Clustering Algorithm

K-means algorithm

Randomly initialize K cluster centroids $\mu_1, \mu_2, \ldots, \mu_K$

Repeat {

 # Assign points to cluster centroids

for i = 1 to m

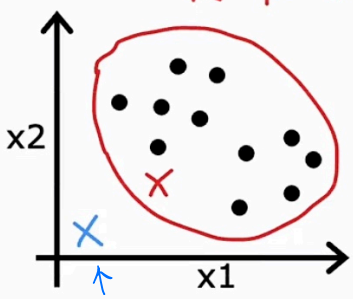
$c^{(i)}$:= index (from 1 to K) of cluster centroid closest to $x^{(i)}$

 # Move cluster centroids

for k = 1 to K

μ_k := average (mean) of points assigned to cluster k

}



但在运行k-means时，如果没有为某簇分配点，消除该簇的做法实际上更常见

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

Random initialization

防止陷入局部极小
因初始质心选择不当而

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)

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

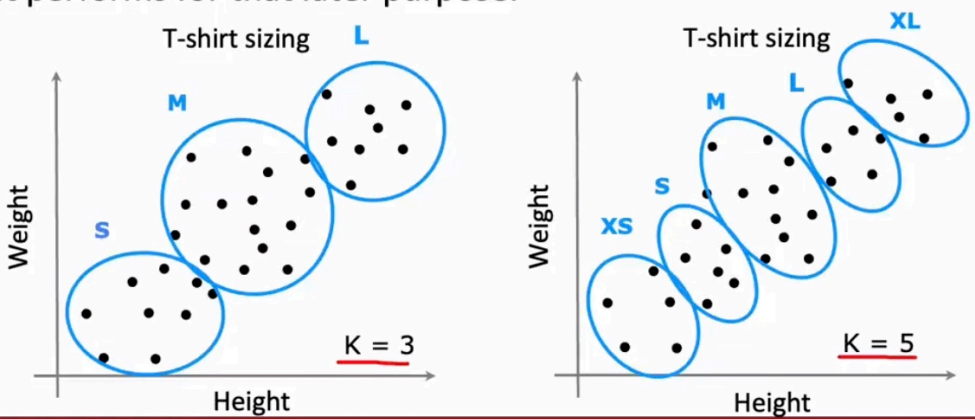
}

Pick set of clusters that gave lowest cost J

Choosing the value of K

Sometimes, you're running K-means to get clusters to use for some later/downstream purpose. Evaluate K-means based on a metric for how well it performs for that later purpose.

E.g.



Anomaly detection example

Aircraft engine features:

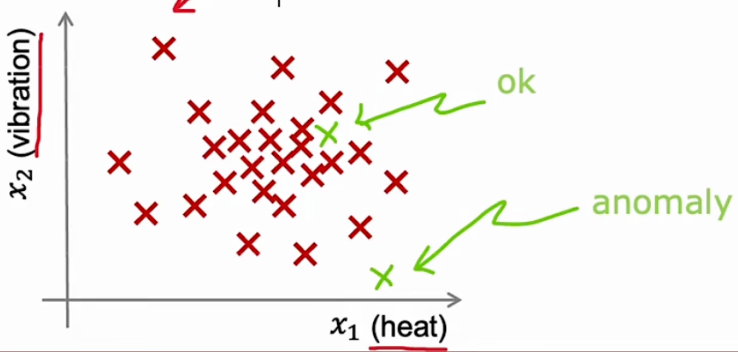
x_1 = heat generated

x_2 = vibration intensity

…

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

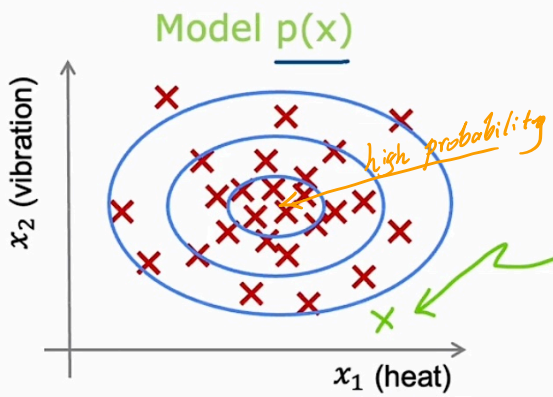
New engine: x_{test}



Density estimation

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

Is x_{test} anomalous?



