IndyCar Anomaly Event Detection

Jiayu Li
Indiana University
jl145@iu.edu

Xinquan Wu
Indiana University
xinqwu@iu.edu

ABSTRACT

We see an enormous increase in the availability of streaming, time-series data. Largely driven by the rise of connected real-time data sources, this data presents technical challenges and opportunities. One fundamental capability for streaming analytics is to model each stream in an unsupervised fashion and detect unusual, anomalous behaviors in real-time. In this paper, we use the actual data generated in the IndyCar 500 game as a data set, applying four different unsupervised real-time anomaly detection algorithms and compare the difference in principle and accuracy of the four algorithms.

KEYWORDS

anomaly detection, machine learning, online learning

ACM Reference Format:
Jiayu Li and Xinquan Wu. 2018. IndyCar Anomaly Event Detection. In *Woodstock '18: ACM Symposium on Neural Gaze Detection, June 03–05, 2018, Woodstock, NY*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

The IndyCar series is the premier level of open-wheel car racing in the US. Many events can happen during the racing period, such as pit stops, crashes, mechanical breakdown, and drivers ranking changes. With more than 500 sensors, we can get a large amount of data, including timing and scoring log data. The task of this term project is to use the data to detect some interesting event which can be treated as an anomaly. In this paper, we will use four different anomaly detection algorithms to analyze the data of IndyCar 500 and compare the difference in principle and accuracy of the four algorithms.

1.1 Indy 500

The IndyCar Series [1], currently known as the NTT IndyCar Series under sponsorship, is the premier level of open-wheel racing in North America. Featuring racing at a combination of superspeedways, short ovals, road courses, and temporary street circuits, the IndyCar Series offers its international lineup of drivers the most diverse challenges in motorsports. Indy500 is its premier event at Indianapolis Motor Speedway where the racing cars reach speeds up to 235 mph.

Woodstock '18, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-9999-9/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

Figure 1: The IndyCar Series

Over the years, the technology of motor sports has evolved and teams use various techniques to tweak engines, fuel, bakes and other components of the cars to increase their winning probability. More importantly, the teams have evolved to a level where they collect and analyze data from tracks (temperature, humidity, etc.), cars (speed, rpm, throttle, gear, break, etc.) and even from drivers (heart beat, stress level, etc.) in real time to make decisions and predictions. Hence data analysis and predictive analytics have become an important and a key factor for winning the game. The velocity and volume of such collected data points are massive and goes up to thousands of data points per car per second, which goes beyond the boundaries of human perception. Hence this analysis task has become an interesting Big Data and a real-time machine learning use case. With the advent of smaller but powerful computational devices, the cars and race tracks come fitted with hundreds of sensors and actuators. The sensors in the car's record and transmit various metrics (speed, engine rpm, throttle, steering direction et al.) to the main server present on premises of the Indy 500 race track. These advanced information technology infrastructures support the racing management and the communication between the drivers and their teams. Each race generates a large volume of the telemetry and timing & scoring data, for example in the race of May 27, 2018, it contains 4,871,355 records with consecutive data arrival interval of 120ms for each car.

1.2 Anomaly detection

Anomaly is defined as a point in time where the behavior of the system is significantly different from normal behavior. Anomalies can be spatial or temporal, depending on whether the abnormality

of data instance is considered only in a specific context. Anomalies can also be collective where the system presents abnormal behavior only with respect to their occurrence together as a collection. Anomaly detection in time-series is a heavily studied area of data science and machine learning [6, 7]. Many anomaly detection approaches exist, both supervised (e.g., support vector machines and decision trees) and unsupervised (e.g., clustering), yet the vast majority of anomaly detection methods are for processing data in batches, and unsuitable for real-time streaming applications. For streaming anomaly detection, the majority of methods used in practice are statistical techniques that are computationally lightweight. These techniques include sliding thresholds, outlier tests such as extreme studentized deviate (ESD, also known as Grubbs') [19] and k-sigma, changepoint detection, statistical hypotheses testing, and exponential smoothing such as Holt-Winters [8]. Most of these techniques focus on spatial anomalies, limiting their usefulness in applications with temporal dependencies. Another challenge a real-world anomaly detection algorithm faces is the problem known as *concept drift* which denotes the continuous changing of the standard behavior patterns in real applications. It requires the detection algorithm to learn continuously and adapt to dynamic environments.

