A Fine-Grained Threshold-Setting Approach for Anomaly Detection in Data Streams

James P. Clark Nathalie Japkowicz
Department of Computer Science
American University
Washington, DC
{jc7553a@student., japkowic@}american.edu

Abstract

As the distribution of a data stream evolves over time, a learner must adapt to these distributional shifts to make accurate predictions. In particular, it is crucial for the learner to distinguish between a natural change in distribution and an anomaly in the data stream. In the literature, anomaly detection is often addressed using a one-class classifier. In such systems, the learning process is divided into two parts: concept learning and threshold determination. The purpose of this paper is to study the issue of threshold determination when shifts in data streams occur. Previous works handled this issue using sliding windows that attempt to combine learning and forgetting mechanisms for the data. This paper proposes a novel solution that uses the concept of a sliding window and combines it with the idea of change point detection. The method is beneficial in two ways: from the machine learning perspective, it allows us to vary the threshold of a one-class learning system in an adaptive and informed way; from a statistical perspective, it expands the reach of univariate change point detection methods by enabling them to work on high-dimensional data.

Introduction

The task of process monitoring or event detection in data streams is becoming increasingly important in modern applications. For example, in the medical realm, continuously monitoring a patient's vital signs and alerting an emergency center in case a deterioration of the patients condition is detected can save lives. In the cybersecurity domain as well, a network can be continuously monitored and an alert issued in the event of a cyber-attack. This too could prevent disastrous situations to develop.

Recently, process monitoring in continuously incoming data streams has been proposed using an extension of a classical one-class learning system (Dong and Japkowicz 2016). In that work, as well as in other one-class learning work applied to batch or streaming data (e.g. (Manevitz and Yousef 2002), (Japkowicz, Myers, and Gluck 1995), etc.), the learning process is divided into two parts: concept learning and threshold-setting. More often than not, the concept learning part has been given a lot of attention while the threshold setting part was considered trivial and uninteresting. Indeed, it has usually been assumed that a threshold can be set by the user through inspection of the inductive learning algorithms results, or determined automatically by a simple statistical procedure. This is not so, however, and as a result, the sophisticated inductive learning part of the system can be compromised by a naive threshold setting system that may negate the gains made by the inductive concept-learning system. This issue has become even more pressing in the context of data stream learning since, in non-stationary environments, the threshold may change over-time making it undesirable for a user to set it once at the beginning of the process deployment and let it run indefinitely.

The focus of this paper is on the task of threshold setting in a data stream monitoring context. One of the challenges of threshold setting is determining whether an anomalous event detected by the system represents a true anomaly (e.g., an event worth reporting) or a negligible anomaly. In the case of data streams, where data arrives continuously, an anomalous event could represent a concept drift (e.g., (Gama et al. 2014),(Webb et al. 2016)) rather than a true threat or emergency. We, thus propose a novel approach for dealing with the problem that applies a statistical change point detection method to the output of a one-class learning approach. The idea is quite simple and consists of letting the window size from which the threshold is calculated increase until a change point is encountered. Once that happens, the window is shrunk back to its initial size, the threshold re-calculated and the process re-started as before.

The approach is compared to one that simply applies a fixed window size on the output of the one-class learning system without using an intermediate change point detection method. Our results on three artificial domains and two real-world domains, using three different one class learning systems and two different change point detection algorithms show that adding change point detection as an intermediate step between one-class learning and threshold determination is beneficial.

Related Works

We now review current methods for anomaly detection in data streams, discuss the issue of concept drifts, and review the different mechanisms for handling concept drifts that are in use today.

Anomaly Detection in Data Streams

Anomaly Detection in data streams is a difficult problem which presents three main challenges (Tan, Ting, and Liu 2011) first, the data stream is continuous so the data needs to be continuously processed rather than stored; second, the data stream contains mostly "normal" data with very few instances of abnormal ones; and third, the data stream is likely to encounter concept drifts overtime. The second challenge makes standard supervised data stream learning algorithms (e.g., (Domingos and Hulten 2000), (Hulten and Domingos 2001), (Rutkowsk et al. 2014)) inapplicable given the dearth of anomalies which turns the problem into an extreme class imbalance problem. As discussed in (Bellinger et al. 2017), when such extremes are encountered, it is sensible to use one-class learning systems. A few researchers have proposed fast one-class learning systems to address this challenge (e.g., (Tan, Ting, and Liu 2011)), but these algorithms are based on decision trees which are unlikely to adapt well to changes, and thus do not address the third challenge. More recently, (Dong and Japkowicz 2016) proposed a neural network approach based on (Japkowicz, Myers, and Gluck 1995) which has the advantage of addressing all three challenges. This is the method that will be considered in this paper, along with two other approaches—One-class SVM and One-Class k-Nearest-Neighbors—which, although not yet adapted to dealing with the first challenge, are able to handle the other two adequately.

