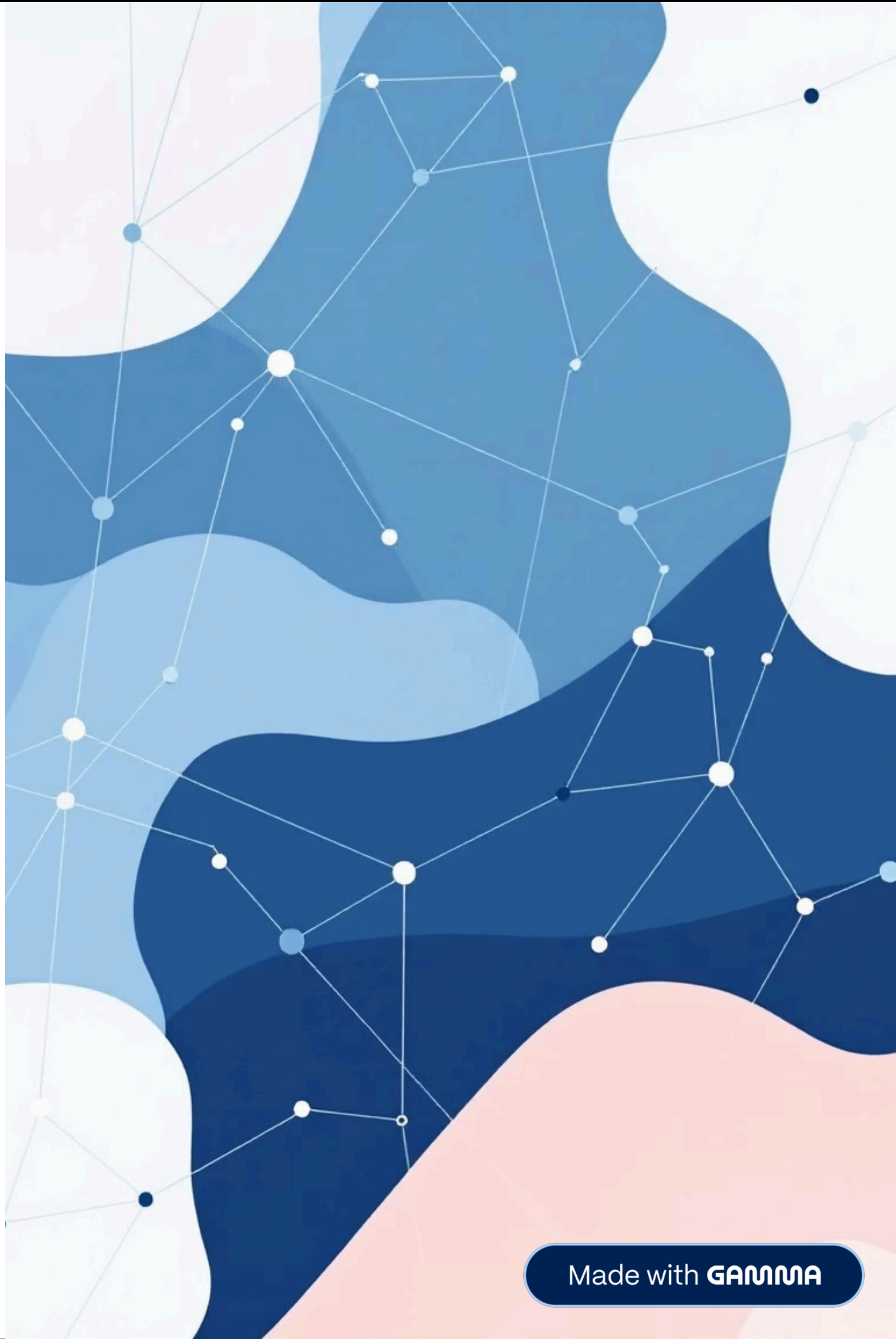
Introduction to Machine Learning Regression

