

# Copyright Notice

These slides are distributed under the Creative Commons License.

[DeepLearning.AI](#) makes these slides available for educational purposes. You may not use or distribute these slides for commercial purposes. You may make copies of these slides and use or distribute them for educational purposes as long as you cite [DeepLearning.AI](#) as the source of the slides.

For the rest of the details of the license, see <https://creativecommons.org/licenses/by-sa/2.0/legalcode>

Stanford  
ONLINE

DeepLearning.AI



# Classification

---

## Motivations

# Classification

Question

Is this email spam?

Is the transaction fraudulent?

Is the tumor malignant?

Answer " $y$ "

no	yes
no	yes
no	yes

$y$  can only be one of **two** values

"binary classification"

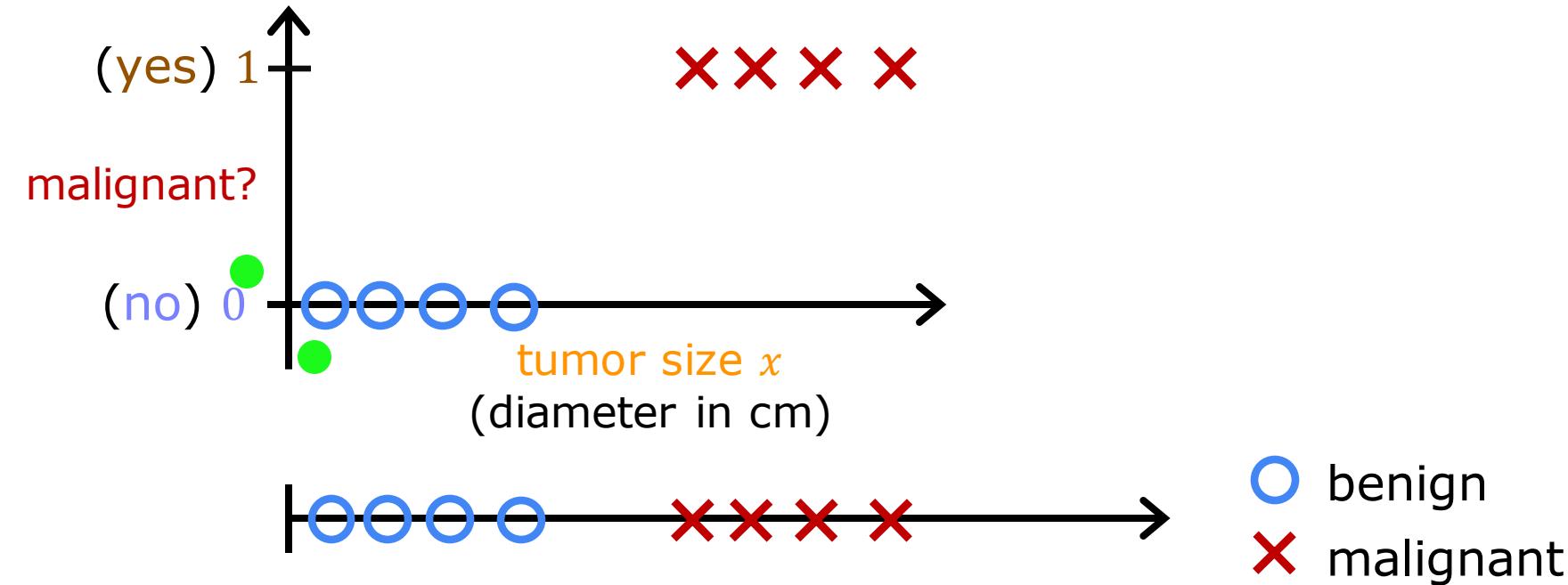
class = category

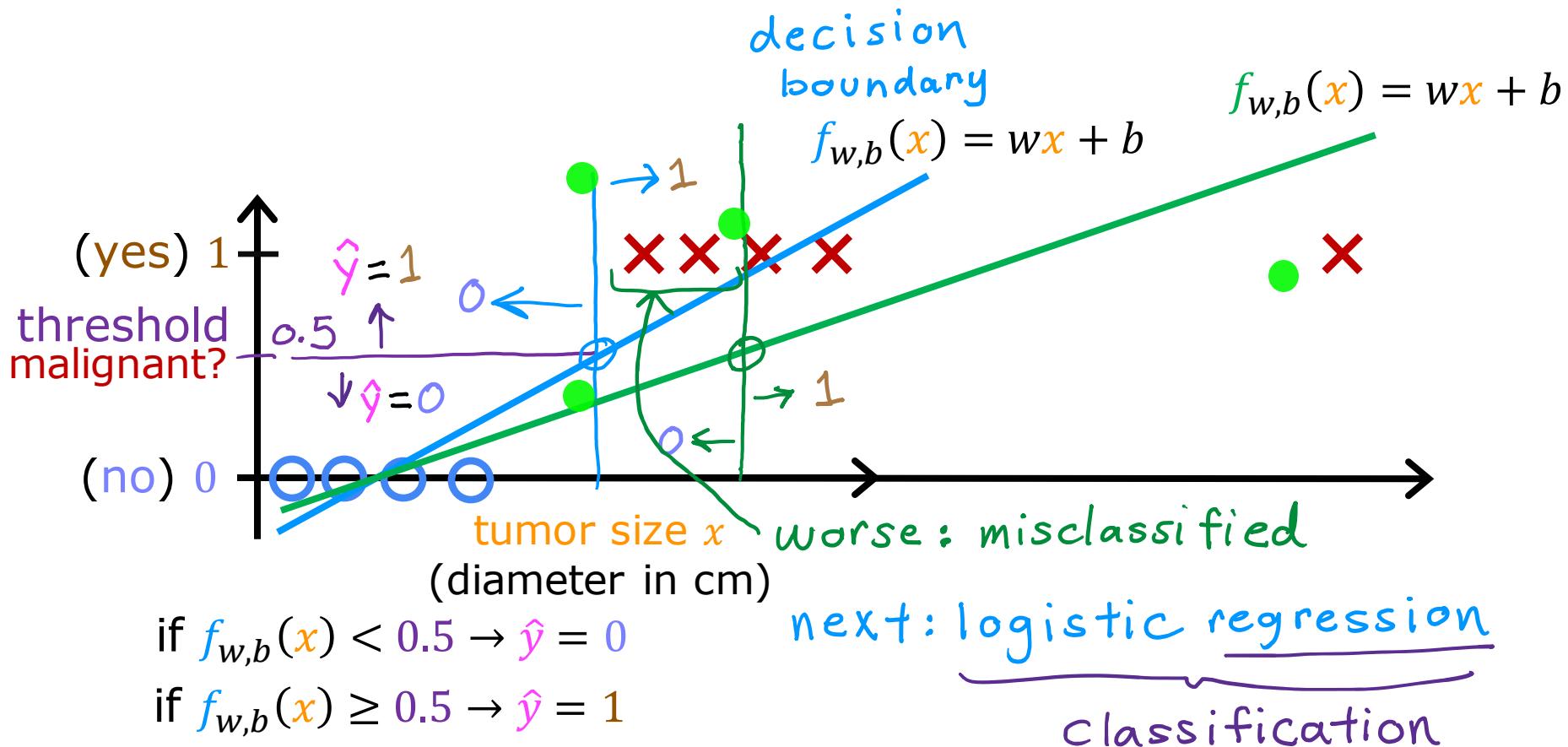
"negative class"  
 $\neq$  "bad"  
absence