Concept Drift

A concept drift, as previously discussed, occurs when the conditional distribution of the target value changes while the distribution of the input values remains unchanged. To put this into a learning context, we can denote $P(Y)$ as the probability distribution over class labels and $P(X)$ as the prior probability distribution over covariates. Together we can say that $P(Y, X)$ is the joint probability distribution over objects and class labels.

According to (Webb et al. 2016)'s definition, a concept must be defined before characterizing a concept drift. Recently a concept has been given a probabilistic definition, defined by (Gama et al. 2014), as the prior class probabilities $P(Y)$ and class conditional probabilities $P(X|Y)$. As $P(Y)$ and $P(X|Y)$ determine the joint distribution $P(X, Y)$ and the other way around for the alternative, this is the same as defining a concept as the joint distribution $P(X, Y)$. To put this all together we can add a subscript noting time j as $P_j(X, Y)$ which states the probability at time j. We can then say that a concept drift occurs between times j and m if

$$P_j(X, Y) \neq P_m(X, Y)$$

Now that a concept drift has been defined, we can see, in Figure 1, several types of concept drifts. The first type is a sudden/abrupt change. This kind of situation can happen when a sensor is replaced by another sensor with a different calibration. In such a case, although the numbers are different, the instance should still be defined as normal. The second type of drift is called incremental. It describes a smooth change to the system which could model, for example, the rise in global temperatures over centuries. Most drifts take on these two characteristics, which are the only two we will consider in this work, and the challenge is not to mix a concept drift with an outlier or a random deviation. For a discussion of the other drifts illustrated in Figure 1, please refer to (Gama et al. 2014).

Figure 1

Adaptive Learning Strategies

To adapt to concept drifts there are different techniques in place as described by (Gama et al. 2014). The two main concepts for deploying these strategies are *memory* and *forgetting*. The memory strategy involves a single example and is better known as the family of online learning algorithms. These types of algorithms involve processing one example at a time and updating the system based on that example. These methods begin by taking the input (x) and making a prediction (y). Once the true value of y has been received, later on in the process, these algorithms are able to compute the loss and update the system based on that loss. This scheme, used by this family of approaches, allows them to adapt to concept drifts because they are continually updating the system with each instance. The downside of these approaches, however, is that they tend to be slow at adapting to abrupt changes in the system. The forgetting strategy involves multiple examples and bases itself on maintaining a model that uses a given set of recent examples. This requires a sliding window that uses a queue (first-in-first-out) data structure. This allows a system to forget older examples by continuously updating itself on more recent ones, by continuously rebuilding the model.

As previously discussed sliding windows are based off of multiple examples and can be deployed in two different ways: a fixed size or a variable size. The size of the window plays a crucial role in how the system adapts to changes, for example, a large sliding window performs very well in time of stability but causes the system to adapt very slowly to concept drifts. The opposite is true for smaller sizes where the system adapts quickly to a concept drift but performs poorly during a stable period. The problem considered in this work is how do we account for this and vary the size of the sliding window appropriately?

Proposed Approach

As just discussed, the purpose of our work is to improve the performance of threshold-setting in a data streaming one-class anomaly detection context by allowing variable window sizes for threshold calculation. There are two different approaches to modifying the threshold, namely blind adaptation and informed adaptation. Blind adaptation does not use any sort of concept-drift detection method. It typically uses a fixed window size and periodically retrains the system with the most recent instances.That is the method most

commonly used to handle concept drifts. Informed adaptation, on the other hand, looks for a trigger to alert the system of a change. When a trigger occurs, the system can vary the window size after it verifies that a concept drift is, indeed, taking place. that method has not, to date, been used and it is the one we propose in this work.

