RObust Clustering using linKs
ROCK

Thiago F. Covões

SCC5895-Análise de Agrupamento de Dados

Sumário

- Motivação
- *Links*
- Função de qualidade
- Algoritmo
- Vantagens/Desvantagens

Motivação

- Atributos categóricos/nominais/discretos
  - Caso específico: booleanos
- *Market basket analysis*
  - <Leite, Manteiga, Pão>
  - <Leite, Bolacha, Suco>

Motivação

- Porque um novo algoritmo?
  - Algoritmos baseados em distância euclidiana não são interessantes
  - Medida de similaridade:
    - Coeficiente de Jaccard

$$similaridade(T_i, T_j) = \frac{|T_i \cap T_j|}{|T_i \cup T_j|}$$

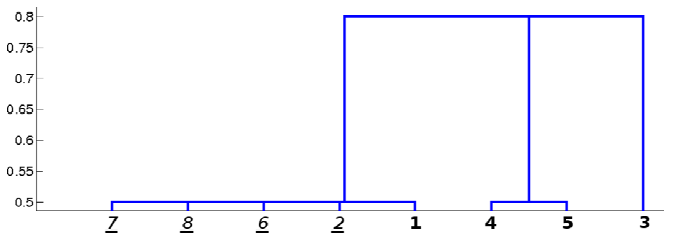
  - Grupos com sobreposição de items

Motivação

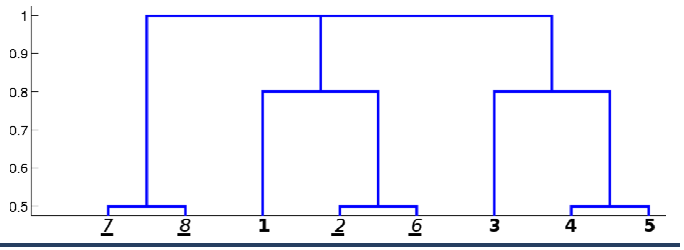
- Exemplo:
  - T_1 = {Pão, Refrigerante, Sal Grosso};
  - T_2 = {Pão, Refrigerante, Fralda};
  - T_3 = {Pão, Cerveja, Picanha};
  - T_4 = {Sal Grosso, Picanha, Farofa};
  - T_5 = {Sal Grosso, Picanha, Maminha};
  - T_6 = {Refrigerante, Fralda, Hipoglós};
  - T_7 = {Fralda, Hipóglos, Lenço Umedecido};
  - T_8 = {Hipoglós, Lenço Umedecido, Papinha};

Motivação

- Single



- Complete



Vizinhos

- Como distinguir entre grupos quando há forte sobreposição?
  - Considerar os *vizinhos* em comum entre dois objetos

$$Vizinhos(T_i) = \{T_j \mid similaridade(T_i, T_j) \geq \theta\}$$

  - Definindo Θ
    - O coeficiente de Jaccard pode assumir um número finito de valores:
      - Min($|T_1|$, $|T_2|$) + 1

Vizinhos

- No exemplo:
  - Todas as transações têm 3 items
  - Valores possíveis:
    - 0/6
    - 1/5
    - 2/4
    - 3/3

$$similaridade(T_i, T_j) = \frac{|T_i \cap T_j|}{|T_i \cup T_j|}$$

Vizinhos de T_1 Θ=0,2 (1/5)

- Exemplo:
  - T_1 = {Pão, Refrigerante, Sal Grosso};
  - T_2 = {Pão, Refrigerante, Fralda};
  - T_3 = {Pão, Cerveja, Picanha};
  - T_4 = {Sal Grosso, Picanha, Farofa};
  - T_5 = {Sal Grosso, Picanha, Maminha};
  - T_6 = {Refrigerante, Fralda, Hipoglós};
  - T_7 = {Fralda, Hipóglos, Lenço Umedecido};
  - T_8 = {Hipoglós, Lenço Umedecido, Papinha};

Vizinhos de T_1 Θ=0,2 (1/5)

