

Outlier Detection

An Introduction to Outlier Detection, Methods to perform Outlier Detection and approaches to Outlier Detection in the context of Big Data

Introduction

- ▶ An outlier is an observation which deviates so much from the other observations as to arouse suspicions that it was generated by a different mechanism
- ▶ An outlier is a pattern in the data that does not conform to the expected behavior
- ▶ Outliers are frequently referred to as anomalies.
- ▶ Outlier Detection has various real-world applications such as cyber intrusions and credit card fraud



Applications of Outlier Detection

Fraud Detection

- ▶ Fraud detection refers to detection of criminal activities occurring in commercial organizations
- ▶ Malicious users might be the actual customers of the organization or might be posing as a customer (also known as identity theft). Examples of fraud are credit card fraud, insurance claim fraud, insider trading, etc.

Challenges

- ▶ Fast and accurate real-time detection
- ▶ Misclassification cost is very high

Anomaly detection can alleviate these limitations



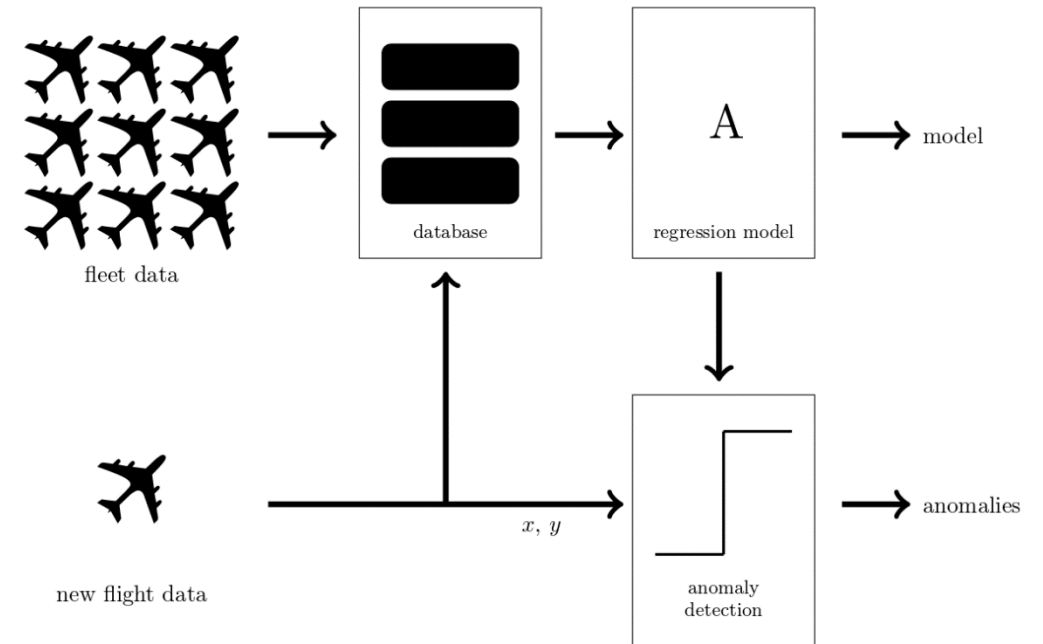
Applications of Outlier Detection

Industrial Damage Detection

- ▶ Industrial damage detection refers to detection of different faults and failures in complex industrial systems, structural damages, intrusions in electronic security systems, suspicious events in video surveillance, abnormal energy consumption, etc.
- ▶ Example: Aircraft Safety
 - ▶ Anomalous Aircraft (Engine) / Fleet Usage
 - ▶ Anomalies in engine combustion data
 - ▶ Total aircraft health and usage management

Challenges

- ▶ Data is extremely huge, noisy and unlabeled
- ▶ Detecting anomalous events typically require immediate intervention



Fleet Data Analysis and Anomaly Detection

Applications of Outlier Detection

Image Processing

- ▶ Detecting outliers in a image monitored over time
- ▶ Detecting anomalous regions within an image

Used in

- ▶ mammography image analysis
- ▶ video surveillance
- ▶ satellite image analysis

Challenges

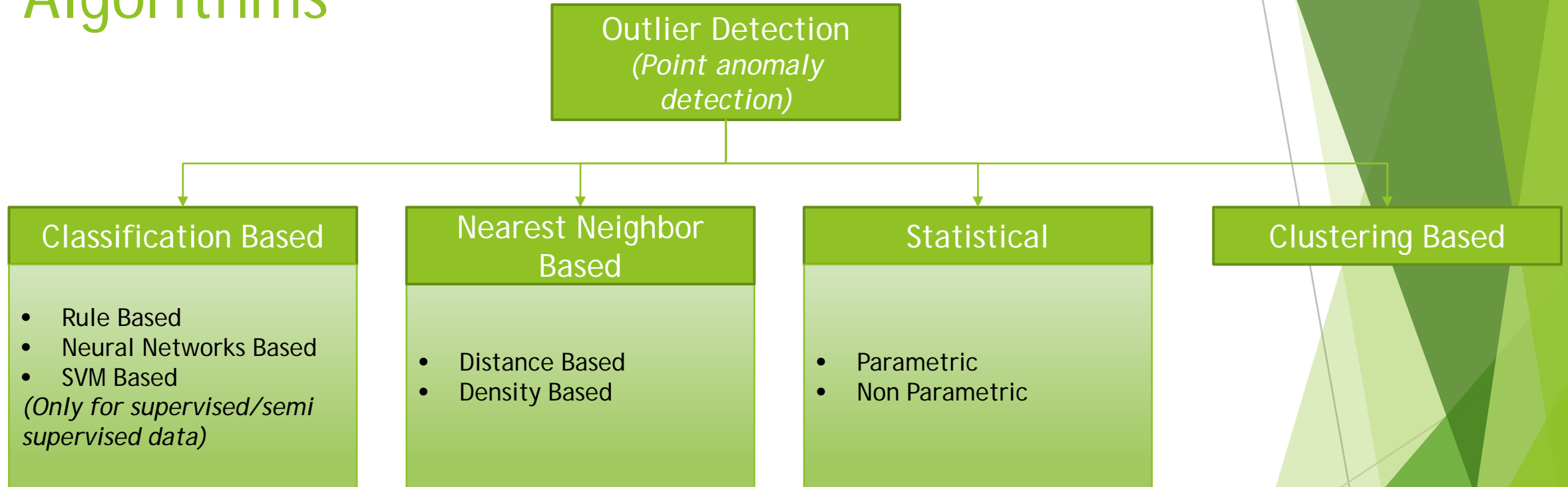
- ▶ Data is extremely huge, noisy and unlabeled



Classification of Outlier Detection Algorithms

- ▶ Based on whether previous examples of outliers can be obtained, outlier detection methods are of 3 types:
 1. Unsupervised Outlier Detection:
 - ▶ No labels on data.
 - ▶ Based on the assumption that anomalies are rare compared to data
 2. Supervised Outlier Detection:
 - ▶ Labels on data for both normal data and anomalous data points.
 - ▶ Based on the assumption that anomalies are rare compared to data
 3. Semi-Supervised Outlier Detection:
 - ▶ Labels on data for only normal data

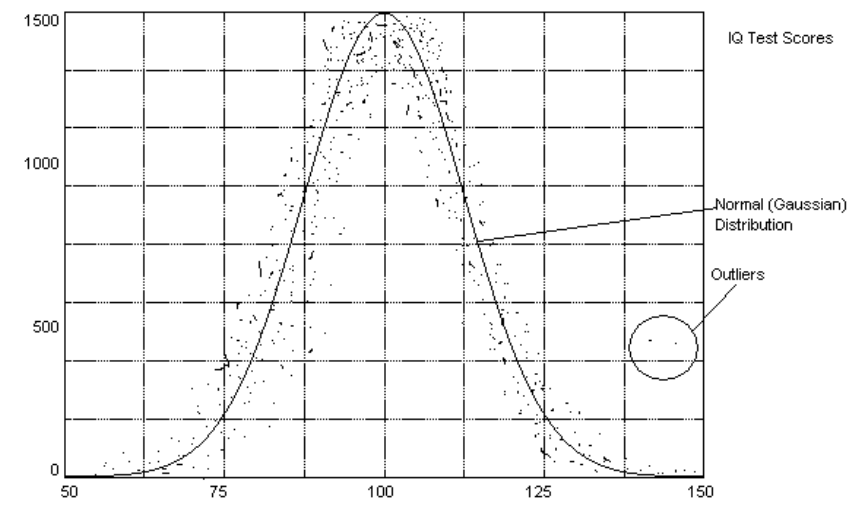
Taxonomy of Outlier Detection Algorithms



Statistical and Probabilistic Outlier Detection Tests

Statistical-Based Outlier Detection (Distribution-based)

- Assumptions:
 - Knowledge of data (distribution, mean, variance)
- Statistical discordancy test
 - Data is assumed to be part of a working hypothesis (working hypothesis)
 - Each data object in the dataset is compared to the working hypothesis and is either accepted in the working hypothesis or rejected as discordant into an alternative hypothesis (outliers)



Working Hypothesis: $H: o_i \in F$, where $i = 1, 2, \dots, n$.

Discordancy Test: is o_i in F within standard deviation = 15

Alternative Hypothesis:

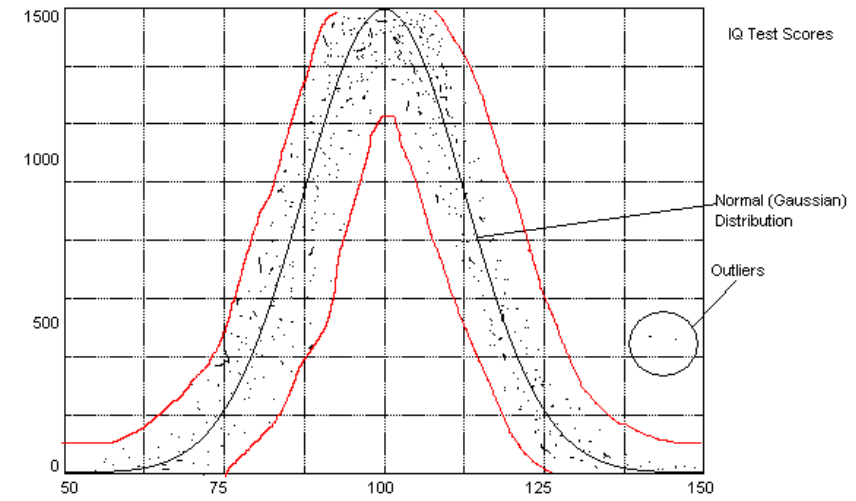
- Inherent Distribution: $\bar{H}: o_i \in G$, where $i = 1, 2, \dots, n$.

- Mixture Distribution: $\bar{H}: o_i \in (1 - \lambda)F + \lambda G$, where $i = 1, 2, \dots, n$.

- Slippage Distribution: $\bar{H}: o_i \in (1 - \lambda)F + \lambda F'$, where $i = 1, 2, \dots, n$.

Statistical-Based Outlier Detection (Distribution-based)

- Assumptions:
 - Knowledge of data (distribution, mean, variance)
- Statistical discordancy test
 - Data is assumed to be part of a working hypothesis (working hypothesis)
 - Each data object in the dataset is compared to the working hypothesis and is either accepted in the working hypothesis or rejected as discordant into an alternative hypothesis (outliers)



Working Hypothesis: $H: o_i \in F$, where $i = 1, 2, \dots, n$.

