

<u>Week 1 : Introduction To Machine Learning</u>	<u>Page No.</u>
1.1 - Overview of Machine Learning	
1.1.1 - Welcome to machine learning	1
1.1.2 - Applications of machine learning	
1.2 - Supervised vs. Unsupervised Machine Learning	
1.2.1 - What is machine learning?	
1.2.2 - Supervised learning part 1	2
1.2.3 - Supervised learning part 2	
1.2.4 - Unsupervised learning part 1	
1.2.5 - Unsupervised learning part 2	
1.2.6 - Jupyter Notebooks	
Lab: Python and Jupyter Notebooks	
1.3 - Regression Model	
1.3.1 - Linear regression model part 1	
1.3.2 - Linear regression model part 2	
Lab: Optional lab: Model representation	
1.3.3 - Cost function formula	
1.3.4 - Cost function intuition	
1.3.5 - Visualizing the cost function	
1.3.6 - Visualization examples	
Lab: Optional lab: Cost function	
1.4 - Train the model with gradient descent	
1.4.1 - Gradient descent	
1.4.2 - Implementing gradient descent	
1.4.3 - Gradient descent intuition	
1.4.4 - Learning rate	
1.4.5 - Gradient descent for linear regression	
1.4.6 - Running gradient descent	
Lab: Optional lab: Gradient descent	
<u>Week (Part) 2 : Regression with multiple input variables</u>	17
1.1 - Overview of Machine Learning	
1.1.1 - Welcome to machine learning!	
1.1.2 - Applications of machine learning	
1.2 - Supervised vs. Unsupervised Machine Learning	
1.2.1 - What is machine learning?	
1.2.2 - Supervised learning part 1	
1.2.3 - Supervised learning part 2	
1.2.4 - Unsupervised learning part 1	
1.2.5 - Unsupervised learning part 2	
1.2.6 - Jupyter Notebooks	
Lab: Python and Jupyter Notebooks	
1.3 - Regression Model	
1.3.1 - Linear regression model part 1	
1.3.2 - Linear regression model part 2	
Lab: Optional lab: Model representation	
1.3.3 - Cost function formula	
1.3.4 - Cost function intuition	
1.3.5 - Visualizing the cost function	
1.3.6 - Visualization examples	
Lab: Optional lab: Cost function	
1.4 - Train the model with gradient descent	
1.4.1 - Gradient descent	
1.4.2 - Implementing gradient descent	
1.4.3 - Gradient descent intuition	
1.4.4 - Learning rate	
1.4.5 - Gradient descent for linear regression	
1.4.6 - Running gradient descent	
Lab: Optional lab: Gradient descent	

Week (Part) 3 : Classification with logistic regression

33

3.1 - Classification with logistic regression

3.1.1 - Motivations 9m

3.1.2 - Logistic regression 9m

3.1.3 - Decision boundary 10m

3.2 - Cost function for logistic regression 11m

3.2.1 - Cost function for logistic regression

3.2.2 - Simplified Cost Function for Logistic Regression 5m

3.3 - Gradient descent for logistic regression

3.3.1 - Gradient Descent Implementation 6m

3.4 - The problem of overfitting 11m

3.4.1 - The problem of overfitting

3.4.2 - Addressing overfitting 8m

3.4.3 - Cost function with regularization 9m

3.4.4 - Regularized linear regression 8m

3.4.5 - Regularized logistic regression 5m

Andrew Ng and Fei-Fei Li on Human-Centered AI 41m

1 reading

Acknowledgments 2m

4 practice exercises

Practice quiz: Classification with logistic regression 30m

Practice quiz: Cost function for logistic regression 30m

Practice quiz: Gradient descent for logistic regression 30m

Practice quiz: The problem of overfitting 30m

Week 1 : Introduction To Machine Learning

1.1 - Overview of Machine Learning

- 1.1.1 - Welcome to machine learning!
- 1.1.2 - Applications of machine learning

1.2 - Supervised vs. Unsupervised Machine Learning

- 1.2.1 - What is machine learning?
- 1.2.2 - Supervised learning part 1
- 1.2.3 - Supervised learning part 2
- 1.2.4 - Unsupervised learning part 1
- 1.2.5 - Unsupervised learning part 2
- 1.2.6 - Jupyter Notebooks

Lab: Python and Jupyter Notebooks

1.3 - Regression Model

- 1.3.1 - Linear regression model part 1
- 1.3.2 - Linear regression model part 2

Lab: Optional lab: Model representation

- 1.3.3 - Cost function formula
- 1.3.4 - Cost function intuition
- 1.3.5 - Visualizing the cost function
- 1.3.6 - Visualization examples

Lab: Optional lab: Cost function

1.4 - Train the model with gradient descent

- 1.4.1 - Gradient descent
- 1.4.2 - Implementing gradient descent
- 1.4.3 - Gradient descent intuition
- 1.4.4 - Learning rate
- 1.4.5 - Gradient descent for linear regression
- 1.4.6 - Running gradient descent

Lab: Optional lab: Gradient descent

1.2 - Supervised vs. Unsupervised Machine Learning : 1.2.1 - What is machine learning? :

it's the science of getting computers to learn without being explicitly programmed.

machine learning is the science of getting computers to learn without being explicitly programmed.

Learning Objectives

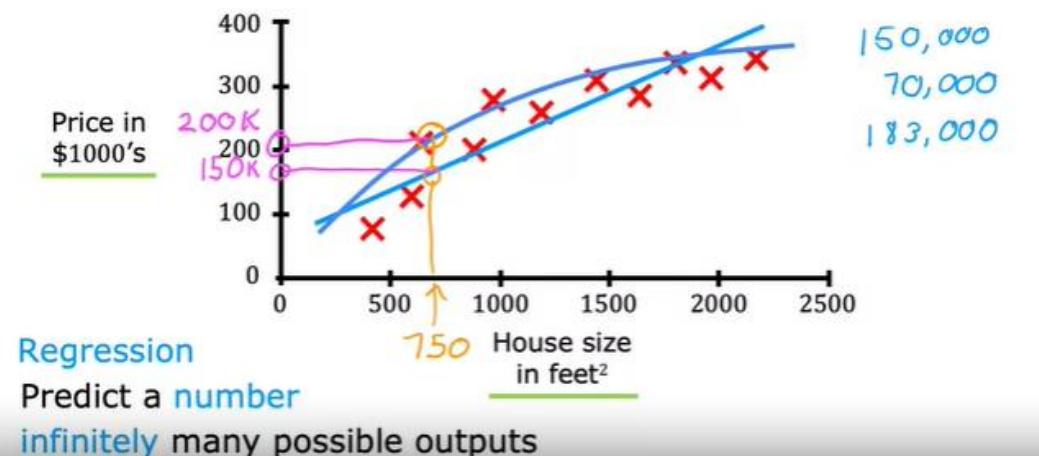
- Define machine learning
- Define supervised learning
- Define unsupervised learning
- Write and run Python code in Jupyter Notebooks
- Define a regression model
- Implement and visualize a cost function
- Implement gradient descent
- Optimize a regression model using gradient descent

supervised learning :

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

1.2.2 - supervised learning :part 1- Regression

Regression: Housing price prediction



1.2.3 - supervised learning :part 2- Classification

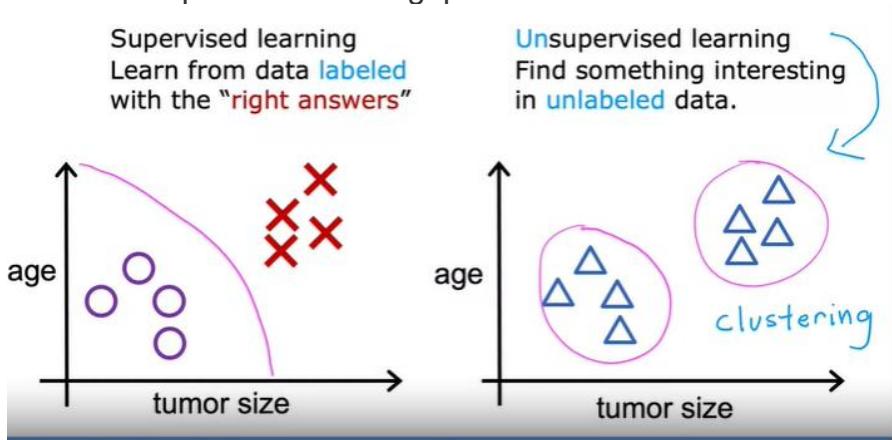
Classification: Breast cancer detection

- benign
 - ✗ malignant type 1
 - △ malignant type 2



Classification *class category*
predict categories cat dog benign malignant 0, 1, 2
small number of possible outputs

1.2.4 - Unsupervised learning : part 1-



1.2.4 - Unsupervised learning : Clustering - Eg. 1 - Google News

Clustering: Google news

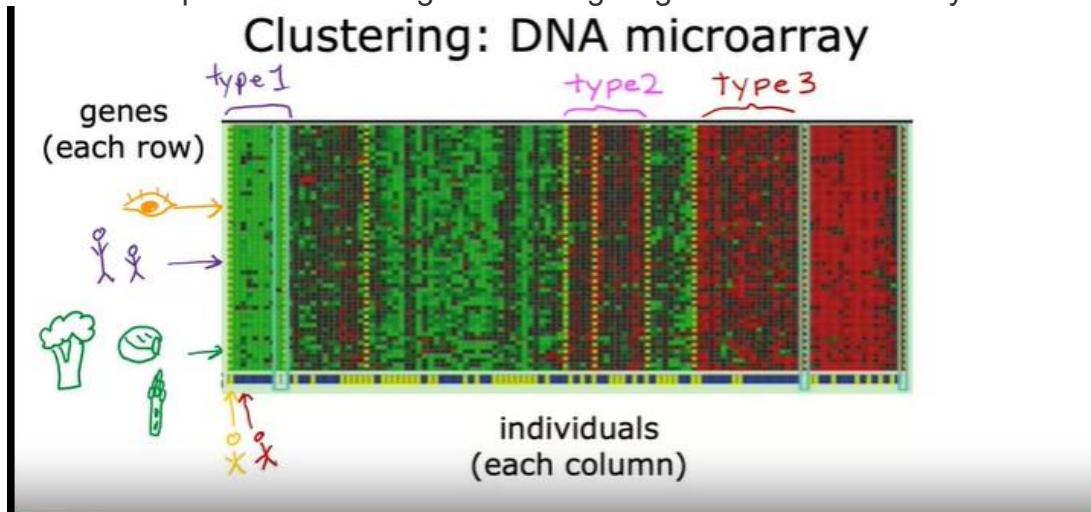
Giant **panda** gives birth to rare **twin cubs** at Japan's oldest **zoo**
USA TODAY · 6 hours ago

- Giant **panda** gives birth to **twin cubs** at Japan's oldest **zoo**
CBS News · 7 hours ago
- Giant **panda** gives birth to **twin cubs** at Tokyo's Ueno **Zoo**
WHBL News · 16 hours ago
- A Joyful Surprise at Japan's Oldest **Zoo**: The Birth of **Twin Pandas**
The New York Times · 1 hour ago
- Twin **Panda** **Cubs** Born at Tokyo's Ueno **Zoo**
PEOPLE · 6 hours ago

[View Full Coverage](#)

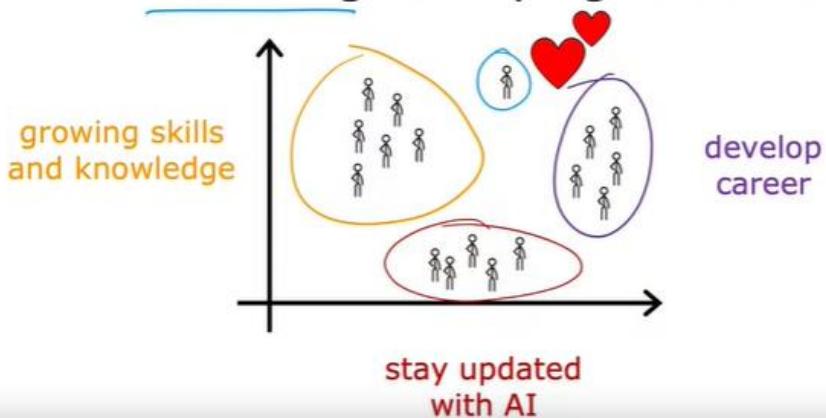
- the algorithm has to figure out on his own without supervision, what are the clusters of news articles today. So that's why this clustering algorithm, is a type of **unsupervised learning algorithm**.
- I don't know what the different types of people are but can you automatically find structure into data. And automatically figure out what are the major types of individuals, since we're not given the algorithm the right answer for the examples in advance. This is **unsupervised learning**

1.2.4 - Unsupervised learning : Clustering - Eg. 2 - DNA microarray



1.2.4 - Unsupervised learning : Clustering - Eg. 3 - Grouping Customers

Clustering: Grouping customers



1.2.5 - 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.

Clustering

Group similar data points together.

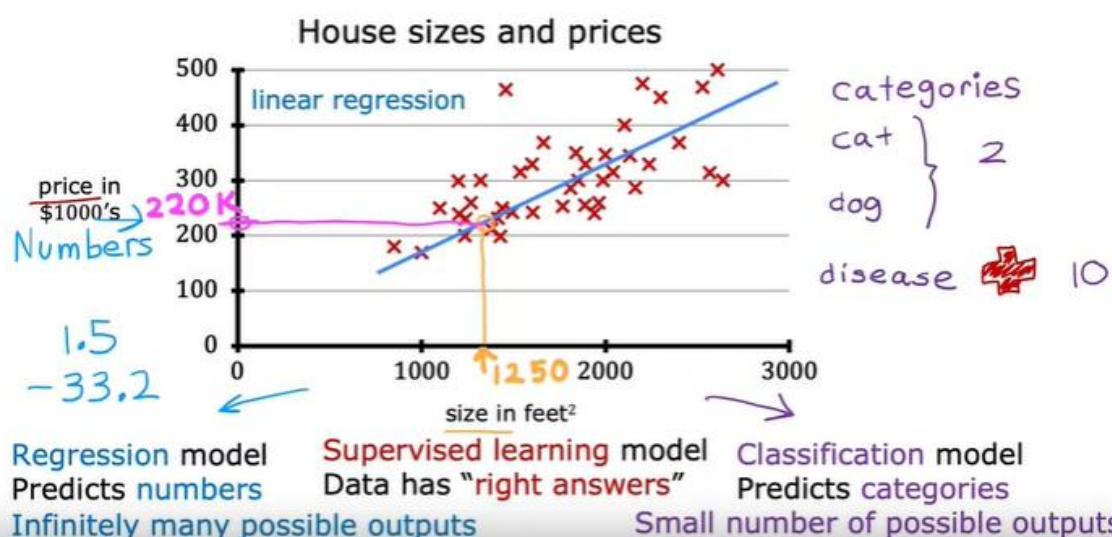
Dimensionality reduction

Compress data using fewer numbers.

Anomaly detection

Find unusual data points.

1.3 - Regression Model : 1.3.1 - Linear regression model part 1 :





Terminology

Training Data used to train the model

set: x size in feet² y price in \$1000's

	x	y	
(1)	2104	400	
(2)	1416	232	
(3)	1534	315	
(4)	852	178	
...	
(47)	3210	870	

$m = 47$

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

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

$x^{(2)} = 1416 \quad x^{(2)} \neq x^2 \text{ not exponential}$

Notation:

x = “input” variable
feature

y = “output” variable
“target” variable

m = number of training examples

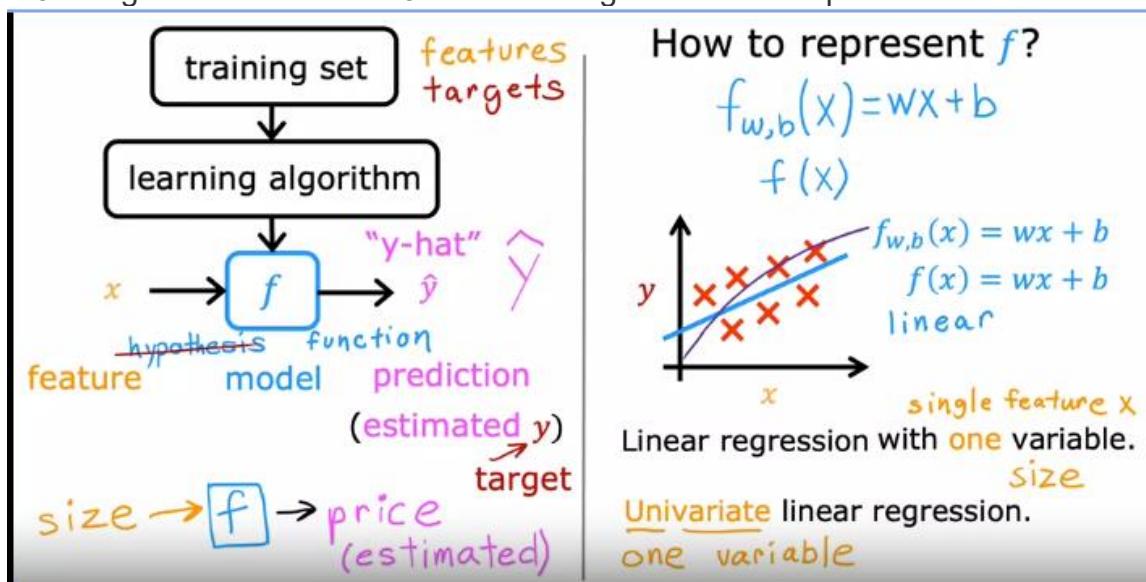
(x, y) = single training example

$(x^{(i)}, y^{(i)})$

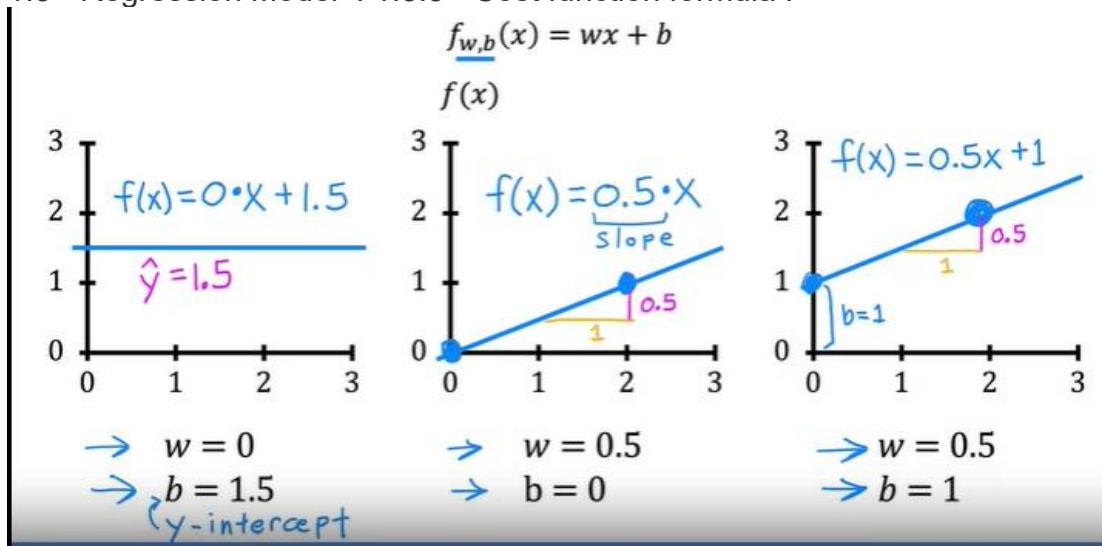
$(x^{(i)}, y^{(i)})$ = i^{th} training example

Index (1st, 2nd, 3rd ...)

