KNN Algorithm for approximating a discrete valued function.

Training Algorithm

* for each training example $\langle x, f(x) \rangle$,
  add the example to the list t_example

Classification Algorithm

* Given a query instance x_q to be classified
  - Let $x_1 \dots x_k$ denote k instance from t_example that are nearest to x_q

discrete

Return

$$\hat{f}(x_q) = \underset{v \in V}{\operatorname{argmax}} \sum_{i=1}^{k} \delta(v, f(x_i))$$

where $\delta(a,b) = 1$ if $a = b$ and where $\delta(a,b) = 0$ otherwise.

KNN for Continuous valued target function

$\hat{f} : R \to R$

$$\hat{f}(x_q) = \frac{\sum_{i=1}^{k} f(x_i)}{k}$$

$$\hat{f}(x_q) = \frac{\sum_{i=1}^{k} w_i f(x_i)}{\sum_{i=1}^{k} w_i}$$

Distance weighted NN Algorithm.

↳ giving greater weight to closer Neighbor.

discrete

discrete

$$\hat{f}(x_q) = \underset{v \in V}{\operatorname{argmax}} \sum_{i=1}^{k} w_i \delta(v, f(x_i))$$

Real Value

$$\hat{f}(x_q) = \frac{\sum_{i=1}^{k} w_i f(x_i)}{\sum_{i=1}^{k} w_i}$$

$$w_i = \frac{1}{d(x_q, x_i)^2}$$

$\hat{f}(x_q) =$

Scanned with OKEN Scanner

: Technique for efficiently indexing training examples is required.

* It considers all attributes of the instances when attempting to retrieve similar training examples from memory.

↳ if the target concept depends on only a few of the many available attributes then the instances that are truly most "similar" may well be a large distance apart.

K-Nearest Neighbor learning

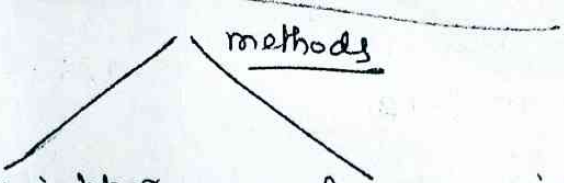
The nearest neighbors of an instance are defined in terms of the standard Euclidean distance.

Let an instance x be described by the feature vector,

$$\langle a_1(x), a_2(x), \ldots a_n(x) \rangle$$

$a_r(x)$ → denotes the value of the rth attribute of the instance x.

Then the distance between two instance x_i and x_j is defined as $d(x_i, x_j)$ where



Learning in these algorithms

↳ storing the presented training data

↳ when a new query instance is encountered a set of similar related instance is retrieved from memory & used to classify the new query instance.

Difference between these approach & other methods.

Instance based approach	other methods
+ It can construct a different approximation to the target function for each distinct query instance that must be classified.	- It constructs only a local approximation to the target function - never construct an approximation designed to perform well over the entire instance space.

Disadvantage:

↳ cost of classifying new instance can be high.

↳ due to the fact that nearly all computation takes place at classification time rather than when the training examples are first encountered.

Scanned with OKEN Scanner

Locally weighted Linear Reg.

assume

$$\hat{f}(x) = w_0 + w_1 a_1(x_q) + \cdots w_n a_n(x)$$

$a_i(x) \Rightarrow i^{th}$ attri of instance x.

To minimize the error $\Rightarrow$ (Gradient decent)

$$E = \frac{1}{2} \sum_{x \in D} (f(x) - \hat{f}(x))^2$$

GD Training Rule

$$\Delta w_j = \eta \sum_{x \in D} (f(x) - \hat{f}(x)) \underline{a_j(x)}$$

η : learning rate

$a_j(x) \hookrightarrow x_j$

Modify the above to local approx

3 criteria used

① $E_1(x_q) = \frac{1}{2} \sum_{x \in k \text{ nearest}} (f(x) - \hat{f}(x))^2$

effient methods to fit linear function.

Lazy and Eager learning

Lazy: wait for query befor generalizing

k-nearest neighbar, case based reasoning

ager: generalize befor seeing query.

Radial basis function N/ws, ID3, Backprob etc

Eager ~~learner~~ learner ⇒ create global approx

Lazey learner ⇒ create local approx.

me same H lazy ⇒ represent more complex function.

