UNIVERSITY AT ALBANY
STATE UNIVERSITY OF NEW YORK

CSI 401 (Fall 2025)

Numerical Methods

Lecture 10: Optimization: Gradient Descent

Chong Liu

Department of Computer Science

Oct 20, 2025

Recap: Matrix rank of A ($m \times n$ matrix)

- Definition:
  - Maximal number of linearly independent columns or maximal number of linearly independent rows.

- Two examples: Find ranks using Gaussian Elimination.

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix}$$

$$B = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 6 \\ 0 & 1 & 1 \end{bmatrix} \qquad \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 6 \\ 0 & 1 & 1 \end{bmatrix} \xrightarrow{R_2 \leftarrow R_2 - 2R_1} \begin{bmatrix} 1 & 2 & 3 \\ 0 & 0 & 0 \\ 0 & 1 & 1 \end{bmatrix}$$

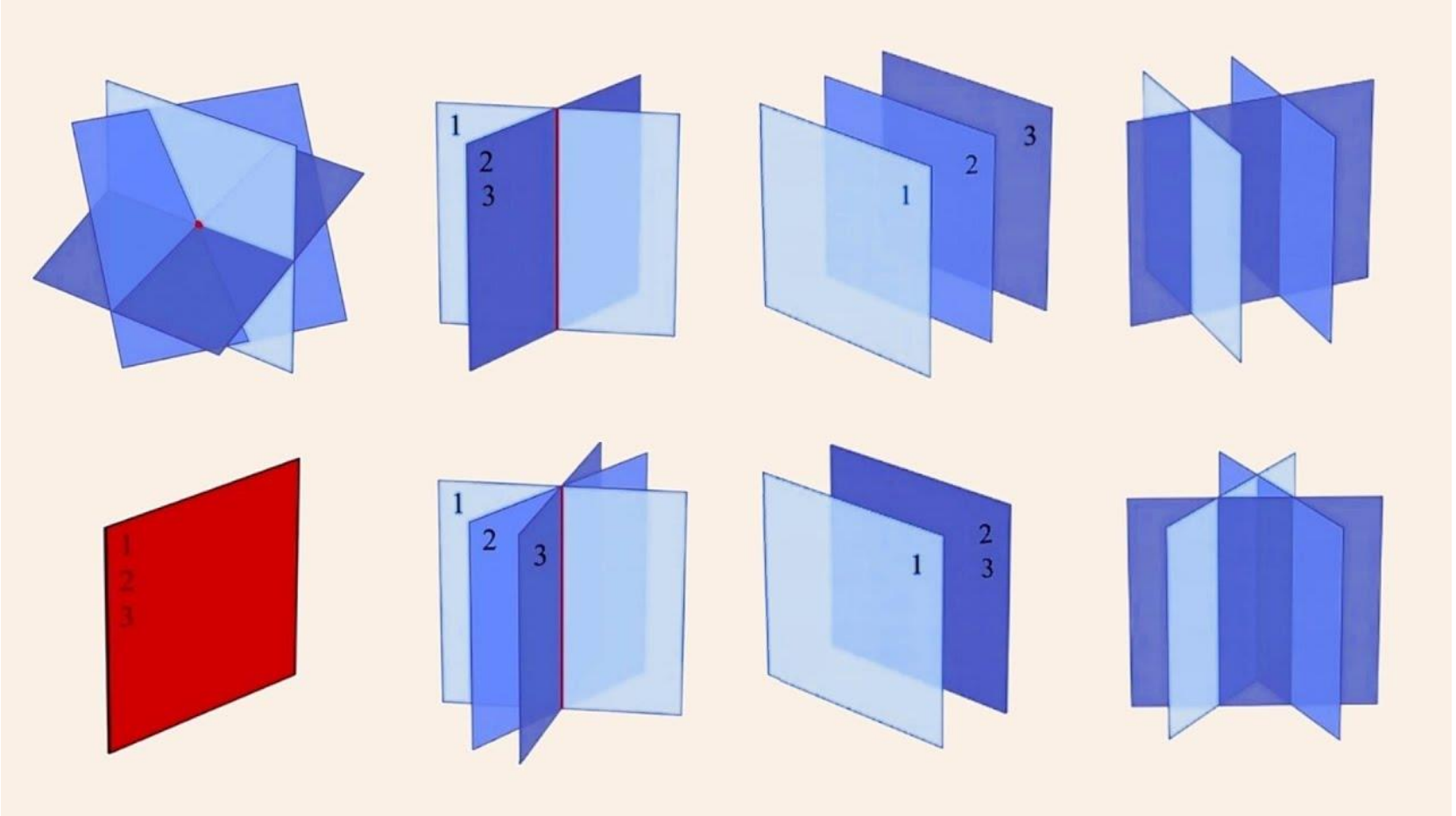
- Rank is number of pivots after Gaussian Elimination.

