Published in final edited form as:

Proc IEEE Int Conf Big Data. 2019 December ; 2019: 74–83. doi:10.1109/BigData47090.2019.9006512.

A Streaming model for Generalized Rayleigh with extension to Minimum Noise Fraction

Soumyajit Gupta,

Dept. of Computer Science, University of Texas at Austin, Austin, TX, USA

Chandrajit Bajaj

Dept. of Computer Science and Oden Institute, University of Texas, Austin, Austin, TX, USA

Abstract

The Rayleigh quotient optimization is the maximization of a rational function, or a max-min problem, with simultaneous maximization of the numerator function and minimization of the denominator function. Here, we describe a low-rank, streaming solution for Rayleigh quotient optimization applicable for big-data scenarios where the data matrix is too large to be fully loaded into main memory. We apply this for a maximization of the Signal to Noise ratio of big-data, of very large static and dynamic data. Our implementation is shown to achieve faster processing time compared to a standard data read into memory. We demonstrate the trade-offs with synthetic and real data, on different scales to validate the approach in terms of accuracy, speed and storage.

Keywords

Generalized Rayleigh Quotient; Streaming; Low-Rank Projection; Hyperspectral Image (HSI); Minimum Noise Fraction (MNF)

I. INTRODUCTION

Due to the massive increase in the scale of data collections, it has become impossible to load the entire raw data matrix of such collections into main memory, let alone do algebraic manipulations on them. To combat this growing problem, there has been considerable work on using low-rank approximations of the data domain [1], [2], constructed using streaming models of the Singular Value Decomposition (SVD). All these models process streaming data, where the computer receives one column of the large matrix at a time, and saves a low rank projection of it in memory. This allows the user to generate a small sketch of the data in main memory for all subsequent manipulations on it. These sketched models utilize the guarantee that the projection maps satisfy RIP like properties [3]–[5], thereby approximately maintaining distances (inner products) between samples from the high-dimensional space. The SVD computed on this much smaller sized sketched matrix is thus still able to recover the singular values and singular vectors of the original big data matrix, within some bounded error.

When working in such big data settings, there is however still the need to optimally set various hyperparameters of such sketching algorithms, for each particular limited memory environment system. The size or chunk of the data matrix to be streamed and progressively processed is dependent on the processing machine's memory capacity. Reading too many samples or very large vectors, but keeping only severely lowdim projections of them because of memory constraints will diminish the expected accuracy. Optimal memoryaccuracy-speed trade-offs need to be achieved during stream processing of large scale big-data in limited memory settings. Given a fixed chunk of data, the way it is loaded into memory also proves to be a bottleneck in subsequent processing. These issues have not been comprehensively addressed in prior work, and is the focus of this paper in an attempt to achieve practical stream processing algorithms for big-data applications.

One such example application is of a push broom scanner [6] which is currently used for recording spectroscopic data. The scanning pattern for each frequency over the imaging sample is in generated in an old raster scan fashion (all spectra are acquired within a single row, before moving to the next row and the next etc). The data recording time depends on the application and the physical dimensions of the sample being imaged for each frequency of such hyper-spectral imaging (HSI). Once the data is collected, it is stored in various formats in disk, and subsequently accessed for further processing. Such delays in data collection, and saving is a major bottleneck in HSI imaging applications, specially in surveillance or cancer diagnosis, where near real-time results are expected to follow up on subsequent steps.

Considering the issues described above, we present a fast unified solution of the generalized Rayleigh quotient optimization, and extend it to a Sketchy version of Minimum Noise Fraction (MNF) [7]. This streaming (sketchy) version is able to perform MNF denoising on extremely large HSI with reduced computational complexity and storage but maintaining similar accuracy. We show that the Sketchy version allows us to process scalably large data matrices, specially in a big data setting for Geostationary and Microscopy data. We also show that a streaming scheme of reading data is able to nullify the bottleneck of data reading time into memory. Not only does it allow us to work with reduced sketches in memory, but also allows the method to be integrated into sensors that uses push-broom protocols of HSI data acquisition. This directly helps with the fast denoising of the input data. keep on accumulating low dimensional sketches of the data directly from the sensors, so that is ready to be deployed for further image analysis applications. We attempt tp automate the process further, by providing rules of choosing the input parameter *w.r.t.* the SNR spread across in the data. Additionally, we show that our framework can be easily extended to solve any other problems which can be expressed in a generalized Rayleigh Quotient form.

The main contributions of the paper are as follows:

- 1) Streaming framework for generalized Rayleigh quotient with applications to Minimum Noise Fraction (MNF) for HSI denoising.
- 2) Parameter estimation for setting up low-rank projections for limited-memory machines.
- 3) Accuracy vs. memory vs. time trade-off as a function of the parameter set in (2).

