Online anomaly detection in univariate data streams

Carlos Manuel M. Branco
Lisboa, Portugal
carlos.m.branco@tecnico.ulisboa.pt

ABSTRACT
Anomaly detection is a crucial field nowadays, ranging from fault detection in machinery, surveillance, fraud detection, to many others. However, it is not always easy to tackle the volume and speed of arriving data, and issue anomaly scores on it in an instantaneous manner, as usual in datastreams. The present work centers on the problem of detecting anomalies over continuous and endless univariate time series - *datastreams*. In particular, we propose to use the Matrix Profile method for identifying discords, and managing them both probabilistic and similarity wise, over both the original and differentiated datastream. Additionally, we used the fourier analysis to identify the main frequencies within the time series, in order define the window size parameter. Finally, we validate our method using experimental results over well known dataset in anomaly detection.

Author Keywords
Anomaly detections; Timeseries; Datastreams; All Pair Similarity; Discord Management; Matrix Profile.

INTRODUCTION

Finding interesting behaviours and events is hardwired into our brains due to millennia of evolution. But what characterises this difference, and distinctness in events? Is it something coarse, or more subtle to a point where one might almost not distinguish it from what is normal?

Anomaly detection has garnered considerable attention in the last couple of years [2]. In the case of datastreams, several problems are frequent, such as the volume of data generated, possible endlessness of data, timely detection of change and anomaly, lack of standards for classifying said streams, and the ever changing nature of data, a phenomena known as *concept drift*, are recurrent.

In order to address the problem at hand, detailed objectives for the task should be set out to guarantee the quality of the model. Numerous works have emerged in this area of study over the past years, each of which tackling its unique problems. Putting things briefly, the main problem is on how to maintain knowledge of the past in order to infer whether the present is normal or abnormal, while keeping as few data as possible, answering in minimal time.

In this work, we propose an unsupervised algorithm capable of learning the normal behaviour of a datastream, calculating an anomaly score for each time instant, with minimal and constant overhead. This learning mechanism works on top of Eammon Keogh's Matrix Profile [7], using its resulting distances to identify anomalies.

The main contribution of this work is the adoption of Eammon Keogh Matrix Profiles algorithm to work over a univariate datastream, for the detection of anomalies in a given domain. This is done via two main approaches for the management of discords reported by the computed matrix profiles, namely one probabilistic, and one distance based. In addition, we propose to apply the methodology not only over the original domain, but also on the differentiated sequences.

Finally, a method for automating the choice of an appropriate window size via the Fourier analysis is also introduced and validated.

The rest of the paper is organized as follows: In section 2 we review related work and introduce the necessary background definitions. In section 3 we introduce our algorithm and its statistical. sequence, and automatic variants. Section 4 sees a detailed evaluation, and comparison of our algorithms stating whether they are, or not fit for the detection task. Finally, we offer conclusions and directions for future work respectively.

BACKGROUND

Anomalies can be widely defined as patterns in data that do not conform to expected behaviour. These can fit into one of the following three types [2]: *Point Anomalies*, if an individual data instance can be considered anomalous with respect to the remaining data; *Contextual Anomalies*, if a data instance is anomalous in a specific context, or *Collective Anomalies* if a collection of data points is anomalous with respect to the entire data set.

A data stream is a potentially unbounded, and ordered sequence of instances that arrive over time. A simplification is to assume that the data at hand was generated from sources that do not experience byzantine faults, i.e. sensors will always emit "correct" measurements, and do not fall under states where inaccurate data is generated for the mining process.

Due to the evolving nature of generated data and patterns present in it, we must differentiate between anomaly and novelty. Novelty detection corresponds to identifying an incoming pattern as being hitherto unknown. This phenomena arises if we consider that data is expected to evolve over time, by being generated from dynamically changing environments where non-stationarity is typical [4].

If we blindly applied traditional anomaly detection methods to non-stationary time series, resulting conclusions would be deemed meaningless as values only have a meaning within specific context frames, and no meaning in others. Furthermore, we may want to integrate these new concepts in our models in order to better approximate real life usage condi-

tions. Concepts can change regarding prior assumptions or posterior concepts. This can be either by changing apriori distributions, or evolving posterior concepts [4] (something that was once unusual has become normal).

Literature Review

Machine learning algorithms can learn in two different modes namely offline or online. *Offline learning* algorithms receive the complete sequence beforehand, whereas *Online learning* algorithms are continuously presented with the data produced, up to the current moment, one point at a time [5].

Because of this clear distinction, and given the nature of streaming data along with the necessity to issue instant decisions, online training modes are mandatory.

Furthermore, we can group algorithms based on their ability to handle received data, and whether or not it is labelled. *Unsupervised, Semi-Supervised and Supervised* tasks are the go to when we have no labels, one class labelling or full labelling of data, respectively. The labelling process itself (required for some tasks) can be extremely costly with labels becoming available only after long periods in some cases [2]. Hence, unsupervised learning is usually the most appropriate approach in the context of anomaly detection in data streams.

Orthogonally, anomaly detection techniques can be grouped based on the major approach they follow.

Model based algorithms use classification methods to train a *model*, or rather a classifier, to distinguish between normal and anomalous instances, in a given feature space. One way to tackle the problem is via neural networks [6]. The basic idea for this set of algorithms is that during a first phase a neural network is trained on the normal training data to learn what the normal behaviour is, and then, provided a given input, the network will output if said instance is normal or abnormal. Another way to approach anomaly detection is via a Bayesian approach where the likelihood of observing the test instance given a class prior is estimated from the data set and can be used to detect an anomaly.

Parametric techniques assume that normal data is generated by a parametric distribution with an associated probability density function. Usually, the anomaly score of a test instance is the inverse of the probability density function. Alternatively, statistical hypothesis tests can be used to see if a given test accepts or rejects a value as an anomaly. A couple examples of said methods are the ESD (extreme studentized deviate test) also known as Grubbs test which is used for concept drift detection, or the mean confidence interval. Gaussian based models can be used to estimate maximum likelihood scores and other parameters. Regression models can also be used, however, the presence of outliers and anomalies in the training data can influence the regression model learned.

