Machine Learning definition

- Arthur Samuel (1959). Machine Learning: Field of study that gives computers the ability to learn without being explicitly programmed.
- Tom Mitchell (1998) Well-posed Learning Problem: A computer program is said to *learn* from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E.

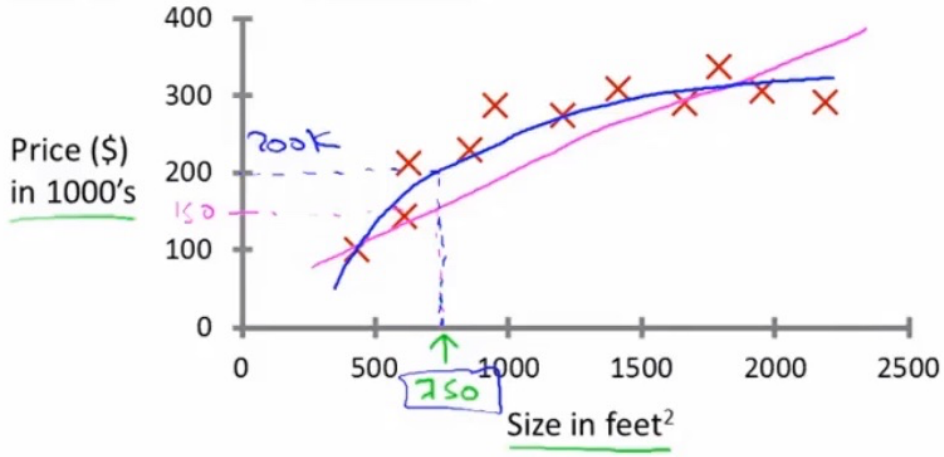
Task:

"A computer program is said to *learn* from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E."

Suppose your email program watches which emails you do or do not mark as spam, and based on that learns how to better filter spam. What is the task T in this setting?

- ○ Classifying emails as spam or not spam. T
- ○ Watching you label emails as spam or not spam. E
- ○ The number (or fraction) of emails correctly classified as spam/not spam. P
- ○ None of the above—this is not a machine learning problem.

Housing price prediction.



Supervised Learning

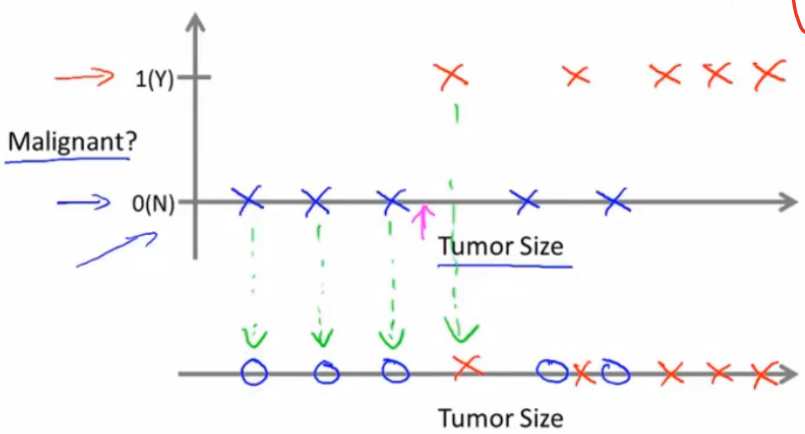
"right answers" given

Data Set

Regression: Predict continuous valued output (price)

Andrew Ng

Breast cancer (malignant, benign)



Classification

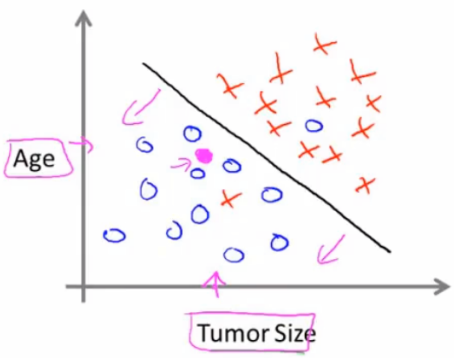
Discrete valued output (0 or 1)

0, 1, 2, 3

benign

type 1 Cancer

Andrew Ng



- Clump Thickness
- Uniformity of Cell Size
- Uniformity of Cell Shape

...

More Features

Regression model
Predicts numbers

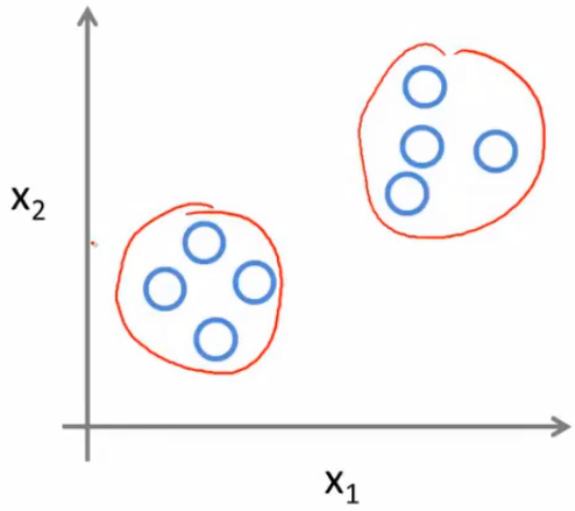
Supervised learning model
Data has "right answers"

Classification model
Predicts categories

Infinitely many possible outputs

Small number of possible outputs

Unsupervised Learning



No Lable

Andrew Ng

这就是聚类算法
So this is called a clustering algorithm.

Unsupervised learning

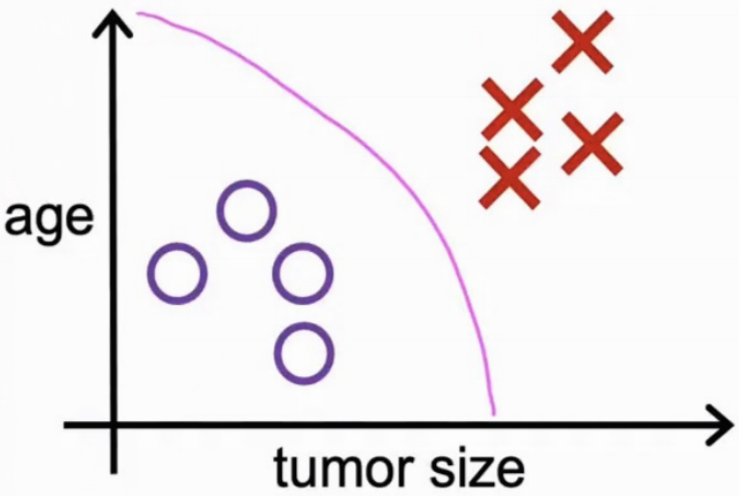
Data only comes with inputs x, but not output labels y.
Algorithm has to find structure in the data.

Clustering
Group similar data points together.

