Анализ данных

smvfe

Данный конспект не является официальным учебным материалом и не претендует на полноту и истинность изложенного. Использование данного материала подразумевает понимание контекста и использование критического мышления.

Содержание

1 Табличный набор данных

Табличный набор данных (tabular dataset) — это структурированное представление данных в виде двумерной таблицы, где:

- **Строки** (rows) — представляют отдельные наблюдения, объекты или примеры (samples, instances, observations).
- **Столбцы** (columns) — представляют признаки, атрибуты или переменные (features, attributes, variables).

Формально табличный набор данных можно представить как матрицу $X \in \mathbb{R}^{n \times m}$, где:

- n — количество объектов (строк)
- m — количество признаков (столбцов)
- x_{ij} — значение j-го признака для i-го объекта

1.1 Типы признаков

1. **Числовые (количественные)**:
   - *Непрерывные* — могут принимать любые значения из интервала (рост, вес, температура)
   - *Дискретные* — принимают счётные значения (количество комнат, возраст в годах)
2. **Категориальные (качественные)**:
   - *Номинальные* — нет естественного порядка (цвет, страна, пол)
   - *Порядковые (ординальные)* — есть естественный порядок (образование, рейтинг)
3. **Бинарные** — принимают только два значения (да/нет, 0/1)

1.2 Целевая переменная

В задачах обучения с учителем выделяют **целевую переменную** y (target variable, label):

- В задаче **классификации**: $y \in \{1, 2, \ldots, K\}$ — номер класса
- В задаче **регрессии**: $y \in \mathbb{R}$ — вещественное число

2 Форматы CSV, TSV, ARFF

2.1 CSV (Comma-Separated Values)

CSV — текстовый формат для хранения табличных данных, где значения разделены запятыми.
Особенности:

- Первая строка обычно содержит названия столбцов (заголовок)
- Каждая последующая строка — одна запись
- Значения разделены запятыми (или точкой с запятой в некоторых локалях)
- Текстовые значения могут быть заключены в кавычки

- Пустые значения обозначаются пустой строкой между разделителями

Пример:

```
name,age,salary
Alice,30,50000
Bob,25,45000
Charlie,35,60000
```

2.2 TSV (Tab-Separated Values)

TSV — аналог CSV, где в качестве разделителя используется символ табуляции (`\t`).

Преимущества перед CSV:

- Табуляция реже встречается в данных, чем запятая
- Меньше проблем с экранированием
- Проще визуально выравнивать столбцы

2.3 ARFF (Attribute-Relation File Format)

ARFF — формат, разработанный для системы машинного обучения WEKA. Содержит метаданные о типах атрибутов.

Структура файла:

1. `@RELATION` — имя набора данных
2. `@ATTRIBUTE` — описание каждого атрибута с указанием типа
3. `@DATA` — секция с данными

Типы атрибутов в ARFF:

- `NUMERIC` или `REAL` — числовой
- `INTEGER` — целочисленный
- `STRING` — строковый
- `{value1, value2, ...}` — номинальный (перечисление)
- `DATE` — дата

Пример:

```
@RELATION employees

@ATTRIBUTE name STRING
@ATTRIBUTE age NUMERIC
@ATTRIBUTE department {IT, HR, Sales}
@ATTRIBUTE salary NUMERIC

@DATA
'Alice',30,IT,50000
'Bob',25,HR,45000
'Charlie',35,Sales,60000
```

3 Категории и числа. One-hot кодирование

Большинство алгоритмов машинного обучения работают только с числовыми данными, поэтому категориальные признаки необходимо преобразовывать.

3.1 Методы кодирования категориальных признаков

3.1.1 Label Encoding (метки)

Каждой категории присваивается уникальное целое число:

$$\text{Red} \to 0, \quad \text{Green} \to 1, \quad \text{Blue} \to 2$$

Недостаток: вводит ложный порядок между категориями (алгоритм может решить, что Blue $>$ Green $>$ Red).

3.1.2 One-Hot Encoding (унитарное кодирование)

Категориальный признак с K уникальными значениями преобразуется в K бинарных признаков.
Для категории c_k создаётся вектор длины K:

$$\mathbf{e}_k = (0, \ldots, 0, \underbrace{1}_{k\text{-я позиция}}, 0, \ldots, 0)$$

Пример: признак «Цвет» с значениями {Red, Green, Blue}:

Цвет	is_Red	is_Green	is_Blue
Red	1	0	0
Green	0	1	0
Blue	0	0	1

Преимущества:

- Не вводит ложный порядок
- Подходит для большинства алгоритмов

Недостатки:

- Увеличивает размерность данных (проблема для признаков с большим числом категорий)
- Создаёт разреженные (sparse) матрицы
- **Dummy variable trap** — линейная зависимость между признаками (решается удалением одного столбца)

3.1.3 Другие методы

- **Target Encoding** — замена категории на среднее значение целевой переменной
- **Frequency Encoding** — замена на частоту встречаемости
- **Binary Encoding** — представление номера категории в двоичном виде

4 Мин-макс нормализация. Стандартизация

Масштабирование признаков (feature scaling) — преобразование числовых признаков к единому масштабу. Необходимо для:

- Алгоритмов, чувствительных к масштабу (kNN, SVM, нейронные сети, градиентный спуск)
- Ускорения сходимости оптимизации
- Корректного сравнения важности признаков

4.1 Мин-макс нормализация (Min-Max Scaling)

Линейное преобразование значений признака в диапазон $[0, 1]$:

$$x'_i = \frac{x_i - x_{\min}}{x_{\max} - x_{\min}}$$

где $x_{\min}$ и $x_{\max}$ — минимальное и максимальное значения признака.
Для произвольного диапазона $[a, b]$:

$$x'_i = a + \frac{(x_i - x_{\min})(b - a)}{x_{\max} - x_{\min}}$$

Свойства:

- Все значения гарантированно в заданном диапазоне
- Сохраняет форму распределения
- Чувствителен к выбросам (outliers)

4.2 Стандартизация (Z-score Normalization)

Преобразование к распределению со средним 0 и стандартным отклонением 1:

$$x'_i = \frac{x_i - \mu}{\sigma}$$

где:

- $\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$ — среднее значение
- $\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2}$ — стандартное отклонение

Свойства:

- Среднее становится равным 0, дисперсия равна 1
- Менее чувствителен к выбросам, чем Min-Max
- Не ограничивает значения определённым диапазоном
- Предпочтителен для алгоритмов, предполагающих нормальное распределение

4.3 Robust Scaling

Устойчивый к выбросам метод, использующий медиану и межквартильный размах (IQR):

$$x'_i = \frac{x_i - \text{median}(x)}{\text{IQR}(x)} = \frac{x_i - Q_2}{Q_3 - Q_1}$$

4.4 Сравнение методов

Метод	Диапазон	Выбросы	Применение
Min-Max	$[0, 1]$	Чувствителен	Нейросети, изображения
Стандартизация	$(-\infty, +\infty)$	Умеренно	Большинство алгоритмов
Robust	$(-\infty, +\infty)$	Устойчив	Данные с выбросами

5 Инварианты табличного набора данных

Инварианты — это свойства данных, которые должны сохраняться или учитываться при обработке и моделировании.

5.1 Основные инварианты

1. **Инвариантность к перестановке строк**

   Порядок объектов в выборке обычно не несёт информации (исключение — временные ряды). Результат обучения не должен зависеть от порядка примеров.

2. **Инвариантность к перестановке столбцов**

   Для большинства алгоритмов порядок признаков не важен (исключение — свёрточные сети для изображений).

3. **Инвариантность к масштабированию**

   Для некоторых алгоритмов (деревья решений) результат не зависит от масштаба признаков. Для других (kNN, SVM) масштабирование критично.

4. **Инвариантность к кодированию**

   Независимость от кодирования (способа представления) числовых и категориальных признаков.

5.2 Требования к данным

- **IID-предположение** (Independent and Identically Distributed) — объекты независимы и одинаково распределены
- **Представительность** — обучающая выборка должна отражать генеральную совокупность
- **Однородность** — данные получены из одного источника/процесса

5.3 Проблемы нарушения инвариантов

- **Data Leakage** — утечка информации из тестовой выборки в обучающую
- **Concept Drift** — изменение распределения данных со временем
- **Sample Selection Bias** — систематическое смещение в отборе данных

6 Сильный и слабый ИИ. Тест Тьюринга

6.1 Слабый (узкий) ИИ (Narrow AI / Weak AI)

Слабый ИИ — системы, предназначенные для решения конкретных, узкоспециализированных задач.

Характеристики:

- Решает одну или несколько связанных задач
- Не обладает общим интеллектом или сознанием
- Не понимает контекст за пределами своей задачи
- Все современные системы ИИ относятся к этому типу

Примеры:

- Распознавание изображений
- Рекомендательные системы
- Голосовые ассистенты
- Шахматные программы (Deep Blue, Stockfish)
- Системы машинного перевода

6.2 Сильный (общий) ИИ (General AI / Strong AI / AGI)

Сильный ИИ (AGI — Artificial General Intelligence) — гипотетическая система, обладающая интеллектом человеческого уровня.

Характеристики:

- Способность решать произвольные интеллектуальные задачи
- Понимание, рассуждение, обучение в любой области
- Возможное наличие сознания и самосознания
- Способность к переносу знаний между областями
- На данный момент не существует

6.3 Тест Тьюринга

Тест Тьюринга (1950) — критерий определения, может ли машина «мыслить».

Процедура:

1. Человек-судья ведёт текстовый диалог с двумя собеседниками
2. Один собеседник — человек, другой — машина
3. Судья не знает, кто есть кто
4. Задача судьи — определить, кто из собеседников машина
5. Если судья не может надёжно отличить машину от человека, машина проходит тест

Критика теста Тьюринга:

- **Китайская комната** (Джон Сёрл) — система может манипулировать символами без понимания их смысла
- Тест проверяет имитацию, а не реальное мышление
- Человек может не пройти тест (например, из-за языкового барьера)
- Тест антропоцентричен — измеряет похожесть на человека, а не интеллект

Альтернативные тесты:

- **Тест кофе** (Возняк) — сделать кофе в незнакомой кухне
- **Тест студента** — поступить в университет и получить диплом
- **Тест занятости** — выполнять экономически значимую работу

7 Дилемма смещения-дисперсии

Bias-Variance Tradeoff — фундаментальная проблема машинного обучения, описывающая компромисс между двумя источниками ошибок.

7.1 Разложение ошибки

Для задачи регрессии с истинной функцией $f(x)$ и моделью $\hat{f}(x)$ математическое ожидание квадрата ошибки:

$$\mathbb{E}\left[(y - \hat{f}(x))^2\right] = \underbrace{\text{Bias}^2[\hat{f}(x)]}_{\text{смещение}} + \underbrace{\text{Var}[\hat{f}(x)]}_{\text{дисперсия}} + \underbrace{\sigma^2}_{\text{шум}}$$

где:

- $\text{Bias}[\hat{f}(x)] = \mathbb{E}[\hat{f}(x)] - f(x)$ — смещение (систематическая ошибка)
- $\text{Var}[\hat{f}(x)] = \mathbb{E}\left[(\hat{f}(x) - \mathbb{E}[\hat{f}(x)])^2\right]$ — дисперсия (разброс предсказаний)
- σ^2 — неустранимая ошибка (шум в данных)

7.2 Смещение (Bias)

Смещение — ошибка из-за упрощающих предположений модели.

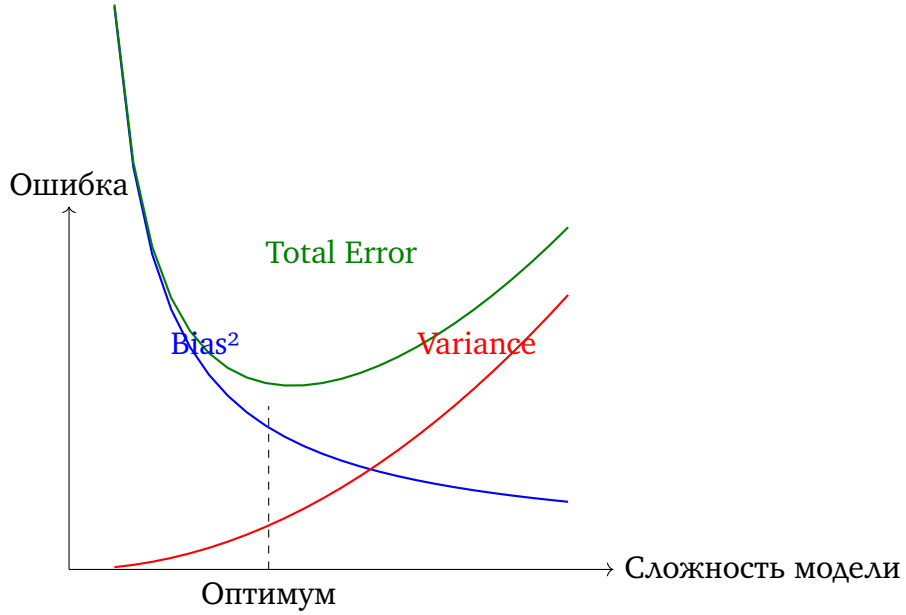
- **Высокое смещение**: модель слишком простая, не может уловить закономерности
- Приводит к **недообучению** (underfitting)
- Примеры: линейная регрессия для нелинейных данных

7.3 Дисперсия (Variance)

Дисперсия — чувствительность модели к флуктуациям в обучающих данных.

