

T-61.6020 Popular Algorithms in Data Mining and Machine Learning

BIRCH: Balanced Iterative Reducing and Clustering using Hierarchies

Sami Virpioja

Adaptive Informatics Research Centre
Helsinki University of Technology

April 23, 2008

Outline

- 1 Introduction
- 2 Clustering Features
- 3 CF Tree
- 4 BIRCH Clustering
- 5 Performance and Experiments
- 6 Conclusions

Outline

- 1 Introduction
- 2 Clustering Features
- 3 CF Tree
- 4 BIRCH Clustering
- 5 Performance and Experiments
- 6 Conclusions

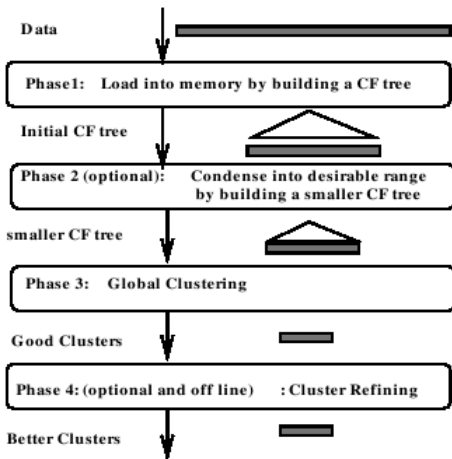
Task

- Data clustering with **computational constraints**:
- Given number of clusters K , a dataset of N points, and a distance-based measurement function, find a partition of the dataset that minimizes the function.
- Limited amount of memory (less than the dataset size).
- Minimize the time used for I/O operations.
 - I.e., do not read the data multiple times from the disk.

Properties of BIRCH

- **On-line algorithm**: Single scan of the dataset enough.
 - Optional refining with additional scans
- **I/O cost minimization**: Organize data in a in-memory, height-balanced tree.
- **Local decisions**: Each clustering decision is made without scanning all the points or clusters.
- **Outlier detection** (optional): Points in sparse regions are treated separately as outliers.

Algorithm Phases (1)



BIRCH overview [Zhang et al. 1997]

Algorithm Phases (2)

- Two main phases:
- Phase 1: Construction of CF tree using local clustering.
 - Incremental, one scan of the data enough.
- Phase 3: Global clustering of subclusters from the CF tree.
 - Use some standard clustering algorithm (e.g. hierarchical clustering, K-means).
- The rest is optional.

Outline

- 1 Introduction
- 2 Clustering Features**
- 3 CF Tree
- 4 BIRCH Clustering
- 5 Performance and Experiments
- 6 Conclusions

Clustering Features

- Clustering Feature: Triple $CF = (N, \mathbf{LS}, SS)$
 - N is the number of data points in cluster (0th order moment).
 - \mathbf{LS} is the linear sum of data points $\sum_{i=1}^N \mathbf{X}_i$ (1st order moment).
 - SS is the square sum of data points $\sum_{i=1}^N \|\mathbf{X}_i\|^2$ (2rd order moment).
- **CF Additivity Theorem:** Assume that for two disjoint clusters $CF_1 = (N_1, \mathbf{LS}_1, SS_1)$ and $CF_2 = (N_2, \mathbf{LS}_2, SS_2)$. For the merged cluster $CF_1 + CF_2 = (N_1 + N_2, \mathbf{LS}_1 + \mathbf{LS}_2, SS_1 + SS_2)$.

