

Supervised Machine Learning

Regression

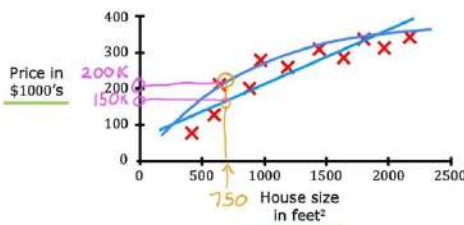
predict a number
from many possible
outputs

Classification

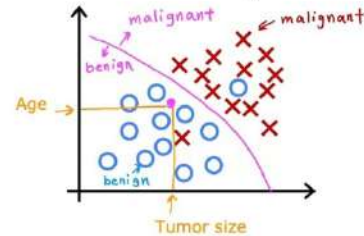
small number
of outputs

tumor → benign
→ malignant

Regression: Housing price prediction

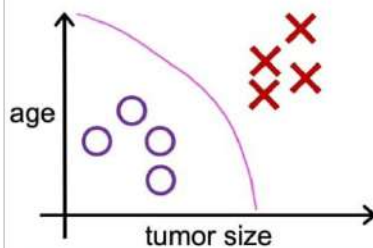


Two or more inputs

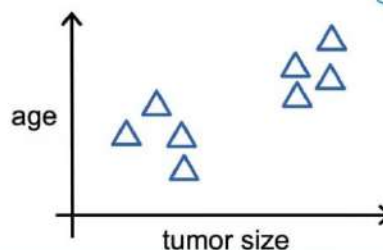


Supervised vs Unsupervised

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



Unsupervised learning
Find something interesting
in **unlabeled** data.



Unsupervised Machine Learning

Clustering

take data without labels and automatically group them to clusters

Anomaly Detection

find unusual data points

Dimensionality Reduction

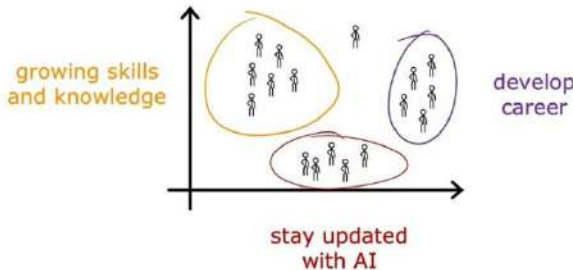
Compress data using fewer numbers

Clustering: Google news



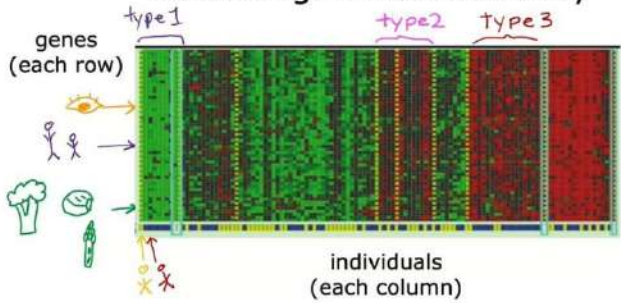
Stanford ONLINE DeepLearning.AI Andrew Ng

Clustering: Grouping customers



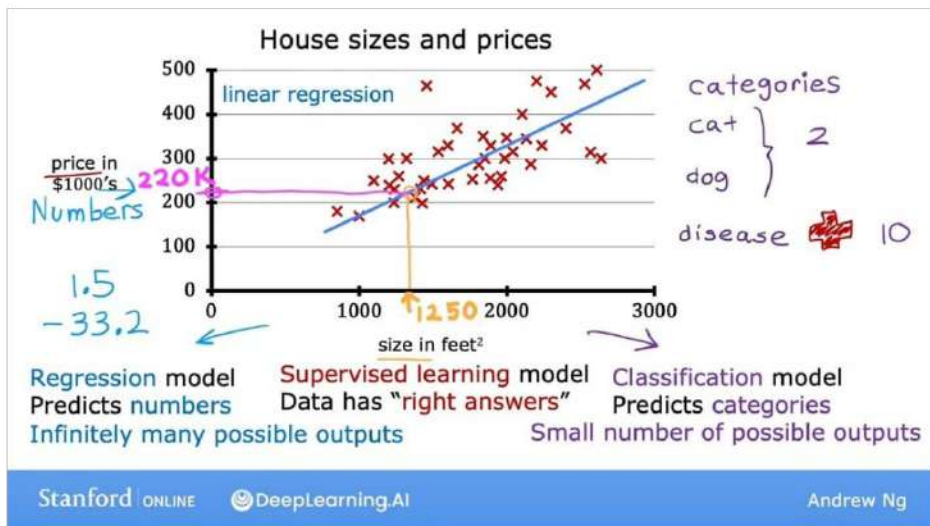
Stanford ONLINE DeepLearning.AI Andrew Ng

Clustering: DNA microarray



Stanford ONLINE DeepLearning.AI Andrew Ng

Linear Regression



Terminology

Training set: Data used to train the model

x	y
size in feet ²	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 exponent