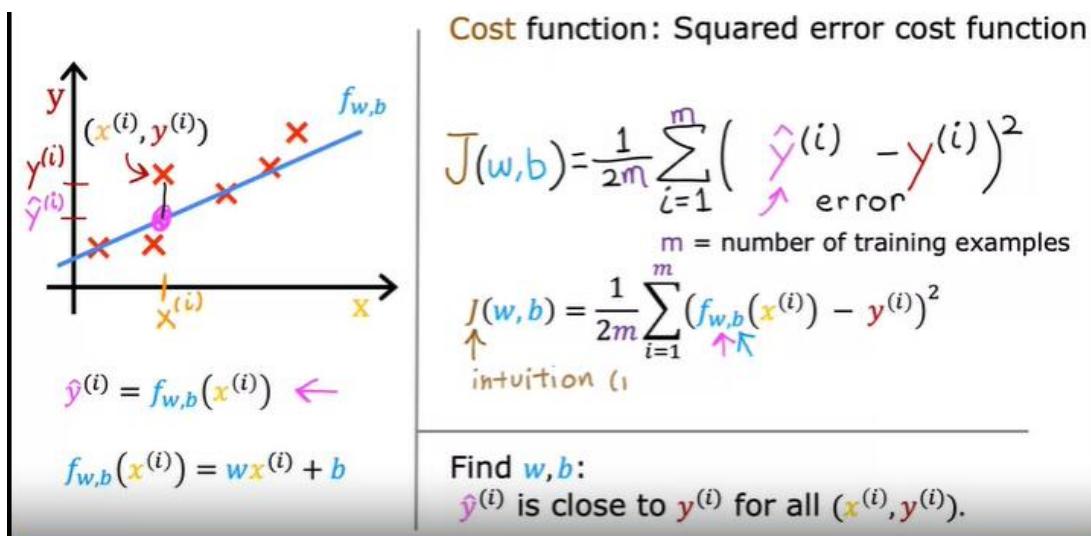
1.3 - Regression Model : 1.3.2 - Linear regression model part 2 :



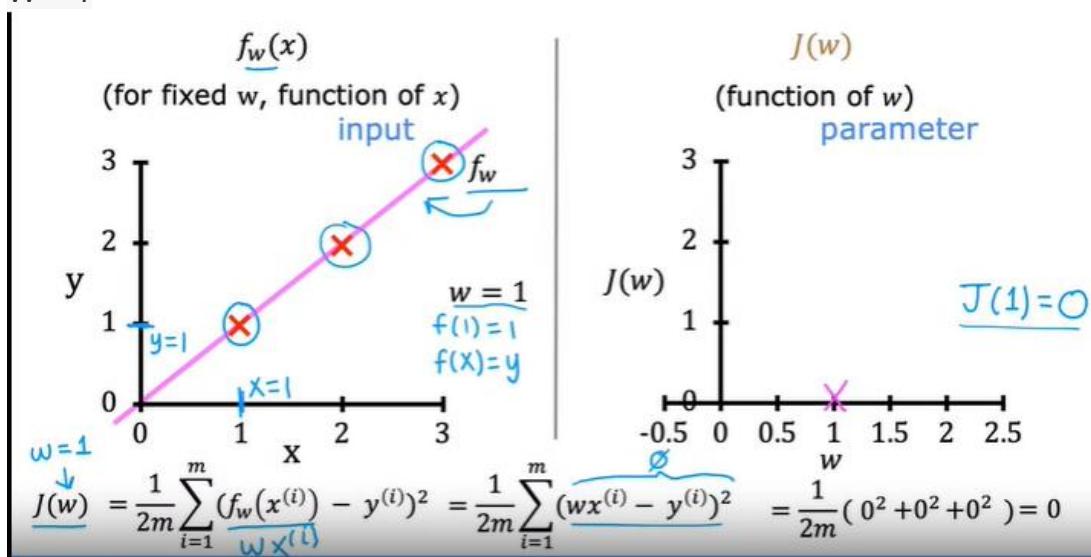
1.3 - Regression Model : 1.3.3 - Cost function formula :



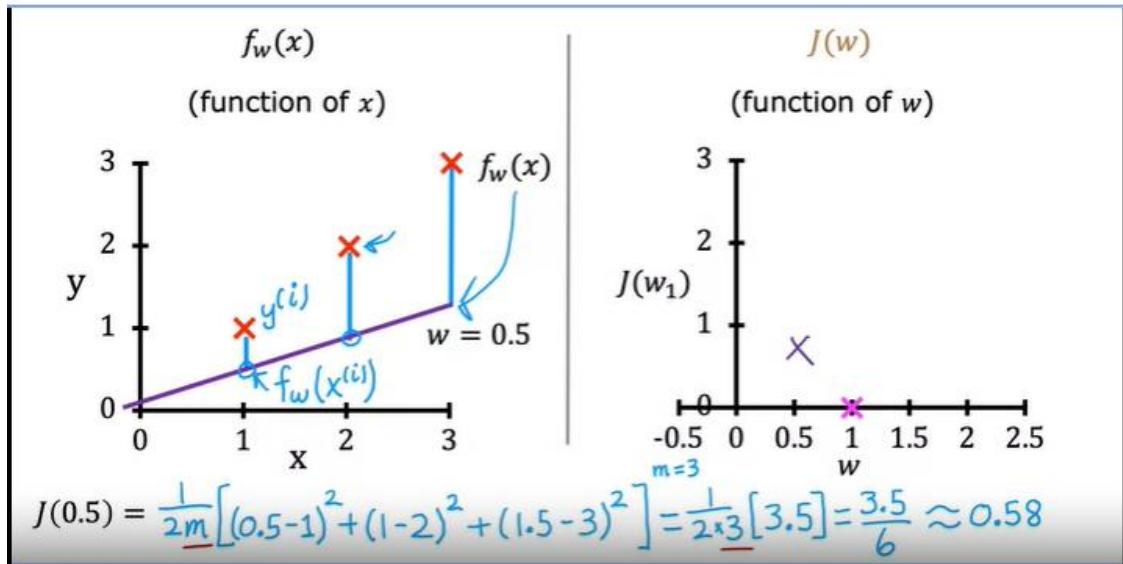
Cost function formula :



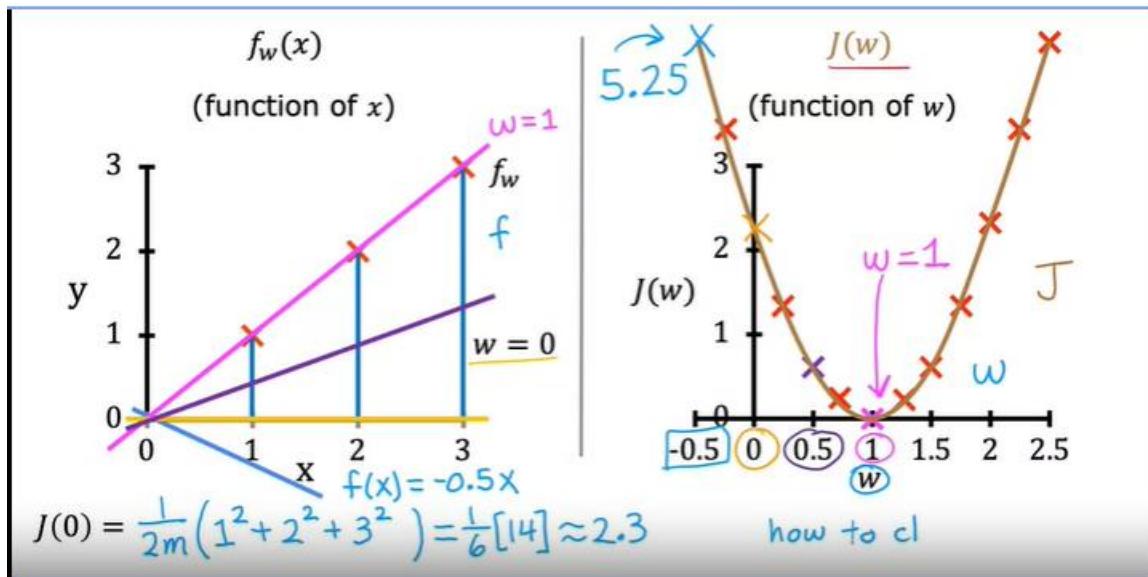
1.3 - Regression Model : 1.3.5 - Visualizing the cost function :

 $W = 1$ 

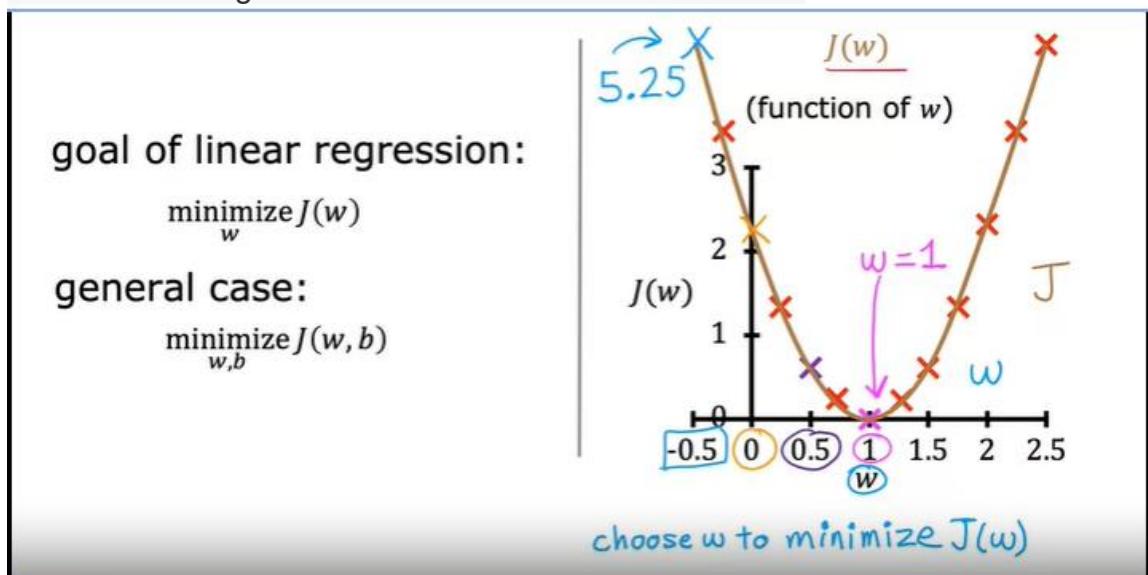
W = 0.5



W = 0, -0.5, ...



Goal of linear regression : To minimize the cost function



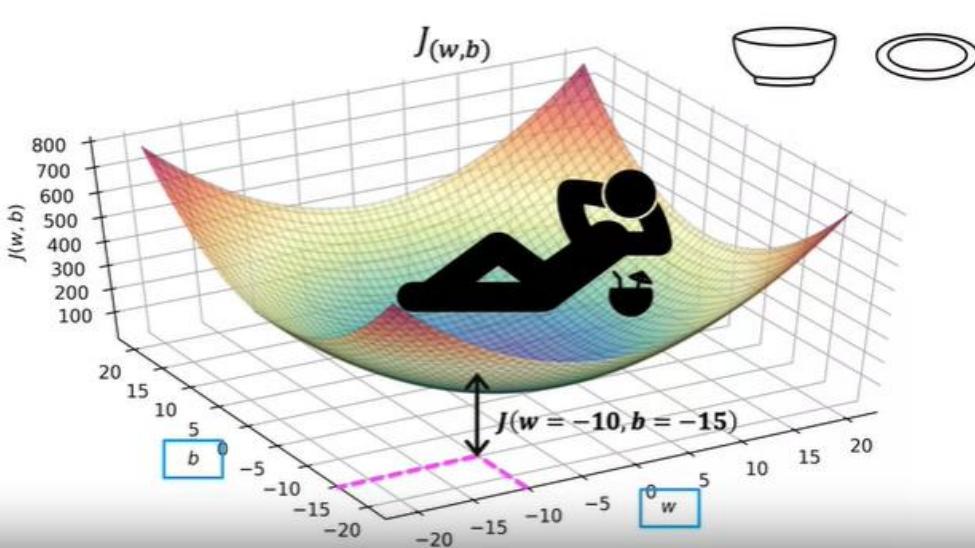
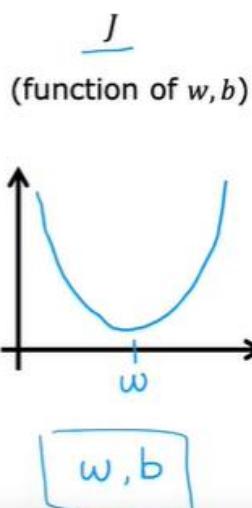
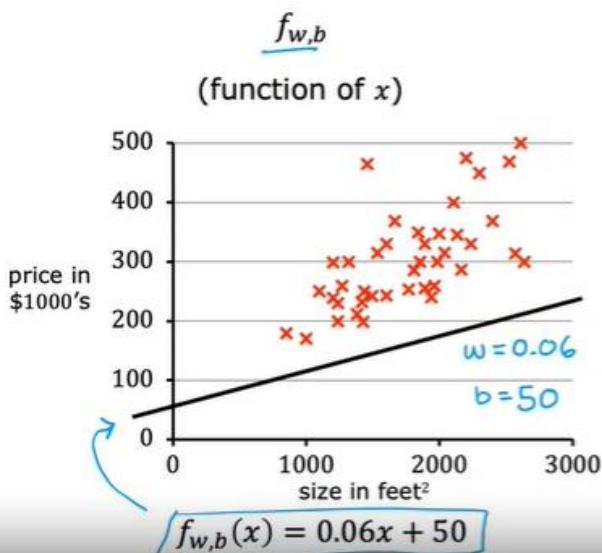
1.3 - Regression Model : 1.3.6 - Visualization examples :

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

Parameters w, b before: $b=0$

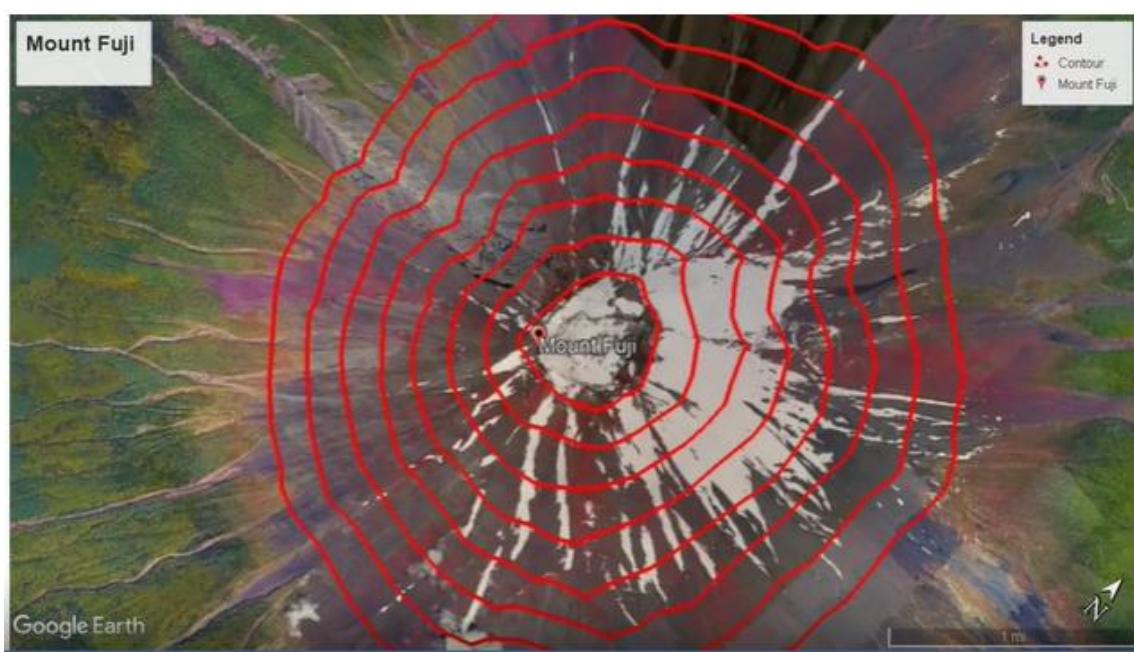
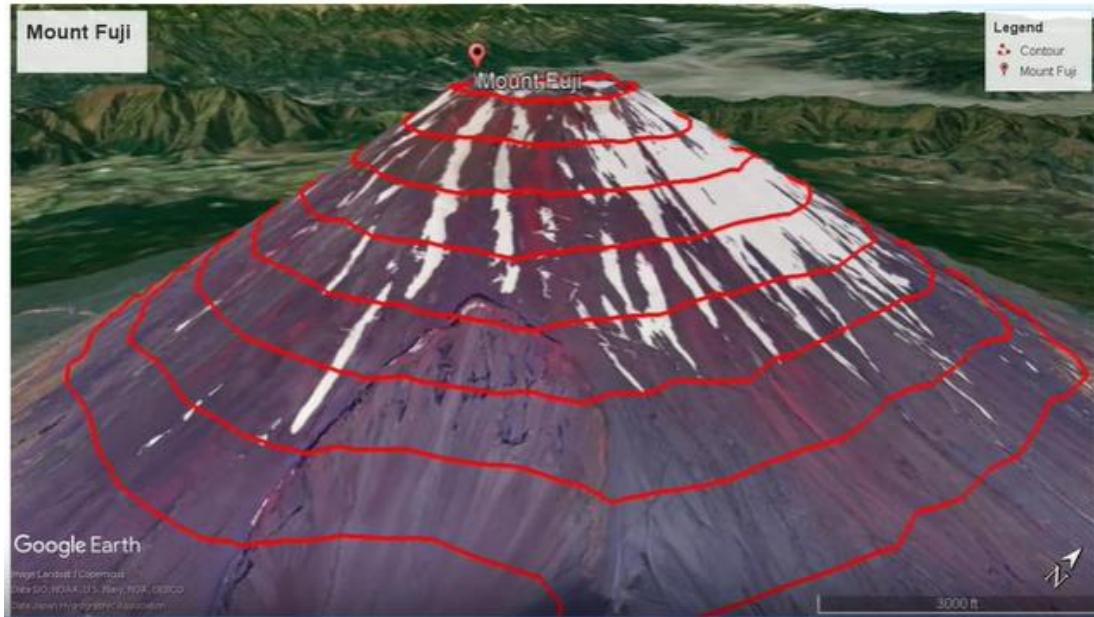
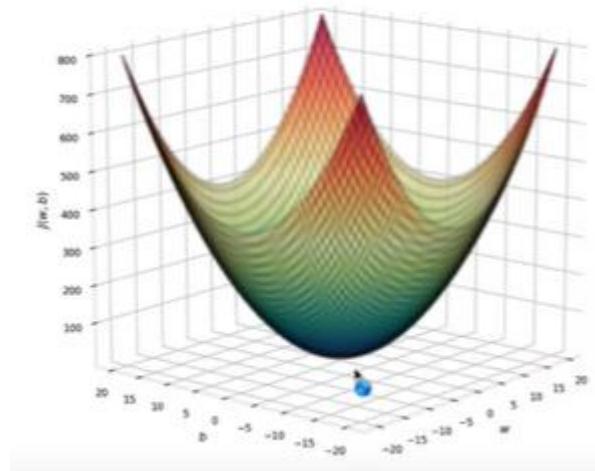
Cost Function $J(w, b) = \frac{1}{2m} \sum_{i=1}^m (f_{w,b}(x^{(i)}) - y^{(i)})^2$

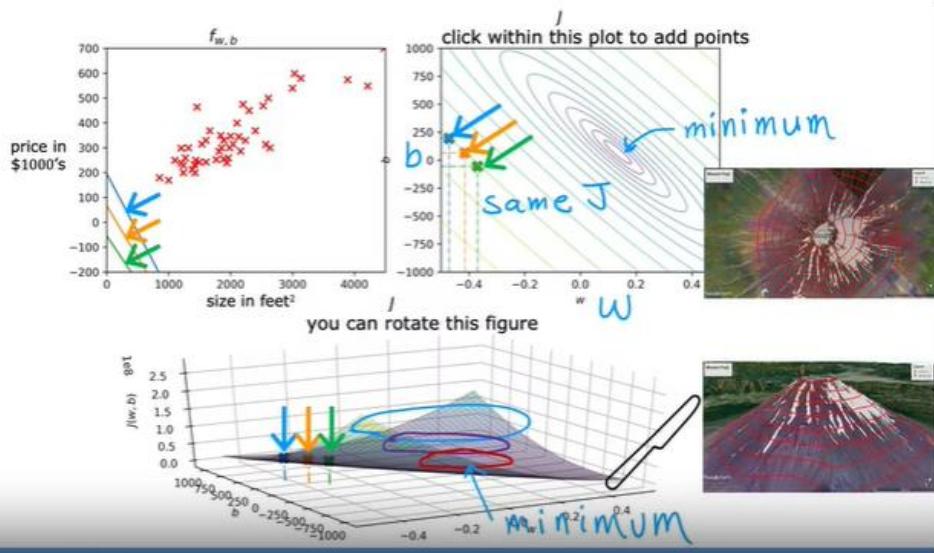
Objective $\underset{w,b}{\text{minimize}} J(w, b)$



