Finding Structure with Randomness: Probabilistic Algorithms for Constructing Approximate Matrix Decompositions

Oussama ENNAFII

December 26, 2014

Abstract

Matrix factorization is a powerful tool to achieve tasks efficiently in numerical linear algebra. A problems arises when we compute low-rank approximations for massive matrices: we have to reduce the algorithms' complexity in time to hope to be efficient. A way to adapt these techniques for such computational environments is randomization. This report presents a framework for these new techniques, based mainly on the work presented in : « Finding Structure with Randomness : Probabilistic Algorithms for Constructing approximate Matrix Decompositions ». The main intuition behind the various algorithms presented herein is using random sampling to apprehend the action of the matrix in a « compressed » subspace. We can apply then the classical methods on the resulting matrix – the one acting on the « compressed » subspace – to obtain a low-rank approximation. An other application that rises from this explanation is the fact that this method is thus more robust addressing incomplete data sets that one can get in information sciences, while other .The benefits of such methods will depend on the matrix. We will decline the results for three main classes: Dense matrices where the new complexity is of order $mnlog(k)$, comparing with O(mnk) for classical methods – k is the numerical rank ; Sparse matrices and in general structured matrices ; Massive matrices that do not fit in fast memory for which the access time – which surpasses computation time – can be reduced to one, comparing to $O(k)$.

Contents

1 Introduction:

1.1 Presentation of the problem:

Matrix factorization is listed amongst the 10 most influential algorithms. In deed, Barry A. Cipra,[1] lists Krylov subspace methods (3rd position), decompositional approach to matrix computations (4th position) and the QR algorithm (6th position) amongst the top 10 algorithms.

In fact, linear algebra computational techniques should be separated from its applications. One should focus only on one specialization. Matrix manipulators should develop frameworks with algorithms to solve numerical linear algebra problems.

The classical algorithms, dating back to the middle of the previous century, are no longer adequate for some applications, which are the result of the developments in computer hardware, information sciences and big data.

Of the new applications we can site modern data mining applications, inaccurate data in information science and new architectures involving, for instance, graphics processing units. In fact, new data mining applications involve very large matrices that classical methods cannot hope to solve efficiently. Otherwise, in information sciences, it goes without mentioning the fact that data is usually missing or inaccurate. So it seems misguided to spend too much of the highly expensive computational capacity on inaccurate information. In addition, some data matrices are so big that they cannot be stored in easy access memory. Data access time surpasses thus the computational requirements while classical algorithms are multipass.

1.1.1 Low-rank approximation:

We will present herein a special case of matrix decomposition techniques. The approximation by low-rank matrices.

As we have seen before hand, the QR algorithm and the Krylov subspace methods are sited amongst the top 10 algorithms apart from the decompositional approach to matrix computations. This fact highlights the importance of low rank approximations which includes the QR factorisation, the singular eigenvalue decomposition (SVD). These methods expose the range of the

matrix.

In general, we get:

$$\underbrace{A}_{\text{m x n}} \approx \underbrace{B}_{\text{m x k}} \; x \; \underbrace{C}_{\text{k x n}}$$

k is the numerical rank of the matrix. When $k << min(m, n)$, we can store the matrix and compute linear operations on it quite efficiently.

Low rank approximation has many applications. The principal component analysis (PCA), in statistics, for one, is nothing but a low rank approximation. Low rank approximations are also of use in parameter estimation with least squares, as the design matrix may not be inverted. We use then the QR decomposition to calculate the Moore-Penrose inverse.

1.2 Randomized matrix approximation:

Randomized algorithms provide simple and effective tools to perform matrix approximate factorizations. The randomized methods are faster and more robust and, with a trade of accuracy for speed, it yields results up to any precision.

In general, matrix approximation through randomisation is done following the paradigm:

(i) Preprocessing: computing sampling probabilities.

(ii) Sapling; a linear function of the matrix.

(iii) Postprocessing; by applying classical techniques of linear algebra on the samples.

We will site here the most common techniques:

- **Sparcification:** replace the matrix by a substitute that contains much less nonzero entries. We can also quantize entries such that we obtain an approximate matrix. The resulting matrix multiplication by a vector is less time consuming than multiplying the original matrix by a vector.

