Real Time Principal Component Analysis

Conference Paper · April 2019

DOI: 10.1109/ICDE.2019.00171

3 authors:

Ranak Roy Chowdhury
University of California, San Diego
2 PUBLICATIONS **0** CITATIONS
SEE PROFILE

Muhammad Abdullah Adnan
Bangladesh University of Engineering and Technology
52 PUBLICATIONS **144** CITATIONS
SEE PROFILE

Rajesh Gupta
University of California, San Diego
503 PUBLICATIONS **13,808** CITATIONS
SEE PROFILE

Real Time Principal Component Analysis

Ranak Roy Chowdhury*, Muhammad Abdullah Adnan*, and Rajesh K. Gupta†
*Bangladesh University of Engineering and Technology (BUET), Dhaka, Bangladesh
†University of California San Diego, California, USA
Email: ranakrc@gmail.com, abdullah.adnan@gmail.com, gupta@eng.ucsd.edu

***Abstract*—By processing the data in motion, real-time data processing enables us to extract instantaneous results from online input data that ensures timely responsiveness to events as well as a much enhanced capacity to process large data sets. This is especially important when decision loops include querying and processing data on the web where size and latency considerations make it impossible to process raw data in real-time. This makes dimensionality reduction techniques, like principal component analysis (PCA), an important data preprocessing tool to gain insights into data. In this paper, we propose a variant of PCA, that is suited for real-time applications. In the real-time version of the PCA problem, we maintain a window over the most recent data and project every incoming row of data into lower dimensional subspace, which we generate as the output of the model. The goal is to minimize the reconstruction error of the output from the input. We use the reconstruction error as the termination criteria to update the eigenspace as new data arrives. To verify whether our proposed model can capture the essence of the changing distribution of large datasets in real-time, we have implemented the algorithm and evaluated performance against carefully designed simulations that change distributions of data sources over time in a controllable manner. Furthermore, we have demonstrated that our algorithm can capture the changing distributions of real-life datasets by running simulations on datasets from a variety of real-time applications e.g. localization, customer expenditure, etc. We propose algorithmic enhancements that rely upon spectral analysis to improve dimensionality reduction. Results show that our method can successfully capture the changing distribution of data in a real-time scenario, thus enabling real-time PCA.**

***Index Terms*—Big Data; Real Time; Dimensionality Reduction; PCA;**

I. Introduction

Real big data processing has ubiquitous applications in real world. For example, during times of a calamity or holocaust, social media platforms, like Facebook and Twitter, are deluged with posts and pictures relating to that event [1] [2] [3]. During such an event, the sudden shifts in the distribution of words across data streams, may encode valuable information. To perform real time social media analysis and trend detection [4] [5] [6], it is crucial to identify such change points and accurately approximate the new distribution immediately from fresh data. However, processing data of such magnitude renders the system slow, which is costly during critical times [7] [8]. Hence, the system must process the data to filter out the minor changes and focus only on the **major shifts** occurring in the data distribution. Therefore any system, where it is vital to capture the shift in distribution of data, requires fast data processing and on the fly computation. Moreover, the magnitude of data that needs to be processed is getting bigger, making this task more difficult. Existing real-time data preprocessing architectures are neither fast nor scalable to handle the enormous data generated every second.

In this paper, we propose a model that exploits the potential of big data and is fast enough for a real-time application. Hence, we use a popular dimensionality reduction technique, known as Principal Component Analysis (PCA) [9] [10] [11], which reduces the dimensionality of data, yet retains as much of the underlying information as possible. PCA is a statistical procedure that is used for dimensionality reduction, clustering, classification, factor analysis, correlation analysis and many more advanced scientific computing and research fields. So, Principal Component Analysis computes the most meaningful basis to express our data in a lower dimensional representation.

However, standard offline PCA requires the whole data to be present before computation begins; an unreasonable assumption in a real-time system. In a real-time system, streaming data arrives and they must be discarded as soon as they are computed upon. Otherwise the huge incoming data becomes a burden on computational resources. The computation must be completed before the next data arrives. **Online PCA** [12] [13] also computes PCA for streaming data but aims to provide the best low dimensional representation of the **whole data** seen so far which is not real-time. To the best of our knowledge, **real-time PCA** is the first work that provides a low dimensional representation of the **current distribution** of the data using PCA, besides **retaining the major components** of the previous distributions seen so far.

