Mach Learn (2016) 105:305–333
DOI 10.1007/s10994-016-5567-7

Expected similarity estimation for large-scale batch and streaming anomaly detection

Markus Schneider[1,2] · **Wolfgang Ertel**[2] · **Fabio Ramos**[3]

Received: 28 September 2015 / Accepted: 16 April 2016 / Published online: 18 May 2016

Abstract We present a novel algorithm for anomaly detection on very large datasets and data streams. The method, named *EXPected Similarity Estimation* (EXPoSE), is kernel-based and able to efficiently compute the similarity between new data points and the distribution of regular data. The estimator is formulated as an inner product with a reproducing kernel Hilbert space embedding and makes no assumption about the type or shape of the underlying data distribution. We show that offline (batch) learning with EXPoSE can be done in *linear* time and online (incremental) learning takes *constant* time per instance and model update. Furthermore, EXPoSE can make predictions in *constant* time, while it requires only *constant* memory. In addition, we propose different methodologies for concept drift adaptation on evolving data streams. On several real datasets we demonstrate that our approach can compete with state of the art algorithms for anomaly detection while being an order of magnitude faster than most other approaches.

Keywords Anomaly detection · Large-scale data · Kernel methods · Hilbert space embedding · Mean map

1 Introduction

What is an anomaly? An *anomaly* is an element whose properties differ from the majority of other elements under consideration which are called the *normal* data. "*Anomaly*

Editor: João Gama.

✉ Markus Schneider
mschneider@linkdot.org

1 Institute of Neural Information Processing, University of Ulm, Ulm, Germany

2 Institute for Artificial Intelligence, University of Applied Sciences Ravensburg-Weingarten, Weingarten, Germany

3 School of Information Technologies, The University of Sydney, Sydney, Australia

detection refers to the problem of finding patterns in data that do not conform to expected behavior. These non-conforming patterns are often referred to as *anomalies*[...]" (Chandola et al. 2009).

Typical applications of anomaly detection are network intrusion detection, credit card fraud detection, medical diagnosis and failure detection in industrial environments. For example, systems which detect *unusual* network behavior can be used to complement or replace traditional intrusion detection methods which are based on experts' knowledge in order to defeat the increasing number of attacks on computer based networks (Kumar 2005). Credit card transactions which differ significantly from the usual shopping behavior of the card owner can indicate that the credit card was stolen or a compromise of data associated with the account occurred (Aleskerov et al. 1997). The diagnosis of radiographs can be supported by automated systems to detect breast cancers in mammographic image analysis (Spence et al. 2001). Unplanned downtime of production lines caused by failing components is a serious concern in many industrial environments. Here anomaly detection can be used to detect unusual sensor information to predict possible faults and enabling condition-based maintenance (Zhang et al. 2011). Novelty detection can be used to detect new interesting or unusual galaxies in astronomical data such as the Sloan Digital Sky Survey (Xiong et al. 2011).

Obtaining labeled training data for all types of anomalies is often too expensive. Imagine the labeling has to be done by a human expert or is obtained through costly experiments (Hodge and Austin 2004). In some applications anomalies are also very rare as in air traffic safety or space missions. Hence, the problem of anomaly detection is typically unsupervised, however it is implicitly assumed that the dataset contains only very few anomalies. This assumption is reasonable since it is quite often possible to collect large amounts of data for the normal state of a system as, for example usual credit card transactions or network traffic of a system not under attack.

The computational complexity and memory requirements of classical algorithms become the limiting factor when applied to large-scale datasets as they occur nowadays. To solve this problem we propose a new anomaly detection algorithm called *EXPected Similarity Estimation* (EXPoSE). As explained later in detail, the EXPoSE anomaly detection classifier

$$\eta(z) = \langle \phi(z), \mu[\mathbb{P}] \rangle$$

calculates a score (the likelihood of z belonging to the class of normal data) using the inner product between a feature map ϕ and the kernel mean map $\mu[\mathbb{P}]$ of the distribution of normal data $\mathbb{P}$. We will show that this inner product can be evaluated in *constant* time, while $\mu[\mathbb{P}]$ can be estimated in *linear* time, has *constant* memory consumption and is designed to solve *very* large-scale anomaly detection problems.

Moreover, we will see that the proposed EXPoSE classifier can be learned incrementally making it applicable to *online* and *streaming* anomaly detection problems. Learning on data streams directly is unavoidable in many applications such as network traffic monitoring, video surveillance and document feeds as data arrives continuously in fast streams with a volume too large or impractical to store.

Only a few anomaly detection algorithms can be applied to large-scale problems and even less are applicable to streaming data. The proposed EXPoSE anomaly detector fills this gap.

Our main contributions are:

- We present an efficient anomaly detection algorithm, called *EXPected Similarity Estimation* (EXPoSE), with $\mathcal{O}(n)$ training time, $\mathcal{O}(1)$ prediction time and only $\mathcal{O}(1)$ memory requirements with respect to the dataset size n.
- We show that EXPoSE is especially suitable for parallel and distributed processing which makes it scalable to very large problems.
- We demonstrate how EXPoSE can be applied to online and streaming anomaly detection, while requiring only $\mathcal{O}(1)$ time for a model update, $\mathcal{O}(1)$ time per prediction and $\mathcal{O}(1)$ memory.
- We introduce two different approaches which allow EXPoSE to be efficiently used with the most common techniques for concept drift adaptation.
- We evaluate EXPoSE on several real datasets, including surveillance, image data and network intrusion detection.

This paper is organised as follows: we first provide a formal problem description including a definition of batch and streaming anomaly detection. Section 3 provides an overview of related work and a comparison of these techniques. Section 4 introduces the EXPoSE anomaly detection algorithm along with the necessary theoretical framework. Subsequently we show in Sect. 5 how EXPoSE can be applied to streaming anomaly detection problems. The key to EXPoSE s computational performance is subject to Sect. 6. In Sect. 7 we empirically compare EXPoSE with several state of the art anomaly detectors.

This is an extended and revised version of a preliminary conference report that was presented in the International Joint Conference on Neural Networks 2015 (Schneider et al. 2015). This work reviews the EXPoSE anomaly detection algorithm and provides a derivation that makes fewer assumptions on the input space and kernel function. It provides an online version of EXPoSE that is applicable to large-scale and evolving data streams. The experimental section is extended comparing more algorithms and additional datasets with a statistical analysis of the results.

2 Problem definition

Even though there is a vast amount of literature on anomaly detection, there is no unique definition of what anomalies are and what exactly anomaly detection is. In this section we will state the problem of anomaly detection in batch and streaming application.

Definition 1 (*Input Space*) The *input space* for an observation X is a measurable space[1] $(\mathcal{X}, \mathscr{X})$ containing all values that X might take. We denote the realization after measurement of the random variable X with $X = x$.

We make no assumptions about the nature of the input space $\mathcal{X}$ which can consist of simple numerical vectors, but also can contain images, video data or trajectories of vehicles and people. We assume that there is a true (but unknown) distribution $\mathbb{P}_X : \mathscr{X} \to [0, 1]$ of the data.

Definition 2 (*Output/Label Space*) In anomaly detection an observation $X = x$ can belong to the class of normal data C_N or can be an anomaly C_A. This is called *label* of the observation

[1] A *measurable space* is a tuple $(\mathcal{X}, \mathscr{X})$, where $\mathcal{X}$ is a nonempty set and $\mathscr{X}$ is a σ-algebra of its subsets. We refer the reader unfamiliar with this topic to Kallenberg (2006) for an overview.

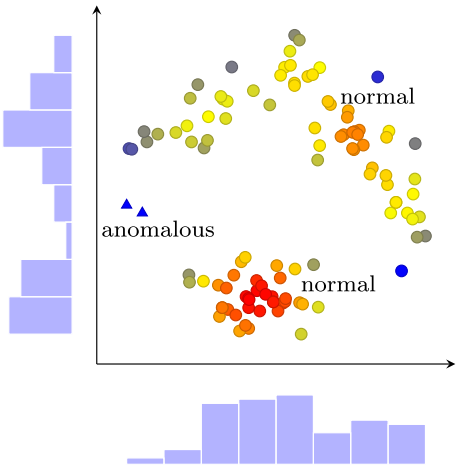

Fig. 1 Example of two instances (*triangle*) which are different from the distribution of normal data (*circle*) along with histograms of the marginal distributions

and denoted by the random variable Y. The collection of all labels is given by the measurable space $(\mathcal{Y}, \mathscr{Y})$ called *label space* or *output space* (Fig. 1).

The distribution of the observation $x \in \mathcal{X}$ is stochastic and depends on the label Y and hence is distributed according to $\mathbb{P}_{X|Y}$.

Definition 3 (*Prediction/Decision Space*) Based on the outcome $X = x$ of an observation, the objective of an anomaly detection algorithm is to make a prediction $\vartheta \in \mathcal{Q}$, where the measurable space $(\mathcal{Q}, \mathscr{Q})$ is called the *prediction space* or sometimes *decision space*.

The prediction space $\mathcal{Q}$ is not necessarily equal to label space $\mathcal{Y}$. Especially in anomaly detection and classification many algorithms calculate a probability or a score for a label. Such a score is called *anomaly score* if it quantifies the likelihood of x belonging to C_A and *normal score* if it determines the degree of certainty to which x belongs to C_N.

Scoring based algorithms are more flexible than techniques which assign hard class labels since anomalies can be ranked and prioritized according their score or a domain specific discrimination threshold can be applied to separate anomalies from normal data. For example we can define a mapping $\tau : (\mathcal{Q}, \mathscr{Q}) \to (\mathcal{Y}, \mathscr{Y})$ as

$$\tau_\theta(\vartheta) = \begin{cases} C_N & \text{if } \vartheta > \theta \\ C_A & \text{else} \end{cases}$$

based on the threshold θ. Such a domain specific threshold depends on the costs of *false positives* (an anomaly is reported when the observation is normal) and *false negatives* (no anomaly is reported when the observation is anomalous).

Definition 4 (*Classifier/Predictor*) A measurable function $\eta : (\mathcal{X}, \mathscr{X}) \to (\mathcal{Q}, \mathscr{Q})$ is called a *classifier* or *predictor*.

A classifier calculates a prediction for an observation $X = x$. In the context of anomaly detection our goal is to find a good predictor which can distinguish normal from anomalous data. However, the distribution $\mathbb{P}_X \otimes \mathbb{P}_Y$ is typically unknown and hence we have to build a classifier solely based on observations. The estimation of such a functional relationship between the input space $\mathcal{X}$ and the prediction space $\mathcal{Q}$ is called *learning* or *training*.