- Exemplo:
  - T_1 = {**Pão, Refrigerante, Sal Grosso**};
  - T_2 = {**Pão, Refrigerante**, Fralda};
  - T_3 = {**Pão**, Cerveja, Picanha};
  - T_4 = {**Sal Grosso**, Picanha, Farofa};
  - T_5 = {**Sal Grosso**, Picanha, Maminha};
  - T_6 = {**Refrigerante**, Fralda, Hipoglós};
  - T_7 = {Fralda, Hipóglos, Lenço Umedecido};
  - T_8 = {Hipoglós, Lenço Umedecido, Papinha};

Vizinhos de T_1 Θ=0,5 (2/4)

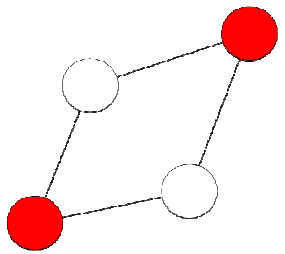
- Exemplo:
  - T_1 = {**Pão, Refrigerante, Sal Grosso**};
  - T_2 = {**Pão, Refrigerante**, Fralda};
  - T_3 = {Pão, Cerveja, Picanha};
  - T_4 = {Sal Grosso, Picanha, Farofa};
  - T_5 = {Sal Grosso, Picanha, Maminha};
  - T_6 = {Refrigerante, Fralda, Hipoglós};
  - T_7 = {Fralda, Hipóglos, Lenço Umedecido};
  - T_8 = {Hipoglós, Lenço Umedecido, Papinha};

Links

- *link*(T_i, T_j) é o número de vizinhos comuns a T_i e T_j

$$link(T_i,T_j) = | Vizinhos(T_i) \cap Vizinhos(T_j) - \{T_i,T_j\} |$$

- Grafo de vizinhos:
  - Número de caminhos de tamanho 2 distintos



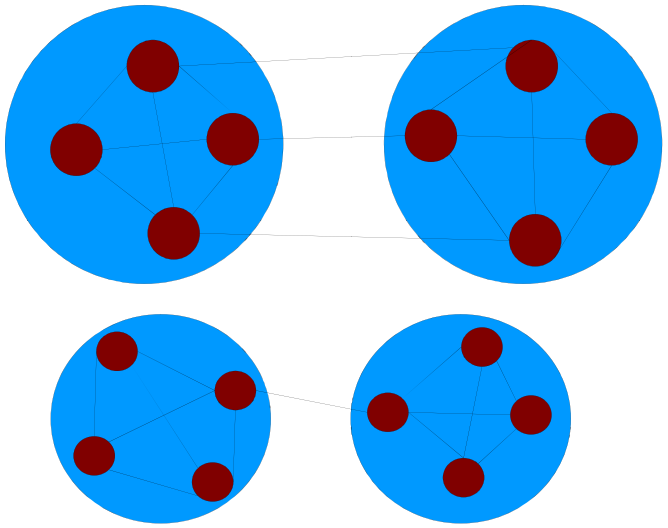
link(T_1,T_2) ; Θ=0,2

- Exemplo:
  - T_1 = {Pão, Refrigerante, Sal Grosso};
  - T_2 = {Pão, Refrigerante, Fralda};
  - T_3 = {Pão, Cerveja, Picanha};
  - T_4 = {Sal Grosso, Picanha, Farofa};
  - T_5 = {Sal Grosso, Picanha, Maminha};
  - T_6 = {Refrigerante, Fralda, Hipoglós};
  - T_7 = {Fralda, Hipóglos, Lenço Umedecido};
  - T_8 = {Hipoglós, Lenço Umedecido, Papinha};

ROCK

- Algoritmo hierárquico
  - Aglomerativo
  - Utiliza *links* no lugar das proximidades
  - Como definir qual par de grupos deve ser unido?