In this paper, we propose the concept of Real Time PCA algorithm. Our algorithm operates on streaming high-dimensional input data. The window size and the target dimension are specified as input parameters. In our proposed algorithm, we construct a window that slides past input data. We use reconstruction error between the data in the window and its corresponding projected output to decide how much to modify the lower dimensional subspace, given by PCA.

We conduct extensive experiments to run these algorithms on synthetic and real datasets from various real-time applications. We experimentally verify that our algorithm can capture the changing distributions of streaming data in reduced dimension. Through the use of spectral analysis, we verify whether the algorithms demonstrate their anticipated behavior whenever there is a change in data distribution. We also make use of a popular statistical method known as the Bhattacharyya coefficient method [14], which measures the degree of similarity between two probability distributions, to show how well the output distributions, generated by our algorithms, correspond to the actual input distributions. To further strengthen the validity of real-time PCA, we use Bhattacharyya coefficient to also show that the statistical changes occurring across the input distributions are approximately similar to those occurring across the output distributions.

The rest of the paper is organized as follows. Section II discusses the problem definition, data structures used and the proposed algorithm, along with its time and space complexity. Section III discusses the performance and validation of our algorithms through experiments on synthetic and real datasets. Section IV provides the conclusion and future works.

II. Proposed Algorithm

This section formulates the problem of real-time PCA and states the meaning of the mathematical notations used. Then we propose our Algorithm **Real Time PCA (RPCA)**.

In the real-time version of the PCA problem, at each timestep t, input vector x_t of dimension d, is given as an input to the algorithm. The target dimension l and the window size k are specified as input parameters. l denotes the target dimension in Real Time PCA, therefore $l < d$ for dimensionality reduction to occur. At timestep t, the input data x_t arrives and we build a window B_t with the latest k instances of data. Hence, B_t is of size $k \times d$.

The goal of online PCA is to minimize the frobenius norm of the reconstruction error $\|X(I - UU^T)\|_F^2$ [12] or the spectral norm of the reconstruction error $\|X(I - UU^T)\|_2^2$ [13]. Here, X is the entire dataset of size $n \times d$ and U is the eigenvector matrix of size $d \times l$. But the goal of real-time PCA is to minimize the spectral norm of the reconstruction error $\|B_t(I - UU^T)\|_2^2$ with respect to B_t which contains only the most recent k data items upto timestep $t - k + 1$. So Real Time PCA aims to minimize the reconstruction error of the **current block** B_t, whereas online PCA tries to minimize the reconstruction error with respect to the **entire dataset**.

We generate the corresponding output vector y_t of dimension l from $y_t = x_t U$, for timestep t, $t = 1, 2, ..., n$. We must generate the output y_t corresponding to each input x_t before getting the next input x_{t+1}. Hence the input and output of a desired real-time PCA algorithm follows:

- **Input:** Set of vectors $X = [x_1, ..., x_n]$ in $\mathbb{R}^{n \times d}$, a target dimension l, $l < d$ and a window size k .
- **Output:** Set of vectors $Y = [y_1, ..., y_n]$ in $\mathbb{R}^{n \times l}$ that minimizes the reconstruction error $\|B_t(I - UU^T)\|_2^2$, for $t = 1, 2, ..., n$.

The explanation of the mathematical notation of the different data structures used for computation are as follows:

- $U \leftarrow$ eigenvector matrix, initialized as an all zeros matrix of size $d \times l$.
- $B \leftarrow$ window, initialized as an all zeros matrix of size $k \times d$.
- $\sigma_{arr} \leftarrow$ array of eigenvalues corresponding to the eigenvectors in U, initialized as an all zeros array of size l.

The algorithm formulated in this paper uses a function named $constructWindow$ to construct a window B_t by incorporating the input data x_t at timestep t into B_t and discarding the data which arrived at timestep $t - k$ from B_t.