2 ARCHITECTURE AND IMPLEMENTATION

2.1 Comparison of anomaly detection algorithms

The algorithm we use is taken from NAB. NAB is a benchmark for evaluating algorithms for anomaly detection in streaming, real-time applications. The NAB scores are normalized such that the maximum possible is 100.0 (i.e., the perfect detector), and a baseline of 0.0 is determined by the "null" detector (which makes no detections).

Table 1: Comparison of anomaly detection algorithms

Detector	Standard Profile	Low FP	Low FN
Numenta HTM	70.5-69.7	62.6-61.7	75.2-74.2
CAD OSE	69.9	67.0	73.2
earthgecko Skyline	58.2	46.2	63.9
KNN CAD	58.0	43.4	64.8
Relative Entropy	54.6	47.6	58.8
Random Cut Forest	51.7	38.4	59.7
Twitter ADVec v1.0.0	47.1	33.6	53.5
Windowed Gaussian	39.6	20.9	47.4
Etsy Skyline	35.7	27.1	44.5
Bayesian Changepoint	17.7	3.2	32.2
EXPoSE	16.4	3.2	26.9
Random	11.0	1.2	19.5

We can see that HTM is the highest-scoring anomaly detection algorithm, and we will focus on HTM.

2.2 Hierarchical temporal memory

Hierarchical temporal memory (HTM) is a biologically constrained theory (or model) of intelligence, initially described in the 2004 book On Intelligence by Jeff Hawkins with Sandra Blakeslee. HTM is based on neuroscience and the physiology and interaction of pyramidal neurons in the neocortex of the mammalian (in particular, human) brain.

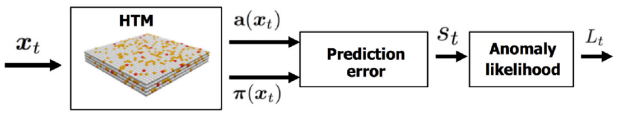


Figure 2: Anomaly Detection System Based on HTM

At the core of HTM are learning algorithms that can store, learn, infer and recall high-order sequences. Unlike most other machine learning methods, HTM learns (in an unsupervised fashion) time-based patterns in unlabeled data on a continuous basis. HTM is robust to noise, and it has a high capacity, meaning that it can learn multiple patterns simultaneously. When applied to computers, HTM is well suited for prediction, anomaly detection, classification, and ultimately sensorimotor applications.

The theory has been tested and implemented in software through example applications from Numenta and a few commercial applications from Numenta's partners. Hierarchical Temporal Memory (HTM) is capable of detecting anomalies from data streams in real-time and performs well on the concept of drift problems. Numenta Anomaly Benchmark (NAB) [16] attempts to provide a controlled and repeatable environment of open-source tools to test and measure anomaly detection algorithms on streaming data. This included algorithms for real-time anomaly detection include HTM, Skyline, Twitter ADVec, KNN CAD, Relative Entropy, Windowed Gaussian, etc. Evaluation on 58 datasets in NAB benchmark shows that HTM is one of the state-of-the-art algorithms that provide stable and high accuracy. Furthermore, HTM is applicable to various operations such as visual data processing [11], object categorization [9], pattern discovery [18], and data classification [17]. Joost van Doremalen et al. applied HTM to speech recognition [10], and David Rozado et al. used Hierarchical Temporal Memory Networks to identify gaze gestures [20]. In addition to academic papers, there are many patents for HTM applications [12, 15] in recent years.

We adopt HTM as the core anomaly detection algorithm in our system. In this subsection, we present a brief introduction to the algorithm. The basic idea of HTM is that it imitates the process of sequential learning in the neocortex of the brain, which is involved in higher cognitive functions such as reasoning, conscious thoughts, language, and motor commands [3–5, 14].