$p(x_{test}) < \varepsilon \rightarrow$ Flag anomaly

$p(x_{test}) \geq \varepsilon \rightarrow$ "ok"

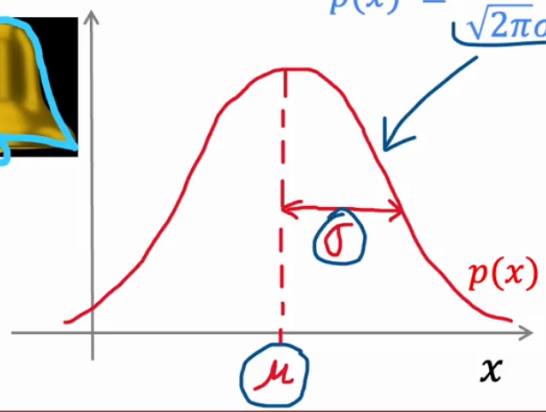
Gaussian (Normal) distribution

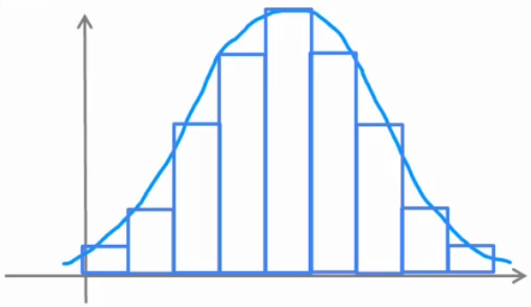
Say x is a number. If x is a distributed Gaussian with mean μ, σ^2 variance .

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$$

$$\mu = \frac{1}{m}\sum_{i=1}^{m} x_i$$

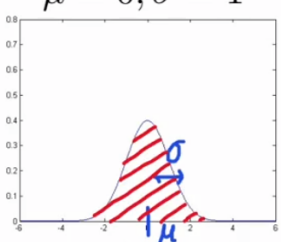
$$\sigma^2 = \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu)^2$$



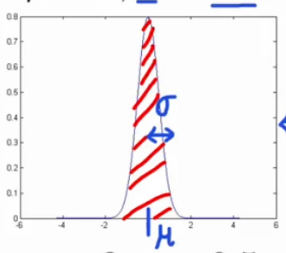


Gaussian distribution example

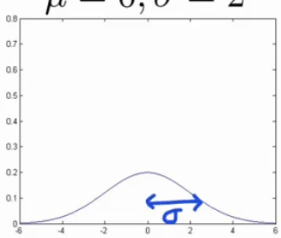
$\mu = 0, \sigma = 1$



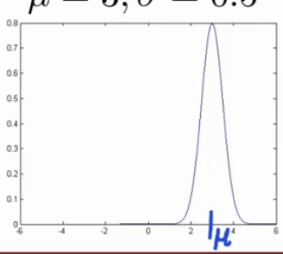
$\mu = 0, \sigma = 0.5$ $\sigma^2 = 0.25$



$\mu = 0, \sigma = 2$ $\sigma^2 = 4$



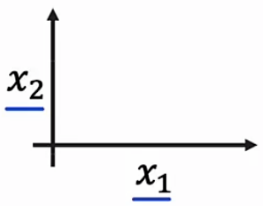
$\mu = 3, \sigma = 0.5$



Density estimation

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

Each example x_i has n features



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

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

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

$\sum \rightarrow +$ $\prod \rightarrow \times$

$p(x_1 = \text{high temp}) = 1/10$

$p(x_2 = \text{high vibra}) = 1/20$

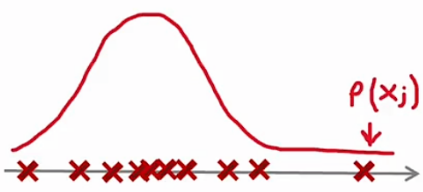
$p(x_1, x_1) = p(x_1) * p(x_2)$

$$= \frac{1}{10} * \frac{1}{20} = \frac{1}{200}$$

Anomaly detection algorithm

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

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

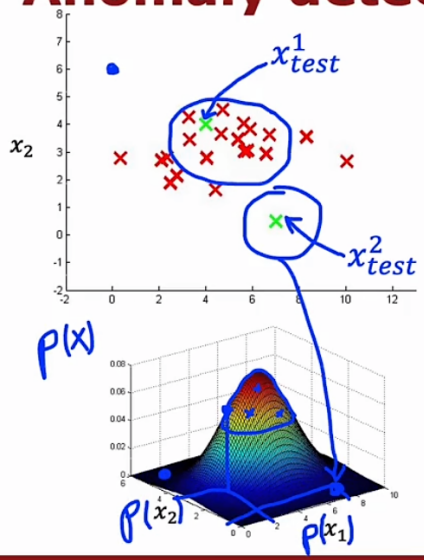


3. Given new example x, compute $p(x)$:

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

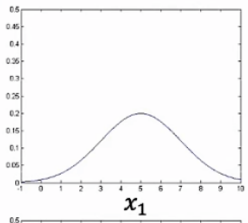
Anomaly if $p(x) < \varepsilon$

Anomaly detection example

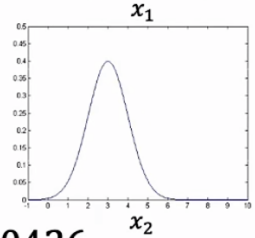


$\mu_1 = 5, \sigma_1 = 2$

$\mu_2 = 3, \sigma_2 = 1$



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



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

$\varepsilon = 0.02$

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

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

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. ($y = 0$ if normal, $y = 1$ if anomalous).