- 4) We share our source code¹ and a large Fourier Transform InfraRed dataset with unknown spectral-correlated noise for public use.
- 5) The model can be integrated with the sensor and thereby allows read time data collection and denoising, thereby an end-to-end framework for BIG data pre-processing before analysis is done.

II. PRIOR WORK

Random projection forms the foundation of sketching based algorithms where the goal is to form a sketch of the original matrix that preserves important properties of it. One of the earliest works were for Gaussian Random Projection matrices based on the Johnson-Lindenstrauss [8] lemma and Subsampled Randomized Hadamard Transform (SRHT) [4]. Both of these methods involve loading the entire data into memory, forming the projection matrix and then computing the sketch. Count Sketch based projection [5] overcomes this by directly computing the sketch without loading the entire data.

The core idea of randomized decomposition approaches is to make one or two passes over the data and compute efficient sketches. They can be broadly categorized into four main branches: Sampling based methods (Subset Selection [9] and Randomized CUR [10]), Random Projection based QR [3], Randomized SVD [3] and Nystrom Methods [11]. The sketches can represent any combination of row space, column space or the space generated by the intersection of rows and columns (core space). However, they are limited by their need to see the entire data in memory.

Due to restrictions of size on big data and localized storage cost, streaming algorithms are used to achieve the decomposition of the data under low-rank assumptions. They rely heavily on sketching based approximations. There exists a broad set of sketch based streaming algorithms [3], [4], [12]–[14] which involves generating the combination of the mentioned projection spaces. The most recent one is by Tropp [2] which provides practical approximations compared to the earlier works.

It was shown by Green *et al.* [7] that the variance of HSIIs does not necessarily reflect the real SNR, due to unequal noise variances in different channels, with noise variance dominating the signal variance in some bands. They developed the Minimum Noise Fraction (MNF) transform based on maximization of SNR, so that the transformed principal components are ranked by SNR rather than variance as used in PCA. MNF does not impose any structure on the noise model and works for unknown noise models as long as the noise covariance estimation is good. However, it is limited by the computation cost and a manual component of selecting the number of components to retain in the MNF transformed space. Recently, it was showed in [15] that the same can be fully automated at an order lower of cost reduction both in processing and storage time.

III. PROBLEM STATEMENT

The given input matrix is $A \in \mathbb{R}^{m \times n}$, where m , n represents the number of samples and features respectively. Its feature covariance $\Sigma_A \in \mathbb{R}^{n \times n}$ is the summation of two independent

covariances $\Sigma_B, \Sigma_C \in \mathbb{R}^{n \times n}$ where $\Sigma_A = \Sigma_B + \Sigma_C$. The problem of Generalized Rayleigh Quotient is to solve max-min function of these two variables (Σ_B, Σ_C) , via some projection vectors ξ . Attaining maximum value of the function involves maximizing the ratio which is effectively maximizing the numerator and minimizing the denominator simultaneously in Eq. 1. It is known that $\text{rank}(\Sigma_B) = r$ is extremely low rank and $\text{rank}(\Sigma_A) = k$ is low rank, where $r \ll k \ll n < m$. We look for Rayleigh vectors $(\xi_1, \dots, \xi_r)$ which captures the first r components of Σ_B such that $(\lambda_{B(1)}/\lambda_{C(1)}) \quad (\lambda_{B(2)}/\lambda_{C(2)}) \quad \dots \quad (\lambda_{B(r)}/\lambda_{C(r)})$.

$$\xi^* = \underset{\xi}{\operatorname{argmax}} \frac{\xi^T \Sigma_B \xi}{\xi^T \Sigma_C \xi} \quad (1)$$

This is different from the Rayleigh Quotient which maximizes a single covariance in the numerator:

$$\xi^* = \underset{\xi}{\operatorname{argmax}} \frac{\xi^T \Sigma_A \xi}{\xi^T \xi} \quad (2)$$

In the denoising setting of Minimum Noise Fraction, $\Sigma_A, \Sigma_B, \Sigma_C$ are the covariance matrix for Data, Signal and Noise $\Sigma_D, \Sigma_S, \Sigma_N$ respectively. There are three major challenges in this setting:

- 1) Need to process data matrix A which is too large to be completely loaded into memory
- 2) For low k rank projections, compute k based on memory limitations.
- 3) With low rank projections of matrix A , we need to accurately compute Σ_B, Σ_C

We handle the above issues it by reading the data in a streaming fashion (one column at a time) and keeping low dimensional sketches of it in memory. Sketchy SVD [2] allows for such a framework, but needs a parameter “ K ” as input, which acts as a intermediate rank surrogate. For problem 2 of choosing an optimal value of k , we show through experiments that it can be set to our guess of $\text{rank}(A)$ and some memory budget mapping, given the size of the full data. For problem 3, we show that given these sketches, we can still compute the covariances approximately such that the top r components of ξ are preserved. The streaming model speeds up the data loading pipeline immensely.

