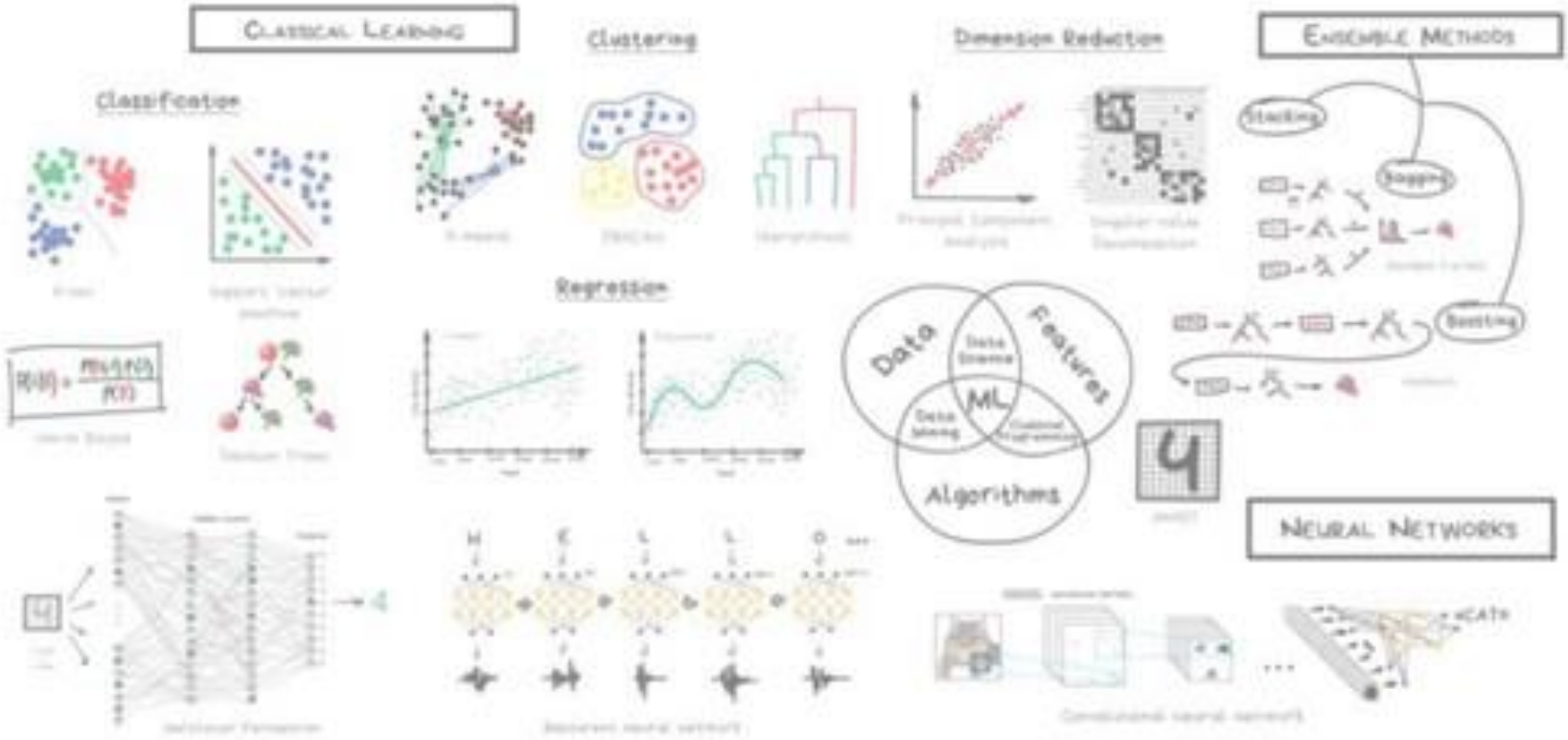
МАШИНЕ НАВЧАННЯ

Навчання без вчителя. Асоціації та пошук правил



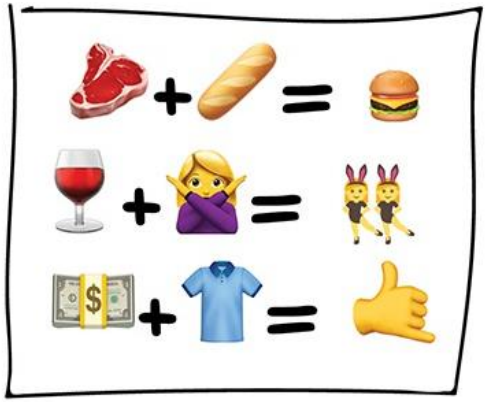
Лекція №12

Асоціація

- **Асоціація** шукає закономірності у потоці замовлень
- **Навчання асоціативним правилам** або **пошук асоціативних правил** — це метод навчання машин на базі правил виявлення зв'язків, що цікавлять нас, між змінними у великій базі даних.
- Метод пропонується для встановлення сильних правил, виявлених у базі даних за допомогою деяких заходів цікавості. Цей заснований на правилах підхід також генерує нові правила в міру аналізу додаткових даних. Кінцевою метою, виходячи з досить великого набору даних, допомогти машині імітувати виділення людських ознак і створити можливість знаходження абстрактних асоціацій з нових некласифікованих даних.

Сьогодні використовують для:

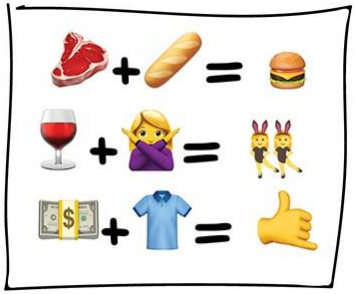
- Прогноз акцій та розпродажів
- Аналіз товарів, що купуються разом
- Розміщення товарів на полицях
- Аналіз патернів поведінки на веб-сайтах



Популярні алгоритми: Apriori, FP-growth

Асоціація

Постановка задачі



Задача пошуку асоціативного правила визначається так:

Нехай X — множина об'єктів (множина операцій або транзакцій

$X^\ell = \{x_1, \dots x_\ell\} \subset X$ — навчальна вибірка

$\mathcal{F} = \{f_1, \dots, f_n\}$, $f\colon\ X \to \{0,1\}$ — множина двійкових ознак

Правило визначається як імплікація форми:

$A \Rightarrow B$, де $A, B \subseteq \mathcal{F}$

