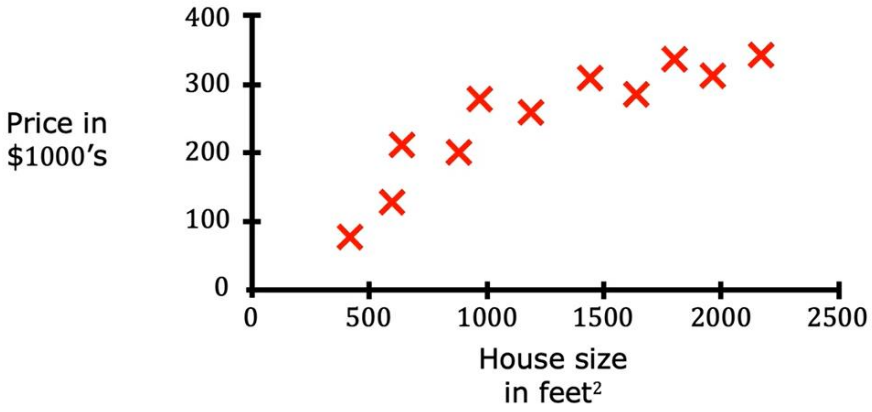
Input (X)		Output (Y)	Application
email	→	spam? (0/1)	spam filtering
audio	→	text transcripts	speech recognition
English	→	Spanish	machine translation
ad, user info	→	click? (0/1)	online advertising
image, radar info	→	position of other cars	self-driving car
image of phone	→	defect? (0/1)	visual inspection

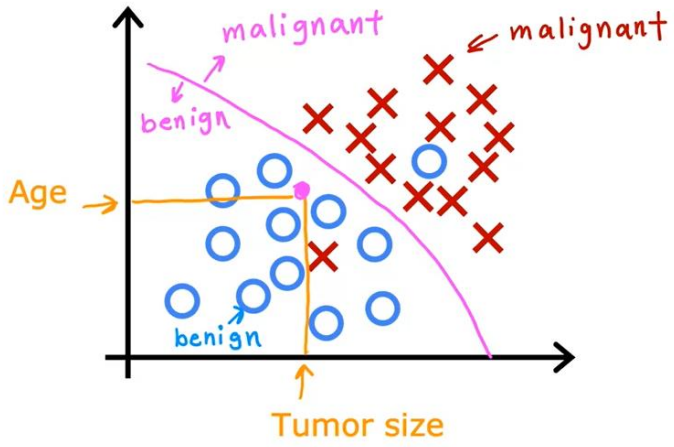
appropriate
corresponding output y.

Regression: Housing price prediction



is the size of the
house in square feet.

Two or more inputs



uniformity of the cell shape and so on.

Stanford ONLINE DeepLearning.AI Andrew Ng

6:23 / 7:16

Supervised learning

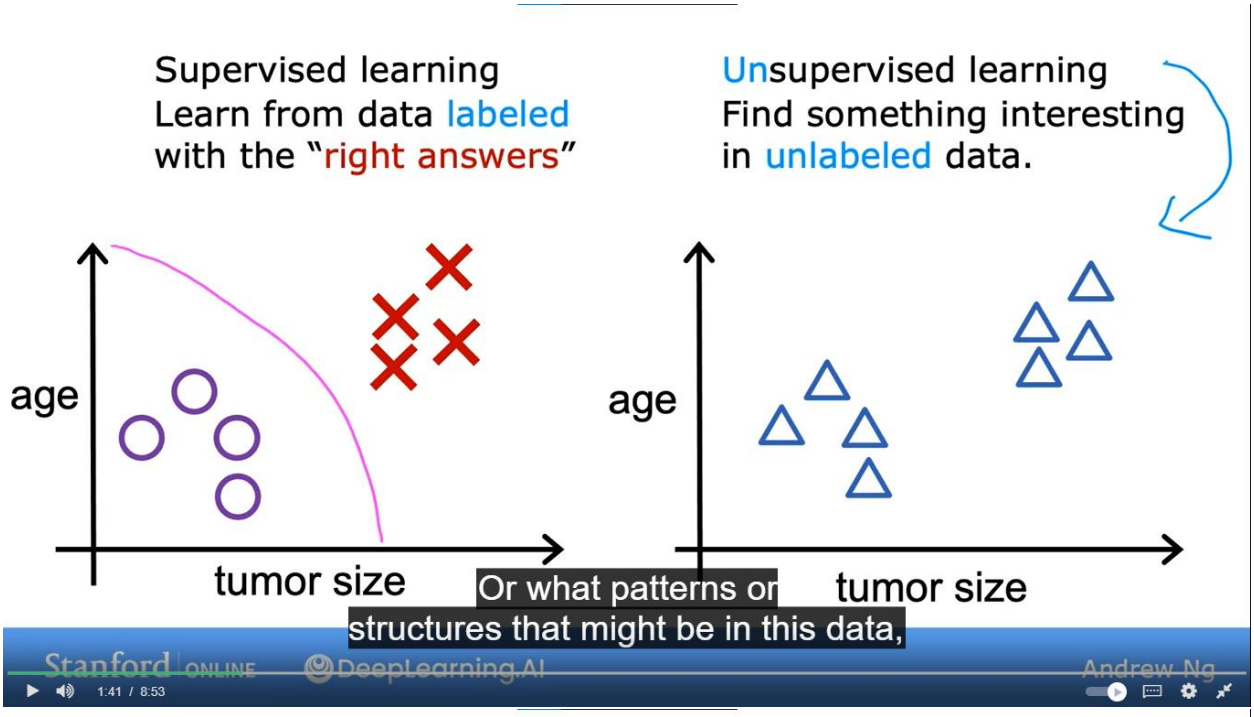
Learns from being given "right answers"

