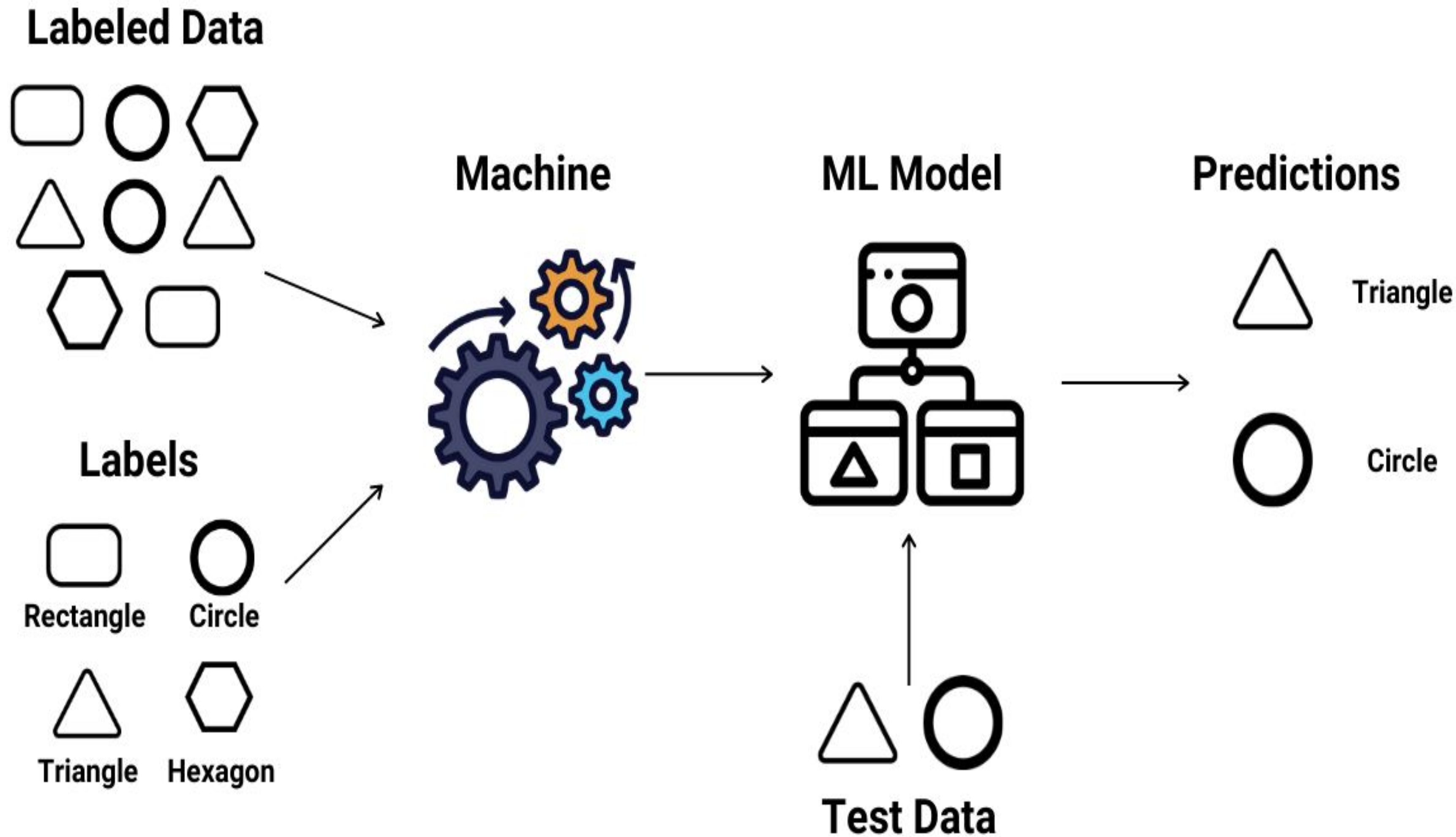
Supervised Machine Learning Algorithm

Linear Regression

What is supervised learning?

- Use of labeled datasets to train algorithms that to classify data or predict outcomes accurately. As input data is fed into the model, it adjusts its weights until the model has been fitted appropriately.
- Supervised learning helps organizations solve for a variety of real-world problems at scale, such as classifying spam in a separate folder from your inbox.



Labeled Data
Machine
ML Model
Predictions
Triangle
Circle
Labels
Rectangle
Circle
Triangle
Hexagon
Test Data

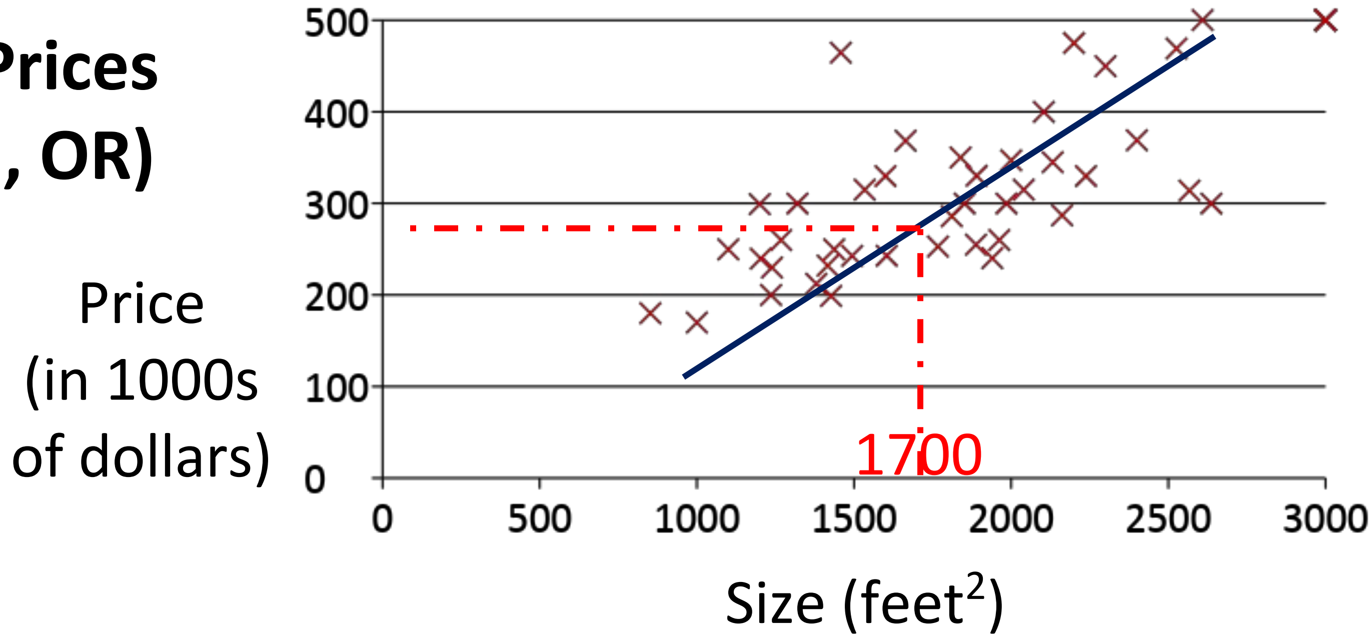
Supervised Machine Learning Algorithms:

- Linear Regression
- Logistic Regression
- Decision Tree
- K Nearest Neighbors
- Random Forest
- Naive Bayes
- SVM
- ANN

Linear regression with one variable

Model representation

Housing Prices (Portland, OR)



<u>Supervised Learning</u>

Given the “right answer” for each example in the data.

<u>Regression Problem</u>

Predict real-valued output

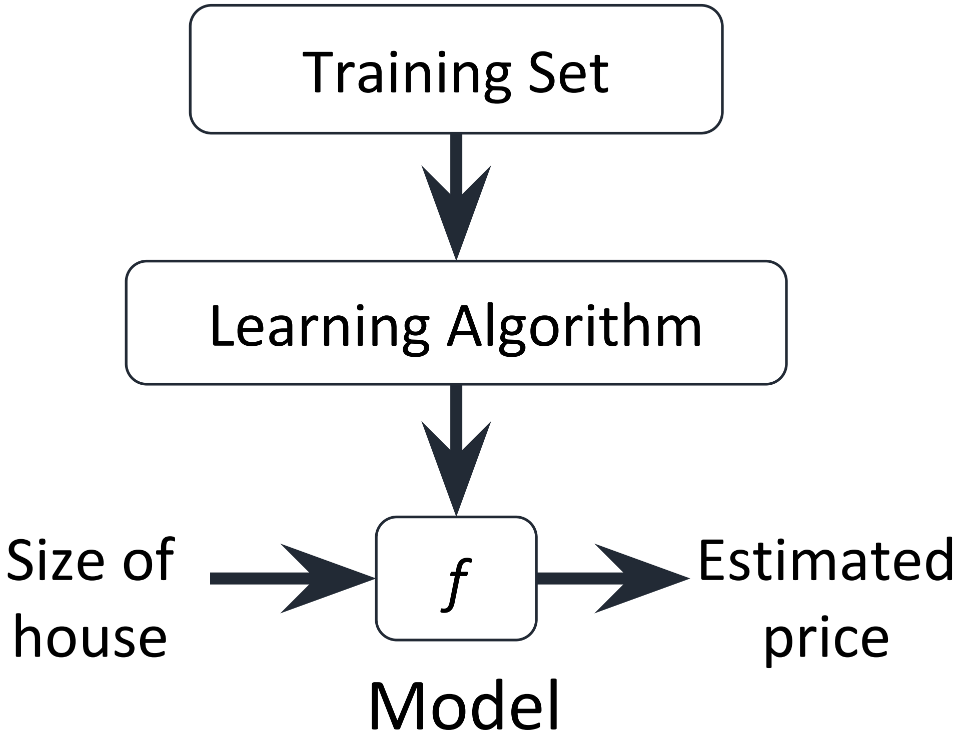
Training set of housing prices

Size in feet^2 (x)	Price ($) in 1000's (y)
2104	460
1416	232
1534	315
852	178
…	…

Notation:

m = Number of training examples
x's = "input" variable / features
y's = "output" variable / "target" variable

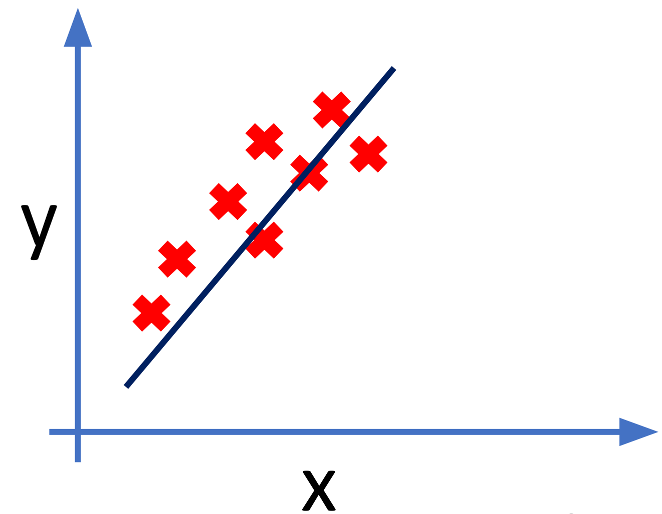
(x,y) → One Training Example
(x^i, y^i)→ ith Training Example



f maps from x to y

How do we represent f ?