Rule based anomaly detection techniques learn rules that capture the normal behaviour of a system, while unseen/abnormal behaviours are classified as anomalies. Association rule mining has been used for both network intrusion detection, and credit card fraud detection. Frequent itemsets are generated in the intermediate steps of association rule mining. A proposed approach is that for categorical data sets the anomaly score of a test instance is inversely proportional to the number of frequent itemsets in which it occurs.

Distance based approaches rely on distance metrics (like Euclidean, Jaccard, Mahalanobis and others) to infer the outcome of an observation. The assumption here is that similar, thus close, observations will have the same outcome. Furthermore, we can use the notion of density as it is inherently distance related. Nearest neighbour techniques [1] take into account the assumption that normal data instances occur in dense neighbourhoods, while anomalies occur far from their closest neighbours, or in neighbourhoods of their own. Support Vector Machines are another class of learning techniques, used to learn the separating boundary between normal and abnormal instances. Kernels, such as radial basis functions, can be used to learn complex spaces. Several domains make use of SVMs such as novelty detection in power generation, anomaly detection in temporal series, anomaly detection in audio, and others [2].

Density based anomaly detection techniques estimate the density of the neighbourhood of each data instance. Instances that lie in a neighbourhood with low density are declared to be anomalous. For a given data instance the distance to its k^{th} nearest neighbour is equivalent to the radius of a hypersphere, centred at the given data instance. Density based techniques tend to perform poorly if the data has regions of varying densities. Inherently, density measures require the use of a distance metric.

Clustering-based anomaly detection techniques can follow one of three assumptions [2]: Normal data instances belong to a cluster in the data while anomalies do not belong in any cluster; Normal data instances lie close to their closest cluster centroid while anomalies are far away from said centroid, or that anomalies form clusters by themselves different from the normal ones. These methods are similar to Nearest Neighbor based techniques as both require distance computation between pairs of instances. The key difference between the two techniques is that clustering based techniques evaluate each instance with respect to the cluster it belongs to, while Nearest Neighbor techniques analyze each instance with respects to its local neighbourhood.

Contrary to parametric techniques *non-parametric* statistical models are used , such that the model structure is not defined a priori, but is instead determined from the data. Techniques under this class make fewer assumptions regarding the data such as smoothness of density, when compared to parametric techniques. Here we can find histogram based techniques, kernel based function techniques, hidden markov models and expectation maximization.

All Pair Similarity

Another way to look at the problem is via symbolic representations, and their relative distances (which implicitly model the discovery of patterns in sequences with the distance as a measure of similarity/dissimilarity).

One way to approach this problem is via the all pairs similarity joins for time series [7] which makes extensive use of distance

based methods.

The basic task is that, given a collection of data objects, we want to retrieve the nearest neighbour for each object, where an object can be a sequence of events, belonging to a larger subsequence, present in the time-series.

The output of the method is a series of distance profiles containing the distance between each sequence and all other possible contiguous sequences of a given size, with the highest values corresponding to the largest discords between the selected pattern at a given point in the time-series, and all others. The lower distances for points correspond to a high similarity between said sequence pair. Therefore, it is easy to infer that when looking for anomalies, we are actually looking for distance profiles with high distances, thus indicating the uniqueness (and disagreement) of the sequence with respect to the entire time series.

Evaluation

Despite a series of well known formulas for evaluating models being approached, none seems to tackle the problem of evaluating data streams, as most of the methods are after the fact and do not take into account the continuous nature of time. However, Numenta has proposed a benchmark in order to aggregate a series of these formulas under a constrained environment, in order to provide a platform for model comparison. The Numenta benchmark [6] seems to be the only one so far to aggregate these metrics and provide a benchmarking platform for online time series anomaly detection, thus we will be using it in order to compare our achieved solution with other proposals. This provides a uniform evaluation environment in parameters such as timely detection, and outcome profile which takes into account the weight of false positive *vs* false negatives.

This is the most advanced benchmark for algorithm comparison (in the streaming anomaly detection domain) as it comprises not only annotated datasets but also incentives early detection, penalizes late/out of time detections while providing profiles for weighting false positives and false negatives (for cases where false negatives are much more costly than a closer inspection triggered by a false positive). It uses weighting profiles that work by giving different weights to false positives, and false negatives, and takes into account timely detections via an inverse sigmoid scoring function.

In addition, there is a high unbalance in the label outcome (predominantly no anomaly), something that is also taken into account by the Numenta scoring tool. Multiple evaluation metrics for models are approached in the literature [3]. Among these measures we have Accuracy, Error Rate, Sensitivity or Recall, Precision, Specificity, and F-Measure. Nonetheless, some pitfalls should be avoided when choosing the evaluation steps, and methods as these may not reflect the true quality of the proposed model, additionally biasing the learning process, and model selection. While some metrics take into account class imbalance others do not, where some are pessimistic by reflecting errors made during a warmup phase, others apply fading factors to circumvent this problem.

ANOMALY DETECTION VIA DISCORD MANAGEMENT

In this work, we propose to address the problem of anomaly detection over data streams, through the use of matrix profiles and all pair similarity search, introduced by Eammon Keogh [7]. In this manner, we aim for using the matrix profiles for computing and storing the distances between the arriving data and the stored, while making use of a distance-based similarity measure to identify anomalies. As previously noted, we face several challenges, inherent to the streaming nature of our problem:

- Guarantee the time and space needed by our method is kept almost constant;
- Find a set of parameters as close as possible to the optimal solution of the problem;

What we propose to do is to make use of, and adapt the *Eammon Keogh matrix profile* and the *all pair similarity search* (APSS), in order to work with data streams and detect anomalies in an online manner.

One way to approach the first problem is by maintaining a compact enough representation of the last seen instances, as it will guarantee the constant requirement. Furthermore, by setting a maximum size for the number of discords/motifs to keep, we can further ensure the previous statement.

