ЛЕКЦИЯ 4. ПОИСК ЧАСТО ВСТРЕЧАЮЩИХСЯ НАБОРОВ ДАННЫХ (FREQUENT PATTERNS). АЛГОРИТМ APRIORI

ДИСЦИПЛИНА: «ИНТЕЛЛЕКТУАЛЬНЫЙ АНАЛИЗ ДАННЫХ»

ЦЕЛЬ ЛЕКЦИИ

- Сформировать у обучающихся представление о задаче поиска частых наборов (**Frequent Patterns**) в транзакционных данных, ввести понятия поддержки и минимальной поддержки, изучить свойство антимонотонности и алгоритм **Apriori**, а также научить применять его для поиска частых наборов на практическом примере.

ПЛАН ЛЕКЦИИ

- Поиск частых наборов (Frequent Patterns): постановка задачи
- Основные понятия: транзакция, набор, поддержка
- Пространство поиска и свойство антимонотонности
- Алгоритм Apriori: идея, генерация и отсев кандидатов
- Пошаговый пример работы алгоритма
- Сложность, оптимизации и ограничения Apriori
- Реализация на Python и переход к ассоциативным правилам

ЧТО ТАКОЕ ЧАСТЫЕ НАБОРЫ

- **Частые наборы (Frequent Patterns)** — наборы элементов, подпоследовательности или подструктуры, которые встречаются в данных не реже заданного порога.
- Задача поставлена Р. Агравалом, Т. Имелинским и А. Свами в 1993 г. для анализа покупательских корзин.
- Виды шаблонов: **частые наборы элементов** (itemsets), **последовательные шаблоны** (sequential patterns), **структурные шаблоны** (подграфы, деревья).
- Частые наборы — основа для **ассоциативных правил**, корреляционного анализа, классификации и кластеризации.

АНАЛИЗ РЫНОЧНОЙ КОРЗИНЫ

- **Анализ рыночной корзины** (market basket analysis): какие товары покупают вместе?
- Исходные данные — чеки покупок, каждый чек — транзакция.
- Результат — наборы товаров, встречающиеся в чеках достаточно часто.
- Пример: «Хлеб + Сыр» есть в большинстве чеков.

Области применения

• Ритейл: выкладка товаров, акции, рекомендации

• Интернет-магазины: «с этим товаром покупают»

• Веб: наборы посещаемых страниц

• Банки и телеком: наборы услуг

• Медицина: сочетания симптомов и диагнозов

• Биоинформатика: совместная активность генов

ОСНОВНЫЕ ПОНЯТИЯ

Понятие	Определение
Элемент (item)	Отдельный товар, событие или признак; $I = \{i_1, i_2, \ldots, i_n\}$ — множество всех элементов
Набор (itemset)	Подмножество элементов $X \subseteq I$; набор из k элементов называют k-набором
Транзакция T	Набор элементов одного события (чек, корзина), $T \subseteq I$
База транзакций D	Множество всех транзакций $D = \{T_1, T_2, \ldots, T_m\}$, $\lvert D \rvert = m$
Вхождение	Транзакция T содержит набор X, если $X \subseteq T$

ПРИМЕР БАЗЫ ТРАНЗАКЦИЙ

№	Товары в чеке
T1	Хлеб, Сыр
T2	Молоко, Масло, Яйца
T3	Хлеб, Масло, Сыр
T4	Хлеб, Молоко, Сыр
T5	Хлеб, Масло, Сыр
T6	Хлеб, Молоко, Сыр

Учебная база

I = {Хлеб, Молоко, Масло, Яйца, Сыр}

|I| = 5 элементов

|D| = 6 транзакций

Эти данные используем во всех дальнейших примерах.

