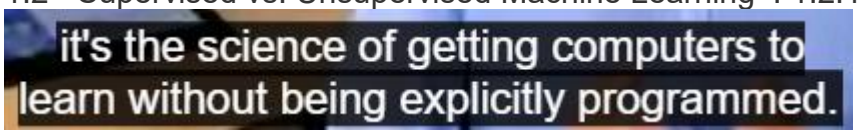
Supervised Machine Learning: **Index**

Week (Part) 3 : Classification with logistic regression **33**

Week 1 : Introduction To Machine Learning

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



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

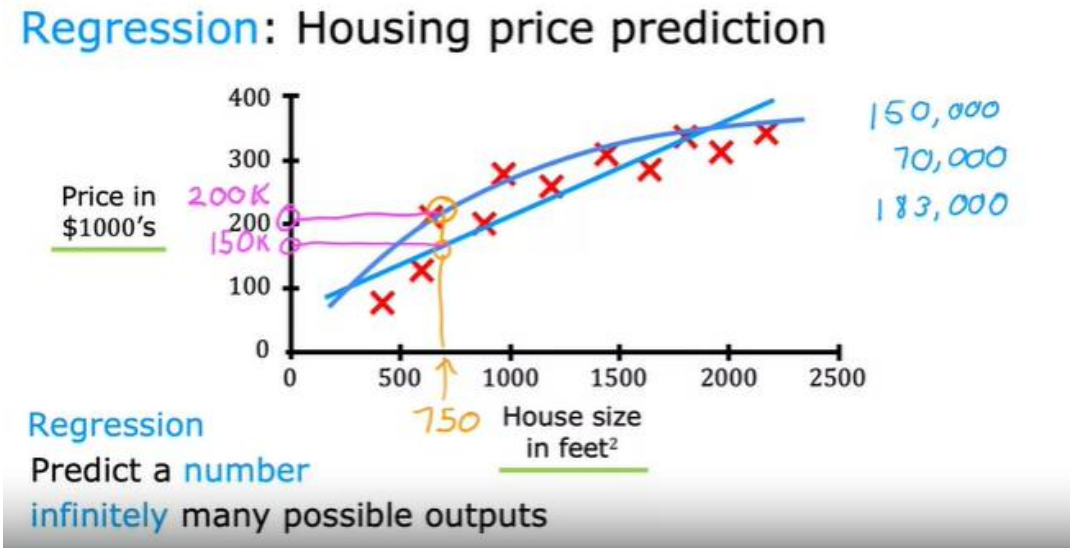
Learning Objectives

- 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

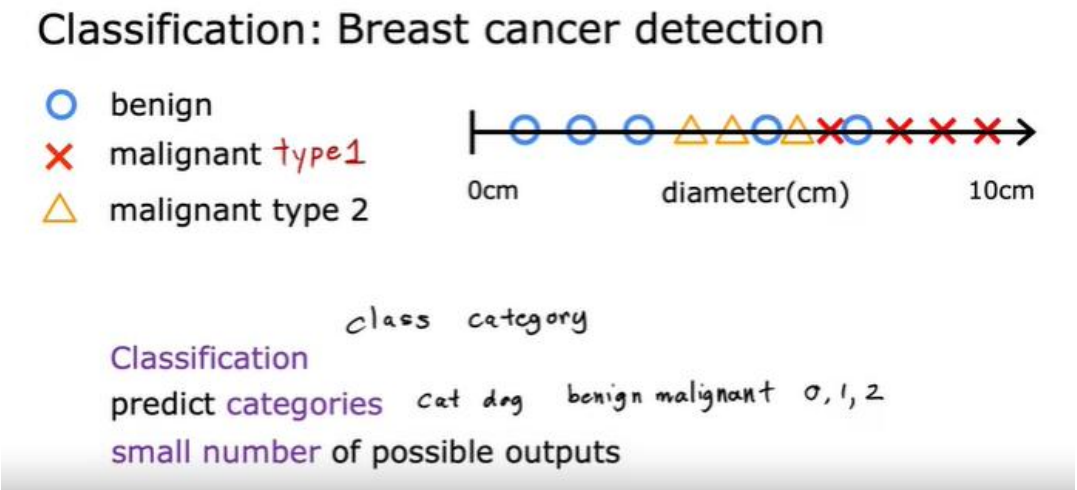
supervised learning :

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

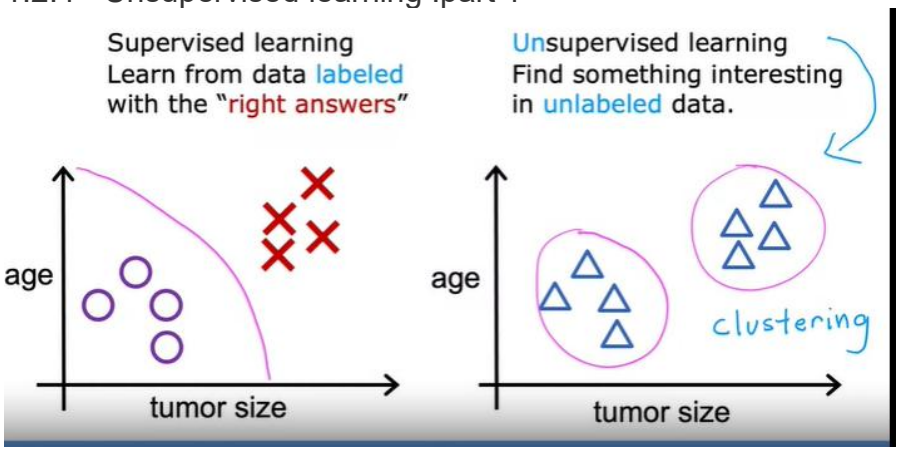
1.2.2 - supervised learning :part 1- Regression



1.2.3 - supervised learning :part 2- Classification



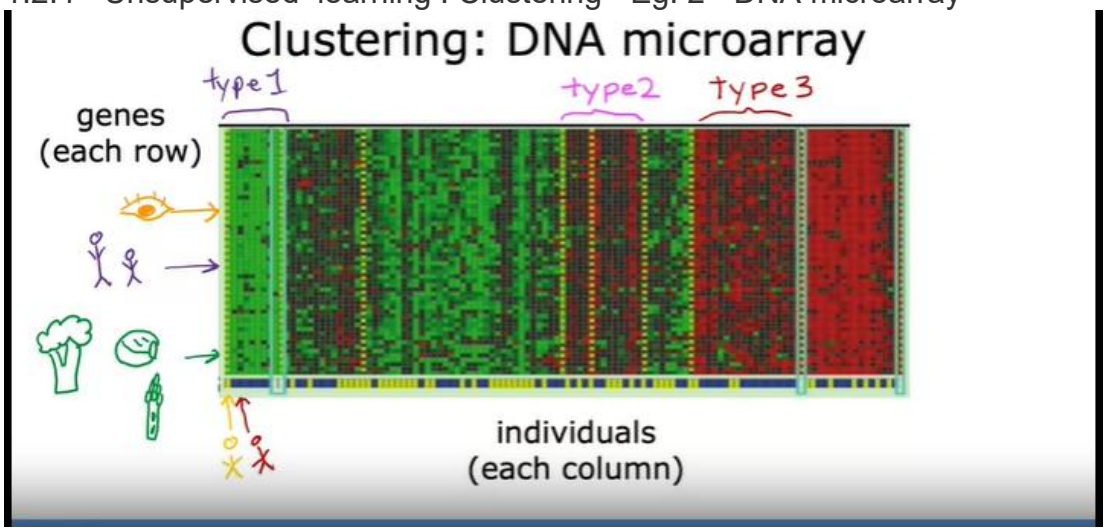
1.2.4 - Unsupervised learning :part 1-



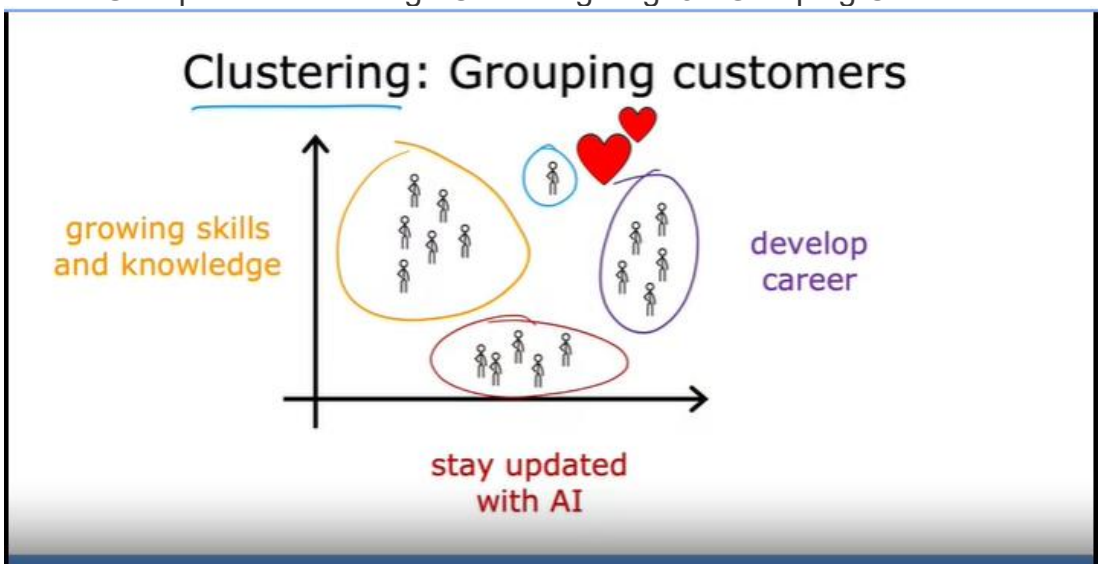
1.2.4 - Unsupervised learning : Clustering - Eg. 1 - Google News

- 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

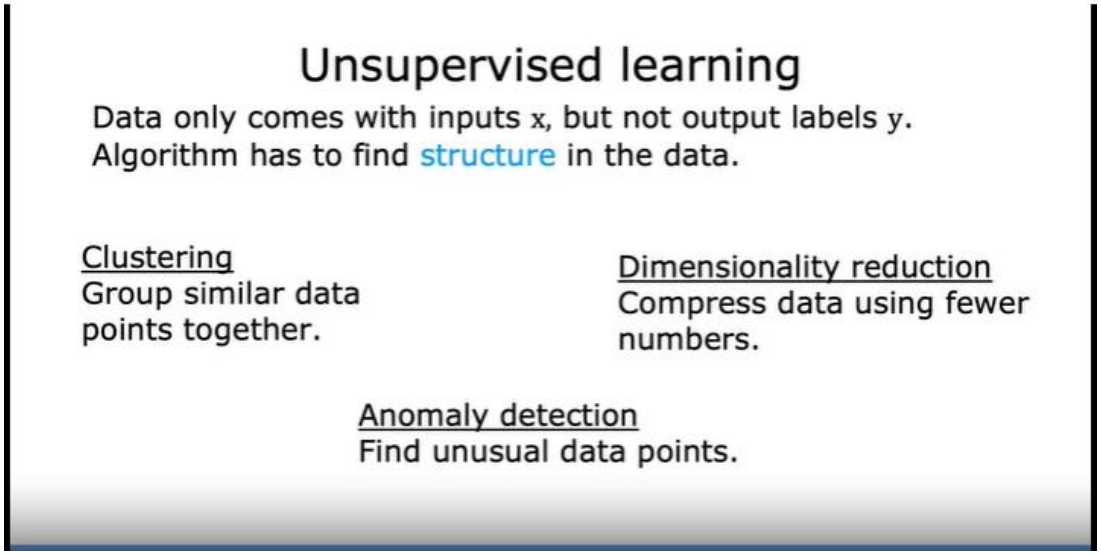
1.2.4 - Unsupervised learning : Clustering - Eg. 2 - DNA microarray



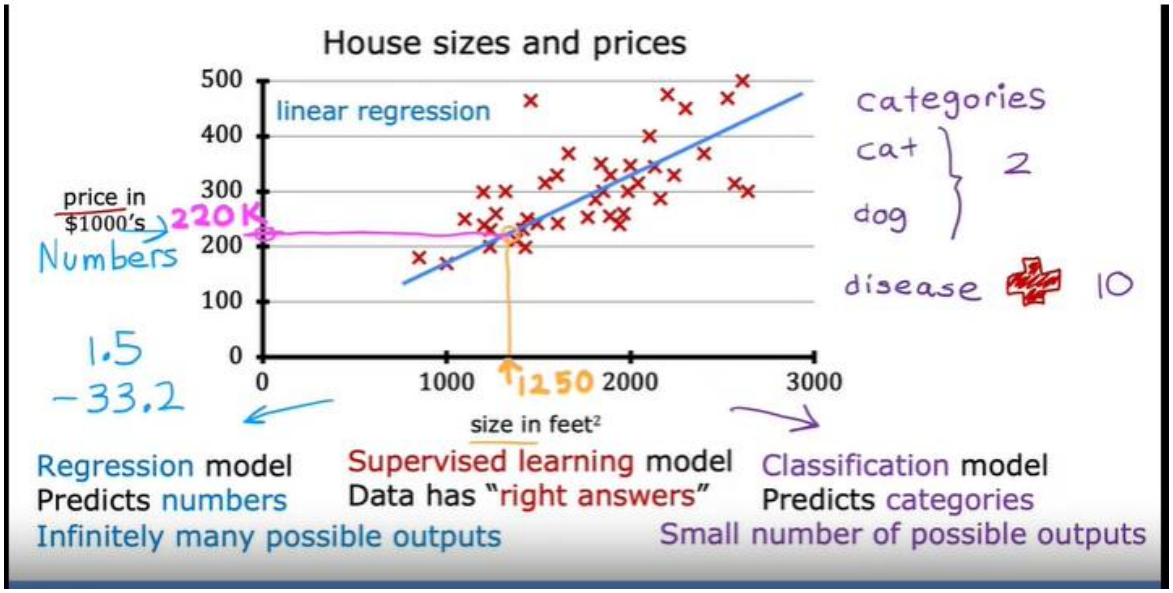
1.2.4 - Unsupervised learning : Clustering - Eg. 3 - Grouping Customers

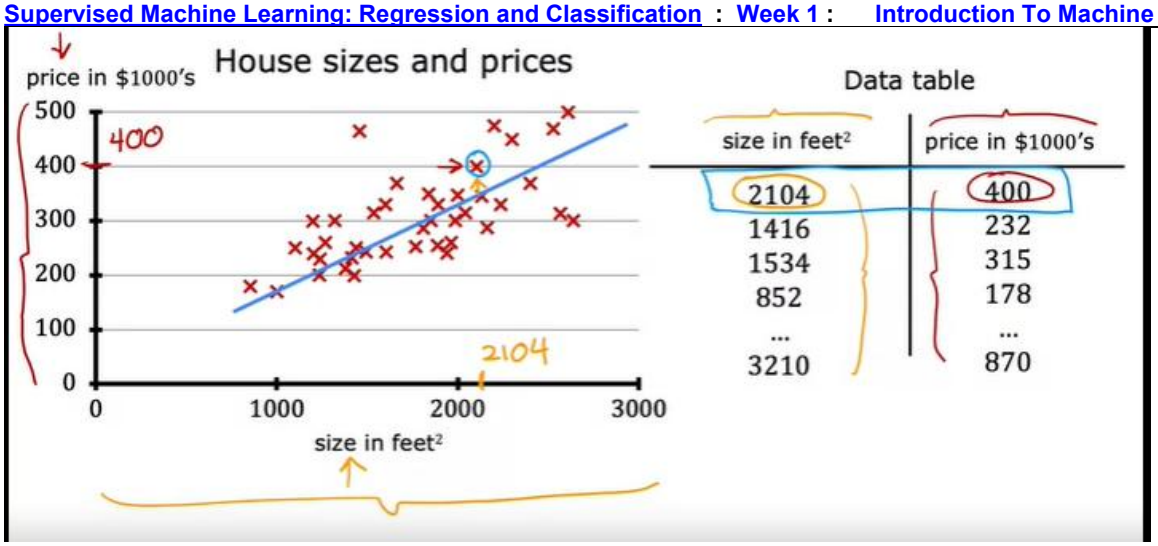


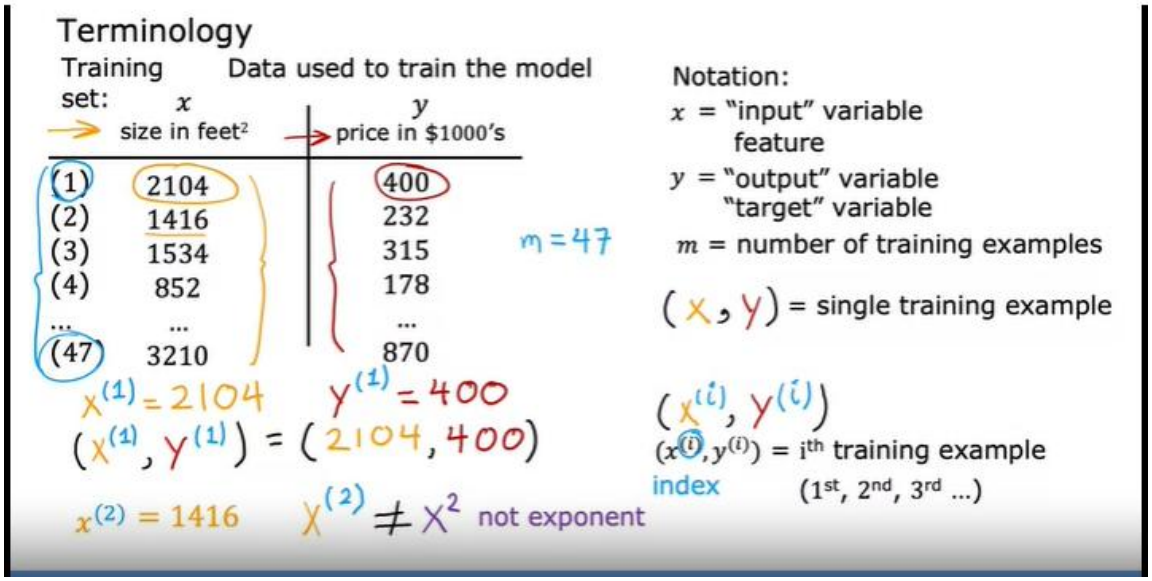
1.2.5 - Unsupervised learning :part 2-



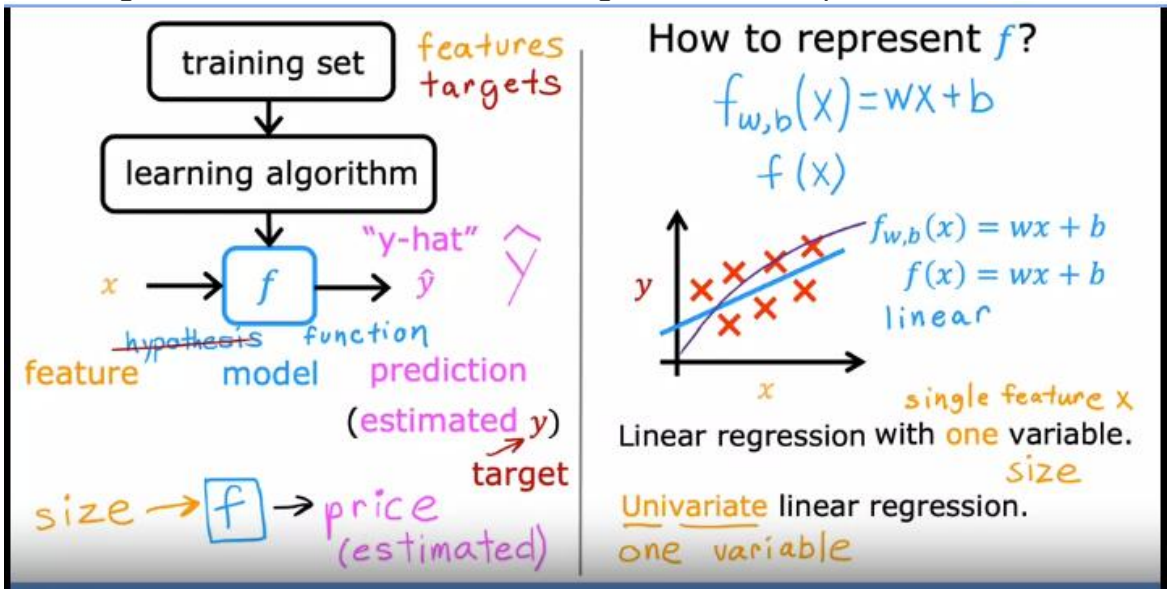
1.3 - Regression Model : 1.3.1 - Linear regression model part 1 :



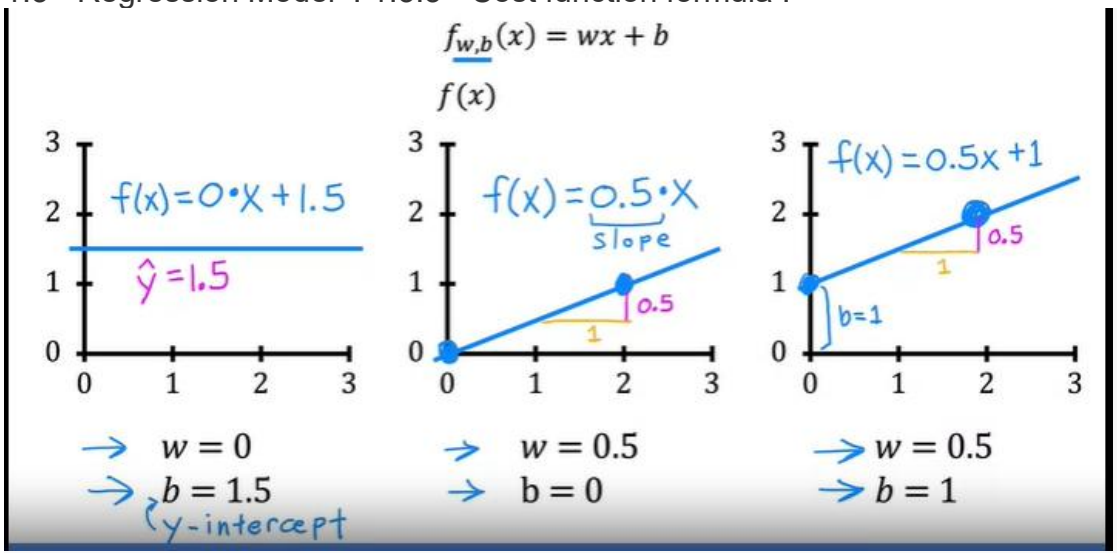




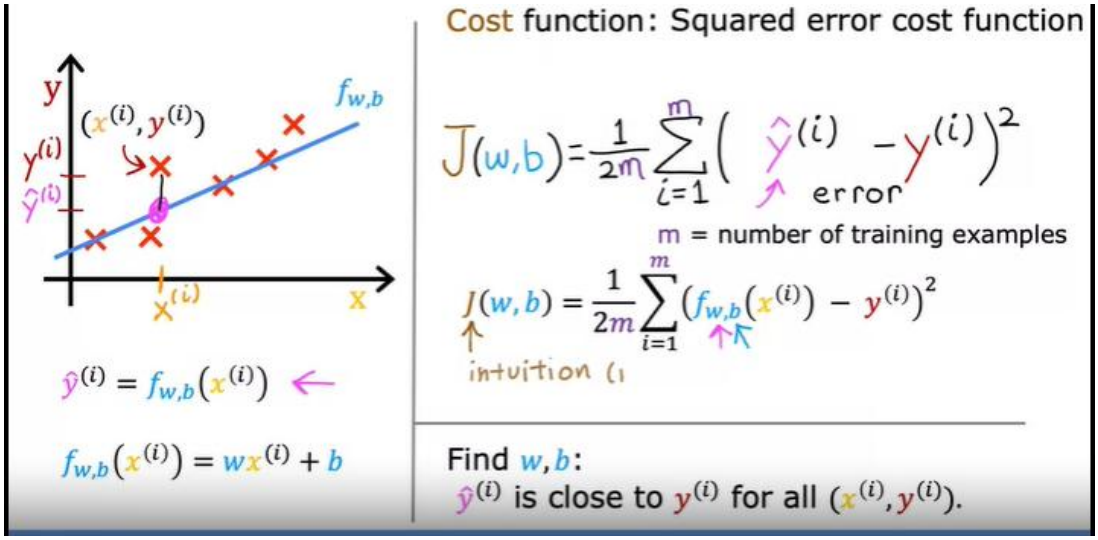
1.3 - Regression Model : 1.3.2 - Linear regression model part 2 :



1.3 - Regression Model : 1.3.3 - Cost function formula :

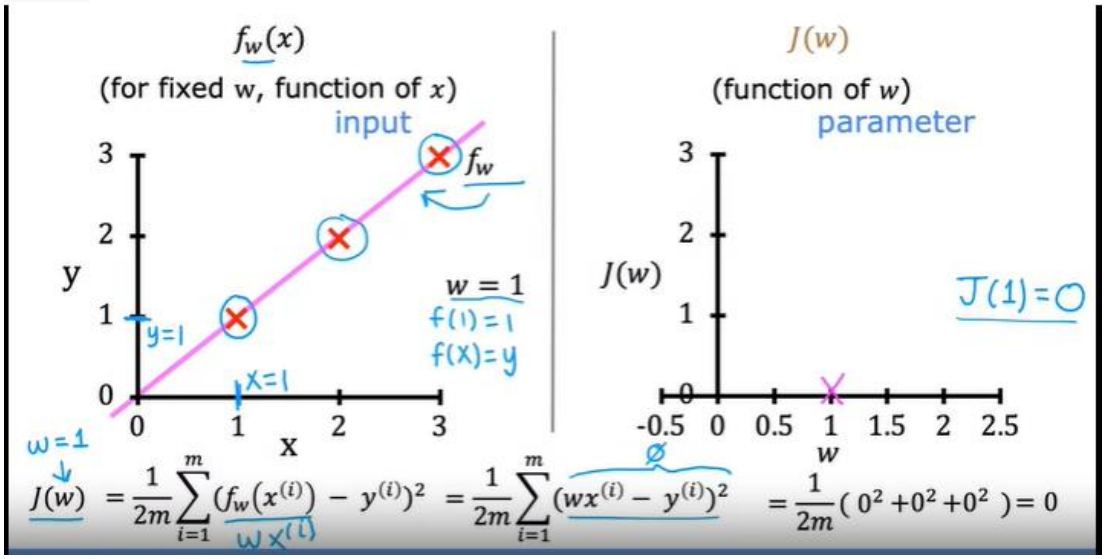


Cost function formula :