- **Высокая дисперсия**: модель слишком сложная, запоминает шум
- Приводит к **переобучению** (overfitting)
- Примеры: полином высокой степени, глубокое дерево решений

7.4 Компромисс



Методы борьбы:

- С высоким смещением: увеличить сложность модели, добавить признаки
- С высокой дисперсией: регуляризация, больше данных, ансамбли, dropout

8 Гиперпараметры. Регуляризация

8.1 Гиперпараметры

Параметры модели — значения, которые модель обучает на данных (веса, коэффициенты).

Гиперпараметры — настройки, задаваемые до обучения, управляющие процессом обучения или структурой модели.

Примеры гиперпараметров:

- Скорость обучения (learning rate) в градиентном спуске
- Количество деревьев в случайном лесе
- Глубина дерева решений
- Количество соседей k в kNN
- Коэффициент регуляризации λ
- Размер батча, количество эпох
- Архитектура нейросети (количество слоёв, нейронов)

Методы подбора гиперпараметров:

- Grid Search (перебор по сетке)
- Random Search (случайный поиск)
- Bayesian Optimization
- Optuna, Hyperopt

8.2 Регуляризация

Регуляризация — методы предотвращения переобучения путём ограничения сложности модели.

8.2.1 L2-регуляризация (Ridge, гребневая регрессия)

Добавление штрафа на квадрат нормы весов:

$$L_{\text{Ridge}} = L_{\text{original}} + \lambda \sum_{j=1}^{m} w_j^2 = L_{\text{original}} + \lambda \|\mathbf{w}\|_2^2$$

Свойства:

- Уменьшает веса, но не обнуляет их
- Устойчива к мультиколлинеарности
- Имеет аналитическое решение

8.2.2 L1-регуляризация (Lasso)

Добавление штрафа на сумму модулей весов:

$$L_{\text{Lasso}} = L_{\text{original}} + \lambda \sum_{j=1}^{m} |w_j| = L_{\text{original}} + \lambda \|\mathbf{w}\|_1$$

Свойства:

- Обнуляет некоторые веса (отбор признаков)
- Создаёт разреженные модели
- Не имеет аналитического решения

8.2.3 Elastic Net

Комбинация L1 и L2 регуляризации:

$$L_{\text{ElasticNet}} = L_{\text{original}} + \lambda_1 \|\mathbf{w}\|_1 + \lambda_2 \|\mathbf{w}\|_2^2$$

или с параметром смешивания $\alpha \in [0, 1]$:

$$L_{\text{ElasticNet}} = L_{\text{original}} + \lambda \left(\alpha \|\mathbf{w}\|_1 + (1 - \alpha) \|\mathbf{w}\|_2^2 \right)$$

8.2.4 Другие методы регуляризации

- **Dropout** — случайное отключение нейронов при обучении
- **Early Stopping** — остановка обучения при ухудшении на валидации
- **Data Augmentation** — увеличение объёма данных
- **Batch Normalization** — нормализация активаций
- **Pruning** — обрезка деревьев решений

9 Валидация на отложенных данных. Перекрёстная проверка

9.1 Проблема оценки качества

Нельзя оценивать качество модели на тех же данных, на которых она обучалась — это приведёт к оптимистичной оценке и не покажет способность к обобщению.

9.2 Hold-out валидация (отложенная выборка)

Данные разбиваются на части:

- **Обучающая выборка** (train set) — для обучения модели (60-80%)
- **Валидационная выборка** (validation set) — для подбора гиперпараметров (10-20%)
- **Тестовая выборка** (test set) — для финальной оценки (10-20%)

Преимущества:

- Простота реализации
- Быстрота (одно обучение)

Недостатки:

- Результат зависит от конкретного разбиения
- Неэффективно при малом объёме данных
- Высокая дисперсия оценки

9.3 Перекрёстная проверка (Cross-Validation)

9.3.1 K-Fold Cross-Validation

1. Данные разбиваются на K равных частей (фолдов)
2. Для каждого i от 1 до K:
   - i-й фолд используется как валидационный
   - Остальные $K-1$ фолдов — как обучающие
   - Модель обучается и оценивается
3. Итоговая оценка — среднее по всем фолдам

$$\text{CV Score} = \frac{1}{K}\sum_{i=1}^{K} \text{Score}_i$$

Типичные значения: $K = 5$ или $K = 10$.

9.3.2 Stratified K-Fold

Стратифицированная версия, сохраняющая пропорции классов в каждом фолде. Важна для несбалансированных данных.

9.3.3 Leave-One-Out (LOO)

Частный случай K-Fold, где $K = n$ (количество объектов):

- Каждый объект по очереди становится валидационным
- Минимальное смещение оценки
- Очень высокая вычислительная сложность: $O(n)$ обучений

9.3.4 Repeated K-Fold

Многократное повторение K-Fold с разными разбиениями для уменьшения дисперсии оценки.

9.3.5 Time Series Split

Для временных рядов: обучение всегда на прошлых данных, валидация на будущих:

- Fold 1: Train [1], Test [2]
- Fold 2: Train [1,2], Test [3]
- Fold 3: Train [1,2,3], Test [4]
- ...

9.4 Сравнение методов

Метод	Смещение	Дисперсия	Время
Hold-out	Высокое	Высокая	Низкое
5-Fold CV	Среднее	Средняя	Среднее
10-Fold CV	Низкое	Средняя	Высокое
LOO	Очень низкое	Высокая	Очень высокое

10 ROC-AUC, F-мера, Матрица ошибок

10.1 Матрица ошибок (Confusion Matrix)

Для задачи бинарной классификации матрица ошибок имеет вид:

	Предсказано: 1	**Предсказано: 0**
Истинно: 1	TP (True Positive)	FN (False Negative)
Истинно: 0	FP (False Positive)	TN (True Negative)

Обозначения:

- **TP** (True Positive) — истинно положительные (правильно предсказанные положительные)
- **TN** (True Negative) — истинно отрицательные (правильно предсказанные отрицательные)
- **FP** (False Positive) — ложноположительные (ошибка I рода, «ложная тревога»)
- **FN** (False Negative) — ложноотрицательные (ошибка II рода, «пропуск цели»)

10.2 Базовые метрики классификации

Accuracy (точность, доля правильных ответов):

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

Precision (точность, положительная прогностическая ценность):

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

Доля действительно положительных среди всех предсказанных положительных.

Recall (полнота, чувствительность, TPR):

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

Доля найденных положительных среди всех истинно положительных.

Specificity (специфичность, TNR):

$$\text{Specificity} = \text{TNR} = \frac{TN}{TN + FP}$$

False Positive Rate (FPR):

$$\text{FPR} = \frac{FP}{FP + TN} = 1 - \text{Specificity}$$

10.3 F-мера (F-score)

Гармоническое среднее между Precision и Recall:

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2TP}{2TP + FP + FN}$$

F_β-мера — обобщение с параметром β:

$$F_\beta = (1 + \beta^2) \cdot \frac{\text{Precision} \cdot \text{Recall}}{\beta^2 \cdot \text{Precision} + \text{Recall}}$$

- $\beta = 1$: баланс между Precision и Recall
- $\beta < 1$: больший вес Precision
- $\beta > 1$: больший вес Recall (например, F_2 для медицинской диагностики)

10.4 ROC-кривая и AUC

ROC-кривая (Receiver Operating Characteristic) — график зависимости TPR от FPR при различных порогах классификации.

- Ось X: FPR (False Positive Rate)
- Ось Y: TPR (True Positive Rate = Recall)
- Диагональ $y = x$ соответствует случайному классификатору
- Идеальный классификатор проходит через точку $(0, 1)$

AUC (Area Under Curve) — площадь под ROC-кривой:

$$\text{AUC} = \int_0^1 \text{TPR}(\text{FPR}^{-1}(x))\, dx$$

Интерпретация AUC:

- AUC $= 1.0$ — идеальный классификатор
- AUC $= 0.5$ — случайное угадывание
- AUC < 0.5 — хуже случайного (инвертировать предсказания)
- $0.7 \leq$ AUC < 0.8 — приемлемое качество
- $0.8 \leq$ AUC < 0.9 — хорошее качество
- AUC ≥ 0.9 — отличное качество

Вероятностная интерпретация: AUC равна вероятности того, что случайно выбранный положительный пример получит более высокий score, чем случайно выбранный отрицательный.

10.5 PR-кривая

Precision-Recall кривая — график зависимости Precision от Recall.

Предпочтительна для несбалансированных данных, когда отрицательный класс значительно преобладает.

AP (Average Precision) — площадь под PR-кривой.

11 MSE, RMSE, NRMSE, MAE, MAPE, SMAPE

Метрики для оценки качества регрессионных моделей.

11.1 MAE (Mean Absolute Error)

Средняя абсолютная ошибка:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i|$$

Свойства:

- Интерпретируется в единицах целевой переменной
- Робастна к выбросам (по сравнению с MSE)
- Не дифференцируема в нуле

11.2 MSE (Mean Squared Error)

Среднеквадратичная ошибка:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

Свойства:

- Сильнее штрафует большие ошибки

- Дифференцируема (удобна для оптимизации)
- Чувствительна к выбросам
- Единицы измерения — квадрат единиц целевой переменной

11.3 RMSE (Root Mean Squared Error)

Корень из среднеквадратичной ошибки:

$$\text{RMSE} = \sqrt{\text{MSE}} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

Свойства:

- Интерпретируется в единицах целевой переменной
- RMSE $\geq$ MAE (равенство при одинаковых ошибках)

11.4 NRMSE (Normalized RMSE)

Нормализованная RMSE для сравнения моделей на разных данных:

$$\text{NRMSE} = \frac{\text{RMSE}}{y_{\max} - y_{\min}} \quad \text{или} \quad \text{NRMSE} = \frac{\text{RMSE}}{\bar{y}}$$

11.5 MAPE (Mean Absolute Percentage Error)

Средняя абсолютная процентная ошибка:

$$\text{MAPE} = \frac{100\%}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right|$$

Свойства:

- Безразмерная (в процентах)
- Не определена при $y_i = 0$
- Асимметрична: недооценка и переоценка штрафуются по-разному

11.6 SMAPE (Symmetric MAPE)

Симметричная версия MAPE:

$$\text{SMAPE} = \frac{100\%}{n}\sum_{i=1}^{n}\frac{|y_i - \hat{y}_i|}{(|y_i| + |\hat{y}_i|)/2}$$

Свойства:

- Диапазон: $[0\%, 200\%]$
- Симметрична относительно переоценки и недооценки
- Проблема при $y_i = \hat{y}_i = 0$

11.7 R^2 (Коэффициент детерминации)

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} = 1 - \frac{\mathrm{SS}_{\text{res}}}{\mathrm{SS}_{\text{tot}}}$$

Интерпретация:

- Доля дисперсии, объяснённая моделью
- $R^2 = 1$ — идеальная модель
- $R^2 = 0$ — модель не лучше константного предсказания $\bar{y}$
- $R^2 < 0$ — модель хуже константы

11.8 Сравнение метрик

Метрика	Выбросы	Интерпретация	Оптимизация
MAE	Устойчива	В единицах y	Медиана
MSE	Чувствительна	В y^2	Среднее
RMSE	Чувствительна	В единицах y	Среднее
MAPE	Умеренно	В %	—
R^2	Чувствительна	Безразмерная	—

12 Метод окна Парзена. Формула Надарая-Ватсона

12.1 Метод окна Парзена (Parzen Window)

Метод окна Парзена — непараметрический метод оценки плотности распределения.
Для оценки плотности $p(x)$ по выборке $\{x_1, \ldots, x_n\}$:

$$\hat{p}(x) = \frac{1}{nh^d} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right)$$

где:

- $K(\cdot)$ — ядерная функция (kernel)
- h — ширина окна (bandwidth), параметр сглаживания
- d — размерность пространства

Требования к ядру:

- $K(u) \geq 0$ — неотрицательность
- $\int K(u)du = 1$ — нормировка
- $K(-u) = K(u)$ — симметричность (обычно)

Примеры ядер:

Ядро	Формула
Прямоугольное	$K(u) = \frac{1}{2}\mathbf{1}_{\lvert u\rvert \leq 1}$
Треугольное	$K(u) = (1 - \lvert u\rvert)\mathbf{1}_{\lvert u\rvert \leq 1}$
Епанечникова	$K(u) = \frac{3}{4}(1 - u^2)\mathbf{1}_{\lvert u\rvert \leq 1}$
Гауссово	$K(u) = \frac{1}{\sqrt{2\pi}} e^{-u^2/2}$

Выбор ширины окна h:

- Малое h: модель с высокой дисперсией, «пилообразная» оценка
- Большое h: модель с высоким смещением, сглаженная оценка
- Правило Сильвермана: $h = 1.06 \cdot \hat{\sigma} \cdot n^{-1/5}$

12.2 Формула Надарая-Ватсона

Оценка Надарая-Ватсона — ядерная регрессия, обобщение метода окон на задачу регрессии.

