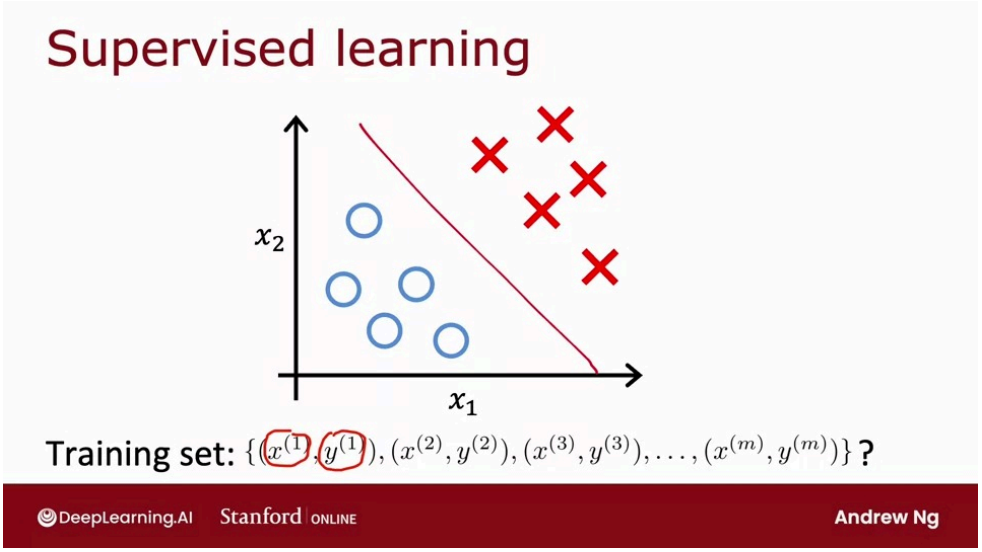
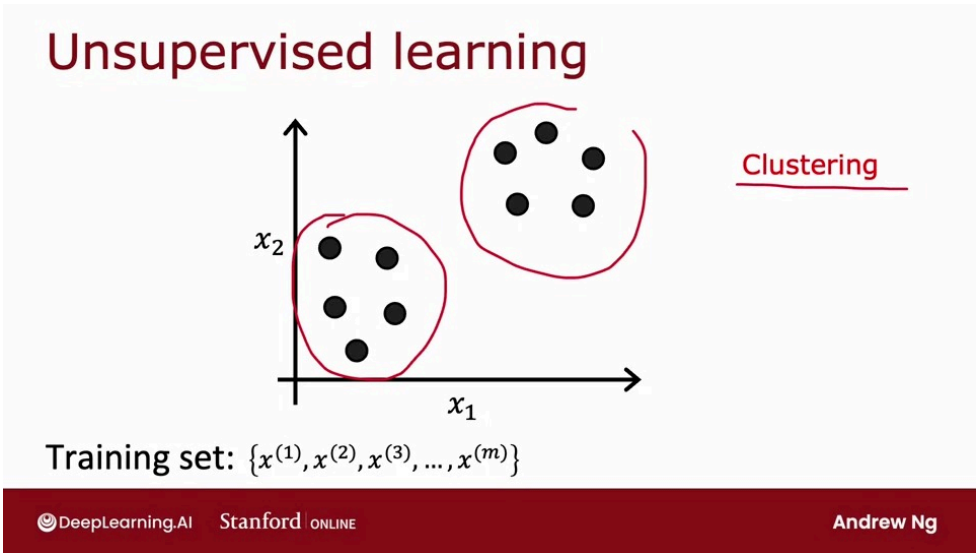
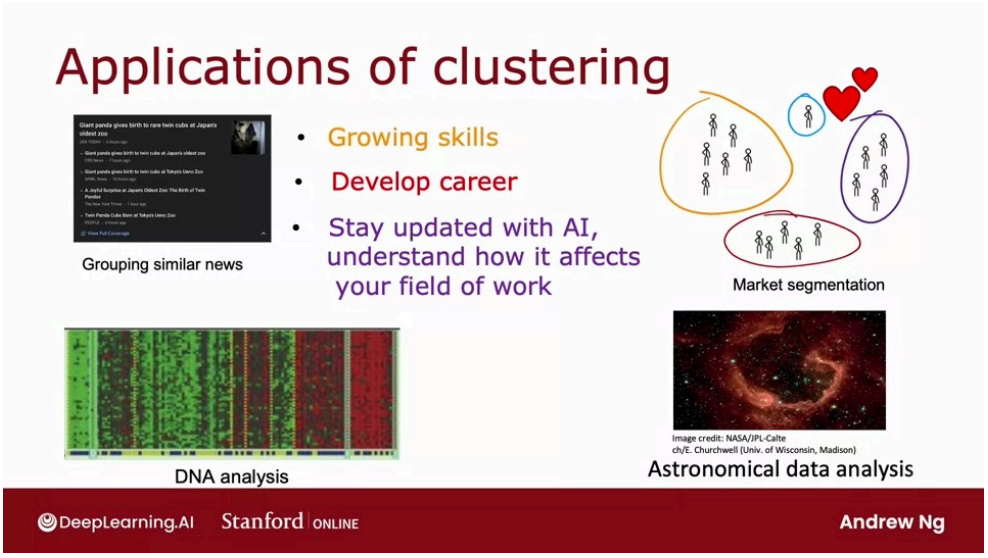
Unsupervised Learning, Recommenders, Reinforcement Learning

