Class Outlier Detection

Zuzana Pekarčíková

Outline

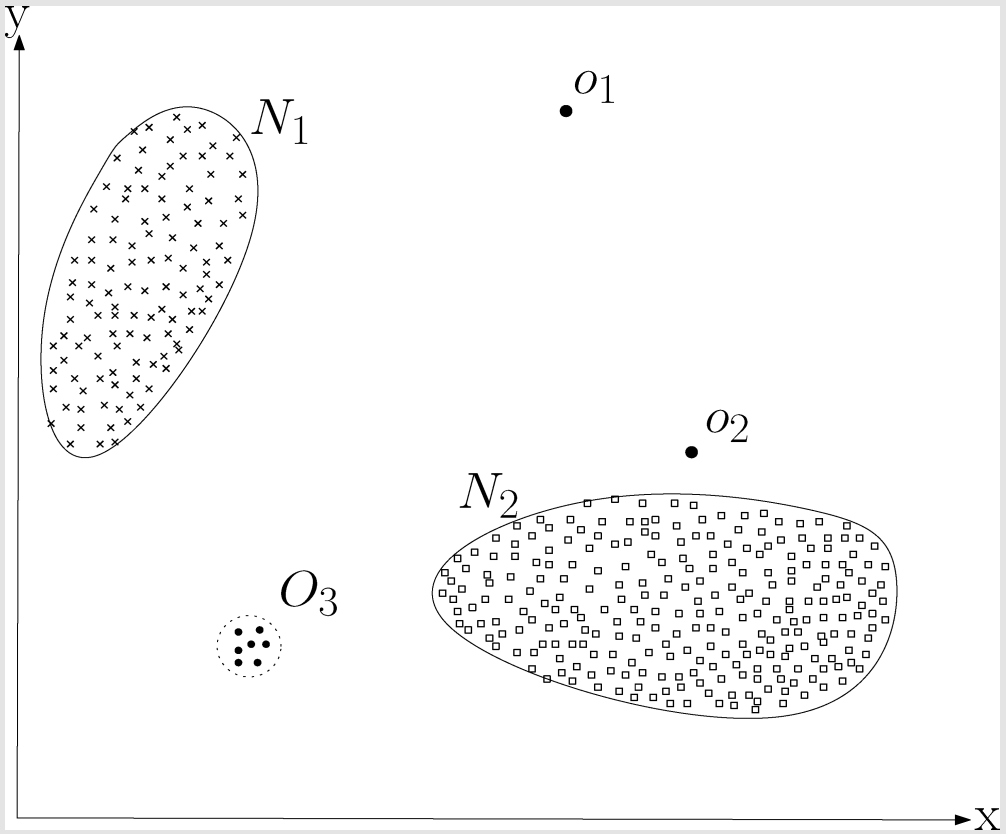
- What is an Outlier ?
- Applications of Outlier Detection
- Types of Outliers
- Outlier Detection Methods Types
- Basic Outlier Detection Methods
- High-dimensional Outlier Detection Methods
- Class Outlier Detection – Random Forests
- Context-based Approach

- Weka Extensions by Frederick Livingston
- Weka CODB Extensions
- Our Implementation
- Our Future Plans

What is an Outlier ?

- Definition of Hawkins [Hawkins 1980]:

"An outlier is an observation which deviates so much from the other observations as t