Notation:

x = "input" variable
feature

y = "output" variable
"target" variable

m = number of training examples

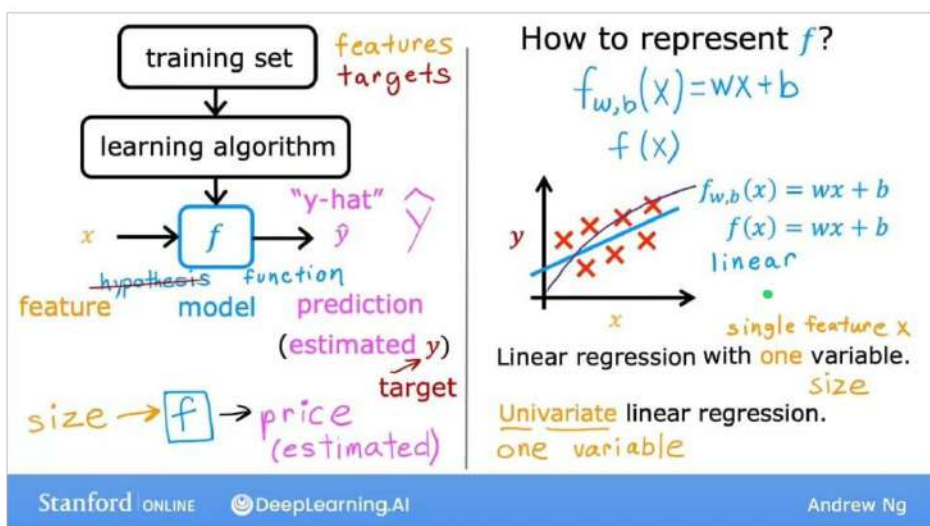
(x, y) = single training example

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

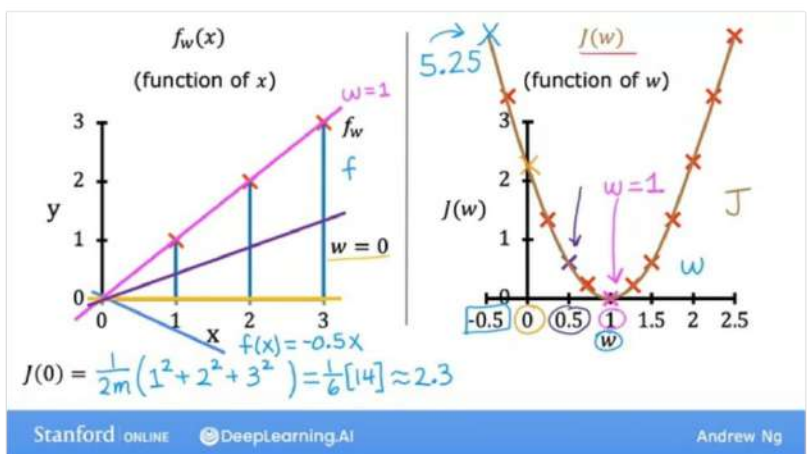
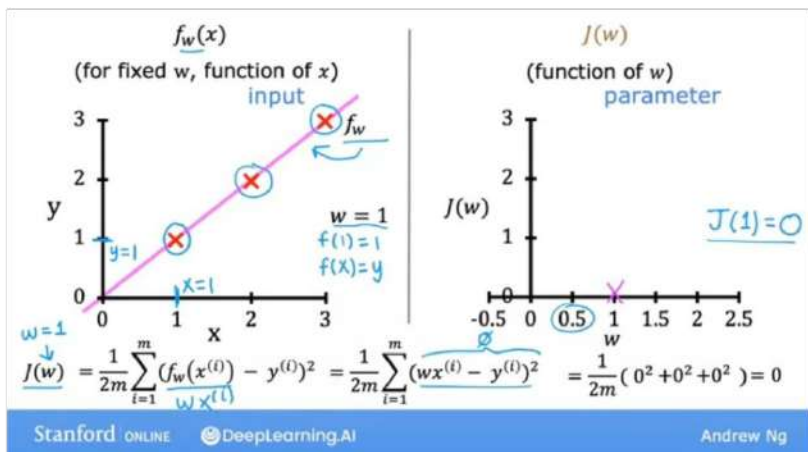
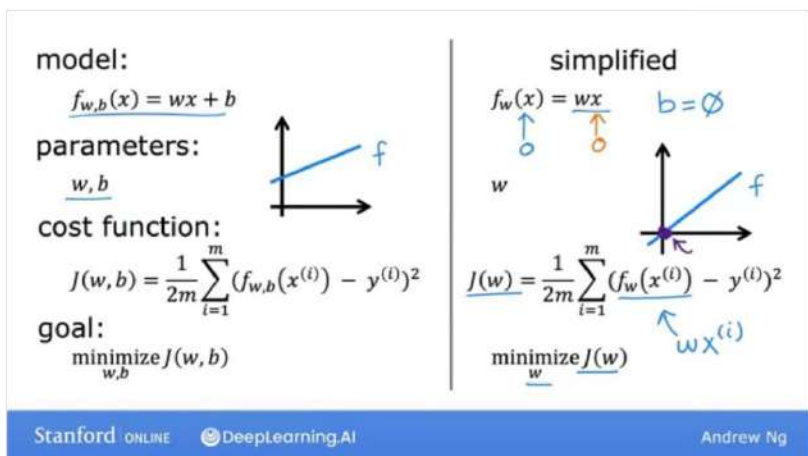
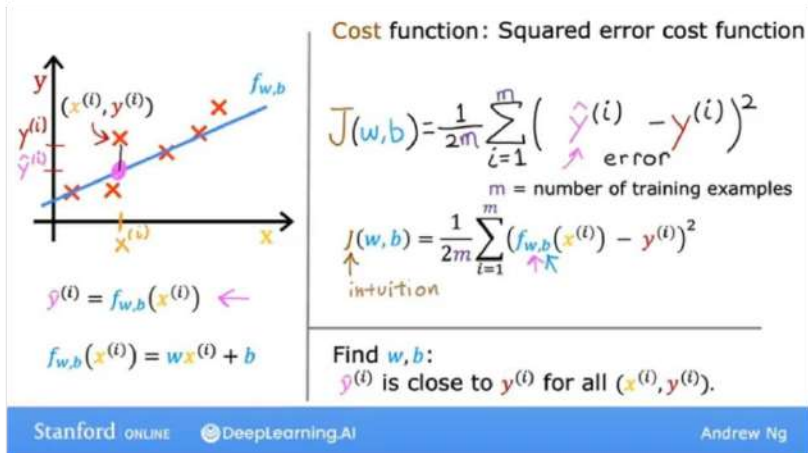
$(x^{(i)}, y^{(i)})$ = i^{th} training example

