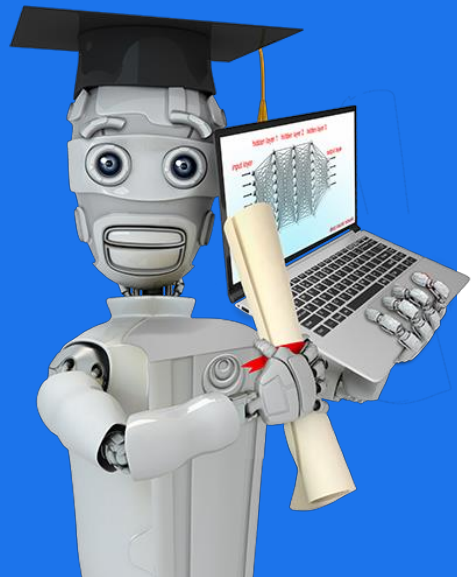
Stanford
ONLINE

DeepLearning.AI



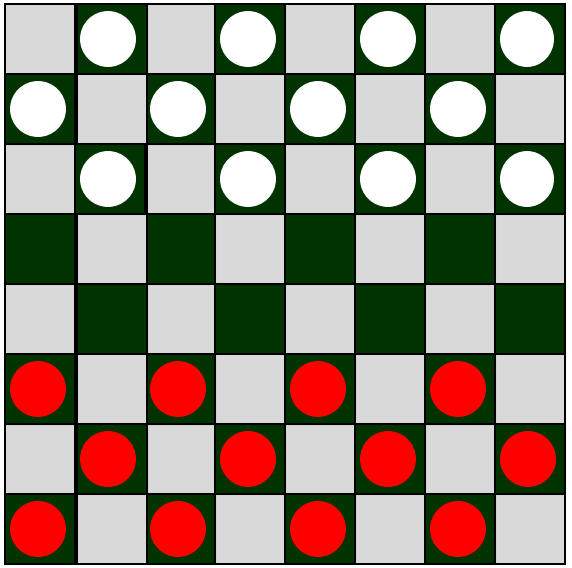
Machine Learning Overview

What is Machine Learning?

Machine learning

"Field of study that gives computers the ability to learn without being explicitly programmed."

Arthur Samuel (1959)



Question

If the checkers program had been allowed to play only ten games (instead of tens of thousands) against itself, a much smaller number of games, how would this have affected its performance?

- Would have made it better
- Would have made it worse

Machine learning algorithms

rapid advancements

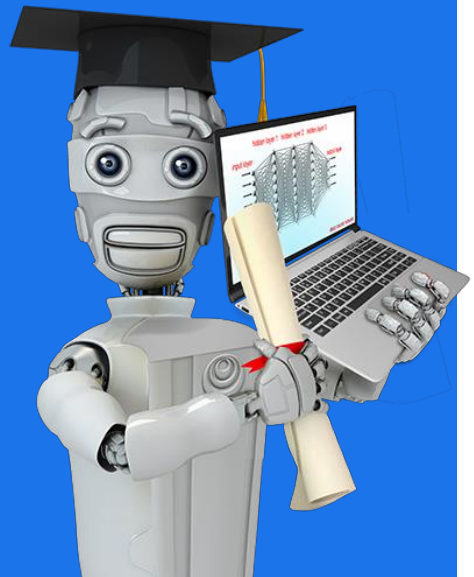
used most in real-world applications

- Supervised learning ← course 1, 2
- Unsupervised learning ← course 3
- Recommender systems
- Reinforcement learning

Practical advice for applying learning algorithms

Stanford
ONLINE

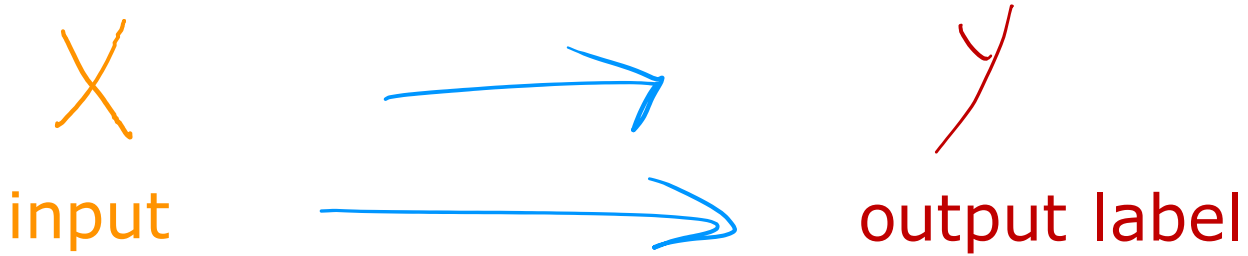
DeepLearning.AI



Machine Learning Overview

Supervised Learning Part 1

Supervised learning

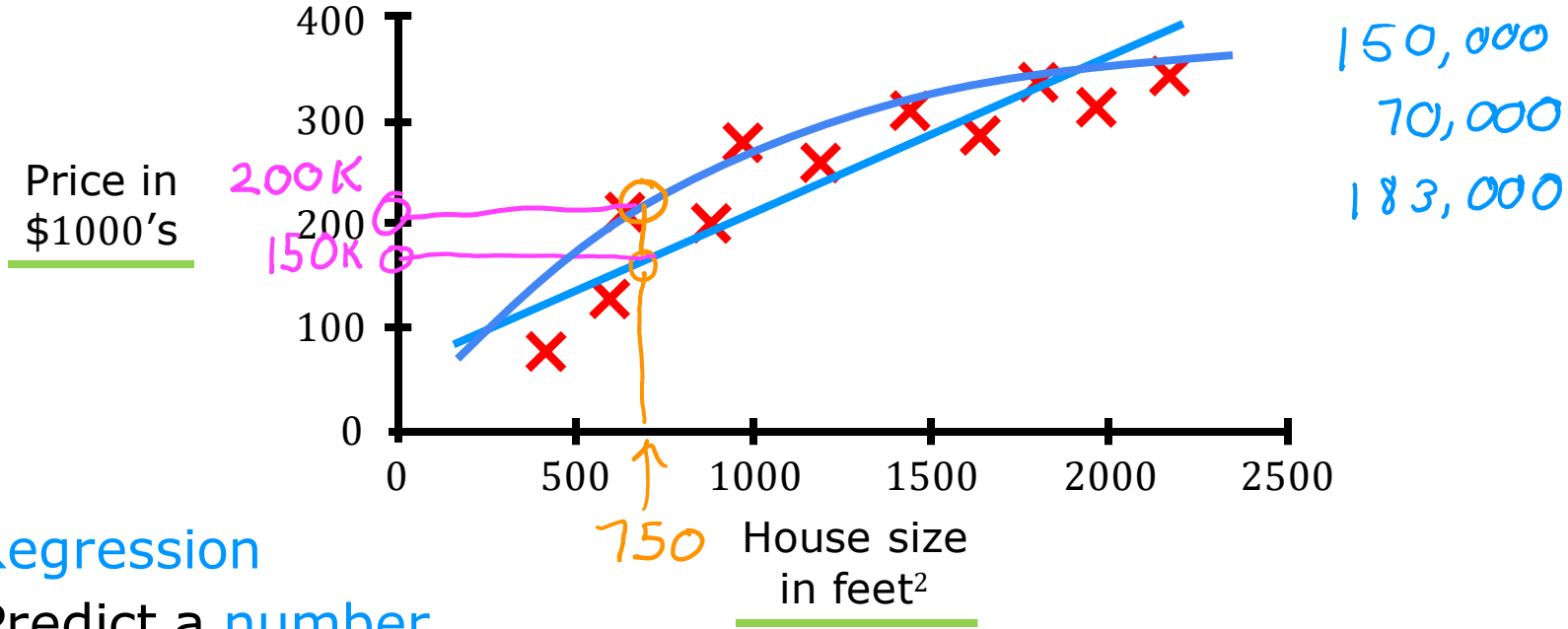


Learns from being given "right answers"

(correct label y for input x)

Input (X)	Output (Y)	Application
email	spam? (0/1)	spam filtering
audio	text transcripts	speech recognition
English	Spanish	machine translation
ad, user info	click? (0/1)	online advertising
image, radar info	position of other cars	self-driving car
image of phone	defect? (0/1)	visual inspection

Regression: Housing price prediction



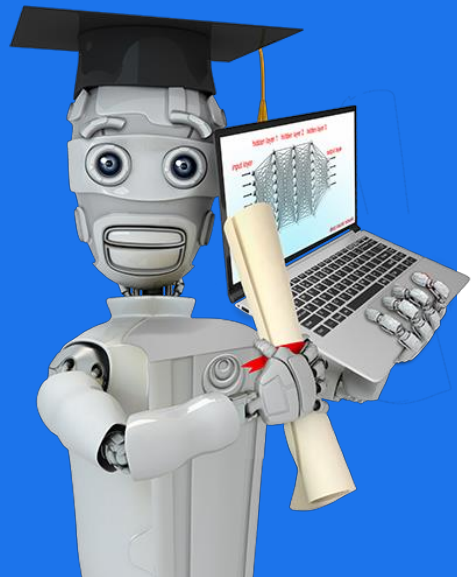
Regression

Predict a number

infinitely many possible outputs

Stanford
ONLINE

DeepLearning.AI

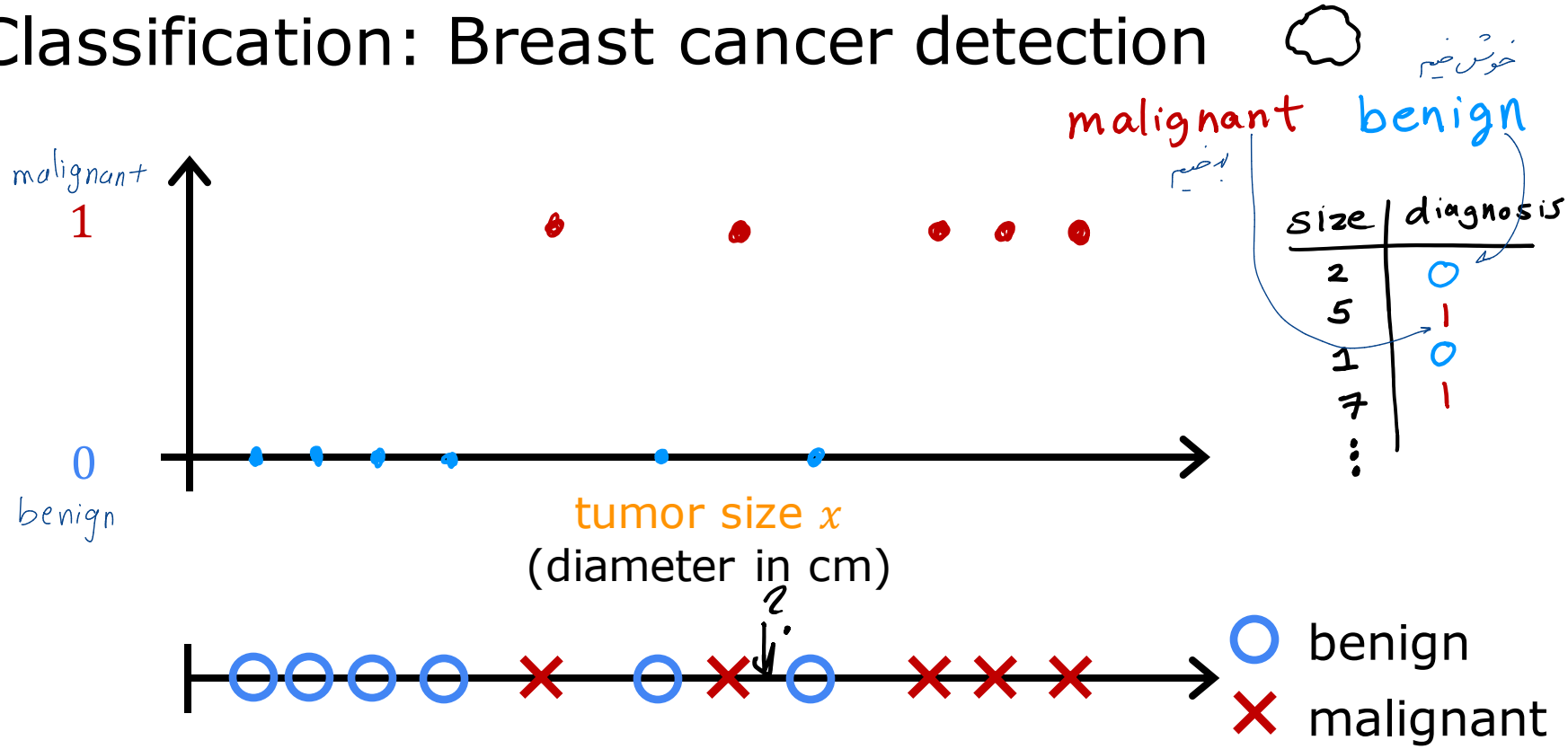


Machine Learning Overview

predict X to Y mappings

Supervised Learning Part 2

Classification: Breast cancer detection

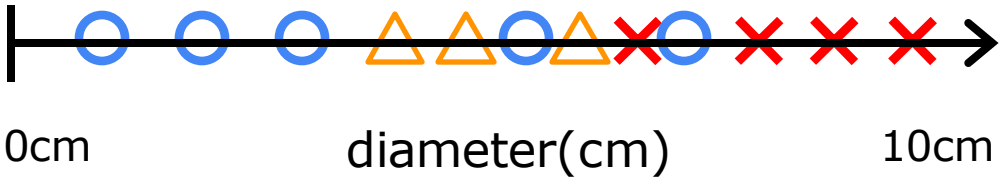


Classification: Breast cancer detection

benign

malignant type1

malignant type 2



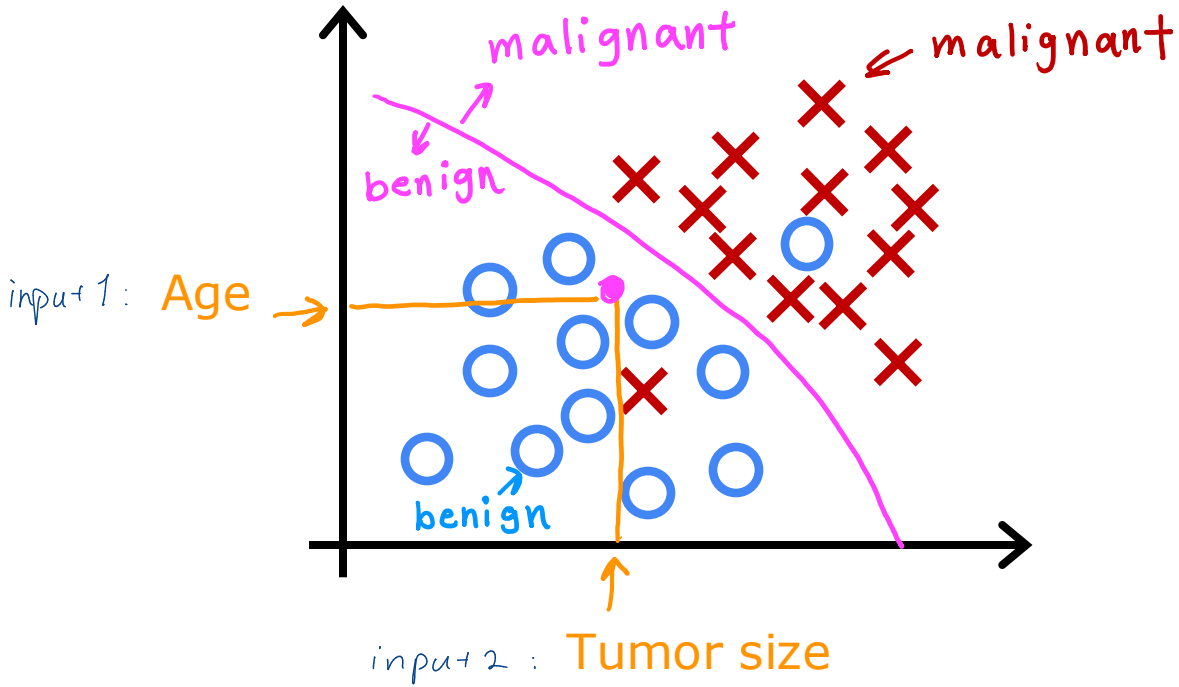
class / category

Classification

predict categories cat dog / benign malignant / 0, 1, 2

small number of possible outputs

Two or more inputs



Supervised learning

Learns from being given "right answers"

1

Regression

Predict a number

infinitely many possible outputs

2

Classification

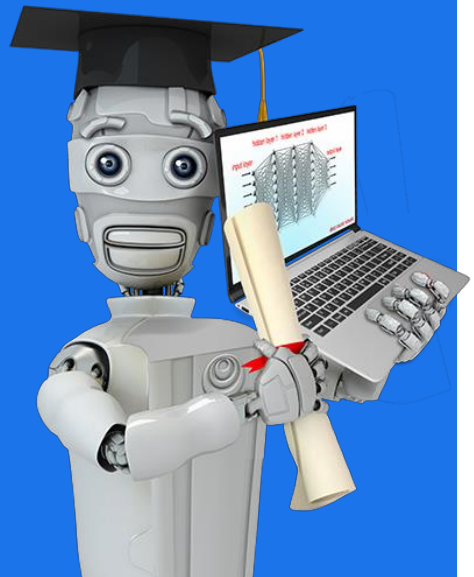
predict categories

small number of possible outputs

finite

Stanford
ONLINE

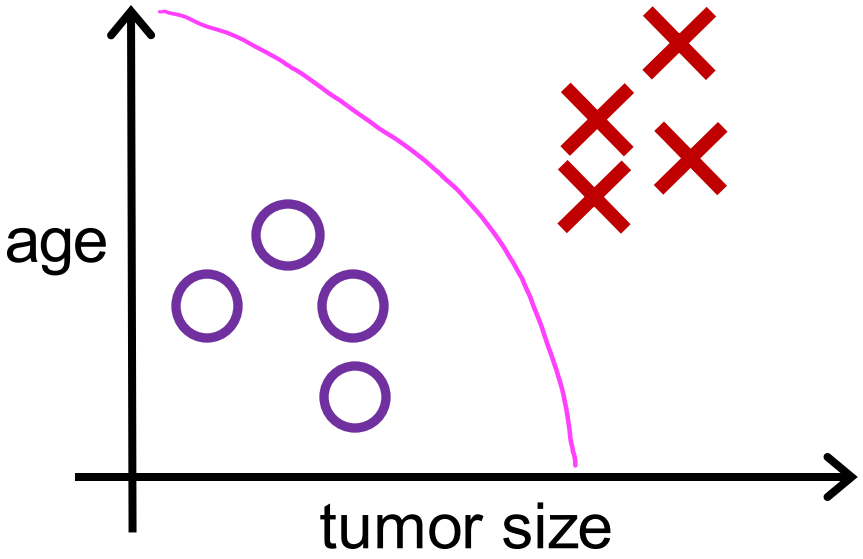
DeepLearning.AI



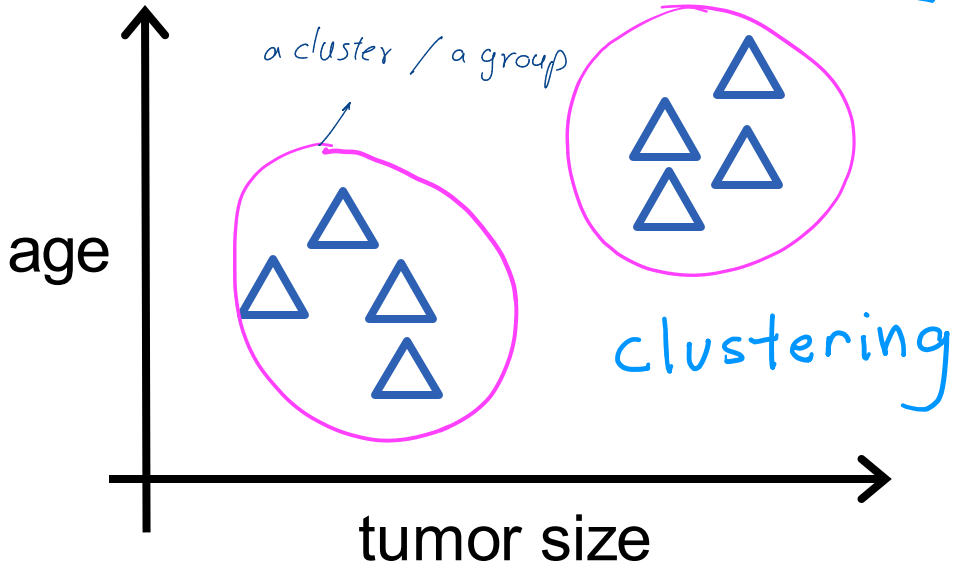
Machine Learning Overview

Unsupervised Learning Part 1

Supervised learning
Learn from data labeled with the "right answers"

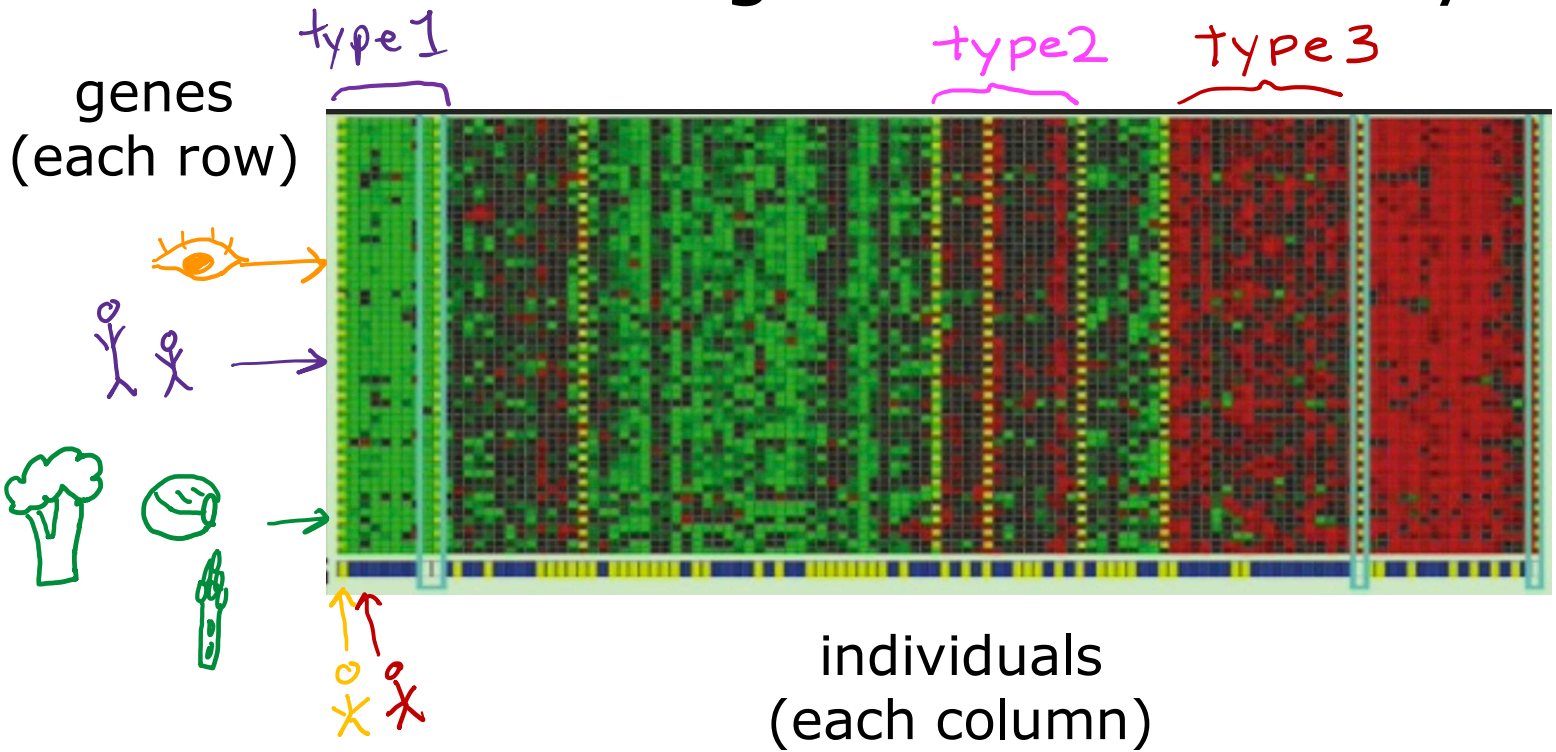


Unsupervised learning
Find something interesting in unlabeled data.

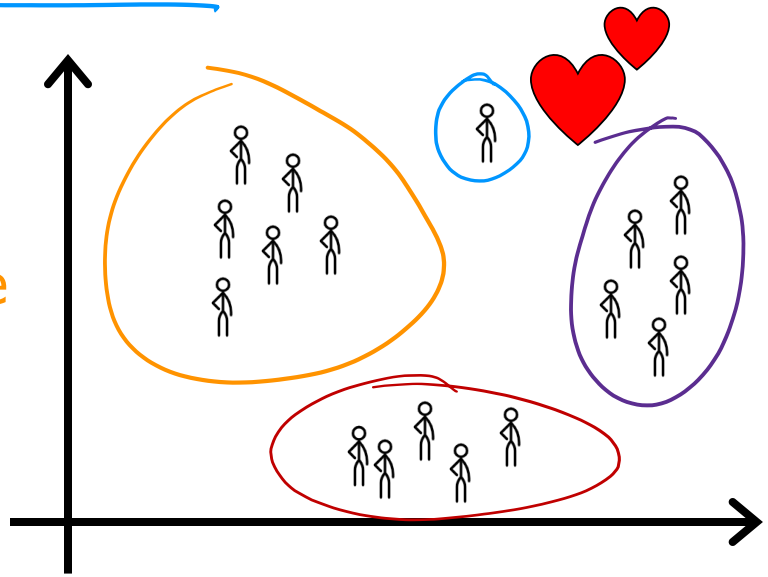


Clustering: Google news

Clustering: DNA microarray



Clustering: Grouping customers



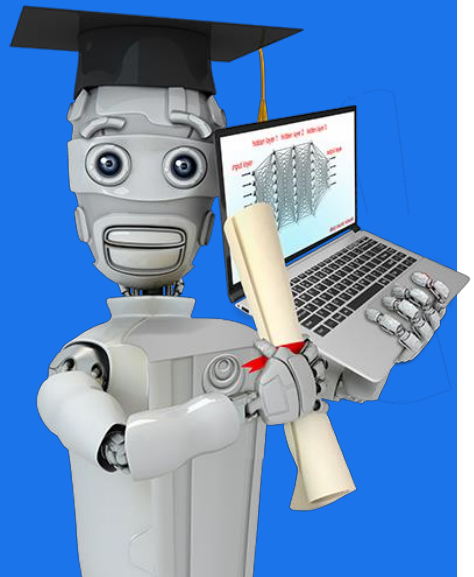
growing skills and knowledge

develop career

stay updated with AI

Stanford
ONLINE

DeepLearning.AI



Machine Learning Overview

Unsupervised Learning Part 2

Unsupervised learning

Data only comes with inputs x, but not output labels y.
Algorithm has to find structure in the data.

or
pattern

Clustering
Group similar data points together.

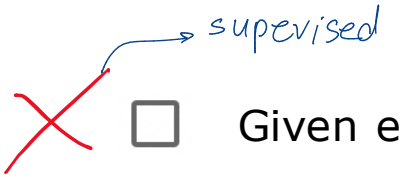
Dimensionality reduction
Compress data using fewer numbers.

Anomaly detection
Find unusual data points.

or
events

Question

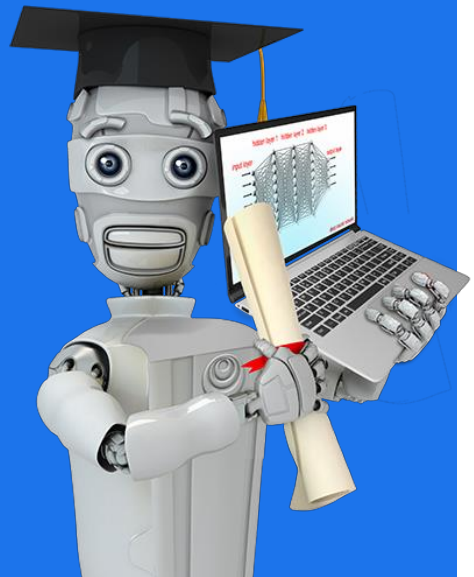
Of the following examples, which would you address using an unsupervised learning algorithm?



- [ ] Given email labeled as spam/not spam, learn a spam filter.
- [x] Given a set of news articles found on the web, group them into sets of articles about the same story.
- [x] 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

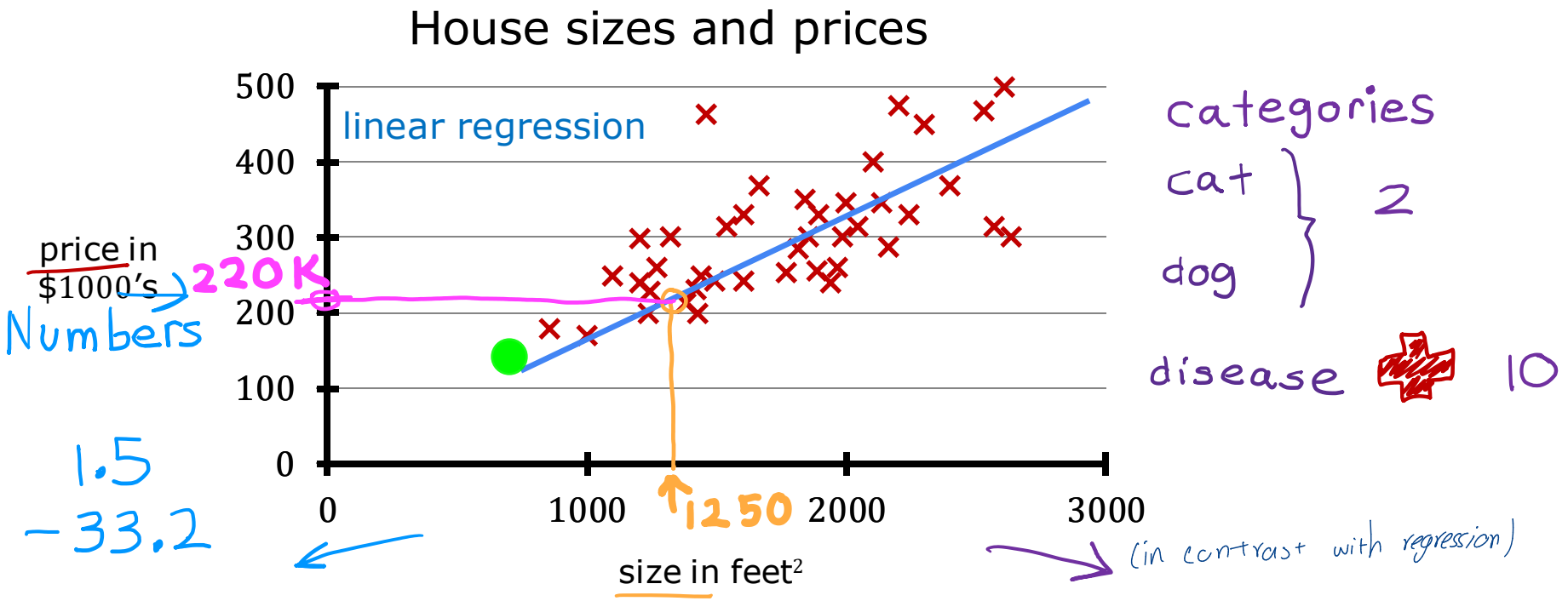
Stanford
ONLINE

DeepLearning.AI



Linear Regression with One Variable

Linear Regression Model Part 1



Regression model
Predicts numbers
Infinitely many possible outputs

Supervised learning model
Data has "right answers"

Classification model
Predicts categories
Small number of possible outputs

House sizes and prices

Data table

size in feet²	price in $1000's
2104	400
1416	232
1534	315
852	178
...	...
3210	870

Terminology

Training set: Data used to train the model

	x size in feet^2	y price in \$1000's
(1)	2104	400
(2)	1416	232
(3)	1534	315
(4)	852	178
…	…	…
(47)	3210	870

$m = 47$

$x^{(1)} = 2104$ $y^{(1)} = 400$

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

$x^{(2)} = 1416$ $y^{(2)} \neq x^2$ not exponent

Notation:

x = "input" variable / input feature
feature

y = "output" variable
"target" variable

m = number of training examples

(x, y) = single training example

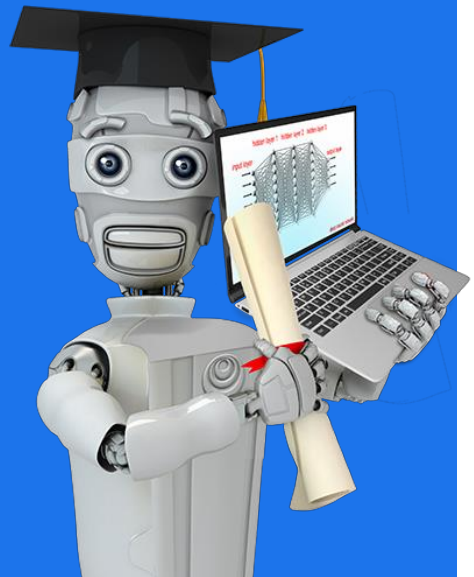
$(x^{(i)}, y^{(i)})$ → specific row

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

index (1^{st}, 2^{nd}, 3^{rd} …)

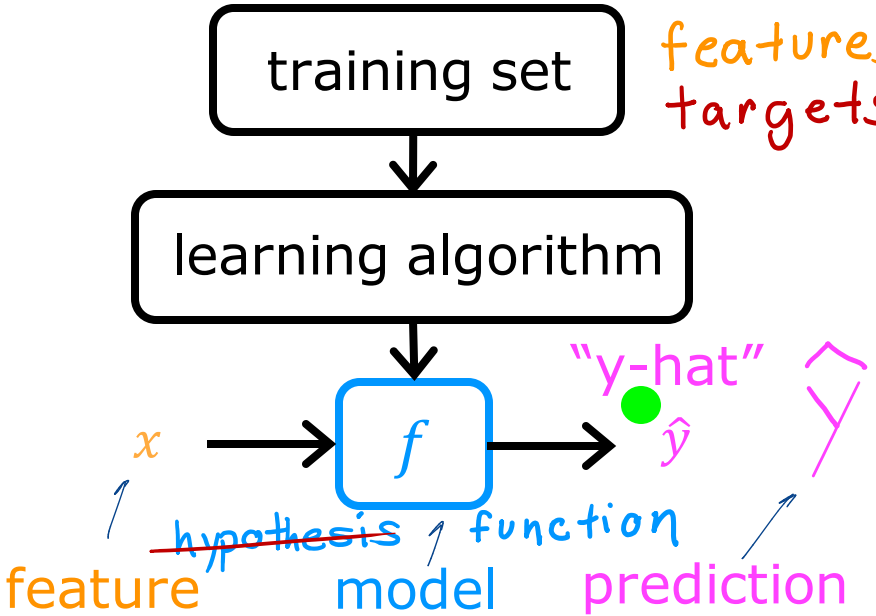
Stanford
ONLINE

DeepLearning.AI



Linear Regression with One Variable

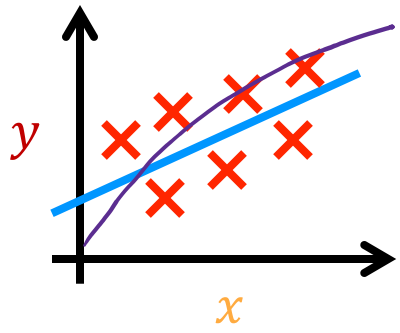
Linear Regression Model Part 2



How to represent f?

