Mining and Storing Data Streams for Reliability Analysis

Article · January 2010

CITATIONS
2

READS
46

4 authors, including:

Taghi Khoshgoftaar
Florida Atlantic University
395 PUBLICATIONS **7,179** CITATIONS
SEE PROFILE

Ankur Agarwal
Florida Atlantic University
70 PUBLICATIONS **271** CITATIONS
SEE PROFILE

Mining and Storing Data Streams for Reliability Analysis

Janell Duhaney, Taghi M. Khoshgoftaar, Ankur Agarwal, John C. Sloan
Florida Atlantic University, Boca Raton, Florida, USA
Contact author: khoshgof@fau.edu

Keywords: reliability analysis, data stream mining, databases

***Abstract*—Sensor networks, in addition to stock tickers, network event logs, scientific simulations, credit card transactional flows, and surveillance video cameras, output a continuous flow of data, called a *data stream*. Mining data from a network of sensors provides insight into the health and reliability of a system. This process is known as data stream mining. This paper examines the issues associated with mining these sensor data streams and the challenges related to storing some or all of this data. This paper will identify opportunities for future work in mining and storage of data streams.**

I. Introduction

Reliability analysis is the study of the ability of a component or system to operate under given conditions for a specific period of time according to some performance requirement. It involves determining the possible points of failure within a system and identifying those components which contribute most towards the system's unreliability. The field of system reliability encompasses multiple key research areas including software reliability (the measure of the quality of a software design), reliability prediction (the forecast of the failure rate of a system) and failure analysis (the determination of the cause and consequence of a system failure). The failure rate or failure ratio, according to [1], is the proportion of the number of failures to a given unit of measure, such as failures per unit of time, failures per number of transactions, or failures per number of revolutions. The use of the term "reliability analysis" henceforth in this paper will pertain only to failure analysis and/or reliability prediction.

Automated reliability analysis has been made possible by the availability of cost effective sensor technology and increased computing performance. Sensors produce a continuous feed of measurements, called a data stream, which can be analyzed to determine the state of the component being measured and the presence of faults. The process of inferring knowledge from a continuous stream of data is known as data stream mining.

Over the years, data stream mining has been the focus of many research papers and is used in many domains including fraud detection [2], medical monitoring [3] and stock market analysis [4]. This paper examines the challenges associated with mining and storing data streams and surveys research efforts which address some of these issues. By exploring these issues and possible solutions, we hope to highlight additional research opportunities in this area.

This paper is arranged as follows. Section II will provide a brief background into data streams for reliability analysis. Issues in data stream mining and data stream management will be discussed in sections III-A and III-B respectively. Finally, concluding remarks are made in Section IV.

II. Background

Data stream mining techniques can be used in several ways in reliability analysis, such as in a Machine Condition Monitoring (MCM) system to determine the state and the reliability of the machine being monitored. MCM systems enable real time health assessment, prognostics, and advisory generation by continuously recording and processing streams of measurements taken from sensors attached to components of the machine being monitored [5]. These systems have been developed to mitigate maintenance costs and improve the lifetime and reliability of equipment. To measure various aspects of the system, different types of sensors such as accelerometers for measuring vibration data, tachometers for measuring rotational velocity, oil sensors for measuring oil level and quality, pressure sensors, and temperature sensors are often used. These sensors produce a stream of measurements — each at their own pace — which can be mined to determine the existence of faults, the cause of faults, and the probability of failure.

Another way in which data stream mining is used in reliability analysis is in the mining of changes in the sensor data. Mining changes in sensor data may provide useful information regarding the state of that sensor and reduce conflicts in knowledge fusion systems. Knowledge fusion systems attempt to integrate data from multiple sensors to deduce the state of a component in a system or of the entire system itself.

Another important aspect of reliability analysis for which data stream mining has proved itself useful is survivability. The survivability of a system is its ability to operate efficiently in the presence of catastrophic failures [6]. Survival analysis is used to represent the time to an event, such as the failure time of a mechanical system.

Data stream mining encompasses techniques which utilize information gathered from past cases to label incoming data for prediction or classification of faults, and those which identify faults based on their similarity to a previous case using clustering. In the following section, we review issues

plaguing data stream mining and management as it relates to reliability analysis.

