Dimensionality Reduction for Low-latency High-throughput Fraud Detection on Datastreams

Cristian Axenie
Huawei German Research Center
Munich, Germany
cristian.axenie@huawei.com

Radu Tudoran
Huawei German Research Center
Munich, Germany
radu.tudoran@huawei.com

Stefano Bortoli
Huawei German Research Center
Munich, Germany
stefano.bortoli@huawei.com

Mohamad Al Hajj Hassan
Huawei German Research Center
Munich, Germany
mohamad.alhajjhassan@huawei.com

Carlos Salort Sánchez
Huawei German Research Center
Technical University of Munich
Munich, Germany
carlos.salort@huawei.com

Goetz Brasche
Huawei German Research Center
Munich, Germany
goetz.brasche@huawei.com

***Abstract*—Given the exponential data growth and the recent focus on understanding high-dimensional "in-motion" data, fundamental machine learning tools, such as Principal Component Analysis (PCA), require computation-efficient streaming algorithms that operate near-real-time. Despite the different streaming PCA flavors, there is no algorithm that provably recovers the principal components in the same precision regime as the batch PCA algorithm does, while maintaining low-latency and high-throughput processing. This work, introduces a novel temporal accumulate / retract learning framework for streaming PCA. We consider the accumulate / retract framework implementation of several competitive PCA algorithms with proven theoretical advantages. We benchmark the improved PCA algorithms on real-world streams (i.e. bank transactions fraud detection) and prove their low-latency (millisecond level) and high-throughput (thousands events/second) processing guarantees.**

***Index Terms*—Stream Processing, PCA, Online Learning**

I. Introduction

Given today's data deluge, carrying time-varying information, practitioners need to process streams of data under tight computational constraints for decision making (e.g. fraud / no fraud). Yet, data streams are seen as stochastic processes in which events occur continuously and independently from each other. Querying, processing and executing machine learning models on data streams is 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 tasks in signal processing and machine learning. Hence, due to its practical relevance, there is renewed interest in this problem [1], [2], [3], [4]. PCA is good for maintaining data structure in reduced subspaces in an unsupervised way. It is also useful in updating the decision boundaries and adding discriminately informative features with newly added samples and then updating the feature vectors by incremental eigenvector updates [5], [6]. This is highly relevant in multi-class classification problems, such as fraud detection. Basically, PCA is performing a (partial) singular value decomposition, and much work over the last half century has focused on efficient algorithms [7]. Computation-efficient algorithms that operate on streaming data are plentiful in the literature and many seem to do well in practice. However, there is no algorithm that provably recovers the principal components in the same precision regime as the batch PCA algorithm does, while maintaining low-latency high-throughput processing.

II. Related work

The recent focus on understanding streaming high-dimensional data, where the dimensionality of the data can potentially scale together with the number of available sample points, has led to an exploration of the complexity of covariance estimation underlying PCA [8], [9]. Such algorithms have provable complexity guarantees [10], [11] but either store all samples (i.e. for looping through samples) or explicitly maintain the covariance matrix. In high-dimensional applications, where data points are high-resolution images, video or bank transactions data, storing all data is prohibitive. Different from previous approaches, our work brings the focus on two critical quantities: latency and throughput. In the streaming setting, many approaches for incremental or online PCA have been developed, some focusing on replacing the inefficient steps (i.e. high latency of low-rank modifications of the singular value decomposition [12], others on conditioning in householder transformation for QR decomposition etc.) or computation-efficient operations [13], [14], [15]. Despite the multitude of successful dedicated algorithms ([16], [17], [18]), there is no algorithm that brings the focus on the two critical quantities of focus in streaming analytics. Our work utilizes a new paradigm for stream processing and learning [19] and proposes the implementation of several competitive streaming PCA algorithms for efficient computation with low-latency and high-throughput (i.e. through orchestration of the stream data flow). Our accumulate / retract learning framework offers a solution for incrementally computing on data streams

978-1-7281-4550-1/19/$31.00 ©2019 IEEE
DOI 10.1109/ICMLA.2019.00194

simple, complex or combinations of features, from statistical measures to machine learning model parameters. It assumes no need to recompute features by looping through all the acquired data, rather only through simple updates based only on the new values, [20], [21], and eliminated values and the old feature value [22], [23]. 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) [24]. 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. fraud detection) 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.

