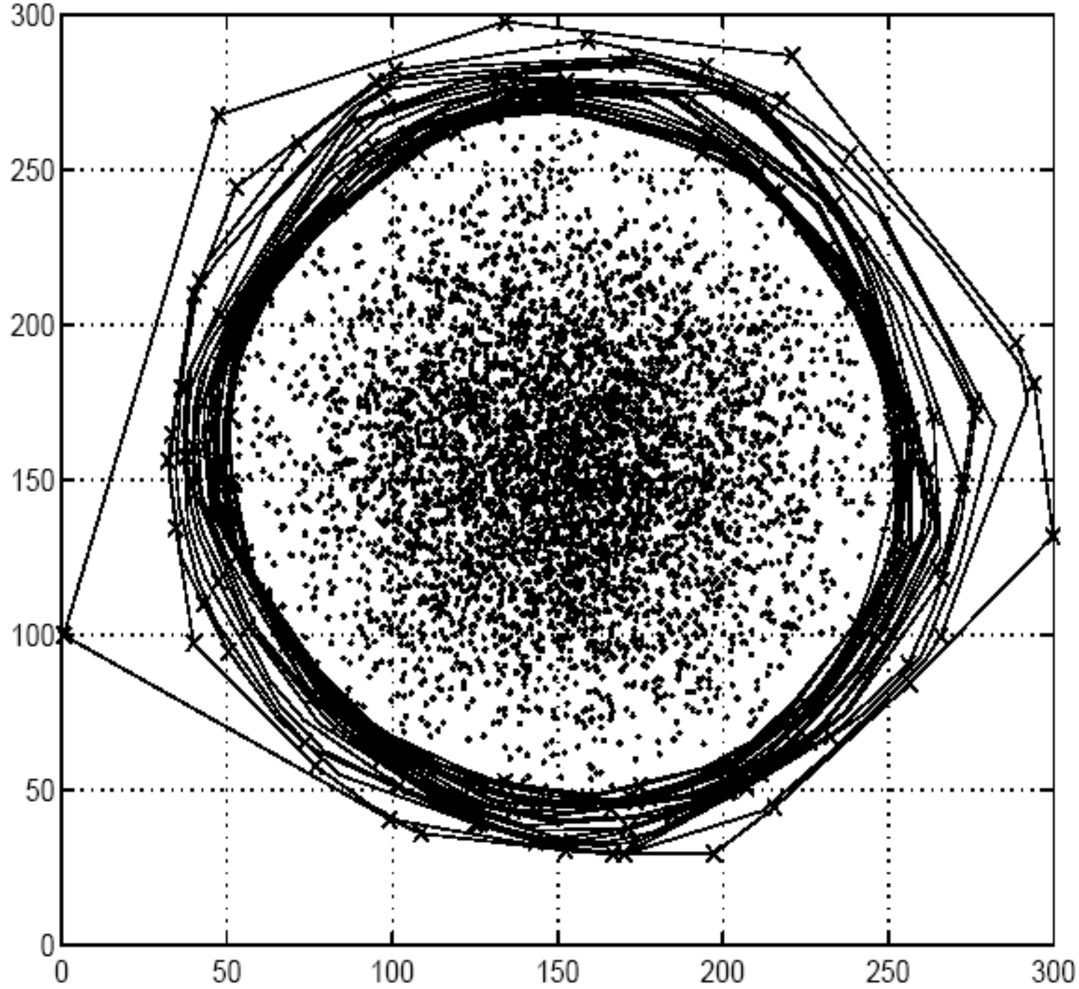
Proximity-based Outlier Detection

- Objects far away from the others are outliers
- The proximity of an outlier deviates significantly from that of most of the others in the data set
- Distance-based outlier detection: An object o is an outlier if its neighborhood does not have enough other points
- Density-based outlier detection: An object o is an outlier if its density is relatively much lower than that of its neighbors

Depth-based Methods

- Organize data objects in layers with various depths
  - The shallow layers are more likely to contain outliers
- Example: Peeling, Depth contours
- Complexity $O(N^{\lceil k/2 \rceil})$ for k-d datasets
  - Unacceptable for k>2

Depth-based Outliers: Example



Distance-based Outliers

- A DB(p, D)-outlier is an object O in a dataset T such that at least a fraction p of the objects in T lie at a distance greater than distance D from O
- The larger D, the more outlying
- The larger p, the more outlying

Index-based Algorithms

- Find DB(p, D) outliers in T with n objects
  - Find an objects 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

Index-based Algorithms: Pros & Cons

- 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 $count_j <= \lfloor n(1-p) \rfloor$ then output j as an outlier;
- No explicit index construction
  - $O(N^2)$
- Many database scans

