Proceedings of the 4th International Conference on Signal Processing and Machine Learning
DOI: 10.54254/2755-2721/55/20241429

Efficient computation of eigenvalues in diffusion maps: A multi-strategy approach

Yixiong Fang[1,*], Weixi Yang[2,3]

[1]School of Electronic Information and Electrical Engineering, Shanghai JiaoTong University, Shanghai, 200240, China
[2]Sydney Smart Technology College, Northeastern University, Qinhuangdao, 066004, China

*Corresponding author email: fangyixiong@sjtu.edu.cn
[3]202119293@stu.neu.edu.cn

Abstract. In the pursuit of accelerating the computation of k-largest eigenvalues and eigenvectors, this work presents novel methodologies and insights across three main areas. 1) Speed Arnoldi Iteration Up: By observing existing algorithms, we propose an innovative approach that leverages matrix decomposition to accelerate the computation process. The implementation focuses on iteratively computing orthogonal projections and efficiently storing computed vectors in Krylov subspace. 2) Special Case: Eigenvector for $\lambda = 1$: We examine a specific scenario concerning Markov matrices, where the largest eigenvalue is 1. The work provides a detailed proof and analysis of this property, contributing to a deeper understanding of eigenvalues and eigenvectors in the context of stochastic processes. 3) Gaussian Approximation for Markov Matrices: This section delves into the Gaussian approximation for Markov matrices, denoted by P. The work covers theoretical insights, practical challenges, computational efficiency, and empirical validation, providing a comprehensive exploration of this critical method. Together, these sections form a cohesive study aimed at enhancing the computational efficiency of significant algorithms within the field of dimensionality reduction and matrix analysis. The findings may find broad applications in various domains, including image segmentation, speaker verification, anomaly detection, and more.

Keywords: diffusion map, Arnoldi Iteration, Eigenvector, Eigenvalue, Gaussian Approximation.

1. Introduction

Diffusion maps are a powerful tool in the field of non-linear dimension reduction, enabling the extraction of underlying manifolds within complex data structures. With applications spanning diverse fields such as face recognition, spectral clustering, 3D modeling, anomaly detection, and brain resting state network analysis, diffusion maps have become indispensable in modern computational research. A critical step in the computation of diffusion maps involves finding the k-largest eigenvalues and eigenvectors of a Markov matrix. This task, while fundamental, can become prohibitively time-consuming and computationally expensive, particularly for large matrices and higher values of k. Moreover, in many big data processing algorithms, the need for absolute precision can be relaxed, thus opening up opportunities for optimization

DOI: 10.54254/2755-2721/55/20241429

[1-6].

Our work explores three innovative strategies to accelerate this essential process, each tailored to different matrix sizes and values of k. First, we utilize Singular Value Decomposition (SVD) to transform the problem into a more manageable form, reducing computational complexity without a significant loss of accuracy. Next, we introduce a special case handling mechanism for $\lambda = 1$, leveraging known properties of Markov matrices to streamline the calculation. Lastly, we present a Gaussian approximation method, where by introducing $P^2 = PGG^TP$ with $G \in \mathbb{R}^{n\times k}$, Gaussian distribution, we obtain a close approximation to P^2, especially when k is sufficiently large. This approach leads to more steady results, and through a careful comparison between various approximation methods like PGG^T, PGG^TPGG^TP, we demonstrate that PGG^TP provides optimal results, particularly in applications like diffusion map where exact eigenvalues are not essential.

In summary, our work offers a well-rounded solution to a long-standing challenge in diffusion map computation, striking a balance between speed, accuracy, and reliability.

2. Background knowledge

To further discuss how to speed up the computation, we should first make clear that how diffusion map works and how to compute eigenvalues/eigenvectors in standard library.