III. Formalizing the problem

While much work has focused on resources-constrained PCA, [25], [26], there is little work that provides complexity guarantees competitive with batch algorithms, [12], and, to our knowledge, no work on low-latency, high-throughput in large-scale streaming machine learning applications with programmatic resource allocation. In its basic formulation, PCA computes the eigenvectors and eigenvalues of the sample covariance matrix, using a numerical method such as the QR decomposition or Gram-Schmidt orhtogonalization [1]. This approach requires that all the training data are available before the principal components can be estimated. An incremental method, on the other side, is required to compute the principal components for observations arriving sequentially, where the principal components estimates are updated by each arriving observation [27]. In order to formalize the problem, consider a stream of n-dimensional data vectors $x(t)$. The problem of extracting incrementally the principal components assumes reducing the number of linear combinations of inputs by discarding low variances and only keeping the combinations with large variance:

$$w_1^T x, w_2^T x, w_3^T x, ..., w_p^T x, p \leq n \tag{1}$$

which maximize the expectation

$$E\{(w_i^T x)^2\}_{i=1:p} \tag{2}$$

under the constraints

$$w_i^T w_j = \delta_{ij}, j < i, \tag{3}$$

where δ_{ij} is the Kronecker product. The solution for the vectors $w_1, w_2, w_3, ..., w_p$ are the p dominant eigenvectors of the data covariance matrix calculated incrementally,

$$C = E\{xx^T\}. \tag{4}$$

These are p orthogonal unit vectors $z_1, z_2, z_3, ..., z_p$ given by

$$Cz_i = \lambda_i z_i \tag{5}$$

Data: Stream of n-dimensional data vectors $x(t)$
Result: Eigen features w_p and λ_p
Compute the mean feature vector $\mu = \frac{1}{n}\sum_k^n x_k(t)$;
Compute the covariance matrix
$C = E\{x - \mu\}\{x - \mu\}^T$;
Compute eigenvalues λ_i and eigenvectors z_i of C;
while *Estimate high-value eigenvectors* **do**
  Sort eigenvalues λ_i in decreasing order (i.e. histogram sort / counting);
  Choose a threshold θ ;
  Choose the first p dominant λ_i to satisfy $(\sum_i^p \lambda_i)(\sum_i^n \lambda_i)^{-1} \geq \theta$;
  Select eigenvectors w_p corresponding to λ_p;
end
Extract principal components from x, $P = V^T x$ where V is the matrix of principal components

Algorithm 1: Basic PCA processing steps.

where $\lambda_1, \lambda_2, \lambda_3, ..., \lambda_p$ are the largest p eigenvalues of C in descending order $\lambda_1 > \lambda_2 > \lambda_3 > ... > \lambda_p$. The first principal component is $c_1^T x$ whereas the minor components are linear combinations $c_n^T x, c_{n-1}^T x$ where c_n is the eigenvector corresponding to the smallest eigenvalue. In a structured form, the basic formulation, PCA follows the steps in algorithm 1. For large problems this routine is not computationally efficient and various numerical methods have been used to improve it, using QR decomposition [28] or the Householder reflections [29] replacing Gram-Schmidt, known to lead to cancellation that causes inaccuracy of the computation and a non-orthogonal matrix. In our work we analyzed the inefficient internal operations which impede PCA to achieve the two critical quantities: latency and throughput, when operating on streams. We identified three aspects in existing approaches which impact the streaming PCA formulation:

- the continuous calculation of the mean μ and other descriptive statistics (i.e. covariance) on the datastream;
- sorting the dominant eigenvalues in the rank update of the QR decomposition and the ordering of lower/upper triangular sub-matrices;
- the complexity of computations performed at each training step.

As we see, stream processing applications distinguish themselves from the traditional store-and-process data analysis through four features: continuous data sources, continuous and long-running analysis, time-to-respond performance requirements, and failure tolerance requirements [30], [24]. These characteristics together with the identified bottlenecks will constitute design objectives for our approach, we leverage the advantages of streaming PCA for low-latency, high-throughput, efficient stream processing, as shown in the next sections.

IV. Materials and 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. Our motivation for this work stems from the need to guarantee performance and efficiency and the fact that no other method looked at the two quantities as a whole. 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 [19], capable of incrementally computing machine learning models on data streams, we successfully instantiated multiple streaming PCA models for high-performance (i.e. low-latency, high-throughput processing) in an intensive real-world scenario (i.e. fraud detection). 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 as the stream progresses. The accumulate-retract framework allows for incremental calculation of statistical quantities used in streaming PCA. For example, the average calculation in the eigenvalue update in incremental form can be visualized in Figure 1. Such incremental computation tackles successfully

