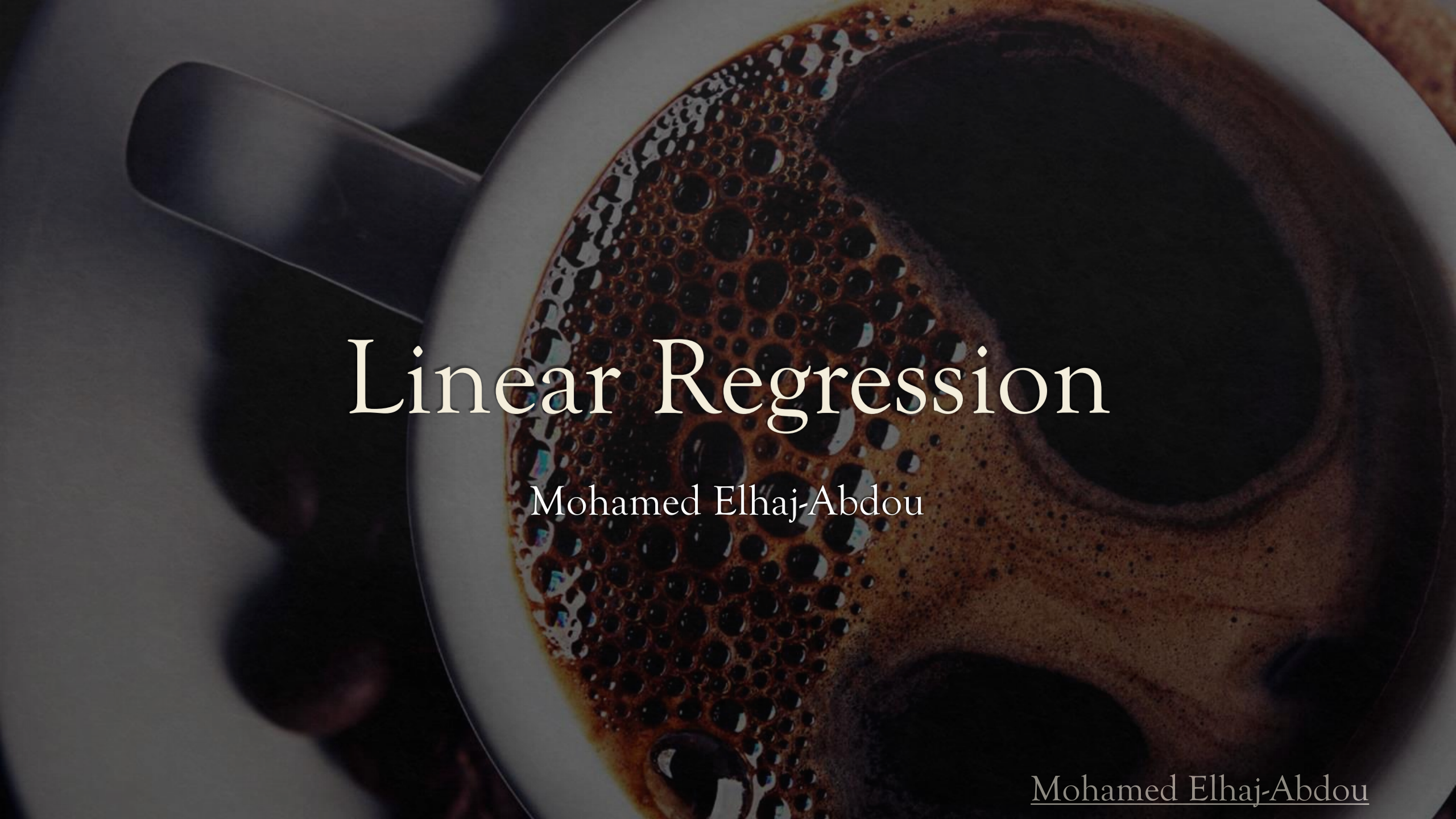


# Linear Regression

Mohamed Elhaj-Abdou

# Linear Regression

Linear Regression one Variable

Linear Regression Multiple variable

Polynomial Regression

Mohamed Elhaj-Abdou

# Linear Regression

Before we talk about Linear Regression

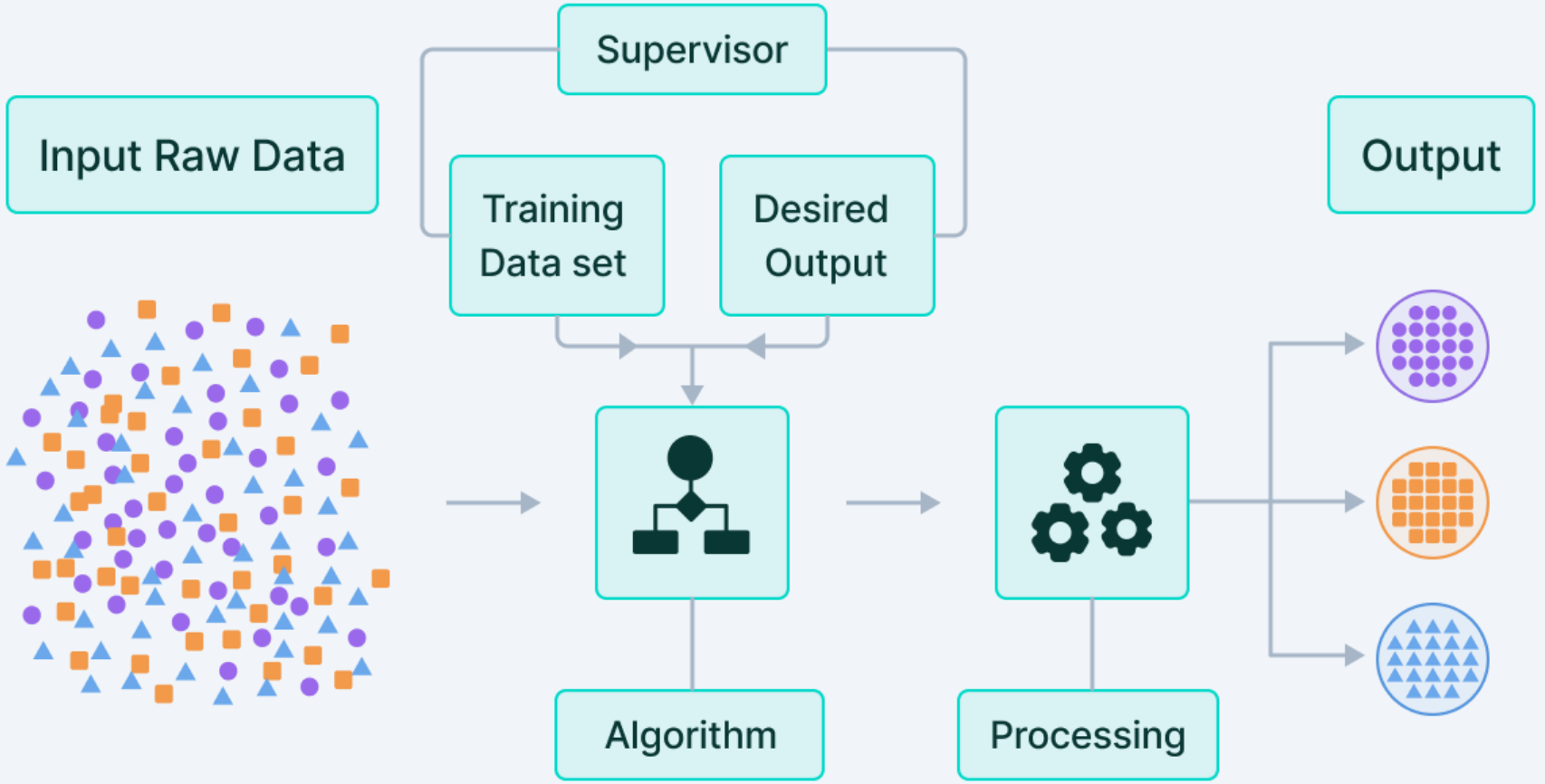
We need o have a discussion about what is supervised learning Vs Unsupervised learning

