\limits_{i=1}^{n}\zeta_i\right) \\ y_i(x_i{}^T w + w_0) \geq 1 - \zeta_i; \zeta_i \geq 0 \end{cases}$$

Используем ζ_i т.к нет гарантии линейной разделимости классов

Дуальная формулировка проблемы (квадратичное программирование):

$$\begin{cases} \max\limits_{\alpha}\left(\frac{1}{2}\sum\limits_{i=1}^{n}\alpha_i - \sum\limits_{i=1}^{n}\sum\limits_{j=1}^{n}\alpha_i\alpha_j y_i y_j x_i^T x_j\right) \\ 0 \leq \alpha_i \leq C; \sum\limits_{i=1}^{n}\alpha_i\, y_j = 0 \end{cases} \longrightarrow \alpha_i{}^* \longrightarrow w = \sum_{i=1}^{n}\alpha_i{}^* x_i y_i$$

Метод опорных векторов: Hinge Loss

Лагранжева формулировка для нахождения разделяющей гиперплоскости ({y}={-1,1}):

$$\begin{cases} \min\limits_{w,w_0,\zeta}\left(\frac{1}{2}|w|^2 + C\sum\limits_{i=1}^{n}\zeta_i\right) \\ y_i({x_i}^T w + w_0) \geq 1 - \zeta_i; \zeta_i \geq 0 \end{cases}$$

Эквивалентная формулировка в виде функции потерь (Hinge Loss) и регуляризации:

$$L = \frac{1}{C}|w|^2 + \sum_{i=1}^{n}\max(0, 1 - y_i\,({x_i}^T w + w_0))$$

Можно пытаться минимизировать данную функцию потерь численно

Метод опорных векторов: Kernel Trick

Что если мы хотим построить нелинейную разделяющую границу? Для этого можно нелинейно отобразить исходные вектора примеров в некоторое пространство, в котором будет строиться линейная граница. Выберем в качестве такого отображения h(x). Дуальная проблема:

$$\max_{\alpha}\left(\frac{1}{2}\sum_{i=1}^{n}\alpha_i - \sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j h(x_j)^T h(x_j)\right.$$

Тогда оптимальные веса можно записать в виде:

$$w = \sum_{i=1}^{n}\alpha_i{}^* h(x_i) y_i$$

Общее решение:

$$f(x) = h(x)^T w + w_0 = \sum_{i=1}^{n}\alpha_i{}^* h(x)^T h(x_i) y_i + w_0$$

Метод опорных векторов: Kernel Trick

Что если мы хотим построить нелинейную разделяющую границу? Для этого можно нелинейно отобразить исходные вектора примеров в некоторое пространство, в котором будет строиться линейная граница. Выберем в качестве такого отображения h(x). Дуальная проблема:

$$\max_{\alpha}\left(\frac{1}{2}\sum_{i=1}^{n}\alpha_i - \sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j h(x_j)^T h(x_j)\right.$$

Тогда оптимальные веса можно записать в виде:

$$w = \sum_{i=1}^{n}\alpha_i{}^* h(x_i) y_i$$

Общее решение:

$$f(x) = h(x)^T w + w_0 = \sum_{i=1}^{n}\alpha_i{}^* h(x)^T h(x_i) y_i + w_0$$

Достаточно определить $K(x, x') = h(x)^T h(x')$!

Метод опорных векторов: Kernel Trick

Что если мы хотим построить нелинейную разделяющую границу? Для этого можно нелинейно отобразить исходные вектора примеров в некоторое пространство, в котором будет строиться линейная граница. Выберем в качестве такого отображения h(x). Дуальная проблема:

$$\max_{\alpha}\left(\frac{1}{2}\sum_{i=1}^{n}\alpha_i - \sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j K(x_j, x_j)\right.$$

Тогда оптимальные веса можно записать в виде:

$$w = \sum_{i=1}^{n}\alpha_i{}^* h(x_i) y_i$$

Общее решение:

$$f(x) = h(x)^T w + w_0 = \sum_{i=1}^{n}\alpha_i{}^* K(x_j, x_j) + w_0$$

Достаточно определить $K(x, x') = h(x)^T h(x')$!

Метод опорных векторов: Kernel Trick

Для построения нелинейной разделяющей границы достаточно выбрать ядро $K(x_j, x_j)$, имеющее смысл выражения для скалярного произведения преобразованных векторов. Само преобразование выбирать не нужно.

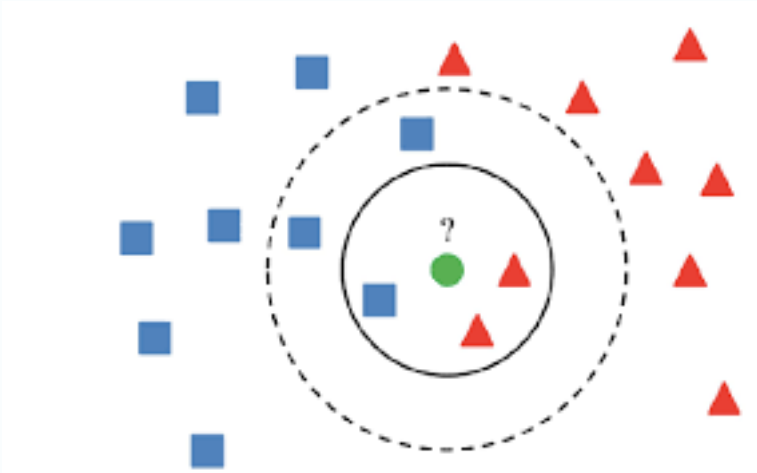
$$\max_{\alpha}\left(\frac{1}{2}\sum_{i=1}^{n}\alpha_i - \sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j K(x_j, x_j)\right)$$

$$f(x) = \sum_{i=1}^{n}\alpha_i{}^{*} K(x_j, x_j) + w_0$$

- Линейное ядро $K(x, x') = x^T x'$
- Полиномиальное ядро $K(x, x') = (1 + x^T x')^{\,d}$
- Ядро на радиально-базисных функциях $K(x, x') = \exp(-\gamma|x - x'|^2)$

Метод K ближайших соседей

Один из самых простых алгоритмов. Пусть дан тренировочный набор X, y и новый пример x'. Тогда ответ для этого примера можно вычислить, взяв K ближайших примеров к x' по некоторой метрике (например, евклидово расстояние) и усреднив их ответы



https://vitalflux.com/k-nearest-neighbors-explained-with-python-examples/

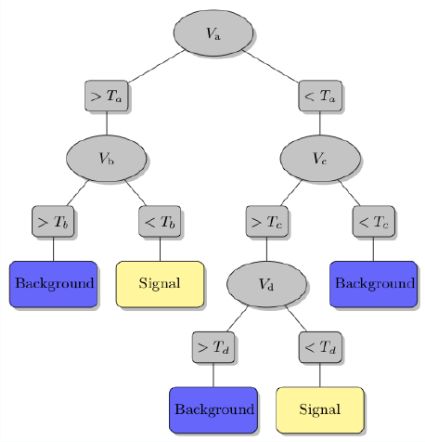
Метод K ближайших соседей

Метод имеет следующие гиперпараметры:

- Количество соседей K
- Метрика, по которой считается расстояние
- Вес каждого примера в окрестности (например, можно брать веса, обратно пропорциональные расстоянию

Дерево решений

Алгоритм классификации имитирует «человеческую» логику принятия решений. Для получения ответа на примере x необходимо рекурсивно пройтись по дереву, каждый узел которого представляет собой развилку на основе критерия $I[x_j>C]$ (где x_j – j-й признак примера). Финальный ответ определяется в конечном узле дерева путем усреднения по ответам в этом узле. Пример дерева для отделения фона от сигнала:



https://www.researchgate.net/figure/Schematic-of-a-decision-tree-Decision-nodes-for-each-event-variable-V-a-V-b-V-c_fig2_236935633

Построение дерева решений

Построение дерева решений – рекурсивный процесс. Необходимо выбрать признак и разбиение по нему для исходного тренировочного набора. затем проделать то же самое для двух получившихся частей и так далее. Для проведения этой процедуры необходимы следующие определения:

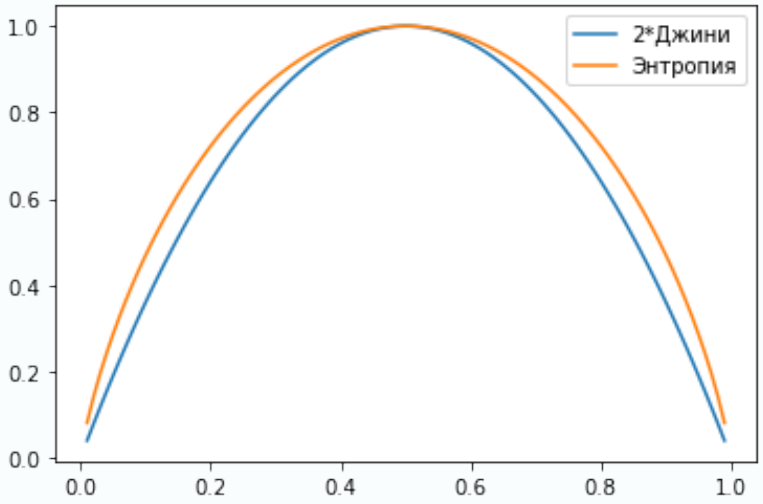
- Метрика неоднородности ответов выборки
- Сравнение однородности подвыборок, получившихся путем разбиения
- Критерий остановки построения дерева

Энтропия и Индекс Джини

Определимся с метриками неоднородности

Пусть p_k- доля примеров выборки, принадлежащих к классу k (полное число классов K)

- Индекс Джини = $\sum_{k=1}^{K} p_k(1-p_k)$
- Энтропия = $-\sum_{k=1}^{K} p_k \log(p_k)$



Критерий выбора разбиения

Для осуществления разбиения выборки по признаку необходима пара (j,C) – где j – номер признака, C – значение, по которому производится разбиение. Тогда разбиение можно записать в виде $I[x_j>C]$.

S(X) – критерий неоднородности выборки (энтропия, Джини, ..).

Изменение неоднородности в результате разбиения (или прирост информации):

$$IG = S(X_0) - \frac{N_1}{N_0} S(X_1) - \frac{N_2}{N_0} S(X_2)$$

Где X_0 - исходная выборка, N_0 - размер исходной выборки, $X_{1,2}$- выборки, полученные в результате разбиения

Алгоритм построения дерева решений

Упрощенная рекурсивная версия:

Для узла дерева:

1. Для каждого возможного разбиения каждого признака посчитать прирост информации
2. Выбрать разбиение соответствующий наибольшему приросту
3. Если прирост меньше порога – остановиться
4. Создать два дочерних узла соответствующих разбиению
5. Для каждого узла повторить процедуру

Алгоритм построения дерева решений

~~Упрощенная рекурсивная версия:~~

~~Для узла дерева:~~

~~1. Для каждого возможного разбиения каждого признака посчитать прирост информации~~

~~2. Выбрать разбиение соответствующий наибольшему приросту~~

~~3. Если прирост меньше порога – остановиться~~

~~4. Создать два дочерних узла соответствующих разбиению~~

~~5. Для каждого узла повторить процедуру~~

Алгоритм построения дерева решений

На деле есть гораздо больше нюансов.

Известные алгоритмы построения деревьев решений:

CART

Breiman L., et al., “Classification and regression trees”, 1984

C4.5

Quinlan, J. R. “C4.5: Programs for Machine Learning”, Morgan Kaufmann Publishers, 1993.

Регуляризация деревьев решений

- Ограничение на максимальную глубину дерева
- Ограничение на количество рассматриваемых признаков в каждом узле (случайно выбираем m признаков из p и рассматриваем их разбиения)
- Ограничение на количество примеров в конечном узле
- Ограничение на количество примеров в разделяемом узле
- Ограничение на максимальное количество конечных узлов дерева

Свойства деревьев решений

- Склонны к переобучению
- Велика доля случайности – при повторных построениях могут получиться разные деревья
- Легко интерпретируются
- Можно приспособить для задачи регрессии, если использовать в качестве метрики MSE

Бутстрэп и бэггинг

Бутстреп

Процедура получения множества выборок (X_1, X_2, X_3, …) из исходной выборки X. Каждая выборка получается путем многократного выбора с повторением из исходной выборки.

Бэггинг

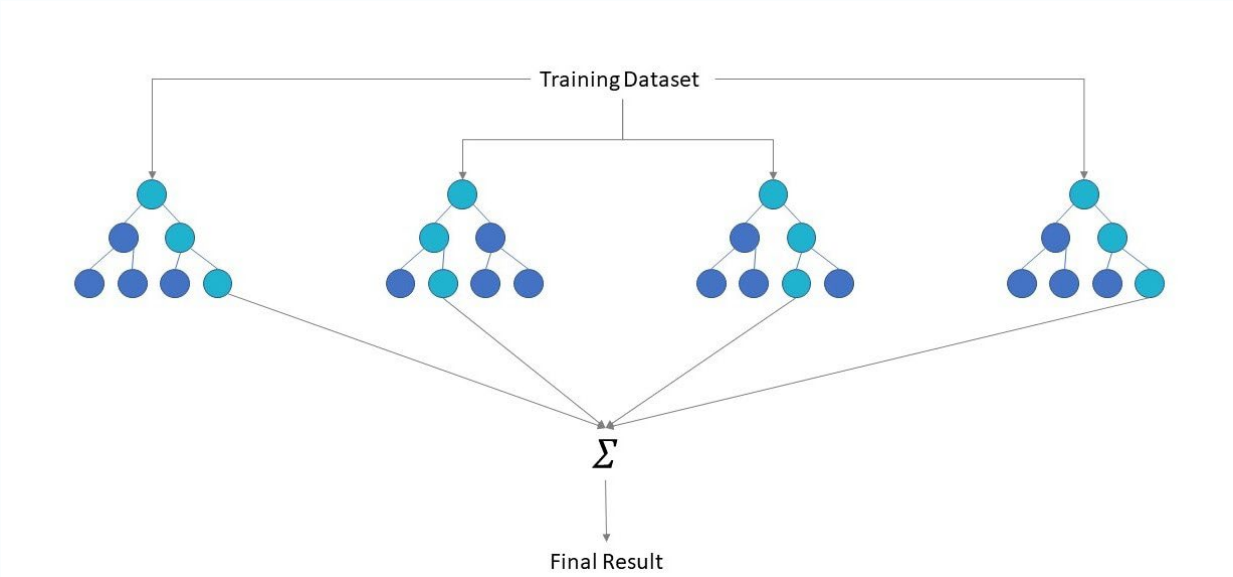
Способ ансамблирования моделей. Обучаем исходную модель на N подвыборках полученных с помощью бутстрэпа. Усредняем предсказания полученных моделей.

Теория подсказывает, что усреднение большого количества некоррелированных моделей уменьшает разброс предсказаний. Надеемся, что бэггинг обеспечивает достаточную некоррелированность

Случайный лес

Случайный лес = Бэггинг над деревьями решений

Алгоритм легко распараллеливается. Качество обычно улучшается с ростом количества деревьев



https://www.ibm.com/cloud/learn/random-forest

Градиентный бустинг

Бустинг – способ ансамблирования, при которым модели итеративно обучаются на остатках предыдущей модели

Один из самых популярных алгоритмов машинного обучения – градиентный бустинг. В качестве базовой модели в этом алгоритме используется дерево решений

Реализации градиентного бустинга:

- XGBoost https://xgboost.readthedocs.io/en/stable/
- CatBoost https://catboost.ai/
- LightGBM https://lightgbm.readthedocs.io/en/v3.3.2/

Спасибо за внимание!

Лаборатория адаптивных методов обработки данных

Учебный курс

«Машинное обучение в физике»

Занятие №8 (лекция).

Многослойные персептроны. Алгоритм обратного распространения ошибки.

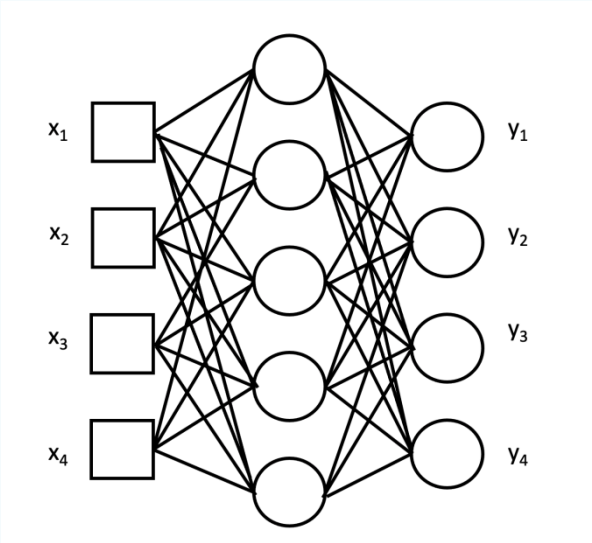
Авторы курса:

С.А. Доленко, И.М. Гаджиев, А.О. Ефиторов, И.В. Исаев, В.Р. Широкий

Многослойный персептрон. Общая схема

Математическая модель восприятия информации мозгом. Предложена в 1958 году Фрэнком Розенблаттом. Персептрон MARK-1 распознавал буквы алфавита с матрицы 20x20

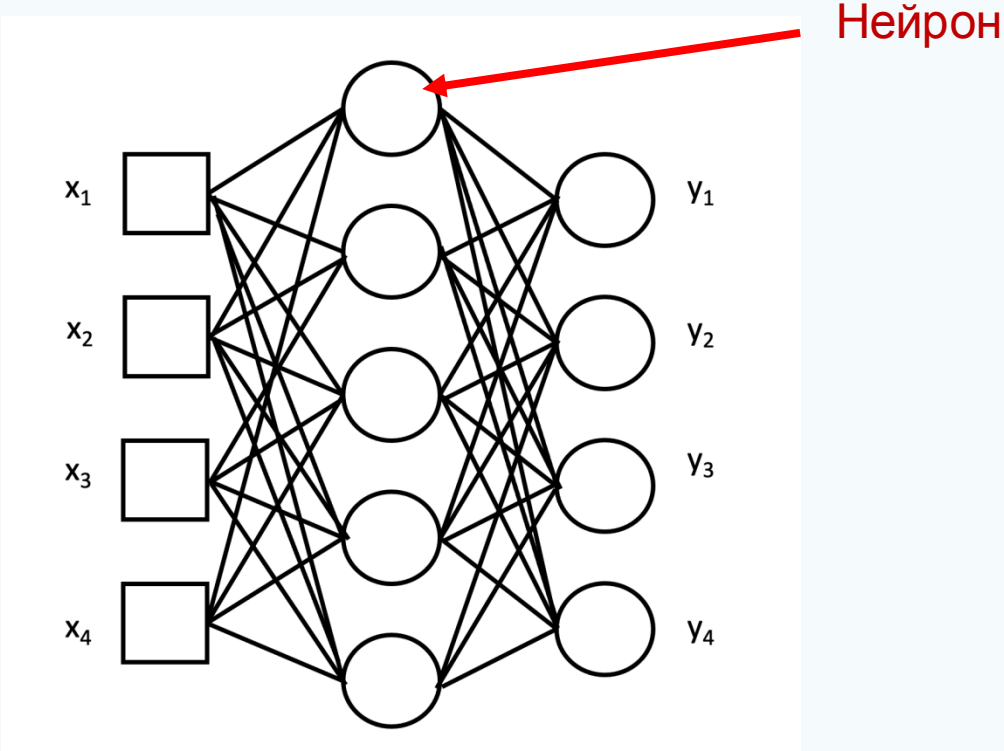
Frank Rosenblatt (1928 – 1971)
https://ru.wikipedia.org/



Rosenblatt, F., “The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain”, Cornell Aeronautical Laboratory, Psychological Review, Vol.65, No. 6, pp. 386-408, 1958.

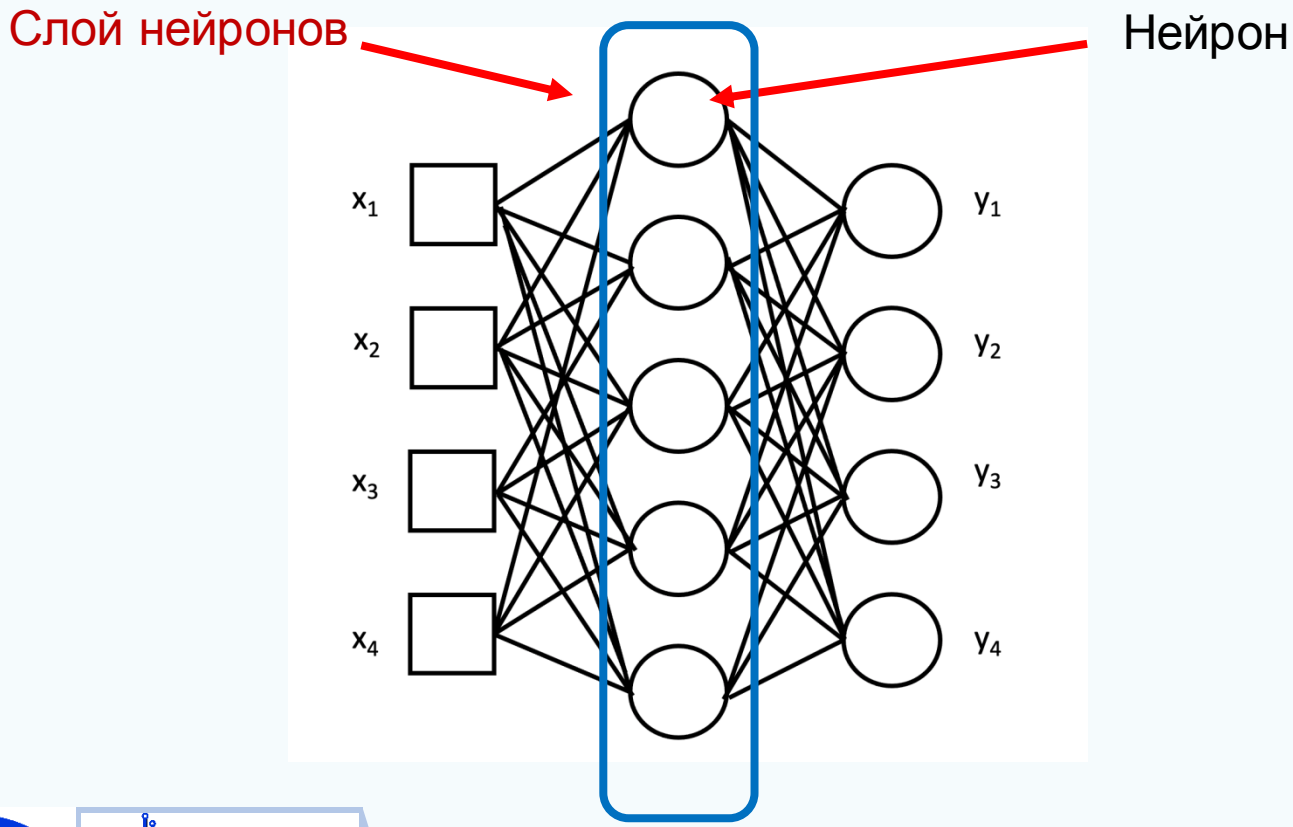
Многослойный персептрон. Общая схема

Базовый вычислительный элемент – математический нейрон. Нейроны организованы в слои. Каждый нейрон получает информацию от всех нейронов предыдущего слоя с помощью связей.



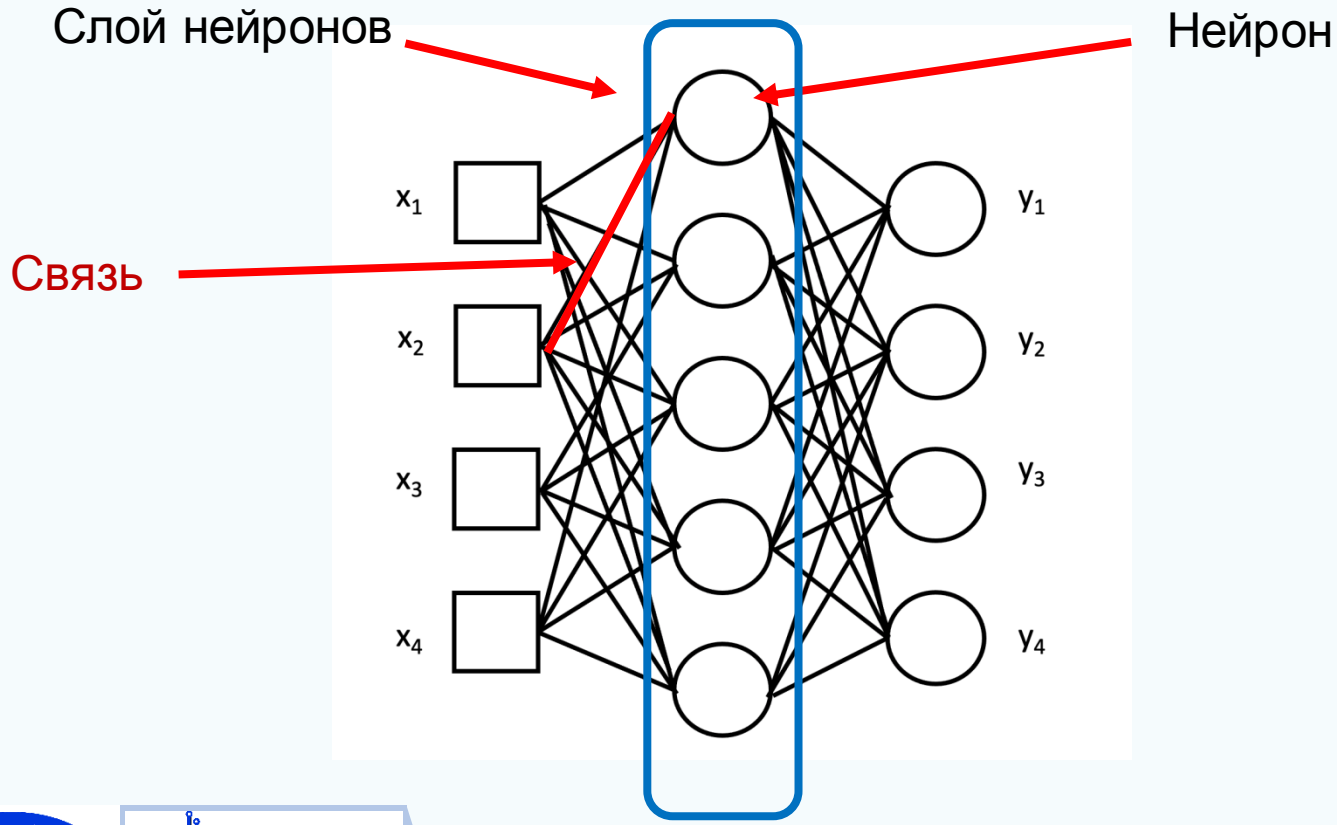
Многослойный персептрон. Общая схема

Базовый вычислительный элемент – математический нейрон. Нейроны организованы в слои. Каждый нейрон получает информацию от всех нейронов предыдущего слоя с помощью связей.



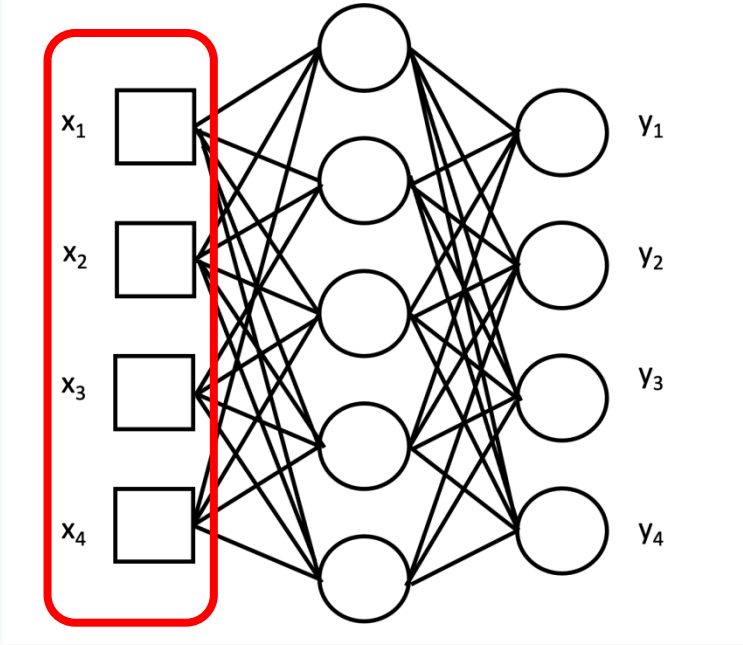
Многослойный персептрон. Общая схема

Базовый вычислительный элемент – математический нейрон. Нейроны организованы в слои. Каждый нейрон получает информацию от всех нейронов предыдущего слоя с помощью связей.



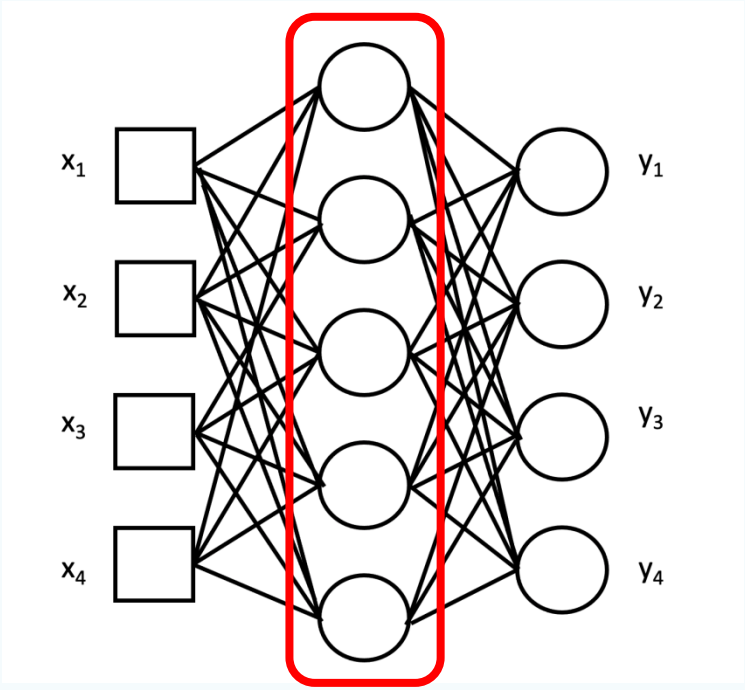
Многослойный персептрон. Общая схема

Вектор признаков $\mathbf{x}_i$ подается на входной слой и обрабатывается первым скрытым слоем (их может быть несколько). На вход каждому следующему слою подаются выходы предыдущего слоя. Результат обработки вектора персептроном – на выходе последнего (выходного слоя).



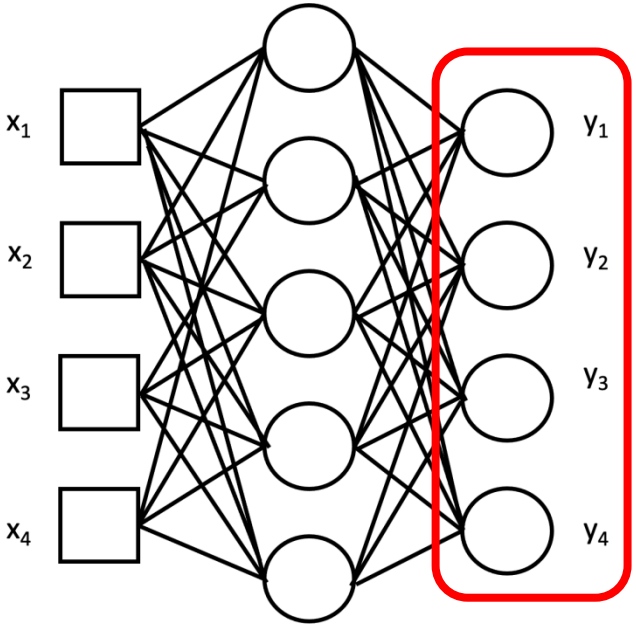
Многослойный персептрон. Общая схема

Вектор признаков $\mathbf{x}_i$ подается на входной слой и обрабатывается первым скрытым слоем (их может быть несколько). На вход каждому следующему слою подаются выходы предыдущего слоя. Результат обработки вектора персептроном – на выходе последнего (выходного слоя).



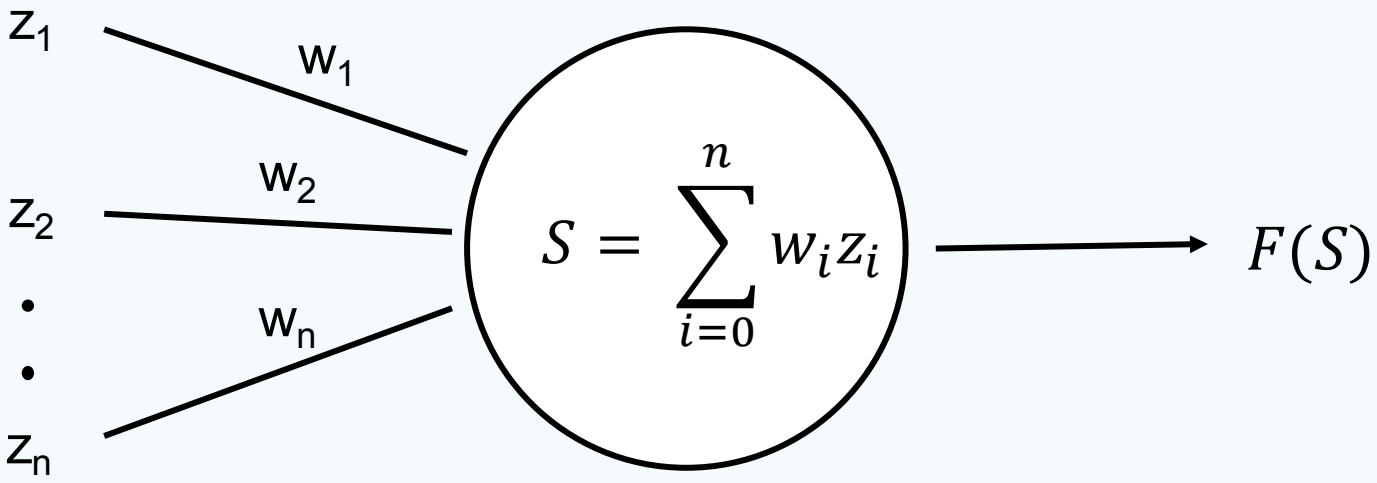
Многослойный персептрон. Общая схема

Вектор признаков $\mathbf{x}_i$ подается на входной слой и обрабатывается первым скрытым слоем (их может быть несколько). На вход каждому следующему слою подаются выходы предыдущего слоя. Результат обработки вектора персептроном – на выходе последнего (выходного слоя).



Формальный нейрон

Математическая модель нейрона предложена Уорреном Мак-Каллоком и Уолтером Питтсом в 1943 г.