Clustering

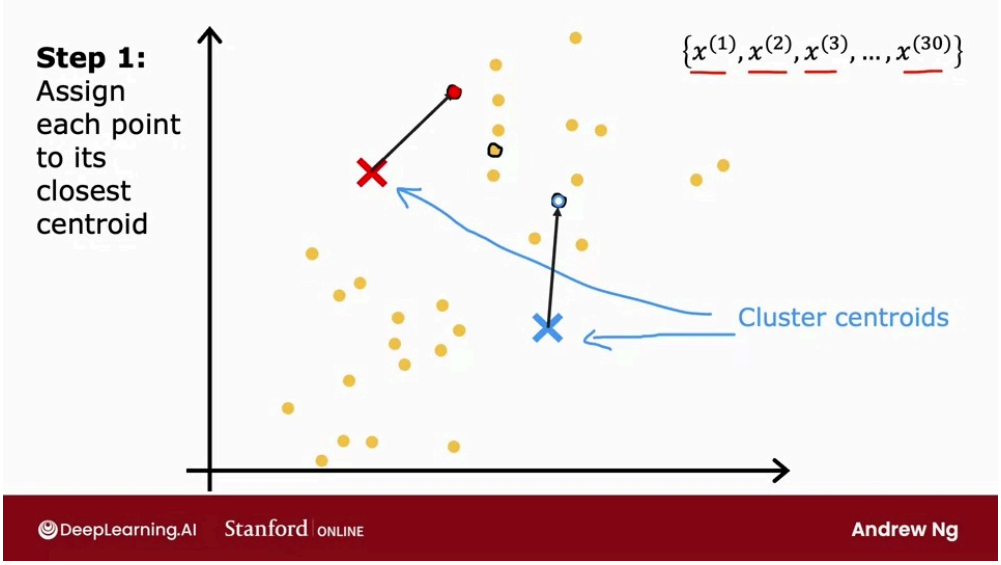
Week1

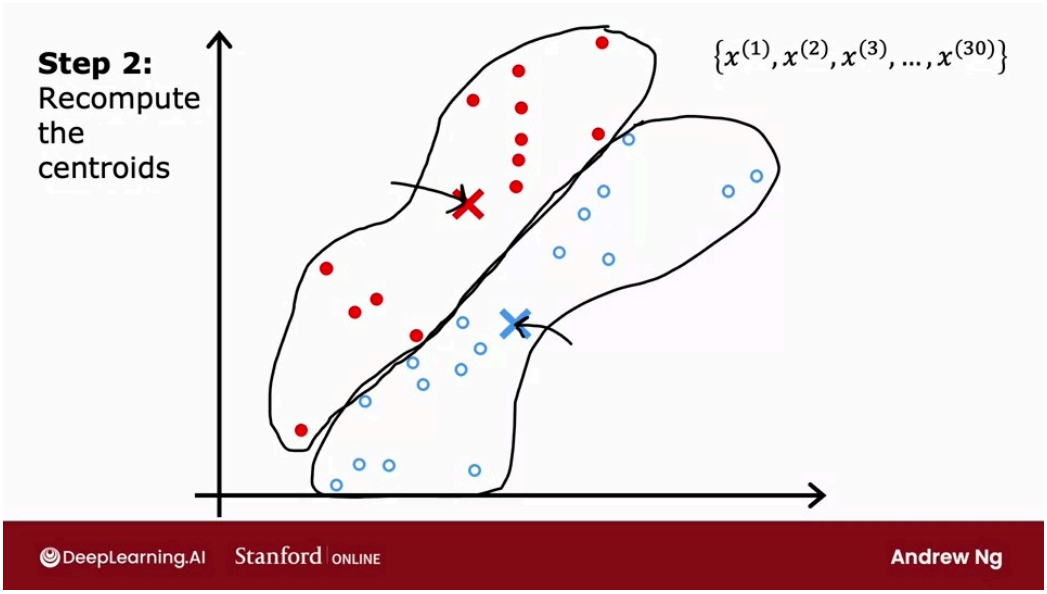


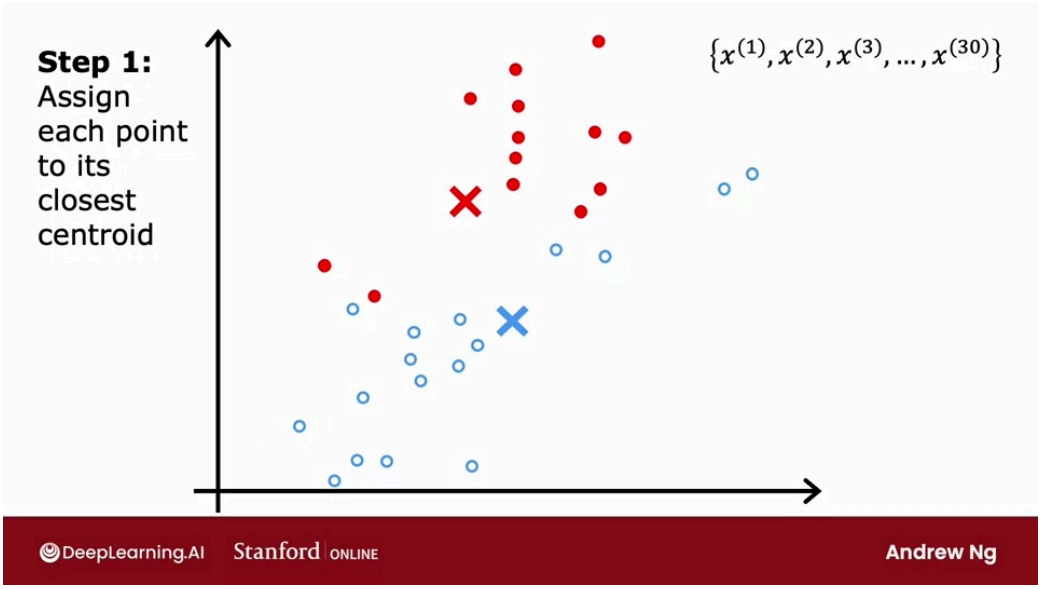


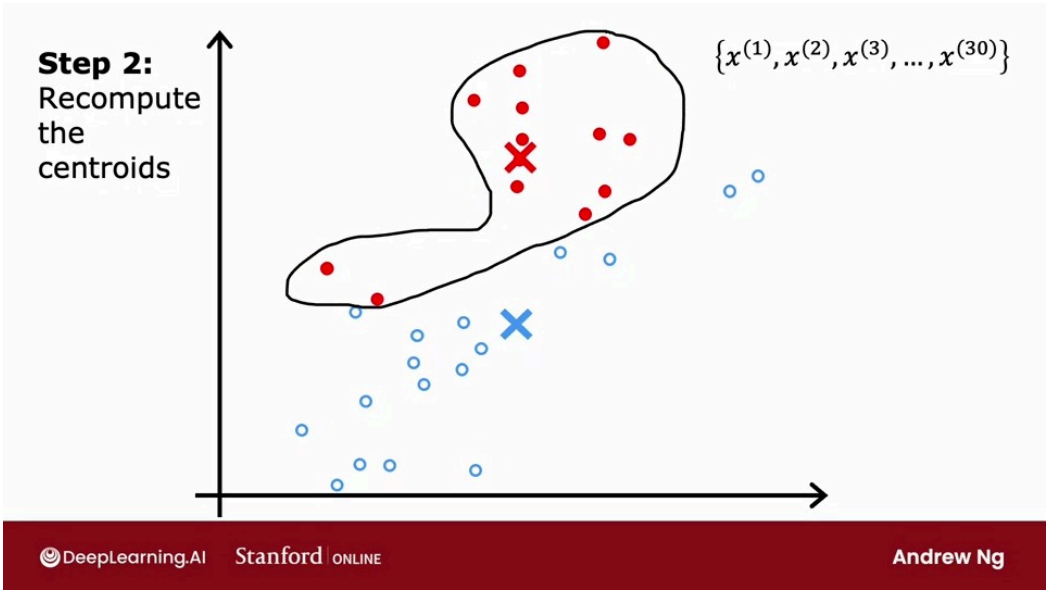


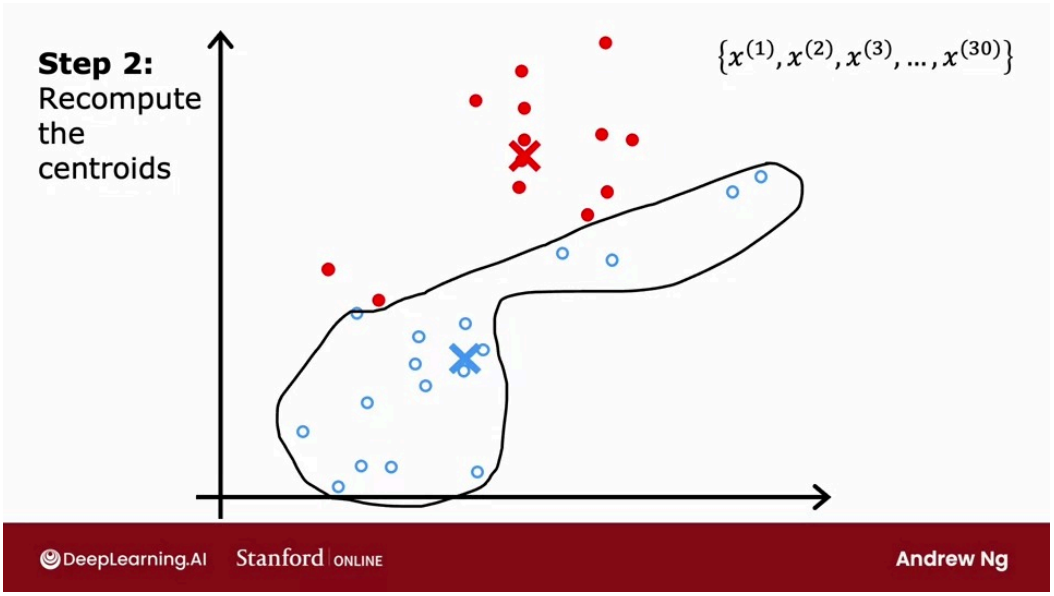
K-means intuition

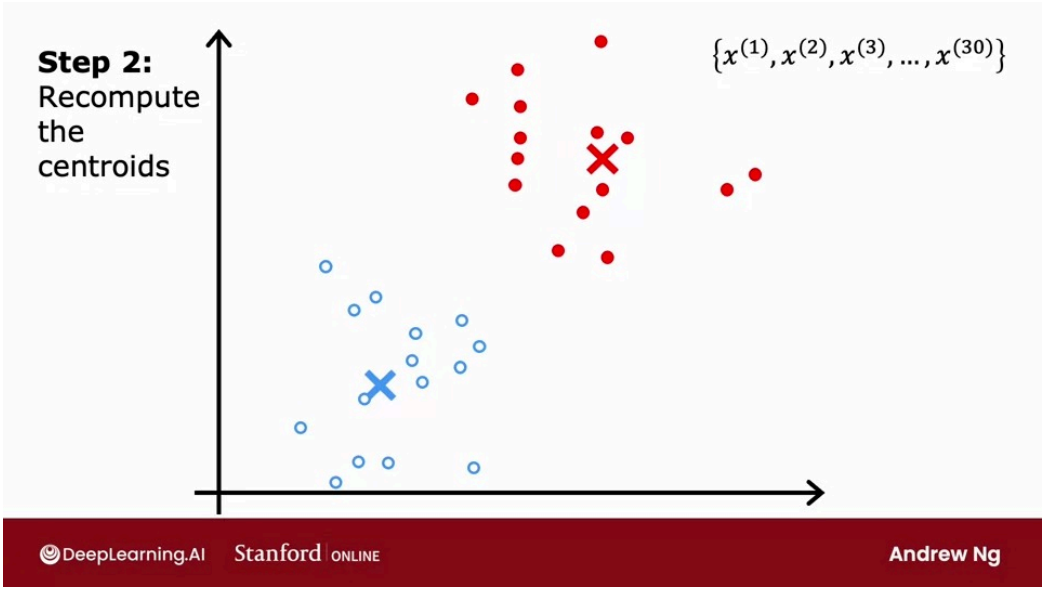


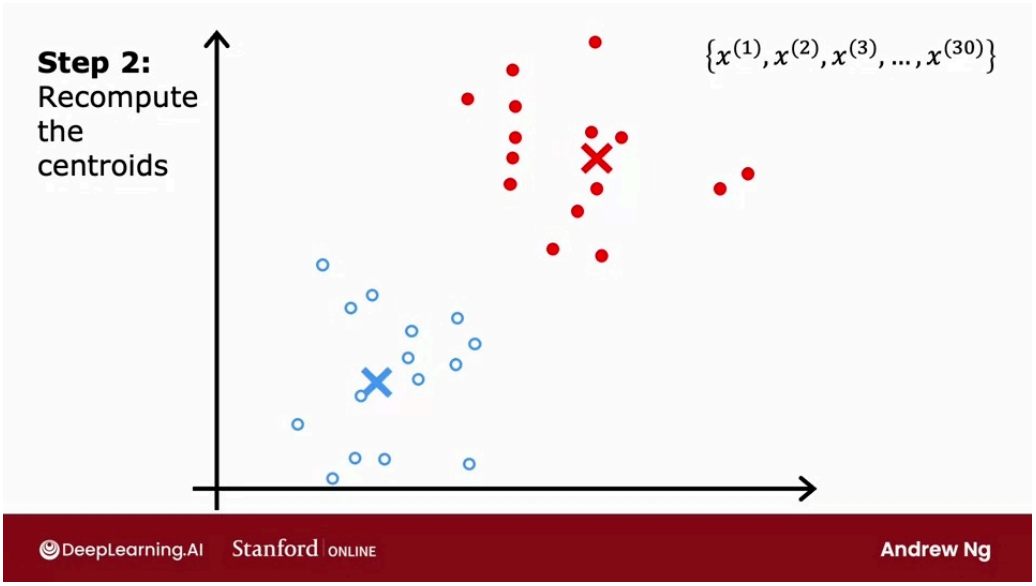




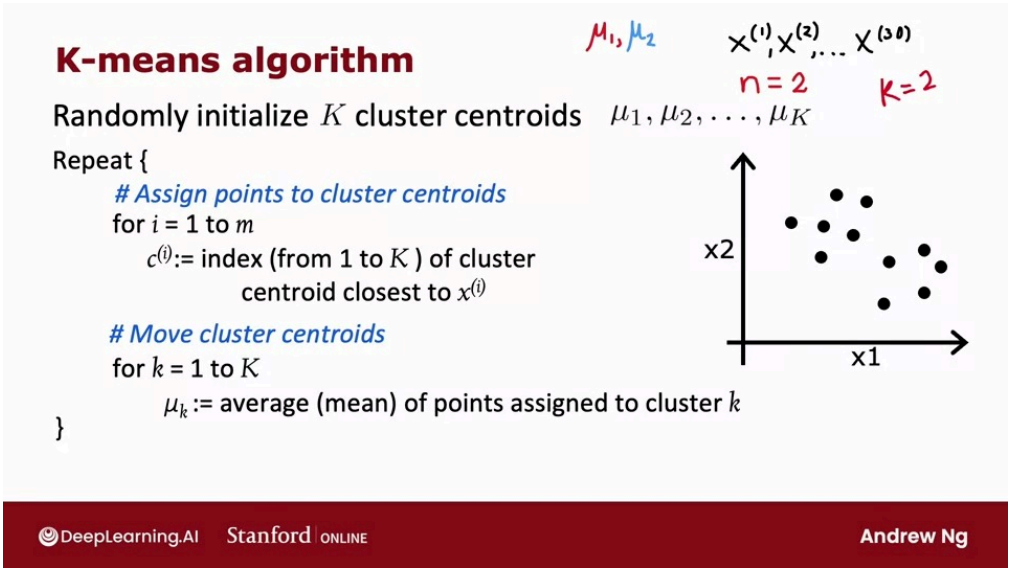

Step 2:
Recompute
the
centroids
$\{x^{(1)}, x^{(2)}, x^{(3)}, \ldots, x^{(30)}\}$
DeepLearning.AI
Stanford ONLINE
Andrew Ng

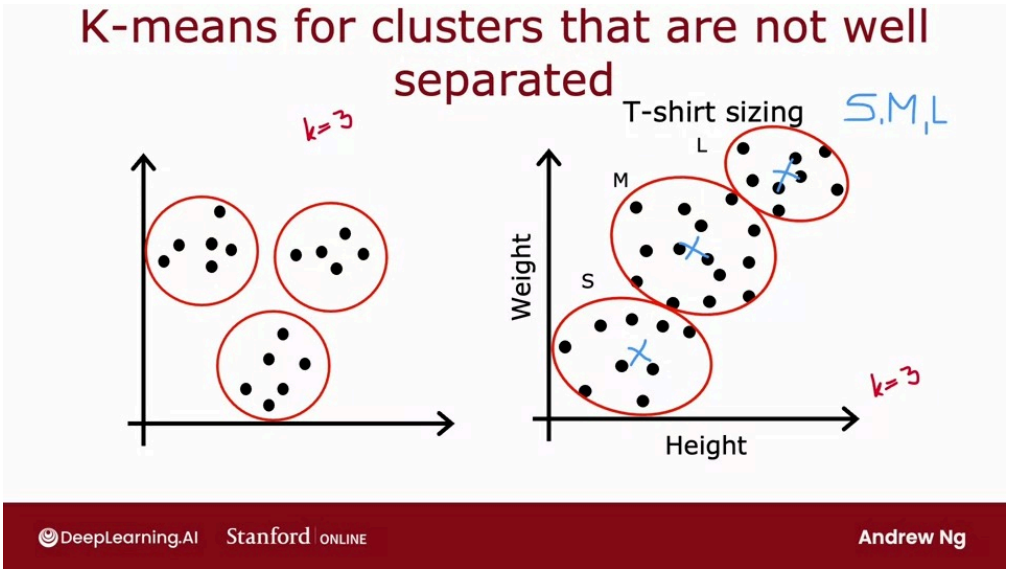

Step 2:
Recompute
the
centroids
$\{x^{(1)}, x^{(2)}, x^{(3)}, \ldots, x^{(30)}\}$
DeepLearning.AI
Stanford ONLINE
Andrew Ng


Step 2:
Recompute
the
centroids
$\{x^{(1)}, x^{(2)}, x^{(3)}, \ldots, x^{(30)}\}$
DeepLearning.AI
Stanford ONLINE
Andrew Ng



K-means algorithm





Optimization objective

K-means optimization objective

$c^{(i)}$ = index of cluster $(1, 2, \ldots, K)$ to which example $x^{(i)}$ is currently assigned

μ_k = cluster centroid k

$\mu_{c^{(i)}}$ = cluster centroid of cluster to which example $x^{(i)}$ has been assigned

Cost function

$$J(c^{(1)}, \ldots, c^{(m)}, \mu_1, .., \mu_K) = \frac{1}{m}\sum_{i=1}^{m} \|x^{(i)} - \mu_{c^{(i)}}\|^2$$

$$\min_{\substack{c^{(1)}, \ldots, c^{(m)} \\ \mu_1, \ldots, \mu_K}} J(c^{(1)}, \ldots, c^{(m)}, \mu_1, .., \mu_K)$$

Distortion

DeepLearning.AI Stanford ONLINE Andrew Ng

Cost function for K-means

$$J(c^{(1)}, \ldots, c^{(m)}, \mu_1, .., \mu_K) = \frac{1}{m}\sum_{i=1}^{m} \|x^{(i)} - \mu_{c^{(i)}}\|^2$$

Repeat {

 # Assign points to cluster centroids

for i = 1 to m

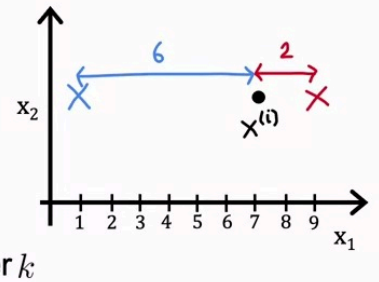
$c^{(i)}$:= index of cluster centroid closest to $x^{(i)}$

 # Move cluster centroids

for k = 1 to K

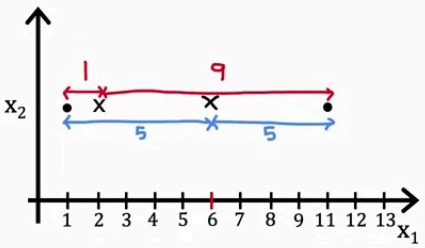
μ_k := average of points in cluster k

}



DeepLearning.AI Stanford ONLINE Andrew Ng

Moving the centroid



$\frac{1}{2}(1^2 + 9^2) = \frac{1}{2}(1+81) = 41$

$\frac{1}{2}(1 + 11) = 6$

$\frac{1}{2}(5^2 + 5^2) = 25$

DeepLearning.AI Stanford ONLINE Andrew Ng

Initializing K-means

K-means algorithm

Step 0: Randomly initialize K cluster centroids $\mu_1, \mu_1, \ldots, \mu_k$

Repeat {

Step 1: Assign points to cluster centroids

Step 2: Move cluster centroids

}

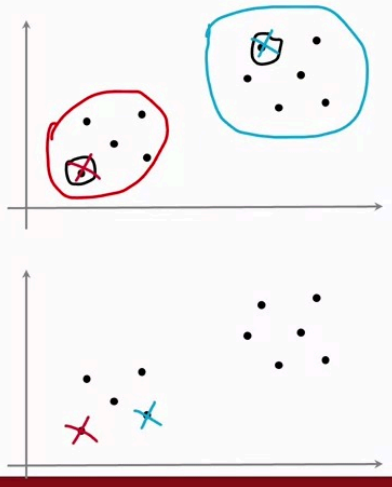
DeepLearning.AI Stanford ONLINE Andrew Ng

Random initialization

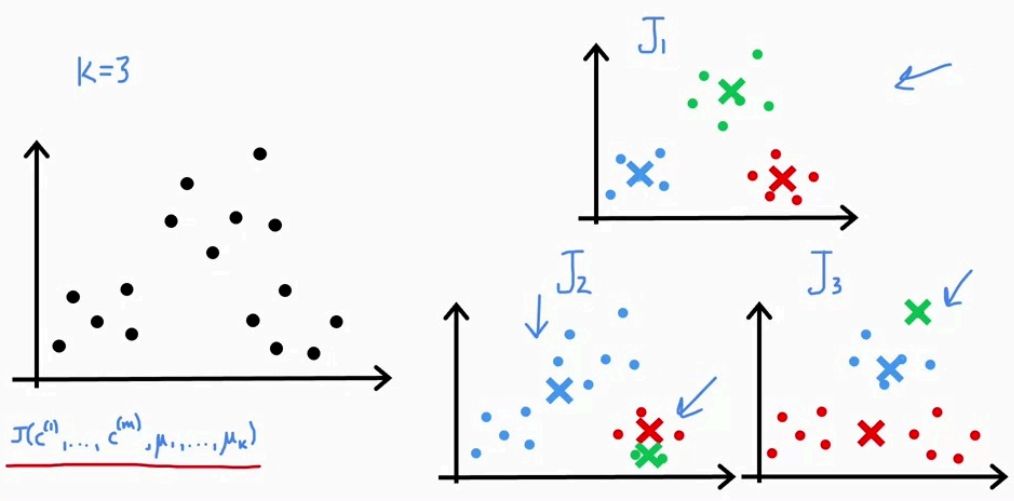
Choose $K < m$

Randomly pick K training examples.

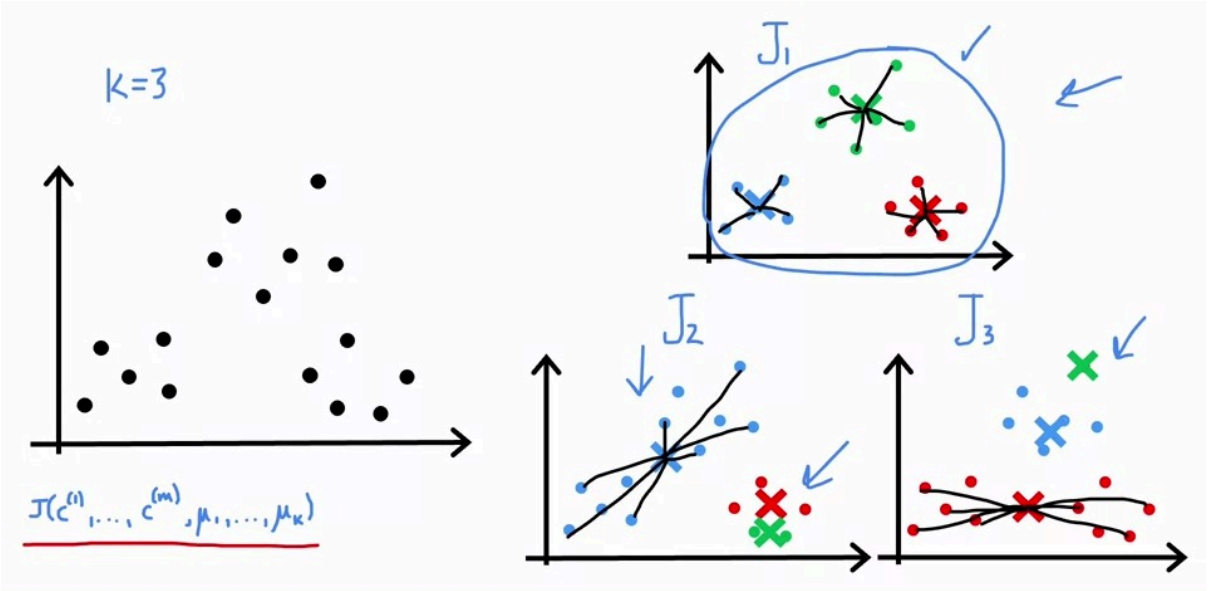
Set $\mu_1, \mu_1, \ldots, \mu_k$ equal to these K examples.



DeepLearning.AI Stanford ONLINE Andrew Ng



DeepLearning.AI Stanford ONLINE Andrew Ng



Random initialization

For $i = 1$ to 100 {

50-1000

k random examples

Randomly initialize K-means.

Run K-means. Get $c^{(1)}, \ldots, c^{(m)}, \mu_1, \mu_1, \ldots, \mu_k$

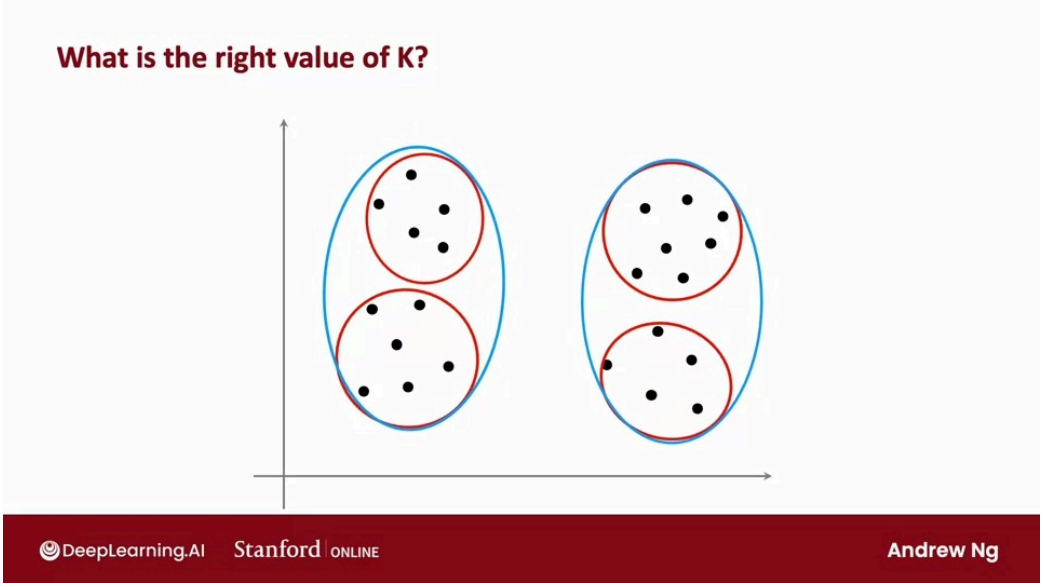
Computer cost function (distortion)