Для пар (x_i, y_i) предсказание в точке x:

$$\hat{y}(x) = \frac{\sum_{i=1}^{n} K\left(\frac{x-x_i}{h}\right) y_i}{\sum_{i=1}^{n} K\left(\frac{x-x_i}{h}\right)} = \sum_{i=1}^{n} w_i(x) \cdot y_i$$

где веса:

$$w_i(x) = \frac{K\left(\frac{x-x_i}{h}\right)}{\sum_{j=1}^{n} K\left(\frac{x-x_j}{h}\right)}$$

Интерпретация: взвешенное среднее значений y_i, где веса определяются близостью x_i к точке x.

Свойства:

- Непараметрический метод — не предполагает конкретную форму зависимости
- Локальная модель — предсказание зависит только от ближайших точек
- Сумма весов равна 1: $\sum_i w_i(x) = 1$
- Является частным случаем локально взвешенной регрессии

13 Ядерное сглаживание. Ядро для kNN

13.1 Ядерное сглаживание (Kernel Smoothing)

Ядерное сглаживание — семейство методов непараметрической регрессии, использующих ядерные функции для взвешивания наблюдений.

Общая формула:

$$\hat{f}(x) = \sum_{i=1}^{n} w_i(x) \cdot y_i$$

где веса $w_i(x)$ определяются ядерной функцией и убывают с расстоянием от x.

Виды ядерного сглаживания:

1. **Оценка Надарая-Ватсона** — локально константная модель
2. **Локально линейная регрессия** — в каждой точке строится линейная модель
3. **Локально полиномиальная регрессия** — полином степени p

13.2 Ядро для kNN

В методе k ближайших соседей (kNN) можно использовать ядерные функции для взвешивания соседей.

Стандартный kNN (без ядра):

$$\hat{y}(x) = \frac{1}{k} \sum_{i \in N_k(x)} y_i$$

где $N_k(x)$ — множество k ближайших соседей точки x.

Взвешенный kNN с ядром:

$$\hat{y}(x) = \frac{\sum_{i \in N_k(x)} K\left(\frac{\rho(x,x_i)}{h}\right) y_i}{\sum_{i \in N_k(x)} K\left(\frac{\rho(x,x_i)}{h}\right)}$$

где $\rho(x, x_i)$ — расстояние между x и x_i.

Адаптивная ширина окна:

$$h(x) = \rho(x, x_{(k)})$$

где $x_{(k)}$ — k-й ближайший сосед. Это делает ширину окна адаптивной к локальной плотности данных.

Популярные веса для kNN:

- **Равные веса:** $w_i = 1/k$
- **По расстоянию:** $w_i = 1/\rho(x, x_i)$
- **Гауссово ядро:** $w_i = \exp\left(-\frac{\rho(x,x_i)^2}{2h^2}\right)$
- **Епанечникова:** $w_i = \left(1 - \left(\frac{\rho(x,x_i)}{h}\right)^2\right)_+$

14 SMOTE. Tomek Links. LOWESS

14.1 SMOTE (Synthetic Minority Over-sampling Technique)

SMOTE — метод борьбы с дисбалансом классов путём генерации синтетических примеров миноритарного класса.

Алгоритм:

1. Для каждого примера x_i миноритарного класса:
2. Найти k ближайших соседей того же класса
3. Случайно выбрать одного соседа x_j
4. Создать синтетический пример на отрезке между x_i и x_j:

$$x_{\text{new}} = x_i + \lambda \cdot (x_j - x_i), \quad \lambda \sim U(0,1)$$

Варианты SMOTE:

- **Borderline-SMOTE** — генерация только для граничных примеров
- **SMOTE-ENN** — комбинация с удалением шумных примеров
- **ADASYN** — адаптивная генерация с учётом сложности примеров

14.2 Tomek Links

Tomek Links — метод очистки данных (undersampling), удаляющий неинформативные или шумные примеры.

Определение: Пара (x_i, x_j) образует Tomek Link, если:

- x_i и x_j принадлежат разным классам
- x_i — ближайший сосед x_j
- x_j — ближайший сосед x_i

Применение:

- Удаление примеров мажоритарного класса из Tomek Links
- Или удаление обоих примеров пары
- Очищает границу между классами
- Часто комбинируется со SMOTE

14.3 LOWESS / LOESS

LOWESS (Locally Weighted Scatterplot Smoothing) / **LOESS** (Local Regression) — метод локально взвешенной полиномиальной регрессии.

Алгоритм для предсказания в точке x_0:

1. Определить окрестность: k ближайших соседей или точки в радиусе h
2. Вычислить веса по расстоянию (обычно трикубическое ядро):

$$w_i = \left(1 - \left(\frac{|x_i - x_0|}{d_{\max}}\right)^3\right)^3$$

3. Решить взвешенную задачу наименьших квадратов:

$$\min_{\beta} \sum_{i=1}^{n} w_i (y_i - \beta_0 - \beta_1 (x_i - x_0))^2$$

4. Предсказание: $\hat{y}(x_0) = \hat{\beta}_0$

Параметры:

- Степень полинома (обычно 1 или 2)
- Ширина окна α — доля точек в окрестности

Применение:

- Сглаживание зашумлённых данных
- Визуализация трендов
- Выявление нелинейных зависимостей
- Робастная версия устойчива к выбросам

15 Метод максимального правдоподобия

Метод максимального правдоподобия (Maximum Likelihood Estimation, MLE) — способ оценки параметров статистической модели.

15.1 Функция правдоподобия

Пусть $X = \{x_1, \ldots, x_n\}$ — выборка из распределения с плотностью $p(x|\theta)$, где θ — неизвестный параметр.

Функция правдоподобия (likelihood function):

$$L(\theta|X) = \prod_{i=1}^{n} p(x_i|\theta)$$

Лог-правдоподобие (log-likelihood):

$$\ell(\theta) = \log L(\theta) = \sum_{i=1}^{n} \log p(x_i|\theta)$$

15.2 Оценка максимального правдоподобия

$$\hat{\theta}_{\text{MLE}} = \arg\max_{\theta} L(\theta) = \arg\max_{\theta} \ell(\theta)$$

Нахождение:

1. Записать функцию правдоподобия $L(\theta)$
2. Взять логарифм: $\ell(\theta) = \log L(\theta)$
3. Найти производную: $\dfrac{\partial \ell}{\partial \theta} = 0$
4. Решить уравнение относительно θ
5. Проверить, что это максимум (вторая производная отрицательна)

15.3 Примеры

Нормальное распределение $\mathcal{N}(\mu, \sigma^2)$:

$$\hat{\mu}_{\text{MLE}} = \bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

$$\hat{\sigma}^2_{\text{MLE}} = \frac{1}{n}\sum_{i=1}^{n} (x_i - \bar{x})^2$$

Распределение Бернулли Bernoulli(p):

$$\hat{p}_{\text{MLE}} = \frac{1}{n}\sum_{i=1}^{n} x_i = \frac{\text{число успехов}}{n}$$

15.4 Свойства MLE-оценок

- **Состоятельность**: $\hat{\theta}_n \xrightarrow{P} \theta$ при $n \to \infty$
- **Асимптотическая нормальность**: $\sqrt{n}(\hat{\theta}_n - \theta) \xrightarrow{d} \mathcal{N}(0, I^{-1}(\theta))$
- **Асимптотическая эффективность**: достигает границы Крамера-Рао
- **Инвариантность**: если $\hat{\theta}$ — MLE для θ, то $g(\hat{\theta})$ — MLE для $g(\theta)$

15.5 Связь с машинным обучением

- Минимизация MSE эквивалентна MLE при нормальном шуме
- Логистическая регрессия — MLE для модели Бернулли
- Кросс-энтропия — отрицательное лог-правдоподобие

16 Линейная регрессия. МНК

16.1 Модель линейной регрессии

Линейная регрессия — модель зависимости целевой переменной от признаков:

$$y = w_0 + w_1 x_1 + w_2 x_2 + \ldots + w_m x_m + \varepsilon = \mathbf{w}^T \mathbf{x} + \varepsilon$$

В матричной форме для n объектов:

$$\mathbf{y} = X\mathbf{w} + \boldsymbol{\varepsilon}$$

где:

- $\mathbf{y} \in \mathbb{R}^n$ — вектор целевых значений
- $X \in \mathbb{R}^{n \times (m+1)}$ — матрица признаков (с единичным столбцом для w_0)
- $\mathbf{w} \in \mathbb{R}^{m+1}$ — вектор весов
- $\boldsymbol{\varepsilon}$ — вектор ошибок

16.2 Метод наименьших квадратов (МНК / OLS)

Задача: минимизировать сумму квадратов отклонений:

$$L(\mathbf{w}) = \sum_{i=1}^{n} (y_i - \mathbf{w}^T \mathbf{x}_i)^2 = \|\mathbf{y} - X\mathbf{w}\|_2^2 \to \min_{\mathbf{w}}$$

Решение: Берём градиент и приравниваем к нулю:

$$\nabla_{\mathbf{w}} L = -2X^T(\mathbf{y} - X\mathbf{w}) = 0$$

$$X^T X \mathbf{w} = X^T \mathbf{y}$$

Нормальное уравнение:

$$\hat{\mathbf{w}} = (X^T X)^{-1} X^T \mathbf{y}$$

Псевдообратная матрица Мура-Пенроуза:

$$X^+ = (X^T X)^{-1} X^T$$

16.3 Условия применимости

1. **Линейность** — зависимость линейна по параметрам
2. **Независимость ошибок** — $\text{Cov}(\varepsilon_i, \varepsilon_j) = 0$ для $i \neq j$
3. **Гомоскедастичность** — $\text{Var}(\varepsilon_i) = \sigma^2 = \text{const}$
4. **Нормальность ошибок** — $\varepsilon \sim \mathcal{N}(0, \sigma^2)$ (для статистических тестов)
5. **Отсутствие мультиколлинеарности** — столбцы X линейно независимы

16.4 Проблема мультиколлинеарности

Если признаки сильно коррелированы, матрица $X^T X$ плохо обусловлена:

- Большая дисперсия оценок весов
- Неустойчивость решения
- Решение: регуляризация (Ridge) или отбор признаков

16.5 Теорема Гаусса-Маркова

При выполнении условий 1-3, МНК-оценка является **BLUE** (Best Linear Unbiased Estimator):

- **Best** — минимальная дисперсия среди всех линейных несмещённых оценок
- **Linear** — линейная функция от **y**
- **Unbiased** — $\mathbb{E}[\hat{\mathbf{w}}] = \mathbf{w}$

17 Сингулярное разложение (SVD)

Сингулярное разложение (Singular Value Decomposition, SVD) — представление матрицы в виде произведения трёх матриц.

17.1 Определение

Для любой матрицы $A \in \mathbb{R}^{m \times n}$ существует разложение:

$$A = U \Sigma V^T$$

где:

- $U \in \mathbb{R}^{m \times m}$ — ортогональная матрица левых сингулярных векторов
- $\Sigma \in \mathbb{R}^{m \times n}$ — диагональная матрица сингулярных значений
- $V \in \mathbb{R}^{n \times n}$ — ортогональная матрица правых сингулярных векторов

Сингулярные значения $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r > 0$, где $r = \text{rank}(A)$.

