Regras de Associação

- A compra de um produto quando um outro produto é comprado representa uma Regra de Associação
- Regras de Associação são frequentemente utilizadas para apoiar campanhas de *marketing* e controle de estoque de lojas mas também em outras áreas, tais como descrever falhas em linhas de comunicação, ações na interface do usuário, crimes cometidos por uma pessoa, ocorrências de doenças recorrentes

José Augusto Baranauskas
Departamento de Física e Matemática – FFCLRP-USP
E-mail: augusto@usp.br
URL: http://dfm.ffclrp.usp.br/~augusto

Introdução

- Algoritmo Apriori proposto por Agrawal et al (1993)
- É um modelo estudado extensivamente pelas comunidades de bancos de dados e aprendizado de máquina
- Assume que os dados não categóricos; portanto, não se aplica a dados numéricos
- Inicialmente utilizado na análise de cesta de compras em supermercados (*Market Basket Analysis*) para determinar como os itens comprados por clientes estão relacionados
  - {leite, pão} $\rightarrow$ {manteiga} [sup = 5%, conf = 100%]

Introdução

- A geração de **Regras de Associação** (RA) é uma tarefa **descritiva** (Aprendizado Não Supervisionado)
  - Outros termos: AR (*Association Rule*), ARM (*Association Rule Mining*), *Link Analysis*, *Affinity Analysis, Market Basket Analysis*
- RAs são utilizadas para mostrar o relacionamento entre os exemplos (itens de dados)
- Os relacionamentos descobertos **não são inerentes** aos exemplos e eles **não representam** nenhum tipo de causa ou correlação
- Ao invés, RAs detectam o uso comum de itens

Introdução

- A tarefa de associação tem como objetivo encontrar elementos que implicam na presença de outros elementos em uma mesma transação, ou seja, encontrar relacionamentos ou padrões freqüentes entre conjuntos de exemplos
- Portanto, Regras de Associação representam padrões existentes nas transações de um banco de dados
- Um exemplo típico consiste em transações nas quais são armazenados os itens adquiridos por clientes
  - Uma regra de associação para este exemplo poderia ser **{leite, pão}** $\rightarrow$ **{manteiga}**, que indica que se o cliente compra **leite** e **pão**, com um determinado grau de certeza, ele também compra **manteiga**

Formato dos Exemplos

- Os exemplos podem estar armazenados como transações ou no formato de tabela atributo-valor
  - Formato de transação
    - A, B
    - A, C, D, E
    - A, D, F
  - Formato de tabela atributo-valor
    - X_1 X_2 X_3
    - A, B, D
    - B, C, E
- Dados no formato de tabela precisam ser convertidos para o formato de transação para aplicar o algoritmo de regras de associação
  - Usando a tabela atributo-valor anterior, a transformação para o formato de transações é
    - X_1=A, X_2=B, X_3=D
    - X_1=B, X_2=C, X_3=E

Regra de Associação

- Uma **regra** assume a forma **if L then R** que é equivalente a $L \rightarrow R \equiv R \leftarrow L \equiv R$:- L
- Normalmente, as partes esquerda L e direita R são complexos sem atributos comuns entre eles, ou seja
  - atributos(L) ∩ atributos(R) = Ø
- A parte esquerda L é denominada **condição**, **premissa**, **antecedente**, **cauda** ou **corpo** da regra
- A parte direita R é denominada **conclusão** ou **cabeça** da regra
- Note que em uma **regra de associação** não existe uma definição explícita de classe e qualquer atributo (ou atributos) pode ser usado na conclusão da regra
- Exemplo
  - if X_3='S' and $X_5 > 2$ then X_1='N‘ and $X_2<1$

Complexo

- É uma conjunção no formato atributo-valor, da forma:

$$X_i \; op \; \text{valor}$$

onde X_i é um atributo, *op* é um operador e valor é constante válida para o atributo X_i

- Exemplos
  - aparência=sol
  - temperatura=quente $\wedge$ umidade=alta
  - aparência=sol $\wedge$ umidade=alta $\wedge$ jogar=não

Item

- Na nomenclatura de RA, existem algumas definições adicionais e específicas da área, tais como **item** e ***itemset*** que são derivadas do conceito de **complexo**
- Um **item** pode ser visto como um único teste do tipo atributo-valor, ou seja, é um complexo contendo uma única condição
- Exemplos
  - aparência=sol
  - temperatura=quente
  - umidade=alta

Itemset

- Um ***itemset*** é uma conjunção de ***item***s, ou seja, os testes em um ***itemset*** estão ligados entre si pelo conectivo lógico **e** ($\wedge$)
- É óbvio que não há interesse em *itemsets* contendo dois diferentes valores para o mesmo atributo, já que ele não pode ocorrer nos exemplos de treinamento
  - Por exemplo: aparência=sol $\wedge$ aparência=nublado
- Normalmente omite-se a representação do conectivo, usando espaços ou vírgulas por simplificação
- Exemplos
  - aparência=sol
  - aparência=sol, umidade=normal
  - aparência=sol temperatura=quente umidade=alta jogar=não

Itemset

- O número de condições determina a dimensão de um *itemset*
- Um *itemset* contendo teste em um único atributo é denominado ***1-itemset***
- Um *itemset* contendo testes em dois atributos é denominado ***2-itemset***
- De forma geral, um *itemset* contendo **r** testes (**r** ≤ **m**, onde **m** é o número de atributos do conjunto de exemplos) é um ***r-itemset***

Exemplos (Esporte Favorito)

- *1-itemsets*
  - aparência=sol
  - aparência=nublado
  - aparência=chuva
  - temperatura=fria
  - temperatura=agradável
  - ...
- *2-itemsets*
  - aparência=sol, temperatura=fria
  - aparência=sol, temperatura=quente
  - aparência=sol, umidade=normal
  - ...
- *3-itemsets*
  - aparência=sol, temperatura=quente, umidade=alta
  - aparência=sol, temperatura=quente, jogar=não
  - ...
- *4-itemsets*
  - aparência=sol, temperatura=quente, umidade=alta, jogar=não
  - aparência=sol, umidade=alta, ventando=falso, jogar=não
  - ...

