Birch: An efficient data clustering method for very large databases

Jonathan de Andrade Silva

Sumário

- Motivação
- CF – *Cluster Feature*
- Algoritmo
- Considerações Finais

Motivação

- Tradicionais algoritmos assumem que o conjunto de dados preenche a memória
- Algoritmos de agrupamento realizam vários acessos ao conjunto de dados

CF – *Cluster Features*

- Representação compacta dos dados pertencentes ao grupo
- Mantém estatísticas suficientes para as operações no algoritmo
- Permite realizar operações de junção de grupos

CF – *Cluster Features*

- Dado um grupo C_i com N_i objetos n-dimensionais $C_i = \{ x_1, \ldots, x_{N_i} \}$
  - CF(N_i,ls,ss)
    - ls = $\sum_{j=1}^{N_i} x_j$
    - ss = $\sum_{j=1}^{N_i} x_j^2$
  - Permite a junção de grupos:
    - CF = CF¹ + CF² = (N_i^1+N_i^2, ls¹ + ls², ss¹ + ss²)

CF – *Cluster Features*

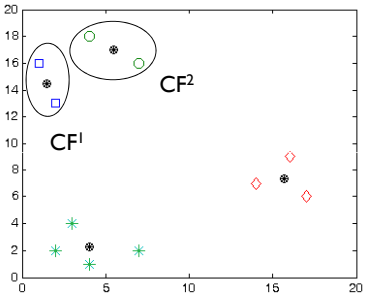
- Exemplo:

CF¹

ID	a_1	a_2
x_1	2	13
x_2	1	16
$\sum x$	3	29
$\sum x^2$	5	425

CF²

ID	a_1	a_2
x_3	7	16
x_4	4	18
$\sum x$	11	34
$\sum x^2$	65	580



CF^1=(2,[3 29],[5 425]) ; CF^2=(2,[11 34],[65 580])
CF = CF^1+CF^2=(4,[14 63],[70 1005])

CF – *Cluster Features*

- Medidas utilizadas:
  - Diâmetro:

$$\text{Diam(CF)} = \sqrt{\frac{2*N*ss - 2*ls^2}{N*(N-1)}}$$

  - Distância média entre grupos:

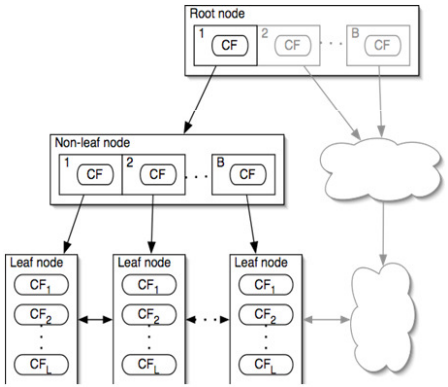
$$d(CF^1, CF^2) = \sqrt{\frac{N_1*ss_2 + N_2*ss_1 - 2*ls_1*ls_2}{N_1*N_2}}$$

Algoritmo

- Utiliza uma estrutura em árvore para manter os vetores CF na memória
- Similar a estrutura em árvore B+
- Possui 3 parâmetros :
  - *N. de filhos do nó não-folha (B)*
  - *N. de elementos do nó folha (L)*
  - *Limiar (T)*

Algoritmo

- Cada nó não-folha tem B entradas CF
- Cada nó folha tem L entradas CF que satisfazem o limiar T



Obtido do original de Hung Lai

Algoritmo

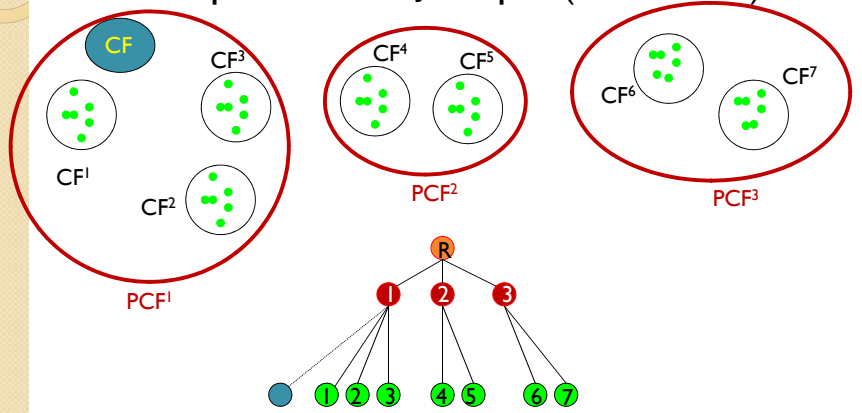
- Construção da árvore de CF com apenas uma leitura dos dados
- Aplicar um algoritmo de agrupamento nos nós folhas da árvore construída.

