SPICE: Streaming PCA Fault Identification and Classification Engine in Predictive Maintenance

Cristian Axenie(✉), Radu Tudoran, Stefano Bortoli, Mohamad Al Hajj Hassan, Alexander Wieder, and Goetz Brasche

Huawei German Research Center, Riesstrasse 25, 80992 Munich, Germany
{cristian.axenie,radu.tudoran,stefano.bortoli,mohamad.alhajjhassan, alexander.wieder,goetz.brasche}@huawei.com

Abstract. Data-driven predictive maintenance needs to understand high-dimensional "in-motion" data, for which fundamental machine learning tools, such as Principal Component Analysis (PCA), require computation-efficient algorithms that operate near-real-time. Despite the different streaming PCA flavors, there is no algorithm that precisely recovers the principal components as the batch PCA algorithm does, while maintaining low-latency and high-throughput processing. This work introduces a novel processing framework, employing temporal accumulate/retract learning for streaming PCA. The framework is instantiated with several competitive PCA algorithms with proven theoretical advantages. We benchmark the framework in a real-world predictive maintenance scenario (i.e. fault classification in a coal coke production line) and prove its low-latency (millisecond level) and high-throughput (thousands events/second) processing guarantees.

Keywords: Stream processing · PCA · Online learning · Predictive maintenance · Fault identification · Fault classification

1 Introduction

Given today's industrial IoT sensory streams, predictive maintenance systems need to process streams of data under tight computational constraints for decision making (e.g. fault / normal operation). Processing data streams is quite different from querying static data, as data might be transient and follow a non-stationary distribution. These complications impose significant constraints on the problem of streaming PCA, which is an essential building block for many inference and decision making tasks. Due to its practical relevance, there is renewed interest in this problem [17]. PCA is good at maintaining data structure in reduced subspaces in an unsupervised way. It is also useful in updating the decision boundaries by adding discriminately informative features with newly added samples through updating the feature vectors by incremental eigenvector estimates [8]. This is highly relevant in multi-class classification problems, such

P. Cellier and K. Driessens (Eds.): ECML PKDD 2019 Workshops, CCIS 1168, pp. 333–344, 2020.
https://doi.org/10.1007/978-3-030-43887-6_27

as fault identification in predictive maintenance [6]. However, there is no algorithm that recovers the principal components in the same precision regime as the batch PCA algorithm does, employing low-latency high-throughput processing. The core contribution of the paper resides in exploiting a novel streaming learning paradigm, termed accumulate-retract learning [2], that leverage state-of-the-art PCA algorithms to achieve low-latency high-throughput processing, in real-world predictive maintenance fault identification and classification.

2 Related Work

Recent focus in predictive maintenance is on understanding streaming high-dimensional data, where the dimensionality of the data can potentially scale together with the number of available sample points [12,19]. Current trends in predictive maintenance have led to an exploration of the complexity of covariance estimation underlying PCA [3]. Such algorithms have provable complexity guarantees [7], but either store all samples (i.e. for looping through samples) or explicitly maintain the covariance matrix. In high-dimensional applications, such as sensory-rich production lines, storing all data is prohibitive. Different from previous approaches, our work brings the focus on two critical quantities: latency and throughput. In the streaming, data-driven, setting many approaches for incremental or online PCA have been developed, some focusing on replacing the inefficient steps in the traditional PCA algorithm [4]. Despite the multitude of successful dedicated algorithms, such as [18], there is no algorithm that brings the focus on the two critical quantities of focus in streaming predictive maintenance. Our work utilizes a new paradigm for stream processing and learning and proposes the implementation of several competitive streaming PCA algorithms for efficient computation with low-latency and high-throughput. Our accumulate/retract learning framework [2] offers a solution for incrementally computing combinations of statistics and learning, as needed in fault identification. The proposed framework is among the first attempts towards this paradigm shift initially set by Massive Online Analytics (MOA) and algorithms like Adaptive Windowing (ADWIN) [1]. Despite the fact that ADWIN implementations have theoretical guarantees, they do not guarantee latency and throughput, the goal of our system. Our system finds a trade-off between these two guarantees to be able to perform inference (i.e. fault identification) with low-latency and high-throughput. Similar to ADWIN, a main advantage of our approach is that it does not require any prior about how fast or how often the stream will drift, as it continuously estimates that while updating the models.

