Linear Regression and Gradient Descent

CMSC 173

September 7, 2025

Outline

What is Linear Regression?

- **Definition:** Linear regression is a method to model the relationship between a dependent variable y and one or more independent variables $x_1, x_2, \ldots, x_n$.
- **Motivation:**
  - Predict future outcomes based on observed data.
  - Understand the strength and form of relationships between variables.
  - Provide a simple, interpretable baseline model.
- **Mathematical Formulation:**

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$$

- **Definitions:**
  - $y \in \mathbb{R}$: target (dependent) variable.
  - $\hat{y} \in \mathbb{R}$: predicted value of y.
  - $x_j \in \mathbb{R}$: j-th independent feature, $j = 1, \ldots, n$.
  - n: number of features (excluding bias term).
  - $\theta_0 \in \mathbb{R}$: intercept (bias term).
  - $\theta_j \in \mathbb{R}$: coefficient for feature x_j.

Linear Regression: Vectorized Form

- **Vectorized Formulation:**

$$\hat{y} = \mathbf{x}^\top \boldsymbol{\theta}$$

- **Definitions:**
  - $\mathbf{x} = \begin{bmatrix} 1 & x_1 & x_2 & \cdots & x_n \end{bmatrix}^\top \in \mathbb{R}^{(n+1)}$: feature vector with bias term.
  - $\boldsymbol{\theta} = \begin{bmatrix} \theta_0 & \theta_1 & \theta_2 & \cdots & \theta_n \end{bmatrix}^\top \in \mathbb{R}^{(n+1)}$: parameter vector.
  - $\hat{y} \in \mathbb{R}$: predicted output (scalar).
- **Generalization to m samples:**

$$\hat{\mathbf{y}} = \mathbf{X}\boldsymbol{\theta}, \quad \mathbf{X} \in \mathbb{R}^{m \times (n+1)}, \ \hat{\mathbf{y}} \in \mathbb{R}^m$$

- **Expanded Form:**

$$\mathbf{X} = \begin{bmatrix} 1 & x_1^{(1)} & x_2^{(1)} & \cdots & x_n^{(1)} \\ 1 & x_1^{(2)} & x_2^{(2)} & \cdots & x_n^{(2)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(m)} & x_2^{(m)} & \cdots & x_n^{(m)} \end{bmatrix}, \quad \hat{\mathbf{y}} = \begin{bmatrix} \hat{y}^{(1)} \\ \hat{y}^{(2)} \\ \vdots \\ \hat{y}^{(m)} \end{bmatrix}$$

- **Row Expansion:**

$$\hat{y}^{(i)} = \theta_0 + \theta_1 x_1^{(i)} + \theta_2 x_2^{(i)} + \cdots + \theta_n x_n^{(i)}, \quad i = 1, \ldots, m$$

Example Dataset: House Prices

- **Scenario:** Predict house price (y) using features such as:
  - x_1 = floor area (in m^2)
  - x_2 = number of bedrooms
  - x_3 = age of the house (in years)

House	Floor Area (m^2)	Bedrooms	Age (yrs)	Price (\$1000s)
1	85	2	10	150
2	**120**	**3**	**5**	**230**
3	60	2	20	100
4	200	4	2	400
5	150	3	8	280

- **Model:**

$$y \approx \hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3$$

- **Vector Form for House 2:**

$$\mathbf{x}^{(2)} = \begin{bmatrix} 1 \\ 120 \\ 3 \\ 5 \end{bmatrix}, \quad \hat{y}^{(2)} = \mathbf{x}^{(2)\top} \boldsymbol{\theta}$$

