Supervised Learning

Algorithms learn

X ⟶ Y

maps ? input output label

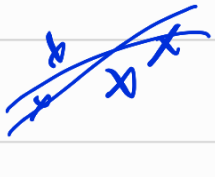
"learn from being given "right answers".

Input	Output
email	⟶ spam ?
audio	⟶ text transcript ?
ad, user info	⟶ click ?

learn from x, y pairs ⟶ then take new

Regression ⟶ predicts a number from many possible outputs (infinite)

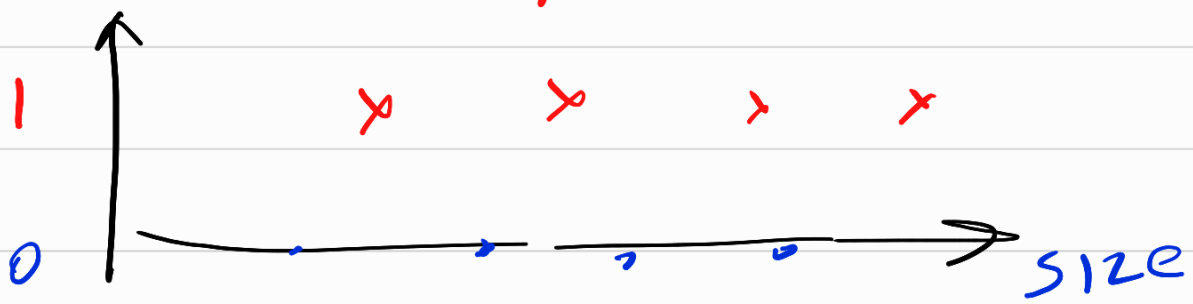
price



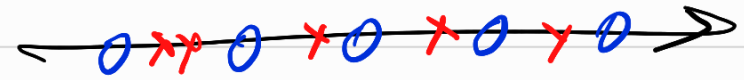
eg:-

Classify : Breast cancer Detection

malignant or benign



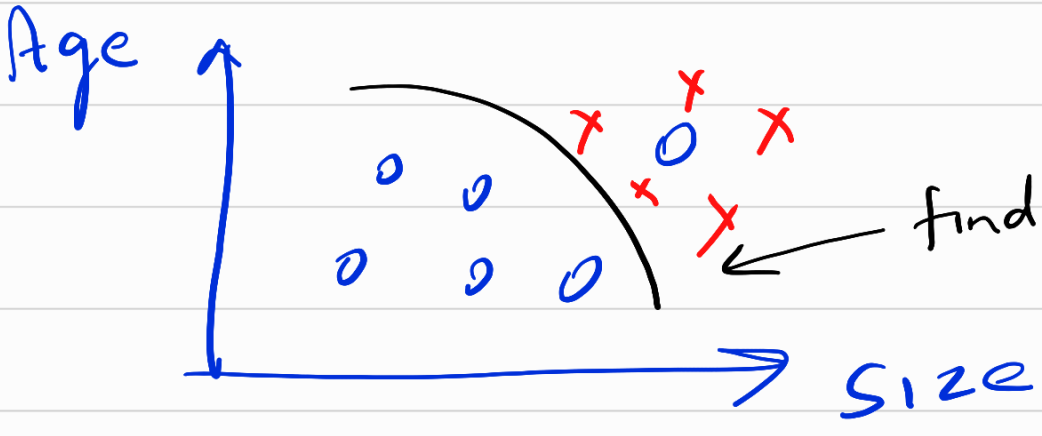
only two possible outputs



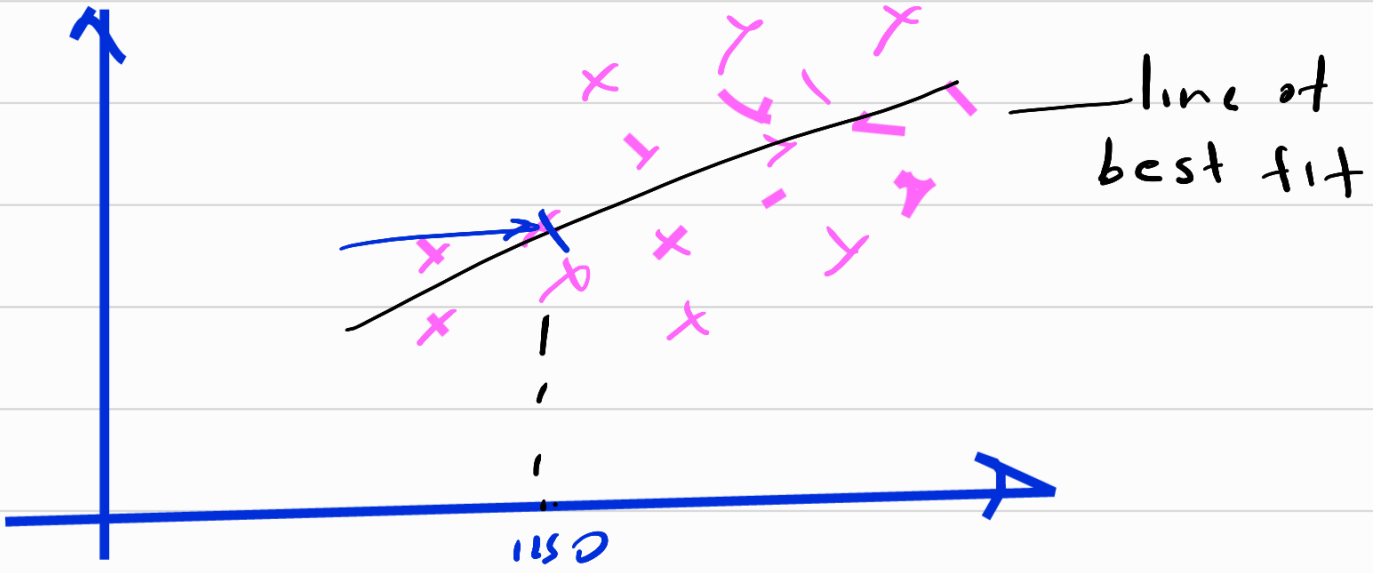
or △ △ — 3 types, ...

Classification Algorithms ⟶ predict categories (finite set of outputs)