HTM sequence memory models one layer of the cortex which is organized into a set of columns of cells, or neurons, as shown in Fig. 3. Each neuron models the dendritic structure of neuron in the cortex. Sufficient activity from lateral dendrite will cause a neuron to enter an active state. An inhibition mechanism provides a sparse data representation in HTM, in which a cell activated by lateral connections prevents other cells in the same column to enter the active state. Sparse representations enable HTM to model high-order sequences (sequences with long-term dependencies), as in Fig. 3, the same input B in a different context can have different internal representations in the continuous prediction process, and

same input "C" in two sequences invokes different prediction of either D or Y depending on the context several steps ago.

The connection between the neurons is learned from input data continuously. The input, $\mathbf{x_t}$, is fed to an encoder and create a sparse binary vector representation $\mathbf{a}(\mathbf{x_t})$. Then, all neurons update its status by the inputs from connected neurons with active cells. It outputs predictions in the form of another sparse vector $\pi(\mathbf{x_t})$. The prediction error, S_t, a scalar value inversely proportional to the number of bits common between the actual and predicted binary vectors is given by

$$S_t = 1 - \frac{\pi(\mathbf{x_{t-1}}) \cdot \alpha(\mathbf{x_t})}{|\alpha(\mathbf{x_t})|} \tag{1}$$

where $|\alpha(\mathbf{x_t})|$ is the scalar norm, i.e. the total number of 1 bits in $\mathbf{a}(\mathbf{x_t})$. In equation (1) the error will be 0 if the current $\mathbf{a}(\mathbf{x_t})$ perfectly matches the prediction, and 1 if the two binary vectors are orthogonal. Furthermore, anomaly likelihood can be calculated from the prediction error by assuming it follows a normal distribution which is estimated in a previous window. As the likelihoods are minimal numbers, a log transform is used to output the final anomaly score. For example, a likelihood of 0.00001 means we see this much predictability about one out of every 10,000 records, and the final anomaly score is 0.5.

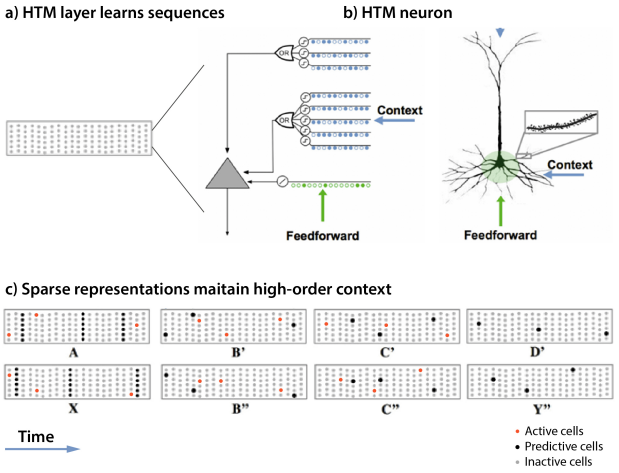


Figure 3: Working of HTM sequence memory [5].

2.3 Bayesian Changepoint

Implementation of the online Bayesian changepoint detection algorithm as described in Ryan P. Adams, David J.C. MacKay, "Bayesian Online Changepoint Detection"[2]. The algorithm computes, for each record at step x in a data stream, the probability that the current record is part of a stream of length n for all $n \leq x$. For a given record, if the maximimum of all the probabilities corresponds to a stream length of zero, the record represents a changepoint in the data stream. These probabilities are used to calculate anomaly scores.