false      true  
0            1

useful for  
classification

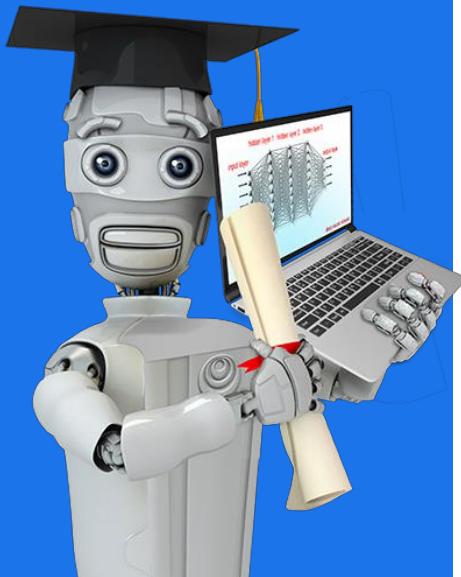
"positive class"  
 $\neq$  "good"  
presence





Stanford  
ONLINE

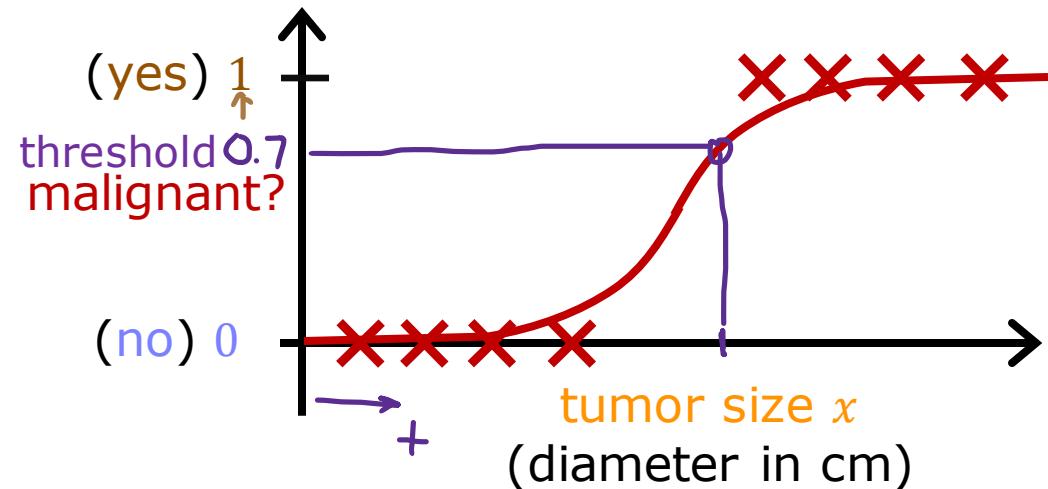
DeepLearning.AI



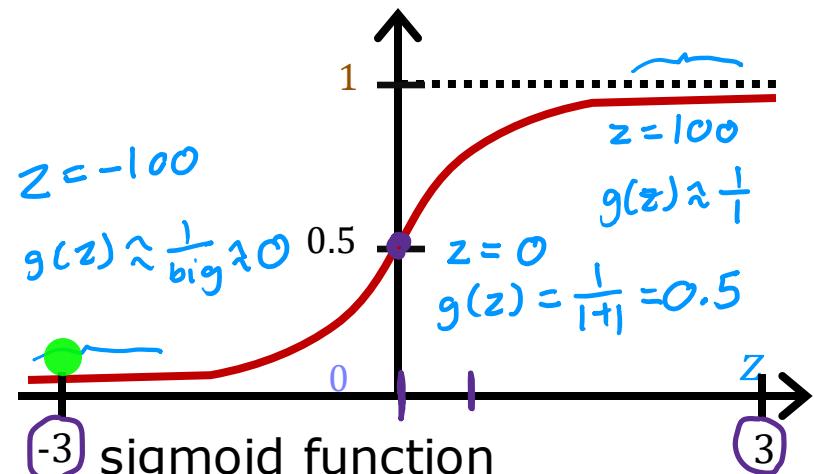
# Classification

---

## Logistic Regression



Want outputs between 0 and 1

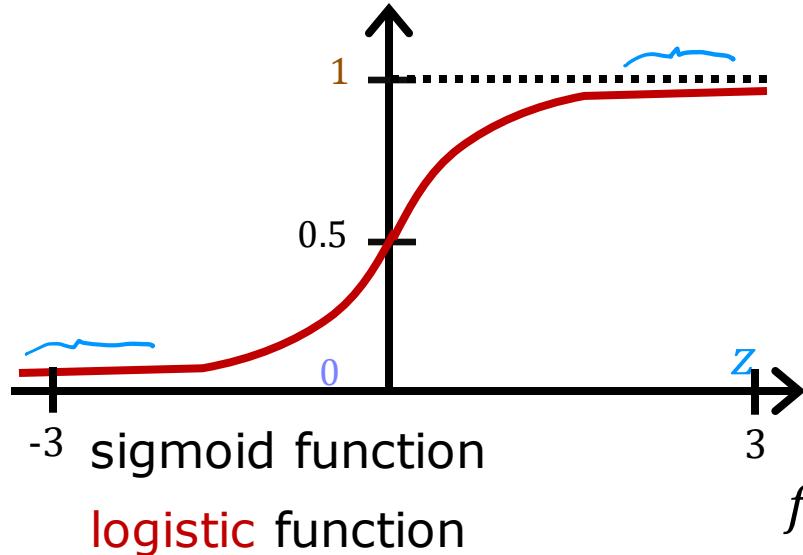


logistic function

outputs between 0 and 1

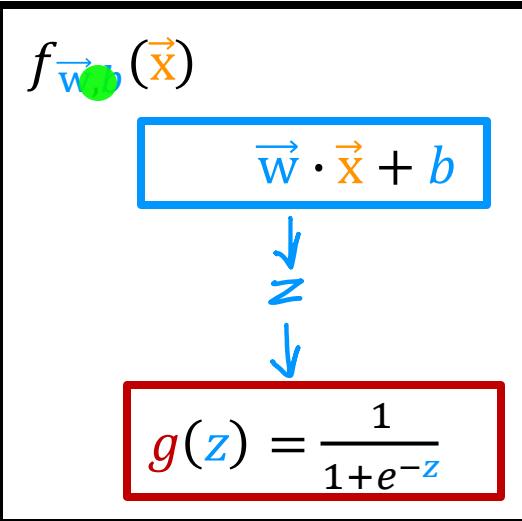
$$g(z) = \frac{1}{1+e^{-z}} \quad 0 < g(z) < 1$$

