

# Week 3

## Classification with Logistic Regression

Classification with Logistic Regression

Why not to use Linear Regression on Classification Problems

Logistic Regression

Calculating the logistic function

Decision Boundary

Linear decision boundary

Non-linear decision boundary

Cost Function for Logistic Regression

Using Cost function in Linear vs Logistic regression

Logistic Loss Function

Simplified Loss Function

Gradient Descent for Logistic Regression

Overfitting

Addressing overfitting

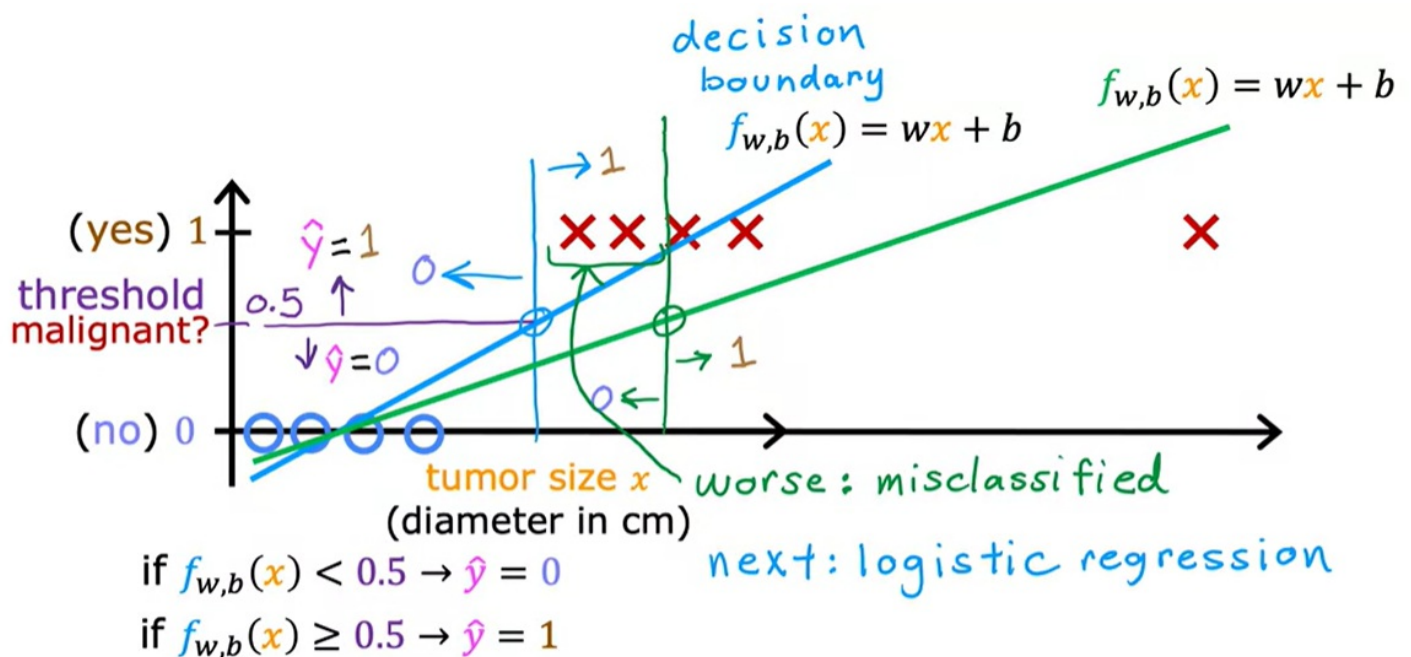
Regularization

Cost function with regularization

Regularization for Gradient Descent

### Why not to use Linear Regression on Classification Problems

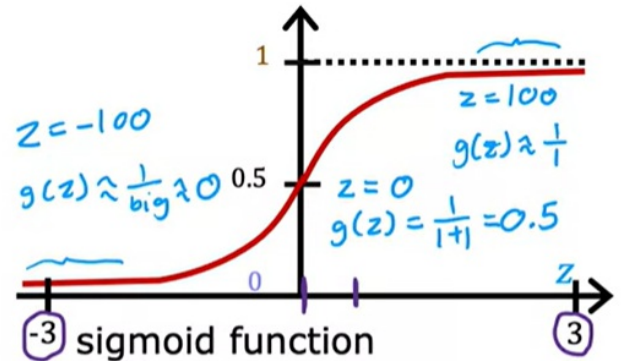
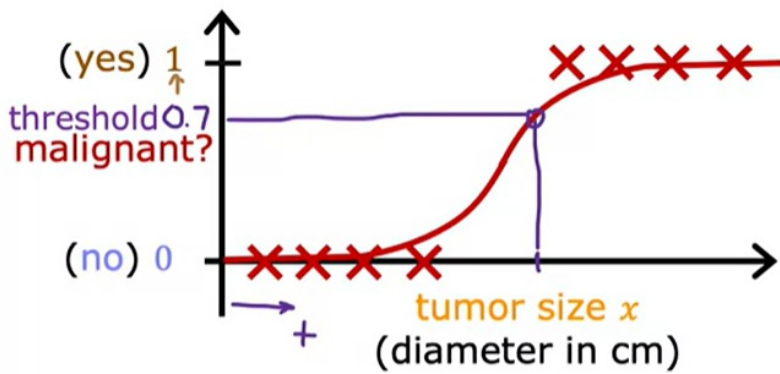
- As more data is added, the best fit line can change drastically along with the decision boundary threshold → bad as threshold shouldn't be shifting