Algoritmo

- Inserção na CF-Tree:
  - Identificação do nó folha apropriado
  - Atualização do nó folha
  - Atualização dos nós não-folha do caminho

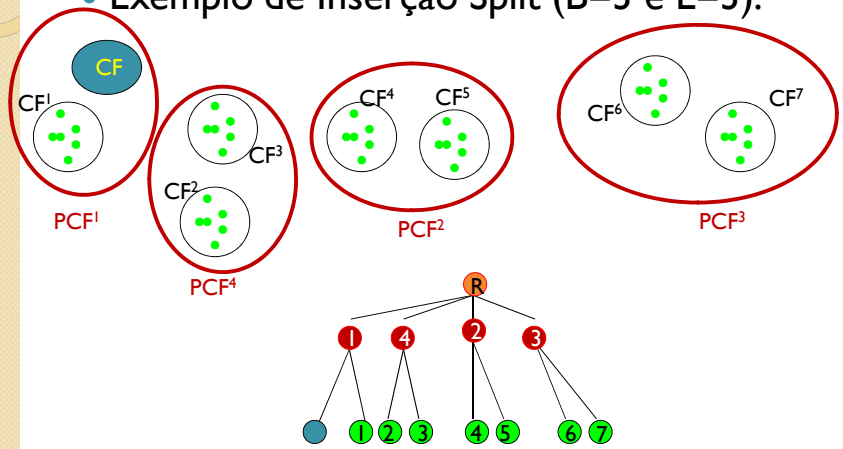
Algoritmo

- Exemplo de Inserção Split (B=3 e L=3):



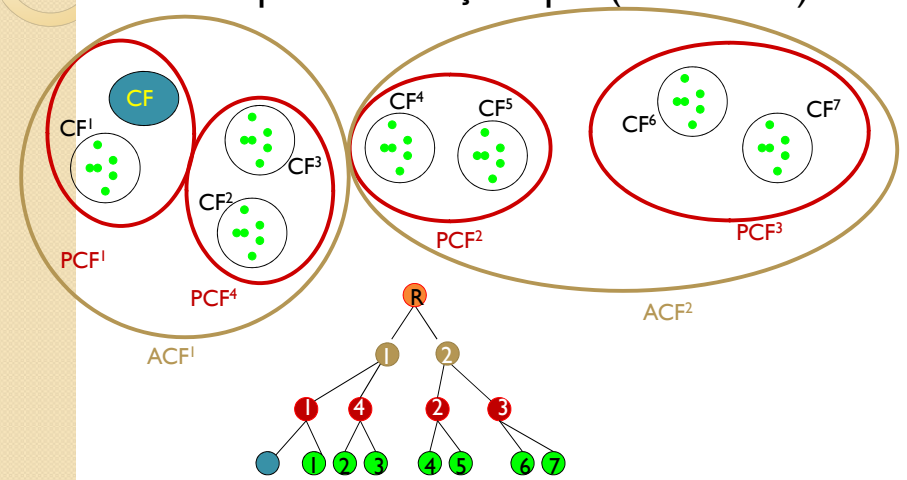
Algoritmo

- Exemplo de Inserção Split (B=3 e L=3):