Definition 5 (*Batch Anomaly Detection*) In *unsupervised batch learning* we have access to $n \in \mathbb{N}$ unlabeled independent realizations $(x_1, \ldots, x_n)$, identically distributed according to $\otimes_{i=1}^{n} \mathbb{P}_X$ which form a *training set*. In anomaly detection we make the following assumptions:

- The objective is to estimate a predictor η based on an unlabeled training set.
- The training set contains mostly normal instances from $\mathbb{P}_{X|Y=C_N}$ and only a few anomalies as, by definition, anomalies are rare events.
- It is assumed that the algorithm has complete access to all n elements of the dataset at once.
- We may have access to a small labeled fraction of the training data to configure our algorithm.

Definition 6 (*Online & Streaming Anomaly Detection*) In contrast to batch learning, where the full dataset $(x_1, \ldots, x_n)$ is permanently available, *online* learning algorithms observe each x_t, $(1 \leq t \leq n)$ only once and in a sequential order. Typically these algorithms have limited memory and thus can only store a small set of previously observed samples. Hence it is necessary to continuously update the prediction function based on the new observations

$$(x_t, \eta_t) \mapsto \eta_{t+1}$$

to build a new predictor. This can be generalized to *streaming anomaly detection* where data arrives in a possible infinite sequence $x_1, x_2, x_3, \ldots$ of observations. Moreover the input and label space distributions may evolve over time, a problem known as *concept drift*. (Formally we now have to consider the stochastic processes $\{X_t\}_{t\in\mathbb{N}}$, $\{Y_t\}_{t\in\mathbb{N}}$ and their corresponding distributions). It is therefore necessary that an algorithm can adapt to the changes e.g. by forgetting outdated information while incorporating new knowledge (Gama et al. 2014). These classes of algorithms assume that more recent observations carry more relevant information than older data. In summary streaming anomaly detection has the following key characteristics:

- The data stream is possible infinite which requires the algorithm to learn incrementally since it is not possible to store the whole stream.
- Most instances in the data stream belong to the class of normal data and anomalies are rare.
- The stream can evolve over time, forcing algorithms to adapt to changes in the data distribution.
- Only a small time frame at the beginning of the stream is available to configure the algorithm's parameter.
- We have to instantly make a prediction $\eta_t(x_t)$ as soon as an observation x_t is available. This requires that predictions can be made fast.

3 Related work

Many approaches from statistics and machine learning can be used for anomaly detection (Chandola et al. 2009; Gupta et al. 2014), but only a few are applicable on high-dimensional, large-scale problems, where a vast amount of information has to be processed. We review several algorithms with focus on their computational complexity and memory requirements.

3.1 Distribution based models

One of the oldest, statistical methods for anomaly detection is the kernel (or Parzen) density estimator (KDE). With $\mathcal{O}(n)$ time for predictions the KDE is too slow for large amounts of data and known to be problematic in the case of increasing data dimensionality (Gretton et al. 2012). Fitting parametric distributions such as the Normal, Gamma, etc. is problematic since in general, the underlying data distribution is unknown. Therefore, a mixture of Gaussians is often used as a surrogate for the true distribution as, for example, done by SmartSifter (Yamanishi et al. 2004). SmartSifter can handle multivariate data with both, continuous and categorical observations. The main disadvantage of this approach is the high number of parameters required for the mixture model which grows quadratically with the dimension (Tax 2001).

3.2 Distance based models

Distance based models are popular since most of them are easy to implement and interpret. Knorr et al. (2000) and Knorr and Ng (1998) labels an observation as a *distance based outlier* (anomaly) if at least a fraction of points in the dataset have a distance of more than a threshold (based on the fraction) to this point. The authors proposed two simple algorithms which have both $\mathcal{O}(n^2)$ runtime and a cell-based version which runs linear in n, but exponential with the dimension d. Ramaswamy et al. (2000) argues that the threshold can be difficult to determine and proposes an *outlier score* which is simply the distance from a query point to its kth nearest neighbor. The algorithm is called KNNOutlier and suffers from the problem of efficient nearest neighbor search. If the input space is of low dimension and n is much larger than 2^d then finding 1 nearest neighbor in a $k-d$ tree with randomly distributed points takes $\mathcal{O}(\log n)$ time on average. However this does not hold in high dimensions, where such a tree is not better than an exhaustive search with $\mathcal{O}(n)$ (Goodman and O'Rourke 2004). Also the algorithm proposed by Ramaswamy et al. is only used to identify the top outliers in a given dataset. An alternative algorithm was proposed by Angiulli and Pizzuti (2002) using the sum of distances from its k-nearest neighbors. Ott et al. (2014) simultaneously perform clustering and anomaly detection in an integer programming optimization task.

Popular approaches from data mining for distance based novelty detection on streams are OLINDDA (Spinosa et al. 2007) and its extension MINAS (Faria et al. 2013) which both represent normal data as a union of spheres obtained by clustering. This representation becomes problematic if data within one cluster exhibits high variance since then the decision boundary becomes too large to detect novelties. Both algorithms are designed to incorporate novel classes into their model of normal data and hence barely applicable to anomaly detection.

The STream OutlieR Miner (STORM) (Angiulli and Fassetti 2007, 2010) offers an efficient solution to the problem of distance-based outlier detection over windowed data streams using a new data structure called Indexed Stream Buffer. Continuous Outlier Detection (COD; Kontaki et al. 2011) aims to further improve the efficiency of STORM by reducing the number of range queries.

3.3 Density based models

Nearest neighbor data description (Tax 2001) approximates a local density while using only distances to its first neighbor. The algorithm is very simple and often used as a baseline. It is also relatively slow approaching $\mathcal{O}(n)$ per prediction. More sophisticated is the local density

based approach called Local Outlier Factor (LOF; Breunig et al. 2000). It considers a point to be an anomaly if there are only *relatively* few other points in its neighborhood. LOF was extended to work on data streams (Pokrajac 2007), however both (the batch and incremental approach) are relatively slow with training time between $\mathcal{O}(n \log n)$ and $\mathcal{O}(n^2)$ and $\mathcal{O}(n)$ memory consumption.

The angle based outlier detection for high-dimensional data (ABOD) proposed by Kriegel and Zimek (2008) is able to outperform LOF, however requires $\mathcal{O}(n^2)$ time per prediction with the exact model and $\mathcal{O}(n + k^2)$ if the full dataset is replaced by the k-nearest neighbors of the query point (FastAbod).

3.4 Classification and tree based models

The One-class support vector machine (OC-SVM; Schölkopf et al. 2001; Tax and Duin 2004) is a kernel based method which attempts to find a hyperplane such that most of the observations are separated from the origin with maximum margin. This approach does not scale very well to large datasets where predictions have to be made with high frequency. As Steinwart (2003) showed, the number of support vectors can grow linearly with the dataset size. There exist One-class support vector machines which can be learned incrementally (Gretton and Desobry 2003).

Hoeffding Trees (Domingos and Hulten 2000) are anytime decision trees to mine high-speed data streams. The Hoeffding Trees algorithm is not applicable to solve the unsupervised anomaly detection problem considered in this work since it requires the availability of class labels. Streaming Half-Space-Trees (HSTa; Tan et al. 2011) randomly construct a binary tree structure without any data. It selects a dimension at random and splits it in half. Each tree then counts the number of instances from the training set at each node referred to as "mass". The score for a new instance is then proportional to the mass in the leaf in which new instance hits after passing down the tree. Obviously, an ensemble of such trees can be built-in constant time and the training is linear in n. However, randomly splitting a very high-dimensional space will not yield in a tree sufficiently fine-grained for anomaly detection. The RS-Forest (Wu et al. 2014) is a modification of HSTa in which each dimension is not splitted in half, but at a random cut-point. Also the assumption that "[...] once each instance is scored, streaming RS-Forest will receive the true label of the instance [...]" (Wu et al. 2014) does not always hold. The Isolation Forest (*i*Forest) is an algorithm which uses a tree structure to isolate instances (Liu et al. 2012). The anomaly score is based on the path length to an instance. *i*Forests achieve a constant training time and space complexity by sub-sampling the training set to a fixed size. The characteristics of the most relevant anomaly detection algorithms is summarized in Table 1. All complexities are given with respect to the dataset size n in high-dimensional spaces.

Most methods discussed do not scale to very large problems since either the training time is non-linear with the number of samples or the time to make a single prediction increases with the dataset size (stream length). We now PRESENT a novel anomaly detection algorithm to overcome these problems.

4 Expected similarity estimation

As before, let X be a random variable taking values in a measurable space $(\mathcal{X}, \mathscr{X})$. We are primarily interested in the distribution of normal data $\mathbb{P}_{X|Y=C_N}$ for which we will simply use

Table 1 Comparison of anomaly detection techniques

	Training	Prediction	Memory	Batch	Online	Streaming	Problem size
EXPoSE	$\mathcal{O}(n)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$	✓	✓	✓	Large
OC-SVM	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	✓	✓	✗	Medium
LOF	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	✓	✓	✗	Medium
KDE	$\mathcal{O}(1)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	✓	✓	✗	Small
FastAbod	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	✓	✗	✗	Small
*i*Forest	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$	✓	✗	✗	Large
STORM	$\mathcal{O}(n)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$	✗	✗	✓	Small
COD	$\mathcal{O}(n)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$	✗	✗	✓	Small
HSTa	$\mathcal{O}(n)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$	✗	✗	✓	Medium

the shorthand notation $\mathbb{P}$ in the remainder of this work. Next we introduce some definitions which are necessary in the following.

A Hilbert space $(\mathcal{H}, \langle\cdot,\cdot\rangle)$ of functions $f\colon \mathcal{X} \to \mathbb{R}$ is said to be a *reproducing kernel Hilbert space* (RKHS) if the evaluation functional $\bar{\delta}_x\colon f \mapsto f(x)$ is continuous. A function $k\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ which satisfies the reproducing property

$$\langle f, k(x,\cdot)\rangle = f(x) \quad \text{and in particular}$$
$$\langle k(x,\cdot), k(y,\cdot)\rangle = k(x,y)$$

is called *reproducing kernel* of $\mathcal{H}$ (Steinwart and Christmann 2008).[2] The map $\phi : \mathcal{X} \to \mathcal{H}$, $\phi\colon x \mapsto k(x,\cdot)$ with the property that

$$k(x,y) = \langle \phi(x), \phi(y)\rangle$$

is called *feature map*.

Throughout this work we assume that the reproducing kernel Hilbert space $(\mathcal{H}, \langle\cdot,\cdot\rangle)$ is *separable* such that ϕ is measurable. We therefore assume that the input space $\mathcal{X}$ is a separable topological space and the kernel k on $\mathcal{X}$ is continuous, which is sufficient for $\mathcal{H}$ to be separable (Steinwart and Christmann 2008, Lemma 4.33).

