Fast and Near-Optimal Matrix Completion via Randomized Basis Pursuit

Zhisu Zhu, Anthony Man-Cho So, Yinyu Ye

CSE 521: Design and Analysis of Algorithms

Lukshya Ganjoo, Ben Zhang

December 15, 2023

History

Motivation

In 2006, the Netflix Prize competition was introduced, which involved predicting user ratings based off of a large, incomplete, and sparse data set of previous ratings. Such competition sparked interest in an already growing field of *data recovery*.

Many problems today are centered around inference from datasets. Such a problem can be represented as reconstructing a matrix, with the hope of finding some structure to rely on.

A couple problems today involve **recommendation systems** (similar to the Netflix problem), which further work to optimize suggestions to increase engagement, and **recovering structure from motion**, which can be quite integral to systems such as self-driving cars and other computer vision applications.

Motivation

In 2006, the Netflix Prize competition was introduced, which involved predicting user ratings based off of a large, incomplete, and sparse data set of previous ratings. Such competition sparked interest in an already growing field of *data recovery*.

Many problems today are centered around inference from datasets. Such a problem can be represented as reconstructing a matrix, with the hope of finding some structure to rely on.

A couple problems today involve **recommendation systems** (similar to the Netflix problem), which further work to optimize suggestions to increase engagement, and **recovering structure from motion**, which can be quite integral to systems such as self-driving cars and other computer vision applications.

Motivation

In 2006, the Netflix Prize competition was introduced, which involved predicting user ratings based off of a large, incomplete, and sparse data set of previous ratings. Such competition sparked interest in an already growing field of *data recovery*.

Many problems today are centered around inference from datasets. Such a problem can be represented as reconstructing a matrix, with the hope of finding some structure to rely on.

A couple problems today involve **recommendation systems** (similar to the Netflix problem), which further work to optimize suggestions to increase engagement, and **recovering structure from motion**, which can be quite integral to systems such as self-driving cars and other computer vision applications.

Theoretical Minimum

Formally, for a rank r matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$, we are seeking to see how small our sample set Γ should be, where Γ contains indices to inspect.

We can get a sense of how many entries we need to inspect to reconstruct $\boldsymbol{A}$ through the SVD, $\boldsymbol{A} = \mathbf{U}\Sigma\mathbf{V}^\top$.

There are r degrees of freedom in specifying Σ. For the i^{th} column of $\mathbf{U}$, we may have $m - i$ degrees of freedom, due to orthonormality constraints. The same follows for $\mathbf{V}$, giving

$$\Delta = r(m + n - r) = rm + rn - r^2$$

total degrees of freedom, exactly describing a lower bound on $|\Gamma|$.

Theoretical Minimum

Formally, for a rank r matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$, we are seeking to see how small our sample set Γ should be, where Γ contains indices to inspect.

We can get a sense of how many entries we need to inspect to reconstruct $\boldsymbol{A}$ through the SVD, $\boldsymbol{A} = \mathbf{U}\Sigma\mathbf{V}^\top$.

There are r degrees of freedom in specifying Σ. For the i^{th} column of $\mathbf{U}$, we may have $m-i$ degrees of freedom, due to orthonormality constraints. The same follows for $\mathbf{V}$, giving

$$\Delta = r(m+n-r) = rm + rn - r^2$$

total degrees of freedom, exactly describing a lower bound on $|\Gamma|$.

Theoretical Minimum

Formally, for a rank r matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$, we are seeking to see how small our sample set Γ should be, where Γ contains indices to inspect.

We can get a sense of how many entries we need to inspect to reconstruct $\boldsymbol{A}$ through the SVD, $\boldsymbol{A} = \mathbf{U}\Sigma\mathbf{V}^\top$.

There are r degrees of freedom in specifying Σ. For the i^{th} column of $\mathbf{U}$, we may have $m-i$ degrees of freedom, due to orthonormality constraints. The same follows for $\mathbf{V}$, giving

$$\Delta = r(m+n-r) = rm + rn - r^2$$

total degrees of freedom, exactly describing a lower bound on $|\Gamma|$.

Previous Work & Contribution

In May 2008, Candès and Recht[1] defined *coherence*, which gives a sense of how critical information is "spread out" on the matrix.

Namely, they proposed the following algorithm:

$$\begin{aligned} \text{minimize} \quad & \|\mathbf{X}\|_* \\ \text{subject to} \quad & X_{ij} = A_{ij} \quad \text{for } (i,j) \in \Gamma \\ & \mathbf{X} \in \mathbb{R}^{m \times n} \end{aligned}$$

where $\|\mathbf{X}\|_*$ is the *nuclear norm* of $\mathbf{X}$ (the sum of its singular values).

Their paper demonstrates that if $\boldsymbol{A}$ has low coherence, then for $|\Gamma| = \Omega(N^{5/4} r \log N)$ (where $N = \max\{m, n\}$ and $r = \operatorname{rank} \boldsymbol{A}$), the solution is unique and equal to $\boldsymbol{A}$ with high probability.

Yet, the time is of order $\max\{N^{9/2} r^2 \log^2 N\,,\, N^{15/4} r^3 \log^3 N\}$.

[1] https://arxiv.org/pdf/0805.4471.pdf

Previous Work & Contribution

In May 2008, Candès and Recht[1] defined *coherence*, which gives a sense of how critical information is "spread out" on the matrix.

Namely, they proposed the following algorithm:

$$\begin{aligned} \text{minimize} \quad & \|\mathbf{X}\|_* \\ \text{subject to} \quad & X_{ij} = A_{ij} \quad \text{for } (i,j) \in \Gamma \\ & \mathbf{X} \in \mathbb{R}^{m \times n} \end{aligned}$$

where $\|\mathbf{X}\|_*$ is the *nuclear norm* of $\mathbf{X}$ (the sum of its singular values).

Their paper demonstrates that if A has low coherence, then for $|\Gamma| = \Omega(N^{5/4} r \log N)$ (where $N = \max\{m, n\}$ and $r = \operatorname{rank} \boldsymbol{A}$), the solution is unique and equal to A with high probability.

Yet, the time is of order $\max\{N^{9/2} r^2 \log^2 N\,,\, N^{15/4} r^3 \log^3 N\}$.

[1] https://arxiv.org/pdf/0805.4471.pdf

Previous Work & Contribution

In May 2008, Candès and Recht[1] defined *coherence*, which gives a sense of how critical information is "spread out" on the matrix.

Namely, they proposed the following algorithm:

$$\begin{aligned} \text{minimize} \quad & \|\mathbf{X}\|_* \\ \text{subject to} \quad & X_{ij} = A_{ij} \quad \text{for } (i,j) \in \Gamma \\ & \mathbf{X} \in \mathbb{R}^{m \times n} \end{aligned}$$

where $\|\mathbf{X}\|_*$ is the *nuclear norm* of $\mathbf{X}$ (the sum of its singular values).

Their paper demonstrates that if $\boldsymbol{A}$ has low coherence, then for $|\Gamma| = \Omega(N^{5/4} r \log N)$ (where $N = \max\{m, n\}$ and $r = \operatorname{rank} \boldsymbol{A}$), the solution is unique and equal to $\boldsymbol{A}$ with high probability.

Yet, the time is of order $\max\{N^{9/2} r^2 \log^2 N\,,\, N^{15/4} r^3 \log^3 N\}$.

[1] https://arxiv.org/pdf/0805.4471.pdf

Previous Work & Contribution

In March 2009, Candès and Tao[2] improved the bound of Γ to be of size $|\Gamma| = \Omega(Nr \log^6 N)$ for matrices with low coherence, which is only a polylogarithmic factor away from the theoretical minimum. It turns out that the logarithmic factor is due to the coupon collector's problem.

However, the runtime is still quite high – on the order of $N^4 r^2 \log^{12} N$.

In September 2009, Keshavan, Montanari, and Oh(!)[3] showed via an SVD-based algorithm that if the rank of A is bounded by $N^{1/2}$ and there are certain bounds on the singular values of A, we only need to sample $|\Gamma| = \Omega(Nr \max\{\log N\ ,\ r\})$ entries.

This algorithm has no theoretical running time bound.

[2] https://arxiv.org/pdf/0903.1476.pdf
[3] https://arxiv.org/pdf/0901.3150.pdf

Previous Work & Contribution

In March 2009, Candès and Tao[2] improved the bound of Γ to be of size $|\Gamma| = \Omega(Nr \log^6 N)$ for matrices with low coherence, which is only a polylogarithmic factor away from the theoretical minimum. It turns out that the logarithmic factor is due to the coupon collector's problem.

However, the runtime is still quite high – on the order of $N^4 r^2 \log^{12} N$.