The main disadvantage of matrix profiles is being a visual method. Consequently, there is no automatic strategy for the identification of the the best parameters, namely the window size, number of top discords, size of exclusion zone, and others. Therefore, we will try to address this issue by testing different dataset sizes (kept as memory of a near past) to infer whether or not they impact the final results achieved.

Nonetheless, other problems have to be addressed such as the size of the window to represent the discovered patterns, the length of each considered temporal context (for how long is data considered current data, and after how much time we refresh the motifs/discord storage, for example).

Algorithm cycle overview

Prior to feeding any data, the algorithm must be initialized with a set of parameters, namely the dataset size to keep (*S*), window size to scan the dataset (*W*), number of discords to extract (*top_K_discords*), number of discords to keep (*n_discords_keep*), time to live of the discords (*TTL*), the threshold for the similarity function to trigger an insertion (*discords_sim_threshold*), decay rate of kept anomalous subsequences (*decay*), a number of points after which the issuing of scores begins (*start_evaluation*), and an exclusion zone of the retrieved anomalous subsequences (*ex_zone*). Only then our loop is ready to work.

After initialization, for each single data point received, *P*, the learned model is kept via an handle function that receives each data tuple. Said tuples are comprised of a timestamp and its respective value. Multiple calls on the handle function cause the data points to build up an ordered set, which is the basis of our profiling analysis. The size *S* of this set can be seen as an event horizon, as exceeding this size will cause the loss of information due to being dropped from the dataset.

On top of this dataset, we will use the matrix profiles with a sliding window of size *W*, with which we scan the previously kept entities, and extract the corresponding matrix profiles. Consequently, this window corresponds to the granularity at which we are comparing patterns, more specifically their length. Upon extraction of the matrix profile from the dataset, we can compute the `top_K_discords`. On the iteration previous to the beginning of the evaluation we will also set (if specified) the minimum distance. The minimum distance corresponds to the first discord's distance profile.

Then, and with the extracted *top k discords*, we start keeping an anomaly frame of size *n_discords_keep*, which will accommodate the found anomalies, thus working as an anomaly database.

The anomaly score issued by the proposed method is tightly connected to this frame since the anomaly score will be 1, thus indicating anomaly whenever a new anomalous pattern is inserted, or replaced in case the database is full. The insertion of these patterns is controlled by a threshold cut on the resulting value returned by a similarity function of two sequences, where the first sequence is a retrieved anomaly sequence from the *top_k discords* method, and the other is a sequence already within the anomaly database. The first discording sequence is inserted trivially.

As mentioned previously, this process will run indefinitely and can be summarized as shown ahead in figure 3.1.

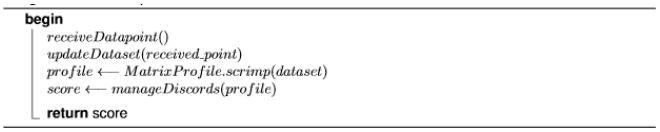


Figure 1. Adapted Matrix profile overview algorithm

Dataset maintenance

The function that maintains the dataset bound to S elements, *updateDataset(data point)*, works by gatekeeping a data window of size **S**. Until this window is full points are trivially added. Upon being full the oldest point is deleted, followed by appending the new one at the beginning said window. The indexes are then reset in order to reflect the change in time, with all points moving down one position in the index of the window. Therefore, we can state that this dataset is in fact a buffer similar to a first in last out (FILO) queue based on it's behaviour.

Further ahead we will be analysing the impact of the size of this dataset on the results obtained by the method.

Matrix Profile Computation

The Matrix Profile is calculated as usual. By providing a dataset of size **S**, and sliding a window of size **W**, we retrieve the corresponding data profiles. The profiles correspond to the distance of the subsequence under analysis to all other subsequences. Then, with these profiles we will extract the highest profile distances, corresponding to anomalous sequence beginnings. With these partial results we can later reconstruct the sequences and check their degree of abnormality. However, this process will only happen for discords with a distance greater than the minimum, which is the case of our examples further ahead.

Our assumption is that if the dataset analysis for the last seen instances produces a very different anomaly list than the ones kept, then the seen point must be anomalous due to its impact on the analysis.

Furthermore, and due to the trivial match limiting zone introduced in Eammon's Matrix Profile implementation [7], we can only start evaluating once the dataset is, at least, twice as large as the window.

Working under these assumptions, we will see how we can set a triggering anomaly mechanism around the similarities of the retrieved discording sequences.

Discord Management

After computing the matrix profile for the data arriving, we can compute the `top_K_discords` by selecting the K largest distance profile values. If a point is contained in a subsequence that cannot have length *W*, namely sequences starting in indexes higher than the dataset size minus the window size (S-W), it is said to be non valid and consequently not retrieved.

First, we start by iterating the distance profile searching for the highest valued index. Then, we extract that entry into the `top_K_discords` list, setting all consecutive values before and after it to INF, from the reported position forward and backward, up to a quarter of the window length *W*, corresponding to the exclusion zone. After doing this process k times we end up with k different anomalous points. If not enough points can be found (due to the length of the matrix profile being too short for a given k number) we simply report a sentinel value that is later filtered.

After retrieving these indexes, they are filtered for uniqueness, eliminating points that occur when the method is unable to provide K different, non contained, distinct discording sequence starting points.

Another criteria used to filter out points is their distance profile. A minimum distance, corresponding to the maximum distance for the highest distance profile in the first evaluation iteration, is further used to filter out results. If any given subsequence start position's distance profile is greater than the first one found, the resulting subsequence is then deemed as valid for comparison, and discarded.

Distance Based Discord Management

With the previously found points we can reconstruct the valid sequences by going to the respective indexes in the dataset kept, retrieving their values and appending the W succeeding values.

The aforementioned sequence is then directly inserted in the discord database if the latter is empty. Otherwise, the sequence is compared for similarity with previously existing sequences present in the discord database, in order to assert whether the anomalous sequence is inserted or not, as resumed in the algorithm in figure 2.