can used to 2 inputs too, or more



Linear Regression



Regression model predicts numbers
infinite

Classification model → categories
(small no. of outputs)
— discrete finite

training set → data used to train model

x = input variable, feature

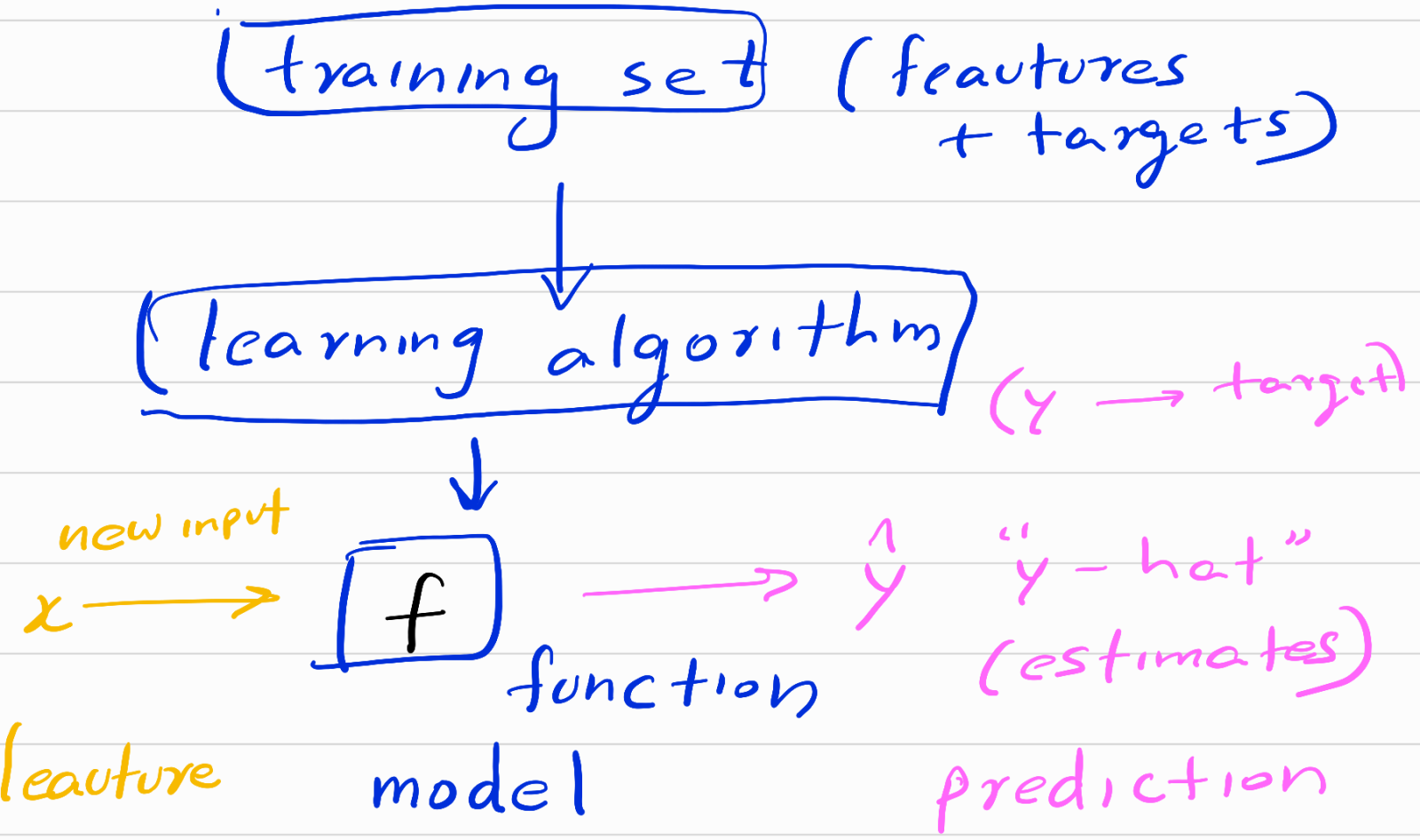
y = output variable, target

m = no. of training examples (rows)

(x, y) = single training example

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

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

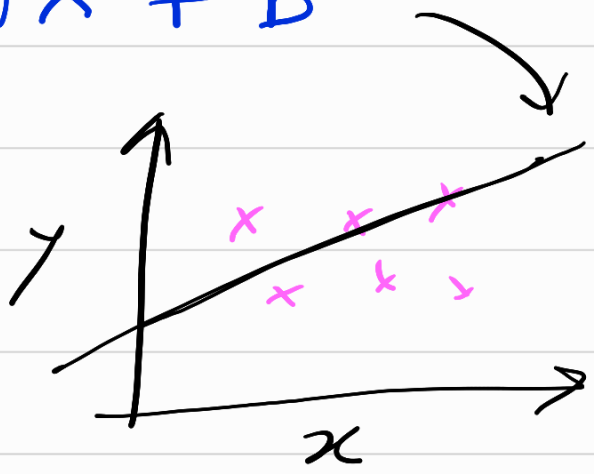


eg:-

size → f → price (estimated y)

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

$f(x)$



Univariate linear regression
(one variable)

$$\frac{y - 300}{x - 1} = \frac{500 - 300}{2 - 1}$$

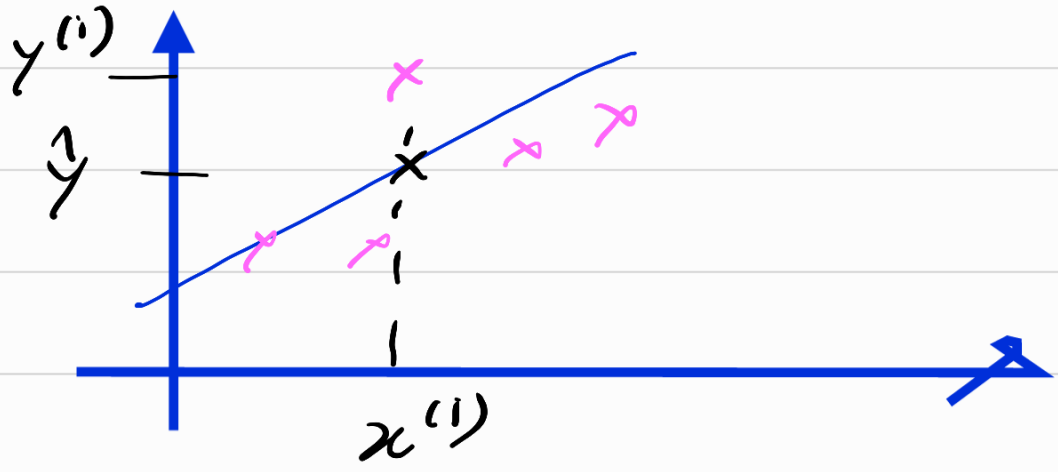
$$y = 200(x - 1) + 300$$

$$= 200x + 100$$

Cost Function:

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

w, b : parameters / coefficients / weights



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

error for all training examples

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

$2m$ (make neater)