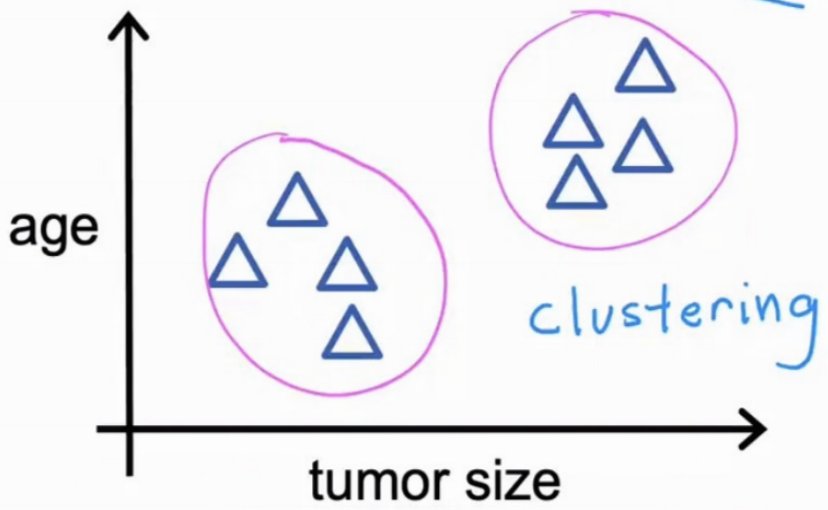
Dimensionality reduction
Compress data using fewer numbers.

Anomaly detection
Find unusual data points.

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



Unsupervised learning
Find something interesting in unlabeled data.



Training set of housing prices (Portland, OR)

Size in feet^2 (x)	Price ($) in 1000's (y)
2104	460
1416	232
1534	315
852	178
...	...

m = 47

Notation:
- **m** = Number of training examples
- **x**'s = "input" variable / features
- **y**'s = "output" variable / "target" variable

(x, y) - one training example

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

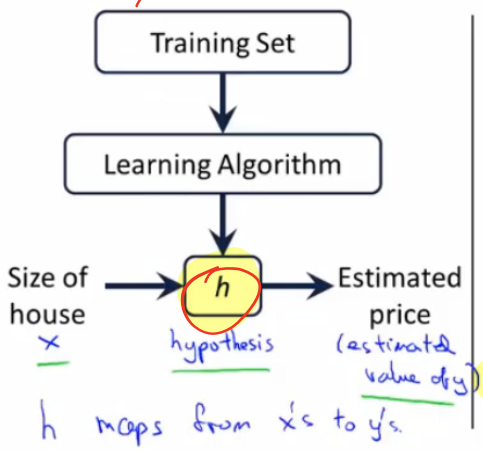
$x^{(1)} = 2104$

$x^{(2)} = 1416$

$y^{(1)} = 460$

Andrew Ng

Supervised learning Model

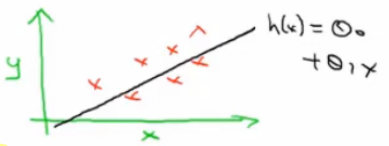


h maps from x's to y's.

How do we represent *h* ?

$h_\theta(x) = \theta_0 + \theta_1 x$

Shorthand: $h(x)$



basic

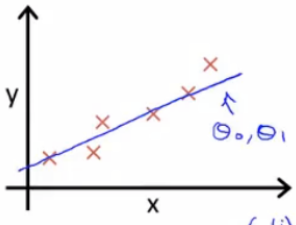
Linear regression with one variable. (x)

Univariate linear regression.

└ one variable

Andrew Ng

Cost Function



#training examples

$$\underset{\theta_0,\theta_1}{\text{minimize}} \; \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

$$h_\theta(x^{(i)}) = \theta_0 + \theta_1 x^{(i)}$$

$$J(\theta_0,\theta_1) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

Idea: Choose θ_0, θ_1 so that $h_\theta(x)$ is close to y for our training examples (x, y)

$\underset{\theta_0,\theta_1}{\text{minimize}} \; J(\theta_0,\theta_1)$

Cost function

Squared error function

x, y

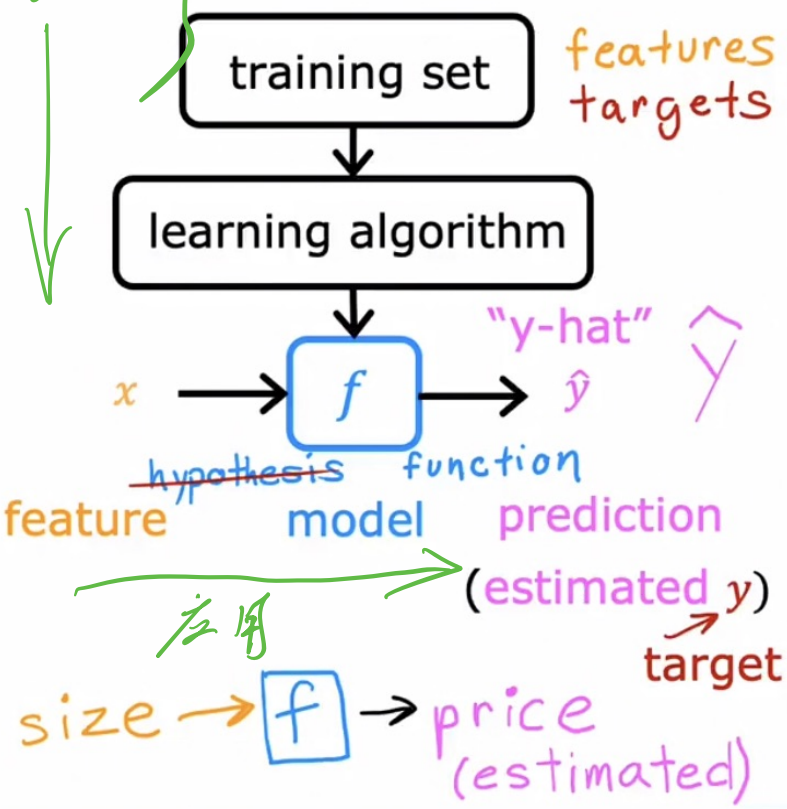
Andrew Ng

error cost function is probably the most commonly used one for regression problems.

但是平方误差代价函数可能是解决回归问题最常用的手段了

Regression

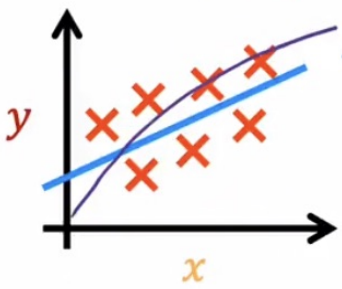
training



How to represent f?

