ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

MATH-403 Mini-Project

Sketching for low-rank matrix approximation

Zilan CHENG

December 17, 2020

Report Type:	Course Project
Student:	Zilan CHENG
Teacher:	Prof. Dr. Daniel Kressner
Teaching Assistant:	Alice Cortinovis
Date:	December 17, 2020

Sketching for low-rank matrix approximation

Answer 1

First, we use $Python$ to implement the SKETCHING algorithm. To specify, we employ Jupyter Notebook and the .ipynb file is submitted in the zip file.

Then, we compute the asymptotic time complexity.

Step 1: Draw Gaussian random matrices $X \in \mathbb{R}^{m\times r}$ and $Y \in \mathbb{R}^{n\times(r+l)}$.

Step 2: Compute the products AX and $Y^T A$.

- $A \in \mathbb{R}^{n\times m}$, $X \in \mathbb{R}^{m\times r}$, then $AX \in \mathbb{R}^{n\times r}$. The time complexity of calculating AX is $O(nmr)$.
- $Y^T \in \mathbb{R}^{(r+l)\times n}$, $A \in \mathbb{R}^{n\times m}$, then $Y^T A \in \mathbb{R}^{(r+l)\times m}$. The time complexity of calculating $Y^T A$ is $O((r+l)mn)$.

Step 3: Compute an economy QR factorization $QR = Y^T AX$.

(1) Since we have computed $Y^T A$, so if we would like to compute $Y^T AX \in \mathbb{R}^{(r+l)\times r}$ we just need to multiple $Y^T A$ with X.The time complexity of this step is $O(mr(r+l))$.

(2) Then we would like to compute the economic QR factorization. The time complexity of computing $Q \in \mathbb{R}^{(r+l)\times(r+l)}$ and $R \in \mathbb{R}^{(r+l)\times r}$ is $O((r+l)r^2)$.

Step 4: Set $B \leftarrow AXR^\dagger$ and $C \leftarrow A^T YQ$.

First of all, we need to concentrate on computing $R^\dagger$. In Python's package $numpy$, we use SVD method to compute Moore-Penrose pseudo-inverse: $R = U\Sigma V$, $R^\dagger = V\Sigma^\dagger U$.$U \in \mathbb{R}^{(r+l)\times(r+l)}$, $\Sigma \in \mathbb{R}^{(r+l)\times r}$, $V \in \mathbb{R}^{r\times r}$, $\Sigma^\dagger \in \mathbb{R}^{r\times(r+l)}$.

- The complexity of decomposition is $O(r^2(r+l) + r(r+l)^2)$.
- The complexity of computing $V\Sigma^\dagger U$ is $O(r^2(r+l) + r(r+l)^2)$.
- The complexity of computing $B \leftarrow AXR^\dagger$ is $O(nmr + nr(r+l))$.
- The complexity of computing A^T is $O(mn)$.
- The complexity of computing $C \leftarrow A^T YQ$ is $O(mn(r+l) + m(r+l)^2)$.

To sum up: Given $n \gg m \geq r$ and $0 < r \leq l$,The asymptotic time complexity of the algorithm is: O(mnl).

Answer 2

In the proof below, we use such two properties of Moore-Penrose pseudo-inverse:

Property 1: $A \in \mathbb{R}^{a\times b}$, $B \in \mathbb{R}^{b\times c}$. If A has orthonormal columns or B has orthonormal rows, then $(AB)^\dagger = B^\dagger A^\dagger$.

Property 2: For a matrix A having linearly independents rows, there exists a left inverse as $A^{\dagger}A = I$. For a matrix A having linearly independents columns, there exists a right inverse as $AA^{\dagger} = I$. I is the Indentity Matrix.

Since $r \leq m \ll n$,we can assume that $2r < n$.

If $U_0\Sigma_0 V_0$ ($U_0 \in \mathbb{R}^{n\times n}$,$\Sigma_0 \in \mathbb{R}^{n\times m}$,$V_0 \in \mathbb{R}^{m\times m}$) is the singular vector factorization(SVD) of $A \in \mathbb{R}^{n\times m}$, we can give an economy-form SVD (different from what we defined as economy size SVD) based on that $rank(A) = r$. That is, $A = U\Sigma V$, $U \in \mathbb{R}^{n\times 2r}$ whose entry $u(i,j)(1 \leq i \leq n$, $1 \leq j \leq 2r)$ is the same as U_0, $\Sigma \in \mathbb{R}^{2r\times r}$ whose entry $\sigma(i,j)(1 \leq i \leq 2r$, $1 \leq j \leq r)$ is the same as Σ_0, $V \in \mathbb{R}^{2r\times m}$ whose entry $u(i,j)(1 \leq i \leq n$, $1 \leq j \leq 2r)$ is the same as V_0.

