Universidade Federal do Acre
Programa de Pós-Graduação em Ciência da Computação - PPgCC

Mineração de Dados

Algoritmos de Extração de RA's

Prof. Dr. Daricélio Soares

Mineração de RA's

→ Recorde que:

Se Sup(X→Y) ≥ SupMin então os itens de X∪Y aparecem com freqüência desejada nas transações da base de dados.

→ Dizemos então que:

O conjunto de itens Z = X∪Y possui suporte mínimo e é chamado um conjunto freqüente.

→ Desta forma, podemos dividir o problema de minerar regras de associação em duas fases.

Fase 1: Encontrar cada conjunto freqüente Z (Sup(Z) ≥ SupMin);

Fase 2: Para cada conjunto freqüente Z, identificar seus possíveis subconjuntos X e Y, de tal forma que:

$Z = X \cup Y$ e $Conf(X \rightarrow Y) \geq ConfMin$

(neste caso, X → Y será uma regra de interesse).

Mineração de RA's

Fase 1: Encontrar cada conjunto freqüente Z (Sup(Z) $\geq$ SupMin);

Fase 2: Para cada conjunto freqüente Z (de tamanho maior ou igual a 2), identificar seus possíveis subconjuntos X e Y, de tal forma que:

$Z = X \cup Y$ e $Conf(X \rightarrow Y) \geq ConfMin$.

Fase 1: Identificação dos conjuntos freqüentes.

$\rightarrow$ É a fase computacionalmente cara.

$\rightarrow$ Para um conjunto de itens de tamanho n, existem 2^n possíveis subconjuntos freqüentes.

$\rightarrow$ Dois algoritmos básicos propostos para esta fase:

- *Apriori*
- *Partition*

Fase 2: Identificação das regras a partir dos conjuntos freqüentes.

Estratégia *Apriori*

O algoritmo *Apriori* considera as seguintes propriedades com o objetivo de diminuir o espaço de busca, ou seja, evitar que todos os 2^n subconjuntos sejam avaliados.

→ Todo subconjunto de um conjunto freqüente é freqüente.
(Se {A,B,C} é freqüente, então {A,B} é freqüente.)

→ Pela contra-positiva: Todo conjunto que contém um subconjunto não freqüente também não é freqüente.
(Se {A,B} não é freqüente, então {A,B,C} não é freqüente.)

Estratégia *Apriori*

• Calcular o suporte de todos os conjuntos de tamanho 1 e, em seguida, eliminar aqueles que não possuem o suporte mínimo.

• Formar todos os possíveis conjuntos de tamanho 2 a partir daqueles de tamanho 1 resultantes do passo anterior. Em seguida, eliminar os novos conjuntos que não possuem o suporte mínimo.

• Repetir o procedimento anterior até que, no k-ésimo passo, nenhum novo conjunto de tamanho k, obtido a partir dos conjuntos de tamanho k-1, tenha suporte mínimo.

Estratégia *Apriori*

TID	Itens Comprados
101	leite, pão, suco
792	leite, suco
1130	leite, pão, ovos
1735	pão, biscoito, café

Considerar SupMin = 50% (2 tuplas)

Sup({leite}) = 3
Sup({pão}) = 3
Sup({suco}) = 2
Sup({ovos}) = 1
Sup({biscoito}) = 1
Sup({café}) = 1

Freqüentes:
{leite}
{pão}
{suco}

Sup({leite, pão}) = 2
Sup({leite, suco}) = 2
Sup({pão, suco}) = 1

Freqüentes:
{leite, pão}
{leite, suco}

Sup({leite, pão, suco}) = 1

Freqüentes:

Estratégia *Apriori*

1. F_1 = conjuntos freqüentes de tamanho 1;
2. k = 1;
3. **enquanto** ($F_k \neq \emptyset$) **faça**
4. k = k + 1;
5. Gerar C_k (todos os candidatos de tamanho k) a partir de F_{k-1};
6. **para cada** transação t pertencente a base de dados **faça**
7. Incrementar o contador associado a todo candidato C_k cujos itens pertençam a t;
8. F_k = todos os candidatos pertencentes a C_k com suporte maior ou igual a SupMin;
9. **fim-enquanto**;
10. Resposta = união de todos os conjuntos F_k;

Observe que, em cada uma das k iterações, o algoritmo *Apriori* percorre toda a base de dados.

Gerar C_k (todos os candidatos de tamanho k) a partir de F_{k-1}:

A estratégia de geração de C_k a partir de F_{k-1} também considera a propriedade de que todo subconjunto de um conjunto freqüente é freqüente. Desta forma, diminui a quantidade de candidatos gerados, eliminando alguns que são garantidamente não freqüentes.

- Considere que, dentro de cada conjunto, os itens estejam ordenados. Então, o conjunto {1,2,3,4} só poderá ser gerado se {1,2,3} e {1,2,4} forem freqüentes.