Training set: $x^{(1)}, x^{(2)}, \ldots, x^{(m)}$ (assume normal examples/not anomalous) $y = 0$

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 $y = 1$

Aircraft engines monitoring example

10000 good (normal) engines

20 flawed engines (anomalous)

Training set: 6000 good engines

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

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

Alternative:

Training set: 6000 good engines

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

No test set

Algorithm evaluation

Fit model $p(x)$ on training set $x^{(1)}, x^{(2)}, \ldots, x^{(m)}$

On a cross validation/test example x, predict

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

Possible evaluation metrics:

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

and that these are alternative metrics to classification accuracy that could work better when your data distribution is very skewed.

并且这些是分类准确性的替代指标，当您的数据分布非常倾斜时，它们可以更好地工作

Anomaly detection vs. Supervised learning

Very small number of positive examples ($y = 1$). (0-20 is common). Large number of negative ($y = 0$) examples.

Many different "types" of anomalies. Hard for any algorithm to learn from positive examples what the anomalies look like; future anomalies may look nothing like any of the anomalous examples we've seen so far.

Fraud

Large number of positive and negative examples.

20 positive examples

Enough positive examples for algorithm to get a sense of what positive examples are like, future positive examples likely to be similar to ones in training set.

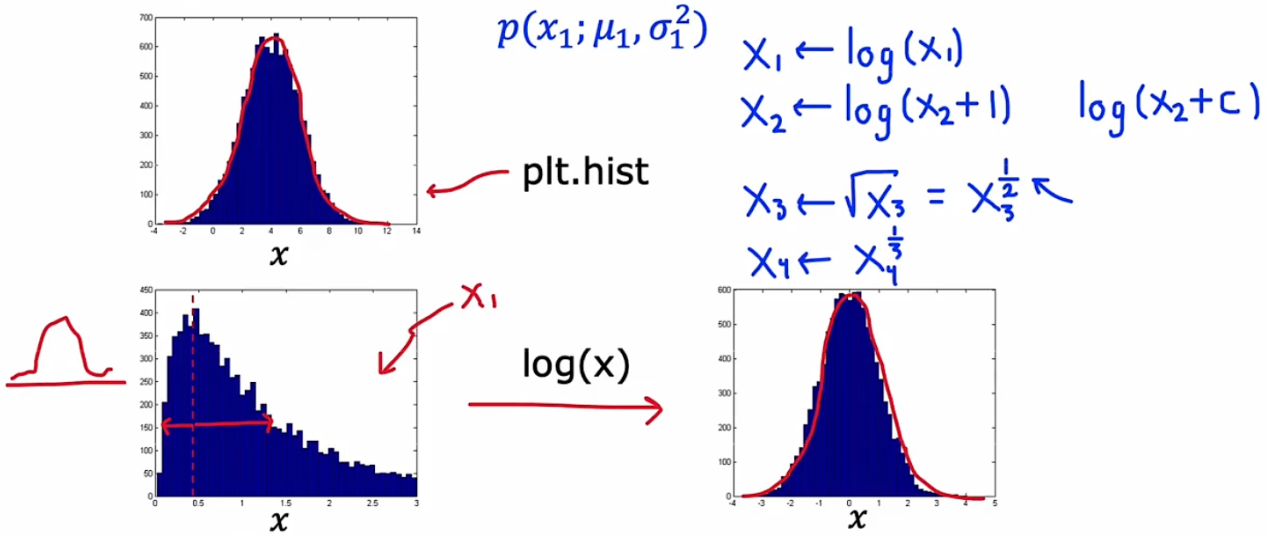
Spam

Anomaly detection vs. Supervised learning

- Fraud detection
- Manufacturing - Finding new previously unseen defects in manufacturing.(e.g. aircraft engines)
- Monitoring machines in a data center

- Email spam classification
- Manufacturing - Finding known, previously seen defects
  scratches
- Weather prediction (sunny/rainy/etc.)
- Diseases classification

Non-gaussian features



DeepLearning.AI Stanford ONLINE Andrew Ng

我有时会观察我的特征，如果画完直方图发现某个特征非常的不高斯，我可能会选择用这样或其他的转换让它变得更高斯

I'll sometimes take a look at my features and if I see any of the highly non Gaussian by plotting a histogram, I might choose transformations like these or others in order to try to make it more Gaussian.