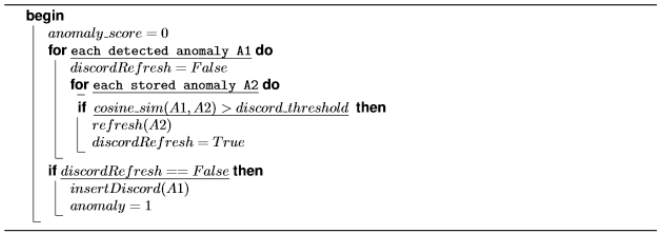


Figure 2. Discord Maintenance overview algorithm

Upon insertion, we register not only the found sequence but also the time of occurrence, a counter of occurrences starting at 1, and a **time to live** (TTL) . The *TTL* of the anomaly works as a forgetting mechanism. In each iteration of the proposed algorithm all discords *TTL*'s are decreased by a decay factor. If the *TTL* of any discording sequence recorded ever reaches zero the anomaly is removed from the anomaly database. In contrast, if the discord list ever reaches a size of **n_discords_keep**, the one with the lowest ttl is discarded and the new discord is inserted.

As a consequence at least 1 anomalous sequence will always be kept, with the first insertion not counting towards the anomaly score as it is trivially inserted. Moreover, this mechanism ensures that multiple consecutive sequences are only reported once, as they will have multiple high similarity scores among themselves.

For this task the cosine similarity was chosen as a similarity metric. This similarity function choice was motivated by the bounded nature [0,1] of the method, that results in a simple and intuitive threshold selection parameter representing percentage of similarity. Other measures, mainly unbound ones, might suffer from the problem of interpretation of the result. The euclidean distance was also considered as a similarity metric. However, it was not chosen as it is a distance metric rather than a similarity one, which would result in a more difficult interpretation and calibration of the resulting value, namely the similarity threshold as a distance. The cosine similarity is generally used as a metric for measuring distance when the magnitude of the vectors does not matter.

Nonetheless, it is important to stress that many other distance/similarity measures could have been chosen such as the Kullback-Leibler, Chebyshev distance or even dynamic time warping (DTW). If the nature of the input were to be symbolic, different measures such as the Jaro or Jaccard similarities would be more appropriate.

After finding their similarity (via the cosine similarity), if it is higher than a given threshold, **discord_sim_threshold**, it will cause a refresh on the anomaly database.

If the result of the comparison between a sequence and all the discords in the discord database never exceeds a similarity threshold then the sequence is considered to be a new anomaly and inserted, corresponding to a no refresh and triggering a maximum anomaly score.

Upon insertion, the return value of the anomaly score for that time step is set to one and later returned, with multiple insertions having the same impact in the score as a single insertion.

Otherwise, if any sequence comparison is equal or higher than the **anomaly_threshold** it will cause the anomaly being compared to, in the discord database, to refresh it's TTL. In such case, the anomaly score will remain the same as it was before that given iteration. The anomaly score is then returned after all sequence comparisons are made.

After completion of the discord database maintenance task an anomaly result for the current data point will have been found.

The result of putting all the aforementioned steps together is a 3 parameter model with the parameters **S**, **W**, and **anomaly_threshold**. This model is expected to be able to discriminate, for each data point, if it is normal or abnormal.

Probabilistic based Discord Management

Another way to tackle the aforementioned problem is by using the raw distance value. The most straightforward test we can run on the raw distance profile is a mean confidence interval. The objective here is to check whether a value is within a given confidence interval. For this method, once we filter out all the resulting discording distances, we can start accumulating them. Once we have a universe of distances greater than 3 samples, we can start performing a mean confidence interval test. Then, we iterate through the discording distances and compute the mean, lower bound, and upper bound for all the distances seen so far.

Similarly to what was previously approached, we will also need a scoring function, responsible for issuing anomaly scores for each time step. One way to approach this is via the mean confidence interval over the reported discording distance profiles.

If a distance value is within the mean confidence interval, at a given percentage, then the resulting anomaly score is zero. Consequently, the anomaly score for that timepoint is the greatest value of the calculation of the distance profile minus the mean, for all the reported discords in that timestep that are not within the confidence interval.

The resulting score is normalized by the max seen distance within the kept distance vector. Finally, all distance points that were filtered are simply added to the seen distance vector, and the anomaly score returned.

EXPERIMENTAL RESULTS

The goal of the experimental procedure is to achieve the best possible classifier with minimal overhead in variable adjustment.

Note that the method made available by the matrix profile foundation will end in error due to the subsequence length being too large if, at the moment of analysis, the dataset is lower than two times the window size.

A total of three parameters, namely dataset size (*S*), matrix profile window size (*W*), discord similarity threshold (`disc_sim_threshold`) are fully tested.

In addition, the parameter that controls the start of evaluation is permanently set to be the dataset_size length minus one. The decay rate parameter will be fixed to one, with the time to live of the discord (in cycles) equal to the dataset size (TTL = dataset_size). In addition, the exclusion zone of the discord calculation is fixed to a quarter of the window size.

The choice for the window sizes is between 1 hour (namely two datapoints) and 48 hours (96 data points), that are tested for increasing dataset sizes between four times and nine times the window size.

We then select the best classifier, and the similarity threshold is varied in order to detect whether a more lax or restrictive approach is better.

Another set of experiments will be done over the differentiated version of the dataset (where each point corresponds to the difference between itself and the previous observation). This set of experiments will hereon be called as the sloped or differentiated version of the dataset, as it corresponds to the momentums of the original domain trivially represented as slopes.

The used dataset is the NYC taxi domain, containing passenger usage numbers of taxis, reported in an hourly basis. This dataset is contained, along others, in the Numenta Benchmark.

Results are shown for each dataset size variation by window size. Window sizes are represented in the X axis, nab score (blue), sensitivity (black), specificity (red), and precision (green) in the Y axis. Each vertical line is a simulation with its respective results.

