CS-466/566: Math for AI

Module 03: Linear Regression

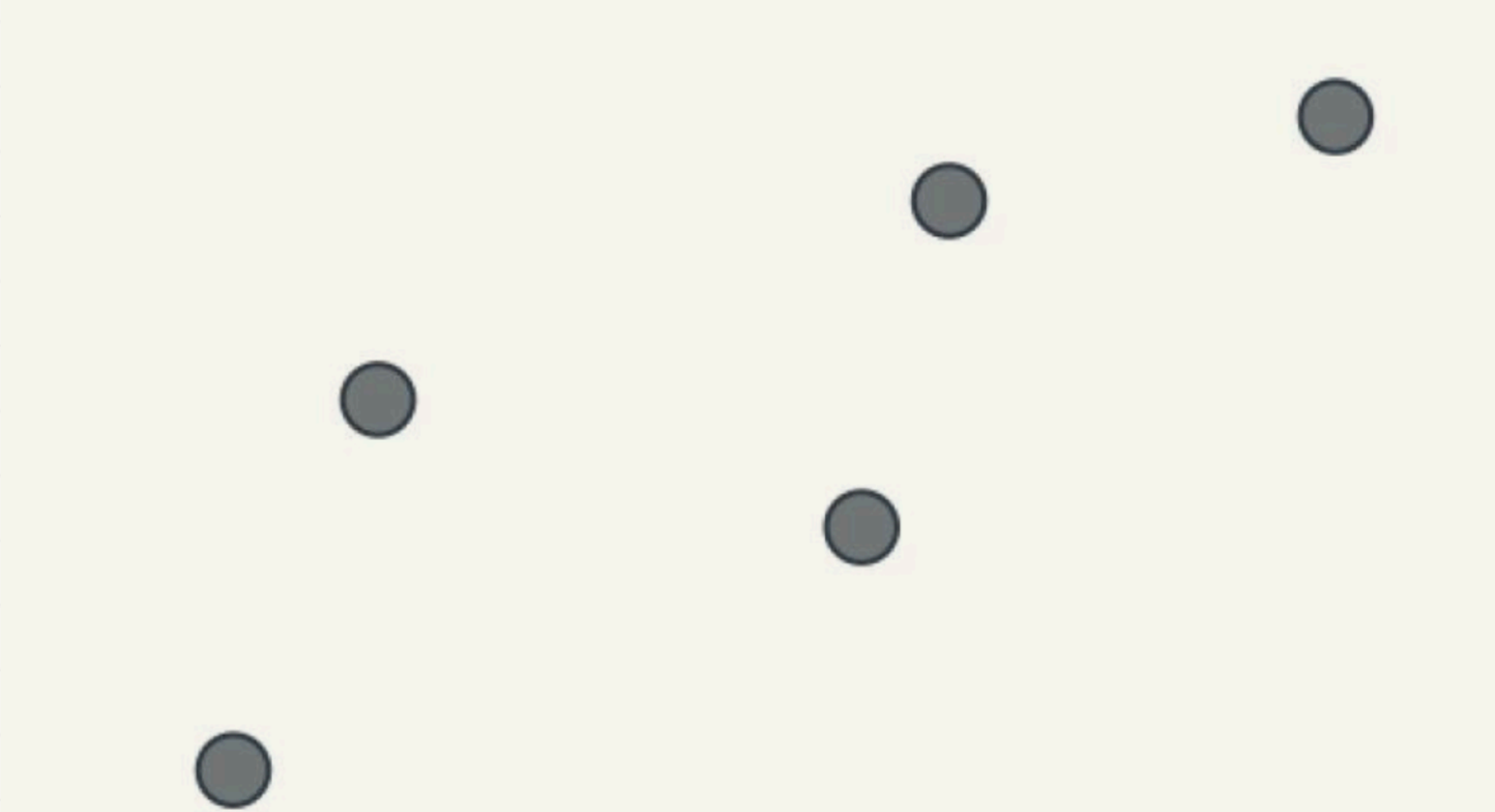
Dr. Mahmoud Mahmoud
The University of Alabama