3D surface plot

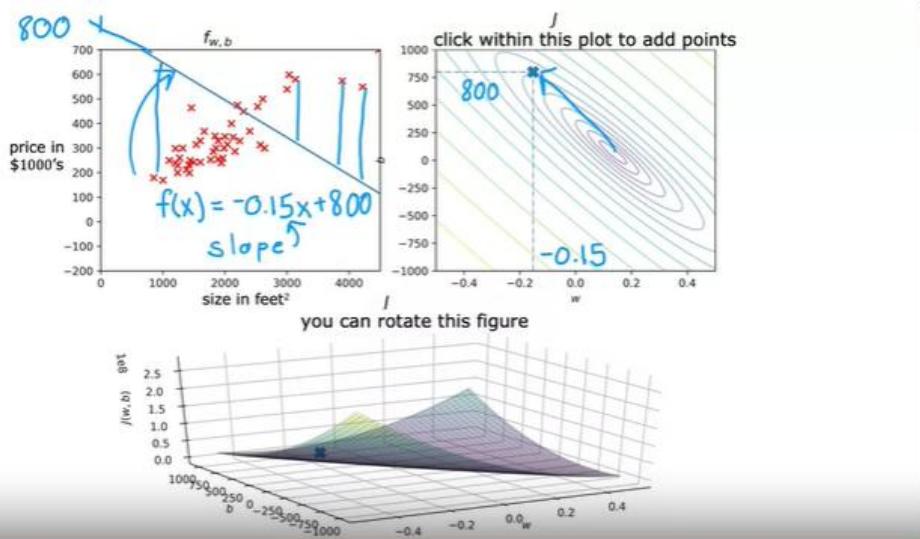
$J(w, b)$
[You can rotate this figure]



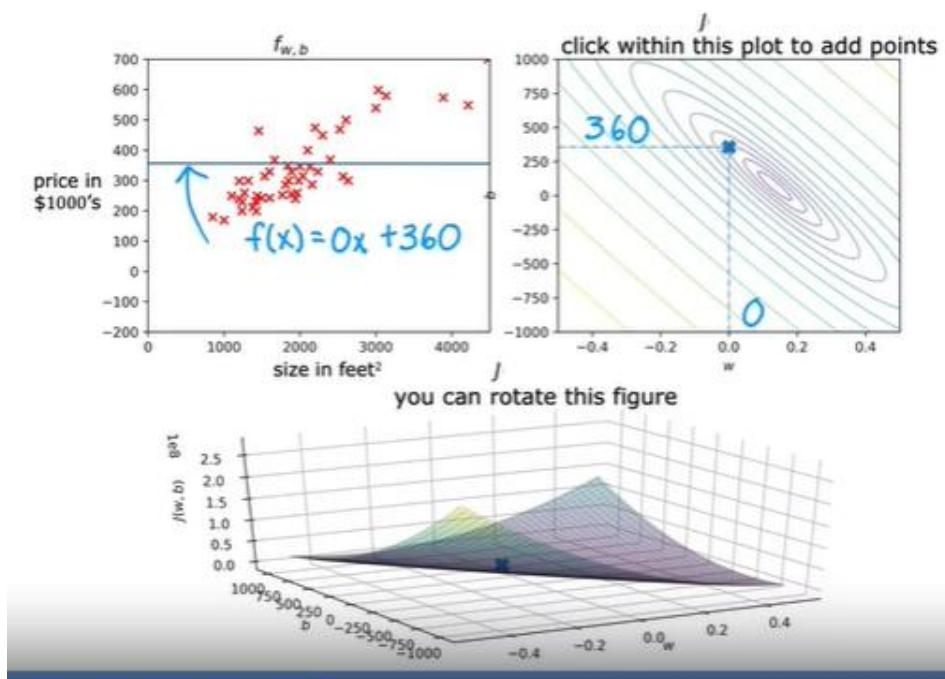


Visualization examples :

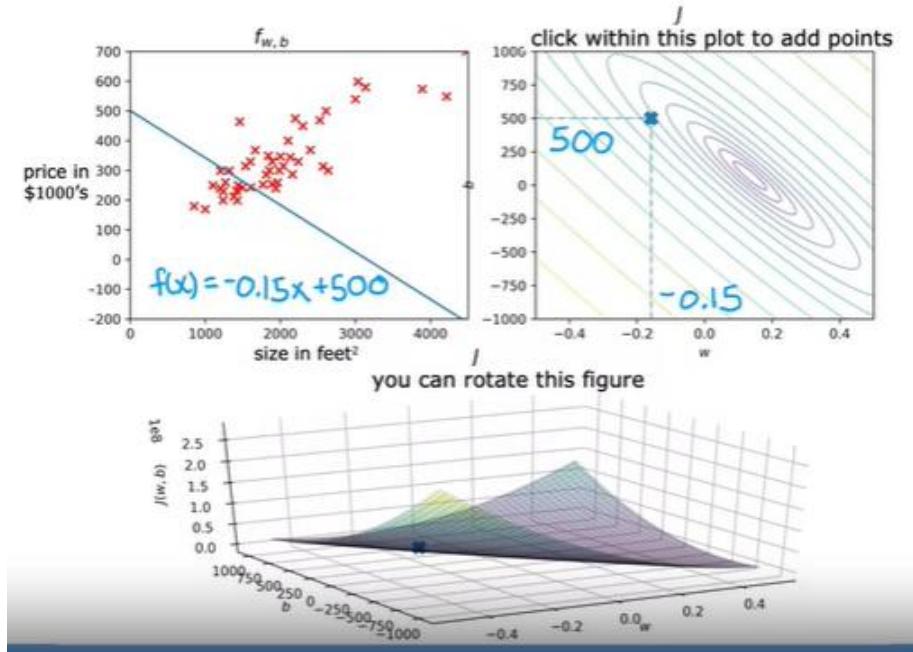
1) Bad fit



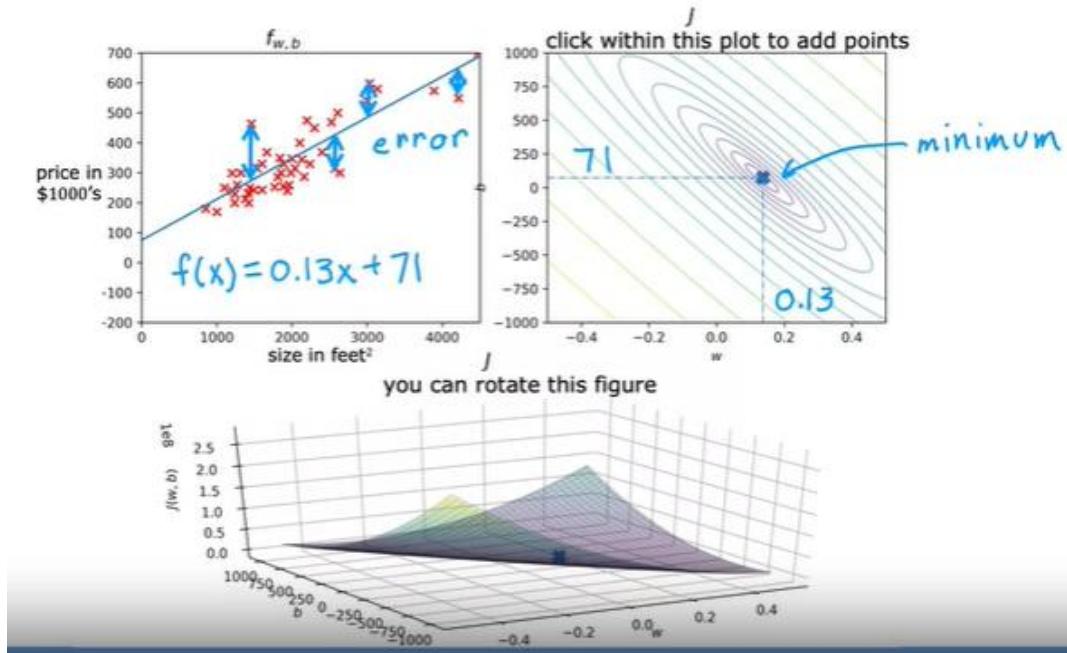
2) Not That bad fit



3) Not The Great fit



4) Pretty Good fit



1.4 - Train the model with gradient descent : 1.4.1 - Gradient descent :

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

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

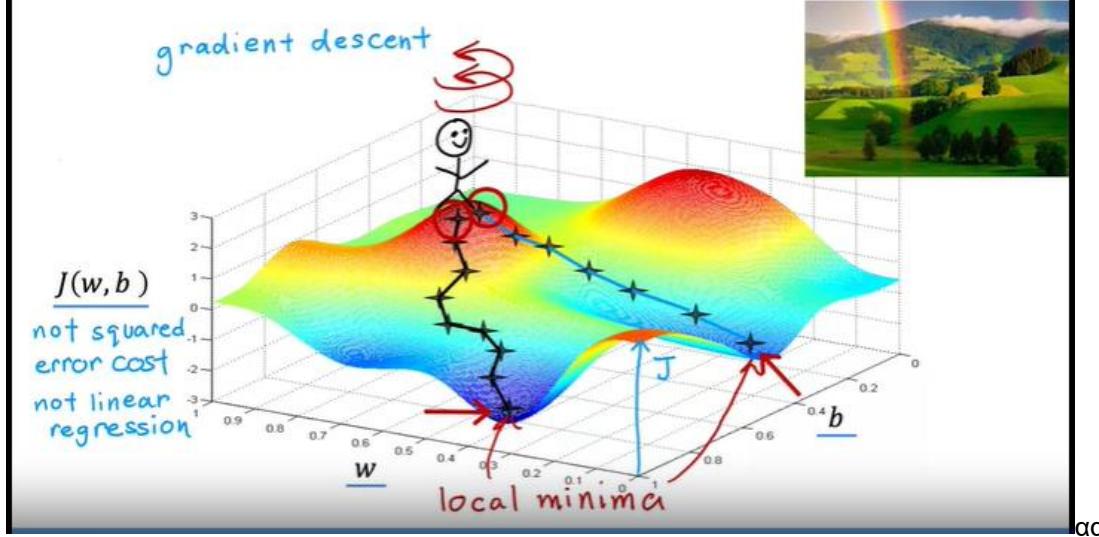
Outline:

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

Keep changing w, b to reduce $J(w, b)$ J not always

Until we settle at or near a minimum

may have >1 minimum

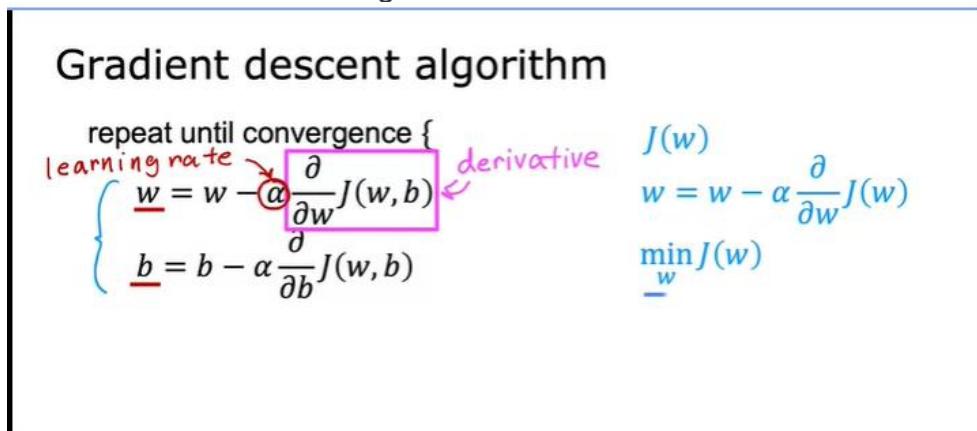


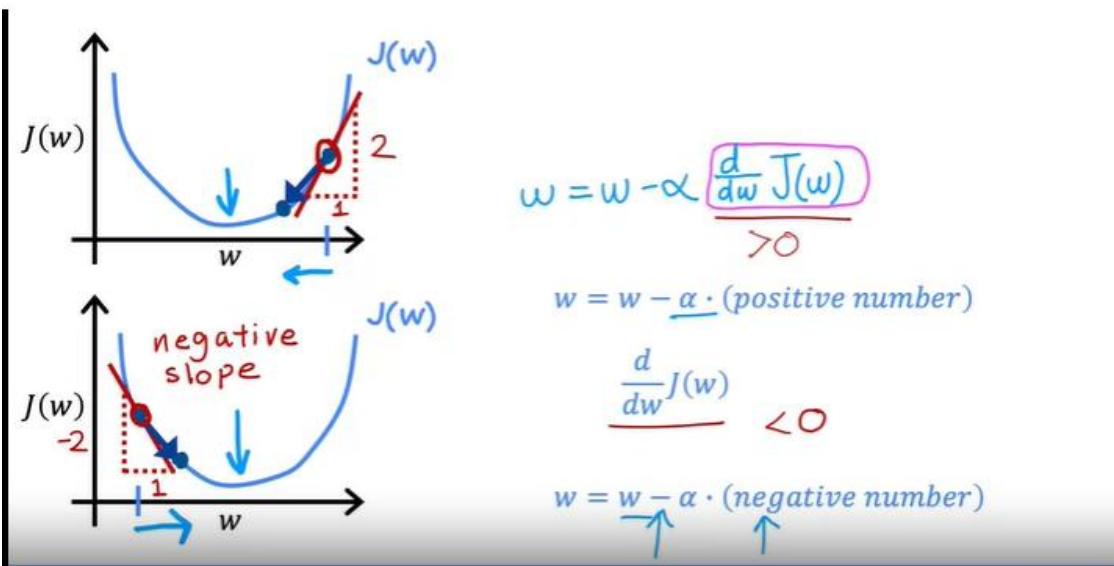
1.4 - Train the model with gradient descent : 1.4.2 - Implementing gradient descent :
 α = Learning rate, 0 to 1, (say 0.01) = size of the baby step

Partial derivative = $\frac{\partial}{\partial w} J(w, b)$ = Direction for the baby step

<u>Gradient descent algorithm</u>		Assignment	Truth assertion
Repeat until convergence		$a = c$	$a = c$
$w = w - \alpha \frac{\partial}{\partial w} J(w, b)$	Learning rate Derivative	$a = a + 1$	$a = a + 1$
$b = b - \alpha \frac{\partial}{\partial b} J(w, b)$	Simultaneously update w and b	Code	Math $a == c$
Correct: Simultaneous update		Incorrect	
$tmp_w = w - \alpha \frac{\partial}{\partial w} J(w, b)$		$tmp_w = w - \alpha \frac{\partial}{\partial w} J(w, b)$	
$tmp_b = b - \alpha \frac{\partial}{\partial b} J(w, b)$		$w = tmp_w$	
$w = tmp_w$		$tmp_b = b - \alpha \frac{\partial}{\partial b} J(w, b)$	
$b = tmp_b$		$b = tmp_b$	

1.4 - Train the model with gradient descent : 1.4.3 - Gradient descent intuition :





1.4 - Train the model with gradient descent : 1.4.4 - Learning rate :

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

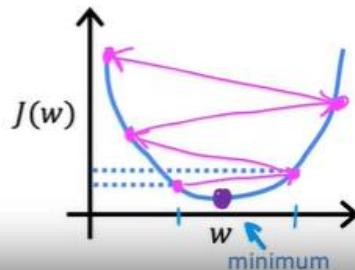
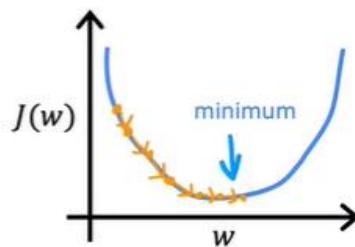
If α is too small...

Gradient descent may be slow.

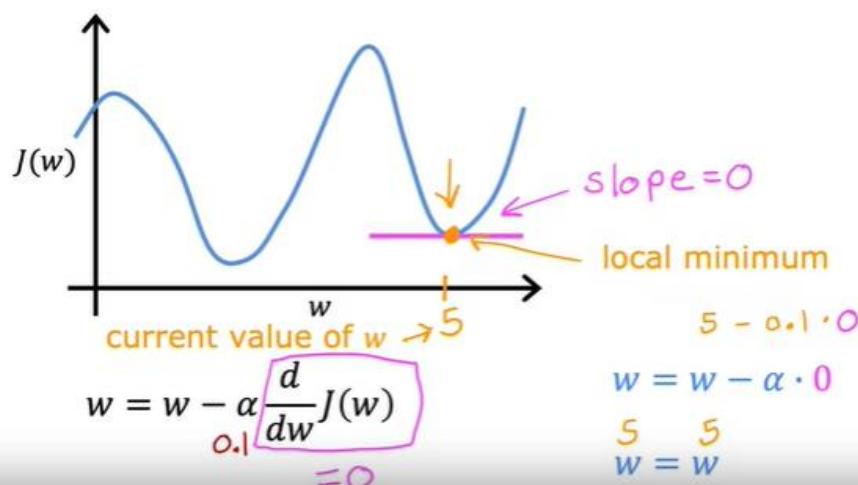
If α is too large...