Statistical approach over the taxi dataset

The workflow for this classifier is as previously introduced in section 3.4.2. Results for a T student confidence interval of 99% over the filtered distance profiles, using the original taxi dataset, are show in figure 3.

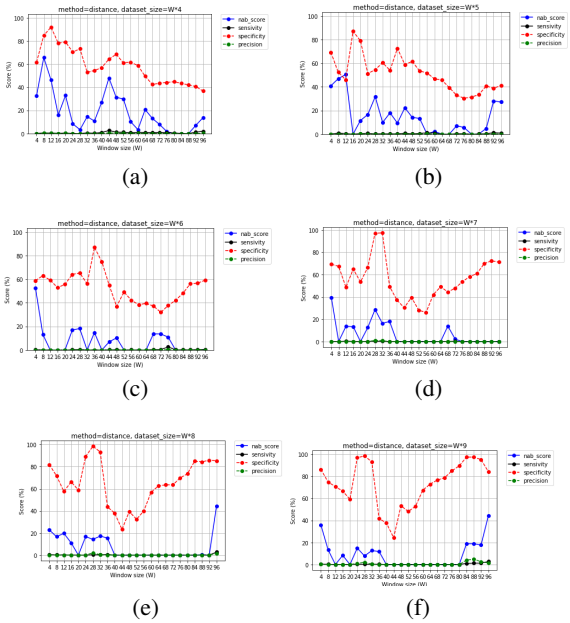
The first thing that stands out in the analysis of the results for this experiment is the extremely oscillating specificity. Without a mechanism to control the maintenance of sequences it is clear that, by solely using the distance profile of the discords, we cannot establish a proper baseline behaviour. The NAB score, on the other hand, shows that despite the low precision, the learned models managed to detect some anomalous data-points.

However, taking into account that the precision values are also associated with the very low specificity values, we have indication that the detections were merely coincidental (as a collateral result of the low specificity). In addition, the detection labels are binary, but the detection score is not, this creates a problem for basic metrics.

Given the aforementioned aspects, we should always look at the NAB score for indication of performance, and only then analyse the remaining metrics. Traditional metrics will fail given that the anomaly score is equal to the normalized distance to the mean of previously seen distance values, and this [0-1] value cannot be translated as precision. specificity or sensitivity straight forward. The Numenta benchmark takes care of this via an optimizer, that reflects the score for the best threshold found (given all detection values).

In addition, the size of the dataset does not seem to influence the learning process. If we take a closer look figures 3(a), 3(b) 3(c), and we compare them against 3(d),3(e),3(f), the average score went down as the dataset size went up, in most cases underachieving the results of the first three.

Accordingly, the best classifier identified corresponds to figure 3(a), when the window size is equal to 8.



(a) (b) (c) (d) (e) (f)

Figure 3. Results for multiple runs with varying dataset sizes of the probabilistic method at a confidence interval of 0.99;

Distance Based Discord Management over the original dataset

The following subsections concern with the non probabilistic version of distance based method. The algorithm tested in the following subsections is as introduced in section 3.4.1. For the first set of experiments the unmodified taxi domain will be used, whereas for the second part of the experiment the differentiated dataset is used.

For the distance based discord management algorithm over the original dataset something recurring in all simulations is that the specificity parameter remains relatively high (over 95%). Despite the window size variation, no major changes can be found within each run. On the other hand, the sensitivity parameter remained relatively low ($<$10%), despite being higher than all experiments so far. This is due to the fact that when we define the temporal period for which an anomaly detection interval is valid, we inherently define a zone where the true label is extended through time not being a single point anymore, with this being true for all experiments.

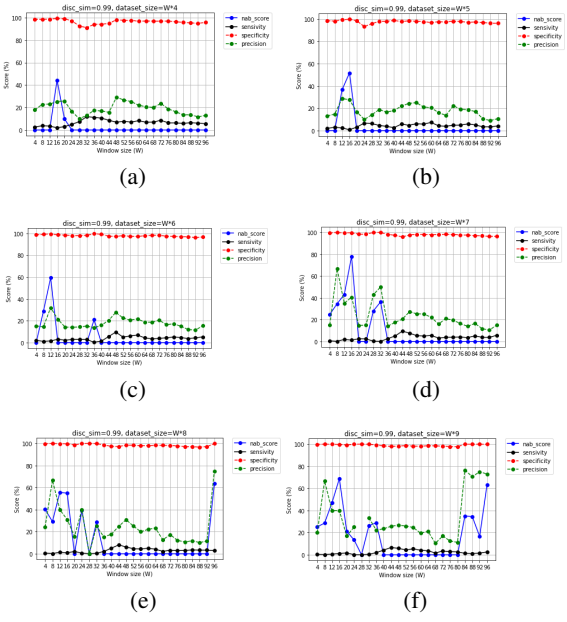
In other words, only the earliest valid detection would be enough to get a high score however, all points around the anomaly would also have be to equally classified in order

to achieve a high sensitivity. Consequently, the analysis of this parameter by itself might not be sufficient to explain the quality of the achieved model.

Nonetheless, it provides valuable insight on the behaviour of the model when used with other indicators. All sub-figures in figure 4 show this behaviour clearly, where once the sensitivity increases, the NAB score decreases and vice versa. This trade-off is due to the increase in wrong classification, thus it suffers from the same problem as the specificity in the previous section. In order to achieve both high sensitivity and specificity, we would need to make continuous correct, and exact predictions for score for the whole interval around the anomaly, which would be quite difficult.

On the other hand, we can also see that high precision models tend to have high NAB scores such as, the beginning, and the end of the window size range (low and high respectively) in figures 4(d),4(e), and 4(f).

Furthermore, we can see that an increase in the dataset size resulted in an increase in the average NAB score for the run. It is however important to stress that some runs may not have a precision score at all (represented by the lack of connection between the two neighbouring points). In these cases no point tested as anomalous, and the run stands out by the lack of a connected line for this parameter. e.g figure 4(f), window size = 28.