Representativity of Clustering Features (1)

- **CF Representativity Theorem:** Given the CF entries of subclusters, the following measurements can be computed accurately:
- For a single cluster X :
 - Centroid $\mathbf{X}_0 = \frac{1}{N} \sum_{i=1}^N \mathbf{X}_i$
 - Radius $R = \left(\frac{1}{N} \sum_{i=1}^N \|\mathbf{X}_i - \mathbf{X}_0\|^2 \right)^{\frac{1}{2}}$
 - Diameter $D = \left(\frac{1}{N(N-1)} \sum_{i=1}^N \sum_{j=1}^N \|\mathbf{X}_i - \mathbf{X}_j\|^2 \right)^{\frac{1}{2}}$

Representativity of Clustering Features (2)

- For two clusters X and Y :
 - Euclidian centroid distance $D0 = \|\mathbf{X}_0 - \mathbf{Y}_0\|$
 - Manhattan centroid distance $D1 = |\mathbf{X}_0 - \mathbf{Y}_0|$
 - Average inter-cluster distance

$$D2 = \left(\frac{1}{N_X N_Y} \sum_{i=1}^{N_X} \sum_{j=1}^{N_Y} \|\mathbf{X}_i - \mathbf{Y}_j\|^2 \right)^{\frac{1}{2}}$$
 - Average intra-cluster distance $D3 = \dots$
 - Variance increase distance $D4 = \dots$
- For example:

$$\begin{aligned} \sum_{i=1}^{N_X} \sum_{j=1}^{N_Y} \|\mathbf{X}_i - \mathbf{Y}_j\|^2 &= \sum_{i=1}^{N_X} \sum_{j=1}^{N_Y} (\mathbf{X}_i^T \mathbf{X}_i - 2\mathbf{X}_i^T \mathbf{Y}_j + \mathbf{Y}_j^T \mathbf{Y}_j) \\ &= SS_X + SS_Y - 2\mathbf{L}\mathbf{S}_X^T \mathbf{L}\mathbf{S}_Y \end{aligned}$$

CF Summary

- Thus, instead storing a set of data points with the set of their vectors, it can be **summarized** as a CF entry.
 - Sometimes called micro-clusters.
- Additivity: CF entries can be stored and calculated incrementally.
- Representativity: Some distances can be calculated without the individual data points.

Outline

- 1 Introduction
- 2 Clustering Features
- 3 CF Tree**
- 4 BIRCH Clustering
- 5 Performance and Experiments
- 6 Conclusions

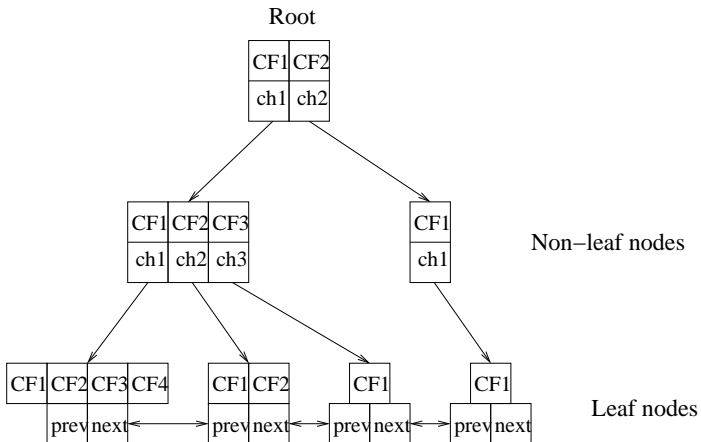
CF Tree

- Height-balanced tree with branching factor B , leaf size L and leaf threshold T .
- **Non-leaf nodes** contain at most B entries of the form $[CF_i, child_i]$.
 - Represents a cluster made up of all the subclusters of the entries.
- **Leaf nodes** contain at most L entries of the form $[CF_i]$ and pointers to previous and next leaf nodes.
 - All entries in a leaf node must satisfy **threshold requirement**, i.e., diameter or radius of the subcluster must be less than T .

CF Tree Example

$B = 3$

$L = 4$



Size of CF Tree

- In BIRCH, it is required that each node fits to a page of size P .
- Given the dimensionality (and type) of the data, B and L can be determined from P .
- Thus, size of the tree will be function of T :
Larger T allows more points in the same entries.

Operations of CF Tree

- Structure of CF tree is similar to B+-tree (used for representing sorted data).
- Basic operation: Insert a new CF entry (single point or subcluster) into the tree.
- Additional operation: Rebuild the tree with increased threshold T .