Links Cruzados



Função de qualidade

- Número de *links* cruzados entre dois grupos

$$link[C_i, C_j] = \sum_{T_q \in C_i, T_r \in C_j} link(T_q, T_r)$$

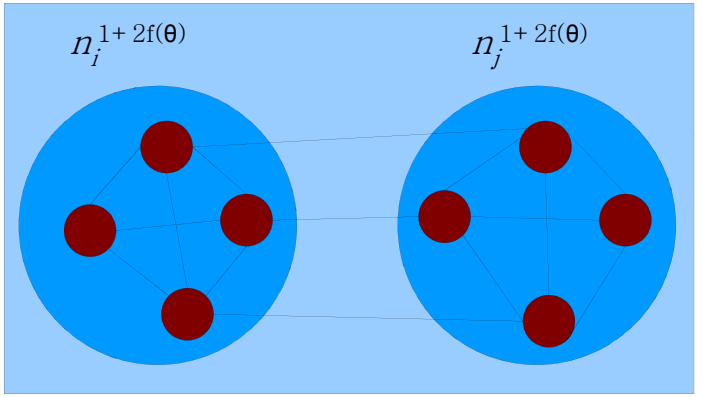
- Grupos *maiores* dominam
- Necessário normalizar

Função de qualidade

- Número **esperado** de *links* em um grupo
  - Assumindo que existe uma função f(θ) tal que o número de vizinhos de cada objeto do grupo C_i é aproximadamente $n_i^{f(\theta)}$
    - n_i é o número de objetos no grupo C_i
  - Cada objeto contribui $n_i^{2f(\Theta)}$ *links*
  - Portanto, o esperado são $n_i^{1+2f(\theta)}$ *links*

Função de qualidade

$(n_i + n_j)^{1+2f(\theta)}$



Função de qualidade

- Qualidade da união dos grupos C_i e C_j

$$g(C_i, C_j) = \frac{link[C_i, C_j]}{(n_i + n_j)^{1+2f(\theta)} - n_i^{1+2f(\theta)} - n_j^{1+2f(\theta)}}$$

Função de qualidade

- Como definir f(θ)?
  - Para este tipo de base de dados, uma possível função é:

$$f(\theta) = \frac{1-\theta}{1+\theta}$$

  - Θ=1 → f(Θ)=0 → $n_i^0 = 1$
  - Θ=0 → f(Θ)=1 → $n_i^1 = n_i$

Algoritmo

- Três etapas principais
  - Cálculo dos *links*
  - Inicialização de estruturas auxiliares
  - Laço de união de grupos

Cálculo dos *Links*

- Seja A_{NxN} uma matriz de adjacências onde a_{ij}=1 se T_i e T_j são vizinhos e 0 caso contrário
- Basta calcular L_{NxN}=AA

Inicialização de estrutura auxiliar

- Cada objeto inicia em um grupo
- Para cada grupo é gerada uma **lista local**
  - Uma entrada para cada grupo que possue pelo menos um *link* cruzado
  - O valor da qualidade da união para cada grupo

Inicialização de estrutura auxiliar

- No exemplo (com θ=0,2):
  - *link*(T_1,T_2)=2

$$g(C_1 = \{T_1\}, C_2 = \{T_2\}) = \frac{2}{2^{1+2(0.67)} - 1^{1+2(0.67)} - 1^{1+2(0.67)}} = 0.65$$

  - Lista local do grupo C_1

C_2 - 0.65
C_3 - 0.98
C_4 - 0.65
C_5 - 0.65
C_6 - 0.32
C_7 - 0.65
C_8 - 0.32

Inicialização de estrutura auxiliar

- Após a inicialização das N listas locais
- Inicializa-se uma **lista global**, com o valor da melhor união possível para cada grupo

Inicialização de estrutura auxiliar

- No exemplo (com θ=0,2):

