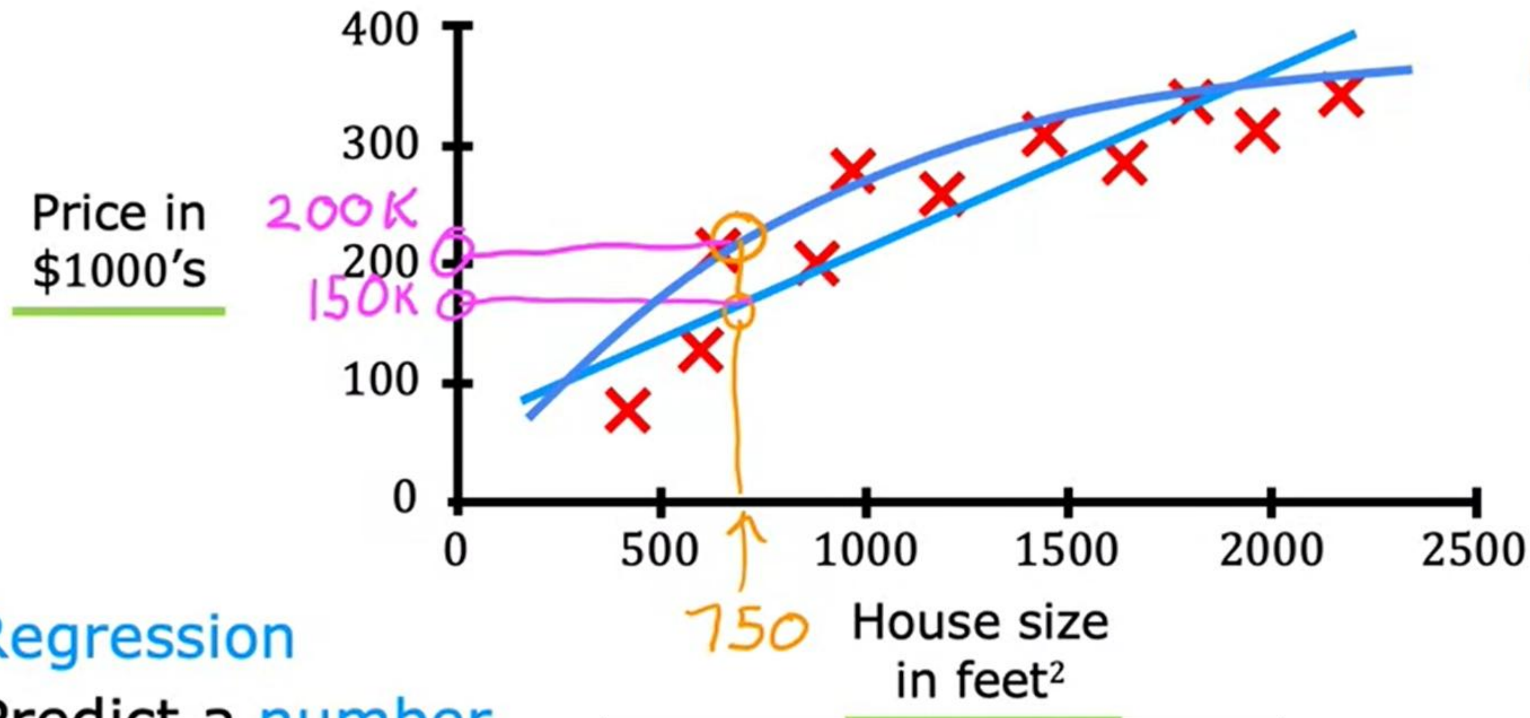
Regression: Housing price prediction



150,000
70,000
183,0

Regression

Predict a **number**

infinitely many possible outputs

70,000 or 183,000 or any other number in between.

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 {

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

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

}

except that it doesn't have
that xi term at the end.

(Optional)

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

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

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

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

Now you have these two
expressions for the derivatives.

Parameters and features

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

b is a number

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

linear algebra: count from 1

NumPy

w[0] w[1] w[2]

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

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

code: count from 0

Without vectorization

$$f_{\vec{w},b}(\vec{x}) = \left(\sum_{j=1}^{n} w_j x_j\right) + b \qquad \sum_{j=1}^{n} \rightarrow j=1\ldots n \quad 1,2,3$$

range(0,n) → j = 0...n-1

range(n)

```
f = 0
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
```

But is fp equals np
dot dot w comma x and

Gradient descent

One feature

repeat {

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

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

simultaneously update w, b

}

different with multiple features

Feature and parameter values

$\widehat{price} = \underset{\text{size}}{w_1 x_1} + \underset{\#\text{bedrooms}}{w_2 x_2} + b$

x_1: size (feet^2)
range: 300 – 2,000
large

x_2: # bedrooms
range: 0 – 5
small

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

size of the parameters w_1, w_2?

