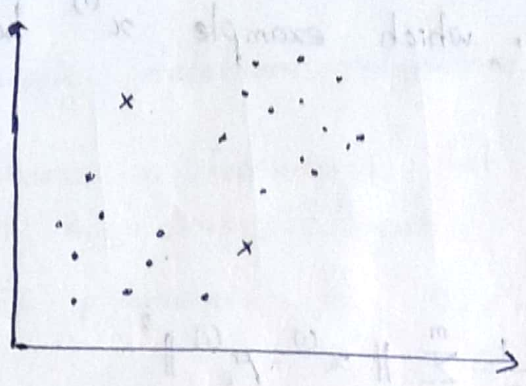
Unsupervised Learning, Recommender Systems and Reinforcement Learning

K - means intution



1) Assign each point to its closest centroid

2) Recompute the centroids

K - means algorithm

Randomly initialize K clusters centroids $(\mu_1, \ldots, \mu_k)$

Repeat {

\# Assign points to cluster centroids

for i = 1 to m

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

min $\|x^{(i)} - \mu_k\|^2$

\# Move cluster centroids

for k = 1 to K

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

}

Optimization

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

μ_k : cluster centroid k

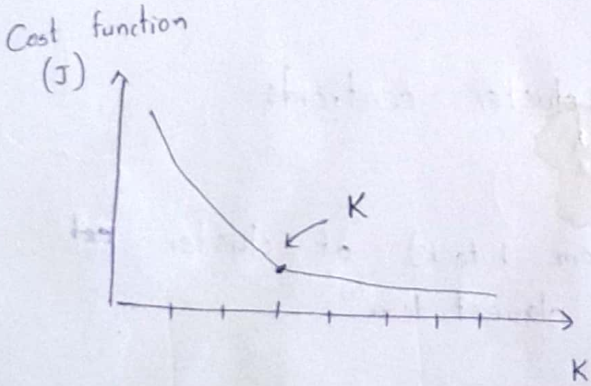
$\mu_{c^{(i)}}$: cluster centroid of cluster to which example $x^{(i)}$ has been assigned

Cost function

$$J\left(c^{(1)} \dots c^{(m)}, \mu^{(1)}, \dots, \mu^{(k)}\right) = \frac{1}{m} \sum_{i=1}^{m} \| x^{(i)} - \mu_{c^{(i)}} \|^2$$

(Distortion cost function)

Eldow method $\longrightarrow$ find value of K



Anomaly Detection example

Fraud Detection

Manufacture industry in fault detection

Monitor computers in data centers

Anomaly detection algorithm

1. Choose n features x_i that you think might be indicative of anomalous examples

2. Fit parameters $\mu_1, \ldots, \mu_n, \sigma_1^2, \ldots, \sigma_n^2$

$$\mu_j = \frac{1}{m} \sum_{i=1}^{m} x_j^{(i)}$$

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

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

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

$$= \prod_{j=1}^{n} \frac{1}{\sqrt{2\pi}\,\sigma} \exp\left(- \frac{(x_j - \mu_j)^2}{2\sigma_j^2} \right)$$

Anomaly if $p(x) < \varepsilon$

Developing & Evaluating an anomaly detection

Aircraft engine monitoring example

10000 - good

20 - flawed

Training : 6000 good

CV : 2000 good $(y=0)$, 10 anomalous $(y=1)$ ← Tune ε

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

If you don't have enough anomaly data then combine CV and test sets and continue.

Algorithm evaluation

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

On a cross validation / test example x, predict

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

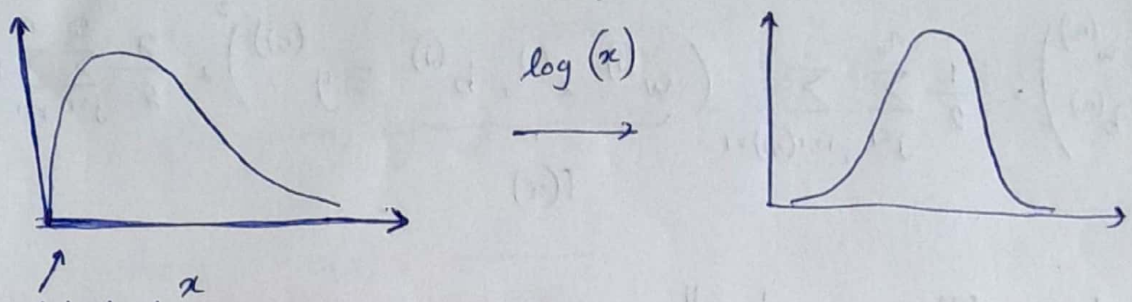
Possible evaluation metrics

- True positive, false positive, false negative, true negative
- Precision / Recall
- F1 Score

