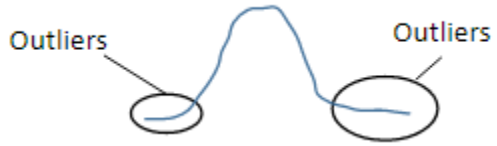
Outlier Detection - Top N LOF (2017)

Introduction

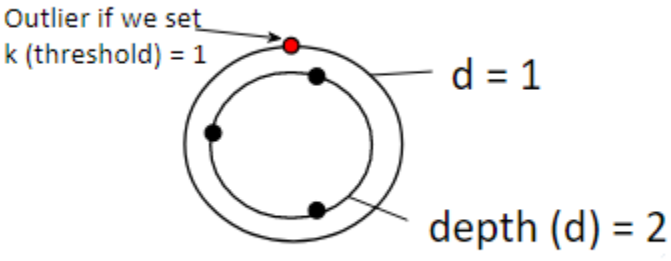
5 broad categories for outlier detection:

1. Distribution-based



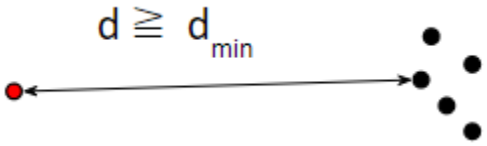
   Normal distribution, Poisson distribution, etc

2. Depth-based

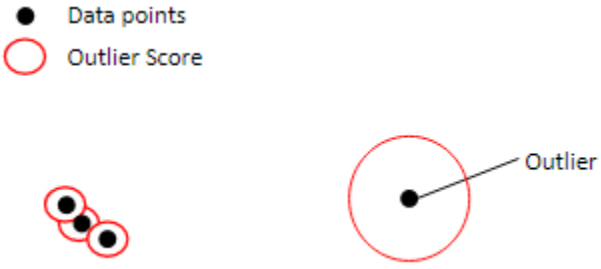


   Uses the concept of k-d convex hulls

3. Distance-based



4. Clustering-based
   - DBSCAN, BIRCH, CURE, etc
   - By products of clustering
5. Density-based



   Top N LOF is in this category
   - Uses nearest neighbor concept

- Improves on the clustering-based algorithms

Motivation

Past work done: Sklearn outlier detection package (sklearn.neighbours.LocalOutlierFactor)
Drawback: Have to compute the LOF for every point (not scalable)
Top N LOF solution:

- Most points have small LOF values and are unlikely to be outliers. These points should be pruned first to avoid computing their final LOF scores, thus reducing time complexity.
- Pruning is done after estimating the LOF bounds for each point.

Use cases:

Prune anomalous data in preprocessing

- https://dspace.networks.imdea.org/bitstream/handle/20.500.12761/1593/Netpredict3_Cleaning_Matters__Preprocessing_enhanced_Anomaly_Detection_and_Classification_in_Mobile_Networks.pdf?sequence=1
  This paper (2021) removes features with a lot of outliers. Technique is a statistical method.
  1. For each feature, each data point is determined if it is an outlier. It is deemed an outlier if its value is greater or less than median ± 3MAD (Mean Absolute Deviation)
  2. For each feature, the number of outliers will be calculated.
  3. Features with their number of outliers ≥ median + 3MAD will be discarded

Anomaly detection

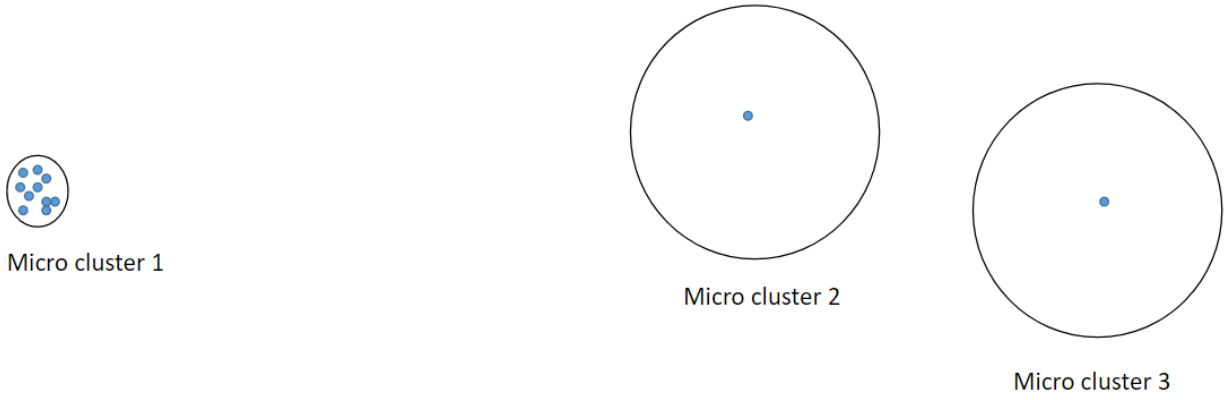
- https://datrics.ai/anomaly-detection-definition-best-practices-and-use-cases
  This link describes 3 broad categories for anomaly detection.
  1. Density-based anomaly detection
     KNN methods, link to LOF
  2. Clustering-based anomaly detection
     By-product of clustering algorithm
  3. SVM-based anomaly detection
     Use a soft-margin svm. Points outside the soft margins are deemed outliers

Detecting attrition

Outlier detection can be used to detect rare events (for example, fraud events, etc)
The definition of outliers is a minority anyway.