Geração de RA

- Regras de associação são similares a regras de classificação
- Seria possível empregar o algoritmo de indução de regras para cada ***itemset*** possível que poderia ocorrer na conclusão (parte direita) da regra
- Entretanto, isso geraria um número enorme de regras de associação
  - Para ***m*** itens únicos, existem 2^{m} *itemsets* candidatos (na realidade, 2^{m}-1 *itemsets* se desprezarmos o *itemset* vazio, já que não temos interesse nele)
- Assim, algum critério tem que ser escolhido para diminuir a quantidade de regras

Geração de RA

$$\binom{5}{0} = 1$$

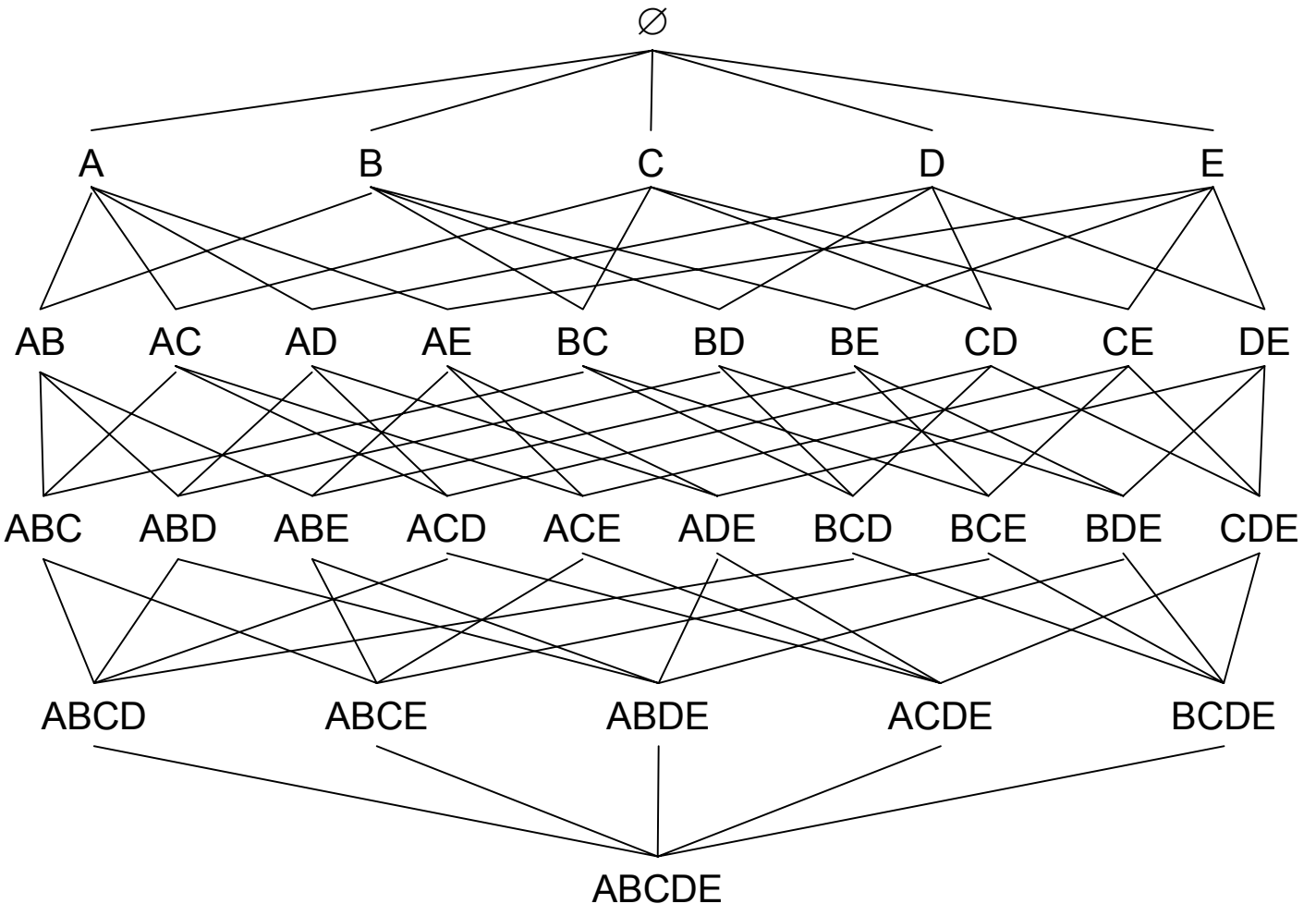
$$\binom{5}{1} = 5$$

$$\binom{5}{2} = 10$$

$$\binom{5}{3} = 10$$

$$\binom{5}{4} = 5$$

$$\binom{5}{5} = 1$$



Geração de RA

- Dados *m* itens (únicos)
  - Há $\binom{m}{r}$ *r-itemsets*
  - Número total de *itemsets*

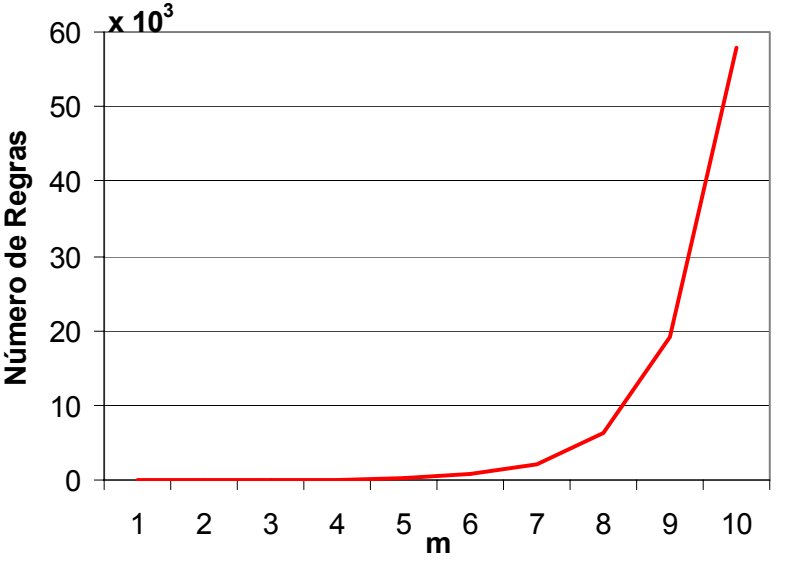
$$= \sum_{r=0}^{m} \binom{m}{r} = 2^m$$

  - Cada *r*-itemset gera 2^r-1 regras em potencial
  - Portanto, o número total de regras de associação é

$$= \sum_{r=1}^{m} \binom{m}{r} \times (2^r - 1)$$

$$= 3^m - 2^m$$

m	Itemsets	Número de Regras
1	2	1
2	4	5
3	8	19
4	16	65
5	32	211
6	64	665
7	128	2059
8	256	6305
9	512	19171
10	1024	58025