As mentioned in the introduction, EXPoSE calculates a score which can be interpreted as the likelihood of an instance $z \in \mathcal{X}$ belonging to the distribution of normal data $\mathbb{P}$. It uses a kernel function k to measure the similarity between instances of the input space $\mathcal{X}$.

Definition 7 (*Expected Similarity Estimation*) The *expected similarity* of $z \in \mathcal{X}$ with respect to the (probability) distribution $\mathbb{P}$ is defined as

$$\eta(z) = \mathbb{E}\left[\phi(z)\right] = \int_{\mathcal{X}} k(z,x)\,\mathrm{d}\mathbb{P}(x),$$

where $k\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a reproducing kernel.

Intuitively the query point z is compared to all other points of the distribution $\mathbb{P}$. We will shown that this equation can be rewritten as an inner product between the feature map $\phi(z)$

[2] The notation $k(x,\cdot)$ indicates that the second function argument is not bound to a variable.

and the *kernel mean map* $\mu[\mathbb{P}]$ of $\mathbb{P}$. This reformulation is of central importance and will enable us to efficiently compute all quantities of interest. Given a reproducing kernel k, the kernel mean map can be used to embed a probability measure into a RKHS where it can be manipulated efficiently. It is defined as follows.

Definition 8 (*Kernel Embedding*) Let $\mathbb{P}$ be a Borel probability measure on $\mathcal{X}$. The *kernel embedding* or *kernel mean map* $\mu[\mathbb{P}]$ of $\mathbb{P}$ is defined as

$$\mu[\mathbb{P}] = \int_{\mathcal{X}} k(x, \cdot)\, \mathrm{d}\mathbb{P}(x),$$

where k is the associated continuous, bounded and positive-definite kernel function.

We assume that the kernel k is bounded in expectation i.e.

$$\int_{\mathcal{X}} \sqrt{k(x,x)}\, \mathrm{d}\mathbb{P}(x) < \infty,$$

such that $\mu[\mathbb{P}]$ exists for all Borel probability measures $\mathbb{P}$ (Sejdinovic et al. 2013, Page 8). This is a weaker assumption than k being bounded. We can now continue to formulate the central theorem of our work.

Theorem 1 *Let $(\mathcal{H}, \langle\cdot,\cdot\rangle)$ be a RKHS with reproducing kernel $k\colon \mathcal{X}\times\mathcal{X} \to \mathbb{R}$. The expected similarity of $z \in \mathcal{X}$ with respect to the distribution $\mathbb{P}$ can be expressed as*

$$\begin{aligned}\eta(z) &= \int_{\mathcal{X}} k(z,x)\, \mathrm{d}\mathbb{P}(x) \\ &= \langle \phi(z), \mu[\mathbb{P}] \rangle,\end{aligned}$$

where $\mu[\mathbb{P}]$ is the kernel embedding of $\mathbb{P}$.

This reformulation has several desirable properties. At this point we see how the EXPoSE classifier can make prediction in *constant* time. After the kernel mean map $\mu[\mathbb{P}]$ of $\mathbb{P}$ is learned, EXPoSE only needs to calculate a single inner product in $\mathcal{H}$ to make a prediction. However there are some crucial aspects to consider i.e. in Hilbert spaces, integrals and continuous linear forms are not in general interchangeable. In the proof of Theorem 1 we will thus use the weak integral and show that it coincides with the strong integral (see Definitions 9 and 10 in the "Appendix"). A sufficient condition therefore is provided by the following lemma.

Lemma 1 *If ϕ is strong (Bochner) integrable then ϕ is weak (Pettis) integrable and the two integrals coincide. (Aliprantis and Border 2006, Theorem 11.50)*

We are now in the position to proof Theorem 1.

Proof (Theorem 1) By definition of the feature map ϕ we have

$$\int_{\mathcal{X}} k(x,z)\, \mathrm{d}\mathbb{P}(x) = \int_{\mathcal{X}} \langle \phi(z), \phi(x) \rangle\, \mathrm{d}\mathbb{P}(x)$$

By the assumption that k is bounded in expectation it follows that

$$\int_{\mathcal{X}} \|\phi(x)\|\, \mathrm{d}\mathbb{P}(x) = \int_{\mathcal{X}} \sqrt{k(x,x)}\, \mathrm{d}\mathbb{P}(x) < \infty$$

and therefore ϕ is strongly integrable and hence weakly integrable (Lemma 1). By definition of the weak integral we get for all $z \in \mathcal{X}$

$$\begin{aligned}\int_{\mathcal{X}} \langle \phi(z), \phi(x) \rangle \, \mathrm{d}\mathbb{P}(x) &= \langle \phi(z), \oint_{\mathcal{X}} \phi(x) \, \mathrm{d}\mathbb{P}(x) \rangle \\ &= \langle \phi(z), \int_{\mathcal{X}} \phi(x) \, \mathrm{d}\mathbb{P}(x) \rangle \\ &= \langle \phi(z), \mu[\mathbb{P}] \rangle\end{aligned}$$

for all probability measures $\mathbb{P}$. □

In anomaly detection, we cannot assume to know the distribution of normal data $\mathbb{P}$. However we assume to have access to $n \in \mathbb{N}$ independent realizations $(x_1, \ldots, x_n)$ sampled from $\mathbb{P}$. It is common in statistics to estimate $\mathbb{P}$ with the empirical distribution

$$\mathbb{P}_n = \frac{1}{n} \sum_{i=1}^{n} \delta_{x_i},$$

where δ_x is the Dirac measure. The empirical distribution $\mathbb{P}_n$ can also be used to construct an approximation $\mu[\mathbb{P}_n]$ of $\mu[\mathbb{P}]$ as

$$\mu[\mathbb{P}] \approx \mu[\mathbb{P}_n] = \frac{1}{n} \sum_{i=1}^{n} \phi(x_i)$$

which is called *empirical kernel embedding* (Smola et al. 2007). This is an efficient estimate since it can be shown (Schneider 2016) that under the assumption $\|\phi(X)\| \leq c$ with $c > 0$ the difference between $\mu[\mathbb{P}]$ and $\mu[\mathbb{P}_n]$ is in probability

$$P\left(\|\mu[\mathbb{P}] - \mu[\mathbb{P}_n]\| \geq \epsilon\right) \leq 2 \exp\left(-\frac{n\epsilon^2}{8c^2}\right)$$

for all $\epsilon > 0$.

As a consequence we can substitute $\mu[\mathbb{P}]$ with $\mu[\mathbb{P}_n]$ whenever the distribution $\mathbb{P}$ is not directly accessible yielding

$$\begin{aligned}\eta(z) &= \langle \phi(z), \mu[\mathbb{P}_n] \rangle \\ &= \left\langle \phi(z), \frac{1}{n} \sum_{i=1}^{n} \phi(x_i) \right\rangle\end{aligned}$$

as the (empirical) EXPoSE anomaly detector. The empirical kernel embedding $\mu[\mathbb{P}_n]$ is responsible for the *linear* training computational complexity of EXPoSE. We will call $\mu[\mathbb{P}_n]$ the EXPoSE *model*. One of the important observations is, that EXPoSE makes no assumption about the type or shape of the data distribution $\mathbb{P}$ as such assumption can be wrong, causing erroneous predictions. This is an advantage over other statistical approaches that try to approximate the distribution directly with parametric models.

4.1 Parallel and distributed processing

Parallel and distributed data processing is the key to scalable machine learning algorithms. The formulation of EXPoSE as $\eta(z) = \langle \phi(z), \mu[\mathbb{P}_n] \rangle$ is especially appealing for this kind of operations. We can use a SPMD (single program, multiple data) technique to achieve

parallelism. One of the first programming paradigms on this line is Googles *MapReduce* for processing large data sets on a cluster (Dean and Ghemawat 2008).

Assume a partition of the dataset $(x_1, \ldots, x_n)$ into $m \leq n$ distinct collections $s_1, \ldots, s_m$ which can be distributed on different computational nodes. Obviously, the feature map ϕ can be applied in parallel to all instances in $x_1, \ldots, x_n$. We also note that the partial sums

$$p(s_i) = \sum_{x \in s_i} \phi(x)$$

can be calculated without any communication or data-sharing across concurrent computations. Solely the partial sums $p(s_i)$, which are elements of $\mathcal{H}$, need to be transmitted and combined as

$$\mu[\mathbb{P}_n] = \frac{1}{n} \sum_{i=1}^{m} p(s_i)$$

by a central processing node.

Summary

In this section we derived the EXPoSE anomaly detection algorithm. We showed how EXPoSE can be expressed as an inner product $\langle \phi(z), \mu[\mathbb{P}_n] \rangle$ between the kernel mean map of $\mathbb{P}$ and the feature mapping of a query point $z \in \mathcal{X}$ for which we need to make a prediction. Evaluating this inner product takes *constant* time while estimating the model $\mu[\mathbb{P}_n]$ can be done in *linear* time and with *constant* memory. We will explain the calculation of ϕ in more detail in Sect. 6 and will explore now how EXPoSE can be learned incrementally and applied to large-scale data streams.

5 Online and streaming EXPoSE

In this section we will show how EXPoSE can be used for online and streaming anomaly detection. To recap, a *data stream* is an often infinite sequence of observations $(x_1, x_2, x_3, \ldots)$, where $x_t \in \mathcal{X}$ is the instance arriving at time t. A source of such data can be, for example, continuous sensor readings from an engine or a video stream from surveillance cameras.

Domingos and Hulten (2001) identified the following requirement for algorithms operating on "the high-volume, open-ended data streams we see today".

- Require small constant time per instance.
- Use only a fixed amount of memory, independent of the number of past instances.
- Build a model using at most one scan over the data.
- Make a usable predictor available at any point in time.
- Ability to deal with concept drift.
- For streams without concept drift, produce a predictor that is equivalent (or nearly identical) to the one that would be obtained by an offline (batch) learning algorithm.

In this section we will show that the online version of EXPoSE fulfills all requirements, starting with the last item of the list.