Since $Q \in \mathbb{R}^{2r\times 2r}$ is the orthogonal matrix, Q is invertible as $QQ^{-1} = I$, where I is the Identity Matrix. So from $QR = Y^T AX$ we could obtain that

$$\begin{aligned} R &= Q^{-1}Y^T AX \\ &= Q^{-1}(Y^T U)\Sigma(VX) \end{aligned} \tag{1}$$

$Y^T U \in \mathbb{R}^{2r\times 2r}$,$VX \in \mathbb{R}^{r\times r}$ are square Gaussian random matrices with probability 1.

From the **Hint**, we obtain that there exists $(Y^T U)^{-1}$ and $(VX)^{-1}$ with probability 1.

From the **Property 1**, we obtain that

$$\begin{aligned} R^{\dagger} &= [Q^{-1}(Y^T U)\Sigma(VX)]^{\dagger} \\ &= (VX)^{-1}\Sigma^{\dagger}(Y^T U)^{-1}Q \end{aligned} \tag{2}$$

with probability 1.

From the **Property 2**, we obtain that $\Sigma\Sigma^{\dagger} = I$.

Then

$$\begin{aligned} BC^T &= (AXR^{\dagger})(A^T YQ)^T \\ &= AXR^{\dagger}Q^T Y^T A \\ &= U\Sigma(VX)(VX)^{-1}\Sigma^{\dagger}(Y^T U)^{-1}(QQ^T)(Y^T U)\Sigma V \\ &= U(\Sigma\Sigma^{\dagger}\Sigma V \\ &= U\Sigma V \\ &= A \end{aligned} \tag{3}$$

with probability 1.

To summarize, SKETCHING(A,r,r) returns an exact low-rank factorization of A with probablity 1.

Answer 3

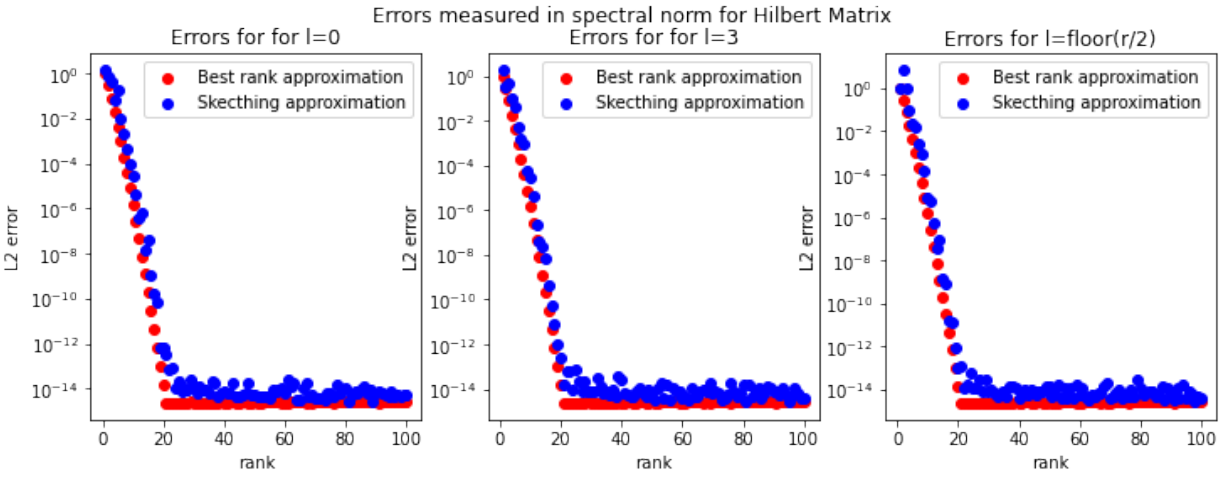
First, we plot the error of the low-rank approximations that you obtain with ranks 1,...,100, for $l = 0, l = 3, l = floor(r/2)$.

The norm we employ to calculate the error here is spectral norm (which equals to the largest singular value).The errors ploting with Frobenius norm almost have the same features with