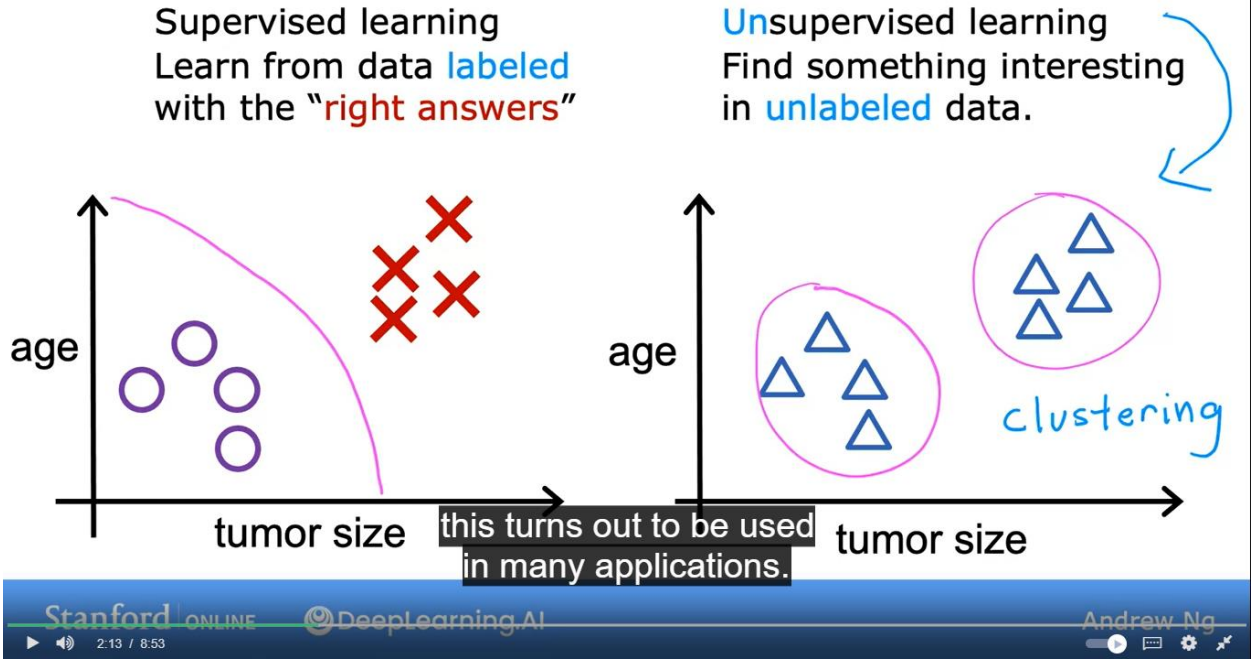
Regression	Classification
Predict a number	predict categories
infinitely many possible outputs	small number of possible outputs

a category,
all of a small set of possible outputs.