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

$$f(x)$$



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

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

linear

single feature x

Linear regression with one variable.

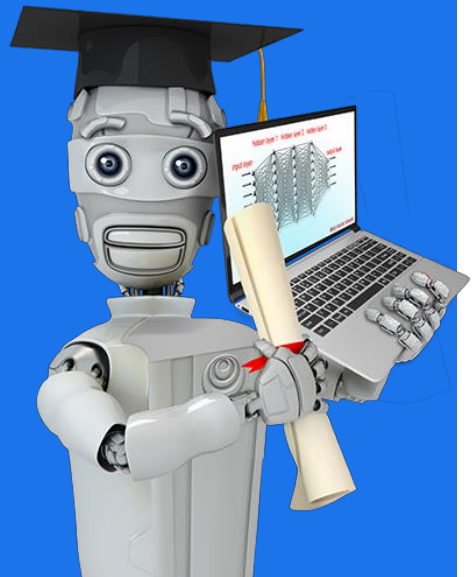
size

Univariate linear regression.

one variable

Stanford
ONLINE

DeepLearning.AI



Linear Regression with One Variable

Cost Function

Training set

features | targets

size in feet² (x)	price \$1000's (y)
2104	460
1416	232
1534	315
852	178
...	...

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

w, b: parameters

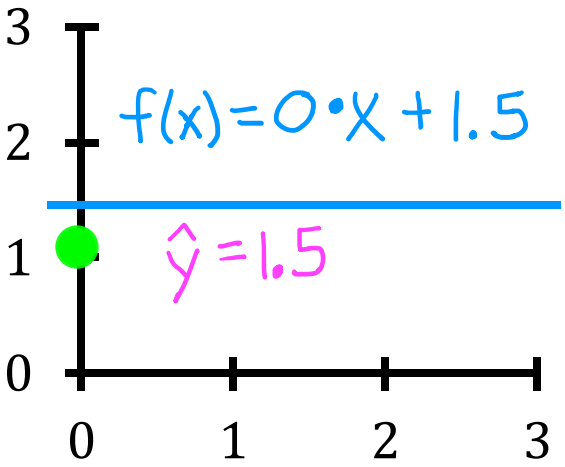
coefficients

weights

What do w, b do?

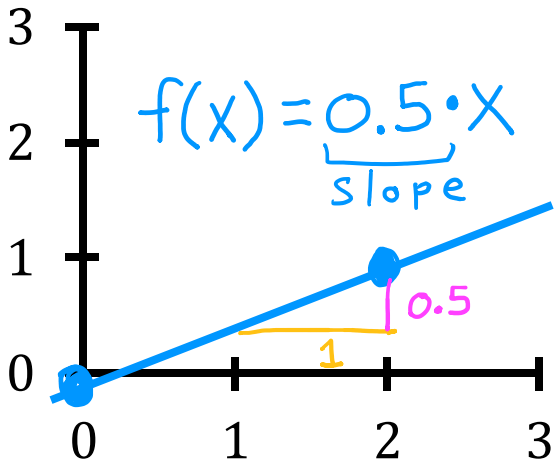
$$f_{w,b}(x) = wx + b$$

$f(x)$



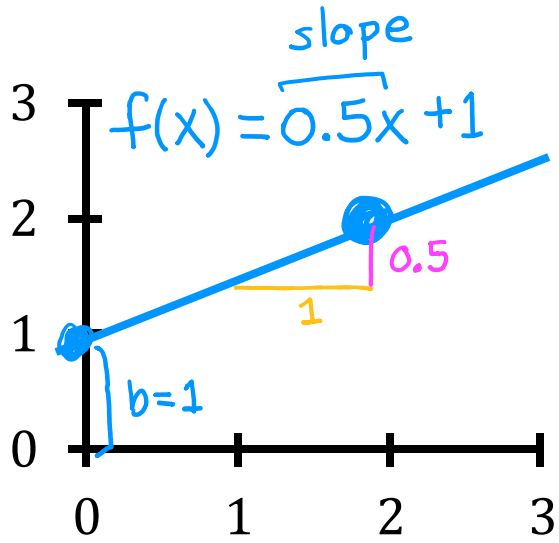
→ $w = 0$

→ $b = 1.5$ (y-intercept)



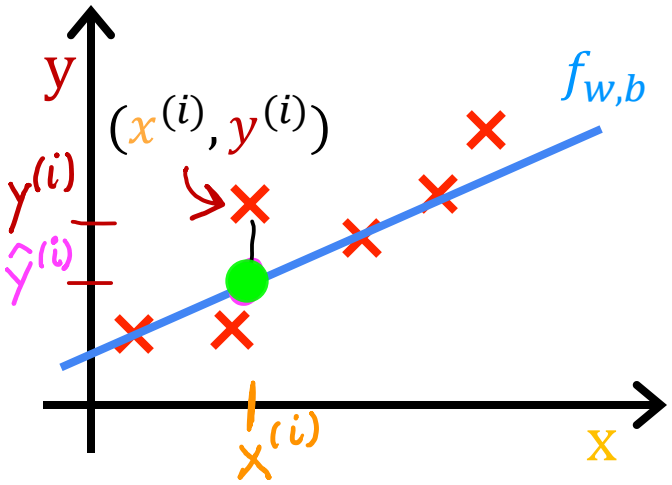
→ $w = 0.5$

→ $b = 0$



→ $w = 0.5$

→ $b = 1$



$$\hat{y}^{(i)} = f_{w,b}(x^{(i)})$$

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

Cost function: Squared error cost function

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

$\hat{y}^{(i)}$ − $y^{(i)}$: error

m = number of training examples

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

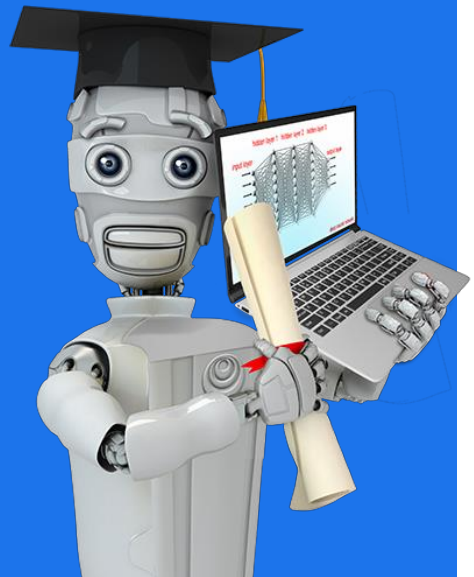
$J(w,b)$: intuition (next!)

Find w, b:

$\hat{y}^{(i)}$ is close to $y^{(i)}$ for all $(x^{(i)}, y^{(i)})$.

Stanford
ONLINE

DeepLearning.AI



Linear Regression with One Variable

Cost Function Intuition

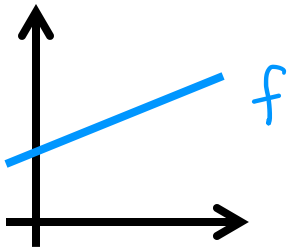
درک مستقیم / بینش

model:

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

parameters:

w, b



cost function:

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

prediction

true value

goal:

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

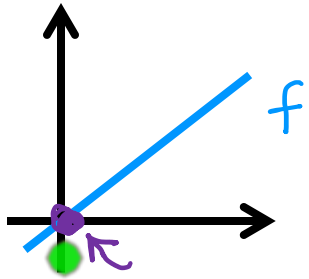
simplified

$$f_w(x) = wx$$

zero

$b = 0$

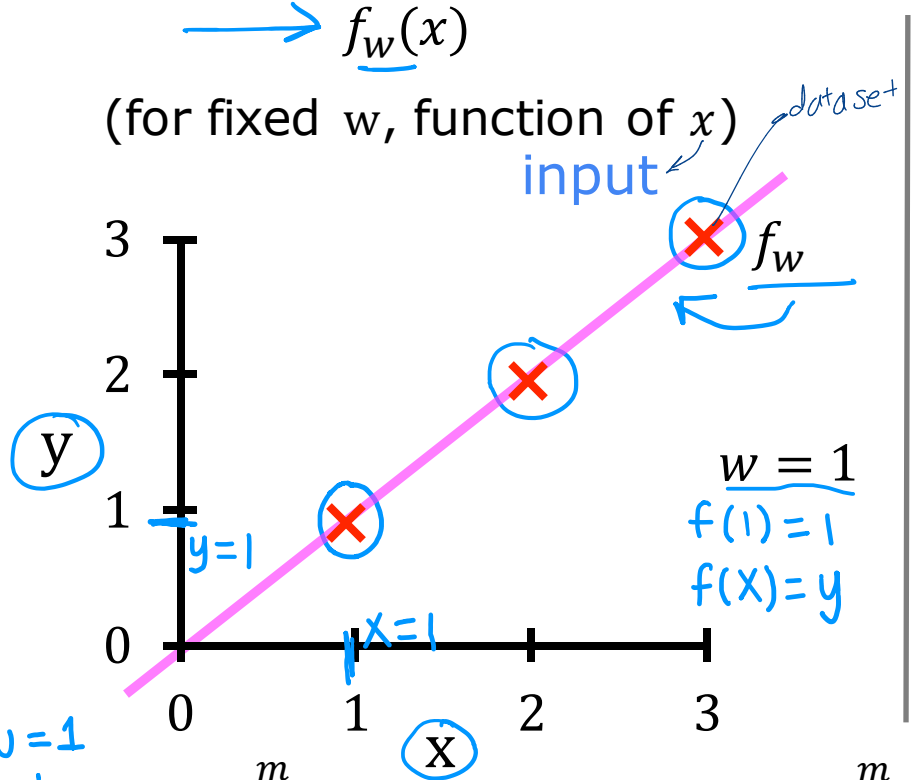
w

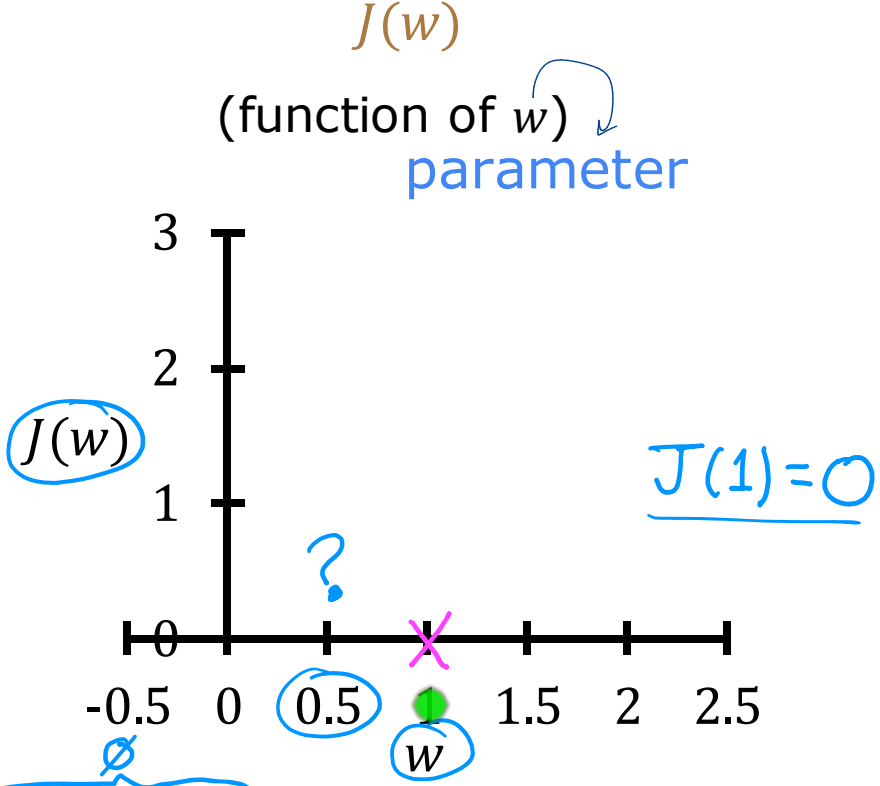


$$J(w) = \frac{1}{2m}\sum_{i=1}^{m}(f_w(x^{(i)}) - y^{(i)})^2$$

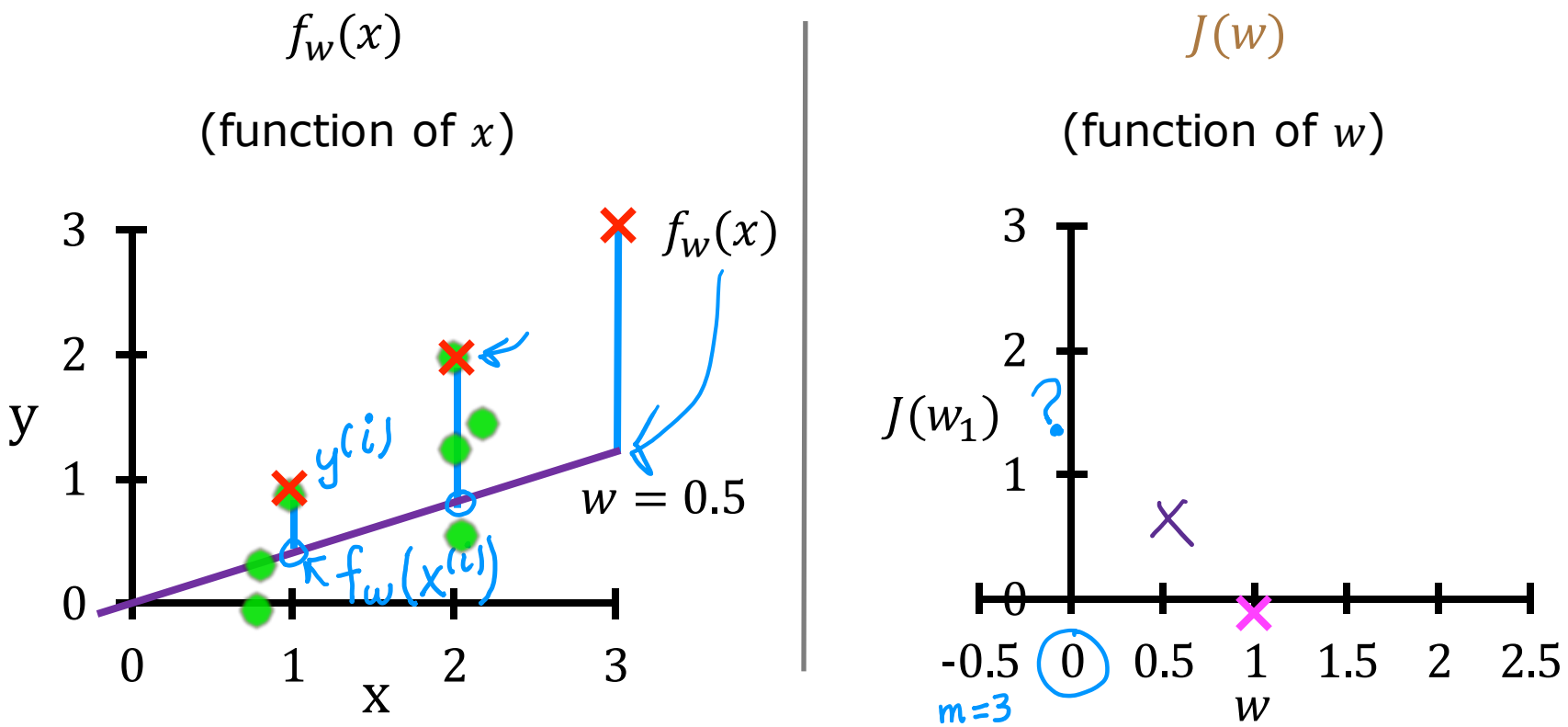
$wx^{(i)}$

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

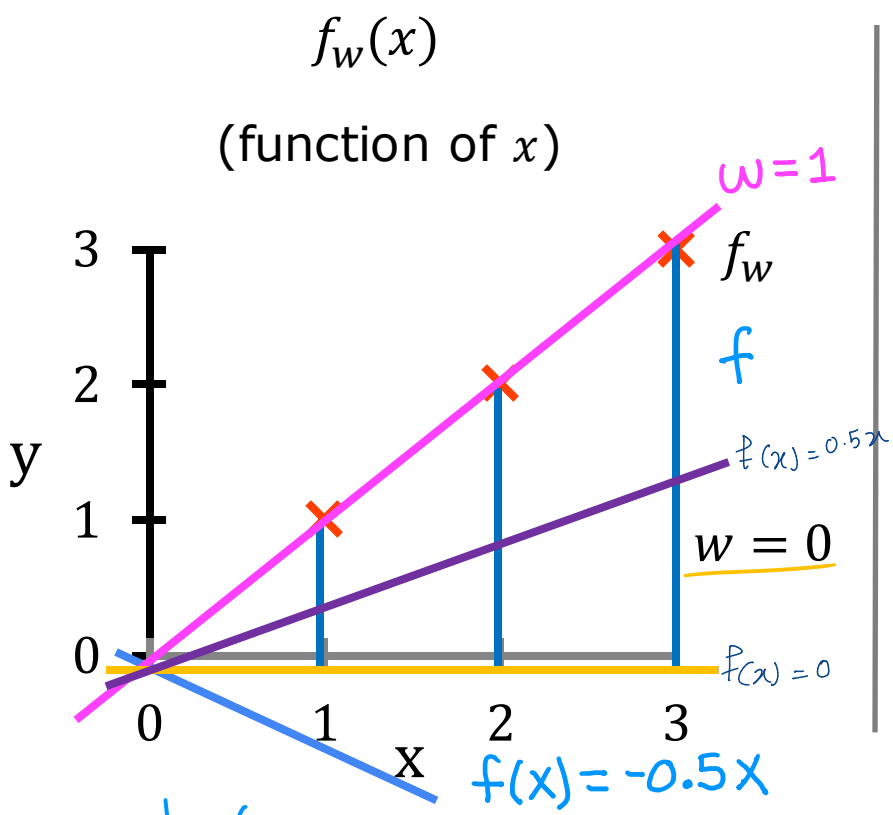


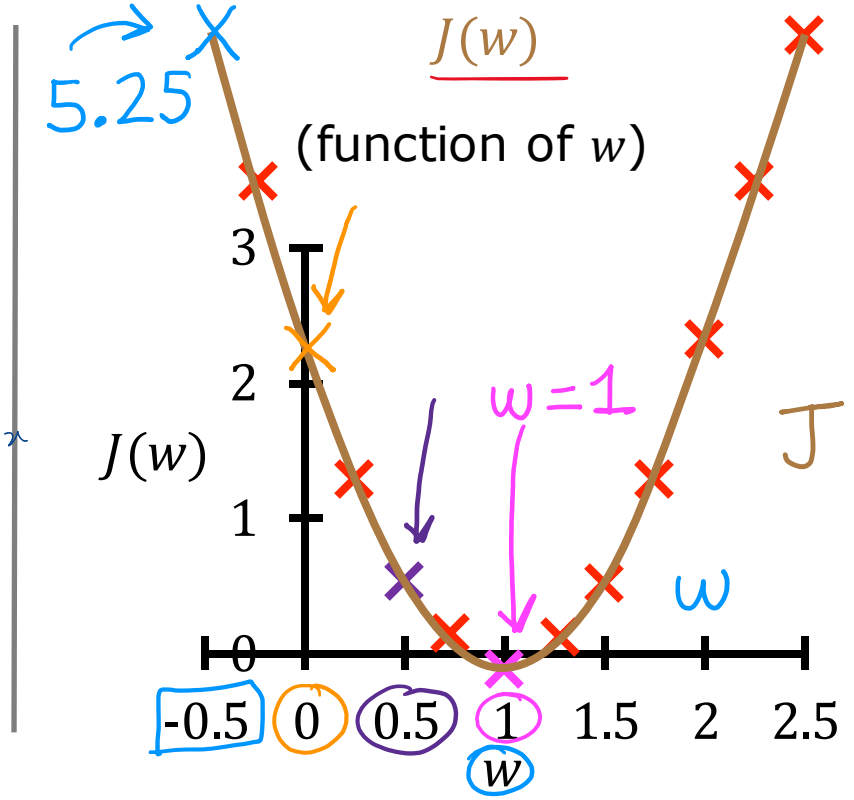


$$J(w) = \frac{1}{2m}\sum_{i=1}^{m}(f_w(x^{(i)}) - y^{(i)})^2 = \frac{1}{2m}\sum_{i=1}^{m}(wx^{(i)} - y^{(i)})^2 = \frac{1}{2m}(0^2 + 0^2 + 0^2) = 0$$



$$J(0.5) = \frac{1}{2m}\left[(0.5-1)^2+(1-2)^2+(1.5-3)^2\right] = \frac{1}{2\times 3}[3.5] = \frac{3.5}{6} \approx 0.58$$





$$J(0) = \frac{1}{2m}\left(1^2+2^2+3^2\right) = \frac{1}{6}[14] \approx 2.3$$

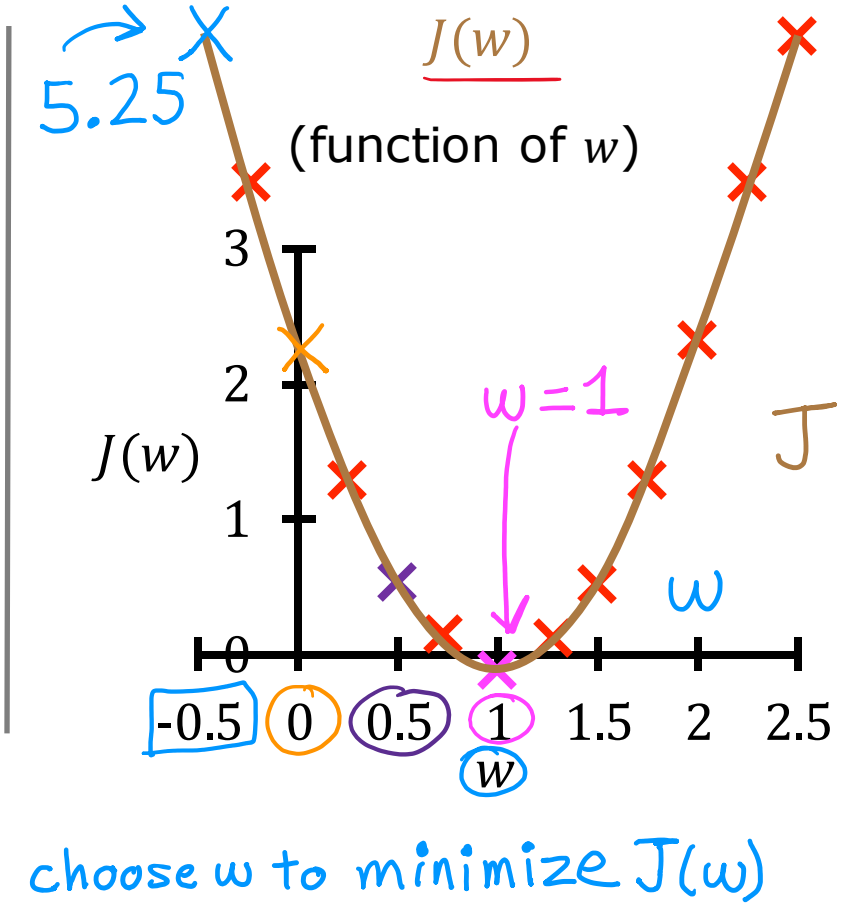
how to choose w?

goal of linear regression:

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

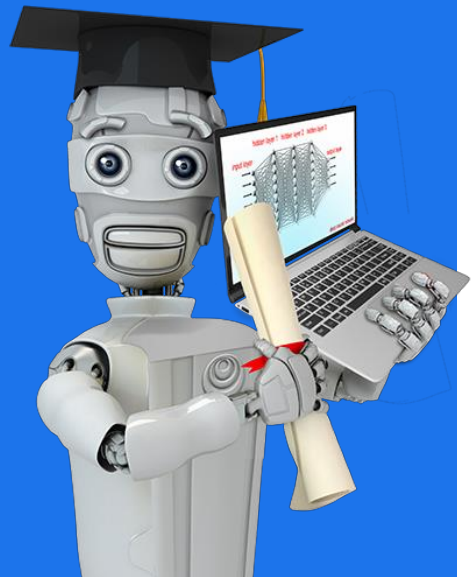
general case:

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



Stanford
ONLINE

DeepLearning.AI



Linear Regression with One Variable

Visualizing the Cost Function

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

~~before: $b = 0$~~

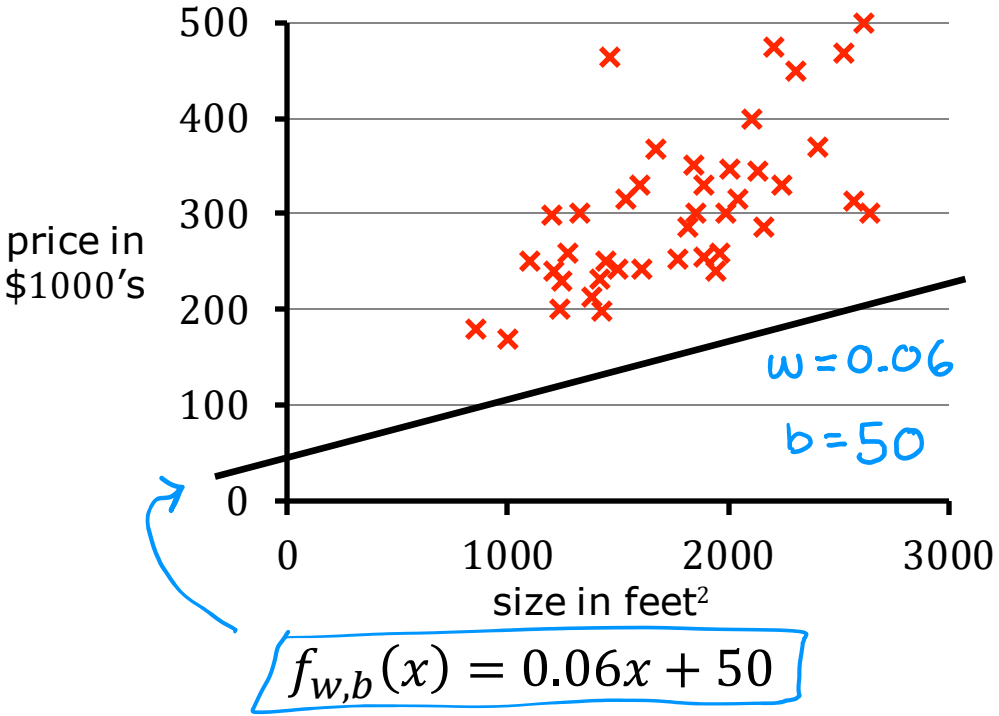
Parameters w, b

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

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

$f_{w,b}$

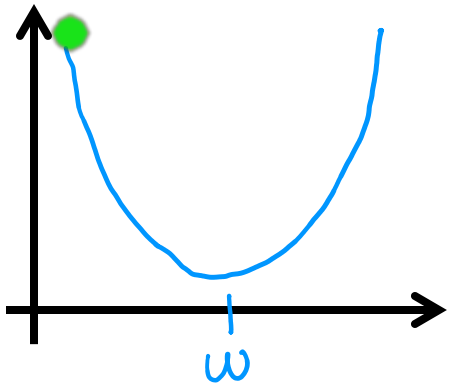
(function of x)



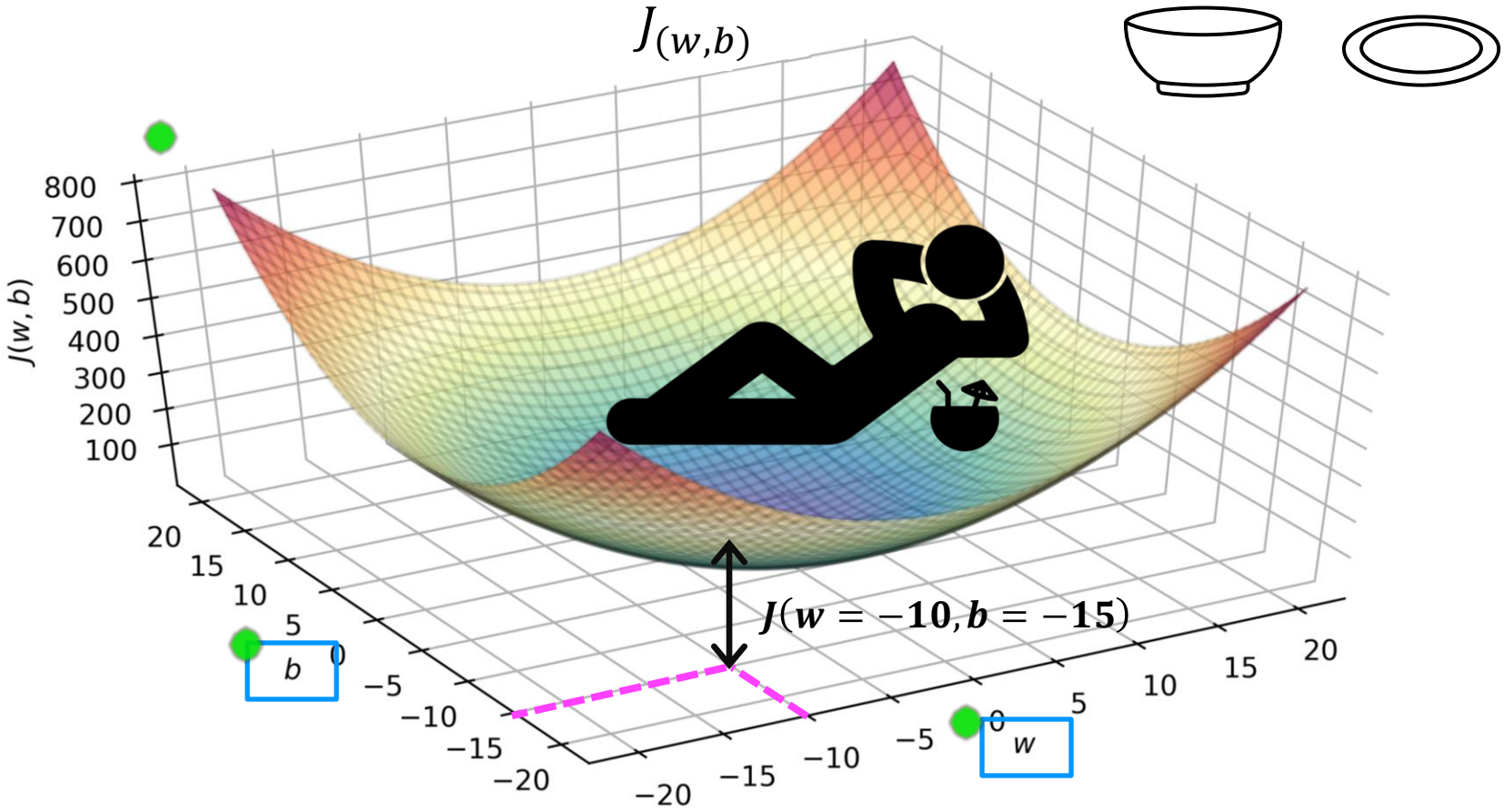
$f_{w,b}(x) = 0.06x + 50$

J

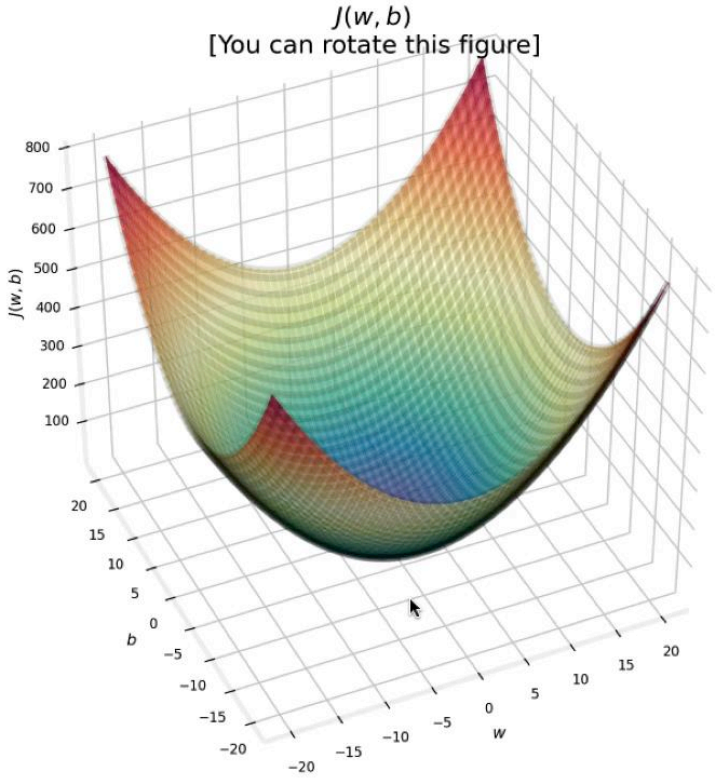
(function of w, b)



