Regularized Matrix Computations

Andrew E. Yagle
Department of EECS, The University of Michigan, Ann Arbor, MI 48109-2122

Abstract— **We review the basic results on: (1) the singular value decomposition (SVD); (2) sensitivity and conditioning of solutions of linear systems of equations; (3) regularization; and (4) iterative solution of linear systems of equations. These are applied to the specific problem of computing a matrix null vector.**

I. SINGULAR VALUE DECOMPOSITION

A. Basics

The singular value decomposition (SVD) of any $(M \times N)$ matrix A is

$$\begin{array}{c} A = USV^H \\ U^HU = I \\ U \text{ is } M \times M \\ \\ V^HV = I \\ V \text{ is } N \times N \\ \sigma_1 \geq \ldots \geq \sigma_N \\ S \text{ is } M \times N \end{array} \qquad S = \begin{bmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \sigma_N \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix} \tag{1}$$

- S is depicted in (1) for $M > N$ ("tall" matrix A);
- If $M < N$ use the transpose of the S in (1) (for a "reclining" matrix A);
- Important: S has the same dimensions as A;
- This also specifies the sizes of U and V.

Geometrically, the SVD shows that any linear operator can be regarded as the following:

1. An orthonormal rotation V^H, followed by
2. Scaling the rotated axes by σ_i, where
3. Singular values $\sigma_i \geq 0$, followed by
4. An orthonormal rotation U.

An example of a singular value decomposition:

$$\begin{bmatrix} 1.704 & 0.128 \\ -.928 & 1.104 \end{bmatrix} = \begin{bmatrix} .8 & .6 \\ -.6 & .8 \end{bmatrix} \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} .96 & .28 \\ -.28 & .96 \end{bmatrix}^T$$

B. Computation

The SVD can be computed by solving the following two eigenvalue problems:

$$\begin{array}{l} AA^H = (US^HV^H)(VSU^H) = U(S^HS)U^H \\ A^HA = (VS^HU^H)(USV^H) = V(S^HS)V^H \\ \quad \rightarrow (AA^H)u_i = \sigma_i^2 u_i \text{ and } (A^HA)v_i = \sigma_i^2 v_i \end{array} \tag{2}$$

- U is the matrix of eigenvectors for AA^H;
- V is the matrix of eigenvectors for A^HA;
- This is why $A = USV^H$ instead of $A = USV$;
- AA^H and A^HA have same positive eigenvalues σ_i^2
- If $M \neq N$, there are some extra zero eigenvalues.

Although this is not how the SVD is actually computed, it will do here.

Using Matlab `[U,S,V]=svd(A)` changes the signs of the last column of U and the last row of V. You should be able to see why this makes no difference.

The *null vector* of overdetermined A is the last column of V if $M > N$ and the singular values are in decreasing order (usually, but check first!).

For convenience, this paper will omit pathological cases such as repeated eigenvalues, defective (non-diagonalizable) matrices, and the like.

II. APPLICATIONS OF THE SVD

Inserting the SVD $A = USV^H$ into linear system $Ax = b$ results in

$$Ax = (USV^H)x = b \rightarrow S(V^Hx) = (U^Hb) \tag{3}$$

This gines the solutions to the following 4 problems.

A. Over-Determined Systems

If $\mathbf{M > N}$, the system $Ax = b$ is *overdetermined* (more equations than unknowns). Then there is almost surely no solution to the system $Ax = b$. However the x that minimizes (recall U is unitary)

$$||Ax-b|| = ||U^H(USV^H)x-U^Hb|| = ||S(V^Hx)-U^Hb|| \tag{4}$$

is easily seen to be the solution to *the first N rows* of (3), since we can solve the first N rows exactly, while the last $M-N$ rows will always have zero on the left side, so the best we can do is replace the right side with zero as well.

To obtain a closed-form solution, premultiply (3) by VS^H. This multiplies the last $M-N$ rows by zero, yielding

$$\begin{array}{r} S(V^Hx) = (U^Hb) \rightarrow (V(S^HS)V^H)x = (VS^HU^H)b \\ \rightarrow (A^HA)x = A^Hb \end{array} \tag{5}$$

which is the pseudo-inverse or Penrose inverse.

