Large Matrix Manipulation for Unfolding Optimization

Shih-Kai Lin

Colorado State University

April 30, 2019

Overview

- Currently, the number of iterations we use for unfolding was optimized in 1D neutrino energy only, but not in the full 3D $(\cos\theta_\mu, T_\mu, E_\nu)$ space. We would like to re-optimize the number of iterations in the full 3D space.
- So far, I have been following other analyses to use RooUnfold for calculating the bin-to-bin covariance matrix.
- The total number of bins in $(\cos\theta_\mu, T_\mu, E_\nu)$ is $14 \times 22 \times 22 = 6776$. As a result, the covariance matrix is of size 6776×6776.
- It turns out large matrices are hard to manipulate numerically.
- This study is done with `RyROOT`, `root_numpy`, and `numpy.linalg`.

Welcome to numerical linear algebra!

Difficulties

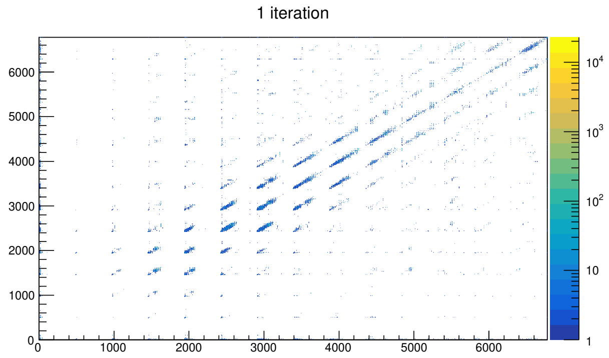
1. Using RooUnfold to calculate the covariance matrix takes exceedingly long time.
   - `RooUnfold::Ereco()`[1] with option `RooUnfold::ErrorTreatment::kCovariance` is used to obtain the covariance matrix.
   - We break down the time spent, and found doing more iterations takes no time, but retrieving the covariance matrix takes a full day. We think 10^8 operations should be easy for modern computers.
   - We might want to look at tens of different iterations, and the time spent will become unmanageable.
2. The metric for optimization involves a matrix inversion, and we found the covariance matrix are all nearly singular.
   - When I calculate the inverse matrix, singular matrix warning messages are printed on the terminal reminding me of large numerical errors.

Strategy:

1. RooUnfold might be doing something inefficient. We plan to implement our own covariance matrix calculation.
2. I looked into the singular matrix problem and summarize what I learned in this talk.

[1]The only other public method I know of, `RooUnfold::GetMeasuredCov()`, gives me an empty matrix.

Visualizing a Covariance Matrix



- Log scale is used to make the structure more visible.
- Obviously this matrix is singular due to rows and columns with elements all zero. Those are unoccupied bins and unavoidable with our binning.

Metric for Iteration Optimization and Inverting (Nearly) Singular Matrices

- For the metric, we use average global correlation coefficient[1].

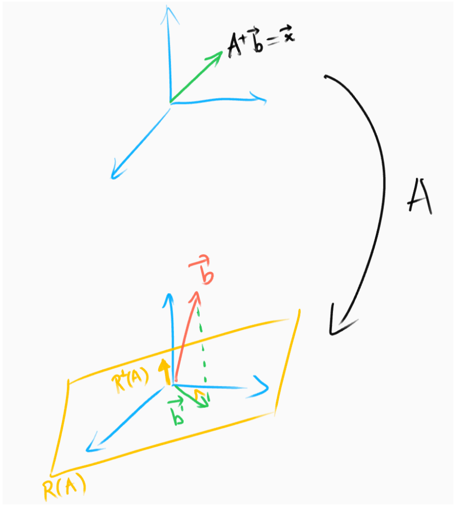
$$\rho_{avg} = \frac{1}{N}\sum_{j=1}^{N}\rho_j$$
$$\rho_j = \sqrt{1 - \frac{1}{[\mathbf{V}]_{jj}[\mathbf{V}^{-1}]_{jj}}}$$

, where j runs through all bins and $\mathbf{V}$ the covariance matrix.

- Since $\mathbf{V}$ is singular, I have tried several ways to deal with this.
- I tried removing the zero rows and columns. I will talk about this later.
- I also tried pseudoinverse (Moore-Penrose inverse). I will not define it, but cite important properties here.
    1. For any matrix $\mathbf{A}$, there exists a unique pseudoinverse of it, denoted $\mathbf{A}^{+}$.
    2. If $\mathbf{A}$ is invertible, $\mathbf{A}^{-1} = \mathbf{A}^{+}$.

