Clustering

CS 145

Fall 2015

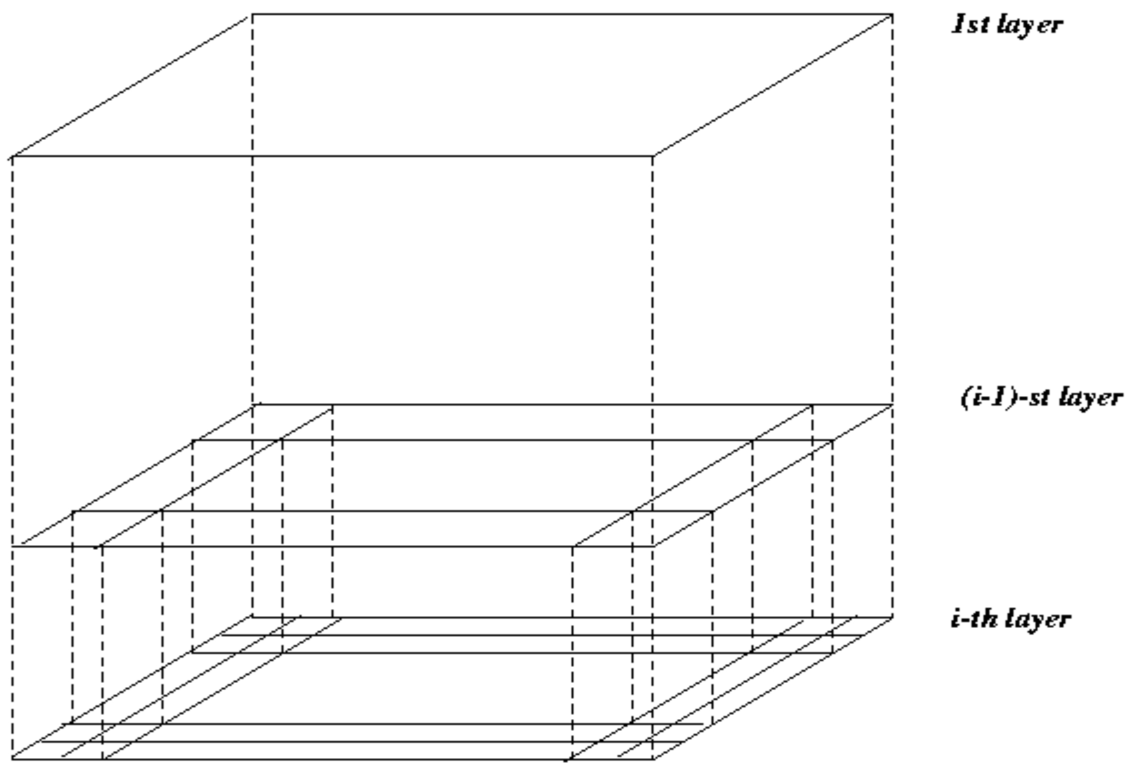
Wei Wang

Grid-based Clustering Methods

- Ideas
  - Using multi-resolution grid data structures
  - Use dense grid cells to form clusters
- Several interesting methods
  - STING
  - CLIQUE

STING: A Statistical Information Grid Approach

- The spatial area area is divided into rectangular cells
- There are several levels of cells corresponding to different levels of resolution