Recap: Summary of Conditions for Solutions of a Linear System $Ax = \mathrm{b}$

Case	Rank Condition	Number of Solutions	Geometric Interpretation
No Solution	$\mathrm{rank}(A) < \mathrm{rank}([A \mid \mathbf{b}])$	None (inconsistent system)	Hyperplanes do not intersect (contradictory equations)
Unique Solution	$\mathrm{rank}(A) = \mathrm{rank}([A \mid \mathbf{b}]) = n$	Exactly **one**	Hyperplanes intersect at a single point
Infinitely Many Solutions	$\mathrm{rank}(A) = \mathrm{rank}([A \mid \mathbf{b}]) < n$	Infinitely many	Hyperplanes intersect along a line, plane, or higher-dimensional subspace

Recap: Geometric view of solutions (3-d case)

- Discussion:
  - Which figure shows a *unique* solution?
  - Which figure shows *infinitely many* solutions?
  - Which figure shows *no* solution?



Recap: Linear model for housing price prediction

- In vector form:
  - $\text{Price}(x) = x^{\text{T}} w$
  - $x = [x_1, x_2, \dots, x_8]$: feature vector
  - $w = [w_1, w_2, \dots, x_8]$: parameter vector

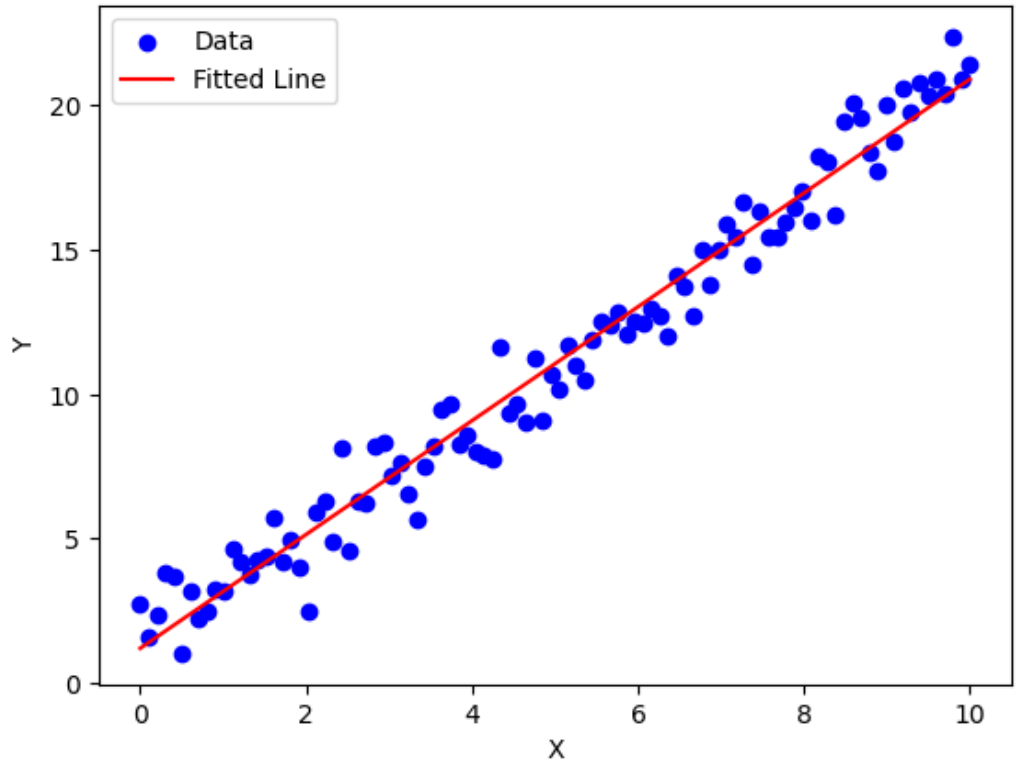
- As long as we find a good w, we have a good linear model.

- In a general form:
  - Xw
    - X is a $n \times d$ matrix.
      - n is the number of houses.
      - d is the number of features for describing the house.
    - w is a d-dimensional vector.