- **Column selection methods:** select a small set of columns that describes most of the range of the matrix. This method is based on the fact that [2]:

$$\|A - CC^{\dagger}A\| \leq \sqrt{1 + k(n-k)}\ \|A - A_{(k)}\| \tag{1}$$

  The dagger denotes the pseudoinverse, $A_{(k)}$ represents the best k-rank approximation of A and C is k-column submatrix of A. Althought it is NP-hard to choose the best k columns, there are efficient techniques based on the QR method[?].

- **Dimension reduction:** We start from the fact that the matrix rows are dependent: they can be embedded in a low dimension subspace without altering the geometric properties based on the Johnson–Lindenstrauss lemma.

1.3 The two stage approach framework:

The main idea in this framework is to capt the most of the action of the matrix:i.e. We want to apply the matrix on a dimention-reduced space so as to obtain an approximation of the result obtained in the full space. It is a dual to the dimension reduction technique. We then apply the classical methods on the "reduced" matrix.In this framework, we are using then two stage algorithms.

Formaly, suppose that k is the numerical range of the mxn matrix A. Let Q be the matrix representation of k orthonormal vectors(columns). The "reduced" matrix is:

$$B = Q^{*}A$$

If the subspace generated by Q captures the most of A's action, we should get:

$$A \approx QQ^{*}A$$

We can then describe briefly the algorithms as follows:

Stage A:

Produce a mxk matrix Q such that:

$$A \approx QQ^{*}A$$

Stage B:

Apply classical techniques - such as QR and SVD - on the "reduced" matrix:

$$B = Q^* A$$

It is in Stage A that the randomization intervenes. In fact, we will draw k random vectors to sweep over the range of A.In deed, the resulting vectors will form an approximation of the range of the matrix we are studying. We just apply an orthormalization technique - such as Gramm-Schmidt - to deduct Q. This is not new. The new idea presented here is based on the fact that we sample $l(> k)$ vectors. Let us try explain why.

Suppose we drawed k vectors ω^i. We hope that these vectors being drawn randomnly are independent and the resulting subspace does not intersect with the matrix's kernel. We are the confident that the vectors y^i will describe the range of A, where:

$$y^i = A\omega^i \quad , i = 1...k$$

The problem is that when we approximate A the error we make shifts these vectors out from the range of A. We thus have to oversample: i.e. draw more ω^i.

Another issue arises. All this time we considered that k is already set. The problem is that we don't know k, in fact, we want to find k. We reformulate then the Stage A equation:

$$||A - QQ^* A|| < \epsilon \tag{2}$$

The numerical rank depends then on the precision: $k = k(\epsilon)$.

To solve the problem, we will use the SVD. Let σ_j the jth largest singular value. We know that:

$$\min_{range(X) \leq k} ||A - X|| = \sigma_{j+1}$$

One way to find the minimizer we choose $X = QQ^* A$ where Q are k dominant left singular vectors of A and k is determined by the fact that $\sigma_{k+1} < \epsilon$. We go back then to the *fixed rank problem.*

The *fixed rank problem* can be solved then by this proto-algorithm:

ProtoAlgorithm
1. Draw an n x (k + p) random matrix $\mathbf{\Omega}$
2. Form the matrix $\mathbf{Y} = \mathbf{A\Omega}$
3. Construct a matrix $\mathbf{Q}$ whose columns form an othonormal basis for the range of $\mathbf{Y}$

More discussion on the choices of $\mathbf{\Omega}$ will come in the next sections.

2 The matrix approximation framework algorithms:

2.1 Stage A Algorithms:

2.1.1 Randomized Range Finder:

The simplest implementation of the proto-algorithm is the randomized range finder algorithm. Given a matrix $\mathbf{A}$ and a integer $l(> k)$, the oversampled numerical rank, it computes an orthonormal basis $\mathbf{Q}$. The $\mathbf{\Omega}$ matrix is drawn using a Gaussian distribution with mean 0 and variance 1.

