LECTURE 3 AND 4: ALGORITHMS FOR LINEAR ALGEBRA

STAT 545: INTRODUCTION TO COMPUTATIONAL STATISTICS

Vinayak Rao

Department of Statistics, Purdue University

August 26, 2019

Solving a system of linear equations

Preliminaries: cost of standard matrix algebra operations

Solving a system of linear equations

Preliminaries: cost of standard matrix algebra operations

Consider $AX = b$, where A is $N \times N$, and X and b are $N \times k$.

$$\text{Solve for } X: \quad X = A^{-1}b$$

Solving a system of linear equations

Preliminaries: cost of standard matrix algebra operations

Consider $AX = b$, where A is $N \times N$, and X and b are $N \times k$.

$$\text{Solve for } X: \quad X = A^{-1}b$$

Calculate the inverse of A and multiply? No!

Solving a system of linear equations

Preliminaries: cost of standard matrix algebra operations

Consider $AX = b$, where A is $N \times N$, and X and b are $N \times k$.

$$\text{Solve for } X: \quad X = A^{-1}b$$

Calculate the inverse of A and multiply? No!

- Directly solving for X is faster, and more stable numerically
- A^{-1} need not even exist

```
> solve(A,b)        # Directly solve for b
> solve(A) %*% b    # Return inverse and multiply
```

http://www.johndcook.com/blog/2010/01/19/dont-invert-that-matrix/

Gauss-Jordan elimination

$$A \cdot X = b$$

$$A \cdot \left[\, X \,,\ A^{-1} \,\right] = [\, b \,,\ I \,]$$

Gauss-Jordan elimination

$$A \cdot X = b$$

$$A \cdot \left[\, X \,,\; A^{-1} \,\right] = [\, b \,,\; I \,]$$

$$\begin{bmatrix} 1 & 0 & 1 \\ 2 & 0 & 1 \\ 1 & -2 & -1 \end{bmatrix} \begin{bmatrix} x_1 & v_{11} & v_{12} & v_{13} \\ x_2 & v_{21} & v_{22} & v_{23} \\ x_3 & v_{31} & v_{32} & v_{33} \end{bmatrix} = \begin{bmatrix} 4 & 1 & 0 & 0 \\ 10 & 0 & 1 & 0 \\ 3 & 0 & 0 & 1 \end{bmatrix}$$

$$A \cdot X = b$$

$$A \cdot \left[\, X \,,\; A^{-1} \,\right] = [\, b \,,\; I \,]$$

$$\begin{bmatrix} 1 & 0 & 1 \\ 2 & 0 & 1 \\ 1 & -2 & -1 \end{bmatrix} \begin{bmatrix} x_1 & v_{11} & v_{12} & v_{13} \\ x_2 & v_{21} & v_{22} & v_{23} \\ x_3 & v_{31} & v_{32} & v_{33} \end{bmatrix} = \begin{bmatrix} 4 & 1 & 0 & 0 \\ 10 & 0 & 1 & 0 \\ 3 & 0 & 0 & 1 \end{bmatrix}$$

Manipulate to get: $\quad I \cdot \left[\, X \,,\; A^{-1} \,\right] = [\, \hat{c}_1 \,,\; \hat{C}_2 \,]$

$$A \cdot X = b$$

$$A \cdot \left[\, X\ ,\ A^{-1} \,\right] = [\, b\ ,\ I \,]$$

$$\begin{bmatrix} 1 & 0 & 1 \\ 2 & 0 & 1 \\ 1 & -2 & -1 \end{bmatrix} \begin{bmatrix} x_1 & v_{11} & v_{12} & v_{13} \\ x_2 & v_{21} & v_{22} & v_{23} \\ x_3 & v_{31} & v_{32} & v_{33} \end{bmatrix} = \begin{bmatrix} 4 & 1 & 0 & 0 \\ 10 & 0 & 1 & 0 \\ 3 & 0 & 0 & 1 \end{bmatrix}$$

Manipulate to get: $I \cdot \left[\, X\ ,\ A^{-1} \,\right] = [\, \hat{c}_1\ ,\ \hat{C}_2 \,]$

At step i:

- Make element $a_{ii} = 1$ (by scaling or pivoting)
- Set other elements in column i to 0 by multiplying and subtracting that row

$$\begin{bmatrix} 1 & 0 & 1 \\ 2 & 0 & 1 \\ 1 & -2 & -1 \end{bmatrix} \begin{bmatrix} x_1 & v_{11} & v_{12} & v_{13} \\ x_2 & v_{21} & v_{22} & v_{23} \\ x_3 & v_{31} & v_{32} & v_{33} \end{bmatrix} = \begin{bmatrix} 4 & 1 & 0 & 0 \\ 10 & 0 & 1 & 0 \\ 3 & 0 & 0 & 1 \end{bmatrix}$$

Gauss-Jordan elimination

$$\begin{bmatrix} 1 & 0 & 1 \\ 2 & 0 & 1 \\ 1 & -2 & -1 \end{bmatrix} \begin{bmatrix} x_1 & v_{11} & v_{12} & v_{13} \\ x_2 & v_{21} & v_{22} & v_{23} \\ x_3 & v_{31} & v_{32} & v_{33} \end{bmatrix} = \begin{bmatrix} 4 & 1 & 0 & 0 \\ 10 & 0 & 1 & 0 \\ 3 & 0 & 0 & 1 \end{bmatrix}$$

Multiply row 1 by 2 and subtract

Gauss-Jordan elimination

$$\begin{bmatrix} 1 & 0 & 1 \\ 0 & 0 & -1 \\ 1 & -2 & -1 \end{bmatrix} \begin{bmatrix} x_1 & v_{11} & v_{12} & v_{13} \\ x_2 & v_{21} & v_{22} & v_{23} \\ x_3 & v_{31} & v_{32} & v_{33} \end{bmatrix} = \begin{bmatrix} 4 & 1 & 0 & 0 \\ 2 & -2 & 1 & 0 \\ 3 & 0 & 0 & 1 \end{bmatrix}$$

Gauss-Jordan elimination

$$\begin{bmatrix} 1 & 0 & 1 \\ 0 & 0 & -1 \\ 0 & -2 & -2 \end{bmatrix} \begin{bmatrix} x_1 & v_{11} & v_{12} & v_{13} \\ x_2 & v_{21} & v_{22} & v_{23} \\ x_3 & v_{31} & v_{32} & v_{33} \end{bmatrix} = \begin{bmatrix} 4 & 1 & 0 & 0 \\ 2 & -2 & 1 & 0 \\ -1 & -1 & 0 & 1 \end{bmatrix}$$

Subtract row 1

$$\begin{bmatrix} 1 & 0 & 1 \\ 0 & -2 & -2 \\ 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} x_1 & v_{11} & v_{12} & v_{13} \\ x_2 & v_{21} & v_{22} & v_{23} \\ x_3 & v_{31} & v_{32} & v_{33} \end{bmatrix} = \begin{bmatrix} 4 & 1 & 0 & 0 \\ -1 & -1 & 0 & 1 \\ 2 & -2 & 1 & 0 \end{bmatrix}$$

Pivot

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 & v_{11} & v_{12} & v_{13} \\ x_2 & v_{21} & v_{22} & v_{23} \\ x_3 & v_{31} & v_{32} & v_{33} \end{bmatrix} = \begin{bmatrix} 6 & -1 & 1 & 0 \\ 2.5 & -1.5 & 1 & 0.5 \\ -2 & 2 & -1 & 0 \end{bmatrix}$$

Continue till we get an identity matrix

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 & v_{11} & v_{12} & v_{13} \\ x_2 & v_{21} & v_{22} & v_{23} \\ x_3 & v_{31} & v_{32} & v_{33} \end{bmatrix} = \begin{bmatrix} 6 & -1 & 1 & 0 \\ 2.5 & -1.5 & 1 & 0.5 \\ -2 & 2 & -1 & 0 \end{bmatrix}$$

What is the cost of this algorithm?

Gauss elimination with back-substitution

$$A \cdot [\, X \,,\ A^{-1} \,] = [\, b \,,\ I \,]$$

Gauss elimination with back-substitution

$$A \cdot [\, X \, , \, A^{-1} \,] = [\, b \, , \, I \,]$$

$O(N^3)$ manipulation to get:

$$U \cdot [\, X \, , \, A^{-1} \,] = [\, \hat{c}_1 \, , \, \hat{C}_2 \,]$$

Here, U is an upper-triangular matrix.

Gauss elimination with back-substitution

$$A \cdot [\, X \,,\; A^{-1} \,] = [\, b \,,\; I \,]$$

$O(N^3)$ manipulation to get:

$$U \cdot [\, X \,,\; A^{-1} \,] = [\, \hat{c}_1 \,,\; \hat{C}_2 \,]$$

Here, U is an upper-triangular matrix.