Improving Nested-loop Algorithm

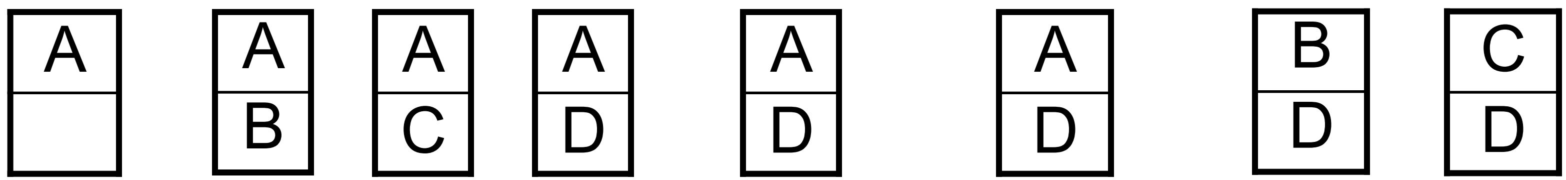
- Once an object has at least $\lfloor n(1-p) \rfloor$ neighbors with radius D, no need to count further
- Use the data in main memory as much as possible
  - Reduce the number of database scans

Block-based Nested-loop Algorithm

- Partition the available memory into two blocks with an equivalent size
- Fill the first block, compare objects in the block, mark non-outliers
- Read remaining objects into the second block, compare objects from the first and second block
  - Mark non-outliers, only compare potential outliers in the first block
  - Output unmarked objects in the first block as outliers
- Swap the names of the first and second blocks, until all objects have been processed

Example

Dataset has four blocks: A, B, C, and D



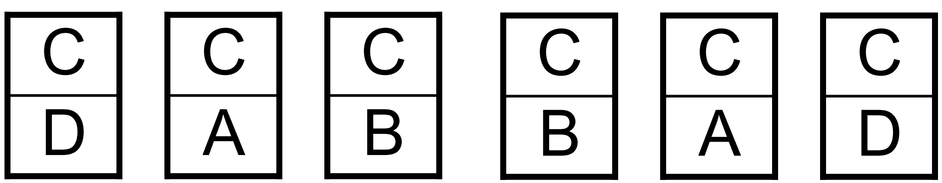
Compare objects in A (1 read)

Compare objects in A to those in B, C, and D (3 reads)

Compare objects in D (0 read)

Compare objects in D to those in A (0 read)

Compare objects in D to those in B and C (2 reads)



Compare objects in C to those in C, D, A, and B (2 reads)

Compare objects in B to those in B, C, A, and D (2 reads)

10 blocks are read in total
10/4=2.5 passes over T

Nested-loop Algorithm: Analysis

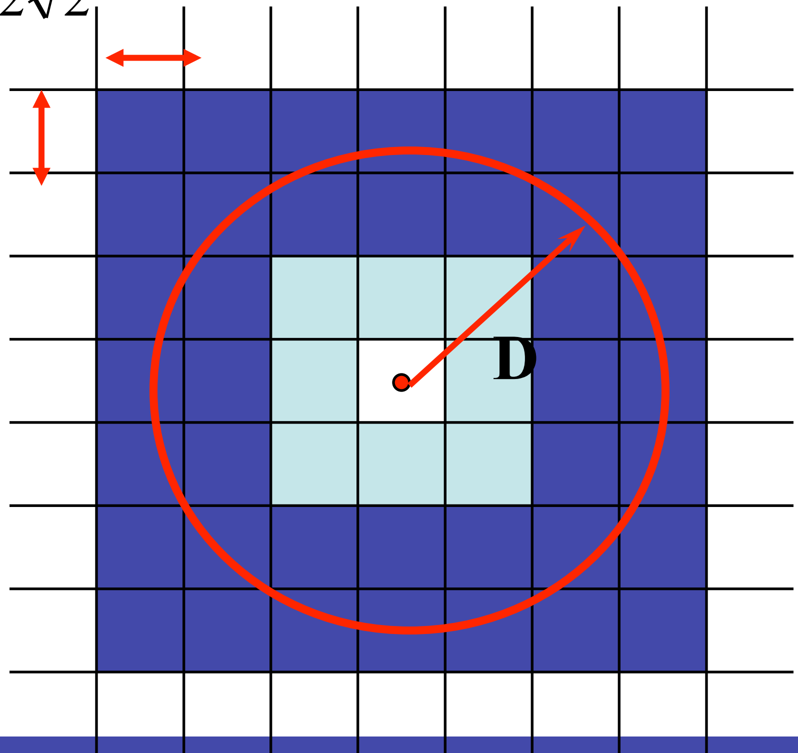
- The data set is partition into n blocks
- Total number of block reads:
  - $n+(n-2)(n-1)=n^2-2n+2$
- The number of passes over the dataset
  - $\geq (n-2)$
- Many passes for large datasets

A Cell-based Approach

$$L_1(C_{x,y}) = \{C_{u,v} \mid |u-x| \le 1, |v-y| \le 1, C_{u,v} \ne C_{x,y}\}$$

$$l = \frac{D}{2\sqrt{2}}$$

$$L_2(C_{x,y}) = \{C_{u,v} \mid |u-x| \le 3, |v-y| \le 3, C_{u,v} \notin L_1(C_{x,y}), C_{u,v} \ne C_{x,y}\}$$