Gradient descent may:

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



Learning rate : Tricky situation of 2 local minimas



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

1.4 - Train the model with gradient descent : 1.4.5 - Gradient descent for linear regression :

Linear regression model Cost function

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

Gradient descent algorithm

repeat until convergence {

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

Gradient descent for linear regression : with calculus (partial differentiation) steps

(Optional)

$$\begin{aligned} \frac{\partial}{\partial w} J(w, b) &= \frac{\partial}{\partial w} \frac{1}{2m} \sum_{i=1}^m (f_{w,b}(x^{(i)}) - y^{(i)})^2 = \frac{\partial}{\partial w} \frac{1}{2m} \sum_{i=1}^m (wx^{(i)} + b - y^{(i)})^2 \\ &= \cancel{\frac{1}{2m} \sum_{i=1}^m} (wx^{(i)} + b - y^{(i)}) \cancel{2x^{(i)}} = \boxed{\frac{1}{m} \sum_{i=1}^m (f_{w,b}(x^{(i)}) - y^{(i)}) x^{(i)}} \end{aligned}$$

$$\begin{aligned} \frac{\partial}{\partial b} J(w, b) &= \frac{\partial}{\partial b} \frac{1}{2m} \sum_{i=1}^m (f_{w,b}(x^{(i)}) - y^{(i)})^2 = \frac{\partial}{\partial b} \frac{1}{2m} \sum_{i=1}^m (wx^{(i)} + b - y^{(i)})^2 \\ &= \cancel{\frac{1}{2m} \sum_{i=1}^m} (wx^{(i)} + b - y^{(i)}) \cancel{2} = \boxed{\frac{1}{m} \sum_{i=1}^m (f_{w,b}(x^{(i)}) - y^{(i)})} \end{aligned}$$

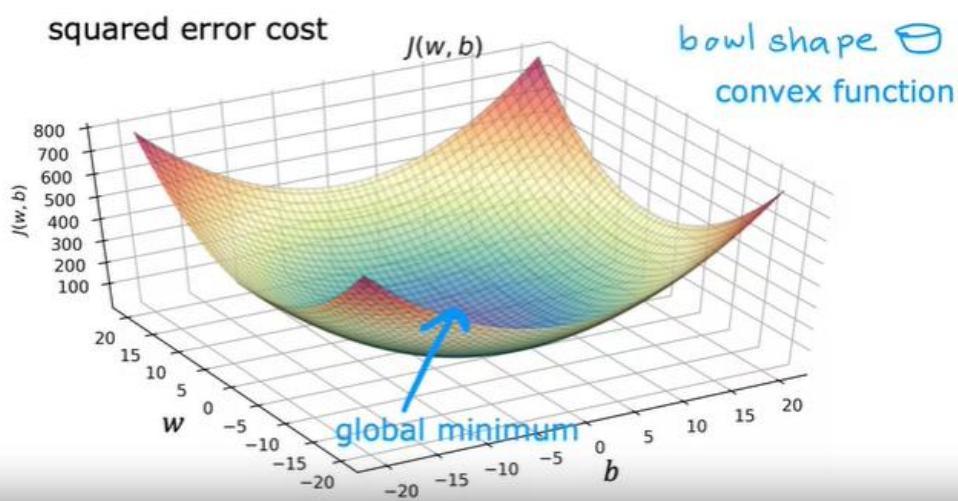
no $x^{(i)}$

Gradient descent for linear regression : Final compilation

Gradient descent algorithm

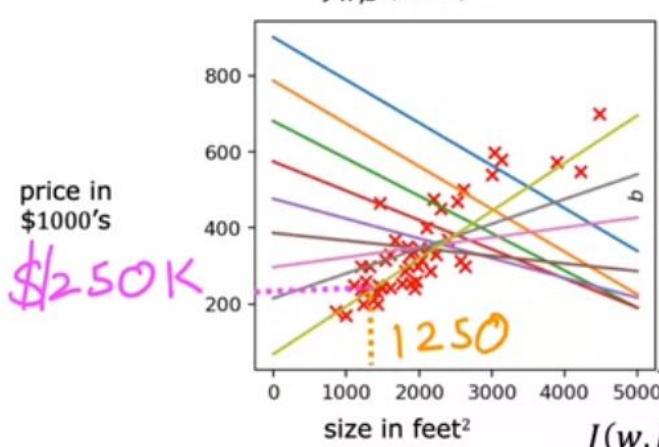
$$\begin{aligned}
 & \text{repeat until convergence} \{ \\
 & \quad w = w - \alpha \frac{1}{m} \sum_{i=1}^m (f_{w,b}(x^{(i)}) - y^{(i)}) x^{(i)} \\
 & \quad b = b - \alpha \frac{1}{m} \sum_{i=1}^m (f_{w,b}(x^{(i)}) - y^{(i)}) \\
 & \} \\
 & \quad \frac{\partial}{\partial w} J(w, b) \\
 & \quad f_{w,b}(x^{(i)}) = w x^{(i)} + b
 \end{aligned}$$

Update w and b simultaneously

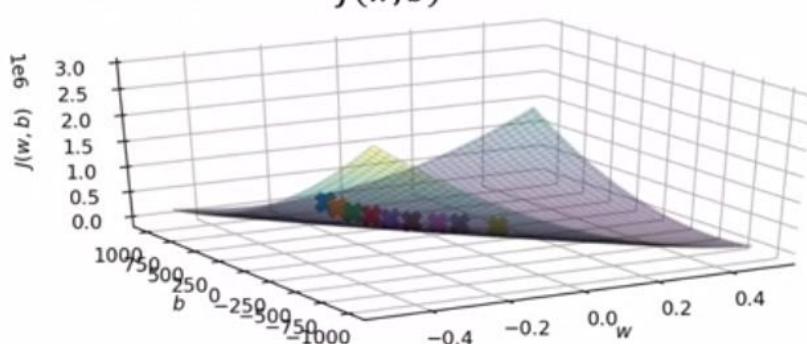
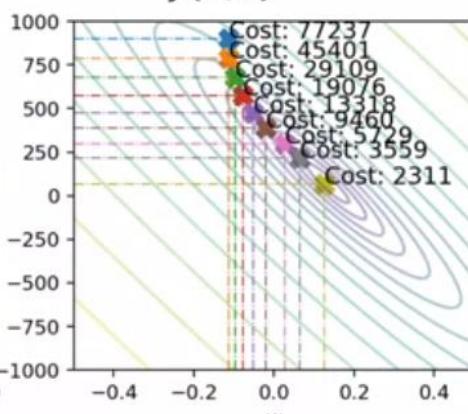


1.4 - Train the model with gradient descent : 1.4.6 - Running gradient descent :

$f_{w,b}(\text{size})$



$J(w, b)$



Batch gradient descent :

"Batch" gradient descent

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

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

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

other gradient descent: subsets

coursera

jupyter C1_W2_Lab03_Gradient_Descent_Soln Last Checkpoint: Last Monday at 4:42 AM (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 O

Optional Lab: Gradient Descent for Linear Regression

Linear Regression with One Variable

Gradient Descent algorithm

Repeat until convergence

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

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

learning rate derivative

Simultaneous update

Contour plot of cost function, use with path of gradient descent

Gradient Descent

"Batch" gradient descent

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

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

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

Goals

In this lab, you will:

- automate the process of optimizing w and b using gradient descent.

Week (Part) 2 : Regression with multiple input variables

1.1 - Overview of Machine Learning

- 1.1.1 - Welcome to machine learning!
- 1.1.2 - Applications of machine learning

1.2 - Supervised vs. Unsupervised Machine Learning

- 1.2.1 - What is machine learning?
- 1.2.2 - Supervised learning part 1
- 1.2.3 - Supervised learning part 2
- 1.2.4 - Unsupervised learning part 1
- 1.2.5 - Unsupervised learning part 2
- 1.2.6 - Jupyter Notebooks

Lab: Python and Jupyter Notebooks

1.3 - Regression Model

- 1.3.1 - Linear regression model part 1
- 1.3.2 - Linear regression model part 2
- Lab: Optional lab: Model representation
- 1.3.3 - Cost function formula
- 1.3.4 - Cost function intuition
- 1.3.5 - Visualizing the cost function
- 1.3.6 - Visualization examples

Lab: Optional lab: Cost function

1.4 - Train the model with gradient descent

- 1.4.1 - Gradient descent
- 1.4.2 - Implementing gradient descent
- 1.4.3 - Gradient descent intuition
- 1.4.4 - Learning rate
- 1.4.5 - Gradient descent for linear regression
- 1.4.6 - Running gradient descent

Lab: Optional lab: Gradient descent

1.3 - Regression Model : 1.3.6 - Visualization examples :

1.2 - Supervised vs. Unsupervised Machine Learning : 1.2.1 - What is machine learning? :

it's the science of getting computers to learn without being explicitly programmed.

machine learning is the science of getting computers to learn without being explicitly programmed.

Learning Objectives

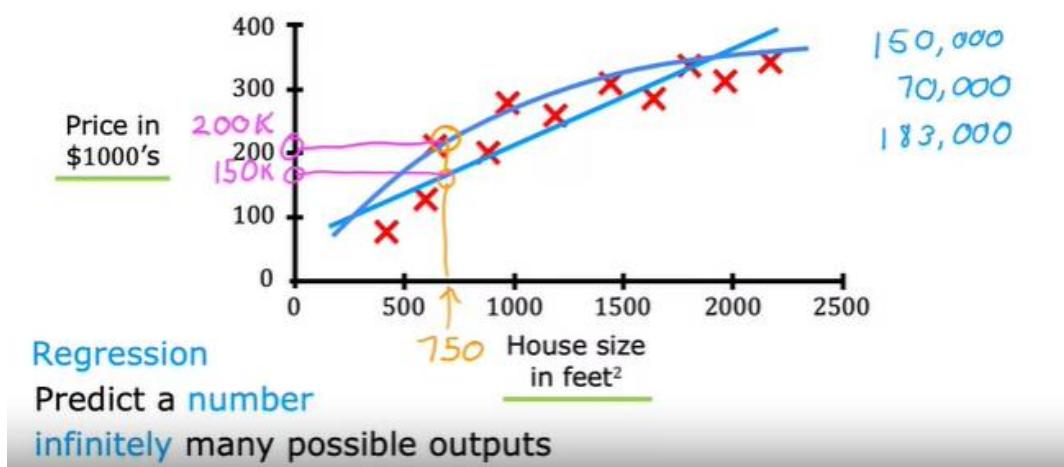
- Define machine learning
- Define supervised learning
- Define unsupervised learning
- Write and run Python code in Jupyter Notebooks
- Define a regression model
- Implement and visualize a cost function
- Implement gradient descent
- Optimize a regression model using gradient descent

supervised learning :

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

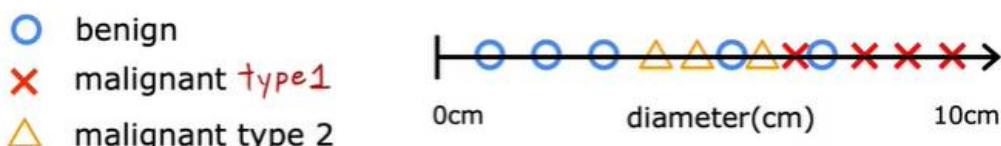
1.2.2 - supervised learning :part 1- Regression

Regression: Housing price prediction



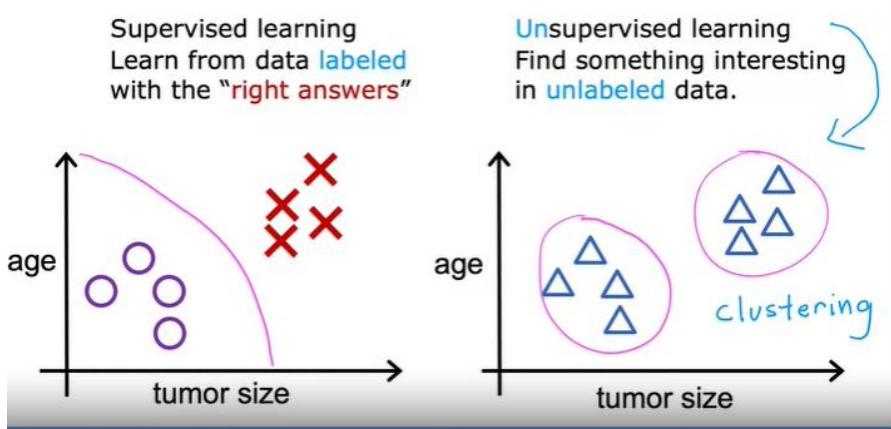
1.2.3 - supervised learning :part 2- Classification

Classification: Breast cancer detection



Classification
predict categories cat dog benign malignant 0, 1, 2
small number of possible outputs

1.2.4 - Unsupervised learning :part 1-

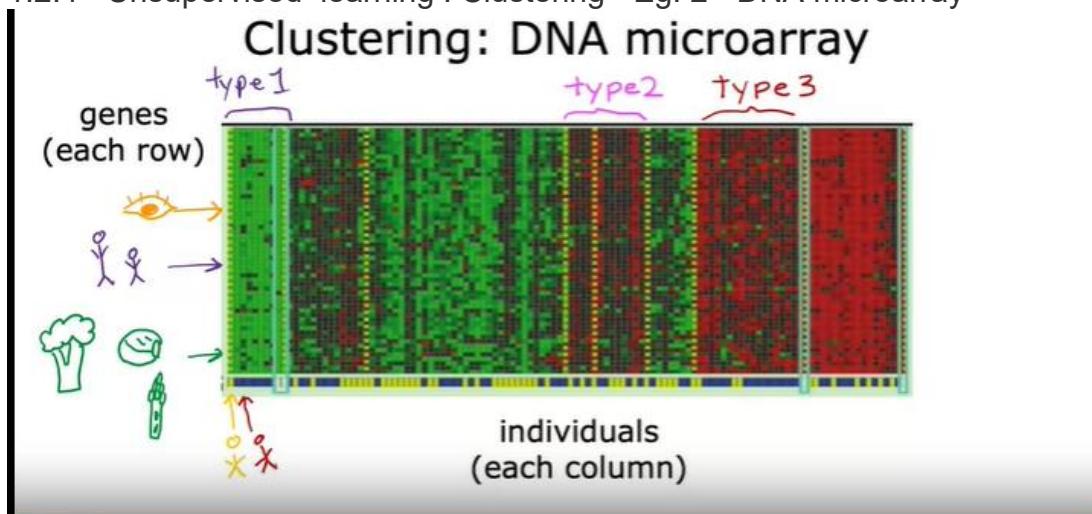


1.2.4 - Unsupervised learning : Clustering - Eg. 1 - Google News

Clustering: Google news

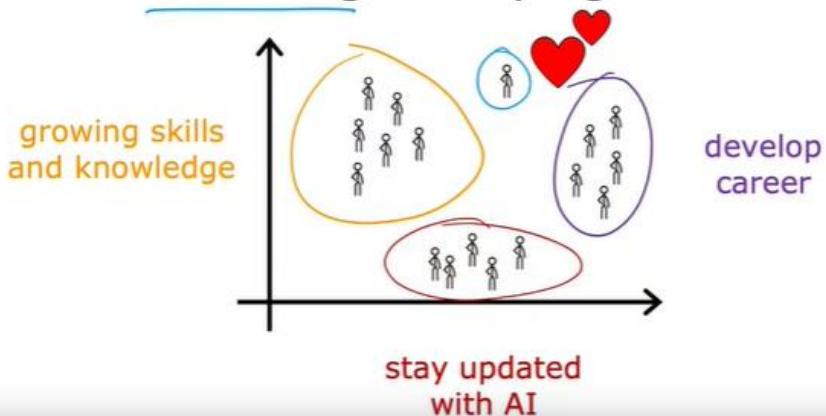
- the algorithm has to figure out on his own without supervision, what are the clusters of news articles today. So that's why this clustering algorithm, is a type of **unsupervised learning algorithm**.
- I don't know what the different types of people are but can you automatically find structure into data. And automatically figure out what are the major types of individuals, since we're not given the algorithm the right answer for the examples in advance. This is **unsupervised learning**

1.2.4 - Unsupervised learning : Clustering - Eg. 2 - DNA microarray



1.2.4 - Unsupervised learning : Clustering - Eg. 3 - Grouping Customers

Clustering: Grouping customers



1.2.5 - 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.

Clustering

Group similar data points together.

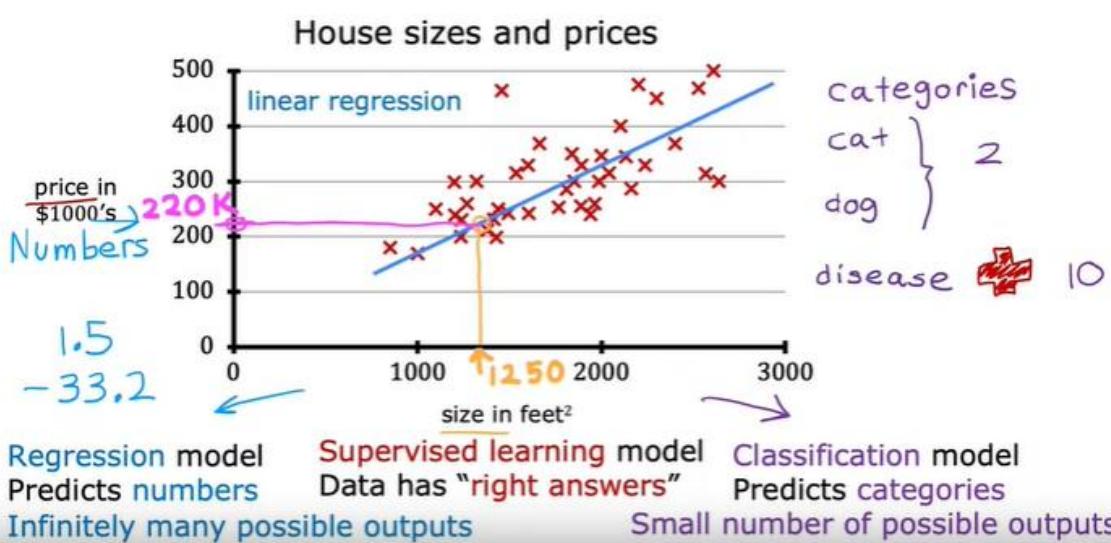
Dimensionality reduction

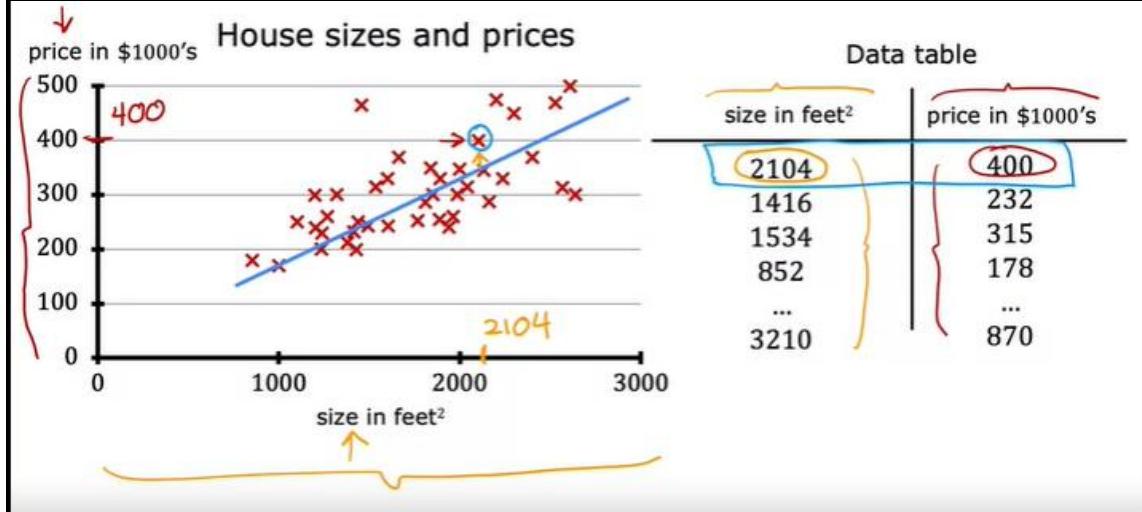
Compress data using fewer numbers.

Anomaly detection

Find unusual data points.

1.3 - Regression Model : 1.3.1 - Linear regression model part 1 :





Terminology

Training Data used to train the model

set: x size in feet² y price in \$1000's

(1)	2104	400
(2)	1416	232
(3)	1534	315
(4)	852	178
...
(47)	3210	870

$m = 47$

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

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

Notation:

`x = "input" variable
feature`

y = “output” variable
“target” variable

m = number of training examples

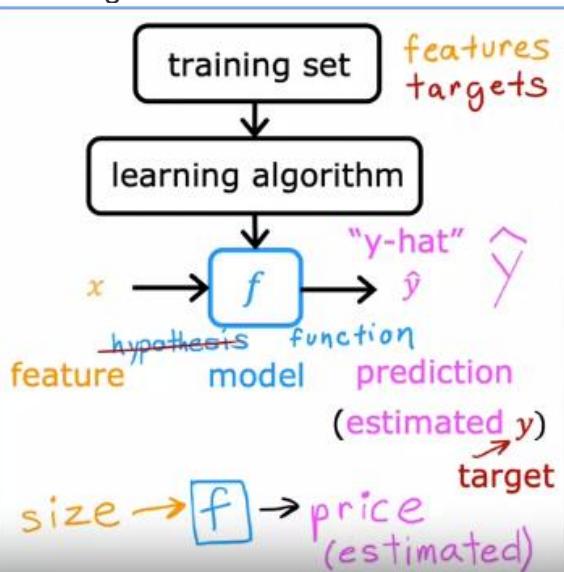
(x, y) = single training example

$(x^{(i)}, y^{(i)})$

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

index (1st, 2nd, 3rd ...)

1.3 - Regression Model : 1.3.2 - Linear regression model part 2 :



How to represent f ?

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

$f(x)$

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

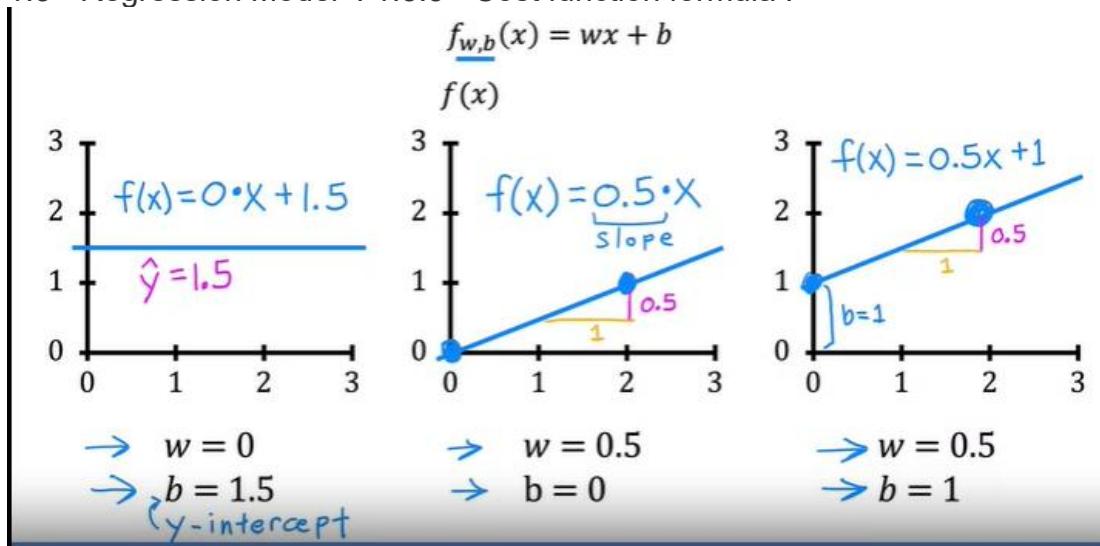
$$f(x) =$$

Linear regression with one variable.