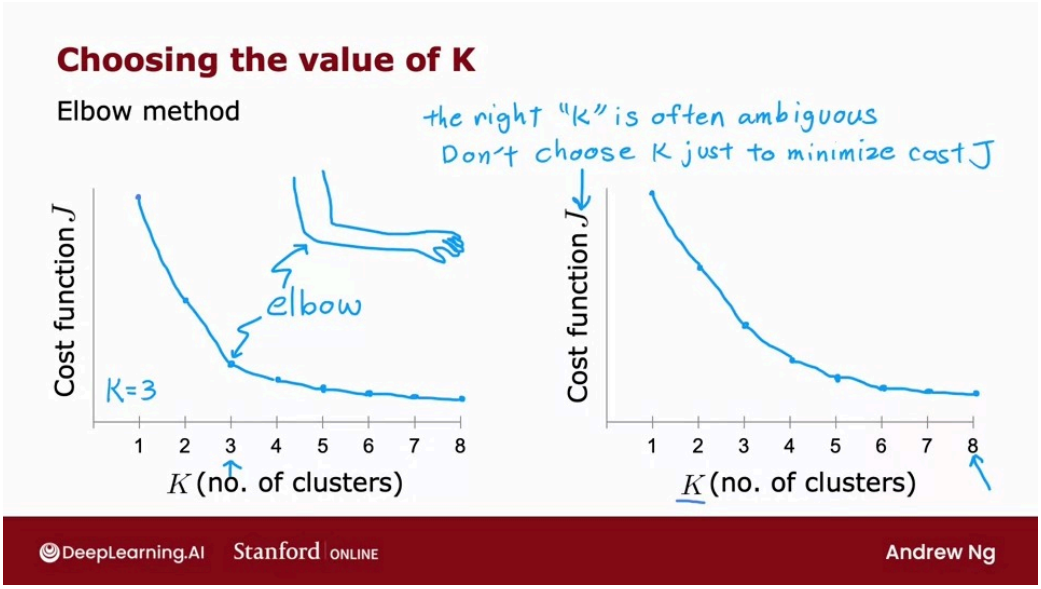
$J(c^{(1)}, \ldots, c^{(m)}, \mu_1, \mu_1, \ldots, \mu_k)$

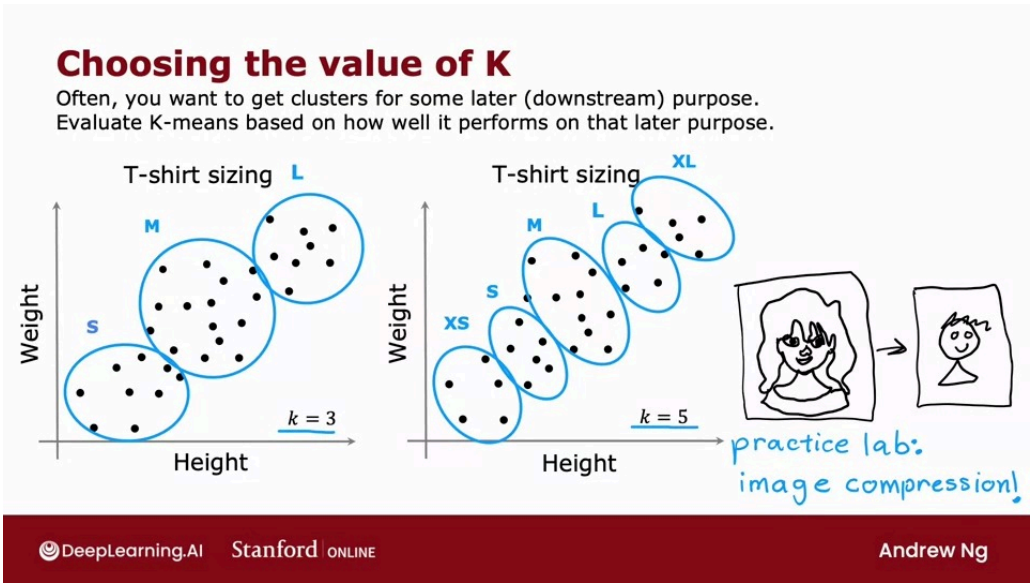
}

Pick set of clusters that gave lowest cost J

Choosing the number of clusters







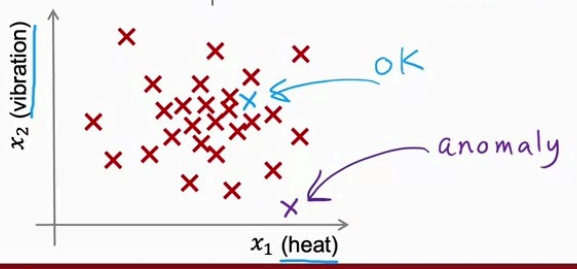
Anomaly detection

Anomaly detection example

Aircraft engine features:
x_1 = heat generated
x_2 = vibration intensity
...

Dataset: $\{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\}$

New engine: x_{test}



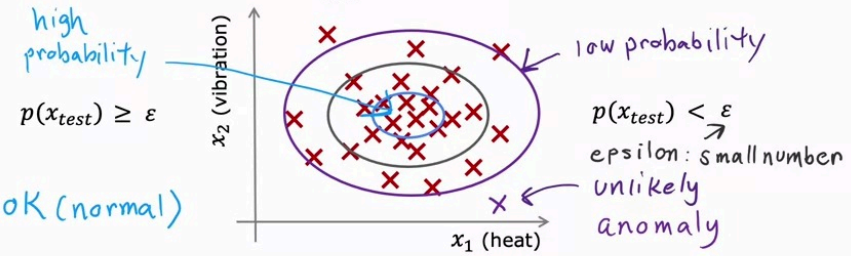
Density estimation

Dataset: $\{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\}$

probability of x being seen in dataset

Model $p(x)$

Is x_{test} anomalous?



Anomaly detection example

how often log in?
how many web pages visited?
transactions?
posts? typing speed?

Fraud detection:
- $x^{(i)}$ = features of user i's activities
- Model $p(x)$ from data.
- Identify unusual users by checking which have $p(x) < \varepsilon$

perform additional checks to identify real fraud vs. false alarms

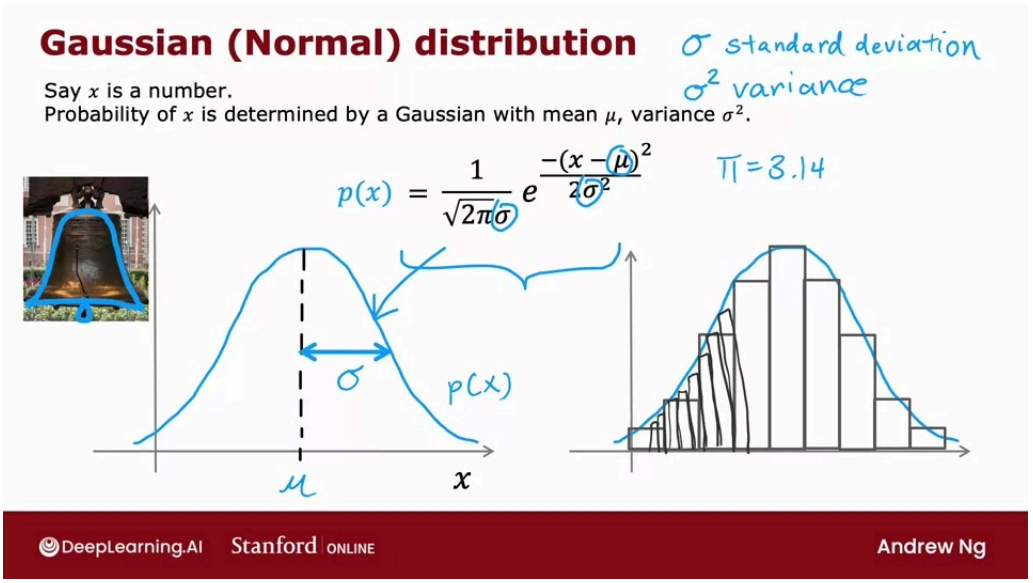
Manufacturing:
$x^{(i)}$ = features of product i

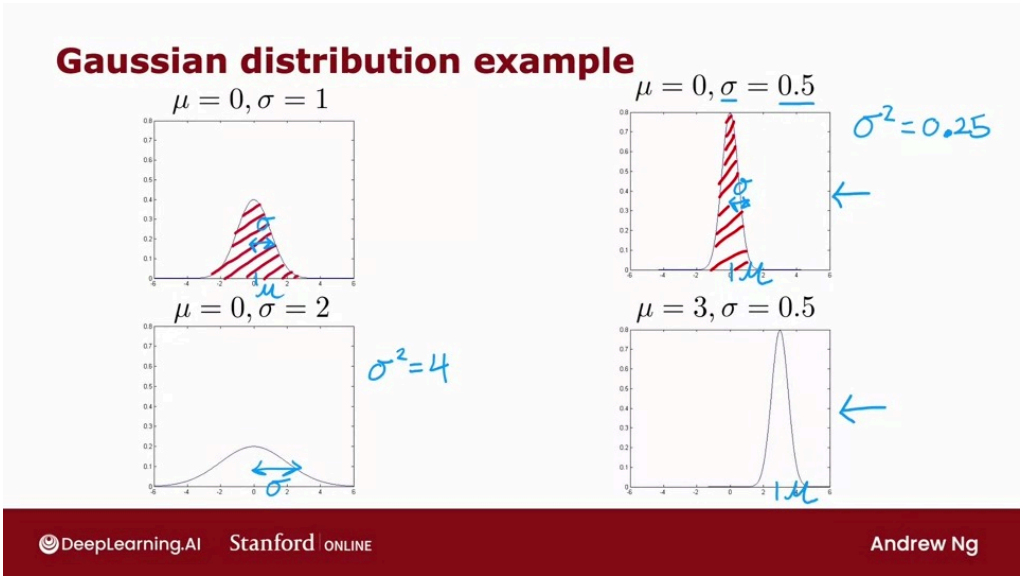
airplane engine
circuit board
smartphone

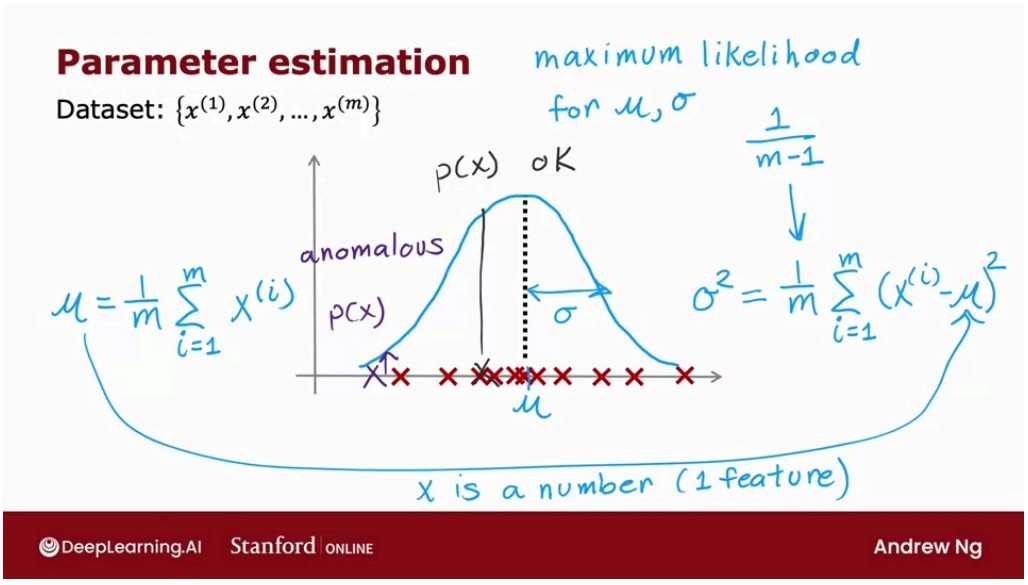
ratios

Monitoring computers in a data center:
$x^{(i)}$ = features of machine i
- x_1 = memory use,
- x_2 = number of disk accesses/sec,
- x_3 = CPU load,
- x_4 = CPU load/network traffic.

Gaussian (normal) distribution



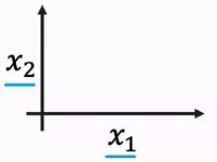




Anomaly detection algorithm

Density estimation

Training set: $\{\vec{x}^{(1)}, \vec{x}^{(2)}, \ldots, \vec{x}^{(m)}\}$
Each example $\vec{x}^{(i)}$ has n features



$$\vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

$$p(\vec{x}) = p(x_1; \mu_1, \sigma_1^2) * p(x_2; \mu_2, \sigma_2^2) * p(x_3; \mu_3, \sigma_3^2) * \cdots * p(x_n; \mu_n, \sigma_n^2)$$

$$= \prod_{j=1}^{n} p(x_j; \mu_j, \sigma_j^2)$$

$\sum$ "add" $\prod$ "multiply"

$p(x_1 = \text{high temp}) = 1/10$
$p(x_2 = \text{high vibra}) = 1/20$
$p(x_1, x_2) = p(x_1) * p(x_2)$
$= \frac{1}{10} \times \frac{1}{20} = \frac{1}{200}$

DeepLearning.AI Stanford ONLINE Andrew Ng

Anomaly detection algorithm

1. Choose n features x_i that you think might be indicative of anomalous examples.
2. Fit parameters $\mu_1, \ldots, \mu_n, \sigma_1^2, \ldots, \sigma_n^2$

$$\mu_j = \frac{1}{m}\sum_{i=1}^{m} x_j^{(i)} \qquad \sigma_j^2 = \frac{1}{m}\sum_{i=1}^{m} (x_j^{(i)} - \mu_j)^2$$

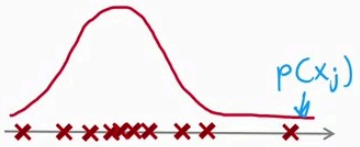
Vectorized formula

$$\vec{\mu} = \frac{1}{m}\sum_{i=1}^{m} \vec{x}^{(i)} \qquad \vec{\mu} = \begin{bmatrix} \mu_1 \\ \mu_2 \\ \ldots \\ \mu_n \end{bmatrix}$$

3. Given new example x, compute $p(x)$:

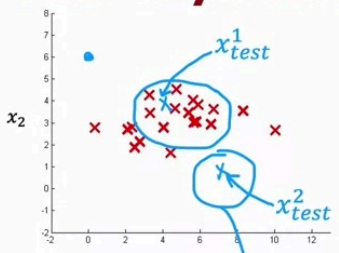
$$p(x) = \prod_{j=1}^{n} p(x_j; \mu_j, \sigma_j^2) = \prod_{j=1}^{n} \frac{1}{\sqrt{2\pi}\sigma_j} exp(-\frac{(x_j - \mu_j)^2}{2\sigma_j^2})$$

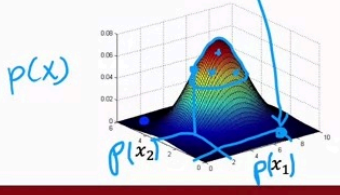
Anomaly if $p(x) < \varepsilon$

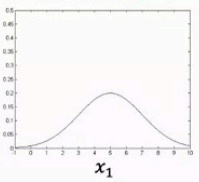


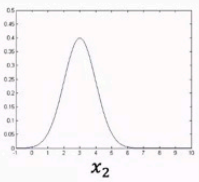
DeepLearning.AI Stanford ONLINE Andrew Ng

Anomaly detection example









$\mu_1 = 5, \sigma_1 = 2$
$p(x_1; \mu_1, \sigma_1^2)$

$\mu_2 = 3, \sigma_2 = 1$
$p(x_2; \mu_2, \sigma_2^2)$

$\varepsilon = 0.02$

$p\left(x_{test}^{(1)}\right) = 0.0426 \longrightarrow$ "ok"

$p\left(x_{test}^{(2)}\right) = 0.0021 \longrightarrow$ anomaly

DeepLearning.AI Stanford ONLINE Andrew Ng

Developing and evaluating an anomaly detection system

The importance of real-number evaluation

When developing a learning algorithm (choosing features, etc.),
making decisions is much easier if we have a way of evaluating our learning algorithm.

Assume we have some labeled data, of anomalous and non-anomalous examples.