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



Linear regression with one variable.

Univariate linear regression.

How to Choose $w$and b to best fit the line→

Cost Function

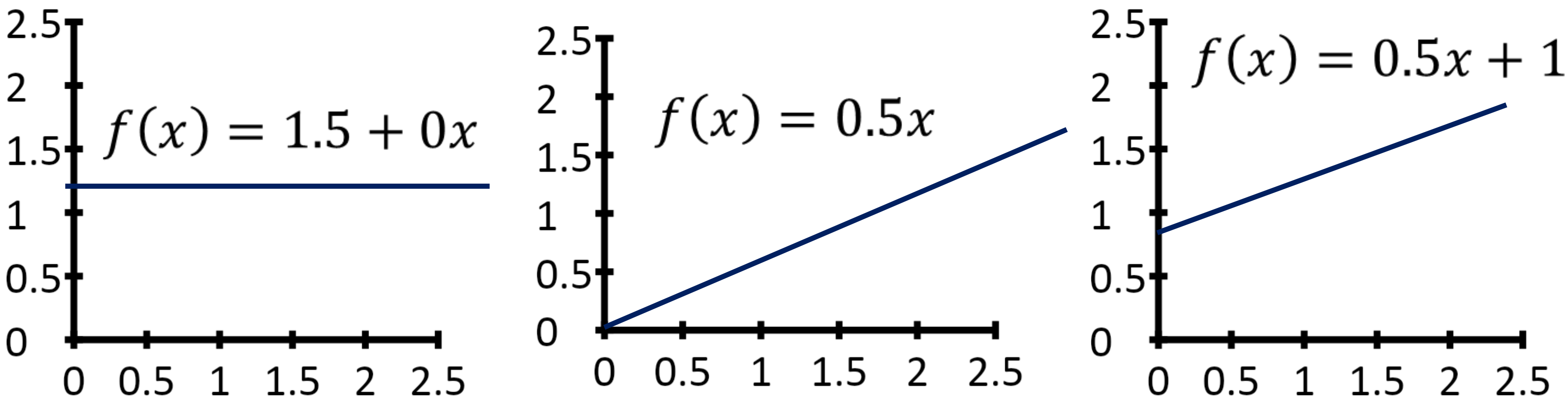
Size in feet^2 (x)	Price ($) in 1000's (y)
2104	460
1416	232
1534	315
852	178
…	…

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

w,b are called parameters/ coefficients/weights

What this w and b do in Machine Learning Algorithm?

$$f(x) = wx + b$$



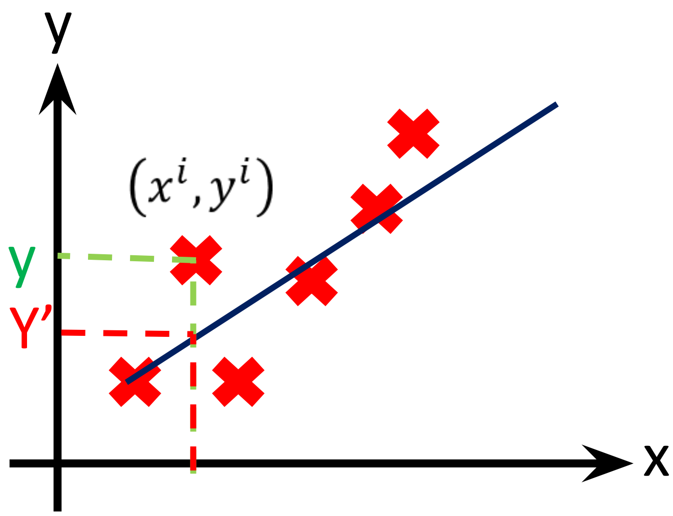
$w = 0$, $b = 1.5$

$w = 0.5$, $b = 0$

$w = 0.5$, $b = 1$

$$y' = f_{w,b}(x)$$

Cost Function



$$y' = f_{w,b}(x)$$

$$f(x) = wx + b$$

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

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

Squared (Sum of squares) error Cost Function

Goal: To minimize $J(w,b)$

Find w, b:

y' is closed to y for all (x^i, y^i)

Cost Function Intuition: w

simplifed example

$$f(x) = wx$$

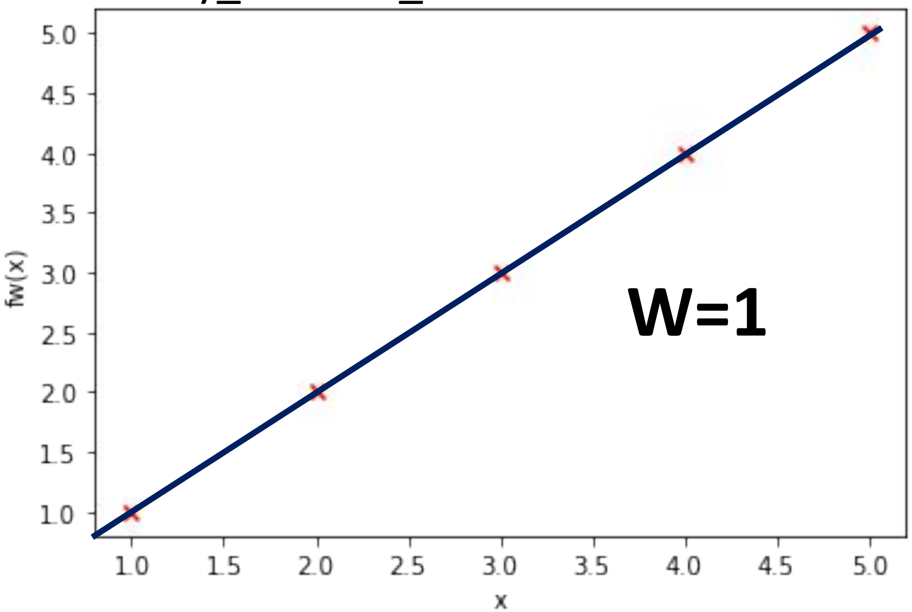
$$J(w) = \frac{1}{2m}\sum_{i=1}^{m}\left(f_w(x^{(i)}) - y^{(i)}\right)^2$$

Goal: To minimize $J(w)$

$$f(x) = wx$$

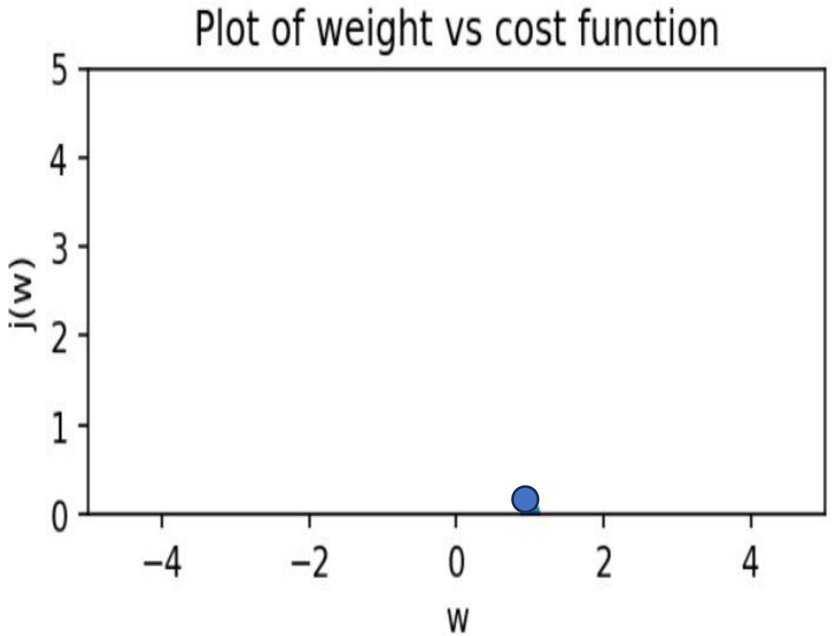
Function of x (input)

x_train = np.array([1,2,3,4,5])
y_train = x_train



$$J(w)$$

Function of w (parameter)



