A comparative evaluation of outlier detection algorithms: experiments and analyses

Rémi Domingues[a], Maurizio Filippone[a], Pietro Michiardi[a], Jihane Zouaoui[b]

[a]*Department of Data Science, EURECOM, Sophia Antipolis, France*
[b]*Amadeus, Sophia Antipolis, France*

Abstract

We survey unsupervised machine learning algorithms in the context of outlier detection. This task challenges state-of-the-art methods from a variety of research fields to applications including fraud detection, intrusion detection, medical diagnoses and data cleaning. The selected methods are benchmarked on publicly available datasets and novel industrial datasets. Each method is then submitted to extensive scalability, memory consumption and robustness tests in order to build a full overview of the algorithms' characteristics.

Keywords: Outlier detection; fraud detection; novelty detection; variational inference; isolation forest

1. Introduction

Numerous applications nowadays require thorough data analyses to filter out outliers and ensure system reliability. Such techniques are especially useful for fraud detection where malicious attempts often differ from most nominal cases and can thus be prevented by identifying outlying data. Those *anomalies* can be defined as observations which deviate sufficiently from most observations to consider that they were generated by a different generative process. These observations are called *outliers* when their number is significantly smaller than the proportion of nominal cases, typically lower than 5%. Outlier detection techniques have proven to be efficient for applications such as network intrusions, credit card fraud detection and telecommunications fraud detection [10].

Excluding outliers from a dataset is also a task from which most data mining algorithms can benefit. An outlier-free dataset allows for accurate modelling tasks, making outlier detection methods extremely valuable for data cleaning [18]. These techniques have also been successfully applied to fault detection on critical systems and result in improved damage control and component failure prediction [31]. Medical diagnoses may also profit from the identification of outliers in application such as brain tumor detection [24] and cancerous masses recognition in mammograms [28].

Numerous machine learning methods are suitable for anomaly detection. However, supervised algorithms are more constraining than unsupervised methods as they need to be provided with a labeled dataset. This requirement is particularly expensive when the labeling must be performed by humans. Dealing with a heavily imbalanced class distribution, which is inherent to anomaly detection, can also affect the efficiency of supervised algorithms [12]. This paper focuses on unsupervised machine learning algorithms to isolate outliers from nominal samples. In previous reviews, such methods were shown to be proficient for outlier and novelty detection [23, 33]. Unsupervised algorithms make use of unlabeled data to assign a score to each sample, representing the observation normality. Binary segmentations can be further made by thresholding the scores.

Outlier detection is a notoriously hard task: detecting anomalies can be difficult when overlapping with nominal clusters, and these clusters should be dense enough to build a reliable model. The problem of contamination, i.e. using an input dataset contaminated by outliers, makes this task even trickier as anomalies may degrade the final model if the training algorithm lacks robustness. These issues, which are present in many real-world datasets, are not always addressed in works describing new unsupervised methods, as these algorithms may target a different use case. These reasons motivate the need for a thorough benchmark bringing together distinctive techniques on complex datasets.

Our paper extends previous works [7, 33] by using 12 publicly available labeled datasets, most of which are recommended for outlier detection in [7], in addition to 3 novel industrial datasets from the production environments of a major company in the travel industry. The benchmark made in [7] used fewer datasets and solely numerical features while we benchmark and address ways to handle categorical data. While the previous works perform outlier detection on training data, our study test for the generalization ability of all methods by predicting outliers on unseen testing data. The selected parametric and non-parametric algorithms come from a variety of approaches including probabilistic algorithms, nearest-neighbor based methods, neural networks, information theoretic and isolation methods. The performance on labeled datasets are compared with the area under the ROC and precision-recall curves.

In order to give a full overview of these methods, we also benchmark the training time, prediction time, memory usage and robustness of each method when increasing the number of samples, features and the background noise. These scalability measurements allow us to compare algorithms not only based on their outlier detection performance but also on their scalability, robustness and suitability for large dimensional problems.

The paper is organized as follows: section 2 introduces the research fields and methods targeted by the benchmark; section 3 details the experimental setup, the public and proprietary datasets used and the method implemented to generate synthetic datasets; section 4 presents the results of our benchmark and section 5 summarizes our conclusions.

2. Outlier detection methods

The algorithms described in this section belong to a wide range of approaches. These methods build a model representing the nominal classes, i.e. dense clusters of similar data points, during a training phase. Online or batch predictions can thereafter be applied to new data based on the trained model to assign an anomaly score to the new observations. Applying a threshold on the returned scores provides a decision boundary separating nominal samples from outliers.

The evaluation presented in this study contains both parametric and nonparametric machine learning algorithms. While parametric approaches model the underlying data with a fixed number of parameters, the number of parameters of nonparametric methods is potentially infinite and can increase with the complexity of data. If the former are often computationally faster, they require assumptions about the data distribution, e.g. the number of clusters, and may result in a flawed model if based on erroneous assumptions. The latter make fewer assumptions about the data distribution and may thus generalize better while requiring less knowledge about the data.

2.1. Probabilistic methods

Probabilistic algorithms estimate the probability density function of a dataset X, by inferring the model parameters θ. Data points having the smallest likelihood $P(X|\theta)$ are identified as outliers. Most probabilistic methods described in this section can be trained incrementally, i.e. presenting new input data to an existing model will cause the model to adapt to the new data while remembering past ones.

The first algorithm used in our benchmark is the **Gaussian Mixture Model (GMM)**, which fits a given number of Gaussian distributions to a dataset. The model is trained using the Expectation-Maximization (EM) algorithm [6] which maximizes a lower bound of the likelihood iteratively. This method has been successfully applied to identify suspicious and possibly cancerous masses in mammograms by novelty detection in [28]. However, assessing the number of components of the mixture by data exploration can be complex and motivates the use of nonparametric alternatives described hereafter.

In [3], Blei et al. describe the **Dirichlet Process Mixture Model (DPMM)**, a nonparametric Bayesian algorithm which optimizes the model parameters and tests for convergence by monitoring a non-decreasing lower bound on the log-marginal likelihood. The result is a mixture model where each component is a product of exponential-family distributions. Most likelihoods, e.g. Gaussian or Categorical, can be written in exponential-family form; in this formulation, it is possible to obtain suitable conjugate priors to facilitate inference.

The number of components is inferred during the training and requires an upper bound K. Weights π_i are represented by a Dirichlet Process modelled as a truncated stick-breaking process (equation 1). The variable v_i follows a Beta distribution, where α_k and β_k are variational parameters optimized during the training for each component.

$$\pi_i(\boldsymbol{v}) = v_i \prod_{j=1}^{i-1}(1 - v_j) \qquad q_{\boldsymbol{\alpha},\boldsymbol{\beta}}(\boldsymbol{v}) = \prod_{k=1}^{K-1} Beta(\alpha_k, \beta_k) \tag{1}$$

The training optimizes the parameters of the base distributions, e.g. the parameters of a Gaussian-Wishart for a multivariate Gaussian likelihood. The scoring is then made by averaging the log likelihood computed from each mixture of likelihoods sampled from the conjugate priors. The algorithm is applied to intrusion detection on the KDD 99 dataset in [8] and outperforms SVM and KNN algorithms. The cluster centroids provided by the model can also be valuable to an end-user as they represent the average nominal data points.

Kernel density estimators (KDE), also called *Parzen windows* estimators, approximate the density function of a dataset by assigning a kernel function to each data point then summing the local contributions of the kernels. A bandwidth parameter acts as a smoothing parameter on the density shape and can be estimated by methods such as Least-Squares Cross-Validation (LSCV). As shown in [28], KDE methods are efficient when applied to novelty detection problems. However, these approaches are sensitive to outliers and struggle in finding a good estimate of the nominal data density in datasets contaminated by outliers. This issue is shown by Kim et al. in [13] where the authors describe a **Robust Kernel Density Estimator (RKDE)**, algorithm which uses M-estimation methods, such as robust loss functions, to overcome the limitations of plain KDE.

Probabilistic principal component analysis (PPCA) [29] is a latent variable model which estimates the principal components of the data. It allows for the projection of a d-dimensional observation vector Y to a k-dimensional vector of latent variables X, with k the number of components of the model. The relationship $\boldsymbol{Y} = \boldsymbol{W}\boldsymbol{X} + \boldsymbol{\mu} + \boldsymbol{\epsilon}$ is trained by expectation-maximization. The authors suggest to use the log-likelihood as a degree of novelty for new data points.

More recently, **Least-squares anomaly detection (LSA)** [25] developed by Quinn et al. extends a multi-class probabilistic classifier to a one-class problem. The approach is compared against KNN and One-class SVM using the area under the ROC curve.