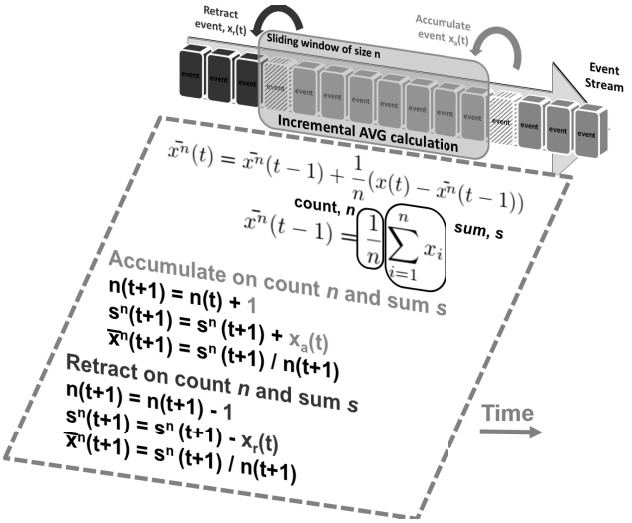

Fig. 1. Streaming PCA incremental average update using accumulate/retract.

the first bottleneck, namely the incremental calculation of the mean and other descriptive statistics on the datastream. As shown in Figure 1, the average computation is split in dual operations that keep sub-quantities 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 Figure 2. In this case the caches (i.e. fixed memory partitions) 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 - following histogram sorting. Each time new eigenvalues are computed sorting is triggered. 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. 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 Figure 2. There are many models for learning PCA

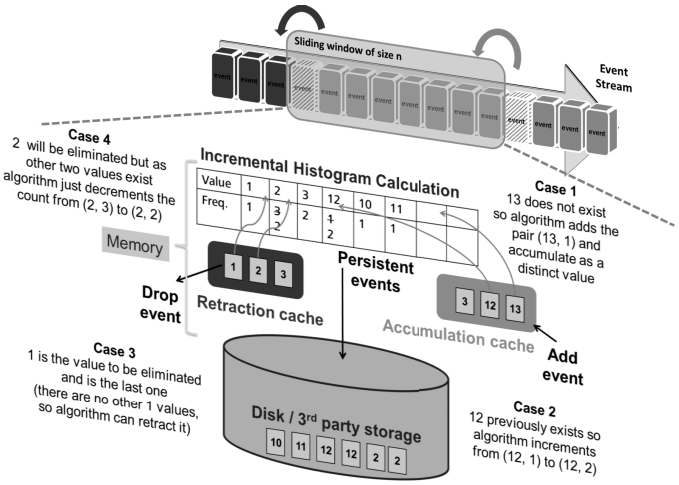

Fig. 2. Streaming PCA eigenvalues sorting on streams: histogram update using accumulate/retract.

efficiently [31], from stochastic approximation models (i.e. Hebbian and Ojas Learning Rules [32]), to subspace learning [33], nonlinear PCA neural networks [34] up to denoising autoencoders [35]. 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 fraud detection scenarios.

The first model considered in our experiments is a single-layer neural network based on the Stochastic Gradient Ascent (SGA) [33]. We chose this model because it efficiently provides a description of the covariance matrix, which is typically too expensive to be estimated online [36]. The SGA-based Streaming PCA is depicted in Figure 3. The accumulate / retract learning maps easily on the structure of the network and the corresponding learning rule in Equation 6 due to its dual operation when forward propagating information from the input $x(k)$ to the hidden layer $y(k)$ by updating the weight matrix W.

$$\Delta w_j(t-1) = \gamma(t) y_j(t)(x(t) - y_j(t) w_j(t-1) \quad (6)$$

$$-2 \sum_{i<j} y_i(t) w_i(t-1)), \quad (7)$$

where $\gamma(t)$ is the learning rate.

The second model employed in our experiments is the Generalized Hebbian Algorithm (GHA) [37]. The model uses Hebbian learning to achieve optimal unsupervised extraction

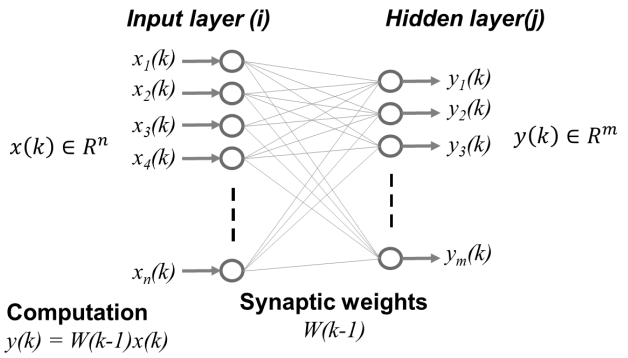

Fig. 3. Neural PCA network.