1) Data is collected from people to know the quality of a special tissue as good or bad

ID	Acid Durab x_1	Strength x_2	class
1	7	7	Bad
2	7	4	Bad
3	3	4	Good
4	1	4	Good

Find the quality of a new paper tissue with $x_1 = 3$ and $x_2 = 7$ using KNN with $(k=1)$, $k=3$

$q(3,7)$

distance computation $= \sqrt{\sum_{r=1}^{2} a_r(x_i) - a_r(x_j)^2}$

x_i, x_j are instances i & j respectively

$d(x_1, x_5) = \sqrt{(3-7)^2 + (7-7)^2} = 4$

$d\{x$

$(x_3$

$(x_4$

16.4, 16.8, 17.6

18.1, 20.8

13.1, 13.6

new rule.

multiply distance penalty.

only k nearest

training examp

rks

1) Cost of more complex fun. ⟹ high
2) simple appr. target fun ⟹ quite well over a small subregion of instance space

Type of NN
⇒ ① Global approx
② use image
convert to local appr
classifier

al Basis function

* related to distance weighted regression.

③ closely related to distance weighted reg., but eager instead of lazy.

* ANN is learn Radial basic function.

yphothesi } function of the form
Learning }

$$\hat{f}(x) = w_0 + \sum_{u=1}^{k} w_u K_u d(x_u, x))$$

each x_u is an instance from X

$K_u(d(x_u, x))$ ⇒ decrease ⇒ distance incr

Al. Local method

$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}}$ the curve

global appro is localized to a region near by pt x_u using $K_u d(x_u, x)$

gaussian fund

$$K_u d(x_u, x) = e^{\frac{1}{2\sigma_u^2} d^2(x_u, x)}$$

$(x-\sigma)^2$

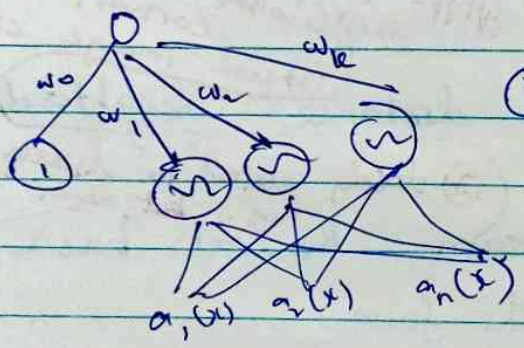
ANN Radial base function N/w

① 2 layer N/ws.

① First layer computes $K_u(d(x_u, x))$

② Second layer computes → linear combination of first layer

Radial base function



① k hidden units. each unit is defined by choosing values of x_u and σ_u^2

$K_u(d(x_u - x))$

to define

② weight w_u are trained to maximize the fit of N/w to the training data.

=> mo

difference

I
① lazy method => query instance x_q possible to generalize beyond the training data ①
② Eager => cannot, during query global approx generalization is already there

II
~~Lazy~~ Lazy and eager same hyp space H
eager → only one hyp space (linear functions)
single global approx.
single linear fun cover entire instance space
lazy
↓
uses richer hypothesis
use many different local linear funct.
to form its implicit global approx.
to target function.
(target function = Combination of many local approx.

$d_4 = \sqrt{(6-5)^2 + (8-5)^2} = \sqrt{1+9} = \sqrt{10} = 3.16$

$d_5 \quad \sqrt{(6-8)^2 + (8-8)^2} = \sqrt{4+0} = \sqrt{4} = 2$ (3)

3 neighbours $k=3$

d_2, d_3, d_5

(P) (P) (P) $\Rightarrow$ o/p is = Pass

Locally weighted Regression

Statistical tool used to understand and quantify the relation b/w 2 (or) more variables.

Linear Regression

$$y = \beta_0 + \beta_1 x + \varepsilon$$

y - dependent Vari
x = indep Var
β_0 - constant / intercept
β_1 - x-slop / coeff
ε - error

best suited for linearly the data.

Nlokal method

Curse of dimentionality.

Distance b/w neighbors will be dominated by the large No of irrelevant attributes

~~This~~ many irrelevant attributes are present, ~~Curse~~ **Curse of dimentionality.**

- weight to each attri differenty. distance b/w two instances
  - Stretching the axis in Euclidean ~~Space~~ Space
  - done automatically by cross validation approach
  - ~~to~~ shortening the axis when relevent attri
  - Values $z_1, \cdots z_n$ choose min true classi error.

Algorithm

1. To select random subset of available data.
2. determine values $z_1, \cdots z_n$ that lead to min error in classifying the remaining examples.
3. Repeat this ~~Reduce error~~ estimate weight factors

Regression: approximating a real value target fun.