Приклад бази даних з 5 продажами та 5 предметами

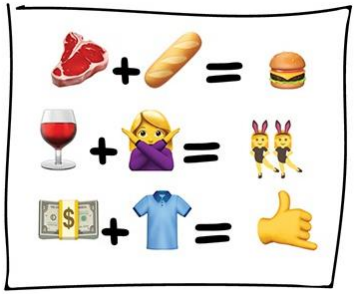
ID операції	молоко	хліб	масло	пиво	підгузники
1	1	1	0	0	0
2	0	0	1	0	0
3	0	0	0	1	1
4	1	1	1	0	0
5	0	1	0	0	0

Прикладом правила для супермаркета може слугувати правило: {масло,хліб}=>{молоко} яке означає, що, якщо покупець придбав масло та хліб, то він також придбає молоко.

Кожна *транзакція* має унікальний ID (номер) транзакції та складається з підмножини об'єктів.

Асоціація

Постановка задачі



Задача пошуку асоціативного правила визначається так:

Нехай X — множина об'єктів (множина операцій або транзакцій

$X^\ell = \{x_1, \dots x_\ell\} \subset X$ — навчальна вибірка

$\mathcal{F} = \{f_1, \dots, f_n\}$, $f\colon\ X \to \{0,1\}$ — множина двійкових ознак

Правило визначається як імплікація форми:

$A \Rightarrow B$, де $A, B \subseteq \mathcal{F}$

Кожній підмножині $\varphi \subseteq \mathcal{F}$ відповідає кон'юнкція*

$$\varphi(x) = \bigwedge_{f \in \varphi} f(x), \qquad x \in X$$

Якщо $\varphi(x) = 1$, то ознаки з φ сумісно зустрічаються і у x.

Приклад бази даних з 5 продажами та 5 предметами

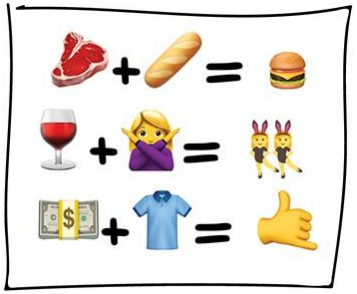
ID операції	молоко	хліб	масло	пиво	підгузники
1	1	1	0	0	0
2	0	0	1	0	0
3	0	0	0	1	1
4	1	1	1	0	0
5	0	1	0	0	0

φ(молоко ∧ хліб)
1
0
0
1
0

* Кон'юнкція (лат. conjangere — об'єднувати) (операція **AND**) — двомісна логічна операція, що має значення «істина», якщо всі операнди мають значення «істина».