Want outputs between 0 and 1



outputs between 0 and 1

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



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

"logistic regression"

$e \approx 2.7$

# Interpretation of logistic regression output

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

“probability” that class is 1

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

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

Example:

$x$  is “tumor size”

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

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

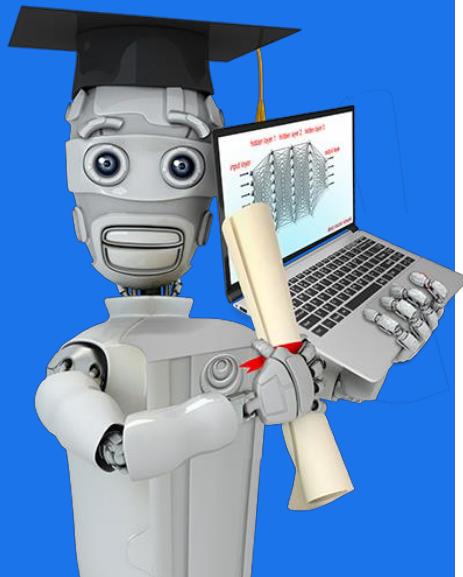
$$f_{\vec{w}, b}(\vec{x}) = 0.7$$

70% chance that  $y$  is 1



Stanford  
ONLINE

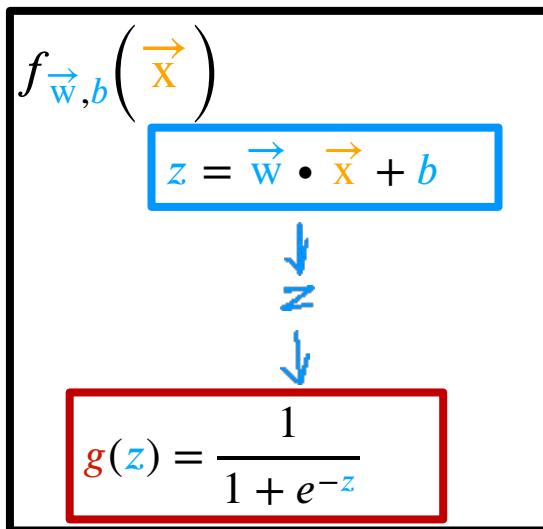
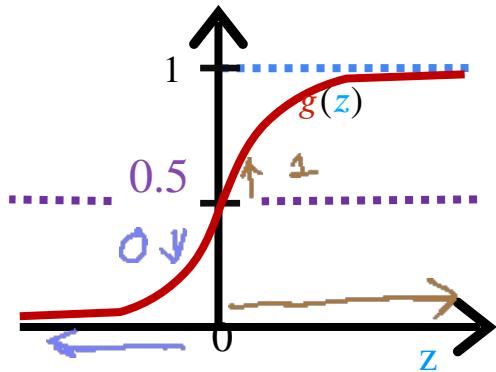
DeepLearning.AI



# Classification

---

## Decision Boundary



$$f_{\vec{w}, b}(\vec{x}) = g(\underbrace{\vec{w} \cdot \vec{x}}_z + \bar{b}) = P(y = 1 | \vec{x}; \vec{w}, b)$$