Teaching computers to predict continuous values through pattern recognition



Core Concept: Predicting Continuous Values

Problem Statement

Given: house size x

Predict: price y

Goal: learn $y = f(x)$

Input Feature

Size in m^2 → x

Target Variable

Price in $ → y

Model Output

Predicted value $\hat{y}$

Linear Model: Line of Best Fit

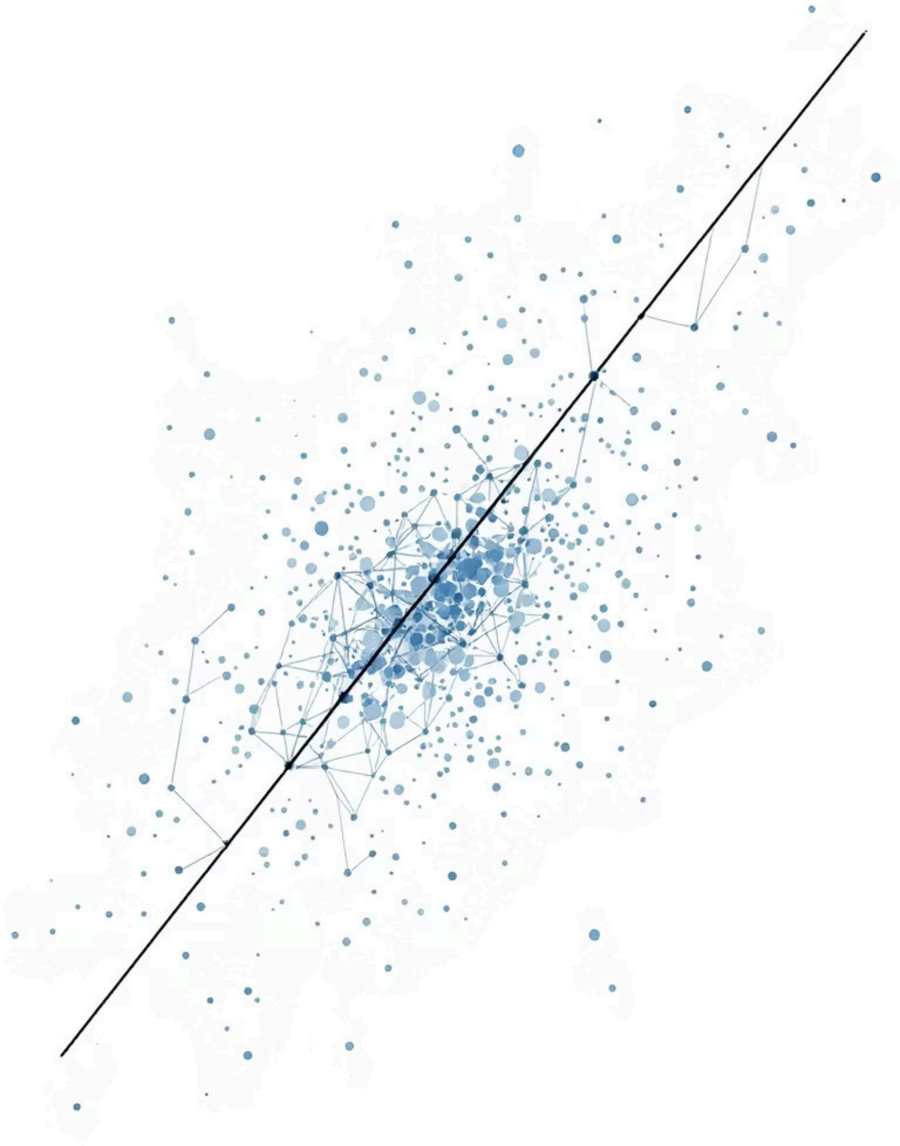
Model Equation

$$\hat{y} = w_1 x + w_0$$

Parameters

w_1: slope (rate of change)

w_0: intercept (baseline)



Matrix Representation

Design Matrix

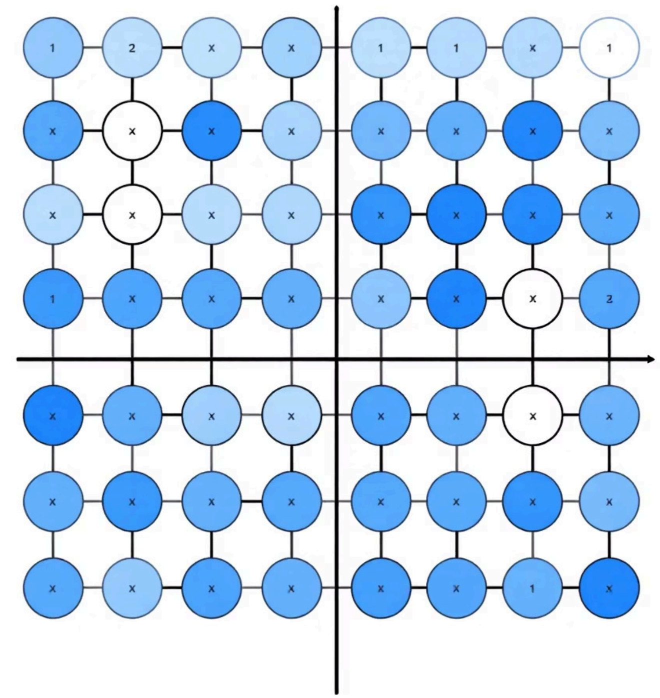
$$\mathbf{X} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix}$$

Weight Vector

$$\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix}$$

Prediction

$$\hat{\mathbf{y}} = \mathbf{X}\mathbf{w}$$



Matrix form enables efficient computation across all data points simultaneously

Matrix Representation: Scaling to Multiple Features

Real-world predictions often use **multiple features** (e.g., house size, age, rooms). Individual equations become unwieldy. Matrix notation offers a compact way to represent features, weights, and data for efficient computation.

