Outlier Detection Methods

L. Torgo

ltorgo@fc.up.pt
Faculdade de Ciências / LIAAD-INESC TEC, LA
Universidade do Porto

June, 2015

Outlier Detection Methods

Main Goal: Find patterns in a data set that are not in agreement with what is expected (i.e. what is more frequently found in the data).

Importance: In many applications outliers correspond to critical situations with high associated costs and requiring preventive and/or corrective actions to be taken.

Outliers and noise: outliers are frequently treated as noise or data errors and thus eliminated from posterior analyzes. Here we are interested in other types of applications where outliers correspond to interesting situations for the user.

Main Challenges

- It is not always easy to characterize “normal” behavior.
- The frontier between normal and deviating behavior is frequently imprecise.
- In applications where outliers are associated with malicious activity, the agents performing these actions try to disguise this activity as normal.
- Frequently the available data also contains noise or errors, which makes the task of identifying “real” outliers harder.
- Frequently the notion of outlier changes with time.

Types of Outliers

- Point outliers
  Cases that either individually or in small groups are very different from the others.
- Contextual outliers
  Cases that can only be regarded as outliers when taking the context where they occur into account.
- Collective outliers
  Cases that individually cannot be considered strange, but together with other associated cases are clearly outliers.

Types of Outlier Detection Methods

- Supervised methods
  When we have access to an historical data set with cases that were tagged as being normal or outliers.
- Non-supervised methods
  When there is no previous information on the available data concerning the cases being or not outliers.
- Semi-supervised methods
  Some of the available cases are tagged while others are not (the latter being frequently the majority).

Type of Results Obtained by the Outlier Detection Methods

- Tags
  Each case is tagged as being or not an outlier.
- Scores
  Each case is given a score that can be seen as a kind of degree of outlyingness. These scores can be used to obtain rankings of outliers, which is interesting for several applications.

Uni-variate Methods

These methods try to look at the observed distribution of values of a continuous variable comparing it to an expected distribution in order to conclude whether there are outliers in the sample.
An example is the method used to obtain boxplots. This method assumes a near-normal distribution of the values and tags as outliers any values outside the interval,

$$[Q_1 - 1.5 \times IQR \cdots Q_3 + 1.5 \times IQR]$$

where Q_1 (Q_3) is the 1st (3rd) quartile and IQR is the inter-quartile range ($= Q_3 - Q_1$)

Methods based on Probability Distributions

- Assume a certain probability distribution (e.g. a multi-normal distribution).
- Use statistical tests (known as *discordancy tests* to obtain the probability of a certain case belonging or not (being an outlier) to the selected distribution.
- On complex problems it may be difficult to find the adequate distribution.
- Parametric methods use well-known probability distribution functions
- Non-parametric methods do not assume a known probability distribution function. They estimate it from the data (e.g. using kernel approaches)

Distance-based Methods

Basic Assumption

Outliers are very different from other cases thus given a distance metric we should not see a lot of other cases in the neighborhood of an outlier

Some Key Issues

- Define a proper distance metric that reflects the actual differences among cases
- Efficiently compute distances and neighborhoods
- Decide on which is a reasonable of a neighborhood
- Decide on what is “a lot of other cases”

The Notion of Similarity

- The key issue on outlier detection is the notion of similarity
- This notion is strongly related with the notion of distance between observations
- Distances among observations in a data set can be used to decide on whether cases are or not outliers

How to Calculate the Distance between 2 Cases?