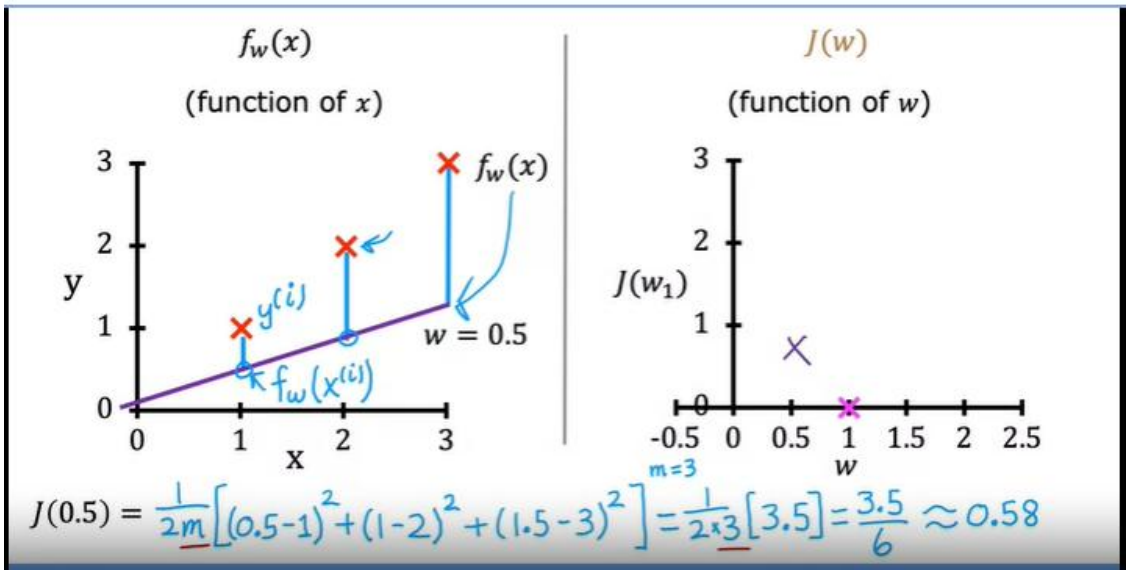


1.3 - Regression Model : 1.3.5 - Visualizing the cost function :

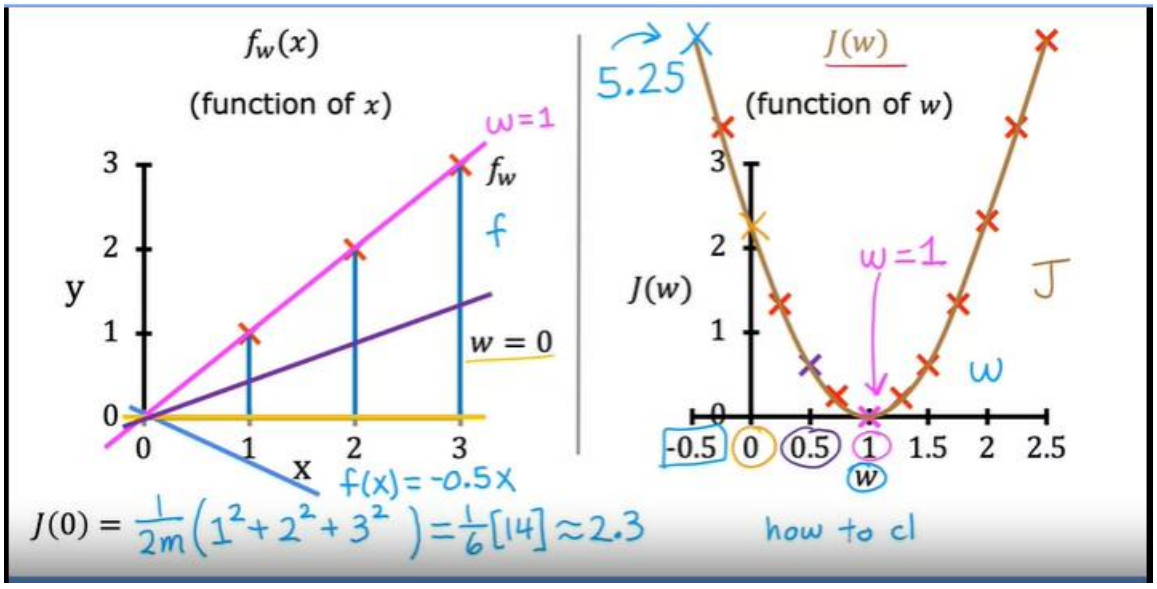
W = 1



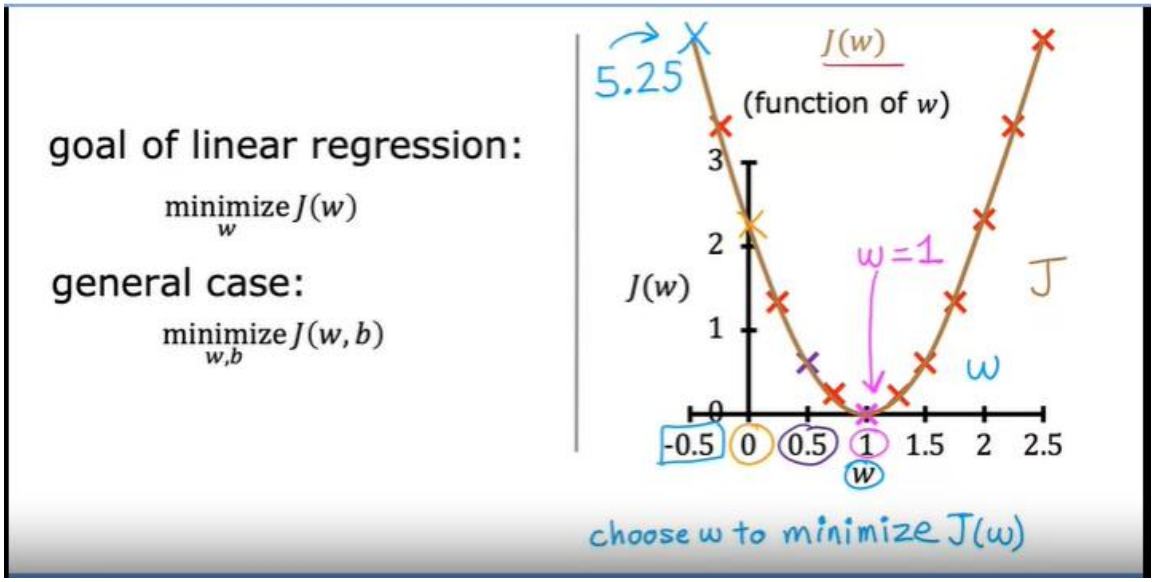
W = 0.5



W = 0, -0.5, …

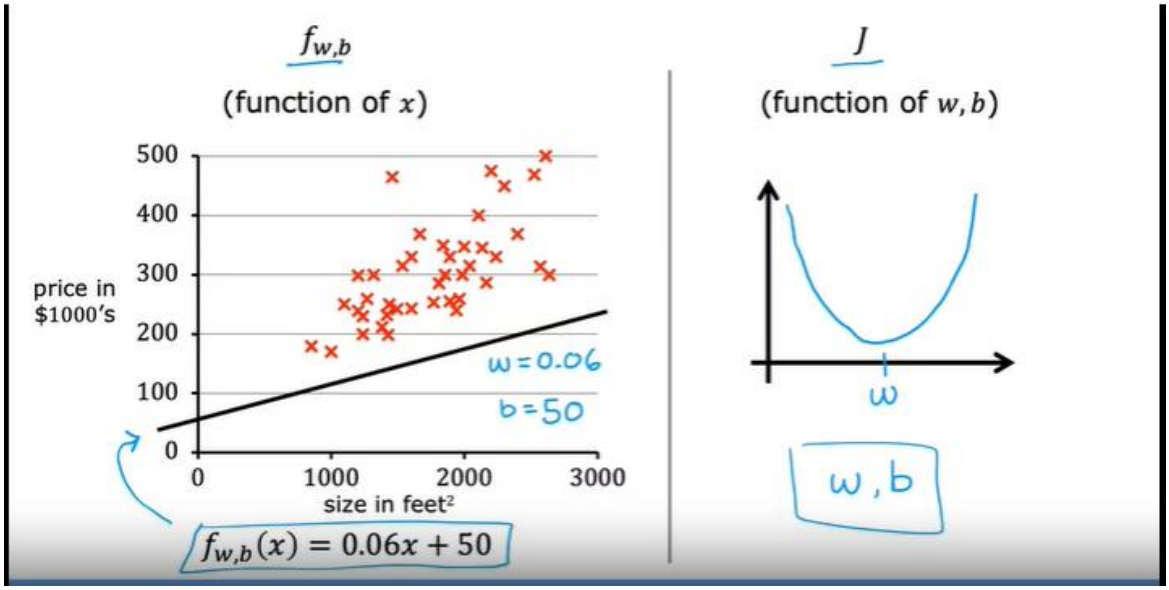


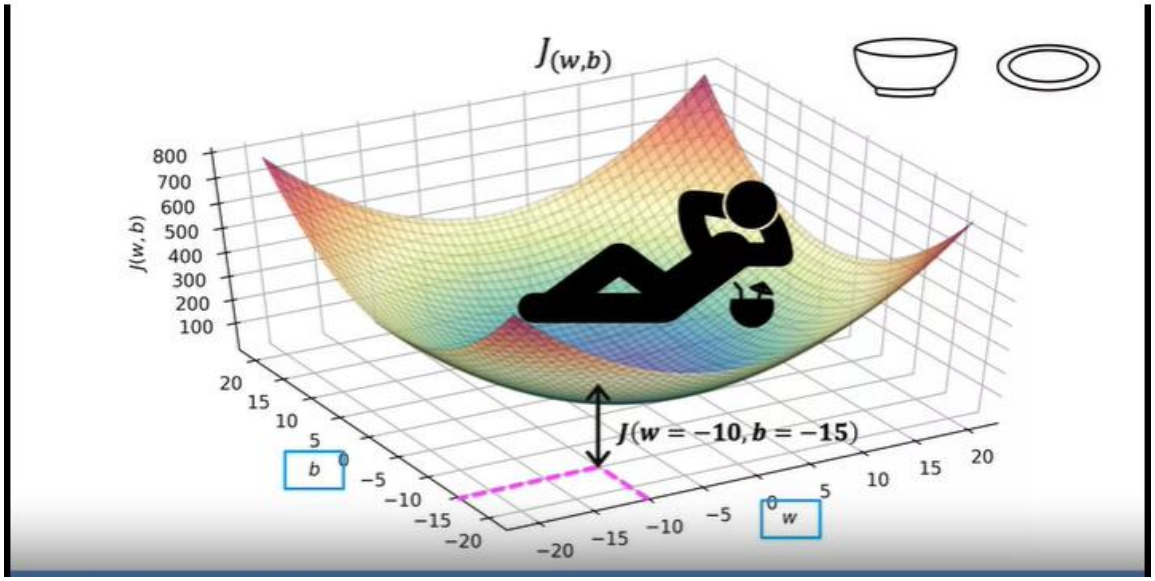
Goal of linear regression : To minimize the cost function

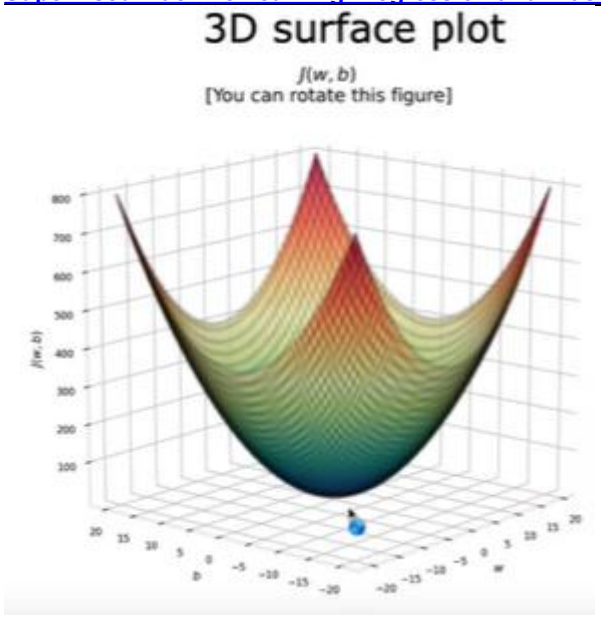


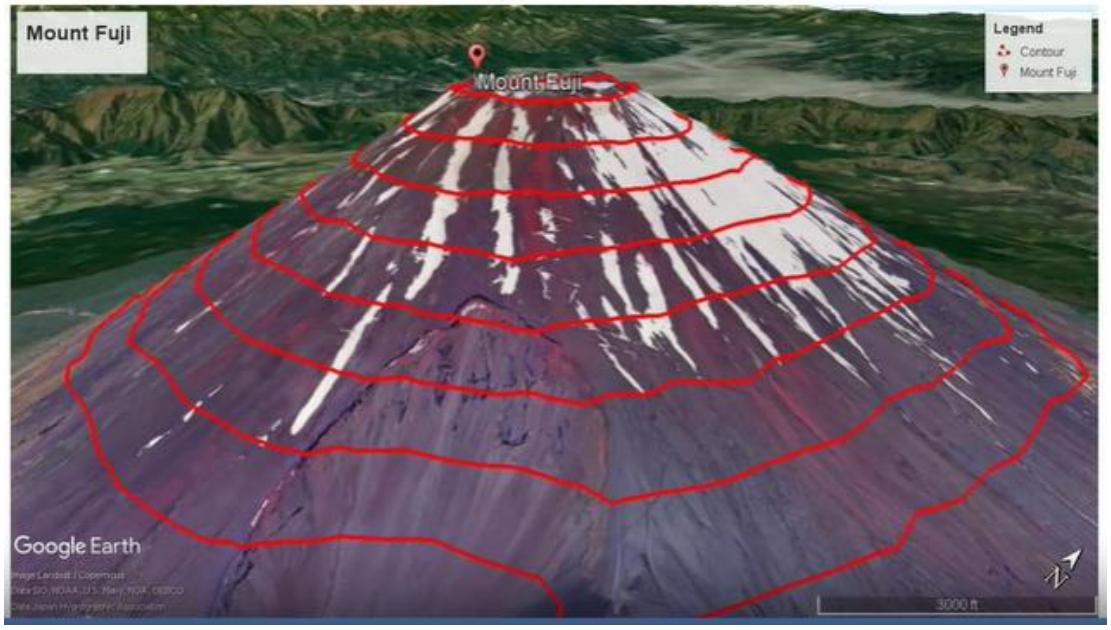
1.3 - Regression Model : 1.3.6 - Visualization examples :

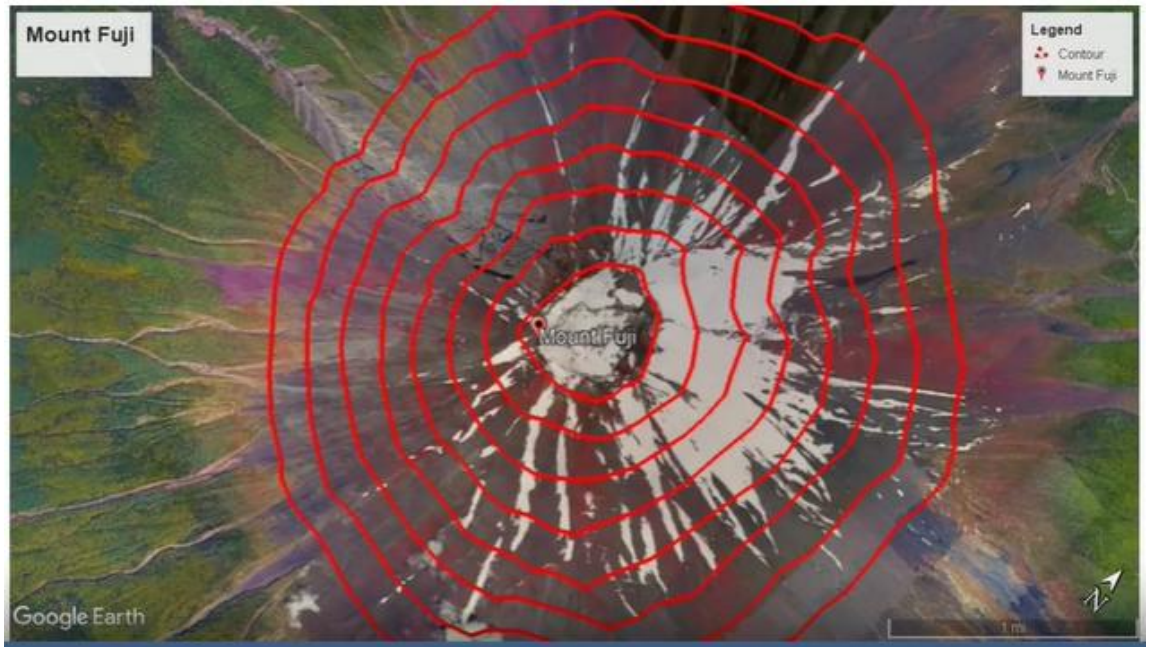
Model	$f_{w,b}(x) = wx + b$
Parameters	w, b ~~before: b=0~~
Cost Function	$J(w,b) = \frac{1}{2m}\sum_{i=1}^{m}(f_{w,b}(x^{(i)}) - y^{(i)})^2$
Objective	$\underset{w,b}{\text{minimize}}\, J(w,b)$

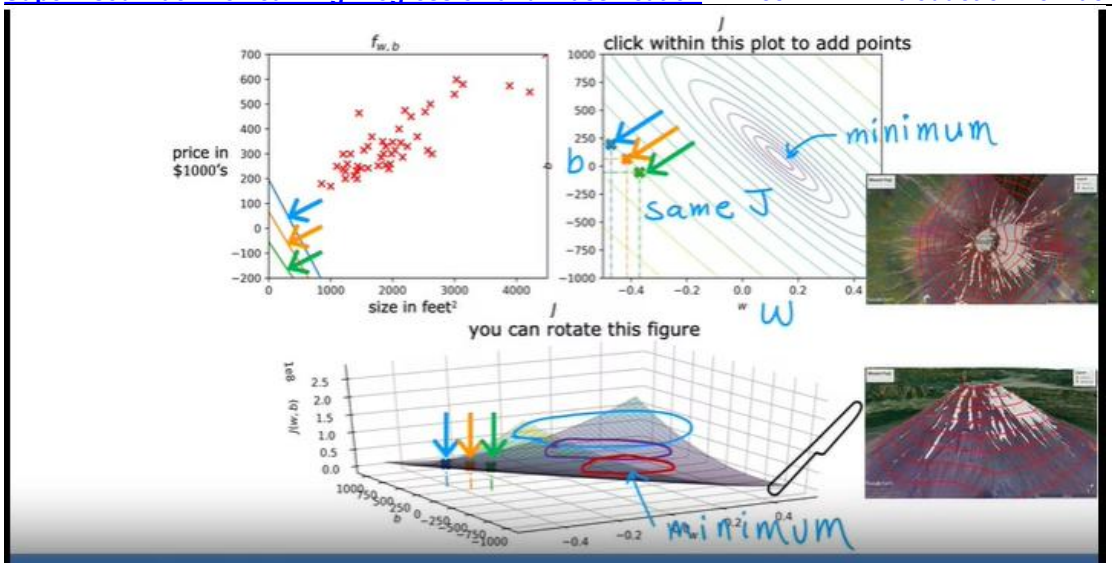





3D surface plot
J(w, b)
[You can rotate this figure]
J(w, b)
w
b

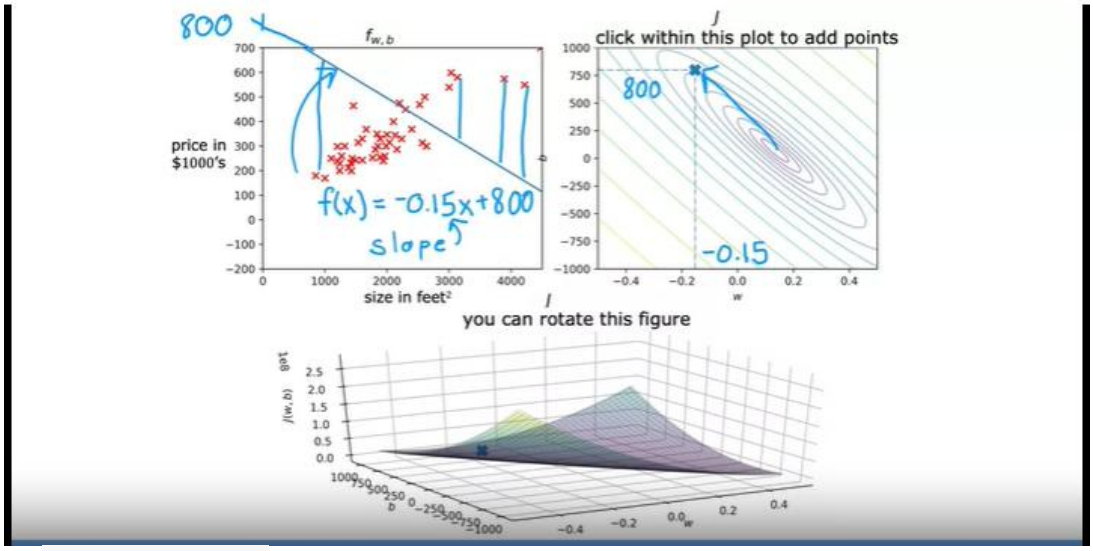

Mount Fuji
Legend
Contour
Mount Fuji
Mount Fuji
Google Earth


Mount Fuji
Legend
Contour
Mount Fuji
Mount Fuji
Google Earth

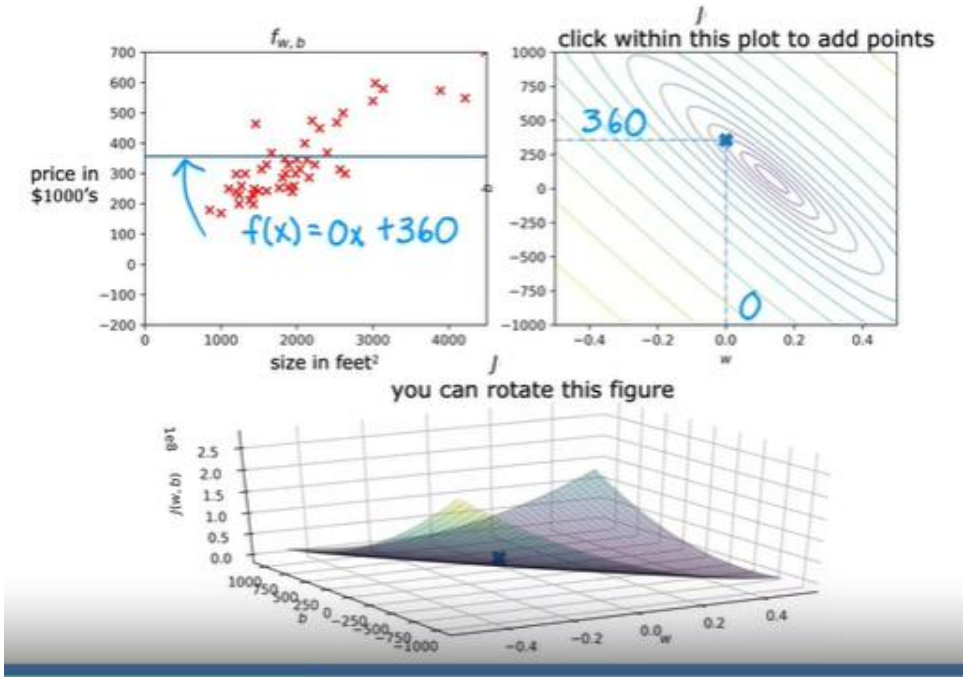


Visualization examples :

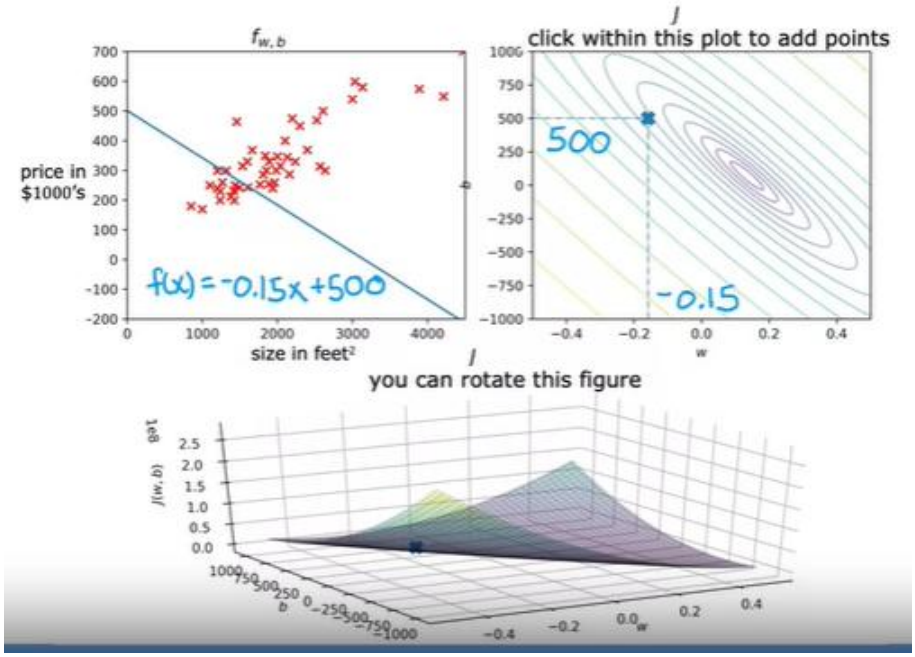
1) Bad fit



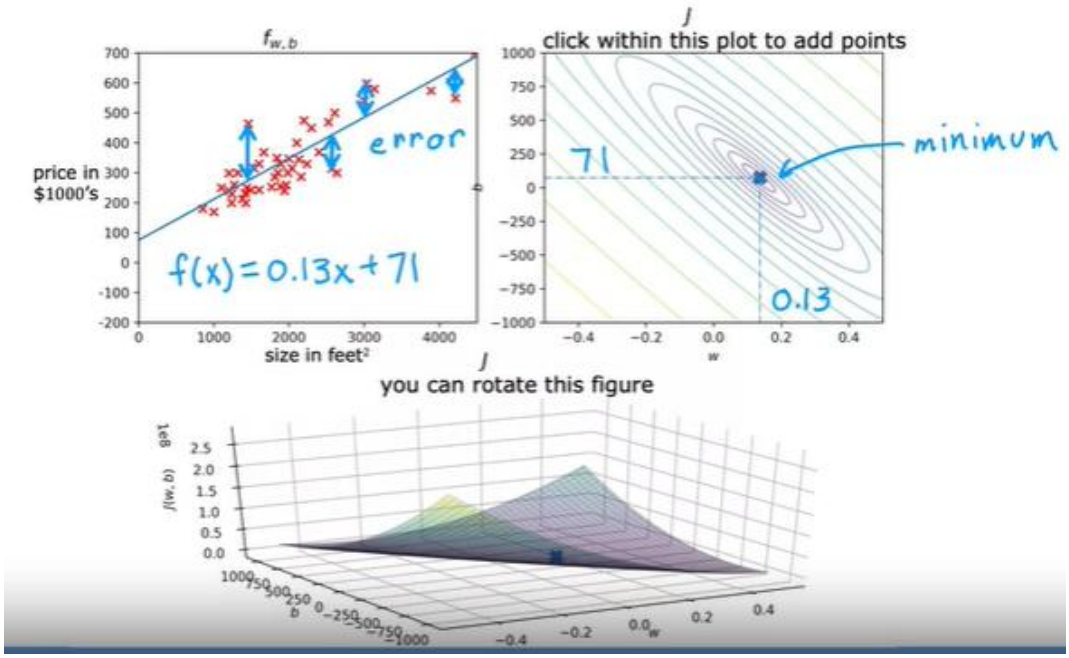
2) Not That bad fit



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

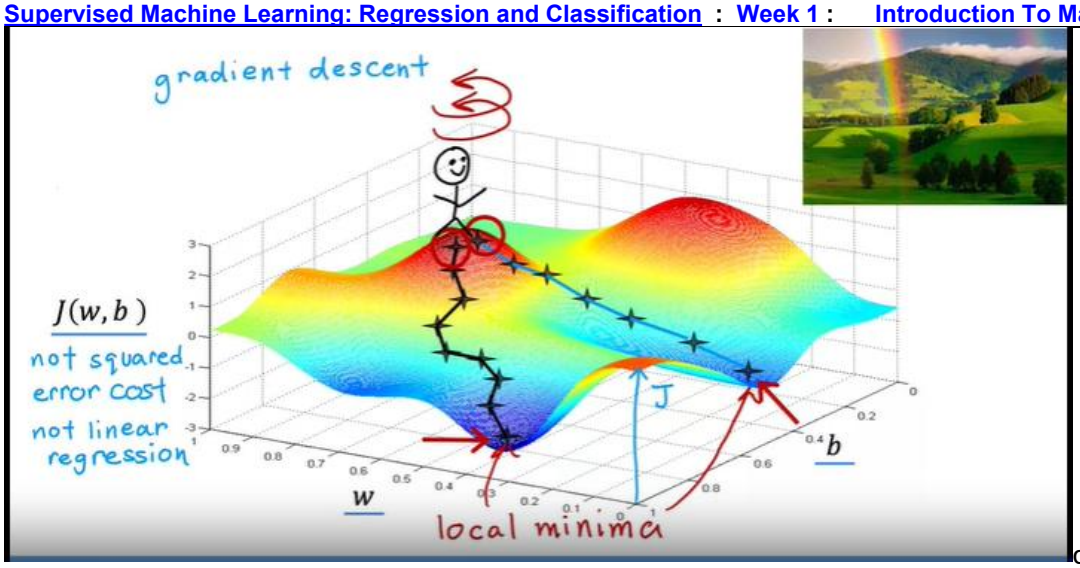
Want $\min_{w,b} J(w,b)$ $\min_{w_1,\ldots,w_n,b} J(w_1,w_2,\ldots,w_n,b)$