Stanford ONLINE DeepLearning.AI Andrew Ng

6:57 / 7:16


Supervised learning
Learn from data labeled with the "right answers"
Unsupervised learning
Find something interesting in unlabeled data.
age
tumor size
age
tumor size
Or what patterns or structures that might be in this data,
Stanford ONLINE
DeepLearning.AI
Andrew Ng
1:41 / 8:53


Supervised learning
Learn from data labeled with the "right answers"
Unsupervised learning
Find something interesting in unlabeled data.
age
tumor size
age
clustering
tumor size
this turns out to be used in many applications.
Stanford ONLINE
DeepLearning.AI
Andrew Ng
2:13 / 8:53

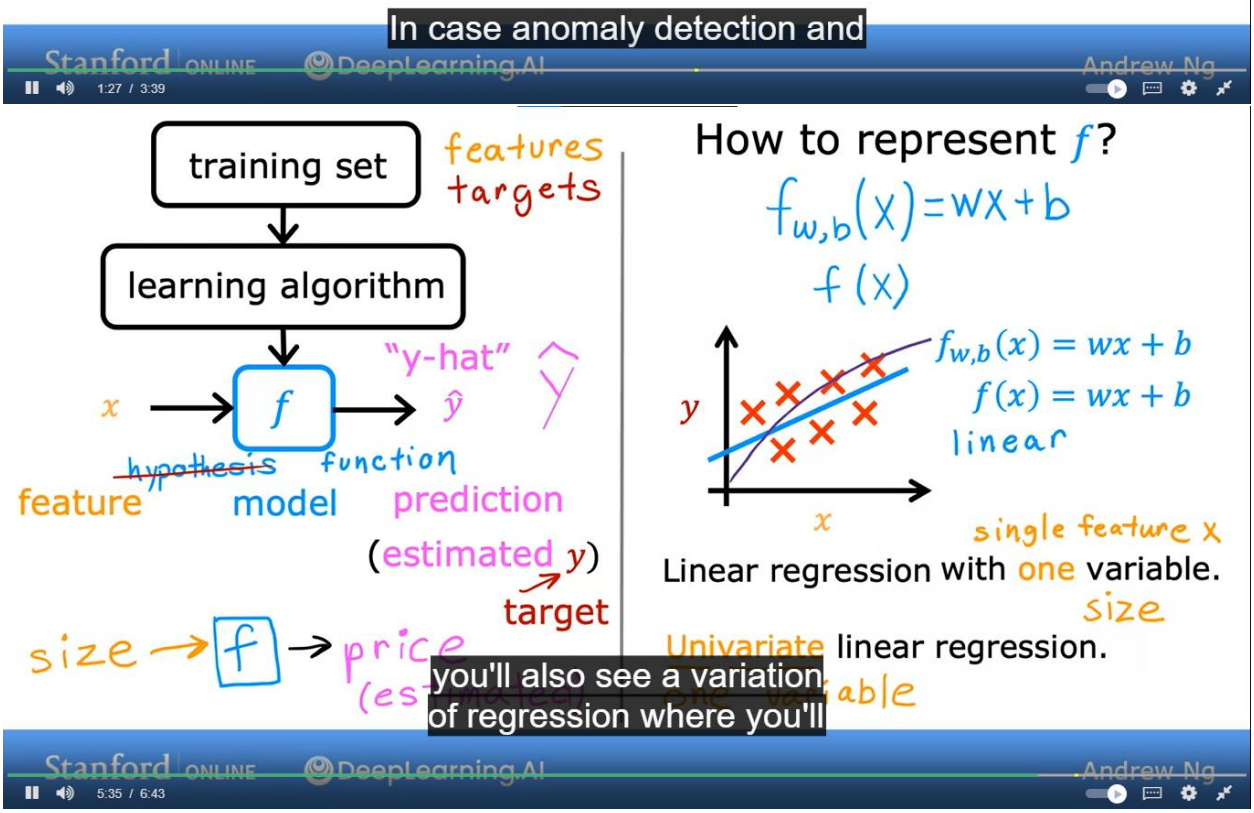
Unsupervised learning

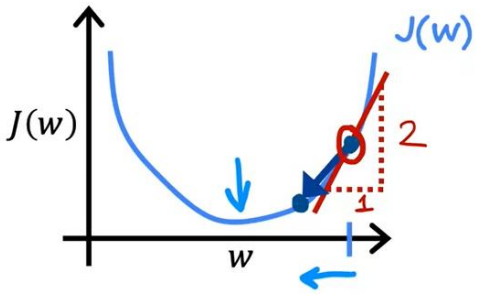
Data only comes with inputs x, but not output labels y.
Algorithm has to find structure in the data.

Clustering
Group similar data points together.

Dimensionality reduction
Compress data using fewer numbers.

Anomaly detection
Find unusual data points.

