AI VIET NAM
@aivietnam.edu.vn

Linear Regression

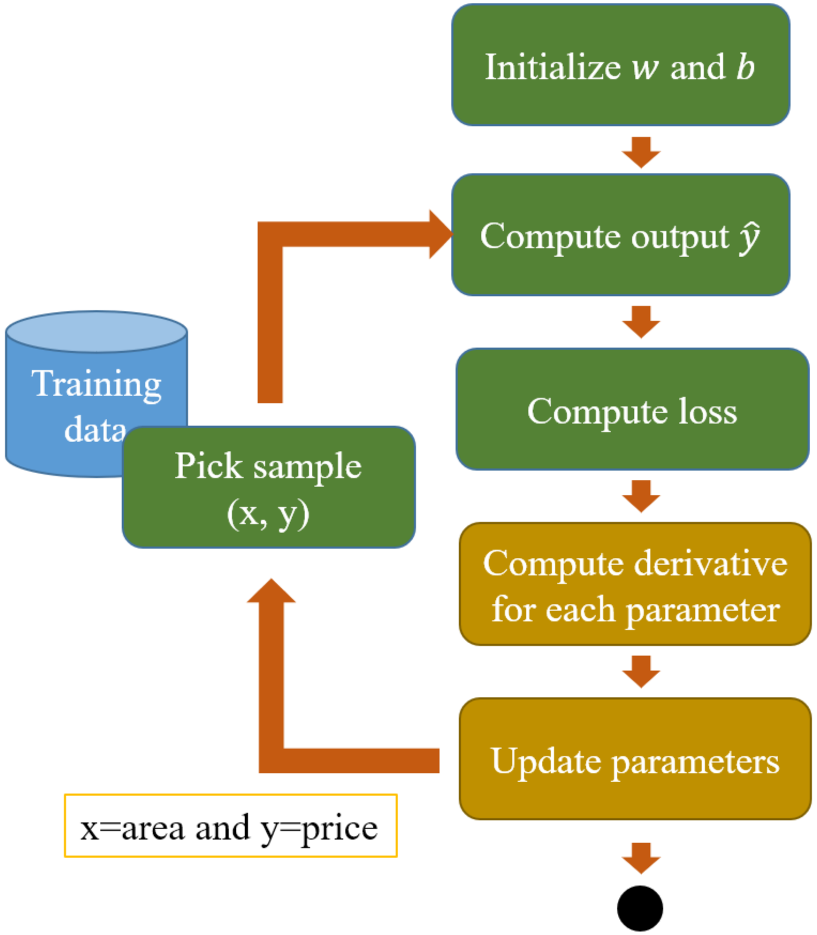
A Simple Approach

Quang-Vinh Dinh
PhD in Computer Science

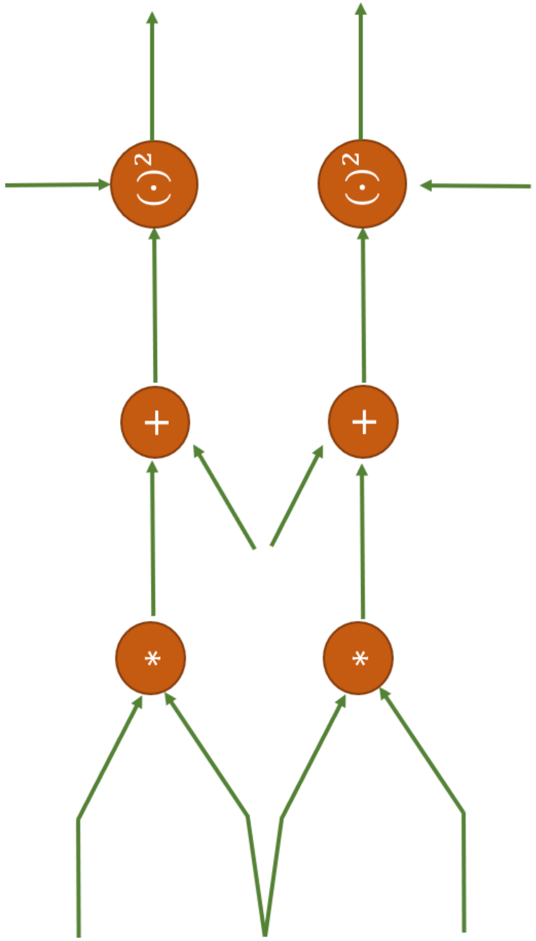
Year 2024

Objectives

Linear Regression



Computational Graph



Batch Training

1) Pick all the N samples $(x^{(i)}, y^{(i)})$ from training data

2) Compute output $\hat{y}^{(i)}$

$$\hat{y}^{(i)} = wx^{(i)} + b \qquad \text{for } 0 \le i < N$$

3) Compute loss

$$L^{(i)} = (\hat{y}^{(i)} - y^{(i)})^2 \qquad \text{for } 0 \le i < N$$

4) Compute derivatives

$$\frac{\partial L^{(i)}}{\partial w} = 2x^{(i)}(\hat{y}^{(i)} - y^{(i)})$$

$$\frac{\partial L^{(i)}}{\partial b} = 2(\hat{y}^{(i)} - y^{(i)}) \qquad \text{for } 0 \le i < N$$

5) Update

$$w = w - \eta \frac{\sum_i \frac{\partial L^{(i)}}{\partial w}}{N} \qquad b = b - \eta \frac{\sum_i \frac{\partial L^{(i)}}{\partial b}}{N}$$

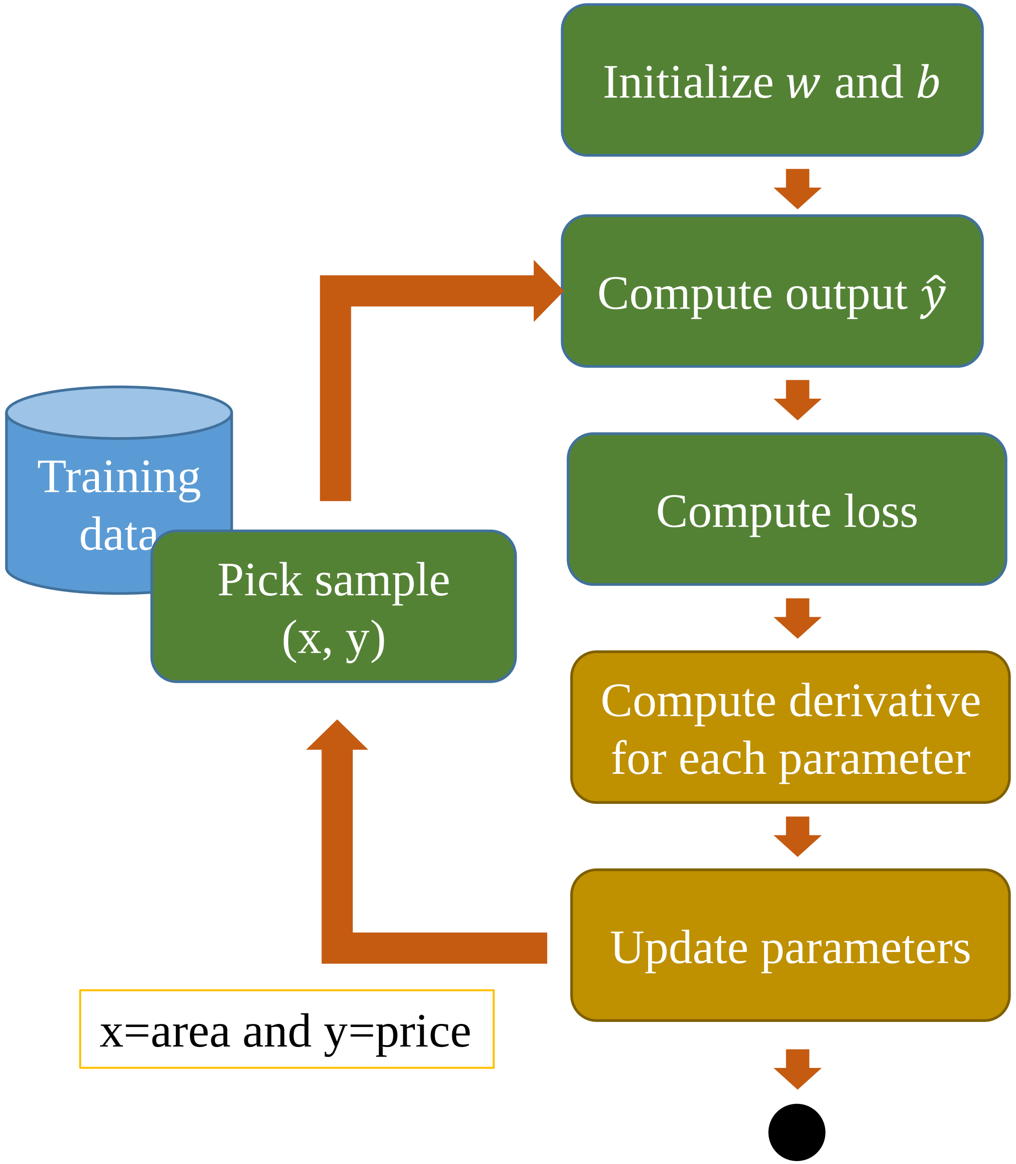
Outline

SECTION 1
Linear Regression

SECTION 2
Mini-batch Training

SECTION 3
Batch Training

SECTION 4
Loss Functions



Linear Regression

Introduction

Feature | Label

area	price
6.7	9.1
4.6	5.9
3.5	4.6
5.5	6.7

House price data

Features | Label

TV	Radio	Newspaper	Sales
230.1	37.8	69.2	22.1
44.5	39.3	45.1	10.4
17.2	45.9	69.3	12
151.5	41.3	58.5	16.5
180.8	10.8	58.4	17.9

Advertising data

if area=6.0, price=?

if TV=55.0, Radio=34.0, and Newspaper=62.0, price=?

Features | Label

crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black	lstat	medv
0.00632	18	2.31	0	0.538	6.575	65.2	4.09	1	296	15.3	396.9	4.98	24
0.02731	0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.9	9.14	21.6
0.03237	0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
0.06905	0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.9	5.33	36.2
0.08829	12.5	7.87	0	0.524	6.012	66.6	5.5605	5	311	15.2	395.6	12.43	22.9

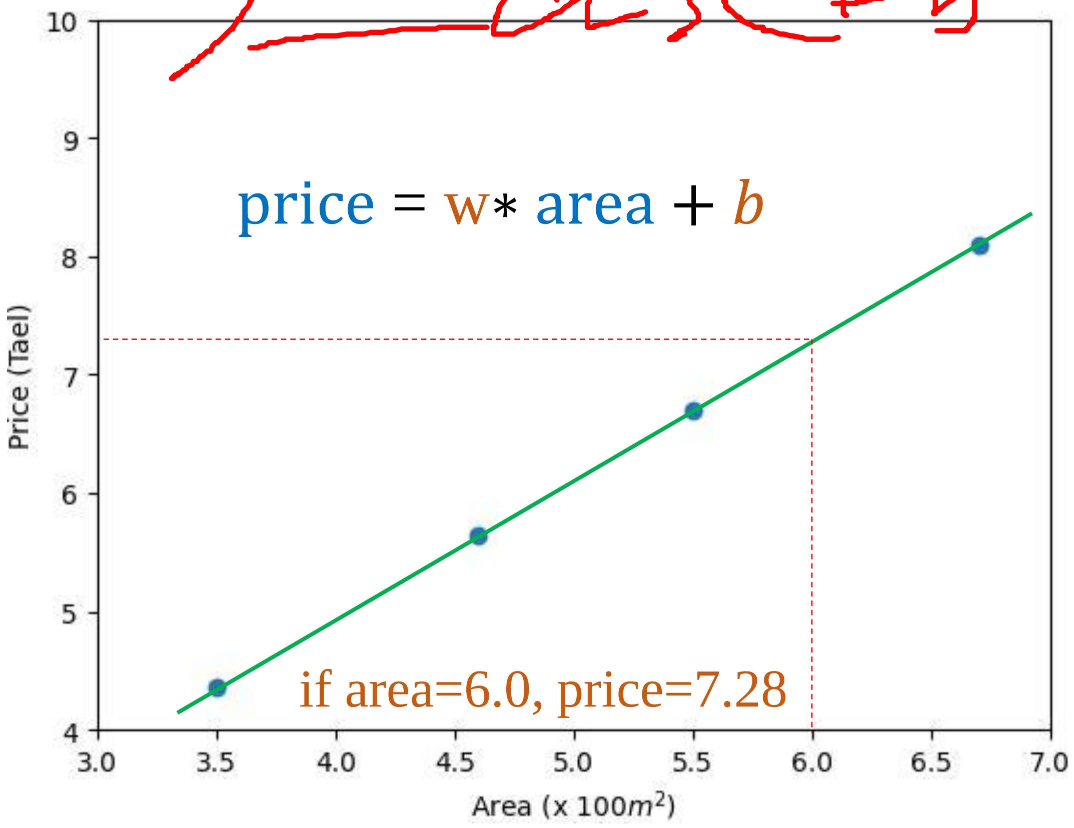
Boston House Price Data

House Price Prediction

Feature	Label
area	price
6.7	8.1
4.6	5.6
3.5	4.3
5.5	6.7

House price data

$y = ax + b$



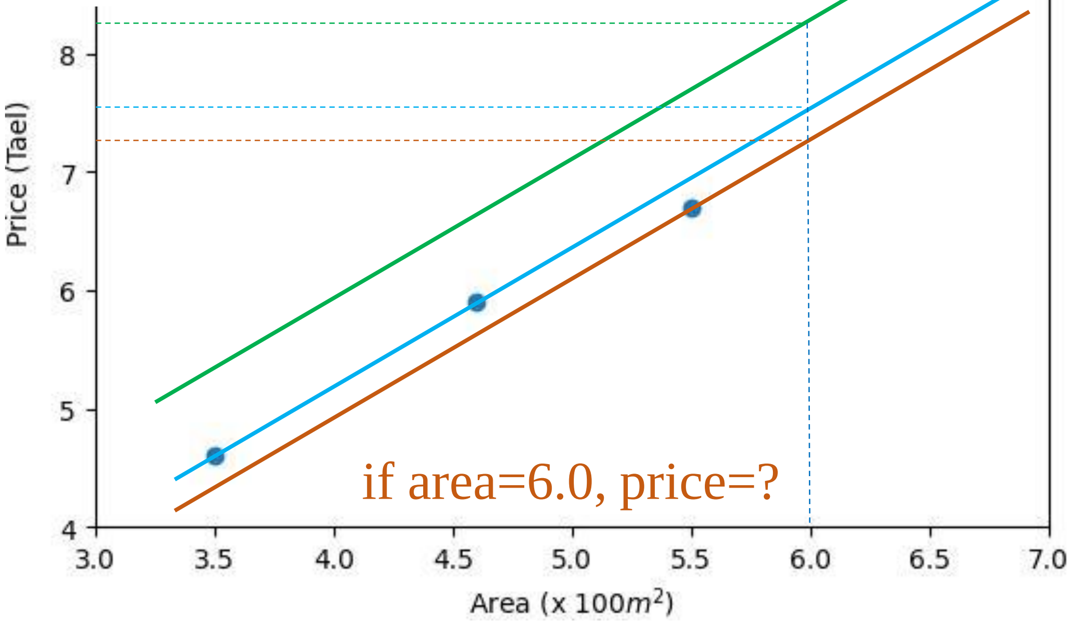
Feature	Label
area	price
6.7	9.1
4.6	5.9
3.5	4.6
5.5	6.7

House price data

$$\text{price} = w_1 * \text{area} + b_1$$

$$\text{price} = w_2 * \text{area} + b_2$$

$$\text{price} = w_3 * \text{area} + b_3$$

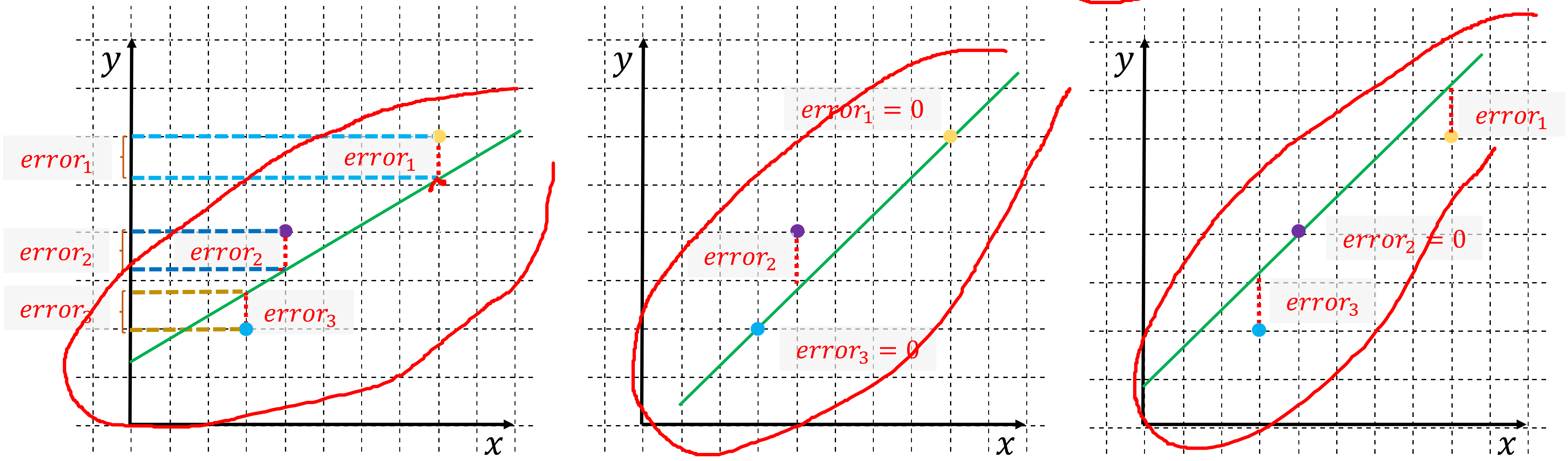


Linear Regression

❖ Area-based house price prediction

Training data

$$error_i = distance(\hat{y}_i, y_i)$$



$\hat{y} = w_1 x + b_1$ $\hat{y} = w_2 x + b_2$ $\hat{y} = w_3 x + b_3$

Find w and b whose model has the smallest error, where $error = \frac{1}{N}\sum_i error_i$ How?

Linear Regression

❖ Area-based house price prediction

weight bias

predicted_price = w * area + b

error = (predicted_price − real_price)^2

$$\hat{y}_i = wx_i + b$$

$$L(\hat{y}_i, y_i) = \boxed{(\hat{y}_i - y_i)^2}$$

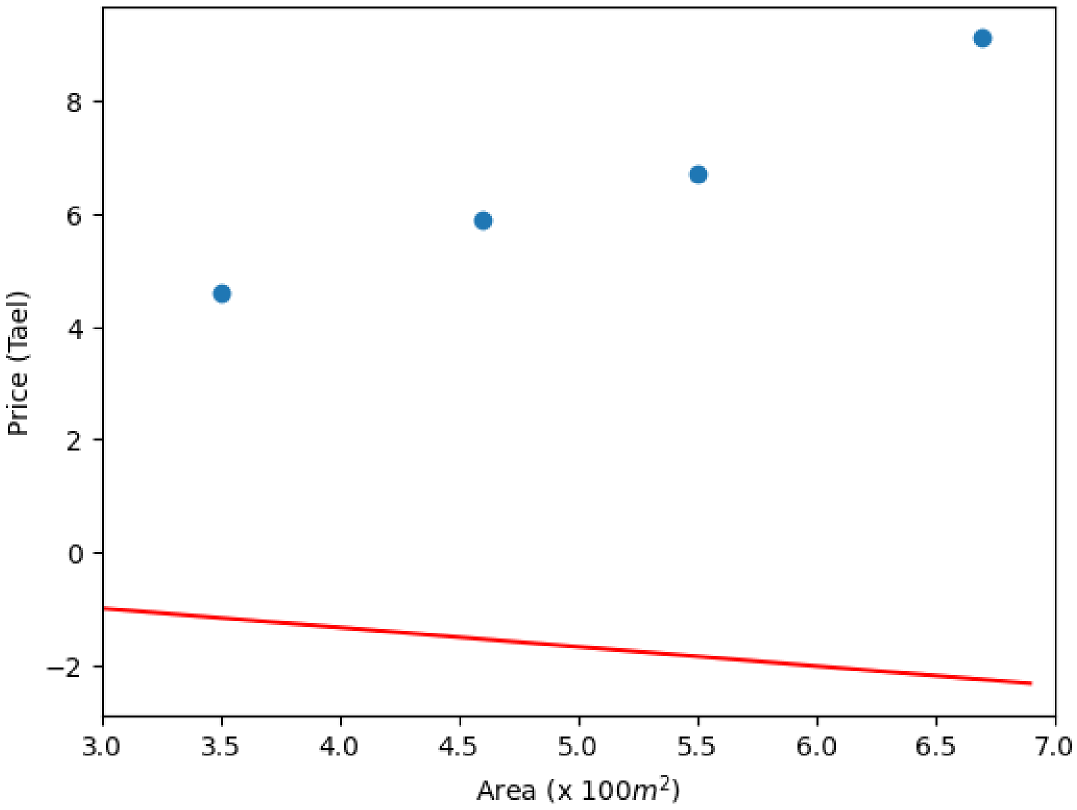
$\hat{y}$

y

area	price	predicted	error
6.7	9.1	-2.238	128.55
4.6	5.9	-1.524	55.11
3.5	4.6	-1.15	33.06
5.5	6.7	-1.83	72.76

w = -0.34

b = 0.04



Linear Regression

❖ Area-based house price prediction

$$w = 1.17$$

$$b = 0.26$$

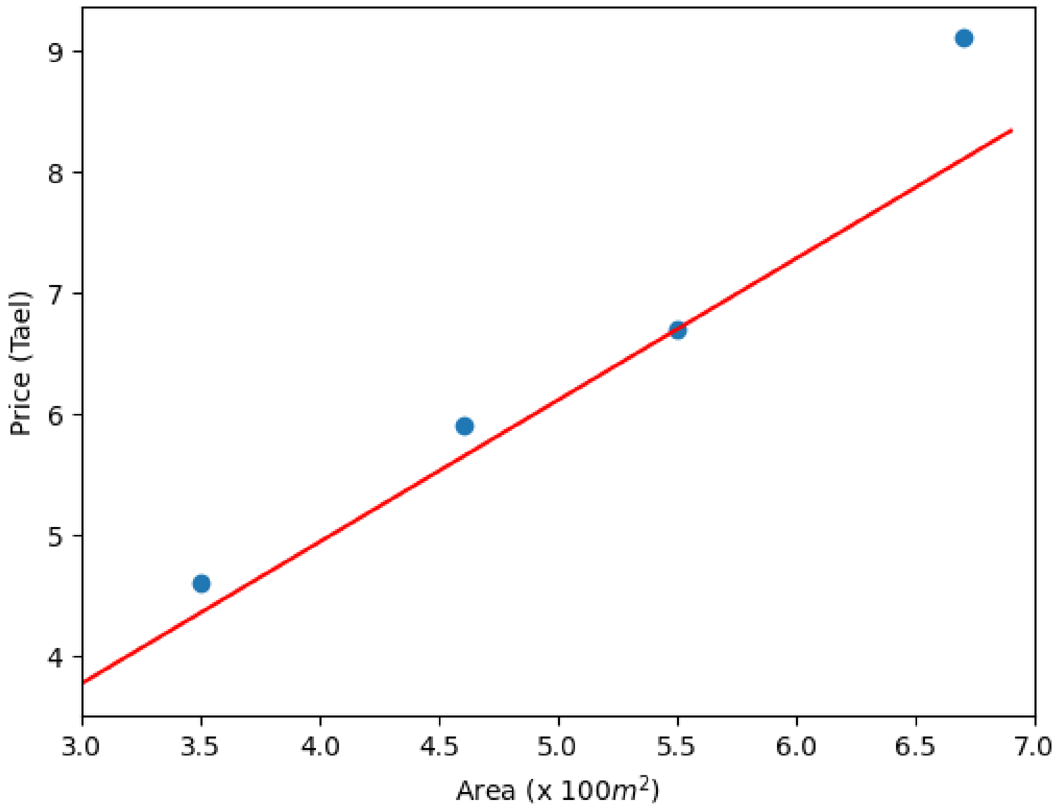
$$\text{predicted\_price} = w * \text{area} + b$$

$$\text{error} = (\text{predicted\_price} - \text{real\_price})^2$$

$$\hat{y}_i = wx_i + b$$

$$L(\hat{y}_i, y_i) = (\hat{y}_i - y_i)^2$$

area	price	predicted	error
6.7	9.1	8.099	1.002
4.6	5.9	5.642	0.066
3.5	4.6	4.355	0.06
5.5	6.7	6.695	0.00002