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 in Figure 3, the GHA-based Streaming PCA learning rule is given by:

$$\Delta w_{ij}(t) = \gamma(t)(y_j(t)x_i(t) - y_j(t)\sum_{k<i} w_{kj}(t)y_k(t)) \quad (8)$$

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) [36]. 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,

$$\lambda x = Ax \quad (9)$$

and

$$v = \lambda x \quad (10)$$

then the learning rule for CCIPCA is given by:

$$v(n) = \frac{n-1-l}{n}v(n-1) + \frac{1+l}{n}u(n)u^T(n)\frac{v(n-1)}{||v(n-1)||}, \quad (11)$$

where l is the "amnesic parameter" (i.e. a forgetting parameter).

The SGA learning rule is purely local in the sense that the change in each individual weight only depends on factors that would be locally available at that neuron's position. This rule behaves better for extracting the dominant eigenvectors than other methods, such as GHA [37]. The weight update takes advantage of the accumulate-retract framework, incrementally incorporating new knowledge while decreasing the impact that old data has upon the update, as show in Figure4. 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 11, which is analog to w in neural models. In the accumulate / retract framework such an update is performed as new elements are available from the stream, Figure 4. For each of the considered models,

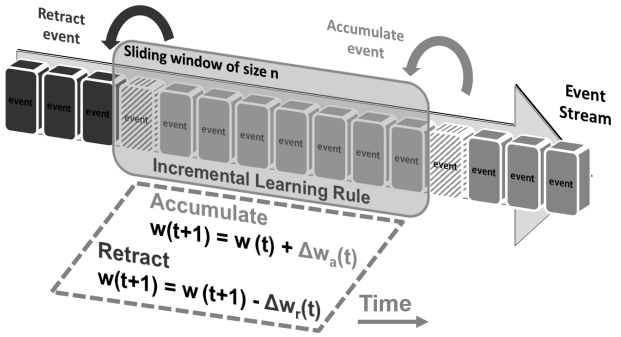

Fig. 4. Streaming PCA weight update in the accumulate/retract framework.

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

$$Cz_i = \lambda_i z_i \quad (12)$$

where λ_i are the eigenvalues of the covariance matrix, $C[c_{jk}]$. We can then rewrite C as

$$c_{jk} = \frac{1}{n-1}\sum_{i=1}^{n}(z_{ij} - \bar{z_j})(z_{ik} - \bar{z_k}) \quad (13)$$

where

$$\bar{z} = \frac{1}{n}\sum_{i=1}^{n} z_i \quad (14)$$

is the incremental average. Hence, we can rewrite the covariance matrix C as

$$C = \frac{1}{n-1}\sum_{i=1}^{n}(z_i - \bar{z})(z_i - \bar{z})^T \quad (15)$$

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

$$v = \lambda z \quad (16)$$

estimate at time t is

$$v(t) = \frac{1}{n}\sum_{i=1}^{n} x(t)x^T(t)z(t). \quad (17)$$

We can now calculate the eigenvalues and eigenvectors given v as we know that

$$\lambda = \|v\|. \quad (18)$$

If we consider

$$z = \frac{v}{\|v\|} \tag{19}$$

we can rewrite

$$v(t) = \frac{1}{n}\sum_{i=1}^{n} x(t)x^T(t)\frac{v(t-1)}{\|v(t-1)\|}. \tag{20}$$

Such computation steps provide a closed form implementation of learning rule in the accumulate-retract framework [19].

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

$$E\{(x - \hat{x})^2\} \tag{21}$$

when the rows of W span the first p eigenvectors of C and the Linear Least Squares (LLS) estimate of x given y is

$$\hat{x} = CW^T(WCW^T)^{-1}y. \tag{22}$$

If the rows of W are the first eigenvectors then

$$WW^T = I \tag{23}$$

and

$$C = W^T\Lambda W, \tag{24}$$

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

$$\min_f \sum_{i=1}^{n}(y_i - f(x_i))^2. \tag{25}$$

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

$$\min_f \sum_{i=1}^{n}(y_i - ax_i - b)^2. \tag{26}$$

In the accumulate-retract framework such a problem is incrementally solved as the datastream progresses, using simple updates shown in Figure 5

Given that covariance can be incrementally calculated as

$$cov_{xy}(t) = \frac{n-2}{n-1}cov_{xy}(t-1) \tag{27}$$

$$+\frac{1}{n}(x^n(t) - \bar{x}^{n-1}(t))(y^n(t) - \bar{y}^{n-1}(t)) \tag{28}$$

the problem in closed form assumes calculating incrementally a and b as

$$a(t) = \frac{cov_{xy}(t)}{m_2(t)}, b(t) = \bar{y}(t) - a\bar{x}(t) \tag{29}$$

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 accurately. This is not true