Discordancy Test: is o_i in F within standard deviation = 15

Alternative Hypothesis:

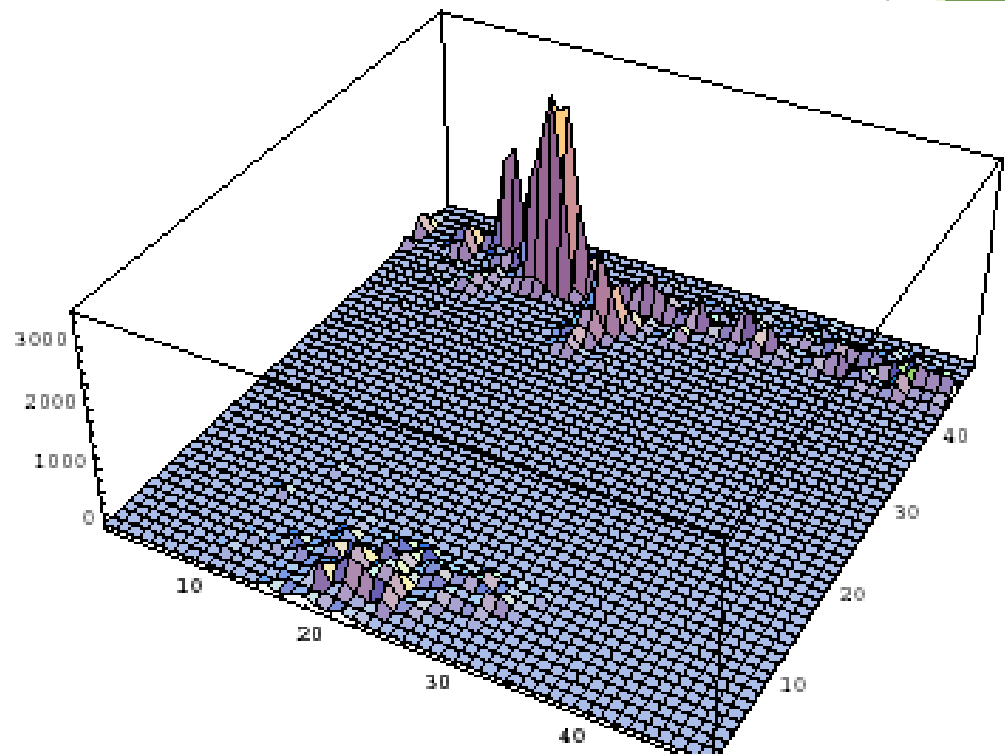
- Inherent Distribution: $\bar{H}: o_i \in G$, where $i = 1, 2, \dots, n$.

- Mixture Distribution: $\bar{H}: o_i \in (1 - \lambda)F + \lambda G$, where $i = 1, 2, \dots, n$.

- Slippage Distribution: $\bar{H}: o_i \in (1 - \lambda)F + \lambda F'$, where $i = 1, 2, \dots, n$.

Statistical-Based Outlier detection (Depth-based)

- Data is organized into layers according to some definition of depth
- Shallow layers are more likely to contain outliers than deep layers
- Can efficiently handle computation for $k < 4$



YU99 - FindOut: Finding Outliers in Very Large Datasets

Statistical-Based Outlier Detection

► Strengths

- Most outlier research has been done in this area, many data distributions are known

► Weakness

- Almost all of the statistical models are univariate (only handle one attribute) and those that are multivariate only efficiently handle $k < 4$
- All models assume the distribution is known -this is not always the case
- Outlier detection is completely subjective to the distribution used

Deviation-Based Outlier Detection

- Simulate a mechanism familiar to human being: after seeing a series of similar data, an element disturbing the series is considered an exception
- Sequential Exception Techniques
- OLAP Data Cube Techniques

Sequential Exception

- ▶ Select subsets of data I_j ($j=1,2,\dots,n$) from the dataset I
- ▶ Compare the dissimilarity of I and $(I - I_j)$ $\frac{1}{N} \times \sum_{i=1}^N (x_i - \bar{x})^2$
- ▶ Find out the minimum subset I_j that reduce the dissimilarity the most
- ▶ Smoothing factor
 - ▶ D is a dissimilarity function
 - ▶ C is a cardinality function, for example, the number of elements in the dataset

$$SF(I_j) := C(I - I_j) * (D(I) - D(I - I_j))$$

Example

Let the data set I be the set of integer values $\{1, 4, 4, 4\}$

$$SF(I_j) := C(I - I_j) * (D(I) - D(I - I_j))$$

I_j	$I - I_j$	$C(I - I_j)$	$D(I - I_j)$	$SF(I_j)$
$\{\}$	$\{1, 4, 4, 4\}$	4	1.69	0.00
$\{4\}$	$\{1, 4, 4\}$	3	2.00	-0.93
$\{4, 4\}$	$\{1, 4\}$	2	2.25	-1.12
$\{4, 4, 4\}$	$\{1\}$	1	0.00	1.69
$\{1\}$	$\{4, 4, 4\}$	3	0.00	5.07
$\{1, 4\}$	$\{4, 4\}$	2	0.00	3.38
$\{1, 4, 4\}$	$\{4\}$	1	0.00	1.69

Note, when $I_j = \{\}$, $D(I) = D(I - I_j) = 1.69$, $SF(I_j) = 0$

When $I_j = \{1\}$, $SF(I_j)$ has the maximum value, so $\{1\}$ is the outlier set

Outlier Detection in Univariate Data

- ▶ Z-value test
- ▶ Grubbs' test

Z-value test

- ▶ Given n values x_1, x_2, \dots, x_n having normal distribution with mean μ and standard deviation σ .
- ▶ Z-number z_i of an observed value x_i can be computed as follows:

$$z_i = (x_i - \mu) / \sigma$$

- ▶ Z-number corresponds to a scaled and translated normal random variable, which is also known as the standard normal distribution with mean 0 and variance 1

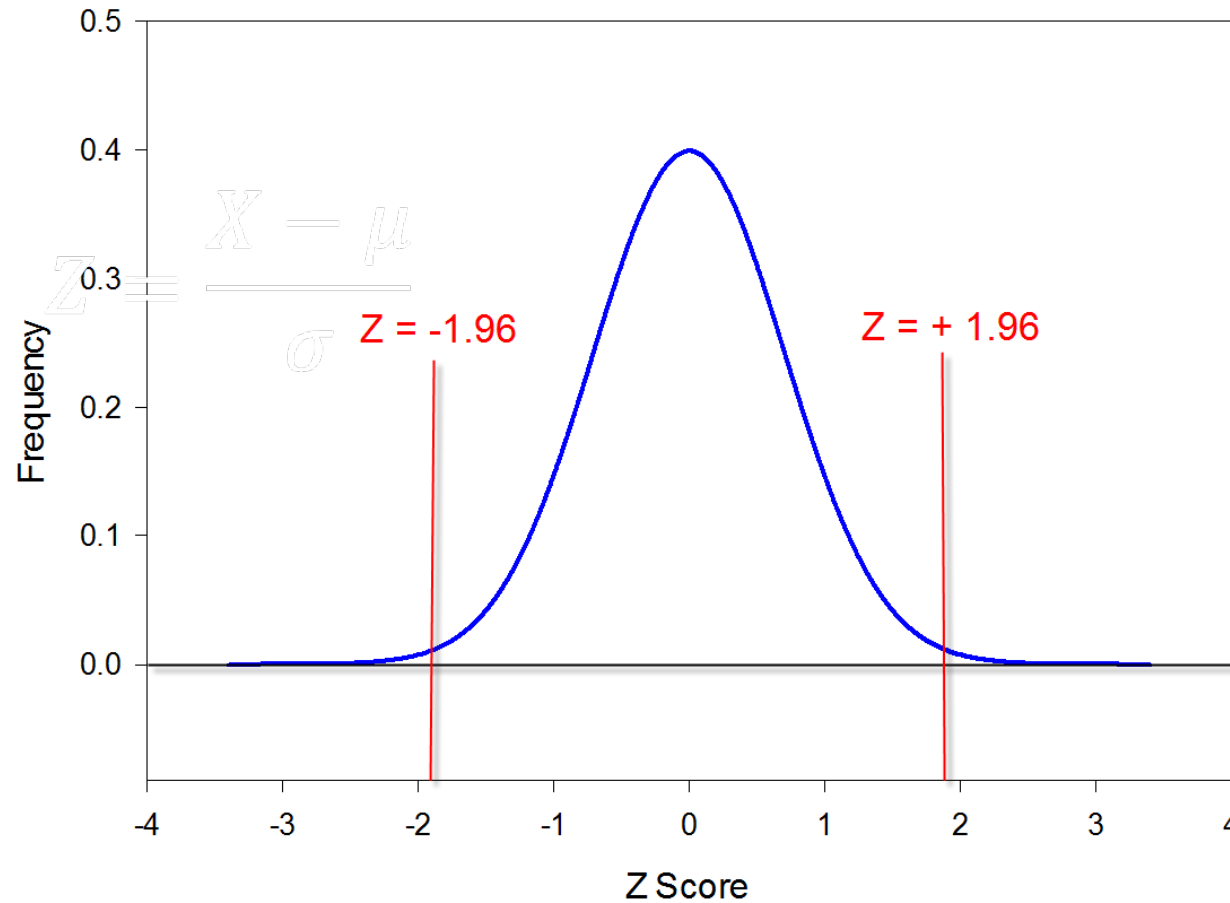
Z-value test

- ▶ Normal distribution tables are used in order to map the different values of z_i to probabilities.
- ▶ This provides a statistical level of significance, which can be interpreted directly as a probability of the data point being an outlier.
- ▶ In practise a point x_i may be classified as an outlier if $|z_i| > 2.5$.

Limitations of z-value test

- ▶ The mean and standard deviation of the distribution needs to be known known based on domain knowledge or can be estimated very accurately from a large number of samples.
- ▶ Does not function well for less number of data points.

A Z Score is a Standardized Statistic

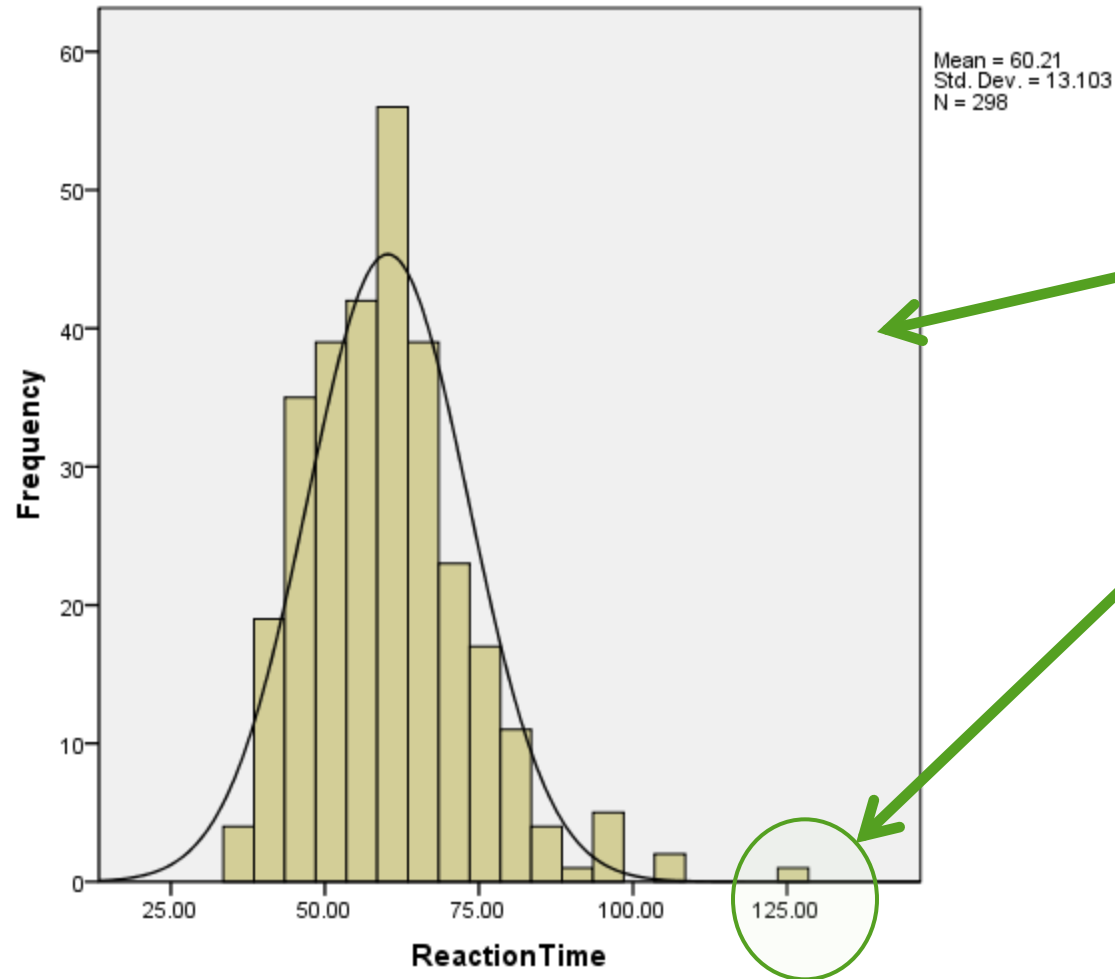


95.0% of the scores fall between a Z of -1.96 to +1.96

97.4% of the scores fall between a Z of -3.00 to +3.00

99.9% of the scores fall between a Z of -3.30 to +3.30

Histogram with Normal Curve



Seems to obey normal distribution

High Probability of being an outlier. Z value will be high

Z value test with map reduce

- $\mu = (\sum x_i)/n$ and $\sigma^2 = (\sum x_i^2)/n - \mu^2$.
- Since the formula's involve summations, they can be calculated using Map Reduce efficiently
- After both the values have been calculated, another Map Reduce task can be used to find the z score for each value and report the outliers having $|z_i| > T$, where T is some threshold determined.