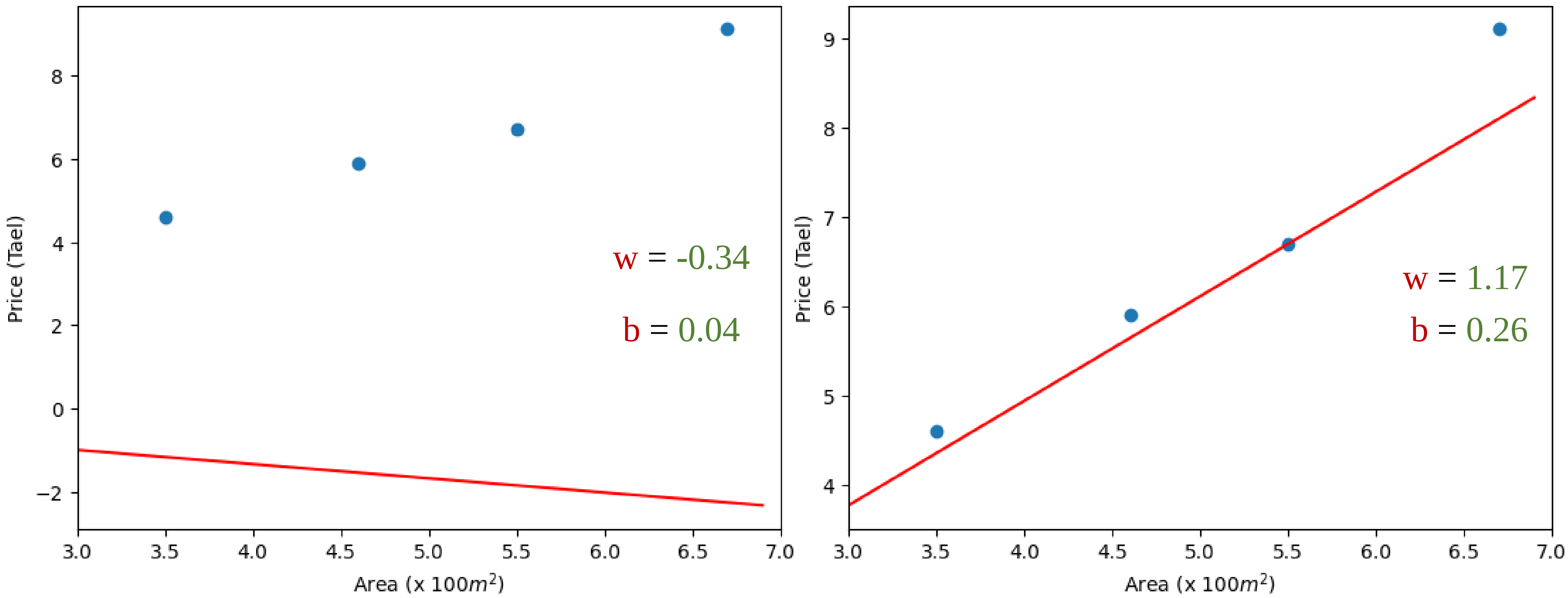
Linear Regression

❖ Area-based house price prediction

$$\hat{y}_i = wx_i + b$$

$$L(\hat{y}_i, y_i) = (\hat{y}_i - y_i)^2$$

How to change w and b so that $L(\hat{y}_i, y_i)$ reduces



Linear Regression

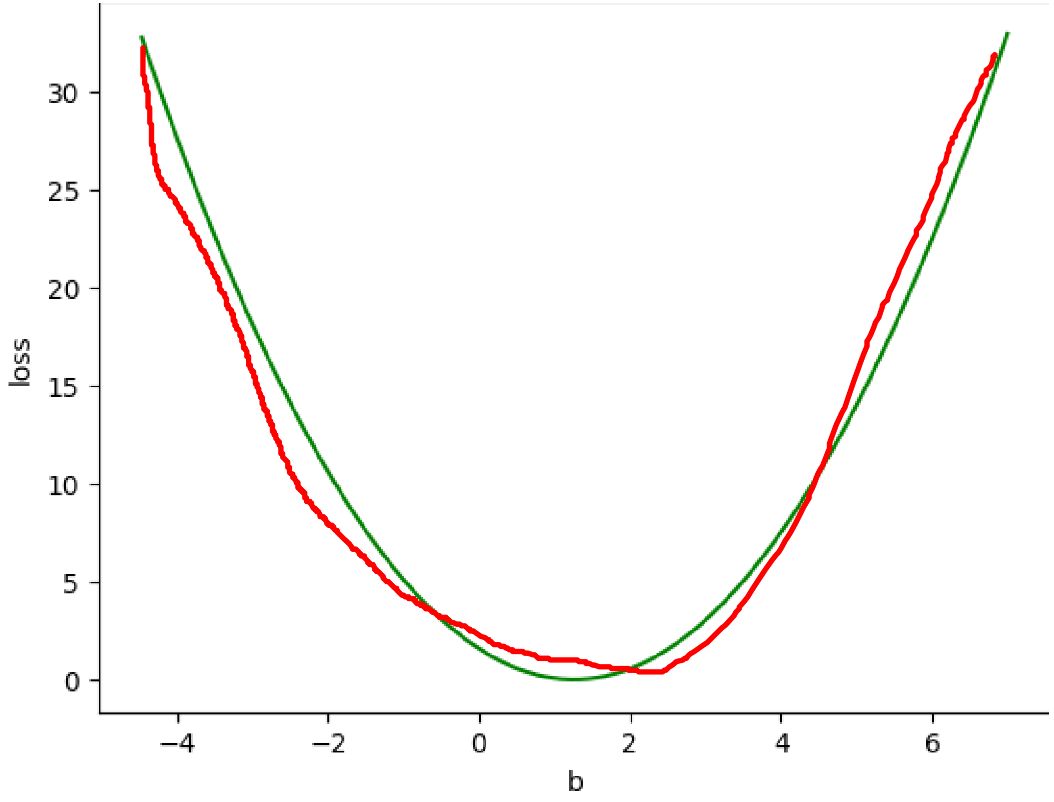
$$\hat{y}_i = wx_i + b$$

$$L(\hat{y}_i, y_i) = (\hat{y}_i - y_i)^2$$

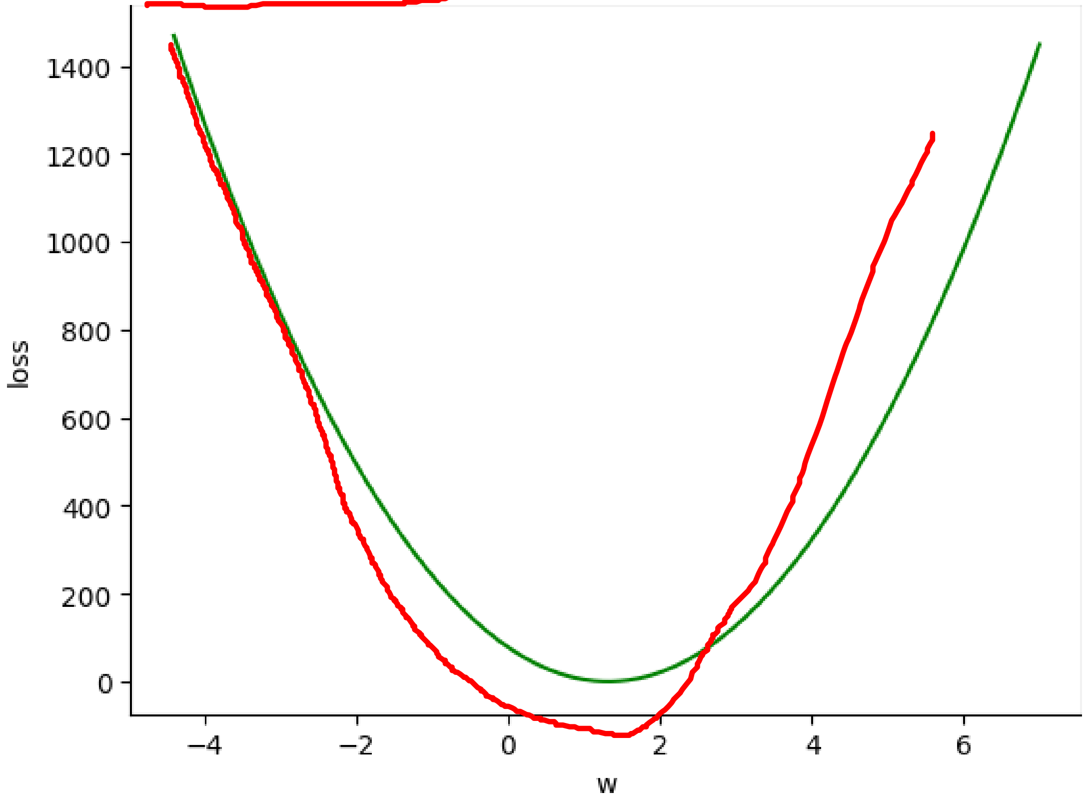
How to change w and b so that $L(\hat{y}_i, y_i)$ reduces

❖ Understanding the loss function

w b



Different b values with a fixed w value



Different w values with a fixed b value

Linear Regression

Linear equation

$$\hat{y} = wx + b$$

where $\hat{y}$ is a predicted value,

w and b are parameters

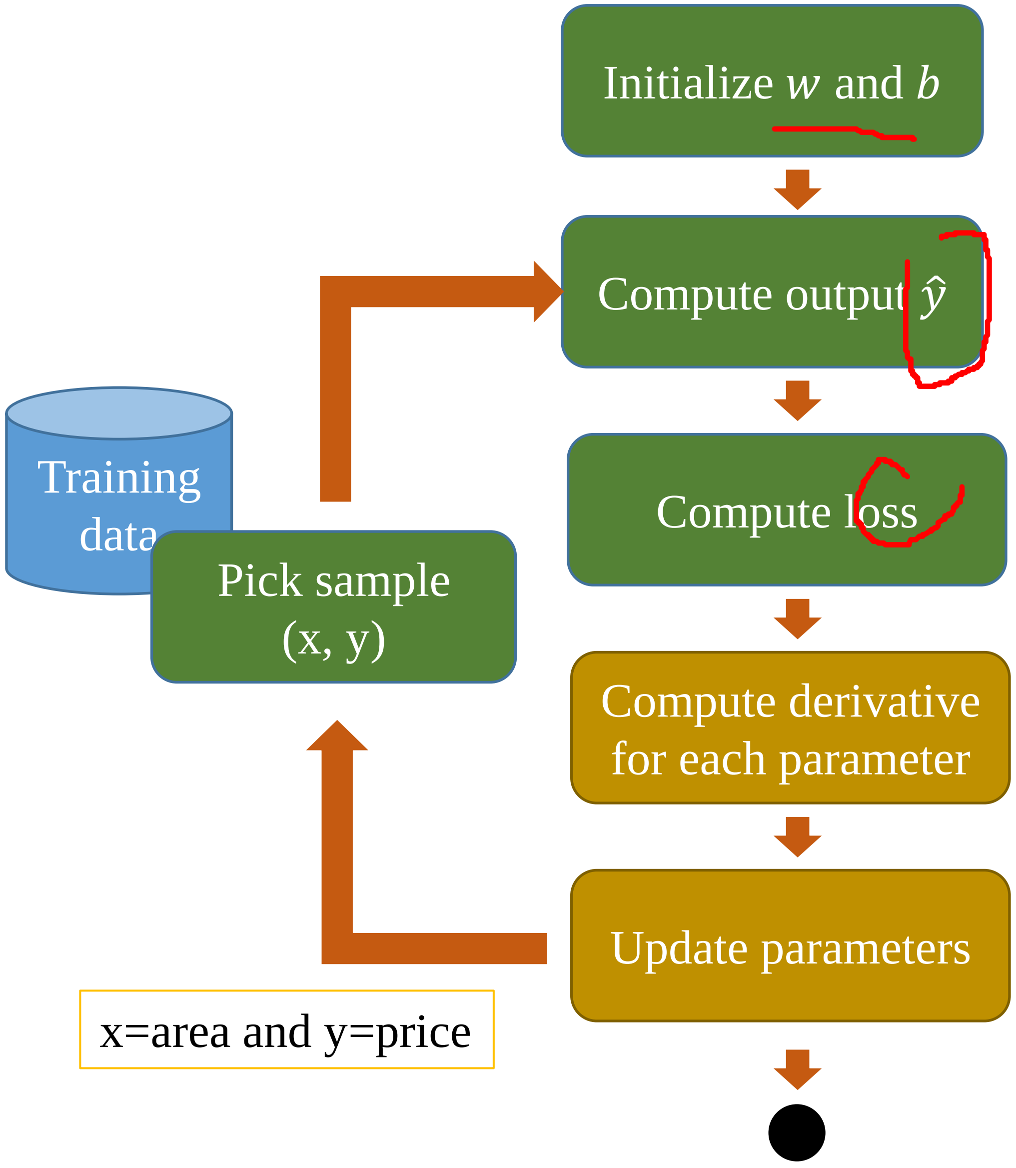
and x is an input feature

Error (loss) computation

Idea: compare predicted values $\hat{y}$ and label values y

Squared loss

$$L(\hat{y}, y) = (\hat{y} - y)^2$$



Linear Regression

Linear equation

$$\hat{y} = wx + b$$

where $\hat{y}$ is a predicted value,

w and b are parameters

and x is an input feature

Error (loss) computation

Idea: compare predicted values $\hat{y}$ and label values y

Squared loss

$$L(\hat{y}, y) = (\hat{y} - y)^2$$

Find better w and b

Use gradient descent to minimize the loss function

Compute derivate for each parameter

$$\frac{\partial L}{\partial w} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial w} = 2x(\hat{y} - y)$$

$$\frac{\partial L}{\partial b} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial b} = 2(\hat{y} - y)$$

Update parameters

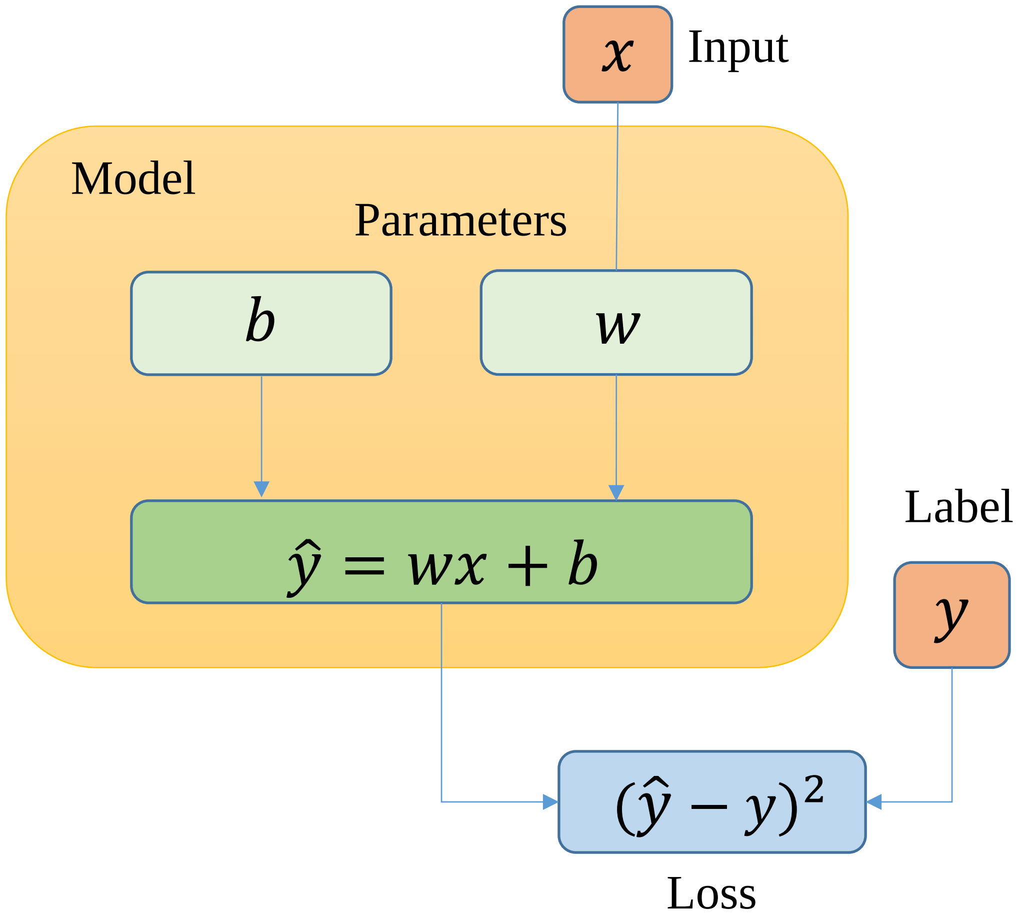
$$w = w - \eta \frac{\partial L}{\partial w} \qquad b = b - \eta \frac{\partial L}{\partial b}$$

η is learning rate

Linear Regression

❖ Simple example

Diagram



Cheat sheet

Compute the output $\hat{y}$

$$\hat{y} = wx + b$$

Compute the loss

$$L = (\hat{y} - y)^2$$

Compute derivative

$$\frac{\partial L}{\partial w} = 2x(\hat{y} - y)$$

$$\frac{\partial L}{\partial b} = 2(\hat{y} - y)$$

Update parameters

$$w = w - \eta \frac{\partial L}{\partial w}$$

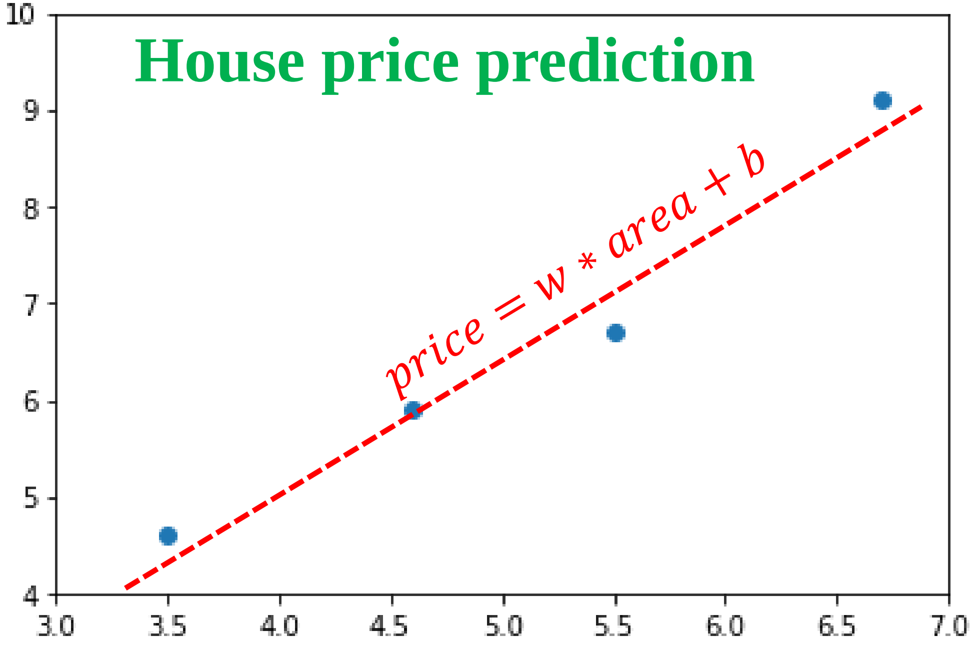
$$b = b - \eta \frac{\partial L}{\partial b}$$