(a) (b) (c) (d) (e) (f)

Figure 4. Results for multiple runs with varying window size, and fixed similarity threshold of 0.99;

If we take a close look, we will find that figure 4(f) contains the best run found, due to the high precision rate associated with high NAB score. Thus, we will set the dataset size to 9 times, and will vary the discord similarity threshold parameter as show in figure 5

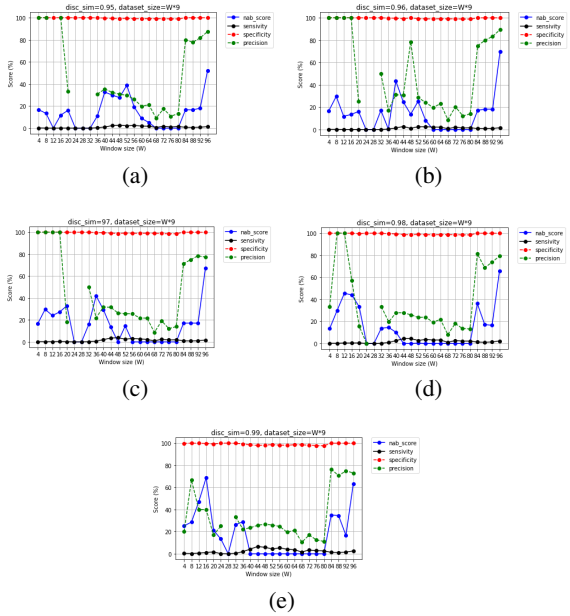
Likewise, we also see the same lack of precision, always associated with a NAB score, and sensitivity. As a consequence, the specificity parameter for these points is almost 100%, as intended and expected.

If we further test the impact of the similarity threshold, represented in figure 5, we can see that for thresholds between 0.95 up to 0.98 at a dataset size of 9x, some of the models with lower window sizes managed to achieve perfect precision scores, implying that all the detections made were correct. Yet, the NAB score for these models was relatively low. This is due to the fact that for lower thresholds we are too lax, only allowing big changes in the sequences to be detect. As a result, some abnormal sequences manage to fly under the radar, going by undetected.

With this in mind, we expected the more restricted models to achieve better results. However, a slight relaxation in the similarity of the sequences proved to increase the precision vastly and in some cases, and also the nab score in others.

These experiments were run for for all similarity thresholds, window, and dataset size combinations, despite not being shown for the sake of simplicity while having similar repeated behaviour.

We conclude that, when compared to the simple statistical method, we have achieved a much better detection with less false positives, less false negatives and more true positives. Furthermore, the best found setup corresponds to the window size of 96 in figure 5(b). The NAB was score was respectively 69.99.



(a) (b) (c) (d) (e)

Figure 5. Results for window size variation by discord similarity threshold, for a fixed dataset size of 9x the window size.

Distance Based Discord Management over the slopped dataset

Similarly to the experiments in the previous section we will repeat the same evaluation process except now, we will be using the differentiated dataset. The results for this new dataset,

corresponding to the sloped taxi time series, can be found in figure 6.

If we take a closer look, we can see that, as expected, the normal behaviour is broadly captured. In all experiments 6(a) through 6(f), the specificity remained relatively high, indicating the majority of points were indeed true negatives. On the other hand, and contrary to the previous results on figure 5, we see a lower specificity rate. Despite this, the NAB score and precision turned out to be relatively high in comparison.

We can also see that throughout all figures, when the window size is lower than 48 points, the sensitivity is higher than any other previous experiment. This tells us that any time an anomalous point is identified, most surrounding points were also found to be anomalous. However, this came at the cost of an unstable baseline behaviour that, once again, when we reach the 48 mark, stopped happening. In addition, the instability in the specificity associated with defining the baseline behaviour stops after said period.

It is curious to find that the method over the differentiated series spikes (similarly to the previous subsection) in all measures around the 48 point mark, corresponding to the 24 hour period in the dataset.

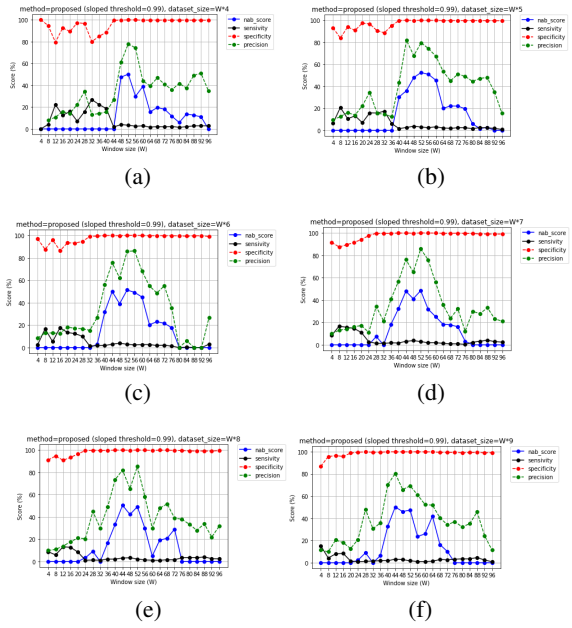


Figure 6. Results for window size variation by dataset size, for a given similarity threshold, corresponding to the sloped (differentiated) version of the taxi domain.

Finally, the variation of the threshold parameter (similar to the previous subsection) did not produce any noticeable change in the algorithm results over the differentiated dataset. Thus, we can say that the differentiated series is far more resilient to variations in the similarity parameter than the original dataset. The choice for this new set of experiments was done as in the previous subsection, by identifying the best classifier in the first set, and setting the window size constant while varying the discord similarity threshold. For the sake of simplicity, the results for this new set of experiments will not be plotted.

In sum, the best classifier found for this set of experiments corresponds to figure 6(c) where *window size = 52*, *dataset size = window size x 6*, and *discord similarity = 0.99*. The final NAB score for this classifier was of 49.11.

Automatic window selection via Fourier Transform

Finally, the automation mechanism will be tested in order to assert whether or not the classifier does produce viable results.