Grubbs' test

1. Check most extreme value for being an outlier.
2. If it is, remove it.
3. Check for the next extreme value using the new, smaller sample. It is smaller because the first outlier was removed.
4. Repeat the process.

Grubbs' test

Grubbs' test is defined for the hypothesis:

H_0 :	There are no outliers in the data set
H_a :	There is exactly one outlier in the data set

Test Statistic:

The Grubbs' test statistic is defined as: $G = \max |Y_i - \bar{Y}| / s$ with \bar{Y} and s denoting the sample mean and standard deviation, respectively. The Grubbs' test statistic is the largest absolute deviation from the sample mean in units of the sample standard deviation

Test whether for $Y_i = Y_{\min}$ $G > G_{\text{critical}}$
and whether for $Y_i = Y_{\max}$ $G > G_{\text{critical}}$

Grubbs' test

Significance Level:

α

Critical Region:

For the two-sided test, the hypothesis of no outliers is rejected if

$$G > \frac{(N-1)}{\sqrt{N}} \sqrt{\frac{(t_{\alpha/(2N), N-2})^2}{N-2 + (t_{\alpha/(2N), N-2})^2}}$$

with $t_{\alpha/(2N), N-2}$ denoting the critical value of the t distribution with $(N-2)$ degrees of freedom and a significance level of $\alpha/(2N)$.

Extreme Value Analysis in Multivariate Data

- ▶ Deviation based Methods
- ▶ Angle based Outlier Detection
- ▶ Distance Distribution-based Methods

Deviation based Methods

- ▶ Define smoothing factor $SF(R)$ of a dataset D to be the reduced data set variance when set of points in R are removed from the data.
- ▶ Outliers are defined as the exception set E such that their removal causes maximum reduction in variance. i.e. $SF(E) \geq SF(R)$ for every subset R .
- ▶ If more than one set have same reduction in variance then the smaller one is preferred.

Limitation of Deviation Based Method

- Determination of optimal set is a difficult problem as 2^N possibilities exist for a data set of N points.

Pros of Deviation based method

- ▶ One good aspect of this approach is that it is distribution dependent and can be applied to any kind of data set (as long as definition of smoothing factor can be constructed).

Angle Based Outlier Detection

- ▶ Data points lying on the boundaries of data are likely to enclose the entire data within a smaller angle.
- ▶ Consider three data points X , Y and Z . Then the angle between the vectors $Y-X$ and the $Z-X$, will not vary much for different values of Y and Z , when X is an outlier.

Angle Based Outlier Detection

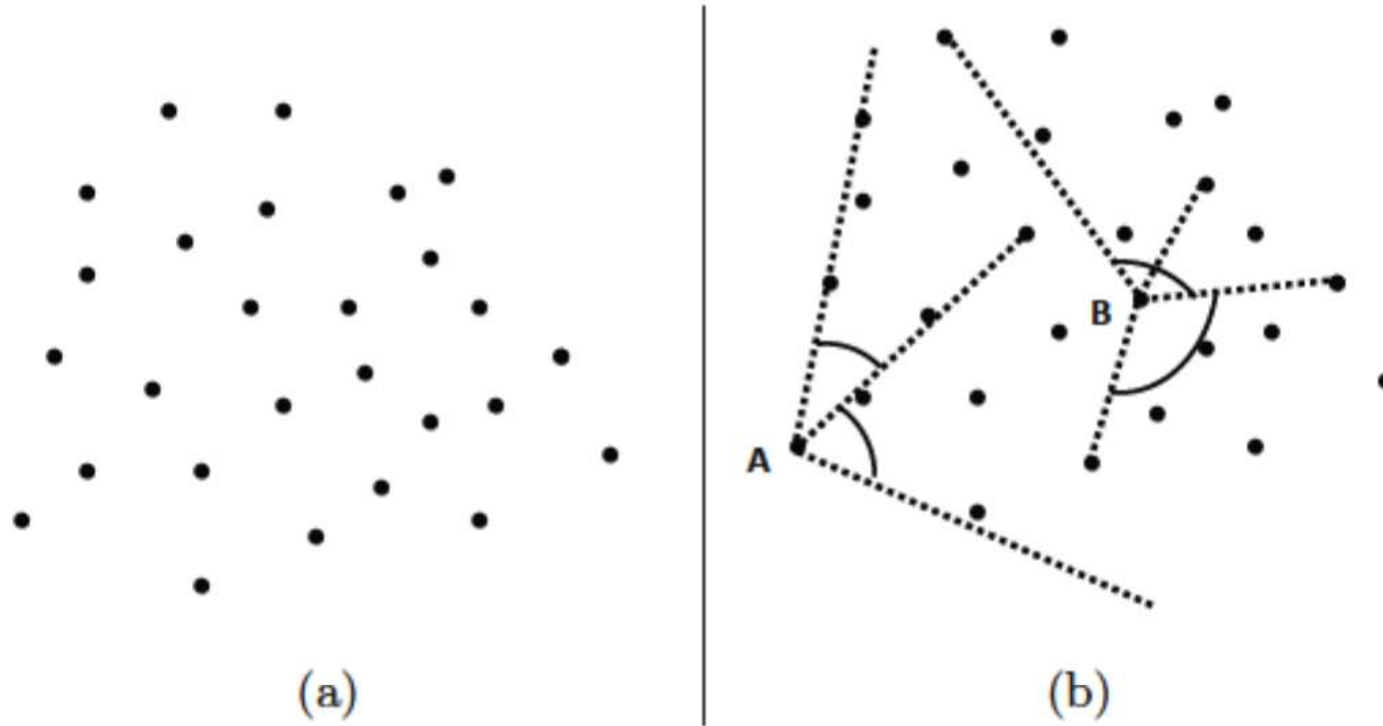
- The weighted cosine is defined as

$$WCos(\bar{Y} - \bar{X}, \bar{Z} - \bar{X}) = \frac{\langle (\bar{Y} - \bar{X}), (\bar{Z} - \bar{X}) \rangle}{\|\bar{Y} - \bar{X}\|_2^2 \cdot \|\bar{Z} - \bar{X}\|_2^2}$$

- $\|\cdot\|_2$ represents L_2 norm and $\langle \cdot, \cdot \rangle$ represents scalar product.
- The angle based outlier factor is defined as

$$ABOF(\bar{X}) = Var_{\{Y, Z \in \mathcal{D}\}} WCos(\bar{Y} - \bar{X}, \bar{Z} - \bar{X})$$

Angle Based Outlier Detection



- $ABOF(A)$ is clearly less than $ABOF(B)$ in this image. Hence A is more likely to be reported as an outlier.

Angle Based Outlier Detection

- ▶ Data points which are outliers will have smaller spectrum angles.
- ▶ In order to speed up computations a sample of k points can be used to be used in the ABOF function

Limitations

- ▶ Exact correct computation of ABOF give as $O(N^3)$ time complexity.
- ▶ In most data sets outliers lie not just on the boundaries of the data, but also in the interior of the data which are not likely to be reported by this approach.

Distance Distribution-based Methods

- ▶ A distribution-dependent approach is to model the entire data set to be normally distributed about its mean in the form of a multivariate Gaussian distribution.
- ▶ Let μ be the d -dimensional mean vector of a d -dimensional data set, and Σ be its $d \times d$ co-variance matrix.
- ▶ In this case, the (i, j) th entry of the covariance matrix is equal to the covariance between the dimensions i and j .

Distance Distribution-based Methods

- Then, the probability distribution $f(X)$ for a d -dimensional data point X can be defined as follows:

$$f(X) = \frac{1}{\sqrt{|\Sigma|} \cdot (2 \cdot \pi)^{(d/2)}} \cdot \exp\left(-\frac{1}{2} \cdot (X - \mu) \cdot \Sigma^{-1} \cdot (X - \mu)^T\right)$$

- Values of $f(X)$ will be used to classify the point as an outlier.

The mean vector is $\boldsymbol{\mu} = \begin{bmatrix} \mu_{x_1} \\ \mu_{x_2} \\ \vdots \\ \mu_{x_V} \end{bmatrix}$

The covariance matrix is $\boldsymbol{\Sigma} = \begin{bmatrix} \sigma_{x_1}^2 & \sigma_{x_1 x_2} & \cdots & \sigma_{x_1 x_V} \\ \sigma_{x_1 x_2} & \sigma_{x_2}^2 & \cdots & \sigma_{x_2 x_V} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{x_1 x_V} & \sigma_{x_2 x_V} & \cdots & \sigma_{x_V}^2 \end{bmatrix}$

- ▶ μ_{x_i} = mean of attribute x_i over all samples
- ▶ $\text{Cov}(X, Y) = \sigma_{XY} = E[(X - \mu_X)(Y - \mu_Y)]$ between any two variables X and Y

Pros

- ▶ The Mahalanobis method is robust to increasing dimensionality, because it uses the covariance matrix in order to summarize the high dimensional deviations in a statistically effective way.
- ▶ The process involves a lot of matrix multiplication all of which can be done with Map Reduce for large data

Probabilistic Mixture Modeling for Outlier Analysis

- ▶ The broad principle of a mixture based generative model is to assume that the data was generated from a mixture of k distributions with the probability distributions $G_1 \dots G_k$ with the use of the following process:
- ▶ Pick a data distribution with probability α_i , where $i \in \{1 \dots k\}$, in order to pick one of the k distributions. Let us assume that the r th one is picked.
- ▶ Generate a data point from G_r .

Probabilistic Mixture Modeling for Outlier Analysis

- ▶ We denote this generative model by M
- ▶ The different values of α_i , and the parameters of the different distributions G_r are not known in advance.
- ▶ In some simplified settings, the values of the prior Probabilities α_i may be fixed to $1/k$, though this value also needs to be learned from the data in the most general case.
- ▶ Typical forms for the distribution of G_r include the Gaussian distribution. These need to be estimated from the data, so that the data has the maximum likelihood fit of being generated.

Probabilistic Mixture Modeling for Outlier Analysis

- ▶ Let us assume that the density function of G_i is given by $f_i(\cdot)$. The probability of the data point X_j being generated by the model is given by:
- ▶ Then, for a data set D containing N records the probability of the data set being generated by the model M is:

$$f^{point}(\overline{X_j}|\mathcal{M}) = \sum_{i=1}^k \alpha_i \cdot f^i(\overline{X_j})$$

Probabilistic Mixture Modeling for Outlier Analysis

$$f^{data}(\mathcal{D}|\mathcal{M}) = \prod_{j=1}^N f^{point}(\overline{X}_j|\mathcal{M})$$

- The log-likelihood $\text{fitL}(\mathcal{D}|\mathcal{M})$ of the data set \mathcal{D} with respect to \mathcal{M} is the logarithm of the above expression and can be (more conveniently) represented as a sum of values over the different data points.

$$\mathcal{L}(\mathcal{D}|\mathcal{M}) = \log\left(\prod_{j=1}^N f^{point}(\overline{X}_j|\mathcal{M})\right) = \sum_{j=1}^N \log\left(\sum_{i=1}^k \alpha_i \cdot f^i(\overline{X}_j)\right)$$

Probabilistic Mixture Modeling for Outlier Analysis

- ▶ This log-likelihood fit needs to be optimized to determine the model parameters, and therefore maximize the fit of the data points to the generative model.
- ▶ There are two ways to do this:
- ▶ **E-Step:** simply computes the probability density of the data point X_j being generated by each component of the mixture, and then computes the fractional value for each component. This provides the Bayes probability that the data point X_j was generated by component i (with model parameters fixed to the current set of the parameters Θ). Therefore, we have:

Probabilistic Mixture Modeling for Outlier Analysis

$$P(\overline{X_j} \in \mathcal{G}_i | \Theta) = \frac{\alpha_i \cdot f^{i, \Theta}(\overline{X_j})}{\sum_{r=1}^k \alpha_r \cdot f^{r, \Theta}(\overline{X_j})}$$

- **M-Step:** In order to optimize the fit, we need to compute the partial derivative of the log-likelihood fit with respect to corresponding model parameters, and set them to 0 in order to determine the optimal value.