0 or 1? threshold

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

Yes:  $\hat{y} = 1$

No:  $\hat{y} = 0$

When is

$$f_{\vec{w}, b}(\vec{x}) \geq 0 \quad g(z) \geq 0.5$$

$$z \geq 0$$

$$z < 0$$

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

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

$$\hat{y} = 1$$

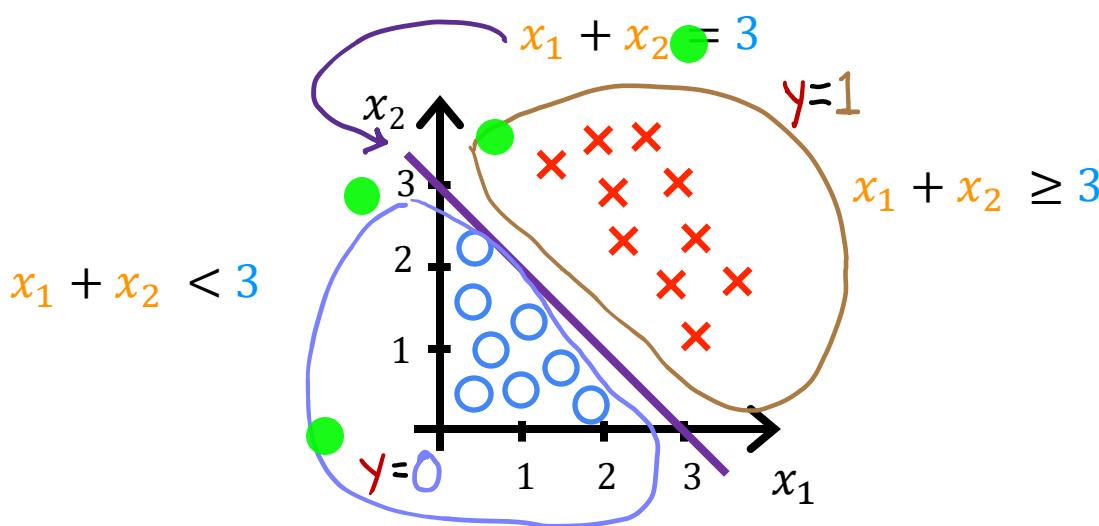
$$\hat{y} = 0$$

# Decision boundary

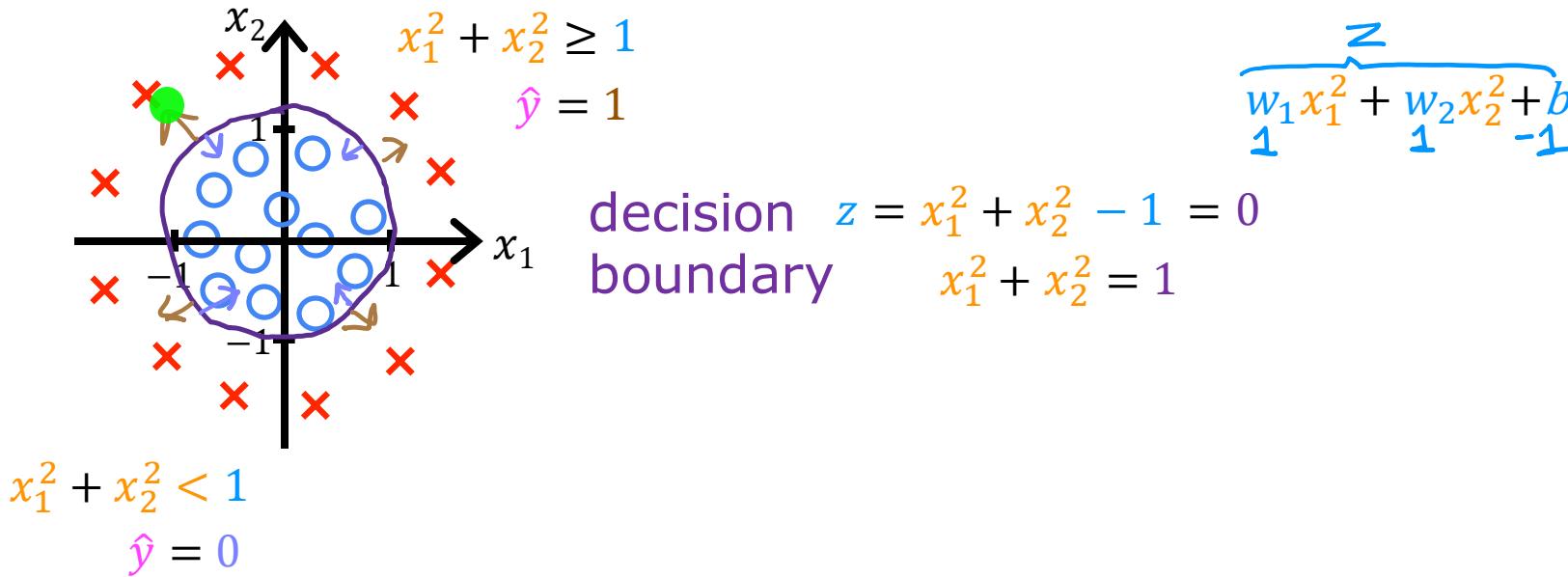
$$f_{\vec{w}, b}(\vec{x}) = g(z) = g(w_1 x_1 + w_2 x_2 + b)$$

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

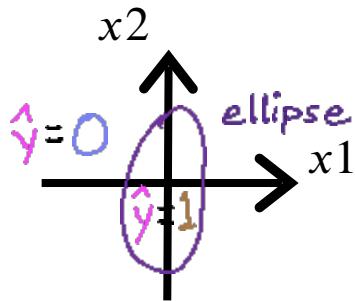
$$z = x_1 + x_2 - 3 = 0$$



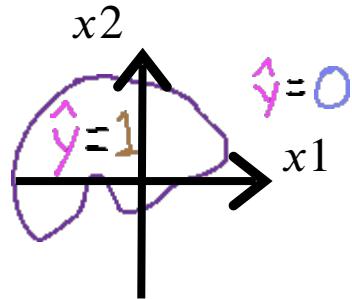
# Non-linear decision boundaries



# Non-linear decision boundaries



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



Stanford  
ONLINE

DeepLearning.AI



# Cost Function

---

Cost Function for  
Logistic Regression

# • Training set

	tumor size (cm) $x_1$	...	patient's age $x_n$	malignant? $y$	$i = 1, \dots, m$ ↪ training examples
$i=1$	10		52	1	target $y$ is 0 or 1
:	2		73	0	
:	5		55	0	
$i=m$	12		49	1	
	...		...	...	