Can also use cross validation set to choose parameter ε

Anomaly Detection	Supervised Learning
Very small number of positive examples $(y=1)$. (0 - 20 is common) Large number of negative $(y=0)$ ex..	Large number of positive & negative (20 positive)
* Future anomalies may look nothing like any of the anomalous examples we've seen so far	Future positive examples likely to be similar to once in training set.
Fraud	Spam
- Fraud Detection - Manufacturing : new previously unseen defects - Monitoring machines in data centers	- Email spam classification - Manufacturing - Previously seen defect - Weather prediction - Diseases classification

Non gaussian features



plt.hist (x, bins = , colors =)

np.log (x + 0.0..)

positive value එකක් වෙන්න ඕන.

Error analysis for anomaly detection

Problem:

$p(x)$ is comparable (say, both large) for normal and anomalous examples.

Recomonder Systems

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

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

$w^{(j)}, b^{(j)}$: parameter 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.

number of features

Cost function = $J(w^{(j)}, b^{(j)})$

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

min $w^{(j)}, b^{(j)}$

for all the parameters

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

Collaborative filtering algorithm

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

to learn $x^{(i)}$

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

To learn $x^{(1)}, x^{(2)}, \dots, x^{(n_m)}$

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

Put A and B together : $J(w, b, x)$

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

Use gradient descent

repeat {

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

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

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

}

For binary labels

Predict that the probability of $y^{(i,j)} = 1$

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

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

Loss function

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

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

loss for single example

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

Mean normalization

For user j on movie i predict :

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

mean

Tensorflow Implementation

Finding related items

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.

predict rating of user j on movie i as

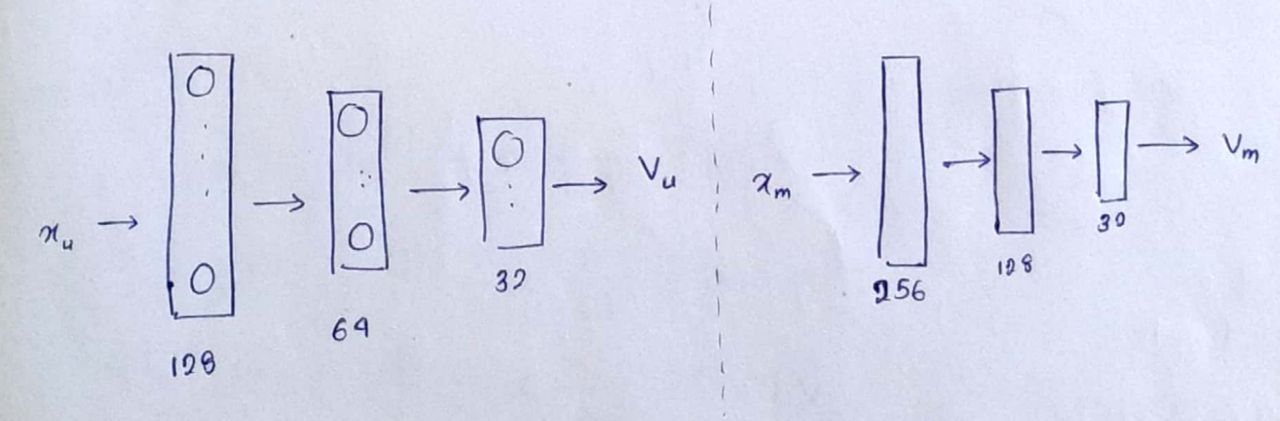
$$w^{(j)} \cdot x^{(i)} + b^{(j)}$$

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

computed from $x_u^{(j)}$ computed from $x_m^{(i)}$

Deep learning based Content Based filtering

$X_u \longrightarrow V_u$ (User network) | $X_m \longrightarrow V_m$ (Movie network)