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

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

$\widehat{price} = \$100{,}050.5k$ =

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

$\widehat{price} = \underbrace{0.1 * 2000k}_{200K} + \underbrace{50 * 5}_{250K} + \underbrace{50}_{50K}$

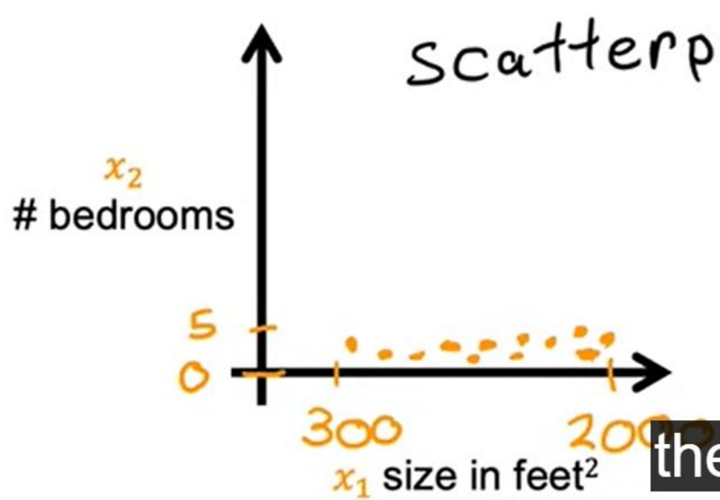
k more reasonable

Likewise, when the possible
values of the feature are small,

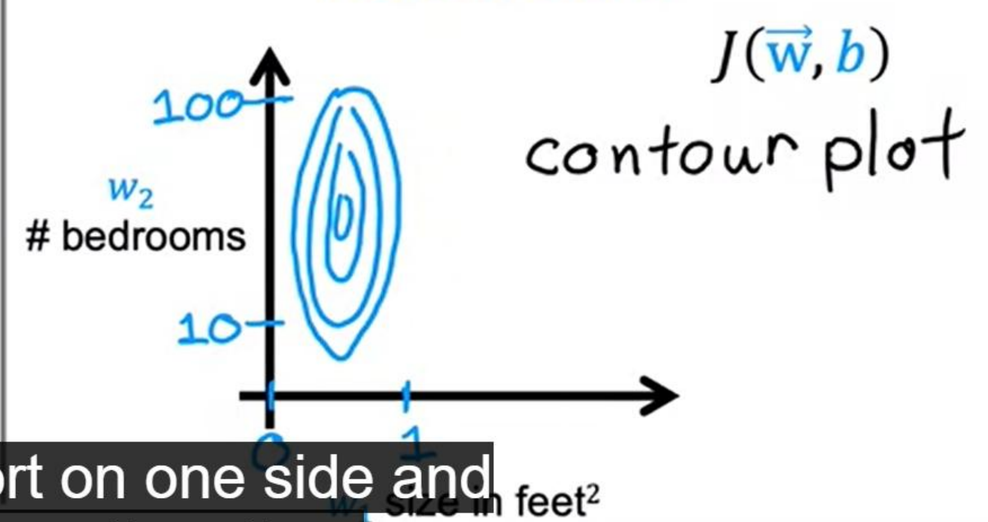
Feature size and parameter size

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

Features

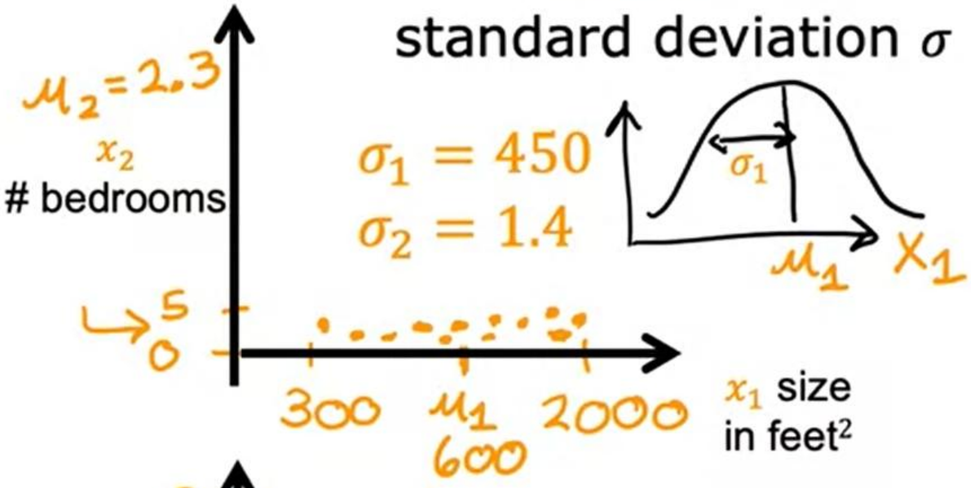


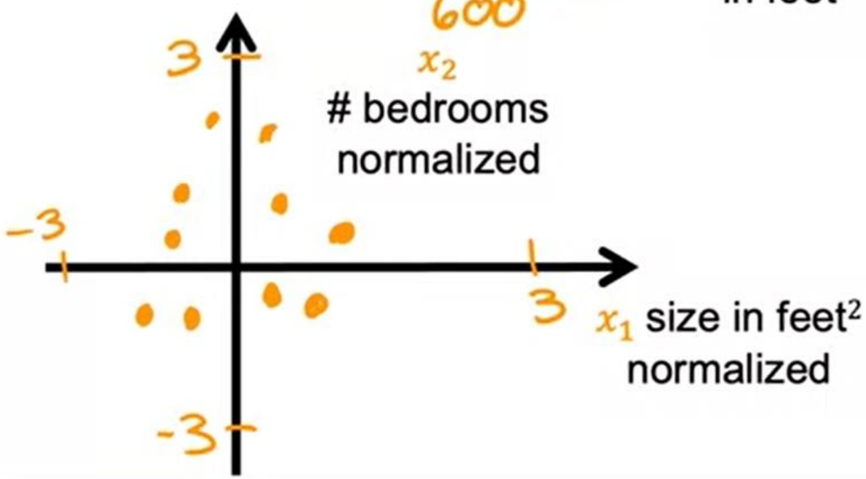
Parameters



they're short on one side and
longer on the other.