Exemplo (Esporte Favorito)

Exemplo	Aparência	Temperatura	Umidade	Ventando	Jogar
z_1	sol	quente	alta	falso	não
z_2	sol	quente	alta	verdadeiro	não
z_3	nublado	quente	alta	falso	sim
z_4	chuva	agradável	alta	falso	sim
z_5	chuva	fria	normal	falso	sim
z_6	chuva	fria	normal	verdadeiro	não
z_7	nublado	fria	normal	verdadeiro	sim
z_8	sol	agradável	alta	falso	não
z_9	sol	fria	normal	falso	sim
z_{10}	chuva	agradável	normal	falso	sim
z_{11}	sol	agradável	normal	verdadeiro	sim
z_{12}	nublado	agradável	alta	verdadeiro	sim
z_{13}	nublado	quente	normal	falso	sim
z_{14}	chuva	agradável	alta	verdadeiro	não

Itemsets	**Tamanho**
1-itemsets	12
2-itemsets	57
3-itemsets	134
4-itemsets	156
5-itemsets	72
Total	**431**

Algoritmo Apriori

- O algoritmo Apriori é o algoritmo de regras de associação mais conhecido e utilizado em diversas aplicações
- Ele pode ser dividido em duas etapas
  - Encontrar os *itemsets* freqüentes (aqueles com suporte ≥ **min_sup**)
  - Gerar as regras a partir dos *itemsets* freqüentes (com confiança ≥ **min_conf**)
- Os valores **min_sup** e **min_conf** são parâmetros que devem ser fornecidos ao algoritmo

Apriori: Ordem dos Itens

- Os itens e *itemsets* são ordenados em ordem lexicográfica (que é uma ordem total)
  - Se $\{i_1, i_2, ..., i_r\}$ representa um *r*-itemset composto pelos itens $i_1, i_2, ..., i_r$ então $i_1 < i_2 < ... < i_r$ de acordo com a ordem lexicográfica
- A ordem é utilizada pelo algoritmo na geração de cada *itemset*

Suporte

- Para reduzir o número de regras evita-se gerar regras que cobrem poucos exemplos
- Para tanto, é definindo um valor mínimo para o suporte (**min_sup**)
- Somente os *itemsets* com suporte ≥ **min_sup** são considerados pelo algoritmo
  - Os *itemsets* com suporte ≥ **min_sup** são denominados ***itemsets* freqüentes** (*large* or *frequent itemsets*)
  - Os *itemsets* com suporte < **min_sup** são ***itemsets* não freqüentes**