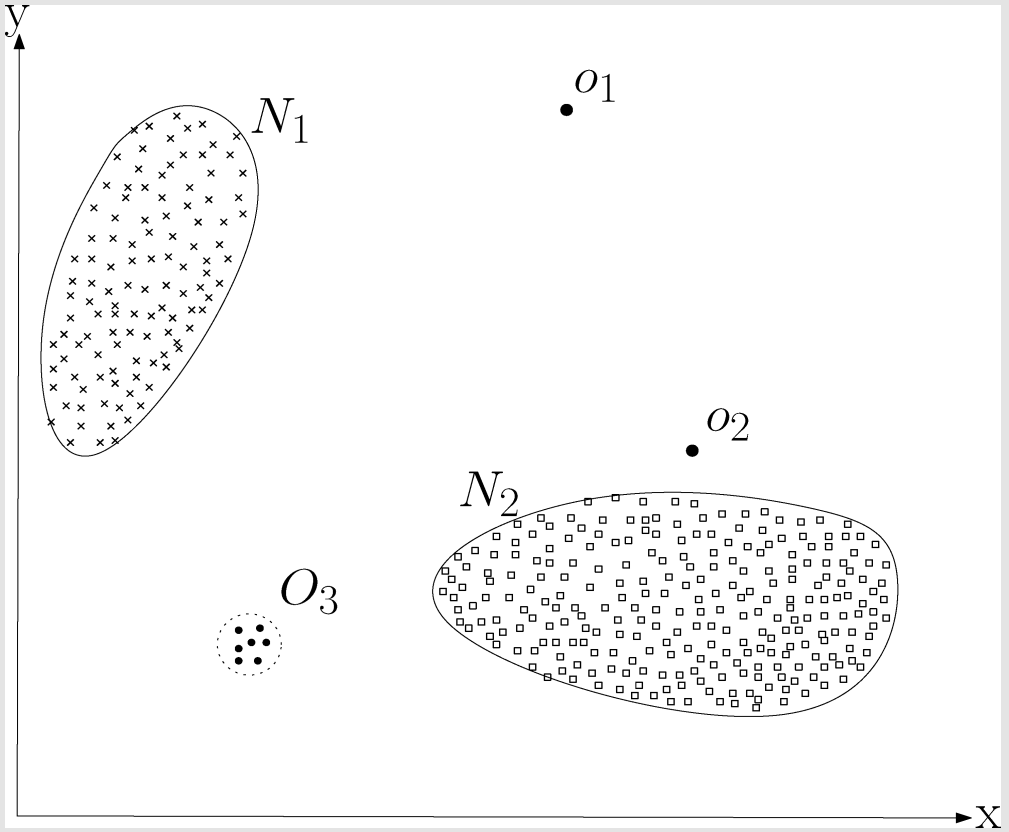


Applications of Outlier Detection

- Fraud detection
  - Purchasing behavior of a credit card owner usually changes when the card is stolen
- Medicine
  - Unusual symptoms or test results may indicate potential health problems of a patient
  - Whether a particular test result is abnormal may depend on other characteristics of the patients
- Detecting measurement errors
  - Data derived from sensors may contain measurement errors
  - Removing such errors can be important in other data mining and data analysis tasks

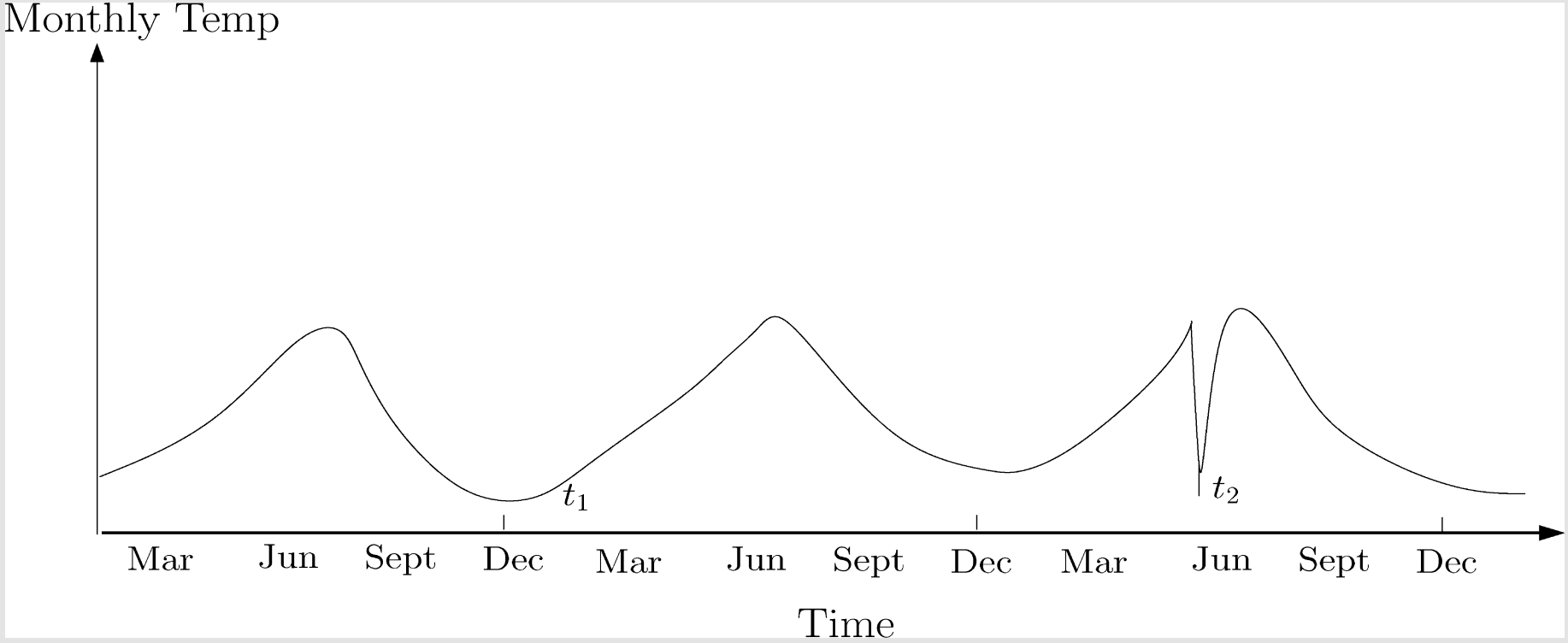
Types of Outliers

- Point Anomalies
- An individual data instance can be considered as anomalous with respect to the rest c
- The simplest type of Outliers.
- Example: credit card fraud detection