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

Stanford ONLINE DeepLearning.AI Andrew Ng



Cost function



Visualize cost function

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

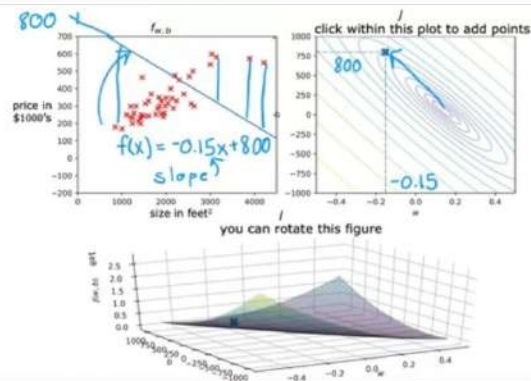
Parameters w, b

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

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

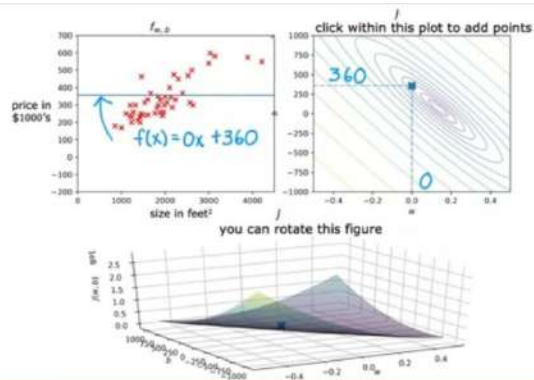
Stanford ONLINE DeepLearning.AI

Andrew Ng



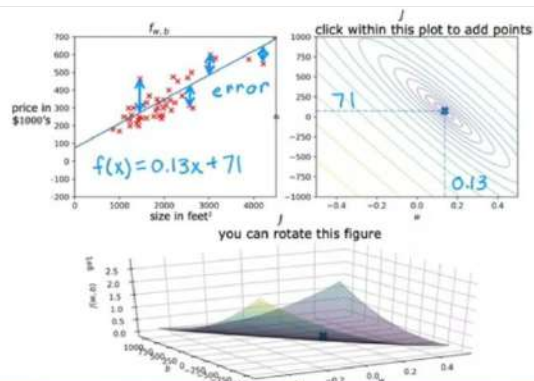
Stanford ONLINE DeepLearning.AI

Andrew Ng



Stanford ONLINE DeepLearning.AI

Andrew Ng



Stanford ONLINE DeepLearning.AI

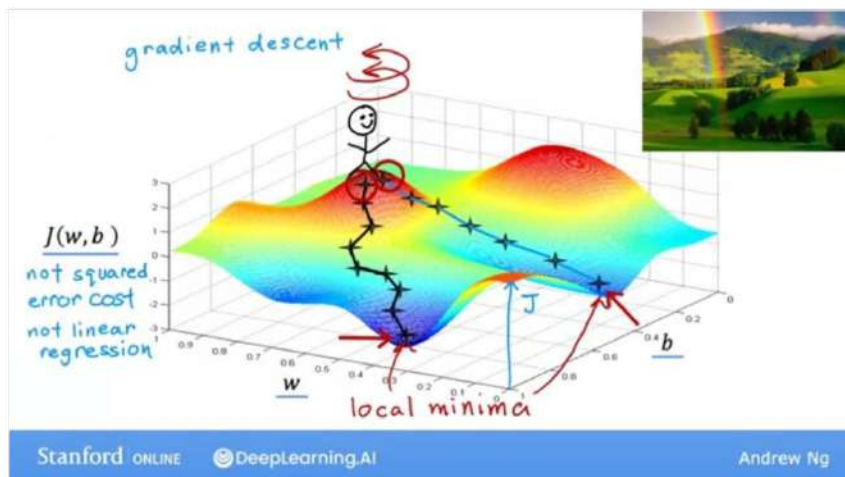
Andrew Ng

Gradient Descent

Have some function $J(w, b)$ for linear regression or any function
 Want $\min_{w, b} J(w, b)$ $\min_{w_1, \dots, w_n, b} J(w_1, w_2, \dots, w_n, b)$