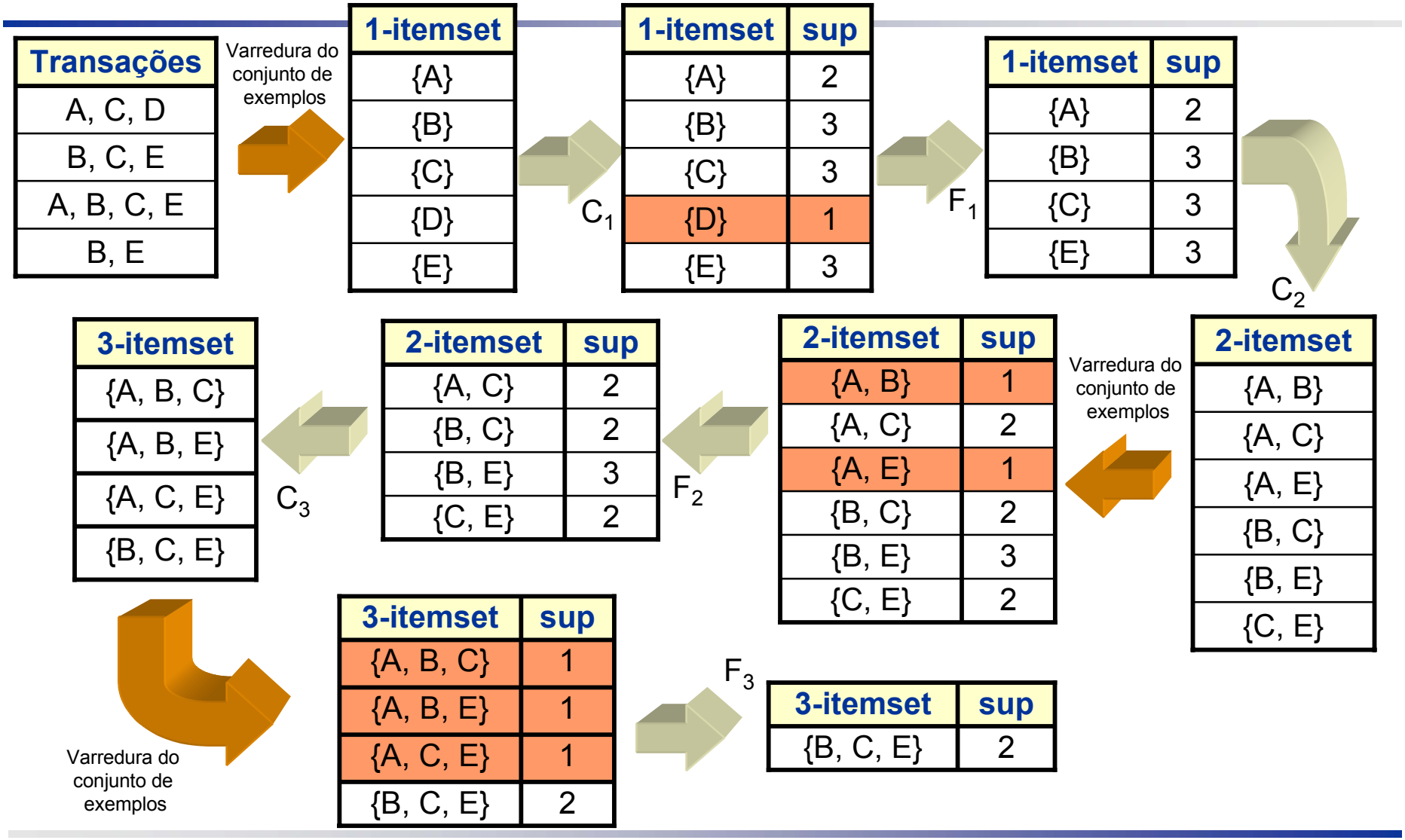
Suporte

- Em RA, o suporte pode ser expresso de forma absoluta ou relativa:
  - Suporte (absoluto): número de exemplos cobertos (corretamente) pelo *itemset*
  - Suporte (relativo): número de exemplos cobertos (corretamente) pelo *itemset* dividido pelo número total de exemplos no conjunto de exemplos de treinamento
  - Não há ambigüidade no uso de ambos:
    - se 0 ≤ suporte ≤ 1 então trata-se de um valor relativo
    - se suporte ≥ 1 e assume valores inteiros (número de exemplos) então trata-se de um valor absoluto
- Vamos ignorar por alguns momentos a distinção entre os lados esquerdo e direito de uma regra e procurar por *itemsets* freqüentes

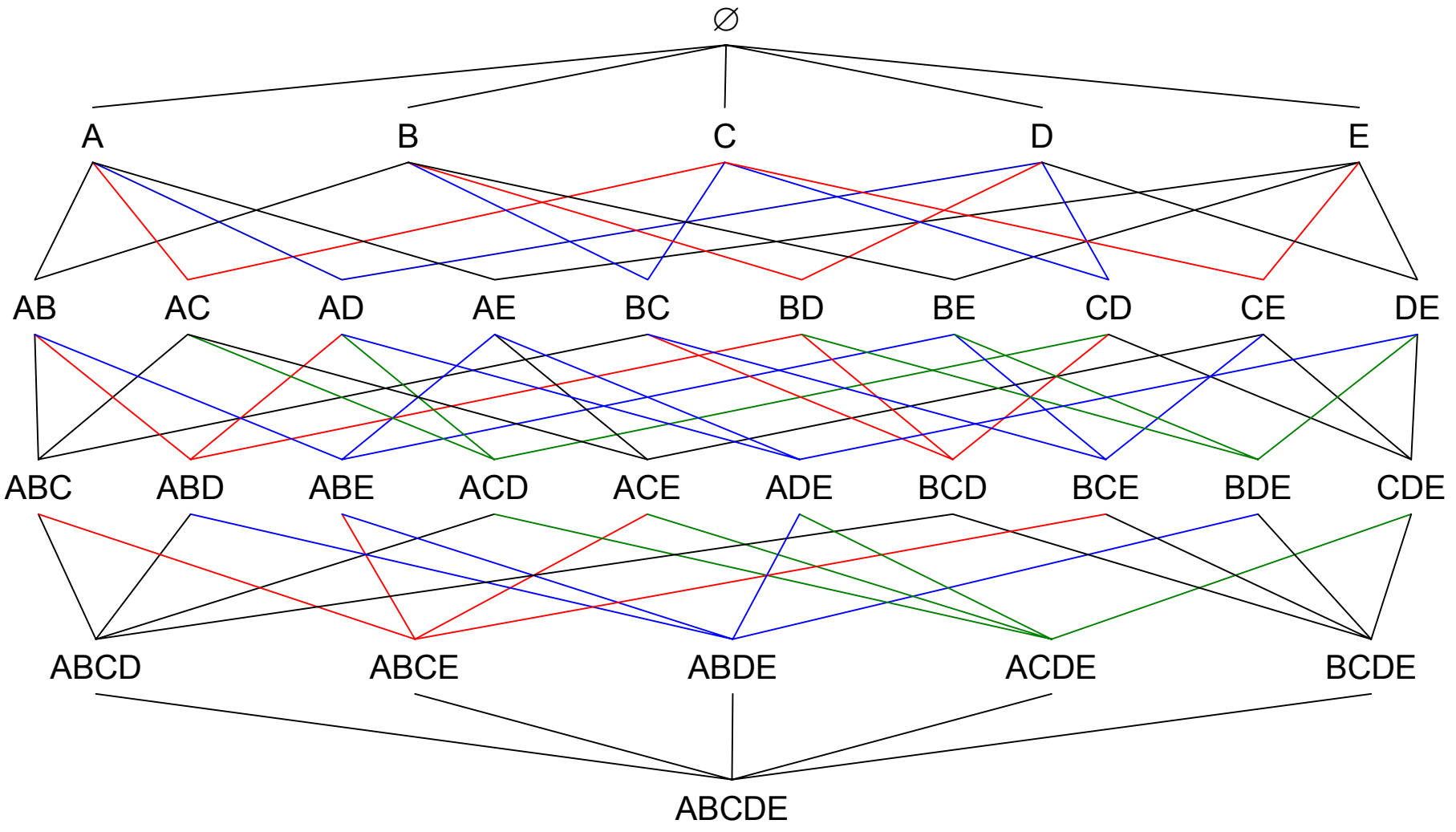
Exemplo, min_sup=2

- Suponha um supermercado, no qual quatro clientes fizeram compras de produtos (itens)
- Os exemplos no formato transacional são
  - Cliente 1 comprou produtos A, C, D
  - Cliente 2 comprou produtos B, C, E
  - Cliente 3 comprou produtos A, B, C, E
  - Cliente 4 comprou produtos B, E
- Vamos gerar os *itemsets* freqüentes, assumindo **min_sup**=2