3 Methods

This section covers the design, implementation details, and the motivation to tackle the inherent problems in traditional PCA impeding it to achieve low-latency and high-throughput processing. We identified three aspects in existing approaches which impact the streaming PCA formulation: (a) the continuous

calculation of the mean and other descriptive statistics (i.e. covariance) on the datastream; (b) sorting the dominant eigenvalues in the rank update of the QR decomposition and the ordering of lower/upper triangular sub-matrices; (c) the complexity of computations performed at each training step. Dissecting the theory of PCA, we found those bottlenecks that kept PCA away from streaming applications. Employing a novel method and system for online machine learning [2], capable of incrementally computing machine learning models on data streams, we successfully instantiated multiple streaming PCA approaches in a predictive maintenance scenario (i.e. fault identification in coal coke production) and prove processing performance (i.e. low-latency, high-throughput). The underlying computational mechanism in our approach is the accumulate-retract framework. Such a framework allows for dual model updates as soon as new data comes into the system and leaves the system, respectively, as the stream progresses. The technique builds on top of the sliding window paradigm, and provides a novel, incremental computation model for closed-form learning rules used in streaming PCA. For example, the average calculation in the eigenvalue update, in incremental form, can be visualized in Fig. 1. Such incremental computation tackles successfully the first bottleneck, namely the incremental calculation of the mean and other descriptive statistics on the datastream. As shown in Fig. 1, the average computation is split in dual operations that keep subquantities in the update of the mean consistent as the stream progresses (i.e. window slide). Going further, the second problem that the accumulate-retract framework solves, is sorting the dominant eigenvalues in the rank update of the QR decomposition. For this, let's assume we need to sort the current list of dominant eigenvalues, as shown in Fig. 2. In this case, the caches (i.e. fixed memory areas) are used to store content (i.e. in buckets) on updates depending on counts (i.e. histogram). Updates are done in buckets, which contain sorted eigenvalues - which in follow a histogram sorting. Each time new eigenvalues are computed sorting is triggered (Step 1). In such an instantiation the retraction cache stores the last calculated eigenvalues (in time) in each bucket, whereas when the accumulation cache moves according to the sliding convention, new buckets are brought and the entire structure is sorted (Steps 2, 3, 4). Moreover, in this instantiation, the buckets stored on disk or 3rd party storage devices contain data organized based on value/indexes. The last eigenvalue (time-wise) in each bucket has a reference in the retraction cache, as shown in Fig. 2. There are many models for learning PCA efficiently: from stochastic approximation models (i.e. Hebbian and Oja's Learning Rules [9]), to subspace learning, and nonlinear PCA denoising autoencoders [13]. Yet, in order to explore the potential that the accumulate/retract learning framework offers, we selected three Streaming PCA algorithms, which employ only local learning rules and have proven theoretical advantages. Their dual, accumulate/retract implementation is one novel aspect of this work, enabling such theoretical model to perform in real-world predictive maintenance scenarios.

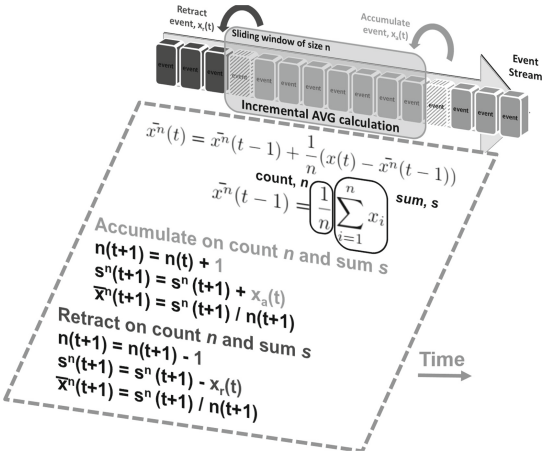

Fig. 1. Streaming PCA incremental average update using accumulate/retract.