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

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

# Squared error cost

*cost*

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

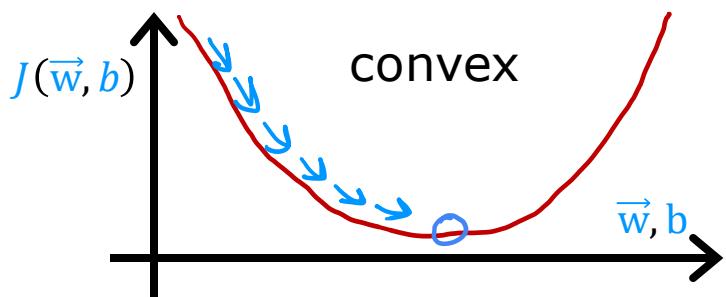
average of training set

*loss*

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

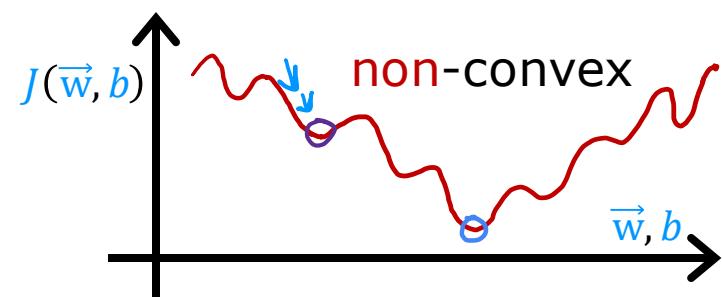
linear regression

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



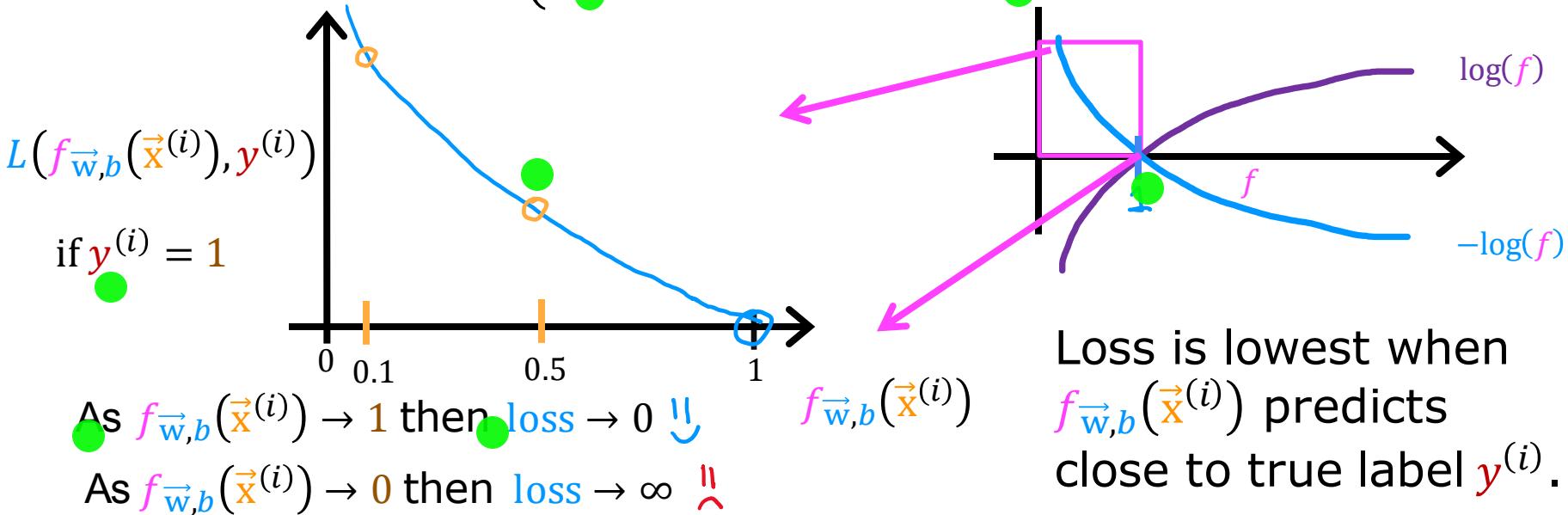
logistic regression

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



# Logistic loss function

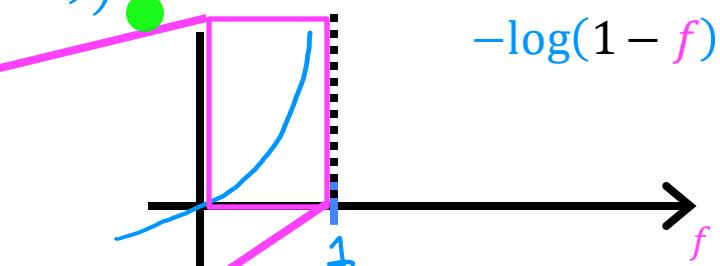
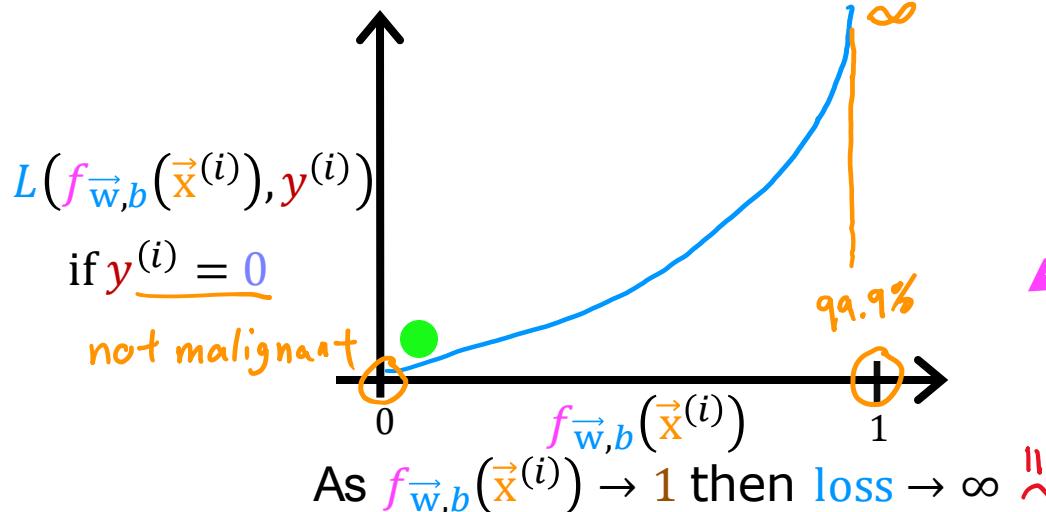
$$L(f_{\vec{w}, b}(\vec{x}^{(i)}), y^{(i)}) = \begin{cases} -\log(f_{\vec{w}, b}(\vec{x}^{(i)})) & \text{if } y^{(i)} = 1 \\ -\log(1 - f_{\vec{w}, b}(\vec{x}^{(i)})) & \text{if } y^{(i)} = 0 \end{cases}$$



# Logistic loss function

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

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



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

# Cost

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

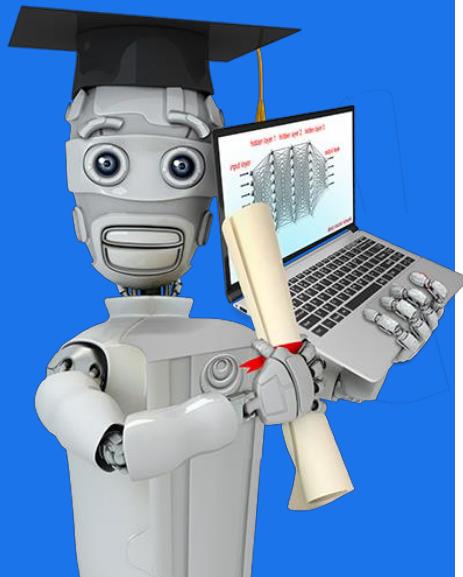
$$= \begin{cases} -\log(f_{\vec{w}, b}(\vec{x}^{(i)})) & \text{if } y^{(i)} = 1 \\ -\log(1 - f_{\vec{w}, b}(\vec{x}^{(i)})) & \text{if } y^{(i)} = 0 \end{cases}$$

*Convex*  
*can reach a global minimum*

find  $w, b$  that minimize cost  $J$

Stanford  
ONLINE

DeepLearning.AI



# Cost Function

---

Simplified Cost  
Function for Logistic  
Regression

# Simplified loss function

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

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

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

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

# Simplified loss function

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

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

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

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

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

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

# Simplified cost function

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

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

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

↑ convex  
(single global minimum)

maximum likelihood  
(don't worry about it!)

Stanford  
ONLINE

DeepLearning.AI



# Gradient Descent

---

## Gradient Descent Implementation

# Training logistic regression

Find  $\vec{w}, b$

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

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

# Gradient descent

*cost*

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

repeat {  
   $j = 1 \dots n$   
   $w_j = w_j - \alpha \frac{\partial}{\partial w_j} J(\vec{w}, b)$   
   $b = b - \alpha \frac{\partial}{\partial b} J(\vec{w}, b)$   
}  
} simultaneous updates

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

# Gradient descent for logistic regression

repeat {

looks like linear regression!

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

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

} simultaneous updates

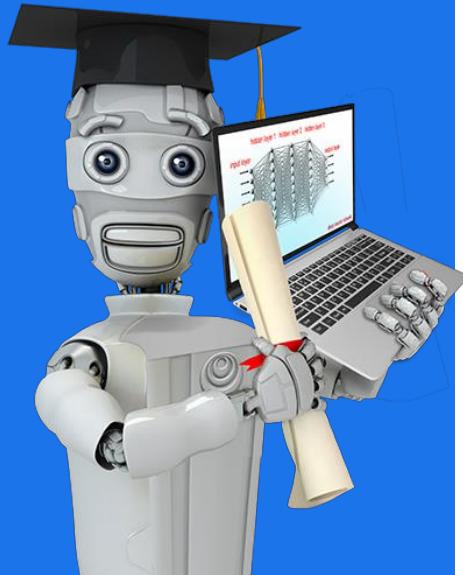
- Same concepts:
- Monitor gradient descent (learning curve)
  - Vectorized implementation
  - Feature scaling