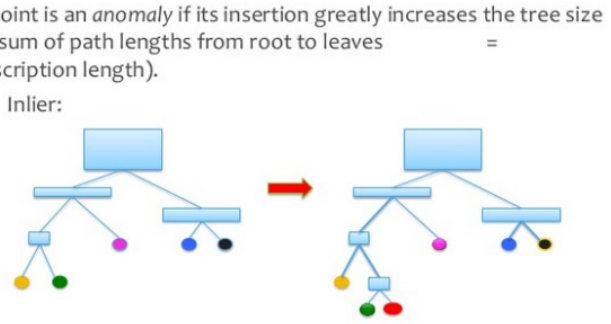


Figure 4: Random Cut Forest

2.4 Random Cut Forest

The main idea behind the RCF algorithm is to create a forest of trees where each tree is obtained using a partition of a sample of the training data. As shown in Figure 4. For example, a random sample of the input data is first determined. [13]The random sample is then partitioned according to the number of trees in the forest. Each tree is given such a partition and organizes that subset of points into a k-d tree. The anomaly score assigned to a data point by the tree is defined as the expected change in the complexity of the tree. As a result adding that point to the tree; which, in approximation, is inversely proportional to the resulting depth of the point in the tree. The random cut forest assigns an anomaly score by computing the average score from each constituent tree and scaling the result with respect to the sample size.

2.5 EXPected Similarity Estimation

This detector is an implementation of The EXPoSE (EXPected Similarity Estimation) algorithm as described in Markus Schneider, Wolfgang Ertel, Fabio Ramos, "Expected Similarity Estimation for Lage-Scale Batch and Streaming Anomaly Detection"[21]. EXPoSE calculates the likelihood of a data point being normal by using the inner product of its feature map with kernel embedding of previous data points. This measures the similarity of a data point to previous points without assuming an underlying data distribution. There are three EXPoSE variants: incremental, windowing and decay. This implementation is based on EXPoSE with decay.