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

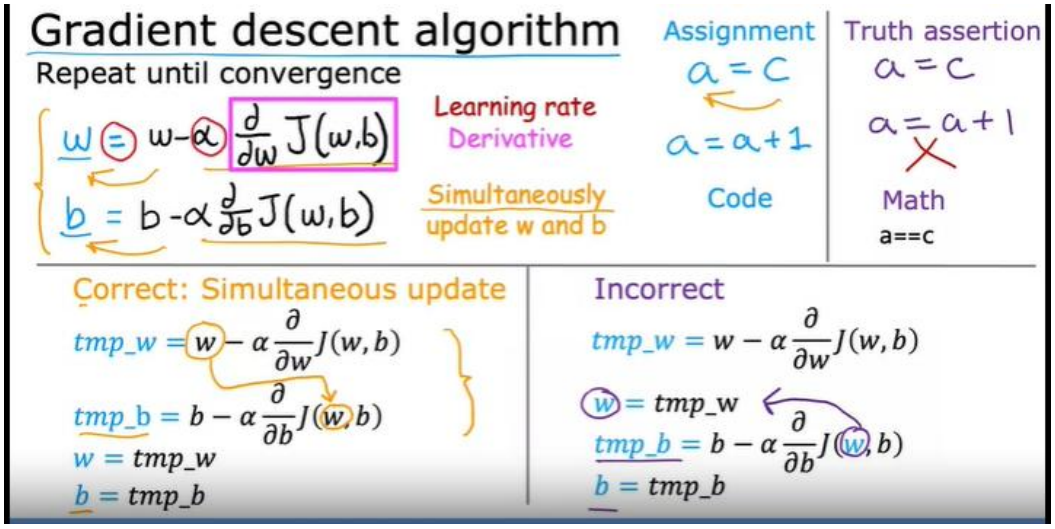


αα

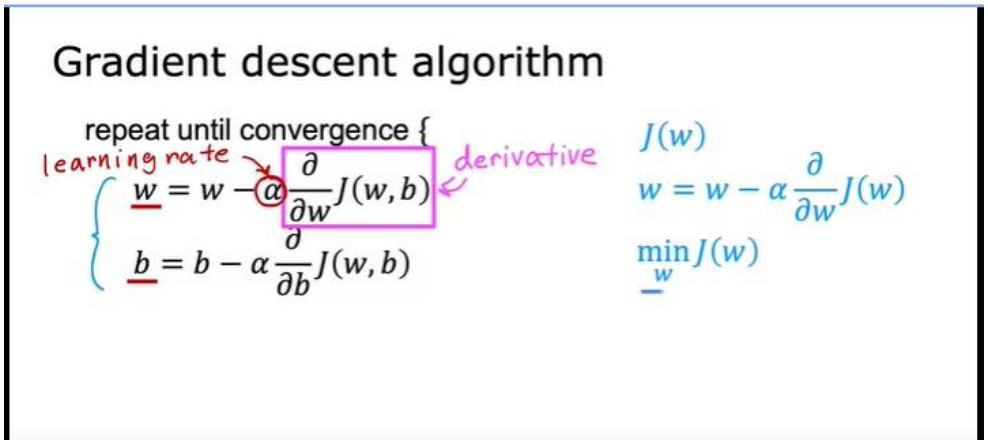
1.4 - Train the model with gradient descent : 1.4.2 - Implementing gradient descent :

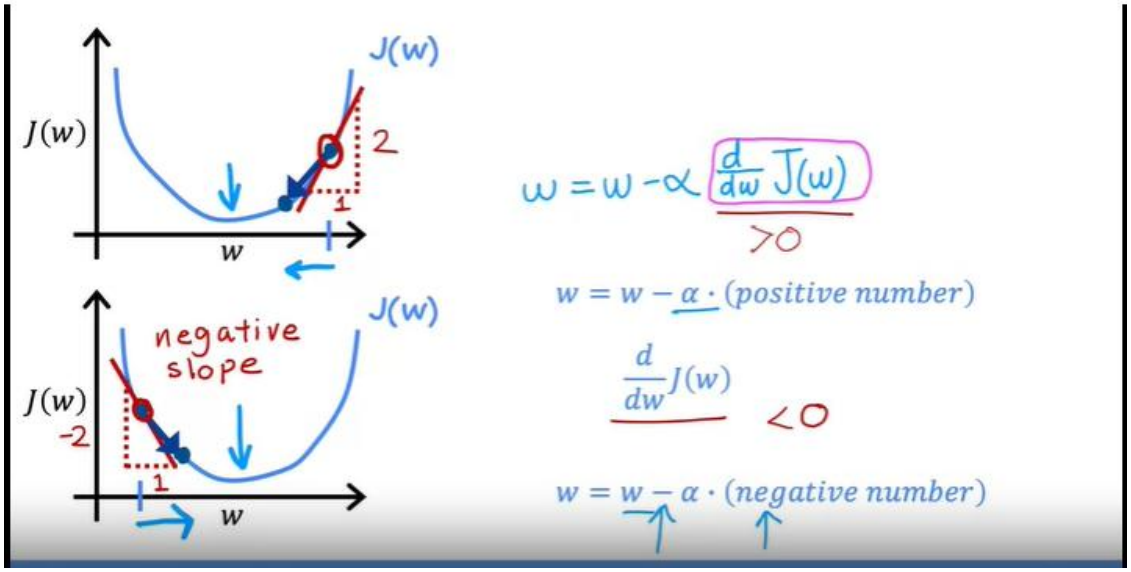
α = Learning rate, 0 to 1, (say 0.01) = size of the baby step

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

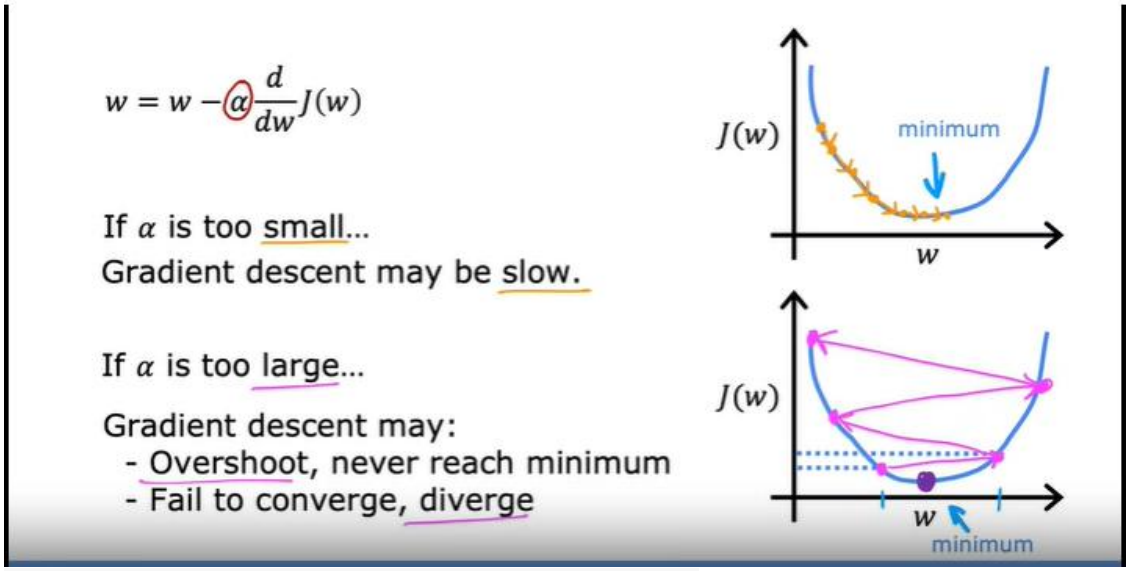


1.4 - Train the model with gradient descent : 1.4.3 - Gradient descent intuition :

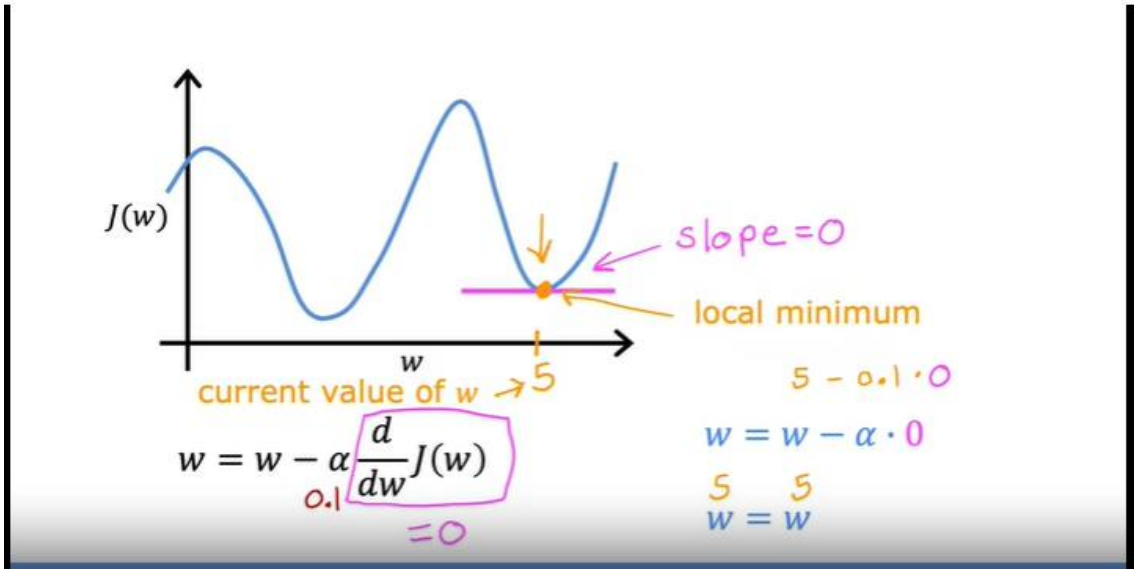


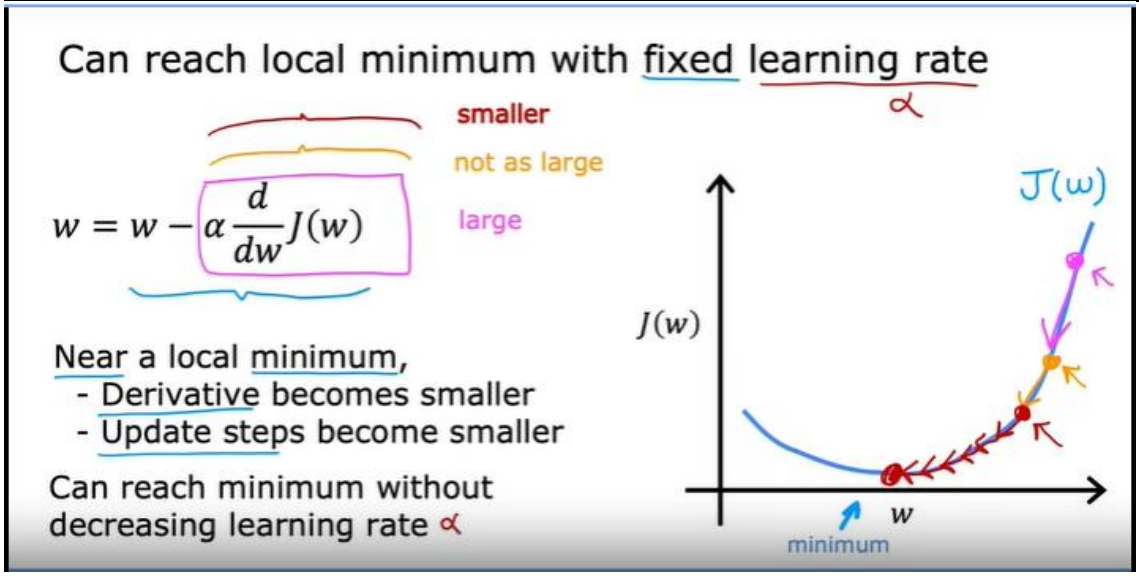


1.4 - Train the model with gradient descent : 1.4.4 - Learning rate :

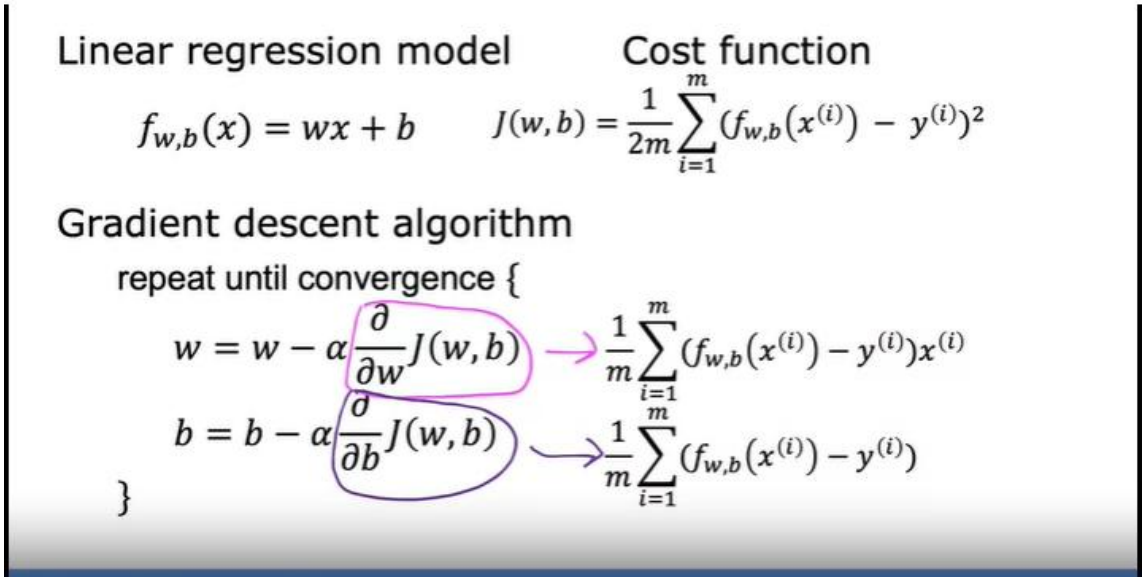


Learning rate : Tricky situation of 2 local minimas

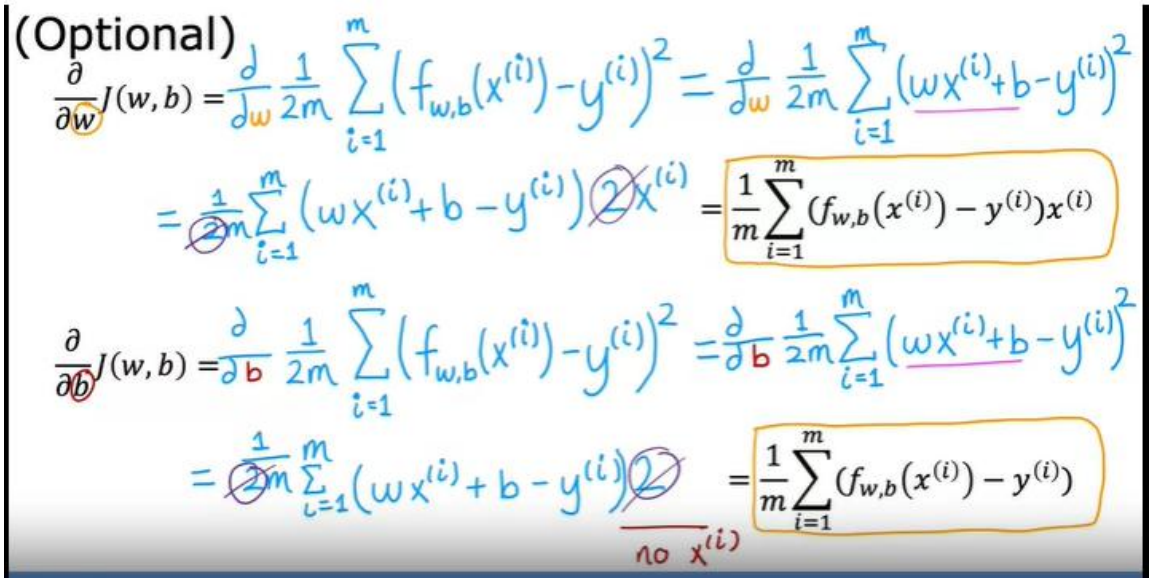




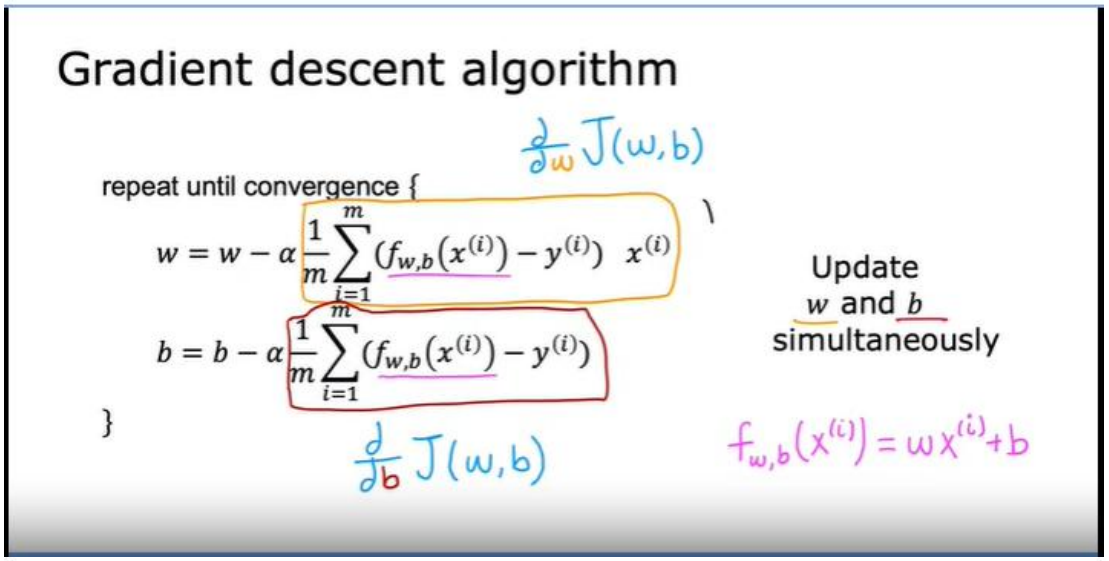
1.4 - Train the model with gradient descent : 1.4.5 - Gradient descent for linear regression :

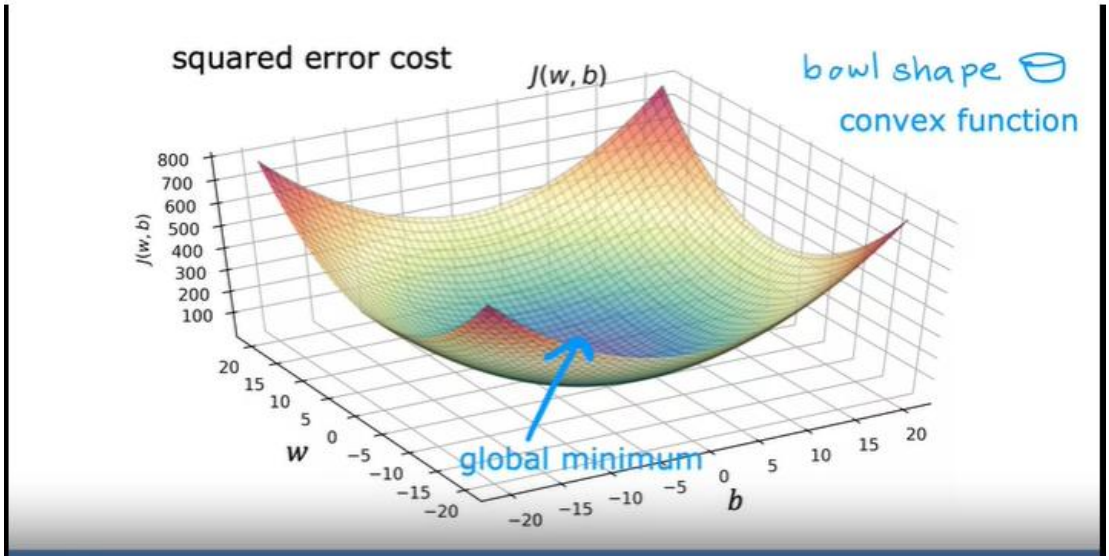


Gradient descent for linear regression : with calculus (partial differentiation) steps

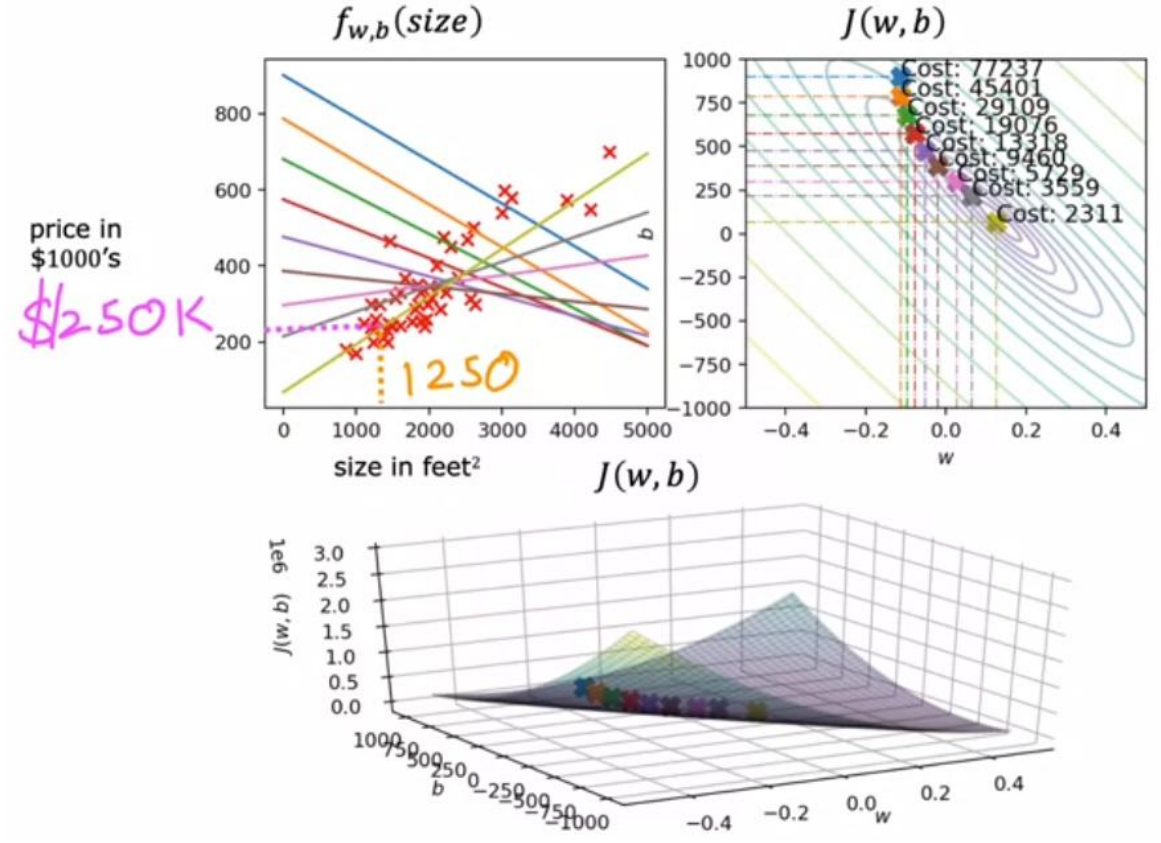


Gradient descent for linear regression : Final compilation

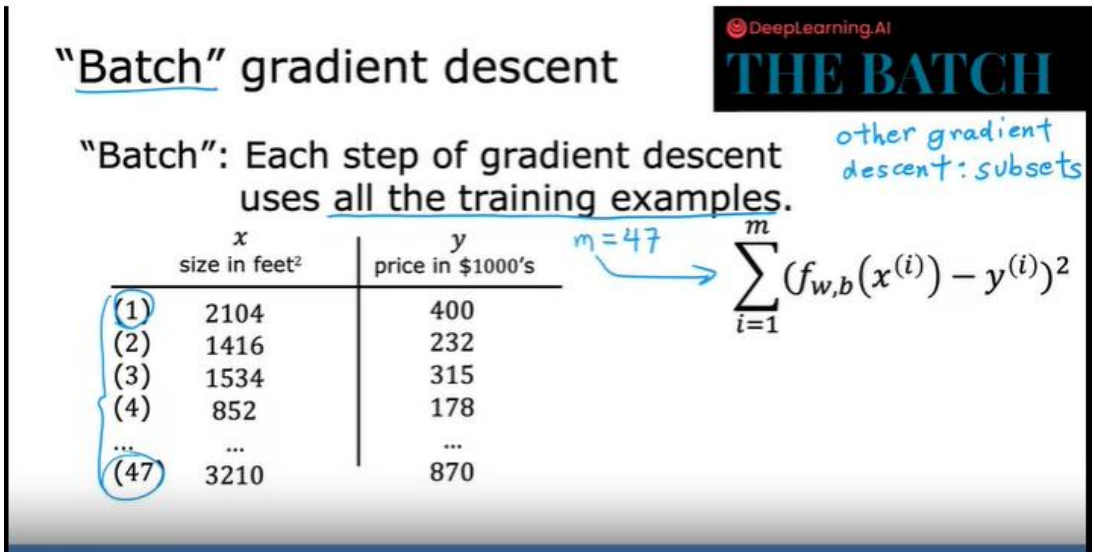


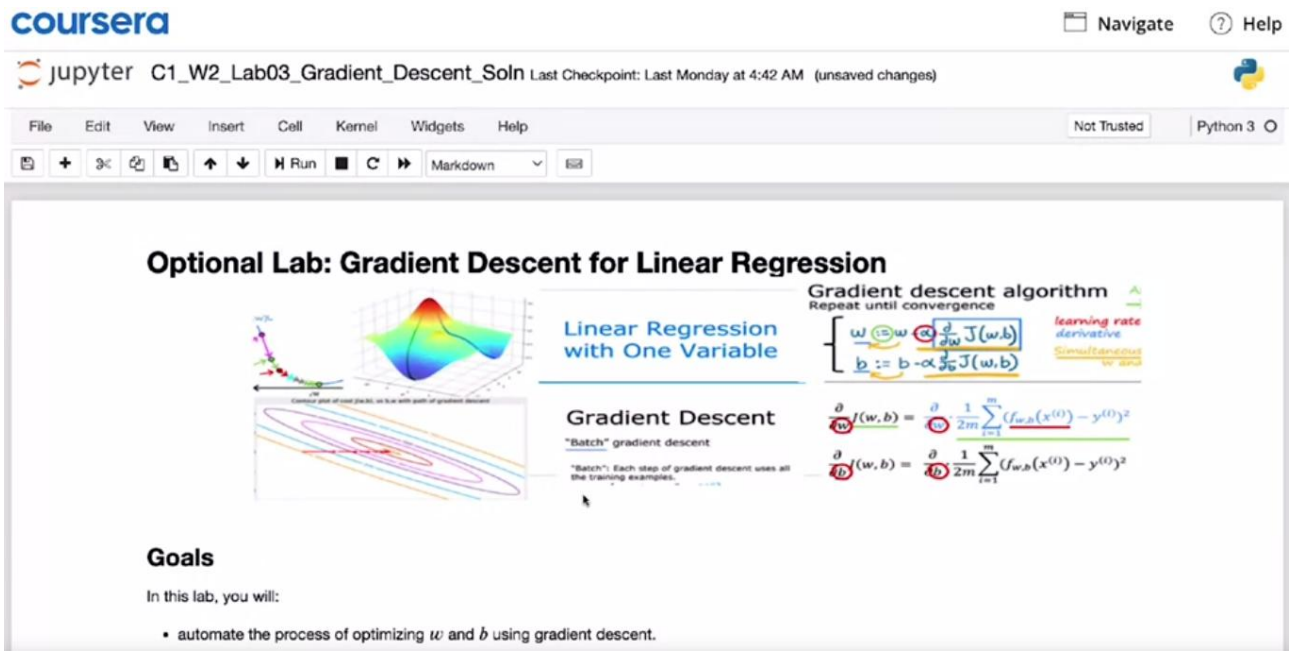


1.4 - Train the model with gradient descent : 1.4.6 - Running gradient descent :



