Data 345

Applied Linear Algebra for Statistical Learning

Class 6 (Sep. 15, 2025)

Systems of Linear Equations

- The set of all 4-dimensional vectors satisfying this condition is called the **general solution** of the system of equations.
- In this case, the general solution would be

$$\left\{ \boldsymbol{x} \in \mathbb{R}^4 : \ \boldsymbol{x} = \begin{bmatrix} 42 \\ 8 \\ 0 \\ 0 \end{bmatrix} + \lambda_1 \begin{bmatrix} 8 \\ 2 \\ -1 \\ 0 \end{bmatrix} + \lambda_2 \begin{bmatrix} -4 \\ 12 \\ 0 \\ -1 \end{bmatrix} \quad (\lambda_1, \lambda_2 \in \mathbb{R}) \right\}$$

- Since λ_1 and λ_2 may be *any* real number, there are infinitely many possible solutions, and in particular, the defining expression constitutes a plane in $\mathbb{R}^4$.

Systems of Linear Equations

- We used a basic general procedure:
  - Find a single (particular) solution to $A\boldsymbol{x} = \boldsymbol{b}$.
  - Find a way to describe all solutions $A\boldsymbol{x} = \boldsymbol{0}$.
  - Combine the solutions above to get a general solution to $A\boldsymbol{x} = \boldsymbol{b}$.
- This was easy to do in our example since the matrix A was already set up nicely, but how can we get it in such a form?
- With matrices, as you might recall, we can perform basic operations without changing our solution set.
  - Exchange places of two equations (i.e., switch two rows in the matrix)
  - Multiply both sides of an equation by a constant (i.e., multiply a row by a constant)
  - Add two equations' respective sides (add two rows in the matrix)

(Reduced) Row-Echelon Form

- A matrix is in **row-echelon form** if:
  - All rows containing only zeros are at the bottom of the matrix, and
  - The first nonzero entry of a nonzero row (called a **pivot**)is always strictly to the right of the first nonzero entry of the row above it.
- Example:

$$\begin{aligned} -2x_1 + 4x_2 - 2x_3 - x_4 + 4x_5 &= -3 \\ 4x_1 - 8x_2 + 3x_3 - 3x_4 + x_5 &= 2 \\ x_1 - 2x_2 + x_3 - x_4 + x_5 &= 0 \\ x_1 - 2x_2 - 3x_4 + 4x_5 &= -1 \end{aligned}$$

$$\begin{bmatrix} -2 & 4 & -2 & -1 & 4 \\ 4 & -8 & 3 & -3 & 1 \\ 1 & -2 & 1 & -1 & 1 \\ 1 & -2 & 0 & -3 & 4 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} -3 \\ 2 \\ 0 \\ -1 \end{bmatrix}$$

$$A\boldsymbol{x} = \boldsymbol{b}$$

$$\left[\begin{array}{ccccc|c} -2 & 4 & -2 & -1 & 4 & -3 \\ 4 & -8 & 3 & -3 & 1 & 2 \\ 1 & -2 & 1 & -1 & 1 & 0 \\ 1 & -2 & 0 & -3 & 4 & -1 \end{array}\right]$$

"augmented matrix" $[A|\boldsymbol{b}]$

(Reduced) Row-Echelon Form

$$\left[\begin{array}{ccccc|c} -2 & 4 & -2 & -1 & 4 & -3 \\ 4 & -8 & 3 & -3 & 1 & 2 \\ 1 & -2 & 1 & -1 & 1 & 0 \\ 1 & -2 & 0 & -3 & 4 & -1 \end{array}\right] \xrightarrow[R_1 \leftrightarrow R_3]{} \left[\begin{array}{ccccc|c} 1 & -2 & 1 & -1 & 1 & 0 \\ 4 & -8 & 3 & -3 & 1 & 2 \\ -2 & 4 & -2 & -1 & 4 & -3 \\ 1 & -2 & 0 & -3 & 4 & -1 \end{array}\right]$$

$$\xrightarrow[\substack{R_2 - 4R_1 \\ R_3 + 2R_1 \\ R_4 - R_1}]{} \left[\begin{array}{ccccc|c} 1 & -2 & 1 & -1 & 1 & 0 \\ 0 & 0 & -1 & 1 & -3 & 2 \\ 0 & 0 & 0 & -3 & 6 & -3 \\ 0 & 0 & -1 & -2 & 3 & -1 \end{array}\right] \xrightarrow[R_4 - R_2]{} \left[\begin{array}{ccccc|c} 1 & -2 & 1 & -1 & 1 & 0 \\ 0 & 0 & -1 & 1 & -3 & 2 \\ 0 & 0 & 0 & -3 & 6 & -3 \\ 0 & 0 & 0 & -3 & 6 & -3 \end{array}\right]$$

$$\xrightarrow[R_4 - R_3]{} \left[\begin{array}{ccccc|c} 1 & -2 & 1 & -1 & 1 & 0 \\ 0 & 0 & -1 & 1 & -3 & 2 \\ 0 & 0 & 0 & -3 & 6 & -3 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right] \xrightarrow[\substack{-1 \cdot R_2 \\ \left(-\frac{1}{3}\right) \cdot R_3}]{} \left[\begin{array}{ccccc|c} 1 & -2 & 1 & -1 & 1 & 0 \\ 0 & 0 & 1 & -1 & 3 & -2 \\ 0 & 0 & 0 & 1 & -2 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right]$$