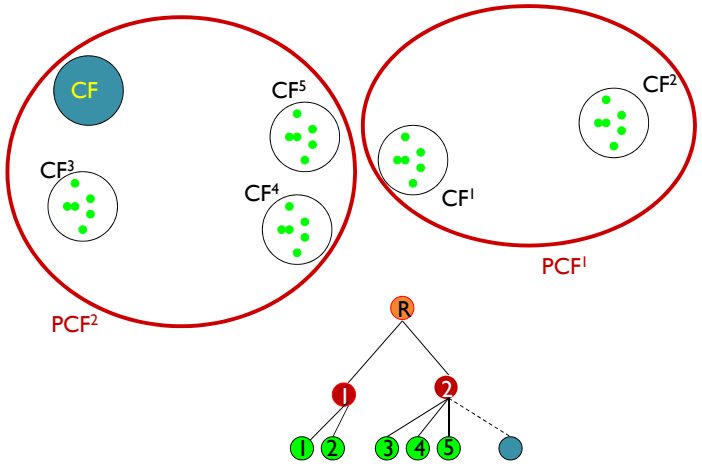
Algoritmo

- Exemplo de Inserção Split (B=3 e L=3):



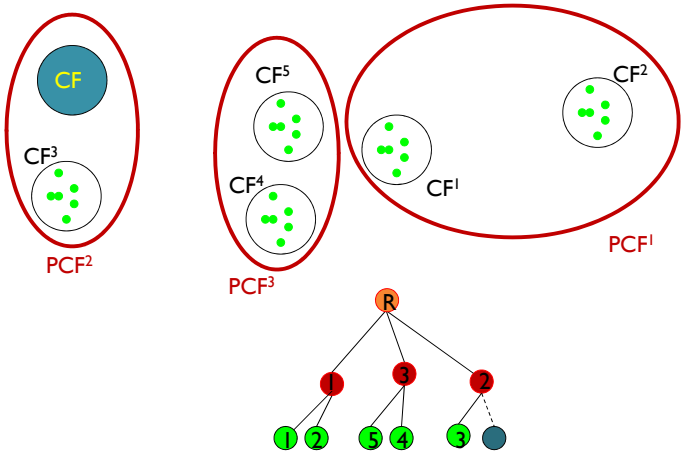
Algoritmo

- Exemplo de Inserção Merge (B=3 e L=3):



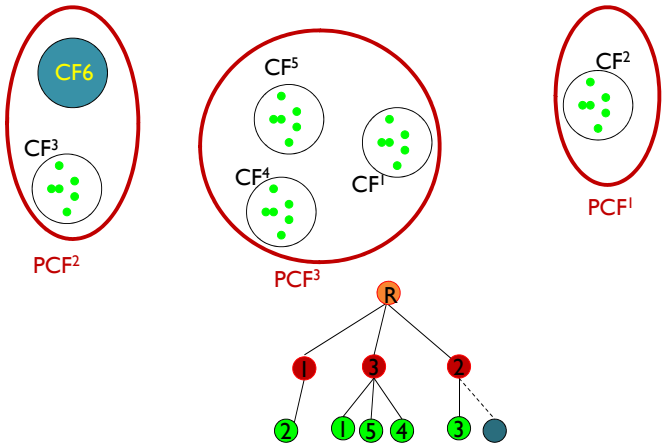
Algoritmo

- Exemplo de Inserção Merge (B=3 e L=3):



Algoritmo

- Exemplo de Inserção Merge (B=3 e L=3):

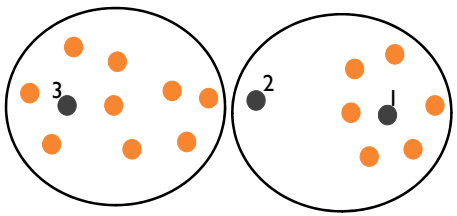


Considerações Finais

- Complexidade linear O(n) para acesso aos dados
- Complexidade $O(n*N*B*\log_B M/P)$ para inserção na árvore
- Possui 2 fases opcionais:
  - Estimação do limiar T e redução da árvore
  - Agrupamento refinado, porém necessita acessar os dados novamente.
- Sensível à ordem dos dados.
- Encontra grupos globulares
- Apenas para atributos numéricos

Considerações Finais

- Dependência a ordem dos dados:
  - Supondo d(3,2) < T, d(2,1) < T e d(3,1) > T.
  - Inserindo os objetos na árvore na seguinte ordem: 1-> 2 -> 3



Referências

- Zhang, T.; R., R.; Livny, M., *BIRCH: An efficient data clustering method for very large databases, SIGMOD '96, ACM Press.*
- Zhang, Tian , *Data clustering for very large datasets plus applications,* University of Wisconsin at Madison, 1997, PhD. Thesis.