Batch gradient descent :

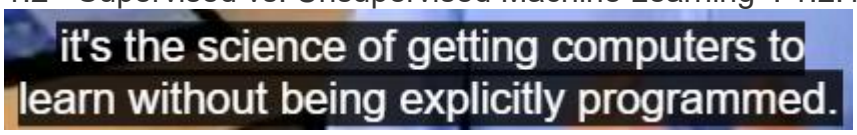




Week (Part) 2 : Regression with multiple input variables

1.3 - Regression Model : 1.3.6 - Visualization examples :

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



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

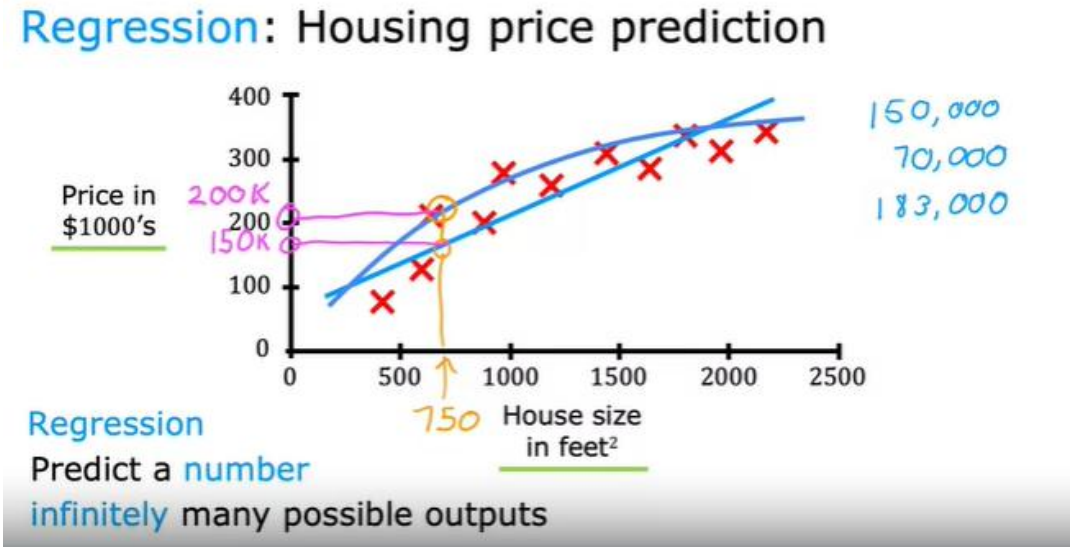
Learning Objectives

- 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

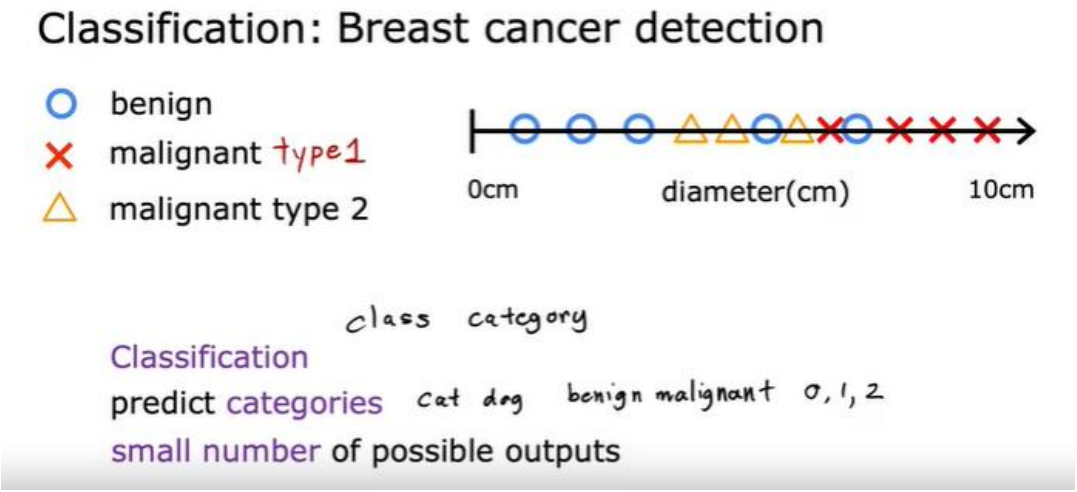
supervised learning :

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

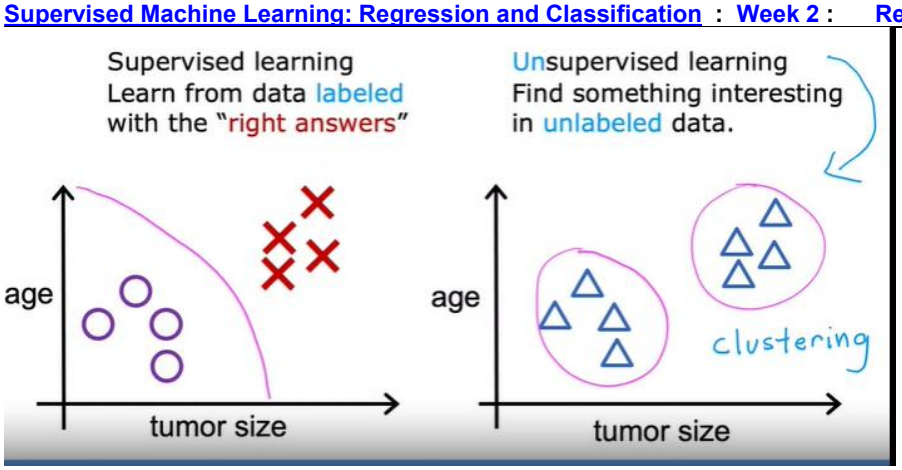
1.2.2 - supervised learning :part 1- Regression



1.2.3 - supervised learning :part 2- Classification



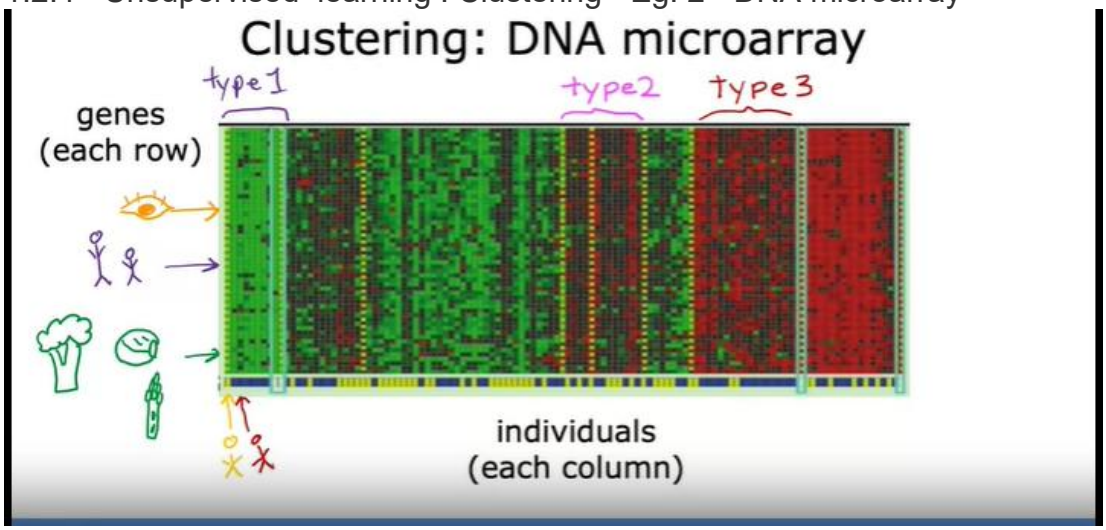
1.2.4 - Unsupervised learning :part 1-



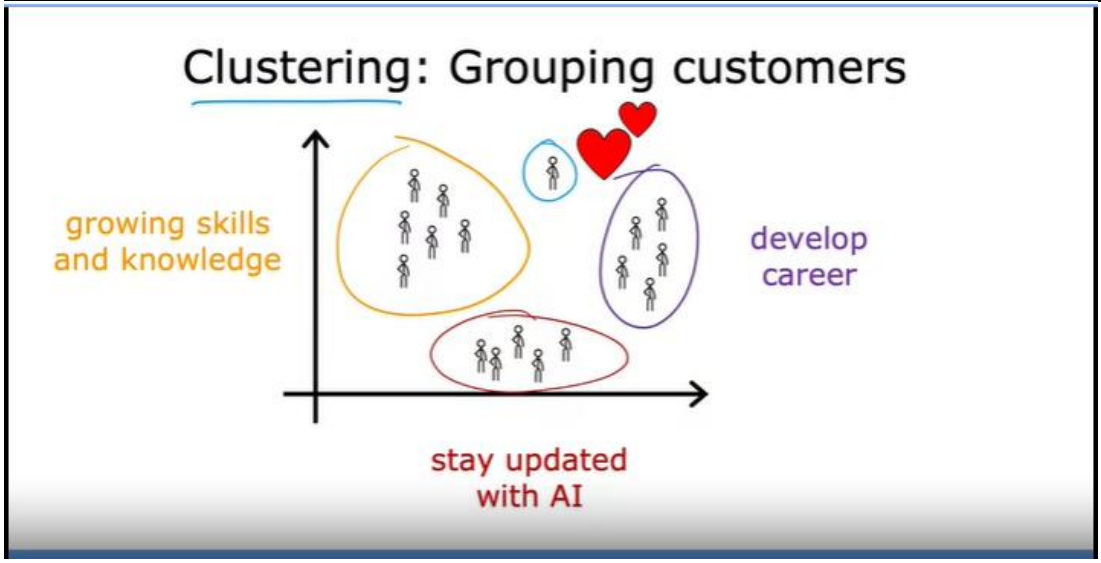
1.2.4 - Unsupervised learning : Clustering - Eg. 1 - Google News

- 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

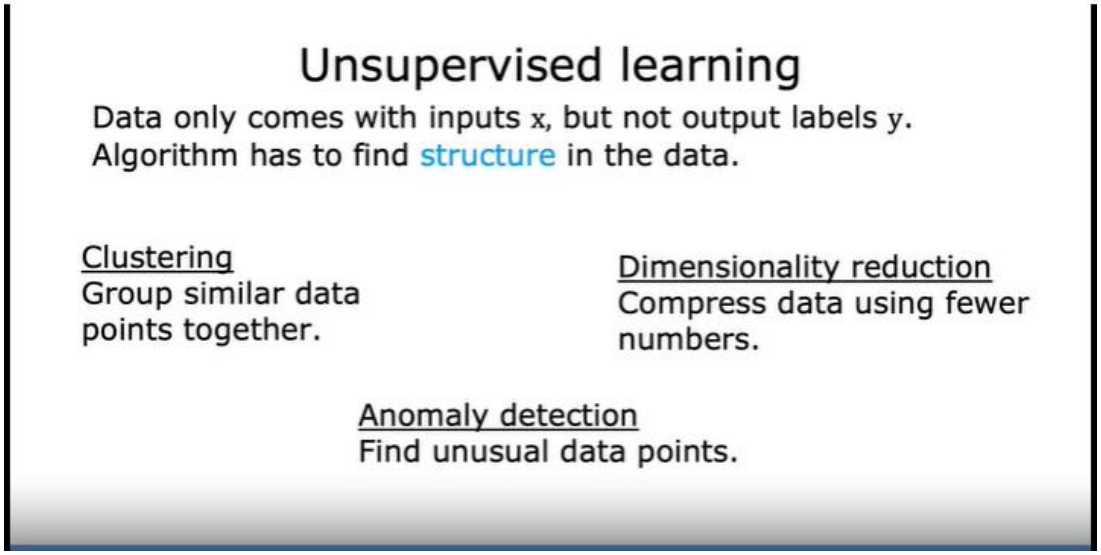
1.2.4 - Unsupervised learning : Clustering - Eg. 2 - DNA microarray



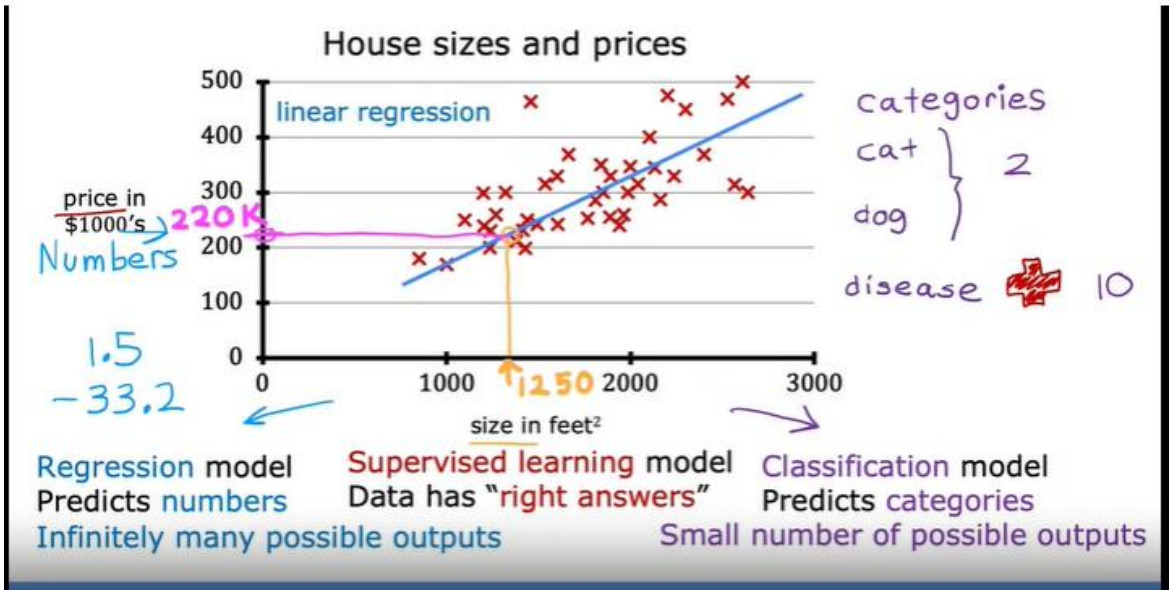
1.2.4 - Unsupervised learning : Clustering - Eg. 3 - Grouping Customers

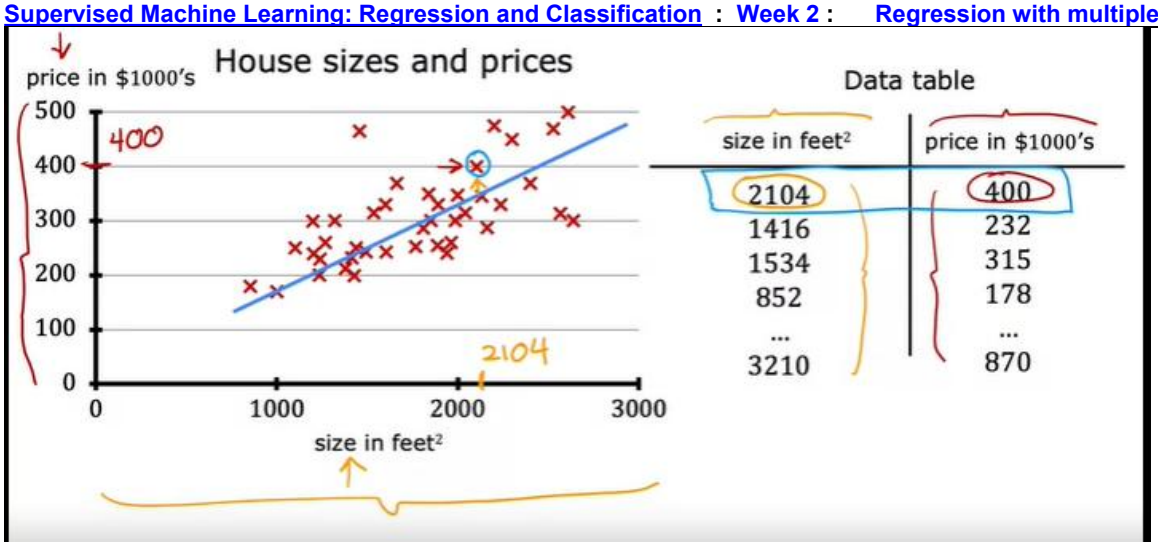


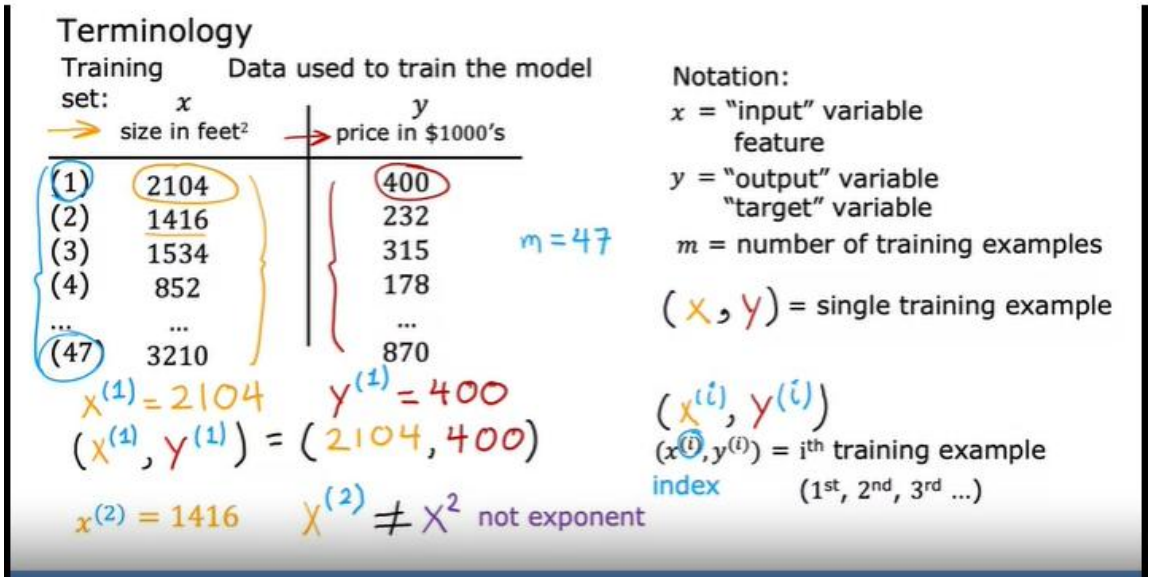
1.2.5 - Unsupervised learning :part 2-



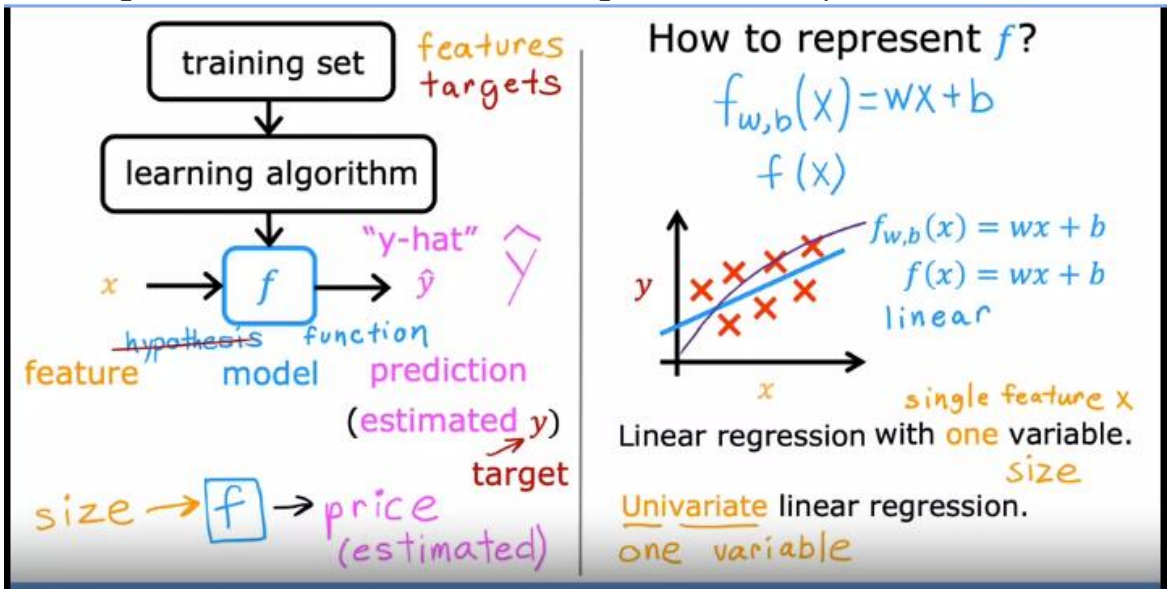
1.3 - Regression Model : 1.3.1 - Linear regression model part 1 :



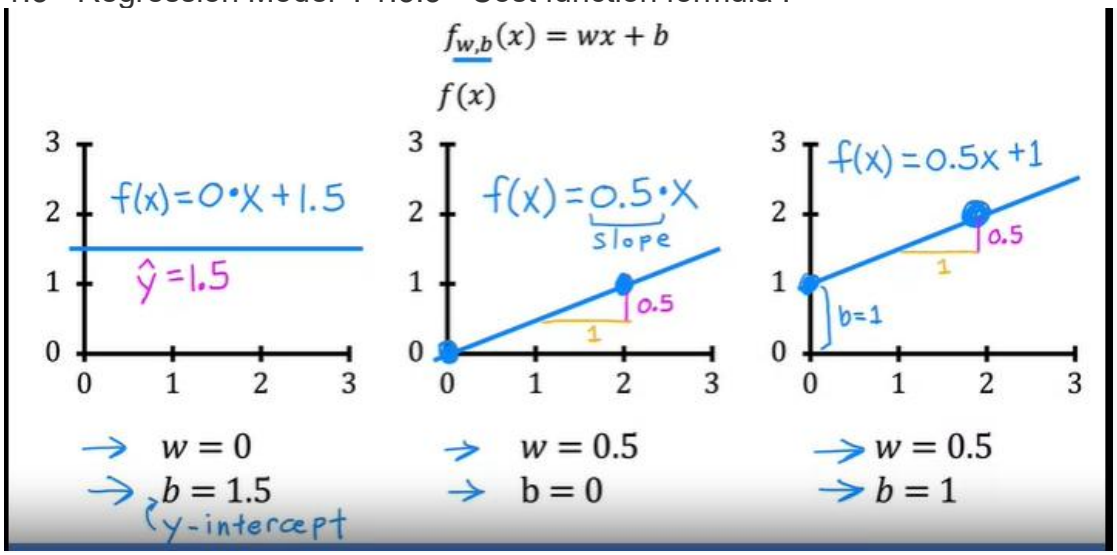




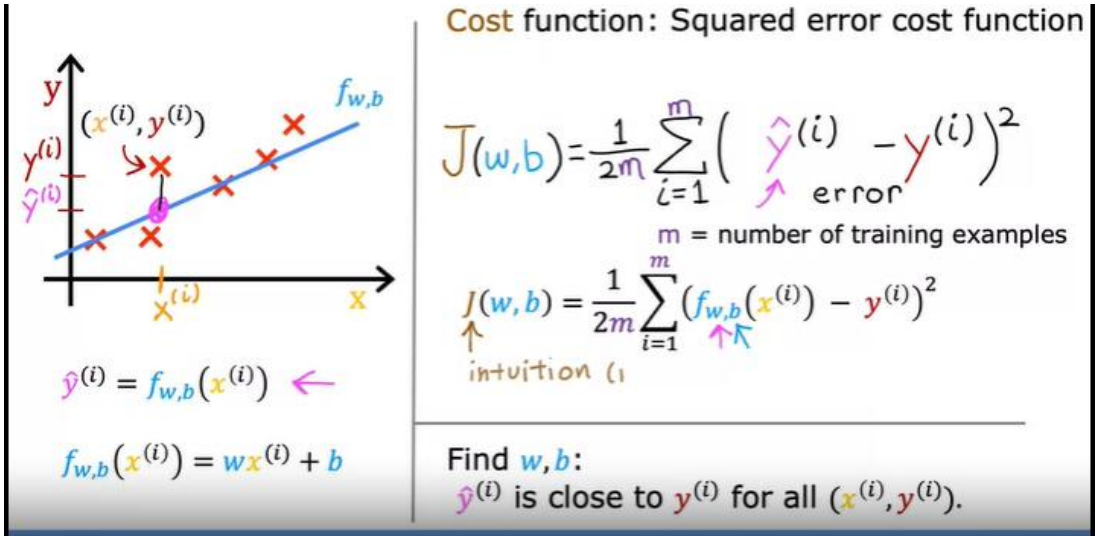
1.3 - Regression Model : 1.3.2 - Linear regression model part 2 :



1.3 - Regression Model : 1.3.3 - Cost function formula :

