Inteligência Artificial

Técnicas de Mineração de Dados

Árvores de Decisão

Regras de Associação

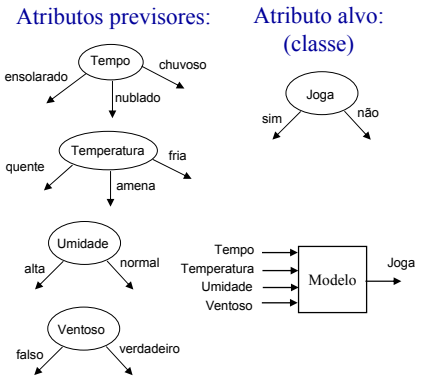
Árvores de Decisão

- As árvores de decisão (AD) são ferramentas poderosas para classificação cuja maior vantagem é a sua expressividade, já que elas representam um conjunto consistente de regras de produção.
- Uma AD representa uma série de perguntas (*testes*) encadeadas acerca de atributos de um objeto do domínio.
- Um dado entra na árvore pelo nó *raiz*, tradicionalmente colocado no topo da representação gráfica, e a árvore se desenvolve para baixo, até chegar a um nó *folha*, representando uma classe.
- A partir de um nó (*pai*), é feito um teste para decidir qual *nó filho* deve ser pesquisado a seguir.
- Existe apenas um caminho da raiz até cada folha. Este caminho é uma expressão da regra usada para classificar os dados.

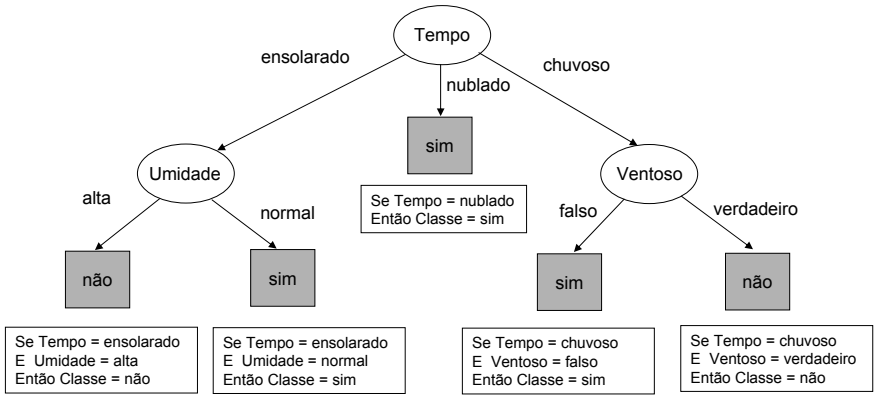
Problema do tempo

- Assumindo que as condições meteorológicas influenciam a decisão de um jogador de jogar ou não (tênis, golfe, ...), determine, a partir de um conjunto de observações passadas, um modelo capaz de prever se o jogador vai ou não jogar num determinado dia, dadas as condições meteorológicas.

Tempo	Temperatura	Umidade	Ventoso	Joga
ensolarado	quente	alta	falso	não
ensolarado	quente	alta	verdadeiro	não
nublado	quente	alta	falso	sim
chuvoso	amena	alta	falso	sim
chuvoso	fria	normal	falso	sim
chuvoso	fria	normal	verdadeiro	não
nublado	fria	normal	verdadeiro	sim
ensolarado	amena	alta	falso	não
ensolarado	fria	normal	falso	sim
chuvoso	amena	normal	falso	sim
ensolarado	amena	normal	verdadeiro	sim
nublado	amena	alta	verdadeiro	sim
nublado	quente	normal	falso	sim
chuvoso	amena	alta	verdadeiro	não