Z-score normalization





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

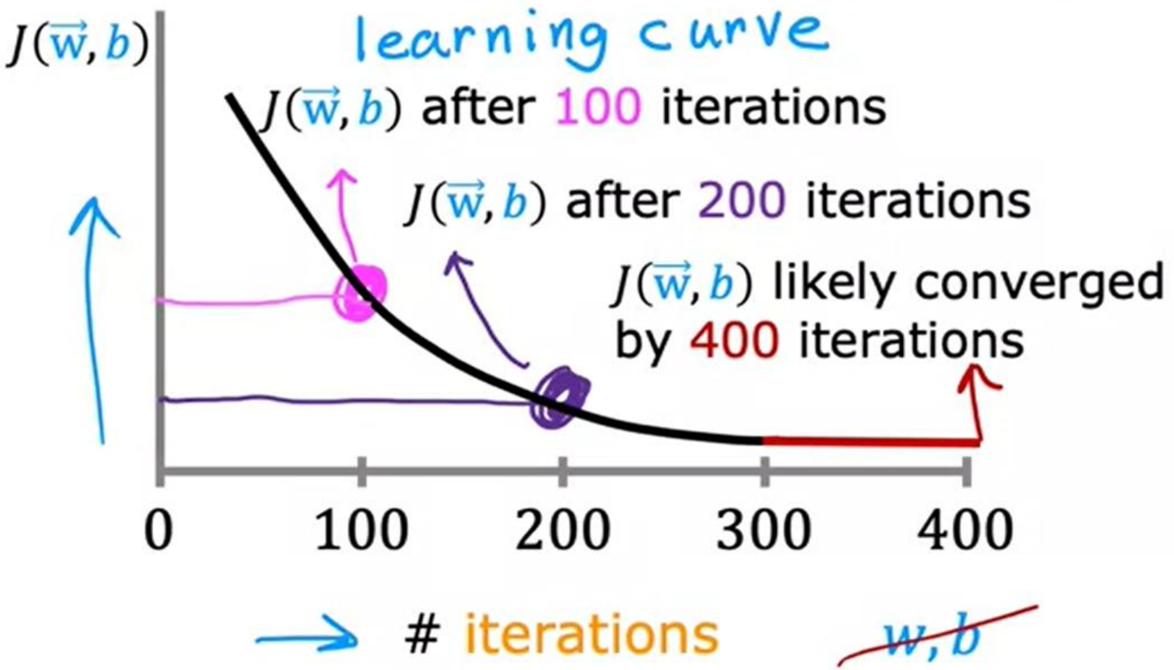
$$x_1 = \frac{x_1 - \mu_1}{\sigma_1} \qquad x_2 = \frac{x_2 - \mu_2}{\sigma_2}$$

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

Make sure gradient descent is working correctly

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

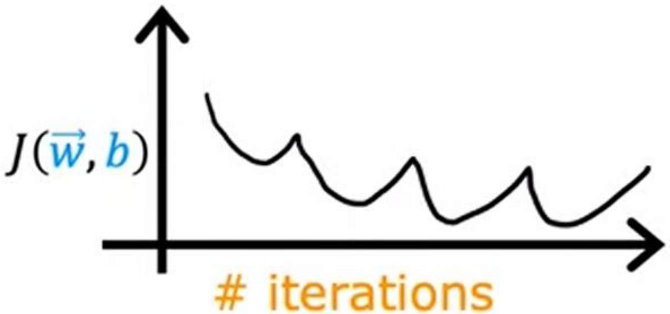
$J(\vec{w},b)$ should decrease after every iteration