### Logistic Regression

- Sigmoid function
- zzz is some constant and eee is euler's number

Want outputs between 0 and 1



logistic function

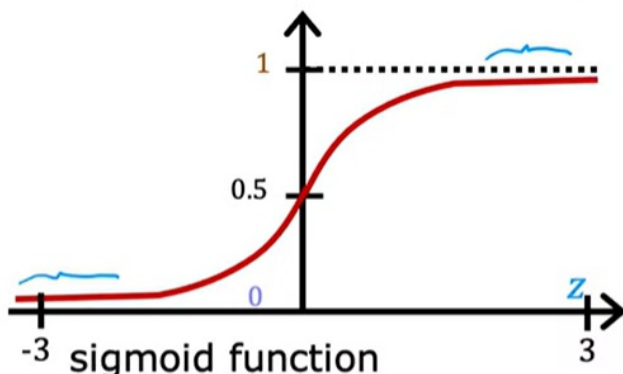
outputs between 0 and 1

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

### Calculating the logistic function

- Outputting the probability that the function will return 1 (probability of  $y = 1$ ).
- $0 \leq f(x) \leq 1$

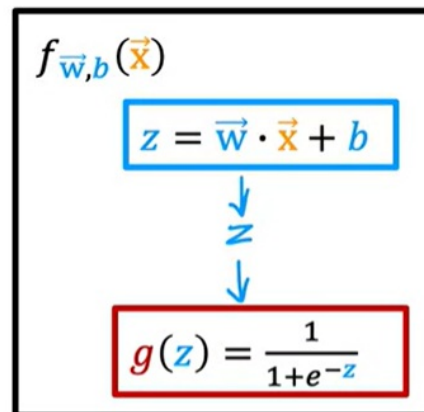
Want outputs between 0 and 1



logistic function

outputs between 0 and 1

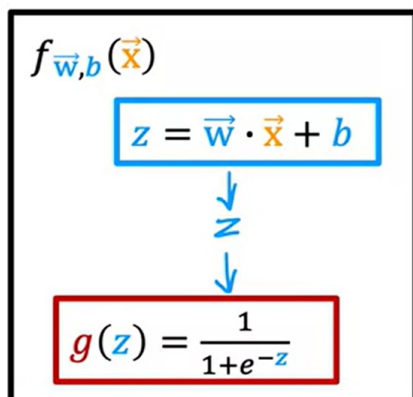
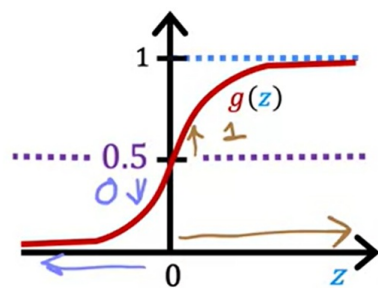
$$g(z) = \frac{1}{1+e^{-z}} \quad 0 < g(z) < 1$$



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

"logistic regression"

- If  $z$  is huge, then  $g(z)$  will be closer to 1, vice versa.