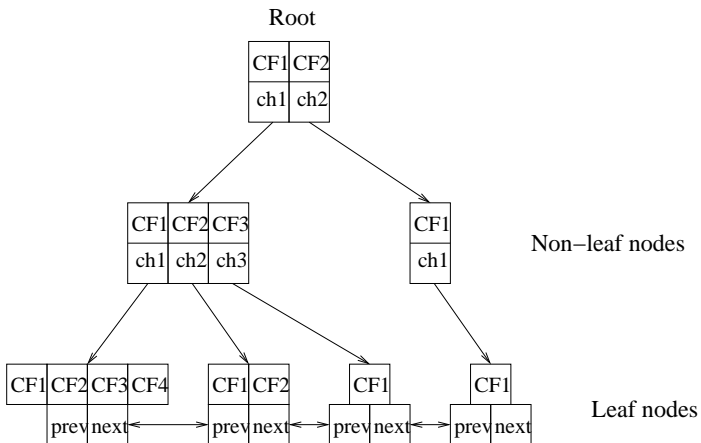
Insertion Algorithm

- 1 Identifying the appropriate leaf:
 - Descend the tree by always choosing the closest child node according to the chosen distance metric (D_0 – D_4).
- 2 Modifying the leaf:
 - Find the closest entry.
 - If threshold is not exceeded, **absorb** into the entry.
 - Otherwise, if L is not exceeded, **add** new entry to the leaf.
 - Otherwise, **split** the node: Choose the farthest entries and redistribute the rest, each to the closest one.
- 3 Modifying the path to the leaf:
 - If the child was not split, just update CF.
 - If there was a split, and B is not exceeded, add the new node to the parent. Otherwise, split the parent.
 - Recurse upwards.
- 4 A merging refinement:
 - When the split propagated up to some non-leaf node, test if the split pair are the closest of the entries.
 - If not, merge the closest entries and resplit if needed.