2026-02-25

TABLE OF CONTENTS

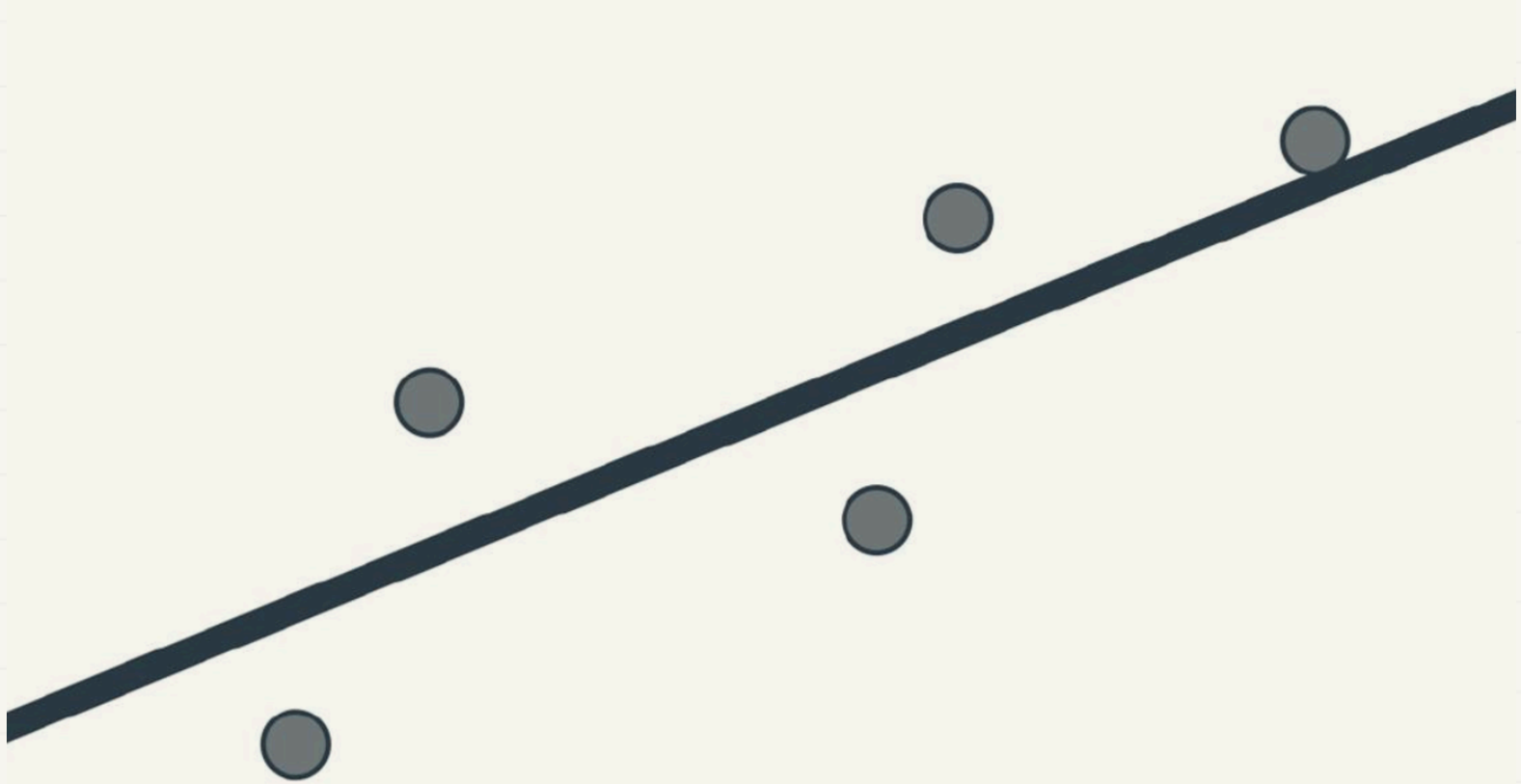
Linear Regression

Can you design a **linear road** that pass by all these houses equally?!



Linear Regression

Can you design a **linear road** that pass by all these houses equally?!



Housing Prices Problem

- What are the features and labels here?
- Is it classification or regression problem?
- What is the price of the house of 4 rooms?
- What is the price per extra room? **(#Room × weight)**
- What is the base price? **(Bias)**
- **What is the equation that represents the price?**
- Price = 100 + 50 * (#Rooms)

Number of rooms	Price
1	150
2	200
3	250
4	?
5	350
6	400
7	450

TABLE OF CONTENTS

Weights and Bias

Price Equation

Price = 100 + 50 * (#Rooms)

WEIGHTS

Each feature gets multiplied by a corresponding factor. These factors are the **weights**. In the above formula the only feature is the number of rooms, and its value is **50**.

BIAS

Constant that is not attached to any of the features. It is called the **bias**. In this model, the bias is **100** and it corresponds to the base price of a house.

How do machines learn this equation?

How machines learn it? [Remember-Formulate-Predict]

Number of rooms	Price
1	150
2	200
3	250
4	?
5	350
6	400
7	450

How machines learn it? [Remember-Formulate-Predict]

Linear Regression

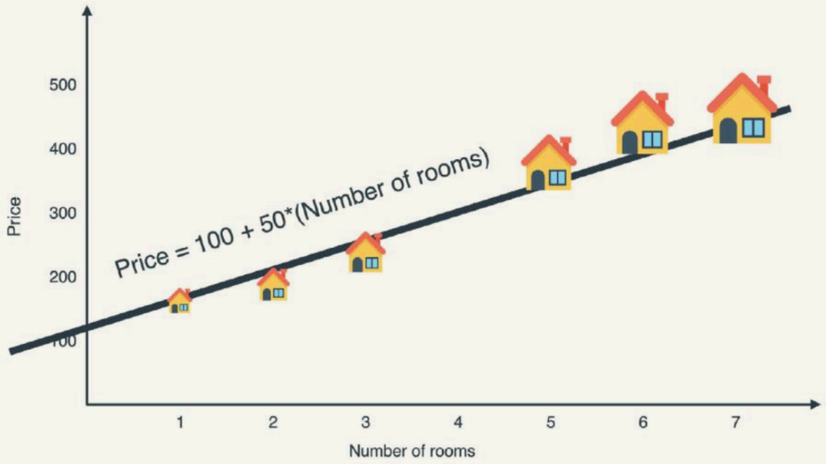
Price = 100 + 50 * (#Rooms)

Slope

Measures how steep the line is.