IV. SOLUTION STRATEGY FOR MINIMIZATION OF GENERALIZED RAYLEIGH QUOTIENT

For real data matrix A , the covariance Σ_A is never full rank. Let us assume that $\text{rank}(\Sigma_A) = k$, where $k \ll n < m$. Also, $\text{rank}(\Sigma_B) = r$ is extremely low rank, thus $r \ll k$. From algebra, we know that if $\Sigma_A = \Sigma_B + \Sigma_C$, then $\text{rank}(\Sigma_A) \leq \text{rank}(\Sigma_B) + \text{rank}(\Sigma_C)$. So $\text{rank}(\Sigma_C) \geq k - r$.

We can tune Eq. 1 to a slightly different form to help with the optimization as follows:

$$\xi^* = \underset{\xi}{\operatorname{argmax}} \frac{\xi^T \Sigma_B \xi}{\xi^T \Sigma_C \xi} \approx \left(\frac{\xi^T \Sigma_B \xi}{\xi^T \Sigma_C \xi} + 1 \right) = \frac{\xi^T \Sigma_A \xi}{\xi^T \Sigma_C \xi} \quad (3)$$

A. Approximate Solution

Following the above formulation, we can solve the original problem in a setting when Σ_B is unknown. We can frame the objective function of Eq. 1 as follows, with $\operatorname{tr}()$ representing the trace of a matrix.

$$\begin{aligned} \Xi^* &= \underset{\Xi}{\operatorname{argmax}} \frac{\operatorname{tr}(\Xi^T \Sigma_B \Xi)}{\operatorname{tr}(\Xi^T \Sigma_C \Xi)} \\ &= \underset{\Xi}{\operatorname{argmax}} \frac{\operatorname{tr}(\Xi^T U_{B(r)} \Lambda_{B(r)} U_{B(r)}^T \Xi)}{\operatorname{tr}(\Xi^T U_{C(k)} \Lambda_{C(k)} U_{C(k)}^T \Xi)} \end{aligned} \quad (4)$$

We ensure that $(\lambda_{A(1)}/\lambda_{C(1)}) \ (\lambda_{A(2)}/\lambda_{C(2)}) \ \dots \ (\lambda_{A(r)}/\lambda_{C(r)})$ are the eigenvalues corresponding to the Generalized Rayleigh vectors $\xi_1, \dots, \xi_r$. The working criteria of this formulation is that its robust to the strength of λ_C , as long as the ratio of λ_B/λ_C is maintained above a certain threshold, guided by the application. For MNF, r corresponds to the top signal components with SNR above a threshold (5.0).

B. Algorithm

Algorithm 1

Streaming Projection

Input: $A \in \mathbb{R}^{m \times n}$, k : expected overall (surrogate) rank

Output: $\hat{A} \in \mathbb{R}^{m \times k}$ the surrogate rank k -dim data

- 1: Initialize $s = 2k + 1$ ▷ Oversampling parameter
- 2: Projection maps: $\Upsilon \in \mathbb{R}^{k \times m}$, $\Omega \in \mathbb{R}^{k \times n}$, $\Phi \in \mathbb{R}^{s \times m}$, $\Psi \in \mathbb{R}^{s \times n}$
- 3: Projection matrices: $X \in \mathbb{R}^{k \times n}$, $Y \in \mathbb{R}^{m \times k}$, $Z \in \mathbb{R}^{s \times s}$ as empty
- 4: **for** $i = 1 : n$ **do** ▷ Streaming Update
- 5: Form $H \in \mathbb{R}^{m \times n}$ as a sparse empty matrix
- 6: $H(:, i) = A(:, i)$ ▷ Streamed columns
- 7: $X \leftarrow X + \Upsilon H$ ▷ Update Co-Range
- 8: $Y \leftarrow X + H \Omega^T$ ▷ Update Range
- 9: $Z \leftarrow Z + \Phi H \Psi^T$ ▷ Update Core Sketch
- 10: $Q \in \mathbb{R}^{m \times k} \leftarrow \operatorname{qr_econ}(Y)$ ▷ Basis for Range
- 11: $P \in \mathbb{R}^{n \times k} \leftarrow \operatorname{qr_econ}(X^T)$ ▷ Basis for Co-Range
- 12: $C \in \mathbb{R}^{k \times k} \leftarrow ((\Phi Q) \setminus Z) / (\Psi P)$ ▷ Core Matrix
- 13: $\hat{A} \in \mathbb{R}^{m \times k} \leftarrow QC$ ▷ Reconstruct Row space

Ideally, we would want to load the entire data into memory to perform any factorization on it. However, for big data and remote sensing images, the size is too large to do so. Following Alg. 1, we read in data in a streaming fashion and save low dimensional projections of it in memory. The container matrix H defined in **line 5** starts empty each iteration. Being defined in such a sparse manner, although its size is $m \times n$, its effective memory usage is negligible. Only its i -th column gets filled with the streamed column, hence its maximum memory usage is $m \times \mathcal{O}(1)$. This sparse structure helps by acting as a pseudo container for the streamed data.