2.1. Diffusion map

A diffusion map is a dimensionality reduction technique that can be computed through the following steps:

1) **Construct a Similarity Matrix:** Measure the local affinities between data points using a Gaussian kernel, forming the similarity matrix K.
2) **Build a Normalized Graph Laplacian:** Normalize K by using the degree matrix D, leading to the stochastic matrix $P = D^{-1}K$.
3) **Compute the Diffusion Operator:** Compute the diffusion operator T through further normalization, such as $T = D^{-1/2}PD^{-1/2}$, embedding the manifold's global geometry.
4) **Eigenvalue Decomposition:** Perform eigenvalue decomposition on T, ordering the eigenvectors corresponding to the largest eigenvalues.
5) **Construct the Diffusion Map:** Form the diffusion map by selecting eigenvectors corresponding to the largest non-trivial eigenvalues, scaling them by the respective eigenvalues. These eigenvectors provide the new coordinates that preserve the manifold's geometry.

Among these, the eigen-decomposition step stands out as the most computationally expensive part of the process. This computational cost must be considered, especially when working with large datasets, as it may become the bottleneck in the efficiency of the diffusion map technique.

2.2. Arnoldi iteration

The Arnoldi iteration method is an iterative algorithm used to approximate the top k eigenvalues and eigenvectors of a given matrix, especially useful for large and sparse matrices like a Markov matrix. The method consists of the following steps:

1) **Initialization:** Select an arbitrary unit vector v_1 to initialize the process.
2) **Iterative Process:** For $j = 1,2,\dots,k$:
   – Compute $w = Av_j$.
   – Orthogonalize w against previous vectors $v_1, v_2, \dots, v_j$ through the Gram-Schmidt process.
   – Store the coefficients of the orthogonalization in a matrix H.
   – Normalize w to obtain the next vector v_{j+1}, and add it to the growing Krylov subspace.
3) **Form Hessenberg Matrix:** Form the Hessenberg matrix H_k from the stored coefficients.
4) **Compute Eigenvalues and Eigenvectors:** Solve the eigenvalue problem for H_k to obtain the approximations for the eigenvalues and eigenvectors of the original matrix A. The eigenvectors of H_k can be transformed to approximate the eigenvectors of A using the Krylov subspace vectors.

DOI: 10.54254/2755-2721/55/20241429

The Arnoldi iteration provides an efficient means to compute a subset of the spectrum of a matrix, encapsulating the vital characteristics needed for various applications.

2.3. Singular value decomposition

Singular value decomposition (SVD) is a mathematical method that factorizes a complex matrix and generates the eigen decomposition of a square normal matrix[7]. It provides a set of orthogonal eigenvectors (i.e. eigen basis) for any non-square matrix.

A complex matrix $A \in \mathbb{R}^{m\times n} = U\Sigma V^T$ is decomposed into the product of three matrices, where U (left-sigular matrix) and V (right-singular matrix) are unitary matrices. Under this case V^T is the conjugate transpose of V while Σ is a rectangular diagonal matrix with non-negative real numbers. For any complex matrix, this method always exists, providing a solid backup for many well-known applications under general scenarios.

2.4. Markov matrix

A Markov matrix is a square matrix where each element describes the probability of transitioning from one state to another in a Markov chain. Denote $\mathcal{S} = \{1,2,\dots,m\}$ a finite state set over perticular space with cardinality m. The transitional probability within one time step is $P(X_t = j|X_{t-1} = i) = p_{i,j}$ with property $\Sigma_{j=1}^{m} p_{i,j} = 1$[7].

$$P = \begin{bmatrix} p_{1,1} & p_{1,2} & \cdots & p_{1,m} \\ p_{2,1} & p_{2,2} & \cdots & p_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ p_{m,1} & p_{m,2} & \cdots & p_{m,m} \end{bmatrix}$$

Pray tribute to the Gershgorin Circle Theorem, for a Markov matrix, all eigenvalues have absolute values that are less than or equal to one[8,9]. Intuitively and evidently, the eigenvalue 1 corresponds to $1 \in \mathbb{R}^{m\times 1}$ (i.e. an eigenvector of all ones). This special property is utilized in Section. 5 to significantly improve the algorithmic complexity under specific cases.