Main Idea

The idea of estimating the LOF bounds for a point is extended to a cluster



1. The upper and lower bounds of LOF for each micro cluster are computed.
2. Those micro clusters with very small upper bound LOF (i.e. Micro Cluster 1) will be discarded.
3. The LOF scores for those points in micro clusters with high upper bound LOF (i.e. Micro Clusters 2 & 3) will be calculated.

Algorithm

Step 1: Preprocessing

Objective is to get all micro clusters.
1. Insert all points into a CF (Clustering Feature) tree (from BIRCH clustering algorithm). Output is all CF nodes (centres of the clusters).
2. Assign all points to the nearest centres. Each CF node will form a micro cluster. The number of points in each micro cluster and the radius of each micro cluster will be recorded.

Lit review of BIRCH

BIRCH - Balanced Iterative Reducing and Clustering using Hierarchies (1996)
(https://dl.acm.org/doi/pdf/10.1145/235968.233324)
Main Idea:
Step 1: Build a Clustering Feature (CF) tree

- What is CF?
  CF is basically a vector with 3 values (N, LS, SS) that summarises that information of a cluster.
  N is the number of data points in a cluster.
  LS is a vector which is the Linear Sum of the N data points, summing across each dimension.

SS is the Square Sum of the N data points. The concept is similar to LS, except that each value is squared first before summing across each dimension.
LS and SS are used to assign points to nodes (clusters).

- What is a CF tree?
  First, a tree is made up of non-leaf and leaf nodes.
  Each node has a CF vector. This is probably why it is called a CF tree.
  Non Leaf node: Contains the CF vector and a pointer to each of the child nodes (just like in C++). It doesn't contain the CF vectors of the child nodes.
  Leaf node: Contains its own CF vector. Additionally, it contains pointers to indicate which is the previous leaf node and next leaf node. This is to save time when scanning the tree subsequently.
- Note, a leaf node can represent a cluster of points, but the cluster must be small enough (the radius cannot exceed a threshold T which is specified by users). The threshold may be adjusted in the later stages of the algorithm when ram cannot store the tree in memory.
  The checking condition can be found at (https://github.com/yizhouyan/TopNLOFKDD/blob/39598b293192375493b8c5d07ed0ee0df95105cc/src/net/cftree/mcrtree/CFNode.java#L131)
- If the dataset is very big, then what if the tree cannot be stored in memory as the tree is being built?
  Increase T. T is too small already. There are too many leaf nodes, increasing the size of the tree, causing it to be too big to be stored in memory.
  Now, the tree needs to be rebuilt. How? This is where the storing of the previous and next leaf node is important.
  The leaf nodes are scanned in order. The leaf nodes are checked if they can be merged because T has increased. If it can, then it will be merged then those ex-parent nodes of the merged leaf nodes will be the leaf nodes if the whole leaf node is merged.
  Once all ex-leaf nodes are checked, scanning of the dataset will resume to build the tree.
  If the tree is too big to fit into memory again, this process will repeat.
- How is T increased?
  In Top N LOF, first of all, a distance variable dist will be calculated.
  For each data point in the leaf nodes, the nearest point to it will be determined to calculate the distance d and increment dist by d.
  New threshold = dist / total number of data points in the leaf nodes
  If this value is smaller than or equal to the current threshold, then the new threshold is set as 2 * current threshold.
  This will guarantee that the new threshold is always larger than the current threshold.

Step 2: Build a smaller CF tree (optional, and not implemented in top n lof)

Step 3: Global Clustering (not implemented in top n lof)
This step clusters the leaf nodes to obtain better clusters.

Step 4: Cluster refining (not implemented in top n lof)

Reassign all data points according to the new clusters.
Requires looking through all data points again.

The output is a CF tree which will be used to generate microclusters. Each leaf node will be a microcluster. Subsequently, all points will be assigned to the nearest microcluster.