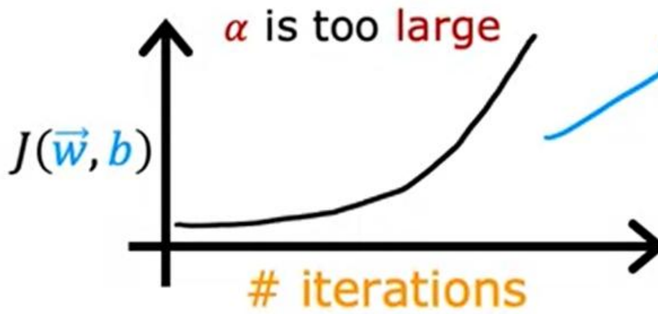


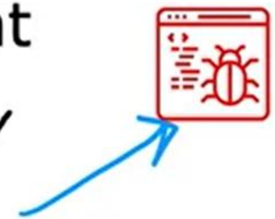
iterations needed varies

In one application, it may

Identify problem with gradient descent







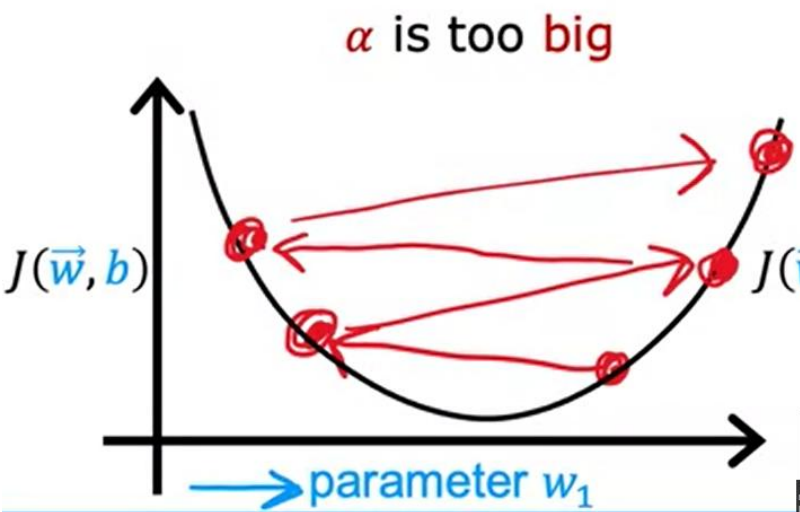
or learning rate is too large

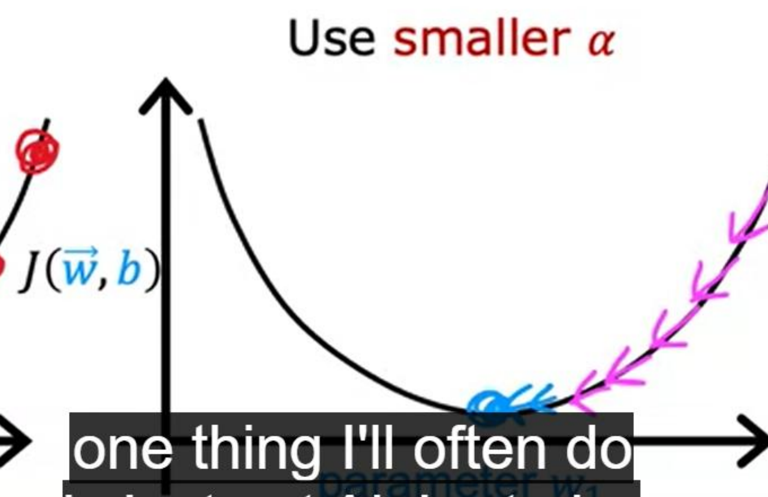
$w_1 = w_1 + \alpha d_1$

use a minus sign

$w_1 = w_1 - \alpha d_1$

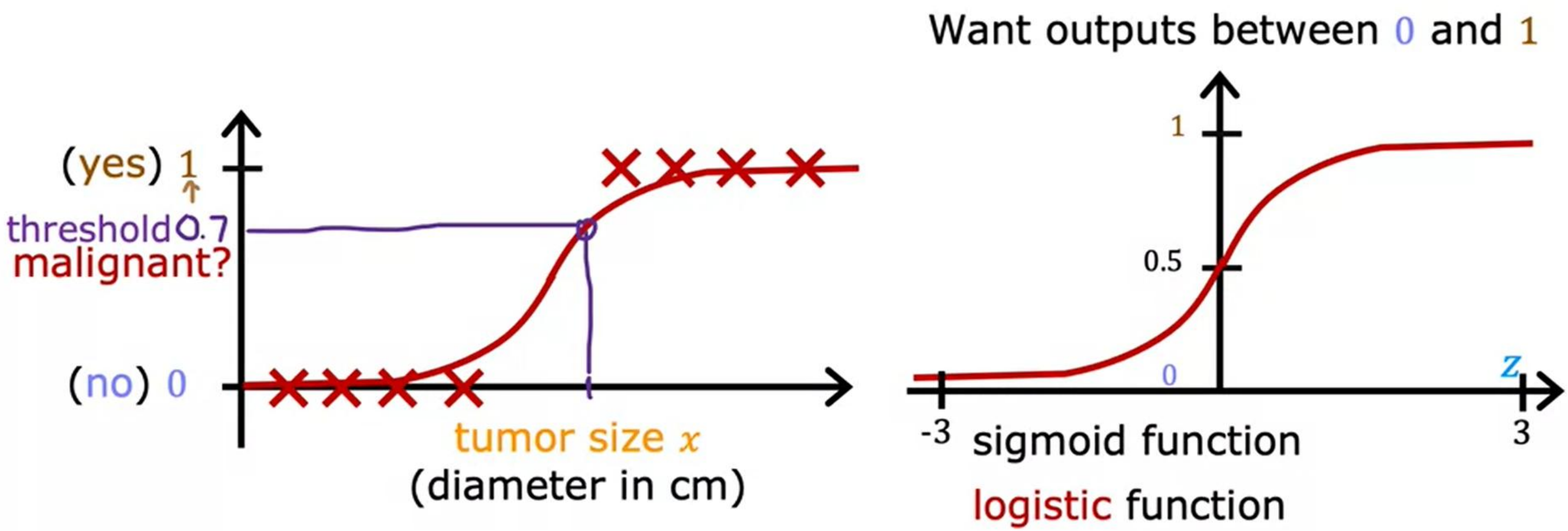
Adjust learning rate





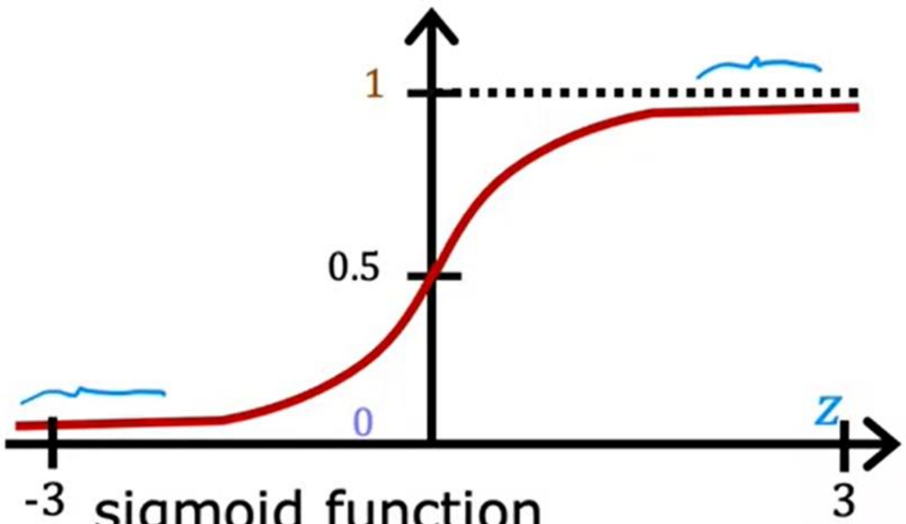
With a small enough α, $J(\vec{w}, b)$ should decrease on every iteration

one thing I'll often do
is just set Alpha to be



the graph on the left
and right are different.