Y-intercept

It is the height at which the line crosses the vertical (y) axis.



Linear Equation

Linear Equation is the equation of a line. $y = mx + b$

How many lines can solve the problem?

How machines learn it? [Remember-Formulate-Predict]

Price = 100 + 50 × (#Rooms)
Price = 100 + 50 × (4) = **300**

Some Questions:

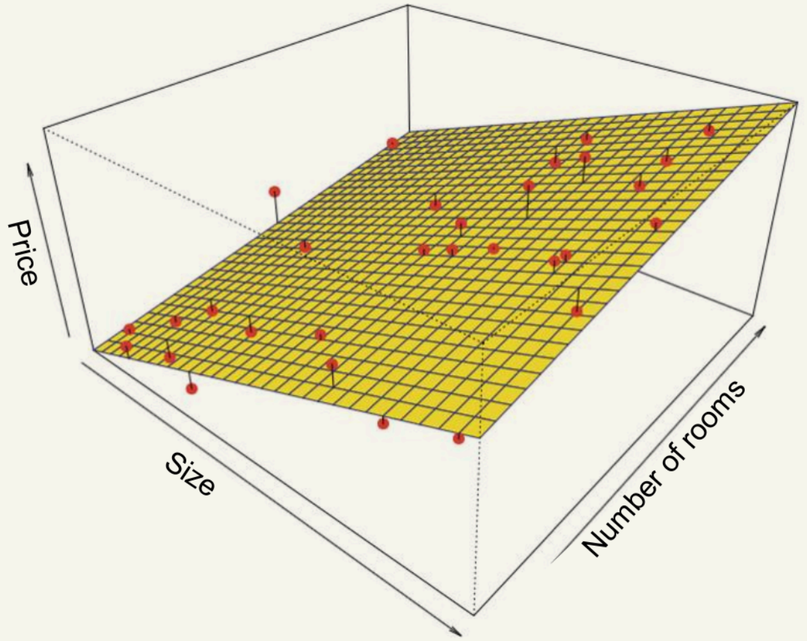
- Can we have multiple features data?
- How does computer learn this equation?

Multivariate Linear Regression

Price = 30 * (#Rooms) + 1.5 * (Size) + 10 * (Schools Quality) - 2 * (Age) + 50

What do you notice in this equation?

- 1 bias and multiple weights
- Different sign of weights
- Different weights value
- What is the shape of the model? Still linear?



How machines formulate this equation? [Overview]

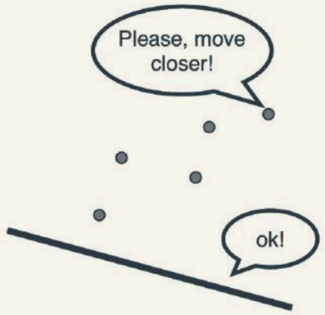
Inputs: A dataset of points.
Outputs: A linear regression model that fits that dataset.

Procedure:

- Pick a model with random weights and a random bias.
- Repeat many times:
  1. Pick a random data point.
  2. Slightly adjust the weights (Slope) and bias (y-intercept) in order to improve the prediction for that particular data point.
- Return the model you've obtained.

How machines formulate this equation? [Overview]

Step 1 | Step 2 | Step 3 | Step 4 | Step 5 | Final



Linear Relationship

A linear relationship

- True, the line doesn't pass through every dot.
- However, the line does clearly show the relationship between rooms and price.

$$y' = mx + b$$

where: - **y'**: is the price value that we're trying to predict. - **m**: is the slope of the line. - **x**: is the number of rooms value of our input feature. - **b**: is the y-intercept.

Linear Relationship in Machine Learning

In machine learning, we'll write the equation for a model slightly differently:

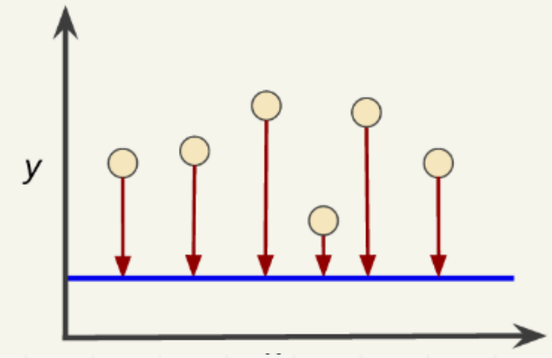
$$y' = w_1 x_1 + w_0$$

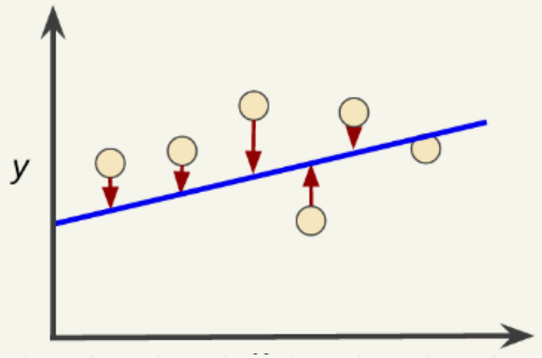
where: - **y'**: is the predicted label (a desired output). - **w_1**: is the weight of feature 1. Weight is the same concept as the "slope". - **x_1**: is feature 1. - **w_0 or b**: is the bias (the y-intercept).