3. Speed Arnoldi iteration up

3.1. implementation

Observing existing algorithms, we discover that before iteratively computing orthogonal projection, a new candidate vector should be created to store computed vectors in Krylov subspace, which is directly obtained by performing dot product between the initial matrix A and a candidate column vector, resulting in a computational complexity of $O(n^2)$.

To accelerate the computation process, we draw inspiration from matrix decomposition and propose the following steps to speed up the operation:

- **Step 1: Run SVD fraction on A**

First, we perform Singular Value Decomposition (SVD) on target matrix A, denoted $A = U\Sigma V^T$. In a more general context, we consider $A \in \mathbb{R}^{m\times n}$ as a non-square matrix where $U \in \mathbb{R}^{m\times m}$, $\Sigma \in \mathbb{R}^{m\times n}$, and $V \in \mathbb{R}^{n\times n}$.

- **Step 2: Maintain top l singular values to obtain fraction A_l**

With the goal of solving the eigenvalue and eigenvector problem of diffusion maps, we may reduce the requirement for complete accuracy by retaining the top l singular values and their corresponding unitary matrices. These capture the highest degree of essential information.

From these elements, an l-dimensional sub-matrix can be generated, denoted $A_l = U_l\Sigma_l V_l^T$. Subsequent computations use this A_l as an approximation of the original target matrix A, accepting a certain level of accuracy loss.

- **Step 3: Decompose into Two Submatrix Calculations**

In this step, the traditional computation of Ax (with $A \in \mathbb{R}^{n\times n}$ and $x \in \mathbb{R}^{n\times 1}$) would cost $O(n^2)$ time complexity. However, we can make this process more efficient by using the previously computed

DOI: 10.54254/2755-2721/55/20241429

V_l and A_l. Specifically, we calculate $V_l^T x$ first, and then add $A_l \Sigma_l$, thereby reducing the time complexity to $O(lN)$.

This optimization leverages the fact that we are working with the top l singular values and corresponding vectors, and takes advantage of the properties of SVD to create a more computationally efficient approach.

3.2. Limitation

Given our focus on specific matrices, we can decompose the calculations into two submatrices, allowing for more manageable computations.

Condition: Suitability for Specific Matrices

This method is particularly tailored to matrices with pronounced differences in the top k eigenvalues. It's most effective when the largest k eigenvalues are significantly greater than the rest.

Condition: Size of l

A key requirement for this method is that the value of l must be at least 2 times smaller than n. In testing, this has proven to yield very accurate results even when l is small, provided the top k eigenvalues are substantially large. It underlines the applicability of the method to matrices where a precise relationship between l and n is maintained.

These conditions pinpoint the scenarios where the method is applicable and effective, and underscore its limitations with respect to matrices that do not meet these criteria.

3.3. Alignment with Diffusion Maps

The nature of the diffusion map is such that it aims to extract intrinsic geometric properties of the original data. This essentially boils down to capturing the structure and relationships embedded within the data, often represented by the manifold on which the data resides.

In this context, the large eigenvalues of the matrix are of primary interest. These eigenvalues represent the most significant directions or dimensions in which the data varies, and encapsulating these directions can reveal the underlying geometry of the data.

The method proposed earlier, focusing on matrices with significantly large top k eigenvalues, is therefore particularly well-suited for diffusion maps. By concentrating on the major eigenvalues and the corresponding eigenvectors, the method aligns with the goal of diffusion maps, which is to capture the essential characteristics of the original data.

Therefore, the method presents a strategic fit for the specific requirements of diffusion maps, making it a promising approach for applications that prioritize the extraction of fundamental properties from data while only caring about the large eigenvalues.