Step 2.1: Compute k-distance bound for each micro cluster

Intuition of k-distance: When the k-distance of an object is small, it means that there are a lot of points near the object
Input: A set of micro clusters
Output: k_{min}-distance & k_{max}-distance of each micro cluster

Step 2.2 - Compute LOF bound for each micro cluster

Input: Each micro cluster and its k-distance bound
Output: LOF bound for each micro cluster

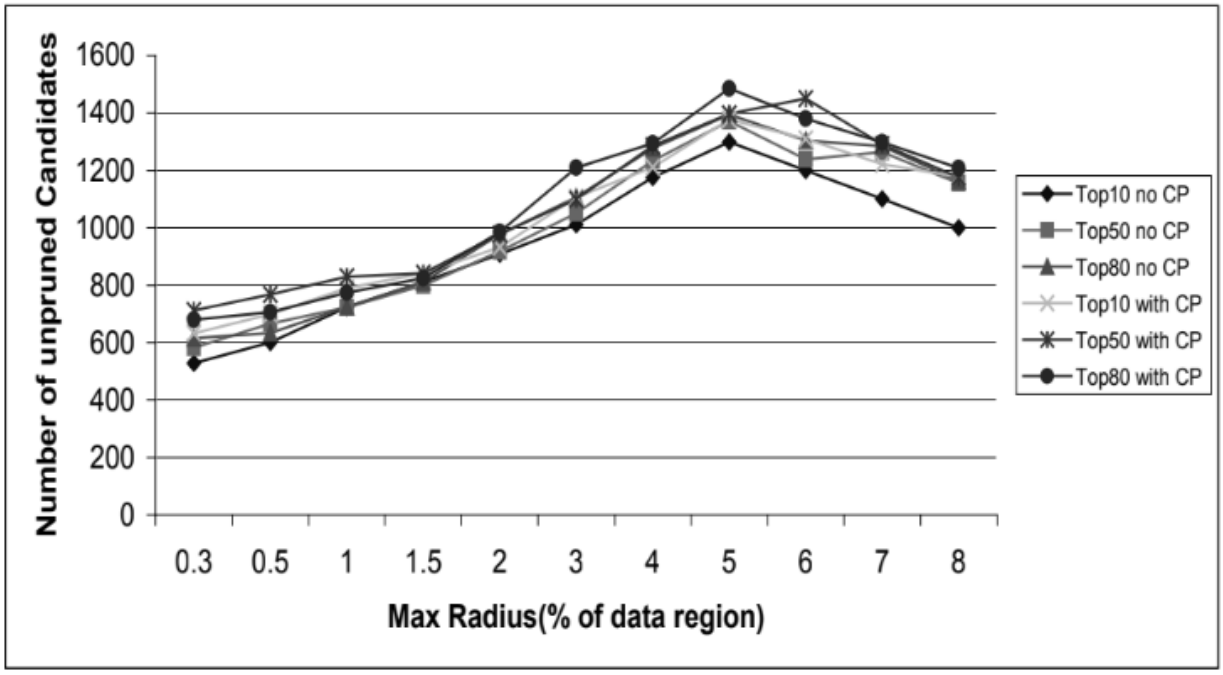
Step 3 - Rank Top-n Local Outliers

Input: Each micro cluster and its LOF-bound
Output: Top-n local outliers

Results

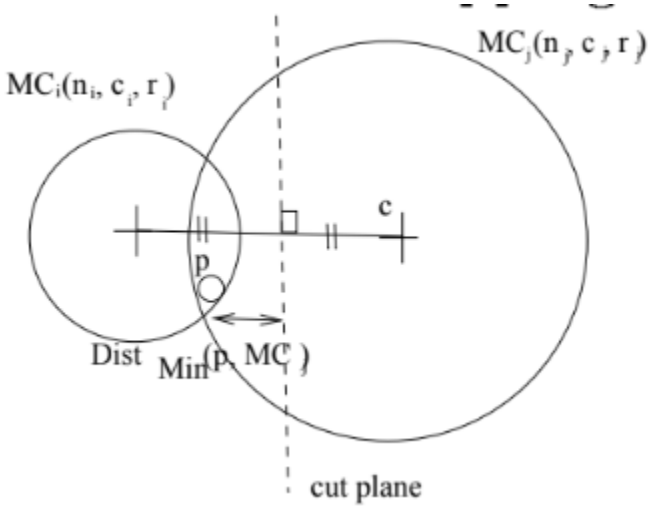
Number of unpruned candidates vs Max Radius (evaluation metric 1 in current paper)

- Unpruned candidates refer to points who eventually are not flagged as outliers but has a high upper LOF score
- Max Radius refer to the cluster radius



$$LOF(MC_i).upper = r_{max}(MC_i)/r_{min}(MC_i);$$
$$LOF(MC_i).lower = r_{min}(MC_i)/r_{max}(MC_i);$$

- As max radius increases, more points will be unpruned as the upper LOF will increase.
- This phenomenon continues until a point where there are too few clusters formed because of the large max radius. This will then result in a decrease in the number of unpruned candidates.

