Trường Đại học Khoa học Tự nhiên
Khoa Công nghệ Thông tin

TÀI LIỆU LÝ THUYẾT KTDL & UD

KHAI THÁC MẪU PHỔ BIẾN và LUẬT KẾT HỢP (P1)

Giảng viên: ThS. Lê Ngọc Thành
Email: lnthanh@fit.hcmus.edu.vn

Winter 2012

Nội dung

- Dữ liệu giao dịch
- Khái niệm cơ bản về:
  - Mẫu phổ biến
  - Luật kết hợp
- Khai thác luật kết hợp
- Thuật toán Apriori
  - Phát sinh tập hạng mục phổ biến
  - Xây dựng luật kết hợp từ tập phổ biến

Dữ liệu giao dịch (1/5)

- $I = \{i_1, i_2, \ldots, i_m\}$: là tập hợp các hạng mục
- Giao dịch t: là tập hợp các hạng mục sao cho $t \subseteq I$.
- Cơ sở dữ liệu giao dịch T: tập hợp các giao dịch $T = \{t_1, t_2, \ldots, t_n\}$.

CSDL *T*

TID	Transaction
10	{Bread, Cheese, Juice}
20	{Milk, Bread, Yogurt}
30	{Bread, Juice, Milk}
40	{Eggs, Bread, Cheese, Juice}
50	{Cheese, Juice, Milk}

I = {Bread, Cheese, Eggs, Milk, Juice, Yogurt}

t_{10} = {Bread, Cheese, Juice} $\subset$ I

Dữ liệu giao dịch (2/5)

- Dữ liệu siêu thị
  - Các giao dịch giỏ mua hàng
    - t_1: {bread, cheese, milk}
    - t_2: {apple, eggs, salt, yogurt}

    … …
    - t_n: {biscuit, eggs, milk}
  - Khái niệm
    - Hạng mục: một món hàng/mục trong giỏ hàng
    - *I*: tập hợp các mặt hàng có bán trong siêu thị
    - Giao dịch t_i: những món hàng trong một giỏ hàng, được gán mã giao dịch TID (transaction ID)
    - Cơ sở dữ liệu giao dịch: là tập hợp các giao dịch

Dữ liệu giao dịch (3/5)

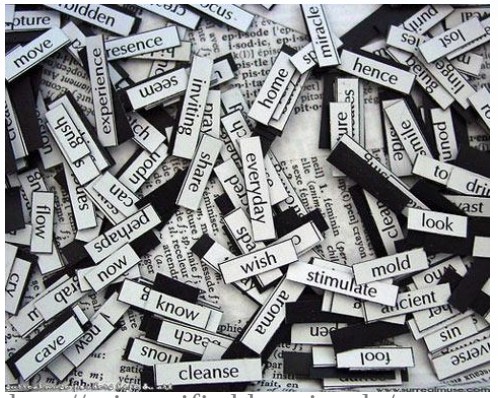
- Dữ liệu các tài liệu văn bản: mỗi văn bản chứa một số từ khóa
  - doc1: Student, Teach, School
  - doc2: Student, School
  - doc3: Teach, School, City, Game
  - doc4: Baseball, Basketball
  - doc5: Basketball, Player, Spectator
  - doc6: Baseball, Coach, Game, Team
  - doc7: Basketball, Team, City, Game

Dữ liệu giao dịch (4/5)

- Biểu diễn dữ liệu siêu thị ở dạng giao dịch là hình thức đơn giản của các giỏ hàng.
- Một số thông tin quan trọng không được xét đến. Ví dụ:
  - Số lượng của mỗi món hàng được mua
  - Giá của món hàng

Dữ liệu giao dịch (5/5)

- Tài liệu văn bản hay câu trong văn bản có thể xem là giao dịch.
  - Không xét đến thứ tự từ và số lần từ xuất hiện.
  - Các từ trùng bị loại bỏ.
- Cho trước một tập hợp văn bản hay câu, ta có thể tìm quan hệ đồng xuất hiện của từ.



http://asiapacific.blogs.ie.edu/

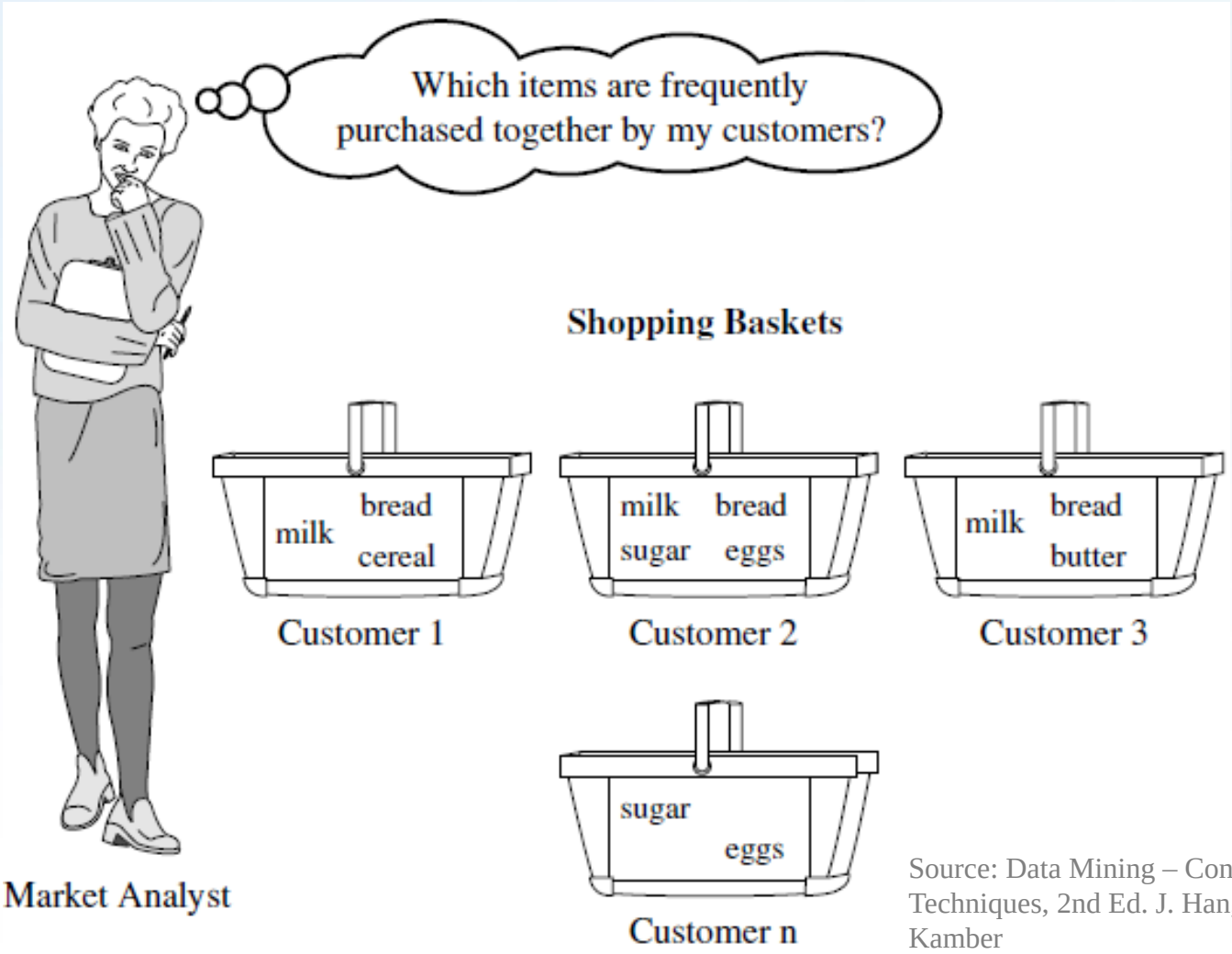
Nội dung

- Dữ liệu giao dịch
- Khái niệm cơ bản về:
  - Mẫu phổ biến
  - Luật kết hợp
- Khai thác luật kết hợp
- Thuật toán Apriori
  - Phát sinh tập hạng mục phổ biến
  - Xây dựng luật kết hợp từ tập phổ biến

Mẫu phổ biến

- Mẫu phổ biến: là mẫu (tập các hạng mục, chuỗi con, cấu trúc con, đồ thị con, …) xuất hiện thường xuyên trong tập DL
- Mục tiêu: tìm mọi mối quan hệ đồng xuất hiện (kết hợp) giữa các hạng mục dữ liệu.