Exemplo de árvore de decisão



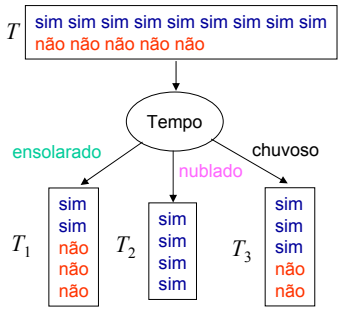
Construção de árvores de decisão

- Procedimento padrão: de cima para baixo, usando recursivamente "dividir e conquistar".
  - Primeiro: *seleciona-se* o atributo do nó raiz e cria-se um ramo para cada valor possível do atributo.
  - Então: os exemplos são *divididos* em subconjuntos (um para cada ramo que sai do nó), conforme o resultado do teste.
  - Finalmente: o procedimento é repetido recursivamente para cada ramo, usando apenas os exemplos que chegam no ramo considerado.
- O processo termina quando todos os exemplos do subconjunto do ramo têm a mesma classe.

Exemplo: após teste do atributo "Tempo"

Tempo	Temperatura	Umidade	Ventoso	Joga
ensolarado	quente	alta	falso	não
ensolarado	quente	alta	verdadeiro	não
nublado	quente	alta	falso	sim
chuvoso	amena	alta	falso	sim
chuvoso	fria	normal	falso	sim
chuvoso	fria	normal	verdadeiro	não
nublado	fria	normal	verdadeiro	sim
ensolarado	amena	alta	falso	não
ensolarado	fria	normal	falso	sim
chuvoso	amena	normal	falso	sim
ensolarado	amena	normal	verdadeiro	sim
nublado	amena	alta	verdadeiro	sim
nublado	quente	normal	falso	sim
chuvoso	amena	alta	verdadeiro	não

conjunto original: 9 "sim", 5 "não"



conjuntos resultantes do teste

Critério para seleção de atributo

- Qual é o melhor atributo para testar num nó?
  - Aquele que produz a menor árvore (implica gerar todas as possibilidades!!).
  - Heurística: escolher o atributo que produz os nós mais "puros" (homogêneos em relação à classe majoritária).
- Critério popular para "impureza": quantidade de informação, ou entropia de um subconjunto.
- O *ganho de informação* representa o quanto se ganha em "pureza" ao se dividir um conjunto segundo um atributo.
- Estratégia: escolher o atributo que produz o maior ganho de informação.

Cálculo da informação

- A quantidade de informação é medida em *bits*.
  - Dada uma distribuição (de classes), calcula-se a quantidade de informação necessária para prever um evento (uma classe).
  - É equivalente à *entropia* da distribuição
  - A entropia representa a informação necessária em bits (ou fração de bits!)
- Fórmula para calcular a entropia de um conjunto:

$$\text{entropia}(p_1, p_2, ..., p_n) = -p_1 \log p_1 - p_2 \log p_2 \ldots - p_1 \log p_n$$

onde $p_1, p_2, ..., p_n$ são as probabilidades (taxa de ocorrência) das classes 1, 2, ..., n, neste conjunto.

- Obs: o logaritmo é na base 2 para a medida em bits!

O algoritmo C4.5

- O algoritmo C4.5 (Quinlan 93) produz árvores com número de ramos variável.
- Cada valor de um dado categórico gera um ramo.
- A entropia ou *ganho de informação* tem prevalecido como fator de escolha do atributo a ser testado num nó.
- A partição do espaço de características começa pelo nó raiz e continua para os nós filhos da mesma maneira, ou seja, escolhendo-se em cada nó o melhor atributo para a partição, até que um atributo assuma um único valor.
- Neste caso, nós rotulamos este nó como folha.