III. Related Work

A. Data Stream Mining

1) Concept drifts and model updates: Knowing when to update a mining model is one of the challenges in data stream mining. A mining model could be updated periodically, incrementally (with every change in data) or reactively (rebuild the model only when it no longer suits the data) [7]. Frequent updates of the mining model wastes resources on insignificant changes while infrequent updates risk model inaccuracy and the resulting system degradation [7].

To complicate this issue, the underlying concepts in a data streams have a tendency to change over time. Mining concept drifting data streams has been the focus of many articles, including [8], [9], [10], and [11]. Algorithms adapted to concept drifting streams include incremental decision trees such as the concept-adapting very fast decision tree [9] and ensemble classifiers such as the streaming ensemble algorithm (SEA) [9].

Adjusting the mining model to adapt to changes in the state of the system being monitored must also be considered. Inaccessible machines, for example, may have to operate with non-critical faults until a scheduled maintenance date when the system state would revert to normal. If the classifier was trained based on data generated while the system was in an abnormal state, a recurring fault introduced after the system has been returned to normal may not be correctly classified by the mining model since the classifier may now consider the existence of that fault as a part of the normal operation of the system.

2) Selecting an appropriate mining method: In general, data stream mining approaches generate a model from historical records and use this model to classify new instances in the data stream. Such a method permits only a single pass through the training data since a second scan of the data is infeasible due to the high rate of incoming data, and will need to incorporate new data from the stream as appropriate. Some data mining techniques such as decision trees do not adapt well to handle continuous data [12], but others such as ensemble classifiers ([13], [14]), one-versus-all classifiers [9], and k-means clustering [15] have been applied successfully to data stream mining.

The possibility of redundancies within ensemble classifiers and the inability of single components of incremental classifiers to be updated independently when a concept has changed are challenges which must be addressed. Redundancies within ensemble classifiers can be avoided using pruning techniques such as instance based pruning [13] to identify the subset of classifiers that produce the same results as the entire ensemble. The challenge of determining which classifier within an ensemble is to be updated is discussed in [9] but remains an open issue.

3) Model overfitting: Model overfitting, which occurs when a mining model is too specific or is too sensitive to the training dataset that was used to generate that model, is more likely to occur in streaming environments. This may be because of a lack of training data and possible biases in the training dataset resulting from the data originating from a single source [8]. Traditional data mining techniques such as cross validation are not well suited for data streams because they require more than a single pass over the training data. The framework proposed by Wang et. al in [8] addresses the overfitting problem by harnessing concept drifting patterns. Efficient feature selection algorithms such as [16] will also reduce the risk of overfitting.

4) Cost sensitive learning: In reliability analysis, the cost of a false positive (or false alarm) is typically not equal to the cost of a false negative. Inaccessible systems, for example, have a low tolerance to false alarms because of the expenses associated with retrieving the equipment for repair. One approach to cost sensitive learning from data streams involved weighting classifiers within an ensemble based on their mean square error [13].

5) Imbalanced datasets in classification problems: In reliability analysis, it is typical for the dataset to be imbalanced, meaning that the ratio of positive (faulty) instances to negative (no fault) instances is skewed in favor of the negative instances. Learning algorithms such as decision trees have been known to perform poorly on imbalanced dataset problems because of their tendency to classify all instances as negative to maximize accuracy. [9] investigated the use of a novel under-sampling scheme for their ensemble classifier which restricts each negative instance in the training set from being used in the models of more than two classifiers within the ensemble. This scheme, along with techniques such as [17] and [18], could address this issue.

6) Missing or incomplete data: Missing or incomplete data is not uncommon in sensor networks, so any stream mining algorithm or framework should make provisions for handling missing, delayed or out-of-sequence data. [19] proposed a framework based on a compressed sensing (CS) theory [20] for detecting anomalies in incomplete data by either sampling a subset of the sensors or of the number of frames in a temporal stream. Three techniques for substituting values for missing data points, namely Bayesian multiple imputation, *k* Nearest Neighbour imputation and Mean imputation, were investigated by [21] as solutions to the missing data problem.

7) Modeling changes in mining results: Observing temporal changes in data streams may provide useful information about the system. By sensing the fluctuation in the data stream from a particular sensor, for example, it may be possible to identify that a sensor is failing based on increasing variances as time progresses. Algorithms such as *MAIDS* [22] have been designed for this purpose.

