KAUST Academy & Tech Camp AI Week

Presented By: Ali Alqutayfi & Hassain Alsayhah

Linear Regression

Logistic Regression

Neural Networks

Deep Learning

Artificial Intelligence and Machine Learning

Linear Regression

Lecture 1: Outline

- What is Machine Learning?
- Linear Regression
- Loss Function (Mean Squared Error)
- How Do We Minimize MSE?
- Evaluating Our Model
- Regularization

What is Machine Learning?

Traditional Programming:

- Study problem → Write rules → Evaluate → Launch

Machine Learning:

- Study problem → Train algorithm with data → Evaluate → Launch

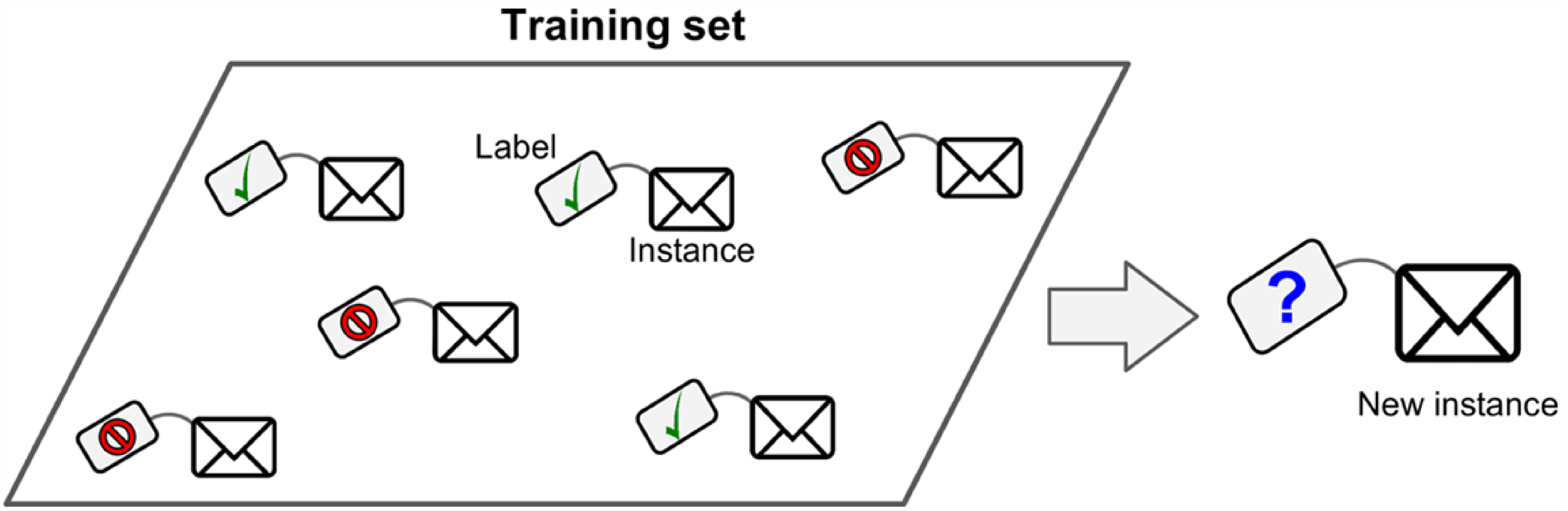
Key difference: Instead of writing rules manually, we let the algorithm learn patterns from data.

ML Algorithms Types

- **Supervised:** The algorithm l
  - **Regression:** Predict continuous value (e.g. house prices).
  - **Classification:** Predict discrete value (e.g. spam/not-spam).

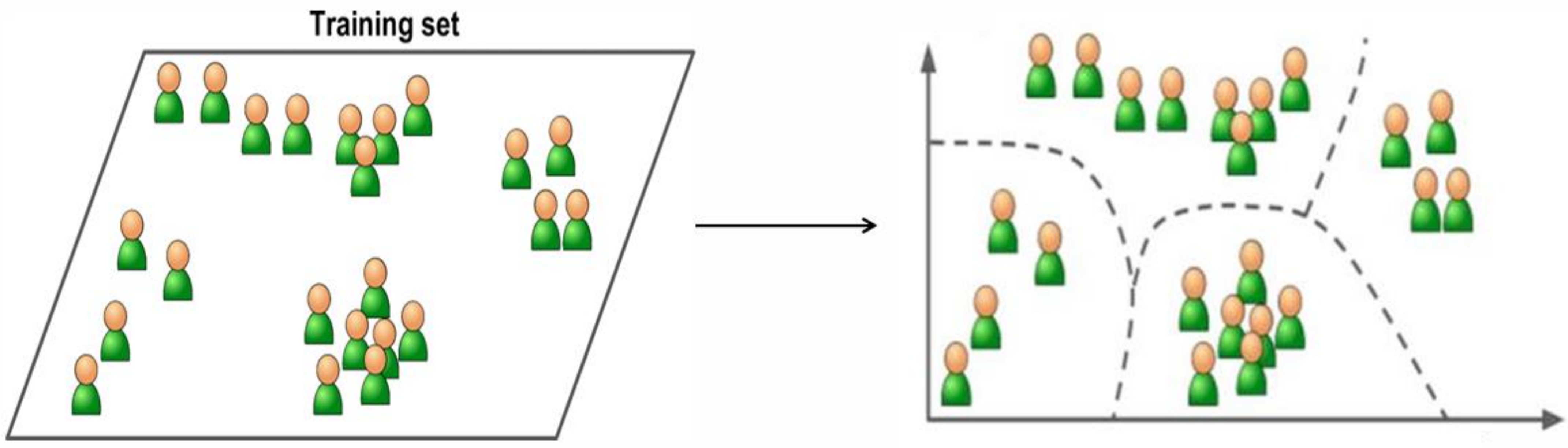
- **Unsupervised:** The algorithm works on unlabeled data. We are interested in things like:
  - **Clustering:** Grouping