Mohamed Elhaj-Abdou

# What is Supervised Learning?!

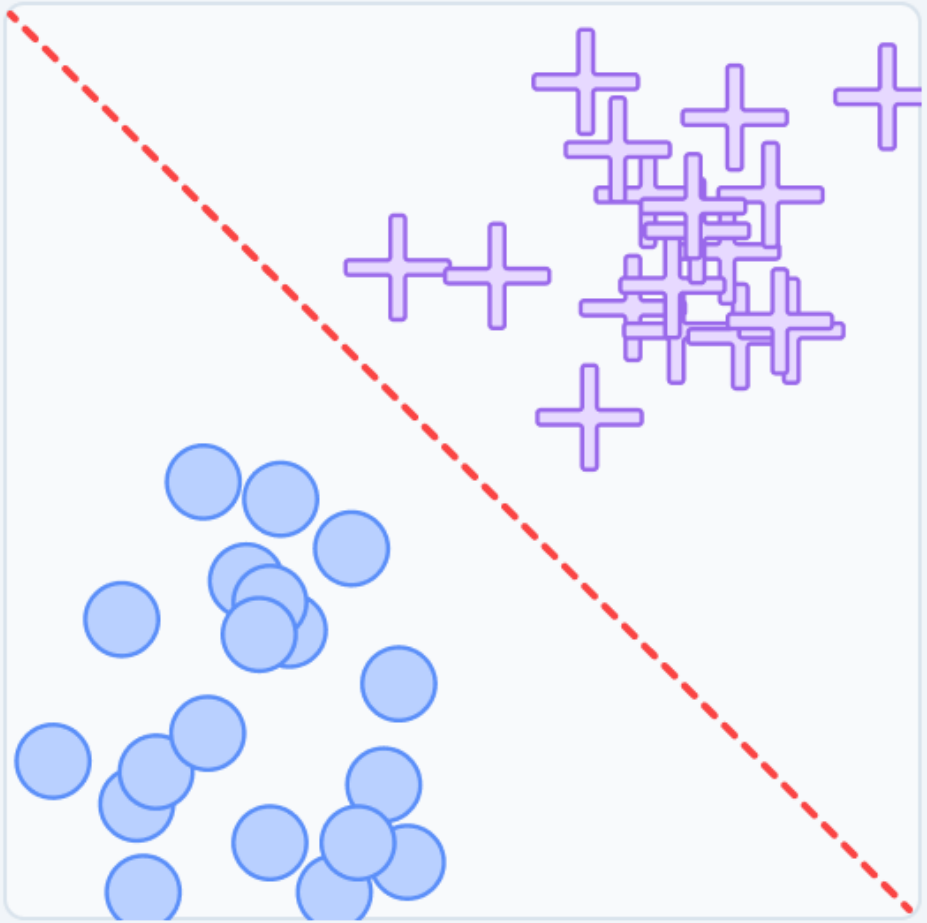
Mohamed Elhaj-Abdou

# Supervised Learning

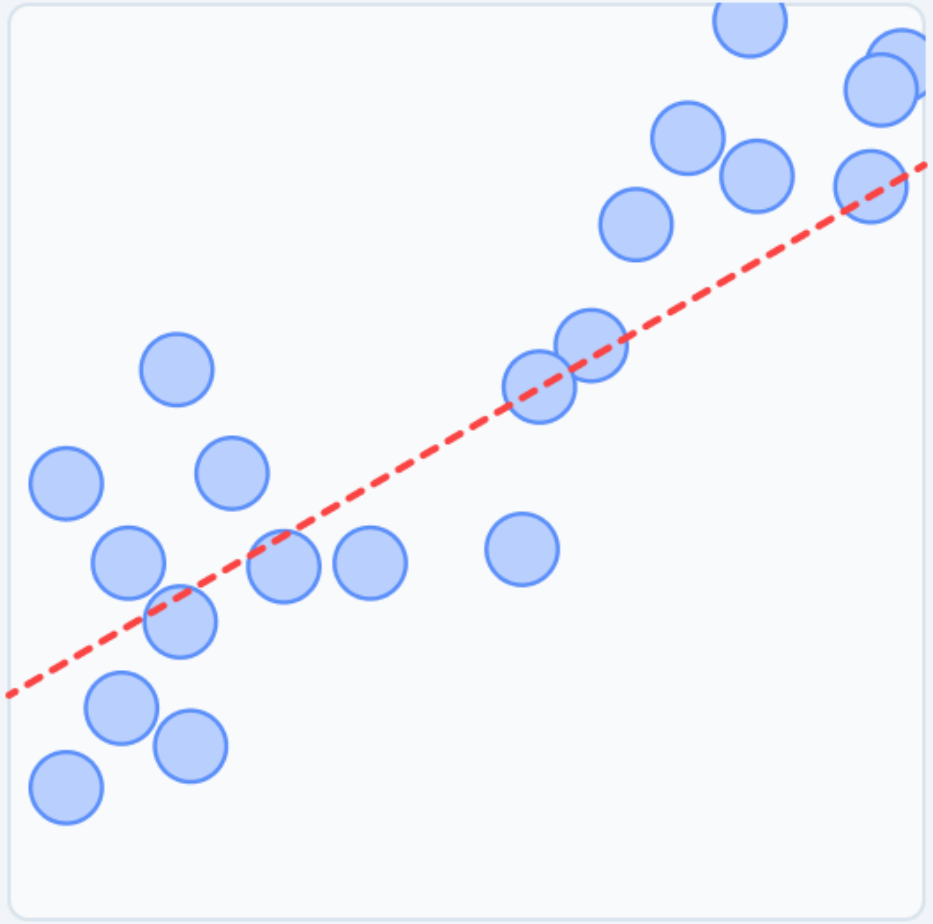



Mohamed Elhaj-Abdou

## Classification



## Regression



## Unsupervised Learning

### Dimensionality Reduction

- Feature Elicitation
- Meaningful Compression
- Structure Discovery
- Big data visualization

### Clustering

- Recommender Systems
- Targeted Marketing
- Customer Segmentation

## Reinforcement Learning

- Real-time decisions
- Game AI
- Robot Navigation
- Learning Tasks
- Skill Acquisition

## Supervised Learning

### Classification

- Identity Fraud Detection
- Image Classification
- Customer Retention
- Diagnostics

### Regression

- Population Growth Prediction
- Estimating life expectancy
- Market Forecasting
- Weather Forecasting
- Advertising Popularity Prediction

Mohamed Elhaj-Abdou

Suppose that you want to predict the price of the house based on the size of the house

The red points in X symbols represents the dataset

Where each datapoint represents the price of the house given the size in square feet

This is a dataset already available meaning that these houses already sold by this prices and sizes

Lets say you are real state agent and you are helping ppl to sell the houses

And its important when a client gives you a features or characteristics of the house you should give the approx. price

You started to measure the house size which is

You tell me how much you think this house will be sold for

# We can do that by Building Linear Regression Model

Linear Regression will try to fit this data in order to predict the price correct

The line could be drawn like this


And the LR model will try to find the price of the house from the intersection between the LR line and the price Vs size

Which could be like this


Data table

| size in feet² | price in $1000's |
|---|---|
| 2104 | 400 |
| 1416 | 232 |
| 1534 | 315 |
| 852 | 178 |
| ... | ... |
| 3210 | 870 |

The data used in the training process is called training set

The data that is used to test the model is test set

Mohamed Elhaj-Abdou

The features are called X-training

The required output are called Y-training

The number of training examples are called m

The data used in the training process is called training set

Data table

| size in feet² | price in $1000's |
|---|---|
| 2104 | 400 |
| 1416 | 232 |
| 1534 | 315 |
| 852 | 178 |
| ... | ... |
| 3210 | 870 |

The data that is used to test the model is test set

# Terminology

Training set: Data used to train the model

| | $x$ size in feet$^2$ | $y$ price in \$1000's |
|---|---|---|
| (1) | 2104 | 400 |
| (2) | 1416 | 232 |
| (3) | 1534 | 315 |
| (4) | 852 | 178 |
| ... | ... | ... |
| (47) | 3210 | 870 |

$m = 47$

$x^{(1)} = 2104$ $y^{(1)} = 400$

$(x^{(1)}, y^{(1)}) = (2104, 400)$

$x^{(2)} = 1416$ $x^{(2)} \neq x^2$ not exponent

Notation:

