Mach Learn (2016) 102:275–304
DOI 10.1007/s10994-015-5521-0

Loda: Lightweight on-line detector of anomalies

Tomáš Pevný[1,2]

Received: 2 November 2014 / Accepted: 25 June 2015 / Published online: 21 July 2015

Abstract In supervised learning it has been shown that a collection of weak classifiers can result in a strong classifier with error rates similar to those of more sophisticated methods. In unsupervised learning, namely in anomaly detection such a paradigm has not yet been demonstrated despite the fact that many methods have been devised as counterparts to supervised binary classifiers. This work partially fills the gap by showing that an ensemble of very weak detectors can lead to a strong anomaly detector with a performance equal to or better than state of the art methods. The simplicity of the proposed ensemble system (to be called Loda) is particularly useful in domains where a large number of samples need to be processed in real-time or in domains where the data stream is subject to concept drift and the detector needs to be updated on-line. Besides being fast and accurate, Loda is also able to operate and update itself on data with missing variables. Loda is thus practical in domains with sensor outages. Moreover, Loda can identify features in which the scrutinized sample deviates from the majority. This capability is useful when the goal is to find out what has caused the anomaly. It should be noted that none of these favorable properties increase Loda's low time and space complexity. We compare Loda to several state of the art anomaly detectors in two settings: batch training and on-line training on data streams. The results on 36 datasets from UCI repository illustrate the strengths of the proposed system, but also provide more insight into the more general questions regarding batch-vs-on-line anomaly detection.

Electronic supplementary material The online version of this article (doi:10.1007/s10994-015-5521-0) contains supplementary material, which is available to authorized users.

Editor: Joao Gama.

✉ Tomáš Pevný
pevnak@gmail.com

[1] Department of Computers and Engineering, Czech Technical University in Prague, Karlovo naměstí 13, 121 35 Prague, Czech Republic

[2] Cisco Systems, Inc., Cognitive Research Team in Prague, Prague, Czech Republic

1 Introduction

Imagine identifying anomalous users in a social network (Šourek et al. 2013), where user's behavior constantly changes and their numbers are enormous, detecting weirdly behaving computers (frequently an indication of an infection by malware) in a network of large corporation with hundreds of thousands of computers, whose traffic constantly changes (Pevný et al. 2012), or identification of fraudulent card transactions (Akhilomen 2013) realized thorough big credit providers. These domains share similar features, which is processing of enormous number of samples with constantly changing characteristics. Most fast versions of existing anomaly detection methods, especially those based on indexing techniques, require the data to be available in one single batch and to fit in the memory, which is in aforementioned domains clearly impossible. Moreover, data's non-stationarity forces detector's models to be continuously updated, which is again very difficult with indexing techniques, as created indexes would need to be recalculated, which is usually expensive. Other methods, such as Bay and Schwabacher (2003) assumes some additional knowledge which might not be available. The presented anomaly detector has been designed with respect to these constraints, and it has been shown to achieve state of the art accuracy measured by area under ROC curve.

The definition of an anomaly, outlier, or novelty is not unified in the literature. Recent book (Aggarwal 2013a) considers all these terms to be equivalent, and defines them as follows: *"An outlier is an observation which deviates so much from the other observations as to arouse suspicions that it was generated by a different mechanism."* According to this definition, *outliers* are generated by the same probability distribution as normal samples, but they are very rare. In this text, we define *anomalies* to be samples generated by a different probability distribution than normal ones, and their presence in the data to be rare, less than 1–10 % of the data. One-class problem is similar to the anomaly-detection problem with the difference that training set contains samples exclusively from the normal class and testing set contains may contain samples from other classes at any rate.

An ideal detector would model joint probability of data-generating processes. Although this goes against the principle of never solving a more difficult process than is needed (density estimation vs. classification), the knowledge of the probability of observed samples is useful[1] information in making a decision about their anomalousness. Modeling joint probability is generally difficult in spaces of many dimensions and in practice some simplifications have to be made. The detector presented here, further called Loda, approximates the joint probability by using a collection of one-dimensional histograms, where every one-dimensional histogram is constructed on an input space projected onto a randomly generated vector. The rationale behind the use of one-dimensional histograms is that they can be efficiently constructed in one pass over data and the query operation needed during classification is simple. Consequently, Loda's complexity is linear with respect to the number of training samples n and the dimension of the input space d.

Although one one-dimensional histogram is a very weak anomaly detector, their collection yields to a strong detector. This phenomenon (collection of weak classifiers result in a strong classifier) is already a well established paradigm in supervised classification (Kuncheva 2004; Freund and Schapire 1996), but has not been demonstrated in unsupervised anomaly detection, as most ensemble systems used in anomaly detection (Aggarwal 2013b; Lazarevic and Kumar 2005; Tan et al. 2011; Liu et al. 2008) use individual detectors of much higher complexity than that of a one-dimensional histogram. The comprehensive experimental section

[1] The hypothetical knowledge of the probability function generating the data would allow to formulate the problem as a hypothesis test.

(see Sect. 4) demonstrates that the proposed ensemble of one-dimensional histograms has accuracy measured by the area under ROC curve competitive with established solutions of much higher complexities. Loda therefore achieves a very good accuracy to complexity ratio and therefore it is well suited for processing large data.

Besides being fast, Loda is able to deal with missing variables and can rank features according to their contribution to sample's anomalousness. Both of these capabilities are important for practitioners. For example, the knowledge of which features caused the anomaly can serve as guidance for further investigation and can decrease overall cost of anomaly investigation. This knowledge can also be used to cluster similar anomalies together which is again useful during investigation. Interestingly, none of these abilities substantially increase Loda's computational complexity. Note that both abilities can be achieved with any homogenous ensemble-based anomaly detector, where individual detectors within are diversified by sub-space sampling. Detectors with sub-space sampling were recently proposed (Keller et al. 2012; Nguyen et al. 2010; Muller et al. 2011) but neither the robustness against missing features or the explanation of the cause of a sample's anomalousness had been mentioned.

Loda is controlled by two hyper-parameters: number of one dimensional histograms and number of histogram bins. Because the hyper-parameter setting is particularly cumbersome in most anomaly detection methods where few or no anomalous samples are available (most anomaly detection), a method to determine these parameters solely on the basis of observed samples is presented. This makes Loda hyper-parameter free.

Loda's accuracy is firstly extensively compared to the prior art trained in the batch mode on 36 problems downloaded from the UCI database (Frank and Asuncion 2010) in the category of classification problems with numerical attributes. The reason, why Loda is compared to detectors processing only data presented in one batch is that we wanted to compare Loda to state of the art detectors, even though they are not applicable for intended scenario. The experimental results reveal that Loda's measured by area under ROC curve is equal to and many times better than that of other algorithms with higher computational complexity requiring batch training (we expect algorithms trained in batch to perform better than on-line algorithms). Consequently Loda provides a good alternative to the already established solutions on problems with a large number of features and samples that need to be processed efficiently. Secondly, Loda is compared to the prior art on streaming problems and it is shown when it is better to continuously update histogram and when to do so in batches. Finally, Loda's capability to efficiently process big datasets is demonstrated on the WEB UCI dataset (Ma et al. 2009) with 2.4 million samples and 3.2 million features.

The contribution of this paper is threefold: (i) it demonstrates that an ensemble of very weak anomaly detectors can lead to a strong anomaly detector; (ii) it presents a lightweight anomaly detector suitable for data-streams robust to missing variables and identifying causes of anomalies; (iii) it extensively compares existing and the proposed anomaly detector, which sheds some light on conditions in which a particular anomaly detector excels.

This paper is organized as follows: The next section reviews relevant prior art, shows its computational complexity, and discusses issues related to on-line learning and classification. Loda is presented in Sect. 3. In Sect. 4, Loda is experimentally compared to the prior art, its low computational requirements are demonstrated on artificial data, and its efficiency is demonstrated on a large-scale dataset (Ma et al. 2009). The same section also demonstrates Loda's ability to handle data with missing values and explain a sample's anomalousness. Finally Sect. 5 concludes the paper.

For better reproducibility, all source code, results, datasets, and their creation is available at http://agents.cz/~pevnak.

2 Related work

The recent book (Aggarwal 2013a) and survey (Chandola et al. 2009) contain many methods for anomaly and outlier detection. Below, those relevant to this work are reviewed.

2.1 Model-centered detectors

Basic model-centered anomaly detectors assume that data follow a known distribution. Detectors (Shyu et al. 2003) based on principal component analysis (PCA) assume that the data fit a multi-variate normal distribution. Despite this being rarely true in practice, they frequently outperform more complicated detectors. The complexity of their training is $O(nd^3)$, where d is the dimension of the input space and n is the number of samples used for training.

A One-Class Support Vector Machine (Schölkopf et al. 2001) (1-SVM) does not make any assumptions about the distribution of data. It finds the smallest area where $1-\nu$ fraction of data are located (ν is parameter of the method specifying desired false-positive rate). This is achieved by projecting the data to a high-dimensional (feature) space and then finding a hyperplane best separating the data from the origin. It has been noted that when 1-SVM is used with a linear kernel it introduces a bias to the origin (Tax and Duin 2004), which can be removed by using a Gaussian kernel. The Support Vector Data Description algorithm (Tax and Duin 2004) (SVDD) removes the bias in 1-SVM by replacing the separation hyperplane by a sphere encapsulating most of the data. The complexity of both methods is super-linear with respect to the number of samples, n, being $O(n^3 d)$ in the worst-case.

Isolation Forest (Liu et al. 2008) (IForest) relies on a collection of Isolation Trees grown by using unlabeled data. During growth of a tree, its internal nodes are added until the terminal leafs contain one sample or the maximum depth is reached. The anomaly score is proportional to the level of leaf reached by the sample, as the idea is that anomalies will reach leafs at the base of the tree (close to the root), while legitimate samples reach leafs closer to the root. The complexity of training one tree is $O(n \log n)$ and of classification $O(\log n)$, where the authors recommend subsampling of training samples to increase diversity in the ensemble, robustness against anomalies within the data, and simultaneously decrease complexity of the training and classification.

The recently proposed FRAC (Noto et al. 2012) aimed to bridge the gap between supervised and unsupervised learning. FRAC is an ensemble of models, each estimating one feature based on other features (for data of a dimension d, FRAC uses d different models). The rationale behind this is that anomalous samples exhibit different dependencies among features, which can be detected from prediction errors modeled by histograms. FRAC's complexity depends on the algorithm used to implement individual models, which can be large, considering that a search for possible hyper-parameters needs to be undertaken. Because of this, an ordinary linear least-square regression is used here leading to the complexity $O(nd^4)$. FRAC is not well suited for on-line learning since an update of models changes the distributions of errors that cannot be estimated from one sample.

The on-line training of model-centered detectors is generally difficult as the algorithms used to create the model have non-trivial complexity with respect to the dimension, or models cannot be updated incrementally (e.g. principal component analysis). The on-line adaptation of 1-SVM is discussed in Kivinen et al. (2004), but the solution is an approximation of the solution returned by the batch version. The exact on-line version of SVDD is described in Tax and Laskov (2003), but the algorithm requires substantial bookkeeping thereby increasing the complexity. The proposed Loda is also model-based detector and as will be shown in

Sect. 4, its one dimensional histograms can be easily updated based on upcoming samples which makes the whole algorithm friendly to on-line learning.

2.2 Data-centered detectors

Data-centered detectors do not have any built-in model. A sample's anomalousness is determined according to its distance to all previously observed samples. Consequently, there is no training phase as new samples are just added to the set of already observed samples. However, this increases the complexity of the classification phase, which is a linear function of the number of samples n.

A k-nearest neighbor (Knorr and Ng 1999) (KNN) is a popular method to identify outliers inspired by the corresponding method from classification. It ranks samples according to their distance to kth-nearest neighbor. It has been recently shown that a variant of KNN (Sricharan and Hero (2011)) converges to the optimal density-level detector for a given false positive rate. Nevertheless KNN has been criticized for not being able to detect outliers in data with clusters of different densities (Breunig et al. 2000). The local outlier factor (Breunig et al. 2000) (LOF) solves this problem by defining the outlier score as a fraction of sample's distance to its kth-nearest neighbor and the average of the same distance of all its k nearest neighbors. True inliers have a score around one while outliers have much greater score. We refer to Zimek et al. (2012) for a comprehensive review of the prior art.