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

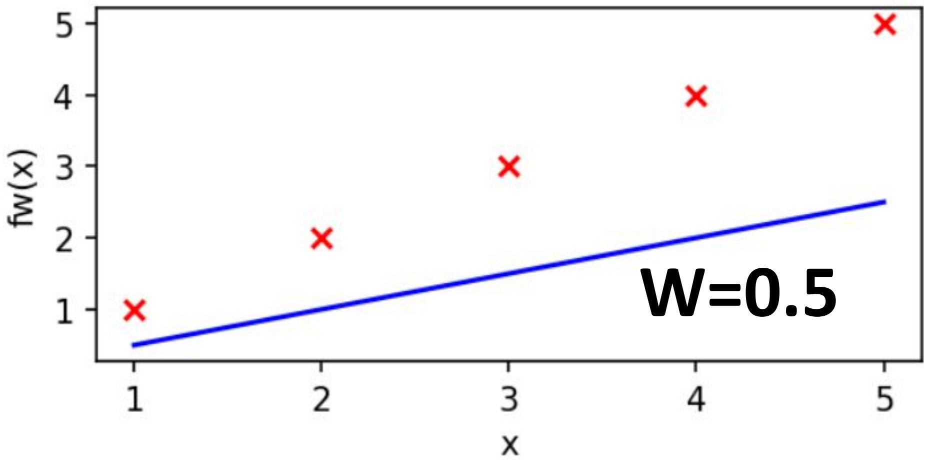
$$J(1) = \frac{1}{2*5}(0^2 + 0^2 + 0^2 + 0^2 + 0^2)$$

$$J(1) = 0$$

$$f(x) = wx$$

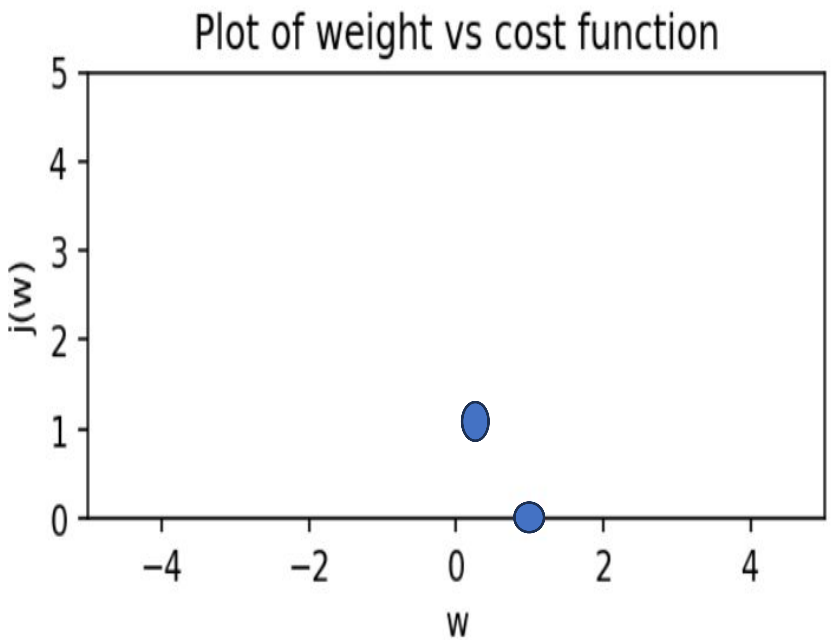
Function of x (input)

x_train = np.array([1,2,3,4,5])
y_train = x_train



$$J(w)$$

Function of w (parameter)



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

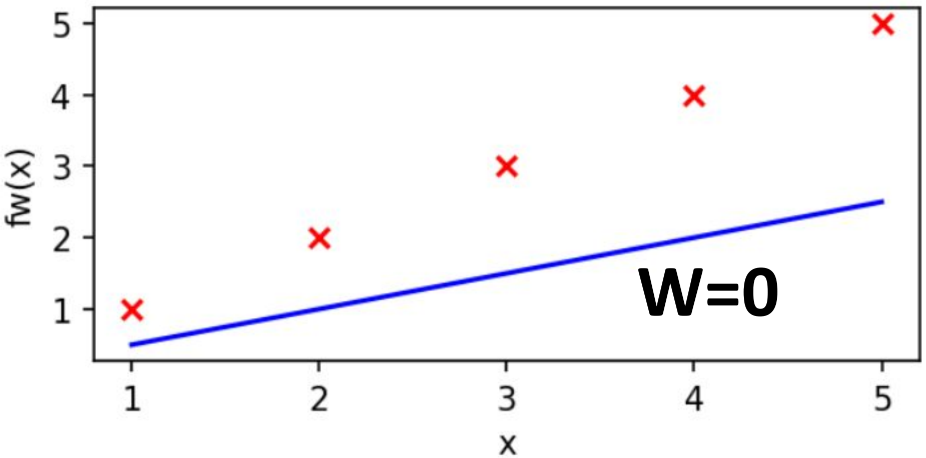
$$J(0.5) = \frac{1}{2*5}(0.25 + 1 + 2.25 + 4 + 6.25)$$

$$J(0.5) = 1.375$$

$$f(x) = wx$$

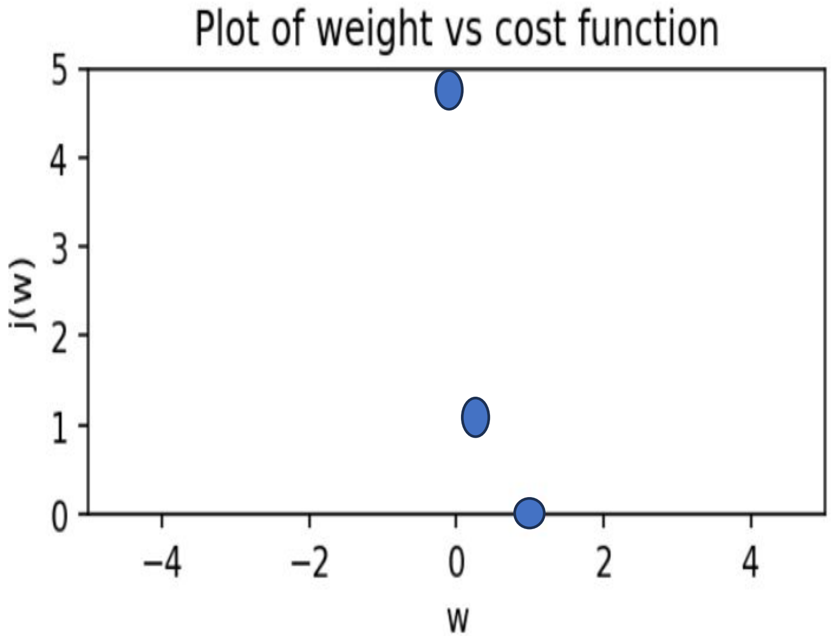
Function of x (input)

x_train = np.array([1,2,3,4,5])
y_train = x_train



$$J(w)$$

Function of w (parameter)



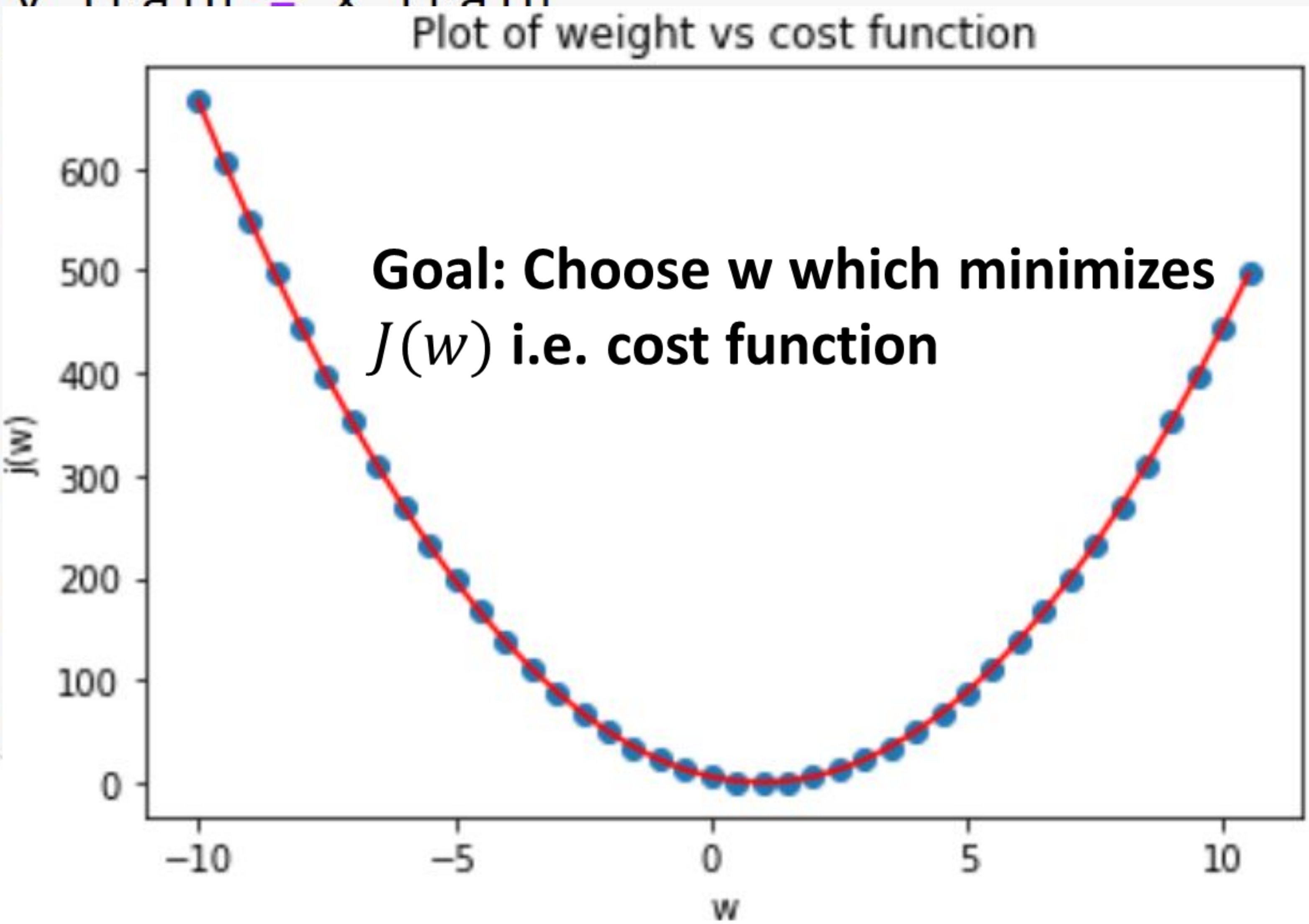
$$J(w) = \frac{1}{2m}\sum_{i=1}^{m}\left(f_w(x^{(i)}) - y^{(i)}\right)^2 \quad J(w) = \frac{1}{2m}\sum_{i=1}^{m}\left(wx^{(i)} - y^{(i)}\right)^2$$

$$J(0) = \frac{1}{2*5}(1 + 4+ 9+ 16+ 25)$$

$$J(0) = 5.5$$

```
#const function plotting

x_train = np.array([1,2,3,4,5])
y_train = x_train
```