$\frac{1}{2m}$ ← average for one

$\hat{y}^{(i)} = f(x^{(i)})$

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

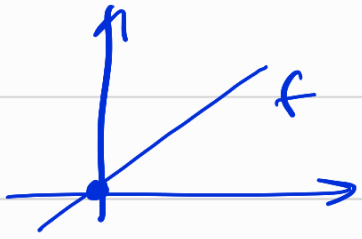
$\hat{y}^{(i)} - y^{(i)}$: error

Squared error cost function

Simplified only for W no b

goal: minimize $J(w, b)$

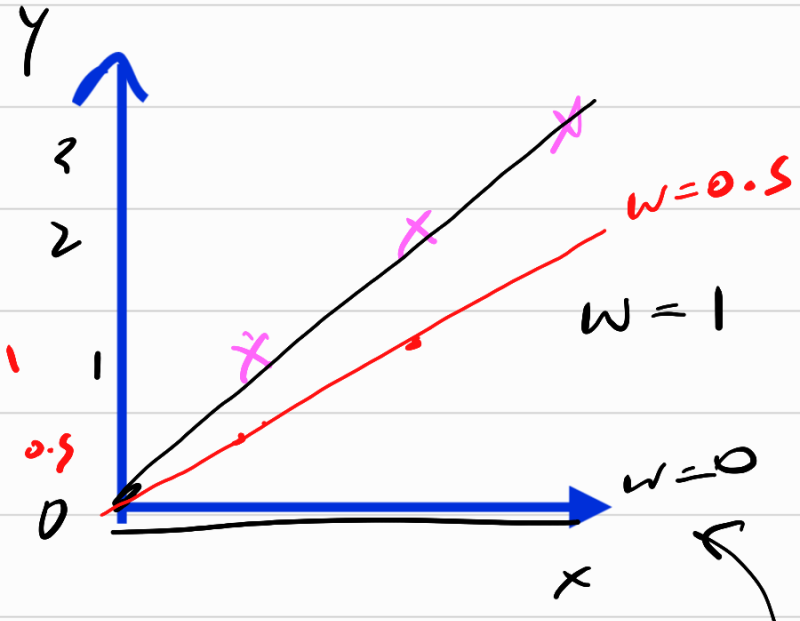
$f_w(x) = wx \quad b = \emptyset$



$f_w(x)$

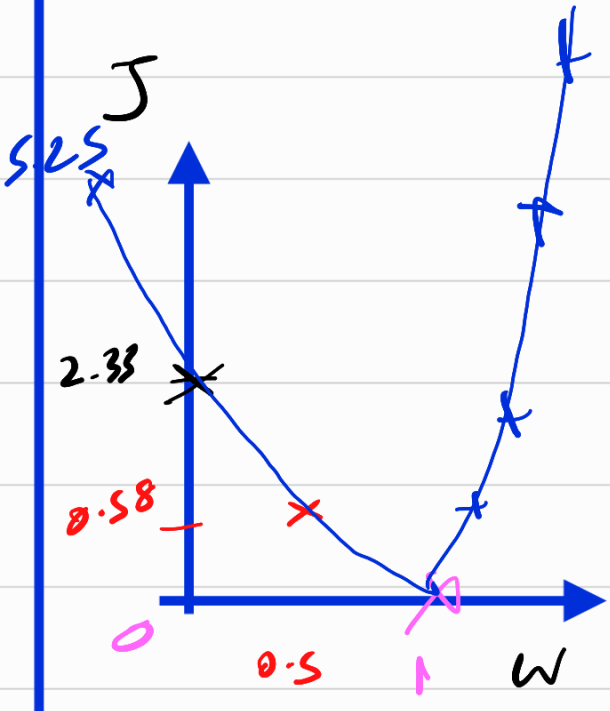
(for fixed w, fun of x)

input



$J(w)$

(func of w)



$$\frac{1}{2m} \sum_{i=1}^{m} (wx^{(i)} - y^{(i)})^2$$

$0 + 0 + 0$

$w = 0.5$

$$\frac{1}{2(3)}\left\{0.5^2 + 1^2 + 1.5\right)^2 = \frac{3.5}{6} \approx 0.58$$

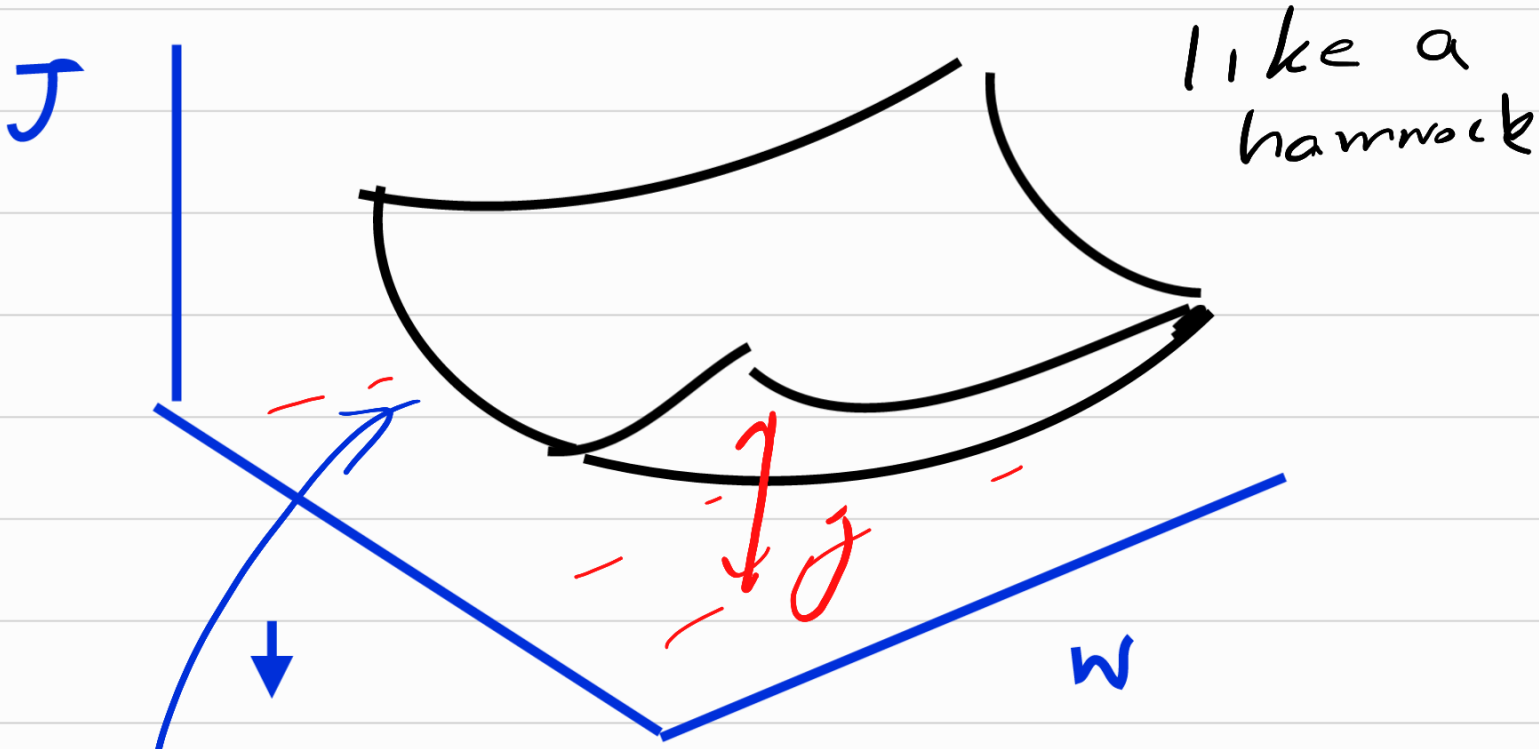
$w=0$ $\frac{1}{2(3)}(1^2+2^2+3^2) = \frac{1}{2(3)} \times 14 = 2.33$