8) Data preprocessing: The design of a lightweight preprocessing technique which can guarantee the quality of the mining results remains as an open problem in data stream mining [23].

9) Formalizing data stream computing: Formalizing data stream mining within a theory of stream computation enables the design and development of mathematically sound stream

mining algorithms [24]. A formalization of data streams in signal processing traditionally [25] uses Z-transforms — a discrete version of Laplace transforms. An approach geared to the composition, or the gluing together, of software components was more recently proposed in [26]. Using coinductive stream calculus to define signal flow graphs, this approach may make integration of learners into a stream processing environment easier to implement.

B. Data Stream Management

1) Centralized vs. decentralized processing: Processing live data may be either centralized or distributed (i.e. permitting some computation to take place at the individual sensor site). With centralized processing, the amount of bandwidth required to transmit all the data continuously (and in real time, in most cases) to a central storage system for immediate querying must be considered. With a distributed approach, each sensor node has the capability of processing its own data and then transmitting the results to a centralized location for further analysis. In distributed sensor data management, it is important to consider the possibility of data loss during transmission as well as how processing and storing its own data will affect the power consumption at a node. These concerns can create a bottleneck for the performance of the entire system.

2) Data aging: Another challenge to data stream management is the timely identification of stale data. An optimal data aging strategy would need to determine which data in the training set is no longer relevant, such as those pertaining to previous concepts in a concept drifting stream. As explained in [13], discarding data after a predetermined length of time may lead to some interesting problems. If the selected lifespan L is large, then the possibility of having outdated instances within the training dataset is high; if L is small, the risk of overfitting is greater since there may be insufficient instances in the training dataset. The aging strategy presented in [13] provides a workaround by considering the class distribution in addition to the arrival time when selecting the data lifespan.

3) Storage space restrictions: The continuous flow of a data stream demands an unlimited storage system to hold unprocessed streaming data and/or the results of the mining operation. In most cases, there is minimal storage at sensor nodes, so data storage may be centralized. Novel stream management systems (e.g. StonesDB [27], DIMENSIONS [28], TinyDB, Cougar and Diffusion) were designed to provide efficient storage solutions for data streams.

4) Limited availability of resources: One of the biggest stream management issues relates to the lack of available resources including storage space, processing power, network connectivity, and energy. In some domains, including ocean systems, available network connectivity may be limited, spotty, and incapable of sustaining high data transfer rates. More reliable network connections tend to be difficult to acquire or expensive, and may not be feasible offshore in the case of oceanic systems. An ideal solution would therefore need to minimize communication overhead.

Because of the constant transmission of data, the lifetime of the battery in a battery-operated sensor is greatly reduced. A novel approach to resolving this issue in an ocean turbine is to wire the sensors in to the turbine's battery pack which can be recharged by the flow of ocean currents. Other solutions attempt to conserve energy consumption by reducing the quantity of data to be transferred or by transmitting the data in batches. By reducing the volume of data, the amount of time the sensor spends transmitting data is also decreased. The volume of a data stream can be reduced via well known summarization methods: aggregation, sampling, load shedding and approximation.

Sampling techniques in general involve altering the number of data points to achieve desired results, and are typically used in class imbalance problems to alter the class distributions. Undersampling techniques, which involve selecting a subset of the negative (not faulty) instances from the original data stream, reduces the size of the data stream by ignoring or discarding some of the original data points. In load shedding, a sequence of items within the data stream are discarded. While it has been applied successfully to data streams, it presents the same problems as sampling [23]. Aggregation employs statistical measures such as average, maximum, minimum, etc. while approximation techniques ([29], [30], [31]) involve replacing the original data stream with an approximated signal (with error bounds) which is tailored for the application domain [30].

5) Handling the continuous flow of data streams: Passive systems such as traditional database systems typically have high latencies due to the cost of a database storage operation and constant polling for data [4]. Active stream processing engines (SPE) which utilize specialized primitives and constructs like time sliding windows to perform on-the-fly processing of streaming data but store data only when necessary are popular solutions to this problem. The querying language used within an SPE typically resembles SQL for databases, and often includes constructs permitting joins and cross querying of multiple streams. StreamSQL is one of the most popular variants of SQL and was specifically designed to express queries on continuous data streams.