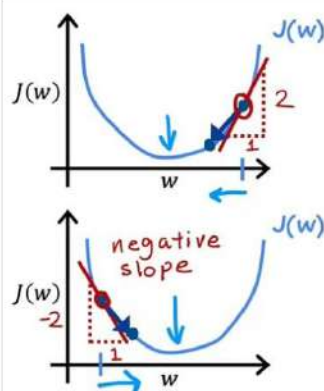
Outline:

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



Gradient descent algorithm

repeat until convergence {
learning rate $w = w - \alpha \frac{\partial}{\partial w} J(w, b)$ derivative
 $b = b - \alpha \frac{\partial}{\partial b} J(w, b)$
 $\min_w J(w)$



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

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

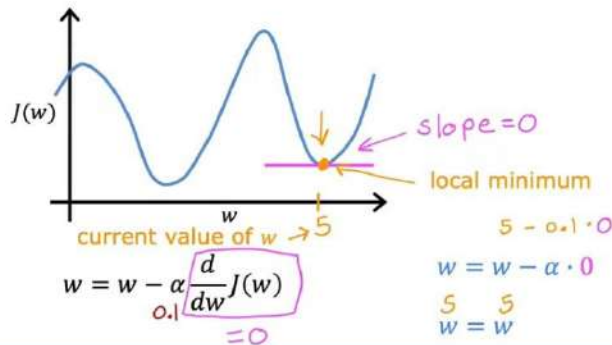
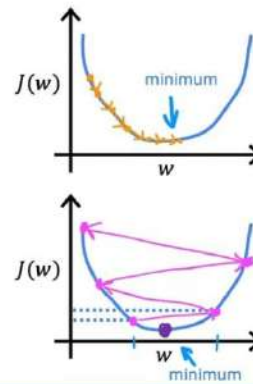
$$\frac{d}{dw} J(w) < 0$$

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

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

If α is too small...
Gradient descent may be slow.

If α is too large...
Gradient descent may:
- Overshoot, never reach minimum
- F



Gradient Descent for Linear Regression

Linear regression model

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

Cost function

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

Gradient descent algorithm

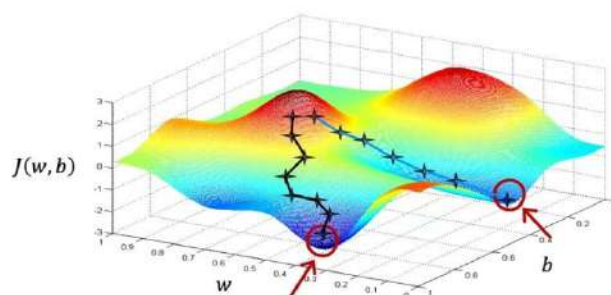
repeat until convergence {

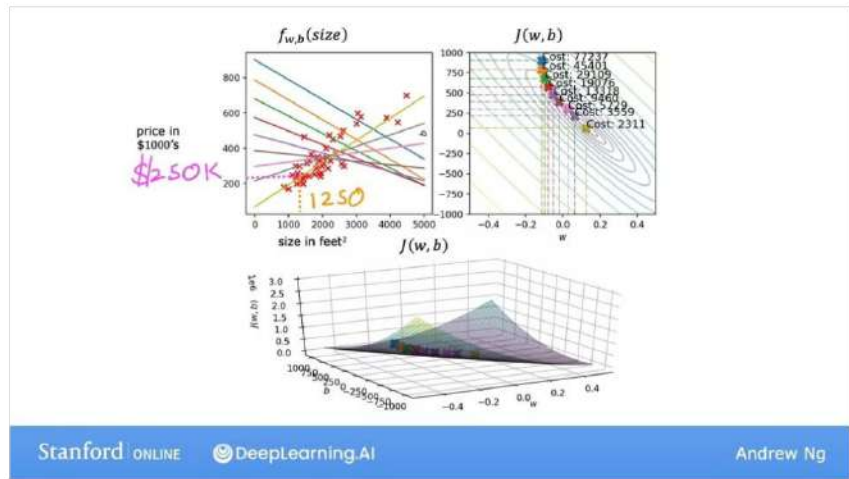
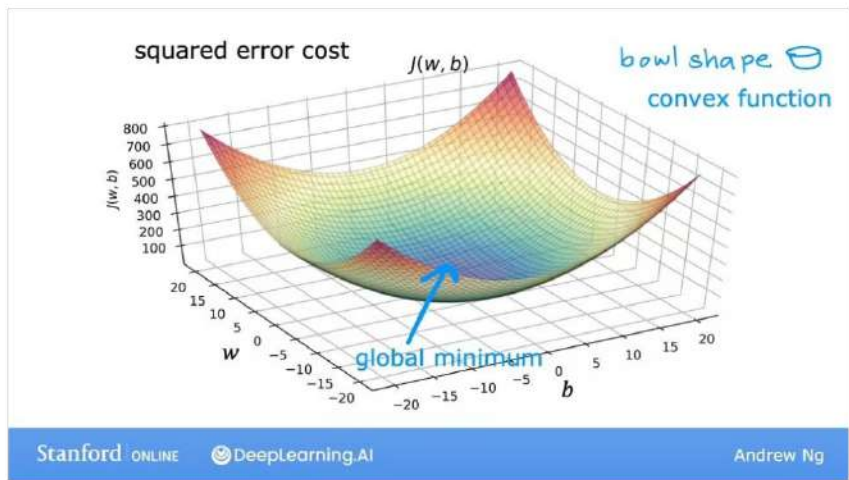
$$w = w - \alpha \frac{\partial}{\partial w} J(w,b) \rightarrow \frac{1}{m} \sum_{i=1}^m (f_{w,b}(x^{(i)}) - y^{(i)}) x^{(i)}$$