In Algorithm 1 (RPCA), at each iteration, we compute the top eigenvalue σ and its corresponding eigenvector u from the

Algorithm 1: Real Time PCA (RPCA)

```
   Input: $X$, $k$, $l$
   Output: $Y$
1  $U \leftarrow$ all zeros matrix of size $d \times l$
2  $B \leftarrow$ a window of size $k \times d$
3  $\sigma_{arr} \leftarrow$ all zeros array of size $l$
4  for $x_t \in X$ do
5      $B_t$ = constructWindow($x_t, B_{t-1}$)
6      $min \leftarrow$ minimum of $\sigma_{arr}$
7      $[\sigma, u] \leftarrow$ top eigenvalue and corresponding
        eigenvector computed from pca of $B_t(I - UU^T)$
8      if $\sigma > min$ then
9          Replace vector from $U$ by $u$ and $min$ in $\sigma_{arr}$ by
            $\sigma$
10     end
11     $y_t = x_t U$
12 end
```

reconstruction error $B_t(I - UU^T)$. Then we find the minimum eigenvalue of the eigenvectors currently in U. Therefore, min is the minimum of the eigenvalues currently in σ_{arr}. If σ is larger than min, we update σ_{arr} and U by substituting min in σ_{arr} and its corresponding eigenvector in U by σ and u, respectively. We do this once at every iteration. The time complexity and space complexity of Algorithm 1 (RPCA) are $\mathcal{O}(l^2 dn)$ and $\mathcal{O}(n + d)l$, respectively.

Our algorithm is **time-efficient** because the computation of $\|B_t(I - UU^T)\|_2^2$, which is a bottleneck, is performed only once for a single input x_t. Besides, it is also **robust to noise** because it updates at most one eigenvector from U at each iteration. Hence, if a noise sample arrives, U will not be drastically modified. However, if the sample is **not** noise but belongs to a **new** distribution, then a series of successive samples that belong to the latest distribution will arrive over the next couple of iterations. Hence, this algorithm will update U over the next couple of iterations to capture the eigenspace of the new distribution. Thus our algorithm can capture the latest distribution while being robust to noise.

III. Experimental Evaluation

In this section, we first describe our experimental settings. Then we verify the correctness of our algorithms using Spectral Diagram and Bhattacharyya coefficient.

A. Experimental Settings

We perform experiments on both synthetic and real datasets from various real-life applications. The description of each dataset is provided in the first two columns of Table I. n, d and $distibutions$ in the second column of Table I refer to the number of instances, dimensionality and number of distributions, respectively. For datasets with a single value for $ChangePoint$, there is a periodic change in data distribution after that many instances of data. For datasets where $ChangePoint$ is specified as a set of values, changes in data distributions occur at those particular instances.

Algorithm 2: Synthetic data generation

```
    Input: $d$, $c$, $obs$
    Output: $X$
1   $\mu \leftarrow$ all zeros matrix of size $d$
2   $\Sigma \leftarrow$ all zeros array of size $d \times d$
3   for $i$ in $c$ do
4       Generate $\mu_i$ and $\Sigma_i$ randomly
5       $Y_i \leftarrow$ Generate $obs_i$ number of samples randomly
          from a Gaussian distribution defined by $(\mu_i, \Sigma_i)$
6       $X.append(Y_i)$
7   end
```

1) Synthetic Dataset: The synthetic dataset is generated according to standard Gaussian distributions. An advantage of using simulated datasets is that we can control the time when a change in the distribution occurs. Therefore, we can see how our algorithm behaves in those particular change points. The synthetic data generation procedure is illustrated in Algorithm 2. d refers to the input dimension, c is the number of distributions in the dataset and obs refers to an array of the number of samples to be drawn from each distribution. So obs_i refers to the number of samples to be drawn from the i-th distribution. μ_i and Σ_i refer to the mean and covariance matrix of the i-th distribution, respectively.

2) Real Datasets: The real datasets used are standard benchmark datasets collected from UCI Machine Learning Repository. They are a primary source of benchmark machine learning datasets. We chose datasets from multifarious applications to show that our algorithms operate successfully in multiple domains. We selected a diverse set of data with vastly different dimensionality, number of distributions and instances.

B. Experimental Verification

In this section, we verify that our algorithm implements PCA in real-time through spectral analysis and Bhattacharyya coefficient [14], BC. We illustrate how well our algorithm captures the changing distributions of streaming data in real-time. The third column of Table I shows the input values of window size, k and target dimension, l.

1) Verification through Spectral Diagram: In this section, we use spectral diagrams to verify our claim for Real Time PCA. Whenever an eigenvector is added to U, we increment a counter that counts the number of eigenvectors added at each iteration. The intuition is, whenever the underlying data distribution changes, a large proportion of eigenvectors will be replaced immediately following the change in distribution. Hence, the occurrence of spikes should signal when a change in distribution has occurred.