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

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

Stanford  
ONLINE

DeepLearning.AI

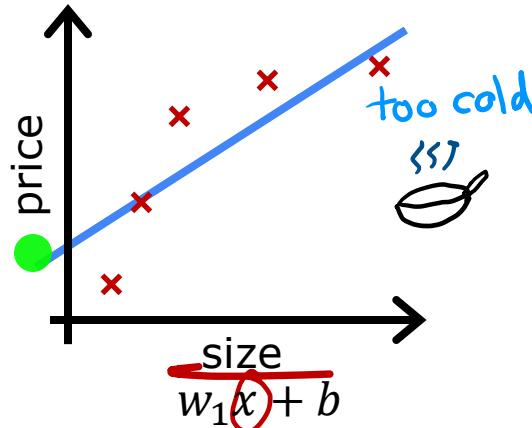


# Regularization to Reduce Overfitting

---

## The Problem of Overfitting

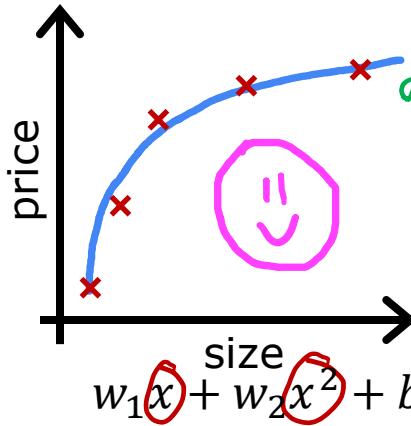
# Regression example



underfit

- Does not fit the training set well

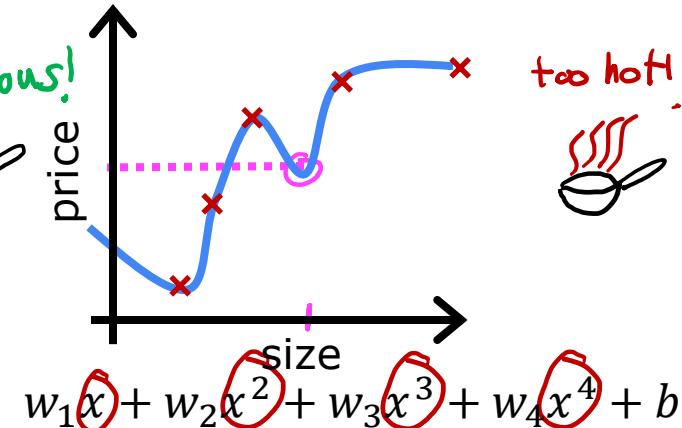
high bias



just right

- Fits training set pretty well

generalization

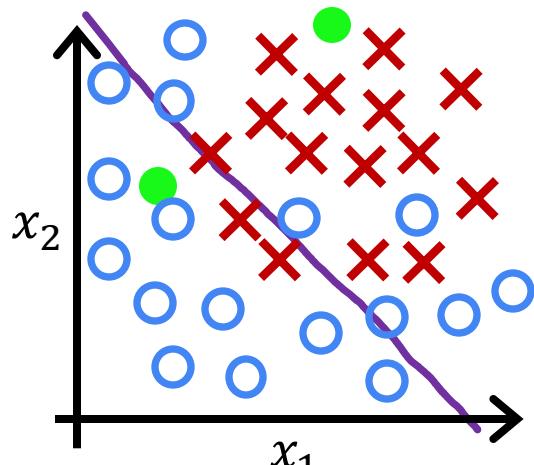


overfit

- Fits the training set extremely well

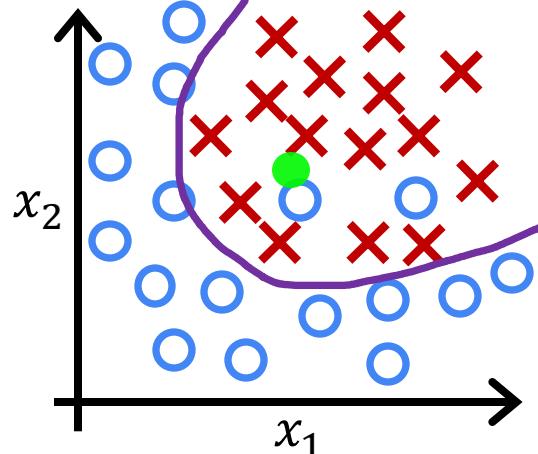
high variance

# Classification



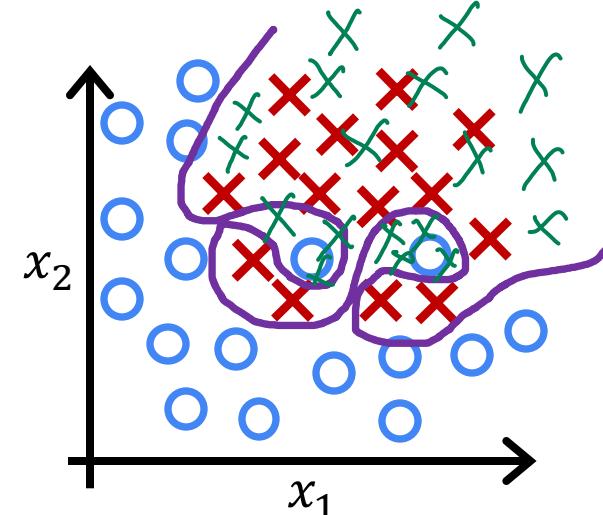
$$z = w_1 x_1 + w_2 x_2 + b$$
$$f_{\vec{w}, b}(\vec{x}) = g(z)$$

$g$  is the sigmoid function  
underfit      high bias



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

just right



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

Overfit

Stanford  
ONLINE

DeepLearning.AI

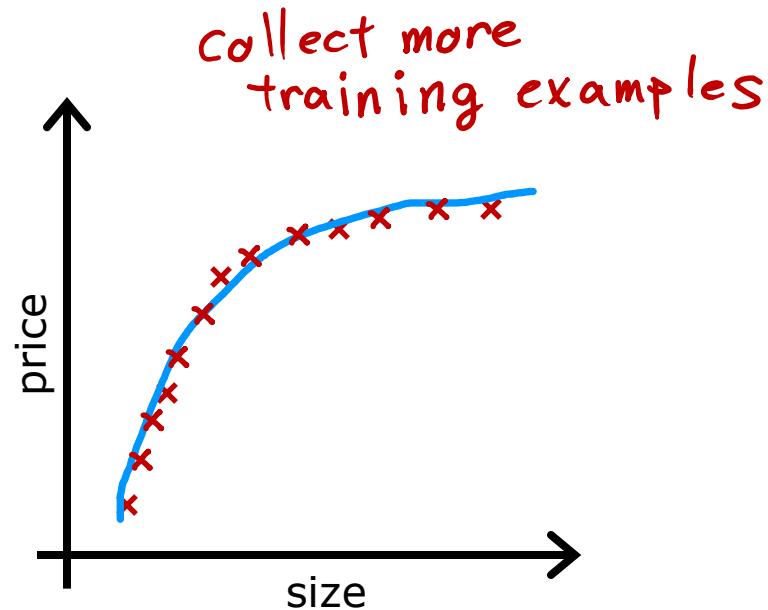
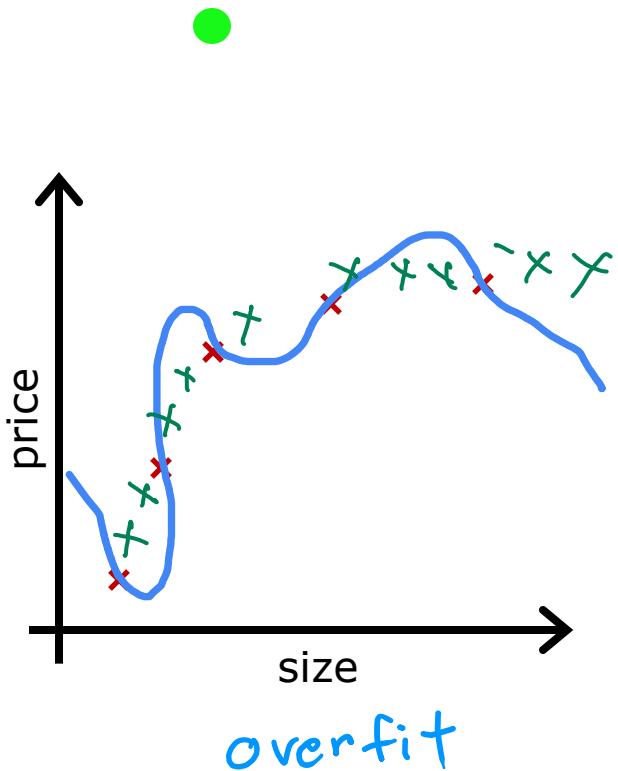