ML Algorithms Types



An example of Supervised Learning: Spam Classification

ML Algorithms Types



An example of Unsupervised Learning: Clustering

Linear Regression - The Goal

Problem: Given input data (x), predict a continuous output (y)

Examples:

- House size → House price
- Years of experience → Salary
- Temperature → Ice cream sales

Our goal: Find the best line that fits through our data points

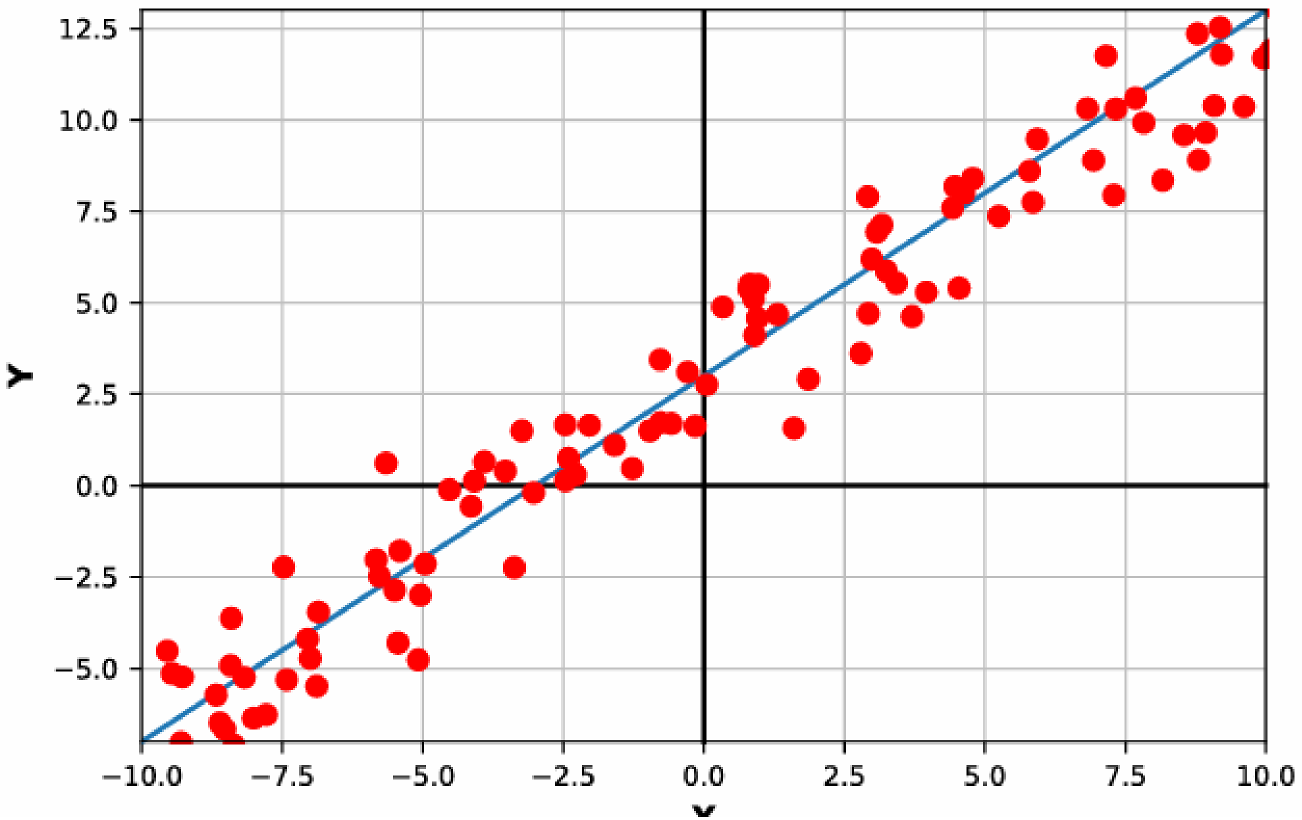
The Linear Regression Equation

$$y = mx + b$$

- **y**: What we want to predict (target)
- **x**: Input feature
- **m**: Slope (how much y changes when x changes)
- **b**: Intercept (y-value when x = 0)

In ML notation: $\hat{y} = \theta_0 + \theta_1 x$

Finding the Best Line



Finding the Best Line

Question: How do we find the best values for m and b?