Training and Loss

- **Training** a model simply means learning (determining) good values for all the weights and the bias from labeled examples.
- **Loss** is the penalty for a bad prediction.
  - Perfect prediction means the **loss is zero**.
  - Bad model have large loss.

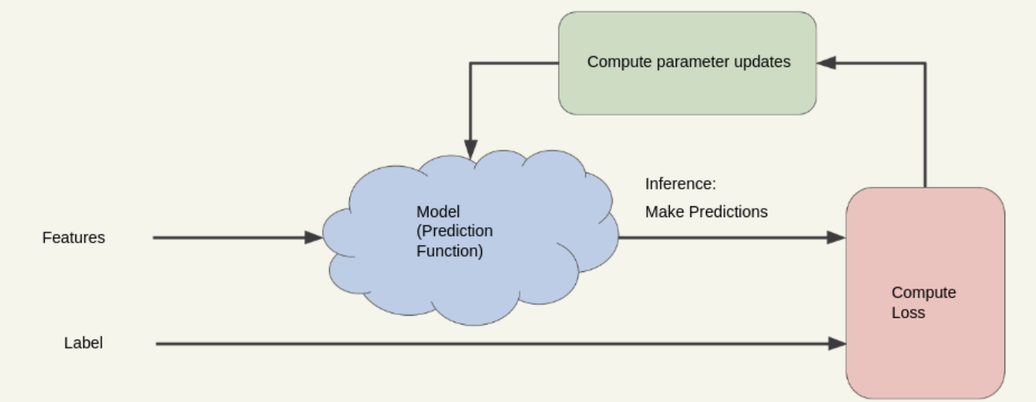
Suppose we selected the following weights and biases. **Which of them have lower loss?**





Reducing Loss

- **Training** is a **feedback iterative process** that uses the **loss function** to improve the **model parameters.**



Some Questions: - How to define **loss** to measure the performance of the model? - What **initial values** should we set for w_1 and w_0? - How to **update** w_1 and w_0?

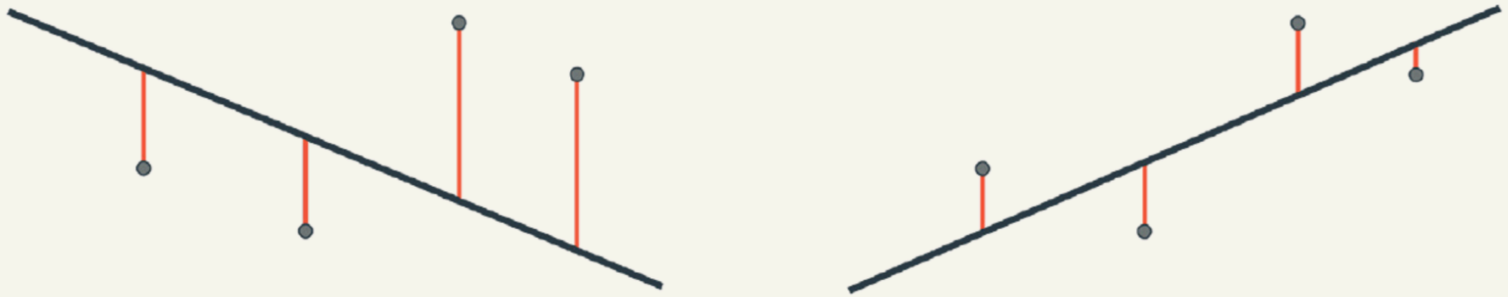
TABLE OF CONTENTS

Loss Definition

Which model is **better** and why? Which model have a **lower loss**?

Absolute Loss (L1 Loss)

