Randomized Algorithms
in
Linear Algebra & Scientific Computing

A Short Course

by

Tamara Kolda

and

Brett Larsen

Typeset on May 13, 2024.

Acknowledgments: These exercises originated from a short course taught by Tamara Kolda at Northwestern University in May 2023 (IEMS490/COMP SIC496). Thanks to the students (some of whom were postdocs and professors) for their corrections during and between classes, including Young Cai, Shuotao Diao, Shima Dezfulian, Julia Gaudio, Charlie Guan, Addison Howe, Youngmin Ko, Jiaqi Lei, Yuchen Lou, Max Mattessich, Mohsen Mohammidi Dehchesmeh, Phawin Prongpaophan, and Dev Shah. Thanks to Juan C. Meza for reading through these notes and providing many helpful corrections. Thanks to the many professors who posted their notes for similar courses online. Emulation is the sincerest form of flattery, and we have adapted your materials (with credit) because we could not see how to do better!

Related Materials: The data files and other materials for exercises are available online at https://github.com/tgkolda/randalgslabs/.

Citation: Tamara G. Kolda and Brett W. Larsen, *Randomized Algorithms in Linear Algebra & Scientific Computing*, https://github.com/tgkolda/randalgslabs/, May 2024

Chapter 1

Lab: Gaussian random matrices

To generate a Gaussian random vector or matrix in MATLAB, there are two equivalent methods. Using the Statistics Toolbox, the following generates and $m \times n$ matrix whose entries are $\mathcal{N}(0, v)$:

```
A = normrnd(0,sqrt(v),m,n);
```

Otherwise, with standard MATLAB, use

```
A = sqrt(v) * randn(m,n);
```

Exercise 1.1 (Adapted from Martinsson, 2020) Set a fixed vector $\mathbf{v} \in \mathbb{R}^n$ with $n = 100$ from a uniform random distribution on $[-1, 1]$, a constant vector, or something else which seems interesting. (Don't use a normal distribution, which makes it too easy to estimate.)

1. Compute $\mu = \|\mathbf{v}\|_2^2$.
2. Let $Y = \boldsymbol{\omega}^\intercal \mathbf{v}$ be a random variable where $\boldsymbol{\omega} \sim \mathcal{N}(0, 1)^n$. Empirically estimate the mean and variance of Y.
3. Choose $d > 1$. Let $Z = \|\boldsymbol{\Omega}\mathbf{v}\|_2$ where $\boldsymbol{\Omega} \in \mathbb{R}^{m \times n}$ and $\omega_{ij} \sim \mathcal{N}(0, 1/m)$. Empirically estimate the mean and variance of Z. (Bonus: Derive the theoretical mean.)
4. How do these quantities compare?

Exercise 1.2 Let $\boldsymbol{X}$ be an $m \times n$ matrix whose entries are independent $\mathcal{N}(0, 1/m)$. Use, e.g., $m = 200, n = 150$.

1. What is $\mathbb{E}\big[\|\boldsymbol{X}\|_2\big]$?
2. Run a sequence of simulations and verify this claim.
3. What is the probability that $\|\boldsymbol{X}\|_2 > 1.01$?
4. Run a sequence of simulations and note the failure rate. How does it compare?

References

Martinsson, Per-Gunnar (2020). *Randomized methods in linear algebra and their applications in data science*. URL: http://users.oden.utexas.edu/~pgm/2020_kth_course/.

Chapter 2

Lab: RRF

First, we see that using SVD and QR provide the same subspaces via the following exercise. To compute the left singular vectors of an economy SVD, do

```
[U,~,~] = svd(A,0);
```

An alternate method (using the matrix QR decomposition) that finds the same subspace is

```
[Q,~,~] = qr(A,0);
```

Exercise 2.1 Let $\mathbf{A}$ be any full-rank matrix of size, say, 100×5. (Most random matrices of these dimensions will be full rank per Gordon's theorem!) Computing the column span using both SVD and QR. Are the matrices the same? Are the projectors (e.g., $\mathbf{U}\mathbf{U}^\mathsf{T}$ and $\mathbf{Q}\mathbf{Q}^\mathsf{T}$) the same? Which is faster to compute? (If you cannot see a speed difference, try size 500×50.)

We will try RRF on some matrices. To create an approximately low-rank matrix with fast decay in singular values, we recommend the following code adopted from P.-G. Martisson:

Create approximate low-rank random matrix

```
[U,~,~] = qr(randn(m,k),0);
[V,~,~] = qr(randn(n,k),0);
A = U*diag(1:k)*V';
Noise = 0.1/(sqrt(m)+sqrt(n)) * randn(m,n);
A = A + Noise;
```

Exercise 2.2 Let's compare the speed an quality of RRF (without and with power iterations) to computing the optimal subspace via SVD.