$x$ = "input" variable
feature

$y$ = "output" variable
"target" variable

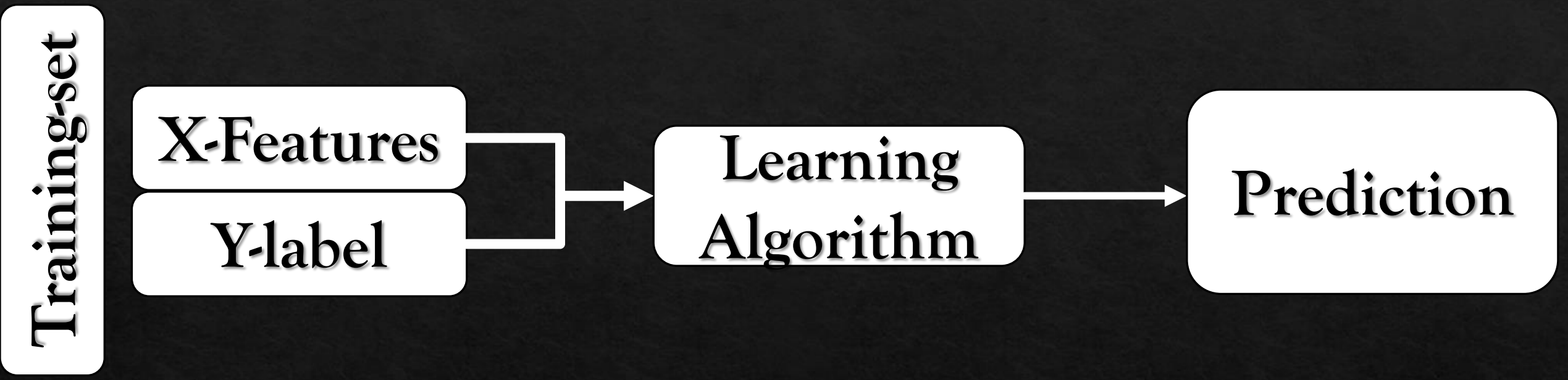
$m$ = number of training examples

$(x, y)$ = single training example

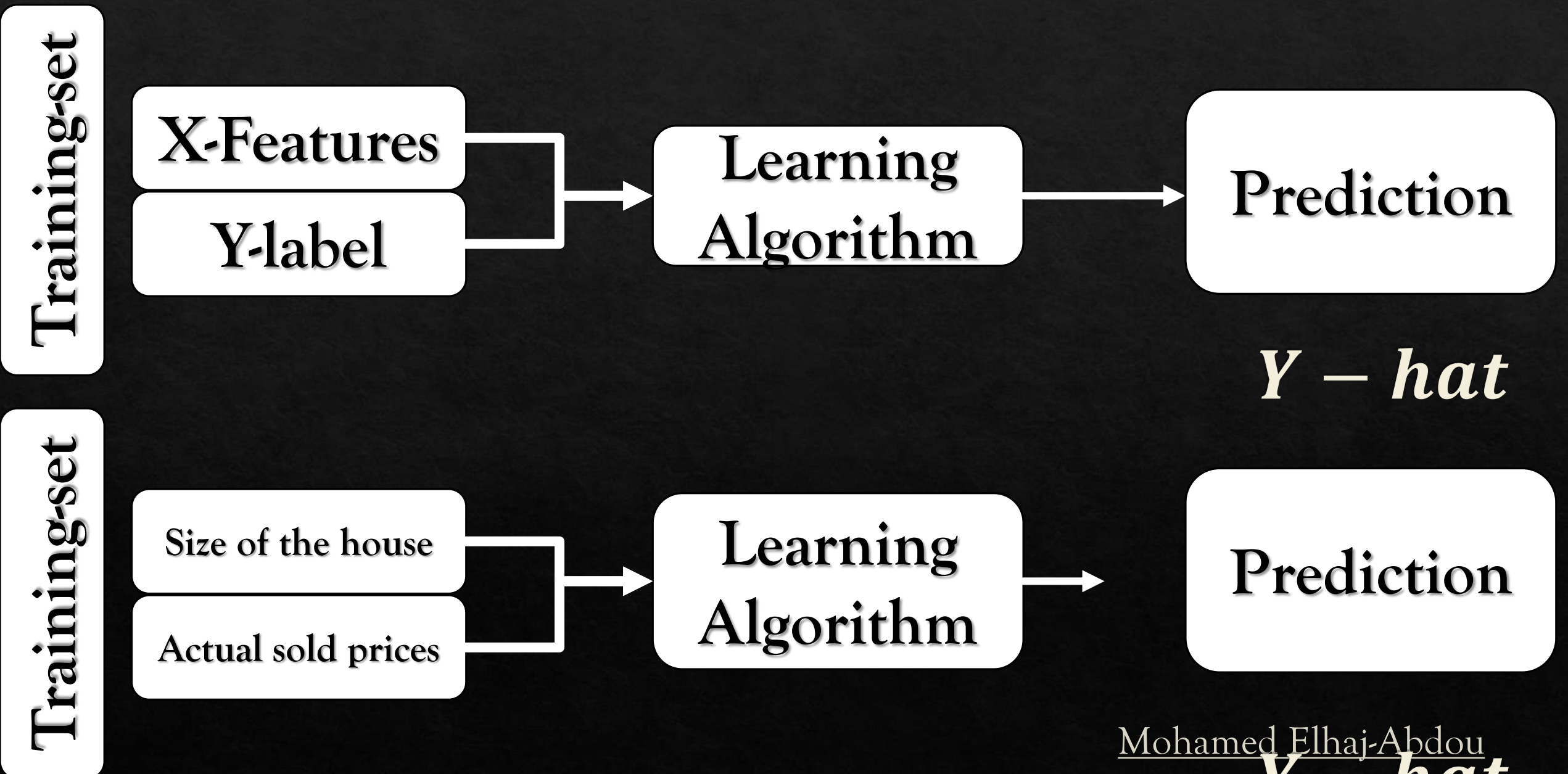
$(x^{(i)}, y^{(i)})$

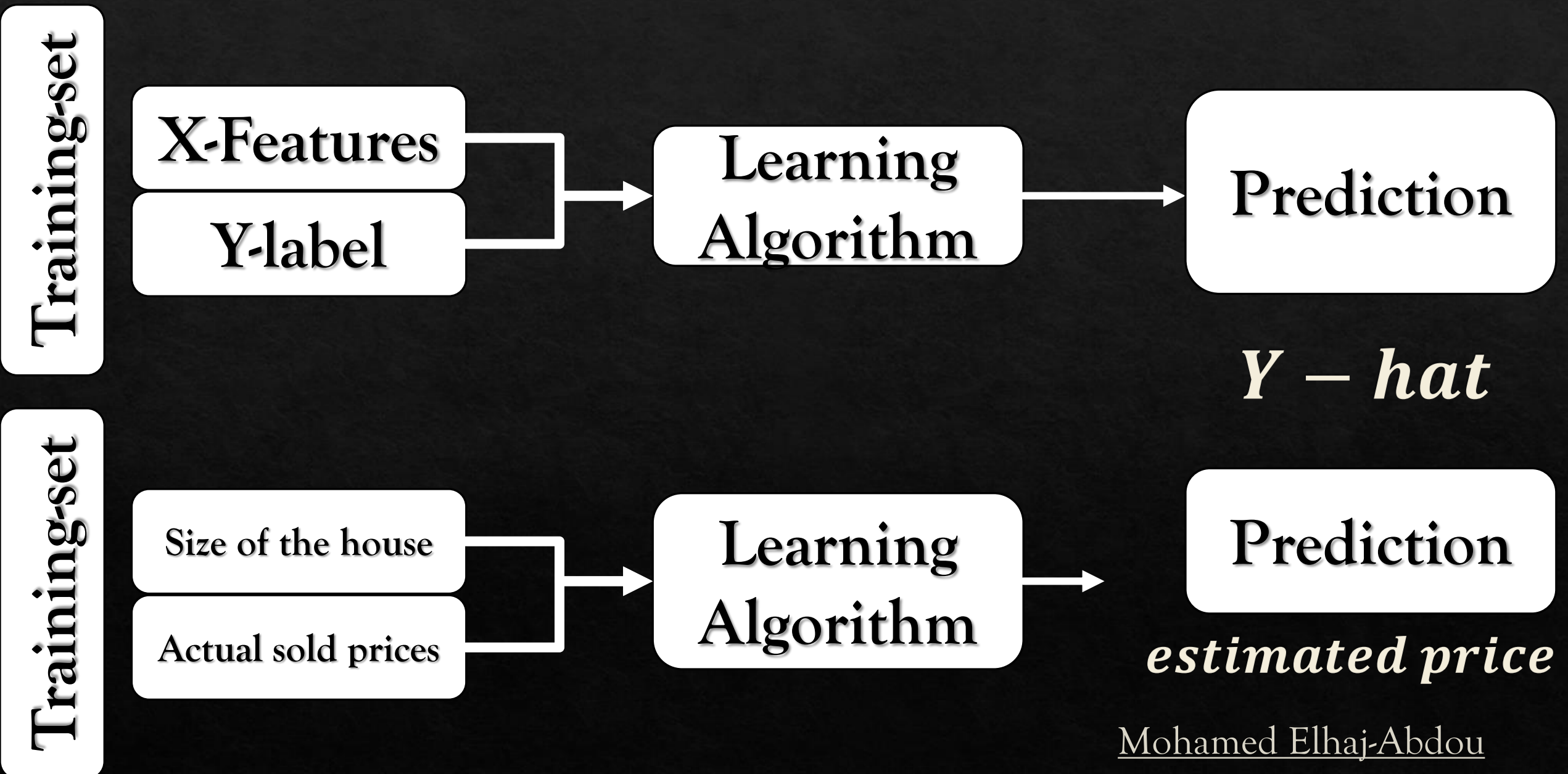
$(x^{(i)}, y^{(i)})$ = $i^{th}$ training example

index ($1^{st}$, $2^{nd}$, $3^{rd}$ ...)




Mohamed Elhaj-Abdou


Training-set
X-Features
Y-label
Learning Algorithm
Prediction
Y – hat
Training-set
Size of the house
Actual sold prices
Learning Algorithm
Prediction




Mohamed Elhaj-Abdou

# How to represent the Y-hat, or Y-estimated, or Estimated price

Mohamed Elhaj-Abdou

$$f_{(w,b)}X = WX + b$$

Mohamed Elhaj-Abdou

# Training set

features

targets

| size in feet$^2$ ($x$) | price \$1000's ($y$) |
|---|---|
| 2104 | 460 |
| 1416 | 232 |
| 1534 | 315 |
| 852 | 178 |
| ... | ... |

Model: $f_{w,b}(x) = wx + b$

$w, b$: parameters

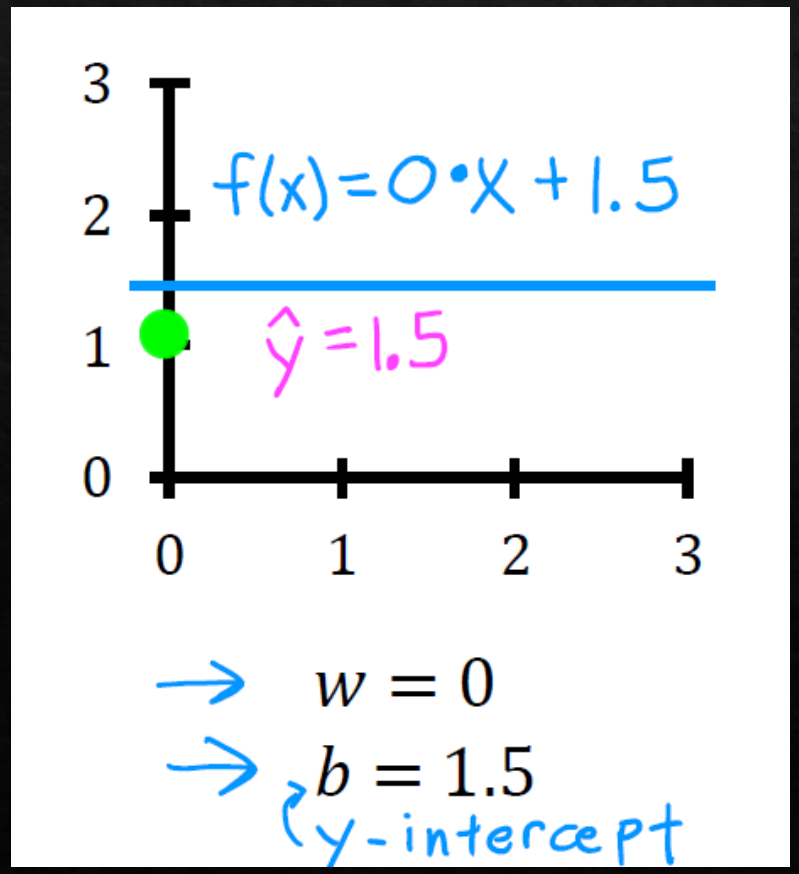
coefficients

weights

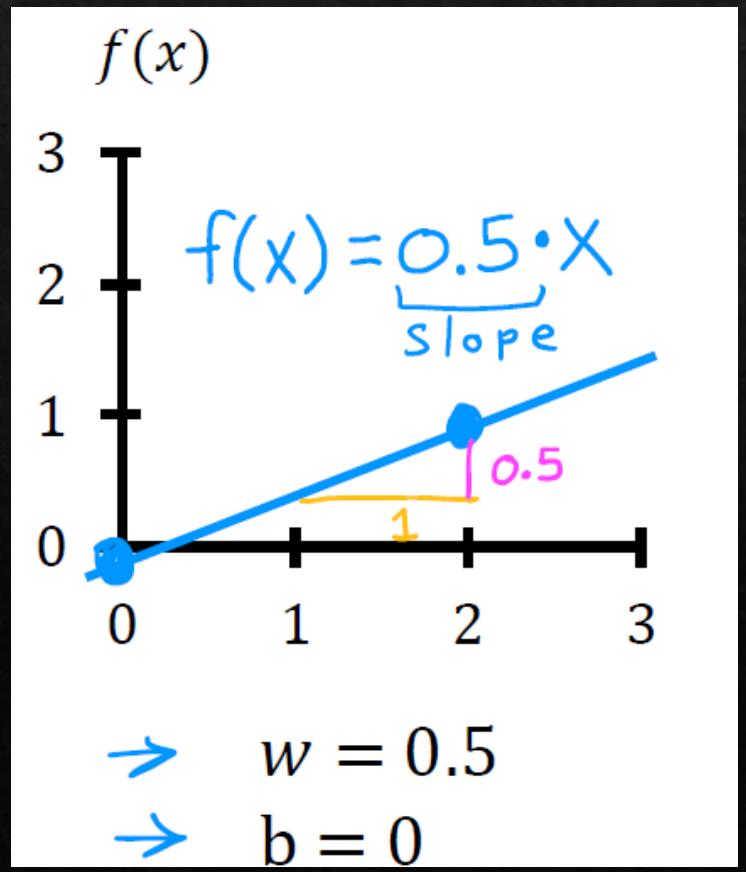
What do $w, b$ do?