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

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

0 or 1? threshold

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

$$\text{Yes: } \hat{y} = 1$$

$$\text{No: } \hat{y} = 0$$

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

$$g(z) \geq 0.5$$

$$z \geq 0$$

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

$$\hat{y} = 1$$

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

$$\hat{y} = 0$$

## Decision Boundary

### Linear decision boundary

- Calculated by assuming  $w_1 = w_2 = 1$ , then solving for  $x_1$  and  $x_2$

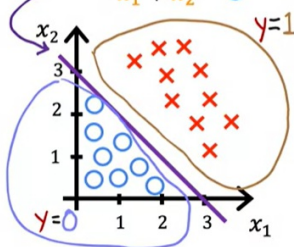
### Decision boundary

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

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

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

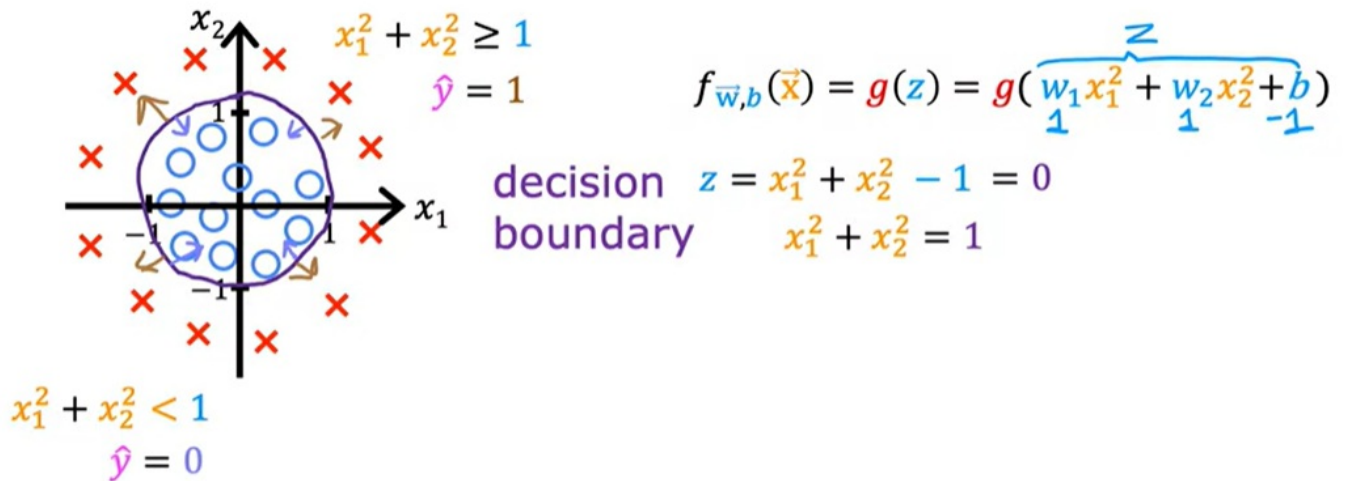
$$x_1 + x_2 = 3$$



### Non-linear decision boundary

- polynomial terms form the decision boundary

# Non-linear decision boundaries



→ low threshold → greater probability that y will be 1

## Cost Function for Logistic Regression

### Using Cost function in Linear vs Logistic regression

- Squared error cost function cannot be used for logistic regression as the cost function will be **stuck** at a local minima.
- Notice that 1/2 is inside the summation.