Cannot just read off solution. Need to backsolve.

What are we actually doing?

$$A = LU$$

Here L and U are lower and upper triangular matrices.

$$L = \begin{bmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{bmatrix}, U = \begin{bmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{bmatrix}$$

LU decomposition

What are we actually doing?

$$A = LU$$

Here L and U are lower and upper triangular matrices.

$$L = \begin{bmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{bmatrix}, U = \begin{bmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{bmatrix}$$

Is this always possible?

$$A = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

LU DECOMPOSITION

What are we actually doing?

$$A = LU$$

Here L and U are lower and upper triangular matrices.

$$L = \begin{bmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{bmatrix}, U = \begin{bmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{bmatrix}$$

Is this always possible?

$$A = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

$$PA = LU, \quad P \text{ is a permutation matrix}$$

Crout's algorithm, $O(N^3)$, stable, L, U can be computed in place.

Backsubstitution

$$AX = b$$

Backsubstitution

$$AX = b$$

$$LUX = Pb$$

First solve Y by forward substitution

$$LY = Pb$$

Backsubstitution

$$AX = b$$

$$LUX = Pb$$

First solve Y by forward substitution

$$LY = Pb$$

$$\begin{bmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} \hat{b}_1 \\ \hat{b}_2 \\ \hat{b}_3 \end{bmatrix}$$

Backsubstitution

$$AX = b$$

$$LUX = Pb$$

First solve Y by forward substitution

$$LY = Pb$$

$$\begin{bmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} \hat{b}_1 \\ \hat{b}_2 \\ \hat{b}_3 \end{bmatrix}$$

Then solve X by back substitution

$$UX = Y$$

Comments

- LU-decomposition can be reused for different b's.
- Calculating LU decomposition: $O(N^3)$.
- Given LU decomposition, solving for X: $O(N^2)$.
- $|A| = |P^{-1}LU| = (-1)^S \prod_{i=1}^{N} u_{ii}$ (S: num. of exchanges)
- $LUA^{-1} = PI$, can solve for A^{-1}. (back to Gauss-Jordan)

Cholesky decomposition

If A is symmetric positive-definite:

$$A = LL^T \quad \text{(but now } L \text{ need not have a diagonal of ones)}$$

- 'Square-root' of A
- More stable.
- Twice as efficient.
- Related: $A = LDL^T$ (but now L has a unit diagonal).

Eigenvalue decomposition

An $N \times N$ matrix A: a map from $\mathbb{R}^N \to \mathbb{R}^N$.

An eigenvector v undergoes no rotation:

$$Av = \lambda v$$

λ is the corresponding eigenvalue, and gives the 'rescaling'.

Eigenvalue decomposition

An $N \times N$ matrix A: a map from $\mathbb{R}^N \to \mathbb{R}^N$.
An eigenvector v undergoes no rotation:

$$Av = \lambda v$$

λ is the corresponding eigenvalue, and gives the 'rescaling'.

Let $\Lambda = \mathrm{diag}(\lambda_1, \cdots, \lambda_N)$ with $\lambda_1 \geq \lambda_2, \ldots, \lambda_N$, and $V = [v_1, \cdots, v_N]$ be the matrix of corresponding eigenvectors

$$AV = V\Lambda$$

Eigenvalue decomposition

An $N \times N$ matrix A: a map from $\mathbb{R}^N \to \mathbb{R}^N$.
An eigenvector v undergoes no rotation:

$$Av = \lambda v$$

λ is the corresponding eigenvalue, and gives the 'rescaling'.

Let $\Lambda = \text{diag}(\lambda_1, \cdots, \lambda_N)$ with $\lambda_1 \geq \lambda_2, \ldots, \lambda_N$, and $V = [v_1, \cdots, v_N]$ be the matrix of corresponding eigenvectors

$$AV = V\Lambda$$

Real Symmetric matrices have

- real eigenvalues
- different eigenvalues have orthogonal eigenvectors

Calculating eigenvalues and -vectors

Let A be any real symmetric matrix.

How to calculate the (absolute) largest eigenvalue and vector?

Calculating eigenvalues and -vectors

Let A be any real symmetric matrix.

How to calculate the (absolute) largest eigenvalue and vector?