The complexity of the classification phase of nearest-neighbor based detectors is driven by the nearest-neighbor search, which is an $O(n)$ operation in the worst case. More efficient approaches based on bookkeeping (Pokrajac et al. 2007), better search structures like KD-trees (Bentley 1975), or approximate search (Andoni and Indyk 2006) have been adopted. Nevertheless, the complexity of all methods depends in some way on the number of training samples n, and better search structures are usually useful only in low dimensions.

2.3 On-line anomaly detectors

There are few works in on-line anomaly detection. The closest work to this is Half-Space trees (Tan et al. 2011) (HS-trees), which is a method similar to Isolation Forest with the difference that decision rules within tree-nodes are generated randomly. The output of each HS-tree can be interpreted as a crude estimate of the probability density function, which is further refined by taking a sufficient number of them. It is worth noting that HS-trees assume that the data is scaled such that values of features are bounded in [0, 1]. This is in a sharp contrast to Loda only requiring features to have approximately the same scale, which is a more relaxed condition, especially if concept drift occurs. HS-trees handle concept-drift by dividing data-streams into sample batches of size 256, where HS-trees trained on a previous batch are used to scrutinize samples in a given batch. Simultaneously new HS-trees are learned on a current batch and once all samples from the current batch are processed, new HS-trees replaces the old one.

OLINDDA (Spinosa et al. 2009) uses standard k-means clustering to group previously observed samples into clusters called concepts. Anomalies not belonging to any cluster are grouped into candidate clusters, which based on cohesiveness criteria are either promoted as a new cluster and novel concept is reported, or evicted. Thus OLINDDA focuses on detection of a novel concepts in streams rather than on identification of anomalies.

2.4 Ensembles and random projections

Ensembles have so far been underutilized in anomaly detection. A significant portion of the prior art focuses on a unification of anomaly scores (Gao and Tan 2006; Schubert 2012), because different detectors within one ensemble may provide anomaly scores of different magnitudes (Nguyen et al. 2010). Diversifying detectors by random subsets of features was presented (Lazarevic and Kumar 2005; Keller et al. 2012) to improve the accuracy, especially on high dimensional problems.

Projections of the input space onto the arbitrary sub-space have been utilized mainly in distance-based outlier detection schemes to speedup the search of nearest neighbors. de Vries et al. (2010) performs the kth-NN search in the LOF first in the random sub-space on a larger neighborhood and then it is refined by the search in the original input space. Similarly, Pham and Pagh (2012) estimates the distribution of angles between samples in the input space from the distribution of angles in the sub-space (distribution of angles has been proposed in Kriegel and Zimek (2008) as a good anomaly criterion). In experiments in Sect. 4.3 the similar approach is used on prior art to compare it to Loda.

Most prior art employs ensembles and random projections with data-centered detectors, which have high computational complexity and are not well suited for real-time processing. Exceptions are Isolation Forest (Liu et al. 2008) and Half-Space Trees (Tan et al. 2011) relying on an ensemble of trees, each having a relatively low accuracy. This is similar to Loda which uses even simpler model (one-dimensional histogram) with even lower complexity, but their combination is similarly powerful. Random projections in Loda are used to project the input space into a single dimension, which simplifies the complexity of all operations over it. Loda's sparse random projections can be considered as a sub-space sampling method. Unlike the prior art, the sub-space method is not used only to increase efficacy, but also to gain robustness against missing variables and the ability to find causes of anomalousness of a given sample.

Interpreting the decision in anomaly detection is very important in practice since it can reduce the cost of subsequent analysis and increases trust of the detector. The work of Knorr and Ng (1999) has focused on finding a minimal sub-space in which a given sample is an outlier. Algorithms presented therein are data-centered and suitable only for low-dimensions. HiCS algorithm (Keller et al. 2012) and the algorithm of Dang et al. (2013) both aim to identify a sub-space in which a scrutinized sample is an outlier. Both are data-centered algorithms and consequently their computational complexity is prohibitive for processing data-streams.

Loda's method to identify relevant features bears similarity to HiCS algorithm. But unlike HiCS, Loda's method is general and can be used on all ensembles diversified by random sub-space sampling. Finally, contrary to all prior art, the identification of relevant features does not increase Loda's computational complexity in big O notation.

3 Description of Loda

Loda is comprised of a collection of k one-dimensional histograms $\{h_i\}_{i=1}^k$, each approximating the probability density of input data projected onto a single projection vector $\{w_i \in \mathbb{R}^d\}_{i=1}^k$. Projection vectors $\{w_i\}_{i=1}^k$ diversify individual histograms, which is an essential requirement for ensemble systems to improve performance of a single classifier / detector. Their initialization is described in detail below in Sect. 3.1.

Loda's output, $f(x)$, on a sample, x, is an average of the logarithm of probabilities estimated on individual projection vectors. Adopting $\hat{p}_i$ to denote the probability estimated by

Algorithm 1: Loda's training (update) routine.

input: data samples $\{x_i \in \mathbb{R}^d\}_{i=1}^n$;
output: histograms $\{h_1, \ldots, h_n\}$, projection vectors $\{w_i\}_{i=1}^k$. ;
initialize projection vectors with $\left[d^{-\frac{1}{2}}\right]$ non zero elements $\{w_i\}_{i=1}^k$;
initialize histograms $\{h_i\}_{i=1}^k$;
for $j \leftarrow 1$ **to** n **do**
  for $i \leftarrow 1$ **to** k **do**
    $z_i = x_j^{\mathrm{T}} w_i$;
    update histogram h_i by z_i;
  end
end
return $\{h_i\}_{i=1}^k$ and $\{w_i\}_{i=1}^k$.

ith histogram and w_i to denote the corresponding projection vector, Loda's output $f(x)$ can be written as

$$f(x) = -\frac{1}{k}\sum_{i=1}^{k} \log \hat{p}_i(x^{\mathrm{T}} w_i), \tag{1}$$

which can be reformulated as

$$\begin{aligned} f(x) &= -\log\left(\prod_{i=1}^{k} \hat{p}_i(x^{\mathrm{T}} w_i)\right)^{\frac{1}{k}} \\ &\overset{(a)}{\sim} -\log p(x^{\mathrm{T}} w_1, x^{\mathrm{T}} w_2, \ldots, x^{\mathrm{T}} w_k), \end{aligned} \tag{2}$$

where $p(x^{\mathrm{T}} w_1, x^{\mathrm{T}} w_2, \ldots, x^{\mathrm{T}} w_k)$ denotes the joint probability of projections. Equation (2) shows that Loda's output is proportional to the negative log-likelihood of the sample, which means that the less likely a sample is, the higher the anomaly value it receives. This holds under a strong assumption (a) in (2) that probability distributions on projection vectors w_i and w_j are independent $\forall i, j \in k,\ i \neq j$. Even though this is almost never true in practice, Loda still delivers very good results. We believe that the reasons are similar to those in naïve Bayes classifiers theoretically studied by Zhang (2004), which give conditions under which the effects of conditional dependencies cancel out. These conditions depend on the probability distribution of both classes and they are difficult to be verified in practice as they require an exact knowledge of conditional dependencies among features. Due to Loda's similarity to the Parzen window detector (Yeung and Chow 2002), the similar argumentation might explain Loda's surprisingly good performance.

In high-dimensional spaces Loda can be related to a PCA based detector (Shyu et al. 2003), because projection vectors w_i and $w_j, i \neq j$ will be approximately orthogonal (this is due to their random construction described below). Assuming again the independence of $x^{\mathrm{T}} w_i$ and $x^{\mathrm{T}} w_j$, the projected data are orthogonal and uncorrelated, which are properties of Principal Component Analysis (PCA). Loda's histogram on random projections are therefore similar to histograms on principal components.

Loda's training and classification routines are summarized in Algorithms 1 and 2, respectively. Loda is initialized by generating a set of sparse random projections $\{w_i\}_{i=1}^k$, $w_i \sim N(0, \mathbf{1}^d)$ with $d^{\frac{1}{2}}$ non-zero components and initializing the corresponding set of histograms

Algorithm 2: Loda's classification routine on sample x.

input: sample x, set of histograms $\{h_i\}_{i=1}^k$ and projection vectores $\{w_i\}_{i=1}^k$.;
output: anomaly value $f(x)$;

for $i \leftarrow 1$ **to** k **do**
  $z_i = x^{\mathrm{T}} w_i$;
  obtain $\hat{p}_i = \hat{p}_i(z_i)$ from h_i ;
end
return $f(x) = -\frac{1}{k}\sum_{i=1}^{k} \log \hat{p}_i(z_i)$;

$\{h_i\}_{i=1}^k$. Each histogram is updated by a training sample by projecting the sample onto a vector and then the corresponding histogram bin is updated. The classification procedure follows similar steps, but instead of updating histograms, they return probabilities whose logarithms are averaged and returned. Notice that the construction requires only one pass over the data and can be used on data-streams by first classifying a new sample and then updating all histograms.

The rest of this section describes creation of and rationale behind sparse projection vectors w_i, presents how sparse projections enable robustness against missing variables and allow explanation of sample's anomalousness, and closes by commenting issues related to building on-line histograms.

3.1 Random projections

Each sparse projection vector $\{w_i\}_{i=1}^k$ is created during initialization of the corresponding histogram by first randomly selecting $d^{-\frac{1}{2}}$ non-zero features (d is the dimension of the input space) and then randomly generating non-zero items according to $N(0, 1)$. The choice of normal distribution of non-zero items comes from the Johnson–Lindenstrauss (1984) lemma showing that with this choice, L_2 distances between points in the *projected space* approximate the same quantity in the *input space*. The $d^{-\frac{1}{2}}$ sparsity is due to Li (2007) showing that L_2 distances can be preserved with random projections with only $d^{-\frac{1}{2}}$ non-zero elements. Another justification for the use of sparse random projections is to increase diversity among histograms by making them work on different sub-subspaces. This is a popular diversification technique used for example in Random Forest (Ho 1998).

One can ask a question, are random projections actually needed? Would a detector based on an ensemble of histograms of individual features (in this section called per-feature detector) have similar accuracy? According to the experimental results on problems used throughout this paper (described in Sect. 4 in detail), random projections consistently improve the performance (see Fig. 10 in "Appendix"). On four out of five rate of anomalies Loda was better than the per-feature detector. Wilcoxon signed-rank test assessing if both detectors delivers the same performance was rejected with a p-value 0.028. Which means that the Loda is better and the difference is statistically significant. For more details see comments under Fig. 10.

3.2 Missing variables

Sparse projections enable Loda to handle missing variables by calculating the output only from those histograms whose projection vector has a zero on the place of missing variables. To formalize the approach, assume that $x \in \mathbb{R}^d$ is a sample with missing variables, and $\mathcal{J}$ is the index-set of missing variables. Let $\mathcal{I}(x)$ be a set of indices of histograms whose

projections have all entries in $\mathcal{J}$ zero, i.e. $(\forall i \in \mathcal{I}(x))\,(\forall j \in \mathcal{J})\,\left(w_{ij} = 0\right)$, where w_{ij} is jth element of the projection vector of ith histogram. Then, the anomaly score for the sample x is calculated as

$$f(x) = -\frac{1}{|\mathcal{I}(x)|} \sum_{i \in \mathcal{I}(x)} \log h_i(x). \tag{3}$$

Since the output of all histograms within Loda have the same meaning there is no need for calibration as in Nguyen et al. (2010). The final output is reliable even if several histograms are omitted. Notice that by using the same mechanism, Loda can be also trained on data with missing variables, as only detectors from $\mathcal{I}(x)$ are updated upon observing sample x.

3.3 Explaining the cause of an anomaly

Two independent works (Rondina et al. 2014; Somol et al. 2011) propose a feature selection method for a binary classification based on the comparison of scores (e.g. classification of error) of classifiers on randomly generated sub-spaces. Recognizing that each histogram with *sparse* projections in Loda provide an anomaly score on a randomly generated sub-space, this method can be used to rank features according to theirs contribution to sample's anomalousness.