Over iterations only one column of Y gets updated each iteration in **line 8**, while X, Z have dense updates, incurring $\mathcal{O}(kmn)$ cost on average as $s = k \times \mathcal{O}(1)$. The QR basis vectors are each of rank k since “economy” mode is used taking $\mathcal{O}(k^2m)$, $\mathcal{O}(k^2n)$ for Q, P respectively. The core matrix C is finally computed as a solution of the three subspaces X, Y, Z at cost $\mathcal{O}(k^3 + k^2(m + n))$. As we will be using the data for successive steps, we need to reconstruct the row space as each sample needs to be denoised or classified, incurring cost of $\mathcal{O}(k^2m)$. Thus we end up with $\hat{A}$ which gives us a k -dim representation of the data samples. Since it is shown in Eq. 4 that the data is indeed low k rank, this subspace is enough to approximate the required Rayleigh vectors for the proceeding applications.

Algorithm 2

Generalized Rayleigh

Input: Low column rank data $A \in \mathbb{R}^{m \times k}$

Output: Rayleigh projection vectors $\Xi \in \mathbb{R}^{k \times r}$

- 1: Compute Σ_C and either of Σ_A, Σ_B
 - 2: $[E, \Lambda_C, E^T] = \text{svd}(\Sigma_C, k)$
 - 3: $A_W \in \mathbb{R}^{m \times n} = AE(\Lambda_C^{-1/2})$ ► Whiten data
 - 4: $\Sigma_W \in \mathbb{R}^{k \times k} = \text{Cov}(A_W)$
 - 5: $[U_r, \Lambda_r, U_r^T] = \text{svd}(\Sigma_W, r)$
 - 6: $\hat{\Xi} \in \mathbb{R}^{k \times r} = E * \Sigma_C^{-1/2} * U_r$, ► Forward vectors
 - 7: $\tilde{\Xi} \in \mathbb{R}^{k \times r} = E * \Lambda_C^{1/2} * U_r$ ► Inverse vectors
 - 8: $\tilde{A} = A\tilde{\Xi}\tilde{\Xi}^T$ ► Approximated Data
-

The data $\hat{A} \in \mathbb{R}^{m \times k}$ generated from Alg. 1 is passed to Alg. 2 which computes the Generalized Rayleigh projection vectors $\Xi \in \mathbb{R}^{k \times r}$. Note that, hence in this k -dim space, we are looking for the extremely low r -rank subspace that maximizes the numerator and minimizes the denominator simultaneously. The first step involves computing the covariance of the denominator Σ_C . Since k was chosen as a rank surrogate, it might be greater than the actual rank k^* of Σ_C . We perform an approximate SVD of rank- k using the Block Lanczos [11] method. This saves computation time and space compared to full SVD. Once the eigenvalues Λ_C and eigen vectors E are computed, the data is scaled by them, so that the contribution of the singular values of C along its singular vectors are reduced to unity. This step is also called data whitening transform. This step changes the Generalized Rayleigh

formulation in Eq. 1 to a Rayleigh formulation in Eq. 2. We can therefore look for the extremely low rank- r subspace from this data A_W . We again do a block Lanczos decomposition to compute only the top r components. The eigenvalues in this space represent the top- r eigenvalues of the Generalized Rayleigh solution of Eq. 3. Thus the actual eigenvalues for Eq. 1 are $\Lambda_r - 1$. The eigenvectors of Eq. 1 are given in line 6 for both the forward and inverse transform into the Generalized Rayleigh space and back to the k -dim space.

C. Complexity

For the matrix $A^{m \times n}$ in a big data setting, we can assume that $n < m$ i.e. without loss of generality, the matrix is row heavy. The total cost for Generalized Rayleigh Quotient involves the cost of:

- 1) Computing the covariance matrices in $O(mn^2)$ time and $O(n^2)$ space.
- 2) Computing the eigen decomposition in $O(n^3)$ time and $O(mn)$ space.

Sketchy version involves low rank column space projection of the data followed by estimating Rayleigh vectors.