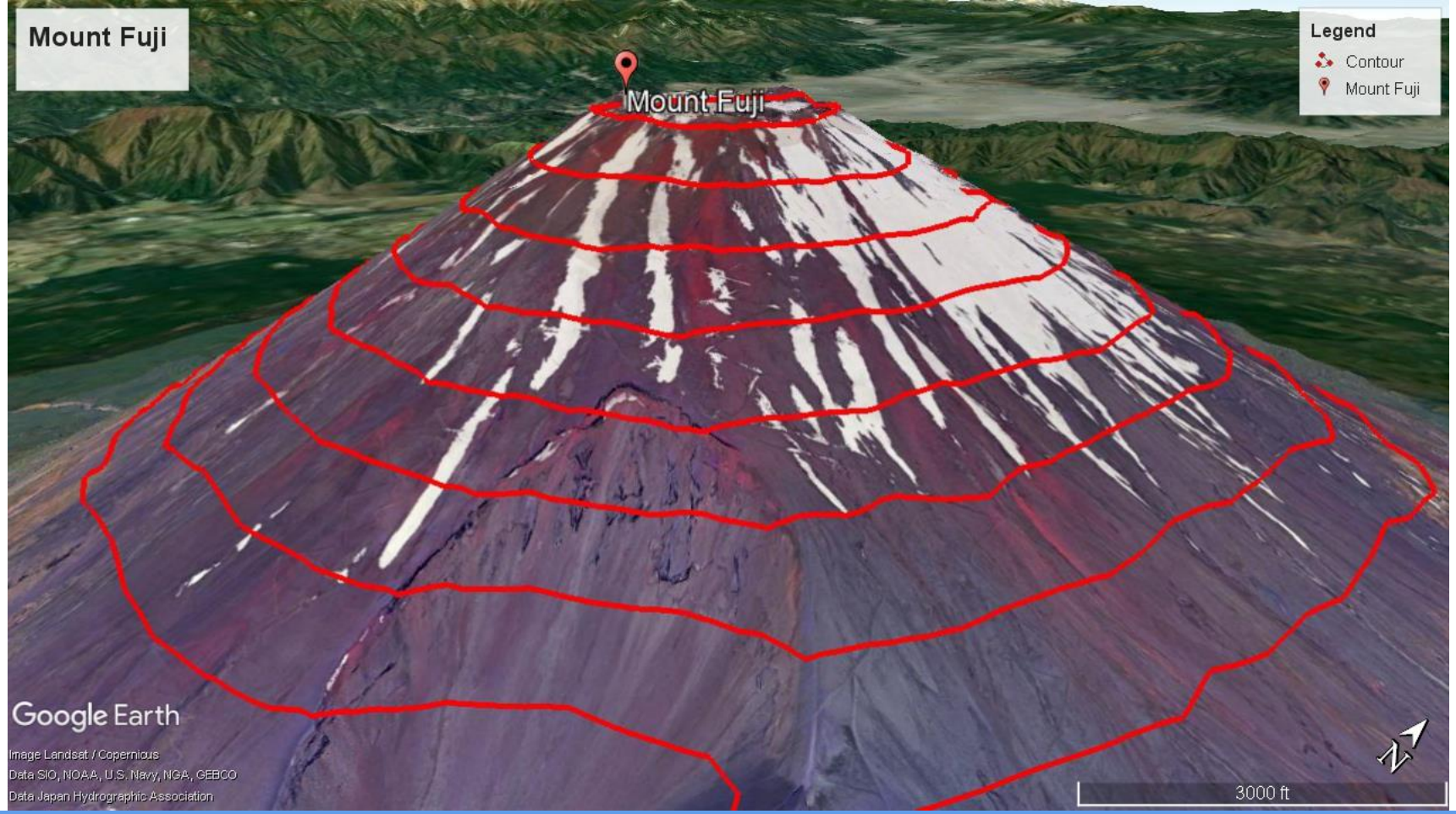
w , b

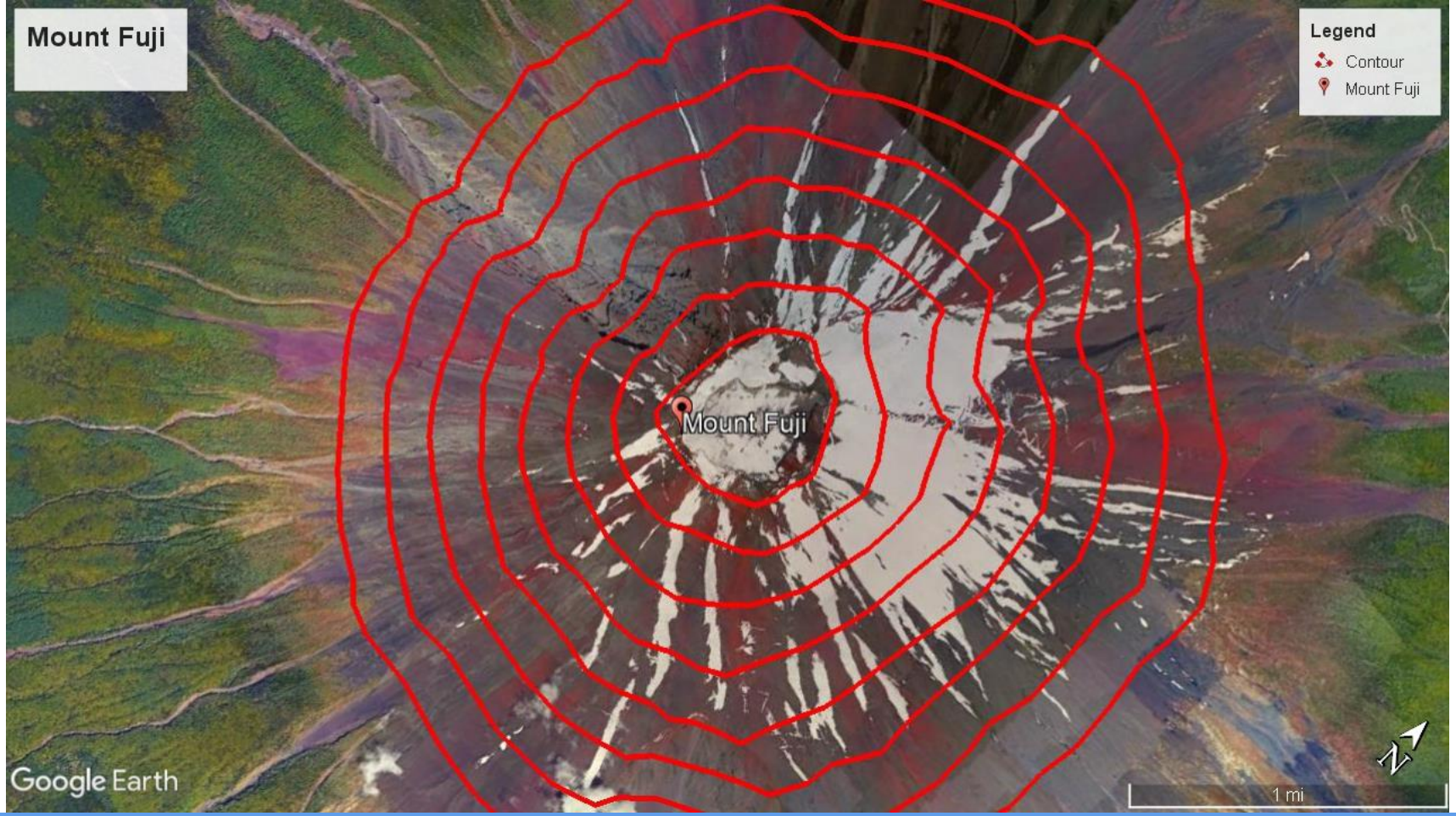

$J_{(w,b)}$
800
700
600
500
400
300
200
100
$J(w,b)$
20
15
10
5
0
b
−5
−10
−15
−20
−20
−15
−10
−5
0
5
10
15
20
w
$J(w=-10, b=-15)$

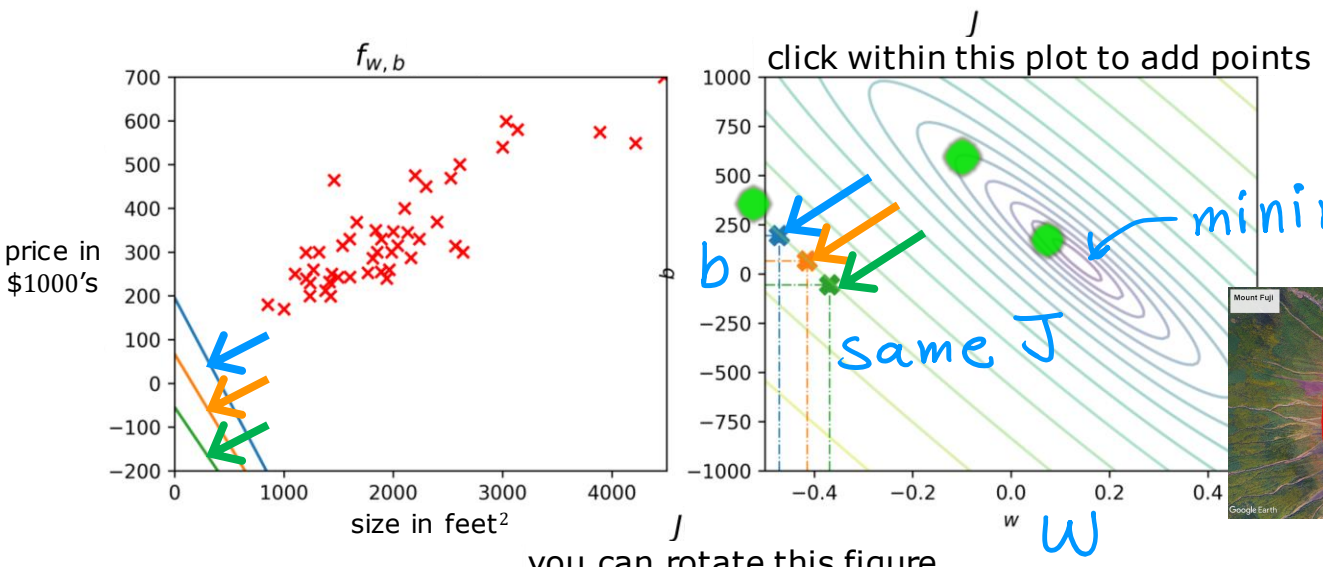
3D surface plot

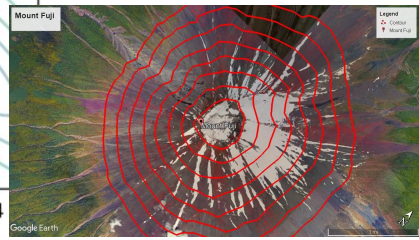


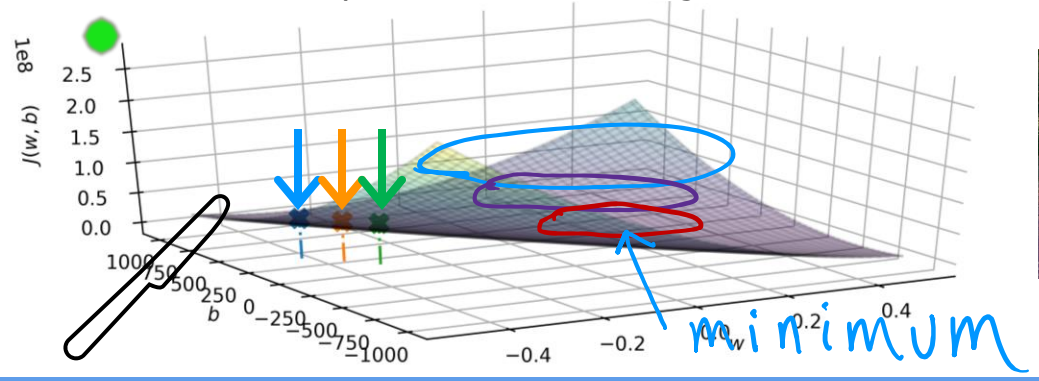
Alternative contour plot

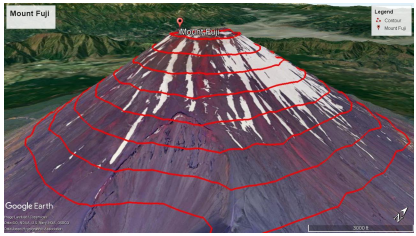

Mount Fuji
Legend
Contour
Mount Fuji
Mount Fuji
Google Earth
Image Landsat / Copernicus
Data SIO, NOAA, U.S. Navy, NGA, GEBCO
Data Japan Hydrographic Association
3000 ft


Mount Fuji
Legend
Contour
Mount Fuji
Mount Fuji
Google Earth
1 mi


$f_{w,b}$
price in $1000's
size in feet^2
J
click within this plot to add points
b
w
minimum
same J

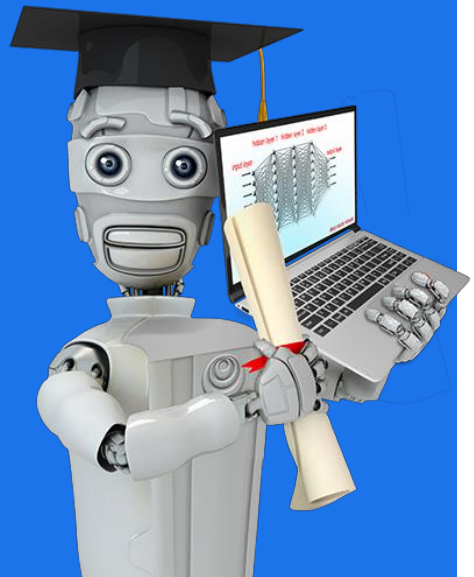

Mount Fuji
Google Earth


J
you can rotate this figure
J(w, b)
1e8
b
w
minimum


Mount Fuji
Google Earth

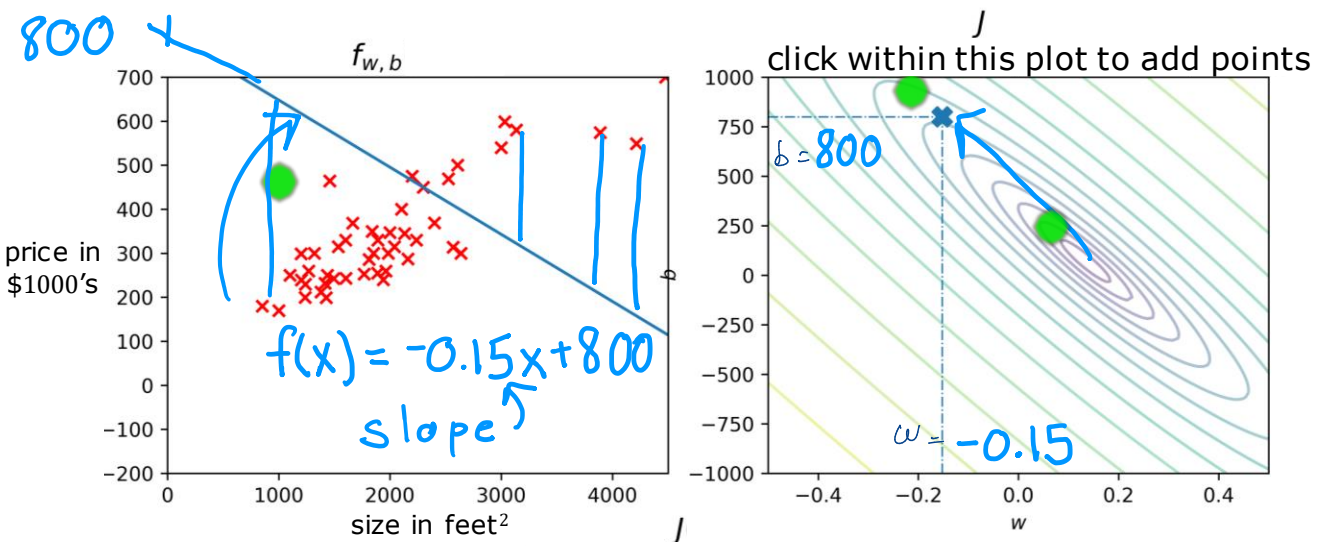
Stanford
ONLINE

DeepLearning.AI

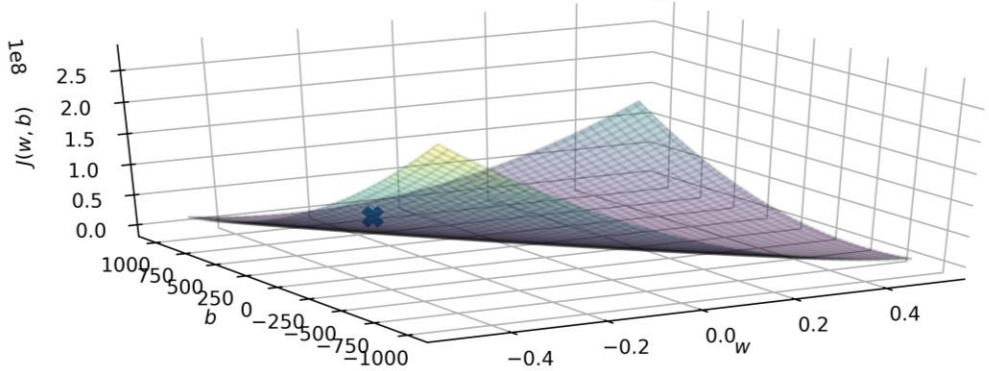


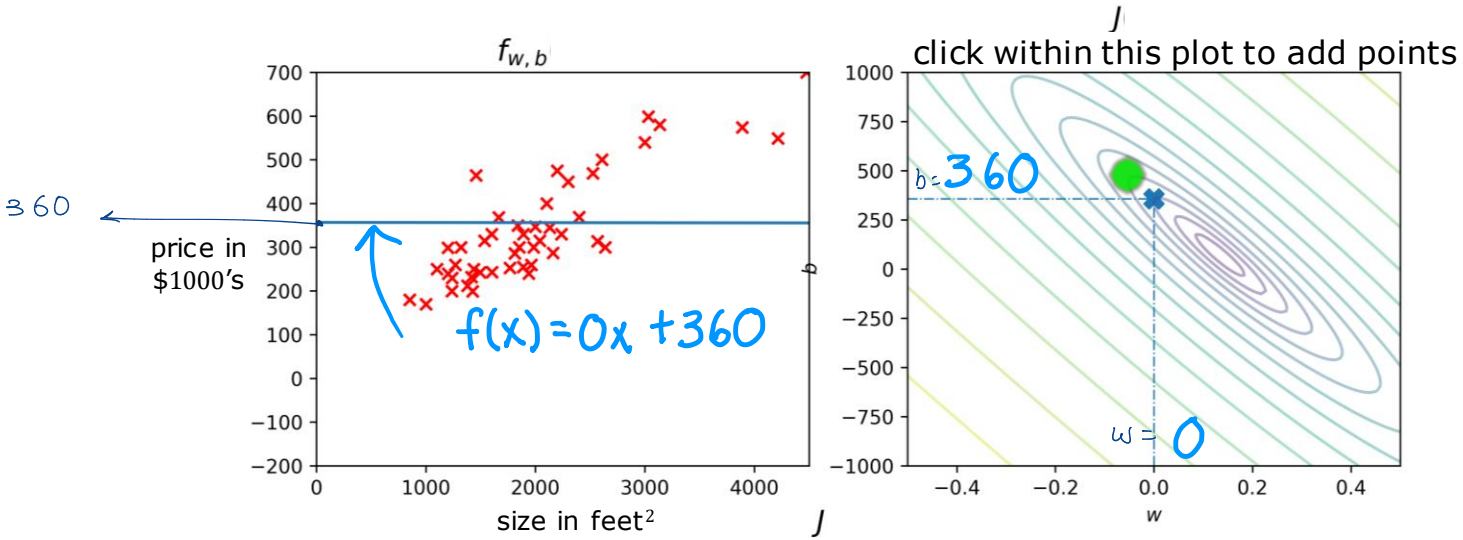
Linear Regression with One Variable

Visualization examples

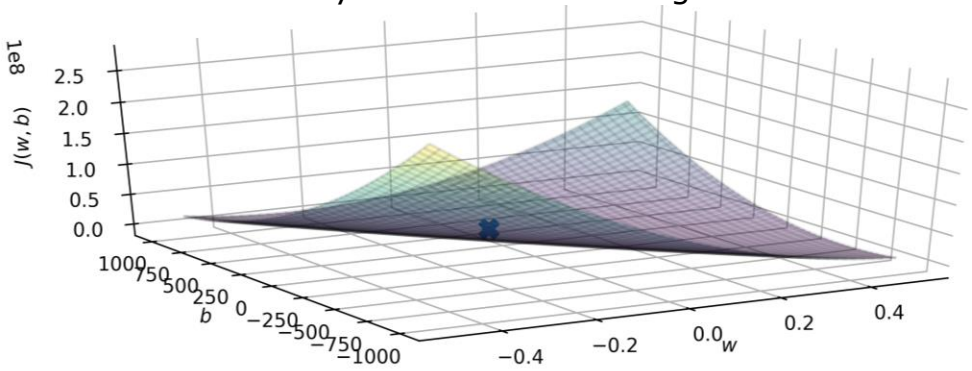


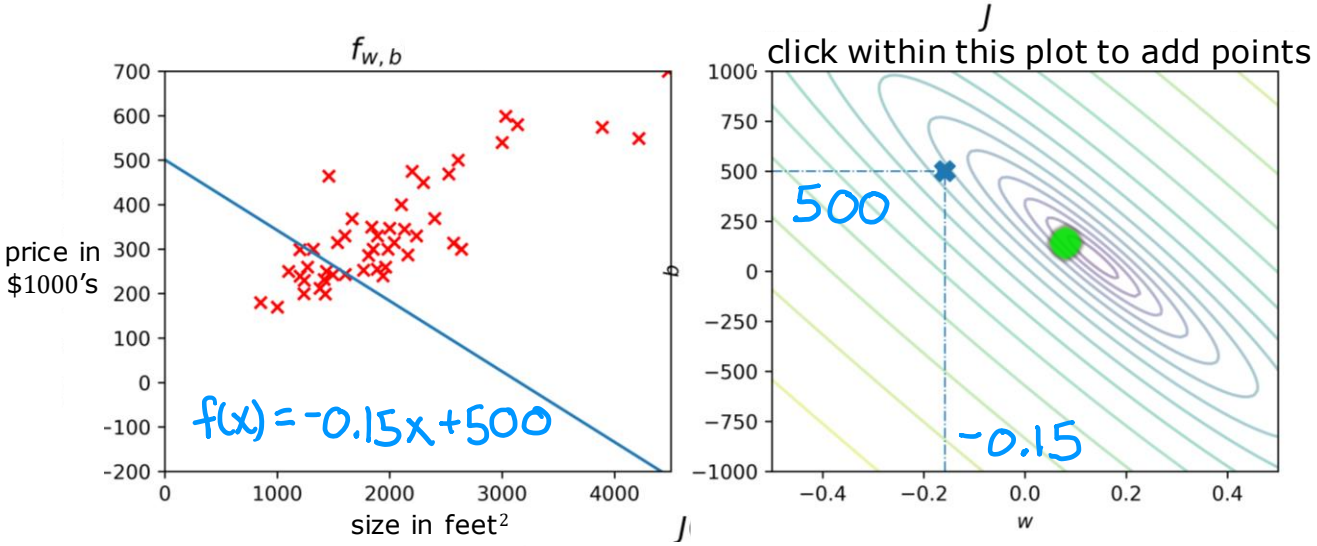
you can rotate this figure



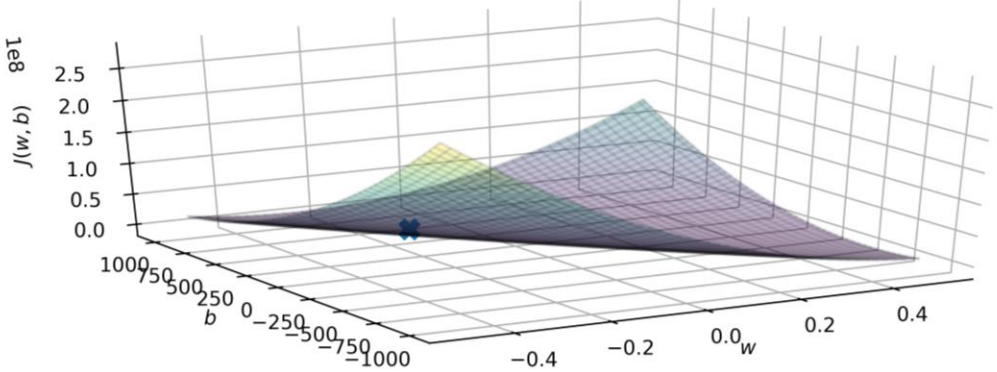


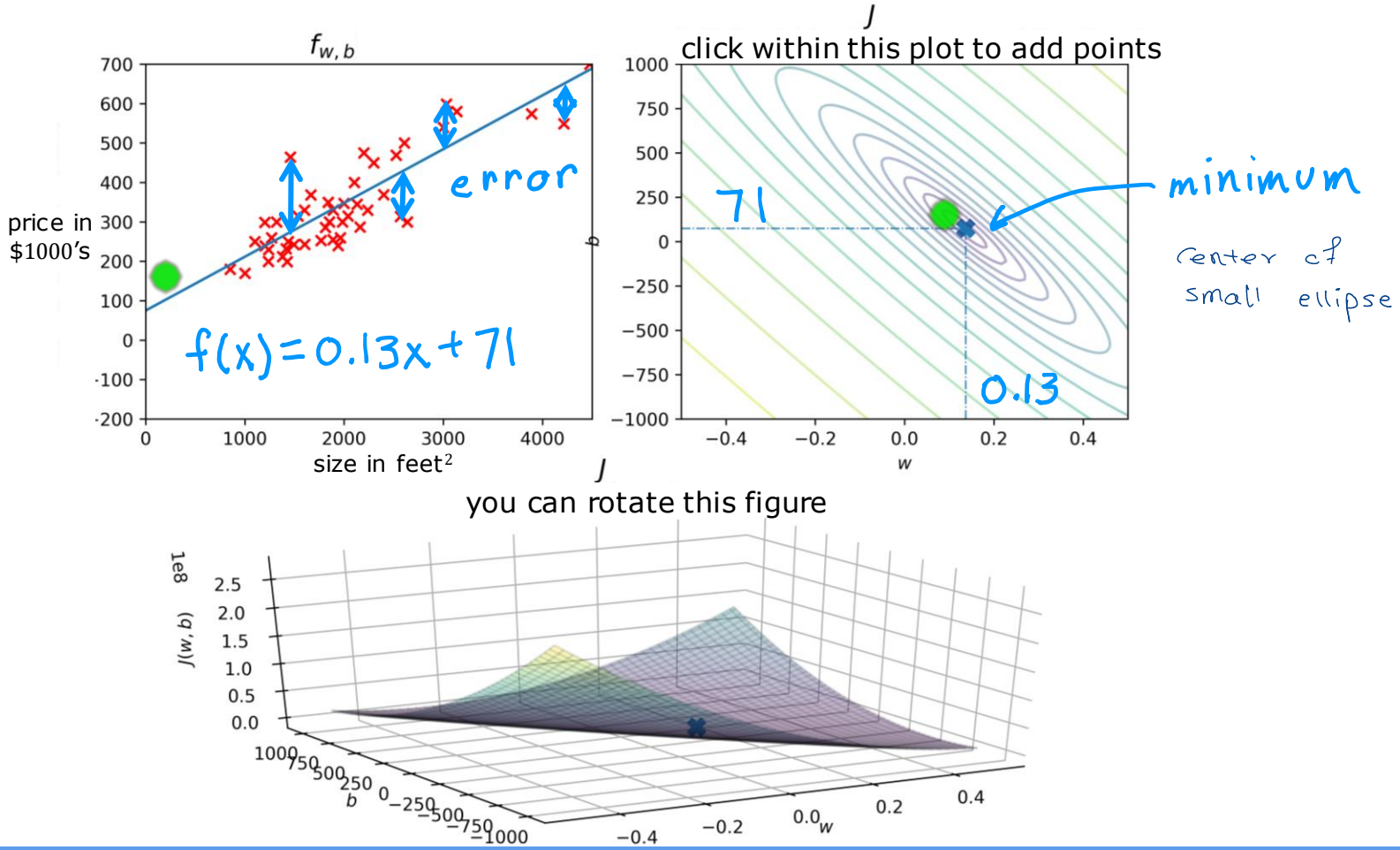
you can rotate this figure





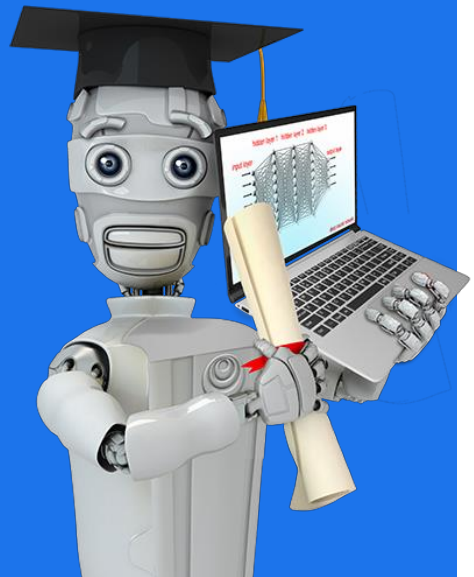
you can rotate this figure




$f_{w,b}$
price in $1000's
size in feet^2
error
$f(x) = 0.13x + 71$
J
click within this plot to add points
b
w
71
0.13
minimum
center of small ellipse
J
you can rotate this figure
1e8
J(w, b)
b
w

Stanford
ONLINE

DeepLearning.AI



Training Linear Regression

Gradient Descent

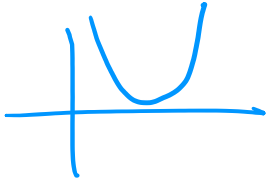
Have some function $J(w,b)$ for linear regression or any function

Want $\min_{w,b} J(w,b)$ $\min_{w_1,\dots,w_n,b} J(w_1,w_2,\dots,w_n,b)$

Outline:

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

J not always



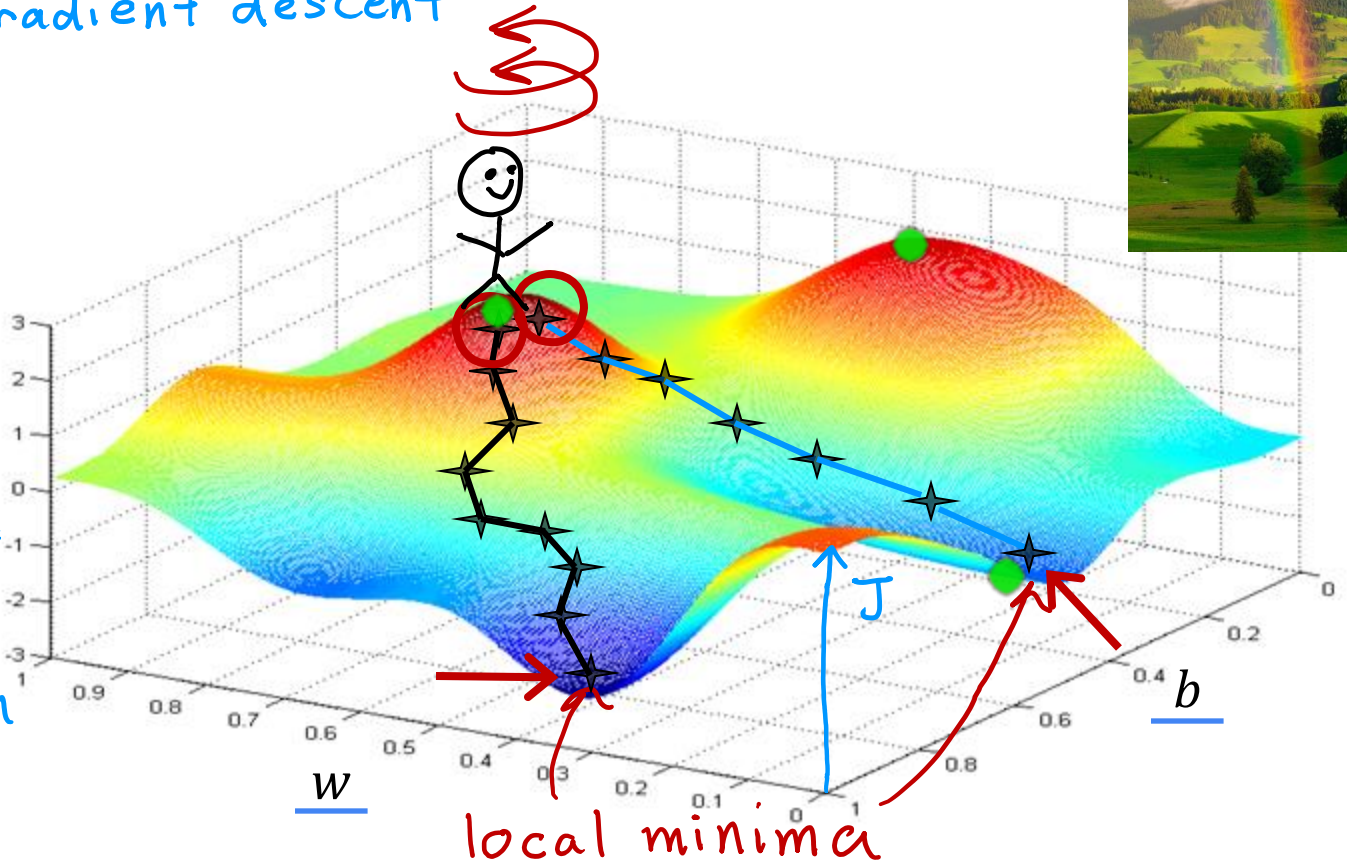
may have >1 minimum

gradient descent

$J(w, b)$

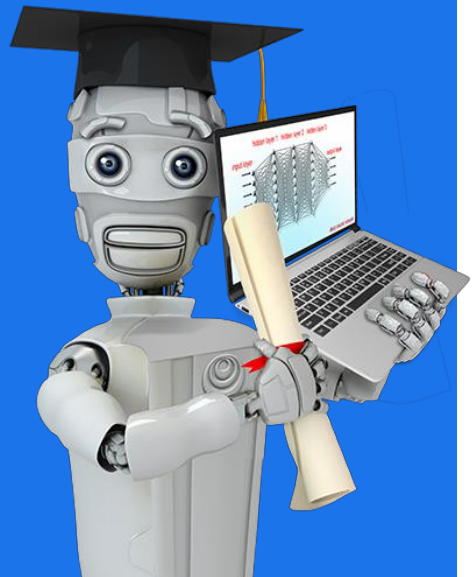
not squared error cost

not linear regression



Stanford
ONLINE

DeepLearning.AI



Training Linear Regression

Implementing Gradient Descent

Gradient descent algorithm

Repeat until convergence

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

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

$0 < \alpha < 1$

Learning rate

Derivative

Simultaneously update w and b

Assignment

$a = c$ (assign)

$a = a + 1$

Code

Truth assertion

$a = c$ (equal)

$a = a + 1$ ✗

Math

a==c

Correct: Simultaneous update

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

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

$$w = tmp\_w$$

$$b = tmp\_b$$

Incorrect

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

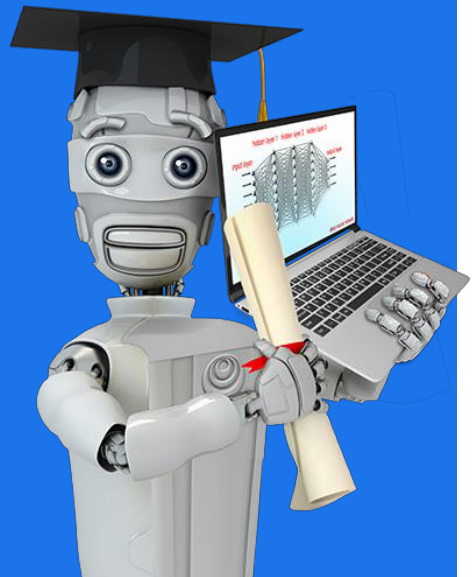
$$w = tmp\_w$$

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

$$b = tmp\_b$$

Stanford
ONLINE

DeepLearning.AI



Training Linear Regression

Gradient Descent Intuition

Gradient descent algorithm

repeat until convergence {

learning rate

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

derivative

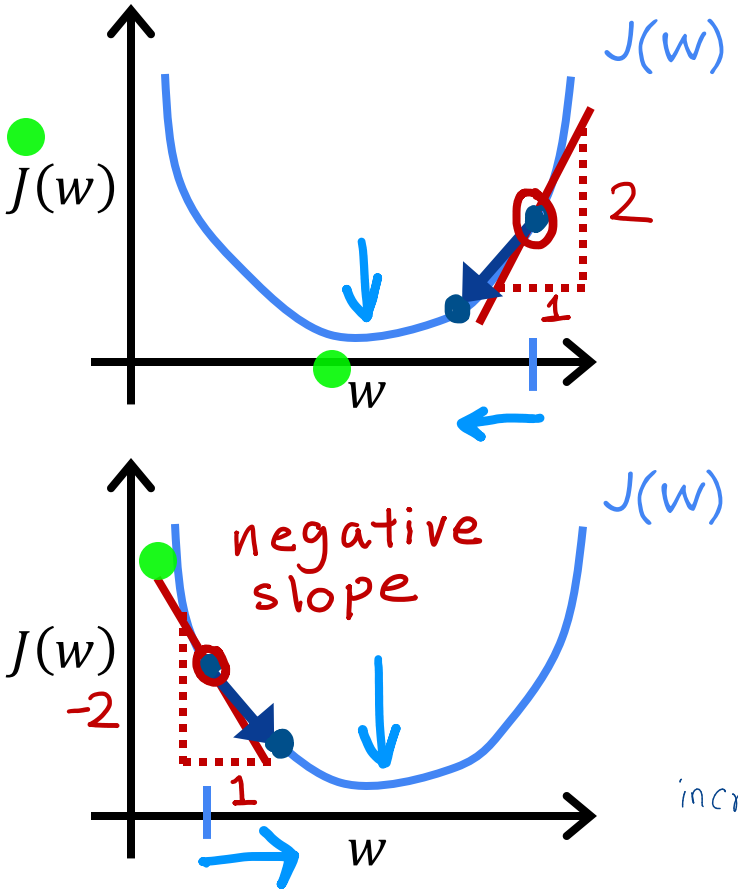
$$b = b - \alpha \frac{\partial}{\partial b} J(w, b)$$

}

$J(w)$

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

$$\min_{w} J(w)$$



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

> 0

always +

$$w = w - \alpha \cdot (\text{positive number})$$

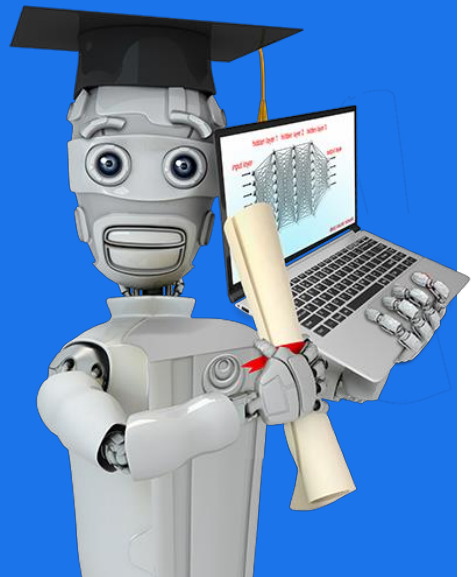
decrease w

$$\frac{d}{dw} J(w) < 0$$

increase w ← $w = w - \alpha \cdot (\text{negative number})$

Stanford
ONLINE

DeepLearning.AI



Training Linear Regression

Learning Rate

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

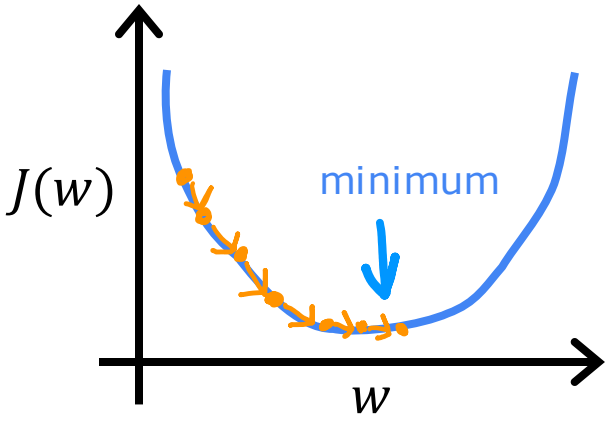
If α is too small…

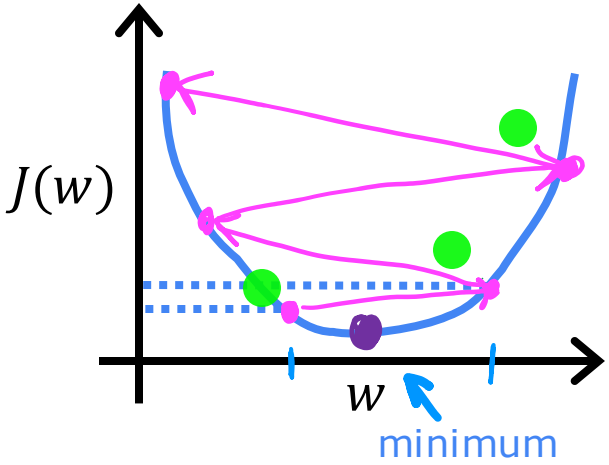
Gradient descent may be slow.

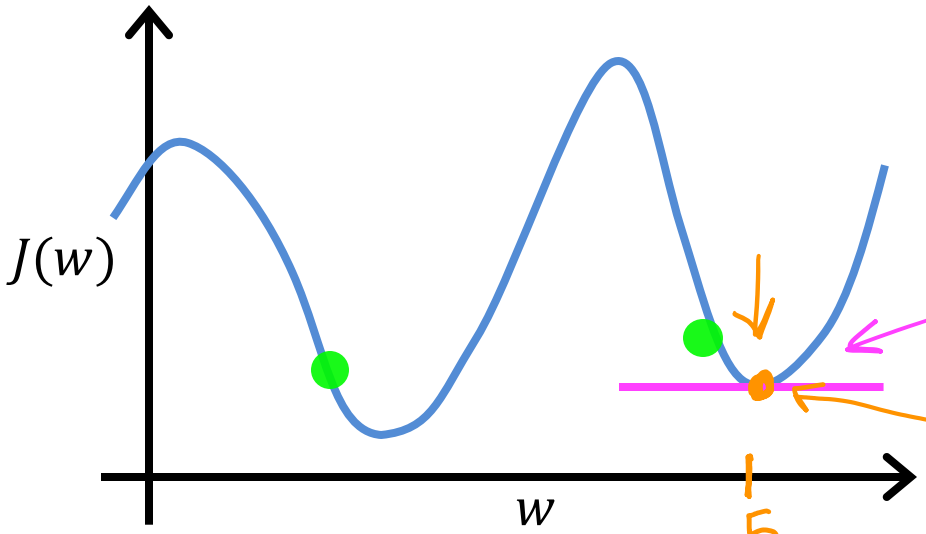
If α is too large…

Gradient descent may:

- Overshoot, never reach minimum
- Fail to converge, diverge







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

0.1

$=0$

$5 - 0.1 \cdot 0$

$w = w - \alpha \cdot 0$

$w = w$

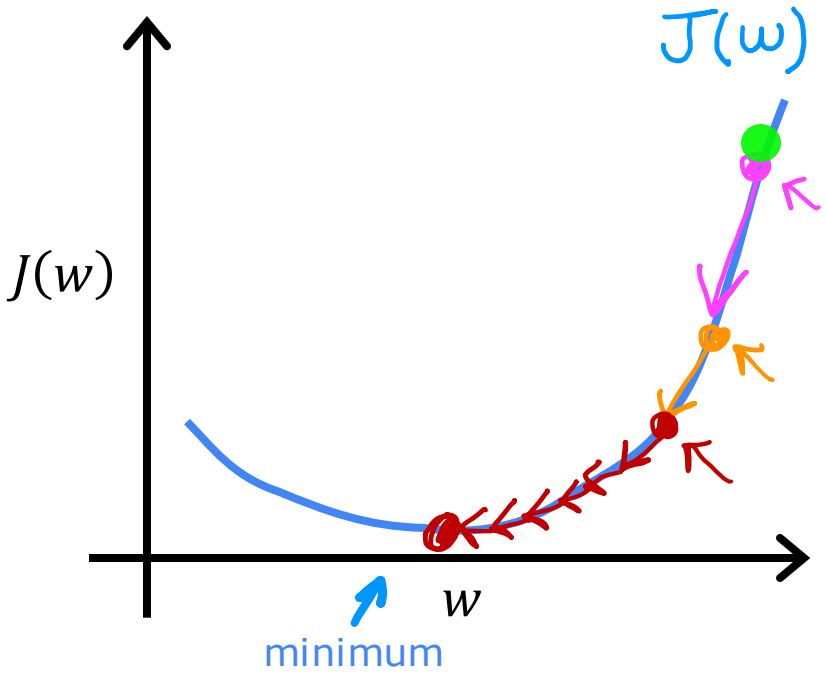
Can reach local minimum with fixed learning rate

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

smaller

not as large

large

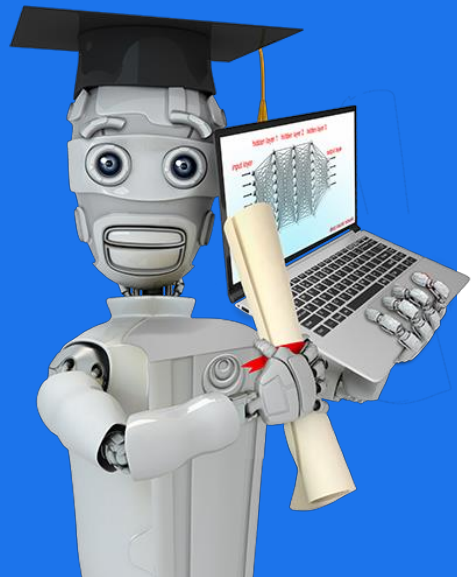


Near a local minimum,
- Derivative becomes smaller
- Update steps become smaller

Can reach minimum without decreasing learning rate α

Stanford
ONLINE

DeepLearning.AI



Training Linear Regression

Gradient Descent for Linear Regression

Linear regression model

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

Cost function

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

Gradient descent algorithm

repeat until convergence {

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

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

}

next slide is optional!

(Optional)