Start with a random vector $\mathbf{u}_0$

- Define $\mathbf{u}_1 = A\mathbf{u}_0$, and normalize length.
- Repeat: $\mathbf{u}_i = A\mathbf{u}_{i-1}$, $\mathbf{u}_i = \mathbf{u}_i / \|\mathbf{u}_i\|$

Calculating eigenvalues and -vectors

Let A be any real symmetric matrix.

How to calculate the (absolute) largest eigenvalue and vector?

Start with a random vector $\mathbf{u}_0$

- Define $\mathbf{u}_1 = A\mathbf{u}_0$, and normalize length.
- Repeat: $\mathbf{u}_i = A\mathbf{u}_{i-1}$, $\mathbf{u}_i = \mathbf{u}_i / \|\mathbf{u}_i\|$

$$\mathbf{u}_i \rightarrow \mathbf{v}_1 \qquad \text{(Why?)}$$

Calculating eigenvalues and -vectors

Let A be any real symmetric matrix.

How to calculate the (absolute) largest eigenvalue and vector?

Start with a random vector $\mathbf{u}_0$

- Define $\mathbf{u}_1 = A\mathbf{u}_0$, and normalize length.
- Repeat: $\mathbf{u}_i = A\mathbf{u}_{i-1}$, $\mathbf{u}_i = \mathbf{u}_i / \|\mathbf{u}_i\|$

$$\mathbf{u}_i \to \mathbf{v}_1 \qquad \text{(Why?)}$$

This is the power method (Google's PageRank).

Calculating eigenvalues and -vectors

Let A be any real symmetric matrix.

How to calculate the (absolute) largest eigenvalue and vector?

Start with a random vector $\mathbf{u}_0$

- Define $\mathbf{u}_1 = A\mathbf{u}_0$, and normalize length.
- Repeat: $\mathbf{u}_i = A\mathbf{u}_{i-1}$, $\mathbf{u}_i = \mathbf{u}_i / \|\mathbf{u}_i\|$

$$\mathbf{u}_i \to \mathbf{v}_1 \qquad \text{(Why?)}$$

This is the power method (Google's PageRank).

How would we calculate λ_1?

Calculating eigenvalues and -vectors

Let A be any real symmetric matrix.

How to calculate the (absolute) largest eigenvalue and vector?

Start with a random vector $\mathbf{u}_0$

- Define $\mathbf{u}_1 = A\mathbf{u}_0$, and normalize length.
- Repeat: $\mathbf{u}_i = A\mathbf{u}_{i-1}$, $\mathbf{u}_i = \mathbf{u}_i / \|\mathbf{u}_i\|$

$$\mathbf{u}_i \to \mathbf{v}_1 \qquad \text{(Why?)}$$

This is the power method (Google's PageRank).

How would we calculate λ_1?

What if we wanted the second eigenvector?

Calculating eigenvalues and -vectors

Let A be any real symmetric matrix.

How to calculate the (absolute) largest eigenvalue and vector?

Start with a random vector $\mathbf{u}_0$

- Define $\mathbf{u}_1 = A\mathbf{u}_0$, and normalize length.
- Repeat: $\mathbf{u}_i = A\mathbf{u}_{i-1}$, $\mathbf{u}_i = \mathbf{u}_i / \|\mathbf{u}_i\|$

$$\mathbf{u}_i \to \mathbf{v}_1 \qquad \text{(Why?)}$$

This is the power method (Google's PageRank).

How would we calculate λ_1?

What if we wanted the second eigenvector?

- Adjust A so eigenvalue corresponding to $\mathbf{v}_1$ equals 0

$$\begin{bmatrix} 10 & 7 & 8 & 7 \\ 7 & 5 & 6 & 5 \\ 8 & 6 & 10 & 9 \\ 7 & 5 & 9 & 10 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 32 \\ 23 \\ 33 \\ 31 \end{bmatrix}$$

What is the solution? How about for $b = [32.1, 22.9, 33.1, 30.9]^T$?

$$\begin{bmatrix} 10 & 7 & 8 & 7 \\ 7 & 5 & 6 & 5 \\ 8 & 6 & 10 & 9 \\ 7 & 5 & 9 & 10 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 32 \\ 23 \\ 33 \\ 31 \end{bmatrix}$$

What is the solution? How about for $b = [32.1, 22.9, 33.1, 30.9]^T$?

Why the difference?

- the determinant?
- the inverse?
- the condition number?