Ví dụ mẫu phổ biến



Source: Data Mining – Concepts and Techniques, 2nd Ed. J. Han, M. Kamber

Tính chứa và tính phủ

- Giao dịch $t_i \in T$ được gọi là chứa (contain) tập hạng mục X nếu X là tập con của t_i. Nói cách khác, tập hạng mục X phủ (cover) t_i.
  - Ví dụ: giao dịch t = {Bread, Cheese, Juice} và X = {Cheese, Juice}

Đếm hỗ trợ

- Đếm hỗ trợ (support count) của X trong *T*, ký hiệu *X.count*, là số giao dịch trong *T* chứa *X*.
  - Ví dụ:
    - {Bread}.count = 4
    - {Milk, Eggs}.count = 0

CSDL *T*

TID	Transaction
10	{Bread, Cheese, Juice}
20	{Milk, Bread, Yogurt}
30	{Bread, Juice, Milk}
40	{Eggs, Bread, Cheese, Juice}
50	{Cheese, Juice, Milk}

Độ phổ biến

- **Độ phổ biến** (supp) của tập hạng mục X trong T là tỷ lệ giữa *số các giao dịch chứa X* trên *tổng số các giao dịch trong T*

  Supp(X) = count(X) / | T |

- **Tập phổ biến** (frequent itemsets) là tập có *độ phổ biến thỏa mãn độ phổ biến tối thiểu minsupp* (do người dùng xác định)

  Nếu supp(X) ≥ minsupp thì X - tập phổ biến

  .

Tính chất mẫu phổ biến

- Tất cả các tập con của mẫu phổ biến đều là mẫu phổ biến
- Thảo luận:
  - Tại sao? Chứng minh.
  - Nếu tập con không phổ biến thì tập bao nó (tập cha) có phổ biến hay không ?

Ví dụ mẫu phổ biến

Transaction	Items
t_1	Bread,Jelly,PeanutButter
t_2	Bread,PeanutButter
t_3	Bread,Milk,PeanutButter
t_4	Beer,Bread
t_5	Beer,Milk

Minsupp = 60%

I = { Beer, Bread, Jelly, Milk, PeanutButter}

X= {Bread,PeanutButter} ; **Count(X) = 3 và |T| = 5**

→ supp(X) = 60%→ X- tập phổ biến

X_2 = {Bread} → supp(X_2) = ?

X_3 = {PeanutButter} → supp(X_3) = ?; X_2 và X_3 có phổ biến ?

X_4 = {Milk}, X_5={Milk, Bread} → X_4 và X_5 có phổ biến ?

Mẫu tối đại

- **Mẫu tối đại** (Max-Pattern) là:
  - *Mẫu phổ biến* và *không tồn tại tập nào bao nó là phổ biến* (Bayardo – SIGMOD'98)
    - {B, C, D, E}, {A, C, D} - tập phổ biến tối đại
    - {B, C, D} - không phải tập phổ biến tối đại

Minsupp=2

Tid	Items
10	A,B,C,D,E
20	B,C,D,E,
30	A,C,D,F

Mẫu đóng

- **Mẫu đóng** (Closed-Pattern) là:

  - *Mẫu phổ biến* và *không tồn tại tập nào bao nó có cùng độ phổ biến như nó.* (Pasquier, ICDT'99)

- Ví dụ :

  - {A, B}, {A, B, D}, {A,B, C}: tập phổ biến đóng.

  - {A, B}: không phải tập phổ biến tối đại

TID	Items
10	a, b, c
20	a, b, c
30	a, b, d
40	a, b, d,
50	c, e, f

Minsupp=2

Luật kết hợp

- Luật kết hợp được thể hiện theo dạng

  $X \rightarrow Y$, trong đó $X, Y \subset I$ và $X \cap Y = \varnothing$

  – X (hoặc Y) là tập hợp các hạng mục, gọi là tập hạng mục (itemset).
- Sức mạnh của luật được đánh giá bằng độ đo support và độ đo confidence.
- Ví dụ:
  – A = {Beef, Chicken, Cheese} là tập hạng mục.
  – Một luật kết hợp có thể suy ra từ A là

    Beef, Chicken $\rightarrow$ Cheese

    X Y

Độ hỗ trợ của luật - Support

- Độ hỗ trợ (support) của luật $X \rightarrow Y$ là tỉ lệ giao dịch trong T chứa $X \cup Y$.
- Gọi n là số giao dịch trong T.

$$support = \Pr(X \cup Y) = \frac{(X \cup Y).count}{n}$$

- Support (viết tắt là sup) xác định mức độ phổ biến của luật áp dụng trên cơ sở dữ liệu giao dịch T.

Độ tin cậy của luật - Confidence

- Độ tin cây (confidence) của luật $X \rightarrow Y$ là tỉ lệ giao dịch trong T khi chứa X thì cũng chứa Y.
- Công thức tính:

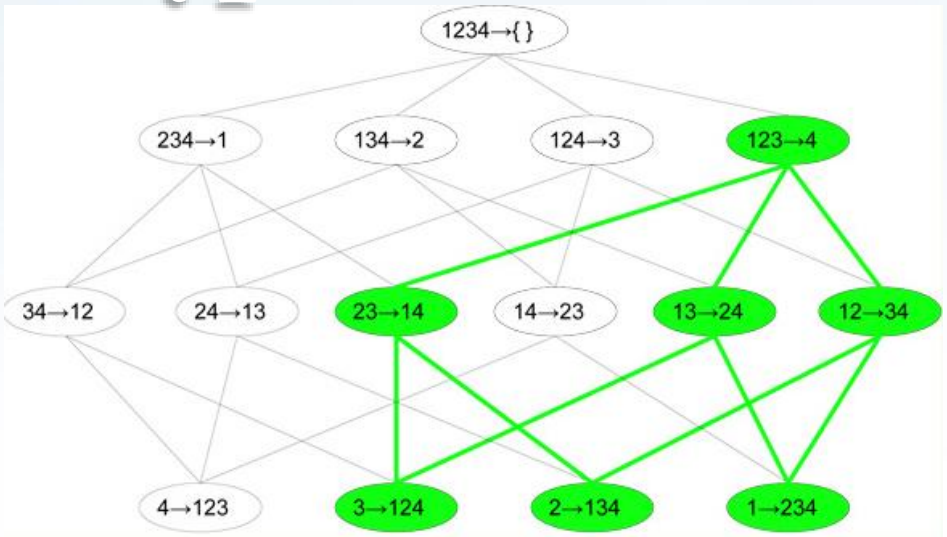
$$confidence = \Pr(Y|X) = \frac{(X \cup Y).count}{X.count}$$

- Confidence (viết tắt là conf) xác định tính có thể dự đoán trước của luật.

Nội dung

- Dữ liệu giao dịch
- Khái niệm cơ bản về:
  - Mẫu phổ biến
  - Luật kết hợp
- Khai thác luật kết hợp
- Thuật toán Apriori
  - Phát sinh tập hạng mục phổ biến
  - Xây dựng luật kết hợp từ tập phổ biến

Khai Thác Luật Kết Hợp



Bài toán khai thác LKH (1/2)

- Mục tiêu: Cho cơ sở dữ liệu giao dịch *T*, phát hiện mọi luật kết hợp trong T có
  - Độ hỗ trợ support lớn hơn hay bằng giá trị tối thiểu minsup.
  - Độ tin cậy confidence lớn hơn hay bằng giá trị tối thiểu minconf.
- Đặc điểm chính:
  - Tính đầy đủ: tìm mọi luật
  - Khai thác trên dữ liệu có kích thước lớn

Bài toán khai thác LKH (2/2)

- Là một tác vụ khai thác dữ liệu cơ bản.
  - Sáng kiến mô hình quan trọng nhất
  - Được cộng đồng khai thác dữ liệu và cơ sở dữ liệu nghiên cứu rộng rãi.
- Được giới thiệu lần đầu tiên bởi Agrawal et al. vào năm 1993.
- Giả sử mọi dữ liệu rời rạc, chưa có thuật toán tốt cho dữ liệu số.