Design Matrix ($\mathbf{X}$)

Rows are data points, columns are features. An initial column of **ones** accounts for the intercept (w_0).

Weight Vector ($\mathbf{w}$)

This vector holds all model parameters: the intercept weight (w_0) and weights for each feature ($w_1, \ldots, w_n$).

$$\mathbf{X} = \begin{bmatrix} 1 & x_{11} & \cdots & x_{1n} \\ 1 & x_{21} & \cdots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{m1} & \cdots & x_{mn} \end{bmatrix}$$

$$\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_n \end{bmatrix}$$

Predictions

Matrix multiplication yields all predictions simultaneously:

$$\hat{\mathbf{y}} = \mathbf{X}\mathbf{w}$$

Example Calculation

$$\begin{bmatrix} 1 & 2 \\ 1 & 3 \\ 1 & 4 \end{bmatrix} \begin{bmatrix} 0.5 \\ 2 \end{bmatrix} = \begin{bmatrix} (1 \cdot 0.5) + (2 \cdot 2) \\ (1 \cdot 0.5) + (3 \cdot 2) \\ (1 \cdot 0.5) + (4 \cdot 2) \end{bmatrix} = \begin{bmatrix} 0.5 + 4 \\ 0.5 + 6 \\ 0.5 + 8 \end{bmatrix} = \begin{bmatrix} 4.5 \\ 6.5 \\ 8.5 \end{bmatrix}$$

This compact form is fundamental for scaling linear regression to large datasets and numerous features, underpinning most modern machine learning libraries.

Loss Function: Mean Squared Error

Objective

Minimize prediction error

MSE Formula

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Matrix Form

$$\text{MSE} = \frac{1}{n}(\mathbf{y} - \mathbf{X}\mathbf{w})^T(\mathbf{y} - \mathbf{X}\mathbf{w})$$

Squaring penalizes large deviations; averaging normalizes across dataset size

Analytical Solution: Normal Equation

Optimization Problem

$$\min_{\mathbf{w}} \frac{1}{n}(\mathbf{y} - \mathbf{X}\mathbf{w})^T(\mathbf{y} - \mathbf{X}\mathbf{w})$$

Set Derivative to Zero

$$\frac{\partial \text{MSE}}{\partial \mathbf{w}} = 0$$

Closed-Form Solution

$$\mathbf{w} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

Direct computation; computationally expensive for large n due to matrix inversion

