Anomaly Detection in Streams with Extreme Value Theory

Alban Siffer
Amossys, IRISA, Inria
alban.siffer@irisa.fr

Pierre-Alain Fouque
Univ. Rennes 1, IUF, IRISA
pierre-alain.fouque@inria.fr

Alexandre Termier
Univ. Rennes 1, Inria, IRISA
alexandre.termier@irisa.fr

Christine Largouet
AgroCampus, Inria, IRISA
christine.largouet@irisa.fr

ABSTRACT

Anomaly detection in time series has attracted considerable attention due to its importance in many real-world applications including intrusion detection, energy management and finance. Most approaches for detecting outliers rely on either manually set thresholds or assumptions on the distribution of data according to Chandola, Banerjee and Kumar.

Here, we propose a new approach to detect outliers in streaming univariate time series based on Extreme Value Theory that does not require to hand-set thresholds and makes no assumption on the distribution: the main parameter is only the risk, controlling the number of false positives. Our approach can be used for outlier detection, but more generally for automatically setting thresholds, making it useful in wide number of situations. We also experiment our algorithms on various real-world datasets which confirm its soundness and efficiency.

CCS CONCEPTS

• **Computing methodologies** → **Anomaly detection**; • **Mathematics of computing** → *Time series analysis*; • **Information systems** → Data stream mining;

KEYWORDS

Outliers in time series, Extreme Value Theory, Streaming

1 INTRODUCTION

Anomaly detection is an important research area in data mining. Many types of anomalies, or *outliers*, are described in the literature [23]. One of the most fundamental type of anomalies are the extreme values (maximum and minimum).

Many work have been proposed for solving this problem. However, they require some knowledge about the data: either they make assumptions on the underlying distribution or they need manually set thresholds.

When the data is static, or is a stream coming from an extremely controlled environment, such assumptions can safely be made. But in the general case of streaming data from an open environment, these assumptions are no longer true. They may fail in unexpected cases.

The issue is that nowadays, more and more critical applications rely on *high throughput streaming numerical data* like energy management [30], cyber-security [32] or finance [26]. For example, in intrusion detection, some network attack techniques rely on intensive scans of the network, which are characterized by an unusually high number of SYN packets [32].

The main challenge is to learn "normality" in an ever changing environment and to automatically adapt the detection method accordingly.

The problem of detecting extreme values in streams can be expressed as follows: Let $(X_t)_{t\geq 0}$ be a streaming time series of iid observations. Can we set a threshold z_q such that for any $t \geq 0$, the probability to observe $X_t > z_q$ is lower than q (for q as small as desired) ?

To solve this problem, we use the statistical powerful tool of *Extreme Value Theory* (EVT). This theory was developed to study the law of extreme values in a distribution function after the following dramatic event. In the night of January 31 to February 1 of the year 1953, a set of rare conditions occurred in the North Sea, leading to a "perfect storm" scenario. On the coast of Netherlands, the waves generated overwhelmed the dikes, causing extensive flooding. The flooding led to the death of 1800+ people in the Netherlands alone. In the dike case, X_t is the height of the waves, and z_q is the height of the dike.

In the aftermath of this disaster, scientists were tasked to determine a minimal dike height such that the probability for waves to exceed this height is extremely low. Statisticians devised an elegant theory for the study of such rare events [10]. One of the most elegant result of EVT is that the distribution of the extreme values is almost independent of the distribution of the data with a theorem similar to the central limit theorem, for min, max instead of the mean value.

The main contribution of this paper is to propose an approach for outlier detection in high throughput streaming univariate and unimodal time series. Thanks to EVT, our approach makes no distribution assumption on the data: it is thus a solution to "Research Issue 6" for outlier detection in data streams as stated by Sadik and Gruenwald [29] in *SIGKDD Explorations 2014*. We decline our approach into two algorithms: SPOT for streaming data having any stationary distribution, and DSPOT for streaming data that can be

KDD '17, August 13-17, 2017, Halifax, NS, Canada

ACM ISBN 978-1-4503-4887-4/17/08…$15.00
https://doi.org/10.1145/3097983.3098144

subject to concept drift. Through detailed experiments on synthetic and real data, we show that our approach is accurate for detecting outliers, is computationally efficient, and for DSPOT reacts quickly to any change in the stream. For instance, we decide to test our algorithm on incoming streams without knowledge on their distribution. We show that we detect very efficiently and accurately: (*i*) network SYN attacks on a labeled data stream and (*ii*) peaks that allows to take decision on stock market (quickly react for buying or selling shares). Our experiments also confirm the EVT theory with accuracy and fast convergence.

Analyzing streaming data require the computation of EVT to be fast and resilient: as a secondary contribution, we propose two improvements on the general method for solving the EVT problem, that improve both its speed and its robustness. They are used in our algorithms, but they are not specific to streaming data and can immediately be applied to most algorithms using EVT.

2 RELATED WORK

Classically, anomaly detectors have to highlight what will be considered as an anomaly, also called *outlier*. As we propose a statistical method to find anomalies, we rely on the assumption given by Chandola, Banerjee and Kumar in [13]: "Normal data instances occur in high probability regions of a stochastic model, while anomalies occur in the low probability regions of the stochastic model".

A great deal of algorithms for static outlier detection are given in the literature. The main approaches are distance based [7], nearest-neighbor based [11] or clustering based [14] and are very well detailed in [13]. Nonetheless, as mentioned in [31], most existing outlier detection methods need to scan several times the data and/or have high time complexity, thus they cannot be used in data streams.

In [28, 29], Sadik details the specificities of the stream environment and the hardships of outlier detection in this context. The main constraints are the following: data cannot be scanned twice and new concepts may keep evolving.

Many works which address the streaming case present distance based algorithms for outlier detection (STORM [8], CORM [15], DBOD-DS [28], attributes weighting [31]). These methods are able to work on multidimensional streams with categorical features but they need user defined thresholds which could be a real hindrance in practice.

Current statistical approaches to perform outlier detection in data stream suffer from the inherent problem, namely the distribution assumption. In [6], Agarwal assumes a gaussian model to detect anomalies in multidimensional arrays and recommends a Box-Cox transformation if it is not the case. In [16], a more general mixture model is presented by Eskin, based on a majority distribution and an anomaly one. However both models need to be learned, so data of each distribution are required. In [24], the authors use a probabilist threshold ϵ to discriminate normal or abnormal data (observations with probability lower than ϵ are anomalies), but its possible values are lower-bounded by $1/(k+1)$ where k is the number of training elements. Thus it needs a huge training sample if we want a very low false positive rate.

