Chapter 9:
Outlier Analysis

Jilles Vreeken

IRDM '15/16

8 Dec 2015

UNIVERSITÄT DES SAARLANDES

CLUSTER OF EXCELLENCE

max planck institut informatik

IRDM Chapter 9, overview

1. Basics & Motivation
2. Extreme Value Analysis
3. Probabilistic Methods
4. Cluster-based Methods
5. Distance-based Methods

You'll find this covered in:
Aggarwal, Ch. 8, 9

December 14th – 18th
Tutorials on Graph Mining

January 4th – 8th
No Tutorials

The Second Midterm Test

December 10th 2015

When: from 14:15 to 15:25

Where: Günter-Hotz-Hörsaal (E2 2)

Material: Patterns, Clusters, and Classification

You are allowed to bring one (1) sheet of A4 paper with handwritten or printed notes on both sides .

No other material (notes, books, course materials) or devices (calculator, notebook, cell phone, spoon, etc) allowed.

Bring an ID; either your UdS card, or passport.

Chapter 9.1:
The Basics & Motivation

Aggarwal Ch. 8.1

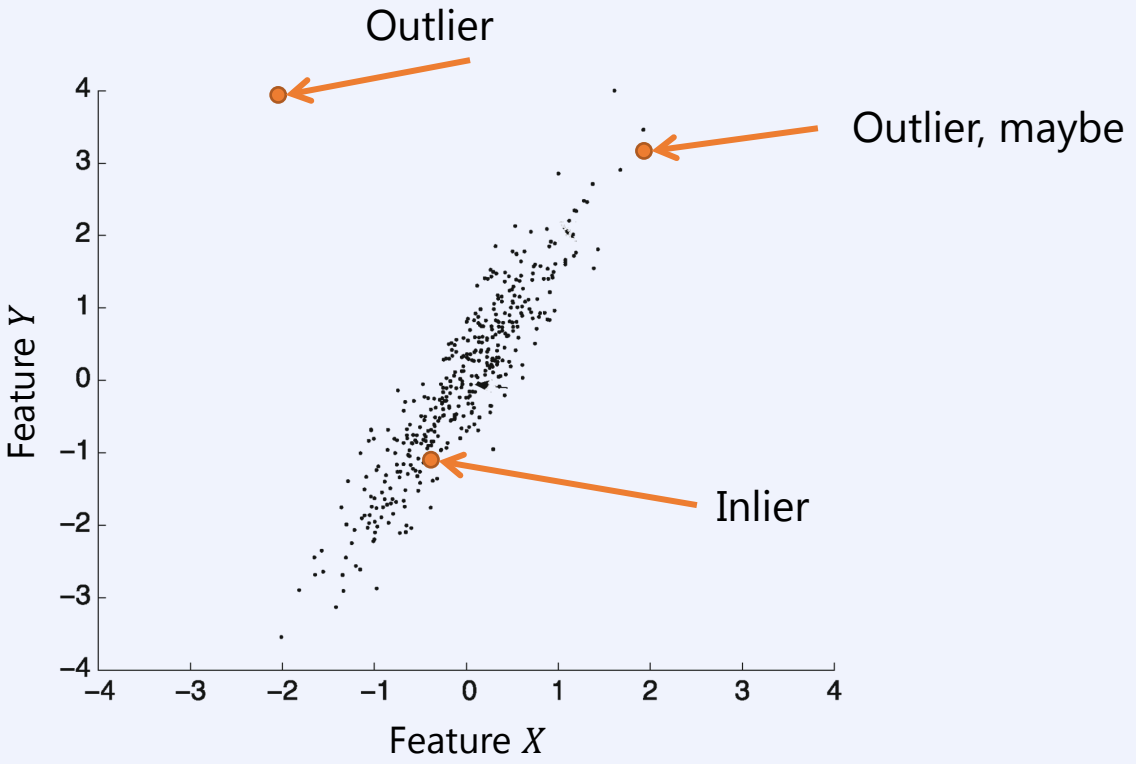
Outliers

An **outlier** is a data point very different from most of the remaining data.

- the standard definition is by Hawkins

> "An outlier is an observation which deviates so much from the other observations as to arouse suspicion it was generated by a different mechanism"

Example Outliers



Outliers

An **outlier** is a data point very different from most of the remaining data.

- the standard definition is by Hawkins

> "An outlier is an observation which deviates so much from the other observations as to arouse suspicion it was generated by a different mechanism"

Outliers are also known as

- **anomalies**, abnormalities, discordants, deviants

Why bother?

Outlier analysis is a key area of data mining

Unlike pattern mining, clustering, and classification, it aims to describe what is **not** normal

Applications are many

- data cleaning
- fraud detection
- intrusion detection
- rare disease detection
- predictive maintenance

Not noise

Outliers are not noise

- noise is uninteresting, outliers are
- noise is random, outliers aren't

Outliers are generated by a **different process**

- e.g. Lionel Messi, or credit card fraudsters, or rare disease patients
- we have too little data to infer that process exactly
- detected outliers help us to better understand the data

Outliers everywhere

Many many different outlier detection methods exist

- many different methods needed
  - e.g. continuous vs. discrete data
  - e.g. tables, sequences, graphs

The **key problem**, and why outlier analysis is interesting: beforehand, we do not know what we are looking for

- what is weird?
- what is normal?

Three Types of Outliers

Global outliers

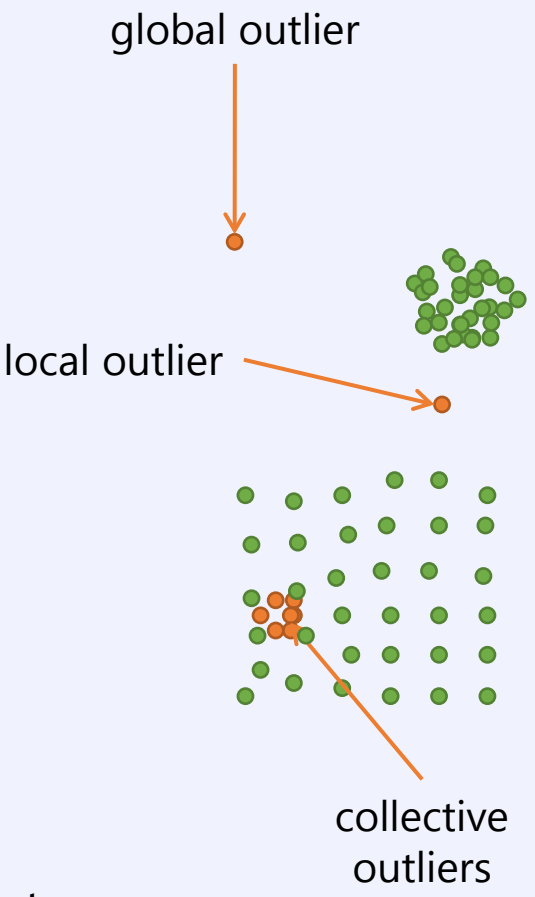
- object that deviate from **the rest of the data set**
- main issue: find a good measure of deviation

Local outliers

- object that deviates from **a selected context**
  e.g. differs strongly from its neighboring objects
- main issue: how to define the local context?