Iterative Optimization: Gradient Descent

Initialize

$w_0, w_1 \leftarrow$ random values

Predict

$\hat{y}_i = w_1 x_i + w_0$

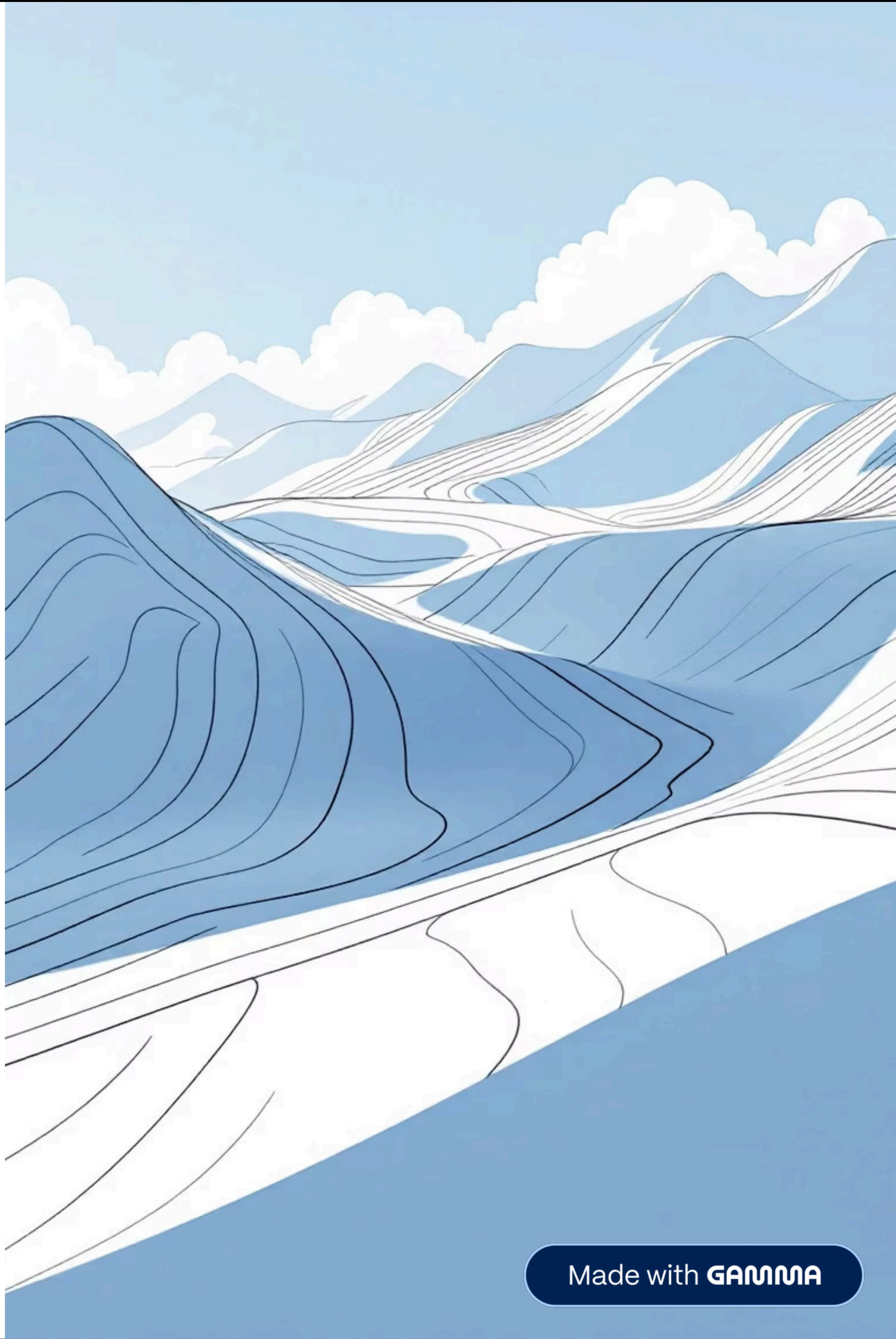
Compute Loss

Calculate MSE

Update Weights

$$w_j := w_j - \alpha \frac{\partial \mathrm{MSE}}{\partial w_j}$$

α: learning rate (step size)