Асоціація

Основні визначення



Підтримка (*Support*) вказує наскільки часто набір предметів з'являється у наборі даних. Підтримка набору A відносно B визначається як частка транзакцій, що містять A і B у наборі даних, від загального набору даних: Частота зустрічі (підтримка, **support**) φ у вибірці X^{ℓ} визначається як:

$$\nu(A \cup B) = \frac{\text{кількість транзакцій що містять } A \text{ і } B}{\text{загальна кількість транзакцій}}$$

$$\nu(\varphi) = \frac{1}{\ell}\sum^{\ell} \varphi(x_i)$$

Приклад бази даних з 5 продажами та 5 предметами

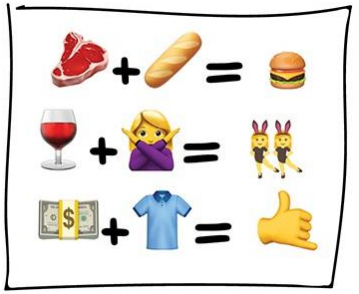
ID операції	молоко	хліб	масло	пиво	підгузники
1	1	1	0	0	0
2	0	0	1	0	0
3	0	0	0	1	1
4	1	1	1	0	0
5	0	1	0	0	0

ν(молоко ∩ хліб)
2/5 = 0.4 (40%)
ν(пиво ∩ підгузки)
1/5 = 0.2 (20%)
ν(масло ∩ хліб)
1/5 = 0.2 (20%)

Якщо $\nu(\varphi) \geq \delta$, то набір φ вважається «частим» (**frequent itemset**)
Параметр δ – мінімальна підтримка (**minimal support – MinSup**)

Асоціація

Основні визначення



Значущість (*Confidence*) вказує на те, як часто виконується правило. Значення *значущості* у правилі $A \Rightarrow B$ відносно множини транзакцій ℓ є часткою транзакцій, які містять набір A, який також містить набір B. Впевненість визначається так:

$$\nu(A|B) = \frac{\text{кількість транзакцій що містять } A \text{ і } B}{\text{кількість транзакцій що містять } A}$$

$$\nu(\varphi|y) = \frac{\nu(\varphi \cup y)}{\nu(\varphi)}$$

Приклад бази даних з 5 продажами та 5 предметами

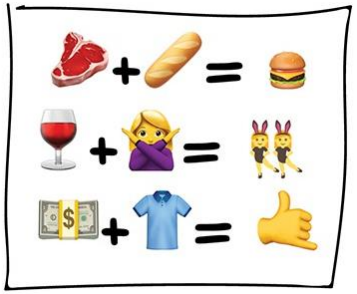
ID операції	молоко	хліб	масло	пиво	підгузники
1	1	1	0	0	0
2	0	0	1	0	0
3	0	0	0	1	1
4	1	1	1	0	0
5	0	1	0	0	0

Наприклад, правило $\{\text{масло}, \text{хліб}\} \Rightarrow \{\text{молоко}\}$ має значущість 0.2/0.2 = 1, що означає, що у 100 % випадків споживач, який купив масло та хліб, так само купив і молоко. Якщо куплений $\{\text{хліб}\}$ то з імовірністю 67% буде куплене й $\{\text{молоко}\}$, хоча обидва товари купляються з ймовірністю 40%.

Якщо $\nu(\varphi|y) \geq \kappa$, то набір φ вважається «значущим»

Параметр κ – мінімальна значущість (**minimal confidence – MinConf**)

Асоціація
Побудова правил



Оскільки $\varphi(x) = \Lambda_{f\in\varphi} f(x), x \in X$ – кон'юнкція, то має місце **властивість антимонотонності:**

для будь-яких $\psi, \varphi \subset \mathcal{F}$ за умови $\varphi \subset \psi$ отримуємо $\nu(\varphi) \geq \nu(\psi)$

Наслідки:

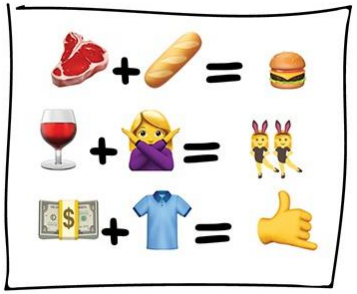
- якщо набір ψ часто зустрічається, то і всі набори φ, що є його підмножинами, $\varphi \subset \psi$ також часто зустрічаються
- якщо φ зустрічається рідко, то й всі набори ψ, які його включають, $\psi \supset \varphi$ також зустрічаються рідко
- для будь-яких ψ, φ має місце співвідношення: $\nu(\varphi \cup \psi) \leq \nu(\varphi)$