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

$f(x)$



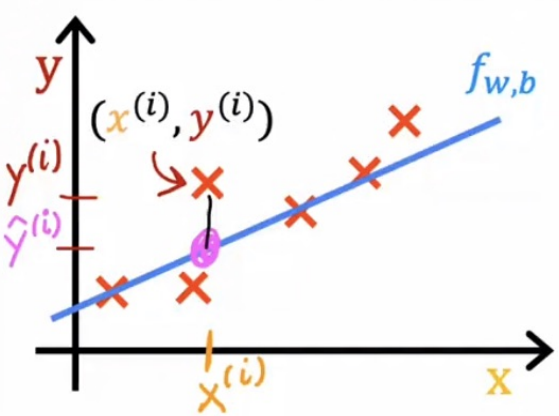
single feature x

Linear regression with one variable.

size

Univariate linear regression.

one variable



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

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

Cost function: Squared error cost function

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

$\hat{y}^{(i)}$: error

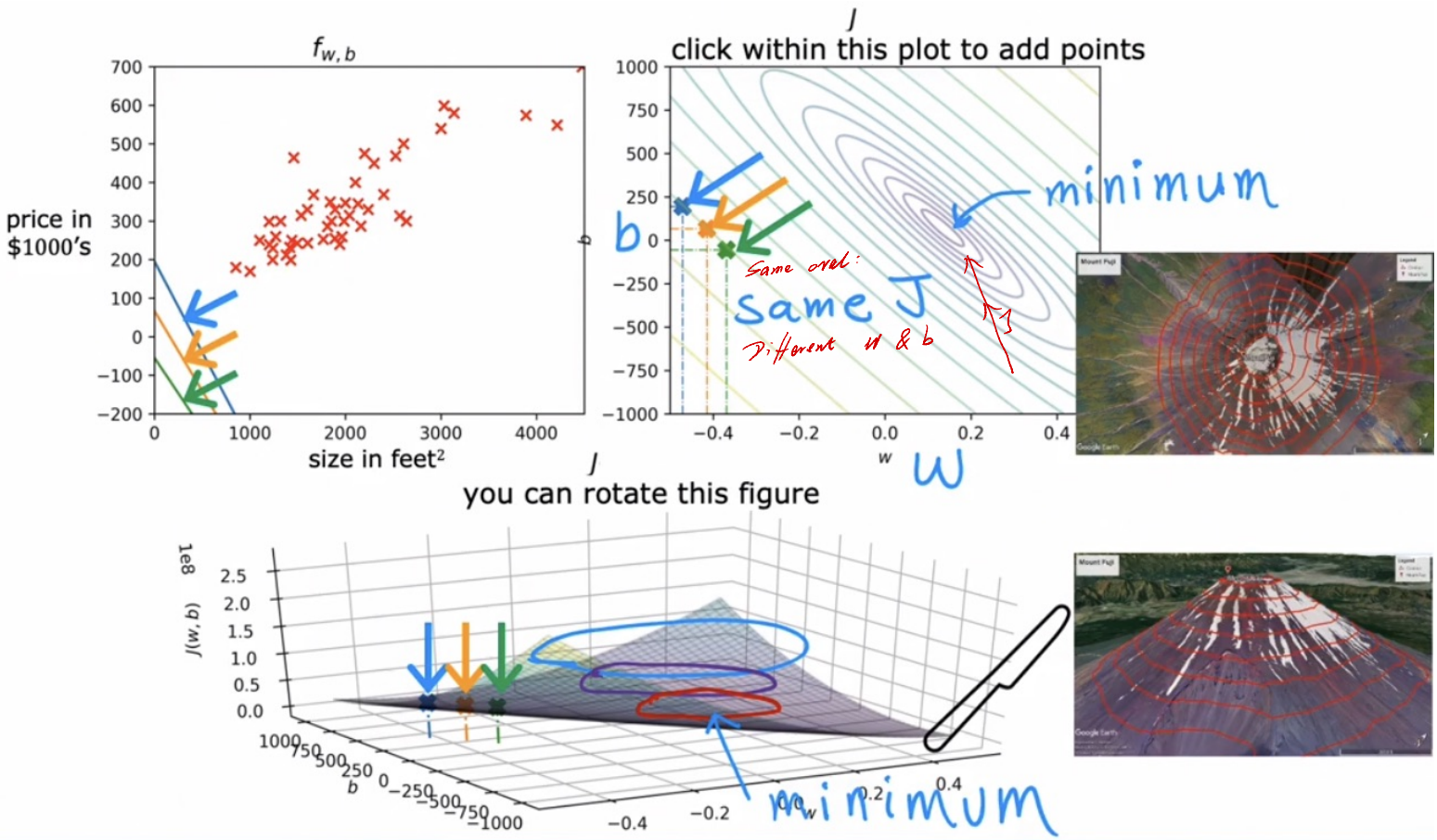
m = number of training examples

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

Find w, b:

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

$\min J(w,b)$ 返回 J^* $\arg\min J(w,b)$ 返回 w^*, b^*



To find w & b that min J

↓

Gradient Descent

梯度下降

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

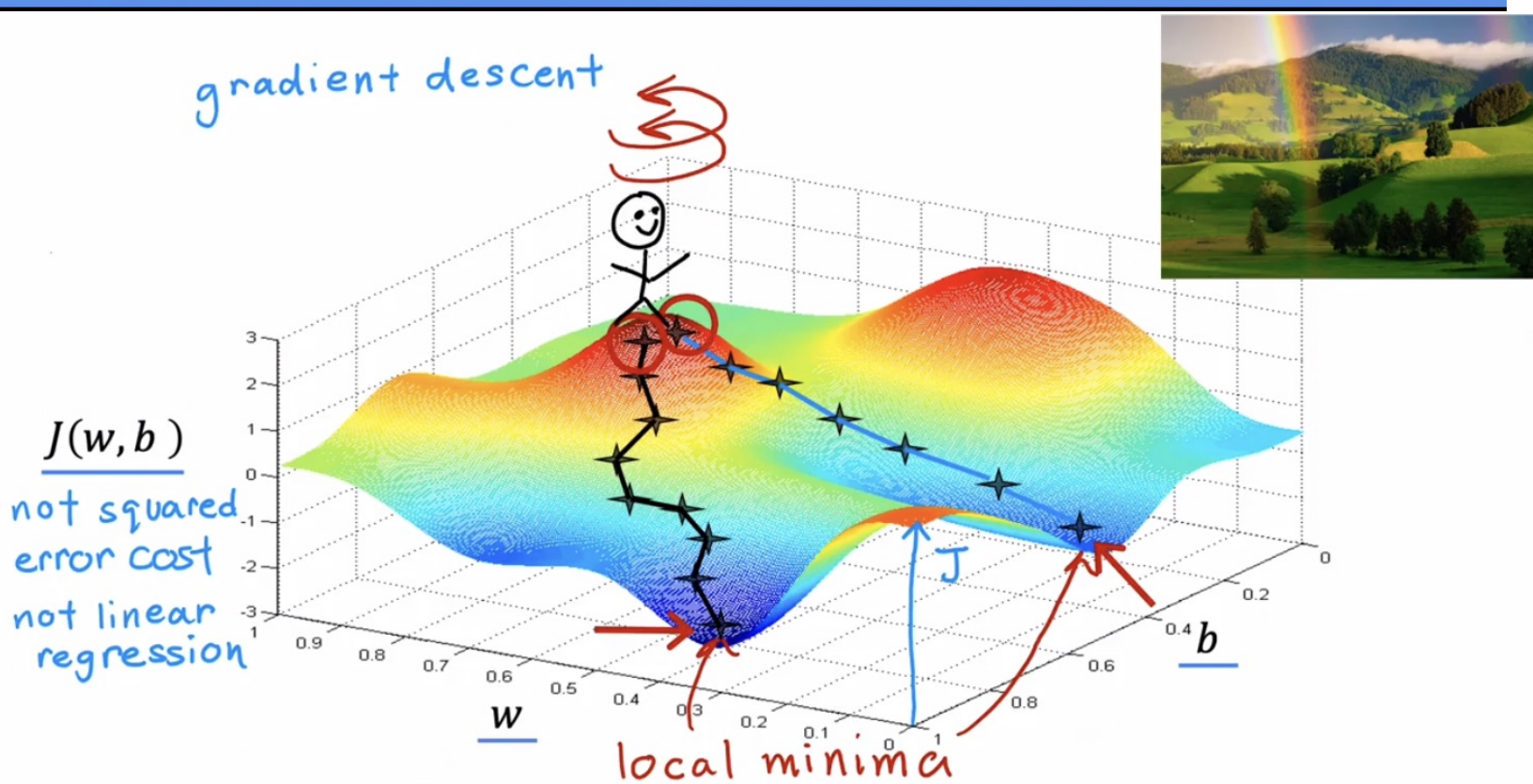
Want $\min_{w,b} J(w,b)$ $\min_{w_1,\ldots,w_n,b} J(w_1,w_2,\ldots,w_n,b)$