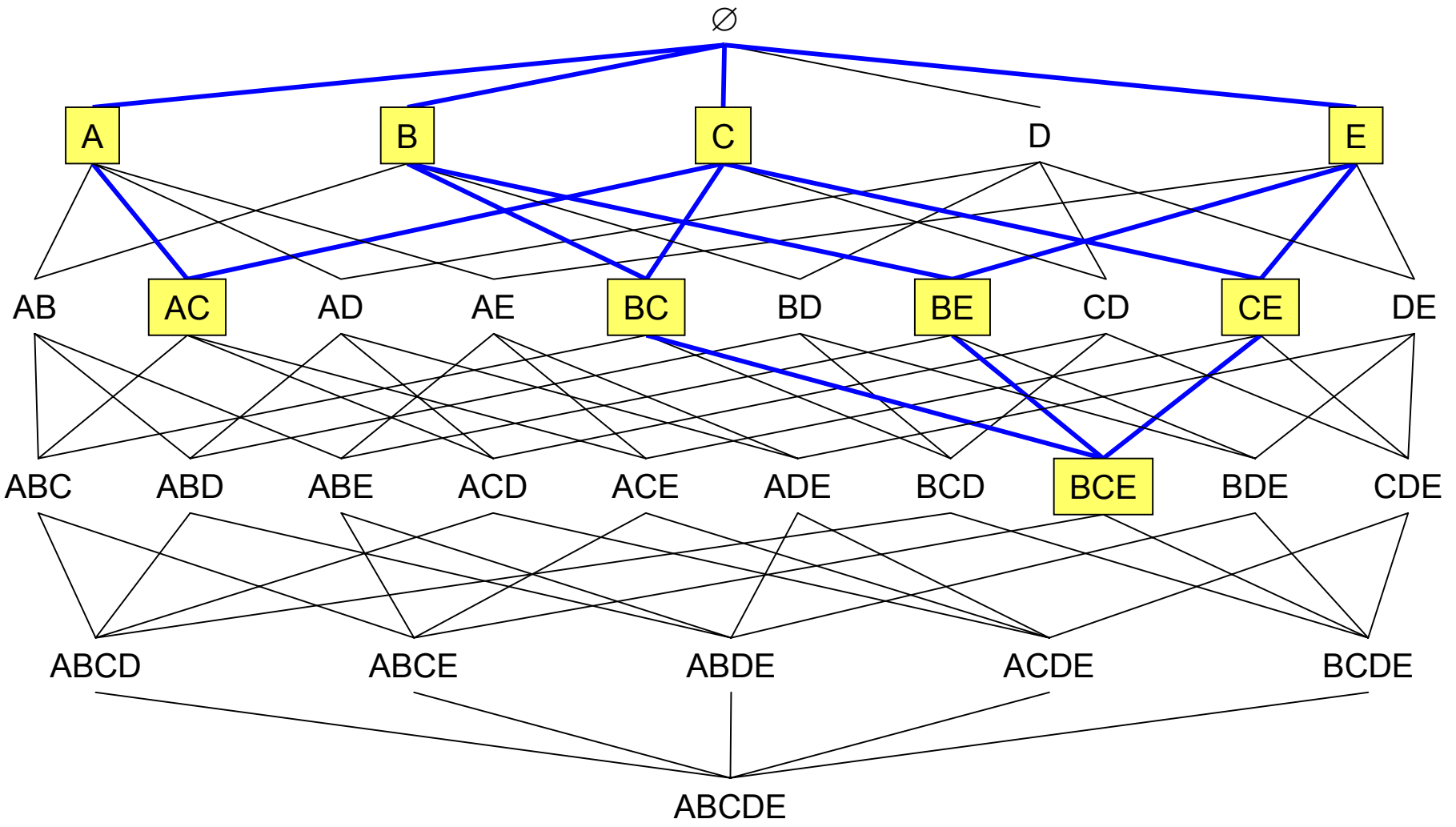
Exemplo, min_sup=2



Exemplo, min_sup=2



Exemplo, min_sup=2



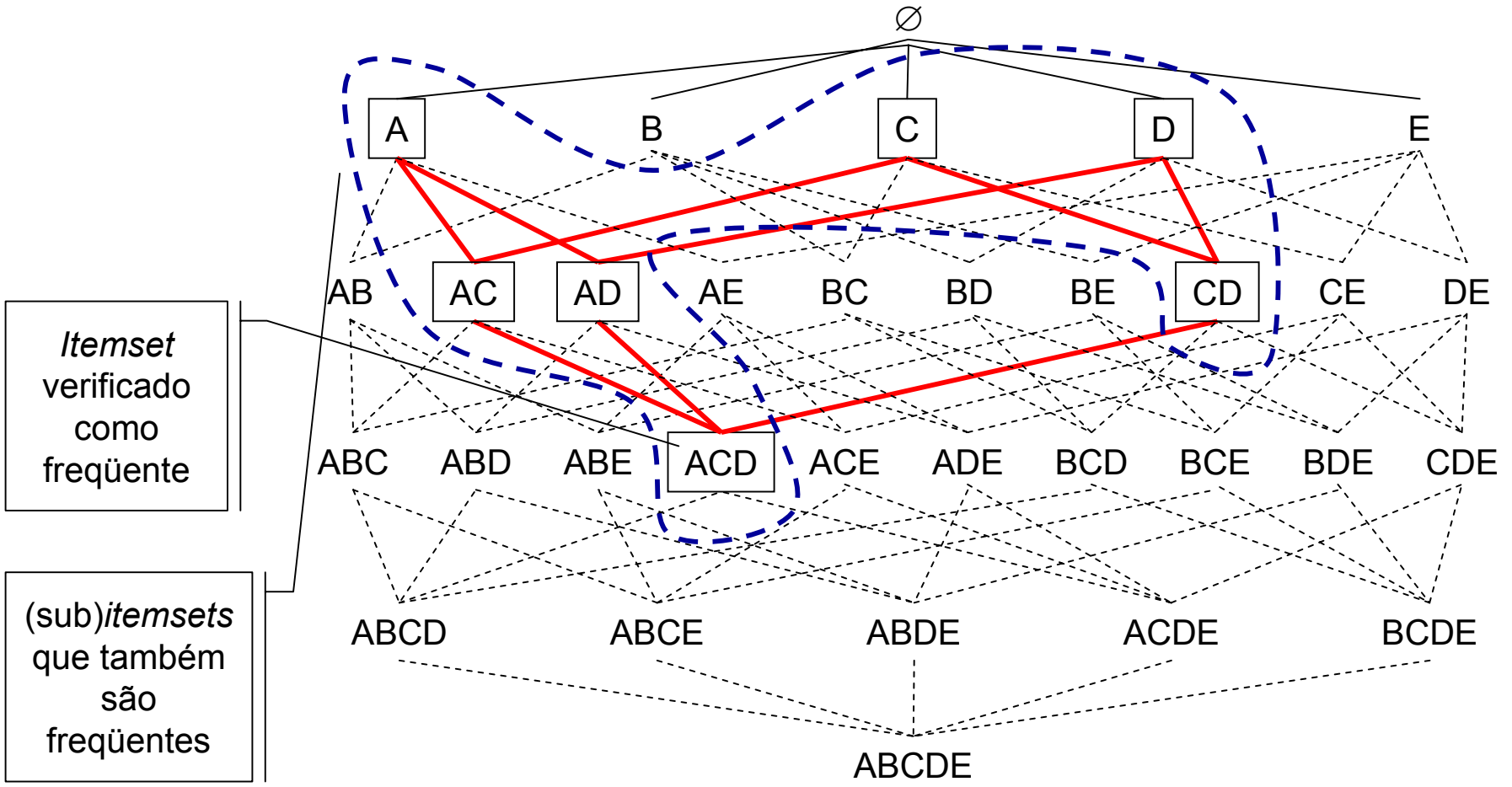
Algoritmo Apriori

- Apriori utiliza o Princípio do *Itemsets* Freqüentes
  - Qualquer subconjunto de um *itemset* freqüente também é freqüente
- Este princípio é válido devido à seguinte propriedade da métrica suporte:
  - $\forall X \, \forall Y : X \subseteq Y \rightarrow \text{suporte}(X) \geq \text{suporte}(Y)$
    - O suporte de um *itemset* Y nunca excede o suporte de seus sub-*itemsets* X
  - Também conhecida como propriedade anti-monotônica do suporte
- O contra-positivo dessa propriedade é que se um *itemset* não é freqüente então não é necessário gerar quaisquer superconjuntos dele como candidatos, porque eles também serão não freqüentes