- The notion of distance is related to the differences between the values on the variables describing the cases

ID	Income	Sex	Position	Age
1	2500	f	manager	35
2	2750	f	manager	30
3	4550	m	director	50

Case 1 is “closer” to case 2 than to 3

The Euclidean Distance Function

$$d(\mathbf{x},\mathbf{y}) = \sqrt{\sum_{i=1}^{p}(x_i - y_i)^2}$$

where x_i is the value of case $\mathbf{x}$ on variable i

Example

Given two cases $\mathbf{x} =< 3, 5, 1 >$ and $\mathbf{y} =< 12, 5.4, -3 >$ their Euclidean distance is given by

$$d(\mathbf{x},\mathbf{y}) = \sqrt{(3-12)^2 + (5-5.4)^2 + (1-(-3))^2} = 9.85697$$

A Generalization - the Minkowski distance

$$d(\mathbf{x},\mathbf{y}) = \left(\sum_{i=1}^{p} |x_i - y_i|^r\right)^{1/r}$$

where if

- $r = 1$ we have what is known as the Manhattan distance (or L_1-norm)
- $r = 2$ we have the Euclidean distance
- etc.

Potential Problems with Distance Calculation

In domains where cases are described by many variables several problems may arise that may distort the notion of distance between any two cases.

- Different scales of variables
- Different importance of variables
- Different types of data (e.g. both numeric and nominal variables, et.c)
- etc.

Heterogeneous Distance Functions

How to calculate the distance between two cases described by variables with different type (e.g. numeric and nominal variables)?
A possible solution,

$$d(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{p} \delta_i(x_i, y_i)$$

emque,

$$\delta_i(v_1, v_2) = \begin{cases} 1 & \text{if } i \text{ is nominal e } v_1 \neq v_2 \\ 0 & \text{if } i \text{ is nominal e } v_1 = v_2 \\ \frac{|v_1 - v_2|}{range(i)} & \text{if } i \text{ is numeric} \end{cases}$$

The distance between $<$ 2500, f, director, 35 $>$ and $<$ 2750, f, director, 30 $>$ would be given by $\frac{|2500-2750|}{range(Salary)} + 0 + 0 + \frac{|35-30|}{range(Age)}$

Distance-based Outlier Detection Methods

Global Methods

A case c is an outlier if less than k cases are within a distance λ of c

- They usually proceed in two main steps:
  1. Calculating the neighborhood of each case
  2. Inspect the size of this neighborhood and decide whether the case is or not an outlier

Global Methods

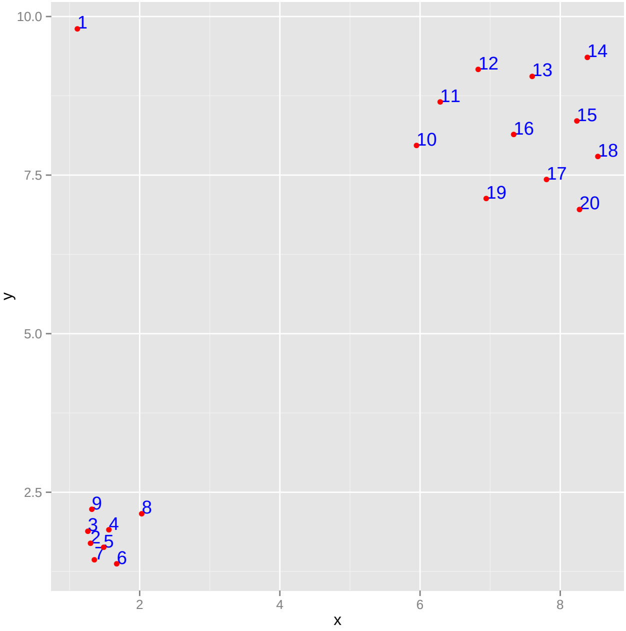
The Big Questions

- How to set the values of *k* and λ?
  - Trial and error is the standard answer to this
- How to make the overall process computationally efficient?
  - Use specialized algorithms to make the distance calculation efficient - e.g. R trees, k-d trees, nested loop algorithm to minimize I/O, the cell-based algorithm for memory-based data sets, etc.

Global Methods

Main Problem

Global distance-based outlier detection methods have difficulties in data sets with different density regions (i.e. non-uniform distributions).



The DBSCAN Method

This is a clustering method based on the notion of “density” of the observations

Key Idea: The density of a single observation is estimated by the number of observations that are within a certain radius (a parameter of the method)

Based on this idea observations are classified as:

- *core points*: if the number of observations within its radius are above a certain threshold
- *border points*: if the number of observations within their radius does not reach the threshold but they are within the radius of a core point
- *noise points*: they do not have enough observations within their radius, nor are they sufficiently close to any core point

The DBSCAN Method (cont.)

