Computational Statistics

Sep 29, 2021

Recap: ① Optimization → Newton, exact Hessian
→ Quasi-Newton, approximate Hessian
↳ BFGS rank-2 updats to approx Hessian

② Num. Lin. Alg.
- Operation counts
- Condition number
(sens. of output to input)
- Vector / matrix norms.

Today Matrix factorizations, dimension reduction

$A = LU$ (or LL^T) → solving linear systems
→ generating random variables (later)

$= QR$ → least squares problems

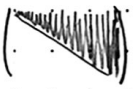
$= USV^T$ → matrix approximation, dimension reduction

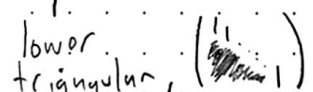
Generally, when done directly all matrix factorizations of an $N \times N$ matrix require $\mathcal{O}(N^3)$ operations.
↑ "big oh" notation

Heuristically, each element of the matrix is "touched" N times (e.g. interacts with all columns or rows)
↑ N^2 of them

The LU Factorization

This factorization is equivalent to Gaussian elimination.

Writing $A = LU$ ← upper triangular 

↑ lower triangular, 
ones on diagonal

} Finding L, U costs $O(N^3)$ ops

Then to solve $A\vec{x} = \vec{b}$, instead first solve

$LU\vec{x} = \vec{b}$

$L\vec{y} = \vec{b}$ } Forward substitution, $O(N^2)$

then $U\vec{x} = \vec{y}$ } Backward substitution, $O(N^2)$

Failure mode of LU factorization is the same as for Gaussian elimination – a zero in the pivot spot.

To avoid this, permute rows this is called partial pivoting, and is numerically stable. => $PA = LU$ (mention full pivoting)

↑ permutation matrix.

Ex $A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$ / $\begin{pmatrix} \epsilon & 1 \\ 1 & 1 \end{pmatrix}\vec{x} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$

$$\begin{pmatrix} 1 & 0 \\ -\frac{c}{a} & 1 \end{pmatrix} A = \begin{pmatrix} a & b \\ 0 & d - bc/a \end{pmatrix}$$

$$\Rightarrow A = \begin{pmatrix} 1 & 0 \\ c/a & 1 \end{pmatrix}\begin{pmatrix} a & b \\ 0 & d - bc/a \end{pmatrix}$$

L U

Cholesky Factorization

- special case when A is symmetric positive definite
- $A = LL^T$ ↑ not 1's on diagonal
- pivoting never necessary

Furthermore, $\det A = \det LU = \det L \cdot \det U$

$= 1 \cdot \det U$

$= \prod_{i=1}^{N} u_{ii}$ ← product of diagonals

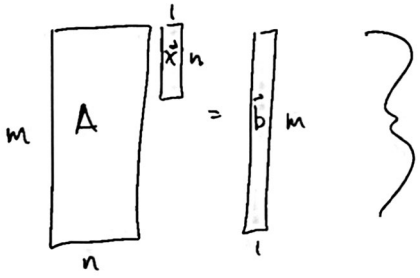
Least Squares

Two canonical problems in linear algebra:

① Solve $A\vec{x} = \vec{b}$ $\Rightarrow$ A is a square $n \times n$ matrix.

② Find "the best" solution to a system $A\vec{x} = \vec{b}$ where A is an $m \times n$ matrix.

Ex:



} In general, not solvable because $m > n$.

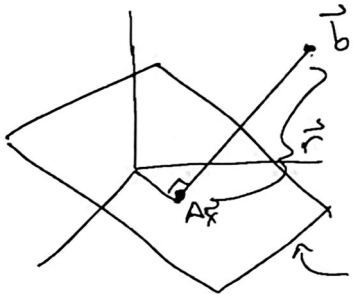
One version of "the best" solution is the least squares solution:

Least squares: For $A \in \mathbb{R}^{m \times n}$, with $m > n$, find $\vec{x}$ such that $\| A\vec{x} - \vec{b} \|_2$ is as small as possible.

↑ The fact that this is the 2-norm is important

Ex: $A \in \mathbb{R}^{3 \times 2}$

Geometrically



$\min \| A\vec{x} - \vec{b} \|_2$

minimize the distance between $\vec{b}$ and $A\vec{x}$.