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

}

More than one local minimum





"Batch" gradient descent

DeepLearning.AI

THE BATCH

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

other gradient descent: subsets

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

$m = 47$

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

Week 2

Multiple Linear Regression

Multiple features (variables)

	Size in feet ²	Number of bedrooms	Number of floors	Age of home in years	Price (\$) in \$1000's	
	x_1	x_2	x_3	x_4		$j=1...4$ $n=4$
	2104	5	1	45	460	
$i=2$	1416	3	2	40	232	
	1534	3	2	30	315	
	852	2	1	36	178	
	

$x_j = j^{\text{th}}$ feature
 n = number of features
 $\vec{x}^{(i)}$ = features of i^{th} training example
 $x_j^{(i)}$ = value of feature j in i^{th} training example

$\vec{x}^{(2)} = [1416 \ 3 \ 2 \ 40]$
 $x_3^{(2)} = 2$

Stanford ONLINE DeepLearning.AI

Andrew Ng

Model:

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

example

$$f_{w,b}(x) = w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + b$$

$$f_{w,b}(x) = 0.1x_1 + 4x_2 + 10x_3 - 2x_4 + 80$$

size #bedrooms #floors years base price

$$f_{w,b}(x) = w_1x_1 + w_2x_2 + \dots + w_nx_n + b$$

Stanford ONLINE DeepLearning.AI

Andrew Ng

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

$\vec{w} = [w_1 \ w_2 \ w_3 \ \dots \ w_n]$ parameters of the model
 b is a number

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

$$f_{\vec{w},b}(\vec{x}) = \vec{w} \cdot \vec{x} + b = w_1x_1 + w_2x_2 + w_3x_3 + \dots + w_nx_n + b$$

dot product multiple linear regression
 (not multivariate regression)

Stanford ONLINE DeepLearning.AI

Andrew Ng

Multiple Features

Multiple features (variables)

	Size in feet ²	Number of bedrooms	Number of floors	Age of home in years	Price (\$) in \$1000's	
	x_1	x_2	x_3	x_4		$j=1...4$ $n=4$
$i=2$	2104	5	1	45	460	
	1416	3	(2)	40	232	$\vec{x}^{(2)} = [1416 \ 3 \ (2) \ 40]$
	1534	3	2	30	315	
	852	2	1	36	178	$x_3^{(2)} = 2$
	

$x_j = j^{th}$ feature
 n = number of features
 $\vec{x}^{(i)}$ = features of i^{th} training example
 $x_j^{(i)}$ = value of feature j in i^{th} training example

Model:

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

example

$$f_{w,b}(x) = w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + b$$

$$f_{w,b}(x) = 0.1 \underset{\substack{\uparrow \\ \text{size}}}{x_1} + 4 \underset{\substack{\uparrow \\ \text{\# bedrooms}}}{x_2} + 10 \underset{\substack{\uparrow \\ \text{\# floors}}}{x_3} - 2 \underset{\substack{\uparrow \\ \text{years}}}{x_4} + 80 \underset{\substack{\uparrow \\ \text{base price}}}{b}$$

$$f_{w,b}(x) = w_1x_1 + w_2x_2 + \dots + w_nx_n + b$$

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

$\vec{w} = [w_1 \ w_2 \ w_3 \ \dots \ w_n]$ parameters of the model
 b is a number

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

$$f_{\vec{w},b}(\vec{x}) = \vec{w} \cdot \vec{x} + b = w_1x_1 + w_2x_2 + w_3x_3 + \dots + w_nx_n + b$$

\uparrow
 dot product multiple linear regression
 (not multivariate regression)