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

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

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

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

no $x^{(i)}$

Gradient descent algorithm

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

repeat until convergence {

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

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

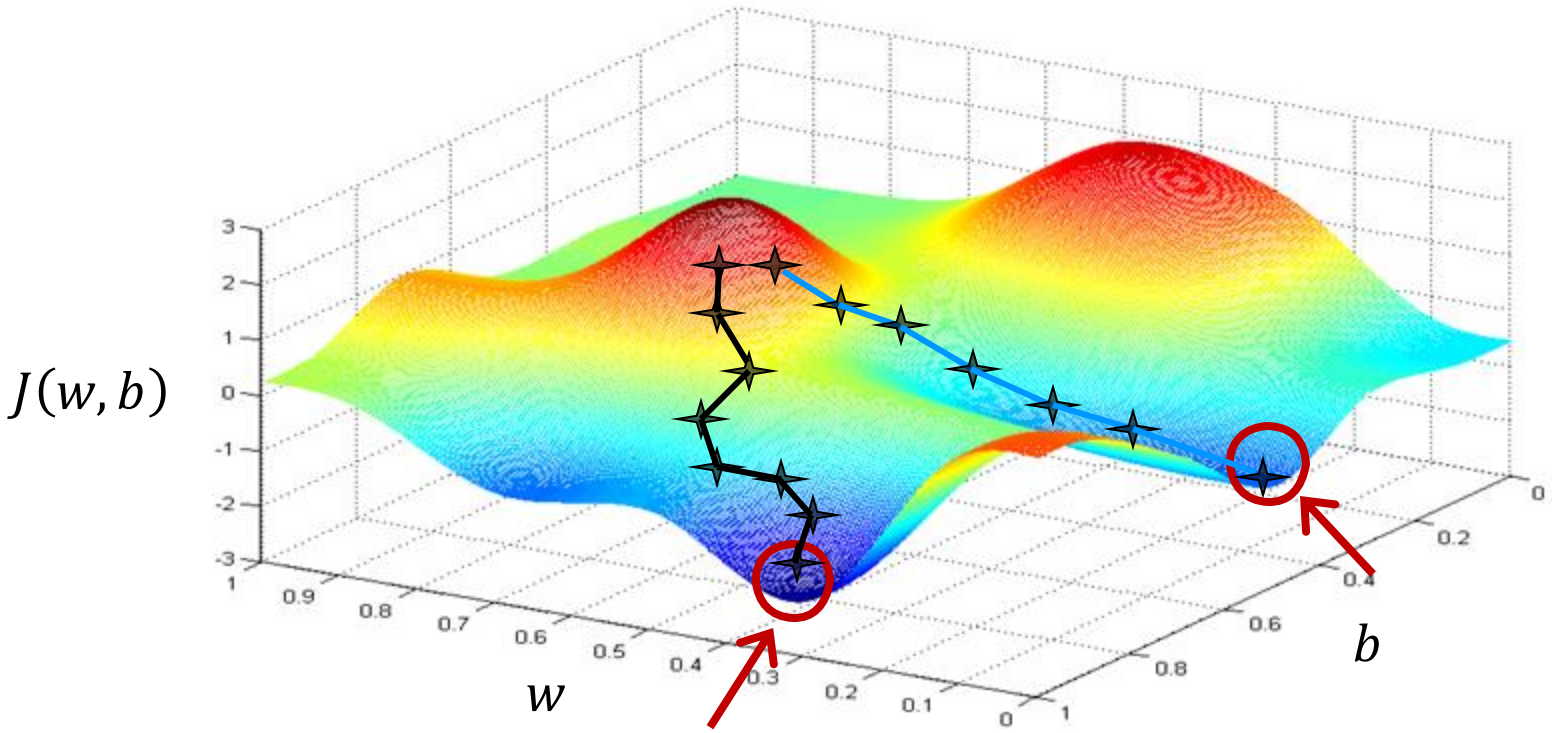
}

Update w and b simultaneously

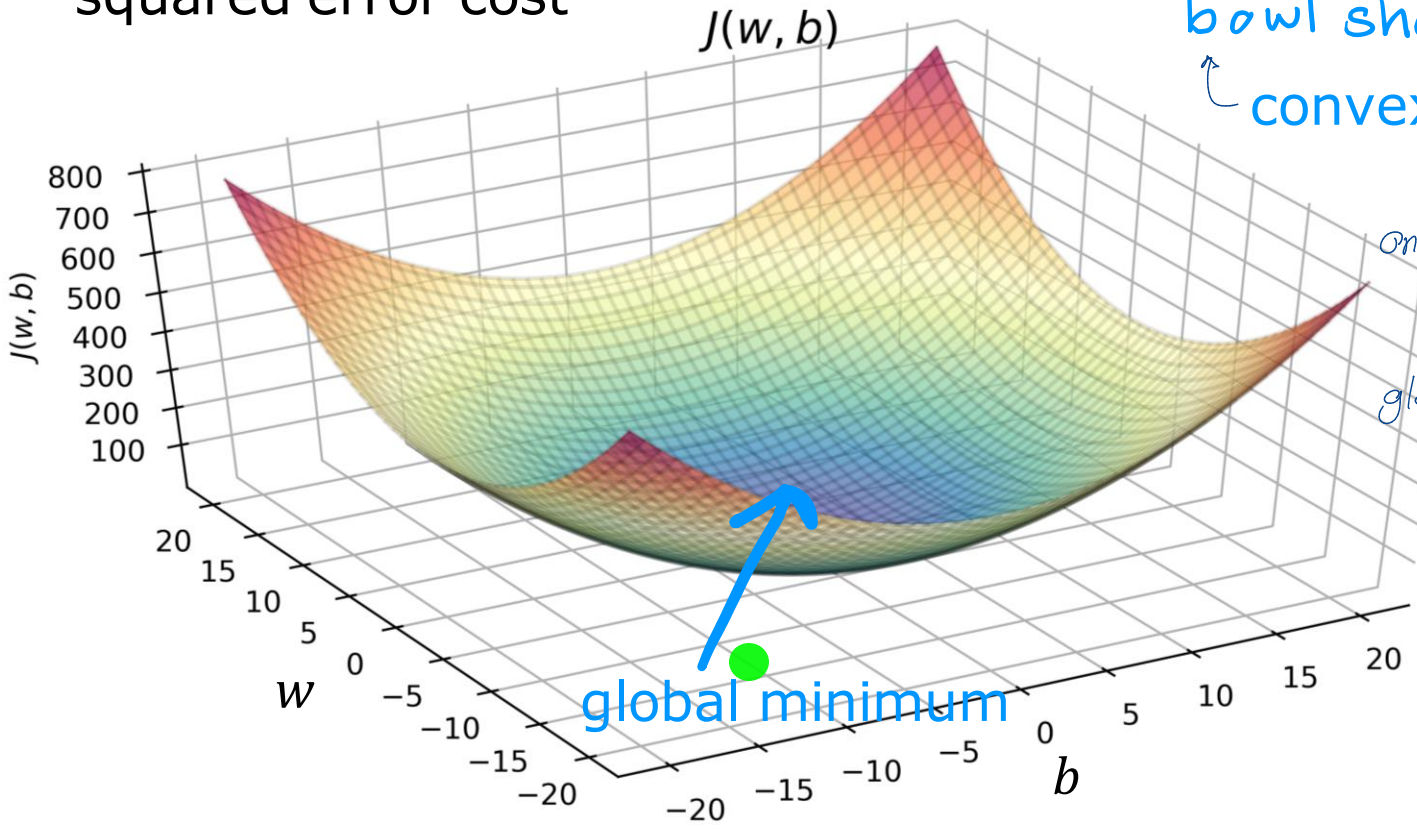
$\frac{\partial}{\partial b} J(w,b)$

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

More than one local minimum



squared error cost



bowl shape

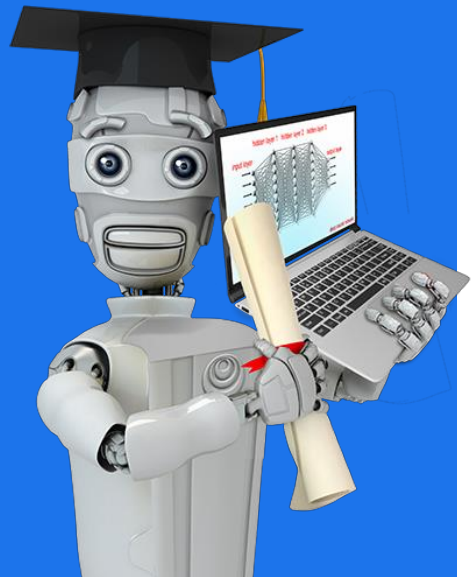
convex function

one single min

global min

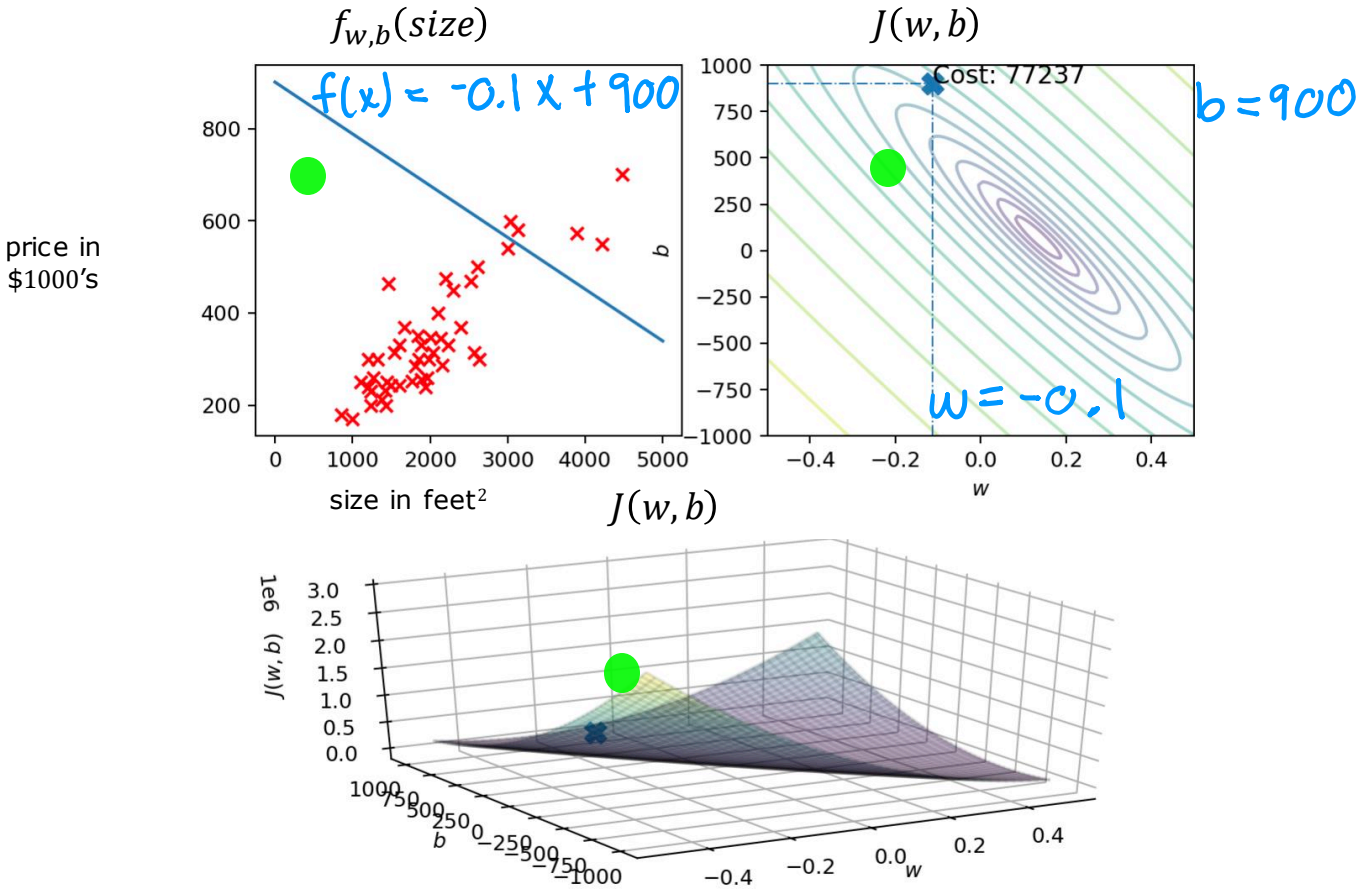
Stanford
ONLINE

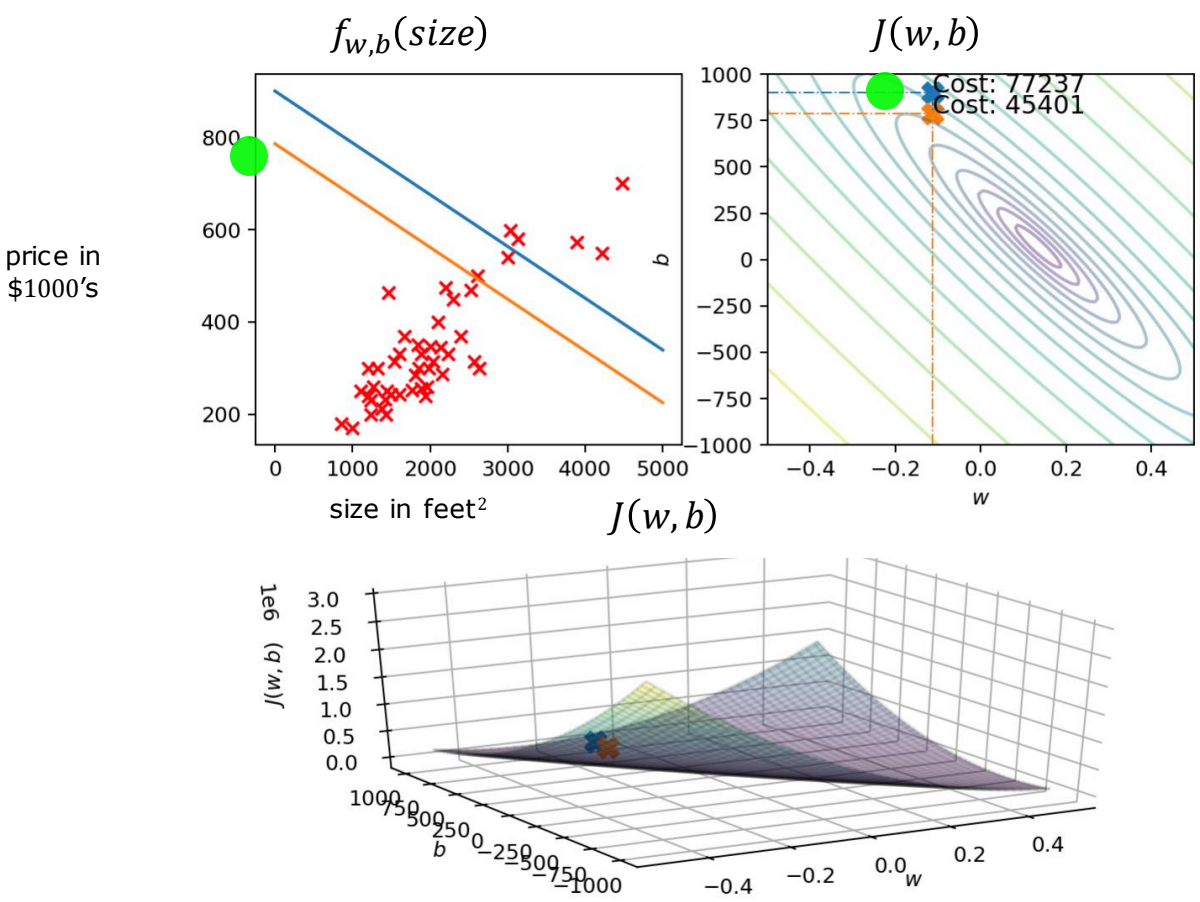
DeepLearning.AI

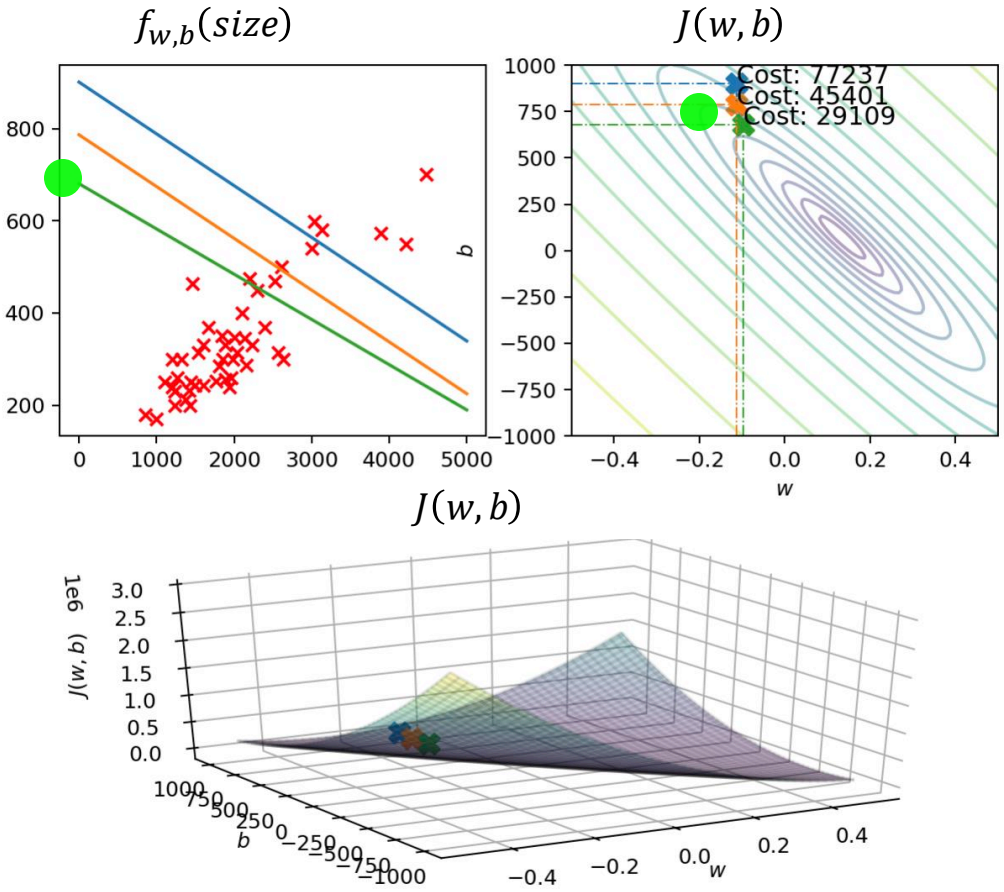


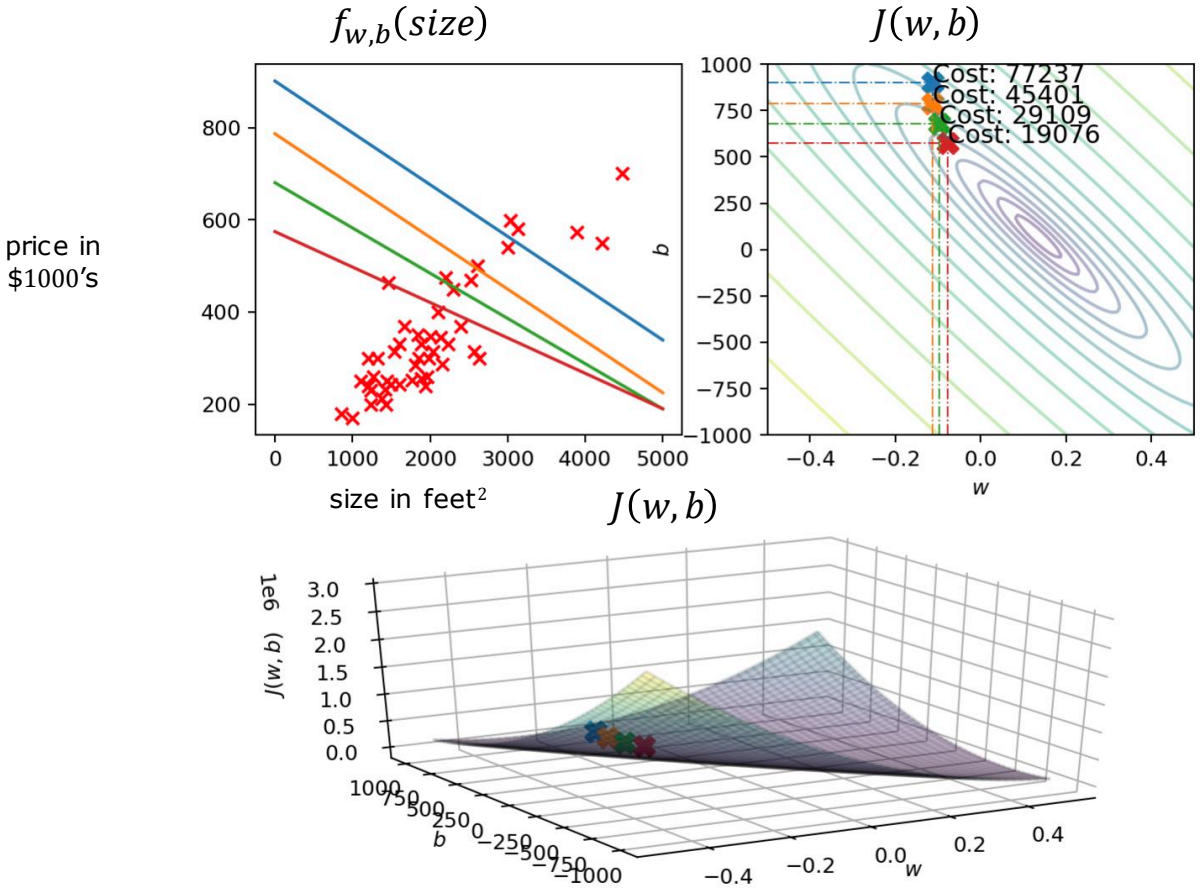
Training Linear Regression

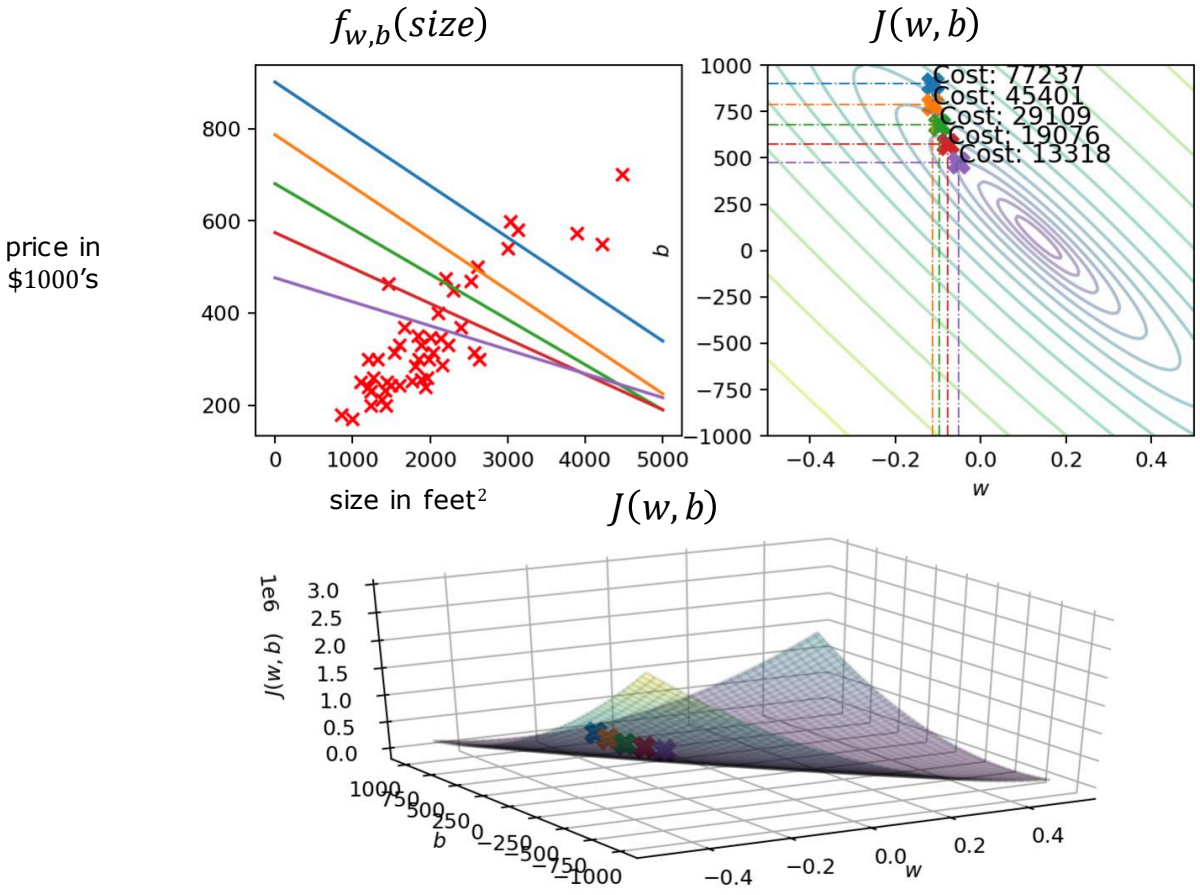
Running Gradient Descent

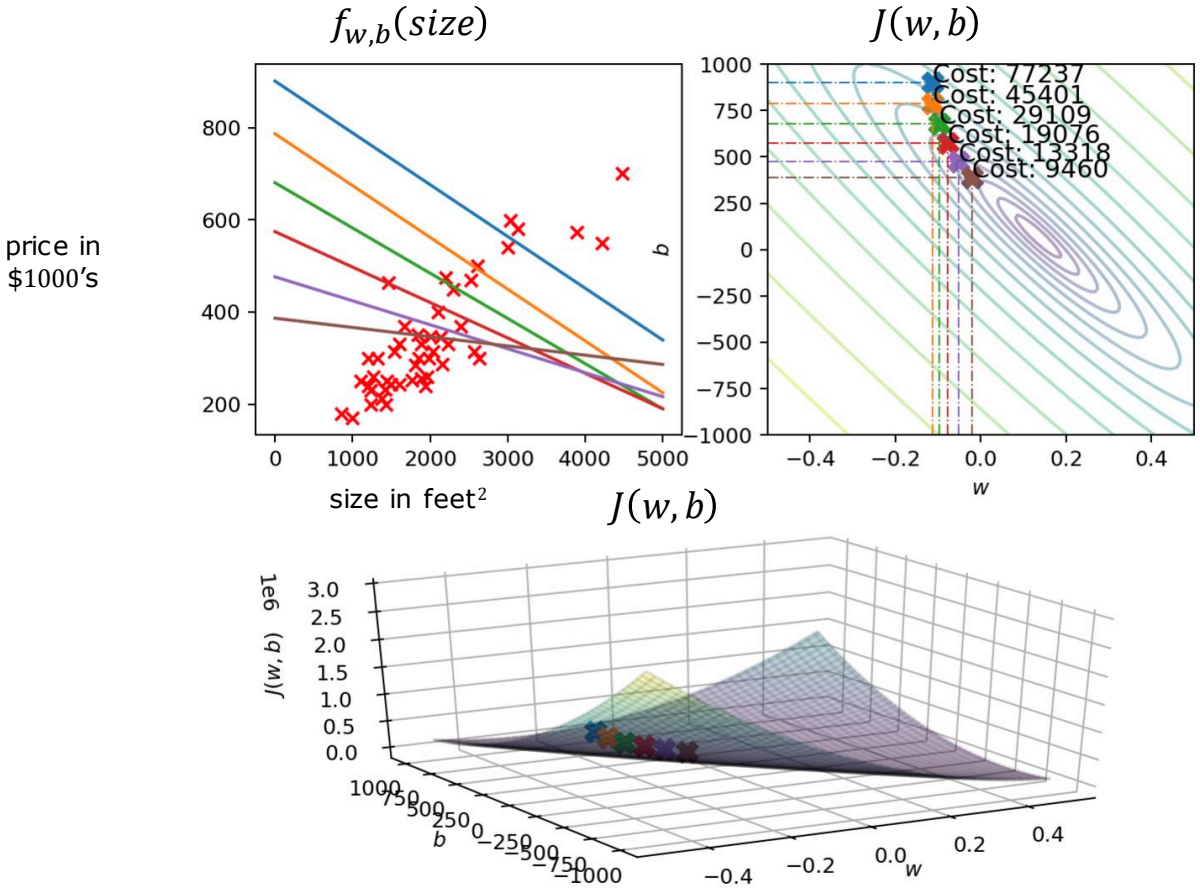

$f_{w,b}(size)$
$J(w,b)$
f(x) = -0.1 x + 900
Cost: 77237
b = 900
w = -0.1
price in $1000's
size in feet²
b
w
$J(w,b)$
$J(w,b)$
3.0
2.5
2.0
1.5
1.0
0.5
0.0
1000
750
500
250
0
−250
−500
−750
−1000
−0.4
−0.2
0.0
0.2
0.4

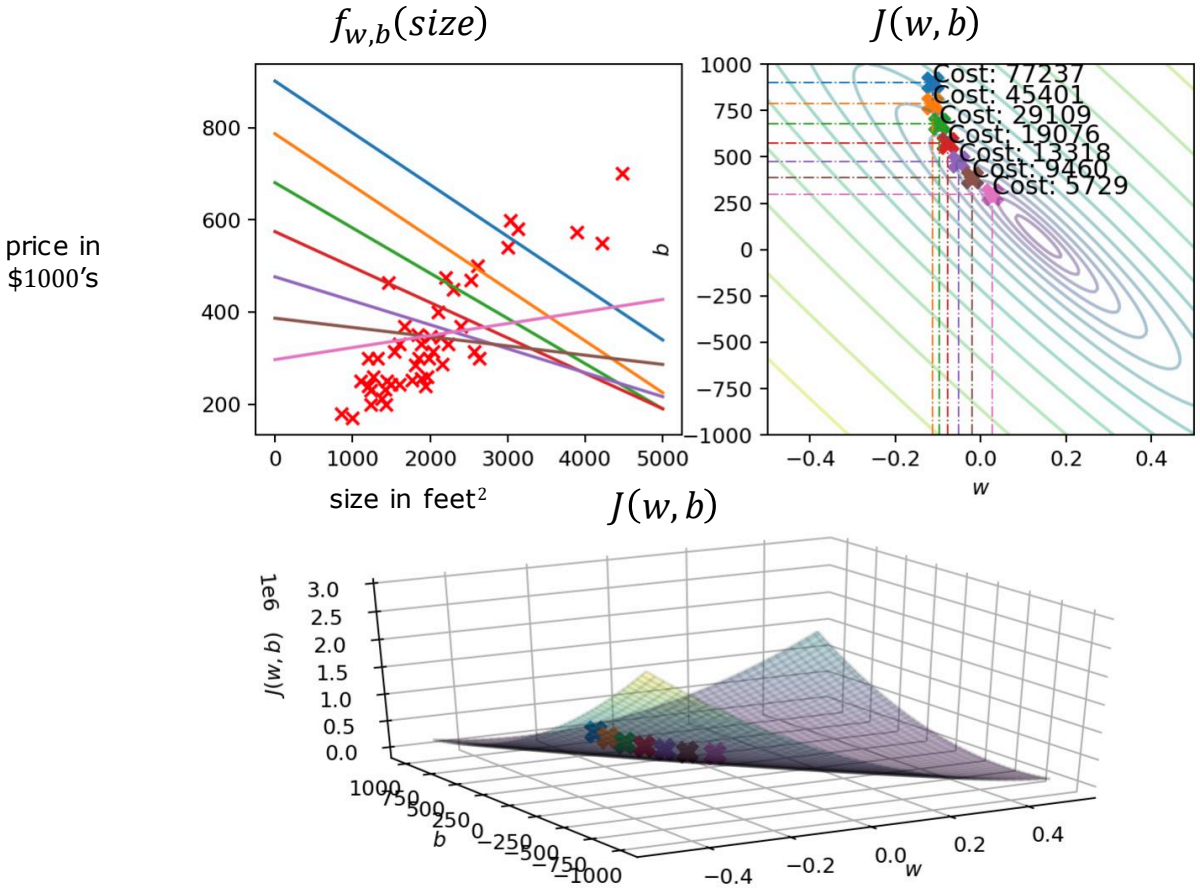

$f_{w,b}(size)$
price in $1000's
size in feet²
$J(w,b)$
Cost: 77237
Cost: 45401
$J(w,b)$

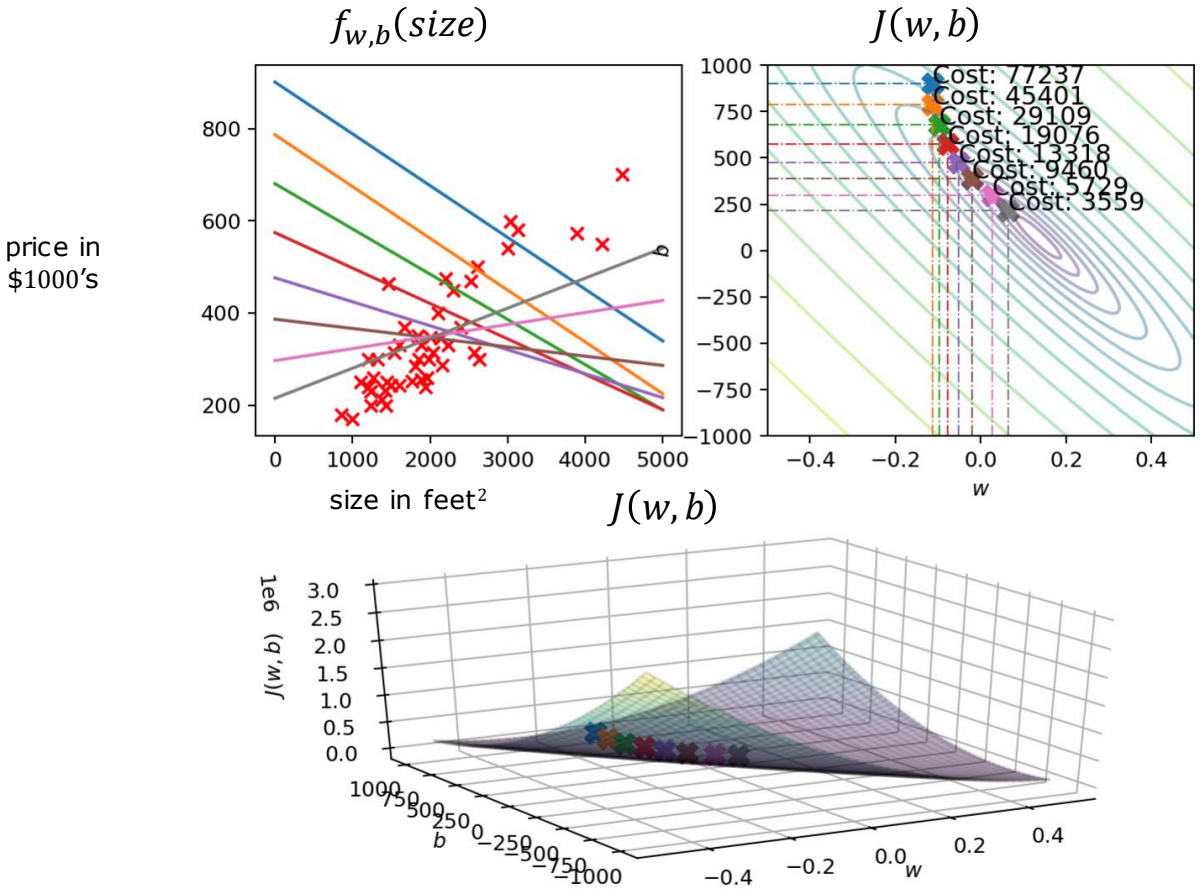

$f_{w,b}(size)$
800
600
400
200
0
1000
2000
3000
4000
5000
b
$J(w,b)$
1000
750
500
250
0
−250
−500
−750
−1000
Cost: 77237
Cost: 45401
Cost: 29109
−0.4
−0.2
0.0
0.2
0.4
w
$J(w,b)$
log
3.0
2.5
2.0
1.5
1.0
0.5
0.0
$J(w,b)$
1000
750
500
250
0
−250
−500
−750
−1000
b
−0.4
−0.2
0.0
0.2
0.4
w

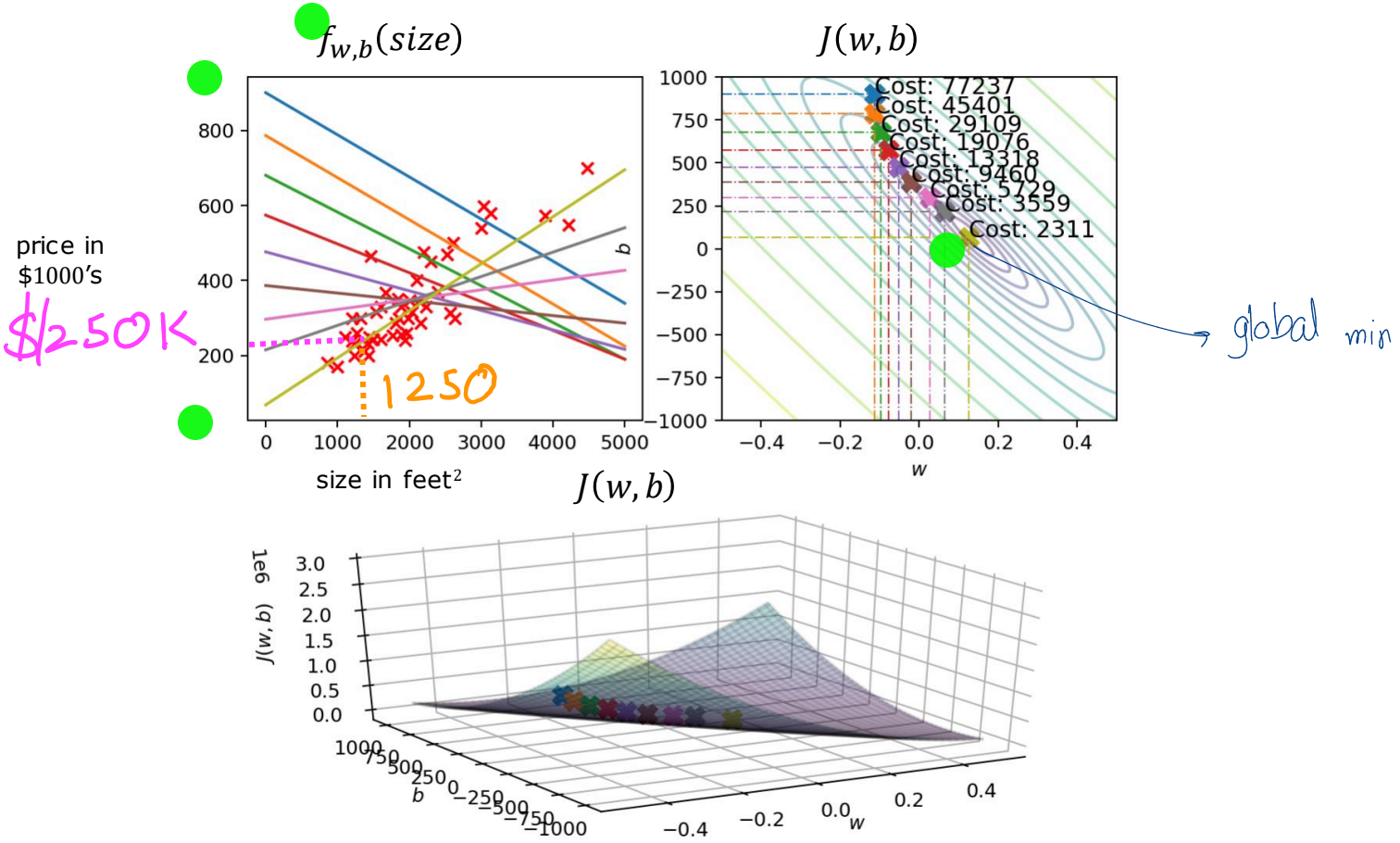

$f_{w,b}(size)$
price in $1000's
size in feet²
$J(w,b)$
Cost: 77237
Cost: 45401
Cost: 29109
Cost: 19076
$J(w,b)$


$f_{w,b}(size)$
price in $1000's
size in feet^2
$J(w,b)$
Cost: 77237
Cost: 45401
Cost: 29109
Cost: 19076
Cost: 13318
w
b
$J(w,b)$
log
3.0
2.5
2.0
1.5
1.0
0.5
0.0


$f_{w,b}(size)$
price in $1000's
size in feet^2
$J(w,b)$
Cost: 77237
Cost: 45401
Cost: 29109
Cost: 19076
Cost: 13318
Cost: 9460
$J(w,b)$


$f_{w,b}(size)$
price in $1000's
size in feet²
$J(w,b)$
Cost: 77237
Cost: 45401
Cost: 29109
Cost: 19076
Cost: 13318
Cost: 9460
Cost: 5729
w
b
$J(w,b)$
log
$J(w,b)$


$f_{w,b}(size)$
price in $1000's
size in feet^2
$J(w,b)$
Cost: 77237
Cost: 45401
Cost: 29109
Cost: 19076
Cost: 13318
Cost: 9460
Cost: 5729
Cost: 3559
w
b
$J(w,b)$


$f_{w,b}(size)$
price in $1000's
$250K
1250
size in feet²
$J(w,b)$
Cost: 77237
Cost: 45401
Cost: 29109
Cost: 19076
Cost: 13318
Cost: 9460
Cost: 5729
Cost: 3559
Cost: 2311
global min
$J(w,b)$
log
$J(w,b)$
b
w

"Batch" gradient descent

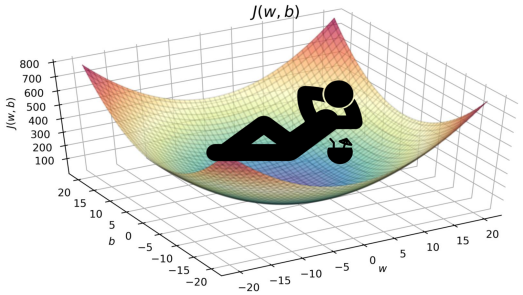
"Batch": Each step of gradient descent uses all the training examples.

other gradient descent: subsets

	x size in feet^2	y price in \$1000's
(1)	2104	400
(2)	1416	232
(3)	1534	315
(4)	852	178
...	...	...
(47)	3210	870

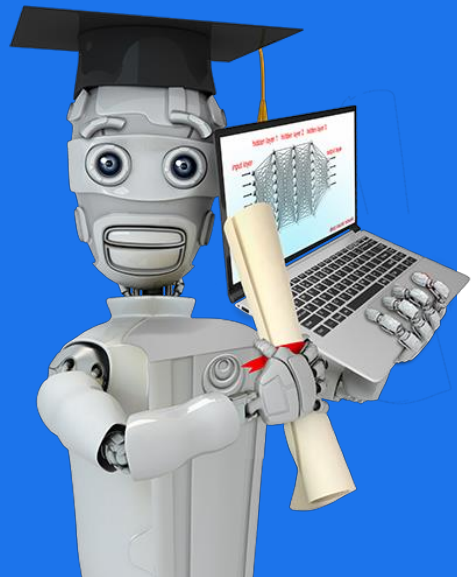
$m = 47$

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



Stanford
ONLINE

DeepLearning.AI



Linear Regression with Multiple Variables

Week 2

Multiple Features

Multiple features (variables)

one feature →

Size in feet^2 (x)	Price ($) in 1000's (y) ←
2104	400
1416	232
1534	315
852	178
…	…

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

Multiple features (variables)

Size in feet²	Number of bedrooms	Number of floors	Age of home in years	Price ($) in $1000's
x_1	x_2	x_3	x_4	
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
...	...	...	...	...

$j = 1 \ldots 4$

$n = 4$

$i = 2$

x_j = j^{th} feature

n = number of features

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

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

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

$x_3^{(2)} = 2$

Model:

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

$$f_{w,b}(X) = w_1 X_1 + w_2 X_2 + w_3 X_3 + w_4 X_4 + b$$

example

$$f_{w,b}(X) = 0.1 X_1 + 4 X_2 + 10 X_3 + -2 X_4 + 80$$

X_1: size
X_2: #bedrooms
X_3: #floors
X_4: years
80: base price

$$f_{w,b}(x) = w_1 x_1 + w_2 x_2 + \cdots + w_n x_n + b$$

$$f_{\vec{w},b}(\vec{x}) = w_1 x_1 + w_2 x_2 + \cdots + w_n x_n + b$$

$\vec{w} = [w_1 \; w_2 \; w_3 \ldots w_n]$ parameters of the model

b is a number

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

$$f_{\vec{w},b}(\vec{x}) = \vec{w} \cdot \vec{x} + b = w_1 x_1 + w_2 x_2 + w_3 x_3 + \cdots + w_n x_n + b$$

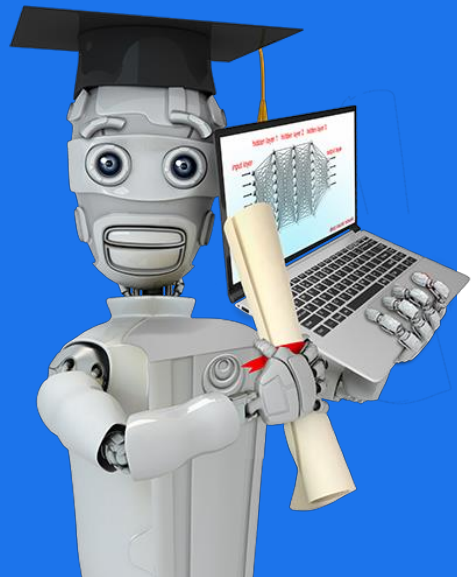
dot product

multiple linear regression

(not multivariate regression)

Stanford
ONLINE

DeepLearning.AI



Linear Regression with Multiple Variables

Vectorization Part 1

Parameters and features

$\vec{w} = [w_1 \quad w_2 \quad w_3]$ $n=3$

b is a number

$\vec{x} = [x_1 \quad x_2 \quad x_3]$

linear algebra: count from 1

NumPy

w[0] w[1] w[2]

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

x[0] x[1] x[2]

code: count from 0

Without vectorization

$n = 100{,}000$

$f_{\vec{w},b}(\vec{x}) = w_1 x_1 + w_2 x_2 + w_3 x_3 + b$

```
f = w[0] * x[0] +
    w[1] * x[1] +
    w[2] * x[2] + b
```

Without vectorization

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

$\sum_{j=1}^{n} \rightarrow j = 1 \ldots n$ 1, 2, 3

range(0, n) → j = 0 ... n-1

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

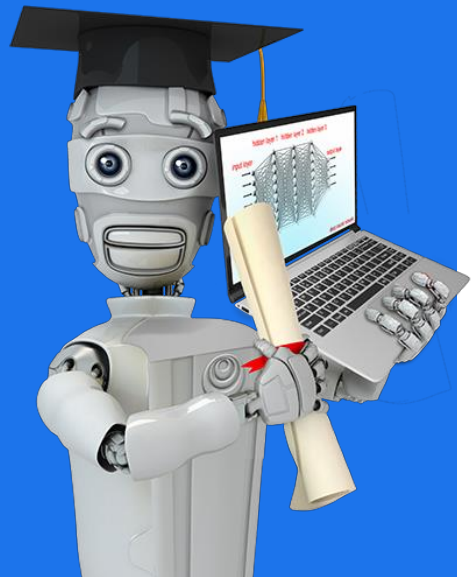
Vectorization

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

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

Stanford
ONLINE

DeepLearning.AI



Linear Regression with Multiple Variables

Vectorization Part 2

Without vectorization

```
for j in range(0,16):
    f = f + w[j] * x[j]
```

t_0
```
f + w[0] * x[0]
```

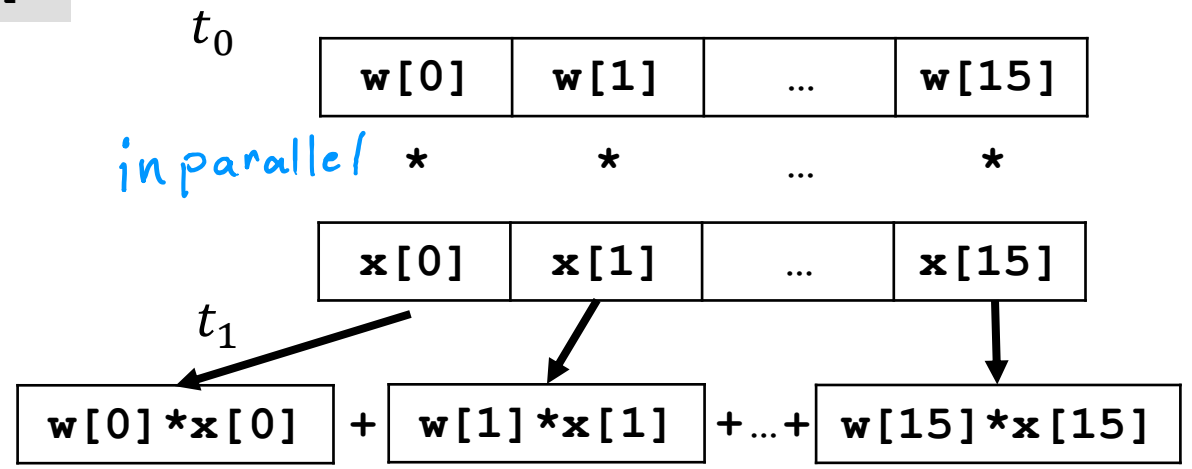
t_1
```
f + w[1] * x[1]
```

...

t_{15}
```
f + w[15] * x[15]
```

Vectorization

```
np.dot(w,x)
```



efficient → scale to large datasets

Gradient descent

$\vec{w} = (w_1 \quad w_2 \quad \cdots \quad w_{16})$ ~~b~~ ignore b parameters

derivatives $\vec{d} = (d_1 \quad d_2 \quad \cdots \quad d_{16})$

```
w = np.array([0.5, 1.3, … 3.4])
d = np.array([0.3, 0.2, … 0.4])
```

learning rate α

compute $w_j = w_j - 0.1 d_j$ for $j = 1 \ldots 16$

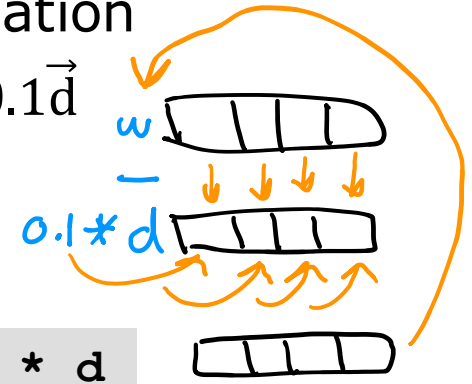
Without vectorization

$$w_1 = w_1 - 0.1 d_1$$
$$w_2 = w_2 - 0.1 d_2$$
$$\vdots$$
$$w_{16} = w_{16} - 0.1 d_{16}$$

```
for j in range(0,16):
    w[j] = w[j] - 0.1 * d[j]
```

With vectorization

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

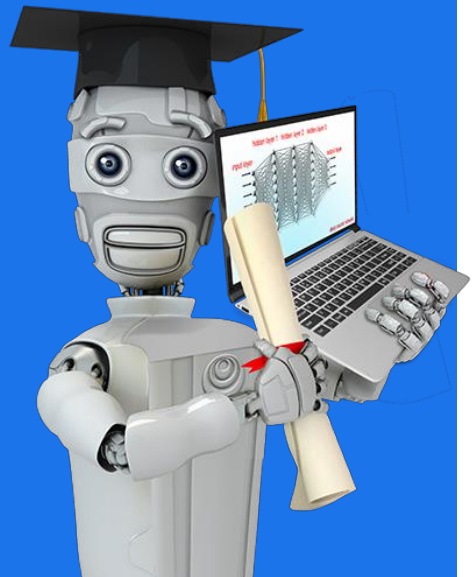


in code :

```
w = w - 0.1 * d
```

Stanford
ONLINE

DeepLearning.AI



Linear Regression with Multiple Variables

Gradient Descent for Multiple Regression

Previous notation

Parameters $w_1, \cdots, w_n$

b

Model $f_{\vec{w},b}(\vec{x}) = w_1 x_1 + \cdots + w_n x_n + b$

Cost function $J(w_1, \cdots, w_n, b)$

Vector notation

vector of length n

$\vec{w} = [w_1 \quad \cdots \quad w_n]$

b still a number

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

dot product

$J(\vec{w}, b)$

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 {

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

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

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

simultaneously update w, b

}

n features ($n \geq 2$)

repeat {

$j = 1$

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

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

$\vdots$

$j = n$

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

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

simultaneously update

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

}

An alternative to gradient descent

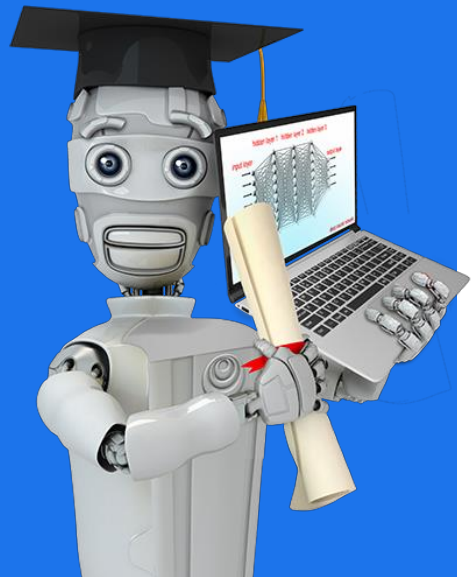
Normal equation
- Only for linear regression
- Solve for w, b without iterations

Disadvantages
- Doesn't generalize to other learning algorithms.
- Slow when number of features is large (> 10,000)

What you need to know
- Normal equation method may be used in machine learning libraries that implement linear regression.
- Gradient descent is the recommended method for finding parameters w,b

Stanford
ONLINE

DeepLearning.AI



Practical Tips for Linear Regression

enable gradient descend to run much faster

Feature Scaling Part 1

Feature and parameter values

$\widehat{price} = w_1 x_1 + w_2 x_2 + b$

size #bedrooms

x_1: size (feet^2)
range: $300 - 2{,}000$

large

x_2: # bedrooms
range: $0 - 5$

small

House: $x_1 = 2000$, $x_2 = 5$, $price = \$500\text{k}$

one training example

size of the parameters w_1, w_2?