提醒一下，无论你对训练集使用什么转换

And just as a reminder, whatever transformations you apply to the training set

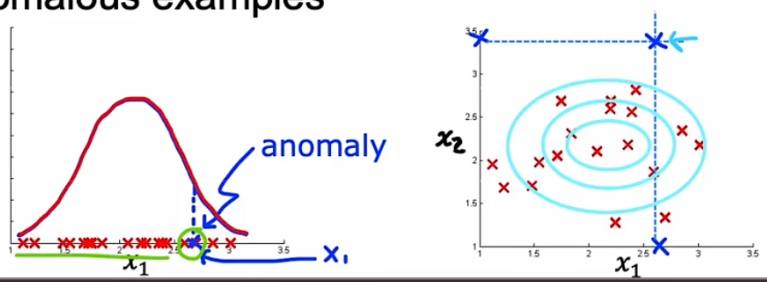
请记住对交叉验证和测试集数据也进行相同的转换

Please remember to apply the same transformation to your cross validation and test set data as well.

Error analysis for anomaly detection

Want $p(x) \geq \varepsilon$ large for normal examples x.
$p(x) < \varepsilon$ small for anomalous examples x.

Most common problem:
$p(x)$ is comparable (say, both large) for normal and anomalous examples



先训练模型，然后查看算法在交叉验证集中没能检测出的异常，然后查看这些样本，考虑是否有必要新建一个特征，从而使算法能够发现这些异常

What if we have features of the movies?

$n_u = 4$
$n_m = 5$
$n = 2$

Movie	Alice(1)	Bob(2)	Carol(3)	Dave(4)	x_1 (romance)	x_2 (action)
Love at last	5	5	0	0	0.9	0
Romance forever	5	?	?	0	1.0	0.01
Cute puppies of love	?	4	0	?	0.99	0
Nonstop car chases	0	0	5	4	0.1	1.0
Swords vs. karate	0	0	5	?	0	0.9

$x^{(1)} = \begin{bmatrix} 0.9 \\ 0 \end{bmatrix}$

$x^{(3)} = \begin{bmatrix} 0.99 \\ 0 \end{bmatrix}$

For user 1: Predict rating for movie i as: $w^{(1)} \cdot x^{(i)} + b^{(1)}$ ← just linear regression

$w^{(1)} = \begin{bmatrix} 5 \\ 0 \end{bmatrix}$ $b^{(1)} = 0$ $x^{(3)} = \begin{bmatrix} 0.9 \\ 0 \end{bmatrix}$ $w^{(1)} \cdot x^{(3)} + b^{(1)} = 4.95$

For user j: Predict user j's rating for movie i as $w^{(j)} \cdot x^{(i)} + b^{(j)}$

所以这里的参数wj和bj是用于预测用户j给电影i的评分的参数，它是xi的函数，是电影i的一个特征

So here the parameters WJ and BJ are the parameters used to predict user J rating for movie I which is a function of XI which is a features of movie i

Cost function

Notation:

- $r(i,j) = 1$ if user j has rated movie i (0 otherwise)
- $y^{(i,j)}$ = rating given by user j on movie i (if defined)
- $w^{(j)}, b^{(j)}$ = parameters for user j
- $x^{(i)}$ = feature vector for movie i

For user j and movie i, predict rating: $w^{(j)} \cdot x^{(i)} + b^{(j)}$

$m^{(j)}$ = no. of movies rated by user j

To learn $w^{(j)}, b^{(j)}$

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

number of features

Regularization term

Cost function

① 电影特征 → 用户的

To learn parameters $w^{(j)}, b^{(j)}$ for user j :

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

To learn parameters $w^{(1)}, b^{(1)}, w^{(2)}, b^{(2)}, \cdots w^{(n_u)}, b^{(n_u)}$ for all users :

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

Cost function

② 用户评价 → 电影特征

Given $w^{(1)}, b^{(1)}, w^{(2)}, b^{(2)}, \cdots, w^{(n_u)}, b^{(n_u)}$

to learn $x^{(i)}$:

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

→ To learn $x^{(1)}, x^{(2)}, \cdots, x^{(n_m)}$:

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

Collaborative filtering

① + ② 协同过滤

		$j=1$ Alice	$j=2$ Bob	$j=3$ Carol
$i=1$ →	Movie1	5	5	?
$i=2$ →	Movie2	?	2	3

Cost function to learn $w^{(1)}, b^{(1)}, \cdots w^{(n_u)}, b^{(n_u)}$:

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

Cost function to learn $x^{(1)}, \cdots, x^{(n_m)}$:

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

Put them together:

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

Gradient Descent

collaborative filtering

Linear regression (course 1)

repeat {

$$w_i = w_i - \alpha \frac{\partial}{\partial w_i} J(w,b) \qquad w_i^{(j)} = w_i^{(j)} - \alpha \frac{\partial}{\partial w_i^{(j)}} J(w,b,x)$$

$$b = b - \alpha \frac{\partial}{\partial b} J(w,b) \qquad b^{(j)} = b^{(j)} - \alpha \frac{\partial}{\partial b^{(j)}} J(w,b,x)$$

$$x_k^{(i)} = x_k^{(i)} - \alpha \frac{\partial}{\partial x_k^{(i)}} J(w,b,x)$$

}

parameters w, b, x — x is also a parameter

Cost function for binary application

Previous cost function:

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

$f(x)$

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

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

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

然后，这就给你了可用于二元标签协同过滤的代价函数

then this gives you the cost function that you can use for collaborative filtering on binary labels.

Mean Normalization

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

$$\mu = \begin{bmatrix} 2.5 \\ 2.5 \\ 2 \\ 2.25 \\ 1.25 \end{bmatrix}$$

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

For user j, on movie i predict:

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

$y^{(i,j)}$

$$w^{(j)}, b^{(j)}, x^{(i)}$$

User 5 (Eve):

$$w^{(5)} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad b^{(5)} = 0$$

$$\underbrace{w^{(5)} \cdot x^{(1)} + b^{(5)}}_{0} + \mu_1 = 2.5$$

将新用户标为均值

Collaborative filtering vs Content-based filtering

- **Collaborative filtering:**
  Recommend items to you based on rating of users who gave similar ratings as you

- **Content-based filtering:**
  Recommend items to you based on features of user and item to find good match

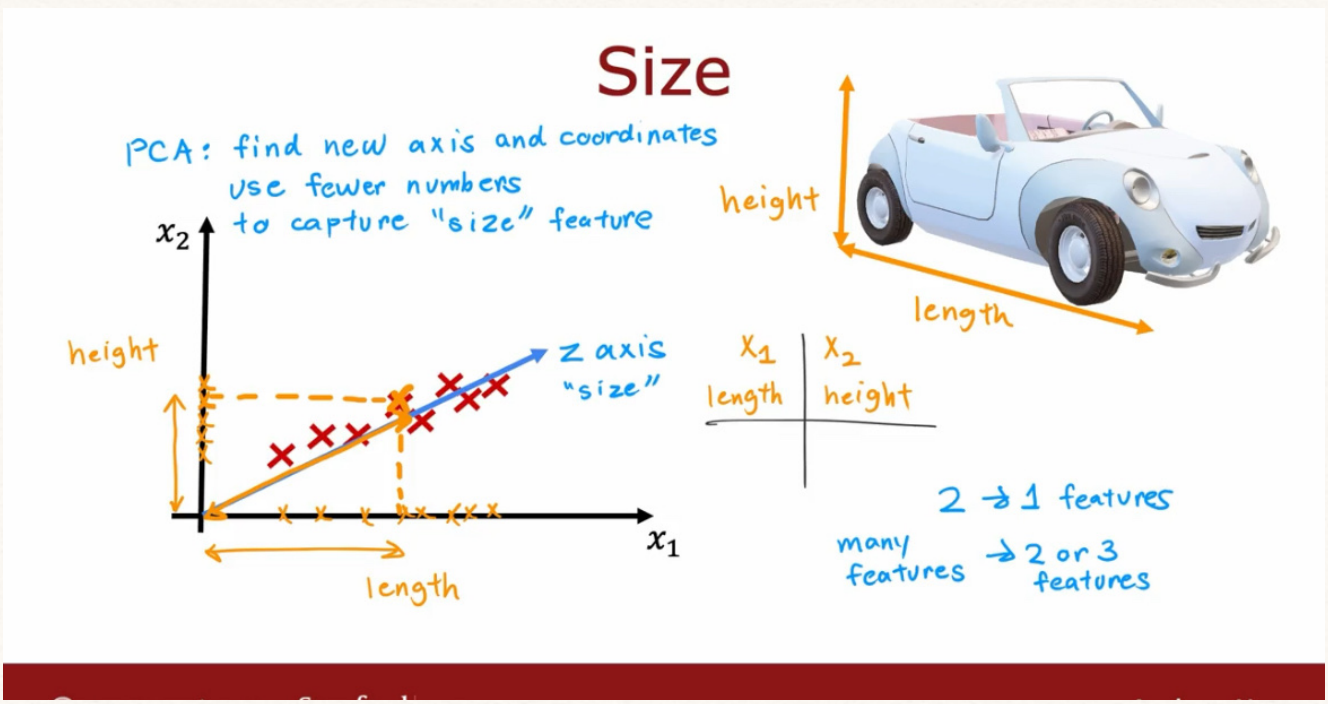
$r(i,j) = 1$ if user j has rated item i

$y^{(i,j)}$ rating given by user j on item i (if defined)