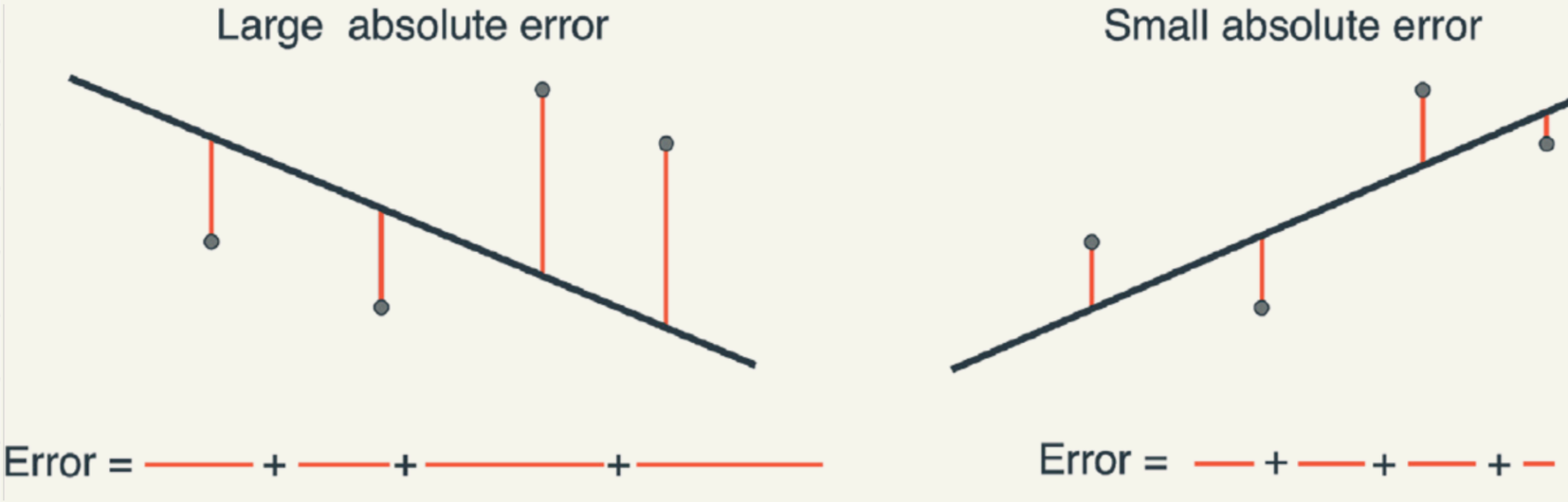
The absolute loss is the **sum** of the absolute differences between the observed and predicted values.



$$|\text{observation}(x) - \text{prediction}(x)| = |(y - y')|$$

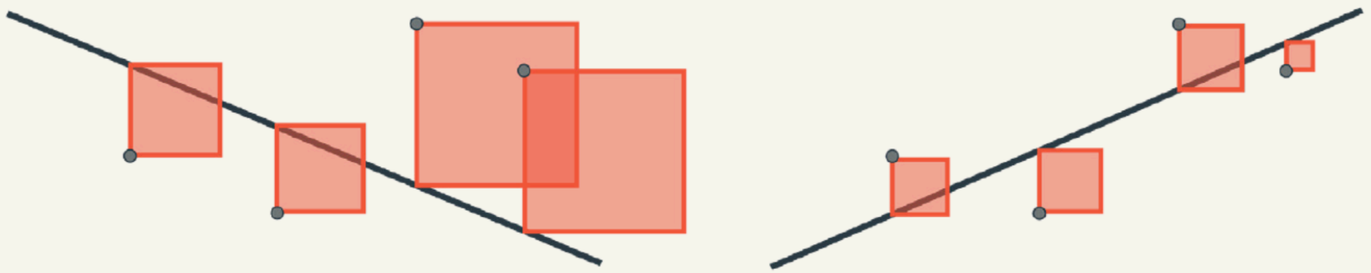
Absolute Loss (L1 Loss)

The absolute loss is the **sum** of the absolute differences between the observed and predicted values.



Squared Loss (L2 Loss)

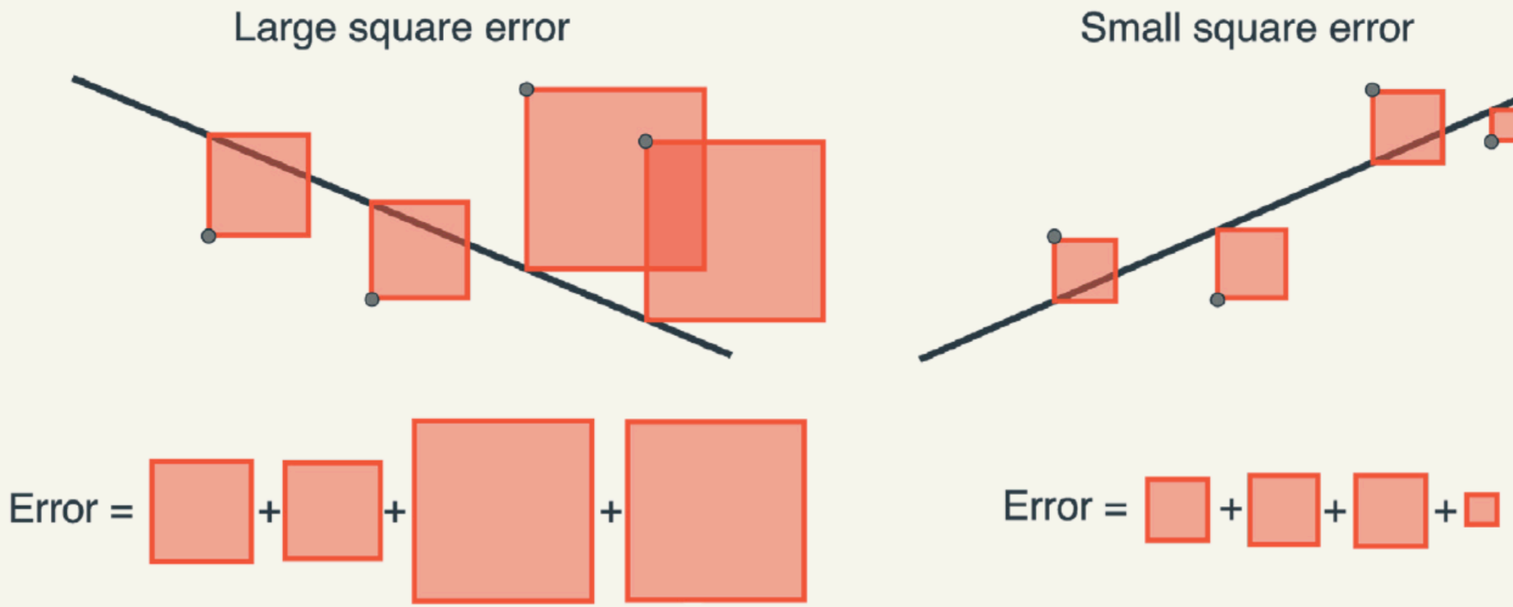
The squared loss is the **sum** of the squared differences between the observed and predicted values.



