→ Supervised Machine Learning : Regression & Classification

Supervised vs. unsupervised learning

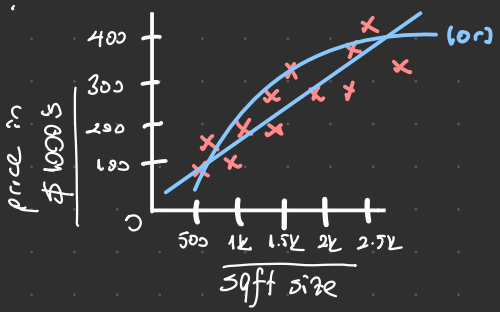
Quick Quest! If Arthur Samuel's checkers-playing program had been allowed to play only 10 games against itself, how would this affected its performance compared to when it was allowed to play over 10k games?
→ would have made it worse

⇒ Supervised Learning: learns from data labeled with "right answers"

e.g.

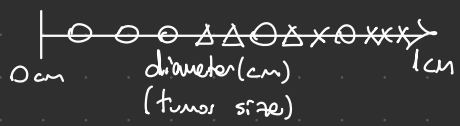
email	→	spam? (0/1)	⇒ spam filtering
audio	→	text transcripts	⇒ speech recognition
ad, user info	→	click? (0/1)	⇒ online advertising
image, radar info	→	pos. of other cars	⇒ self-driving car

→ Regression: Predict a number infinitely many possible outputs

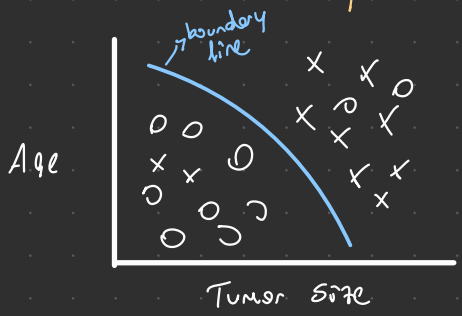


→ Classification: Predict categories, small number of possible outputs

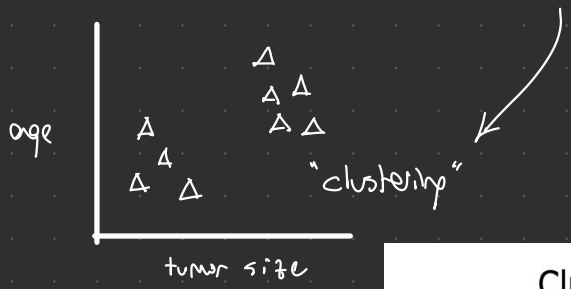
e.g. O benign
X malignant 1
Δ malignant 2

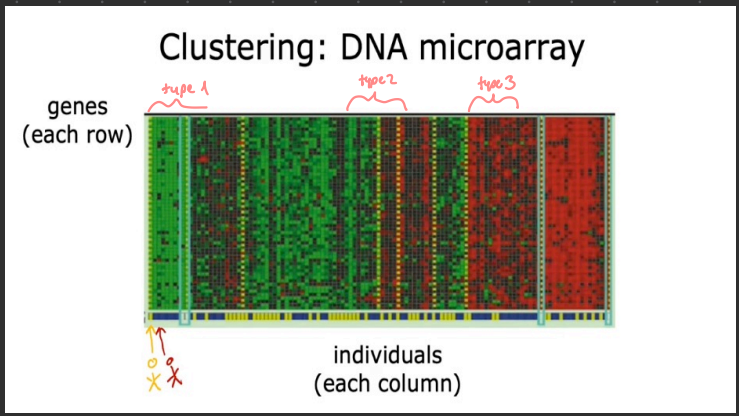


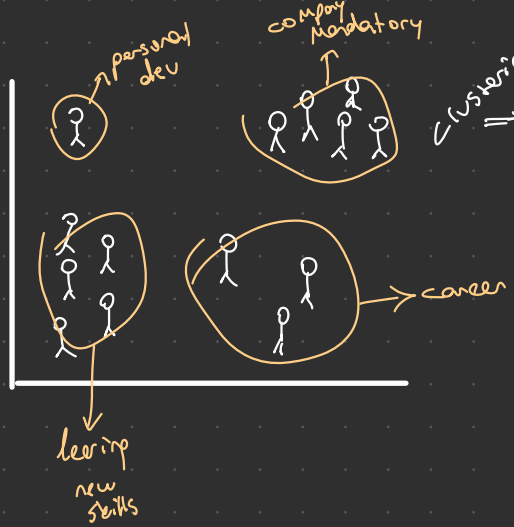
Two or more inputs



⇒ Unsupervised learning: find something (pattern etc.) interesting in unlabeled data







Clustering => Grouping Customers

→ Data only comes with input X, but not output labels y
<> Algorithm has to find structure in the data

Clustering
Group similar data points together

Anomaly Detection
Find unusual data points

Dimensionality Reduction
Compress data using fewer numbers

Question

Of the following examples, which would you address using an unsupervised learning algorithm?