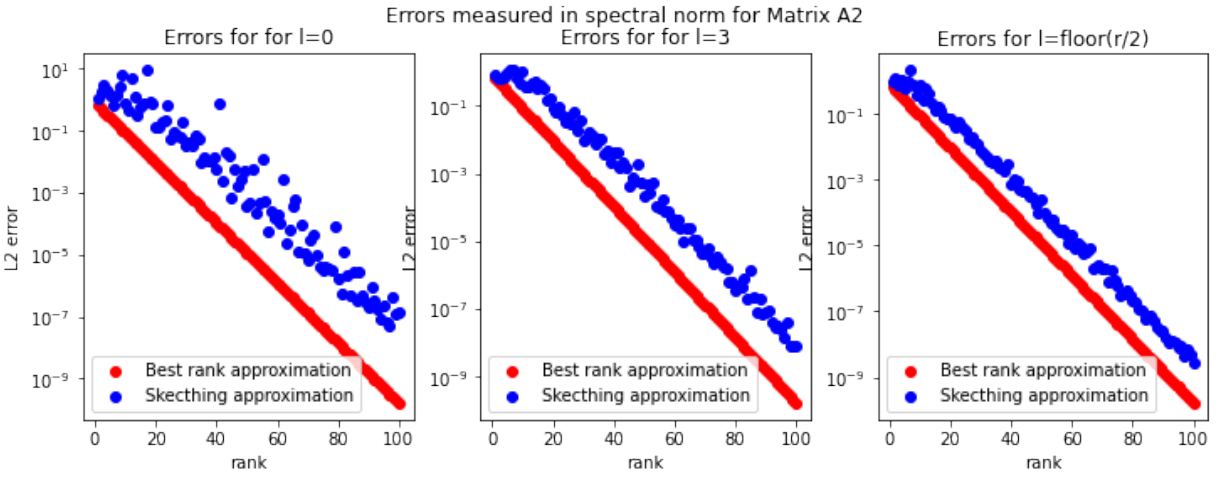
the errors ploting with spectral norm, so we just discuss about spectral norm here.(If we would like to obtain the outcome of Frobenius form, we just need to change the parameter ord in the $np.linalg.norm$ function,).

To more intuitively see the changing trend of L_2 error with rank, we use $ax.set\_yscale('log')$ to rescale the y axis. Without this setting of y axis, the dots almost form a line begin from $rank < 10$ and we almost can not notice anything from the plots.

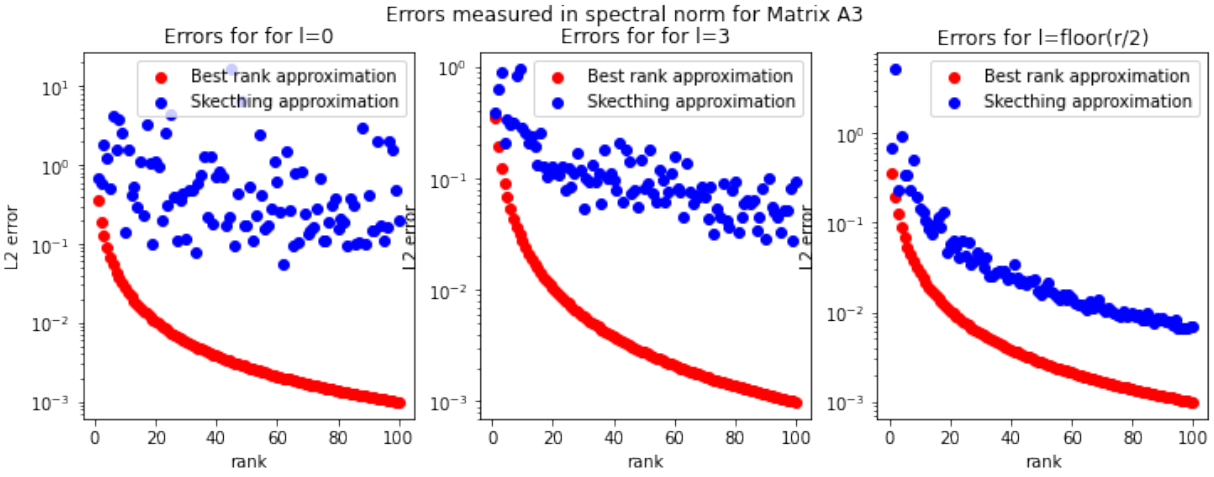
(1)The plot below is the errors measured in spectral norm for Hilbert Matrix.



(2)The plot below is the errors measured in spectral norm for matrix A_2.



(2)The plot below is the errors measured in spectral norm for matrix A_3.