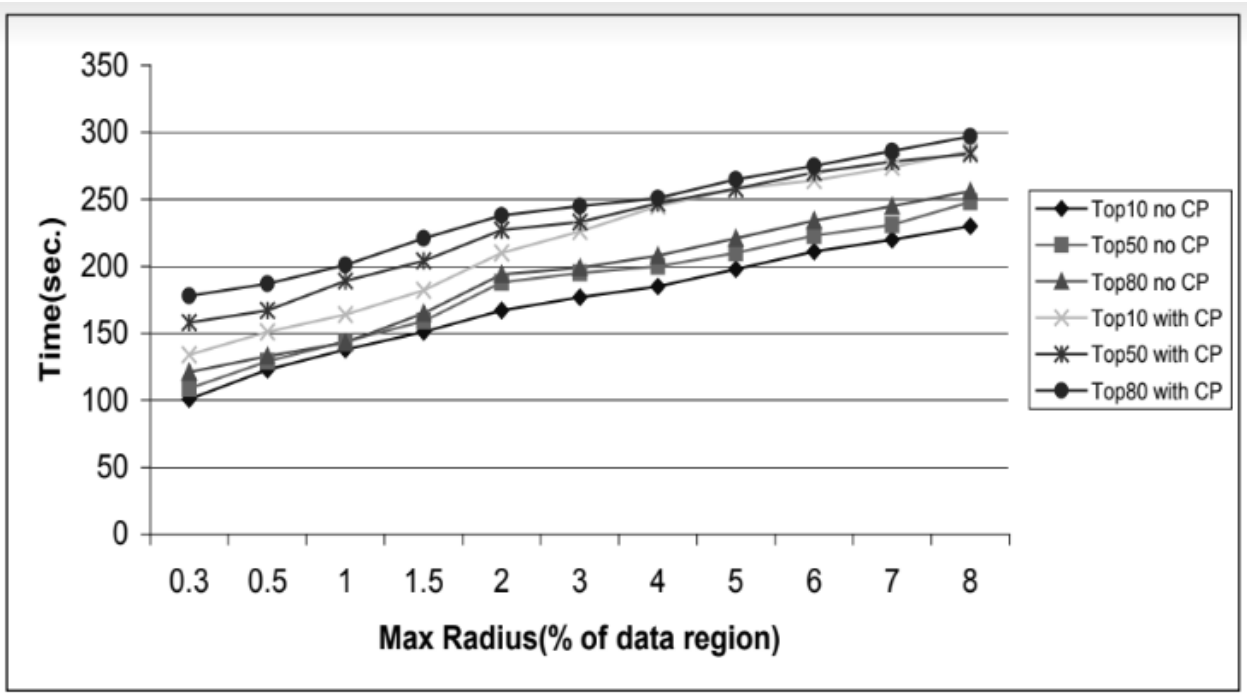


- CP is defined as a cut plane. With CP, the distance between P and MC increases from 0 to a positive number. Hence, it is more likely that points

become outliers. This leads to more unpruned candidates (false positives).

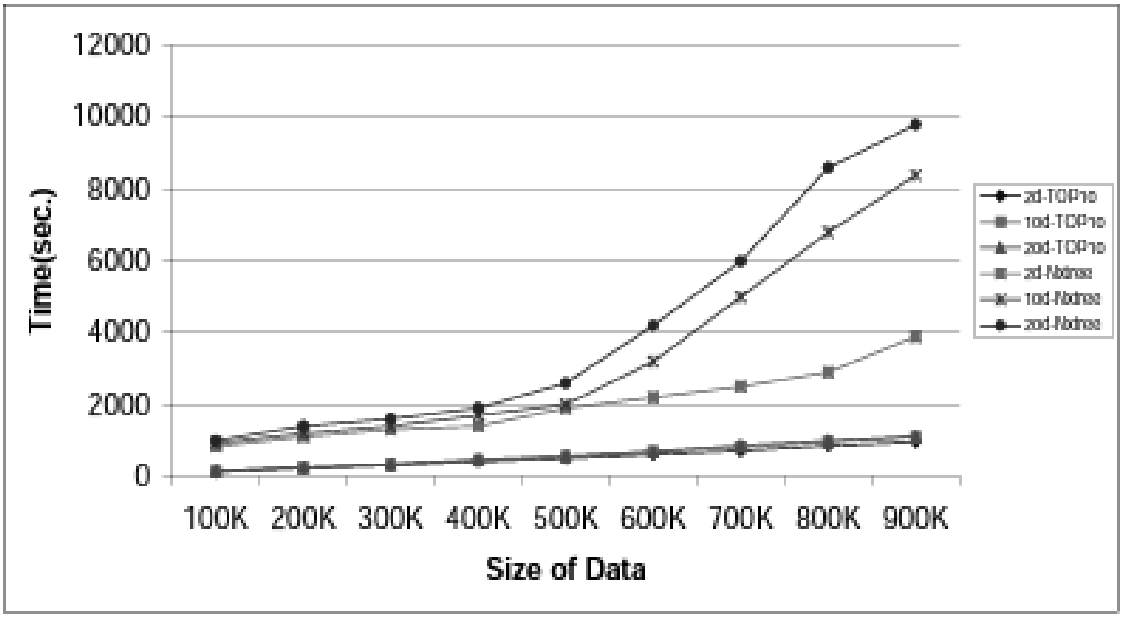
- This metric is only useful if the algorithm uses a similar technique, where it will prune candidates. It is not relevant for sklearn since sklearn calculates lof scores for all points. It is not a universal metric like accuracy / runtime.
- The intention is to show that increasing max radius will not significantly increase the number of unpruned candidates. Hence, this will not significantly increase runtime.