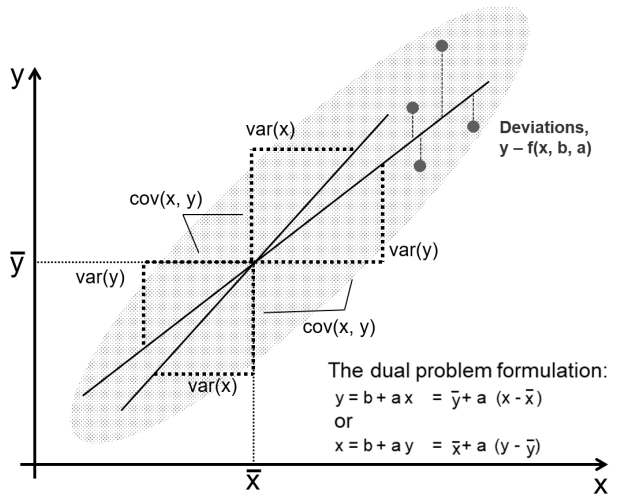

Fig. 5. Streaming PCA optimization using LLS in accumulate/retract framework.

in practical applications, thus we are seeking the values of a and b that minimize

$$\min_{a,b} \sum_{i=1} \frac{(y_i - ax_i - b)^2}{1 + a^2}, \tag{30}$$

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) [38], 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

$$E_{TLS} = \min_d \frac{(dx + b)^2}{d^2} \tag{31}$$

over all inputs x (including y). We can rewrite the same constraint as

$$E_{TLS} = n \min_d \frac{d^t Rd + 2bd^T E(x) + b^2}{d^T d} \tag{32}$$

where

$$R = \frac{1}{n}\sum_{i=1}^{n} x_i x_i^T \tag{33}$$

is the autocorrelation of the input and

$$E(x) = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{34}$$

is the average of the input. At convergence ($\frac{dE_{TLS}}{dz} = 0$) we must have $Rd + bE(x) - \theta d = 0$ where

$$\theta = \frac{d^T Rd + 2bd^T E(x) + b^2}{d^T d}. \tag{35}$$

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

$$\theta = \frac{d^T Cd}{d^T d} \tag{36}$$

and

$$C = R - E(xx^T) \tag{37}$$

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 [39].

V. Experiments and discussion

This section introduces the results and the analysis of the three streaming PCA models instantiated in our framework using Apache Flink [30]. 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 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 our 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.

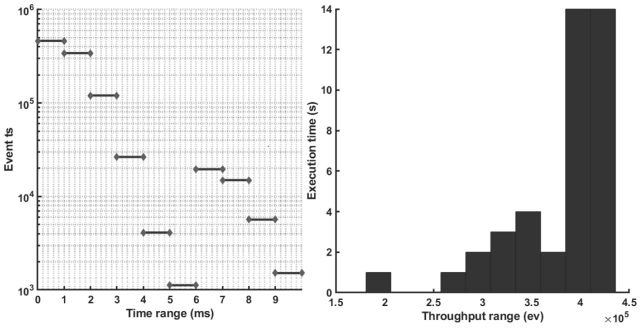

Fig. 6. Analysis of QR-based PCA without accumulate/retract (left latency, right throughput).

We used a real-world stream with online bank transactions (reader can also refer to the similar PKDD'99 Discovery Challenge - Guide to the Financial Data Set) and queried the eigenvalues and eigenvectors prior a multi-class classifier. The dataset has 10 input features (i.e. transaction id, account id, transaction amount, balance after transaction, transaction bank partner, transaction account partner, transaction type, transaction operation, transaction symbol, transaction date) describing various aspects of the executed transaction.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 label for the multi-class classification task (i.e. various types of valid and fraud transactions - in our scenario, we consider 10 classes: normal transaction and 9 types of frauds). To give the user a sample of the possible output, some sample class labels were, "high-risk fraud", "recurrent fraud", "low-risk valid", "recurrent valid".

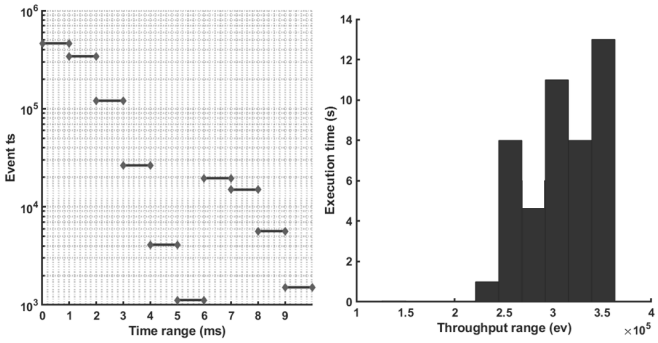

Fig. 7. Analysis of streaming PCA (SGA model) without accumulate/retract framework (left latency, right throughput).