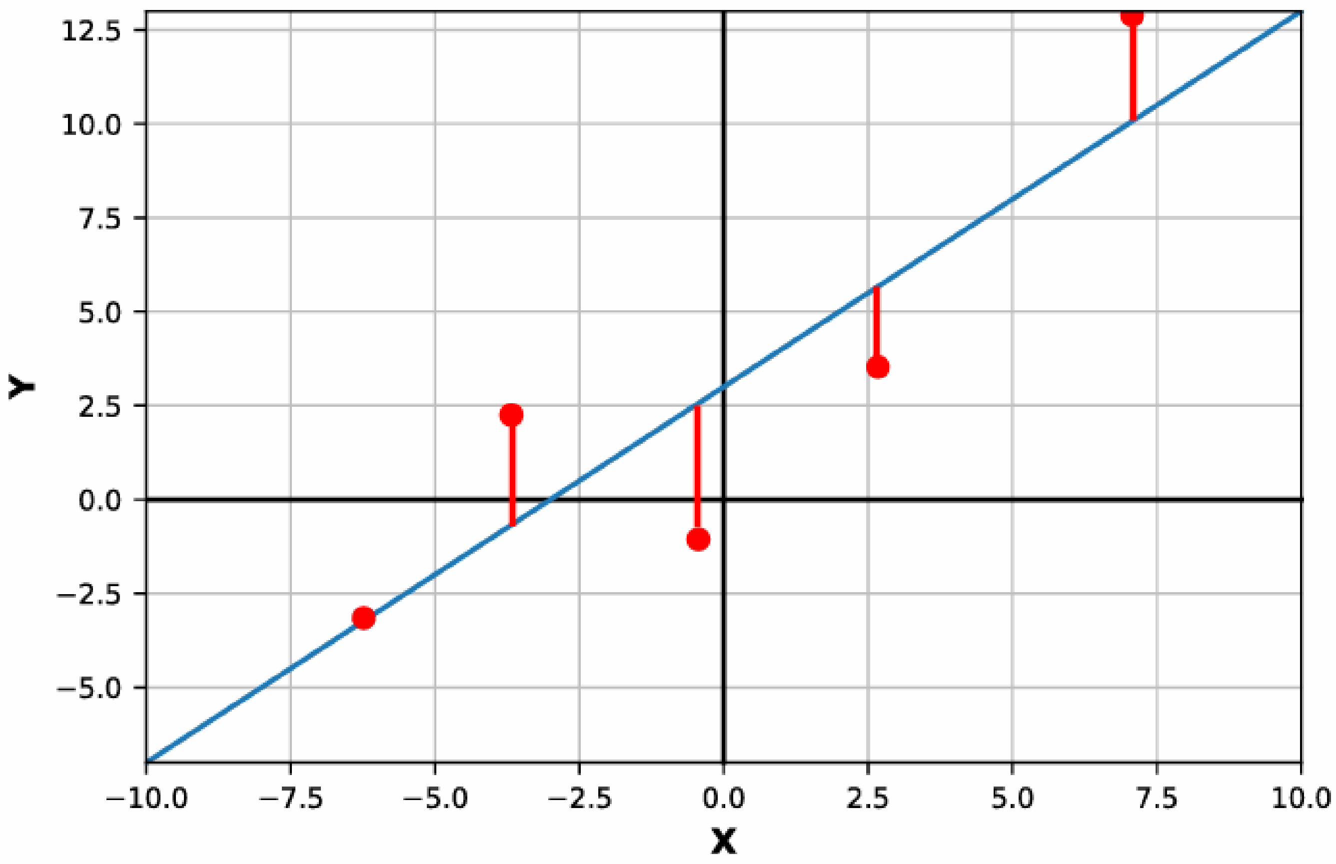
Answer: Minimize the prediction errors!

Error for one point: error = actual_y - predicted_y

Problem: Errors can be positive or negative, they cancel out!

Solution: Square the errors: $\text{error}^2 = (\text{actual\_y} - \text{predicted\_y})^2$

Finding the Best Line



Loss Function (Mean Squared Error)

$$MSE = (1/n) \times \Sigma(y_i - \hat{y}_i)^2$$

- Sum up all squared errors
- Divide by number of data points
- This gives us average squared error

Goal: Find m and b that minimize MSE

How Do We Minimize MSE?