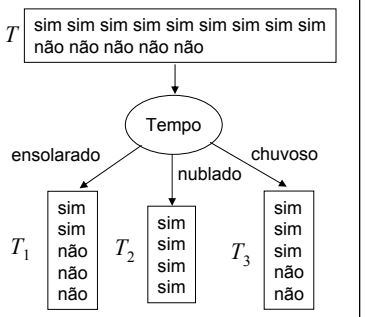
Avaliação dos atributos do tempo

Tempo	Temperatura	Umidade	Ventoso	Joga
ensolarado	quente	alta	falso	não
ensolarado	quente	alta	verdadeiro	não
nublado	quente	alta	falso	sim
chuvoso	amena	alta	falso	sim
chuvoso	fria	normal	falso	sim
chuvoso	fria	normal	verdadeiro	não
nublado	fria	normal	verdadeiro	sim
ensolarado	amena	alta	falso	não
ensolarado	fria	normal	falso	sim
chuvoso	amena	normal	falso	sim
ensolarado	amena	normal	verdadeiro	sim
nublado	amena	alta	verdadeiro	sim
nublado	quente	normal	falso	sim
chuvoso	amena	alta	verdadeiro	não

Qual é o ganho de informação do teste para o atributo "Tempo"?

Exemplo: atributo "Tempo"

conjunto original: 9 "sim", 5 "não"



conjuntos resultantes do teste

Ganho de informação do teste para o atributo "Tempo":

0,940 – 0,693 = 0,247 bits

entropia(p_1,..., p_n) = $-p_1\log p_1 - p_2\log p_2 \ldots - p_1\log p_n$

Entropia do conjunto original:

entropia(9/14, 5/14) = –9/14log(9/14) –5/14log(5/14) = **0,940 bits**

Entropia dos conjuntos resultantes do teste:

Tempo = ensolarado:

entropia(2/5, 3/5) = –2/5log(2/5) –3/5log(3/5) = = **0,971 bits**

Tempo = nublado:

entropia(4/4, 0/4) = –1log(1) –0log(0) = **0 bits**

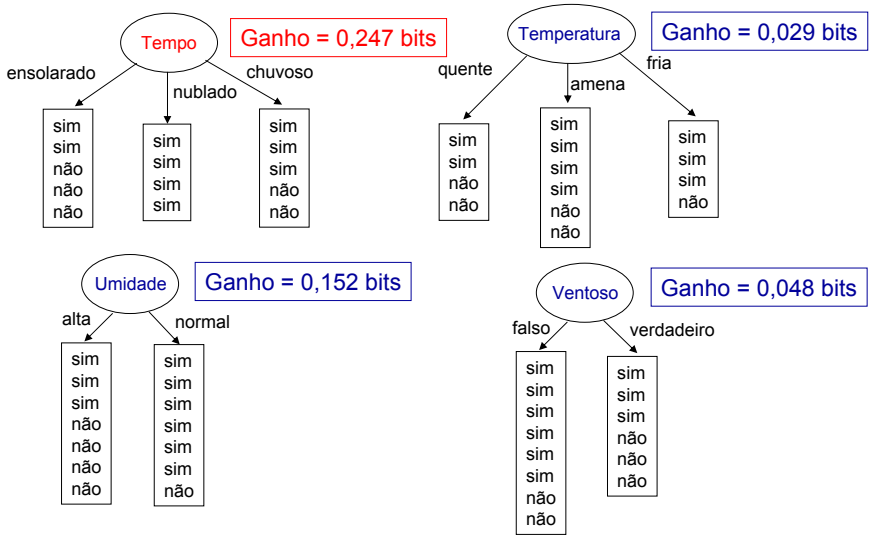
Tempo = chuvoso:

entropia(3/5, 2/5) = –3/5log(3/5) –2/5log(2/5) = = **0,971 bits**

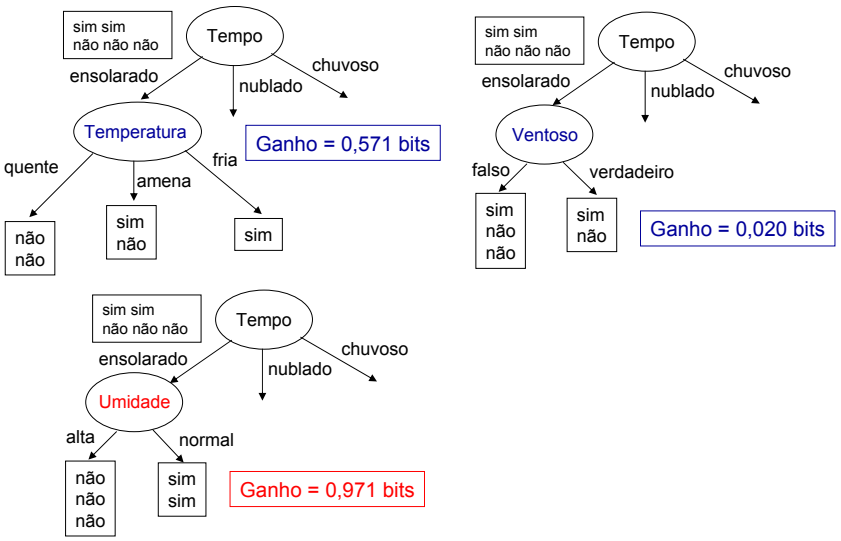
Entropia média dos conjuntos resultantes:

(5/14)×0,971 + (4/14)×0 + (5/14)×0,971 = **0,693 bits**

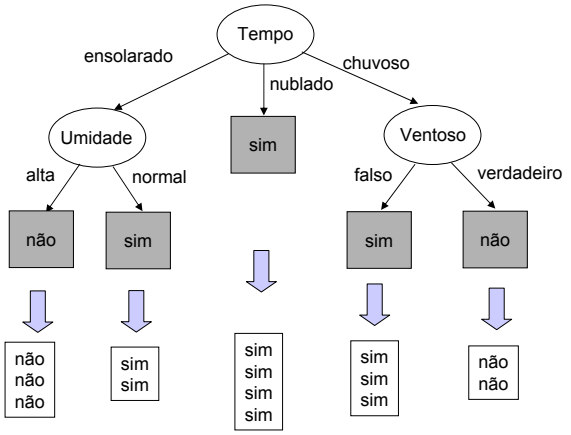
Qual atributo deve ser selecionado?



Continuar a dividir



Árvore de decisão final



Distribuição das classes dos exemplos que chegam nas folhas

Regras de Associação

- O objetivo da Descoberta de Regras Associativas (DRA) é a identificação de associações entre atributos que *freqüentemente* ocorrem juntos nos dados, com alta *confiança*.
- Se um subconjunto de atributos (*itens*) está presente numa instância, então pode-se prever a presença de um outro subconjunto de atributos.
- Esta tarefa está diretamente relacionada ao chamado problema da *cesta de compras* (PCC), que visa analisar um conjunto de transações equivalentes a cestas de compras de um supermercado, procurando encontrar associações entre os *itens* que são freqüentemente comprados juntos.

Definições

- A associação entre subconjuntos de itens freqüentes é representada como uma regra associativa, do tipo:
  "Se item X aparece na cesta, Então item Y também aparece na cesta", ou ainda, $X \Rightarrow Y$.
- A capacidade preditiva desta regra, ou *confiança* da regra, pode ser estimada utilizando a teoria das probabilidades (TP).
- Na TP, a probabilidade condicional $P(Y|X)$ corresponde à probabilidade de Y ocorrer dado que X tenha ocorrido, correspondendo à *confiança* da regra $X \Rightarrow Y$.
- Além disso, procuramos regras associativas envolvendo apenas conjuntos *freqüentes* de itens $\{X_1, X_2, ..., X_n, Y\}$, ou seja, que apareçam numa percentagem mínima de cestas, chamada de *limiar de suporte*.