A special case is finding the **null vector** of an overdetermined system. We solve $(A^H A)x = 0$, i.e., find the eigenvector of $A^H A$ associated with its zero eigenvalue. From the definition of SVD above, x is the column V_N of V (NOT V^H) associated with minimum singular value $\sigma_N = 0$. Indeed, the **rank** of A can be found by counting the number of non-zero σ_i.

B. Under-Determined Systems

If $\mathbf{M} = \mathbf{N}$ and $\sigma_N > 0$, then the system is nonsingular and the solution is

$$x = V\text{diag}[1/\sigma_1, 1/\sigma_2 \ldots 1/\sigma_N]U^H b \tag{6}$$

"If real life were only like this!"–Woody Allen line in *Annie Hall* (Oscar winner for Best Picture of 1977, beating out *Star Wars*).

If $\mathbf{M} < \mathbf{N}$, the system $Ax = b$ is *underdetermined* (more unknowns than equations). Then there is an infinite number of solutions. However, the x that minimizes $||x||$ has the final $(N - M)$ values of $V^H x$ equal to zero, since these values get multiplied by zero in (3) anyways, and having them be nonzero would only increase $||x||$.

To obtain a closed-form solution, premultiply (3) by $VS^H(U^H U)(SS^H)^{-1}$, yielding

$$\begin{aligned} S(V^H x) = (U^H b) &\rightarrow V[S^H(SS^H)^{-1}S]V^H x \\ &= [VS^H U^H][U(SS^H)^{-1}U^H]b \end{aligned} \tag{7}$$

which, with the artful placement of parentheses above, simplifies to

$$V\begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} V^H x = x = A^H(AA^H)^{-1}b \tag{8}$$

since S is diagonal and the final $(N - M)$ values of $V^H x$ are constrained to be zero. I haven't seen that formula since my graduate school days. Implement:

$$(AA^H)z = b; \quad x = A^H z \rightarrow V^H x = S^H U^H z \tag{9}$$

so the final $(N - M)$ values of x will indeed be zero.

III. SENSITIVITY AND CONDITIONING

A. What's the Problem?

Consider the solution to the linear system

$$\frac{1}{60}\begin{bmatrix} 60 & 30 & 20 \\ 30 & 20 & 15 \\ 20 & 15 & 12 \end{bmatrix}\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} b \\ c \\ d \end{bmatrix} \tag{10}$$

The matrix is only (3×3), is symmetric, and has integer elements. But note that while

$$\begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} 110 \\ 65 \\ 47 \end{bmatrix} \rightarrow \begin{bmatrix} x \\ y \\ z \end{bmatrix} \rightarrow \begin{bmatrix} 60 \\ 60 \\ 60 \end{bmatrix} \tag{11}$$

a slight change in the right side yields

$$\begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} 111 \\ 65 \\ 47 \end{bmatrix} \rightarrow \begin{bmatrix} x \\ y \\ z \end{bmatrix} \rightarrow \begin{bmatrix} 69 \\ 24 \\ 90 \end{bmatrix} \tag{12}$$

Slightly different right sides yield very different solutions! This means:

- The solution to this problem will be sensitive to noise in the data [b c d]';
- Worse, how do we know whether the *true* data is b=110 or b=111?
- The very concept of a "right" answer is in question!

In fact, even though the matrix is nonsingular, for practical purposes it may as well be singular, since there are any number of solutions that could be the "right" one.

Another example, which illustrates some ideas:

$$\begin{bmatrix} 1 & 1000 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1.00 \\ 0.00 \end{bmatrix} \rightarrow \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \tag{13}$$

But changing the right side "data" slightly gives

$$\begin{bmatrix} 1 & 1000 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1.00 \\ 0.01 \end{bmatrix} \rightarrow \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} -9 \\ 0.01 \end{bmatrix} \tag{14}$$

Is data so good that an error of 0.01 is impossible?

B. Symptoms of the Problem

That 1000 suggests a problem, but in fact that by itself isn't it. How can we determine when a problem will be super-sensitive?

These look like symptoms, but aren't:

- Large elements like 1000? No–plenty of matrices have large elements without being super-sensitive;
- The determinant? No–the determinant=1;
- The eigenvalues? No–both eigenvalues are one (just like the identity matrix).

These don't look like symptoms, but are:

- Although the matrix has determinant=1, changing the lower left value from 0 to 0.001 makes the determinant=0 and the eigenvalues 0 and 2;
- The singular values are 0.001 and 1000, and their ratio is 1 million.