Solving Linear Regression with Least Squares

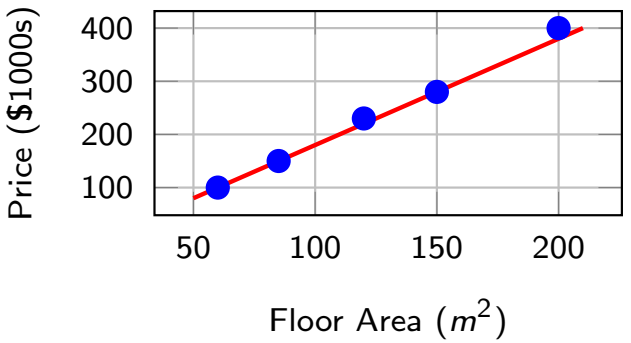
- **Cost Function:**

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(y^{(i)} - \hat{y}^{(i)} \right)^2$$

- **Closed-form solution:**

$$\hat{\theta} = (\boldsymbol{X}^\top \boldsymbol{X})^{-1} \boldsymbol{X}^\top \boldsymbol{y}$$

- Minimizes squared error between predictions $\hat{y}$ and true values y.



Least Squares: Step-by-step Guide

1. **Form the design matrix.** Construct

$$\mathbf{X} = \begin{bmatrix} 1 & x_1^{(1)} & \cdots & x_n^{(1)} \\ 1 & x_1^{(2)} & \cdots & x_n^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(m)} & \cdots & x_n^{(m)} \end{bmatrix} \in \mathbb{R}^{m \times (n+1)},$$

where m is the number of training samples, n is the number of features (excluding bias), and row i is $\mathbf{x}^{(i)\top}$.

2. **Compute normal-system components.**

$$\mathbf{A} = \mathbf{X}^\top \mathbf{X} \in \mathbb{R}^{(n+1)\times(n+1)}, \qquad \mathbf{b} = \mathbf{X}^\top \mathbf{y} \in \mathbb{R}^{(n+1)}.$$

Here $\mathbf{y} \in \mathbb{R}^m$ is the target vector with entries $y^{(i)}$.

3. **Solve the linear system (normal equations).**

$$\mathbf{A}\,\boldsymbol{\theta} = \mathbf{b}.$$

If $\mathbf{A}$ is invertible (i.e. $\mathbf{X}$ has full column rank $n+1$), compute

$$\hat{\boldsymbol{\theta}} = \mathbf{A}^{-1}\mathbf{b} = (\mathbf{X}^\top \mathbf{X})^{-1}\mathbf{X}^\top \mathbf{y},$$

where $\hat{\boldsymbol{\theta}} \in \mathbb{R}^{(n+1)}$ is the estimated parameter vector (including θ_0).

4. **Computational costs (rough).** forming $\mathbf{X}^\top \mathbf{X}$: $O\big(m(n+1)^2\big)$; solving $\mathbf{A}\boldsymbol{\theta} = \mathbf{b}$ by direct methods: $O\big((n+1)^3\big)$.

Least Squares: Derivation of Normal Equations

Cost function (least squares):

$$J(\boldsymbol{\theta}) \;=\; \frac{1}{2m}\,\|\mathbf{X}\boldsymbol{\theta} - \mathbf{y}\|_2^2$$

where $\mathbf{X} \in \mathbb{R}^{m\times(n+1)}$ is the design matrix, $\boldsymbol{\theta} \in \mathbb{R}^{(n+1)}$ the parameter vector, $\mathbf{y} \in \mathbb{R}^m$ the target vector, and m the number of samples.

Expand the quadratic:

$$J(\boldsymbol{\theta}) = \frac{1}{2m}\left(\boldsymbol{\theta}^\top \mathbf{X}^\top \mathbf{X}\,\boldsymbol{\theta} - 2\,\boldsymbol{\theta}^\top \mathbf{X}^\top \mathbf{y} + \mathbf{y}^\top \mathbf{y}\right).$$

Each symbol as above; note $\mathbf{X}^\top \mathbf{X} \in \mathbb{R}^{(n+1)\times(n+1)}$.