Probabilidades Condicionais

- A probabilidade condicional $P(B|A)$ é a probabilidade de B ocorrer dado que A tenha ocorrido.
- Ela é calculada a partir da probabilidade conjunta $P(A,B)$:

$$P(B|A) = \frac{P(A,B)}{P(A)}$$

- Esta probabilidade pode ser interpretada como um *fator de confiança* que se pode inferir a partir dos dados na regra:

$$P(A) \rightarrow P(B)$$

- $P(A)$ e $P(B)$ são as *probabilidades marginais* de A e B.
- Elas são também chamadas de probabilidades a priori destes valores de variáveis.

Objetivos do problema da cesta de compras

- A associação entre subconjuntos de itens freqüentes é representada como uma regra associativa, do tipo:

"Se item X aparece na cesta, Então item Y também aparece na cesta", ou ainda, $X \Rightarrow Y$.

- Este problema pode ser analisado utilizando a teoria das probabilidades (TP).
- Na TP, a probabilidade condicional P(Y|X) corresponde à probabilidade de Y ocorrer dado que X tenha ocorrido, correspondendo à *confiança* da regra.

Mineração de Regras Associativas

- Suponha a seguinte descrição formal do problema:
- Seja $I = \{i_1, i_2, ..., i_m\}$ um conjunto de literais, chamados *itens*, D um conjunto de transações, onde cada transação T é um conjunto de itens tal que $T \subseteq I$.
- Em outras palavras, I é um conjunto de atributos sobre o domínio binário $\{0, 1\}$.
- Uma tupla T da base de dados D é representada pelos atributos com valor 1. A cada transação está associado um identificador *TID*.
- Um *conjunto de itens* é denominado X, com $X \subset I$. Dizemos que uma transação T *contém* um conjunto de itens X, se $X \subseteq T$.

Mineração de Regras Associativas

- Uma *regra associativa* é uma implicação da forma:

$X \Rightarrow Y$, onde $X \subset I$, $Y \subset I$ e $X \cap Y = \varnothing$.

- A regra $X \Rightarrow Y$ é válida no conjunto de transações D com *confiança* c, se c% das transações em D que contêm X, também contêm Y.
- A regra $X \Rightarrow Y$ tem *suporte* s no conjunto de transações D se s% das transações em D contêm $X \cup Y$.
- Dado um conjunto de transações D, o problema de minerar regras associativas consiste em gerar todas as regras associativas que têm um suporte mínimo (*supmin*) e confiança mínima (*confmin*), especificados.

Solução ingênua para minerar Regras Associativas

- O primeiro passo no processo de descoberta de RA é determinar os conjuntos de itens que aparecem freqüentemente juntos em cestas de compras.
- Dado um BD com *N* atributos, representando itens possíveis de serem encontrados numa cesta de compras, existe um número limitado e conhecido de conjuntos de itens que podem aparecer juntos numa cesta: $(2^N - 1)$.
- Uma solução imediata (e ingênua) para o problema de RA é criar $(2^N - 1)$ contadores, um para cada combinação possível de itens, e, *posteriormente*, ler cada registro do BD e incrementar cada contador que é coberto pelo conjunto de itens deste registro.
- No final, é necessário podar os conjuntos cujo contador não satisfaz o limiar mínimo especificado (suporte mínimo).
- Problema: complexidade computacional é exponencial!

Explosão combinatória

- A solução de verificar *a posteriori* se a condição do suporte mínimo foi satisfeita por cada contador só faz sentido para *N* pequeno, pois o número de contadores gerados aumenta exponencialmente.
- Além disso, a experiência mostra que apenas uma pequena fração dos contadores satisfaz a condição de suporte mínimo.

$2^4 - 1 = 15$

$2^5 - 1 = 31$

$2^6 - 1 = 63$

$2^7 - 1 = 127$

:

$2^{30} - 1 \approx 10^9$ (um milhão de combinações → *giga*)

$2^{40} - 1 \approx 10^{12}$ (um bilhão de combinações → *tera*)

$2^{50} - 1 \approx 10^{15}$ (um milhão de bilhões de combinações!)