Два етапи пошуку асоціативних правил:

1. Пошук частих наборів
   (багатократний перегляд транзакційної бази даних)
2. Виділення асоціативних правил
   (проста ефективна процедура в оперативній пам'яті)

Асоціація

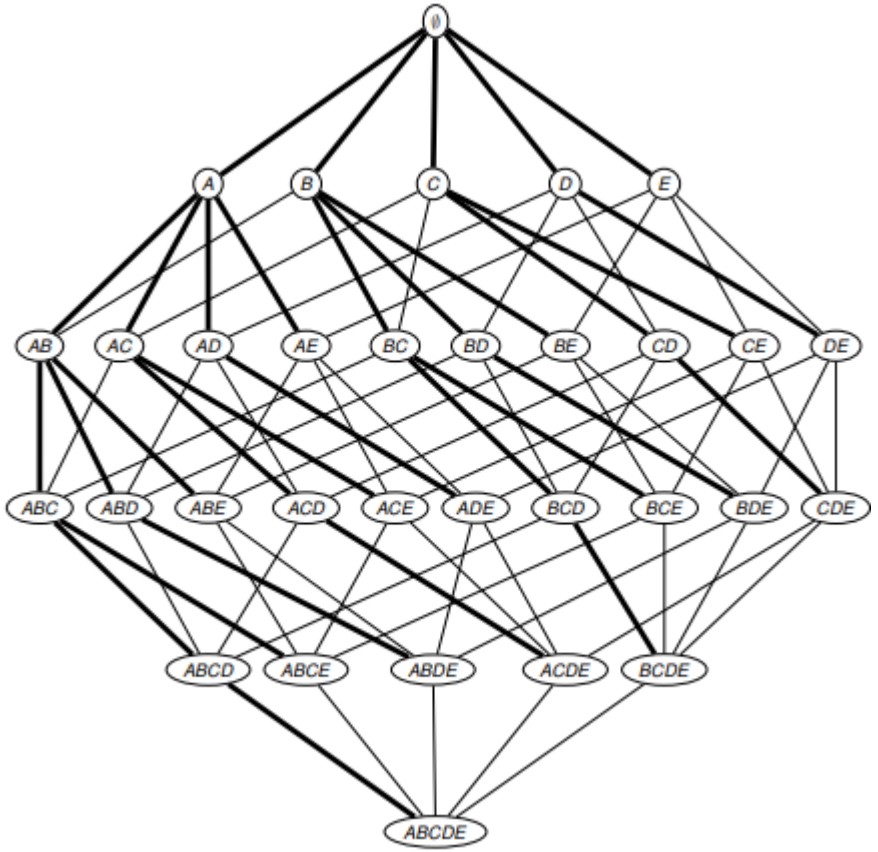
Алгоритм APriory – пошук в ширину



Природа Apriori:
Всі непусті підмножини частих наборів елементів також мають бути частими.
Властивість Apriori засновано на такому спостереженні:

- Якщо набір елементів X не відповідає мінімальному порогу підтримки δ, то X зустрічається не часто, тобто $\mathrm{P}(X) < \delta$.
- Якщо елемент A добавлено до X, результуючий набір елементів (тобто $X \cup A$) не може з'являтися частіше, ніж X:

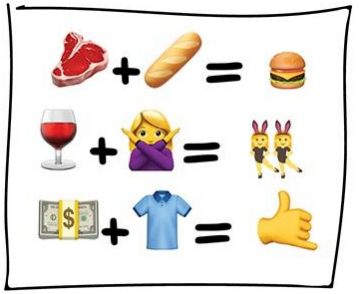
$$if\ X \subseteq Y, \nu(X) \geq \nu(Y)$$



Апріорна теорема: Якщо набір елементів є частим, то всі його підмножини також мають бути частими.