$$f_{(w,b)}X = WX + b \qquad or\ Y - hat = WX + b$$

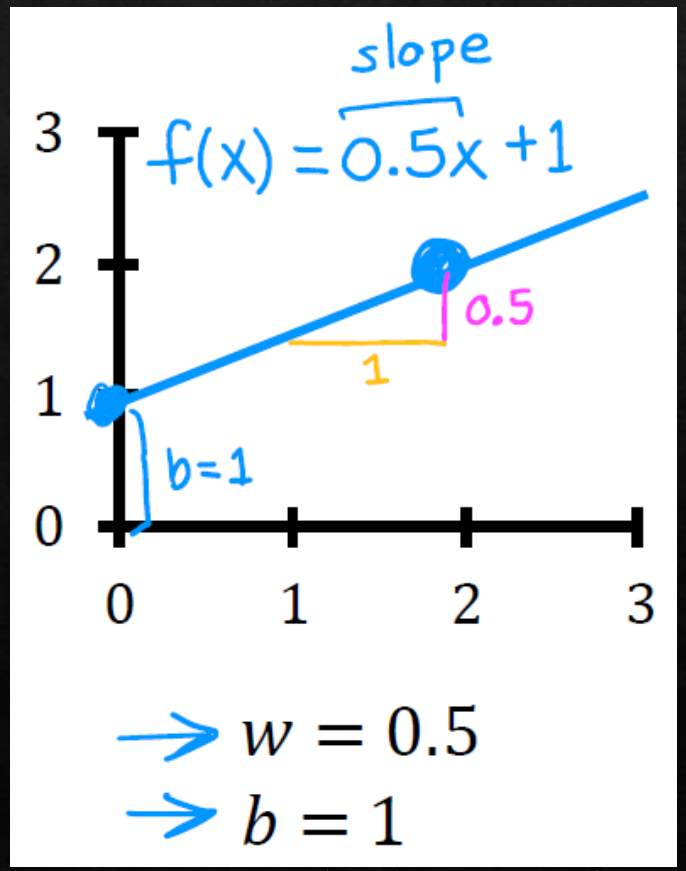
W=0
B=1.5




W=0.5
B=0




W=0.5
B=1

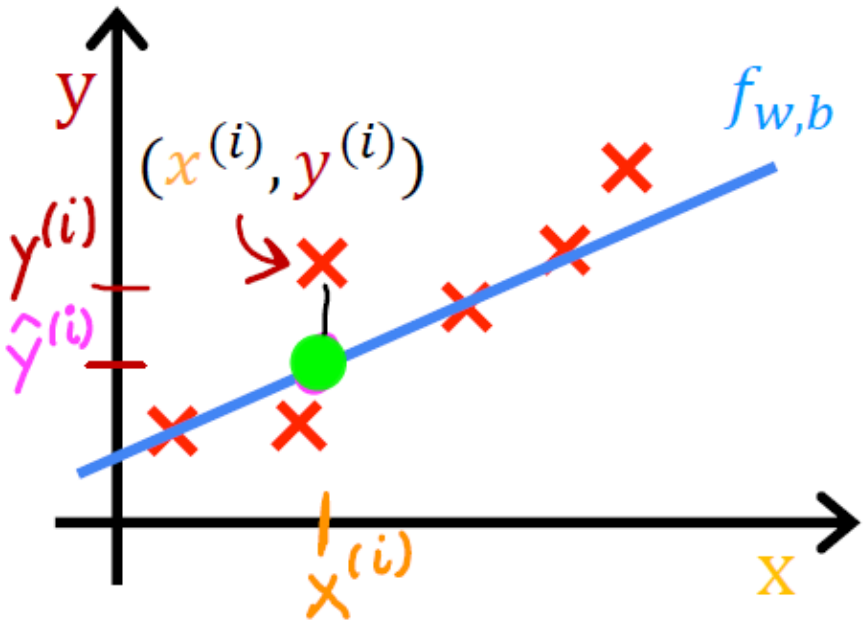



Mohamed Elhaj-Abdou

How we can know that the model is doing good or bad during the training process

Cost Function is our key indicator

How we calculate the cost function?!




$$\hat{y}^{(i)} = f_{w,b}(x^{(i)})$$

$$f_{w,b}(x^{(i)}) = wx^{(i)} + b$$

## Cost function: Squared error cost function

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

$\hat{y}^{(i)}$ error $y^{(i)}$

m = number of training examples

$$J(w,b) = \frac{1}{2m}\sum_{i=1}^{m}\left(f_{w,b}(x^{(i)}) - y^{(i)}\right)^2$$

$J(w,b)$: intuition (next!)

Find $w, b$:

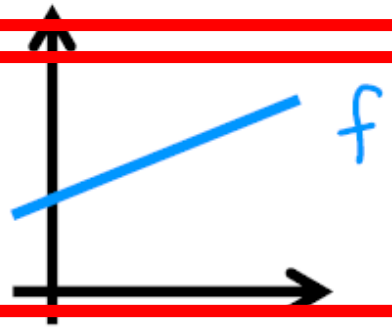
$\hat{y}^{(i)}$ is close to $y^{(i)}$ for all $(x^{(i)}, y^{(i)})$.

model:

$$f_{w,b}(x) = wx + b$$

parameters:

$w, b$




cost function:

$$J(w,b) = \frac{1}{2m}\sum_{i=1}^{m}(f_{w,b}(x^{(i)}) - y^{(i)})^2$$

goal:

$$\underset{w,b}{\text{minimize}}\, J(w,b)$$

Based on the parameters that the model puts it it could be a straight line this which should be the line that need to fit the training data

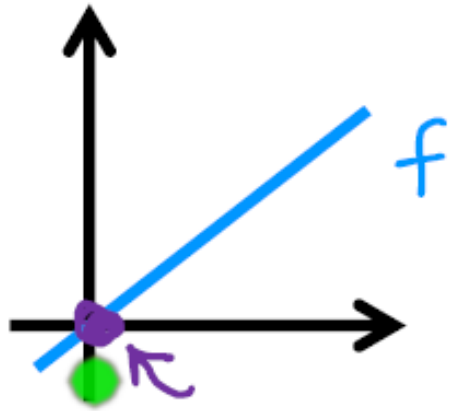
To measure how well the model is going to fit the data we need the cost function which is the difference between the Y-estimated Minus the Y-True

Our Target Is to make this difference equals zero or as minimum as possible can you tell me why ?

Mohamed Elhaj-Abdou

## simplified

$f_w(x) = wx$ $\quad b = \emptyset$




$w$

$$J(w) = \frac{1}{2m}\sum_{i=1}^{m}(f_w(x^{(i)}) - y^{(i)})^2$$

$wx^{(i)}$

$$\underset{w}{\text{minimize}}\, J(w)$$

Can we say that b as bias or constant equal to zero. This to simplify the problem

Y-hat will be W*X
Where W is the weights or parameter
X is the training example feature (size of the house)

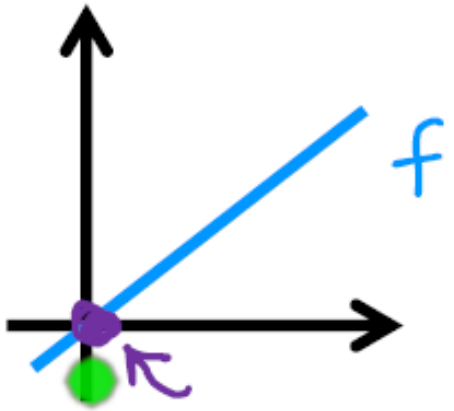
So J(W) is a function in just one variable which is W

That's why its called LR with one variable

Mohamed Elhaj-Abdou

# simplified

$$f_w(x) = wx \qquad b = \emptyset$$




$w$

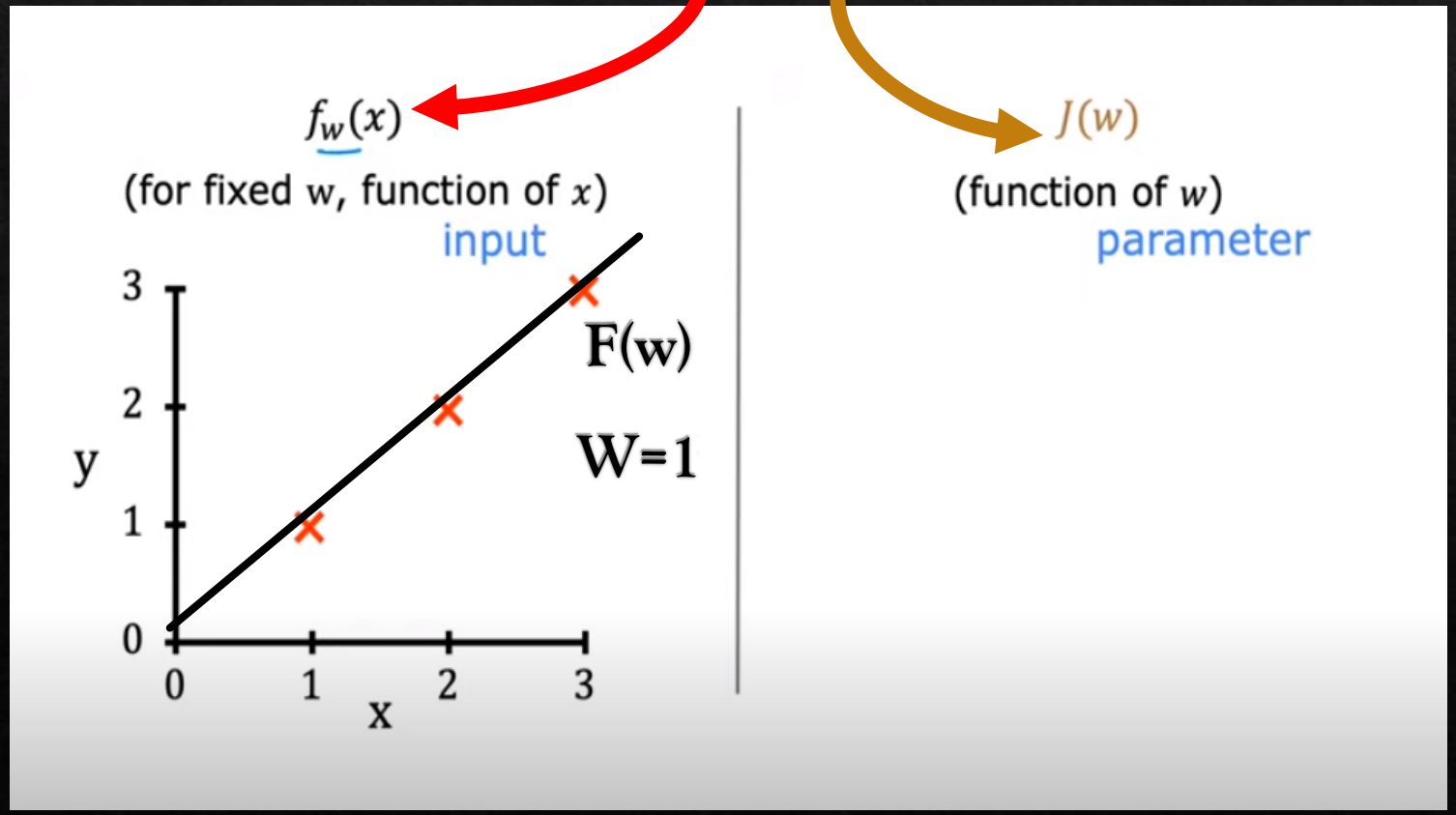
$$J(w) = \frac{1}{2m}\sum_{i=1}^{m}(f_w(x^{(i)}) - y^{(i)})^2$$