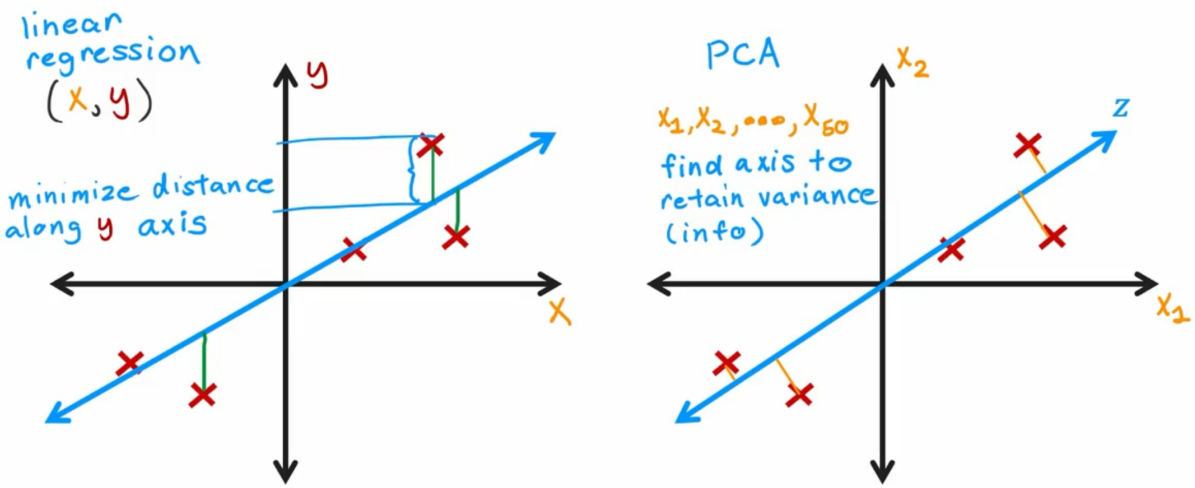
比起只用协同过滤算法，基于内容过滤可以更好的匹配用户和物品

to find better matches than potentially a pure collaborative filtering approach might be able to.

PCA

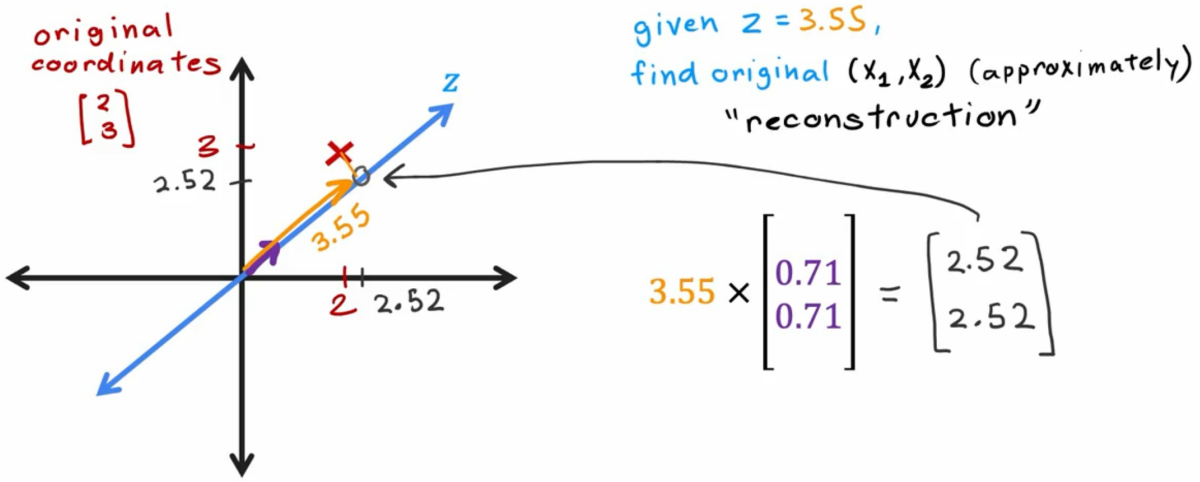

Size
PCA: find new axis and coordinates
use fewer numbers
to capture "size" feature
x_2
height
z axis
"size"
x_1
length
height
length
X_1
length
X_2
height
2 → 1 features
many features → 2 or 3 features

PCA is not linear regression



move to be projected down to Z.