3.4. Accuracy Test

In order to validate the optimization method proposed earlier, we conducted a comprehensive accuracy test with varying matrix sizes n. The focus of the test was to compare the top 3 eigenvectors of both matrices A and A_l, particularly for a constructed matrix having 3 large top eigenvalues. Below are the detailed steps and logic behind the test:

1) **Varying n:** The test was performed for different values of n to ascertain the method's adaptability and performance across various matrix sizes.
2) **Matrix Construction:** A matrix was constructed with 3 pronouncedly large top eigenvalues to create a clear distinction between the top and the rest of the eigenvalues.
3) **Eigenvector Comparison:** The top 3 eigenvectors of both A and A_l were extracted. Since A_l is derived from A by maintaining only the top l singular values, this comparison aims to assess how closely A_l approximates the significant characteristics of A.
4) **Accuracy Metric:** The absolute distance was chosen as the metric for accuracy, providing a quantitative measure of the difference between the eigenvectors.

DOI: 10.54254/2755-2721/55/20241429

5) **Varying l:** To gauge the robustness of the method and its sensitivity to the choice of l, the test was performed for different values of l. This enabled an understanding of how the size of l affects the approximation.

6) **Results:** The test revealed that the fraction does not affect the result as A_l shares the same eigenvalue as A, and provided insights into the optimal choice of l that balances efficiency and accuracy.

This test affirms the method's capability to efficiently approximate matrices with large top eigenvalues while maintaining accuracy, and provides guidelines for choosing the size of l based on specific requirements and the matrix size n.

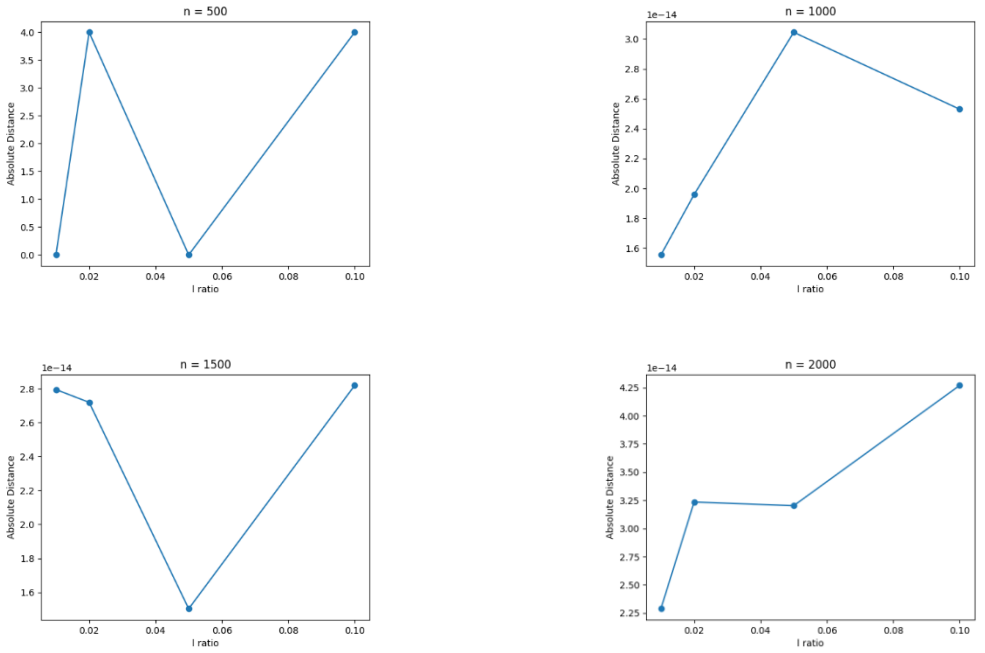

Figure 1. Accuracy test results for varying values of n: (top left) $n = 500$, (top right) $n = 1000$, (bottom left) $n = 1500$, (bottom right) $n = 2000$.

The result shows that l only needs to be a little bit larger than k to maintain the accuracy.

3.5. Speed Test

Assessing the computational efficiency of the proposed method is crucial for its practical application. While rewriting the standard library would be overly complex, we opted to gauge the speed of the Arnoldi iteration as a proxy for evaluating the method's efficiency. A benchmark was conducted using a fixed iteration time (e.g., 500 iterations) as the standard. Below are the details of the speed test:

1) **Test Parameters:** The test was conducted for different values of n (500, 600, 700, 800) and l (0.1n, 0.05n, 0.02n, 0.01n), thereby offering a comprehensive analysis of the method's performance across various scenarios.