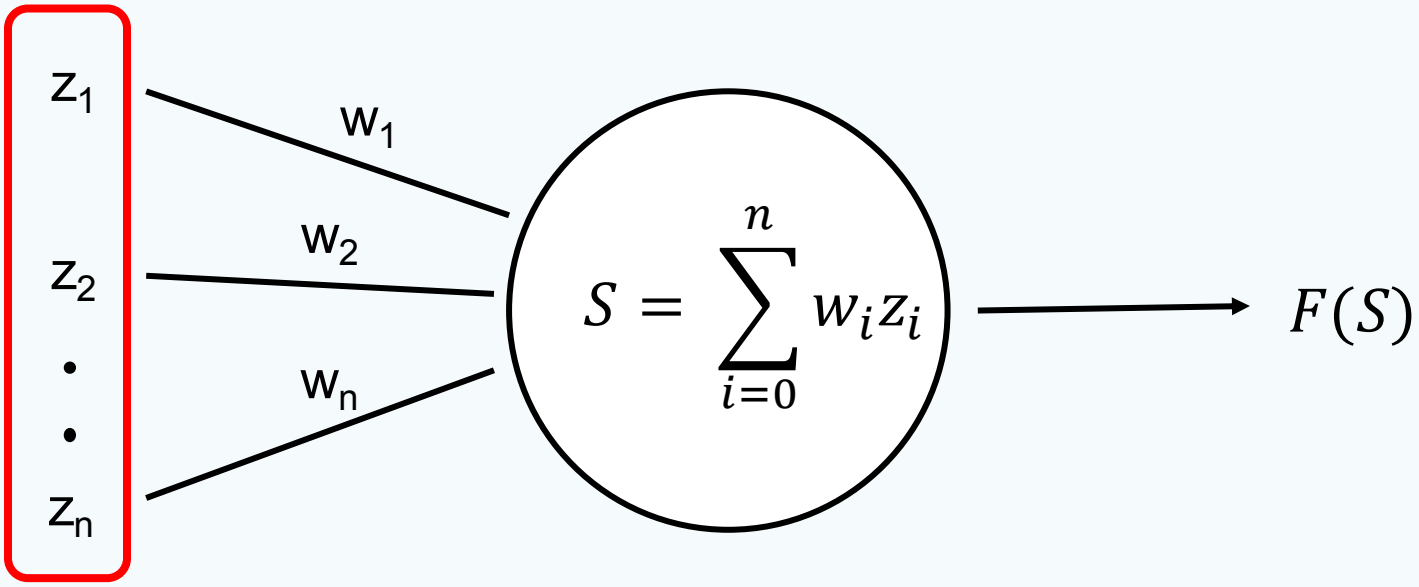


Warren S. McCulloch et al., "A Logical Calculus of the Ideas Immanent in Nervous Activity". In: Bulletin of Mathematical Biophysics, Vol. 5, pp. 115–133, 1943.

Формальный нейрон

На вход нейрона подаются выходы предыдущего слоя - z_i (z_0=1).

Передача происходит через связи, каждой из которых присвоен вес w_i.

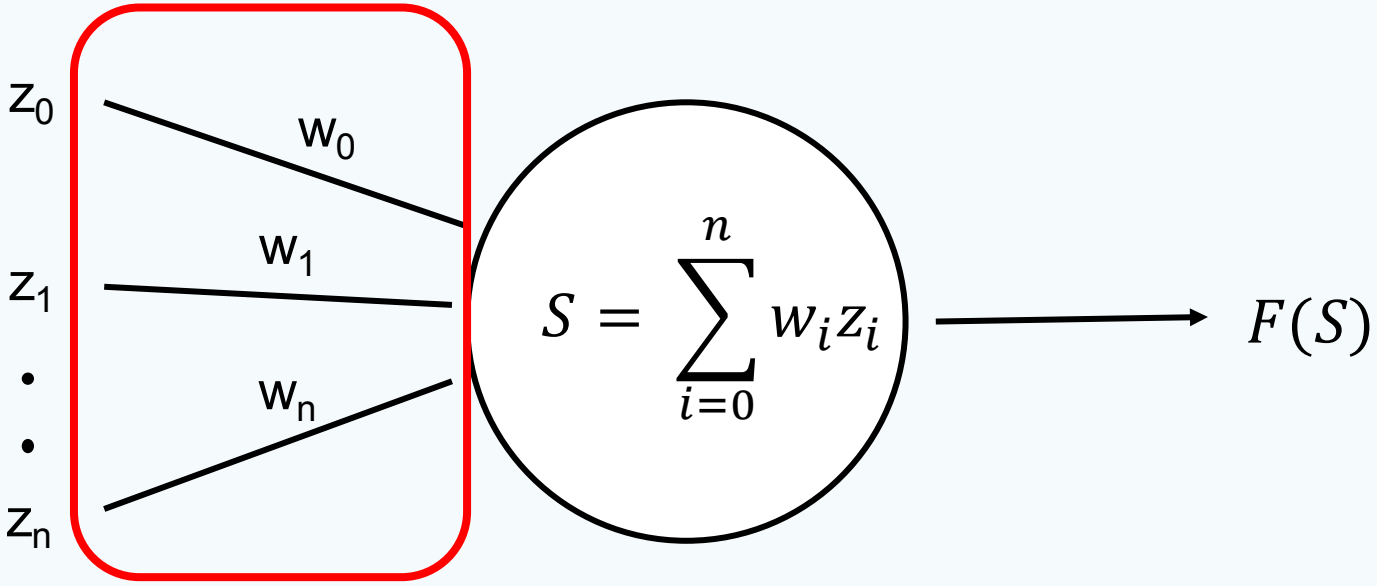


Warren S. McCulloch et al., "A Logical Calculus of the Ideas Immanent in Nervous Activity". In: Bulletin of Mathematical Biophysics, Vol. 5, pp. 115–133, 1943.

Формальный нейрон

На вход нейрона подаются выходы предыдущего слоя - z_i (z_0=1).

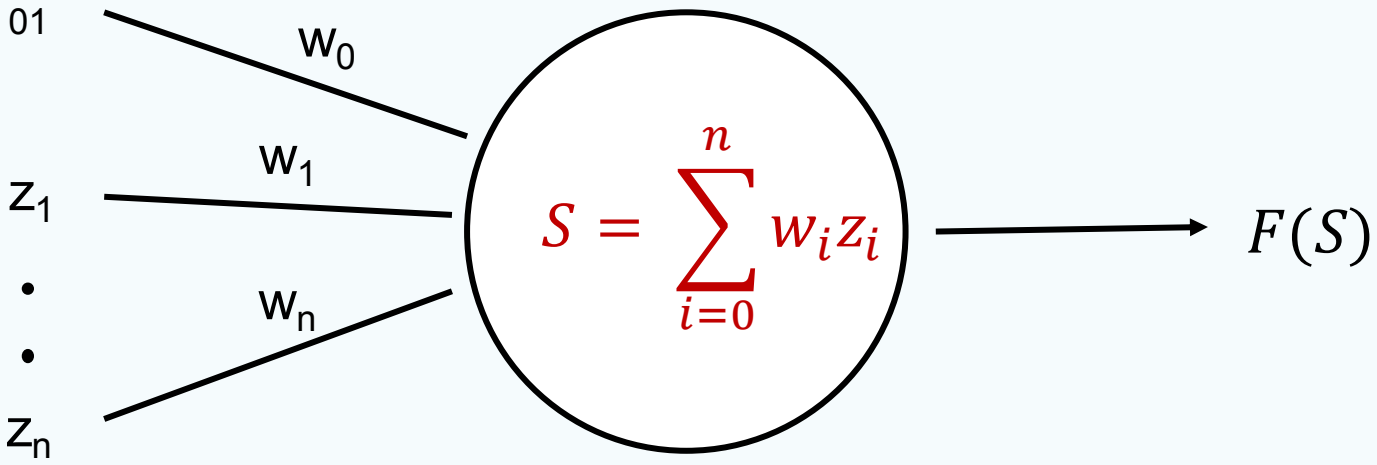
Передача происходит через связи, каждой из которых присвоен вес w_i.



Warren S. McCulloch et al., "A Logical Calculus of the Ideas Immanent in Nervous Activity". In: Bulletin of Mathematical Biophysics, Vol. 5, pp. 115–133, 1943.

Формальный нейрон

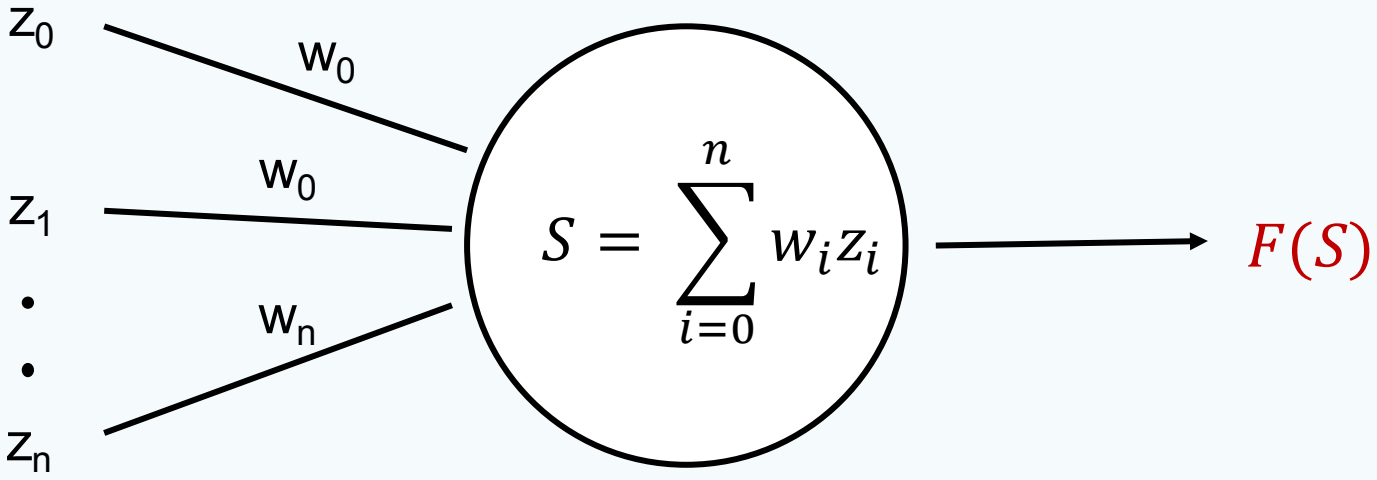
Нейрон суммирует z_i с весами соответствующих связей. К полученному после суммирования значению S применяется **нелинейная** функция активации F(S) – результат подается на выход нейрона.



Биологическая аналогия – возбуждение нейрона при некотором значении импульса, переданного нейрону.

Формальный нейрон

Нейрон суммирует z_i с весами соответствующих связей. К полученному после суммирования значению S применяется **нелинейная** функция активации F(S) – результат подается на выход нейрона.



Биологическая аналогия – возбуждение нейрона при некотором значении импульса, переданного нейрону.

Математическая формализация

Каждому слою t соответствует матрица весов $\mathbf{W}_{ij}^{(t)}$ и векторная функция активации $\mathbf{F}(x_i)$. $\mathbf{Y} = \mathbf{F}(\mathbf{Wz})$.

На вход следующему слою передается вектор (1, **S**). **x –** вектор признаков **G(x, W^t)** – полная функция сети (t=1…K), получаемая на выходном слое.

Требование нелинейности F:

Иначе сеть с K скрытыми слоями = сеть с 1 слоем

Обучение персептрона:

Тренировочный набор – (X, y) размера N

Сводится к процедуре выбора весов W для всех слоев, **минимизирующих** функционал ошибки $L = \sum_{i=0}^{N-1} l(G(X_i, W), y_i)$

***l* –** функция потерь для одного примера

Персептрон – универсальный аппроксиматор

Теорема Колмогорова:

Любая непрерывная функция от n переменных $F(x_1,x_2,\ldots, x_n)$ может быть представлена в виде:

$$F(x_1, x_2, \ldots, x_n) = \sum_{j=1}^{2n+1} g_j\left(\sum_{i=1}^{n} h_{ij}(x_i)\right),$$

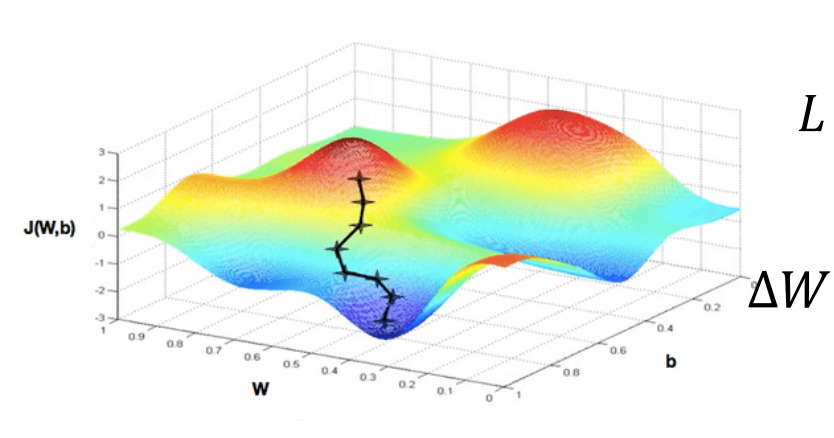
Следовательно, любую непрерывную функцию от нескольких переменных можно точно реализовать с помощью персептрона с одним скрытым слоем

Функция потерь и ее минимизация

x_i – вектор признаков, p_i – вектор предсказаний

- Классификация – в качестве функции потерь чаще всего выбирается кросс-энтропия $l = -\sum y_i \log(p_i)$
- Регрессия – СКО $l = \sum (y_i - p_i)^2$ (иногда САО, коэфф. корреляции etc.)

Для минимизации обычно используют градиентные методы (первого или второго порядков). Простейший метод – градиентный спуск



$$L = \sum l_i$$

$$\Delta W = -\varepsilon \frac{\partial L}{\partial W}$$

https://umu.to/blog/2018/06/29/hill-climbing-irl

Стохастический градиентный спуск

Недостатки градиентного спуска (не является полным списком):

- В случае наличия нескольких локальных экстремумов сходится к ближайшему (недостаток всех градиентных методов)
- В формуле $\Delta W = -\varepsilon \frac{\partial L}{\partial W}$ суммирование в L производится по всему тренировочному набору – дорого и повышает вероятность «застревания» в локальном экстремуме

Стохастический градиентный спуск

Недостатки градиентного спуска (не является полным списком):

- В случае наличия нескольких локальных максимумов сходится к ближайшему (недостаток всех градиентных методов)
- В формуле $\Delta W = -\varepsilon \frac{\partial L}{\partial W}$ суммирование в L производится по всему тренировочному набору – дорого и повышает вероятность «застревания» в локальном экстремуме

Решение – стохастический градиентный спуск

Подстраиваем веса на каждом примере (L=l).

Примеры подаем сети в случайном порядке.

Проблема – большое количество операций подстройки весов.

Стохастический градиентный спуск

Недостатки градиентного спуска (не является полным списком):

- В случае наличия нескольких локальных максимумов сходится к ближайшему (недостаток всех градиентных методов)
- В формуле $\Delta W = -\varepsilon \frac{\partial L}{\partial W}$ суммирование в L производится по всему тренировочному набору – дорого

Решение – стохастический градиентный спуск

Подстраиваем веса на каждом примере (L=l).

Примеры подаем сети в случайном порядке.

Проблема – большое количество операций подстройки весов.

Решение (улучшенное) – использование пакетов (mini-batch)

Подстраиваем веса на пакете из p примеров ($L = \sum_{i=0}^{p-1} l_i$)

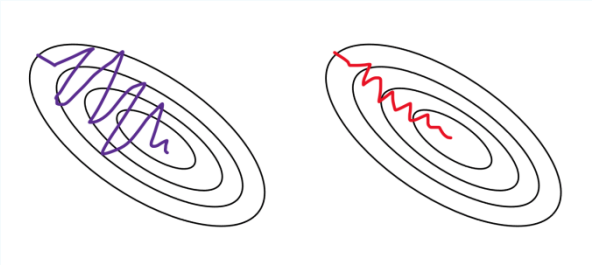
Обычно используют p=16, 32, 64,… (из возможностей RAM)

Использование момента

Недостатки градиентного спуска (продолжаем):

- Осцилляции в узких ущельях

Решение – использование момента (аналогия – импульс шарика, катящегося по склону)



$$v_t = \gamma v_{t-1} + \varepsilon \frac{\partial L}{\partial W}$$

$$\Delta W = -v_t$$

Момент Нестерова – заглядываем в будущее и ускоряем сходимость.

Вычисляем градиент в точке $W - \gamma v_{t-1}$

Ускорение сходимости оптимизации

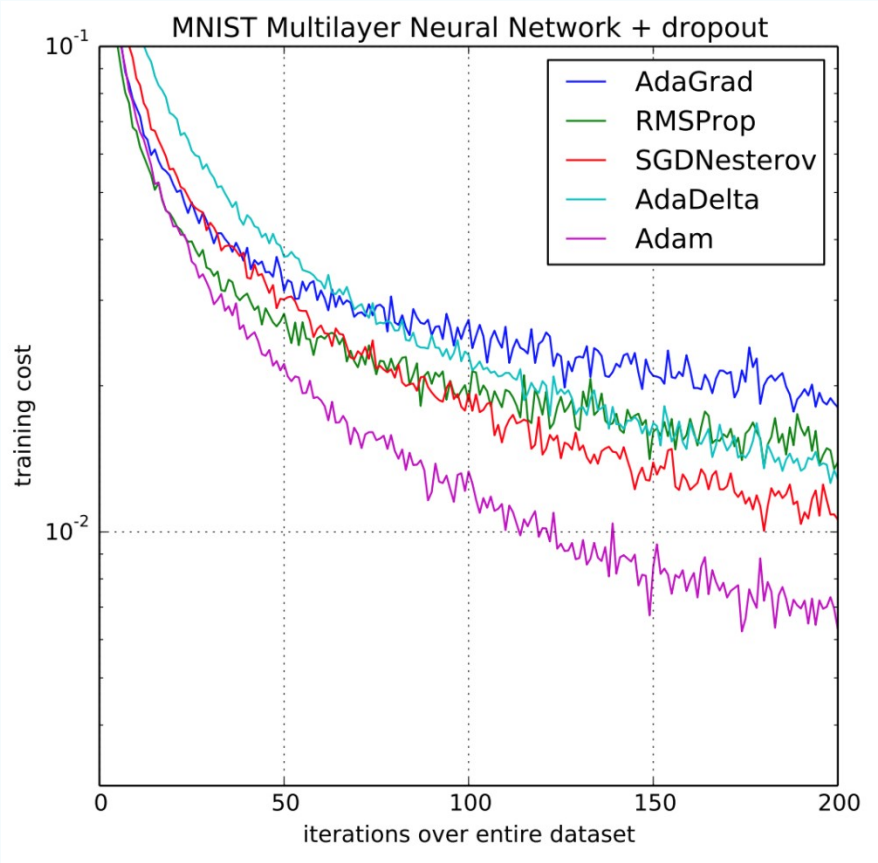
Хороший обзор - https://ruder.io/optimizing-gradient-descent

Модификации градиентного спуска:

- Adagrad – адаптивная скорость обучения для каждого веса в зависимости от степени изменения ассоциированных с ним признаков
- Adadelta – модификация Adagrad с ограничением на уменьшение скорости обучения (схожая идея в RMSProp)
- Adam – модификация Adagrad с дополнительным вычислением момента (Nadam – модификация с моментом Нестерова). Популярный выбор в качестве оптимизатора по умолчанию
- Adamax, AMSGrad, ….

Ускорение сходимости оптимизации

Adam – хороший начальный выбор, но важно помнить, что правильный выбор зависит от задачи



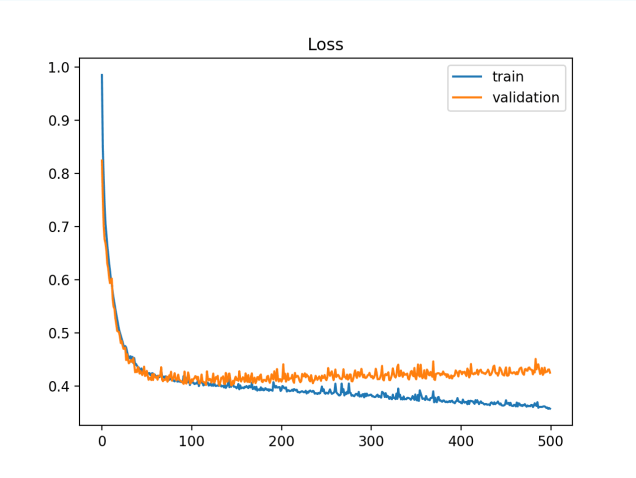
https://www.researchgate.net/figure/Multilayer-network-training-cost-on-MNIST-dataset-using-different-adaptive-learning_fig9_332662087

Ранняя остановка по валидационному набору

В какой момент нужно остановить обучение?

- После заранее определенного количества эпох (почему?)
- После прекращения уменьшения функции потерь (проблема переобучения)

Хорошее решение – случайно выделяем из тренировочного набора валидационный. Измеряем функцию потерь на этом наборе после каждой эпохи. Прекращаем обучения после N эпох без улучшения. В качестве финальной модели используем модель на лучшей эпохе.



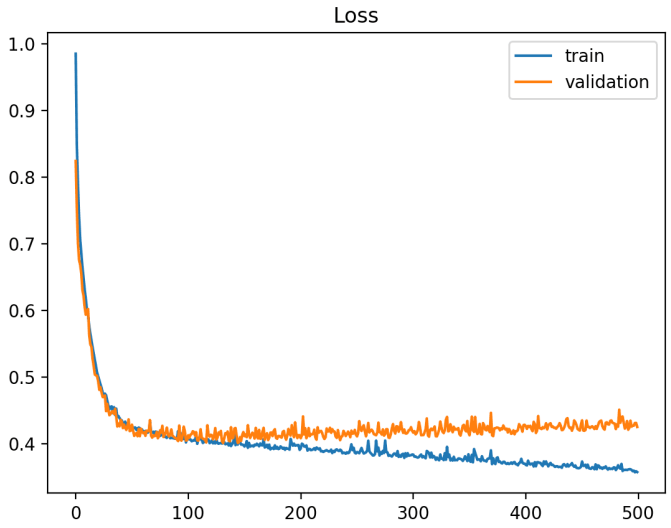
https://machinelearningmastery.com/learning-curves-for-diagnosing-machine-learning-model-performance/

Ранняя остановка по валидационному набору (бонус)

Случайно выделяем из тренировочного набора валидационный.

Измеряем функцию потерь на этом наборе после каждой эпохи.

Прекращаем обучения после N эпох без улучшения.



https://machinelearningmastery.com/learning-curves-for-diagnosing-machine-learning-model-performance/

Snapshot ensembling – сохраняем модель в локальных минимумах ошибки на валидационном наборе.

Используем ансамблирование полученных моделей.

Алгоритм обратного распространения ошибки

То, о чем умолчали ранее: как нам получить $\frac{\partial L}{\partial W}$?

Пусть z – ответы слоя перед выходным, **F** – векторная функция активации. Тогда $L(x)=L(F(W^t z(W, x)))$, t – номер выходного слоя.

Тогда (в терминах матричной производной):

$$\frac{\partial L}{\partial W^t} = W^t \frac{\partial L}{\partial F}$$

$$\frac{\partial L}{\partial W^{t-1}} = \cdots$$

Вычислить производную функцию потерь можно, используя производную сложной функции.

Как сделать это эффективно – алгоритм обратного распространения ошибки

Алгоритм обратного распространения ошибки

Предложен в 1986 г. Дэвидом Румельхартом

- Инициализируем веса случайными значениями
- Прямой проход по сети – считаем значения активации последовательно на каждом слое и сохраняем результат
- Обратный проход по сети – рекурсивно считаем производную по весам на каждом слое, используя сохраненные значения активации

Библиотеки для реализации нейросетей умеют выполнять дифференциирование произвольных архитектур

Детали - https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/

1986, Rumelhart, David E. et al., "Learning representations by back-propagating errors". Nature. Vol. 323 No. 6088, pp. 533–536.Bibcode, doi:10.1038/323533a0, 1986.

Инициализация весов

- Равномерное, нормальное распределения
- Xavier, 2010 – инициализация весов равномерным распределением, с масштабом, зависящим от количества нейронов в предыдущем слое
- Kaiming, 2015

Хорошее объяснение:

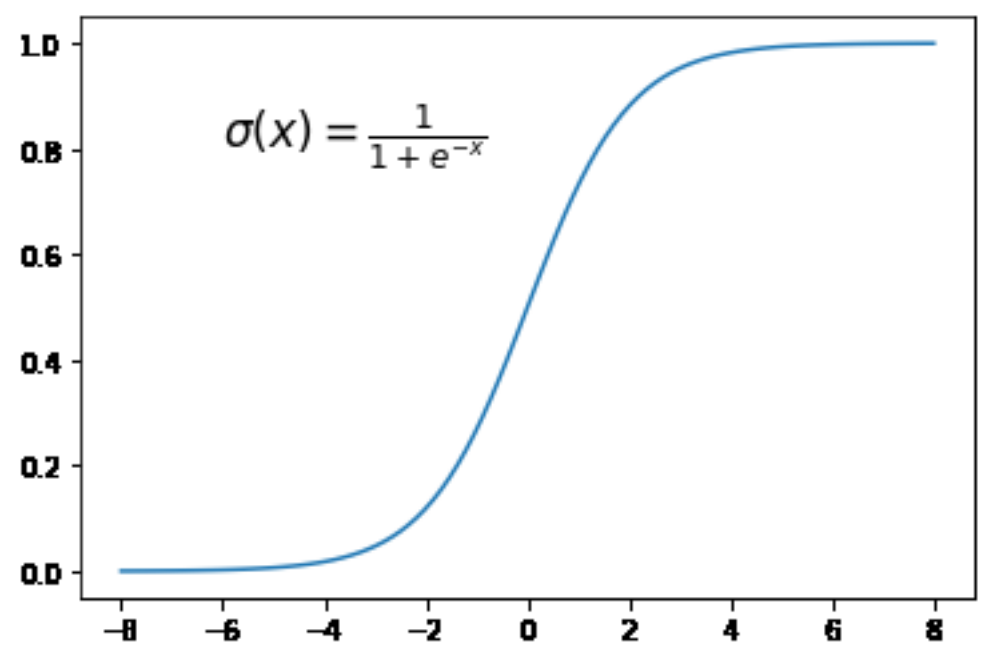
https://towardsdatascience.com/weight-initialization-in-neural-networks-a-journey-from-the-basics-to-kaiming-954fb9b47c79

Xavier Glorot et al., “Understanding the difficulty of training deep feedforward neural networks”, Journal of Machine Learning Research - Proceedings Track. Vol 9. pp. 249-256, 2010.

Kaiming He et al., “Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification”, 2015 IEEE International Conference on Computer Vision (ICCV), pp. 1026-1034, doi: 10.1109/ICCV.2015.123, 2015,.

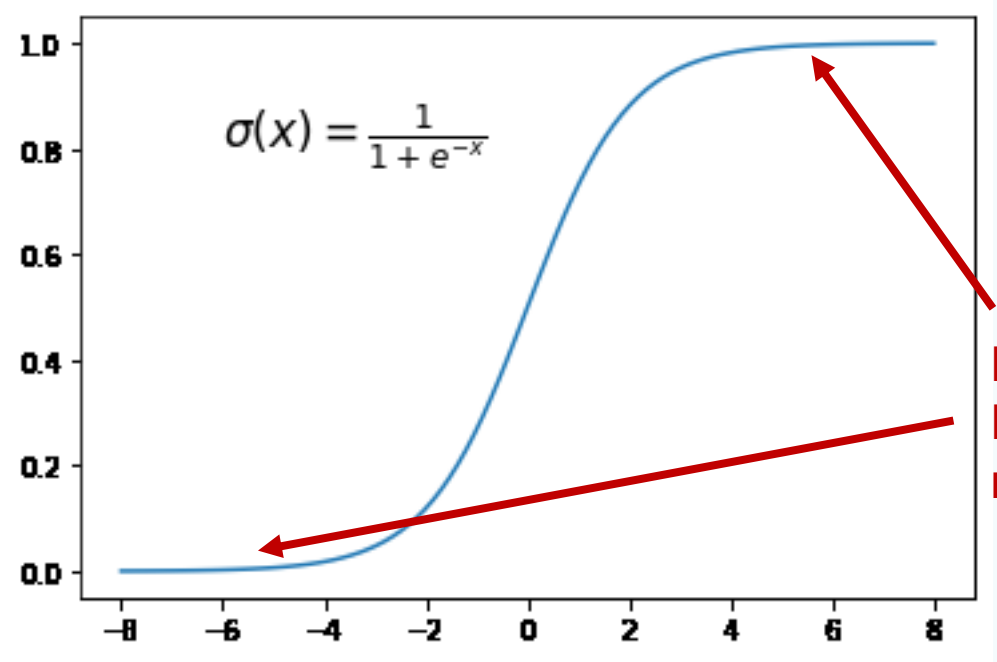
Выбор функции активации. Сигмоида

Сигмоида – одна из первых широко используемых функций активаций для нейрона. Нормирована непрерывно от 0 до 1, что позволяет моделировать вероятность.



Выбор функции активации. Сигмоида

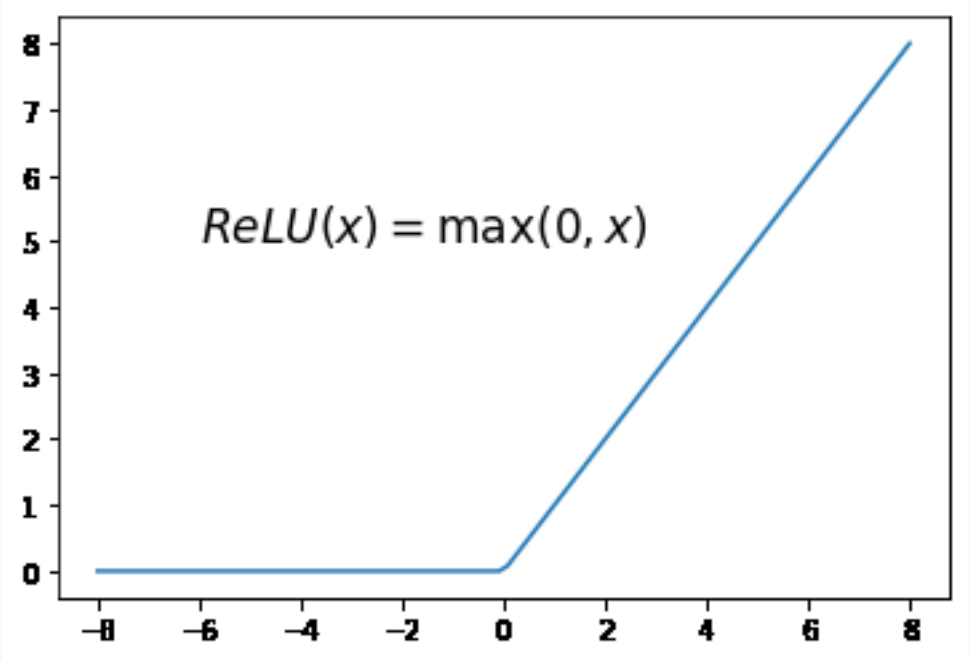
Сигмоида – одна из первых широко используемых функций активаций для нейрона. Нормирована непрерывно от 0 до 1, что позволяет моделировать вероятность.



Проблема!
Насыщение
градиента

Выбор функции активации. ReLU

ReLU=max(0, x). На сегодняшний наиболее частый выбор для активации скрытых слоев. Решена проблема насыщения градиента.



Выбор функции активации

Другие часто используемые варианты:

- tanh (-1, 1)
- Leaky ReLU = max(ax, x) – решение проблемы зануления RelU при x<0
- PReLU – Leaky ReLU с обучаемым параметром a
- …и другие

Выбор функции активации выходного слоя

Задача классификации

На выходном слое используют сигмоиду (в случае бинарной классификации) или softmax (в случае многоклассовой классификации) – активации должны быть нормированы от 0 до 1 и $\sum p = 1$

$$p = \frac{e^{x_i}}{\sum e^{x_i}}$$

Задача регрессии

На выходном слое обычно используют линейную функцию активации

Контроль скорости обучения

Чем ниже скорость обучения – тем оно устойчивее и тем меньше финальное значение ошибки. Однако при этом увеличивается время обучения. Стратегии:

- Эмпирическое правило, $\varepsilon \leq 0.01$.
- Уменьшении скорости обучения в K раз каждые N эпох
- Уменьшение скорости обучения на плато – выбираем метрику, за которой будем следить (чаще всего ошибка на валидационном наборе). Уменьшаем скорость в K раз, если не происходит улучшения в течение N эпох. Используют в том числе и с оптимизаторами с адаптивной скоростью обучения! (Adam, RMSProp,..)
- Цикличная скорость обучения (экзотика) – помогает выбраться из локальных минимумов. Например, скорость обучения может меняться синусом с экспоненциальным убыванием

Количество нейронов в скрытых слоях

- Использование эмпирических оценок $N = 2\sqrt{td}; 0.5\,(t+d) + \sqrt{P}$, где t – количество входов, d – количество выходов, P – количество примеров
- Увеличение количества нейронов в большую сторону и увеличение параметров регуляризации
- Выбор оптимальной архитектуры с помощью кросс-валидации
- Вариантов ответа множество, чаще всего выбирается экспериментально

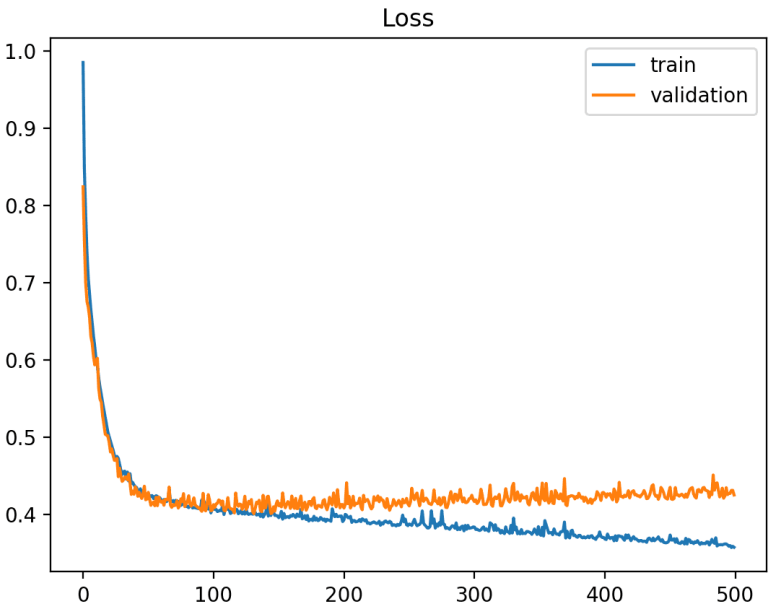
Хороший обзор:

https://machinelearningmastery.com/how-to-configure-the-number-of-layers-and-nodes-in-a-neural-network/

Проблема переобучения

Общая проблема для всех методов машинного обучения, не обошла стороной и персептрон. Хорошая иллюстрация – поведение ошибки на тренировочном и валидационном наборе.

Первый метод борьбы с переобучением – остановка по валидационному набору



https://machinelearningmastery.com/learning-curves-for-diagnosing-machine-learning-model-performance/

Распад весов (регуляризация)

Используется для того, чтобы стимулировать модель использовать меньшее численное значение весов для решения задачи (можно показать, что таким образом уменьшается эффективное количество степеней свободы). Аналогичная методика используется в гребневой регрессии.

$$\check{l} = l + \lambda \sum |w_i|^p$$

- p=1, L1 регуляризация (используется для полного зануления некоторых весов)
- p=2, L2 регуляризация (подавление весов)
- p=1,2, Elastic Net регуляризация

Иногда L2 регуляризацию называют также распадом весов (Weight decay, Hinton 1986)

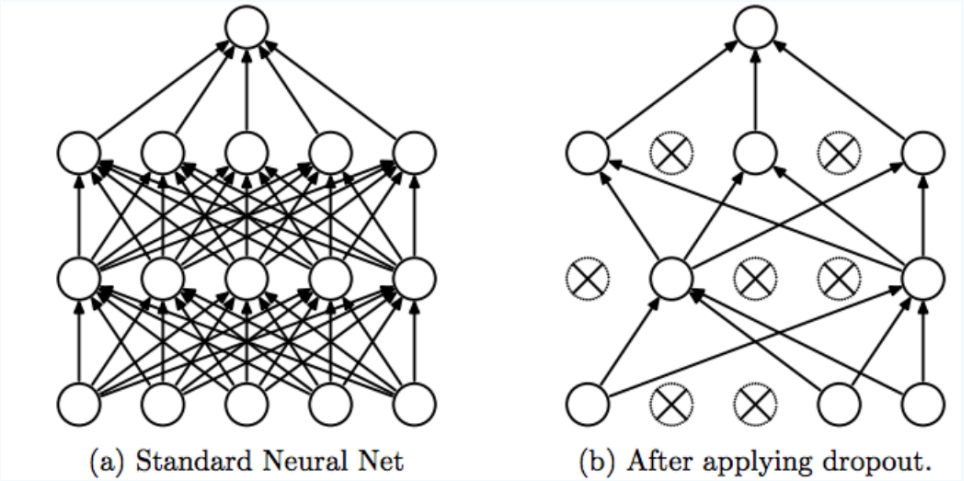
Часто параметр регуляризации выбирают для каждого слоя

Dropout

Регуляризация, призванная улучшить обобщающие способности сети => побороть проблему переобучения.