Randomized Range Finder
1. Draw an n x l Gaussian random matrix $\mathbf{\Omega}$
2. Form the m x l matrix $\mathbf{Y} = \mathbf{A\Omega}$
3. Construct the m x l matrix $\mathbf{Q}$ using the QR factorization $\mathbf{Y} = \mathbf{QR}$

The oversampling parameter $p := l - k > 0$ depends on the matrix dimensions, the decrease of the ordered singular spectrum and the random test matrices. For Gaussian matrices, p of the order of 5 or 10, yields good results and there is no need for more oversampling than k.

The most consuming part is the second step product.

The QR factorization is done either with Gramm-Schmidt algorithm, householder reflections or Givens rotations.

The number of flops necessary for this algorithm is:

$$lnT_{rand} + lT_{mult} + l^2m$$

where:

- T_{rand} time to sample l n-length standard gaussian vectors.
- T_{mult} time to evaluate the matrix-vector product.

2.1.2 Adaptive Randomized Range Finder:

In the previous setting, k is fixed. Actually, we can do better by addapting the algorithm almost for free. In fact, due to following lemma[?], we can evaluate the error by setting $\alpha = 10$ and applying it to $B = (I - QQ^*A$. We get, with probability $1 - \alpha^{-r}$:

$$\|(\mathbf{I} - \mathbf{Q}\mathbf{Q}^T)\mathbf{A})\| \leq 10\sqrt{\frac{2}{\pi}} \max \|(\mathbf{I} - \mathbf{Q}\mathbf{Q}^T)\mathbf{A}\omega^{(i)})\| \tag{3}$$

Lemma 2.1 *Let $\boldsymbol{B}$ be a real m x n matrix. Fix $r > 0$ an integer and a real number $\alpha > 1$. Draw an independent family $\{\omega^i : i = 1...r\}$ of standard Gaussian vectors. Then, with probability $1 - \alpha^{-r}$:*

$$\|\mathbf{B}\| \leq \alpha\sqrt{\frac{2}{\pi}} \max \|\mathbf{B}\omega^{(i)}\|$$

In consequence, we don't need to fixe the oversampling parameter. As l will be determined depending on ϵ. To be more precise, we want to find l such as the m x l $\mathbf{Q}^{(l)}$ matrix verifies:

$$\|(\mathbf{I} - \mathbf{Q}^{(l)}(\mathbf{Q}^{(l)})^*)\mathbf{A})\| \leq \epsilon$$

We then addapt the algorithm by adding collumns to Q so that we get the desired precision that we can now measure now before hand.

Adaptive Randomized Range Finder

1 Draw standard Gaussian vectors $\{\omega^i : i = 1...r\}$ of lenght n.

2 For $i = 1...r$ compute $y^{(i)} = A\omega^i$.

3 $j := 0$

4 $\mathbf{Q}^{(0)} := [\,]$

5 **While** $max_{i=1...r}||y^{(i+j)}|| > \frac{\epsilon}{10\sqrt{\frac{2}{\pi}}}$

6 j++

7 $y^{(i)} := (\mathbf{I} - \mathbf{Q}^{(j-1)}(\mathbf{Q}^{(j-1)})^*)y^{(j)}$

8 $q^{(i)} = \frac{y^{(i)}}{||y^{(i)}||}$

9 $\mathbf{Q}^{(j)} = [\mathbf{Q}^{(j-1)}q^{(i)}]$

10 Draw ω^{j+r} of length n.

11 $y^{(j+r)} := (\mathbf{I} - \mathbf{Q}^{(j)}(\mathbf{Q}^{(j)})^*)A\omega^{j+r}$