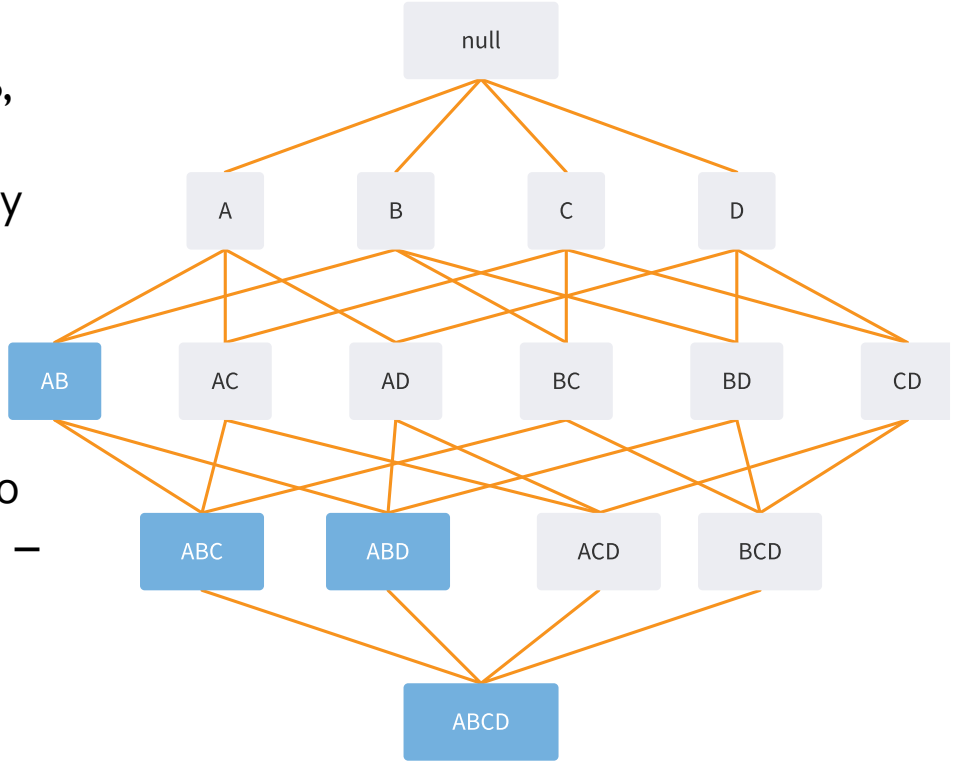
Асоціація

Алгоритм APriory – пошук в ширину



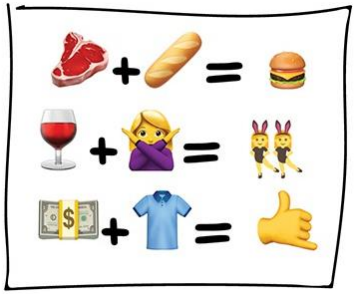
Всі можливі набори елементів можна подати у вигляді гратки, що починається з порожньої множини, потім на 1 рівні – 1-елементні набори, на 2-му – 2-елементні і т.д. На рівні k представлені k-елементні набори, пов'язані з усіма своїми (k−1)-елементними підмножинами.

Розглянемо Малюнок, що ілюструє набір елементів A, B, C, D. Припустимо, що набір з елементів {A,B} має підтримку нижче заданого порога і, відповідно, не є частим. Тоді, згідно з властивістю анти-монотонності, всі його супермножини також не часто зустрічаються і відкидаються – вся ця гілка, починаючи з {A, B}, виділено синім. Використання цієї евристики дозволяє значно скоротити простір пошуку.



Асоціація

Алгоритм APriory – пошук в ширину



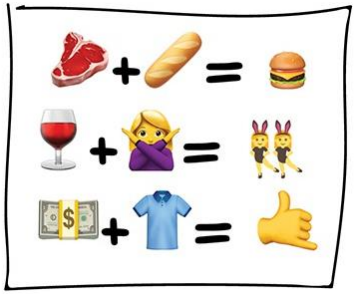
Алгоритм Апріорі - це послідовність кроків, яких слід дотримуватися, щоб знайти найчастіший набір елементів у даній базі даних. Ця техніка видобутку даних послідовно повторює кроки з'єднання та обрізки, поки не буде досягнуто найчастіший набір елементів. У проблемі вказано мінімальний поріг підтримки або він передбачається користувачем.

1.Знайдіть усі найпоширеніші набори предметів.
2.Створіть правила асоціації з вищезазначених частих наборів предметів.

- На першому кроці алгоритму підраховуються 1-елементні набори, що часто зустрічаються. Для цього необхідно пройтися по всьому наборі даних, і підрахувати їм підтримку, тобто скільки разів зустрічається у базі.
- Наступні кроки будуть складатися з двох частин: генерації потенційно найпоширеніших наборів елементів (їх називають кандидатами) та підрахунку підтримки для кандидатів.