На каждой итерации обучения «отключаем» (зануляем) долю p случайных скрытых нейронов и их связи с весами. Обновляем веса. Степень регуляризации контролируется параметром p.

На стадии предсказания не применяется. Выходы нейронов корректируются в (1-p) раз.



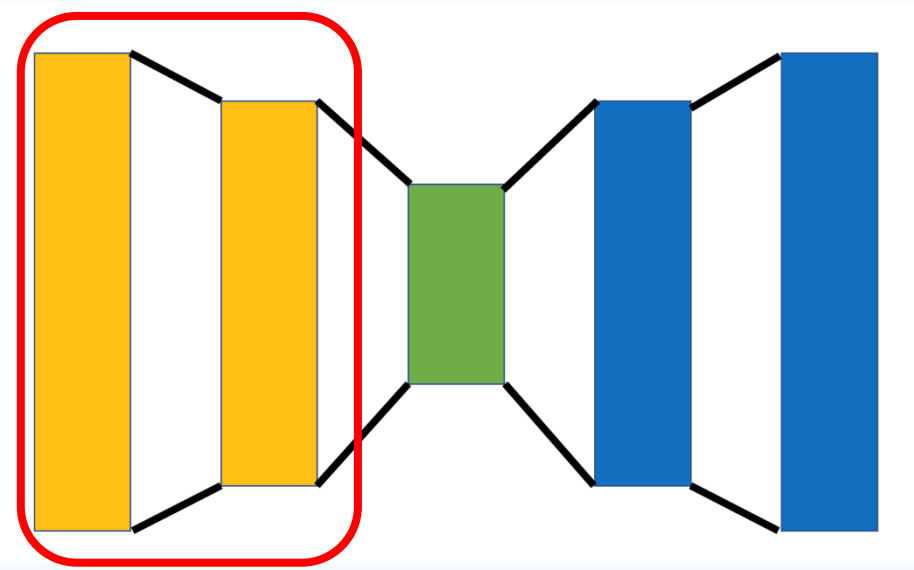
(a) Standard Neural Net (b) After applying dropout.

N.Srivastava et al., A Simple Way to Prevent Neural Networks from Overfitting J. of Machine Learning Research, 2014, V.15, pp.1929-1958.

Автоэнкодеры

Позволяет производить сжатие данных и нелинейный анализ главных компонент. Состоит из слоев энкодера, слоев декодера и слоя, называемого «узким горлом», который связывает энкодер и декодер.

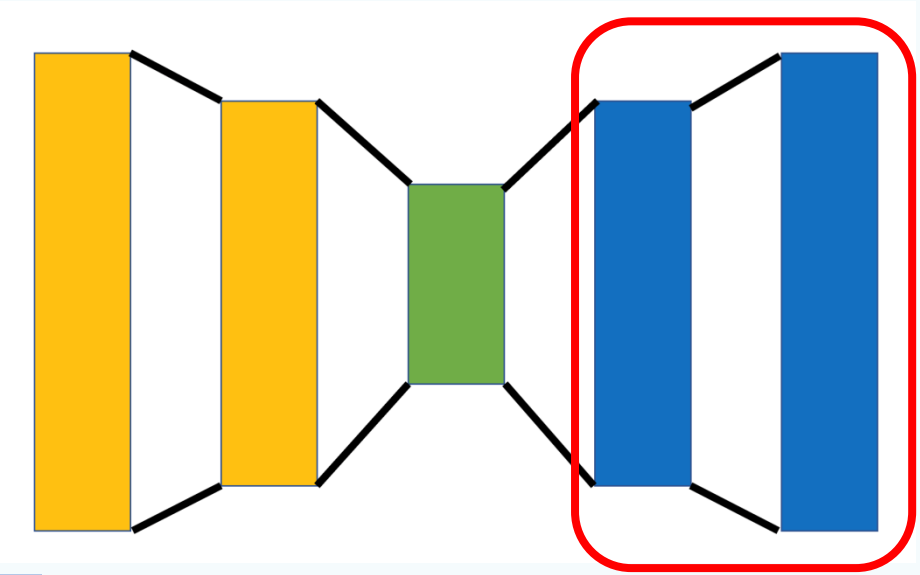
В качестве и примера и ответа подаем сети пример. Энкодер сжимает его, формируя внутреннее представление в узком горле. Декодер восстанавливает пример из его представления.



Автоэнкодеры

Позволяет производить сжатие данных и нелинейный анализ главных компонент. Состоит из слоев энкодера, слоев декодера и слоя, называемого «узким горлом», который связывает энкодер и декодер.

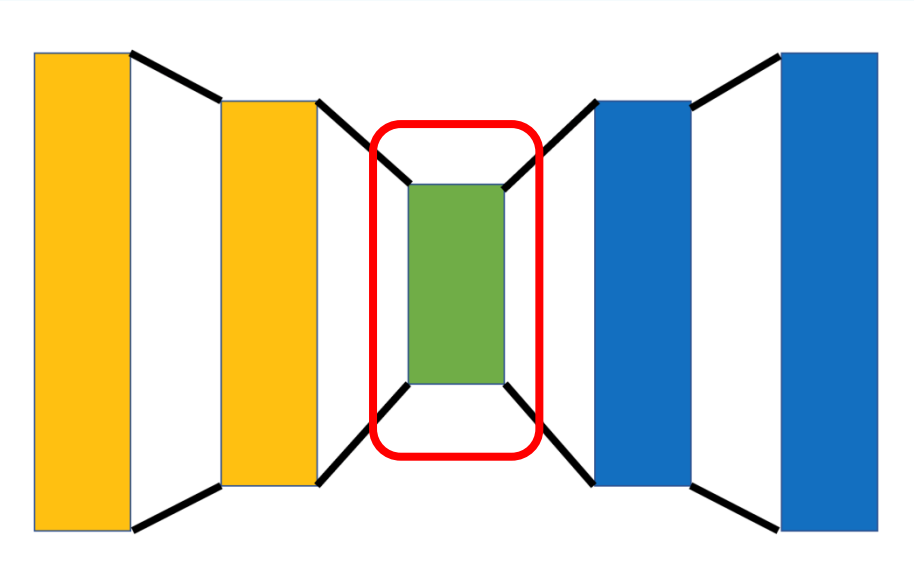
В качестве и примера и ответа подаем сети пример. Энкодер сжимает его, формируя внутреннее представление в узком горле. Декодер восстанавливает пример из его представления.



Автоэнкодеры

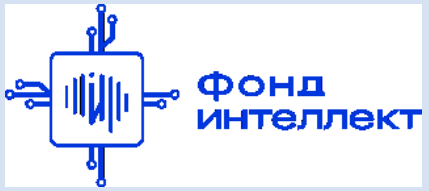
Позволяет производить сжатие данных и нелинейный анализ главных компонент. Состоит из слоев энкодера, слоев декодера и слоя, называемого «узким горлом», который связывает энкодер и декодер.

В качестве и примера и ответа подаем сети пример. Энкодер сжимает его, формируя внутреннее представление в узком горле. Декодер восстанавливает пример из его представления.



Спасибо за внимание!

Лаборатория адаптивных методов обработки данных



Учебный курс

«**Машинное обучение в физике**»

Занятие №10 (лекция).

Глубокие и свёрточные нейронные сети

Авторы курса:

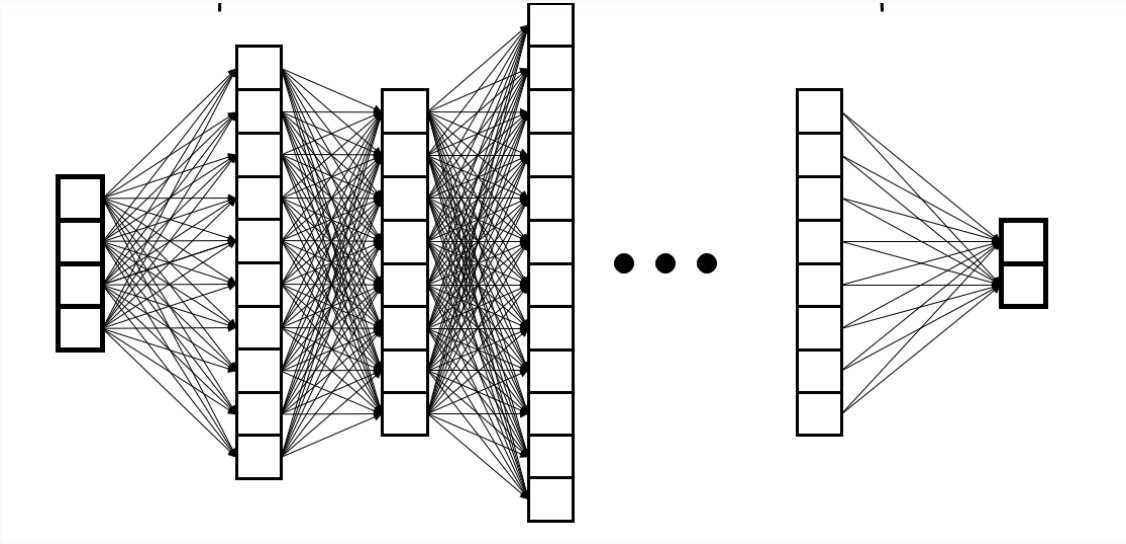
С.А. Доленко, И.М. Гаджиев, А.О. Ефиторов, И.В. Исаев, В.Р. Широкий

Глубокие нейронные сети

Обычно под понятием «глубокие нейронные сети» подразумевают сети с большим количеством скрытых слоев.

Архитектуры, которые обычно относят к глубоким сетям:

- Многослойные персептроны (N >>1)
- Свёрточные сети
- Рекуррентные сети
- Трансформеры
- …



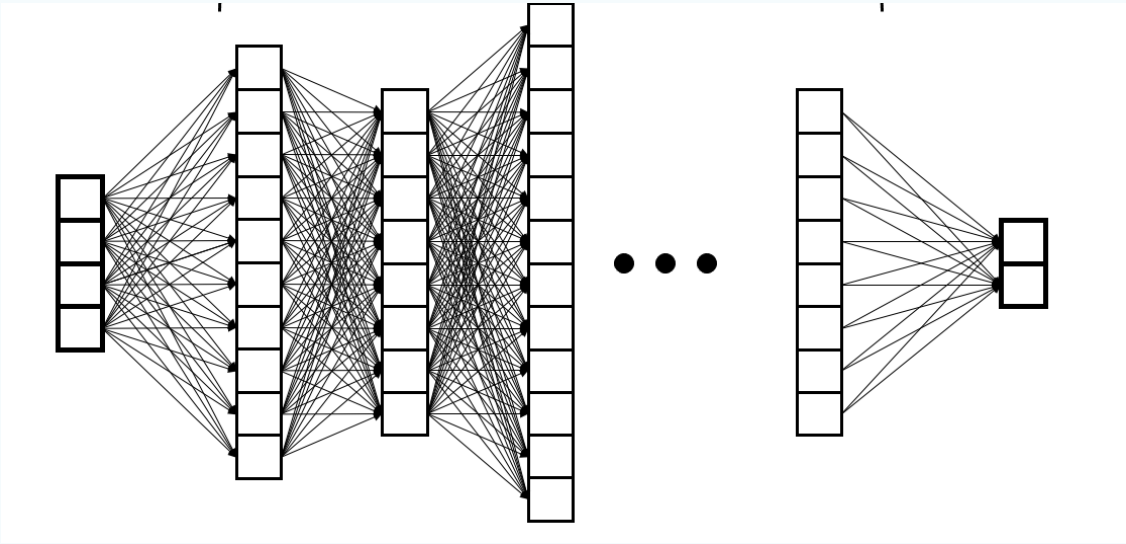
https://commons.wikimedia.org/wiki/File:Example_of_a_deep_neural_network.png

Глубокие нейронные сети

Обычно под понятием «глубокие нейронные сети» подразумевают сети с большим количеством скрытых слоев.

Архитектуры, которые обычно относят к глубоким сетям:

- Многослойные персептроны (N >>1)
- Свёрточные сети
- Рекуррентные сети
- Трансформеры
- …



https://commons.wikimedia.org/wiki/File:Example_of_a_deep_neural_network.png

Свёрточные нейронные сети

Применяются для решения задач:

- Задачи классификации рядов без временной направленности (например, спектры) и временных рядов (реже) (1D)
- Задачи классификации, детектирования, локализации и сегментации по изображению (2D)
- Задача классификации, детектирования, локализации и сегментации 3D объектов

Существенные свойства входных данных для таких задач:

- однородность
- упорядоченность
- наличие устойчивых взаимосвязей входных признаков (~ скоррелированность)

Операция свёртки

Операцию свёртки матрицы A по матрице B (ядро свёртки) можно представить следующим образом:

- проходимся окном размера B по матрице A,
- перемножаем поэлементно элементы матрицы B в окне с элементами матрицы B. Математически:

$$C_{mn} = A * B = \sum_{j=-\infty}^{+\infty} A_{ij} B_{m-i,n-j}$$

Легко обобщить на случай 1D, 2D, 3D, …

4	1	4	9	2
7	4	6	5	4
1	3	5	3	6
6	9	6	2	4
0	7	7	0	3

✖

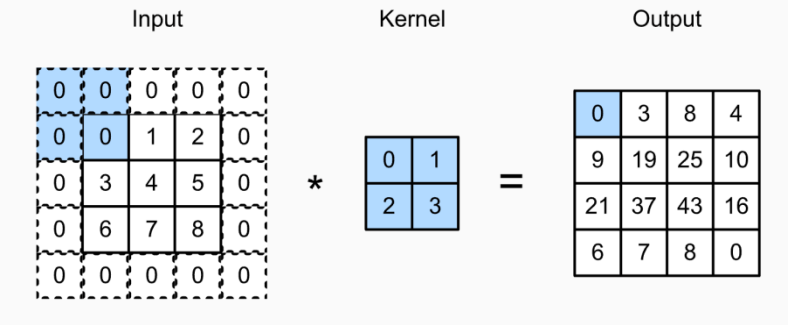
1	0	0
0	0	0
0	0	0

=

4	1	4
7	4	6
1	3	5

Операция свёртки. Padding и шаг

- При прохождении ядром свёртки по матрице элементы по краям матрицы остаются задействованными меньшее число раз.
- Для того, чтобы избежать этой проблемы, создаются дополнительные строки и столбцы матрицы по краям, заполненные нулями (padding).



- Для ускорения вычисления свёртки ее зачастую выполняют с некоторым шагом (stride)

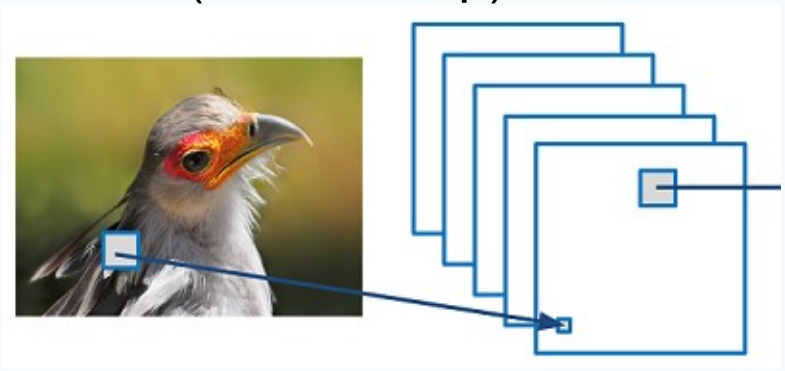
Подробнее https://d2l.ai/chapter_convolutional-neural-networks/padding-and-strides.html

Свёрточный нейрон

- У свёрточного нейрона (или фильтра) веса представляют собой ядро свёртки некоторого размера (MxN).
- На вход свёрточному нейрону обычно подается пример с несколькими каналами (временные каналы, цвет в RGB, …).
- Свёртка происходит отдельно по каждому каналу, затем результат суммируется поэлементно.
- Активационные функции используются такие же, как и для обычного персептрона (ReLU, сигмоида, …).
- Активация применяется к результату свёртки поэлементно.

Свёрточный слой

- Свёрточный слой – слой нейронной сети, составленный из свёрточных нейронов (фильтров).
- На вход свёрточному слою с K фильтрами подается матрица с N каналами, на выходе – матрица с K каналами (то есть результат по каждому фильтру образует отдельный канал).
- Результат обработки по каждому фильтру обычно называют картой признаков (feature map)



Результат работы свёрточного слоя с 5 фильтрами по матрице изображения (исходное количество каналов 3)

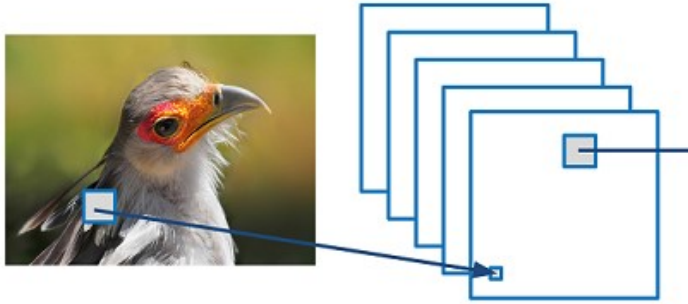
https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/

Свёрточный слой 1D. Пример

- Чаще всего 1D слои используются для извлечения признаков из последовательностей – например, из временного ряда.
- На вход подается временной ряд в формате (T, F), где T – количество временных шагов, F – размерность вектора признаков временного ряда, в общем смысле - количество каналов.
- В свёрточном слое K фильтров, каждый из них выполняет свёртку по временному ряду.
- В простейшем случае (без паддинга и шага), результат обработки имеет формат (T, K).
- Результат передается последующим слоям.

Свёрточный слой 2D. Пример

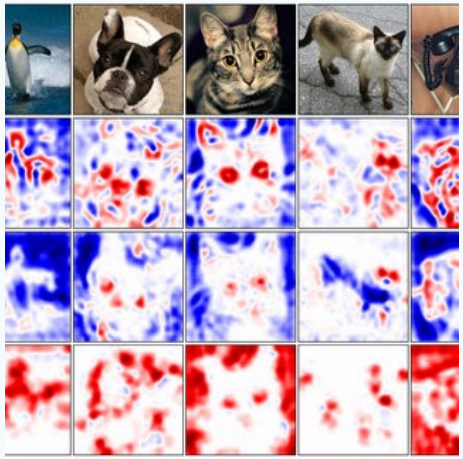
- Чаще всего 2D слои используются для извлечения признаков из изображений.
- На вход подается матрица изображения в формате $(W, H, 3)$, где W – ширина, H – высота, 3 – глубина (RGB), в общем смысле - количество каналов.
- В свёрточном слое K фильтров, каждый из них выполняет свёртку по изображению.
- В простейшем случае (без паддинга и шага), результат обработки имеет формат (W, H, K). Результат передается последующим слоям.

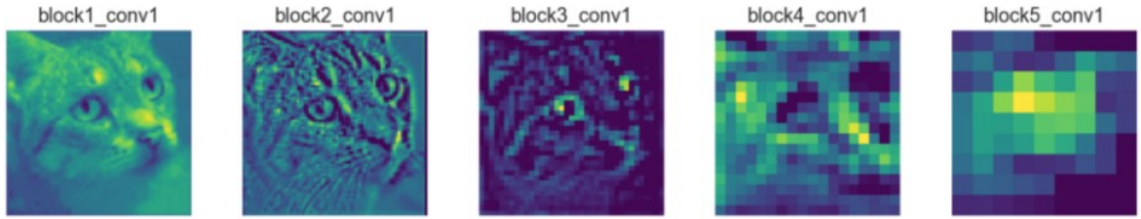


https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/

Свёрточный слой. Биологическая интуиция

- Пусть задача состоит в классификации вида животных. Тогда каждый фильтр во входном свёрточном слое учиться извлекать из изображения некоторые существенные признаки, характерные для каждого вида.
- Свойство свертки позволяет выявлять признаки вне зависимости от их положения на картинке. Например, глаза, уши, контур тела.
- Последующие свёрточные слои извлекают некоторые мета-признаки на основе признаков, полученных предыдущим слоем.

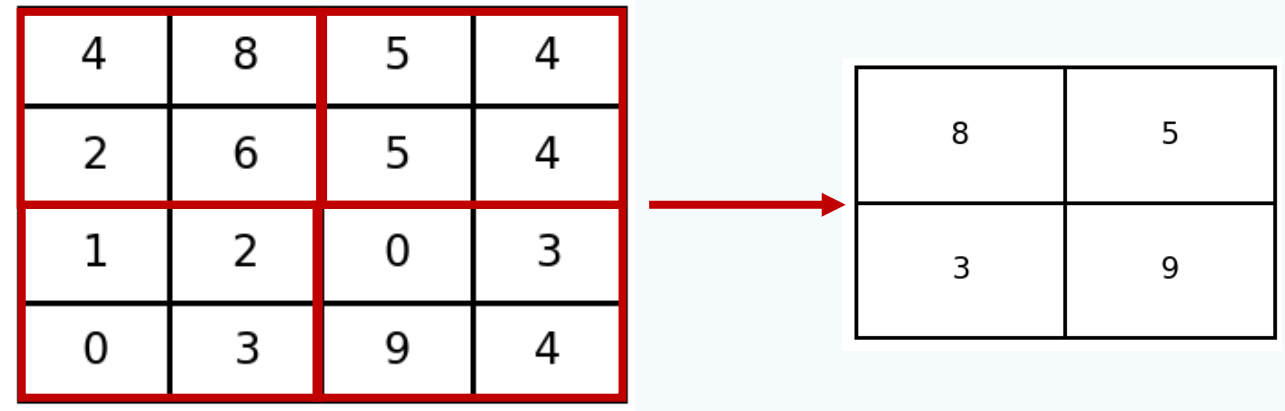




https://www.researchgate.net/figure/Visualization-of-three-different-feature-maps-taken-from-the-inception-3a-output_fig3_313739370

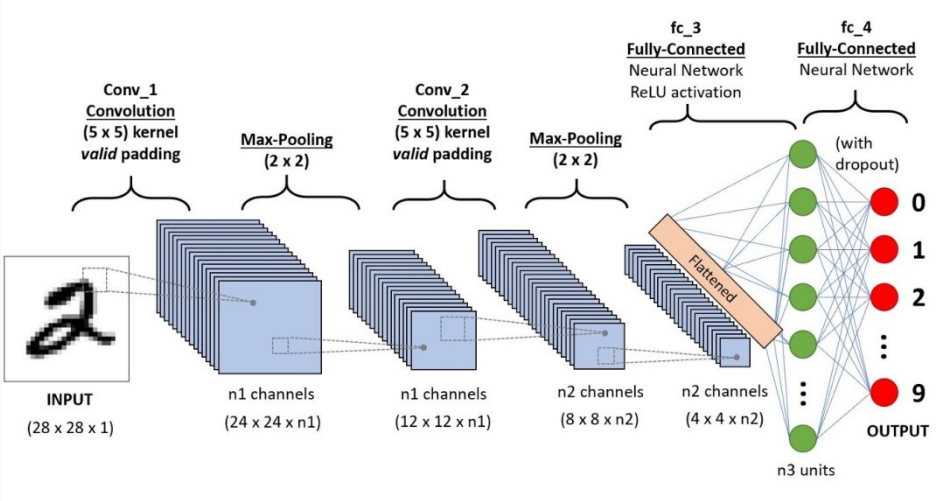
Пулинг (агрегация)

- Несмотря на то, что свёрточный слой может выявлять характерные формы (признаки) вне зависимости от их позиции, на получаемой карте признаков сигнал имеет четкую позицию.
- Чтобы сделать свёрточные сети менее восприимчивыми к мелким деталям (а также маленьким искажениям формы), применяется слой пулинга.
- Карта признаков разбивается на кусочки размером MxN (обычно 2x2).
- Для каждого такого кусочка считается среднее или максимум.



Архитектура свёрточной нейронной сети

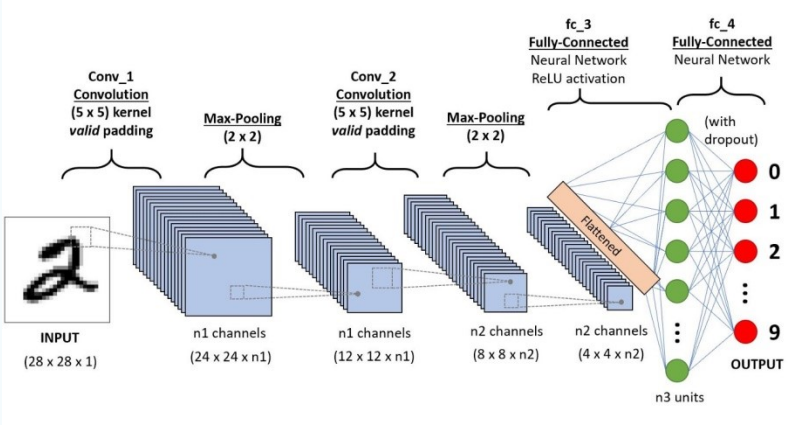
- Свёрточные слои (каждый сопровождается пулингом)
- Слой реорганизации полученных карт признаков в 1D вектор (flattening)
- Полносвязная НС (персептрон) – иногда эту часть называют «головой»



https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

Архитектура свёрточной нейронной сети

- Свёрточные слои (каждый сопровождается пулингом)
- Global average (max) pooling – взятие среднего (максимума) по каждой из карт признаков
- Слой реорганизации полученных карт признаков в 1D вектор (flattening)
- Полносвязная НС (персептрон) – иногда эту часть называют «головой»



https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

Batch normalization (нормировка пакета)

- Необходимость этой процедуры возникает при увеличении количества свёрточных слоев.
- Проблема внутреннего сдвига ковариации (internal covariate shift):

Обновление весов слоя i + 1 происходит с учетом формы распределения выходов слоя i.

Однако в результате обновления весов слоя i распределение его выходов меняется -> увеличивается ошибка на слое i+1

Batch normalization (нормировка пакета)

- Необходимость этой процедуры возникает при увеличении количества сверточных слоев.
- Проблема внутреннего сдвига ковариации (internal covariate shift) :

Обновление весов слоя i + 1 происходит с учетом формы распределения выходов слоя i.

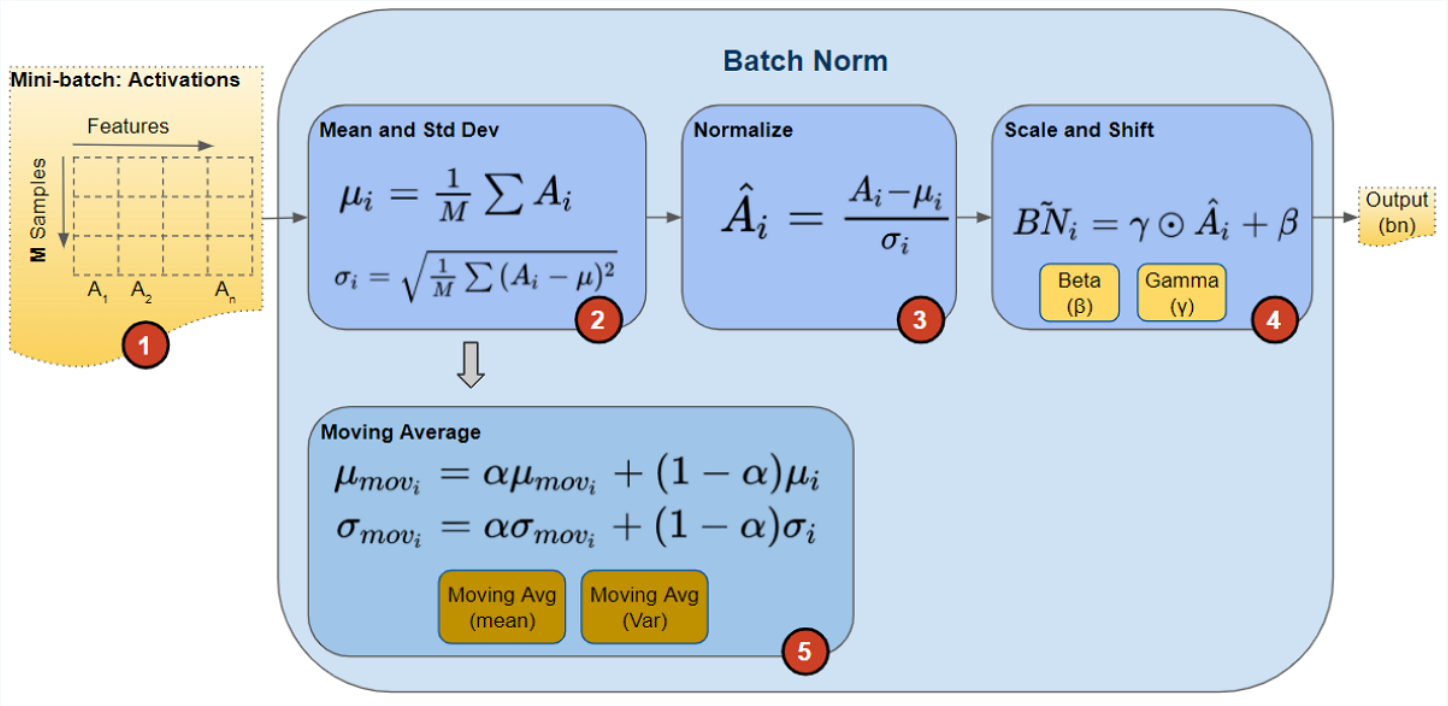
Однако в результате обновления весов слоя i распределение его выходов меняется -> увеличивается ошибка на слое i+1

- Слой batch normalization:

- Проводим масштабирование по среднему и стандартному отклонению для каждого пакета (batch).
- При этом параметры корректировки масштабирования выучиваем в ходе обучения, а не просто считаем скользящее среднее и отклонение, т.к. они могут быть нерепрезентативными.

Batch normalization (нормировка пакета)

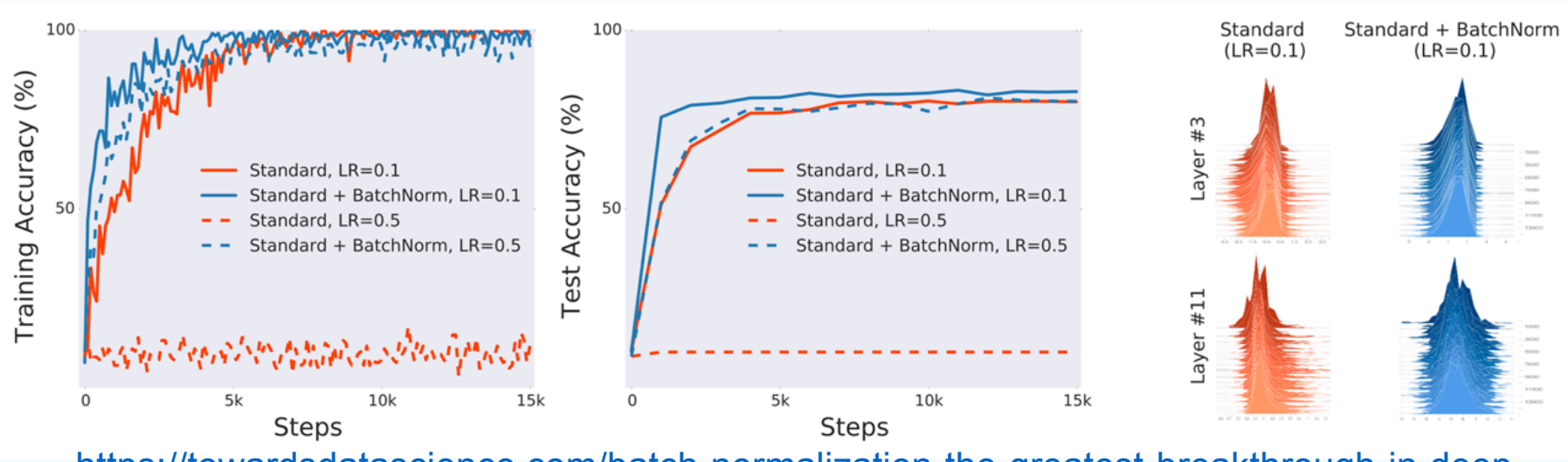
https://towardsdatascience.com/batch-norm-explained-visually-how-it-works-and-why-neural-networks-need-it-b18919692739



Batch normalization (нормировка пакета)

Добавление batch normalization после комбинации сверточного слоя и пулинга повышает эффективную скорость обучения глубоких сверточных НС

Sergey Ioffe et al., Batch Normalization: “Accelerating Deep Network Training by Reducing Internal Covariate Shift”, In Proceedings of the 32nd International Conference on International Conference on Machine Learning – vol. 37 (ICML'15). JMLR.org, pp. 448–456, 2015



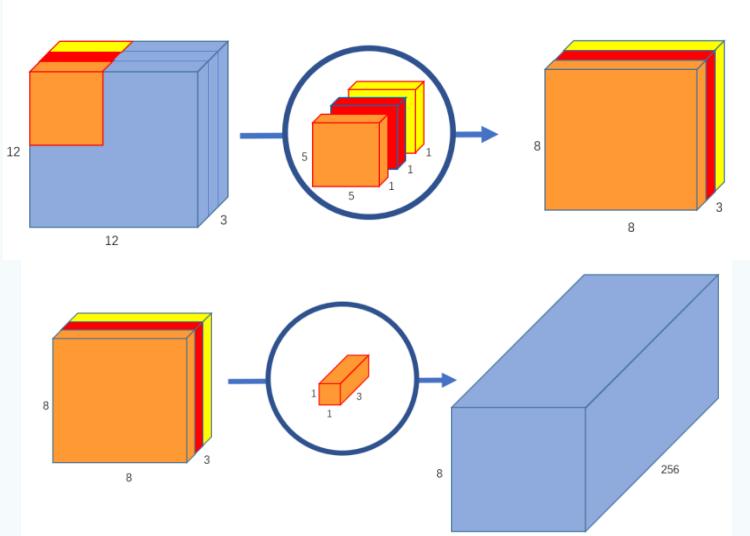
https://towardsdatascience.com/batch-normalization-the-greatest-breakthrough-in-deep-learning-77e64909d81d

Depthwise separable convolution

Разделяем свертку на 2 этапа:

- Пространственная свертка по каждому каналу отдельно
- Свертка по глубине ядром 1xN, где N – число каналов

Получаем сокращение числа перемножений



https://towardsdatascience.com/a-basic-introduction-to-separable-convolutions-b99ec3102728

Skip (jump) connections – Обходные соединения

Проблема «взрыва градиента»:

Возникает в сетях с большим количеством слоев. При протекании градиента через сеть на каждом слое он «умножается» на веса.