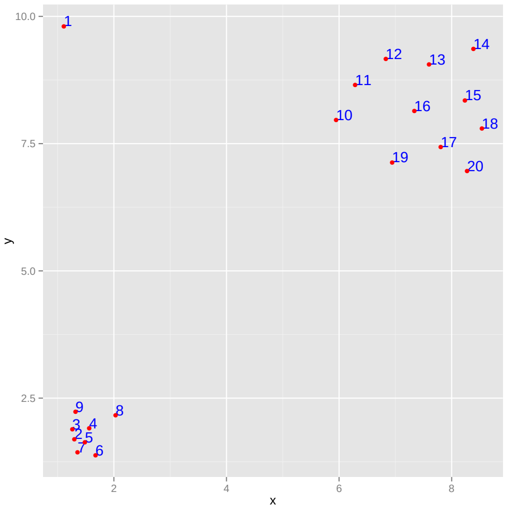
The DBSCAN algorithm

- Classify each observation in one of the three possible alternatives
- Eliminate the *noise points* from the formation of the groups
- All *core points* that are within a certain distance of each other are allocated to the same group
- Each *border point* is allocated to the group of the nearest *core point*

NOTE: Cluster 0 contains the noise points (outliers)

The simple problem with local outliers

DBSCAN

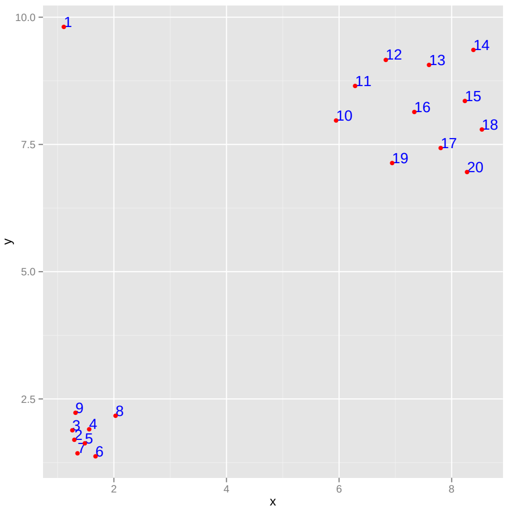


```
(d <- dbscan(dados,1,2))

## dbscan Pts=20 MinPts=2 eps=1
##         0 1  2
## border 1 0  0
## seed   0 8 11
## total  1 8 11

d$cluster

##  [1] 0 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2

which(d$cluster==0)

## [1] 1
```

```
library(fpc)
load("pontos.Rdata")
```

The simple problem with local outliers

DBSCAN - 2



```
(d <- dbscan(dados,1.2,5))

## dbscan Pts=20 MinPts=5 eps=1.2
##         0 1  2
## border 2 0  6
## seed   0 8  4
## total  2 8 10

d$cluster

##  [1] 0 1 1 1 1 1 1 1 1 0 2 2 2 2 2 2 2 2 2 2

which(d$cluster==0)

## [1]  1 10
```

Distance-based Methods

Local Methods

The Idea

Look at the neighborhood of a case, namely its density, before deciding whether it is or not an outlier.

- They usually proceed in two main steps:
  1. Calculating the density on different regions of the input space
  2. Find outliers based on the density information

The *LOF* Method

- *LOF* is probably the most well-know local distance-based outlier detection algorithm.
- The result of *LOF* is a local outlier score for each case.
- This means that the result can be regarded as a kind of ranking of outlyingness of the data set

The *LOF* Method (cont.)

Key Notions in *LOF*

- The k-distance of an object o ($dist_k(o)$) is the distance from o to its kth nearest neighbor
- The k-distance neighborhood of o ($N_k(o)$) are the set of k nearest neighbors of o
- The *reachability-distance* of an object o with respect to another object p is defined as $reach.dist_k(o,p) = \max\{dist_k(p), d(o,p)\}$. If p and o are faraway from each other the reachability distance will be equal to their actual distance. If they are close to each other then this distance is substituted by $dist_k(p)$.