- Se {1,2,3,4} for gerado em C_k, será podado se possuir algum subconjunto que não seja freqüente.

  O conjunto candidato {1,2,3,4} será eliminado de C_4 se, por exemplo, {2,3,4} não for um conjunto freqüente.

Gerar C_k (todos os candidatos de tamanho k) a partir de F_{k-1}:

A estratégia de geração de C_k a partir de F_{k-1} divide-se então em duas fases: junção e poda.

Junção: Para cada dois conjuntos $\{a_1, a_2, ..., a_{k-1}\}$ e $\{b_1, b_2, ..., b_{k-1}\}$ de F_{k-1}:

Se $(a_1 = b_1) \wedge (a_2 = b_2) \wedge ... \wedge (a_{k-2} = b_{k-2}) \wedge (a_{k-1} < b_{k-1})$ então

gere o candidato $\{a_1, a_2, ..., a_{k-1}, b_{k-1}\}$ em C_k.

Se {1,2,3} e {1,2,4} são conjuntos de F_k, então gerar {1,2,3,4} em C_k

Poda: Para cada conjunto de C_k, eliminar aqueles que possuem um subconjunto não freqüente.

(O conjunto candidato {1,2,3,4} será eliminado de C_k se, por exemplo, {2,3,4} não for um conjunto freqüente.)

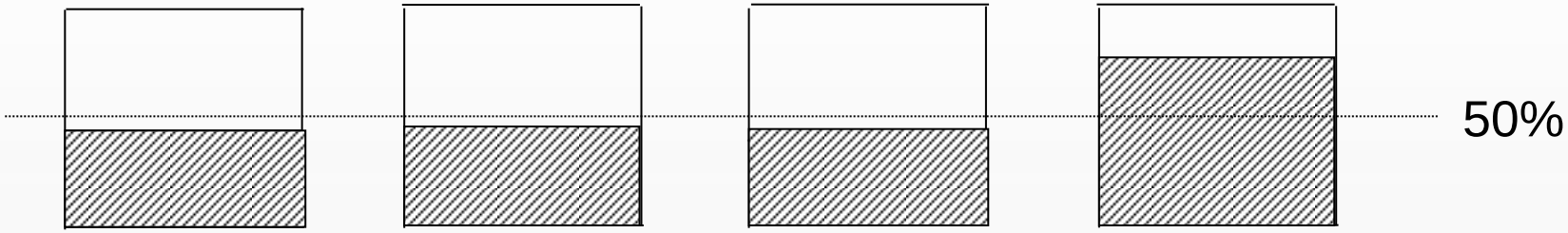
Gerar C_k (todos os candidatos de tamanho k) a partir de F_{k-1};

F_k	Junção	C_k	Poda	C_k (podado)
{A,B,C}	$\Rightarrow$	{A,B,C,D}	$\Rightarrow$	{A,B,C,D}
{A,B,D}		{B,C,D,E}		
{A,C,D}				
{B,C,D}				
{B,C,E}				

Estratégia *Partition*

O algoritmo *Partition* considera a seguinte propriedade com o objetivo de diminuir o número de leituras a toda a base de dados.

→ Considere a base de dados de transações dividida em n partições. Se um conjunto F é freqüente em relação a toda a base de dados (freqüência global), então F é freqüente em relação a pelo menos uma partição (freqüência local), ou seja, possui suporte maior ou igual ao mínimo dentro desta partição.



Estratégia *Partition*

A estratégia *Partition* é dividida em duas fases: na primeira, são gerados os conjuntos candidatos e, na segunda, dentre estes são identificados os freqüentes. Em cada fase é realizada (apenas) uma leitura em toda a base de dados.

Fase I:

→ A base de dados é dividida em partições que caibam na memória principal. Para cada partição, são gerados os conjuntos freqüentes locais, utilizando-se as idéias da estratégia *Apriori*.

→ Desta forma, em um único acesso a toda a base de dados, os conjuntos freqüentes locais de cada partição são gerados. Estes conjuntos são os candidatos a freqüentes globais.

Fase II:

→ Todas as transações da base de dados são percorridas para verificar quais freqüentes locais (candidatos globais) são freqüentes globais.

Apriori x *Partition*

→ Na estratégia *Partition*, a base de dados é lida apenas duas vezes.
Na estratégia *Apriori*, a base de dados é lida em cada uma das k iterações.

→ Se, por um lado, a estratégia *Apriori* realiza um número maior de leituras à base de dados, estas várias leituras permitem que, dentre os conjuntos candidatos, apenas os freqüentes passem à iteração seguinte.

→ Na estratégia *Partition*, passam para a última fase e devem ser processados todos os freqüentes locais (candidatos globais), identificados em cada partição. Este fato, dependendo do número de candidatos gerados que não são de fato freqüentes, pode comprometer o desempenho deste algoritmo.

Questões?