$wx^{(i)}$

$$\underset{w}{\text{minimize}}\, J(w)$$

So the Goal is to Find the W that minimize the J(W) ==Error

Mohamed Elhaj-Abdou




1- Lets pick a value of W=1

2- Lets calculate the J(w)
**W*X-Y-true**
**=1*1-1=0**

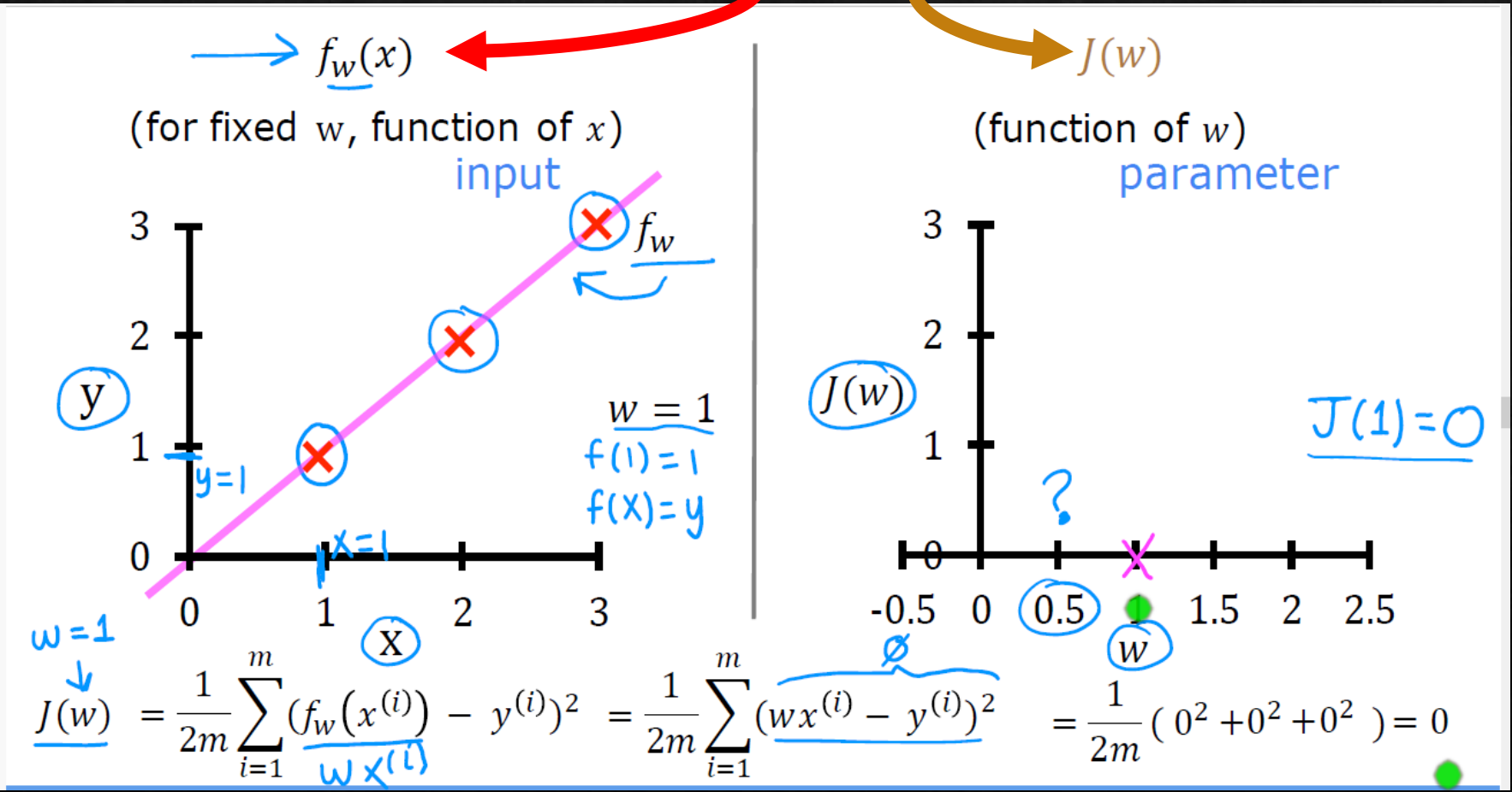
2- J(W) is the error between the Y-hat and Y-true, this means that the error is equals to 0

Mohamed Elhaj-Abdou

$$J(w) = \frac{1}{2m}\sum_{i=1}^{m}(f_w(x^{(i)}) - y^{(i)})^2$$

minimize $J(w)$

$w x^{(i)}$




$$J(w) = \frac{1}{2m}\sum_{i=1}^{m}(f_w(x^{(i)}) - y^{(i)})^2 = \frac{1}{2m}\sum_{i=1}^{m}(wx^{(i)} - y^{(i)})^2 = \frac{1}{2m}(0^2 + 0^2 + 0^2) = 0$$

1- Lets pick a value of W=1

2- Lets calculate the J(w)
**W*X-Y-true**
**=1*1-1=0**

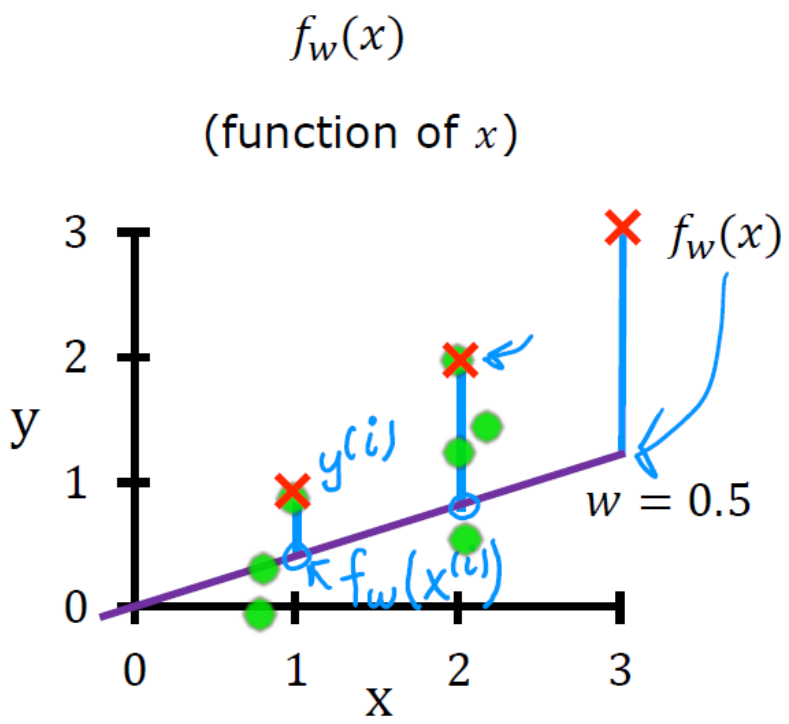
2- J(W) is the error between the Y-hat and Y-true, this means that the error is equals to 0

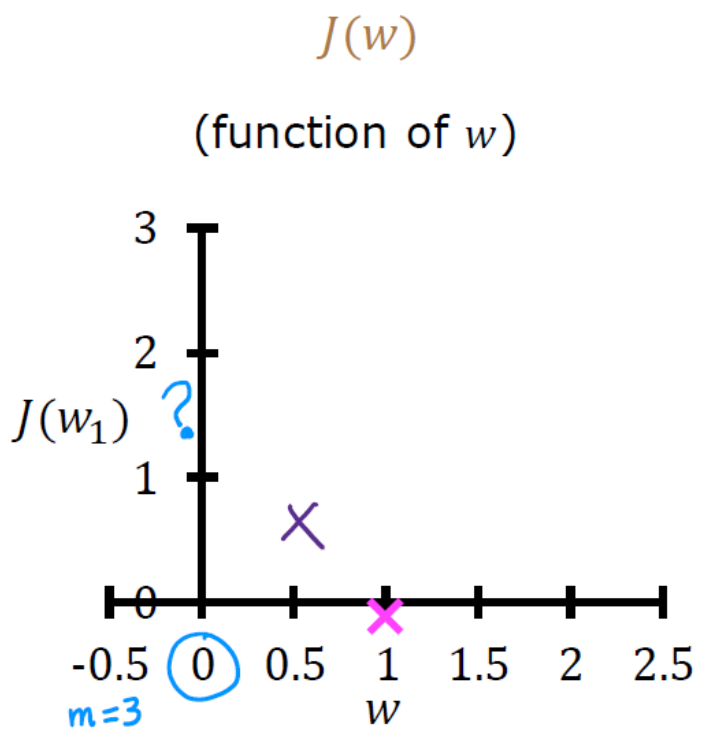
Mohamed Elhaj-Abdou

$$J(w) = \frac{1}{2m}\sum_{i=1}^{m}(f_w(x^{(i)}) - y^{(i)})^2$$

$$\min_w J(w)$$

$wx^{(i)}$




$$J(0.5) = \frac{1}{2m}\left[(0.5-1)^2 + (1-2)^2 + (1.5-3)^2\right] = \frac{1}{2\times 3}[3.5] = \frac{3.5}{6} \approx 0.58 \quad (m=3)$$