Residual: error $f(x) - \hat{f}(x)$

kernal fun: function of distance that is used to determine the weight of each train example

$$w_i = K(d(x_i, x_q))$$

Global method

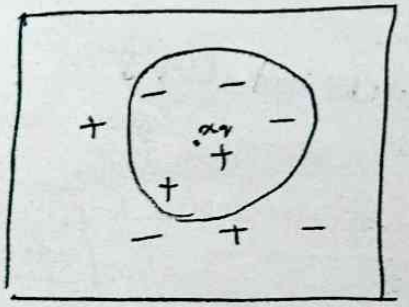
If all training examples are considered when classifying a new query instance. then it is called as global method.

local method

If only the nearest training examples are considered it is called as local method.

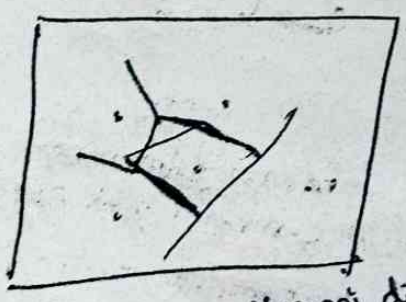
Shepard method

$$\hat{f}(x_q) = \frac{\sum_{i=1}^{n} w_i f(x_i)}{\sum_{i=1}^{n} w_i}$$



K=1 x_q = +

K=5 x_q = −



→ Combination of convex polygon

Voronoi digarm.

Scanned with OKEN Scann

Approx target fun $f(x)$ at single query point $x = x_q$

* construct f over a local region surrounding x_q
* Locally weighted Region uses nearby distance weighted training examples to form local approx to f
* find approx target function
  may use linear functions quadratic form

Locally weighted Linear Reg

local ⇒ function is approx. based on data near query point

weighted ⇒ each training examples is weighted by its distance from query point

Regression ⇒ problem of approx real Value fun

x_q - new query instance

⇒ $\hat{f}$ - fits training example in neighbourhood surrounding x_q

approx: $\hat{f}(x_q)$

* memories and then apply

* Instead of performing explicit generalization it compares new problem with instances in training which are stored in memory

example: spam mails → spam filters.

memory based learning / lazy learning.

Instance based learning

- Lazy learners (LWL)
- Radial based function (RBF)
- case based Reasoning (CBR)

data query = x (maths = 6, CS = 8) & k = 3

Maths	CS	Result
4	3	F
6	7	P
7	8	P
5	5	F
8	8	P
6	8	?

Euclidean distance (d)

$$d = \sqrt{(x_{o_1} - x_{a_1})^2 + (x_{o_2} - x_{a_2})^2}$$

o - observed value

a = actual value

Case Based Reasoning

* It is lazy learning (· query comes ⇒ then do
* Instance are rich symbolic Representation

* It is applied to complex conceptual problems.
  design of mech. devices (or) legal reasoning

Application of CBR

Design : landscape, building, mech, conceptual design of aircraft subsys
planning : repair schedule
Diagnosis : medical
Adversarial : legal

