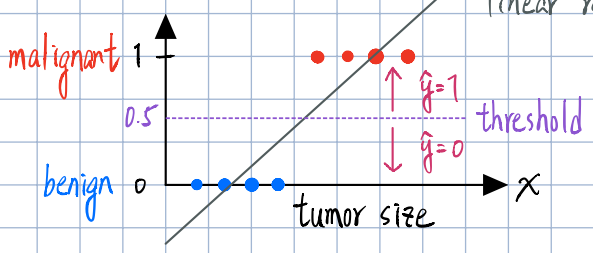
Classification with Logistic Regression

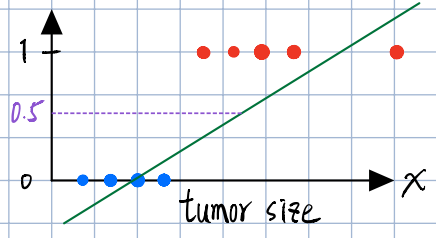
Motivation: binary classification

Decision Boundry

linear regression $f(x) = wx + b$



Using threshold here makes sense. But, there's a new malignant sample as shown below.

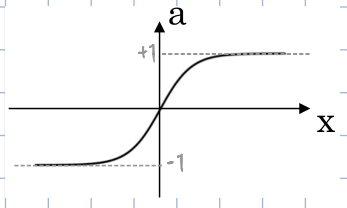


$f(x)$ The new line destroys the threshold approach, making it hard to determine the threshold value.

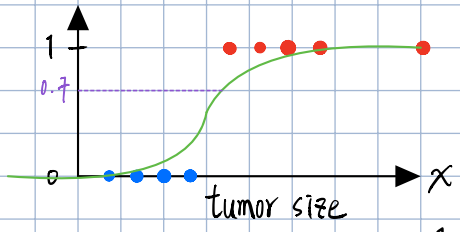
This is why linear regression doesn't work on classification problems. We need Logistic Regression.

Hyperbolic tangent:

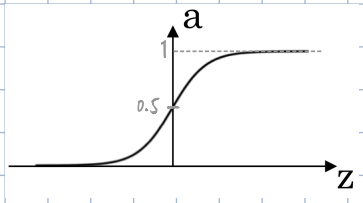
$$a = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$



Logistic Regression



$$f(x) = g(w \cdot x + b) = g(z) = \frac{1}{1 + e^{-z}}$$



Sigmoid/Logistic function:

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

Interpretation of logistic regression output:

$f_{w,b}(x) = g(wx + b)$ → Probability that class is 1. $P(y=0) + P(y=1) = 1$

E.g.: x is tumor size
y is 0 benign
is 1 malignant

$f(x) = 0.7$. 70% chance that y is 1.

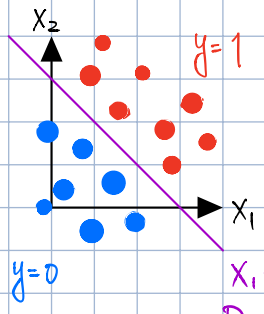
Notation: $f(x) = P(y=1 \mid x; w, b)$

The probability that y is 1 given x, parameterized by w, b.

Decision Boundry

Given $f(x) = g(z) = \frac{1}{1+e^{-z}} = P(y=1|x;w,b)$

$\hat{y} = 1 \rightarrow f(x) \geq 0.5 \rightarrow g(z) \geq 0.5 \rightarrow z \geq 0 \rightarrow \boxed{wx+b \geq 0}$ Decision Boundry



$f(x) = g(z) = g(w_1 x_1 + w_2 x_2 + b)$
$w_1 = 1, \quad w_2 = 1, \quad b = -3$

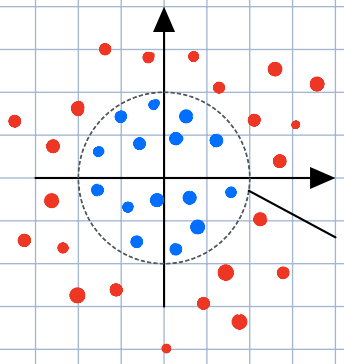
$z = \vec{w} \cdot \vec{x} + b = 0$

$z = x_1 + x_2 - 3 = 0$

$x_1 + x_2 = 3$

Decision Boundry

nonlinear decision boundry



$f(x) = g(z) = g(w_1 x_1^2 + w_2 x_2^2 + b)$
$w_1 = 1, \quad w_2 = 1, \quad b = -2$