We also have that

$$\| A\vec{x} - \vec{b} \|_2^2 = \sum_{i=1}^{m} \left(\sum_{j=1}^{n} a_{ij} x_j - b_i \right)^2 = f(\vec{x})$$

If all you knew was calculus, then you could solve this by find the solution to

$$\begin{matrix} \frac{\partial f}{\partial x_1} = 0 \\ \vdots \\ \frac{\partial f}{\partial x_n} = 0 \end{matrix}$$

} Finding a zero of f' $\Rightarrow$ minimization or maximization

Let $\vec{r} = \vec{b} - A\vec{x}$ = residual vector.

$\Rightarrow \|\vec{r}\|$ = distance from col(A) to $\vec{b}$.

If $\vec{x}$ is the least squares solution, then $\vec{r} \perp$ col A (perpendicular.),

in particular, $A^T\vec{r} = \vec{0}$.

$$\Rightarrow A^T\vec{r} = \vec{0} = A^T(\vec{b} - A\vec{x})$$
$$= A^T\vec{b} - A^TA\vec{x}$$

$\Rightarrow$ The least squares solution $\vec{x}$ solves $\underbrace{A^TA\vec{x} = A^T\vec{b}}$.

"normal equations"

Then equations have at least one solution.

From a numerical point of view, how best to find $\vec{x}$ to min $\|A\vec{x} - \vec{b}\|_2$?

Options: ① Solve the normal equations:

$$A^TA\vec{x} = A^T\vec{b}$$

A^TA: mxn

Drawback: Solving $A^TA\vec{x} = A^T\vec{b}$ has a condition number that is the square of $A\vec{x} = \vec{b}$.

Ex: If A is mxn with rank n, then

$$A = U_{mxn}\, S_{nxn}\, V^T_{nxn} \Rightarrow \text{cond}(A) = \frac{\sigma_1}{\sigma_n}$$

Give quick proof

$$A^TA = V S^2 V^T \Rightarrow \text{cond}(A^TA) = \frac{\sigma_1^2}{\sigma_n^2}$$

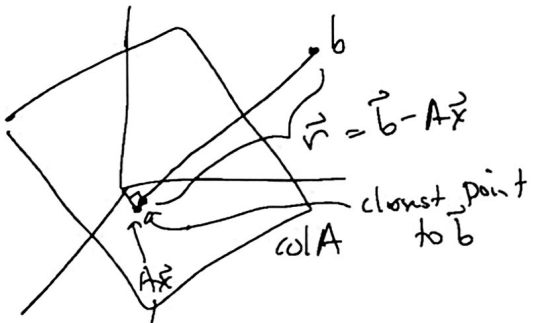
Return to SVD later

So if cond$(A) = 10^4$, then cond$(A^TA) = 10^8$.

Option ② Use calculus to min $\|A\vec{x} - \vec{b}\|$

Option ③ : Reformulate the problem to be consistent.

Goal: $\min \|A\vec{x} - \vec{b}\|$ without solving the normal equations.



Idea: Construct a linear system that is consistent (i.e. has a solution), without computing A^TA.

Instead of solving $A\vec{x} = \vec{b}$, (which has no solution in general),

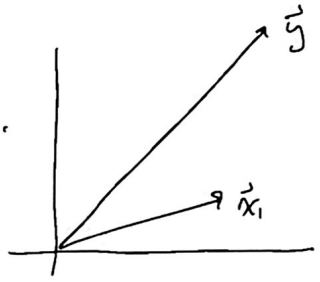
solve $A\vec{x} = \vec{b}'$ ← $\vec{b}'$ = orthogonal projection of $\vec{b}$ onto the column space of A.

$$A\vec{x} = \text{proj}_{\text{colA}}\vec{b} = \vec{b}'$$

How do we compute $\text{proj}_{\text{colA}}\vec{b}$?

How do we compute a projection of one vector onto another?

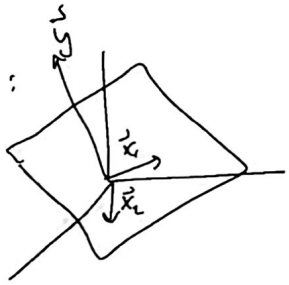
In 2D.