$w_1 = 50, \quad w_2 = 0.1, \quad b = 50$

$\widehat{price} = \underbrace{50 * 2000}_{100{,}000\text{k}} + \underbrace{0.1 * 5}_{0.5\text{k}} + \underbrace{50}_{50\text{k}}$

$\widehat{price} = \$100{,}050.5\text{k} = \$100{,}050{,}500$

not good

$w_1 = 0.1, \quad w_2 = 50, \quad b = 50$

small large

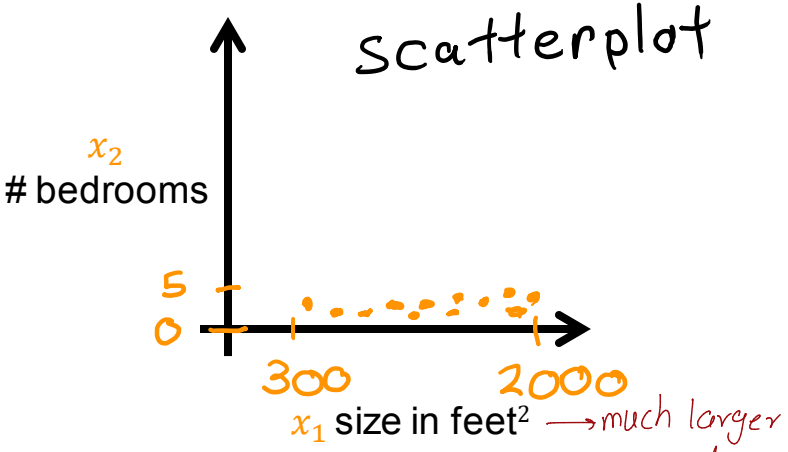
$\widehat{price} = \underbrace{0.1 * 2000k}_{200\text{k}} + \underbrace{50 * 5}_{250\text{k}} + \underbrace{50}_{50\text{k}}$

$\widehat{price} = \$500\text{k}$ more reasonable

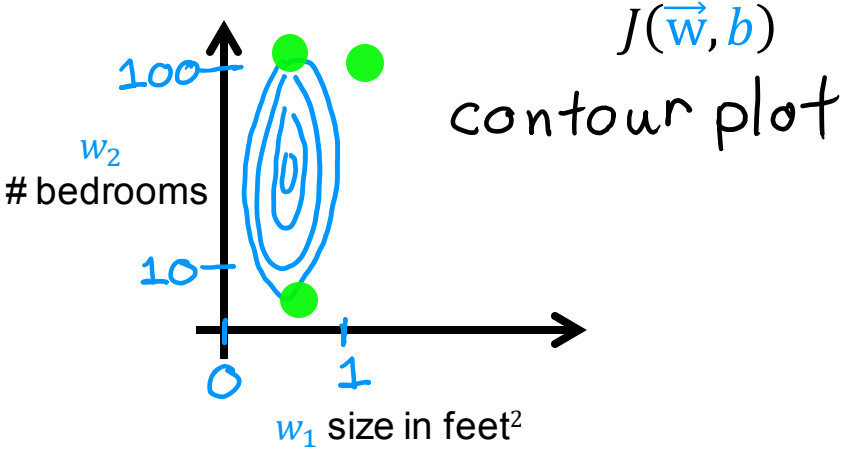
Feature size and parameter size

	size of feature x_j	size of parameter w_j
size in feet^2	⟷	⟷
#bedrooms	⟷	⟷

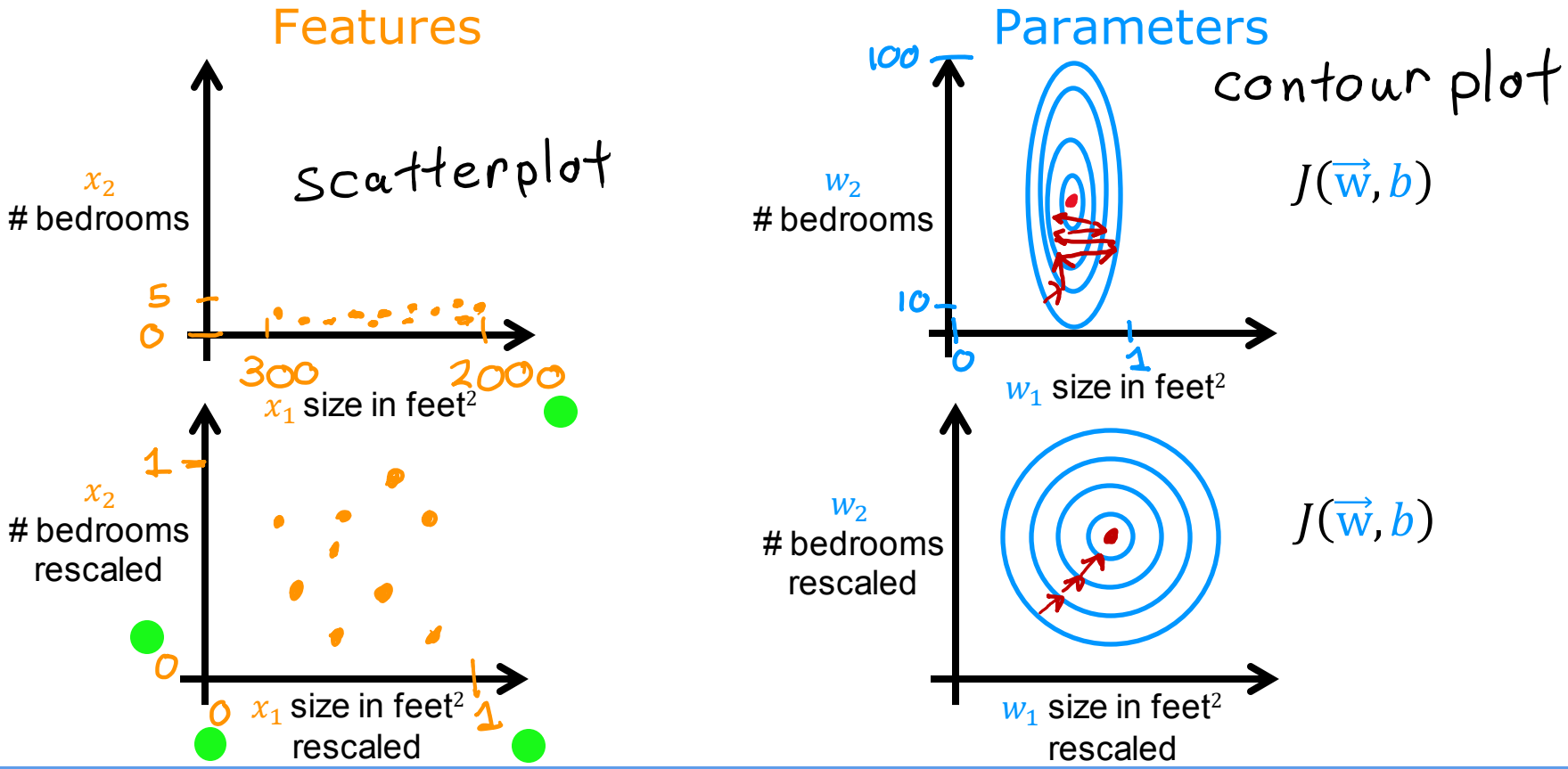
Features



Parameters

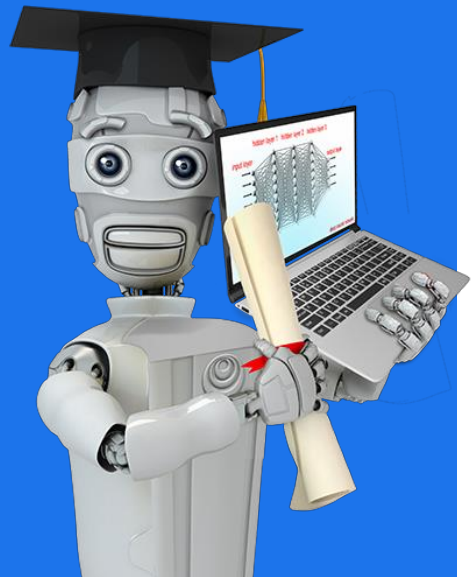


Feature size and gradient descent



Stanford
ONLINE

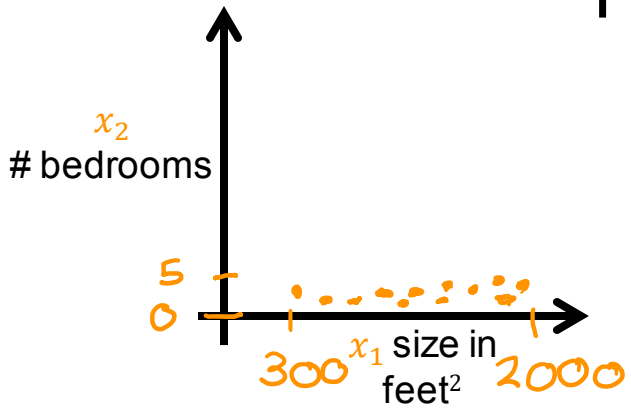
DeepLearning.AI

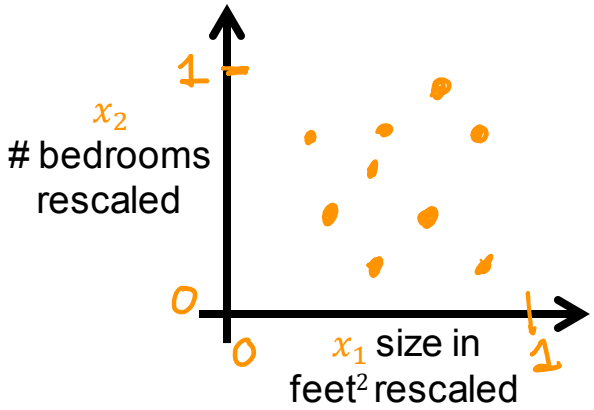


Practical Tips for Linear Regression

Feature Scaling Part 2

Feature scaling





$300 \leq x_1 \leq 2000$

$x_{1,scaled} = \frac{x_1}{2000}$ max

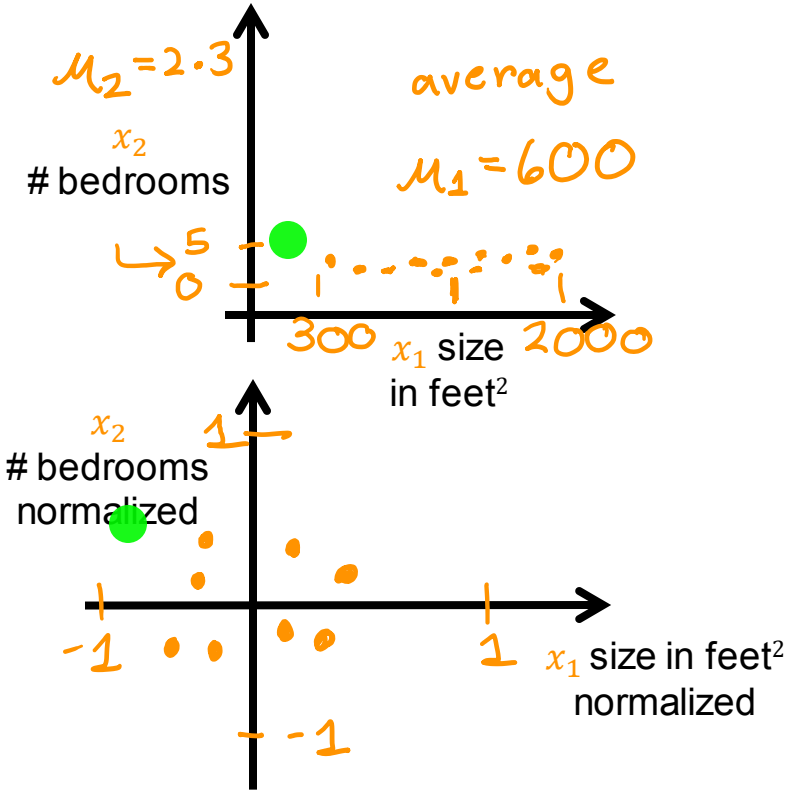
$0.15 \leq x_{1,scaled} \leq 1$

$0 \leq x_2 \leq 5$

$x_{2,scaled} = \frac{x_2}{5}$ max

$0 \leq x_{2,scaled} \leq 1$

Mean normalization



$$300 \le x_1 \le 2000$$

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

max-min

$$-0.18 \le x_1 \le 0.82$$

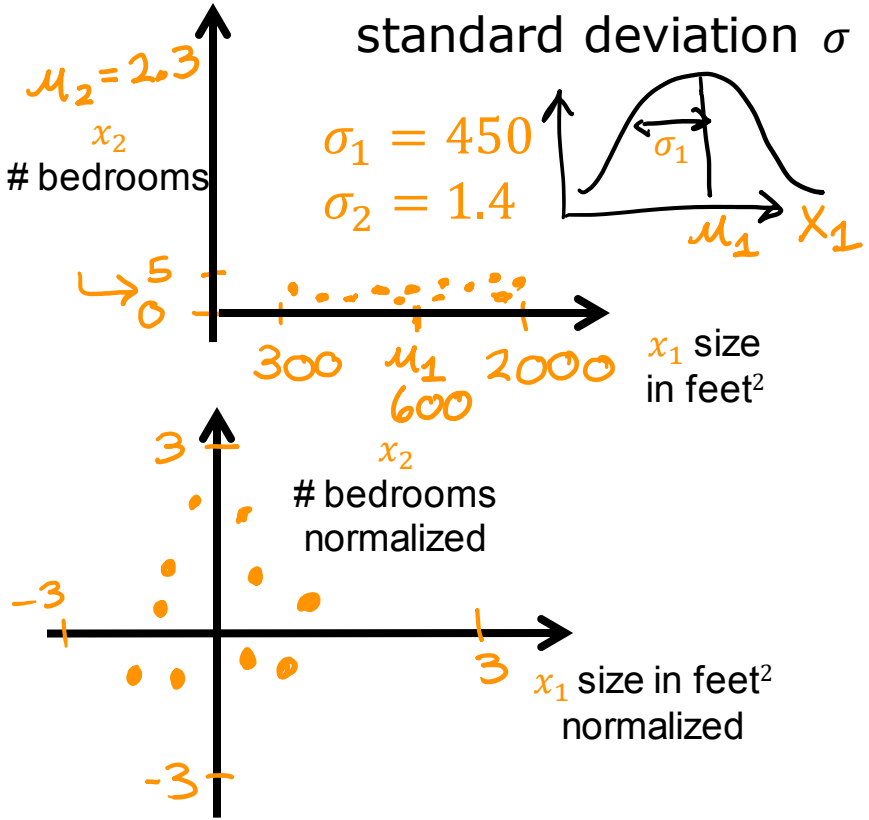
$$0 \le x_2 \le 5$$

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

max-min

$$-0.46 \le x_2 \le 0.54$$

Z-score normalization



$300 \le x_1 \le 2000$ $\quad$ $0 \le x_2 \le 5$

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

$-0.67 \le x_1 \le 3.1$ $\quad$ $-1.6 \le x_2 \le 1.9$

Feature scaling

aim for about $-1 \leq x_j \leq 1$ for each feature x_j

$-3 \leq x_j \leq 3$

$-0.3 \leq x_j \leq 0.3$

} acceptable ranges

$0 \leq x_1 \leq 3$ okay, no rescaling

$-2 \leq x_2 \leq 0.5$ okay, no rescaling

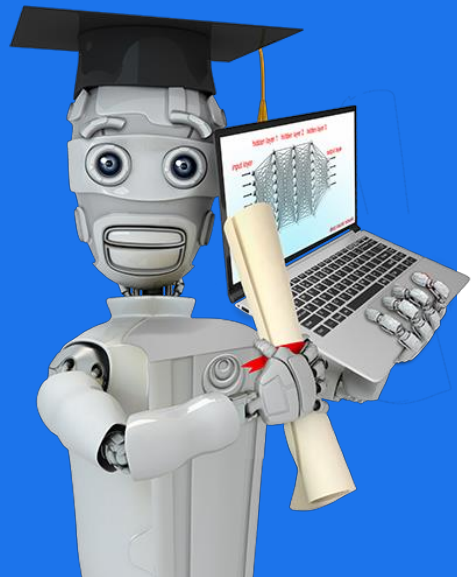
$-100 \leq x_3 \leq 100$ too large → rescale

$-0.001 \leq x_4 \leq 0.001$ too small → rescale

$98.6 \leq x_5 \leq 105$ too large → rescale

Stanford
ONLINE

DeepLearning.AI



Practical Tips for Linear Regression

Checking Gradient Descent for Convergence

Gradient descent

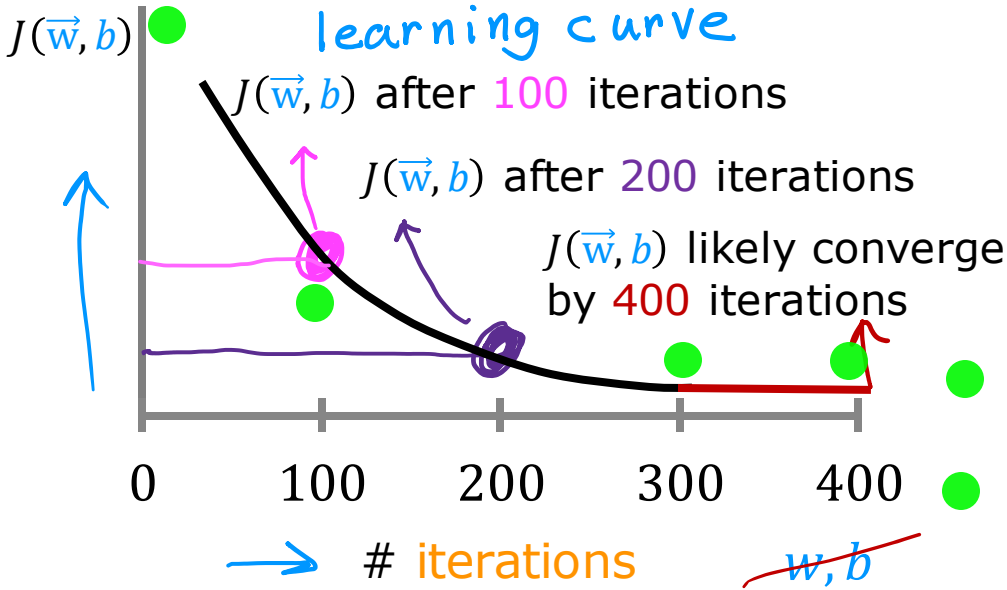
$$w_j = w_j - \alpha \frac{\partial}{\partial w_j} J(\vec{w}, b)$$

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

Make sure gradient descent is working correctly

objective: $\min_{\vec{w},b} J(\vec{w},b)$

$J(\vec{w},b)$ should decrease after every iteration



iterations needed varies 30 1,000 100,000

Automatic convergence test

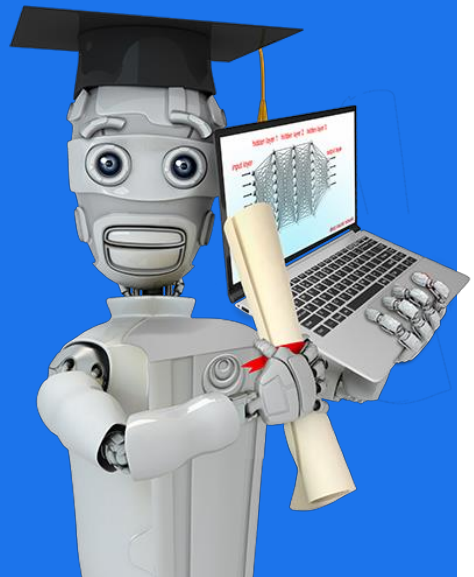
Let ε "epsilon" be 10^{-3}. 0.001

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

(found parameters $\vec{w}, b$ to get close to global minimum)

Stanford
ONLINE

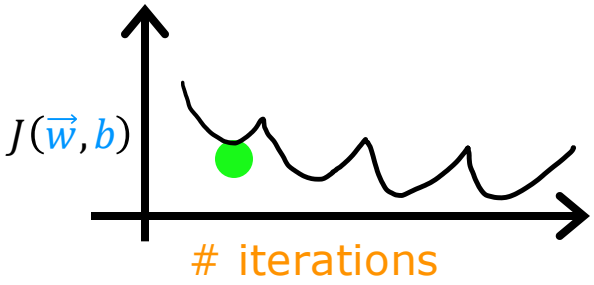
DeepLearning.AI

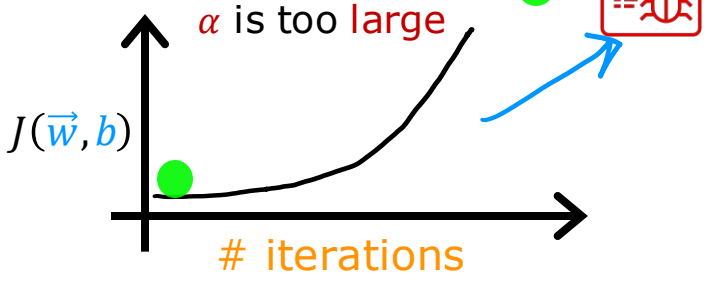


Practical Tips for Linear Regression

Choosing the Learning Rate

Identify problem with gradient descent





or learning rate is too large

$w_1 = w_1 + \alpha d_1$ (increase)

use a minus sign

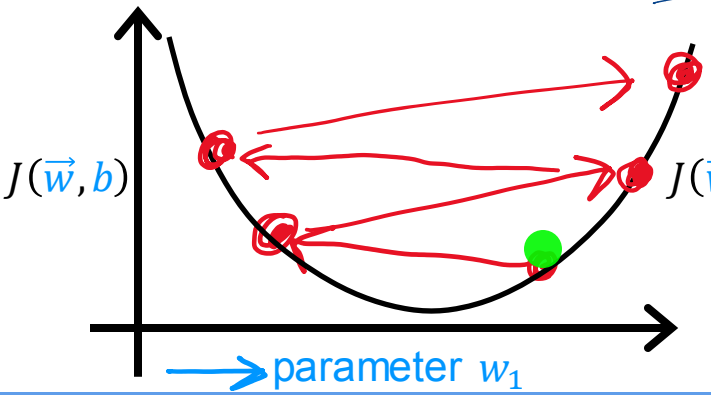
$w_1 = w_1 - \alpha d_1$

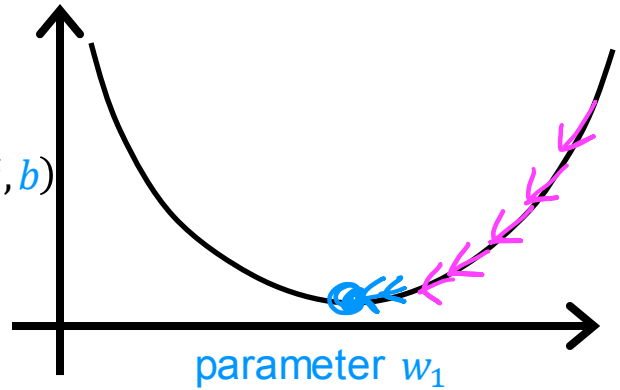
Adjust learning rate

α is too big

to fix

Use smaller α





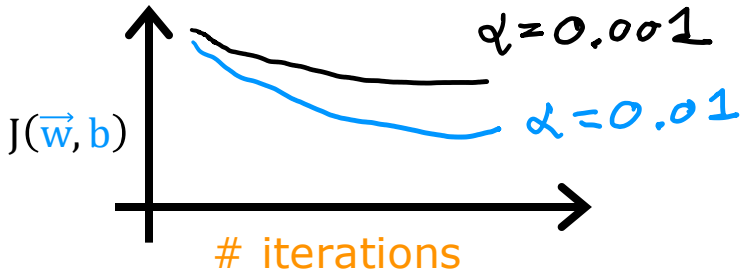
With a small enough α, $J(\vec{w}, b)$ should decrease on every iteration

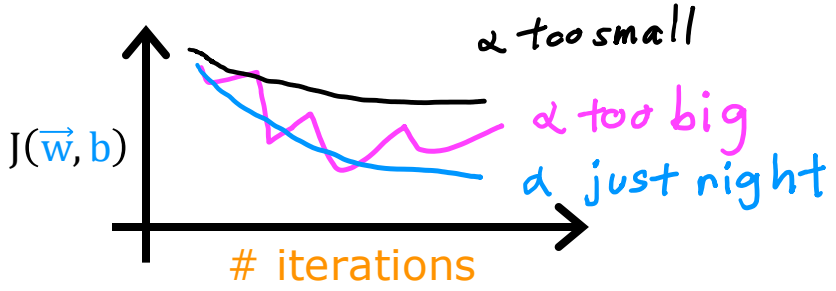
If α is too small, gradient descent takes a lot more iterations to converge

Values of α to try:

... 0.001 0.003 0.01 0.03 0.1 0.3 1 ...

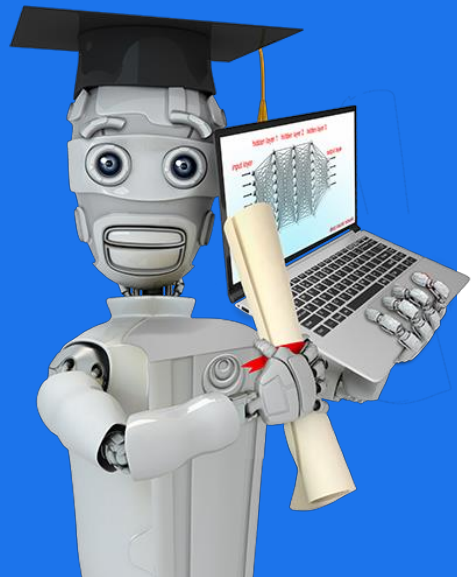
3x ≈3x 3x ≈3x 3x ≈3x





Stanford
ONLINE

DeepLearning.AI



Practical Tips for Linear Regression

Feature Engineering

Feature engineering

$$f_{\vec{w},b}(\vec{x}) = w_1 x_1 + w_2 x_2 + b$$

frontage depth

$$area = frontage \times depth$$

$x_3 = x_1 x_2$

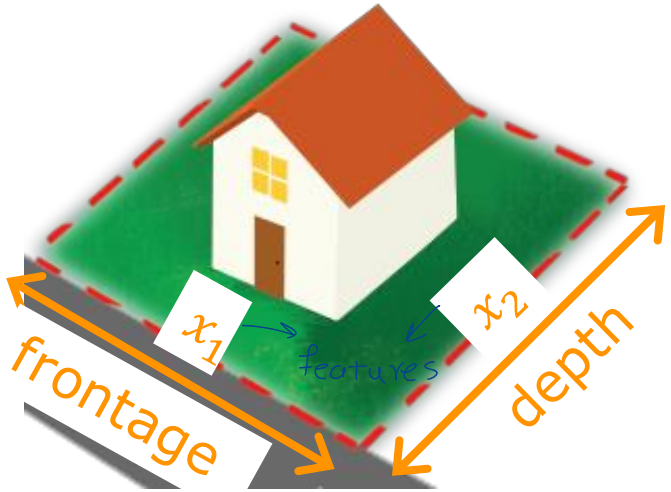
new feature

Creating new feature

↓

Feature Engineering

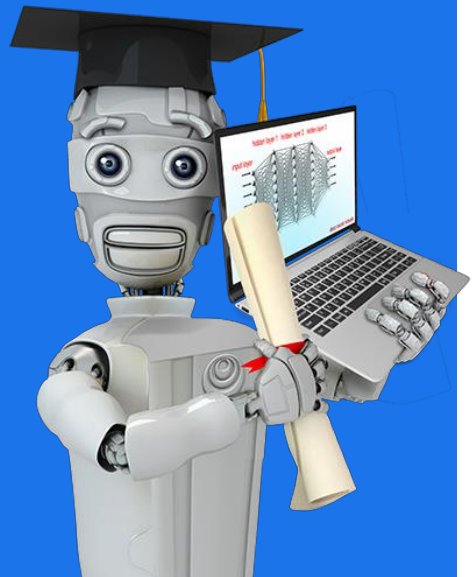
$$f_{\vec{w},b}(\vec{x}) = w_1 x_1 + w_2 x_2 + w_3 x_3 + b$$



Feature engineering:
Using intuition to design new features, by transforming or combining original features.