Running time vs max radius (evaluation metric 2 in current paper)



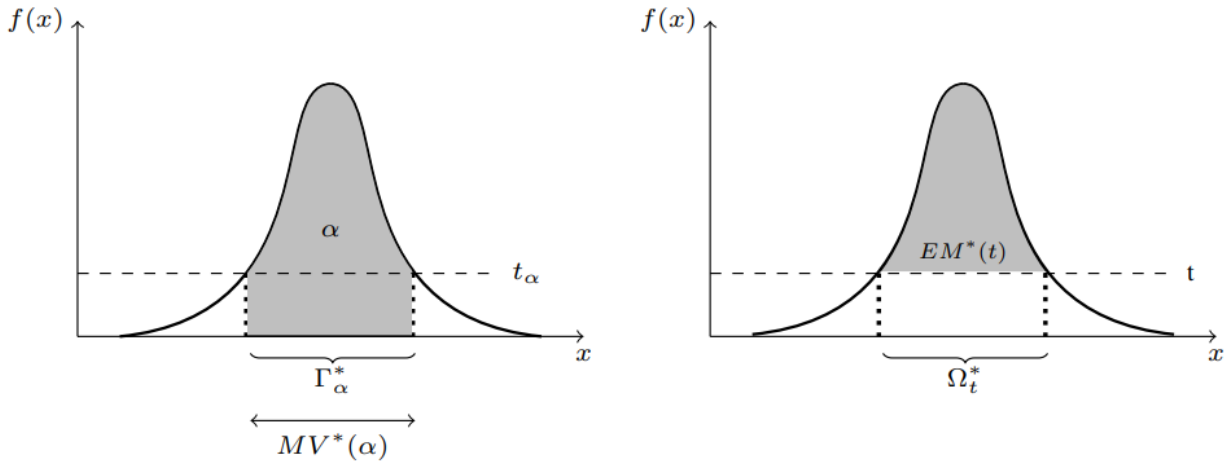
- As max radius increases, runtime increases as each cluster has more points. For those shortlisted clusters, more LOF scores need to be calculated.
- This metric is only useful if the algorithm does some clustering before detecting outliers. It is not relevant for sklearn. It is not a universal metric like accuracy / runtime.
- The intention is to show that increasing max radius will not significantly increase run time.

Runtime vs size of data (evaluation metric 3 in current paper)



Outperforms naive X-tree based LOF method for high dimensions.
Rather than using run time, there are better metrics (https://arxiv.org/pdf/1607.01152.pdf).
For labelled datasets, ROC / PR / AUC can be used
The EM and MV metrics can be used on unlabeled datasets. (2016)
Code is in Python
The algorithm can be tweaked to handle high dimensions.

Figure 1. Comparison between $MV^*(\alpha)$ and $EM^*(t)$



Source Codes

Background

Language: Java
Input: Dataframe
Output: Top n local outliers
Other parameters:

1. k for KNN (k)
2. N to indicate how many outliers to return (n)
3. Dimensionality of dataset (d)
    If specify less, take fewer columns. If specify more, will throw error
4. Domain Range (for BIRCH clustering) (r)
    Domain range of the given dataset [0,domainRange]
    Something like the magnitude of the maximum value…?
    Clarified with Prof, r is the distance between the center of the data and the furthest data point
5. Cluster radius (for BIRCH clustering) (c)

Threshold T = domain range * cluster radius

Metrics Used

Either L1 or L2 norm (look at src.metricspace.MetricFactory)
Default is L2 norm (look at src.util.SQConfig)
Possible to add custom metrics (L3 Metric, etc)

Command to extract jar files: jar xf [filename].jar

Java command (to compile): javac -cp lib -sourcepath src -d lib src/microcluster/topnlof/TopNLOFDetection.java

Java command (to run): java -cp lib microcluster.topnlof.TopNLOFDetection -c 2 -d 2 -i data/accuracy/dimension_20.csv -k 2 -n 9000 -o data/accuracy_results/results_20 -r 2

LOF scores

The process of computing the LOF score for both Top N LOF and sklearn are compared.

Methodology

Step 1: Find the K-neighbours of a point and obtain the distances to the neighbours

Step 2: Calculate the local reachability density for each point.