Lista global

C_1 - 0.98
C_2 - 0.65
C_3 - 0.98
C_4 - 0.65
C_5 - 0.65
C_6 - 0.65
C_7 - 0.65
C_8 - 0.65

Lista local de C_1

C_2 - 0.65
C_3 - 0.98
C_4 - 0.65
C_5 - 0.65
C_6 - 0.32
C_7 - 0.65
C_8 - 0.32

Laço de união de grupos

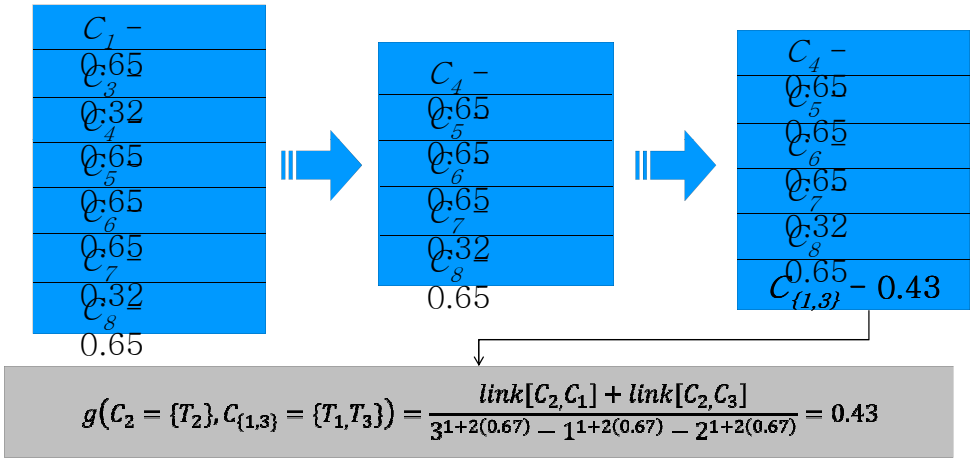
- Seja:
  - C_u = Grupo com maior valor na lista global
  - C_v = Grupo com maior valor na lista local de C_u
- Une C_u e C_v formando C_w
  - Substituir entradas referentes a C_u e C_v em listas locais por C_w
    - Número de *links* entre um grupo e C_w é a soma do número de *links* de C_i a C_u e C_v
  - Uma nova lista local para C_w é criada
  - Lista global é atualizada

Laço de união de grupos

- No exemplo (com θ=0,2):
  - $C_u = C_1 = \{ T_1 \}$; $C_v = C_3 = \{ T_3 \}$
  - $C_w = C_1 \cup C_3 = \{ T_1, T_3 \}$
  - Grupos que tem C_u ou C_v em sua lista local: { C_2, C_4, C_5, C_6, C_7, C_8 }
  - Vamos considerar a atualização em relação ao grupo C_2

Laço de união de grupos

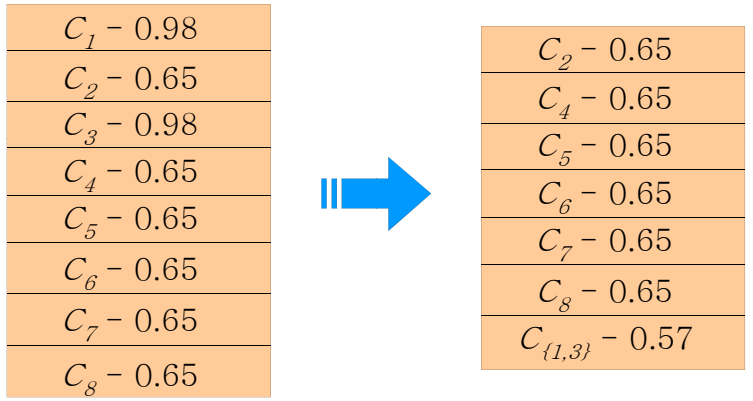
Lista local de C_2



$$g(C_2 = \{T_2\}, C_{\{1,3\}} = \{T_1, T_3\}) = \frac{link[C_2, C_1] + link[C_2, C_3]}{3^{1+2(0.67)} - 1^{1+2(0.67)} - 2^{1+2(0.67)}} = 0.43$$