M+ objects in $C_{x,y}$

➔ no outlier in $C_{x,y}$

M+ objects in $C_{x,y} \cup L_1(C_{x,y})$

➔ no outlier in $C_{x,y}$

M- objects in $C_{x,y} \cup L_1(C_{x,y}) \cup L_2(C_{x,y})$

➔ all objects in $C_{x,y}$ are outliers

The Algorithm

- Quantize each object to its appropriate cell
- Label all cells having m+ objects red
  - No outlier in red cells
- Label L_1 neighbours of red cells, and cells having m+ objects in $C_{x,y} \cup L1(C_{x,y})$ pink
  - No outlier in pink cells
- Output objects in cells having m- objects in $C_{x,y} \cup L_1(C_{x,y}) \cup L_2(C_{x,y})$ as outliers
- For remaining cells, check them one by one

Cell-based Approach: Analysis

- A typical cell has 8 L_1 neighbours and 40 L_2 neighbours
- Complexity: O(m+N) (m: # of cells)
  - The worst case: no red/pink cell at all
  - In practice, many red/pink cells
- The method can be easily generalized to k-d space and other distance functions

Handling Large Datasets

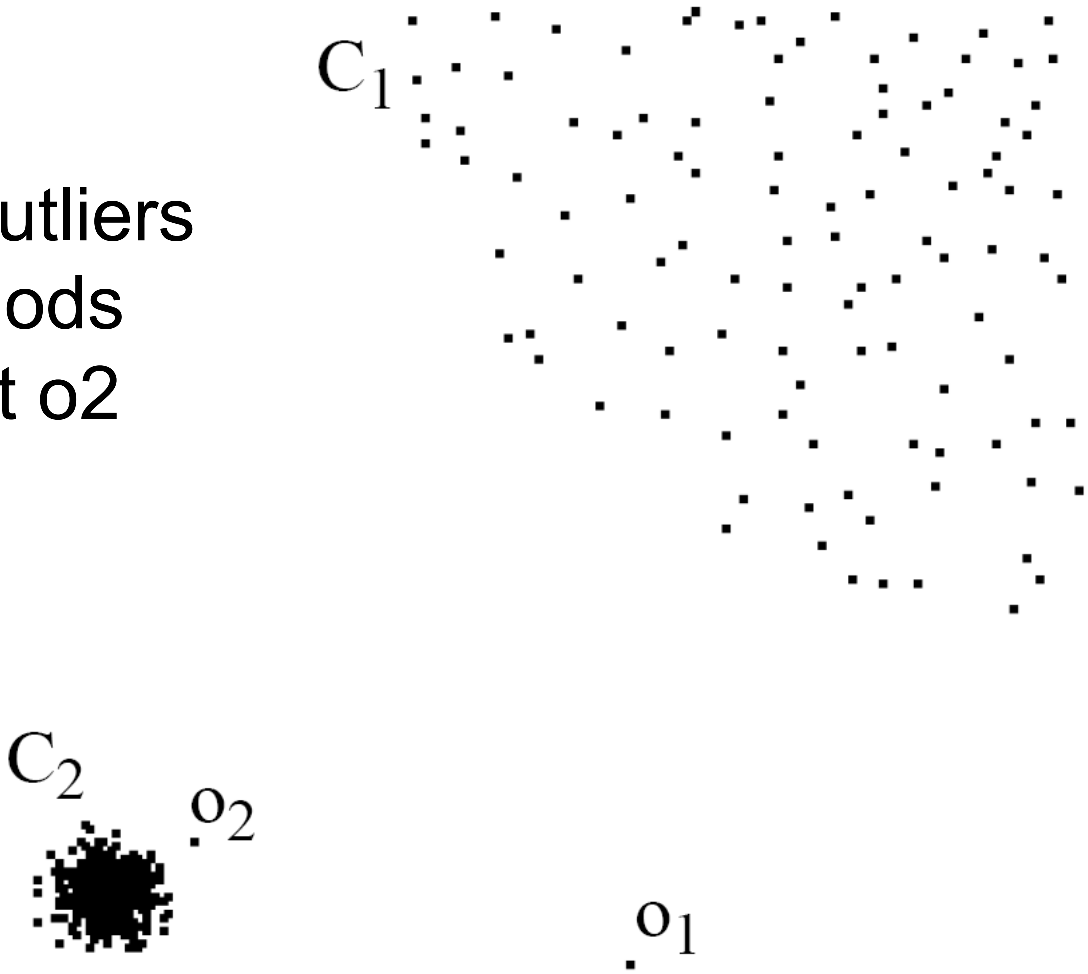
- Where do we need page reads?
  - Quantize objects to cells: 1 pass
  - Object-pairwise: many passes
- Idea: only keep white objects in main memory
  - White objects are in cells not red nor pink

Reducing Disk Reads

- Classify pages in datasets
  - A: contain some white objects
  - B: contain no white objects but L_2 neighbours of white objects
  - C: other pages
    - Object-pairwise don't need class C pages
- Scheduling pages A and B properly
- At most 3 passes

Density-based Local Outlier

Both o1 and o2 are outliers
Distance-based methods can detect o1, but not o2



Intuition

- Outliers comparing to their local neighborhoods, instead of the global data distribution
- The density around an outlier object is significantly different from the density around its neighbors
- Use the relative density of an object against its neighbors as the indicator of the degree of the object being outliers