Another issue pertaining to the continuous flow of data streams is the inconsistency in data transfer rates. Algorithm output granularity (AOG) is a resource-aware data analysis approach capable of handling very high data rates which performs data analysis locally on a resource constrained device. AOG works by performing mining operations, followed by adapting to the data stream rates and resource availability, and then by merging knowledge structures when the memory is filled past a certain capacity [23].

6) Merging distributed streams: Typical monitoring systems consist of sensors installed in different locations on the machine or system being monitored. The resulting data streams must be merged, sometimes in real-time, to allow an overall system analysis. Associated with merging of data streams is the topic of data fusion which is surveyed in these proceedings [32]. [33] discusses a stream merging strategy

based on a common key.

IV. Conclusion

Data stream mining refers to the techniques for inferring knowledge from a continuous stream of data. This paper reviewed fundamental issues and challenges plaguing data stream mining and management for reliability analysis, as well as research efforts invested in addressing these problems. While a lot of work has been done in this area, there is room for additional research as some of the issues mentioned in this paper are yet to be addressed. These open issues were: the need for lightweight data stream preprocessing techniques, the lack of an approach to determine which single component within an ensemble classifier is to be updated, and the problem of efficiently readjusting the data mining model within a system whose state has been returned to normal after operating in the presence of a fault for some time.

Acknowledgements

The work discussed here grew from collaborations within the Prognostics and Health Monitoring (PHM) working group of the Center for Ocean Energy Technology (COET) at Florida Atlantic University and was funded through COET by the State of Florida.

References

[1] *IEEE Standard Glossary of Software Engineering Terminology*. USA, December 1990. IEEE Std 610.12-1990; standard glossary; software engineering terminology;.

[2] T. M. Nguyen, J. Schiefer, and A. M. Tjoa, "Sense & response service architecture (saresa): an approach towards a real-time business intelligence solution and its use for a fraud detection application," in *DOLAP '05: Proceedings of the 8th ACM international workshop on Data warehousing and OLAP*, (New York, NY, USA), ACM, 2005, pp. 77–86.

[3] A. D. Jurik and A. C. Weaver, "Control, analysis and visualization of body sensor streams," 2009.

[4] M. Stonebraker, U. Çetintemel, and S. Zdonik, "The 8 requirements of real-time stream processing," *SIGMOD Rec.*, vol. 34, no. 4, 2005, pp. 42–47.

[5] P.-P. Beaujean, T. M. Khoshgoftaar, J. C. Sloan, N. Xiros, and D. Vendittis, "Monitoring ocean turbines: a reliability assessment," in *Proceedings of the 15th ISSAT International Reliability and Quality in Design Conference*, 2009, pp. 367–371.

[6] Z. Ma and A. Survival, "Survival analysis approach to reliability, survivability and prognostics and health management (phm)," in *Aerospace Conference, 2008 IEEE*, march 2008, pp. 1 –20.

[7] R. Wolff, K. Bhaduri, and H. Kargupta, "A generic local algorithm for mining data streams in large distributed systems," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 21, april 2009, pp. 465 –478.

[8] H. Wang, J. Yin, J. Pei, P. S. Yu, and J. X. Yu, "Suppressing model overfitting in mining concept-drifting data streams," in *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, (New York, NY, USA), ACM, 2006, pp. 736–741.

[9] S. Hashemi, Y. Yang, Z. Mirzamomen, and M. Kangavari, "Adapted one-versus-all decision trees for data stream classification," *IEEE Trans. on Knowl. and Data Eng.*, vol. 21, no. 5, 2009, pp. 624–637.

[10] W. Fan, "Systematic data selection to mine concept-drifting data streams," *Proceedings of the SIGKDD04*, 2004, pp. 128–137.

[11] M. Last, "Online classification of nonstationary data streams," *Intelligent Data Analysis*, vol. 6, no. 2, 2002, pp. 129–147.

[12] R. Bhowmik, "Data mining techniques in fraud detection," *Journal of Digital Forensics, Security and Law*, vol. 3, no. 2, 2008.

[13] H. Wang, W. Fan, P. S. Yu, and J. Han, "Mining concept-drifting data streams using ensemble classifiers," in *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, (New York, NY, USA), ACM, 2003, pp. 226–235.