In order to evaluate the three streaming PCA models, we also implemented an efficient QR-based PCA of [28] 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 with fixed / programmable resource allocation. 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 Figure 6 and Figure 7, 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 Figure 6 and Figure 7, 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 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

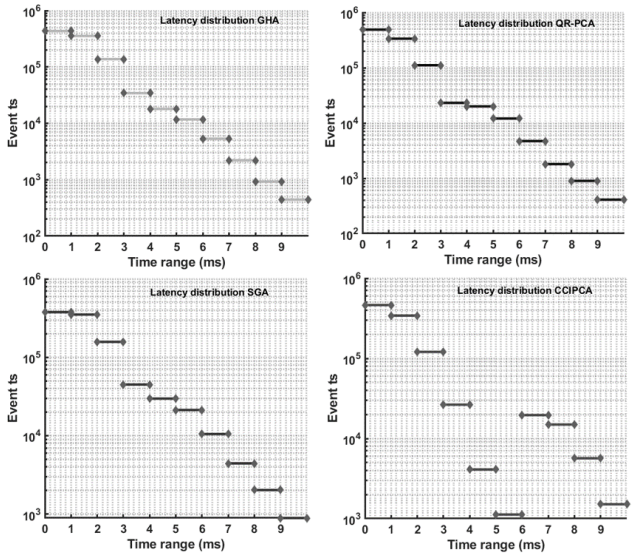

Fig. 8. Comparative analysis of streaming PCA models latency in the accumulate/retract framework.

the accumulate-retract learning framework. We now analyze a series of large-scale experiments meant at extracting the eigenfeatures for the multi-class fraud detection scenario (i.e. 2M events streamed at 40 kHz, purchases on Single's Day in China). In terms of latency one can observe that the streaming

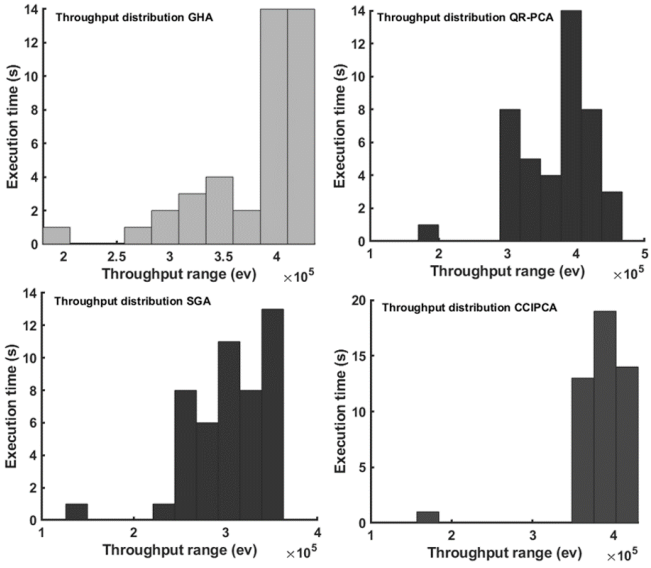

Fig. 9. Comparative analysis of streaming PCA models throughput in the accumulate/retract framework.

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 Figure 8. This is supported by the gain of $\sim$1k events throughput, as shown in Figure 9 in the accumulate / retract framework and up to $\sim$10k more than in the baseline experiments. This is also visible in the core distribution of throughput ranges peaking at around 40k events/s. 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

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

TABLE I
EIGENVALUES AND EIGENVECTORS ESTIMATES ANALYSIS

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 fraud detection. The code repository for the Streaming PCA benchmarking is available at [1].

VI. CONCLUSION

Tackling the theoretical bottlenecks in traditional PCA algorithms and focusing on two critical quantities, namely latency and throughput, the current study supports the renewed interest and performance improvements in streaming machine learning. Traditional statistical packages require to have available prior to the calculation, a batch of examples from the distribution being investigated. While it is possible to run multiple flavors of PCA models with the accumulate-retract framework, the strong advantage is brought by the adaptive and incremental processing and learning capabilities of the framework addressing all of the two critical quantities mentioned before, namely low-latency and high-throughput processing. Strictly speaking, PCA is only defined for stationary distributions. However, in realistic situations, as streaming data, it is often the case that we are interested in compressing data from distributions which are a function of time; in this situation, the streaming PCA is a good solution in that it tracks the moving statistics of the distribution and provides as close to batch PCA. Following a distribution's statistics, streaming PCA is an example of trade-off between tracking capability and accuracy of convergence. Finally, as our experiments show, streaming PCA models can be leveraged by the accumulate / retract framework towards high-performance