In September 2009, Keshavan, Montanari, and Oh(!)[3] showed via an SVD-based algorithm that if the rank of $\boldsymbol{A}$ is bounded by $N^{1/2}$ and there are certain bounds on the singular values of $\boldsymbol{A}$, we only need to sample $|\Gamma| = \Omega(Nr \max\{\log N\,,\, r\})$ entries.

This algorithm has no theoretical running time bound.

[2]https://arxiv.org/pdf/0903.1476.pdf
[3]https://arxiv.org/pdf/0901.3150.pdf

Previous Work & Contribution

In this paper, Zhu, So, and Ye give a *randomized basis pursuit* algorithm, which (on a class of *stable* matrices) only needs order $\mathcal{O}(nr \log n)$ entries to reconstruct the matrix with high probability (where r is known). The running time also compares favorably to SDP-based algorithms, with time bounded by $\mathcal{O}(nr^2 \log n + n^2 r)$.

Notably, the algorithm has two core aspects:

1. Columns of the matrix are sampled at random (instead of entries), without needing bounds on the norms of each column.
2. There is no optimization involved and an *exact* reconstruction is achieved in polynomial time.

Additionally, the algorithm can be extended to reconstruct A *when the rank is not known*. When A tends to be more "stable", the running time changes minimally.

Previous Work & Contribution

In this paper, Zhu, So, and Ye give a *randomized basis pursuit* algorithm, which (on a class of *stable* matrices) only needs order $\mathcal{O}(nr \log n)$ entries to reconstruct the matrix with high probability (where r is known). The running time also compares favorably to SDP-based algorithms, with time bounded by $\mathcal{O}(nr^2 \log n + n^2 r)$.

Notably, the algorithm has two core aspects:

1. Columns of the matrix are sampled at random (instead of entries), without needing bounds on the norms of each column.
2. There is no optimization involved and an *exact* reconstruction is achieved in polynomial time.

Additionally, the algorithm can be extended to reconstruct A *when the rank is not known*. When A tends to be more "stable", the running time changes minimally.

Previous Work & Contribution

In this paper, Zhu, So, and Ye give a *randomized basis pursuit* algorithm, which (on a class of *stable* matrices) only needs order $\mathcal{O}(nr \log n)$ entries to reconstruct the matrix with high probability (where r is known). The running time also compares favorably to SDP-based algorithms, with time bounded by $\mathcal{O}(nr^2 \log n + n^2 r)$.

Notably, the algorithm has two core aspects:

1. Columns of the matrix are sampled at random (instead of entries), without needing bounds on the norms of each column.
2. There is no optimization involved and an *exact* reconstruction is achieved in polynomial time.

Additionally, the algorithm can be extended to reconstruct $\boldsymbol{A}$ *when the rank is not known*. When $\boldsymbol{A}$ tends to be more "stable", the running time changes minimally.

k-Stable Matrices

Preliminaries

For a matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$,

- We will denote the **column vectors** of $\boldsymbol{A}$ as $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_n \in \mathbb{R}^m$.
- We will denote the **row vectors** of $\boldsymbol{A}$ as $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_m \in \mathbb{R}^n$.
- We will always use r to denote the rank of a matrix.
- Let $[n]$ denote the set $\{0, 1, 2, \ldots, n\}$.

What is a k-stable matrix?

Definition 1

A matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ with rank r is said to be k-**stable** if

1. Every $\mathbb{R}^{m \times (n-k)}$ sub-matrix of $\boldsymbol{A}$ has rank r
2. There exists an $\mathbb{R}^{m \times (n-k-1)}$ sub-matrix of $\boldsymbol{A}$ with rank $r-1$.

Intuitively, we may see k as the *maximum number of columns* we can remove without disturbing the rank. Clearly $k \in [n-r]$ since we cannot remove more than $n-r$ columns without changing the rank.[4]

Definition 2

Let $\mathcal{M}^{m \times n}(k, r)$ denote the set of all k-stable rank r matrices in $\mathbb{R}^{m \times n}$, and let $\mathcal{M}^{m \times n}(k)$ be the set of all k-stable matrices in $\mathbb{R}^{m \times n}$ matrices. Both are subsets of $\mathbb{R}^{m \times n}$, and the first is a subset of the second.

[4] **Definition 1** applies to the notion of column k-stability. We can similarly define for row stability, but we will focus on column stability.

What is a k-stable matrix?

Definition 1

A matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ with rank r is said to be k-**stable** if

1. Every $\mathbb{R}^{m \times (n-k)}$ sub-matrix of $\boldsymbol{A}$ has rank r
2. There exists an $\mathbb{R}^{m \times (n-k-1)}$ sub-matrix of $\boldsymbol{A}$ with rank $r-1$.

Intuitively, we may see k as the *maximum number of columns* we can remove without disturbing the rank. Clearly $k \in [n-r]$ since we cannot remove more than $n-r$ columns without changing the rank.[4]

Definition 2

Let $\mathcal{M}^{m \times n}(k, r)$ denote the set of all k-stable rank r matrices in $\mathbb{R}^{m \times n}$, and let $\mathcal{M}^{m \times n}(k)$ be the set of all k-stable matrices in $\mathbb{R}^{m \times n}$ matrices. Both are subsets of $\mathbb{R}^{m \times n}$, and the first is a subset of the second.

[4] **Definition 1** applies to the notion of column k-stability. We can similarly define for row stability, but we will focus on column stability.

What is a k-stable matrix?

Definition 1

A matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ with rank r is said to be k-**stable** if

1. Every $\mathbb{R}^{m \times (n-k)}$ sub-matrix of $\boldsymbol{A}$ has rank r
2. There exists an $\mathbb{R}^{m \times (n-k-1)}$ sub-matrix of $\boldsymbol{A}$ with rank $r-1$.

Intuitively, we may see k as the *maximum number of columns* we can remove without disturbing the rank. Clearly $k \in [n-r]$ since we cannot remove more than $n-r$ columns without changing the rank.[4]

Definition 2

Let $\mathcal{M}^{m \times n}(k, r)$ denote the set of all k-stable rank r matrices in $\mathbb{R}^{m \times n}$, and let $\mathcal{M}^{m \times n}(k)$ be the set of all k-stable matrices in $\mathbb{R}^{m \times n}$ matrices. Both are subsets of $\mathbb{R}^{m \times n}$, and the first is a subset of the second.

[4] **Definition 1** applies to the notion of column k-stability. We can similarly define for row stability, but we will focus on column stability.

Examples

Consider the identity matrix $\mathbf{I}_n$:

$$\mathbf{I}_n = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix} \in \mathbb{R}^{n \times n}$$

Every column of $\mathbf{I}_n$ is crucial to its rank. Therefore, it is a 0-stable rank n matrix.

Examples

Consider the identity matrix $\mathbf{I}_n$:

$$\mathbf{I}_n = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix} \in \mathbb{R}^{n \times n}$$

Every column of $\mathbf{I}_n$ is crucial to its rank. Therefore, it is a 0-stable rank n matrix.

Examples

Let $\mathbf{a} \in \mathbb{R}^n$ be any vector with no zero component – for example, $a_i = i$. Then for $m = 4, n = 3$, define

$$\boldsymbol{A} = \begin{bmatrix} 1 & 2 & 3 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \in \mathbb{R}^{4\times 3}$$

Clearly, $\operatorname{rank} \boldsymbol{A} = 1$. Since we can choose any column as a valid sub-matrix with rank one $\boldsymbol{A}$ is rank one $(n-1)$-stable.

Examples

Let $\mathbf{a} \in \mathbb{R}^n$ be any vector with no zero component – for example, $a_i = i$. Then for $m = 4, n = 3$, define

$$\boldsymbol{A} = \begin{bmatrix} 1 & 2 & 3 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \in \mathbb{R}^{4\times 3}$$

Clearly, $\operatorname{rank} \boldsymbol{A} = 1$. Since we can choose any column as a valid sub-matrix with rank one $\boldsymbol{A}$ is rank one $(n-1)$-stable.

Examples

For $n = 2t + 1 \geq 1$ (n odd), let $\mathbf{1} \in \mathbb{R}^n$ be the all-ones vector, and define $\mathbf{a} \in \mathbb{R}^n$ as

$$\mathbf{a}_i = -\frac{n-1}{2} + (i-1) = -t + (i-1)\,,$$

i.e., $\mathbf{a}$ has entries ranging from $-t$ to t. For $m = 4, n = 5$, we may see

$$A = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ -2 & -1 & 0 & 1 & 2 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \in \mathbb{R}^{4\times 5}$$

has $\operatorname{rank} A = 2$. We can delete at most three columns to keep the sub-matrix with rank two, so $A \in \mathcal{M}^{4\times 5}(3, 2)$.

These are the kind of matrices we seek out. High stability and low rank gives a notion that we don't need to choose too many columns to fully collect the rank of the original matrix A.