- Calculate the reachability distance of a point

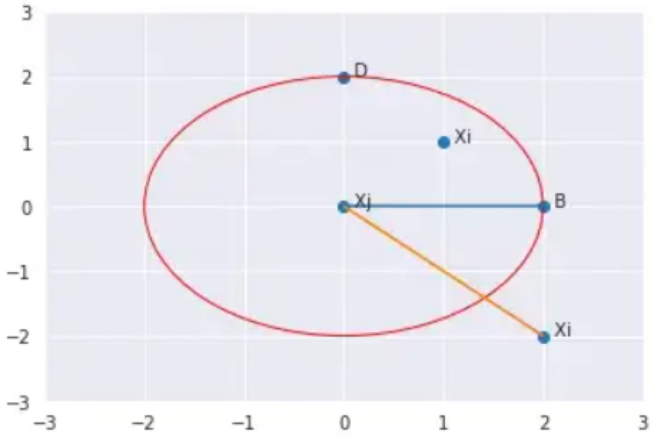


Illustration of reachability distance with K=2

> In layman terms, if a point Xi lies within the K-neighbors of Xj, the reachability distance will be K-distance of Xj (blue line), else reachability distance will be the distance between Xi and Xj (orange line).

The output is a reachability distance for each neighbour for a focal point.
In the figure above, suppose the focal point is Xi and Xj is a neighbour. The reachability distance between Xi and Xj can be determined in the above manner.

- Calculate local reachability density

$$Local\ Reachability\ Density\ = \frac{1}{average\ of\ the\ reachability\ distances}$$

Step 3: Compute lof score

- Obtain the ratios of the local reachability density of the neighbours and the focal point

$$Formula\ = average(\frac{local\ reachability\ density\ for\ each\ neighbour\ of\ the\ point}{local\ reachability\ density\ for\ the\ point})$$

- LOF Score = negative of the average of these ratios

The difference between Top N LOF and sklearn in calculating the LOF score is that Top N LOF does not negate the average of the ratios at the end. However, it is to note that negation does not affect the definition/interpretation of the LOF scores.

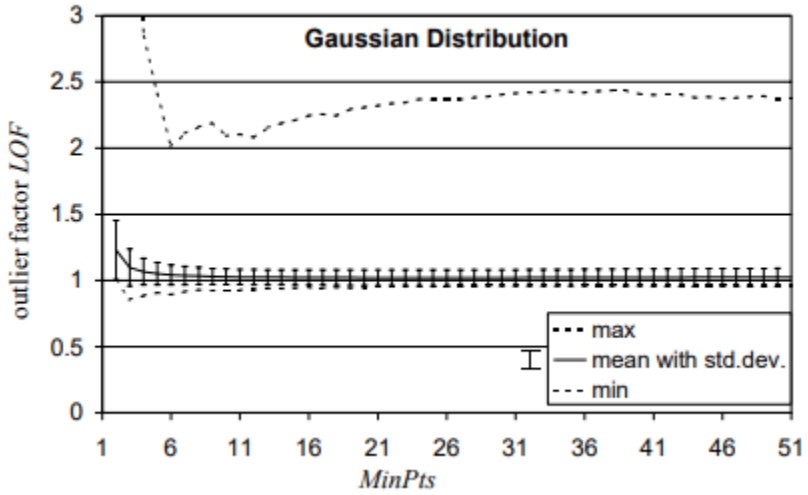
How does sklearn labels outliers or inliers?
(source: https://github.com/scikit-learn/scikit-learn/blob/f3f51f9b6/sklearn/neighbors/_lof.py#L19, refer to the fit function)

If the contamination parameter is not set, then the threshold is 1.5.

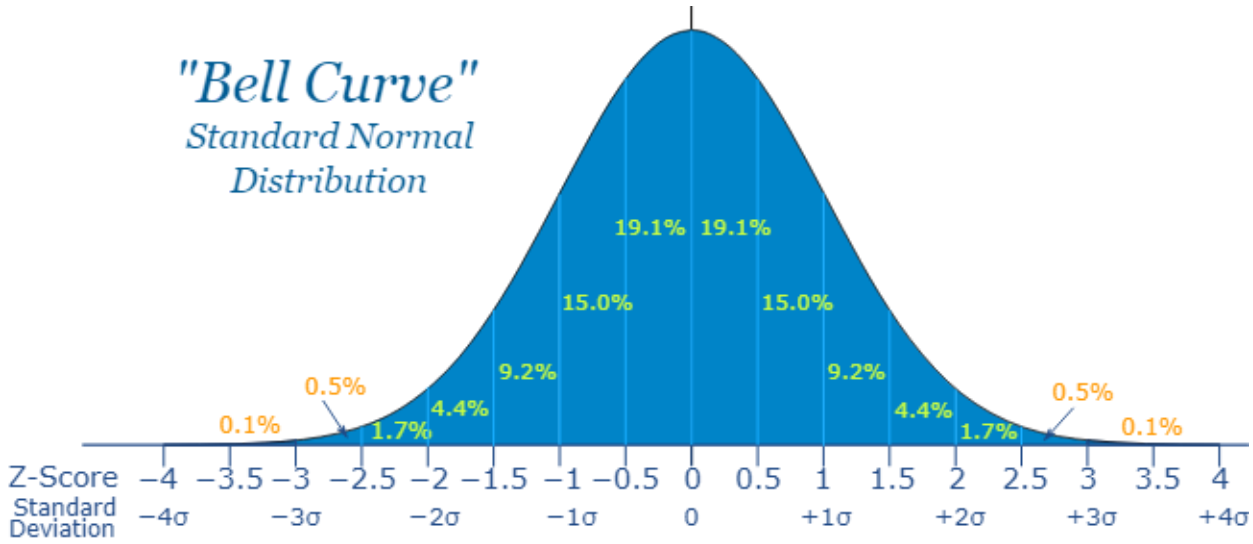
If the contamination parameter is set, then the threshold depends on the nth percentile. For example, if the contamination parameter is set as 0.1, then the top 10% will be labelled as outliers.