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

$g(v_u^{(j)} \cdot v_m^{(i)})$ to predict the probability that $y^{(i,j)}$ is 1

Cost function $J = \sum_{(i,j): r(i,j)=1} \left(v_u^{(j)} \cdot v_m^{(i)} - y^{(i,j)} \right)^2 +$ Regularization Term

Recommendating from larger catlog

Retrival : Generate large list of plausible item candidates

Combine retrieved items into list, removing duplicates and items alredy watched / purchased

Ranking : Take list retrived and rank using learned model

Rainforcement Learning

Mars rover example

terminal state					terminal state
100	0	0	0	0	40
1	2	3	4	5	6

← →
left right

state

4 3 2 1

0 0 0 100

0 0 40

0 0 0 0 0 100 ← එක පාරම decide කර ගන්න බැරි වෙලා. මේක හොඳ නෑ.

$$(s, a, R(s), s')$$

$$(4, \leftarrow, 0, 3)$$

Return of reinforcement learning

$$\text{Return} : 0 + (0.9).0 + (0.9)^2.0 + (0.9)^3.100 = 72.9$$

$$= R_1 + \gamma R_2 + \gamma^2 R_3 + \dots \quad (\text{until terminal state})$$

$\gamma \leftarrow$ Discount factor (0.9)

0.99, 0.999 මේ වගේ value එකක් normaly use කරන්නෙ.

$\gamma = 0.5$ වගේ දුන්නහම delay එක අඩු එකට යනවා.

100	50	25	12.5	6.25	40	← return
100	← 0	← 0	← 0	← 0	40	← reward

$\gamma = 0.5$

return depend on your action

$0 + 0 \times 0.5$, $(0.5)^3 \times 40$

$0 + 0.5 \times 40$

100	2.5	5	10	20	40
100	→ 0	→ 0	→ 0	→ 0	40

100	50	25	12.5	20	40
←	←	←	←	→	
100	0	0	0	0	40
1	2	3	4	5	6

$0 + (0.5) \times 40 = 20$

Making decisions : Policies in Rainforcement learning

state $\xrightarrow{\text{policy}}$ action

s $\quad \pi \quad$ a

$\pi(s) = a$

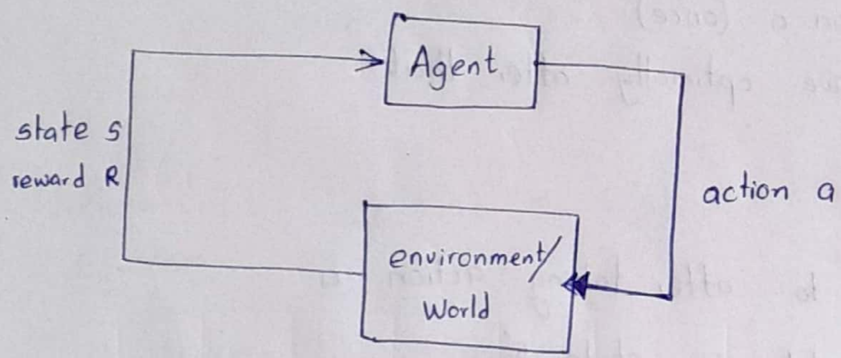
$\pi(2) = \leftarrow$

$\pi(3) = \leftarrow$

$\pi(4) = \leftarrow$

$\pi(5) = \rightarrow$

Markov Decision Process (MDP)



State action value function (Q function)

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

100	50	25	12.5	20	40	← return
	←	←	←	→		← action
100	0	0	0	0	40	← reward
1	2	3	4	5	6	