In more detail, our approach takes as input the time-series emanating from a one-class anomaly detection concept learner and begins by computing a threshold based on an initial default window size. In this particular study, the threshold corresponds to the sum of the mean and one stadard deviation from the data. More sophisticated threshold determination methods could be considered in the future. Once the initial threshold has been computed, the procedure continuously adds the next input from the time-series to the window, provided that it is not an actionnable change point, and re-calculates the threshold based on this addition. This allows the window to grow and results in a robust threshold in times of stability. If, however, the next point in the time-series has been flagged as a change-point and if this change point lies beneath the current threshold value, then the change point is assessed to determine whether it corresponds to a true anomaly or to a concept drift. This is done by analyzing the loss value at this change point. If it is larger than a given default value, then the change point is assessed to be a true anomaly. Otherwise, it is assessed to be a concept-drift. In cases when the change point is assessed to indicate a concept drift, then we consider it to be an actionnable change point and shrink the window back to its default size. As a result, the threshold will change more effectively due to our algorithm. The algorithm just discussed is described more formally in the pseudo-code shown Algorithm 1.

Algorithm 1 Variable Threshold

I$nput$: Losses are total losses created during testing phase of one class learner
Window Size is the size one would shrink their window down to
O$utput$ Anomaly = True or False

procedure VARIABLETHRESHOLD(Losses, WindowSize)
$Window = Losses[0 \rightarrow WindowSize]$
Threshold = CalculateThreshold(Window)
  for each $L \in Losses$ **do**
    if L = ChangePoint **then**
      if $Losses[L] \leq Threshold$ **then**
        $Window = Losses[L - WindowSize \rightarrow L]$
        $Threshold = CalculateThreshold(Window)$
      end if
    end if
    if Losses[L] $\leq$ Threshold **then**
      Losses[L] = "normal"
    else
      Losses[L] = "abnormal"
    end if
  $Window.append(Losses[L])$
  $Threshold = CalculateThreshold(Window)$
  end for
end procedure

Methodology

This section provides formal details about the different tools used in this work. We first introduce the concept of one-class learning and describe the three learning algorithms employed in this study: Autoencoders, One Class Support Vector Machines, and K Nearest Neighbor. Second, we introduce the concept of change point detection and describe the two techniques we used: PELT and Mann Whitney.

One-Class Learning

One-Class Learning was first proposed in (Japkowicz, Myers, and Gluck 1995) and (Moya and Hush 1996) and further refined in (Japkowicz 2001). The main idea is that instead of learning a concept by *discriminating* between positive and negative instances of that concept, learning only considers positive instances of that concept and learns how to *recognize* such instances (Japkowicz 2001). Various approaches were proposed for one-class learning. We now describe three of them.

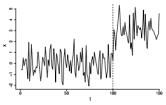
Autoencoder The design of an autoencoder is a feed-forward neural network. The network of choice was a three level architecture with m inputs, m outputs and k neurons where $k < m$. The network is trained, under the normal backpropogation algorithm to learn the identity function on the training samples.

We use the autoencoder as a one class learning system where it is trained on only normal instances. When the network reconstructs it′s input there will be a error which we will define as the "reconstruction error". From this value the network can distringuish normal instances from abnormal by a threshold value that splits the classes apart. The idea of the autoencoder is explained in thorough detail by (Japkowicz 2001).

One Class SVM Initially brought up by (Scholkopf et al. 2001) which created an adaption of the SVM method to one-class classification.As explained by (Manevitz and Yousef 2002) after transforming the features of the data set via a kernal, the origin is considered the only member of the second class. Then by using *"relaxation parameters"* one can seperate the features form the origin. After training has been done usual two-class SVM techniques may be used. More of this is thoroughly covered in (Manevitz and Yousef 2002).

K-Nearest Neighbor The Nearest Neighbor algorithm is a very simple algorithm and can be understood quite simply. For each training example in the training set we map each example into a space of R^d where d is the number of features. In the testing phase for each testing example we can compute the distance that point is from each of the training points (different distance measures can be used, but in this work, we used the Minkowski distance). We select the k smallest distances, and average them together. Like with the autoencoder, if this average is smaller than a given threshold, we conclude that the testing example belongs to the concept represented by the training examples; Otherwise, it is believed not to belong to that concept.

Figure 2: Mean Based Change Point



Change Point Detection

The statistical method that is vital to this paper is change point detection. To describe change point detection there must be a setting which contains a set of random variables labeledX_1, X_2, X_3, etc. that undergo abrupt changes to the distribution at change points C_1, C_2, etc. A simple example is shown in Figure 2 where a Gaussian distribution suffers a change in mean, and the change point is given by the dotted vertical line. Change point detection outputs the points in a time series at which changes occurred.