- ▶ The values of α_i are easy to estimate
- ▶ and simply equal to the expected fraction of the points assigned to each
- ▶ cluster, based on the current values of $P(X_j \in G_i | \Theta)$. In practice, in
- ▶ order to obtain more robust results for smaller data sets, the expected number of data points belonging to each cluster in the numerator is augmented by 1, and the total number of points in the denominator is $N+k$. Therefore, the estimated value is:

$$\frac{(1 + \sum_{j=1}^N P(X_j \in G_i | \Theta))}{k + N}$$

- ▶ This approach is also referred to as Laplacian smoothing.

Limitations

- ▶ When the particular assumptions of the model are too restrictive, the data is unlikely to fit the model well. As a result, a lot of spurious data points may be reported as outliers.
- ▶ When the model is too general, the number of parameters to describe the model increases. This may overfit the data, and miss the true outliers. This case is more common in parametric modeling, especially when the data sizes are small.

Nearest Neighbor based Outlier Detection

Distance-based Approaches

- ▶ General Idea
 - ▶ Judge a point based on the distance(s) to its neighbors
 - ▶ Several variants proposed
- ▶ Basic Assumption
 - ▶ Normal data objects have a dense neighborhood
 - ▶ Outliers are far apart from their neighbors, i.e., have a less dense neighborhood

Distance-based Approaches

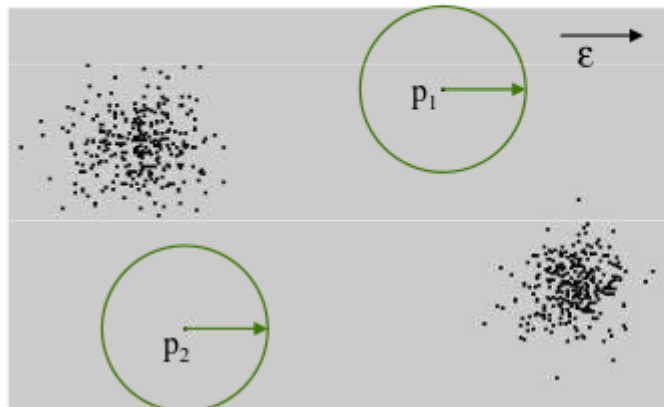
► DB(ε, π)-Outliers

► Basic model [Knorr and Ng 1997]

- ❖ Given a radius ε and a percentage π
- ❖ A point p is considered an outlier if at most π percent of all other points have a distance to p less than ε

$$OutlierSet(\varepsilon, \pi) = \{p \mid \frac{Card(\{q \in DB \mid dist(p, q) > \varepsilon\})}{Card(DB)} < \pi\}$$

range-query with radius ε



Distance-based Approaches

► Algorithms

- **Index-based** [Knorr and Ng 1998]
 - Compute distance range join using spatial index structure
 - Exclude point from further consideration if its ε -neighborhood contains more than $Card(DB) \cdot \pi$ points
- **BNested-loop based** [Knorr and Ng 1998]
 - Divide buffer in two parts
 - Use second part to scan/compare all points with the points from the first part
- **Grid-based** [Knorr and Ng 1998]
 - Build grid such that any two points from the same grid cell have a distance of at most ε to each other
 - Points need only compared with points from neighboring cells

Index-based Algorithm [KN98]

- Indexing Structures such as R-tree (R+-tree), K-D (K-D-B) tree are built for the multi-dimensional database
- The index is used to search for neighbors of each object O within radius D around that object.
- Once K ($K = N(1-p)$) neighbors of object O are found, O is not an outlier.
- Worst-case computation complexity is $O(K \cdot n^2)$, K is the dimensionality and n is the number of objects in the dataset.
- **Pros:** scale well with K
- **Cons:** the index construction process may cost much time

Nested-loop Algorithm

- Divides the buffer space into two halves (first and second arrays)
- Break data into blocks and then feed two blocks into the arrays.
- Directly computes the distance between each pair of objects, inside the array or between arrays
- Decide the outlier. Same computational complexity as the index-based algorithm
- Pros: Avoid index structure construction
- Try to minimize the I/Os

Example - stage 1

Buffer

A
B

DB

A	B
C	D

Starting Point of Stage 1

A
D

A	B
C	D

End Point of Stage 1

A is the target block on stage 1

Load A into the first array (1R)

Load B into the second array (1R)

Load C into the second array (1R)

Load D into the second array (1R)

Total: 4 Reads

Example - stage 2

Buffer

A
D

Starting Point of Stage 2

DB

A	B
C	D

C
D

End Point of Stage 2

A	B
C	D

D is the target block on stage 2

D is already in the buffer (no R)

A is already in the buffer (no R)

Load B into the first array (1R)

Load C into the first array (1R)

Total: 2 Reads

Example - stage 3

Buffer

C
D

Starting Point of Stage 3

DB

A	B
C	D

C
B

End Point of Stage 3

A	B
C	D

C is the target block on stage 3

C is already in the buffer (no R)

D is already in the buffer (no R)

Load A into the second array
(1R)

Load B into the second array
(1R)

Total: 2 Reads

Example - stage 4

Buffer

C
B

DB

A	B
C	D

Starting Point of Stage 4

D
B

A	B
C	D

End Point of Stage 4

B is the target block on stage 4

B is already in the buffer (no R)

C is already in the buffer (no R)

Load A into the first array (1R)

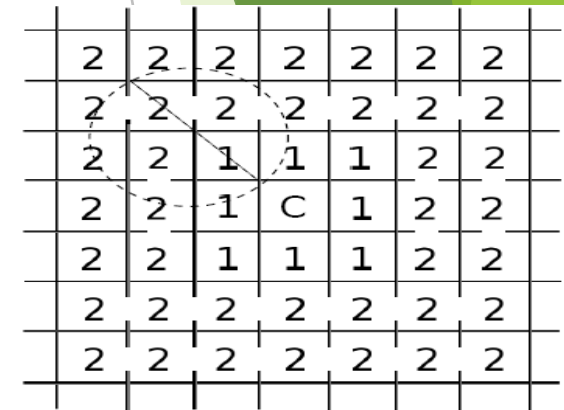
Load D into the first array (1R)

Total: 2 Reads

Every block is $\frac{1}{4}$ of the DB. From stage 1-4, a grand total of 10 blocks are read, amounting to $10/4$ passes over the entire dataset.

Distance-Based Outlier Detection: A Grid-Based Method

- ❖ Why efficiency is still a concern? When the complete set of objects cannot be held into main memory, cost I/O swapping
- ❖ The major cost: (1) each object tests against the whole data set, why not only its close neighbor? (2) check objects one by one, why not group by group?
- ❖ Grid-based method (CELL): Data space is partitioned into a multi-D grid. Each cell is a hyper cube with diagonal length $r/2$
- Pruning using the level-1 & level 2 cell properties:
 - For any possible point x in cell C and any possible point y in a level-1 cell, $\text{dist}(x,y) \leq r$
 - For any possible point x in cell C and any point y such that $\text{dist}(x,y) \geq r$, y is in a level-2 cell



2	2	2	2	2	2	2	
2	2	2	2	2	2	2	
2	2	1	1	1	2	2	
2	2	1	C	1	2	2	
2	2	1	1	1	2	2	
2	2	2	2	2	2	2	
2	2	2	2	2	2	2	

Thus we only need to check the objects that cannot be pruned, and even for such an object o , only need to compute the distance between o and the objects in the level-2 cells (since beyond level-2, the distance from o is more than r)

Distance-based Approaches

- ▶ Deriving intensional knowledge [Knorr and Ng 1999]
 - Relies on the $DB(\varepsilon, \pi)$ -outlier model
 - Find the minimal subset(s) of attributes that explains the “outlierness” of a point, i.e., in which the point is still an outlier

Distance-based Approaches

- ▶ Outlier scoring based on k NN distances
 - ▶ General models
 - Take the k NN distance of a point as its outlier score [Ramaswamy et al 2000]
 - Aggregate the distances of a point to all its 1NN, 2NN, ..., k NN as an outlier score [Angiulli and Pizzuti 2002]
 - ▶ Algorithms
 - ▶ General approaches
 - ▶ Nested-Loop
 - Naïve approach:
For each object: compute k NNs with a sequential scan
 - Enhancement:
use index structures for k NN queries
 - ▶ Partition-based
 - Partition data into micro clusters
 - Aggregate information for each partition (e.g. minimum bounding rectangles)
 - Allows to prune micro clusters that cannot qualify when searching for the k NNs of a particular point

Distance-based Approaches

- ▶ Sample Algorithms (computing top- n outliers)
 - ▶ Nested-Loop [Ramaswamy et al 2000]
 - Simple NL algorithm with index support for k NN queries
 - Partition-based algorithm (based on a clustering algorithm that has linear time complexity)
 - Algorithm for the simple k NN-distance model
 - ▶ Linearization [Angiulli and Pizzuti 2002]
 - Linearization of a multi-dimensional data set using space-fill curves
 - 1D representation is partitioned into micro clusters
 - Algorithm for the average k NN-distance model
 - ▶ ORCA [Bay and Schwabacher 2003]
 - NL algorithm with randomization and simple pruning
 - Pruning: if a point has a score greater than the top- n outlier so far (cut-off), remove this point from further consideration
 - ▶ => non-outliers are pruned
 - ▶ => works good on randomized data (can be done in linear time)
 - ▶ => worst-case: naïve NL algorithm
 - Algorithm for both k NN-distance models and the $DB(\epsilon, \pi)$ -outlier model

Distance-based Approaches

□ Sample Algorithms (cont.)

- RBRP [Ghoting et al. 2006],
 - Idea: try to increase the cut-off as quick as possible => increase the pruning power
 - Compute approximate k NNs for each point to get a better cut-off
 - For approximate k NN search, the data points are partitioned into micro clusters and k NNs are only searched within each micro cluster
 - Algorithm for both k NN-distance models
- Further approaches
 - Also apply partitioning-based algorithms using micro clusters [McCallum et al 2000], [Tao et al. 2006]
 - Approximate solution based on reference points [Pei et al. 2006]

□ Discussion

- Output can be a scoring (k NN-distance models) or a labeling (k NN-distance models and the $DB(\epsilon, \pi)$ -outlier model)
- Approaches are local (resolution can be adjusted by the user via ϵ or k)

Distance-based Approaches

□ Variant

▪ Outlier Detection using In-degree Number [Hautamaki et al. 2004]

• Idea

- Construct the k NN graph for a data set
 - ▶ Vertices: data points
 - ▶ Edge: if q belongs to k NN(p) then there is a directed edge from p to q
- A vertex that has an indegree less than equal to T (user defined threshold) is an outlier

• Discussion

- The indegree of a vertex in the k NN graph equals to the number of reverse kNNs (RkNN) of the corresponding point
- The RkNNs of a point p are those data objects having p among their k NNs
- Intuition of the model: outliers are
 - ▶ points that are among the k NNs of less than T other points have less than T RkNNs
- Outputs an outlier label
- Is a local approach (depending on user defined parameter k)

Distance-based Approaches

□ Resolution-based outlier factor (ROF) [Fan et al. 2006]

► Model

- Depending on the resolution of applied distance thresholds, points are outliers or within a cluster
- With the maximal resolution R_{max} (minimal distance threshold) all points are outliers
- With the minimal resolution R_{min} (maximal distance threshold) all points are within a cluster
- Change resolution from R_{max} to R_{min} so that at each step at least one point changes from being outlier to being a member of a cluster
- Cluster is defined similar as in DBSCAN [Ester et al 1996] as a transitive closure of r -neighborhoods (where r is the current resolution)
- ROF value

□ Discussion

- Outputs a score (the ROF value)
- Resolution is varied automatically from local to global

Limitations of distance-based approach

- ▶ When it comes to Big data problem, the complexity $O(n^2)$ is not at all affordable as n is very large. Hence the distance-based approach is not very suitable.