$w=-0.5 \rightarrow 5.25$

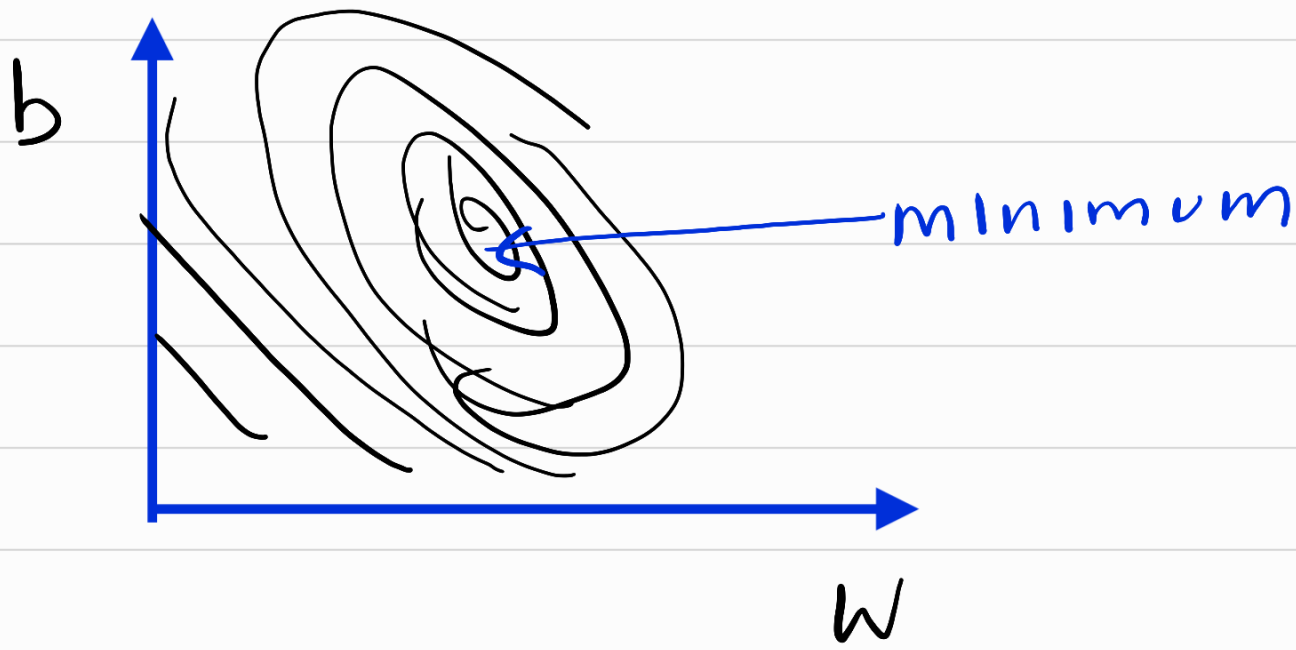
Choose w to minimize $J(w)$

$w = 1$

With both w & b



get horizontal sizes (same heights) and make a counter plot



Gradient Descent

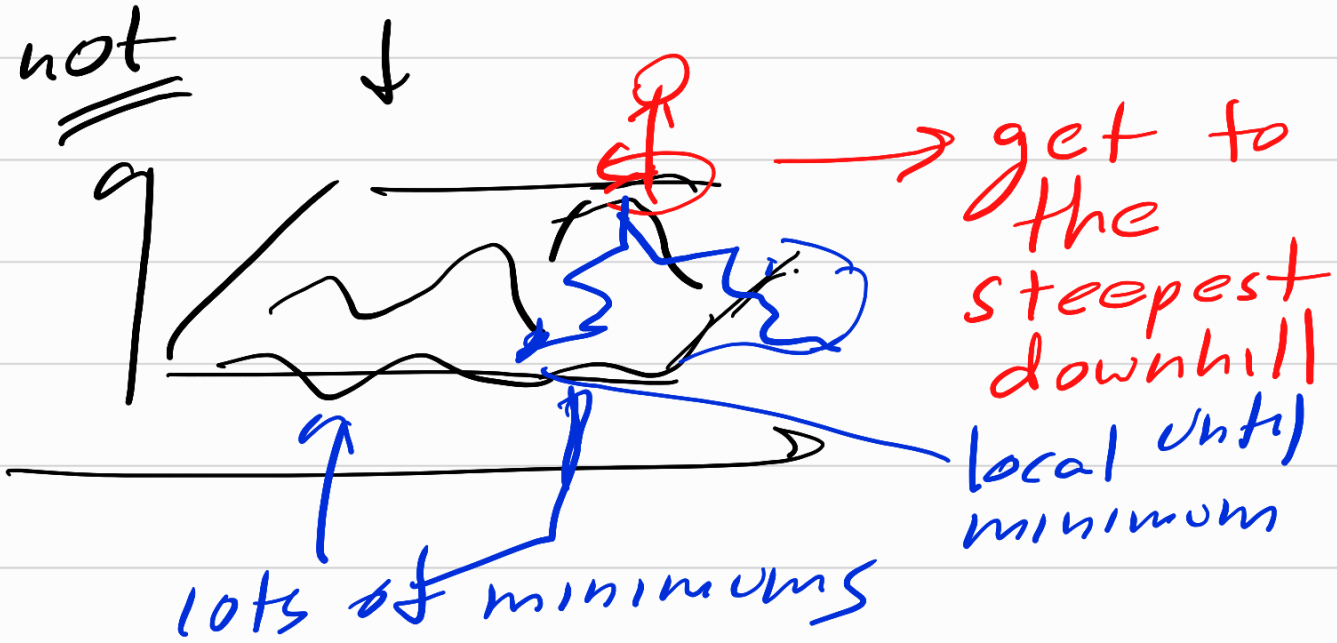
without manually minimizing J
find smallest J

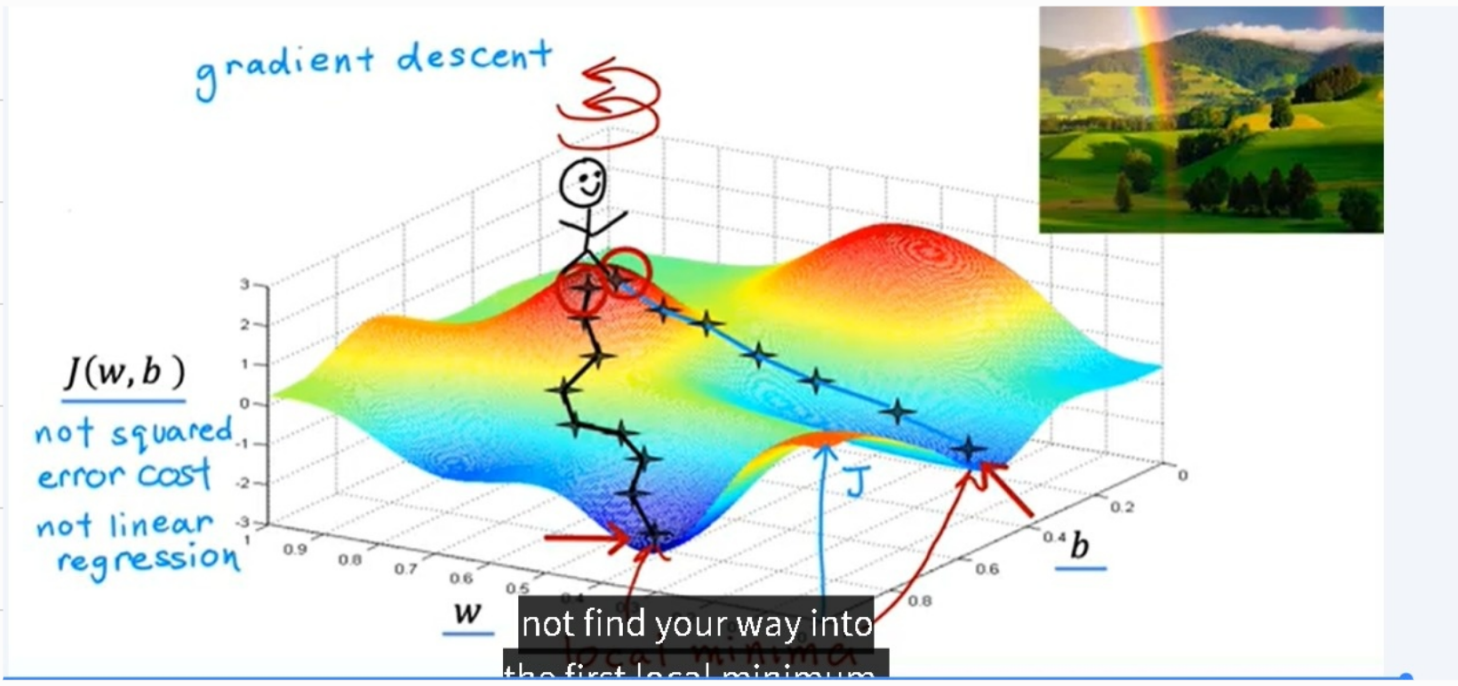
Start with some w, b ($w=0$, $b=0$)

keep changing w, b to reduce J

Find until we settle at or near a minimum.