Let $\hat{p}_i$ denote the probability estimated by ith histogram on a sample x, and $\mathcal{I}_j / \overline{\mathcal{I}}_j$ denote index sets of histograms that use/do not use jth feature. Formally $(\forall i \in \mathcal{I}(x))\,\left(w_{ij} = 0\right)$ and $\left(\forall i \in \overline{\mathcal{I}}(x)\right)\left(w_{ij} \neq 0\right)$, where w_i is the projection vector of ith histogram. The score function proposed in the prior art calculates the difference of means of $-\log \hat{p}_i$ over $\mathcal{I}_j$ and $\bar{\mathcal{I}}_j$, which means that if an anomaly score with a feature being present is much higher than that of without, the feature is important for identification of outliers. We improve this score function by recognizing that it's main goal is to assert if the contribution of jth feature is statistically significant, for which we use one-tailed two-sample t test with a test statistic

$$t_j = \frac{\mu_j - \bar{\mu}_j}{\sqrt{\frac{s_j^2}{|\mathcal{I}_j|} + \frac{\bar{s}_j^2}{|\bar{\mathcal{I}}_j|}}}, \tag{4}$$

where $\mu_j / \bar{\mu_j}$ is the mean and $s_j^2 / \bar{s}_j^2$ variance of $-\log \hat{p}_i$ calculated with $i \in \mathcal{I}_j / i \in \bar{\mathcal{I}}_j$. The higher the t_j the more important the j^{th} feature is. Since the complexity of the contrast function (4) is linear with respect to the number of projections k and number of features d, the calculation of a feature's contrast increases Loda's complexity by a constant in big O notation. Experiments demonstrating the feature selection are presented in Sect. 4.6.

3.4 Histogram

The one-dimensional histogram on random projections is an important design element in Loda, as it determines its learning mode (batch vs. on-line). The prior art on histogram construction and the optimal number of bins is very rich. In the following only the tiny subset relevant to on-line construction on streams is recapitulated.

The most common approach for batch data are equi-width or equi-depth histograms with bins having either the same length or containing the same number of samples. In database research, V-Optimal histograms (Poosala et al. 1996) minimizing weighted variance of estimated frequency are popular, but their construction has prohibitive complexity $O(n^2)$. Moving to data-steams, approximations of V-Optimal histograms have been studied in Guha et al. (2006), but the algorithms therein are quite complex. Simpler Partition Incremental

Discretization (Gama and Pinto 2006) constructs a histogram in two steps: (i) create an equi-width histogram with small bins in a single iteration through the data; (ii) use this fine histogram to return either equi-width or equi-depth histogram with a given number of bins. The advantage is that user-provided partition in step (i) is only indicative, as bins can be split if they already contain too many samples. The disadvantage for Loda is that the second step needs to be triggered before classifying a sample if the fine histogram was updated. DADO algorithm (Donjerkovic et al. 2000) constructs incremental histograms close in quality to V-optimal histograms. The key idea behind this technique is to internally represent every bin by two sub-bins. Counts in sub-bins are used for bin-split and bin-merge operations and for their triggering. DADO's biggest advantage for Loda is a fixed number of buckets which implies fixed memory requirements. An interesting alternative was proposed in Ben-Haim and Tom-Tov (2010) originally for determining splitting points in decision trees. Unlike all previous approaches it does not require any knowledge about the range of values in histogram (see Appendix "On-line histogram" for its recapitulation), but according to our results Loda with equi-width histogram is better.

With the exception of the last algorithm, all above approaches require knowledge of the range of modeled values in advance. This can be usually estimated from a sample of data, but in the case of severe non-stationarity, the histogram's support can shift outside the initial range. A simple solution for equi-width histograms is to specify bin width and store bin counts in a hash-map structure with keys $\kappa = \left\lfloor \frac{x}{\Delta} \right\rfloor$, where κ is an integer key, Δ is the bin width, and x is the value to be inserted. This approach has $O(1)$ time complexity and if coupled with count-min-sketch (Cormode and Muthukrishnan 2005) its space complexity can be upper bounded. It also allows the modeling fixed length windows in a stream by keeping samples (indexes of bins) to be later removed in memory. This can have potentially large memory footprint, which can be decreased by using exponential buckets proposed in Datar et al. (2002). Alternatively, two histograms as in HS-Trees (Tan et al. 2011) can be used, where the older one is used for classification while the newer one is being constructed on newly arrived samples. Once the construction of the new one is finished, it replaces the old one and the construction of the new one is started. Both constructions are experimentally compared in Sect. 4.2 with interesting conclusions.

The determination of an optimal number of histogram bins b is an important design parameter. For an *equi-width histogram*, there exists a simple method to determine optimal number of histograms (Birgé and Rozenholc 2006), which is used in Loda. The method of Birgé and Rozenholc (2006) maximizes penalized maximum likelihood in the form

$$\sum_{i=1}^{b} n_i \log \frac{b n_i}{N} - \left[b - 1 + (\log b)^{2.5} \right],$$

where n_i is the number of samples that falls in ith bin and $N = \sum_{i=1}^{b} n_i$ is the total number of samples. Penalization factor $b - 1 + (\log b)^{2.5}$ penalizes histograms with too many bins.

3.5 Computational and storage complexity

Loda's complexity is mainly determined by the type of the histogram. Assuming equi-width histogram and sparse random projections the time complexity of learning is $O(nkd^{-\frac{1}{2}})$, where n is the number of training samples, d is the dimensionality of the input space, and k is the number of histograms. The time complexity of classification is $O(kd^{-\frac{1}{2}})$. As discussed

Table 1 Time and space complexity of the anomaly detection algorithms compared in this paper

	Time complexity		Space complexity
	Batch training	Classification	
Batch			
FRAC	$n \cdot d^4$	$O(d^2)$	$d^2 b$
KNN	$O(1)$	$O(nd)$	$O(nd)$
LOF	$O(1)$	$O(knd)$	$O(nd)$
PCA	$O(nd^4)$	$O(kd)$	$O(kd)$
1-SVM	$O(n^2 d) \sim O(n^3 d)$	$O(nd)$	$O(nd)$
IForest	$O(kl \log l)$	$O(k \log l)$	$O(kl)$
On-line			
HS-Trees	–	$O(k(h+l))$	$O(kh^2)$
Loda two hist.	$O(nkd^{-\frac{1}{2}})$	$O(k(d^{-\frac{1}{2}}+b))$	$O(k(d^{-\frac{1}{2}}+b))$
Loda cont.	$O(nkd^{-\frac{1}{2}})$	$O(kd^{-\frac{1}{2}})$	$O(k(d^{-\frac{1}{2}}+b+l))$

The classification times for on-line detectors (HS-Trees, Loda with continuous histograms and Loda with two alternating histograms) include time to update the detector. The time to train Loda on batch data is included for comparison purposes. n is the number of samples in the training set, d is the number of features, k in LOF is the number of nearest-neighbor points, k in PCA is the number of components retained after the projection, k in IForest, HS-Trees, and Loda is the number of trees or histograms, h in HS-Trees is a maximal height of the tree, l in IForest is the number of samples used to construct one tree, l in HS-Trees and Loda with continuous histogram is the length of observation window, and finally b in FRAC and Loda is the number of histogram bins. The FRAC implementation assumes ordinary least-square regression as features predictors. "Loda cont." denotes Loda with naïve implementation of continuously updated histogram. "Loda two hist." denotes Loda with two alternating histograms

in the previous sub-section the space complexity can be made $O(k(d^{-\frac{1}{2}}+b))$ with b being number of histogram bins.

For a floating window histogram over a length of l the space complexity can be made $O(k(d^{-\frac{1}{2}}+b\log l))$ by using the algorithm of Datar et al. (2002) if space is constrained, otherwise $O(k(d^{-\frac{1}{2}}+b+l))$ for a naïve approach storing all l values for discounting them on expiration.

Alternatively, on-line histogram can be implemented by using two alternating histograms as in Tan et al. (2011), where the older histogram is used for the classification while the new one is constructed. Once the construction of the new one is finished (it has accommodated l samples), it replaces the older one for the classification and the new construction of the new histogram one is started. The advantage is possibly smaller space complexity $O(k(d^{-\frac{1}{2}}+b))$ and according to the experimental results in Sect. 4.2 better robustness to clustered anomalies. Table 1 shows the time and space complexity of Loda with continuous and two equi-width alternating histograms and the prior art used in the experimental section.

4 Experiments

Below Loda is compared to relevant state of the art in three different scenarios. The first scenario simulates anomaly detection on stationary problems, where all detectors are first

trained on a training set and then evaluated on a separate testing set. This scenario is included because it enables the comparison of Loda to other anomaly detectors, which are designed for batch training and their accuracy should be superior due to more sophisticated training algorithms. The second scenario mimics streamed data, where anomaly detectors first classify a sample and then use it to update their model. The third scenario uses a dataset with millions of samples and features to demonstrate that Loda is able to efficiently handle big data. The section is concluded by an experimental comparison of detectors time complexity in classify and update scenarios and by the demonstration of robustness against missing variables and the identification of causes of anomalies.

4.1 Stationary data

Loda has been compared to the following anomaly detectors chosen to represent different approaches to anomaly detection: PCA based anomaly detector (Shyu et al. 2003), 1-SVM (Schölkopf et al. 2001), FRAC (Noto et al. 2012), δ-version of kth-nearest neighbor detector (Harmeling et al. 2006) (KNN) which is equivalent to Sricharan and Hero (2011) with $\gamma = 1$ and $s = d$ (in notation of the referenced work) if the area under ROC curve (AUC) is used for evaluation, LOF (Breunig et al. 2000), and IForest (Liu et al. 2008).

Benchmarking data were constructed by the methodology proposed in Emmott et al. (2013) converting real-world datasets (Frank and Asuncion 2010) to anomaly detection problems. The methodology produces set of normal and anomalous samples, where samples from different classes have different probability distribution, which is aligned with our definition of anomalous samples. Created problems are divided according to (i) difficulty of finding an anomaly (*easy*, *medium*, *hard*, *very hard*), (ii) scatter / clusteredness of the anomalies (*high scatter*, *medium scatter*, *low scatter*, *low clusteredness*, *medium clusteredness*, *high clusteredness*), (iii) and finally with respect to the rate of anomalies within the data {0, 0.005, 0.01, 0.05, 0.1}. The training and testing set were created such that they have the same properties (difficulty, scatter/clusteredness, and rate of anomalies)[2] with clustered anomalies located around the same point. Note that the total number of unique combinations of problem's properties is up to 120 for each dataset. The construction of individual problems is recapitulated in Appendix "Construction of datasets".

The quality of the detection is measured by the usual area under ROC curve (AUC). Since from every dataset (out of 36) up to 120 problems with different combinations of problem's properties can be derived, it is impossible to present all 7×4200 AUC values. Tables with AUCs averaged over difficulty of detecting anomaly are in the supplemental and cover 15 pages. They are therefore presented in an aggregated form by using critical difference diagrams (Demšar 2006) which show average rank of detectors together with an interval in which Bonferroni–Dunn correction of Wilcoxon signed ranks test cannot reject the hypothesis that the detectors within the interval have the same performance as the best one. The average rank comparing AUC is calculated over all datasets on which all detectors provided the output, following Demšar (2006) lower rank is better. AUCs used for ranking on each dataset are an average over all combinations of problem parameters with fixed parameter(s) of interest. For example, to compare detectors on different rates of anomalies within the data, for each rate of anomalies the average AUC is calculated over all combinations of difficulty and scatter. Average AUCs of different detectors on the same dataset are ranked, and the average of ranks over datasets is the final average rank of a detector on one rate of anomalies within the data.

[2] The dataset with no anomalies contained no anomalies in the training data and 10 % of anomalies in the evaluation data, which captures the case the case when the anomaly detector is trained on clean data.