In our work, we do not assume the distribution of the value we monitor but we rely on powerful theoretical results to estimate accurately low probability areas and then discriminate outliers. With our single parameter algorithms, we are able to detect outliers in both stationary and drifting contexts.

3 BACKGROUND

In this section we describe the theoretical background of the Extreme Value Theory (EVT). We try to explain the main results and how they could be used to address our problem (the reader could refer to the rich reference of Beirlant *et al.* [10] for more details). This part is not a prerequisite to understand the purpose of our algorithm but it gathers some elements to precise its fundamental basis.

Many techniques allow the scientist to find statistical thresholds (quantiles). For instance, we can compute them empirically or assume a distribution. However data do not necessarily follow well-known distributions (Gaussian, uniform, exponential etc.) so the model step (the choice of the distribution) could be hard, even inappropriate. Moreover, if we want to predict *extreme* events, like rare or unprecedented events (as tidal waves), the empirical method will not give accurate estimation (an unprecedented event would have a probability equal to zero). The extreme value theory addresses these problems by inferring the distribution of the extreme events we might monitor, without strong hypothesis on the original distribution.

Mathematically, X is a random variable and F its cumulative distribution function: $F(x) = \mathbb{P}(X \leq x)$. We denote by $\bar{F}$ the "tail" of the distribution: $\bar{F}(x) = 1 - F(x) = \mathbb{P}(X > x)$. We use X_i to denote both random variables and their outcomes, however the context will precise their meanings. For a random variable X and a given probability q we note z_q its quantile at level $1-q$, i.e. z_q is the smallest value s.t. $\mathbb{P}(X \leq z_q) \geq 1-q$ i.e. $\mathbb{P}(X > z_q) < q$.

3.1 Extreme value distributions

The goal of the extreme value theory is to find the law of extreme events (e.g. the law of the daily maximum of temperature, or the law of the monthly maximal tide height). A beautiful result from Fisher, Tippett [18] and later Gnedenko [20] states that, under a weak condition, these extreme events have the same kind of distribution, regardless of the original one. For instance the maximum of temperatures or tide heights have more or less the same distribution whereas the distributions of the temperatures and the tide heights are not likely to be the same. This extreme laws are called the Extreme Value Distributions (EVD) and they have the following form :

$$G_\gamma : x \mapsto \exp\left(-(1+\gamma x)^{-\frac{1}{\gamma}}\right), \quad \gamma \in \mathbb{R}, \quad 1+\gamma x > 0.$$

All the extremes of common standard distributions follow such a distribution and the *extreme value index* γ depends on this original law. For example, if $X_1, \ldots X_n$ are n iid variables (e.g. gaussian $\mathcal{N}(0,1)$) then $M_n = \max_{1 \leq i \leq n} X_i$ is likely to follow an EVD which extreme value index γ is given by the initial distribution (for the Gaussian distribution $\gamma = 0$).

This result may seem very counterintuitive but we can give some elements to catch the idea. Indeed, we can easily imagine that for most distributions the probabilities decrease when events are extreme, ie $\mathbb{P}(X > x) \to 0$ when x increases. The function $\bar{F}(x) = \mathbb{P}(X > x)$ represents the *tail* of the distribution of X. Actually, there

are not many possible shapes for this tail and G_γ tries to fit them. The table 1 presents the three possible shapes of the tail and the link with the extreme value index γ. It gives also an example of standard distribution which follows each tail behavior. The parameter τ represents the bound of the initial distribution, so it could be finite (ex: uniform cdf) or infinite (ex: normal cdf). The figure 1 depicts an example of the three behaviors.

Tail behavior ($x \to \tau$)	Domain	Example
Heavy tail, $\mathbb{P}(X > x) \simeq x^{-\frac{1}{\gamma}}$	$\gamma > 0$	Frechet
Exponential tail, $\mathbb{P}(X > x) \simeq e^{-x}$	$\gamma = 0$	Gamma
Bounded, $\mathbb{P}(X > x) \underset{x \geq \tau}{=} 0$	$\gamma < 0$	Uniform

Table 1: Relation between F and γ

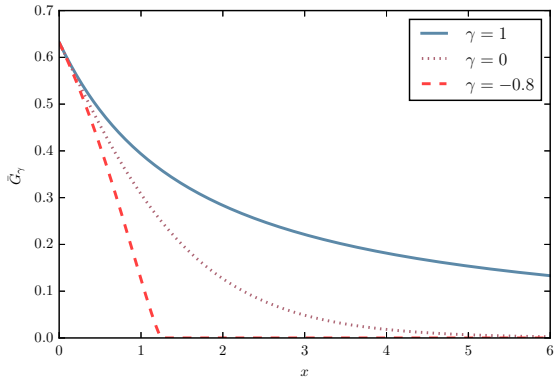

Figure 1: Tail distribution $\bar{G}_\gamma$ according to γ

3.2 Power of EVT

This phenomenon allows us to accurately compute probabilities without inferring the initial law that can be really complex. It "regularizes" the initial distribution. Indeed, the central limit theorem states that the mean of n iid random variables converges in distribution to a normal distribution. The EVT theorem states the same result for the maximum.

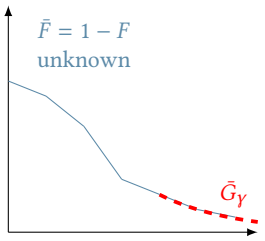

Figure 2: EVD fit of an unknown cdf

By fitting an EVD to the unknown input distribution tail (see figure 2), it is then possible to evaluate the probability of potential extreme events. In particular, from a given probability q it is possible to calculate z_q such that $\mathbb{P}(X > z_q) < q$. To solve this problem, the natural way will be to estimate γ. Several estimates exist such as Hill's estimate [22] and Pickands' estimate [27] but they give good results only for certain tail behaviors. Its estimation is hard and nowadays we do not know a general and efficient method to compute it (i.e. for all $\gamma \in \mathbb{R}$).

Another way exists to fit the tail of the distribution: the Peaks-Over-Threshold (POT) approach.

3.3 Peaks-Over-Threshold (POT) approach

The Peaks-Over-Threshold (POT) approach relies on the Pickands-Balkema-de Haan theorem [9, 27] (also called *second theorem* in EVT in comparison to the initial result of Fisher, Tippett and Gnedenko) given below.