An *ill-conditioned* problem can strongly amplify errors.

Gauss-Jordan elimination

$$\begin{bmatrix} 10 & 7 & 8 & 7 \\ 7 & 5 & 6 & 5 \\ 8 & 6 & 10 & 9 \\ 7 & 5 & 9 & 10 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 32 \\ 23 \\ 33 \\ 31 \end{bmatrix}$$

What is the solution? How about for $b = [32.1, 22.9, 33.1, 30.9]^T$?

Why the difference?

- the determinant?
- the inverse?
- the condition number?

An *ill-conditioned* problem can strongly amplify errors.

- Even without any rounding error

Stability analysis

The operator norm of a matrix A is

$$\|A\|_2 = \|A\| = \max_{\|v\|=1} \|Av\| = \max_{v} \frac{\|Av\|}{\|v\|}$$

The operator norm of a matrix A is

$$\|A\|_2 = \|A\| = \max_{\|v\|=1} \|Av\| = \max_{v} \frac{\|Av\|}{\|v\|}$$

For a symmetric, real matrix, $\|A\| = \lambda_{\max}(A)$ (why?)

For a general, real matrix, $\|A\| = \sqrt{\lambda_{\max}(A^T A)}$ (why?)

The operator norm of a matrix A is

$$\|A\|_2 = \|A\| = \max_{\|v\|=1} \|Av\| = \max_{v} \frac{\|Av\|}{\|v\|}$$

For a symmetric, real matrix, $\|A\| = \lambda_{\max}(A)$ (why?)

For a general, real matrix, $\|A\| = \sqrt{\lambda_{\max}(A^T A)}$ (why?)

For A in $\mathbb{R}^{N \times N}$ and any $v \in \mathbb{R}^N$,

$$\|Av\| \leq \|A\| \|v\| \qquad \text{(why?)}$$

The operator norm of a matrix A is

$$\|A\|_2 = \|A\| = \max_{\|v\|=1} \|Av\| = \max_{v} \frac{\|Av\|}{\|v\|}$$

For a symmetric, real matrix, $\|A\| = \lambda_{\max}(A)$ (why?)

For a general, real matrix, $\|A\| = \sqrt{\lambda_{\max}(A^T A)}$ (why?)

For A in $\mathbb{R}^{N \times N}$ and any $v \in \mathbb{R}^N$,

$$\|Av\| \leq \|A\| \|v\| \qquad \text{(why?)}$$

If $A = BC$, and all matrices are in $\mathbb{R}^{N \times N}$,

$$\|A\| \leq \|B\| \|C\| \qquad \text{(why?)}$$

Stability analysis

For a perturbation, δb let δx be the change in solution to $Ax = b$

$$A(x + \delta x) = b + \delta b$$

$\frac{\|\delta x\|}{\|x\|}$ is the relative change in the solution from the change $\frac{\|\delta b\|}{\|b\|}$

For a perturbation, δb let δx be the change in solution to $Ax = b$

$$A(x + \delta x) = b + \delta b$$

$\frac{\|\delta x\|}{\|x\|}$ is the relative change in the solution from the change $\frac{\|\delta b\|}{\|b\|}$

From $b = Ax$ and $\delta x = A^{-1}\delta b$, we have $\|b\| \leq \|A\|\|x\|$, and $\|\delta x\| \leq \|A^{-1}\|\|\delta b\|$:

Stability analysis

For a perturbation, δb let δx be the change in solution to $Ax = b$

$$A(x + \delta x) = b + \delta b$$

$\frac{\|\delta x\|}{\|x\|}$ is the relative change in the solution from the change $\frac{\|\delta b\|}{\|b\|}$

From $b = Ax$ and $\delta x = A^{-1}\delta b$, we have $\|b\| \leq \|A\|\|x\|$, and $\|\delta x\| \leq \|A^{-1}\|\|\delta b\|$:

$$\frac{\|\delta x\|}{\|x\|} \leq \|A\|\|A^{-1}\|\frac{\|\delta b\|}{\|b\|}$$

Stability analysis

For a perturbation, δb let δx be the change in solution to $Ax = b$

$$A(x + \delta x) = b + \delta b$$

$\frac{\|\delta x\|}{\|x\|}$ is the relative change in the solution from the change $\frac{\|\delta b\|}{\|b\|}$

From $b = Ax$ and $\delta x = A^{-1}\delta b$, we have $\|b\| \leq \|A\|\|x\|$, and $\|\delta x\| \leq \|A^{-1}\|\|\delta b\|$:

$$\frac{\|\delta x\|}{\|x\|} \leq \|A\|\|A^{-1}\|\frac{\|\delta b\|}{\|b\|}$$

Condition number of a matrix A is given by

$$\kappa(A) = \|A\|\|A^{-1}\|$$

Large condition number implies unstable solution

Stability analysis

$\kappa(A) \geq 1$ (why?)