- ☐ Given email labeled as spam/not spam, learn a spam filter.
- ☑ Given a set of news articles found on the web, group them into sets of articles about the same story.
- ☑ Given a database of customer data, automatically discover market segments and group customers into different market segments.
- ☐ Given a dataset of patients diagnosed as either having diabetes or not, learn to classify new patients as having diabetes or not

Regression Model → predicts numbers

=> Linear Regression!

→ Terminology!

Training Set: Data used to train model

	x: sqft size	y: price	
(1)	2104	400	
(2)	1416	232	
(3)	1534	315	M = 47
⋮			
(47)	3210	870	

NOTATION:

x = "input" variable
"feature"

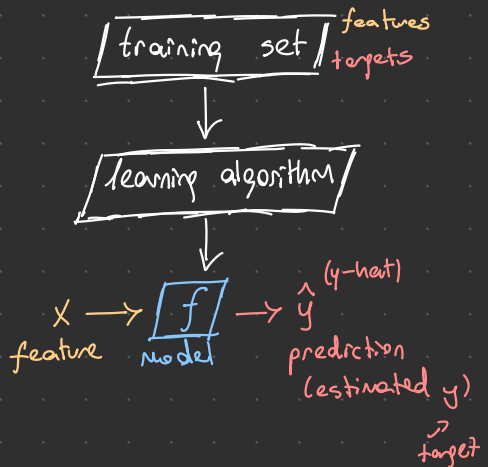
y = "output" variable
"target" variable

m = number of training examples

(x, y) = single training example

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

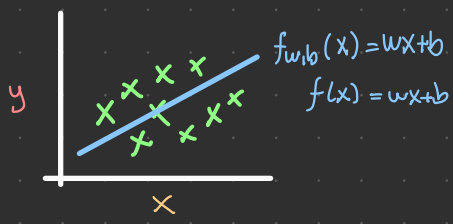
e.g. $(x^{(1)}, y^{(1)}) = (2104, 400)$



How to represent f ?

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

$$f(x)$$



linear regression with one variable

Univariate linear regression

one variable

=> Cost function (squarred error cost function)

$$J(w,b) = \frac{1}{2M} \sum_{i=1}^{M} \left(\underbrace{\hat{y}^{(i)} - y^{(i)}}_{\text{error}} \right)^2$$

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

→ cost function intuition

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

parameters: w, b

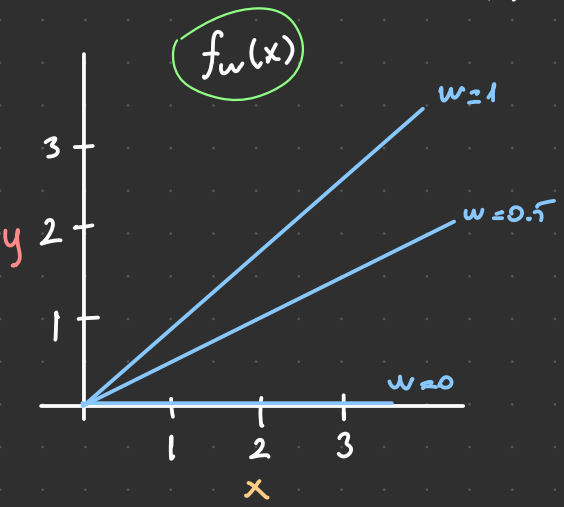
cost function: $\frac{1}{2m} \sum_{i=1}^{m} \left(f_{w,b}(x^{(i)}) - y^{(i)}\right)^2$