Two approaches:

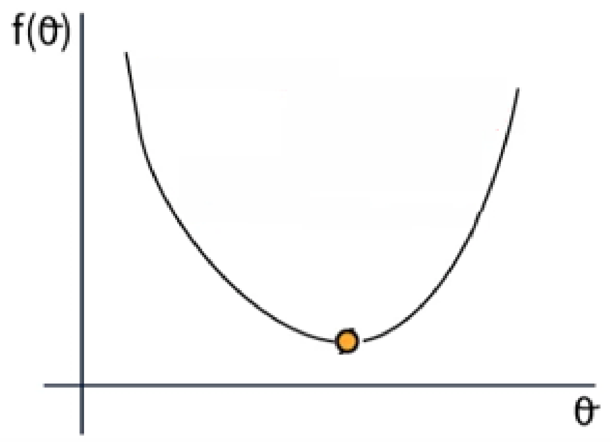
- **Closed-form solution** (for simple linear regression):

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

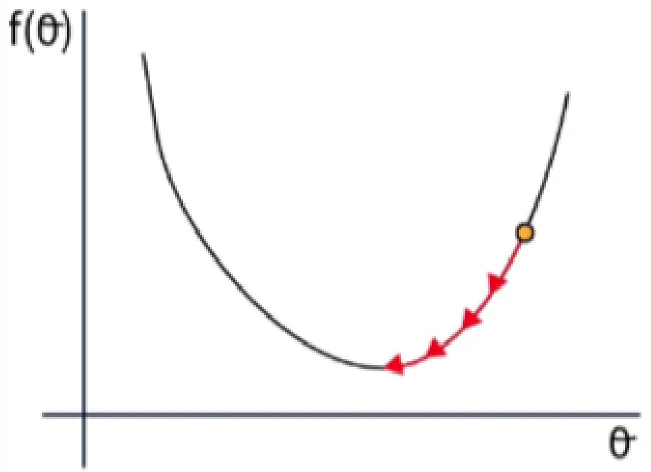
- **Gradient Descent** (iterative approach):
  - Start with random m, b
  - Calculate error
  - Adjust m, b to reduce error
  - Repeat until convergence

How Do We Minimize MSE?

Closed-form:



Iterative:



Evaluating Our Model - R^2

R-squared (R^2): How well does our line explain the data?

R^2 = 1 - (Sum of Squared Residuals)/(Total Sum of Squares)

Interpretation:

- $R^2 = 1.0$: Perfect fit (100% of variance explained)
- $R^2 = 0.0$: No better than guessing the average
- $R^2 = 0.8$: 80% of variance explained (pretty good!)

Train/Test Split - Avoiding Cheating

Problem: How do we know if our model works on new data?

Solution: Split data into:

- **Training set (80%):** Use to learn m and b
- **Test set (20%):** Use to evaluate final performance

Rule: Never let your model see the test data during training!

The Overfitting Problem

Simple model (just x): May be too simple (underfitting) **Complex model (x, x^2, x^3, x^4...):** May memorize training data (overfitting)

Overfitting signs:

- Perfect performance on training data
- Poor performance on test data
- Model learned noise, not patterns

Regularization - Preventing Overfitting

Idea: Penalize complex models

Ridge Regression: Add penalty for large coefficients

Loss = MSE + λ × (sum of squared coefficients)

LASSO Regression: Can set some coefficients to exactly zero

Loss = MSE + λ × (sum of absolute coefficients)

λ (lambda): Controls how much we penalize complexity

Multiple Features

Real world: Usually have multiple input features

$y = \theta_0 + \theta_1 \times \text{feature1} + \theta_2 \times \text{feature2} + \theta_3 \times \text{feature3} + \ldots$

Example - House Prices:

$\text{price} = \theta_0 + \theta_1 \times \text{size} + \theta_2 \times \text{bedrooms} + \theta_3 \times \text{age} + \theta_4 \times \text{location\_score}$

Same process: Find θ values that minimize MSE

Ready to Code!

What we've learned:

- Linear regression finds the best line through data
- We minimize Mean Squared Error (MSE)
- We evaluate using R^2 and train/test split
- We prevent overfitting with regularization
- We can handle multiple features

Ready to Code!

Next: Let's implement this in Python!

Tools we'll use:

- NumPy (for math)
- Pandas (for data)
- Scikit-learn (for models)
- Matplotlib (for visualization)

Exercise

Salary Dataset - Simple linear regression

Your task is to visit the link below and create a simple linear regression to **predict the salary** of employees based on the **years of experience** they have.

Link: https://www.kaggle.com/datasets/abhishek14398/salary-dataset-simple-linear-regression/data