Ví dụ khai thác LKH

TID	Transaction
t1	Beef, Chicken, Milk
t2	Beef, Cheese
t3	Cheese, Boots
t4	Beef, Chicken, Cheese
t5	Beef, Chicken, Clothes, Cheese, Milk
t6	Chicken, Clothes, Milk
t7	Chicken, Milk, Clothes

CSDL *T*

Minsup = 30%

Minconf = 80%

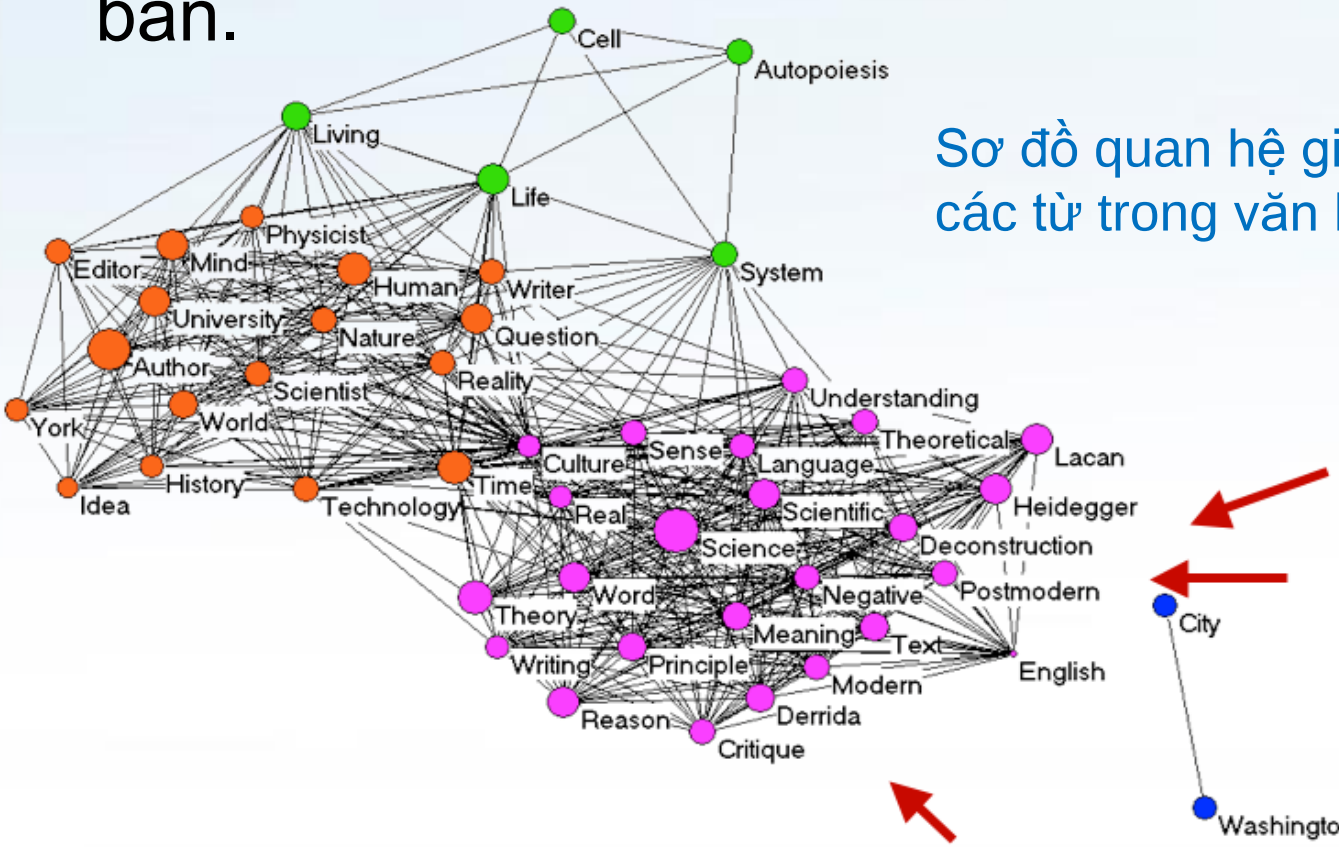
- Chicken, Clothes $\rightarrow$ Milk [sup = 3/7, conf = 3/3]
  - Là luật hợp lệ vì sup > 30% và conf > 80%
- Chicken $\rightarrow$ Clothes [sup = 3/7, conf = 3/5]
  - Không hợp lệ vì sup > 30% nhưng conf < 80%

Ứng dụng khai thác LKH (1/3)

- Phân tích dữ liệu giỏ mua hàng: ứng dụng cơ bản của khai thác luật kết hợp.
  - Mục tiêu: phát hiện sự liên quan giữa các món hàng được mua trong siêu thị (cửa hàng).
  - Ví dụ: luật Cheese $\rightarrow$ Beer [support = 10%, confidence = 80%]
    - 10% khách hàng mua *Cheese* và *Beer* chung
    - 80% khách hàng hễ mua *Cheese* thì sẽ mua *Beer* cùng

Ứng dụng khai thác LKH (2/3)

- Khai thác tài liệu văn bản: tìm mối quan hệ đồng xuất hiện của các từ trong văn bản.



Sơ đồ quan hệ giữa các từ trong văn bản

Ứng dụng khai thác LKH (3/3)

- Khai thác tài liệu Web: phát hiện các mẫu hành vi sử dụng Web.
  - Ứng dụng: xây dựng hệ thống tư vấn khách hàng, phân tích thiết kê Web,…
  - Ví dụ mẫu truy cập của người dùng
    - 60% người dùng truy cập /home/products/file1.html, sẽ đi theo chuỗi /home ==> /home/whatsnew ==> /home/products ==> /home/products/file1.html

Phương pháp khai thác LKH

- Có rất nhiều phương pháp khai thác luật kết hợp.
  - Khác nhau về chiến lược và cấu trúc dữ liệu
  - Tập luật kết quả đều giống nhau.
    - Cho cơ sở dữ liệu giao dịch *T*, giá trị minsup và minconf, tập các luật kết hợp tồn tại trong T là xác định đơn nhất..
- Các thuật toán phải cho ra cùng tập luật kết quả mặc dù khác nhau về hiệu quả tính toán và yêu cầu bộ nhớ.

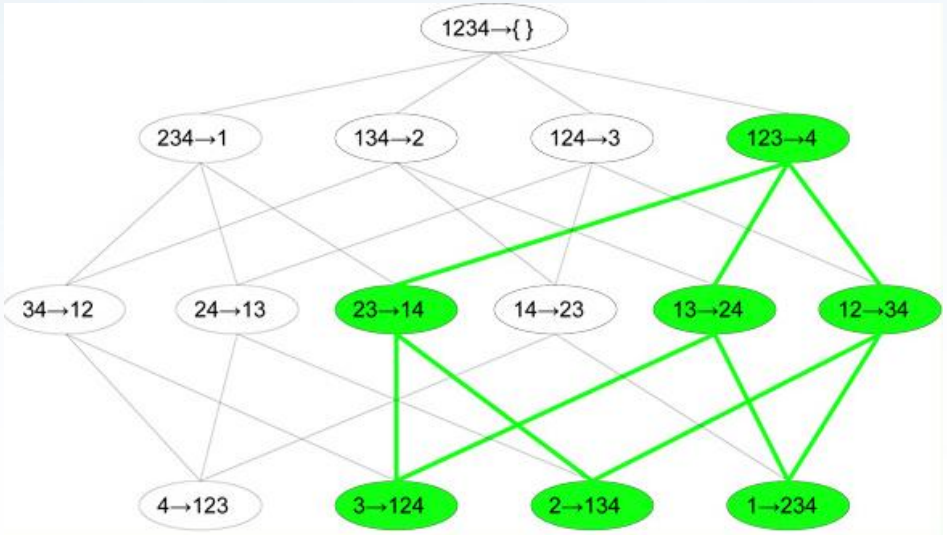
Thuật toán khai thác LKH

- Một số thuật toán khai thác luật kết hợp
  - Apriori (1994): tìm kiếm theo chiều rộng
  - Paritition (1995): tương tự Apriori, dùng phần giao tập hợp để xác định giá trị support.
  - Eclat (1997): kết hợp duyệt chiều sâu và phần giao tidlist.
  - FP-Growth (2000): duyệt cây phát triển mẫu theo chiều sâu

Nội dung

- Dữ liệu giao dịch
- Khái niệm cơ bản về:
  - Mẫu phổ biến
  - Luật kết hợp
- Khai thác luật kết hợp
- Thuật toán Apriori
  - Phát sinh tập hạng mục phổ biến
  - Xây dựng luật kết hợp từ tập phổ biến

THUẬT TOÁN APRIORI



Giới thiệu thuật toán

- R. Agrawal và R. Srikant đề xuất Apriori vào năm 1994.
  - Bài báo “Fast algorithms for mining association rules in large databases”, VLDB.
- Là một trong những thuật toán khai thác luật kết hợp căn bản và nổi tiếng nhất.
- Được thiết kế cho cơ sở dữ liệu giao dịch.
  - Ví dụ: các lượt mua hàng trong siêu thị, những mẫu hành vi trên Web,…

Thuật toán Apriori

- Là thuật toán tìm kiếm theo chiều rộng
- Thuật toán Apriori bao gồm hai bước chính:
  1. Phát sinh mọi tập hạng mục phổ biến
     - Tập hạng mục phổ biến (frequent itemset) là tập hạng mục có giá trị support ≥ minsup.
  2. Phát sinh mọi luật kết hợp tin cậy từ tập hạng mục phổ biến
     - Luật kết hợp tin cậy là luật có giá trị confidence ≥ minconf.

