Birch: An efficient data clustering method for very large databases

Tian Zhang, Raghu Ramakrishnan, Miron Livny

CPSC 504

Presenter: Joel Lanir

Discussion: Dan Li

Outline

- What is data clustering
- Data clustering applications
- Previous Approaches
- Birch's Goal
- Clustering Feature
- Birch clustering algorithm
- Clustering example

What is Data Clustering?

A cluster is a closely-packed group.

A collection of data objects that are similar to one another and treated collectively as a group.

Data Clustering is the partitioning of a dataset into clusters

Data Clustering

- Helps understand the natural grouping or structure in a dataset
- Large set of multidimensional data
- Data space is usually not uniformly occupied
- Identify the sparse and crowded places
- Helps visualization

Discussion

- Can you give some examples for very large databases? What applications can you imagine that require such large databases for clustering?
- What are the special requirements that "large" databases pose on clustering, or more general on data mining?

Some Clustering applications

- Biology – building groups of genes with related patterns
- Marketing – partition the population of consumers to market segments
- Division of WWW pages into genres.
- Image segmentations – for object recognition
- Land use – Identification of areas of similar land use from satellite images
- Insurance – Identify groups of policy holders with high average claim cost

Data Clustering – previous approaches

- probability based (Machine learning): make wrong assumption that distributions on attributes are independent on each other
- Probability representations of clusters is expensive

Approaches

Distance Based (statistics)

- Must be a distance metric between two items
- assumes that all data points are in memory and can be scanned frequently
- Ignores the fact that not all data points are equally important
- Close data points are not gathered together
- Inspects all data points on multiple iterations

These approaches do not deal with dataset and memory size issues!

Clustering parameters

- Centroid – Euclidian center
- Radius – average distance to center
- Diameter – average pairwise difference within a cluster

Radius and diameter are measures of the tightness of a cluster around its center. We wish to keep these low.

Clustering parameters

- Other measurements (like the Euclidean distance of the centroids of two clusters) will measure how far away two clusters are.

A good quality clustering will produce high intra-clustering and low inter-clustering

A good quality clustering can help find hidden patterns

Birch's goals:

- Minimize running time and data scans, thus formulating the problem for large databases
- Clustering decisions made without scanning the whole data
- Exploit the non uniformity of data – treat dense areas as one, and remove outliers (noise)

Clustering Feature (CF)

- CF is a compact storage for data on points in a cluster
- Has enough information to calculate the intra-cluster distances
- Additivity theorem allows us to merge sub-clusters

Clustering Feature (CF)

Given N d-dimensional data points in a cluster: $\{X_i\}$ where i = 1, 2, …, N,

CF = (*N*, *LS*, *SS*)

N is the number of data points in the cluster,

LS is the linear sum of the N data points,

SS is the square sum of the N data points.

CF Additivity Theorem

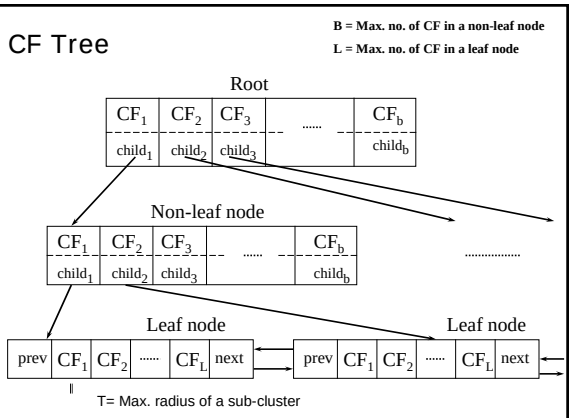
If CF1 = (N_1, LS_1, SS_1), and

CF2 = (N_2 ,LS_2, SS_2) are the CF entries of two disjoint subclusters.

The CF entry of the subcluster formed by merging the two disjoin subclusters is:

CF1 + CF2 = (N_1 + N_2 , LS_1 + LS_2, SS_1 + SS_2)

CF Tree



CF TREE

- T is the threshold for the diameter or radius of the leaf nodes
- The tree size is a function of T. The bigger T is, the smaller the tree will be.
- The CF tree is built dynamically as data is scanned.

CF Tree Insertion

- Identifying the appropriate leaf: recursively descending the CF tree and choosing the closest child node according to a chosen distance metric
- Modifying the leaf: test whether the leaf can absorb the node without violating the threshold. If there is no room, split the node
- Modifying the path: update CF information up the path.

Birch Clustering Algorithm

- Phase 1: Scan all data and build an initial in-memory CF tree.
- Phase 2: condense into desirable length by building a smaller CF tree.
- Phase 3: Global clustering
- Phase 4: Cluster refining – this is optional, and requires more passes over the data to refine the results

Birch – Phase 1

- Start with initial threshold and insert points into the tree
- If run out of memory, increase threshold value, and rebuild a smaller tree by reinserting values from older tree and then other values
- Good initial threshold is important but hard to figure out
- Outlier removal – when rebuilding tree remove outliers

Birch - Phase 2

- Optional
- Phase 3 sometime have minimum size which performs well, so phase 2 prepares the tree for phase 3.
- Removes outliers, and grouping clusters.

Birch – Phase 3

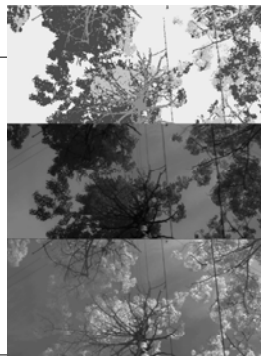
- Problems after phase 1:
  - Input order affects results
  - Splitting triggered by node size
- Phase 3:
  - cluster all leaf nodes on the CF values according to an existing algorithm
  - Algorithm used here: agglomerative hierarchical clustering

Birch – Phase 4

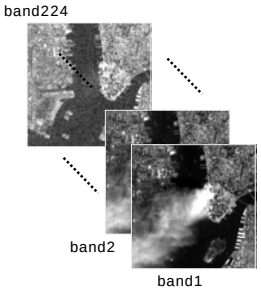
- Optional
- Additional scan/s of the dataset, attaching each item to the centroids found.
- Recalculating the centroids and redistributing the items.
- Always converges

Clustering example

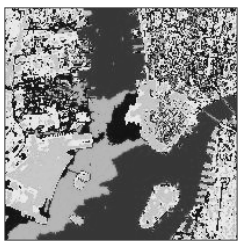
Pixel classification in images
From top to bottom:
- *BIRCH classification*
- *Visible wavelength band*
- *Near-infrared band*



Clustering example



K-means Clustering
to 5 classes



Conclusions

- Birch performs faster than then existing algorithms on large datasets
- Scans whole data only once
- Handles outliers

Discussion

- After reading the two papers for data mining, what do you think is the criteria to say if a data mining algorithm is “good”?
  - Efficiency?
  - I/O cost?
  - Memory/disk requirement?
  - Stability?
  - Immunity to abnormal data?

Thanks for listening