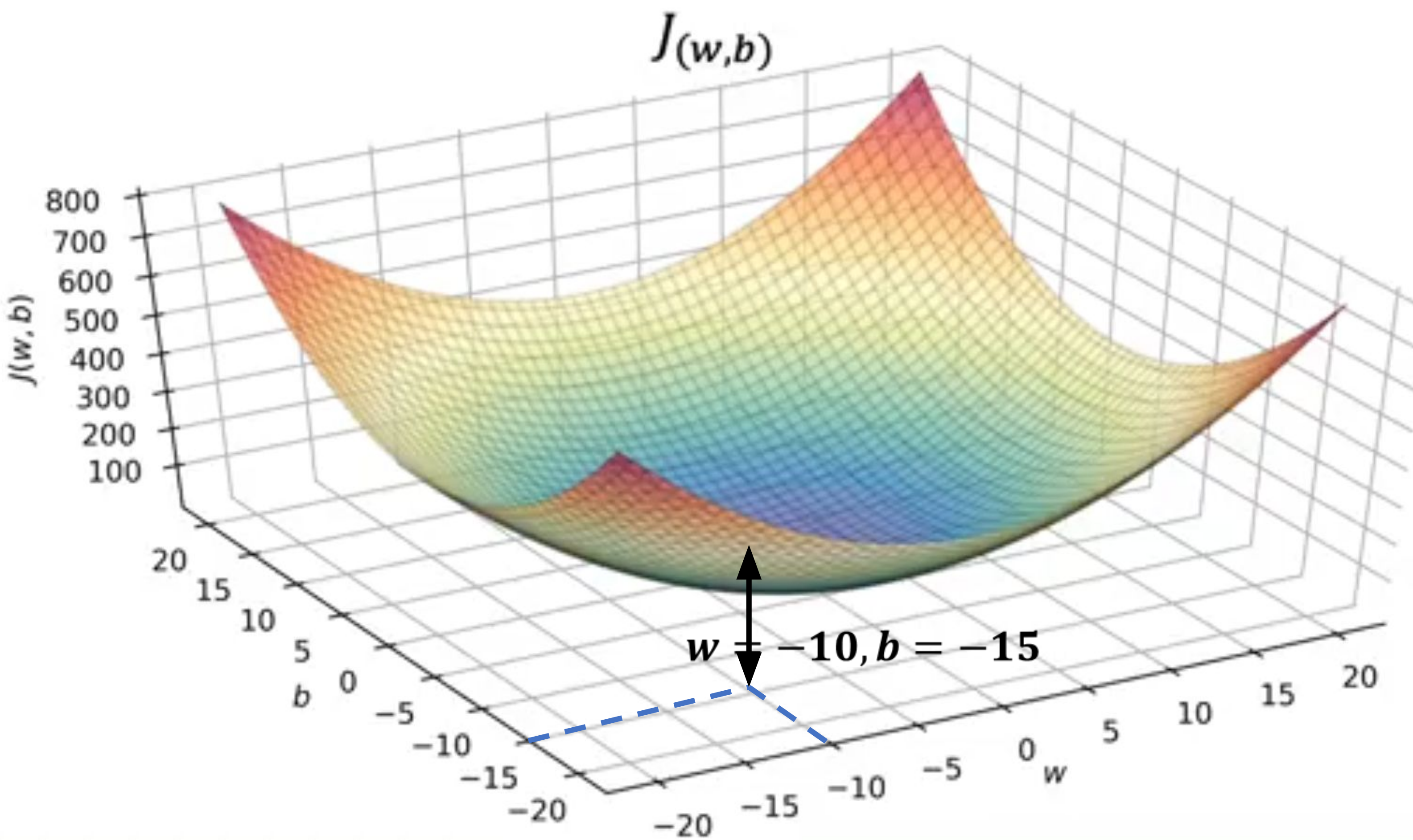


Goal: Choose w which minimizes $J(w)$ i.e. cost function

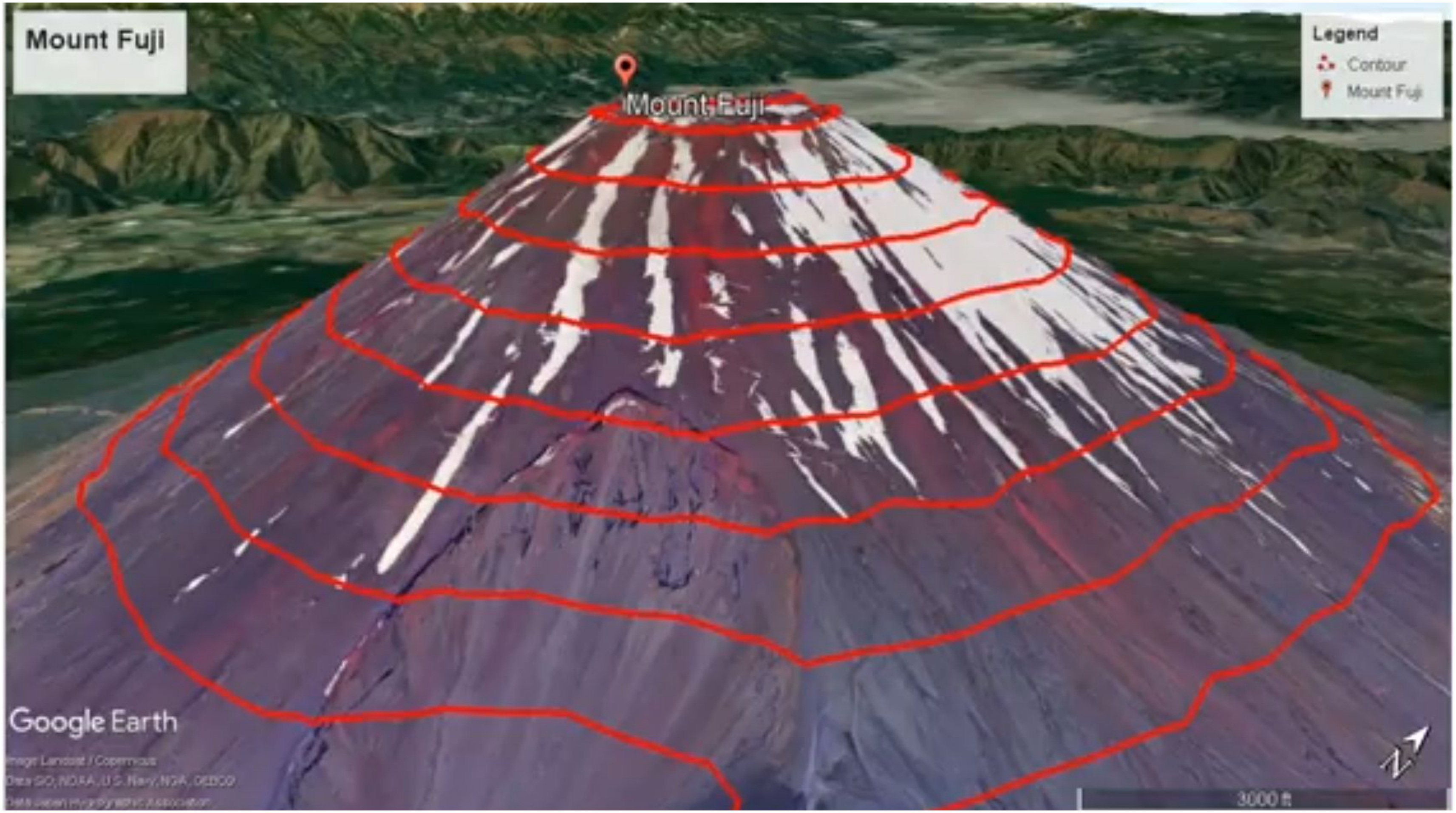
```
x_train)
edict))
n")
```

Model	$f_{w,b}(x) = wx + b$
Parameters	w, b
Cost Function	$J(w,b) = \frac{1}{2m}\sum_{i=1}^{m}(f_{w,b}(x^{(i)}) - y^{(i)})^2$
Objective	$\underset{w,b}{\text{minimize}}\, J(w,b)$