C. Condition Number

We have the following two results on the sensitivity of the solution to a linear system of equations to perturbations in either the right-hand-side or elements of the matrix. Define the *condition number*

$$\kappa(A) = \text{cond}(A) = ||A|| \cdot ||A^{-1}|| = \sigma_1/\sigma_N \qquad (15)$$

$||A|| = ||A||_2 = \sigma_1$=maximum singular value of A. For the above matrix, $\kappa(A) = \frac{1000}{0.001} = 10^6$. Now:

- Consider the linear system $Ax = b$:
- Perturb A to $A + \Delta A$ **or:**
- Perturb b to $b + \Delta b$ resulting in
- Perturb x to $x + \delta x$ so that
- $(A + \Delta A)(x + \delta x) = b$ **or** $A(x + \delta x) = (b + \delta b)$.
- Then the normalized=relative=percentage error in the computed x is [1,p.194-5]:

$$\begin{aligned} ||\delta x||/||x|| \leq & \quad \kappa(A)||\delta b||/||b|| \\ ||\delta x||/||x|| \leq & \quad \kappa(A)||\Delta A||/||A|| \end{aligned} \qquad (16)$$

That is, the condition number $\kappa(A)$ of A is a magnifying factor for perturbations in A or b. Unitary or orthogonal matrices $A^H = AA^H = I$ have $\kappa(A) = 1$, which is the best possible. For example, the DFT matrix has $\kappa(A) = 1$, so computing the inverse DFT does not amplify errors at all.

A linear system with a relatively small (say $\kappa(A) < 1000$) condition number is *well-conditioned*, so errors are not amplified significantly. A linear system with a relatively large (say $\kappa(A) > 10,000$) condition number is *ill-conditioned*, so that small errors can be magnified significantly. An ill-conditioned system *might as well* be underdetermined, even if $M > N$.

Although the first bound is often cited for explaining the significance of $\kappa(A)$, the second bound is actually much tighter [1].

IV. REGULARIZATION

In the real world, condition numbers are large ($\sigma_1 >> \sigma_N$), matrices that are supposed to have null vectors don't ($\sigma_N \neq 0$), etc. What are we to do?

The solution is *regularization*, which means replacing the original problem with a different problem whose solution:

- Roughly matches the desired solution (low bias);
- Is less sensitive to perturbations of, and noise in, the data (low variance);
- Has a parameter that allows bias-variance tradeoff.

We now discuss some regularization techniques:

A. Truncated SVD

Without loss of generality, scale the problem so $\sigma_1 >> 1 >> \sigma_N$. Going right to left, rewrite (6):

$$x = \sum_{i=1}^{N} v_i(u_i^H b/\sigma_i) \qquad (17)$$

which shows that the component of x in the direction v_N is most sensitive to noise in the data, since its magnification factor $1/\sigma_N$ is largest. However, what is magnified is the component of the data in direction u_N, not v_N.

Truncated SVD simply truncates the sum in (17) at $i = (N-K)$, where the smallest K singular values $\sigma_{N-K+1} \ldots \sigma_N < \epsilon$ for some threshold ϵ. Any noise in the data is magnified so much that these terms $v_i(u_i^H b/\sigma_i)$ are meaningless anyways, so we may as well eliminate them and accept the loss.

This seems like a bad idea–Eliminate the most significant terms?! But it works better than it sounds if the noise is white (equally spread out over all u_i). This is because the data b was *constructed* from Ax, so the components of b in the direction u_N is known to be small. So these components are known to have low signal-to-noise ratios.

For example, if A is a lowpass operator, the data b will have only low frequencies in it. So eliminating the high frequencies in b only discards components having low signal-to-noise ratios. The inverse filter magnifies these components in an attempt to restore high frequencies, but magnifies the noise much more. So dump the high frequencies altogether, and restore only the low frequencies. Remember that A is a lowpass operator, not an ideal lowpass filter, so there is work to be done even at the low frequencies!

Truncated SVD has two problems:

- We have to *compute* and *store* the SVD to use it;
- The truncated solution has "ringing," e.g., Gibbs's phenomenon in truncated Fourier series.

The latter can be solved by gradually windowing the small singular value components to zero, rather than sharply cutting them off. But if we are doing that, we don't need to compute the SVD, as we now show.