Collective outliers

- a subset of objects that **collectively** deviate
  from the data or context, e.g. intrusion detection
- main issue: combinatorial number of sets of objects



Ranking versus Thresholding

Most outlier analysis methods give a real-valued score

How to decide whether a point is worth looking at?

- we set a threshold, or look at the top-k
- no best answer, depends on situation

How to evaluate?

- very, very difficult
- is there a 'true' outlier ranking?
- how bad is it to miss one, or to report two too many?

Supervised Outlier Detection

Given sufficient data, we can construct a classifier

- and then simply use it to predict how outlying an object is
- typically does not fly in practice

Problem 1: Insufficient training data

- outliers are rare
- we can boost (resample) a training set from a small set of known outliers
- we can train on artificial samples

Problem 2: Recall

- recall is more important than accuracy
  - we want to catch them all

Chapter 9.2:
Extreme Value Analysis

Aggarwal Ch. 8.2

Extreme Values

The traditional statistical approach to identifying outliers is **extreme value analysis**

Those points $x \in \boldsymbol{D}$ that are in the **statistical tails** of the probability distribution p of $\boldsymbol{D}$ are outliers.

- only identifies very specific outliers

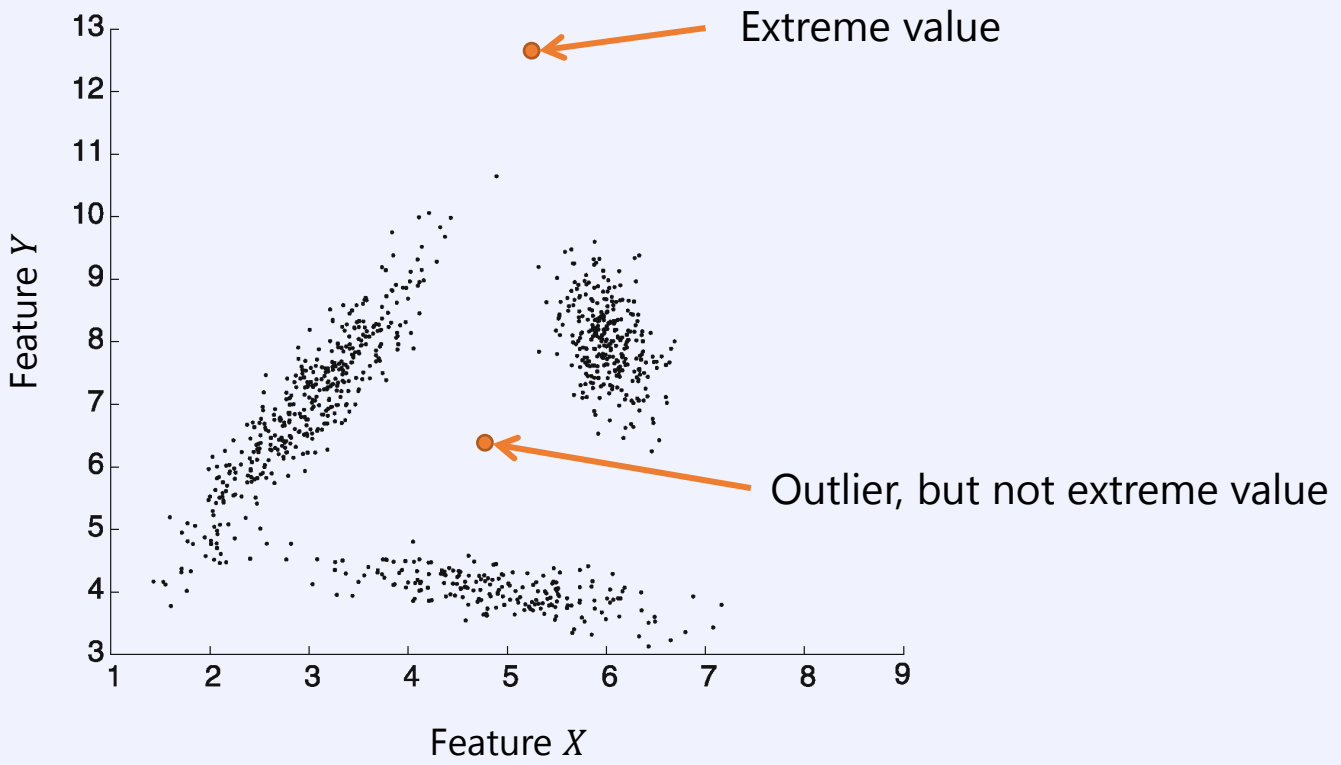
For example, for $\{1,3,3,3,50,97,97,97,100\}$