Approximation to the original data



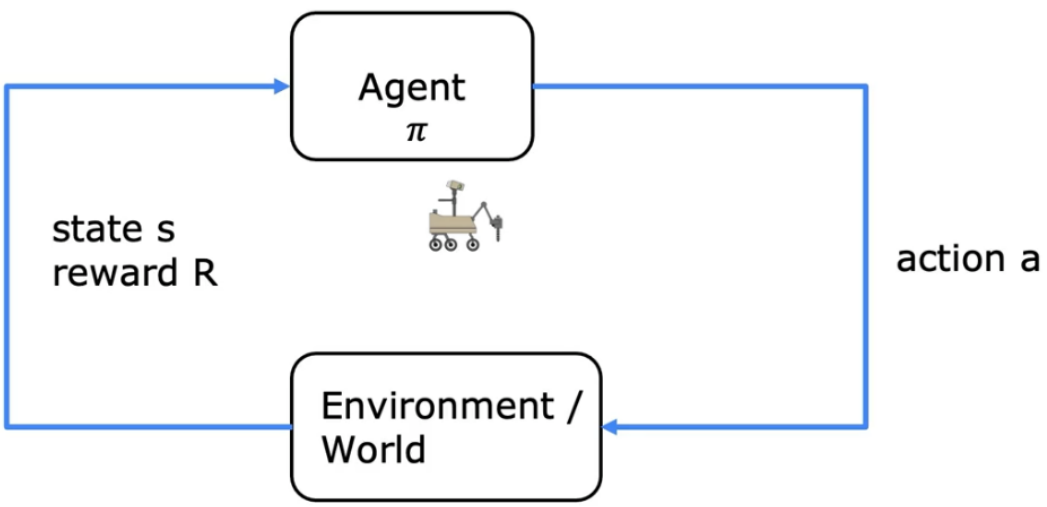
we could get a reasonable approximation to

Reinforcement Learning

	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	100 ← ← ← → 40	Find $\pi(s) = a$	Find $\pi(s) = a$

DeepLearning.AI Stanford ONLINE Andrew Ng

Markov Decision Process (MDP)

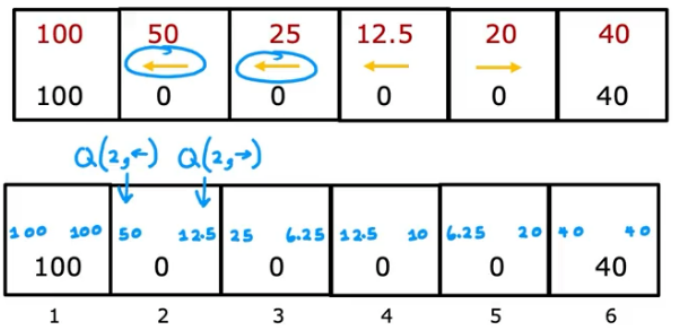


State action value function

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

Given Policy

$Q(s,a)$



return
action
reward

$Q(2,\rightarrow) = 12.5$

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

$Q(2,\leftarrow) = 50$

$0+(0.5)100$

$Q(4,\leftarrow) = 12.5$

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

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

$R(s)$ = reward of current state

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

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

Expected Return

Goal of Reinforcement Learning:

Choose a policy $\pi(s) = a$ that will tell us what action a to take in state s so as to maximize the expected return.

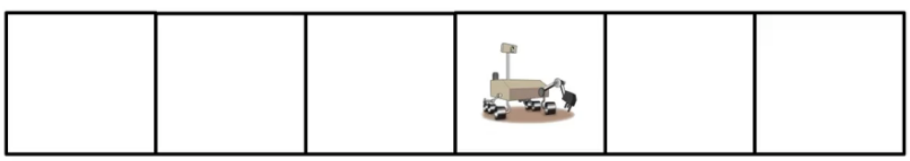
Bellman Equation:

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

3 ← 2 or 4

Discrete vs Continuous State

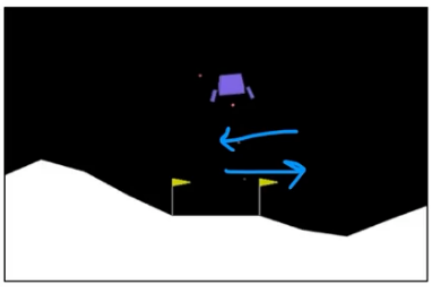
Discrete State:



Continuous State:

Reward Function

- Getting to landing pad: 100 – 140
- Additional reward for moving toward/away from pad.
- Crash: -100
- Soft landing: +100
- Leg grounded: +10
- Fire main engine: -0.3
- Fire side thruster: -0.03