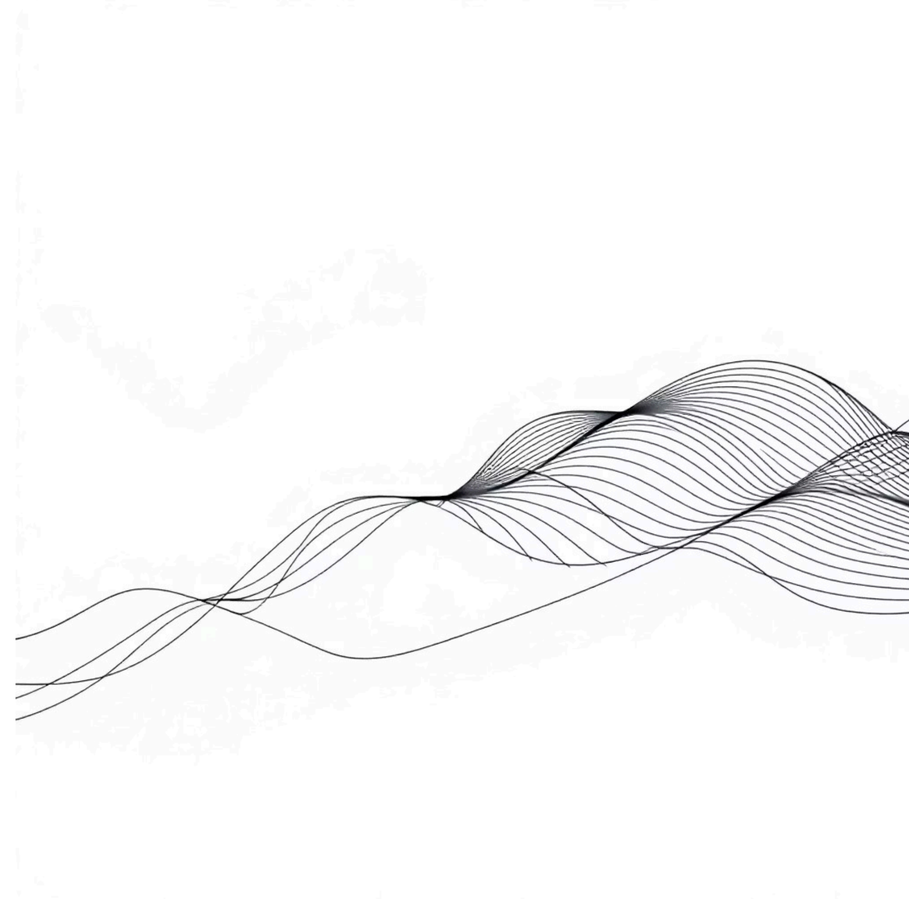
Gradient Computation

Partial Derivative

$$\frac{\partial \text{MSE}}{\partial w_j} = -\frac{2}{n}\sum_{i=1}^{n} x_{ij}(y_i - \hat{y}_i)$$

Vectorized Update Rule

$$\mathbf{w} := \mathbf{w} + \frac{2\alpha}{n}\mathbf{X}^T(\mathbf{y} - \mathbf{X}\mathbf{w})$$



Foundation for training all neural networks and deep learning models

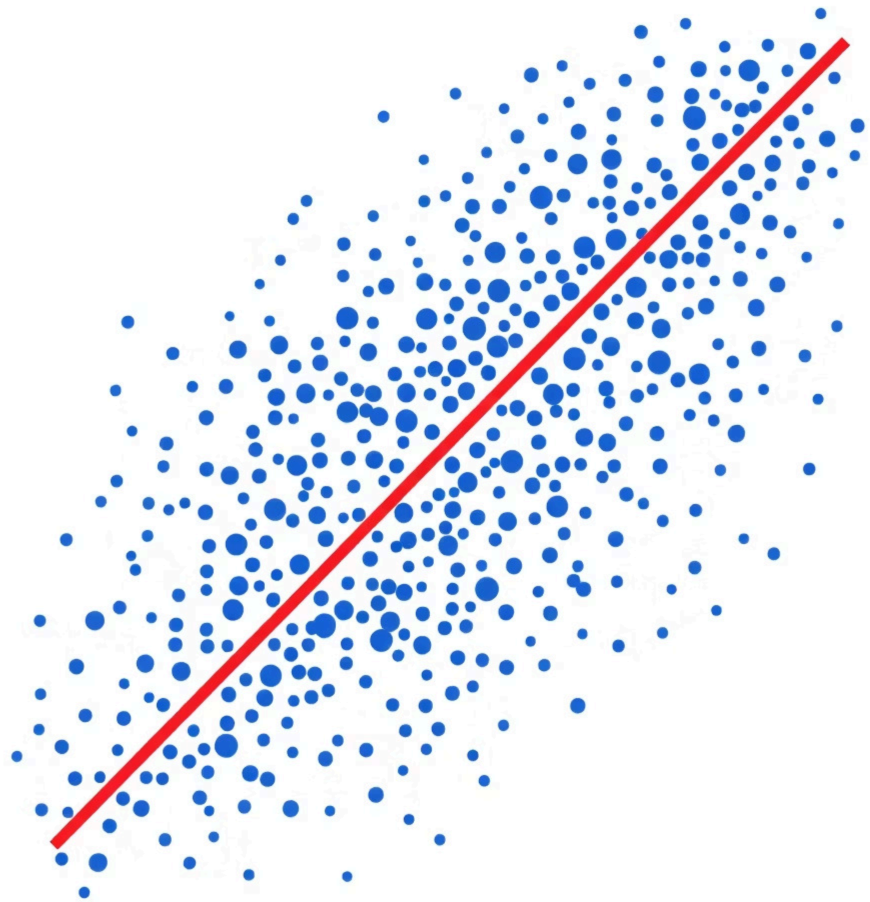
Implementation Example

Python Code

```
from sklearn.linear_model import LinearRegression

X = [[1],[2],[3],[4],[5]]
y = [2.1,4.1,5.9,8.2,10.1]

model = LinearRegression().fit(X,y)
print(model.coef_, model.intercept_)
```



Training data points

Fitted regression line

Key Takeaways

01

Model Form

$\hat{y} = \mathbf{X}\mathbf{w}$

02

Training Objective

Minimize MSE loss function

03

Analytical Method

Normal Equation (exact)

04

Iterative Method

Gradient Descent (scalable)

05

Core ML Principle

Learn by minimizing loss