Theorem 3.1 (Pickands-Balkema-de Haan). *The cumulative distribution function $F \in \mathcal{D}_\gamma$[1] if and only if a function σ exists, for all $x \in \mathbb{R}$ s.t. $1 + \gamma x > 0$:*

$$\frac{\bar{F}(t + \sigma(t)x)}{\bar{F}(t)} \underset{t \to \tau}{\longrightarrow} (1 + \gamma x)^{-\frac{1}{\gamma}}.$$

A clearer view of the theorem is the following:

$$\bar{F}_t(x) = \mathbb{P}(X - t > x \mid X > t) \underset{t \to \tau}{\sim} \left(1 + \frac{\gamma x}{\sigma(t)}\right)^{-\frac{1}{\gamma}}.$$

This result shows that the excess over a threshold t, written $X - t$, are likely to follow a Generalized Pareto Distribution (GPD) with parameters γ, σ. In fact, the GPD needs a third parameter, the location μ, but it is null in our case. Rather than fitting an EVD to the extreme values of X, the POT approach tries to fit a GPD to the excesses $X - t$.

In the case we get estimates $\hat{\gamma}$ and $\hat{\sigma}$ (our method will be described in 3.4), the quantile can be computed through :

$$z_q \simeq t + \frac{\hat{\sigma}}{\hat{\gamma}}\left(\left(\frac{qn}{N_t}\right)^{-\hat{\gamma}} - 1\right), \tag{1}$$

where t is a "high" threshold (details will be given in 4.3.3), q the desired probability, n the total number of observations, N_t the number of *peaks* i.e the number of X_i s.t. $X_i > t$.

Some classical methods can be used to perform the estimation of γ and σ, as the Method of Moments (MOM) or the Probability Weigted Moments (PWM) but they are less efficient and robust than the maximum likelihood estimation [10] that we describe below.

3.4 Maximum likelihood estimation

3.4.1 Likelihood expression. The maximum likelihood estimation remains a natural way to evaluate the parameters through observations. If $X_1, \dots X_n$ are n independent realizations of the random variable X which density (noted f_θ) is parametrized by θ (possibly a vector), the likelihood function is defined by:

$$\mathcal{L}(X_1, \dots X_n; \theta) = \prod_{i=1}^{n} f_\theta(X_i).$$

It represents joint density of these n observations. As $X_1, \dots X_n$ are fixed in our context, we try to find the parameter θ such that the likelihood is maximized. It means that we are looking for the value of θ which makes our observations the most probable. Practically, we work on the log-likelihood, so in our case (GPD fit) we have to maximize :

$$\log \mathcal{L}(\gamma, \sigma) = -N_t \log \sigma - \left(1 + \frac{1}{\gamma}\right) \sum_{i=1}^{N_t} \log\left(1 + \frac{\gamma}{\sigma} Y_i\right),$$

[1] It means that the extrema of the distribution of F converge in distribution to G_γ.

where $Y_i > 0$ are the excesses of X_i over t ($Y_i = X_i - t$ for $X_i > t$). Unfortunately, the optimization must be done numerically, implying the classical numerical issues. To perform it, the procedure of Grimshaw [21] can be used.

In a strict GPD case (if the Y_i follow exactly a GPD), the Maximum Likelihood Estimate (MLE) has some good convergence properties in comparison to other estimates (MOM or PWM). It converges in distribution to a Gaussian distribution when the number of peaks $N_t \to \infty$ when $\gamma > -\frac{1}{2}$ (with a rate of consistency $\sqrt{N_t}$) and is superefficient when $-1 < \gamma < -\frac{1}{2}$ with a rate of consistency $N_t^{-\gamma}$.

3.4.2 The Grimshaw's trick. The trick of the Grimshaw's procedure is to reduce the two variables optimization problem to a single variable equation. Let us write $\ell(\gamma, \sigma) = \log \mathcal{L}(\gamma, \sigma)$. As we find an extremum of ℓ, we look for solutions of the system $\nabla \ell(\gamma, \sigma) = 0$. Grimshaw has shown that if we get a solution (γ^*, σ^*) of this system then the variable $x^* = \gamma^*/\sigma^*$ is solution of the scalar equation $u(x)v(x) = 1$ where:

$$u(x) = \frac{1}{N_t} \sum_{i=1}^{N_t} \frac{1}{1 + xY_i} \qquad v(x) = 1 + \frac{1}{N_t} \sum_{i=1}^{N_t} \log(1 + xY_i)\,.$$

Moreover, by finding a solution x^* of this equation, we can retrieve $\gamma^* = v(x^*) - 1$ and $\sigma^* = \gamma^*/x^*$. Nevertheless, the solutions of this equation give only possible candidates for the maximum of ℓ, so we have to get all the roots, to calculate the corresponding likelihood and keep the best tuple $(\hat{\gamma}, \hat{\sigma})$ as our final estimates.

We have to pay attention to how this numerical root search is done. In fact, the values $1 + xY_i$ must be strictly positives. As the Y_i are positive, we must find x^* on $\left(-\frac{1}{\mathbf{Y}^M}, +\infty\right)$ where $\mathbf{Y}^M = \max Y_i$. Grimshaw calculates also an upper-bound $x^*_{\max}$ for this root search:

$$x^*_{\max} = 2\frac{\overline{\mathbf{Y}} - \mathbf{Y}^m}{(\mathbf{Y}^m)^2},$$

where $\mathbf{Y}^m = \min Y_i$ and $\overline{\mathbf{Y}}$ is the mean of the Y_i. Finally, the number of roots is not known and 0 is always a solution so the implementation must find all the solutions and pick up those which maximizes the likelihood.

4 OUR CONTRIBUTION

The extreme value theory, through the POT approach, gives us a way to estimate z_q such that $\mathbb{P}(X > z_q) < q$ without any strong assumption on the distribution of X and without any clear knowledge about its distribution.

In this section we use this result to build a streaming outlier detector. First we present the initialization step which computes an threshold z_q from n observations $X_1, \ldots X_n$. Then, we detail our two streaming algorithms which update z_q with the incoming data and use it as a decision bound. We propose SPOT which works in stationary cases, and DSPOT which takes into account a drift component. Finally we give some theoretical and technical improvements making our bound update fast and sturdy.

4.1 Initialization step

Let us sum up the basic idea of our algorithm. We have n observations $X_1, \ldots X_n$, and we have fixed a risk q. The goal is to compute a first threshold z_q verifying $\mathbb{P}(X > z_q) < q$. The figure 3 shows what we do on this initial batch (calibration). The idea is to set a high threshold t (e.g. a high empirical quantile practically), retrieve the *peaks* (the excesses over t) and fit a GPD (Generalized Pareto Distribution) to them. So that we infer the distribution of the extreme values and we can compute the threshold z_q.