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

↳ simplified

$f_w(x) = wx \quad (b = 0)$

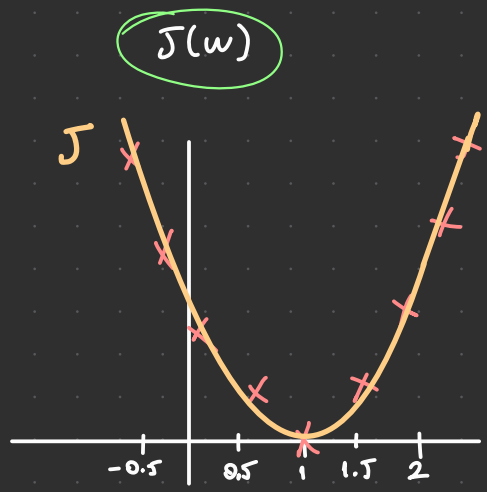
$J(w) = \frac{1}{2m} \sum_{i=1}^{m} \left(f_w(x^{(i)}) - y^{(i)}\right)^2$, $\underset{w}{\text{minimize}}\ J(w)$



$J(0) = \frac{1}{6}(1^2 + 2^2 + 3^2) = \frac{1}{6}[14]$

$J(0.5) = \frac{1}{6}((0.5-1)^2 + (1-2)^2 + (1.5-3)^2)$
$= \frac{1}{6}[3.5]$

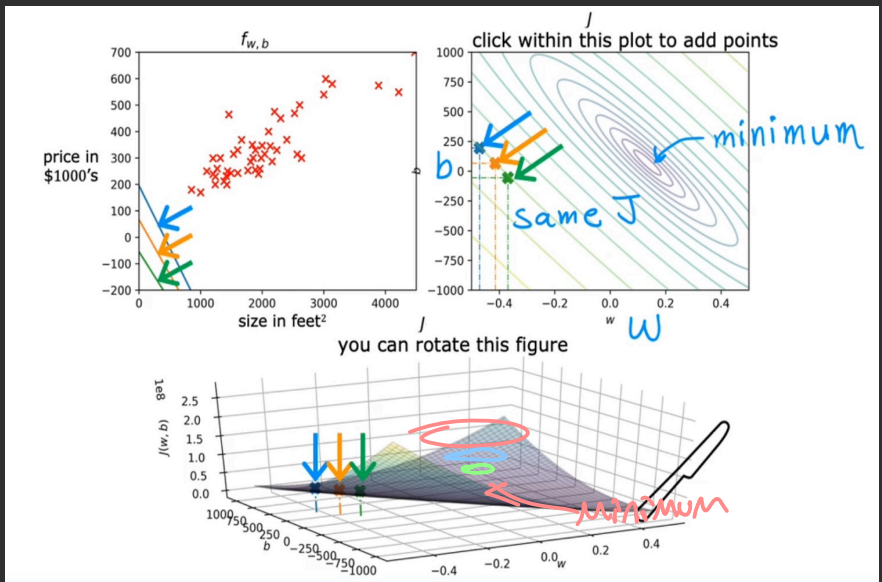
$J(1) = 0$



goal of linear regression: $\underset{w}{\text{minimize}}\ J(w)$

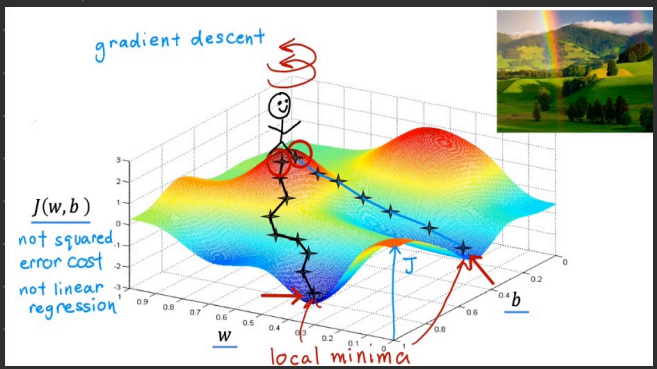
general case: $\underset{w,b}{\text{minimize}}\ J(w,b)$

=> Visualizing cost - function



TRAIN THE MODEL WITH GRADIENT DESCENT

Gradient Descent Algorithm: an optimization algorithm used to minimize a function by iteratively moving towards the steepest descent



using to fit a model to holding data

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

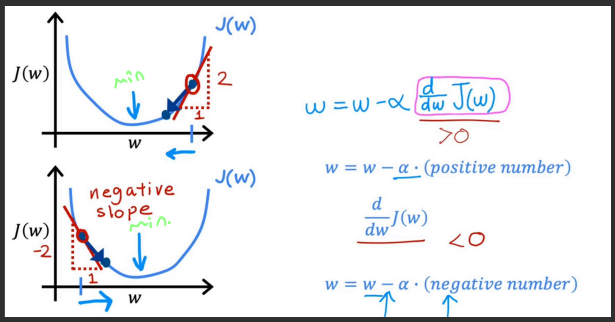
- $w =$: assigning
- α : learning rate (taking step)
- $\frac{\partial}{\partial w} J(w,b)$: (partial) derivative