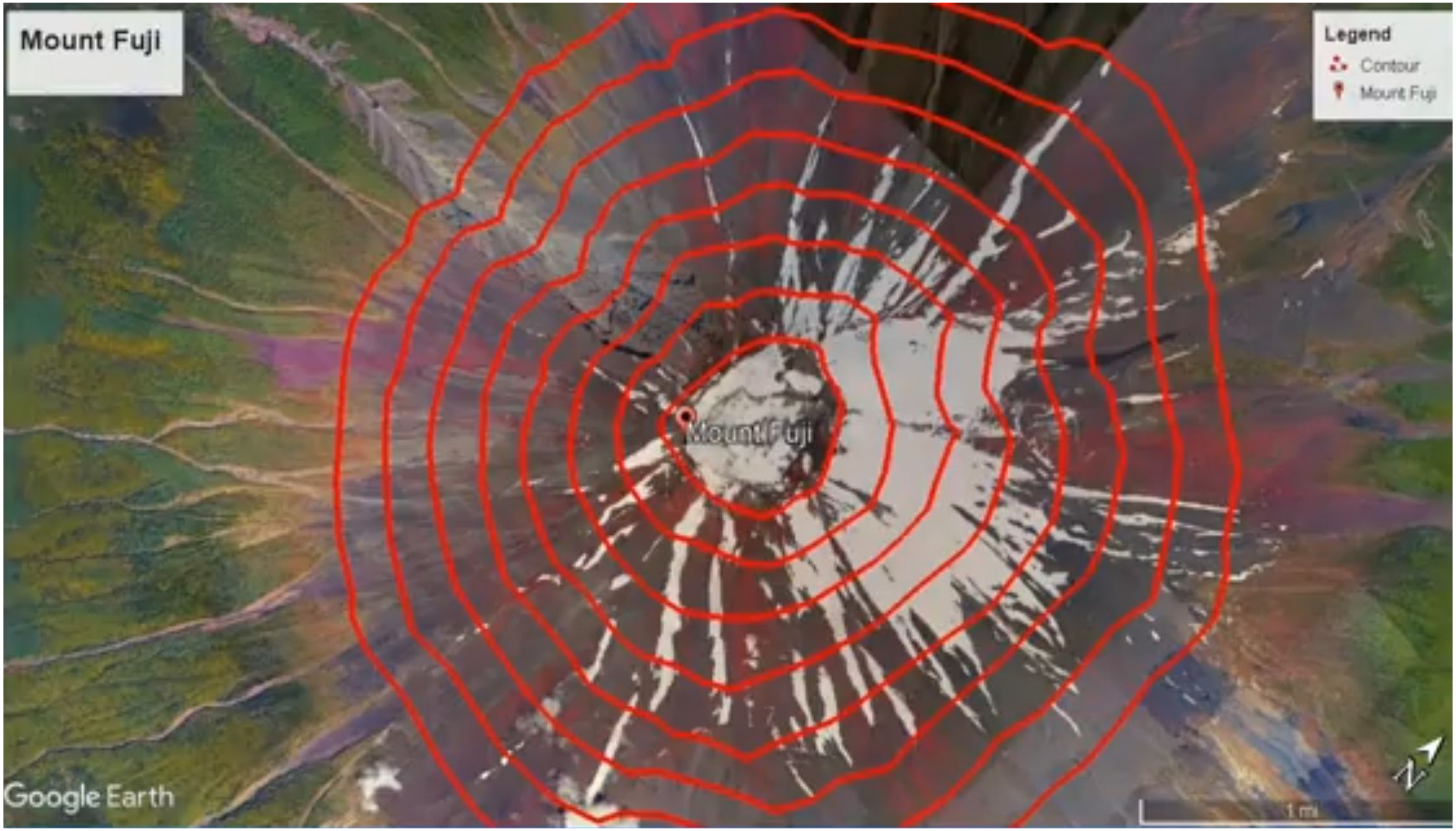
Cost Function Intuition: w,b

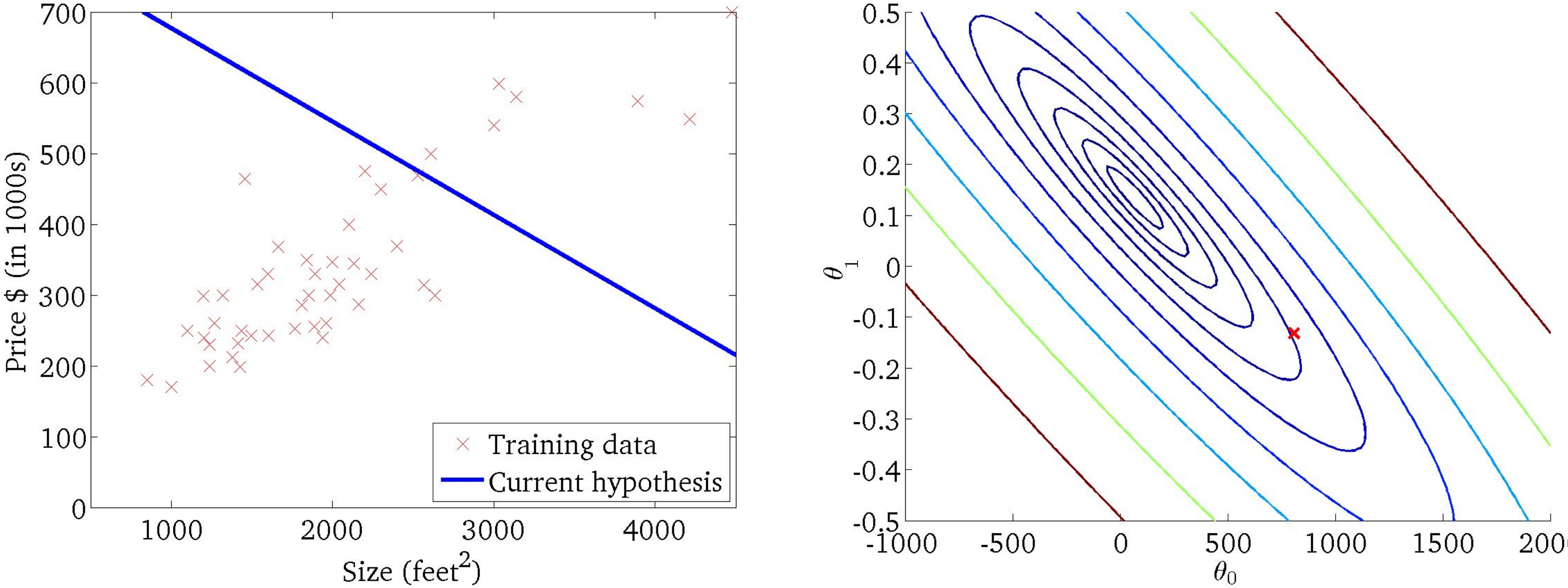


Contour plot

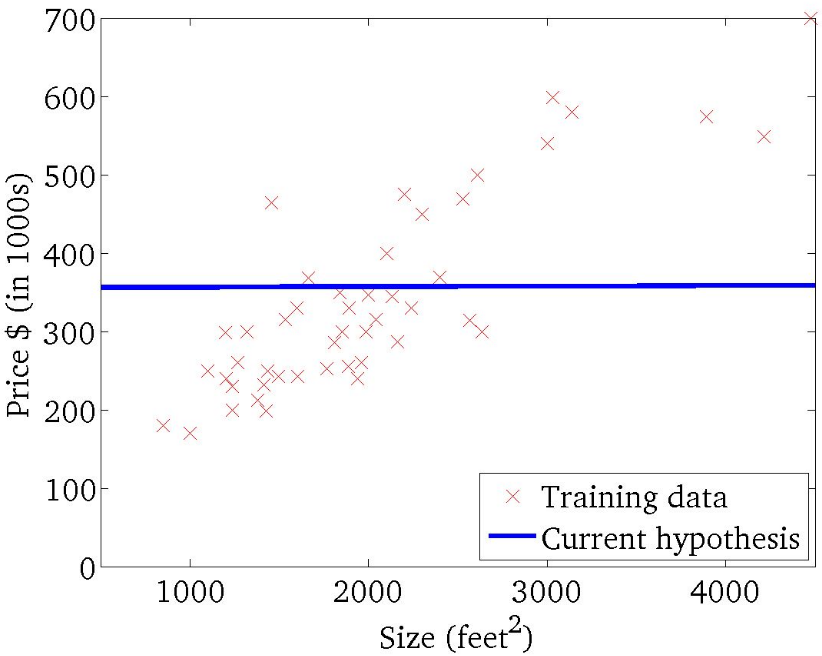


Contour plot

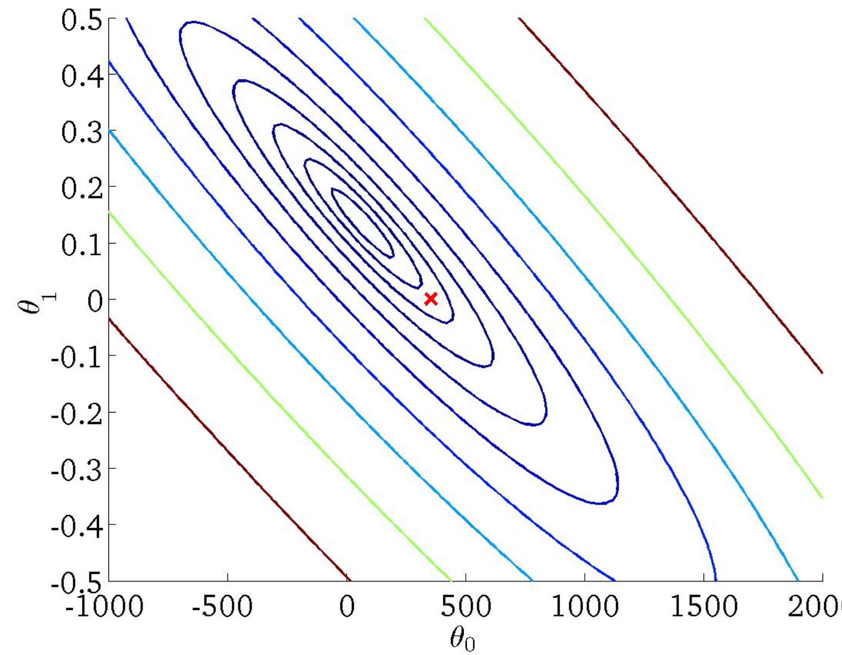




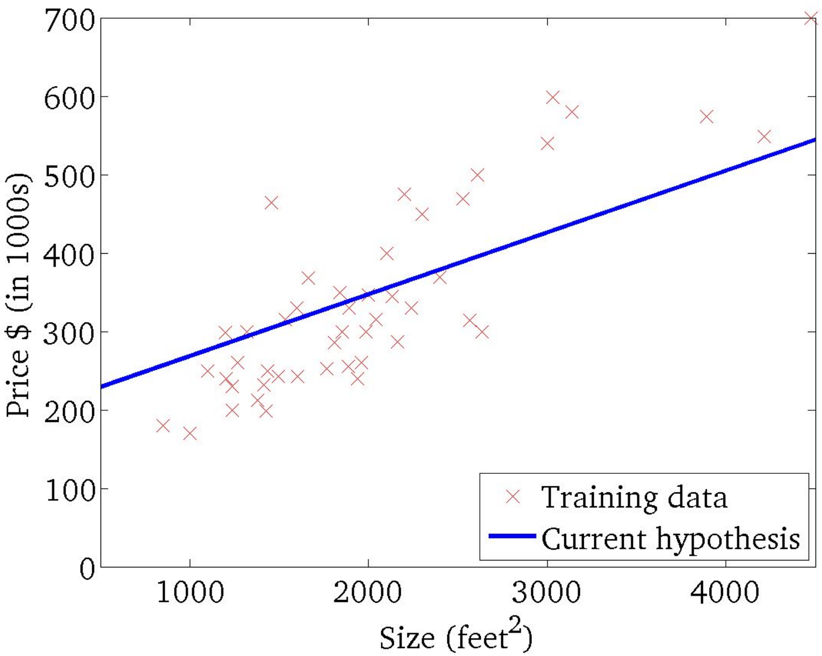
Price $ (in 1000s)
Size (feet^2)
Training data
Current hypothesis
θ_1
θ_0