This initialization step is summarized in the algorithm 1. The choice of t will be discussed in 4.3.3. The set $\mathbf{Y}_t$ is the *peaks set* where we store the observed excesses over t. The GPD fit is performed with the Grimshaw trick (we detail our likelihood optimization in 4.3.2) and then we can compute z_q with equation 1.

Algorithm 1 POT (Peaks-over-Threshold)

```
procedure POT(X_1, ... X_n, q)
    t ← SetInitialThreshold(X_1, ... X_n)
    Y_t ← {X_i − t | X_i > t}
    γ̂, σ̂ ← Grimshaw(Y_t)
    z_q ← CalcThreshold(q, γ̂, σ̂, n, N_t, t)
    return z_q, t
end procedure
```

4.2 Finding anomalies in a stream

The POT primitive returns a threshold z_q which we use to define a "normality bound" (figure 3).

In our streaming algorithms the POT primitive (algorithm 1) is used as an initialization step.

The POT primitive may be seen as a *training* step but this is partly wrong because the initial batch $X_1, \ldots X_n$ is not labeled and is not considered as a ground truth in our algorithm. The initialization is more a *calibration* step. Our streaming anomaly detector uses the next observations to both detect anomalies and refine the anomaly threshold z_q.

4.2.1 Stationary case. The way how the POT estimate is built is really stream-ready. As we do not have to store the whole time series (only the peaks), it requires low memory so we can use it in a stream. However, the stream must contain values from the same distribution, so this distribution cannot been time-dependent (what we call *stationary*). In case of time-dependency, we will show that our algorithm can be adapted to drifting cases (see 4.2.2).

The principle of the SPOT algorithm is the following : we want to detect abnormal events in a stream $(X_i)_{i>0}$ in a blind way (without knowledge about the distribution). Firstly, we perform a POT estimate on the n first values ($n \sim 1000$) and we get an initial threshold z_q (initialization). Then for all the next observed values we can flag the events or update the threshold (see figure 3). If a value exceeds our threshold z_q then we consider it as abnormal (we can retrieve this anomaly in a list $\mathbf{A}$). The anomalies are not taken into account for the model update. In the other cases, either X_i is greater than the initial threshold (peak case) either it is a "common" value (normal case). In the peak case, we add the excess to the peaks set and we update the threshold z_q.

In this algorithm we perform the maximum number of threshold updates but it is possible to do it off-line at fixed time interval. Of course we illustrate the principle only with upper-bound thresholds

Algorithm 2 SPOT (Streaming POT)

1: **procedure** SPOT($(X_i)_{i>0}, n, q$)
2: $\mathbf{A} \leftarrow \emptyset$ ▷ set of the anomalies
3: $z_q, t \leftarrow \text{POT}(X_1, \dots X_n, q)$
4: $k \leftarrow n$
5: **for** $i > n$ **do**
6: **if** $X_i > z_q$ **then** ▷ anomaly case
7: Add (i, X_i) in $\mathbf{A}$
8: **else if** $X_i > t$ **then** ▷ real peak case
9: $Y_i \leftarrow X_i - t$
10: Add Y_i in $\mathbf{Y}_t$
11: $N_t \leftarrow N_t + 1$
12: $k \leftarrow k + 1$
13: $\hat{\gamma}, \hat{\sigma} \leftarrow \text{Grimshaw}(\mathbf{Y}_t)$
14: $z_q \leftarrow \text{CalcThreshold}(q, \hat{\gamma}, \hat{\sigma}, k, N_t, t)$
15: **else** ▷ normal case
16: $k \leftarrow k + 1$
17: **end if**
18: **end for**
19: **end procedure**

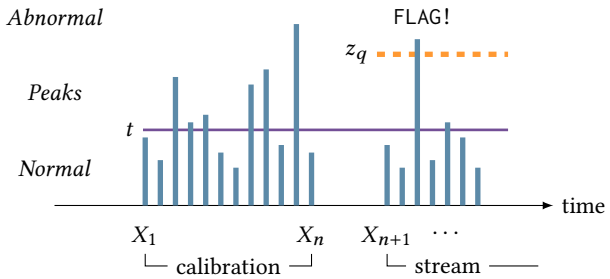

Figure 3: Anomaly detection overview

but the method is the same for lower-bound ones and we can even combine both (performances will be presented in 5.4).

4.2.2 Drifting case. SPOT assumes that the distribution of the X_i does not change over time but it might be restrictive. For instance, a mid-term seasonality cannot be taken into account, making local peaks undetectable. In this section we overcome this issue by modeling an average local behavior and applying SPOT on relative gaps.

We propose Drift SPOT (DSPOT) which makes SPOT run not on the absolute values X_i but on the relative ones. We use the variable change $X'_i = X_i - M_i$ where M_i models the local behavior at time i (see figure 4). In our implementation we used a moving average $M_i = (1/d) \cdot \sum_{k=1}^{d} X^*_{i-k}$ with $X^*_{i-1}, \dots X^*_{i-d}$ the last d "normal" observations (so d is a window parameter). In this new context we assume that the local variations X'_i come from a same stationary distribution (the hypothesis assumed for X_i in SPOT is now assumed for X'_i).

This variant uses an additional parameter d, which can be viewed as a window size. The distinctive features of this window (noted W^*) are the following: it might be non continuous and it does not contain abnormal values.

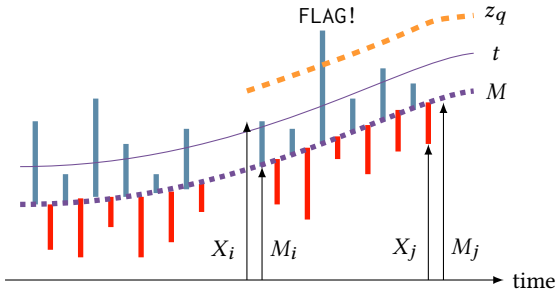

Figure 4: Anomaly detection with drift

Algorithm 3 DSPOT (Streaming POT with drift)