Linear Regression

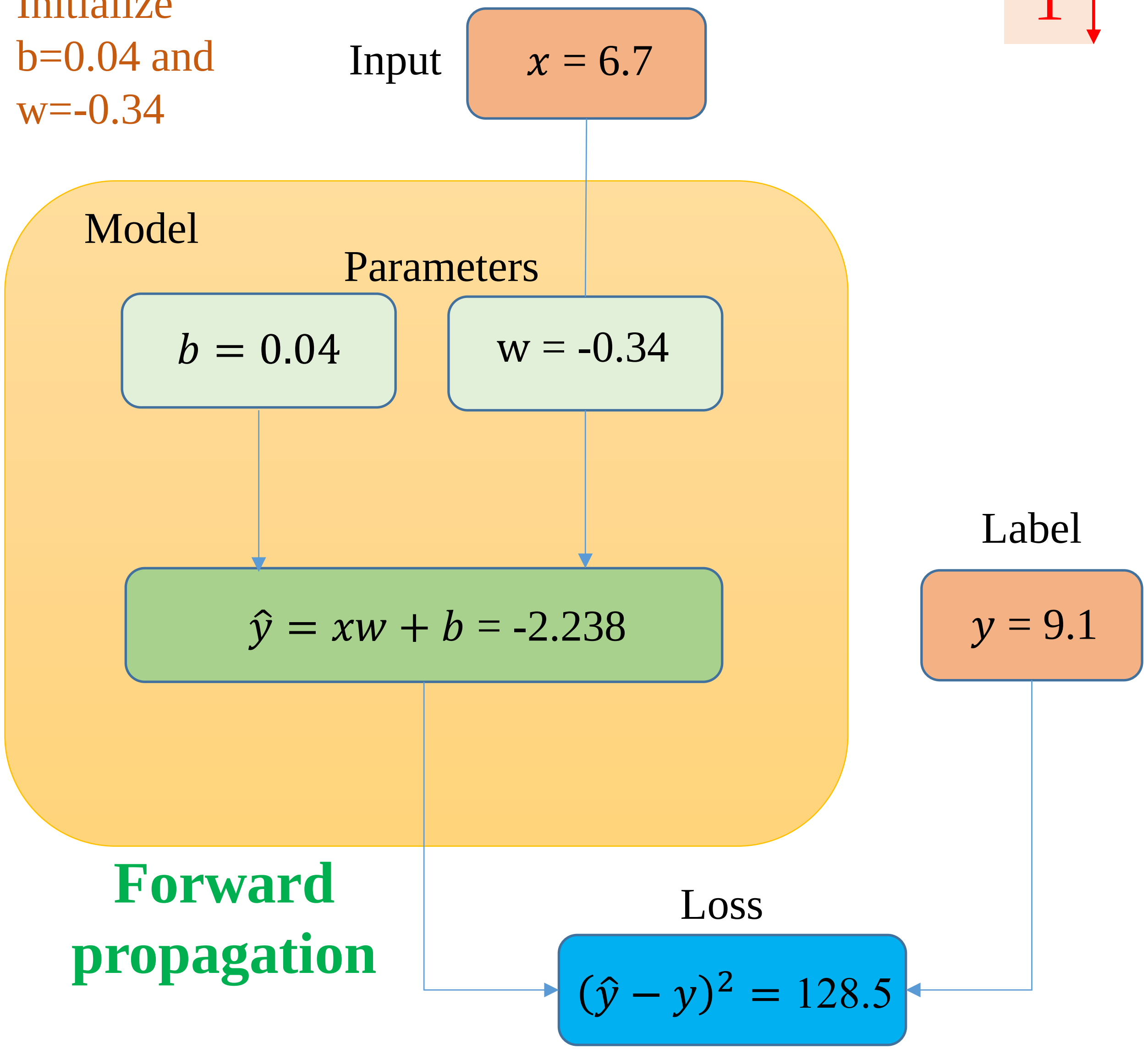
1

Given sample data

	Feature	Label
	area	price
	6.7	9.1
	4.6	5.9
	3.5	4.6
	5.5	6.7

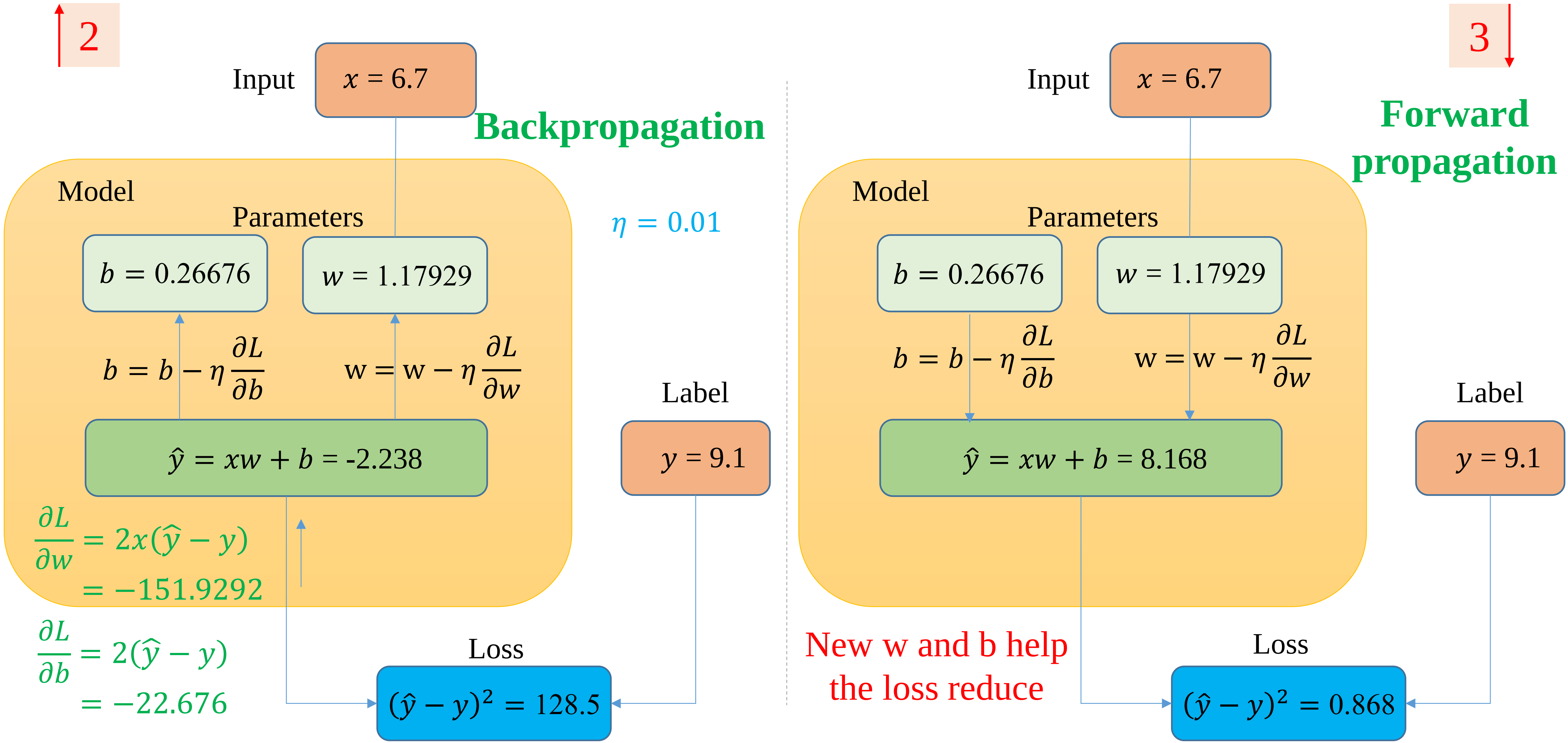


Initialize
b=0.04 and
w=-0.34



Forward propagation

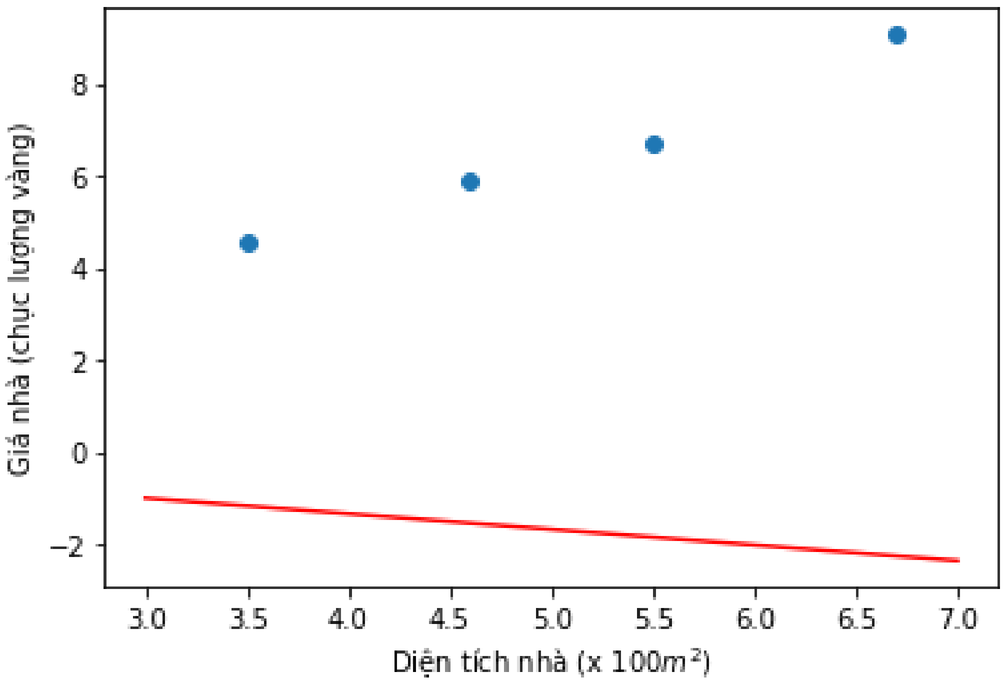
Linear Regression



Linear Regression

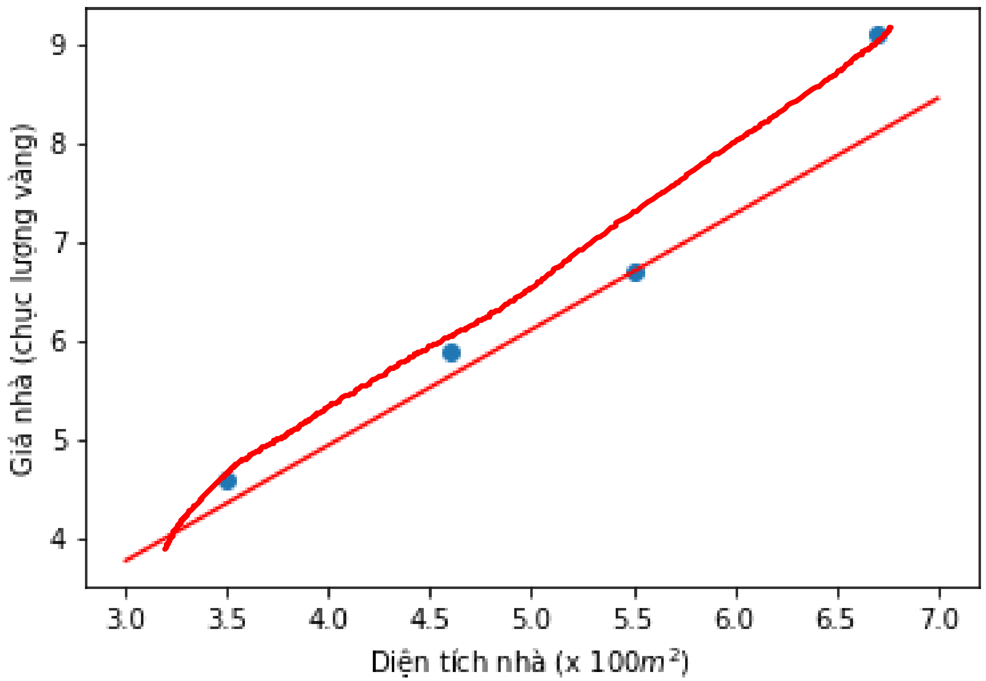
❖ Simple example

Model prediction before and after the first update



w = -0.34 b = 0.04 L = 128.55

Before updating

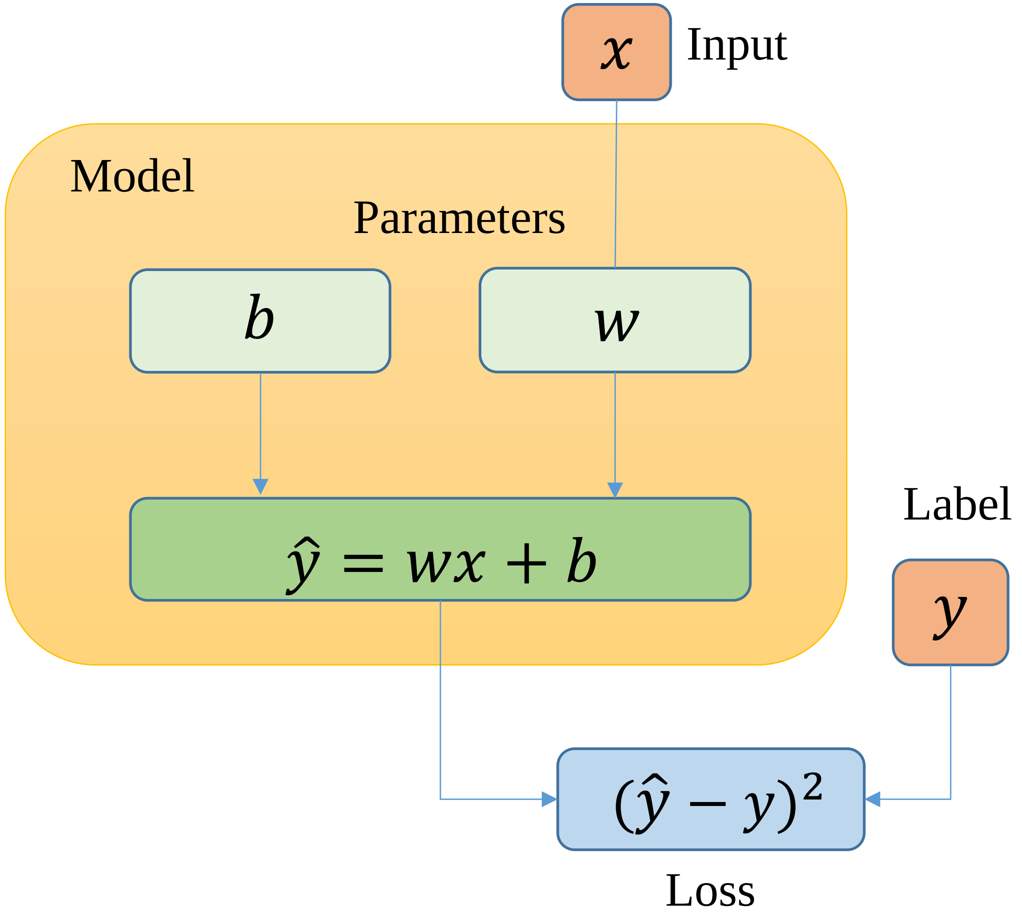


w = 1.179292 b = 0.26676 L = 0.868

After updating

Linear Regression

❖ Summary (simple version)



1) Pick a sample (x, y) from training data

2) Compute the output $\hat{y}$

$$\hat{y} = wx + b$$

3) Compute loss

$$L = (\hat{y} - y)^2$$

4) Compute derivative

$$\frac{\partial L}{\partial w} = 2x(\hat{y} - y) \qquad \frac{\partial L}{\partial b} = 2(\hat{y} - y)$$

5) Update parameters

$$w = w - \eta \frac{\partial L}{\partial w} \qquad b = b - \eta \frac{\partial L}{\partial b}$$

η is learning rate

Linear Regression

❖ Implementation

Cheat sheet

Compute the output $\hat{y}$

$$\hat{y} = wx + b$$

Compute the loss

$$L = (\hat{y} - y)^2$$

Compute derivative

$$\frac{\partial L}{\partial w} = 2x(\hat{y} - y)$$

$$\frac{\partial L}{\partial b} = 2(\hat{y} - y)$$

Update parameters

$$w = w - \eta \frac{\partial L}{\partial w}$$

$$b = b - \eta \frac{\partial L}{\partial b}$$

```python
# forward
def predict(x, w, b):
    return x*w + b

# compute gradient
def gradient(y_hat, y, x):
    dw = 2*x*(y_hat-y)
    db = 2*(y_hat-y)

    return (dw, db)

# update weights
def update_weight(w, b, lr, dw, db):
    w_new = w - lr*dw
    b_new = b - lr*db

    return (w_new, b_new)
```

Computational Graph
(A different viewpoint)