- Os *itemsets* candidatos gerados ainda precisam ser confirmados se efetivamente são freqüentes, pela varredura do conjunto de exemplos

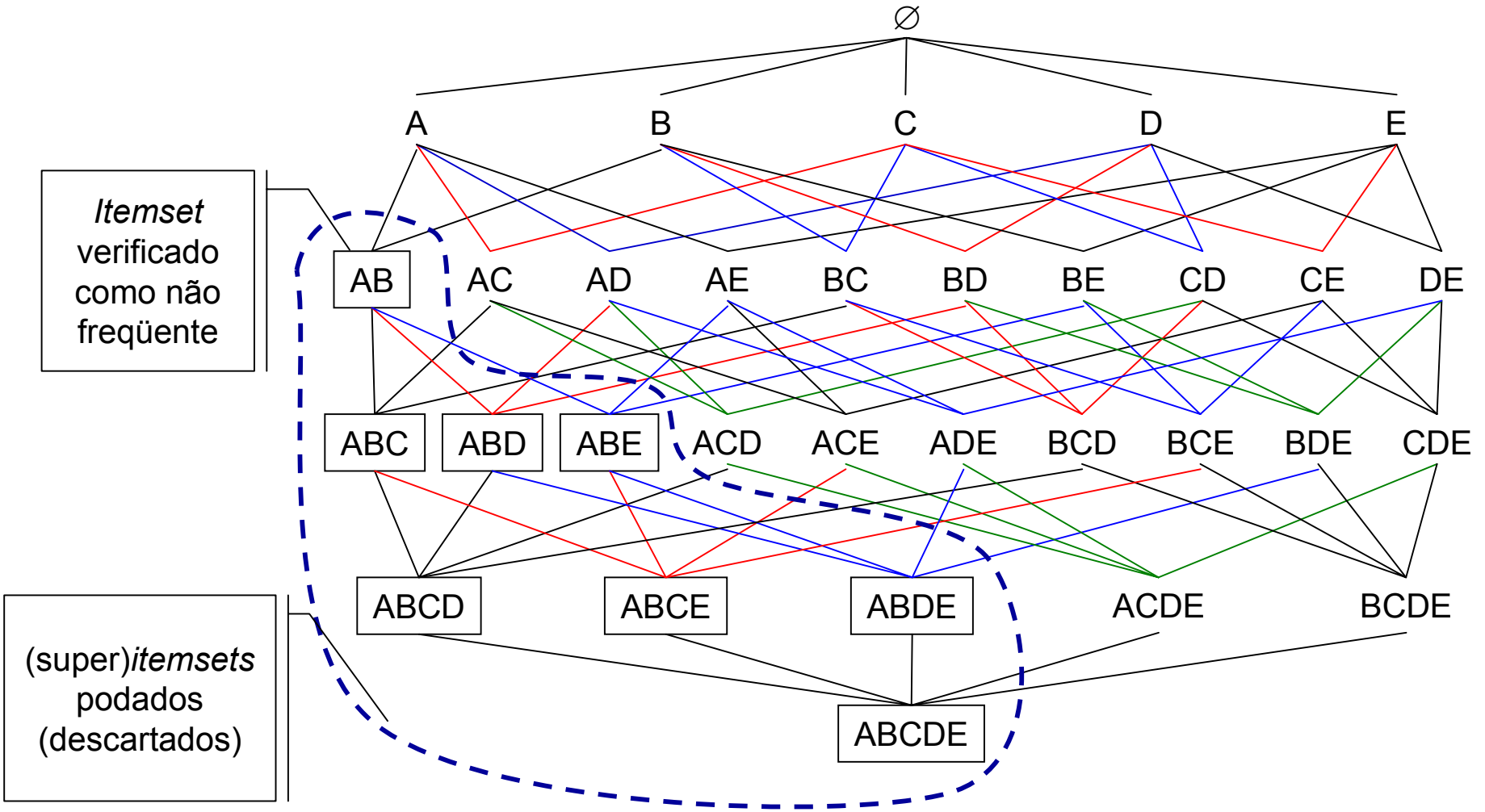
Geração de Itemsets

- Como pode ser observado, os *itemsets* formam um conjunto parcialmente ordenado (reticulado ou *lattice*)
- Assim, na geração de *itemsets* freqüentes é utilizada a seguinte idéia: usar os *1-itemsets* freqüentes para gerar *2-itemsets*, usar os *2-itemsets* freqüentes para gerar *3-itemsets* e assim por diante
  - Se {A,B} é um *itemset* freqüente, então {A} e {B} devem também ser *itemsets* freqüentes
  - Em geral, se X é um ***r**-itemset* freqüente, então todos os (**r**-1)-*itemsets* de X são também freqüentes
  - Assim, a computação de um ***r**-itemset* é feita pela fusão de (**r**-1)-*itemsets*