Table 2 Summary of detectors hyper-parameters used in experiments on stationary data in Sect. 4.1.2

Detector	Hyper-parameters
FRAC	–
PCA	Used all components with eigenvalues greater than 0.01
KNN	$k = \max\{10, 0.03 \times n\}$
LOF	$k = \max\{10, 0.03 \times n\}$
1-SVM	$\nu = 0.05$, $\gamma =$ inverse of median of L_2 distances of samples
IForest	100 trees each constructed with 256 samples
Loda	All hyper-parameters optimized automatically

To decrease the noise in the data caused by statistical variations, experiments for every combination of problem properties were repeated 100 times. This means that every detector was evaluated on up to $100 \times 120 \times 35 = 4.32 \times 10^5$ problems. Benchmark datasets created as described in Appendix "Construction of datasets" were varied between repetitions by randomly selecting 50 % of normal samples (but maximum of 10,000) to be in the training set and putting the remaining data to the set. Similarly, anomalous samples were first selected randomly from all anomalous samples to reach the desired fraction of outliers within the data, and then randomly divided between training and testing set. The data in the training and testing set have the same properties in terms of difficulty, clusteredness, and fraction of anomalies within.

4.1.1 Settings of hyper-parameters

Setting hyper-parameters in anomaly detection is generally a difficult unsolved problem unless there is a validation ground truth available for the parameter tuning. Wrong parameters can cause an otherwise well designed anomaly detector to fail miserably. Nevertheless, in this context we do not aim to solve the problem as our intention here is primarily to compare multiple detectors against each other. For this purpose we follow hyper-parameter setting guidelines given by the authors of the respective methods. Note that our proposed method does not require manual hyper-parameter setting. Employed parameter settings are detailed below and in Table 2.

Settings of the number of nearest neighbors in our implementations of LOF and KNN algorithms followed (Emmott et al. 2013; Breunig et al. 2000) and was set to $\max\{10, 0.03 \times n\}$, where n is the number of samples in the training data. 1-SVM with a Gaussian kernel used $\nu = 0.05$ and width of the kernel γ equal to an inverse of median L_2 distance between training data. SVM implementation has been taken from the libSVM library (Chang and Lin 2011). Our implementation of the FRAC detector used ordinary linear least-square estimators, which in this setting does not have any hyper-parameters. Our implementation of PCA detector based on principal component transformation used top k components capturing more than 95 % variance. IForest [implementation taken from Jonathan et al. (2014)] used parameters recommended in Liu et al. (2008): 100 trees each constructed with 256 randomly selected samples.

The number of histograms in Loda, k, was determined similarly to the number of probes in feature selection in Somol et al. (2013). Denoting $f_k(x)$ Loda's output with k histograms, the reduction of variance after adding another histogram can be estimated as

$$\hat{\sigma}_k = \frac{1}{n}\sum_{i=1}^{n} |f_{k+1}(x_i) - f_k(x_i)| .$$

Although $\hat{\sigma}_k \to 0$ as $k \to \infty$, its magnitude is problem-dependent making it difficult to set a threshold on this quantity. Therefore $\hat{\sigma}_k$ is normalized by $\hat{\sigma}_1$, and the k is determined as

$$\arg\min_k \frac{\hat{\sigma}_k}{\hat{\sigma}_1} \geq \tau,$$

where τ is the threshold. In all experiments presented in this paper, τ was set to 0.01. Unless said otherwise, Loda used equi-width histogram with the number of histogram bins, b, determined for each histogram separately by maximizing penalized maximum likelihood method of Birgé and Rozenholc (2006) described briefly in Sect. 3.4. With b being set, width of histogram bins in equi-width histograms was set to $\frac{1}{b}(x_{\max} - x_{\min})$.

Setting of hyper-parameters is summarized in Table 2.

4.1.2 Experimental results

Figure 1a–e shows the average rank of detectors plotted against rate of anomalies within the training set for different clusteredness of anomalies (AUCs were averaged only with respect to the difficulty of anomalies). Since highly scattered anomalies could not be produced from the used datasets, the corresponding plot is omitted. The number of datasets creating problems with medium scattered anomalies is also low, and we should not draw any conclusions as they would not be statistically sound.

For problems with low number of low-scattered and low-clustered anomalies the data-centered detectors (KNN and LOF) are good. With increasing number of anomalies in data and with their increasing clusteredness, however, the performance of data-centered detectors appears to deteriorate (note LOF in particular). Additional investigation revealed that this effect can be mitigated by hyper-parameter tweaking, specifically by increasing the number of neighbours k. In practice this is a significant drawback as there is no way to optimize k unless labeled validation data is available.

The model-centered detectors (Loda, IForest as well as 1-SVM) appear to be comparatively more robust with respect to the increase in number of anomalies or their clusteredness, even if used with fixed parameters (see Sect. 4.1.1.).

In terms of statistical hypothesis testing the experiments do not show marked differences between the detectors in question. Note that in isolated cases FRAC and 1-SVM perform statistically worse than the respective best detector.

Figure 1f shows the detectors time to train and classify samples with respect to the size of the problem measured by the dimension multiplied by the number of samples.[3] On small problems KNN, LOF, and 1-SVM are very fast, as their complexity is mainly determined by the number of samples, but Loda is not left behind too much. As the size of problems gets bigger Loda quickly starts to dominate all detectors followed by PCA. Surprisingly IForest with low theoretical complexity had the highest running times. Since its running time is almost independent of the problem size, it is probably due to large multiplication constant absorbed in big O notation.

Comparing detectors by the performance/time complexity ratio, we recommend KNN for small problems with scattered anomalies, while for bigger problems and problems with

[3] Because all algorithms have not used any search for hyper-parameters, Loda used fixed number of histogram being equal to $\sqrt{n}$, where n is the number samples.

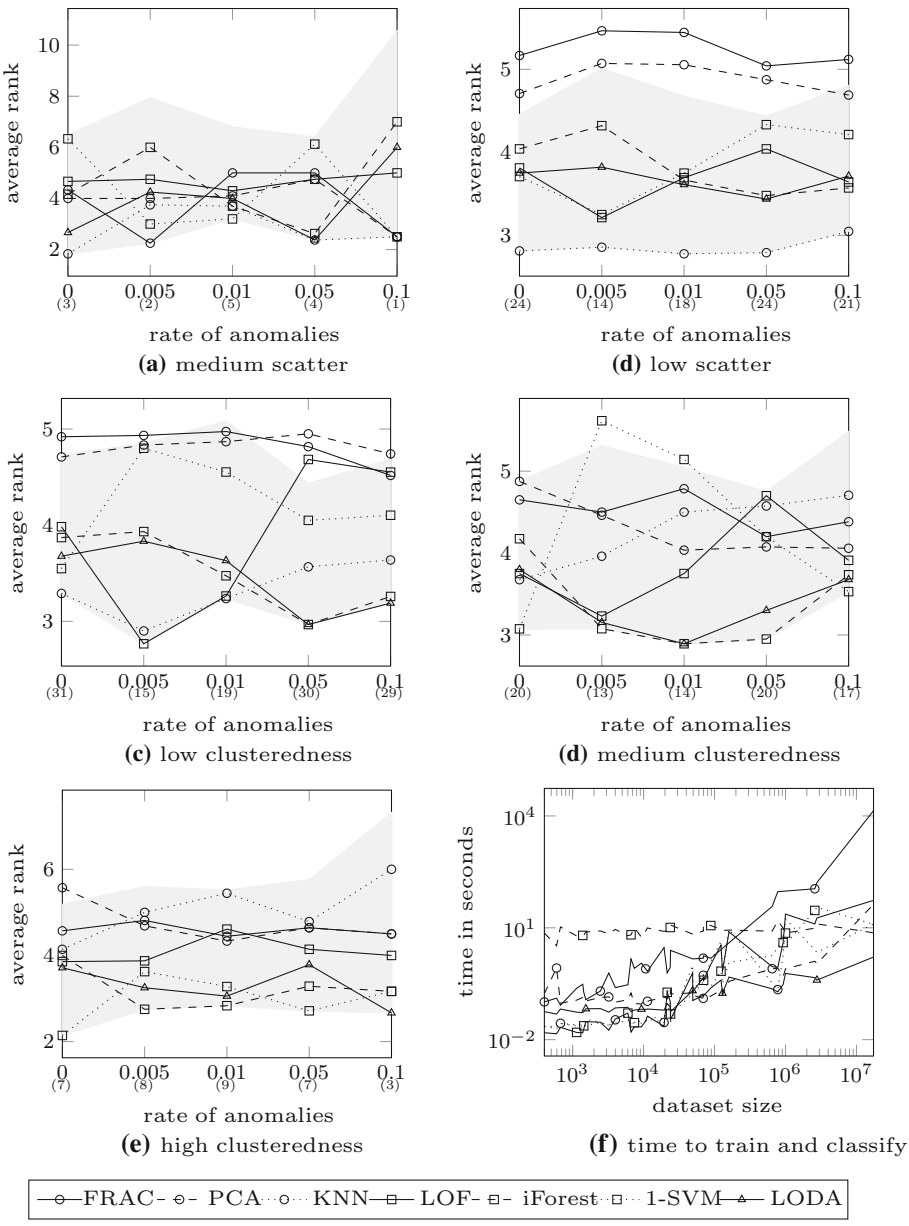
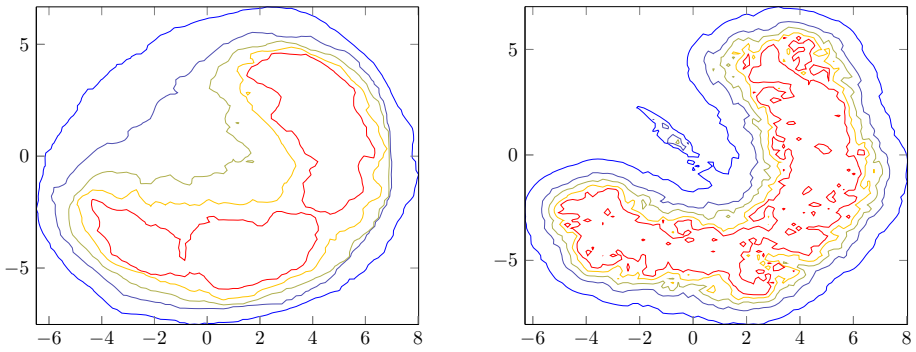


Fig. 1 **a–e** The average rank of detectors with respect to the rate of anomalies within the training data for various levels of clusteredness. The average rank is calculated by averaging rank of detectors over datasets, where the rank of detectors on a single dataset compares AUCs. The *small number in parentheses* shows the number of datasets from which benchmarking problems with a given combination of clusteredness and rate of anomalies were created. Critical difference is shown as a *light grey*—it is the area in which a in which Bonferroni–Dunn correction of Wilcoxon signed ranks test cannot reject the hypothesis that the detectors within has the same performance as the best detector. **f** The time to train the detector and classify all testing samples



(a) Loda **(b)** k-nearest neighbor

Fig. 2 Isolines at false positive rates {0.01, 0.05, 0.1, 0.2, 0.4} of Loda and k-nearest neighbor detector on a banana dataset, **a** Loda, **b** k-nearest neighbor

clustered anomalies we recommend the proposed Loda, as its performance is similar to IForest but it is much faster.

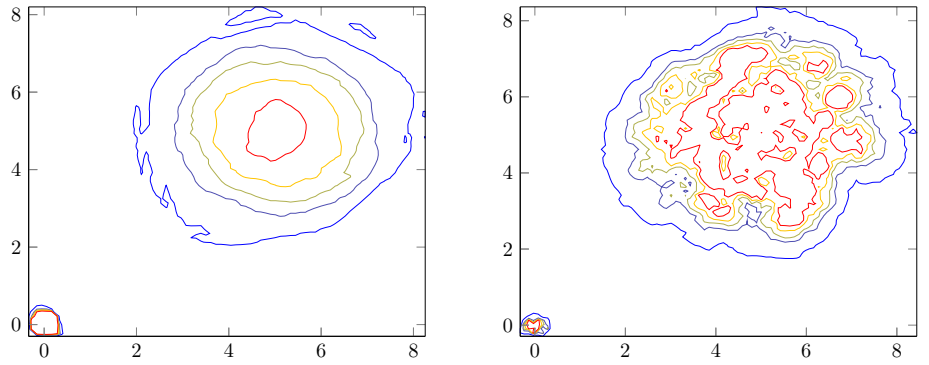
The investigation of problems on which Loda is markedly worse than KNN revealed that Loda performs poorly in cases where the support of the probability distribution of nominal class is not convex and it encapsulates the support of the probability distribution of the anomalous class. A typical example is a "banana" dataset in Fig. 2 showing isolines at false positive rates {0.01, 0.05, 0.1, 0.2, 0.4} of Loda and KNN. Isolines show that Loda has difficulty modeling the bay inside the arc. Although the adopted construction of problems from UCI datasets (see Appendix "Construction of datasets" for details) aimed to create these difficult datasets, Loda's performance was overall competitive to other, more difficult detectors.