12 **for** $i = 1...r - 1$

13 $y^{(i+j)} := y^{(i+j)} - q^{(i+j)} < q^{(i+j)}, y^{(i+j)} >$

14 **end for**

15 **end while**

16 $\mathbf{Q} = \mathbf{Q}^{(j)}$

2.1.3 Randomized Power iteration:

The Randomized Range Finder supposes that matrices have singular values that decay quickly. As it performs poorly on matrices that have singular values that decay slowly or are too large. Singular vector associated with the small singular values interfere in the calculation. The idea behind the

power iteration is to reduce the weight associated to those values. In order to do so, we compute the matrix:

$$\mathbf{B} = (\mathbf{A}(\mathbf{A})^*)^q \mathbf{A}\mathbf{\Omega}$$

.We get then its singular values:

$$\sigma_j(\mathbf{B}) = \sigma_j(\mathbf{A})^{2q+1}, j = 1, 2, 3... \tag{4}$$

RANDOMIZED POWER ITERATION
1. Draw an n x l Gaussian random matrix $\mathbf{\Omega}$
2. For the m x l matrix $\mathbf{Y} = (\mathbf{A}\mathbf{A}^*)^q \mathbf{A}\mathbf{\Omega}$ via alternative application of $\mathbf{A}$ and $\mathbf{A}^*$.
3. Construct the m x l matrix $\mathbf{Q}$ using the QR factorization $\mathbf{Y} = \mathbf{QR}$

Remark Rounding errors in this algorithm for floating-point operaions result in killing singular modes associated with singular values small compared to $\|A\|$. One way to overcome this issue is done by orthonomalizing the columns after each application of $\mathbf{A}$ and $\mathbf{A}^*$.

RANDOMIZED SUBSPACE ITERATION
1. Draw an n x l Gaussian random matrix $\mathbf{\Omega}$
2. $\mathbf{Y_0} := \mathbf{A}\mathbf{\Omega}$ and compute: $Y_0 == Q_0 R_0$
3. **for** $j = 1...q$
   $Y_j := A^* Q_{j-1}$ and compute:$Y'_j == Q'_j R'_j$.
   $Y_j := A Q'_j$ and compute: $Y_j == Q_j R_j$ **end**
4. $Q = Q_q$

Remark Like in the previous case, we can derive an adaptive algorithm for the power iteration algorithm, using the same trick to evaluate error online.

2.1.4 Fast Randomized Range Finder:

Random Gaussian matrices are not adapted for dense matrices. One way to adapt to dense matrices without special structures is by using the Fast Fourier Transform(FFT). We use then the subsampled random Fourier transform(SFRT). Subsampled because we sample only for l columns, randomn

because we randomly rotate on each direction. An SFRT is :

$$\mathbf{\Omega} = \sqrt{\frac{n}{l}}\mathbf{DFR} \tag{5}$$

where:

- **D** is a n x n diagonal matrix whose entries are random variables distributed on the complex unit circle
- **F** is the n x n unitary discrete Fourier transform
- **R** is an n x l matrix whose l column are drawn from the n x n identity matrix.

FAST RANDOMIZED RANGE FINDER 1. Draw an $n * l$ SRFT test matrix $\mathbf{\Omega}$ as defined by
2. Form the $m * l$ matrix $\mathbf{Y} = \mathbf{A\Omega}$ using subsampled FFT
3. Construct the $m * l$ matrix $\mathbf{Q}$ using the QR factorization.

The number of flops necessary for this algorithm is:

$$mnlog(l) + l^2m$$

2.2 Stage B:

2.2.1 From one factorization to another:

The main low rank approximations are the QR decomposition and the truncated SVD. In general, we can derive any matrix factorization from any other one. In deed, suppose we got the following approximation:

$$\| \underbrace{A}_{\text{m x n}} - \underbrace{B}_{\text{m x k}} \; x \; \underbrace{C}_{\text{k x n}} \| \leq \epsilon$$

We can get the QR factorization through: 1. Computing the QR decomposition of $C = Q_1R_1$. 2. Forming the product$D = R_1B$ and computing its QR decomposition: $D = Q_2R$. 3.Form the matrix $Q = Q_1Q_2$.

We thus get:

$$\|A - QR\| \leq$$

We can also get the SVD by: 1. Computing the QR decomposition of $C = Q_1 R_1$. 2. Forming the product$D = R_1 B$ and computing its SVD:$D = U_2(\Sigma)V^*$. 3. Forming the product:$Q = Q_1 U_2$.

The resulting decomposition is:

$$||A - U\Sigma V^*|| \leq \epsilon$$

We can applu the same technique for other factorizations.