Want outputs between 0 and 1



sigmoid function

logistic function

$$\boxed{\begin{array}{l}\text{outputs between 0 and 1}\\ g(z) = \frac{1}{1+e^{-z}} \qquad 0 < g(z) < 1\end{array}}$$

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

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

$$\downarrow$$

$$z$$

$$\downarrow$$

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

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

"logistic regression"

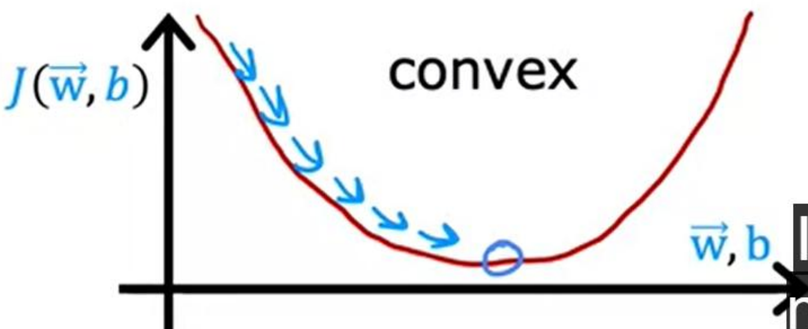
and what it does is it
inputs feature or set

Squared error cost

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

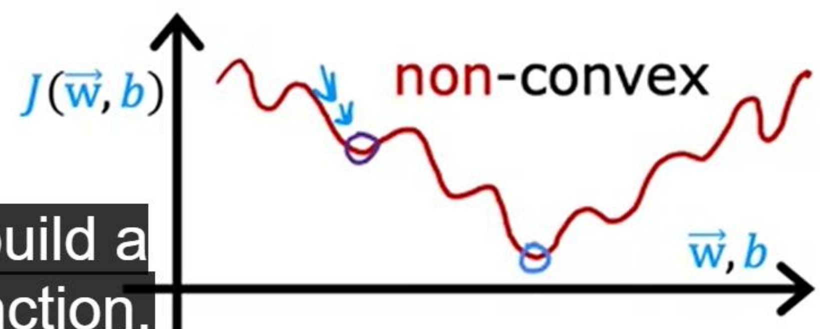
linear regression

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



logistic regression

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



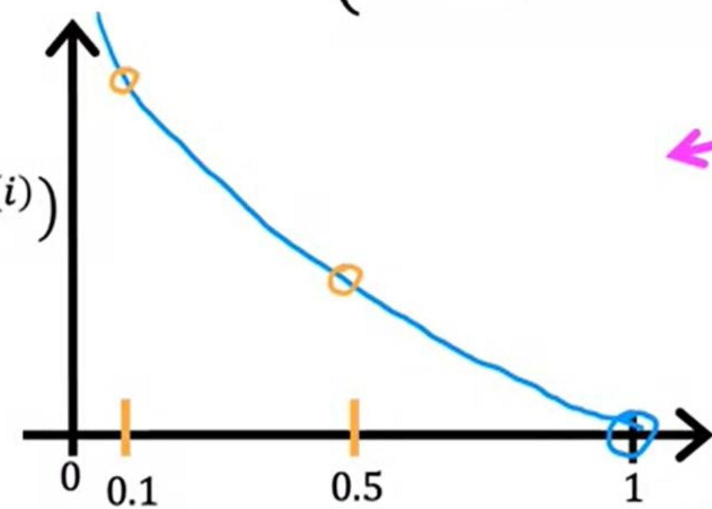
In order to build a
new cost function,

Logistic loss function

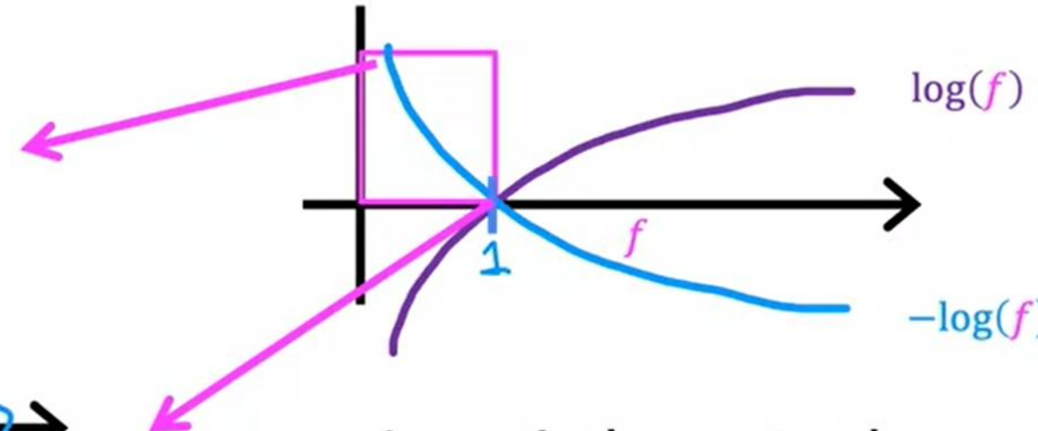
$$L\left(f_{\vec{\mathrm{w}},b}\left(\vec{\mathrm{x}}^{(i)}\right), y^{(i)}\right) = \begin{cases} -\log\left(f_{\vec{\mathrm{w}},b}\left(\vec{\mathrm{x}}^{(i)}\right)\right) & \text{if } y^{(i)} = 1 \\ -\log\left(1 - f_{\vec{\mathrm{w}},b}\left(\vec{\mathrm{x}}^{(i)}\right)\right) & \text{if } y^{(i)}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