Gradient w.r.t. $\boldsymbol{\theta}$:

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \;=\; \frac{1}{m}\left(\mathbf{X}^\top \mathbf{X}\,\boldsymbol{\theta} - \mathbf{X}^\top \mathbf{y}\right).$$

(Standard matrix calculus: derivative of $\frac{1}{2}\boldsymbol{\theta}^\top \mathbf{M}\boldsymbol{\theta}$ is $\mathbf{M}\boldsymbol{\theta}$ for symmetric $\mathbf{M}$.)

Set gradient to zero (first-order optimality):

$$\mathbf{X}^\top \mathbf{X}\,\hat{\boldsymbol{\theta}} \;=\; \mathbf{X}^\top \mathbf{y}.$$

This is the *normal equations*. Symbols: $\mathbf{X}^\top \mathbf{X} \in \mathbb{R}^{(n+1)\times(n+1)}$, $\mathbf{X}^\top \mathbf{y} \in \mathbb{R}^{(n+1)}$.

Hessian and convexity:

$$\nabla^2_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \;=\; \frac{1}{m}\,\mathbf{X}^\top \mathbf{X},$$

which is positive semidefinite. If $\mathbf{X}$ has full column rank ($\operatorname{rank}(\mathbf{X}) = n+1$) then $\mathbf{X}^\top \mathbf{X}$ is positive definite and the solution is unique.

Closed-form solution (when invertible):

$$\boxed{\hat{\boldsymbol{\theta}} \;=\; (\mathbf{X}^\top \mathbf{X})^{-1}\mathbf{X}^\top \mathbf{y}}$$

Least Squares: Model

Linear regression model (scalar form):

$$\hat{y}^{(i)} = \theta_0 + \sum_{j=1}^{n} \theta_j x_j^{(i)}, \qquad i = 1, \ldots, m,$$

where

- m = number of training samples,
- n = number of features (excluding bias),
- $x_j^{(i)}$ = j-th feature of sample i,
- θ_0 = intercept, θ_j = parameters,
- $\hat{y}^{(i)}$ = predicted output.

Least Squares: Cost Function

Least squares objective:

$$J(\boldsymbol{\theta}) \;=\; \frac{1}{2m}\sum_{i=1}^{m}\left(y^{(i)} - \hat{y}^{(i)}\right)^2$$

Substitute model:

$$J(\boldsymbol{\theta}) \;=\; \frac{1}{2m}\sum_{i=1}^{m}\left(y^{(i)} - \theta_0 - \sum_{j=1}^{n}\theta_j x_j^{(i)}\right)^2,$$

where $y^{(i)}$ is the observed target.

Derivative with respect to θ_0

$$\frac{\partial J}{\partial \theta_0} = -\frac{1}{m} \sum_{i=1}^{m} \left(y^{(i)} - \theta_0 - \sum_{j=1}^{n} \theta_j x_j^{(i)} \right).$$

Set to zero (optimality condition):

$$\sum_{i=1}^{m} y^{(i)} = m\theta_0 + \sum_{j=1}^{n} \theta_j \sum_{i=1}^{m} x_j^{(i)}.$$

Define means:

$$\bar{y} = \frac{1}{m} \sum_{i=1}^{m} y^{(i)}, \quad \bar{x}_j = \frac{1}{m} \sum_{i=1}^{m} x_j^{(i)}.$$

So:

$$\theta_0 = \bar{y} - \sum_{j=1}^{n} \theta_j \bar{x}_j.$$

Derivative with respect to θ_k

For $k = 1, \ldots, n$:

$$\frac{\partial J}{\partial \theta_k} = -\frac{1}{m} \sum_{i=1}^{m} x_k^{(i)} \left(y^{(i)} - \theta_0 - \sum_{j=1}^{n} \theta_j x_j^{(i)} \right).$$

Set to zero:

$$\sum_{i=1}^{m} x_k^{(i)} y^{(i)} = \theta_0 \sum_{i=1}^{m} x_k^{(i)} + \sum_{j=1}^{n} \theta_j \sum_{i=1}^{m} x_k^{(i)} x_j^{(i)}.$$

These n equations plus the θ_0 equation form a linear system for the unknowns $\theta_0, \ldots, \theta_n$.