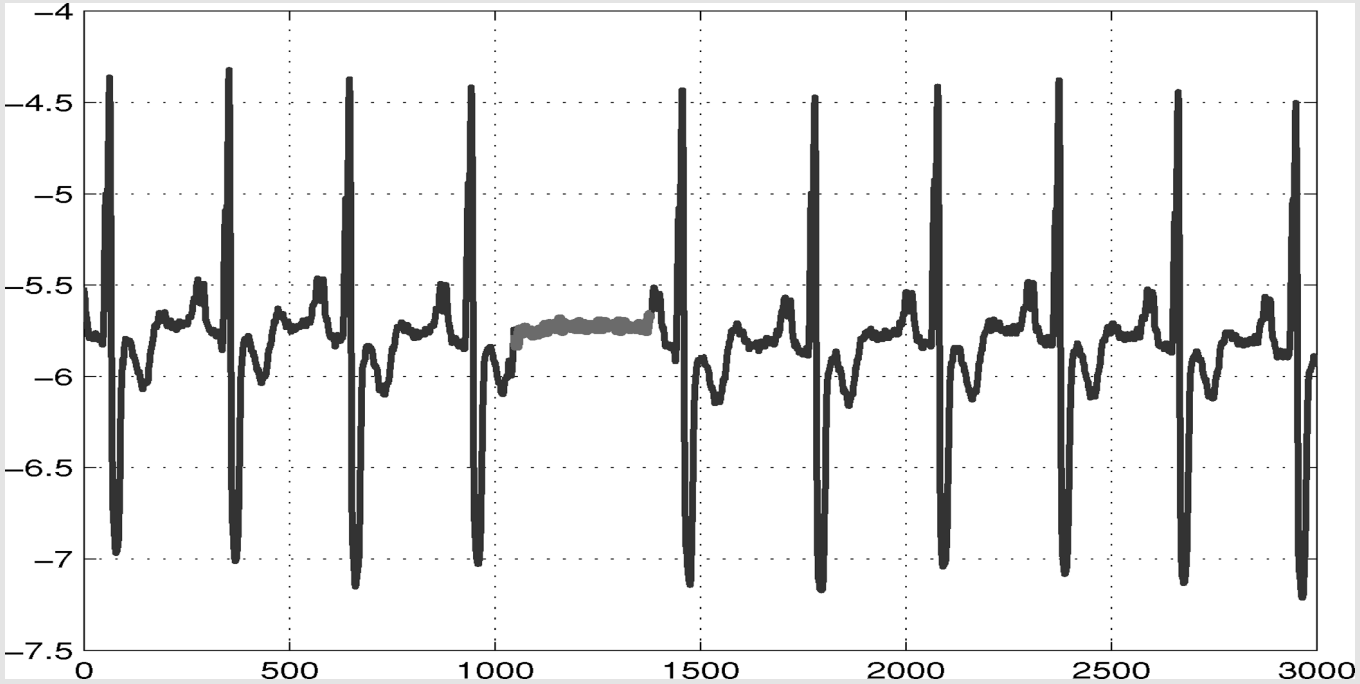
Types of Outliers

- Contextual Anomalies
- If a data instance is anomalous in a specific context, but not otherwise. Example: temp
  - t1 = t2, however t1 is in winter whereas t2 in summer



Types of Outliers

- Collective Anomalies
- A collection of related data instances is anomalous with respect to the entire data set.
- The individual data instances in a collective anomaly may not be anomalies by themse
- Example:
- human
- cardiogram