17.2 Свойства

- $U^TU = I_m$, $V^TV = I_n$ (ортогональность)
- Столбцы U — собственные векторы AA^T
- Столбцы V — собственные векторы A^TA
- σ_i^2 — собственные значения A^TA (и AA^T)
- $\|A\|_2 = \sigma_1$ — спектральная норма
- $\|A\|_F = \sqrt{\sum_i \sigma_i^2}$ — норма Фробениуса

17.3 Усечённое SVD (Truncated SVD)

Приближение матрицы рангом $k < r$:

$$A_k = U_k\Sigma_k V_k^T = \sum_{i=1}^{k} \sigma_i \mathbf{u}_i \mathbf{v}_i^T$$

Теорема Эккарта-Янга: A_k — наилучшее приближение ранга k в смысле норм $\|\cdot\|_2$ и $\|\cdot\|_F$.

17.4 Применения в ML

1. **Решение линейных систем**:
$$\hat{\mathbf{w}} = V\Sigma^{-1}U^T\mathbf{y}$$
(устойчивее прямого обращения)
2. **PCA** (метод главных компонент):
$$X = U\Sigma V^T \Rightarrow \text{PC} = XV = U\Sigma$$
3. **Снижение размерности**: Проекция на k главных компонент
4. **Латентный семантический анализ (LSA)**: Разложение матрицы документ-терм
5. **Рекомендательные системы**: Матричная факторизация
6. **Сжатие изображений**: Аппроксимация низким рангом
7. **Псевдообратная матрица**:
$$A^+ = V\Sigma^+U^T$$

18 Регуляризация. L1, L2, ElasticNet (расширенное)

18.1 Геометрическая интерпретация

Регуляризация эквивалентна решению задачи с ограничением:

L2 (Ridge):

$$\min_{\mathbf{w}} \|X\mathbf{w} - \mathbf{y}\|^2 \quad \text{при} \quad \|\mathbf{w}\|_2^2 \le t$$

Ограничение — шар. Решение обычно не на осях.

L1 (Lasso):

$$\min_{\mathbf{w}} \|X\mathbf{w} - \mathbf{y}\|^2 \quad \text{при} \quad \|\mathbf{w}\|_1 \le t$$

Ограничение — ромб (гиперкуб). Решение часто на вершинах/рёбрах $\Rightarrow$ разреженность.

18.2 Аналитические решения

Ridge регрессия:

$$\hat{\mathbf{w}}_{\text{Ridge}} = (X^T X + \lambda I)^{-1} X^T \mathbf{y}$$

Через SVD: $X = U\Sigma V^T$:

$$\hat{\mathbf{w}}_{\text{Ridge}} = V(\Sigma^2 + \lambda I)^{-1}\Sigma U^T \mathbf{y}$$

Эффект на сингулярные значения:

$$\hat{w}_j = \frac{\sigma_j^2}{\sigma_j^2 + \lambda} w_j^{\text{OLS}}$$

При больших λ коэффициенты «сжимаются» к нулю.

18.3 Байесовская интерпретация

- **L2**: априорное распределение весов — $\mathbf{w} \sim \mathcal{N}(0, \tau^2 I)$
- **L1**: априорное распределение весов — Лапласа: $p(w_j) \propto e^{-|w_j|/b}$

MAP-оценка (Maximum A Posteriori) при нормальном шуме совпадает с регуляризованным МНК.

18.4 Выбор λ

- Перекрёстная проверка (Cross-Validation)
- Информационные критерии (AIC, BIC)
- L-кривая (L-curve)
- Обобщённая перекрёстная проверка (GCV)

19 Линейная классификация. Логистическая регрессия

19.1 Линейный классификатор

Линейный классификатор разделяет классы гиперплоскостью:

$$a(\mathbf{x}) = \text{sign}(\mathbf{w}^T\mathbf{x} + b) = \text{sign}\left(\sum_{j=1}^{m} w_j x_j + b\right)$$

Разделяющая гиперплоскость: $\mathbf{w}^T\mathbf{x} + b = 0$
Свойства:

- Вектор $\mathbf{w}$ — нормаль к гиперплоскости
- Расстояние от точки $\mathbf{x}$ до гиперплоскости: $\dfrac{|\mathbf{w}^T\mathbf{x} + b|}{\|\mathbf{w}\|}$
- Отступ (margin): $M_i = y_i(\mathbf{w}^T\mathbf{x}_i + b)$

19.2 Логистическая регрессия

Логистическая регрессия — линейный классификатор, предсказывающий вероятность класса.

Модель:

$$P(y = 1|\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w}^T\mathbf{x}}}$$

где $\sigma(z) = \dfrac{1}{1 + e^{-z}}$ — сигмоидная функция (логистическая функция).

Свойства сигмоиды:

- $\sigma(z) \in (0, 1)$
- $\sigma(-z) = 1 - \sigma(z)$
- $\sigma'(z) = \sigma(z)(1 - \sigma(z))$
- $\sigma(0) = 0.5$

19.3 Функция потерь

Лог-правдоподобие (для $y \in \{0, 1\}$):

$$\ell(\mathbf{w}) = \sum_{i=1}^{n} [y_i \log \hat{p}_i + (1 - y_i) \log(1 - \hat{p}_i)]$$

Логистическая функция потерь (logloss, cross-entropy):

$$L(\mathbf{w}) = -\frac{1}{n} \sum_{i=1}^{n} [y_i \log \hat{p}_i + (1 - y_i) \log(1 - \hat{p}_i)]$$

Для $y \in \{-1, +1\}$:

$$L(\mathbf{w}) = \frac{1}{n} \sum_{i=1}^{n} \log(1 + e^{-y_i \mathbf{w}^T \mathbf{x}_i})$$

19.4 Оптимизация

Аналитического решения нет. Используется:

- Градиентный спуск
- Стохастический градиентный спуск (SGD)
- Метод Ньютона-Рафсона (IRLS)
- L-BFGS

Градиент:

$$\nabla L = \frac{1}{n} \sum_{i=1}^{n} (\hat{p}_i - y_i)\mathbf{x}_i = \frac{1}{n} X^T(\hat{\mathbf{p}} - \mathbf{y})$$

19.5 Многоклассовая классификация

One-vs-Rest (OvR):

- Обучить K бинарных классификаторов
- Предсказать класс с максимальной уверенностью

Multinomial (Softmax) регрессия:

$$P(y = k|\mathbf{x}) = \frac{e^{\mathbf{w}_k^T \mathbf{x}}}{\sum_{j=1}^{K} e^{\mathbf{w}_j^T \mathbf{x}}}$$

19.6 Регуляризация в логистической регрессии

$$L_{\text{reg}}(\mathbf{w}) = L(\mathbf{w}) + \lambda R(\mathbf{w})$$

- L2: $R(\mathbf{w}) = \|\mathbf{w}\|_2^2$
- L1: $R(\mathbf{w}) = \|\mathbf{w}\|_1$
- ElasticNet: $R(\mathbf{w}) = \alpha\|\mathbf{w}\|_1 + (1-\alpha)\|\mathbf{w}\|_2^2$

20 Градиентный спуск

Градиентный спуск (Gradient Descent) — итеративный метод оптимизации для нахождения минимума функции.

20.1 Основная идея

Градиент $\nabla f(\mathbf{w})$ указывает направление наибольшего роста функции. Двигаясь в противоположном направлении, мы уменьшаем значение функции:

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta \nabla f(\mathbf{w}^{(t)})$$

где $\eta > 0$ — **скорость обучения** (learning rate).

20.2 Виды градиентного спуска

20.2.1 Batch Gradient Descent (полный)

Градиент вычисляется по всей выборке:

$$\nabla L(\mathbf{w}) = \frac{1}{n} \sum_{i=1}^{n} \nabla \ell_i(\mathbf{w})$$

Свойства:

- Стабильная сходимость
- Высокие вычислительные затраты на итерацию
- Требует хранения всех данных в памяти

20.2.2 Stochastic Gradient Descent (SGD)

Градиент вычисляется по одному случайному примеру:

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta \nabla \ell_{i_t}(\mathbf{w}^{(t)})$$

Свойства:

- Быстрые итерации
- Высокий шум в оценке градиента
- Может выходить из локальных минимумов
- Требует уменьшения η со временем

20.2.3 Mini-batch Gradient Descent

Компромисс: градиент по подвыборке размера B:

$$\nabla L \approx \frac{1}{B} \sum_{i \in \text{batch}} \nabla \ell_i(\mathbf{w})$$

Типичные размеры батча: 32, 64, 128, 256.

20.3 Выбор скорости обучения

- **Слишком большая** η: расходимость, осцилляции
- **Слишком маленькая** η: медленная сходимость
- **Расписание** (learning rate schedule): уменьшение η со временем

20.4 Продвинутые оптимизаторы

20.4.1 Momentum (импульс)

Добавление «инерции» для сглаживания траектории:

$$\mathbf{v}^{(t+1)} = \gamma \mathbf{v}^{(t)} + \eta \nabla L(\mathbf{w}^{(t)})$$

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \mathbf{v}^{(t+1)}$$

где $\gamma \approx 0.9$ — коэффициент импульса.

20.4.2 AdaGrad

Адаптивная скорость для каждого параметра:

$$w_j^{(t+1)} = w_j^{(t)} - \frac{\eta}{\sqrt{G_{jj}^{(t)} + \varepsilon}} \cdot g_j^{(t)}$$

где $G_{jj}^{(t)} = \sum_{\tau=1}^{t} (g_j^{(\tau)})^2$ — накопленная сумма квадратов градиентов.

20.4.3 RMSprop

Экспоненциальное скользящее среднее квадратов градиентов:

$$v_j^{(t)} = \beta v_j^{(t-1)} + (1-\beta)(g_j^{(t)})^2$$

$$w_j^{(t+1)} = w_j^{(t)} - \frac{\eta}{\sqrt{v_j^{(t)} + \varepsilon}} \cdot g_j^{(t)}$$

20.4.4 Adam (Adaptive Moment Estimation)

Комбинация momentum и RMSprop:

$$m^{(t)} = \beta_1 m^{(t-1)} + (1-\beta_1) g^{(t)}$$

$$v^{(t)} = \beta_2 v^{(t-1)} + (1-\beta_2)(g^{(t)})^2$$

$$\hat{m}^{(t)} = \frac{m^{(t)}}{1-\beta_1^t}, \quad \hat{v}^{(t)} = \frac{v^{(t)}}{1-\beta_2^t}$$

$$w^{(t+1)} = w^{(t)} - \frac{\eta}{\sqrt{\hat{v}^{(t)}} + \varepsilon} \hat{m}^{(t)}$$

Типичные значения: $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\varepsilon = 10^{-8}$.

20.5 Условия сходимости

Для выпуклой функции с липшицевым градиентом:

- GD сходится со скоростью $O(1/t)$
- Для сильно выпуклых функций: $O(\rho^t)$ (линейная сходимость)
- SGD: $O(1/\sqrt{t})$ при убывающем η_t

21 SoftArgMax. SoftMax

21.1 Функция Softmax

Softmax — функция, преобразующая вектор вещественных чисел в вектор вероятностей:

$$\text{softmax}(\mathbf{z})_k = \frac{e^{z_k}}{\sum_{j=1}^{K} e^{z_j}}, \quad k = 1, \ldots, K$$

Свойства:

- $\text{softmax}(\mathbf{z})_k \in (0, 1)$ для всех k
- $\sum_{k=1}^{K} \text{softmax}(\mathbf{z})_k = 1$ (нормировка)
- Сохраняет порядок: если $z_i > z_j$, то $\text{softmax}(z)_i > \text{softmax}(z)_j$
- Инвариантна к сдвигу: $\text{softmax}(\mathbf{z} + c) = \text{softmax}(\mathbf{z})$

21.2 Температурный параметр

$$\text{softmax}(\mathbf{z}; T)_k = \frac{e^{z_k/T}}{\sum_{j=1}^{K} e^{z_j/T}}$$

- $T \to 0$: приближается к argmax (жёсткий выбор)
- $T = 1$: стандартный softmax
- $T \to \infty$: равномерное распределение

21.3 SoftArgMax

SoftArgMax — дифференцируемая аппроксимация функции argmax:

$$\text{softargmax}(\mathbf{z}) = \sum_{k=1}^{K} k \cdot \text{softmax}(\mathbf{z})_k$$

Возвращает «мягкий» индекс максимального элемента — взвешенную сумму индексов.
Альтернативная интерпретация:

$$\text{softargmax}(\mathbf{z}) = \mathbb{E}_{k \sim \text{softmax}(\mathbf{z})}[k]$$

21.4 Применение в машинном обучении

1. **Многоклассовая классификация**:

$$P(y = k|\mathbf{x}) = \text{softmax}(\mathbf{W}\mathbf{x} + \mathbf{b})_k$$

2. **Механизм внимания** (Attention):

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

3. **Выходной слой нейросети** для классификации
4. **Policy в reinforcement learning**

21.5 Численная стабильность

Для предотвращения переполнения используют трюк:

$$\text{softmax}(\mathbf{z})_k = \frac{e^{z_k - \max_j z_j}}{\sum_{j=1}^{K} e^{z_j - \max_j z_j}}$$

21.6 Связь с кросс-энтропией

Функция потерь для многоклассовой классификации:

$$L = -\sum_{k=1}^{K} y_k \log \text{softmax}(\mathbf{z})_k$$

Градиент имеет простой вид:

$$\frac{\partial L}{\partial z_k} = \text{softmax}(\mathbf{z})_k - y_k$$

22 Метод опорных векторов (SVM)

Метод опорных векторов (Support Vector Machine, SVM) — алгоритм классификации, максимизирующий зазор между классами.