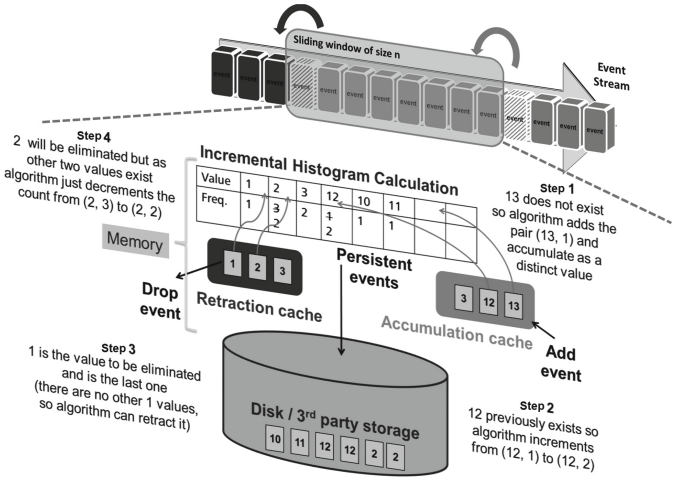

Fig. 2. Streaming PCA eigenvalues sorting on streams: histogram update using accumulate/retract.

4 Materials

The first model considered in our experiments is a single-layer neural network based on the Stochastic Gradient Ascent (SGA) [9]. We chose this model because it efficiently provides a description of the covariance matrix, which is typically too expensive to be estimated online. The accumulate/retract learning maps easily on the structure of the network and the corresponding learning rule in

Eq. 1 due to its dual operation when forward propagating from the input $x(k)$ to the hidden layer $y(k)$ by updating the weight matrix W.

$$\Delta w_j(t-1) = \gamma(t)y_j(t)(x(t) - y_j(t)w_j(t-1) - 2\sum_{i<j} y_i(t)w_i(t-1)) \quad (1)$$

where $\gamma(t)$ is the learning rate. The second model employed in our experiments is the Generalized Hebbian Algorithm (GHA) [10]. The model uses Hebbian learning to achieve optimal unsupervised extraction of the eigenvectors of the autocorrelation matrix of the input distribution, given samples from that distribution. Each output of such a trained network represents the response to one eigenvector, whereas outputs are ordered by decreasing eigenvalues. Considering the same single-layer feed-forward neural network, the GHA-based Streaming PCA learning rule is given by:

$$\Delta w_{ij}(t) = \gamma(t)(y_j(t)x_i(t) - y_j(t)\sum_{k<i} w_{kj}(t)y_k(t)) \quad (2)$$

GHA requires only the computation of the outer products yx^T and yy^T so that if the number of outputs is small the computational and storage requirements can be correspondingly decreased.

Finally, the third Streaming PCA model employed in our experiments is the Candid Covariance-free Incremental PCA (CCIPCA) [14]. This Streaming PCA model is able to compute the principal components of a sequence of samples, incrementally, without estimating the covariance matrix. The method is based on the concept of statistical efficiency (i.e. the estimate has the smallest variance given the observed data). In order to achieve this, CCIPCA keeps the scale of observations and computes the mean of observations incrementally. Assuming that we have a $d - dimensional$ input stream $u(n), n = 1, 2, ...$ with covariance matrix A and given the definition, $\lambda x = Ax$, and $v = \lambda x$, then the learning rule for CCIPCA is given by:

$$v(n) = \frac{n-1-l}{n}v(n-1) + \frac{1+l}{n}u(n)u^T(n)\frac{v(n-1)}{||v(n-1)||} \quad (3)$$

where l is the "amnesic parameter" (i.e. a forgetting parameter). Independent of the learning rule the weights w_j converge to the eigenvectors c_i as the streaming PCA model finds the unique set of weights which is both optimal and gives uncorrelated outputs. This is also true for the $v(n)$, in 3, analog to w. In the accumulate/retract framework such an update is performed as new elements are available from the stream. The accumulate event (i.e. incoming) triggers an incremental update of the weight matrix, whereas the retraction event (i.e. outgoing) triggers a decremental update of the weight matrix. This ensures the weight matrix stays consistent as the stream progresses. For each of the considered models, we derived the closed form incremental learning rules in the accumulate-retract framework. As shown previously, the eigenvectors $z_1, z_2, z_3, ..., z_p$ are given by $Cz_i = \lambda_i z_i$, where λ_i are the eigenvalues of the covariance matrix, $C[c_{jk}]$.