$$[\text{observation}(x) - \text{prediction}(x)]^2 = [(y - y')]^2$$

Squared Loss (L2 Loss)

The squared loss is the **sum** of the squared differences between the observed and predicted values.



Why Squared Loss?

- The squared loss is used in many machine learning algorithms because **it penalizes larger errors more than smaller ones**. This property makes it more sensitive to outliers compared to absolute loss.

- The squared loss also leads to **a nicer derivative** compared to the absolute loss, which simplifies optimization algorithms like gradient descent.

Mean Square Error (MSE)

Is the **average squared loss** per example over the whole dataset.

$$\text{MSE} = \frac{1}{N} \sum_{(x,y)\in D} (y - \text{prediction}(x))^2$$

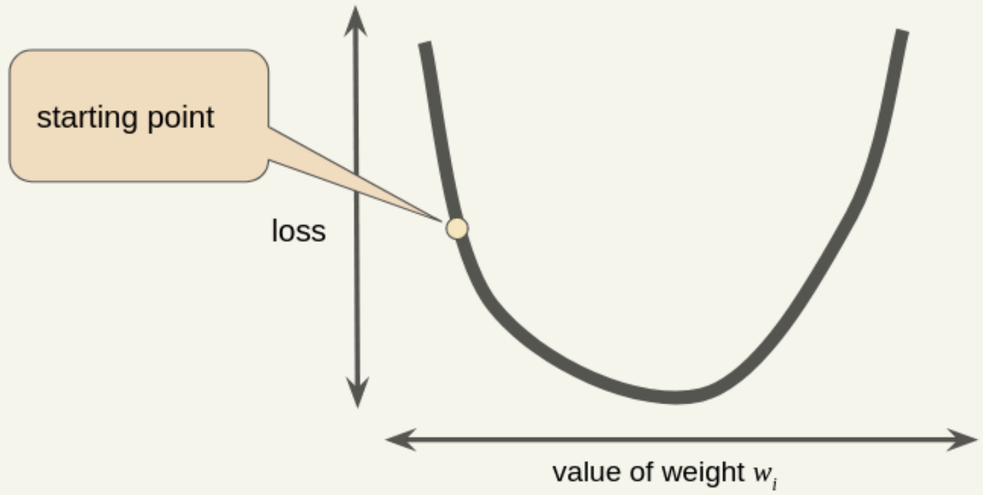
- **(x,y)** is an example in which
  - **y** is the label
  - **x** is a feature
- **prediction(x)** is equal to $y' = w_1 x + w_0$
- **D** is the dataset that contains all (x,y) pairs
- **N** is the number of samples in D

TABLE OF CONTENTS

Gradient Descent (Recap)

Concept | Direction | Magnitude

Note that a gradient is a vector, so it has both of the following characteristics: - Magnitude - Direction



$$w_{new} = w_{old} - \eta \cdot \frac{d\ \text{loss}}{dw}$$

Gradient for Linear Regression

- For linear regression, the gradient of the loss function with respect to the weights w_1 and w_0 can be derived as follows:
- The loss function for linear regression is typically the l2 loss:

$$\text{loss}(w_0, w_1) = [y - y']^2 = [y - (w_1 x + w_0)]^2$$

- The gradient of the loss function with respect to the weights w_1 and w_0 can be derived as follows:

$$\frac{d\ \text{loss}}{dw_1} = 2(y - y')(-x) = 2x(y' - y)$$

$$\frac{d\ \text{loss}}{dw_0} = 2(y - y')(-1) = 2(y' - y)$$

Full Gradient for Linear Regression

Procedure:

- Pick random weight w_1 and a random bias w_0.
- Repeat **many times**:
  1. Pick a random data point $(x^{(i)}, y^{(i)})$.
  2. Compute Model Prediction $y'^{(i)} = w_1 x_1^{(i)} + w_0$
  3. Update the weights and bias using the following equations:

$$w_1 = w_1 - \eta \frac{d\ \text{loss}}{dw_1} = w_1 - \eta 2 x_1^{(i)} (\underbrace{y'^{(i)} - y^{(i)}}_{\text{error}})$$

$$w_0 = w_0 - \eta \frac{d\ \text{loss}}{dw_0} = w_0 - \eta 2 (\underbrace{y'^{(i)} - y^{(i)}}_{\text{error}})$$

- Return the model you've obtained.