Exemplo de aplicação do método ingênuo para RA

TID	leite	café	cerveja	pão	manteiga	arroz	feijão
1	não	sim	não	sim	sim	não	não
2	sim	não	sim	sim	sim	não	não
3	não	sim	não	sim	sim	não	não
4	sim	sim	não	sim	sim	não	não
5	não	não	sim	não	não	não	não
6	não	não	não	não	sim	não	não
7	não	não	não	sim	não	não	não
8	não	não	não	não	não	não	sim
9	não	não	não	não	não	sim	sim
10	não	não	não	não	não	sim	não

7 atributos:
$2^7 - 1 = 127$
127 contadores

- Inicialmente são gerados 127 contadores
- A seguir, cada registro é lido e é verificado quais são (todas) as combinações de itens que aparecem juntos na respectiva cesta de compra.
- Os contadores correspondentes a estas combinações são incrementados.
- Poda-se as combinações cujos contadores não atingem o limiar de suporte mínimo.

Extração das tuplas de cada registro

TID	leite	café	cerveja	pão	manteiga	arroz	feijão
1	não	sim	não	sim	sim	não	não
2	sim	não	sim	sim	sim	não	não
3	não	sim	não	sim	sim	não	não
4	sim	sim	não	sim	sim	não	não
5	não	não	sim	não	não	não	não
6	não	não	não	não	sim	não	não
7	não	não	não	sim	não	não	não
8	não	não	não	não	não	não	sim
9	não	não	não	não	não	sim	sim
10	não	não	não	não	não	sim	não

TID	leite	café	cerveja	pão	manteiga	arroz	feijão
1	0	1	0	1	1	0	0
2	1	0	1	1	1	0	0
3	0	1	0	1	1	0	0
4	1	1	0	1	1	0	0
5	0	0	1	0	0	0	0
6	0	0	0	0	1	0	0
7	0	0	0	1	0	0	0
8	0	0	0	0	0	0	1
9	0	0	0	0	0	1	1
10	0	0	0	0	0	1	0

TID	tuplas
1	{café,pão,manteiga}
2	{leite,cerveja,pão,manteiga}
3	{café,pão,manteiga}
4	{leite,café,pão,manteiga}
5	{cerveja}
6	{manteiga}
7	{pão}
8	{feijão}
9	{arroz,feijão}
10	{arroz}

Geração das combinações cobertas por cada registro

Considere que o suporte mínimo seja 3

TID	tupla
1	{café,pão,manteiga}
2	{leite,cerveja,pão,manteiga}
3	{café,pão,manteiga}
4	{leite,café,pão,manteiga}
5	{cerveja}
6	{manteiga}
7	{pão}
8	{feijão}
9	{arroz,feijão}
10	{arroz}

TID	itens	duplas	trincas	quadras
1	{café} {pão} {manteiga}	{café,pão} {café,manteiga} {pão,manteiga}	{café,pão,manteiga}	
2	{leite} {cerveja} {pão} {manteiga}	{leite,cerveja} {leite,pão} {leite,manteiga} {cerveja,pão} {cerveja,manteiga} {pão,manteiga}	{leite,cerveja,pão} {leite,cerveja,manteiga} {leite,pão,manteiga} {cerveja,pão,manteiga}	{leite,cerveja,pão,manteiga}
3	{café} {pão} {manteiga}	{café,pão} {café,manteiga} {pão,manteiga}	{café,pão,manteiga}	
4	{leite} {café} {pão} {manteiga}	{leite,café} {leite,pão} {leite,manteiga} {café,pão} {café,manteiga} {pão,manteiga}	{leite,café,pão} {leite,café,manteiga} {leite,pão,manteiga} {café,pão,manteiga}	{leite,café,pão,manteiga}
5	{cerveja}			
6	{manteiga}			
7	{pão}			
8	{feijão}			
9	{arroz} {feijão}	{arroz,feijão}		
10	{arroz}			