Vectorization

Parameters and features

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

b is a number

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

linear algebra: count from 1

NumPy

$w[0] \ w[1] \ w[2]$
 $w = \text{np.array}([1.0, 2.5, -3.3])$

$b = 4$ $x[0] \ x[1] \ x[2]$

$x = \text{np.array}([10, 20, 30])$

code: count from 0

Without vectorization $n=100,000$

$f_{\vec{w},b}(\vec{x}) = w_1x_1 + w_2x_2 + w_3x_3 + b$

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



Without vectorization

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

$\text{range}(0, n) \rightarrow j=0 \dots n-1$

$f = 0$ $\text{range}(n)$
 for j in $\text{range}(0, n)$:
 $f = f + w[j] * x[j]$
 $f = f + b$



Vectorization

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

$f = \text{np.dot}(w, x) + b$



Stanford ONLINE DeepLearning.AI

Andrew Ng

Without vectorization

for j in $\text{range}(0, 16)$:
 $f = f + w[j] * x[j]$

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

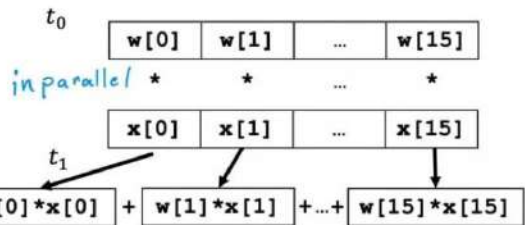
t_1
 $f + w[1] * x[1]$

...

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

Vectorization

$\text{np.dot}(w, x)$



efficient \rightarrow scale to large datasets

Stanford ONLINE DeepLearning.AI

Andrew Ng

Gradient descent $\vec{w} = (w_1 \ w_2 \ \dots \ w_{16})$ ~~b~~ parameters

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

$w = \text{np.array}([0.5, 1.3, \dots, 3.4])$

$d = \text{np.array}([0.3, 0.2, \dots, 0.4])$

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

Without vectorization

$w_1 = w_1 - 0.1d_1$

$w_2 = w_2 - 0.1d_2$

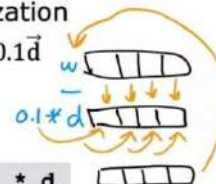
\vdots

$w_{16} = w_{16} - 0.1d_{16}$

for j in $\text{range}(0, 16)$:
 $w[j] = w[j] - 0.1 * d[j]$

With vectorization

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



$w = w - 0.1 * d$

Stanford ONLINE DeepLearning.AI

Andrew Ng

Gradient Descent for multiple linear regression

	Previous notation	Vector notation
Parameters	w_1, \dots, w_n b	$\vec{w} = [w_1 \dots w_n]$ ← vector of length n b still a number
Model	$f_{\vec{w},b}(\vec{x}) = w_1 x_1 + \dots + w_n x_n + b$	$f_{\vec{w},b}(\vec{x}) = \vec{w} \cdot \vec{x} + b$
Cost function	$J(w_1, \dots, w_n, b)$	$J(\vec{w}, b)$ ← dot product
Gradient descent	repeat { $w_j = w_j - \alpha \frac{\partial}{\partial w_j} J(w_1, \dots, w_n, b)$ $b = b - \alpha \frac{\partial}{\partial b} J(w_1, \dots, w_n, b)$ }	repeat { $w_j = w_j - \alpha \frac{\partial}{\partial w_j} J(\vec{w}, b)$ $b = b - \alpha \frac{\partial}{\partial b} J(\vec{w}, b)$ }

Stanford ONLINE

DeepLearning.AI

Andrew Ng

Gradient descent	
<p>One feature</p> <p>repeat {</p> $\vec{w} = \vec{w} - \alpha \frac{\partial}{\partial \vec{w}} J(\vec{w}, b)$ $b = b - \alpha \frac{\partial}{\partial b} J(\vec{w}, b)$ <p>simultaneously update w, b</p> <p>}</p>	<p>n features ($n \geq 2$)</p> <p>repeat {</p> $w_1 = w_1 - \alpha \frac{\partial}{\partial w_1} J(\vec{w}, b)$ \vdots $w_n = w_n - \alpha \frac{\partial}{\partial w_n} J(\vec{w}, b)$ $b = b - \alpha \frac{\partial}{\partial b} J(\vec{w}, b)$ <p>simultaneously update w_j (for $j = 1, \dots, n$) and b</p> <p>}</p>

Stanford ONLINE

DeepLearning.AI

Andrew Ng

An alternative to gradient descent

→ Normal equation

- Only for linear regression
 - Solve for w, b without iterations
- Disadvantages
- Doesn't generalize to other learning algorithms.
 - Slow when number of features is large ($> 10,000$)

What you need to know

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

Stanford ONLINE

DeepLearning.AI

Andrew Ng

Feature Scaling

Feature and parameter values

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

\downarrow
size
 \downarrow
bedrooms
 \downarrow
large
 \downarrow
small

x_1 : size (feet²)
range: 300 – 2,000

x_2 : # bedrooms
range: 0 – 5

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

size of the parameters w_1, w_2 ?