Squared error ⟶ bow/hanrock





Gradient descent algorithm

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

↑ assertion

derivative

Math

$a = c$

$a = a + 1 - X$

$a == c$

α - learning rate (size of the step)

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

* Repeat until convergence

- Simultaneously update w & b

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

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

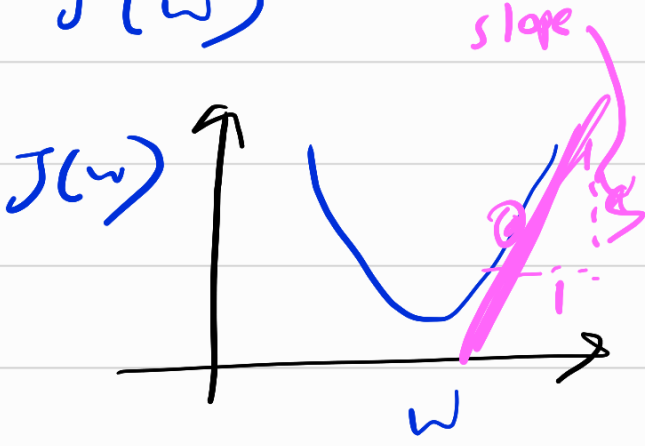
use same w before

$$w = tmp\_w$$
$$b = tmp\_b$$

$\frac{d}{dw}$

for simplicity

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

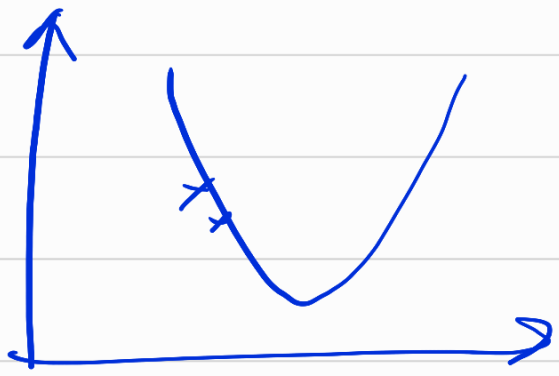


$\alpha \geq 0$ always

if $\frac{d}{dw} > 0$ ← $\therefore$ w decreases ← moves left

$\frac{d}{dw} < 0$ → w increase → moves right

α



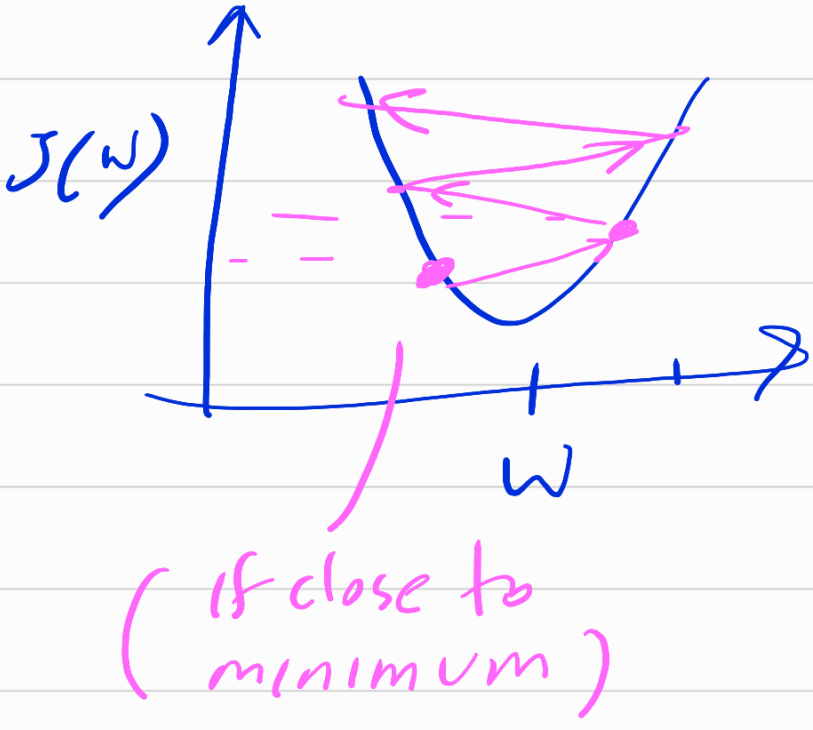
if α too small

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

(reduce is small)

tiny steps to get to minimum

if α is too large

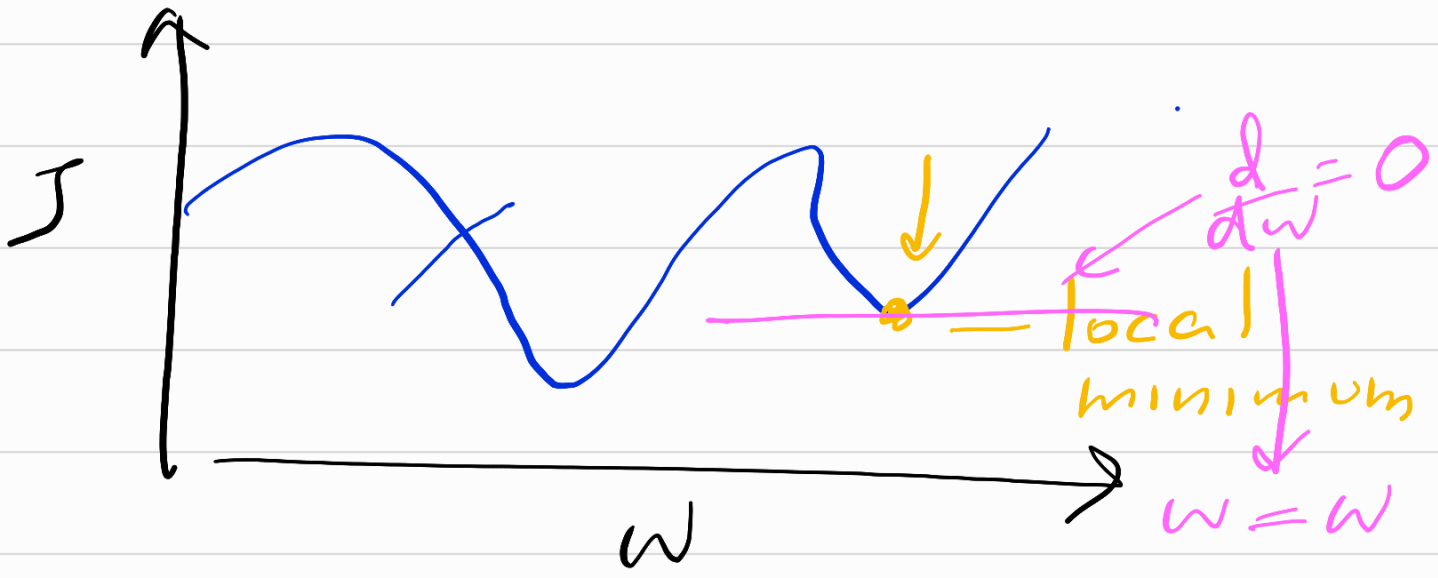


(if close to minimum)