2.2. Distance-based methods

This class of methods uses solely the distance space to flag outliers. As such, the **Mahalanobis distance** is suitable for anomaly detection tasks targeting multivariate datasets composed of a single Gaussian-shaped cluster [2]. The model parameters are the mean and inverse covariance matrix of the data, thus similar to a one-component GMM with a full covariance matrix.

2.3. Neighbor-based methods

Neighbor-based methods study the neighborhood of each data point to identify outliers. **Local outlier factor (LOF)** described in [4] is a well-known distance based approach corresponding to this description. For a given data point x, LOF computes its *degree* $d_k(x)$ of being an outlier based on the Euclidean distance d between x and its k^{th} closest neighbor n_k, which gives $d_k(x) = d(x, n_k)$. The scoring of x also takes into account for each of its neighbors n_i, the maximum between $d_k(n_i)$ and $d(x, n_i)$. As shown in [7], LOF outperforms Angle-Based Outlier Detection [16] and One-class SVM [26] when applied on real-world datasets for outlier detection, which makes it a good candidate for this benchmark.

Angle-Based Outlier Detection (ABOD) [16] uses the radius and variance of angles measured at each input vector instead of distances to identify outliers. The motivation is here to remain efficient in high-dimensional space and to be less sensible to the curse of dimensionality. Given an input point x, ABOD samples several pairs of points and computes the corresponding angles at x and their variance. Broad angles imply that x is located inside a major cluster as it is surrounded by many data points, while small angles denote that x is positioned far from most points in the dataset. Similarly, a higher variance will be observed for points inside or at the border of a cluster than for outliers. The authors show that their method outperforms LOF on synthetic datasets containing more than 50 features. According to the authors, the pairs of vectors can be built from the entire dataset, a random subset or the k-nearest neighbors in order to speed up the computation at the cost of lower outlier detection performance.

The **Subspace outlier detection (SOD)** [15] algorithm finds for each point p the set of m neighbors shared between p and its k-nearest neighbors. The outlier score is then the standard deviation of p from the mean of a given subspace, which is composed of a subset of dimensions. The attributes having a small variance for the set of m points are selected to be part of the subspace.

2.4. Information theory

The **Kullback-Leibler (KL) divergence** was used as an information-theoretic measure for novelty detection in [9]. The method first trains a Gaussian mixture model on a training set, then estimates the *information content* of new data points by measuring the KL divergence between the estimated density and the density estimated on the training set and the new point. This reduces to an F-test in the case of a single Gaussian.

2.5. Neural networks

In [19], Marsland et al. propose a reconstruction-based nonparametric neural network called **Grow When Required (GWR) network**. This method is based on Kohonen networks, also called Self-Organizing maps (SOM) [14], and fits a graph of adaptive topology lying in the input space to a dataset. While training the network, nodes and edges are added or removed in order to best fit the data, the objective being to end up with nodes positioned in all dense data regions while edges propagate the displacement of neighboring nodes.

Outliers from artificial datasets are detected using fixed-topology SOM in an experimental work [21]. The paper uses two thresholds t_1 and t_2 to identify data points having their closest node further than t_1, or projecting on an outlying node, i.e. a neuron having a median interneuron distance (MID) higher than t_2. The MID of each neuron is computed by taking the median of the distance between a neuron and its 8 neighbors in a network following a 2-dimensional grid topology. Severe outliers and dense clouds of outliers are correctly identified with this technique, though some nominal samples can be mistaken as mild outliers.

2.6. Domain-based methods

Additional methods for outlying data identification rely on the construction of a boundary separating the nominal data from the rest of the input space, thus estimating the domain of the nominal class. Any data point falling outside of the delimited boundary is thus flagged as outlier.

One-class SVM [26], an application of support vector machine (SVM) algorithms to one-class problems, belongs to this class of algorithms. The method computes a separating hyperplane in a high dimensional space induced by kernels performing dot products between points from the input space in high-dimensional space. The boundary is fitted to the input data by maximizing the margin between the data and the origin in the high-dimensional space. The algorithm prevents overfitting by allowing a percentage υ of data points to fall outside the boundary. This percentage υ acts as regularization parameter; it is used as a lower bound on the fraction of support vectors delimiting the boundary and as an upper bound on the fraction of margin errors, i.e. training points remaining outside the boundary.

The experiment of the original paper targets mostly novelty detection, i.e. anomaly detection using a model trained on a dataset free of anomalies. Our benchmark uses contaminated datasets to assess the algorithm robustness with a regularization parameter significantly higher than the expected proportion of outliers.

2.7. Isolation methods

We include an isolation algorithm in this study, which focuses on separating outliers from the rest of the data points. This method differs from the previous methods as it isolates anomalies instead of profiling normal points.

The concept of **Isolation forest** was brought by Liu in [17] and uses random forests to compute an isolation score for each data point. The model is built by performing recursive random splits on attribute values, hence generating trees able to isolate any data point from the rest of the data. The score of a point is then the average path length from the root of the tree to the node containing the single point, a short path denoting a point easy to isolate due to attribute values significantly different from nominal values.

The author states that his algorithm provides linear time complexity and demonstrates outlier detection performance significantly better than LOF on real-world datasets.

3. Experimental evaluation

3.1. Metrics

We evaluate the performance of the outlier detection algorithms by comparing two metrics based on real-world labeled datasets detailed in section 3.2. For this purpose, we use the receiver operating characteristic (ROC) curve (true positive rate against false positive rate) and the precision-recall (PR) curve (equations 2 and 3), where the positive class represents the anomalies and the negative class represents the nominal samples. The comparison is then based on the area under the curve (AUC) of both metrics, respectively the ROC AUC and the average precision (AP).

$$precision = \frac{true\ positives}{true\ positives + false\ positives} \tag{2}$$

$$recall = \frac{true\ positives}{true\ positives + false\ negatives} \tag{3}$$

3.2. Datasets

Our evaluation uses 15 datasets ranging from 723 to 20,000 samples and containing from 6 to 107 features. Of those datasets, 12 are publicly available on the UCI [1] or OpenML [30] repositories while the 3 remaining datasets are novel proprietary datasets containing production data from the company Amadeus. Table 1 gives an overview of the datasets characteristics. Our study assesses if the models are able to generalize to future datasets, which is a novel approach in outlier detection works. This requires that algorithms support unseen testing data, and is achieved by performing a Monte Carlo cross validation of 5 iterations, using 80% of the data for the training phase and 20% for the prediction. Training and testing datasets contain the same proportion of outliers, and ROC AUC and AP are measured based on the predictions made. For 7 of the publicly available datasets, the outlier classes are selected according to the recommendations made in [7], which are based on extensive datasets comparisons. However, the cited experiment discards all categorical data, while we keep those features and performed one-hot encoding to binarize them, keeping

Table 1: UCI, OpenML and proprietary datasets benchmarked - (# categ. dims) is the average number of binarized features obtained after transformation of the categoricals

Dataset	**Nominal class**	**Anomaly class**	**Numeric dims**	**Categ. dims**	**Samples**	**Anomalies**
ABALONE	8, 9, 10	3, 21	7	1 (3)	1,920	29 (1.51%)
ANN-THYROID	3	1	21	0 (0)	3,251	73 (2.25%)
car	unacc, acc, good	vgood	0	6 (21)	1,728	65 (3.76%)
COVTYPE	2	4	54	0 (0)	10,000[1]	95 (0.95%)
GERMAN-SUB	1	2	7	13 (54)	723	23 (3.18%)[2]
KDD-SUB	normal	u2r, probe	34	7 (42)	10,000[1]	385 (3.85%)
MAGIC-GAMMA-SUB	g	h	10	0 (0)	12,332	408 (3.20%)[2]
MAMMOGRAPHY	-1	1	6	0 (0)	11,183	260 (2.32%)
MUSHROOM-SUB	e	p	0	22 (107)	4,368	139 (3.20%)[2]
SHUTTLE	1	2, 3, 5, 6, 7	9	0 (0)	12,345	867 (7.02%)
WINE-QUALITY	4, 5, 6, 7, 8	3, 9	11	0 (0)	4,898	25 (0.51%)
YEAST [3]	CYT, NUC, MIT	ERL, POX, VAC	8	0 (0)	1,191	55 (4.62%)
PNR	0	1, 2, 3, 4, 5	82	0 (0)	20,000	121 (0.61%)
SHARED-ACCESS	0	1	49	0 (0)	18,722	37 (0.20%)
TRANSACTIONS	0	1	41	1 (9)	10,000[1]	21 (0.21%)

[1] Subsets of the original datasets are used, with the same proportion of outliers.
[2] Anomalies are sampled from the corresponding class, using the average percentage of outliers depicted in [7].
[3] The first feature corresponding to the protein name was discarded.

all information from the dataset at the cost of a higher dimensionality. Normalization is further achieved by centering numerical features to the mean and scaling them to unit variance.