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



局部极小

global minima: 全局最小

Gradient descent algorithm

Repeat until convergence

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

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

$\alpha \sim (0,1)$ Learning rate

Derivative

Simultaneously update w and b

Assignment

$a = c$

$a = a + 1$

Code

Truth assertion

$a = c$

$a = a + 1$ ✗

Math

a==c

Correct: Simultaneous update

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

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

$$w = tmp\_w$$

$$b = tmp\_b$$

同步更新

Incorrect

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

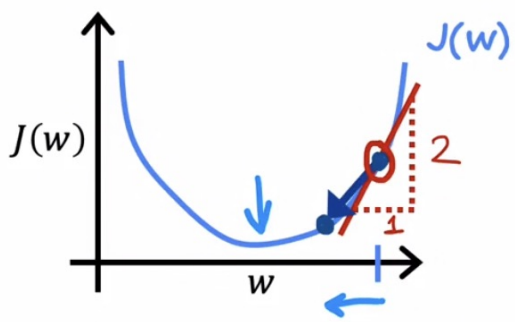
$$w = tmp\_w$$

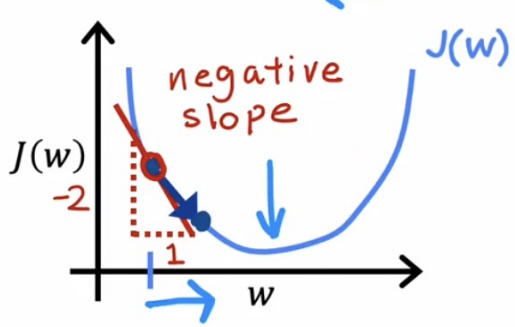
$$tmp\_b = b - \alpha \frac{\partial}{\partial b} J(w,b)$$

$$b = tmp\_b$$

Stanford ONLINE DeepLearning.AI Andrew Ng

Derivative





设 $b=0$:

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

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

$$w = w - \alpha \cdot (positive\ number)$$

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

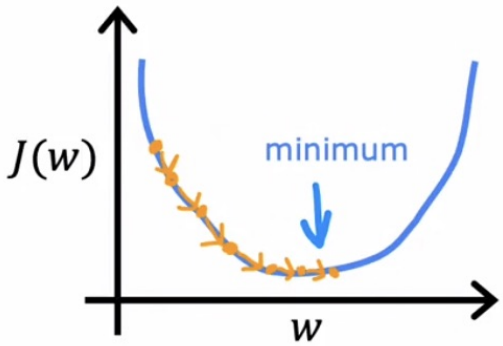
$$w = w - \alpha \cdot (negative\ number)$$

Stanford ONLINE DeepLearning.AI Andrew Ng

some of the intuition behind what a derivative term

The Learning Rate controls how big of a step you take when updating the model's parameters.

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

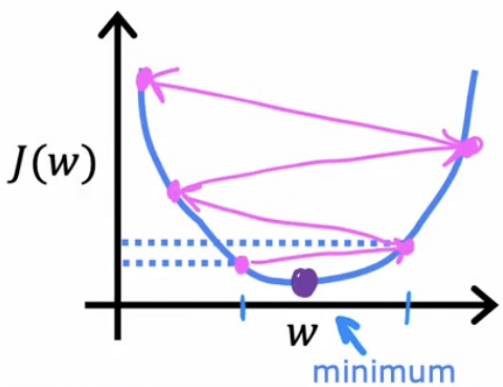


If α is too small...
Gradient descent may be slow.

If α is too large...

Gradient descent may:
- Overshoot, never reach minimum
- Fail to converge, diverge

无法收敛，发散



Can reach local minimum with fixed learning rate

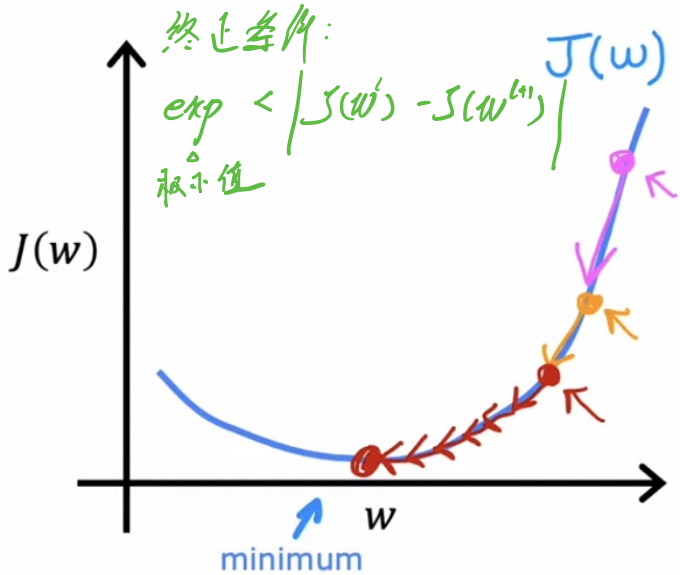
α

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

smaller

not as large

large



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

Can reach minimum without decreasing learning rate α

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

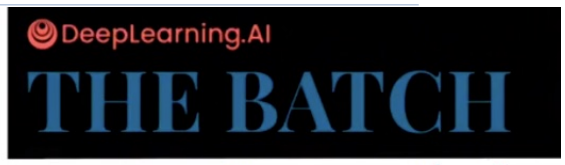
repeat until convergence {

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

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

}

"Batch" gradient descent



other gradient descent: subsets

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

	x size in feet²	y price in $1000's
(1)	2104	400
(2)	1416	232
(3)	1534	315
(4)	852	178
...	...	...
(47)	3210	870

$m = 47$

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

多元(特征)线性回归

Multiple features (variables)

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

$j = 1 \dots 4$

$n = 4$

$i = 2$

x_j = j^{th} feature

n = number of features

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

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

特征向量

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

$x_3^{(2)} = 2$

Stanford ONLINE DeepLearning.AI Andrew Ng

有时为了强调x2不是一个数字 它实际上是一个向量的数字列表

actually a list of numbers that is a vector,

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

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

b is a number

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

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

dot product 点积

multiple linear regression

(not multivariate regression)