B. Tikhonov Regularization

This most-commonly-used (I think) regularization technique works as follows. In its simplest form, instead of just minimizing $||Ax - b||$, we minimize the cost functional

$$||Ax - b||^2 + \lambda^2||x||^2 \qquad (18)$$

This can be interpreted in several ways:

- Penalize spikes in x which are likely due to noise;
- Trade off (using λ) mean square error and size x;
- Incorporate a Gaussian prior on x with Gaussian noise $Ax - b$;
- A classical bias-variance tradeoff using λ.

The x minimizing (18) can be derived using derivatives, but a more elegant derivation is as follows. We want the least-squares solution to

$$\begin{bmatrix} A \\ \lambda I \end{bmatrix} x = \begin{bmatrix} b \\ 0 \end{bmatrix} \tag{19}$$

since the mean square error for this problem is (18). The solution is the pseudo-inverse (see above)

$$(A^H A + \lambda^2 I)x = [A^H \quad \lambda I] \begin{bmatrix} b \\ 0 \end{bmatrix} = A^H b \tag{20}$$

so we merely need to add λ^2 to the diagonal of $A^H A$.

C. Analysis of Tikhonov Regularization

This has the following advantages:

- It is simple–no SVD required;
- It preserves the (e.g., sparse) structure of $A^H A$;
- Analysis of its effects is easy.

The effects of Tikhonov regularization are easily seen to be as follows:

- Squared singular values σ_i^2 are now bounded away from zero by λ^2;
- Condition number is reduced from $\frac{\sigma_1}{\sigma_N}$ to $\frac{\sqrt{\sigma_1^2+\lambda^2}}{\sqrt{\sigma_N^2+\lambda^2}}$;
- Singular values become $\sqrt{\sigma_i^2 + \lambda^2}$;
- As $\lambda \to 0$, the effects of regularization disappear;
- The solution (17) is modified to (note right side)

$$x = \sum_{i=1}^{N} v_i \frac{\sigma_i}{\sigma_i^2 + \lambda^2} (u_i^H b) \tag{21}$$

Thus the magnification factor $1/\sigma_i$ is replaced with the gentler magnification factor $\sigma_i/(\sigma_i^2 + \lambda^2)$ which is bounded. For this reason, Tikhonov regularization is also known as damped least squares.

D. Generalized Tikhonov Regularizaion

Tikhonov regularization can be generalized as follows. Suppose we wish not for x to be small, but for the components of x to be smoothly varying. Then penalize not $||x||$ but $||Bx||$ for a suitable matrix B:

- If x=samples of a 1-D signal, Bx takes differences;
- If x=samples of an image, Bx is the Laplacian.

The cost functional is modified to

$$||Ax - b||^2 + \lambda^2 ||Bx||^2; B = \begin{bmatrix} 1 & 0 & \dots \\ -1 & 1 & \dots \\ \ddots & \ddots & \ddots \end{bmatrix} \tag{22}$$

and modifying the above derivation results in

$$(A^H A + \lambda^2 B^H B)x = A^H b \tag{23}$$

which still preserves the structure of $A^H A$.

E. Example of Generalized Tikhonov

To show how well this can work, even on a trivially small example, return to

$$\frac{1}{60} \begin{bmatrix} 60 & 30 & 20 \\ 30 & 20 & 15 \\ 20 & 15 & 12 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 111 \\ 65 \\ 47 \end{bmatrix} \tag{24}$$

This system has the solution

$$[x, y, z] = [69, 24, 90]. \tag{25}$$

Suppose we are interested, for physical reasons, in obtaining a smooth solution. Although we don't "know" this, perturbing 111 to 110 yields the solution

$$[x, y, z] = [60, 60, 60]. \tag{26}$$

We define the matrices A, B, b as, respectively,

$$\frac{1}{60} \begin{bmatrix} 60 & 30 & 20 \\ 30 & 20 & 15 \\ 20 & 15 & 12 \end{bmatrix}; \begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & -1 & 1 \end{bmatrix}; \begin{bmatrix} 111 \\ 65 \\ 47 \end{bmatrix} \tag{27}$$

Then we solve the following system:

$$(A^T A + \lambda^2 B^T B)x = A^T b \tag{28}$$

obtaining these solutions for λ^2=0 and 0.001:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 69.00 \\ 24.00 \\ 90.00 \end{bmatrix} \rightarrow \quad A \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 111.0 \\ 65.0 \\ 47.0 \end{bmatrix}$$
$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 61.24 \\ 61.53 \\ 56.39 \end{bmatrix} \rightarrow \quad A \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 110.8 \\ 65.2 \\ 47.1 \end{bmatrix} \tag{29}$$

Thus we have found a smooth solution that satisfies the system almost exactly. Note that it is not necessarily true that the "exact" solution is $[60, 60, 60]$; there is no exact solution to this problem.