Examples

For $n = 2t + 1 \geq 1$ (n odd), let $\mathbf{1} \in \mathbb{R}^n$ be the all-ones vector, and define $\mathbf{a} \in \mathbb{R}^n$ as

$$\mathbf{a}_i = -\frac{n-1}{2} + (i-1) = -t + (i-1)\ ,$$

i.e., $\mathbf{a}$ has entries ranging from $-t$ to t. For $m = 4, n = 5$, we may see

$$\boldsymbol{A} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ -2 & -1 & 0 & 1 & 2 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \in \mathbb{R}^{4\times 5}$$

has $\operatorname{rank} \boldsymbol{A} = 2$. We can delete at most three columns to keep the sub-matrix with rank two, so $\boldsymbol{A} \in \mathcal{M}^{4\times 5}(3, 2)$.

These are the kind of matrices we seek out. High stability and low rank gives a notion that we don't need to choose too many columns to fully collect the rank of the original matrix A.

Examples

For $n = 2t + 1 \geq 1$ (n odd), let $\mathbf{1} \in \mathbb{R}^n$ be the all-ones vector, and define $\mathbf{a} \in \mathbb{R}^n$ as

$$\mathbf{a}_i = -\frac{n-1}{2} + (i-1) = -t + (i-1)\ ,$$

i.e., $\mathbf{a}$ has entries ranging from $-t$ to t. For $m = 4, n = 5$, we may see

$$\boldsymbol{A} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ -2 & -1 & 0 & 1 & 2 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \in \mathbb{R}^{4\times 5}$$

has $\operatorname{rank} \boldsymbol{A} = 2$. We can delete at most three columns to keep the sub-matrix with rank two, so $\boldsymbol{A} \in \mathcal{M}^{4\times 5}(3, 2)$.

These are the kind of matrices we seek out. High stability and low rank gives a notion that we don't need to choose too many columns to fully collect the rank of the original matrix $\boldsymbol{A}$.

The Algorithm: Randomized Basis Pursuit

The Algorithm

Algorithm RANDOMIZED BASIS PURSUIT

Input: $\boldsymbol{A} \in \mathbb{R}^{m \times n}$, and $r = \operatorname{rank} \boldsymbol{A}$.
$\boldsymbol{A}^* \leftarrow$ empty matrix
$C \leftarrow \varnothing$
$T \leftarrow \{1, 2, \ldots, n\}$
while $T \neq \varnothing$ **or** $|C| < r$ **do**
    $j \xleftarrow{\$} T$
    Examine all the entries of $\mathbf{u}_j \in \mathbb{R}^m$ and write them to $\boldsymbol{A}^*$
    if $\mathbf{u}_j$ is in the span of the columns indexed by C **then**
        continue
    else
        $C \leftarrow C \cup \{j\}$
        $T \leftarrow T \setminus \{j\}$

The Algorithm

Algorithm RANDOMIZED BASIS PURSUIT

Input: $\boldsymbol{A} \in \mathbb{R}^{m \times n}$, and $r = \operatorname{rank} \boldsymbol{A}$.
$\boldsymbol{A}^* \leftarrow$ empty matrix
$C \leftarrow \varnothing$
$T \leftarrow \{1, 2, \ldots, n\}$
while $T \neq \varnothing$ **or** $|C| < r$ **do**
    $j \xleftarrow{\$} T$
    Examine all the entries of $\mathbf{u}_j \in \mathbb{R}^m$ and write them to $\boldsymbol{A}^*$
    if $\mathbf{u}_j$ is in the span of the columns indexed by C **then**
        continue
    else
        $C \leftarrow C \cup \{j\}$
        $T \leftarrow T \setminus \{j\}$

The Algorithm

Algorithm RANDOMIZED BASIS PURSUIT

Input: $\boldsymbol{A} \in \mathbb{R}^{m \times n}$, and $r = \operatorname{rank} \boldsymbol{A}$.
$\boldsymbol{A}^* \leftarrow$ empty matrix
$C \leftarrow \varnothing$
$T \leftarrow \{1, 2, \ldots, n\}$
while $T \neq \varnothing$ **or** $|C| < r$ **do**
    $j \overset{\$}{\leftarrow} T$
    Examine all the entries of $\mathbf{u}_j \in \mathbb{R}^m$ and write them to $\boldsymbol{A}^*$
    if $\mathbf{u}_j$ is in the span of the columns indexed by C **then**
        continue
    else
        $C \leftarrow C \cup \{j\}$
        $T \leftarrow T \setminus \{j\}$

The Algorithm

Algorithm RANDOMIZED BASIS PURSUIT ALGORITHM

13: **if** $T = \varnothing$ **then**

14: **return** A^* ▷ We have read every column of A

15: Let $A_C \in \mathbb{R}^{m \times r}$ be the sub-matrix of A with columns from C

16: Find r linearly independent rows in A_C – index them with R.

17: Let $A_{C,R} \in \mathbb{R}^{r \times r}$ be the sub-matrix from indices in C and R

18: **for** $i \in R$ **do**

19: Examine the entries in $\mathbf{v}_i \in \mathbb{R}^n$ and write them to A^*

20: **for** $j \notin C$ **do**

21: Express $(a_{ij})_{i \in R} \in \mathbb{R}^r$ as a linear combination in $\operatorname{col} A_{C,R}$

22: Let $\mathbf{u}_j^*$ be the reconstruction by taking the same linear combination with $\mathbf{u}_i$ for $i \in C$. Write the vector to A^*

23: **return** A^*

The Algorithm

Algorithm RANDOMIZED BASIS PURSUIT ALGORITHM

13: **if** $T = \varnothing$ **then**
14: **return** A^* ▷ We have read every column of A
15: Let $A_C \in \mathbb{R}^{m \times r}$ be the sub-matrix of A with columns from C
16: Find r linearly independent rows in A_C – index them with R.
17: Let $A_{C,R} \in \mathbb{R}^{r \times r}$ be the sub-matrix from indices in C and R
18: **for** $i \in R$ **do**
19: Examine the entries in $\mathbf{v}_i \in \mathbb{R}^n$ and write them to A^*
20: **for** $j \notin C$ **do**
21: Express $(a_{ij})_{i \in R} \in \mathbb{R}^r$ as a linear combination in $\operatorname{col} A_{C,R}$
22: Let $\mathbf{u}_j^*$ be the reconstruction by taking the same linear combination with $\mathbf{u}_i$ for $i \in C$. Write the vector to A^*
23: **return** A^*

The Algorithm

Algorithm RANDOMIZED BASIS PURSUIT ALGORITHM

if $T = \varnothing$ **then**
    return $\boldsymbol{A}^*$ ▷ We have read every column of $\boldsymbol{A}$
Let $\boldsymbol{A}_C \in \mathbb{R}^{m \times r}$ be the sub-matrix of $\boldsymbol{A}$ with columns from C
Find r linearly independent rows in A_C – index them with R.
Let $\boldsymbol{A}_{C,R} \in \mathbb{R}^{r \times r}$ be the sub-matrix from indices in C and R
for $i \in R$ **do**
    Examine the entries in $\mathbf{v}_i \in \mathbb{R}^n$ and write them to $\boldsymbol{A}^*$
for $j \notin C$ **do**
    Express $(a_{ij})_{i \in R} \in \mathbb{R}^r$ as a linear combination in $\operatorname{col} \boldsymbol{A}_{C,R}$
    Let $\mathbf{u}_j^*$ be the reconstruction by taking the same linear combination with $\mathbf{u}_i$ for $i \in C$. Write the vector to $\boldsymbol{A}^*$
return $\boldsymbol{A}^*$

Correctness

Proposition 1

If a matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ has rank r, then we may form an invertible submatrix $\boldsymbol{M} \in \mathbb{R}^{r \times r}$ by taking r linearly independent rows and columns from $\boldsymbol{A}$ and creating a submatrix by taking the intersection of the rows and columns.

The following an exercise from Michael Artin's Algebra text.[5] We will sketch a solution.

Sketch.

Since $\operatorname{rank} A = r$, it must have r linearly independent rows and columns. Assume without loss of generality that they are in the first r rows and columns (we may apply permutation matrices to achieve this). We may then assign M to be the upper $r \times r$ matrix of A.

[5] Chapter 4, Section 2 Exercise 2.5.

Correctness

Proposition 1

If a matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ has rank r, then we may form an invertible submatrix $\boldsymbol{M} \in \mathbb{R}^{r \times r}$ by taking r linearly independent rows and columns from $\boldsymbol{A}$ and creating a submatrix by taking the intersection of the rows and columns.

The following an exercise from Michael Artin's Algebra text.[5] We will sketch a solution.

Sketch.

Since $\operatorname{rank} \boldsymbol{A} = r$, it must have r linearly independent rows and columns. Assume without loss of generality that they are in the first r rows and columns (we may apply permutation matrices to achieve this). We may then assign $\boldsymbol{M}$ to be the upper $r \times r$ matrix of $\boldsymbol{A}$.