Exemplo: ACD é freqüente



Exemplo: AB é não freqüente



Exemplo (Esporte Favorito), min_sup=2

Exemplo	Aparência	Temperatura	Umidade	Ventando	Jogar
z_1	sol	quente	alta	falso	não
z_2	sol	quente	alta	verdadeiro	não
z_3	nublado	quente	alta	falso	sim
z_4	chuva	agradável	alta	falso	sim
z_5	chuva	fria	normal	falso	sim
z_6	chuva	fria	normal	verdadeiro	não
z_7	nublado	fria	normal	verdadeiro	sim
z_8	sol	agradável	alta	falso	não
z_9	sol	fria	normal	falso	sim
z_{10}	chuva	agradável	normal	falso	sim
z_{11}	sol	agradável	normal	verdadeiro	sim
z_{12}	nublado	agradável	alta	verdadeiro	sim
z_{13}	nublado	quente	normal	falso	sim
z_{14}	chuva	agradável	alta	verdadeiro	não

Itemsets	**Tamanho**
1-itemsets	12
2-itemsets	47
3-itemsets	39
4-itemsets	6
5-itemsets	0
Total	**104**

1-Itemsets com min_sup=2

1. aparência=sol (5)
2. aparência=nublado (4)
3. aparência=chuva (5)
4. temperatura=fria (4)
5. temperatura=agradável (6)
6. temperatura=quente (4)
7. umidade=normal (7)
8. umidade=alta (7)
9. ventando=verdadeiro (6)
10. ventando=falso (8)
11. jogar=sim (9)
12. jogar=não (5)

2-Itemsets com min_sup=2

1. aparência=sol temperatura=agradável (2)
2. aparência=sol temperatura=quente (2)
3. aparência=sol umidade=normal (2)
4. aparência=sol umidade=alta (3)
5. aparência=sol ventando=verdadeiro (2)
6. aparência=sol ventando=falso (3)
7. aparência=sol jogar=sim (2)
8. aparência=sol jogar=não (3)
9. aparência=nublado temperatura=quente (2)
10. aparência=nublado umidade=normal (2)
11. aparência=nublado umidade=alta (2)
12. aparência=nublado ventando=verdadeiro (2)
13. aparência=nublado ventando=falso (2)

...

38. umidade=normal ventando=falso (4)
39. umidade=normal jogar=sim (6)
40. umidade=alta ventando=verdadeiro (3)

...

47. ventando=verdadeiro jogar=não (2)

3-Itemsets com min_sup=2

1. aparência=sol temperatura=quente umidade=alta (2)
2. aparência=sol temperatura=quente jogar=não (2)
3. aparência=sol umidade=normal jogar=sim (2)
4. aparência=sol umidade=alta ventando=falso (2)
5. aparência=sol umidade=alta jogar=não (3)
6. aparência=sol ventando=falso jogar=não (2)
7. aparência=nublado temperatura=quente ventando=falso (2)
8. aparência=nublado temperatura=quente jogar=sim (2)
9. aparência=nublado umidade=normal jogar=sim (2)
10. aparência=nublado umidade=alta jogar=sim (2)
11. aparência=nublado ventando=verdadeiro jogar=sim (2)
12. aparência=nublado ventando=falso jogar=sim (2)
13. aparência=chuva temperatura=fria umidade=normal (2)

...

38. umidade=normal ventando=falso jogar=sim (4)
39. umidade=alta ventando=falso jogar=não (2)

4-Itemsets com min_sup=2

1. aparência=sol
   temperatura=quente
   umidade=alta
   jogar=não (2)
2. aparência=sol
   umidade=alta
   ventando=falso
   jogar=não (2)
3. aparência=nublado
   temperatura=quente
   ventando=falso
   jogar=sim (2)
4. aparência=chuva
   temperatura=agradável
   ventando=falso
   jogar=sim (2)
5. aparência=chuva
   umidade=normal
   ventando=falso
   jogar=sim (2)
6. temperatura=fria
   umidade=normal
   ventando=falso
   jogar=sim (2)

Confiança

- Assim que todos os *itemsets* freqüentes foram gerados, o próximo passo consiste em transformar cada *itemset* em uma regra (ou conjunto de regras) com uma precisão (confiança) mínima especificada, denotada por **min_conf**
- Dado um *itemset* somente as regras com confiança ≥ min_conf são geradas
- De forma geral, a confiabilidade positiva (prel) é utilizada como métrica de precisão (confiança)

$$\text{prel}(L \rightarrow R) = p(R \mid L) = \frac{lr}{l}$$

- Para uma dada precisão, alguns *itemsets* podem produzir mais de uma regra; outros nenhuma

Exemplo

- Dado o 3-itemset (nº 38) com suporte=4
  - umidade=normal, ventando=falso, jogar=sim (4)
- Este 3-itemset gera 7 regras em potencial
  - Um *r*-itemset gera 2^r-1 regras em potencial

	Regra em Potencial	prel
1	**if** umidade=normal **and** ventando=falso **then** jogar=sim	4/4 = 100.00%
2	**if** umidade=normal **and** jogar=sim **then** ventando=falso	4/6 = 66.67%
3	**if** ventando=falso **and** jogar=sim **then** umidade=normal	4/6 = 66.67%
4	**if** umidade=normal **then** ventando=falso **and** jogar=sim	4/7 = 57.14%
5	**if** ventando=falso **then** umidade=normal **and** jogar=sim	4/8 = 50.00%
6	**if** jogar=sim **then** umidade=normal **and** ventando=falso	4/9 = 44.44%
7	**if** - **then** umidade=normal **and** ventando=falso **and** jogar=sim	4/14 = 28.57%

Exemplo

- Assumindo **min_conf=100**% apenas a primeira das regras anteriores irá para o conjunto final de regras
- A última regra não tem condições (ou equivalentemente, é sempre verdadeira) e o seu denominador é o número total de exemplos no conjunto de treinamento
- O *slide* seguinte mostra o conjunto final de RAs com **min_sup=2** e **min_conf=100**% para o exemplo de jogar o esporte favorito
- Há 58 regras
  - 3 com suporte=4
  - 5 com suporte=3
  - 50 com suporte=2
  - Apenas 7 têm dois testes na conclusão e nenhuma tem mais de dois