[1]https://github.com/omlstreaming/icmla2019

as the amount of events in the data stream increases. Such a system offers flexibility, allowing for arbitrary combinations of multiple functions underlying complex machine learning models (i.e. average, least squares regression, 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. The fraud detection benchmark is based on the capability of the system to fingerprint potential fraud types, based on the underlying eigenvectors / eigenvalues configurations extracted from the incoming stream. This allows fast convergence to one of the class labels encoding a type of fraud.

REFERENCES

[1] A. Sarveniazi, "An Actual Survey of Dimensionality Reduction," *American Journal of Computational Mathematics*, vol. 4, no. 4, pp. 55–72, 2014.

[2] S. Bubeck, N. Cesa-Bianchi *et al.*, "Regret analysis of stochastic and nonstochastic multi-armed bandit problems," *Foundations and Trends® in Machine Learning*, vol. 5, no. 1, pp. 1–122, 2012.

[3] R. Arora, A. Cotter, K. Livescu, and N. Srebro, "Stochastic optimization for pca and pls," in *2012 50th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 2012, pp. 861–868.

[4] J. Zhan, B. Lois, H. Guo, and N. Vaswani, "Online (and offline) robust pca: Novel algorithms and performance guarantees," in *Artificial intelligence and statistics*, 2016, pp. 1488–1496.

[5] B. Nadler, "Finite sample approximation results for principal component analysis: A matrix perturbation approach," *Ann. Statist.*, vol. 36, no. 6, pp. 2791–2817, 12 2008.

[6] B. Lois and N. Vaswani, "A correctness result for online robust pca," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015, pp. 3791–3795.

[7] G. H. Golub and C. F. Van Loan, *Matrix Computations (3rd Ed.)*. Baltimore, MD, USA: Johns Hopkins University Press, 1996.

[8] C. G. Baker, K. A. Gallivan, and P. Van Dooren, "Low-rank incremental methods for computing dominant singular subspaces," *Linear Algebra and its Applications*, vol. 436, no. 8, pp. 2866–2888, 2012.

[9] M. Brand, "Incremental singular value decomposition of uncertain data with missing values." in *ECCV (1)*, ser. Lecture Notes in Computer Science, A. Heyden, G. Sparr, M. Nielsen, and P. Johansen, Eds., vol. 2350. Springer, 2002, pp. 707–720. [Online]. Available: http://dblp.uni-trier.de/db/conf/eccv/eccv2002-1.html

[10] F. Hallgren and P. Northrop, "Incremental kernel pca and the nystr\" om method," *arXiv preprint arXiv:1802.00043*, 2018.

[11] Y. Li, "On incremental and robust subspace learning," *Pattern recognition*, vol. 37, no. 7, pp. 1509–1518, 2004.

[12] I. Mitliagkas, C. Caramanis, and P. Jain, "Memory limited, streaming pca," in *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, ser. NIPS'13. USA: Curran Associates Inc., 2013, pp. 2886–2894.

[13] C. Boutsidis, D. Garber, Z. Karnin, and E. Liberty, "Online principal components analysis," in *Proceedings of the twenty-sixth annual ACM-SIAM symposium on Discrete algorithms*, 2015, pp. 887–901.

[14] T.-J. Chin and D. Suter, "Incremental kernel principal component analysis," *IEEE Transactions on Image Processing*, vol. 16, no. 6, pp. 1662–1674, 2007.

[15] A. Balsubramani, S. Dasgupta, and Y. Freund, "The fast convergence of incremental pca," in *Advances in Neural Information Processing Systems*, 2013, pp. 3174–3182.

[16] Y. Yin, D. Xu, X. Wang, and M. Bai, "Online state-based structured svm combined with incremental pca for robust visual tracking," *IEEE transactions on cybernetics*, vol. 45, no. 9, pp. 1988–2000, 2015.

[17] P. Jain, C. Jin, S. M. Kakade, P. Netrapalli, and A. Sidford, "Streaming pca: Matching matrix bernstein and near-optimal finite sample guarantees for ojas algorithm," in *Conference on Learning Theory*, 2016, pp. 1147–1164.

[18] F. Zhao, I. Rekik, S.-w. Lee, J. Liu, J. Zhang, and D. Shen, "Two-phase incremental kernel pca for learning massive or online datasets," *Complexity*, vol. 2019, 2019.