# Regularization to Reduce Overfitting

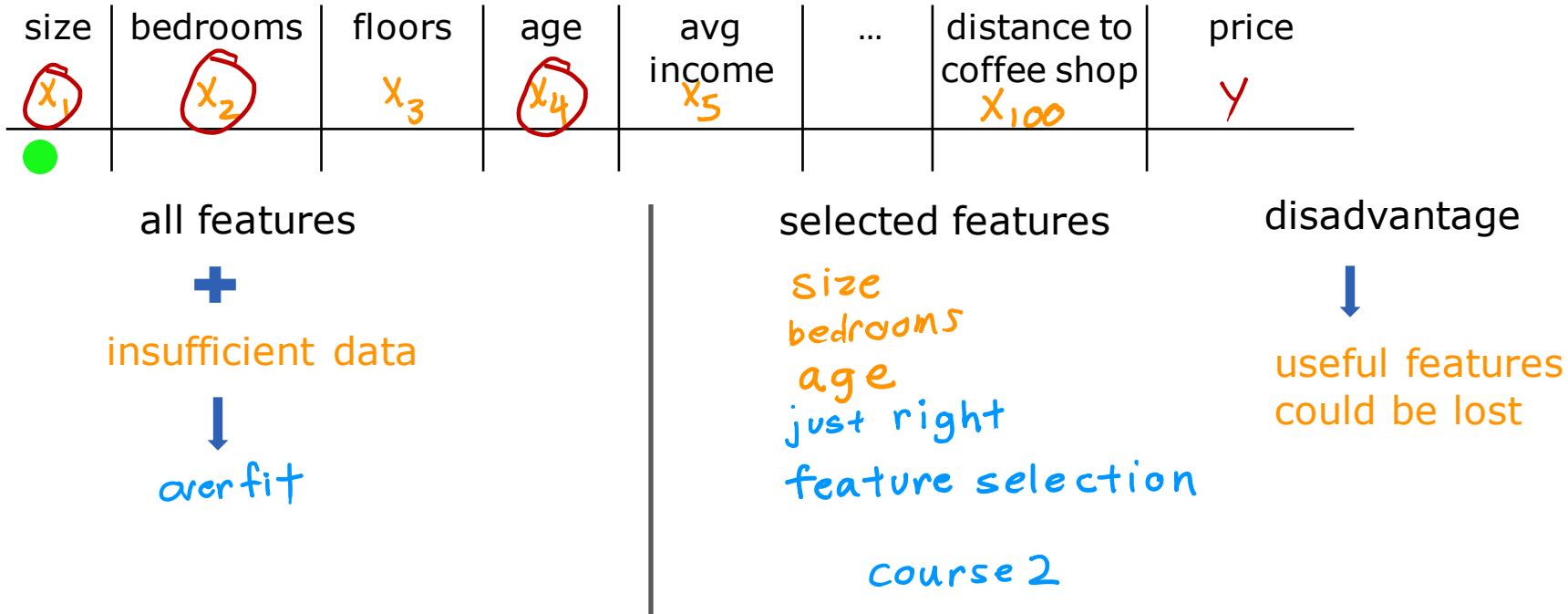
---

## Addressing Overfitting

# Collect more training examples

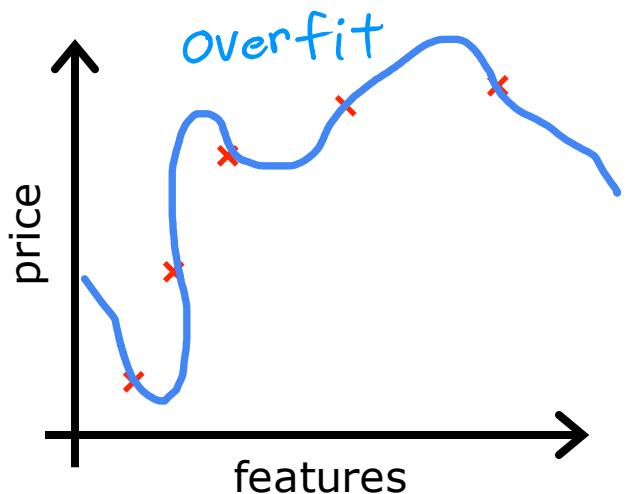


# Select features to include/exclude



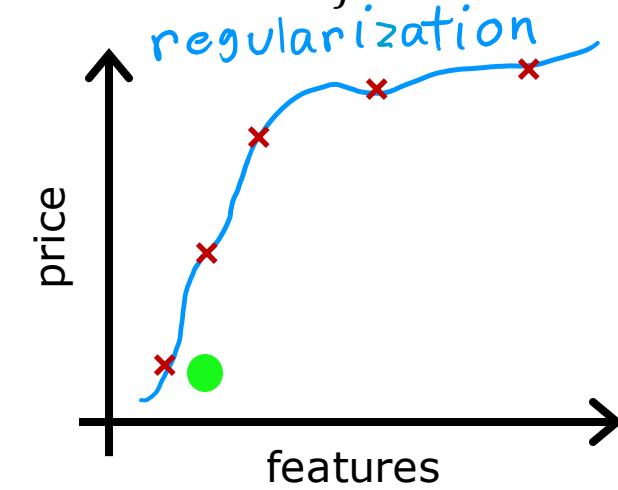
# Regularization

Reduce the size of parameters  $w_j$



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

large values for  $w_j$       eliminate feature



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

small values for  $w_j$

# Addressing overfitting

## Options

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

Stanford  
ONLINE

DeepLearning.AI

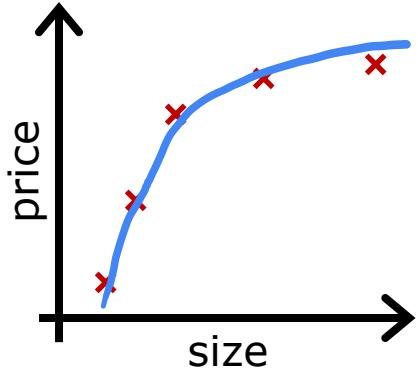


# Regularization to Reduce Overfitting

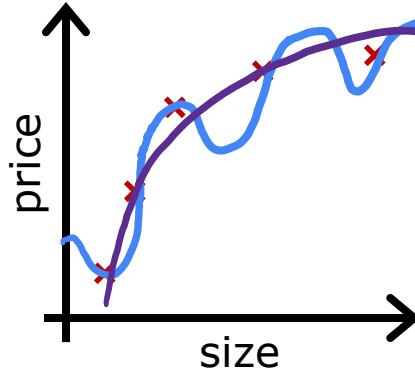
---

## Cost Function with Regularization

# Intuition



$$w_1x + w_2x^2 + b$$



$$w_1x + w_2x^2 + \underbrace{w_3x^3}_{\approx 0} + \underbrace{w_4x^4}_{\approx 0} + b$$

make  $w_3, w_4$  really small ( $\approx 0$ )

$$\min_{\vec{w}, b} \frac{1}{2m} \sum_{i=1}^m (f_{\vec{w}, b}(\vec{x}^{(i)}) - y^{(i)})^2 + 1000 \underbrace{w_3^2}_{0.001} + 1000 \underbrace{w_4^2}_{0.002}$$

# Regularization

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

simpler model

$$w_3 \approx 0$$

less likely to overfit

$$w_4 \approx 0$$

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

$n$  features       $n = 100$

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

$$J(\vec{w}, b) = \frac{1}{2m} \sum_{i=1}^m \left( f_{\vec{w}, b}(\vec{x}^{(i)}) - y^{(i)} \right)^2 + \underbrace{\frac{\lambda}{2m} \sum_{j=1}^n w_j^2}_{\text{"lambda"} \quad \text{regularization parameter}} + \underbrace{\frac{\lambda}{2m} b^2}_{\lambda > 0}$$

can include or exclude  $b$

# Regularization

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

fit data

regularization term

Keep  $w_j$  small

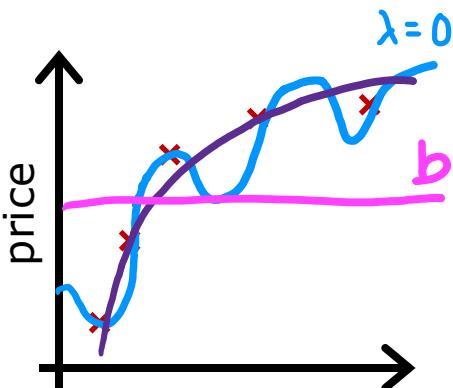
$\lambda$  balances both goals

choose  $\lambda = 10^{10}$

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

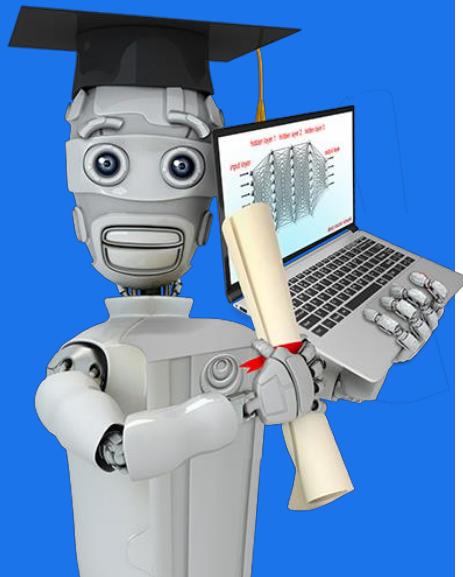
$$f(x) = b$$