2.2.2 Direct SVD:

In the Stage B, we perform classic algorithm on $B = Q^* A$.

DIRECT SVD
Given a matrix $\mathbf{A}$ and an orthornomal basis $\mathbf{Q}$ such that:
1. Form the matrix $\mathbf{B} = \mathbf{Q}^*\mathbf{A}$
2. Compute the SVD of the matrix $\mathbf{B} = \tilde{\mathbf{U}}\mathbf{\Sigma}(\mathbf{V})^*$
3. Form the orthonormal matrix $\mathbf{U} = \mathbf{Q}\tilde{\mathbf{U}}$

This algorithm yields then:

$$||A - U\Sigma V^*|| \leq \epsilon$$

2.3 Theory:

2.3.1 Preliminaries:

In this subsection we will present some interesting properties of positive some definite matrices.

Lemma 2.2 *Let $M \geq 0$. Then:*

$$I - (I + M)^{-1} \leq M$$

Proof

$$I - (I + M)^{-1} = M(I + M)^{-1} = M^{-1/2}(I + M)^{-1}M^{-1/2} \leq M$$

.

Lemma 2.3 *Let M be a matrix such that:*

$$M = \begin{bmatrix} A & B \\ B^* & C \end{bmatrix}$$

We can say that:

$$||M|| \leq ||A|| + ||C||$$

Let P_M the unique orthogonal projector such that $range(M) = range(P_M)$. If M has full range, we can write:

$$P_M = M(M^*M)^{-1}M^*$$

Lemma 2.4 *Suppose U is unitary.*

$$U^*P_MU = P_{U^*M}$$

Proof

$$U^*P_MUU^*P_MU = U^*P_MU$$

So U^*P_MU is an othogonal projector since it is Hermitian. Since the orthogonal projectors are detremined by there range, it suffices to determine the range of U^*P_MU :

$$range(U^*P_MU) = U^*range(P_M) = range(P_{U^*M})$$

.

Proposition 2.5 *Let P an orthogonal projector, D a non negative diagonal matrix and $t \geq 1$.*

$$||PM|| \leq ||P(MM^*)^qM||^{1/(2q+1)}, \forall q \geq 0$$

Proposition 2.6 *Set S and T, and draw a standard Gaussian matrix G. Then:*

$$(\mathbb{E}||SGT||_F^2)^{1/2} = ||S||_F||T||_F \tag{6}$$

$$\mathbb{E}||SGT|| \leq ||S||\ ||T||_F + ||T||\ ||S||_F \tag{7}$$

The second equation results froms the work of Gordon[62,63].

Proposition 2.7 *Let G be a k x k+p standard Gaussian matrix with $p, k \geq 2$. Then:*

$$(\mathbb{E}\|G^{\dagger}\|_F^2)^{1/2} = \sqrt{\frac{k}{p-1}} \tag{8}$$

$$\mathbb{E}\|G^{\dagger}\| = \frac{e\sqrt{k+p}}{p} \tag{9}$$

The first equality is a standard result[100]. The second one is due to Chen and Dongarra[25].

Proposition 2.8 ***Concentration inequality*** *Let h be a L-Lipschitz function of matrices and G a standard gaussian matrix. Then:*

$$\mathbb{P}\{h(G) \geq \mathbb{E}h(G) + Lt\} \leq e^{-t^2/2} \tag{10}$$

2.3.2 Error bounds:

We rewrite :

$$A = U \begin{pmatrix} \Sigma_1 & \\ & \Sigma_2 \end{pmatrix} \begin{pmatrix} V_1^* \\ V_2^* \end{pmatrix}$$

and

$$\Omega_1 = V_1^* \Omega \text{ and } \Omega_2 = V_2^* \Omega$$

$$Y = A\Omega = U \begin{pmatrix} \Sigma_1^* \Omega_1 \\ \Sigma_2^* \Omega_2 \end{pmatrix}$$

$$\|A - QQ^*A\| = \|(I - P_Y)A\|$$

Error bounds for the proto-algorithm: The most impotant theorem is the one that follows. The rest may be quite directly deduced from this one using the properties in the preliminary subsection.