We can then rewrite C as $c_{jk} = \frac{1}{n-1}\sum_{i=1}^{n}(z_{ij} - \bar{z_j})(z_{ik} - \bar{z_k})$ where $\bar{z} = \frac{1}{n}\sum_{i=1}^{n} z_i$ is the incremental average. Hence, we can rewrite the covariance matrix C as $C = \frac{1}{n-1}\sum_{i=1}^{n}(z_i - \bar{z})(z_i - \bar{z})^T$, which is, in fact, the autocorrelation. The problem is that these measures are not robust statistics and hence not resistant to outliers. We replace z with its estimate at time t $z(t)$ so that $v = \lambda z$ estimate at time t is $v(t) = \frac{1}{n}\sum_{i=1}^{n} x(t)x^T(t)z(t)$. We can now calculate the eigenvalues and eigenvectors given v as we know that $\lambda = \|v\|$. If we consider $z = \frac{v}{\|v\|}$ we can rewrite $v(t) = \frac{1}{n}\sum_{i=1}^{n} x(t)x^T(t)\frac{v(t-1)}{\|v(t-1)\|}$. Such computation steps provide a closed form learning rule in the accumulate-retract framework [2]. Independent of the considered streaming PCA model, the system converges from an initially random set of weights to the eigenvectors of the input autocorrelation in the eigenvalues order. The optimal weights are found by minimizing the linear reconstruction error $E\{(x - \hat{x})^2\}$ when the rows of W span the first p eigenvectors of C and the Linear Least Squares (LLS) estimate of x given y is $\hat{x} = CW^T(WCW^T)^{-1}y$. If the rows of W are the first eigenvectors then $WW^T = I$ and $C = W^T \Lambda W$ where Λ is the diagonal matrix of C in descending order. Then, $y = Wx$ is the Karhunen-Loève Transformation (KLT). In the LLS optimization routine, if we have an unknown function f the best estimator of y is $\min_f \sum_{i=1}^{n}(y_i - f(x_i))^2$. Moreover, if f is linear in x and $y = ax + b$ then the best estimator is the search for the best a, b that minimize $\min_f \sum_{i=1}^{n}(y_i - ax_i - b)^2$. In the accumulate-retract framework such a problem is incrementally solved as the datastream progresses, using simple updates shown in Fig. 3 Given that covariance can be incrementally calculated as $cov_{xy}(t) = \frac{n-2}{n-1}cov_{xy}(t-1) + \frac{1}{n}(x^n(t) - \bar{x}^{n-1}(t))(y^n(t) - \bar{y}^{n-1}(t))$, the problem in closed form assumes calculating incrementally a and b as $a(t) = \frac{cov_{xy}(t)}{m_2(t)}$, $b(t) = \bar{y}(t) - a\bar{x}(t)$ where $m_2(t)$ is the 2^{nd} statistical moment in incremental form. Yet, up to now we made the assumption that only y values contain errors while x are known

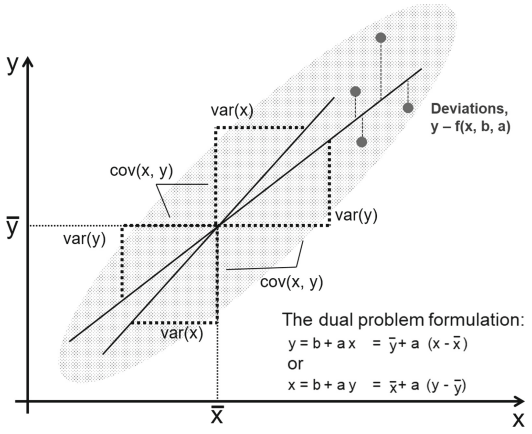

Fig. 3. Streaming PCA optimization using LLS in accumulate/retract framework.