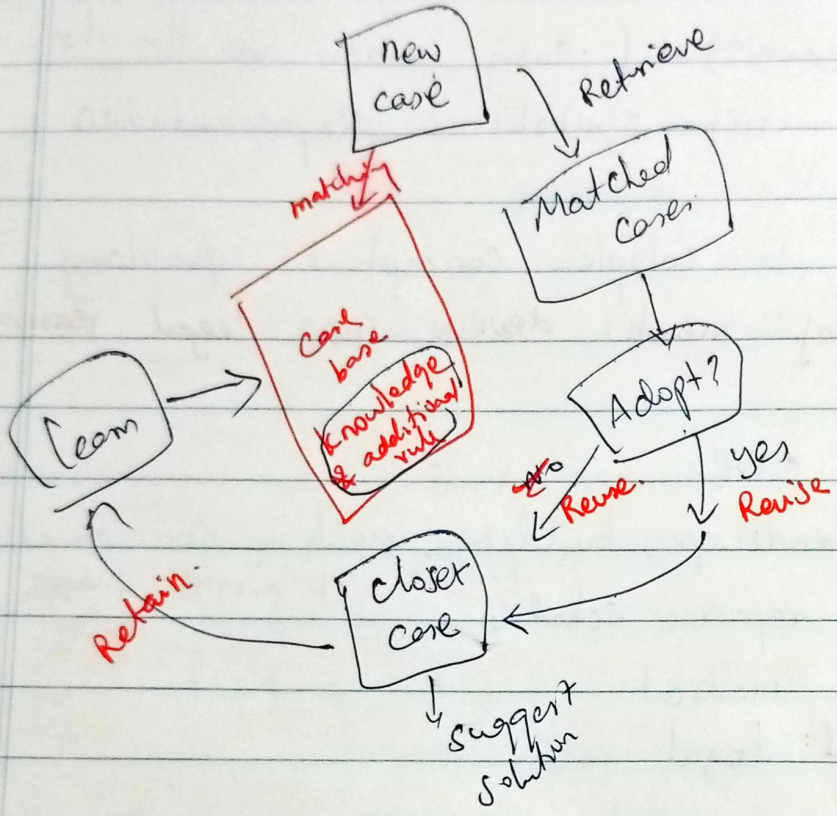
CBR Methodology

- Instance Represented by rich symb descriptions
- Search for similar cases, multiple retrieved cases may be combined
- Tight coupled b/w case retrieval, knowledge based reasoning and problem solving

challenges

- find good similarity metric
- Indexing based on syntatic similarity measure when failure, do backtracking and adopting to additional cases

CBR Process



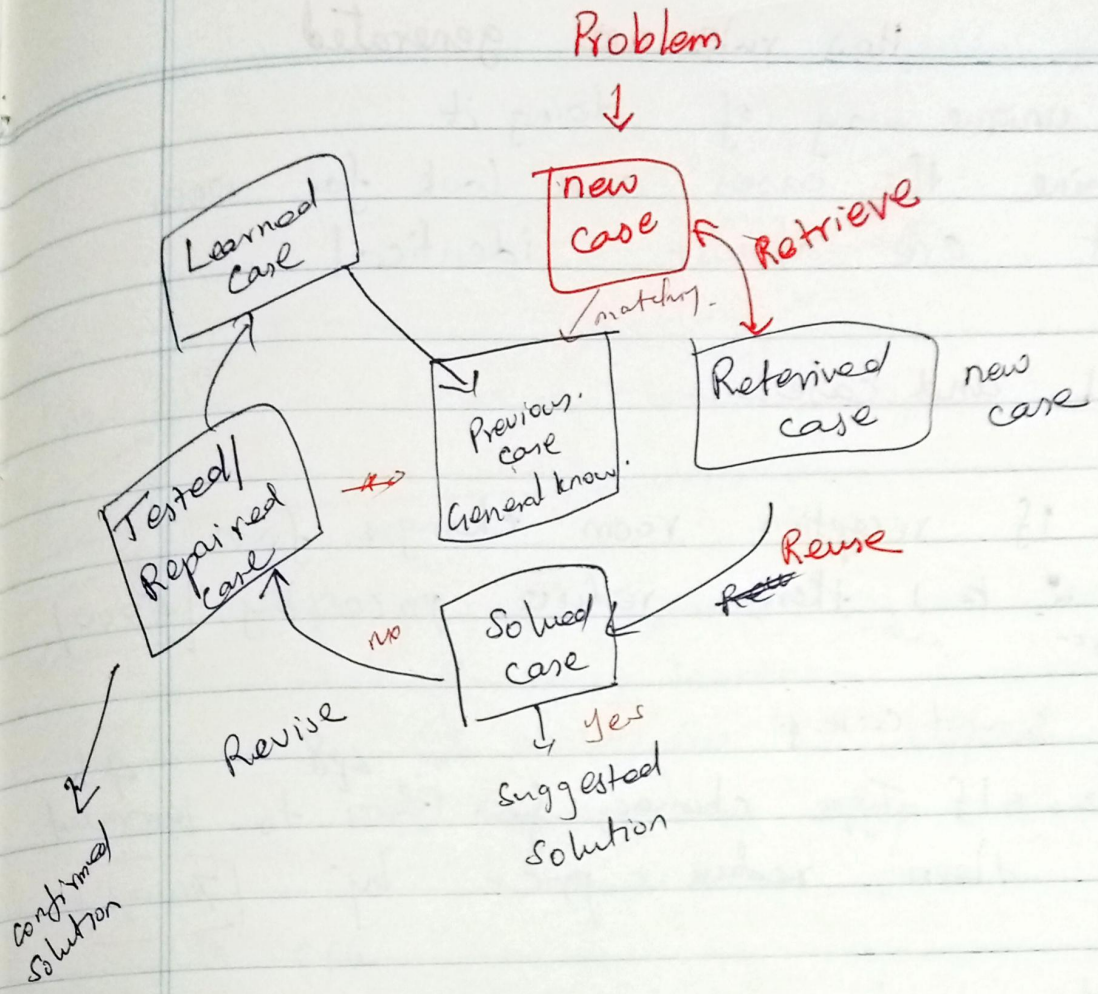
Retrieve - Given new case ~~revise~~ retrieve similar cases from the case base