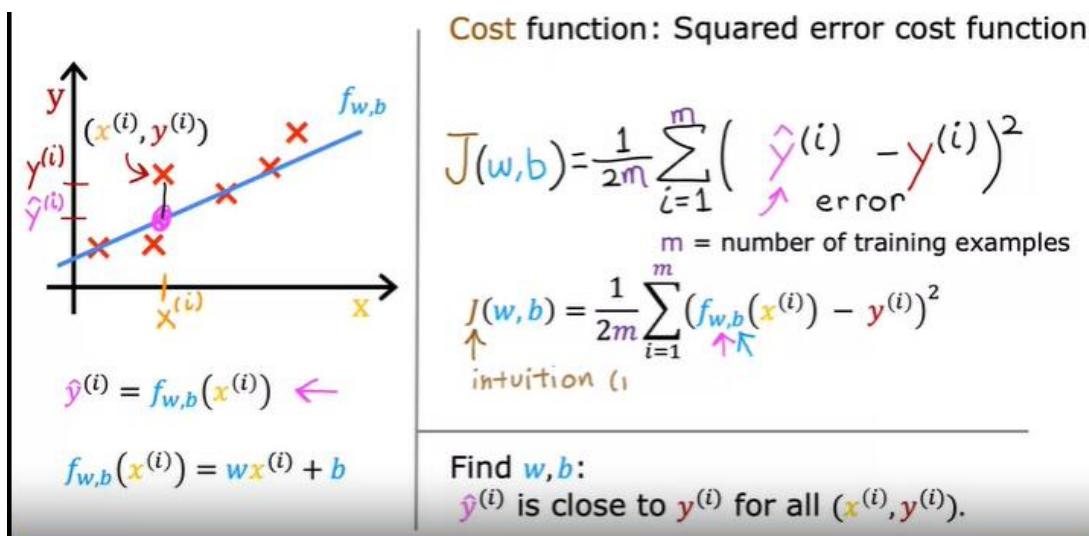
Linear Regression with one variable

Univariate linear regression.

1.3 - Regression Model : 1.3.3 - Cost function formula :

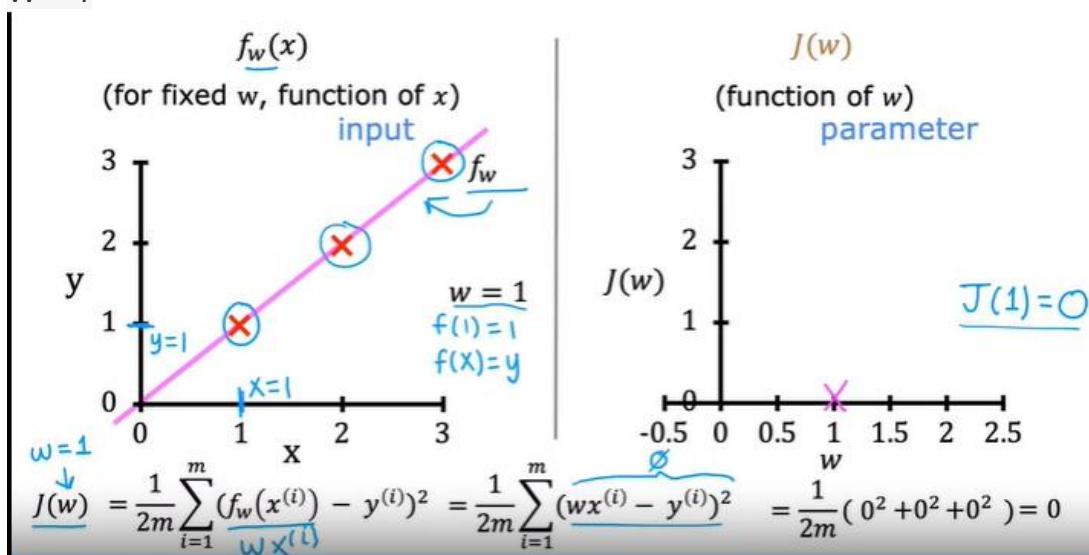


Cost function formula :

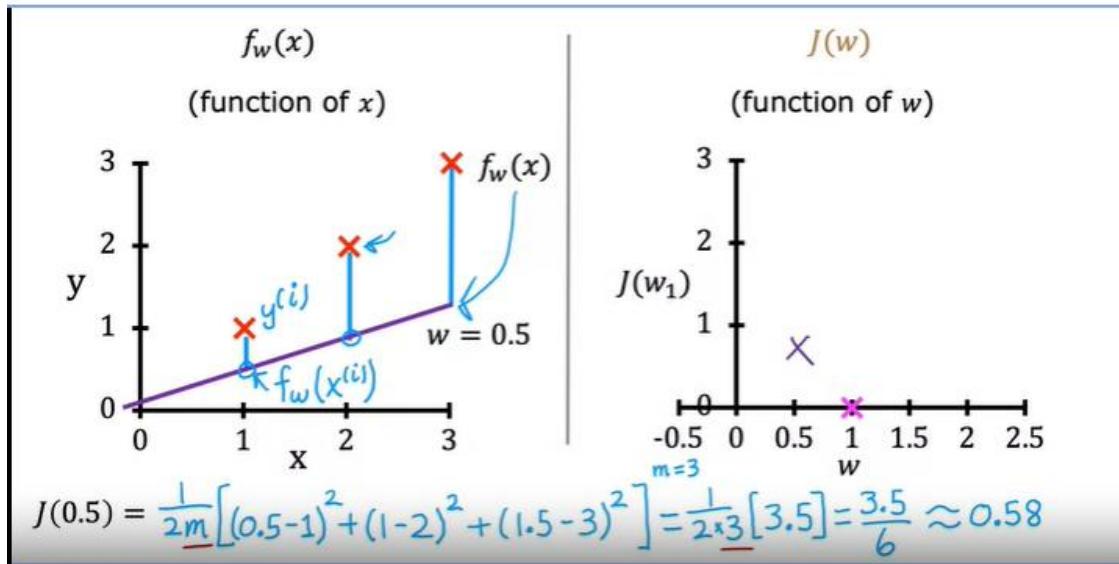


1.3 - Regression Model : 1.3.5 - Visualizing the cost function :

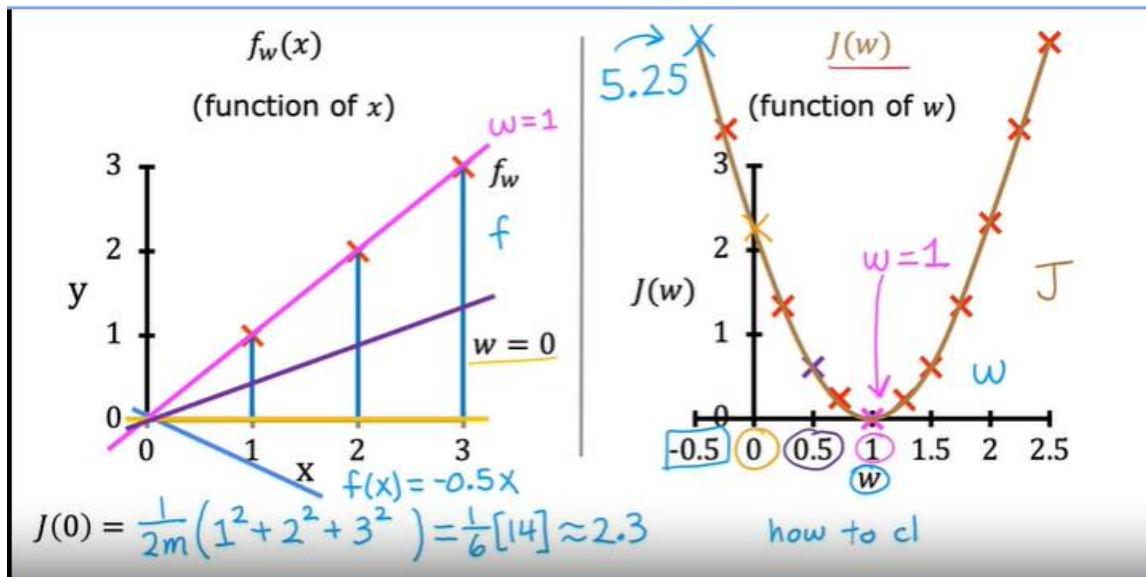
W = 1



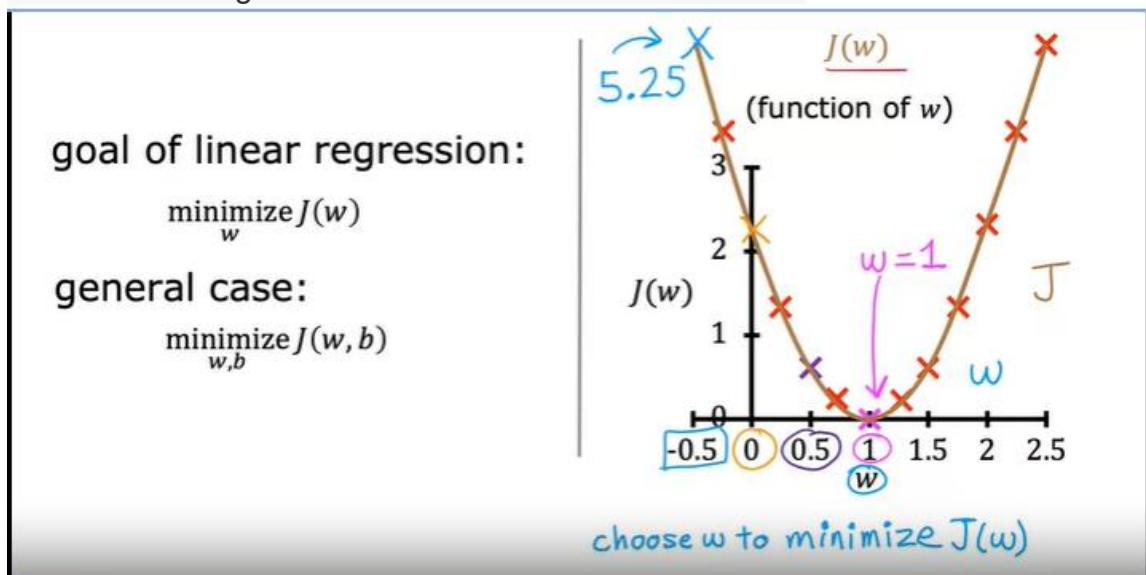
W = 0.5



W = 0, -0.5, ...



Goal of linear regression : To minimize the cost function



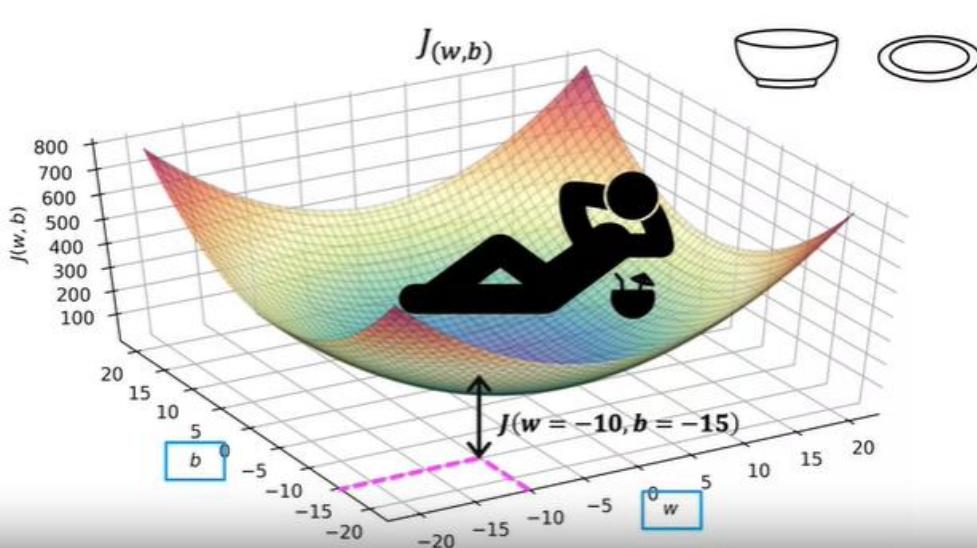
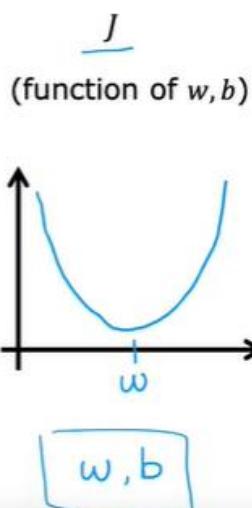
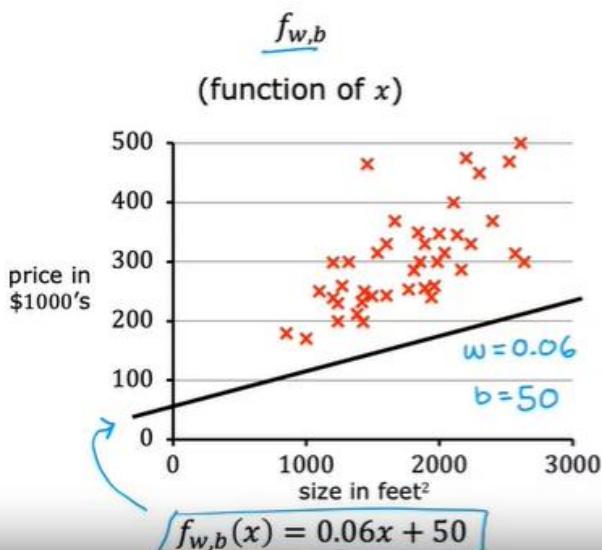
1.3 - Regression Model : 1.3.6 - Visualization examples :

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

Parameters w, b before: $b=0$

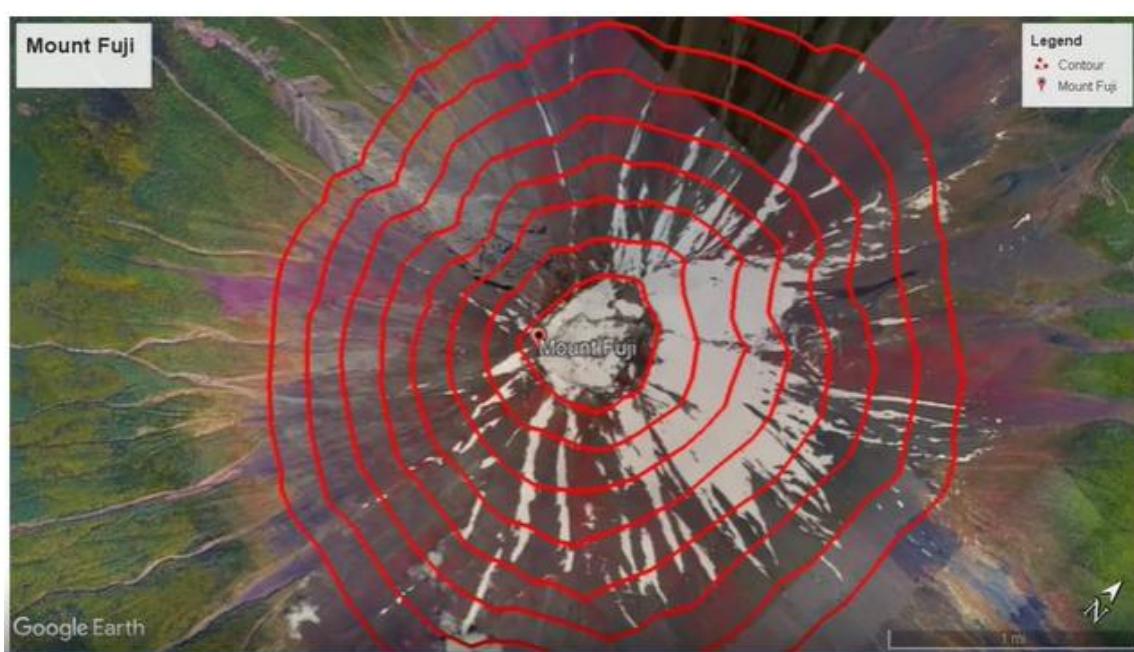
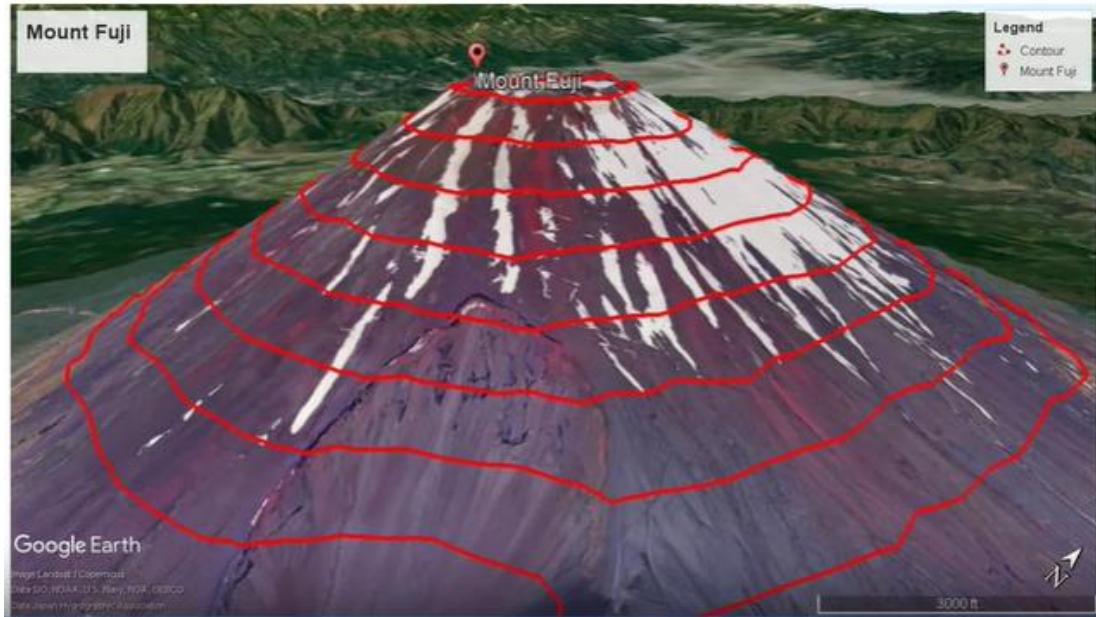
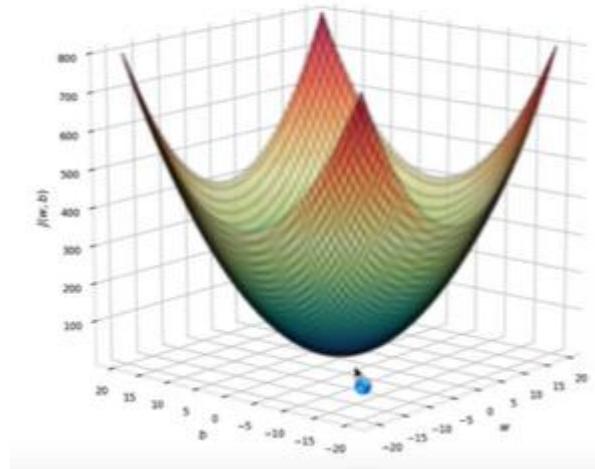
Cost Function $J(w, b) = \frac{1}{2m} \sum_{i=1}^m (f_{w,b}(x^{(i)}) - y^{(i)})^2$