(Reduced) Row-Echelon Form

Pivot (x_1)

$$\left[\begin{array}{ccccc|c} 1 & -2 & 1 & -1 & 1 & 0 \\ 0 & 0 & 1 & -1 & 3 & -2 \\ 0 & 0 & 0 & 1 & -2 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right]$$

Pivot (x_3)

Pivot (x_4)

Variables x_1, x_3, and x_4 are called **basic** variables, while x_2 and x_5 are called **free** variables. Free variables may be chosen to be any real number, which then determines the values of the basic variables.

Suppose $x_2 = x_5 = 0$.

$$x_4 - 2x_5 = 1 \longrightarrow x_4 - 2 \cdot 0 = 1 \longrightarrow x_4 = 1$$

$$x_3 - x_4 + 3x_5 = -2 \longrightarrow x_3 - 1 + 3 \cdot 0 = -2 \longrightarrow x_3 = -1$$

$$x_1 - 2x_2 + x_3 - x_4 + x_5 = 0 \longrightarrow x_1 - 2 \cdot 0 - 1 - 1 + 0 = 0 \longrightarrow x_1 = 2$$

So, a particular solution would be $\begin{bmatrix} 2 \\ 0 \\ -1 \\ 1 \\ 0 \end{bmatrix}$

(Reduced) Row-Echelon Form

Particular solution: $\begin{bmatrix} 2 \\ 0 \\ -1 \\ 1 \\ 0 \end{bmatrix}$

Since $\boldsymbol{c}_2 = -2\boldsymbol{c}_1$ we know that $2\boldsymbol{c}_1 + \boldsymbol{c}_2 = \boldsymbol{0}$.

Similarly, $\boldsymbol{c}_5 = -2\boldsymbol{c}_1 + \boldsymbol{c}_3 - 2\boldsymbol{c}_4$, so $2\boldsymbol{c}_1 - \boldsymbol{c}_3 + 2\boldsymbol{c}_4 + \boldsymbol{c}_5 = \boldsymbol{0}$.

General solution: $\left\{ \boldsymbol{x} \in \mathbb{R}^5 : \boldsymbol{x} = \begin{bmatrix} 2 \\ 0 \\ -1 \\ 1 \\ 0 \end{bmatrix} + \lambda_1 \begin{bmatrix} 2 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \lambda_2 \begin{bmatrix} 2 \\ 0 \\ -1 \\ 2 \\ 1 \end{bmatrix} \quad (\lambda_1, \lambda_2 \in \mathbb{R}) \right\}$

(Reduced) Row-Echelon Form

- A matrix in row-echelon form is in **reduced row-echelon form** provided:
  - Every pivot value is 1.
  - The pivot is the only nonzero entry in its column.

$$\left[\begin{array}{ccccc|c} 1 & -2 & 1 & -1 & 1 & 0 \\ 0 & 0 & 1 & -1 & 3 & -2 \\ 0 & 0 & 0 & 1 & -2 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right] \xrightarrow{\text{(additional row operations)}} \left[\begin{array}{ccccc|c} 1 & -2 & 0 & 0 & -2 & 2 \\ 0 & 0 & 1 & 0 & 1 & -1 \\ 0 & 0 & 0 & 1 & -2 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right]$$

In row-echelon form, but **not** in reduced row-echelon form

Now in reduced row-echelon form

- The process/algorithm by which you use elementary row operations to bring a matrix into reduced row-echelon form is called **Gaussian elimination**.

Why RREF?

- Remember that a matrix in RREF only has 1 as its pivot values and no other nonzero values in that column.
- This makes obtaining the general solution **much easier** when the matrix is in RREF, if there are infinitely many solutions to the system.
- In the case where the solution is unique, performing Gaussian elimination on the matrix $[A|\boldsymbol{b}]$ will simply yield $[I_n|\boldsymbol{x}]$, which means that the last column in the matrix *is* the unique solution.
- For matrices that have inverses, the inverse is computed by Gaussian elimination.
- Supposing an $n \times n$ matrix A has inverse A^{-1}, performing Gaussian elimination on the augmented matrix $[A|I_n]$ yields $[I_n|A^{-1}]$.

$$A = \begin{bmatrix} 1 & 0 & 2 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 2 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \longrightarrow \left[\begin{array}{cccc|cccc} 1 & 0 & 2 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 2 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{array}\right] \longrightarrow \left[\begin{array}{cccc|cccc} 1 & 0 & 0 & 0 & -1 & 2 & -2 & 2 \\ 0 & 1 & 0 & 0 & 1 & -1 & 2 & -2 \\ 0 & 0 & 1 & 0 & 1 & -1 & 1 & -1 \\ 0 & 0 & 0 & 1 & -1 & 0 & -1 & 2 \end{array}\right]$$

A^{-1}

Solving Systems of Linear Equations

- If you felt like sitting through a relatively small Gaussian elimination was incredibly tedious, it's not just you.
- As an algorithm, Gaussian elimination on an $n \times n$ matrix is $\mathcal{O}(n^3)$ time complexity (big-oh n cubed). This means that the time to complete the algorithm grows roughly like a cubic polynomial.
- For small (2×2, 3×3, and even 4×4) systems this is manageable by hand, if unpleasant. Even when automating, Gaussian elimination is prohibitively slow for large systems.
- It's not super useful to implement automated Gaussian elimination for this reason, even though the algorithm itself isn't extremely complicated.