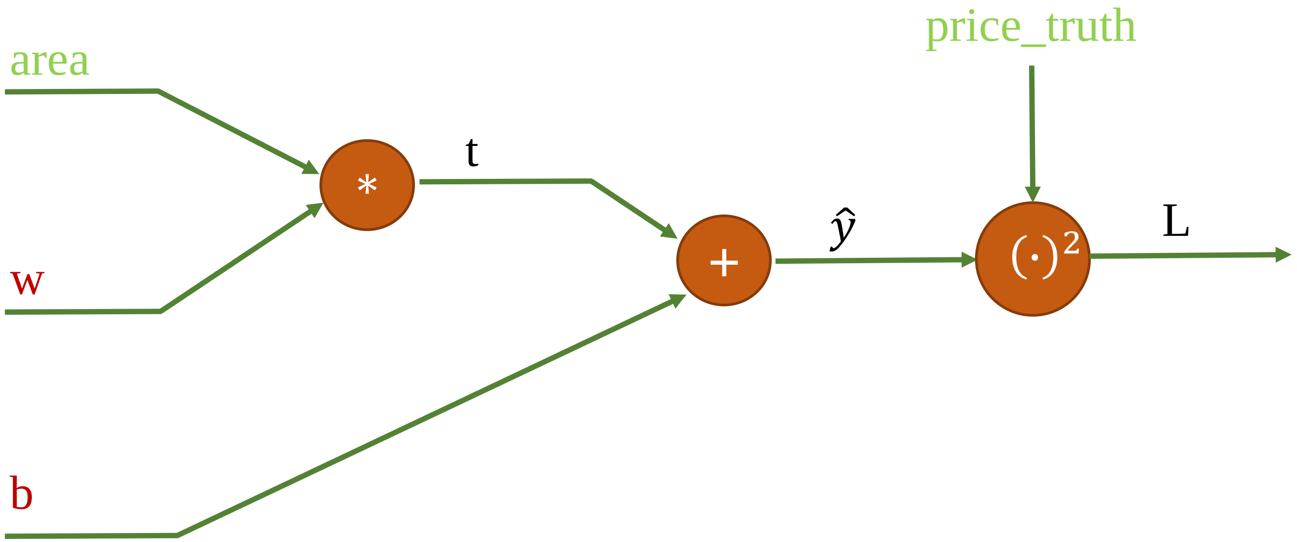
Computational graph

❖ House price prediction

❖ One-sample training

$$price = w * area + b$$

$$t = w * area$$

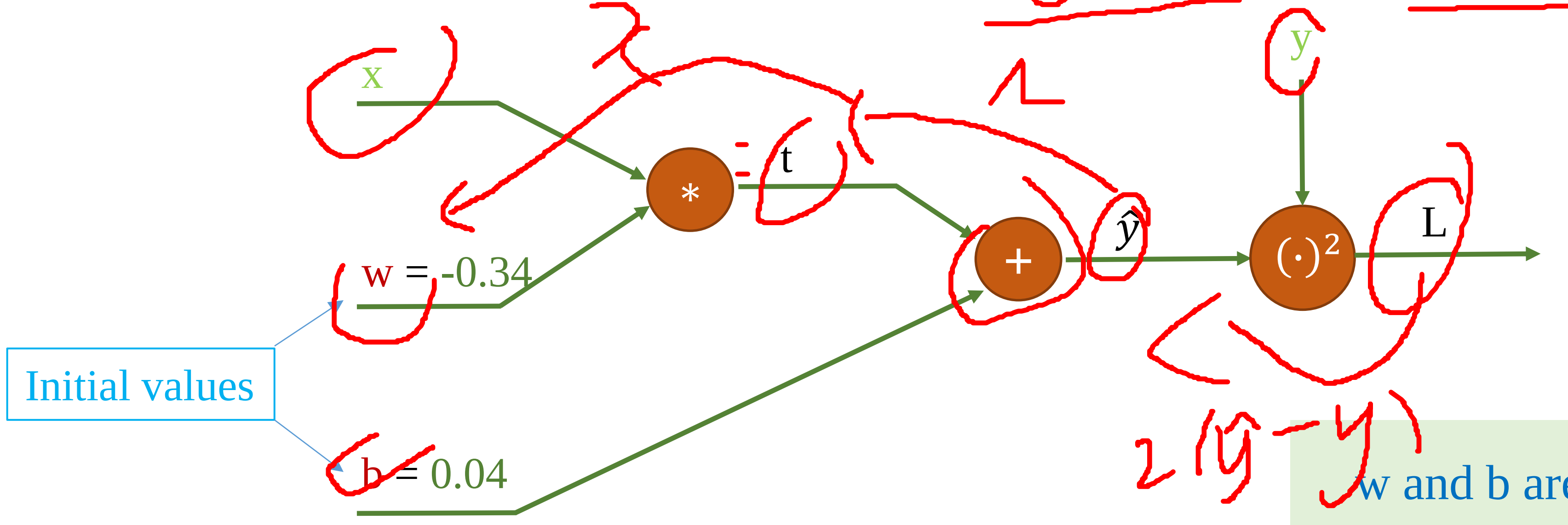


Computational graph

- **House price prediction**
  - **One-sample training**

$\hat{y} = wx + b$

$t = wx$ $\qquad$ $\hat{y} = t + b$ $\qquad$ $L = (\hat{y} - y)^2$



$\frac{dL}{dw} = 2x(\hat{y} - y)$

w and b are usually set by random values

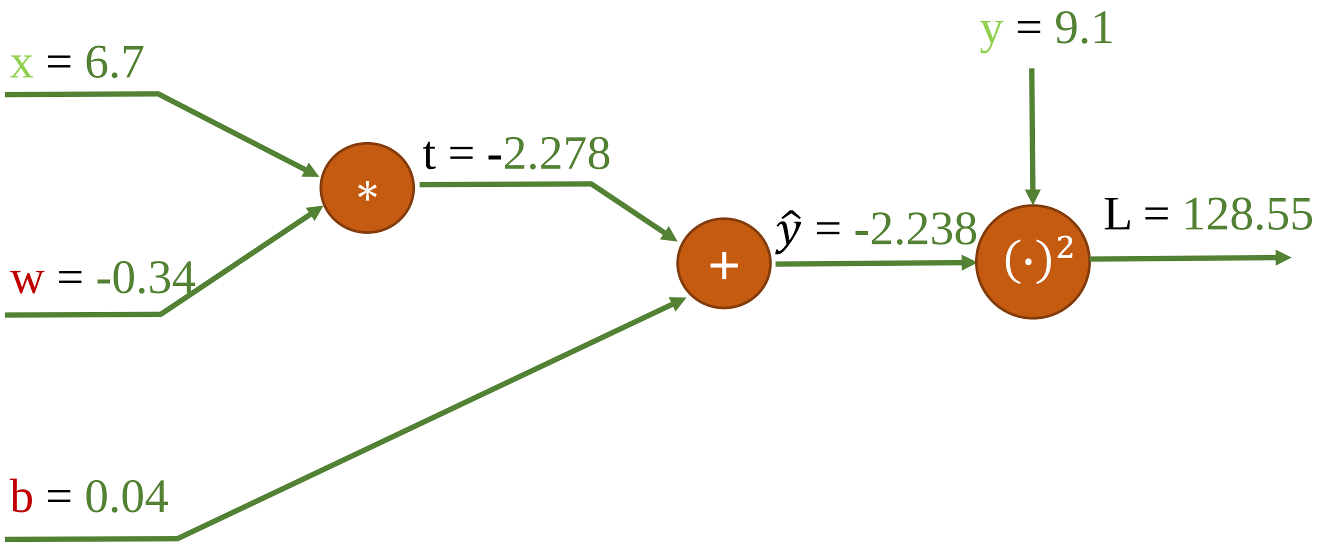
For example, $N(0, \sigma)$ where σ is small

Computational graph

❖ House price prediction

❖ One-sample training

$t = wx$ $\quad\quad$ $\hat{y} = t + b$ $\quad\quad$ $L = (\hat{y} - y)^2$



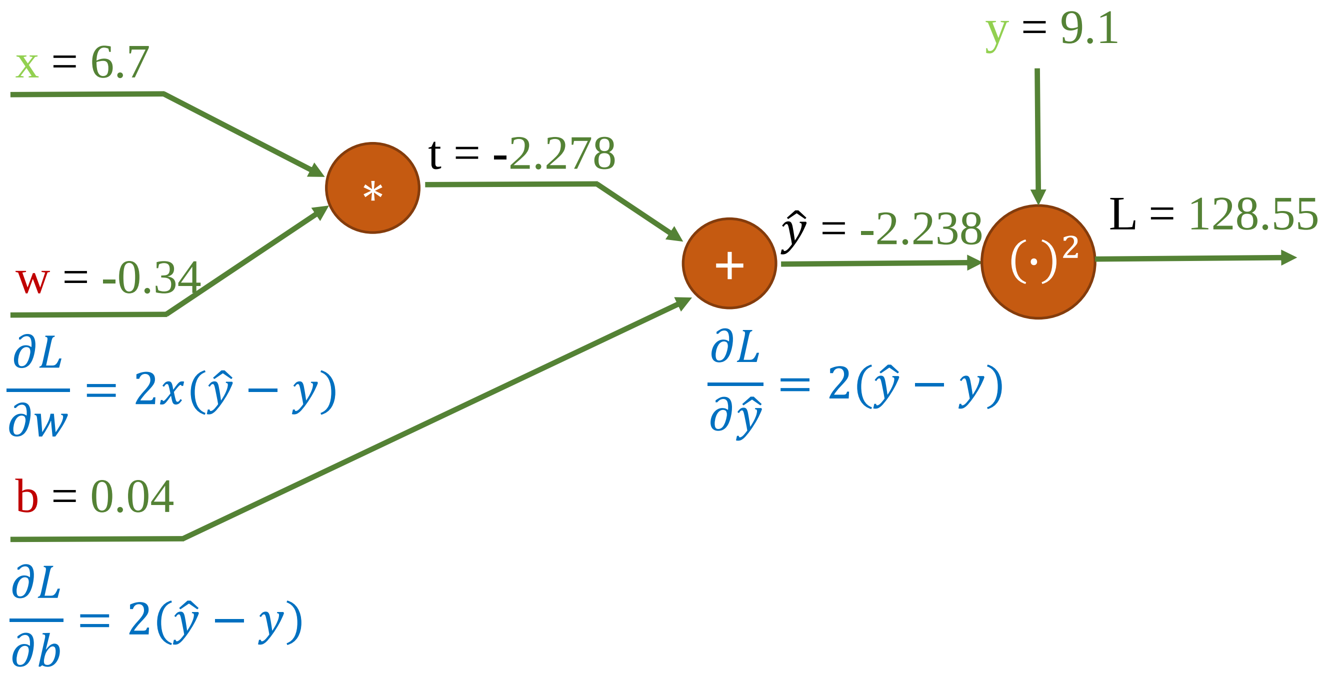
Feature	Label
area	price
6.7	9.1
4.6	5.9
3.5	4.6
5.5	6.7

Computational graph

❖ House price prediction

❖ One-sample training

$t = wx \qquad \hat{y} = t + b \qquad L = (\hat{y} - y)^2$

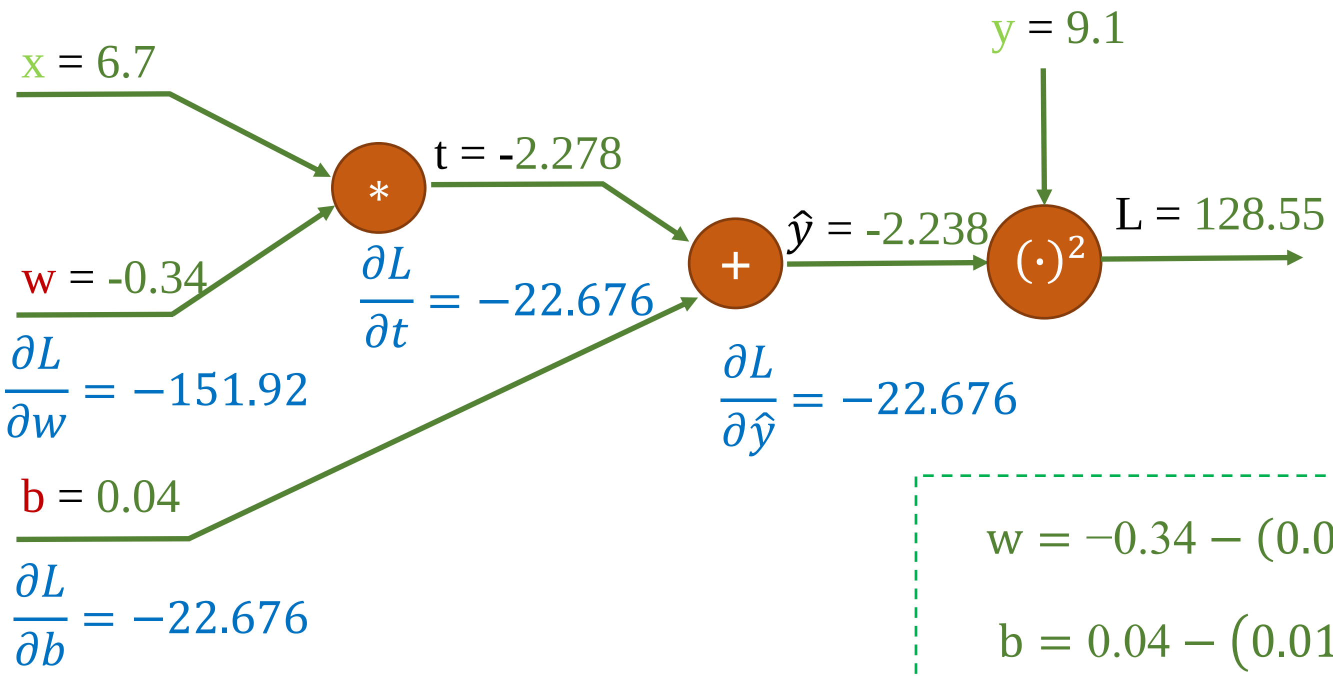


Computational graph

❖ House price prediction

❖ One-sample training

$$t = wx \qquad \hat{y} = t + b \qquad L = (\hat{y} - y)^2$$



Update w and b

$$w = w - \eta * \frac{\partial L}{\partial w}$$

$$b = b - \eta * \frac{\partial L}{\partial b}$$

Learning rate $\eta = 0.01$

$$w = -0.34 - (0.01 * (-151.9)) = 1.179$$

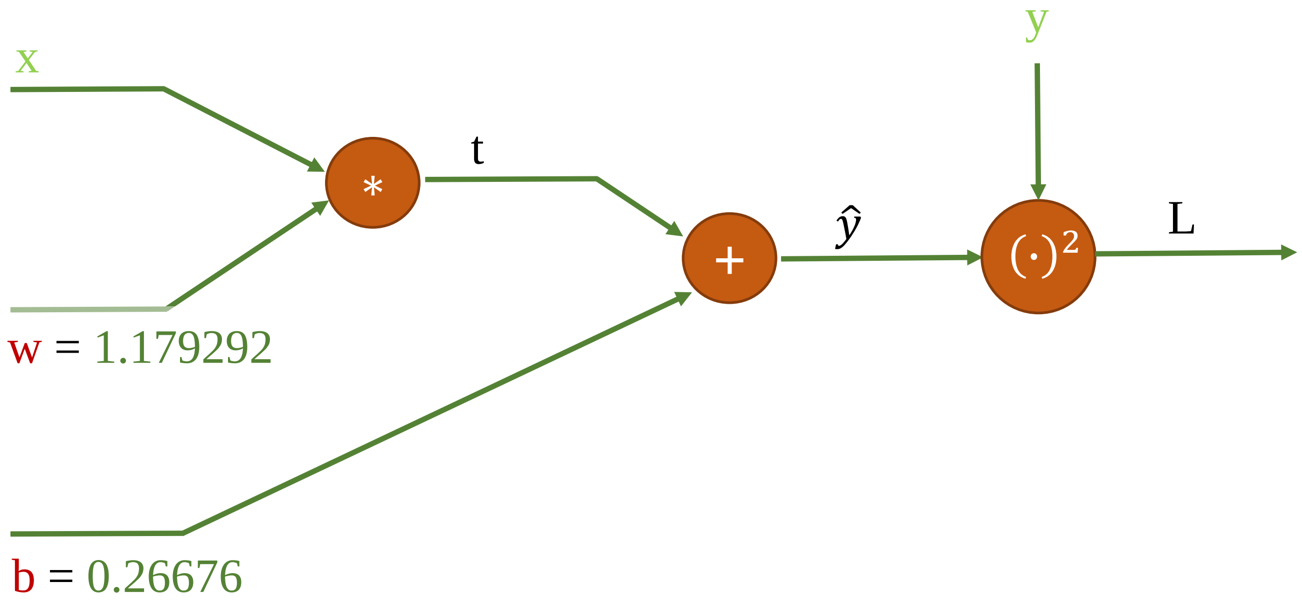
$$b = 0.04 - \big(0.01 * (-22.67)\big) = 0.266$$

Computational graph

❖ House price prediction

❖ One-sample training

$t = wx$ $\quad\quad \hat{y} = t + b \quad\quad L = (\hat{y} - y)^2$



Feature	Label
area	price
6.7	9.1
4.6	5.9
3.5	4.6
5.5	6.7

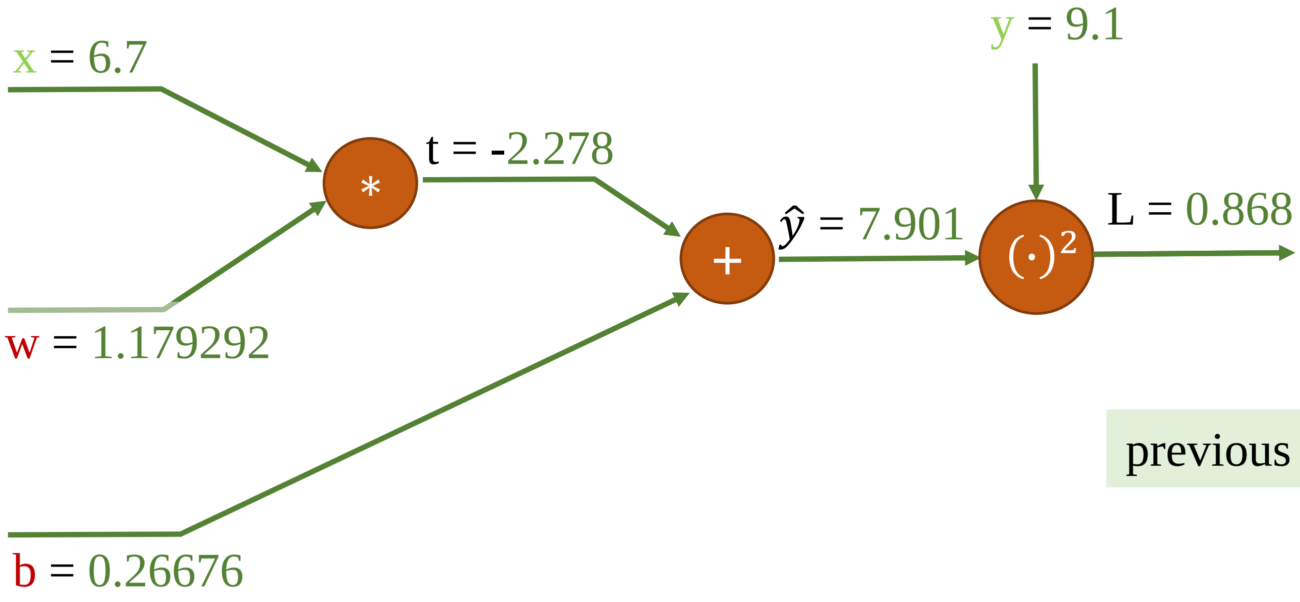
Computational graph

❖ House price prediction

❖ One-sample training

Feature	Label
area	price
6.7	9.1
4.6	5.9
3.5	4.6
5.5	6.7

$t = wx$ $\hat{y} = t + b$ $L = (\hat{y} - y)^2$



previous L = 128.55

Updated a and b values help to reduce the L value

Implementation

❖ One-sample training

Feature	Label
area	price
6.7	9.1
4.6	5.9
3.5	4.6
5.5	6.7

column index=0   column index=1

```python
# data preparation
import numpy as np
import matplotlib.pyplot as plt

def get_column(data, index):
    result = [row[index] for row in data]
    return result

data = np.genfromtxt('data.csv',
                     delimiter=',').tolist()

x_data = get_column(data, 0)
y_data = get_column(data, 1)
N = len(x_data)

print(f'areas: {x_data}')
print(f'prices: {y_data}')
print(f'data_size: {N}')
```

```
areas: [6.7, 4.6, 3.5, 5.5]
prices: [9.1, 5.9, 4.6, 6.7]
data_size: 4
```

Implementation

❖ House price prediction

❖ One-sample training

1) Pick a sample (x, y) from training data

2) Compute the output $\hat{y}$

$$\hat{y} = wx + b$$

3) Compute loss

$$L = (\hat{y} - y)^2$$

4) Compute derivative

$$\frac{\partial L}{\partial w} = 2x(\hat{y} - y) \qquad \frac{\partial L}{\partial b} = 2(\hat{y} - y)$$

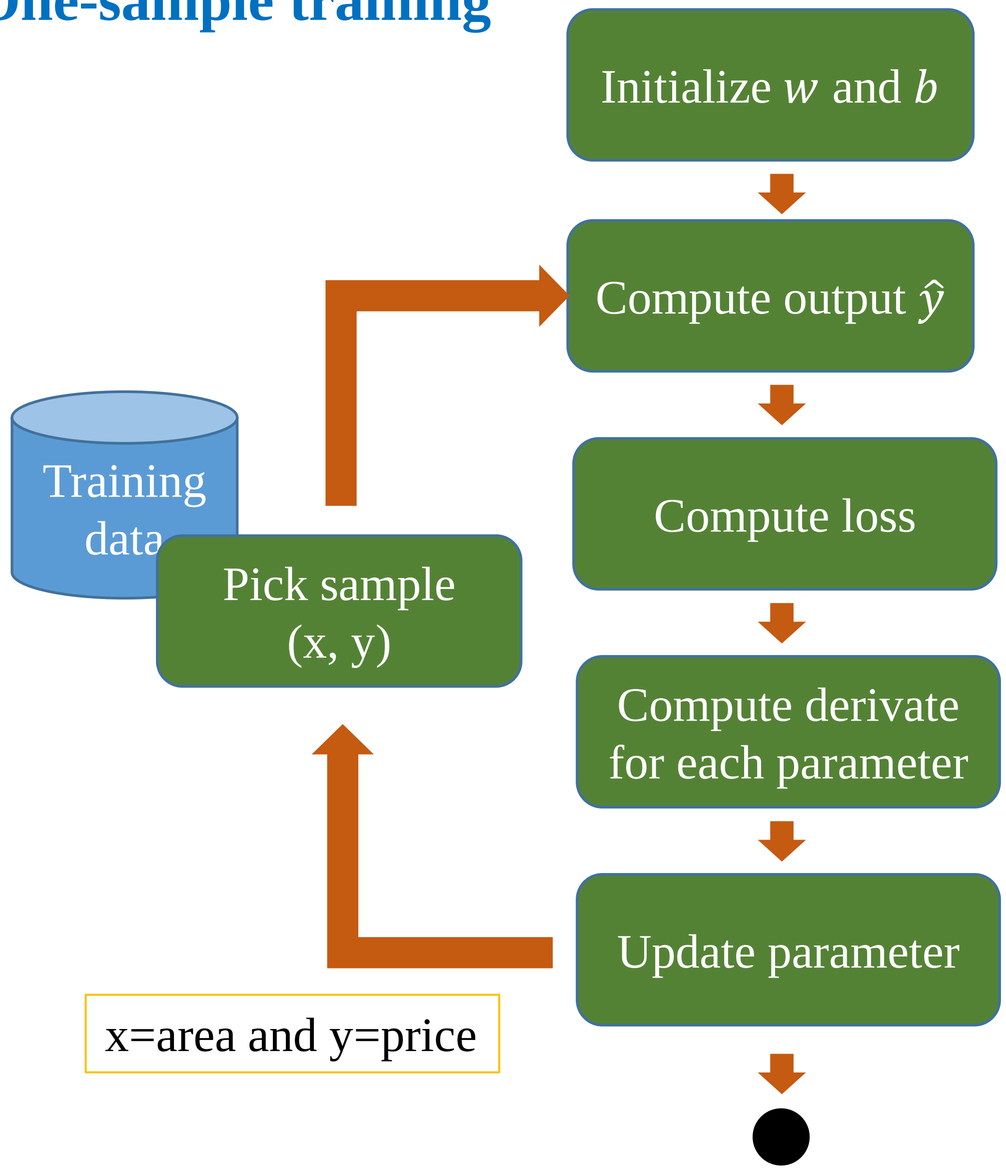
5) Update parameters

$$w = w - \eta \frac{\partial L}{\partial w} \qquad b = b - \eta \frac{\partial L}{\partial b}$$

```python
# forward
def predict(x, w, b):
    return x*w + b

# compute gradient
def gradient(y_hat, y, x):
    dw = 2*x*(y_hat-y)
    db = 2*(y_hat-y)

    return (dw, db)

# update weights
def update_weight(w, b, lr, dw, db):
    w_new = w - lr*dw
    b_new = b - lr*db

    return (w_new, b_new)
```