Breunig et al. (2000) has criticized KNN, which has performed the best on clean training data, for its inability to detect local outliers occurring in datasets with areas of very different densities. Figure 3 shows again isolines at false positive rates {0.01, 0.05, 0.1, 0.2, 0.4} of Loda and LOF (designed to detect local outliers) on a data generated by two Gaussians with very different standard deviations (a typical example of such data). Loda's isolines around the small compact cluster (lower left) are very close to each other, whereas isolines of the loose cluster (upper right) are further away which demonstrates that Loda is able do detect local outliers similarly to LOF. LOF's wiggly isolines also demonstrates the over-fitting to the training data.

Figure 4 visualizes the average rank of Loda with equi-width, equi-depth, and on-line (Ben-Haim and Tom-Tov 2010) histograms on different rates of anomalies within the data. Although on almost all rates Friedman's statistical test accepted the hypothesis that all three versions of Loda performs the same, Loda with equi-width histograms is obviously better. This is important because equi-width histograms can be efficiently constructed from a stream of data if the bin-width is determined, for example by using sample data.

4.2 Streamed and non-stationary data

This section compares Loda with floating-window histogram and two alternating histograms (see Sect. 3.4) to Half-Space Trees (HS-Trees) (Tan et al. 2011) [implementation taken from Tan (2014)]. On-line version of 1-SVM was omitted from the comparison due to the difficulties with hyper-parameter setting and inferior performance when algorithms were



(a) Loda on Two Gaussians **(b)** Local outlier factor on Two Gaussians

Fig. 3 Isolines at false positive rates {0.01, 0.05, 0.1, 0.2, 0.4} of Loda and Local Outlier Factor on a dataset composed of two Gaussians with different SD, **a** Loda on Two Gaussians, **b** Local outlier factor on Two Gaussians

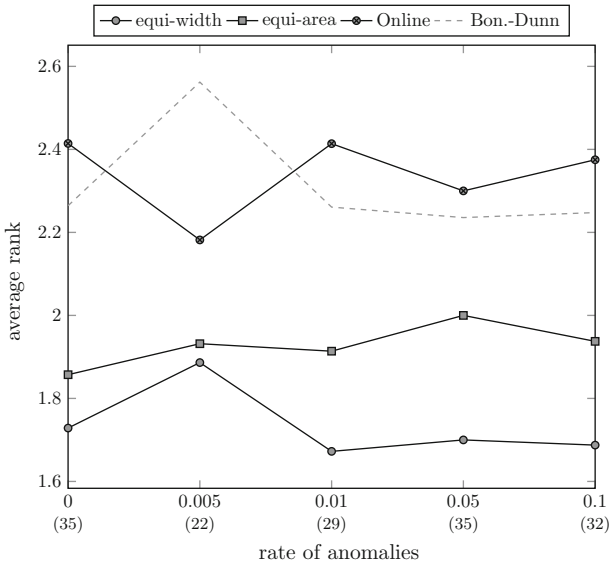


Fig. 4 Average rank of Loda with different types of histograms with respect to the rate of anomalies within the training data. The rank of three compared detectors is calculated for each problem separately, and then the average over all problems with a given rate of anomalies in the training data is shown in the *graph*

compared in the batch learning. The comparison is made on seven datasets: Shuttle, Covertype, HTTP and SMTP data from kdd-cup 99. HTTP and SMTP data are used in two variants: (i) with all 41 features (called HTTP-full and SMTP-full) and (ii) with 3 features (called HTTP-3 and SMTP-3) used in Tan et al. (2011).[4]

In every repetition of an experiment the testing data were created by replacing maximum of 1 % of randomly selected normal samples by randomly selected anomalous samples. Order

[4] The original publication (Tan et al. 2011) does not specify which three features from kdd-cup dataset were used, but datasets used in the publication are available at Tan (2014).

Table 3 AUCs of Loda and HS-Trees (Tan et al. 2011) on streaming problems with various levels of clusteredness

	Dataset	Continuous		Two histograms		HS-Trees	
		AUC	Time	AUC	Time	AUC	Time
Clusteredness = 1	Covertype	**0.987**	4.36	0.985	3.04	0.980	19.00
	http-3	**0.995**	7.42	0.989	5.91	0.993	37.82
	http	**0.998**	7.84	0.983	5.68	0.995	35.30
	shuttle	0.992	0.54	0.992	0.39	**0.999**	4.90
	smtp	0.994	1.25	0.989	1.01	**0.998**	8.25
	smtp-3	0.890	1.52	0.883	1.27	**0.915**	8.38
	smtp+http	0.992	6.51	0.987	5.88	**0.998**	42.56
Clusteredness = 4	Covertype	0.972	4.42	**0.989**	3.00	0.980	19.05
	http-3	0.992	7.51	**0.994**	5.24	0.983	40.17
	http	0.991	8.40	**0.993**	6.00	0.990	34.87
	shuttle	0.980	0.49	0.994	0.41	**0.999**	5.02
	smtp	0.970	1.34	0.994	1.06	**0.996**	8.58
	smtp-3	0.871	1.35	0.886	1.11	**0.914**	8.31
	smtp+http	0.989	9.65	0.993	7.99	**0.998**	42.70
Clusteredness = 16	Covertype	0.937	4.42	**0.989**	3.01	0.980	19.33
	http-3	0.987	6.71	**0.996**	4.95	0.980	40.84
	http	0.982	8.43	**0.993**	6.27	0.987	34.52
	shuttle	0.943	0.51	0.994	0.42	**0.999**	5.12
	smtp	0.947	1.34	0.993	1.06	**0.995**	8.63
	smtp-3	0.863	1.38	0.882	1.17	**0.915**	8.35
	smtp+http	0.983	7.99	0.995	4.80	**0.998**	42.20

Bold-faced values are the best values for a given dataset and clusterdness

of normal samples has not been permuted in order to preserve data continuity. Thus, the variation between repetitions comes from (i) the position to where anomalous samples were inserted and (ii) from the selection of anomalous samples to be mixed in. To study the effect of clustered anomalies, anomalies were first selected to form clusters in space (as in the previous section) and then they replaced sequences of normal data of the same size. By this process data-streams contained anomalies clustered in time and space of sizes {1, 4, 16} (this quantity is hereafter called clusteredness). Experiments on each dataset for every value of clusteredness of anomalies were repeated 100 times.

HS-Trees used recommended configuration of 50 trees of depth 15 with minimum of 20 samples in leaf node. Loda's configuration, namely number of histograms and bins were optimized on first 256 samples in every repetition, thus there were no parameters to be set manually. Both algorithms used a window of length 256 samples, which is the hard-coded length in implementation of HS-Trees provided by authors.

Table 3 shows AUCs and average time to process samples of Loda and that of HS-Trees. We can see that both algorithms deliver nearly the same performance, while Loda is on average 7–8 times faster (the time complexity is treated in more detail in the next subsection). Figure 5 shows AUCs on segments of 120,000 continuous samples on datasets Covertype, HTTP, HTTP-3, and SMTP+ HTTP with clusters of 16 anomalies (omitted datasets did not

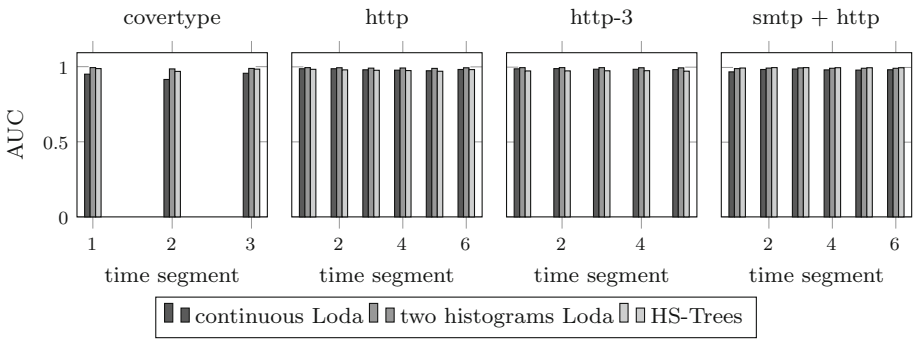


Fig. 5 AUCs of Loda with continuous histogram, Loda with two alternating histograms, and HS-Trees on segments of continuous 12,000 samples from the stream. AUCs in every segment is an average from 100 repetitions

contained enough samples). Since all AUCs of all three detectors remains stable, it can be concluded that all detectors equally well combat the concept drift in datasets.

Further investigation of results reveals that Loda with two alternating histograms (construction similar to that in HS-Trees) is more robust to clustered anomalies than Loda with continuous histogram. Our explanation of this phenomenon is that clustered anomalies are classified most of the time by histograms built on clean data without anomalies, while polluted histograms classify most of the time clean data, which is not influenced by anomalies in training data. These results demonstrate that the right choice of histogram for non-stationary data should depend on the type of anomalies (time and space clustered vs. scattered).

4.3 Big data

The demonstration of Loda's ease of handling big data is done on a URL dataset (Ma et al. 2009) containing 2.4 million samples with 3.2 million features. Normal samples contain features extracted from random URLs, while anomalous samples have features extracted from links in spam e-mails during 120 days at rate 20,000 samples per day (see Ma et al. 2009 for dataset details). Due to the high number of "anomalous" samples and cleanliness of normal samples, the dataset belongs to a category of one-class problems where the goal is to separate class of legitimate URLs from others.

Since the data are presented in batches of 20,000 samples per day, all methods evaluated in Sect. 4.1 can be theoretically applied by training them on normal samples from the previous day, i.e. detectors classifying samples from lth day are trained on normal samples from the $(l-1)$th day. The problem is that the computational complexity effectively prevents a direct application of FRAC, 1-SVMs, PCA, LOF, and KNN, as methods would not finish in reasonable time. The only methods that can be used without modification is Loda and IForest. Their settings were same as in Sect. 4.1 with the difference that (i) Loda used 500 projections and histograms bins were optimized on data from day zero and then stay fixed; (ii) number of trees within IForest was increased to 1000. Rows labeled 3.2×10^6 in Table 4 show AUCs, average rank over 120 days, and the average time to classify new samples and retrain the detectors. We can see that Loda was not only significantly better (AUC of IForest is close to the detector answering randomly), but also more than 15 times faster.