Proposition 1 *The* EXPoSE *model $\mu[\mathbb{P}_n]$ can be learned incrementally, where each model update can be performed in $\mathcal{O}(1)$ time and memory.*

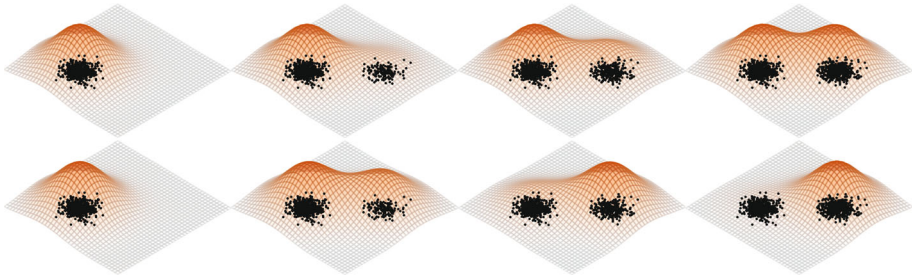

Fig. 2 An illustration of the difference between online (incremental) learning and model adaption to concept drift. The eight plots represent EXPoSE predictions for the observations indicated by the *black dots*. Each row displays four snapshots with increasing time from *left* to *right* as data becomes available from two clusters. We first sampled only from the *left* cluster and later only from the *right*. In the *top row* we incrementally build the model by adding more knowledge to it, whereas in the bottom row the model evolves and slowly forgets outdated observations (the points from the *left* cluster)

Proof Given a stream $(x_1, x_2, x_3, \dots)$ of observations and let $\mu[\mathbb{P}_1] = \phi(x_1)$. Whenever a new observation x_t is made at $t > 1$, the new model $\mu[\mathbb{P}_t]$ can be incrementally calculated as

$$\begin{aligned}\mu[\mathbb{P}_t] &= \frac{1}{t}\sum_{i=1}^{t}\phi(x_i) \\ &= \mu[\mathbb{P}_{t-1}] + \frac{1}{t}\Big(\phi(x_t) - \mu[\mathbb{P}_{t-1}]\Big)\end{aligned}$$

using the previous model $\mu[\mathbb{P}_{t-1}]$. □

We see that online learning of EXPoSE does neither increase the computational complexity nor the memory requirements of EXPoSE. We also emphasize that online learning yields the exact same model as the EXPoSE offline learning procedure.

5.1 Learning on evolving data streams

Sometimes it can be expected that the underlying distribution of the stream evolves over time. This is a property known as *concept drift* (Sadik and Gruenwald 2014). For example in environmental monitoring, the definition of "normal temperature" changes naturally with seasons. We can also expect that human behavior changes over time which requires us to redefine what anomalous actions are. In Fig. 2 we illustrate the difference between incremental learning as in Proposition 1 and a model which adapts itself to changes in the underlying distribution. In the following we will use w_t to denote the EXPoSE model at time t since the equation $w_t = \mu[\mathbb{P}_t]$ will *not* necessarily hold when concept drift adaptation is implemented.

In this work we are not concerned with the detection of concept drift (Gama 2010), but we will show how EXPoSE can be used efficiently with the most common approaches to concept drift adaption which either utilize *windowing* or *forgetting* mechanisms (Gama et al. 2014).

5.1.1 Windowing

Windowing is a straight forward technique which uses a buffer (the window) of $l \in \mathbb{N}$ previous observations. Whenever a new observation is added to the window, the oldest one is discarded. We can efficiently implement windowing for EXPoSE as follows.

Proposition 2 *Concept drift adaption on data streams using a sliding window mechanism can be implemented for* EXPoSE *with* $\mathcal{O}(1)$ *time and* $\mathcal{O}(l)$ *memory consumption, where* $l \in \mathbb{N}$ *is the window size.*

Proof Given a data stream $(x_1, x_2, x_3 \dots)$ and the window size l. For $t < l$ we set $w_t = \frac{1}{t}\sum_{i=1}^{t} \phi(x_i)$ and use the incremental update

$$\begin{aligned} w_t &= \frac{1}{l} \sum_{i=t-l+1}^{t} \phi(x_i) \\ &= w_{t-1} + \frac{1}{l}\phi(x_t) - \frac{1}{l}\phi(w_{t-1}), \end{aligned}$$

whenever $t \geq l$. □

The downside of a sliding window mechanism is the requirement to keep the past $l \in \mathbb{N}$ events in memory. Also the sudden discard of a data point can lead to abrupt changes in predictions of the classifier which is sometimes not desirable. Another question is how to choose the correct window size. A shorter sliding window allows the algorithm to react faster to changes and requires less memory though the available data might not be representative or noise has too much negative impact. On the other hand a wider window may take too long to adapt to concept drift. The window size is therefore often dynamically adjusted (Widmer and Kubat 1996) or multiple competing windows are used (Lazarescu et al. 2004).

5.1.2 Gradual forgetting (decay)

The problems of sliding window approaches can be avoided if a forgetting mechanism is applied, where the influence of older data gradually vanishes. Typically a parameter can be used to control the tradeoff between fast adaptation to new observations and robustness against noise in the data. We can realize such a forgetting mechanism for EXPoSE by replacing the factor $\frac{1}{t}$ in Proposition 1 by a constant $\gamma \in [0, 1)$ yielding

$$w_t = \begin{cases} \phi(x_t) & \text{for } t = 1 \\ \gamma\phi(x_t) + (1-\gamma)w_{t-1}, & \text{for } t > 1 \end{cases}$$

where, with $\gamma = 0$, no new observations are integrated into the model. This operation can be performed in constant time as summarized in the next proposition.

Proposition 3 *Concept drift adaptation on data streams using a forgetting mechanism can be implemented for* EXPoSE *in* $\mathcal{O}(1)$ *time and memory.*

Proof This is a direct consequence from Proposition 1. □

In general, weighting with a fixed γ or using a static window size is called *blind* adaptation since the model does not utilize information about changes in the environment (Gama 2010). The alternative is *informed* adaptation where one could, for example, use an external change detector (Gama et al. 2014) and weight new samples more if a concept drift was detected. We could also apply more sophisticated decay rules making γ a function of t or x_t.

A summary of characteristics for each proposed online learning variant of EXPoSE is listed in Table 2 and a general discussion can be found in literature, e.g. the work of Gama (2010).

Table 2 Comparison of online learning techniques for EXPoSE

	Pros	Cons
Prop. 1: online	✓ Equivalent to batch version	✗ No concept drift adaptation
Prop. 2: window	✓ Concept drift adaptation	✗ Possibly sudden changes in predictions
		✗ Difficult to choose window size
		✗ Increased memory requirements for window buffer
Prop. 3: decay	✓ Concept drift adaptation	
	✓ Gradual vanishing influence of outdated information	
	✓ No increased memory requirements	

5.2 Predictions on data streams

We introduced three different approaches to learn the model for EXPoSE on data streams. One incremental (online) learning approach and two evolving techniques. In order to make a prediction as a new observation is made we have to normalize the calculated predicted score. This is necessary as the score would continuously change, even if exactly the same data would be observed again. This problem is not present in the batch version of EXPoSE since the model does not change anymore at the time we make predictions. To avoid this problem we divide by the total volume

$$\begin{aligned}\int_{\mathcal{X}}\int_{\mathcal{X}} k(x,y)\,\mathrm{d}\mathbb{P}(x)\mathrm{d}\mathbb{P}(y) &= \langle \mu[\mathbb{P}]\mu[\mathbb{P}]\rangle \\ &\approx \langle w_t w_t\rangle\end{aligned}$$

yielding

$$\eta(z) = \frac{\langle \phi(z) w_t\rangle}{\|w_t\|^2},$$

as the EXPoSE classifier. We emphasize that the calculation of the normalization constant does not change the limiting behavior of runtime and memory we derived earlier in this section since we have constant time access to w_t anyway.

6 Approximate feature maps

We showed in the previous part how EXPoSE can be expressed as an inner product $\langle \phi(z), \mu[\mathbb{P}_n]\rangle$ between the kernel mean map and the feature map of a query point $z \in \mathcal{X}$ and derived a similar expression for the incremental and streaming variants of EXPoSE. However, the feature map ϕ (and hence $\mu[\mathbb{P}_n]$) can not always be calculated explicitly as

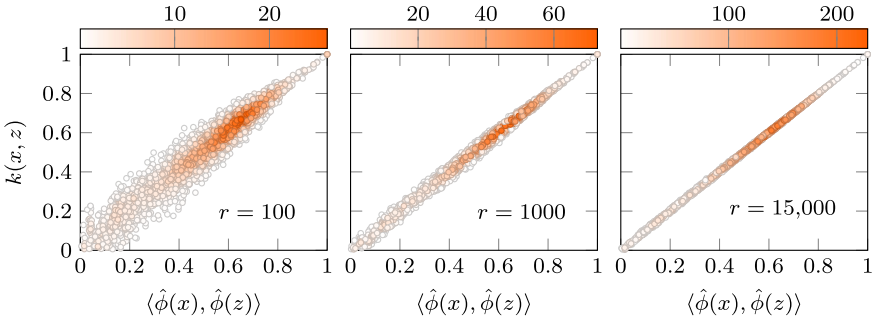

Fig. 3 A comparison between the values calculated by a Gaussian RBF kernel $k(x, z)$ and the RKS approximation $\left\langle\hat{\phi}(x)\hat{\phi}(z)\right\rangle$ for random images of the MNIST dataset. The individual plots show how the number of kernel expansions r affect the approximation quality. *Color* indicates the density

$\phi(z) = k(\cdot, z)$. One possible solution is to resort to approximate feature maps which we review in this section. The key idea behind approximate feature maps is to find a function $\hat{\phi}$ such that

$$k(x, z) \approx \left\langle\hat{\phi}(x)\hat{\phi}(z)\right\rangle$$

and $\hat{\phi}(x) \in \mathbb{R}^r$ for some $r \in \mathbb{N}$. We will see, that this can be done efficiently.

6.1 Random kitchen sinks

A way to efficiently create a feature map ϕ from a kernel k is known as *random kitchen sinks* (Rahimi and Recht 2007, 2008). The random kitchen sinks (RKS) approximation is based on Bochner's theorem for translation invariant kernels (such as Laplace, Matérn, Gaussian RBF, etc.) and states that such a kernel can be represented as

$$k(x, y) = \int_z \phi_z^\star(x)\phi_z(y)\lambda(z) \text{ with } \phi_z(x) = \mathrm{e}^{\mathrm{i}\langle zx\rangle},$$

where $\phi^\star$ is the conjugate transpose of ϕ. A Monte Carlo approximation of this integral can then be used to estimate the expression above as

$$k(x, y) \approx \frac{1}{r}\sum_{i=1}^{r}\phi_{z_i}^\star(x)\phi_{z_i}(y) = \left\langle\hat{\phi}(x)\hat{\phi}(y)\right\rangle \quad \text{with} \quad z_i \sim \lambda.$$

For kernels such as the Gaussian RBF $k(x, y) = \exp(-\frac{1}{2}\|x - y\|^2/\sigma^2)$, the measure λ can be found with the help of the inverse Fourier transform yielding

$$\hat{\phi}(x) = \frac{1}{\sqrt{r}}\exp(\mathrm{i}Zx) \quad \text{with} \quad Z_{ij} \sim \mathcal{N}(0, \sigma^2) \text{ and } Z \in \mathbb{R}^{r\times d}$$

where d is the input space dimension. The parameter $r \in \mathbb{N}$ determines the number of kernel expansions and is typically around 20,000. Larger r result in better kernel approximations as the Monte Carlo approach becomes more accurate (Fig. 3). Recently Le et al. (2013) proposed an approximation of Z such that the product Zx can be calculated in $\mathcal{O}(r \log d)$ time complexity while requiring only $\mathcal{O}(r)$ storage.