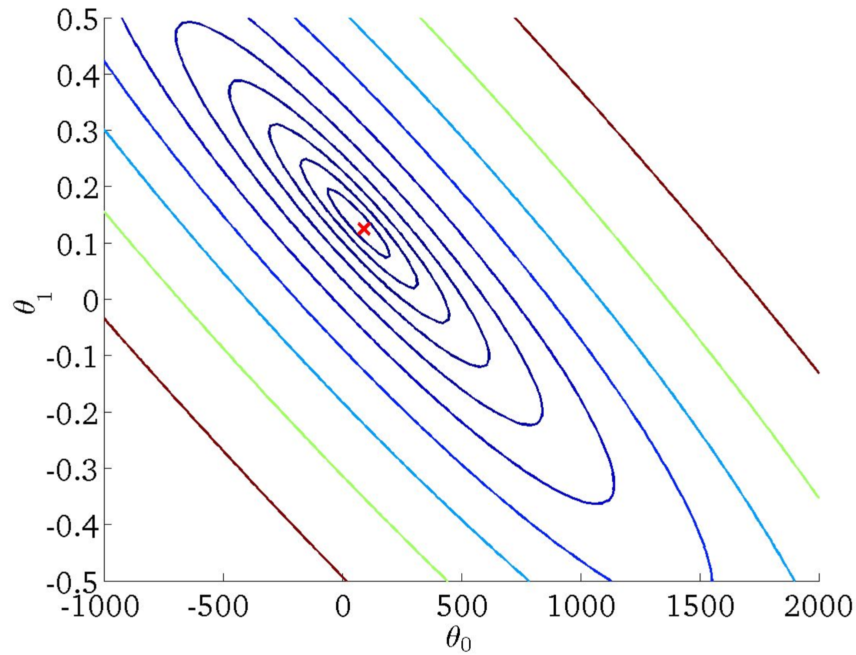
700
600
500
400
300
200
100
0
1000
2000
3000
4000
Price $ (in 1000s)
Size (feet^2)
Training data
Current hypothesis



0.5
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
-0.5
-1000
-500
0
500
1000
1500
2000
θ_1
θ_0



700
600
500
400
300
200
100
0
Price $ (in 1000s)
1000
2000
3000
4000
Size (feet^2)
Training data
Current hypothesis



0.5
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
-0.5
θ_1
-1000
-500
0
500
1000
1500
2000
θ_0

Machine Learning

Gradient descent in machine learning is simply used to find the values of a function's parameters (coefficients) that minimize a cost function as far as possible

Linear regression with one variable

Gradient descent

Gradient descent

You have $J(w, b)$

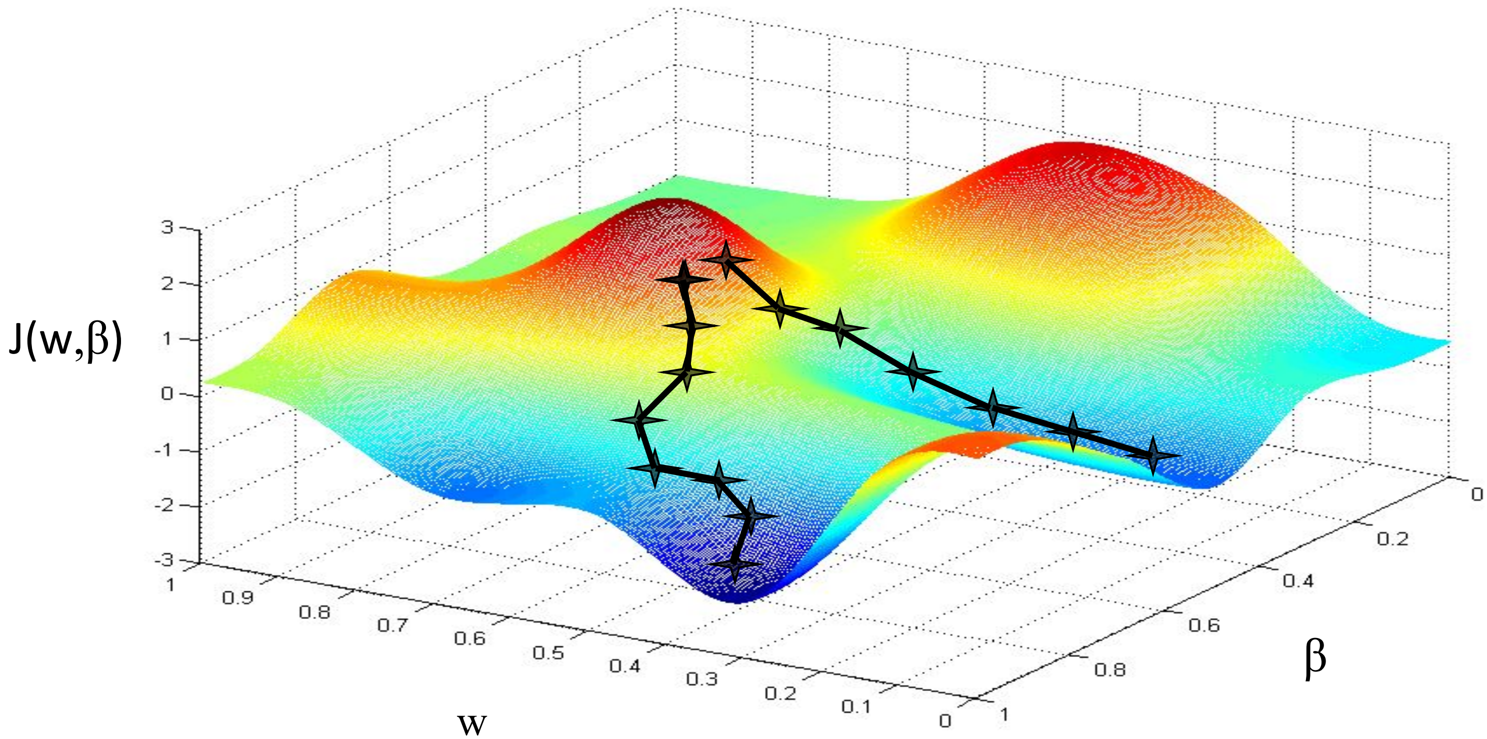
Goal: Minimize $J(w, b)$

Algorithm:

Start with some value of w, b (common choice w=0, b=0)

Keep changing w, b to reduce J(w,b)

Until we settle at or near minimum



J(w,β)
3
2
1
0
-1
-2
-3
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
w
0
0.2
0.4
0.6
0.8
1
β

Gradient descent: Algorithm

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

$$w = w - \alpha \frac{\partial}{\partial w} \left(\frac{1}{2m} \sum_{i=1}^{m} \left(f_w(x^{(i)}) - y^{(i)} \right)^2 \right)$$

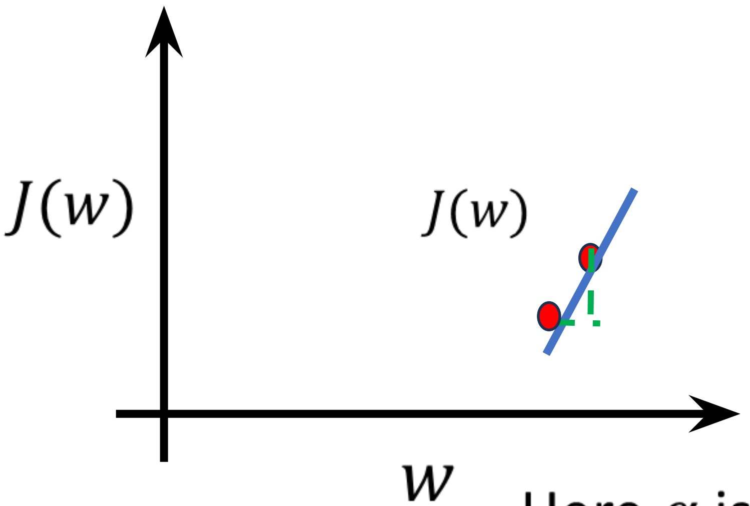
$$w = w - \alpha \frac{\partial}{\partial w} \left(\frac{1}{2m} \sum_{i=1}^{m} \left((w * x^{(i)} + b) - y^{(i)} \right)^2 \right)$$

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

$$\mathrm{b} = b - \alpha \frac{\partial}{\partial b} \left(\frac{1}{2m} \sum_{i=1}^{m} \left((w * x^{(i)} + b) - y^{(i)} \right)^2 \right)$$

Update of w and b should be simultaneous.