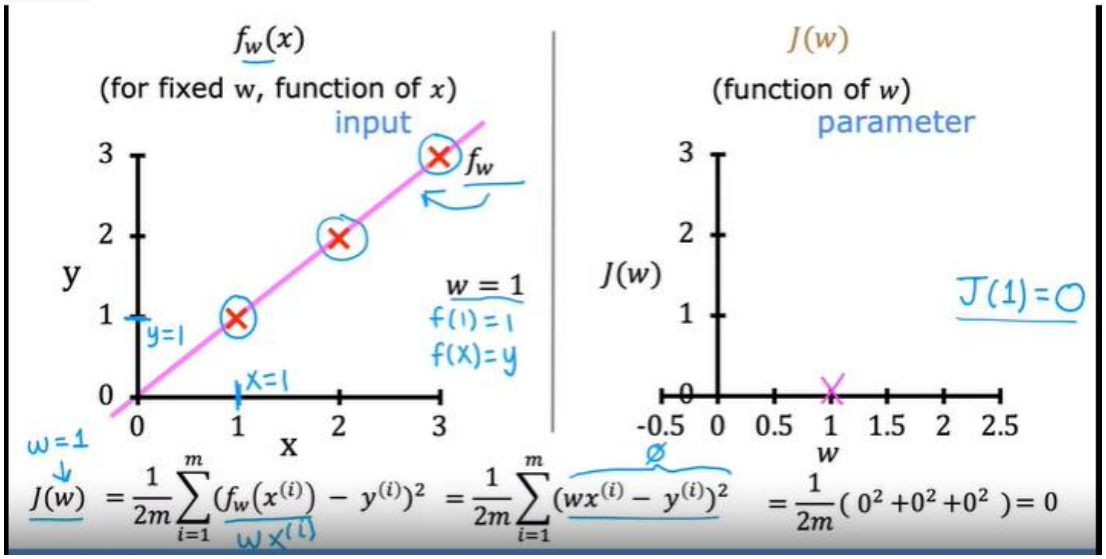


Cost function formula :

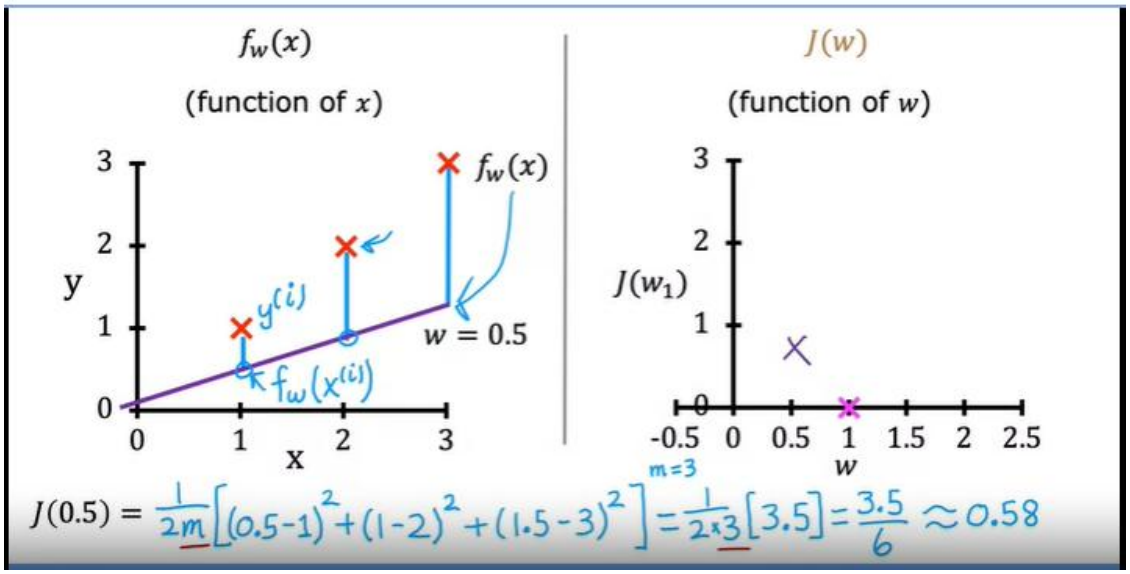


1.3 - Regression Model : 1.3.5 - Visualizing the cost function :

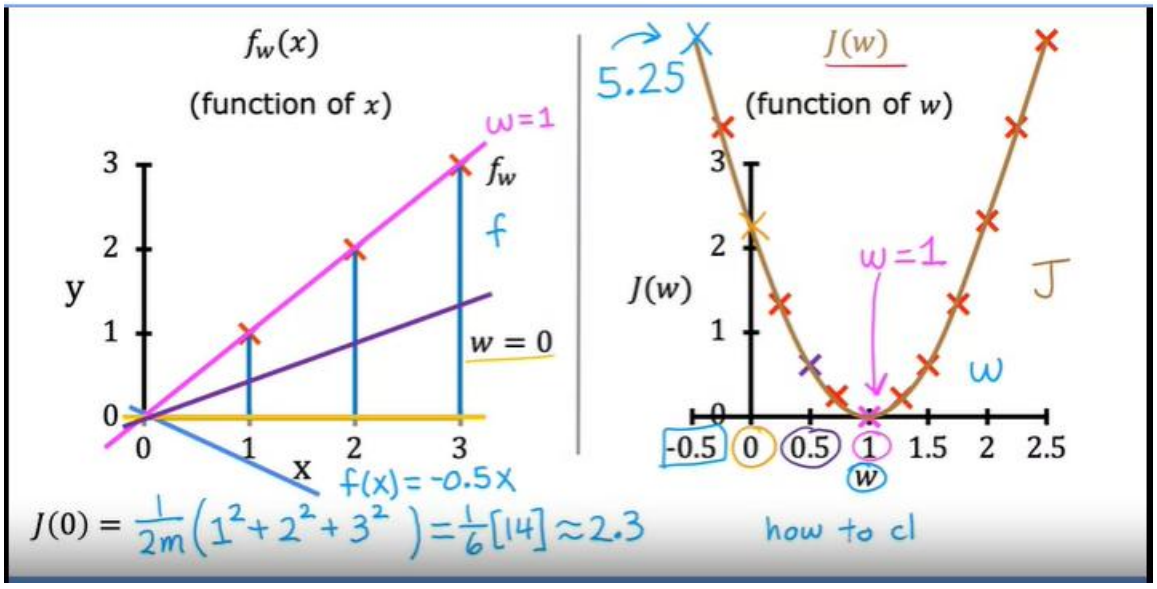
W = 1



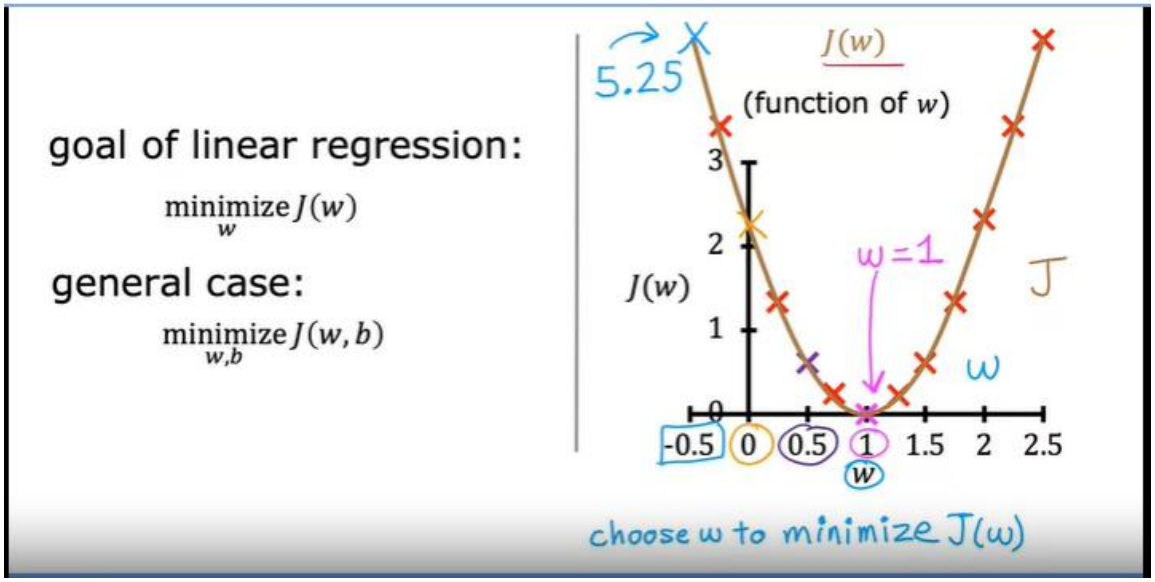
W = 0.5



W = 0, -0.5, …

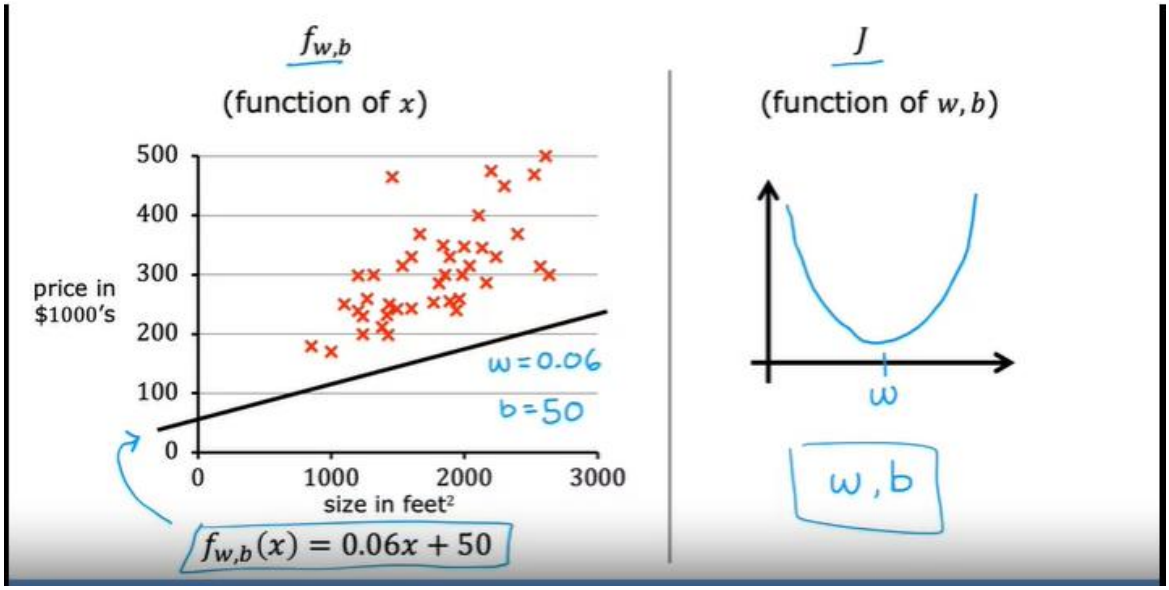


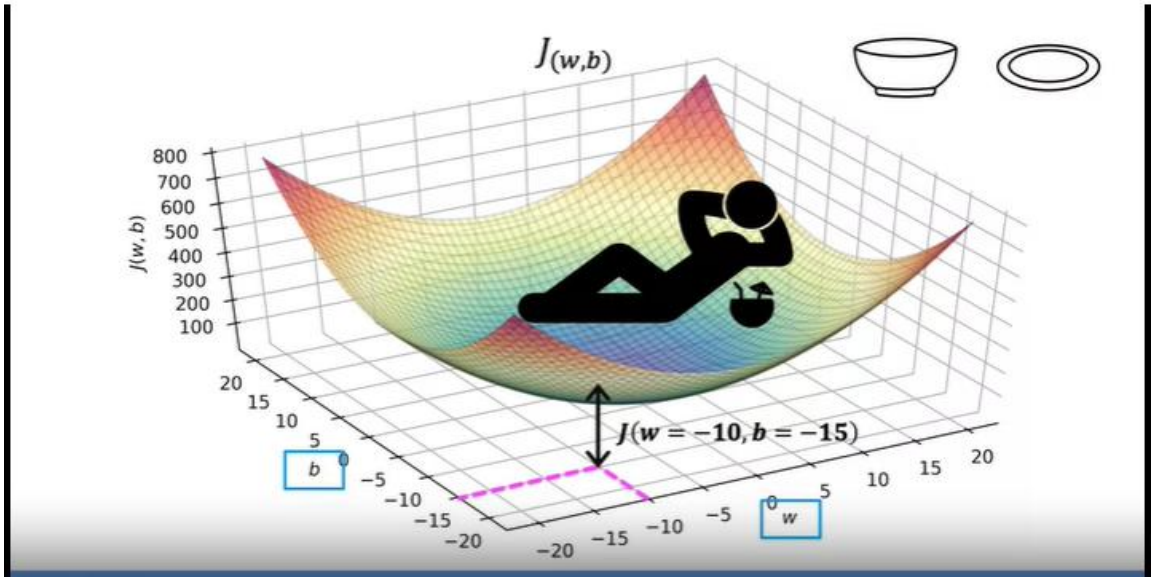
Goal of linear regression : To minimize the cost function

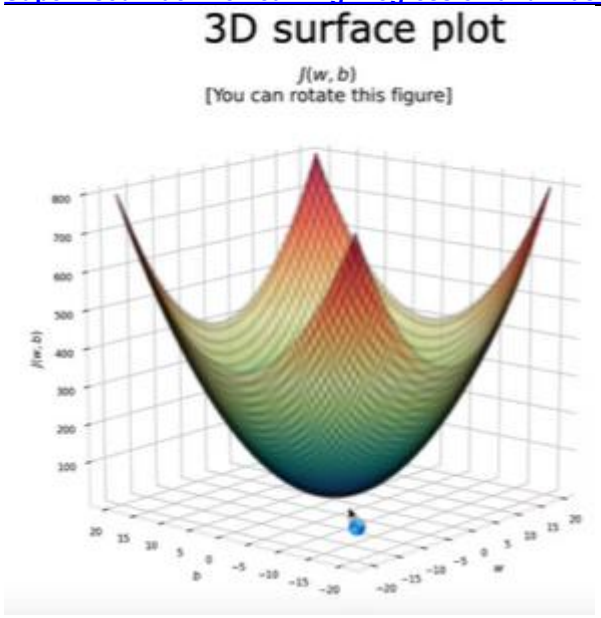


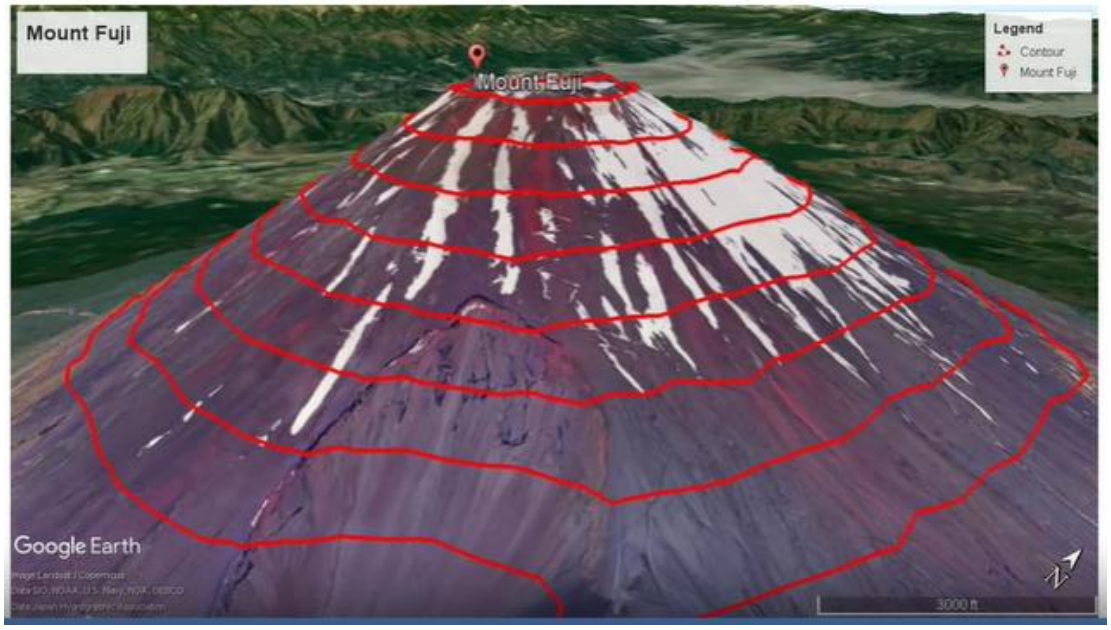
1.3 - Regression Model : 1.3.6 - Visualization examples :

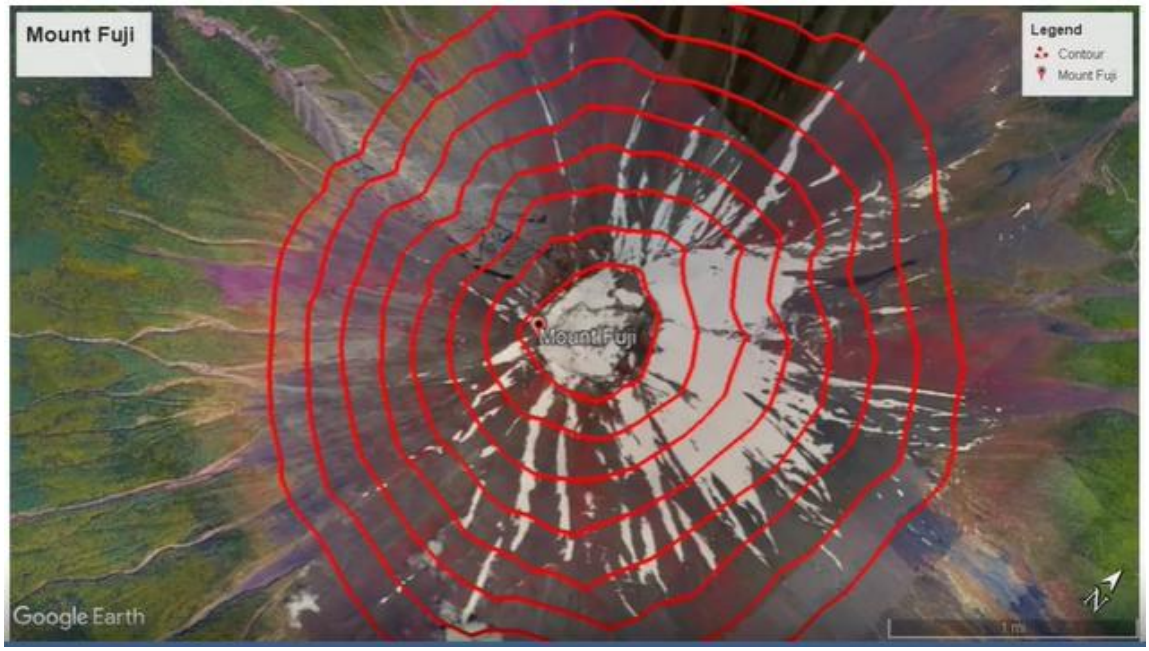
Model	$f_{w,b}(x) = wx + b$
Parameters	w, b ~~before: b=0~~
Cost Function	$J(w,b) = \frac{1}{2m}\sum_{i=1}^{m}(f_{w,b}(x^{(i)}) - y^{(i)})^2$
Objective	$\underset{w,b}{\text{minimize}}\, J(w,b)$

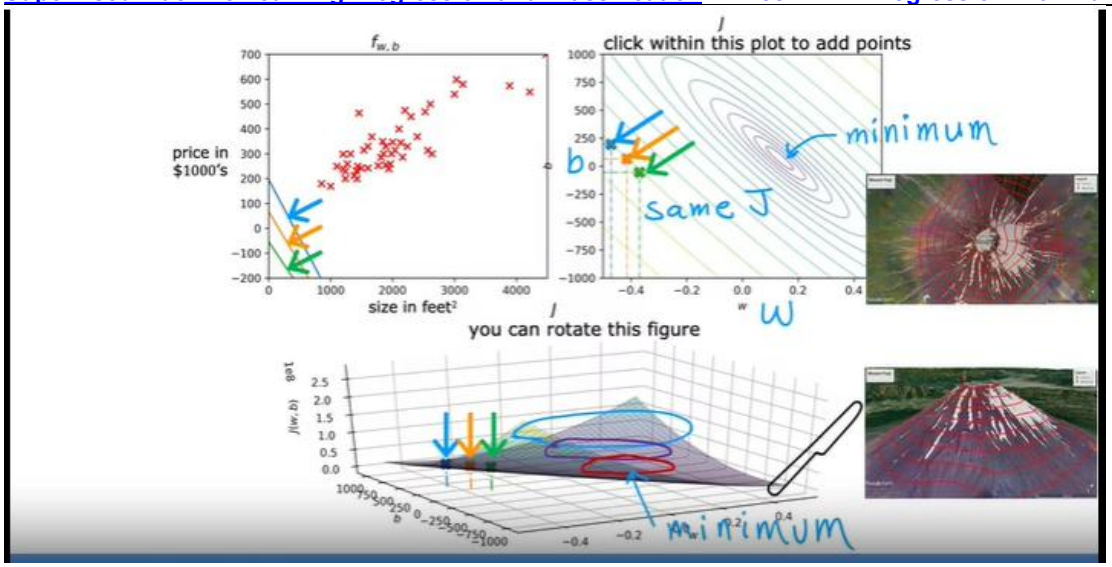





3D surface plot
J(w, b)
[You can rotate this figure]
J(w, b)
w
b

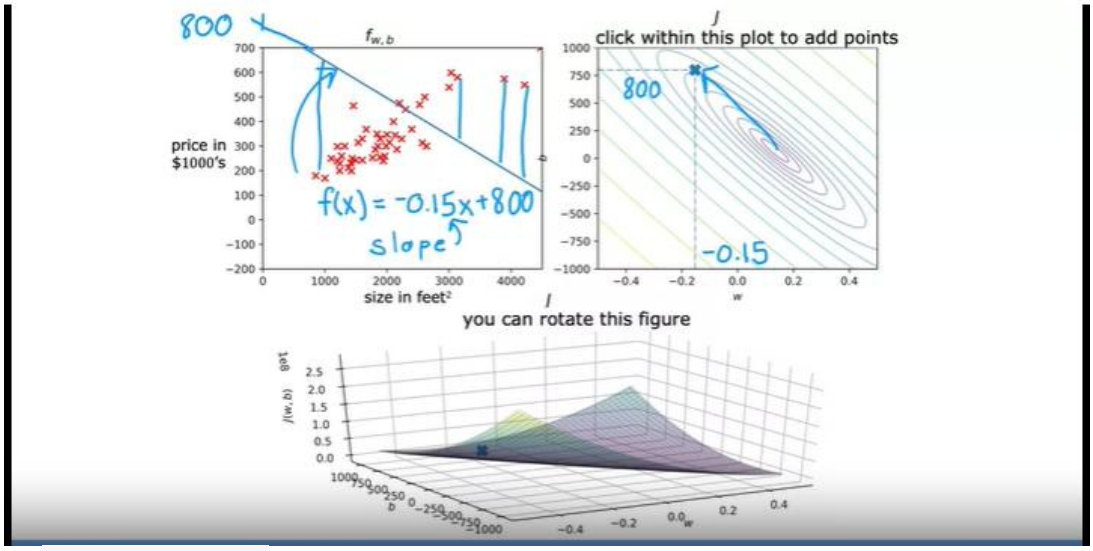

Mount Fuji
Legend
Contour
Mount Fuji
Mount Fuji
Google Earth


Mount Fuji
Legend
Contour
Mount Fuji
Mount Fuji
Google Earth

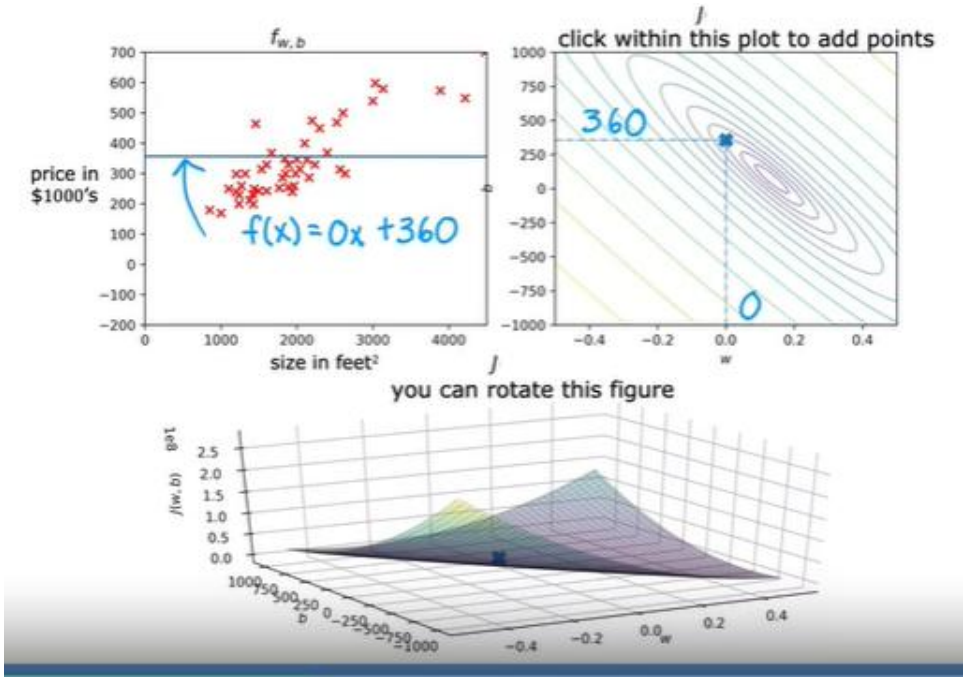


1.3 - Regression Model : 1.3.6 - Visualization examples :