1- Lets pick a value of W=0.5

2- Lets calculate the J(w) **W*X-Y-true**

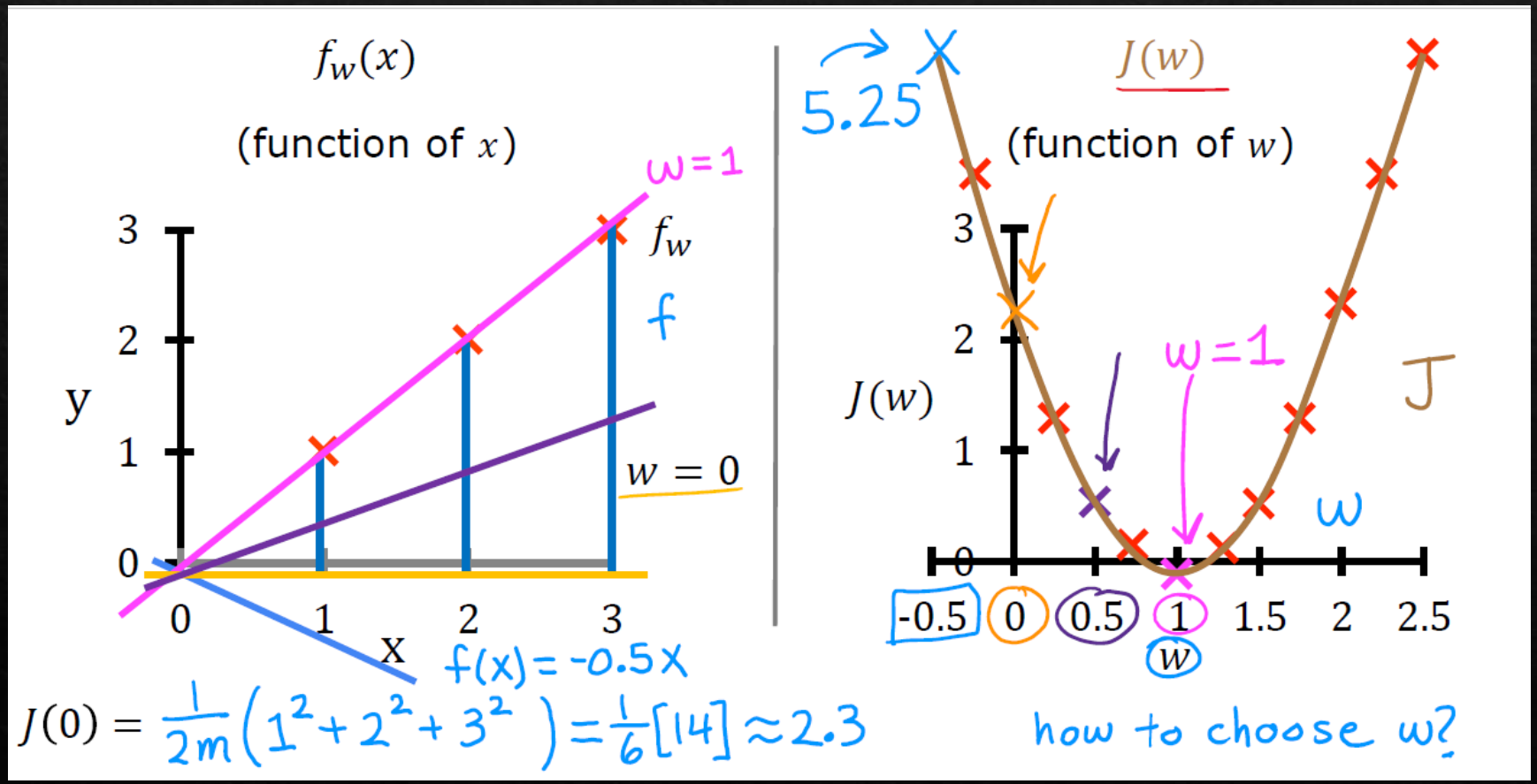
3- J(W) is the error between the Y-hat and Y-true, this is calculated in the blue colored lines

Mohamed Elhaj-Abdou

$$J(w) = \frac{1}{2m}\sum_{i=1}^{m}(f_w(x^{(i)}) - y^{(i)})^2$$

$wx^{(i)}$

$$\underset{w}{\text{minimize}}\, J(w)$$




1- Lets pick a value of W=zero

2- Lets calculate the J(w) **W*X-Y-true**

3- J(W) is the error between the Y-hat and Y-true, this is calculated in the blue colored lines

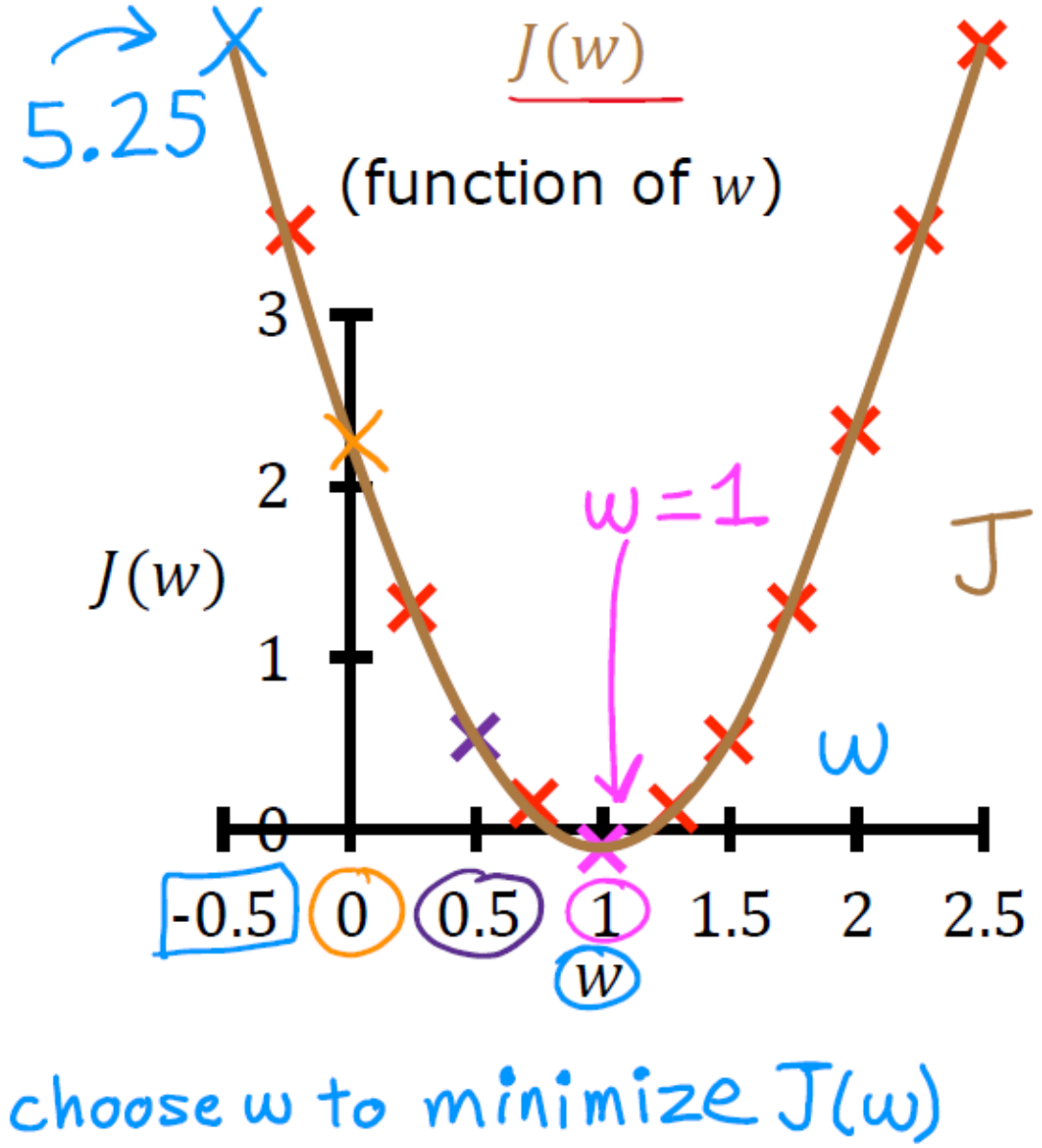
Mohamed Elhaj-Abdou

goal of linear regression:

$$\underset{w}{\text{minimize}}\, J(w)$$

general case:

$$\underset{w,b}{\text{minimize}}\, J(w,b)$$



***How we can choose the W and b during the learning***

***Gradent descent***

Mohamed Elhaj-Abdou

Have some function $J(w, b)$ for linear regression or any function

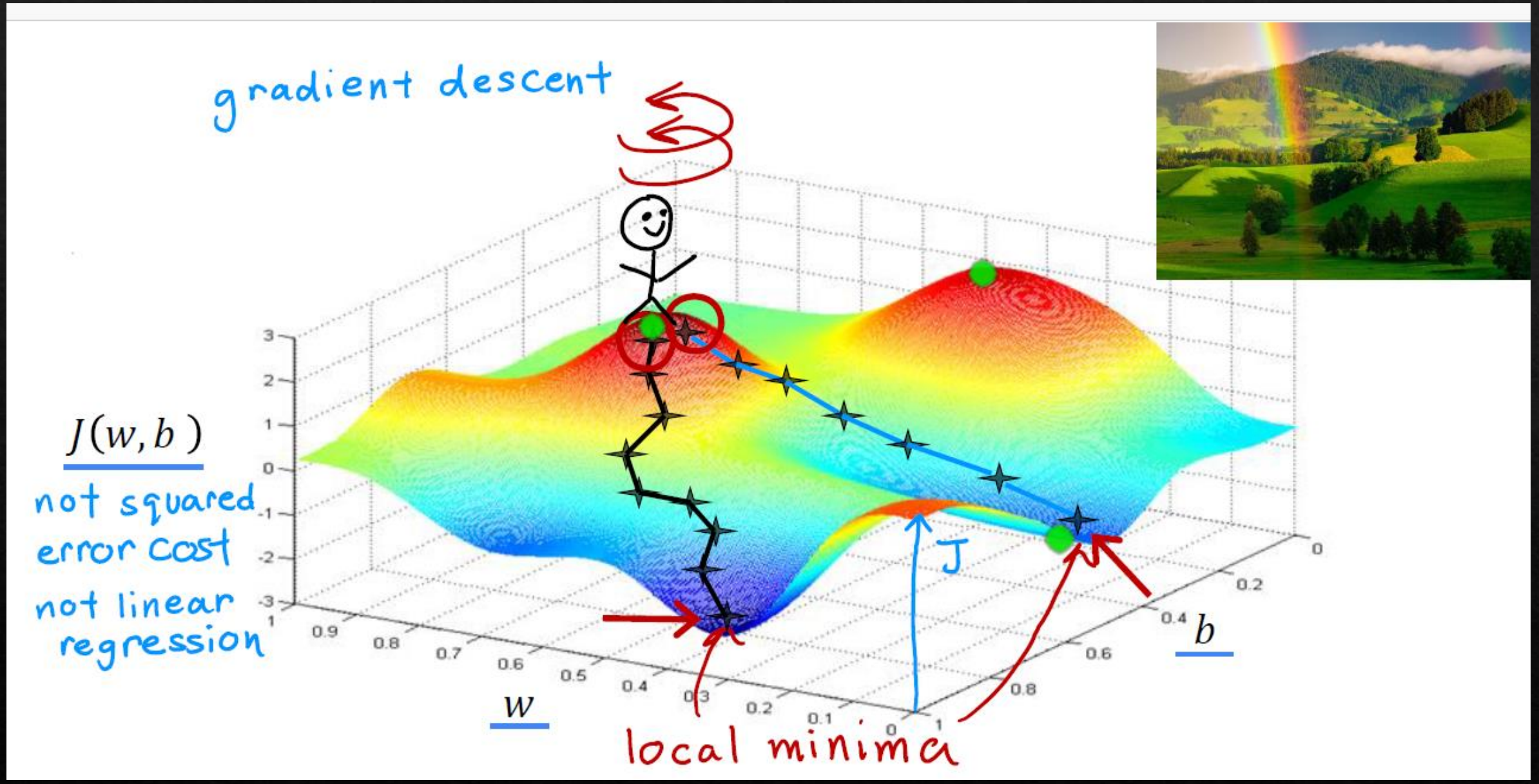
Want $\min_{w,b} J(w, b)$ $\min_{w_1, \ldots, w_n, b} J(w_1, w_2, \ldots, w_n, b)$