ПОДДЕРЖКА НАБОРА (SUPPORT)

Абсолютная поддержка: $\sigma(X) = |\{ T \in D : X \subseteq T \}|$

Относительная поддержка: $supp(X) = \sigma(X) / |D|$

Примеры по учебной базе:

- $\sigma(\{\text{Хлеб}\}) = 5$, $supp = 5/6 \approx 83\ \%$
- $\sigma(\{\text{Хлеб, Сыр}\}) = 5$, $supp = 5/6 \approx 83\ \%$
- $\sigma(\{\text{Молоко, Масло}\}) = 1$, $supp = 1/6 \approx 17\ \%$

ЧАСТЫЙ НАБОР И МИНИМАЛЬНАЯ ПОДДЕРЖКА

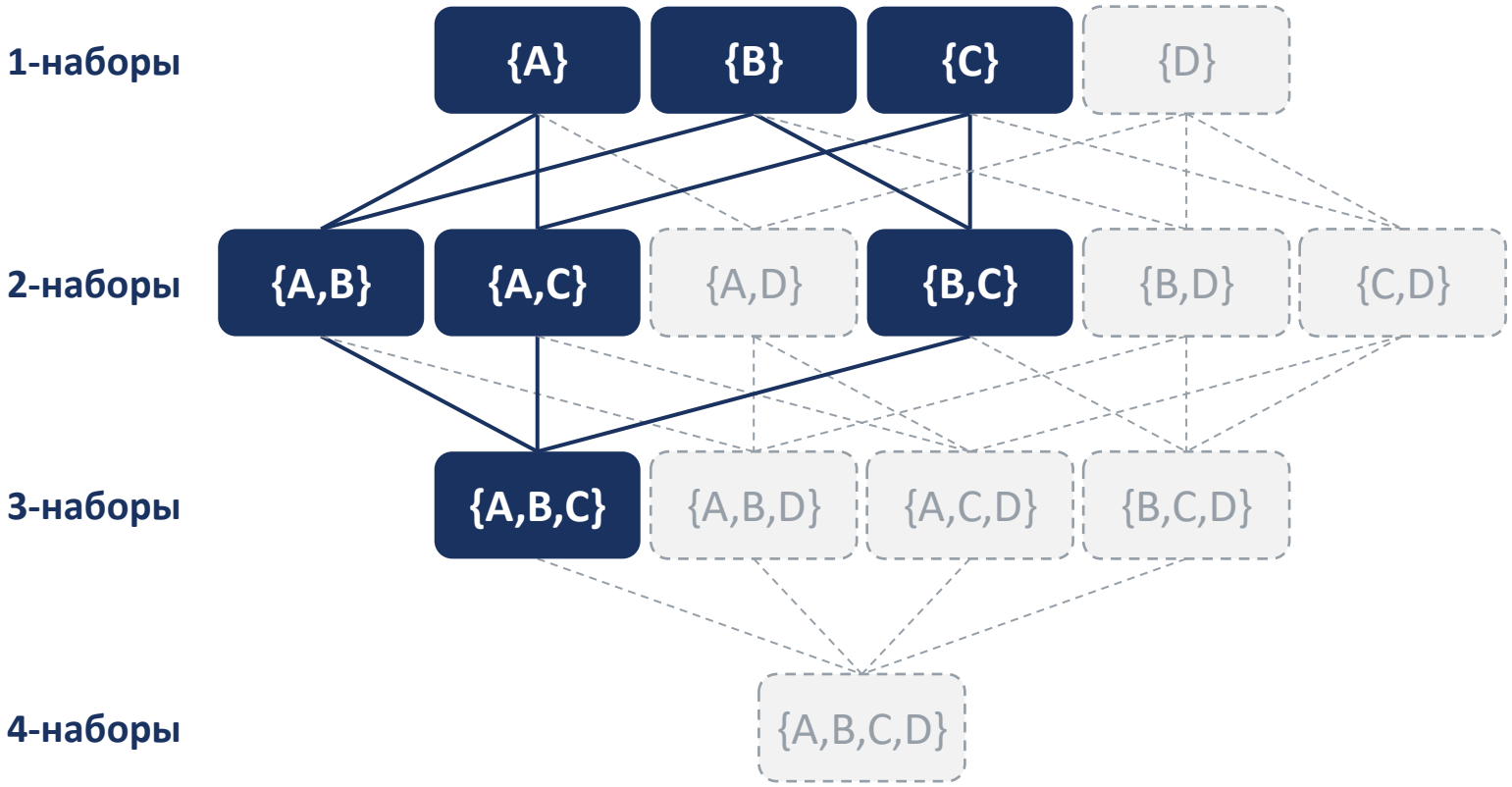
- Порог **min_support** задаёт пользователь — в долях или в числе транзакций.
- Набор X называется **частым**, если $supp(X) \geq min_support$.
- Задача: найти **все** частые наборы в базе D.
- В учебном примере $min_support = 2$ транзакции ($\approx 33\ \%$).
- Слишком низкий порог даёт «взрыв» числа наборов, слишком высокий — теряет редкие, но ценные закономерности.