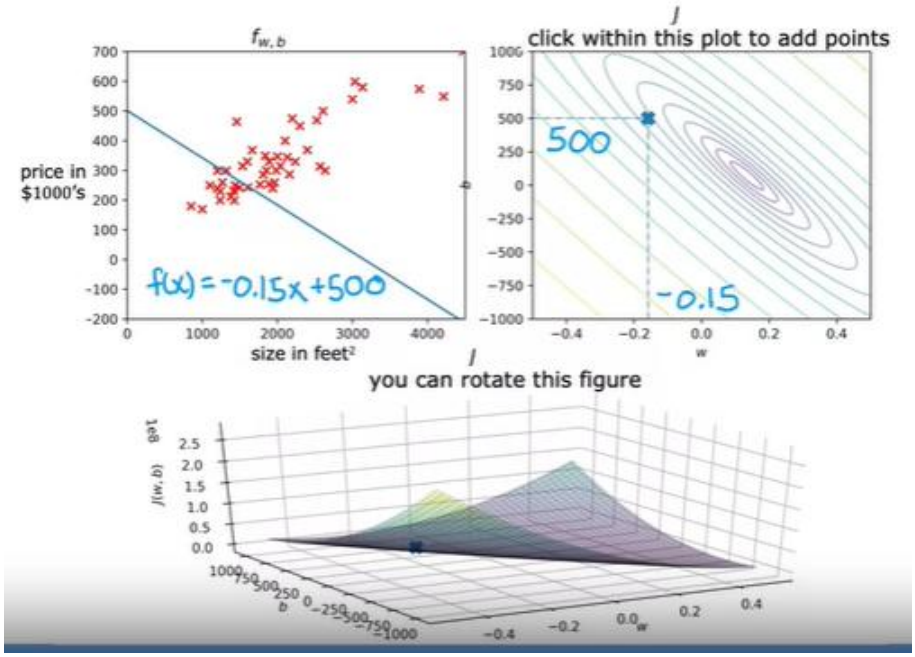
1) Bad fit



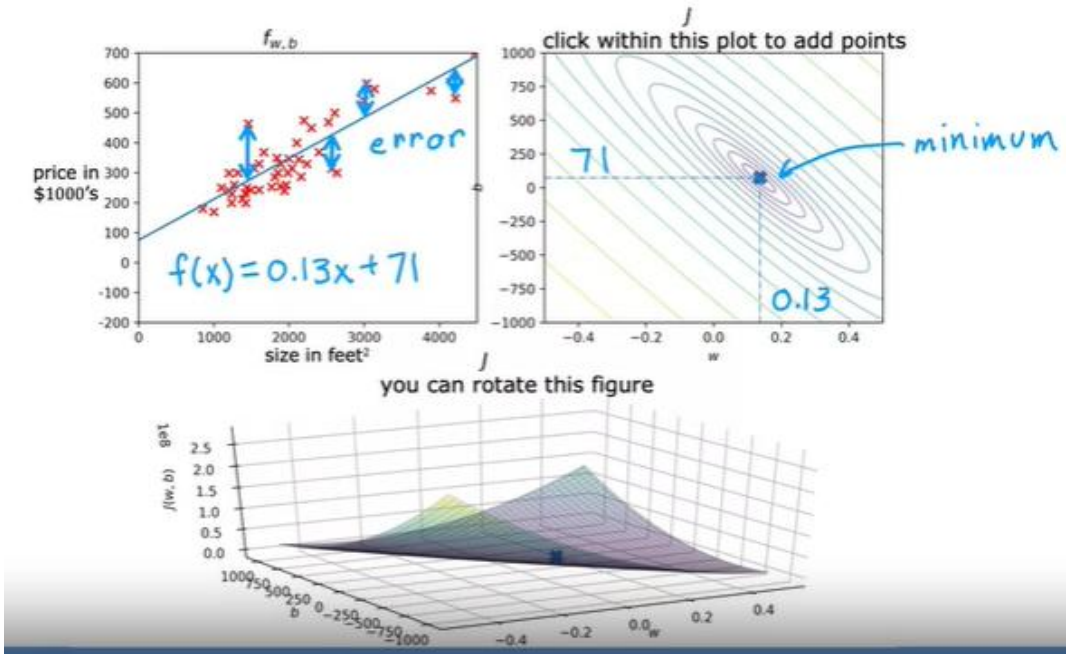
2) Not That bad fit



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

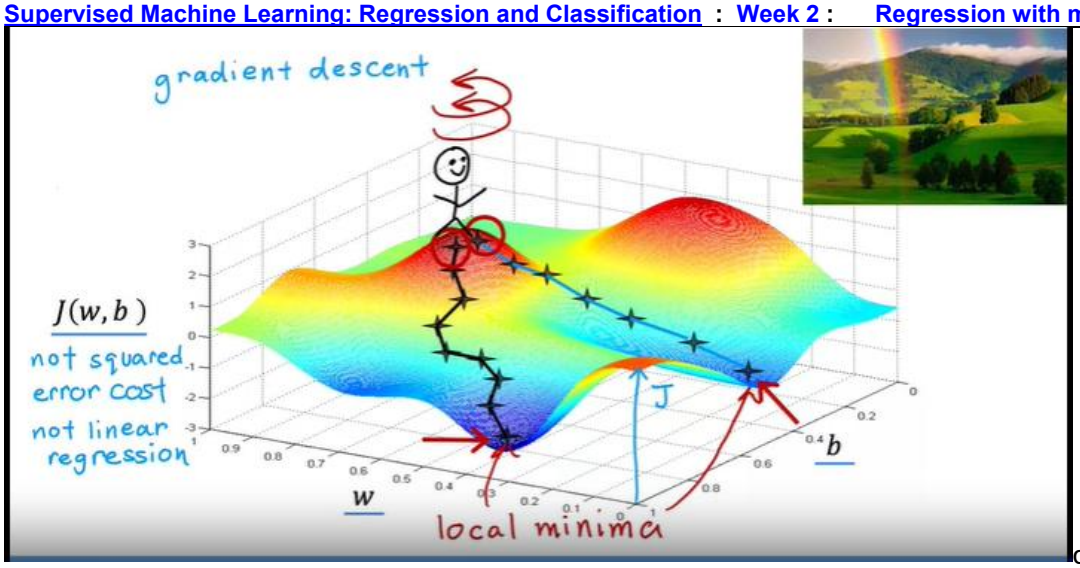
Want $\min_{w,b} J(w,b)$ $\min_{w_1,\ldots,w_n,b} J(w_1,w_2,\ldots,w_n,b)$

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

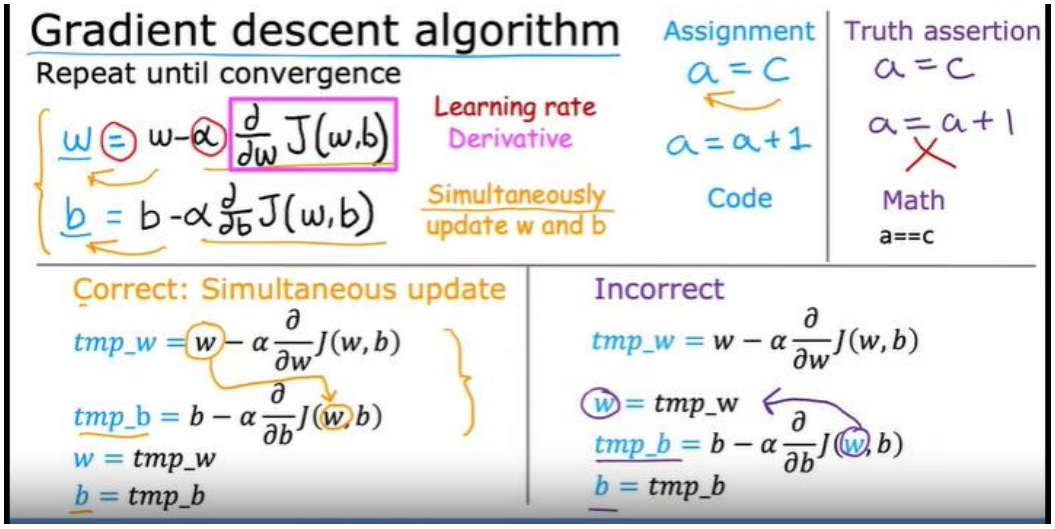


αα

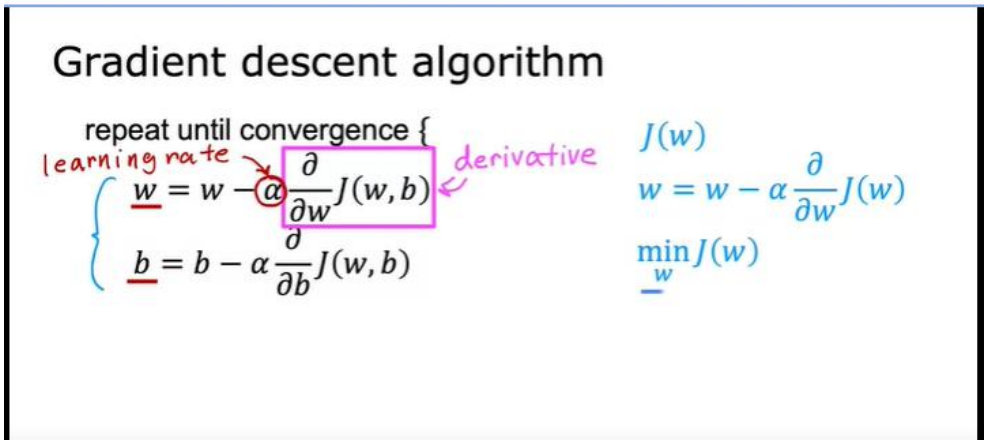
1.4 - Train the model with gradient descent : 1.4.2 - Implementing gradient descent :

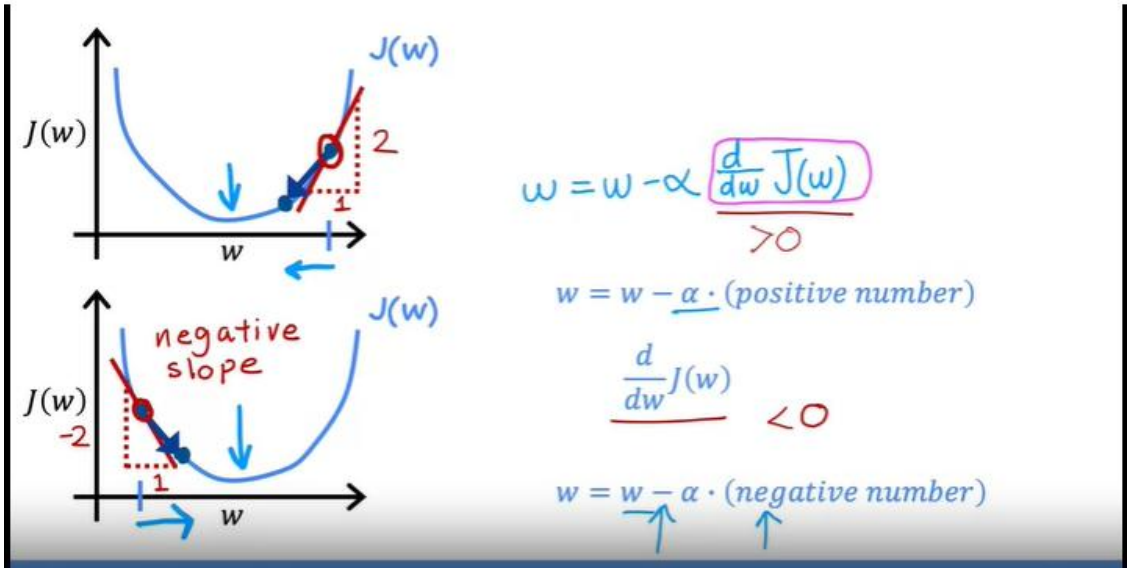
α = Learning rate, 0 to 1, (say 0.01) = size of the baby step

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

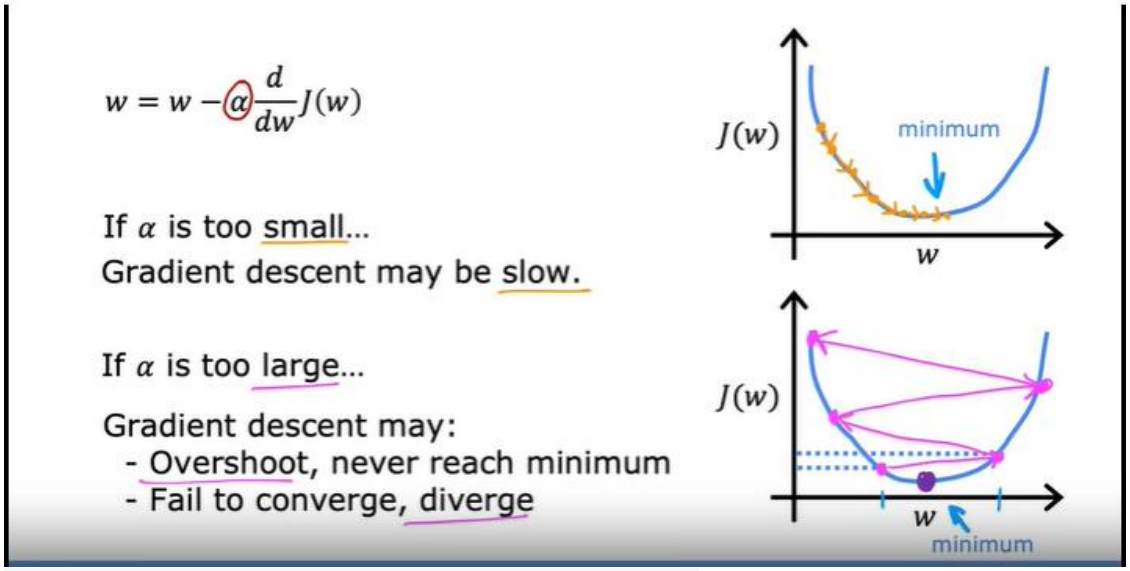


1.4 - Train the model with gradient descent : 1.4.3 - Gradient descent intuition :

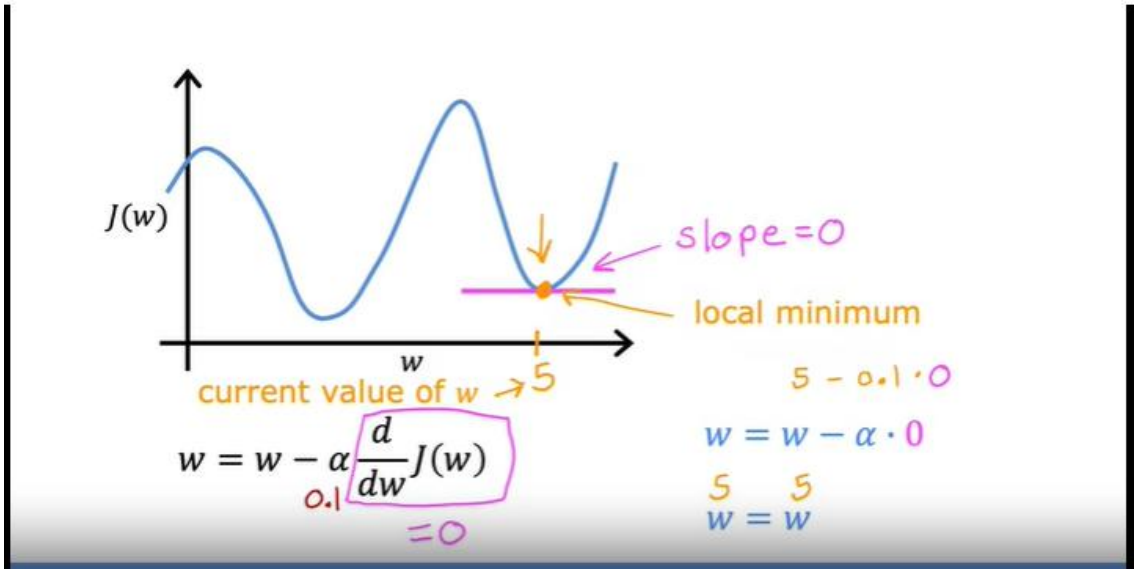


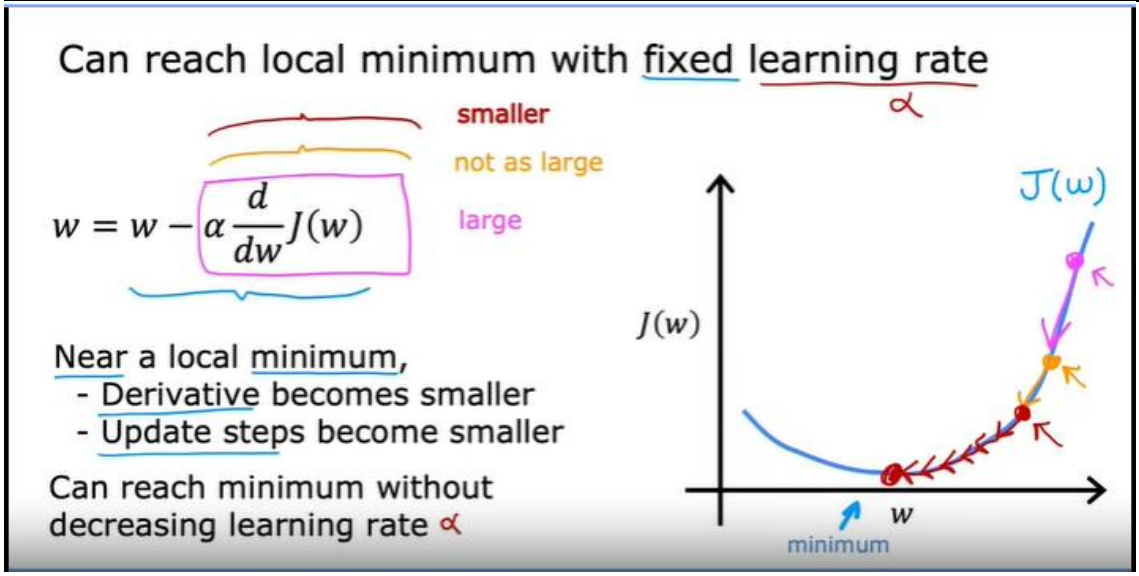


1.4 - Train the model with gradient descent : 1.4.4 - Learning rate :

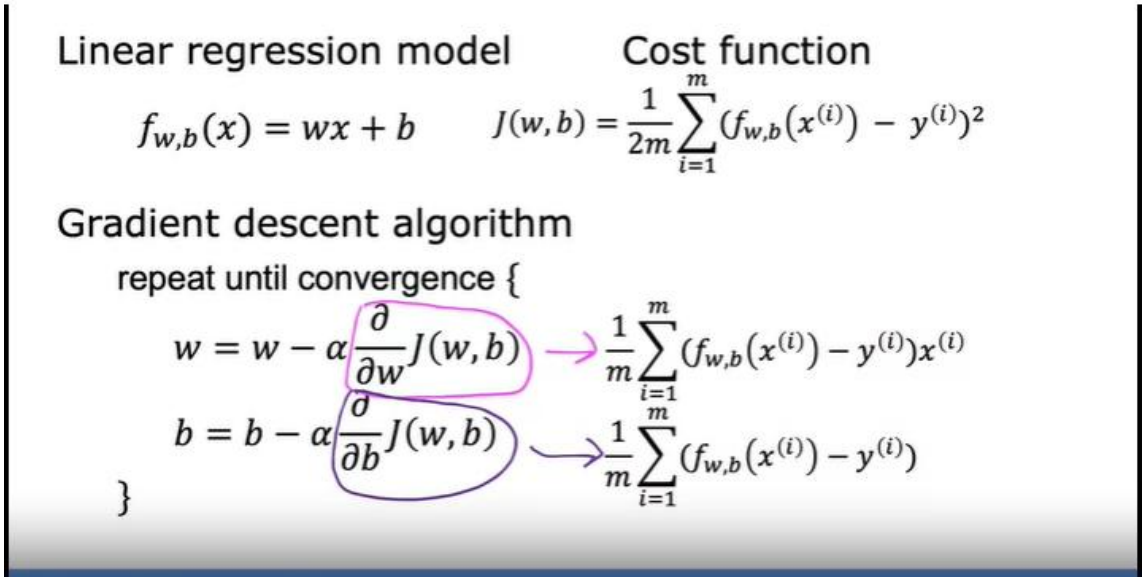


Learning rate : Tricky situation of 2 local minimas

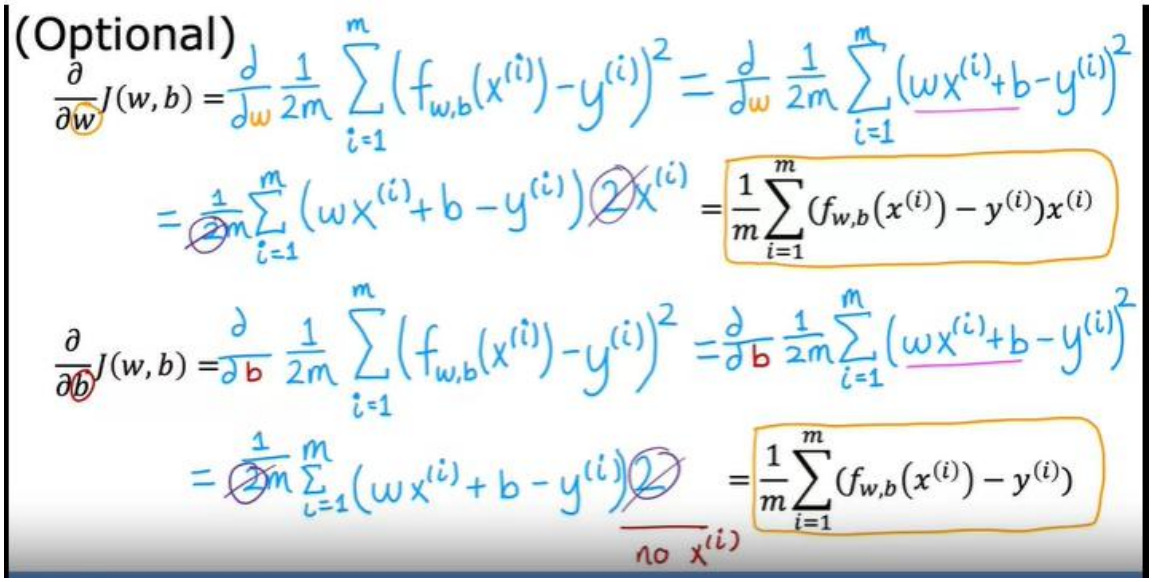




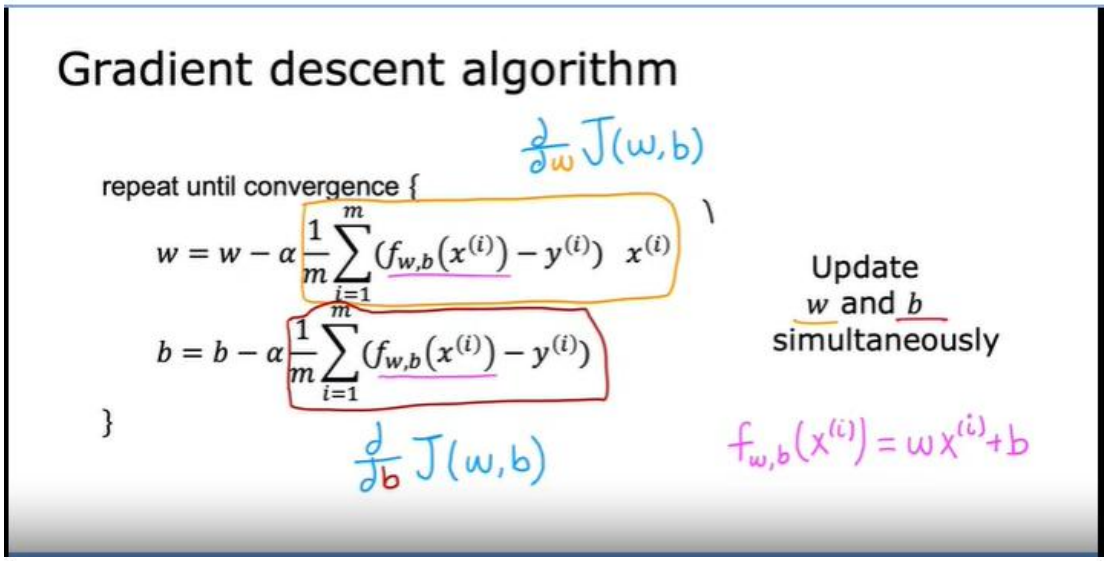
1.4 - Train the model with gradient descent : 1.4.5 - Gradient descent for linear regression :

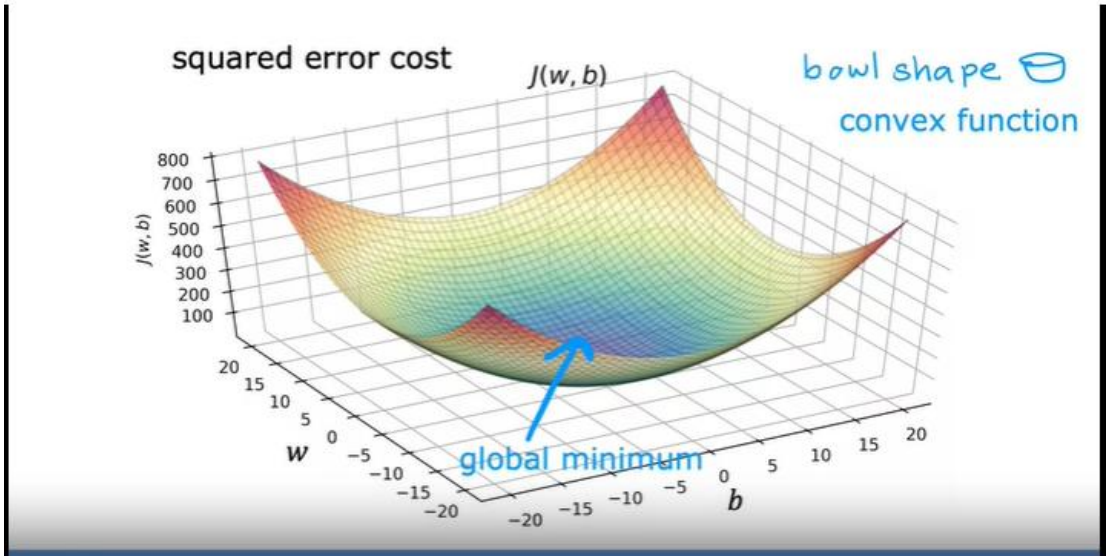


Gradient descent for linear regression : with calculus (partial differentiation) steps