- 1) Computing the low rank column space projection matrix in $O((m+n)k^2)$ and $O((m+n)k)$ space.
- 2) Computing the covariance matrices in $O(mk^2 + mkr)$ time and $O(k^2 + kr)$ space.
- 3) Computing the eigen decomposition in $O(k^3 + k^2r)$ time and $O(m(k+r))$ space.

A further reduction in resource is achieved as given the k -dim column space in $\hat{A}$ we are computing a rank r subspace from it where $r \ll k$, hence by using the Block Lanczos scheme to compute the subspace prevents computing the full SVD. Thus instead of $O(mk^2 + k^3)$ time and $O(k^2 + mk)$ space, we incur $O(mk^2 + mkr + k^3 + k^2r)$ and $O(k^2 + kr + m(k+r))$.

D. Streaming vs. No streaming

In regular setting, we load the data into memory by pointing to its starting pointer location in the external storage and then sweep the contents in a raster fashion. For HSI data, which are spatio-spectral in nature, the sweep occurs for the entire spatial dimension for one wavelength and then moves to do the same for the next one. This raster order proves to be a bottleneck while loading the data from disk into memory. Streaming raster order for each spectral band completely eliminates this bottleneck, thereby allowing faster processing times.

E. Selection of Streaming parameters

Given a fixed memory budget M , we need to optimize for the amount of data that we can load into memory and process it in one go. Owing to low-dim projection of the data, we can achieve a factor increase in the amount of data being processed. The memory efficiency is:

$$\begin{aligned} \text{Memory} &= \frac{c1 \cdot (mn + n^2)}{c2 \cdot ((m+n)k + k^2 + kr + m(k+r))} \\ &\approx \frac{c1 \cdot (m+n)n}{c2 \cdot (m+n)k} = O(n/k) \end{aligned}$$

Thus, if we want to load more data into memory, then we can keep picking lower values of k . However, the true solution still holds for the optimal k^* in Eq. 4, which is the optimal rank of the unknown matrix Σ_C . Any value of $k < k^*$ will result in decrease in accuracy and slightly noisy reconstruction into the original data space, as we will see in the experimental sections.

V. STREAMING MINIMUM NOISE FRACTION

Although MNF has been fully automated [15], the algorithm still requires a full copy of the data in memory. This is an issue with current HSIs and histopathology slides which are of very large dimension, of the order $10^5 \times 10^5 \times 10^3$. Since the noisy data itself is low rank, we use the matrix $\hat{A}$ from Alg. 1. Because MNF requires an approximation of the noise covariance, this makes sure that we have enough sample to compute a good estimate of Σ_N . Thus, the input to the actual MNF algorithm is a low-dim matrix $\hat{A} \in \mathbb{R}^{m \times k}$, which contains both the high strength signal and noise components. Once we denoise the data through MNF in Alg. 2, we get a clean approximation of the signal $\hat{A} \in \mathbb{R}^{m \times k}$, that still lies in the low k -dimensional space. To reconstruct the data to the original feature space, it is multiplied by P^T , which returns the full dimensional low-rank denoised signal component as $A_c \in \mathbb{R}^{m \times n}$. Since MNF ensures that noise components are reduced to unity, the selection of k should be it contains all the top Signal components and unit variance noise. Thus while projecting back into the original space using P^T , although it contains eigen vectors of both signal and noise, it does not hamper the reconstruction for the signal relevant channels.

VI. EXPERIMENTS

We provide extensive evaluation on real datasets with both synthetic noise (known distribution) and real noise (unknown distribution) added for Minimum Noise Fraction. The experiments are run on MATLAB R2018 on a 64-bit Linux machine with Intel Xeon E5 16 core CPU clocking at 3.20 GHz and 64 GB of RAM. To compare the different approaches, we mainly use three error metrics: Frobenius Norm, Root Mean Square Error (RMSE) and Spectral Angle Mapper (SAM) [16].

A. Hyperspectral data with Synthetic Noise Added

We went with the AVIRIS data, which are geo-spatial data over regions on earth. Since the data comes almost noise free, to show the performance of Sketchy MNF vs. MNF, we add synthetic noise of varying strength to data. For the noise, we set it to zero mean, but having heterogeneous and inter-channel correlated variance.

$$\begin{aligned} N &= \rho n^{-1} \mathcal{N}(0, \Sigma_N) \\ \Sigma_N &= G^T G, G \in \mathbb{R}^{n \times n} \sim \mathcal{N}(0, 1) \end{aligned} \quad (5)$$

where N is a multivariate random matrix and G is a standard normal matrix. In addition to that, to account for the noise strength ρ . We chose the Indian Pines dataset ($(w, h, n) = 145 \times 145 \times 200$) which has both high and low scale spectral features. From the results in Fig. 1 we observe that MNF is immune to the strength of noise added, producing same results for all the three cases, since it has access to the full data. Sketchy MNF also produces equally similar result for low and medium noise setting. For High strength noise, its rank was greater than 30. However, the cutoff was chosen to be 30 for all three settings. Thus for high strength noise, all the spectral features are recovered precisely, while there is some low remnant noise in the non-relevant sections. Table I validates the increase in error with increasing noise, but very low nonetheless showing that SMNF produced results very close to MNF. Fig. 2 shows the time and space comparisons highlighting the benefits.