Laço de união de grupos

Lista global

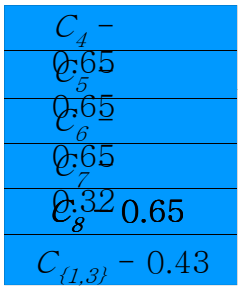


Laço de união de grupos

Lista global

C_2 – 0.65
C_4 – 0.65
C_5 – 0.65
C_6 – 0.65
C_7 – 0.65
C_8 – 0.65
$C_{\{1,3\}}$ – 0.57

Lista local de C_2

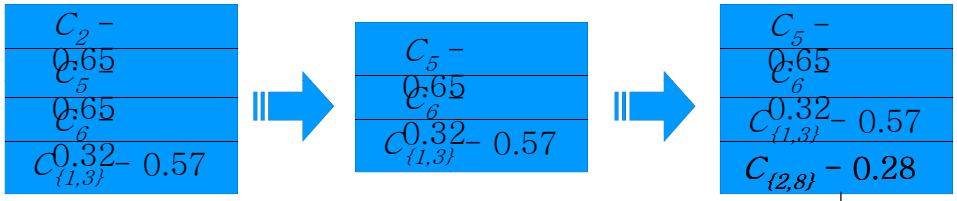


Laço de união de grupos

- Próxima iteração
  - $C_u = C_2 = \{ T_2 \}$; $C_v = C_8 = \{ T_8 \}$
  - $C_w = C_2 \cup C_8 = \{ T_2, T_8 \}$
  - Grupos que tem C_u ou C_v em sua lista local: { C_4, C_5, C_6, C_7, $C_{\{1,3\}}$ }
  - Vamos considerar a atualização em relação ao grupo C_4

Laço de união de grupos

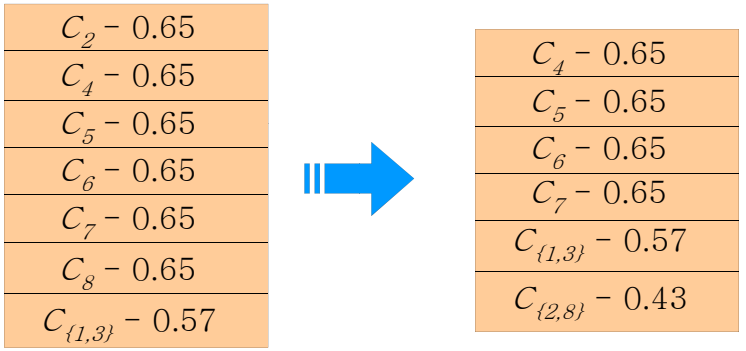
Lista local de C_4



$$g(C_4 = \{T_4\}, C_{\{2,8\}} = \{T_2, T_8\}) = \frac{link[C_4, C_2] + link[C_4, C_8]}{3^{1+2(0.67)} - 1^{1+2(0.67)} - 2^{1+2(0.67)}} = 0.28$$

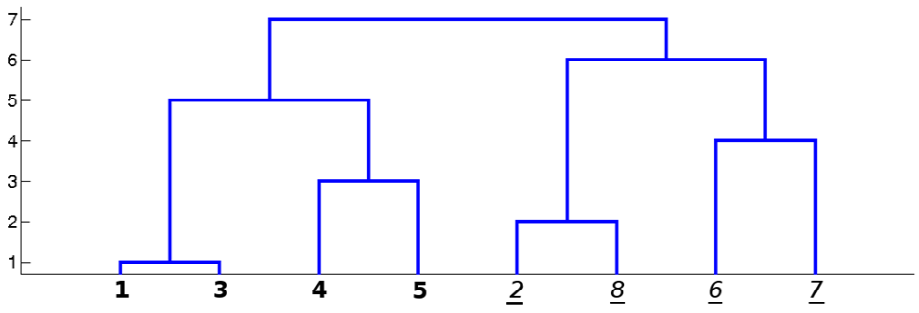
Laço de união de grupos

Lista global



Resultado

- Continuando o algoritmo, é obtida a seguinte hierarquia



Custo computacional

- Cálculo dos *links*
  - $O(N^{2,37})$
- Criação das lista*s* locais usando *heaps*
  - $O(N^2)$
- Criação da lista global usando *heap*
  - $O(N)$
- Laço de união dos grupos
  - $O(N^2 \log N)$

(Des)Vantagens

- Considera informação de vizinhança no agrupamento de objetos
- Robusto em relação a *outliers*, já que eles vão ter poucos *links*, pouco afetam o agrupamento
- Se o número de grupos for subestimado, o algoritmo para antes
  - Quando não há *links* entre grupos
  - Pode não obter uma hierarquia completa
- Como definir Θ e f(Θ) ?

Referências

- Guha, S.; Rastogi, R.; Shim, K.; ROCK: a robust clustering algorithm for categorical attributes, In 15th International Conference on Data Engineering, 1999.