[19] C. Axenie, R. Tudoran, S. Bortoli, M. A. H. Hassan, D. Foroni, and G. Brasche, "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, December 17-20, 2018*, 2018, pp. 1115–1122.

[20] D. Logothetis, C. Olston, B. Reed, K. C. Webb, and K. Yocum, "Stateful bulk processing for incremental analytics," in *Proceedings of the 1st ACM Symposium on Cloud Computing*, ser. SoCC '10. New York, NY, USA: ACM, 2010, pp. 51–62. [Online]. Available: http://doi.acm.org/10.1145/1807128.1807138

[21] D. Peng and F. Dabek, "Large-scale incremental processing using distributed transactions and notifications," in *Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation*, ser. OSDI'10. Berkeley, CA, USA: USENIX Association, 2010, pp. 251–264. [Online]. Available: http://dl.acm.org/citation.cfm?id=1924943.1924961

[22] C. Demetrescu, D. Eppstein, Z. Galil, and G. F. Italiano, *Algorithms and Theory of Computation Handbook*. Chapman & Hall/CRC, 2010. [Online]. Available: http://dl.acm.org/citation.cfm?id=1882757.1882766

[23] Z. Huang and P. Peng, "Dynamic graph stream algorithms in space," *CoRR*, vol. abs/1605.00089, 2016. [Online]. Available: http://arxiv.org/abs/1605.00089

[24] G. H. Albert Bifet, Ricard Gavald and B. Pfahringer, *Machine Learning for Data Streams with Practical Examples in MOA*. MIT Press, 2018.

[25] M. K. Warmuth and D. Kuzmin, "Randomized pca algorithms with regret bounds that are logarithmic in the dimension," in *Advances in neural information processing systems*, 2007, pp. 1481–1488.

[26] R. Vershynin, "How close is the sample covariance matrix to the actual covariance matrix?" *Journal of Theoretical Probability*, vol. 25, no. 3, pp. 655–686, Sep 2012.

[27] O. Shamir, "Convergence of stochastic gradient descent for pca," in *International Conference on Machine Learning*, 2016, pp. 257–265.

[28] A. Sharma, K. K. Paliwal, S. Imoto, and S. Miyano, "Principal component analysis using qr decomposition," *International Journal of Machine Learning and Cybernetics*, vol. 4, no. 6, pp. 679–683, Dec 2013. [Online]. Available: https://doi.org/10.1007/s13042-012-0131-7

[29] A. S. Householder, "Unitary triangularization of a nonsymmetric matrix," *J. ACM*, vol. 5, no. 4, pp. 339–342, Oct. 1958. [Online]. Available: http://doi.acm.org/10.1145/320941.320947

[30] P. Carbone, A. Katsifodimos, S. Ewen, V. Markl, S. Haridi, and K. Tzoumas, "Apache flink: Stream and batch processing in a single engine," *IEEE Data Eng. Bull.*, vol. 38, pp. 28–38, 2015. [Online]. Available: https://flink.apache.org/introduction.html

[31] J. Qiu, H. Wang, J. Lu, B. Zhang, and K.-L. Du, "Neural network implementations for pca and its extensions," *ISRN Artificial Intelligence*, vol. 2012, 2012.

[32] E. Oja, "Simplified neuron model as a principal component analyzer," *Journal of mathematical biology*, vol. 15, no. 3, pp. 267–273, 1982.

[33] ——, "Principal components, minor components, and linear neural networks," *Neural networks*, vol. 5, no. 6, pp. 927–935, 1992.

[34] M. A. Kramer, "Nonlinear principal component analysis using autoassociative neural networks," *AIChE journal*, vol. 37, no. 2, pp. 233–243, 1991.

[35] H. Valpola, "From neural pca to deep unsupervised learning," in *Advances in Independent Component Analysis and Learning Machines*. Elsevier, 2015, pp. 143–171.

[36] J. Weng, Y. Zhang, and W.-S. Hwang, "Candid covariance-free incremental principal component analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 8, pp. 1034–1040, 2003.

[37] T. D. Sanger, "Optimal unsupervised learning in a single-layer linear feedforward neural network," *Neural Networks*, vol. 2, no. 6, pp. 459 – 473, 1989. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0893608089900440

[38] L. Xu, E. Oja, and C. Y. Suen, "Modified hebbian learning for curve and surface fitting," *Neural Networks*, vol. 5, no. 3, pp. 441–457, 1992.

[39] S. Woo and C. Lee, "Incremental feature extraction based on decision boundaries," *Pattern Recognition*, vol. 77, pp. 65 – 74, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S003132031730496X