22.1 Линейно разделимый случай

Для линейно разделимых данных ищем гиперплоскость $\mathbf{w}^T\mathbf{x} + b = 0$, максимизирующую **зазор** (margin).

Зазор — расстояние от гиперплоскости до ближайших точек каждого класса:

$$\text{margin} = \frac{2}{\|\mathbf{w}\|}$$

Задача оптимизации:

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2$$

$$\text{при } y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1, \quad i = 1, \ldots, n$$

22.2 Опорные векторы

Опорные векторы — точки, лежащие на границе зазора:

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) = 1$$

- Только опорные векторы определяют решение
- Удаление других точек не изменит гиперплоскость
- Обычно опорных векторов немного

22.3 Мягкий зазор (Soft Margin)

Для линейно неразделимых данных вводятся **slack-переменные** $\xi_i \geq 0$:

$$\min_{\mathbf{w},b,\boldsymbol{\xi}} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n} \xi_i$$

$$\text{при } y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0$$

Параметр C:

- Большое C: жёсткий зазор, меньше ошибок на обучении
- Малое C: мягкий зазор, лучшее обобщение

22.4 Двойственная задача

Через множители Лагранжа $\alpha_i \geq 0$:

$$\max_{\boldsymbol{\alpha}} \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{n} \alpha_i\alpha_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j$$

$$\text{при } \sum_{i=1}^{n} \alpha_i y_i = 0, \quad 0 \leq \alpha_i \leq C$$

Решение:

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$$

Предсказание:

$$f(\mathbf{x}) = \text{sign}\left(\sum_{i:\alpha_i>0} \alpha_i y_i \mathbf{x}_i^T \mathbf{x} + b\right)$$

22.5 Функции потерь SVM

Hinge loss (кусочно-линейная):

$$\ell(y, f(\mathbf{x})) = \max(0, 1 - y \cdot f(\mathbf{x})) = (1 - y \cdot f(\mathbf{x}))_+$$

SVM минимизирует:

$$\frac{1}{n}\sum_{i=1}^{n} \max(0, 1 - y_i(\mathbf{w}^T\mathbf{x}_i + b)) + \frac{\lambda}{2}\|\mathbf{w}\|^2$$

23 Ядро для метода опорных векторов

23.1 Ядерный трюк (Kernel Trick)

Двойственная задача SVM зависит только от скалярных произведений $\mathbf{x}_i^T\mathbf{x}_j$.

Идея: заменить скалярное произведение на **ядерную функцию**:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j)$$

где $\phi : \mathbb{R}^m \to \mathcal{H}$ — отображение в пространство признаков (возможно, бесконечномерное).

Преимущество: не нужно явно вычислять $\phi(\mathbf{x})$!

23.2 Условие Мерсера

Функция $K(\mathbf{x}, \mathbf{x}')$ является допустимым ядром, если:

- K симметрична: $K(\mathbf{x}, \mathbf{x}') = K(\mathbf{x}', \mathbf{x})$
- Матрица Грама $K_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$ положительно полуопределена для любого набора точек

23.3 Популярные ядра

1. **Линейное ядро:**
$$K(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T\mathbf{x}'$$
2. **Полиномиальное ядро:**
$$K(\mathbf{x}, \mathbf{x}') = (\gamma\mathbf{x}^T\mathbf{x}' + r)^d$$
где d — степень полинома, γ, r — параметры.
3. **RBF (Radial Basis Function) / Гауссово ядро:**
$$K(\mathbf{x}, \mathbf{x}') = \exp\left(-\gamma\|\mathbf{x} - \mathbf{x}'\|^2\right) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\sigma^2}\right)$$
Соответствует бесконечномерному пространству признаков.
4. **Сигмоидное ядро:**
$$K(\mathbf{x}, \mathbf{x}') = \tanh(\gamma\mathbf{x}^T\mathbf{x}' + r)$$
(не всегда положительно определено)

23.4 Выбор ядра и параметров

- **RBF** — универсальный выбор, хорошо работает в большинстве случаев
- Малое γ в RBF: гладкая граница, возможно недообучение
- Большое γ: сложная граница, возможно переобучение
- Подбор C и γ через перекрёстную проверку

23.5 Свойства ядер

Если K_1 и K_2 — допустимые ядра, то:

- $K_1 + K_2$ — ядро
- $c \cdot K_1$ для $c > 0$ — ядро
- $K_1 \cdot K_2$ — ядро
- $f(\mathbf{x})K_1(\mathbf{x}, \mathbf{x}')f(\mathbf{x}')$ — ядро

24 Формула Байеса. Наивный байесовский классификатор

24.1 Формула Байеса

Теорема Байеса связывает условные вероятности:

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$$

В контексте классификации:

$$P(y|\mathbf{x}) = \frac{P(\mathbf{x}|y) \cdot P(y)}{P(\mathbf{x})}$$

Терминология:

- $P(y)$ — **априорная** вероятность (prior)
- $P(\mathbf{x}|y)$ — **правдоподобие** (likelihood)
- $P(y|\mathbf{x})$ — **апостериорная** вероятность (posterior)
- $P(\mathbf{x})$ — **свидетельство** (evidence), нормировочная константа

24.2 Байесовский классификатор

Оптимальный байесовский классификатор предсказывает класс с максимальной апостериорной вероятностью:

$$\hat{y} = \arg\max_y P(y|\mathbf{x}) = \arg\max_y P(\mathbf{x}|y)P(y)$$

(знаменатель $P(\mathbf{x})$ не зависит от y)

24.3 Наивный байесовский классификатор

Предположение наивного Байеса: признаки условно независимы при известном классе:

$$P(\mathbf{x}|y) = P(x_1, x_2, \ldots, x_m|y) = \prod_{j=1}^{m} P(x_j|y)$$

Классификатор:

$$\hat{y} = \arg\max_y P(y) \prod_{j=1}^{m} P(x_j|y)$$

Для численной стабильности используют логарифм:

$$\hat{y} = \arg\max_y \left[\log P(y) + \sum_{j=1}^{m} \log P(x_j|y) \right]$$

24.4 Виды наивного Байеса

24.4.1 Categorical (Multinomial) Naive Bayes

Для категориальных признаков или счётчиков слов:

$$P(x_j = v|y = c) = \frac{N_{cjv} + \alpha}{N_c + \alpha|V_j|}$$

где N_{cjv} — количество примеров класса c с $x_j = v$.

24.4.2 Gaussian Naive Bayes

Для непрерывных признаков с нормальным распределением:

$$P(x_j|y = c) = \frac{1}{\sqrt{2\pi\sigma_{cj}^2}} \exp\left(-\frac{(x_j - \mu_{cj})^2}{2\sigma_{cj}^2} \right)$$

24.4.3 Bernoulli Naive Bayes

Для бинарных признаков:

$$P(\mathbf{x}|y = c) = \prod_{j=1}^{m} p_{cj}^{x_j} (1 - p_{cj})^{1-x_j}$$

24.5 Преимущества и недостатки

Преимущества:

- Простота и быстрота обучения: $O(nm)$
- Хорошо работает при малом объёме данных
- Устойчив к нерелевантным признакам
- Интерпретируемость

Недостатки:

- Предположение независимости часто нарушается
- Плохо калиброванные вероятности
- Проблема нулевых вероятностей (решается сглаживанием)

25 Оценка плотности. Сглаживание Лапласа

25.1 Оценка плотности

Оценка плотности (Density Estimation) — задача восстановления функции плотности распределения по выборке.

Параметрические методы:

- Предполагают определённое семейство распределений
- Оценивают параметры (например, μ, σ для нормального)
- MLE, метод моментов

Непараметрические методы:

- Не предполагают конкретную форму распределения
- Гистограмма, ядерная оценка плотности (KDE)

25.2 Проблема нулевых вероятностей

В наивном Байесе, если значение признака не встречалось в обучающей выборке:

$$P(x_j = v|y = c) = 0 \Rightarrow P(y = c|\mathbf{x}) = 0$$

Одно ненаблюдавшееся значение полностью «убивает» класс!

25.3 Сглаживание Лапласа (Additive Smoothing)

Сглаживание Лапласа — добавление псевдосчётчиков для предотвращения нулевых вероятностей:

$$P(x_j = v|y = c) = \frac{N_{cjv} + \alpha}{N_c + \alpha|V_j|}$$

где:

- N_{cjv} — количество наблюдений $(x_j = v, y = c)$
- N_c — общее количество примеров класса c
- $|V_j|$ — количество возможных значений признака x_j
- $\alpha > 0$ — параметр сглаживания

Частные случаи:

- $\alpha = 1$ — сглаживание Лапласа (Laplace smoothing)
- $\alpha = 0.5$ — сглаживание Джеффриса (Jeffreys)
- $\alpha < 1$ — сглаживание Лидстоуна (Lidstone)

25.4 Байесовская интерпретация

Сглаживание Лапласа эквивалентно байесовской оценке с априорным распределением Дирихле:

$$\boldsymbol{\theta} \sim \text{Dir}(\alpha, \alpha, \ldots, \alpha)$$

При $\alpha = 1$ это равномерное априорное распределение.

25.5 Сглаживание для текстов

Для модели мешка слов:

$$P(w|c) = \frac{\text{count}(w,c) + \alpha}{\sum_{w' \in V} \text{count}(w',c) + \alpha|V|}$$

где $|V|$ — размер словаря.

26 Сезонность, периодичность, тренд, стационарность

Основные компоненты временных рядов.

26.1 Тренд (Trend)

Тренд — долгосрочная тенденция изменения временного ряда.

Типы трендов:

- Линейный: $T_t = a + bt$
- Полиномиальный: $T_t = a_0 + a_1 t + a_2 t^2 + \ldots$
- Экспоненциальный: $T_t = a \cdot e^{bt}$
- Логистический (S-образный)

Выделение тренда:

- Скользящее среднее
- Регрессия на время
- Экспоненциальное сглаживание
- Разностный оператор: $\nabla y_t = y_t - y_{t-1}$

26.2 Сезонность (Seasonality)

Сезонность — периодические колебания с фиксированным периодом.

Характеристики:

- Период известен заранее (день, неделя, год)
- Регулярное повторение паттерна
- Примеры: продажи мороженого летом, трафик в часы пик

Модели сезонности:

- **Аддитивная**: $Y_t = T_t + S_t + \varepsilon_t$
- **Мультипликативная**: $Y_t = T_t \cdot S_t \cdot \varepsilon_t$

26.3 Периодичность (Cyclicity)

Цикличность — колебания с непостоянным или неизвестным периодом.

- Период может меняться
- Связана с экономическими/бизнес-циклами
- Труднее моделировать, чем сезонность

26.4 Стационарность (Stationarity)

Стационарный процесс — статистические свойства не меняются со временем.

Слабая (ковариационная) стационарность:

1. $\mathbb{E}[Y_t] = \mu = \text{const}$ (постоянное среднее)
2. $\text{Var}(Y_t) = \sigma^2 = \text{const}$ (постоянная дисперсия)
3. $\text{Cov}(Y_t, Y_{t+h}) = \gamma(h)$ (автоковариация зависит только от лага h)

Проверка стационарности:

- Визуальный анализ графика
- Тест Дики-Фуллера (ADF test)
- KPSS тест
- Анализ автокорреляционной функции (ACF)

Приведение к стационарности:

- Дифференцирование: $\nabla y_t = y_t - y_{t-1}$
- Логарифмирование (для стабилизации дисперсии)
- Сезонное дифференцирование: $\nabla_s y_t = y_t - y_{t-s}$

27 AR. MA. ARMA. ARIMA. SARIMA

Семейство моделей для анализа и прогнозирования временных рядов.