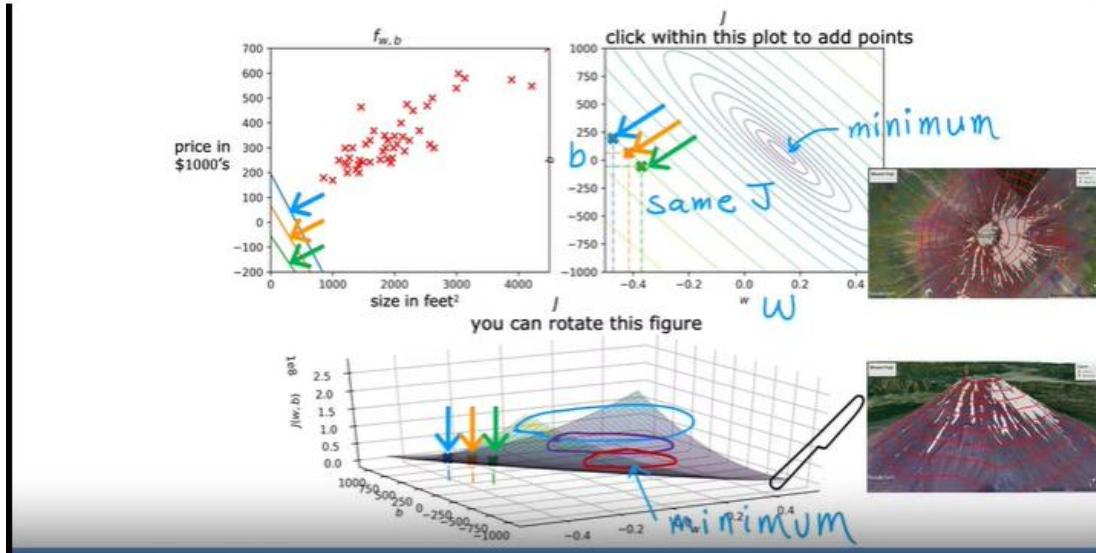
Objective $\underset{w,b}{\text{minimize}} J(w, b)$



3D surface plot

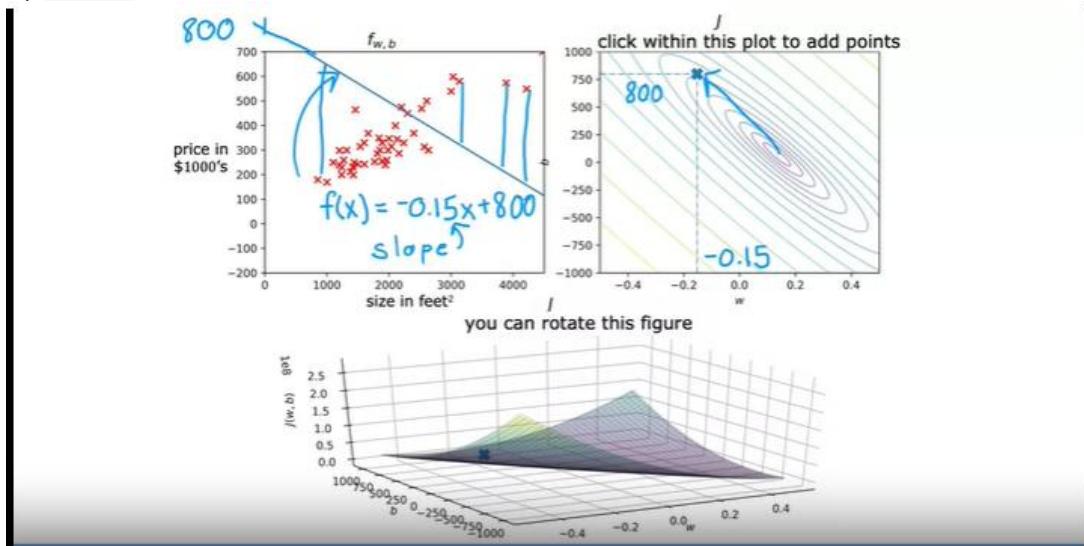
$J(w, b)$
[You can rotate this figure]



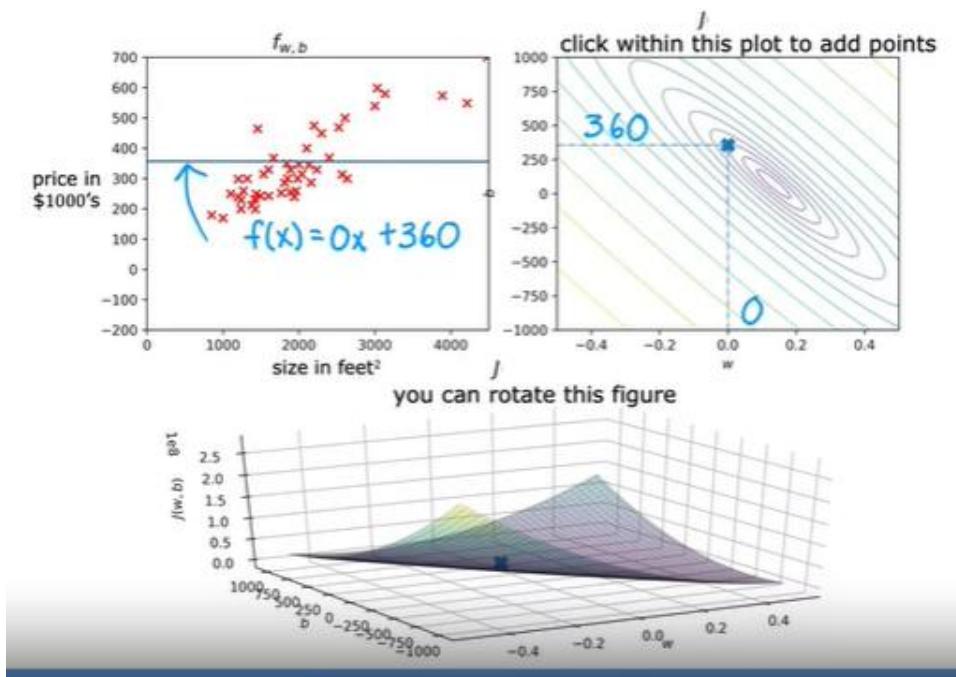


1.3 - Regression Model : 1.3.6 - Visualization examples :

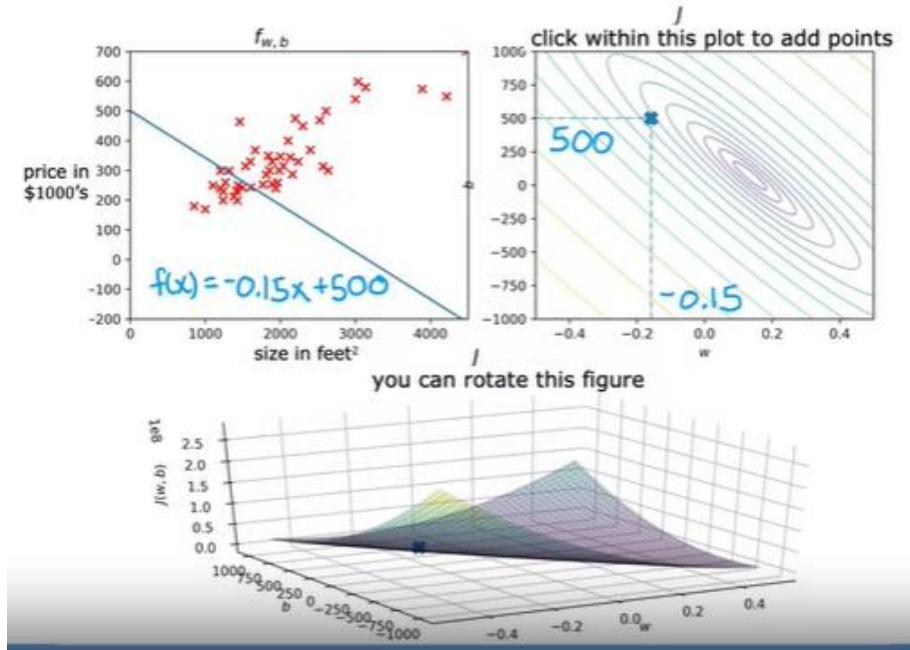
1) Bad fit



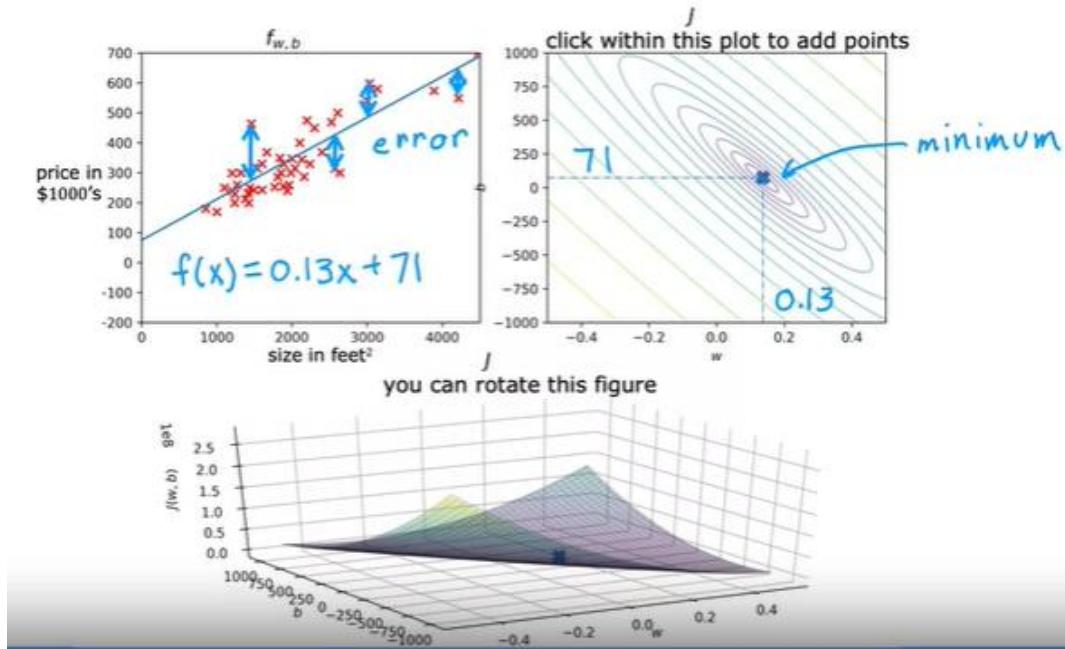
2) Not That bad fit



3) Not The Great fit



4) Pretty Good fit



1.4 - Train the model with gradient descent : 1.4.1 - Gradient descent :

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

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

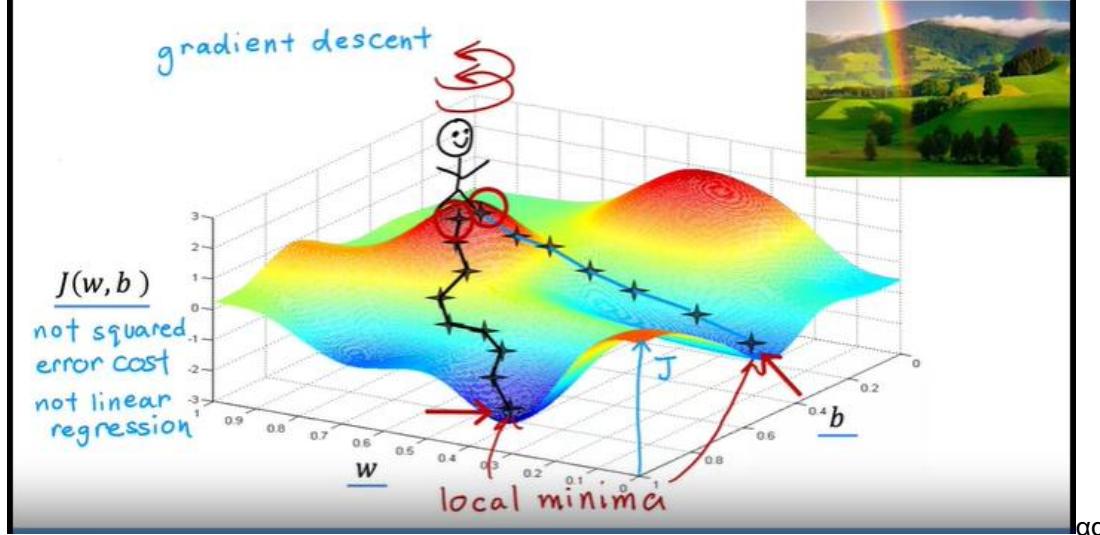
Outline:

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

Keep changing w, b to reduce $J(w, b)$ J not always

Until we settle at or near a minimum

may have >1 minimum

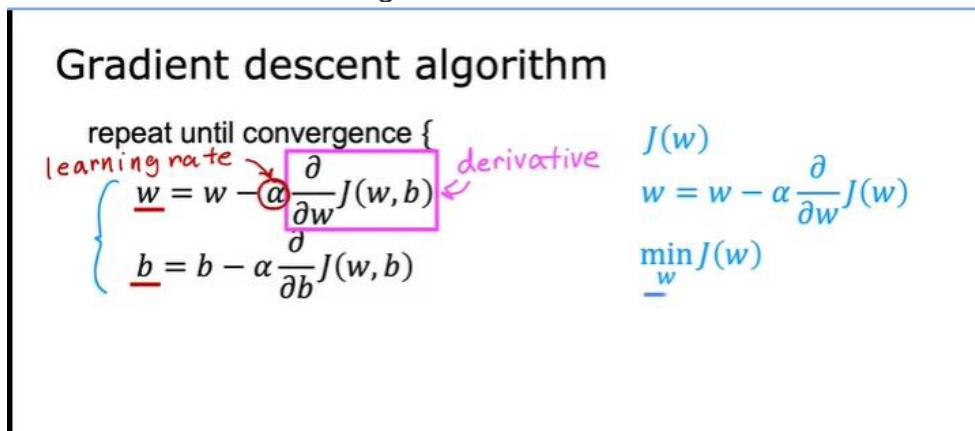


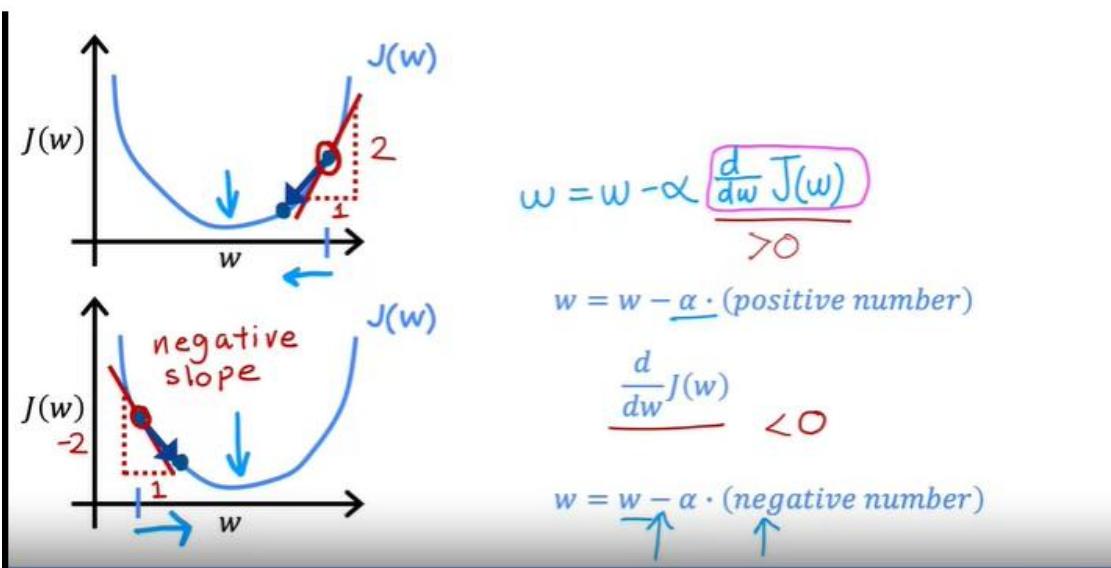
1.4 - Train the model with gradient descent : 1.4.2 - Implementing gradient descent :
 α = Learning rate, 0 to 1, (say 0.01) = size of the baby step

Partial derivative = $\frac{\partial}{\partial w} J(w, b)$ = Direction for the baby step

<u>Gradient descent algorithm</u>		Assignment	Truth assertion
Repeat until convergence		$a = c$	$a = c$
$w = w - \alpha \frac{\partial}{\partial w} J(w, b)$	Learning rate Derivative	$a = a + 1$	$a = a + 1$
$b = b - \alpha \frac{\partial}{\partial b} J(w, b)$	Simultaneously update w and b	Code	Math $a == c$
Correct: Simultaneous update		Incorrect	
$tmp_w = w - \alpha \frac{\partial}{\partial w} J(w, b)$		$tmp_w = w - \alpha \frac{\partial}{\partial w} J(w, b)$	
$tmp_b = b - \alpha \frac{\partial}{\partial b} J(w, b)$		$w = tmp_w$	
$w = tmp_w$		$tmp_b = b - \alpha \frac{\partial}{\partial b} J(w, b)$	
$b = tmp_b$		$b = tmp_b$	

1.4 - Train the model with gradient descent : 1.4.3 - Gradient descent intuition :





1.4 - Train the model with gradient descent : 1.4.4 - Learning rate :

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

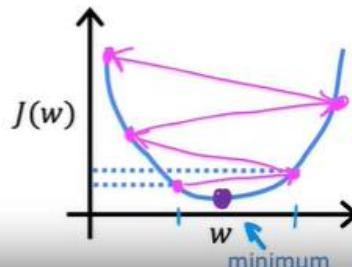
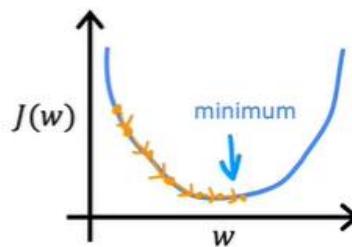
If α is too small...

Gradient descent may be slow.

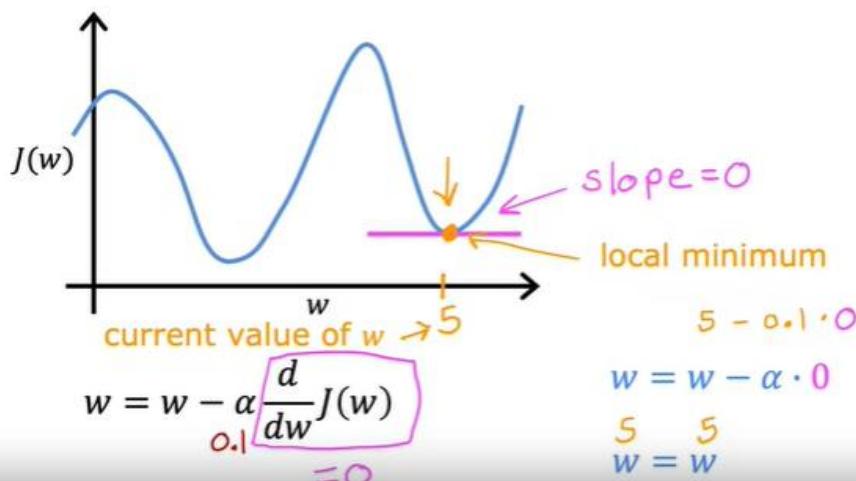
If α is too large...