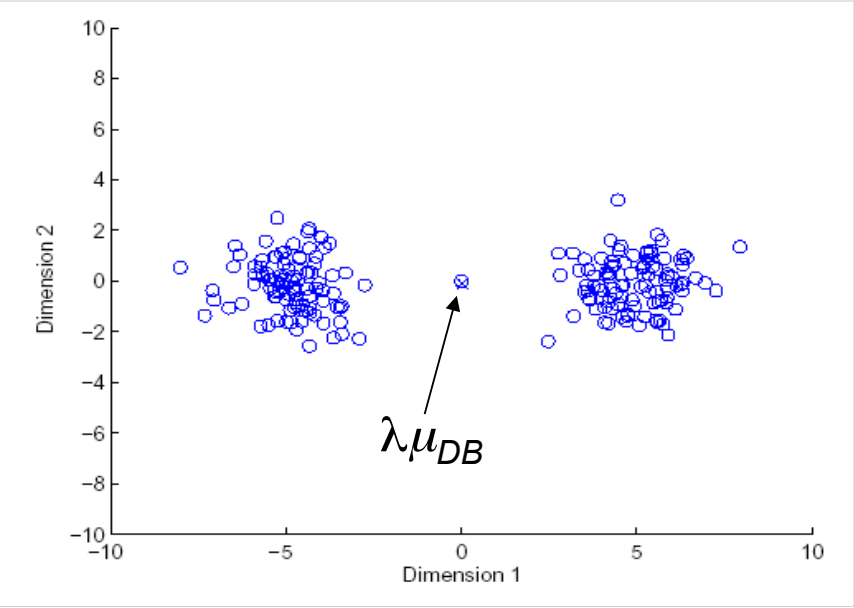


Outlier Detection Methods Types

- The *labels* associated with a data instance denote whether that instance is normal or anomalous
- Supervised Methods
  - availability of a training data set that has labeled instances for normal as well as anomaly classes
  - building a predictive model for normal vs. anomaly classes – problem is transforma
  - Problems:
    - anomalous instances are far fewer than normal instances
    - obtaining acurate labels for the anomaly class is challenging
- Semi-supervised Methods
  - training data has labeled instances only for the normal class
- Unsupervised Methods
  - no labels, most widely used
  - assumption: normal instances are far more frequent than anomalies in the test dat

Outlier Detection Methods

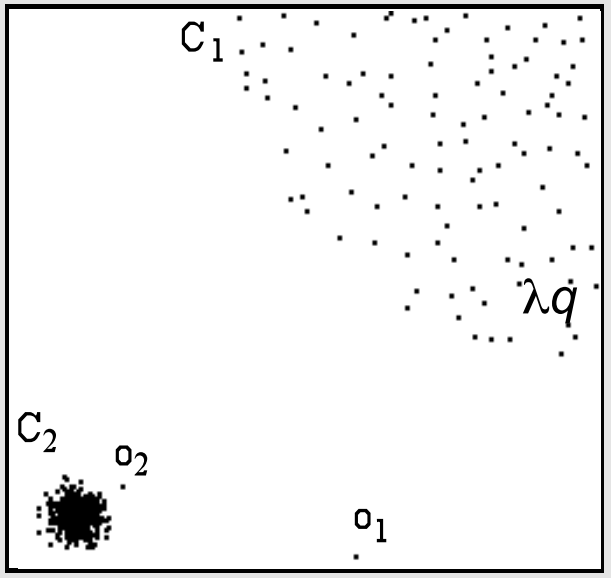
- Statistical Methods
  - normal data objects are generated by a statistical (stochastic) model, and data not fo
  - Example: statistical distribution: Gaussina
    - Outliers are points that have a low probability to be generated by Gaussian distril
  - Problems: Mean and standard deviation are very sensitive to outliers
    - These values are computed for the complete data set (including potential outliers)
  - Advantage: existence of statistical
    - proof why the object is an
    - outlier



Outlier Detection Methods

- Proximity-Based Methods
- An object is an outlier if the proximity of the object to its neighbors significantly deviates fr
  - Distance-based Detection
    - Radius *r*
    - *k* nearest neighbors
  - Density-based Detection
    - Relative density of object counted
    - from density of its neighbors

- Clustering-Based Methods
- Normal data objects belong to large
- and dense clusters, whereas outliers
- belong to small or sparse clusters,
- or do not belong to any clusters.



Outlier Detection Methods

- Classification-Based Methods
  - Main idea: training a classification model that can distinguish normal data from outlier
  - Problem: imbalanced classes
  - Solution: using one-class model – classifier describe only the normal class and samp

Hight-dimensional Outlier Detection Methods

Problems in high-dimensional:

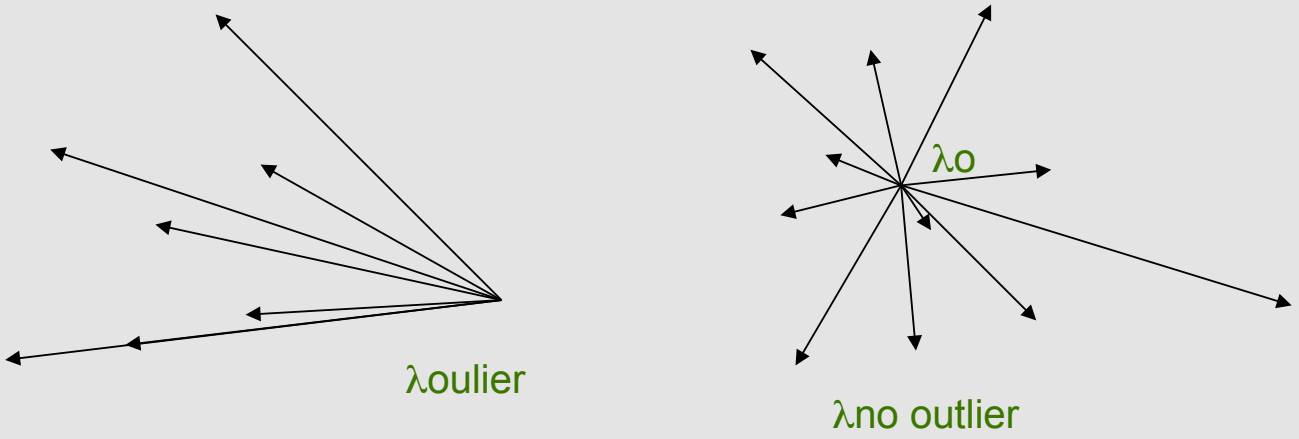
- Relative contrast between distances decreases with increasing dimensionality
- Data are very sparse, almost all points are outliers
- Concept of neighborhood becomes meaningless

Solutions:

- Use more robust distance functions and find full-dimensional outliers
- Find outliers in projections (subspaces) of the original feature space

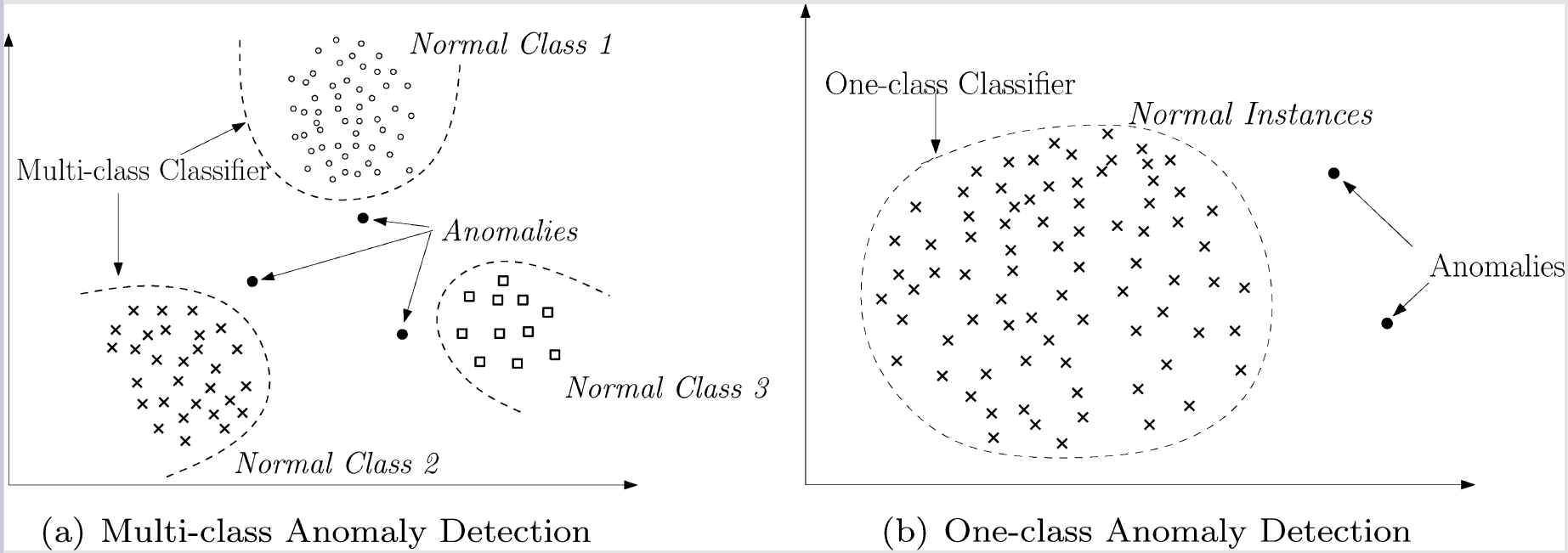
ABOD – angle-based outlier degree

- Object o is an outlier if most other objects are located in similar directions
- Object o is no outlier if many other objects are located in varying directions



Class Outlier Detection

- ‘semantic outlier’
- A semantic outlier is a data point, which behaves differently with other data points in the s