Change point detection can detect changes in two different settings, batch detection and sequential detection. In a batch detection setting, which is the focus of this paper, the input is a fixed size of n random variables labeled X_1, . , . ,X_n and the procedure tests to see if there exist multiple change points in the sequence. This allows the detection method to look at all variables in the given sequence to determine changes.

Parametric Change Point Detection—PELT In parametric change point detection algorithms, it is assumed that the underlying distribution of the data is normal. To further explain the parametric change point detection approach, consider a batch setting where there contains a sequence of n random variables, X_1,.,.,X_n which contains at least one change point. If that change point exists at a given time C_0 then the distribution leading up to point C_0 will be denoted now as F_0. After time C_0 there will exist another distribution calledF_1 where $F_0 \neq F_1$. The way the change point is determined is by performing a two-sample hypothesis test where the test statistic must be chosen by what the assumption of the underlying distribution is. An example of a test statistic could be a shift in the mean, the variance, or the mean variance.

Once a test statistic, D_k has been given the data will be partitioned into contiguous segments at each given point. From each segment the test statistic will be computed and standardized by subtracting the mean and dividing by the standard deviation of each value in the set $D_1., .D_k$ and taking the absolute value. It will be determined that there a change has taken place if point $D_n > h_n$ where h_n is a chosen threshold value. This value is calculated by bounding the Type 1 error rate. In other words the null hypothesis of no change is rejected, and the process starts all over again starting from point D_n.

Non-Parametric Change Point Detection—Mann-Whitney If the distrubution is unknown then ther exists a nonparametric approach to determining change points uses the Mann-Whitney two sample test. The Mann-Whitney U test is a nonparametric test of the null hypothesis that says it is equally likely that a random selected value from one sample will be less than or greater than a randomly selected value from a second sample. To explain let′s assume there exists a set of independent continuous random variables $X_1, X_2..X_k, ...X_n$, where the value k is a change point and the size of the set is n. There exists a U-statistic proposed by (Pettitt 1979) which is based on the Mann-Whitney two sided test. Let

$$D_{ij} = sgn(x_i - x_j)$$

Where sgn(x) = 1 if x$>$ 0, 0 if x = 0, -1 if x $<$ 0 then consider the equation

$$U_{i,t} = \sum_{i=1}^{t} \sum_{j=t+1}^{T} D_{ij}$$

This statistic called the U-Statistic is equivalent to the Mann-Whitney test explained previously. To be clear the U statistic is calculated for all values from 1 to n inclusive.

After having calculated all U statistics the next step in the process is to use another statistic to test out hypothesis of no change or change. Let,

$$K_t = maxabs(U_{i,t})$$

and then for changes in one direction we will need to involve the statistics.

$$K_t^+ = maxU_{i,t}$$
$$K_t^- = minU_{i,t}$$

To be clear the null hypothesis is that,$E(D_{i,j}) = 0$ and the distribution of $U_{i,t}$, is symmetric around zero, which means that K_t^+ and K_t^- both have the same null distribution, thus no change has occurred. If the distributions are different and K_t^+ is large than there has been a shift down in the series. Also if K_t^i is larger than there has been a shift upwards in the series.

Experimental Setup

In this section we will provide our experimental settings and which datasets were used.

Artifical Data Sets The artificial data takes on three different shapes given as shown in Figures 3-5. Each of these represents a shift in the underlying distribution over a given time scale in different ways. The first two represent gradual shifts of different degree, while the third one represents two abrupt shifts (where the second shift brings the distribution back to the original one. The artificial data was created specifically to test our variable sized window method against a fixed sized window. There were no learners involved in this process and the ultimate goal was to show that change point detection is a viable option under different graphical representations.

Real Data Sets The two datasets from UCI are namely, Shuttle Dataset and CoverType. Experiments on these will allow us to determine whether there is practical value in our method. In more detail, the shuttle data set contains statlog data from NASA that was generated over a time sequence. The set consists of 9 continuous numerical variables, although the first variable is time and the last is the classification. The covertype dataset was collected by the US Forest Service in the Roosevelt National Forest of northern Colorado. The goal is to predict the forest cover type from cartographic variables. The problem is defined by 54 variables of

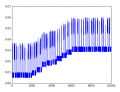


Figure 3

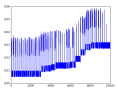


Figure 4

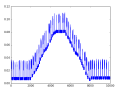


Figure 5

different types being continuous and categorical. The dataset contains 581,012 examples.