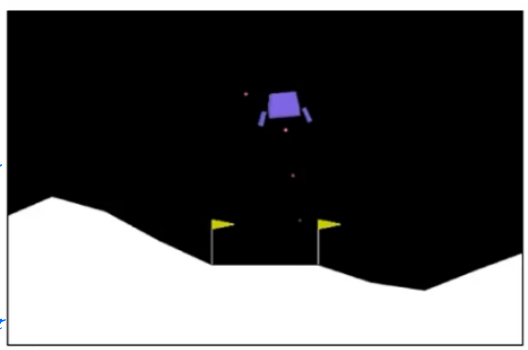


Lunar Lander Problem

Learn a policy π that, given

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

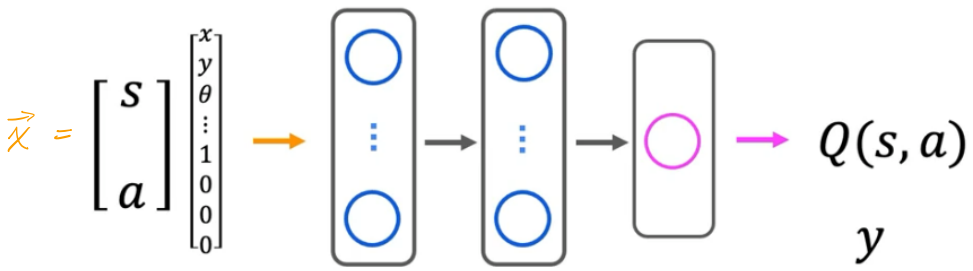
x, y: position
$\dot{x}, \dot{y}$: speed in direction
θ: angle
$\dot{\theta}$: angle changes speed
l, r: whether left/right leg has touched ground



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

$\gamma = 0.985$

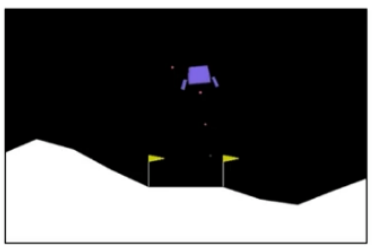
Deep Reinforcement Learning



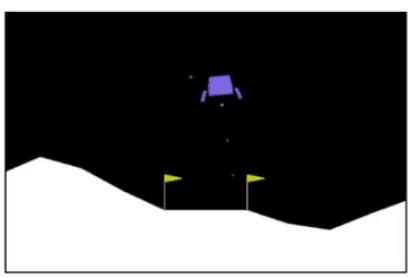
In a state s, use neural network to compute

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

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



Learning Algorithm



Initialize neural network randomly as guess of $Q(s,a)$.

Repeat {

Take actions in the lunar lander. Get $(s, a, R(s), s')$.

Store 10,000 most recent $(s, a, R(s), s')$ tuples.

Replay Buffer

Train neural network:

Create training set of 10,000 examples using

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

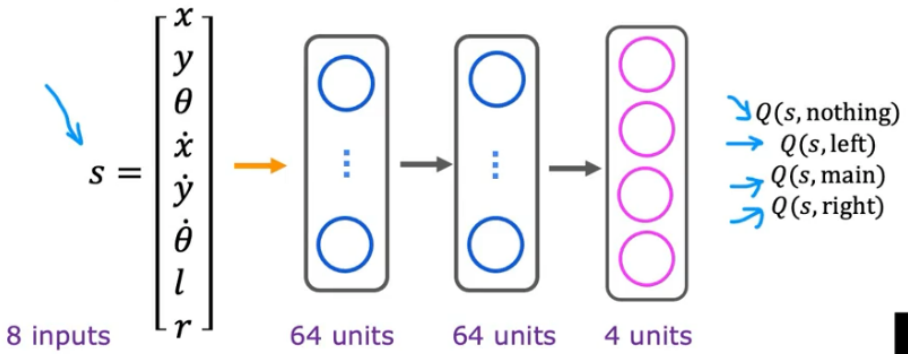
Train Q_{new} such that $Q_{new}(s,a) \approx y$.

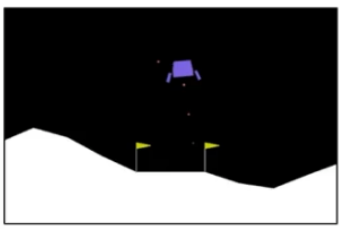
Set $Q = Q_{new}$.

$f_{w,b}(x) \approx y$

x, y $\quad x^{(1)}, y^{(1)}$ $\quad \vdots$ $\quad x^{(10000)}, y^{(10000)}$

Deep Reinforcement Learning





In a state s, input s to neural network.

Pick the action a that maximizes $Q(s,a)$. $R(s) + \gamma \max_{a'} Q(s', a')$

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

why exploration:
if an action always takes low Q, the model may never choose this action, but the action can be effective sometimes.

Start ε high
1.0 → 0.01
Gradually decrease

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.