ПРОСТРАНСТВО ПОИСКА

Число элементов n	Непустых наборов $2^n - 1$
5	31
10	1 023
20	1 048 575
30	1 073 741 823
100	$\approx 1{,}27 \cdot 10^{30}$

- Число возможных наборов растёт экспоненциально.
- В магазине n — тысячи и десятки тысяч товаров.
- Полный перебор невозможен: нужно отсекать бесперспективные наборы.

РЕШЁТКА НАБОРОВ И ОТСЕЧЕНИЕ



Решётка наборов

Допустим, элемент D — редкий.

Тогда все наборы, содержащие D (пунктир), тоже редкие и их поддержку считать не нужно: отсекаются 7 наборов из 15.

СВОЙСТВО АНТИМОНОТОННОСТИ

Свойство Apriori (антимонотонность поддержки)

Если $X \subseteq Z$, то $supp(X) \geq supp(Z)$.

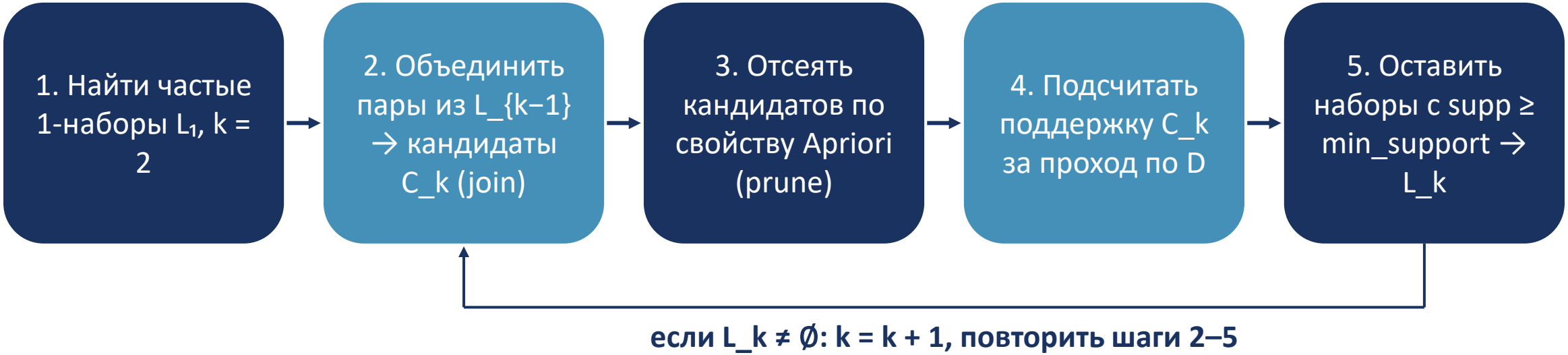
Любое подмножество частого набора — частое; любое надмножество редкого набора — редкое.

- Причина: каждая транзакция, содержащая Z, содержит и X.
- Следствие: k-кандидата отбрасывают без подсчёта, если хотя бы одно его $(k-1)$-подмножество редкое.
- Пример: {Молоко, Масло} редкий ($\sigma = 1$), поэтому {Хлеб, Молоко, Масло} не проверяется.

АЛГОРИТМ APRIORI: ИДЕЯ

- Предложен Р. Агравалом и Р. Шрикантом в 1994 г.
- **Поуровневый поиск** (level-wise): сначала частые 1-наборы, затем 2-наборы и т. д.
- Принцип **«сгенерируй и проверь»** (generate-and-test): кандидаты C_k строятся из частых наборов L_{k-1}.
- На каждом уровне — один проход по базе для подсчёта поддержки кандидатов.
- Остановка, когда новых частых наборов нет ($L_k = \emptyset$).

СХЕМА РАБОТЫ APRIORI



Если $L_k = \emptyset$ — конец. Результат: $L = L_1 \cup L_2 \cup \ldots \cup L_{k-1}$.

ГЕНЕРАЦИЯ КАНДИДАТОВ: JOIN И PRUNE

Шаг join (объединение)

Пары частых (k − 1)-наборов, отличающиеся одним элементом, объединяются в k-кандидата.