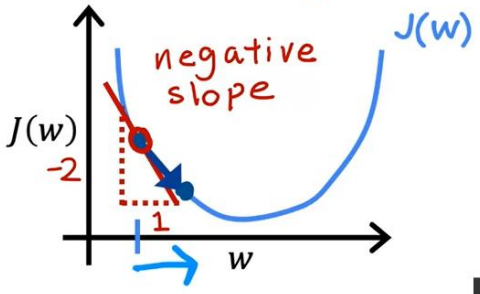




$$w = w - \alpha \frac{d}{dw} J(w)$$

> 0

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



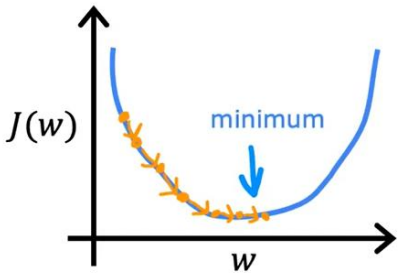
$$\frac{d}{dw} J(w) < 0$$

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

Again, it looks like

Stanford ONLINE DeepLearning.AI Andrew Ng

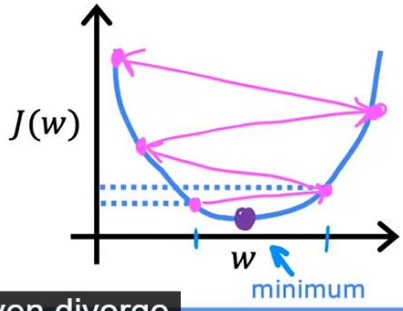
$$w = w - \alpha \frac{d}{dw} J(w)$$



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



may fail to converge and may even diverge.

Stanford ONLINE DeepLearning.AI Andrew Ng

4:13 / 9:03

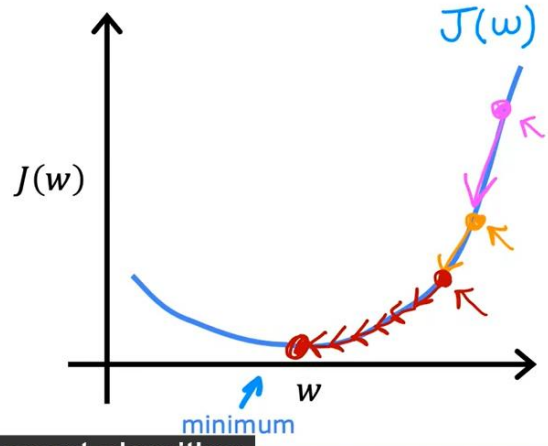
Can reach local minimum with fixed learning rate

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

smaller

not as large

large



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

Can reach minimum without decreasing learning rate α

So that's the gradient descent algorithm,

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

$\vec{w} = [w_1 \; w_2 \; w_3 \ldots w_n]$ 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

multiple linear regression

which has just one feature.

Without vectorization

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

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

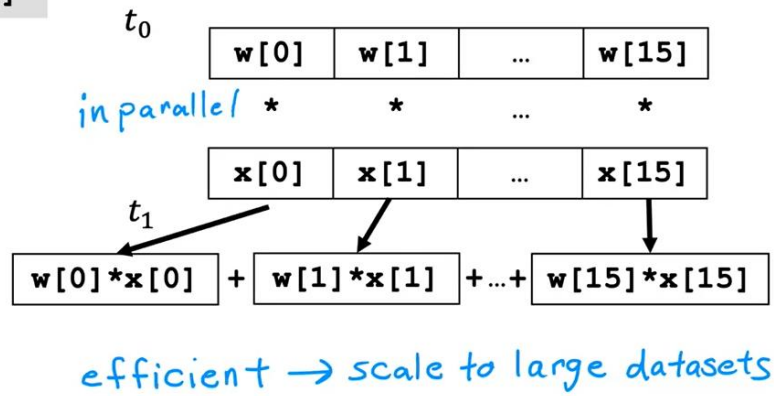
t_1
`f + w[1] * x[1]`

...

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

Vectorization

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



efficient → scale to large datasets

now have to operate on.