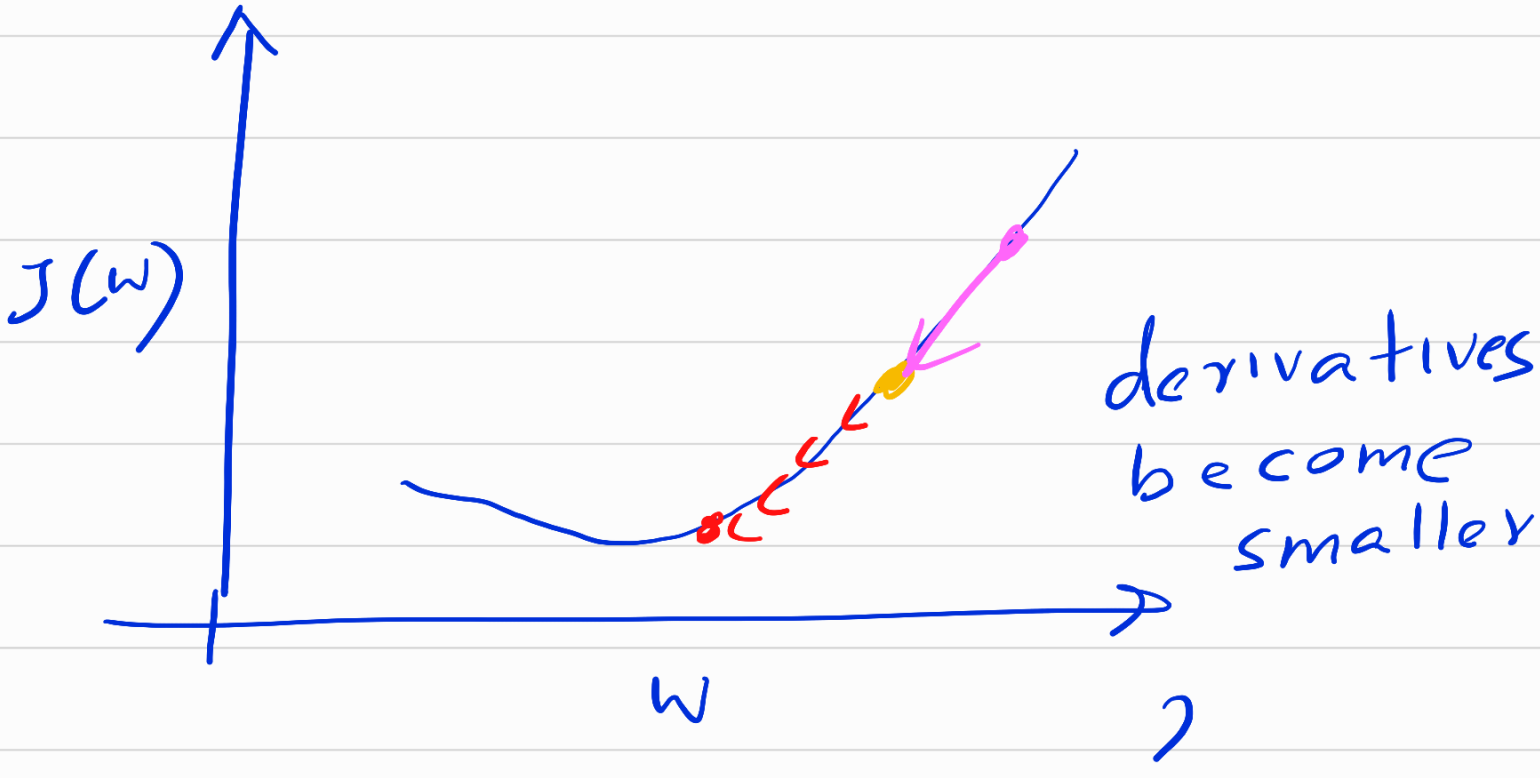
Overshoot, (fail to converge)

never reach minimum

if already in minimum;



(not sq. error → many minimums)



derivatives become smaller

↓

updates become smaller

G.D. Algorithm

(Optional)

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

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

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

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

You can plug them into the gradient descent algorithm

with b

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

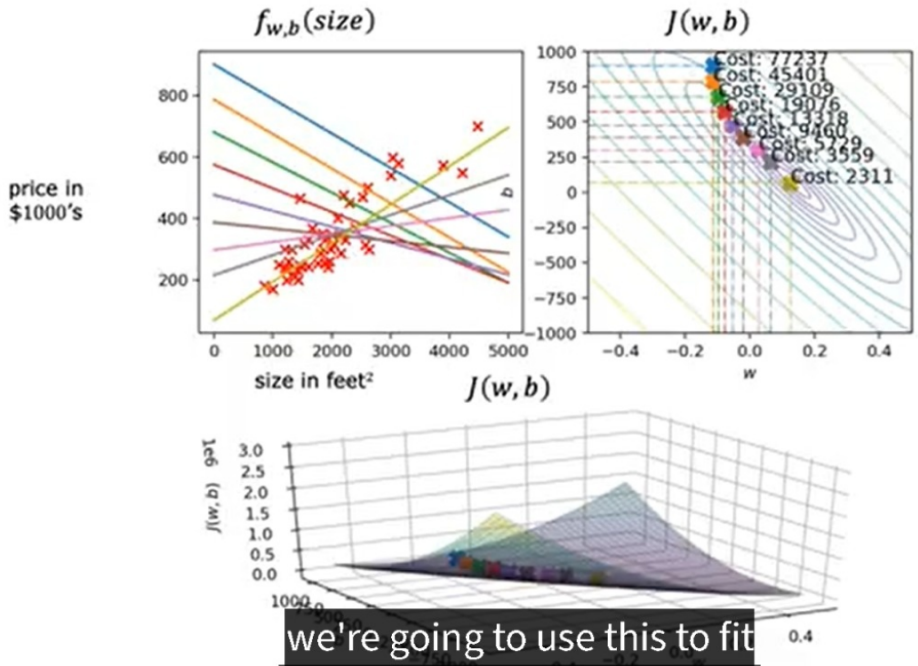
$$b = b - \alpha \frac{1}{m}\sum_{i=1}^{m}\left(f_{w,b}(x^i) - y^i\right)$$

update simultaneously until convergence.

Convex function (bowl)

→ single global minimum

eg:- Squared error cost



we're going to use this to fit a model to the holding data.