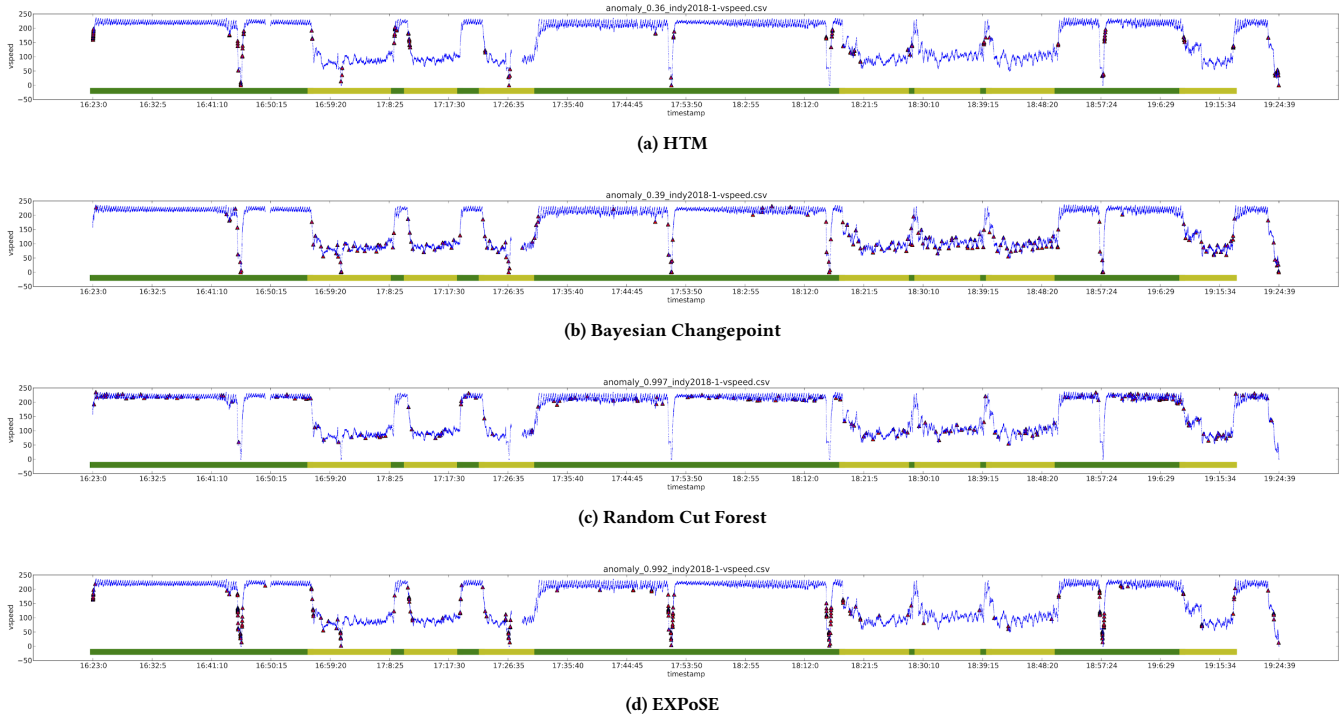
3 EXPERIMENTS

3.1 Hardware

In the experiments, we use a single node of a dual-socket Intel(R) Xeon(R) CPU E5-2670 v3 (architecture Haswell) with 128GB memory. The Operating System is Red Hat Enterprise Linux Server version 7.6.

3.2 Data set

The data set is obtained by the IndyCar timing system which provides two methods to retrieve timing data, that is, serial or sequential data feed for live data and report querying for historical or archived data. The timing data is streamed through TCP/IP socket technology. The data is serially streamed by breaking it as different record types. The multi-loop protocol (MLP) protocol definition



(a) HTM

(b) Bayesian Changepoint

(c) Random Cut Forest

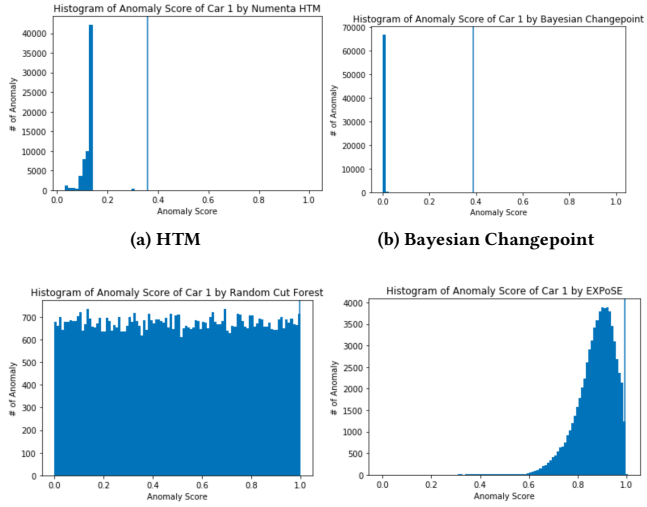
(d) EXPoSE

Figure 5: Comparison of output results of 4 anomaly detection algorithms

includes definitions for record types such as completed lap results, completed section results, flag information, track information and many more.

All archived timing data will be made available on the IRIS web site in the Timing and Scoring area. These databases can be accessed via IRIS on the Internet. The timing database on this site contains data from all IndyCar sanctioned events starting in 2001. This includes all race events. IndyCar has chosen to use Microsoft SQL Reporting Services as the platform for accessing the data in the timing databases. The platform allows end users to run simple queries to access data via a web page interface.

The data set that we have is from one of the IndyCar races on May 27th, 2018 which consist of 768 MB of data and a total of 4871353 records in the log file.



(a) HTM

(b) Bayesian Changepoint

(c) Random Cut Forest

(d) EXPoSE

Figure 6: Anomaly Score Distribution

4 RESULTS

4.1 Threshold selection

The anomaly score is a value between 0 and 1. Since the principles of different algorithms vary greatly, the distribution of the anomaly scores of their outputs is also different. As shown in figure 6, we compute the distribution of anomaly scores of the four algorithms. In order to make the experiment more fair, we need to select different thresholds for the four algorithms so that they can output the same number of anomalous events.