Асоціація

Алгоритм APriory – пошук в ширину



Кроки в Apriory

 # 1) На першій ітерації алгоритму кожен предмет приймається як кандидат набору з одного предмета. Алгоритм буде рахувати випадки появи кожного елемента.

 # 2) Нехай буде якась мінімальна підтримка, min_sup. Визначається набір 1-елементних наборів предметів, поява яких задовольняє мінімальну суму. Тільки тих кандидатів, для яких підтримка більша або дорівнює min_sup, приймають на наступну ітерацію, а інших обрізають.

 # 3) Далі виявляються часті набори предметів. Для цього на етапі об'єднання набір із двох елементів формується шляхом формування групи з 2 шляхом комбінування елементів між собою.

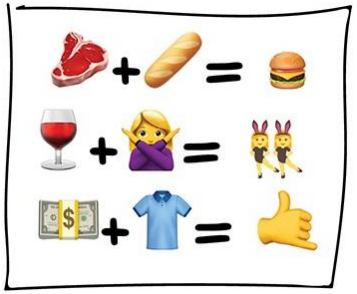
 # 4) Кандидати з 2 елементів обрізаються з використанням порогового значення min-sup. Тепер у таблиці будуть дво-елементні набори з мінімальною сумою.

 # 5) Наступна ітерація сформує три-елементні набори, використовуючи кроки об'єднання та обрізки. Ця ітерація буде слідувати властивості анти-монотонності. Якщо всі підмножини з 2 елементів є частими, тоді надмножина буде частою, інакше вона буде обрізана.

 # 6) Наступним кроком буде створення набору 4-елементів, з'єднавши 3-набір елементів із собою та обрізання, якщо його підмножина не відповідає критеріям min_sup. Алгоритм зупиняється, коли досягається найчастіший набір елементів.

Асоціація

Алгоритм APriory – пошук в ширину



Приклад Apriory: поріг підтримки = 50%, впевненість = 60%

Транзакція	Перелік предметів
T1	I1, I2, I3
T2	I2, I3, I4
T3	I4, I5
T4	I1, I2, I4
T5	I1, I2, I3, I5
T6	I1, I2, I3, I4

Елемент	Підтримка
I1	4
I2	5
I3	4
I4	4
I5	2

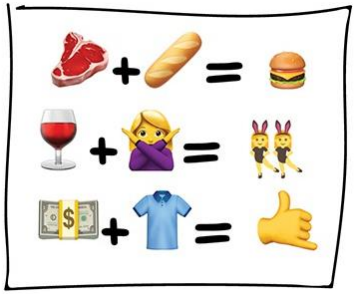
Елемент I5 не відповідає min_sup = 3
Він видаляється.
Лише I1, I2, I3, I4 задовольняє min_sup

Поріг підтримки = 50%
$\Rightarrow$ 0,5 * 6 = 3
$\Rightarrow$ min_sup = 3

Елемент	Підтримка
I1	4
I2	5
I3	4
I4	4

Асоціація

Алгоритм APriory – пошук в ширину



Приклад Apriory: поріг підтримки = 50%, впевненість = 60%

Транзакція	Перелік предметів
T1	I1, I2, I3
T2	I2, I3, I4
T3	I4, I5
T4	I1, I2, I4
T5	I1, I2, I3, I5
T6	I1, I2, I3, I4

Набір	Підтримка
I1, I2	4
I1, I3	3
I1, I4	2
I2, I3	4
I2, I4	3
I3, I4	2

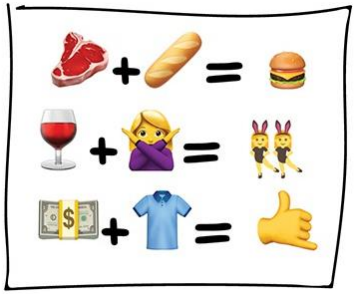
Поріг підтримки = 50%
⇒ 0,5 * 6 = 3
⇒ min_sup = 3

Набір	Підтримка
I1, I2	4
I1, I3	3
I2, I3	4
I2, I4	3

набір елементів {I1, I4} та {I3, I4} не відповідає min_sup, тому вони видаляються.