[5] Chapter 4, Section 2 Exercise 2.5.

Correctness

Sketch, cont.

Suppose that $\operatorname{rank} \boldsymbol{M} < r$. Then the row-rank of $\boldsymbol{M}$ is less than r. Notably, we can do row operations on $\boldsymbol{M}$ (and hence $\boldsymbol{A}$) to create an all-zero row within $\boldsymbol{M}$, i.e., for some $j \in \{1, \ldots, r\}$, we have $u_{i,j} = 0$ for all $i \in \{1, \ldots, r\}$.

Note that row operations do not impact the arrangement of the supposed linearly independent columns in $\boldsymbol{A}$. Since the first r columns are linearly independent, all other $n - r$ columns must be linear combinations of the first r columns. But since all the columns have 0 at the j^{th} index, all other columns must have 0 in the j^{th} index as well. Thus the j^{th} row of $\boldsymbol{A}$ must have been the all zero row, contradicting our construction. □

We may now proceed to show correctness, given termination.

Correctness

Sketch, cont.

Suppose that $\operatorname{rank} \boldsymbol{M} < r$. Then the row-rank of $\boldsymbol{M}$ is less than r. Notably, we can do row operations on $\boldsymbol{M}$ (and hence $\boldsymbol{A}$) to create an all-zero row within $\boldsymbol{M}$, i.e., for some $j \in \{1, \ldots, r\}$, we have $u_{i,j} = 0$ for all $i \in \{1, \ldots, r\}$.

Note that row operations do not impact the arrangement of the supposed linearly independent columns in $\boldsymbol{A}$. Since the first r columns are linearly independent, all other $n - r$ columns must be linear combinations of the first r columns. But since all the columns have 0 at the j^{th} index, all other columns must have 0 in the j^{th} index as well. Thus the j^{th} row of $\boldsymbol{A}$ must have been the all zero row, contradicting our construction. □

We may now proceed to show correctness, given termination.

Correctness

Sketch, cont.

Suppose that $\operatorname{rank} \boldsymbol{M} < r$. Then the row-rank of $\boldsymbol{M}$ is less than r. Notably, we can do row operations on $\boldsymbol{M}$ (and hence $\boldsymbol{A}$) to create an all-zero row within $\boldsymbol{M}$, i.e., for some $j \in \{1, \ldots, r\}$, we have $u_{i,j} = 0$ for all $i \in \{1, \ldots, r\}$.

Note that row operations do not impact the arrangement of the supposed linearly independent columns in $\boldsymbol{A}$. Since the first r columns are linearly independent, all other $n - r$ columns must be linear combinations of the first r columns. But since all the columns have 0 at the j^{th} index, all other columns must have 0 in the j^{th} index as well. Thus the j^{th} row of $\boldsymbol{A}$ must have been the all zero row, contradicting our construction. □

We may now proceed to show correctness, given termination.

Correctness

Sketch, cont.

Suppose that $\operatorname{rank} \boldsymbol{M} < r$. Then the row-rank of $\boldsymbol{M}$ is less than r. Notably, we can do row operations on $\boldsymbol{M}$ (and hence $\boldsymbol{A}$) to create an all-zero row within $\boldsymbol{M}$, i.e., for some $j \in \{1, \ldots, r\}$, we have $u_{i,j} = 0$ for all $i \in \{1, \ldots, r\}$.

Note that row operations do not impact the arrangement of the supposed linearly independent columns in $\boldsymbol{A}$. Since the first r columns are linearly independent, all other $n - r$ columns must be linear combinations of the first r columns. But since all the columns have 0 at the j^{th} index, all other columns must have 0 in the j^{th} index as well. Thus the j^{th} row of $\boldsymbol{A}$ must have been the all zero row, contradicting our construction. □

We may now proceed to show correctness, given termination.

Correctness

Theorem 1

If a matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ has rank r, then there exists an invertible $r \times r$ submatrix $\boldsymbol{M}$ that may be constructed by taking r linearly independent columns from $\boldsymbol{A}$, then r linearly independent rows from those columns.

Proof.

After we have selected r linearly independent columns from A, choosing r linearly independent rows from the respective $m \times r$ submatrix will also select r linearly independent rows from A (if two vectors are linearly independent in $\mathbb{R}^r$, then they will also be linearly independent when extended to $\mathbb{R}^n$ with any coefficients attached – note $n \geq r$). Applying Proposition 1 concludes the proof. □

Thus, once lines 5 through 19 are completed, the algorithm will always terminate with an exact reconstruction of A.

Correctness

Theorem 1

If a matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ has rank r, then there exists an invertible $r \times r$ submatrix $\boldsymbol{M}$ that may be constructed by taking r linearly independent columns from $\boldsymbol{A}$, then r linearly independent rows from those columns.

Proof.

After we have selected r linearly independent columns from $\boldsymbol{A}$, choosing r linearly independent rows from the respective $m \times r$ submatrix will also select r linearly independent rows from $\boldsymbol{A}$ (if two vectors are linearly independent in $\mathbb{R}^r$, then they will also be linearly independent when extended to $\mathbb{R}^n$ with any coefficients attached – note $n \geq r$). Applying Proposition 1 concludes the proof. □

Thus, once lines 5 through 19 are completed, the algorithm will always terminate with an exact reconstruction of A.

Correctness

Theorem 1

If a matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ has rank r, then there exists an invertible $r \times r$ submatrix $\boldsymbol{M}$ that may be constructed by taking r linearly independent columns from $\boldsymbol{A}$, then r linearly independent rows from those columns.

Proof.

After we have selected r linearly independent columns from $\boldsymbol{A}$, choosing r linearly independent rows from the respective $m \times r$ submatrix will also select r linearly independent rows from $\boldsymbol{A}$ (if two vectors are linearly independent in $\mathbb{R}^r$, then they will also be linearly independent when extended to $\mathbb{R}^n$ with any coefficients attached – note $n \geq r$). Applying Proposition 1 concludes the proof. □

Thus, once lines 5 through 19 are completed, the algorithm will always terminate with an exact reconstruction of $\boldsymbol{A}$.

Analysis

There is only one place in the **RBP** algorithm where randomization is used – when we draw uniformly at random from T.

Thus, it suffices to compute a probabilistic bound on the number of times the basis pursuit step occurs.

Theorem 2

Suppose that the input matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ has rank r and is k-stable for some $k \in [n - r]$, i.e. $\boldsymbol{A} \in \mathcal{M}^{m \times n}(k, r)$. Let $\delta \in (0, 1)$, then with probability at least $1 - r\delta$, the algorithm will terminate in steps at most

$$\frac{nr}{k+1}\left(1 + \ln\left(\frac{1}{\delta}\right)\right) \ .$$

Intuitively, if $\boldsymbol{A}$ has low rank but high stability, it has many candidate basis columns, and the basis pursuit step will terminate faster.

Analysis

The following estimate will be used in the proof of the main theorem:

Proposition 2

Let $X \sim \mathsf{Geo}(p)$ *where* $p \in (0, 1)$. *Then for any* $\delta > 0$,

$$\Pr\left[X > \frac{1+\delta}{p}\right] \leq e^{-\delta}$$

Analysis

Proposition 2

Let $X \sim \mathsf{Geo}(p)$ where $p \in (0,1)$. Then for any $\delta > 0$,

$$\Pr\left[X > \frac{1+\delta}{p}\right] \leq e^{-\delta}$$

Proof.

Note that $\lceil a \rceil \leq a+1$ and $\lceil a+1 \rceil = \lceil a \rceil + 1$ holds for any $a > 0$. Then

$$\begin{aligned}
\Pr\left[X > \frac{1+\delta}{p}\right] &\leq \sum_{j=\lceil (1+\delta)/p \rceil}^{\infty} p(1-p)^{j-1} \\
&= p(1-p)^{\lceil (1+\delta)/p \rceil - 1} \cdot \sum_{j=0}^{\infty} (1-p)^{j} \\
&\leq (1-p)^{\delta/p} \approx e^{-\delta}
\end{aligned}$$

□

Analysis

Proposition 2

Let $X \sim \mathsf{Geo}(p)$ where $p \in (0,1)$. Then for any $\delta > 0$,

$$\Pr\left[X > \frac{1+\delta}{p}\right] \leq e^{-\delta}$$

Proof.

Note that $\lceil a \rceil \leq a+1$ and $\lceil a+1 \rceil = \lceil a \rceil + 1$ holds for any $a > 0$. Then

$$\begin{aligned}
\Pr\left[X > \frac{1+\delta}{p}\right] &\leq \sum_{j=\lceil (1+\delta)/p \rceil}^{\infty} p(1-p)^{j-1} \\
&= p(1-p)^{\lceil (1+\delta)/p \rceil - 1} \cdot \sum_{j=0}^{\infty} (1-p)^{j} \\
&\leq (1-p)^{\delta/p} \approx e^{-\delta}
\end{aligned}$$

□

Analysis

Proposition 2

Let $X \sim \mathsf{Geo}(p)$ where $p \in (0, 1)$. Then for any $\delta > 0$,

$$\Pr\left[X > \frac{1+\delta}{p}\right] \leq e^{-\delta}$$

Proof.