27.1 AR (Autoregressive) — авторегрессия

AR(p) — модель, где текущее значение зависит от p предыдущих:

$$Y_t = c + \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \ldots + \phi_p Y_{t-p} + \varepsilon_t$$

или в операторной форме:

$$\Phi(L) Y_t = c + \varepsilon_t$$

где $\Phi(L) = 1 - \phi_1 L - \ldots - \phi_p L^p$, L — оператор лага ($LY_t = Y_{t-1}$).

Условие стационарности: корни характеристического уравнения $\Phi(z) = 0$ по модулю больше 1.

Свойства ACF и PACF:

- ACF: экспоненциальное/колебательное затухание
- PACF: обрывается после лага p

27.2 MA (Moving Average) — скользящее среднее

MA(q) — модель, где текущее значение зависит от q предыдущих ошибок:

$$Y_t = c + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \ldots + \theta_q \varepsilon_{t-q}$$

или:

$$Y_t = c + \Theta(L)\varepsilon_t$$

где $\Theta(L) = 1 + \theta_1 L + \ldots + \theta_q L^q$.

Свойства:

- Всегда стационарна
- ACF: обрывается после лага q
- PACF: экспоненциальное затухание

27.3 ARMA — авторегрессия со скользящим средним

ARMA(p, q) — комбинация AR(p) и MA(q):

$$Y_t = c + \sum_{i=1}^{p} \phi_i Y_{t-i} + \varepsilon_t + \sum_{j=1}^{q} \theta_j \varepsilon_{t-j}$$

$$\Phi(L) Y_t = c + \Theta(L)\varepsilon_t$$

Применима только к стационарным рядам.

27.4 ARIMA — интегрированная ARMA

ARIMA(p, d, q) — ARMA после d-кратного дифференцирования:

$$\Phi(L)(1 - L)^d Y_t = c + \Theta(L)\varepsilon_t$$

- p — порядок авторегрессии
- d — порядок интегрирования (число дифференцирований)
- q — порядок скользящего среднего

Выбор параметров:

- d: по тесту ADF или визуально
- p, q: по ACF/PACF или информационным критериям (AIC, BIC)

27.5 SARIMA — сезонная ARIMA

SARIMA(p, d, q)(P, D, Q)$_s$ — ARIMA с сезонной компонентой:

$$\Phi(L)\Phi_s(L^s)(1 - L)^d(1 - L^s)^D Y_t = c + \Theta(L)\Theta_s(L^s)\varepsilon_t$$

где:

- (p, d, q) — несезонные параметры
- (P, D, Q) — сезонные параметры
- s — период сезонности (12 для месячных данных с годовой сезонностью)

27.6 Процедура Бокса-Дженкинса

1. **Идентификация**: анализ ACF/PACF, тесты стационарности
2. **Оценка**: подбор параметров (MLE)
3. **Диагностика**: анализ остатков (должны быть белым шумом)
4. **Прогнозирование**: если модель адекватна

28 n-граммы, BPE, word2vec

Методы представления текстов и слов.

28.1 n-граммы

n-грамма — последовательность из n элементов (символов или слов).

Примеры (для текста «машинное обучение»):

- Символьные биграммы: «ма», «аш», «ши», «ин», ...
- Символьные триграммы: «маш», «аши», «шин», ...
- Словесные биграммы: («машинное», «обучение»)

Языковая модель на n-граммах:

$$P(w_t|w_1, \ldots, w_{t-1}) \approx P(w_t|w_{t-n+1}, \ldots, w_{t-1})$$

Применение:

- Представление текстов (Bag of n-grams)
- Языковое моделирование
- Определение языка
- Детекция опечаток

28.2 BPE (Byte Pair Encoding)

BPE — алгоритм сегментации текста на подслова.

Алгоритм обучения:

1. Начать со словаря отдельных символов
2. Найти наиболее частую пару соседних токенов
3. Объединить их в новый токен
4. Повторять до достижения нужного размера словаря

Пример: «low», «lower», «lowest» $\rightarrow$ «low», «low» + «er», «low» + «est»

Преимущества:

- Фиксированный размер словаря
- Нет проблемы OOV (out-of-vocabulary)
- Баланс между символами и словами
- Используется в GPT, RoBERTa

Варианты: WordPiece (BERT), Unigram (SentencePiece)

28.3 Word2Vec

Word2Vec — метод обучения плотных векторных представлений слов.

Идея: слова с похожим контекстом должны иметь похожие векторы.

28.3.1 Skip-gram

Предсказание контекста по центральному слову:

$$P(w_{\text{context}}|w_{\text{center}}) = \frac{\exp(\mathbf{v}_{w_c}^T \mathbf{v}_{w_t})}{\sum_{w \in V} \exp(\mathbf{v}_w^T \mathbf{v}_{w_t})}$$

Функция потерь:

$$L = -\sum_{t=1}^{T} \sum_{-c \le j \le c, j \neq 0} \log P(w_{t+j}|w_t)$$

28.3.2 CBOW (Continuous Bag of Words)

Предсказание центрального слова по контексту:

$$P(w_{\text{center}}|w_{\text{context}})$$

28.3.3 Оптимизации

- **Negative Sampling**: вместо softmax по всему словарю — бинарная классификация
- **Hierarchical Softmax**: дерево Хаффмана для ускорения
- **Subsampling**: прореживание частых слов

28.4 Свойства word2vec

Семантическая арифметика:

$$\vec{\text{king}} - \vec{\text{man}} + \vec{\text{woman}} \approx \vec{\text{queen}}$$

Параметры:

- Размерность векторов (100-300)
- Размер окна контекста (5-10)
- Минимальная частота слова

Альтернативы: GloVe, FastText, ELMo, BERT

29 Лемматизация и стемминг

Методы нормализации слов для обработки естественного языка.

29.1 Стемминг (Stemming)

Стемминг — отсечение окончаний и суффиксов для получения основы слова.

Особенности:

- Использует эвристические правила
- Быстрый, не требует словаря
- Результат может не быть реальным словом
- Может давать ошибки (over-stemming, under-stemming)

Популярные стеммеры:

- **Porter Stemmer** — классический для английского
- **Snowball** — многоязычный
- **Lancaster** — агрессивный
- **Mystem** — для русского (Яндекс)

Пример (Porter):

- «running» → «run»
- «studies» → «studi»
- «presumably» → «presum»

29.2 Лемматизация (Lemmatization)

Лемматизация — приведение слова к начальной форме (лемме) с учётом морфологии.

Особенности:

- Требует морфологического анализа
- Использует словари и правила языка
- Результат — реальное слово
- Медленнее стемминга
- Точнее стемминга

Инструменты:

- **WordNet Lemmatizer** (NLTK)
- **spaCy**
- **pymorphy2** — для русского
- **Mystem** — для русского

Пример:

- «running» → «run»
- «better» → «good»
- «studies» → «study»
- «шёл» → «идти»

29.3 Сравнение

Критерий	Стемминг	Лемматизация
Скорость	Быстрый	Медленный
Точность	Низкая	Высокая
Результат	Основа (не слово)	Лемма (слово)
Требования	Правила	Словарь + правила
Учёт контекста	Нет	Частично

29.4 Выбор метода

- **Стемминг**: поиск, IR-системы, когда важна скорость
- **Лемматизация**: анализ текста, когда важна точность
- Современные модели (BERT) часто не требуют явной нормализации

30 Деревья принятия решений

Дерево решений (Decision Tree) — модель, представляющая последовательность условий в виде древовидной структуры.

30.1 Структура дерева

- **Корень** — начальный узел
- **Внутренние узлы** — условия разбиения (предикаты вида $x_j < t$)
- **Рёбра** — результаты проверки условий
- **Листья** — финальные предсказания (класс или значение)

30.2 Критерии разбиения для классификации

30.2.1 Энтропия и Information Gain

Энтропия — мера неопределённости:

$$H(S) = -\sum_{k=1}^{K} p_k \log_2 p_k$$

где p_k — доля примеров класса k в выборке S.
Information Gain — уменьшение энтропии после разбиения:

$$IG(S, A) = H(S) - \sum_{v \in \text{values}(A)} \frac{|S_v|}{|S|} H(S_v)$$

30.2.2 Индекс Джини (Gini Impurity)

$$Gini(S) = 1 - \sum_{k=1}^{K} p_k^2 = \sum_{k=1}^{K} p_k(1 - p_k)$$

Интерпретация: вероятность неправильной классификации случайного примера.

30.2.3 Сравнение критериев

- Gini вычислительно проще (нет логарифма)
- Энтропия более чувствительна к изменению распределения
- На практике дают схожие результаты

30.3 Критерии для регрессии

MSE (дисперсия):

$$\text{MSE}(S) = \frac{1}{|S|} \sum_{i \in S} (y_i - \bar{y}_S)^2$$

MAE:

$$\text{MAE}(S) = \frac{1}{|S|} \sum_{i \in S} |y_i - \text{median}(S)|$$

30.4 Алгоритмы построения

- **ID3** — только категориальные признаки, Information Gain
- **C4.5** — числовые признаки, Gain Ratio (нормализованный IG)
- **CART** — бинарные деревья, Gini или MSE

30.5 Регуляризация деревьев

Для предотвращения переобучения:

- **max_depth** — максимальная глубина
- **min_samples_split** — минимум примеров для разбиения
- **min_samples_leaf** — минимум примеров в листе
- **max_leaf_nodes** — максимальное число листьев
- **Pruning** (обрезка) — удаление поддеревьев после построения

30.6 Преимущества и недостатки

Преимущества:

- Интерпретируемость
- Работа с разными типами данных
- Не требует масштабирования
- Инвариантность к монотонным преобразованиям
- Автоматический отбор признаков

Недостатки:

- Склонность к переобучению

- Нестабильность (малое изменение данных — другое дерево)
- Жадный алгоритм (локальный оптимум)
- Плохо аппроксимируют гладкие функции

31 Ансамбли. Бэггинг и стэкинг

Ансамблевые методы — объединение нескольких моделей для улучшения качества предсказаний.

31.1 Идея ансамблей

Мудрость толпы: усреднение независимых предсказаний уменьшает дисперсию ошибки.

Для M независимых моделей с одинаковой дисперсией σ^2:

$$\text{Var}\left(\frac{1}{M}\sum_{m=1}^{M} f_m\right) = \frac{\sigma^2}{M}$$

31.2 Бэггинг (Bootstrap Aggregating)

Алгоритм:

1. Создать M бутстреп-выборок (с возвращением) из обучающих данных
2. Обучить модель на каждой выборке
3. Агрегировать предсказания:
   - Классификация: голосование большинством
   - Регрессия: усреднение

$$\hat{f}(x) = \frac{1}{M}\sum_{m=1}^{M} f_m(x)$$

Out-of-Bag (OOB) оценка:

- Примерно 37% примеров не попадают в каждую бутстреп-выборку
- Можно использовать для валидации без отложенной выборки

31.2.1 Random Forest

Случайный лес — бэггинг деревьев решений с дополнительной рандомизацией:

- В каждом узле рассматривается случайное подмножество признаков (обычно $\sqrt{m}$ для классификации, $m/3$ для регрессии)
- Деревья строятся до максимальной глубины (без обрезки)

Свойства:

- Снижает корреляцию между деревьями
- Устойчив к переобучению
- Важность признаков (feature importance)

31.3 Стэкинг (Stacking)

Стэкинг — обучение мета-модели на предсказаниях базовых моделей.

Алгоритм:

1. Обучить M разнородных базовых моделей (level-0)
2. Получить их предсказания на валидационной выборке
3. Обучить мета-модель (level-1) на этих предсказаниях

Важно: использовать out-of-fold предсказания для обучения мета-модели (избежать утечки данных).

Мета-признаки:

- Предсказания базовых моделей
- Вероятности классов
- Можно добавить исходные признаки

31.4 Сравнение методов

Метод	Цель	Модели	Агрегация
Бэггинг	Снижение дисперсии	Однородные	Голосование/среднее
Стэкинг	Снижение смещения	Разнородные	Мета-модель
Бустинг	Снижение смещения	Последовательные	Взвешенная сумма

32 Задача бустинга и градиентный бустинг

32.1 Идея бустинга

Бустинг — последовательное построение ансамбля, где каждая следующая модель исправляет ошибки предыдущих.

$$F_M(x) = \sum_{m=1}^{M} \gamma_m h_m(x)$$

где h_m — базовые (слабые) модели, γ_m — их веса.