$z = x_1^2 + x_2^2 - 2 = 0$

Cost Function for Logistic Regression

Recall Squared Error Cost

Given m training samples ($i = 1, \dots, m$), and each sample has n features ($j = 1, \dots, n$)

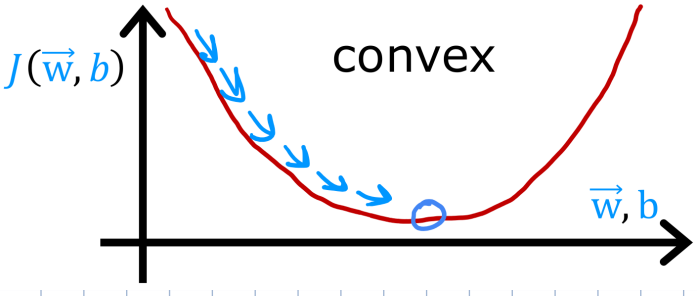
target y is 0 or 1. $f(x^{(i)}) = g(z) = \frac{1}{1+e^{-z}}$. $0 < f(x^{(i)}) < 1$

$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m} \boxed{\frac{1}{2}(f(x^{(i)}) - y^{(i)})^2} \longrightarrow \text{Loss } L(f(x^{(i)}), y^{(i)})$$

The two images below show that why squared error cost function is not a good choise for logistic regression.

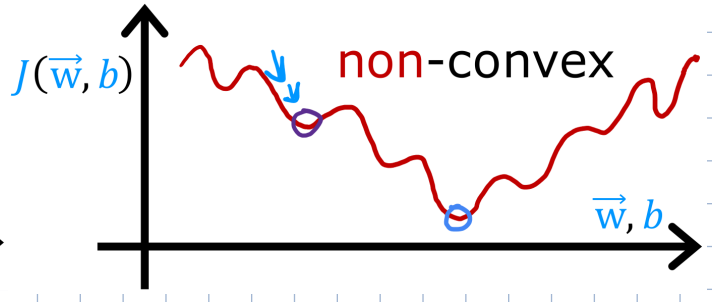
linear regression

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



logistic regression

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



Instead, a good choice of cost function for logistic regression is:

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

Try to explain why the loss makes sense here.

The loss function can be simplified as follow:

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

Convex, single global minimum

Thus, the cost $J(w,b) = \frac{1}{m}\sum_{i=1}^{m} L(f_{w,b}(x^{(i)}), y^{(i)})$

$$= \frac{-1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log(f(x^{(i)})) + (1 - y^{(i)})\log(1 - f(x^{(i)}))\right]$$

Gradient Descent for Logistic Regression

Find W, b.

Given new $\vec{x}$, output $f(\vec{x}) = \frac{1}{1+e^{-(wx+b)}} \cdot P(y=1|X;w,b)$

Gradient Descent

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

repeat until convergence {

$j = 1,\dots,n$

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

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

}

repeat until convergence {

$j = 1,\dots,n$

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

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

}

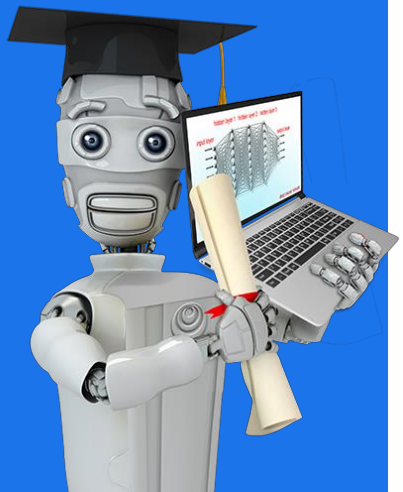
As a sidenote, in programming,

The name of the cost function for classification with exactly two classes is usually refer as Binary Cross Entropy.

The name of the cost function for regression is usually refer as Mean Squared Error.

