Supervised learning

Regression
↳ Output is infinite

Classification
↳ Output is discrete

Terminology:

x → input/feature
y → output/target
m → Total training examples
(x^i, y^i) = i^{th} training example
↳ specific row

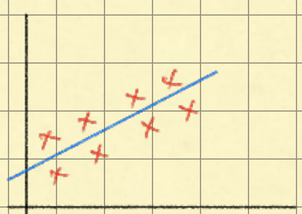
Training Set
↓
Learning algorithm
↓

$$\underset{\text{new input}}{x} \longrightarrow \underset{\text{model}}{f} \longrightarrow \hat{y}$$

(estimated y) ← target

How to represent f?

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



linear regression with $\overbrace{\text{one variable}}^{\text{one } x \text{ only}}$ / Univariate

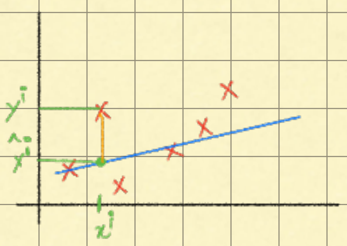
np.array([x, y]) ⇒ numpy array

f_wb = np.zeros(0) → gives 1D array

Cost function:

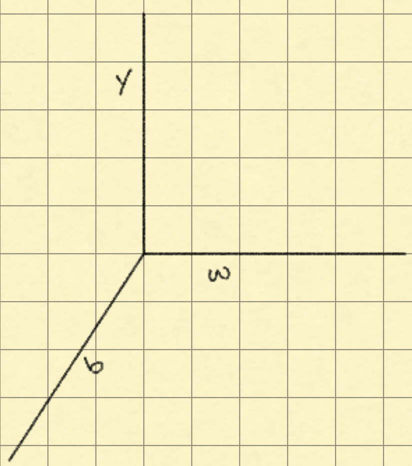
↳ Tells how well your model is doing

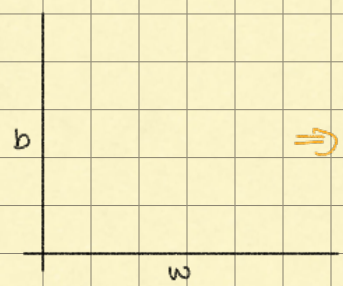
w, b : parameters/coefficient



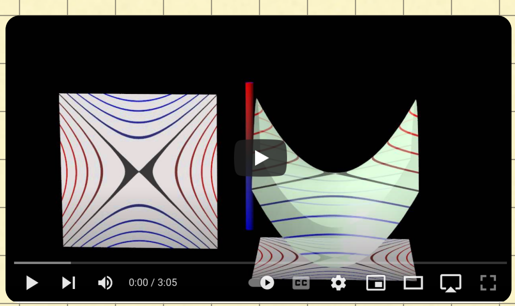
$$J(w,b) = \frac{1}{2m}\sum_{i=1}^{m}\left(\hat{y}^{(i)} - y^{(i)}\right)^2$$

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





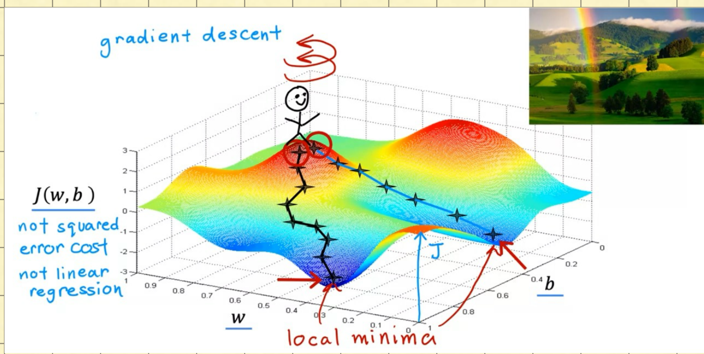
⇒ contours
visual 3D in 2D



we choose the f(w,b) which corresponds to the lowest J(w,b) in the graph.

Gradient descent:

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



mathematically,

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

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

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

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

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

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

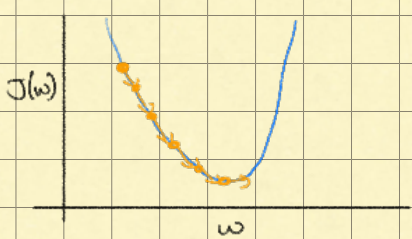
h

α = learning rate (0 – 1) ⟶ Step length

change w & b till w & b don't change by muc after each computation.

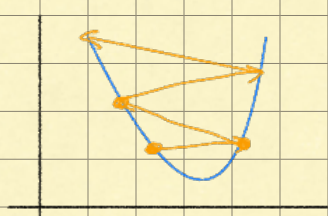
Learning Rate:

if α is too small,



↳ may be slow

if α is too large,



↳ may overshoot