The parameters were kept as in the previous experiments corresponding to the sequence comparison. The parameter variations remains the same almost the same. namely the dataset size and similarity threshold. For the the window size parameter, the dominant period found by the Fourier Transform over the accumulated dataset is used, prior to starting the evaluation;. The dataset will range from as little as 1 week to to an entire month, with a 3 day step increment.

The first thing we notice from this set for experiments, represented in figure 7, is that the specificity parameter remained almost at 100%, as well as the sensitivity near 0%. This seems to be a behaviour congruent with the ones previously shown for the distance based algorithm in figures 4 and 5, thus reassuring the learning of the normal behaviour. Next, we can see that for all runs the precision score was lower for lower dataset sizes and higher for higher dataset sizes. Furthemore, with the increase in the dataset size, not only did the precision increase but so did the NAB score. Thus we can state that the larger the dataset provided was, the better the results we got (specially true for experiments 7(a),7(b), and 7(c)).

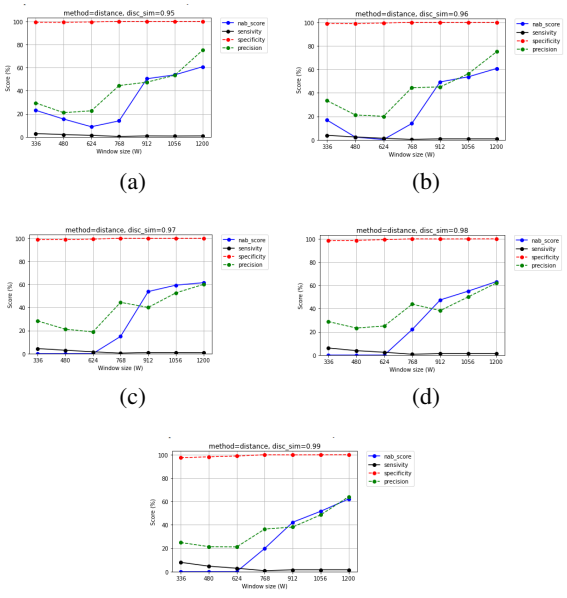
However, something quite remarkable for this set of experiments is that the discord similarity threshold parameter tested had minimal impact in results, as we can see through experiments 7(a) to 7(e). Its increase did however impact the results for dataset sizes between 480 and 912 points, slightly lowering the results of the NAB for tighter similarity values in this interval. This implies that the higher the similarity threshold, the later the detection took place, thus the lower NAB score with median precision.

In sum, for this batch of experiences the best classifier corresponds to a dataset size of 1200 points, for a discord similarity of 0.96 (located in figure 7(e)) that got a total score of 60.71 with 9 true positive classifications, and 3 false positive.

Best model configuration results

In figure 8, we can see the classification score,point by point, when using the Numenta Hierarchical Temporal Memory algorithm. Despite a couple of errors throughout the complete execution, this classifier is capable of identifying correctly 4 out of 5 anomalies with only 1 true error (around the 17th of September), thus making it a very strong candidate for a baseline comparison with a final NAB score of 74.38.

Once again, and given the optimizer that finds the best threshold cut for the classification step (in case the label is continuous and not binary) we may be lead to think that the score would not be humanly readable. However, we see see a clear



(a) (b) (c) (d) (e)

Figure 7. Results for the automatic window selection procedure over the original taxi domain. X axis represents the dataset size in days. Multiple quality metrics shown for the final results of each run.

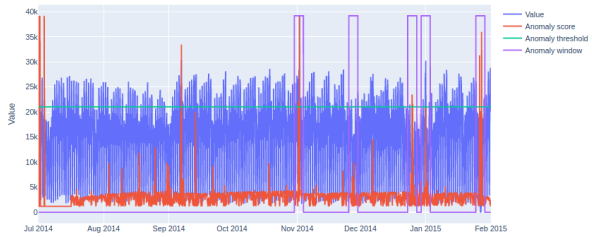


Figure 8. Results for the Numenta classifier.

separation between "normal" score values (below the 15k mark when scaled) and real anomalies (seen as global spikes) with the bare eye.

The best model for the probabilistic version over the taxi domain managed to achieved a score of 65.66 for a window size of 8 and dataset size of 4 times window size. Exact results for the entire classification process can be found below in figure 9

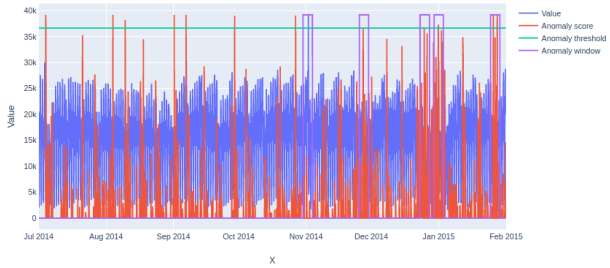


Figure 9. Results for the statistical classifier over the original dataset.

For the Distance Based Anomaly Detection model, using the sequence comparison approach introduced in section 3.4 over the original taxi domain, the results for the best classifier can be found in figure 10. It is the best achieved model due to the high NAB score of 69.99 combined with a very high precision. The setup used to achieved the results corresponds to a window size of 96, for a similarity threshold of 0.96 at a dataset size of 9 times the window. The binary label also removes the problems associated with score value interpretation and optimization.

With this classifier we have managed to detect 4 out of five anomalies, with minimal mistakes, and when these happened they were located around a real anomaly.

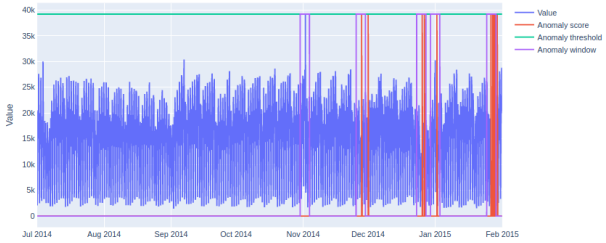


Figure 10. Results for the best sequence comparison based classifier over the original taxi dataset, with anomaly value visualisation. Parameters used are *window size = 96*, *dataset size = 9 x window size* for a `disc_sim_threshold` of 0.96.