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

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



$f_{\vec{\mathrm{w}},b}\left(\vec{\mathrm{x}}^{(i)}\right)$



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

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

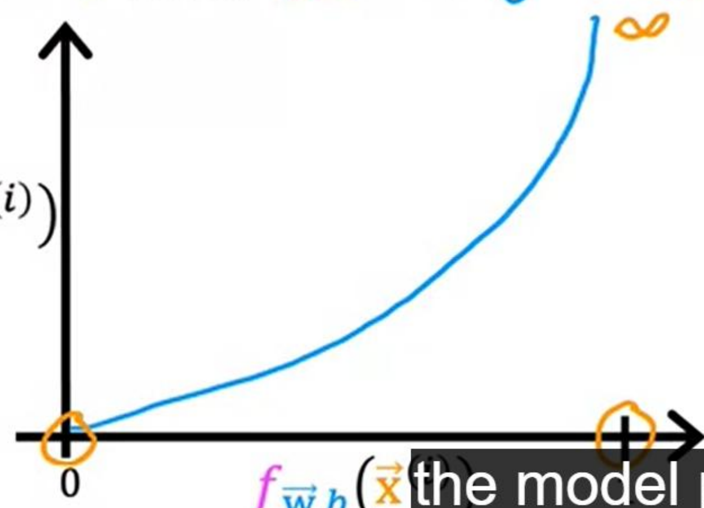
when y is equal to 1.

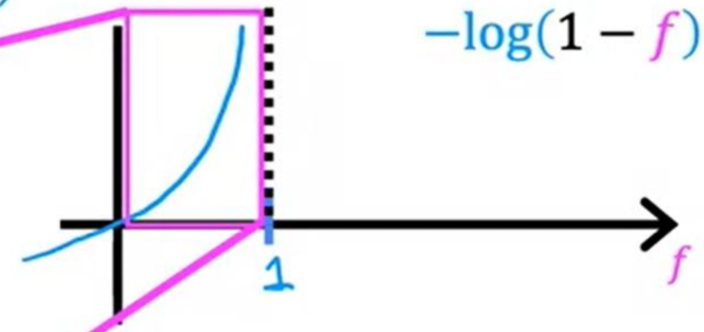
Loss is lowest when $f_{\vec{\mathrm{w}},b}\left(\vec{\mathrm{x}}^{(i)}\right)$ predicts close to true label $y^{(i)}$.

Logistic loss function

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

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





As $f_{\vec{\mathrm{w}},b}(\vec{\mathrm{x}}^{(i)}) \to 1$ t

the model predicts that
the patient's tumor is

Simplified loss function

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

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

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

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

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

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

equivalent to the more
complex expression up here,

Simplified cost function

loss

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

cost

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

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

statistics using a
statistical principle

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

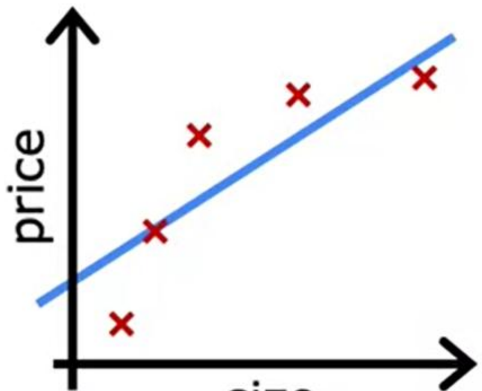
Same concepts:
- Monitor gradient descent (learning curve)
- Vectorized implementation

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

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

gradient descent for
logistic regression.