- extreme values are 1 and 100, although 50 is the most isolated

Tails are naturally defined for univariate distributions

- defining the multivariate tail area of a distribution is more tricky

Problems with multivariate tails



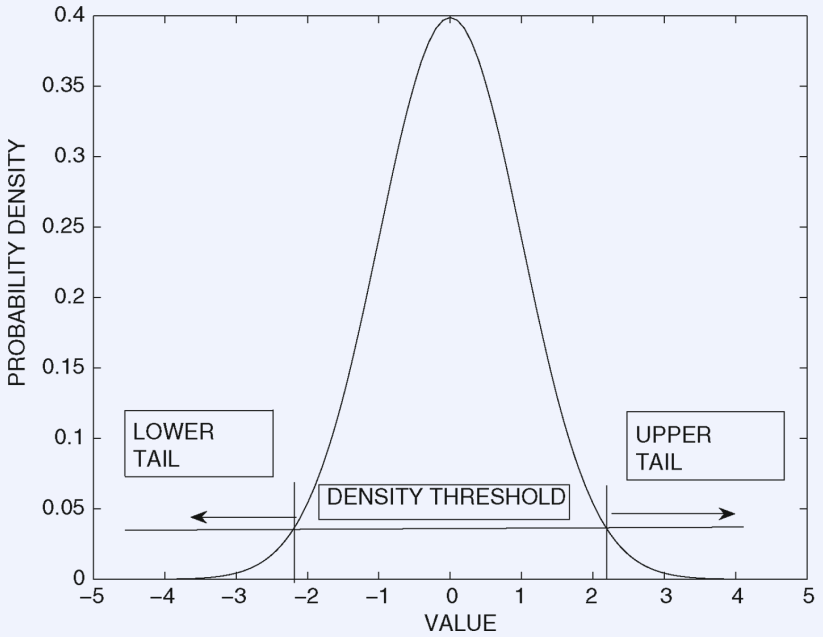
Univariate Extreme Value Analysis

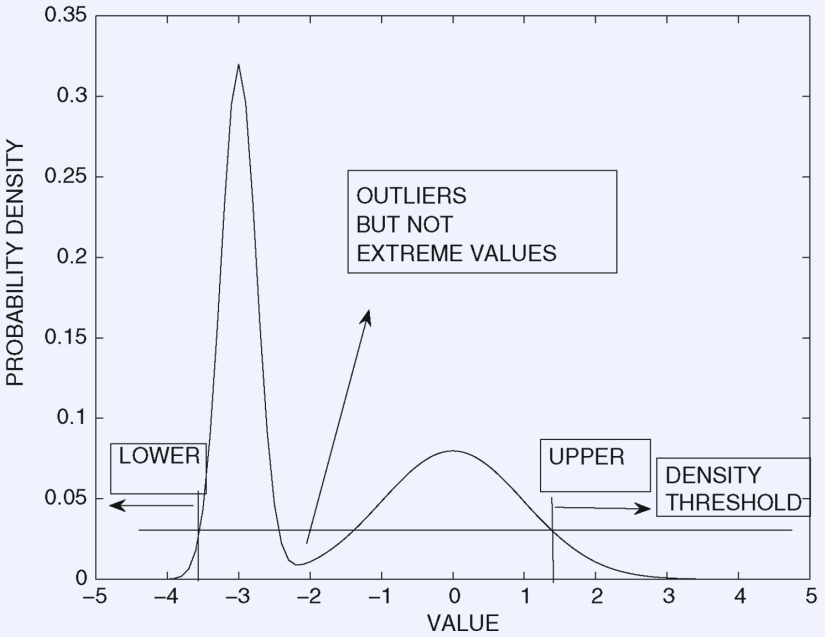
Strong relation to statistical tail confidence tests

Assume a distribution, and consider the probability density function $f_X(x)$ for attribute X

- the **lower tail** are then those values $x < l$ for which for all $f_X(x) < \epsilon$
- the **upper tail** are then those values $x > u$ for which for all $f_X(x) < \epsilon$

Not a density threshold.





Not all distributions have two tails

- exponential distributions, for example

Univariate

For example, for a Gaussian

$$f_X(x) = \frac{1}{\sigma \cdot \sqrt{2 \cdot \pi}} \cdot e^{-\frac{(x-\mu)^2}{2 \cdot \sigma^2}}$$

- with sufficient data we can estimate σ and μ with high accuracy

We can then compute z-scores, $z_i = (x_i - \mu)/\sigma$

- large positive values correspond to upper tail, large negative to lower tail

We can write the pdf in terms of z-scores as

$$f_X(z_i) = \frac{1}{\sigma \cdot \sqrt{2 \cdot \pi}} \cdot e^{-\frac{x_i^2}{2}}$$

- the **cumulative normal distribution** then tells the area of the tail larger than z_i
- as rule of thumb, z-scores with absolute values larger than 3 are extreme

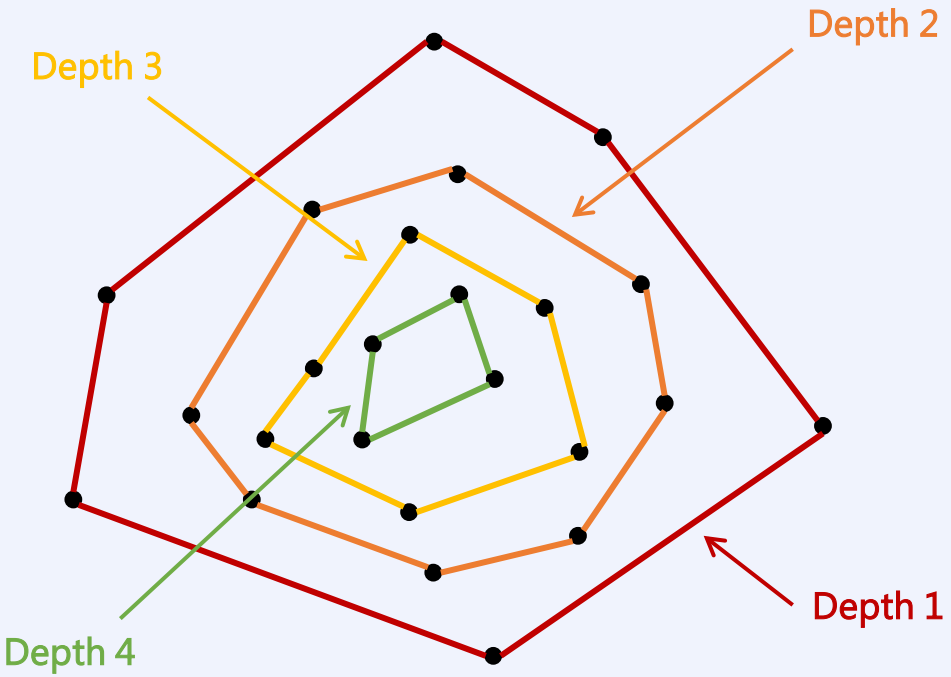
Depth-based methods

The main idea is that the **convex-hull** of a set of data points represents the **pareto-optimal extremes** of the set

- find the convex hull, and assign k to all $x \in hull(\boldsymbol{D})$
- remove $hull(\boldsymbol{D})$ from $\boldsymbol{D}$, increase k and repeat until $\boldsymbol{D}$ is empty