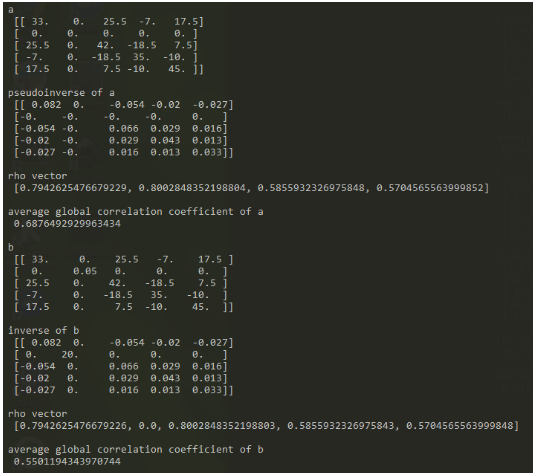
[1] Data Unfolding Methods in High Energy Physics

Geometric Meaning of Pseudoinverse



- Given a linear equation $\mathbf{Ax} = \mathbf{b}$, $\mathbf{A}^+\mathbf{b}$ is the least square solution to the problem. That is, $\mathbf{x} = \mathbf{A}^+\mathbf{b}$ minimizes $||\mathbf{Ax} - \mathbf{b}||^2$.
- Geometrically, it first project orthogonally the vector $\mathbf{b}$ to the subspace, range of $\mathbf{A}$, and invert from there.
- Algorithmically, it is obtained effortlessly from singular value decomposition (SVD).

Playing with Pseudoinverse



```
a
 [[ 33.    0.   25.5  -7.   17.5]
 [  0.    0.    0.    0.    0. ]
 [ 25.5   0.   42.  -18.5   7.5]
 [ -7.    0.  -18.5  35.  -10. ]
 [ 17.5   0.    7.5 -10.   45. ]]

pseudoinverse of a
 [[ 0.082  0.    -0.054 -0.02  -0.027]
 [-0.    -0.    -0.    -0.     0.   ]
 [-0.054 -0.     0.066  0.029  0.016]
 [-0.02  -0.     0.029  0.043  0.013]
 [-0.027 -0.     0.016  0.013  0.033]]

rho vector
 [0.7942625476679229, 0.8002848352198804, 0.5855932326975848, 0.5704565563999852]

average global correlation coefficient of a
 0.6876492929963434

b
 [[ 33.     0.    25.5   -7.    17.5 ]
 [  0.     0.05   0.     0.     0.  ]
 [ 25.5    0.    42.   -18.5    7.5 ]
 [ -7.     0.   -18.5   35.   -10.  ]
 [ 17.5    0.     7.5  -10.    45.  ]]

inverse of b
 [[ 0.082  0.    -0.054 -0.02  -0.027]
 [ 0.    20.     0.     0.     0.   ]
 [-0.054  0.     0.066  0.029  0.016]
 [-0.02   0.     0.029  0.043  0.013]
 [-0.027  0.     0.016  0.013  0.033]]

rho vector
 [0.7942625476679226, 0.0, 0.8002848352198803, 0.5855932326975843, 0.5704565563999848]

average global correlation coefficient of b
 0.5501194343970744
```

1. I generated a random symmetric matrix $\mathbf{A}$, and manually set the second row and column to zero.
2. The pseudoinverse of $\mathbf{A}$ is calculated.
3. The $\boldsymbol{\rho}$ vector of $\mathbf{A}$ is calculated.
4. A second matrix $\mathbf{B}$ is generated by copying $\mathbf{A}$ and setting the 0 on the diagonal to 0.05. (Following Matt Judah)
5. The inverse of $\mathbf{B}$ is calculated.
6. The $\boldsymbol{\rho}$ vector of $\mathbf{B}$ is calculated.

- The two results are identical except where they are different to start with.
- I did the bench testing because even I made the zero diagonal elements 0.05 in the real covariance matrices, they were still singular.

Nonphysical Values of ρ_j's

- As the paper and the name (correlation coefficient) suggest, a valid ρ_j should be in the range

$$0 \leq \rho_j = \sqrt{1 - \left(\mathbf{V}_{jj}\mathbf{V}_{jj}^{-1}\right)^{-1}} \leq 1 \tag{1}$$

- Eq. (1) is satisfied if and only if

$$\mathbf{V}_{jj}\mathbf{V}_{jj}^{-1} \geq 1 \tag{2}$$

- I went on to use pseudoinverse to calculate ρ_{avg}, and found that only a fraction of all ρ_j's satisfy Eq. (2).
- I started to suspect that the inverse process involved is numerically unstable.