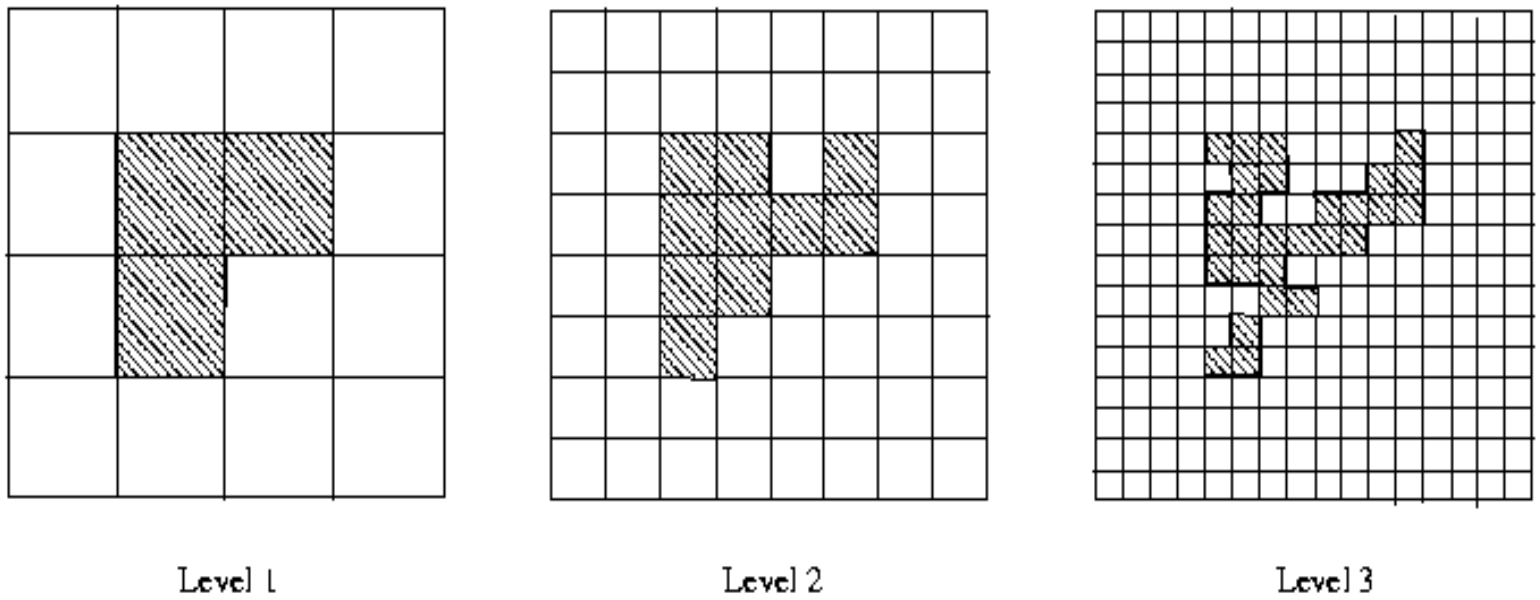


STING: A Statistical Information Grid Approach (2)

- Each cell at a high level is partitioned into a number of smaller cells in the next lower level
- Statistical information of each cell is calculated and stored beforehand and is used to answer queries
- Parameters of higher level cells can be easily calculated from parameters of lower level cell
  - *count*, *mean*, *s*, *min*, *max*
  - type of distribution—normal, *uniform*, etc.
- Use a top-down approach to answer spatial data queries
- Start from a pre-selected layer—typically with a small number of cells
- For each cell in the current level compute the confidence interval

STING: A Statistical Information Grid Approach (3)

- Remove the irrelevant cells from further consideration
- When finish examining the current layer, proceed to the next lower level
- Repeat this process until the bottom layer is reached



STING: A Statistical Information Grid Approach (4)

- Advantages:
  - Query-independent, easy to parallelize, incremental update
  - $O(K)$, where K is the number of grid cells at the lowest level
- Disadvantages:
  - All the cluster boundaries are either horizontal or vertical, and no diagonal boundary is detected