As the spectral diagrams in the fourth column of Table I show, a cluster of spikes are generated **immediately** following the change in the input data distribution. Hence, the frequency of eigenvectors added to U is higher when distribution changes from one class to another. Therefore, by updating U to account for the **latest** distribution, our algorithm is operating **real-time**.

2) Verification through Comparison between Corresponding Distributions of Input and Output Data: In this section, we use the statistical measure, Bhattacharyya coefficient [14], BC, to show similarity between input and output distributions. The intuition is, if the algorithm is updating U in accordance with the current data distribution, then the distribution of the projected output, generated from U, will be statistically similar to the corresponding input distribution.

We use BC to compute the statistical similarity between the corresponding distributions of the input and output. The mean of the BC values for all the distributions in a dataset are shown in Column 5 of Table I. A BC value close to 1 indicates similarity between two probability distributions [19]. As the values in the table show, all the mean BC values are close to 1, indicating that the output distributions are statistically similar to the corresponding input distributions. Despite the changing input distributions, our algorithm can successfully update U in accordance with the current distribution. So our algorithm captures the essence of **changing distributions** in **streaming** data over time and simultaneously **retain the major components of previous distributions** seen so far, thereby validating **real-time** PCA. Moreover, Column 6 of Table I shows that the values for the standard deviation of BC values are quite low. Therefore, the BC values are consistently high and have low variability across all distributions.

*3) Verification through Comparison between the **Change** in Corresponding Distributions of Input and Output Data:* In the previous section, we showed that the output distributions are statistically similar to those of the input. In this section, we reinstate the validity of our algorithm by showing that the changes occurring across the input distributions are also statistically similar to those observed across the output. The intuition is, if the input distributions are changing, then the algorithm should update U in a way such that these changes are reflected across the output distributions.

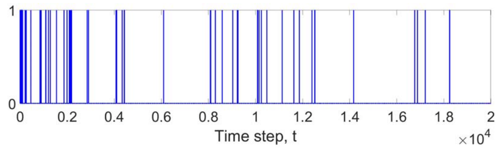
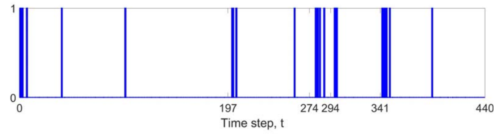
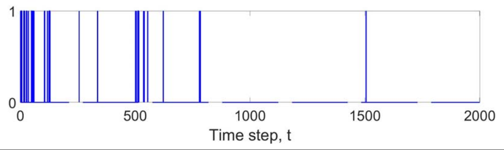
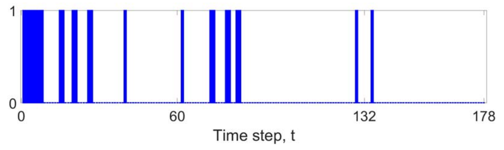
We measure the statistical change between two successive input distributions using Bhattacharyya coefficient. Then we similarly compute the statistical change between the corresponding two successive output distributions. We take the absolute value of the difference between the two BC scores. The lower this value, the closer the distribution change of the output is to that of the input. We repeat this procedure for all pairs of successive distributions in the data.

Column 7 of Table I shows the results for all the datasets used. Our algorithm performs reasonably well for all datasets, giving values very close to zero. This indicates that the changes observed across the output distributions are approximately similar to those observed across the input distributions.

IV. Conclusion and Future Work

This paper focuses on computing PCA in real-time. We proposed a novel approach to compute the eigenspace based on a sliding window model in Algorithm 1 (RPCA). Then we used Spectral Diagram and Bhattacharyya coefficient to experimentally verify that our algorithm **successfully** reduces

TABLE I
Datasets used and Results obtained

Dataset Name, Characteristics, Attribute Properties	Dataset Properties (n, d, distributions, Change Point)	Parameters (k, l)	Spectral Diagram	Mean BC Value	Std. dev. BC Value	Mean Diff. betn. change in BC value
Synthetic, Multivariate, Real	20000, 20, 10, 2000	100, 5		0.8648	0.0530	0.092350
Wholesale Customers [15], Multivariate, Integer	440, 8, 5, (197, 274, 294, 341)	20, 3		0.8023	0.1157	0.054558
Wireless Indoor Localization [16] [17], Multivariate, Real	2000, 7, 4, 500	30, 3		0.8852	0.0762	0.052322
Wine [18], Multivariate, (Integer, Real)	178, 13, 3, (60, 131)	20, 6		0.9244	0.0788	0.135342

data dimensionality in real-time in accordance with the **current** distribution, besides preserving the **major** components of the **previous** distributions. This research has also opened up a number of interesting problems like how to smartly estimate initial window size, how to dynamically adjust window size and how to apply various matrix sketching methods over the sliding window [20] in a real-time system.