Recap: Considering conditions of linear systems

- In real-world applications, there are many challenges.
  - No solution
  - Noisy data
  - Overdetermined systems (most common case)
    - Fitting a hyperplane (a line in 2-d) to too many data points.

- So our goal reduces to find the an approximate w that best describes the data!
- How?



Recap: The objective function for learning linear regression under square loss

- $\widehat{w} = \text{argmin}_w \frac{1}{n} \sum_{i=1}^{n} (x_i^T w - y_i)^2 = \text{argmin}_w \|Xw - y\|_2^2$
  - aka: Ordinary Least Square (OLS)
- In-class exercise: solve this optimization problem by setting gradient of the objective function to 0.

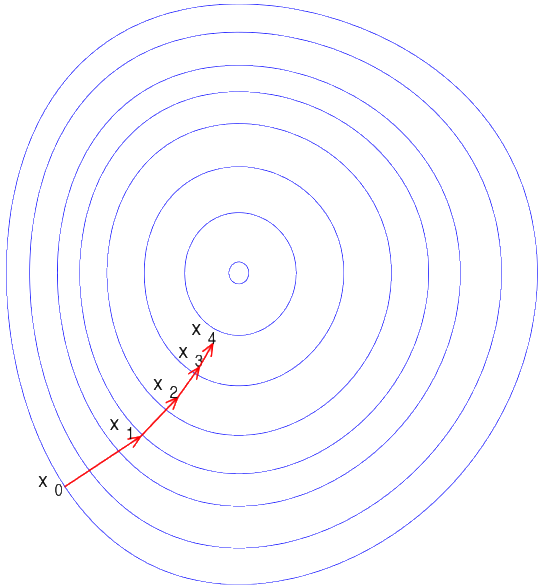
Agenda

- Gradient Descent
- How to solve linear regression?
  - Direct solver
  - Gradient Descent
- Improved version of GD: Stochastic gradient descent
- Time complexity analysis

How do we optimize a continuously differentiable function in general?

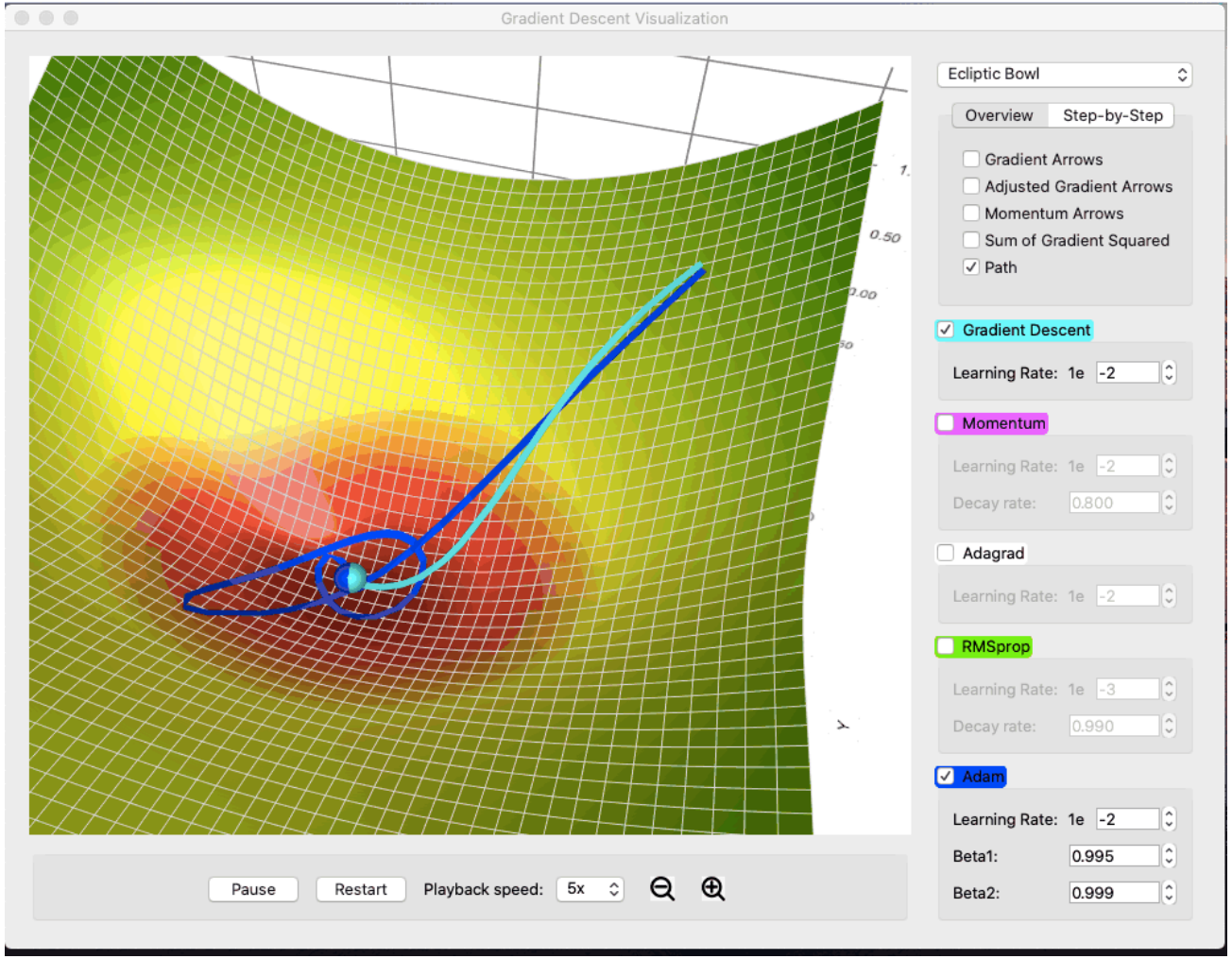
- The problem: $\min_{\theta} f(\theta)$
- Discussion: How do you solve this optimization problem?
- Gradient descent in iterations