6.2 Nyström's approximation

An alternative to random kitchen sinks are Nyström methods (Williams and Seeger 2001) which project the data into a subspace $\mathcal{H}_r \subset \mathcal{H}$ spanned by $r \leq n$ randomly chosen elements $\phi(x_1), \ldots, \phi(x_r)$.

The Nyström feature map $\hat{\phi}$ is then given by $\hat{\phi}(x) = (\hat{\phi}_1(x), \ldots, \hat{\phi}_r(x))$ with

$$\hat{\phi}_i(x) = \frac{1}{\sqrt{\lambda_i}} \sum_{j=1}^{r} u_{ji} k(x_j, x), \qquad 1 \leq i \leq r,$$

where λ_i and u_i denote the ith eigenvalue and the i-th eigenvector of kernel matrix $K \in \mathbb{R}^{r \times r}$ with $K_{i,j} = k(x_i, x_j)$.

The Nyström approximation needs in general less basis functions, r, than the RKS approach (typically around 1000). However the approximation is data dependent and hence becomes erroneous if the underlying distribution changes or when we are not able to get independent samples from the dataset. This is a problem for online learning and streaming applications with concept drift. We therefore suggest to avoid the Nyström feature map in this context.

Random kitchen sinks and the Nyström approximation the most common feature map approximations. We refer to the corresponding literature for a discussion of other approximate feature maps such as Li et al. (2010), Vedaldi and Zisserman (2012) and Kar and Karnick (2012), which can be used as well for EXPoSE.

6.3 EXPoSE and approximate feature maps

Recall from the previous sections that EXPoSE uses the inner product $\langle \phi(z), \mu[\mathbb{P}_n] \rangle$ to calculate the score and make predictions. Using an approximate feature map $\hat{\phi}$, it is now possible to explicitly represent the feature function and consequently also the mean map $\mu[\mathbb{P}_n]$ as

$$\eta(z) \approx \left\langle \hat{\phi}(z) \frac{1}{n} \sum_{i=1}^{n} \hat{\phi}(x_i) \right\rangle \tag{1}$$

for the EXPoSE classifier.

We emphasize that with an efficient approximation of ϕ, as showed here, the training time of this algorithm is *linear* in the number of samples n and an evaluation of $\eta(z)$ for predictions takes only *constant time*. Moreover we need only $\mathcal{O}(r)$ memory to store the model which is also independent of n and the input dimension d.

7 Experimental evaluation

In this section we show in several experiments how EXPoSE compares to other state of the art anomaly detection techniques in prediction and runtime performances. We first explain which statistical test are used to compare the investigated algorithms.

7.1 Statistical comparison of algorithms

When comparing multiple (anomaly detection) algorithms over multiple datasets one cannot simply compare the raw numbers obtained from the area under *receiver operating char-*

acteristic (AUC) or *precision-recall* curves. Webb (2000) warns against averaging these numbers: "It is debatable whether error rates in different domains are commensurable, and hence whether averaging error rates across domains is very meaningful" (Webb 2000).

As Demšar (2006) points out, it is also dangerous to use tests which are designed to compare a pair of algorithms for more than two: "A common example of such questionable procedure would be comparing seven algorithms by conducting all 21 paired t-tests [...]. When so many tests are made, a certain proportion of the null hypotheses is rejected due to random chance, so listing them makes little sense." (Demšar 2006)

Demšar suggests to use the Friedman test with the corresponding post-hoc Nemenyi test for comparison of more classifiers over multiple data sets. A methodology we summarize in the following.

7.1.1 The Friedman test

The Friedman test (Friedman 1937) is a non-parametric statistical test which ranks algorithms for each dataset individually starting from 1 as the best rank. Its purpose is to examine whether there is a significant difference between the performances of the individual algorithms. Assume we compare k algorithms on m datasets and let r_{ij} be the rank of the jth algorithm on the ith dataset. We use $\bar{r}_j$ to denote the average rank of algorithm j given by $\bar{r}_j = m^{-1} \sum_i r_{ij}$. The Friedman statistic

$$\chi_F^2 = \frac{12m}{k(k+1)} \left(\sum_{j=1}^{k} \bar{r}_j^2 - \frac{k(k+1)^2}{4} \right)$$

is undesirably conservative and therefore Iman and Davenport (1980) suggest to use

$$F_F = \frac{(m-1)\chi_F^2}{m(k-1) - \chi_F^2}$$

which is distributed according to the F-distribution with $k-1$ and $(k-1)(m-1)$ degrees of freedom. If the null-hypothesis (all algorithms are equivalent) is rejected one can proceed with a post-hoc test.

7.1.2 The Nemenyi test

The Nemenyi test (Nemenyi 1963) is a post-hoc test to compare all (anomaly detection) algorithms with each other. Hereby the performance of two algorithms is significantly different if their average ranks differ by at least

$$\mathrm{CD} = q_\alpha \sqrt{\frac{k(k+1)}{6m}},$$

called the *critical difference*. Here q_α is the Studentised range statistic divided by $\sqrt{2}$.

Demšar (2006) also suggests to visually represent the results of the Nemenyi test in a critical difference diagram as in Fig. 4. In this diagram we compare 5 algorithms on 20 datasets against each other. Algorithms not connected by a bar have a significantly different performance.

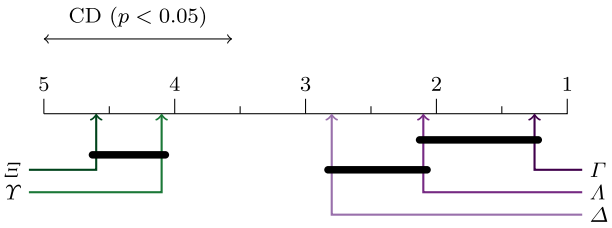

Fig. 4 Visualization of the post-hoc Nemenyi test in form of a critical difference diagram. The position on the *line* indicates the algorithms's average rank. Algorithms which are significantly different (at $p < 0.05$) are *not* connected with a bar. The critical difference (CD) is given by the *double arrow in the top left*

7.2 Batch anomaly detection

The aim of this experiment is to compare EXPoSE against *i*Forest, OC-SVM, LOF, KDE and FastAbod in terms of anomaly detection performance and processing time in a learning task without concept drift. In order to be comparable, we follow Liu et al. (2012) and perform an outlier selection task with the objective to identify anomalies in a given dataset.

7.2.1 Datasets

For performance analysis and evaluation we take the following datasets which are often used in literature for comparison of anomaly detection algorithms as for example in Schölkopf et al. (2001), Tax and Duin (2004), and Liu et al. (2012). We use several smaller benchmark datasets with known anomaly classes such as *Ionosphere*, *Arrhythmia*, *Pima*, *Satellite*, *Shuttle* (Lichman 2013), *Biomed* and Wisconsin Breast Cancer (*Breastw*) (Tax and Duin 2004). These datasets are set up as described in Liu et al. (2012) where all nominal and binary attributes are removed.

The larger datasets are the KDD CUP 99 network intrusion data (KDDCUP) and Forest Cover Type (ForestCover). For KDDCUP instances we follow the setup of (Yu et al. 2003) and obtain a total of 127 attributes. Furthermore, we add two high-dimensional image datasets MNIST and the Google Street View House Numbers (SVHN) (Netzer et al. 2011). We use the scaled version of MNIST (Chang and Lin 2011) and create HOG features (Vondrick et al. 2013) for SVHN. The methodology suggested by Schölkopf et al. (2001) and Tax (2001) is used to create anomaly detection datasets from MNIST and SVHN in the following way. We take all images of digit 1 from MNIST as normal instances. The images of the remaining digits $(2, 3, \ldots, 9)$ are used as anomalies. We then create a dataset comprising all normal instances and a random subset anomalies such that anomalies account for 1 % of the elements in the set. We repeat this process for MNIST images of digits $2, 3, \ldots, 9$ and do the same with the 9 digit classes of SVHN to create 18 anomaly detection datasets. The subset of anomalies is independently sampled for each repetition of an experiment. Table 3 provides an overview of the dataset properties and how the anomaly classes are defined.

In the experiment we provide a dedicated labeled random subset of 1 % or 2000 instances (whichever is smaller) to configure the algorithms parameters. We emphasize that this subset is not used to evaluate the predictive performance. The parameter configuration is done by a pattern search (Torczon 1997) using cross-validation. Examples of parameters being optimized are the number of nearest neighbors in LOF and FastAbod, the kernel bandwidth of EXPoSE, KDE and OC-SVM or the number of trees for *i*Forest (see Table 8 for a complete list of parameters which are optimized). We do not optimize over different distance metrics and

Table 3 Batch dataset properties

	Size (n)	Dim. (d)	C_A (anomaly class)	C_A proportion (%)
KddCup	1,036,241	127	"Attack"	0.3
ForestCover	286,048	10	Class 4 vs. 2	9
MNIST 1	101,968	784	2,3,4,5,6,7,8,9	1
MNIST 2	90,196	784	1,3,4,5,6,7,8,9	1
MNIST 3	92,763	784	1,2,4,5,6,7,8,9	1
MNIST 4	88,417	784	1,2,3,5,6,7,8,9	1
MNIST 5	82,062	784	1,2,3,4,6,7,8,9	1
MNIST 6	89,536	784	1,2,3,4,5,7,8,9	1
MNIST 7	94,771	784	1,2,3,4,5,6,8,9	1
MNIST 8	88,568	784	1,2,3,4,5,6,7,9	1
MNIST 9	90,034	784	1,2,3,4,5,6,7,8	1
SVHN 1	91,475	2592	2,3,4,5,6,7,8,9	1
SVHN 2	75,466	2592	1,3,4,5,6,7,8,9	1
SVHN 3	61,376	2592	1,2,4,5,6,7,8,9	1
SVHN 4	51,135	2592	1,2,3,5,6,7,8,9	1
SVHN 5	54,034	2592	1,2,3,4,6,7,8,9	1
SVHN 6	41,965	2592	1,2,3,4,5,7,8,9	1
SVHN 7	44,438	2592	1,2,3,4,5,6,8,9	1
SVHN 8	35,709	2592	1,2,3,4,5,6,7,9	1
SVHN 9	34,793	2592	1,2,3,4,5,6,7,8	1
Shuttle	58,000	9	Classes 2,3,4,5,7	6
Satellite	6435	36	Classes 2,4,5	32
Pima	768	8	"Pos"	35
Breastw	683	9	"Malignant"	35
Arrhythmia	452	274	Classes 3,4,5,7,8,9,14,15	14
Ionosphere	351	32	"Bad"	36
Biomed	194	5	"Carrier"	34

various kernels functions, but use the most common Euclidean distance and squared exponential kernel, respectively. However, we remark that the choice of these functions pose a possibility to include domain and expert knowledge into the system. Each experiment is repeated five times and their AUC scores are used to perform the Friedman test. Since *i*Forest is a randomized algorithm we conduct five trials in each repetition to get an average results. If not stated otherwise we use EXPoSE in combination with Nyström's approximation for batch anomaly detection and random kitchen sinks in the streaming experiments as discussed in Sect. 6.