2) **Arnoldi Iteration:** The Arnoldi iteration was selected as the core computational step for the speed test. With 500 iterations as the benchmark, the test provides insights into how the choice of l impacts the time complexity.

3) **Exclusion of SVD Time:** It's worth noting that the SVD time was not considered in the speed test, as a highly efficient SVD algorithm was assumed to be available, minimizing its contribution to the overall computation time.

4) **Results and Implications:** The speed test results provide valuable information on the scalability and efficiency of the method, especially in the context of varying the size of n and l. The findings

DOI: 10.54254/2755-2721/55/20241429

support the method's applicability for large-scale problems, where computational efficiency is paramount.

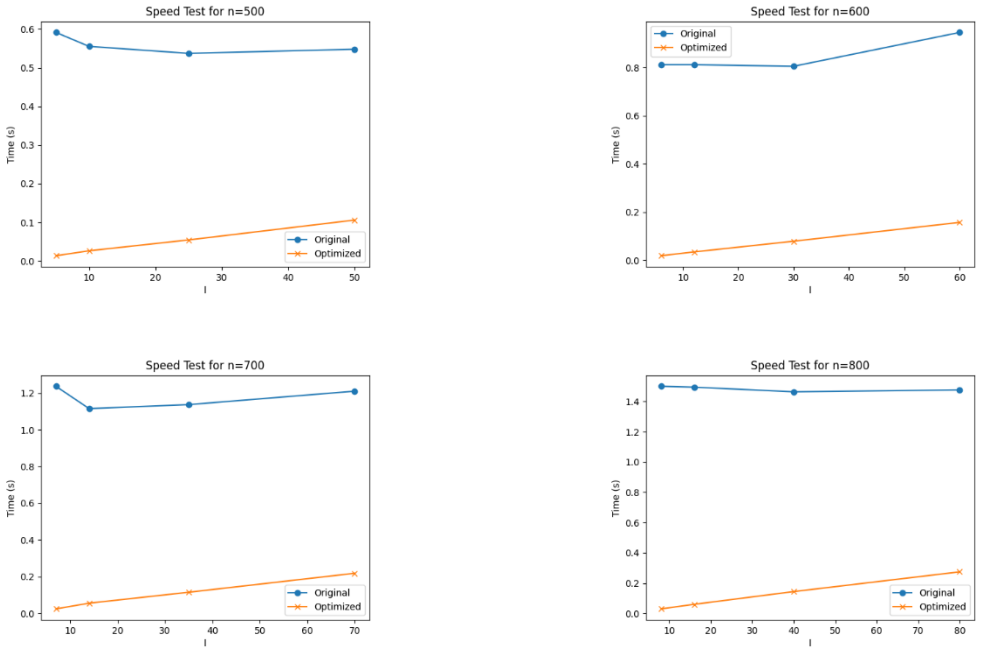

Figure 2. Speed test results for varying values of n: (top left) $n = 500$, (top right) $n = 600$, (bottom left) $n = 700$, (bottom right) $n = 800$.

4. Special Case: Eigenvector for $\lambda = 1$

4.1. Proof

Consider a Markov matrix **P**, where each row sums to one. This matrix represents a stochastic process and has some interesting properties regarding its eigenvalues and eigenvectors. Specifically, the largest eigenvalue will always be $\lambda = 1$, and the corresponding eigenvector will have all entries equal to 1.

The proof of this property is straightforward. Since each row of A sums to one, we can write the matrix equation as:

$$\begin{aligned}\mathbf{P}x &= \begin{bmatrix} p_{1,1} & p_{1,2} & \cdots & p_{1,n} \\ p_{2,1} & p_{2,2} & \cdots & p_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ p_{n,1} & p_{n,2} & \cdots & p_{n,n} \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} p_{1,1} + p_{1,2} + \cdots + p_{1,n} \\ p_{2,1} + p_{2,2} + \cdots + p_{2,n} \\ \vdots \\ p_{n,1} + p_{n,2} + \cdots + p_{n,n} \end{bmatrix} \\ &= \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} = x.\end{aligned}$$

Hence, the matrix A has an eigenvalue $\lambda = 1$, and the corresponding eigenvector is a vector where all elements are 1. This property can be leveraged in various applications, especially when dealing with stochastic processes.