B. Extremely Noisy Data

Let us consider the AVIRIS data where the noise profile is high strength and does not decay, almost mimicking very high strength white noise. Figs. 3, 4 show the signal and noise profiles, the recovered noise profile via SMNF vs Full MNF and also the Frobenius Norm, RMSE and Spectral Angle Mapper (SAM) between the SMNF vs. MNF recovered signals. We clearly see that even for a high strength flat profile noise, even setting $k = 100 \quad 0.5 \times n (= 200)$ gives perfect reconstruction. We can also go with lower values of k , where the relevant spectral features are still preserved and there is small noise in the non-relevant regions of the spectrum.

C. USAF: Unknown Noise Distribution

We experiment on a subset of the USAF dataset which is of dimension $(128 \times 128 \times 754)$. This results in matrix of size 16384×754 . The noise distribution is unknown here. It is only known that the noise is high strength and correlated between channels. From Fig. 5 it is evident that the signal is very low rank as it falls off very quickly beyond channel 5. The noise although remains at a steady strength until channel 240. Thus if we select our MNF cutoff to be roughly around $SNR = 5.0$, we can stop after picking up the first five bands in the MNF space, which hold true for both MNF and Sketchy MNF. From Fig. 6 we see that selecting $k = 150$ incurs the lowest RMSE and SAM error. Going beyond that, we are taking into account contribution of noise bands that are unity but higher the signal strength on those corresponding channels, which is causing the sharp increase in both metric values at 200. The values later stabilize due to the same fact that noise strength, although unity is still greater than corresponding signal strength. The reconstruction plot shows that using $k = 150$, we are able to recover all the relevant spectral features between channels (0 – 200 and 500 – 600).

D. BR1003: Unknown Noise Distribution

The BR1003 is a breast cancer data collected using Fourier Transform InfraRed (FTIR) spectroscopy on unstained biopsy samples of breast tissue. It is very large (~ 1.4 TB) in size

of dimension $(w, h, n) = (11200 \times 20600 \times 1506)$. Being a real dataset, the noise distribution is unknown to us. The data is stored in Band-Sequential (BSQ) format in which the spatial dimension is traversed in a traditional raster order for each temporal slice. Being in this form, it was ideal for us to demonstrate a practical big data application due to its size and the fact that the column dimension is frequency and each column of the data now represents the unfolded spatial slice which we can stream to the algorithm. Fig. 8 shows the denoising profile. Fig. 9 shows that for working on a data of 6000×6000 spatial slice, the bottleneck of the algorithm is the data loading time. The plot shows how sluggish (672sec $\sim$ 11min) reading in that data for all 1506 spectral dimension can be. Compared to this loading time, this data can be denoised using MNF at around $\sim$ 3min. For large scale data, streaming is a viable option for any randomized algorithm, since they need to look through the data and not keep a full copy of it. We emulated streaming by reading the dataset in spectral chunks and sending reshape spatial vector one at a time. The improvement in acquisition is evident from the figure, where the same data was loaded into memory in 9.97sec, thereby eliminating the bottleneck.

E. Flow data with Synthetic Noise Added

We use a Navier Stokes flow system which generates vorticity patterns for an incompressible fluid under some boundary conditions. The flow is converted from a mesh grid into a matrix format of size $(w, h, n) = (100 \times 50 \times 200)$. For each point in the grid, there are recorded velocity values in both x and y directions for those 200 time instances. The flow has a periodicity associated with it every $\sim$ 60 time step.

For demonstrating the denoising performance, we choose the y-component of the velocity field, reshaping the data as a spatio-temporal matrix $A \in \mathbb{R}^{5000 \times 200}$. Temporally correlated noise is then added to the data following Eq. 5 where $\rho = 0.1$ for high noise setting.

Sketchy MNF is applied on the noisy data with $k = 30$. We can see the spatial profile of a timestamp in Fig. 7 where Sketchy MNF perfectly preserves spatial features in the flow. From Fig. 10 we see that even under a noise free setting $r = 11$ suffices for a low rank representation of the data. The underlying sinusoidal flow is recovered both by MNF and Sketchy MNF, even though a fraction of the feature space (30/200) is used to compute the data and noise covariances. This is further validated by the eigen vectors in Fig. 11 where in the noisy and denoised setting, the top 11 components are retained. In a realistic setting the true signal will never be known to us, so the best we can expect to recover under an unknown noise setting are the components which have clear segregation between the signal and noise components.