Stanford
ONLINE

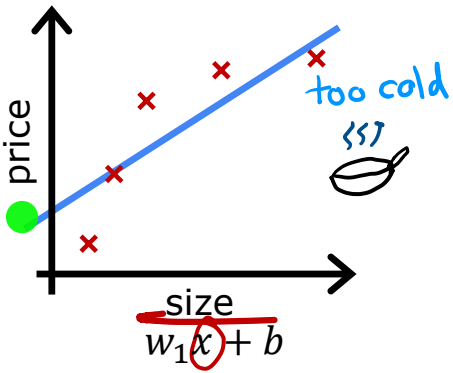
DeepLearning.AI



Regularization to Reduce Overfitting

The Problem of Overfitting

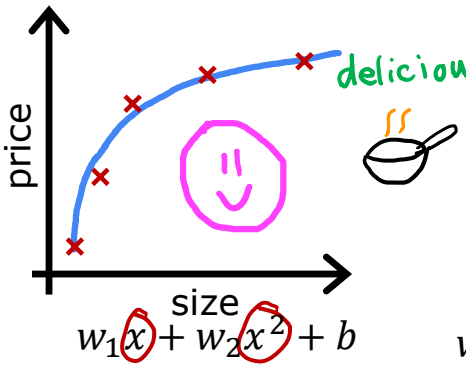
Regression example



underfit

- Does not fit the training set well

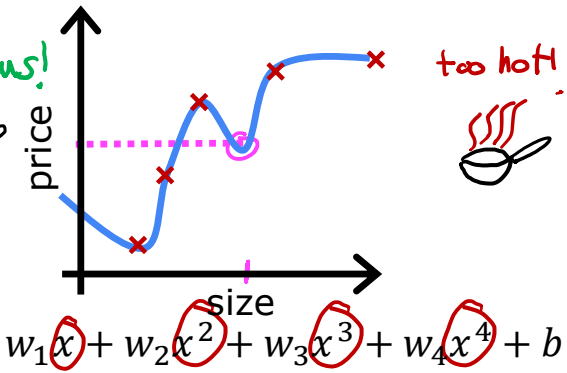
high bias



just right

- Fits training set pretty well

generalization

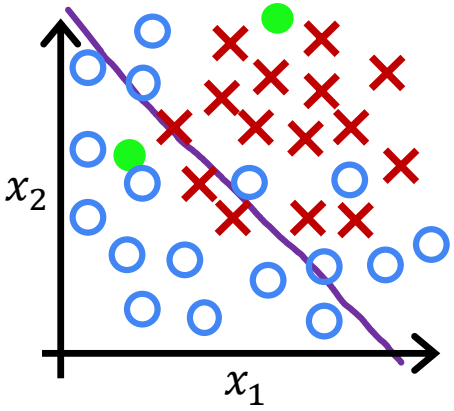


overfit

- Fits the training set extremely well

high variance

Classification

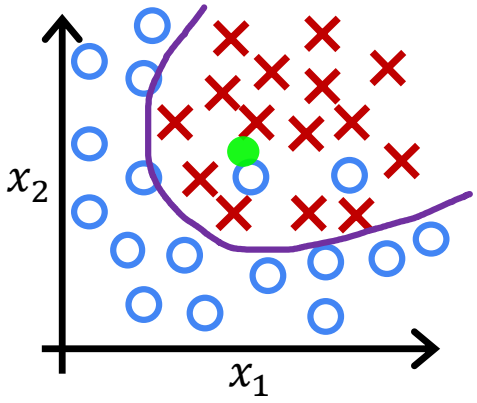


$$z = w_1 x_1 + w_2 x_2 + b$$

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

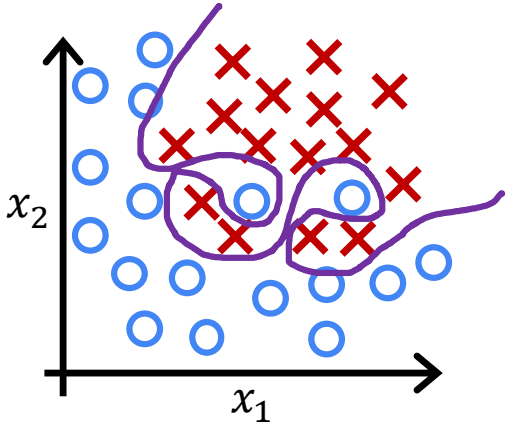
g is the sigmoid function

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

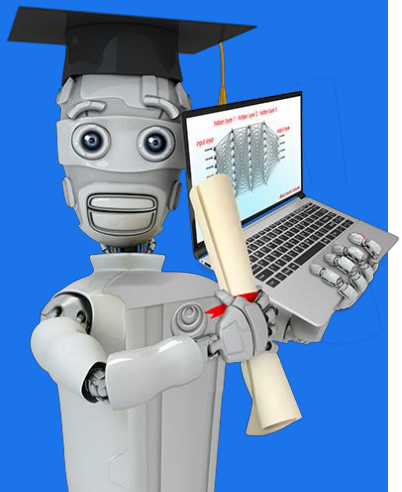


$$z = w_1 x_1 + w_2 x_2 + w_3 x_1^2 x_2 + w_4 x_1^2 x_2^2 + w_5 x_1^2 x_2^3 + w_6 x_1^3 x_2 + \cdots + b$$