Alternatives to Gaussian Elimination

- If we are to solve $A\boldsymbol{x} = \boldsymbol{b}$ and A is a square $n \times n$ matrix whose inverse is known, then there's an easy shortcut to solving for $\boldsymbol{x}$.

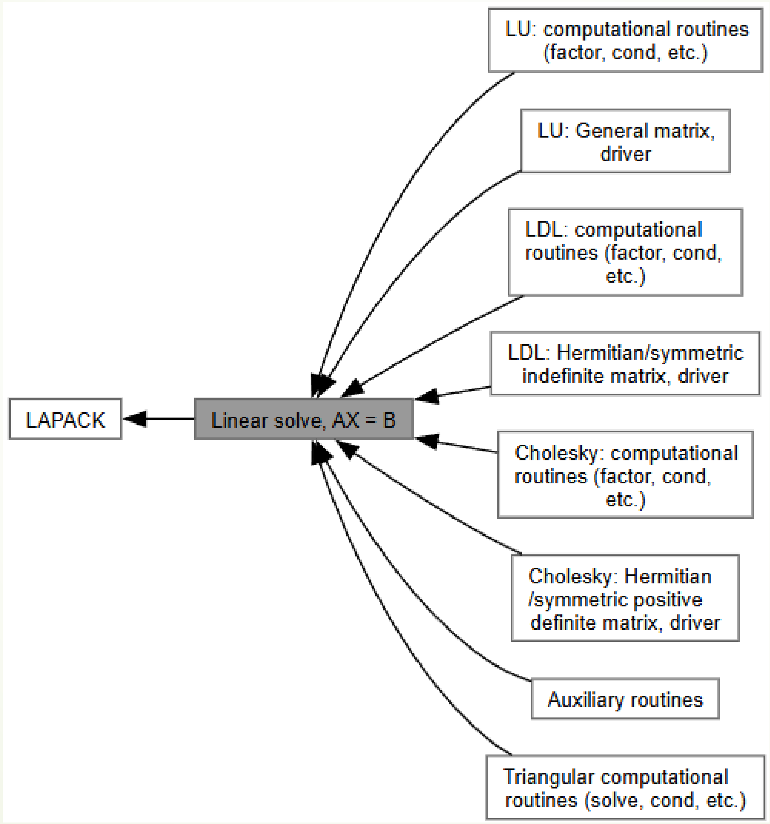
$$\begin{aligned} A\boldsymbol{x} &= \boldsymbol{b} \\ A^{-1}(A\boldsymbol{x}) &= A^{-1}\boldsymbol{b} \\ (A^{-1}A)\boldsymbol{x} &= A^{-1}\boldsymbol{b} \\ I_n\boldsymbol{x} &= A^{-1}\boldsymbol{b} \\ \boldsymbol{x} &= A^{-1}\boldsymbol{b} \end{aligned}$$

- This requires A to be both square and invertible, which is not necessarily realistic. In addition, it requires the inverse. If the inverse isn't known, then it must be computed.

Alternatives to Gaussian Elimination

- The package NumPy has a linear algebra solver `<np.linalg.solve()>` where it can take in a matrix A and a column vector $\boldsymbol{b}$ and output the solution to the system $A\boldsymbol{x} = \boldsymbol{b}$, but this solver assumes that A is square and invertible.
- Uses a general procedure called LU-decomposition to compute the solution (basically all solvers do), which is $\sim\mathcal{O}(n^3)$, but there are many conditions A could satisfy that would significantly decrease this computation time, so it also runs tests and solves on a roughly case-by-case basis.
- If the matrix A is not square or not invertible then the package is not programmed to pursue an **exact** solution but rather an approximate one. That is, the vector $\boldsymbol{x}^*$ returned by NumPy's "least square" solver is the vector so that the distance $||\boldsymbol{b} - A\boldsymbol{x}^*||$ is minimized.

Alternatives to Gaussian Elimination



Basically, most numerical/computational approaches to solving systems of linear equations rely on cleverly “factoring” a matrix to be a product of matrices in a nicer form.

Linear Algebra For the Rest of Us

- The business of solving linear equations is incredibly tricky, especially for large systems.
- So many quantities of interest require solutions to some system of equations or another, even if you aren't originally setting out to solve a system of equations.
- **We will solve small systems using Gaussian elimination**, but resort to "outside" solvers in most cases for the systems we need to solve so we can focus on other aspects of linear algebra.
- To solve small systems, all we need to do is implement the three basic row operations.