Choose  $\lambda$



Stanford  
ONLINE

DeepLearning.AI



# Regularization to Reduce Overfitting

---

## Regularized Linear Regression

# Regularized linear regression

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

Gradient descent

repeat {

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

$j = 1, \dots, n$

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

} simultaneous update

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

don't have to regularize  $b$

# Implementing gradient descent

repeat {

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

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

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



# Implementing gradient descent

repeat {

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

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

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

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

shrink  $w_j$

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

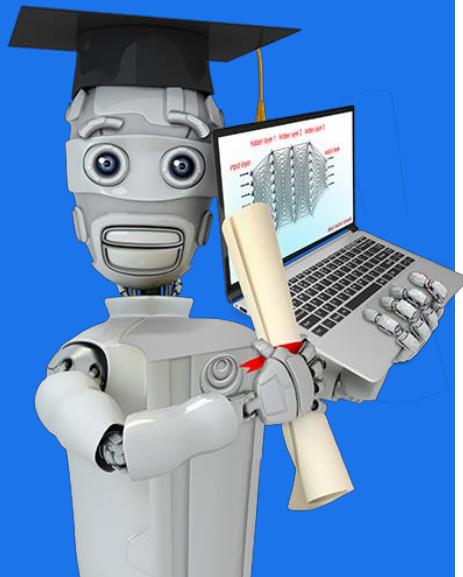


# How we get the derivative term (optional)

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

Stanford  
ONLINE

DeepLearning.AI

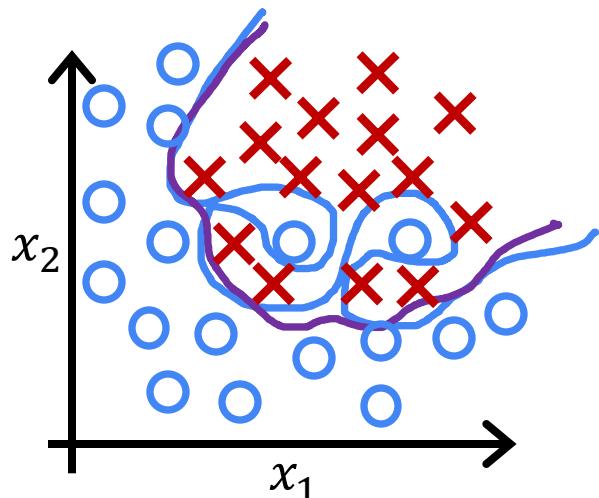


# Regularization to Reduce Overfitting

---

## Regularized Logistic Regression

# Regularized logistic regression



$$z = w_1x_1 + w_2x_2 + w_3x_1^2x_2 + w_4x_1^2x_2^2 + w_5x_1^2x_2^3 + \dots + b$$
$$f_{\vec{w}, b}(\vec{x}) = \frac{1}{1 + e^{-z}}$$

## Cost function

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

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

# Regularized logistic regression

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

Gradient descent

repeat {

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

j = 1 ... n

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

}

Looks same as  
for linear regression!

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

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

logistic regression

don't have to  
regularize b