Density-based Approaches

- General idea
 - ❖ Compare the density around a point with the density around its local neighbors
 - ❖ The relative density of a point compared to its neighbors is computed as an outlier score
 - ❖ Approaches essentially differ in how to estimate density
- Basic assumption
 - The density around a normal data object is similar to the density around its neighbors
 - The density around an outlier is considerably different to the density around its neighbors

Density-based Approaches

- Local Outlier Factor (LOF) [Breunig et al. 1999], [Breunig et al. 2000]

- ❖ Motivation:

- ❖ Distance-based outlier detection models have problems with different densities
- ❖ How to compare the neighborhood of points from areas of different densities?
- ❖ Example

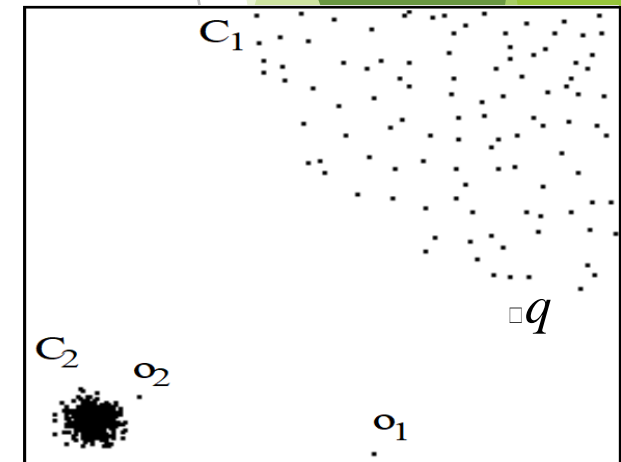
- DB(ϵ , π)-outlier model

- Parameters ϵ and π cannot be chosen
 - so that o_2 is an outlier but none of the
 - points in cluster C_1 (e.g. q) is an outlier

- Outliers based on kNN-distance

- kNN-distances of objects in C_1 (e.g. q)
 - are larger than the kNN-distance of o_2

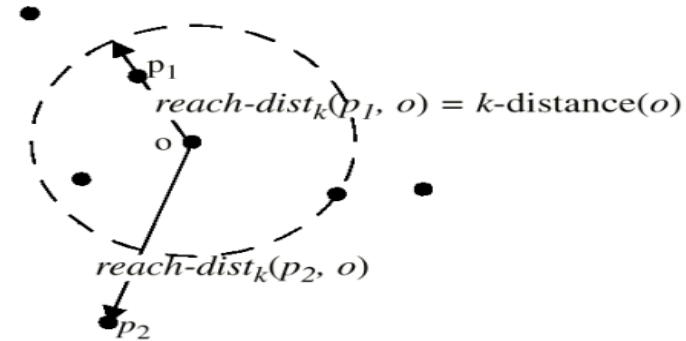
- ❑ Solution: consider relative density



Density-based Approaches

□ Model

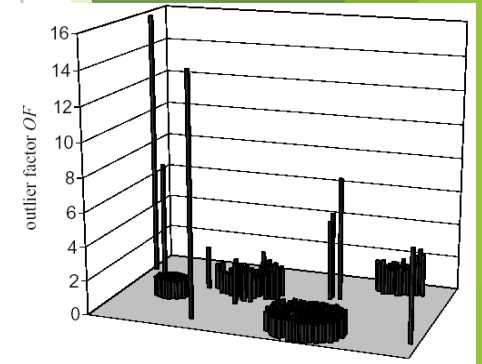
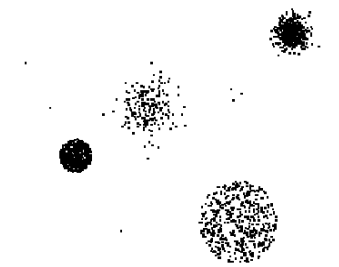
- Reachability distance
 - Introduces a smoothing factor
- Local reachability distance (lrd) of point p
 - Inverse of the average reach-dists of the k NNs of p
- Local outlier factor (LOF) of point p
 - Average ratio of lrd's of neighbors of p and lrd of p



Density-based Approaches

□ Properties

- $LOF \approx 1$: point is in a cluster (region with homogeneous density around the point and its neighbors)
- $LOF \gg 1$: point is an outlier



□ Discussion

- Choice of k (*MinPts* in the original paper) specifies the reference set
- Originally implements a local approach (resolution depends on the user's choice for k)
- Outputs a scoring (assigns an LOF value to each point)

Density-based Approaches

▣ Variants of LOF

➤ Mining top- n local outliers [Jin et al. 2001]

❖ Idea:

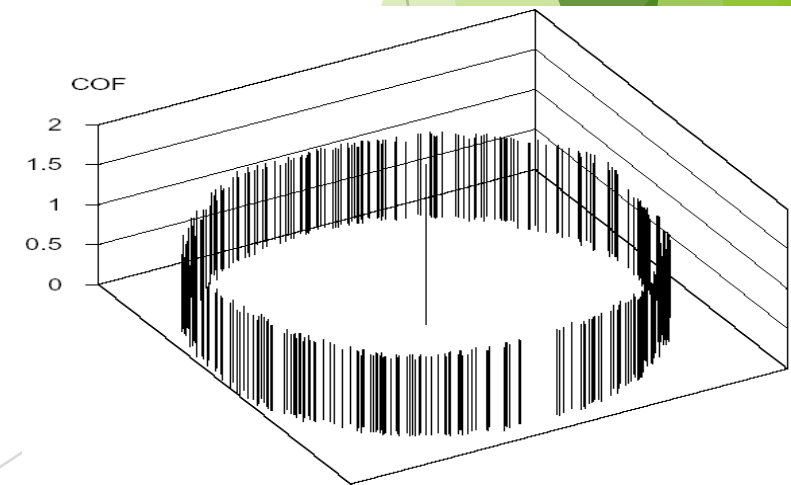
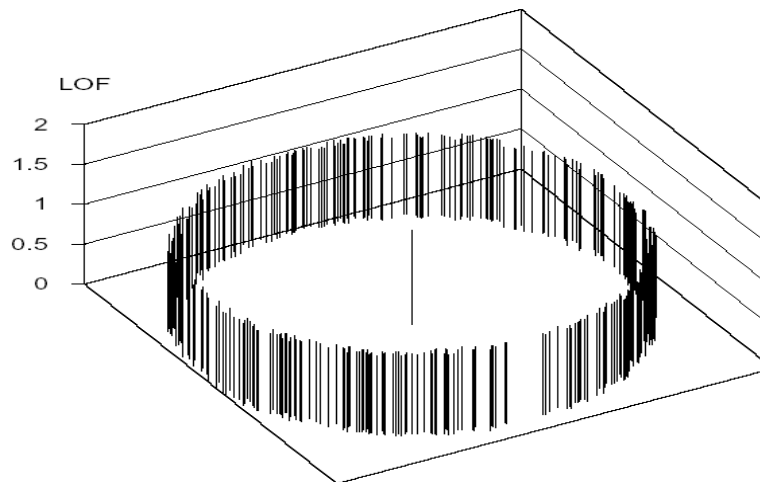
- Usually, a user is only interested in the top- n outliers
- Do not compute the LOF for all data objects => save runtime

❖ Method

- Compress data points into micro clusters using the CFs of BIRCH [Zhang et al. 1996]
- Derive upper and lower bounds of the reachability distances, lrd-values, and LOF-values for points within a micro clusters
- Compute upper and lower bounds of LOF values for micro clusters and sort results w.r.t. ascending lower bound
- Prune micro clusters that cannot accommodate points among the top- n outliers (n highest LOF values)
- Iteratively refine remaining micro clusters and prune points accordingly

Density-based Approaches

- ❑ Variants of LOF (cont.)
 - ❖ Connectivity-based outlier factor (COF) [Tang et al. 2002]
 - Motivation
 - In regions of low density, it may be hard to detect outliers
 - Choose a low value for k is often not appropriate
 - Solution
 - Treat “low density” and “isolation” differently
 - Example

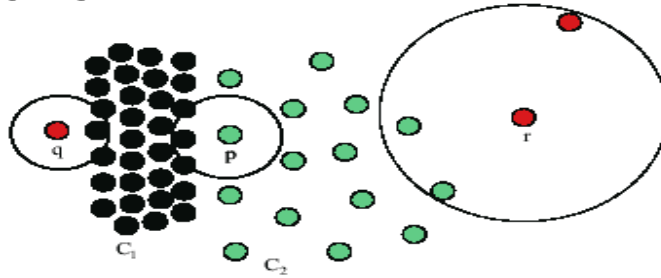


Density-based Approaches

□ Influenced Outlierness (INFLO) [Jin et al. 2006]

➤ Motivation

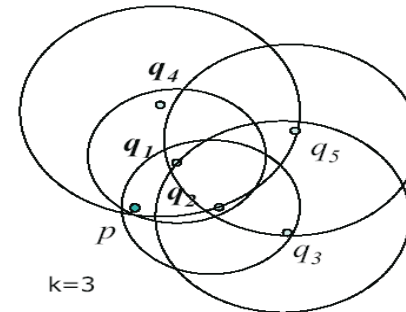
- If clusters of different densities are not clearly separated, LOF will have problems



Point p will have a higher LOF than points q or r which is counter intuitive

➤ Idea

- Take symmetric neighborhood relationship into account
- Influence space ($kIS(p)$) of a point p includes its $kNNs$ ($kNN(p)$) and its reverse $kNNs$ ($RkNN(p)$)



$$\begin{aligned} kIS(p) &= kNN(p) \cup RkNN(p) \\ &= \{q_1, q_2, q_4\} \end{aligned}$$

Density-based Approaches

□ Model

- Density is simply measured by the inverse of the k NN distance, i.e.,
 - ▶ $den(p) = 1/k\text{-distance}(p)$
- Influenced outlieriness of a point p
- INFLO takes the ratio of the average density of objects in the neighborhood of a point p (i.e., in $kNN(p) \cup RkNN(p)$) to p 's density

□ Proposed algorithms for mining top- n outliers

- Index-based
- Two-way approach
- Micro cluster based approach

Density-based Approaches

□ Properties

- Similar to LOF
- $\text{INFLO} \approx 1$: point is in a cluster
- $\text{INFLO} \gg 1$: point is an outlier

□ Discussion

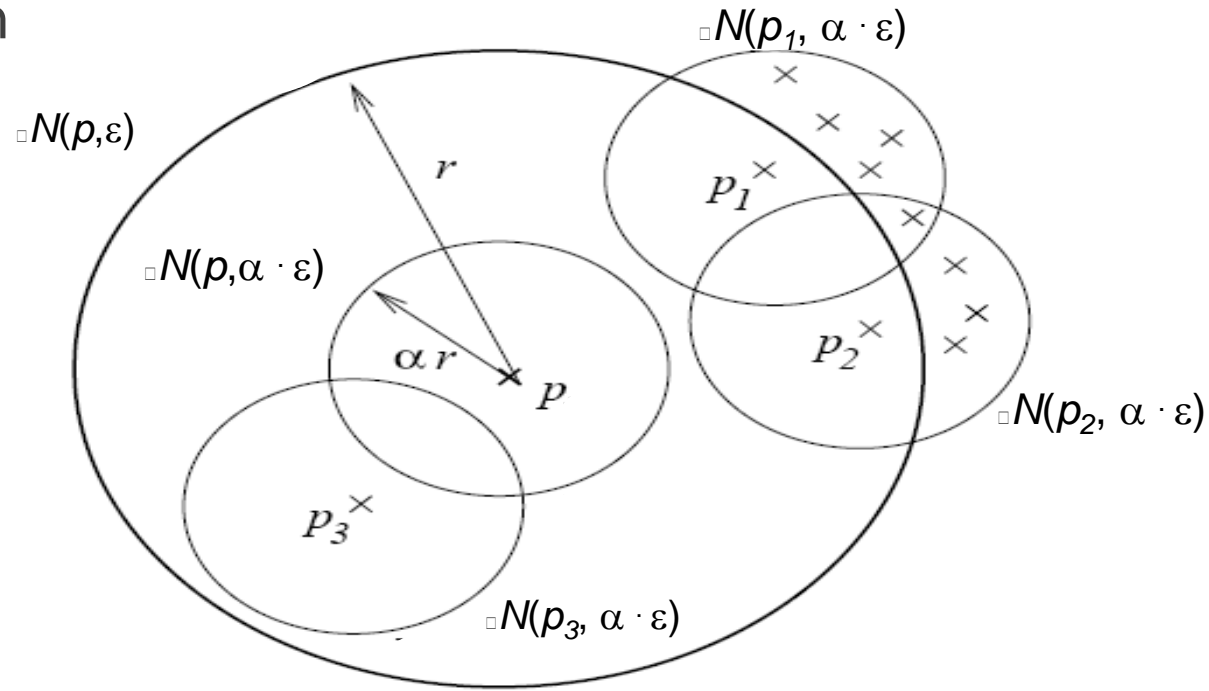
- Outputs an outlier score
- Originally proposed as a local approach (resolution of the reference set kIS can be adjusted by the user setting parameter k)