Stanford
ONLINE

DeepLearning.AI



Practical Tips for Linear Regression

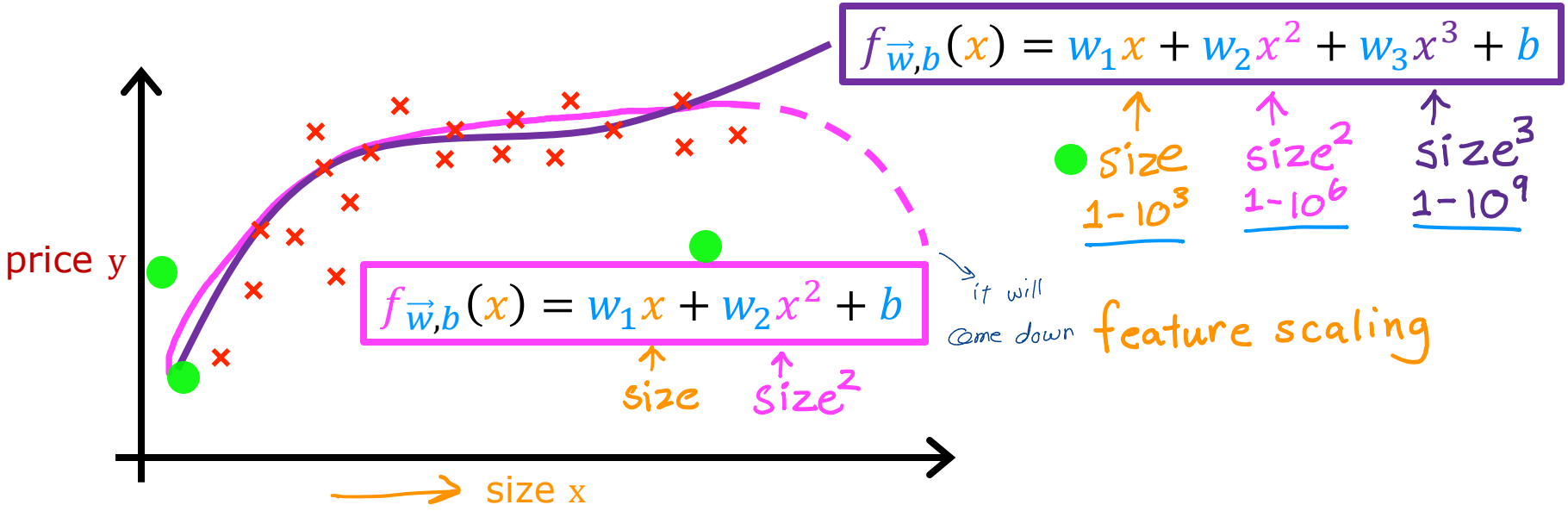
multiple linear regression + feature engineering

=

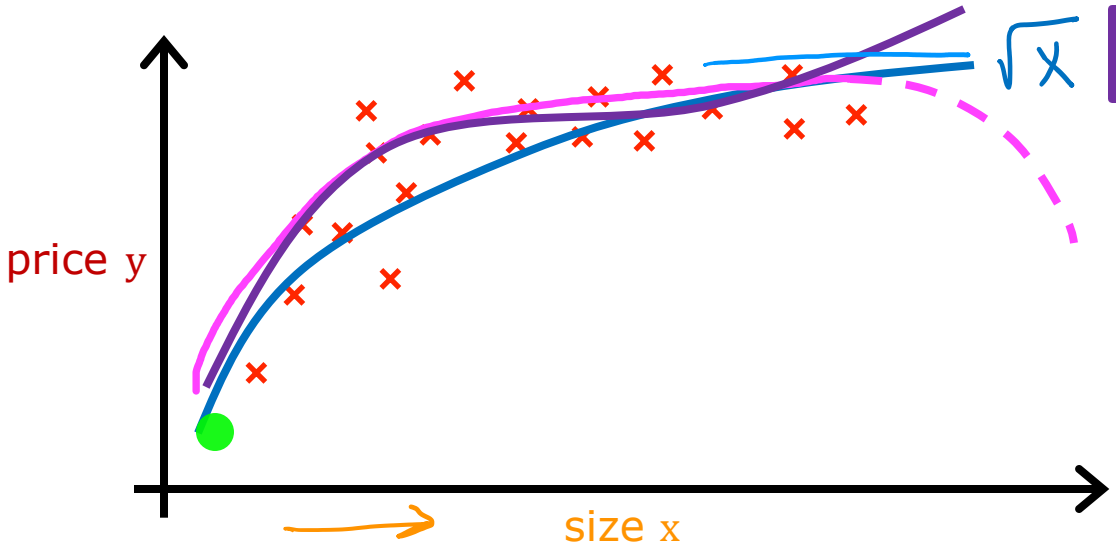
Polynomial Regression

↳ let you fit curves non-linear functions, to your data

Polynomial regression



Choice of features



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

size

√size

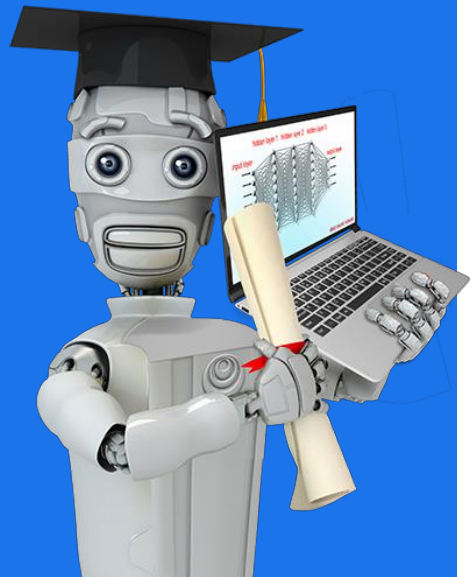
what features to use?

↳ course 2

Week 3

Stanford
ONLINE

DeepLearning.AI



Classification

Motivations

Classification

Question	Answer "y"	
Is this email spam?	no	yes
Is the transaction fraudulent?	no	yes
Is the tumor malignant?	no	yes

y can only be one of two values

"binary classification"

class = category

false true

0 1

useful for classification

"negative class"

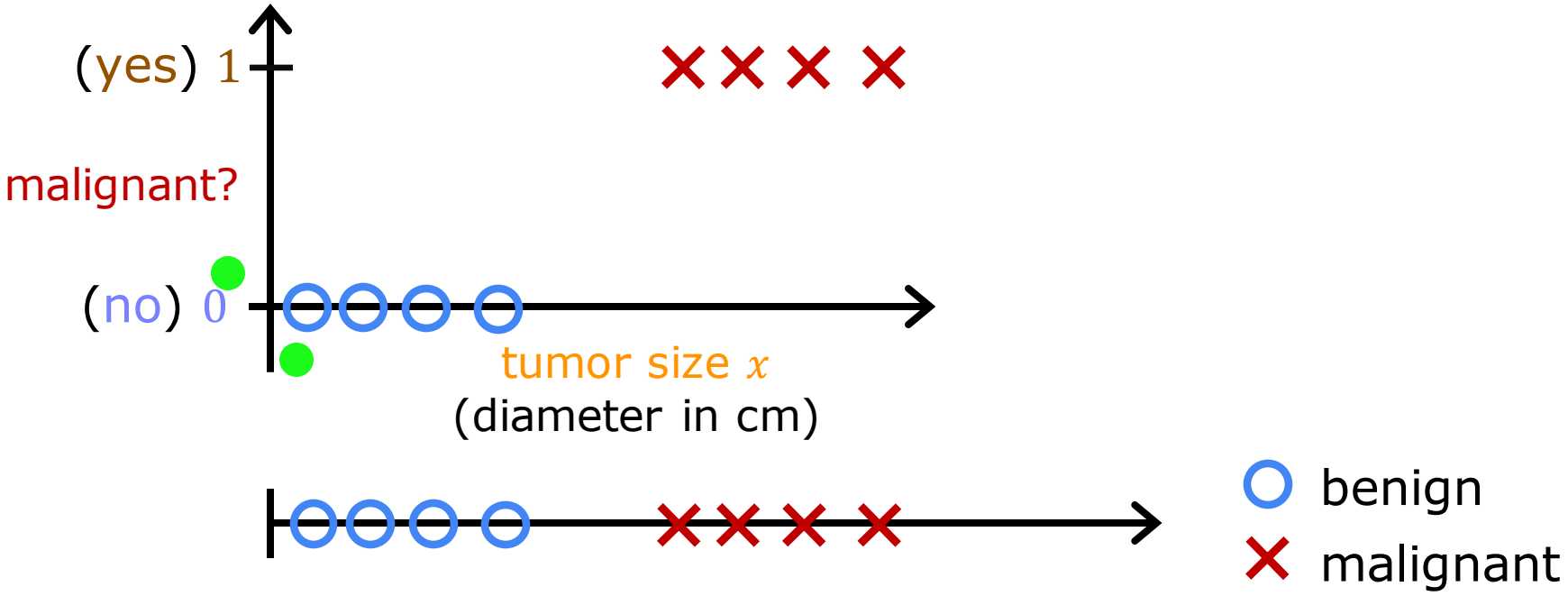
≠ "bad"

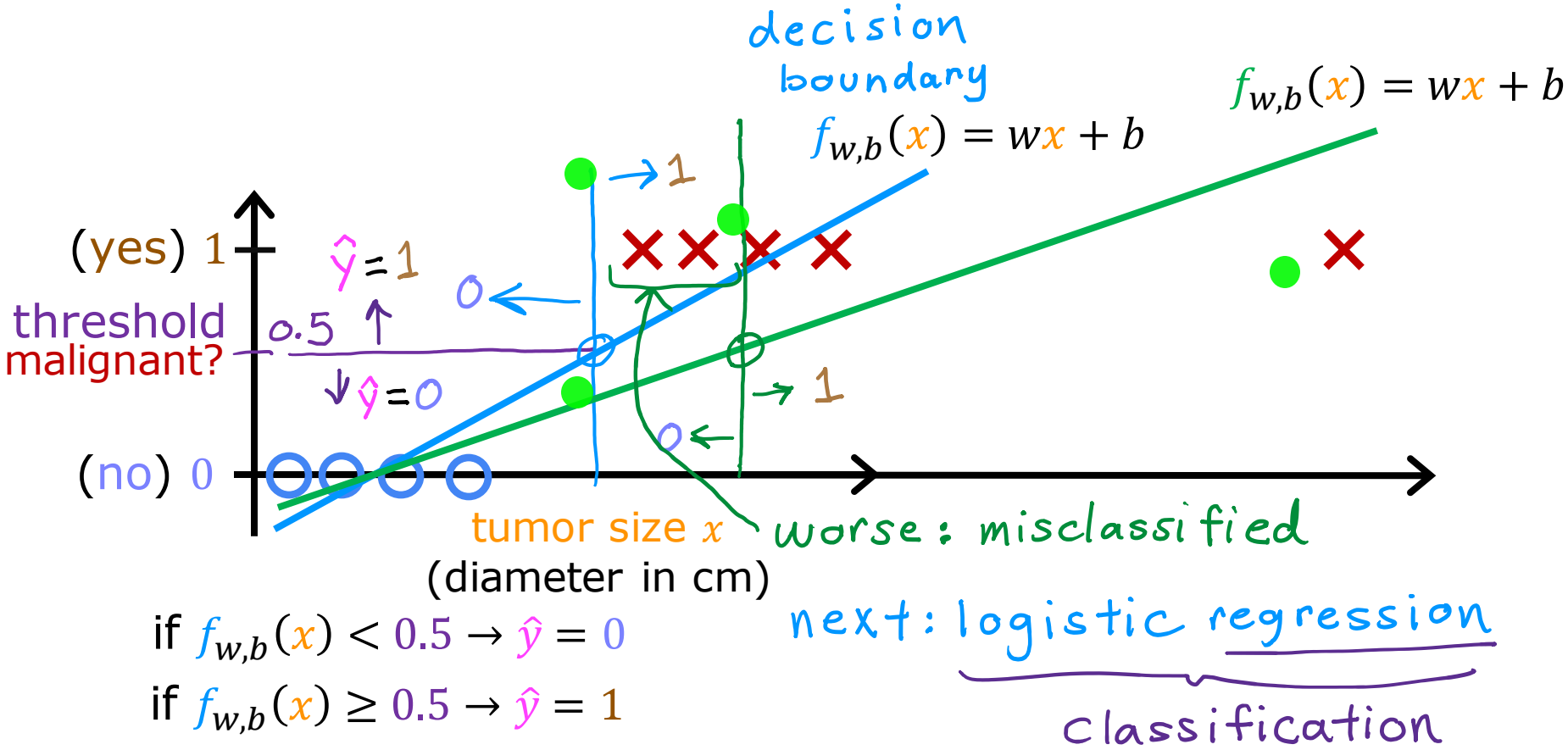
absence

"positive class"

≠ "good"

presence


(yes) 1
malignant?
(no) 0
tumor size x
(diameter in cm)
benign
malignant



if $f_{w,b}(x) < 0.5 \rightarrow \hat{y} = 0$

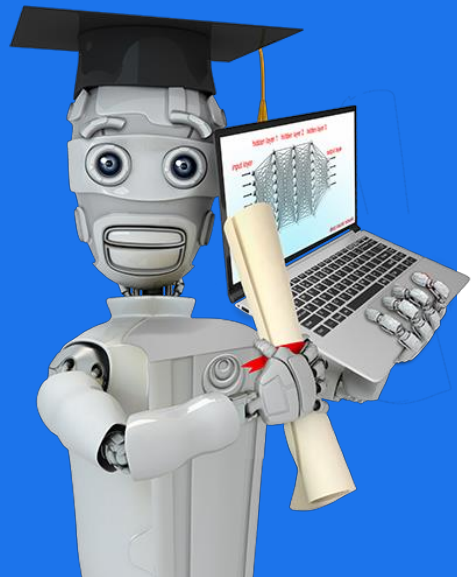
if $f_{w,b}(x) \geq 0.5 \rightarrow \hat{y} = 1$

next: logistic regression

classification

Stanford
ONLINE

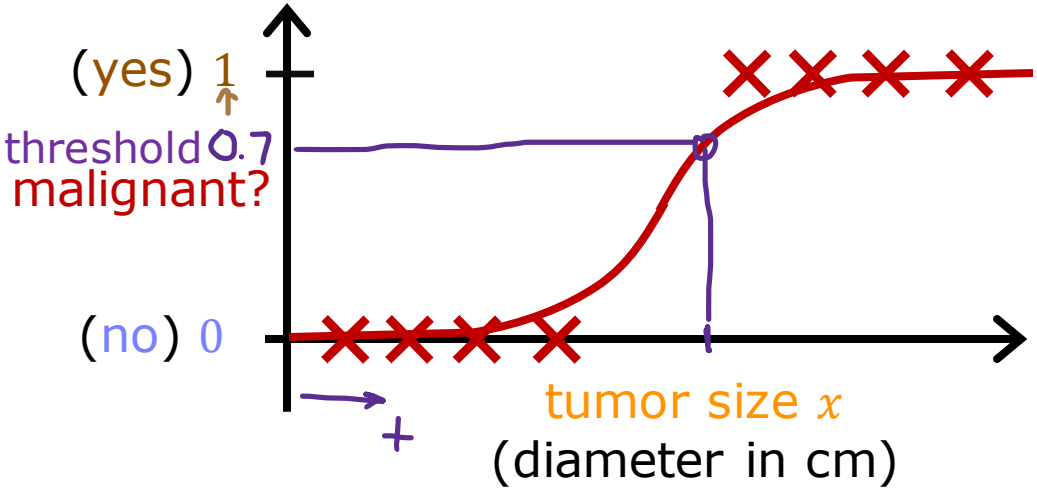
DeepLearning.AI



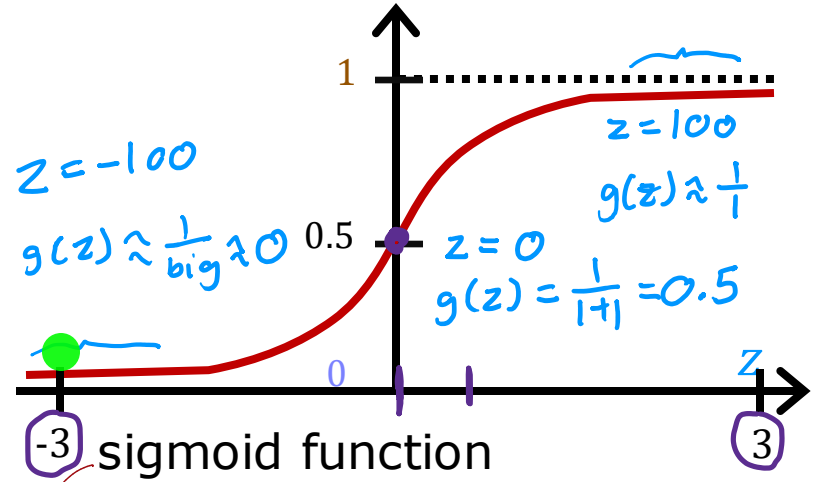
Classification

Binary classification

Logistic Regression



Want outputs between 0 and 1



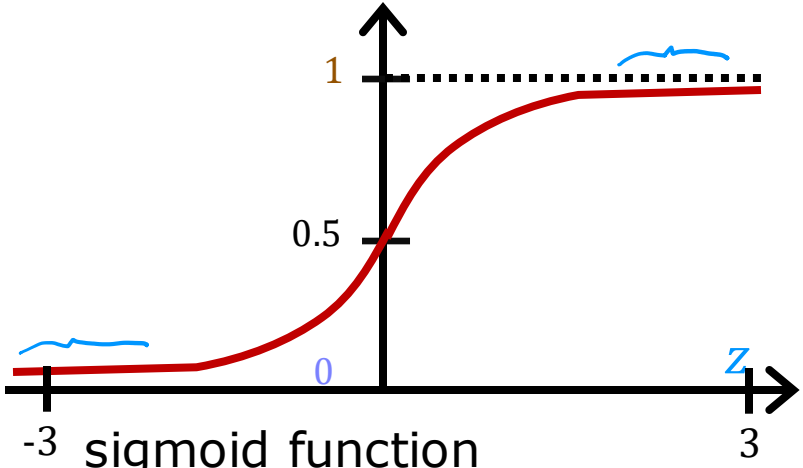
sigmoid function

or logistic function

outputs between 0 and 1

Sigmoid Func $\longrightarrow$ $g(z) = \frac{1}{1+e^{-z}}$ $\quad 0 < g(z) < 1$

Want outputs between 0 and 1



sigmoid function

logistic function

outputs between 0 and 1

$$g(z) = \frac{1}{1+e^{-z}} \qquad 0 < g(z) < 1$$

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

$\vec{w} \cdot \vec{x} + b$

z

$g(z) = \frac{1}{1+e^{-z}}$

$$f_{\vec{w},b}(\vec{x}) = g(\underbrace{\vec{w} \cdot \vec{x} + b}_{z}) = \frac{1}{1 + e^{-(\vec{w} \cdot \vec{x} + b)}}$$

"logistic regression"

$e \approx 2.7$

Interpretation of logistic regression output

$$f_{\vec{w},b}(\vec{x}) = \frac{1}{1 + e^{-(\vec{w}\cdot\vec{x}+b)}}$$

"probability" that class is 1

$$f_{\vec{w},b}(\vec{x}) = P(y = 1|\vec{x}; \vec{w}, b)$$

Probability that y is 1, given input $\vec{x}$, parameters $\vec{w}, b$

Example:

x is "tumor size"

y is 0 (not malignant) or 1 (malignant)

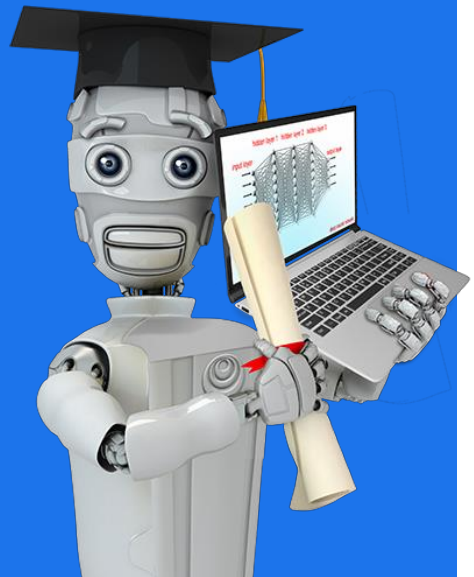
$$P(y = 0) + P(y = 1) = 1$$

$f_{\vec{w},b}(\vec{x}) = 0.7$

70% chance that y is 1

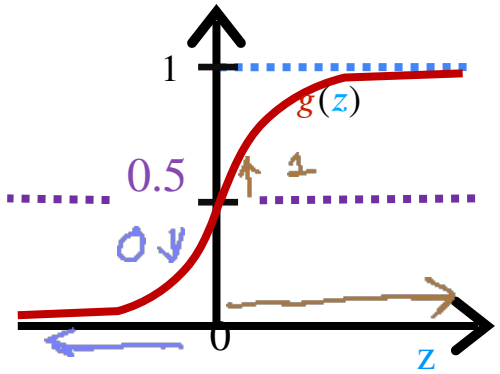
Stanford
ONLINE

DeepLearning.AI



Classification

Decision Boundary



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

$$g(z) = \frac{1}{1+e^{-z}}$$

$$= \frac{1}{1+e^{-\left(\vec{w} \cdot \vec{x} + b\right)}}$$

$$= P(y = 1 \mid x; \vec{w}, b)$$

0.7 0.3

0 or 1?

threshold

Is $f_{\vec{w},b}(\vec{x}) \geq 0.5$?

Yes: $\hat{y} = 1$ No: $\hat{y} = 0$

When is $f_{\vec{w},b}(\vec{x}) \geq 0.5$?

$g(z) \geq 0.5$

$z \geq 0$

$\vec{w} \cdot \vec{x} + b \geq 0$

$\hat{y} = 1$

$z < 0$

$\vec{w} \cdot \vec{x} + b < 0$

$\hat{y} = 0$

Decision boundary

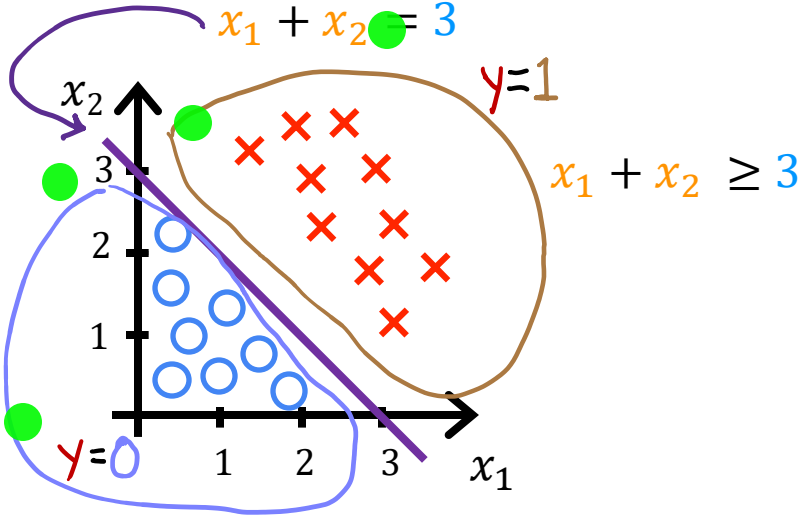
$$f_{\vec{w},b}(\vec{x}) = g(z) = g(w_1 x_1 + w_2 x_2 + b)$$

$w_1 = 1$, $w_2 = 1$, $b = -3$

Decision boundary $z = \vec{w} \cdot \vec{x} + b = 0$

$z = x_1 + x_2 - 3 = 0$

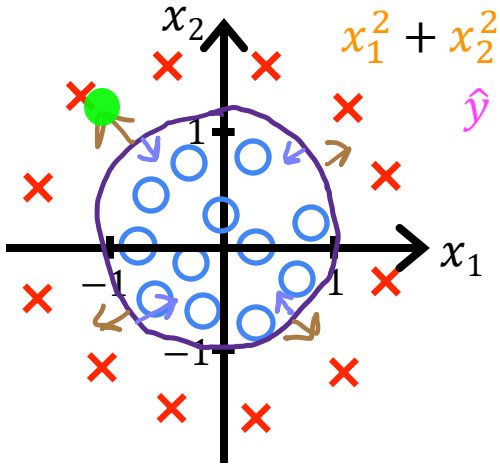
$x_1 + x_2 = 3$



$x_1 + x_2 \geq 3$

$x_1 + x_2 < 3$

Non-linear decision boundaries



$x_1^2 + x_2^2 \geq 1$

$\hat{y} = 1$

$x_1^2 + x_2^2 < 1$

$\hat{y} = 0$

$f_{\vec{w},b}(\vec{x}) = g(z) = g(\overbrace{w_1 x_1^2 + w_2 x_2^2 + b}^{z})$

$w_1 = 1, \quad w_2 = 1, \quad b = -1$

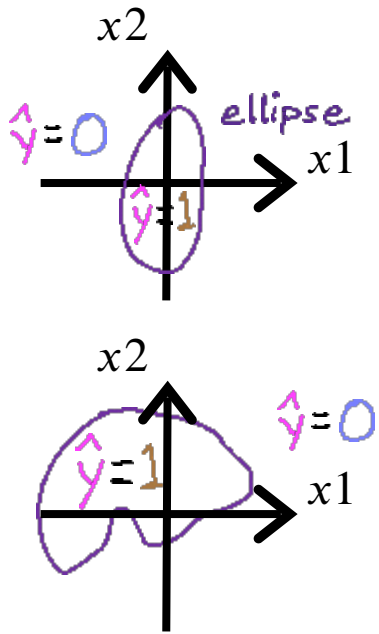
decision boundary $z = x_1^2 + x_2^2 - 1 = 0$

$x_1^2 + x_2^2 = 1$

a circle with $r = 1$

دائرة نصف قطرها 1

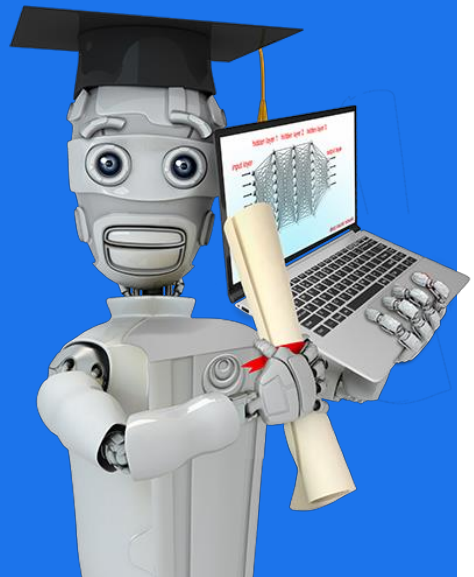
Non-linear decision boundaries



$$f_{\vec{w},b}\left(\vec{x}\right) = g(z) = g(w_1 x_1 + w_2 x_2 + w_3 x_1^2 + w_4 x_1 x_2 + w_5 x_2^2 + w_6 x_1^3 + \cdots + b)$$

Stanford
ONLINE

DeepLearning.AI



Cost Function

Cost Function for Logistic Regression

Training set

each row = a patient

binary classification

	tumor size (cm) x_1	...	patient's age x_n	malignant? y
$i=1$	10		52	1
	2		73	0
	5		55	0
	12		49	1
$i=m$	...		...	...

$i = 1, \ldots, m$ ← training examples

$j = 1, \ldots, n$ ← features

target y is 0 or 1

$$f_{\vec{w},b}(\vec{x}) = \frac{1}{1 + e^{-(\vec{w} \cdot \vec{x} + b)}}$$

How to choose $\vec{w} = [w_1 \quad w_2 \quad \cdots \quad w_n]$ and b?

Squared error cost

cost

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

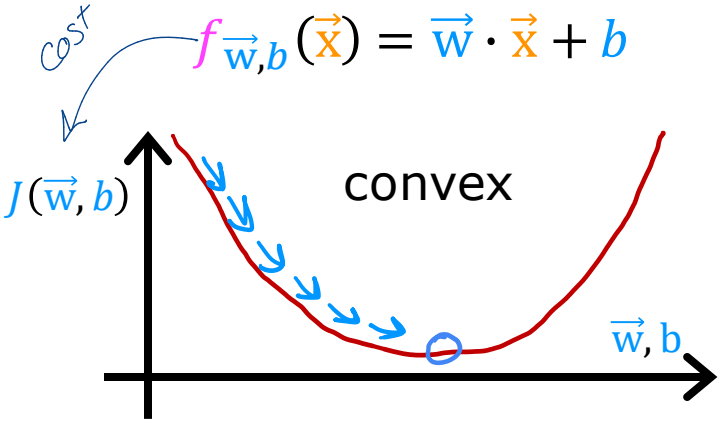
average of training set

loss

$$L\left(f_{\vec{w},b}\left(\vec{x}^{(i)}\right), y^{(i)}\right)$$

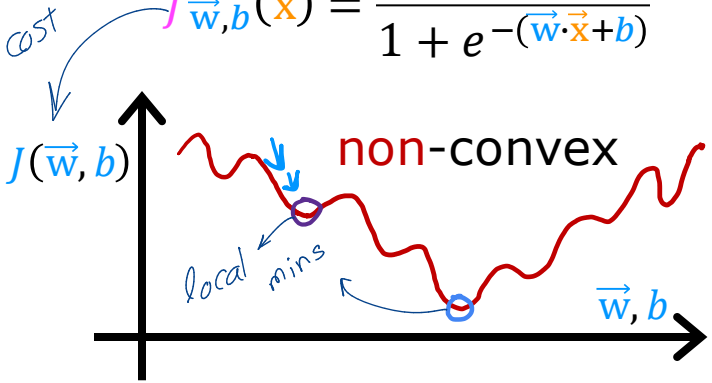
linear regression

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



logistic regression

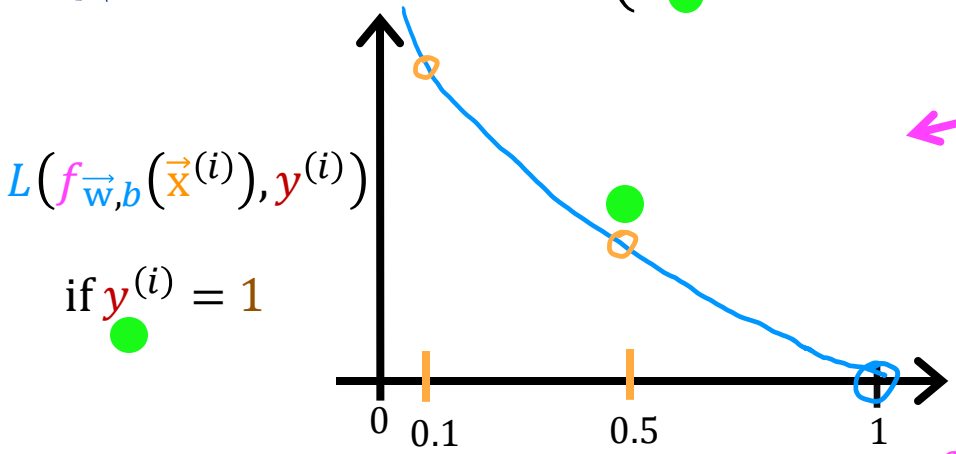
$$f_{\vec{w},b}(\vec{x}) = \frac{1}{1 + e^{-(\vec{w}\cdot\vec{x}+b)}}$$

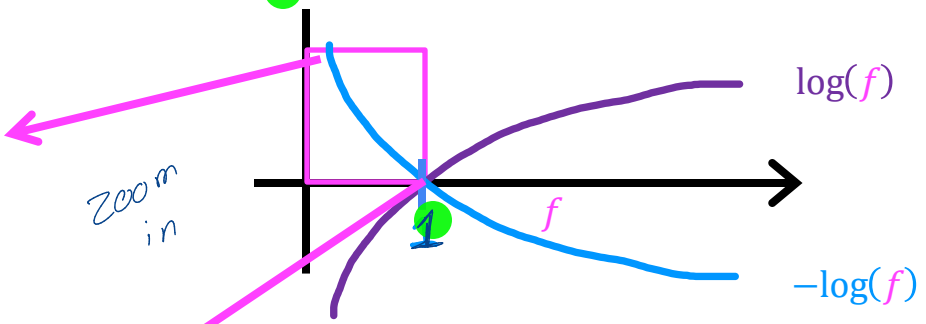


Logistic loss function

$$L\left(f_{\vec{w},b}\left(\vec{x}^{(i)}\right), y^{(i)}\right) = \begin{cases} -\log\left(f_{\vec{w},b}\left(\vec{x}^{(i)}\right)\right) & \text{if } y^{(i)} = 1 \\ -\log\left(1 - f_{\vec{w},b}\left(\vec{x}^{(i)}\right)\right) & \text{if } y^{(i)} = 0 \end{cases}$$

one data





As $f_{\vec{w},b}\left(\vec{x}^{(i)}\right) \to 1$ then loss $\to 0$

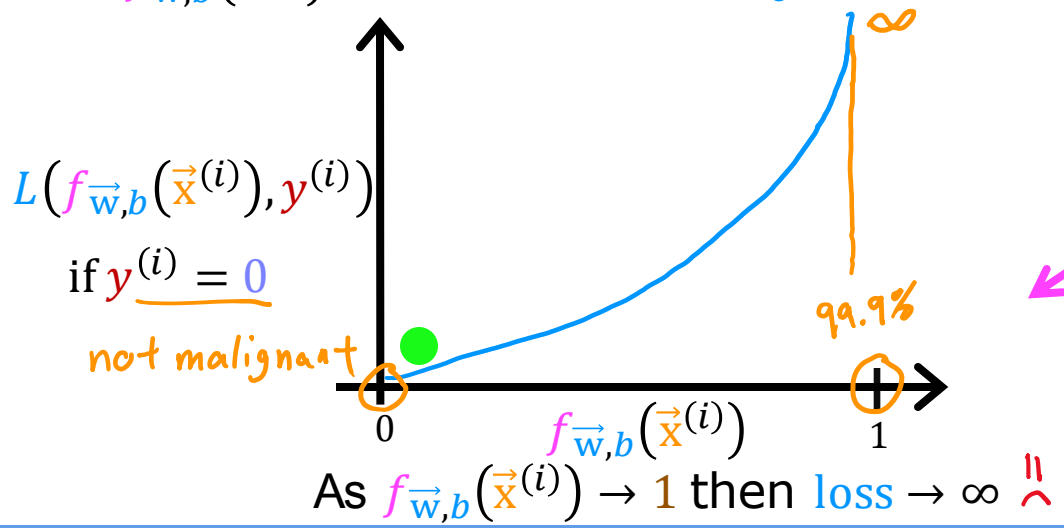
As $f_{\vec{w},b}\left(\vec{x}^{(i)}\right) \to 0$ then loss $\to \infty$

Loss is lowest when $f_{\vec{w},b}\left(\vec{x}^{(i)}\right)$ predicts close to true label $y^{(i)}$.

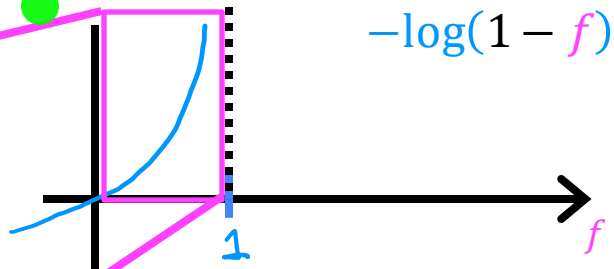
Logistic loss function

$$L\left(f_{\vec{w},b}\left(\vec{x}^{(i)}\right), y^{(i)}\right) = \begin{cases} -\log\left(f_{\vec{w},b}\left(\vec{x}^{(i)}\right)\right) & \text{if } y^{(i)} = 1 \\ -\log\left(1 - f_{\vec{w},b}\left(\vec{x}^{(i)}\right)\right) & \text{if } y^{(i)} = 0 \end{cases}$$

As $f_{\vec{w},b}\left(\vec{x}^{(i)}\right) \to 0$ then loss $\to 0$



As $f_{\vec{w},b}\left(\vec{x}^{(i)}\right) \to 1$ then loss $\to \infty$



The further prediction $f_{\vec{w},b}\left(\vec{x}^{(i)}\right)$ is from target $y^{(i)}$, the higher the loss.