overfit

Stanford
ONLINE

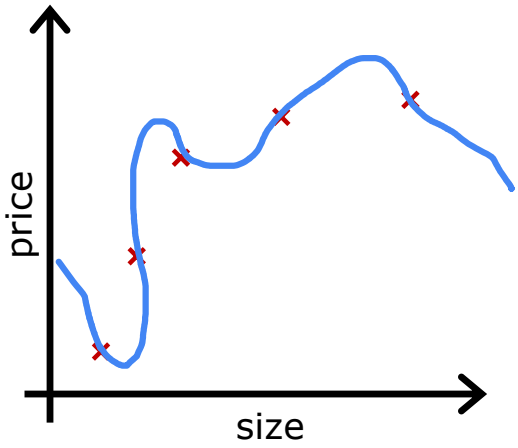
DeepLearning.AI

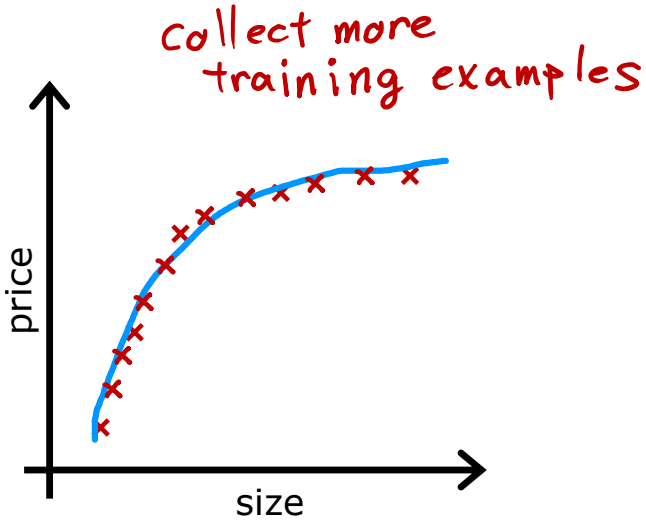


Regularization to Reduce Overfitting

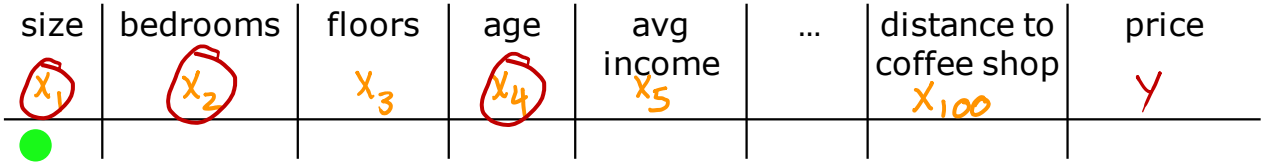
Addressing Overfitting

Collect more training examples





Select features to include/exclude



all features

\+

insufficient data

↓

overfit

selected features

size
bedrooms
age
just right

feature selection

course 2

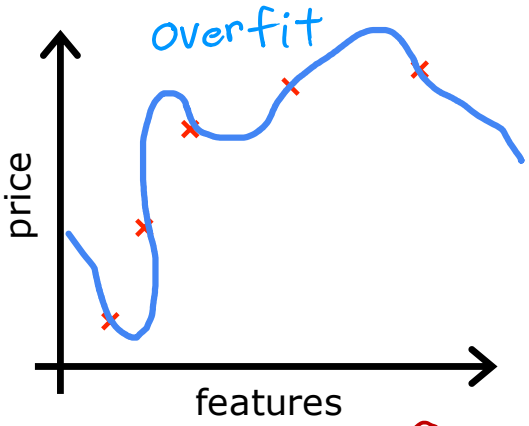
disadvantage

↓

useful features could be lost

Regularization

Reduce the size of parameters w_j

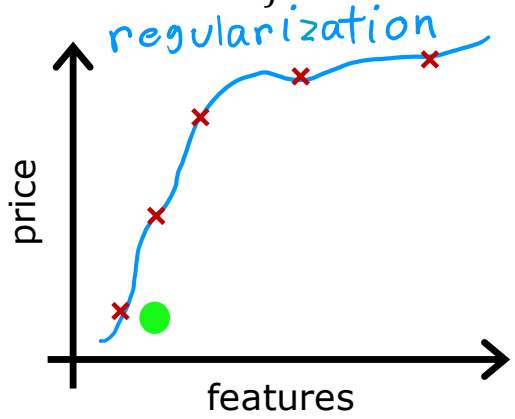


$f(x) = 28x - 385x^2 + 39x^3 - 174x^4 + 100$

large values for w_j

0

eliminate feature



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

small values for w_j

Addressing overfitting

Options

1. Collect more data

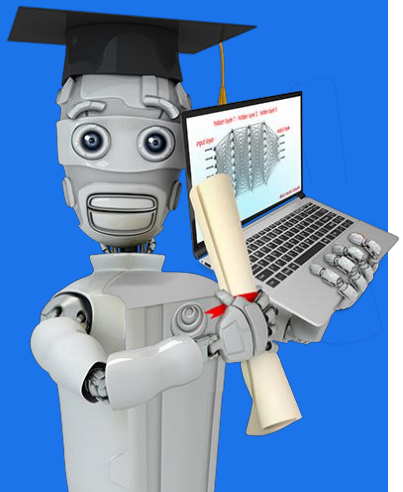
2. Select features