The depth k identifies how extreme a point is

Example, depth



Depth-based methods

The main idea is that the **convex-hull** of a set of data points represents the **pareto-optimal extremes** of the set

- find set S of corners of convex hull of $\boldsymbol{D}$
- assign depth k to all $x \in S$, and repeat until $\boldsymbol{D}$ is empty

The depth of a point identifies how extreme it is

Very sensitive to dimensionality

- recall, how are typically distributed over the hull of a hypersphere
- computational complexity

Multivariate Extreme Value Analysis

We can also define tails for **multivariate distributions**

- areas of extreme values with probability density less than some threshold

More complicated than univariate

- and, only works for **unimodal distributions with single peak**

Multivariate Extreme Value Analysis

For a **multivariate Gaussian**, we have its density as

$$f(x) = \frac{1}{\sqrt{|\Sigma|} \cdot (2 \cdot \pi)^{\frac{d}{2}}} \cdot e^{-\frac{1}{2} \cdot (x-\mu)\Sigma^{-1}(x-\mu)^T}$$

- where Σ is the d-by-d covariance matrix, and $|\Sigma|$ is its determinant

The exponent resembles **Mahalanobis distance**…

Mahalonobis distance

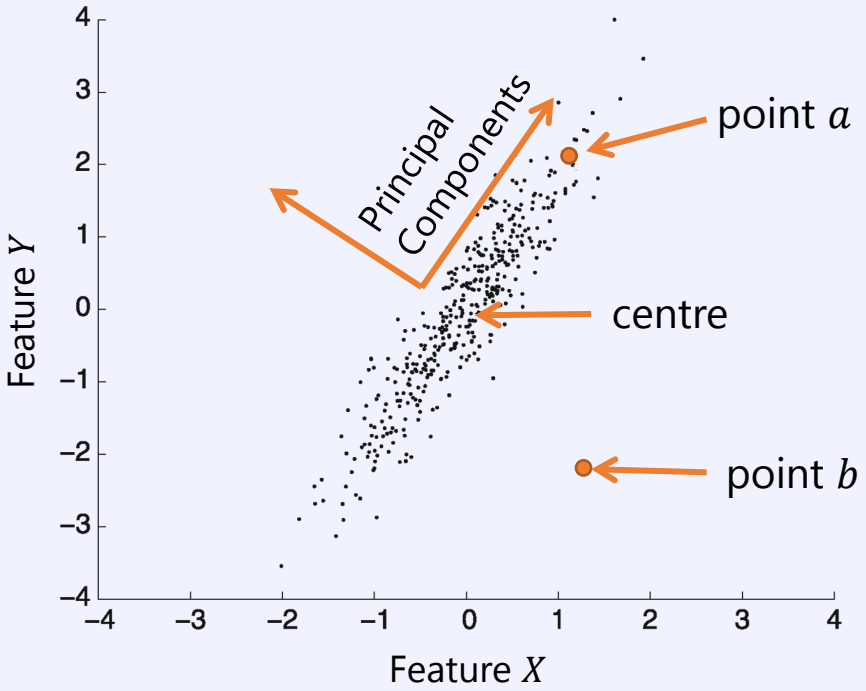
Mahalanobis distance is defined as

$$M(x, \mu, \Sigma) = \sqrt{(x-\mu)\Sigma^{-1}(\mathrm{x}-\mu)^T}$$

- Σ is a d-by-d covariance matrix, and μ a mean-vector

Essentially Euclidean distance, after applying PCA, and after dividing by standard deviation

- very useful in practice
- e.g. for example on the left, $M(b, \mu, \Sigma) > M(a, \mu, \Sigma)$



(Mahalanobis, 1936)

Multivariate Extreme Value Analysis

For a **multivariate Gaussian**, we have its density as

$$f(x) = \frac{1}{\sqrt{|\Sigma|} \cdot (2 \cdot \pi)^{\frac{d}{2}}} \cdot e^{-\frac{1}{2} \cdot (x-\mu)\Sigma^{-1}(x-\mu)^T}$$

- where Σ is the d-by-d covariance matrix, and $|\Sigma|$ is its determinant

The exponent is half squared Mahalanobis distance

$$f(x) = \frac{1}{\sqrt{|\Sigma|} \cdot (2 \cdot \pi)^{\frac{d}{2}}} \cdot e^{-\frac{1}{2} \cdot M(x,\mu,\Sigma)^2}$$

- for the probability density to fall below a threshold, the Mahalonobis distance needs to be larger than a threshold.

Probably extreme

Mahalanobis distance to the mean is an extremity score

- larger values imply more extreme behavior

The probability of being extreme may be more insightful

- how to model?

Mahalanobis considers axes-rotated and scaled data

- each component along the principal components can be modeled as an independent standard Gaussian, which means we can model by $\mathcal{X}^2$
- points for which the Mahalanobis distance is larger than the cumulative probability are potential outliers

Extreme downsides

Extreme value analysis is a rather basic technique.

- only works when data has only a single-peaked distribution
- requires **assuming a distribution** (e.g. Gaussian)

Depth-based methods are very brittle in practice

- do not scale well with dimensionality

Chapter 9.3:
Probabilistic Methods

Aggarwal Ch. 8.3

Mixtures

Mahalanobis distance works well if there is a single peak

- what if there are multiple?

We can generalise to **multiple distributions** using **mixture modelling**

- to this end, we'll re-employ EM clustering.

Fit and Unfit

We assume the data was generated by a **mixture** of k distributions $\mathcal{G}_1 \ldots \mathcal{G}_k$ and the generation process was