K-Distance

- The k-distance of p is the distance between p and its k-th nearest neighbor
- In a set D of points, for any positive integer k, the k-distance of object p, denoted as k-distance(p), is the distance d(p, o) between p and an object o such that
  - For at least k objects $o' \in D \setminus \{p\}$, $d(p, o') \leq d(p, o)$
  - For at most (k-1) objects $o' \in D \setminus \{p\}$, $d(p, o') < d(p, o)$

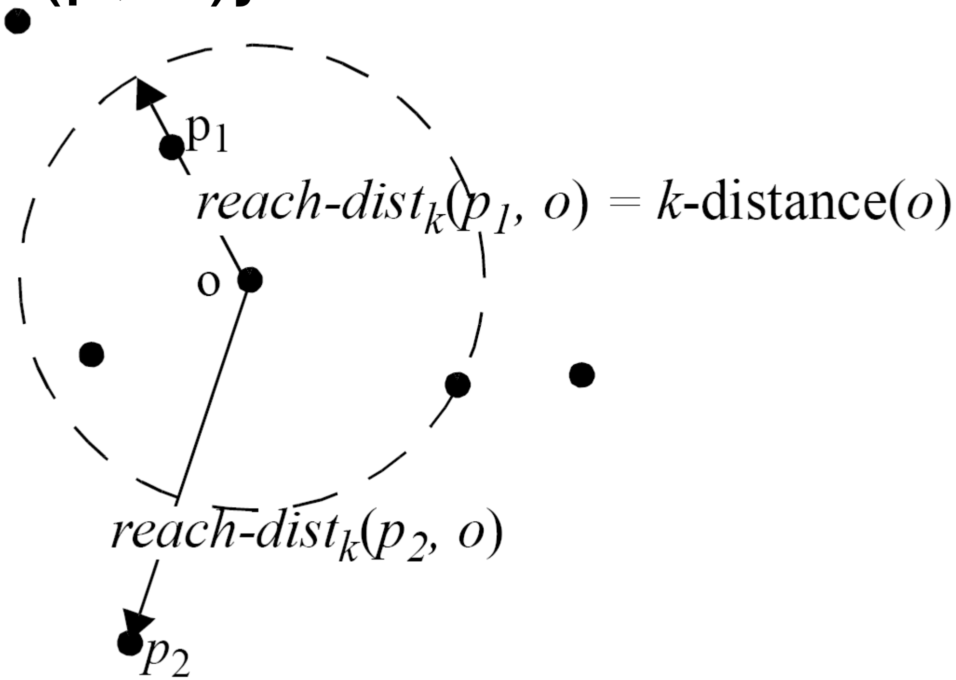
K-distance Neighborhood

- Given the k-stance of p, the k-distance neighborhood of p contains every object whose distance from p is not greater than the k-distance
  - $N_{k\text{-distance}(p)}(p) = \{q \in D \backslash \{p\} \mid d(p, q) \leq k\text{-distance}(p)\}$
  - $N_{k\text{-distance}(p)}(p)$ can be written as $N_k(p)$

Reachability Distance

- The reachability distance of object p with respect to object o is $\text{reach-dist}_k(p, o) = \max\{k\text{-distance}(o), d(p, o)\}$

If p and o are close to each other, reach-dist(p, o) is the k-distance, otherwise, it is the real distance



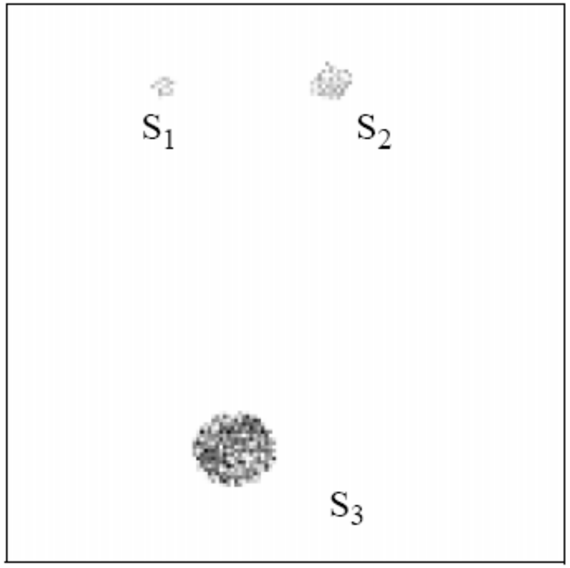
Local Reachability Density

$$lrd_k(o) = \frac{| N_k(o) |}{\sum_{o' \in N_k(o)} reachdist_k(o' \leftarrow o)}$$