Condition Number[2] and $\mathbf{V}_{jj}\mathbf{V}_{jj}^{-1}$

- Consider the linear equation $\mathbf{A}\mathbf{x} = \mathbf{b}$. If small changes in $\mathbf{b}$ imply small changes in the solution $\mathbf{x}$, then $\mathbf{A}$ has a small condition number. Otherwise $\mathbf{A}$ has a big condition number.
- Not surprisingly, nearly singular matrices have large condition numbers.
- Since I don't know how to prove Eq. (2) mathematically, I did what an experimentalist does: do some experiments.
- I generated random positive semidefinite matrices to simulate covariance matrices, and look at their condition numbers and $\mathbf{V}_{jj}\mathbf{V}_{jj}^{+}$'s.

[2] Smallness of $det(\mathbf{A})$ is not a good measure for invertibility. For example, $10^{-50}\mathbf{I}_5$ has extremely small determinant, but is readily invertible with a condition number 1.

Condition Number and $\mathbf{V}_{jj}\mathbf{V}_{jj}^{-1}$ (Cont.)

```
        well-conditioned example

positive semidefinite matrix V
[[ 80.844  101.687   -7.097    3.693  -37.207]
 [101.687  227.373  -87.639   15.71   -42.425]
 [ -7.097  -87.639  223.402   64.78    61.958]
 [  3.693   15.71    64.78   118.352   24.211]
 [-37.207  -42.425   61.958   24.211   78.489]]

condition number of V
34.26

V_ii*(V^+)_ii
[4.675  4.628  2.953  1.431  2.122]
```

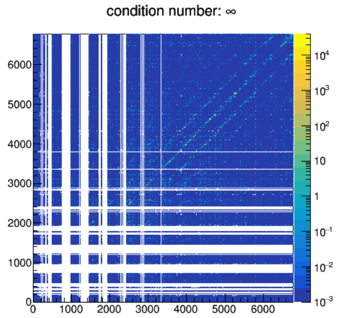
```
        ill-conditioned example

positive semidefinite matrix V
[[ 53.276  -19.187  -71.864  -33.168  -44.791]
 [-19.187   10.654   23.82     3.057   17.234]
 [-71.864   23.82    98.07    49.632   59.81 ]
 [-33.168    3.057   49.632   41.747   25.265]
 [-44.791   17.234   59.81    25.265   37.982]]

condition number of V
1.582e+17

V_ii*(V^+)_ii
[0.13  0.097  0.204  1.237  0.142]
```

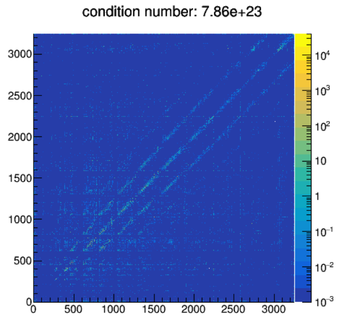
Out of those tens of random positive semidefinite matrices, they showed invariably that,

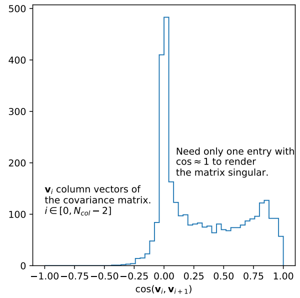
- If the condition number is small (say, < 1000), $\mathbf{V}_{jj}\mathbf{V}_{jj}^{+} \geq 1$ is always true for all j.
- If the condition number is large (i.e., in scientific notation), $\mathbf{V}_{jj}\mathbf{V}_{jj}^{+} \geq 1$ is not satisfied for some or all j.

Bottom line: $\mathbf{V}_{jj}\mathbf{V}_{jj}^{+} \geq 1$ is true for all j only if $\mathbf{V}$ is well-conditioned.

Removing Zero Rows and Columns Does Not Help!