1. Write RRF with optional power iterations.
2. Create a 500×500 matrix that is approximately low rank, e.g., $r \approx 25$ (see code above).
3. Compute (and time) the SVD of this matrix.
4. For values of k ranging from $5, 10, \ldots, 4r$, compute an approximate subspace using both SVD (optimal) and RRF with $p = 0$ and 0,1, or 2 power iterations. Time the RRF computations.
5. Plot the subspace error versus k (maybe using log scaling on the y-axis). How do the methods compare? Approximately what does p need to be for RRF to match the subspace accuracy of the optimal method?
6. Plot the computation time versus k. How do the methods compare on this metric?

Exercise 2.3 Bonus: Try this out with other matrices of interest in your own research, etc.

Exercise 2.4 Try out other random matrices besides Gaussians. What do you observe?

Chapter 3

Lab: RSVD

We obtained the Jupiter image from https://planetary.s3.amazonaws.com/assets/images/5-jupiter/20130714_jupiter_cassini_20001229_sc_0-060-000_f840.jpg. You can download it from https://github.com/tgkolda/randalgslabs/ under lecture 4 labs.

To load the image, convert it to grayscale, and display it, use the following commands in MATLAB. This requires the Image Processing Toolbox.

```
%% Load jupiter pic and view
A=imread('20130714_jupiter_cassini_20001229_sc_0-060-000_f840.jpg');
A=double(rgb2gray(A));
figure(1); clf;
imagesc(A), axis equal, axis off, colormap gray
title('original')
```

The lab will apply SVD and RSVD to this image, of size 1097×840.

Exercise 3.1 (Adapted from Brunton and Kutz (2019)) Let $\mathbf{A}$ be the 1097×840 matrix that represents the Jupiter image above.

1. Compute the economy SVD of $\mathbf{A}$. How long does it take?
2. Using the SVD, compute the best rank-100 approximation. What is the error?
3. View the image created by the SVD approximation. How does it compare to the original?
4. Compute the rank-100 randomized SVD with $p = 10$ and $q = 0$. What is the error? How does it compare to the full SVD in terms of compute time and error?
5. View the image created by the RSVD approximation. How does it compare to the original?
6. Compute the rank-100 randomized SVD with $p = 10$ and $q = 1$. What is the error? How does it compare to the full SVD in terms of compute time and error?
7. View the image created by the RSVD approximation. How does it compare to the original?
8. MATLAB has a function called `svdsketch`. Try this with a tolerance of 0.1. What is the rank of the result? What is the error? How long does it take to compute? How do these compare?
9. View the SketchSVD approximation. How does it compare to the original?

References

Brunton, S. L. and J. N. Kutz (2019). *Data Driven Science & Engineering: Machine Learning, Dynamical Systems, and Control*. Cambridge University Press.

Chapter 4

Lab: Randomized Least Squares

The data from this lab can be downloaded from https://github.com/tgkolda/randalgslabs/ under lecture 6 labs.

The MATLAB command to compute a weighted sample according to distribution $\mathbf{p} \in \mathbb{R}^n$ as

$$\xi_k = \text{RANDSAMPLE}(s, \mathbf{p}) \quad \text{for all} \quad k \in [s],$$

do the following:

```
xi = randsample(n,s,true,p);
```

Exercise 4.1 Load the file `testprobsparse.mat`. It contains a sparse matrix $\mathbf{A}$ of size 20000×200 with $\text{nnz}(\mathbf{A}) = 379910$. It also has a vector $\mathbf{b} \in \mathbb{R}^{5000}$.

1. Measure the time to compute the least squares solution using the MATLAB backslash operator, i.e., `xopt = A \ b`. (The `linsolve` method doesn't work with sparse matrices.)
2. Measure the time to compute the row norms of $\mathbf{A}$, i.e., via `rownormssqr=full(vecnorm(A')).^2`.
3. For varying numbers of samples (50, 100, ..., 2500), time and compute the least squares solution of the sketched problem. How does the time compare (including time for computing the column norms) to the full solution? How do the residuals compare, i.e., $\|\mathbf{Ax} - \mathbf{b}\|$.
4. How would uniform random sampling do?
5. Compute the misestimation factor β for the probabilities derived from normalizing the row norms squared (you will need to first compute the leverage scores of $\mathbf{A}$).