DOI: 10.54254/2755-2721/55/20241429

As the largest value of Markov matrix is also 1[7], we can simply construct it in constant time. However, the original implementation of diffusion map doesn't use this property.

4.2. Optimization Test Results

The optimization of the algorithm was tested for the special case where $k = 1$ using a random Markov matrix, where every row's sum is 1, and every element is $0 < a < 1$. The test was conducted for various values of n, such as 1000, 1500, 2000, 2500, 3000, and 3500.

4.3. Results

The optimization led to a remarkable improvement in the speed of computation. In fact, for the tested special case, the optimization resulted in the operation taking virtually no time.

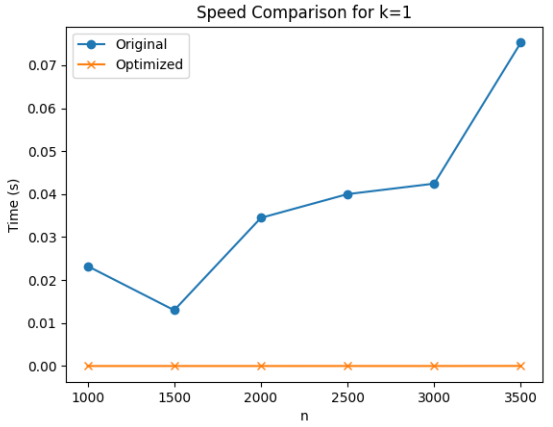

Figure 3. Speed comparison between original and optimized implementations for different values of n.

The figure above illustrates the time comparison between the original implementation and the optimized version for various n values. It can be seen that the optimized code provides a significant reduction in computational time, making it an effective choice for handling the given special case.

5. Gaussian Approximation for Markov Matrices

In this section, we describe the Gaussian approximation for Markov matrices denoted by P, elaborating on the theoretical insights, practical challenges, computational efficiency, and empirical validation of this method.

5.1. Background and Theory

Markov matrices, with the property that every row sums to 1 and each element $0 < a < 1$, play a crucial role in stochastic processes, big data, and artificial intelligence. Our primary focus is on approximating P^2 using Gaussian distribution, a critical advancement for efficient computation.

5.2. Construction of Gaussian Approximation

Consider a matrix $G \in \mathbb{R}^{n\times k}$ where entries in each column follow an i.i.d Gaussian Distribution. We introduce PGG^TP as an approximation for P.

Below we prove the validity of this Gaussian matrix product approximation method, i.e. to prove $\mathbf{E}(PGG^TP) = \mathbf{E}(P)$, where $\mathbf{E}(\cdot)$ calculates the expectation.

Proof.

Let $g_{i,j}$ be each entry in G with a constant k.

DOI: 10.54254/2755-2721/55/20241429

$$G = \frac{1}{\sqrt{k}} \begin{bmatrix} g_{1,1} & g_{1,2} & \cdots & g_{1,k} \\ g_{2,1} & g_{2,2} & \cdots & g_{2,k} \\ \vdots & \vdots & \ddots & \vdots \\ g_{n,1} & g_{n,2} & \cdots & g_{n,k} \end{bmatrix}$$

For a give matrix P, the relationship between matrix expectations $\mathbf{E}(PGG^TP) = P\mathbf{E}(GG^T)P$ holds. By matrix calculation, we obtain a diagonal matrix.

$$\mathbf{E}(GG^T) = \frac{1}{k}\mathbf{E}\begin{bmatrix} \sum_{j=1}^{k} g_{1,j}^2 & 0 & \cdots & 0 \\ 0 & \sum_{j=1}^{k} g_{2,j}^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sum_{j=1}^{k} g_{n,j}^2 \end{bmatrix}$$