   - Feature selection in course 2

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

Andrew Ng uses regularization all the time.

Stanford
ONLINE

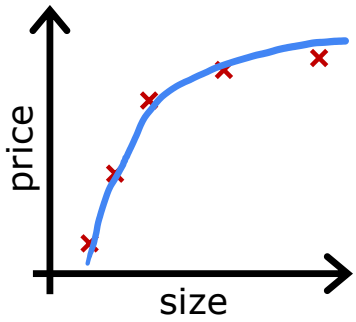
DeepLearning.AI



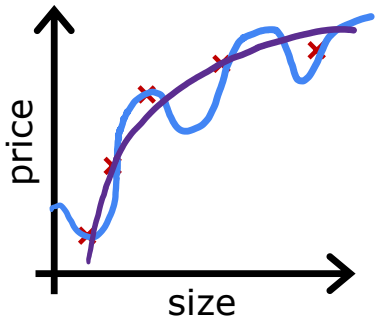
Regularization to Reduce Overfitting

Cost Function with Regularization

Intuition



$w_1 x + w_2 x^2 + b$



$w_1 x + w_2 x^2 + w_3 x^3 + w_4 x^4 + b$

$w_3 \approx 0$ $w_4 \approx 0$

make w_3, w_4 really small (≈ 0)

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

$w_3 \approx 0.001$ $w_4 \approx 0.002$

It penalizes the model for w_3 and w_4 being large.

Regularization

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

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

$n = 100$

n features

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

Penalize all n features $w_1, \ldots, w_n$. regularization term

Andrew doesn't regularize b though.