Пример:

{Хлеб, Молоко} + {Хлеб, Масло} → {Хлеб, Молоко, Масло}

Шаг prune (отсев)

Если хотя бы одно (k − 1)-подмножество кандидата не входит в L_{k-1}, кандидат удаляется без подсчёта.

Пример:

{Молоко, Масло} $\notin L_2$, значит {Хлеб, Молоко, Масло} отсеян.

ПРИМЕР, ШАГ 1: КАНДИДАТЫ C_1 И L_1

Кандидат C_1	σ	Результат (min_support = 2)
{Хлеб}	5	частый
{Молоко}	3	частый
{Масло}	3	частый
{Яйца}	1	редкий
{Сыр}	5	частый

Вывод

Один проход по базе: считаем поддержку каждого элемента.

{Яйца} ($\sigma = 1$) отбрасываем.

L_1 = {Хлеб, Молоко, Масло, Сыр}

ПРИМЕР, ШАГ 2: КАНДИДАТЫ C_2 И L_2

Кандидат C_2	σ	Результат
{Хлеб, Молоко}	2	частый
{Хлеб, Масло}	2	частый
{Хлеб, Сыр}	5	частый
{Молоко, Масло}	1	редкий
{Молоко, Сыр}	2	частый
{Масло, Сыр}	2	частый

Вывод

Кандидаты — все пары из L_1 (6 штук); {Яйца} в них не входит.

{Молоко, Масло} встречается лишь в T2 — редкий.

L_2 состоит из 5 наборов.

ПРИМЕР, ШАГ 3: КАНДИДАТЫ C_3 И L_3

Кандидат C_3	Все подмножества в L_2?	σ	Результат
{Хлеб, Молоко, Масло}	нет: {Молоко, Масло} ∉ L_2	—	**отсеян (prune)**
{Молоко, Масло, Сыр}	нет: {Молоко, Масло} ∉ L_2	—	**отсеян (prune)**
{Хлеб, Молоко, Сыр}	да	2	**частый**
{Хлеб, Масло, Сыр}	да	2	**частый**

- Поддержка считалась только для двух кандидатов из четырёх.
- L_3 = {{Хлеб, Молоко, Сыр}, {Хлеб, Масло, Сыр}}

ПРИМЕР, ШАГ 4: ОСТАНОВКА

- Объединяем наборы из L_3: {Хлеб, Молоко, Сыр} + {Хлеб, Масло, Сыр} → {Хлеб, Молоко, Масло, Сыр}.
- Проверка prune: подмножество {Хлеб, Молоко, Масло} ∉ L_3, значит кандидат отсеян.
- $C_4 = \emptyset$, следовательно $L_4 = \emptyset$ — **алгоритм останавливается**.
- Всего выполнено 3 прохода по базе (k = 1, 2, 3).

ИТОГ: ВСЕ ЧАСТЫЕ НАБОРЫ

k	Частые k-наборы L_k (набор — σ)	Число
1	{Хлеб} — 5; {Сыр} — 5; {Молоко} — 3; {Масло} — 3	4
2	{Хлеб, Сыр} — 5; {Хлеб, Молоко} — 2; {Хлеб, Масло} — 2; {Молоко, Сыр} — 2; {Масло, Сыр} — 2	5
3	{Хлеб, Молоко, Сыр} — 2; {Хлеб, Масло, Сыр} — 2	2

Всего **11** частых наборов. Поддержка считалась для 5 + 6 + 2 = **13** кандидатов вместо 31 возможного набора.

ПСЕВДОКОД АЛГОРИТМА APRIORI

```
Вход: база D, порог min_support
Выход: множество всех частых наборов L

L1 = {частые 1-наборы}
k = 2
пока L(k-1) не пусто:
    Ck = join(L(k-1))                  # кандидаты
    Ck = prune(Ck, L(k-1))             # отсев
    для каждой транзакции T из D:
        для каждого c из Ck, если c входит в T:
            count[c] += 1
    Lk = {c из Ck : count[c] >= min_support}
    k = k + 1
вернуть L1 + L2 + ... + L(k-1)
```

СЛОЖНОСТЬ И ОПТИМИЗАЦИИ

- Число проходов по базе равно длине наибольшего частого набора.
- Основные затраты — подсчёт поддержки кандидатов; $|C_2| = |L_1|(|L_1| - 1)/2$, при 1000 частых товаров $\approx$ 500 тыс. пар.
- Оптимизации: **хеширование (DHP)**, **сокращение транзакций**, **разбиение (Partition)**, **выборка (Sampling)**, **динамический подсчёт (DIC)**.