The three following datasets contain production data collected by Amadeus, a Global Distribution System (GDS) providing online platforms to connect the travel industry. This company manages almost half of the flight bookings worldwide and is targeted by fraud attempts leading to revenue losses and indemnifications. The datasets do not contain information traceable to any specific individuals.

The PASSENGER NAME RECORDS (PNR) dataset contains booking records, mostly flight and train bookings, containing 5 types of frauds labeled by fraud experts. The features in this dataset describe the most important changes applied to a booking from its creation to its deletion. Features include time-based information, e.g. age of a PNR, percentage of cancelled flight segments or passengers, and means and standard deviations of counters, e.g. number of passenger modifications, frequent traveller cards, special service requests (additional luggage, special seat or meal), or forms of payment.

The TRANSACTIONS dataset is extracted from a Web application used to perform bookings. It focuses on user sessions which are compared to identify bots and malicious users. Examples of features are the number of distinct IPs and organizational offices used by a user, the session duration and means and standard deviations applied to the number of bookings and number of actions. The most critical actions are also monitored.

The SHARED-ACCESS dataset was generated by a backend application used to manage shared rights between different entities. It enables an entity to grant specific reading (e.g. booking retrieval, seat map display) or writing (e.g. cruise distribution) rights to another entity. Monitoring the actions made with this application should prevent data leaks and sensible right transfers. For each user account, features include the average number of actions performed per session and time unit, the average and standard deviation for some critical actions per session, and the targeted rights modified.

3.3. Datasets for scalability tests

As the choice of an outlier detection algorithm may not only be limited to its accuracy but is often subject to computational constraints, our experiment includes training time, prediction time, memory usage and noise resistance (through precision-recall measurements) of each algorithm on synthetic datasets.

For these scalability tests, we generate datasets of different sizes containing a fixed proportion of background noise. The datasets range from 10 samples to 10 million samples and from 2 to 10,000 numerical features. We also keep the number of features and samples fixed while increasing the proportion of background noise from 0% to 90% to perform robustness measurements. The experiment is repeated 5 times, using the same dataset for training and testing. We allow up to 24 hours for training or prediction steps to be completed and stop the computation after this period of time. Experiments reaching a timeout or requiring more than 256 GB RAM do not report memory usage nor robustness in section 4.

In order to avoid advantaging some algorithms over others, the datasets are generated using two Student's T distributions. The distributions are respectively centered in $(0, 0..., 0)$ and $(5, 5..., 5)$, while the covariance matrices are computed using $c_{ij} = \rho^{|i-j|}$ where c_{ij} is entry (i, j) of the matrix. The parameter ρ follows a uniform distribution $\rho \sim U(0, 1)$ and the degrees of freedom parameter follows a Gamma distribution $\upsilon \sim \Gamma(1, 5)$. We then add outliers uniformly sampled from a hypercube 7 times bigger than the standard deviation of the nominal data.

3.4. Algorithms implementations and parameters

Most implementations used in this experiment are publicly available. Table 2 details the programming languages and initialization parameters selected. A majority of methods have flexible parameters and perform very well without an extensive tuning. The **Matlab Engine API for Python** and the **rpy2** library allow us to call Matlab and R code from Python.

DPGMM is our own implementation of a Dirichlet Process Mixture Model and follows the guidelines given in [3] where we place a Gamma prior on the scaling parameter β. Making our own implementation of this algorithm is motivated by its capability of handling a wide range of probability distributions, including categorical distributions. We thus benchmark DPGMM, which uses only Gaussian distributions to model data and thus uses continuous and binarized features as all other algorithms, and DPMM which uses a mixture of multivariate Gaussian / Categorical distributions, hence requiring fewer data transformations and working on a smaller number of features. This algorithm is the only one capable of using the raw string features from the datasets.

Note that DPGMM and DPMM converge to the same results when applied to non-categorical data, and that our DPGMM performs similarly to the corresponding scikit-learn implementation called **Bayesian Gaussian Mixture (BGM)**.

Table 2: Implementations and parameters selected for the evaluation

Algorithm	**Language**	**Parameters**
GMM [1]	Python	$components = 1$
DPGMM	Python	$\boldsymbol{\Gamma_\theta} = (\alpha = 1, \beta = 0)), k_{max} = 10$
DPMM		$\boldsymbol{\mu_\theta} = mean(data), \boldsymbol{\Sigma_\theta} = var(data)$
RKDE [23]	Matlab	$bandwidth = LKCV, loss = Huber$
PPCA	Python	$components = mle, svd = full$
LSA	Python	$\sigma = 1.7, \rho = 100$
MAHA	Python	n/a
LOF	Python	$k = \max(n * 0.1, 50)$
ABOD [2]	R	$k = \max(n * 0.01, 50)$
SOD [2]	R	$k = \max(n * 0.05, 50), k_{shared} = \frac{k}{2}$
KL [12]	R	$components = 1$
GWR [2]	Matlab	$it = 15, t_{hab} = 0.1, t_{insert} = 0.7$
OCSVM	Python	$\nu = 0.5$
IFOREST	Python	$contamination = 0.5$

[1] Parameter tuning is required to maximize the mean average precision (MAP).
[2] We extend these algorithms to add support for predictions on unseen data points.
[3] We use Scott's rule-of-thumb $h = n^{-1/(d+4)}$ [27] to estimate the bandwidth when it cannot be computed due to a high number of features.

However, we did not optimize our implementation which uses a more general exponential-family representation for probability distributions. This greatly increases the computational cost and results in a much higher training and prediction time.

GWR has a nonparametric topology. Identifying outlying neurons as described in [21] to detect outliers from a 2D grid topology may thus not be applicable to the present algorithm. The node connectivity can indeed differ significantly from one node to another, and we need a ranking of the outliers for our performance measurements more than a two-parameter binary classification. Therefore, the score assigned to each observation is here the squared distance between an observation and the closest node in the network. Note that regions of outliers sufficiently dense to attract neurons may not be detected with this technique.

4. Results

For each dataset, the methods described in section 2 are applied to the 5 training and testing subsets sampled by Monte Carlo cross validation. We report here the average and standard deviation over the runs. The programming language and optimizations applied to the implementations may affect the training time, prediction time and memory usage measured in sections 4.3 and 4.4. For this reason, our analysis focuses more on the curves slope and the algorithms complexity than on the measured values. The experiments are performed on a VMware virtual platform running Ubuntu 14.04 LTS and powered by an Intel Xeon E5-4627 v4 CPU (10 cores at 2.6GHz) and 256GB RAM. We use the Intel distribution of Python 3.5.2, R 3.3.2 and Matlab R2016b.

Table 3: Rank aggregation through Cross-Entropy Monte Carlo

Algorithm	GMM	DPGMM	DPMM	RKDE	PPCA	LSA	MAHA	LOF	ABOD	SOD	KL	GWR	OCSVM	IFOREST
PR AUC	6	12	9	2	5	14	7	11	10	4	8	13	3	1
ROC AUC	11	4	5	1	7	9	6	10	13	8	12	14	3	2

4.1. ROC and precision-recall

The area under the ROC curve is estimated using trapezoidal rule while the area under the precision-recall curve is computed by average precision (AP). Figure 2 shows the mean and standard deviation AP per algorithm and dataset while figure 3 reports the ROC AUC. For the clarity of presentation, figure 1 shows the global average and standard deviation average of both metrics, sorting algorithms by decreasing mean average precision (MAP).

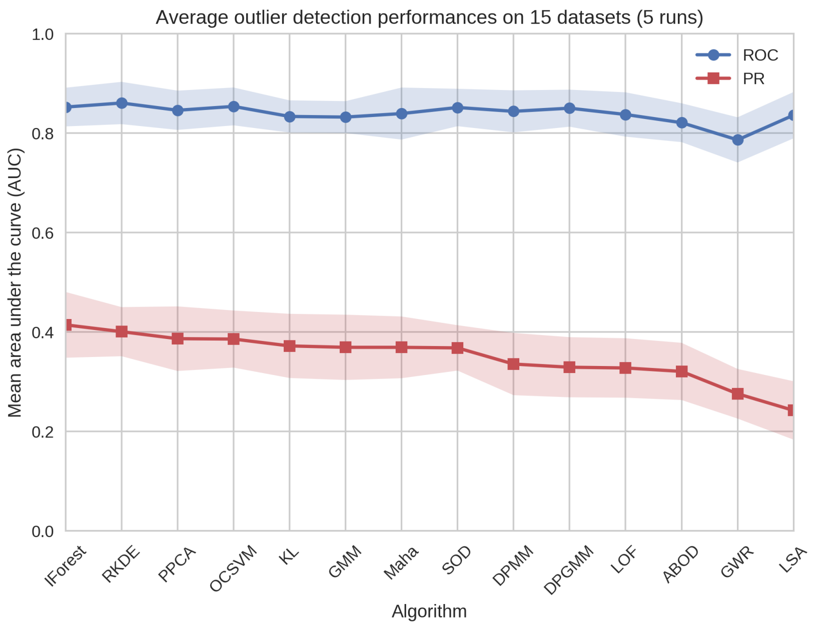

Figure 1: Mean area under the ROC and PR curve per algorithm (descending PR)

Since averaging these metrics may induce a bias in the final ranking caused by extreme values on some specific datasets (e.g. AP of IFOREST on KDD-SUB), we also rank the algorithms per dataset and aggregate the ranking lists without considering the measured values. The Cross-Entropy Monte Carlo algorithm and the Spearman distance are used for the aggregation [22]. The resulted rankings presented in Table 3 are similar to the rankings given in figure 1 and confirm the previous trend observed. The rest of this paper will thus refer to the rankings introduced in figure 1.