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])
```

compute $w_j = w_j - 0.1 d_j$ for $j = 1 \ldots 16$ (learning rate α)

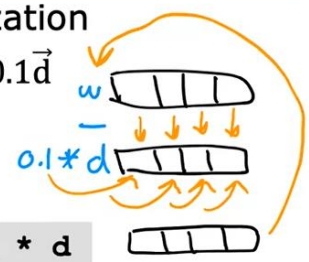
Without vectorization

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

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

With vectorization

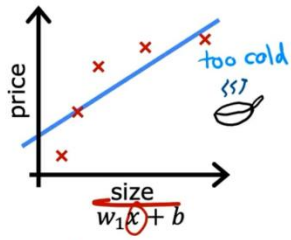
$\vec{w} = \vec{w} - 0.1\vec{d}$




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

w is assigned to w minus 0.1 times d. Behind the scenes,

Regression example

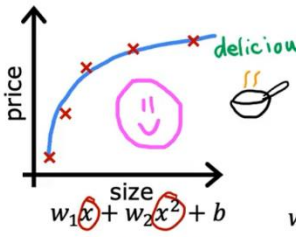


$w_1x + b$

underfit

- Does not fit the training set well

high bias

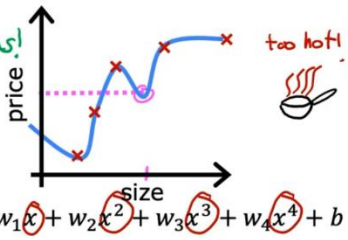


$w_1x + w_2x^2 + b$

just right

- Fits training set pretty well

generalization



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

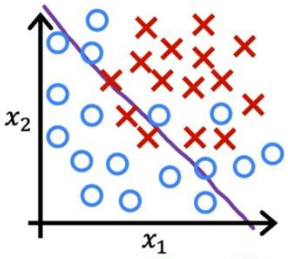
overfit

- Fits the training set extremely well

high variance

that seems to be just right.

Classification

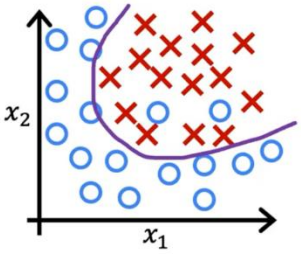


$z = w_1x_1 + w_2x_2 + b$

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

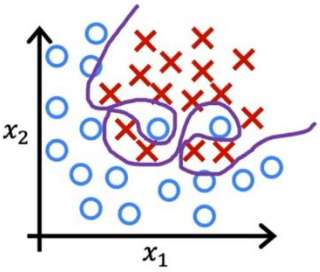
g is the sigmoid function

underfit high bias



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

just right

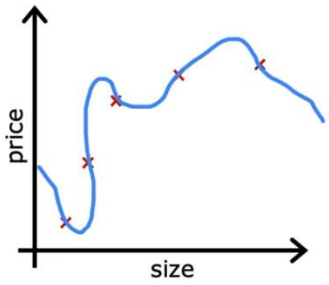


$z = w_1x_1 + w_2x_2 + w_3x_1^2x_2 + w_4x_1^2x_2^2 + w_5x_1^2x_2^3 + w_6x_1^3x_2 + \cdots + b$

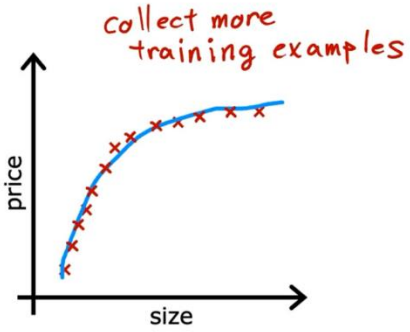
overfit

doesn't look like it'll generalize well to new examples.

Collect more training examples



overfit



been sold in this location,

Select features to include/exclude

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

all features

\+

insufficient data

↓

overfit

selected features

size
bedrooms
age
just right

feature selection

disadvantage

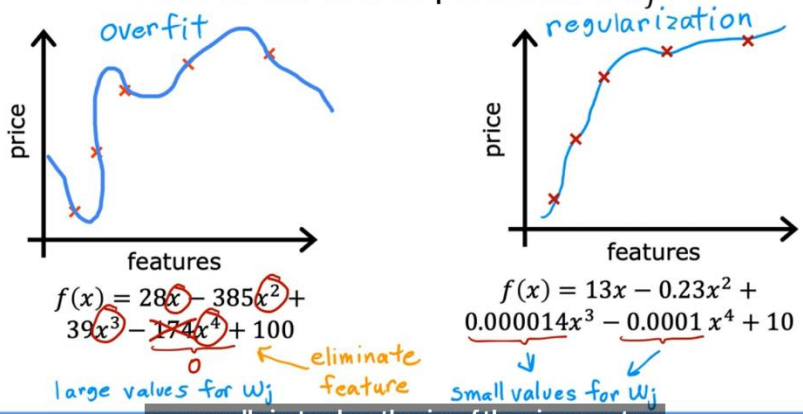
↓

useful features could be lost

the information by throwing away some of the features.

Regularization

Reduce the size of parameters w_j



we normally just reduce the size of the wj parameters,

Addressing overfitting

Options

1. Collect more data

2. Select features
   - Feature selection in course 2

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

for training learning algorithms,