Gradient Descent for Linear Regression

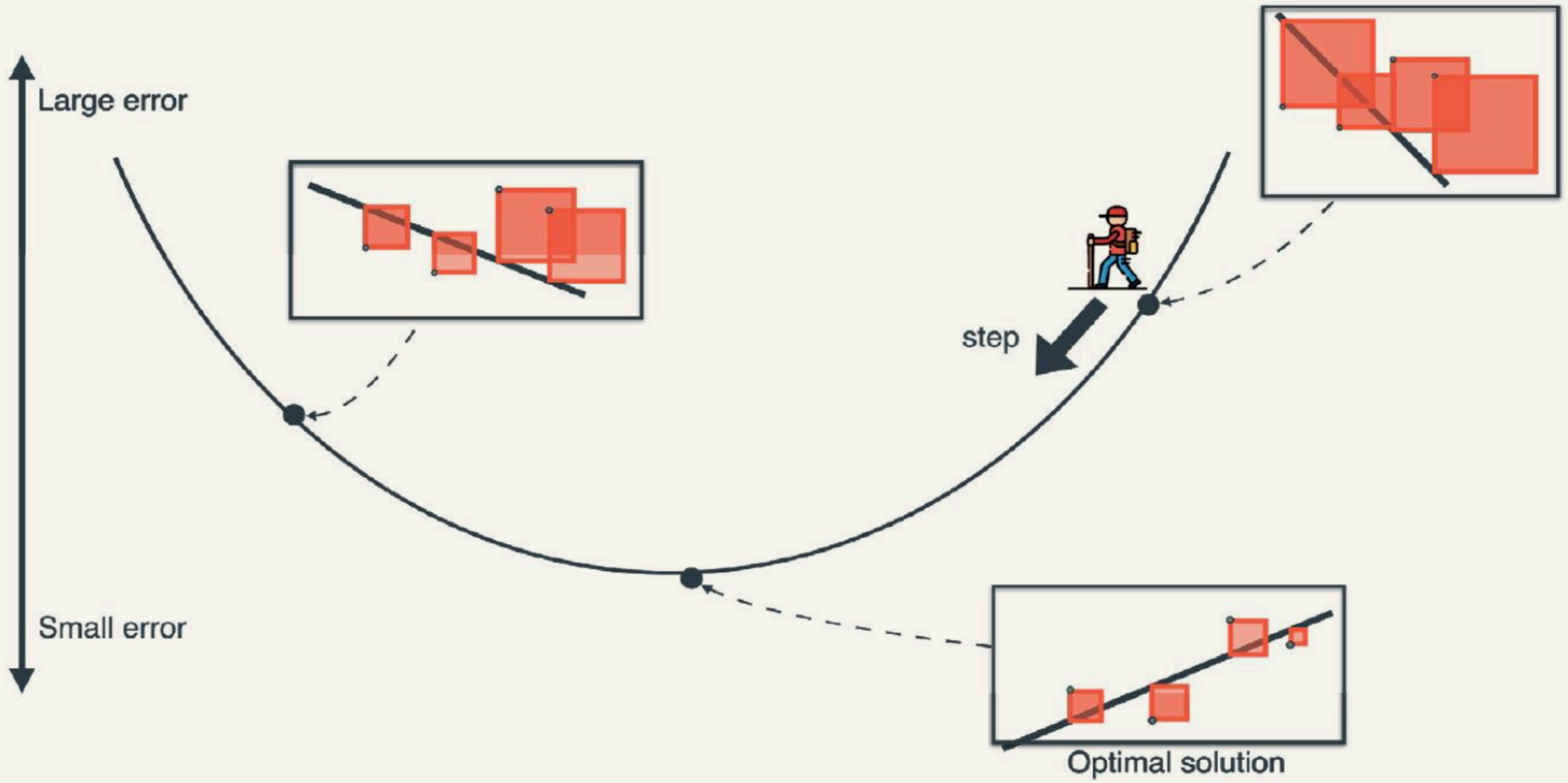


TABLE OF CONTENTS

Convergence Criteria

- For convex functions, optimum occurs when:

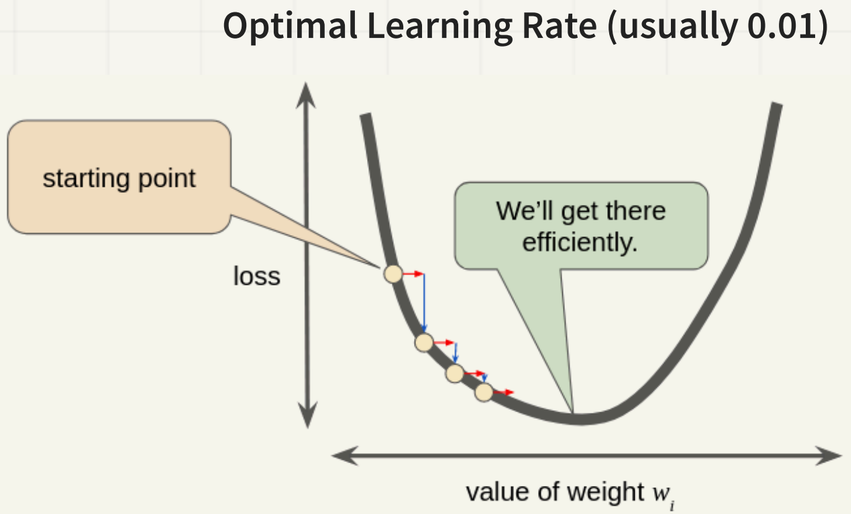
$$\left| \frac{d \text{ loss}}{dw} \right| = 0$$

- In practice, stop when:

$$\left| \frac{d \text{ loss}}{dw} \right| \leq \epsilon$$

Learning Rate

- Gradient descent algorithms **multiply the gradient** by a scalar known as the **learning rate** (also sometimes called step size).
- How can we choose the learning rate?