Class Outlier Detection

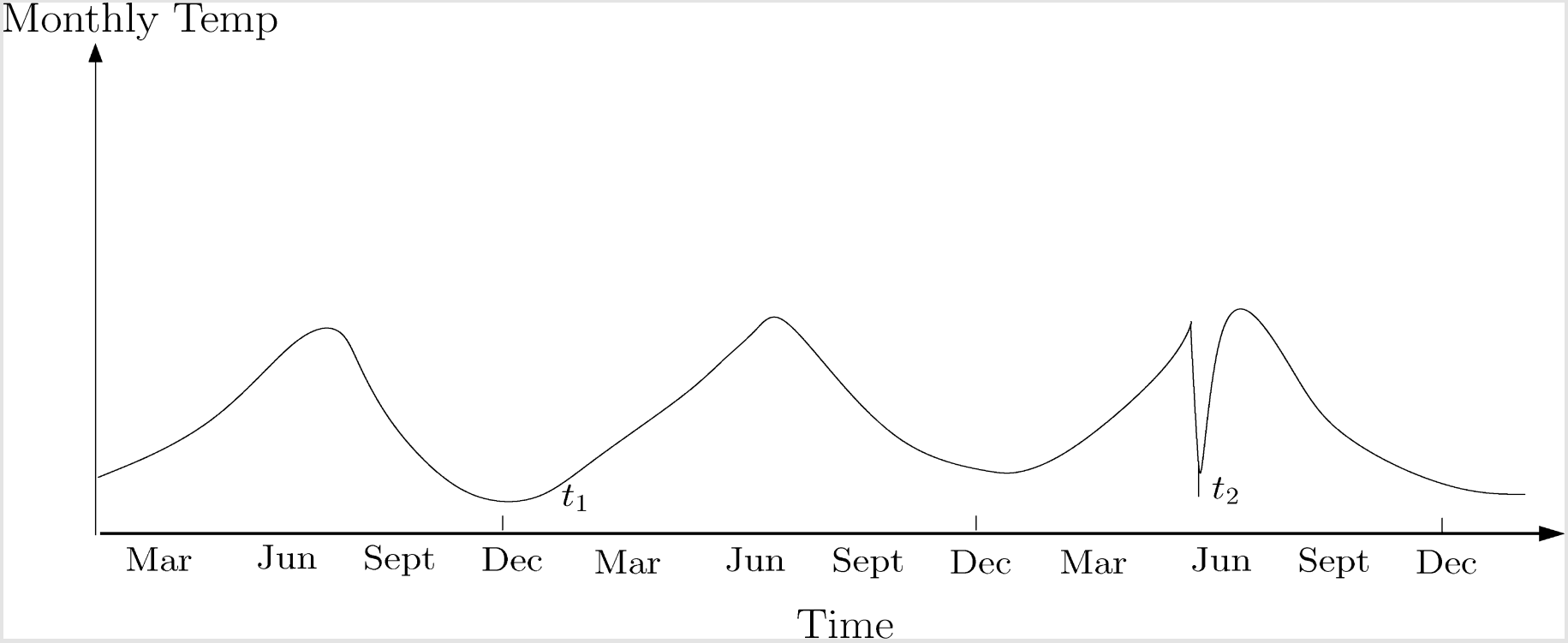
- Multi-class classification based anomaly detection techniques assume that the training da
- Anomaly detection techniques teach a classifier to distinguish between each normal class

Class Outlier Detection – Random Forests

- ***Random Forests*** is an ***enensemble*** classification and regression approach.
- *Ensemble methods* use multiple models to obtain better predictive performance than cou
- Random Forests:
  - consists of many classification trees
  - 1/3 of all samples are left out – ***OOB (out of bag) data*** – for classification error
  - each tree is constructed by a different bootstrap sample from the original data
  - all data are run down the tree and proximities are computed for each pair of cases –
  - Outliers are cases whose proximities to all other cases in the data are generally smal
  - Used in outliers relative to their class – an outlier in class j is a case whose prosimitie

Context-based Approach

Is the temperature 28°C outlier?
If we are in Brno in summer NO
If we are in Brno in winter YES
→ it dependes on the location and time – CONTEXT



Context-based Approach

Contextual outlier significantly deviates from model with respect to a specific contex

Generally the attributes of the data objects are divided into two groups:

- **Contextual attributes**: Define the object's context. In the example, the contextual attributes may
- **Behavioral attributes**: Define the object's characteristics, and are used to evaluate whether the

Contextual outlier detection methods can be devided into two categories according to whe

- Transforming Contextual Outlier Detection to Conventional Outlier Detectio
  - The context can be easily identified.
- Modeling Normal Behavior with Respect to Contexts
  - The context identification is more difficult

Context-based Approach

Transforming Contextual Outlier Detection to Conventional Outlier Detection

General Idea:

- Evaluation wheater the object is an outlier is done in two steps:
  - identifycation the context of the object using the contextual attributes
  - calculation the outlier score for the object in the context using a conventional outlier c