Note that $\lceil a \rceil \leq a + 1$ and $\lceil a + 1 \rceil = \lceil a \rceil + 1$ holds for any $a > 0$. Then

$$\Pr\left[X > \frac{1+\delta}{p}\right] \leq \sum_{j=\lceil (1+\delta)/p \rceil}^{\infty} p(1-p)^{j-1}$$

$$= p(1-p)^{\lceil (1+\delta)/p \rceil - 1} \cdot \sum_{j=0}^{\infty} (1-p)^{j}$$

$$\leq (1-p)^{\delta/p} \approx e^{-\delta}$$

□

Proposition 2

Let $X \sim \mathsf{Geo}(p)$ where $p \in (0,1)$. Then for any $\delta > 0$,

$$\Pr\left[X > \frac{1+\delta}{p}\right] \leq e^{-\delta}$$

Proof.

Note that $\lceil a \rceil \leq a+1$ and $\lceil a+1 \rceil = \lceil a \rceil + 1$ holds for any $a > 0$. Then

$$\begin{aligned}
\Pr\left[X > \frac{1+\delta}{p}\right] &\leq \sum_{j=\lceil (1+\delta)/p \rceil}^{\infty} p(1-p)^{j-1} \\
&= p(1-p)^{\lceil (1+\delta)/p \rceil - 1} \cdot \sum_{j=0}^{\infty} (1-p)^{j} \\
&\leq (1-p)^{\delta/p} \approx e^{-\delta}
\end{aligned}$$

□

Proposition 2

Let $X \sim \mathsf{Geo}(p)$ where $p \in (0,1)$. Then for any $\delta > 0$,

$$\Pr\left[X > \frac{1+\delta}{p}\right] \leq e^{-\delta}$$

Proof.

Note that $\lceil a \rceil \leq a+1$ and $\lceil a+1 \rceil = \lceil a \rceil + 1$ holds for any $a > 0$. Then

$$\begin{aligned}
\Pr\left[X > \frac{1+\delta}{p}\right] &\leq \sum_{j=\lceil (1+\delta)/p \rceil}^{\infty} p(1-p)^{j-1} \\
&= p(1-p)^{\lceil (1+\delta)/p \rceil - 1} \cdot \sum_{j=0}^{\infty} (1-p)^{j} \\
&\leq (1-p)^{\delta/p} \approx e^{-\delta}
\end{aligned}$$

□

Analysis

Proof of Theorem 2.

Note that we may optimize the algorithm to remove j from T when we draw it. However, avoiding so further simplifies the analysis.

We will divide the execution of the basis pursuit steps into epochs. Let the i^{th} epoch start *right after* $|S| = i$; we are seeking to add the $i + 1^{\text{th}}$ basis column vector. Note $i \in \{0, \ldots, r-1\}$.

At the i^{th} epoch, the probability of sampling a valid basis column is

$$p_i \geq \frac{k+1}{n-i}$$

since $\boldsymbol{A}$ is k-stable. One may see this as: if $\boldsymbol{A}$ is k-stable and rank r, it must have $k + r$ feasible candidates such that any subset of r vectors forms a linearly independent set. Since $i = |S| < r$, we may have the above bound (there are $k + r - i \geq k + 1$ possible candidates).

Analysis

Proof of Theorem 2.

Note that we may optimize the algorithm to remove j from T when we draw it. However, avoiding so further simplifies the analysis.

We will divide the execution of the basis pursuit steps into epochs. Let the i^{th} epoch start *right after* $|S| = i$; we are seeking to add the $i + 1^{\text{th}}$ basis column vector. Note $i \in \{0, \ldots, r-1\}$.

At the i^{th} epoch, the probability of sampling a valid basis column is

$$p_i \geq \frac{k+1}{n-i}$$

since $\boldsymbol{A}$ is k-stable. One may see this as: if $\boldsymbol{A}$ is k-stable and rank r, it must have $k + r$ feasible candidates such that any subset of r vectors forms a linearly independent set. Since $i = |S| < r$, we may have the above bound (there are $k + r - i \geq k + 1$ possible candidates).

Analysis

Proof of Theorem 2.

Note that we may optimize the algorithm to remove j from T when we draw it. However, avoiding so further simplifies the analysis.

We will divide the execution of the basis pursuit steps into epochs. Let the i^{th} epoch start *right after* $|S| = i$; we are seeking to add the $i + 1^{\text{th}}$ basis column vector. Note $i \in \{0, \ldots, r-1\}$.

At the i^{th} epoch, the probability of sampling a valid basis column is

$$p_i \geq \frac{k+1}{n-i}$$

since $\boldsymbol{A}$ is k-stable. One may see this as: if $\boldsymbol{A}$ is k-stable and rank r, it must have $k + r$ feasible candidates such that any subset of r vectors forms a linearly independent set. Since $i = |S| < r$, we may have the above bound (there are $k + r - i \geq k + 1$ possible candidates).

Analysis

Proof of Theorem 2.

Note that we may optimize the algorithm to remove j from T when we draw it. However, avoiding so further simplifies the analysis.

We will divide the execution of the basis pursuit steps into epochs. Let the i^{th} epoch start *right after* $|S| = i$; we are seeking to add the $i + 1^{\text{th}}$ basis column vector. Note $i \in \{0, \ldots, r-1\}$.

At the i^{th} epoch, the probability of sampling a valid basis column is

$$p_i \geq \frac{k+1}{n-i}$$

since $\boldsymbol{A}$ is k-stable. One may see this as: if $\boldsymbol{A}$ is k-stable and rank r, it must have $k + r$ feasible candidates such that any subset of r vectors forms a linearly independent set. Since $i = |S| < r$, we may have the above bound (there are $k + r - i \geq k + 1$ possible candidates).

Proof, cont.

Define Y_i to be a random variable representing the number of times the basis pursuit step is executed in the i^{th} epoch.

Since we draw with replacement, Y_i is a geometric random variable with parameter p_i (as mentioned before).

Then the number of times the basis pursuit step is executed throughout the entire course of the algorithm is given by

$$Y = \sum_{i=0}^{r-1} Y_i \ .$$

Analysis

Proof, cont.

Define Y_i to be a random variable representing the number of times the basis pursuit step is executed in the i^{th} epoch.

Since we draw with replacement, Y_i is a geometric random variable with parameter p_i (as mentioned before).

Then the number of times the basis pursuit step is executed throughout the entire course of the algorithm is given by

$$Y = \sum_{i=0}^{r-1} Y_i \ .$$

Proof, cont.

Define Y_i to be a random variable representing the number of times the basis pursuit step is executed in the i^{th} epoch.

Since we draw with replacement, Y_i is a geometric random variable with parameter p_i (as mentioned before).

Then the number of times the basis pursuit step is executed throughout the entire course of the algorithm is given by

$$Y = \sum_{i=0}^{r-1} Y_i \ .$$

Analysis

Proof, cont.

We may define the bad event as:

$$\Pr\left[Y > \sum_{i=0}^{r-1} \frac{1+\ln(1/\delta)}{p_i}\right] = \Pr\left[\sum_{i=0}^{r-1} Y_i > \sum_{i=0}^{r-1} \frac{1+\ln(1/\delta)}{p_i}\right]$$

$$\leq \Pr\left[\bigcup_{i=0}^{r-1}\left(Y_i > \frac{1+\ln(1/\delta)}{p_i}\right)\right]$$

$$\leq \sum_{i=0}^{r-1} \Pr\left[Y_i > \frac{1+\ln(1/\delta)}{p_i}\right]$$

$$\leq \sum_{i=0}^{r-1} e^{-\ln(1/\delta)} = r\delta \ ,$$

where the last inequality follows from Proposition 2.

Analysis

Proof, cont.

We may define the bad event as:

$$\Pr\left[Y > \sum_{i=0}^{r-1} \frac{1+\ln(1/\delta)}{p_i}\right] = \Pr\left[\sum_{i=0}^{r-1} Y_i > \sum_{i=0}^{r-1} \frac{1+\ln(1/\delta)}{p_i}\right]$$

$$\leq \Pr\left[\bigcup_{i=0}^{r-1}\left(Y_i > \frac{1+\ln(1/\delta)}{p_i}\right)\right]$$

$$\leq \sum_{i=0}^{r-1} \Pr\left[Y_i > \frac{1+\ln(1/\delta)}{p_i}\right]$$

$$\leq \sum_{i=0}^{r-1} e^{-\ln(1/\delta)} = r\delta \ ,$$

where the last inequality follows from Proposition 2.