Reuse - → adopt retrieved cases to ~~find~~ fit new case

Revise → Evaluate solution and revise it based on how weel it works

Retain ↓

decide whether to retain this new case in the case base



CBR example: property pricing

case	Location code	Bedroom	Recep room	Type	floors	Cond	price
1	8	2	1	terraced	1	Poor	20500
2	8	2	2	terraced	1	fair	25000
3	5	1	2	semi	2	good	48000
4	5	1	2	terrace.	2	good	41000
Test instance							
5	7	2	2	semi	1	poor	???

How rules are generated

- No unique way of doing it
- Examine the cases and look for ones that are almost identical.

case 1 and case 2

R_1 If reception room changes from 2 to 1 then reduce price by 5000/-

case 3 and case 4

R_2: If Type changes from Semi to terraced then reduce price by 7000/-

Adopting

Reverse rule 2

- If type changes from terraced to semi increase price by 7000

- Apply Reversed rule 2

Testing

7	2	2	Semi	1	poor
8	2	2	terr	1	fair

25000

25,000/- + change terraced to semi.

increased price by 7000.

new instance ⇒ 25000 + 7000 = 32,000/-

3 properties of instance based learners

1. lazy learners
2. classification is different for each inst
3. Instance are Rep with in dimensi euclidean space

In CBR

everything is considered as case based on previ case we propose a solut

- Instanse are represented as symbols (not valu

CBC has 3 components

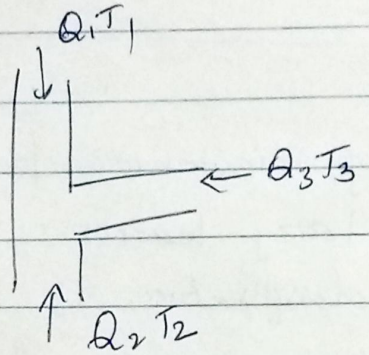
1. Similarity fun (or) distance measure
2. Approx / Adj of instance
3. Symbolic repr of instances

for modelling CBR we use CADET System
(Case base design tool)