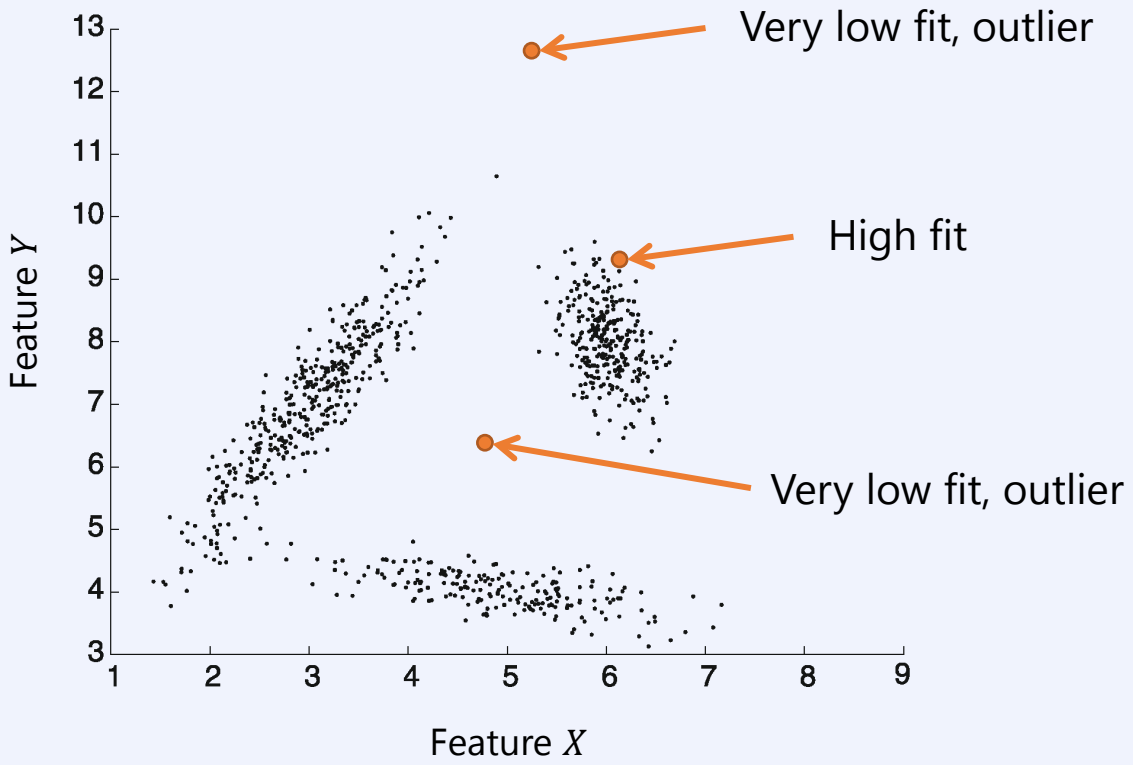
1. select a mixture component j with prior probability a_i where $i \in \{1 \ldots k\}$.
2. generate a data point from $\mathcal{G}_j$

The probability of point x generated by model $\mathcal{M}$ is

$$f(x \mid \mathcal{M}) = \sum_{i}^{k} a_i \cdot f^i(x)$$

- outliers will naturally have low fit probabilities

Unfit Outliers



Fit and Unfit

The probability of point x generated by model $\mathcal{M}$ is

$$f(x \mid \mathcal{M}) = \sum_{i}^{k} a_i \cdot f^i(x)$$

- outliers will naturally have low fit probabilities

To find the parameters for $\mathcal{M}$, we need to optimise

$$f^{data}(\boldsymbol{D} \mid \mathcal{M}) = \sum_{x \in \boldsymbol{D}} \log f(x \mid \mathcal{M})$$

such that the log likelihood of the data is maximized.

- this we do using EM (see lecture V-1)

Ups and Downs

Mixture modelling works very well

- when we know the family of distributions of the components
- when we have sufficient data to estimate their parameters
- and allows to include background knowledge, e.g. correlations

In practice, however…

- we do not know the number of components
- we do not know the distributions
- we do not have have sufficient data

Due to overfitting we are likely to miss outliers

Chapter 9.4:
Cluster-based Methods

Aggarwal Ch. 8.4

Clusters and Outliers

In both the probabilistic and cluster based approaches we define outliers as **points that deviate from the norm**

In the probabilistic approach the norm is a **distribution**.

- points with a low fit are outliers

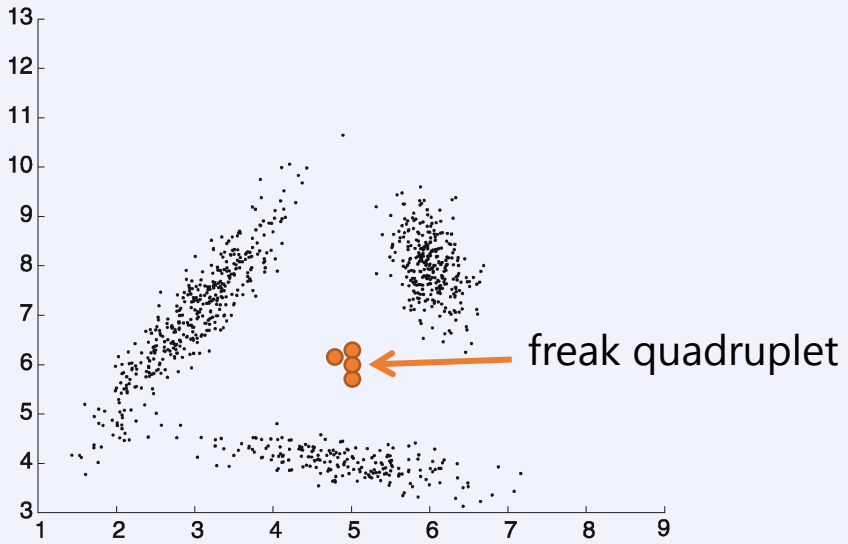
In the cluster-based approach the norm is a **clustering**.

- points far away from these clusters are outliers

A simplistic approach is to say that every point that does not belong to a cluster is an outlier.

- many clustering algorithms claim to find outliers as a side-product
- data points on the boundaries of clusters, however, are not real outliers

Clusters or Freaks



Cluster of outliers

Simple approach

The simple cluster-based approach to outlier detection

1. cluster your data
2. distance to closest centroid is outlier score for point x

Raw distances can be deceiving

- what if clusters are of different density?
- what if clusters are of different shape?

We need a score that takes the **context** into account

Local Mahalanobis

Mahalanobis distance does consider shape and density

- it is a **global** score, for single peaked unimodal distributions

We can, however, define a **local** Mahalanobis score

- compute mean vector μ_r and covariance matrix Σ_r **per cluster** $C_r \in \mathcal{C}$
- $(i, j) \in \Sigma_r$ is the covariance of dimensions i and j **in cluster** C_r

$$M(x, \mu_r, \Sigma_r) = \sqrt{(x - \mu_r)\Sigma_r^{-1}(x - \mu_r)^T}$$

- we can directly use it as an outlier score, higher is weirder

Cluster$f$ ...