Tập k-hạng mục

- Kích thước tập hạng mục: là số hạng mục có trong tập hợp.
- Tập *k*-hạng mục (*k*-itemset) là tập hạng mục có kích thước *k*.
- Ví dụ:
  - Tập 1-hạng mục: {Beef}, {Chicken}, {Boots}
  - Tập 2-hạng mục: {Beef, Boots}, {Clothes, Milk}
  - Tập 3-hạng mục: {Chicken, Clothes, Milk}

Ví dụ thuật toán Apriori (1/2)

TID	Transaction
t1	Beef, Chicken, Milk
t2	Beef, Cheese
t3	Cheese, Boots
t4	Beef, Chicken, Cheese
t5	Beef, Chicken, Clothes, Cheese, Milk
t6	Chicken, Clothes, Milk
t7	Chicken, Milk, Clothes

CSDL *T*

Minsup = 30%

Minconf = 80%

- {Chicken, Clothes, Milk} là tập 3-hạng mục phổ biến vì có support = 3/7.

Ví dụ thuật toán Apriori (2/2)

TID	Transaction
t1	Beef, Chicken, Milk
t2	Beef, Cheese
t3	Cheese, Boots
t4	Beef, Chicken, Cheese
t5	Beef, Chicken, Clothes, Cheese, Milk
t6	Chicken, Clothes, Milk
t7	Chicken, Milk, Clothes

CSDL *T*

Minsup = 30%

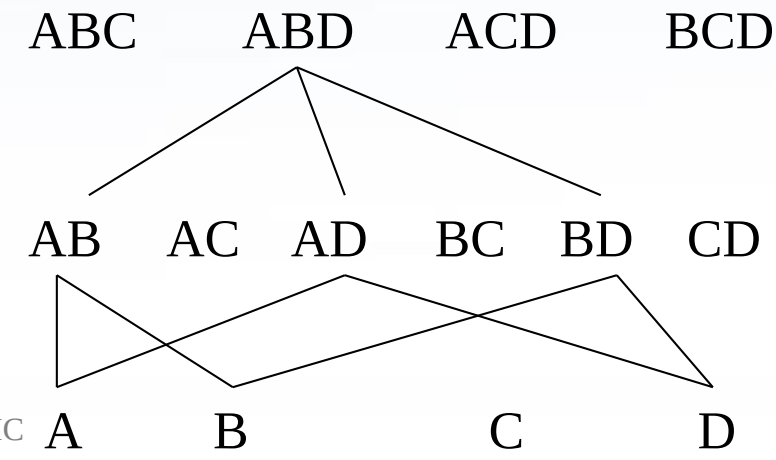
Minconf = 80%

- Từ {Chicken, Clothes, Milk} có các luật
  - Chicken, Clothes → Milk [sup = 3/7, conf = 3/3]
  - Clothes, Milk → Chicken [sup = 3/7, conf = 3/3]
  - Clothes → Milk, Chicken [sup = 3/7, conf = 3/3]

Tính chất Apriori (1/2)

- Tính chất *apriori*, hay bao đóng hướng xuống (downward closure):

  ***Nếu một tập hạng mục thỏa độ hỗ trợ tối thiểu thì mọi tập con không rỗng của nó cũng thỏa độ hỗ trợ tối thiểu*.**



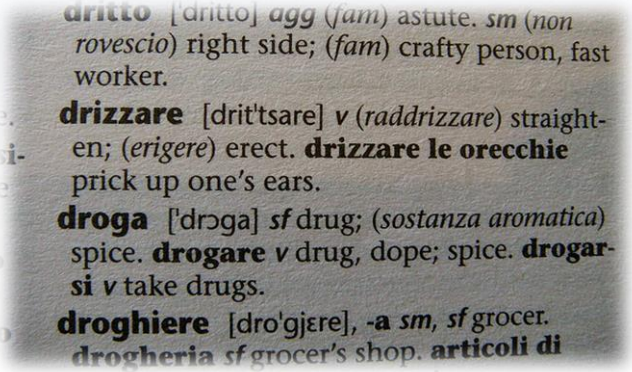
Source: CS583, Bing Liu, UIC

Tính chất Apriori (2/2)

- Tính chất Apriori***: Nếu một tập hạng mục thỏa độ hỗ trợ tối thiểu thì mọi tập con không rỗng của nó cũng thỏa độ hỗ trợ tối thiểu***.
- Thảo luận:
  - Lý giải tại sao tính chất này đúng?
  - Nếu tập con không thỏa độ hỗ trợ tối thiểu thì tập bao nó (tập cha) có thỏa độ hỗ trợ tối thiểu hay không?

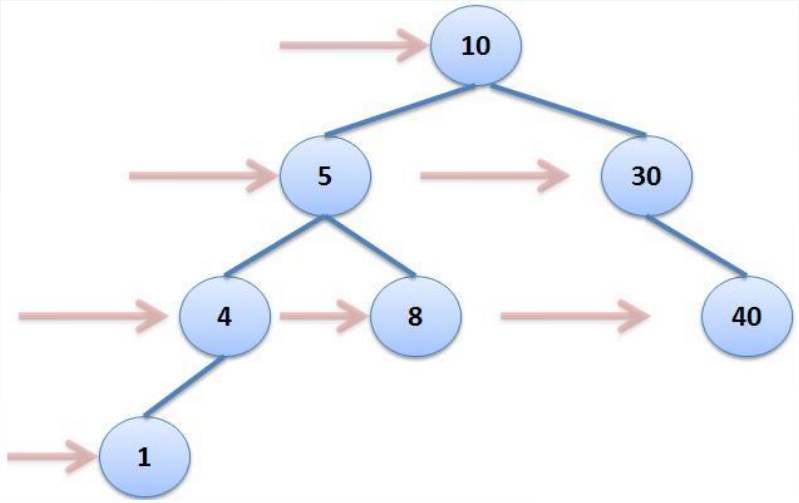
Phát sinh tập phổ biến (1/2)

- **Qui ước**: các hạng mục trong *l* được sắp xếp theo thứ tự từ điển.
  - Áp dụng cho mọi tập hạng mục và trong suốt quá trình thuật toán.
  - Kí hiệu {*w*[1], *w*[2],…, *w*[*k*]} là tập *k* hạng mục, trong đó *w*[1] < *w*[2] <…<w[*k*] theo thứ tự từ điển.


dritto ['dritto] *agg* (*fam*) astute. *sm* (*non rovescio*) right side; (*fam*) crafty person, fast worker.
drizzare [drit'tsare] *v* (*raddrizzare*) straighten; (*erigere*) erect. **drizzare le orecchie** prick up one's ears.
droga ['drɔga] *sf* drug; (*sostanza aromatica*) spice. **drogare** *v* drug, dope; spice. **drogarsi** *v* take drugs.
droghiere [dro'gjɛre], **-a** *sm, sf* grocer.
drogheria *sf* grocer's shop. **articoli di**

Phát sinh tập phổ biến (2/2)

- Thuật toán Apriori phát sinh tập hạng mục phổ biến dựa trên tìm kiếm theo mức (level-wise search).



- Quá trình phát sinh thực hiện nhiều lần duyệt dữ liệu.

Thuật toán Apriori

Algorithm Apriori(T)

1 $C_1 \leftarrow$ init-pass(T); // the first pass over T
2 $F_1 \leftarrow \{f \mid f \in C_1, f.\text{count}/n \geq minsup\}$; // n is the no. of transactions in T
3 **for** ($k = 2$; $F_{k-1} \neq \emptyset$; k++) **do**
4 $C_k \leftarrow$ candidate-gen(F_{k-1});
5 **for** each transaction $t \in T$ **do**
6 **for** each candidate $c \in C$
7 **if** c is contained in t **then**
8 $c.count$++;
9 **endfor**
10 **endfor**
11 $F_k \leftarrow \{c \in C_k \mid c.count/n \geq minsup\}$
12 **endfor**
13 **return** $F \leftarrow \bigcup_k F_k$;

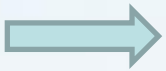
Lần duyệt thứ 1

- Tính độ hỗ trợ cho từng hạng mục
- Xác định F_1 gồm những tập phổ biến 1-hạng mục

Ví dụ thuật toán Apriori

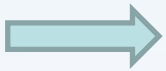
CSDL *T* **Minsupp = 40%**

TID	Transaction
10	{Bread, Cheese, Juice}
20	{Milk, Bread, Yogurt}
30	{Bread, Juice, Milk}
40	{Eggs, Bread, Cheese, Juice}
50	{Cheese, Juice, Milk}



C_1

Item
Bread
Cheese
Eggs
Juice
Milk
Yogurt



F_1

Item	Sup
Bread	4
Cheese	3
Juice	4
Milk	3

Xem thêm kĩ thuật đếm độ trợ trong phần Phụ Lục!