Insertion Algorithm

$B = 3$

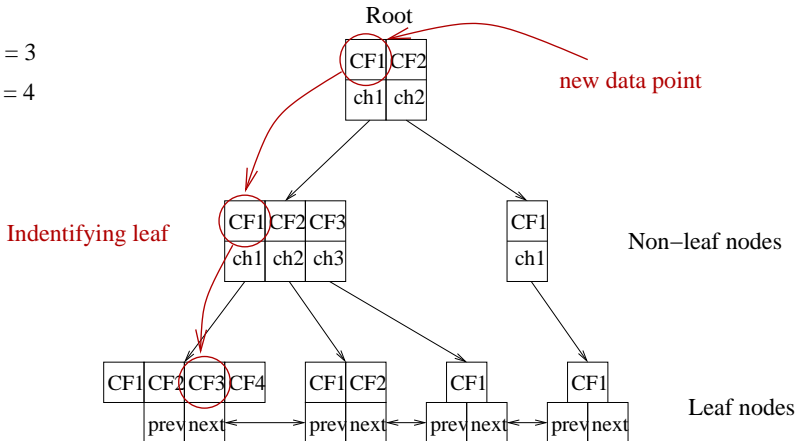
$L = 4$



Insertion Algorithm

$B = 3$

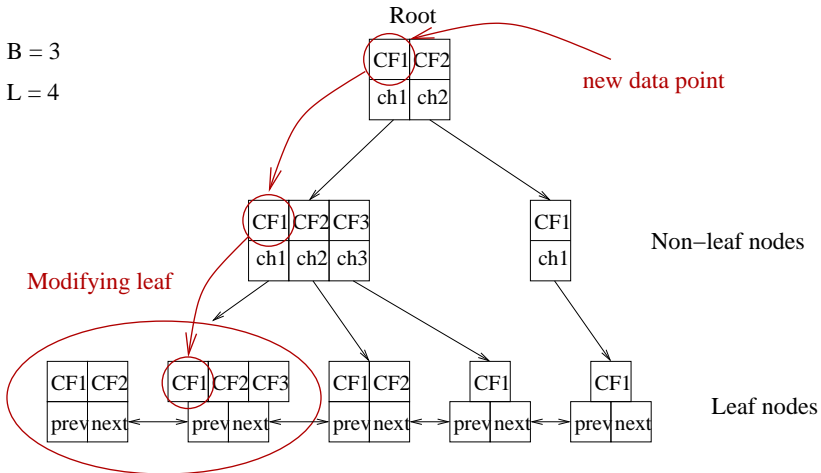
$L = 4$



Insertion Algorithm

$B = 3$

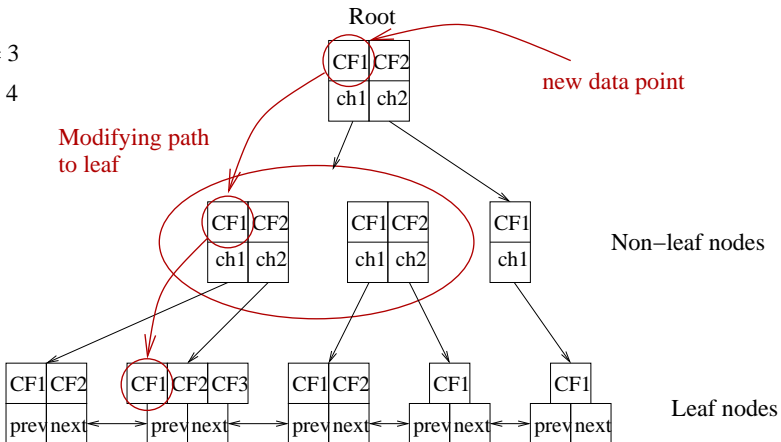
$L = 4$



Insertion Algorithm

$B = 3$

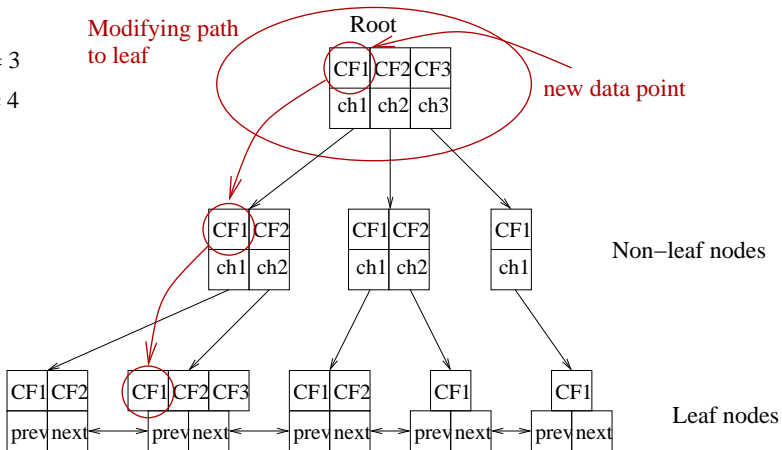
$L = 4$



Insertion Algorithm

$B = 3$

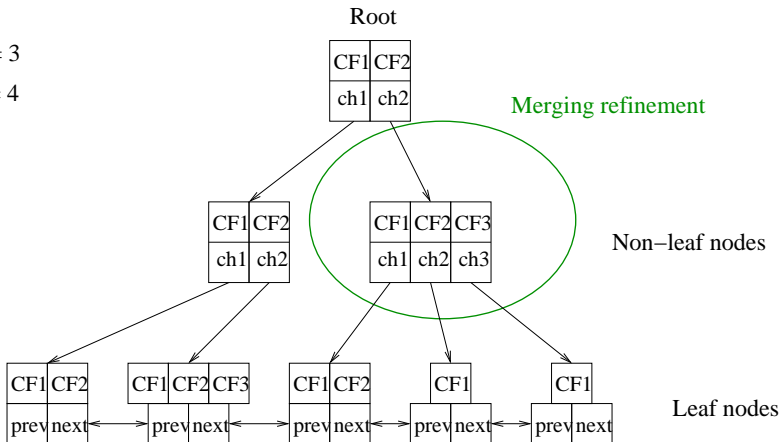
$L = 4$



Insertion Algorithm

$B = 3$

$L = 4$



Rebuilding Algorithm

- If the selected T led to a tree that is too large for the memory, it must be rebuilt with a larger T .
- Can be done with small extra memory and without scanning the data again.
- Assume that original tree t_i with threshold T_i , size S_i and height h_i .
- Proceed one path from a leaf to the root at a time:
 - ① Copy the path to the new tree.
 - ② Insert the leaf entries to the new tree. If they can be absorbed or fit in to the closest path of the tree, insert there. Otherwise insert to the leaf of the new path.
 - ③ Remove empty nodes from both the old and new tree.
- If for new tree t_{i+1} , if $T_{i+1} \geq T_i$, then $S_{i+1} \leq S_i$ and the transformation from t_i to t_{i+1} needs at most h_i extra pages of memory.

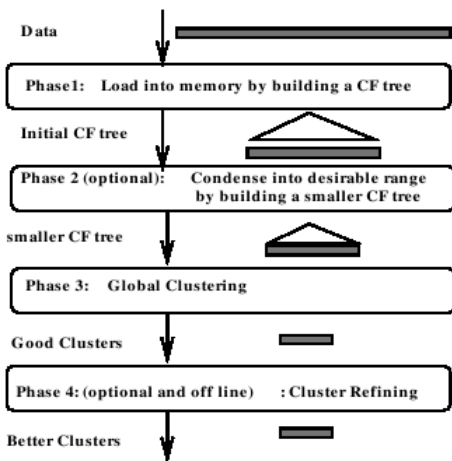
Anomalies

- Limited number of entries per node and skewed input order can produce some anomalies:
 - Two subclusters that should have been in one cluster are split.
 - Two subclusters that should not be in one cluster are kept in same node.
 - Two instances of the same data point might be entered into distinct leaf entries.
- Global clustering (BIRCH phase 3) solves the first two anomalies.
- Further passes over the data are needed for the latter (optional phase 4).

Outline

- 1 Introduction
- 2 Clustering Features
- 3 CF Tree
- 4 BIRCH Clustering**
- 5 Performance and Experiments
- 6 Conclusions

BIRCH Phases