Conjunto Final de RAs

	Regra de Associação			Suporte	Confiança
1	aparencia=nublado	→	jogar=sim	4	100%
2	temperatura=fria	→	umidade=normal	4	100%
3	umidade=normal ventando=falso	→	jogar=sim	4	100%
4	aparencia=sol jogar=nao	→	umidade=alta	3	100%
5	aparencia=sol umidade=alta	→	jogar=nao	3	100%
6	aparencia=chuva jogar=sim	→	ventando=falso	3	100%
7	aparencia=chuva ventando=falso	→	jogar=sim	3	100%
8	temperatura=fria jogar=sim	→	umidade=normal	3	100%
9	aparencia=sol temperatura=quente	→	umidade=alta	2	100%
10	temperatura=quente jogar=nao	→	aparencia=sol	2	100%
11	aparencia=sol temperatura=quente	→	jogar=nao	2	100%
12	aparencia=sol jogar=sim	→	umidade=normal	2	100%
13	aparencia=sol umidade=normal	→	jogar=sim	2	100%
...	...	...	...	...	...
57	temperatura=fria umidade=normal ventando=falso	→	jogar=sim	2	100%
58	temperatura=fria ventando=falso	→	umidade=normal jogar=sim	2	100%

Apriori

- Algoritmo Iterativo: Encontre todos os 1-*itemsets* freqüentes, depois encontre os 2-*itemsets* freqüentes e assim por diante
  - Em cada iteração *k*, apenas considere k-*itemsets* que contenham (*k*-1)-*itemsets* freqüentes
- Encontre itemsets de tamanho 1: F_1 (varredura inicial)
- A partir de $k = 2$
  - C_k = candidatos de tamanho *k*: aqueles *itemsets* de tamanho *k* que podem ser freqüentes, dado F_{k-1}
  - F_k = *itemsets* que realmente são freqüentes, $F_k \subseteq C_k$ (necessita uma varredura do conjunto de exemplos)

Exemplo, min_sup=2

- Varredura inicial de T
  - C_1: {A}:2, {B}:3, {C}:3, {D}:1, {E}:3
  - F_1: {A}:2, {B}:3, {C}:3, {E}:3
  - C_2: {A,B}, {A,C}, {A,E}, {B,C}, {B,D}, {C,E}
- 2a. Varredura de T
  - C_2: {A,B}:1, {A,C}:2, {A,E}:1, {B,C}:2, {B,E}:3, {C,E}:2
  - F_2: {A,C}:2, {B,C}:2, {B,E}:3, {C,E}:2
  - C_3: {B,C,E}
- 3a. Varredura de T
  - C_3: {B, C, E}:2
  - F_3: {B, C, E}

Transações
A, C, D
B, C, E
A, B, C, E
B, E

Apriori

```
Algoritmo Apriori(T)
   C1 ← varredura_inicial(T);
   F1 ← {f | f ∈ C1, f.count ≥ min_sup};
   for (k = 2; Fk-1 ≠ ∅; k++) do
       Ck ← Apriori_gen(Fk-1);
       for cada transação t ∈ T do
           for cada candidato c ∈ Ck do
             if c está contido em t then
                c.count++;
              endif
           next candidato
       next transação
        Fk ← {c ∈ Ck | c.count ≥ min_sup};
   end
return F ← ⋃k Fk;
```

Apriori Gen

```
Function Apriori_gen(F_{k-1})
// Input: conjunto F_{k-1}
// Output: candidatos para F_k
   C_k ← ∅;
   forall f_1, f_2 ∈ F_{k-1}
        com f_1 = {i_1, … , i_{k-2}, i_{k-1}}
        e   f_2 = {i_1, … , i_{k-2}, i'_{k-1}}
        e   i_{k-1} < i'_{k-1} do
        c ← {i_1, …, i_{k-1}, i'_{k-1}};  // juntar f_1 e f_2
        C_k ← C_k ∪ {c};
        for cada (k-1)-subconjunto s de c do
          if (s ∉ F_{k-1}) then
            remova c de C_k; // podar
          endif
        next
   next
return C_k;
```

Considerações Importantes

- Utilizando um único valor para min_sup implica que o algoritmo assume que todos os itens possuem freqüências similares
- Em muitos domínios, isso não é válido, pois alguns itens aparecem muito frequentemente enquanto outros raramente aparecem
- Por exemplo
  - Em um supermercado, clientes compram panelas e frigideiras muito menos frequentemente do que pão e leite
  - Em um hospital, pacientes com doenças de alta complexidade ocorrem com muito mais freqüência do que de baixa complexidade

Considerações Importantes

- Escolha de min_sup
  - Diminuir o valor de min_sup resulta em maior quantidade de *itemsets* freqüentes
  - Isso pode aumentar o número de candidatos e a dimensão máxima dos *itemsets* freqüentes (gerando regras com mais condições/conclusões)
- Número de itens no conjunto de exemplos
  - Quanto mais itens, maior o espaço necessário para armazenar o suporte de cada item
  - Se o número de *itemsets* freqüentes também for grande, o tempo de computação e de I/O também podem aumentar
- Número de transações (tamanho do banco de dados ou conjunto de exemplos)
  - O tempo de execução pode aumentar com o aumento do número de transações, uma vez que o algoritmo Apriori efetua múltiplas varreduras

Considerações Importantes

- Se há uma grande variação na freqüência dos itens, há dois problemas
  - Se min_sup for muito grande as regras que envolvem itens raros não serão encontradas
  - Para encontrar regras que envolvem tanto itens freqüentes como itens raros, min_sup deve ser bem pequeno
    - Isso pode causar uma explosão combinatória já que os itens freqüentes (1-*itemsets*) serão associados entre si de todas as formas possíveis, anulando a poda no reticulado

Resumo

- Duas métricas usualmente utilizadas em RA são suporte e confiança
- Há duas etapas na indução de RA
  - Geração de *itemsets* com **min_sup**
    - gera todos os *1-itemsets* com **min_sup** fornecido e então os usa para gerar os *2-itemsets*, *3-itemsets* e assim por diante
      - Cada operação envolve uma varredura no conjunto de exemplos para contar os itens em cada *itemset*
  - A partir de cada *itemset* determinar as regras que possuem **min_conf**
    - avalia cada *itemset* e gera regras, verificando se elas têm **min_conf** especificada
- Na prática, o tempo de computação necessário depende do **min_sup** especificado, pois ele determina o número de varreduras efetuadas em todo o conjunto de exemplos