The orthogonal projection of $\vec{y}$ onto $\vec{x}_1$ is the component of $\vec{y}$ pointing in the $\vec{x}_1$ direction.

$$\text{proj}_{\vec{x}_1}\vec{y} = \underbrace{\left(\vec{y}, \frac{\vec{x}_1}{\|\vec{x}_1\|}\right)}_{\text{inner product}} \left(\frac{\vec{x}_1}{\|\vec{x}_1\|}\right) \longrightarrow \text{unit vector pointing in the direction of } \vec{x}_1.$$

In 3D:



$\text{proj}_{\mathcal{X}}\vec{y}$ = the component of $\vec{y}$ in the $\vec{x}_1$-$\vec{x}_2$ plane.

$$= \underbrace{(\vec{y}, \hat{x}_1)\,\vec{x}_1}_{\hat{x}_1 = \frac{\vec{x}_1}{\|\vec{x}_1\|}} + \underbrace{\left(\vec{y} - (\vec{y}, \hat{x}_1)\,\hat{x}_1\right)}_{\text{the remaining part of } \vec{y} \text{ after projecting onto } \vec{x}}, \hat{x}_2 \Big)\,\hat{x}_2$$

If $(\hat{x}_1, \hat{x}_2) = 0$, (orthogonal), then

$$\text{proj}_{\mathcal{X}}\vec{y} = (\vec{y}, \hat{x}_1)\,\hat{x}_1 + (\vec{y}, \hat{x}_2)\,\hat{x}_2$$

Gram Schmidt process.

So this suggests that we want to find an <u>orthonormal</u> basis for col(A), $U = \text{span}\{\hat{u}_1, \dots, \hat{u}_n\}$ and then project $\vec{b}$ onto U, forming $\vec{b}'$.

$$\vec{b}' = (\vec{b}, \hat{u}_1)\hat{u}_1 + (\vec{b}, \hat{u}_2)\hat{u}_2 + \dots + (\vec{b}, \hat{u}_n)\hat{u}_n$$

$$= \begin{pmatrix} \hat{u}_1 & \hat{u}_2 & \cdots & \hat{u}_n \end{pmatrix} \begin{pmatrix} (\vec{b}, \hat{u}_1) \\ (\vec{b}, \hat{u}_2) \\ \vdots \\ (\vec{b}, \hat{u}_n) \end{pmatrix}$$

$$= \begin{pmatrix} \hat{u}_1 & \hat{u}_2 & \cdots & \hat{u}_n \end{pmatrix} \begin{pmatrix} \hat{u}_1^T \\ \hat{u}_2^T \\ \vdots \\ \hat{u}_n^T \end{pmatrix} \vec{b}$$

$$= \underbrace{U U^T \vec{b}}$$

orthogonal projection of $\vec{b}$ onto the columnspace of A

$\Rightarrow$ The linear system $A\vec{x} = UU^T\vec{b}$ is <u>consistent</u>.

The columns of U are orthonormal, so U is an <u>orthogonal</u> matrix $\Rightarrow$ $U^TU = I$.

Use the Gram-Schmidt process to construct U.

(Mention numerical stability.)

G-S Process:

Goal: Given some set of vectors $\vec{a}_1 \dots \vec{a}_n$, form another sequence of vectors $\hat{q}_1, \hat{q}_2 \dots \hat{q}_n$ that are orthonormal.

Let: $\hat{q}_1 = \vec{a}_1 / \|\vec{a}_1\| \Rightarrow \|\hat{q}_1\| = 1$ All norms are l_2 norms.

$$\vec{q}_2' = \vec{a}_2 - (\vec{a}_2, \hat{q}_1)\hat{q}_1$$

inner product $(\vec{x}, \vec{y}) = \vec{x}^T \vec{y}$.