Thuật toán Apriori

```
Algorithm Apriori(T)
1    C1 ← init-pass(T);                          // the first pass over T
2    F1 ← {f | f ∈ C1, f.count/n ≥ minsup};      // n is the no. of transactions in T
3    for (k = 2; Fk−1 ≠ ∅; k++) do               // subsequent passes over T
4        Ck ← candidate-gen(Fk−1);
5        for each transaction t ∈ T do
6            for each candidate c ∈ Ck
7                if c is contained in t th
8                    c.count++;
9            endfor
10       endfor
11       Fk ← {c ∈ Ck | c.count/n ≥ minsup}
12   endfor
13   return F ← ⋃k Fk;
```

Lần duyệt thứ k, $k \geq 2$

1. Phát sinh tập hạng mục ứng viên C_k từ F_{k-1} bằng hàm candidate-gen.

Ví dụ thuật toán Apriori

CSDL *T* **Minsupp = 40%**

TID	Transaction
10	{Bread, Cheese, Juice}
20	{Milk, Bread, Yogurt}
30	{Bread, Juice, Milk}
40	{Eggs, Bread, Cheese, Juice}
50	{Cheese, Juice, Milk}

1st scan

C_1

Item	Sup
Bread	4
Cheese	3
Eggs	1
Juice	4
Milk	3
Yogurt	1

F_1

Item	Sup
Bread	4
Cheese	3
Juice	4
Milk	3

C_2

Item
Bread, Cheese
Bread, Juice
Bread, Milk
Cheese, Juice
Cheese, Milk
Juice, Milk

Thuật toán Apriori

```
Algorithm Apriori(T)
1    C1 ← init-pass(T);                          // the first pass over T
2    F1 ← {f | f ∈ C1, f.count/n ≥ minsup};      // n is the no. of transactions in T
3    for (k = 2; Fk-1 ≠ ∅; k++) do               // subsequent passes over T
4        Ck ← candidate-gen(Fk-1);
5        for each transaction t ∈ T do           // scan the data once
6            for each candidate c ∈ Ck do
7                if c is contained in t then
8                    c.count++;
9            endfor
10       endfor
11       Fk ← {c ∈ Ck | c.count/n ≥ m
12   endfor
13   return F ← ⋃k Fk;
```

Lần duyệt thứ k, $k \geq 2$

1. Phát sinh tập hạng mục ứng viên C_k từ F_{k-1} bằng hàm candidate-gen.
2. Duyệt dữ liệu và tính support cho mỗi ứng viên c trong C_k.

Ví dụ thuật toán Apriori

CSDL *T*

TID	Transaction
10	{Bread, Cheese, Juice}
20	{Milk, Bread, Yogurt}
30	{Bread, Juice, Milk}
40	{Eggs, Bread, Cheese, Juice}
50	{Cheese, Juice, Milk}

Minsupp = 40%

1st scan

C_1

Item	Sup
Bread	4
Cheese	3
Eggs	1
Juice	4
Milk	3
Yogurt	1

F_1

Item	Sup
Bread	4
Cheese	3
Juice	4
Milk	3
Yogurt	1

C_2

Item
Bread, Cheese
Bread, Juice
Bread, Milk
Cheese, Juice
Cheese, Milk
Juice, Milk

2nd scan

C_2

Item	Sup
Bread, Cheese	2
Bread, Juice	3
Bread, Milk	2
Cheese, Juice	3
Cheese, Milk	1
Juice, Milk	2

Thuật toán Apriori

```
Algorithm Apriori(T)
1    C1 ← init-pass(T);                                  // the first pass over T
2    F1 ← {f | f ∈ C1, f.count/n ≥ mi
3    for (k = 2; Fk−1 ≠ ∅; k++) do
4       Ck ← candidate-gen(Fk−1);
5       for each transaction t ∈ T do
6          for each candidate c ∈ Ck
7             if c is contained in t th
8                c.count++;
9          endfor
10      endfor
11      Fk ← {c ∈ Ck | c.count/n ≥ minsup}
12   endfor
13   return F ← ⋃k Fk;
```

Lần duyệt thứ *k*, *k* ≥ 2

1. Phát sinh tập hạng mục ứng viên C_k từ F_{k-1} bằng hàm candidate-gen.
2. Duyệt dữ liệu và tính support cho mỗi ứng viên *c* trong C_k.
3. Xác định các tập *k*-hạng mục phổ biến từ tập ứng viên.

Ví dụ thuật toán Apriori

CSDL *T*

TID	Transaction
10	{Bread, Cheese, Juice}
20	{Milk, Bread, Yogurt}
30	{Bread, Juice, Milk}
40	{Eggs, Bread, Cheese, Juice}
50	{Cheese, Juice, Milk}

Minsupp = 40%

1st scan

C_1

Item	Sup
Bread	4
Cheese	3
Eggs	1
Juice	4
Milk	3
Yogurt	1

F_1

Item	Sup
Bread	4
Cheese	3
Juice	4
Milk	3
Yogurt	1

C_2

Item
Bread, Cheese
Bread, Juice
Bread, Milk
Cheese, Juice
Cheese, Milk
Juice, Milk

2nd scan

C_2

Item	Sup
Bread, Cheese	2
Bread, Juice	3
Bread, Milk	2
Cheese, Juice	3
Cheese, Milk	1
Juice, Milk	2

F_2

Item	Sup
Bread, Cheese	2
Bread, Juice	3
Bread, Milk	2
Cheese, Juice	3
Juice, Milk	2

Thuật toán Apriori

```
Algorithm Apriori(T)
1    C1 ← init-pass(T);                          // the first pass over T
2    F1 ← {f | f ∈ C1, f.count/n ≥ minsup};      // n is the no. of transactions in T
3    for (k = 2; Fk-1 ≠ ∅; k++) do               // subsequent passes over T
4        Ck ← candidate-gen(Fk-1);
5        for each transaction t ∈ T do           // scan the data once
6            for each candidate c ∈ Ck do
7                if c is contained in t then
8                    c.count++;
9            endfor
10       endfor
11       Fk ← {c ∈ Ck | c.count/n ≥ minsup}
12   endfor
13   return F ← ⋃k Fk;
```

Đầu ra

Tập *F* gồm tất cả các tập hạng mục phổ biến.

Ví dụ thuật toán Apriori

C_3

Item
Bread, Cheese, Juice
Bread, Juice, Milk

3rd scan

C_3

Item	Sup
Bread, Cheese, Juice	2
Bread, Juice, Milk	1

F_3

Item	Sup
Bread, Cheese, Juice	2

$$F = F_1 \cup F_2 \cup F_3$$

Hàm Candidate-gen

- Hàm phát sinh ứng viên gồm 2 bước:
  - Gia nhập (join step): kết hợp hai tập (k-1)-hạng mục phổ biến để tạo ra ứng viên tiềm năng c.
    - Tập hạng mục phổ biến f_1 và f_2 có các hạng mục hoàn toàn giống nhau trừ hạng mục cuối cùng.
    - c được bổ sung vào tập ứng viên C_k.
  - Tỉa nhánh (prune step): kiểm tra mọi tập con kích thước k-1 của c có thuộc F_{k-1} hay không.
    - Nếu tồn tại tập con không thuộc F_{k-1}, c không phổ biến theo tính chất *apriori*. Xóa c ra khỏi C_k.

Hàm Candidate-gen

```
Function candidate-gen(F_{k-1})
1    C_k ← ∅;                                  // initialize the set of candidates
2    forall f_1, f_2 ∈ F_{k-1}                 // find all pairs of frequent itemsets
3      with f_1 = {i_1, ..., i_{k-2}, i_{k-1}} // that differ only in the last item
4      and f_2 = {i_1, ..., i_{k-2}, i'_{k-1}}
5      and i_{k-1} < i'_{k-1} do               // according to the lexicographic order
6        c ← {i_1, ..., i_{k-1}, i'_{k-1}};    // join the two itemsets f_1 and f_2
7        C_k ← C_k ∪ {c};                      // add the new itemset c to the candidates
8        for each (k-1)-subset s of c
9          if (s ∉ F_{k-1}) then
10           delete c from C_k;
11       endfor
12   endfor
13   return C_k;                               // return the generated candidates
```

Bước gia nhập

Kết hợp hai tập (*k*-1) hạng mục phổ biến tạo ứng viên *c*.

Hàm Candidate-gen

```
Function candidate-gen(F_{k-1})
1    C_k ← ∅;                                  // initialize the set of candidates
2    forall f_1, f_2 ∈ F_{k-1}                 // find all pairs of frequent itemsets
3       with f_1 = {i_1, ... , i_{k-2}, i_{k-1}}   // that differ only in the last item
4       and f_2 = {i_1, ... , i_{k-2}, i'_{k-1}}
5       and i_{k-1} < i'_{k-1} do
6          c ← {i_1, ..., i_{k-1}, i'_{k-1}};
7          C_k ← C_k ∪ {c};
8          for each (k-1)-subset s of c do
9             if (s ∉ F_{k-1}) then
10               delete c from C_k;          // delete c from the candidates
11         endfor
12   endfor
13   return C_k;                               // return the generated candidates
```

Bước tỉa nhánh

Kiểm tra mọi tập con kích thước k-1 của c có thuộc F_{k-1}. Nếu tồn tại tập con không thuộc F_{k-1} thì xóa c ra khỏi C_k.