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

$$\widehat{\text{price}} = \frac{50 * 2000}{100,000\text{K}} + \frac{0.1 * 5}{0.5\text{K}} + \frac{50}{50\text{K}}$$



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

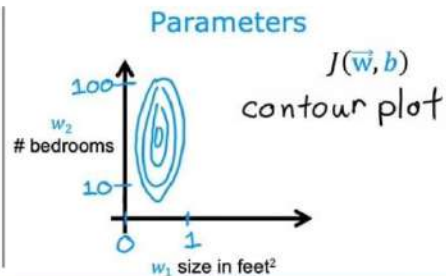
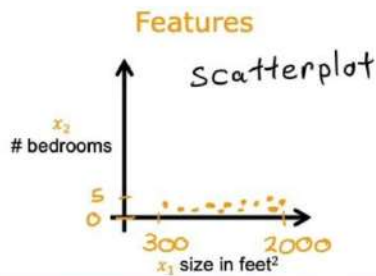
$w_1 = 0.1$, $w_2 = 50$, $b = 50$
small large

$$\widehat{\text{price}} = \frac{0.1 * 2000\text{k}}{200\text{K}} + \frac{50 * 5}{150\text{K}} + \frac{50}{50\text{K}}$$

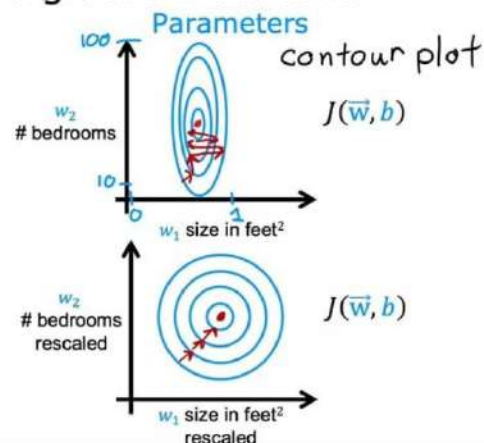
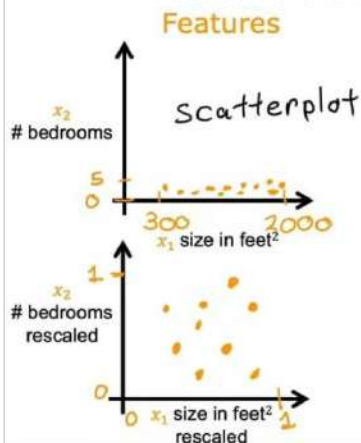
$$\widehat{\text{price}} = \$500\text{k} \text{ more reasonable}$$

Feature size and parameter size

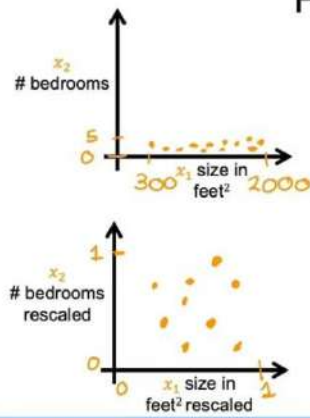
	size of feature x_j	size of parameter w_j
size in feet ²		
# bedrooms		