Regression example

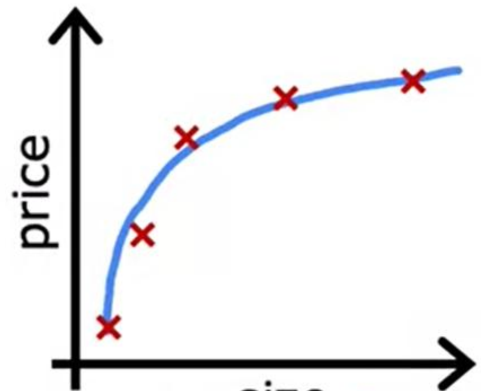


$w_1 x + b$

underfit

- Does not fit the training set well

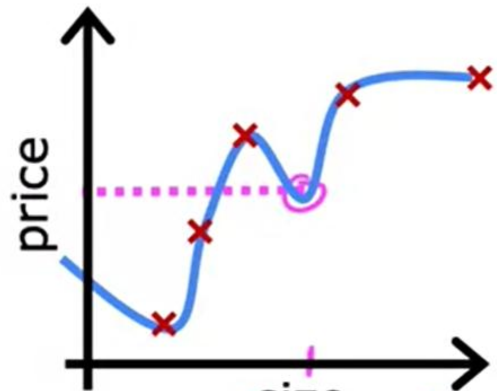
high bias



$w_1 x + w_2 x^2 + b$

- Fits training set pretty well

generalization



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

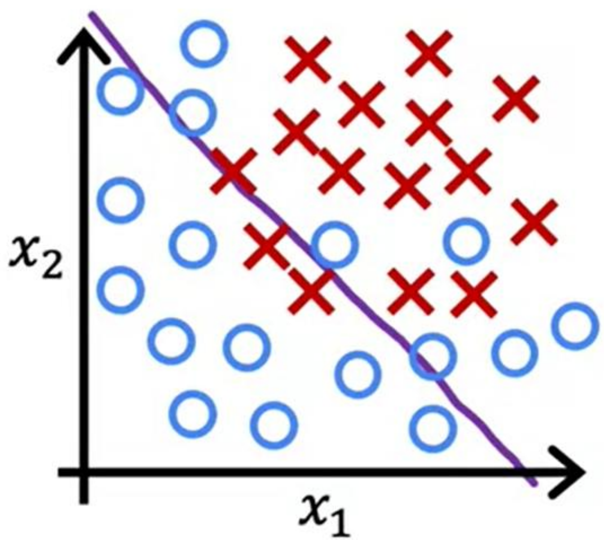
overfit

- Fits the training set extremely well

high varian

that the algorithm
has high variance.