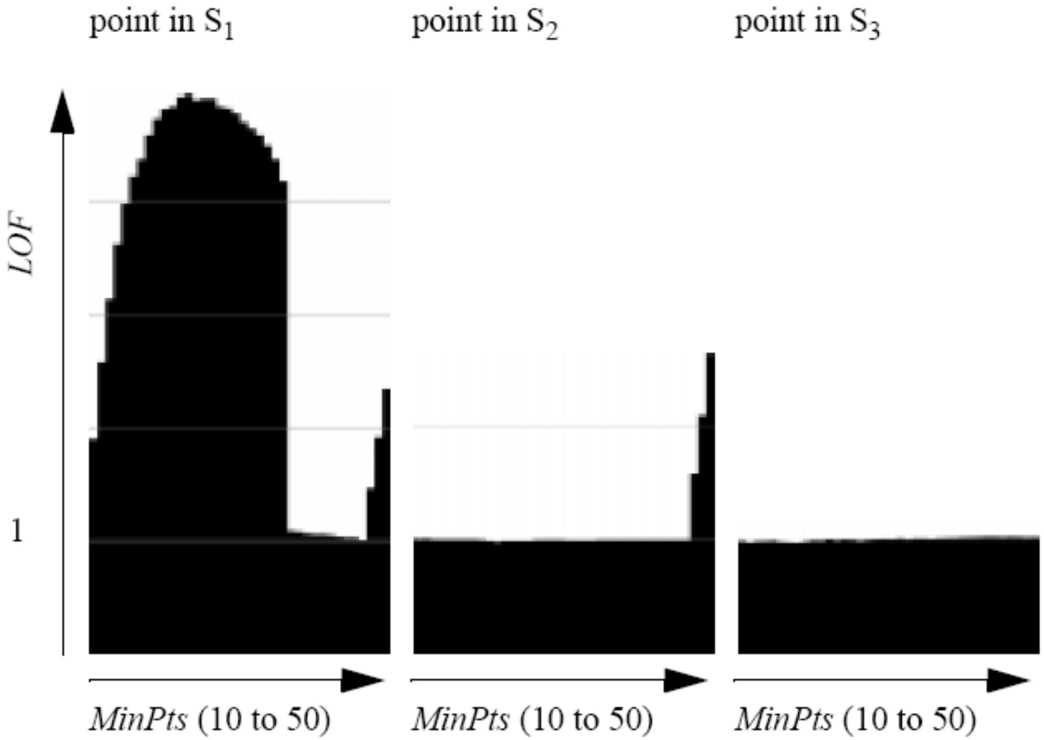
Local outlier factor

$$LOF_{MinPts}(p) = \frac{\sum_{o \in N_{MinPts}(p)} \frac{lrd_{MinPts}(o)}{lrd_{MinPts}(p)}}{|N_{MinPts}(p)|}$$