$y = 1$ $y = 0$

Training set: $x^{(1)}, x^{(2)}, \ldots, x^{(m)}$ (assume normal examples/not anomalous)

$y = 0$ for all training examples

Cross validation set: $\left(x_{cv}^{(1)}, y_{cv}^{(1)}\right), \ldots, \left(x_{cv}^{(m_{cv})}, y_{cv}^{(m_{cv})}\right)$
Test set: $\left(x_{test}^{(1)}, y_{test}^{(1)}\right), \ldots, \left(x_{test}^{(m_{test})}, y_{test}^{(m_{test})}\right)$

include a few anomalous examples $y = 1$

mostly normal examples $y = 0$

Aircraft engines monitoring example

10000 good (normal) engines
20 flawed engines (anomalous)

2 to 50

2

$y = 0$ $y = 1$

Training set: 6000 good engines

train algorithm on training set

CV: 2000 good engines ($y = 0$) 10 anomalous ($y = 1$)

use cross validation set

tune ε tune x_j

Test: 2000 good engines ($y = 0$), 10 anomalous ($y = 1$)

Alternative: No test set

Use if very few labeled anomalous examples

higher risk of overfitting

Training set: 6000 good engines

2

CV: 4000 good engines ($y = 0$), 20 anomalous ($y = 1$)

tune ε tune x_j

Algorithm evaluation

course 2 week 3

skewed datasets

Fit model $p(x)$ on training set $x^{(1)}, x^{(2)}, \ldots, x^{(m)}$
On a cross validation/test example x, predict

$$y = \begin{cases} 1 & if\ p(x) < \varepsilon\ (anomaly) \\ 0 & if\ p(x) \geq \varepsilon\ (normal) \end{cases}$$

10

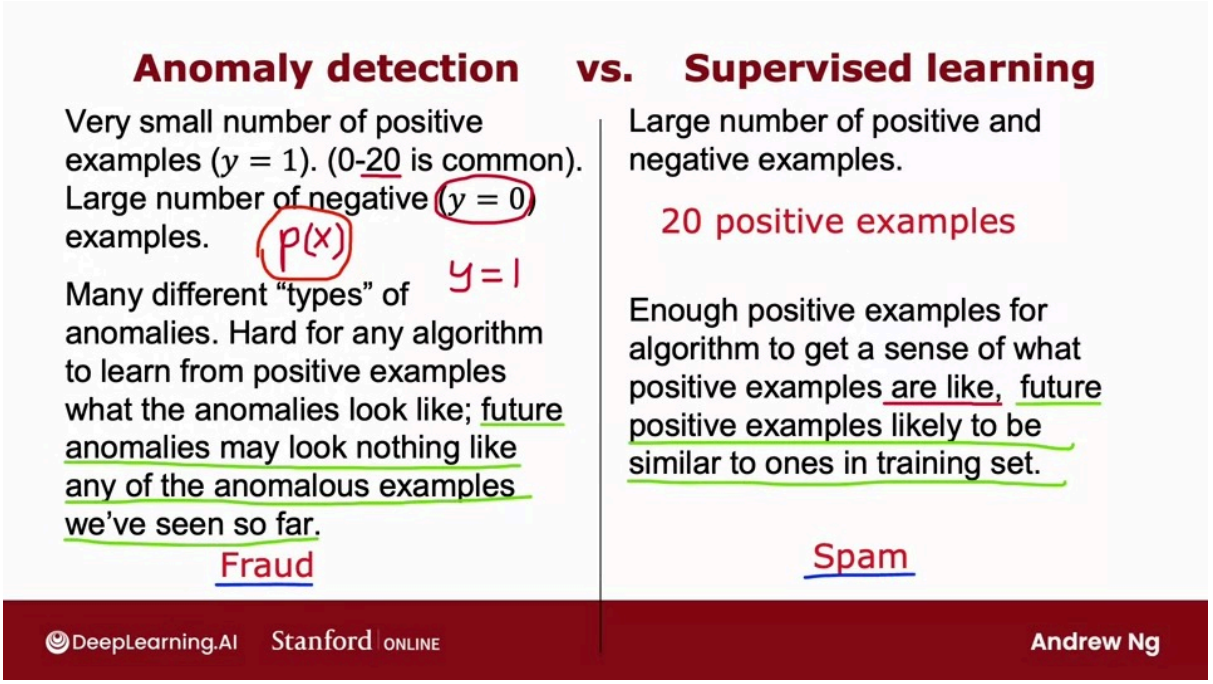
2000

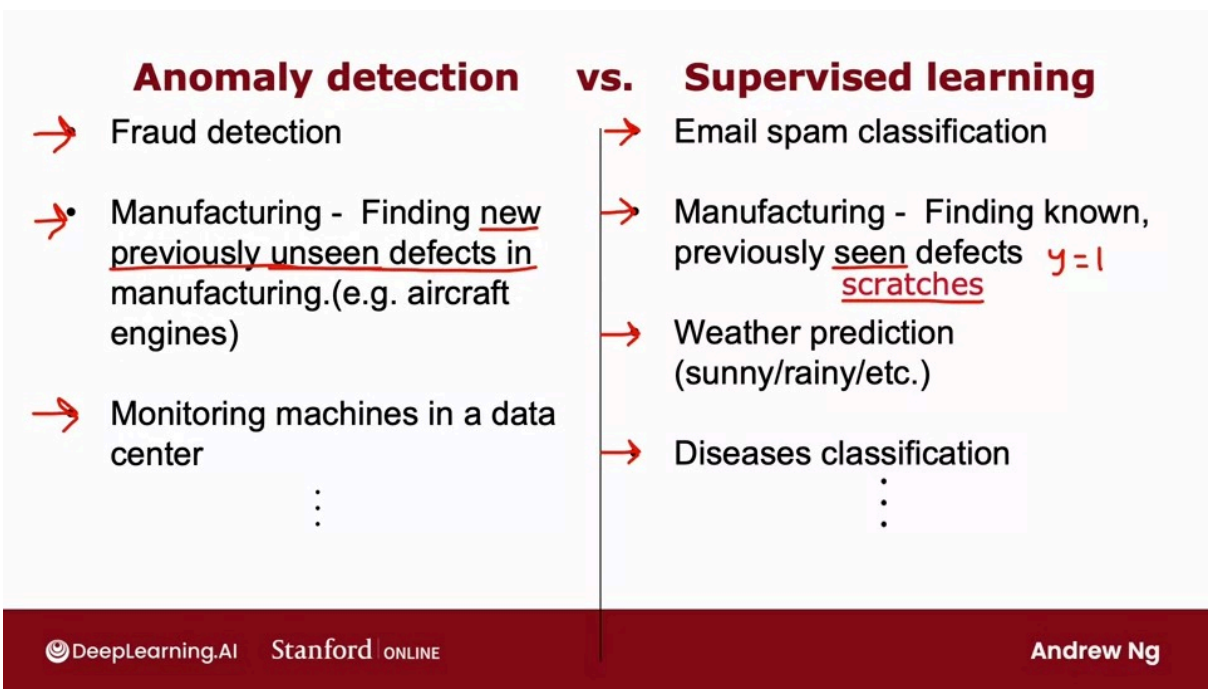
Possible evaluation metrics:

- True positive, false positive, false negative, true negative
- Precision/Recall
- F_1-score

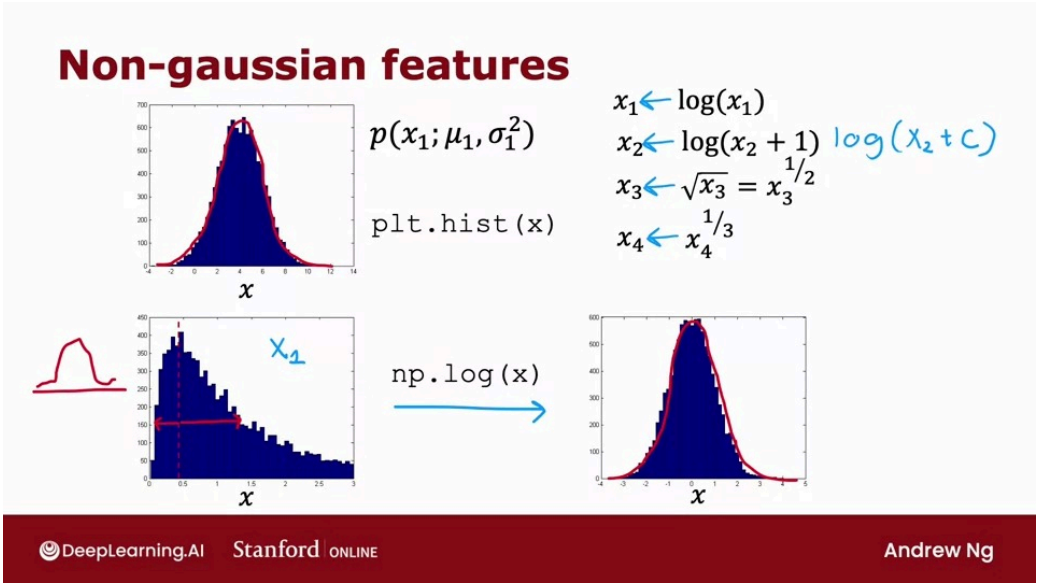
Use cross validation set

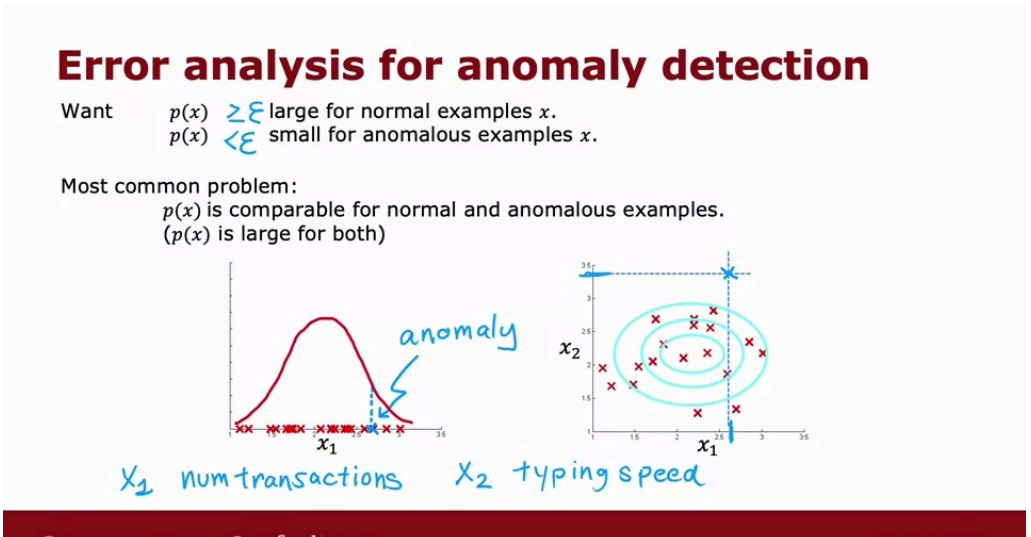
Anomaly detection vs. supervised learning