has 75 predefined examples

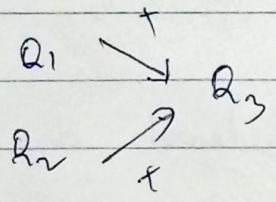
Example : modern water taps

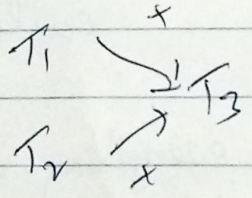
T-Junction Pipe



$Q \Rightarrow$ water flow
$T \Rightarrow$ temperature

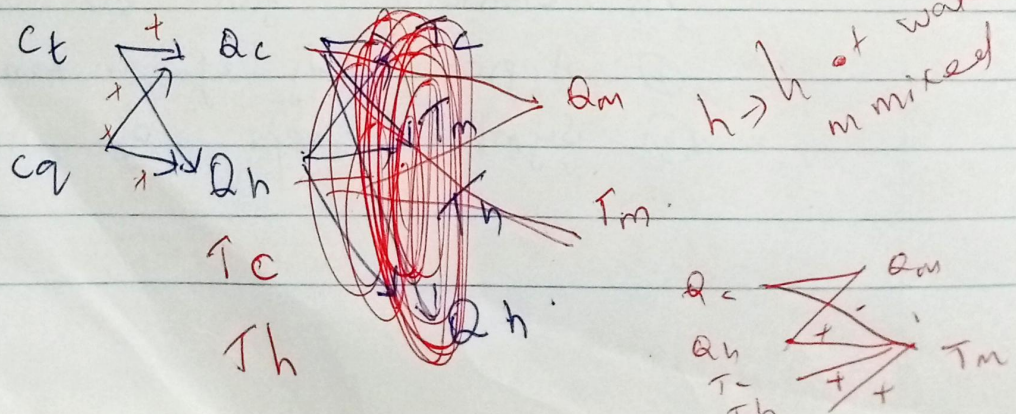
functions





another tab: control temp and waterflow
- help of above, modify something

control temp quantity



Lazy learning

① Store training data and waits to get a test tuple

② less training time, more prediction time

③ KNN, Case based Reasoning

eager learning

① constr general model during training

② more training time, less prediction time

③ Decision tree, Naive bayes, ANN

Generatic Algorithm

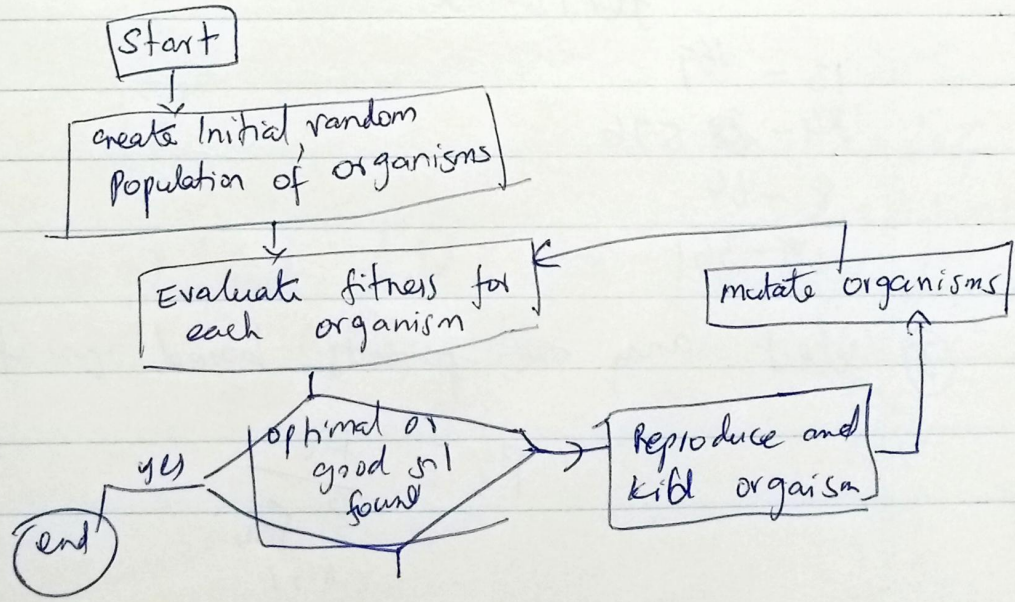
① belongs to evolutionary Alg

② adoptive heuristic search Algorithm

less attributes / examples / large → shortcuts

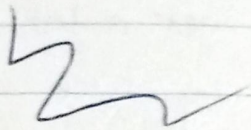
③ based on genetic and natural selection

④ used to generate high quality solution for an optimization problem



operations of Genetic algorithm

1. selection
2. crossover
3. mutation
4. Encoding



Example for genetic algorithm

$f(x) = x^2$ maximize this fun with x in intervals $[0; 31]$

1. Generate initial population at random (called gene)

N = 4

01101 (13)
11000 (24)
01000 (8)
10011 (19)

2. calculate fitness

$f(x) = x^2$

13 - 169
24 - 576
8 - 64
19 - 361

3. select any 2 parents based on fitness

$$P_i = \frac{F_i}{\sum_{j=1}^{} F_i}$$

No	initial population	Value x	$F(x)=x^2$	$p_i = \frac{F(i)}{\sum F(j)}$	Expected count $N \times p$
1	01101	13	~~13~~ 169	$\frac{169}{1170} = 0.14$	0.56
2	11000	24	~~24~~ 576	0.49	1.97
3	01000	8	~~8~~ 64	0.06	0.22 Remove
4	10011	19	~~19~~ 361	0.31	1.23
			1170		4.00

~~add to~~ change high to low in population

4. Cross over

canbe either one point / 2 point / n point

100 | 11101
101 | 10011
⟹ 10010011
⟹ 10111101

No	initial	cross over point	After crossover	x	$fx = x^2$
1	0110|1	4	01100	12	144
2	1100|0	4	11001	25	625
3	11|000	2	11011	27	729
4	10|011	2	10000	16	256
					1754

⑤ mutation

applied at each child after cross over (except first two large)

No	after crossover	After mutation	x	$f(x) = x^2$
1	[0]1100	11100	26	676
2	✓ 11001	11001	25	625
3	✓ 11011	11011	27	729
4	10[0]00	10100	18	324
				2354

K-mediods

- To overcome the drawback of k-means algorithm

- K means algorith is sensitive to outliers because such objects are far away from the majority of data, and thus when assigned to a cluster, they can dramatically desort the mean Value of cluster.