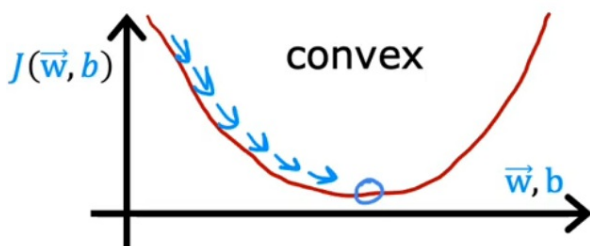
## Squared error cost

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

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

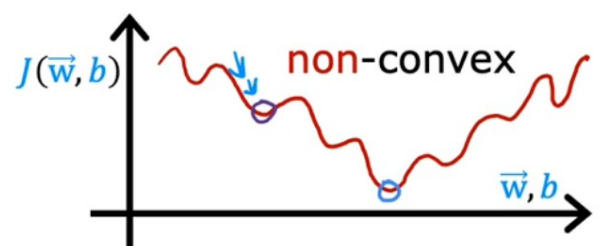
linear regression

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



logistic regression

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



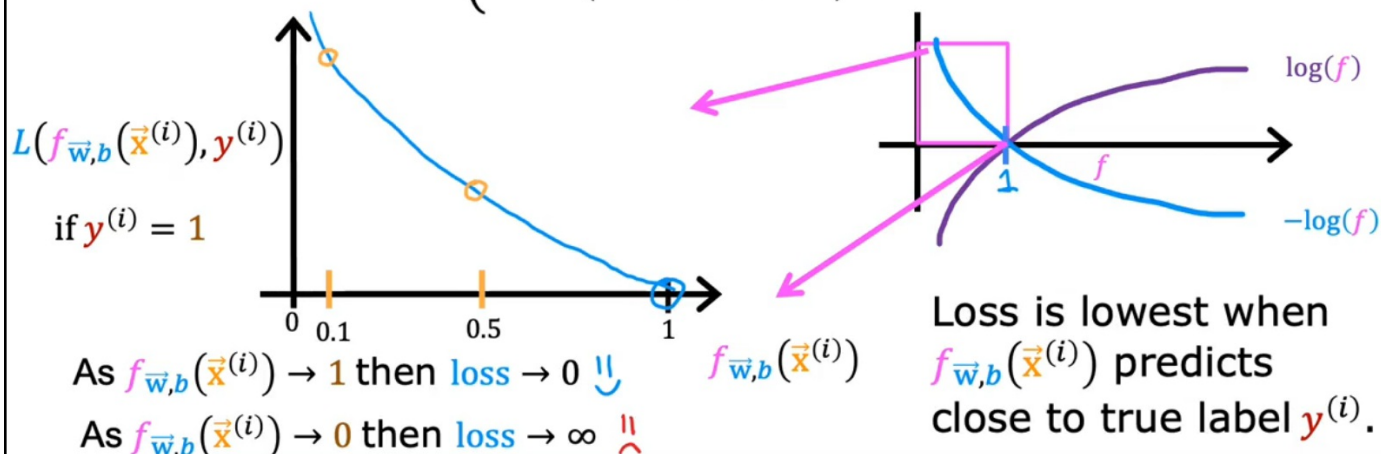
### Logistic Loss Function

- Reminder: The loss function measures the accuracy of one one training set as it measures the entire training set
- Model is penalised if prediction is 0.99 when it is supposed to be 0 → e.g Model predicts tumor to be 0.99 malignant but is actually benign
- y-axis: Loss
- x-axis: value of f (between 0 and 1 only)

When actual data is positive ( $y=1$ ):