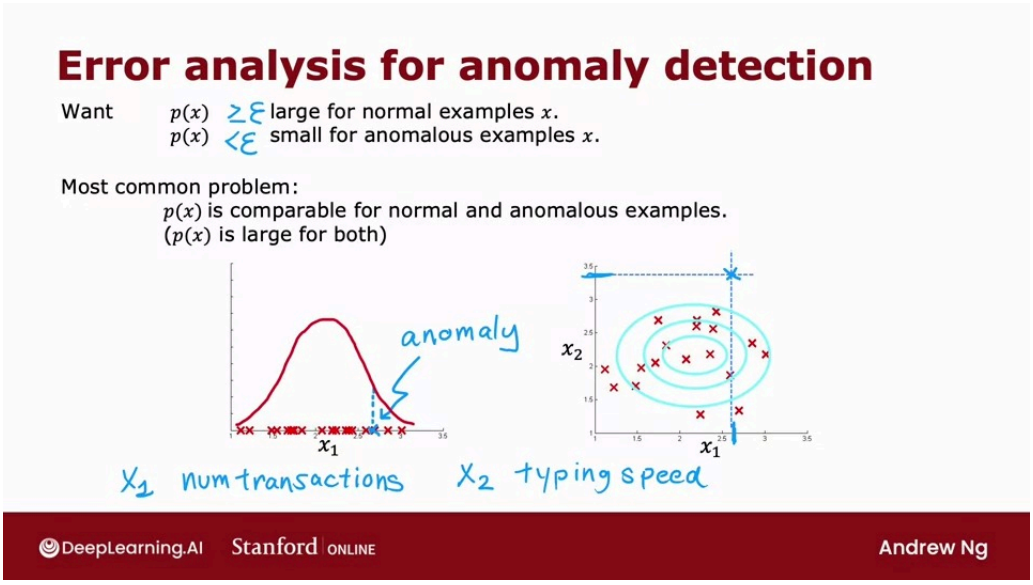




Choosing what features to use



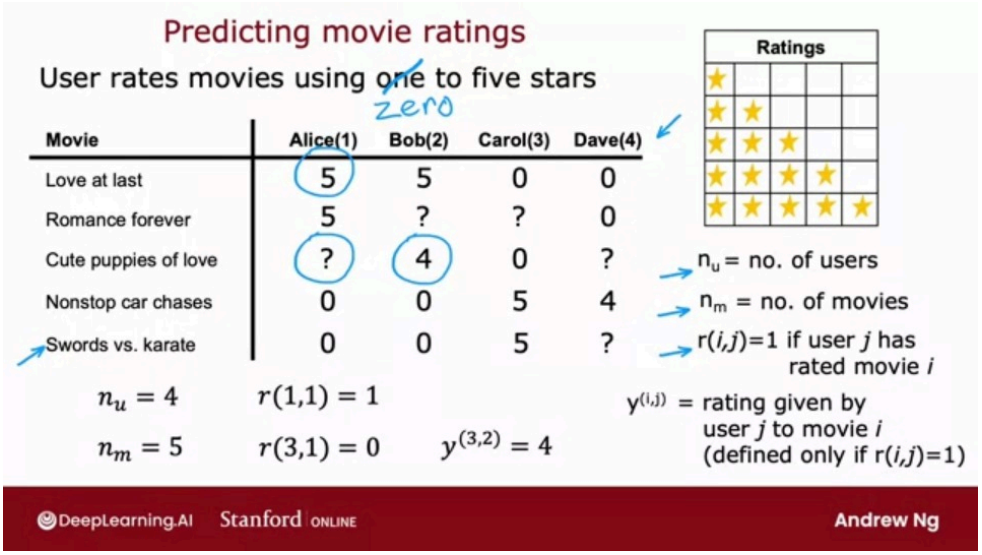




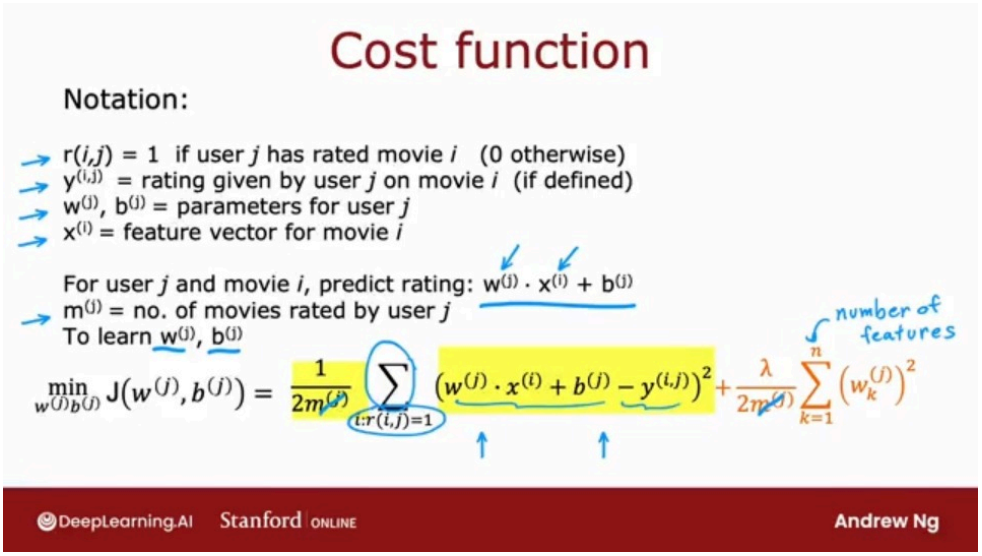
Week 2

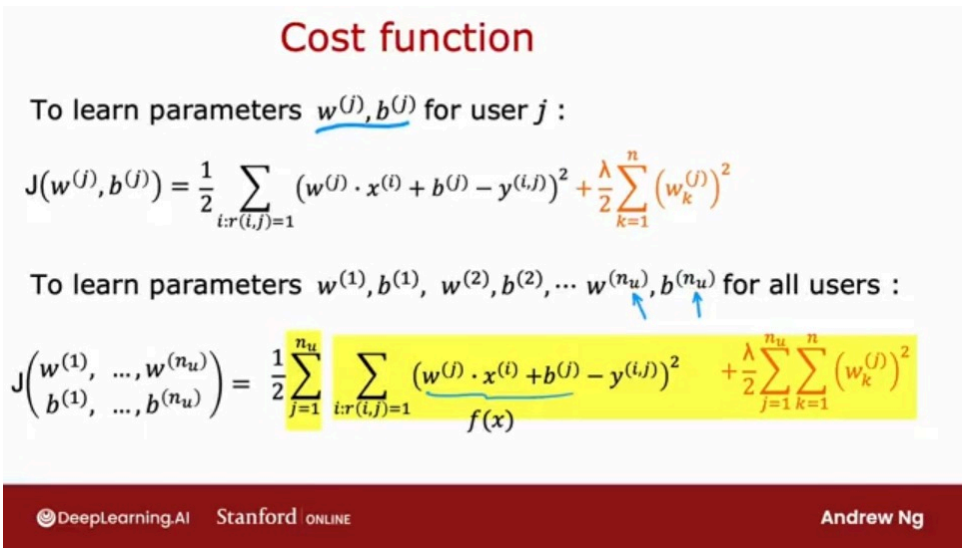
Collaborative filtering

Making recommendations

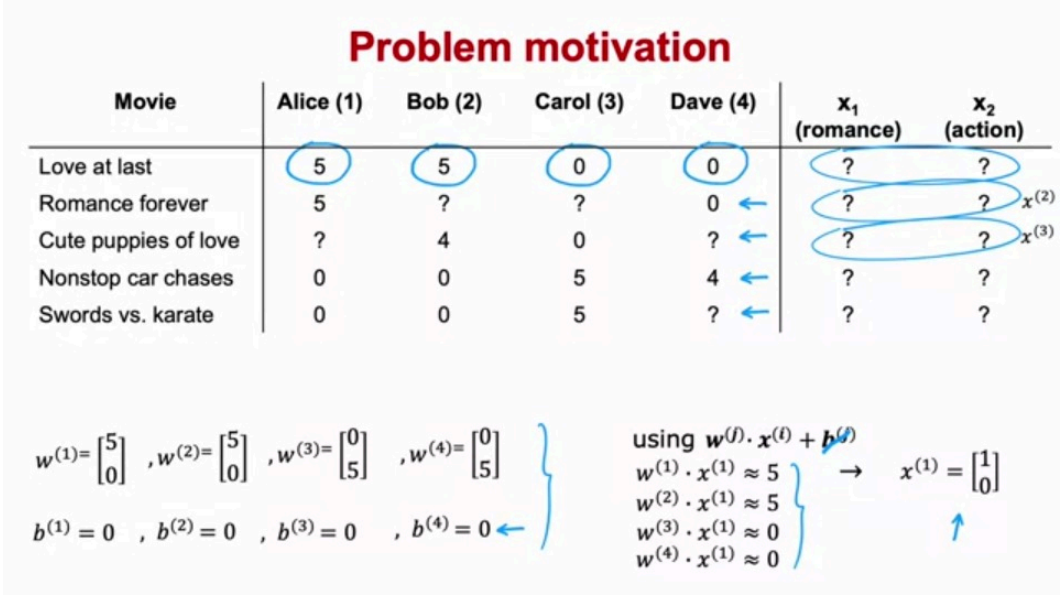


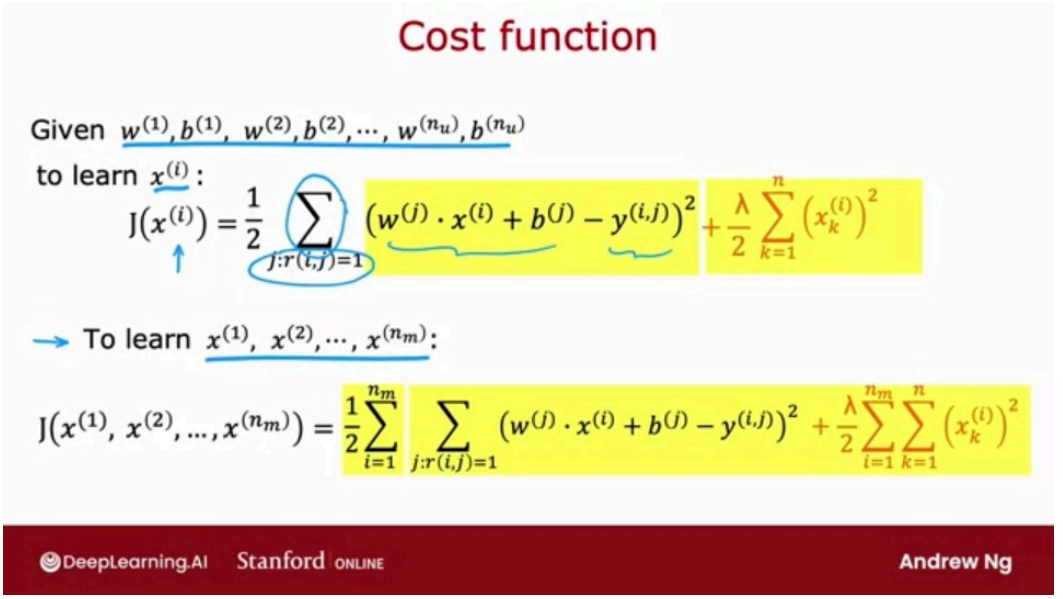
Using per-item features

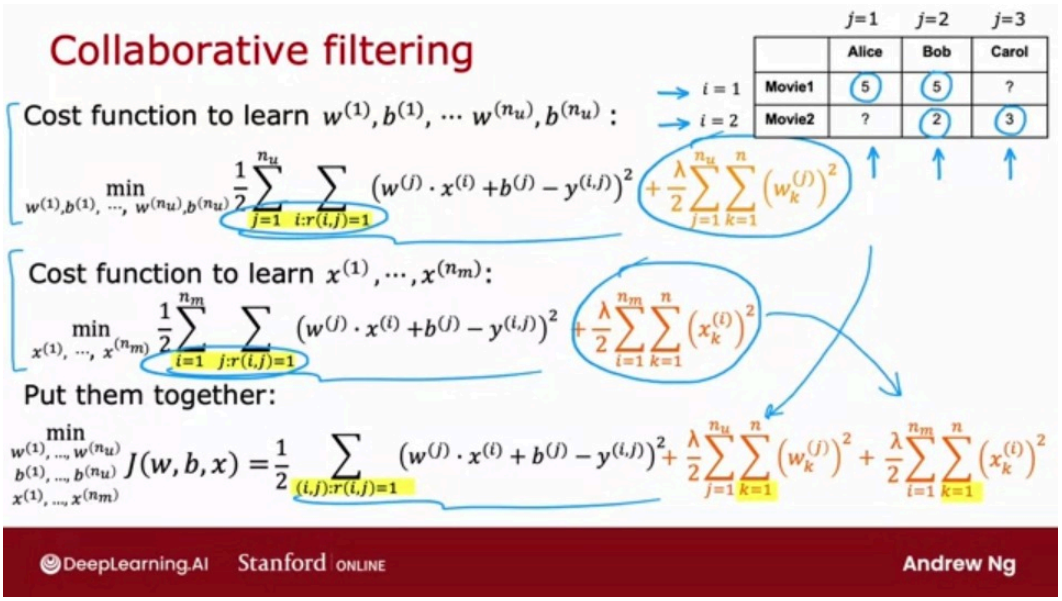




Collaborative filtering algorithm







Binary labels: favs, likes and clicks

Binary labels

Movie	Alice(1)	Bob(2)	Carol(3)	Dave(4)
Love at last	1	1	0	0
Romance forever	1	?	?	0
Cute puppies of love	?	1	0	?
Nonstop car chases	0	0	1	1
Swords vs. karate	0	0	1	?

1
0
?

DeepLearning.AI Stanford ONLINE Andrew Ng

Example applications

1. Did user j purchase an item after being shown? 1, 0, ?
2. Did user j fav/like an item? 1, 0, ?
3. Did user j spend at least 30sec with an item? 1, 0, ?
4. Did user j click on an item? 1, 0, ?

Meaning of ratings:
- 1 - engaged after being shown item
- 0 - did not engage after being shown item
- ? - item not yet shown

DeepLearning.AI Stanford ONLINE Andrew Ng

From regression to binary classification

- Previously:
  - Predict $y^{(i,j)}$ as $w^{(j)} \cdot x^{(i)} + b^{(j)}$
- For binary labels:
  - Predict that the probability of $y^{(i,j)} = 1$ is given by $g(w^{(j)} \cdot x^{(i)} + b^{(j)})$

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

DeepLearning.AI Stanford ONLINE Andrew Ng

Cost function for binary application

Previous cost function:

$$\frac{1}{2}\sum_{(i,j):r(i,j)=1}\underbrace{\left(w^{(j)}\cdot x^{(i)}+b^{(j)}\right)}_{f(x)}-y^{(i,j)}\Big)^2+\frac{\lambda}{2}\sum_{i=1}^{n_m}\sum_{k=1}^{n}\left(x_k^{(i)}\right)^2+\frac{\lambda}{2}\sum_{j=1}^{n_u}\sum_{k=1}^{n}\left(w_k^{(j)}\right)^2$$

Loss for binary labels $y^{(i,j)}$: $f_{(w,b,x)}(x)=g(w^{(j)}\cdot x^{(i)}+b^{(j)})$

$$L\left(f_{(w,b,x)}(x),y^{(i,j)}\right)=-y^{(i,j)}\log\left(f_{(w,b,x)}(x)\right)-\left(1-y^{(i,j)}\right)\log\left(1-f_{(w,b,x)}(x)\right)$$ ← Loss for single example

$$J(w,b,x)=\sum_{(i,j):r(i,j)=1}L\left(f_{(w,b,x)}(x),y^{(i,j)}\right)$$ ← cost for all examples

$g(w^{(j)}\cdot x^{(i)}+b^{(j)})$

DeepLearning.AI Stanford ONLINE Andrew Ng

Recommender systems implementation detail

Users who have not rated any movies

Movie	Alice(1)	Bob (2)	Carol (3)	Dave (4)	Eve (5)	
Love at last	5	5	0	0	?	0
Romance forever	5	?	?	0	?	0
Cute puppies of love	?	4	0	?	?	0
Nonstop car chases	0	0	5	4	?	0
Swords vs. karate	0	0	5	?	?	0

$$\begin{bmatrix}5&5&0&0&?\\5&?&?&0&?\\?&4&0&?&?\\0&0&5&4&?\\0&0&5&0&?\end{bmatrix}$$

$$\min_{\substack{w^{(1)},\dots w^{(n_u)}\\ b^{(1)},\dots b^{(n_u)}\\ x^{(1)},\dots x^{(n_m)}}}\frac{1}{2}\sum_{(i,j):r(i,j)=1}\left(w^{(j)}\cdot x^{(i)}+b^{(j)}-y^{(i,j)}\right)^2+\frac{\lambda}{2}\sum_{j=1}^{n_u}\sum_{k=1}^{n}\left(w_k^{(j)}\right)^2+\frac{\lambda}{2}\sum_{i=1}^{n_m}\sum_{k=1}^{n}\left(x_k^{(i)}\right)^2$$

$w^{(5)}=\begin{bmatrix}0\\0\end{bmatrix}$ $b^{(5)}=0$ $w^{(5)}\cdot x^{(i)}+b^{(5)}$

Mean Normalization

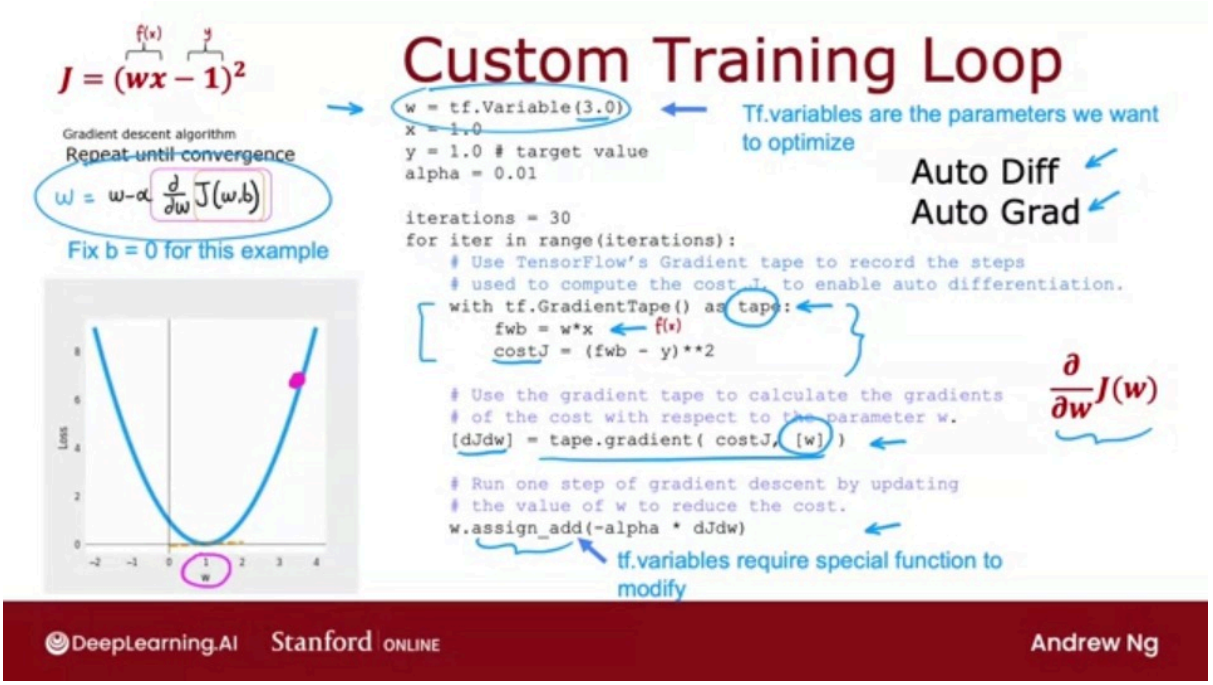
$$\begin{bmatrix}5&5&0&0&?\\5&?&?&0&?\\?&4&0&?&?\\0&0&5&4&?\\0&0&5&0&?\end{bmatrix}\begin{matrix}2.5\\2.5\\2\\2.25\\1.25\end{matrix}\qquad\mu=\begin{bmatrix}2.5\\2.5\\2\\2.25\\1.25\end{bmatrix}$$

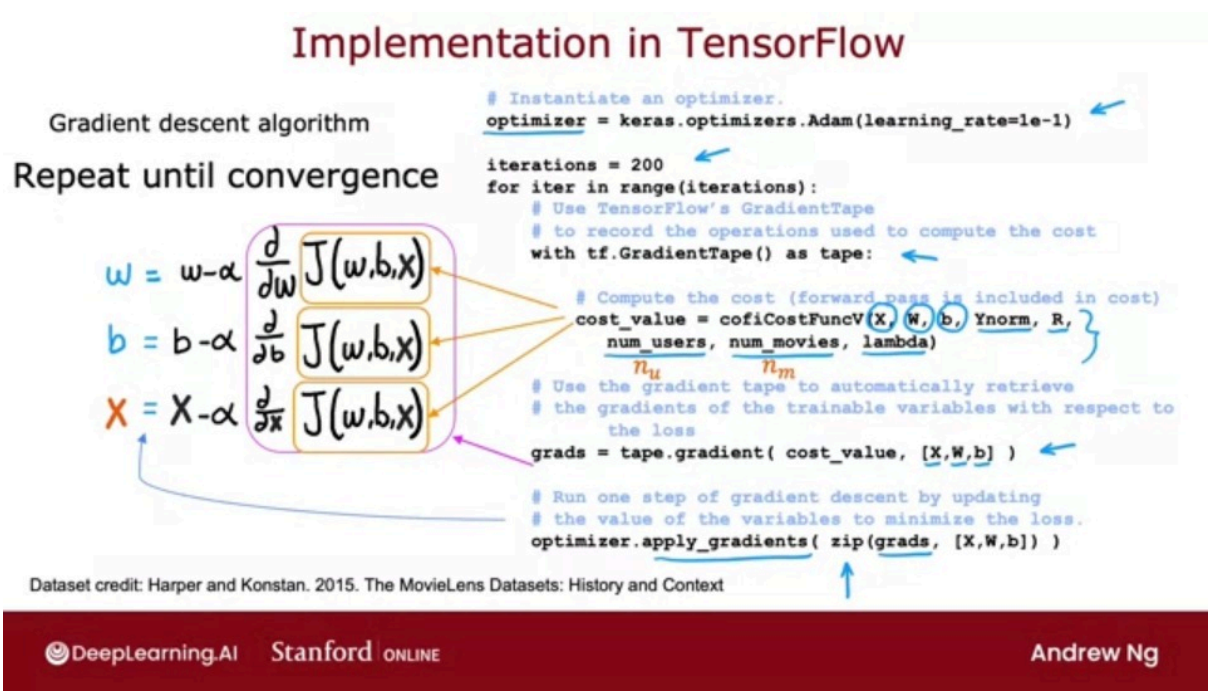
$$\begin{bmatrix}2.5&2.5&-2.5&-2.5&?\\2.5&?&?&-2.5&?\\?&2&-2&?&?\\-2.25&-2.25&2.75&1.75&?\\-1.25&-1.25&3.75&-1.25&?\end{bmatrix}$$

For user j, on movie i predict: $w^{(j)}\cdot x^{(i)}+b^{(j)}+\mu_i$ $\quad y^{(i,j)}$ $\quad w^{(j)}\cdot b^{(j)}\cdot x^{(i)}$