Если веса << 1 -> градиент быстро устремляется к 0,

если >> 1 -> градиент быстро устремляется к бесконечности

(тем быстрее, чем больше слоев, т.к. множитель на первом слое W^n)

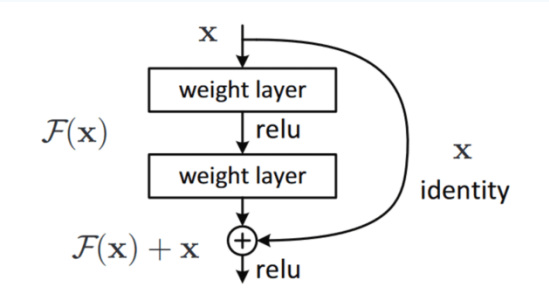
Skip (jump) connections – Обходные соединения

Проблема «взрыва градиента»:

Возникает в сетях с большим количеством слоев. При протекании градиента через сеть на каждом слое он «умножается» на веса.
Если веса << 1 -> градиент быстро устремляется к 0,
если >> 1 -> градиент быстро устремляется к бесконечности
(тем быстрее, чем больше слоев, т.к. множитель на первом слое W^n)

Решение:

Skip connections - обходные соединения между слоями. Операция суммирования не увеличивает градиент (относительно умножения)



https://theaisummer.com/skip-connections/

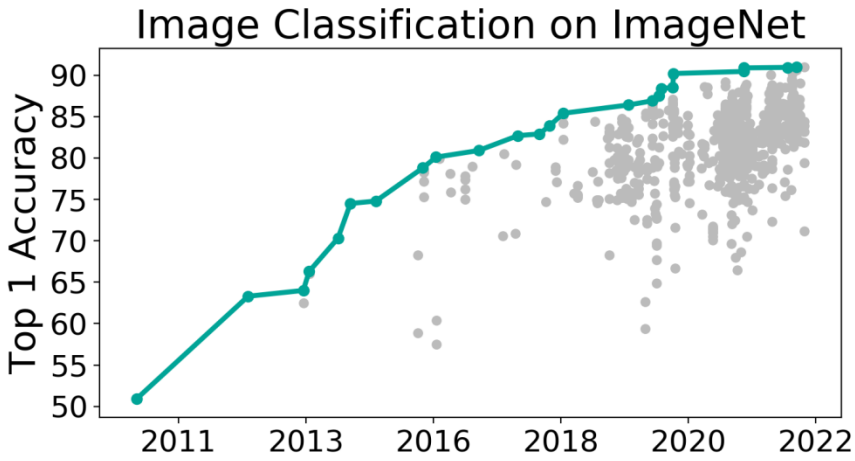
Эталонный набор (benchmark) ImageNet

Для сравнения моделей классификации изображений часто используют набор ImageNet (14 197 122 изображения из 21841 категорий).

Метрика – доля правильных ответов по топ 1, 5, 10 предсказаниям сети.

Текущий топ-5 (2022) – 99.02% (человек – 95,0%, побежден в 2015 г.)

https://www.image-net.org/

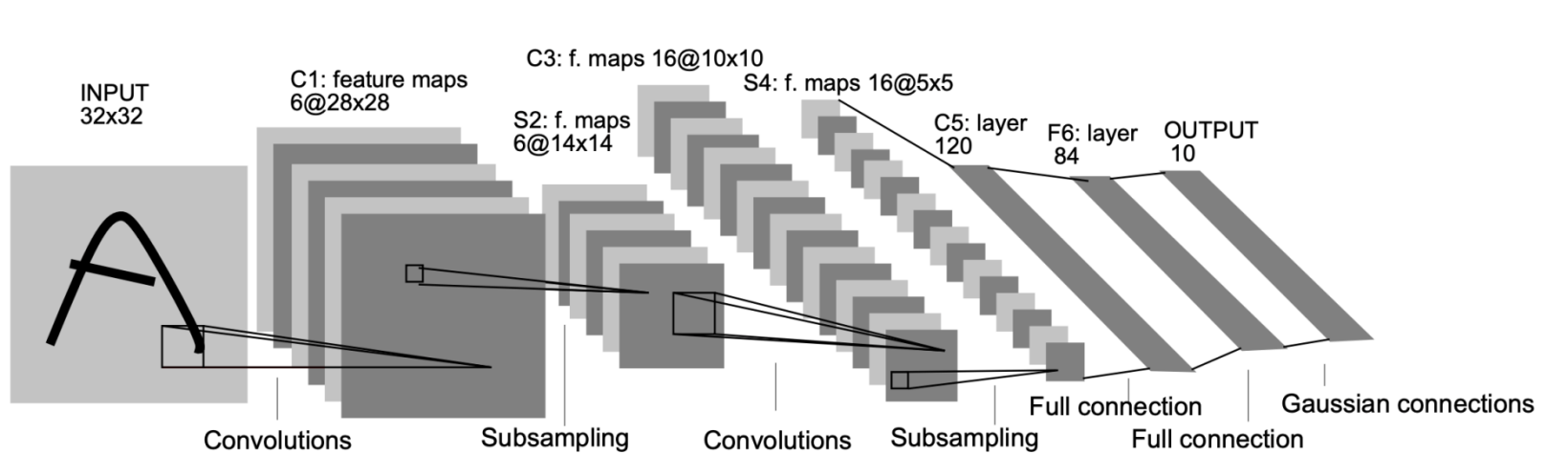


https://paperswithcode.com/sota/image-classification-on-imagenet

LeNet-5 (1998)

Одна из первых архитектур сверточных НС

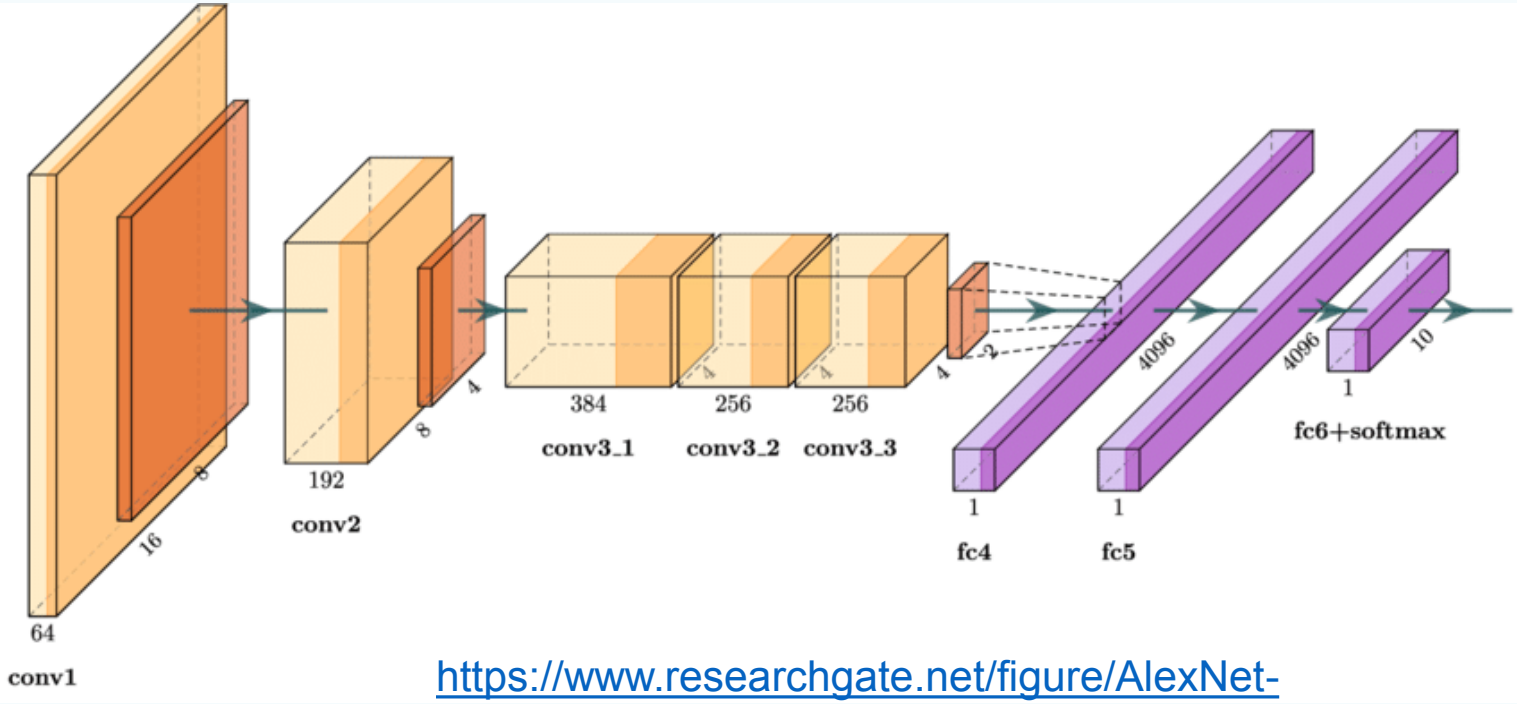
Yann LeCun et al., “Gradient Based Learning Applied to Document Recognition”, Proceedings of the IEEE. Vol. 86, pp. 2278 – 2324, doi: 10.1109/5.726791, 1998



AlexNet (2012)

84.6% top-5 ImageNet, 63.3% top-1, 60M параметров

Krizhevsky et al., “ImageNet Classification with Deep Convolutional Neural Networks”, Neural Information Processing Systems, Vol. 25, doi: 10.1145/3065386, 2012

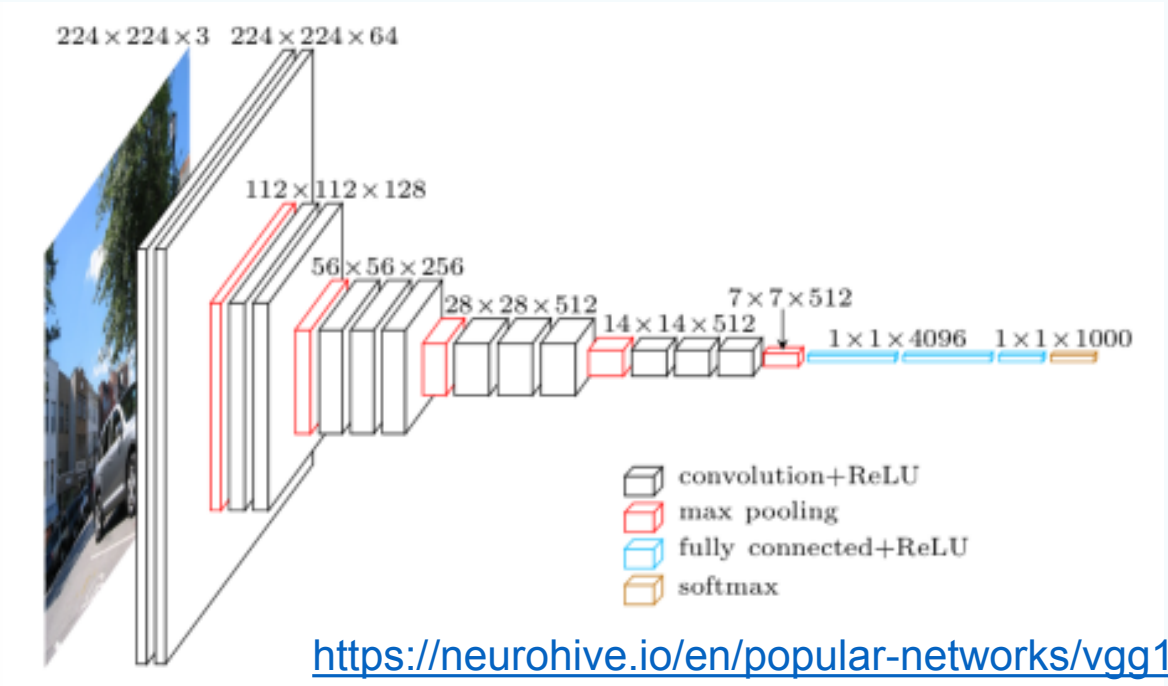


https://www.researchgate.net/figure/AlexNet-architecture-used-as-the-baseline-model-for-the-analysis-of-results-on-the_fig5_339756908

VGG16 (2014)

Хорошая базовая свёрточная НС, 92.7% top-5 ImageNet, 74.4% top-1, 138M параметров

Karen Simonyan et al., “Very Deep Convolutional Networks for Large-Scale Image Recognition”, arXiv 1409.1556, 2014

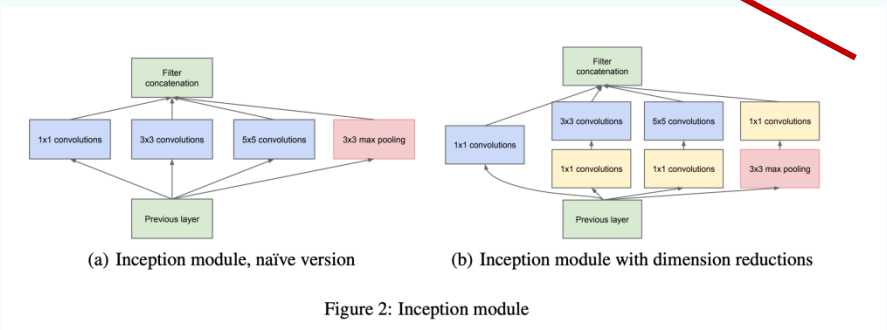


https://neurohive.io/en/popular-networks/vgg16/

Inception (GoogLeNet) (2014)

Архитектура с увеличением размера сети и с попыткой уменьшить стоимость обучения – использование фильтров разного размера на одном и том же слое –> параллелизация, 92.2% top-5 ImageNet, 74.8% top-1, 23M параметров

Christian Szegedy et al, “Going deeper with convolutions”, IEEE Conference on Computer Vision and Pattern Recognition, pp. 1-9, 2015



(a) Inception module, naïve version (b) Inception module with dimension reductions

Figure 2: Inception module

+ Доп. классификаторы на нескольких уровнях

Inception v2 (2015)

Три разных варианта Inception блока, использованных последовательно,

ImageNet top-1 74.8%, top-5 – 92.2%, 11.2M параметров

Christian Szegedy et al, “Rethinking the Inception Architecture for Computer Vision”, pp. 2818-2826, doi: 10.1109/CVPR.2016.308, 2016

type	patch size/stride or remarks	input size
conv	3×3/2	299×299×3
conv	3×3/1	149×149×32
conv padded	3×3/1	147×147×32
pool	3×3/2	147×147×64
conv	3×3/1	73×73×64
conv	3×3/2	71×71×80
conv	3×3/1	35×35×192
3×Inception	As in figure 5	35×35×288
5×Inception	As in figure 6	17×17×768
2×Inception	As in figure 7	8×8×1280
pool	8 × 8	8 × 8 × 2048
linear	logits	1 × 1 × 2048
softmax	classifier	1 × 1 × 1000

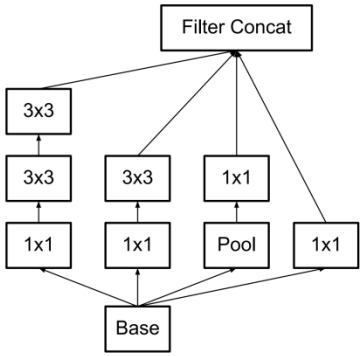


Figure 5. Inception modules where each 5 × 5 convolution is placed by two 3 × 3 convolution, as suggested by principle 3 Section 2.

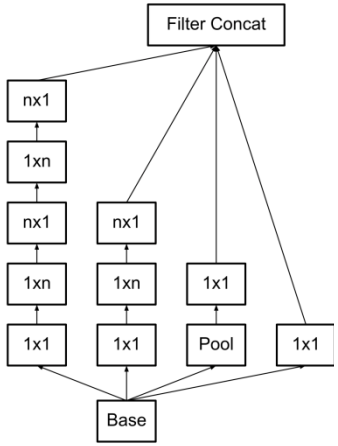


Figure 6. Inception modules after the factorization of the $n \times n$ convolutions. In our proposed architecture, we chose $n = 7$ for the 17×17 grid. (The filter sizes are picked using principle 3)

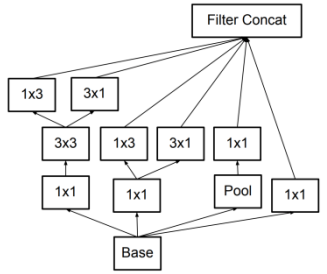


Figure 7. Inception modules with expanded the filter bank outputs. This architecture is used on the coarsest (8 × 8) grids to promote high dimensional representations, as suggested by principle 2 of Section 2. We are using this solution only on the coarsest grid, since that is the place where producing high dimensional sparse representation is the most critical as the ratio of local processing (by 1 × 1 convolutions) is increased compared to the spatial aggregation.

Inception v3 (2015)

Модификация Inception V2 с Batch Normalization в доп. классификаторах, использованием сверток 7x7 и использованием Label Smoothing – (добавление шума в метки для борьбы с переобучением), ImageNet top-1 77.12%

Christian Szegedy et al, “Rethinking the Inception Architecture for Computer Vision”, pp. 2818-2826, doi: 10.1109/CVPR.2016.308, 2016

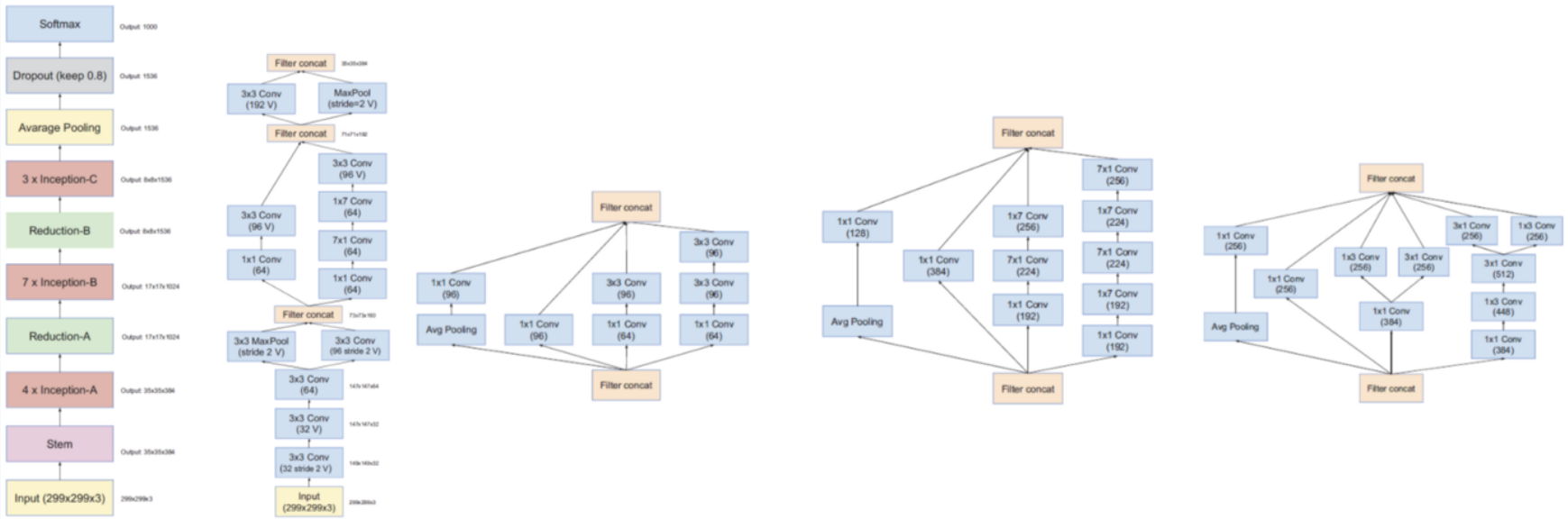
Модификация Inception V3 – Xception (2016), ImageNet top-1 79%, top-5 94.5%

François Chollet, “Xception: Deep Learning with Depthwise Separable Convolutions”, pp. 1800-1807, doi: 10.1109/CVPR.2017.195, 2017

Inception v4 (2016)

Модификация Inception V3 с большим количеством Inception модулей, 80% top-1, 95% top-5

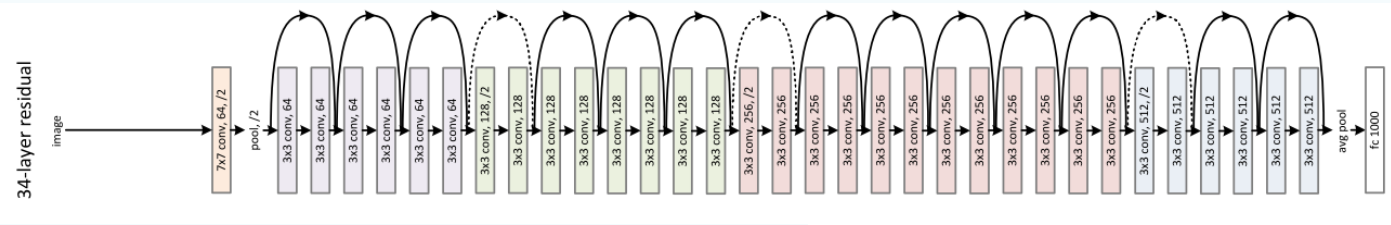
Christian Szegedy et al, Inception-v4, “Inception-ResNet and the Impact of Residual Connections on Learning”, AAAI Conference on Artificial Intelligence, Vol. 31, doi: 10.1609/aaai.v31i1.11231, 2016



ResNet (2015)

Архитектура с использованием обходных соединений (18, 34, 50, 101, 152 слоя) - популярный выбор в качестве архитектуры по умолчанию

Kaiming He et al, “Deep Residual Learning for Image Recognition”, 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770-778, doi: 10.1109/CVPR.2016.90, 2016.



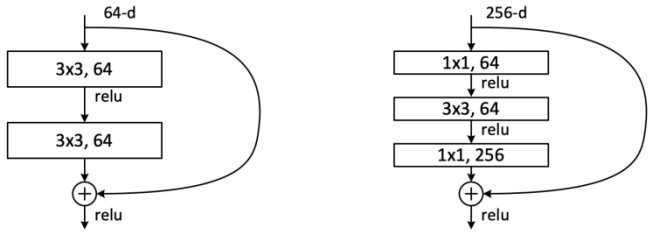


Figure 5. A deeper residual function $\mathcal{F}$ for ImageNet. Left: a building block (on 56×56 feature maps) as in Fig. 3 for ResNet-34. Right: a “bottleneck” building block for ResNet-50/101/152.

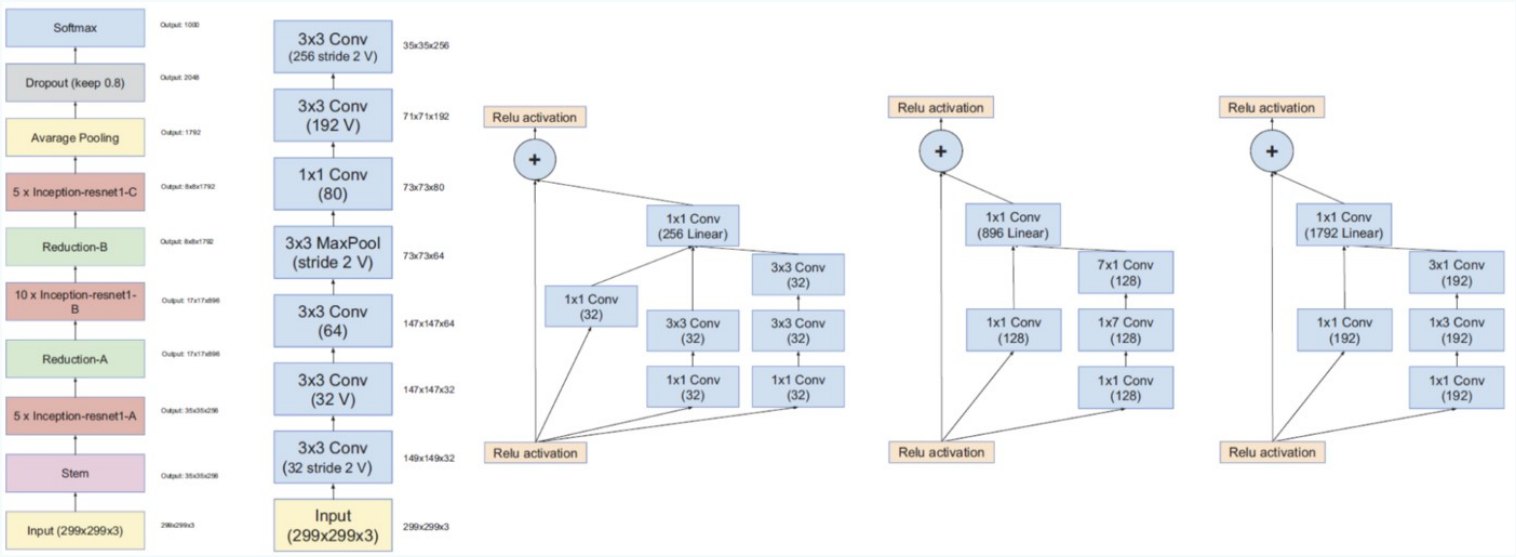
model	top-1 err.	top-5 err.
VGG-16 [41]	28.07	9.33
GoogLeNet [44]	-	9.15
PReLU-net [13]	24.27	7.38
plain-34	28.54	10.02
ResNet-34 A	25.03	7.76
ResNet-34 B	24.52	7.46
ResNet-34 C	24.19	7.40
ResNet-50	22.85	6.71
ResNet-101	21.75	6.05
ResNet-152	**21.43**	**5.71**

Table 3. Error rates (%, **10-crop** testing) on ImageNet validation. VGG-16 is based on our test. ResNet-50/101/152 are of option B that only uses projections for increasing dimensions.

Inception-ResNet v1 (2016)

Модификация Inception V4 со Skip Connections, 78.7% top-1, 94.5% top-5

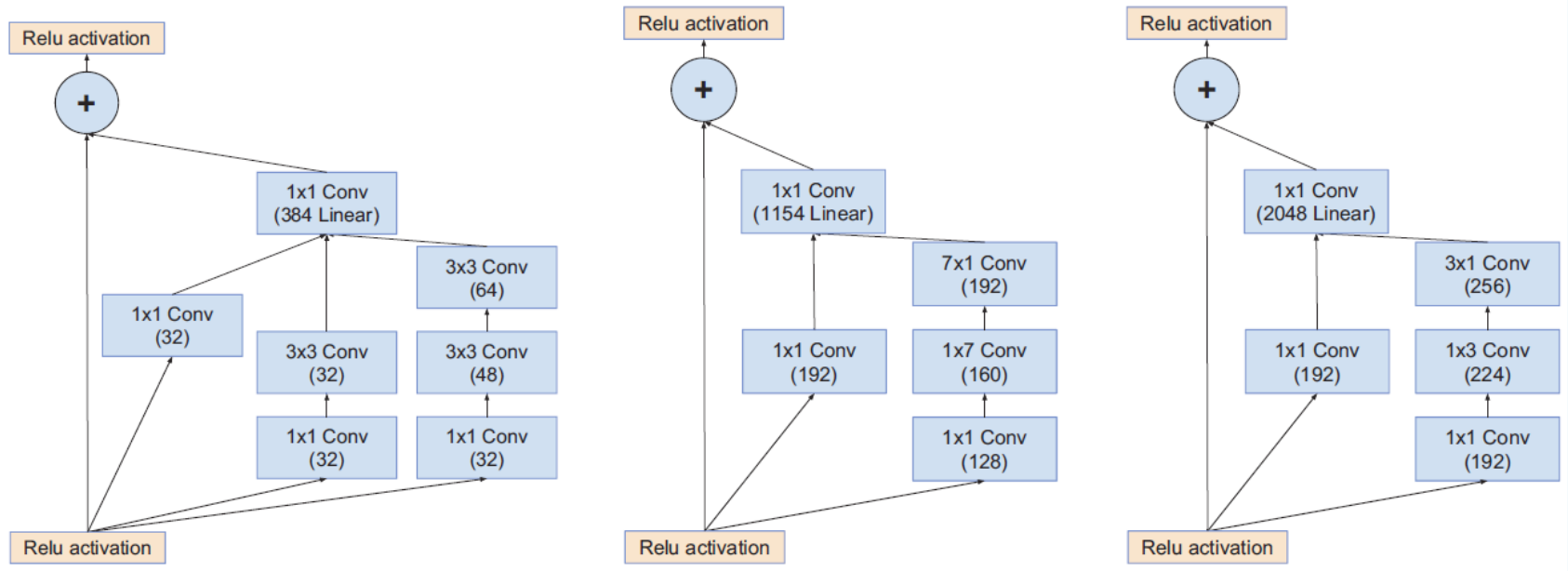
Christian Szegedy et al, Inception-v4, “Inception-ResNet and the Impact of Residual Connections on Learning”, AAAI Conference on Artificial Intelligence, Vol. 31, doi: 10.1609/aaai.v31i1.11231, 2016



Inception-ResNet v2 (2016)

Модификация Inception V4 со Skip Connections, 80.1% top-1, 95.1% top-5, 55.8 M параметров

Christian Szegedy et al, Inception-v4, “Inception-ResNet and the Impact of Residual Connections on Learning”, AAAI Conference on Artificial Intelligence, Vol. 31, doi: 10.1609/aaai.v31i1.11231, 2016



MobileNet (2017)

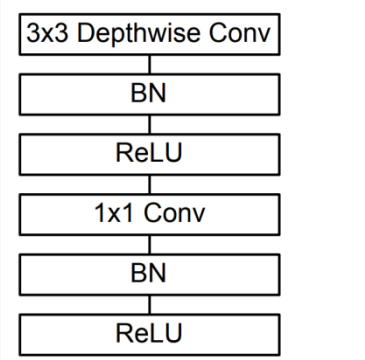
Архитектура для слабых вычислителей, использующая разделяемые свертки и batch normalization, 70.6% top-1 ImageNet 224x224,

4.2M параметров

Andrew G. Howard et al, “MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications”, arXiv:1704.04861, 2017

Table 1. MobileNet Body Architecture

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$ Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
$5\times$ Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$



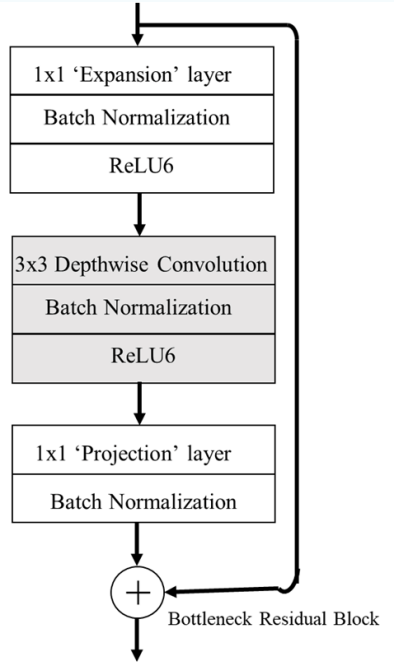
MobileNet V2 (2018)

Модификация MobileNet со skip connections, 72% (74.7 v1.4) top-1 ImageNet, 3.4M параметров

Andrew G. Howard et al., “MobileNetV2: Inverted Residuals and Linear Bottlenecks”, pp. 4510-4520. doi: 10.1109/CVPR.2018.00474, 2018

Input	Operator	t	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1x1	-	k	-	

Bottleneck блок



MobileNet V3 (2019)

Модификация MobileNetV2, с использованием Squeeze and Excite блока, в двух вариантах – L,S. 75.2% top-1 ImageNet, 5.4M параметров

Andrew G. Howard et al., “Searching for MobileNetV3”, pp. 1314-1324, doi: 10.1109/ICCV.2019.00140, 2019

Input	Operator	exp size	#out	SE	NL	s
$224^2 \times 3$	conv2d	-	16	-	HS	2
$112^2 \times 16$	bneck, 3x3	16	16	-	RE	1
$112^2 \times 16$	bneck, 3x3	64	24	-	RE	2
$56^2 \times 24$	bneck, 3x3	72	24	-	RE	1
$56^2 \times 24$	bneck, 5x5	72	40	✓	RE	2
$28^2 \times 40$	bneck, 5x5	120	40	✓	RE	1
$28^2 \times 40$	bneck, 5x5	120	40	✓	RE	1
$28^2 \times 40$	bneck, 3x3	240	80	-	HS	2
$14^2 \times 80$	bneck, 3x3	200	80	-	HS	1
$14^2 \times 80$	bneck, 3x3	184	80	-	HS	1
$14^2 \times 80$	bneck, 3x3	184	80	-	HS	1
$14^2 \times 80$	bneck, 3x3	480	112	✓	HS	1
$14^2 \times 112$	bneck, 3x3	672	112	✓	HS	1
$14^2 \times 112$	bneck, 5x5	672	160	✓	HS	2
$7^2 \times 160$	bneck, 5x5	960	160	✓	HS	1
$7^2 \times 160$	bneck, 5x5	960	160	✓	HS	1
$7^2 \times 160$	conv2d, 1x1	-	960	-	HS	1
$7^2 \times 960$	pool, 7x7	-	-	-	-	1
$1^2 \times 960$	conv2d 1x1, NBN	-	1280	-	HS	1
$1^2 \times 1280$	conv2d 1x1, NBN	-	k	-	-	1

MobileNet V2 + Squeeze and Excite блок

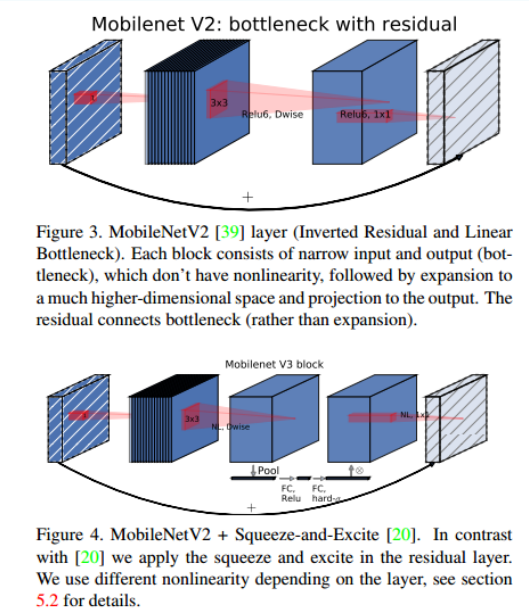


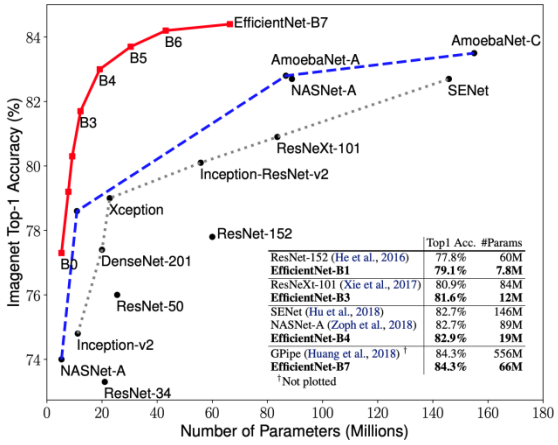
Figure 3. MobileNetV2 [39] layer (Inverted Residual and Linear Bottleneck). Each block consists of narrow input and output (bottleneck), which don’t have nonlinearity, followed by expansion to a much higher-dimensional space and projection to the output. The residual connects bottleneck (rather than expansion).

Figure 4. MobileNetV2 + Squeeze-and-Excite [20]. In contrast with [20] we apply the squeeze and excite in the residual layer. We use different nonlinearity depending on the layer, see section 5.2 for details.