Density-based Approaches

- Local outlier correlation integral (LOCI) [Papadimitriou et al. 2003]
 - Idea is similar to LOF and variants
 - Differences to LOF
 - Take the ε -neighborhood instead of k NNs as reference set
 - Test multiple resolutions (here called “granularities”) of the reference set to get rid of any input parameter
 - Model ε -neighborhood of a point p : $N(p, \varepsilon) = \{q \mid \text{dist}(p, q) \leq \varepsilon\}$
 - Local density of an object p : number of objects in $N(p, \varepsilon)$
 - Average density of the neighborhood
 - Multi-granularity Deviation Factor (MDEF)

Density-based Approaches

Intuition



□ $\sigma\text{MDEF}(p, \varepsilon, \alpha)$ is the normalized standard deviation of the densities of all points from $N(p, \varepsilon)$

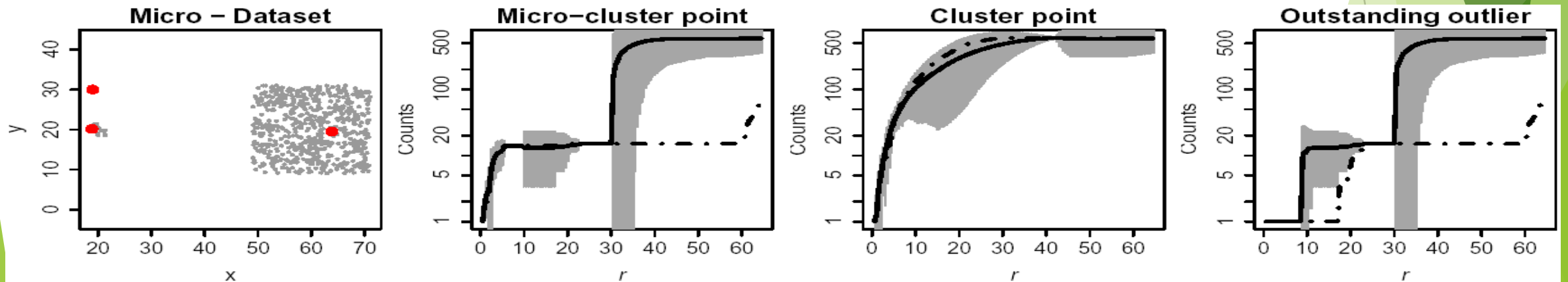
Properties

- $\text{MDEF} = 0$ for points within a cluster
- $\text{MDEF} > 0$ for outliers or $\text{MDEF} > 3 \cdot \sigma\text{MDEF} \Rightarrow \text{outlier}$

Density-based Approaches

□ Features

- Parameters ε and α are automatically determined
- In fact, all possible values for ε are tested
- LOCI plot displays for a given point p the following values w.r.t. ε
 - ▶ $Card(N(p, \alpha \cdot \varepsilon))$
 - ▶ $den(p, \varepsilon, \alpha)$ with a border of $\pm 3 \cdot \sigma den(p, \varepsilon, \alpha)$



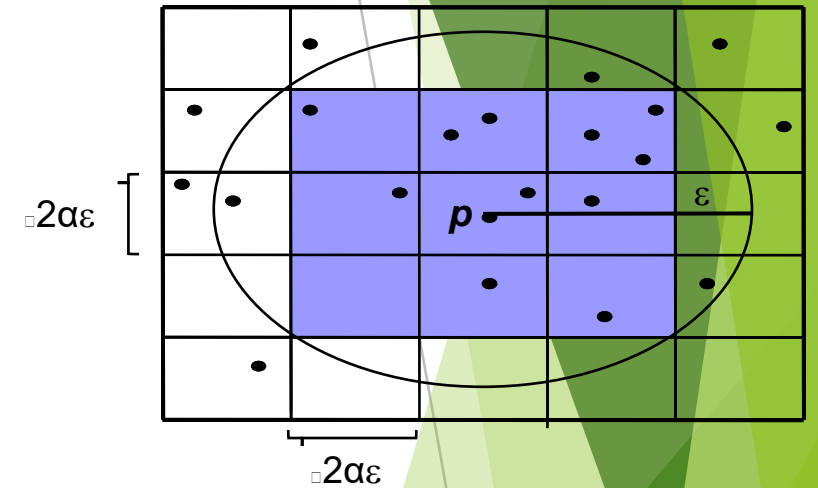
Density-based Approaches

□ Algorithms

- ❖ Exact solution is rather expensive (compute MDEF values for all possible ε values)
- ❖ aLOCI: fast, approximate solution
 - Discretize data space using a grid with side length $2\alpha\varepsilon$
 - Approximate range queries through grid cells
 - ε - neighborhood of point p : $\zeta(p, \varepsilon)$
 - all cells that are completely covered by ε -sphere around p
 - Then,

$$Card(N(q, \alpha \cdot \varepsilon)) = \frac{\sum_{c_j \in \zeta(p, \varepsilon)} c_j^2}{\sum_{c_j \in \zeta(p, \varepsilon)} c_j}$$

- where c_j is the object count the corresponding cell
- Since different ε values are needed, different grids are constructed with varying resolution
- These different grids can be managed efficiently using a Quad-tree



Density-based Approaches

□ Discussion

- ❖ Exponential runtime w.r.t. data dimensionality
- ❖ Output:
 - Score (MDEF) or
 - Label: if MDEF of a point $> 3\sigma\text{MDEF}$ then this point is marked as outlier
 - LOCI plot
 - At which resolution is a point an outlier (if any)
 - Additional information such as diameter of clusters, distances to clusters, etc.
- ❖ All interesting resolutions, i.e., possible values for ϵ , (from local to global) are tested

Limitations of the density-based approach

- The major drawback of LOF algorithm lies in computing reachability distances.
- Computing reachability distance involves computing distances of all objects within the neighbourhood, and each compared with the k-distance of that neighbourhood which is very expensive.
- Secondly, LOF has to be computed for every object before the few outliers are detected. This is not a desirable exercise since outliers constitute only small fraction of the entire dataset.

An example of outlier detection using Hadoop Mapreduce

- ▶ We have a very simple data model. Each credit card transaction contains the following 4 attributes
 1. Transaction ID
 2. Time of the day
 3. Money spent
 4. Vendor type
- ▶ Here are some examples. The last one is an outlier, injected into the data set.

YX66AJ9U 1025 20.47	drug store
98ZCM6B1 1910 55.50	restaurant
XXXX7362 0100 1875.40	jewellery store

Distance Calculation

- ▶ For **numerical attribute** (e.g. money amount), distance is the difference in values
- ▶ For **unranked categorical attribute** (e.g. vendor type), the distance is 0 if they are same and 1 otherwise. The distances could be set softly between 0 and 1 (e.g product color).
- ▶ If the unranked categorical attributes have **hierarchical relationship**, the minimum no of edges to traverse from one node to the other could be used as distance (e.g., vendor type hierarchy)

Distance Aggregation

- ▶ We **aggregate across all attributes** to find the net distance between 2 entities
- ▶ **Different ways to aggregate:** Euclidean, Manhattan. Attributes can be weighted during aggregation, indicative of their relative importance

Pair Wise Distance Calculation MR

- ▶ It's an $O(n \times n)$ problem. If there are 1 million transactions, we need to perform 1 trillion computation.
- ▶ The work will be divided up between the reducers. If we have a 100 node Hadoop cluster with 10 reducers slots per node, each reducer will roughly perform 1 billion distance calculation.
- ▶ How do we divide up the work? Use partitioned hashing. If $h1 = \text{hash}(\text{id1})$ and $h2 = \text{hash}(\text{id2})$, we use function of $h1$ and $h2$ as the key of the mapper output. For example $f(h1, h2) = h1 \ll 10 \mid h2$.
- ▶ All the transactions with id hashed to $h1$ or $h2$ will end up with the same reducer.

Partitioned Hashing

► Code snippet from SameTypeSimilarity.java

```
String partition = partitonOrdinal >= 0 ? items[partitonOrdinal] :  
"none";
```

```
    hash = (items[idOrdinal].hashCode() % bucketCount +  
bucketCount) / 2 ;
```

```
    for (inti = 0; i < bucketCount; ++i) {  
if (i < hash) {  
    hashPair = hash * 1000 + i;  
    keyHolder.set(partition, hashPair, 0);  
    valueHolder.set("0" + value.toString());  
} else {  
  
    hashPair = i * 1000 + hash;  
    keyHolder.set(partition, hashPair, 1);  
    valueHolder.set("1" + value.toString());  
}
```

```
context.write(keyHolder, valueHolder);
```

Output of Distance MR

- The output has 3 fields: the first transaction ID, second transaction ID and the distance

6JHQ79UA	JSXNUV9R	5
6JHQ79UA	Y1AWCM5P	89
6JHQ79UA	UFS5ZM0K	172

Nearest Neighbor MR

- ▶ Next we need to find the **nearest k neighbors of each data point**. We essentially need the neighbors of a data point sorted by distance.
- ▶ Use a technique called **secondary sorting**. We tag some extra data to the key which will force the key to be sorted by the data tagged as the mapper emits it's key and value.
- ▶ Going back to the output of the previous MR, this is how the mapper of this MR will **emit key, value**

key -> (6JHQ79UA, 5) value -> (JSXNUV9R, 5)

key -> (JSXNUV9R,, 5) value -> (6JHQ79UA, 5)

key -> (6JHQ79UA, 89) value -> (Y1AWCM5P, 89)

key -> (Y1AWCM5P, 89) value -> (6JHQ79UA, 172)

Nearest Neighbor MR (contd)

- ▶ On the reducer side when the reducer gets invoked, we will get a **transaction ID as a key** and a **list of neighboring transaction ID and distance pair as the value**
- ▶ In the reducer, we iterate through the values and **take the average distance** and emit the transaction ID and average distance as output. We could use median also.

1KVOMZE, 5

1JIOA0UE, 173

1KWBJ4W3, 278

.....

XXXX7362, 538

- ▶ As expected we find the outlier we injected into the dataset having a very large average distance to it's neighbor.