Cluster-based anomaly detection makes intuitive sense

- it works decent in practice, even when not tailored for outliers
- can detect small clusters of outliers

Noise is a big problem

- clustering techniques do not distinguish between ambient noise and isolated points
- neither appear in a cluster, so both are outliers
- neither global nor local distances to centroids help

Chapter 9.5: Distance-based Methods

Aggarwal Ch. 8.5

Distance-based outliers

We can identify outliers **instance-based**, as opposed to model based, by using a distance measure.

> "The distance-based outlier score of an object x is its distance to its kth nearest neighbor."

In practice

- you choose a (meaningful) distance measure d
- you choose a (low) number of neighbors k
  - object x is **not part** of its own k-nearest neighbors
  - this avoids scores of 0 when $k = 1$

Computing V^k

Distance-based methods

- finer granularity than clustering or model-based methods

Let $V^k(x)$ be the distance of x to its k^{th} nearest neighbor

- $V^k(x) = d\big(x, y \mid y \text{ is the } k^{th}\ nearest\ neighbor\ of\ x\big)$

Naively computing $V^k(x)$ takes $O(n)$

- for all data points $x \in \boldsymbol{D}$ cost is $O(n^2)$, which is infeasible for large $\boldsymbol{D}$

We can speed up by indexing

- but, for high-dimensional data effectiveness degrades

Bounding through Sampling

First, we choose a sample S of $r < s \ll n$ objects from $\boldsymbol{D}$

- we compute all pairwise distances between S and $\boldsymbol{D}$
  - this costs $O(n \cdot s) \ll O(n^2)$

We have the exact score V^k for all objects S

We have a **lower bound** L^r of the r^{th} score of $\boldsymbol{D}$

- the r^{th} score of S, in pseudo-math: $L^r = \text{sort}(\{\, d(x,y) \mid x,y \in S \,\})_r$
- any object with a lower score will not be in top-r for $\boldsymbol{D}$

We have an **upper bound** U^k for scores of objects in $\boldsymbol{D} - S$

- the k-nearest neighbor distance of object $x \in \boldsymbol{D} - S$ to objects in S
- or, in pseudo-math: $U^k(x) = \text{sort}(\{\, d(x,y) \mid y \in S \,\})_k$

Top-r distance-based outliers

Usually, we only want the top-r most outlying objects

- these we can compute much more efficiently, using two tricks

Trick 1: compute lower and upper bounds by sampling

- compute full score for a set of $s > r$ objects
- gives lower bound on score for r^{th} object
- gives upper bound for score for all objects not in sample

Trick 2: early termination

- compute full score only for candidates that can beat the r^{th} score

Applying the bounds

No $x \in \boldsymbol{D}$ with upper bound $U^k(x) < L^r$ will be in top-r

- we do not have to compute its distances to $\boldsymbol{D} - S$
- only have to compute for $R = \{ x \in \boldsymbol{D} - S \mid U^k(x) > L^r \} \subseteq \boldsymbol{D} - S$

Top-r ranked outliers in $R \cup S$ are returned as final output

In practice, as $|R \cup S| \ll |D|$, this saves a lot of computation

- especially if $\boldsymbol{D}$ is clustered
- especially if we chose S wisely/luckily
  - at least one point per cluster, and r points in sparse regions

How would you choose S?

Early Termination

We can do better, however.

While computing the scores for $x \in R$, every time we discover an object with $V^k(x) > L^r$ we should update L^r

- meaning, pruning further candidates from R

For every $x \in R$ we start with upper bound $U^k(x)$

- initially based on the distances to S, but while computing the distances to $\boldsymbol{D} - S$, we should update it
- once $U^k(x)$ drops below L^r we should terminate

Algorithm

Algorithm TOPr-kNN-OUTLIERS(data $\boldsymbol{D}$, distance d, sample size s)
 $S \leftarrow sample(\boldsymbol{D}, s)$
 compute distances between S and $\boldsymbol{D}$
 $R \leftarrow \{ x \in \boldsymbol{D} - S \mid U^k(x) > L^r \}$
 for each $x \in R$ **do**
  for each $y \in \boldsymbol{D} - S$ **do**
   update current k-nearest neighbor distance estimate $V^k(x)$ by computing distance of y to x
   if $V^k(x) \leq L$ **then** break
  if $V^k(x) > L$ **then**
   include x in current r best outliers
   update L to the new r^{th} best outlier score
 return top-r outliers from $S \cup R$

Locally Outlying Factors

Raw distance measures don't always identify outliers well

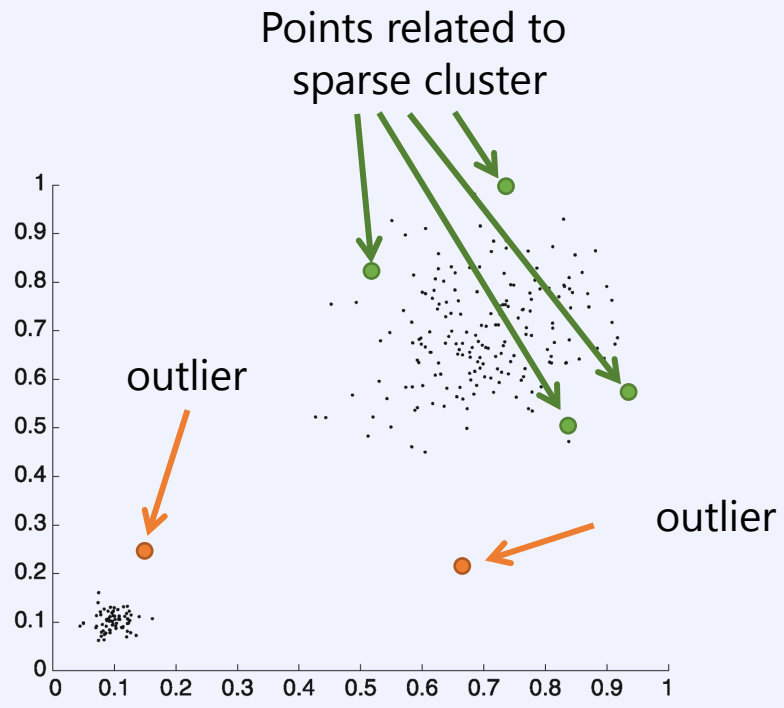
- they do not measure the **intrinsic distances**
- e.g. Euclidean distance, but Mahalanobis neither

The Locally Outlying Factors algorithm (LOF) is one of the earliest proposals to alleviate this