Since entries in each column are variables drawn from i.i.d $\mathcal{N}(0,1)$, $Z = \sum_{j=1}^{k} g_{i,j}^2 \sim \chi^2(k)$ with degrees of freedom k. By properties from Chi-square Distribution, $\mathbf{E}(\chi^2(k)) = k$. Hence, we prove that the Gaussian matrix product approximation method is valid.

$$\mathbf{E}(GG^T) = \frac{1}{k}\begin{bmatrix} k & 0 & \cdots & 0 \\ 0 & k & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & k \end{bmatrix} = I$$

Worthy of notice, in application, when k is sufficiently large, this method suits best.

5.3. Computational Efficiency

For $k \geq 8$(mention that k here means top-k eigenvalues/vectors), direct computation of the top k eigenvalues become slow and unstable. This situation is particularly cumbersome for big data applications where even a single matrix might take up to 110 seconds, making it impractical.

The proposed Gaussian approximation method overcomes this limitation, adding immense value to the field by offering a robust and efficient alternative.

5.4. Exploration of Approximation Methods

The choice of using the Gaussian approximation PGG^TP was not made lightly but was the result of comprehensive exploration and comparison with other methods[7,10]. This section details the various approximations we examined, the unique challenges, and the reasoning behind our final selection.

5.5. Choosing the Right Approximation

Given a Markov matrix P, numerous approaches were investigated for approximating P^2. The methods included:

- PGG^T
- PGG^TPGG^TP

Each approach presented unique characteristics, especially concerning the ordering of eigenvalues. For instance, when approximating with PGG^TP, the eigenvalue would be squared, altering their ordering (e.g. $(-0.5)^2 = 0.25$ and $(-0.1)^2 = 0.01$, changing the sequence of correctly ordered eigenvalues).

5.6. Visualized Experimentation

Through extensive experimentation and diffusion map's visualized experiments, it became evident that PGG^TP offered the best results. It extracted the most salient features using the largest absolute eigenvalues and proved more stable in execution.

DOI: 10.54254/2755-2721/55/20241429

The diffusion map's requirements further supported this choice, as it focuses only on the real part of the eigenvalues and eigenvectors and does not demand high precision.

Figure 4 illustrates the steady performance of this method, especially when computing the top 10 eigenvalues and eigenvectors in a 2000×2000 matrix.

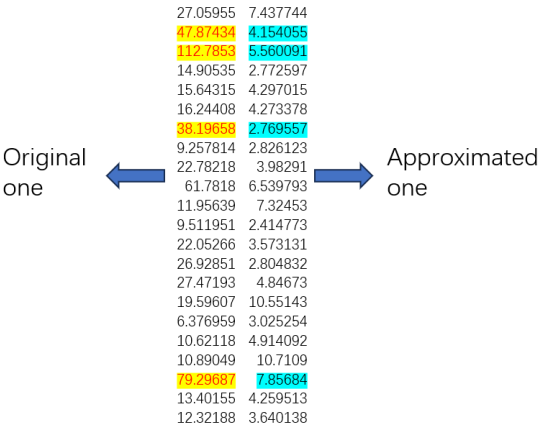

Figure 4. Steady performance of the Gaussian approximation method.

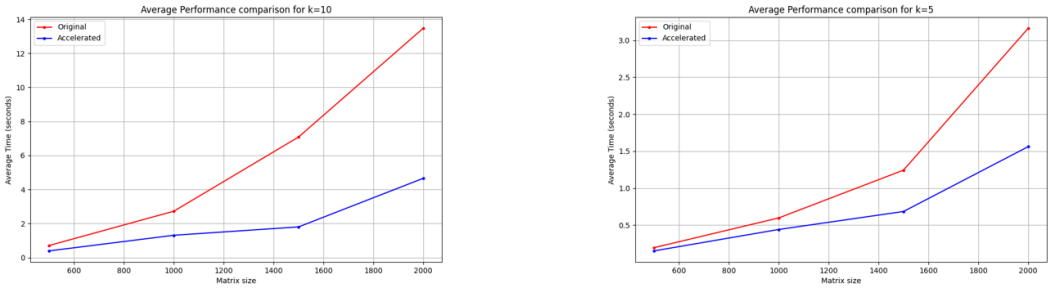

Figure 5. Performance comparison for $k = 5$ (left) and $k = 10$ (right).