Experimental Settings

We compared the performance of using a variable sized sliding window based on the two different change point detection methods described in section ChangePoint Detection (Mann-Whitney, PELT), versus the performance of a fixed sized sliding window. The fixed size sliding window size was 300 for all datasets except for the third artificial set which used 50. A smaller default window size was used in the third case because we simulated an abrupt concept drift which would be badly detected with a larger window size. Generally speaking, the window size is a parameter that needs to be set ahead of time and which depends on the type of distribution observed. For the variable sized window we shrank the size back down to 300 after a concept drift was detected for all the data sets except for the third artificial set for which the default size was 50. For both algorithms the way to determine the threshold value was the same across all datasets by summing the mean and adding one standard deviation.

For the UCI datasets set we implemented the autoencoder using Python and notably Google′s tensor flow package. The activation function for the autoencoder was chosen to be the sigmoid function.The number of hidden units was half of the dimensionality of the input vector given. The learning rate for this experiment was .05. During the training phase we involved both mini-batch gradient descent, with 280,00 epochs, followed by stochastic gradient descent, with 4 epochs.

Both the OCSVM and Nearest Neighbord algorithm were implemented using Python′s scikit-learn package. For the OCSVM the sigmoid function was chosen because it yielded the best results. The kernel coefficient used was half of the dimensionality of the features. For the Nearest Neighbor distance measurement, Minkowski distance which is a distance metric in a normed vector space was chosen. The Minkowski distance is considered to be a generalization of the Euclidean and Manhattan distances. Last, the value of K was chosen to be 3 for all experiments.

The normal class (negative class) in both the shuttle data set and covertype was class 1 notably because it was the largest class and it spanned the entire time of the data set. For the shuttle data set it involved 31,108 training examples taken from the training set provided by UCI. The testing set involved 14,500 examples provided in the given testing set. For covertype the learners were trained on 1,600 instances that were given in the training set. They were then tested on 565,912 examples which is the entire testing set given.The testing methodology is the same on both, with training on the entire training set and testing on the entire testing set.

The change point detection algorithms used were provided by the R package ChangePoints and CPM. As explained in parametric change point detection, a parametric algorithm must be provided a statistical measurement and in this case, we decided to use the variance with the PELT algorithm. From the R package CPM the nonparametric algorithm Mann-Whitney was applied.

We decided to use a train and test method because other known methods such as cross-validation do not apply in the case of streaming data. The performance measurments chosen specificially for this study were precision and false positive rate.

Experimental Results

Result Listings

The results of the artificial data set follow. The graphical representations of each data set was given previously in Figures 3-5, Following the artificial sets comes the real data.

Data	Mann	PELT	Fixed
Artificial 1	94.5	94.5	80.76
Artificial 2	87.39	93.72	82.71
Artificial 3	80.77	83.21	52.00

Table 1: Precision Percentage on Artificial Data

Data	Mann	PELT	Fixed
Artificial 1	2.18	2.18	7.2
Artificial 2	4.03	2.10	5.50
Artificial 3	5.69	5.24	14.85

Table 2: False Positive Rate Percentage on Artificial Data

Data	Mann	PELT	Fixed	Learner
Shuttle	64.69	64.69	22.27	AE
Shuttle	37.86	37.86	22.47	OCSVM
Shuttle	58.97	30.74	22.63	K-NN
CoverType	32.28	32.59	29.99	AE
CoverType	25.88	23.68	25.95	OCSVM
CoverType	15.73	13.73	21.79	K-NN

Table 3: Precision Percentage on Real Data Sets

Data	Mann	PELT	Fixed	Learner
Shuttle	8.77	8.77	17.9	AE
Shuttle	15.64	15.64	19.81	OCSVM
Shuttle	10.57	16.86	18.68	K-NN
CoverType	55.51	55.23	56.32	AE
CoverType	58.93	59.24	58.98	OCSVM
CoverType	63.0	62.45	58.24	K-NN

Table 4: False Positive Rate on Real Data Sets

Discussion

Based on the tables, the results show that our change point method can outperform the fixed sliding window in all cases except, when using nearest neighbor on the cover type data set and slightly against OCSVM there as well. In more detail, the change point approach outperforms the fixed size approach only slightly in the first two artificial datasets, but it does so dramatically on the last artificial set. This is to be expected since in the fixed case, the change in threshold lags behind as the older data is not forgotten fast enough, whereas in the adaptive approach, since the window size is decreased as soon as the first change point is detected, more weight is given to the new data points more quickly. In the shuttle data set our change point method outperforms the fixed size sliding window on all performance measures. On the cover type data set, the results are all low, but this can be explained by the fact that the learners have an exceptionally hard time performing well on this data set both in this study and in previous ones (Gama, Rocha, and Medas 2003).