Table 4 Batch anomaly detection performances [AUC]

	EXPoSE	*i*Forest	OC-SVM	LOF	KDE	FastAbod
KddCup	1.00	0.99	1.00	[a]	[b]	[b]
ForestCover	0.83	0.87	0.89	0.56	[b]	[b]
MNIST 1	1.00	0.99	1.00	0.97	[b]	[b]
MNIST 2	0.79	0.70	0.80	0.85	[b]	[b]
MNIST 3	0.86	0.70	0.80	0.88	[b]	[b]
MNIST 4	0.88	0.81	0.93	0.87	[b]	[b]
MNIST 5	0.89	0.69	0.82	0.89	[b]	[b]
MNIST 6	0.94	0.86	0.94	0.89	[b]	[b]
MNIST 7	0.92	0.88	0.92	0.89	[b]	[b]
MNIST 8	0.78	0.64	0.79	0.84	[b]	[b]
MNIST 9	0.89	0.82	0.90	0.90	[b]	[b]
SVHN 1	0.90	0.88	0.91	0.85	[b]	[b]
SVHN 2	0.88	0.76	0.78	0.78	[b]	[b]
SVHN 3	0.72	0.58	0.59	0.71	[b]	[b]
SVHN 4	0.85	0.74	0.75	0.83	[b]	[b]
SVHN 5	0.83	0.74	0.73	0.74	[b]	[b]
SVHN 6	0.84	0.79	0.80	0.87	[b]	[b]
SVHN 7	0.89	0.86	0.86	0.87	[b]	[b]
SVHN 8	0.83	0.76	0.75	0.88	[b]	[b]
SVHN 9	0.85	0.79	0.80	0.87	[b]	[b]
Shuttle	0.99	1.00	0.91	0.55	[b]	[b]
Satellite	0.79	0.70	0.62	0.57	0.78	0.74
Pima	0.68	0.68	0.62	0.59	0.67	0.65
Breastw	0.99	0.99	0.81	0.45	0.99	0.99
Arrythmia	0.79	0.80	0.71	0.68	0.74	0.79
Ionosphere	0.92	0.85	0.66	0.89	0.81	0.93
Biomed	0.87	0.83	0.76	0.69	0.88	0.88
Average rank	1.85	3.48	2.90	3.06	4.93	4.77

The underlined values indicate the best AUC/average rank
[a] Out of memory
[b] Execution time takes more than 2 days

7.2.2 Evaluation

The average scores for each experiment are reported in Table 4, whereas the runtimes are provided in Table 5. Some algorithms failed on the larger datasets. For example LOF was not

Table 5 Batch anomaly detection runtimes $[t] = s$

	EXPOSE	*i*Forest	OC-SVM	LOF	KDE	FastAbod
KDDCUP	44	70	22213	[a]	[b]	[b]
ForestCover	29	24	25901	47	[b]	[b]
MNIST 1	12	7	1976	23760	[b]	[b]
MNIST 2	11	9	2773	18717	[b]	[b]
MNIST 3	11	8	1991	20109	[b]	[b]
MNIST 4	11	8	1159	17412	[b]	[b]
MNIST 5	10	8	1892	15324	[b]	[b]
MNIST 6	10	8	3091	18208	[b]	[b]
MNIST 7	11	8	2727	20153	[b]	[b]
MNIST 8	11	8	1607	18217	[b]	[b]
MNIST 9	10	7	2383	18542	[b]	[b]
SVHN 1	24	11	9311	18192	[b]	[b]
SVHN 2	20	10	10371	18144	[b]	[b]
SVHN 3	16	9	14122	28508	[b]	[b]
SVHN 4	13	10	4044	19247	[b]	[b]
SVHN 5	14	11	10348	22359	[b]	[b]
SVHN 6	11	7	5389	13166	[b]	[b]
SVHN 7	11	8	6790	14906	[b]	[b]
SVHN 8	9	7	4210	9741	[b]	[b]
SVHN 9	9	6	3831	9290	[b]	[b]
Shuttle	3	7	38	24	[b]	[b]
Satellite	0	3	3	4	1	55
Pima	1	2	0	0	0	9
Breastw	1	3	0	0	0	4
Arrythmia	1	1	0	0	0	4
Ionosphere	1	3	0	0	0	0
Biomed	0	2	0	0	0	0

[a] Out of memory
[b] Execution time takes more than 2 days

able to process the KDDCUP dataset due to the high memory requirements of the tree data structure. However the advantage of a tree data structure for nearest neighbor lookup in low dimensions can be seen when comparing the runtime of LOF on ForestCover and MNIST. Even though the ForestCover dataset has more than 2.5 times the size of MNIST, it takes only a fraction of the time to be processed. This advantage vanishes in higher dimensions. KDE and FastAbod exhibit a good anomaly detection performance on small datasets, however fail as soon as we apply them to medium-sized problems.

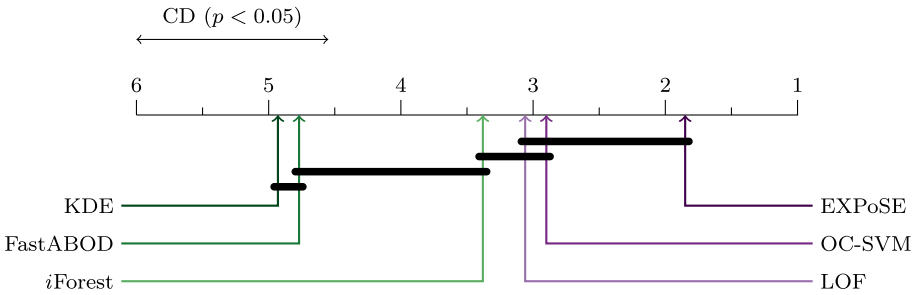

Fig. 5 Critical difference diagram of the batch anomaly detection performance. Algorithms which are not significantly different (at $p < 0.05$) are connected with a *bar*

With the AUC values we can perform the Friedman and post-hoc Nemenyi tests. The Friedman test confirms a statistical significant difference between the performances of the individual algorithms at a p value of 0.05. From the critical difference diagram in Fig. 5 we observe that EXPoSE performs significant better than *i*Forest, FastAbod and KDE. While no significant difference in terms of anomaly detection between EXPoSE, OC-SVM and LOF can be confirmed, EXPoSE is several orders of magnitude faster on large-scale, high-dimensional datasets.

7.3 Streaming anomaly detection

In this set of experiments we compare the streaming variants of EXPoSE against HSTa, STORM and COD. All of these algorithms are *blind* methods as they adapt their model at regular intervals without knowing if a concept drift occurred or not. They can be combined with a concept drift detector to make the adaptation *informed* (Gama 2010).

The evaluation of streaming algorithms is not as straightforward as the rating of batch learning techniques. There are two accepted techniques proposed in literature.

- Using a dedicated subset of the data (*holdout*) and evaluate the algorithm at regular time intervals. The holdout set must reflect the respective stream properties and therefore has to evolve with the stream in case of a concept drift.
- Making a prediction as the instance becomes available (*prequential*).[3] A performance metric can then be applied based on the prediction and the actual label of the instance. Since predictions are made on the stream directly there are no special actions which have to be taken in case of concept drift.

If possible, the holdout method is preferable since it is an unbiased risk estimator and we can use a balanced test set with the same number of normal instances and anomalies. This is a disadvantage of the prequential method since, by definition, the data stream contains only a few anomalies. This is problematic since STORM and COD assign hard class labels and, in contrast to AUC, the classification *accuracy* is highly sensitive to unbalanced data. We will therefore use the *balanced accuracy* defined as

$$\frac{0.5 \cdot \text{true positives}}{\text{true positives} + \text{false negatives}} + \frac{0.5 \cdot \text{true negatives}}{\text{true negatives} + \text{false positives}},$$

which compensates the unequal class distribution.

[3] Prequential originates from predictive and sequential (Dawid 1984).

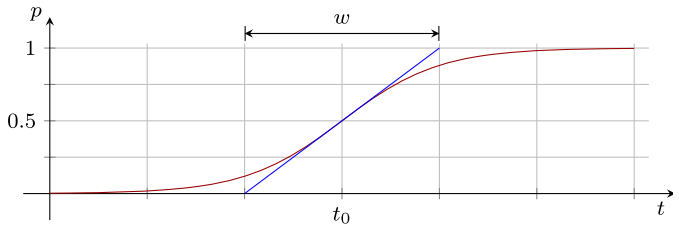

Fig. 6 The sigmoid function used to introduce a smooth drift from one concept to another. We use a Bernoulli distribution, where p is the probability to sample an instance from concept 1 and $(1 - p)$ is the probability to sample from concept 2. The drift occurs at t_0 and w defines the duration during which both concepts are valid

7.3.1 Datasets

There exist only a few non-synthetic datasets for anomaly detection with concept drift. Most of them are based on multi-class datasets, where each class represents a single concept. For example we use the SVHN dataset and stream 9000 randomly sampled instances of the digits 1–9 in sequence, such that the 1000 instances of digit 1 appear first, then 1000 instances of digit 2 until digit 9 (see Fig. 7). Every 25 time steps we calculate the accuracy using the holdout method for a dedicated random test set which contains 500 instances of the normal class and 500 instances of anomalies. Here, the normal class is the digit which is streamed at time step t and anomalies are all other classes. Likewise we proceed with the Satellite and Shuttle datasets.

Similar, Ho (2005) proposed the *three digit data stream* (TDDS) which contains four different concepts. Each concept consists of three digits of the USPS handwritten digits dataset as described.[4] After all instances of concept 1 are processed, the stream switches to the second concept and so on until concept 4. We randomly induce 1 % anomalies to each concept and use the prequential method for evaluation to calculate the balanced accuracy.

All datasets presented so far contain one or more *sudden* (abrupt) concept drifts. Bifet et al. (2009) proposed a methodology to introduce a *smooth* (incremental) drift between two concepts. The instances of the concepts from two classes under consideration are sampled according to a Bernoulli distribution where the class probability smoothly changes from one class to the other according to a sigmoid function (Fig. 6). The concept drift occurs at t_0 and w is the length of the drift interval. During this interval the instances of both concepts belong to the class of normal data. We apply this methodology to USPS and create the *smooth digit drift* (SDD) dataset. We start with digit 1 and then smoothly change to digit 2 at $t_0 = 500$ using $w = 100$. The next drift to digit 3 occurs at $t_0 = 1000$ and we repeat this until digit 9. As before, we randomly add 1 % anomalies to each concept and use the prequential method for evaluation. We summarized the dataset characteristics in Table 6.