Special Case: Simple Linear Regression ($n = 1$)

Model:

$$\hat{y}^{(i)} = \theta_0 + \theta_1 x^{(i)}, \quad i = 1, \ldots, m.$$

Solution:

$$\theta_1 = \frac{\sum_{i=1}^{m}(x^{(i)} - \bar{x})(y^{(i)} - \bar{y})}{\sum_{i=1}^{m}(x^{(i)} - \bar{x})^2},$$

$$\theta_0 = \bar{y} - \theta_1 \bar{x},$$

where $\bar{x} = \frac{1}{m}\sum_i x^{(i)}$, $\bar{y} = \frac{1}{m}\sum_i y^{(i)}$.

Worked Example: House Prices vs Floor Area

Dataset ($m = 5$, $n = 1$):

House No.	Floor Area (x_1)	Bedrooms (x_2)	Age (x_3)	Price (y)
1	85	2	10	200
2	120	3	5	250
3	60	2	20	180
4	200	4	8	300
5	150	3	15	220

Worked Example: Step-by-Step Solution

Step 1: Means

$$\bar{x} = \frac{1}{m}\sum_{i=1}^{m} x^{(i)}, \quad \bar{y} = \frac{1}{m}\sum_{i=1}^{m} y^{(i)}$$

For $m = 5$:

$$\bar{x} = \frac{85+120+60+200+150}{5} = 123, \quad \bar{y} = \frac{200+250+180+300+220}{5} = 230$$

Step 2: Slope

$$\theta_1 = \frac{\sum_{i=1}^{m}(x^{(i)} - \bar{x})(y^{(i)} - \bar{y})}{\sum_{i=1}^{m}(x^{(i)} - \bar{x})^2}$$

$$\theta_1 = \frac{(85-123)(200-230) + \cdots + (150-123)(220-230)}{(85-123)^2 + \cdots + (150-123)^2} \approx 0.74$$

Step 3: Intercept

$$\theta_0 = \bar{y} - \theta_1 \bar{x} = 230 - (0.74)(123) \approx 139.0$$

$$\therefore \hat{y} = 139.0 + 0.74x$$

Worked Example: Scatterplot with Residuals

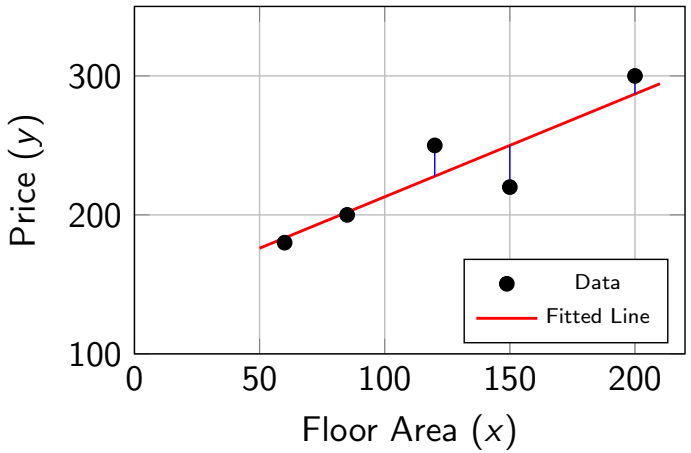
Fitted Line:

$$\hat{y} = 139.0 + 0.74\,x$$

Residuals:

$$r^{(i)} = y^{(i)} - \hat{y}^{(i)}$$

Residuals are larger here because the correlation is weaker.



Gradient Descent (Scalar Form): Model

Linear model (scalar form):

$$\hat{y}^{(i)} = \theta_0 + \sum_{j=1}^{n} \theta_j x_j^{(i)}, \qquad i = 1, \ldots, m.$$

where

- m is the number of training samples,
- n is the number of features (excluding bias),
- $x_j^{(i)}$ is the j-th feature of sample i,
- θ_0 is the intercept (bias), θ_j are parameters,
- $\hat{y}^{(i)}$ is the predicted output for sample i.