Algorithm K medoids

- Arbitarily choose k objects in D as initial Rep objects (OO seeds

- Repeat

① Assign each remainining obj to the cluster with the nearest Rep obj

② Randomly select nonrepresentativ obj O_{rand}

③ Compute total cost S of swapping representative obj O_j with O_{random}.

Until No change

Problem:

cluster the following points into 2 cluster

(2,0) (4,0) (7,0) (12,0), (14,0) (20,0)

Randomly choose 2 data points (2,0) & (20,0) as the representative objects.

Step1	(2,0)	(20,0)	cluster
(2,0)	0	18	C_1
(4,0)	2	16	C_1
(7,0)	5	13	C_1
(12,0)	10	8	C_2
(14,0)	12	6	C_2
(20,0)	18	0	C_2

Distance b/w (2,0) & (2,0) = $\sqrt{(2-2)^2+(0-0)^2} = 0$

Distance b/w (2,0) & (20,0) = $\sqrt{(20-2)^2+(0-0)^2} = \sqrt{18^2} = 18$

...

cost 1 = (0+2+5) + (8+6+0) = 21

Step2: choose a non rep obj (14,0) from 2nd clu

	(2,0)	(14,0)	cluster.
(2,0)	0	12	C_1
(4,0)	2	10	C_2
(7,0)	5	7	C_1
(12,0)	10	2	C_2
(14,0)	12	0	C_2
(20,0)	18	6	C_2

$\text{cost}_2 (0+2+5) + 2+0+6 = 15$

cost2 < cost 1 then swap (20,0) with (14,0) to form new set of representative objects

Step 3 Choose non Representative obj (12, 0) from 2nd cluster

	2,0	12, 0	cluster
20,	0	10	C_1
4,0	2	8	C_1
7,0	5	5	C_1
12,0	10	0	C_2
14 20	12	2	C_2
20,0	12	8	C_2

Un swap.

$\text{cost}3 = 0+2+5 + 0+2+8 = 7+10 = 17$

cost3 > cost 2 so. do not swap (14,0) with (12, 0)

step 4: Suppose select (4,0) as non representative obj

	(4,0)	(14,0)	cluster
2 0	2	12	C_1
4 0	0	10	C_1
7 0	3	7	C_1
12 0	8	2	C_2
14 0	10	0	C_2
20 0	6	6	C_2

cost 4 $2+0+3 + 2+0+6 = 13$

cost3 - unswap so use cost 2

cost 4 < cost 2

∴ swap (2,0) (4,0)

	(7,0)	(14,0)	cluster
(2,0)	5	12	C_1
(4,0)	3	10	C_1
(7,0)	0	7	C_1
(12,0)	5	2	C_2
(14,0)	7	0	C_2
(20,0)	13	6	C_2

Cost 5 = $(5+3+0) + 2+0+6 = 16$

& cost 5 7 cost 4 So no swap.

Final mediods are (4,0) && (14,0)

ENGINEERING COLLEGE

Clustering

Grouping into different groups
↓
similar data points

Similarity measure → using Distance measure

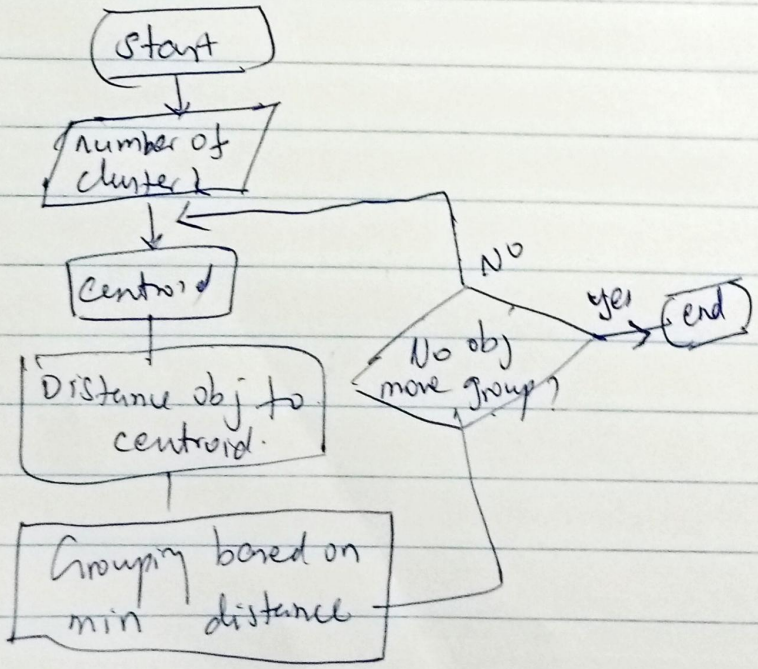
Euclidean distance

$$d(x, y) \sqrt{\sum_{i=1}^{p} (x_i - y_i)^2}$$

K means clustering

cluster n objects based on attributes into k partitions.