Gradient descent may:

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



Learning rate : Tricky situation of 2 local minimas



Can reach local minimum with fixed learning rate α

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

smaller
not as large
large

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

Can reach minimum without decreasing learning rate α

1.4 - Train the model with gradient descent : 1.4.5 - Gradient descent for linear regression :

Linear regression model Cost function

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

Gradient descent algorithm

repeat until convergence {

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

}

Gradient descent for linear regression : with calculus (partial differentiation) steps

(Optional)

$$\begin{aligned} \frac{\partial}{\partial w} J(w, b) &= \frac{\partial}{\partial w} \frac{1}{2m} \sum_{i=1}^m (f_{w,b}(x^{(i)}) - y^{(i)})^2 = \frac{\partial}{\partial w} \frac{1}{2m} \sum_{i=1}^m (wx^{(i)} + b - y^{(i)})^2 \\ &= \cancel{\frac{1}{2m} \sum_{i=1}^m} (wx^{(i)} + b - y^{(i)}) \cancel{2x^{(i)}} = \boxed{\frac{1}{m} \sum_{i=1}^m (f_{w,b}(x^{(i)}) - y^{(i)}) x^{(i)}} \end{aligned}$$

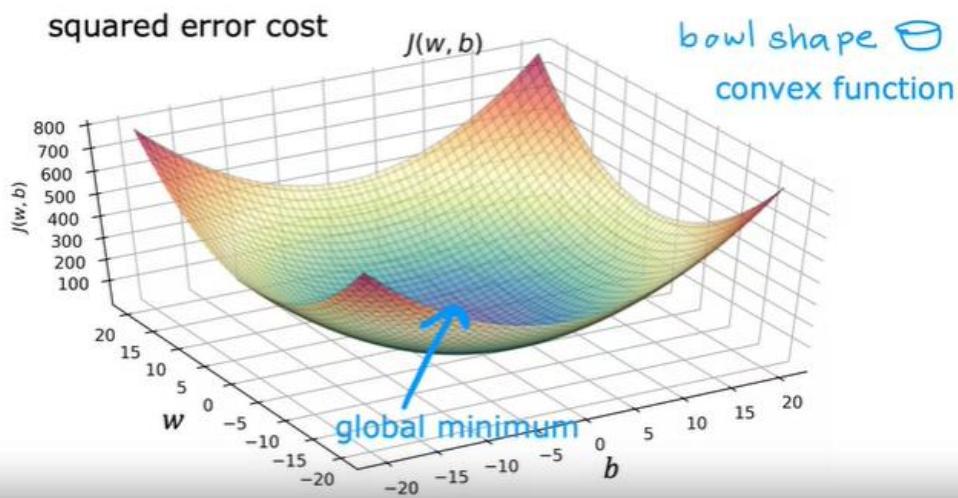
$$\begin{aligned} \frac{\partial}{\partial b} J(w, b) &= \frac{\partial}{\partial b} \frac{1}{2m} \sum_{i=1}^m (f_{w,b}(x^{(i)}) - y^{(i)})^2 = \frac{\partial}{\partial b} \frac{1}{2m} \sum_{i=1}^m (wx^{(i)} + b - y^{(i)})^2 \\ &= \cancel{\frac{1}{2m} \sum_{i=1}^m} (wx^{(i)} + b - y^{(i)}) \cancel{2} = \boxed{\frac{1}{m} \sum_{i=1}^m (f_{w,b}(x^{(i)}) - y^{(i)})} \end{aligned}$$

no $x^{(i)}$

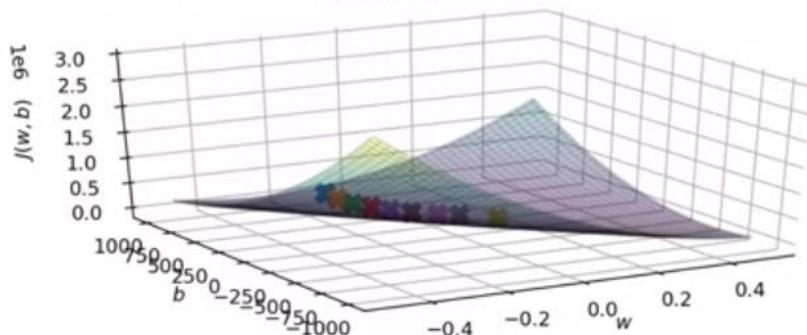
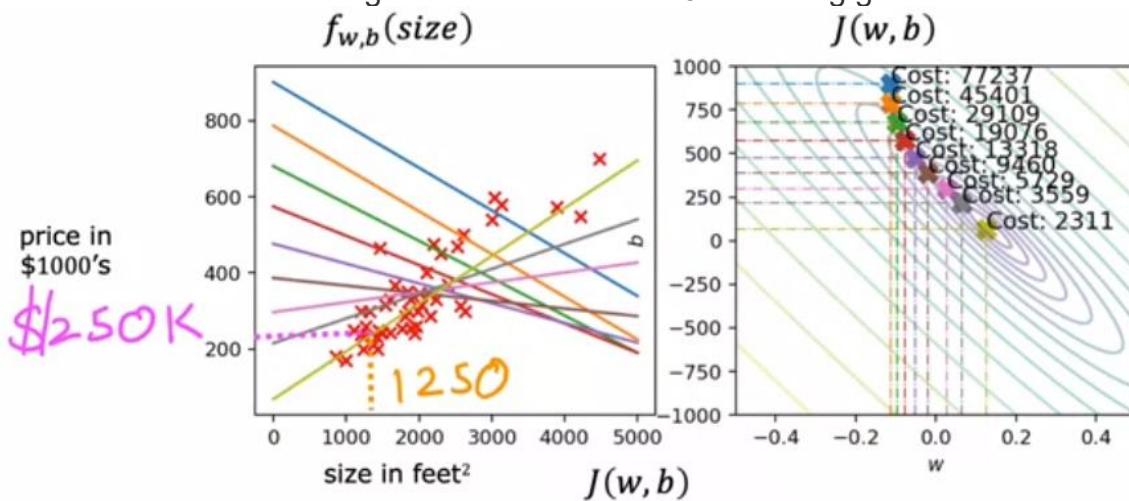
Gradient descent for linear regression : Final compilation

Gradient descent algorithm

$$\begin{aligned}
 & \text{repeat until convergence} \{ \\
 & \quad w = w - \alpha \frac{1}{m} \sum_{i=1}^m (f_{w,b}(x^{(i)}) - y^{(i)}) x^{(i)} \\
 & \quad b = b - \alpha \frac{1}{m} \sum_{i=1}^m (f_{w,b}(x^{(i)}) - y^{(i)}) \\
 & \} \\
 & \quad \frac{\partial}{\partial w} J(w, b) \\
 & \quad \frac{\partial}{\partial b} J(w, b) \\
 & \quad f_{w,b}(x^{(i)}) = w x^{(i)} + b
 \end{aligned}$$



1.4 - Train the model with gradient descent : 1.4.6 - Running gradient descent :



Batch gradient descent :

“Batch” gradient descent

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

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

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

other gradient descent: subsets

coursera

jupyter C1_W2_Lab03_Gradient_Descent_Soln Last Checkpoint: Last Monday at 4:42 AM (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 O

Optional Lab: Gradient Descent for Linear Regression

Linear Regression with One Variable

Gradient Descent algorithm

Repeat until convergence

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

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

learning rate derivative

Simultaneous update

Contour plot of cost function, use with path of gradient descent

Gradient Descent

“Batch” gradient descent

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

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

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

Goals

In this lab, you will:

- automate the process of optimizing w and b using gradient descent.

3.1 - Classification with logistic regression

3.1.1 - Motivations 9m

3.1.2 - Logistic regression 9m

3.1.3 - Decision boundary 10m

3.2 - Cost function for logistic regression 11m

3.2.1 - Cost function for logistic regression

3.2.2 - Simplified Cost Function for Logistic Regression 5m

3.3 - Gradient descent for logistic regression

3.3.1 - Gradient Descent Implementation 6m

3.4 - The problem of overfitting 11m

3.4.1 - The problem of overfitting

3.4.2 - Addressing overfitting 8m

3.4.3 - Cost function with regularization 9m

3.4.4 - Regularized linear regression 8m

3.4.5 - Regularized logistic regression 5m

Andrew Ng and Fei-Fei Li on Human-Centered AI 41m

1 reading

Acknowledgments 2m

4 practice exercises

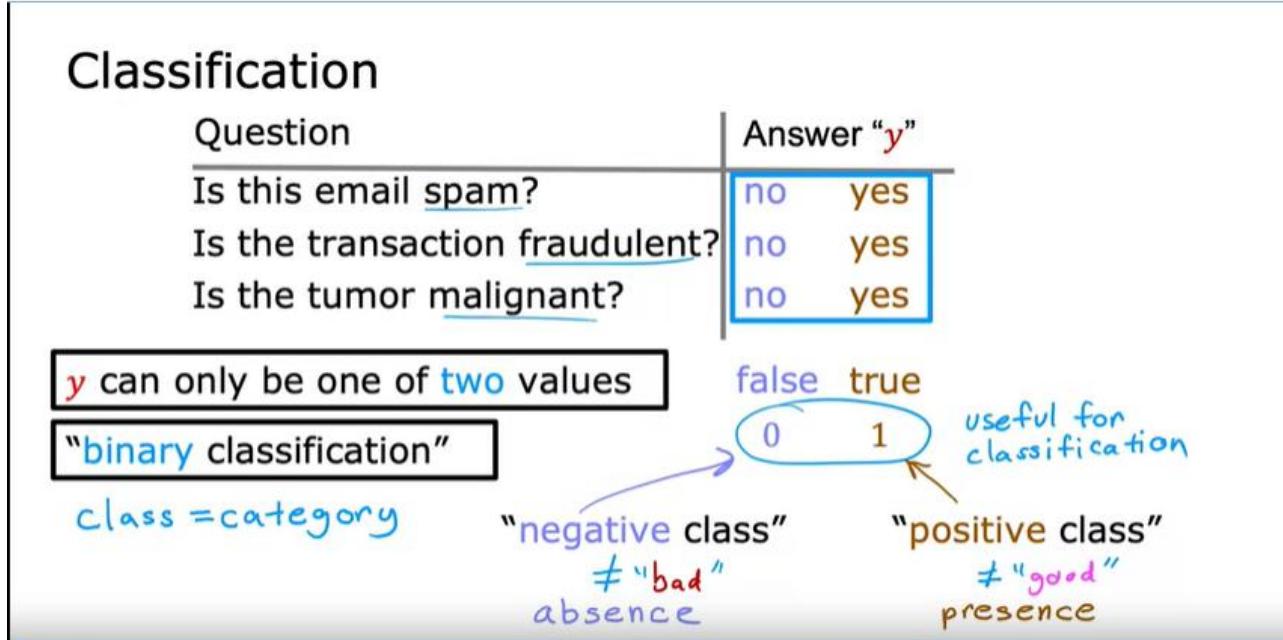
Practice quiz: Classification with logistic regression 30m

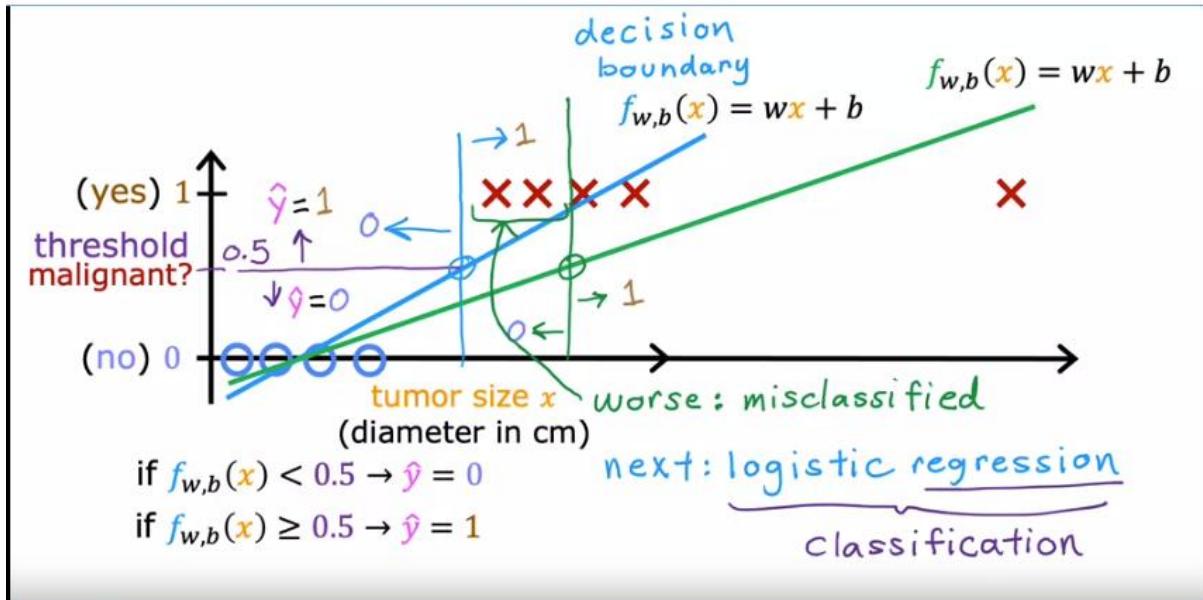
Practice quiz: Cost function for logistic regression 30m

Practice quiz: Gradient descent for logistic regression 30m

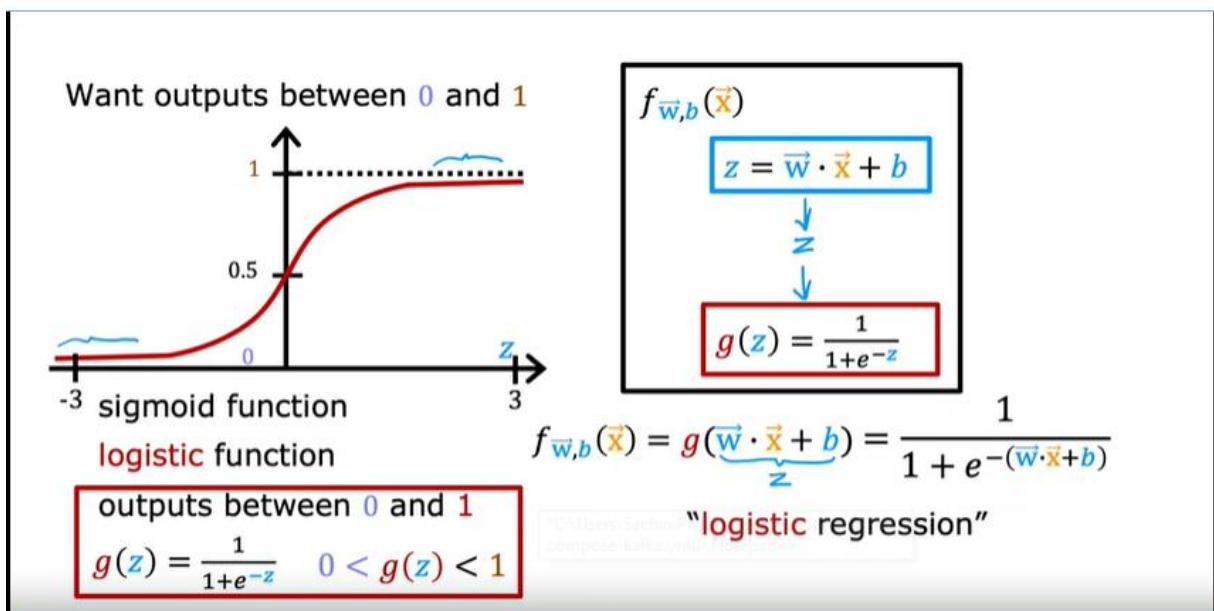
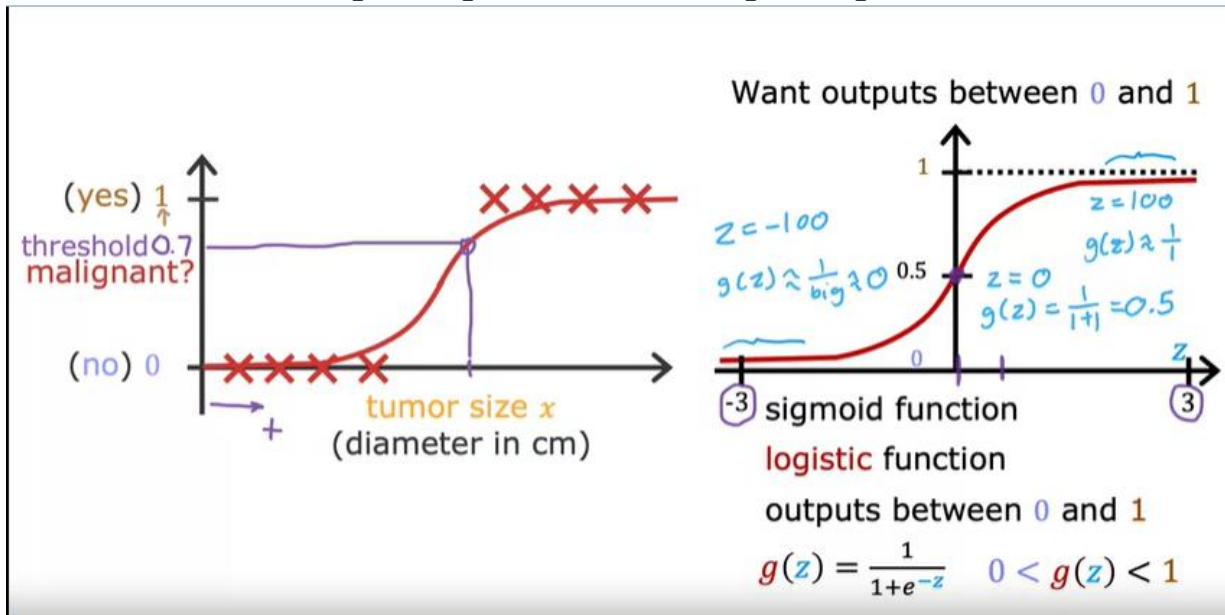
Practice quiz: The problem of overfitting 30m

3.1 - Classification with logistic regression : 3.1.1 - Motivation :





.1 - Classification with logistic regression : 3.1.2 - Logistic regression :



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"

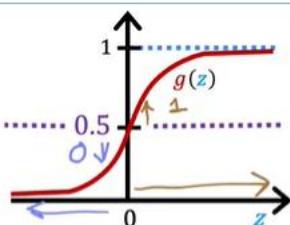
y is 0 (not malignant)
or 1 (malignant)

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

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

70% chance that y is 1

3.1 - Classification with logistic regression : 3.1.3 - Decision boundary



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

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

$$\downarrow$$

$$z$$

$$\downarrow$$

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

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

$$= P(y = 1 | \vec{x}; \vec{w}, b) \quad 0.7 \quad 0.3$$

0 or 1? $\underbrace{\text{threshold}}$

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

Yes: $\hat{y} = 1$

No: $\hat{y} = 0$

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

$$g(z) \geq 0.5$$

$$z \geq 0$$

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

$$\hat{y} = 1$$

$$\hat{y} = 0$$

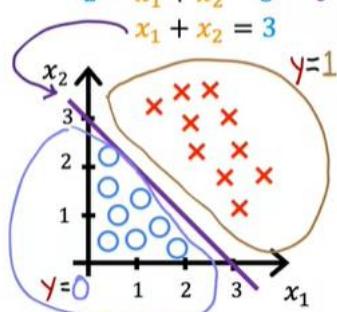
Decision boundary

$$f_{\vec{w}, b}(\vec{x}) = g(z) = g(\underbrace{\vec{w} \cdot \vec{x} + b}_{\substack{1 \\ 1 \\ -3}}) = g(w_1 x_1 + w_2 x_2 + b)$$

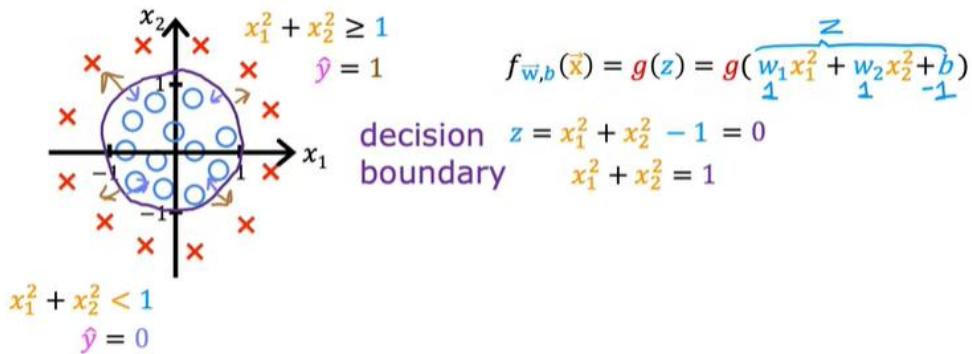
Decision boundary $z = \vec{w} \cdot \vec{x} + b = 0$

$$z = w_1 x_1 + w_2 x_2 - 3 = 0$$

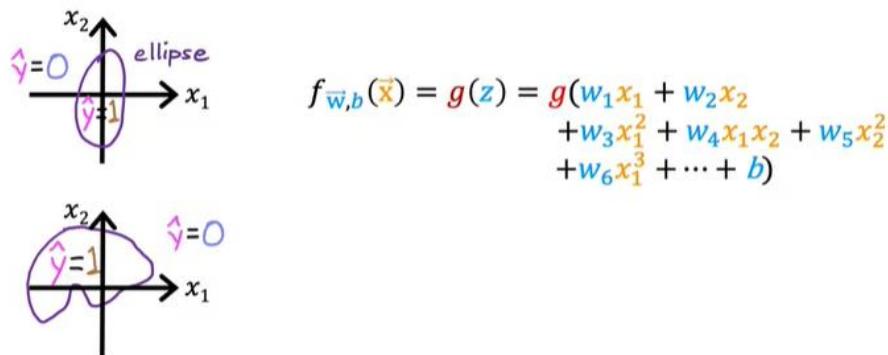
$$x_1 + x_2 = 3$$



Non-linear decision boundaries



Non-linear decision boundaries



Remember that the cost function gives you a way to measure how well a specific set of parameters fits the training data. Thereby gives you a way to try to choose better parameters.