7.3.2 Evaluation

In the following we will denote EXPoSE with a sliding window (Sect. 5.1.1) and EXPoSE with gradual forgetting (Sect. 5.1.2) by w-EXPoSE and γ-EXPoSE, respectively.

A sliding window of length 100 demonstrated to obey an appropriate trade off between drift adaptation and model accuracy. We therefore use this length for all algorithms and all datasets except γ-EXPoSE. A change of the window length affects w-EXPoSE, COD,

[4] See Ho (2005) for a detailed description of the TDDS dataset.

Table 6 Streaming dataset properties

	#concepts	Concept drift type	Evaluation method
SVHN	9	Sudden	Holdout
Satellite	3	Sudden	Holdout
Shuttle	2	Sudden	Holdout
TDDS	4	Sudden	Prequential
SDD	9	Smooth	Prequential

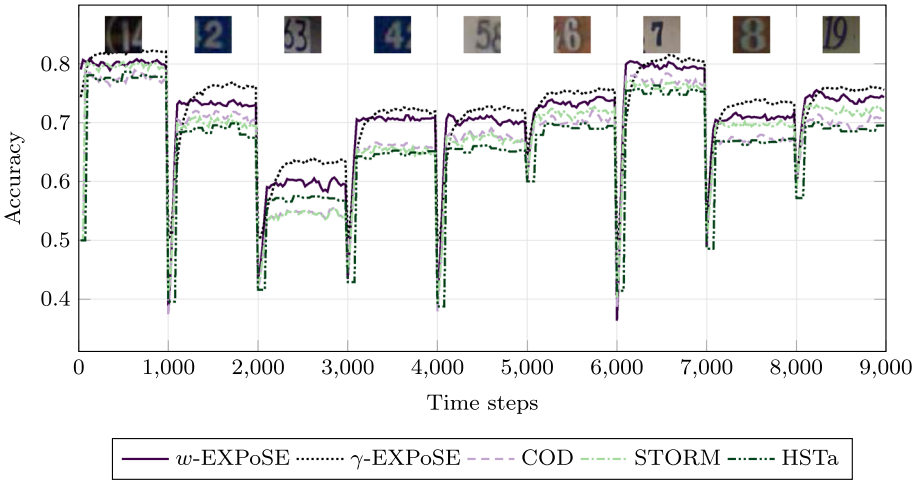

Fig. 7 The streaming SVHN experiment. A comparison of prediction accuracy under concept drift averaged over 5 repetitions

STORM and HSTa in the same way. This is not unexpected as the window size determines the number of instances available to the algorithm. The first 100 instances of each stream are used to configure algorithm parameters (see Table 8 in the "Appendix") via cross-validation using pattern search.

A detailed illustration of the SVHN experiment is shown in Fig. 7. The predictive performance of all algorithms is relatively similar. It can be observed that, as the stream changes from one digit to another, the accuracy suddenly drops which indicates that the current model is not valid anymore. After a short period of time, the model adapts and the accuracy recovers. EXPoSE performs on average better than COD, STORM and HSTa. A possible interpretation of this result is the sound foundation in probability theory of our approach. The suboptimal performance of HSTa indicates the random binary trees constructed by HSTa are not sufficiently fine-grained for this high-dimensional datasets. This interpretation is supported by the experiments with the low-dimensional Shuttle and Satellite data, where HSTa performs better.

The average over all accuracies of the individual experiments can be found in Table 7. The only statistical significance (at $p < 0.05$) is observed between γ-EXPoSE and COD.

Table 7 Streaming anomaly detection performance [accuracy]

	w-EXPoSE	γ-EXPoSE	STORM	COD	HSTa
Shuttle	0.88	0.88	0.75	0.74	<u>0.89</u>
Satellite	<u>0.89</u>	0.88	0.78	0.79	0.88
SVHN	0.71	<u>0.73</u>	0.68	0.68	0.66
TDDS	0.71	<u>0.71</u>	0.67	0.64	0.67
SDD	0.83	<u>0.85</u>	0.79	0.76	0.77
Average rank	1.80	<u>1.70</u>	3.80	4.50	3.20

The underlined values indicate the best AUC/average rank

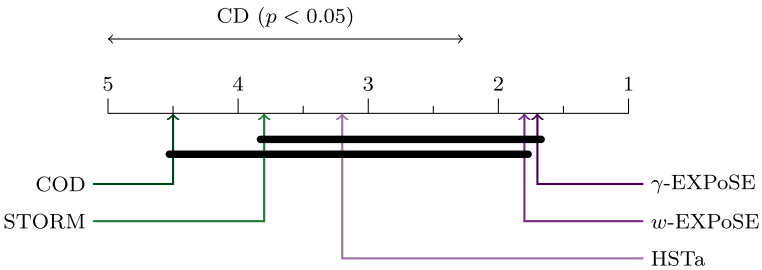

Fig. 8 Critical difference diagram of the stream anomaly detection performance. Algorithms which are not significantly different (at $p < 0.05$) are connected with a *bar*

We could not confirm a significant difference between the other algorithms as illustrated in the critical difference diagram (Fig. 8).

Although these results are promising we recommend to combine the techniques presented here with a concept drift detection technique to make *informed* model updates (Gama 2010).

8 Conclusion

We proposed a new algorithm, EXPoSE, to perform anomaly detection on very large-scale datasets and streams with concept drift. Although anomaly detection is a problem of central importance in many applications, only a few algorithms are scalable to the vast amount of data we are often confronted with.

The EXPoSE anomaly detection classifier calculates a score (the likelihood of a query point belonging to the class of normal data) using the inner product between a feature map and the kernel embedding of probability measures. The kernel embedding technique provides an efficient way to work with probability measures without the necessity to make assumptions about the underlying distributions.

Despite its simplicity EXPoSE obeys a *linear* computational complexity for learning and can make predictions in *constant* time while it requires only *constant* memory. When applied incrementally or online, a model update can also be performed in *constant* time. We demonstrated that EXPoSE can be used as an efficient anomaly detection algorithm with the

same predictive performance as the best state of the art methods while being significant faster than techniques with the same discriminant power.

Appendix

Definition 9 (*Strong Integral*) Let $(\mathcal{X}, \mathscr{X}, \mathbb{P})$ be a σ-finite measure space and let $\phi\colon \mathcal{X} \to \mathcal{H}$ be measurable. Then ϕ is *strong integrable* (*Bochner integrable*) over a set $\mathcal{D} \in \mathcal{X}$ if and only if its norm $\|\phi\|$ is Lebesgue integrable over $\mathcal{D}$, that is,

$$\int_{\mathcal{D}} \|\phi\| \, \mathrm{d}\mathbb{P}(x) < \infty.$$

If ϕ is strong integrable over each $\mathcal{D} \in \mathcal{X}$ we say that ϕ is *strong integrable* (Aliprantis and Border 2006, Theorem 11.44).

Definition 10 (*Weak Integral*) Let $(\mathcal{X}, \mathscr{X}, \mathbb{P})$ be a σ-finite measure space. A function $\phi\colon \mathcal{X} \to \mathcal{H}$ is *weakly integrable* over a set $\mathcal{D} \in \mathcal{X}$ if there exists some $\lambda \in \mathcal{H}$ satisfying

$$\langle f, \lambda \rangle = \int_{\mathcal{D}} \langle f, \phi(x) \rangle \, \mathrm{d}\mathbb{P}(x)$$

for each $f \in \mathcal{H}$. The weak integral is denoted by

$$\lambda = \oint_{\mathcal{D}} \phi \, \mathrm{d}\mathbb{P}(x)$$

and the unique element $\lambda \in \mathcal{H}$ is called *weak integral* of ϕ over $\mathcal{D}$. If the integral exists for each $\mathcal{D} \in \mathcal{X}$ we say that ϕ is *weakly integrable* (Aliprantis and Border 2006, Section 11.10; Table 8).

Table 8 Algorithms and Parameters

	Optimized parameters
EXPoSE	Kernel bandwidth
*i*Forest	Number of trees, sub-sampling size
OC-SVM	Kernel bandwidth, anomaly fraction
LOF	Number of nearest neighbors
KDE	Kernel bandwidth
FastAbod	Number of nearest neighbors
STORM	Number of nearest neighbors, neighborhood radius
COD	Number of nearest neighbors, neighborhood radius
HSTa	Number of trees

References

Aleskerov, E., Freisleben, B., & Rao, B. (1997). Cardwatch: A neural network based database mining system for credit card fraud detection. In *Computational intelligence for financial engineering*. doi:10.1109/cifer.1997.618940

Aliprantis, C. D., & Border, K. (2006). *Infinite dimensional analysis: A hitchhiker's guide*. Berlin: Springer Science & Business Media. doi:10.1007/3-540-29587-9.

Angiulli, F., & Fassetti, F. (2007). Detecting distance-based outliers in streams of data. In *Proceedings of the sixteenth ACM conference on conference on information and knowledge management* (pp. 811–820). New York: ACM. doi:10.1145/1321440.1321552.

Angiulli, F., & Fassetti, F. (2010). Distance-based outlier queries in data streams: The novel task and algorithms. *Data Mining and Knowledge Discovery*, *20*(2), 290–324. doi:10.1007/s10618-009-0159-9.

Angiulli, F., & Pizzuti, C. (2002). Fast outlier detection in high dimensional spaces. In *PKDD*, Vol. 2 (pp. 15–26). Berlin: Springer. doi:10.1007/3-540-45681-3_2.

Bifet, A., Holmes, G., Pfahringer, B., Kirkby, R., & Gavaldà, R. (2009). New ensemble methods for evolving data streams. In *Proceedings of the 15th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 139–148). ACM. doi:10.1145/1557019.1557041.

Breunig, M. M., Kriegel, H. P., Ng, R. T., & Sander, J. (2000). LOF: Identifying density-based local outliers. In *ACM sigmod record*, Vol. 29 (pp. 93–104). ACM. doi:10.1145/335191.335388.

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*(3), 1–27.

Dawid, A. P. (1984). Present position and potential developments: Some personal views: Statistical theory: The prequential approach. *Journal of the Royal Statistical Society Series A (General)* 278–292. doi:10.2307/2981683.

Dean, J., & Ghemawat, S. (2008). MapReduce: Simplified data processing on large clusters. *Communications of the ACM*, *51*(1), 107–113. doi:10.1145/1327452.1327492.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research*, *7*, 1–30.

Domingos, P., & Hulten, G. (2000). Mining high-speed data streams. In *Proceedings of the sixth ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 71–80). ACM. doi:10.1145/347090.347107.