1: **procedure** DSPOT($(X_i)_{i>0}, n, d, q$)
2: $\mathbf{A} \leftarrow \emptyset$ ▷ set of the anomalies
3: $W^* \leftarrow [X_1, \dots X_d]$ ▷ Last d normal values
4: $M_{d+1} = \overline{W^*}$ ▷ Local model with depth d
5: **for** $i \in [\![d+1, d+n]\!]$ **do**
6: $X'_i = X_i - M_i$
7: $W^* \leftarrow [X_{i-d+1}, \dots X_i]$
8: $M_{i+1} \leftarrow \overline{W^*}$
9: **end for**
10: $z_q, t \leftarrow \text{POT}(X'_{d+1}, \dots X'_{d+n}, q)$
11: $k \leftarrow n$
12: **for** $i > d + n$ **do**
13: $X'_i = X_i - M_i$ ▷ variable change
14: **if** $X'_i > z_q$ **then** ▷ anomaly case
15: Add (i, X_i) in $\mathbf{A}$
16: $M_{i+1} \leftarrow M_i$ ▷ no update
17: **else if** $X'_i > t$ **then** ▷ real peak case
18: $Y_i \leftarrow X'_i - t$
19: Add Y_i in $\mathbf{Y}_t$
20: $N_t \leftarrow N_t + 1$
21: $k \leftarrow k + 1$
22: $\hat{\gamma}, \hat{\sigma} \leftarrow \text{Grimshaw}(\mathbf{Y}_t)$
23: $z_q \leftarrow \text{CalcThreshold}(q, \hat{\gamma}, \hat{\sigma}, k, N_t, t)$
24: $W^* \leftarrow W^*[1:] \cup X_i$ ▷ window slide
25: $M_{i+1} \leftarrow \overline{W^*}$ ▷ update of the local model
26: **else** ▷ normal case
27: $k \leftarrow k + 1$
28: $W^* \leftarrow W^*[1:] \cup X_i$ ▷ window slide
29: $M_{i+1} \leftarrow \overline{W^*}$ ▷ update of the local model
30: **end if**
31: **end for**
32: **end procedure**

The algorithm 3 shows our method to capture the local model and perform SPOT thresholding on local variations. It contains some additional steps so as to compute variable changes. For these stages, we principally use a sliding windows over normal observations W^* (lines 3, 7, 24 and 28) in order to calculate a local normal behavior M_i (lines 4, 8, 25 and 29) through averaging. We logically update the local behavior only in normal or peak cases (lines 25 and 29).

We can retrieve sequentially the "real" extreme quantiles by adding M_i to the calculated z_q. Such a choice to model the local

behavior is a very efficient way to adapt SPOT to drifting contexts. As mentioned in the previous paragraph, DSPOT can be adapted to compute upper and lower bounds.

4.3 Numerical optimization

The streaming context requests fast and resilient algorithms. In this part, we optimize the GPD fit by reducing the search of optimal parameters and making it more robust to common numerical stability problems (divergence, absurd values). The proposition 4.1 gives a general result for EVT, improving the Grimshaw's trick. In 4.3.2, we detail how we perform the likelihood optimization and finally we give some details about the initial threshold t.

4.3.1 Reduction of the optimal parameters search. As we have seen in section 3.4.2, the Grimshaw's method for the maximum likelihood estimation requires a numerical root search in a bounded interval. In this paragraph we show that we can reduce this interval.

Gathering the following result and the previous bounds (section 3.4.2), the possible solutions of $u(x)v(x) = 1$ stand in the two intervals

$$\left(-\frac{1}{\mathbf{Y}^M}, 0\right] \text{ and } \left[2\frac{\overline{\mathbf{Y}} - \mathbf{Y}^m}{\overline{\mathbf{Y}}\mathbf{Y}^m}, 2\frac{\overline{\mathbf{Y}} - \mathbf{Y}^m}{(\mathbf{Y}^m)^2}\right].$$

Proposition 4.1. *If x^* is a solution of $u(x)v(x) = 1$,*

$$x^* \leq 0 \quad or \quad x^* \geq 2\frac{\overline{\mathbf{Y}} - \mathbf{Y}^m}{\overline{\mathbf{Y}}\mathbf{Y}^m}.$$

Proof. Since $\forall x > -1, \log(1+x) \geq \frac{2x}{2+x} = 2 - \frac{4}{2+x}$,

$$v(x) \geq 1 + \frac{1}{N_t}\sum_{i=1}^{N_t}\left(2 - \frac{4}{2 + xY_i}\right) \geq 3 - \frac{4}{2 + x\mathbf{Y}^m}.$$

Then applying Jensen's inequality on the convex function $x \mapsto \frac{1}{1+x}$ we get :

$$u(x) \geq \frac{1}{1 + x\overline{\mathbf{Y}}},$$

so that

$$u(x)v(x) \geq \left(3 - \frac{4}{2 + x\mathbf{Y}^m}\right)\left(\frac{1}{1 + x\overline{\mathbf{Y}}}\right).$$

If x^* is a solution of the equation $u(x)v(x) = 1$, we must have

$$1 \geq \left(3 - \frac{4}{2 + x^*\mathbf{Y}^m}\right)\left(\frac{1}{1 + x^*\overline{\mathbf{Y}}}\right).$$

Simplifying this inequality, we get

$$x^*\left(x^*\mathbf{Y}^m\overline{\mathbf{Y}} - 2\left(\overline{\mathbf{Y}} - \mathbf{Y}^m\right)\right) \geq 0.$$

And a simple sign study gives the result. □

4.3.2 How can we maximize the likelihood function? Finding the maximum of the likelihood boils down to apply a root search. But this is not a trivial task: we do not know the number of roots and all the roots are potential candidates to maximize this function.

In [21], Grimshaw gives an analytic-based routine using root-finding algorithm is given however the needed condition $f(a)f(b) < 0$ is not emphasized leading to uncertain results. For this reason we decide to use another method to find these roots.

In our implementation, we set a very small $\epsilon > 0$ ($\sim 10^{-8}$) and we look for the roots of the function $w : x \mapsto u(x)v(x) - 1$ in both intervals

$$\left[-\frac{1}{\mathbf{Y}^M} + \epsilon, -\epsilon\right] \text{ and } \left[2\frac{\overline{\mathbf{Y}} - \mathbf{Y}^m}{\overline{\mathbf{Y}}\mathbf{Y}^m}, 2\frac{\overline{\mathbf{Y}} - \mathbf{Y}^m}{(\mathbf{Y}^m)^2}\right].$$

The real ϵ is used to avoid both cases $x = \frac{1}{\mathbf{Y}^M}$ (where w is not defined) and $x = 0$ (which is always a solution).

Many methods exist to find multiple roots of polynomials (such as Sturm method) but not for the general case of scalar functions. Furthermore, finding a root needs a sign change which may be difficult to detect. Thus we have reduced our root search to a function minimization which requires less assumptions.

To find the zeros of w we solve numerically the following optimization problem in both intervals (that we note I):

$$\min_{x_1,..x_k \in I} \sum_{i=1}^{k} w(x_k)^2.$$

The minimization can be done with a classical algorithm (e.g. L-BFGS-B [12]) starting with k points $x_1^0, ...x_k^0$ ($k \simeq 10$) distributed over I. We use this procedure for three reasons: the optimal configuration $(x_1^*, ..x_k^*)$ is likely to contain the zeros of w in I, we can retrieve several roots (according to k) and optimizing procedures do not require sign change between the bounds.