VII. CONCLUSION

We propose a low rank streaming big data framework for Generalized Rayleigh Quotient. Being extended to MNF, it is shown to be capable of working on larger chunks of data than previously possible for denoising in one go. We also showed the accuracy-speed-memory trade-off with parameter selection based on memory budget for limited memory machines. The framework can be directly integrated into sensor for remote sensing data collection, thereby providing real-time denoising of data. As future work, we would like to extend this

generalized framework to large data scale Linear Discriminant Analysis (LDA), Normalized Cuts (NC).

Acknowledgments

This research was supported in part by a grant from NIH R01GM117594.

REFERENCES

- [1]. Upadhyay Jalaj. Fast and space-optimal low-rank factorization in the streaming model with application in differential privacy. arXiv preprint arXiv:160401429, 2016.
- [2]. Tropp Joel A, Yurtsever Alp, Udell Madeleine, and Cevher Volkan. Streaming low-rank matrix approximation with an application to scientific simulation. arXiv preprint arXiv:190208651, 2019.
- [3]. Halko Nathan, Martinsson Per-Gunnar, and Tropp Joel A. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. SIAM review, 53(2):217–288, 2011.
- [4]. Woodruff David P et al. Sketching as a tool for numerical linear algebra. Foundations and Trends® in Theoretical Computer Science, 10(1–2):1–157, 2014.
- [5]. Clarkson Kenneth L and Woodruff David P. Low-rank approximation and regression in input sparsity time. Journal of the ACM (JACM), 63(6):54, 2017.
- [6]. Mouroulis Pantazis, Green Robert O, and Chrien Thomas G. Design of pushbroom imaging spectrometers for optimum recovery of spectroscopic and spatial information. Applied Optics, 39(13):2210–2220, 2000. [PubMed: 18345127]
- [7]. Green Andrew A, Mark Berman, Paul Switzer, and Craig Maurice D. A transformation for ordering multispectral data in terms of image quality with implications for noise removal. IEEE Transactions on geoscience and remote sensing, 26(1):65–74, 1988.
- [8]. Johnson William B and Lindenstrauss Joram. Extensions of lipschitz mappings into a hilbert space. Contemporary mathematics, 26(189–206):1, 1984.
- [9]. Boutsidis Christos, Drineas Petros, and Malik Magdon-Ismael. Near-optimal column-based matrix reconstruction. SIAM Journal on Computing, 43(2):687–717, 2014.
- [10]. Drineas Petros, Kannan Ravi, and Mahoney Michael W. Fast monte carlo algorithms for matrices ii: Computing a low-rank approximation to a matrix. SIAM Journal on computing, 36(1):158–183, 2006.
- [11]. Musco Cameron and Musco Christopher. Stronger approximate singular value decomposition via the block lanczos and power methods. arXiv preprint arXiv:150405477, 16:27, 2015.
- [12]. Cohen Michael B, Sam Elder, Musco Cameron, Musco Christopher, and Persu Madalina. Dimensionality reduction for k-means clustering and low rank approximation. In Proceedings of the forty-seventh annual ACM symposium on Theory of computing, pages 163–172. ACM, 2015.
- [13]. Boutsidis Christos, Woodruff David P, and Zhong Peilin. Optimal principal component analysis in distributed and streaming models. In Proceedings of the forty-eighth annual ACM symposium on Theory of Computing, pages 236–249. ACM, 2016.
- [14]. Tropp Joel A, Yurtsever Alp, Udell Madeleine, and Cevher Volkan. Practical sketching algorithms for low-rank matrix approximation. SIAM Journal on Matrix Analysis and Applications, 38(4):1454–1485, 2017.
- [15]. Gupta Soumyajit, Mittal Shachi, Kajdacsy-Balla Andre, Bhargava Rohit, and Bajaj Chandrajit. A fully automated, faster noise rejection approach to increasing the analytical capability of chemical imaging for digital histopathology. PloS one, 14(4):e0205219, 2019.
- [16]. Kruse Fred A, Lefkoff AB, Boardman JW, Heidebrecht KB, Shapiro AT, Barloon PJ, and Goetz AFH. The spectral image processing system (sips) interactive visualization and analysis of imaging spectrometer data. Remote sensing of environment, 44(23):145–163, 1993.

Fig. 1:

Visual Comparison of preservation of spectral features after adding spectral-correlated Gaussian noise of different strength. $k = 30$ for Indian Pines. Steaming MNF is able to recover all the relevant features under all three setting, but the rank chosen has an affect on its recovery on the non-relevant regions with slight fluctuations.