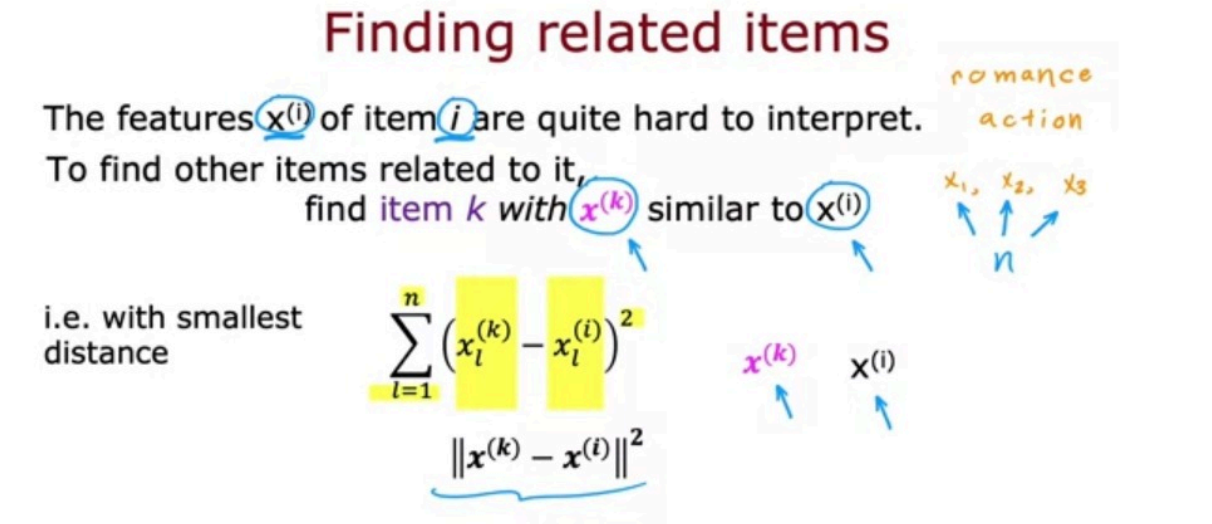
User 5 (Eve): $w^{(5)}=\begin{bmatrix}0\\0\end{bmatrix}$ $b^{(5)}=0$ $\underbrace{w^{(5)}\cdot x^{(i)}+b^{(5)}}_{0}+\mu_1=2.5$

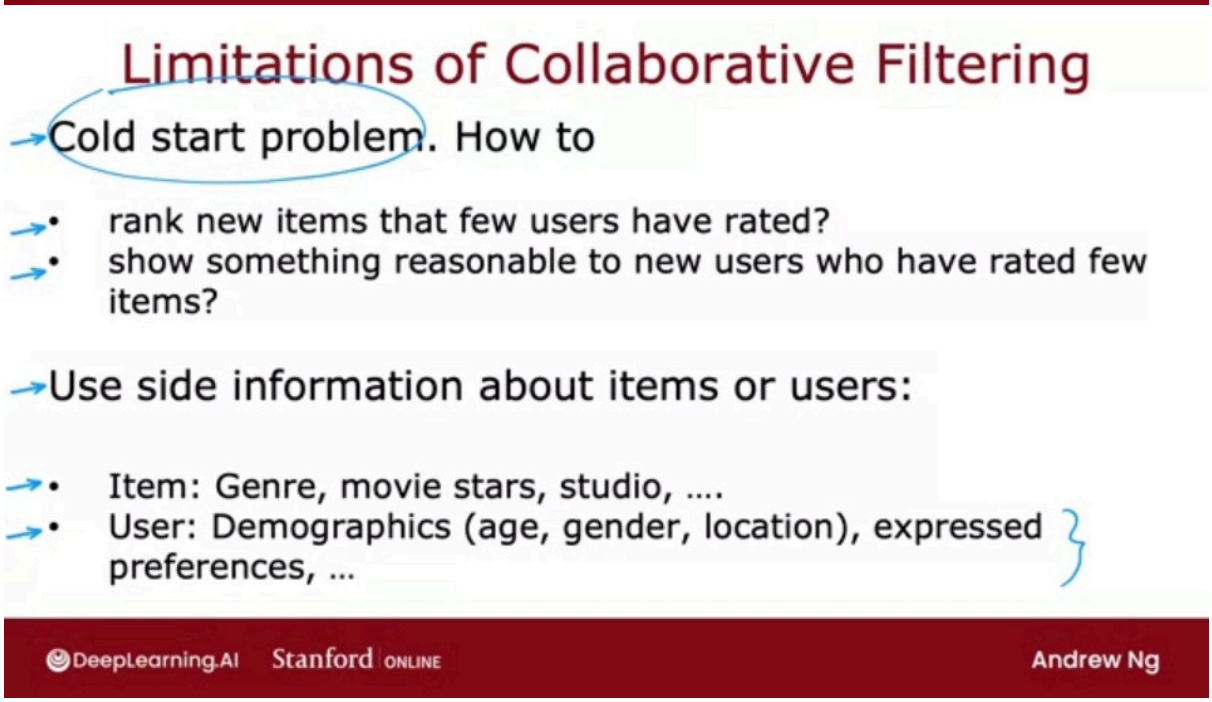
TensorFlow implementation of collaborative filtering



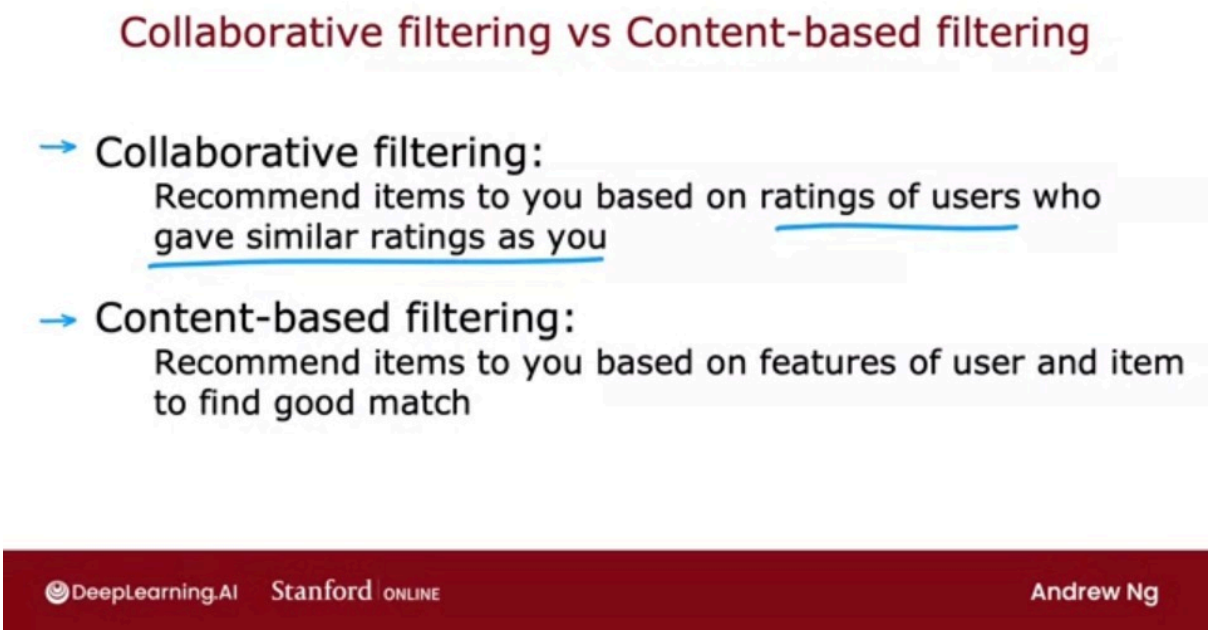


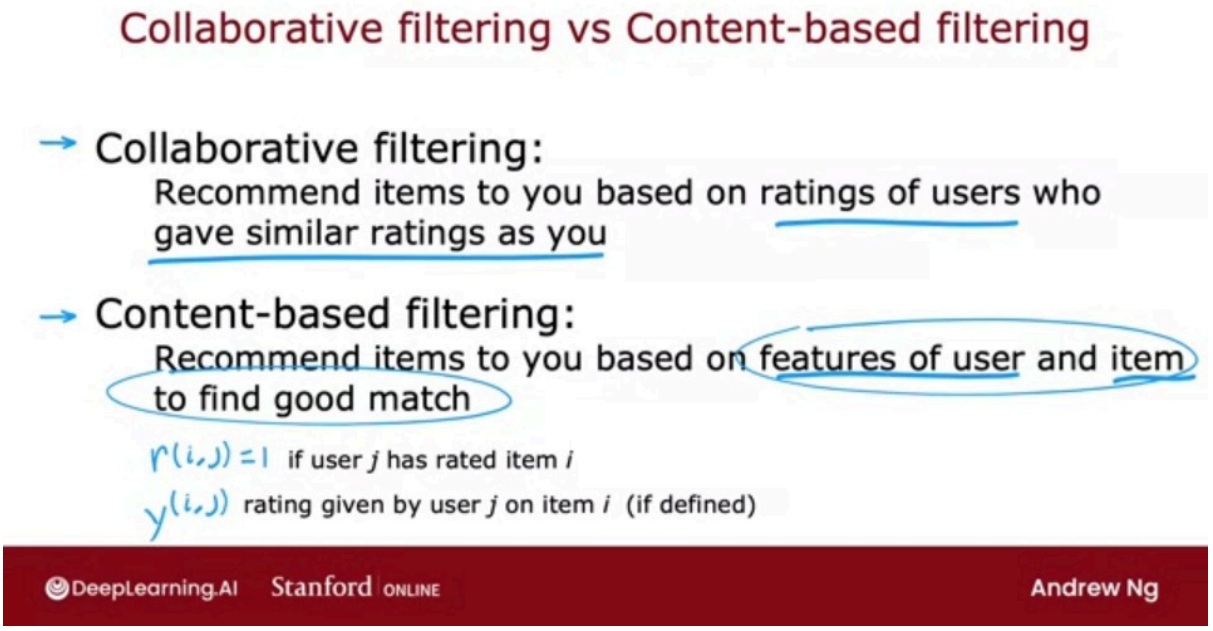
Finding related items





Collaborative filtering vs Content-based filtering





Examples of user and item features

User features:

- Age
- Gender (1 hot)
- Country (1 hot, 200)
- Movies watched (1000)
- Average rating per genre
- ...

Movie features:

- Year
- Genre/Genres
- Reviews
- Average rating
- ...

$x_u^{(j)}$ for user j

$x_m^{(i)}$ for movie i

Vector size could be different

Content-based filtering: Learning to match

Predict rating of user *j* on movie *i* as

$w^{(j)} \cdot x^{(i)} + b^{(j)}$

$v_u^{(j)} \cdot v_m^{(i)}$

computed from $x_u^{(j)}$

computed from $x_m^{(i)}$

$x_u^{(j)}$ → $v_u^{(j)}$ = user's preferences

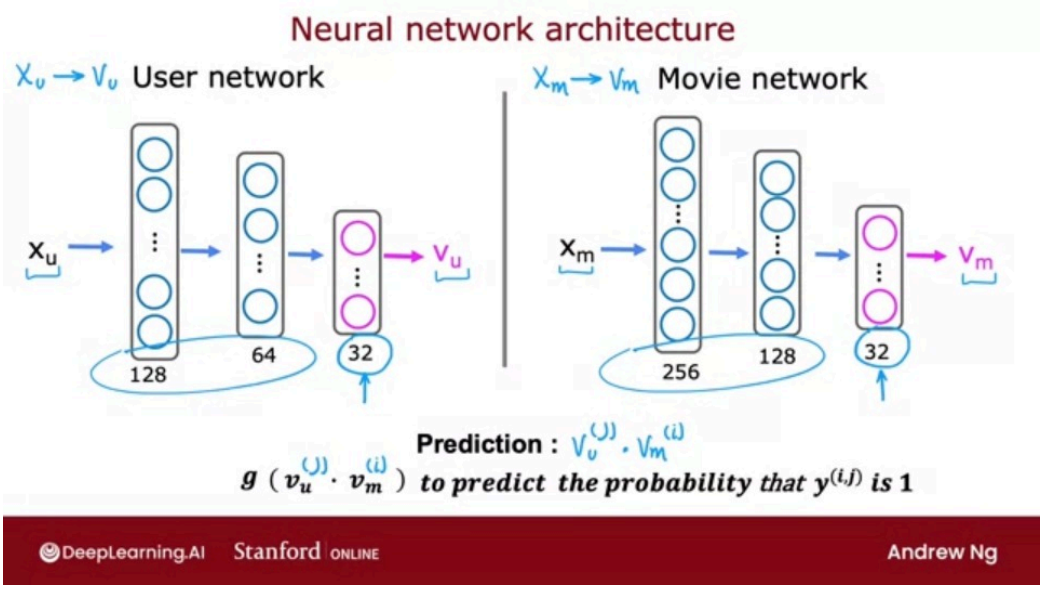
$$\begin{bmatrix} 4.9 \\ 0.1 \\ \vdots \\ 3.0 \end{bmatrix}$$ likes, likes

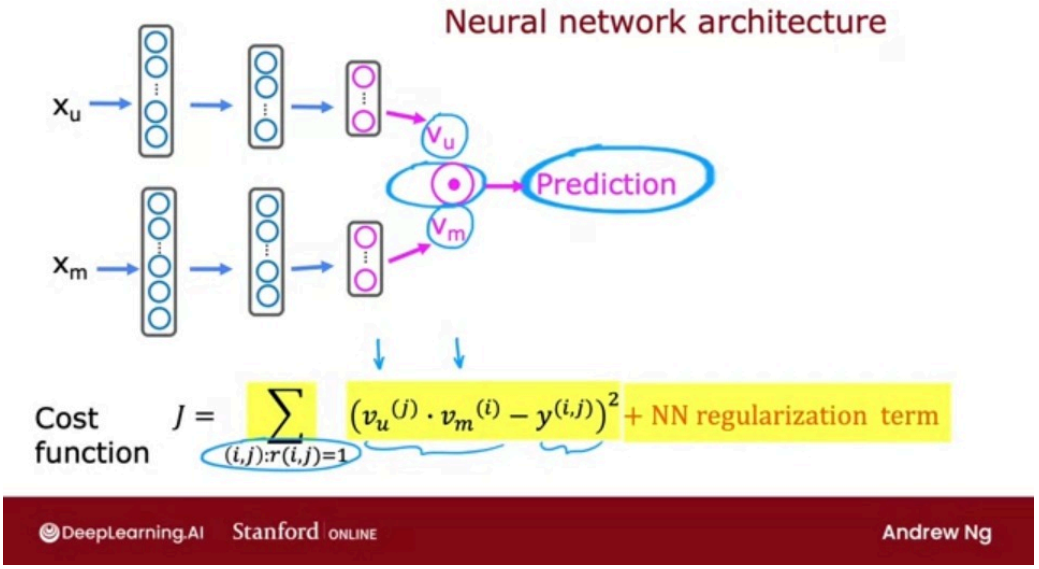
32

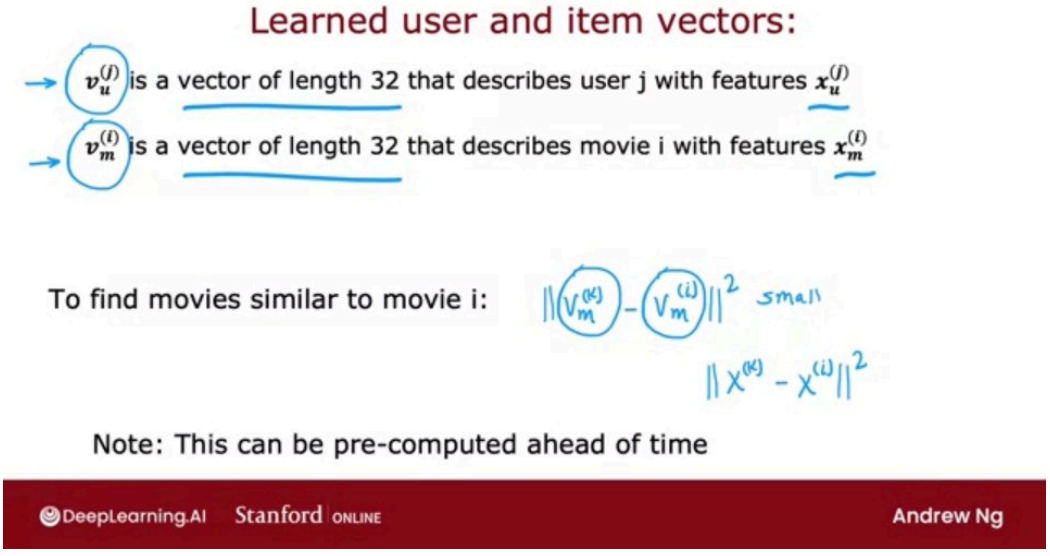
$x_m^{(i)}$ → $v_m^{(i)}$ = movie features

$$\begin{bmatrix} 4.5 \\ 0.2 \\ \vdots \\ 3.5 \end{bmatrix}$$ romance, action

Deep learning for content-based filtering







Recommending from a large catalogue

Two steps: Retrieval & Ranking

Retrieval:

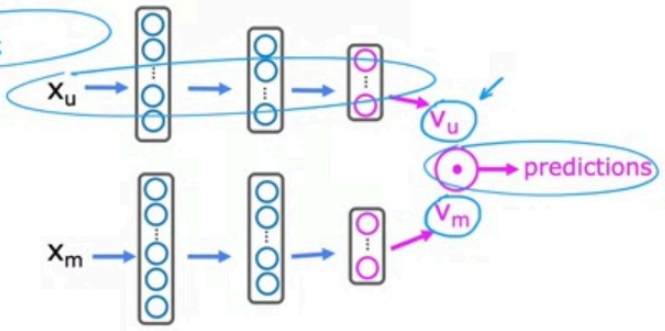
- Generate large list of plausible item candidates ~100s
  e.g.
  1) For each of the last 10 movies watched by the user, find 10 most similar movies
  $\| v_m^{(x)} - v_m^{(i)} \|^2$
  2) For most viewed 3 genres, find the top 10 movies
  3) Top 20 movies in the country
- Combine retrieved items into list, removing duplicates and items already watched/purchased

DeepLearning.AI Stanford ONLINE Andrew Ng

Two steps: Retrieval & ranking

Ranking: ~100s

- Take list retrieved and rank using learned model



- Display ranked items to user

DeepLearning.AI Stanford ONLINE Andrew Ng

Retrieval step

- Retrieving more items results in better performance, but slower recommendations.
- To analyse/optimize the trade-off, carry out offline experiments to see if retrieving additional items results in more relevant recommendations (i.e., $p(y^{(i,j)}) = 1$ of items displayed to user are higher).