Why Partial Derivative?



$$w' = w - \alpha \frac{\partial}{\partial w} \mathrm{J(w)}$$

$$w' = w - \alpha(+ve\ value)$$

$$w' < w$$

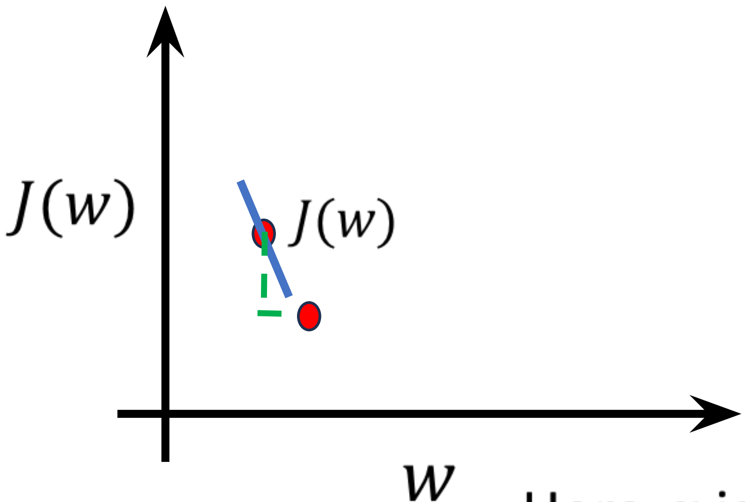
Here α is the learning rate. One of the hyper parameter. That controls to change of w

Why Partial Derivative?

$$w' = w - \alpha \frac{\partial}{\partial w} \text{J(w)}$$

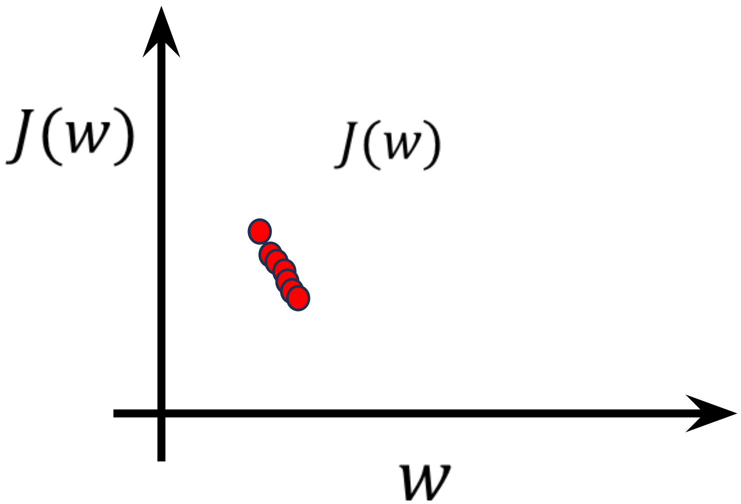
$$w' = w - \alpha(-ve\ value)$$

$$w' > w$$



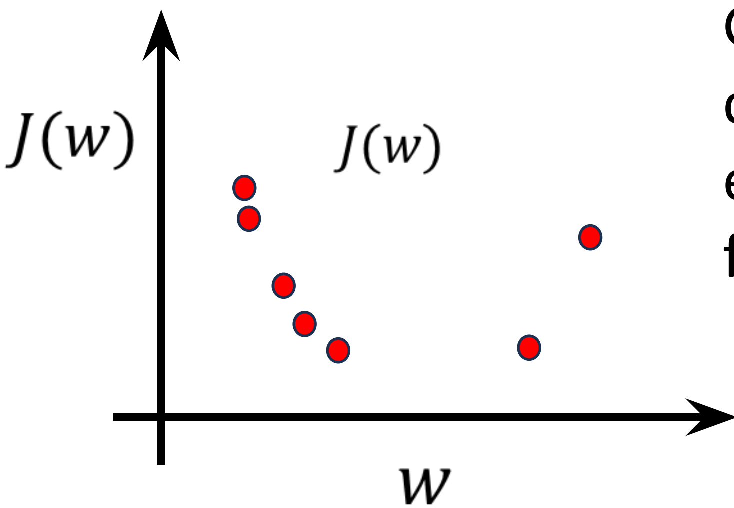
Here α is the learning rate. One of the hyper parameter. That controls to change of w

How to Choose Learning rate?



If α is too small, gradient descent can be slow.

How to Choose Learning rate?



Gradient descent can converge to a local minimum, even with the learning rate α fixed.

As we approach a local minimum, gradient descent will automatically take smaller steps. So, no need to decrease α over time.

If α is too large, gradient descent can overshoot the minimum. It may fail to converge, or even diverge.

Linear Regression with multiple variables

Multiple features

Machine Learning

Multiple features (variables).

Size (feet^2) X_1	Number of bedrooms X_2	Number of floors X_3	Age of home (years) X_4	Price ($1000)
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
…	…	…	…	…

Notation:

n = number of features ⍰ n=4

$x^{(i)}$ = input (features) of i^{th} training example. ⍰ Vector

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

Model:

Previously: $f(x) = wx + b$

$$f(x) = w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + b$$

$$f(x) = b \quad x_0 + w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4$$

$$x_0 = 1$$

$$f(x) = \vec{w}\vec{x} + b$$

$$f(x) = W^T X + b$$

Multivariate linear regression.

Multivariate linear regression: Cost Function

$$f(x) = bx_0 + w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4$$

$$x_0 = 1$$

$$f(x) = \vec{w}\vec{x} + b$$

$$J(w_1, w_2, w_3, w_4, b) = \frac{1}{2m}\sum_{i=1}^{m}\left(y'^{(i)} - y^{(i)}\right)^2$$

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

Linear Regression with multiple variables

Gradient descent for multiple variables

Machine Learning

Gradient descent: $W = [w_1, w_2, w_3, w_4]$

Repeat $\{$

$$w_j = w_j - \alpha \frac{\partial}{\partial w_j} \mathrm{J}(W, b)$$

(simultaneously update for every j=1,….n)

$$b = b - \alpha \frac{\partial}{\partial b} \mathrm{J}(W, b)$$

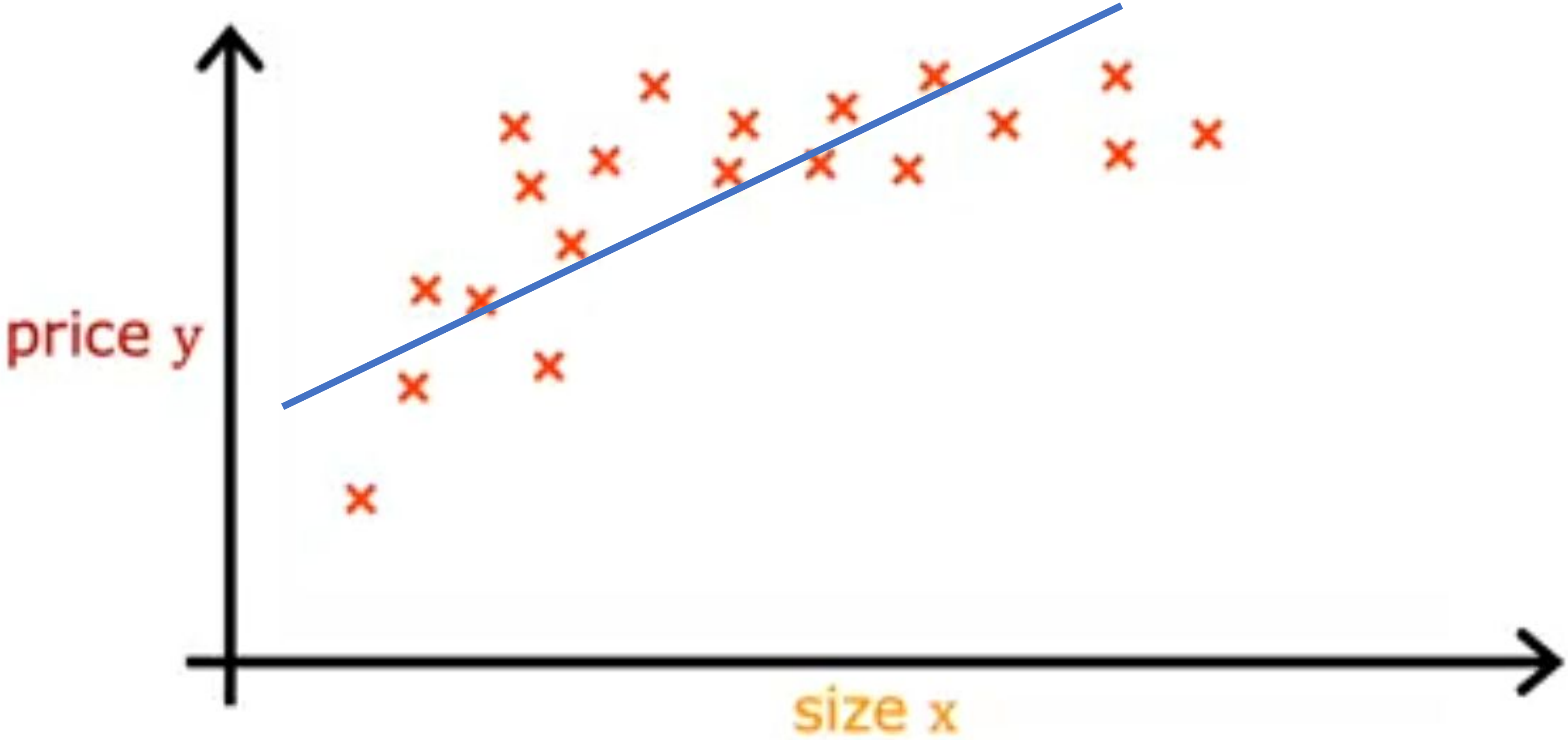
$\}$

Alternative to gradient descent

- Normal equation
  - Only for linear regression
  - Solve for w, b without iterations.
- Disadvantages:
  - Doesn’t generalized to other learning algorithm
  - Method is quite slow if no. of features are large (>10,000)