We perform this optimization in both intervals so we get a list of candidates to maximize the likelihood (the case $x = 0$ is also treated). We keep the best of them and we retrieve the best parameters for the GPD fit.

4.3.3 Initial threshold. We have detailed the Grimshaw procedure (3.4.2 and 4.3) and how the final threshold z_q is calculated (equation 1) but we have not dealt with the initial threshold t. In practice, its value is not paramount except that it must be "high" enough. The higher is t, the more relevant will be the GPD fit (low bias). However, if t is too high, the peaks set $\mathbf{Y}_t$ would be little filled in, so the model would be more variable (high variance). The only important condition is to ensure that t is lower than z_q, meaning that the probability associated to t must be lower than $1 - q$. In practice we set t to a high empirical quantile (98%).

A method based on the *mean excess plot* [10] could be used to set t in a smarter way but it is less stable and likely to output absurd values.

5 EXPERIMENTS

In this section we apply both our algorithms SPOT and DSPOT on several contexts. First, we compare the computed threshold z_q with the theoretical ones through experiments on synthetic data. Then we use real world datasets from several fields (network, physics, finance) to highlight the properties of our algorithms.

Finally we present the performance of our implementation.

The real world datasets used in these experiments are all available on the Internet and we do our utmost to detail experimental protocols making them totally reproducible. Our `python3` implementation is available at [1].

5.1 (D)SPOT reliability

In this section, we compare our computed threshold z_q to the theoretical one. In other words, we want to check if z_q is the desired threshold (which verifies $\mathbb{P}(X > z_q) < q$). In the same time we evaluate the impact of the number of observations n in the initial batch.

In the following experiments we set $q = 10^{-3}$ and we run SPOT on Gaussian white noises of 15000 values, i.e. 15000 independent values from a standard normal distribution ($\mu = 0, \sigma^2 = 1$). For different values of n (300, 500, 1000, 2000 and 5000), we run SPOT $k = 100$ times and we retrieve the averaged error made in comparison to the theoretical threshold:

$$\text{error rate} = \left| \frac{z^{\text{SPOT}} - z^{\text{th}}}{z^{\text{th}}} \right|.$$

Here, z^{th} is the quantile of the standard normal distribution at level $1 - q$, so $z^{\text{th}} \simeq 3.09$. The figure 5 presents the results.

First of all the curves show that, for all initial batch sizes n, the error is low and decreases when the number of observations increases. It means that the computed threshold z^{SPOT} is close to the theoretical one and tends to it.

Secondly, we have to notice that the error curves all converge to the same value regardless of n. Therefore, n is not a paramount parameter. In our experiments, we just have to ensure that n is not too small, otherwise the initialization step is likely to fail because of a lack of peaks to perform the GPD fit. Generally, we use $n \simeq 1000$.

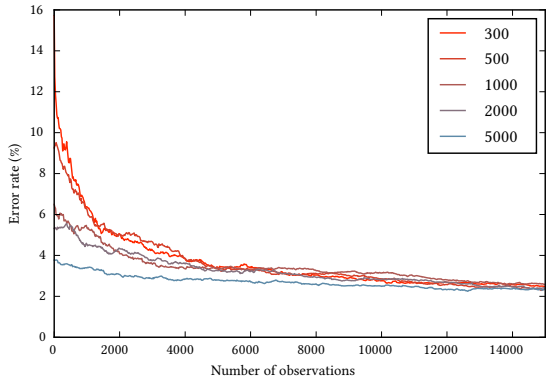

Figure 5: Error rate with the number of observations according to the batch size n

5.2 Finding anomalies with SPOT

5.2.1 Intrusion detection example. SPOT computes a robust threshold estimation (z_q): the more data we monitor, the more accurate the estimation is. So having a lot of data from the same and unknown distribution, SPOT can gradually adapt this threshold in order to detect anomalies. Cyber-security is a typical field where such configurations appear.

To test our algorithm we use real data from the MAWI repository which contains daily network captures (15 minutes a day stored in a `.pcap` file). In these captures, MAWIlab [19] finds anomalies and labels them with the taxonomy proposed by Mazel *et al.* [25]. The anomalies are referred through detailed patterns. To be close to real monitoring systems we converted raw `.pcap` files into NetFlow format, which aggregates packets and retrieves meta-data only, and is commonly used to measure network activity. Then we labeled the flows according to the patterns given by the MAWIlab. In this experiment we use the two captures from the 17/08/2012 and the 18/08/2012.

Classical attacks are network scans where many SYN packets are sent in order to find open and potentially vulnerable ports on several machines. A relevant feature to detect such attack is the ratio of SYN packets in a given time window [17]. From our NetFlow records we compute this feature on successive 50 ms time windows and we try to find extreme events. To initialize SPOT we use the last 1000 values of the 17/08 record and we let the algorithm working on the 18/08 capture.

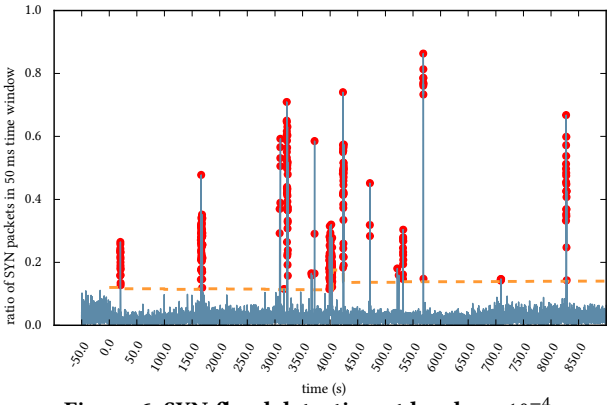

Figure 6: SYN flood detection at level $q = 10^{-4}$

The figure 6 shows the alerts triggered by SPOT (red circles). We recall that each point represents a 50 ms window gathering several flows (possibly benign and malicious). The computed threshold (dashed line) seems nearly constant but this behavior is due to the stability of the measure we monitor (SPOT has quickly inferred the behavior of the feature). By flagging all the flows in the triggered windows, we get a true positive rate equal to 86% with less than 4% of false positives.

5.2.2 The parameter q as a false-positive regulator. In the previous section we noticed that the size of the initial batch n is not an important parameter insofar as it does not affect the overall behavior of SPOT. Here, we study the impact of the main parameter q on the MAWI dataset.

On the figure 7, the ROC curve shows the effect of q on the False Positive rate (FPr). Values of q between 10^{-3} and 10^{-5} allow to have a high TPr while keeping a low FPr: this leaves some room for error when setting q.