The *LOF* Method (cont.)

Key Notions in *LOF* (cont.)

- The *local reachability-distance* is the inverse of the average *reachability-distance* of its k-neighborhood,

$$lrd_k(o) = \frac{|N_k(o)|}{\sum_{p \in N_k(o)} reach.dist_k(o,p)}$$

The *LOF* Method (cont.)

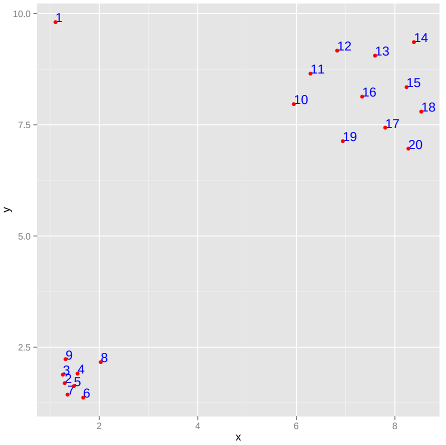
The *LOF* score

The *LOF* score of an observation captures the degree to which we can consider it an outlier. It is given by,

$$LOF_k(o) = \frac{\sum_{p \in N_k(o)} \frac{lrd_k(o)}{lrd_k(p)}}{N_k(o)}$$

This factor is the average of the ratio between the local reachability-distance of o and those of its k-nearest neighbors.

The simple problem with *LOF*



```
library(DMwR)
(lof <- lofactor(dados,3))

##  [1] 4.5144363 0.9854897 0.9931015 1.0286112 0.9382561 1.1684333 1.0468941
##  [8] 1.9385895 1.3457809 1.1942415 1.1480213 0.9852106 1.1583481 1.1061588
## [15] 0.8852066 0.8701919 1.0659487 1.0476725 1.0587496 0.9483320

order(lof,decreasing=T)

##  [1]  1  8  9 10  6 13 11 14 17 19 18  7  4  3  2 12 20  5 15 16
```

Outlier Detection using Clustering Methods

Assumptions/Setup

- There is a historical record of data on some phenomenon
- There is no information on these cases being or not outliers

- Among several techniques we can use in this setting, clustering methods are the most used.
- As a result of a clustering process we can obtain important information regards the similarity of cases
- Namely, outliers are supposed to be very different from other cases and thus should belong to very small groups or even being completely isolated

Using the Clustering Results

A simple approach

- Obtain a clustering of the available data using some algorithm
- Decide on the threshold for a group to be considered too small
- Tag all members of groups with less members than the threshold as outliers

- How many groups should we use during clustering?
- What is the size threshold?

Hypothesis: use the information from the silhouette coefficient

The OR_H algorithm

- Obtain an agglomerative hierarchical clustering of the data set
- Use the information on the “path” of each observation through the dendrogram as a form to determine its degree of outlyingness
  - Idea / motivation: cases that are only merged at later stages are surely very different from others

The OR_H algorithm

- Define the degree of outlyingness of a case as,

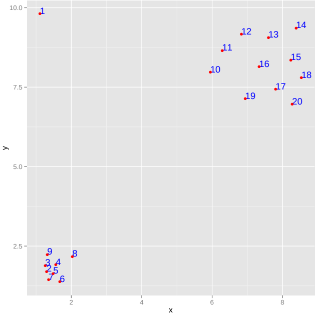
  $$OR_H = \max_i of_i(x)$$

  where i represents the merging iteration within the agglomerative process and can take values from 1 to $N - 1$, where N is the number of observations in the data set.
- The OR_H algorithm proposes several methods to determine the individual score of an observation within each merging step ($of_i(x)$). The default method is called “SizeDiff” and is defined as,

  $$of_i(x) = \max\left(0, \frac{|g_{y,i}| - |g_{x,i}|}{|g_{y,i}| + |g_{x,i}|}\right)$$

  where $g_{y,i}$ and $g_{x,i}$ are the two groups involved in the merging at stage i and $g_{x,i}$ is the group to which x belongs.