$\vec{q}_2'$ is not normalized

$\hat{}$ means unit vector.

$$\hat{q}_2 = \vec{q}_2' / \|\vec{q}_2'\|$$

Proceed ...

$$\vec{q}_3' = \vec{a}_3 - (\vec{a}_3, \hat{q}_2)\hat{q}_2 - (\vec{a}_3, \hat{q}_1)\hat{q}_1$$

$$\hat{q}_3 = \vec{q}_3' / \|\vec{q}_3'\|$$

So when we are done, by construction,

$Q = (\hat{q}_1\ \hat{q}_2 \cdots \hat{q}_n)$ then $Q^T Q = I$

And the projection of $\vec{x}$ onto the column space of A, i.e. $\text{Span}\{\vec{a}_1, \dots, \vec{a}_n\}$ is just $\text{proj}_A \vec{x} = Q Q^T \vec{x}$

Furthermore, this automatically yields a factorization of the matrix A:

$\text{col}\, Q = \text{col}\, A$

So any $\hat{q}_j = c_1 \vec{a}_1 + \dots + c_n \vec{a}_n$

and likewise $\vec{a}_j = d_1 \hat{q}_1 + \dots + d_n \hat{q}_n$

This gives us the QR factorization:

$$A = QR$$

$$(\vec{a}_1 \cdots \vec{a}_n) = (\hat{q}_1\ \hat{q}_2 \cdots \hat{q}_n) \begin{pmatrix} r_{11} & r_{12} & r_{13} & \cdots & r_{1n} \\ 0 & r_{22} & & & \\ 0 & & \ddots & & \\ \vdots & & & & r_{nn} \end{pmatrix}$$

The question is: What are the r_{ij}'s?

r_{ij}'s can be obtained from G-S process.

Ex: From G-S:

$$\hat{q}_1 = \vec{a}_1 / \|\vec{a}_1\| \quad \Rightarrow \quad \vec{a}_1 = \|\vec{a}_1\|\,\hat{q}_1 \quad \Rightarrow \quad r_{11} = \|\vec{a}_1\|$$

$$\vec{q}_2' = \vec{a}_2 - (\vec{a}_2, \hat{q}_1)\,\hat{q}_1$$

$$\Rightarrow \quad \vec{a}_2 = (\vec{a}_2, \hat{q}_1)\,\hat{q}_1 + \vec{q}_2'$$

$$= r_{12}\,\hat{q}_1 + r_{22}\,\hat{q}_2$$

$$\vec{q}_2' = \|\vec{a}_2 - (\vec{a}_2, \hat{q}_1)\,\hat{q}_1\| \cdot \hat{q}_2$$

With sufficient bookkeeping, the G-S process yields the factorization $A = QR$.

So now: to solve least squares:

① $A\vec{x} = Q\,Q^T\vec{b}$

Alternatively: ② Given that $A = QR$

Solve $R\vec{x} = Q^T\vec{b}$

↑ upper triangular system

① Preferred method

Show later on HW or exam...

What is the operation count?

Applications: Linear Regression

Given data: (x_i^1, x_i^2, y_i) for $i = 1 \dots n$

Generative model: $y = a + bx_1 + cx_2 + \epsilon$

$\epsilon \sim N(0, \sigma^2)$ error or measurement noise.

You observe $y_i = a + bx_i^1 + cx_i^2 + \epsilon_i$

y_i: dependent variable; x_i^1, x_i^2: independent variables

Assumption is that noise only appears in y_i.

Given an estimate of a, b, c: $\hat{a}, \hat{b}, \hat{c}$, then the residuals are given as

$$r_i = y_i - \underbrace{\left(\hat{a} + \hat{b}x_i^1 + \hat{c}x_i^2\right)}_{\text{predicted value, given estimate } \hat{a}, \hat{b}, \hat{c}}$$

y_i: observed value

Question: How do we estimate a, b, c?

One option: minimize the squared residuals:

$$\min_{a,b,c} \|\vec{r}\|_2^2 = \min_{a,b,c} \sum_{i=1}^{n} \left(y_i - a - bx_i^1 - cx_i^2\right)^2$$

This is a least squares problem.

Form the least square system:

design matrix

$$\min_{a,b,c} \left\| \underbrace{\begin{pmatrix} 1 & x_1^1 & x_1^2 \\ 1 & x_2^1 & x_2^2 \\ \vdots & \vdots & \vdots \\ 1 & x_n^1 & x_n^2 \end{pmatrix}}_{X} \underbrace{\begin{pmatrix} a \\ b \\ c \end{pmatrix}}_{\vec{a}} - \underbrace{\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}}_{\vec{y}} \right\| = \min_{\vec{a}} \|X\vec{a} - \vec{y}\|$$