$Q(2, \leftarrow) = 50$

$Q(2, \rightarrow)$

$0 + (0.5)^1 \cdot 0 + (0.5)^2 \cdot 0 + (0.5)^3 \cdot 100 = 12.5$

The best possible return from state s is

$$\max_a Q(s,a)$$

The best possible action in state s is the action a that gives $\max_a Q(s,a)$

Q^* ← Optimal Q function

Bellman Equation

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

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

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

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

$s = 2$

$a = \rightarrow$

$s' = 3$

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

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

$$= 0 + 0.5 \times 25$$

$$= 12.5$$

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

$$\Rightarrow R_1 + \gamma R_2 + \gamma^2 R_3 + \gamma^3 R_4 + \dots$$

$$R_1 + \gamma \left[R_2 + \gamma R_3 + \gamma^2 R_4 + \dots \right]$$

Random (Stochastic) Environment

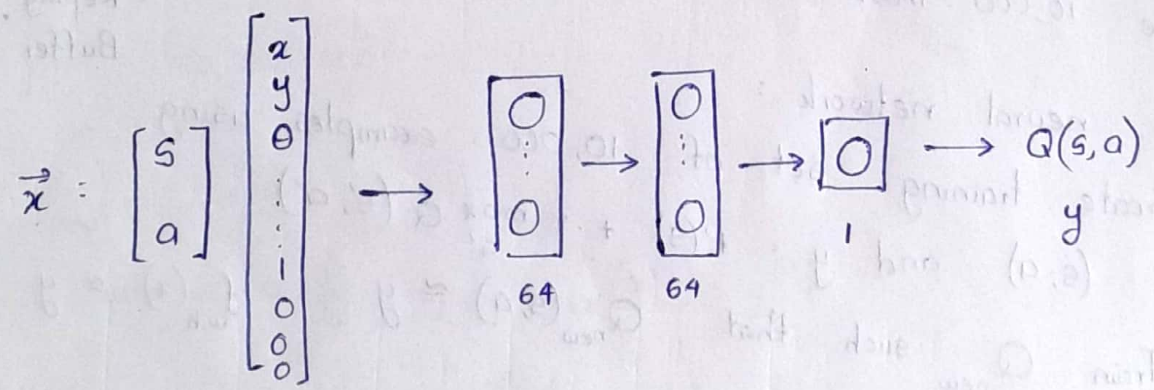
Average ගන්නවා ඇති එක එක path වල තියෙන එකතුව

Expected Return : Average $(R_1 + \gamma R_2 + \gamma^2 R_3 + \gamma^3 R_4 + \dots)$

$$= E\left[R_1 + \gamma R_2 + \gamma^2 R_3 + \gamma^3 R_4 + \dots \right]$$

Bellman Equation : $Q(s,a) = R(s) + \gamma \, E\left[\max_{a'} Q(s', a') \right]$

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 maximze $Q(s,a)$

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

$$f_{W,B}(x) \approx y \qquad y^{(1)} = R(s^{(1)}) + \gamma \max_{a'} Q(s'^{(1)}, a')$$

$(s, a, R(s), s')$

$(s^{(1)}, a^{(1)}, R(s^{(1)}), s'^{(1)})$

$(s^{(2)}, a^{(2)}, R(s^{(2)}), s'^{(2)})$

$(s^{(3)}, a^{(3)}, R(s^{(3)}), s'^{(3)})$

x	y
$x^{(1)} = (s^{(1)}, a^{(1)})$ 8 4	$y^{(1)}$
$x^{(2)} = (s^{(2)}, a^{(2)})$	$y^{(2)}$
$x^{(10,000)}$	$y^{(10,000)}$

Learning Algorithm (DQN 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 $Q_{new}(s, a) \approx y$ $\qquad f_{w,b}(x) \approx y$

Set $Q = Q_{new}$

ε - greedy algorithm

In some state s

Option 1:
pick the action a that maximizes $Q(s, a)$

Option 2:

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

With probability 0.05 pick an action a ~~rat~~ randomly "Exploration"

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

Start ε high
Gradually decrease

Algorithm refinement : Mini - batch and soft update (optional)

How to choose actions while still learning?

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

$m = 100,000,000$

$m' = 1000$

repeat {

$$w = w - \alpha \frac{\partial}{\partial w} \frac{1}{2m'} \sum_{i=1}^{m'} \left(f_{w,b}(x^{(i)}) - y^{(i)} \right)^2$$

$$b = b - \alpha \frac{\partial}{\partial b} \frac{1}{2m'} \sum_{i=1}^{m'} \left(f_{w,b}(x^{(i)}) - y^{(i)} \right)^2$$

}

× Training set එක small pieces වලට කඩනවා.

Soft Update

Set $\underset{w, B}{Q} = \underset{w_{new},\ B_{new}}{Q_{new}}$

$$W = 0.01\, W_{new} + 0.99\, W$$

$$B = 0.01\, B_{new} + 0.99\, B$$