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

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

code: count from 0

Without vectorization

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

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

range(0, n) → $j = 0 \ldots n-1$

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

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
```

Vectorization

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

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

dot product 点积

向量化原理示例

Without vectorization

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

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

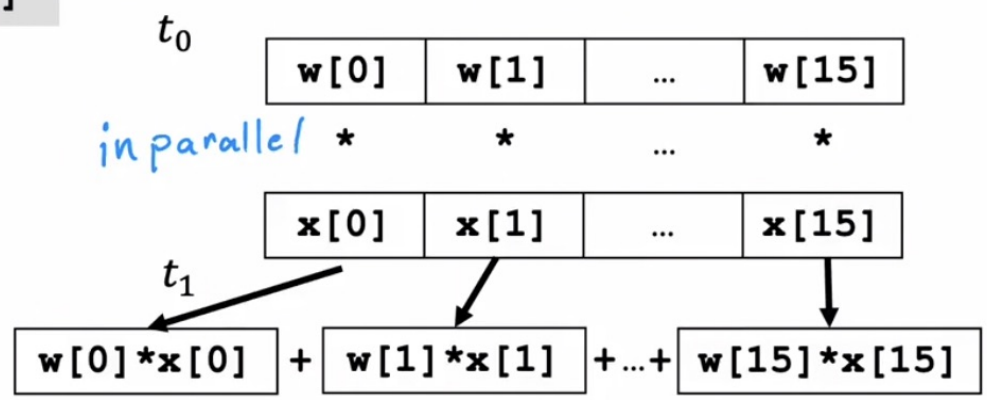
t_1
`f + w[1] * x[1]`

...

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

Vectorization

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



efficient → scale to large datasets

Gradient descent

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

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

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

learning rate α

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

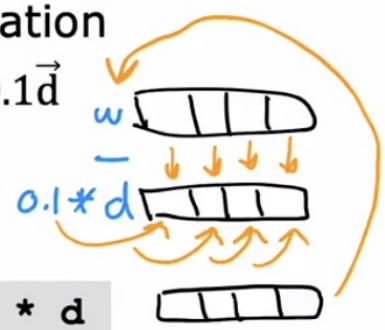
Without vectorization

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

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

With vectorization

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




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

Gradient Descent for Multiple Regression

	Previous notation	Vector notation
Parameters	$w_1, \cdots, w_n$ b	$\vec{w} = [w_1 \quad \cdots \quad w_n]$ ← vector of length n b still a number
Model	$f_{\vec{w},b}(\vec{x}) = w_1 x_1 + \cdots + w_n x_n + b$	$f_{\vec{w},b}(\vec{x}) = \vec{w} \cdot \vec{x} + b$ (dot product)
Cost function	$J(w_1, \cdots, w_n, b)$	$J(\vec{w}, b)$

Gradient descent

repeat {

$w_j = w_j - \alpha \frac{\partial}{\partial w_j} J(w_1, \cdots, w_n, b)$

$b = b - \alpha \frac{\partial}{\partial b} J(w_1, \cdots, w_n, b)$

}

repeat {

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

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

}

Stanford ONLINE DeepLearning.AI Andrew Ng

Gradient descent

One feature

repeat {

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

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

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

simultaneously update w, b

}

n features ($n \geq 2$)

repeat {

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

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

$\vdots$

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

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

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

}

Stanford ONLINE DeepLearning.AI Andrew Ng

Make Gradient Descent faster

Feature scaling

缩放

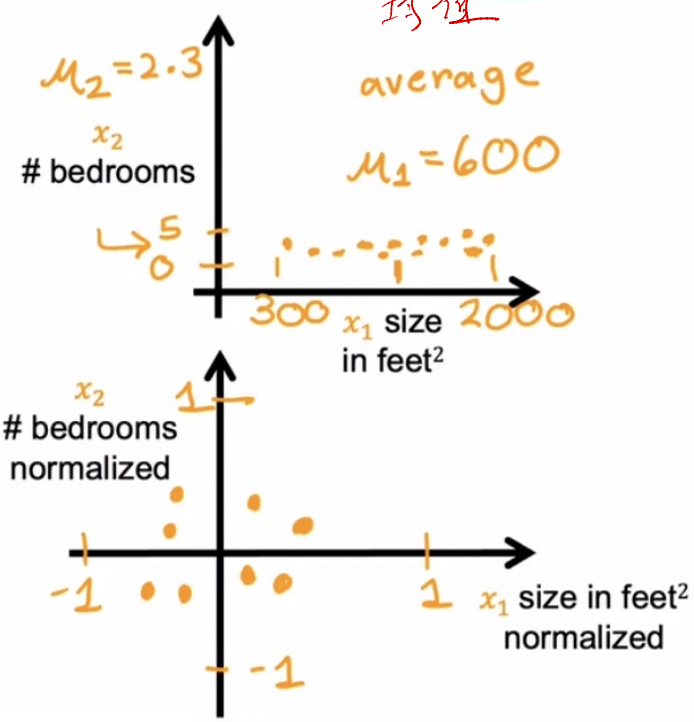
$300 \le x_1 \le 2000$ $\quad$ $0 \le x_2 \le 5$

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

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

Mean normalization

均值

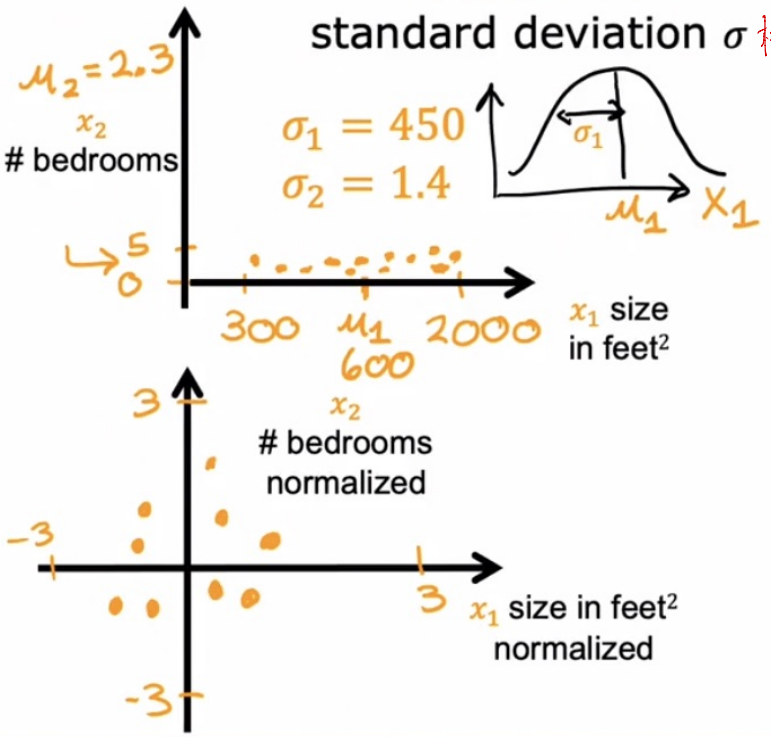