Domingos, P., & Hulten, G. (2001). Catching up with the data: Research issues in mining data streams. In *DMKD*.

Faria, E.R., Gama, J., & Carvalho, A.C. (2013). Novelty detection algorithm for data streams multi-class problems. In *Proceedings of the 28th annual ACM symposium on applied computing* (pp. 795–800). ACM. doi:10.1145/2480362.2480515.

Friedman, M. (1937). The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, *32*(200), 675–701. doi:10.1080/01621459.1937.10503522.

Gama, J. (2010). *Knowledge discovery from data streams*. Boca Raton: CRC Press. doi:10.1201/ebk1439826119-c1.

Gama, J., Žliobaite, I., Bifet, A., Pechenizkiy, M., & Bouchachia, A. (2014). A survey on concept drift adaptation. *ACM Computing Surveys (CSUR)*, *46*(4), 44. doi:10.1145/2523813.

Goodman, J. E., & O'Rourke, J. (2004). *Handbook of discrete and computational geometry* (Vol. 2). Boca Raton: CRC Press.

Gretton, A., & Desobry, F. (2003). On-line one-class support vector machines. An application to signal segmentation. In *Acoustics, speech, and signal processing, 2003. Proceedings (ICASSP'03)*. 2003 IEEE international conference on, IEEE, vol. 2, pp. 2–709. doi:10.1109/icassp.2003.1202465.

Gretton, A., Borgwardt, K. M., Rasch, M. J., Schölkopf, B., & Smola, A. (2012). A kernel two-sample test. *The Journal of Machine Learning Research*, *13*(1), 723–773.

Gupta, M., Gao, J., & Aggarwal, C. C. (2014). Outlier detection for temporal data: A survey. *Knowledge and Data Engineering, IEEE Transactions on*, *26*(9), 2250–2267. doi:10.1109/tkde.2013.184.

Ho, S. S. (2005). A martingale framework for concept change detection in time-varying data streams. In *Proceedings of the 22nd international conference on machine learning* (pp. 321–327). ACM. doi:10.1145/1102351.1102392.

Hodge, V. J., & Austin, J. (2004). A survey of outlier detection methodologies. *Artificial Intelligence Review*, *22*(2), 85–126. doi:10.1023/b:aire.0000045502.10941.a9.

Iman, R. L., & Davenport, J. M. (1980). Approximations of the critical region of the Friedman statistic. *Communications in Statistics-Theory and Methods*, *9*(6), 571–595.

Kallenberg, O. (2006). *Foundations of modern probability*. Berlin: Springer Science & Business Media. doi:10.1007/b98838.

Kar, P., & Karnick, H. (2012). Random feature maps for dot product kernels. In *International conference on artificial intelligence and statistics*, pp. 583–591.

Knorr, E. M., & Ng, R. T. (1998). Algorithms for mining distance-based outliers in large datasets. In *Proceedings of the international conference on very large data bases, citeseer* (pp. 392–403).

Knorr, E. M., Ng, R. T., & Tucakov, V. (2000). Distance-based outliers: Slgorithms and applications. *The VLDB Journal the International Journal on Very Large Data Bases*, *8*(3–4), 237–253. doi:10.1007/s007780050006.

Kontaki, M., Gounaris, A., Papadopoulos, A. N., Tsichlas, K., & Manolopoulos, Y. (2011). Continuous monitoring of distance-based outliers over data streams. In *Data engineering (ICDE), 2011 IEEE 27th international conference on, IEEE*, pp. 135–146. doi:10.1109/icde.2011.5767923.

Kriegel, H. P., & Zimek, A. (2008). Angle-based outlier detection in high-dimensional data. In *Proceedings of the 14th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 444–452). ACM. doi:10.1145/1401890.1401946.

Kumar, V. (2005). Parallel and distributed computing for cybersecurity. *IEEE Distributed Systems Online*, *6*(10), 1. doi:10.1109/mdso.2005.53.

Lazarescu, M. M., Venkatesh, S., & Bui, H. H. (2004). Using multiple windows to track concept drift. *Intelligent Data Analysis*, *8*(1), 29–59.

Le, Q., Sarlós, T., & Smola, A. J. (2013). Fastfood: Approximating kernel expansions in loglinear time. In *Proceedings of the international conference on machine learning*.

Li, F., Ionescu, C., & Sminchisescu, C. (2010). Random Fourier approximations for skewed multiplicative histogram kernels. In *Pattern recognition* Accessed 1 Jan 2016. Berlin: Springer. doi:10.1007/978-3-642-15986-2_27.

Lichman, M. (2013). *UCI machine learning repository*. Irvine: University of California, School of Information and Computer Sciences. http://archive.ics.uci.edu/ml. Accessed 1 Jan 2016.

Liu, F. T., Ting, K. M., & Zhou, Z. H. (2012). Isolation-based anomaly detection. *ACM Transactions on Knowledge Discovery from Data*, *6*(1), 3. doi:10.1145/2133360.2133363.

Nemenyi, P. (1963). Distribution-free multiple comparisons. Ph.D. thesis, Princeton University.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., & Ng, A. (2011). Reading digits in natural images with unsupervised feature learning. In NIPS workshop on deep learning and unsupervised feature learning, p. 4.

Ott, L., Pang, L., Ramos, F. T., & Chawla, S. (2014). On integrated clustering and outlier detection. In *Advances in neural information processing systems* (pp. 1359–1367).

Pokrajac, D. (2007). Incremental local outlier detection for data streams. IEEE symposium on computational intelligence and data mining (pp. 504–515). doi:10.1109/cidm.2007.368917.

Rahimi, A., & Recht, B. (2007). Random features for large-scale kernel machines. In *Advances in neural information processing systems* (pp. 1177–1184).

Rahimi, A., & Recht, B. (2008). Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning. In *Advances in neural information processing systems* (pp. 1313–1320).

Ramaswamy, S., Rastogi, R., & Shim, K. (2000). Efficient algorithms for mining outliers from large data sets. In *ACM SIGMOD record*, Vol. 29, pp. 427–438. ACM. doi:10.1145/335191.335437.

Sadik, S., & Gruenwald, L. (2014). Research issues in outlier detection for data streams. *ACM SIGKDD Explorations Newsletter*, *15*(1), 33–40. doi:10.1145/2594473.2594479.

Schneider, M. (2016). Probability inequalities for kernel embeddings in sampling without replacement. In *Proceedings of the nineteenth international conference on artificial intelligence and statistics*.

Schneider, M., Ertel, W., & Palm, G. (2015). Expected similarity estimation for large scale anomaly detection. In *International joint conference on neural networks, IEEE*, pp. 1–8. doi:10.1109/ijcnn.2015.7280331.

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. doi:10.1162/089976601750264965.

Sejdinovic, D., Sriperumbudur, B., Gretton, A., & Fukumizu, K. (2013). Equivalence of distance-based and RKHS-based statistics in hypothesis testing. *The Annals of Statistics*, *41*(5), 2263–2291. doi:10.1214/13-aos1140.

Smola, A. J., Gretton, A., Song, L., & Schölkopf, B. (2007). A Hilbert space embedding for distributions. In *Algorithmic learning theory*. Berlin: Springer. doi:10.1007/978-3-540-75488-6_5.

Spence, C., Parra, L., & Sajda, P. (2001). Detection, synthesis and compression in mammographic image analysis with a hierarchical image probability model. In *Mathematical methods in biomedical image analysis, 2001. MMBIA 2001. IEEE workshop on, IEEE*, pp. 3–10. doi:10.1109/mmbia.2001.991693.

Spinosa, E. J., de Leon, F., de Carvalho, A. P., & Gama, J. (2007). OLINDDA: A cluster-based approach for detecting novelty and concept drift in data streams. In *Proceedings of the 2007 ACM symposium on applied computing* (pp. 448–452). ACM.

Steinwart, I. (2003). Sparseness of support vector machines. *The Journal of Machine Learning Research*, *4*, 1071–1105.

Steinwart, I., & Christmann, A. (2008). *Support vector machines*. Berlin: Springer.

Tan, S. C., Ting, K. M., & Liu, T. F. (2011). Fast anomaly detection for streaming data. In *IJCAI proceedings-international joint conference on artificial intelligence, Citeseer*, Vol. 22, p. 1511.

Tax, D. M. J. (2001). One-class classification. Ph.D. thesis, Technische Universiteit Delft.

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

Torczon, V. (1997). On the convergence of pattern search algorithms. *SIAM Journal on Optimization*, *7*(1), 1–25.

Vedaldi, A., & Zisserman, A. (2012). Efficient additive kernels via explicit feature maps. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, *34*(3), 480–492. doi:10.1109/cvpr.2010.5539949.

Vondrick, C., Khosla, A., Malisiewicz, T., & Torralba, A. (2013). Hoggles: Visualizing object detection features. In Computer vision (ICCV), 2013 IEEE international conference on, IEEE, pp. 1–8. doi:10.1109/iccv.2013.8.

Webb, G. I. (2000). Multiboosting: A technique for combining boosting and wagging. *Machine Learning*, *40*(2), 159–196.

Widmer, G., & Kubat, M. (1996). Learning in the presence of concept drift and hidden contexts. *Machine Learning*, *23*(1), 69–101. doi:10.1007/bf00116900.

Williams, C., & Seeger, M. (2001). Using the Nyström method to speed up kernel machines. In *Proceedings of the 14th annual conference on neural information processing systems*. MIT Press, EPFL-CONF-161322, pp. 682–688.

Wu, K., Zhang, K., Fan, W., Edwards, A., & Yu, P. S. (2014). RS-forest: A rapid density estimator for streaming anomaly detection. In *Data mining (ICDM), 2014 IEEE international conference on, IEEE*, pp. 600–609. doi:10.1109/icdm.2014.45.

Xiong, L., Poczos, B., Schneider, J., Connolly, A., & Vander Plas, J. (2011). Hierarchical probabilistic models for group anomaly detection. In *AISTATS 2011*.

Yamanishi, K., Takeuchi, J. I., Williams, G., & Milne, P. (2004). On-line unsupervised outlier detection using finite mixtures with discounting learning algorithms. *Data Mining and Knowledge Discovery*, *8*(3), 275–300. doi:10.1145/347090.347160.

Yu, H., Yang, J., & Han, J. (2003). Classifying large data sets using SVMs with hierarchical clusters. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 306–315). ACM. doi:10.1145/956750.956786.

Zhang, B., Sconyers, C., Byington, C., Patrick, R., Orchard, M. E., & Vachtsevanos, G. (2011). A probabilistic fault detection approach: Application to bearing fault detection. *IEEE Transactions on Industrial Electronics*, *58*(5), 2011–2018. doi:10.1109/tie.2010.2058072.