Note that we are dealing with heavily imbalanced class distributions where the anomaly class is the positive class. For this kind of problems where the positive class is more interesting than the negative class though underweighted due to the high number of negative samples, precision-recall curves show to be

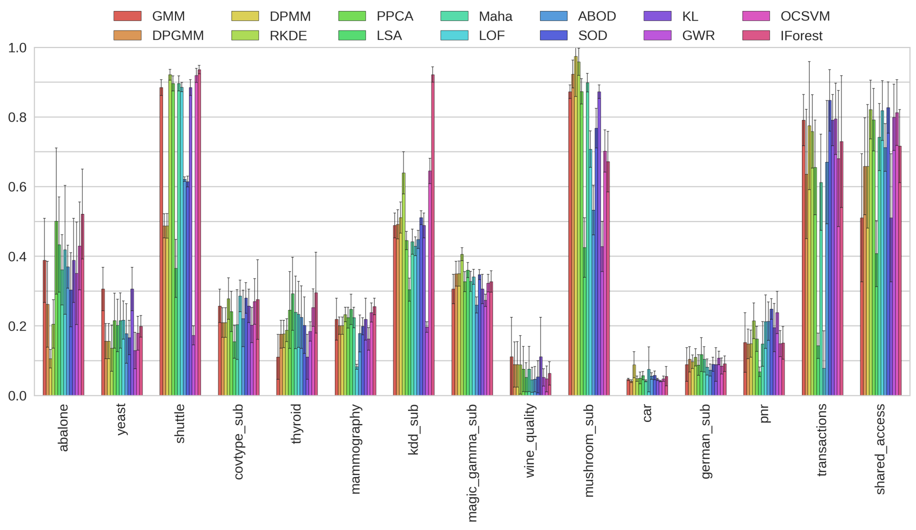

Figure 2: Mean and std area under the precision-recall curve per dataset and algorithm (5 runs)

particularly useful. Indeed, the use of precision strongly penalizes methods raising false positives even if those represent only a small proportion of the negative samples [5].

Looking at the mean average precision, IFOREST, RKDE, PPCA and OCSVM show excellent performance and achieve the best outlier detection results of our benchmark. Outperforming all other algorithms on several datasets (e.g. KDD-SUB, ABALONE or THYROID), IFOREST shows good average performance which makes it a reliable choice for outlier detection. RKDE comes in second position and also shows excellent performance on most datasets, especially when applied to high-dimensionality problems.

One-class SVM achieves good performance without requiring significant tuning. We note that the algorithm perform best on datasets containing a small proportion of outliers, which seems to confirm that the method is well-suited to novelty detection.

The parametric mixture model GMM uses only a single Gaussian, while DPMM and DPGMM often use between 5 and 10 components to model these complex datasets of variable density and shape. The number of components for parametric models, e.g. GMM was selected to maximize the MAP. Regarding most public datasets, note that anomalies are not sparse background noise, but actual samples from one or more classes that are part of classification datasets. The model resulting from nonparametric methods may thus be much more accurate as it could cluster dense clouds of outliers, remnants of former classes. This is confirmed by the results observed for most one-class datasets contaminated by outliers, which are the ones where DPMM and DPGMM outperform GMM, e.g.

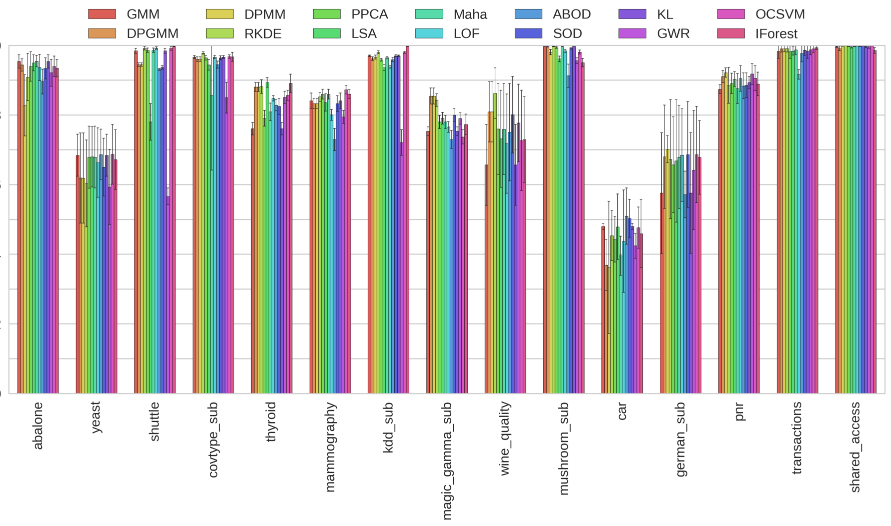

Figure 3: Mean and std area under the ROC curve per dataset and algorithm (5 runs)

MAGIC-GAMMA-SUB, MUSHROOM-SUB and GERMAN-SUB. In addition, the scores and ranking of IFOREST and GMM are much higher for the datasets selected in [7] than for the other datasets where nonparametric methods prevail. Since the selection process described in their study makes use of the performance of several outlier detection methods to perform their selection, this process may benefit algorithms having a behavior similar to the chosen methods.

The good ranking of PPCA, KL and the Mahalanobis distance is also explained by precision-recall measurements very similar to the ones of GMM. Probabilistic PCA is indeed regarded as a GMM with one component, KL is a GMM with a different scoring function, while the model of the Mahalanobis distance is closely related to the multivariate Gaussian distribution. If these simple models perform well on average, they are not suitable for more complex datasets, e.g. the proprietary datasets from Amadeus, where nonparametric methods able to handle clusters of arbitrary shape such as RKDE, SOD or even GWR prevail.

We indeed observe that SOD outperforms the other methods on the Amadeus datasets, and more generally performs very well on large datasets composed of more than 10,000 samples. This method is the best neighbor-based algorithm of our benchmark, but requires a sufficient number of features to infer suitable subspaces through feature selection.

DPMM performs better than DPGMM. As the two methods should converge identically when applied to numerical data, it is the way they handle categorical features that explains this difference. Looking at the detailed average precisions highlighted in Figure 2 for datasets containing categorical features, we notice that DPMM outperforms DPGMM on four datasets (KDD-SUB, MUSHROOM-SUB, CAR and TRANSACTIONS), while it is outperformed on two others (ABALONE and

GERMAN-SUB). This gain of performance for DPGMM is likely to be caused by categorical features strongly correlated to the true class distribution and having a high number of distinct values, resulting in several binarized features and thus a higher weight for the corresponding categorical in the final class prediction. However, when the class distribution is heavily unbalanced, the Chi-Square test based on contingency tables we performed did not allow us to confirm this hypothesis.

The results of DPMM are reached in a smaller training and prediction time due to a smaller dimensionality of the non-binarized input data. Yet, we speculate that DPGMM would reach a smaller computation time if making all its computations for Normal-Wishart distributions instead of exponential-family representations. This was confirmed by measuring the computation time of the BGM implementation in *scikit-learn* on the same datasets.

LOF and ABOD do not stand out, with unexpected drop of performance observed for LOF on TRANSACTIONS and MAMMOGRAPHY that cannot be explained solely based on the dataset characteristics. Using a number of neighbors sufficiently high is important when dealing with large datasets containing a higher number of outliers. Increasing the sample size for ABOD may lead to slightly better performance at the cost of a much higher computation time, for a method which is already slow. We also benchmarked the k-nearest-neighbors approach described in the original paper which showed reduced computation time for $k = 15$ though this did not improve performance. Despite the use of angles instead of distances, this algorithm performs worse than LOF on 4 datasets among the 7 datasets containing more than 40 features. It is however one of the best outlier detection methods on the PNR and TRANSACTIONS datasets.