$$\theta_{t+1} = \theta_t - \eta_t \nabla f(\theta_t)$$



Gradient Descent Demo in 2-D

- An excellent demo tool:
  - https://github.com/lilipads/gradient_descent_viz



Gradient descent for quadratic function

- $\min f(x) = x^2$

- Follow me on the first two examples:

1. Find x_4 given $x_0 = 2, \eta = 0.1$
2. Find x_4 given $x_0 = 2, \eta = 0.4$

- In-class exercise questions:

1. Find x_4 given $x_0 = 4, \eta = 0.4$
2. Find x_4 given $x_0 = 2, \eta = 1.5$ (what did you find?)

Condition of convergence of gradient descent

- Smoothness (Lipschitz-continuous gradient)
  - There exists $L > 0$ such that
    - $\| \nabla f(x) - \nabla f(y) \| \leq L \| x - y \| \quad \forall x, y$
    - L is called the Lipschitz constant of the gradient.
  - Intuitively: the function's curvature isn't too steep.

- If f is L-Lipschitz smooth, GD **converges** to a local minimum when $0 < \eta < \frac{2}{L}$.
  - If $\eta > \frac{2}{L}$:divergence
  - If $\eta = \frac{1}{L}$:optimal rate for fixed-step GD in many cases

Gradient descent for quadratic function

- Without proof, $L = 2$ for x^2

- $\min f(x) = x^2$

- In-class exercise questions:

1. Find x_4 given $x_0 = 2, \eta = 0.5$
2. Find x_4 given $x_0 = 2, \eta = 1$

Discussion: What did you find?

Back to linear regression: How to solve it using Gradient Descent?

- $\widehat{w} = \text{argmin}_w \frac{1}{n} \sum_{i=1}^{n} (x_i^T w - y_i)^2 = \text{argmin}_w \|Xw - y\|_2^2$

- In-class exercise: Write the GD updating rule for solving w.
  - $w \leftarrow w - 2\eta X^T (Xw - y)$

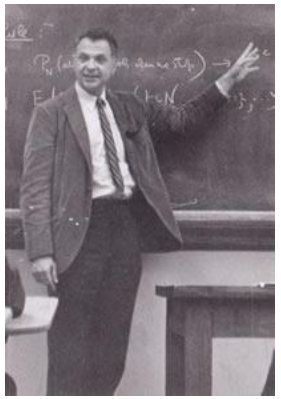
Checkpoint

- Least square:
  - Heavily used in practice, due to
    - Large datasets (many data points)
    - Noisy data
    - No solution based on conditions of linear systems
- Linear regression
  - $\widehat{w} = \text{argmin}_w \frac{1}{n} \sum_{i=1}^{n} (x_i^T w - y_i)^2 = \text{argmin}_w \|Xw - y\|_2^2$
  - Direct solver: $\widehat{w} = (X^T X)^{-1} X^T y$
  - GD: $w \leftarrow w - 2\eta X^T (Xw - y)$

Stochastic Gradient Descent (Robbins-Monro 1951)