Implementation

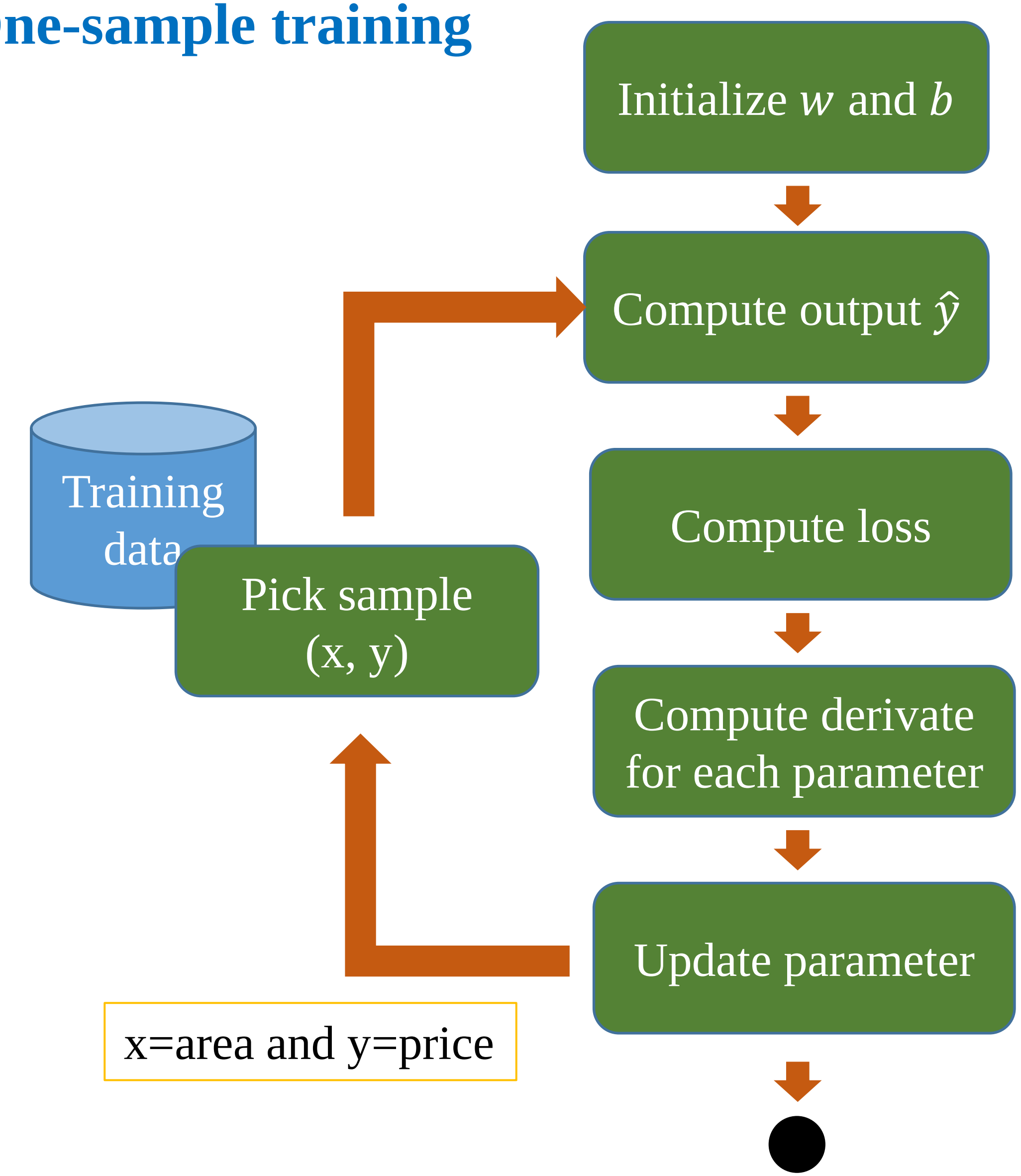
❖ One-sample training




```python
# init weights
b = 0.04
w = -0.34
lr = 0.01

# how long
epoch_max = 10

for epoch in range(epoch_max):
    for i in range(data_size):
        # get a sample
        # ...

        # predict y_hat
        # ...

        # compute loss
        # ...

        # compute gradient
        # ...

        # update weights
        # ...

```

Implementation

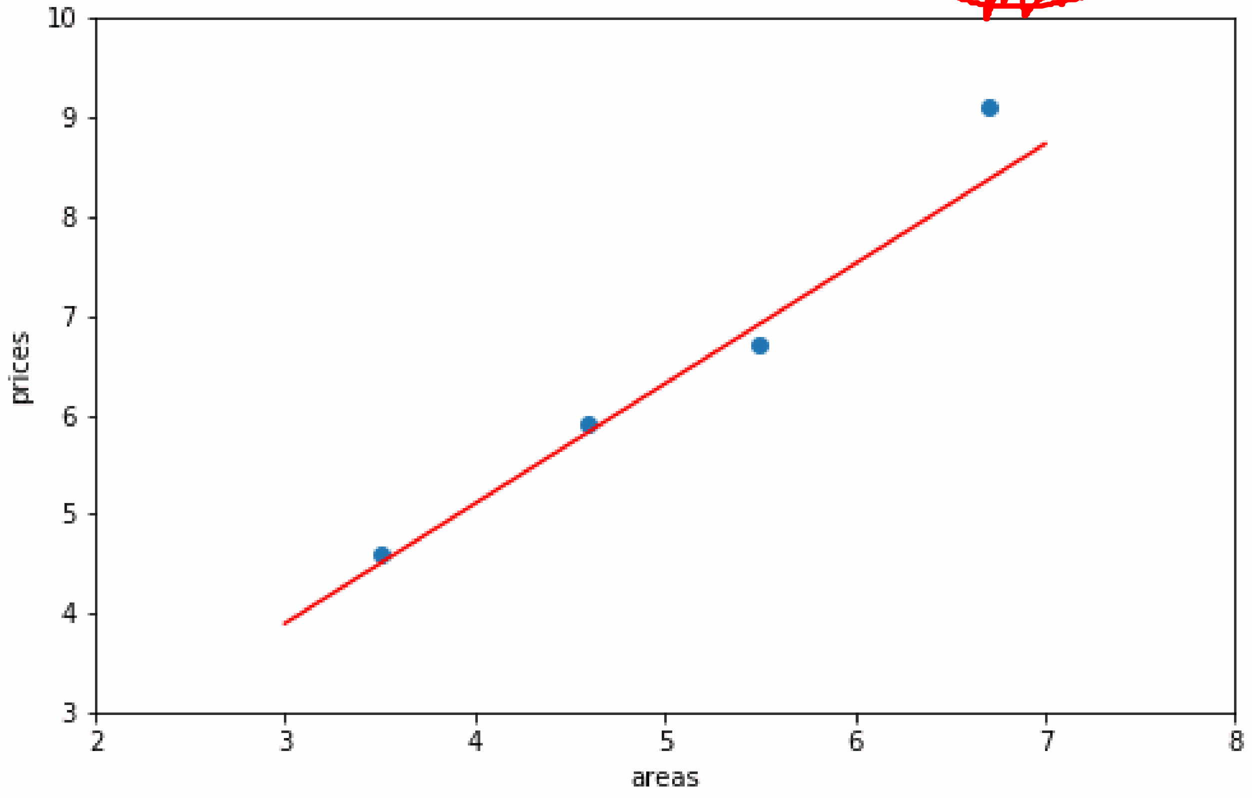
❖ One-sample training

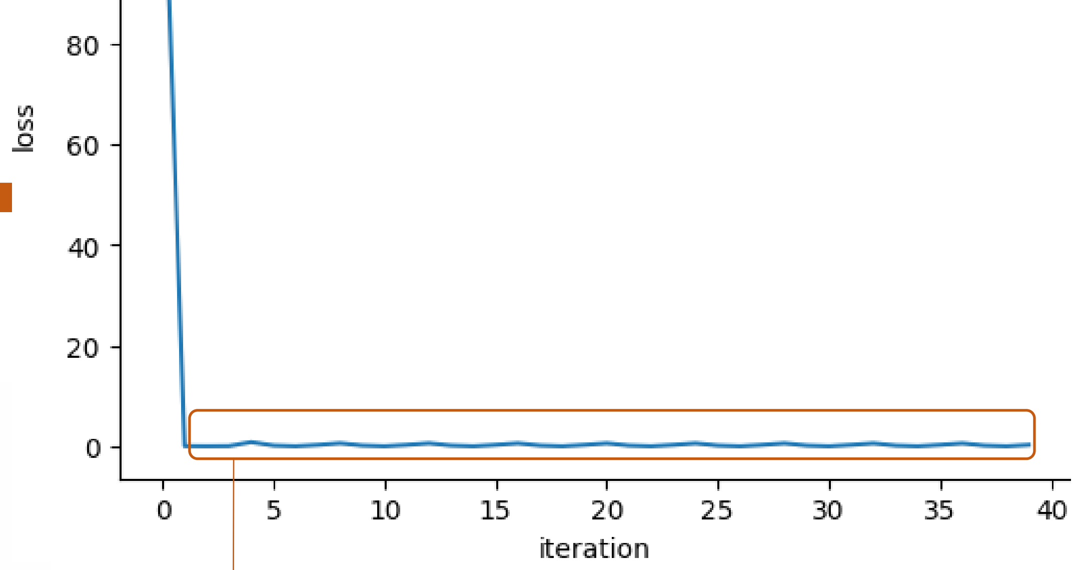


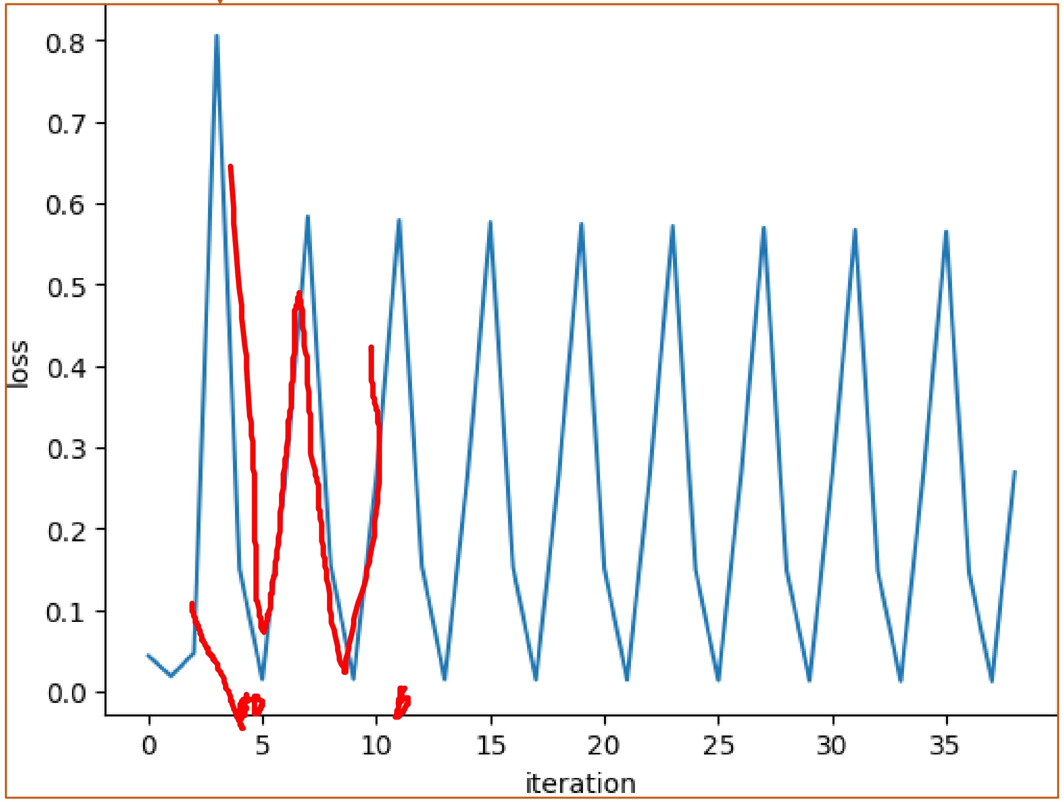

```python
# init weights
b = 0.04
w = -0.34
lr = 0.01

# how long
epoch_max = 10
data_size = 4

for epoch in range(epoch_max):
    for i in range(data_size):
        # get a sample
        x = areas[i]
        y = prices[i]

        # predict y_hat
        y_hat = predict(x, w, b)

        # compute loss
        loss = (y_hat-y)*(y_hat-y)

        # compute gradient
        (dw, db) = gradient(y_hat, y, x)

        # update weights
        (w, b) = update_weight(w, b, lr, dw, db)
```

Implementation

- **House price prediction**
  - **One-sample training**







Model training

Quiz 1: Construct for the following data

Features		Label
Radio	Newspaper	Sales
37.8	69.2	22.1
39.3	45.1	10.4
45.9	69.3	12
41.3	58.5	16.5
10.8	58.4	17.9

Outline

7000 1250

SECTION 1
Linear Regression

SECTION 2
Mini-batch Training

SECTION 3
Batch Training

SECTION 4
Loss Functions

1) Pick m samples $(x^{(i)}, y^{(i)})$ from training data

2) Compute output $\hat{y}^{(i)}$

$$\hat{y}^{(i)} = wx^{(i)} + b \qquad \text{for } 0 \leq i < m$$

3) Compute loss

$$L^{(i)} = (\hat{y}^{(i)} - y^{(i)})^2 \quad \text{for } 0 \leq i < m$$

4) Compute derivatives

$$\frac{\partial L^{(i)}}{\partial w} = 2x^{(i)}(\hat{y}^{(i)} - y^{(i)})$$

$$\frac{\partial L^{(i)}}{\partial b} = 2(\hat{y}^{(i)} - y^{(i)}) \qquad \text{for } 0 \leq i < m$$

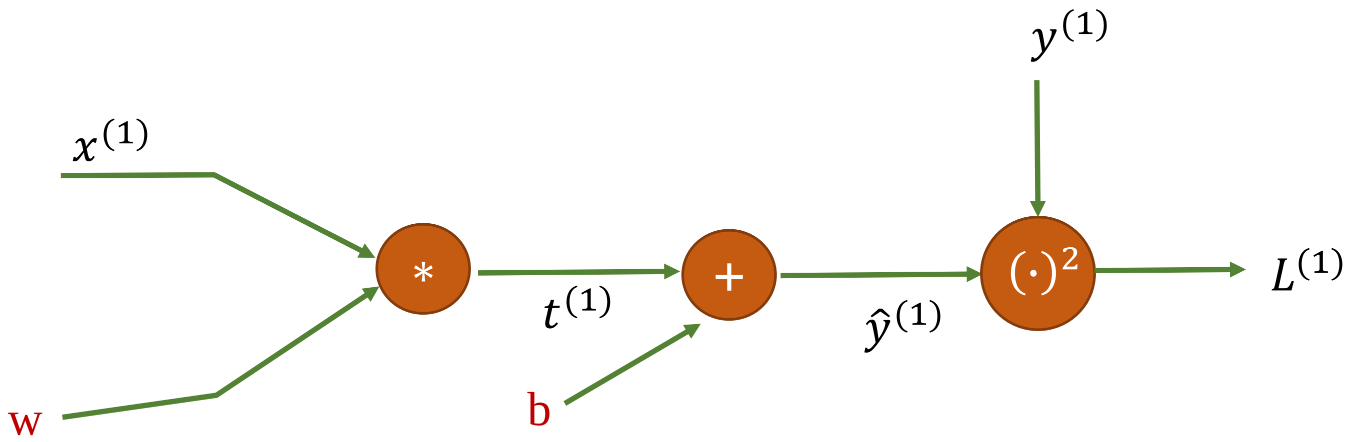
5) Update

$$w = w - \eta \frac{\sum_i \frac{\partial L^{(i)}}{\partial w}}{m} \qquad b = b - \eta \frac{\sum_i \frac{\partial L^{(i)}}{\partial b}}{m}$$