Although GWR achieves lower performance than other methods in our benchmark, in the case of a low proportion of anomalies, e.g. with PNR, SHARED-ACCESS and TRANSACTIONS, the algorithm reaches excellent precision-recall scores as the density of outliers is not sufficient to attract any neuron. SOM can thus be useful for novelty detection targeting datasets free of outliers, or when combined with a manual analysis of the quantization errors and MID matrix as described in [21]. Similarly to GWR, LSA reaches low average performance, especially for large or high-dimensional datasets, but it achieves good results for small datasets.

4.2. Robustness

As our synthetic datasets contain only numerical data and since DPMM and DPGMM are the same method when the dataset does not contain categorical features, we exclude DPMM from our results and add BGM, the *scikit-learn* optimized equivalent of our DPGMM implementation. Figures 4, 5 and 6 measure the area under the precision-recall curve, the positive class being the background noise for the two first figures, and the nominal samples generated by the mixture of Student's T distributions in the last figure.

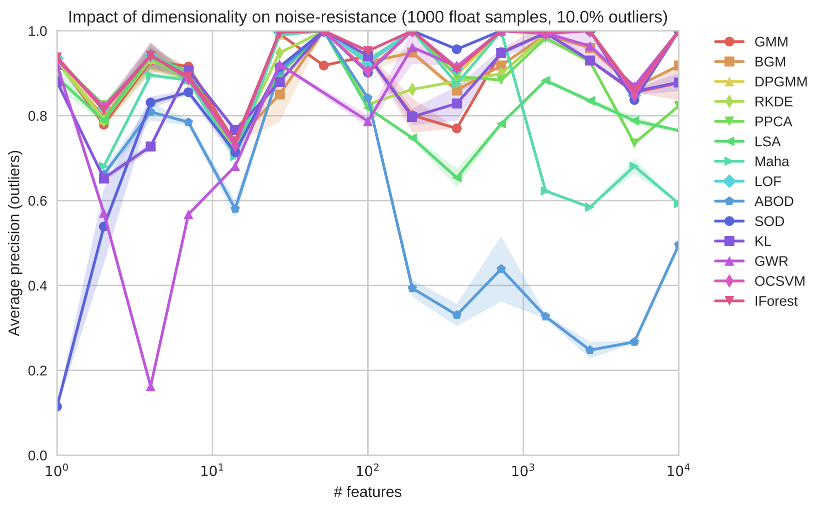

Figure 4: Robustness for increasing number of features

The resistance of each algorithm to the curse of dimensionality is highlighted in figure 4, where we keep a fixed level of background noise while increasing the dataset dimensionality. The results show good performance in average and unexpectedly good results for GWR for more than 10 features. Surprisingly, ABOD which is supposed to efficiently handle high dimensionality performs poorly here. Similarly, LSA and Mahalanobis do not perform well. The difference of results between BGM and DPGMM is likely due to a different cluster responsibility initialization, as BGM uses a K-means to assign data points to clusters centroids scattered among dense regions of the dataset.

Figure 5 shows decreasing outlier detection performance for LSA and GWR while increasing the number of samples in the dataset. SOD and KL do not perform well either, though their results are correlated with the variations of better methods. We thus assume that the current experiment is not well-suited for these method, as SOD applies feature selection and has demonstrated better performance in higher dimensionalities. Increasing the number of samples resulted in an overall increasing precision. The results given for less than 100 data points show a high entropy as the corresponding datasets contain very sparse data in which dense regions cannot be easily identified.

Increasing the proportion of background noise in figure 6 shows a lack of robustness for LSA, SOD, OCSVM, GWR and ABOD. While the two first methods are very sensible to background noise, the three others maintain good performance up to 50% of outliers. Neighbor-based method can only cope with a restricted amount of noise, though increasing the number of samples used to compute the scores could lead to better results. Similarly, most neurons of GWR were attracted by surrounding outliers. In order to avoid penalizing OCSVM, this

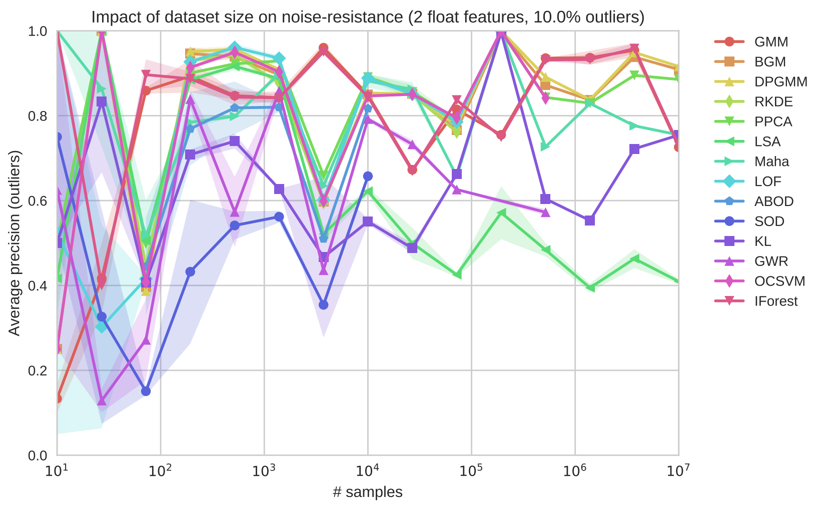

Figure 5: Robustness for increasing number of samples

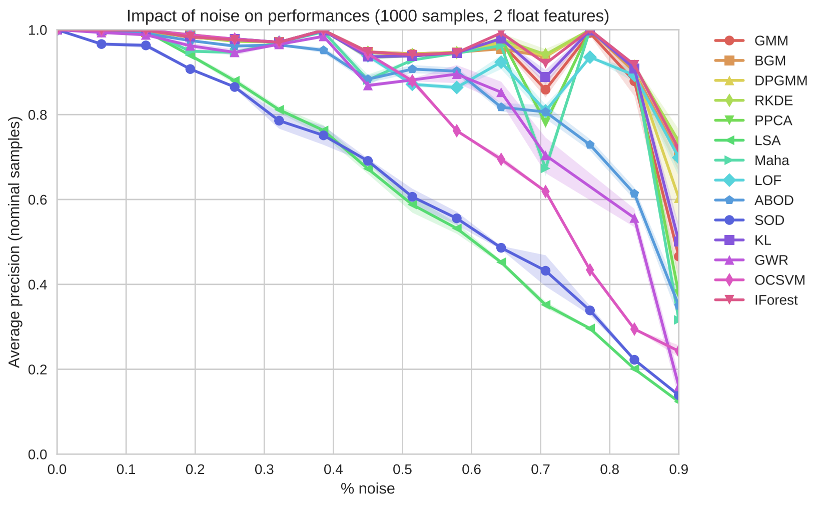

Figure 6: Robustness for increasing noise density

specific evaluation uses an adaptive ν which increases with the proportion of outliers with a minimum value of 0.5. In spite of this measure, OCSVM also shows very poor results above 50% noise. Mahalanobis, KL, LOF and GMM do not perform well either in noisy environments, despite the use of 2 components by GMM and KL. For this experimental setting, the best candidates in a dataset highly contaminated by sparse outliers are IFOREST, DPGMM, BGM, RKDE and PPCA.

To conclude, the robustness measures on synthetic datasets confirm the poor performance of ABOD and GWR showed in our previous ranking. Good average results were observed for IFOREST, OCSVM, LOF, RKDE, DPGMM and GMM. The nearest-neighbor-based methods showed difficulties in handling datasets with a high background noise.

4.3. Complexity

We now focus on the computation and prediction time required by the different methods when increasing the dataset size and dimensionality. The running environment and the amount of optimizations applied to the implementations strongly impacts those measures. For this reason, we focus on the curves' evolutions more than on the actual value recorded. Comparing the measurements of BGM and DPGMM that implement the same algorithm is a good illustration of this statement.

Increasing the number of features in figures 7 and 8 shows an excellent scalability for LSA, GWR and IFOREST with a stable training time and a good prediction time evolution. DPGMM has here the worst training and prediction scaling, reaching the 24 hours timeout for more than 3,000 features. This scaling is confirmed by the increases observed for BGM and GMM. RKDE performs also poorly with a timeout caused by a high bandwidth when the number of features becomes higher than the number of samples. High dimensionality datasets do not strongly affect distance-based and neighbor-based methods, though probabilistic algorithms suffer from the increasing number of dimensions. The use of computationally expensive matrix operations whose complexity depend on the data dimensionality, e.g. matrix factorizations and multiplications, is a major cause of the poor scalability observed. Maximum likelihood estimation fails to estimate the suitable number of components for PPCA for more than 1,000 features. We keep enough components to explain at least 90% of the variance, which explains the decrease of training time.