Outline:

- Start with some $w, b$ (set $w=0, b=0$)
- Keep changing $w, b$ to reduce $J(w, b)$
- Until we settle at or near a minimum

J not always

may have >1 minimum




Mohamed Elhaj-Abdou

# Gradient descent algorithm

Repeat until convergence

$$w = w - \alpha \frac{d}{dw} J(w,b)$$

$$b = b - \alpha \frac{d}{db} J(w,b)$$

Learning rate

Derivative

Simultaneously update w and b

Assignment

$a = c$

$a = a + 1$

Code

Truth assertion

$a = c$

$a = a + 1$ ✗

Math

a==c

## Correct: Simultaneous update

$$tmp\_w = w - \alpha \frac{\partial}{\partial w} J(w,b)$$

$$tmp\_b = b - \alpha \frac{\partial}{\partial b} J(w,b)$$

$$w = tmp\_w$$

$$b = tmp\_b$$

## Incorrect

$$tmp\_w = w - \alpha \frac{\partial}{\partial w} J(w,b)$$

$$w = tmp\_w$$

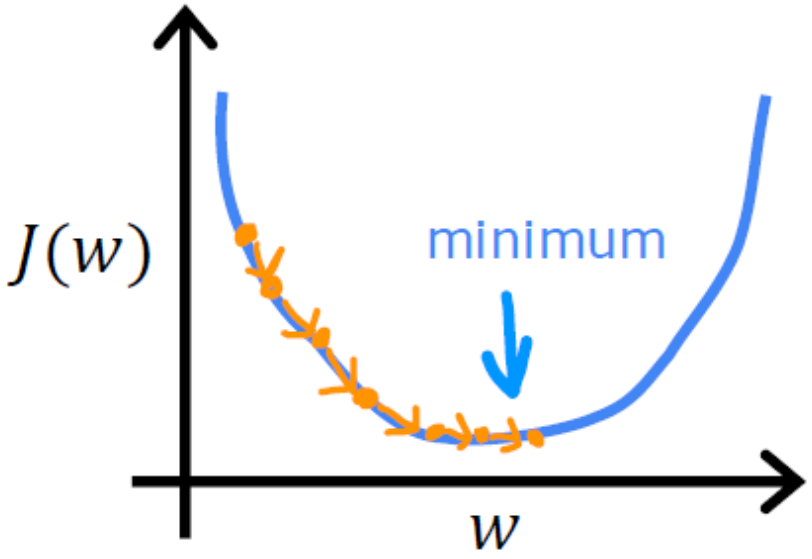
$$tmp\_b = b - \alpha \frac{\partial}{\partial b} J(w,b)$$

$$b = tmp\_b$$

$$w = w - \alpha \frac{d}{dw} J(w)$$

If $\alpha$ is too small...

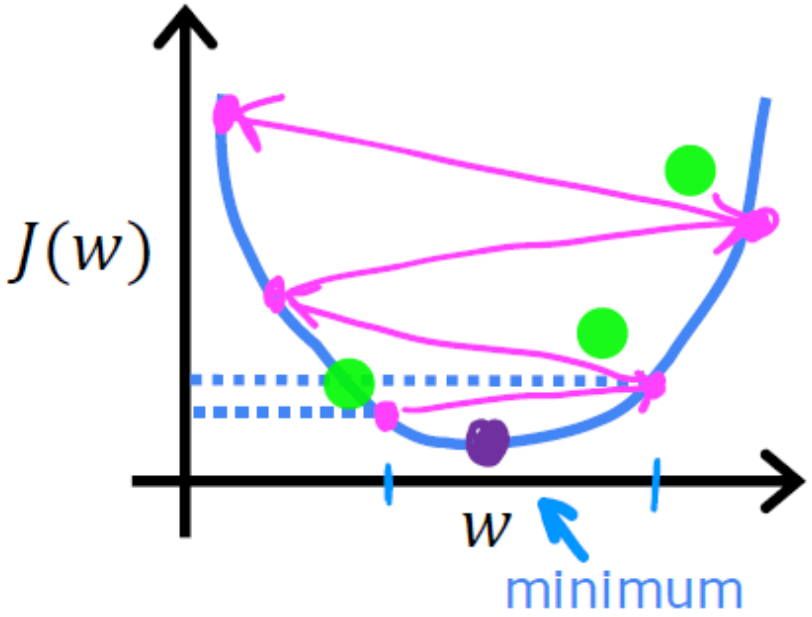
Gradient descent may be slow.



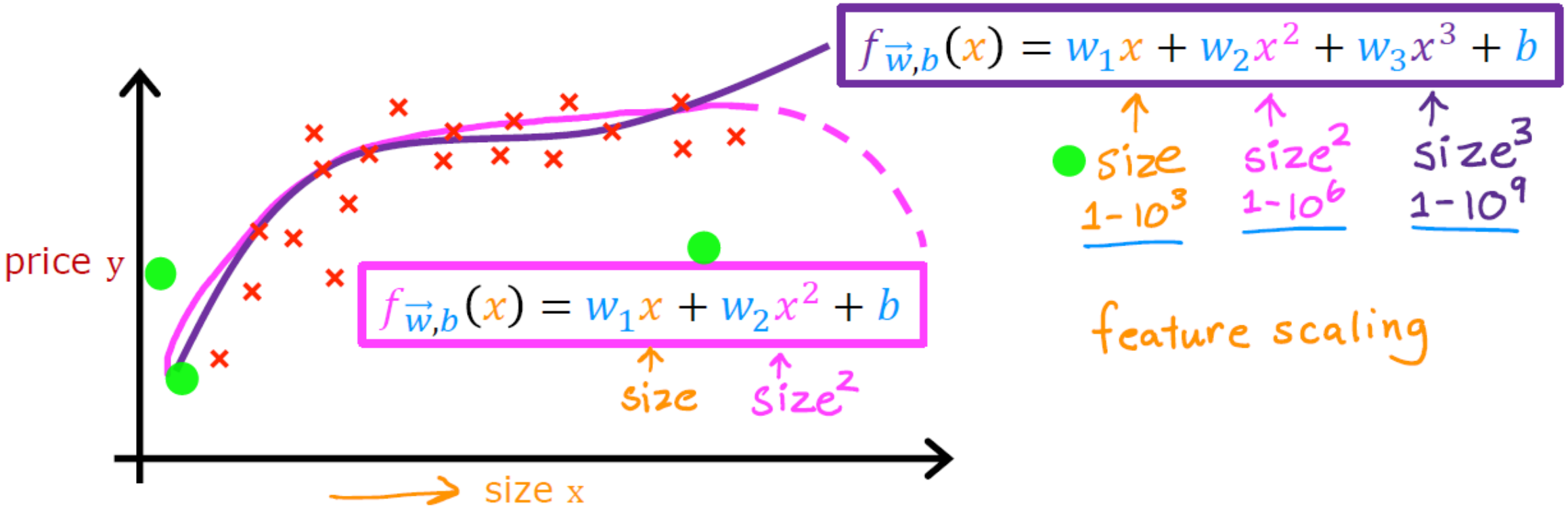

If $\alpha$ is too large...

Gradient descent may:

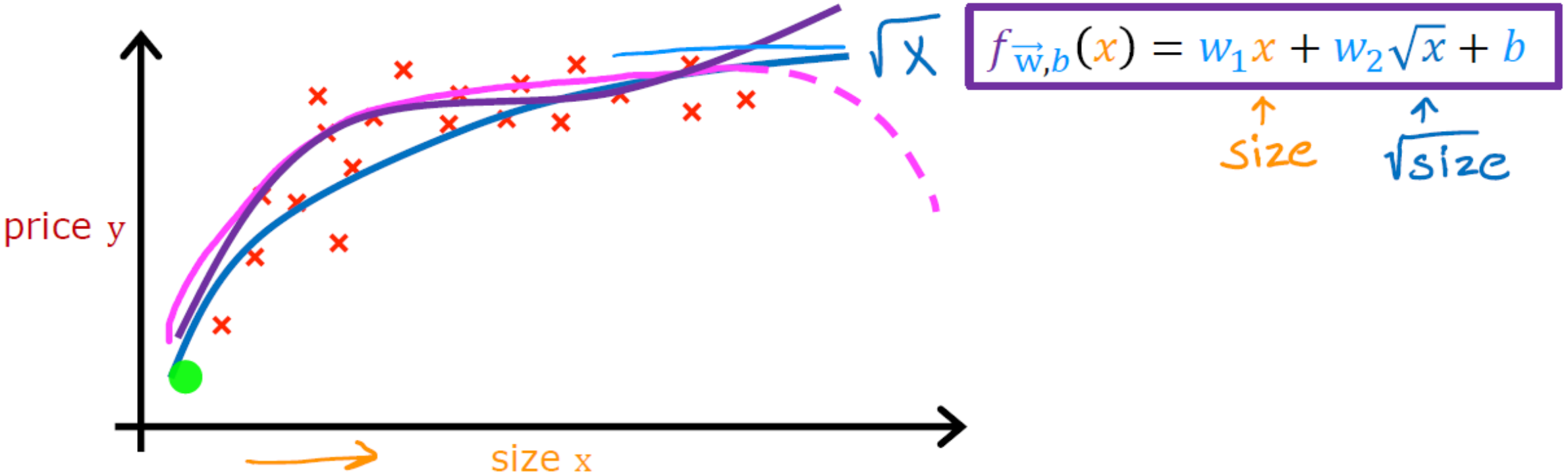
- Overshoot, never reach minimum
- Fail to converge, diverge