BIRCH overview [Zhang et al. 1997]

BIRCH Phase 1

- 1 Start an initial in-memory CF tree t_0 with small threshold T_0 .
- 2 Scan the data and insert points to the current tree t_i .
- 3 If memory runs out before all the data is inserted, rebuild a new tree t_{i+1} with $T_{i+1} > T_i$.
 - Optionally detect outliers while rebuilding:
 - If a leaf entry has “far fewer” data points than on average, write them to disk instead of adding to the tree.
 - If disk space runs out, go through all outliers and re-absorb those that do not increase the tree size.
- 4 Otherwise, proceed to the next phase. (And, optionally, recheck if some of the potential outliers can be absorbed.)

BIRCH Phases 1–2

- After Phase 1, the clustering should be easier in several ways:
 - **Fast** because no I/O operations are needed, and the problem of clustering N points is reduced to clustering of M subclusters (with hopefully $M \ll N$).
 - **Accurate** because outliers are eliminated and the rest data is reflected with finest granularity given the memory limit.
 - **Less order sensitive** because the order of the leaf entries contain better data locality than an arbitrary order.
- Phase 2: If known that the global clustering algorithm in Phase 3 works best with some range of the input size M , continue reducing the tree until the range is reached.

BIRCH Phase 3

- Use a global or semi-global clustering for the M subclusters (CF entries of the leaf nodes).
- Fixes the anomalies caused by limited page size in Phase 2.
- Several possibilities for handling the subclusters:
 - Naive: Use centroid to represent the subcluster.
 - Better: N data points in the centroids.
 - Accurate: As seen, the most pointwise distance and quality metrics can be calculated from the CF vectors.
- The authors of BIRCH used agglomerative hierarchical clustering with distances metrics $D2$ and $D4$.

