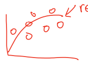
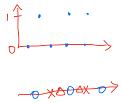
Supervised learning
- given input and its corresponding output, after learning, they can take brand new Input to predict output

Input (x)	output (Y)
English	→ spanish

regression - predict a $\boxed{\text{number}}$ from infinitely many possible outputs

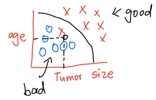


classification - predict on small outputs / it predicts $\boxed{\text{categories}}$

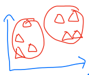


Δ = good 2
x = good
y = bad

two or more Input



unsupervised learning
- Find something interesting from unlabeled data
- Data comes with input x, but not output y & algorithm have to find structure in the data

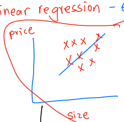


clustering - takes data without labels and automatically group them into clusters

Anomaly detection - Find unusual data points

Dimensionality reduction - compress data using fewer numbers

Linear regression - example of regression model



(x) Size	(y) price
(1) 2100	300
400	600
⋮	⋮
(47) 3210	485

Training set : Data used to train the model

univariate linear regression or linear regression with $\boxed{\text{one}}$ variable → single input

uni → one

Notation
- x = input variable
- y = output/target variable
- m = number of training examples
- (x, y) = single training examples
- $(x^{(i)}, y^{(i)})$ = ith training example

supervised model

$\boxed{\text{training set}}$
↓
$\boxed{\text{learning algorithm}}$
↓
$\boxed{x} \rightarrow \boxed{f} \rightarrow \boxed{\hat{y}} \rightarrow$ estimated y

y-hat

input / feature   model (math formula)

cost function

Squared error cost function / difference

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

m = number of training set

goal: minimize J by changing w, b, the smaller J, the better

Imagine a model

$$f_{w,b}(x) = wx + b \text{ or } f(x) = wx + b$$

w, b = parameters (variables you can change to improve algorithms)
coefficients
weights

Example:

$f_w(x) = wx$

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

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

Gradient descent algorithm to find the smallest value for J / $\min_{w,b} J(w,b)$
- find the value of w, b that best fit the question or called
- $w = w - a\boxed{\frac{d}{dx}(J(w,b))} \rightarrow \frac{1}{m}\sum_{i=1}^{m}(f_{w,b}(x^{(i)}) - y^{(i)})x^{(i)}$

  ↳ learning rate
- $b = b - a\boxed{\frac{d}{dx}J(w,b)} \rightarrow \frac{1}{m}\sum_{i=1}^{m}(f_{w,b}(x^{(i)}) - y^{(i)})$

Just derivative it

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

- If a is too small, Gradient descent may be slow
- If a is too large, Gradient descent may
  - overshoot, never reach minimum
  - fail to converge, diverge

- You have to update w, b simultaneously
- repeat the w, b function until convergence

correct	incorrect
$temp\_w = w - a\frac{d}{dx}J(w,b)$	$temp\_w = w - a\frac{d}{dx}J(w,b)$
$temp\_b = b - a\frac{d}{dx}J(w,b)$	$w = temp\_w$
$w = tmp\_w$	$temp\_b = b - a\frac{d}{dx}J(w,b)$
$b = tmp\_b$	$b = temp\_b$

batch gradient descent

- Each step of gradient descent uses all the training examples