Fig. 2:

Top: Execution time for MNF and SMNF under normal and streaming setting. Streaming the data helps build up the sketches faster, hence significantly low times. Even for non-stream version SMNF is still faster than MNF till $k = 130$. and Bot: Noise profile for different strengths added to the signal.

Fig. 3:

Top: Spectral Profile of extremely high noise setting. Only the top 2–3 bands contain the most variance in AVIRIS data. When the strength of noise is greater than the forth highest signal component, SMNF still is able to achieve near-perfect MNF like reconstruction at lower values of k . Bottom: Error values for different k . We see a major dip at $k = 60$ and then at $k = 100$ every error metric flattens out. Thus even for a very high strength white noise, the full column space is not needed to find the relevant signal bands.

Fig. 4: Spectral profile of Sketchy MNF in increasing of k from bottom to top at $[20, 40, \dots, 200]$. For comparison, the spectrum of the signal, noisy added signal and full MNF are shown. At $k = 60$ most of the noise in the relevant regions are nullified out. Beyond $k = 100$, it starts visually resembling the MNF spectrum.

Fig. 5:

Top: Spectral decay profile of the data. Since the actual noise distribution is unknown, the noise is calculated by subtracting the MNF denoised data from the noisy data. = 150k for Sketchy MNF. Right: Spectral signature for a pixel is shown. MNF and SMNF denoise similarly and preserve the relevant spectral features.

Fig. 6:

Top: RMSE, SAM and Frobenius error metrics between MNF and Sketchy MNF spectrum for different values of k . Bot: Execution time for Sketchy MNF vs. MNF for different k . Since MNF works with the full data, it is independent of k . The red curve shows the timing for Sketchy MNF under non-streaming setting. At $k = 600$ and beyond, it surpasses MNF. This is due to the extra computation time spent in building up the sketches and forming the low rank matrix, which dominates the pipeline if not streamed. The streaming edition takes far less time and the advantage is more pronounced as the data size keeps scaling.

Fig. 7:

Velocity field at time step 1. The clean image, noisy image and recovered image are shown from left to right. The high magnitude ($\rho = 0.1$) noise is causing the smooth halos in the spatial view to have significant scattering artifacts. Also notice that since the noise magnitude is high, the colorbar scale of the noisy image is different from the other two. Sketchy MNF is able to recover the smooth halos and rescale the magnitudes as well as indicated by the same scale colorbar. Although it is not exact recovery, all the spatial features are preserved.

Fig. 8:

Denoising profile of sample from BR1003 FTIR dataset. The value of k was set to 200 and SMNF selected the top 8 components for reconstruction. Although there are slight variations compared to MNF, which had access to the full data, SMNF managed to recover all the relevant spectral features, given the input data is highly noisy and the noise distribution is unknown.

Fig. 9:

Top: Main memory requirements vs. spatial window chosen from the BR1003 dataset for MNF and SMNF. Since MNF needs the full data, we were able to select at most a $2200 \times 2200 \times 1506$ chunk at a time as our setup's memory was capped at 64GB. Comparatively, SMNF was able to push the input data size till $6000 \times 6000 \times 1506$ for processing in a go. Bottom: The data loading time from disk into memory is illustrated here in a stream vs. no-stream setting.

Fig. 10:

Top: Spectral decay. At $r = 11$, the signal strength falls below noise. While the signal has a sharp decay, the noise decay is almost flat and not of low magnitude. SMNF identifies the signal component correctly up to $r = 11$ and then decays exponentially. Bot: Plot of velocity field of pixel across time. The effect of high strength noise is prominent in the data. Full MNF recovers the exact signal as it looks at the entire data $A^{5000 \times 200}$. Sketchy MNF is able to reconstruct the sinusoidal pattern with the low rank matrix $\hat{A}^{5000 \times 30}$.

Fig. 11:

Top 14 eigen components of the velocity field for original, noisy and denoised data from left to right. Sketchy MNF chose to retain the top 11 bands as having channels of high enough SNR. Bands 12–14 were already destroyed by noise, hence were unrecoverable, hence zeroed out. Although bands 8–11 are corrupted with significant noise, there are signal features which the SNR criteria in MNF retained.

TABLE I:

Error metrics for Indian Pines denoising between MNF and Sketchy MNF ($k = 30$).

Noise Setting	Frobenius	RMSE	SAM
Low ($\rho = 10^{-3}$)	2.2714	0.001779	0.02429
Medium ($\rho = 10^{-2}$)	3.9801	0.002788	0.02806
High ($\rho = 10^{-1}$)	5.3201	0.003271	0.03393