32.2 AdaBoost (Adaptive Boosting)

Алгоритм:

1. Инициализировать веса примеров: $w_i^{(1)} = 1/n$
2. Для $m = 1, \ldots, M$:
   - Обучить слабый классификатор h_m на взвешенной выборке
   - Вычислить взвешенную ошибку: $\varepsilon_m = \sum_{i:h_m(x_i)\neq y_i} w_i^{(m)}$
   - Вычислить вес модели: $\alpha_m = \frac{1}{2}\ln\frac{1-\varepsilon_m}{\varepsilon_m}$
   - Обновить веса: $w_i^{(m+1)} \propto w_i^{(m)} \exp(-\alpha_m y_i h_m(x_i))$
3. Итоговый классификатор: $F(x) = \text{sign}\left(\sum_{m=1}^{M} \alpha_m h_m(x)\right)$

32.3 Градиентный бустинг

Gradient Boosting — обобщение бустинга через градиентный спуск в функциональном пространстве.

Идея: каждая следующая модель обучается предсказывать **антиградиент** функции потерь.

Алгоритм:

1. Инициализация: $F_0(x) = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, \gamma)$
2. Для $m = 1, \ldots, M$:
   - Вычислить псевдо-остатки (отрицательный градиент):
   $$r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F=F_{m-1}}$$
   - Обучить базовую модель h_m на (x_i, r_{im})
   - Найти оптимальный шаг: $\gamma_m = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, F_{m-1}(x_i) + \gamma h_m(x_i))$
   - Обновить: $F_m(x) = F_{m-1}(x) + \eta\gamma_m h_m(x)$

где η — скорость обучения (shrinkage).

Псевдо-остатки для разных функций потерь:

- MSE: $r_i = y_i - F(x_i)$ (обычные остатки)
- MAE: $r_i = \text{sign}(y_i - F(x_i))$
- Logistic: $r_i = y_i - \sigma(F(x_i))$

32.4 Популярные реализации

- **XGBoost** — регуляризация, параллелизация, работа с пропусками
- **LightGBM** — leaf-wise рост, категориальные признаки, гистограммы
- **CatBoost** — ordered boosting, обработка категориальных признаков

32.5 Гиперпараметры градиентного бустинга

- **n_estimators** — количество деревьев
- **learning_rate** (η) — скорость обучения
- **max_depth** — глубина деревьев
- **subsample** — доля примеров для каждого дерева
- **colsample_bytree** — доля признаков
- **reg_alpha**, **reg_lambda** — L1/L2 регуляризация

33 Задача кластеризации. Внешние и внутренние меры

33.1 Задача кластеризации

Кластеризация — разбиение объектов на группы (кластеры) так, чтобы:

- Объекты внутри кластера были похожи друг на друга
- Объекты из разных кластеров были различны

Это задача **обучения без учителя** — метки классов неизвестны.

33.2 Типы кластеризации

- **Жёсткая** (hard): каждый объект принадлежит ровно одному кластеру
- **Мягкая** (soft/fuzzy): объект может принадлежать нескольким кластерам с разными степенями
- **Иерархическая**: кластеры образуют дерево (дендрограмму)

33.3 Внешние меры качества

Используются, когда известна истинная разметка (ground truth).

33.3.1 Rand Index (RI)

$$RI = \frac{a + d}{a + b + c + d} = \frac{a + d}{\binom{n}{2}}$$

где:

- a — пары, в одном кластере в обеих разметках
- d — пары, в разных кластерах в обеих разметках
- b, c — пары с расхождением разметок

33.3.2 Adjusted Rand Index (ARI)

Скорректированный на случайность:

$$ARI = \frac{RI - \mathbb{E}[RI]}{\max(RI) - \mathbb{E}[RI]}$$

- $ARI = 1$: идеальное совпадение
- $ARI = 0$: случайная разметка
- $ARI < 0$: хуже случайного

33.3.3 Normalized Mutual Information (NMI)

$$NMI = \frac{2 \cdot I(U; V)}{H(U) + H(V)}$$

где $I(U; V)$ — взаимная информация, H — энтропия.

33.3.4 Другие внешние меры

- **Homogeneity**: все объекты кластера из одного класса
- **Completeness**: все объекты класса в одном кластере
- **V-measure**: гармоническое среднее homogeneity и completeness
- **Fowlkes-Mallows Index**

33.4 Внутренние меры качества

Используются, когда истинная разметка неизвестна.

33.4.1 Silhouette Score

Для объекта i:

$$s(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))}$$

где:

- $a(i)$ — среднее расстояние до объектов своего кластера
- $b(i)$ — минимальное среднее расстояние до объектов другого кластера

$s(i) \in [-1, 1]$: чем больше, тем лучше.
Общий score: $\bar{s} = \frac{1}{n} \sum_i s(i)$

33.4.2 Индекс Дэвиса-Болдина (Davies-Bouldin Index)

$$DB = \frac{1}{K} \sum_{i=1}^{K} \max_{j \neq i} \frac{\sigma_i + \sigma_j}{d(c_i, c_j)}$$

где σ_i — средний разброс в кластере i, $d(c_i, c_j)$ — расстояние между центроидами.
Чем меньше DB, тем лучше.

33.4.3 Индекс Калински-Харабаса (Calinski-Harabasz)

$$CH = \frac{B/(K-1)}{W/(n-K)}$$

где B — межкластерная дисперсия, W — внутрикластерная дисперсия.
Чем больше CH, тем лучше.

33.4.4 Метод локтя (Elbow Method)

График зависимости внутрикластерной суммы квадратов от числа кластеров K. Ищут «локоть» — точку замедления убывания.

34 EM-алгоритмы. Алгоритм k-Means

34.1 Алгоритм k-Means

k-Means — итеративный алгоритм кластеризации, минимизирующий внутрикластерную дисперсию.

Целевая функция (инерция):

$$J = \sum_{k=1}^{K} \sum_{i \in C_k} \|x_i - \mu_k\|^2$$

Алгоритм:

1. Инициализация: выбрать K начальных центроидов $\mu_1, \ldots, \mu_K$
2. **E-шаг** (assignment): присвоить каждый объект ближайшему центроиду:

$$c_i = \arg\min_k \|x_i - \mu_k\|^2$$

3. **M-шаг** (update): пересчитать центроиды:

$$\mu_k = \frac{1}{|C_k|} \sum_{i \in C_k} x_i$$

4. Повторять 2-3 до сходимости

Свойства:

- Сходится за конечное число итераций
- Находит локальный минимум (зависит от инициализации)
- Сложность: $O(n \cdot K \cdot d \cdot T)$

34.1.1 Инициализация k-Means++

Улучшенная инициализация центроидов:

1. Выбрать первый центроид случайно
2. Для каждого следующего центроида выбрать точку с вероятностью пропорциональной $d^2(x, \mu)$

Гарантирует $O(\log K)$-аппроксимацию оптимума.

34.2 EM-алгоритм (Expectation-Maximization)

EM-алгоритм — итеративный метод для нахождения MLE при наличии скрытых переменных.

Задача: максимизировать неполное правдоподобие:

$$L(\theta) = \log p(X|\theta) = \log \sum_Z p(X, Z|\theta)$$

Алгоритм:

1. **E-шаг**: вычислить апостериорное распределение скрытых переменных:

$$Q(Z) = p(Z|X, \theta^{(t)})$$

2. **М-шаг**: максимизировать ожидаемое полное правдоподобие:

$$\theta^{(t+1)} = \arg\max_{\theta} \mathbb{E}_{Z\sim Q}[\log p(X, Z|\theta)]$$

Свойства:

- Монотонно увеличивает правдоподобие
- Сходится к локальному максимуму
- k-Means — частный случай EM для смеси гауссиан с фиксированными дисперсиями

34.3 Gaussian Mixture Models (GMM)

Смесь гауссиан:

$$p(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\mu_k, \Sigma_k)$$

где π_k — веса компонент, $\sum_k \pi_k = 1$.

EM для GMM:

- E-шаг: вычислить ответственности (responsibilities):

$$\gamma_{ik} = \frac{\pi_k \mathcal{N}(x_i|\mu_k, \Sigma_k)}{\sum_j \pi_j \mathcal{N}(x_i|\mu_j, \Sigma_j)}$$

- M-шаг: обновить параметры:

$$\mu_k = \frac{\sum_i \gamma_{ik} x_i}{\sum_i \gamma_{ik}}, \quad \Sigma_k = \frac{\sum_i \gamma_{ik}(x_i - \mu_k)(x_i - \mu_k)^T}{\sum_i \gamma_{ik}}$$

$$\pi_k = \frac{1}{n}\sum_i \gamma_{ik}$$

35 Графовые и плотностные алгоритмы кластеризации

35.1 DBSCAN (Density-Based Spatial Clustering)

DBSCAN — плотностный алгоритм, находящий кластеры произвольной формы.

Параметры:

- ε (eps) — радиус окрестности
- MinPts — минимальное число точек в ε-окрестности

Типы точек:

- **Core point**: в ε-окрестности $\geq$ MinPts точек
- **Border point**: в ε-окрестности core point, но сама не core
- **Noise point**: ни core, ни border

Алгоритм:

1. Найти все core points
2. Соединить core points, находящиеся в ε-окрестности друг друга
3. Каждая связная компонента — кластер
4. Присоединить border points к ближайшим кластерам

Преимущества:

- Не требует задания числа кластеров
- Находит кластеры произвольной формы
- Устойчив к выбросам (выделяет их как noise)

Недостатки:

- Чувствителен к выбору ε и MinPts
- Плохо работает с кластерами разной плотности
- Сложность $O(n^2)$ без пространственного индекса

35.2 HDBSCAN

HDBSCAN — иерархическая версия DBSCAN:

- Строит иерархию кластеров для разных ε
- Автоматически выбирает оптимальные кластеры
- Работает с кластерами разной плотности

35.3 Иерархическая кластеризация

Агломеративная (снизу вверх):

1. Каждый объект — отдельный кластер
2. На каждом шаге объединяем два ближайших кластера
3. Повторяем до одного кластера

Методы связывания (linkage):

- **Single**: $d(A, B) = \min_{a \in A, b \in B} d(a, b)$
- **Complete**: $d(A, B) = \max_{a \in A, b \in B} d(a, b)$
- **Average**: среднее расстояние между всеми парами
- **Ward**: минимизация увеличения внутрикластерной дисперсии

Дендрограмма — визуализация иерархии. Разрез на высоте h даёт кластеризацию.

35.4 Спектральная кластеризация

Идея: использовать собственные векторы матрицы смежности/лапласиана графа.

Алгоритм:

1. Построить граф близости (например, k-NN граф)
2. Вычислить матрицу Лапласа: $L = D - W$
3. Найти K собственных векторов с наименьшими собственными значениями
4. Применить k-Means к строкам матрицы собственных векторов

Преимущества:

- Находит кластеры сложной формы
- Теоретически обоснован (минимизация разреза графа)

36 Алгоритм PCA и t-SNE

36.1 PCA (Principal Component Analysis)

Метод главных компонент — линейное снижение размерности, максимизирующее дисперсию проекции.

Задача: найти ортогональные направления максимальной дисперсии.

Алгоритм:

1. Центрировать данные: $\tilde{X} = X - \bar{X}$
2. Вычислить ковариационную матрицу: $C = \frac{1}{n}\tilde{X}^T\tilde{X}$
3. Найти собственные значения и векторы: $Cv_k = \lambda_k v_k$
4. Отсортировать по убыванию λ_k
5. Выбрать k первых главных компонент
6. Проекция: $Z = \tilde{X}V_k$

Через SVD:

$$\tilde{X} = U\Sigma V^T \Rightarrow Z = U_k\Sigma_k$$

Доля объяснённой дисперсии:

$$\text{explained variance ratio} = \frac{\sum_{i=1}^{k}\lambda_i}{\sum_{i=1}^{m}\lambda_i}$$

Свойства:

- Линейный метод
- Сохраняет глобальную структуру
- Быстрый и масштабируемый
- Чувствителен к масштабу (нужна стандартизация)

36.2 t-SNE (t-Distributed Stochastic Neighbor Embedding)

t-SNE — нелинейный метод снижения размерности для визуализации.
Идея: сохранить локальные расстояния между точками.
Алгоритм:

1. В исходном пространстве: вычислить попарные вероятности:

$$p_{j|i} = \frac{\exp(-\|x_i - x_j\|^2/2\sigma_i^2)}{\sum_{k\neq i}\exp(-\|x_i - x_k\|^2/2\sigma_i^2)}$$

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$$

2. В целевом пространстве: использовать t-распределение:

$$q_{ij} = \frac{(1 + \|y_i - y_j\|^2)^{-1}}{\sum_{k\neq l}(1 + \|y_k - y_l\|^2)^{-1}}$$

3. Минимизировать KL-дивергенцию:

$$KL(P\|Q) = \sum_{i\neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

Параметр perplexity:

- Контролирует эффективное число соседей
- Типичные значения: 5-50
- Влияет на σ_i

Свойства t-SNE:

- Хорошо сохраняет локальную структуру
- Глобальные расстояния не сохраняются
- Стохастический (разные запуски — разные результаты)
- Вычислительно затратный: $O(n^2)$
- Только для визуализации (нельзя проецировать новые точки)

36.3 UMAP

UMAP (Uniform Manifold Approximation and Projection) — современная альтернатива t-SNE:

- Быстрее t-SNE
- Лучше сохраняет глобальную структуру
- Можно проецировать новые точки
- Теоретическое обоснование через топологию

37 Алгоритмы-обёртки и встроенные методы отбора признаков

Отбор признаков (Feature Selection) — выбор подмножества признаков для улучшения качества модели.

