Announcements

1) Math Club talk on SVD, Thursday, 4:00, CB 2062

2) Change in coding question on HW #5 - "one" for loop changed to "two" (problem 4)

3) Scholarships! Apply online through university's system.

If you are a masters student in applied math, apply even if you aren't full-time. These are Math department scholarships, deadline Monday April 7th.

Perturbing A

2×1 case: why is this so much more annoying than perturbing b?

Given $A = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix}$,

perturb to

$$\tilde{A} = \begin{bmatrix} a_1 + \delta a_1 \\ a_2 + \delta a_2 \end{bmatrix}.$$

When solving least squares using the normal equations, we have to calculate the pseudoinverse of A or $\tilde{A}$.

We start with

$$A^*A = \begin{bmatrix} \bar{a}_1 & \bar{a}_2 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix}$$

$$= |a_1|^2 + |a_2|^2$$

$$= \left\| \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} \right\|_2^2$$

(as a vector in $\mathbb{C}^2$)

We then compute

$$A^+ = (A^*A)^{-1} A^*$$

$$= \frac{1}{|a_1|^2+|a_2|^2} A^*$$

$$= \begin{bmatrix} \frac{\bar{a}_1}{|a_1|^2+|a_2|^2} & \frac{\bar{a}_2}{|a_1|^2+|a_2|^2} \end{bmatrix}$$

Writing $\tilde{A} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$, we get

$$(\tilde{A})^{+} = \begin{bmatrix} \dfrac{\overline{b_1}}{|b_1|^2+|b_2|^2} & \dfrac{\overline{b_2}}{|b_1|^2+|b_2|^2} \end{bmatrix}$$

where $b_1 = a_1 + \delta a_1$,

$b_2 = a_2 + \delta a_2$.

We see

$$|b_1|^2 + |b_2|^2$$

$$= (a_1 + \delta a_1)(\bar{a}_1 + \overline{\delta a_1}) + (a_2 + \delta a_2)(\bar{a}_2 + \overline{\delta a_2})$$

$$= |a_1|^2 + |a_2|^2 + 2\,\mathrm{Re}(a_1 \overline{\delta a_1}) + |\delta a_1|^2 + |\delta a_2|^2$$

quadratic terms

To calculate $(\tilde{A})^+$, we have to divide $(\tilde{A})^*$ by this quantity, which introduces quadratic errors instead of linear ones.

Q: How do we keep track of these errors?

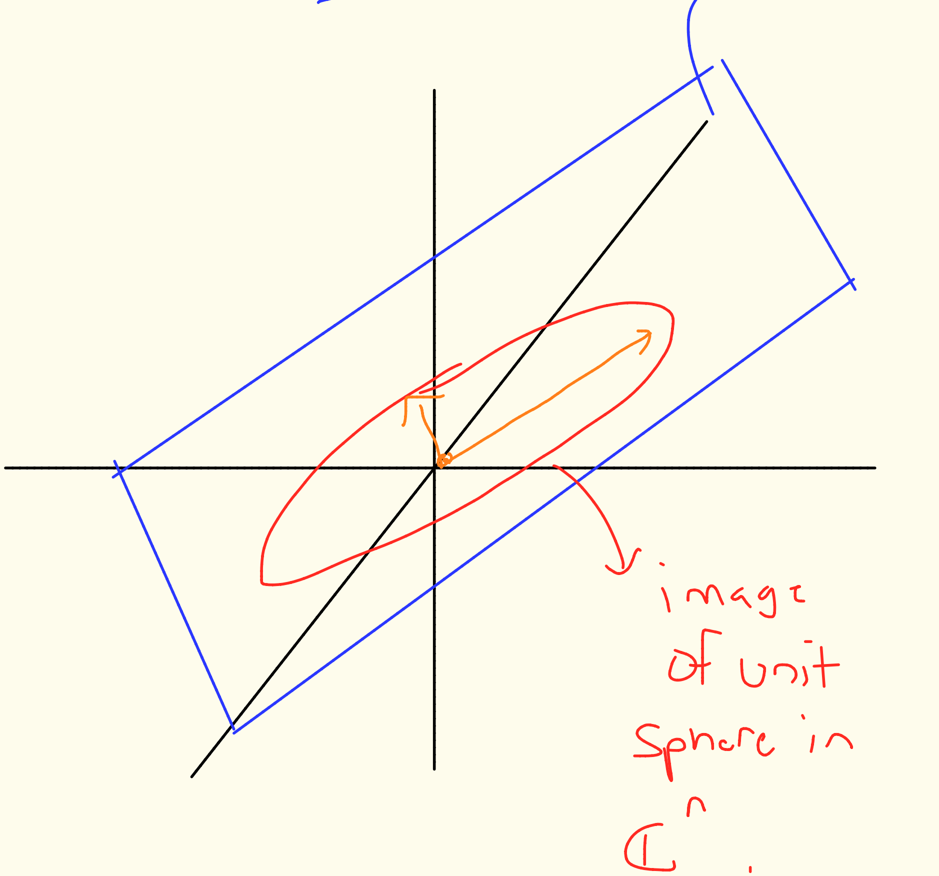
Perturbing A

Think of what perturbing A does to the range:

recall that the image of the unit sphere in $\mathbb{C}^n$ is a "hyperellipse" in ran(A).

Solving for $y = Ax = AA^{+}b$.

Picture



Perturb the range by a vector orthogonal to Ran(A). The directions are determined by the "semi-axes" of the image of the unit sphere in $\mathbb{C}^n$, which are exactly the right singular vectors of A.

The smallest singular value gives you a vector with "maximum tilt", then all you care about is the angle you tilt by.

You bound the angle to bound the condition number.

Stability for Least Squares

Pick your algorithm!

But first recall

Theorem: Let $f: X \to Y$ be a problem and let $\tilde{f}$ be a backwards stable algorithm $\tilde{f}: X \to Y$ for f and some $\varepsilon_{machine}$. Then for all $x \in X$,

$$\frac{\|f(x) - \tilde{f}(x)\|}{\|f(x)\|} = O(\kappa(x)\, \varepsilon_{machine})$$

So you should expect to lose as many digits as the condition number.

Using the example in lecture 19 in the text, we get the following condition numbers:

	y	x
b	1	1.1×10^5
A	2.3×10^{10}	3.2×10^{10}

Best: QR Factorization via Householder

Algorithm: (11.2)

1) Compute the reduced QR decomposition of A

2) Compute $\hat{Q}^* b$

3) Solve $\hat{R}x = \hat{Q}^* b$ for x via back-substitution.

Next best: SVD

Algorithm: (11.3)

1) Compute reduced svd of A.

algorithm?

2) Compute $\hat{U}^* b$

3) Solve $\hat{\Sigma} w = \hat{U}^* b$ via back-substitution

4) Set $x = V w$.

Worst: QR Factorization via Gram-Schmidt

Algorithm: (11, 2)

Same as before, except use Gram-Schmidt instead of Householder to get the reduced QR decomposition.