The number of samples has a strong impact on the training and prediction time of RKDE, SOD, OCSVM, LOF and ABOD which scale very poorly in figures 9 and 10. Those five algorithms would reach the timeout of 24 hours for less than one million samples, though RKDE and LOF exceed the available memory first (section 4.4). All the other algorithms show good and similar scalability, despite a higher base computation time for GWR and DPGMM due to the lack of KDE optimizations. The additional exponential-family computations do not seem to impact the complexity of this algorithm. Training ABOD consists only in making a copy of the training dataset, which explains the low training time reported. Its prediction time is however the least scalable, the true slope being observed for more than 5,000 samples.

In summary and looking at the overall measures, IFOREST and LSA show a very good training and prediction time scaling for both increasing number of features and samples, along with a very small base computation time. DPGMM, GMM and BGM scale well on datasets with a large number of samples and thus

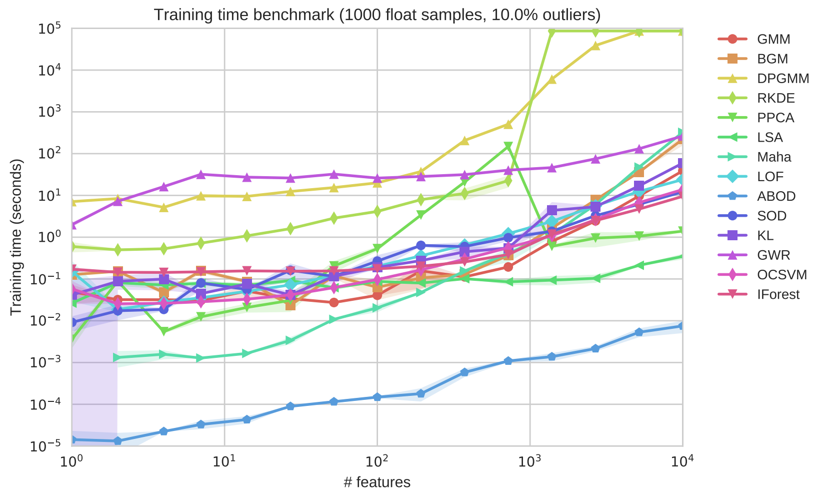

Figure 7: Training time for increasing number of features

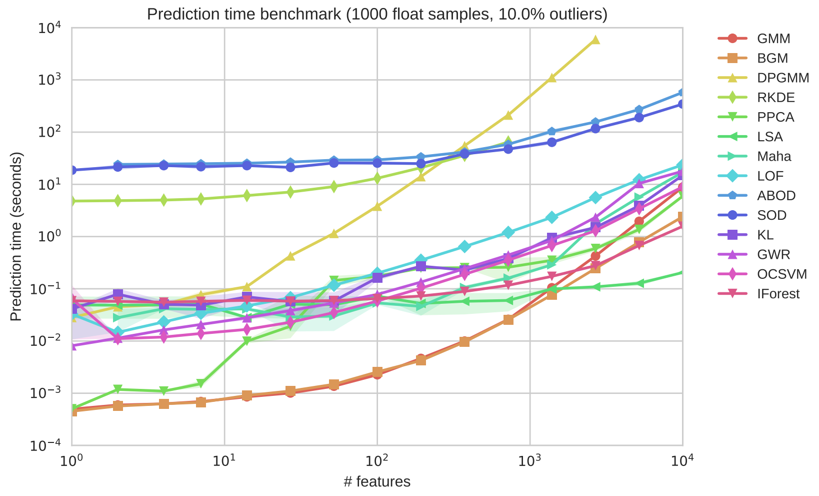

Figure 8: Prediction for increasing number of features

could be suitable for systems where fast predictions matter. The base computation time of DPGMM is however an important issue when the number of features becomes higher than a hundred. RKDE, OCSVM and SOD which have good outlier detection performance on real datasets are thus computationally expensive, which adds interest to IFOREST, DPGMM and simpler models such as GMM, KL, PPCA or Mahalanobis.

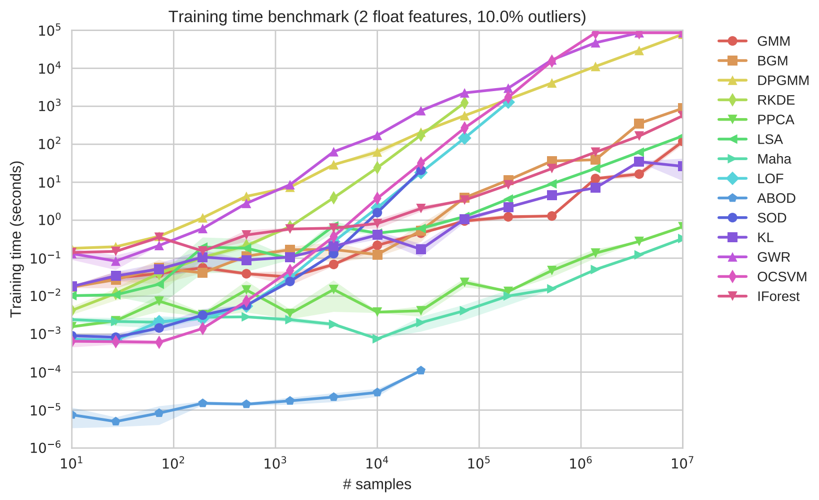

Figure 9: Training time for increasing number of samples

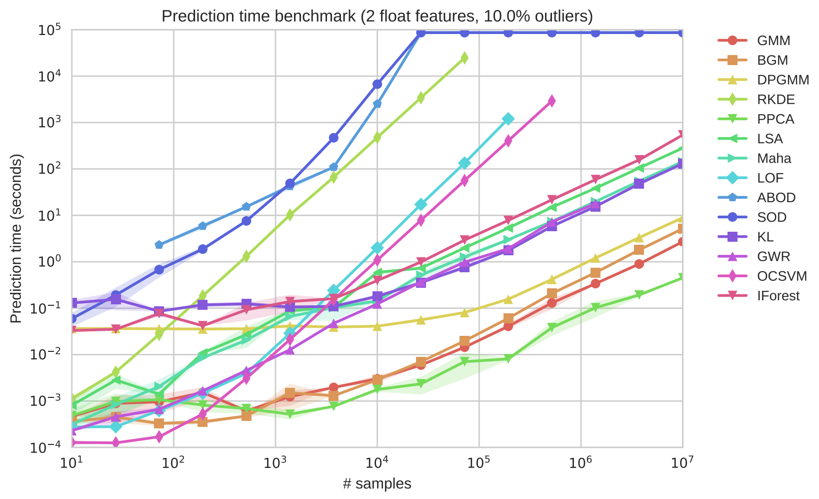

Figure 10: Prediction for increasing number of samples

4.4. Memory usage

We report in figures 11 and 12 the highest amount of memory required by each algorithm when applied to our synthetic datasets during the training or prediction phase. We clear the Matlab objects and make explicit collect calls to the Python and R garbage collectors before running the algorithms. We then measure the memory used by the corresponding running process before starting the algorithm and subtract it to the memory peak observed while running it. This way, our measurements ignore the memory consumption caused by the environment and the dataset which reduces the measurement differences due the running environment, e.g. Matlab versus Python. Measures are taken at intervals of 10^{-4} second using the **memory_profiler** library for Python and R[1] and the UNIX *ps* command for Matlab. Small variations can be observed for measures smaller than 1 MB and are not meaningful.

As depicted in figure 11, most algorithms consume little memory, an amount which does not significantly increase with the number of features and should not impact the running system. IFOREST, OCSVM, LOF and LSA have a constant memory usage below 1 MB while RKDE remains near-constant. GWR, ABOD and SOD also have a good scalability. The other algorithms may require too much memory for high-dimensional problems, with Mahalanobis requiring about 4.5GB to store the mean and covariance matrices of 10,000 features. Allowing many more clusters and storing temporary data structures, DPGMM requires up to 80 GBs when applied to 2,600 features while BGM stores only 14 GBs of data for 10,000 features.