Analysis

Proof, cont.

We may define the bad event as:

$$\Pr\left[Y > \sum_{i=0}^{r-1} \frac{1+\ln(1/\delta)}{p_i}\right] = \Pr\left[\sum_{i=0}^{r-1} Y_i > \sum_{i=0}^{r-1} \frac{1+\ln(1/\delta)}{p_i}\right]$$

$$\leq \Pr\left[\bigcup_{i=0}^{r-1}\left(Y_i > \frac{1+\ln(1/\delta)}{p_i}\right)\right]$$

$$\leq \sum_{i=0}^{r-1} \Pr\left[Y_i > \frac{1+\ln(1/\delta)}{p_i}\right]$$

$$\leq \sum_{i=0}^{r-1} e^{-\ln(1/\delta)} = r\delta \ ,$$

where the last inequality follows from Proposition 2.

Proof, cont.

We may define the bad event as:

$$\begin{aligned}
\Pr\left[Y > \sum_{i=0}^{r-1} \frac{1+\ln(1/\delta)}{p_i}\right] &= \Pr\left[\sum_{i=0}^{r-1} Y_i > \sum_{i=0}^{r-1} \frac{1+\ln(1/\delta)}{p_i}\right] \\
&\leq \Pr\left[\bigcup_{i=0}^{r-1} \left(Y_i > \frac{1+\ln(1/\delta)}{p_i}\right)\right] \\
&\leq \sum_{i=0}^{r-1} \Pr\left[Y_i > \frac{1+\ln(1/\delta)}{p_i}\right] \\
&\leq \sum_{i=0}^{r-1} e^{-\ln(1/\delta)} = r\delta \,,
\end{aligned}$$

where the last inequality follows from Proposition 2.

Proof, cont.

We may define the bad event as:

$$\begin{aligned}
\Pr\left[Y > \sum_{i=0}^{r-1} \frac{1+\ln(1/\delta)}{p_i}\right] &= \Pr\left[\sum_{i=0}^{r-1} Y_i > \sum_{i=0}^{r-1} \frac{1+\ln(1/\delta)}{p_i}\right] \\
&\leq \Pr\left[\bigcup_{i=0}^{r-1}\left(Y_i > \frac{1+\ln(1/\delta)}{p_i}\right)\right] \\
&\leq \sum_{i=0}^{r-1} \Pr\left[Y_i > \frac{1+\ln(1/\delta)}{p_i}\right] \\
&\leq \sum_{i=0}^{r-1} e^{-\ln(1/\delta)} = r\delta \ ,
\end{aligned}$$

where the last inequality follows from Proposition 2.

Analysis

Proof, cont.

We now compute an upper bound on the RHS of the bad event:

$$\begin{aligned}
\sum_{i=0}^{r-1} \frac{1+\ln(1/\delta)}{p_i} &\leq \left(1+\ln\left(\frac{1}{\delta}\right)\right) \sum_{i=0}^{r-1} \frac{n-i}{k+1} \\
&= \left(1+\ln\left(\frac{1}{\delta}\right)\right) \cdot \frac{1}{k+1}\left(nr - \frac{r(r-1)}{2}\right) \\
&= \left(1+\ln\left(\frac{1}{\delta}\right)\right) \cdot \frac{r}{k+1} \cdot \underbrace{\frac{(2n-r+1)}{2}}_{\leq n \text{ for } r \geq 1} \\
&\leq \frac{nr}{k+1} \cdot \left(1+\ln\left(\frac{1}{\delta}\right)\right),
\end{aligned}$$

thus the good event will terminate within the prescribed number of steps, with probability at least $1 - r\delta$. This completes the proof. □

Analysis

Proof, cont.

We now compute an upper bound on the RHS of the bad event:

$$\sum_{i=0}^{r-1} \frac{1+\ln(1/\delta)}{p_i} \leq \left(1+\ln\left(\frac{1}{\delta}\right)\right) \sum_{i=0}^{r-1} \frac{n-i}{k+1}$$

$$= \left(1+\ln\left(\frac{1}{\delta}\right)\right) \cdot \frac{1}{k+1}\left(nr - \frac{r(r-1)}{2}\right)$$

$$= \left(1+\ln\left(\frac{1}{\delta}\right)\right) \cdot \frac{r}{k+1} \cdot \underbrace{\frac{(2n-r+1)}{2}}_{\leq n \text{ for } r \geq 1}$$

$$\leq \frac{nr}{k+1} \cdot \left(1+\ln\left(\frac{1}{\delta}\right)\right),$$

thus the good event will terminate within the prescribed number of steps, with probability at least $1 - r\delta$. This completes the proof. □

Analysis

Proof, cont.

We now compute an upper bound on the RHS of the bad event:

$$\begin{aligned}
\sum_{i=0}^{r-1} \frac{1+\ln(1/\delta)}{p_i} &\leq \left(1+\ln\left(\frac{1}{\delta}\right)\right) \sum_{i=0}^{r-1} \frac{n-i}{k+1} \\
&= \left(1+\ln\left(\frac{1}{\delta}\right)\right) \cdot \frac{1}{k+1}\left(nr - \frac{r(r-1)}{2}\right) \\
&= \left(1+\ln\left(\frac{1}{\delta}\right)\right) \cdot \frac{r}{k+1} \cdot \underbrace{\frac{(2n-r+1)}{2}}_{\leq n \text{ for } r \geq 1} \\
&\leq \frac{nr}{k+1} \cdot \left(1+\ln\left(\frac{1}{\delta}\right)\right),
\end{aligned}$$

thus the good event will terminate within the prescribed number of steps, with probability at least $1 - r\delta$. This completes the proof. □

Proof, cont.

We now compute an upper bound on the RHS of the bad event:

$$\begin{aligned}
\sum_{i=0}^{r-1} \frac{1+\ln(1/\delta)}{p_i} &\leq \left(1+\ln\left(\frac{1}{\delta}\right)\right) \sum_{i=0}^{r-1} \frac{n-i}{k+1} \\
&= \left(1+\ln\left(\frac{1}{\delta}\right)\right) \cdot \frac{1}{k+1} \left(nr - \frac{r(r-1)}{2}\right) \\
&= \left(1+\ln\left(\frac{1}{\delta}\right)\right) \cdot \frac{r}{k+1} \cdot \underbrace{\frac{(2n-r+1)}{2}}_{\leq n \text{ for } r \geq 1} \\
&\leq \frac{nr}{k+1} \cdot \left(1+\ln\left(\frac{1}{\delta}\right)\right) ,
\end{aligned}$$

thus the good event will terminate within the prescribed number of steps, with probability at least $1 - r\delta$. This completes the proof. □

Proof, cont.

We now compute an upper bound on the RHS of the bad event:

$$\begin{aligned}
\sum_{i=0}^{r-1} \frac{1+\ln(1/\delta)}{p_i} &\leq \left(1+\ln\left(\frac{1}{\delta}\right)\right) \sum_{i=0}^{r-1} \frac{n-i}{k+1} \\
&= \left(1+\ln\left(\frac{1}{\delta}\right)\right) \cdot \frac{1}{k+1} \left(nr - \frac{r(r-1)}{2}\right) \\
&= \left(1+\ln\left(\frac{1}{\delta}\right)\right) \cdot \frac{r}{k+1} \cdot \underbrace{\frac{(2n-r+1)}{2}}_{\leq n \text{ for } r \geq 1} \\
&\leq \frac{nr}{k+1} \cdot \left(1+\ln\left(\frac{1}{\delta}\right)\right) ,
\end{aligned}$$

thus the good event will terminate within the prescribed number of steps, with probability at least $1 - r\delta$. This completes the proof. □

Proof, cont.

We now compute an upper bound on the RHS of the bad event:

$$\begin{aligned}
\sum_{i=0}^{r-1} \frac{1+\ln(1/\delta)}{p_i} &\leq \left(1+\ln\left(\frac{1}{\delta}\right)\right) \sum_{i=0}^{r-1} \frac{n-i}{k+1} \\
&= \left(1+\ln\left(\frac{1}{\delta}\right)\right) \cdot \frac{1}{k+1} \left(nr - \frac{r(r-1)}{2}\right) \\
&= \left(1+\ln\left(\frac{1}{\delta}\right)\right) \cdot \frac{r}{k+1} \cdot \underbrace{\frac{(2n-r+1)}{2}}_{\leq n \text{ for } r \geq 1} \\
&\leq \frac{nr}{k+1} \cdot \left(1+\ln\left(\frac{1}{\delta}\right)\right) \ ,
\end{aligned}$$

thus the good event will terminate within the prescribed number of steps, with probability at least $1 - r\delta$. This completes the proof. □

Number of Entries

Each basis step will examine at most m entries, so at most

$$m \cdot \frac{nr}{k+1} \cdot \Big(1 + \ln(1/\delta)\Big)$$

entries will be read over the basis pursuit, as shown in Theorem 2 (this may double count duplicate reads).