To reduce the computational complexity such that other algorithms can be used, the original data of dimension 3.2 million were projected onto 500 dimensional space with a randomly

Table 4 Average rank (caption rank), average AUC (caption AUC), and average time to classify and update detectors on 20 000 samples (caption time) on the full problem ($d = 3.2 \times 10^6$) and on the reduced problem ($d = 500$)

d	Detector	Rank	AUC	Time
500	KNN	1.596	0.818	140.5s
	PCA	2.758	0.803	1.4s
	Loda	3.067	0.795	1.16s
	FRAC	3.958	0.789	283.8s
	IForest	4.733	0.776	6.4s
	LOF	4.987	0.771	307.5s
	1-SVM	6.900	0.528	711.4s
3.2×10^6	Loda	1.025	0.795	21.6s
	IForest	1.975	0.488	369s

The average rank is an average of detector's rank over 120 days

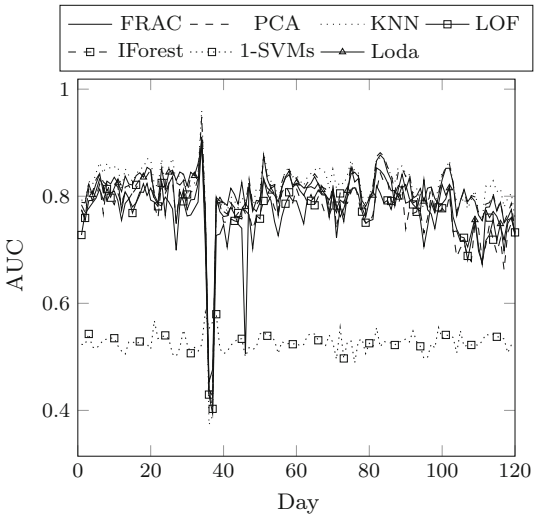


Fig. 6 AUCs of detectors on URL dataset with respect to time. AUCs of PCA, KNN, IForest, and 1-SVMs is on dataset projected to 500 dimension. AUCs of Loda are on the full dataset of dimension 3.2 million

generated matrix $W \in \mathbb{R}^{3.2\times10^6,500}$, $W_{ij} \sim N(0, 1)$. According to the Johnson-Lindenstrauss lemma, L_2 distances between points should be approximately preserved, therefore most detectors should work. Hyper-parameters of all detectors were set as in Sect. 4.1.1. AUCs of all detectors on every day are shown in Fig. 6, and their summaries again in Table 4. On this new dataset the best algorithm was the KNN algorithm followed by PCA and Loda. These results are consistent with those in Sect. 4.1.2, where KNN detector was generally good if trained on clean data. The good performance of PCA can be explained by projected data being close to multivariate normal distribution, although the Lilliefors' test has rejected the hypothesis that marginals follow the normal distribution with 85 % rate. Notice that Loda's execution time was more than 100 times faster than KNN, as its time to classify and update on samples from one day was 1.16s on average, while that of KNN was 140s.

4.4 Time complexity

Loda was primarily designed to have low computational complexity in order to efficiently process large data and data-streams. This feature is demonstrated below, where times to

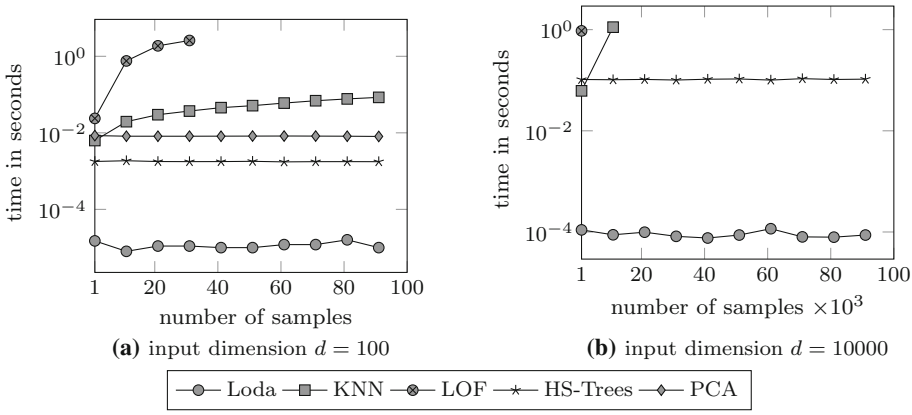


Fig. 7 Average time of classification and update of PCA, KNN, LOF, and Loda detectors against number of observed samples. *Left* and *right* figures show times for the problem of dimension 100 and 10,000 respectively

classify one sample with simultaneous update of the detector is compared for PCA, KNN, LOF, HS-Trees, and Loda with continuous histogram.[5] All settings of individual algorithms were kept the same as in previous experiments. Loda's time includes optimization of hyper-parameters on the first 256 samples of the stream as described in Sect. 4.1.1 (Loda used on average 140 projections with histograms having 16 bins and sliding window of length 256).

The comparison was made on 100,000 artificially generated samples of dimensions 100 and 10,000. As the interest here is on throughput, the details of data generation are skipped as not being relevant for the experiment. The results are summarized in Fig. 7 showing the average time to classify and update one sample against the number of observed samples. The experiment was stopped if the processing of 1000 samples took more than half an hour causing some graphs to be incomplete. The graph for the PCA detector is missing entirely in the left Fig. 7, because processing of 1000 samples took more than 15 h (109s per one sample). Both graphs demonstrate that Loda's efficiency is superior to all other detectors, as it is two order of magnitudes faster than the fastest algorithms from the prior art. This experiment highlights Loda's suitability for efficiently handling data-streams.

4.5 Robustness to missing variables

As mentioned in Sect. 3, Loda with sparse projections can classify and learn from samples with missing variables. This feature was evaluated under the "missing at random" scenario, where missing variables are independent of the class membership. In every repetition of an experiment, before the data was divided into training and testing sets, $\{0, 0.01, 0.02, 0.04, \ldots, 0.20\}$ fraction of all variables in data matrices were made missing (set to `NaN`). Consequently, missing variables were equally present in sets on which Loda were trained and evaluated. The robustness was evaluated on datasets created in the same way as in Sect. 4.1 with clean training sets and 10 % of anomalies in the test sets. Loda's hyper-parameters were determined automatically as described in Sect. 4.1.1.

The effect of missing features on detector's AUC is summarized in Fig. 8 showing the highest missing rate at which Loda had the AUC higher than 99 % of that on the data without

[5] SVM and FRAC were omitted, as their on-line adaptation is not straight-forward. Although the implementation of HS-Trees is in Java, the binary (jar) reports time to process samples, which was used in this comparison.

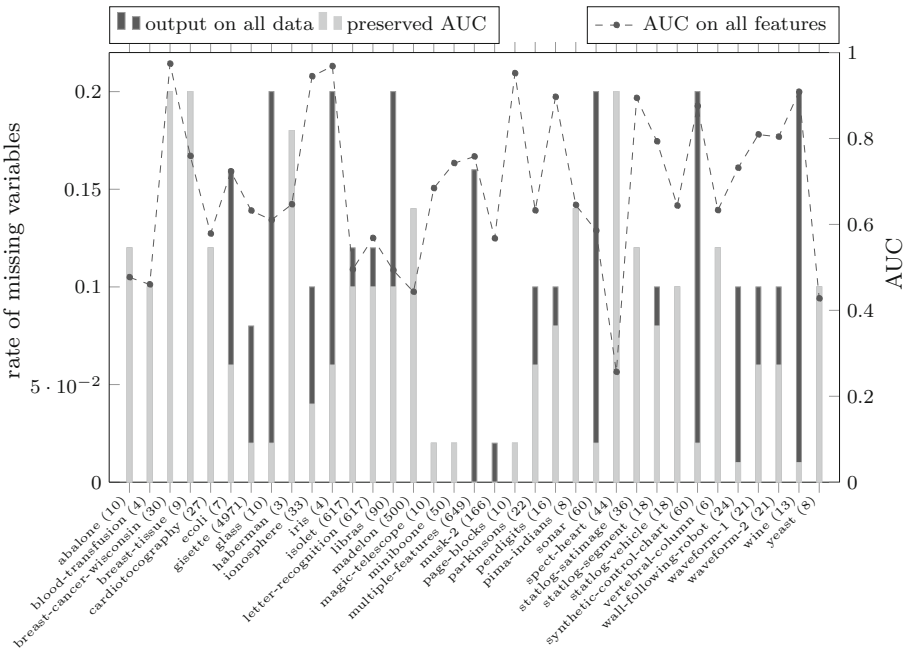


Fig. 8 *Light grey bars* show the maximum rate of missing variables at which Loda had AUC greater than $0.99 \times$ AUC on data without missing variables. *Dark grey bars* show the maximum rate of missing variables at which Loda provided answers for all samples. If *dark grey* are covered by *light grey bars*, then both maximum rates were equal. The *dashed line* shows AUC of Loda on data without missing variables

missing variables (light grey bars). For a large set of problems the tolerance is high and more than 10 % of missing variables are tolerated with negligible impact on the AUC. There are also exceptions, e.g. magic telescope, page-blocks, wine, etc, where the robustness is very small, but we have failed to find a single cause for this fragility. Interestingly, the robustness is independent on the dimension of the problem and the success with which outliers are detected. The exception is datasets on which the detection of outliers is poor (AUC is around 0.5), which is caused by the fact that missing some variables cannot generally improve or decrease already bad AUC. The same figure also shows the rate of missing variables at which Loda stops to provide output (dark grey bars). We can see that this rate is usually much higher than the rate at which Loda retains its AUC.

4.6 Identification of features responsible for an outlier

Explaining the causes of an anomaly detection is relatively new field and there is not yet an established methodology to evaluate and compare the quality of algorithms, hence, the following is adopted here. It starts by calculating the average of Loda's feature scores according to (4) over all anomalous samples in the testing set. Once the scores of individual features are known, they are sorted in decreasing order, which means that features in which anomalous samples deviates the most should be the first. Then, Loda with dense projections utilizing only first d' features is used and its AUC is recorded (d' has varied from 2 to $\min(100, d)$). If the feature relevance score (4) is meaningful, then Loda that uses first couple features (low d') should have the same or better AUC than Loda that uses all features. This is because all

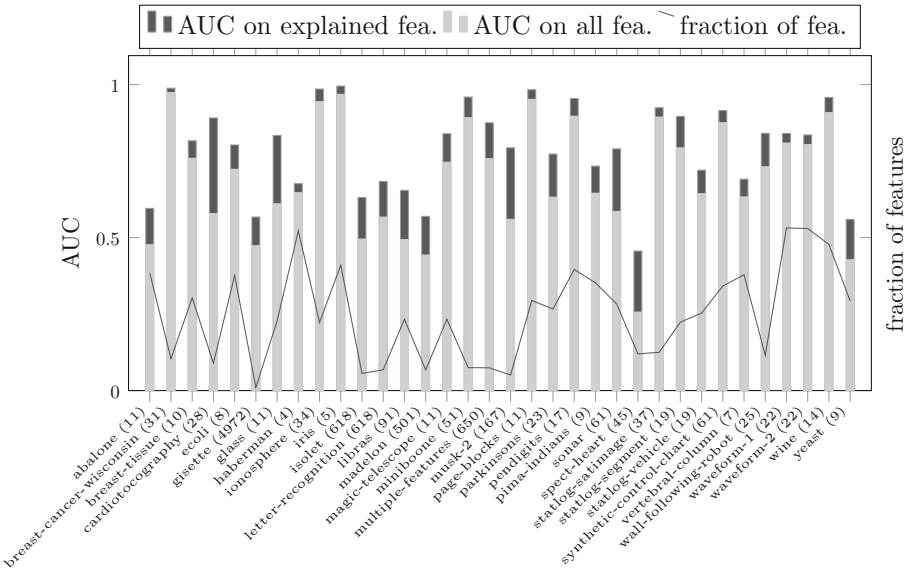


Fig. 9 AUC of Loda with dense projections on features selected according to relevance provided by Loda (captioned "AUC on explained fea.") and AUC of Loda with sparse projections on all features (caption "all"). The *curve* traversing *bars* shows the relative number of features selected according to Loda's relevance

testing problems were originally classification problems which means that anomalous samples were generated by the same probability and hence similar features should be responsible for their anomalousness. Experiments were performed on clean training datasets and all other experimental settings were kept same as in Sect. 4.1.

Figure 9 shows the average AUC of Loda with dense projections using the most important $d_{\min}$ features, where $d_{\min}$ is the least such that AUC by using only first $d_{\min}$ features is higher than 0.99 times the best AUC. On average only 25 % of features are needed to identify outliers with AUC comparable or better than that of Loda with all features. This result confirms that Loda is able to identify features causing the anomaly.

It is interesting that for some problems (e.g. cardiotocography, glass, spect-heart, yeast, etc.) the improvement in the AUC is significant. This can be explained by the curse of dimensionality as anomaly detectors are, by their nature, more sensitive to noise generated by non-informative features.