$300 \le x_1 \le 2000$ $\quad$ $0 \le x_2 \le 5$

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

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

Z-score normalization



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

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

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

Feature scaling

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

$-3 \le x_j \le 3$

$-0.3 \le x_j \le 0.3$

} acceptable ranges

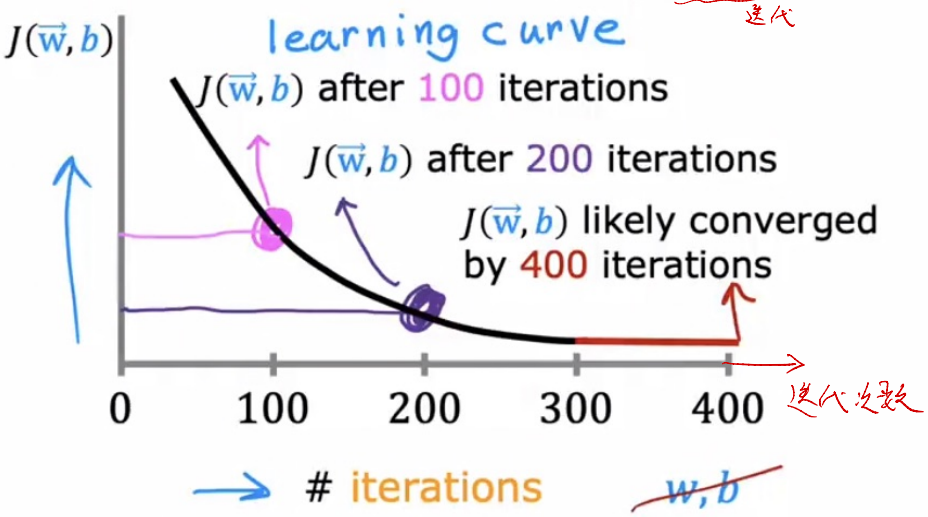
$0 \le x_1 \le 3$	okay, no rescaling
$-2 \le x_2 \le 0.5$	okay, no rescaling
$-100 \le x_3 \le 100$	too large → rescale
$-0.001 \le x_4 \le 0.001$	too small → rescale
$98.6 \le x_5 \le 105$	too large → rescale

Make sure gradient descent is working correctly

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

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

迭代



~~w,b~~

iterations needed varies 30 1,000 100,000

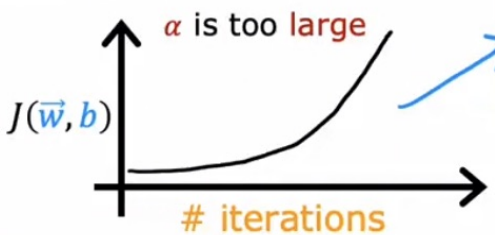
Automatic convergence test

Let ε "epsilon" be 10^{-3}. 0.001

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

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

Identify problem with gradient descent



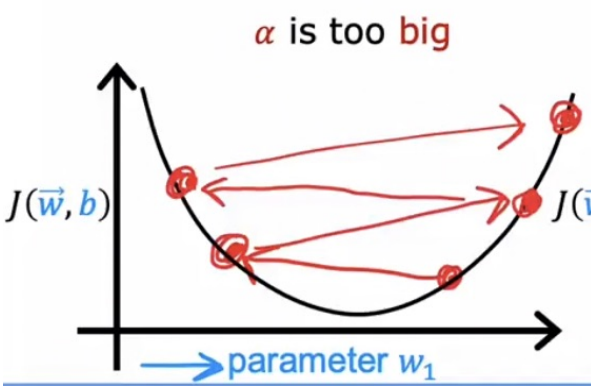
or learning rate is too large

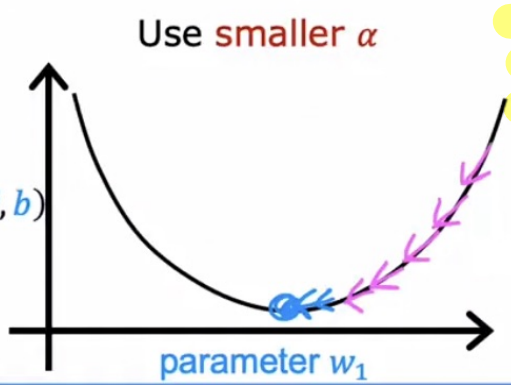
$w_1 = w_1 + \alpha d_1$ ☹

use a minus sign

$w_1 = w_1 - \alpha d_1$ ☺

Adjust learning rate





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

If not, code error/bug

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

Feature engineering

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

frontage depth

$$area = frontage \times depth$$

$$x_3 = x_1 x_2$$

new feature

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

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

Classification

Classification

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

y can only be one of two values

"binary classification"

class = category

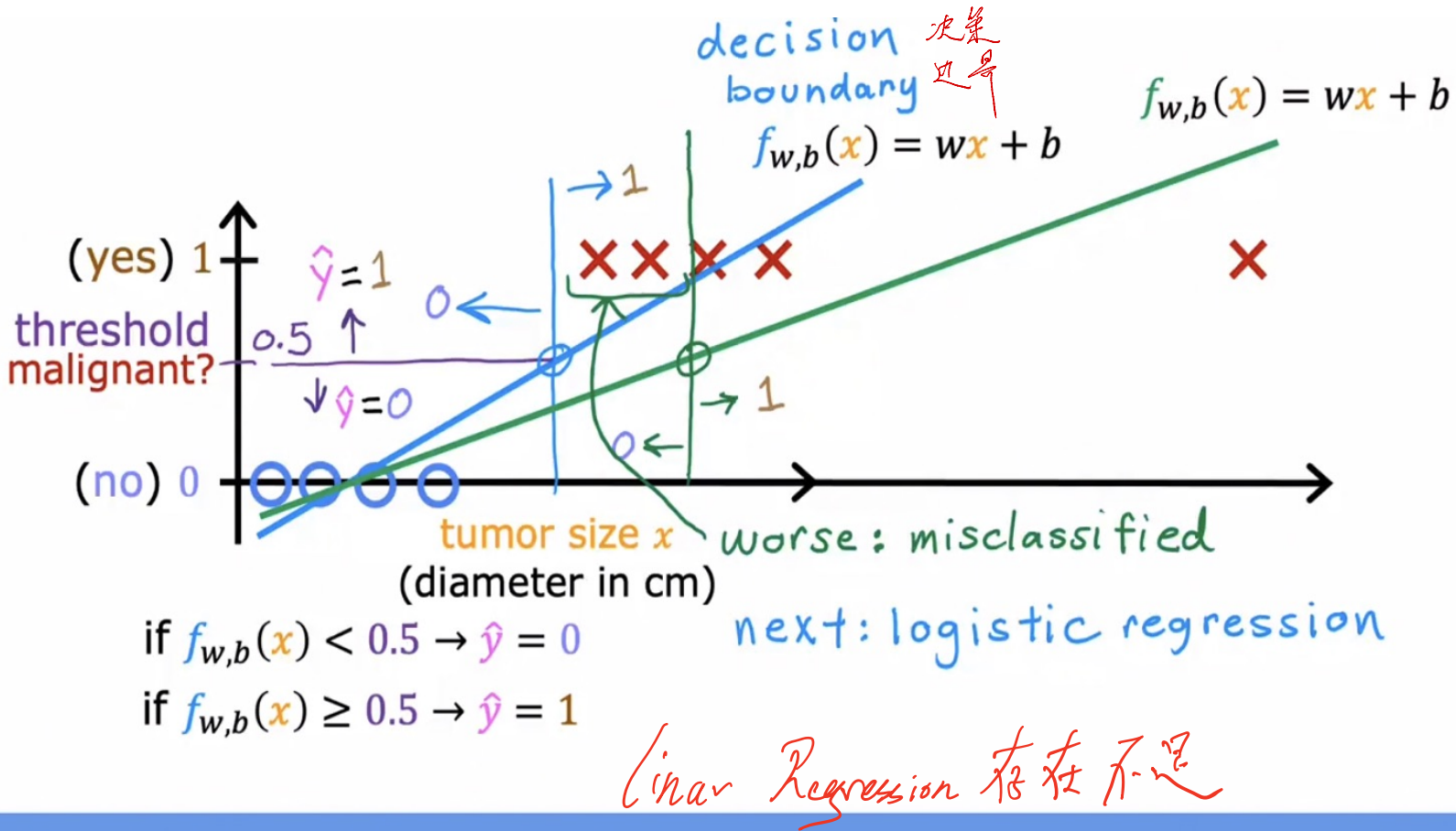
false true

0 1

useful for classification

"negative class" ≠ "bad" absence

"positive class" ≠ "good" presence

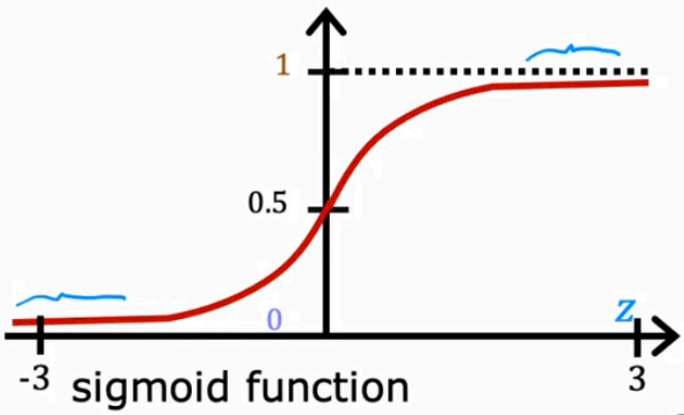


顺便说一下，逻辑回归这个名字让人困惑的一点是，尽管它里面有回归（regression）这个词，但它实际上是用来分类的。

By the way one thing confusing about the name logistic regression is that even though it has the word of regression in it is actually used for classification.

Logistic Regression

Want outputs between 0 and 1



sigmoid function

logistic function

outputs between 0 and 1

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

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

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

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

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

$e \approx 2.7$

"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

$P(1)$

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

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

Example:

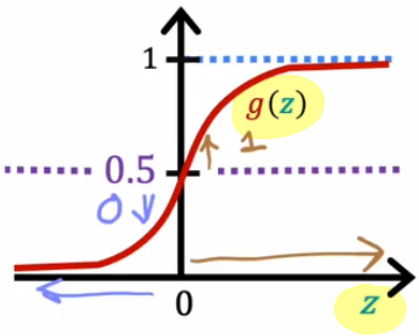
x is "tumor size"

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

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