After r column vectors and r linearly independent rows are selected (we do not examine additional entries), we examine r rows of A, giving at most $n \cdot r$ entries. This totals

$$nr + \frac{mnr}{k+1} \cdot \big(1 + \ln(1/\delta)\big)$$

entries examined in the entire algorithm. We see if k is large, say $k = \Theta(n)$, the number of samples required will be quite small.

Number of Entries

Each basis step will examine at most m entries, so at most

$$m \cdot \frac{nr}{k+1} \cdot \Big(1 + \ln(1/\delta)\Big)$$

entries will be read over the basis pursuit, as shown in Theorem 2 (this may double count duplicate reads).

After r column vectors and r linearly independent rows are selected (we do not examine additional entries), we examine r rows of $\boldsymbol{A}$, giving at most $n \cdot r$ entries. This totals

$$nr + \frac{mnr}{k+1} \cdot \big(1 + \ln(1/\delta)\big)$$

entries examined in the entire algorithm. We see if k is large, say $k = \Theta(n)$, the number of samples required will be quite small.

Number of Entries

Each basis step will examine at most m entries, so at most

$$m \cdot \frac{nr}{k+1} \cdot \Big(1 + \ln(1/\delta)\Big)$$

entries will be read over the basis pursuit, as shown in Theorem 2 (this may double count duplicate reads).

After r column vectors and r linearly independent rows are selected (we do not examine additional entries), we examine r rows of $\boldsymbol{A}$, giving at most $n \cdot r$ entries. This totals

$$nr + \frac{mnr}{k+1} \cdot \big(1 + \ln(1/\delta)\big)$$

entries examined in the entire algorithm. We see if k is large, say $k = \Theta(n)$, the number of samples required will be quite small.

Further Extensions

A rank-free algorithm

Indeed it turns out that extending the previous algorithm to designing a reconstruction procedure that does not need prior knowledge of the rank is quite easy!

We can instead specify a threshold Λ and if the number of attempts by the algorithm to find the next basis column exceeds Λ, we exit the basis pursuit step and proceed to the row-identification step of the algorithm.

A rank-free algorithm

Indeed it turns out that extending the previous algorithm to designing a reconstruction procedure that does not need prior knowledge of the rank is quite easy!

We can instead specify a threshold Λ and if the number of attempts by the algorithm to find the next basis column exceeds Λ, we exit the basis pursuit step and proceed to the row-identification step of the algorithm.

The Algorithm

Algorithm RANK-FREE RANDOMIZED BASIS PURSUIT ALGORITHM

Input: $\boldsymbol{A} \in \mathbb{R}^{m \times n}$, and $\Lambda \geq 1$.
$\boldsymbol{A}^* \leftarrow$ empty matrix
$C \leftarrow \varnothing$
$T \leftarrow \{1, 2, \ldots, n\}$
$\kappa \leftarrow 0$
while $T \neq \varnothing$ and $\kappa \leq \Lambda$ **do**
    $j \overset{\$}{\leftarrow} T$
    Examine all the entries of $\mathbf{u}_j \in \mathbb{R}^m$ and write them to $\boldsymbol{A}^*$
    if $\mathbf{u}_j$ is in the span of the columns indexed by C **then**
        $\kappa \leftarrow \kappa + 1$
    else
        $C \leftarrow C \cup \{j\}$
        $T \leftarrow T \setminus \{j\}$
        $\kappa \leftarrow 0$

The Algorithm

Algorithm RANK-FREE RANDOMIZED BASIS PURSUIT ALGORITHM

Let $\boldsymbol{A}_C \in \mathbb{R}^{m \times |C|}$ be the sub-matrix of $\boldsymbol{A}$ with columns from C

Find $|C|$ linearly independent rows in A_C – index them with R.

Let $\boldsymbol{A}_{C,R} \in \mathbb{R}^{|C| \times |C|}$ be the sub-matrix from indices in C and R

for $i \in R$ **do**

    Examine the entries in $\mathbf{v}_i \in \mathbb{R}^n$ and write them to $\boldsymbol{A}^*$

for $j \notin C$ **do**

    Express $(a_{ij})_{i \in R} \in \mathbb{R}^{|R|}$ as a linear combination in $\operatorname{col} \boldsymbol{A}_{C,R}$

    Let $\mathbf{u}_j^*$ be the reconstruction by taking the same linear combination with $\mathbf{u}_i$ for $i \in C$. Write the vector to $\boldsymbol{A}^*$

return $\boldsymbol{A}^*$

Sampling Complexity

We are once again interested in the sampling complexity of the **RF-RBP** algorithm.

If the input matrix is k-stable for large k, then the sampling complexity of the **RF-RBP** algorithm is comparable to that of the **RBP** algorithm. More formally,

Theorem 3

Let $\boldsymbol{A} \in \mathcal{M}^{m\times n}(k)$ for some $k \geq k_0$ and let $\delta \in (0,1)$ be given. Then, for

$$\Lambda = \log\left(\frac{\delta}{\min(m,n)}\right) / \log\left(1 - \frac{k_0+1}{n}\right) ,$$

with probability at least $1-\delta$, the above algorithm terminates with an exact reconstruction of $\boldsymbol{A}$ and the number of entries examined is at most $nr + m(r+1)\Lambda$, where $r = \operatorname{rank} \boldsymbol{A}$.

Sampling Complexity

We are once again interested in the sampling complexity of the **RF-RBP** algorithm.

If the input matrix is k-stable for large k, then the sampling complexity of the **RF-RBP** algorithm is comparable to that of the **RBP** algorithm. More formally,

Theorem 3

Let $\boldsymbol{A} \in \mathcal{M}^{m\times n}(k)$ for some $k \geq k_0$ and let $\delta \in (0,1)$ be given. Then, for

$$\Lambda = \log\left(\frac{\delta}{\min(m,n)}\right) / \log\left(1 - \frac{k_0+1}{n}\right) ,$$

with probability at least $1-\delta$, the above algorithm terminates with an exact reconstruction of $\boldsymbol{A}$ and the number of entries examined is at most $nr + m(r+1)\Lambda$, where $r = \operatorname{rank} \boldsymbol{A}$.

Sampling Complexity

We are once again interested in the sampling complexity of the **RF-RBP** algorithm.

If the input matrix is k-stable for large k, then the sampling complexity of the **RF-RBP** algorithm is comparable to that of the **RBP** algorithm. More formally,

Theorem 3

Let $\boldsymbol{A} \in \mathcal{M}^{m\times n}(k)$ for some $k \geq k_0$ and let $\delta \in (0,1)$ be given. Then, for

$$\Lambda = \log\left(\frac{\delta}{\min(m,n)}\right) / \log\left(1 - \frac{k_0+1}{n}\right) ,$$

with probability at least $1-\delta$, the above algorithm terminates with an exact reconstruction of $\boldsymbol{A}$ and the number of entries examined is at most $nr + m(r+1)\Lambda$, where $r = \operatorname{rank} \boldsymbol{A}$.

Rank-Free RBP: A probabilistic bound

Proof.

For $j \in \{1, \ldots, r\}$, let q_j be the probability that the **RF-RBP** algorithm finds at least j basis columns before giving up. We claim that

$$q_j \geq \prod_{i=1}^{j} \left(1 - \left(1 - \frac{k+1}{n-i+1}\right)^{\Lambda}\right) \quad \text{for } j \in 1, \ldots, r \ .$$

We argue via induction on j. We continue in the same fashion as we did in the proof for Theorem 2 and divide the execution of step 2 into epochs where the $(j-1)^{\text{st}}$ epoch is defined in the same way.

Additionally, let p_j be the probability that the column selected in an iteration of the $(j-1)^{\text{st}}$ epoch is a basis column. By A's k-stability,

$$p_j \geq \frac{k+1}{n-j+1} \ .$$

Rank-Free RBP: A probabilistic bound

Proof.

For $j \in \{1, \ldots, r\}$, let q_j be the probability that the **RF-RBP** algorithm finds at least j basis columns before giving up. We claim that

$$q_j \geq \prod_{i=1}^{j} \left(1 - \left(1 - \frac{k+1}{n-i+1}\right)^{\Lambda}\right) \quad \text{for } j \in 1, \ldots, r \ .$$

We argue via induction on j. We continue in the same fashion as we did in the proof for Theorem 2 and divide the execution of step 2 into epochs where the $(j-1)^{\text{st}}$ epoch is defined in the same way.

Additionally, let p_j be the probability that the column selected in an iteration of the $(j-1)^{\text{st}}$ epoch is a basis column. By A's k-stability,

$$p_j \geq \frac{k+1}{n-j+1} \ .$$

Rank-Free RBP: A probabilistic bound

Proof.