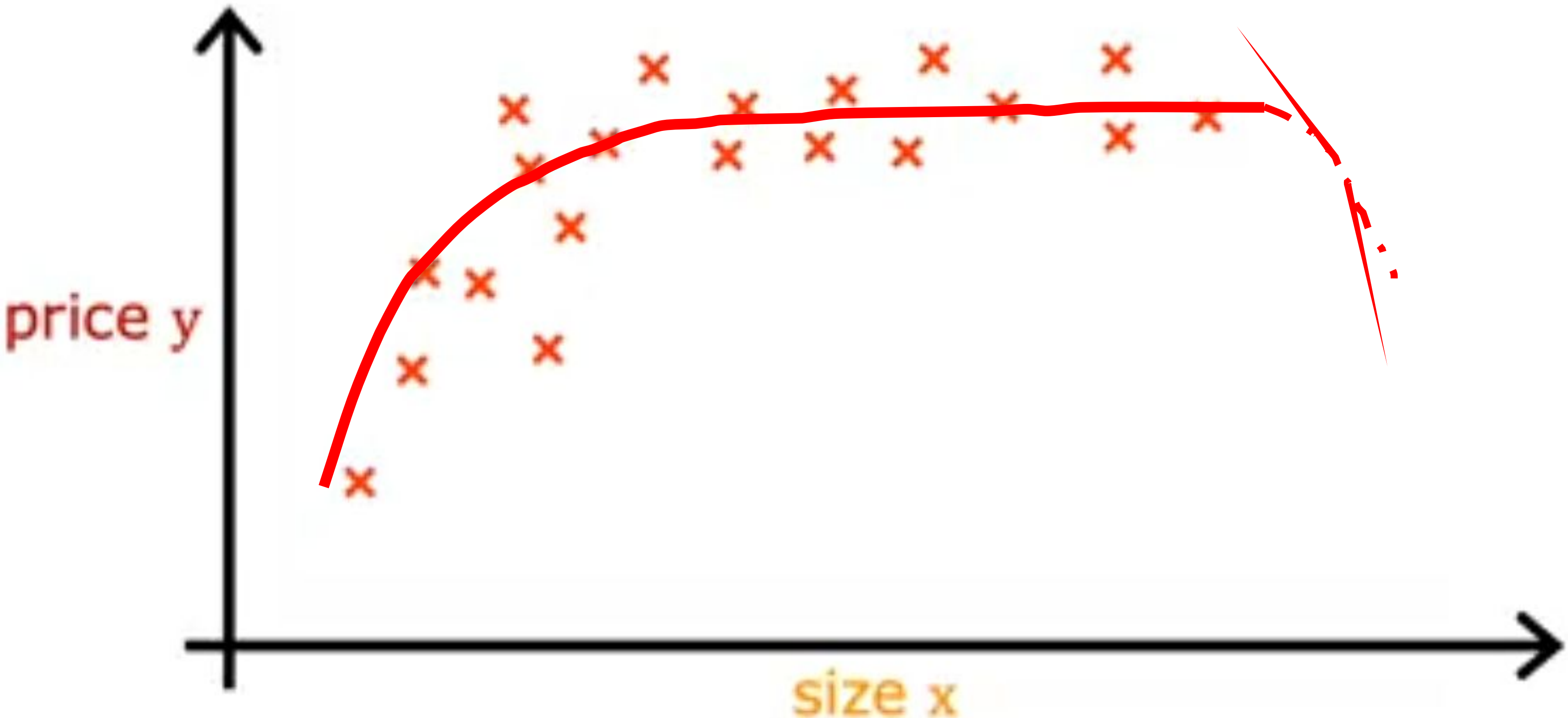
Polynomial Regression

Not very efficient



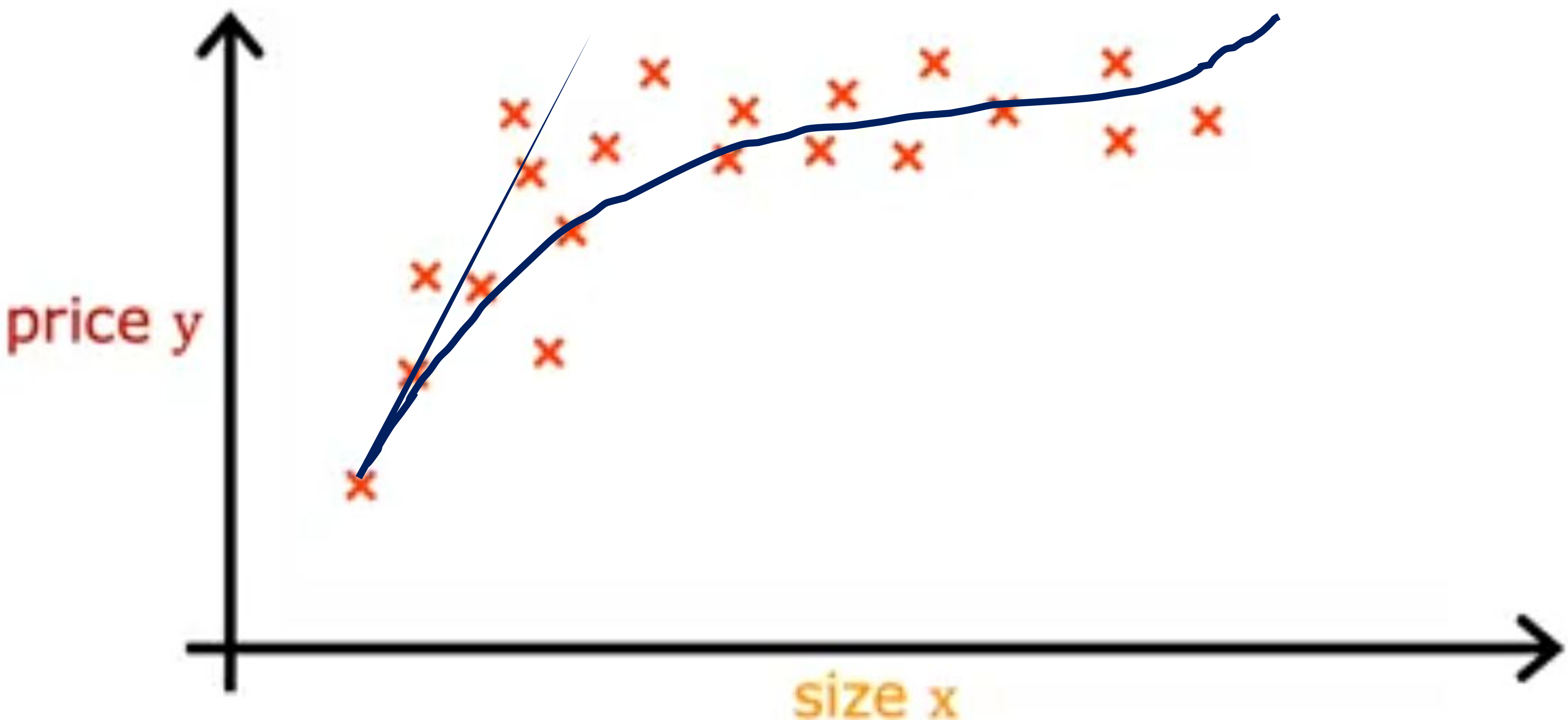
Polynomial Regression

$$f_{(w1,w2,b)}(x) = w_1 x \quad + w_2 x^2 \qquad + b$$



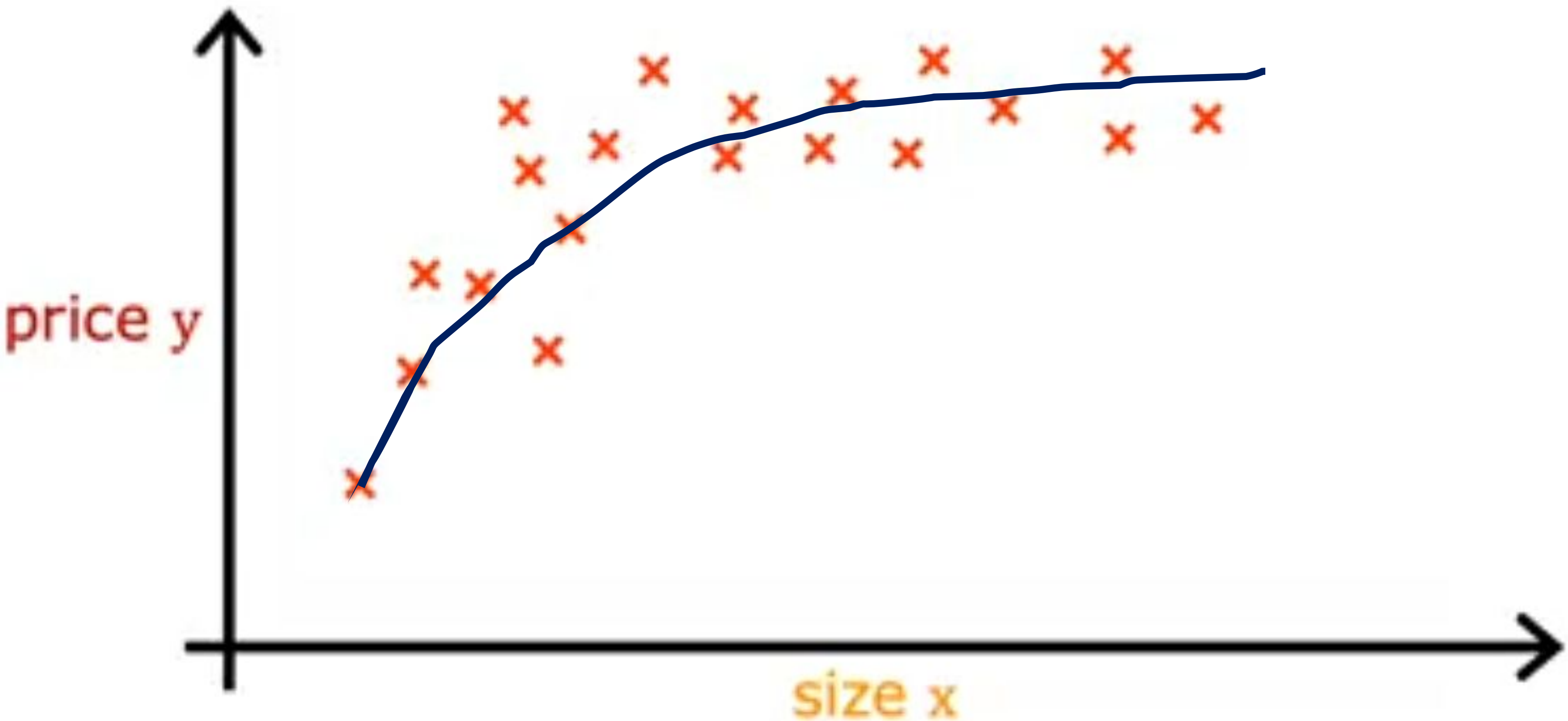
Polynomial Regression

$$f_{(w_1,w_2,w_3,b)}(x) = w_1x + w_2x^2 + w_3x^3 + b$$



Polynomial Regression

$$f_{(w_1,w_2,b)}(x) = w_1 x + w_2\sqrt{x} + \mathrm{b}$$



Problems in Regression Analysis

- https://towardsdatascience.com/five-obstacles-faced-in-linear-regression-80fb5c599fbc