accurately. This is not true in practical applications, thus we are seeking the values of a and b that minimize $\min_{a,b} \sum_{i=1} \frac{(y_i - ax_i - b)^2}{1+a^2}$, which amounts for the Total Least Squares (TLS). It has been shown that the TLS problem can be solved by performing a Minor Component Analysis (MCA) [16], thus finding the linear combinations (or directions) which contain the minimum variance. Since there might be errors in both x and y, we incorporate y into x and reformulate the problem by finding the direction d in $E_{TLS} = \min_d \frac{(dx+b)^2}{d^2}$ over all inputs x (including y). We can rewrite the same constraint as $E_{TLS} = n \min_d \frac{d^t Rd + 2bd^T E(x) + b^2}{d^T d}$ where $R = \frac{1}{n} \sum_{i=1}^{n} x_i x_i^T$ is the autocorrelation of the input and $E(x) = \frac{1}{n} \sum_{i=1}^{n} x_i$ is the average of the input. At convergence ($\frac{dE_{TLS}}{dz} = 0$) we must have $Rd + bE(x) - \theta d = 0$ where $\theta = \frac{d^T Rd + 2bd^T E(x) + b^2}{d^T d}$. We need to find a hyperplane $dx + b = 0$. Taking the expectation we have $C = -dE(x)$ which we can substitute in $Cd - \theta d = 0$ where now $\theta = \frac{d^T Cd}{d^T d}$ and $C = R - E(xx^T)$ is the covariance matrix. We can now see that every eigenvector is a solution of the minimization of E_{TLS}. Given the analytical walk-through of the methods, in the following section, we instantiate the considered streaming PCA models within the accumulate-retract framework for a multi-class classification problem. In such a problem, PCA acts as a preprocessing step and due to its incremental nature, preserves the discriminant information within the data and can provide classification boundaries [15].

5 Experiments and Discussion

This section introduces the results and the analysis of the three streaming PCA models instantiated in our framework using Apache Flink [5]. Flink is an open source system for parallel scalable processing on real-time streaming data. At its core, Flink builds on an optimized distributed dataflow runtime that supports our accumulate-retract framework, crucial in obtaining low-latency

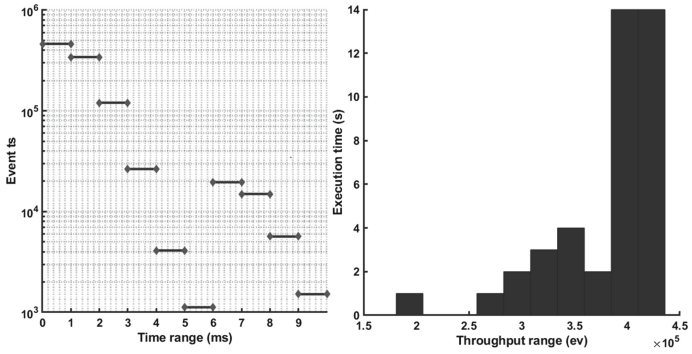

Fig. 4. Analysis of QR-based PCA without accumulate/retract (left latency, right throughput).