- Gradient descent

$$\theta_{t+1} = \theta_t - \eta_t \nabla f(\theta_t)$$



Herbert Robbins
1915 - 2001

- Stochastic gradient descent
  - Using a stochastic approximation of the gradient:

$$\theta_{t+1} = \theta_t - \eta_t \hat{\nabla} f(\theta_t)$$

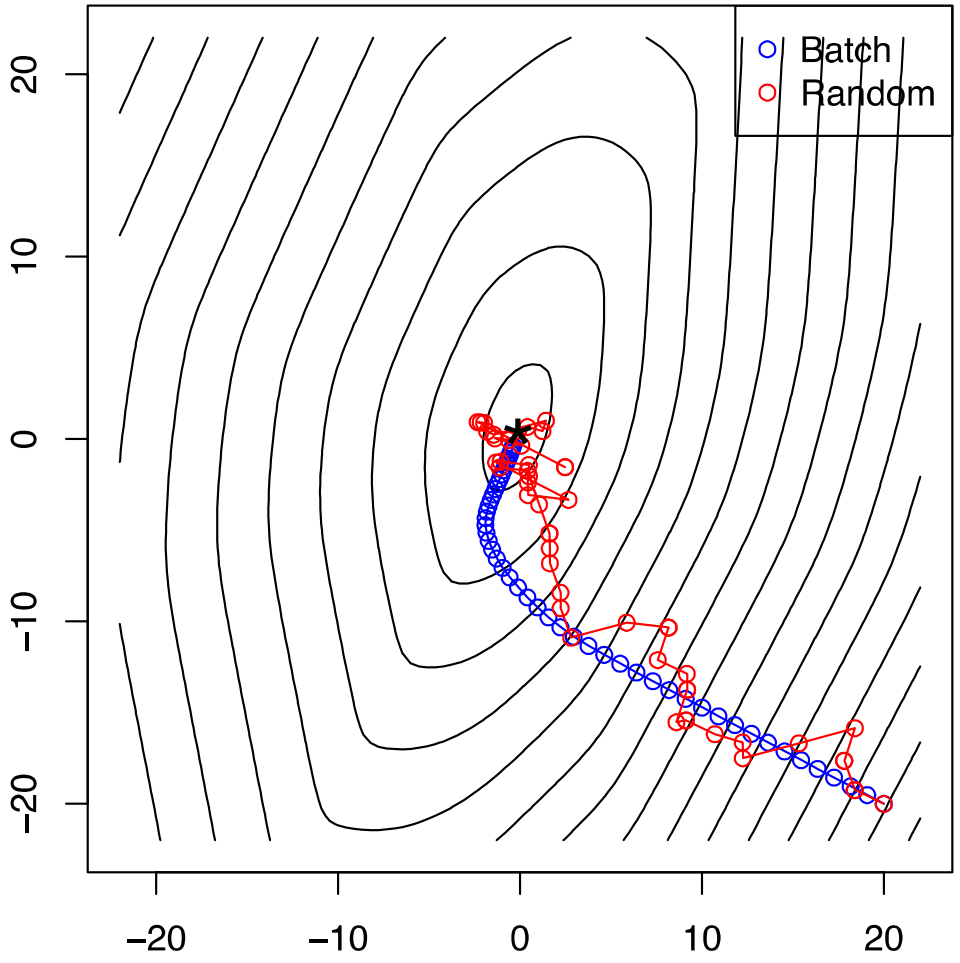
A natural choice of SGD in machine learning

- Recall that

$$\min_{\theta \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} \ell(\theta, (x_i, y_i))$$

- SGD samples a data point i uniformly at random while GD uses all data!
  - Use $\nabla_\theta \ell(\theta, (x_i, y_i))$

Illustration of GD vs SGD



Time complexity:

GD: $O(nd * \text{n\_iterations})$

SGD: $O(d * \text{n\_iterations})$

How to choose the step sizes / learning rates?

- In practice:
  - Fixed learning rate for SGD is usually fine.
  - If it diverges, decrease the learning rate.

The power of SGD

- Extremely simple:
  - A few lines of code
- Extremely scalable
  - Just a few pass of the data, no need to store the data
- Extremely general:
  - In addition to linear regression, in practice it can solve most optimization problems of differentiable functions
    - E.g., Training neural networks, Transformer, Generative Pretrained Transformer
  - Foundational algorithm of the AI revolution as we see today!

Time complexity of direct solver and GD/SGD for solving linear regression

- Direct solver
  - $O(nd^2 + d^3)$
- GD:
  - $O(ndT)$
- SGD:
  - $O(dT)$

- $T = \text{n\_iterations}$