Feature size and gradient descent



Feature scaling

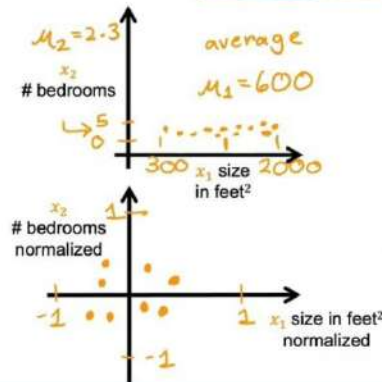


$$300 \leq x_1 \leq 2000 \quad 0 \leq x_2 \leq 5$$

$$x_{1,scaled} = \frac{x_1}{2000} \quad x_{2,scaled} = \frac{x_2}{5}$$

$$0.15 \leq x_{1,scaled} \leq 1 \quad 0 \leq x_{2,scaled} \leq 1$$

Mean normalization

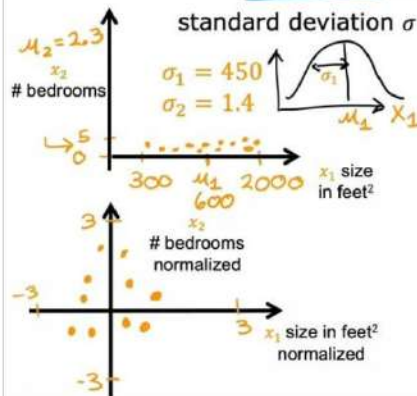


$$300 \leq x_1 \leq 2000 \quad 0 \leq x_2 \leq 5$$

$$x_1 = \frac{x_1 - \mu_1}{2000 - 300} \quad x_2 = \frac{x_2 - \mu_2}{5 - 0}$$

$$-0.18 \leq x_1 \leq 0.82 \quad -0.46 \leq x_2 \leq 0.54$$

Z-score normalization



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

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

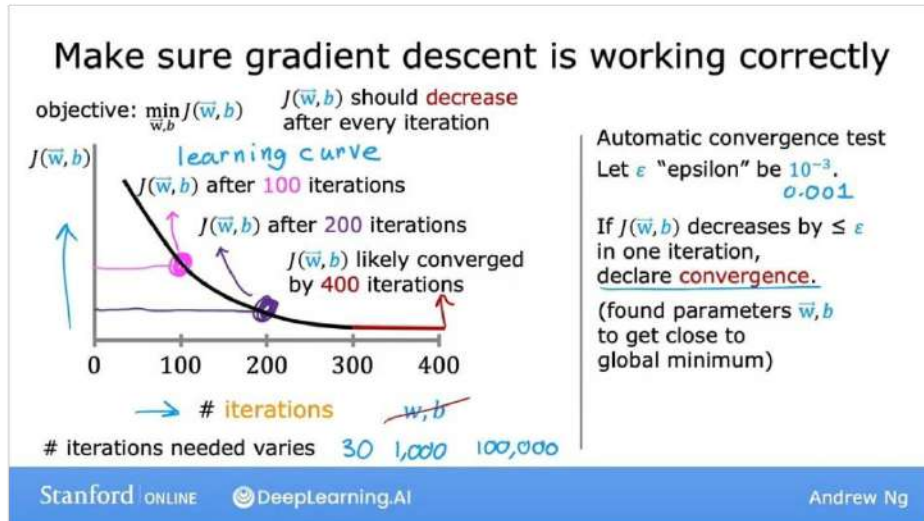
$$-0.67 \leq x_1 \leq 3.1 \quad -1.6 \leq x_2 \leq 1.9$$

Feature scaling

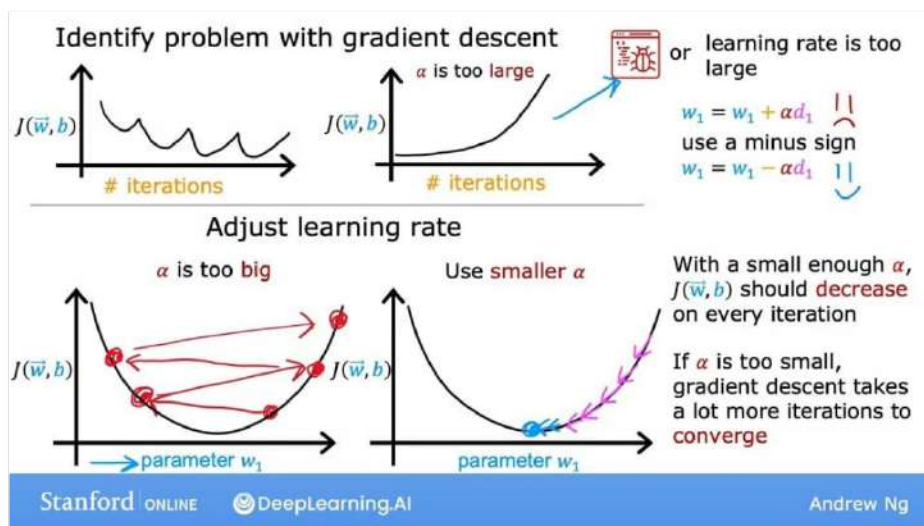
aim for about $-1 \leq x_j \leq 1$ for each feature x_j
 $-3 \leq x_j \leq 3$
 $-0.3 \leq x_j \leq 0.3$ } acceptable ranges

- $0 \leq x_1 \leq 3$ okay, no rescaling
- $-2 \leq x_2 \leq 0.5$ okay, no rescaling
- $-100 \leq x_3 \leq 100$ too large → rescale
- $-0.001 \leq x_4 \leq 0.001$ too small → rescale
- $98.6 \leq x_5 \leq 105$ too large → rescale

convergence of gradient descent



choosing learning rate



Feature Engineering

Feature engineering

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

frontage depth

$$\text{area} = \text{frontage} \times \text{depth}$$

$$x_3 = x_1 x_2$$

new feature

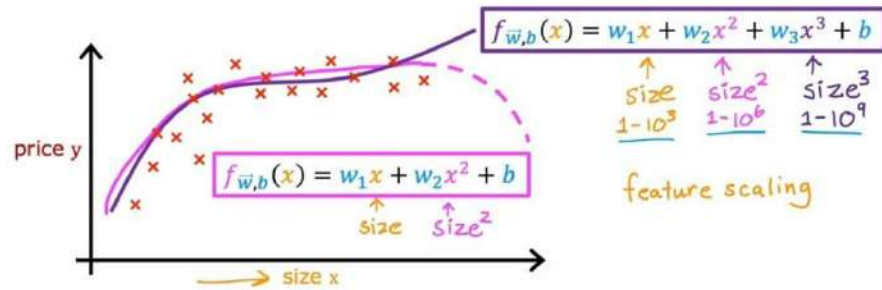
$$f_{\bar{w}, b}(\vec{x}) = w_1 x_1 + w_2 x_2 + w_3 x_3 + b$$



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

Polynomial Regression

Polynomial regression



Choice of features