Stability analysis

$\kappa(A) \geq 1$ (why?)

For a real symmetric matrix, $\kappa(A) = \frac{\lambda_{\max}(A)}{\lambda_{\min}(A)}$ (why?)

Stability analysis

$\kappa(A) \geq 1$ (why?)

For a real symmetric matrix, $\kappa(A) = \frac{\lambda_{\max}(A)}{\lambda_{\min}(A)}$ (why?)

For a real matrix, $\kappa(A) = \sqrt{\frac{\lambda_{\max}(A^T A)}{\lambda_{\min}(A^T A)}}$ (why?)

Stability analysis

$\kappa(A) \geq 1$ (why?)

For a real symmetric matrix, $\kappa(A) = \frac{\lambda_{\max}(A)}{\lambda_{\min}(A)}$ (why?)

For a real matrix, $\kappa(A) = \sqrt{\frac{\lambda_{\max}(A^T A)}{\lambda_{\min}(A^T A)}}$ (why?)

Condition number is a property of a problem

Stability is a property of an algorithm

A bad algorithm can mess up a simple problem

Gaussian elimination with partial pivoting

Consider reducing to upper triangular

$$\begin{bmatrix} v_{11} & v_{12} & v_{13} \\ v_{21} & v_{22} & v_{23} \\ v_{31} & v_{32} & v_{33} \end{bmatrix}$$

Gaussian elimination: divide row 1 by v_{11}

Gaussian elimination with partial pivoting

Consider reducing to upper triangular

$$\begin{bmatrix} v_{11} & v_{12} & v_{13} \\ v_{21} & v_{22} & v_{23} \\ v_{31} & v_{32} & v_{33} \end{bmatrix}$$

Gaussian elimination: divide row 1 by v_{11}

$$\begin{bmatrix} 1e-4 & 1 \\ 1 & 1 \end{bmatrix}$$

Gaussian elimination with partial pivoting

Consider reducing to upper triangular

$$\begin{bmatrix} v_{11} & v_{12} & v_{13} \\ v_{21} & v_{22} & v_{23} \\ v_{31} & v_{32} & v_{33} \end{bmatrix}$$

Gaussian elimination: divide row 1 by v_{11}

$$\begin{bmatrix} 1e-4 & 1 \\ 1 & 1 \end{bmatrix}$$

Partial pivoting: Pivot rows to bring $\max_r v_{r1}$ to top

Can dramatically improve performance. E.g.

Consider reducing to upper triangular

$$\begin{bmatrix} v_{11} & v_{12} & v_{13} \\ v_{21} & v_{22} & v_{23} \\ v_{31} & v_{32} & v_{33} \end{bmatrix}$$

Gaussian elimination: divide row 1 by v_{11}

$$\begin{bmatrix} 1e-4 & 1 \\ 1 & 1 \end{bmatrix}$$

Partial pivoting: Pivot rows to bring $\max_r v_{r1}$ to top

Can dramatically improve performance. E.g.

Why does it work?

Gaussian elimination with partial pivoting

Recall Gaussian elimination decomposes $A = LU$ and solves two intermediate problems.

What are the condition numbers of L and U?

Try

$$\begin{bmatrix} 1e-4 & 1 \\ 1 & 1 \end{bmatrix}$$

Note: `R` does pivoting for you automatically! (see the function `lu` in package `Matrix`)

QR decomposition

In general, for $A = BC$, $\kappa(A) \leq \kappa(B)\kappa(C)$ (why?)

QR decomposition

In general, for $A = BC$, $\kappa(A) \leq \kappa(B)\kappa(C)$ (why?)

QR decomposition:

$$A = QR$$

Here, R is an upper (right) triangular matrix. Q is an orthonormal matrix: $Q^T Q = I$

QR decomposition

In general, for $A = BC$, $\kappa(A) \leq \kappa(B)\kappa(C)$ (why?)