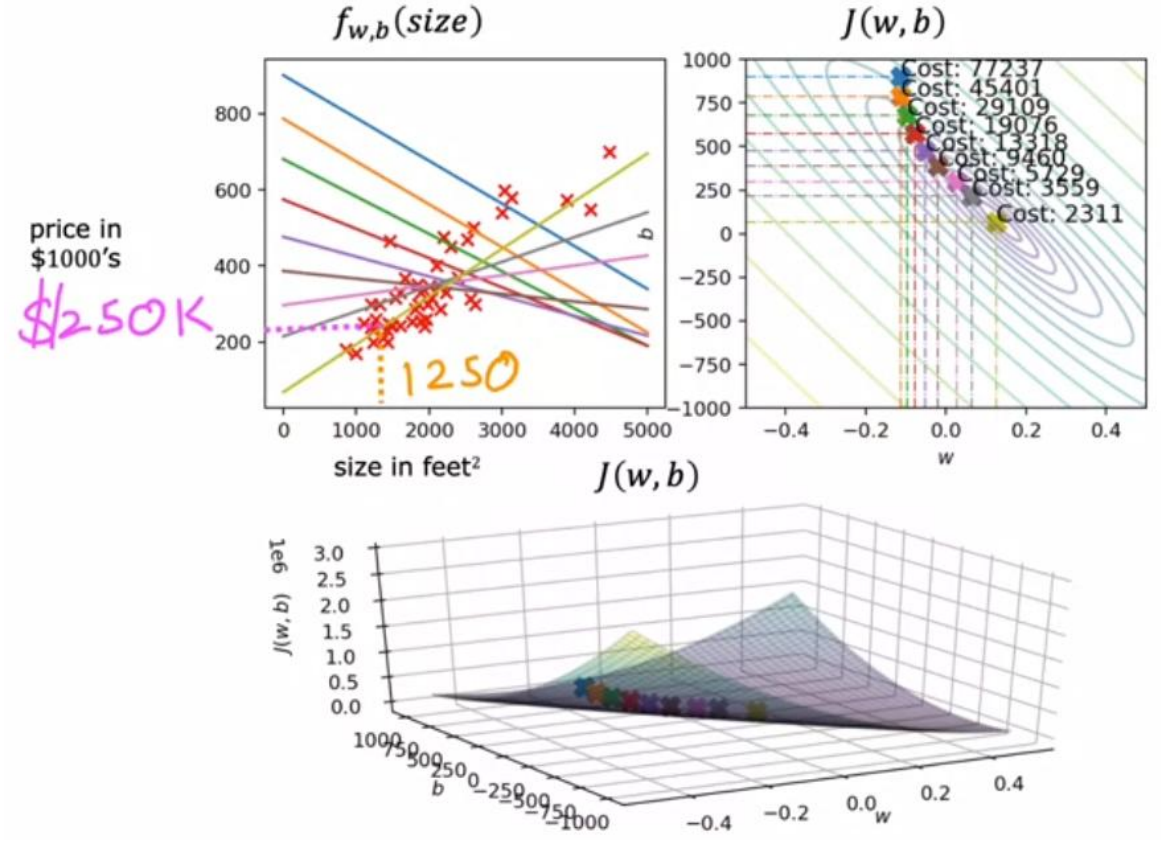


Gradient descent for linear regression : Final compilation

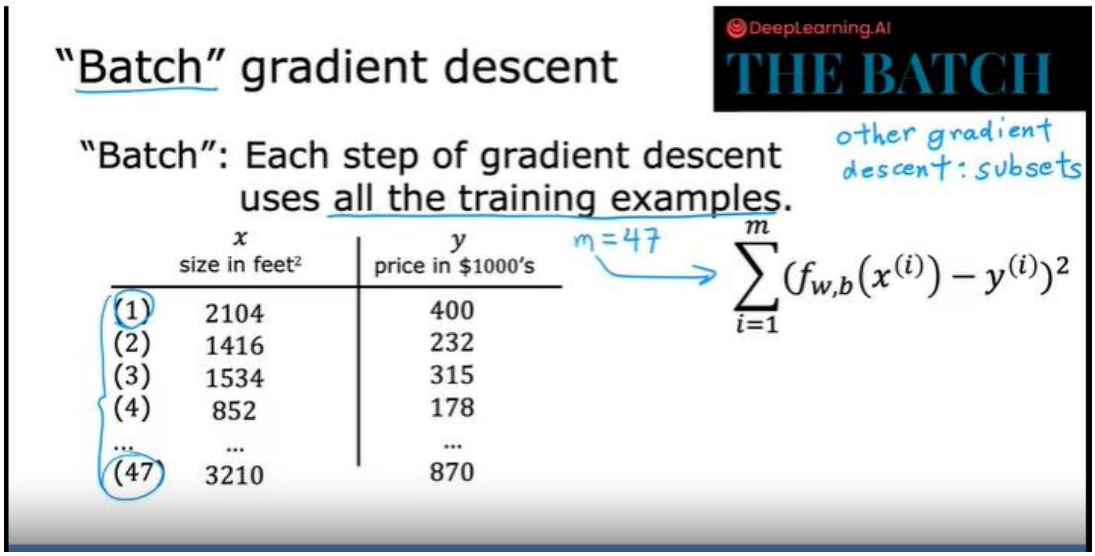


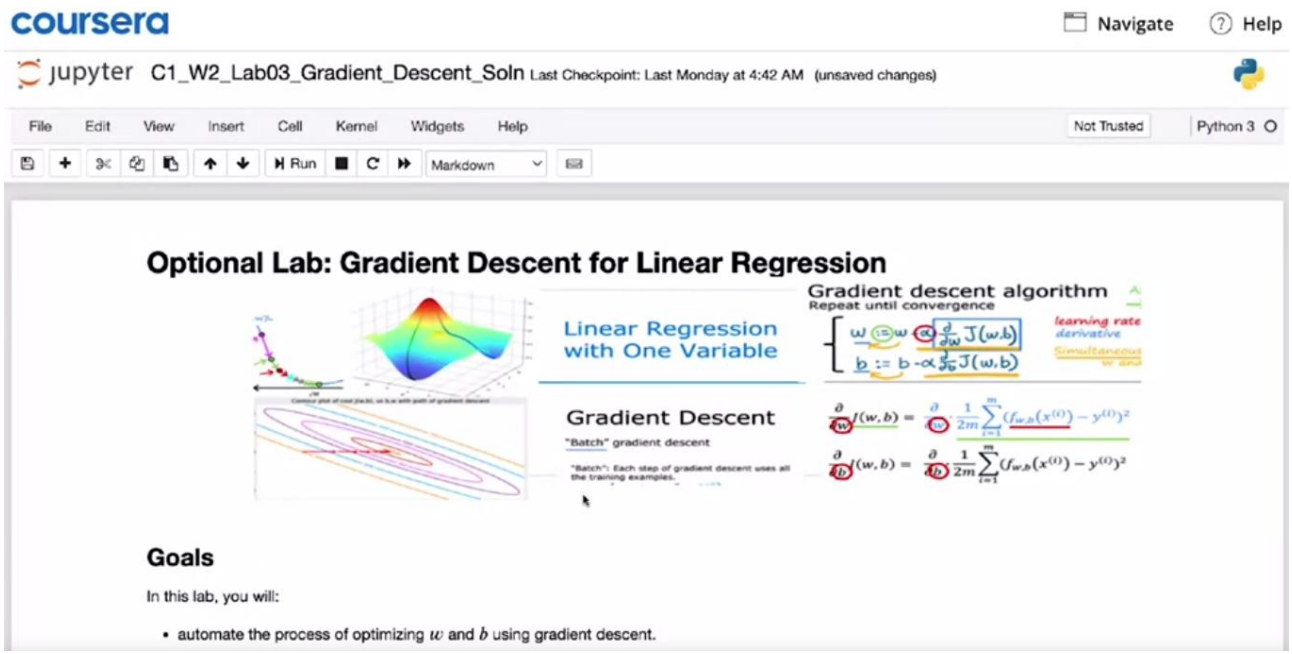


1.4 - Train the model with gradient descent : 1.4.6 - Running gradient descent :



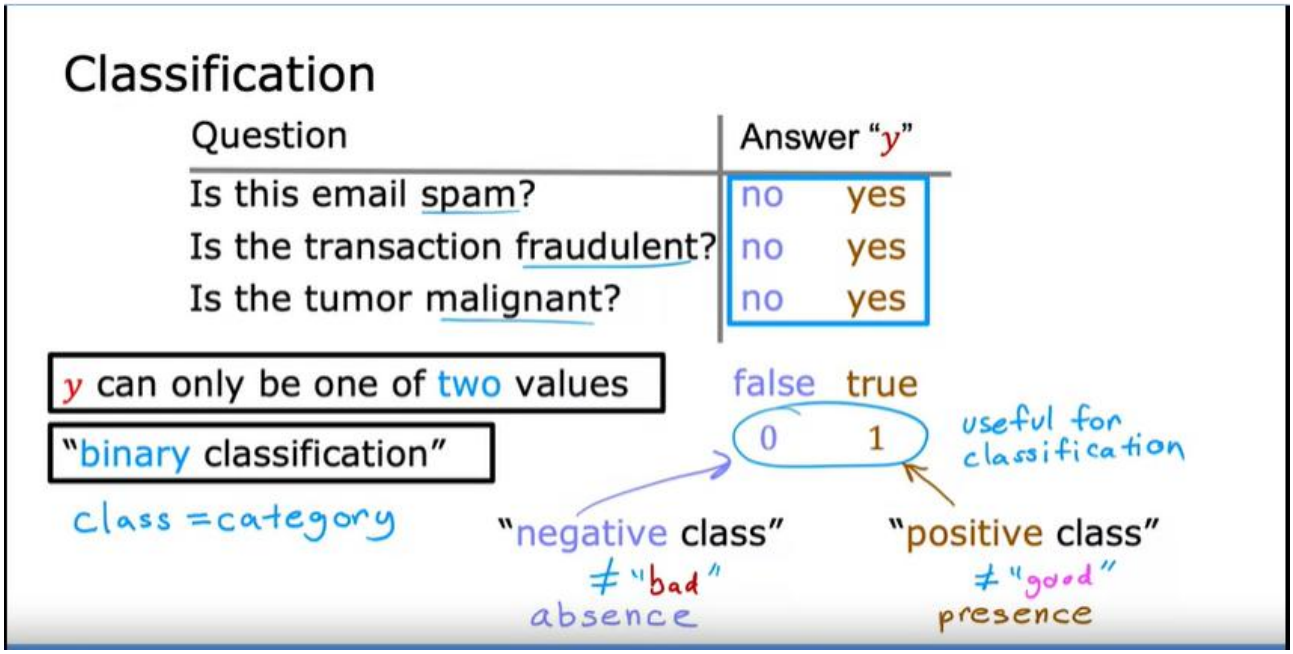
Batch gradient descent :

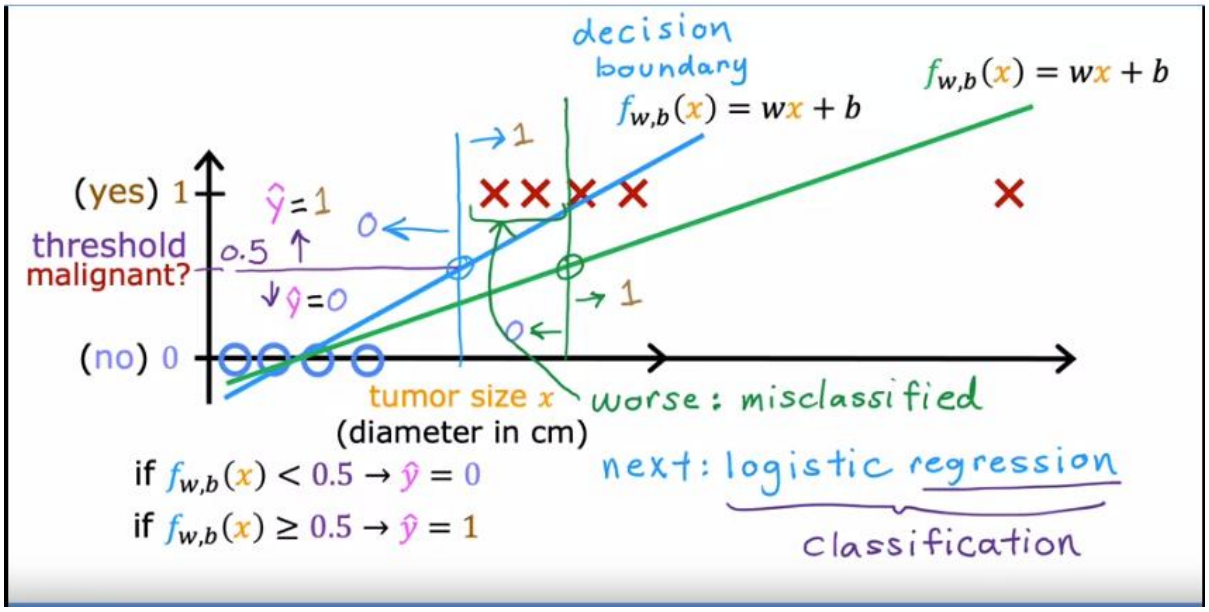




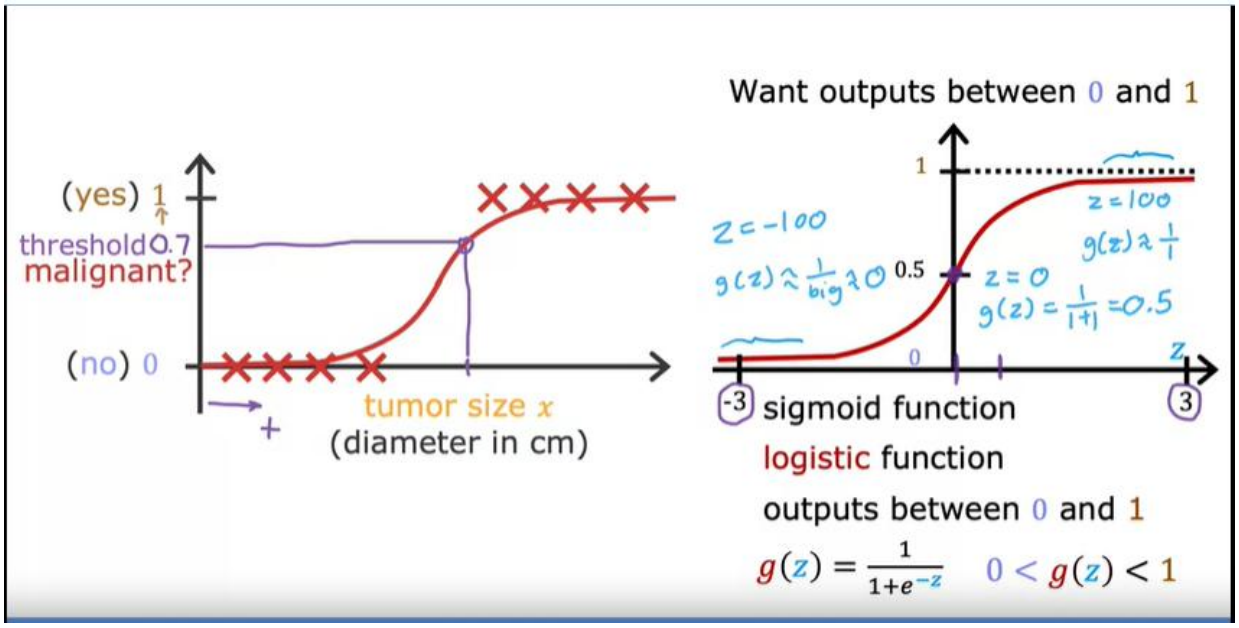
Week (Part) 3 : Classification with logistic regression

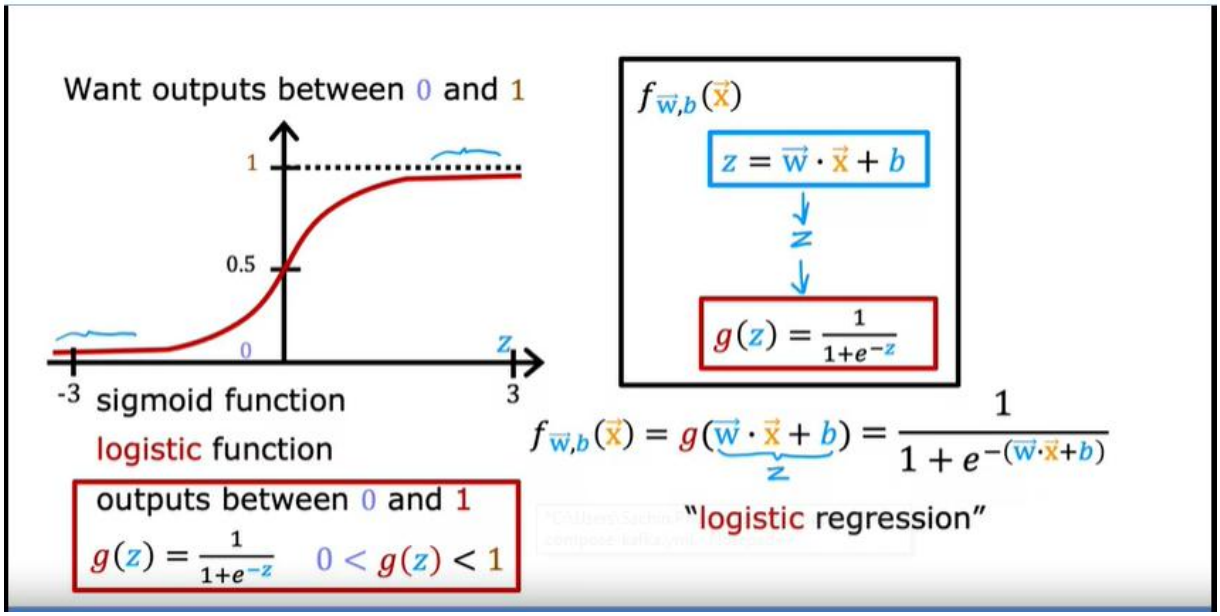
3.1 - Classification with logistic regression : 3.1.1 - Motivation :





.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

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

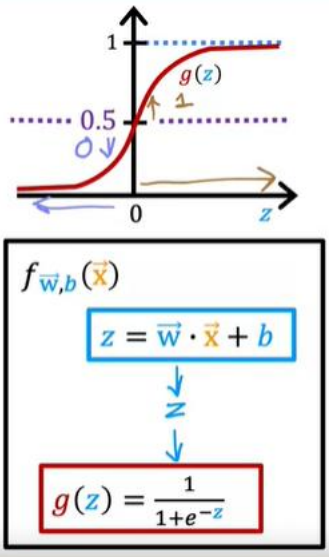
Probability that y is 1, given input $\vec{x}$, parameters $\vec{w}, b$

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



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

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

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

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

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

$g(z) \geq 0.5$

$z \geq 0$

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

$\hat{y}=1$ $\hat{y}=0$

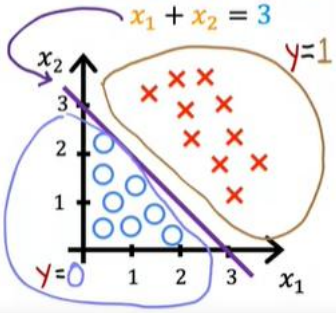
Decision boundary

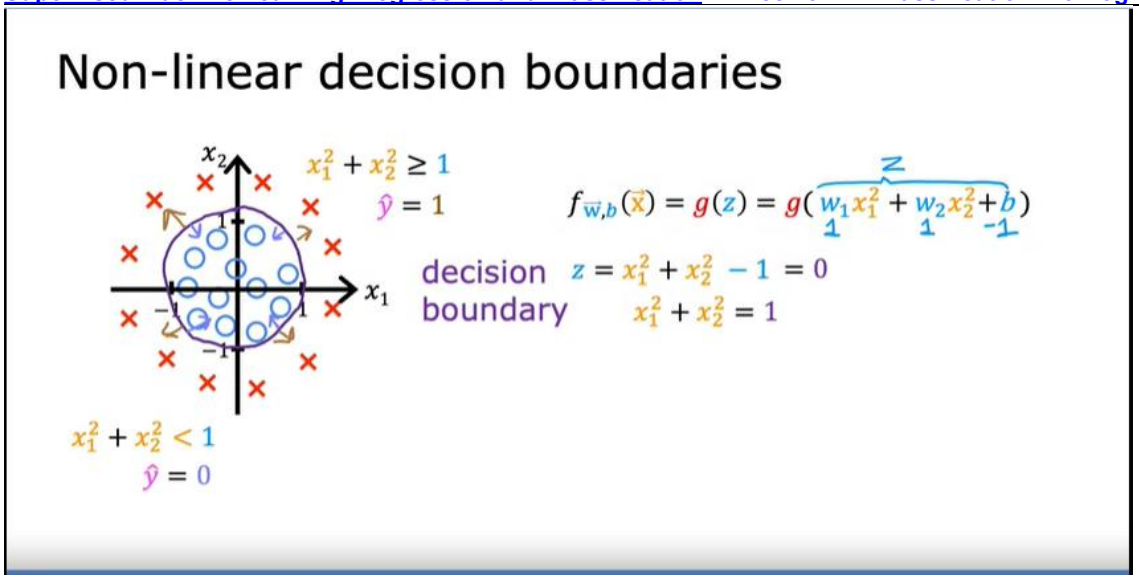
$$f_{\vec{w},b}(\vec{x}) = g(z) = g(w_1x_1 + w_2x_2 + b)$$

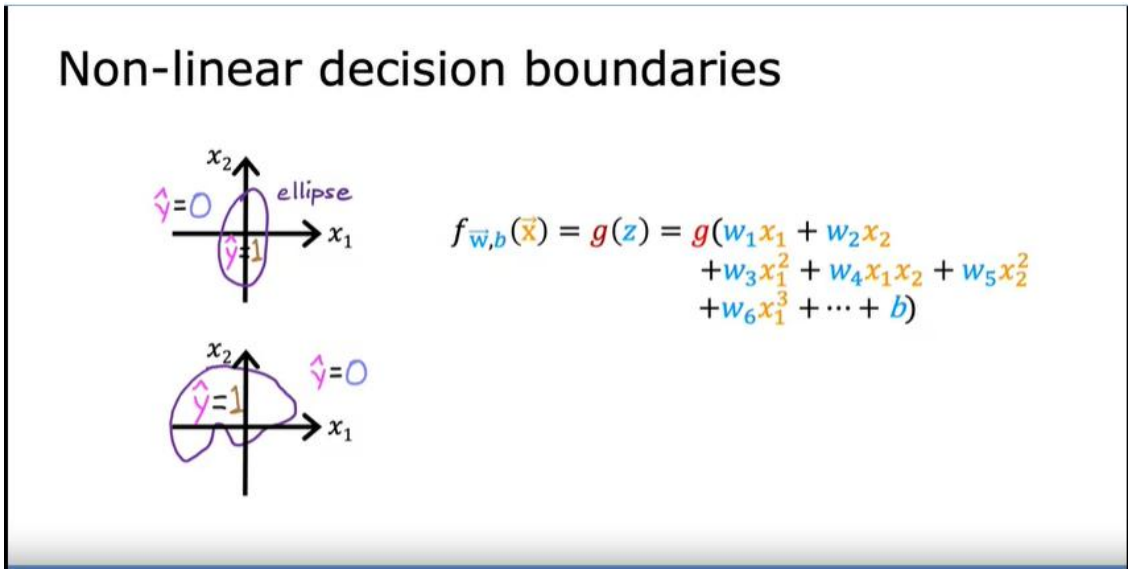
Decision boundary $z = \vec{w}\cdot\vec{x}+b = 0$

$z = x_1 + x_2 - 3 = 0$

$x_1 + x_2 = 3$

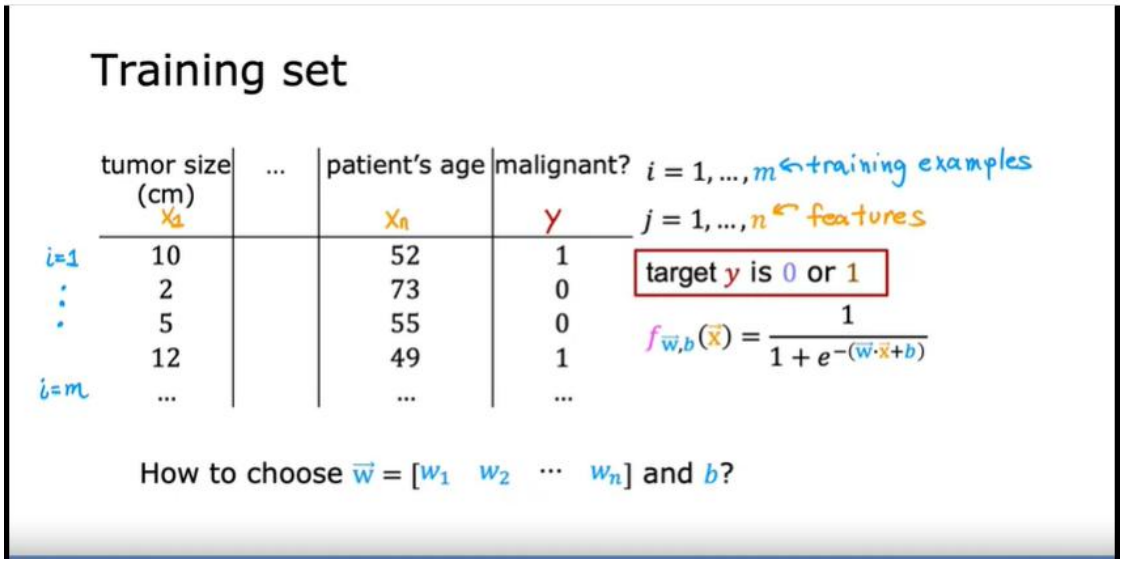






Remember that the cost function gives you a way to measure how well a specific set of parameters fits the training data. Thereby gives you a way to try to choose better parameters.

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



Squared error cost

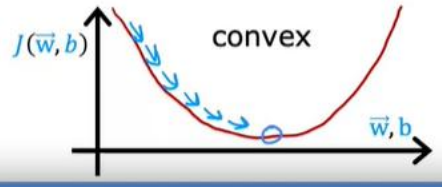
cost

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

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

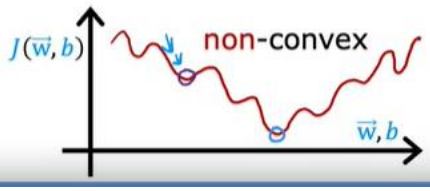
linear regression

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



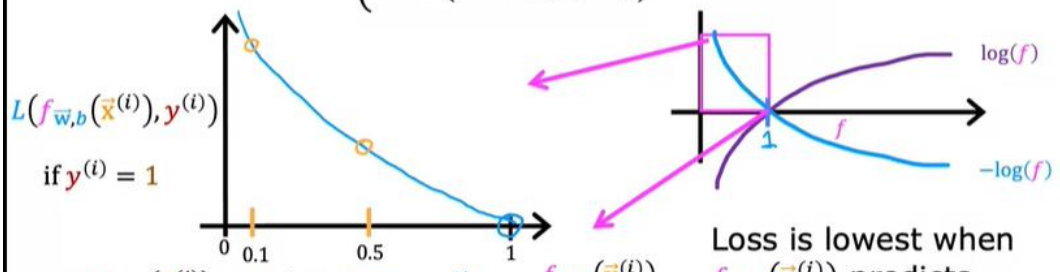
logistic regression

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



Logistic loss function

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



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

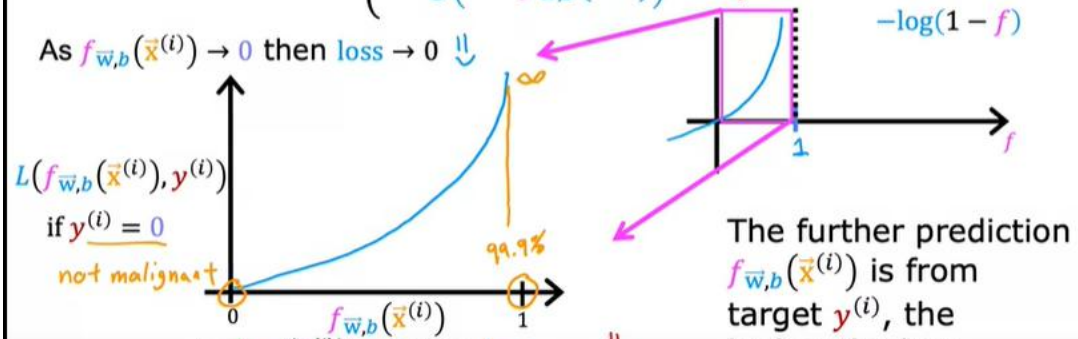
As $f_{\vec{w},b}(\vec{x}^{(i)}) \to 0$ then loss $\to \infty$

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

Logistic loss function

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

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



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

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

Cost

cost

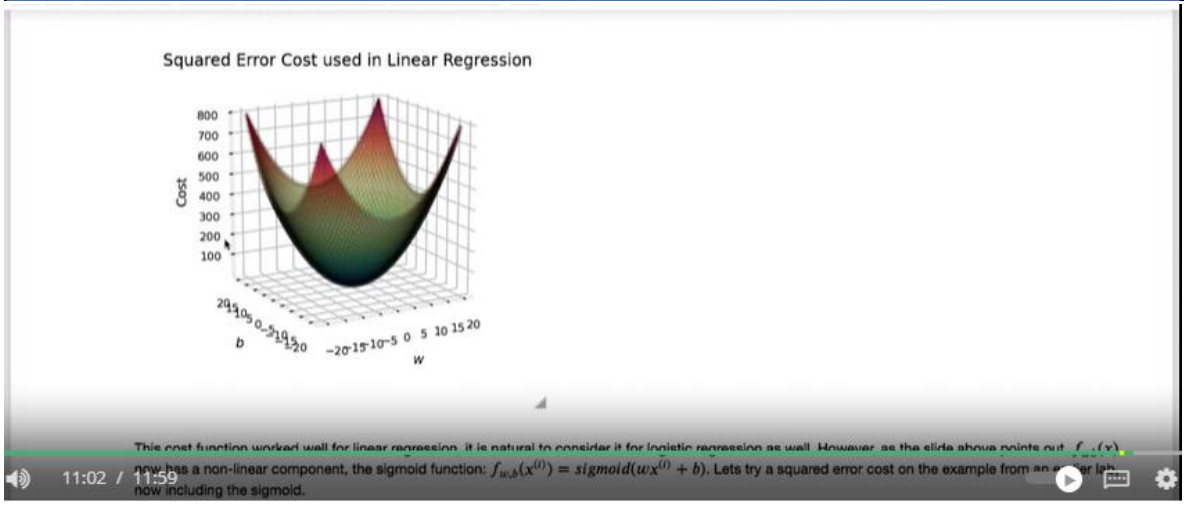
$$J(\vec{w}, b) = \frac{1}{m}\sum_{i=1}^{m} L\left(f_{\vec{w},b}(\vec{x}^{(i)}), y^{(i)}\right)$$

loss

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

convex
↳ can reach a global minimum

find w, b that minimize cost J



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

Simplified loss function

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

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

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

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

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

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

Simplified cost function

loss

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

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!)