5.7. Conclusion on Method Selection

Our exploration revealed that the Gaussian approximation PGG^TP, when k is sufficiently large, brings the matrix close to P^2. It emerged as the optimal solution for its ability to preserve eigenvalue order, stability in computation, and alignment with diffusion map requirements.

The resulting method stands as a practical and theoretically sound solution for the challenges inherent in large-scale computations, fulfilling the essential criteria for successful application in big data and artificial intelligence.

5.8. Discussion

This paper has presented three strategies to accelerate the computation of the k-largest eigenvalues and eigenvectors of a Markov matrix, focusing on the context of Diffusion Maps. Through careful consideration of the matrix size and the value of k, we have achieved more steady and faster computation. Despite some limitations, these methods offer promising directions for ongoing innovation in large-scale data processing and AI applications.

6. Conclusion

This paper has presented three strategies to accelerate the computation of the k-largest eigenvalues and eigenvectors of a Markov matrix, focusing on the context of Diffusion Maps. Through careful

DOI: 10.54254/2755-2721/55/20241429

consideration of the matrix size and the value of k, we have achieved more steady and faster computation. Despite some limitations, these methods offer promising directions for ongoing innovation in large-scale data processing and AI applications.

7. Future Work

The areas for improvement identified in this discussion pave the way for future research. Potential directions include refining the Arnoldi method implementation, exploring additional special cases for further optimization, and balancing the trade-off between speed and accuracy in the Gaussian approximation. Our work serves as a foundational step in accelerating Diffusion Map computations, but there remains ample opportunity to push the boundaries of efficiency and accuracy further.

References

[1] Agrawal, R., Wu, C.-H., Grosky, W., & Fotouhi, F. (2006). Diffusion maps-based image clustering. ACM Transactions on Multimedia Computing, Communications, and Applications, 2(4), 9.

[2] Banerjee, S., & Roy, A. (2014). *Linear Algebra and Matrix Analysis for Statistics* (1st ed.). Chapman & Hall.

[3] Coifman, R. R., Lafon, S., Lee, A. B., Maggioni, M., Nadler, B., Warner, F., & Zucker, S. W. (2005). Geometric diffusions as a tool for harmonic analysis and structure definition of data: Diffusion maps.

[4] Li, B., Zhang, Y., & He, D. (2016). A survey of big data architectures and machine learning algorithms in healthcare. *Journal of King Saud University-Computer and Information Sciences.*

[5] Mishne, G., & Cohen, I. (2013). Multiscale anomaly detection using diffusion maps. *IEEE Journal of Selected Topics in Signal Processing, 7*(1), 111-123.

[6] Thorstensen, N., Étyngier, P., Ségonne, F., & Keriven, R. (2011). Diffusion maps as a framework for shape modeling. *Computer Vision and Image Understanding*, *115*(4), 520-530

[7] Nadler, B., Lafon, S., Coifman, R. R., & Kevrekidis, I. G. (2005). Diffusion Maps, Spectral Clustering and Eigenfunctions of Fokker–Planck Operators.

[8] Kapralov, M., Potluru, V. K., & Woodruff, D. P. (2016). How to Fake Multiply by a Gaussian Matrix. *Proceedings of the 33rd International Conference on Machine Learning, ICML 2016*, 2101-2110.

[9] Cohen, M. B., Nelson, J., & Woodruff, D. P. (2016). Optimal approximate matrix product in terms of stable rank. Submitted on July 8, 2015 (v1), last revised on March 2, 2016 (this version, v3).

[10] Barkan, O., & Aronowitz, H. (2013). Diffusion maps for PLDA-based speaker verification.