QR decomposition:

$$A = QR$$

Here, R is an upper (right) triangular matrix. Q is an orthonormal matrix: $Q^T Q = I$

$$\kappa(A) = \kappa(Q)\kappa(R) \quad \text{(why?)}$$

QR decomposition

In general, for $A = BC$, $\kappa(A) \le \kappa(B)\kappa(C)$ (why?)

QR decomposition:

$$A = QR$$

Here, R is an upper (right) triangular matrix. Q is an orthonormal matrix: $Q^TQ = I$

$$\kappa(A) = \kappa(Q)\kappa(R) \quad \text{(why?)}$$

Can use to solve $Ax = b$ (How?)

Most stable decomposition

QR DECOMPOSITION

In general, for $A = BC$, $\kappa(A) \leq \kappa(B)\kappa(C)$ (why?)

QR decomposition:

$$A = QR$$

Here, R is an upper (right) triangular matrix. Q is an orthonormal matrix: $Q^T Q = I$

$$\kappa(A) = \kappa(Q)\kappa(R) \quad \text{(why?)}$$

Can use to solve $Ax = b$ (How?)

Most stable decomposition

Does this mean we should use QR decomposition?

Gram-Schmidt orthonormalization

Given N vectors $\mathbf{x}_1, \ldots, \mathbf{x}_N$ construct an orthonormal basis:

Gram-Schmidt orthonormalization

Given N vectors $\mathbf{x}_1, \ldots, \mathbf{x}_N$ construct an orthonormal basis:

$$\mathbf{u}_1 = \mathbf{x}_1 / \|\mathbf{x}_1\|$$

$$\tilde{\mathbf{u}}_i = \mathbf{x}_i - \sum_{j=1}^{i-1} (\mathbf{x}_i^T \mathbf{u}_j) \mathbf{u}_i, \quad \mathbf{u}_i = \tilde{\mathbf{u}}_i / \|\tilde{\mathbf{u}}_i\| \quad i = 2 \ldots, N$$

QR DECOMPOSITION

$$A \quad = \quad Q \quad R$$

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} q_{11} & q_{12} & q_{13} \\ q_{21} & q_{22} & q_{23} \\ q_{31} & q_{32} & q_{33} \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ 0 & r_{22} & r_{23} \\ 0 & 0 & r_{33} \end{bmatrix}$$

QR decomposition

$$A \quad = \quad Q \quad R$$

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} q_{11} & q_{12} & q_{13} \\ q_{21} & q_{22} & q_{23} \\ q_{31} & q_{32} & q_{33} \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ 0 & r_{22} & r_{23} \\ 0 & 0 & r_{33} \end{bmatrix}$$

QR decomposition: Gram-Schmidt on columns of A
(can you see why?)

QR decomposition

$$\begin{matrix} A & = & Q & R \end{matrix}$$

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} q_{11} & q_{12} & q_{13} \\ q_{21} & q_{22} & q_{23} \\ q_{31} & q_{32} & q_{33} \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ 0 & r_{22} & r_{23} \\ 0 & 0 & r_{33} \end{bmatrix}$$

QR decomposition: Gram-Schmidt on columns of A
(can you see why?)

Of course, there are more stable/efficient ways of doing this
(Householder rotation/Givens rotation)

$O(N^3)$ algorithms (though about twice as slow as LU)

QR algorithm

Algorithm to calculate all eigenvalues/eigenvectors of a (not too-large) matrix

Start with $A_0 = A$. At iteration i:

- $A_i = Q_i R_i$
- $A_{i+1} = R_i Q_i$

Then A_i and A_{i+1} have the same eigenvalues (why?), and the diagonal contains the eigenvalues.

QR algorithm

Algorithm to calculate all eigenvalues/eigenvectors of a (not too-large) matrix

Start with $A_0 = A$. At iteration i:

- $A_i = Q_i R_i$
- $A_{i+1} = R_i Q_i$

Then A_i and A_{i+1} have the same eigenvalues (why?), and the diagonal contains the eigenvalues. Can be made more stable/efficient.