Computational graph

❖ **Compute derivate for w and b**

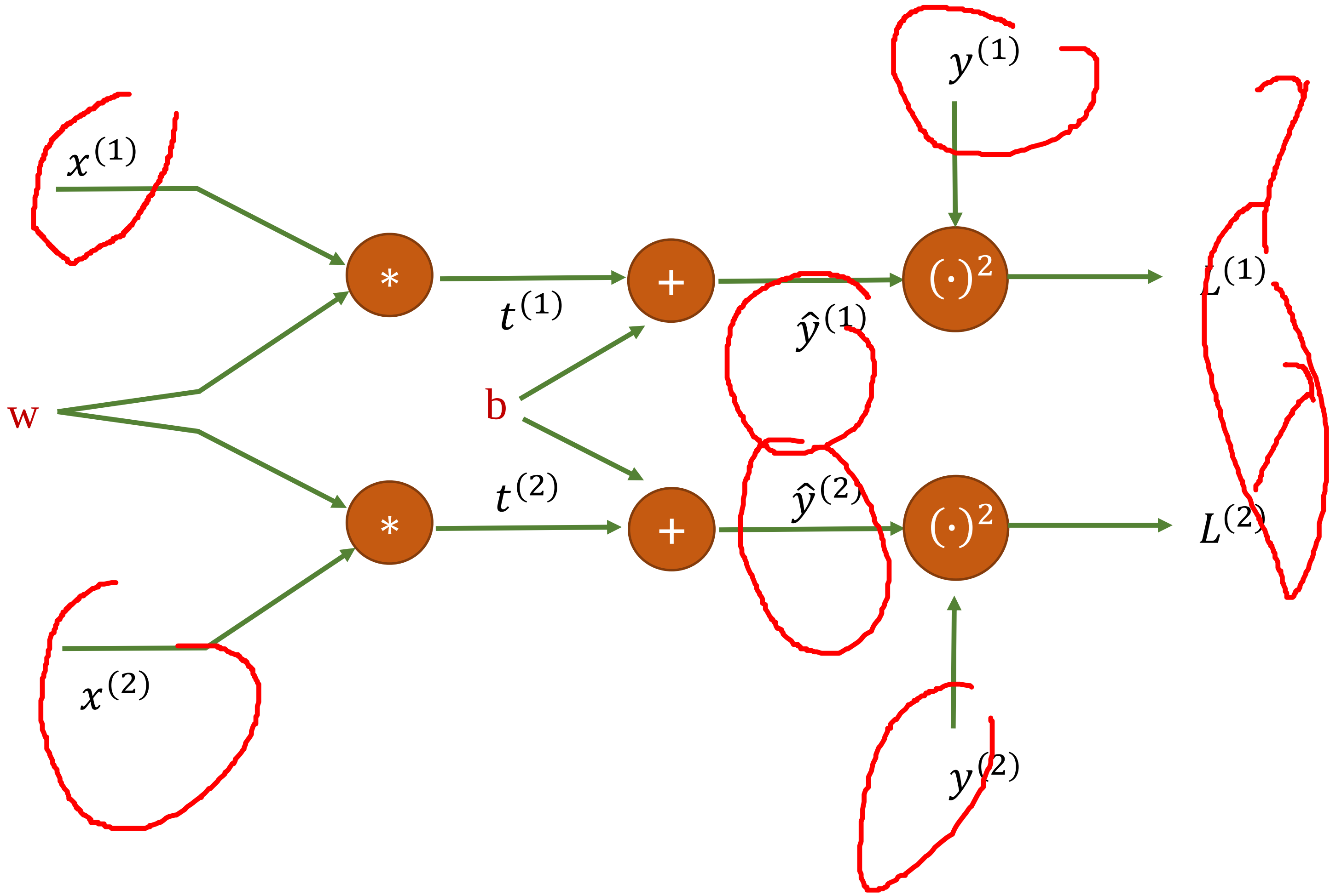
mini-batch m = 2



Computational graph

❖ Compute derivate for w and b

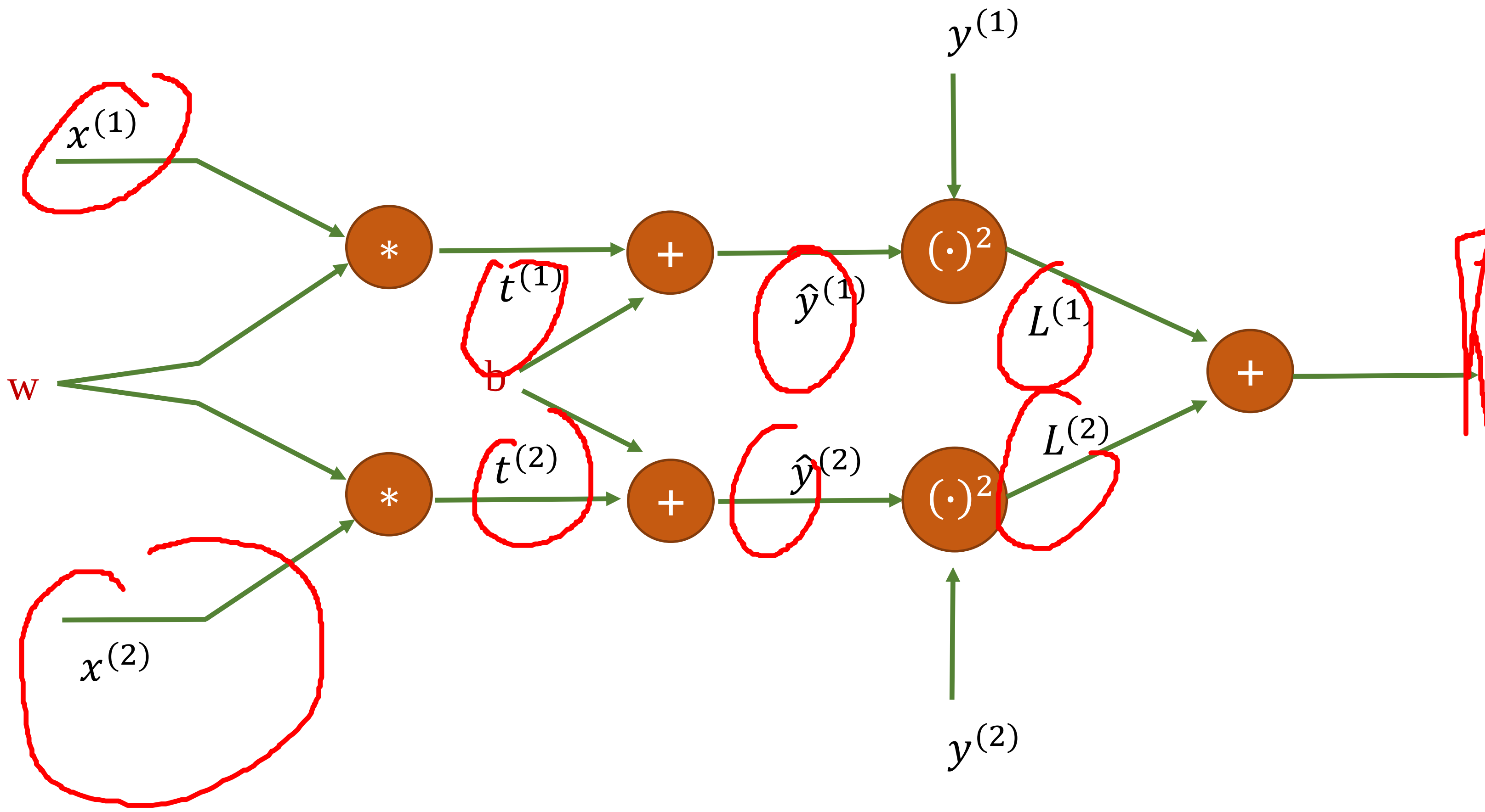
mini-batch m = 2



Computational graph

❖ Compute derivate for w and b

mini-batch m = 2

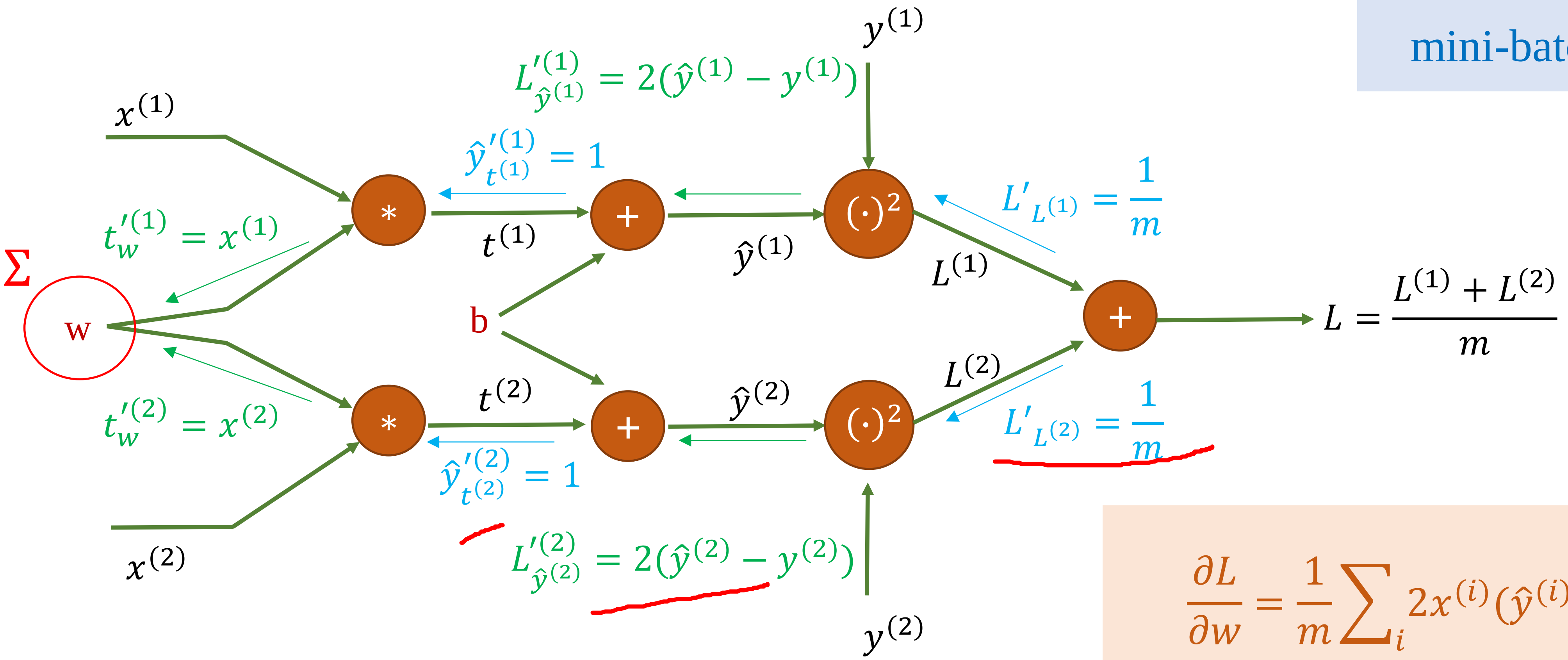


$$L = \frac{L^{(1)} + L^{(2)}}{m}$$

Computational graph

❖ Compute derivate for w and b

mini-batch m = 2

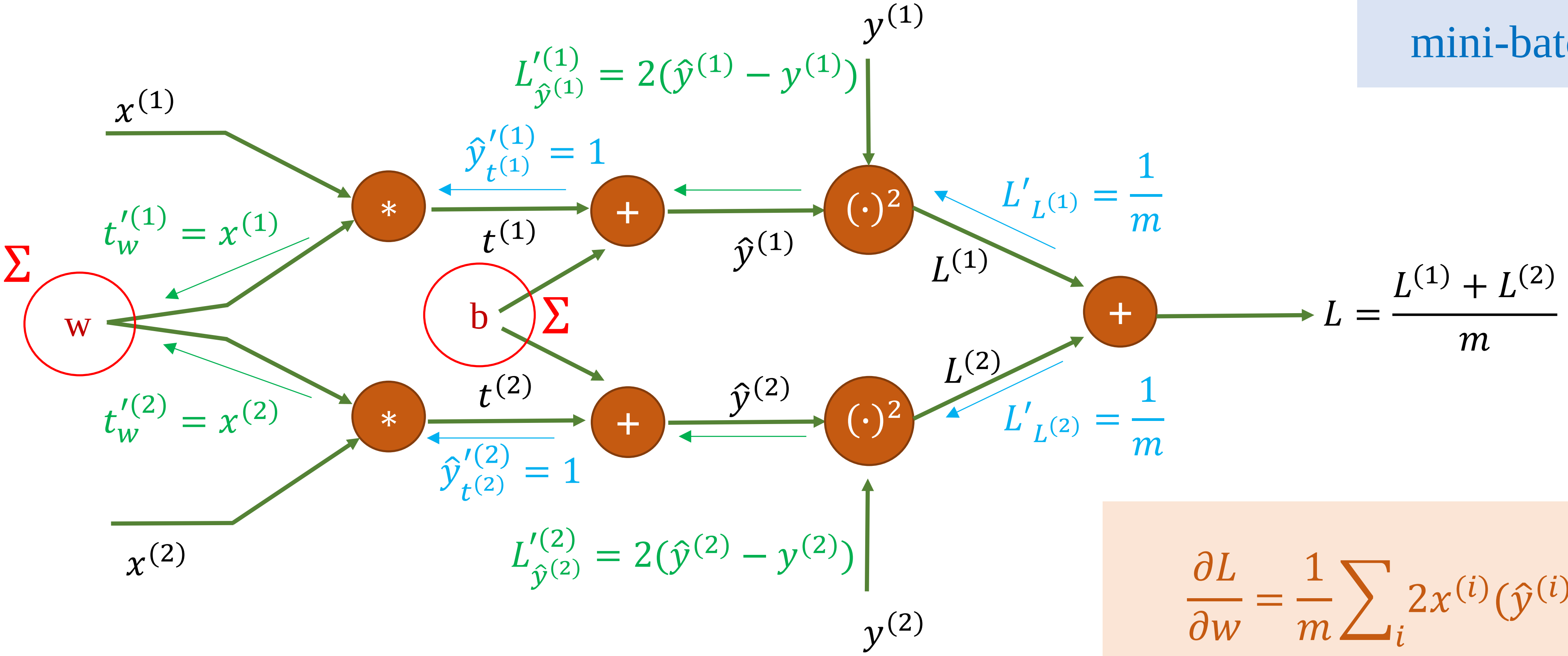


$$\frac{\partial L}{\partial w} = \frac{1}{m}\sum_i 2x^{(i)}(\hat{y}^{(i)} - y^{(i)})$$

Computational graph

❖ Compute derivate for w and b

mini-batch m = 2



$$\frac{\partial L}{\partial w} = \frac{1}{m}\sum_i 2x^{(i)}(\hat{y}^{(i)} - y^{(i)})$$

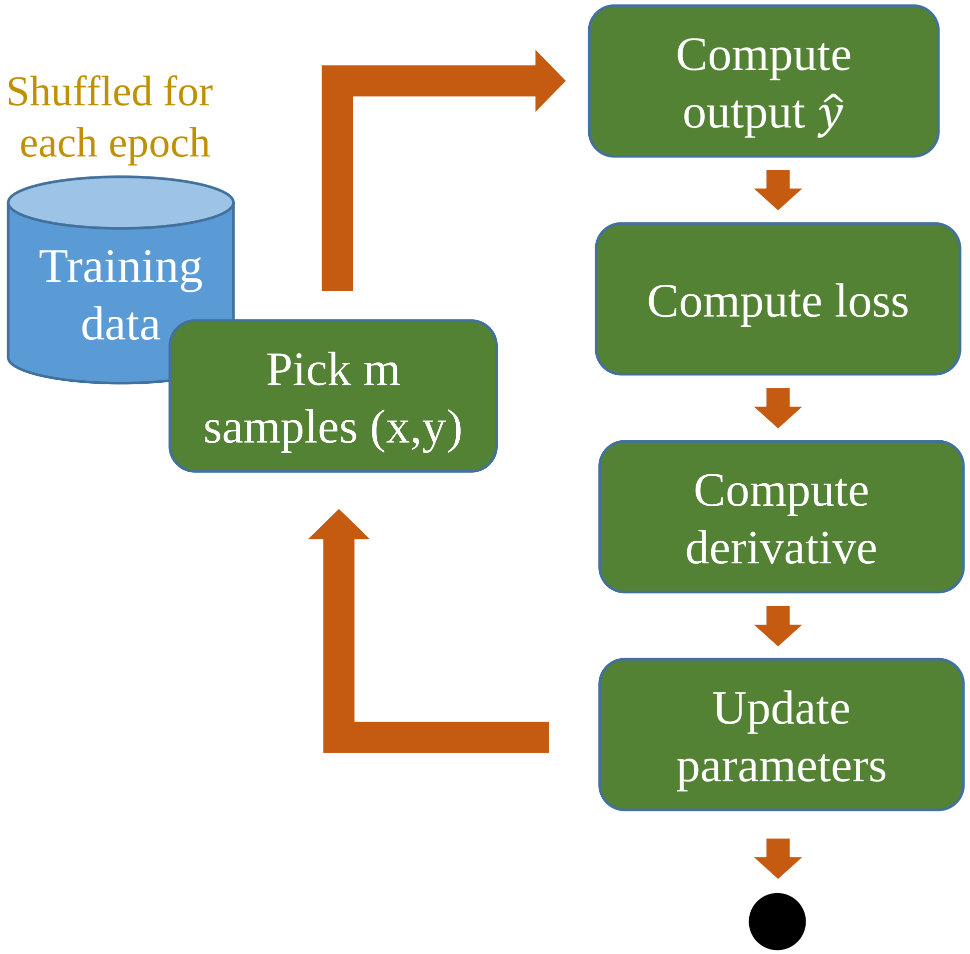
$$\frac{\partial L}{\partial b} = \frac{1}{m}\sum_i 2(\hat{y}^{(i)} - y^{(i)})$$

Computational graph

simplified

❖ House price prediction

❖ m-sample training (1<m<N)



1) Pick m samples $(x^{(i)}, y^{(i)})$ from training data

2) Compute output $\hat{y}^{(i)}$

$$\hat{y}^{(i)} = wx^{(i)} + b \qquad \text{for } 0 \leq i < m$$

3) Compute loss

$$L = \frac{1}{m}\sum_i (\hat{y}^{(i)} - y^{(i)})^2$$

4) Compute derivatives

$$\frac{\partial L^{(i)}}{\partial w} = 2x^{(i)}(\hat{y}^{(i)} - y^{(i)})$$

$$\frac{\partial L^{(i)}}{\partial b} = 2(\hat{y}^{(i)} - y^{(i)}) \qquad \text{for } 0 \leq i < m$$

5) Update

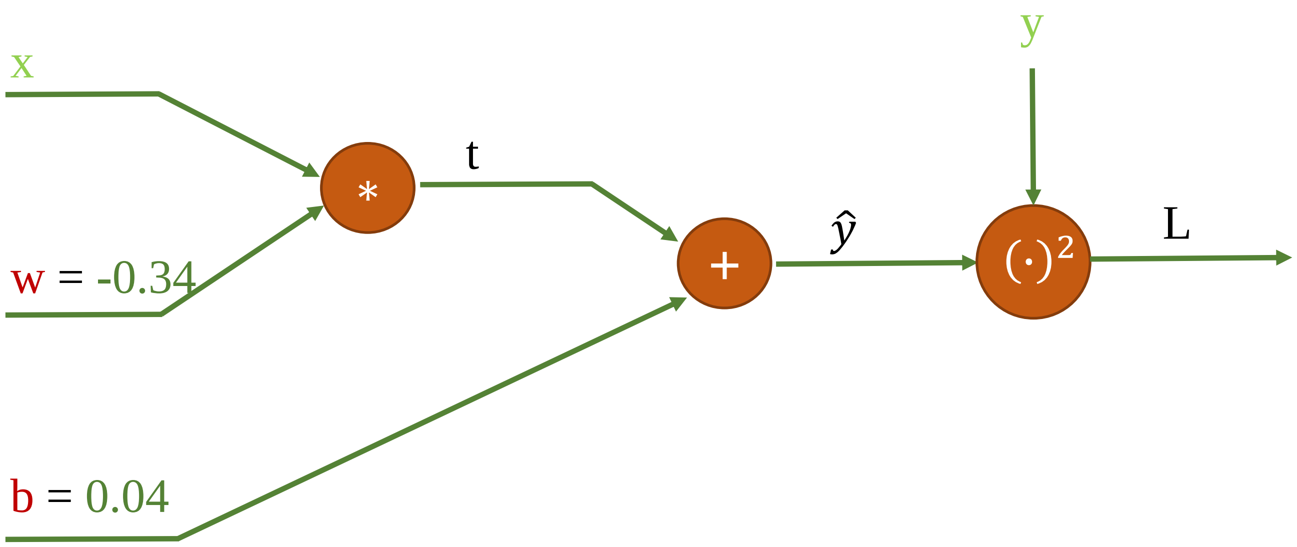
$$w = w - \eta \frac{\sum_i \frac{\partial L^{(i)}}{\partial w}}{m} \qquad b = b - \eta \frac{\sum_i \frac{\partial L^{(i)}}{\partial b}}{m}$$

Computational graph

❖ House price prediction

❖ m-sample training (1<m<N)

$$t = wx \qquad \hat{y} = t + b \qquad L = (\hat{y} - y)^2$$

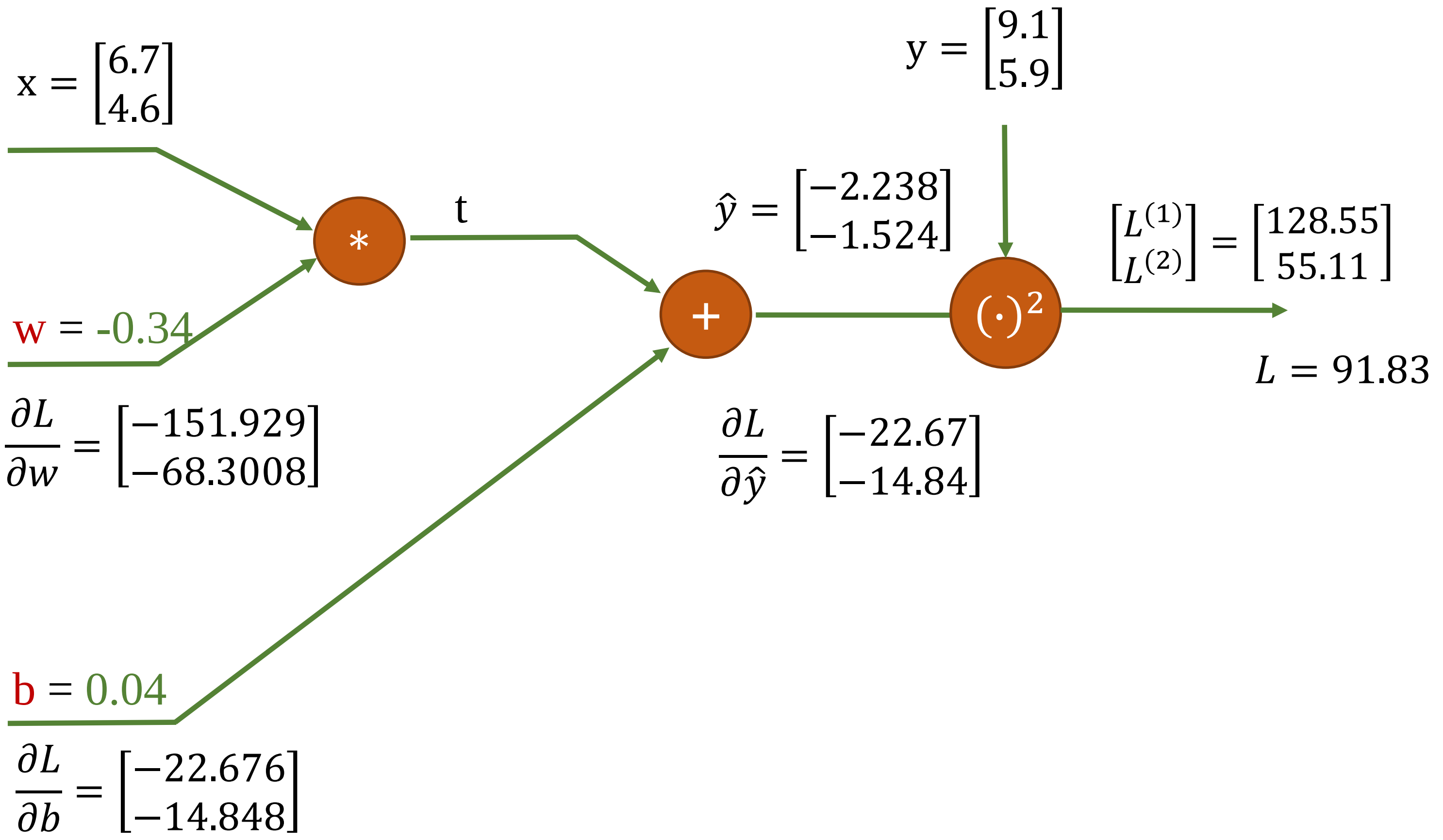


Computational graph

- House price prediction
  - m-sample training (1<m<N)

Feature	Label
area	price
6.7	9.1
4.6	5.9
3.5	4.6
5.5	6.7

m = 2



$$\frac{sum(L'_w)}{m} = -110.115$$

$$\frac{sum(L'_b)}{m} = -18.762$$

$$t = wx \qquad \hat{y} = t + b \qquad L = (\hat{y} - y)^2$$

Computational graph

❖ House price prediction

❖ m-sample training (1<m<N)