Cost Function (Least Squares) – Scalar

Least-squares objective (scalar summation form):

$$J(\boldsymbol{\theta}) = \frac{1}{2m}\sum_{i=1}^{m}\left(y^{(i)} - \hat{y}^{(i)}\right)^2 = \frac{1}{2m}\sum_{i=1}^{m}\left(y^{(i)} - \theta_0 - \sum_{j=1}^{n}\theta_j x_j^{(i)}\right)^2.$$

where

- $y^{(i)}$ is the observed (true) target for sample i,
- $\hat{y}^{(i)}$ defined in previous frame,
- $J(\boldsymbol{\theta})$ is the scalar cost to minimize over $\boldsymbol{\theta} \in \mathbb{R}^{n+1}$.

Gradient: derivative w.r.t. θ_0 (detailed)

Compute partial derivative of J with respect to θ_0 using the chain rule:

$$\frac{\partial J}{\partial \theta_0} = \frac{\partial}{\partial \theta_0}\left(\frac{1}{2m}\sum_{i=1}^{m}\left(y^{(i)} - \theta_0 - \sum_{j=1}^{n}\theta_j x_j^{(i)}\right)^2\right)$$

$$= -\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)} - \theta_0 - \sum_{j=1}^{n}\theta_j x_j^{(i)}\right).$$

where

- the derivative used: $\frac{d}{du}\frac{1}{2}u^2 = u$,
- the inner residual for sample i is $r^{(i)} = y^{(i)} - \hat{y}^{(i)}$.

Set to zero (normal-equation form for bias):

$$\sum_{i=1}^{m} y^{(i)} = m\theta_0 + \sum_{j=1}^{n}\theta_j \sum_{i=1}^{m} x_j^{(i)}.$$

where sums are as defined above.

Gradient: derivative w.r.t. θ_k (general k)

For $k \in \{1, \ldots, n\}$ apply the chain rule:

$$\frac{\partial J}{\partial \theta_k} = -\frac{1}{m} \sum_{i=1}^{m} x_k^{(i)} \left(y^{(i)} - \theta_0 - \sum_{j=1}^{n} \theta_j x_j^{(i)} \right).$$

where

- $x_k^{(i)}$ multiplies the residual because $\partial(\theta_j x_j^{(i)})/\partial \theta_k = x_k^{(i)}$,
- the term is the scalar inner product between feature k and residuals.

Equivalently, define per-sample residual $r^{(i)} = y^{(i)} - \hat{y}^{(i)}$, then

$$\frac{\partial J}{\partial \theta_k} = -\frac{1}{m} \sum_{i=1}^{m} x_k^{(i)} \, r^{(i)}.$$

Gradient Descent: Coordinate-wise Update

Batch gradient descent update (scalar coordinate form):

$$\theta_k \leftarrow \theta_k - \alpha \frac{\partial J}{\partial \theta_k}, \qquad k = 0, 1, \ldots, n.$$

where

- $\alpha > 0$ is the learning rate (step size),
- $\frac{\partial J}{\partial \theta_k} = -\frac{1}{m} \sum_{i=1}^{m} x_k^{(i)} \, r^{(i)}$

Substitute the derivative to obtain the explicit update:

$$\boxed{\theta_k \leftarrow \theta_k + \frac{\alpha}{m} \sum_{i=1}^{m} x_k^{(i)} \left(y^{(i)} - \hat{y}^{(i)} \right)}$$

where $x_0^{(i)} \equiv 1$ for the bias (so the update for θ_0 uses $x_0^{(i)} = 1$).