Altogether, the results show that during distribution changes to classes, the variable sliding window is a viable options when trying to handle a concept drift while still accounting for anomalies. In most cases, that method is shown to work better than the fixed size window approach currently used in most studies. Furthermore, we can see that for almost every dataset the nonparametric approached tends to perform better than the parametric approach. This could be due to the fact that the distribution is usually unknown to the scientist and usually does not follow a normal distribution.

Future Work

Although this study represents a positive step in the direction of anomaly detection from data streams, there remains a lot of research to be done in this area.

The initial window size that was used in all the experiments was 300 (except in the case of the third artificial dataset where it was set to 50). The question of selecting an appropriate default window size is a problem that remains to be solved because there may be an optimal sizes that work better. In this case, we estimated the default size experimentally while balancing this search with the desire to use the same standard value in all experiments (except in the case where the data deviated drastcally from the other cases). Second, this work should be seen as a preliminary study into the issue. As mentioned earlier, we used a very simple threshold setting method. In the future, it would be useful to experiment with more sophisticated approaches. Similarly, there is continuous work to be done on designing new change point detection methods from both batch detection and sequential detection. The change point methods used in this study may not have been the optimal ones. With more research, we expect to determine which change point detection methods work best in which cases. Using sequential rather than batch change point detection is another area that would allow us to conduct our experiments in a more practical framework.

References

Bellinger, C.; Sharma, S.; Zaane, O. R.; and Japkowicz, N. 2017. Sampling a longer life: Binary versus one-class classification revisited. In *Proceedings of the First International Workshop on Learning with Imbalanced Domains: Theory and Applications*, volume 74. ECML-PKDD, Skopje, Macedonia: PMLR.

Domingos, P., and Hulten, G. 2000. Mining high-speed data streams. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '00. New York, NY, USA: ACM.

Dong, Y., and Japkowicz, N. 2016. Threaded ensembles of supervised and unsupervised neural networks for stream learning.

Gama, J. a.; Žliobaitė, I.; Bifet, A.; Pechenizkiy, M.; and Bouchachia, A. 2014. A survey on concept drift adaptation. *ACM Comput. Surv.* 46(4).

Gama, J. a.; Rocha, R.; and Medas, P. 2003. Accurate decision trees for mining high-speed data streams. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '03. New York, NY, USA: ACM.

Hulten, Geoff Spencer, L., and Domingos, P. 2001. Mining time-changing data streams. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '01. New York, NY, USA: ACM.

Japkowicz, N.; Myers, C.; and Gluck, M. A. 1995. A novelty detection approach to classification. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, volume 2.

Japkowicz, N. 2001. Concept-learning in the presence of between-class and within-class imbalances. In *Proceedings of the 14th Biennial Conference of the Canadian Society on Computational Studies of Intelligence: Advances in Artificial Intelligence*. London, UK, UK: Springer-Verlag.

Manevitz, L. M., and Yousef, M. 2002. One-class svms for document classification. *J. Mach. Learn. Res.* 2.

Moya, M. M., and Hush, D. R. 1996. Network constraints and multi-objective optimization for one-class classification. *Neural Networks* 9(3).

Pettitt, A. N. 1979. A non-parametric approach to the change-point problem. *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 28(2).

Rutkowsk, i. L.; Jaworski, M.; Pietruczuk, L.; and Duda, P. 2014. Decision trees for mining data streams based on the gaussian approximation. *IEEE Trans. on Knowl. and Data Eng.* 26(1).

Scholkopf, B.; Platt, J. C.; Shawe-Taylor, J. C.; Smola, A. J.; and Williamson, R. 2001. Estimating the support of a high-dimensional distribution. *Neural Comput.* 13(7).

Tan, S. C.; Ting, K. M.; and Liu, T. F. 2011. Fast anomaly detection for streaming data. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Two*, IJCAI'11. AAAI Press.

Webb, G. I.; Hyde, R.; Cao, H.; Nguyen, H. L.; and Petitjean, F. 2016. Characterizing concept drift. *Data Mining and Knowledge Discovery* 30(4).