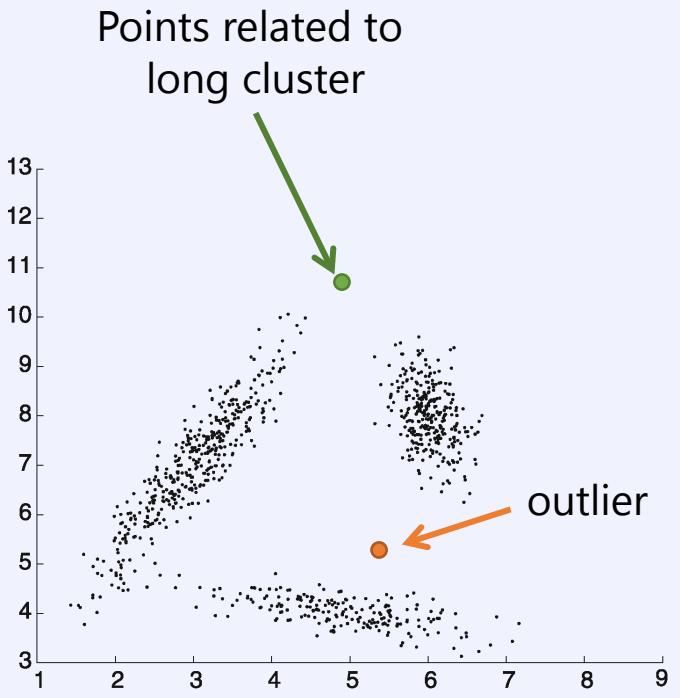
- also one of the most used* local outlier detection techniques

* or at least, most often compared against and beaten by more modern methods

Impact of Locality



Different density



Different shapes

LOF Begins

In LOF we consider our data locally. That is, for a point x we primarily work with its k-nearest neighborhood.

Let $L_k(x)$ be the set of objects that are at most as far as the k^{th} nearest neighbor of x

$$L_k(x) = \{\, y \in \boldsymbol{D} \mid d(x,y) \leq V^k(X) \,\}$$

Usually $L_k(x)$ will contain k points, sometimes more.

LOF, Origins

When a point y is in a dense area of the data, $V^k(y)$ will be low
(i.e. when there are many points close to it.)

When two points x and y are in each others k-nearest neighbors,

$$d(x, y) \leq \min\{V^k(x), V^k(y)\}$$

We can measure **how outlying** an object x is with regard to object y by considering the **reachability distance** between x and y.

$$R_k(x, y) = \max\{d(x, y), V^k(y)\}$$

when x **is not** in the k-nearest neighborhood of y, it is $d(x, y)$
when x **is** in the in the k-nearest neighborhood of y, it is $V^k(y)$
and k is essentially a data-driven smoothing parameter

The Rise of LOF

We compute the **average reachability distances** between object x and the objects in its k-nearest neighborhood

$$AR_k(x) = \text{mean}_{y \in L_k(x)} R_k(x, y)$$

which will be maximal when the nearest neighbors of x are at the edge of a dense cluster

The LOF Awakens

Now, finally, given a database $\boldsymbol{D}$, distance measure d, and a number of neighbors k, we define the **local outlying factor** of a point x as

$$LOF_k(x) = mean_{y \in L_k(x)} \frac{AR_k(x)}{AR_k(y)}$$

For objects inside a cluster, it will take a value close to 1, regardless of how dense the cluster is.

For outliers, $LOF_k(x) \gg 1$, as because x is not in the nearest neighborhoods of its *own* nearest neighbors the denominator will be much smaller than the numerator

The LOF strikes back

LOF works well in practice

- even with raw (Euclidean) distance measures
- regardless of number and density of clusters

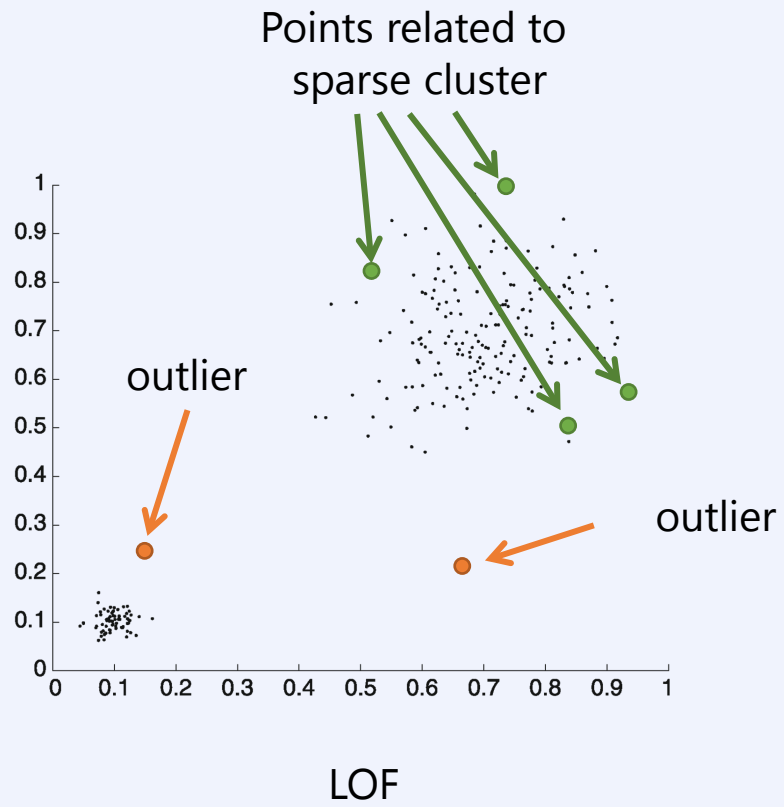
Why?