EfficientNet (2019)

Главная идея архитектуры – максимальное увеличение качества с ростом количества параметров. Линейка архитектур – B1, B2, ...B6. Глубина, ширина и разрешение изображений для каждой сети выбирается совместно с помощью формулы исходя из вычислительной стоимости. Масштабирование производится относительно B0

Mingxing Tan, “EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks”, arXiv:1905.11946, 2019



Stage i	Operator $\hat{\mathcal{F}}_i$	Resolution $\hat{H}_i \times \hat{W}_i$	#Channels $\hat{C}_i$	#Layers $\hat{L}_i$
1	Conv3x3	224×224	32	1
2	MBConv1, k3x3	112×112	16	1
3	MBConv6, k3x3	112×112	24	2
4	MBConv6, k5x5	56×56	40	2
5	MBConv6, k3x3	28×28	80	3
6	MBConv6, k5x5	28×28	112	3
7	MBConv6, k5x5	14×14	192	4
8	MBConv6, k3x3	7×7	320	1
9	Conv1x1 & Pooling & FC	7×7	1280	1

EfficientNet (2019)

Mingxing Tan, “EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks”, arXiv:1905.11946, 2019

Model	Top-1 Acc.	Top-5 Acc.	#Params	Ratio-to-EfficientNet	#FLOPs	Ratio-to-EfficientNet
EfficientNet-B0	**77.1%**	**93.3%**	**5.3M**	**1x**	**0.39B**	**1x**
ResNet-50 (He et al., 2016)	76.0%	93.0%	26M	4.9x	4.1B	11x
DenseNet-169 (Huang et al., 2017)	76.2%	93.2%	14M	2.6x	3.5B	8.9x
EfficientNet-B1	**79.1%**	**94.4%**	**7.8M**	**1x**	**0.70B**	**1x**
ResNet-152 (He et al., 2016)	77.8%	93.8%	60M	7.6x	11B	16x
DenseNet-264 (Huang et al., 2017)	77.9%	93.9%	34M	4.3x	6.0B	8.6x
Inception-v3 (Szegedy et al., 2016)	78.8%	94.4%	24M	3.0x	5.7B	8.1x
Xception (Chollet, 2017)	79.0%	94.5%	23M	3.0x	8.4B	12x
EfficientNet-B2	**80.1%**	**94.9%**	**9.2M**	**1x**	**1.0B**	**1x**
Inception-v4 (Szegedy et al., 2017)	80.0%	95.0%	48M	5.2x	13B	13x
Inception-resnet-v2 (Szegedy et al., 2017)	80.1%	95.1%	56M	6.1x	13B	13x
EfficientNet-B3	**81.6%**	**95.7%**	**12M**	**1x**	**1.8B**	**1x**
ResNeXt-101 (Xie et al., 2017)	80.9%	95.6%	84M	7.0x	32B	18x
PolyNet (Zhang et al., 2017)	81.3%	95.8%	92M	7.7x	35B	19x
EfficientNet-B4	**82.9%**	**96.4%**	**19M**	**1x**	**4.2B**	**1x**
SENet (Hu et al., 2018)	82.7%	96.2%	146M	7.7x	42B	10x
NASNet-A (Zoph et al., 2018)	82.7%	96.2%	89M	4.7x	24B	5.7x
AmoebaNet-A (Real et al., 2019)	82.8%	96.1%	87M	4.6x	23B	5.5x
PNASNet (Liu et al., 2018)	82.9%	96.2%	86M	4.5x	23B	6.0x
EfficientNet-B5	**83.6%**	**96.7%**	**30M**	**1x**	**9.9B**	**1x**
AmoebaNet-C (Cubuk et al., 2019)	83.5%	96.5%	155M	5.2x	41B	4.1x
EfficientNet-B6	**84.0%**	**96.8%**	**43M**	**1x**	**19B**	**1x**
EfficientNet-B7	**84.3%**	**97.0%**	**66M**	**1x**	**37B**	**1x**
GPipe (Huang et al., 2018)	84.3%	97.0%	557M	8.4x	-	-

EfficientNet V2 (2021)

Mingxing Tan, “EfficientNetV2: Smaller Models and Faster Training”, arXiv:2104.00298, 2021

	Model	Top-1 Acc.	Params	FLOPs	Infer-time(ms)	Train-time (hours)
ConvNets & Hybrid	EfficientNet-B3 (Tan & Le, 2019a)	81.5%	12M	1.9B	19	10
	EfficientNet-B4 (Tan & Le, 2019a)	82.9%	19M	4.2B	30	21
	EfficientNet-B5 (Tan & Le, 2019a)	83.7%	30M	10B	60	43
	EfficientNet-B6 (Tan & Le, 2019a)	84.3%	43M	19B	97	75
	EfficientNet-B7 (Tan & Le, 2019a)	84.7%	66M	38B	170	139
	RegNetY-8GF (Radosavovic et al., 2020)	81.7%	39M	8B	21	-
	RegNetY-16GF (Radosavovic et al., 2020)	82.9%	84M	16B	32	-
	ResNeSt-101 (Zhang et al., 2020)	83.0%	48M	13B	31	-
	ResNeSt-200 (Zhang et al., 2020)	83.9%	70M	36B	76	-
	ResNeSt-269 (Zhang et al., 2020)	84.5%	111M	78B	160	-
	TResNet-L (Ridnik et al., 2020)	83.8%	56M	-	45	-
	TResNet-XL (Ridnik et al., 2020)	84.3%	78M	-	66	-
	EfficientNet-X (Li et al., 2021)	84.7%	73M	91B	-	-
	NFNet-F0 (Brock et al., 2021)	83.6%	72M	12B	30	8.9
	NFNet-F1 (Brock et al., 2021)	84.7%	133M	36B	70	20
	NFNet-F2 (Brock et al., 2021)	85.1%	194M	63B	124	36
	NFNet-F3 (Brock et al., 2021)	85.7%	255M	115B	203	65
	NFNet-F4 (Brock et al., 2021)	85.9%	316M	215B	309	126
	LambdaResNet-420-hybrid (Bello, 2021)	84.9%	125M	-	-	67
	BotNet-T7-hybrid (Srinivas et al., 2021)	84.7%	75M	46B	-	95
	BiT-M-R152x2 (21k) (Kolesnikov et al., 2020)	85.2%	236M	135B	500	-
Vision Transformers	ViT-B/32 (Dosovitskiy et al., 2021)	73.4%	88M	13B	13	-
	ViT-B/16 (Dosovitskiy et al., 2021)	74.9%	87M	56B	68	-
	DeiT-B (ViT+reg) (Touvron et al., 2021)	81.8%	86M	18B	19	-
	DeiT-B-384 (ViT+reg) (Touvron et al., 2021)	83.1%	86M	56B	68	-
	T2T-ViT-19 (Yuan et al., 2021)	81.4%	39M	8.4B	-	-
	T2T-ViT-24 (Yuan et al., 2021)	82.2%	64M	13B	-	-
	ViT-B/16 (21k) (Dosovitskiy et al., 2021)	84.6%	87M	56B	68	-
	ViT-L/16 (21k) (Dosovitskiy et al., 2021)	85.3%	304M	192B	195	172
ConvNets (ours)	**EfficientNetV2-S**	83.9%	22M	8.8B	24	7.1
	EfficientNetV2-M	85.1%	54M	24B	57	13
	EfficientNetV2-L	85.7%	120M	53B	98	24
	EfficientNetV2-S (21k)	84.9%	22M	8.8B	24	9.0
	EfficientNetV2-M (21k)	86.2%	54M	24B	57	15
	EfficientNetV2-L (21k)	86.8%	120M	53B	98	26
	EfficientNetV2-XL (21k)	87.3%	208M	94B	-	45

We do not include models pretrained on non-public Instagram/JFT images, or models with extra distillation or ensemble

Аугментация данных в задаче распознавания изображений

Обогащение тренировочного набора синтетическими примерами для увеличения обобщающих способностей модели:

- Повороты, растяжения, искривления, зеркальные отображения
- Внесение случайного шума, увеличение контрастности
- Вырезание случайной части из изображения (crop augmentation)
- Cutmix – изображение составляется из кусков двух изображений, метки каждого из изображений берутся пропорционально занимаемой площади
- MixUp – линейная интерполяция двух изображений
- Создание новых примеров с помощью генеративных состязательных сетей (Generative Adversarial Networks, GAN)

Перенос обучения

Методика улучшения качества работы модели и ускорения сходимости обучения для новой задачи

Сеть предварительно обучается на большем наборе данных (например, ImageNet – такие модели есть в открытом доступе). Далее к свёрточным слоя сети присоединяется «голова» под новую задачу. Две стратегии:

- Полученная сеть доучивается на новой задаче целиком
- Сверточные слои «замораживаются», учится только «голова» сети

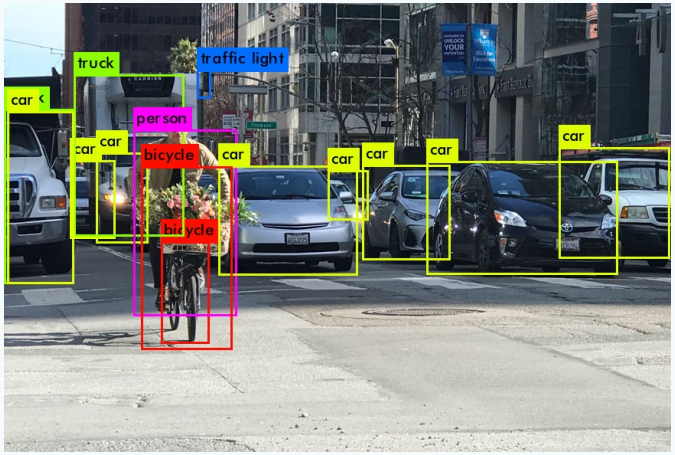
Задача детектирования

Цель – найти все объекты на изображении (указать прямоугольник локализации) и классифицировать их

Архитектуры на основе свёрточных НС:

- RCNN, Fast RCNN, Faster RCNN
- YOLO v1, v2, v3, v4, v5

Хороший обзор: https://machinelearningmastery.com/object-recognition-with-deep-learning/



https://dagshub.com/blog/yolov6/

Задача сегментации

Цель – найти все объекты на изображении (указать сегмент, занимаемый объектом) и классифицировать их

Архитектуры на основе свёрточных НС:

- Mask RCNN
- UNET

Хороший обзор: https://towardsdatascience.com/understanding-semantic-segmentation-with-unet-6be4f42d4b47

https://syncedreview.com/2019/12/26/facebook-pointrend-rendering-image-segmentation/

Спасибо за внимание!

Лаборатория адаптивных методов обработки данных

Учебный курс

«Машинное обучение в физике»

Занятие №11 (лекция)

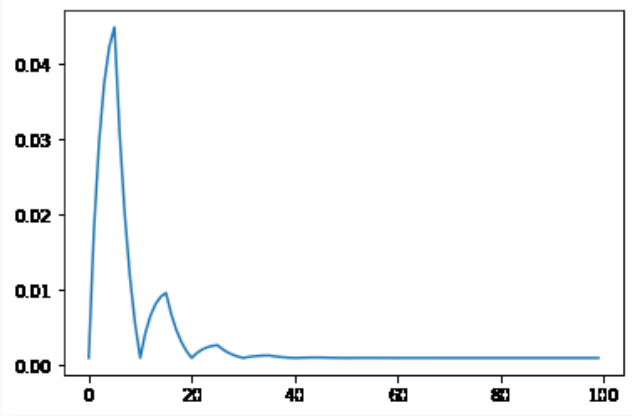
Некоторые технологии работы с глубокими сетями

Авторы курса:

С.А. Доленко, И.М. Гаджиев, А.О. Ефиторов, И.В. Исаев, В.Р. Широкий

Обучение нейронных сетей

Техники для улучшения результатов обучения глубоких сетей:



1) Спадающая косинусоидная (циклическая) скорость обучения

2) Сглаживание значений классов (целевой класс имеет желаемый выход 0.9, остальные - равномерно распределенное малое значение)

3) Уменьшение стандартного значения распада весов (< 0.0001) при использовании других методов регуляризации

https://arxiv.org/pdf/2103.07579.pdf

Обучение нейронных сетей

4) Xavier инициализация весов:

случайные значения внутри границ: $\pm \frac{\sqrt{6}}{\sqrt{n_i + n_{i+1}}}$

где n_i и n_{i+1} - количество “входящих” и “исходящих” связей относительно слоя сети

5) Использование функций активации (Scaled\Leaky) ReLU

$ScReLU_{\alpha,\beta}(x) = \beta \max\{x + \alpha, 0\}$ $LReLU(x) = \max\{x, \alpha x\}$, $0<\alpha<1$

6) Ограничение амплитуды градиента

http://proceedings.mlr.press/v9/glorot10a/glorot10a.pdf

Функция правдоподобия

D = {t} - данные наблюдений

w - параметры модели

p(w) - априорная вероятностная модель (начальная инициализация модели)

p(w|D) - апостериорная вероятность (оценка неопределенности после обучения модели)

p(D|w) - функция правдоподобия (оценивает, насколько соответствуют наблюдаемые примеры параметрам модели)

По теореме Байеса

$$p(\mathbf{w}|\mathcal{D}) = \frac{p(\mathcal{D}|\mathbf{w})p(\mathbf{w})}{p(\mathcal{D})}$$

p(D) - нормализующая константа, гарантирующая, что p(w|D) - валидная апостериорная вероятность

Цель: подобрать такой набор параметров модели **w**, который максимизирует функцию правдоподобия (её логарифм)

Обучение полиномиальной модели

Предполагаем,что данные распределены нормально:

$$p(t|x, \mathbf{w}, \beta) = \mathcal{N}\left(t|y(x, \mathbf{w}), \beta^{-1}\right)$$

y(x, w) - среднее значение целевых значений t

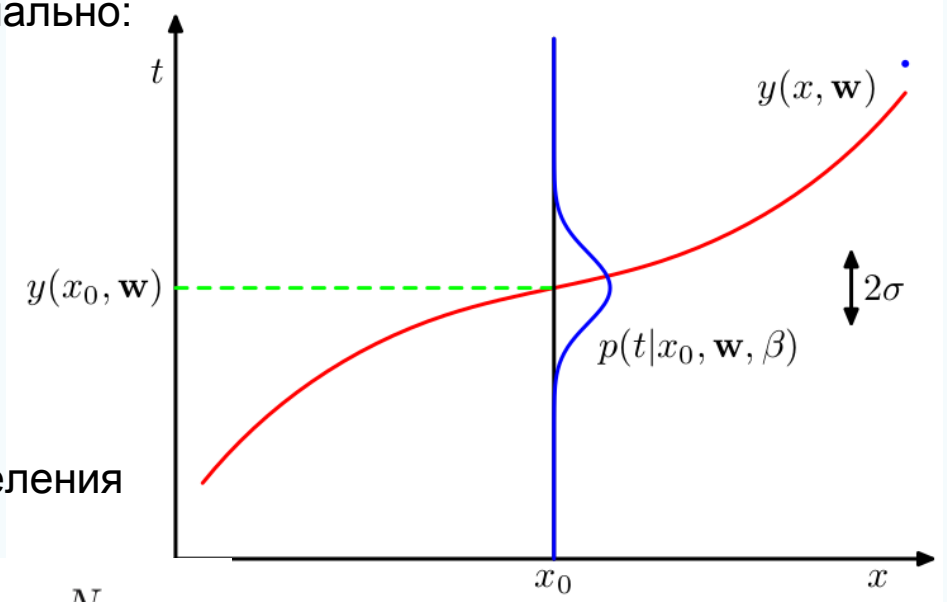
$\beta = 1/\sigma^2$ - обратное значение вариации

Предполагая независимость (n) примеров:

$$p(\mathbf{t}|\mathbf{x}, \mathbf{w}, \beta) = \prod_{n=1}^{N} \mathcal{N}\left(t_n|y(x_n, \mathbf{w}), \beta^{-1}\right)$$

Возьмем логарифм от многомерного распределения Гаусса, расписав его по определению:

$$\ln p(\mathbf{t}|\mathbf{x}, \mathbf{w}, \beta) = -\frac{\beta}{2}\sum_{n=1}^{N}\{y(x_n, \mathbf{w}) - t_n\}^2 + \frac{N}{2}\ln\beta - \frac{N}{2}\ln(2\pi)$$



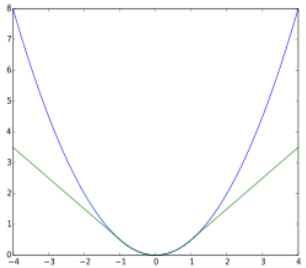
Исключив константные члены, не зависящие от *w,* и нормируя на β, получаем **Квадратичную Ошибку**: $(y(x_n, w) - t_n)^2$

Функции потерь

Средняя абсолютная ошибка (L1): $$MAE = \frac{\sum_{i=1}^{n} | y_i - \hat{y}_i |}{n}$$

Средний квадрат ошибки (L2): $$MSE = \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{n}$$

Функция потерь Хьюбера: $$L = \begin{cases} (y_i - \hat{y}_i)^2/2\alpha + \alpha/2 \text{ , если } | y_i - \hat{y}_i | \le \alpha \\ | y_i - \hat{y}_i | \text{ , если } | y_i - \hat{y}_i | > \alpha \end{cases}$$



https://arxiv.org/pdf/2108.12627.pdf

Максимизация апостериорной вероятности

Предположим, что априорное распределение весов полиномиальной модели порядка *M* подчиняется нормальному распределению (α - обратное значение вариации):

$$p(\mathbf{w}|\alpha) = \mathcal{N}(\mathbf{w}|\mathbf{0}, \alpha^{-1}\mathbf{I}) = \left(\frac{\alpha}{2\pi}\right)^{(M+1)/2} \exp\left\{-\frac{\alpha}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w}\right\}$$

В этом случае можно определить w, максимизируя апостериорную вероятность *p(w|x,t, α, β)*, которая из теоремы Байеса:

$$p(\mathbf{w}|\mathbf{x}, \mathbf{t}, \alpha, \beta) \propto p(\mathbf{t}|\mathbf{x}, \mathbf{w}, \beta)p(\mathbf{w}|\alpha)$$

Логарифмируем и используем ранее полученную оценку правдоподобия:

$$\frac{\beta}{2}\sum_{n=1}^{N}\{y(x_n, \mathbf{w}) - t_n\}^2 + \frac{\alpha}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w}$$

получаем L2 регуляризацию!

Регуляризация: L0, L1, L2

L1 (Лассо, Lasso):

$$Error\,(y, \hat{y}) + \lambda \sum_{i=1}^{i=n} \|W_i\|$$

L2 (Гребневая, Ridge)

$$Error\,(y, \hat{y}) + \lambda \sum_{i=1}^{i=n} W_i^2$$

L1+L2 (Эластичная, Elastic Net):

$$Error\,(y, \hat{y}) + r\lambda \sum_{i=1}^{i=n} \|W_i\| + \frac{1-r}{2}\lambda \sum_{i=1}^{i=n} W_i^2$$

Кросс-энтропия

C_k - бинарный вектор классов (классы независимы)

φ_n - вектор входных признаков (примеры независимы)

t_n - целевые значения (T - матрица размерности N*K)

Функция правдоподобия:

$$p(\mathbf{T}|\mathbf{w}_1,\ldots,\mathbf{w}_K) = \prod_{n=1}^{N}\prod_{k=1}^{K} p(\mathcal{C}_k|\boldsymbol{\phi}_n)^{t_{nk}} = \prod_{n=1}^{N}\prod_{k=1}^{K} y_{nk}^{t_{nk}}$$

Возьмем отрицательный логарифм:

$$E(\mathbf{w}_1,\ldots,\mathbf{w}_K) = -\ln p(\mathbf{T}|\mathbf{w}_1,\ldots,\mathbf{w}_K) = -\sum_{n=1}^{N}\sum_{k=1}^{K} t_{nk}\ln y_{nk}$$

Получаем функцию ***кросс-энтропии***

Относительная энтропия (расстояние Кульбака-Лейблера)

кросс-энтропия между распределениями p(x) и q(x)

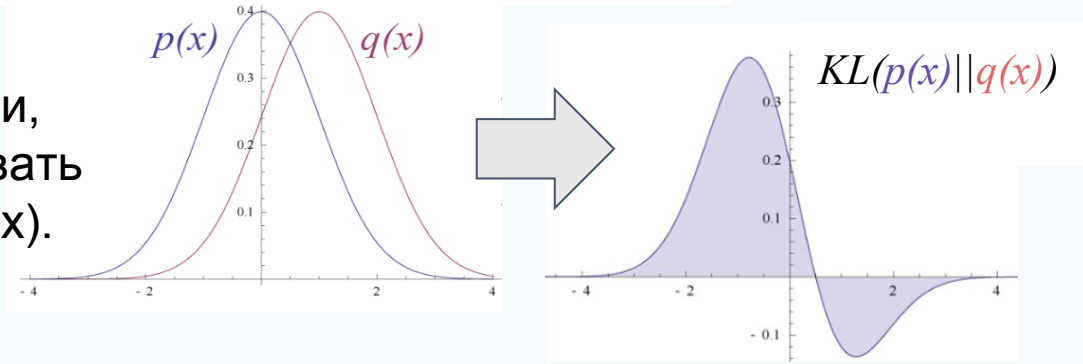
энтропия распределения p(x)

$$\mathrm{KL}(p\|q) = \overbrace{-\int p(\mathbf{x}) \ln q(\mathbf{x})\, \mathrm{d}\mathbf{x}} - \overbrace{\left(-\int p(\mathbf{x}) \ln p(\mathbf{x})\, \mathrm{d}\mathbf{x}\right)}$$

$$= -\int p(\mathbf{x}) \ln \left\{ \frac{q(\mathbf{x})}{p(\mathbf{x})} \right\} \mathrm{d}\mathbf{x}.$$

Несимметрична:
KL(p||q) != KL(q||p)

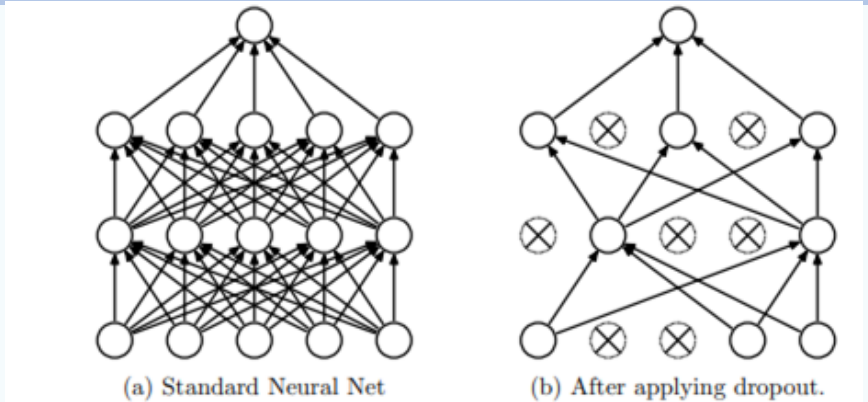
Смысл:
количество неучтённой информации, если распределение q(x) использовать как приближение распределения p(x).



Другие методы регуляризации

Dropout:

“Отключение” нейронов нейронной сети и всех их связей с заданной вероятностью


(a) Standard Neural Net

(b) After applying dropout.

Нормализация пакетов:

Слой нейронной сети, который выполняет шкалирование значений по пакету данных:

$$\widehat{x}^{(k)} = \frac{x^{(k)} - \mathrm{E}[x^{(k)}]}{\sqrt{\mathrm{Var}[x^{(k)}]}}$$

Имеет обучаемые параметры γ и β:

$$y^{(k)} = \gamma^{(k)}\widehat{x}^{(k)} + \beta^{(k)}.$$

https://jmlr.org/papers/volume15/srivastava14a/srivastava14a.pdf

Интерпретация решений нейронных сетей

Почему нейронная сеть принимает те или иные решения?

Интерпретируемое представление данных:

- $x \in R^d$ - оригинальное представление данных (входные признаки)
- $x' \in \{0, 1\}^{d'}$ - интерпретируемое представление в виде бинарных векторов

$$\xi(x) = \underset{g \in G}{\operatorname{argmin}} \; \mathcal{L}(f, g, \pi_x) + \Omega(g)$$

объяснение $g \in G$, где G - класс потенциально интерпретируемых моделей

$g \in \{0, 1\}^{d'}$ - является ли пример интерпретируемым

$\Omega(g)$ - сложность “объяснения”

(глубина дерева, число ненулевых компонентов линейной модели и т.п.)

$\pi_x(z)$ - мера близости между примерами z (интерпретируемый) и x (другой)

f(x) в случае классификации - вероятность принадлежности x к классам

L – локальная функция потерь

https://arxiv.org/pdf/1602.04938.pdf

Разреженное линейное объяснение

Для классификатора изображений:

$$\mathcal{L}(f, g, \pi_x) = \sum_{z, z' \in \mathcal{Z}} \pi_x(z) \left(f(z) - g(z')\right)^2$$

$g(z') = w_g \cdot z'$ (g - экземпляр класса линейных моделей G)

$\pi_x(z) = \exp(-D(x, z)^2 / \sigma^2)$, где D - функция расстояния L2

K - суперпиксели для данного изображения

Algorithm 1 Sparse Linear Explanations using LIME

Require: Classifier f, Number of samples N
Require: Instance x, and its interpretable version x'
Require: Similarity kernel π_x, Length of explanation K

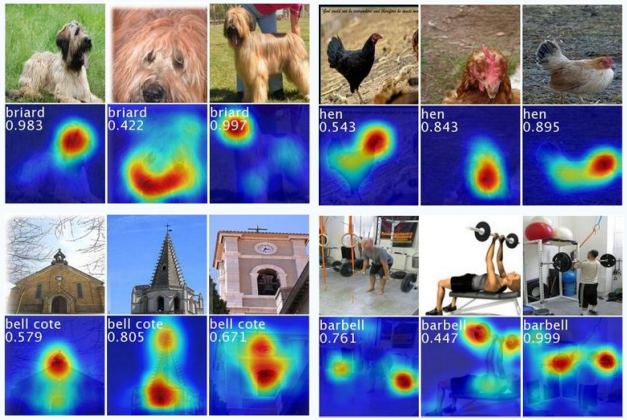
$\mathcal{Z} \leftarrow \{\}$
for $i \in \{1, 2, 3, ..., N\}$ **do**
  $z'_i \leftarrow sample\_around(x')$
  $\mathcal{Z} \leftarrow \mathcal{Z} \cup \langle z'_i, f(z_i), \pi_x(z_i) \rangle$
end for
$w \leftarrow$ K-Lasso$(\mathcal{Z}, K)$ ▷ with z'_i as features, $f(z)$ as target
return w

http://proceedings.mlr.press/v139/garreau21a/garreau21a.pdf

Отображение активации класса

Для сверточных сетей с Global Average Pooling (GAP) слоем предсказание Y^c для класса *c* перепишем как линейную комбинацию карт признаков последнего сверточного слоя A^k:

$$Y^c = \sum_k w_k^c . \sum_i \sum_j A_{ij}^k$$



Для каждого пространственного признака (i,j) класс-специфичная карта значимости L^c:

$$L_{ij}^c = \sum_k w_k^c . A_{ij}^k$$

Вес w^c_k - вес линейной модели (собственной для каждого класса), натренированной решать задачу классификации на основе этих признаков

CAM – Class Activation Mapping

https://arxiv.org/pdf/1512.04150v1.pdf

Градиент отображения активации класса

- A^k не имеют сходную нормализацию\площадь признаков:
- некоторые активации исчезнут в рамках простого суммирования:

$$w_k^c = \frac{1}{Z}\sum_i\sum_j \frac{\partial Y^c}{\partial A_{ij}^k}$$

Решение: взять взвешенное среднее попиксельных градиентов:

$$L_{ij}^c = \sum_k w_k^c . A_{ij}^k$$

где alpha = 0 при отсутствии объекта класса *c*:

$$w_k^c = \sum_i\sum_j \alpha_{ij}^{kc} . relu(\frac{\partial Y^c}{\partial A_{ij}^k})$$

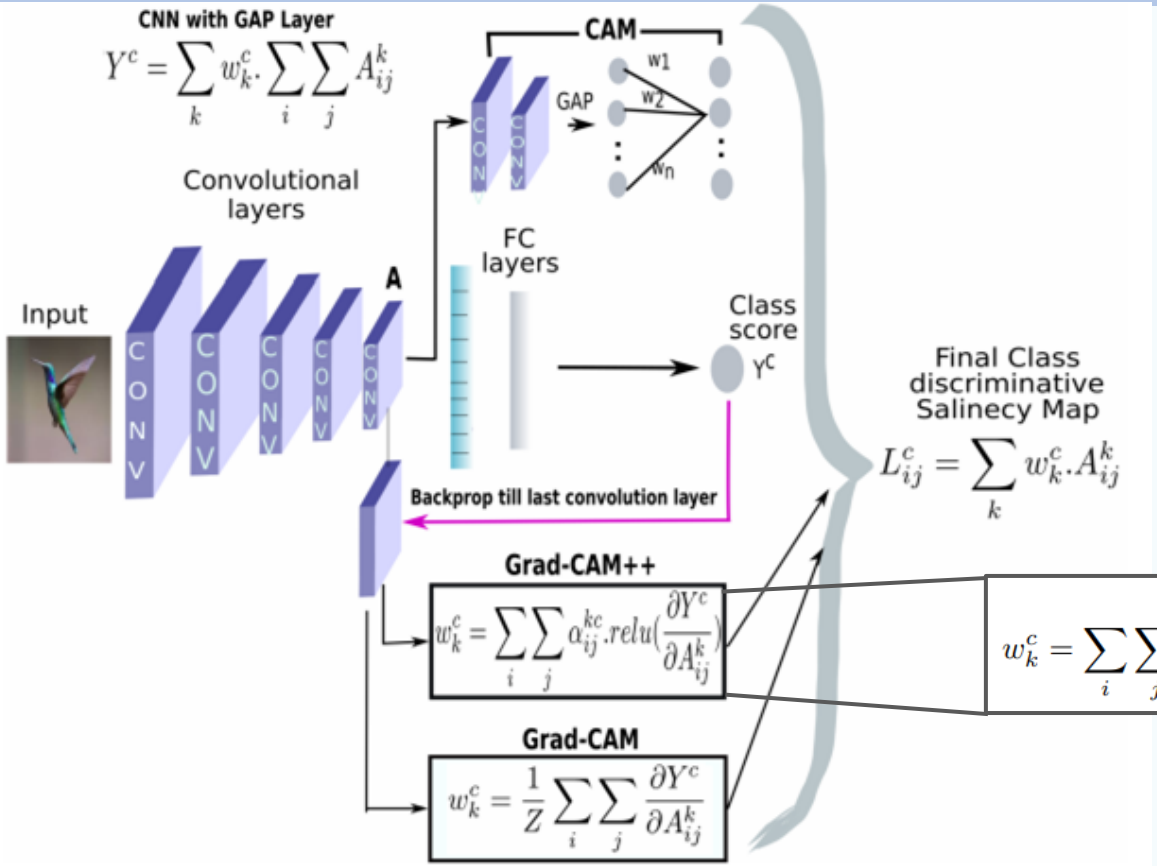
$$\alpha_{ij}^{kc} = \frac{1}{\sum_{l,m} \frac{\partial y^c}{\partial A_{lm}^k}} \quad if \quad \frac{\partial y^c}{\partial A_{ij}^k} = 1$$

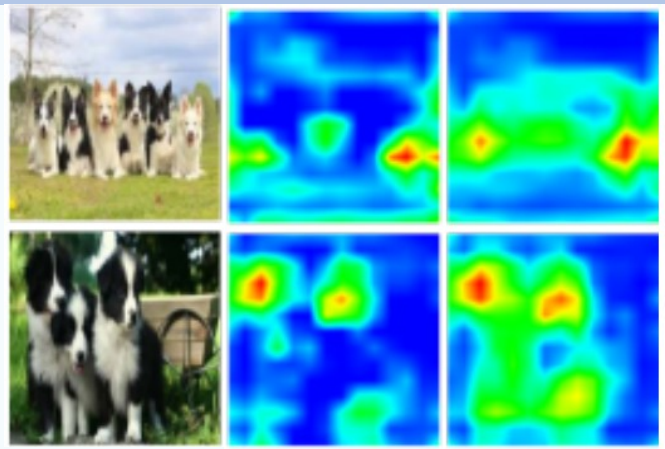
$$= 0 \quad otherwise$$

Grad-CAM

https://arxiv.org/pdf/1610.02391.pdf

Методы отображения активации классов





GradCAM

GradCAM^{++}

$$w_k^c = \sum_i \sum_j [\frac{\frac{\partial^2 Y^c}{(\partial A_{ij}^k)^2}}{2\frac{\partial^2 Y^c}{(\partial A_{ij}^k)^2} + \sum_a \sum_b A_{ab}^k \{\frac{\partial^3 Y^c}{(\partial A_{ij}^k)^3}\}}] . relu(\frac{\partial Y^c}{\partial A_{ij}^k})$$

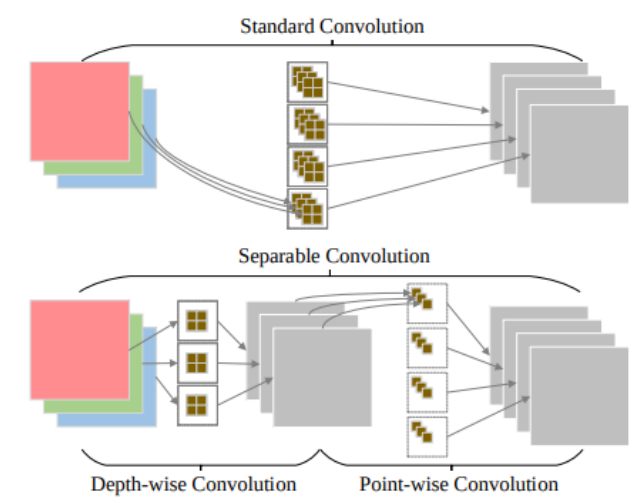
https://arxiv.org/pdf/1710.11063.pdf

“Сжатие” нейронных сетей

https://arxiv.org/pdf/2004.09602.pdf

- квантизация моделей (fp32 -> int8):
- квантизация “вычислительно интенсивных” слоев
- сохранение значений наиболее чувствительных к квантизации слоев в формате чисел с плавающей запятой
- дообучение квантизованной модели

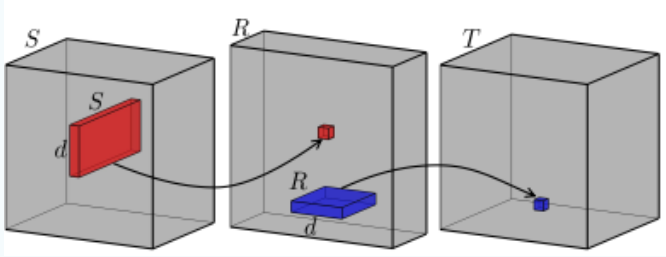
- отделимые по глубине свертки (MobileNet):



https://arxiv.org/pdf/2101.09671.pdf

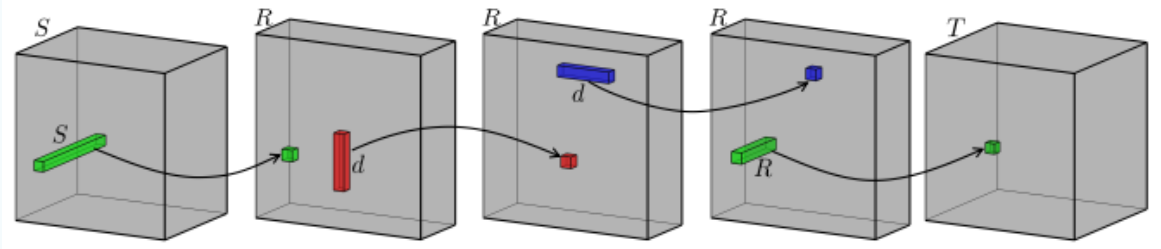
“Сжатие” нейронных сетей

- Факторизация фильтров:



https://arxiv.org/pdf/1405.3866.pdf

- CP (CANDECOMP/PARAFAC) декомпозиция:



https://arxiv.org/pdf/1412.6553.pdf

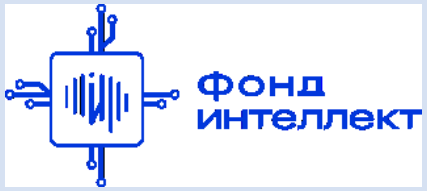
Зоопарк бинаризованных моделей

Type		Method	Key Tech.	Tricks			
				ST	OP	AQ	GA
Naive Binary Neural Networks		BinaryConnect [59]	FP: $\mathtt{sign}(x)$ BP: STE	-	A	-	-
		Bitwise Neural Networks [69]		-	-	-	-
		Binarized Neural Networks [57]		-	AM	-	-
Optimization Based Binary Neural Networks	Minimize the Quantization Error	Binary Weight Networks [57]	$J(\mathbf{b},\alpha)=\Vert\mathbf{x}-\alpha\mathbf{b}\Vert^2$ $\alpha^*,\mathbf{b}^*=\arg\min_{\alpha,\mathbf{b}} J(\mathbf{b},\alpha)$	-	S	-	-
		XNOR-Net [58]		RB+RP	A	-	-
		DoReFa-Net [60]		-	A	-	-
		High-Order Residual Quantization [70]		-	A	-	-
		ABC-Net [71]		-	S	-	-
		Two-Step Quantization [72]		RB	-	-	-
		Binary Weight Networks via Hashing [73]		-	S	-	-
		PArameterized Clipping acTivation [74]		-	A	-	-
		LQ-Nets [61]		RB	-	-	-
		Wide Reduced-Precision Networks [75]		WD	A	-	-
		XNOR-Net++ [76]		-	A	-	-
		Learning Symmetric Quantization [77]		-	-	✓	-
		BBG [78]		SC	-	-	-
		Real-to-Bin [79]		SC	A	-	✓
	Improve Network Loss Function	Distilled Binary Neural Network [80]	$\mathcal{L}^b_{\text{total}}=\mathcal{L}^b_{\text{original}}+\lambda\mathcal{L}^b_{\text{Customized}}$	-	S	-	-
		Distillation and Quantization [81]		-	S	-	-
		Apprentice [82]		-	-	-	-
		Loss-Aware Binarization [83]		-	A	-	-
		Incremental Network Quantization [84]		-	S	✓	-
		BNN-DL [85]		-	R	-	✓
		CI-BCNN [86]		-	R	-	✓
		Main/Subsidiary Network [87]		RB	-	-	-
	Reduce the Gradient Error	Bi-Real Net [62]	Customized `ApproxFunc` (FP) or `QuantFunc` (BP) or `UpdateFunc` (BP)	SC	S	-	✓
		Circulant Binary Convolutional Networks [88]		SC	S	-	✓
		Half-wave Gaussian Quantization [89]		RB	S	-	✓
		BNN+ [90]		RB	A	-	✓
		Differentiable Soft Quantization [63]		-	A	-	✓
		BCGD [91]		-	-	-	✓
		ProxQuant [92]		-	A	-	✓
		Quantization Networks [93]		-	S	-	✓
		Self-Binarizing Networks [94]		-	A	-	✓
		Improved Training BNN [95]		-	A	-	✓
		IR-Net [96]		-	S	✓	✓

https://arxiv.org/pdf/2004.03333.pdf

Спасибо за внимание!