## Logistic loss function

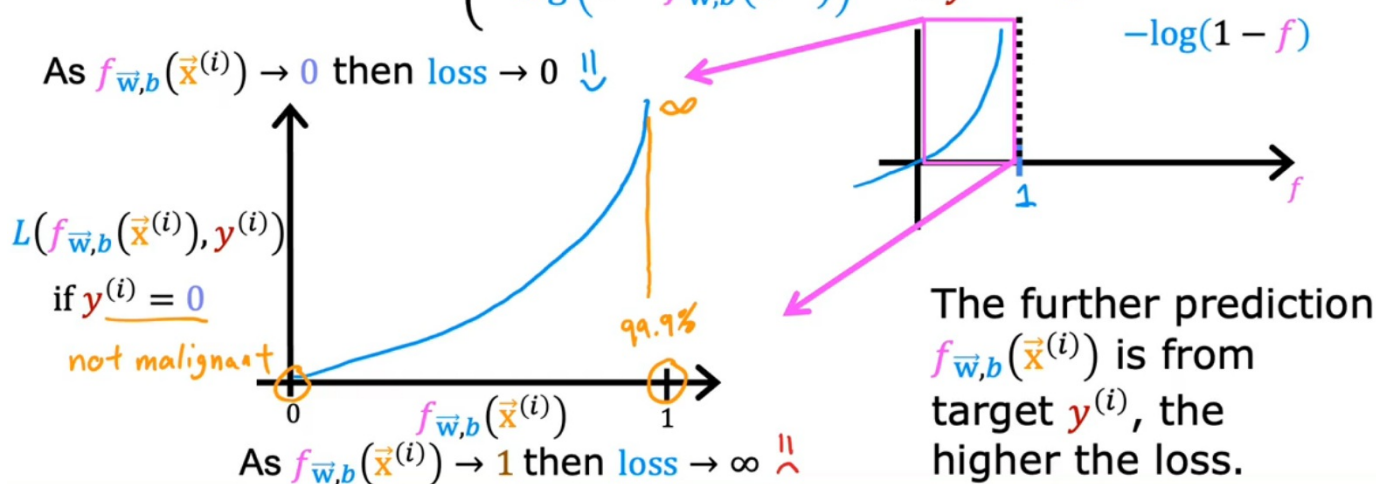
$$L(f_{\bar{w},b}(\vec{x}^{(i)}), y^{(i)}) = \begin{cases} -\log(f_{\bar{w},b}(\vec{x}^{(i)})) & \text{if } y^{(i)} = 1 \\ -\log(1 - f_{\bar{w},b}(\vec{x}^{(i)})) & \text{if } y^{(i)} = 0 \end{cases}$$



When actual data is negative ( $y=0$ ):

## Logistic loss function

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



### Simplified Loss Function

- Cases of  $y_i=0 \Rightarrow y_i=0$  and  $y_i=1 \Rightarrow y_i=1$



## Simplified loss function

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

$$L(f_{\vec{w},b}(\vec{x}^{(i)}), y^{(i)}) = -\underbrace{y^{(i)}}_0 \log(\underbrace{f_{\vec{w},b}(\vec{x}^{(i)})}_{(1-0)}) - (1 - y^{(i)}) \log(1 - f_{\vec{w},b}(\vec{x}^{(i)}))$$

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

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

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

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

- **Reminder: cost function measures the average loss across datasets**
- Cost function rewritten using the simplified loss function formula:

## Simplified cost function

$$\text{loss } L(f_{\vec{w},b}(\vec{x}^{(i)}), y^{(i)}) = -\underbrace{y^{(i)} \log(f_{\vec{w},b}(\vec{x}^{(i)})) + (1 - y^{(i)}) \log(1 - f_{\vec{w},b}(\vec{x}^{(i)}))}_{\text{convex (single global minimum)}}$$

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

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

maximum likelihood  
(don't worry about it!)

## Gradient Descent for Logistic Regression

- Same concept as Linear Regression
- Simultaneously updates  $w$  and  $b$  to minimize the cost
- Feature scaling is applied the same way

# 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

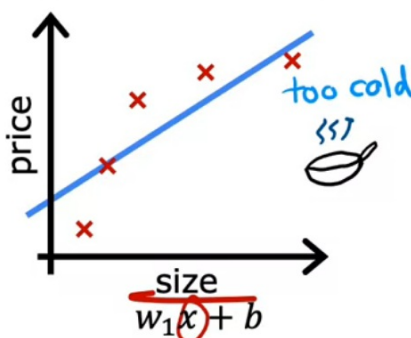
Linear regression  $f_{\vec{w},b}(\vec{x}) = \vec{w} \cdot \vec{x} + b$

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

## Overfitting

- Underfitting → Too few features → Low variance, high bias
- Overfitting → Too many features, model tries too hard to fit all the data and result in a weird curve that leads to inaccurate predictions → Model does not generalize well to new examples