The distance based classifier over slopped the taxi dataset managed once again to achieve great results. Detecting 3 out of 5 anomalies, one of which that had previously gone undetected (corresponding to the anomaly in November). It finished with a final NAB score of 49.11 detecting 3 out of nine anomalies with 1 major error, between January and February.

The used parameters were *window size = 52*, *dataset size = window size x 6*, and *discord similarity = 0.99*. The result of the classification can be found in figure 11, and similarly to the distance based sequence discord management method

over the original dataset, it managed to achieve very promising results with 23 true positives, and only 4 false positives. If we gather the results of the classifier under appreciation, for both the sloped, and non sloped domains of the taxi dataset, it manages to detect all annotated anomalies, with minimal parameter adjustment, thus a combination of both would make a great ensemble for this specific data stream represented.

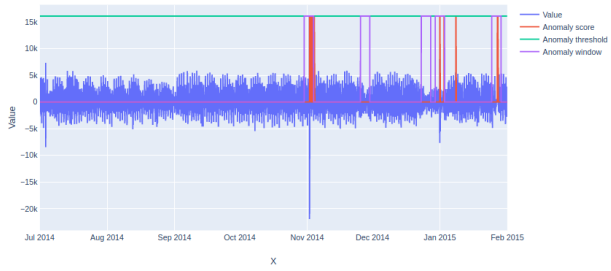


Figure 11. Results for the best classifier found (proposed solution) over the sloped dataset. Anomaly value visualisation.

To sum up this section, the best classifier achieved during this work was our distance based discord management over the original dataset. Not only it achieved an approximate NAB score to the Numenta classifier, but also managed to get a very low false positive rate. However, we consider it to have a much more intuitive score interpretation, thus the choice. It finished with a NAB score of 69.99, issuing 17 true positive labels and only 2 false positive. Last but not least, the distance based sequence comparison algorithm managed to detect the November anomaly only on the differentiated dataset.

CONCLUDING REMARKS

Anomaly detection is a prominent area of analysis for finding anomalous and/or unexpected behaviour in data. One particular area of interested for anomaly detection are datastreams, possible infinite flows of information, where it is often expected to score one information point in time before the next one arrives. Given that that datastreams may be endless, and inherent problem with the volume of data for analysis, space and time arise. One way to do this is to adapt the Matrix Profiles introduced by Eammon Keogh to evaluated univariate timeseries and detect anomalous data points.

Said was shown in this work where two methods are introduced for the detection purpose. Furthermore, a scoring function is designed, as well as a mechanism for the automation of the choice of one of the most important parameters, namely the window size. We have further concluded that it is possible to apply this algorithm to other series of presentations (in our case the differentiated series) with equally good results.

In addition, we have found a way not only to detect anomalies, but also deal with concept drift, as the proposed method inherently detect deviating concepts due the the similarity function used, and the maintenance discord algorithm thoroughly demonstrated.

We have demonstrated that, using the matrix profiles, it is possible to detect anomalies in univariate time series, at par with state of the art anomaly detection algorithms, achieving almost perfect scores. Our best classifier managed to detect 4 out of 5 anomalies from the taxi domain, with a precision of 87%, 17 true positives, 2 false positives and a final NAB score of 69.99%. Consequently, we can state that all the goals set for this work were fully met, namely detection anomalies, in an online manner, over datastreams.

The proposed approach, on Eammon Keogh Matrix Profiles, turned out to be competitive with state of the art algorithms for anomaly detection in univariate datastreams, and parallel computation over the slopped dataset would provide a formidable ensemble detector with minimal overhead added.

ACKNOWLEDGMENTS

This work was supported by national funds by Fundacão para a Ciência e Tecnologia (FCT) through project GameCourse (PTDC/CCI-CIF/28939/2017).

REFERENCES

[1] Evgeny Burnaev and Vladislav Ishimtsev. 2016. Conformalized density- and distance-based anomaly detection in time-series data. (2016).

[2] Varun Chandola, Arindam Banerjee, and Vipin Kumar. 2009. Anomaly Detection: A Survey. *ACM Comput. Surv.* 41, 3, Article 15 (July 2009), 58 pages. DOI: http://dx.doi.org/10.1145/1541880.1541882

[3] João Gama, Raquel Sebastião, and Pedro Pereira Rodrigues. 2009. Issues in Evaluation of Stream Learning Algorithms. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '09)*. ACM, New York, NY, USA, 329–338. DOI: http://dx.doi.org/10.1145/1557019.1557060

[4] João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. 2014. A Survey on Concept Drift Adaptation. *ACM Comput. Surv.* 46, 4, Article 44 (March 2014), 37 pages. DOI: http://dx.doi.org/10.1145/2523813

[5] Richard M. Karp. 1992. On-Line Algorithms Versus Off-Line Algorithms: How Much is It Worth to Know the Future?. In *Proceedings of the IFIP 12th World Computer Congress on Algorithms, Software, Architecture - Information Processing '92, Volume 1 - Volume I*. North-Holland Publishing Co., Amsterdam, The Netherlands, The Netherlands, 416–429. http://dl.acm.org/citation.cfm?id=645569.659725

[6] A. Lavin and S. Ahmad. 2015. Evaluating Real-Time Anomaly Detection Algorithms – The Numenta Anomaly Benchmark. In *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*. 38–44. DOI: http://dx.doi.org/10.1109/ICMLA.2015.141

[7] Chin-Chia Michael Yeh, Yan Zhu, Liudmila Ulanova, Nurjahan Begum, Yifei Ding, Anh Dau, Diego Silva, Abdullah Mueen, and Eamonn Keogh. 2016. Matrix Profile I: All Pairs Similarity Joins for Time Series: A Unifying View That Includes Motifs, Discords and Shapelets. 1317–1322. DOI: http://dx.doi.org/10.1109/ICDM.2016.0179