Лаборатория адаптивных методов обработки данных



Учебный курс

«**Машинное обучение в физике**»

Занятие №12 (лекция)

Рекуррентные нейронные сети

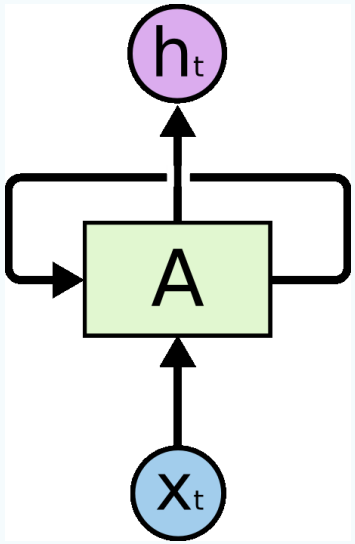
Авторы курса:

С.А. Доленко, И.М. Гаджиев, А.О. Ефиторов, И.В. Исаев, В.Р. Широкий

Скрытое состояние рекуррентной НС

Рекуррентная нейронная сеть (РНС) = Recurrent Neural Network (RNN)

- Скрытое состояние сети зависит
  не только от состояния входа,
  но и от предыдущего скрытого состояния
- Обработка последовательности векторов x
  через использование предыдущего состояния
  на каждом временном шаге:



$$h_t = f_W(h_{t-1}, x_t)$$

новое состояние

функция с параметрами w

старое состояние

вектор входных данных на шаге t

рисунок: http://colah.github.io/posts/2015-08-Understanding-LSTMs

Выходное значение рекуррентной НС

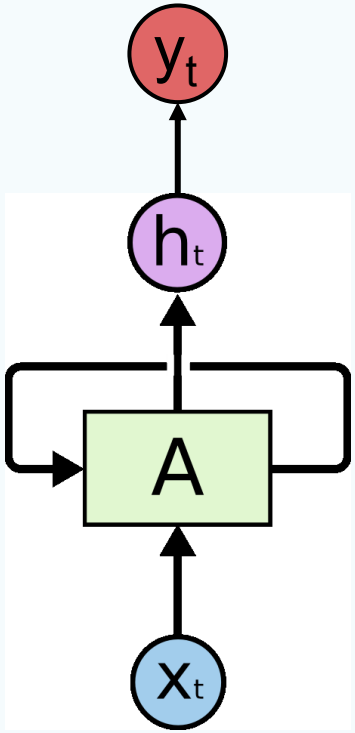
Вектор выходных данных рассчитывается на дополнительном обучаемом слое РНС, принимающим на вход скрытое состояние:

$$y_t = f_{W_{hy}}(h_t)$$

вектор выходных данных

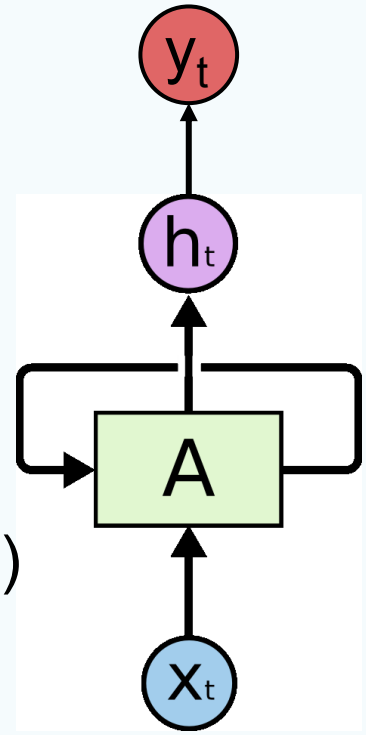
функция с параметрами W_{hy}

новое состояние



Классическая рекуррентная нейронная сеть

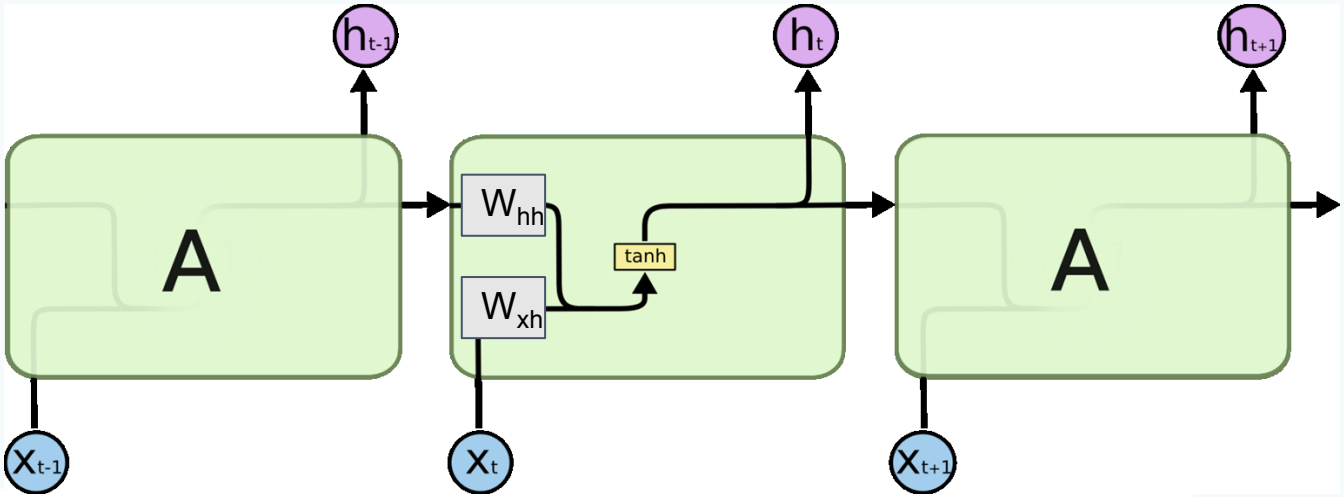
“Скрытое состояние” сети представлено в виде единственного вектора ***h***:



$$h_t = f_W(h_{t-1}, x_t) \longrightarrow h_t = \tanh(W_{hh}\, h_{t-1} + W_{xh}\, x_t)$$

$$y_t = f_{W_{hy}}(h_t) \longrightarrow y_t = W_{hy}\, h_t$$

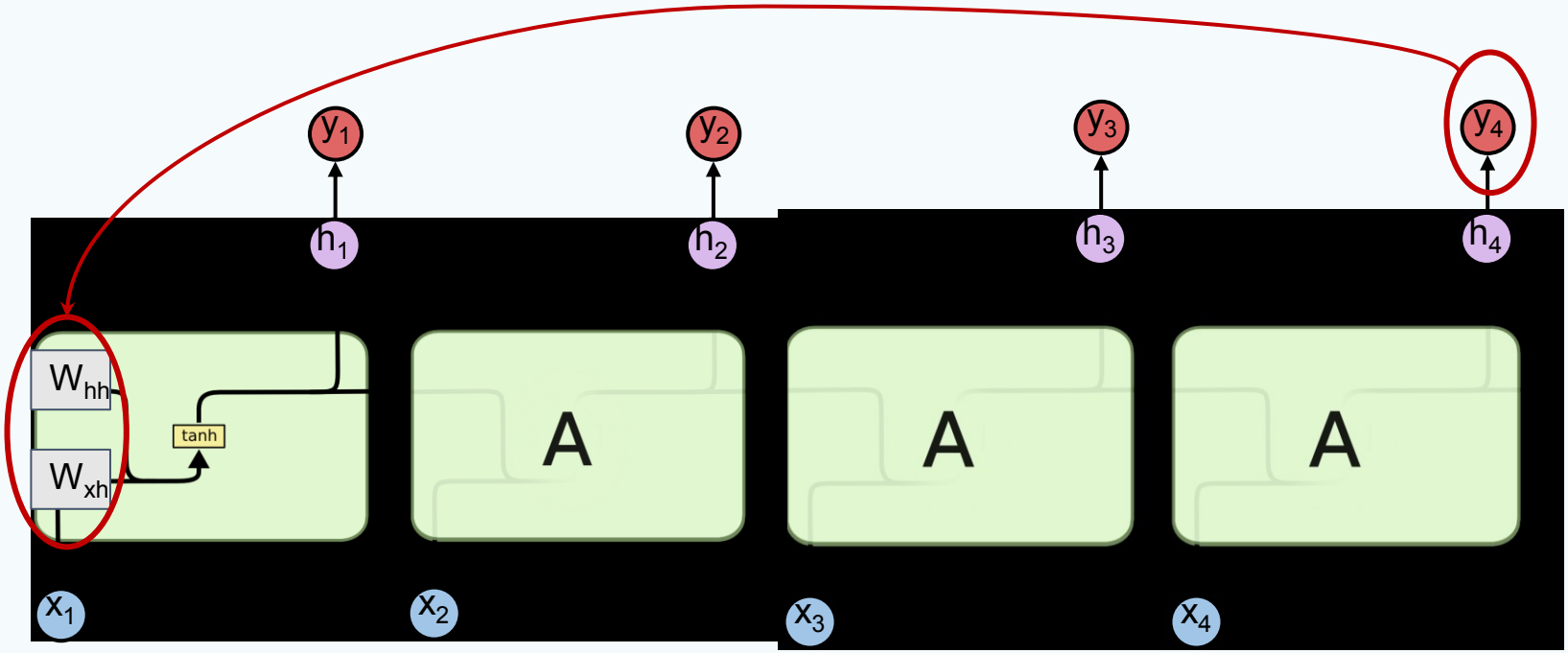
Распространение градиента в RNC



$$\frac{\partial h_t}{\partial h_{t-1}} = \tanh'\left(W_{hh} h_{t-1} + W_{xh} x_t\right) W_{hh}$$

почти всегда < 1

Градиент сквозь время

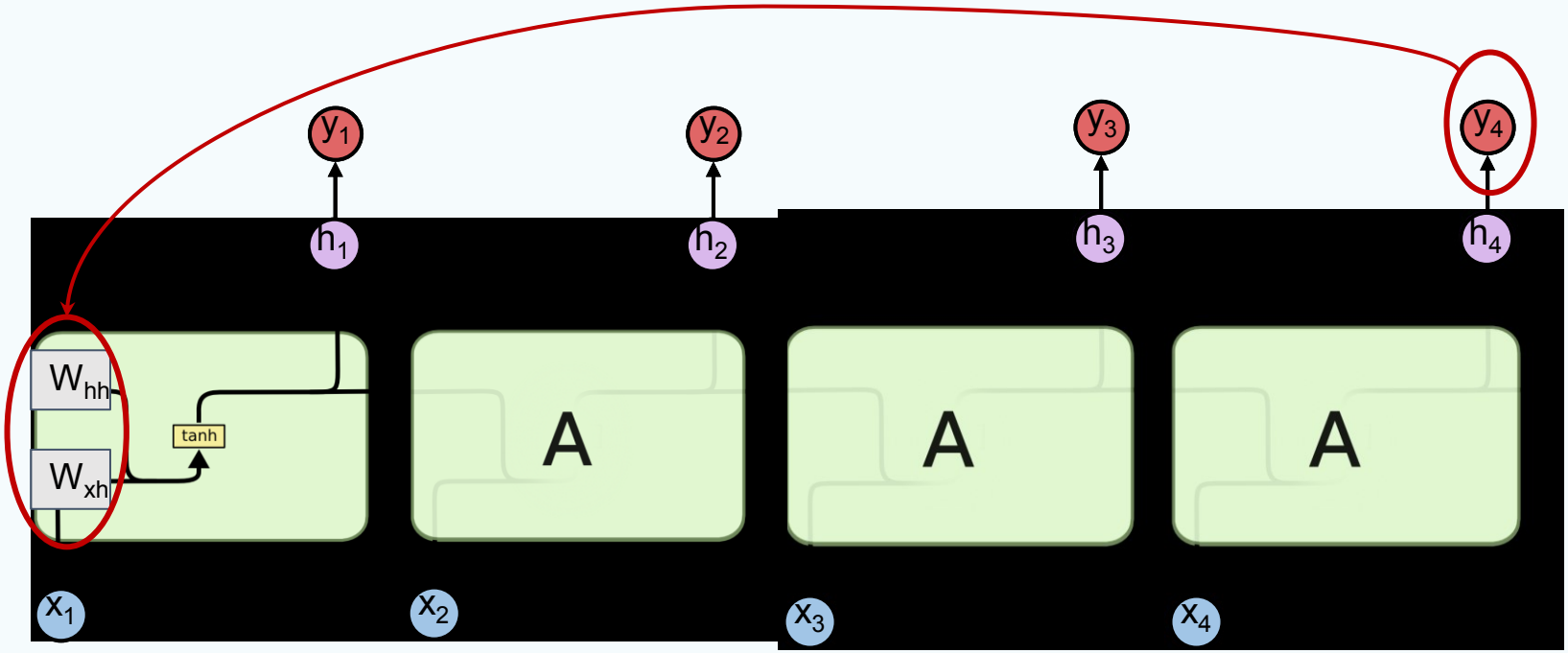


$$\frac{\partial L}{\partial W} = \sum_{t=1}^{T} \frac{\partial L_t}{\partial W}$$, где L - функция потерь

заменим по формуле предыдущего слайда!

$$\frac{\partial L_T}{\partial W} = \frac{\partial L_T}{\partial h_T} \frac{\partial h_t}{\partial h_{t-1}} \cdots \frac{\partial h_1}{\partial W} = \frac{\partial L_T}{\partial h_T} \left(\prod_{t=2}^{T} \frac{\partial h_t}{\partial h_{t-1}}\right) \frac{\partial h_1}{\partial W}$$

Градиент сквозь время

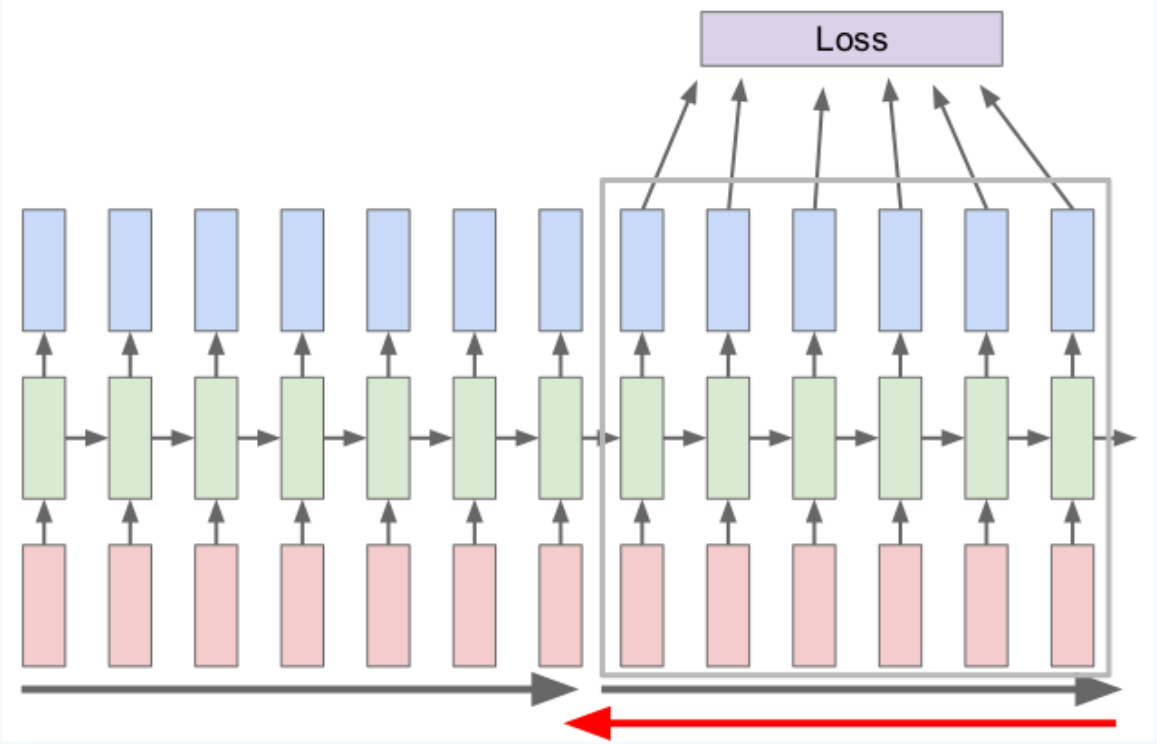


$$\frac{\partial L_t}{\partial W} = \frac{\partial L_t}{\partial h_t} \prod^{T-1} W_{hh} \frac{\partial h_t}{\partial W}$$

- наибольшее сингулярное значение < 1: затухающий градиент
- наибольшее сингулярное значение > 1: взрывающийся градиент

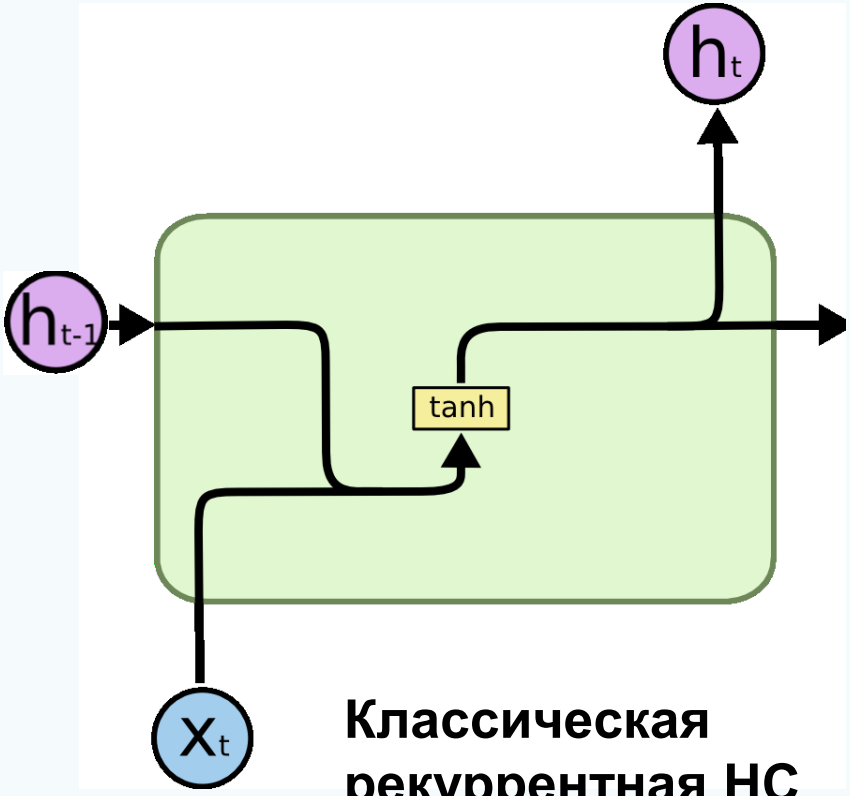
Усеченное распространение градиента

В классической рекуррентной нейронной сети (RNN) предсказания ошибки и корректировки весов сети рассчитываются на основе всей последовательности

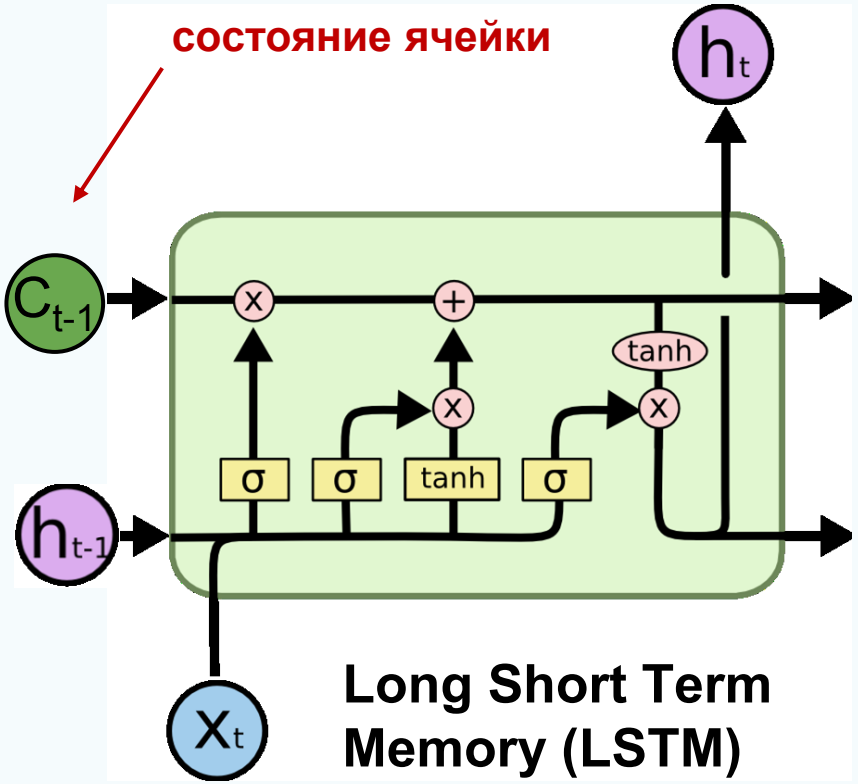


На практике более эффективный подход - работать с подпоследовательностями данных, тут же производя корректировку весов!

Long Short Term Memory (LSTM)



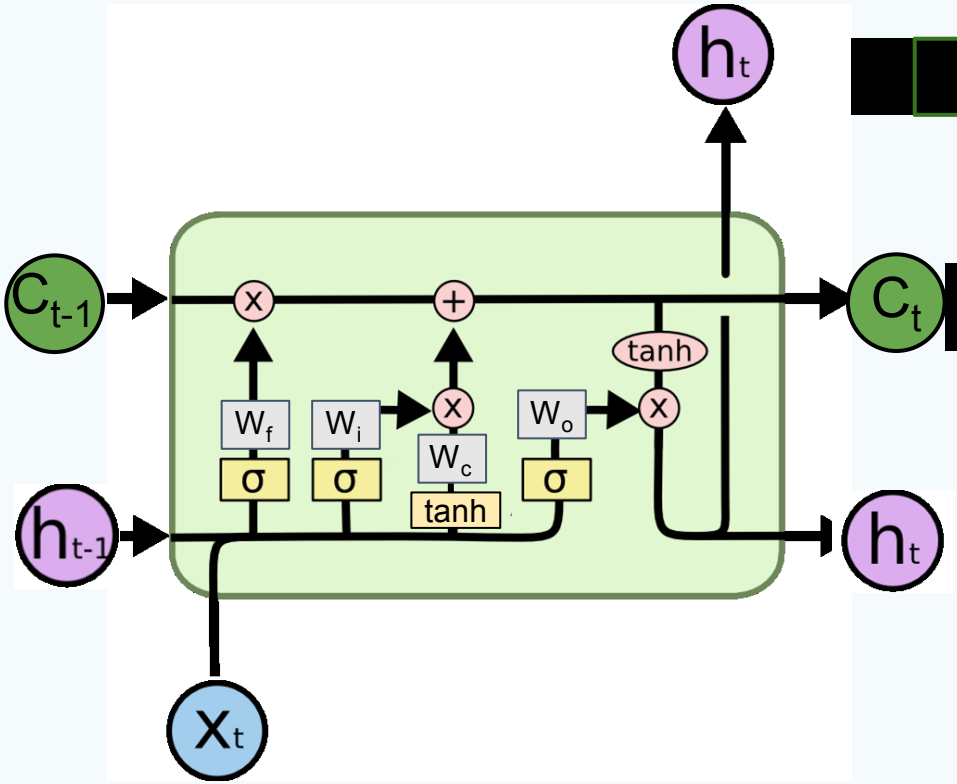
Классическая рекуррентная НС



Long Short Term Memory (LSTM)

рисунок: http://colah.github.io/posts/2015-08-Understanding-LSTMs

Веса ячейки LSTM



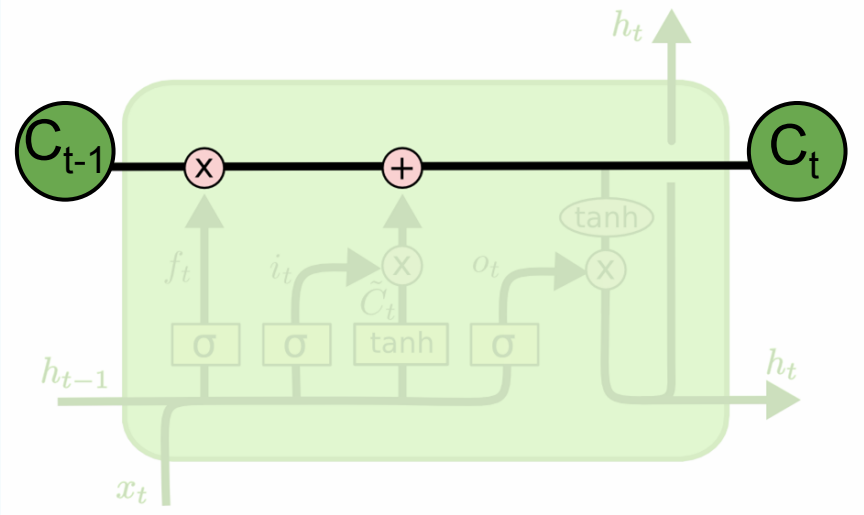
[redacted]

$$h_t = o_t * \tanh(C_t)$$

[redacted]

Распространение градиента в LSTM

Алгоритм обратного
распространения ошибки
более устойчив,
т.к. преобразование из C_t в C_{t-1}
предполагает только
поэлементное умножение с f_t
и нет взаимодействия
с матрицами весов:



$$c_t = f \odot c_{t-1} + i \odot g$$

У LSTM нет проблем с градиентом?

LSTM архитектура облегчает основную задачу
рекуррентной сети о сохранении информации
на протяжении множества последовательных шагов:

- например, если $f = 1$ и $i = 0$,
  тогда информация будет сохранена гарантированно
- для простой рекуррентной сети сложнее обучить матрицу W_h,
  ответственную за сохранение информации
  о внутреннем состоянии

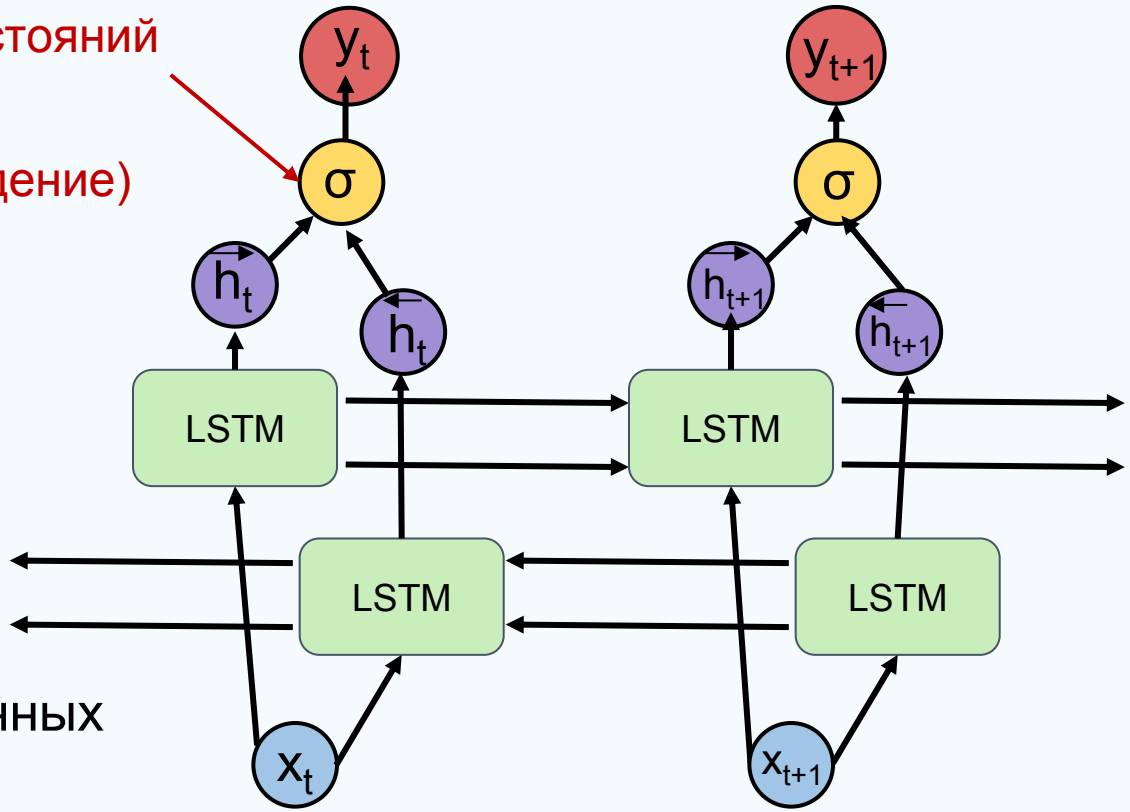
Однако, LSTM архитектура не гарантирует
отсутствия проблем с затуханием и взрывом градиентов

Двунаправленная сеть LSTM

Используется, например, для заполнения пропусков

операция соединения состояний разных направлений (стекинг, сумма, произведение)

Одновременное распространение информации в обоих направлениях последовательности данных



Механизм внимания (attention)

$y_t = decoder(c_t, y_{t-1}, h_{t-1})$

CNN

H*W*D

$z_{0,0}$ $z_{0,1}$ $z_{0,2}$
$z_{1,0}$ $z_{1,1}$ $z_{1,2}$
$z_{2,0}$ $z_{2,1}$ $z_{2,2}$

h_0

Кот за цветами [Конец]

y_1 y_2 y_3 y_4

decoder decoder decoder decoder

h_1 h_2 h_3 h_4

c_1 y_0 c_2 y_1 c_3 y_2 c_4 y_3

MLP

H*W

$e_{1,0,0}$ $e_{1,0,1}$ $e_{1,0,2}$
$e_{1,1,0}$ $e_{1,1,1}$ $e_{1,1,2}$
$e_{1,2,0}$ $e_{1,1,1}$ $e_{1,2,2}$

Alignment

softmax

H*W

$a_{1,0,0}$ $a_{1,0,1}$ $a_{1,0,2}$
$a_{1,1,0}$ $a_{1,1,1}$ $a_{1,1,2}$
$a_{1,2,0}$ $a_{1,2,1}$ $a_{1,2,2}$

Attention

[Старт] Кот за цветами

$c_t = \sum_{i,j} a_{t,i,j} z_{t,i,j}$

- контекстный вектор

Блок внимания в общем виде

Векторы входных данных: x ; (N,D)

Векторы-запросы (Queries): q (M,D_k)

Векторы-ключи: k = xW_k

Векторы-значения: v = xW_v

Матрица выравнивания (Alignment): $e_{i,j} = q_j\, k_j / \sqrt{D}$

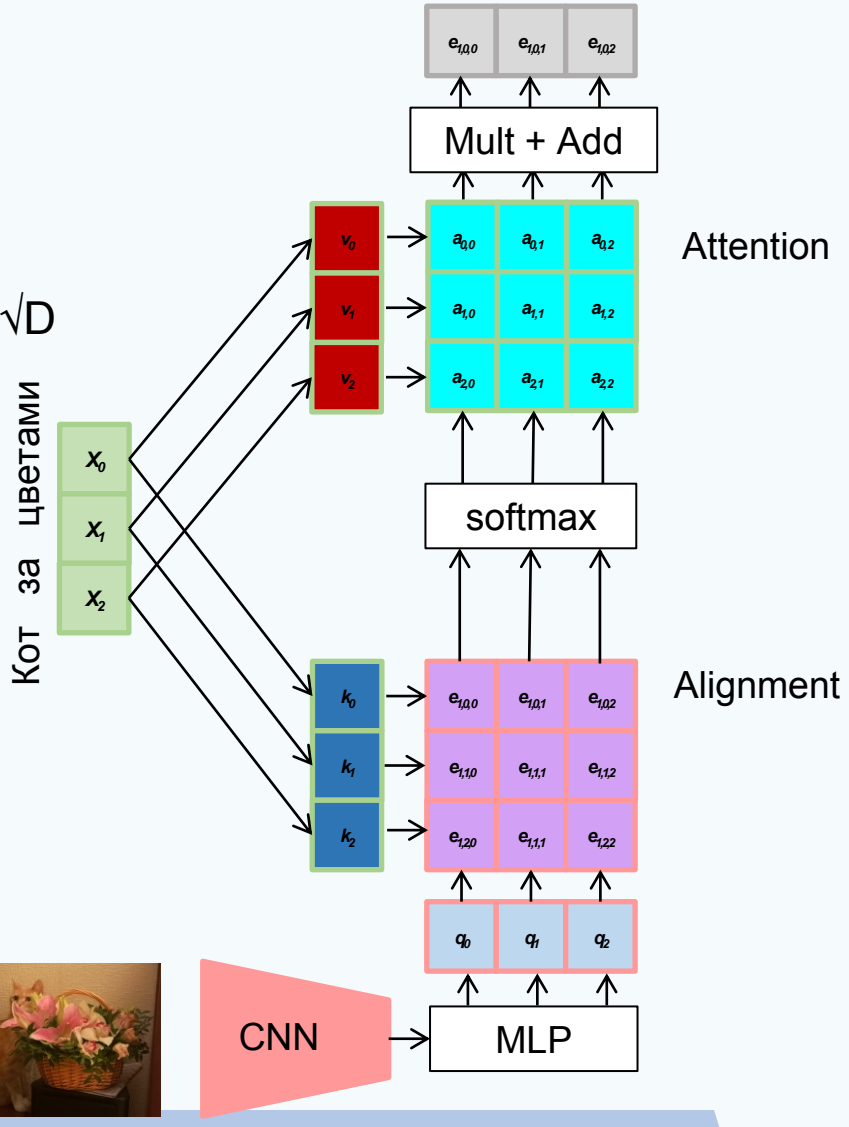
Матрица внимания (Attention): a = softmax(e)

Выходные значения: $y_j = \sum_i a_{ij} v_i$; (D_v)

Массив Queries содержит

пространственные признаки,

полученные из изображения энкодером



Self-attention блок

Векторы входных данных: x ; (N,D)

Векторы-запросы (queries): $q = xW_q$ (M,D_k)

Векторы-ключи: $k = xW_k$

Векторы-значения: $v = xW_v$

Матрица выравнивания (Alignment): $e_{i,j} = q_j\, k_j / \sqrt{D}$

Матрица внимания (Attention): a = softmax(e)

Выходные значения: $y_j = \sum_i a_{ij} v_i$; (D_v)