F. Using Basis Functions

Another way of regularizing an inverse problem is to represent the solution in terms of basis functions. This reduces the dimension of the problem, since we only need to solve for the basis function coefficients. It also introduces bias, since the solution cannot be represented exactly using the basis functions.

For example, downsampling an image assumes that the image is in fact bandlimited, and thus was oversampled originally. The bias appears as aliasing (hopefully small) and the regularization appears not only as fewer unknowns to solve for, but better conditioned unknowns as well, since high-frequency components are no longer being sought. This reduces the variance in the problem, at the price of bias.

Still another way of regularizing an inverse problem is discussed in the next section.

V. ITERATIVE ALGORITHMS

Linear systems of equations in image and signal processing are not solved using Gaussian elimination (not in this lifetime, anyways!). Instead, iterative algorithms are used. These have several advantages:

- Since the major computation at each iteration is $A\hat{x}$, the sparseness or structure of A can be exploited to reduce computation and storage;
- A need not be computed or stored; just compute matrix element a_{ij} when needed;
- A rough reconstruction of the image often appears after a few iterations;
- Convergence behavior of the algorithm can sometimes be studied in detail, and modifed (see below);
- Often we don't *want* to solve the problem exactly.

Following are some commonly-used algorithms for solving linear systems of equations:

A. Conjugate Gradient (CG)

This is the most commonly used algorithm, although it requires a Hermitian matrix A. It reduces $||Ax-b||^2$ at each iteration using a gradient/steepest descent approach. It can also be interpreted as successive projection on the Krylov space, spanned by

$$\{b, Ab, A^2b, A^3b \dots A^Nb\} \tag{30}$$

This sounds good until one notices that A^ib converges to the eigenvector associated with the maximum eigenvalue of A, so it is actually quite ill-conditioned. Hence CG is usually viewed as an iterative algorithm, although in theory it terminates after N recursions. BiConjugate Gradient is used to solve $A^HAx = A^Hb$ instead of $Ax = b$.

B. Landweber Iteration

This algorithm is known in many fields under many names (e.g., optics):

$$\begin{aligned} x_{n+1} &= x_n + A^H(b - Ax_n); \quad x_0 = 0 \\ z &= Ax_n; \quad x_{n+1} = x_n + A^Hb - A^Hz \end{aligned} \tag{31}$$

The second form exploits the sparseness and/or structure of A which is usually not present in A^HA; A^Hb is of course only computed once.

The Landweber iteration is very easy to analyze. Initializing with $x_0 = 0$, after n iterations we have

$$\begin{aligned} x_n &= (I - A^HA)^n x_0 + \sum_{i=0}^{n-1} (I - A^HA)^i (A^Hb) \\ &= (A^HA)^{-1}[I - (I - A^HA)^n]A^Hb \end{aligned} \tag{32}$$

where we have set $C = (I - A^HA)$ in the identity

$$I + C + C^2 + \ldots + C^{n-1} = (I - C)^{-1}(I - C^n) \tag{33}$$

Recalling $A^HA = V(S^HS)V^H$ and $I = VV^H$ yields

$$x_n = \sum_{i=1}^{N} (v_i) \frac{1 - (1 - \sigma_i^2)^n}{\sigma_i} (u_i^Hb) \tag{34}$$

C. Stopping the Landweber Iteration

This equation (34) demonstrates the following:

- If $0 < \sigma_i^2 < 2$, then $|1 - \sigma_i^2| < 1$ and $(1 - \sigma_i^2)^n \to 0$;
- Also, the effect of nonzero x_0 will decay to zero, so we may without loss of generality set $x_0 = 0$;
- Also, x_n converges to (17) (it works!);
- Different components converge at different rates;
- After n iterations, we have a damped x_n like (21);
- The damping is different, but the effects similar:
- $\sigma_i/(\sigma_i^2 + \lambda^2)$ vs. $[1 - (1 - \sigma_i^2)^n]/\sigma_i$
- Both approach $1/\sigma_i$ as $\lambda \to 0$ or $n \to \infty$.