Why does sklearn set 1.5?

Sklearn claims that the 1.5 threshold comes from the original paper. 1.5 was used in 1 of the experiments (Section 7.3), not sure if this is what sklearn claims as 'comes from the original paper'.



(source: https://dl.acm.org/doi/pdf/10.1145/335191.335388 Figure 7, from original paper)
MinPts is the number of nearest neighbours to consider.
This experiment showed that inliers have a LOF score of around 1, while the maximum LOF score is between 2 and 2.5. Hence, 1.5 is a sensible threshold. If the threshold is set as 2, too few outliers will be considered.
The default number of nearest neighbours that sklearn uses is 20.



Roughly 87% of data points fall within 1.5 standard deviations.

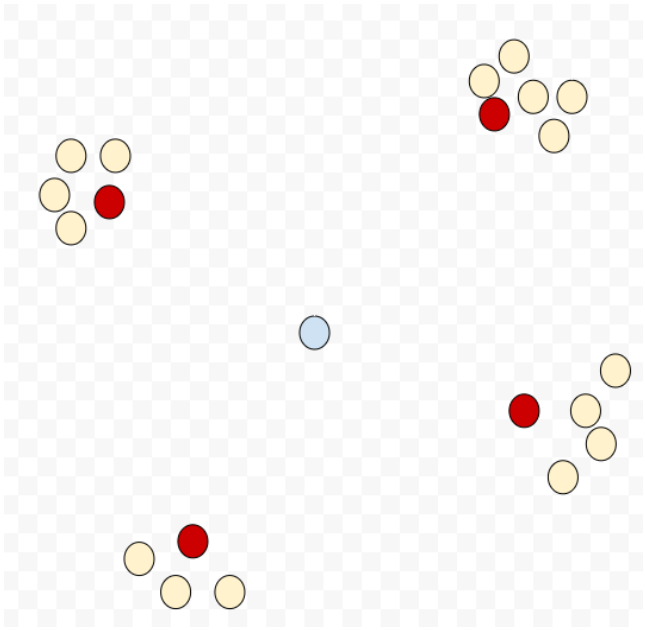
Interpretation

Consider the non-negative LOF score. How to interpret a LOF score of 11?

This means that on average, the ratio of the LRD (Local Reachability Density) of a neighbour to that of the focal point is 11. The local density of a neighbour is 11 times more than that of the focal point.

LRD is the inverse of the average of the reachability distances. If LRD is high, this means that the reachability distances are small. This means that the neighbouring points are close. This means that the density of points is high in that area.

This means that if the densities of the neighbours are high but the density of the focal point is low, as illustrated below.



The blue point is the focal point, the red points are the neighbours of the focal point while the other points are the neighbours of the neighbours. This shows that the focal point is likely to be an outlier.

How to compare 2 points A and B, A with a LOF score of 2 and B with a LOF score of 10?

This means that on average, the local density of the neighbours of B is 5 times more than that of A. Hence, it is 5 times more likely for B to be an outlier than A.