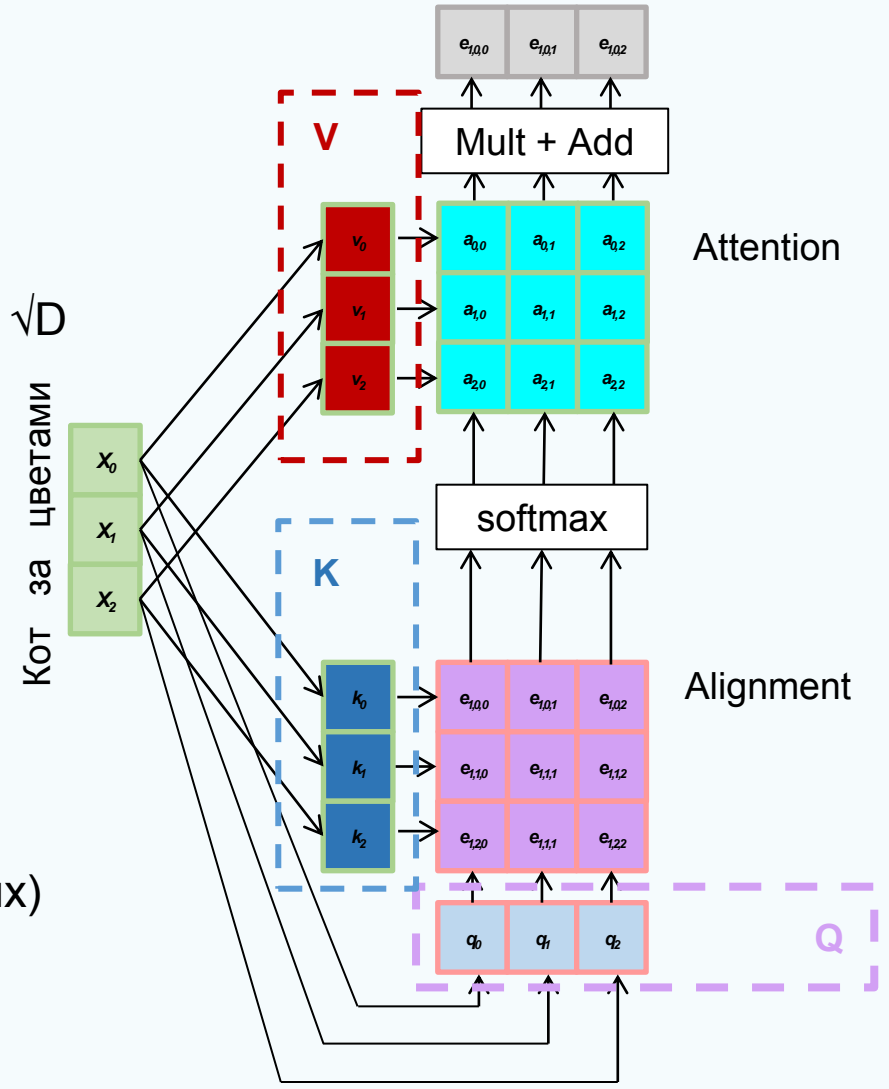
Queries получается напрямую из X

Поскольку все операции матричные

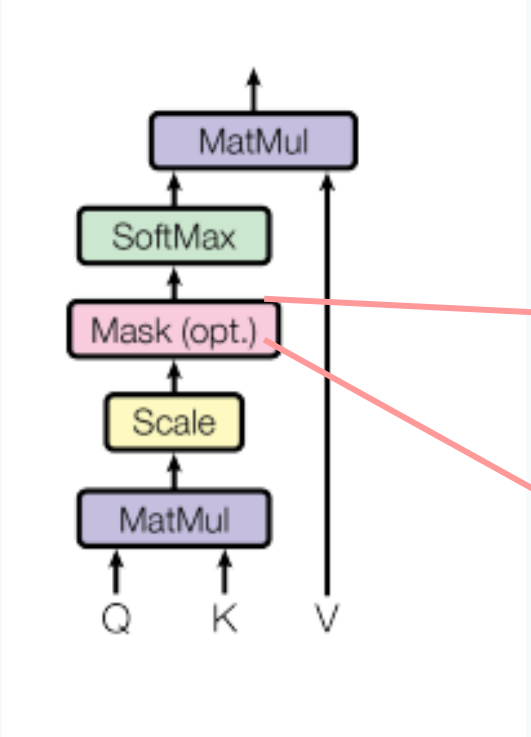
и не зависят от компоненты времени,

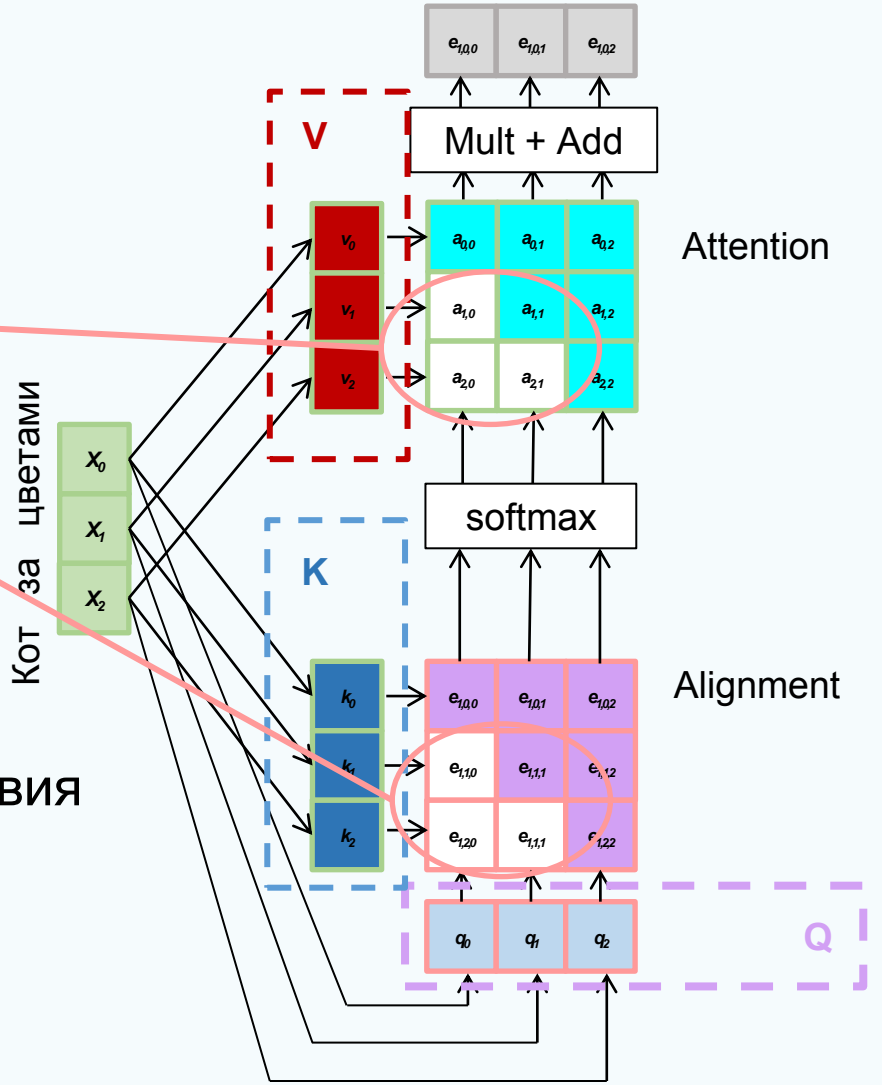
порядок входных (и соответственно, выходных)

элементов можно изменять



Маскированный слой внимания





Предотвращение взаимодействия
входных векторов
с векторами будущего

рисунок: https://arxiv.org/pdf/1706.03762.pdf

Трансформер

Multi-Head Attention – блоки внимания, работающие параллельно

Positional encoding – гармонические функции для кодирования позиции j входного вектора x_j в пространство размерности матриц блока внимания:

$$p(t) = \{[\sin(\omega_1 t), \cos(\omega_1 t)], \ldots [\sin(\omega_d t), \cos(\omega_d t)]\}_d$$

где $\omega_k = 1000^{-2k/d}$

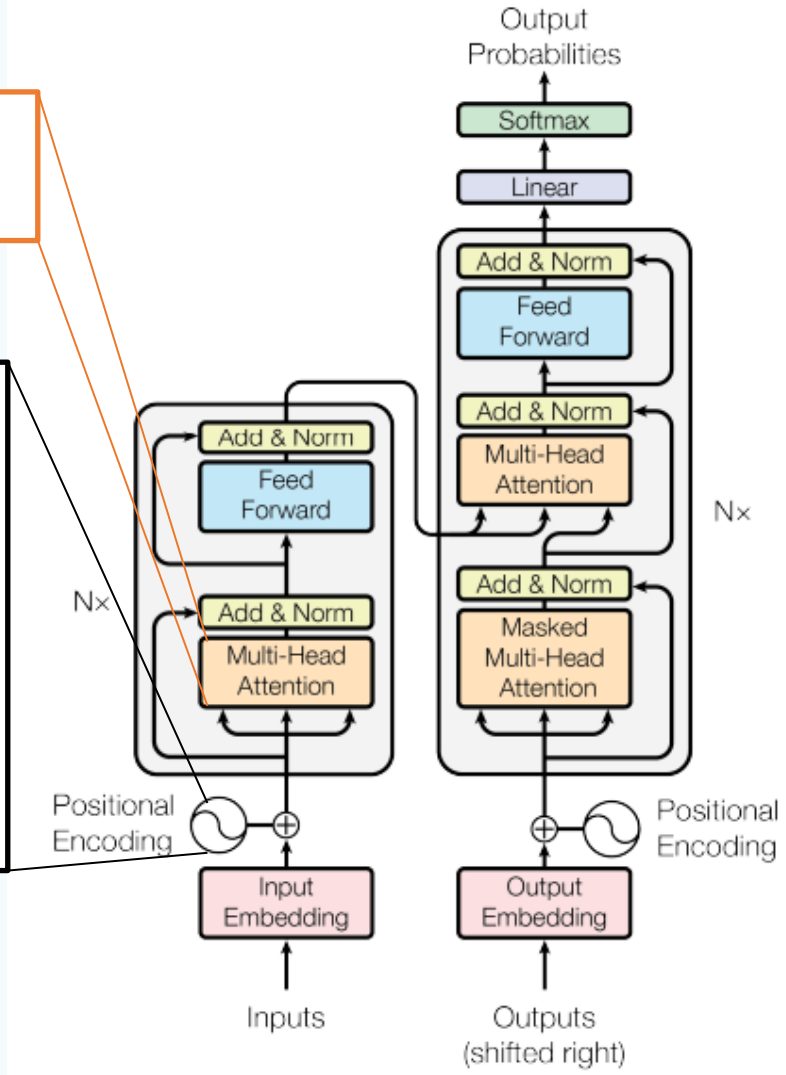
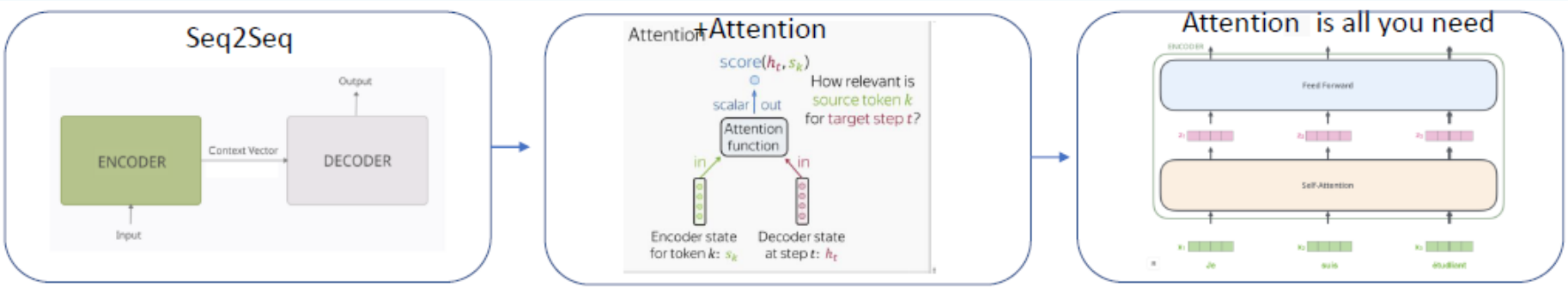


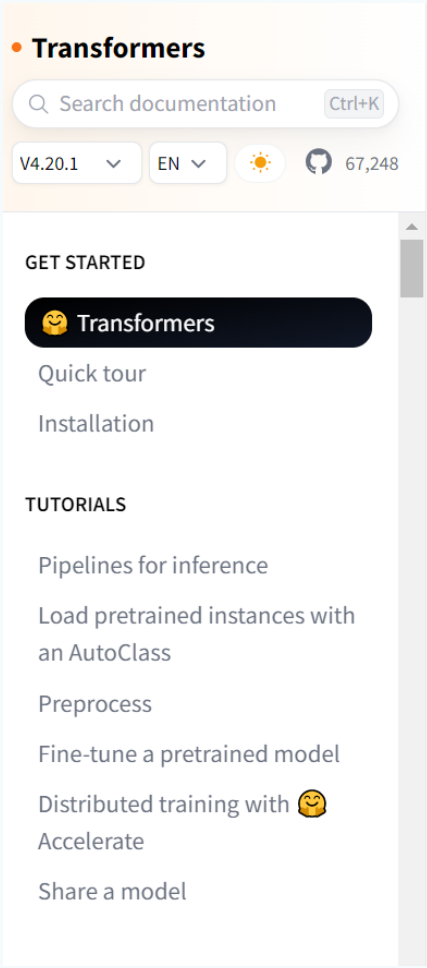
рисунок: https://arxiv.org/pdf/1706.03762.pdf

От рекуррентных сетей к трансформерам



- Seq2Seq: Последовательность векторов от энкодера передается декодеру в виде некоторого вектора;
- Attention – вектор получается суммированием векторов энкодера с подбираемыми весами, отвечающими за контекст;
- Self-attention – отказ от рекуррентности. Учитываем контекст всех слов для каждого слова

Зоопарк трансформеров



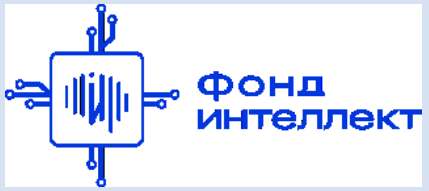
Jax (via Flax), PyTorch, and/or TensorFlow.

Model	Tokenizer slow	Tokenizer fast	PyTorch support	TensorFlow support	Flax Support
ALBERT	✅	✅	✅	✅	✅
BART	✅	✅	✅	✅	✅
BEiT	❌	❌	✅	❌	✅
BERT	✅	✅	✅	✅	✅
Bert Generation	✅	❌	✅	❌	❌
BigBird	✅	✅	✅	❌	✅
BigBird-Pegasus	❌	❌	✅	❌	❌
Blenderbot	✅	✅	✅	✅	✅
BlenderbotSmall	✅	✅	✅	✅	✅
BLOOM	❌	✅	✅	❌	❌
CamemBERT	✅	✅	✅	✅	❌

https://github.com/huggingface/transformers

Спасибо за внимание!

Лаборатория адаптивных методов обработки данных



Учебный курс

«**Машинное обучение в физике**»

Занятие №13 (лекция).

Генеративные состязательные сети. Вариационные автоэнкодеры. Генерация данных

Авторы курса:

С.А. Доленко, И.М. Гаджиев, А.О. Ефиторов, И.В. Исаев, В.Р. Широкий

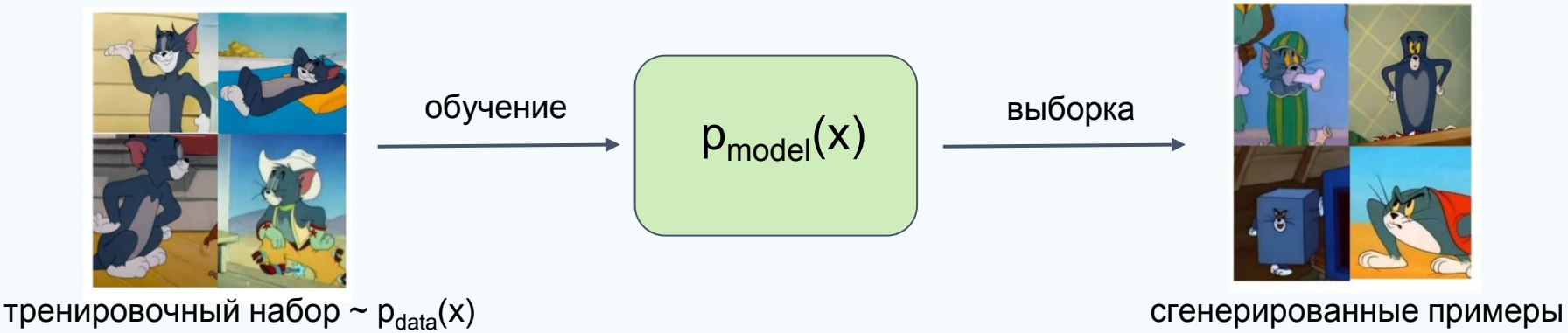
Если данных недостаточно: генерация дополнительных примеров

- Одна из самых серьёзных проблем при увеличении глубины сети – слишком малое число примеров по сравнению с числом весов
- Помимо разных способов регуляризации, можно попробовать увеличить число примеров
- Если нет возможности получить дополнительные примеры из источника, можно попробовать генерировать их
- Это нужно сделать так, чтобы генерированные примеры были похожи на настоящие, но не являлись их прямой комбинацией
- Лучше всего, чтобы генерированные примеры порождались тем же распределением, которое описывает имеющиеся примеры
- Каким образом это можно сделать?

Генеративное моделирование

Цели:

1) Обучить определение параметров модельной функции распределения примеров $p_{model}(x)$, которая аппроксимирует $p_{data}(x)$
2) Произвести выборку новых примеров x из $p_{model}(x)$



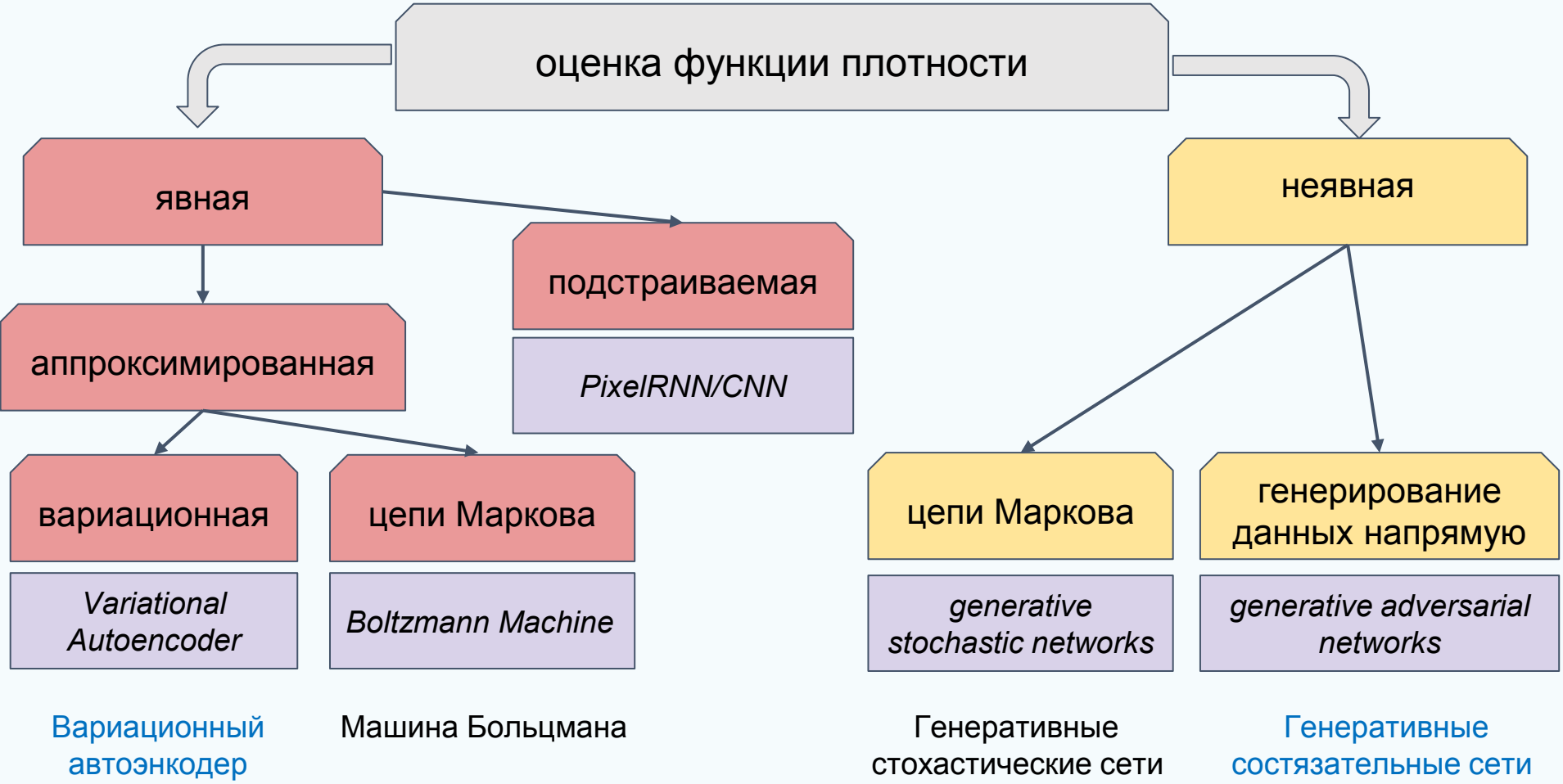
тренировочный набор ~ $p_{data}(x)$

сгенерированные примеры

Задача: оценка плотности вероятности распределения примеров:

- явная оценка функции плотности
- неявная - без определения функции плотности

Таксономия генеративных моделей



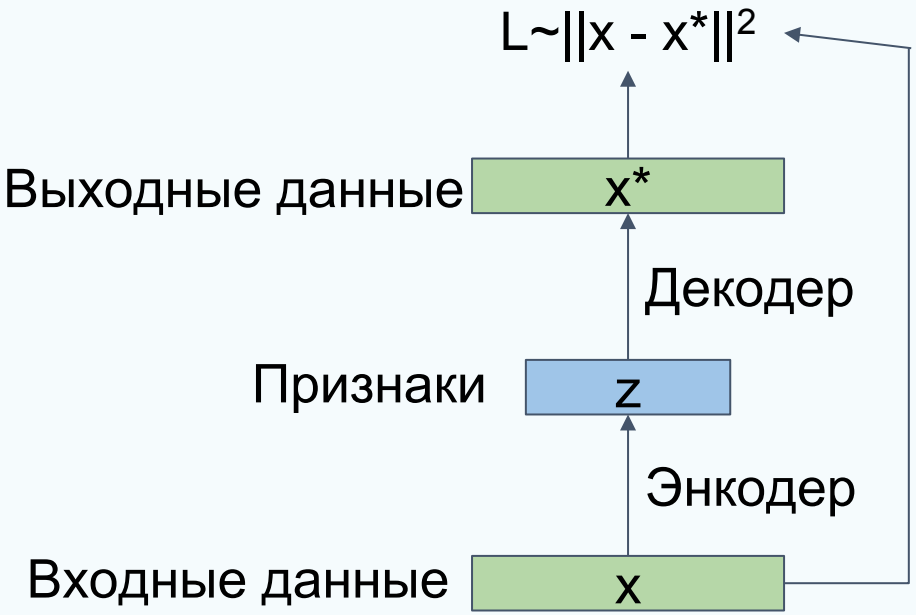
Автоэнкодеры: обучение

Автоэнкодер (автоассоциативная память) –

симметричная сеть с «узким горлом»

Обучение “без учителя”:

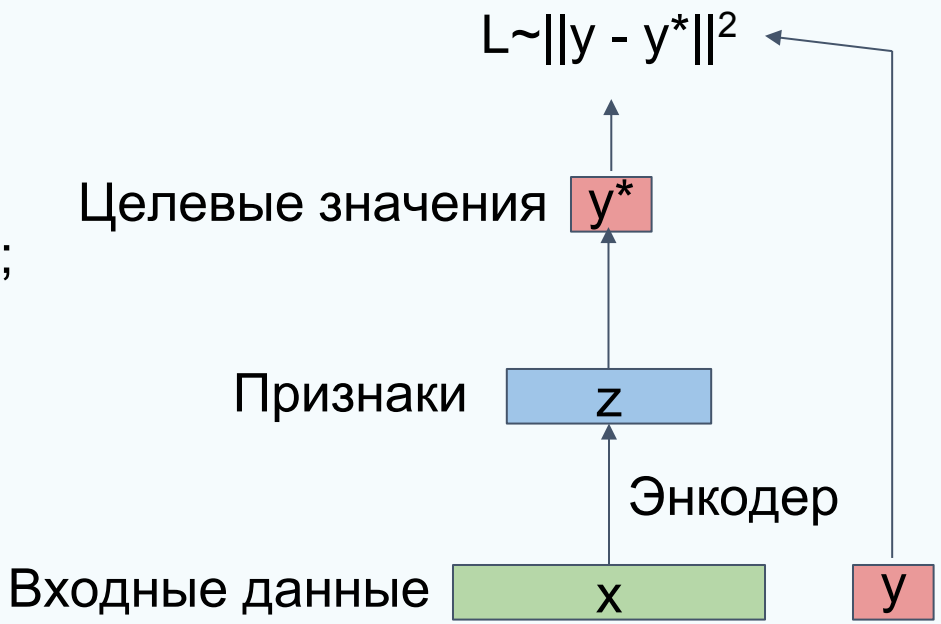
- Модель реконструирует входные примеры
- Метки классов не используются!
- Размерность z < x



Автоэнкодеры: перенос обучения

Перенос обучения (transfer learning):

- Используются веса предобученного энкодера;
- Дополнительный слой для решения целевой задачи;
- Обычно веса энкодера “замораживают”



Вариационные автоэнкодеры

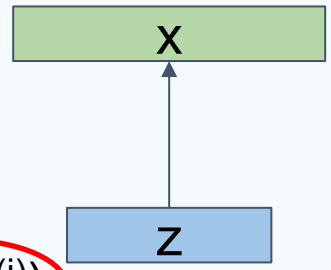
Вариационный автоэнкодер (ВАЭ) = variational autoencoder (VAE)

Пусть у нас имеется обучающий набор данных $\{x^{(i)}\}^N_{i=1,P}$ сгенерированный из распределения латентного пространства z

распределение Гаусса - априорное распределение для $p_\theta(z)$!

Выборка из истинной функции условной вероятности: $p_{\theta^*}(x|z^{(i)})$

Выборка из истинной функции априорной вероятности: $z^{(i)} \sim p_{\theta^*}(z^{(i)})$



Задача: оценить истинные параметры θ* генеративной модели на основе только тренировочных данных x

Вариационные автоэнкодеры

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

Апостериорная функция плотности вероятности:

$$p_\theta(z|x) = p_\theta(x|z) p_\theta(z) / p_\theta(x)$$

Решение: помимо моделирования $p_\theta(x|z)$, обучить $q_\varphi(z|x)$,
которая будет аппроксимировать апостериорную вероятность $p_\theta(z|x)$

Вариационные автоэнкодеры

$\log p_\theta(x^{(i)})$

$$= \mathbf{E}_{z \sim q_\phi(z|x^{(i)})} \left[\log p_\theta(x^{(i)}) \right] \quad (p_\theta(x^{(i)}) \text{ Does not depend on } z)$$

максимизируем функцию правдоподобия сгенерированных примеров

$$= \mathbf{E}_z \left[\log \frac{p_\theta(x^{(i)} \mid z) p_\theta(z)}{p_\theta(z \mid x^{(i)})} \right] \quad \text{(Bayes' Rule)}$$

$$= \mathbf{E}_z \left[\log \frac{p_\theta(x^{(i)} \mid z) p_\theta(z)}{p_\theta(z \mid x^{(i)})} \frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})} \right] \quad \text{(Multiply by constant)}$$

$$= \mathbf{E}_z \left[\log p_\theta(x^{(i)} \mid z) \right] - \mathbf{E}_z \left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)} \right] + \mathbf{E}_z \left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})} \right] \quad \text{(Logarithms)}$$

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

Оценка для сети-декодера может быть произведена напрямую через выборку

Расстояние Кульбака-Лейблера между гауссианами для энкодера и априорным распределением z

Невозможно оценить расстояние Кульбака-Лейблера, но знаем что ≥ 0

Вариационные автоэнкодеры

$\log p_\theta(x^{(i)})$

$$= \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] \quad (p_\theta(x^{(i)}) \text{ Does not depend on } z)$$

максимизируем функцию правдоподобия сгенерированных примеров

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] \quad (\text{Bayes' Rule})$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] \quad (\text{Multiply by constant})$$

$$= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})}\right] \quad (\text{Logarithms})$$

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

$L(x^{(i)}, \theta, \varphi)$ ≥ 0

DKL – расстояние Кульбака-Лейблера

$p_\theta(x^{(i)}|z)$ и KL дифференцируемы => можем вычислять градиент и оптимизировать!

Вариационные автоэнкодеры

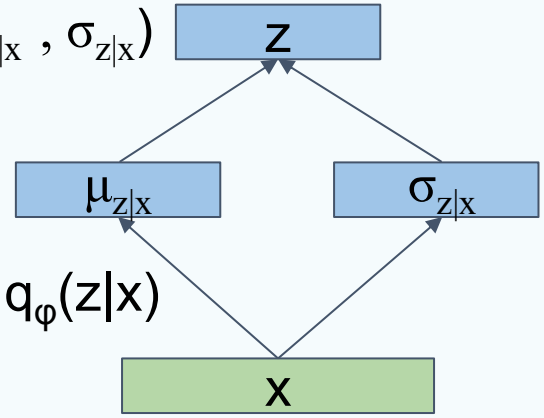
максимизируем нижнюю границу функции правдоподобия!

$$\underbrace{E_z[\log p_\theta(x^{(i)}|z)] - D_{KL}(q_\varphi(z|x^{(i)})||p_\theta(z))}_{L(x^{(i)}, \theta, \varphi)}$$

Репараметризационный трюк для дифференцируемости процедуры выборки:

$$z \sim (\mu_{z|x} + \varepsilon\sigma_{z|x}), \text{ где } \varepsilon \sim N(0,I)$$

Выборка: $z \sim N(\mu_{z|x}, \sigma_{z|x})$



Энкодер: $q_\varphi(z|x)$

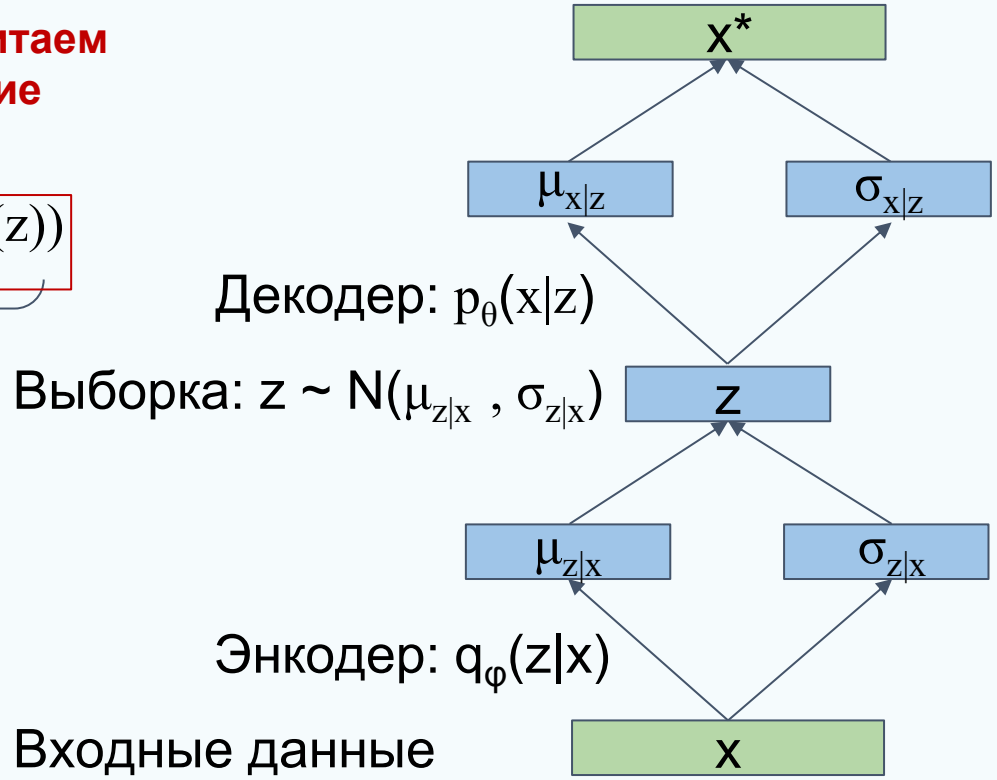
Входные данные

Вариационные автоэнкодеры

для каждого входного примера считаем прямое и обратное распространение

$$E_z[\log p_\theta(x^{(i)}|z)] - D_{KL}(q_\varphi(z|x^{(i)})||p_\theta(z))$$

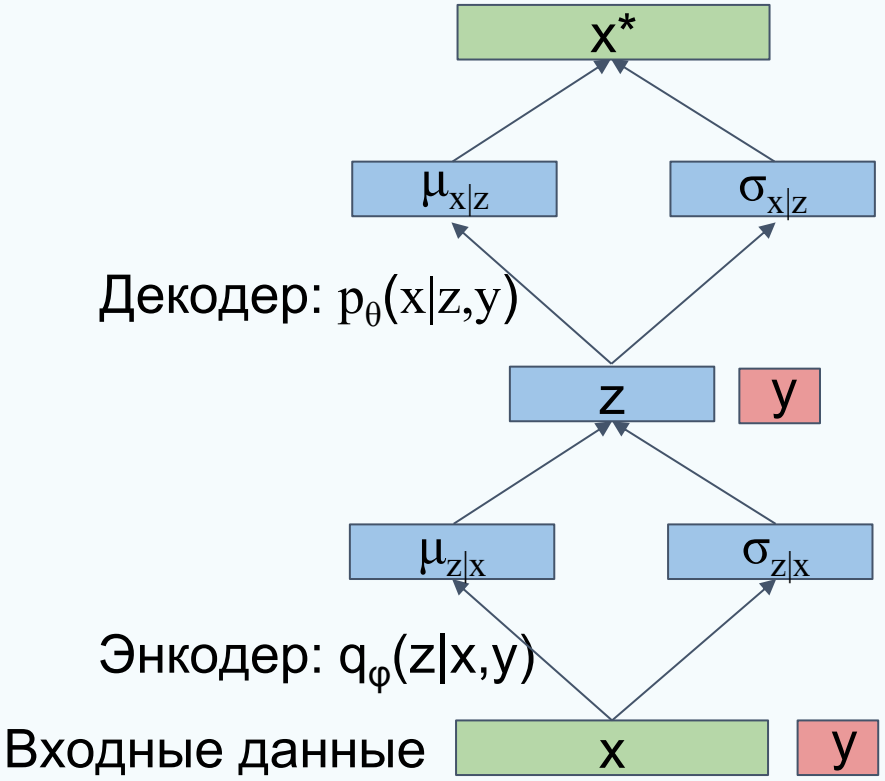
$$L(x^{(i)}, \theta, \varphi)$$



Генерация данных с помощью ВАЭ

- После завершения обучения генерируем примеры, выбирая их из априорного распределения (Гаусса)

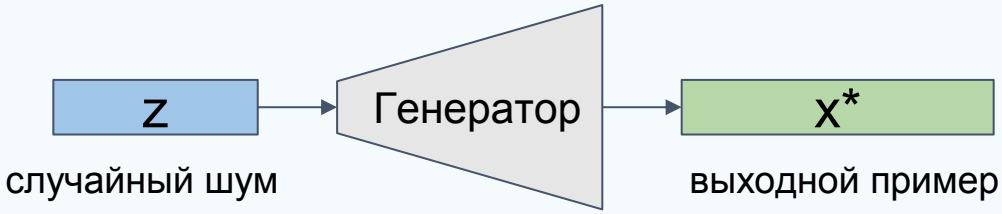
- Если добавить при обучении сети (простой конкатенацией) вектор y ко входным данным и к вектору z, то получим декодер, способный генерировать примеры заданного класса y



Генеративные состязательные сети

Задача: выборка тренировочных примеров
из сложного пространства высокой размерности (изображения)

Решение: выборка векторов из простого распределения (случайный шум)
и их отображение с помощью сети-генератора
в сложное распределение примеров



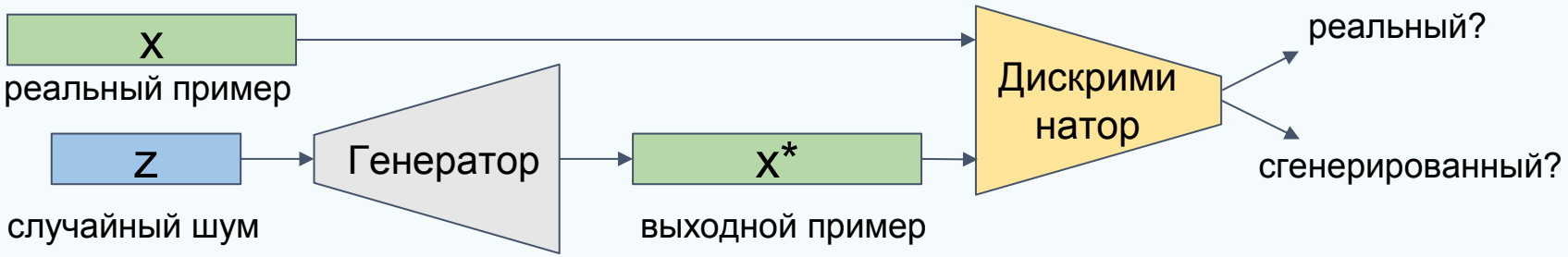
Как учить?