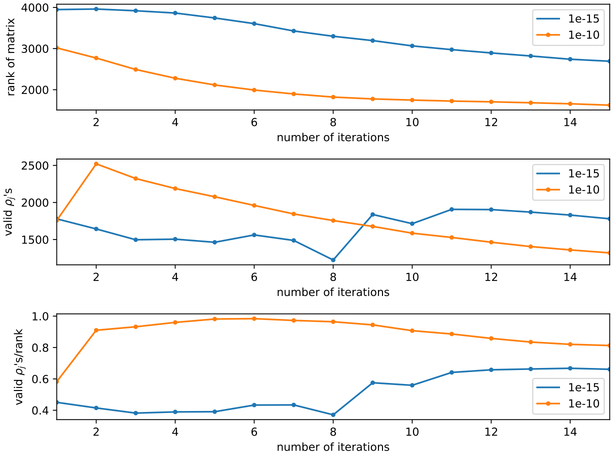


- Plots are with 2 iterations. Similar situation applies to all iterations.
- The reason is a large matrix becomes nearly singular if it contains highly correlated rows or columns.
- The adjacent columns of a covariance matrix are very often highly correlated.
- Of course, smaller matrices are always better. Will move to using the zero suppressed matrices.

The Rank and Inverse of a Large Matrix Actually Depend on a Small Tolerance!

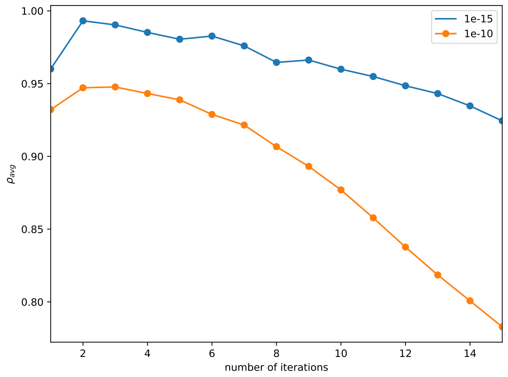
- Intuitively, the number of valid ρ_j's should not exceed the rank of the matrix. Therefore, knowing the rank of the matrix helps understand the problem.
- Turns out the rank of a large matrix depends on a small tolerance ϵ. Singular values obtained by SVD that are smaller than ϵ are set to exact zero to avoid numerical instability. This is called low-rank approximation (or truncated SVD), and regarded as a way of regularizing the linear system.
- The same tolerance applies to obtaining the pseudoinverse of the matrix since the most common algorithm used to calculated the pseudoinverse is using SVD.
- I am going to show you the ranks, the ρ_{avg}'s among valid ρ_j's of the covariance matrices as a function of number of iterations and compare results obtained with two different tolerance values.

Different Tolerance Values Result in Different Number of Valid ρ_j



- Different iterations give different covariance matrices.
- With $\epsilon = 10^{-15}$, some matrices have ranks larger than the size of the zero-suppressed counterparts! This is evidence of numerical instability.
- A criterion for how well a matrix is regularized in this study can be *the ratio of the number of valid ρ_j's to the rank*. In this sense, $\epsilon = 10^{-10}$ should give a more reliable measure.

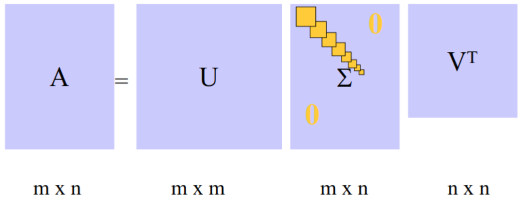
ρ_{avg} Results, Work to Do



- Although the results depend on tolerance, the *trends* with iterations are the same.
- Obviously I need to extend the plots to at least 100. I will sample iterations in $[1, 1000]$ and see if a minimum exists.
- I have to run the analysis on grid, or implement a much faster covariance matrix calculation algorithm.
- If there are any other iteration optimization metrics that are much easier to wield, I am happy to switch over...

Backup

My Working Knowledge on Singular Value Decomposition



r = the rank of A
= number of linearly independent columns/rows

- $\mathbf{U}$ and $\mathbf{V}$ are unitary (orthogonal) matrices. $\mathbf{\Sigma}$ is diagonal with elements $\sigma_1 \geq ... \geq \sigma_{min(m,n)} \geq 0$. σ_i are called singular value.
- Rank of $\mathbf{A}$ equals the number of non-zero singular values.
- Pseudoinverse of $\mathbf{A}$ is $\mathbf{V}\mathbf{\Sigma}^{+}\mathbf{U}^{T}$. $\mathbf{\Sigma}^{+}$ is the reciprocal of each non-zero element on the diagonal of $\mathbf{\Sigma}$, leaving the zeros in place, and then transposing the matrix.