## Regression example



*underfit*

- Does not fit the training set well

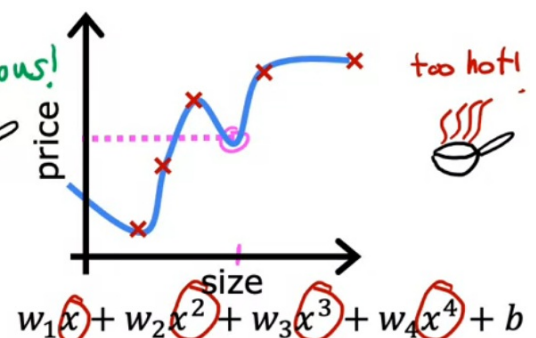
*high bias*



*just right*

- Fits training set pretty well

*generalization*

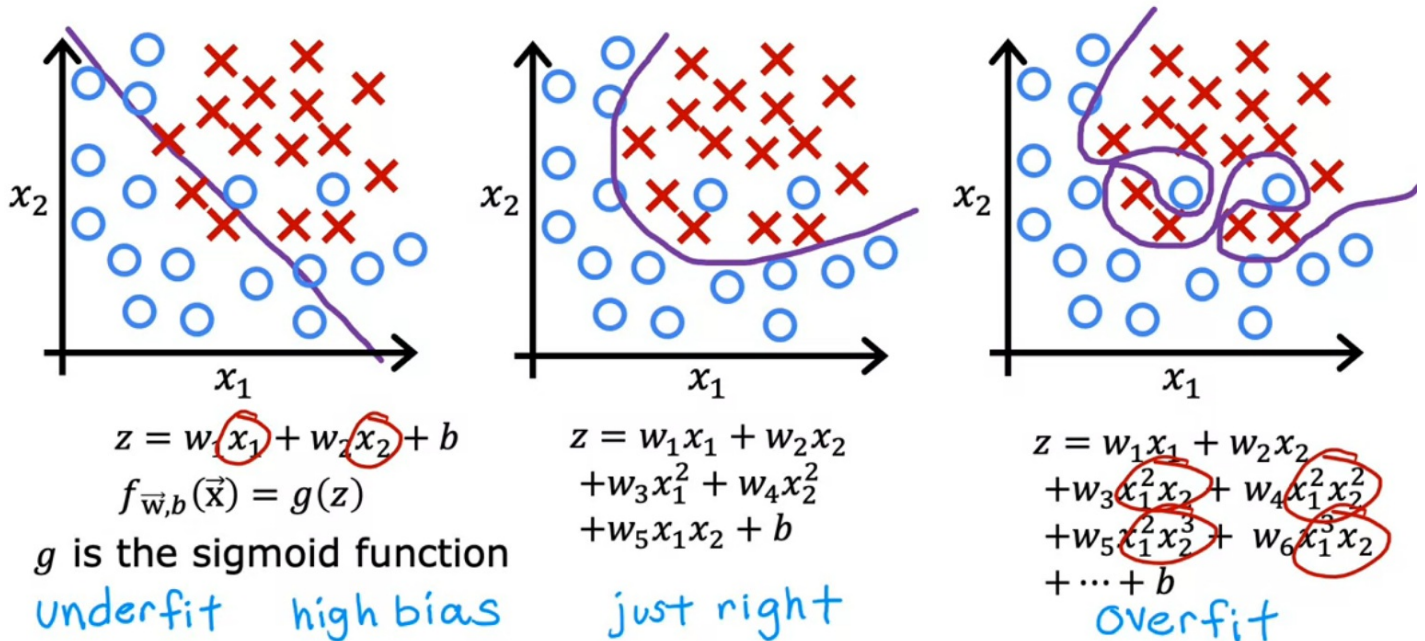


*overfit*

- Fits the training set extremely well

*high variance*

# Classification



## Addressing overfitting

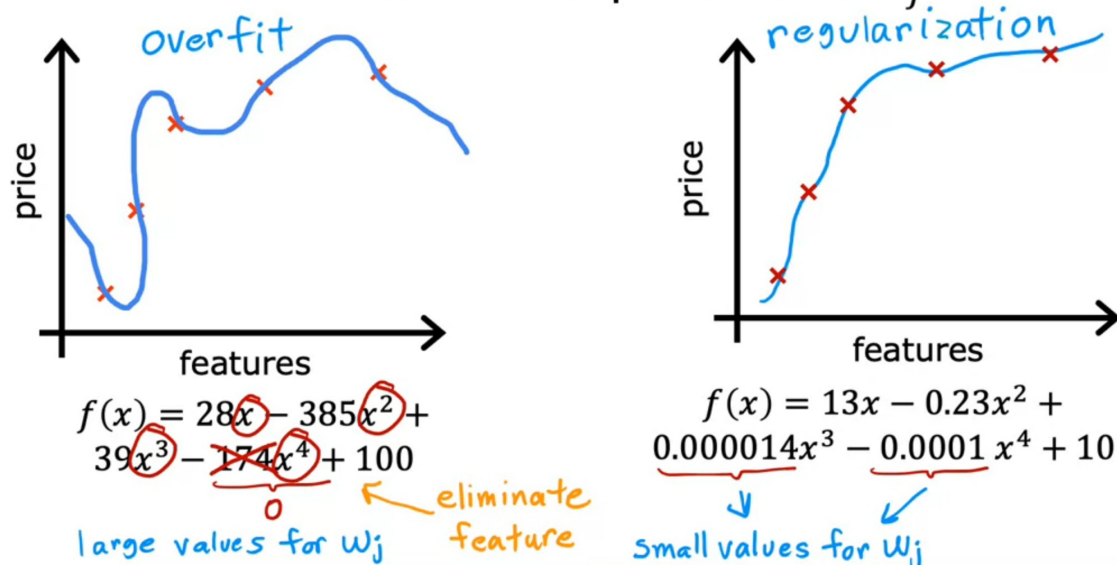
- Collect more training data
- Select features to include/exclude (Maybe exclude some features) → Pick the most appropriate features to use especially when you don't have much data. However, useful features could potentially be lost.

## Regularization

- Reduce size of parameters
- By convention, regularize  $w$  only and not  $b$

# Regularization

Reduce the size of parameters  $w_j$

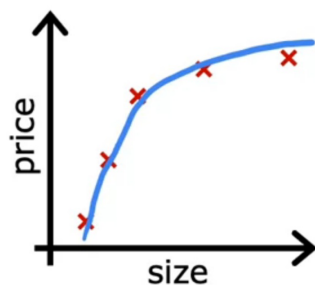


## Cost function with regularization

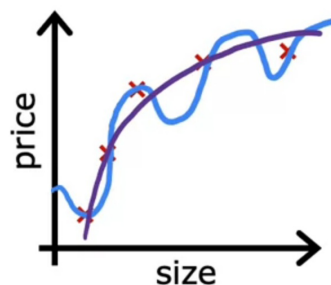
- Idea: Penalize the model for having a huge  $w$  value by multiplying a large number and adding it outside of the cost function → Prevent overfitting



# 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 ( $\approx 0$ )

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

- Reminder: By convention,  $b$  is not regularized
- Lambda = regularization parameter
- Choose lambda appropriately
- Penalize on every  $w$  parameter

## Regularization

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

simpler model

less likely to overfit

$$w_3 \approx 0$$

$$w_4 \approx 0$$

size	bedrooms	floors	age	avg income	...	distance to coffee shop	price
$x_1$	$x_2$	$x_3$	$x_4$	$x_5$		$x_{100}$	$y$
$w_1, w_1, w_2, \dots, w_{100}, b$							
$n$ features							
$n = 100$							

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

regularization term

$\lambda > 0$

## Regularization for Gradient Descent

- For both linear and logistic regression

# 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  $j=1 \dots n$

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

$$\alpha \frac{\lambda}{m} = 0.01 \frac{1}{50} = 0.0002$$

$$w_j \frac{(1 - 0.0002)}{0.9998}$$