Cost

cost

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

all training examples

loss

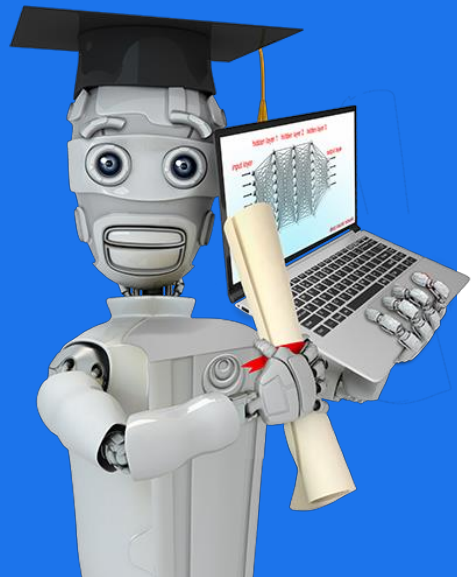
$$= \begin{cases} -\log\left(f_{\vec{w},b}\left(\vec{x}^{(i)}\right)\right) & \text{if } y^{(i)} = 1 \\ -\log\left(1 - f_{\vec{w},b}\left(\vec{x}^{(i)}\right)\right) & \text{if } y^{(i)} = 0 \end{cases}$$

convex
↳ can reach a global minimum

find w, b that minimize cost J

Stanford
ONLINE

DeepLearning.AI



Cost Function

Simplified Cost Function for Logistic Regression

Simplified loss function

$$L\left(f_{\vec{w},b}\left(\vec{x}^{(i)}\right), y^{(i)}\right) = \begin{cases} -\log\left(f_{\vec{w},b}\left(\vec{x}^{(i)}\right)\right) & \text{if } y^{(i)} = 1 \\ -\log\left(1 - f_{\vec{w},b}\left(\vec{x}^{(i)}\right)\right) & \text{if } y^{(i)} = 0 \end{cases}$$

$$L\left(f_{\vec{w},b}\left(\vec{x}^{(i)}\right), y^{(i)}\right) = -y^{(i)}\log\left(f_{\vec{w},b}\left(\vec{x}^{(i)}\right)\right) - \left(1 - y^{(i)}\right)\log\left(1 - f_{\vec{w},b}\left(\vec{x}^{(i)}\right)\right)$$

1

$(1-1)$

0

if $y^{(i)} = 1$:

$$L\left(f_{\vec{w},b}\left(\vec{x}^{(i)}\right), y^{(i)}\right) = -1\log\left(f(\vec{x})\right)$$

Simplified loss function

$$L\left(f_{\vec{w},b}\left(\vec{x}^{(i)}\right), y^{(i)}\right) = \begin{cases} -\log\left(f_{\vec{w},b}\left(\vec{x}^{(i)}\right)\right) & \text{if } y^{(i)} = 1 \\ -\log\left(1 - f_{\vec{w},b}\left(\vec{x}^{(i)}\right)\right) & \text{if } y^{(i)} = 0 \end{cases}$$

$$L\left(f_{\vec{w},b}\left(\vec{x}^{(i)}\right), y^{(i)}\right) = -y^{(i)}\log\left(f_{\vec{w},b}\left(\vec{x}^{(i)}\right)\right) - \left(1 - y^{(i)}\right)\log\left(1 - f_{\vec{w},b}\left(\vec{x}^{(i)}\right)\right)$$

$y^{(i)}$ → 0 ; $(1 - y^{(i)})$ → $(1 - 0)$

if $y^{(i)} = 1$:

$$L\left(f_{\vec{w},b}\left(\vec{x}^{(i)}\right), y^{(i)}\right) = -1 \log\left(f(\vec{x})\right)$$

if $y^{(i)} = 0$:

$$L\left(f_{\vec{w},b}\left(\vec{x}^{(i)}\right), y^{(i)}\right) = -(1-0)\log\left(1 - f(\vec{x})\right)$$

Simplified cost function

Simplified loss func

$$L\left(f_{\vec{w},b}(\vec{x}^{(i)}), y^{(i)}\right) = -y^{(i)}\log\left(f_{\vec{w},b}(\vec{x}^{(i)})\right) - (1-y^{(i)})\log\left(1-f_{\vec{w},b}(\vec{x}^{(i)})\right)$$

convex (single global minimum)

cost

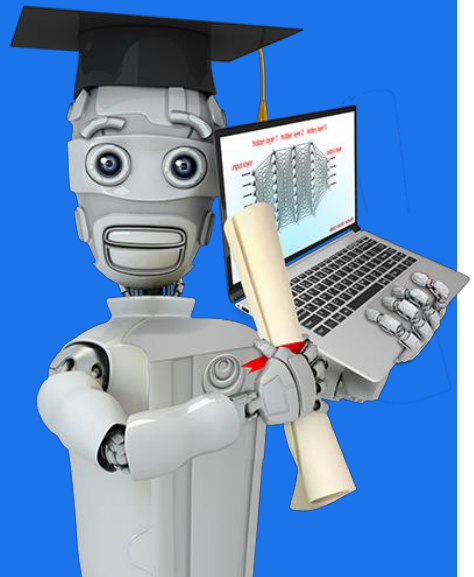
$$J(\vec{w},b) = \frac{1}{m}\sum_{i=1}^{m}\left[L\left(f_{\vec{w},b}(\vec{x}^{(i)}), y^{(i)}\right)\right]$$

$$= -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log\left(f_{\vec{w},b}(\vec{x}^{(i)})\right) + (1-y^{(i)})\log\left(1-f_{\vec{w},b}(\vec{x}^{(i)})\right)\right]$$

maximum likelihood (don't worry about it!)

Stanford
ONLINE

DeepLearning.AI



Gradient Descent

Gradient Descent Implementation

Training logistic regression

Find $\vec{w}, b$

Given new $\vec{x}$, output $f_{\vec{w},b}(\vec{x}) = \frac{1}{1+e^{-(\vec{w}\cdot\vec{x}+b)}}$

$$P(y = 1|\vec{x}; \vec{w}, b)$$

Gradient descent

cost

$$J(\vec{w}, b) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log\left(f_{\vec{w},b}(\vec{x}^{(i)})\right) + (1-y^{(i)})\log\left(1-f_{\vec{w},b}(\vec{x}^{(i)})\right)\right]$$

repeat {

$j = 1 \ldots n$

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

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

} simultaneous updates

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

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

Gradient descent for logistic regression

repeat {

looks like linear regression! (different in $f_{\vec{w},b}$ definition)

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

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

} simultaneous updates

Same concepts:

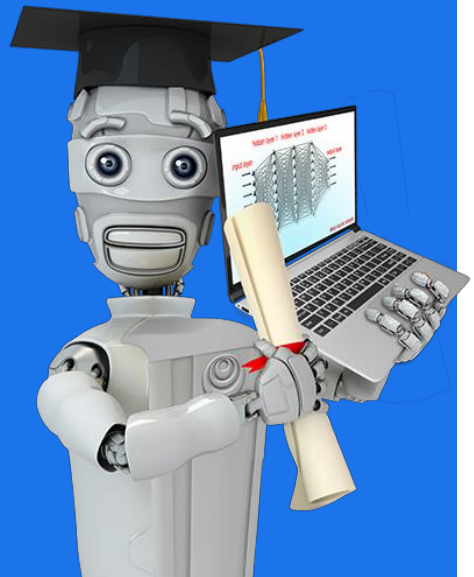
- Monitor gradient descent (learning curve)
- Vectorized implementation
- Feature scaling

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

Logistic regression $f_{\vec{w},b}(\vec{x}) = \dfrac{1}{1 + e^{(-\vec{w} \cdot \vec{x} + b)}}$

Stanford
ONLINE

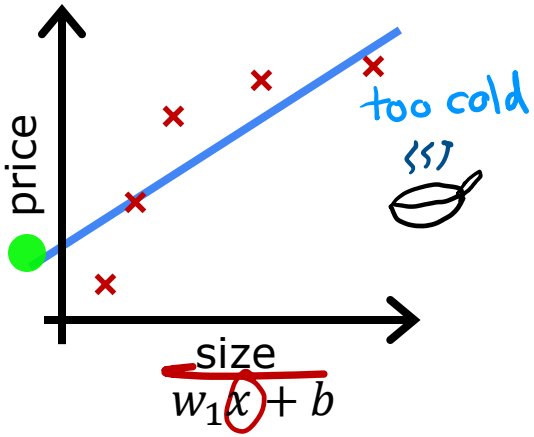
DeepLearning.AI



Regularization to Reduce Overfitting

The Problem of Overfitting

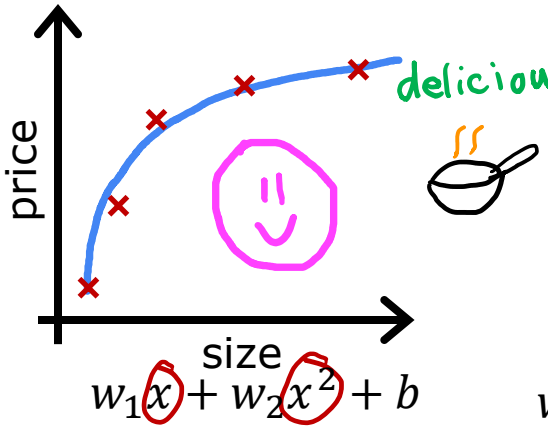
Regression example



underfit

- Does not fit the training set well

high bias

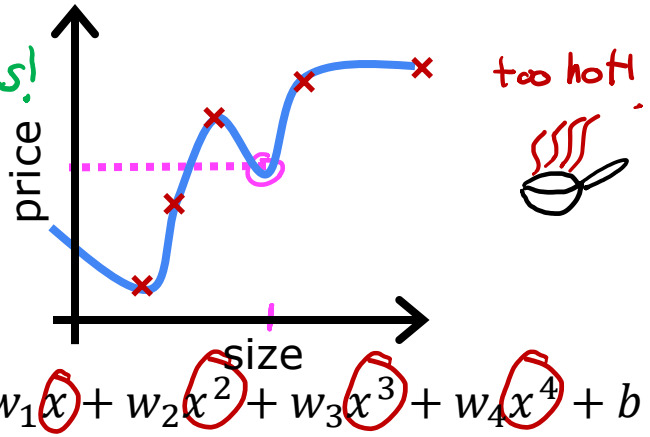


just right

- Fits training set pretty well

generalization

the idea that you want your learning algo to do well, even on examples that are not on the training set → that's called generalization



overfit: fits the data almost too well.

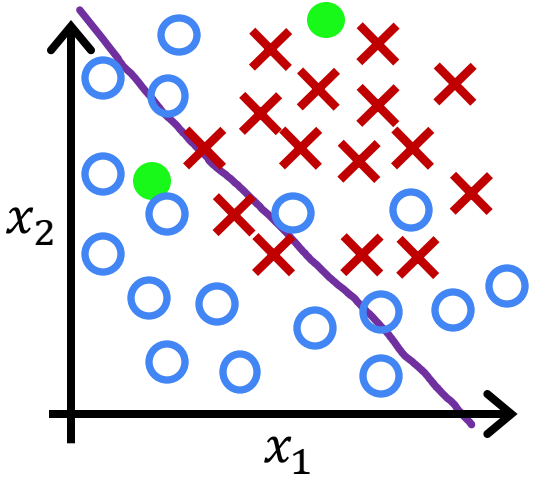
- Fits the training set extremely well

high variance

* underfitting : is a scenario in data science where a data model is unable to capture the relationship between the input and output variables accurately.

* overfitting : is the production of an analysis that corresponds too closely or exactly to a particular set of data , and may therefore fail to fit to additional data or predict future observations reliably.

Classification

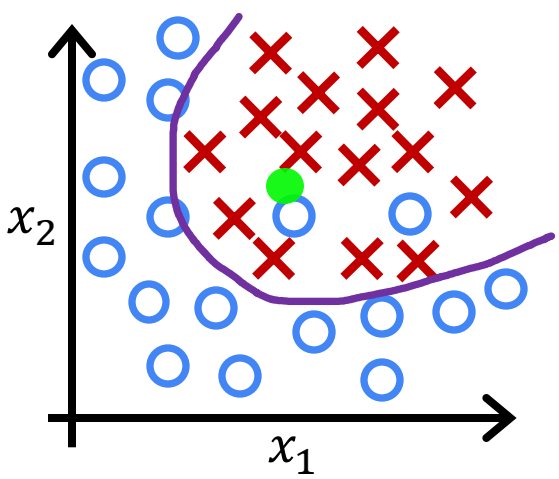


$$z = w_1 x_1 + w_2 x_2 + b$$

$$f_{\vec{w},b}(\vec{x}) = g(z)$$

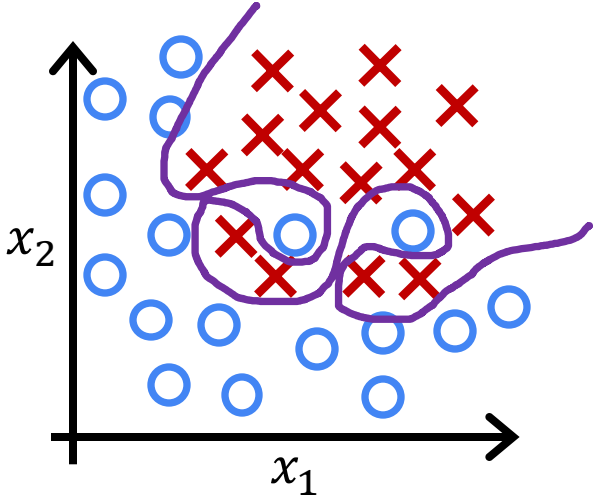
g is the sigmoid function

underfit high bias



$$z = w_1 x_1 + w_2 x_2 + w_3 x_1^2 + w_4 x_2^2 + w_5 x_1 x_2 + b$$

just right

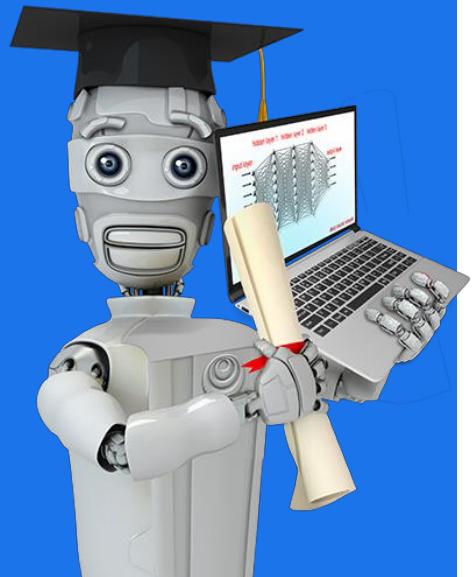


$$z = w_1 x_1 + w_2 x_2 + w_3 x_1^2 x_2 + w_4 x_1^2 x_2^2 + w_5 x_1^2 x_2^3 + w_6 x_1^3 x_2 + \cdots + b$$

overfit

Stanford
ONLINE

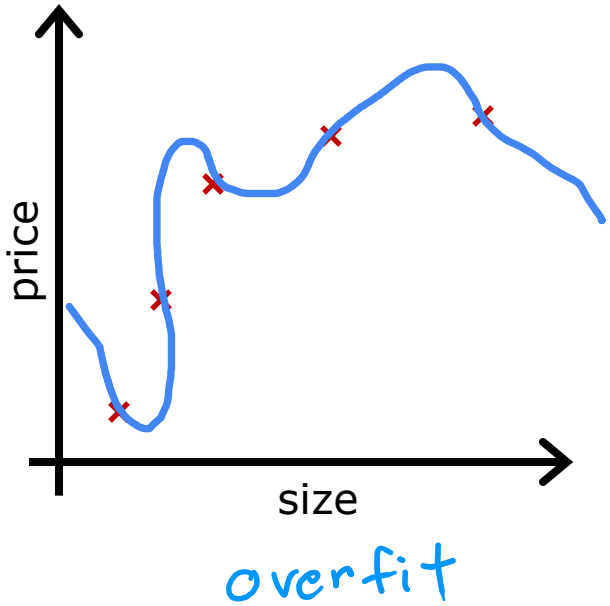
DeepLearning.AI

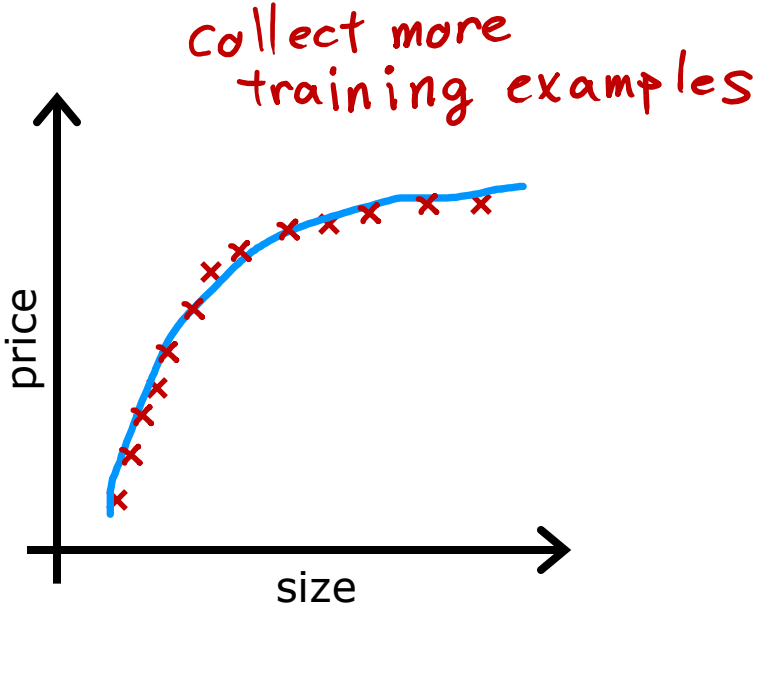


Regularization to Reduce Overfitting

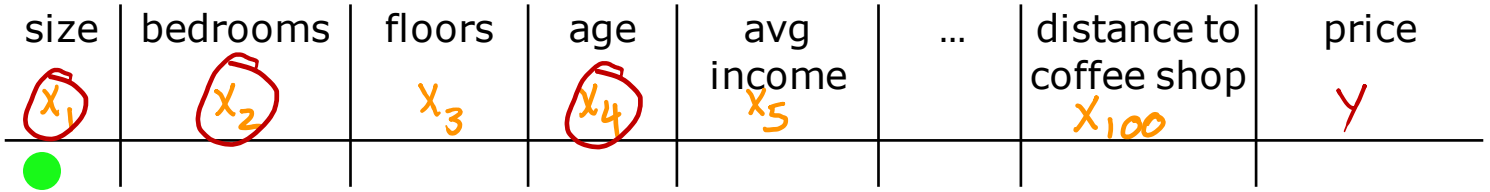
Addressing Overfitting

Collect more training examples





Select features to include/exclude



all features

\+

insufficient data

↓

overfit

selected features

size
bedrooms
age
just right

feature selection

course 2

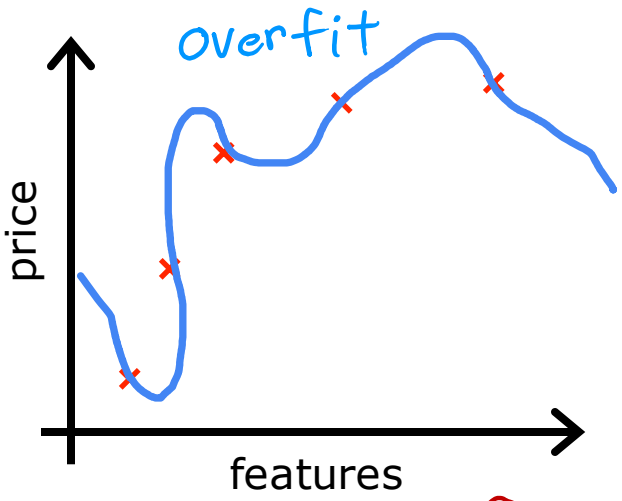
disadvantage

↓

useful features could be lost

Regularization

Reduce the size of parameters w_j

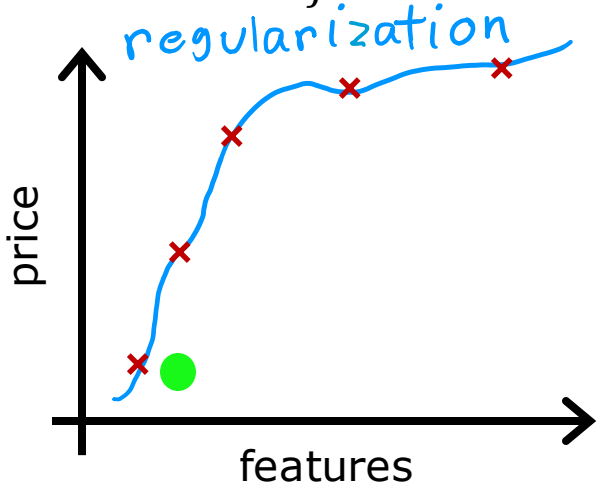


$$f(x) = 28x - 385x^2 + 39x^3 - 174x^4 + 100$$

0

large values for w_j

eliminate feature



$$f(x) = 13x - 0.23x^2 + 0.000014x^3 - 0.0001x^4 + 10$$

small values for w_j (small effect)

Addressing overfitting

Options

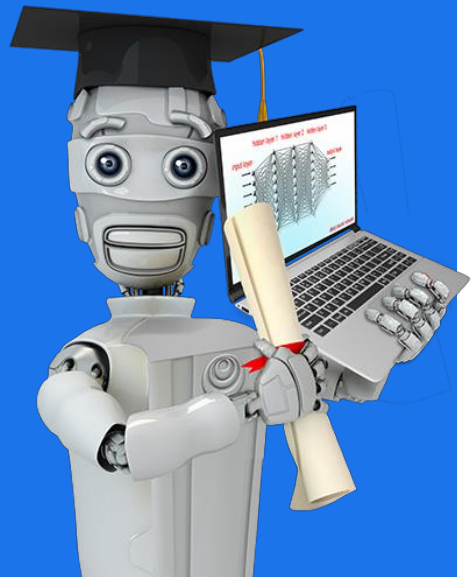
1. Collect more data

2. Select features
   - Feature selection in course 2

3. Reduce size of parameters
   - "Regularization" next videos!

Stanford
ONLINE

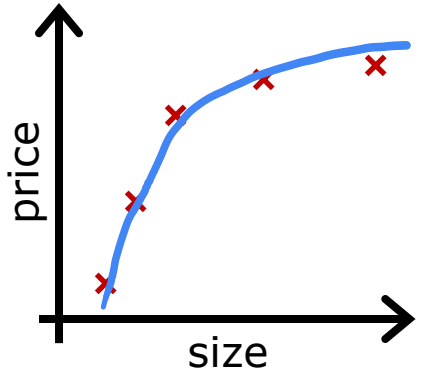
DeepLearning.AI



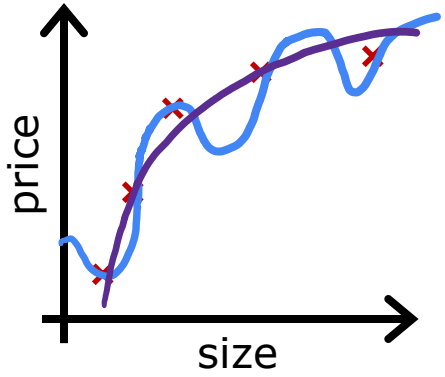
Regularization to Reduce Overfitting

Cost Function with Regularization

Intuition



$w_1 x + w_2 x^2 + b$



$w_1 x + w_2 x^2 + w_3 x^3 + w_4 x^4 + b$

$w_3 \approx 0$ $w_4 \approx 0$

make w_3, w_4 really small (≈ 0)

$$\min_{\vec{w},b} \frac{1}{2m} \sum_{i=1}^{m} \left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)} \right)^2 + 1000 w_3^2 + 1000 w_4^2$$

$w_3 \approx 0.001$ $w_4 \approx 0.002$

Regularization

small values $w_1, w_2, \cdots, w_n, b$

simpler model

less likely to overfit

$w_3 \approx 0$

$w_4 \approx 0$

size	bedrooms	floors	age	avg income	...	distance to coffee shop	price
x_1	x_2	x_3	x_4	x_5		x_{100}	y

$n = 100$

n features

$w_1, w_1, w_2, \cdots, w_{100}, b$

regularization term

$$J(\vec{w}, b) = \frac{1}{2m}\left[\sum_{i=1}^{m}\left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)}\right)^2 + \frac{\lambda}{2m}\sum_{j=1}^{n} w_j^2 + \frac{\lambda}{2m} b^2\right]$$

can include or exclude b

"lambda"

regularization parameter

$\lambda > 0$

Regularization

mean squared error

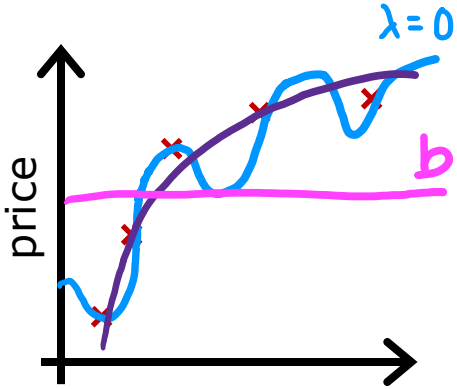
regularization term

$$\min_{\vec{w},b} J(\vec{w},b) = \min_{\vec{w},b} \left[\frac{1}{2m} \sum_{i=1}^{m} \left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)} \right)^2 + \frac{\lambda}{2m} \sum_{j=1}^{n} w_j^2 \right]$$

fit data

Keep w_j small

λ balances both goals



choose $\lambda = 10^{10}$

$$f_{\vec{w},b}(\vec{x}) = \underset{\approx 0}{w_1 x} + \underset{\approx 0}{w_2 x^2} + \underset{\approx 0}{w_3 x^3} + \underset{\approx 0}{w_4 x^4} + b$$

$f(x) = b$

Choose λ

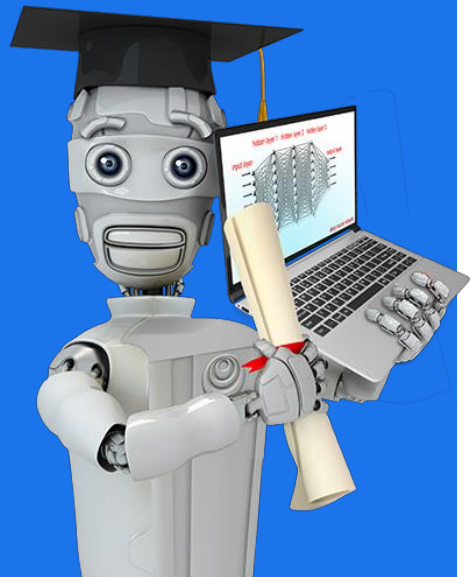
recap

if $\lambda = 0$ → overfit

if $\lambda = \infty$ (large num) → underfit

Stanford
ONLINE

DeepLearning.AI



Regularization to Reduce Overfitting

Regularized Linear Regression

Regularized linear regression

$$\min_{\vec{w},b} J(\vec{w},b) = \min_{\vec{w},b}\left[\frac{1}{2m}\sum_{i=1}^{m}\left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)}\right)^2 + \frac{\lambda}{2m}\sum_{j=1}^{n} w_j^2\right]$$

Gradient descent

repeat {

$$w_j = w_j - \alpha \frac{\partial}{\partial w_j} J(\vec{w},b) \qquad j = 1,\ldots,n$$

$$= \frac{1}{m}\sum_{i=1}^{m}\left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)}\right) x_j^{(i)} + \frac{\lambda}{m} w_j$$

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

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

don't have to regularize b

} simultaneous update

Implementing gradient descent

repeat {

- $w_j = w_j - \alpha \left[\frac{1}{m} \sum_{i=1}^{m} \left[\left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)} \right) x_j^{(i)} \right] + \frac{\lambda}{m} w_j \right]$

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

} simultaneous update $j = 1 \ldots n$

Implementing gradient descent

repeat {

$$w_j = w_j - \alpha \left[\frac{1}{m} \sum_{i=1}^{m} \left[\left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)} \right) x_j^{(i)} \right] + \frac{\lambda}{m} w_j \right]$$

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

} simultaneous update $j = 1 \ldots n$

$$w_j = \underbrace{1 w_j - \alpha \frac{\lambda}{m} w_j}_{w_j \left(1 - \alpha \frac{\lambda}{m}\right)} - \underbrace{\alpha \frac{1}{m} \sum_{i=1}^{m} \left(f_{w,b}(\vec{x}^{(i)}) - y^{(i)} \right) x_j^{(i)}}_{\text{usual update}}$$

shrink w_j

$\alpha \frac{\lambda}{m}$

$0.01 \frac{1}{50} = 0.0002$

$w_j (1 - 0.0002)$

0.9998

How we get the derivative term (optional)

$$\frac{\partial}{\partial w_j} J(\vec{w}, b) = \frac{d}{dw_j}\left[\frac{1}{2m}\sum_{i=1}^{m}\left(\underbrace{f(\vec{x}^{(i)})}_{\vec{w}\cdot\vec{x}^{(i)}+b} - y^{(i)}\right)^2 + \frac{\lambda}{2m}\sum_{j=1}^{n} w_j^2\right]$$

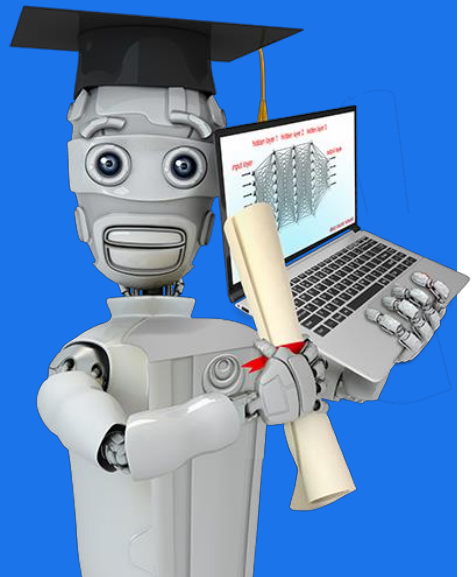
$$= \frac{1}{2m}\sum_{i=1}^{m}\left[\left(\vec{w}\cdot\vec{x}^{(i)} + b - y^{(i)}\right) 2x_j^{(i)}\right] + \frac{\lambda}{2m} 2w_j \quad \text{No } \sum_{j=1}^{n}$$

$$= \frac{1}{m}\sum_{i=1}^{m}\left[\left(\underbrace{\vec{w}\cdot\vec{x}^{(i)} + b}_{f(\vec{x})} - y^{(i)}\right) x_j^{(i)}\right] + \frac{\lambda}{m} w_j$$

$$= \frac{1}{m}\sum_{i=1}^{m}\left[\left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)}\right)x_j^{(i)}\right] + \frac{\lambda}{m} w_j$$

Stanford
ONLINE

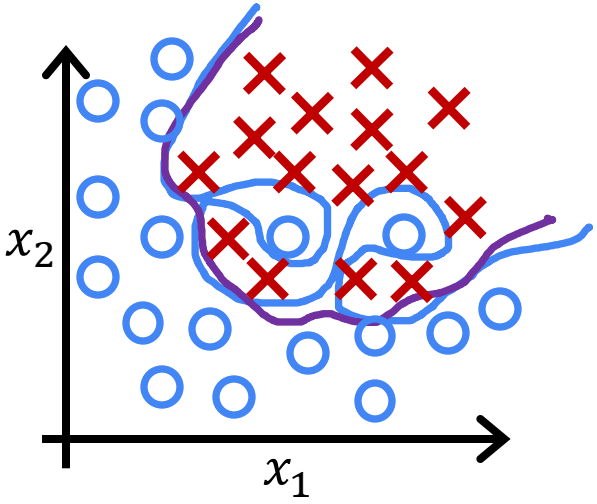
DeepLearning.AI



Regularization to Reduce Overfitting

Regularized Logistic Regression

Regularized logistic regression



$$z = w_1 x_1 + w_2 x_2 + w_3 x_1^2 x_2 + w_4 x_1^2 x_2^2 + w_5 x_1^2 x_2^3 + \cdots + b$$

$$f_{\vec{w},b}(\vec{x}) = \frac{1}{1+e^{-z}}$$

Cost function

$$J(\vec{w},b) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log\left(f_{\vec{w},b}(\vec{x}^{(i)})\right) + (1-y^{(i)})\log\left(1-f_{\vec{w},b}(\vec{x}^{(i)})\right)\right] + \frac{\lambda}{2m}\sum_{j=1}^{n} w_j^2$$

$$\min_{\vec{w},b} J(\vec{w},b) \rightarrow w_j \downarrow$$

Regularized logistic regression

$$J(\vec{w}, b) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log\left(f_{\vec{w},b}(\vec{x}^{(i)})\right) + (1-y^{(i)})\log\left(1 - f_{\vec{w},b}(\vec{x}^{(i)})\right)\right] + \frac{\lambda}{2m}\sum_{j=1}^{n} w_j^2$$

$\min_{\vec{w},b}$

Gradient descent

repeat {

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

$j = 1 \ldots n$

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

}

Looks same as for linear regression!

$$= \frac{1}{m}\sum_{i=1}^{m}\left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)}\right)x_j^{(i)} + \frac{\lambda}{m}w_j$$

logistic regression

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

don't have to regularize b