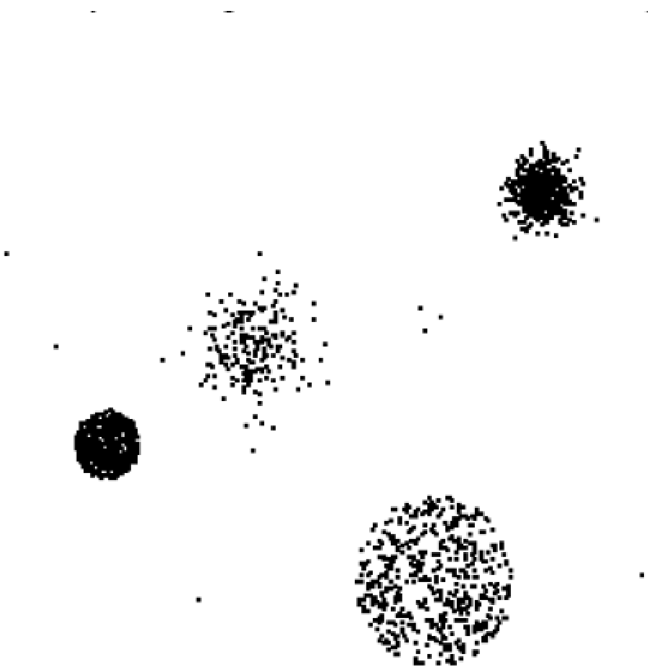
Examples

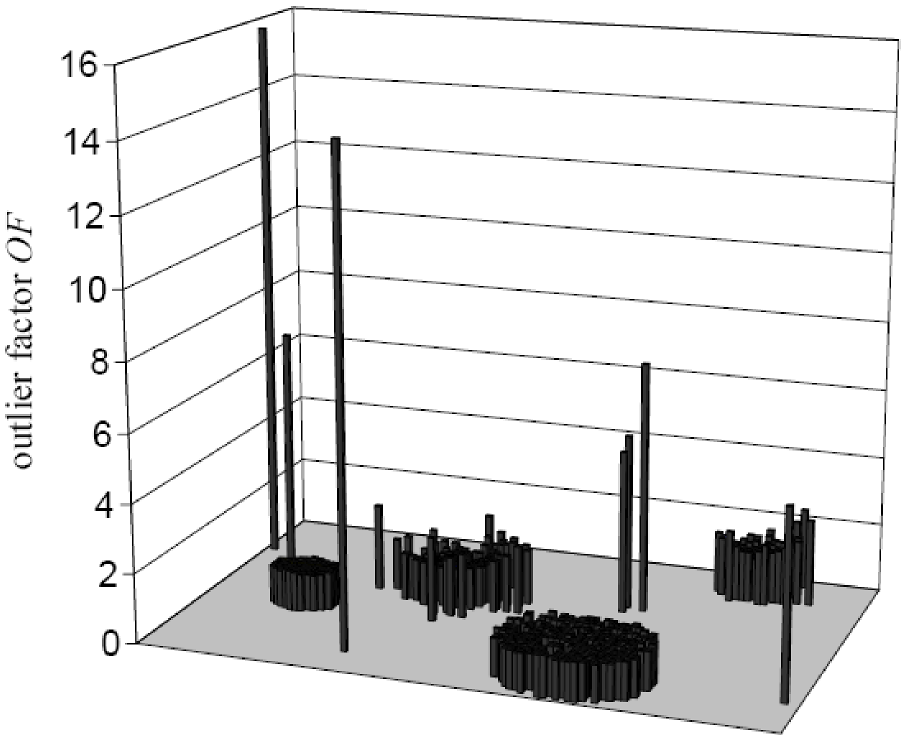


Example dataset



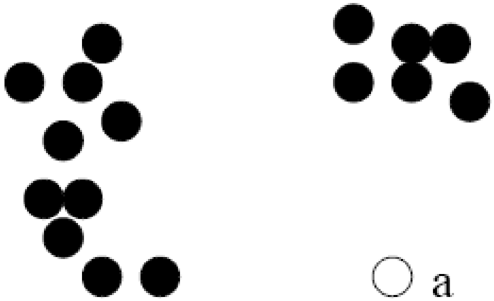
Examples

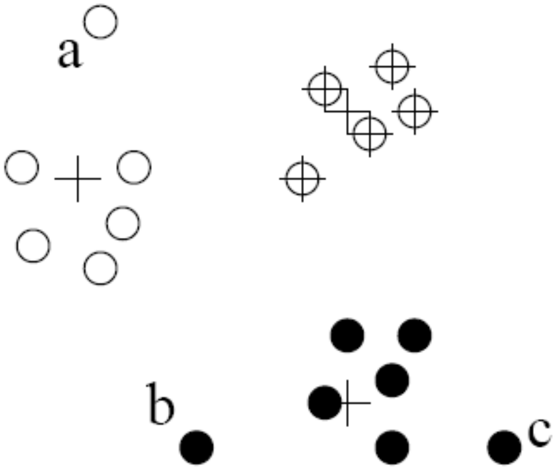


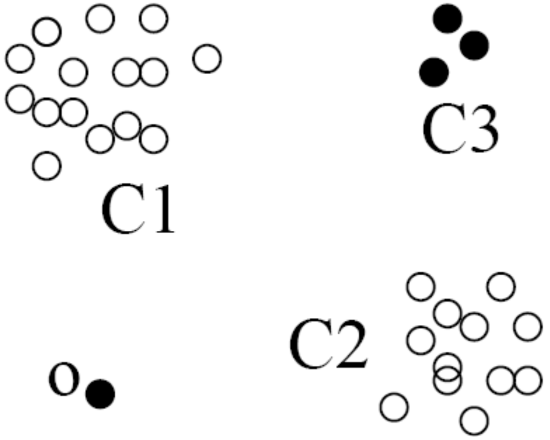


Clustering-based Outlier Detection

- An object is an outlier if
  - It does not belong to any cluster;
  - There is a large distance between the object and its closest cluster ; or
  - It belongs to a small or sparse cluster