5.3 Finding anomalies with DSPOT

5.3.1 Measure of the magnetic field. To show the wide variety of fields on which we can use DSPOT, we apply our algorithm on astrophysics measures from the SPIDR (Space Physics Interactive Data Resource) [4]. SPIDR is an online platform which stores and

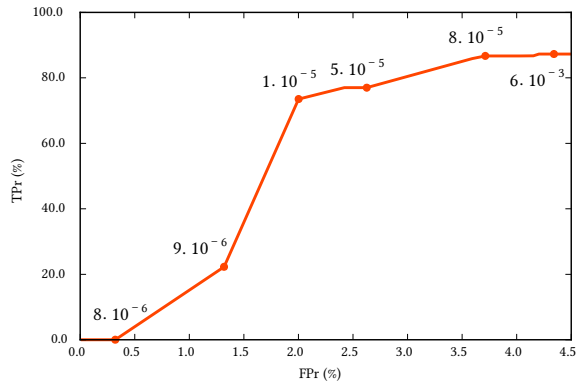

Figure 7: ROC curves on MAWI dataset (the markers give the corresponding value of q)

manages historical space physics data for integration with environment models and space weather forecasts.

Particularly we use a dataset available on Comp-Engine Time Series [3] which gathers some physical measures taken by the ACE satellite between 1/1/1995 and 1/6/1995. In the figure 8 we monitor a component of the magnetic field (in nT) during several minutes.

This time series is very noisy and contains different complex behaviors. We calibrate DSPOT with the $n = 2000$ first values and we run on the 15801 others with $q = 10^{-3}$ and $d = 450$. The results are depicted on the figure 8.

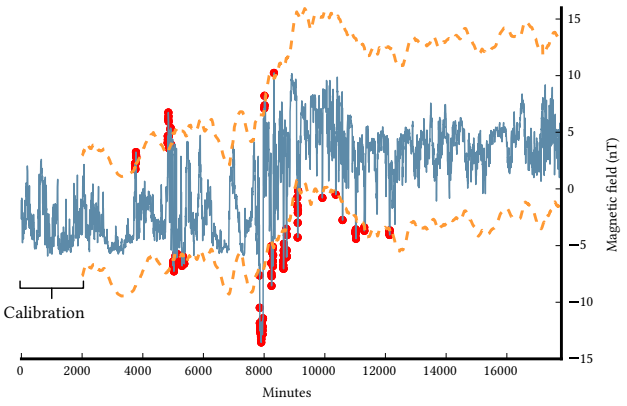

Figure 8: DSPOT run with $q = 10^{-3}, d = 450$

At the first glance, the bounds are following the signal and some alarms are triggered by high peaks. After 9000 minutes, the upper bound seems higher than expected.

To understand why, let us divide the signal into two parts: before and after 8000 minutes. Before 8000, we can observe that “peaks" lean upwards the trend although the opposite phenomenon appears after 8000. During these 8000 first minutes, DSPOT learns that peaks may lean upwards the trend and it keeps this information in memory. Hence after 8000 minutes, the upper bound stays high in order to accommodate for possible peaks above the trend. DSPOT has this behavior because it keeps a global memory of all peaks encountered. If it was not desired, un easy modification is to keep only the last k peaks, for a fixed k.

5.3.2 Stock prices. On Thursday the 9th of February 2017, an explosion happened at Flamanville nuclear plant, in northern France. This power plant is managed by EDF, a French electricity provider. The incident was not in the nuclear zone and did not hurt people [5]. This incident was officially declared at 11:00 a.m. making the EDF stock price fall down. This recent event encouraged us to test DSPOT on EDF stock prices.

Obviously, retrieving financial data with high resolution is not within ours grasp. However, some websites like Google Finance [2] propose intraday financial data with a record per minute. Google Finance keeps these records during 15 days, so we retrieve the records from the 6th to the 8th of February 2017 for calibration (1062 values) and ran DSPOT on the explosion day (379 values). .

On figure 9, we notice that DSPOT follows the average behavior and flags the drop around 11:00 a.m. This may help a trading system to quickly take actions or warn experts.

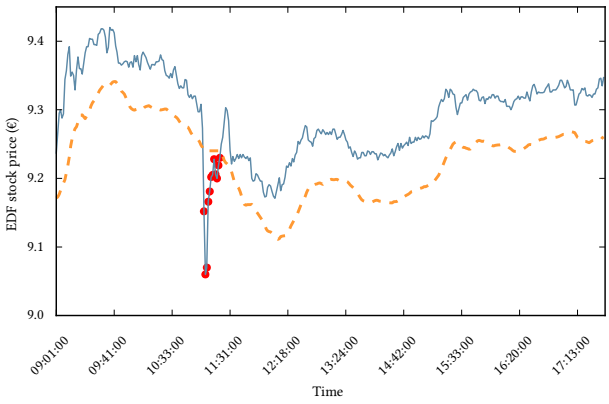

Figure 9: DSPOT run with $q = 10^{-3}, d = 10$

5.4 Performances

Here we give some details about the time and memory performances of our `python3` implementation. All these experiments have been made on a laptop with an Intel i5-5300U CPU @ 2.30GHz (4 cores) and 8 GB RAM.

Both algorithms, SPOT and DSPOT require a fixed memory size for all the variables except for the peak set $\mathbf{Y}_t$ which may grow with the number of observations. To measure the memory performance of our algorithm we report the number of peaks we stored.

To test the performances of our algorithms we run each of them on 100 Gaussian white noises of 15000 values (like the experiment in section 5.1). At every run we measure the averaged time to perform one iteration and the ratio of stored peaks, i.e. the number of peaks over the number of observations.

For these experiments we set $q = 10^{-3}$ and we use $n = 1000$ values for the initial batch. We record these measures in four contexts: SPOT, SPOT both sides (bi-SPOT), DSPOT and DSPOT both sides (bi-DSPOT). The “both sides" runs take into account upper and lower thresholds updates. In drifting cases, we add a drift to the Gaussian white noise and we use a depth $d = 50$.

The table 2 presents the averaged time to perform one iteration (denoted T, measured in μs) and the ratio number of peaks over number of observations at the end of the run (denoted M, in %) in mean, best and worst cases.

