BIRCH

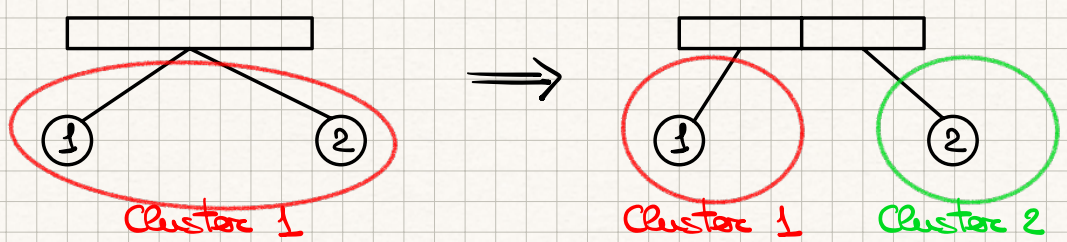
L'acronimo sta per Balanced Iterative Reducing and Clustering using Hierarchies.
Usa un approccio agglomerativo.

Cosa BIRCH tenda di risolvere?

1) La maggior parte degli algoritmi non tiene conto del fatto che i dati non entrino nella memoria principale.
2) La maggior parte degli algoritmi non minimizzano il numero di scansioni del DB.
3) Il costo dell'I/O è di natura alto.

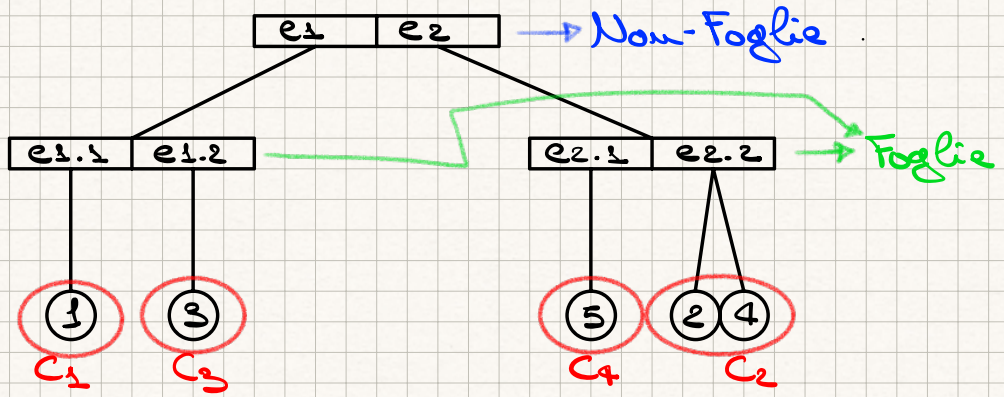
COMPLESSITÀ: $O(n)$

L'idea che sta alla base di BIRCH è quella di costruire i cluster utilizzando una struttura dati ad albero. Ogni volta che si aggiunge un oggetto a un cluster, se esso diventa poco compatto e quindi troppo largo, bisogna dividere in due cluster:



Quelli nello schema sono nodi foglia, che nel secondo caso ha due entrate.

Ogni nodo ha un limite sul numero di entità che può possedere. Quando si supera, viene creato un nuovo livello dell'albero con un nodo padre:



CLUSTER FEATURE

È una struttura dati di tipo array di tre elementi dove ogni posizione ha un valore con significato specifico:

$$CF = \langle N, LS, SS \rangle$$

Ogni valore rappresenta delle statistiche che descrivono il cluster:

N = numero di punti

$LS = \sum_{i=1}^{N} x_i$ // Somma lineare dei punti

$SS = \sum_{i=1}^{N} x_i^2$ // Somma dei quadrati dei punti

Questa struttura dati ha proprietà lineari nel senso che se abbiamo due cluster $C_1 \cup C_2 = C_3$ allora

CF_3 sarà la somma di $CF_1 + CF_2$.

Dai valori dentro un CF possiamo ricavare:

1) Centroide: $x_0 = \frac{\sum x_i}{n} = \frac{LS}{N}$

2) Raggio: $R = \sqrt{\frac{\sum_i^n (x_i - x_0)^2}{n}} = \sqrt{\frac{N \cdot SS - 2 \cdot LS^2 + N \cdot LS^2}{N^2}}$

3) Diametro: $D = \sqrt{\frac{\sum_i^n \sum_j^n (x_i - x_j)^2}{n(n-1)}} = \sqrt{\frac{2 \cdot N \cdot SS - 2 \cdot LS^2}{N \cdot (N-1)}}$

CF-TREE

È la struttura dati che usa BIRCH. Per formarla servono due parametri:

1) Massimo numero di entità per nodo
2) Massimo diametro di tutte le entità in un nodo.

Questa struttura dati salva i vari cluster feature (CF) dei cluster, e sfrutta la proprietà lineare, per cui un nodo padre avrà il $CF = \sum CF_{figli}$.

Altri parametri:

1) B: BRANCHING FACTOR, che specifica il massimo numero di figli per non-leaf node.

2) T: THRESHOLD PARAMETER, che specifica il massimo diametro che un cluster in una foglia può avere.

3) L: Numero massimo di entità in una foglia

ALGORITMO di BIRCH

1) Fase 1: Scansionare l'intero DB per costruire un CF-Tree che sta in memoria.

   a) Quando nel tentativo di aggiungere un oggetto si viola la threshold T:
      - Se c'è posto per creare una nuova entità si fa
      - Altrimenti si deve creare un nodo padre: Riempiamo i due figli prima con i cluster più lontani, gli altri li aggiungo ai nodi più vicini. Aggiorno i vari CF

   b) Se la threshold T non viene violata
      - Inserisci il punto nel cluster più vicino

2) Fase 2: Usare un algoritmo di clustering per clusterizzare i nodi foglia del CF-Tree.

Come fare quando CF-Tree non sta in memoria durante la fase 1? Aumentare la threshold T.
A questo punto non c'è bisogno di rileggere da capo il DB per correggere il CF-Tree, basta creare il nuovo albero dal vecchio, partendo dalle foglie.
Ripetere la procedura finché non sta in memoria.

PRO

1) Trova dei buoni cluster con un solo scan del DB
2) COMPLESSITÀ $O(n)$

CONTRO

1) Può gestire solo dati numerici per via dei dati da calcolare e mettere nel CF: centroide, ecc...
Difatti va fatta una media

2) Sensibile all'ordine dei dati, se vengono inseriti con ordine diverso creano CF-Tree diversi.

3) Dato che fissiamo la dimensione dei nodi foglia, potrebbero uscire cluster innaturali.

4) Dato che si basa sulla misura di raggio e diametro la forma dei cluster tende a essere sferica.