Theorem 2.9 *Assuming that Ω_1 has full row rank,*

$$\|(I - P_Y)A\|^2 \leq \|\Sigma_2\|^2 + \|\Sigma_2 \Omega_2 \Omega_1^{\dagger}\| \tag{11}$$

Proof Let:

$$\tilde{A} = U^*A$$

and

$$\tilde{Y} = \tilde{A}\Omega$$

We get:

$$\|(I - P_Y)A\| = \|U^*(I - P_Y)U\tilde{A}\| = \|(I - P_{\tilde{Y}})\tilde{A}\|$$

There are two cases:

1. Σ_1 is not strictly positive, then $\Sigma_2 = 0$ because of the sigular value ordering : i.e.

$$range(\tilde{A}) = range\begin{pmatrix}\Sigma_1\Omega_1\\0\end{pmatrix} = range(\tilde{Y})$$

It means that:

$$\|(I - P_{\tilde{Y}})\tilde{A}\| = 0$$

2. Σ_1 is strictly positive.
Let

$$W := \begin{pmatrix}\Sigma_1\Omega_1\\0\end{pmatrix}$$

Since Ω_1 has full row rank:

$$range(W) = range\begin{pmatrix}I_k\\0\end{pmatrix}$$

$$P_W = \begin{pmatrix}0 & 0\\0 & I\end{pmatrix}$$

Constructing the matrix Z by flattening the top of $\tilde{Y}$:

$$Z = \tilde{Y}\Omega_1^*\Sigma_1^{-1} \tag{12}$$

By construction, range($\mathbf{Z}$) $\subset$ range($\tilde{\mathbf{Y}}$). That implies that:

$$\|(I - P_{\tilde{Y}}\tilde{A})\| \leq \|(I - P_Z\tilde{A})\| \tag{13}$$

So, we get:

$$|(I - P_{\tilde{Y}}\tilde{A})\| \leq \|\Sigma^*(I - P_Z)\Sigma\| \tag{14}$$

The rest of the proof is obtained by simple application of the Lemma 2.3.

Error bounds for the range finder algorithm:

Theorem 2.10 *We keep the same notations as before.* $k, p \geq 2$ *and* $k + p \leq min(m, n)$*:*

$$\mathbb{E}\|(I - P_Y)A\|_F \leq (1 + \frac{k}{p-1})^{1/2}(\sum_{j>k} \sigma_j^2)^{1/2} \tag{15}$$

and for the spectral norm :

$$\mathbb{E}\|(I - P_Y)A\| \leq (1 + \frac{k}{p-1})\sigma_{k+1} + \frac{e\sqrt{k+p}}{p}(\sum_{j>k} \sigma_j^2)^{1/2} \tag{16}$$

Remark The second inequality is too loose because it is derived from the inequality on the Froebinus norm. We can see also that the minimal error we can get is the one we get with truncated exact SVD.

Theorem 2.11 *Now for* $p \geq 4$ *and* $t \geq 1$*, with probability at most* $2t^{-p} + e^{-u^2/2}$*:*

$$\|(I - P_Y)A\|_F \leq (1 + t\sqrt{\frac{3k}{p+1}})(\sum_{j>k} \sigma_j^2)^{1/2} + ut\frac{e\sqrt{k+p}}{p}\sigma_{k+1}$$

and:

$$\|(I - P_Y)A\| \leq [(1 + t\sqrt{\frac{3k}{p+1}})\sigma_{k+1} + t\frac{e\sqrt{k+p}}{p}(\sum_{j>k} \sigma_j^2)^{1/2}] + ut\frac{e\sqrt{k+p}}{p}\sigma_{k+1}$$

2.4 Numerics:

In this section, is presented the second application in the paper.