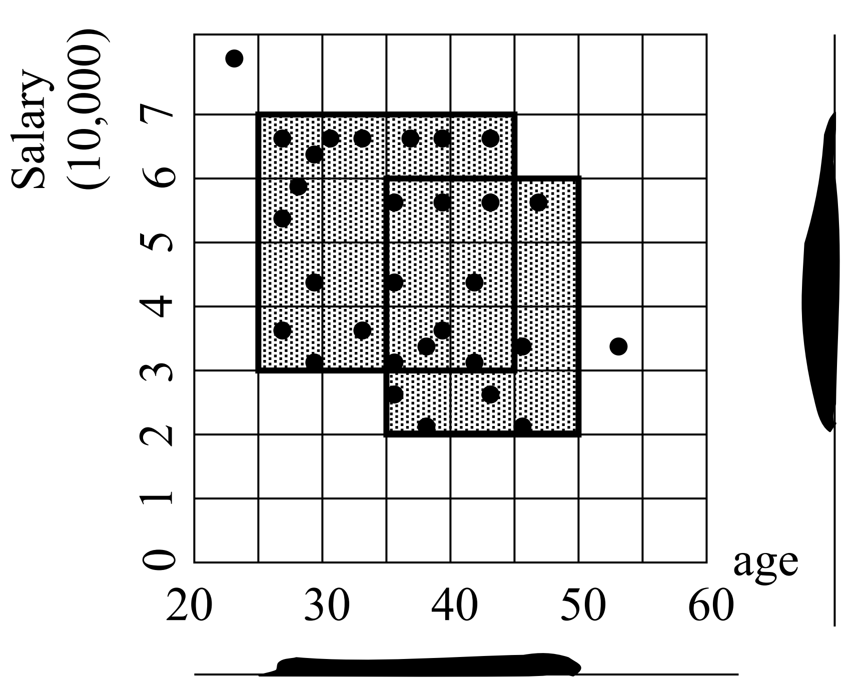
CLIQUE (Clustering In QUEst)

- Automatically identifying subspaces of a high dimensional data space that allow better clustering than original space
- CLIQUE can be considered as both density-based and grid-based
  - It partitions each dimension into the same number of equal length interval
  - It partitions an m-dimensional data space into non-overlapping rectangular units
  - A unit is dense if the fraction of total data points contained in the unit exceeds the input model parameter
  - A cluster is a maximal set of connected dense units within a subspace

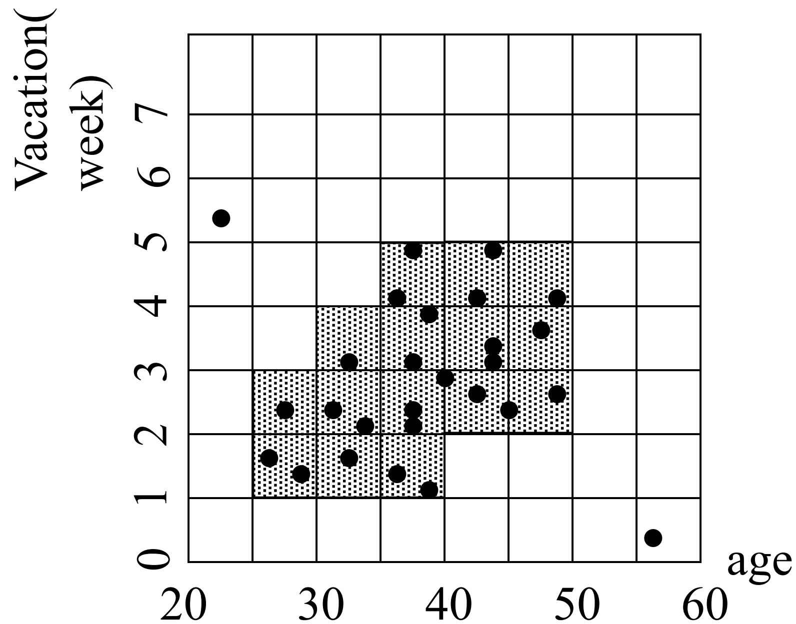
CLIQUE: The Major Steps

- Partition the data space and find the number of points that lie inside each cell of the partition.
- Identify the subspaces that contain clusters using the Apriori principle
- Identify clusters:
  - Determine dense units in all subspaces of interests
  - Determine connected dense units in all subspaces of interests.
- Generate minimal description for the clusters
  - Determine maximal regions that cover a cluster of connected dense units for each cluster
  - Determination of minimal cover for each cluster