Feature	Label
area	price
6.7	9.1
4.6	5.9
3.5	4.6
5.5	6.7

m = 2

Update w and b

$$w = w - \eta * \frac{\partial L}{\partial w}$$

$$b = b - \eta * \frac{\partial L}{\partial b}$$

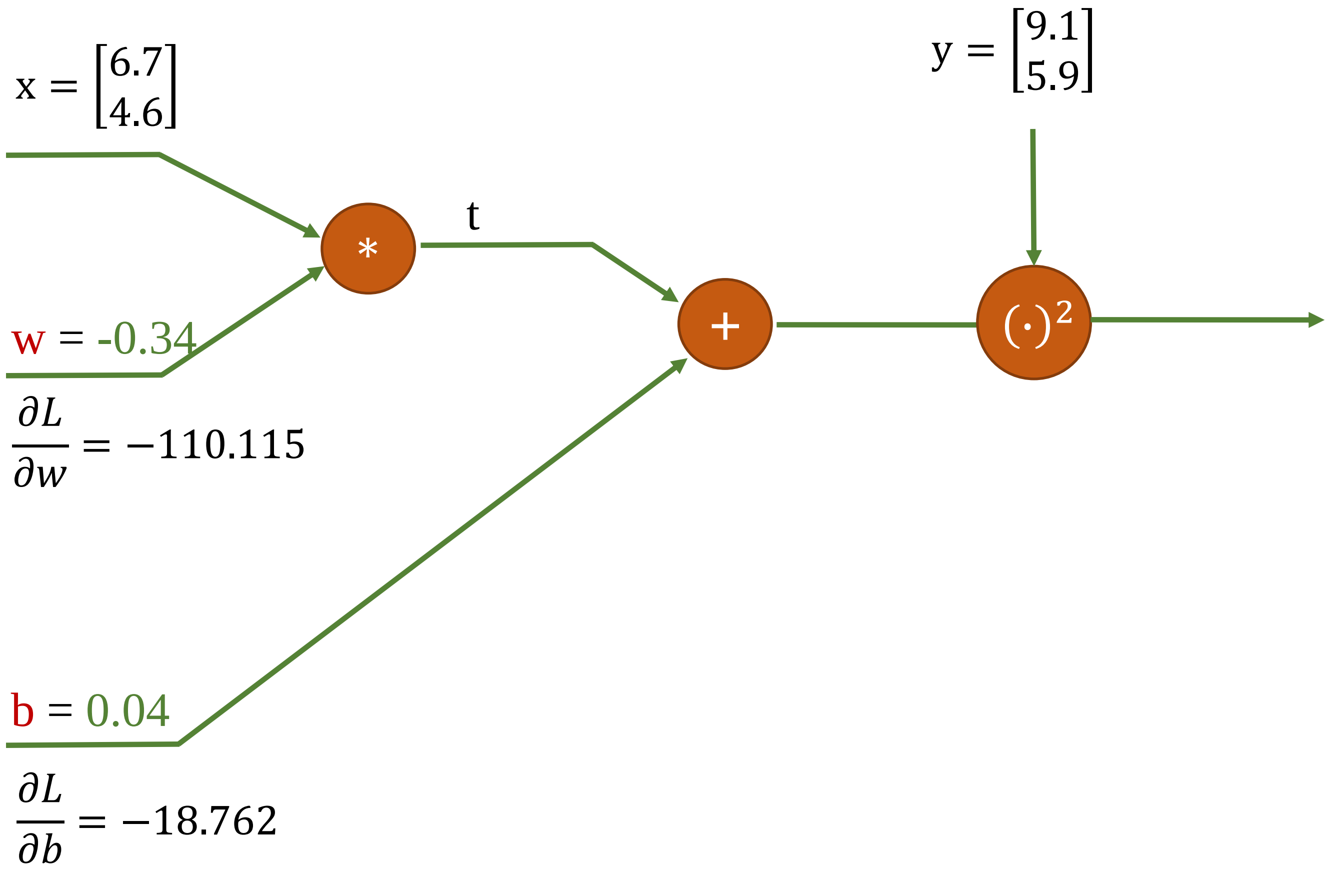
Learning rate $\eta = 0.01$

replace

$$w = -0.34 - (0.01 * (-110.115)) = \boxed{0.761}$$

replace

$$b = 0.04 - \big(0.01 * (-18.762)\big) = \boxed{0.227}$$

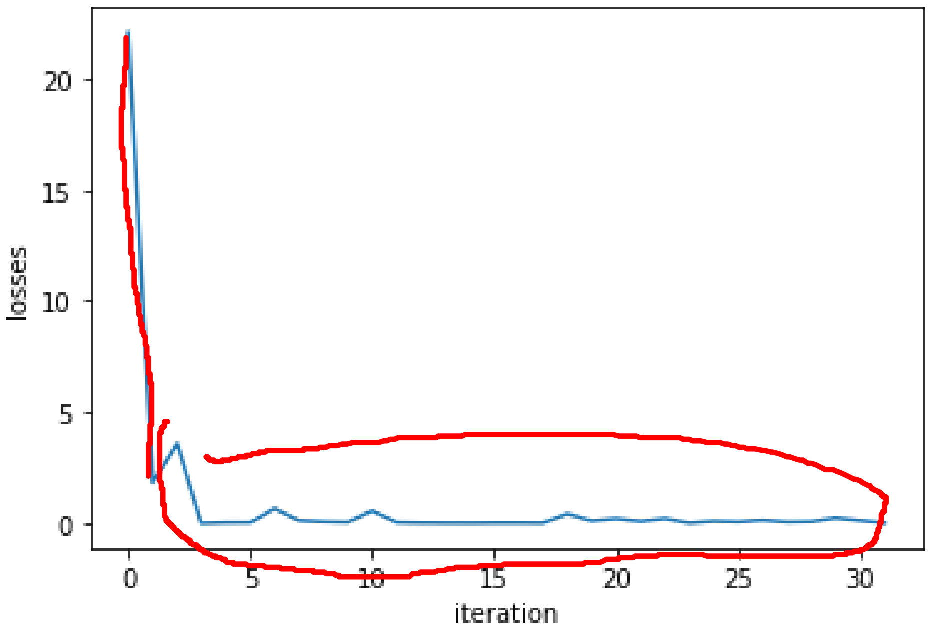


$t = wx$ $\quad \hat{y} = t + b$ $\quad L = (\hat{y} - y)^2$

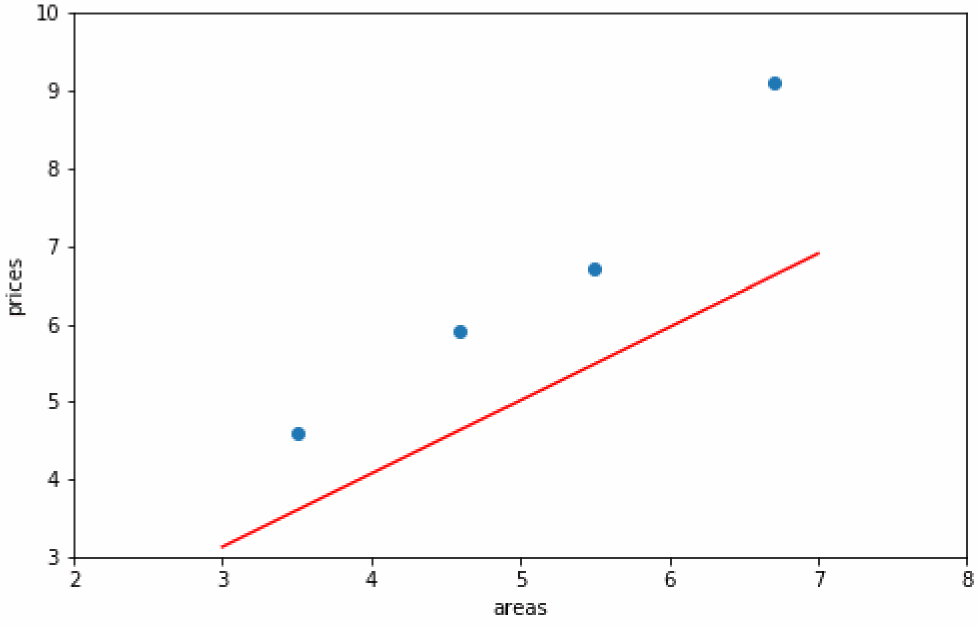
Computational graph

❖ House price prediction

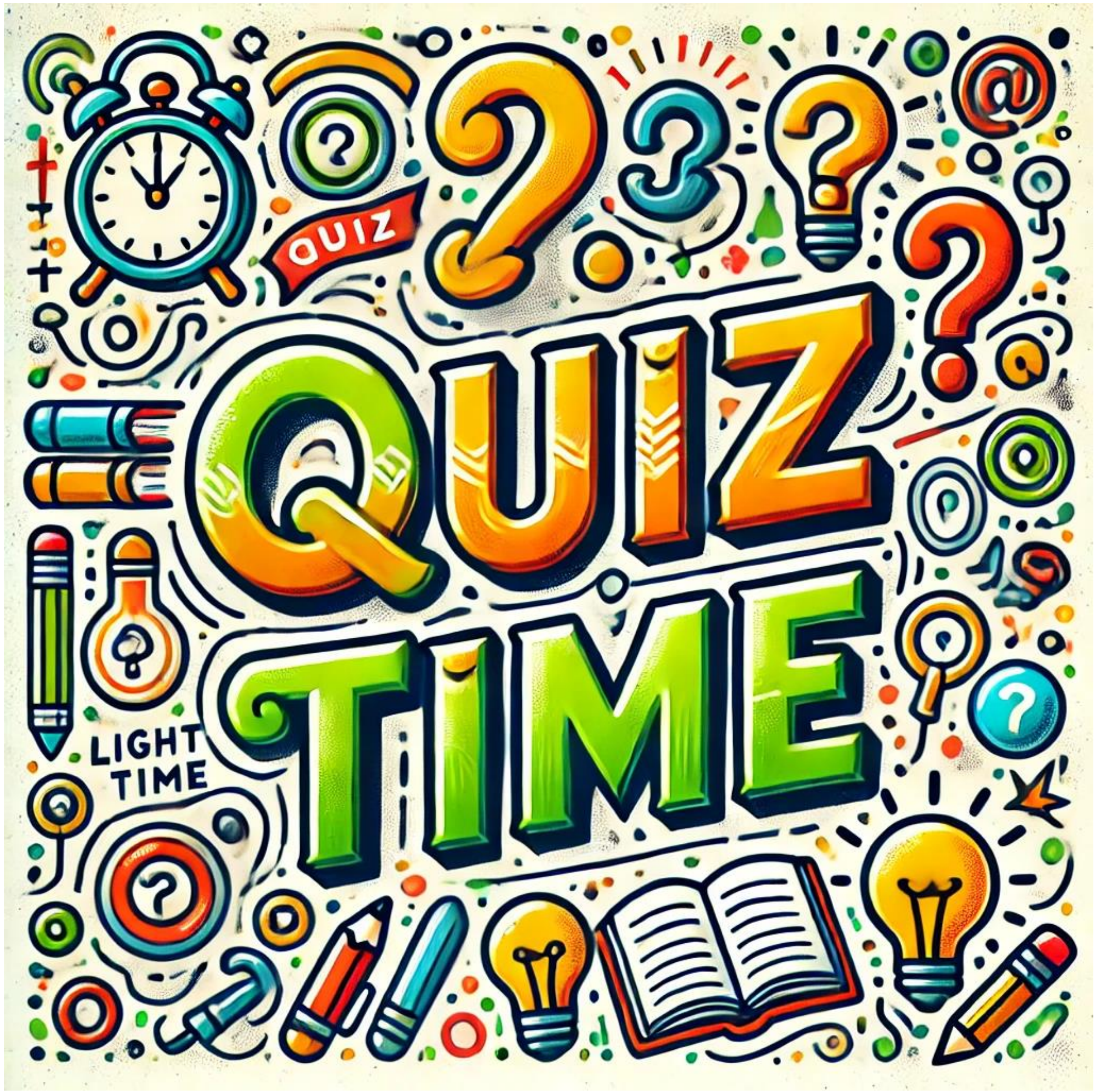
❖ m-sample training (1<m<N)



Losses for 30 iterations



Model updating for different iterations


QUIZ
QUIZ
TIME
LIGHT
TIME

Outline

14 5 → 1

SECTION 1
Linear Regression

SECTION 2
Mini-batch Training

SECTION 3
Batch Training

SECTION 4
Loss Functions

1) Pick all the N samples $(x^{(i)}, y^{(i)})$ from training data

2) Compute output $\hat{y}^{(i)}$

$$\hat{y}^{(i)} = wx^{(i)} + b \qquad \text{for } 0 \le i < N$$

3) Compute loss

$$L^{(i)} = (\hat{y}^{(i)} - y^{(i)})^2 \quad \text{for } 0 \le i < N$$

4) Compute derivatives

$$\frac{\partial L^{(i)}}{\partial w} = 2x^{(i)}(\hat{y}^{(i)} - y^{(i)})$$

$$\frac{\partial L^{(i)}}{\partial b} = 2(\hat{y}^{(i)} - y^{(i)}) \qquad \text{for } 0 \le i < N$$

5) Update

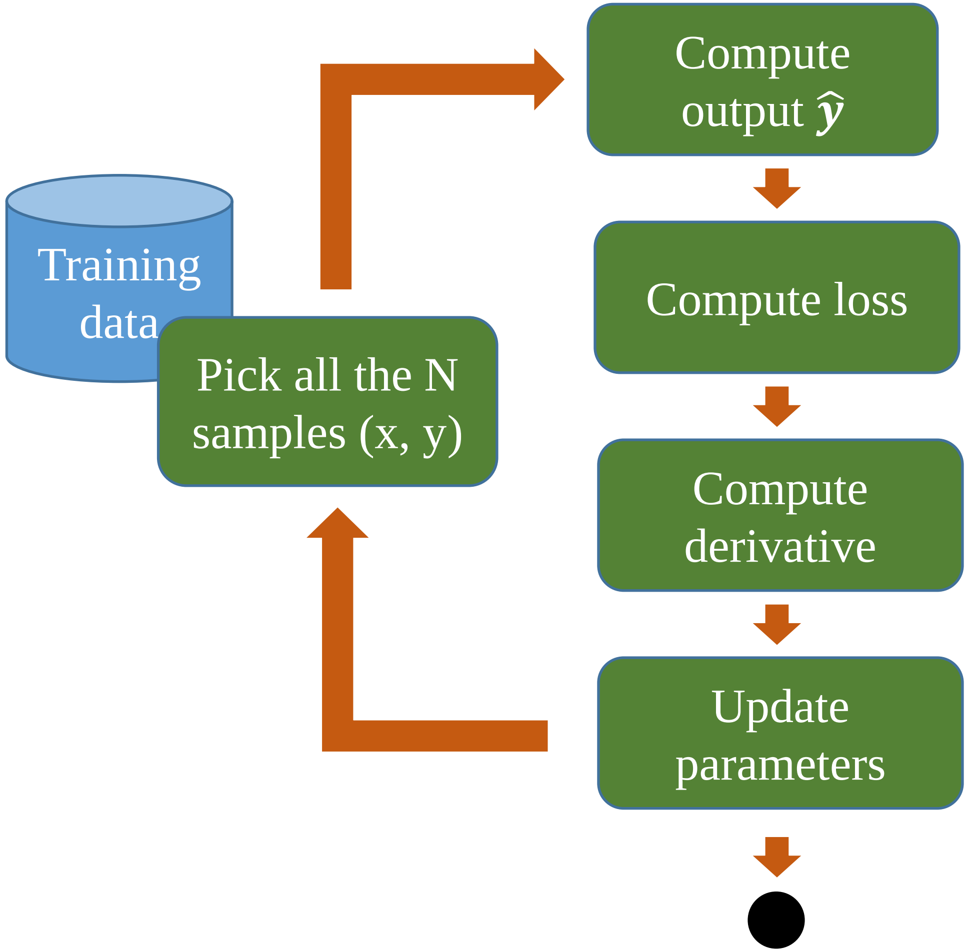
$$w = w - \eta \frac{\sum_i \frac{\partial L^{(i)}}{\partial w}}{N} \qquad b = b - \eta \frac{\sum_i \frac{\partial L^{(i)}}{\partial b}}{N}$$

Computational graph

simplified

❖ House price prediction

❖ N-sample training



1) Pick all the N samples $(x^{(i)}, y^{(i)})$ from training data

2) Compute output $\hat{y}^{(i)}$

$$\hat{y}^{(i)} = wx^{(i)} + b \qquad \text{for } 0 \leq i < N$$

3) Compute loss

$$L = \frac{1}{N}\sum_{i}(\hat{y}^{(i)} - y^{(i)})^2$$

4) Compute derivatives

$$\frac{\partial L^{(i)}}{\partial w} = 2x^{(i)}(\hat{y}^{(i)} - y^{(i)})$$

$$\text{for } 0 \leq i < N$$

$$\frac{\partial L^{(i)}}{\partial b} = 2(\hat{y}^{(i)} - y^{(i)})$$

5) Update

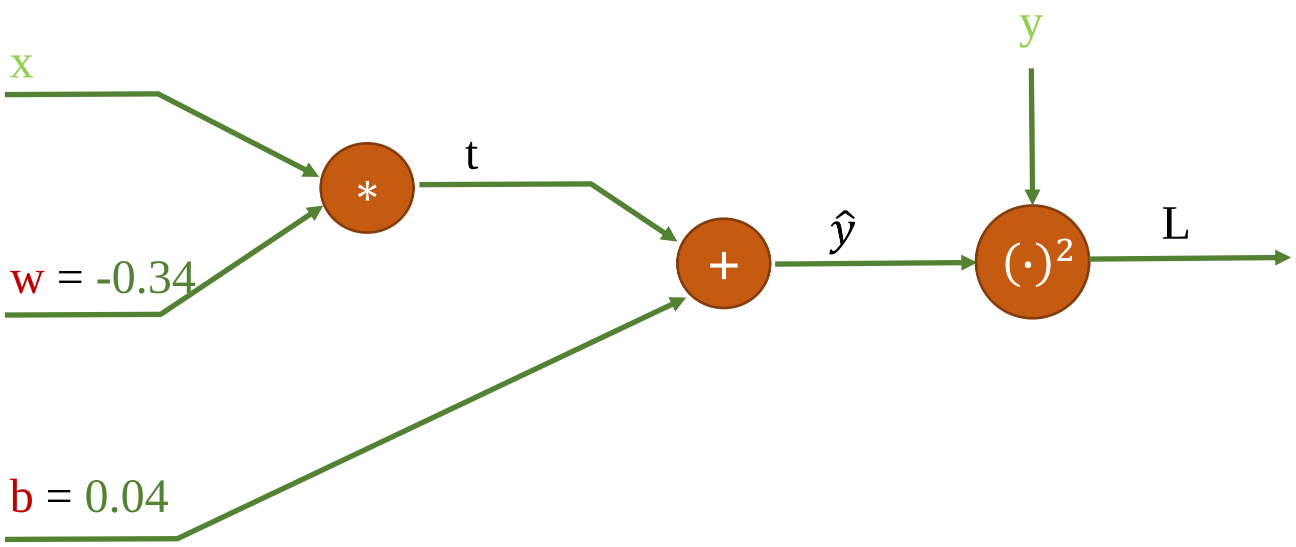
$$w = w - \eta\frac{\sum_i \frac{\partial L^{(i)}}{\partial w}}{N} \qquad b = b - \eta\frac{\sum_i \frac{\partial L^{(i)}}{\partial b}}{N}$$