For $j \in \{1, \ldots, r\}$, let q_j be the probability that the **RF-RBP** algorithm finds at least j basis columns before giving up. We claim that

$$q_j \geq \prod_{i=1}^{j} \left(1 - \left(1 - \frac{k+1}{n-i+1}\right)^{\Lambda}\right) \quad \text{for } j \in 1, \ldots, r \ .$$

We argue via induction on j. We continue in the same fashion as we did in the proof for Theorem 2 and divide the execution of step 2 into epochs where the $(j-1)^{\text{st}}$ epoch is defined in the same way.

Additionally, let p_j be the probability that the column selected in an iteration of the $(j-1)^{\text{st}}$ epoch is a basis column. By A's k-stability,

$$p_j \geq \frac{k+1}{n-j+1} \ .$$

Rank-Free RBP: A probabilistic bound

Proof.

For $j \in \{1, \ldots, r\}$, let q_j be the probability that the **RF-RBP** algorithm finds at least j basis columns before giving up. We claim that

$$q_j \geq \prod_{i=1}^{j} \left(1 - \left(1 - \frac{k+1}{n-i+1}\right)^{\Lambda}\right) \quad \text{for } j \in 1, \ldots, r \ .$$

We argue via induction on j. We continue in the same fashion as we did in the proof for Theorem 2 and divide the execution of step 2 into epochs where the $(j-1)^{\text{st}}$ epoch is defined in the same way.

Additionally, let p_j be the probability that the column selected in an iteration of the $(j-1)^{\text{st}}$ epoch is a basis column. By $\boldsymbol{A}$'s k-stability,

$$p_j \geq \frac{k+1}{n-j+1} \ .$$

Rank-Free RBP: A probabilistic bound

Proof, cont.

Base Case: For $j = 1$, we have:

$$q_1 = 1 - (1 - p_1)^{\Lambda} \geq 1 - \left(1 - \frac{k+1}{n}\right)^{\Lambda} .$$

Inductive Step: Suppose that the claim holds for some $j < r$. Given j basis columns, the probability of obtaining $j + 1$ basis columns is

$$q_{j+1}^{\text{cond}} = 1 - (1 - p_{j+1})^{\Lambda} \geq 1 - \left(1 - \frac{k+1}{n-j}\right)^{\Lambda} .$$

Thus, we have

$$q_{j+1} = q_{j+1}^{\text{cond}} \cdot q_j \geq \prod_{i=1}^{j+1} \left(1 - \left(1 - \frac{k+1}{n-i+1}\right)^{\Lambda}\right)$$

as per the definition of conditional probability and the IH. □

Rank-Free RBP: A probabilistic bound

Proof, cont.

Base Case: For $j = 1$, we have:

$$q_1 = 1 - (1 - p_1)^{\Lambda} \geq 1 - \left(1 - \frac{k+1}{n}\right)^{\Lambda} .$$

Inductive Step: Suppose that the claim holds for some $j < r$. Given j basis columns, the probability of obtaining $j + 1$ basis columns is

$$q_{j+1}^{\text{cond}} = 1 - (1 - p_{j+1})^{\Lambda} \geq 1 - \left(1 - \frac{k+1}{n-j}\right)^{\Lambda} .$$

Thus, we have

$$q_{j+1} = q_{j+1}^{\text{cond}} \cdot q_j \geq \prod_{i=1}^{j+1} \left(1 - \left(1 - \frac{k+1}{n-i+1}\right)^{\Lambda}\right)$$

as per the definition of conditional probability and the IH. □

Rank-Free RBP: A probabilistic bound

Proof, cont.

Base Case: For $j = 1$, we have:

$$q_1 = 1 - (1 - p_1)^{\Lambda} \geq 1 - \left(1 - \frac{k+1}{n}\right)^{\Lambda} .$$

Inductive Step: Suppose that the claim holds for some $j < r$. Given j basis columns, the probability of obtaining $j + 1$ basis columns is

$$q_{j+1}^{\text{cond}} = 1 - (1 - p_{j+1})^{\Lambda} \geq 1 - \left(1 - \frac{k+1}{n-j}\right)^{\Lambda} .$$

Thus, we have

$$q_{j+1} = q_{j+1}^{\text{cond}} \cdot q_j \geq \prod_{i=1}^{j+1} \left(1 - \left(1 - \frac{k+1}{n-i+1}\right)^{\Lambda}\right)$$

as per the definition of conditional probability and the IH. □

Rank-Free RBP: Sampling Complexity

Observe that the **RF-RBP** algorithm will terminate with an exact reconstruction of $\boldsymbol{A}$ iff it finds r basis columns before proceeding to Step 3. Using the above claim and the above definition of Λ, the probability of such an event is at least

$$\prod_{i=1}^{r}\left(1-\left(1-\frac{k+1}{n-i+1}\right)^{\Lambda}\right) \geq \left[1-\left(1-\frac{k+1}{n}\right)^{\Lambda}\right]^{r}$$
$$\geq \left(1-\frac{\delta}{\min(m,n)}\right)^{\min(m,n)}$$
$$\geq 1-\delta\ .$$

The proof is completed by noting that the number of columns inspected by the above algorithm is always bounded above by $(r+1)\Lambda$, thereby implying that the total number of entries inspected by the algorithm is bounded above by $nr + m(r+1)\Lambda$.

Rank-Free RBP: Sampling Complexity

Observe that the **RF-RBP** algorithm will terminate with an exact reconstruction of $\boldsymbol{A}$ iff it finds r basis columns before proceeding to Step 3. Using the above claim and the above definition of Λ, the probability of such an event is at least

$$\begin{aligned}\prod_{i=1}^{r}\left(1-\left(1-\frac{k+1}{n-i+1}\right)^{\Lambda}\right) &\geq \left[1-\left(1-\frac{k+1}{n}\right)^{\Lambda}\right]^{r} \\ &\geq \left(1-\frac{\delta}{\min(m,n)}\right)^{\min(m,n)} \\ &\geq 1-\delta\ .\end{aligned}$$

The proof is completed by noting that the number of columns inspected by the above algorithm is always bounded above by $(r+1)\Lambda$, thereby implying that the total number of entries inspected by the algorithm is bounded above by $nr + m(r+1)\Lambda$.

Rank-Free RBP: Sampling Complexity

Observe that the **RF-RBP** algorithm will terminate with an exact reconstruction of $\boldsymbol{A}$ iff it finds r basis columns before proceeding to Step 3. Using the above claim and the above definition of Λ, the probability of such an event is at least

$$\begin{aligned}\prod_{i=1}^{r}\left(1-\left(1-\frac{k+1}{n-i+1}\right)^{\Lambda}\right) &\geq \left[1-\left(1-\frac{k+1}{n}\right)^{\Lambda}\right]^{r} \\ &\geq \left(1-\frac{\delta}{\min(m,n)}\right)^{\min(m,n)} \\ &\geq 1-\delta\ .\end{aligned}$$

The proof is completed by noting that the number of columns inspected by the above algorithm is always bounded above by $(r+1)\Lambda$, thereby implying that the total number of entries inspected by the algorithm is bounded above by $nr + m(r+1)\Lambda$.

k-stability and Coherence

Definition 3

Let $U \subset \mathbb{R}^n$ be a subspace of dimension $r \geq 1$, and let P_U be the orthogonal projection onto U. Then the **coherence** of U is defined as:

$$\mu(U) := \frac{n}{r} \max_{1 \leq i \leq n} \|P_U \mathbf{e}_i\|_2^2 \,,$$

where $\mathbf{e}_i \in \mathbb{R}^n$ is the i^{th} standard basis vector.

Theorem 4

Let $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ be a rank r matrix with SVD $\boldsymbol{A} = \mathbf{U}\Sigma\mathbf{V}^\top$, where $\mathbf{U} \in \mathbb{R}^{m \times r}, \mathbf{V} \in \mathbb{R}^{n \times r}, \Sigma \in \mathbb{R}^{r \times r}$. Then for any non-negative $k \leq n - r$, if $\mu(V) \leq \mu_0$ for some $\mu_0 \in (0, \frac{n}{rk})$, then $\boldsymbol{A}$ is column k-stable. A similar notion can be defined for row k'-stability with $\mu(U)$.

The theorem shows that coherence and k-stability are closely related.

Acknowledgements

Thanks for listening!

We'd like to thank Shayan for a great quarter and for being an excellent professor. We really enjoyed the course content and look forward to 525 :).

We'd also like to extend our gratitude to Paul, who graciously provided guidance through parts of this paper (we would've spent many more hours without his help).

We'd also like to thank our friend Ferdinand Georg **Frob**enius for somehow showing that $\|\mathbf{R}\|_F^2 \leq r - 1$ (????)

The paper we read can be found here: https://arxiv.org/pdf/0905.1546.pdf