The simple problem with OR_H



```
library(DMwR)
or <- outliers.ranking(dados)
or$rank.outliers


##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  1  8  9  4  6 10 16 19  2  3  5  7 13 14 11 12 15 17 18 20


round(or$prob.outliers,3)


##     1     2     3     4     5     6     7     8     9    10    11    12
## 0.833 0.200 0.200 0.333 0.200 0.333 0.200 0.750 0.500 0.333 0.091 0.091
##    13    14    15    16    17    18    19    20
## 0.200 0.200 0.000 0.333 0.000 0.000 0.333 0.000
```

Supervised Methods of Outlier Detection

Assumptions

- There is an historical record of data
- For each case there is information on whether it is or not an outlier

Goal

The goal of supervised methods is to obtain a model that is able to discriminate among outlier and normal cases

- There are several data mining techniques that can be used to achieve this goal
- The main particularity/difficulty of the data sets of these problems lies on the fact that there are usually much more “normal” cases than outliers.

Supervised Classification Methods

What is a classification problem?

- Given *n* cases described by *p* variables, where one of them (the target variable) describes the class of the case (e.g. is or not an outlier)
- Try to obtain a model that relates possible values of the target variable (known as the classes) to the other variables (sometimes called the predictors) describing the cases

Supervised Classification Methods (cont.)

- There are many possible models that try to capture the relationship between the target and the predictors (e.g. linear discriminants, neural networks, classification trees, SVMs, etc.)
- After we choose a type of model we have an optimization problem:
  - Given a evaluation criterion for the performance of the models
  - Given the data sample we have
  - and given the parameters of the chosen model
  - “Find” the values of the parameters using the given data sample that “optimize” the selected evaluation criterion
  - Selecting the “right” criterion given the unbalanced distribution of the data set is critical for outlier detection!

Predicting outliers

- Problems with two classes (**normal** vs. **outlier**)
- One of the classes is much less frequent and it is also the most relevant
- The confusion matrix for a given set of test cases

		Preds.	
		Pos	Neg
Obs.	Pos	True Positives (TP)	False Negatives (FN))
	Neg	False Positives (FP)	True Negatives (TN)

Precision and *Recall*

		Preds.	
		P	N
Obs.	P	TP	FN
	N	FP	TN

- *Precision* - proportion of the signals (events) of the model that are correct

$$Prec = \frac{TP}{TP + FP}$$

- *Recall* - proportion of the real events that are captured by the model

$$Rec = \frac{TP}{TP + FN}$$

Precision and *Recall*

Examples

		Prevs.	
		P	N
Obs.	P	2	2
	N	1	1

$$Precision = \frac{TP}{TP + FP} = \frac{2}{2+1} = 0.667$$

$$Recall = \frac{TP}{TP + FN} = \frac{2}{2+2} = 0.5$$

$$ErrorRate = 1 - \frac{2+1}{2+2+1+1} = 0.5$$

The F-Measure

Combining Precision and Recall into a single measure

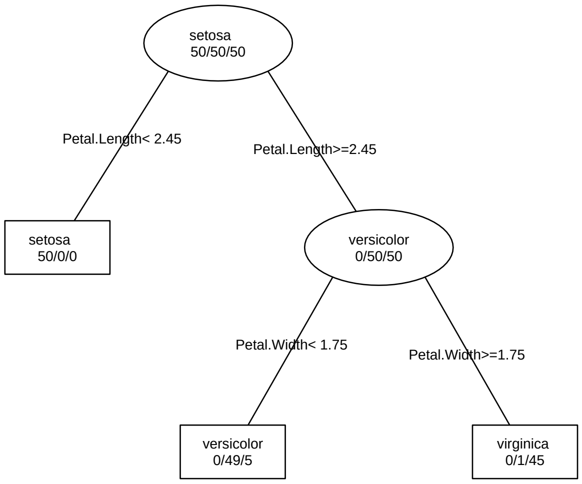
- Sometimes it is useful to have a single measure - e.g. optimization within a search procedure
- Maximizing one of them is easy at the cost of the other (it is easy to have 100% recall - always predict “P”).
- What is difficult is to have both of them with high values
- The F-measure is a statistic that is based on the values of precision and recall and allows establishing a trade-off between the two using a user-defined parameter (β),