high-throughput online learning. The experimental setup for our tests used 4 machines, each with 24 CPU cores and 196 GB RAM, and Flink for cluster management. During the experiments we consider a fixed sliding window, but the system can support also adaptive windowing. At the same time the caches (i.e. in RAM) mechanism allows to maintain new and old data in order to allow the retraction of individual stream events when sliding. This allows our system to learn from continuous data in a single pass. We used a real-world stream with sensory readings from a coal coke prediction production line (i.e. data from 1 preheater temperature sensor, 2 briquetting temperature sensors, 2 cooker temperature sensors, 2 coke quencher temperature sensors, 2 coke transport system temperature sensors and 2 blast furnace temperature sensors). We addressed the problem of identifying faults in the production line and queried the eigenvalues and eigenvectors to extract the normal and faulty operation configuration prior to a multi-class classifier. The datastream contained 2M incoming events at 40 kHz. Moreover the datastream had the property that the eigenvalues of the input X are close to the class labels (i.e. $1, 2, ..., d$) and the corresponding eigenvectors are close to the canonical basis of R^d, where d is the number of principal components to extract and the class number for the multi-class classification task (i.e. various types of faults and normal operation - in our scenario, we consider 10 classes, 9 faults and 1 normal). Basically, the system: (a) supports the automatic generation of a concise, reliable, low-dimensional model which describes the operation mode of the various sensors in the coal coke production line; (b) identifies different alarm type conditions (i.e. the eigenvectors configuration, given the 10 different classes); and (c) conjectures the most likely cause of failure (i.e. the eigenvalues configuration). In order to evaluate the three streaming PCA models, we also implemented an efficient QR-based PCA of [11] using Householder transformation as ground-truth and ran it in the accumulate-retract framework on the same experimental setup. Important to note that the three streaming PCA models do not need to compute the correlation matrix in advance, since the eigenvectors are derived directly from the data. This is an important feature of streaming PCA, particularly if the number of inputs is large. The scope of our analysis is to emphasize that using simple incremental operations and exploiting an efficient data orchestration, the accumulate/retract framework can leverage low-latency high-throughput streaming PCA. Such a platform allows the three streaming PCA models to learn from datastreams in a single pass. In our evaluation, the latency measure refers to the single stream event processing time, whereas the throughput refers to the number of stream events processed in a second. In order to provide a baseline, we performed initial experiments with streaming PCA and the, ground-truth, QR-based PCA without the accumulate-retract framework. As one can see in both Fig. 4 and Fig. 5, respectively, without the accumulate-retract framework, the system can only process up to $\sim$21k events independent of the chosen model. The latency distribution, is centered on values $\sim$1 ms, as one can see in the left panel of both Figs. 4 and 5, respectively. There is an advantage that the streaming PCA holds in the overall event processing latency, with no event processed in over 8 ms. Important to mention that in

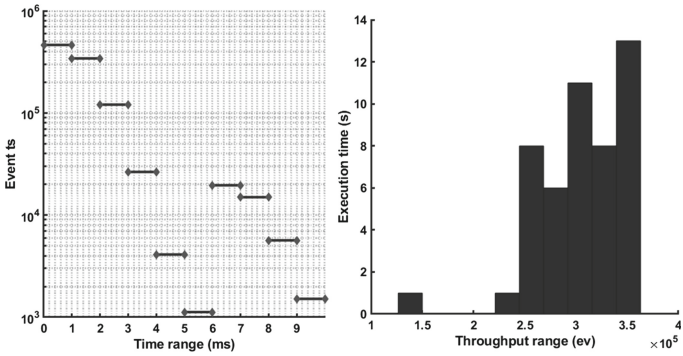

Fig. 5. Analysis of streaming PCA (SGA model) without accumulate/retract (left latency, right throughput).

both experiments the PCA model computed the eigenvectors and the eigenvalues simultaneously for each new incoming stream event. The central experiments of this paper are meant to emphasize the advantages of the accumulate-retract learning framework [2]. We now analyze a series of large-scale experiments meant at extracting the eigenfeatures for the multi-class fault identification scenario (i.e. 2M events streamed at 40 kHz, a coal coke production line in China). In terms of latency, one can observe that the streaming PCA models outperform the QR-based PCA model, with a substantial distribution of events processed at 1 ms and just a limited number of events processed at over 8 ms (less than 1000), as shown in Fig. 6. This is supported by the gain of ∼1k events throughput, as shown in Fig. 7 in the accumulate/retract framework and up to ∼10k more than in the baseline experiments. This is also visible in the core distribution of throughput ranges peaking at around 40k events/s (Table 1). To offer an understanding of the actual estimation performance of the system, the next table shows the eigenvalues of the input and how close they are to the class labels (i.e. $1, 2, ..., d = 10$) and the corresponding eigenvectors variance with respect to the canonical basis of R^d. This analysis didn't address a model comparison, rather an emphasis on the capabilities of the accumulate/retract learning framework to leverage streaming PCA models to reach guarantees of performance. Pushing real-world performance constraints, the accumulate-retract framework instantiation of various streaming PCA models stands out as good candidate for low-latency high-throughput systems for dimensionality reduction, in critical applications such as fault identification in predictive maintenance. The code repository for the Streaming PCA benchmarking is available at [1].

[1] https://github.com/omlstreaming/iotstream2019.