37.1 Алгоритмы-обёртки (Wrapper Methods)

Идея: использовать модель как «чёрный ящик» для оценки подмножеств признаков.

37.1.1 Forward Selection (прямой отбор)

1. Начать с пустого множества признаков
2. На каждом шаге добавить признак, дающий наибольшее улучшение
3. Остановиться, когда улучшение незначительно

37.1.2 Backward Elimination (обратное исключение)

1. Начать со всех признаков
2. На каждом шаге удалить наименее полезный признак
3. Остановиться, когда качество начинает падать

37.1.3 Recursive Feature Elimination (RFE)

1. Обучить модель на всех признаках
2. Удалить признаки с наименьшими весами
3. Повторить до нужного числа признаков

Преимущества обёрток:

- Учитывают взаимодействия признаков
- Адаптированы к конкретной модели

Недостатки:

- Высокая вычислительная сложность
- Риск переобучения
- Зависимость от выбора модели

37.2 Встроенные методы (Embedded Methods)

Идея: отбор признаков интегрирован в процесс обучения модели.

37.2.1 L1-регуляризация (Lasso)

$$\min_{\mathbf{w}} L(\mathbf{w}) + \lambda \|\mathbf{w}\|_1$$

L1 штраф обнуляет веса незначимых признаков.

37.2.2 Деревья решений и ансамбли

Feature Importance в Random Forest / Gradient Boosting:

- **Gini importance**: суммарное уменьшение Gini по всем разбиениям
- **Permutation importance**: падение качества при перемешивании признака

37.2.3 ElasticNet

Комбинация L1 и L2 — отбор признаков с группировкой коррелированных.

Преимущества встроенных методов:

- Эффективнее обёрток
- Отбор и обучение одновременно
- Учитывают специфику модели

38 Фильтрующие методы выбора признаков

Фильтры (Filter Methods) — оценка признаков независимо от модели, по статистическим критериям.

38.1 Методы для числовых признаков

38.1.1 Корреляция с целевой переменной

Корреляция Пирсона (для регрессии):

$$r = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_i (x_i - \bar{x})^2 \sum_i (y_i - \bar{y})^2}}$$

Отбираем признаки с $|r| >$ threshold.

38.1.2 Дисперсия

Удаление признаков с низкой дисперсией (почти константные):

$$\text{Var}(x_j) < \text{threshold}$$

38.1.3 Mutual Information

$$MI(X, Y) = \sum_{x,y} p(x, y) \log \frac{p(x, y)}{p(x)p(y)}$$

Измеряет нелинейную зависимость между признаком и целевой переменной.

38.2 Методы для категориальных признаков

38.2.1 Хи-квадрат (χ^2)

Для категориальных признаков и целевой переменной:

$$\chi^2 = \sum_{i,j} \frac{(O_{ij} - E_{ij})^2}{E_{ij}}$$

где O_{ij} — наблюдаемые частоты, E_{ij} — ожидаемые при независимости.

38.2.2 ANOVA F-test

Для числовых признаков и категориальной целевой:

$$F = \frac{\text{Between-group variance}}{\text{Within-group variance}}$$

38.3 Преимущества и недостатки фильтров

Преимущества:

- Быстрые, масштабируемые
- Независимы от модели
- Не переобучаются

Недостатки:

- Не учитывают взаимодействия признаков
- Могут выбрать избыточные признаки
- Не оптимизированы под конкретную модель

38.4 Сравнение методов отбора

Метод	Скорость	Точность	Взаимодействия
Фильтры	Высокая	Низкая	Нет
Обёртки	Низкая	Высокая	Да
Встроенные	Средняя	Высокая	Частично

39 Задача выбора алгоритма и настройки гиперпараметров

39.1 Проблема выбора модели

Model Selection — выбор:

- Типа алгоритма (линейный, дерево, нейросеть, ...)
- Значений гиперпараметров
- Подмножества признаков
- Архитектуры модели

39.2 AutoML

AutoML — автоматизация процесса машинного обучения:

- Предобработка данных
- Выбор признаков
- Выбор модели
- Настройка гиперпараметров
- Построение ансамблей

Инструменты: Auto-sklearn, H2O AutoML, TPOT, AutoGluon.

39.3 Стратегии оценки

- **Hold-out**: быстро, но высокая дисперсия
- **K-Fold CV**: стандартный подход
- **Nested CV**: для корректной оценки при подборе гиперпараметров

39.4 Nested Cross-Validation

Для несмещённой оценки при подборе гиперпараметров:

1. Внешний цикл: оценка обобщающей способности
2. Внутренний цикл: подбор гиперпараметров

Нельзя использовать тестовые данные для выбора модели!

40 Поиск по сетке. Случайный поиск

40.1 Grid Search (поиск по сетке)

Идея: перебрать все комбинации значений гиперпараметров.
Алгоритм:

1. Задать сетку значений для каждого гиперпараметра
2. Для каждой комбинации:
   - Обучить модель
   - Оценить на валидации (CV)
3. Выбрать комбинацию с лучшим score

Пример:

```
param_grid = {
    'C': [0.1, 1, 10, 100],
    'gamma': [0.01, 0.1, 1],
    'kernel': ['rbf', 'poly']
}
```

Всего $4 \times 3 \times 2 = 24$ комбинации.
Недостатки:

- Экспоненциальный рост числа комбинаций
- Неэффективен при большом числе гиперпараметров
- Равномерная сетка может пропустить оптимум

40.2 Random Search (случайный поиск)

Идея: случайно сэмплировать комбинации гиперпараметров.

Алгоритм:

1. Задать распределения для каждого гиперпараметра
2. Случайно выбрать N комбинаций
3. Для каждой комбинации оценить модель
4. Выбрать лучшую

Преимущества:

- Эффективнее Grid Search при большом числе параметров
- Лучше исследует пространство
- Можно задать бюджет по времени/числу итераций

Теоретическое обоснование (Bergstra & Bengio, 2012):

- При d гиперпараметрах обычно только несколько важны
- Random Search лучше покрывает важные измерения
- 60 случайных точек с вероятностью 95% найдут точку в топ-5%

40.3 Сравнение методов

Критерий	Grid Search	Random Search
Сложность	Экспоненциальная	Линейная
Полнота	Полный перебор	Неполный
Масштабируемость	Плохая	Хорошая
Простота	Высокая	Высокая

41 Optuna

Optuna — современный фреймворк для оптимизации гиперпараметров с использованием байесовской оптимизации.

41.1 Основные концепции

- **Study** — эксперимент по оптимизации
- **Trial** — одна попытка с конкретными гиперпараметрами
- **Objective function** — функция, которую минимизируем/максимизируем

41.2 Алгоритмы сэмплирования

41.2.1 TPE (Tree-structured Parzen Estimator)

TPE — основной алгоритм Optuna:

1. Разделить предыдущие trials на «хорошие» и «плохие» по порогу γ
2. Построить модели плотности $\ell(x)$ для хороших и $g(x)$ для плохих
3. Сэмплировать точки, максимизирующие $\ell(x)/g(x)$

41.2.2 Другие сэмплеры

- **Random Sampler** — случайный поиск
- **Grid Sampler** — поиск по сетке
- **CMA-ES** — эволюционная стратегия

41.3 Pruning (обрезка)

Early stopping для неперспективных trials:

- **MedianPruner**: останавливает trial, если промежуточный результат хуже медианы
- **HyperbandPruner**: адаптивное распределение ресурсов
- **SuccessiveHalvingPruner**: последовательное деление

41.4 Преимущества Optuna

- **Define-by-run API**: динамическое определение пространства поиска
- **Pruning**: раннее прекращение неперспективных экспериментов
- **Параллелизация**: распределённая оптимизация
- **Визуализация**: встроенные графики
- **Интеграции**: PyTorch, TensorFlow, XGBoost, LightGBM

41.5 Сравнение с другими фреймворками

Фреймворк	Алгоритм	Pruning	Параллелизм
Optuna	TPE, CMA-ES	Да	Да
Hyperopt	TPE	Нет	Ограничен
Ray Tune	Разные	Да	Да
Weights & Biases	Bayes, Random	Да	Да

42 Самообучение (self-training) и сообучение (co-training)

Методы **полу-контролируемого обучения** (semi-supervised learning) — использование немаркированных данных.

42.1 Самообучение (Self-Training)

Идея: модель итеративно размечает немаркированные данные и дообучается на них.
Алгоритм:

1. Обучить модель f на размеченных данных L
2. Repeat:
   - Предсказать метки для немаркированных данных U
   - Выбрать примеры с высокой уверенностью: $\{(x, \hat{y}) : \max_k P(y = k|x) > \tau\}$

- Добавить их в L, удалить из U
- Переобучить модель на расширенном L

3. Until: U пусто или нет уверенных предсказаний

Проблемы:

- Накопление ошибок (error propagation)
- Смещение к доминирующему классу
- Зависимость от порога уверенности τ

42.2 Сообучение (Co-Training)

Идея: две модели на разных «представлениях» данных обучают друг друга.

Предположения:

- Признаки можно разделить на два независимых подмножества
- Каждое подмножество достаточно для классификации

Алгоритм:

1. Разделить признаки: $X = (X^{(1)}, X^{(2)})$
2. Обучить два классификатора f_1 и f_2 на разных представлениях
3. Repeat:
   - f_1 размечает примеры для f_2 (с высокой уверенностью)
   - f_2 размечает примеры для f_1
   - Переобучить обе модели

Пример: классификация веб-страниц

- Представление 1: текст страницы
- Представление 2: тексты ссылок на страницу

42.3 Tri-Training

Расширение co-training на три модели:

- Пример добавляется в обучение третьей модели, если две другие согласны
- Не требует разделения признаков

42.4 Применение

- Разметка данных дорогая (медицина, право)
- Много немаркированных данных (веб, соцсети)
- Доменная адаптация

43 Активное обучение

Активное обучение (Active Learning) — стратегия выбора примеров для разметки, минимизирующая количество необходимых меток.

43.1 Сценарии активного обучения

- **Pool-based**: выбор из большого пула немаркированных примеров
- **Stream-based**: решение о разметке каждого поступающего примера
- **Query synthesis**: генерация синтетических примеров для разметки

43.2 Стратегии выбора примеров

43.2.1 Uncertainty Sampling

Выбирать примеры, в которых модель наименее уверена.

Критерии неуверенности:

- **Least Confidence**: $x^* = \arg\min_x \max_k P(y = k|x)$
- **Margin Sampling**: $x^* = \arg\min_x (P(y_1|x) - P(y_2|x))$ где y_1, y_2 — два наиболее вероятных класса
- **Entropy**: $x^* = \arg\max_x H(Y|x) = \arg\max_x \left(-\sum_k P(y = k|x) \log P(y = k|x) \right)$

43.2.2 Query-by-Committee (QBC)

1. Обучить комитет моделей на текущих данных
2. Выбрать примеры, по которым модели максимально расходятся
3. Мера разногласия: энтропия голосов, KL-дивергенция

43.2.3 Expected Model Change

Выбирать примеры, которые максимально изменят модель:

$$x^* = \arg\max_x \mathbb{E}_{Y|x} \|\nabla L(\theta; x, y)\|$$

43.2.4 Density-Weighted Methods

Комбинация неуверенности и репрезентативности:

$$\text{score}(x) = \text{uncertainty}(x) \times \text{representativeness}(x)$$

43.3 Batch Active Learning

Выбор нескольких примеров за раз:

- Учитывать разнообразие в батче
- Кластеризация + выбор из каждого кластера
- Core-set selection

43.4 Преимущества и ограничения

Преимущества:

- Снижение затрат на разметку в 2-10 раз
- Эффективное использование бюджета
- Фокус на информативных примерах

Ограничения:

- Требует обученной начальной модели
- Смещение выборки (sampling bias)
- Не всегда работает лучше случайного выбора
- Вычислительные затраты на переобучение

43.5 Применение

- Медицинская диагностика (дорогая экспертиза)
- Разметка изображений
- Обработка естественного языка
- Аннотация редких событий