5 Conclusion

This paper has focused on creating a detector of anomalous samples being able to quickly process enormous amounts of data produced in many contemporary domains. Although the presented detector (Loda) was aimed to process streams of data with a constantly changing behavior, it has been compared to the state of the art in settings where all data are available at once and they can fit into the memory. The rationale behind was to compare Loda under various conditions differing by the rate of anomalies within the data, their clusteredness/scatterness to state of the art detectors, even though they are not applicable for intended scenario. This comparison, which scale is to our knowledge the biggest so far published has

shown that Loda's accuracy measured by area under ROC curve does not lag behind more sophisticated detectors. Moreover, it has revealed conditions under which particular type of detectors with the recommended setting of its parameters excels. Based on it we can conclude that data-centered detectors such as kth-nearest neighbor or Local Outlier Factor with recommended setting for k are good for problems with a few number of scattered anomalies. Contrary, for problems with clustered anomalies and/or with higher rate of anomalies within the data (more than 1 %), the model-centered detectors such as the proposed Loda, 1-SVM, and Isolation Forest are more suitable. Loda was also the only evaluated method capable to process data with 3.2 millions features in a reasonable time without any modification. Last but not least, it was shown that Loda is well suited for intended domains, where data are in the form of non-stationary streams with samples observed just once, classified, and the detector is updated. The comparison to state of the art Half-Space Trees method on seven problems of this type showed that both detectors have nearly the same detection performance, but Loda is on average 7–8 times faster.

Loda has practically demonstrated that an ensemble of anomaly detectors as weak as one-dimensional histograms yields to a strong anomaly detector. Loda's diversification of weak anomaly detectors relies on sparse random projections, which can be viewed as a random sub-space sampling. It is shown that this type of diversification gives the ensemble capability to explain why a scrutinized sample is anomalous and to be robust against missing values. A thorough evaluation showed that detectors employing only features explaining anomalies have equal or better performance than detectors using all features, which shows that features explaining anomalies were selected correctly (on average 25 % of features explained the anomaly). The similar evaluation showed that the ensemble keeps on most problems its detection performance even if more than 10 % of features are missing.

To conclude, this paper has shown that there is no single detector excelling on all type of problems, as different detectors are suitable for different types of problems. The biggest advantage of the proposed Loda with respect to others is its overall good performance and very good speed to detection performance ratio, as it is order of magnitude faster than prior art on big problems while its detection performance is comparable to them. Besides, it can explain causes of anomalies and it is robust to missing variables in data.

Appendix: On-line histogram

The on-line histogram from Ben-Haim and Tom-Tov (2010) approximates the distribution of data by using a set of pairs $\mathbf{H} = \{(z_1, m_1), \ldots, (z_b, m_b)\}$, where $z_i \in \mathbb{R}$ and $m_i \in \mathbb{N}$, where b is an upper bound on the number of histogram bins. It is assumed that every point z_i is surrounded by m_i points, of which half is to the left and half is to the right to z_i. Consequently, the number of points in the interval $[z_i, z_{i+1}]$ is equal to $\frac{m_i+m_{i+1}}{2}$, and the probability of point $z \in (z_i, z_{i+1})$ is estimated as a weighted average.

The construction of the set $\mathbf{H}$ is described in Algorithm 3. It starts with $\mathbf{H} = \{\}$ being an empty set. Upon receiving a sample, $z = x^{\mathrm{T}}w$, it looks if there is a pair (z_i, m_i) in $\mathbf{H}$ such that z is equal to z_i. If so, the corresponding count m_i is increased by one. If not, a new pair $(z, 1)$ is added to $\mathbf{H}$. If the size of $\mathbf{H}$ exceeds the maximal number of bins b, the algorithm finds the two closest pairs (z_i, m_i), (z_{i+1}, m_{i+1}), and replaces them with an interpolated pair $\left(\frac{z_i m_i + z_{i+1} m_{i+1}}{m_i + m_{i+1}}, m_i + m_{i+1}\right)$. Keeping z_i sorted makes all the above operations efficient.

The estimation of the probability density in point $z = x^{\mathrm{T}}w$ is described in Algorithm 4. Assuming the pairs in $\mathbf{H}$ are sorted according to z_i, the algorithm starts by finding i such

Algorithm 3: Algorithm constructing approximation of the probability distribution of the data $\{x_i \in \mathbb{R}^d\}_{i=1}^n$ projected on the vector $w \in \mathbb{R}^d$.

Input: sample $x \in \mathbb{R}^d$, $w \in \mathbb{R}^d$;
Output: set of pairs $\mathbf{H} = \{(z_1, m_1), \ldots, (z_b, m_b)\}$,

initialize $\mathbf{H} = \{\}$, $z_{\min} = +\infty$, $z_{\max} = -\infty$. ;
for $j \leftarrow 1$ **to** n **do**
  $z = x_j^{\mathrm{T}} w$;
  $z_{\min} = \min\{z_{\min}, z\}$;
  $z_{\max} = \min\{z_{\max}, z\}$;
  if $\exists(z_i == z)$ **then**
    $m_i = m_i + 1$;
    continue
  else
    $\mathbf{H} = \mathbf{H} \cup \{z, 1\}$
  end
  if $|\mathbf{H}| > b$ **then**
    Sort pairs in $\mathbf{H}$ such that $z_1 < z_2 < \ldots < z_{b+1}$;
    Find i minimizing $z_{i+1} - z_i$;
    Replace pairs (z_i, m_i), (z_{i+1}, m_{i+1}), by the pair

$$\left(\frac{z_i m_i + z_{i+1} m_{i+1}}{m_i + m_{i+1}}, m_i + m_{i+1}\right)$$

end
$\mathbf{H} = \mathbf{H} \cup \{(z_{\min}, 0), (z_{\max}, 0)\}$;
Sort pairs in $\mathbf{H}$ such that $z_{\min} < z_1 < z_2 < \ldots < z_{\max}$;

Algorithm 4: Algorithm returning approximate of probability density in point x projected on the vector w.

Input: sample $x \in \mathbb{R}^d$, $w \in \mathbb{R}^d$, set of pairs $\mathbf{H} = \{(z_1, m_1), \ldots, (z_b, m_b)\}$;
Output: return estimate of the probability density $p(x)$.;

$\mathbf{H} = \mathbf{H} \cup \{(z_{\min}, 0), (z_{\max}, 0)\}$;
Sort pairs in $\mathbf{H}$ such that $z_{\min} < z_1 < z_2 < \ldots < z_{\max}$;
$z = x^{\mathrm{T}} w$;
if $\exists(i | z_i < z \leq z_{i+1})$ **then**
  return $p(x) = \frac{z_i m_i + z_{i+1} m_{i+1}}{2M(z_{i+1} - z_i)}$;
else
  return invalid;
end

that $z_i < z \leq z_{i+1}$. If such i exists, then the density in z is estimated as $\frac{z_i m_i + z_{i+1} m_{i+1}}{2M(z_{i+1} - z_i)}$, where $M = \sum_{i=1}^b m_i$. Otherwise it is assumed that z is outside the estimated region.

Construction of datasets

Benchmark problems for the evaluation of batch anomaly detectors were constructed following the process in Emmott et al. (2013). It advocates the creation of anomaly detection benchmarks from real data, since artificial problems can be too far from realistic problems. 36 source datasets from which all benchmark problems have been created were downloaded

from Frank and Asuncion (2010) from a category of classification problems with numerical attributes and without missing variables. The list of downloaded datasets together with informations about the number of normal / anomalous samples is in Table 5

If the source dataset was a binary problem, the larger class was used as a normal class and the smaller as the anomaly class. Multi-class datasets were converted to binary ones as follows:

1. Train a random forest classifier to solve the multi-class problem in the original source dataset using all samples.
2. Estimate the confusion matrix $\mathbf{C}$ with $\mathbf{C}_{k,j} = P(j|x_i, k)$, where the conditional probability is the probability returned by random forest that the sample x_i belongs to the class j when k is true.
3. Create a complete graph with vertices being classes from the datasets and edges having weights from confusion matrix as $\mathbf{C}_{k,j} + \mathbf{C}_{j,k}$.
4. Compute the maximum weight spanning tree of the graph to identify "most-confusable" pairs of classes.
5. Two-color the maximum spanning tree such that no adjacent vertices has the same color. Each color determines set of classes that make the normal and anomalous class.

Notice that the above construction aims to maximize the difficulty (confusion) between normal and anomalous class.

Anomalous samples were divided into four groups according to the probability that a sample from the anomaly class is assigned to an anomaly class estimated the by the kernel logistic regression with a Gaussian kernel (Zhu and Hastie 2005). Based on this score, all anomalous points were assigned a difficulty category as: *easy* ([0.84, 1), *medium* ([0.7, 0.84), *hard* ([0.5, 0.7), *very hard* (0, 0.5].

A concrete benchmark problem with a given fraction of anomalies (experiments used fractions 0.005, 0.01, 0.05, 0.1) and difficulty (easy, medium, hard, very hard) was created as follows:

1. Randomly divide samples into training a testing set allowing maximum of 10000 samples in each set.
2. If anomalies should be clustered, select randomly a pivot and find sufficient number of nearest points in chosen difficulty category such that the final training and testing sets have the desired number of anomalous samples. If anomalies are not clustered, select anomalous samples randomly.
3. Divide the selected anomaly samples into training a testing set and mix with normal samples.

The final benchmark problem is attached a clusteredness category according to a fraction of sample variance of normal samples to the sample variance of anomalous samples as: *high scatter* (0, 0.25), *medium scatter* [0.25, 0.5), *low scatter* [0.5, 1), *low clusteredness* [1, 2), *medium clusteredness* [2, 4), and *high clusteredness* [4, ∞).

Created benchmark problems have three basic properties (rate of anomalies within data, difficulty of anomalies, and clusteredness) according to which they can be divided.

Experimental results

See Fig. 10.

Table 5 Datasets used to create benchmarking problems

Dataset	Dimension	Number of samples				
		Normal	Easy	Medium	Hard	Very hard
Abalone	10	2153	7	44	955	1018
Blood-transfusion	4	384	7	9	49	84
Breast-cancer-wisconsin	30	357	188	18	5	–
Breast-tissue	9	66	17	5	3	15
Cardiotocography	27	1831	143	86	48	18
Ecoli	7	205	81	27	9	14
Gisette	4971	3500	–	1471	2029	–
Glass	10	114	75	20	3	2
Haberman	3	225	4	11	19	47
Ionosphere	33	225	36	86	3	–
Iris	4	100	44	2	2	2
Isolet	617	4497	40	3260	–	–
Letter-recognition	617	4197	35	3565	–	–
Libras	90	216	115	28	–	–
Madelon	500	1300	–	1300	–	–
Magic-telescope	10	12, 332	2808	1074	1079	1727
Miniboone	50	93,565	17,744	6179	5703	6873
Multiple-features	649	1200	63	737	–	–
Musk-2	166	5581	604	212	105	96
Page-blocks	10	4913	315	69	77	99
Parkinsons	22	147	31	13	4	–
Pendigits	16	5539	5286	99	36	32
Pima-indians	8	500	101	76	45	46
Sonar	60	111	55	42	–	–
Spect-heart	44	212	7	46	2	–
Statlog-satimage	36	3594	2520	111	84	126
Statlog-segment	18	1320	866	73	30	21
Statlog-shuttle	8	57,769	10	19	26	176
Statlog-vehicle	18	629	46	86	65	20
Synthetic-control-chart	60	400	197	3	–	–
Vertebral-column	6	410	–	–	68	142
Wall-following-robot	24	2923	1841	380	170	142
Waveform-1	21	3304	1204	279	147	66
Waveform-2	21	3304	1203	269	150	74
Wine	13	107	65	6	–	–
Yeast	8	752	177	214	211	130

The number of samples is after the dataset has been converted to a two-class problem as described in Appendix "Construction of datasets". Columns captioned easy, medium, hard, very-hard shows the number of anomalous samples with a given level of difficulty. Column captioned normal shows the number of normal samples

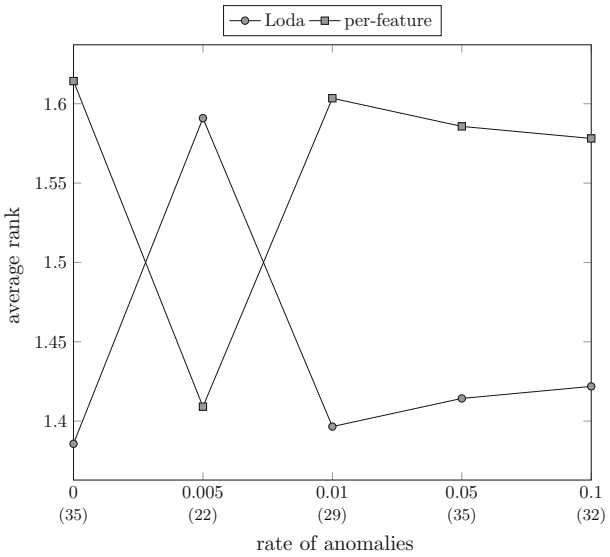


Fig. 10 The figure shows average rank of Loda and detector based on an ensemble of histograms modeling individual features on different rates of anomalies within data. The *small number in parentheses* shows the number of datasets used to create benchmarking problems. Notice that only 22 datasets had enough normal samples such that the problems with rate of anomalies 0.005 can be created. Wilcoxon signed rank test assessing that both detectors delivers the same performance on each rate of anomalies separately accepted hypothesis with p values 0.053, 0.44, 0.11, 0.16, and 0.31 in order of increasing rate of anomalies. The same test on results from all experiments rejected the hypothesis with a p value 0.028, which means that Loda's performance is statistically better

References

Aggarwal, C. C. (2013a). *Outlier analysis*. New York: Springer.