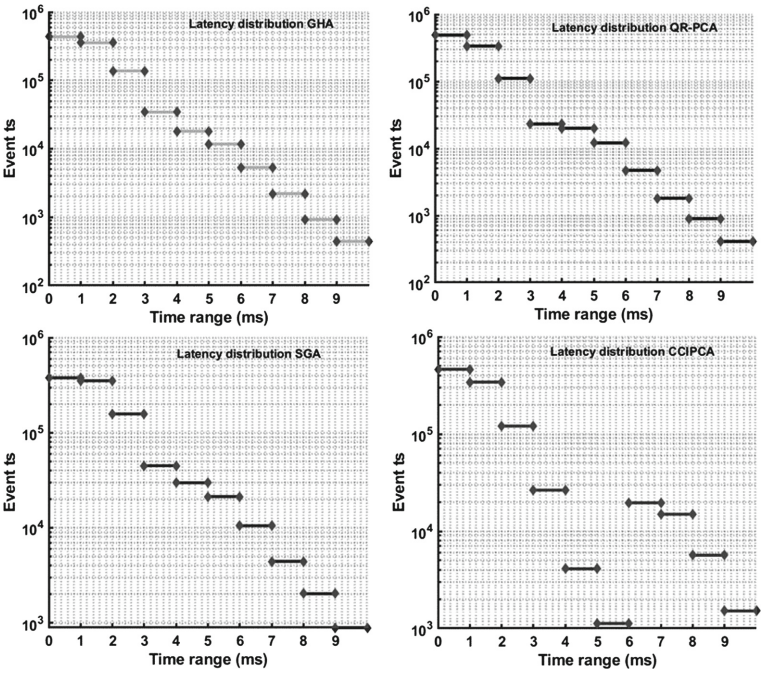

Fig. 6. Comparison of streaming PCA models latency in the accumulate/retract.

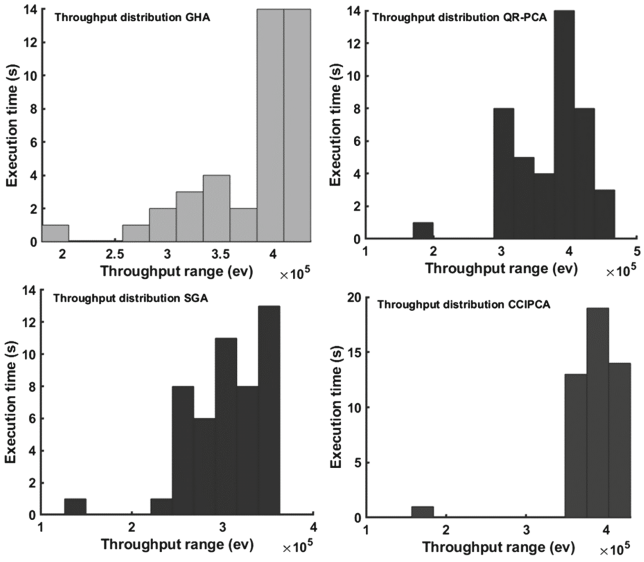

Fig. 7. Comparison of streaming PCA models throughput in the accumulate/retract.

Table 1. Eigenvalues and Eigenvectors estimates analysis

Eigenvalue	Eigenvalue estimate	Eigenvector variance
1	0.994071965	0.033719458
2	1.99658601	0.023661145
3	3.00600192	0.013884741
4	4.00420688	0.025106736
5	5.04173253	0.022354039
6	5.95475267	0.007637369
7	6.88985141	0.011129644
8	7.87972194	0.015864081
9	8.90795326	0.007244545
10	10.0642228	0.014663302

6 Conclusion

Tackling the theoretical bottlenecks in traditional PCA algorithms and focusing on two critical quantities, namely latency and throughput, the current work supports the renewed interest and performance improvements in streaming machine learning. As our experiments show, streaming PCA models can be leveraged by the accumulate/retract framework. Such a system offers flexibility, allowing for arbitrary combinations of multiple functions underlying complex machine learning models (i.e. average, least squares regression, histogram sorting) to be calculated on the stream, with no time- and resource-penalty, by exploiting the underlying hardware, data processing and data management for true low-latency, high-throughput stream processing. In our predictive maintenance benchmark scenario, our system could enable immediate alarming of conditions outside the normal mode of operation (i.e. the system will provide the class label corresponding to normal/fault operation). Failure identification would be based on the capability of the system to "fingerprint" potential failure branch types, based on the underlying eigenvectors/eigenvalues configurations. More experiments are planned to prove the competitiveness of the approach.