4.2 Data Labeling

The difficulty in verifying the accuracy of an anomaly detection algorithm is that the actual time series are not labeled, and labeling the data is considered to have a certain subjective error. Here, we combine the time-speed graph with the game video to mark all the

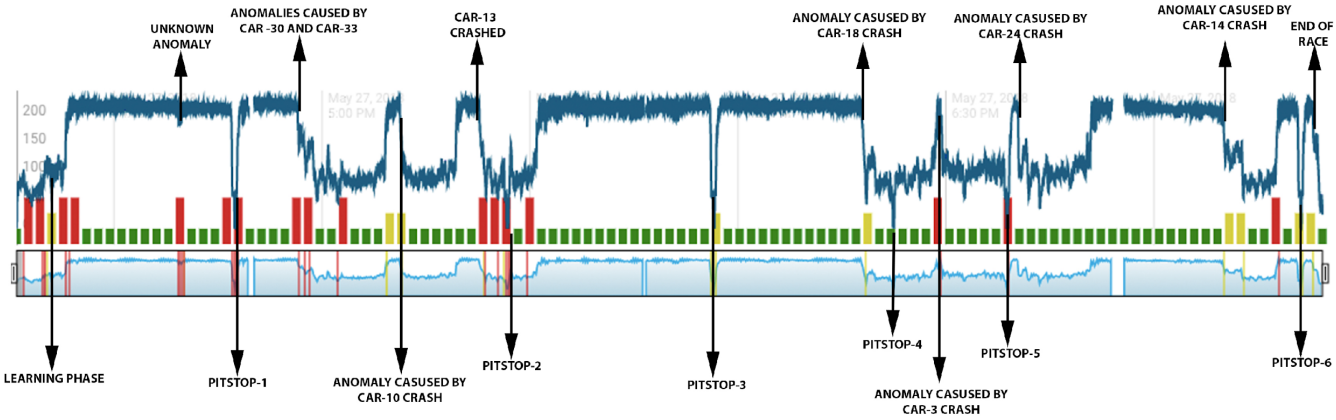


Figure 7: Data Labeling

anomalies as accurately as possible. The types of anomaly events include pit stop, car crash, flag, and other unknown events. All of these exception events we record their start and end times and do not distinguish between anomaly types.

4.3 Discover Rate and False Positive Rate

In pattern recognition, information retrieval and binary classification, precision is the fraction of relevant instances among the retrieved instances. However, in a time series, in fact, every time point can be regarded as a sample, so that we have almost infinite samples, making it difficult to define the precision rate and recall rate. In order to quantitatively analyze the performance of different algorithms, we define the detection rate and false positive rate as follows:

Suppose we manually labeled N anomaly intervals. If an algorithm detects at least 1 anomaly event in the i-th interval, we consider the i-th interval being 'discovered.' The discovery rate is defined as:

$$Discover\ Rate = \frac{intervals\ discovered}{N}$$

Suppose the algorithm finds a total of m anomaly events. Among them, p anomaly events are in a labeled anomaly interval, and f are not in an interval. Therefore, p/m is the true positive rate, and f/m is the false positive rate.

An ideal detection algorithm should have a high discovery rate and a low false positive rate.

Table 2: Comparison of anomaly detection algorithms

Detector	Discover Rate	False Positive Rate
Numenta HTM	68.2%	51.5%
Random Cut Forest	22.7%	96.5%
Bayesian Changepoint	90.9%	77.9%
EXPoSE	90.9%	45.9%

5 CONCLUSION

Real-time sports broadcast and anomaly detection on the IndyCar data stream provides an interesting and also challenging problem to resolve from the perspective view of both machine learning algorithm and distributed system. Through the previous experiments, we obtained: The random cut forest algorithm obtained the average distribution of anomaly scores, and the discover rate is the lowest among the four algorithms. It is not suitable for use on the IndyCar dataset.

The results obtained by the Bayesian Changepoint algorithm are very characteristic: it will mark all the sharp points on the graph. However, this algorithm cannot effectively remember the patterns that have appeared before, causing it to mark many false positive points.

Both HTM and EXPoSE have achieved good discover rate. Among them, EXPoSE will mark more anomaly points in one interval. Here we assume that the repeated detections are counted, so it gets higher discover rate.