Генеративные состязательные сети

Генеративные состязательные сети (ГСС) = Generative Adversarial Networks (GAN)

В отличие от ВАЭ, будем работать с отдельными примерами

«Качество» отображения оценим с помощью сети-дискриминатора:
если сеть перестала различать
реальные (x) и сгенерированные (x*) примеры –
отображение корректное



Обучение ГСС

Целевая функция совместного обучение в минимакс игре:

$$\min_{\varphi}\max_{\theta} [E_{x\sim p(x)}\log D_{\theta}(x) + E_{z\sim p(z)}\log (1 - D_{\theta}(x)(G_{\varphi}(z)))]$$

Генератор G_{φ} минимизирует целевую функцию так, чтобы обмануть сеть-дискриминатор D(G(z)) = 1

Дискриминатор D_{θ} максимизирует целевую функцию так, чтобы D(x) = 1 (для реальных примеров x), D(G(z)) =0 (для примеров, сгенерированных на основе шумового вектора z)

Обучение ГСС

Градиентный подъем для дискриминатора:

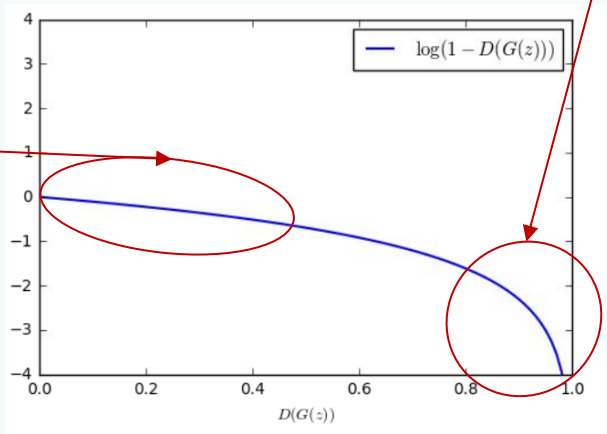
$$\max_\theta [E_{x\sim p(x)} \log D_\theta(x) + E_{z\sim p(z)} \log (1 - D_\theta(x)(G_\varphi(z)))]$$

Градиентный спуск для генератора:

$$\min_\varphi [E_{z\sim p(z)} \log (1 - D_\theta(x)(G_\varphi(z)))]$$

значение градиента максимально для примеров, которые уже неразличимы для дискриминатора

пример, распознанный как поддельный, не обучает сеть генератора (из-за пологости градиента)



Обучение ГСС

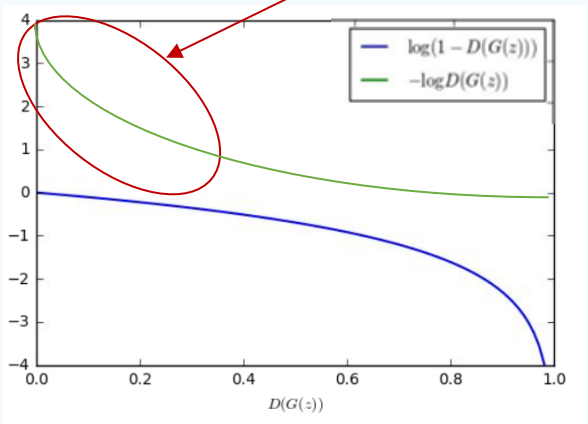
Градиентный подъем для дискриминатора:

$$\max_\theta [E_{x\sim p(x)} \log D_\theta(x) + E_{z\sim p(z)} \log (1 - D_\theta(x)(G_\varphi(z)))]$$

Замена: градиентный подъем для генератора:

$$\max_\varphi [E_{z\sim p(z)} \log (D_\theta(x)(G_\varphi(z)))]$$

Значительная магнитуда градиента для “плохих” примеров



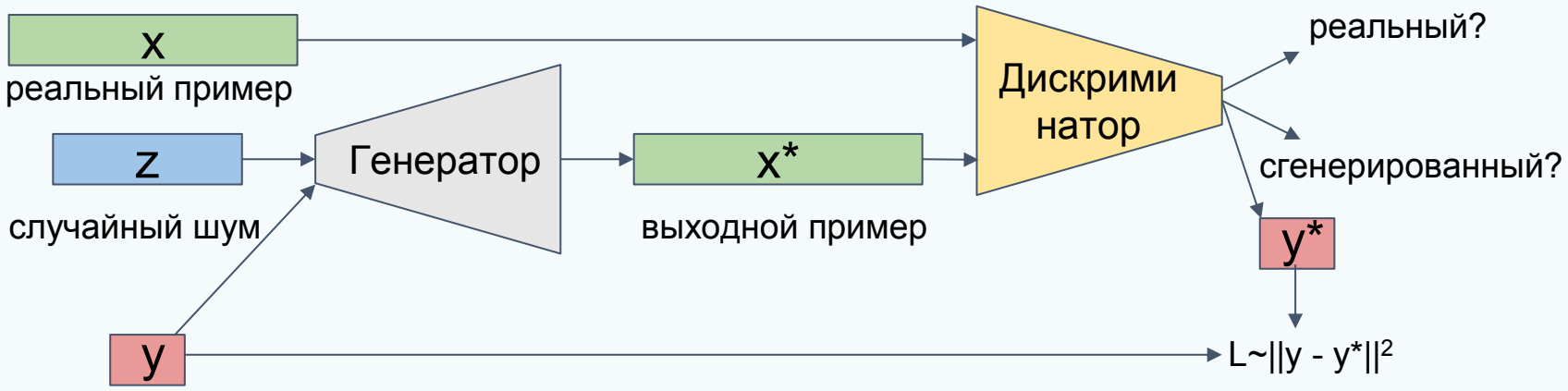
Вместо минимизации функции правдоподобия верного решения от дискриминатора, **максимизируем правдоподобие ошибки** дискриминатора

Обусловленная генерация примеров

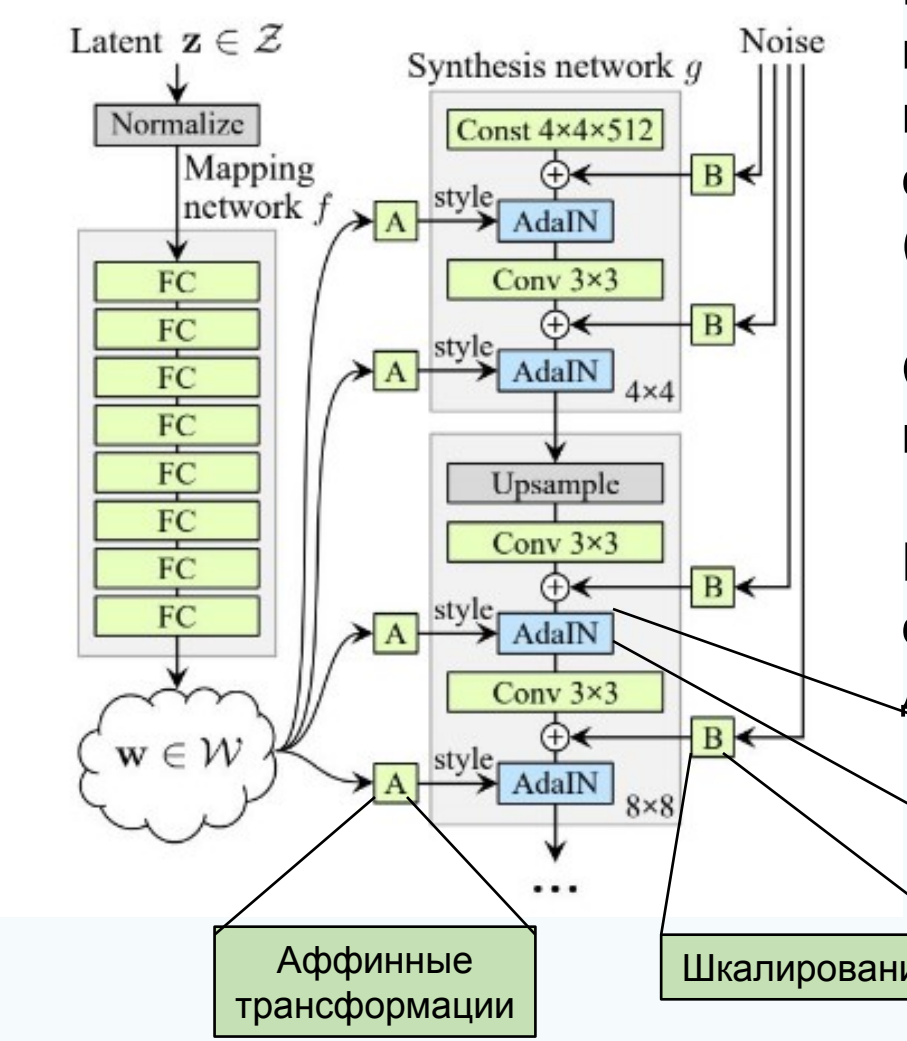
Целевая функция дискриминатора минимальна только для пары:

{X (изображение) - реально, Y (метка) - верная}

- реальное изображение [X] - неверная метка;
- поддельное изображение [X*], верная метка;
- поддельное изображение [X*], неверная метка.



Обусловленная генерация: Stylegan



Изображение кодируется в латентное пространство z меньшей размерности сетью-энкодером (не представлена)

Сеть отображения (mapping) извлекает «стиль» изображения y

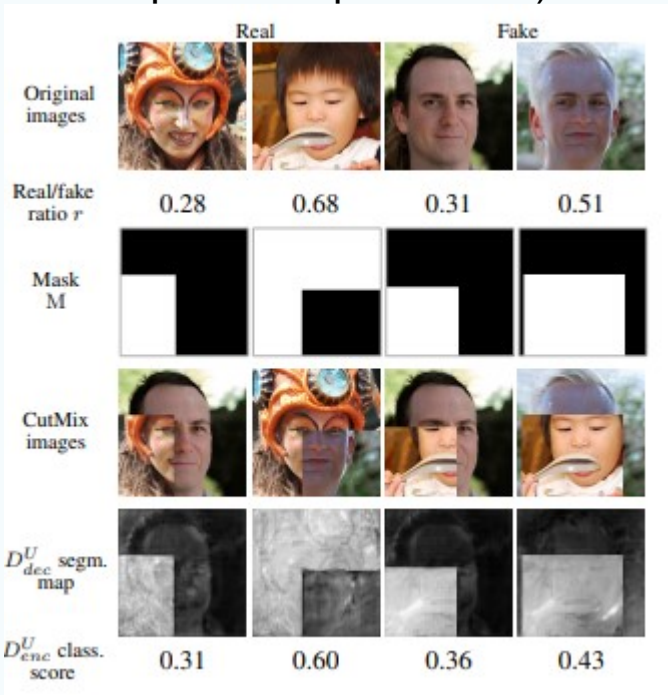
Шум используется для стохастического добавления деталей изображению

$$AdaIN(x_i, y) = y_{s,i}\frac{x_i - \mu(x_i)}{\sigma(x_i)} + y_{b,i}$$

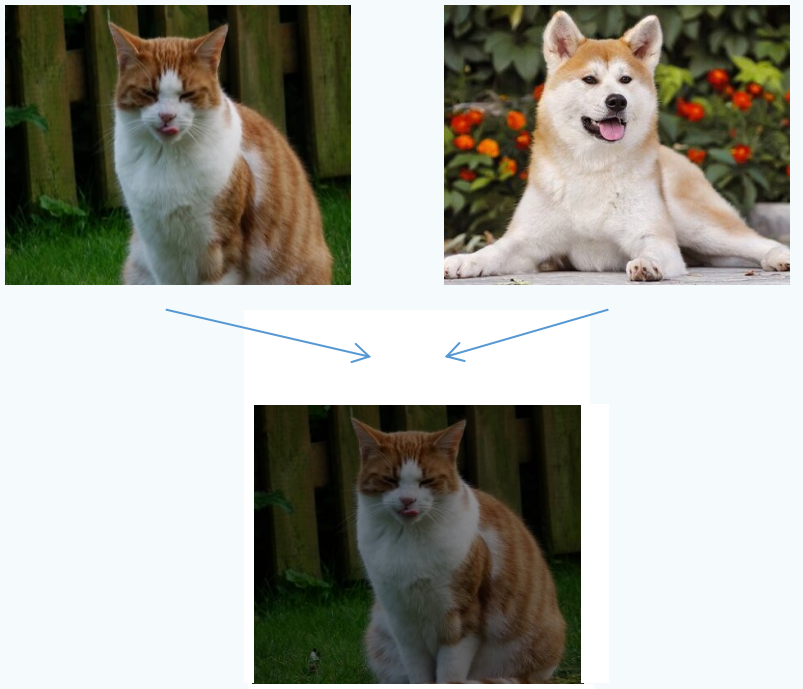
рисунок: https://arxiv.org/pdf/1812.04948.pdf

Аугментация для задачи генерации

Cutmix
(замена части сгенерированного изображения реальным)



Mixup
(смешение сгенерированного и реального изображений)



Задача дискриминатора теперь не бинарная классификация, а определение соотношения реального и сгенерированного изображений

рисунок: https://arxiv.org/pdf/2002.12655.pdf

Зоопарк GAN моделей

- GAN - Generative Adversarial Networks
- 3D-GAN - Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling
- acGAN - Face Aging With Conditional Generative Adversarial Networks
- AC-GAN - Conditional Image Synthesis With Auxiliary Classifier GANs
- AdaGAN - AdaGAN: Boosting Generative Models
- AEGAN - Learning Inverse Mapping by Autoencoder based Generative Adversarial Nets
- AffGAN - Amortised MAP Inference for Image Super-resolution
- AL-CGAN - Learning to Generate Images of Outdoor Scenes from Attributes and Semantic Layouts
- ALI - Adversarially Learned Inference
- AM-GAN - Generative Adversarial Nets with Labeled Data by Activation Maximization
- AnoGAN - Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery
- ArtGAN - ArtGAN: Artwork Synthesis with Conditional Categorial GANs
- b-GAN - b-GAN: Unified Framework of Generative Adversarial Networks
- Bayesian GAN - Deep and Hierarchical Implicit Models
- BEGAN - BEGAN: Boundary Equilibrium Generative Adversarial Networks
- BiGAN - Adversarial Feature Learning
- BS-GAN - Boundary-Seeking Generative Adversarial Networks
- CGAN - Conditional Generative Adversarial Nets
- CaloGAN - CaloGAN: Simulating 3D High Energy Particle Showers in Multi-Layer Electromagnetic Calorimeters with Generative Adversarial Networks
- CCGAN - Semi-Supervised Learning with Context-Conditional Generative Adversarial Networks
- CatGAN - Unsupervised and Semi-supervised Learning with Categorical Generative Adversarial Networks
- CoGAN - Coupled Generative Adversarial Networks
- Context-RNN-GAN - Contextual RNN-GANs for Abstract Reasoning Diagram Generation
- C-RNN-GAN - C-RNN-GAN: Continuous recurrent neural networks with adversarial training
- CS-GAN - Improving Neural Machine Translation with Conditional Sequence Generative Adversarial Nets
- CVAE-GAN - CVAE-GAN: Fine-Grained Image Generation through Asymmetric Training
- CycleGAN - Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks
- DTN - Unsupervised Cross-Domain Image Generation
- DCGAN - Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks
- DiscoGAN - Learning to Discover Cross-Domain Relations with Generative Adversarial Networks
- DR-GAN - Disentangled Representation Learning GAN for Pose-Invariant Face Recognition
- DualGAN - DualGAN: Unsupervised Dual Learning for Image-to-Image Translation
- EBGAN - Energy-based Generative Adversarial Network
- f-GAN - f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization
- FF-GAN - Towards Large-Pose Face Frontalization in the Wild
- GAWWN - Learning What and Where to Draw
- GeneGAN - GeneGAN: Learning Object Transfiguration and Attribute Subspace from Unpaired Data
- Geometric GAN - Geometric GAN
- GoGAN - Gang of GANs: Generative Adversarial Networks with Maximum Margin Ranking
- GP-GAN - GP-GAN: Towards Realistic High-Resolution Image Blending
- IAN - Neural Photo Editing with Introspective Adversarial Networks
- iGAN - Generative Visual Manipulation on the Natural Image Manifold
- IcGAN - Invertible Conditional GANs for image editing
- ID-CGAN - Image De-raining Using a Conditional Generative Adversarial Network
- Improved GAN - Improved Techniques for Training GANs
- InfoGAN - InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets
- LAGAN - Learning Particle Physics by Example: Location-Aware Generative Adversarial Networks for Physics Synthesis
- LAPGAN - Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks

https://github.com/hindupuravinash/the-gan-zoo

Спасибо за внимание!

Лаборатория адаптивных методов обработки данных

Учебный курс

«Машинное обучение в физике»

Занятие №15 (лекция).

Анализ временных рядов. Комбинированные алгоритмы.

Авторы курса:

С.А. Доленко, И.М. Гаджиев, А.О. Ефиторов, И.В. Исаев, В.Р. Широкий

Содержание лекции

- ❑ Временные ряды
  - Определение
  - Алгоритмы анализа
  - Прогнозирование временных рядов
- ❑ Комбинированные алгоритмы
  - Ансамбли моделей
    - ✔ Стекинг
    - ✔ Блендинг
    - ✔ Беггинг
    - ✔ Бустинг
  - Решение задач

Временные ряды

Временной ряд (ВР):

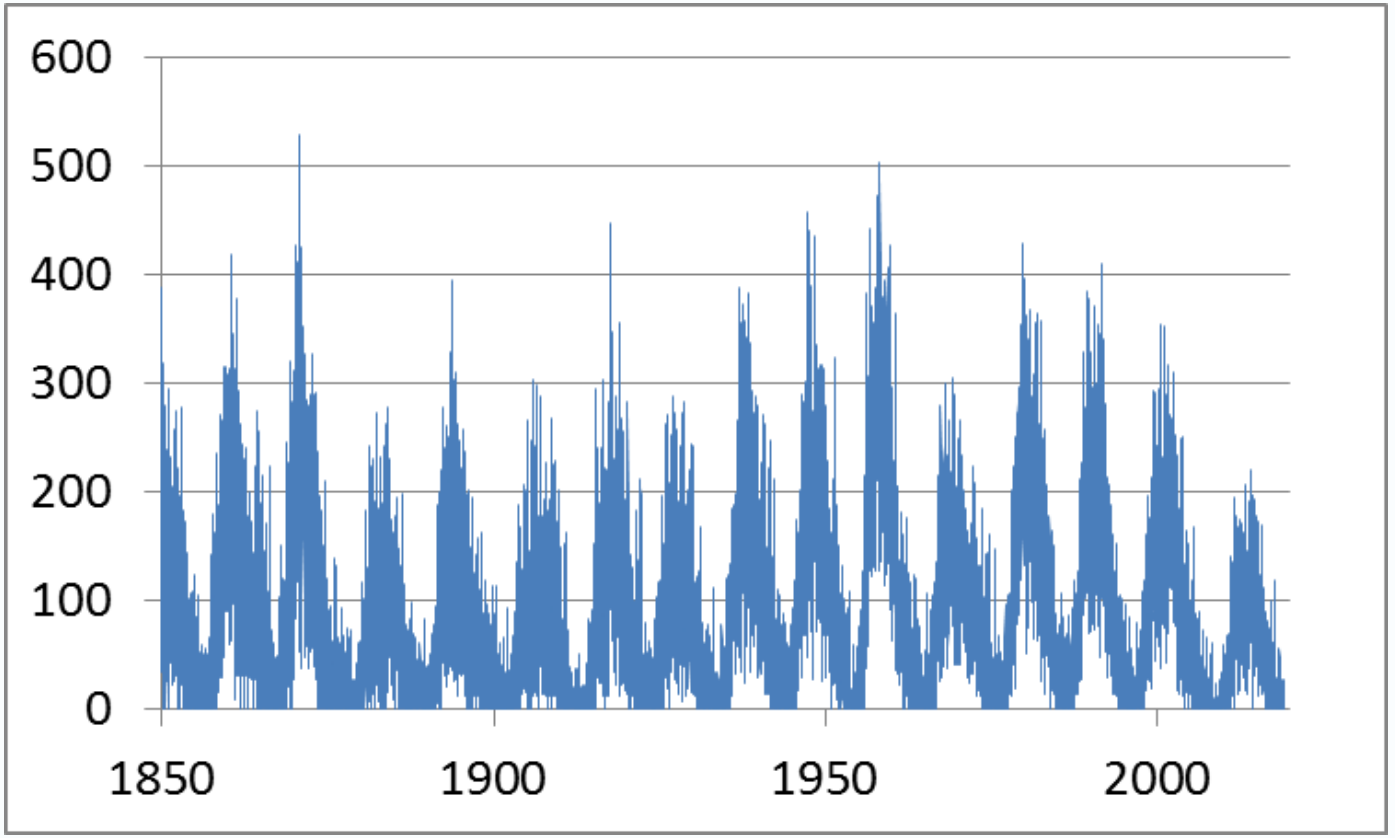
- Характеристика, величина которой изменяется во времени
- Отсчёты эквидистантны по времени
- Характеризует динамическую систему, которая:
  - Описывается набором признаков
  - Текущее состояние зависит от предыдущих состояний и от внешних воздействий
- Совокупность признаков динамической системы является многомерным временным рядом (МВР)

Примеры временных рядов

- Телеметрия автомобиля/самолета/космического аппарата
- Температура воздуха, измеряемая на метеостанции каждый час
- Курсы валют/акций/драгметаллов на бирже
- Число солнечных пятен, измеряемое каждый день (число Вольфа) - используется в дальнейших примерах
- Суточные значения потребления электроэнергии населенным пунктом
- Месячные значения расхода воды в квартире
- Годовые значения объёма стока рек

Примеры временных рядов
Число Вольфа

Число солнечных пятен



Год

Алгоритмы анализа временных рядов
R/S анализ

Цель анализа - определение случайности ВР:

Алгоритм разработан Гарольдом Хёрстом (1880 - 1978)

- Выделяем в ВР из значений θ

  все подряд идущие окна I_a длиной n со значениями $\theta_{n,a}$

$$e_a=(1/n)\sum_{k=1}^{n}\theta_{k,a}$$

$$X_{k,a}=\sum_{k=1}^{n}(\theta_{k,a}-e_a)$$

$$R_{I_a}=\max(X_{k,a})-\min(X_{k,a})$$

$$S_{Ia}=((1/n)\sum(\theta_{k,a}-e_a)^2)^{0.5}$$

$$(R/S)_n=(1/A)\sum_{a=1}^{A}R_{I_a}/S_{I_a}$$

$$R/S=(a*n)^H$$

$$\log(R/S)=H*\log(n)+const$$

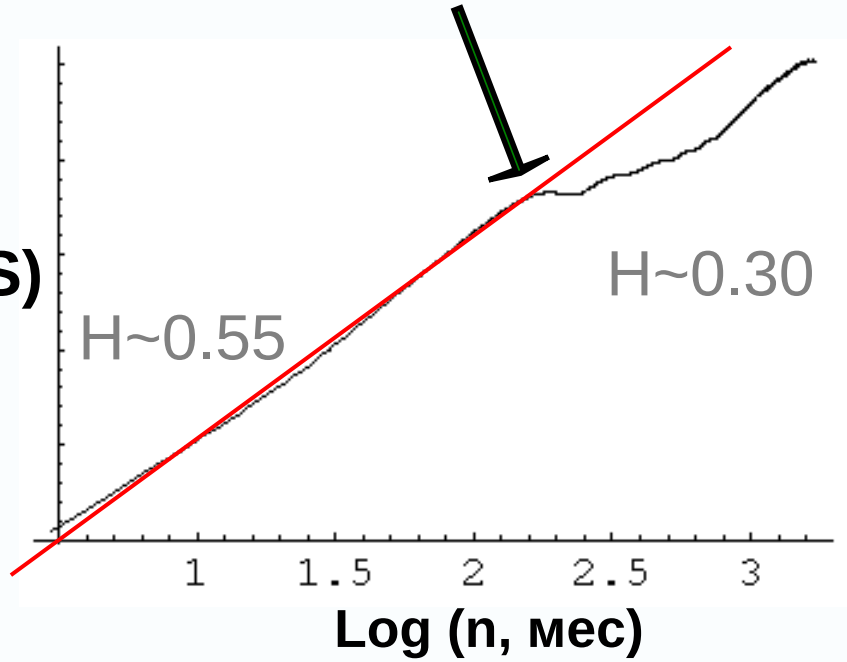
H – *экспонента Хёрста*

Алгоритмы анализа временных рядов
Экспонента Хёрста

Результат R/S анализа –

зависимость экспоненты Хёрста от временного окна

- H=0.5 – случайный процесс
- 0≤H<0.5 – антиперсистентный (эргодический) процесс
- 0.5<H<1 – персистентный (трендовый) процесс



Мера корреляции $C = 2^{(2H-1)} - 1$

Фрактальная размерность $D = 2 - H$

Алгоритмы анализа временных рядов
Автокорреляционная функция (АКФ)

Анализ предполагает нормальное распределение

$\Theta = \theta_1, \theta_2, \theta_3, \ldots, \theta_{(M-2)}, \theta_{(M-1)}, \theta_M$

X_0

$\theta_1, \theta_2, \theta_3, \ldots, \theta_{(M-2)}, \theta_{(M-1)}, \theta_M$

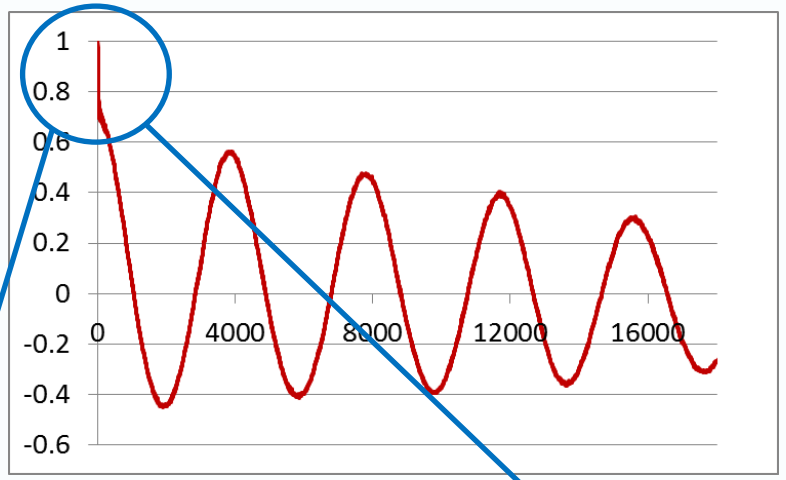
X_1

X_0

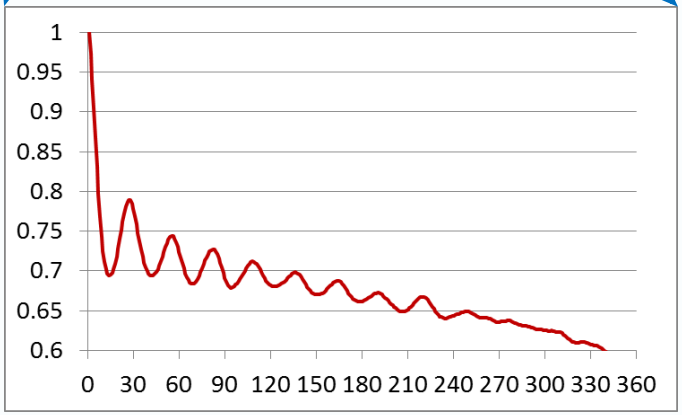
$\theta_1, \theta_2, \theta_3, \ldots, \theta_{(M-2)}, \theta_{(M-1)}, \theta_M$

X_2

$r_1 = corr(X_0, X_1)$ $r_2 = corr(X_0, X_2)$



АКФ по числам Вольфа за 40 лет…



…и

за год

Прогнозирование временных рядов
Общие соображения

- Самая распространённая задача - прогнозирование ВР
- Ключевое свойство системы, позволяющее осуществлять прогнозирование – стационарность (хотя бы квази…)
- Два основных подхода:
  - Погружение (топологическое вложение) ВР
  - Использование рекуррентных сетей
- Необходимые условия:
  - Непрерывность ВР (отсутствие пропусков)
  - Равные временные промежутки измерения

Прогнозирование временных рядов
Погружение временного ряда

Исходный временной ряд

- A, B, C … Y - признаки многомерного ВР
- t - шкала времени

A	B	C		Y	<- признаки
A_{t-n}	B_{t-n}	C_{t-n}		Y_{t-n}	время ->
…	…	…		…	
A_{t-3}	B_{t-3}	C_{t-3}		Y_{t-3}	
A_{t-2}	B_{t-2}	C_{t-2}		Y_{t-2}	
A_{t-1}	B_{t-1}	C_{t-1}		Y_{t-1}	
A_t	B_t	C_t		Y_t	настоящий момент
				Y_{t+1}	
				Y_{t+2}	
				…	

Прогнозирование временных рядов
Погружение временного ряда

Погруженный временной ряд

- Погружение (топологическое вложение, delay embedding) – интерпретация одномерного временного ряда как траектории в d-мерном *лаговом пространстве* (пространстве задержек) $X_{t-d}=\{x_t, x_{t-1}, \ldots x_{t-d}\}$
- Вопрос о глубине погружения d каждой компоненты временного ряда, как правило, требует отдельного исследования

Прогнозирование временных рядов
Погружение временного ряда

Погруженный временной ряд

- Каждая строка – один пример для выбранного алгоритма машинного обучения
- Размерность итогового пространства в общем случае - произведение числа признаков на глубину погружения

A				B		C			Y					<- признаки
A_{t-n}				B_{t-n}		C_{t-n}			Y_{t-n}				...	время ->
...				...		...			...				Y_{t-3}	
A_{t-3}	...			B_{t-3}	...	C_{t-3}	...		Y_{t-3}				Y_{t-2}	
A_{t-2}	A_{t-3}	...		B_{t-2}	B_{t-3}	C_{t-2}	C_{t-3}	...	Y_{t-2}	Y_{t-3}			Y_{t-1}	
A_{t-1}	A_{t-2}	A_{t-3}	...	B_{t-1}	B_{t-2}	C_{t-1}	C_{t-2}	C_{t-3}	Y_{t-1}	Y_{t-2}	Y_{t-3}		Y_t	
A_t	A_{t-1}	A_{t-2}	A_{t-3}	B_t	B_{t-1}	C_t	C_{t-1}	C_{t-2}	Y_t	Y_{t-1}	Y_{t-2}	Y_{t-3}	Y_{t+1}	настоящий момент
									Y_{t+1}				Y_{t+2}	
									Y_{t+2}					
									...					

Комбинированные алгоритмы

Комбинированные алгоритмы -
это совокупность
множества моделей машинного обучения,
решающих одну задачу

Варианты реализации:

- Объединение слабых моделей
- Последовательное исправление ошибки несколькими моделями
- Упрощение задачи одним алгоритмом и решение другим

Комбинированные алгоритмы
Объединение слабых моделей

Стекинг (Stacking)

- Самый простой вид комбинирования
- Строится несколько разных моделей
- Простой вариант комбинированной модели – усреднение ответов “слабых” моделей
- Более эффективный вариант комбинирования – взвешивание вместо усреднения
- Самый мощный вариант стекинга – определение весов алгоритмом верхнего уровня (нейронной сетью) (stacked generalization)

Комбинированные алгоритмы
Объединение слабых моделей

Блендинг (Blending)

- Более сложный вид комбинированных алгоритмов
- Строится несколько разных моделей
  на одной части выборки примеров
- На ответах моделей
  и на другой части выборки примеров
  обучается новая модель

Комбинированные алгоритмы
Объединение слабых моделей

Беггинг (Bagging)

- Множество простых моделей
- Каждая модель обучается на части примеров и/или части признаков исходного набора данных
- Результаты моделей усредняются (регрессия) или определяются голосованием (классификация)
- В случае разбиения по примерам очень важно распределение примеров (стратификация)

Комбинированные алгоритмы
Объединение слабых моделей

Беггинг (Bagging)

- Классический пример подхода – алгоритм случайного леса (Random Forest)
- Строится множество “слабых” классификаторов в виде неглубоких деревьев решений, обученных на части признаков или примеров.
- Ответом модели будет усредненный ответ “слабых” моделей.

Комбинированные алгоритмы
Последовательное исправление ошибок

Бустинг (Boosting)

- Множество простых моделей
- Первая модель обучается на всем наборе данных
- Вторая и последующие модули обучаются
  на результате предыдущей модели
  и на части исходных данных,
  где предыдущая модель ошибается сильнее всего

Комбинированные алгоритмы
Последовательное исправление ошибок

Бустинг (Boosting)

- Классический пример подхода –
  градиентный бустинг на деревьях решений
  (Gradient Boosted Decision Trees)
- Следующее дерево обучается
  на градиенте функционала ошибки предыдущих
- Может обеспечивать высокую точность
- Главный недостаток –
  высокая вычислительная сложность

Комбинированные алгоритмы
Упрощение задачи

Единого рецепта для данного подхода нет,
всё зависит от задачи и примеров данных.

Некоторые примеры комбинирования:

- Данные обладают высокой размерностью
- Можно применить вначале АГК или автоэнкодер,
  чтобы понизить размерность данных,
  а далее уже решить искомую задачу
  в новом пространстве

Комбинированные алгоритмы. Упрощение задачи

Единого рецепта для данного подхода нет,
все зависит от задачи и примеров данных.

Некоторые примеры комбинирования:

- В данных прослеживается различное поведение системы в зависимости от исходных параметров или времени
- Можно проделать кластеризацию данных и решать исходную задачу для каждого из кластеров отдельными моделями

Комбинированные алгоритмы. Упрощение задачи

Единого рецепта для данного подхода нет,
все зависит от задачи и примеров данных.

Некоторые примеры комбинирования:

- Построение иерархической системы при решении задачи многоклассовой классификации
- Алгоритм классификации верхнего уровня разбивает большое количество классов на группы
- Алгоритмы следующего уровня решают задачу классификации внутри каждой группы

Комбинированные алгоритмы. Упрощение задачи

Единого рецепта для данного подхода нет, все зависит от задачи и примеров данных.

Главная цель подхода - решить задачу более точно и более простыми алгоритмами.

Спасибо за внимание!