Batch gradient descent

↳ each steps uses all training examples

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

Multiple features (variables)

x_j = j^{th} feature

n = number of features

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

vector

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

$$X_3^{(2)} = 2$$

previously model,

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

Now,

$$f_{w,b}(x) = w_1x_1 + w_2x_2 + w_3x_3 + \cdots + b$$

eg:-

$0.1 x_1 + 4 x_2 + 10 x_3 - 2 x_4 \ldots + 80$

size | bedrooms | years | base price

$\vec{w} = [w_1 \; w_2 \; w_3 \; \ldots \; w_n]$

parameters of model

b is a number

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

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

dot product

$= w_1 x_1 + w_2 x_2 + \ldots + w_n x_n + b$

multiple linear regression

Vectorization

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

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

python

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

×

↓ Vectorization

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

Gradient Descent

w = np.array([, . .])
d = np.array([0., ...])

$$w_j = w_j - \underbrace{0.1}_{\alpha} d_j$$

without vector ;

```
for j in range(16);
w[j] = w[j] - 0.1 * d[j]
```

with vector ;

$$\vec{w} = \vec{w} - 0.1\,\vec{d}$$

	Previous notation	Vector notation
Parameters	$w_1, \cdots, w_n$ b	$\vec{w} = [w_1 \quad \cdots \quad w_n]$ ← vector of length n b still a number
Model	$f_{\vec{w},b}(\vec{x}) = w_1 x_1 + \cdots + w_n x_n + b$	$f_{\vec{w},b}(\vec{x}) = \vec{w} \cdot \vec{x} + b$ (dot product)
Cost function	$J(w_1, \cdots, w_n, b)$	$J(\vec{w}, b)$

Gradient descent

repeat {

$w_j = w_j - \alpha \frac{\partial}{\partial w_j} J(w_1, \cdots, w_n, b)$

$b = b - \alpha \frac{\partial}{\partial b} J(w_1, \cdots, w_n, b)$

}

repeat {

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

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

}

Gradient descent

One feature

repeat {

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

$\frac{\partial}{\partial w} J(w, b)$

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

simultaneously update w, b

}

n features ($n \geq 2$)

repeat {

$j = 1$: $w_1 = w_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} (f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)}) x_1^{(i)}$

$\frac{\partial}{\partial w_1} J(\vec{w}, b)$

⋮

$j = n$: $w_n = w_n - \alpha \frac{1}{m} \sum_{i=1}^{m} (f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)}) x_n^{(i)}$

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

simultaneously update w_j (for $j = 1, \cdots, n$) and b

}

Alternative to gradient descent

→ Normal equation

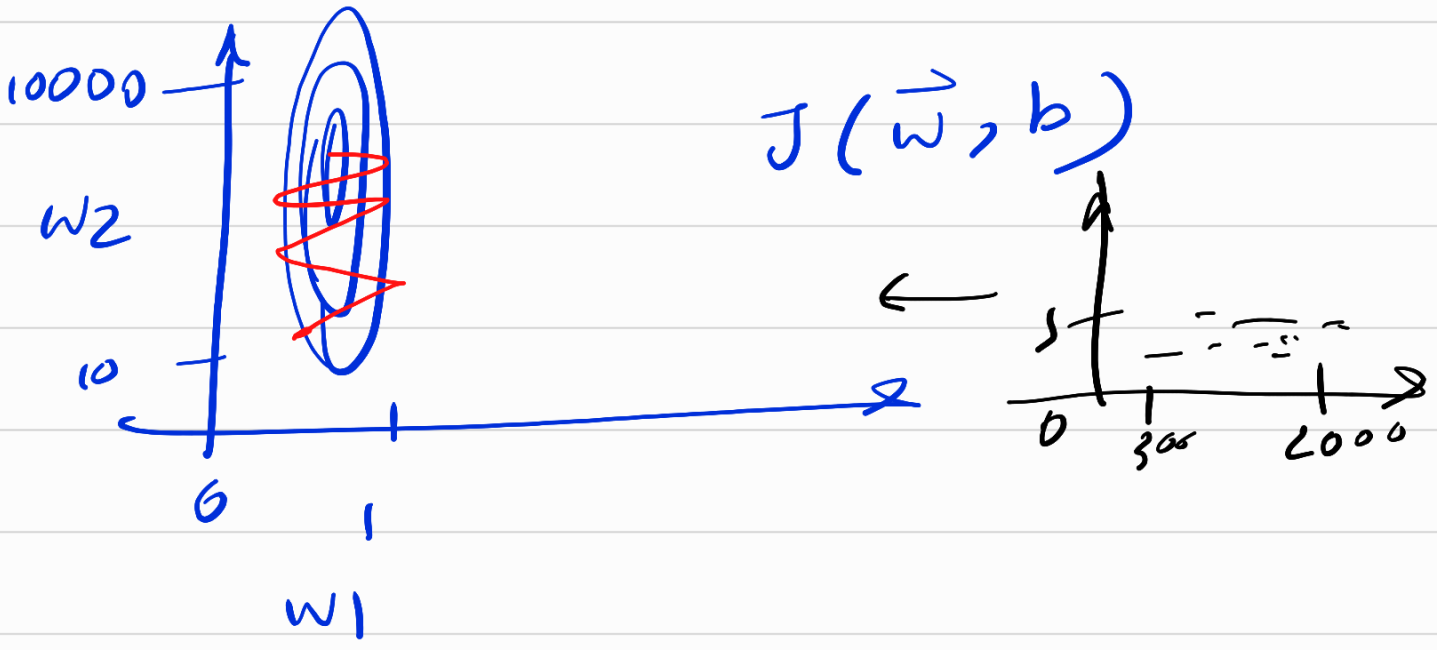
* only for linear regression
  solve for w, b without iterations