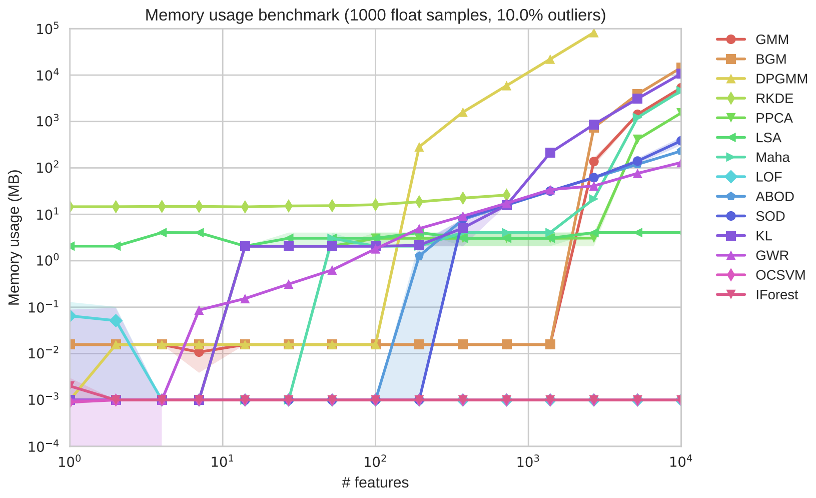

Figure 11: Memory usage for increasing number of features

The increasing number of samples has a higher impact on the RAM consumption depicted in figure 12. RKDE and LOF both run out of memory before the completion of the benchmark and reach, respectively, 158 GB and 118GB memory usage for 72,000 and 193,000 samples. SOD consumes about as much memory as RKDE though reaches the timeout with fewer samples. This amount of memory is mostly caused by the use of a pairwise distance matrix by these algorithms, which requires 76GB of RAM for 100,000 samples using 8 bytes per double precision distance. The other methods scale much better and do not exceed 5GB for 10 million samples, except IFOREST and LSA which allocate

[1] *rpy2* stores R objects in the running Python process. In addition, R prevents concurrent accesses which do not allow us to use dedicated R commands to measure memory.

Table 4: Resistance to the curse of dimensionality, runtime scalability and memory scalability on datasets of increasing size and dimensionality. Performance on background noise detection is also reported for datasets of increasing noise proportion.

	Training/prediction time		**Mem. usage**		**Robustness**		
Algorithm	**↗ Samples**	**↗ Features**	**↗ Samples**	**↗ Features**	**↗ Noise**	**High dim.**	**Stability**
GMM	Low/Low	Medium/Medium	Low	Medium	High	Medium	Medium
BGM	Low/Low	Medium/Medium	Low	Medium	High	Medium	High
DPGMM	Medium/Low	High/High	Low	High	High	High	High
RKDE	High/High	High/High	High	Low	High	High	High
PPCA	Low/Low	High/Low	Low	Low	High	Medium	Medium
LSA	Low/Medium	Low/Low	Medium	Low	Low	Low	Medium
MAHA	Low/Medium	Medium/Low	Low	Medium	Medium	Low	High
LOF	High/High	Low/Low	High	Low	Medium	High	High
ABOD	Low/High	Low/Medium	Low	Low	Medium	Low	Medium
SOD	High/High	Low/Medium	High	Low	Low	High	Medium
KL	Low/Medium	Low/Medium	Low	Medium	High	Medium	High
GWR	Medium/Medium	Medium/Low	Low	Low	Low	High	Medium
OCSVM	High/High	Low/Low	Low	Low	Low	High	High
IFOREST	Low/Medium	Low/Low	Medium	Low	High	High	Medium

60GB and 38GB RAM. For the sake of completeness, we performed the same experiment with the k-nearest neighbors implementation of ABOD with $k = 15$, and observed a scalability and memory usage similar to RKDE.

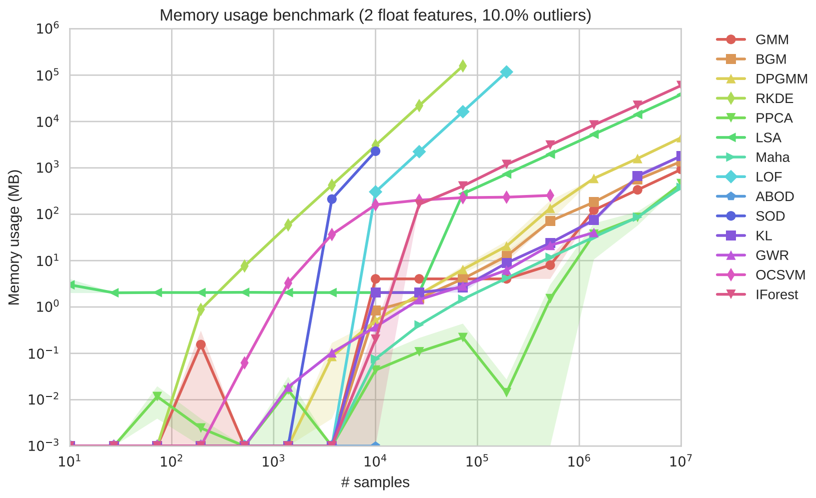

Figure 12: Memory usage for increasing number of samples

We have seen that several algorithms have important memory requirements which must be carefully considered depending on the available hardware. Algorithms relying on multivariate covariance matrices will be heavily impacted by the growing number of features, while methods storing a pairwise distance matrix are not suitable for a large number of samples. Our implementation of DPGMM scales as well as other mixture models though comes with a much higher memory usage on high dimensional datasets. OCSVM, GWR and ABOD have the best memory requirements and scalability and never exceed 250MB RAM, at the cost of a higher computation time since these three methods reach our 24 hours timeout.

4.5. Segmentation contours

Figure 13 shows the normalized density returned by each algorithm when applied to the scaled OLD FAITHFUL dataset. Warm colors depict high anomaly scores. The density was interpolated from the predicted score of 2,500 points distributed on a 50x50 mesh grid.

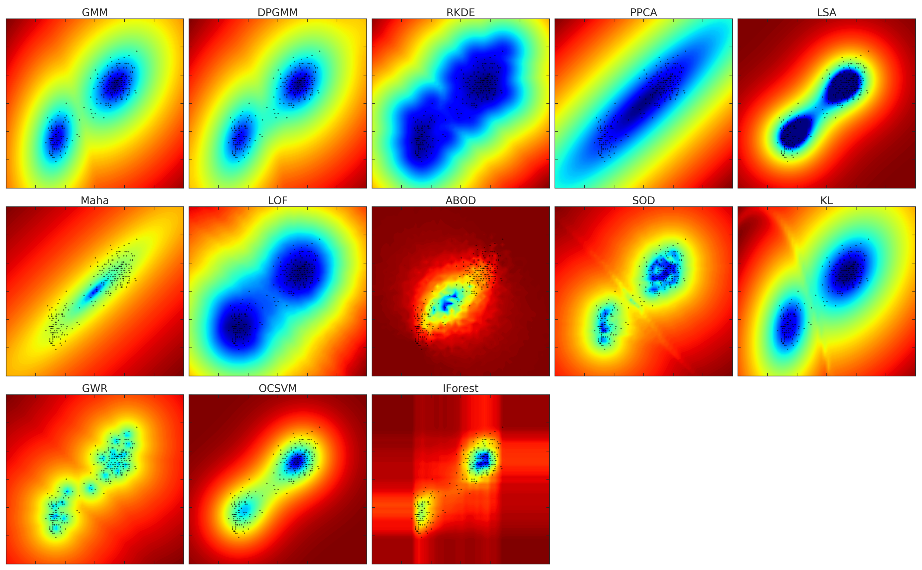

Figure 13: Normalized anomaly scores predicted - Models are trained on the scaled OLD FAITHFUL dataset

The two first plots are the result of Gaussian mixture models, using 2 components for GMM and an upper bound of 10 components for DPGMM. If the plots are very similar, we denote a slightly higher density area between the two clusters for DPGMM. This difference is caused by the remaining mixture components for which the weight is close to 0 and the covariance matrix based on the entire dataset. The information theoretic algorithm based on the KL divergence predicts scores based on a GMM, which explains the similarity between the two.

The contribution of each observation to the overall density is clearly visible for RKDE which finds a density estimation tightly fitting the dataset. In contrast, the models used by PPCA and the Mahalanobis distance are much more constrained and fail at identifying the two clusters. LSA and LOF perform much better though also assign very low anomaly scores to the sparse area located between the two clusters. However, these methods should be able to handle clusters of arbitrary shape.

The density of ABOD is of great interest as it highlights some limitations of the method. In the case of a data distribution composed of several clusters, the lowest anomaly scores are located in the inter-cluster area instead of the cluster centroids due to the sole use of angles variance and values. This is caused by large angles measured when an angle targets two points belonging to distinct clusters, small angles when the points belong to the same cluster and thus a high overall variance for the inter-cluster area. In contrast, the dense areas surrounding the cluster centroids are assigned high anomaly scores since many angles are directed toward the other clusters, suggesting data points isolated and far from a major cluster. The same issue arises for the inter-cluster area when computing the anomaly scores with the alternative k-nearest neighbors approach instead of randomly sampling from the dataset.

SOD is able to estimate a very accurate density, despite some low scores between the clusters. The neurons belonging to the GWR network result in circular blue areas highlighting their position. The presence of a neuron at the center of the plot once again results in very low scores for the inter-cluster area, as low as for the theoretical cluster centroids. Using an additional threshold to detect outlying neurons as suggested in [21] would solve this issue.