ОГРАНИЧЕНИЯ APRIORI И FP-GROWTH

Критерий	Apriori	FP-Growth
Подход	Генерация кандидатов и проверка	Сжатое FP-дерево, без кандидатов
Проходы по базе	До k_max (по числу уровней)	Два прохода
Память	Хранение множества кандидатов	Хранение FP-дерева
Низкий порог поддержки	Резкий рост числа кандидатов	Устойчивее, обычно быстрее
Реализация	Простая, наглядная	Сложнее; предложен в 2000 г. (Хан, Пей, Инь)

РЕАЛИЗАЦИЯ НА PYTHON (MLXTEND)

```
import pandas as pd
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori

transactions = [
    ["Хлеб", "Сыр"],
    ["Молоко", "Масло", "Яйца"],
    ["Хлеб", "Масло", "Сыр"],
    ["Хлеб", "Молоко", "Сыр"],
    ["Хлеб", "Масло", "Сыр"],
    ["Хлеб", "Молоко", "Сыр"],
]
te = TransactionEncoder()
df = pd.DataFrame(te.fit(transactions).transform(transactions),
                  columns=te.columns_)
freq = apriori(df, min_support=2/6, use_colnames=True)
print(freq)
```

Результат

Найдено 11 частых наборов — как и при ручном расчёте.

Параметр min_support — доля транзакций (2/6 ≈ 0,33).

Столбец support содержит относительную поддержку.

ОТ ЧАСТЫХ НАБОРОВ К ПРАВИЛАМ

Правило $X \Rightarrow Y$ ($X \cap Y = \emptyset$)

$\text{support} = \text{supp}(X \cup Y)$

$\text{confidence} = \text{supp}(X \cup Y) / \text{supp}(X)$

$\text{lift} = \text{confidence} / \text{supp}(Y)$

- Из каждого частого набора строятся правила, прошедшие порог confidence.
- Пример: {Молоко} ⇒ {Сыр}: support = 2/6 ≈ 33 %, confidence = 2/3 ≈ 67 %.
- lift = 0,67 / 0,83 ≈ 0,8 < 1: молоко не повышает вероятность покупки сыра.

КОНТРОЛЬНЫЕ ВОПРОСЫ

- Что такое частый набор и чем он отличается от ассоциативного правила?
- Как определяются абсолютная и относительная поддержка?
- Сколько непустых наборов можно составить из n элементов?
- Сформулируйте свойство антимонотонности и покажите его применение в Apriori.
- Из каких шагов состоит итерация Apriori?
- Почему Apriori может работать медленно и чем его заменяют?

ВЫВОДЫ

- Частые наборы — основа поиска ассоциативных закономерностей в транзакционных данных.
- Частота набора измеряется поддержкой и сравнивается с порогом min_support.
- Свойство антимонотонности позволяет отсекать кандидатов, не считая их поддержку.
- Apriori ищет наборы по уровням: join → prune → подсчёт → фильтр по порогу.
- Ограничения Apriori — многократные проходы по базе и большое число кандидатов; альтернатива — FP-Growth.

СПИСОК ЛИТЕРАТУРЫ

- Agrawal R., Srikant R. Fast Algorithms for Mining Association Rules // Proc. 20th VLDB, 1994. P. 487–499.
- Agrawal R., Imielinski T., Swami A. Mining Association Rules between Sets of Items in Large Databases // Proc. ACM SIGMOD, 1993. P. 207–216.
- Han J., Kamber M., Pei J. Data Mining: Concepts and Techniques. 3rd ed. Morgan Kaufmann, 2011.
- Han J., Pei J., Yin Y. Mining Frequent Patterns without Candidate Generation // Proc. ACM SIGMOD, 2000. P. 1–12.
- Tan P.-N., Steinbach M., Kumar V. Introduction to Data Mining. Pearson.
- Zaki M., Meira W. Data Mining and Machine Learning: Fundamental Concepts and Algorithms. Cambridge University Press, 2020.
- Документация библиотеки mlxtend: https://rasbt.github.io/mlxtend/