Fraud or Emerging Normal Behavior

- ▶ We have been able to detect the outlier. But how do we know **whether it's a fraudulent transaction or emerging buying pattern.**
- ▶ Your credit card may have been compromised and someone is using it. Or you have fallen in love and decided to shower him or her with expensive high price ticket items.
- ▶ We can't really tell the difference, **except that once there is enough data points for this emerging behavior, we won't be getting these false positives from our analysis**

Clustering Based Outlier Detection

Clustering Based Outlier Detection

- ▶ Clustering analysis attempts to group similar data instances into clusters. Usually used as a standalone tool to gain insight into a dataset's distribution
- ▶ Clustering aims to detect groups with similar behaviour. Outlier analysis aims to detect instances not similar to any other instance in the data.
- ▶ Key assumption: normal data records belong to large and dense clusters, while anomalies belong do not belong to any of the clusters or form very small clusters

Clustering Based Outlier Detection

- ▶ Categorization according to labels
 - ▶ Semi-supervised
 - ▶ Cluster the normal data to create modes of normal behavior.
 - ▶ Any new test instance is assigned to one of the clusters
 - ▶ If a new instance does not belong to any of the clusters or it is not close to any cluster, it is an anomaly
 - ▶ This approach is applied for novelty detection task in different domains such as novel topic detection in news data
 - ▶ Unsupervised
 - ▶ Post-processing is needed after a clustering step to determine the size of the clusters and the distance from the clusters is required for the point to be anomaly
- ▶ Anomalies detected using clustering based methods can be:
 - ▶ Data records that do not fit into any cluster (residuals from clustering)
 - ▶ Small clusters
 - ▶ Low density clusters or local anomalies (far from other points within the same cluster)

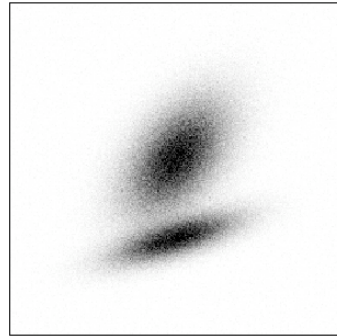
Detecting Outliers Using Clustering

- ▶ Most of the earlier clustering-based outlier detection techniques found outliers as the byproduct of a clustering algorithm
- ▶ Thus any data point which does not fit in any cluster is called an outlier.
- ▶ Since the main aim is to find clusters, such approaches are not optimized to find outliers.
 - ▶ Example: FindOut Algorithm (a by-product of WaveCluster Clustering Algorithm) used to find outliers in data. Discussed on the next slide.
- ▶ Several clustering based techniques focus on detecting outliers. Thus the output of this techniques is actually the outliers and not the cluster
 - ▶ Example: The CLAD algorithm and the FindCBLOF algorithm

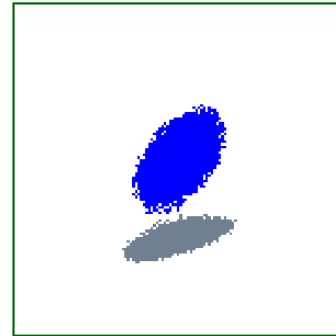
FindOut Algorithm

- ▶ The FindOut algorithm [Yu et al. 2002] is an extension of the WaveCluster algorithm which works as follows:
- ▶ Main idea: Remove the clusters from original data and then identify the outliers
- ▶ Transform data into multidimensional signals using wavelet transformation
 - ▶ High frequency of the signals correspond to regions where there is a rapid change of distribution - boundaries of the clusters
 - ▶ Low frequency parts correspond to the regions where the data is concentrated
- ▶ Remove these high and low frequency parts and all remaining points will be outliers

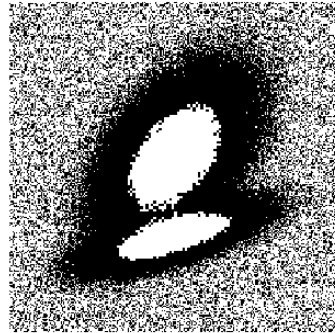
FindOut Algorithm



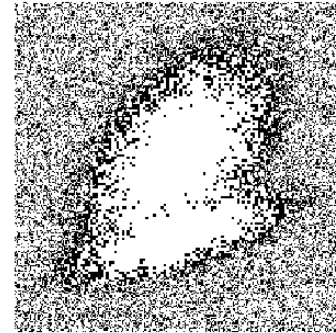
a)



b)



c)



d)

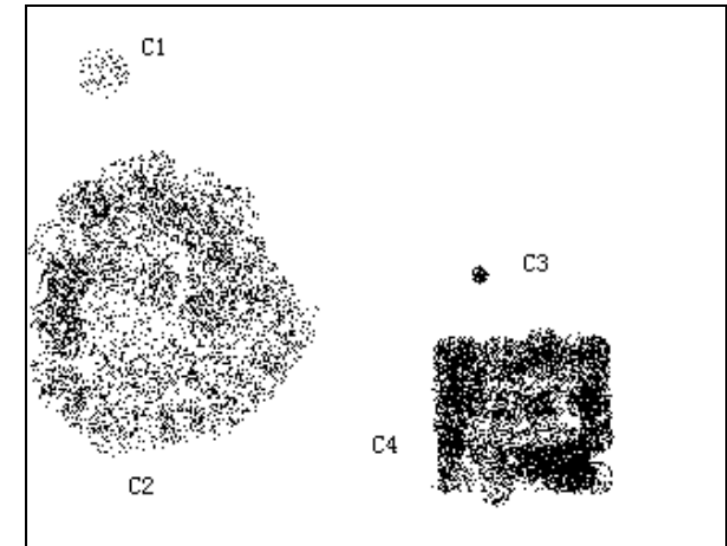
Figure 4: a) Original data; b) Clusters detected by WaveCluster; c) Original data \ominus clusters; d) Outliers detected by FindOut.

FindCBLOF Algorithm [He et al. 2003]

- ▶ Find clusters in a data set, and sort them according to decreasing size. The algorithm assumes that most of the data points are not outliers.
- ▶ It uses a parameter α ($0 < \alpha < 1$) to distinguish large from small clusters. If we arrange the clusters in decreasing order of their size, then the first clusters (say, the first k -clusters) in that list that cover at least a percentage α of the data set are called large clusters. The remaining clusters are referred to as small clusters.
- ▶ To each data point, assign a cluster-based local outlier factor (CBLOF).
 - ▶ For a point belonging to a large cluster, its CBLOF is the product of the clusters size and the similarity between the point and the cluster.
 - ▶ For a point belonging to a small cluster, its CBLOF is calculated as the product of the size of the small cluster and the similarity between the point and the closest large cluster.

FindCBLOF Algorithm (...contd)

- ▶ CBLOF defines the similarity between a point and a cluster in a statistical way representing the probability that the point belongs to the cluster.
 - ▶ The larger the value, the more similar the point and the cluster are.
 - ▶ The CBLOF score can detect outlier points that are far from any clusters.
 - ▶ In addition, small clusters that are far from any large cluster are considered to consist of outliers.
 - ▶ The points with the lowest CBLOF scores are suspected outliers.
- ▶ Consider the figure shown. Here, there are 4 clusters C_1 , C_2 , C_3 and C_4 . Obviously, the data points in C_1 and C_3 should be regarded as outliers. The CBLOF algorithm aims to capture this.



A sample 2D dataset where the outliers form clusters themselves

Clustering Based Outlier Detection

► Advantages:

- No need to be supervised
- Easily adaptable to on-line / incremental mode suitable for anomaly detection from temporal data
- Once the cluster are obtained, need only compare any object against the clusters to determine whether it is an outlier (fast)

► Drawbacks

- The clustering process itself is computationally expensive (usually requires pairwise distance calculation)
- Using indexing structures (k-d tree, R^* tree) may alleviate this problem
- If normal points do not create any clusters the techniques may fail
- In high dimensional spaces, data is sparse and distances between any two data records may become quite similar. Clustering algorithms may not give any meaningful clusters

Clustering Based Outlier Detection – Big Data

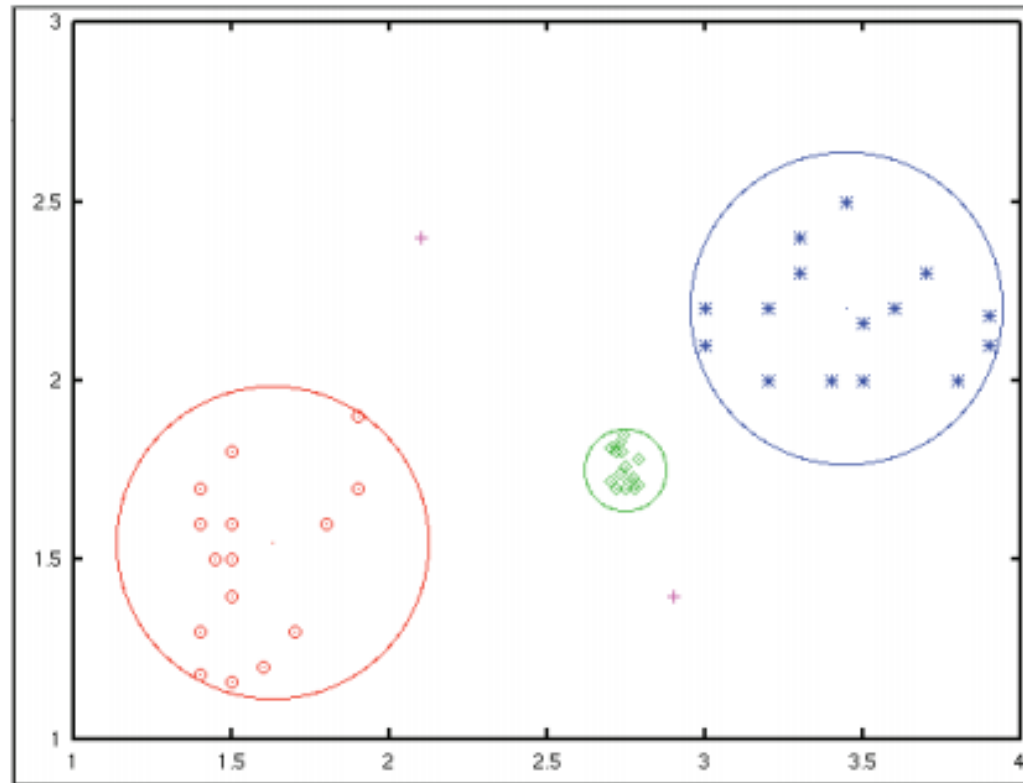
- ▶ Even though the clustering process is computationally expensive, there have been numerous implementations of it for outlier detection applications in Big Dataset. We discuss two such implementations:
 - ▶ *An outlier detect algorithm using Big Data Processing and IoT architecture [Souza and Amazonas 2015]:* Proposes an outlier detection procedure using the K-means algorithm and Big Data processing using the Hadoop platform and Mahout implementation integrated with their chosen Internet of Things architecture.
 - ▶ *Using DBSCAN to detect outliers in time-series data:* Used by Datadog (A SaaS based monitoring and analytics platform) and also by Netflix

Outliers, Big Data and Apache Mahout

1. The application inputs the raw data to create a clustering model.
2. Run the Canopy Clustering algorithm on the initial data to propose an initial value of the number K of centroids, using the Mahout's implementation proposed in 8.
3. Run the K-means algorithm, starting with the centroids proposed by the Canopy algorithm, to create a model of clusters, using the Mahout's implementation.
4. Get the information about the clusters and their centroids and radii generated by the K-means execution.
5. With these values the method `isOutLier` can be used. This implementation calculates the Euclidean 14 distance of the instance parameter to all centroids, and if is greater then each radius, this instance is classified as an outlier.

The figure on the illustrates the proposed approach

Outliers, Big Data and Apache Mahout



In the figure three clusters have been generated: cluster (a) represented by red circles, cluster (b) represented by green diamonds and cluster (c) represented by blue stars, and two outliers points represented by magenta plus, as they are outside the clusters' circles defined by the clusters' radii.

DBSCAN for Outlier Detection

- ▶ DBSCAN, (Density-Based Spatial Clustering of Applications with Noise), captures the insight that clusters are dense groups of points. The idea is that if a particular point belongs to a cluster, it should be near to lots of other points in that cluster.
- ▶ It works like this: First we choose two parameters, a positive number epsilon and a natural number minPoints. We then begin by picking an arbitrary point in our dataset. If there are more than minPoints points within a distance of epsilon from that point, (including the original point itself), we consider all of them to be part of a "cluster". We then expand that cluster by checking all of the new points and seeing if they too have more than minPoints points within a distance of epsilon, growing the cluster recursively if so.
- ▶ Eventually, we run out of points to add to the cluster. We then pick a new arbitrary point and repeat the process. Now, it's entirely possible that a point we pick has fewer than minPoints points in its epsilon ball, and is also not a part of any other cluster. If that is the case, it's considered a "noise point" not belonging to any cluster.
- ▶ To see an interactive demo: www.naftaliharris.com/blog/visualizing-dbscan-clustering/

How Datadog uses DBSCAN for outliers

- Outlier detection works by comparing each host against the others in the group. Using their outlier detection algorithms, they can alert when a host or group of hosts deviates from the pack, while avoiding alerts for expected, group-wide spikes.



Outlier Detection in the context of Big Data

Outlier Detection and Big Data

- ▶ The major limitation of a large number of algorithms described previously is their lack of scalability with respect to the number of points and dimensionality of the dataset.
 - ▶ The algorithms won't scale well to large datasets.
- ▶ Also, Many data sets in real applications may contain categorical attributes, which take on discrete unordered values.
 - ▶ Such attribute values are not ordered, and therefore require different data analysis techniques.

	categorical	continuous	categorical	continuous	binary
<i>Tid</i>	SrcIP	Duration	Dest IP	Number of bytes	Internal
1	206.163.37.81	0.10	160.94.179.208	150	No
2	206.163.37.99	0.27	160.94.179.235	208	No
3	160.94.123.45	1.23	160.94.179.221	195	Yes
4	206.163.37.37	112.03	160.94.179.253	199	No
5	206.163.37.41	0.32	160.94.179.244	181	No

Outlier Detection for Categorical Data

- The AVF Algorithm

- ▶ The AVF algorithm is a simple and fast approach to detect outliers in categorical data, which minimizes the scans over the data, without the need to create or search through different combinations of the attribute values or item sets.
- ▶ It is intuitive that outliers are those points which are infrequent in the dataset, and that the ideal outlier point in a categorical dataset is one whose each and every value is extremely irregular or infrequent.
- ▶ The infrequentness of an attribute value can be measured by computing the number of times this value is assumed by the corresponding attribute in the dataset.