References

1. Bifet, A., Gavaldà, R., Holmes, G., Pfahringer, B.: Machine Learning for Data Streams with Practical Examples in MOA. MIT Press, Cambridge (2018)
2. Axenie, C., Tudoran, R., Bortoli, S., Hassan, M.A.H., Foroni, D., Brasche, G.: STARLORD: sliding window temporal accumulate-retract learning for online reasoning on datastreams. In: 17th IEEE International Conference on Machine Learning and Applications, ICMLA 2018, Orlando, FL, USA, 17–20 December 2018, pp. 1115–1122 (2018)

3. Baker, C.G., Gallivan, K.A., Van Dooren, P.: Low-rank incremental methods for computing dominant singular subspaces. Linear Algebra Appl. **436**(8), 2866–2888 (2012)
4. Boutsidis, C., Garber, D., Karnin, Z., Liberty, E.: Online principal components analysis. In: Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 887–901 (2015)
5. Carbone, P., Katsifodimos, A., Ewen, S., Markl, V., Haridi, S., Tzoumas, K.: Apache Flink^{TM}: stream and batch processing in a single engine. IEEE Data Eng. Bull. **38**, 28–38 (2015). https://flink.apache.org/introduction.html
6. Gajjar, S., Kulahci, M., Palazoglu, A.: Real-time fault detection and diagnosis using sparse principal component analysis. J. Process Control **67**, 112–128 (2018)
7. Hallgren, F., Northrop, P.: Incremental kernel PCA and the Nyström method. arXiv preprint arXiv:1802.00043 (2018)
8. Lois, B., Vaswani, N.: A correctness result for online robust PCA. In: 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 3791–3795. IEEE (2015)
9. Oja, E.: Principal components, minor components, and linear neural networks. Neural Netw. **5**(6), 927–935 (1992)
10. Sanger, T.D.: Optimal unsupervised learning in a single-layer linear feedforward neural network. Neural Netw. **2**(6), 459–473 (1989). http://www.sciencedirect.com/science/article/pii/0893608089900440
11. Sharma, A., Paliwal, K.K., Imoto, S., Miyano, S.: Principal component analysis using QR decomposition. Int. J. Mach. Learn. Cybern. **4**(6), 679–683 (2013). https://doi.org/10.1007/s13042-012-0131-7
12. Susto, G.A., Schirru, A., Pampuri, S., McLoone, S., Beghi, A.: Machine learning for predictive maintenance: a multiple classifier approach. IEEE Trans. Ind. Inform. **11**(3), 812–820 (2014)
13. Valpola, H.: From neural PCA to deep unsupervised learning. In: Bingham, E. (ed.) Advances in Independent Component Analysis and Learning Machines, pp. 143–171. Elsevier, Amsterdam (2015)
14. Weng, J., Zhang, Y., Hwang, W.S.: Candid covariance-free incremental principal component analysis. IEEE Trans. Pattern Anal. Mach. Intell. **25**(8), 1034–1040 (2003)
15. Woo, S., Lee, C.: Incremental feature extraction based on decision boundaries. Pattern Recogn. **77**, 65–74 (2018). http://www.sciencedirect.com/science/article/pii/S003132031730496X
16. Xu, L., Oja, E., Suen, C.Y.: Modified Hebbian learning for curve and surface fitting. Neural Netw. **5**(3), 441–457 (1992)
17. Zhan, J., Lois, B., Guo, H., Vaswani, N.: Online (and offline) robust PCA: novel algorithms and performance guarantees. In: Artificial Intelligence and Statistics, pp. 1488–1496 (2016)
18. Zhao, F., Rekik, I., Lee, S.W., Liu, J., Zhang, J., Shen, D.: Two-phase incremental kernel PCA for learning massive or online datasets. Complexity **2019**, 1–17 (2019)
19. Zheng, H., Wang, R., Yang, Y., Li, Y., Xu, M.: Intelligent fault identification based on multi-source domain generalization towards actual diagnosis scenario. IEEE Trans. Ind. Electron. **67**(2), 1293–1304 (2019)