Combinações possíveis (127):

7 itens
21 pares
35 trincas
35 quadras
21 quinas
7 senas
1 setena

Combinações cobertas pelo BD (26):

7 itens / 3 com suporte mínimo
10 pares / 3 com suporte mínimo
7 trincas / 1 com suporte mínimo
2 quadras / nenhuma com suporte mínimo

Conjunto de itens	suporte
{café}	3
{pão}	5
{manteiga}	5

Conjunto de itens	suporte
{café, pão}	3
{café, manteiga}	3
{pão, manteiga}	4

Conjunto de itens	suporte
{café, pão, manteiga}	3

Solução mais eficiente: Algoritmo Apriori

- Geração (ordenada) *a priori* das combinações cobertas pelo BD, que satisfazem o suporte mínimo

C_1

Conjunto de itens	suporte
{leite}	2
{café}	3
{cerveja}	2
{pão}	5
{manteiga}	5
{arroz}	2
{feijão}	2

L_1

Conjunto de itens	suporte
{café}	3
{pão}	5
{manteiga}	5

$C_2 = L_2$

Conjunto de itens	suporte
{café, pão}	3
{café, manteiga}	3
{pão, manteiga}	4

$C_3 = L_3$

Conjunto de itens	suporte
{café, pão, manteiga}	3

Descoberta de grandes conjuntos de itens

- Decompor o problema em dois subproblemas:

(1) Encontrar todas as combinações de itens com suporte maior ou igual ao mínimo. Estas combinações são chamadas de *grandes conjuntos de itens* e todas as outras combinações são chamadas de *pequenos conjuntos de itens*.

(2) Utilizar os grandes conjuntos de itens para gerar as regras desejadas.

A idéia é que se $ABCD$ e AB são grandes conjuntos de itens, então nós podemos determinar se a regra $AB \Rightarrow CD$ é válida calculando a razão $r = suporte(ABCD) / suporte(AB)$.

A regra será válida somente se $r \geq confmin$.

Note que a regra terá suporte mínimo porque $ABCD$ é grande.

Algoritmo Apriori

(1) Encontre os itens que aparecem ao menos numa fração das cestas igual a *supmin*. Este conjunto é chamado L_1, dos itens freqüentes (grande conjunto de itens).

(2) Os pares dos itens em L_1 se tornam *pares candidatos* C_2 para o segundo passo. Os pares em C_2 cuja contagem alcançar *supmin* são os pares freqüentes L_2.

(3) As trincas candidatas C_3 são aqueles conjuntos $\{A, B, C\}$ tais que todos os $\{A, B\}$, $\{A, C\}$ e $\{B, C\}$ estão em L_2. No terceiro passo, conte a ocorrência das trincas em C_3; aquelas cuja contagem alcançar *supmin* são as trincas freqüentes, L_3.

(4) Proceda da mesma forma para tuplas de ordem mais elevada, até os conjuntos se tornarem vazios. L_i são os conjuntos freqüentes de tamanho i; C_{i+1} é o conjunto de tamanho i+1 tal que cada subconjunto de tamanho i está em L_i.

Exemplo de descoberta de regras associativas

Dada a tabela abaixo onde cada registro corresponde a uma transação de um cliente, com itens assumindo valores binários (sim/não), indicando se o cliente comprou ou não o respectivo item, descobrir todas as regras associativas, determinando o seu suporte (*sup*) e grau de certeza (*conf*).