The AVF Algorithm

- ▶ Let us assume that there are n points in the dataset, $(x_i \text{ where } i = 0, 1, 2, \dots, n)$ and each point has m attributes.
- ▶ We can write $x_i = [x_{i1}, x_{i2}, \dots, x_{im}]$ where x_{ij} is the j -th attribute value of x_i
- ▶ Following the intuition established in the previous slide, the AVF score is a good indicator of deciding whether x_i is an outlier:

- ▶
$$AVFScore(x_i) = \frac{1}{m} \sum_{j=0}^m f(x_{ij})$$

- ▶ Since we essentially have a sum of m positive numbers, the AVF score is minimized when each of the summation terms are individually minimized. Thus, the AVF score will be minimum for the 'ideal' outlier as defined earlier.
- ▶ Once we calculate the AVF score of each point, we designate the AVF score of k points with the smallest AVF scores as the k outliers

The AVF Algorithm

Input : Database \mathcal{D} (n points \times m attributes), Target number of outliers - k
Output: k detected outliers

```
1 Label all data points as non-outliers;
2 foreach point  $\mathbf{x}_i, i = 1 \dots n$  do
3   |   foreach attribute  $l, l = 1 \dots m$  do
4   |   |   Count frequency  $f(x_{il})$  of attribute value  $x_{il}$ ;
5   |   end
6 end
7 foreach point  $\mathbf{x}_i, i = 1 \dots n$  do
8   |   foreach attribute  $l, l = 1 \dots m$  do
9   |   |    $AVFScore(\mathbf{x}_i) += f(x_{il})$ ;
10  |   end
11  |    $AVFScore(\mathbf{x}_i) /= m$ ;
12 end
13 Return  $k$  outliers with  $\min_i(AVFScore)$ ;
```

Figure 7: AVF Pseudocode

Parallelizing the AVF Algorithm using MapReduce

- ▶ Compared to AVF, the other algorithms for categorical datasets are more cumbersome to parallelize
- ▶ The original AVF algorithm calculates AVF over each input record independently, making it amenable to easy parallelization.
- ▶ Using MapReduce allows us to parallelize the algorithm along with the benefits of automatic load balancing and fault tolerance, with no additional effort from the user's perspective.
- ▶ The MapReduce version of the AVF algorithm is called MR-AVF algorithm

The MR-AVF Algorithm

- ▶ Use 2 stages of Map and Reduce, which can be summarized pretty easily as follows:
- ▶ Stage I
 - ▶ The first map function associates each distinct attribute value to the map's output key.
 - ▶ The reduce function computes the frequency counts of each attribute value.
- ▶ Stage II
 - ▶ Finally, the AVF score of each data point is calculated during a second map function.
 - ▶ The second reduce is simply a sorting operation of the computed AVF scores.

The MR-AVF Algorithm

Input : Database \mathcal{D} (n points \times m attributes),
Target number of outliers - k

Output: k detected outliers

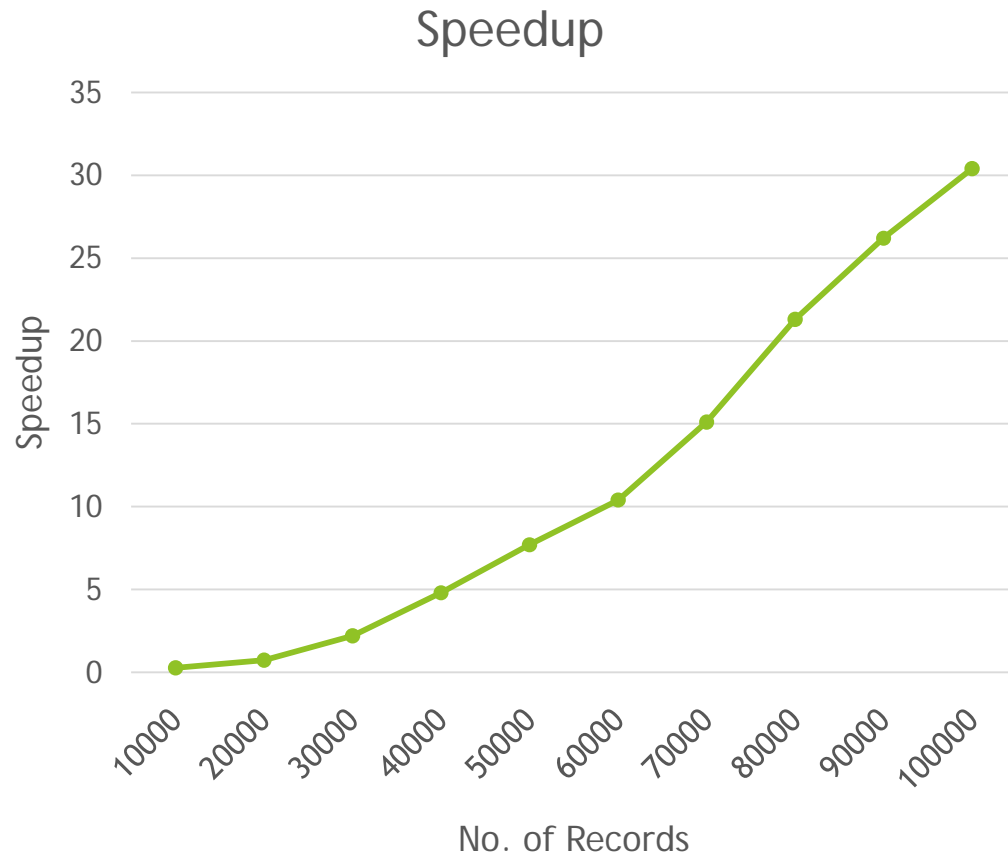
```
1 HashTable  $H$ ;  
2 map  $k_1 = i, v_1 = \mathcal{D}_i = \mathbf{x}_i, i = 1 \dots n$  begin  
3   | foreach  $l \in \mathbf{x}_i, l = 1 \dots m$  do  
4   |   |  $\text{collect}(x_{il}, 1)$ ;  
5   | end  
6 end  
  
7 reduce  $k_2 = x_{il}, v_2$  begin  
8   |  $H(x_{il}) += v_2$ ;  
9 end  
  
10 map  $k_1 = i, v_1 = \mathcal{D}_i = \mathbf{x}_i$  begin  
11   |  $AVF = \sum_{l=1}^m H(x_{il})$ ;  
12   |  $\text{collect}(k_1, AVF)$ ;  
13 end  
  
14 reduce  $k_2 = AVF_i, v_1 = i$ ;
```

Figure 11: MR-AVF Pseudocode

Analysis of the MR-AVF Algorithm

- ▶ In the first pair of Map/Reduce functions (first phase, lines 2-9),
 - ▶ The frequency of each attribute value is extracted from the data set.
 - ▶ If the attribute values across each attribute are unique or the dimension is concatenated to the attribute value (as in our code), this pair of functions is similar to the WordCount problem
- ▶ In the second MapReduce phase (lines 10-14),
 - ▶ The attribute value frequency table resulting from the first phase is loaded into the hash table H by the map function.
 - ▶ The map function then calculates and emits the AVF score for each individual input record, by iterating through the dimensions and adding the frequency of every attribute value.
 - ▶ In order to sort the data points by their outlier score, the AVF score is emitted as the key, and the input point ID is emitted as the value.
- ▶ At the end of the MapReduce process, the result is a list of AVF scores sorted in ascending order with the listed point IDs. As a result, the top-k points represent the outliers of the dataset as they have the k minimum AVF scores.

Speedup of the MR-AVF vs AVF



No of records	Time taken by AVF (s)	Time taken by MR-AVF (s)	Speedup
10000	14	51	0.27
20000	39	53	0.74
30000	118	53	2.2
40000	262	55	4.8
50000	424	55	7.7
60000	570	55	10.4
70000	846	56	15.1
80000	1190	56	21.3
90000	1467	56	26.2
100000	1735	57	30.4

Table 4.1: Variation of speedup with the number of records

Application specific Anomaly Detection Algorithms in Big Data

- ▶ Various other application specific anomaly detection algorithms exist for categorical data. One such case study is presented below for detecting DDoS (Distributed Denial of Service) attacks on servers.
- ▶ HTTP flood is a type of Distributed Denial of Service (DDoS) attack in which the attacker exploits seemingly-legitimate HTTP GET or POST requests to attack a web server or application.
- ▶ HTTP flood attacks are volumetric attacks, often using a botnet “zombie army”—a group of Internet-connected computers, each of which has been maliciously taken over, usually with the assistance of malware like Trojan Horses.
- ▶ A sophisticated Layer 7 attack, HTTP floods do not use malformed packets, spoofing or reflection techniques, and require less bandwidth than other attacks to bring down the targeted site or server.

HTTP Flooding – An Example (Taken from a cyber-security blog)

- ▶ The website of a specific client was down for almost a week. A cyber security firm hired to look into the matter came to know the extent of the attack
- ▶ To be more exact, he was getting 5,233 HTTP requests every single second. From different IP addresses around the world.
- ▶ What is important to note here is how this worked against the client's platform. The client's website was built on WordPress. The uniqueness of the requests were bypassing the caching system, forcing the system to render and respond to every request. This was bringing about system failures as the server quickly became overwhelmed by the requests.
- ▶ What is important to note here is how this worked against the client's platform. The client's website was built on WordPress. The uniqueness of the requests were bypassing the caching system, forcing the system to render and respond to every request. This was bringing about system failures as the server quickly became overwhelmed by the requests.

HTTP Flooding - An Example

- ▶ For illustration purposes, the figure given is a quick geographic distribution of the IP's hitting the site. This is for 1 second in the attack. Yes, every second these IP's were changing.
- ▶ **Stopping the DDoS:** Once we identified the type of attack, blocking was easy enough. By default, they were not passing our anomaly check, causing the requests to get blocked at the firewall. One of the many anomalies we look for are valid user agents, and if you look carefully you see that the requests didn't have one. Hopefully, you'll also noticed that the referrers were dynamic and the packets were the same size, another very interesting signature. Needless to say, this triggered one of our rules, and within minutes his site was back.



DDoS prevention using MapReduce

- ▶ HTTP GET incomplete (also called SlowLoris) tries to keep an http session active continuously for a long period of time. Its a very well known fact that, web server's like Apache works on a threaded or a process based model. Due to which the server will become unavailable for new requests, if all the threads, or processes of a web server are consumed
- ▶ Since the DDoS attack with the low volume of traffic such as the HTTP GET incomplete attack is prevalent in these days, the frequency of page requests from clients will be a more effective factor.
- ▶ Counter-based detection is a simple method that counts the total traffic volume or the number of web page requests

DDoS prevention using MapReduce

- ▶ In the MapReduce algorithm, the map function filters non-HTTP GET packets and generates key values of server IP address, masked timestamp, and client IP address. The masked timestamp with time interval is used for counting the number of requests from a specific client to the specific URL within the same time duration.
- ▶ The reduce function summarizes the number of URL requests, page requests, and server responses. Finally, the algorithm aggregates values per server.
- ▶ When total requests for a specific server exceeds the threshold, the MapReduce job emits records whose response ratio against requests is greater than unbalance ratio, marking them as attackers.
- ▶ Disadvantages:
 - ▶ it needs a prerequisite to know the threshold value from historical monitoring data in advance.
 - ▶ This algorithm needs three input parameters of time interval, threshold and unbalance ratio

DDoS prevention using MapReduce

- ▶ Time interval limits monitoring duration of the page request.
- ▶ Threshold indicates the permitted frequency of the page request to the server against the previous normal status, which determines whether the server should be alarmed or not.
- ▶ The unbalance ratio variable denotes the anomaly ratio of response per page request between a specific client and a server. This value is used for picking out attackers from the clients.