[14] X. Li, P. S. Yu, B. Liu, and S.-K. Ng, "Positive unlabeled learning for data stream classification," in *SDM*, SIAM, 2009, pp. 257–268.

[15] M. E. Orlowska, X. Sun, and X. Li, "Can exclusive clustering on streaming data be achieved?," *SIGKDD Explor. Newsl.*, vol. 8, no. 2, 2006, pp. 102–108.

[16] T. Khoshgoftaar, L. Bullard, and K. Gao, "Attribute selection using rough sets," *International Journal of Reliability, Quality, and*, vol. 16, no. 1, 2009, pp. 73–89.

[17] J. Van Hulse, T. Khoshgoftaar, and N. A., "Experimental perspectives on learning from imbalanced data," *Proceedings of the 24th International Conference on Machine Learning - ICML 2007, Corvallis, OR*, Jun 2007.

[18] X. Su, T. Khoshgoftaar, and R. Greiner, "Making an accurate classifier ensemble by voting on classifications from imputed learning sets," *International Journal of Information and Decision Sciences*, vol. 1, no. 3, 2009, pp. 301–322.

[19] S. Budhaditya, D.-S. Pham, M. Lazarescu, and S. Venkatesh, "Effective anomaly detection in sensor networks data streams," in *Data Mining, 2009. ICDM '09. Ninth IEEE International Conference on*, December 2009, pp. 722 –727.

[20] D. Donoho, "Compressed sensing," 2004.

[21] T. Khoshgoftaar and J. Van Hulse, "Imputation techniques for multivariate missingness in software measurement data," *Software Quality Journal*, vol. 16, Dec 2008, pp. 563–600.

[22] Y. D. Cai, D. Clutter, G. Pape, J. Jan, M. Welge, and L. Auvil, "Maids: Mining alarming incidents from data streams," *Proceedings of the 23rd ACM SIGMOD Intl Conf on Management of Data*, June 2004, pp. 13–18. Paris, France.

[23] M. M. Gaber, A. Zaslavsky, and S. Krishnaswamy, "Mining data streams: a review," *SIGMOD Rec.*, vol. 34, no. 2, 2005, pp. 18–26.

[24] M. M. Gaber, S. Krishnaswamy, and A. Zaslavsky, "Ubiquitous data stream mining," 2004.

[25] H. Freeman and O. Lowenschuss, "Bibliography of sampled-data control systems and z-transform applications," *IRE Transactions on Automatic Control*, vol. 4, Mar 1958, pp. 28 – 30.

[26] J. Rutten, "A tutorial on coinductive stream calculus and signal flow graphs," *Theoretical Computer Science*, vol. 343, no. 3, 2005, pp. 443 – 481. Formal Methods for Components and Objects.

[27] Y. Diao, D. Ganesan, G. Mathur, and P. J. Shenoy, "Rethinking data management for storage-centric sensor networks," in *CIDR*, www.crdrdb.org, 2007, pp. 22–31.

[28] D. Ganesan, B. Greenstein, D. Perelyubskiy, D. Estrin, and J. Heidemann, "An evaluation of multi-resolution storage for sensor networks," in *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, (New York, NY, USA), ACM, 2003, pp. 89–102.

[29] G. Cormode and S. Muthukrishnan, "Summarizing and mining skewed data streams," in *SDM*, 2005.

[30] A. Deligiannakis, Y. Kotidis, and N. Roussopoulos, "Compressing historical information in sensor networks," in *SIGMOD '04: Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, (New York, NY, USA), ACM, 2004, pp. 527–538.

[31] C. Olston, J. Jiang, and J. Widom, "Adaptive filters for continuous queries over distributed data streams," in *SIGMOD '03: Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, (New York, NY, USA), ACM, 2003, pp. 563–574.

[32] J. Duhaney and et. al, "Applications of data fusion in monitoring inaccessible ocean machinery," in *Proceedings of the 16th ISSAT International Reliability and Quality in Design Conference*, 2010.

[33] M. Mazzucco, A. Ananthanarayan, R. L. Grossman, J. Levera, and G. B. Rao, "Merging multiple data streams on common keys over high performance networks," in *Supercomputing '02: Proceedings of the 2002 ACM/IEEE conference on Supercomputing*, (Los Alamitos, CA, USA), IEEE Computer Society Press, 2002, pp. 1–12.