Example:

- In customerrelationship management, we can detect outlier customers in the context of customer g
- 3 attributes:
  - contextual: *age group (25, 25-45, 45-65, and over 65), post code*
  - behavioral: *number of transactions per yer*
- Is customer c outlier?
  - locate the context of *c* using the attributes *age group* and *post code*
  - compare *c* with the other customers in the same group, and use a conventional outlier detection method

Context-based Approach

Modeling Normal Behavior with Respect to Contexts

Context is not easy to identify

Example:

- An online store records the sequence of products seached for by each customer. Outlier behavior i
  - → contexts cannot be easily specified because it is unclear how many products browsed earlie

General idea:

- modelation of normal behaviour with respect to contexts
- With using a training data set a method trains a model that predicts the expected behavior attribute

Is an object outlier?

- We apply the model to the contextual attributes of the object. If the behavior attribute values deviat

Proximities in Random Forests

- It is one of the most important feature when consider Outlier Detection.
- It describes corelations between data elements.
- Proximities Matrix:
  - NxN, where N is the number of data set
  - One for all trees in the forest
  - The cell [8, 10] represents how often the elements 8 and 10 share together one node
  - The outlier score is computed as the average proximity value for each element. If this

Proximities in Random Forests - Example

- Four elements
- Five Random Forest Trees
- Table in the left is ordinary proximities matrix
- Table in the right is normalized by number of trees
- Element C is outlier because it has the minimum sharing notes with the other elements