TID	leite	café	cerveja	pão	manteiga	arroz	feijão
1	não	sim	não	sim	sim	não	não
2	sim	não	sim	sim	sim	não	não
3	não	sim	não	sim	sim	não	não
4	sim	sim	não	sim	sim	não	não
5	não	não	sim	não	não	não	não
6	não	não	não	não	sim	não	não
7	não	não	não	sim	não	não	não
8	não	não	não	não	não	não	sim
9	não	não	não	não	não	sim	sim
10	não	não	não	não	não	sim	não

- Dada uma regra de associação "Se X então Y", os fatores *sup* e *conf* são:

$$sup = \frac{\text{Número de registros com } X \text{ e } Y}{\text{Número total de registros}}$$

$$conf = \frac{\text{Número de registros com } X \text{ e } Y}{\text{Número de registros com } X}$$

(1) Calcular o suporte de conjuntos com um item.
Determinar os itens freqüentes com *sup* $\geq 0{,}3$.

(2) Calcular o suporte de conjuntos com dois itens.
Determinar conjuntos de itens freqüentes com *sup* $\geq 0{,}3$.
Obs: se um item não é freqüente em (1), pode ser ignorado aqui.
Descobrir as regras com alto fator de certeza.

(3) Calcular o suporte de conjuntos com três itens.
Determinar conjuntos de itens freqüentes com *sup* $\geq 0{,}3$.
Obs: pelo mesmo motivo anterior, só é necessário se considerar conjuntos de itens que são freqüentes pelo passo anterior.
Descobrir regras com alto fator de certeza.

C_1

Conjunto de itens	suporte
{leite}	2
{café}	3
{cerveja}	2
{pão}	5
{manteiga}	5
{arroz}	2
{feijão}	2

Conjunto de itens	suporte
{café}	3
{pão}	5
{manteiga}	5

L_1

Conjunto de itens	suporte
{café}	3
{pão}	5
{manteiga}	5

L_1

Conjunto de itens	suporte
{café, pão}	3
{café, manteiga}	3
{pão, manteiga}	4

C_2, L_2

Conjunto de itens	suporte
{café, pão, manteiga}	3

C_3, L_3

Geração de RA a partir de grandes conjuntos de tuplas

Se $\{X,Y\}$ é um grande conjunto de itens, então testar as RA possíveis

$X \rightarrow Y$, $conf_{X \rightarrow Y} = \frac{|X \cap Y|}{|X|}$

$Y \rightarrow X$, $conf_{Y \rightarrow X} = \frac{|X \cap Y|}{|Y|}$

Escolher a RA com a maior *conf*

Conjunto de itens: {café, pão}, |café,pão| = 3, |café| = 3, |pão| = 5

Se café **Então** pão *conf* = 1,0

Se pão **Então** café *conf* = 0,6

Conjunto de itens: {café, manteiga}, |café,manteiga| = 3, |café| = 3, |manteiga| = 5

Se café **Então** manteiga *conf* = 1,0

Se manteiga **Então** café *conf* = 0,6

Conjunto de itens: {pão, manteiga}, |pão,manteiga| = 4, |pão| = 5, |manteiga| = 5

Se pão **Então** manteiga *conf* = 0,8

Se manteiga **Então** pão *conf* = 0,8

- Regras candidatas com três itens com o seu valor de certeza:

Conjunto de itens: {café, pão, manteiga}, |café,pão,manteiga| = 3

Se café, pão **Então** manteiga *conf* = 1,0

Se café, manteiga **Então** pão *conf* = 1,0

Se manteiga, pão **Então** café *conf* = 0,75

Se café **Então** manteiga, pão *conf* = 1,0

Se manteiga **Então** café, pão *conf* = 0,6

Se pão **Então** café, manteiga *conf* = 0,6

- Padrões descobertos, *minsup* = 0,3 e *minconf* = 0,8:

Se café **Então** pão *conf* = 1,0

Se café **Então** manteiga *conf* = 1,0

Se pão **Então** manteiga *conf* = 0,8

Se manteiga **Então** pão *conf* = 0,8

decisão: aumentar *confmin*

Se café, manteiga **Então** pão *conf* = 1,0

Se café, pão **Então** manteiga *conf* = 1,0

Se café **Então** manteiga, pão *conf* = 1,0