Computational graph

❖ House price prediction

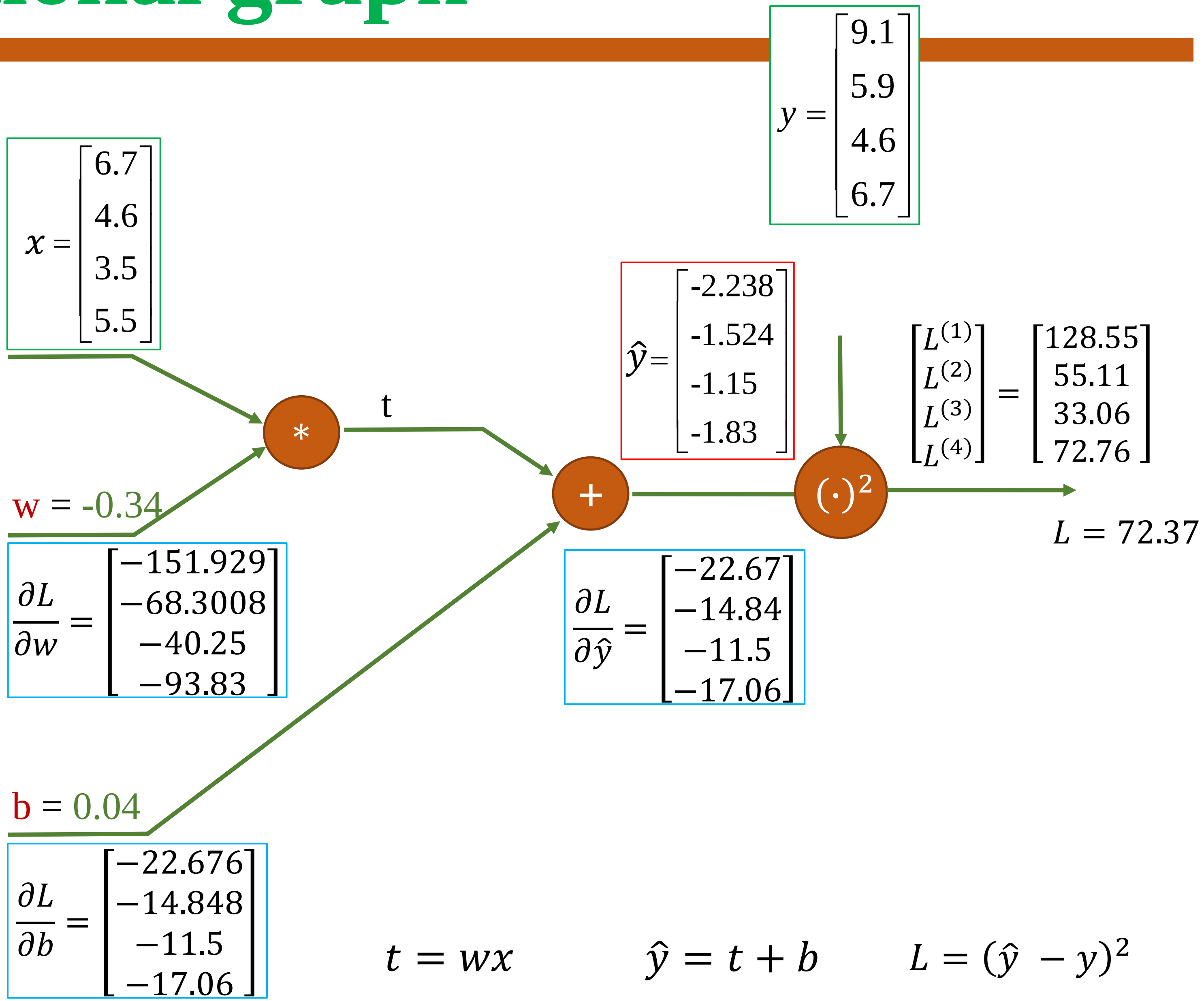
❖ N-sample training

$t = wx$ $\hat{y} = t + b$ $L = (\hat{y} - y)^2$



Computational graph

- ❖ **House price prediction**
  - ❖ **N-sample training**



$$\frac{sum(\frac{\partial L}{\partial w})}{4} = -88.5775$$

$$\frac{sum(\frac{\partial L}{\partial b})}{4} = -16.521$$

$$t = wx \qquad \hat{y} = t + b \qquad L = (\hat{y} - y)^2$$

Computational graph

❖ House price prediction

❖ N-sample training

Update w and b

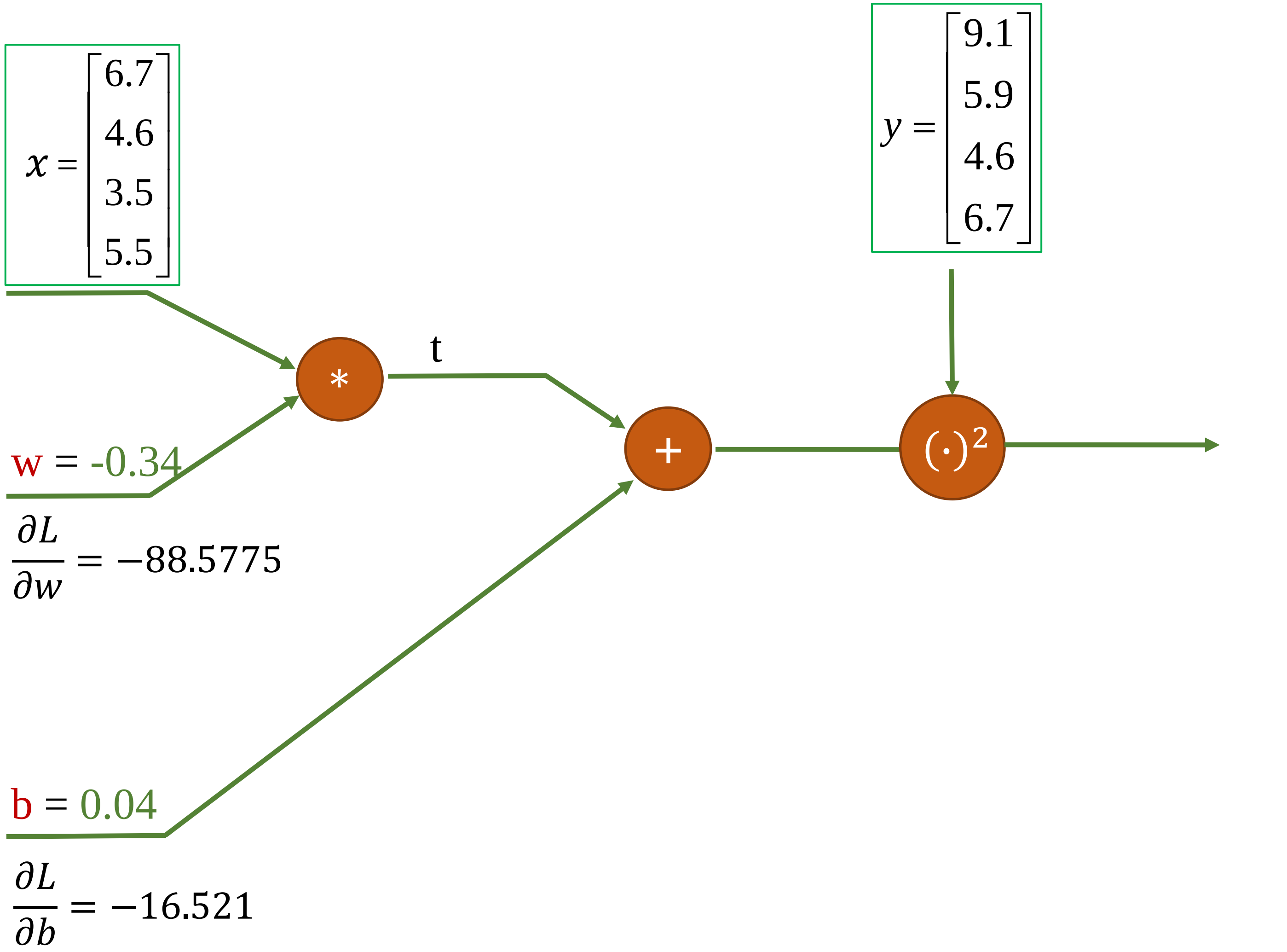
$$w = w - \eta * \frac{\partial L}{\partial w}$$

$$b = b - \eta * \frac{\partial L}{\partial b}$$

Learning rate $\eta = 0.01$

$$w = -0.34 - (0.01 * (-88.5775)) = 0.54$$

$$b = 0.04 - \left(0.01 * (-16.521)\right) = 0.205$$



$t = wx \qquad \hat{y} = t + b \qquad L = (\hat{y} - y)^2$

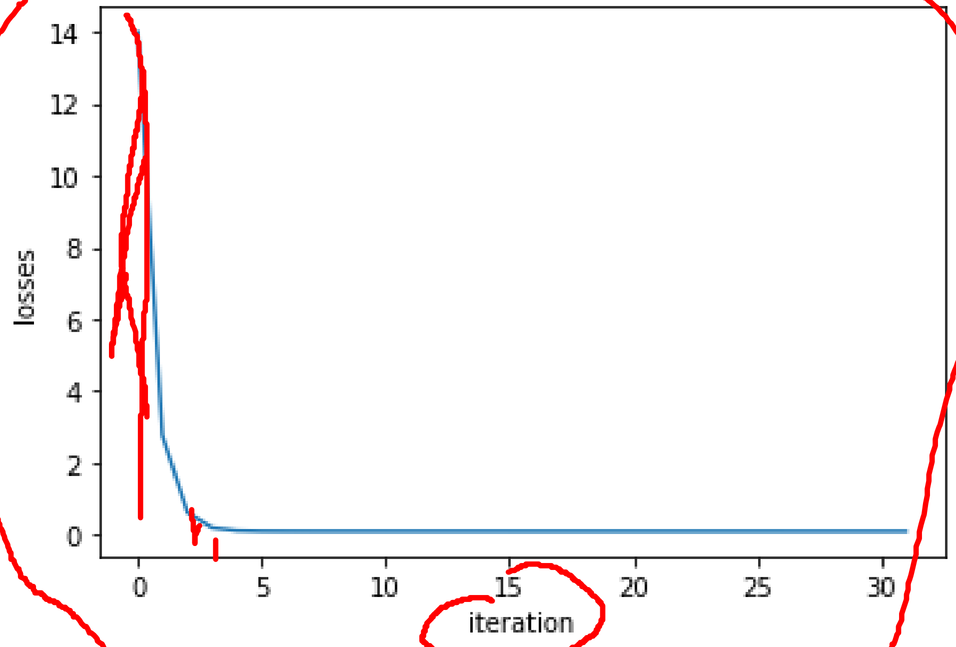
Computational graph

❖ House price prediction

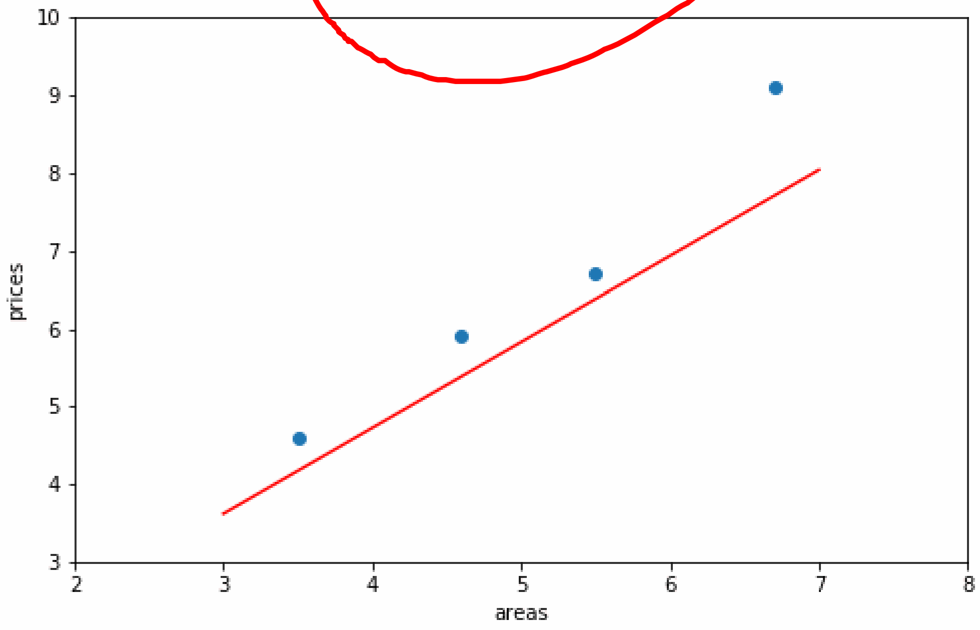
❖ N-sample training

1000

100



Losses for 30 iterations



Model updating for different iterations

Extension

Features			Label
TV	Radio	Newspaper	Sales
230.1	37.8	69.2	22.1
44.5	39.3	45.1	10.4
17.2	45.9	69.3	12
151.5	41.3	58.5	16.5
180.8	10.8	58.4	17.9

Advertising data

Model: $\hat{y} = w_1 x_1 + w_2 x_2 + w_3 x_3 + b$

Sale = $w_1 * TV + w_2 * Radio + w_3 * Newspaper + b$

Linear Regression

❖ General formula

Feature	Label
area	price
6.7	9.1
4.6	5.9
3.5	4.6
5.5	6.7

House price data

Features			Label
TV	Radio	Newspaper	Sales
230.1	37.8	69.2	22.1
44.5	39.3	45.1	10.4
17.2	45.9	69.3	12
151.5	41.3	58.5	16.5
180.8	10.8	58.4	17.9

Advertising data

Model: $\hat{y} = w_1 x_1 + b$

price = a $* \, area + b$

Model: $\hat{y} = w_1 x_1 + w_2 x_2 + w_3 x_3 + b$

Sale = $w_1 * TV + w_2 * Radio + w_3 * Newspaper + b$

Linear Regression

1) Pick a sample (x_1, x_2, x_3, y) from training data

2) Compute the output $\hat{y}$

$$\hat{y} = w_1 * TV + w_2 * R + w_3 * N + b$$

$$\hat{y} = w_1 * x_1 + w_2 * x_2 + w_3 * x_3 + b$$

3) Compute loss

$$L = (\hat{y} - y)^2$$

4) Compute derivative

$$\frac{\partial L}{\partial w_1} = 2x_1(\hat{y} - y) \qquad \frac{\partial L}{\partial w_3} = 2x_3(\hat{y} - y)$$

$$\frac{\partial L}{\partial w_2} = 2x_2(\hat{y} - y) \qquad \frac{\partial L}{\partial b} = 2(\hat{y} - y)$$

5) Update parameters

$$w_1 = w_1 - \eta \frac{\partial L}{\partial w_1} \qquad w_3 = w_3 - \eta \frac{\partial L}{\partial w_3}$$

$$w_2 = w_2 - \eta \frac{\partial L}{\partial w_2} \qquad b = b - \eta \frac{\partial L}{\partial b}$$

Features			Label
TV	Radio	Newspaper	Sales
230.1	37.8	69.2	22.1
44.5	39.3	45.1	10.4
17.2	45.9	69.3	12
151.5	41.3	58.5	16.5
180.8	10.8	58.4	17.9

Advertising data

Model

$$\text{Sale} = w_1 * TV + w_2 * Radio + w_3 * Newspaper + b$$

$$\hat{y} = w_1x_1 + w_2x_2 + w_3x_3 + b$$

1) Pick a sample (x_1, x_2, x_3, y) from training data

2) Compute the output $\hat{y}$

$$\hat{y} = w_1 * TV + w_2 * R + w_3 * N + b$$

$$\hat{y} = w_1 * x_1 + w_2 * x_2 + w_3 * x_3 + b$$

3) Compute loss

$$L = (\hat{y} - y)^2$$

4) Compute derivative

$$\frac{\partial L}{\partial w_1} = 2x_1(\hat{y} - y) \qquad \frac{\partial L}{\partial w_3} = 2x_3(\hat{y} - y)$$

$$\frac{\partial L}{\partial w_2} = 2x_2(\hat{y} - y) \qquad \frac{\partial L}{\partial b} = 2(\hat{y} - y)$$

5) Update parameters

$$w_1 = w_1 - \eta \frac{\partial L}{\partial w_1} \qquad w_3 = w_3 - \eta \frac{\partial L}{\partial w_3}$$

$$w_2 = w_2 - \eta \frac{\partial L}{\partial w_2} \qquad b = b - \eta \frac{\partial L}{\partial b}$$

```python
# compute output and loss
def predict(x1, x2, x3, w1, w2, w3, b):
    return w1*x1 + w2*x2 + w3*x3 + b
def compute_loss(y_hat, y):
    return (y_hat - y)**2

# compute gradient
def compute_gradient_wi(xi, y, y_hat):
    dl_dwi = 2*xi*(y_hat-y)
    return dl_dwi
def compute_gradient_b(y, y_hat):
    dl_db = 2*(y_hat-y)
    return dl_db

# update weights
def update_weight_wi(wi, dl_dwi, lr):
    wi = wi - lr*dl_dwi
    return wi
def update_weight_b(b, dl_db, lr):
    b  = b - lr*dl_db
    return b
```

Features			Label
TV	Radio	Newspaper	Sales
230.1	37.8	69.2	22.1
44.5	39.3	45.1	10.4
17.2	45.9	69.3	12
151.5	41.3	58.5	16.5
180.8	10.8	58.4	17.9

```
def initialize_params():
    w1 = random.gauss(mu=0.0, sigma=0.01)
    w2 = random.gauss(mu=0.0, sigma=0.01)
    w3 = random.gauss(mu=0.0, sigma=0.01)
    b  = 0

    return w1, w2, w3, b

# initialize model's parameters
w1, w2, w3, b = initialize_params()
print(w1, w2, w3, b)
```

```
0.016095064695494 67 0.00607778501208891 0.0023344573891806507 0
```

```
import numpy as np
import random

def get_column(data, index):
    result = [row[index] for row in data]
    return result

data = np.genfromtxt('advertising.csv',
                     delimiter=',',
                     skip_header=1).tolist()

# get tv (index=0)
tv_data = get_column(data, 0)

# get radio (index=1)
radio_data = get_column(data, 1)

# get newspaper (index=2)
newspaper_data = get_column(data, 2)

# get sales (index=0)
sales_data = get_column(data, 3)
```

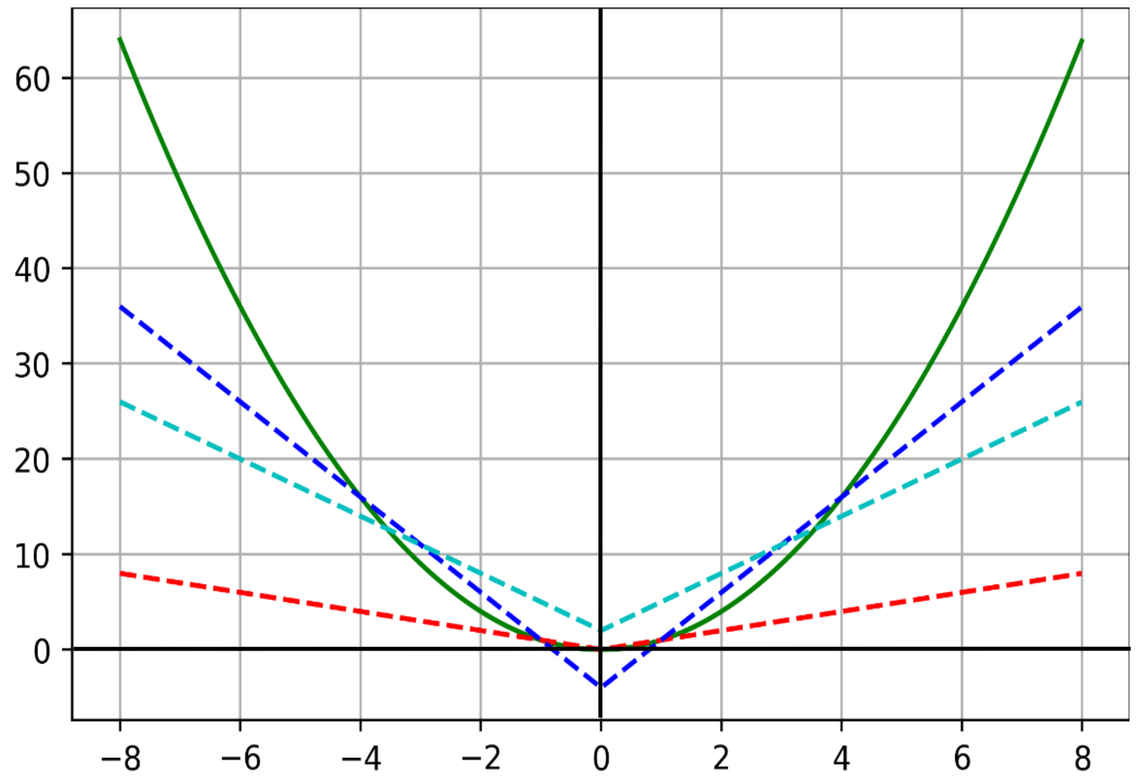
Outline

SECTION 1
Linear Regression

SECTION 2
Mini-batch Training

SECTION 3
Batch Training

SECTION 4
Loss Functions (Optional)



Discussion 1: Is it OK to use the following loss function?

$$L = \frac{1}{2}(\hat{y} - y)^2$$

Discussion 2: if so, construct formulas

$$\frac{\partial L}{\partial \hat{y}} \quad (\hat{y} - y) \cdot 1$$

Discussion 3: What about the following loss function?

$$L = \frac{1}{2}(y - \hat{y})^2$$

$$(y - \hat{y}) \cdot (-1)$$

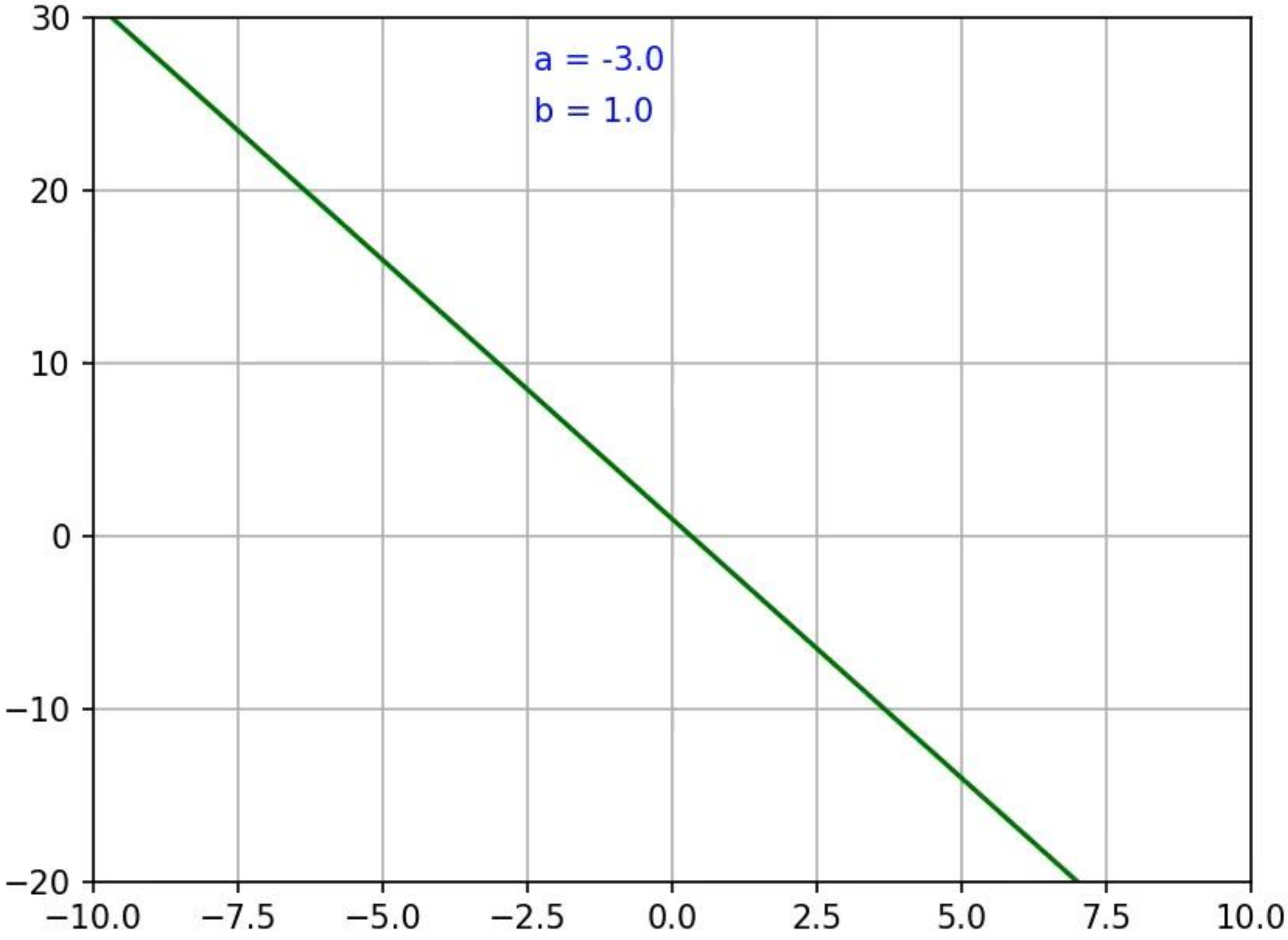
Discussion 4: Construct the connection between the two following losses?

$$\boxed{L_1 = \frac{1}{2}(\hat{y} - y)^2} \qquad L_2 = (\hat{y} - y)^2$$

$\hat{y} - y$

$$\hat{y} = wx + b$$

Discussion 5:
Can we remove b?



$$\hat{y} = wx + b$$

Discussion 5:
Can we remove b?