Doesn't generalize

Feature Scaling

feature large relatively small

w: small large

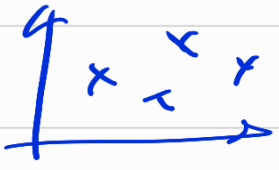


$300 < x_1 < 2000$

$0 < x_2 < 5$

$\text{scaled } x_1 = \frac{x_1}{2000}$

$\text{scaled } x_2 = \frac{x_2}{5}$



or Mean Normalization

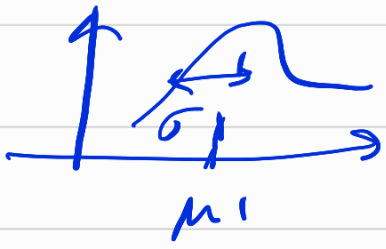
$x_1 = \frac{x_1 - \mu_1}{2000 - 300}$

$-0.18 \leq x_1 \leq 0.82$

$x_2 = \frac{x_2 - \mu_2}{5 - 0}$

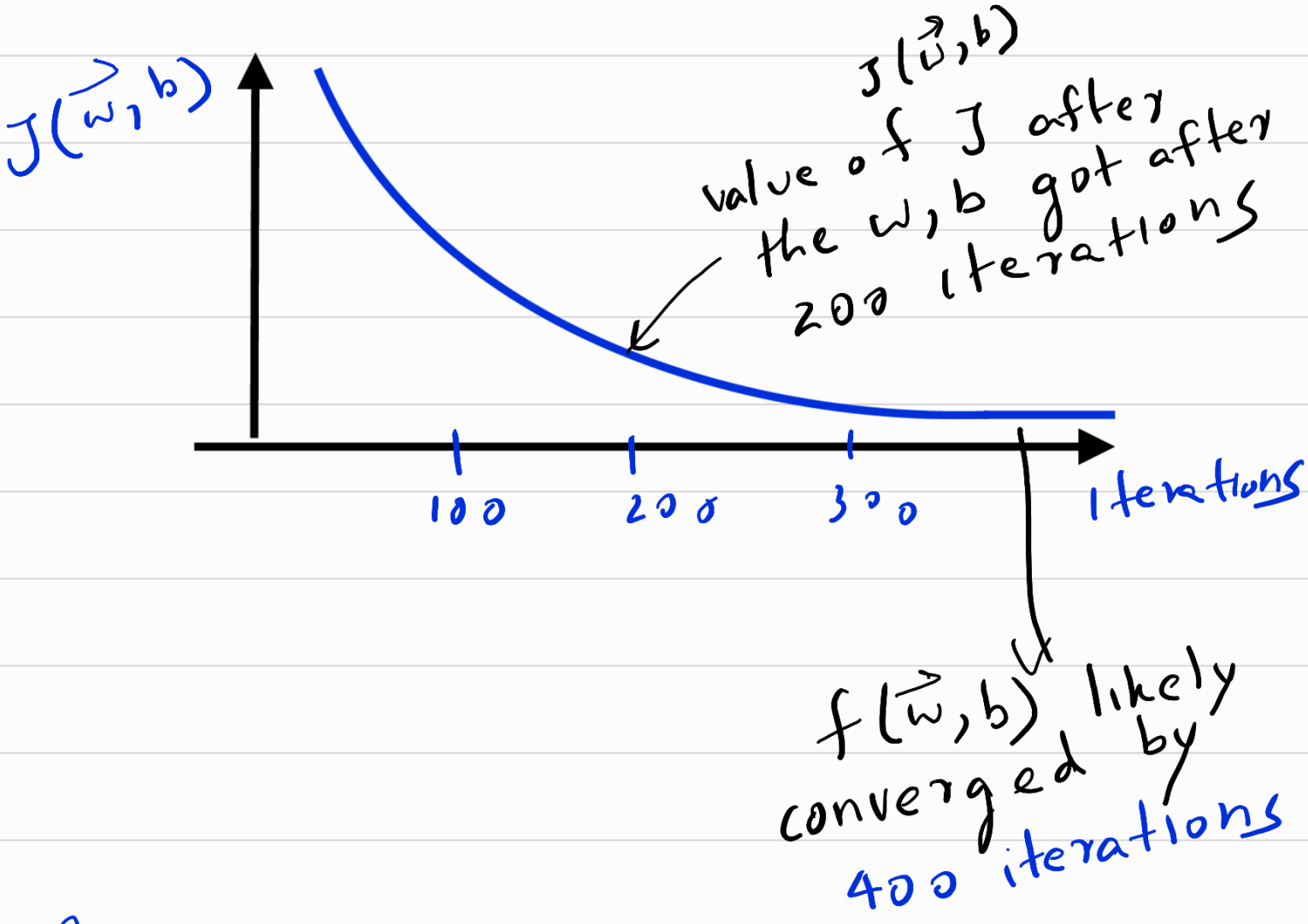
$-0.46 \leq x_2 \leq 0.54$

Z-score normalization



$$x_1 = \frac{x_1 - \mu_1}{\sigma_1} \qquad x_2 = \frac{x_2 - \mu_2}{\sigma_2}$$

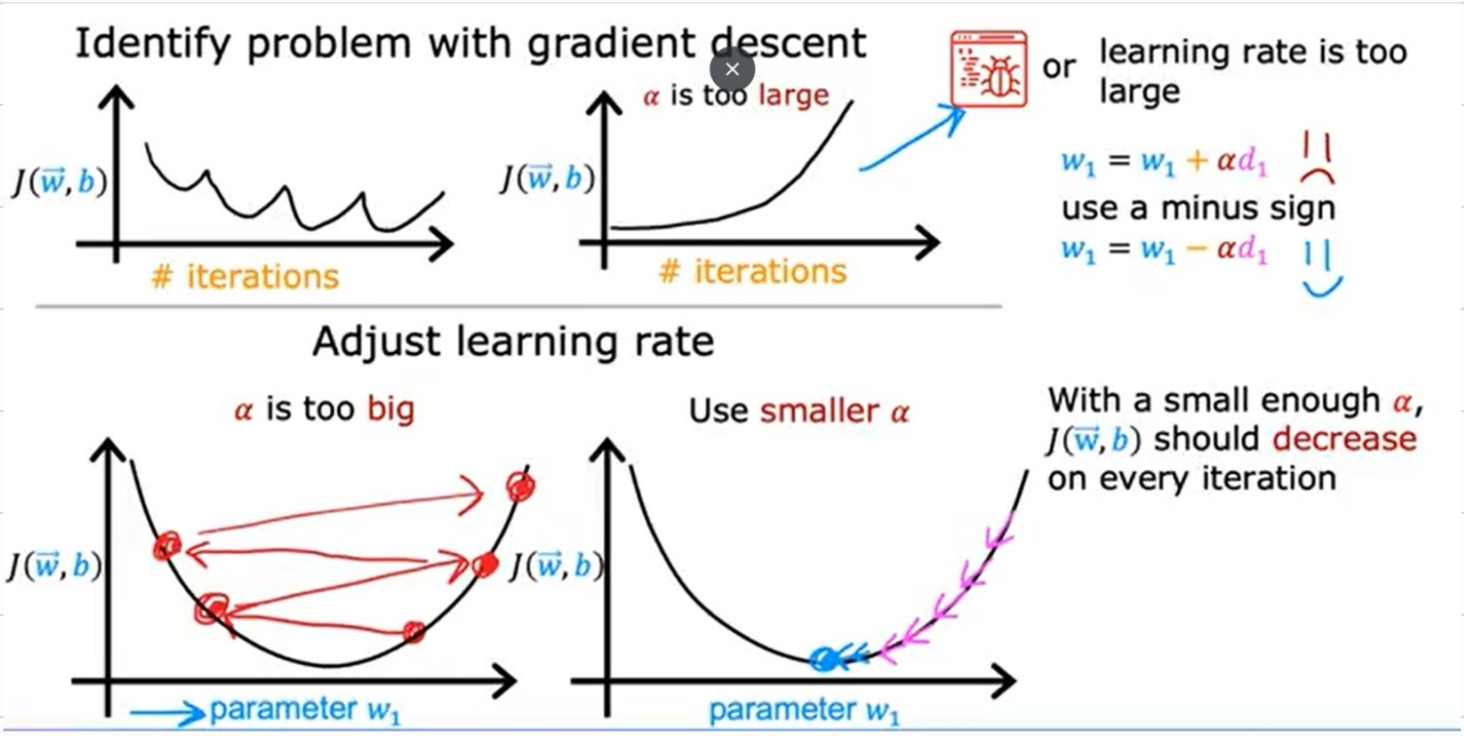
aim about $-1 < x_j < 1$



Automatic convergence Test

if $J(\vec{w},b)$ decreases by $\leq \varepsilon$ in one iteration declare convergence

Choosing Learning Rate



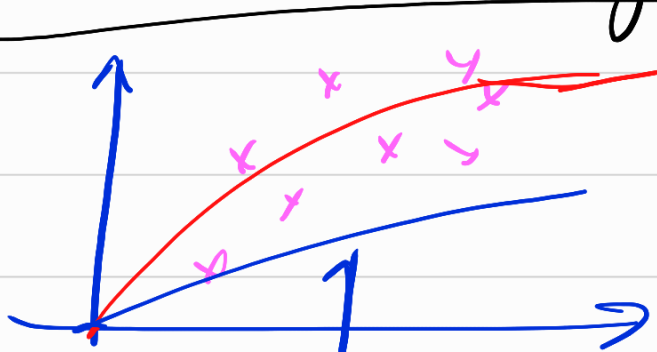
Feature Engineering

transform, combine features

h, w, l

→ * volume

Polynomial Regression



← $w_1 x + w_2 x^2 + w_3 x^3 + b$

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