Aggarwal, C. C. (2013b). Outlier ensembles: Position paper. *ACM SIGKDD Explorations Newsletter*, *14*(2), 49–58.

Akhilomen, J. (2013). Data mining application for cyber credit-card fraud detection system. In P. Perner (Ed.), *Advances in data mining. Applications and theoretical aspects, LNCS* (Vol. 7987, pp. 218–228). Berlin: Springer.

Andoni, A., & Indyk, P. (2006). Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *47th annual IEEE symposium on foundations of computer science* (pp 459–468)

Bay, S. D., & Schwabacher, M. (2003). Mining distance-based outliers in near linear time with randomization and a simple pruning rule. In *Proceedings of the ninth ACM SIGKDD international conference on knowledge discovery and data mining*, ACM, New York, NY, USA, KDD '03, pp 29–38. doi:10.1145/956750.956758

Ben-Haim, Y., & Tom-Tov, E. (2010). A streaming parallel decision tree algorithm. *The Journal of Machine Learning Research*, *11*, 849–872.

Bentley, J. L. (1975). Multidimensional binary search trees used for associative searching. *Communication of ACM*, *18*(9), 509–517.

Birgé, L., & Rozenholc, Y. (2006). How many bins should be put in a regular histogram. *ESAIM: Probability and Statistics*, *10*, 24–45.

Breunig, M. M., Kriegel, H., Ng, R., & Sander, J. (2000). Lof: Identifying density-based local outliers. *SIGMOD Record*, *29*(2), 93–104.

Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. *ACM Computing Surveys (CSUR)*, *41*(3), 1–58.

Chang, C. C., & Lin, C. J. (2011). LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology 2*, 27:1–27:27. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm

Cormode, G., & Muthukrishnan, S. (2005). An improved data stream summary: The count-min sketch and its applications. *Journal of Algorithms*, *55*(1), 58–75.
Dang, X., Micenková, B., Assent, I., & Ng, R. (2013). Local outlier detection with interpretation. In *Machine learning and knowledge discovery in databases* LNCS, Vol. 8190, pp. 304–320, Springer.
Datar, M., Gionis, A., Indyk, P., & Motwani, R. (2002). Maintaining stream statistics over sliding windows. *SIAM Journal on Computing*, *31*(6), 1794–1813.
de Vries, T., Chawla, S., & Houle, M. E. (2010). Finding local anomalies in very high dimensional space. In *10th international conference on data mining*, IEEE, pp. 128–137.
Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, *7*, 1–30.
Donjerkovic, D., Ioannidis, Y.E., & Ramakrishnan, R. (2000). Dynamic histograms: Capturing evolving data sets. In *Proceedings of the international conference on data engineering*, IEEE Computer Society Press, 1998, pp. 86–86.
Emmott, A. F., Das, S., Dietterich, T., Fern, A., & Wong, W. K. (2013). Systematic construction of anomaly detection benchmarks from real data. In *Proceedings of the ACM SIGKDD workshop on outlier detection and description*, ACM, ODD '13, pp. 16–21.
Frank, A., & Asuncion, A. (2010). UCI machine learning repository. http://archive.ics.uci.edu/ml
Freund, Y., Schapire, R. E., et al. (1996). Experiments with a new boosting algorithm. *ICML*, *96*, 148–156.
Gama, J., & Pinto, C. (2006). Discretization from data streams: applications to histograms and data mining. In *Proceedings of the 2006 ACM symposium on applied computing*, pp. 662–667.
Gao, J., & Tan, P. N. (2006). Converting output scores from outlier detection algorithms into probability estimates. In *Sixth international conference on data mining*, IEEE, pp. 212–221.
Guha, S., Koudas, N., & Shim, K. (2006). Approximation and streaming algorithms for histogram construction problems. *ACM Transactions on Database Systems, 31*(1), 396–438.
Harmeling, S., Dornhege, G., Tax, D., Meinecke, F., & Muller, K. (2006). From outliers to prototypes: Ordering data. *Neurocomputing*, *69*(13–15), 1608–1618.
Ho, T. K. (1998). The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *20*(8), 832–844.
Johnson, W. B., & Lindenstrauss, J. (1984). Extensions of lipschitz mappings into a hilbert space. *Contemporary Mathematics*, *26*(189–206), 1.
Jonathan, W., Aryal, S., & Zhou, G. T. (2014). Implementation of isolation forest. http://sourceforge.net/projects/mass-estimation/
Keller, F., Muller, E., & Bohm, K. (2012). HiCS: High contrast subspaces for density-based outlier ranking. In *Proceedings of international conference on data engineering (ICDE)*, IEEE, pp. 1037–1048.
Kivinen, J., Smola, A. J., & Williamson, R. (2004). Online learning with kernels. *IEEE Transactions on Signal Processing*, *52*(8), 2165–2176.
Knorr, E., & Ng, R. T. (1999). Finding intensional knowledge of distance-based outliers. In *Proceedings of the international conference on very large data bases*, pp. 211–222.
Kriegel, H. P., & Zimek, A. (2008). Angle-based outlier detection in high-dimensional data. In *Proceedings of the 14th international conference on Knowledge discovery and data mining*, ACM, pp. 444–452.
Kuncheva, L. (2004). *Combining pattern classifiers: Methods and algorithms*. New York: John Wiley & Sons.
Lazarevic, A., & Kumar, V. (2005). Feature bagging for outlier detection. In *Proceedings of the 11th international conference on Knowledge discovery and data mining*, Vol. 21, pp. 157–166.
Li, P. (2007). Very sparse stable random projections for dimension reduction in $l\,\alpha(0<\alpha<2)$ norm. In *Proceedings of the 13th international conference on Knowledge discovery and data mining*, p. 440.
Liu, F. T., Ting, K. M., & Zhou, Z. H. (2008). Isolation forest. In *Eighth IEEE international conference on data mining*, IEEE, pp. 413–422.
Ma, J., Saul, L. K., Savage, S., & Voelker, G. M. (2009). Identifying suspicious urls: An application of large-scale online learning. In *Proceedings of the 26th annual international conference on machine learning*, ACM, pp. 681–688.
Muller, E., Schiffer, M., & Seidl, T. (2011). Statistical selection of relevant subspace projections for outlier ranking. In *27th international conference on data engineering (ICDE)*, IEEE, pp. 434–445.
Nguyen, H. V., Ang, H. H., & Gopalkrishnan, V. (2010). Mining outliers with ensemble of heterogeneous detectors on random subspaces. In H. Kitagawa, Y. Ishikawa, Q. Li, & C. Watanabe (Eds.), *Database systems for advanced applications. Lecture notes in computer science* (Vol. 5981, pp. 368–383). Berlin, Heidelberg: Springer.
Noto, K., Brodley, C., & Slonim, D. (2012). FRaC: A feature-modeling approach for semi-supervised and unsupervised anomaly detection. *Data Mining and Knowledge Discovery*, *25*, 109–133.
Pevný, T., Rehák, M., & Grill, M. (2012). Detecting anomalous network hosts by means of pca. In *IEEE international workshop on information forensics and security*, pp. 103–108.

Pham, N., & Pagh, R. (2012). A near-linear time approximation algorithm for angle-based outlier detection in high-dimensional data. In *Proceedings of the 18th international conference on knowledge discovery and data mining*, ACM, pp. 877–885.

Pokrajac, D., Lazarevic, A., & Latecki, J. L. (2007). Incremental local outlier detection for data streams. In *2007 IEEE symposium on computational intelligence and data mining*, IEEE, pp. 504–515.

Poosala, V., Haas, P. J., Ioannidis, Y. E., & Shekita, E. J. (1996). Improved histograms for selectivity estimation of range predicates. *SIGMOD Records*, *25*(2), 294–305.

Rondina, J., Hahn, T., de Oliveira, L., Marquand, A., Dresler, T., Leitner, T., et al. (2014). SCoRS—A method based on stability for feature selection and mapping in neuroimaging. *IEEE Transactions on Medical Imaging*, *33*(1), 85–98.

Schölkopf, B., Platt, J. C., Shawe-Taylor, J., Smola, A. J., & Williamson, R. C. (2001). Estimating the support of a high-dimensional distribution. *Neural Computation*, *13*(7), 1443–1471.

Schubert, E., Wojdanowski, R., Zimek, A., & Kriegel, H. P. (2012). On evaluation of outlier rankings and outlier scores. In *Proceedings of the 12th SIAM international conference on data mining*, pp. 1047–1058.

Shyu, M. l., Chen, S. C., Sarinnapakorn, K., & Chang, L. (2003). A novel anomaly detection scheme based on principal component classifier. In *Proceedings of the IEEE foundations and new directions of data mining workshop*, pp. 172–179.

Somol, P., Grim, J., & Pudil, P. (2011). Fast dependency-aware feature selection in very-high-dimensional pattern recognition. In *International conference on systems, man, and cybernetics*, IEEE, pp. 502–509.

Somol, P., Grim, J., Filip, J., & Pudil, P. (2013). On stopping rules in dependency-aware feature ranking. In *Progress in pattern recognition, image analysis, computer vision, and applications*, Springer, pp. 286–293.

Šourek, G., Kuzelka, O., & Železnỳ, F. (2013). Predicting top-k trends on twitter using graphlets and time features. ILP 2013 Late Breaking Papers, pp. 52–57.

Spinosa, E. J., Carvalho, L. F., & Gama, J. (2009). Novelty detection with application to data streams. *Intelligent Data Analysis*, *13*(3), 405–422.

Sricharan, K., & Hero, A. O. (2011). Efficient anomaly detection using bipartite k-nn graphs. *Advances in Neural Information Processing Systems*, *24*, 478–486.

Tan, S. C. (2014). *Implementation of half-space trees*. https://sites.google.com/site/analyticsofthings/recentwork-fast-anomaly-detection-for-streaming-data

Tan, S. C., Ting, K. M., & Liu, T. F. (2011). Fast anomaly detection for streaming data. In *Proceedings of the twenty-second international joint conference on artificial intelligence-volume volume two*, AAAI Press, pp. 1511–1516.

Tax, D. M. J., & Duin, R. P. W. (2004). Support vector data description. *Machine Learning*, *54*(1), 45–66.

Tax, D. M. J., & Laskov, P. (2003). Online svm learning: From classification to data description and back. In *13th workshop on neural networks for signal processing*, IEEE, pp. 499–508.

Yeung, D. Y., & Chow, C. (2002). Parzen-window network intrusion detectors. In *Proceedings of 16th international conference on pattern recognition*, Vol. 4, pp. 385–388.

Zhang, H. (2004). The optimality of naive bayes. In *Proceedings of the seventeenth international Florida artificial intelligence research society conference*.

Zhu, J., & Hastie, T. (2005). Kernel logistic regression and the import vector machine. *Journal of Computational and Graphical Statistics*, *14*(1), 185–205.

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