	A	B	C	D
A	5	3	1	4
B	3	5	1	3
C	1	1	5	4
D	4	3	2	5

	A	B	C	D
A	1	0,6	0,2	0,8
B	0,6	1	0,2	0,6
C	0,2	0,2	1	0,4
D	0,8	0,6	0,4	1

Weka Extensions by Fred Livingston

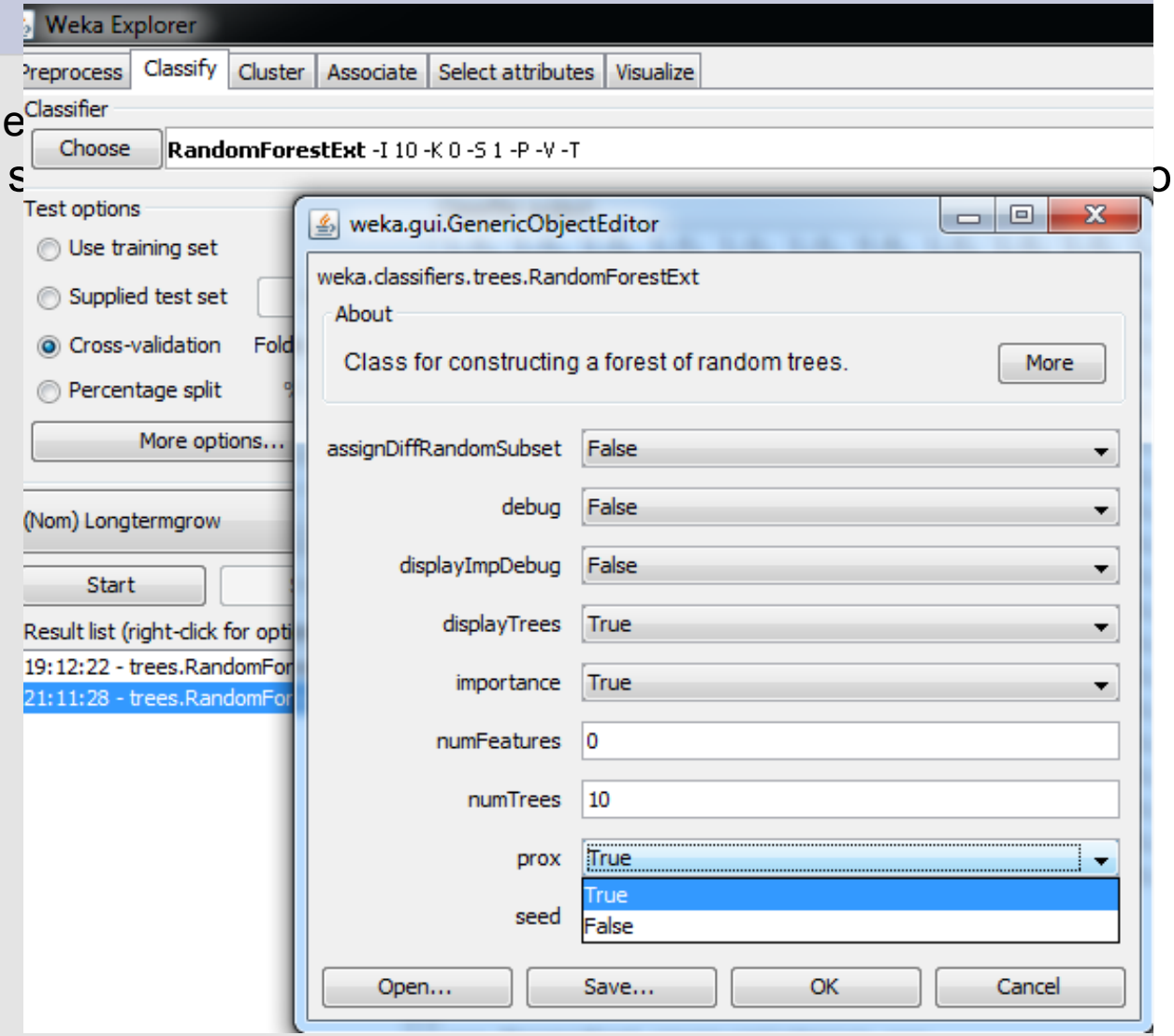
Implementation of Proximities

Extensional classes:
- *weka.classifiers.meta.BaggingExt*
- *weka.trees.RandomForestExt*
- implement the proximities matrix

The version which was used has implemented property *prox_matrix and its* computing *as well,* how

Our Implementation I

- GUI option for computing proximitie
  - It is possible to turn on the s



Our Implementation II

After computation of classificatio

Classifier output

```
Proximities Matrix
====================
1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.8, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.6, 1.0,
1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.8, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.6, 1.0,
1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.8, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.6, 1.0,
1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.8, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.6, 1.0,
1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.8, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.6, 1.0,
1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.8, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.6, 1.0,
0.8, 0.8, 0.8, 0.8, 0.8, 0.8, 1.0, 0.8, 0.8, 0.8, 0.8, 0.8, 0.8, 0.6, 0.8,
1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.8, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.6, 1.0,
1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.8, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.6, 1.0,
1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.8, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.6, 1.0,
1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.8, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.6, 1.0,
1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.8, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.6, 1.0,
1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.8, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.6, 1.0,
0.6, 0.6, 0.6, 0.6, 0.6, 0.6, 0.6, 0.6, 0.6, 0.6, 0.6, 0.6, 0.6, 1.0, 0.6,
1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.8, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.6, 1.0,
0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.4, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.6, 0.2,
0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.7, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.3, 0.5,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.2, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.4, 0.0,
0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.3, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.3, 0.1,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.2, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.4, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.2, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.4, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.2, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.4, 0.0,
```

Our Implementation III

- Ensemble Outlier Detection:
  - *OutlierEnsembleEngine*
    - Is the main class whitch coordinate the whole computation.
    - By the constructor all needed properties are loaded.
      - connector, data, subDataNum, odmArray, odmWeights, odmWeights, randAttrNum
    - Method *compute()*
      - for each data element and for each method computes the outlier score
      - for each data element then it is combined (with using connector) the re
      - returns the array of result ensemble outlier score for each data elemen

Our Implementation IV

OutlierEnsembleEngine pseudocode:

```
public Double[] compute() {
 foreach (Method  m  IN  MethodArray) {
        Element[] randomSubData = getRandomSubData();
        OutliersScoreByMethods[] = m.computeOutliersScores(randomSubData);
    }
    outliersScoreResult = connector.compute(OutliersScoreByMethods, MethodWeig

    return outliersScoreResult;
 }
```

Our Implementation V

- *OutlierEnsembleConnector*
  - Abstract class with abstract method *compute()*
  - Each class, which extends this one, computes the result outlier score for data element wit
  - *OutlierEnsembleConnectorAddition* class
    - extends *OutlierEnsembleConnector*
    - it uses as combination function the addition

Our Implementation VI

- *OutlierDetectionMethod* class
  - Abstract class
  - Each class extending this one take care of computing the outlier score for data elements.
  - *OutlierDetectionMethodUsingRandomForest*
    - Extends *OutlierDetectionMethod*
    - Uses the RandomTree class for computation outlier scores
    - Not implemented yet

Weka CODB Extensions

- Class Outlier Detection
- Distance based approach
- It is based on the Class Outlier Factor (*COF*) whitch represents the degree
- *COF* computes as a deviation of the instance from the instances of the sa

Weka CODB Extensions II

- *PCL(T, K) is the Probability of the class label of the instance T with respect to the class la*
- Deviation(T) is how much the instance T deviates from instances of the same class
- *α* and *β* are factors to control the importance and the effects of *Deviation(T)* and *KDist(T)*
- *KDist(T)* is the summation of distance between the instance *T* and its *K* nearest neighbor

$$COF(T) = K * PCL(T,K) + \alpha * \frac{1}{Dev(T)} + \beta * KDist(T)$$