This suggests that regularizing the inverse problem can be accomplished by *deliberately stopping* the algorithm after a few iterations, since different components of the solution converge at different rates:

- Components with large σ_i will have converged;
- Components with small σ_i will hardly be present (unscramble the numerator of (34); it is small);
- We have filtered out the noisy components;
- We have recovered a lowpass version of the image;
- We have even saved some time.

This effect was observed in a variety of iterative algorithms in the 1980s. Stopping the algorithm produced better reconstructions, while letting it run made spikes (due to noise) appear in the image.

VI. COMPUTING NULL VECTORS

A. Two Different Problems

Computing the null vector (solution to $An = 0$) of rank-deficient matrix A superficially seems simple. Let $A = USV^H$ with $\sigma_N = 0$. Then $Av_n = 0$.

But this does not work in practice; $\sigma_N \neq 0$, due to roundoff error if nothing else. What does work is to declare that all singular values below some threshold ϵ are in fact zero. If $\sigma_{N-1} >> \sigma_N$ this gives good results (Matlab's `null(A)` does this). Use the singular vector v_N associated with the minimum singular value σ_N, even if $\sigma_N \neq 0$. In fact, this minimizes the normalized cost functional $||Ax||/||x||$.

But if $\sigma_{N-2} >> \sigma_{N-1} \approx \sigma_N$, then we have a problem. Do we use v_N, v_{N-1}, or a linear combination $av_N + bv_{N-1}$ of the two? We call this the *double minimum* problem.

And there is another problem: Are perturbations of A greatly magnified in n? How do we define the conditioning of a matrix whose condition number is infinity? We call this the *condition problem.*

B. Condition Problem

Obviously we cannot use the previous results as is, since $\kappa(A) \to \infty$. Instead, we augment the null vector problem and apply previous results to this.

Let n be the null vector for the rank-deficient $(M \times N)$ matrix A, where $M \geq N$. Then n is orthogonal to the rows of A, and augmenting A with an extra row n^H won't alter the problem, except to replace the rank deficiency. We now have

$$\tilde{A}n = \begin{bmatrix} A \\ n^H \end{bmatrix} n = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{35}$$

where we have normalized n so that $||n|| = 1$.

The SVD of augmented matrix $\tilde{A}$ is

$$\tilde{A} = \begin{bmatrix} U & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_{N-1} & 0 \\ 0 & 0 & 1 \end{bmatrix} V^H \tag{36}$$

since $\tilde{A}^H\tilde{A} = A^HA + nn^H$ (eigenvalue 0 is replaced with 1, since n is orthogonal to other eigenvectors, since it is orthogonal to A) and

$$\tilde{A}\tilde{A}^H = \begin{bmatrix} AA^H & An = 0 \\ n^HA^H = 0 & n^Hn = 1 \end{bmatrix} \tag{37}$$

(an extra eigenvalue 1 with eigenvector $[0 \cdots 0, 1]^T$).

Assuming $\sigma_1 > 1 > \sigma_{N-1}$ (easily arranged by scaling A), we have $\kappa(\tilde{A}) = \sigma_1/\sigma_{N-1}$. Hence the conditioning of the null vector problem (sensitivity of n to variations in the elements of A) is the condition number of matrix A, *excluding the zero singular value.*

C. Regularization of Condition Problem

This makes sense (what else could it be?), but regularization is a problem. The obvious strategy of applying Tikhonov regularization to $\tilde{A}$ yields

$$(\tilde{A}^H\tilde{A} + \lambda^2 I)x = \tilde{A}^H[0 \ldots 0, 1]^H \tag{38}$$

which becomes

$$(A^HA + nn^H + \lambda^2 I)x = n. \tag{39}$$

But this has the solution $x = n/(\lambda^2+1)$, so Tikhonov regularization only scales the solution n by $(\lambda^2 + 1)$.

This could have been anticipated. $(A^HA + \lambda^2 I)$ has a smaller ratio of non-zero singular values than (A^HA), but the same minimum singular vector. We must use $B \neq I$ in the above formulation to obtain a non-trivial answer.

D. Double Minimum Problem

If $\sigma_{N-2} >> \sigma_{N-1} \approx \sigma_N$, then any linear combination $x = av_N + bv_{N-1}$ will yield a small $||Ax|/||x||$. The null space of A has dimension=2; this means that there is no unique null vector.