Ví dụ hàm Candidate-gen

- Giả sử ta có các tập hạng mục phổ biến ở mức 3 là

F_3 = {{1, 2, 3}, {1, 2, 4}, {1, 3, 4}, {1, 3, 5}, {2, 3, 4}}

- Sau bước gia nhập:
  - C_k = {{1, 2, 3, 4}, {1, 3, 4 ,5}}
- Sau bước tỉa nhánh:
  - C_k = {{1, 2, 3, 4}}
  - {1, 3 ,4 , 5} không phổ biến vì {1, 4, 5} không thuộc F_3.

Bài tập áp dụng 1

- Cho cơ sở dữ liệu giao dịch *D* và minsup = 30%. Hãy áp dụng thuật toán Apriori để tìm mọi tập hạng mục phổ biến.

TID	Transaction
t1	Beef, Chicken, Milk
t2	Beef, Cheese
t3	Cheese, Boots
t4	Beef, Chicken, Cheese
t5	Beef, Chicken, Clothes, Cheese, Milk
t6	Chicken, Clothes, Milk
t7	Chicken, Milk, Clothes

CSDL *T*

Minsup = 30%

Bài tập áp dụng 1 – Đáp án

TID	Transaction
t1	Beef, Chicken, Milk
t2	Beef, Cheese
t3	Cheese, Boots
t4	Beef, Chicken, Cheese
t5	Beef, Chicken, Clothes, Cheese, Milk
t6	Chicken, Clothes, Milk
t7	Chicken, Milk, Clothes

CSDL *T*

Minsup = 30%

≈ 3 giao dịch

- F_1: {{Beef}:4, {Cheese}:4, {Chicken}:5, {Clothes}:3, {Milk}:4}
- C_2: {{Beef, Cheese}, {Beef, Chicken}, {Beef, Clothes}, {Beef, Milk}, {Cheese, Chicken}, {Cheese, Clothes}, {Cheese, Milk}, {Chicken, Clothes}, {Chicken, Milk}. {Clothes, Milk}}

Bài tập áp dụng 1 – Đáp án

TID	Transaction
t1	Beef, Chicken, Milk
t2	Beef, Cheese
t3	Cheese, Boots
t4	Beef, Chicken, Cheese
t5	Beef, Chicken, Clothes, Cheese, Milk
t6	Chicken, Clothes, Milk
t7	Chicken, Milk, Clothes

CSDL *T*

Minsup = 30%

≈ 3 giao dịch

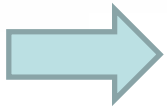
- F_2: {{Beef,Chicken}:3, {Beef, Cheese}:3, {Chicken, Clothes}:3, {Chicken, Milk}:4, {Clothes, Milk}:3}
- C_3: {{Chicken, Clothes, Milk}}
- F_3: {{Chicken, Clothes, Milk}:3}

Nhận xét thuật toán (1/4)

- Apriori là thuật toán có độ phức tạp số mũ.
  - Gọi số hạng mục trong *I* là *m*, không gian tập hạng mục là $O(2^m)$.
  - Tính thưa của dữ liệu và giá trị độ hỗ trợ tối thiểu cao làm cho quá trình khai thác hiệu quả.

CSDL *T*

TID	Transaction
10	{Juice}
20	{Milk, Yogurt}
30	{Bread, Juice, Milk}
40	{Eggs, Bread, Cheese}
50	{Cheese}

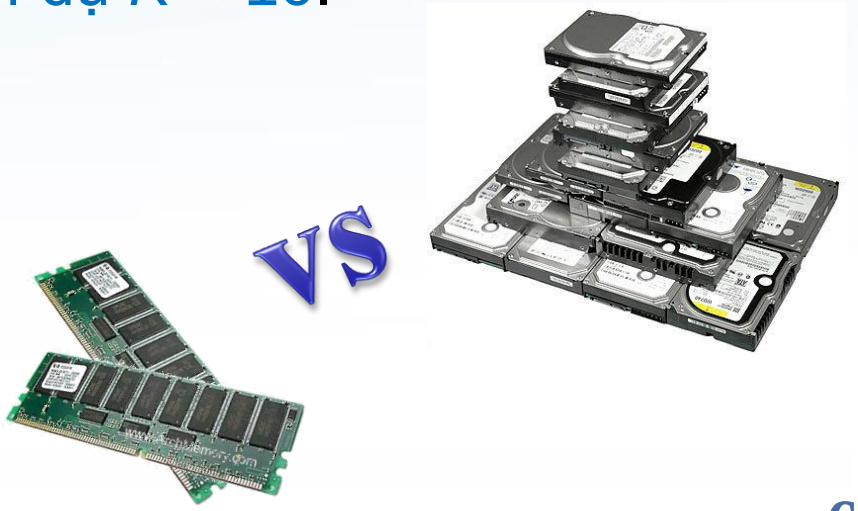


Dữ liệu nhị phân hóa

B	C	E	J	M	Y
0	0	0	1	0	0
0	0	0	0	1	1
1	0	0	1	1	0
1	1	1	0	0	0
0	1	0	0	0	0

Nhận xét thuật toán (2/4)

- Có thể xử lý các tập dữ liệu lớn do chương trình không tải toàn bộ dữ liệu vào bộ nhớ.
  - Duyệt dữ liệu *K* lần, trong đó *K* là kích thước tập hạng mục lớn nhất.
  - *K* thường nhỏ, ví dụ $K < 10$.



Nhận xét thuật toán (3/4)

- Thuật toán thực hiện tìm kiếm theo mức nên có thể linh hoạt dừng ở mức bất kỳ.
  - Thích hợp cho những ứng dụng không cần khai thác tập hạng mục hay luật dài.
- Mọi thuật toán đều tìm ra cùng kết quả tập hạng mục phổ biến.
  - Tính chất này không có ở những tác vụ khai thác dữ liệu khác như phân lớp, gom nhóm,…

Nhận xét thuật toán (4/4)

- Khuyết điểm của Apriori:
  - Phát sinh quá nhiều tập ứng viên
  - Phải duyệt lặp đi lặp lại toàn bộ cơ sở dữ liệu nhiều lần và kiểm tra một tập lớn các ứng viên bằng phương pháp so khớp mẫu.
  - Số lượng tập hạng mục và luật được phát sinh là khổng lồ $\Rightarrow$ khó khăn cho việc phân tích thông tin.
- Các nhà nghiên cứu đề xuất nhiều phương pháp để giải quyết vấn đề này như FP-Growth, bài toán độ thú vị (interestingness problem).

Bài tập áp dụng 2

- Cho cơ sở dữ liệu giao dịch *D* và minsup = 50%. Hãy áp dụng thuật toán Apriori để tìm mọi tập hạng mục phổ biến.

TID	Transaction
t1	Crab, Milk, Cheese, Bread
t2	Cheese, Milk, Apple, Pie, Bread
t3	Apple, Crab, Pie, Bread
t4	Bread, Milk, Cheese

CSDL *T*

Minsup = 50%

Bài tập áp dụng 2 – Đáp án

TID	Transaction
t1	Crab, Milk, Cheese, Bread
t2	Cheese, Milk, Apple, Pie, Bread
t3	Apple, Crab, Pie, Bread
t4	Bread, Milk, Cheese

CSDL *T*

Minsup = 50%

≈ 2 giao dịch

- F_1: {{Apple}:2, {Bread}:4, {Cheese}:3, {Crab}:2, {Milk}:3, {Pie}:2}
- C_2: {{Apple, Bread}, {Apple, Cheese}, {Apple, Crab}, {Apple, Milk}, {Apple, Pie}, {Bread, Cheese}, {Bread, Crab}, {Bread, Milk}, {Bread, Pie}, {Cheese, Crab}, {Cheese, Milk}, {Cheese, Pie}, {Crab, Milk}, {Crab, Pie}, {Milk, Pie}}

Bài tập áp dụng 2 – Đáp án

TID	Transaction
t1	Crab, Milk, Cheese, Bread
t2	Cheese, Milk, Apple, Pie, Bread
t3	Apple, Crab, Pie, Bread
t4	Bread, Milk, Cheese

CSDL *T*

Minsup = 50%

≈ 2 giao dịch

- F_2: {{Apple, Bread}:2, {Apple, Pie}:2, {Bread, Cheese}:3, {Bread, Crab}:2, {Bread, Milk}:3, {Bread, Pie}:2, {Cheese, Milk}:3}
- C_3: {{Apple, Bread, Pie}, {Bread, Cheese, Milk}}
- F_3: {{Apple, Bread, Pie}:2, {Bread, Cheese, Milk}:3}

Nội dung

- Dữ liệu giao dịch
- Khái niệm cơ bản về:
  - Mẫu phổ biến
  - Luật kết hợp
- Khai thác luật kết hợp
- Thuật toán Apriori
  - Phát sinh tập hạng mục phổ biến
  - Xây dựng luật kết hợp từ tập phổ biến

Phát sinh luật kết hợp (1/5)

- Luật kết hợp được phát sinh từ tập hạng mục phổ biến.
- Với mỗi tập phổ biến *f*, với mỗi tập con không rỗng α thuộc *f*, luật tạo ra có dạng

$$(\boldsymbol{f} - \boldsymbol{\alpha}) \rightarrow \boldsymbol{\alpha},$$

nếu $confidence = \frac{f.count}{(f - \propto).count} \geq minconf$

  - Trong đó *f.count* (hay (*f* – α).*count*) là đếm hỗ trợ của *f* (hay (f – α)).