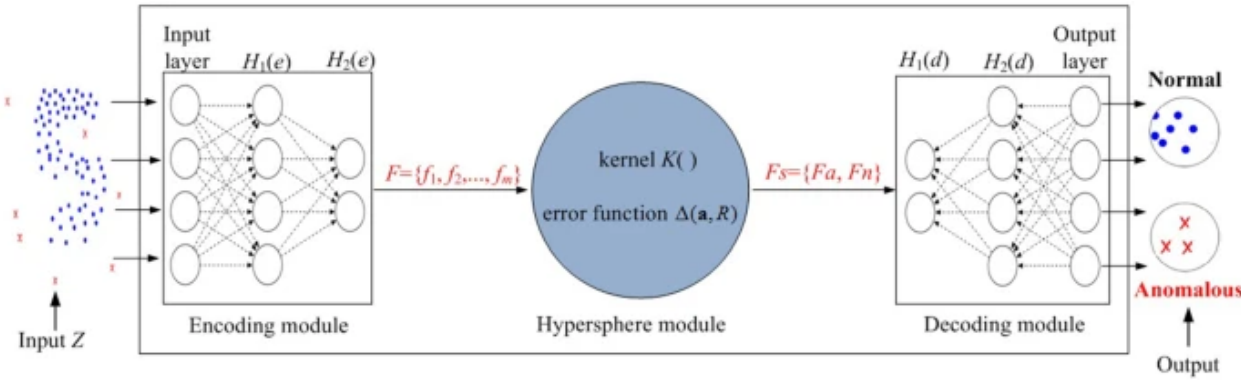
Challenges

High dimensions

- First naive approach

  (https://arxiv.org/ftp/arxiv/papers/1908/1908.04000.pdf)
  Code is in R
  This paper (2019) aims to improve on KNN (but still susceptible to the curse of dimensionality)
  Top N LOF: If there is multiple anomalies together, KNN cannot detect
  Solution:
  1. Calculate the k-nearest neighbor distances
  2. Calculate the successive differences between distances
  3. Select the knn distance with the maximum difference (the elbow in the plot)
- Better approach

  (https://link.springer.com/article/10.1007/s40747-022-00695-9)
  Cannot find code
  Use a DNN (together with hypersphere) to extract low-level features from high dimensions. (2022)
  As long as these extracted features can represent fewer anomalous instances, it is sufficient to identify anomalies from normal instances.

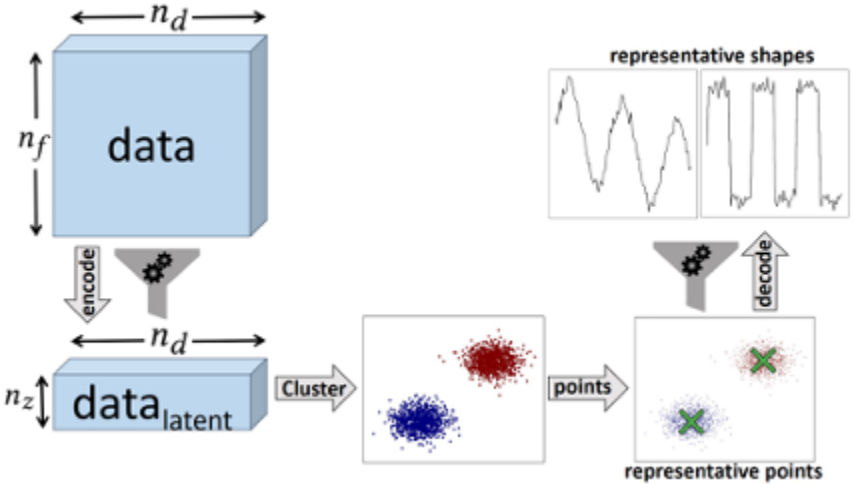


  1. He layers capture the low-dimensional features, where F will contain anomaly and normal features
  2. K will be trained to separate the anomaly features from the normal features. K will send the separated low dimensional features to Hd layers.
  3. The output layer in the decoding module sends out the learned normal and anomaly classes.

  The loss function is a summation of minimizing the error of misclassifying points as anomaly or not and the hypersphere learning a compact space around the target class.

- Alternative approach

  (https://paperswithcode.com/paper/a-flexible-framework-for-anomaly-detection)
  Code is in Python
  The paper (2019) aims to detect anomalies in high dimensions



1. Use encoder to reduce the dimensionality space
2. Perform clustering to get cluster centres
3. Decode to original space (empirically works better)
4. Use distance (Or other metrics) between data points and cluster centres to determine if anomaly or not

Global Outlier Factor (GOF) Scores

Is GOF an alternative of LOF?

Approaches:

1. Statistics (boxplot)
2. Isolation Forest

   The IsolationForest ‘isolates’ observations by randomly selecting a feature and then randomly selecting a split value between the maximum and minimum values of the selected feature.

   Since recursive partitioning can be represented by a tree structure, the number of splittings required to isolate a sample is equivalent to the path length from the root node to the terminating node.

   This path length, averaged over a forest of such random trees, is a measure of normality and our decision function.

Random partitioning produces noticeably shorter paths for anomalies. Hence, when a forest of random trees collectively produce shorter path lengths for particular samples, they are highly likely to be anomalies.

Results

Dimensions	Top N LOF with c = 10, d = 2/10/20, k = 20, n = 900, r = 10	Sklearn LOF with contamination parameter set as 1%	Sklearn Isolation Forest with contamination parameter set as 1%
2	18.8%	18.8%	98.7%
10	14.4%	14.4%	100%
20	71.2%	71.2%	100%

Weak in dealing with local outliers, but good in dealing with global outliers, as seen from the results. Opposite from LOF.