This example involves a large matrix that arises in image processing. Some image processing algorithms uses the geometry of the image for tasks such as denoising and inpainting. They use a graph Laplacian to represent the geometry of the image. We use with a small grayscaled patch of lena, of $50x50$. Each pixel is represented by a value between 0 and 255. Each pixel x is represented by a 5 x 5 patcharound this pixel. We calculate the weight matrix $\tilde{\mathbf{W}}$, reflecting the similarity between the patch with:

$$\tilde{w_{ij}} = exp\{\frac{-(x_i - x_j)^2}{\sigma^2}\}$$

By zeroing all the entries of the weigth matrix $\tilde{\mathbf{W}}$ except the seven largest ones in earch row, we construct our large sparse matrix W. We can then construct the graph Laplacian matrix:

$$\mathbf{L} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{1/2}$$

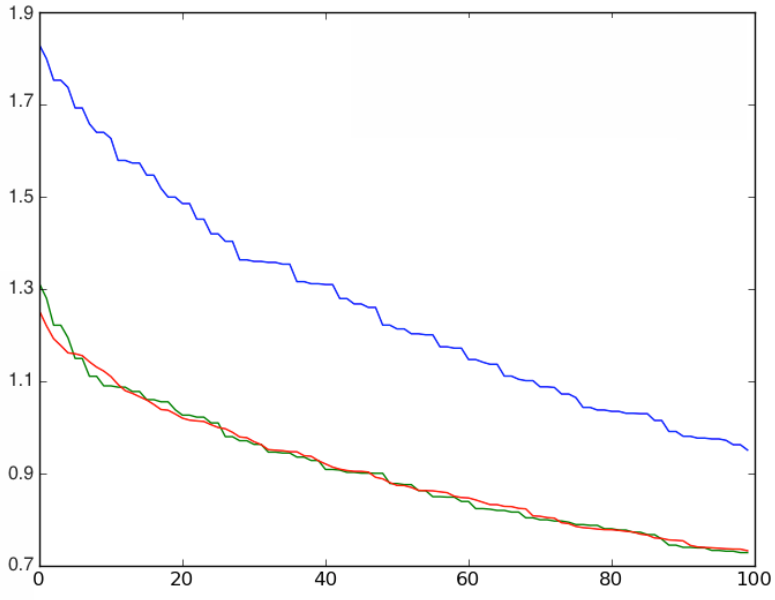
.

Here, we are interested in the eigenvalues of:

$$\mathbf{A} = \mathbf{D}^{1/2}\mathbf{W}\mathbf{D}^{-1/2}$$

which decay very slowly as the figure shows.

The blue plot describes the first algorithm (i.e. $q = 0$). This algorithm runes very fast - around 4.5 seconds.The green plot describes the power iteration algorithm with $q = 4$. It takes around 9.8 seconds to compute but it is fare more accurate. The last plot representes the SFRT method. It takes around 11 seconds to compute and yields similar results to the power iteration algorithm.



3 Conclusion:

The framework presented here is quite complete. It presents algorithms adapted to each case. The main issue was finding a low rank approximation through randomization to reduce the computational time complexity of classical techniques. It could be completed with more detailed sampling methods for other structured matrices and the corresponding error bounds. Efforts should also be made also to try and derive more sophisticated tools and obtain more powerful error estimations. As we seen the bounds given here are very large comparing with the numerics. The goal is to solve the fixed precision problem more efficiently for sparce matrices. Otherwise, one also can improve this framework by determining a more precise way to get the oversampling parameter for the SRFT matrices.

Bibliography:

[1] N. Halko † P. G. Martinsson † J. A. Tropp ,Finding Structure with Randomness: Probabilistic Algorithms for Constructing Approximate Matrix Decompositions , SIAM REVIEW Vol. 53, No. 2, pp. 217–288.

[2] Barry A. Cipra, The Best of the 20th Century: Editors Name Top 10 Algorithms,SIAM News , Volume 33, Number 4

[3] A.F.Ruston , Auerbach's theorem , Math. Proc. Cambridge Philos. Soc., 56 (1964), pp. 476–480.

[4] M. Gu and S. C. Eisenstat , Efficient algorithms for computing a strong rank-revealing QR factorization , SIAM J. Sci. Comput., 17 (1996), pp. 848–869.

[5] F.Woolfe,E.Liberty,V.Rokhlin,andM.Tygert , A fast randomized algorithm for the approximation of matrices , Appl. Comput. Harmon. Anal., 25 (2008), pp. 335–366.