Types of Gradient Descent

- **Batch Gradient Descent:**
  - MSE loss assumes taking gradient for the total number of samples in the data set.
  - Data sets often contain billions or even hundreds of billions of examples.
  - Can take a very long time to compute.
- **Stochastic Gradient Descent (SGD):**
  - Uses only a single example (a batch size of 1) per iteration.
  - Very noisy.
- **Mini-Batch Gradient Descent:**
  - Compromise between full-batch iteration and SGD.
  - Typically a batch of size between 10 and 1,000 examples, chosen at random.

TABLE OF CONTENTS

Multivariate Linear Regression

- The general case of linear regression has more than one input feature.
- The model now becomes:

$$y' = w_1 x_1 + w_2 x_2 + \cdots + w_n x_n + w_0$$

- We can rewrite this as:

$$y' = \sum_{i=0}^{i=n} w_i x_i$$

- Note w_0 is the bias (intercept), and $x_0 = 1$.

Generalization and Gradient

- For n features: $y' = \sum_{i=0}^{i=n} w_i x_i$
- Note w_0 is the bias (intercept), and $x_0 = 1$.
- Vector representation: $y' = \mathbf{w}^T \mathbf{x}$
- Loss = $\ell = (y - y')^2$

Gradient Derivation

$$\begin{aligned} \frac{d\ell}{dw_i} &= \frac{d\ell}{dy'} \frac{dy'}{dw_i} \\ &= 2(y' - y) \cdot x_i \end{aligned}$$

Multivariate Output Visualization

	Size (feet²)	Number of bedrooms	Number of floors	Age of home (years)	Price ($1000)
feature →	X_1	X_2	X_3	X_4	Y
data sample → $X^{(1)}$	2104	5	1	45	460
$X^{(2)}$	1416	3	2	40	232
$X^{(3)}$	1534	3	2	30	315
$X^{(4)}$	852	2	1	36	178
Weights	W_1	W_2	W_3	W_4	

$$X \begin{bmatrix} 2104 & 5 & 1 & 45 \\ 1416 & 3 & 2 & 40 \\ 1534 & 3 & 2 & 30 \\ 852 & 2 & 1 & 36 \end{bmatrix} * W \begin{bmatrix} W_1 \\ W_2 \\ W_3 \\ W_4 \end{bmatrix} + b = Y' \begin{bmatrix} y'_1 \\ y'_2 \\ y'_3 \\ y'_4 \end{bmatrix}$$

[#samples* #features] [#features*1] [#samples*1]

TABLE OF CONTENTS

The Normal Equation

$$X \begin{bmatrix} 2104 & 5 & 1 & 45 \\ 1416 & 3 & 2 & 40 \\ 1534 & 3 & 2 & 30 \\ 852 & 2 & 1 & 36 \end{bmatrix} * W \begin{bmatrix} W_1 \\ W_2 \\ W_3 \\ W_4 \end{bmatrix} + b = Y' \begin{bmatrix} y'_1 \\ y'_2 \\ y'_3 \\ y'_4 \end{bmatrix}$$

[#samples* #features] [#features*1] [#samples*1]

$$X \begin{bmatrix} 1 & 2104 & 5 & 1 & 45 \\ 1 & 1416 & 3 & 2 & 40 \\ 1 & 1534 & 3 & 2 & 30 \\ 1 & 852 & 2 & 1 & 36 \end{bmatrix} * W \begin{bmatrix} b \\ W_1 \\ W_2 \\ W_3 \\ W_4 \end{bmatrix} = Y' \begin{bmatrix} y'_1 \\ y'_2 \\ y'_3 \\ y'_4 \end{bmatrix}$$

[#samples*(#features+1)] [(#features+1)*1] [#samples*1]

Normal Equation Closed-Form

- Normal equation is a closed-form solution to the linear regression problem.
- It provides an exact solution to the model parameters $\mathbf{W}$ that minimize the squared error.
- Since we have:

$$\mathbf{Y} = \mathbf{X} \cdot \mathbf{W}$$

- Multiply both sides by the inverse of $\mathbf{X}$:

$$\mathbf{X}^{-1} \cdot \mathbf{Y} = \mathbf{W}$$

- Since $\mathbf{X}$ is not a square matrix, **we can't find the inverse of $\mathbf{X}$**.
- We can use the following trick:

$$\mathbf{X}^T\mathbf{Y} = \mathbf{X}^T\mathbf{X}\mathbf{W}$$

- Now, multiply both sides by the inverse of $\mathbf{X}^T\mathbf{X}$:

$$(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y} = \mathbf{W}$$

Thank You!