70% chance that y is 1

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



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

z 是中间值，可自由设置

g 是预测值

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

0 or 1? threshold

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

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

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

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

z

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

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

Sigmoid 关于 $g(z)$ 均可作判定条件.

$g(z) \geq 0.5$

$z \geq 0$

$\vec{w}\cdot\vec{x}+b \geq 0$ $\hat{y}=1$

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

Decision boundary

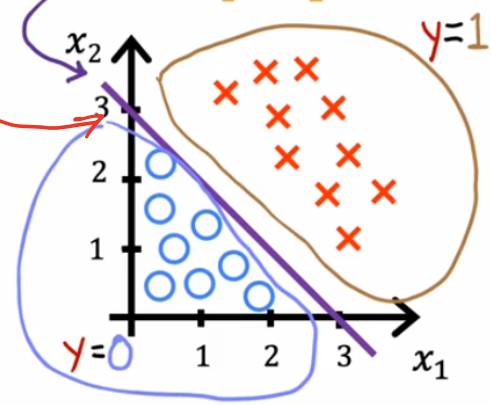
$$f_{\vec{w},b}(\vec{x}) = g(z) = g(w_1x_1 + w_2x_2 + b)$$

1 1 -3

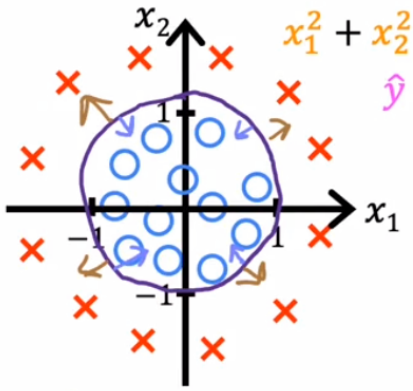
Decision boundary $z = \vec{w}\cdot\vec{x}+b = 0$ $\Rightarrow g(z)=0.5$ 多特征.

$z = x_1 + x_2 - 3 = 0$

$x_1 + x_2 = 3$



Non-linear decision boundaries



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

$\hat{y} = 1$

$x_1^2 + x_2^2 < 1$

$\hat{y} = 0$

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

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

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

$x_1^2 + x_2^2 = 1$

Cost Function

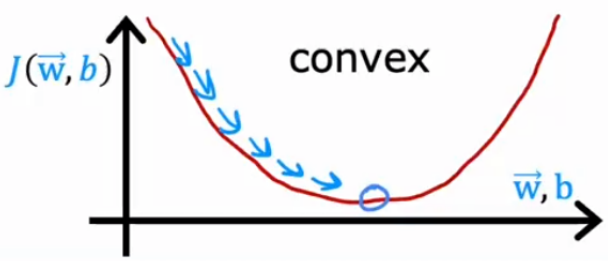
Squared error cost

But Choice

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

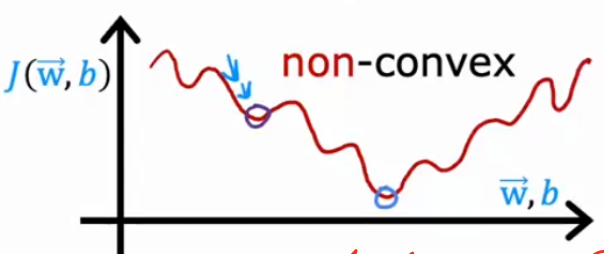
linear regression

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



logistic regression

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

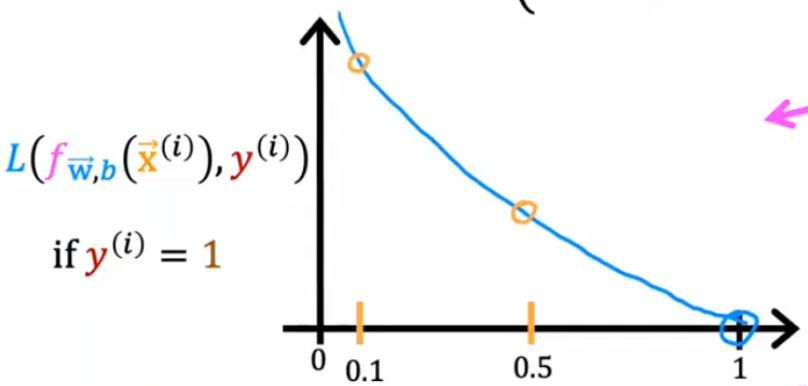


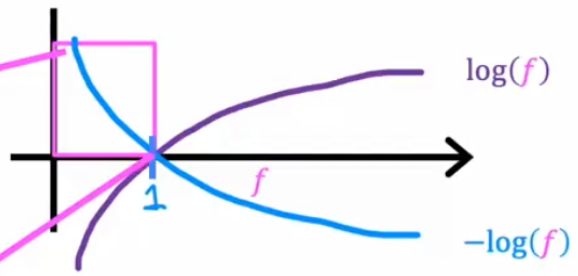
Global min?

Logistic loss function

Part of Cost Function

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





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

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

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

Simplified cost function

loss

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

convex (single global minimum)

cost

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

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

maximum likelihood (don't worry about it!)

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 {

$j = 1 \ldots n$

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

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

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

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

} simultaneous updates

Gradient descent for logistic regression

repeat {

looks like linear regression!

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

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

} simultaneous updates

Same concepts:

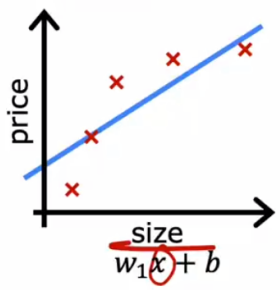
- Monitor gradient descent (learning curve)
- Vectorized implementation
- Feature scaling

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

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

Overfitting & Underfitting