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

simultaneously update w and b

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

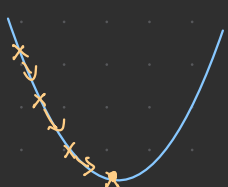
-> Gradient Descent Intuition



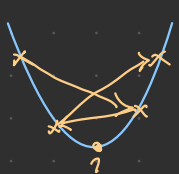
in both example, gradient descent doing something reasonable as it gets you closer to minimum

-> Learning Rate (α)

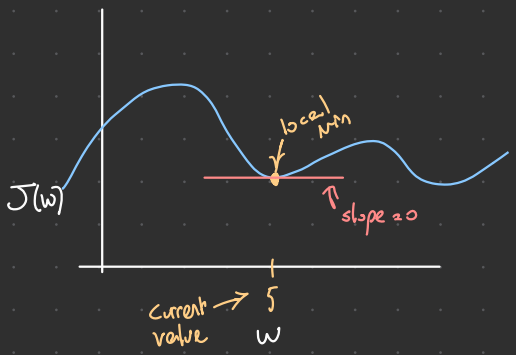
-> if learning rate is too small, gradient descent will work, but it will be slower
(think like taking a small step while walking)



-> if α is too large
- gradient descent may never reach minimum (overshooting)
- fail to converge / diverge



Quick Quest: What happens if you've already reach / at minimum?



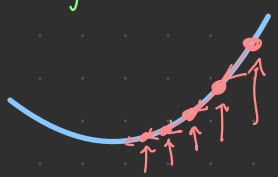
slope = 0 ⇒ derivative = 0

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

$= w = w$

Can reach local minimum with fixed α ?

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



as we get close to local minimum slope gets near zero

so "$\alpha \cdot \frac{\partial}{\partial w} J(w)$" get smaller in each step

so we can reach min. even with fixed α

⇝ Gradient Descent for Linear Regression

linear regression

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

Cost function

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

Gradient Descent Algo.

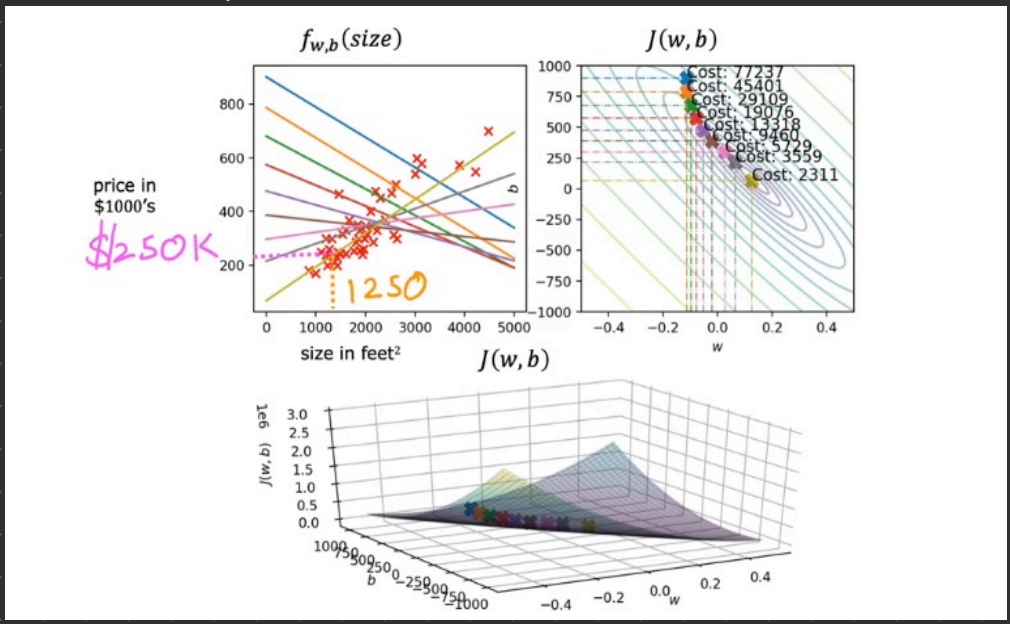
repeat until diverge {

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

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

}

→ Running Gradient Descent



"Batch" Gradient Descent

↓

Each step of gradient descent uses all the training examples

M = 47

prev. example

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