6 ACKNOWLEDGEMENTS

We gratefully acknowledge support from professors, teaching assistants, students at the school of Informatics, Computing, and Engineering of Indiana University. We also appreciate the system support offered by FutureSystems.

REFERENCES

[1] IndyCar Series. https://en.wikipedia.org/wiki/IndyCar_Series/. [Online; accessed 1-Mar-2019].
[2] R. P. Adams and D. J. MacKay. Bayesian online changepoint detection. *arXiv preprint arXiv:0710.3742*, 2007.
[3] S. Ahmad and J. Hawkins. Properties of sparse distributed representations and their application to hierarchical temporal memory. *arXiv preprint arXiv:1503.07469*, 2015.
[4] S. Ahmad and J. Hawkins. How do neurons operate on sparse distributed representations? A mathematical theory of sparsity, neurons and active dendrites. *arXiv preprint arXiv:1601.00720*, 2016.
[5] S. Ahmad, A. Lavin, S. Purdy, and Z. Agha. Unsupervised real-time anomaly detection for streaming data. *Neurocomputing*, 262:134–147, Nov. 2017.
[6] R. A. Ariyaluran Habeeb, F. Nasaruddin, A. Gani, I. A. Targio Hashem, E. Ahmed, and M. Imran. Real-time big data processing for anomaly detection: A Survey. *International Journal of Information Management*, Sept. 2018.
[7] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):15, 2009.

[8] C. Chatfield. The Holt-winters forecasting procedure. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 27(3):264–279, 1978.

[9] A. B. Csapó, P. Baranyi, and D. Tikk. Object categorization using vfa-generated nodemaps and hierarchical temporal memories. In *2007 IEEE International Conference on Computational Cybernetics*, pages 257–262. IEEE, 2007.

[10] J. v. Doremalen and L. Boves. Spoken digit recognition using a hierarchical temporal memory. In *Ninth Annual Conference of the International Speech Communication Association*, 2008.

[11] N. Farahmand, M. H. Dezfoulian, H. GhiasiRad, A. Mokhtari, and A. Nouri. Online temporal pattern learning. In *2009 International Joint Conference on Neural Networks*, pages 797–802. IEEE, 2009.

[12] D. George and R. G. Jaros. Feedback in group based hierarchical temporal memory system, July 19 2011. US Patent 7,983,998.

[13] S. Guha, N. Mishra, G. Roy, and O. Schrijvers. Robust random cut forest based anomaly detection on streams. In *International conference on machine learning*, pages 2712–2721, 2016.

[14] J. Hawkins and S. Ahmad. Why neurons have thousands of synapses, a theory of sequence memory in neocortex. *Frontiers in neural circuits*, 10:23, 2016.

[15] R. G. Jaros, D. George, J. C. Hawkins, and F. E. Astier. Sequence learning in a hierarchical temporal memory based system, Oct. 9 2012. US Patent 8,285,667.

[16] A. Lavin and S. Ahmad. Evaluating real-time anomaly detection algorithms–the numenta anomaly benchmark. In *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, pages 38–44. IEEE, 2015.

[17] W. J. Melis, S. Chizuwa, and M. Kameyama. Evaluation of the hierarchical temporal memory as soft computing platform and its vlsi architecture. In *2009 39th International Symposium on Multiple-Valued Logic*, pages 233–238. IEEE, 2009.

[18] I. Ramli and C. Ortega-Sanchez. Pattern recognition using hierarchical concatenation. In *2015 International Conference on Computer, Control, Informatics and its Applications (IC3INA)*, pages 109–113. IEEE, 2015.

[19] B. Rosner. Percentage points for a generalized ESD many-outlier procedure. *Technometrics*, 25(2):165–172, 1983.

[20] D. Rozado, F. B. Rodriguez, and P. Varona. Gaze gesture recognition with hierarchical temporal memory networks. In *International Work-Conference on Artificial Neural Networks*, pages 1–8. Springer, 2011.

[21] M. Schneider, W. Ertel, and F. Ramos. Expected similarity estimation for large-scale batch and streaming anomaly detection. *Machine Learning*, 105(3):305–333, 2016.