Асоціація

Алгоритм APriory – пошук в ширину



Приклад Apriory: поріг підтримки = 50%, впевненість = 60%

Транзакція	Перелік предметів
T1	I1, I2, I3
T2	I2, I3, I4
T3	I4, I5
T4	I1, I2, I4
T5	I1, I2, I3, I5
T6	I1, I2, I3, I4

Набір	Підтримка
I1, I2	4
I1, I3	3
I2, I3	4
I2, I4	3

Поріг підтримки = 50%

$\Rightarrow$ 0,5 * 6 = 3

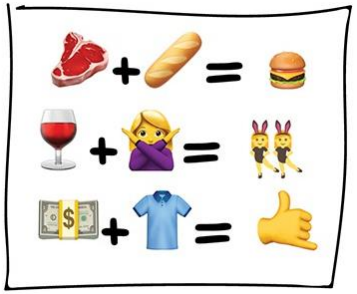
$\Rightarrow$ min_sup = 3

Набір	Підтримка
I1, I2, I3	3
I1, I2, I4	1
I1, I3, I4	1
I2, I3, I4	2

Лише набір елементів {I1, I2, I3} відповідає min_sup. Він є частим.

Асоціація

Алгоритм APriory – пошук в ширину



Приклад Apriory: поріг підтримки = 50%, впевненість = 60%

Транзакція	Перелік предметів
T1	I1, I2, I3
T2	I2, I3, I4
T3	I4, I5
T4	I1, I2, I4
T5	I1, I2, I3, I5
T6	I1, I2, I3, I4

Елемент	Підтримка
I1	4
I2	5
I3	4
I4	4

Набір	Підтримка
I1, I2	4
I1, I3	3
I2, I3	4
I2, I4	3

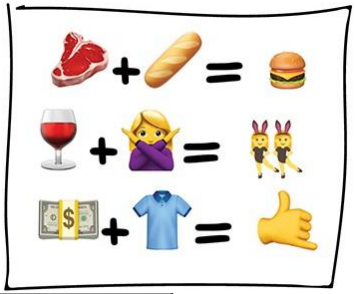
Набір	Підтримка
I1, I2, I3	3

Сформуємо правила асоціації: для трьох частих наборів предметів, виявлених вище:

- {I1, I2} => {I3}
- {I1, I3} => {I2}
- {I2, I3} => {I1}
- {I1} => {I2, I3}
- {I2} => {I1, I3}
- {I3} => {I1, I2}

Асоціація

Алгоритм APriory – пошук в ширину



Елемент	Підтримка
I1	4
I2	5
I3	4
I4	4

Набір	Підтримка
I1, I2	4
I1, I3	3
I2, I3	4
I2, I4	3

Набір	Підтримка
I1, I2, I3	3

впевненість = 60%

Розрахуємо впевненість для шести правил:

- {I1, I2} => {I3}

Впевненість = підтримка {I1, I2, I3} / підтримка {I1, I2} = (3/4) * 100 = 75%

- {I1, I3} => {I2}

Впевненість = підтримка {I1, I2, I3} / підтримка {I1, I3} = (3/3) * 100 = 100%

- {I2, I3} => {I1}

Впевненість = підтримка {I1, I2, I3} / підтримка {I2, I3} = (3/4) * 100 = 75%

- {I1} => {I2, I3}

Впевненість = підтримка {I1, I2, I3} / підтримка {I1} = (3/4) * 100 = 75%

- {I2} => {I1, I3}

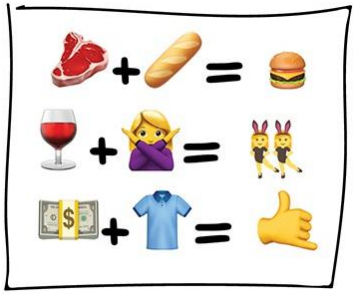
Впевненість = підтримка {I1, I2, I3} / підтримка {I2} = (3/5) * 100 = 60%

- {I3} => {I1, I2}

Впевненість = підтримка {I1, I2, I3} / підтримка {I3} = (3/4) * 100 = 75%

Асоціація

Алгоритм APriory – пошук в ширину



Вхід: X^ℓ – навчальна вибірка;
δ = MinSup – мінімальна підтримка;
κ = MinConf – мінімальна впевненість

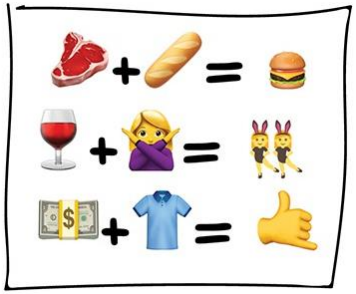
Вихід: $R = \{(\varphi, y)\}$ – список асоціативних правил