# Polynomial regression



# Choice of features



# Multiple features (variables)

one feature →

| Size in feet$^2$ $(x)$ | Price ($) in 1000's $(y)$ ← |
|---|---|
| 2104 | 400 |
| 1416 | 232 |
| 1534 | 315 |
| 852 | 178 |
| ... | ... |

$$f_{w,b}(x) = wx + b$$

# Multiple features (variables)

| | Size in feet² | Number of bedrooms | Number of floors | Age of home in years | Price ($) in $1000's |
|---|---|---|---|---|---|
| | $x_1$ | $x_2$ | $x_3$ | $x_4$ | |
| | 2104 | 5 | 1 | 45 | 460 |
| $i=2$ | 1416 | 3 | 2 | 40 | 232 |
| | 1534 | 3 | 2 | 30 | 315 |
| | 852 | 2 | 1 | 36 | 178 |
| | ... | ... | ... | ... | ... |

$j = 1 \ldots 4$

$n = 4$

$x_j$ = $j^{th}$ feature

$n$ = number of features

$\vec{x}^{(i)}$ = features of $i^{th}$ training example

$x_j^{(i)}$ = value of feature $j$ in $i^{th}$ training example

$\vec{x}^{(2)} = [1416 \ 3 \ 2 \ 40]$

$x_3^{(2)} = 2$

# Model:

Previously: $f_{w,b}(x) = wx + b$

$$f_{w,b}(X) = w_1 X_1 + w_2 X_2 + w_3 X_3 + w_4 X_4 + b$$

example

$$f_{w,b}(X) = 0.1 X_1 + 4 X_2 + 10 X_3 + -2 X_4 + 80$$

$X_1$: size

$X_2$: #bedrooms

$X_3$: #floors

$X_4$: years

80: base price

$$f_{w,b}(x) = w_1 x_1 + w_2 x_2 + \cdots + w_n x_n + b$$

$$f_{\vec{w},b}(\vec{x}) = w_1 x_1 + w_2 x_2 + \cdots + w_n x_n + b$$

$\vec{w} = [w_1 \; w_2 \; w_3 \ldots w_n]$ parameters of the model

$b$ is a number

vector $\vec{x} = [x_1 \; x_2 \; x_3 \ldots x_n]$

$$f_{\vec{w},b}(\vec{x}) = \vec{w} \cdot \vec{x} + b = w_1 x_1 + w_2 x_2 + w_3 x_3 + \cdots + w_n x_n + b$$

dot product

multiple linear regression

(not multivariate regression)

***Victorization method is the method that suites with the large scale dataset with multiple variables***

***Victorization method is a method that implement dot product of the vectors***
*And solving the solution in parallel*

***Victorization method is a method that is bing used in Gradient***
*Descent for Multiple Regression models*

Mohamed Elhaj-Abdou

## Parameters and features

$\vec{w} = [w_1 \quad w_2 \quad w_3]$ $n = 3$

$b$ is a number

$\vec{x} = [x_1 \quad x_2 \quad x_3]$

linear algebra: count from 1

NumPy

w[0] w[1] w[2]

```
w = np.array([1.0,2.5,-3.3])
b = 4
x = np.array([10,20,30])
```

x[0] x[1] x[2]

code: count from 0

## Without vectorization

$$f_{\vec{w},b}(\vec{x}) = \left(\sum_{j=1}^{n} w_j x_j\right) + b$$

$\sum_{j=1}^{n} \rightarrow j = 1 \ldots n$ → 1, 2, 3

range(0, n) → j = 0 ... n-1

```
f = 0          range(n)
for j in range(0,n):
    f = f + w[j] * x[j]
f = f + b
```

## Without vectorization $n = 100,000$

$$f_{\vec{w},b}(\vec{x}) = w_1 x_1 + w_2 x_2 + w_3 x_3 + b$$

```
f = w[0] * x[0] +
    w[1] * x[1] +
    w[2] * x[2] + b
```

## Vectorization

$$f_{\vec{w},b}(\vec{x}) = \vec{w} \cdot \vec{x} + b$$

```
f = np.dot(w,x) + b
```

## Without vectorization

```
for j in range(0,16):
    f = f + w[j] * x[j]
```

$t_0$

```
f + w[0] * x[0]
```

$t_1$

```
f + w[1] * x[1]
```

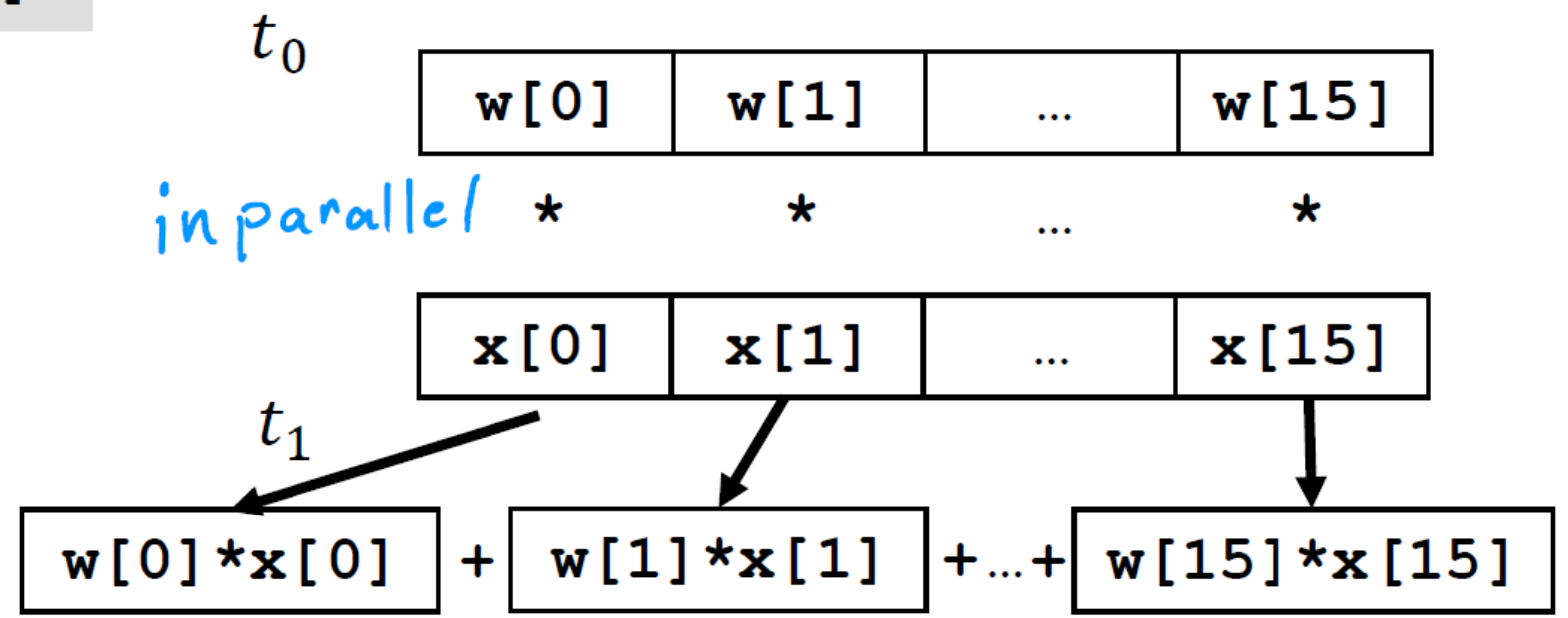
...

$t_{15}$

```
f + w[15] * x[15]
```

## Vectorization

```
np.dot(w,x)
```




efficient → scale to large datasets

# Feature engineering

$$f_{\vec{w},b}(\vec{x}) = w_1 x_1 + w_2 x_2 + b$$

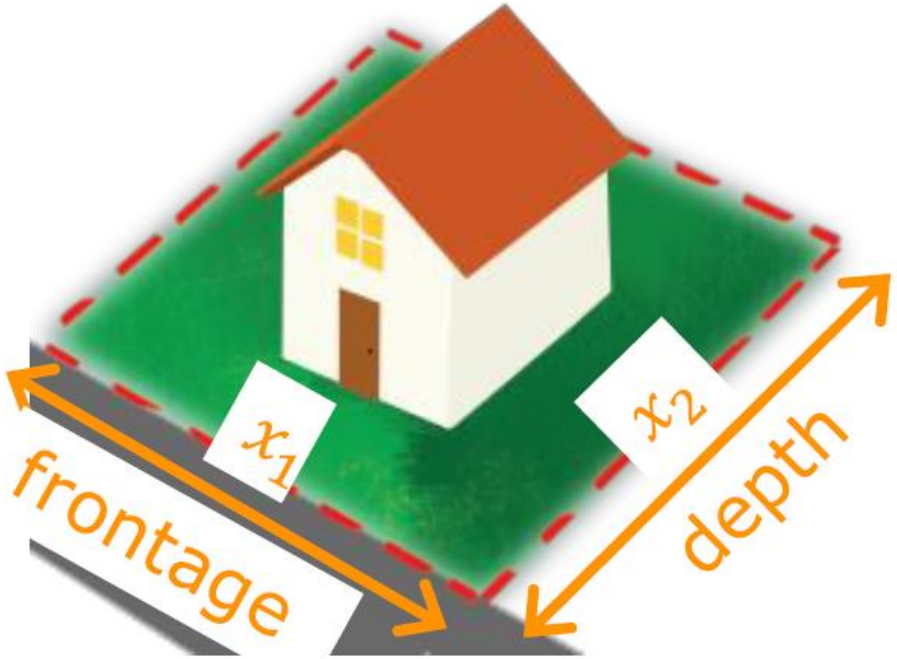
frontage depth

$$area = frontage \times depth$$

$$x_3 = x_1 x_2$$

new feature

$$f_{\vec{w},b}(\vec{x}) = w_1 x_1 + w_2 x_2 + w_3 x_3 + b$$




Feature engineering:
Using intuition to design new features, by transforming or combining original features.