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

can include or exclude b

"lambda" regularization parameter

$\lambda > 0$

Regularization

mean squared error

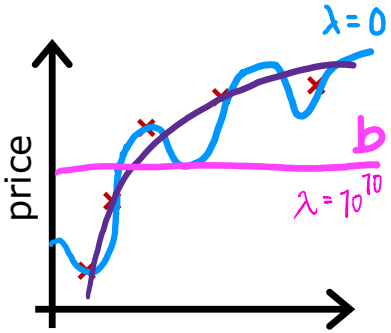
regularization term

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

fit data

Keep w_j small

λ balances both goals



choose $\lambda = 10^{10}$

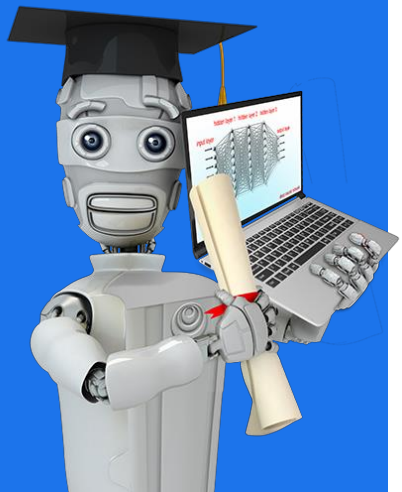
$$f_{\vec{w},b}(\vec{x}) = \underset{\approx 0}{w_1 x} + \underset{\approx 0}{w_2 x^2} + \underset{\approx 0}{w_3 x^3} + \underset{\approx 0}{w_4 x^4} + b$$

$f(x) = b$

Choose λ wisely

Stanford
ONLINE

DeepLearning.AI



Regularization to Reduce Overfitting

Regularized Linear Regression

Regularized linear regression

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

Gradient descent

repeat {

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

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

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

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

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 $j = 1 \ldots n$

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 $j = 1 \ldots n$

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

☆ shrink w_j

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

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

$w_j (1 - 0.0002)$

☆ 0.9998

How we get the derivative term (optional)

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

where $f(\vec{x}^{(i)}) = \vec{w}\cdot\vec{x}^{(i)} + b$

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

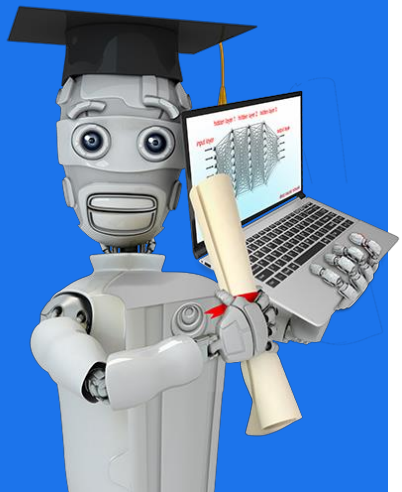
$$= \frac{1}{m}\sum_{i=1}^{m}\left[\left(\vec{w}\cdot\vec{x}^{(i)} + b - y^{(i)}\right) x_j^{(i)}\right] + \frac{\lambda}{m} w_j$$

where $f(\vec{x}) = \vec{w}\cdot\vec{x}^{(i)} + b$

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

Stanford
ONLINE

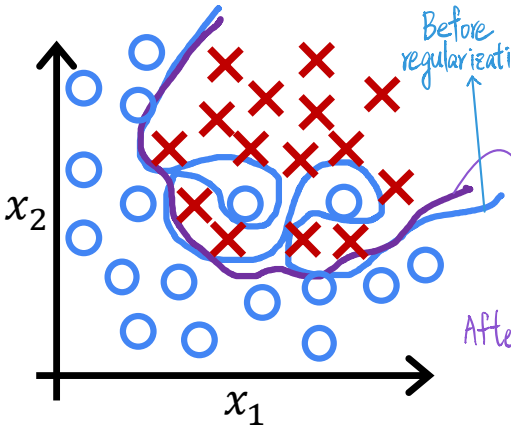
DeepLearning.AI



Regularization to Reduce Overfitting

Regularized Logistic Regression

Regularized logistic regression



Before regularization

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

high order polynomial function

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

After regularization

Cost function

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

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

Regularized logistic regression

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

$\min_{\vec{w},b}$

Gradient descent

repeat {

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

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

}

Looks same as for linear regression!

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

logistic regression

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

don't have to regularize b