Algorithm: Batch Gradient Descent (Scalar Pseudocode)

1. **Inputs:** dataset $\{(x_{1:n}^{(i)}, y^{(i)})\}_{i=1}^{m}$, learning rate α, max iterations T.
2. **Initialize:** $\theta_k^{(0)} = 0$ (or small random) for $k = 0, \ldots, n$.
3. **For** $t = 0, 1, \ldots, T-1$ do:
    1. Compute predictions $\hat{y}^{(i)} = \theta_0^{(t)} + \sum_{j=1}^{n} \theta_j^{(t)} x_j^{(i)}$ for all i.
    2. For each $k = 0, \ldots, n$ compute gradient:

$$g_k^{(t)} \;=\; -\frac{1}{m} \sum_{i=1}^{m} x_k^{(i)} \big(y^{(i)} - \hat{y}^{(i)}\big).$$

    3. Update:

$$\theta_k^{(t+1)} \;=\; \theta_k^{(t)} - \alpha\, g_k^{(t)}.$$

4. **Return** $\theta^{(T)}$.

Definitions: $g_k^{(t)}$ is the partial derivative at iteration t; $x_0^{(i)} = 1$.

Variants: SGD and Mini-batch (Scalar)

Stochastic Gradient Descent (SGD) — per-sample update:

$$\theta_k \leftarrow \theta_k + \alpha\, x_k^{(i)} \big(y^{(i)} - \hat{y}^{(i)}\big),$$

where

- the update uses one sample i (or randomly sampled i),
- note: this expression omits the $1/m$ factor (conventional SGD uses per-sample learning rate).

Mini-batch of indices B:

$$\theta_k \leftarrow \theta_k + \frac{\alpha}{|B|} \sum_{i \in B} x_k^{(i)} \big(y^{(i)} - \hat{y}^{(i)}\big),$$

where $|B|$ is batch size. Definitions: $x_0^{(i)} = 1$.

Practical Considerations (scalar viewpoint)

- **Feature scaling:** Standardize each feature x_j (zero mean, unit std) so typical $\sum_i (x_j^{(i)})^2$ are comparable; this reduces $\lambda_{\max}$ and improves conditioning of H. Definitions: standardize via $\tilde{x}_j^{(i)} = (x_j^{(i)} - \bar{x}_j)/s_j$.
- **Bias update:** If features are centered then θ_0 update simplifies to $\theta_0 \leftarrow \theta_0 + \frac{\alpha}{m} \sum_i r^{(i)}$ and decouples from other θ_j updates.
- **Initialization:** Use zeros or small random values for θ_k.
- **Stopping criteria:** small $\|\Delta\theta\|$, small relative change in J, or fixed iterations.
- **Regularization (ridge) in scalar form:** add $\frac{\lambda}{2m} \sum_{j=1}^{n} \theta_j^2$ (do not regularize θ_0 typically). Then gradient becomes:

$$\frac{\partial J_{\text{ridge}}}{\partial \theta_k} = -\frac{1}{m} \sum_{i=1}^{m} x_k^{(i)} r^{(i)} + \frac{\lambda}{m} \theta_k \quad (k \geq 1).$$

Computational Cost (scalar) & Summary

Per-iteration cost (batch GD):

- computing all predictions $\hat{y}^{(i)}$: $O(m(n+1))$ operations,
- computing gradients g_k: additional $O(m(n+1))$,
- total per iteration: $O(mn)$ (dominant).

Summary (scalar form):

- Model: $\hat{y}^{(i)} = \theta_0 + \sum_{j=1}^{n} \theta_j x_j^{(i)}$.
- Gradient (component): $\frac{\partial J}{\partial \theta_k} = -\frac{1}{m} \sum_{i=1}^{m} x_k^{(i)} r^{(i)}$.
- Update: $\theta_k \leftarrow \theta_k + \frac{\alpha}{m} \sum_{i=1}^{m} x_k^{(i)} r^{(i)}$.
- Convergence: choose α with $0 < \alpha < 2/\lambda_{\max}(H)$, where $H_{k\ell} = \frac{1}{m} \sum_i x_k^{(i)} x_\ell^{(i)}$.