Regression example

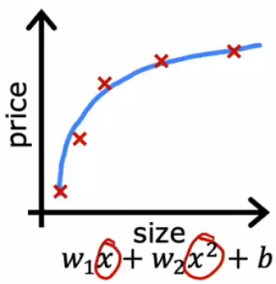


$w_1x + b$

underfit

- Does not fit the training set well

high bias

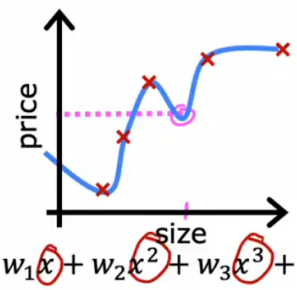


$w_1x + w_2x^2 + b$

just right

- Fits training set pretty well

generalization 泛化



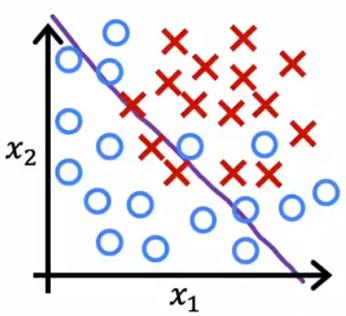
$w_1x + w_2x^2 + w_3x^3 + w_4x^4 + b$

overfit

- Fits the training set extremely well

high variance

Classification

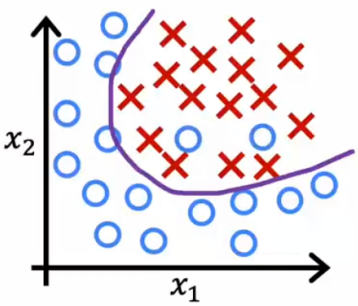


$z = w_1x_1 + w_2x_2 + b$

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

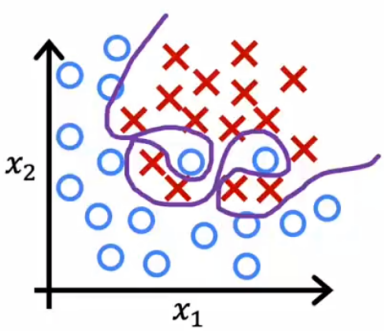
g is the sigmoid function

underfit high bias 高偏差



$z = w_1x_1 + w_2x_2 + w_3x_1^2 + w_4x_2^2 + w_5x_1x_2 + b$

just right



$z = w_1x_1 + w_2x_2 + w_3x_1^2x_2 + w_4x_1^2x_2^2 + w_5x_1^2x_2^3 + w_6x_1^3x_2 + \cdots + b$

overfit

Addressing overfitting

Options

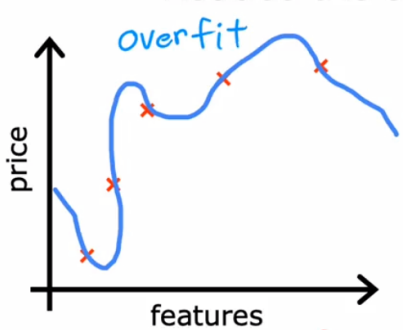
1. Collect more data

2. Select features
   - Feature selection in course 2

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

Regularization

Reduce the size of parameters w_j



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

large values for w_j

eliminate feature

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

small values for w_j

Regularization

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

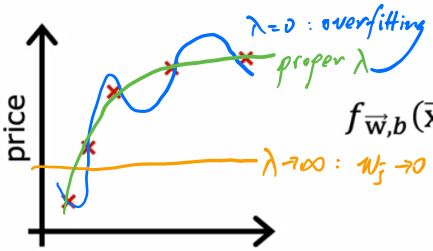
mean squared error

regularization term

fit data

Keep w_j small

λ balances both goals



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

Regularized linear regression

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

Gradient descent

repeat {

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

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

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

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

don't have to regularize b

} simultaneous update

Implementing gradient descent

repeat {

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

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

} simultaneous update $j = 1 \ldots n$

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

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

usual update

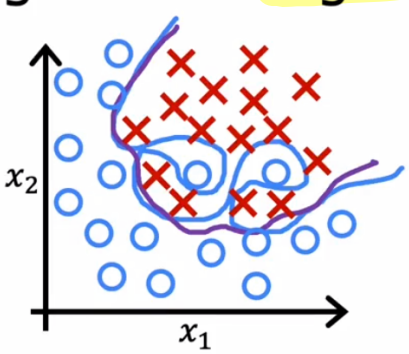
$\alpha \frac{\lambda}{m}$

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

$w_j (1 - 0.0002)$

0.9998

Regularized logistic regression



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

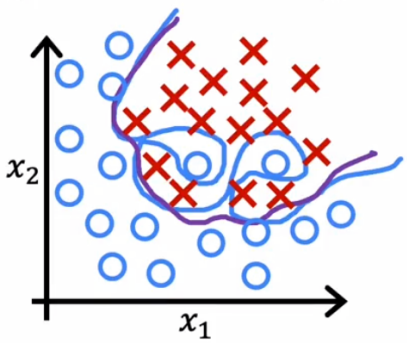
$$f_{\vec{w},b}(\vec{x}) = \frac{1}{1 + e^{-z}}$$

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



$$z = w_1x_1 + w_2x_2 + w_3x_1^2x_2 + w_4x_1^2x_2^2 + w_5x_1^2x_2^3 + \cdots + b$$

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

Cost function

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

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

有一点需要注意，“线性回归”中的“线性”指的是**因变量关于参数是线性的**，而不是关于自变量，比如模型：$Y_i = \beta_1 X_i + \beta_2 X_i^2 + \varepsilon_i$，因变量对于参数 β_1 和 β_2 是线性的，但对于自变量 X_i 来说就是非线性的，这任然是一个线性回归模型。

不过对于**线性回归**，它们的代价函数图像并非上面那样，**线性回归的代价函数图像总是一个凸函数（convex function）**，如下图，它并没有“局部最优解”，它只有一个“全局最优解”。