3.2 - Cost function for logistic regression : 3.2.1 - Cost function for logistic regression :

Training set

	tumor size (cm) x_1	...	patient's age x_n	malignant? y	$i = 1, \dots, m \leftarrow$ training examples
$i=1$	10		52	1	target y is 0 or 1
:	2		73	0	
5	55		0		
12	49		1		$f_{\vec{w}, b}(\vec{x}) = \frac{1}{1 + e^{-(\vec{w} \cdot \vec{x} + b)}}$
$i=m$	

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

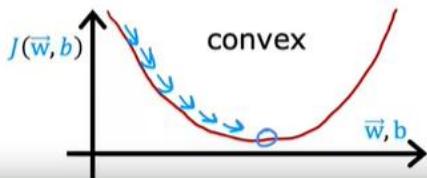
Squared error cost

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

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

linear regression

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



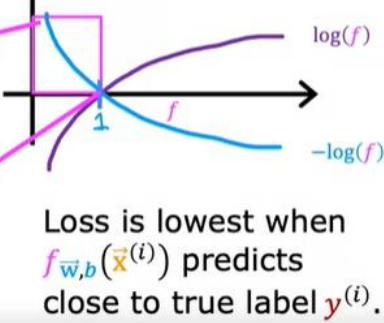
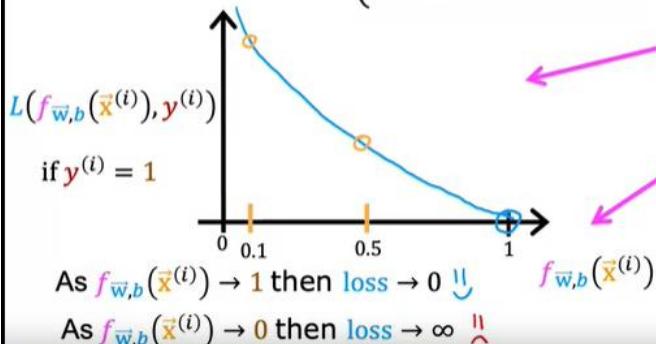
logistic regression

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



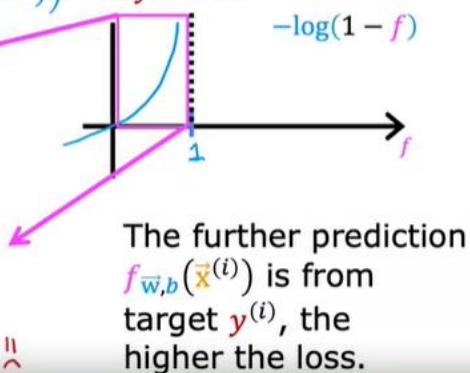
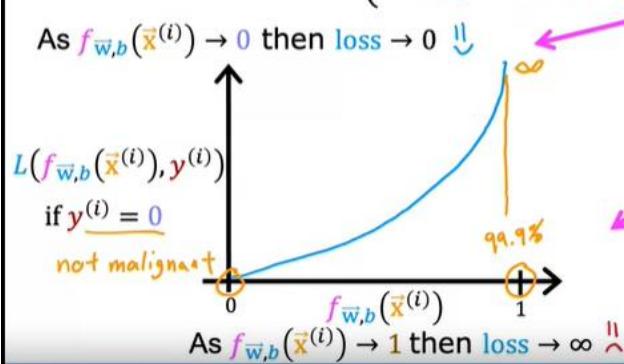
Logistic loss function

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



Logistic loss function

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



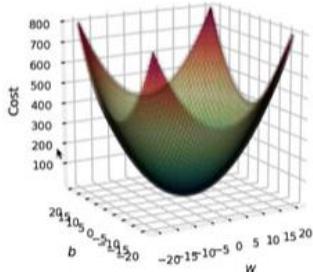
Cost

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

$L(f_{\vec{w}, b}(\vec{x}^{(i)}), y^{(i)}) = \begin{cases} -\log(f_{\vec{w}, b}(\vec{x}^{(i)})) & \text{if } y^{(i)} = 1 \\ -\log(1 - f_{\vec{w}, b}(\vec{x}^{(i)})) & \text{if } y^{(i)} = 0 \end{cases}$
convex
can reach a global minimum

find w, b that minimize cost J

Squared Error Cost used in Linear Regression



This cost function worked well for linear regression, it is natural to consider it for logistic regression as well. However, as the slide shows points out, $f_{w,b}(x)$ now has a non-linear component, the sigmoid function: $f_{w,b}(x^{(i)}) = \text{sigmoid}(wx^{(i)} + b)$. Let's try a squared error cost on the example from an earlier slide, now including the sigmoid.

3.2 - Cost function for logistic regression : 3.2.2 - Simplified Cost Function for Logistic Regression :

Simplified loss function

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

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

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

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

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

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

Simplified cost function

$$\begin{aligned}
 & \text{loss} \\
 L(f_{\vec{w}, b}(\vec{x}^{(i)}), y^{(i)}) &= -y^{(i)} \log(f_{\vec{w}, b}(\vec{x}^{(i)})) - (1 - y^{(i)}) \log(1 - f_{\vec{w}, b}(\vec{x}^{(i)})) \\
 J(\vec{w}, b) &= \frac{1}{m} \sum_{i=1}^m [L(f_{\vec{w}, b}(\vec{x}^{(i)}), y^{(i)})] \quad \text{convex} \\
 &= -\frac{1}{m} \sum_{i=1}^m [y^{(i)} \log(f_{\vec{w}, b}(\vec{x}^{(i)})) + (1 - y^{(i)}) \log(1 - f_{\vec{w}, b}(\vec{x}^{(i)}))] \\
 &\text{maximum likelihood} \\
 &\text{(don't worry about it!)}
 \end{aligned}$$

3.3 - Gradient descent for logistic regression : 3.3.1 - Gradient Descent Implementation :

Training logistic regression

Find \vec{w}, b

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

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

the probability that
the label y is one.

Gradient descent

$$\begin{aligned}
 & \text{cost} \\
 J(\vec{w}, b) &= -\frac{1}{m} \sum_{i=1}^m [y^{(i)} \log(f_{\vec{w}, b}(\vec{x}^{(i)})) + (1 - y^{(i)}) \log(1 - f_{\vec{w}, b}(\vec{x}^{(i)}))] \\
 & \text{repeat } \{ \\
 & \quad j = 1 \dots n \\
 & \quad w_j = w_j - \alpha \frac{\partial}{\partial w_j} J(\vec{w}, b) \\
 & \quad b = b - \alpha \frac{\partial}{\partial b} J(\vec{w}, b) \\
 & \} \text{ simultaneous updates}
 \end{aligned}$$

$$\begin{aligned}
 \frac{\partial}{\partial w_j} J(\vec{w}, b) &= \frac{1}{m} \sum_{i=1}^m (f_{\vec{w}, b}(\vec{x}^{(i)}) - y^{(i)}) \underline{x_j^{(i)}} \\
 \frac{\partial}{\partial b} J(\vec{w}, b) &= \frac{1}{m} \sum_{i=1}^m (f_{\vec{w}, b}(\vec{x}^{(i)}) - y^{(i)}) \underline{1}
 \end{aligned}$$

Gradient descent for logistic regression

repeat { *looks like linear regression!*

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

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

} simultaneous updates

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

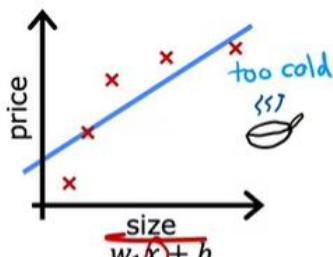
Logistic regression $f_{\bar{w}, b}(\vec{x}) = \frac{1}{1 + e^{-(\vec{w} \cdot \vec{x} + b)}}$

Same concepts:

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

3.4 - The problem of overfitting : 3.4.1 - The problem of overfitting :

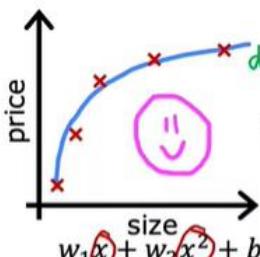
Regression example



underfit

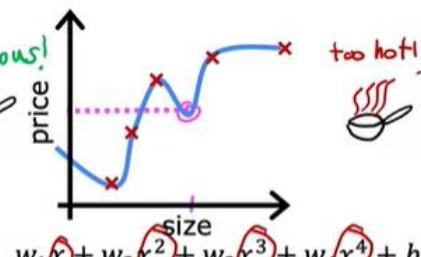
- Does not fit the training set well

high bias



just right

generalization

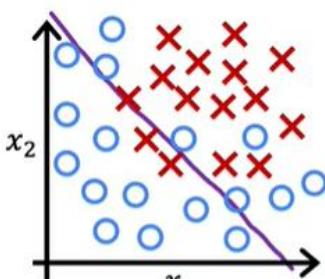


overfit

- Fits the training set extremely well

high variance

Classification

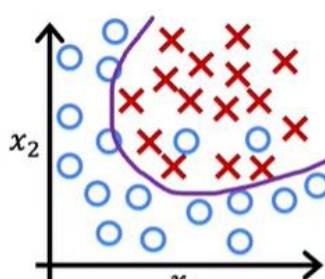


$$z = w_1 x_1 + w_2 x_2 + b$$

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

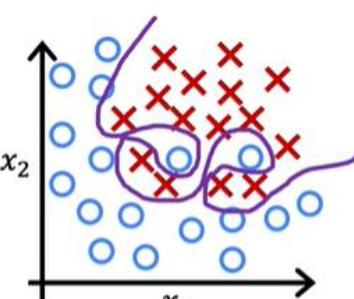
g is the sigmoid function

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

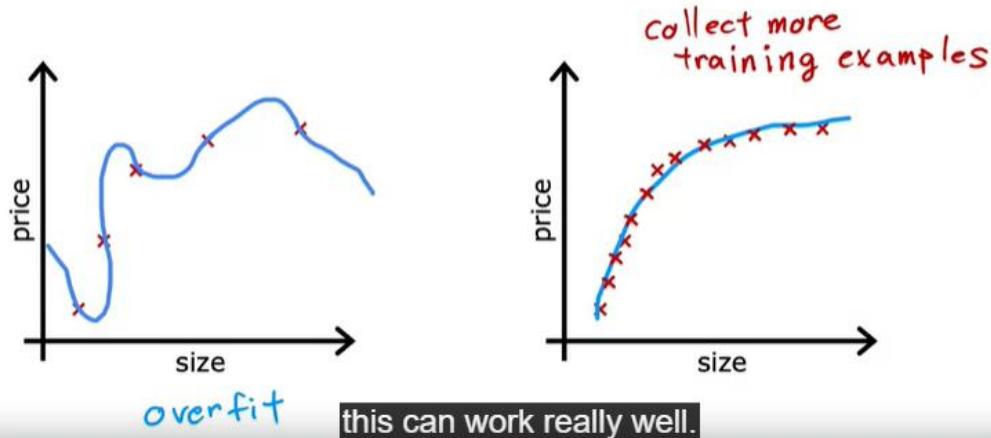


$$z = w_1 x_1 + w_2 x_2 + w_3 x_1^2 x_2 + w_4 x_1^2 x_2^2 + w_5 x_1^2 x_2^3 + w_6 x_1^3 x_2 + \dots + b$$

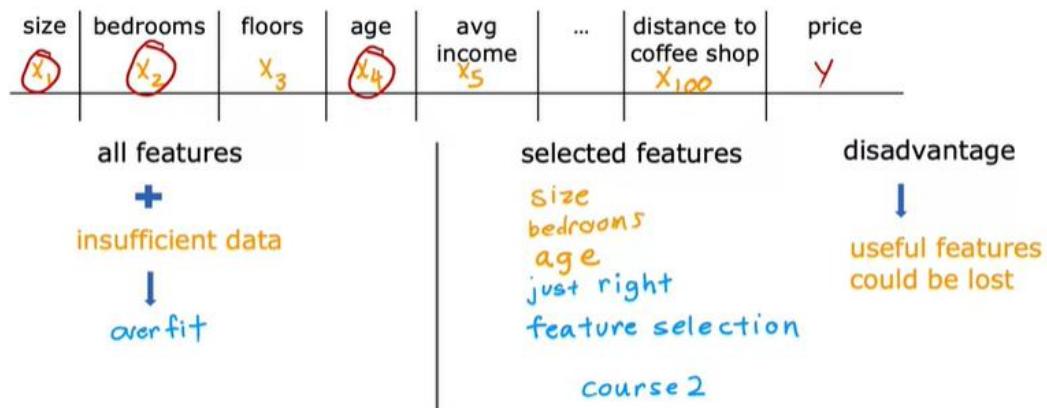
Overfit

3.4 - The problem of overfitting : 3.4.2 - Addressing overfitting :

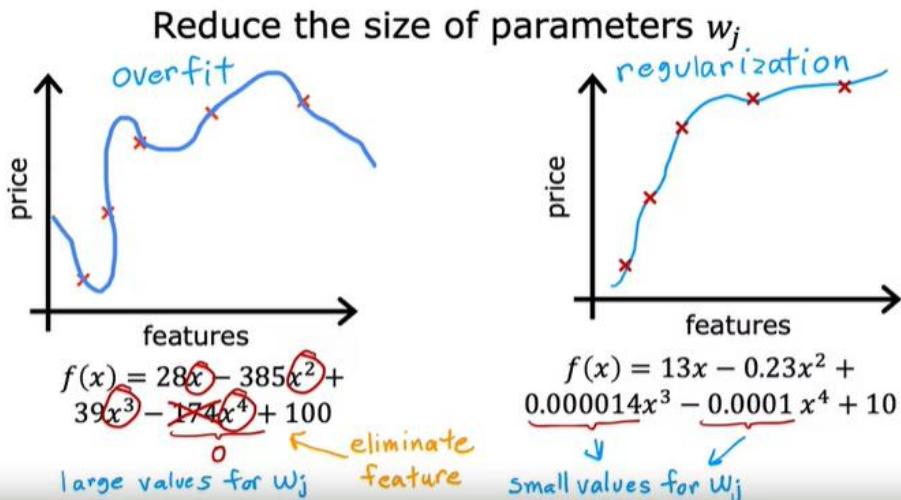
Collect more training examples



Select features to include/exclude



Regularization



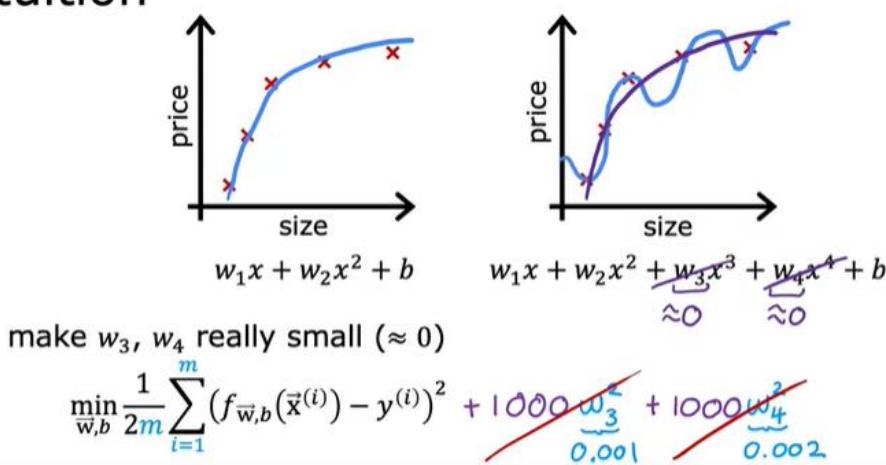
Addressing overfitting

Options

1. Collect more data
2. Select features
 - Feature selection *in course 2*
3. Reduce size of parameters
 - "Regularization" *next videos!*

3.4 - The problem of overfitting : 3.4.3 - Cost function with regularization :

Intuition



Regularization

simpler model
small values w_1, w_2, \dots, w_n, b 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

$w_1, w_2, \dots, w_{100}, b$ n features $n = 100$

regularization term

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

"lambda" regularization parameter $\lambda > 0$ can include or exclude b

Regularization

mean squared error regularization term

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

fit data keep w_j small

λ balances both goals

choose $\lambda = 10^{-10}$

$f_{\vec{w}, b}(\vec{x}) = w_1 x_1 + w_2 x_2 + w_3 x_3 + w_4 x_4 + b$

$f(x) = b$ choose λ

3.4 - The problem of overfitting : 3.4.4 - Regularized linear regression :

Regularized linear regression

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

Gradient descent

repeat {

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

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

} simultaneous update

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

$= \frac{1}{m} \sum_{i=1}^m (f_{\vec{w}, b}(\vec{x}^{(i)}) - y^{(i)})$ don't have to regularize b

Implementing gradient descent

repeat {

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

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

} simultaneous update $j = 1, \dots, n$

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

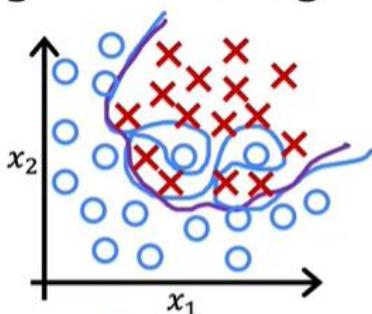
$$\begin{aligned} \alpha \frac{\lambda}{m} &= 0.01 \frac{1}{50} = 0.0002 \\ w_j \left(1 - 0.0002\right) &= 0.9998 \end{aligned}$$

How we get the derivative term (optional)

$$\begin{aligned} \frac{\partial}{\partial w_j} J(\vec{w}, b) &= \frac{\partial}{\partial w_j} \left[\frac{1}{2m} \sum_{i=1}^m (f(\vec{x}^{(i)}) - y^{(i)})^2 + \frac{\lambda}{2m} \sum_{j=1}^n w_j^2 \right] \\ &= \frac{1}{2m} \sum_{i=1}^m [(\vec{w} \cdot \vec{x}^{(i)} + b - y^{(i)}) \cancel{2x_j^{(i)}}] + \frac{\lambda}{2m} \cancel{2w_j} \underset{\text{No } \sum}{\sum_{j=1}^n} \\ &= \frac{1}{m} \sum_{i=1}^m [(\vec{w} \cdot \vec{x}^{(i)} + b - y^{(i)}) X_j^{(i)}] + \frac{\lambda}{m} w_j \\ &= \frac{1}{m} \sum_{i=1}^m [(f_{\vec{w}, b}(\vec{x}^{(i)}) - y^{(i)}) x_j^{(i)}] + \frac{\lambda}{m} w_j \end{aligned}$$

3.4 - The problem of overfitting : 3.4.5 - Regularized logistic regression :

Regularized logistic regression



$$\begin{aligned} z &= w_1 x_1 + w_2 x_2 \\ &\quad + w_3 x_1^2 x_2 + w_4 x_1^2 x_2^2 \\ &\quad + w_5 x_1^2 x_2^3 + \dots + b \\ f_{\vec{w}, b}(\vec{x}) &= \frac{1}{1 + e^{-z}} \end{aligned}$$

Cost function

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

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

Regularized logistic regression

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

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

Gradient descent

repeat {

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

$j = 1 \dots n$

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

}

Looks same as
for linear regression!

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

logistic regression

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

don't have to
regularize