BIRCH Phase 4

- Use the centroids of the clusters from Phase 3 as seeds.
- Make additional scan of the data and redistribute each point to the cluster corresponding to the nearest seed.
- Fixes the misplacement problems of the first phases.
- Can be extended with additional passes (cf. K-means).
- Optionally discard outliers (i.e. those too far from the closest seed).

Outline

- 1 Introduction
- 2 Clustering Features
- 3 CF Tree
- 4 BIRCH Clustering
- 5 Performance and Experiments**
- 6 Conclusions

CPU cost

- N points in \mathbb{R}^d , memory U bytes, page size P bytes
 - Maximum size of the tree $\frac{U}{P}$
- Phases 1 and 2:
 - Inserting all data points: $O(N * d * B * \log_B \frac{U}{P})$
 - Plus something less for the rebuildings.
 - Plus some I/O usage if outliers are written to the disk.
- Phase 3:
 - Depends on the algorithm and the number of subclusters M .
 - If M is limited in Phase 2, it does not depend on N .
- Phase 4:
 - One iteration is $O(N * K)$
- So scales up pretty linearly.

Experiments

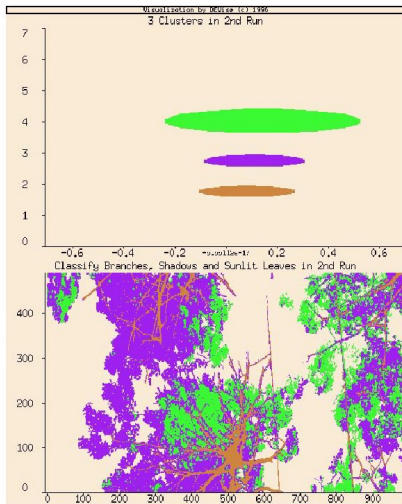
- In [Zhang et al. 1996] and [Zhang et al. 1997], the authors compare the performance to K-means and CLARANS.
 - CLARANS uses a graph of partitions and searches it locally to find a good one.
- Random 2-d datasets of $K = 100$ clusters
 - Centers either on a grid, on a curve of sine function, or random
 - About 1000 points per each cluster
- Unsurprisingly, BIRCH uses less memory and is faster, less order-sensitive and more accurate.
- Scalability seems to be quite good:
 - Linear w.r.t. cluster number and points per cluster.
 - Little worse than linear w.r.t. data dimensionality.

Application: Filtering Trees from Background



Images taken in NIR and VIS [Zhang et al. 1997]

Application: Filtering Trees from Background



Separate branches, shadows and sunlit leaves [Zhang et al. 1997]




Outline

- 1 Introduction
- 2 Clustering Features
- 3 CF Tree
- 4 BIRCH Clustering
- 5 Performance and Experiments
- 6 Conclusions**

Conclusions

- “Single-pass, sort-of-linear time algorithm that results in a sort-of-clustering of large number of data points, with most outliers sort-of-thrown-out.” [Fox & Gribble 1997]
- Lots of heuristics in the overall algorithm.
- The novelty was in dealing with large datasets and using summary information (CF entries) instead of the data itself.
 - Several extensions to the idea: parallel computation, different data types, different summary schemes, different clustering algorithms.

References

-  Tian Zhang, Raghu Ramakrishnan and Miron Livny (1996). BIRCH: An Efficient Data Clustering Method for Very Large Databases. In Proc. 1996 ACM SIGMOD international Conference on Management of Data, pp. 103–114. ACM Press, New York, NY.
-  Tian Zhang, Raghu Ramakrishnan and Miron Livny (1997). BIRCH: A New Data Clustering Algorithm and Its Applications. Data Mining and Knowledge Discovery, 1:2(141–182). Springer Netherlands.
-  Armando Fox and Steve Gribble (1997). Databases Paper Summaries. [On-line, cited 23 April 2008.] Available at: <http://www.cs.berkeley.edu/~fox/summaries/database/>