One of Dongarra & Sullivan (2000)'s list of top 10 algoirithms.
`https://www.siam.org/pdf/news/637.pdf`

See also number 4, "decompositional approach to matrix computations"

$$\log(p(X|\mu,\Sigma)) = -\frac{1}{2}(X-\mu)^T\Sigma^{-1}(X-\mu) - \frac{N}{2}\log 2\pi - \frac{1}{2}\log|\Sigma|$$

Multivariate normal

$$\log(p(X|\mu, \Sigma)) = -\frac{1}{2}(X-\mu)^T\Sigma^{-1}(X-\mu) - \frac{N}{2}\log 2\pi - \frac{1}{2}\log|\Sigma|$$

$$\Sigma = LL^T$$
$$Y = L^{-1}(X-\mu) \quad \text{(Forward solve)}$$

$$\log(p(X|\mu,\Sigma)) = -\frac{1}{2}(X-\mu)^T\Sigma^{-1}(X-\mu) - \frac{N}{2}\log 2\pi - \frac{1}{2}\log|\Sigma|$$

$$\begin{aligned} \Sigma &= LL^T \\ Y &= L^{-1}(X-\mu) \quad \text{(Forward solve)} \\ \log(p(X|\mu,\Sigma)) &= -\frac{1}{2}Y^TY - \frac{N}{2}\log 2\pi - \log|\Sigma| \end{aligned}$$

Multivariate normal

$$\log(p(X|\mu,\Sigma)) = -\frac{1}{2}(X-\mu)^T\Sigma^{-1}(X-\mu) - \frac{N}{2}\log 2\pi - \frac{1}{2}\log|\Sigma|$$

$$\begin{aligned}\Sigma &= LL^T \\ Y &= L^{-1}(X-\mu) \quad \text{(Forward solve)} \\ \log(p(X|\mu,\Sigma)) &= -\frac{1}{2}Y^TY - \frac{N}{2}\log 2\pi - \log|\Sigma|\end{aligned}$$

Can also just forward solve for L^{-1}: $LL^{-1} = I$
(Inverted triangular matrix isn't too bad)

Sampling a multivariate normal

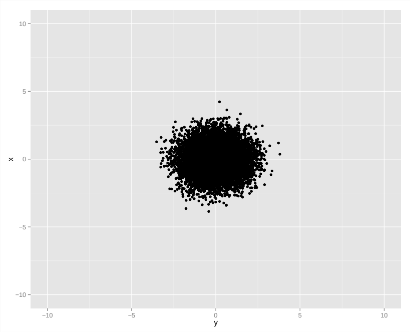
Sampling a univariate normal:

- Inversion method (default for `rnorm`?).
- Box-Muller transform: (Z_1, Z_2) : independent standard normals.

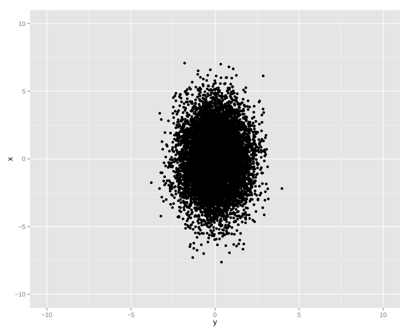
Sampling a multivariate normal

Sampling a univariate normal:

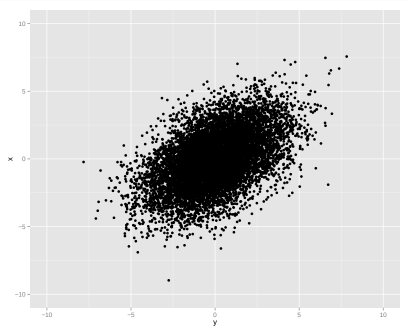
- Inversion method (default for `rnorm`?).
- Box-Muller transform: (Z_1, Z_2) : independent standard normals.
- Let $Z \sim \mathcal{N}(0, I)$
- $X = \mu + LZ$
- $Z = \mathcal{N}(\mu, L^T L)$



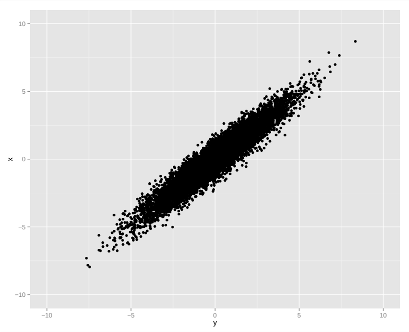
$$\Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$



$$\Sigma = \begin{bmatrix} 4 & 0 \\ 0 & 1 \end{bmatrix}$$



$$\Sigma = \begin{bmatrix} 4 & 2 \\ 2 & 4 \end{bmatrix}$$



$$\Sigma = \begin{bmatrix} 4 & 3.8 \\ 3.8 & 4 \end{bmatrix}$$