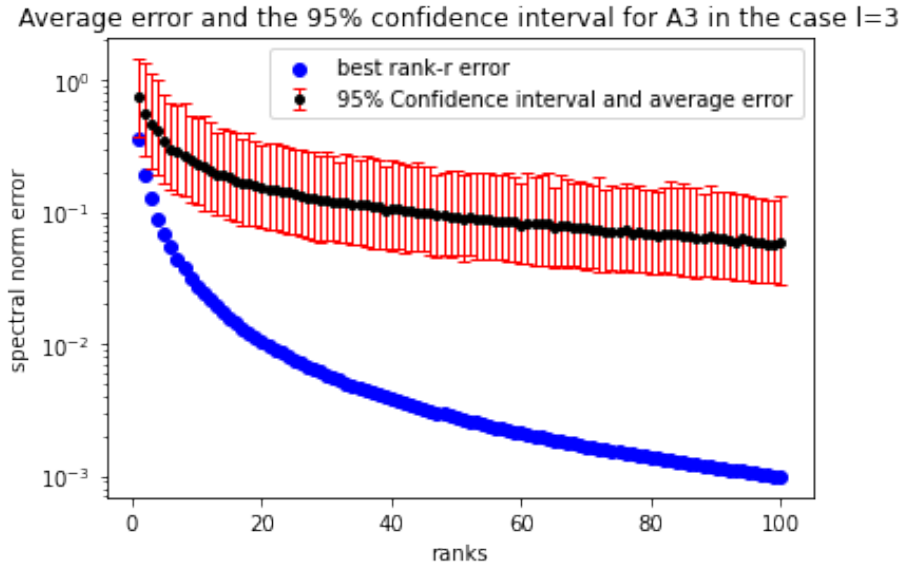


From these plots, I notice that:

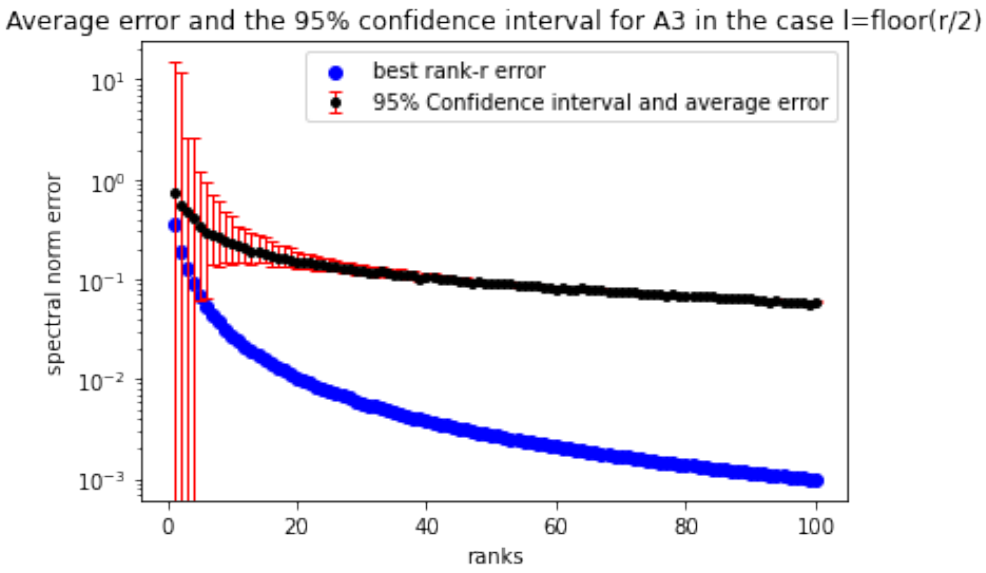
- The error of sketching method is larger than best-rank approximation. It is reasonable.
- The value of l influence the effect of the Sketching method.It seems that $l = floor(r/2)$ is the optimal choice to decrease the error.
- As the rank increases, the error decreases. However, this decrease velocity and effect is sensitive to different matrices. e.g.: The error of the Hilbert Matrix, whose absolute value of slope is very large, has decreased to around 10^{-14} at around rank 20. However,the error of the matrix M_3, whose slope is relatively flat than Hilbert Matrix and the matrix M_2, is still at around 10^{-2} level at rank 100.

Answer 4

(1) The plot below give the information of average error and the 95% confidence interval for Matrix A_3 in the case $l = 3$.



(2) The plot below give the information of average error and the 95% confidence interval for Matrix A_3 in the case $l = floor(r/2)$.



From these plots, we can easily know that, as rank increases, the average error decreases.

However, the 95% confidence interval is sensitive to the value of l. As rank increases, the 95% confidence interval of the case $l = floor(r/2)$ become very small rapidly; however, the 95% confidence interval of the case $l = 3$ seems not change a lot. By going through the reference and the analysis of the plots we make, we recommend using $l = floor(r/2)$ so this random algorithm could be more stable.

1 References

[1] Woodruff, David P. Sketching as a tool for numerical linear algebra.arXiv preprint arXiv:1411.4357 (2014). https://arxiv.org/pdf/1411.4357

[2] Nakatsukasa, Yuji. Fast and stable randomized low-rank matrix approximation.arXiv preprint arXiv:2009.11392 (2020). https://arxiv.org/pdf/2009.11392