$$F_\beta = \frac{(\beta^2 + 1) \cdot Prec \cdot Rec}{\beta^2 \cdot Prec + Rec}$$

where β controls the relative importance of *Prec* and *Rec*. If $\beta = 1$ then *F* is the harmonic mean between *Prec* and *Rec*; When $\beta \rightarrow 0$ the weight of *Rec* decreases. When $\beta \rightarrow \infty$ the weight of *Prec* decreases.

An example - classification trees

Very simple models that have as one of the main advantages the fact that they produce an interpretable description of what characterizes each class.

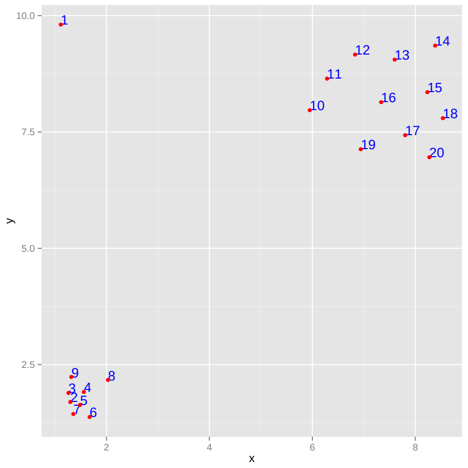


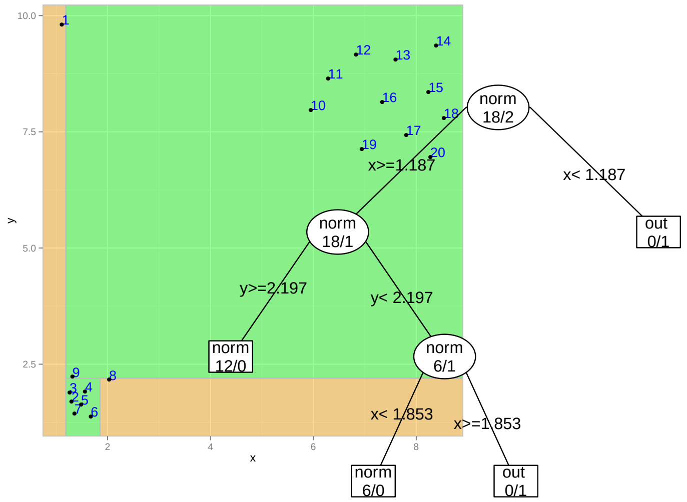
Classification trees

- These models are obtained using an iterative algorithm that builds the trees from top (root node) to the bottom (the leaves)
- One each step (iteration) a logical test is chosen with the goal of trying to discriminate the cases of the existing classes
- Each test divides the data sample in two sub-sets - the one with cases satisfying the test, and the other with the remaining cases
- For each sub-set the same algorithm is applied
- The process ends when certain statistical tests signal that it is safer to stop in terms of generalization power of the resulting tree (i.e. ability to perform well on unseen data)

The simple problem with local outliers

If we tag cases 1 and 8 as outliers and the remaining as normal, we get the following tree:





Semi-Supervised Methods

- On certain applications there is information on the class of the cases only for certain cases.
- This is typical for instance in fraud detection cases where there are no sufficient resources to inspect all historical data.

Alternative Approaches

- Adapt the supervised methods to be able to take advantage of the information on the cases for which no class is known
  - An example is the self-training method
- Change the unsupervised methods to incorporate the information on the classes during the clustering process
  - E.g. try to ensure that cases of different classes are not put on the same group