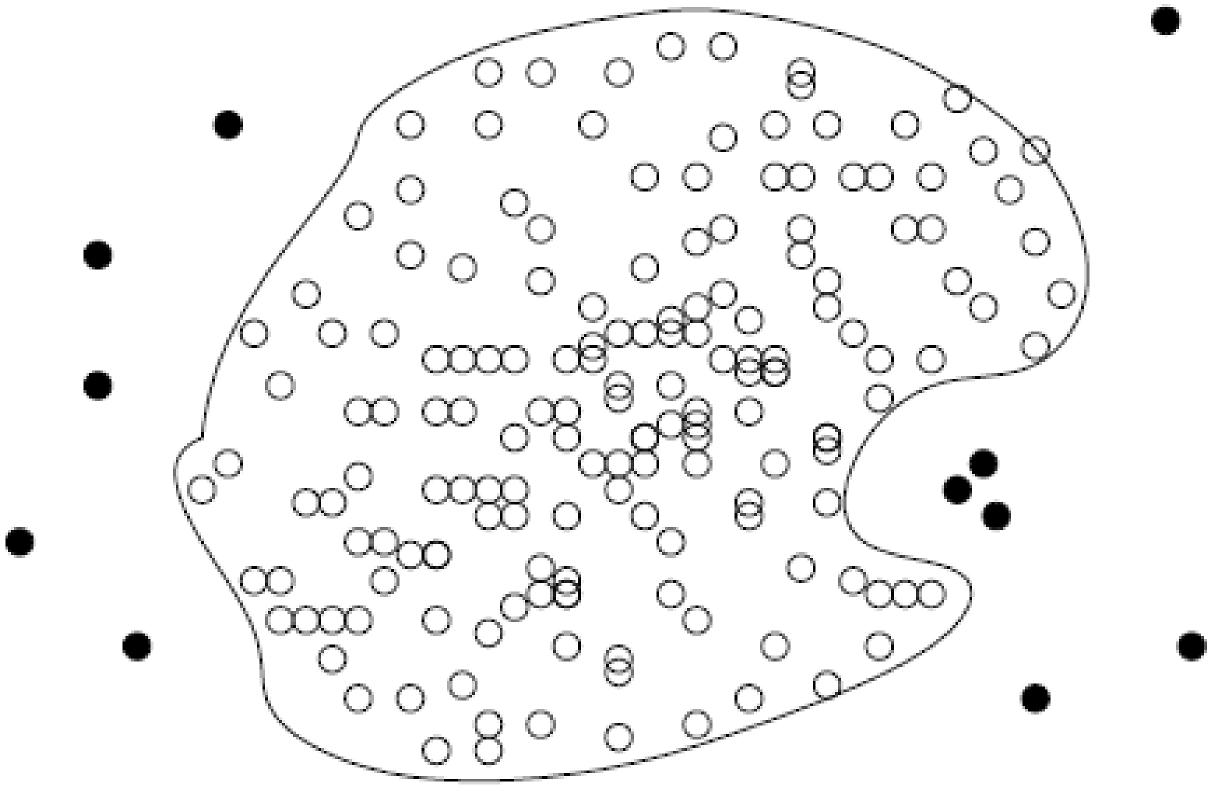
Classification-based Outlier Detection

- Train a classification model that can distinguish “normal” data from outliers
- A brute-force approach: Consider a training set that contains some samples labeled as “normal” and others labeled as “outlier”
  - A training set in practice is typically heavily biased: the number of “normal” samples likely far exceeds that of outlier samples
  - Cannot detect unseen anomaly

One-Class Model

- A classifier is built to describe only the normal class
- Learn the decision boundary of the normal class using classification methods such as SVM
- Any samples that do not belong to the normal class (not within the decision boundary) are declared as outliers
- Advantage: can detect new outliers that may not appear close to any outlier objects in the training set
- Extension: Normal objects may belong to multiple classes

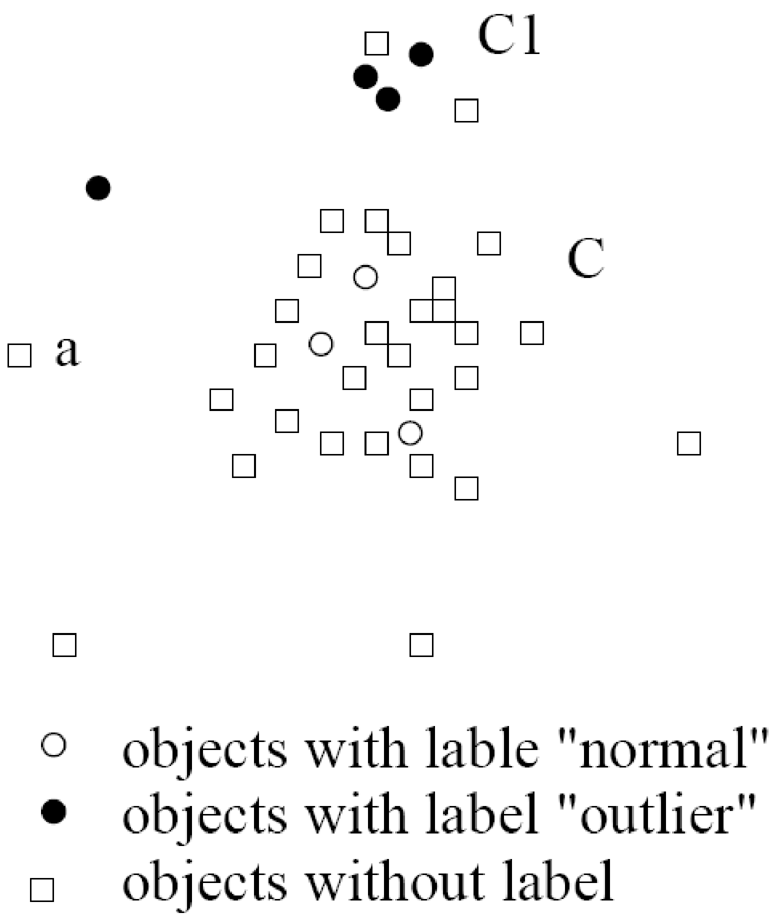
One-Class Model



Semi-Supervised Learning Methods

- Combine classification-based and clustering-based methods
- Method
  - Use a clustering-based approach to find a large cluster, C, and a small cluster, C1
  - Since some objects in C carry the label "normal", treat all objects in C as normal
  - Use the one-class model of this cluster to identify normal objects in outlier detection
  - Since some objects in cluster C1 carry the label "outlier", declare all objects in C1 as outliers
  - Any object that does not fall into the model for C (such as a) is considered an outlier as well

Example



C1
C
a
objects with lable "normal"
objects with label "outlier"
objects without label

Pros and Cons

- Pros: Outlier detection is fast
- Cons: Quality heavily depends on the availability and quality of the training set,
  - It is often difficult to obtain representative and high-quality training data

Contextual Outliers

- An outlier object deviates significantly based on a selected context
  - Ex. Is 10C in Vancouver an outlier? (depending on summer or winter?)
- Attributes of data objects should be divided into two groups
  - Contextual attributes: defines the context, e.g., time & location
  - Behavioral attributes: characteristics of the object, used in outlier evaluation, e.g., temperature
- A generalization of local outliers—whose density significantly deviates from its local area
- Challenge: how to define or formulate meaningful context?