CLIQUE



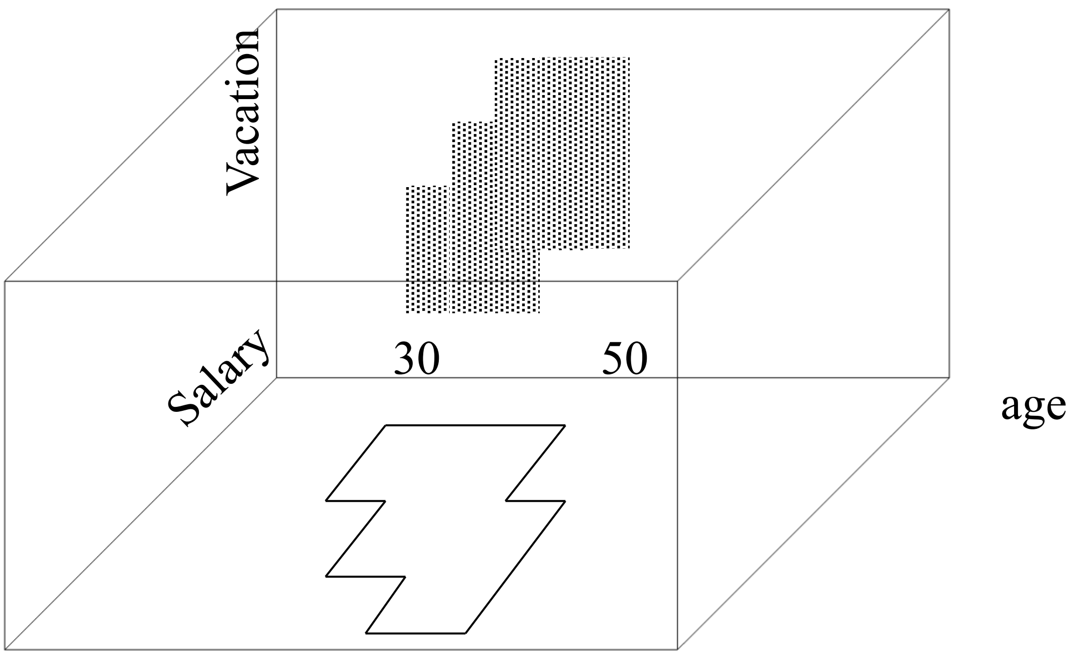
Salary (10,000)
0 1 2 3 4 5 6 7
20 30 40 50 60
age



Vacation(week)
0 1 2 3 4 5 6 7
20 30 40 50 60
age

CLIQUE

$\tau = 3$



Strength and Weakness of *CLIQUE*

- Strength
  - It *automatically* finds subspaces of the highest dimensionality such that high density clusters exist in those subspaces
  - It is *insensitive* to the order of records in input and does not presume some canonical data distribution
  - It scales *linearly* with the size of input and has good scalability as the number of dimensions in the data increases
- Weakness
  - The accuracy of the clustering result may be degraded at the expense of simplicity of the method

Outlier Analysis

- "One person's noise is another person's signal"
- Outliers: the objects considerably dissimilar from the remainder of the data
  - Examples: credit card fraud, Michael Jordon, etc
  - Applications: credit card fraud detection, telecom fraud detection, customer segmentation, medical analysis, etc

Distance-based Outliers

- A DB(p, D)-outlier is an object O in a dataset T s.t. at least fraction p of the objects in T lies at a distance greater than distance D from O
- Algorithms for mining distance-based outliers
  - The index-based algorithm, the nested-loop algorithm, the cell-based algorithm

Index-based Algorithms

- Find DB(p, D) outliers in T with n objects
  - Find an object having at most $\lfloor n(1-p) \rfloor$ neighbors with radius D
- Algorithm
  - Build a standard multidimensional index
  - Search every object O with radius D
    - If there are at least $\lfloor n(1-p) \rfloor$ neighbors, O is not an outlier
    - Else, output O

Pros and Cons of Index-based Algorithms

- Complexity of search $O(kN^2)$
  - More scalable with dimensionality than depth-based approaches
- Building a right index is very costly
  - Index building cost renders the index-based algorithms non-competitive

A Naïve Nested-loop Algorithm

- For j=1 to n do
  - Set $count_j$=0;
  - For k=1 to n do if (dist(j,k)<D) then $count_j$++;
  - If $count_j \le \lfloor n(1-p) \rfloor$ then output j as an outlier;
- No explicit index construction
  - $O(N^2)$
- Many database scans

Optimizations of Nested-loop Algorithm

- Once an object has at least $\lfloor n(1-p) \rfloor$ neighbors with radius D, no need to count further
- Use the data in main memory as much as possible
  - Reduce the number of database scans