Phát sinh luật kết hợp (2/5)

- Độ hỗ trợ của luật là $\frac{f.count}{n}$, với *n* là số giao dịch của CSDL giao dịch *T.*
- Quá trình phát sinh luật không cần duyệt lại dữ liệu.
  - Vì *f* phổ biến nên mọi tập con không rỗng của nó cũng phổ biến.
  - Các giá trị đếm hỗ trợ đã được tính trong quá trình phát sinh tập hạng mục

Phát sinh luật kết hợp (3/5)

- Chiến lược phát sinh luật triệt để như thế không hiệu quả.
- Giải pháp khác:

  Nếu luật $(f - \alpha) \rightarrow \alpha$ hợp lệ thì mọi luật $(f - \alpha_{sub}) \rightarrow \alpha_{sub}$ phải hợp lệ.

  – Trong đó α_{sub} là tập con không rỗng của α.

  – Ví dụ:

  - Cho tập phổ biến {A, B, C, D}
  - Nếu (A,B $\rightarrow$ C,D) hợp lệ thì (A, B, C $\rightarrow$ D), (A, B, D $\rightarrow$ C) phải hợp lệ.

Phát sinh luật kết hợp (4/5)

- Qui trình phát sinh luật kết hợp
  - Từ tập phổ biến *f*, phát sinh mọi luật có hệ quả gồm một hạng mục.
  - Sử dụng tập 1-hạng mục và hàm candidate-gen() để phát sinh ứng viên hệ quả 2-hạng mục.
  - Tiếp tục lặp…

Phát sinh luật kết hợp (5/5)

```
Algorithm genRules(F)                    // F is the set of all frequent itemsets
1    for each frequent k-itemset f_k in F, k ≥ 2 do
2        output every 1-item consequent rule of f_k with confidence ≥ minconf and
             support ← f_k.count / n     // n is the total number of transactions in T
3        H_1 ← {consequents of all 1-item consequent rules derived from f_k above};
4        ap-genRules(f_k, H_1);
5    endfor

Procedure ap-genRules(f_k, H_m)          // H_m is the set of m-item consequents
1    if (k > m + 1) AND (H_m ≠ ∅) then
2       H_{m+1} ← candidate-gen(H_m);
3       for each h_{m+1} in H_{m+1} do
4           conf ← f_k.count / (f_k – h_{m+1}).count;
5           if (conf ≥ minconf) then
6               output the rule (f_k – h_{m+1}) → h_{m+1} with confidence = conf and
                support = f_k.count / n;     // n is the total number of transactions in T
7           else
8               delete h_{m+1} from H_{m+1};
9       endfor
10      ap-genRules(f_k, H_{m+1});
11   endif
```

VÍ DỤ PHÁT SINH LKH

TID	Transaction
10	{Bread, Cheese, Juice}
20	{Milk, Bread, Yogurt}
30	{Bread, Juice, Milk}
40	{Eggs, Bread, Cheese, Juice}
50	{Cheese, Juice, Milk}

CSDL *T*

Minsupp = 40%
Minconf = 75%

- F_1: {{Bread}:4, {Cheese}:3, {Juice}:4, {Milk}:3}
- F_2: {{Bread, Cheese}:2, {Bread, Juice}:3, {Bread, Milk}:2, {Cheese, Juice}:3, {Juice, Milk}:2}
- F_3: {{Bread, Cheese, Juice}:2}
- Ta phát sinh luật từ F_3 (phát sinh luật từ F_2 có thể làm tương tự)

Phát sinh luật kết hợp

```
Algorithm genRules(F)                    // F is the set of all frequent itemsets
1   for each frequent k-itemset f_k in F, k ≥ 2 do
2       output every 1-item consequent rule of f_k with confidence ≥ minconf and
            support ← f_k.count / n      // n is the total number of transactions in T
3       H_1 ← {consequents of all 1-item consequent rules derived from f_k above};
4       ap-genRules(f_k, H_1);
5   endfor

Procedure ap-genRules(f_k, H_m)          // H_m is the set of m-item consequents
1   if (k > m + 1) AND (H_m ≠ ∅) then
2      H_{m+1} ← candidate-gen(H_m);
3      for each h_{m+1} in H_{m+1} do
4          conf ← f_k.count / (f_k − h_{m+1}).count;
5          if (conf ≥ minconf) then
6              output the rule (f_k − h_{m+1}) → h_{m+1} with confidence = conf and
               support = f_k.count / n;  // n is the total number of transactions in T
7          else
8              delete h_{m+1} from H_{m+1};
9      endfor
10     ap-genRules(f_k, H_{m+1});
11  endif
```

Từ tập phổ biến *f*, phát sinh mọi luật có hệ quả gồm một hạng mục.

Ví dụ phát sinh LKH

TID	Transaction
10	{Bread, Cheese, Juice}
20	{Milk, Bread, Yogurt}
30	{Bread, Juice, Milk}
40	{Eggs, Bread, Cheese, Juice}
50	{Cheese, Juice, Milk}

CSDL *T*

Minsupp = 40%
Minconf = 60%

- {Bread, Cheese, Juice} $\in F_3$ phát sinh được các luật ứng viên có hệ quả một hạng mục sau:
  - Luật 1: Bread, Cheese $\rightarrow$ Juice [sup = 2/5, conf = 1]
  - Luật 2: Bread, Juice $\rightarrow$ Cheese [sup = 2/5, conf = 2/3]
  - Luật 3: Cheese, Juice $\rightarrow$ Bread [sup = 2/5, conf = 2/3]
- Theo ràng buộc minconf, cả 3 luật đều hợp lệ

  $\Rightarrow H_1$ = {{Bread}, {Cheese}, {Juice}}

Phát sinh luật kết hợp

```
Algorithm genRules(F)                    // F is the set of all frequent itemsets
1    for each frequent k-itemset f_k in F, k ≥ 2 do
2        output every 1-item consequent rule of f_k with confidence ≥ minconf and
             support ← f_k.count / n     // n is the total number of transactions in T
3        H_1 ←{consequents of all 1-item consequent rules derived from f_k above};
4        ap-genRules(f_k, H_1);
5    endfor

Procedure ap-genRules(f_k,
1    if (k > m + 1) AND (H_
2        H_{m+1} ← candidate-gen(H_m);
3        for each h_{m+1} in H_{m+1} do
4            conf ← f_k.count / (f_k – h_{m+1}).count;
5            if (conf ≥ minconf) then
6                output the rule (f_k – h_{m+1}) → h_{m+1} with confidence = conf and
                 support = f_k.count / n;     // n is the total number of transactions in T
7            else
8                delete h_{m+1} from H_{m+1};
9        endfor
10       ap-genRules(f_k, H_{m+1});
11   endif
```

Sử dụng tập m-hạng mục H_m để phát sinh tập ứng viên hệ quả (m+1)-hạng mục H_{m+1}.

Ví dụ phát sinh LKH

TID	Transaction
10	{Bread, Cheese, Juice}
20	{Milk, Bread, Yogurt}
30	{Bread, Juice, Milk}
40	{Eggs, Bread, Cheese, Juice}
50	{Cheese, Juice, Milk}

CSDL *T*

Minsupp = 40%
Minconf = 60%

- Phát sinh H2 = {{Bread, Cheese}, {Bread, Juice}, {Cheese, Juice}}

Phát sinh luật kết hợp

```
Algorithm genRules(F)                    // F is the set of all frequent itemsets
1    for each frequent k-itemset f_k in F, k ≥ 2 do
2        output every 1-item consequent rule of f_k with confidence ≥ minconf and
             support ← f_k.count / n     // n is the total number of transactions in T
3        H_1 ← {consequents of all 1-item consequent rules derived from f_k above};
4        ap-genRules(f_k, H_1);
5    endfor

Procedure ap-genRules(f_k,
1    if (k > m + 1) AND (H_
2        H_{m+1} ← candidate-ge
3        for each h_{m+1} in H_{m+1} do
4            conf ← f_k.count / (f_k − h_{m+1}).count;
5            if (conf ≥ minconf) then
6                output the rule (f_k − h_{m+1}) → h_{m+1} with confidence = conf and
                     support = f_k.count / n;     // n is the total number of transactions in T
7            else
8                delete h_{m+1} from H_{m+1};
9        endfor
10       ap-genRules(f_k, H_{m+1});
11   endif
```

Tạo luật có phần hệ quả là ứng viên (m+1)-hạng mục $\in H_{m+1}$, kiểm tra ràng buộc conf.