Detection of Contextual Outliers

- If the contexts can be clearly identified, transform it to conventional outlier detection
  - Identify the context of the object using the contextual attributes
  - Calculate the outlier score for the object in the context using a conventional outlier detection method

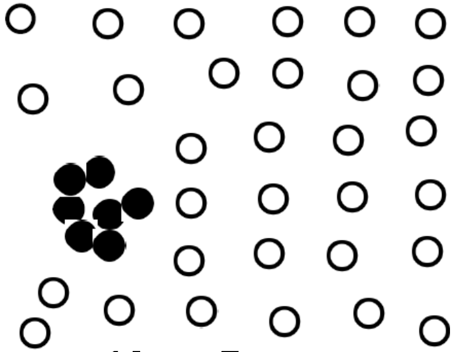
Example

- Detect outlier customers in the context of customer groups
  - Contextual attributes: age group, postal code
  - Behavioral attributes: the number of transactions per year, annual total transaction amount
- Method
  - Locate c's context;
  - Compare c with the other customers in the same group; and
  - Use a conventional outlier detection method

Modeling Normal Behavior

- Model the "normal" behavior with respect to contexts
  - Use a training data set to train a model that predicts the expected behavior attribute values with respect to the contextual attribute values
  - An object is a contextual outlier if its behavior attribute values significantly deviate from the values predicted by the model
- Use a prediction model to link the contexts and behavior
  - Avoid explicit identification of specific contexts
  - Some possible methods: regression, Markov Models, and Finite State Automaton …

Collective Outliers

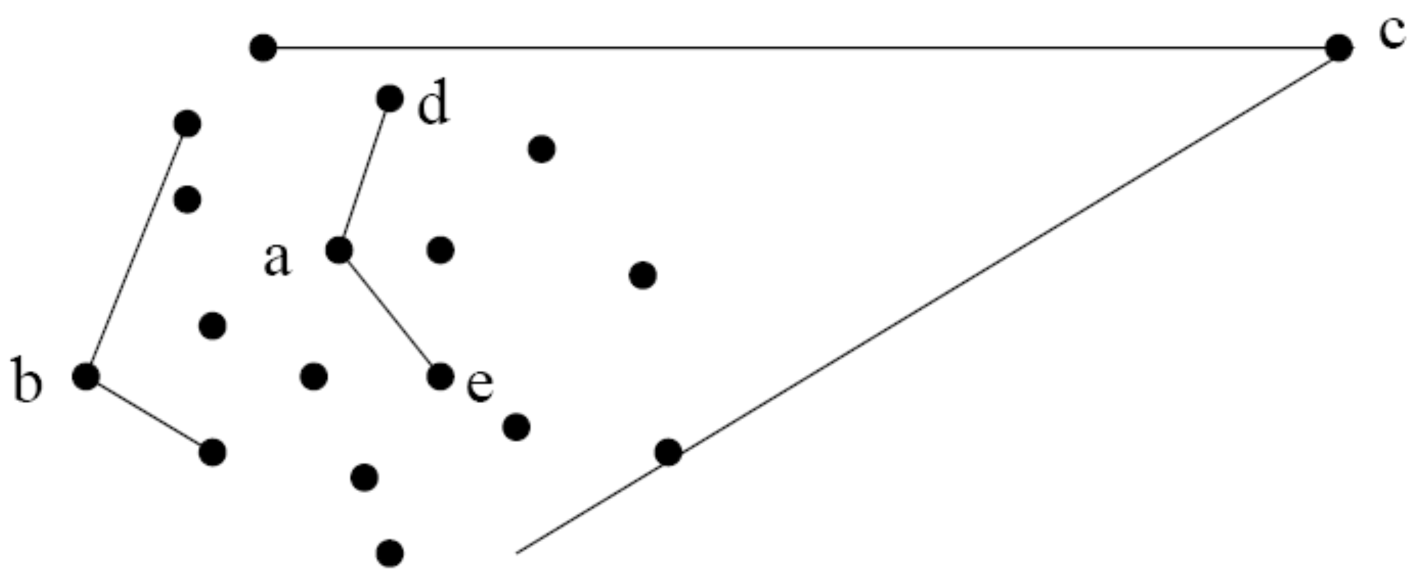


- Objects as a group deviate significantly from the entire data
- Examine the structure of the data set, i.e, the relationships between multiple data objects
  - The structures are often not explicitly defined, and have to be discovered as part of the outlier detection process.

Detecting High Dimensional Outliers

- Interpretability of outliers
  - Which subspaces manifest the outliers or an assessment regarding the “outlying-ness” of the objects
- Data sparsity: data in high-D spaces are often sparse
  - The distance between objects becomes heavily dominated by noise as the dimensionality increases
- Data subspaces
  - Local behavior and patterns of data
- Scalability with respect to dimensionality
  - The number of subspaces increases exponentially

Angle-based Outliers



To-Do List

- Read the rest of Chapter 12.4