Етап 1: пошук частих наборів

Генеруємо множину всіх частих вихідних ознак:
$G_1 := \{f \in \mathcal{F} | \nu(f) \geq \delta\}$;
для всіх $j = 2, \ldots, n$
    множина всіх частих наборів потужності j:
    $G_j := \{\varphi \cup \{f\} | \varphi \in G_{j-1},\ f \in G_1 \backslash \varphi,\ \nu(\varphi \cup \{f\}) \geq \delta\}$;
    якщо $G_j = \emptyset$ то
        вихід з циклу по j;

Асоціація

Алгоритм APriory – пошук в ширину



Етап 2: Рекурсивний алгоритм створення списку асоціативних правил

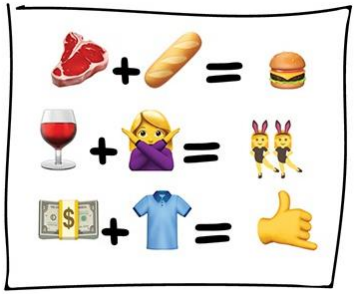
$R := \emptyset$;
для всіх $\psi \in G_j$, $j = 2, \dots, n$
    AssocRules$(R, \psi, \emptyset)$;

функція AssocRules(R, φ, y)
    вхід: набір (φ, y)
    вихід: список асоціативних правил R;
    для всіх $f \in \varphi$: $id_f > \max\limits_{g \in y} id_g$ (щоб уникнути повторів y)
        $\varphi' := \varphi \backslash \{f\}$; $y' := y \cup \{f\}$;
        якщо $\nu(y'|\varphi') \geq \kappa$ то
            додати асоціативне правило (φ', y') в список R;
            якщо $|\varphi'| > 1$ то
                AssocRules(R, φ', y');

id_f – порядковий номер ознаки f у $\mathcal{F} = \{f_1, \dots, f_n\}$

Асоціація

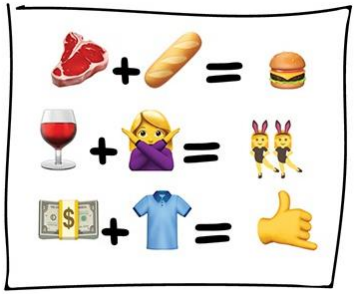
Алгоритм APriory – пошук в ширину



Оптимізація

Набагато швидше та ефективно використовувати підхід, заснований на зберіганні кандидатів у хеш-дереві. Внутрішні вузли дерева містять хеш-таблиці із вказівниками на нащадків, а листя – на кандидатів. Це дерево стане у нагоді для швидкого підрахунку підтримки для кандидатів. Хеш-дерево будується щоразу, коли формуються кандидати. Спочатку дерево складається тільки з кореня, яке є листом, і не містить жодних кандидатів-наборів. Щоразу коли формується новий кандидат, він заноситься в корінь дерева, і так доти, доки кількість кандидатів у корені-листі не перевищить якогось порога. Щойно кількість кандидатів стає більше порога, корінь перетворюється на хеш-таблицю, тобто. стає внутрішнім вузлом, і йому створюються нащадки-листя. І всі приклади розподіляються по вузлах-нащадках згідно з хеш-значення елементів, що входять в набір, і т.д. Кожен новий кандидат хешується на внутрішніх вузлах, поки він не досягне першого вузла-листа, де він і зберігатиметься, поки кількість наборів знову ж таки не перевищить порога.

Асоціація



Алгоритм APriory – пошук в ширину

Оптимізація

Використовуючи хеш-дерево, легко підрахувати підтримку для кожного кандидата. Для цього потрібно «пропустити» кожну транзакцію через дерево і збільшити лічильники тих кандидатів, чиї елементи також містяться у транзакції. На кореневому рівні хеш-функція застосовується до кожного елемента транзакції.

Далі, на другому рівні, хеш-функція застосовується до інших елементів тощо. На k-рівні хешується k-елемент. І так доти, доки не досягнемо листа. Якщо кандидат, що зберігається в листі, є підмножиною транзакції, що розглядається, тоді збільшуємо лічильник підтримки цього кандидата на одиницю.

Після того, як кожна транзакція з вихідного набору даних «пропущена» через дерево, можна перевірити, чи задовольняють значення підтримки кандидатів мінімальному порогу. Кандидати, котрим ця умова виконується, переносяться до розряду ти що часто зустрічаються. Крім того, слід запам'ятати і підтримку набору, вона стане нам у нагоді при вилученні правил. Ці дії застосовуються для знаходження (k+1) - елементних наборів і т.д.

Після того як знайдені всі набори елементів, що часто зустрічаються, можна приступити безпосередньо до генерації правил.

Дякую за увагу