V. Acknowledgements

This research is partially supported by ICT Division Innovation Fund, Government of Peoples' Republic of Bangladesh.

References

[1] T. Sakaki, M. Okazaki, and Y. Matsuo, "Earthquake shakes twitter users: real-time event detection by social sensors," in *Proceedings of the 19th international conference on World wide web*. ACM, 2010, pp. 851–860.

[2] S. E. Middleton, L. Middleton, and S. Modafferi, "Real-time crisis mapping of natural disasters using social media," *IEEE Intelligent Systems*, vol. 29, no. 2, pp. 9–17, 2014.

[3] Q. Li, A. Nourbakhsh, S. Shah, and X. Liu, "Real-time novel event detection from social media," in *Data Engineering (ICDE), 2017 IEEE 33rd International Conference on*. IEEE, 2017, pp. 1129–1139.

[4] M. Mathioudakis and N. Koudas, "Twittermonitor: trend detection over the twitter stream," in *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*. ACM, 2010, pp. 1155–1158.

[5] D. Preotiuc-Pietro, S. Samangooei, T. Cohn, N. Gibbins, and M. Niranjan, "Trendminer: An architecture for real time analysis of social media text," 2012.

[6] N. Phan, S. A. Chun, M. Bhole, and J. Geller, "Enabling real-time drug abuse detection in tweets," in *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*. IEEE, 2017, pp. 1510–1514.

[7] K. Meagher, D. Loiselle, and R. Koopman, "Real time microgrid power analytics portal for mission critical power systems," Nov. 27 2012, uS Patent 8,321,194.

[8] W. Lee, S. J. Stolfo, and K. W. Mok, "A data mining framework for building intrusion detection models," in *Security and Privacy, 1999. Proceedings of the 1999 IEEE Symposium on*. IEEE, 1999, pp. 120–132.

[9] G. H. Dunteman, "Principal component analysis. quantitative applications in the social sciences series (vol. 69)," 1989.

[10] L. I. Smith, "A tutorial on principal components analysis," Tech. Rep., 2002.

[11] I. Jolliffe, *Principal component analysis*. Springer, 2011.

[12] 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*. Society for Industrial and Applied Mathematics, 2015, pp. 887–901.

[13] Z. Karnin and E. Liberty, "Online pca with spectral bounds," in *Conference on Learning Theory*, 2015, pp. 1129–1140.

[14] A. Bhattacharyya, "On a measure of divergence between two statistical populations defined by their probability distributions," *Bull. Calcutta Math. Soc.*, vol. 35, pp. 99–109, 1943.

[15] N. G. C. F. M. Abreu *et al.*, "Analise do perfil do cliente recheio e desenvolvimento de um sistema promocional," Ph.D. dissertation, 2011.

[16] R. B. Bhatt and M. Gopal, "Frct: fuzzy-rough classification trees," *Pattern analysis and applications*, vol. 11, no. 1, pp. 73–88, 2008.

[17] J. G. Rohra, B. Perumal, S. J. Narayanan, P. Thakur, and R. B. Bhatt, "User localization in an indoor environment using fuzzy hybrid of particle swarm optimization & gravitational search algorithm with neural networks," in *Proceedings of Sixth International Conference on Soft Computing for Problem Solving*. Springer, 2017, pp. 286–295.

[18] D. Dheeru and E. Karra Taniskidou, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[19] G. C. (originator). Bhattacharyya distance. [Online]. Available: https://www.encyclopediaofmath.org/index.php/Bhattacharyya˙distance

[20] Z. Wei, X. Liu, F. Li, S. Shang, X. Du, and J.-R. Wen, "Matrix sketching over sliding windows," in *Proceedings of the 2016 International Conference on Management of Data*. ACM, 2016, pp. 1465–1480.