Method	Worst		Mean		Best	
	T	M	T	M	T	M
SPOT	959	2,70	351	1,90	141	1,40
DSPOT	883	3,49	391	2,02	197	1,07
bi-SPOT	1733	5,58	772	4,18	373	2,79
bi-DSPOT	1053	5,79	591	4,02	272	2,74

Table 2: Time (T, in μs) and Memory (M, in %) performances

Our algorithm stores a little ratio of all the stream (few percents). Logically we store about twice more when we compute upper and lower thresholds. Even if the growth speed of the peaks sets size is low it could be an hindrance for a long term monitoring. However, the size of the peaks set could be upper-bounded (with a high bound) making it work like a wide sliding window (very old peaks would be dropped) without loss of accuracy.

Finally the time performances of our algorithms show that our implementation is able to work on streams with more than 1000 values a second.

6 CONCLUSION

This paper has presented a novel approach to detect outliers in high throughput numerical time series. The key points of our approach is that it does not assume the data distribution, and it does not require manually set thresholds. It adapts on multiple and complex contexts, learning how the interest measure behaves. It achieves these results by using the Extreme Values Theory (EVT). To the best of our knowledge, it is the first time EVT has been used to detect outliers in streaming data.

There are two kinds of perspectives. From the theoretical side, in this paper we only exploited a small part of EVT, we would like to extend this work to the multivariate case and to non-iid observations.

From the practical side, one of the immediate application of our approach is *automatic thresholding*: it can provide thresholds with strong statistical guarantees that can adapt to the evolution of a data stream. We would like to explore the use of our approach as a building block into more complex systems that require thresholding. Also the EVT on multivariate cases could address our problem in more general contexts without independence condition.

REFERENCES

[1] https://github.com/Amossys-team/SPOT. (????).
[2] Google Finance. https://www.google.com/finance. (????).
[3] Magnetic field time series. http://www.comp-engine.org/timeseries/time-series_data/data-17481/. (????).
[4] Space Physics Interactive Data Resource. http://spidr.ionosonde.net/spidr/home.do. (????).
[5] http://www.webcitation.org/6oAxqoFkf. (????).
[6] Deepak Agarwal. 2005. An empirical bayes approach to detect anomalies in dynamic multidimensional arrays. In *ICDM*.
[7] Fabrizio Angiulli, Stefano Basta, and Clara Pizzuti. 2006. Distance-based detection and prediction of outliers. *IEEE transactions on knowledge and data engineering* (2006).
[8] Fabrizio Angiulli and Fabio Fassetti. 2007. Detecting distance-based outliers in streams of data. In *Proceedings of the 16th ACM conference on Conference on information and knowledge management*.
[9] August A Balkema and Laurens De Haan. 1974. Residual life time at great age. *The Annals of probability* (1974).
[10] Jan Beirlant, Yuri Goegebeur, Johan Segers, and Jozef Teugels. 2006. *Statistics of extremes: theory and applications*. John Wiley & Sons.
[11] Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. 2000. LOF: identifying density-based local outliers. In *ACM sigmod record*, Vol. 29. ACM, 93–104.
[12] Richard H Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. 1995. A limited memory algorithm for bound constrained optimization. *SIAM J. on Scientific Computing* (1995).
[13] Varun Chandola, Arindam Banerjee, and Vipin Kumar. 2009. Anomaly detection: A survey. *ACM computing surveys* (2009).
[14] Lian Duan, Lida Xu, Ying Liu, and Jun Lee. 2009. Cluster-based outlier detection. *Annals of Operations Research* 168, 1 (2009), 151–168.
[15] Manzoor Elahi, Kun Li, Wasif Nisar, Xinjie Lv, and Hongan Wang. 2008. Efficient clustering-based outlier detection algorithm for dynamic data stream. In *FSKD'08*.
[16] Eleazar Eskin. 2000. Anomaly detection over noisy data using learned probability distributions. In *ICML*.
[17] Guilherme Fernandes and Philippe Owezarski. 2009. Automated classification of network traffic anomalies. In *ICSPCS*.
[18] Ronald Aylmer Fisher and Leonard Henry Caleb Tippett. 1928. Limiting forms of the frequency distribution of the largest or smallest member of a sample. In *Mathematical Proceedings of the Cambridge Philosophical Society*.
[19] Romain Fontugne, Pierre Borgnat, Patrice Abry, and Kensuke Fukuda. 2010. MAWILab: Combining Diverse Anomaly Detectors for Automated Anomaly Labeling and Performance Benchmarking. In *ACM CoNEXT '10*.
[20] Boris Gnedenko. 1943. Sur la distribution limite du terme maximum d'une serie aleatoire. *Annals of mathematics* (1943), 423–453.
[21] Scott D. Grimshaw. 1993. Computing Maximum Likelihood Estimates for the Generalized Pareto Distribution. *Technometrics* 35, 2 (1993), 185–191. https://doi.org/10.1080/00401706.1993.10485040 arXiv:http://amstat.tandfonline.com/doi/pdf/10.1080/00401706.1993.10485040
[22] Bruce M Hill. 1975. A simple general approach to inference about the tail of a distribution. *The annals of statistics* 3, 5 (1975), 1163–1174.
[23] Ludmila I Kuncheva. 2008. Classifier ensembles for detecting concept change in streaming data: Overview and perspectives. In *2nd Workshop SUEMA*.
[24] Rikard Laxhammar and Göran Falkman. 2014. Online learning and sequential anomaly detection in trajectories. *IEEE transactions on pattern analysis and machine intelligence* 36, 6 (2014), 1158–1173.
[25] Johan Mazel, Romain Fontugne, and Kensuke Fukuda. 2014. A taxonomy of anomalies in backbone network traffic. In *IWCMC*.
[26] EWT Ngai, Yong Hu, YH Wong, Yijun Chen, and Xin Sun. 2011. The application of data mining techniques in financial fraud detection: A classification framework and an academic review of literature. *Decision Support Systems* 50, 3 (2011), 559–569.
[27] James Pickands III. 1975. Statistical inference using extreme order statistics. *the Annals of Statistics* (1975).
[28] Md Shiblee Sadik and Le Gruenwald. 2010. DBOD-DS: Distance based outlier detection for data streams. In *International Conference on Database and Expert Systems Applications*. Springer, 122–136.
[29] Shiblee Sadik and Le Gruenwald. 2014. Research issues in outlier detection for data streams. *ACM SIGKDD Explorations Newsletter* 15, 1 (2014), 33–40.
[30] John E Seem. 2007. Using intelligent data analysis to detect abnormal energy consumption in buildings. *Energy and buildings* 39, 1 (2007), 52–58.
[31] Durga Toshniwal. 2012. A framework for outlier detection in evolving data streams by weighting attributes in clustering. *Procedia Technology* 6 (2012), 214–222.
[32] Haining Wang, Danlu Zhang, and Kang G Shin. 2002. Detecting SYN flooding attacks. In *INFOCOM*.