100 500

DeepLearning.AI Stanford ONLINE Andrew Ng

Ethical use of recommender systems

What is the goal of the recommender system?

Recommend:

- Movies most likely to be rated 5 stars by user
- Products most likely to be purchased
- Ads most likely to be clicked on +high bid
- Products generating the largest profit
- Video leading to maximum watch time

DeepLearning.AI Stanford ONLINE Andrew Ng

Ethical considerations with recommender systems

Travel industry

Good travel experience to more users

Bid higher for ads → More profitable

Payday loans

Squeeze customers more

Bid higher for ads → More profit

DeepLearning.AI Stanford ONLINE Andrew Ng

Other problematic cases:

- Maximizing user engagement (e.g. watch time) has led to large social media/video sharing sites to amplify conspiracy theories and hate/toxicity

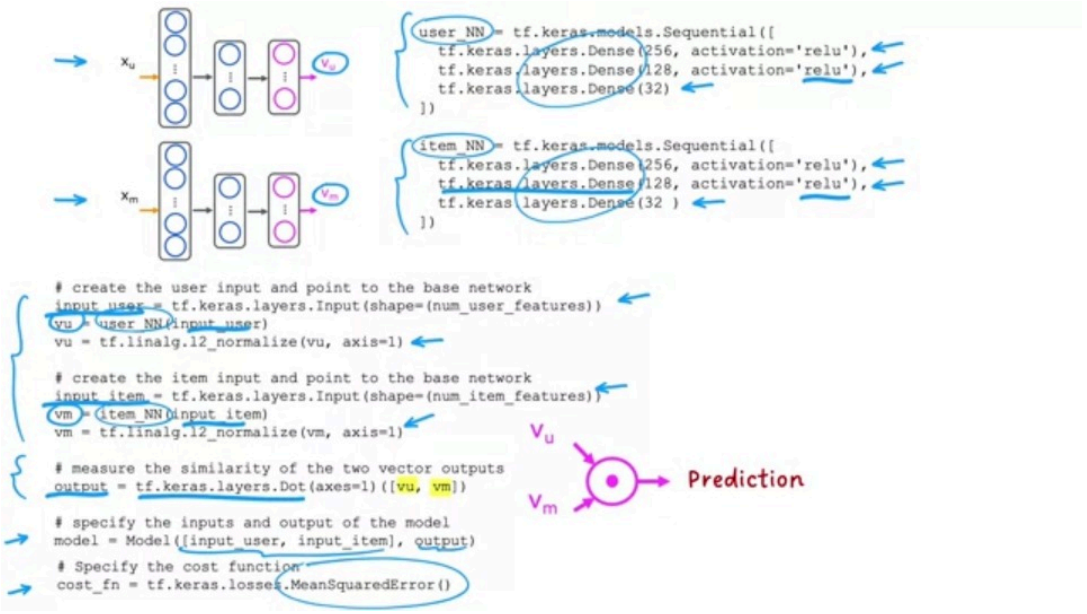
  → Amelioration : Filter out problematic content such as hate speech, fraud, scams and violent content

- Can a ranking system maximize your profit rather than users' welfare be presented in a transparent way?

  → Amelioration : Be transparent with users

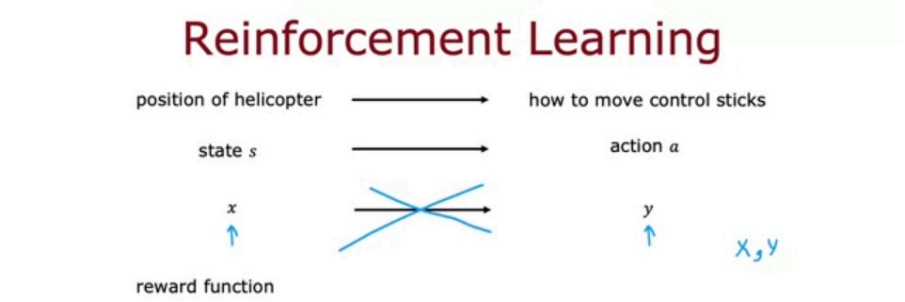
DeepLearning.AI Stanford ONLINE Andrew Ng

TensorFlow implementation of content-based filtering

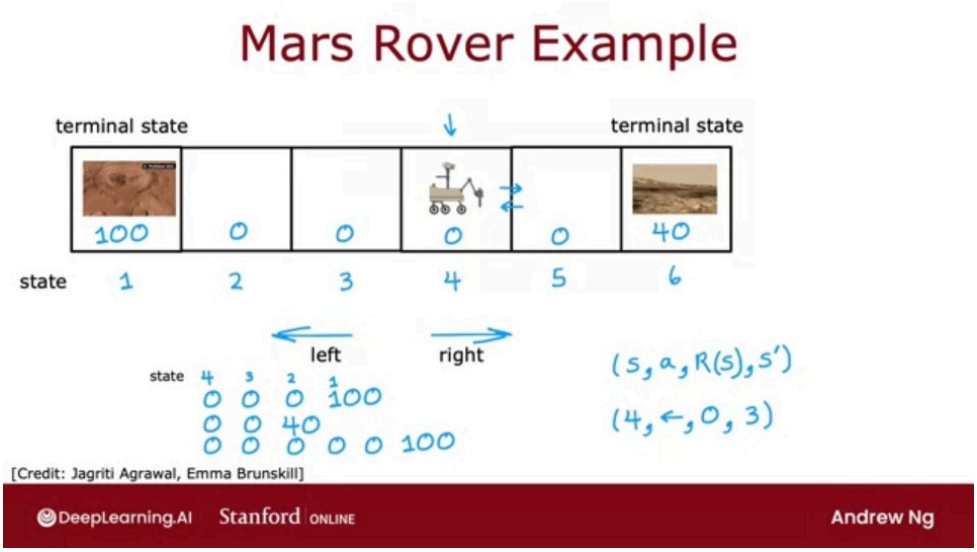


Week 3

Reinforcement learning introduction

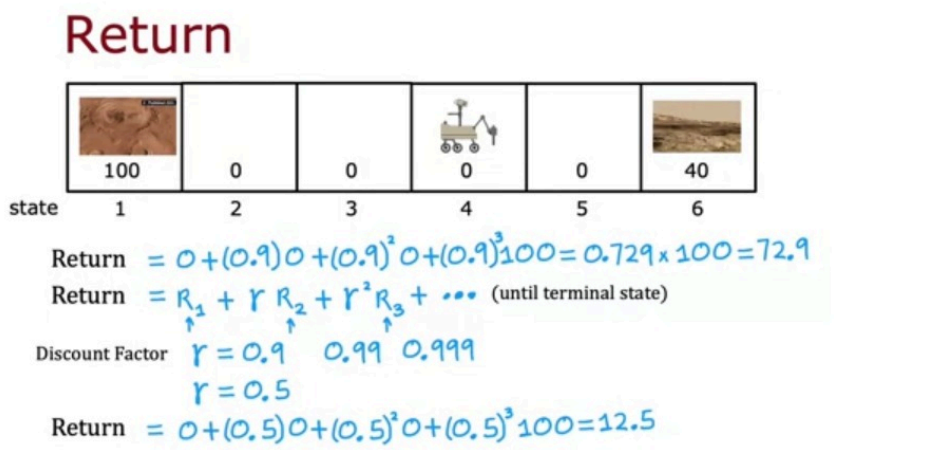


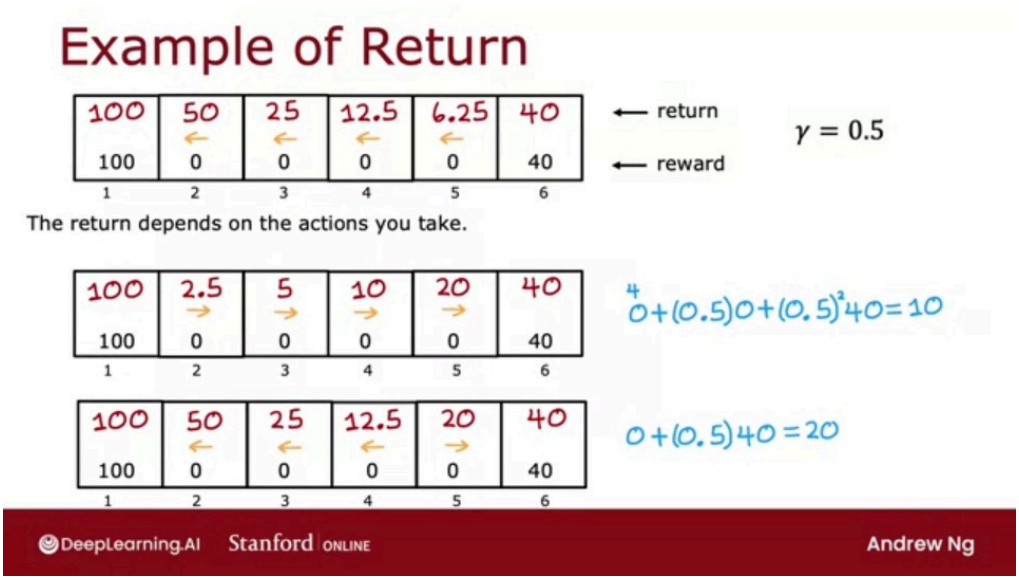
Mars rover example



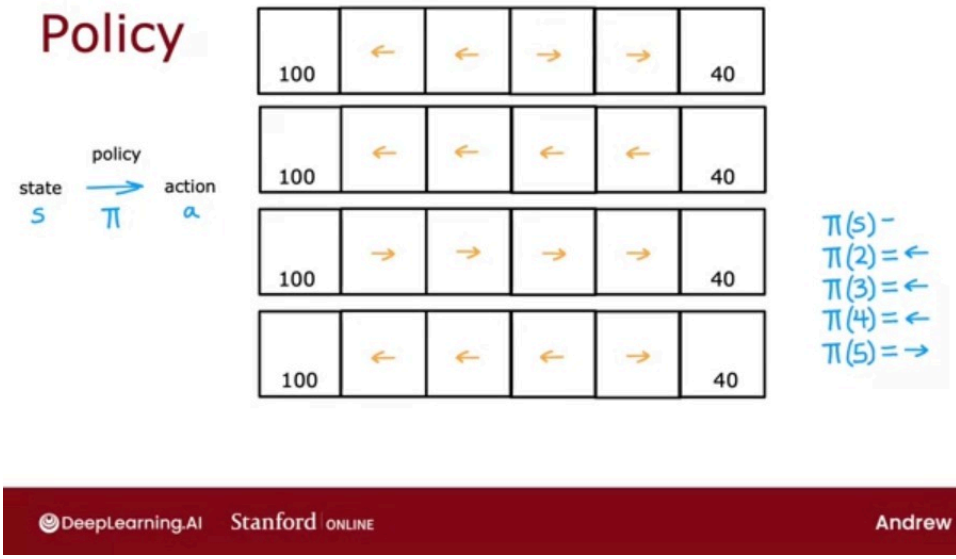
[Credit: Jagriti Agrawal, Emma Brunskill]

The Return in reinforcement learning

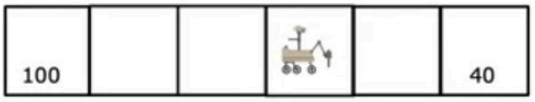




Policies in reinforcement learning



The goal of reinforcement learning

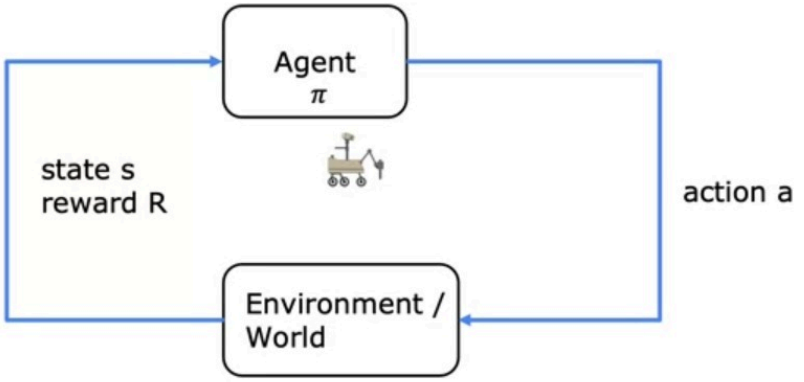


Find a policy π that tells you what action (a = $\pi(s)$) to take in every state (s) so as to maximize the return.

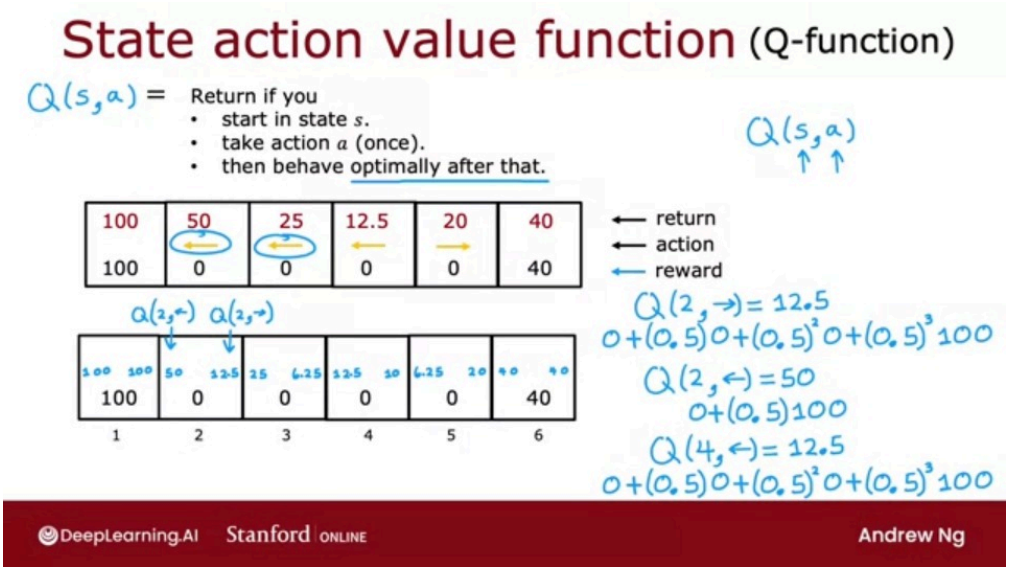
Review of key concepts

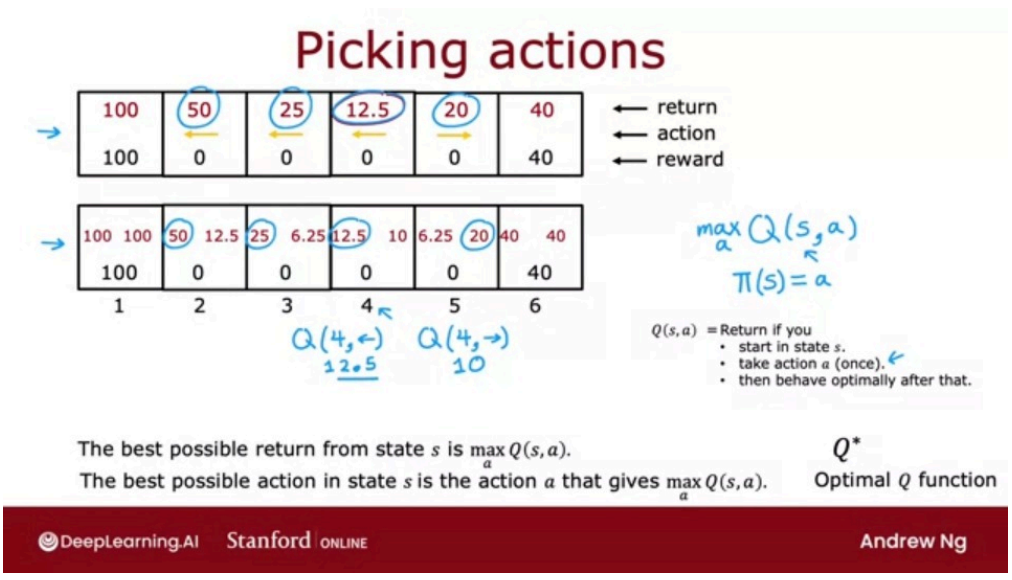
	Mars rover 	Helicopter 	Chess 
states	6 states	position of helicopter	pieces on board
actions	← →	how to move control stick	possible move
rewards	100, 0, 40	+1, −1000	+1, 0, −1
discount factor γ	0.5	0.99	0.995
return	$R_1 + \gamma R_2 + \gamma^2 R_3 + \cdots$	$R_1 + \gamma R_2 + \gamma^2 R_3 + \cdots$	$R_1 + \gamma R_2 + \gamma^2 R_3 + \cdots$
policy π		Find $\pi(s) = a$	Find $\pi(s) = a$

Markov Decision Process (MDP)

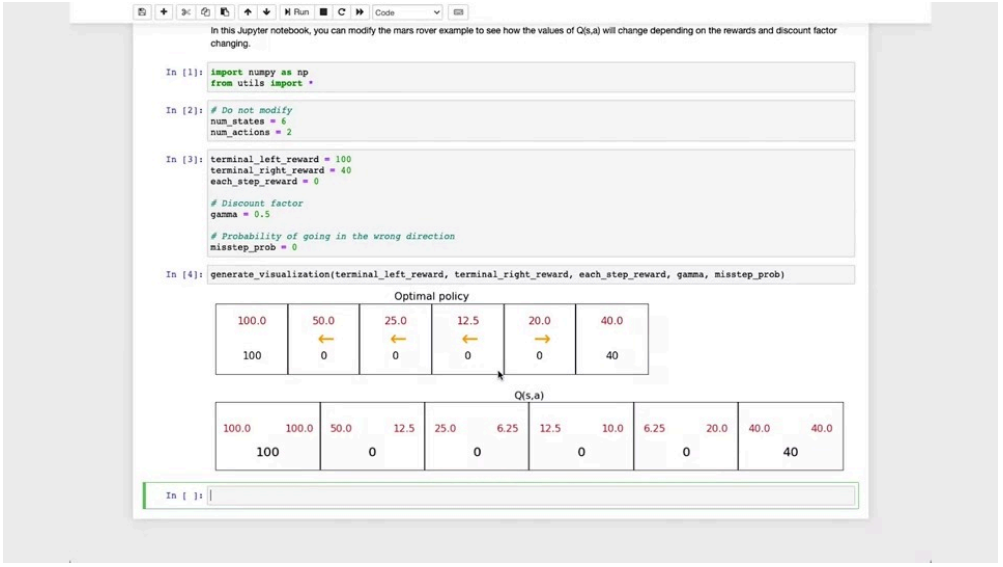


State-action value function definition





State-action value function example



State Action Value Function Example

In this Jupyter notebook, you can modify the mars rover example to see how the values of Q(s,a) will change depending on the rewards and discount factor changing.