Using (32) for the Landweber iteration with $b = 0$ and $x_0 \neq 0$ yields

$$x_n = (I - A^HA)^n x_0 = \sum_{i=1}^{N} v_i(1 - \sigma_i^2)^n (v_i^H x_0) \tag{40}$$

This shows that all components other than $\sigma_N = 0$ eventually decay to zero.

The *inverse power method* iterates

$$\begin{gathered} x_{n+1} = (A^HA)^{-1}x_n/||x_n|| \to \\ x_{n+1} = z/||x_n||; \quad (A^HA)z = x_n \end{gathered} \tag{41}$$

since this converges to the largest eigenvalue $1/\sigma_N^2$ of $(A^HA)^{-1}$ and its associated eigenvector v_N. Note that convergence is *very* fast. The division by $||x_n||$ is just a scale factor to prevent divergence.

As either of these iterations proceed, all components but v_N decay to zero. However, if the iteration is *stopped*, then $x_n = av_N + bv_{N-1}$ for some constants a and b, since all other components $v_{N-2}, v_{N-3} \ldots$ have already decayed to zero. That is, running the algorithm to convergence is making the choice $a = 1$ and $b = 0$, when including v_{N-1} might be better.

The effects of all of this can be summarized as:

- Running the iteration to convergence leaves v_N;
- Stopping the iteration leaves $x_n = av_N + bv_{N-1}$;
- Unless the actual answer is in fact v_N, stopping the iteration will often give a better answer, *even if the ratio a/b is wrong*;
- The overall scale factor is irrelevant, so there is one unknown a/b.

Note that even if we compute both v_N and v_{N-1}, the appropriate relative weighting a/b of the two is not evident unless *a priori* information about n is available. So we might as well use the easy choice dictated by stopping the inverse power method.

E. Example: Inverse Power Method

Suppose the desired signal is known to be a null vector of the following Toeplitz system:

$$\begin{bmatrix} 200.09 & -0.01 & -199.9 & -0.01 \\ -0.01 & 200.09 & -0.01 & -199.9 \\ -199.9 & -0.01 & 200.09 & -0.01 \\ -0.01 & -199.9 & -0.01 & 200.09 \end{bmatrix} \begin{bmatrix} w \\ x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{42}$$

Attempting to find this signal, we run the inverse power method, initialized with

$$[w, x, y, z] = [1, 2, 3, 4] \tag{43}$$

Normalizing by dividing by w at each iteration, we get after one iteration

$$[w, x, y, z] = [1, 1.38, 1, 1.38] \tag{44}$$

But as the iteration proceeds, the signal flattens out. After ten iteration, we get

$$[w, x, y, z] = [1, 1.04, 1, 1.04] \tag{45}$$

The actual signal is

$$[w, x, y, z] = [2, 3, 2, 3] \tag{46}$$

so the iteration was closest at the first iteration, and then moved away from it! What's going on?

The answer is found by examining the SVD of the above matrix, which was constructed using

$$\begin{aligned} U = V = & \begin{bmatrix} 1 & 1 & 1 & 1 \\ -1 & 1 & -1 & 1 \\ -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \\ S = & \; diag[100, 100, .05, .04] \end{aligned} \tag{47}$$

Since the matrix is Hermitian, we have that $U = V$ and $\sigma_i = |\lambda_i|$, i.e., the SVD is essentially a matrix eigenvalue diagonalization.

The two small singular values .05 and .04 mean that the matrix has a nullspace of dimension=2. The actual signal is a linear combination of v_3 and v_4:

$$\begin{bmatrix} 2 \\ 3 \\ 2 \\ 3 \end{bmatrix} = -0.5 \begin{bmatrix} 1 \\ -1 \\ 1 \\ -1 \end{bmatrix} + 2.5 \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \tag{48}$$

After one iteration, the inverse power method had a linear combination of v_3 and v_4, even though it was the wrong one. Meanwhile, v_1 and v_2 were already eliminated. After ten iterations, only v_4 was left. Even an incorrect linear combination of v_3 and v_4 is better than just v_4, in this case.

VII. CONCLUSION

We have attempted to collect the basics of matrix conditioning, sensitivity, roundoff error, and iterative computation together here. The connections between these various topics should now be evident.

References

[1] G.W. Stewart, *Introduction to Matrix Computations*, 1973.