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

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 {

$j = 1 \ldots n$

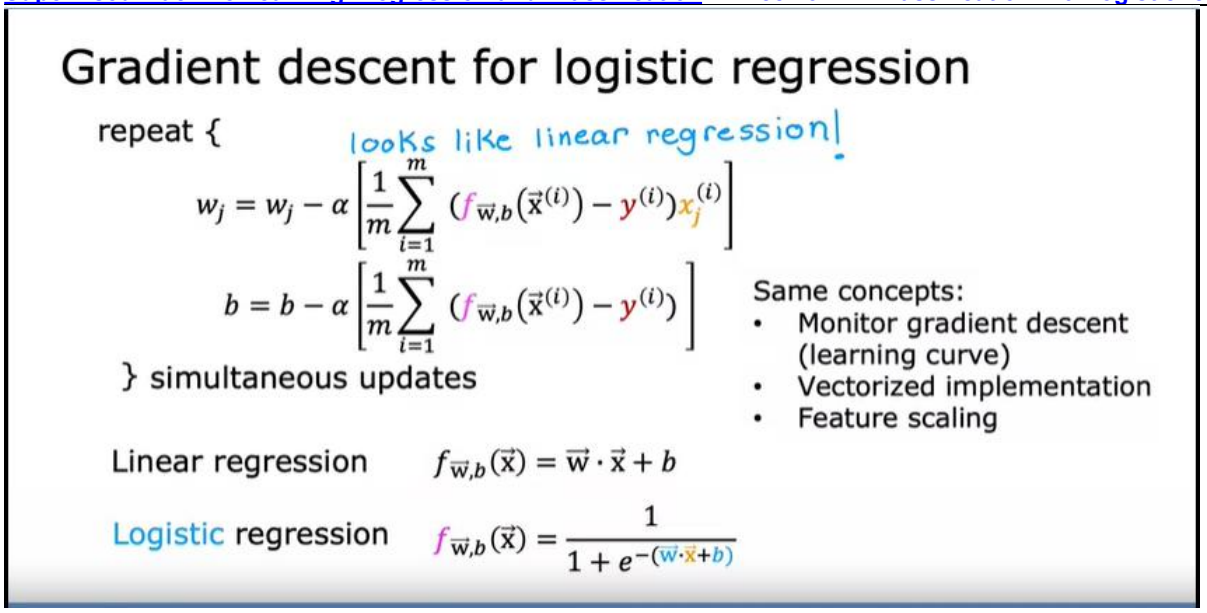
$$w_j = w_j - \alpha\frac{\partial}{\partial w_j}J(\vec{w},b)$$

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

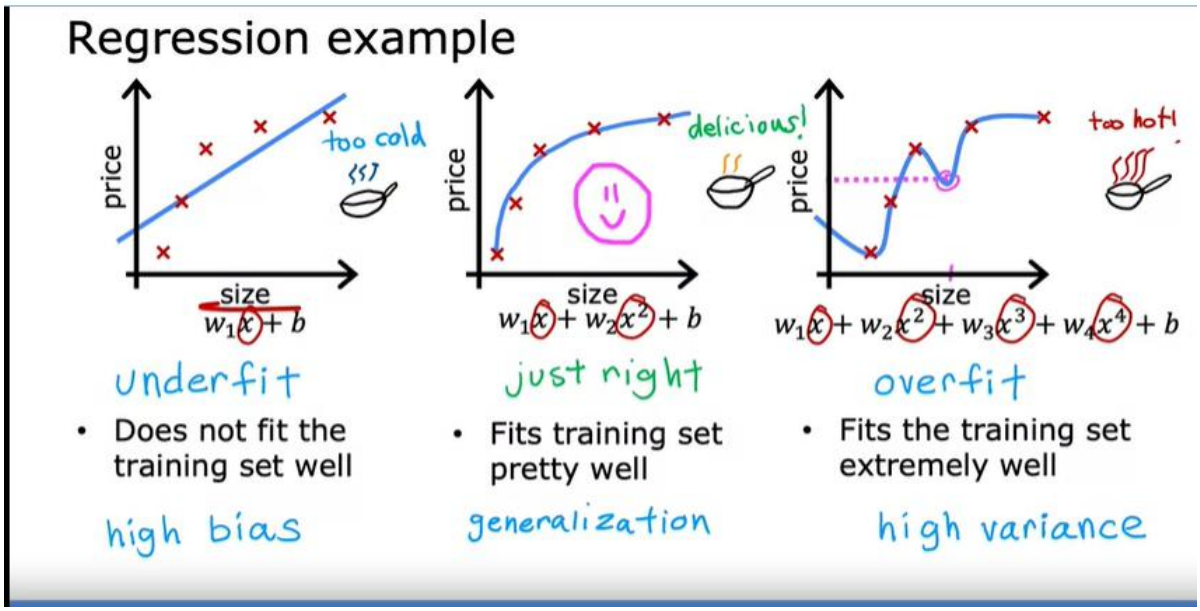
} simultaneous updates

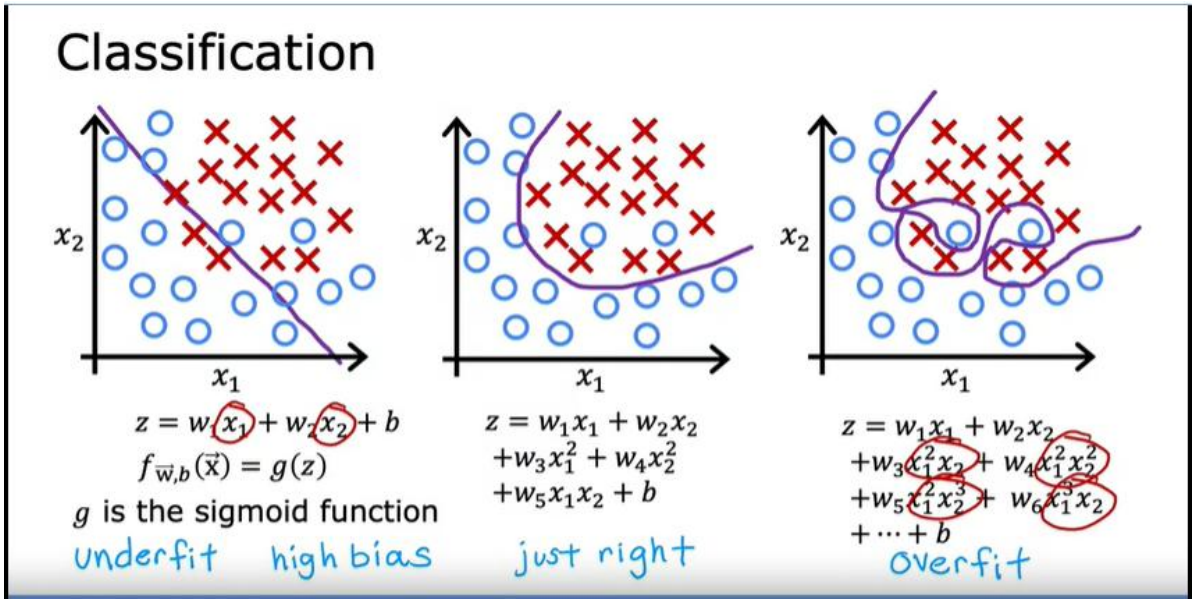
$$\frac{\partial}{\partial w_j}J(\vec{w},b) = \frac{1}{m}\sum_{i=1}^{m}\left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)}\right)x_j^{(i)}$$

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

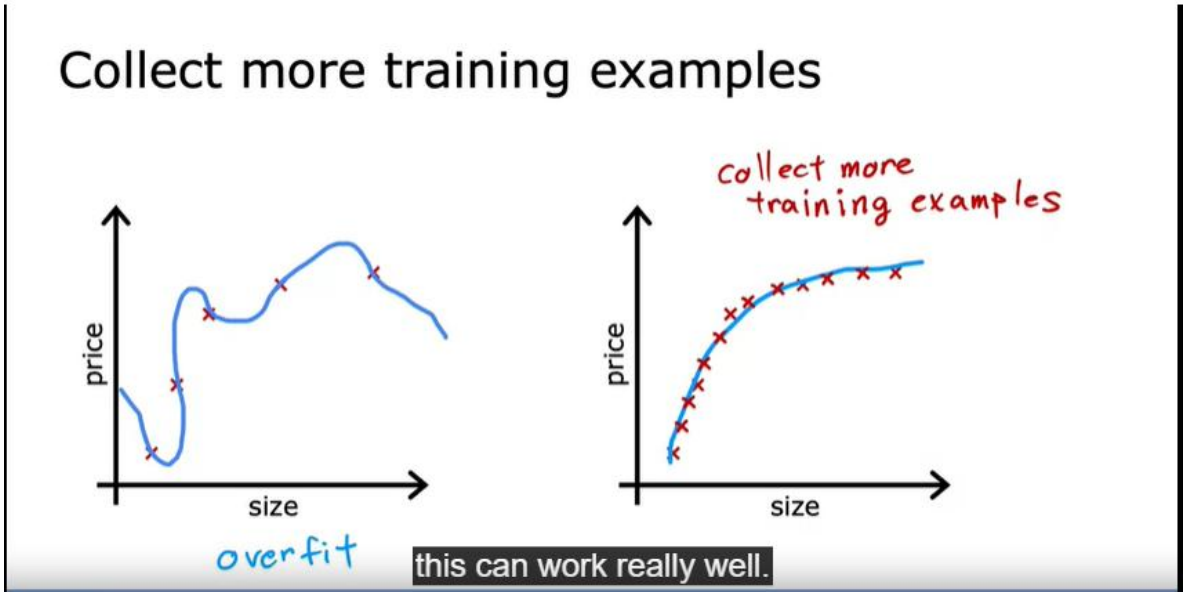


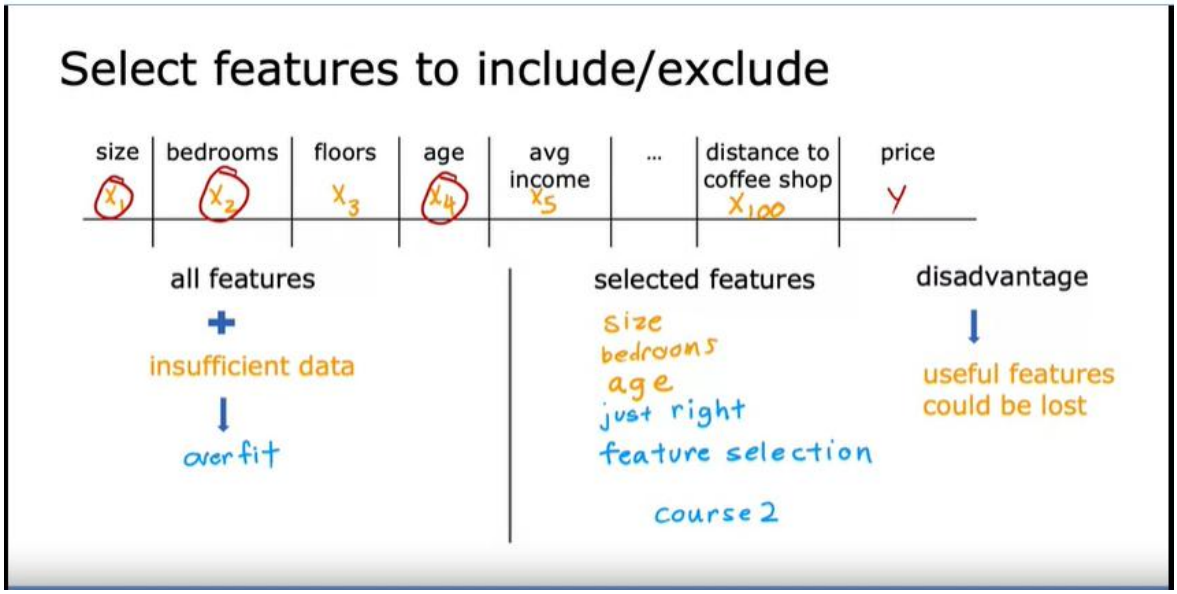
3.4 - The problem of overfitting : 3.4.1 - The problem of overfitting :

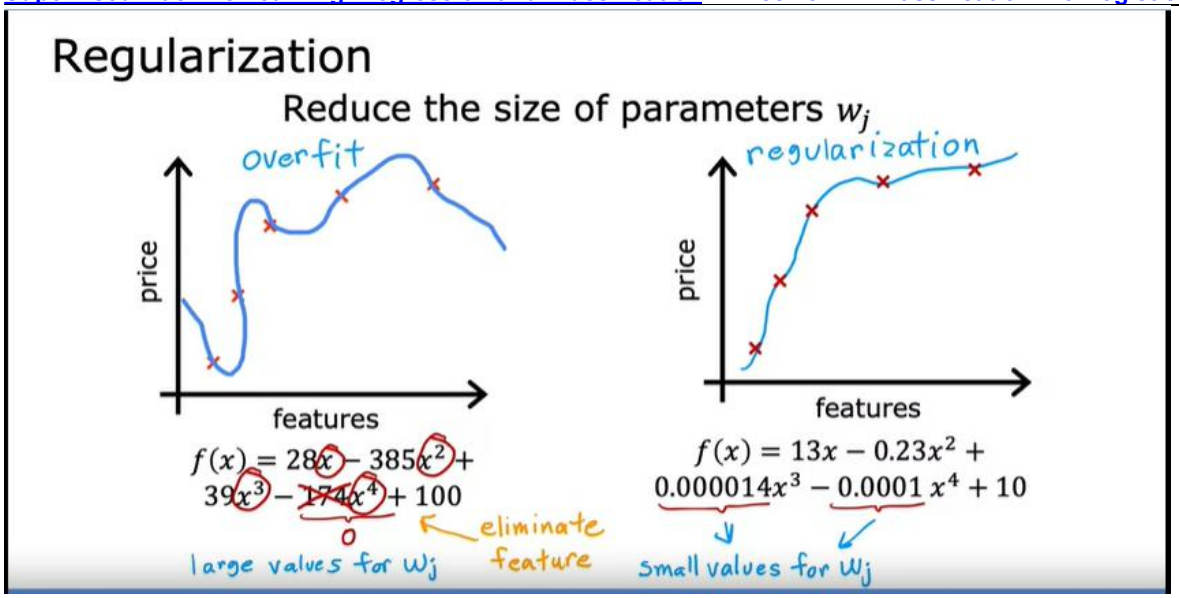


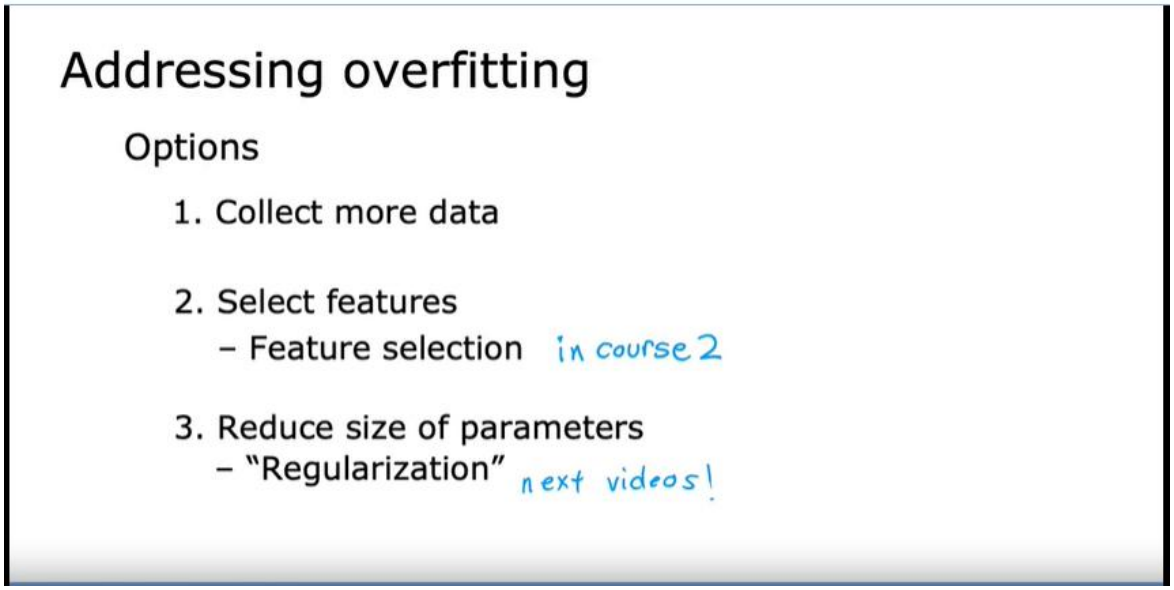


3.4 - The problem of overfitting : 3.4.2 - Addressing overfitting :

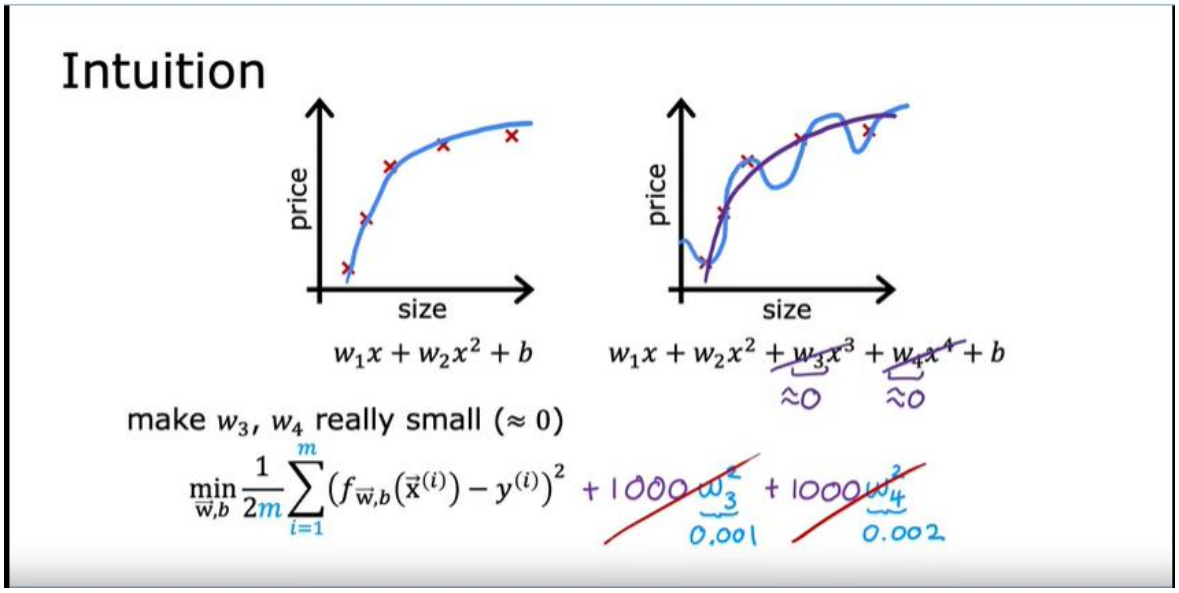


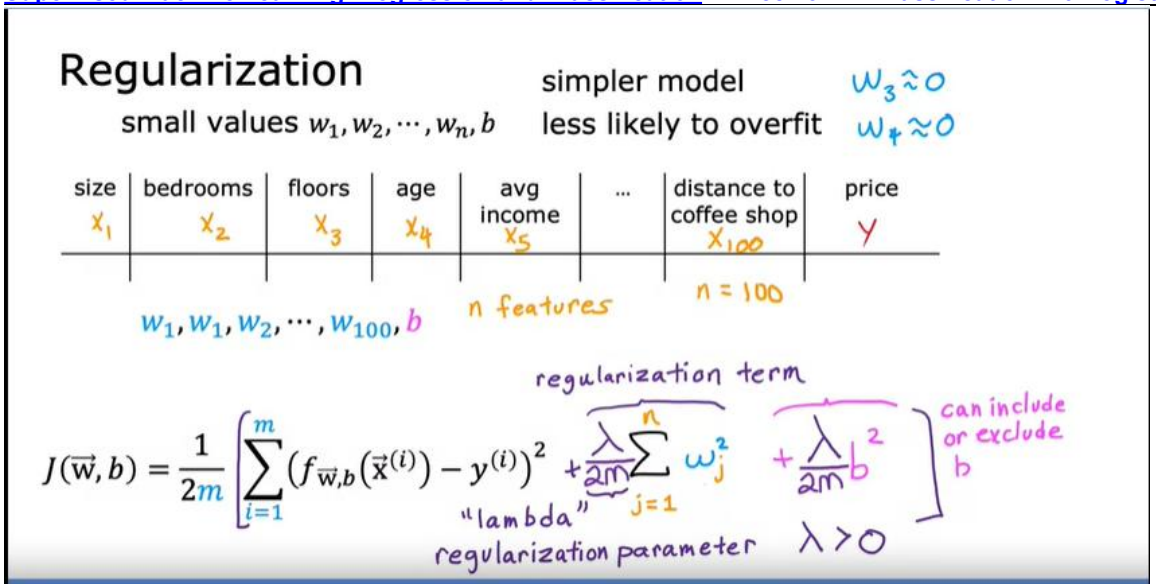


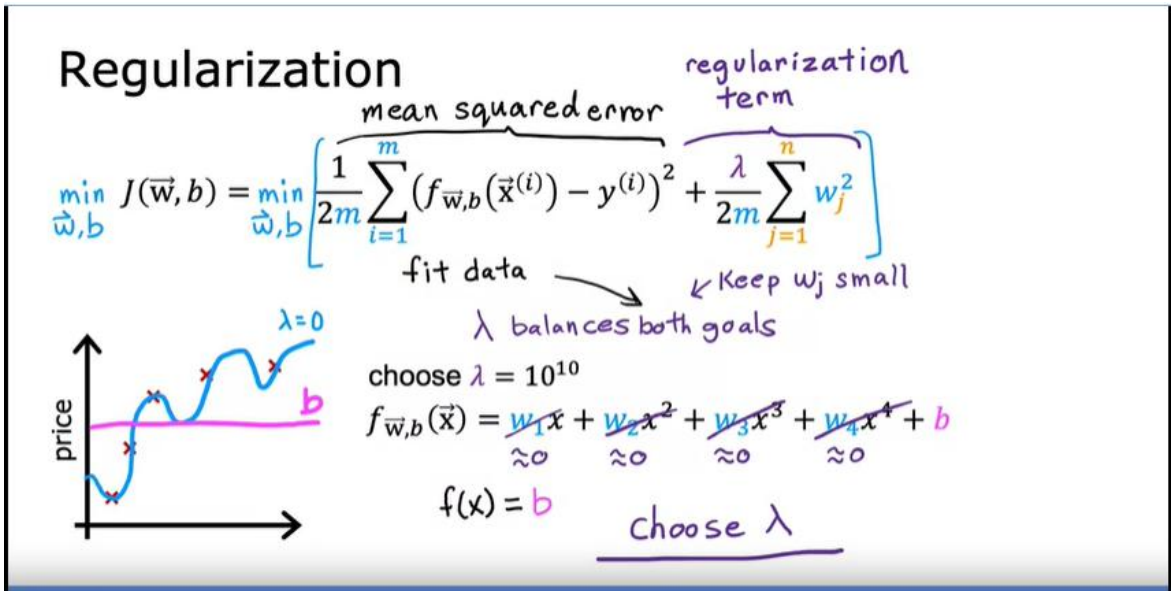




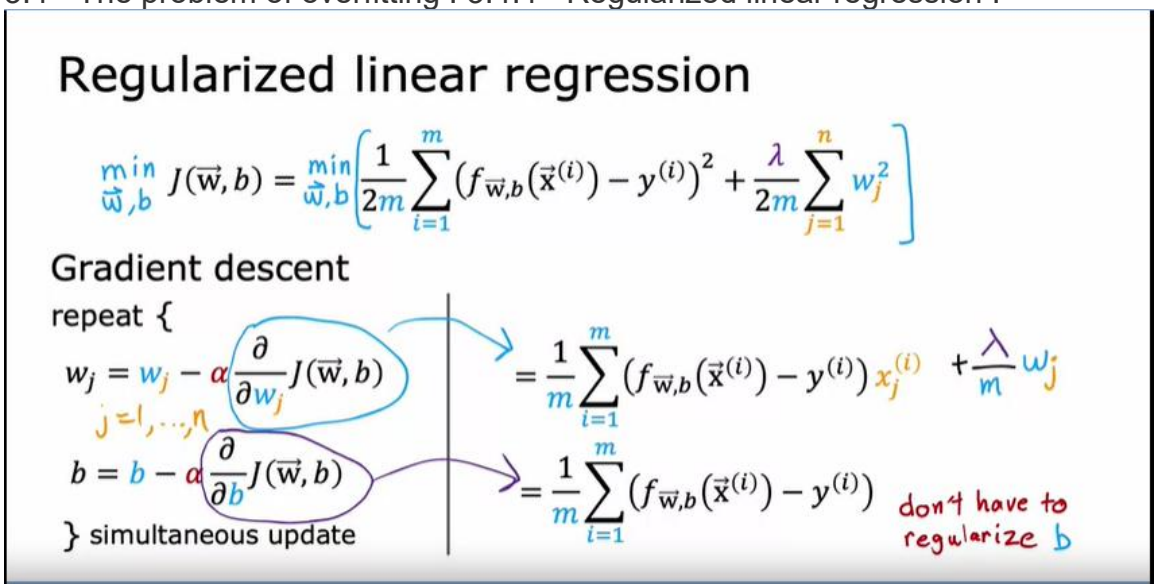
3.4 - The problem of overfitting : 3.4.3 - Cost function with regularization :







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



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 $j = 1 \ldots n$

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

$w_j \left(1 - \alpha \frac{\lambda}{m}\right)$ shrink w_j — usual update

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

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

$w_j (1 - 0.0002)$

0.9998

How we get the derivative term (optional)

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

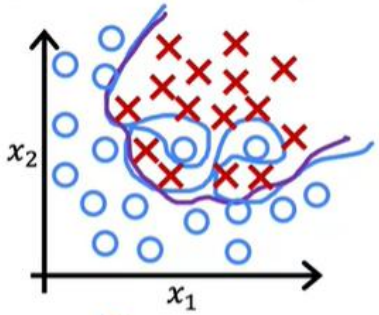
$$= \frac{1}{2m} \sum_{i=1}^{m} \left[\left(\vec{w} \cdot \vec{x}^{(i)} + b - y^{(i)} \right) 2 x_j^{(i)} \right] + \frac{\lambda}{2m} 2 w_j \quad \text{No } \sum_{j=1}^{n}$$

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

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

3.4 - The problem of overfitting : 3.4.5 - 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

Looks same as for linear regression!

repeat {

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

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

logistic regression

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

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

don't have to regularize

}