A high density is also assigned by OCSVM to this region due to the mapping of the continuous decision boundary from the high-dimensional space. The segmentation made by IFOREST is much tighter than previous methods and seem less prone to overfitting than RKDE. Light trails emerging from the clusters can however be observed and may result in anomalous observations receiving a lower score than patterns slightly deviating from the mean.

5. Conclusions

In the context of outlier detection, we benchmarked the average precision, robustness, computation time and memory usage of 14 algorithms on synthetic and real datasets. Our study demonstrates that IFOREST is an excellent method to efficiently identifying outliers while showing an excellent scalability on large datasets along with an acceptable memory usage for datasets up to one million samples. The results suggest that this algorithm is more suitable than RKDE in a production environment as the latter is much more computationally expensive and memory consuming. OCSVM is a good candidate in this benchmark, but it is not suitable either for large datasets.

Sampling a small proportion of outliers from classification datasets as suggested in [7] resulted in dense clouds of outliers which made simple methods such as GMM with one component, KL, PPCA and the Mahalanobis distance perform well. However, these scalable solutions are not generally optimal for datasets where the nominal class has a complex shape or is distributed across several distinct clusters. In these cases, the results indicate that nonparametric clustering alternatives are superior.

SOD showed good outlier detection performance and efficiently handled high-dimensional datasets at the cost of poor scalability. Exponential-family representations for DPMM revealed to be extremely time-consuming without substantially improving the detection of outliers made by Gaussian-based approaches such as DPGMM. Nonetheless, the use of categorical distributions in DPMM resulted in a reduced computation time and better outlier detection performance. LOF, ABOD, GWR, KL and LSA reached the lowest performance while the three first methods also showed poor scalability. The ability to accurately shape nominal data was assessed for each method while a borderline case was highlighted for ABOD by studying the distribution of the anomaly scores in datasets composed of several clusters.

While the coverage of this study should suffice to tackle most outlier detection problems, specific algorithms may be chosen for constrained environments. Distributed implementations, streaming or mini-batch training are a prerequisite to deal with large datasets, and several methods have been extended to support these features, e.g. DPMM, K-MEANS, SVM or GMM on Spark MLlib [20]. Other promising directions leading the research on outlier detection also focus on ensemble learning [32] and detecting outliers from multi-view data [11].

6. Acknowledgments

The authors wish to thank the Amadeus Middleware Fraud Detection team directed by Virginie Amar and Florent Maupay, led by the product owner Christophe Allexandre and composed of Jihane Zouaoui, Jean-Blas Imbert, Jiang Wu and Damien Fontanes for building and labeling the TRANSACTIONS and SHARED-ACCESS datasets. The authors are also grateful to the Amadeus PNR-ADI team, especially to Francesco Buonora, for their work in supplying a usable dataset.

7. References

[1] K. Bache and M. Lichman. UCI machine learning repository, 2013.

[2] I. Ben-Gal. Outlier detection. In *Data Mining and Knowledge Discovery Handbook*, pages 131–146. Springer, 2005.

[3] D. M. Blei and M. I. Jordan. Variational inference for dirichlet process mixtures. *Bayesian Analysis*, 1(1):121–143, 03 2006.

[4] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. LOF: Identifying density-based local outliers. *SIGMOD Record*, 29(2):93–104, May 2000.

[5] J. Davis and M. Goadrich. The relationship between precision-recall and ROC curves. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pages 233–240. ACM, 2006.

[6] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.

[7] A. Emmott, S. Das, T. Dietterich, A. Fern, and W.-K. Wong. Anomaly detection meta-analysis benchmarks. 2016. arXiv:1503.01158v2.

[8] W. Fan, N. Bouguila, and D. Ziou. Unsupervised anomaly intrusion detection via localized bayesian feature selection. In *Proceedings of the 2011 IEEE 11th International Conference on Data Mining*, ICDM '11, pages 1032–1037. IEEE Computer Society, 2011.

[9] M. Filippone and G. Sanguinetti. Information theoretic novelty detection. *Pattern Recognition*, 43(3):805–814, 2010.

[10] V. Hodge and J. Austin. A survey of outlier detection methodologies. *Artificial Intelligence Review*, 22(2):85–126, Oct. 2004.

[11] T. Iwata and M. Yamada. Multi-view anomaly detection via robust probabilistic latent variable models. In *Advances in Neural Information Processing Systems 29*, NIPS '16, pages 1136–1144. Curran Associates, Inc., 2016.

[12] N. Japkowicz and S. Stephen. The class imbalance problem: A systematic study. *Intelligent Data Analysis*, 6(5):429–449, Oct. 2002.

[13] J. Kim and C. D. Scott. Robust kernel density estimation. *Journal of Machine Learning Research*, 13:2529–2565, Sep 2012.

[14] T. Kohonen. The self-organizing map. *Neurocomputing*, 21(1):1–6, 1998.

[15] H.-P. Kriegel, P. Kröger, E. Schubert, and A. Zimek. Outlier detection in axis-parallel subspaces of high dimensional data. In *Advances in Knowledge Discovery and Data Mining: 13th Pacific-Asia Conference, PAKDD 2009 Bangkok, Thailand, April 27-30, 2009 Proceedings*, PAKDD '09, pages 831–838. Springer, 2009.

[16] H.-P. Kriegel, M. S hubert, and A. Zimek. Angle-based outlier detection in high-dimensional data. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, pages 444–452. ACM, 2008.

[17] F. T. Liu, K. M. Ting, and Z.-H. Zhou. Isolation forest. In *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, ICDM '08, pages 413–422. IEEE Computer Society, 2008.

[18] H. Liu, S. Shah, and W. Jiang. On-line outlier detection and data cleaning. *Computers & chemical engineering*, 28(9):1635–1647, 2004.

[19] S. Marsland, J. Shapiro, and U. Nehmzow. A self-organising network that grows when required. *Neural Networks*, 15(8-9):1041–1058, Oct. 2002.

[20] X. Meng, J. Bradley, B. Yavuz, E. Sparks, S. Venkataraman, D. Liu, J. Freeman, D. Tsai, M. Amde, S. Owen, D. Xin, R. Xin, M. J. Franklin, R. Zadeh, M. Zaharia, and A. Talwalkar. Mllib: Machine learning in Apache Spark. *Journal of Machine Learning Research*, 17(1):1235–1241, Jan. 2016.

[21] A. Munoz and J. Muruzábal. Self-organizing maps for outlier detection. *Neurocomputing*, 18(1):33–60, 1998.

[22] V. Pihur, S. Datta, and S. Datta. RankAggreg, an r package for weighted rank aggregation. *BMC Bioinformatics*, 10(1):62, Feb 2009.

[23] M. A. F. Pimentel, D. A. Clifton, L. Clifton, and L. Tarassenko. A review of novelty detection. *Signal Processing*, 99:215–249, June 2014.

[24] M. Prastawa, E. Bullitt, S. Ho, and G. Gerig. A brain tumor segmentation framework based on outlier detection. *Medical Image Analysis*, 8(3):275–283, 2004.

[25] J. A. Quinn and M. Sugiyama. A least-squares approach to anomaly detection in static and sequential data. *Pattern Recognition Letters*, 40:36–40, Apr. 2014.

[26] P. B. Schölkopf, R. C. Williamson, A. J. Smola, J. Shawe-Taylor, and J. C. Platt. Support vector method for novelty detection. In *Advances in Neural Information Processing Systems 12*, NIPS '00, pages 582–588. MIT Press, 2000.

[27] D. W. Scott. *Multivariate density estimation: theory, practice, and visualization.* John Wiley & Sons, 1992.

[28] L. Tarassenko, P. Hayton, N. Cerneaz, and M. Brady. Novelty detection for the identification of masses in mammograms. In *4th International Conference on Artificial Neural Networks*, pages 442–447. IET, 1995.

[29] M. E. Tipping and C. M. Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society. Series B (Methodological)*, 61(3):611–622, 1999.

[30] J. Vanschoren, J. N. van Rijn, B. Bischl, and L. Torgo. OpenML: Networked science in machine learning. *SIGKDD Explorations Newsletter*, 15(2):49–60, June 2014.

[31] K. Worden, G. Manson, and N. Fieller. Damage detection using outlier analysis. *Journal of Sound and Vibration*, 229(3):647–667, 2000.

[32] A. Zimek, R. J. Campello, and J. Sander. Ensembles for unsupervised outlier detection: Challenges and research questions a position paper. *SIGKDD Explorations Newsletter*, 15(1):11–22, Mar. 2014.

[33] A. Zimek, E. Schubert, and H.-P. Kriegel. A survey on unsupervised outlier detection in high-dimensional numerical data. *Statistical Analysis and Data Mining*, 5(5):363–387, 2012.