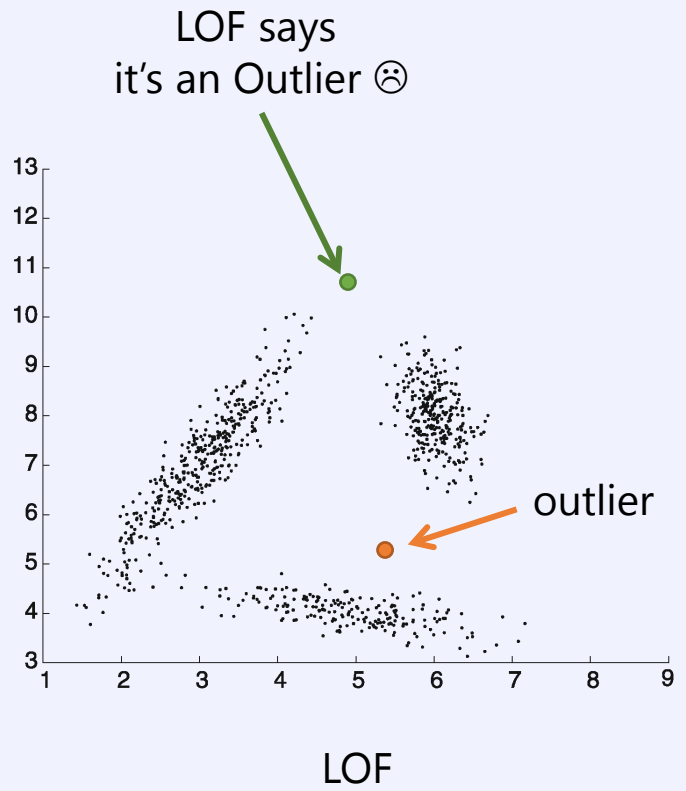
- because of the relative normalisation in the denominator
- it considers **local information** and can adapt to local density

LOF is not perfect

- $O(n^2)$ for high dimensional data, $O(n \log n)$ when we can index
- many variants exist for different cluster shapes

Impact of Locality





The return of the LOF

Euclidean distance-based has a bias to spherical clusters

- single-link clustering does not have this disadvantage

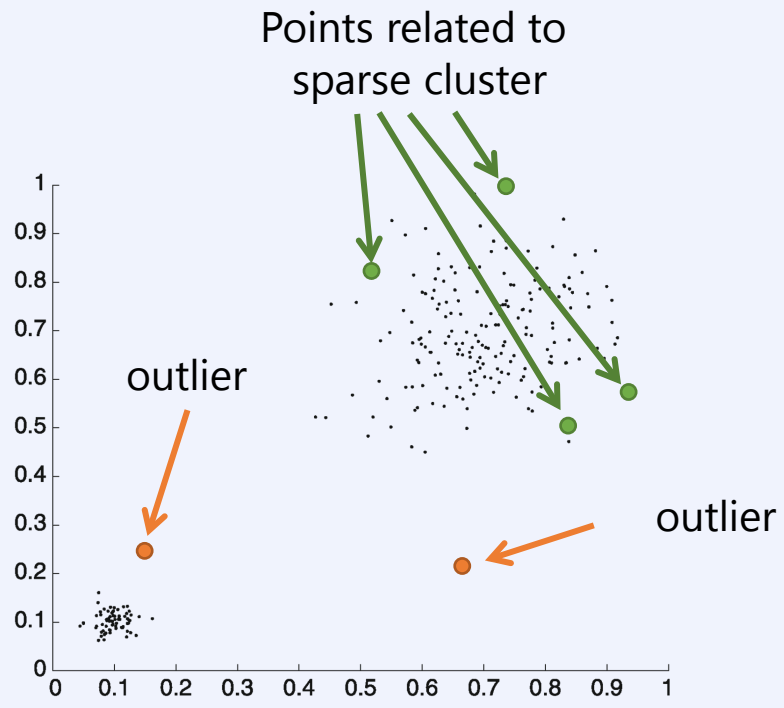
We can fix this by defining $L_k(x)$ using Single-Link

- start with $L_1(x) = \{x\}$
- and then iteratively add that y that is closest to any point in L_k
- $L_{k+1}(x) = L_k(x) \cup \{ \min_{y \in \boldsymbol{D}, y \notin L_k(x)} \{ d(y,z) \mid z \in L_k \} \}$

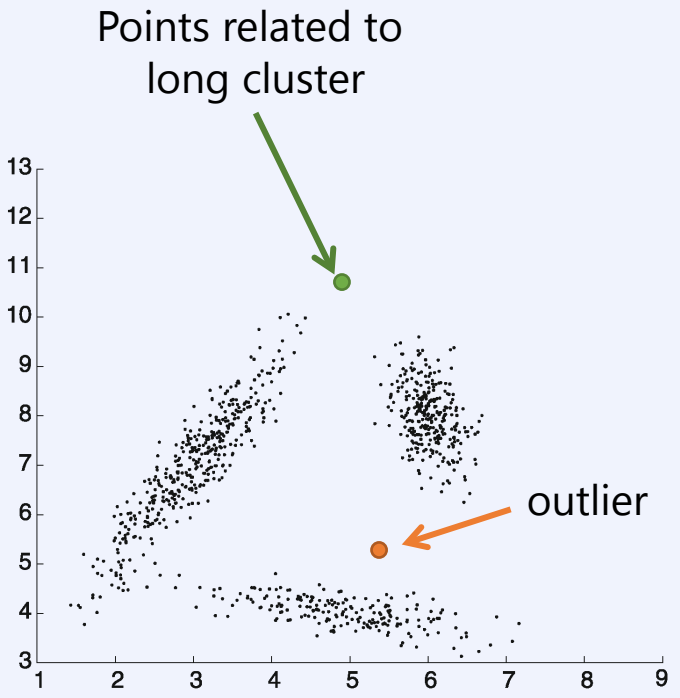
We can also again employ **local** Mahalanobis distance

- simply compute $M(\cdot)$ over $L_k(x)$, i.e. $LM_k(x) = M(x, \mu_k, \Sigma_k)$
- tells how extreme a point x is with regard to its local neighborhood
- no need to normalise, as M does that behind the scenes!

Impact of Locality



LOF



Local Mahalanobis

Conclusions

Outliers are generated by a **different process**

- not noise, but 'nuggets of knowledge', identifying exceptions in your data

Discovering outliers is **non-trivial**

- reduces to the core question of data mining: what is normal?

We have seen four different **classic** approaches

- extreme value analysis, probabilistic, cluster, and distance-based methods

Discovering outliers in **complex data** is **very challenging**

- what does outlying mean in high-dimensional data?
- what does outlying mean in a graph?

Thank you!

Outliers are generated by a **different process**

- not noise, but 'nuggets of knowledge', identifying exceptions in your data

Discovering outliers is **non-trivial**

- reduces to the core question of data mining: what is normal?

We have seen four different **classic** approaches

- extreme value analysis, probabilistic, cluster, and distance-based methods

Discovering outliers in **complex data** is **very challenging**

- what does outlying mean in high-dimensional data?
- what does outlying mean in a graph?