Classification

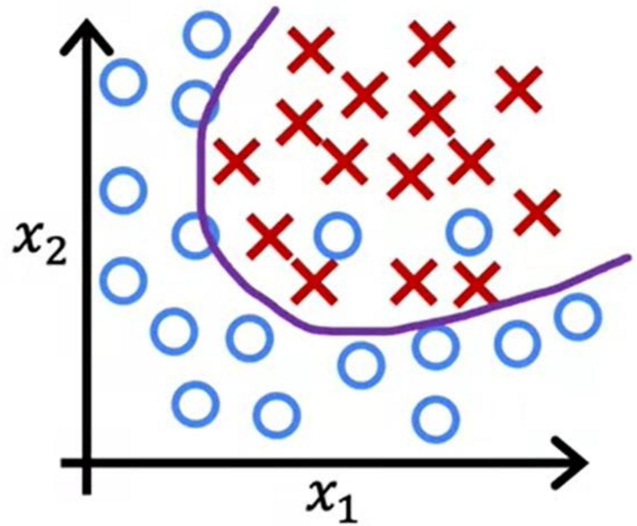


$$z = w_1 x_1 + w_2 x_2 + b$$

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

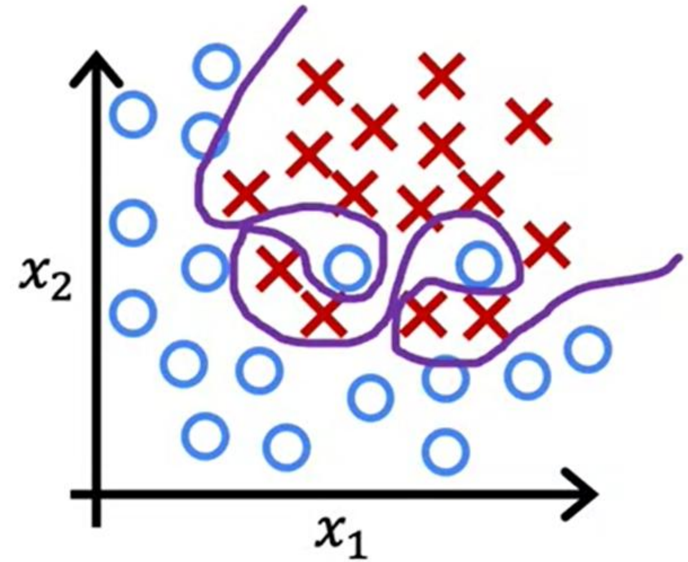
g is the sigmoid function

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

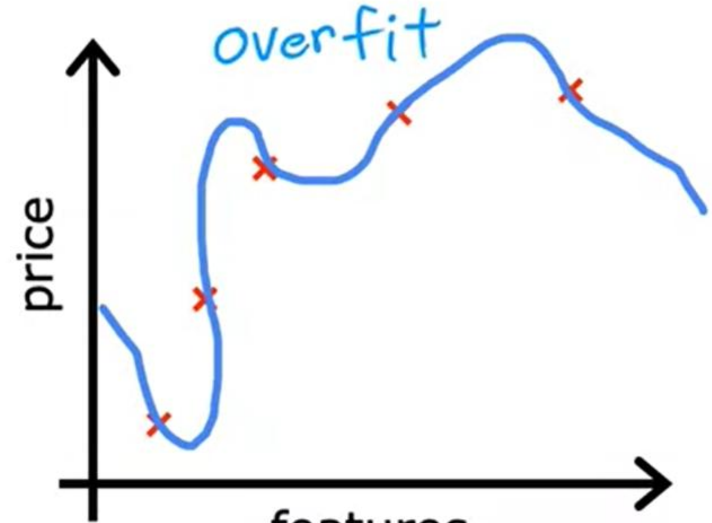


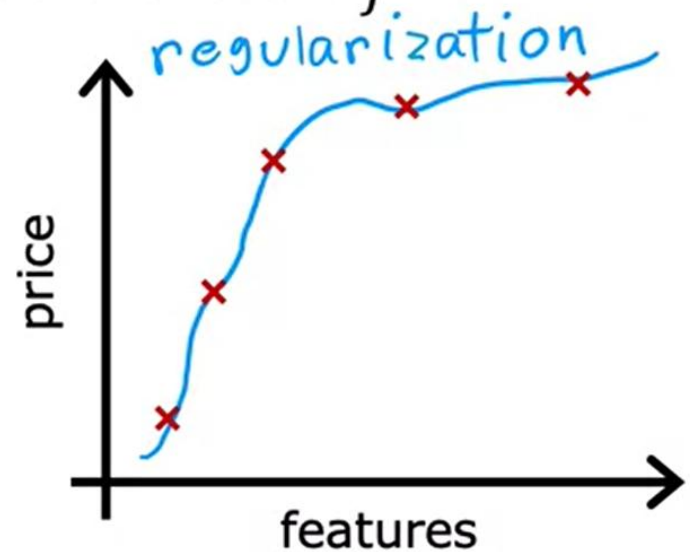
$$z = w_1 x_1 + w_2 x_2 + w_3 x_1^2 x_2 + w_4 x_1^2 x_2^2 + w_5 x_1^2 x_2^3 + w_6 x_1^3 x_2 + \cdots + b$$

Having all these higher-order polynomial features

Regularization

Reduce the size of parameters w_j





$$f(x) = 28x - 385x^2 + 39x^3 - \underbrace{\cancel{174}}_{0}x^4 + 100$$

large values for w_j

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

values for w_j

but they just prevents
the features from

Addressing overfitting

Options

1. Collect more data

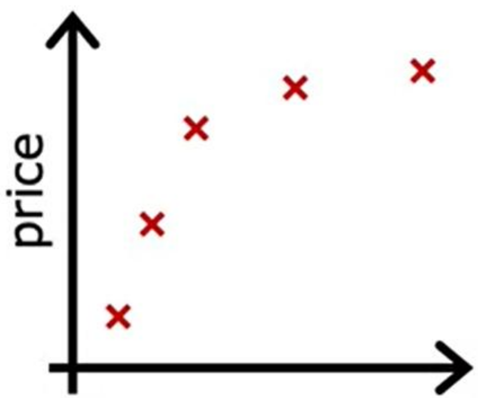
2. Select features
   – Feature selection in course 2

3. Reduce size of parameters
   – "Regularization"

This will be the subject
of the next video as well.

Regularization

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



$$f_{\vec{w},b}(\vec{x}) = w_1x + w_2x^2 + w_3x^3 + w_4x^4 + b$$

which will tend to reduce overfitting.

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 {

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

$j=1,\dots,n$

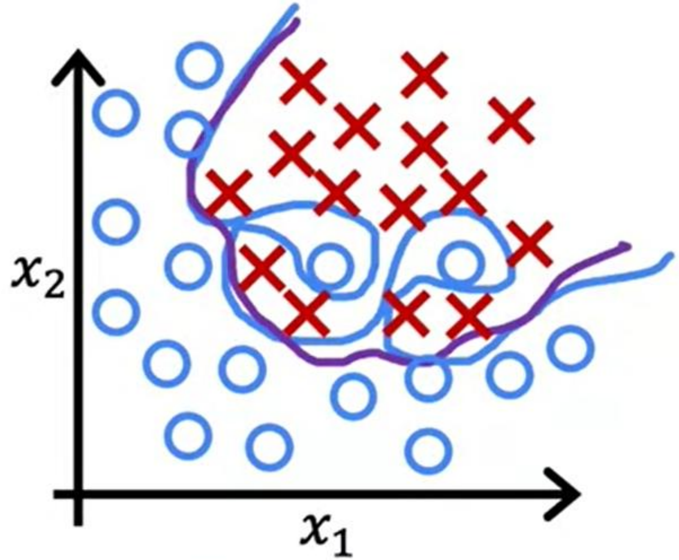
$$b = b - \alpha \frac{\partial}{\partial b} J(\vec{w},b) \qquad = \frac{1}{m}\sum_{i=1}^{m}\left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)}\right)$$

don't have to regularize b

} simultaneous update

the derivatives and
put them back into

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

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

$w_j \downarrow$

wb that includes the
regularization term?

Cost function

In a previous lab, you developed the *logistic loss* function. Recall, loss is defined to apply to one example. Here you combine the losses to form the **cost**, which includes all the examples.

Recall that for logistic regression, the cost function is of the form

$$J(\mathbf{w},b) = \frac{1}{m} \sum_{i=0}^{m-1} \left[loss(f_{\mathbf{w},b}(\mathbf{x}^{(i)}), y^{(i)}) \right] \tag{1}$$

where

- $loss(f_{\mathbf{w},b}(\mathbf{x}^{(i)}), y^{(i)})$ is the cost for a single data point, which is:

$$loss(f_{\mathbf{w},b}(\mathbf{x}^{(i)}), y^{(i)}) = -y^{(i)} \log\left(f_{\mathbf{w},b}\left(\mathbf{x}^{(i)} \right) \right) - \left(1 - y^{(i)}\right) \log \left(1 - f_{\mathbf{w},b}\left(\mathbf{x}^{(i)} \right) \right) \tag{2}$$

- where m is the number of training examples in the data set and:

$$\begin{align}
f_{\mathbf{w},b}(\mathbf{x^{(i)}}) &= g(z^{(i)}) \tag{3} \\
z^{(i)} &= \mathbf{w} \cdot \mathbf{x}^{(i)}+ b \tag{4} \\
g(z^{(i)}) &= \frac{1}{1+e^{-z^{(i)}}} \tag{5}
\end{align}$$