In [1]:
```
import numpy as np
from utils import *
```

In [2]:
```
# Do not modify
num_states = 6
num_actions = 2
```

In [5]:
```
terminal_left_reward = 100
terminal_right_reward = 10
each_step_reward = 0

# Discount factor
gamma = 0.5

# Probability of going in the wrong direction
misstep_prob = 0
```

In [6]:
```
generate_visualization(terminal_left_reward, terminal_right_reward, each_step_reward, gamma, misstep_prob)
```

Optimal policy

100.0	50.0	25.0	12.5	6.25	10.0
100	← 0	← 0	← 0	← 0	10

Q(s,a)

100.0	100.0	50.0	12.5	25.0	6.25	12.5	3.12	6.25	5.0	10.0	10.0
100		0		0		0		0		10	

In [ ]:

State Action Value Function Example

In this Jupyter notebook, you can modify the mars rover example to see how the values of Q(s,a) will change depending on the rewards and discount factor changing.

In [1]:
```
import numpy as np
from utils import *
```

In [2]:
```
# Do not modify
num_states = 6
num_actions = 2
```


In [7]:
```
terminal_left_reward = 100
terminal_right_reward = 40
each_step_reward = 0

# Discount factor
gamma = 0.9

# Probability of going in the wrong direction
misstep_prob = 0
```

In [8]:
```
generate_visualization(terminal_left_reward, terminal_right_reward, each_step_reward, gamma, misstep_prob)
```

Optimal policy

100.0	90.0	81.0	72.9	65.61	40.0
100	← 0	← 0	← 0	← 0	40

Q(s,a)

100.0	100.0	90.0	72.9	81.0	65.61	72.9	59.05	65.61	36.0	40.0	40.0
100		0		0		0		0		40	

In [ ]:

State Action Value Function Example

In this Jupyter notebook, you can modify the mars rover example to see how the values of Q(s,a) will change depending on the rewards and discount factor changing.

In [1]:
```
import numpy as np
from utils import *
```

In [2]:
```
# Do not modify
num_states = 6
num_actions = 2
```


In [9]:
```
terminal_left_reward = 100
terminal_right_reward = 40
each_step_reward = 0

# Discount factor
gamma = 0.3

# Probability of going in the wrong direction
misstep_prob = 0
```

In [8]:
```
generate_visualization(terminal_left_reward, terminal_right_reward, each_step_reward, gamma, misstep_prob)
```

Optimal policy

100.0	90.0	81.0	72.9	65.61	40.0
100	← 0	← 0	← 0	← 0	40

Q(s,a)

100.0	100.0	90.0	72.9	81.0	65.61	72.9	59.05	65.61	36.0	40.0	40.0
100		0		0		0		0		40	

In [ ]:

Bellman Equation

Bellman Equation

$Q(s,a)$ = Return if you
- start in state s.
- take action a (once).
- then behave optimally after that.

$R(1)=100$ $R(2)=0$ ... $R(6)=40$

s : current state
a : current action
s' : state you get to after taking action a
a' : action that you take in state s'

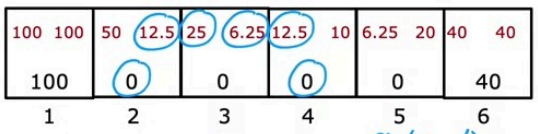
$R(s)$ = reward of current state

$$Q(s,a) = R(s) + \gamma \max_{a'} Q(s',a')$$

Bellman Equation

$$Q(s,a) = R(s) + \gamma \max_{a'} Q(s',a')$$

$Q(s,a) = R(s)$



$s=2$
$a=\rightarrow$
$s'=3$

$Q(2,\rightarrow) = R(2) + 0.5 \max_{a'} Q(3,a')$
$= 0 + (0.5)25 = 12.5$

$s=4$
$a=\leftarrow$
$s'=3$

$Q(4,\leftarrow) = R(4) + 0.5 \max_{a'} Q(3,a')$
$= 0 + (0.5)25 = 12.5$

Explanation of Bellman Equation

$Q(s,a)$ = Return if you
- start in state s.
- take action a (once).
- then behave optimally after that.

$s \rightarrow s'$

→ The best possible return from state s' is $\max_{a'} Q(s',a')$

$$Q(s,a) = R(s) + \gamma \max_{a'} Q(s',a')$$

Reward you get right away

Return from behaving optimally starting from state s'.

$R_1 + \gamma R_2 + \gamma^2 R_3 + \gamma^3 R_4 + \cdots$

$Q(s,a) = R_1 + \gamma [R_2 + \gamma R_3 + \gamma^2 R_4 + \cdots]$

Explanation of Bellman Equation

$$Q(s,a) = R(s) + \gamma \max_{a'} Q(s',a')$$

100	100	50	12.5	25	6.25	12.5	10	6.25	20	40	40
100		0		0		0		0		40	
1		2		3		4		5		6	

$Q(4,\leftarrow)$

$= 0 + (0.5)0 + (0.5)^2 0 + (0.5)^3 100$

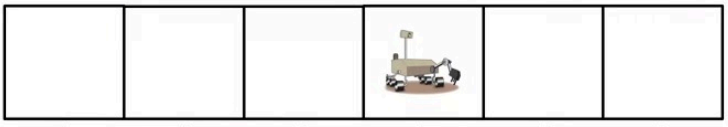
$= R(4) + (0.5)[0 + (0.5)0 + (0.5)^2 100]$

$= R(4) + (0.5) \max_{a'} Q(3,a')$

continuous state space applications

Discrete vs Continuous State

Discrete State:



1 2 3 4 5 6

Continuous State:

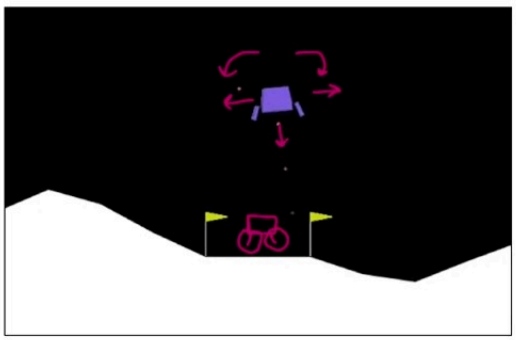
$$s = \begin{bmatrix} x \\ y \\ \theta \\ \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix}$$

Autonomous Helicopter

$$s = \begin{bmatrix} x \\ y \\ z \\ \phi \\ \theta \\ \omega \\ \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\omega} \end{bmatrix}$$

Lunar lander

Lunar Lander



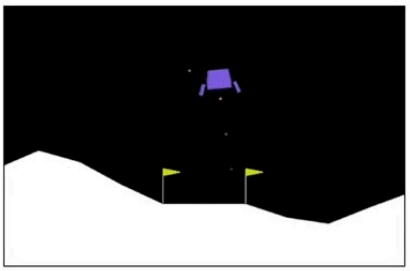
actions:
- do nothing
- left thruster
- main thruster
- right thruster

$s = \begin{bmatrix} x \\ y \\ \dot{x} \\ \dot{y} \\ \theta \\ \dot{\theta} \\ l \\ r \end{bmatrix}$ 0 or 1

DeepLearning.AI Stanford ONLINE Andrew Ng

Lunar Lander Problem

Learn a policy π that, given $s = \begin{bmatrix} x \\ y \\ \dot{x} \\ \dot{y} \\ \theta \\ \dot{\theta} \\ l \\ r \end{bmatrix}$



picks action $a = \pi(s)$ so as to maximize the return.

$\gamma = 0.985$

DeepLearning.AI Stanford ONLINE Andrew Ng

Learning the state-value function

Deep Reinforcement Learning

$\vec{x} = \begin{bmatrix} s \\ a \end{bmatrix} \begin{bmatrix} x \\ y \\ \theta \\ \vdots \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$ → [⋮] → [⋮] → [ ] → $Q(s,a)$

$x \to y$

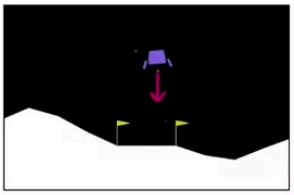
y

12 inputs 64 units 64 units 1 unit

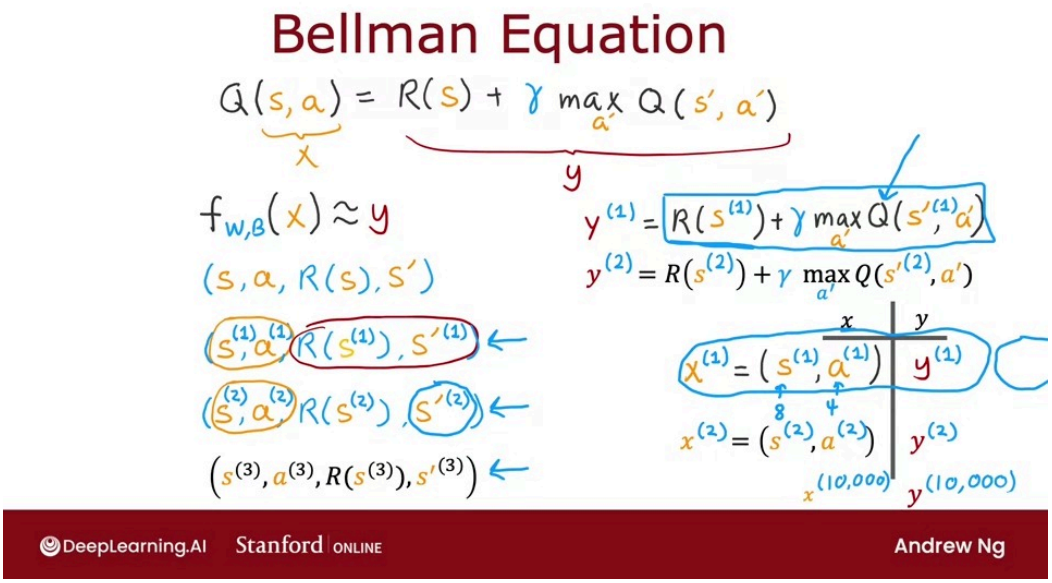
In a state s, use neural network to compute

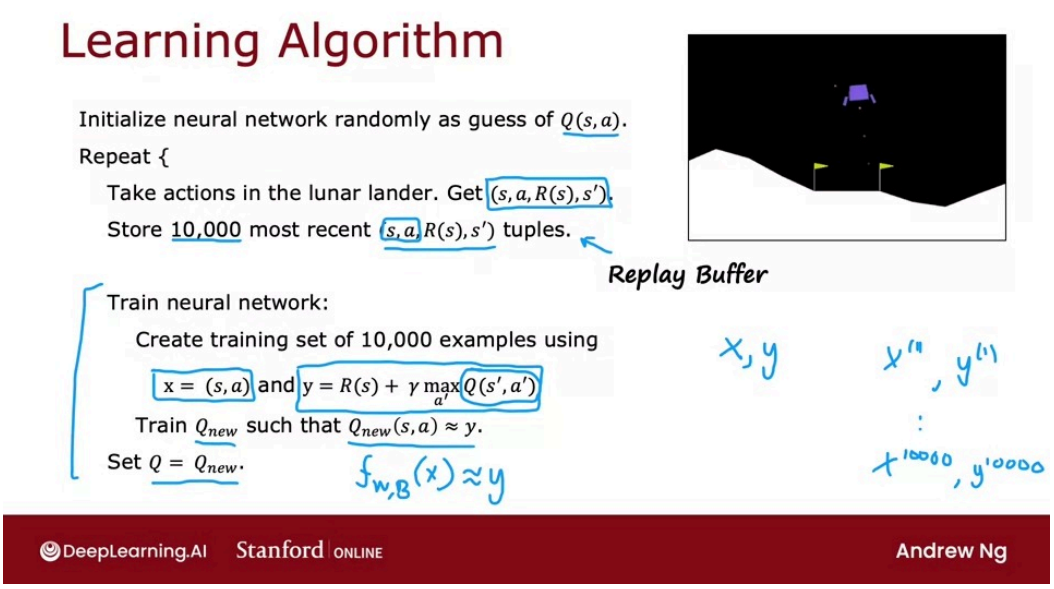
$Q(s, \text{nothing})$, $Q(s, \text{left})$, $Q(s, \text{main})$ $Q(s, \text{right})$

Pick the action a that maximizes $Q(s,a)$

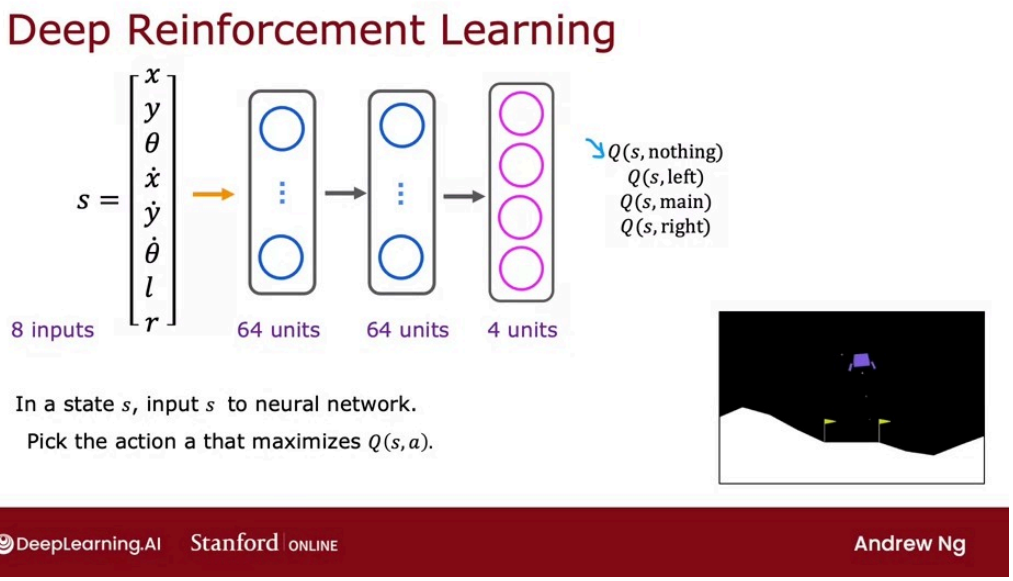


DeepLearning.AI Stanford ONLINE Andrew Ng





Algorithm refinement: Improved neural network architecture



Algorithm refinement: ϵ-greedy policy

How to choose actions while still learning?

In some state s

Option 1:

Pick the action a that maximizes $Q(s,a)$.

Q(s,main) is low

↑

a

Option 2:

With probability 0.95, pick the action a that maximizes $Q(s,a)$. Greedy, "Exploitation"

With probability 0.05, pick an action a randomly. "Exploration"

ε–greedy policy ($\varepsilon = 0.05$)

0.95

Start ε high

1.0 → 0.01

Gradually decrease

state of reinforcement learning

Limitations of Reinforcement Learning

- Much easier to get to work in a simulation than a real robot!
- Far fewer applications than supervised and unsupervised learning.
- But … exciting research direction with potential for future applications.