Ví dụ phát sinh LKH

TID	Transaction
10	{Bread, Cheese, Juice}
20	{Milk, Bread, Yogurt}
30	{Bread, Juice, Milk}
40	{Eggs, Bread, Cheese, Juice}
50	{Cheese, Juice, Milk}

CSDL *T*

Minsupp = 40%
Minconf = 60%

- Phát sinh H2 = {{Bread, Cheese}, {Bread, Juice}, {Cheese, Juice}}
- Các luật sau được phát sinh
  - ~~Luật 4: Juice → Bread, Cheese [sup = 2/5, conf = 2/4]~~
  - Luật 5: Cheese → Bread, Juice [sup = 2/5, conf = 2/3]
  - ~~Luật 6: Bread → Cheese, Juice [sup = 2/5, conf = 2/4]~~
- Các luật thu được bao gồm: 1, 2, 3, 5.

Bài tập áp dụng 1 (tt)

TID	Transaction
t1	Beef, Chicken, Milk
t2	Beef, Cheese
t3	Cheese, Boots
t4	Beef, Chicken, Cheese
t5	Beef, Chicken, Clothes, Cheese, Milk
t6	Chicken, Clothes, Milk
t7	Chicken, Milk, Clothes

CSDL *T*

Minsup = 30%

≈ *3 giao dịch*

Minconf = 80%

- F_1: {{Beef}:4, {Cheese}:4, {Chicken}:5, {Clothes}:3, {Milk}:4}
- F_2: {{Beef,Chicken}:3, {Beef, Cheese}:3, {Chicken, Clothes}:3, {Chicken, Milk}:4, {Clothes, Milk}:3}
- F_3: {{Chicken, Clothes, Milk}:3}

Bài tập áp dụng 1 – Đáp án

- Luật phát sinh từ F_2:
  - ~~Luật 1: Beef → Chicken [sup = 3/7, conf = 3/4]~~
  - ~~Luật 2: Chicken → Beef [sup = 3/7, conf = 3/5]~~
  - ~~Luật 3: Beef → Cheese [sup = 3/7, conf = 3/4]~~
  - ~~Luật 4: Cheese → Beef [sup = 3/7, conf = 3/4]~~
  - ~~Luật 5: Chicken → Clothes [sup = 3/7, conf = 3/5]~~
  - Luật 6: Clothes → Chicken [sup = 3/7, conf = 3/3]
  - Luật 7: Chicken → Milk [sup = 4/7, conf = 4/5]
  - Luật 8: Milk → Chicken [sup = 4/7, conf = 4/4]
  - Luật 9: Clothes → Milk [sup = 3/7, conf = 3/3]
  - Luật 10: Milk → Clothes [sup = 3/7, conf = 3/4]

Bài tập áp dụng 1 – Đáp án

- Luật phát sinh từ F_3: Chicken, Clothes, Milk}:

  Các luật có hệ quả gồm 1 hạng mục

  - Luật 11: Chicken, Clothes $\rightarrow$ Milk [sup = 3/7, conf = 3/3]
  - ~~Luật 12: Chicken, Milk $\rightarrow$ Clothes [sup = 3/7, conf = 3/4]~~
  - Luật 13: Clothes, Milk $\rightarrow$ Chicken [sup = 3/7, conf = 3/3]

  H1 = {{Chicken}, {Milk}} $\Rightarrow$ H2 = {{Chicken, Milk}}

  - Luật 14: Clothes $\rightarrow$ Milk, Chicken [sup = 3/7, conf = 3/3]

- Các luật phát sinh được là: 6, 7, 8, 9, 10, 11, 13.

Bài tập áp dụng 2

TID	Transaction
t1	Crab, Milk, Cheese, Bread
t2	Cheese, Milk, Apple, Pie, Bread
t3	Apple, Crab, Pie, Bread
t4	Bread, Milk, Cheese

CSDL *T*

Minsup = 50%

≈ *2 giao dịch*

Minconf = 80%

- F_1: {{Apple}:2, {Bread}:4, {Cheese}:3, {Crab}:2, {Milk}:3, {Pie}:2}
- F_2: {{Apple, Bread}:2, {Apple, Pie}:2, {Bread, Cheese}:3, {Bread, Crab}:2, {Bread, Milk}:3, {Bread, Pie}:2, {Cheese, Milk}:3}
- F_3: {{Apple, Bread, Pie}:2, {Bread, Cheese, Milk}:3}

Bài tập áp dụng 2 – Đáp án

- Luật phát sinh từ F_2:
  - Luật 1: Apple → Bread [sup = 2/4, conf = 2/2]
  - ~~Luật 2: Bread → Apple [sup = 2/4, conf = 2/4]~~
  - Luật 3: Apple → Pie [sup = 2/4, conf = 2/2]
  - Luật 4: Pie → Apple [sup = 2/4, conf = 2/2]
  - ~~Luật 5: Bread → Cheese [sup = 3/4, conf = 3/4]~~
  - Luật 6: Cheese → Bread [sup = 3/4, conf = 3/3]
  - ~~Luật 7: Bread → Crab [sup = 2/4, conf = 2/4]~~
  - Luật 8: Crab → Bread [sup = 2/4, conf = 2/2]
  - ~~Luật 9: Bread → Milk [sup = 3/4, conf = 3/4]~~
  - Luật 10: Milk → Bread [sup = 3/4, conf = 3/3]

Bài tập áp dụng 2 – Đáp án

- Luật phát sinh từ F_2:
  - ~~Luật 11: Bread → Pie [sup = 2/4, conf = 2/4]~~
  - Luật 12: Pie → Bread [sup = 2/4, conf = 2/2]
  - Luật 13: Cheese → Milk [sup = 3/4, conf = 3/3]
  - Luật 14: Milk → Cheese [sup = 3/4, conf = 3/3]

- Luật phát sinh từ F_3:
  - Luật 15: Apple, Bread → Pie [sup = 2/4, conf = 2/2]
  - Luật 16: Apple, Pie → Bread [sup = 2/4, conf = 2/2]
  - Luật 17: Bread, Pie → Apple [sup = 2/4, conf = 2/2]

Bài tập áp dụng 2 – Đáp án

- Luật phát sinh từ F_3:
  - Luật 18: Apple → Bread. Pie [sup = 2/4, conf = 2/2]
  - ~~Luật 19: Bread → Apple, Pie [sup = 2/4, conf = 2/4]~~
  - Luật 20: Pie → Apple, Bread [sup = 2/4, conf = 2/2]

  - Luật 21: Bread, Cheese → Milk [sup = 3/4, conf = 3/3]
  - Luật 22: Bread, Milk → Cheese [sup = 3/4, conf = 3/3]
  - Luật 23: Cheese, Milk → Bread [sup = 3/4, conf = 3/3]
  - ~~Luật 24: Bread → Cheese, Milk [sup = 3/4, conf = 3/4]~~
  - Luật 25: Cheese → Bread, Milk [sup = 3/4, conf = 3/3]
  - Luật 26: Milk → Bread, Cheese [sup = 3/4, conf = 3/3]

TỔNG KẾT

- Các thuật ngữ cơ bản về tập hạng mục phổ biến và luật kết hợp.
- Các độ đo đánh giá độ mạnh của luật: độ hỗ trợ support và độ tin cậy confidence
- Thuật toán Apriori: phát sinh tập hạng mục phổ biến và xây dựng luật kết hợp
  - Cài đặt chương trình chạy thuật toán Apriori
  - Thực hiện chạy tay các bước của thuật toán

Tài liệu tham khảo

- Chapter 2. B. Liu, *Web Data Mining-Exploring Hyperlinks, Contents, and Usage Data*, Springer Series on Data-Centric Systems and Applications, 2007.
- Chapter 6. Data Mining: Concepts & Technique, 3nd edition, J.Han, M.Kamber

Hỏi & Đáp

How to Count Supports of Candidates?

- Why counting supports of candidates a problem?
  - The total number of candidates can be very huge
  - One transaction may contain many candidates
- Method:
  - Candidate itemsets are stored in a *hash-tree*
  - *Leaf* node of hash-tree contains a list of itemsets and counts
  - *Interior* node contains a hash table
  - *Subset function*: finds all the candidates contained in a transaction

Counting Supports of Candidates Using Hash Tree