Algorithm



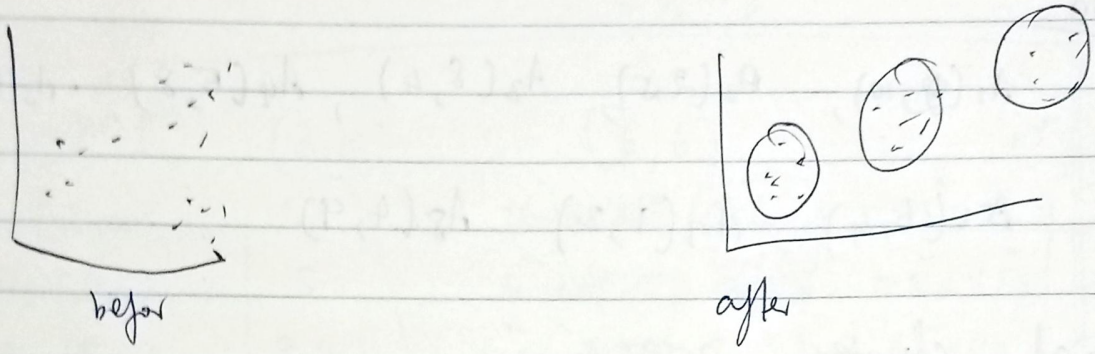
K mean

① unsupervised

② partitioned into k predefined distinct clusters

cluster

a collection of data pts exhibits certain similarities



Partitions

* Each datapoint belong to a cluster with nearest mean
* data pts ⇒ one cluster ⇒ highest similarity

Alg

① choose k

② Randomly select k datapts as cluster center (select farther as possible)

③ calculate distance b/w each data point and each cluster center

④ assign each dp to some cluster (nearest cluster)

⑤ comput center of new formed cluste

⑥ Repeat ③ & ④. upto center not changed datapts in same cluster, max iteration reached.

disadv

① require k

② not handle noisy data

Problem

$A_1(2,10)$, $A_2(2,5)$, $A_3(8,4)$, $A_4(5,8)$, $A_5(7,5)$

$A_6(6,4)$ $A_7(1,2)$ $A_8(4,9)$

initial cluster arar

	$A_1(2,10)$	$A_4(5,8)$	$A_7(1,2)$
A(2,10)	0 = 0	9+4 = 13 = 3.6	1+64 65 = 8.06
2,5	25 = 5	9+9 = 18 = 4.24	1+9 10 = 3.16
8,4	36+36 = 72 = 8.4	9+16 = 25 = 5	49+4 53 = 7.28
5,8	9+4 = 13 = 3.6	0 = 0	16+36 52 = 7.07
7,5	25+25 = 50 = 7.07	4+9 13 = 3.6	6+9 15 = 3.87
6,4	16+36 = 52 = 7.211	1+16 17 = 4.12	25+4 29 = 5.38
1,2	1+64 = 65 = 8.06	16+36 52 = 7.211	0 = 0
4,9	4+1 = 5 = 2.23	1+1 2 = 1.414	9+49 58 = 7.61

cluster 1 A1(2,10) cluster2 cluste 3

A3(8,4) A_2 (2,5)
A4 (5,8) A2 (1,2)
A5 (7,5)
A6 (6,4)
A8 (4,9)

1) iteration
new center pts (2,10) (6,6) 1.5 3.5

A1(2,10)	0	$16+16$	5.6	$0.5^2+6.5^2$ =	7 = C_1
A_2(2,5)	5	4^2+1	4.12	$0.5^2+1.5^2$ =	2 = C_3
A3(8,4)	8.4	2^2+2^2	2	$6.5^2+0.5^2$ =	7 = C_2
A4(5,8)	3.6	1^2+2^2	2.23	$3.5^2+4.5^2$ =	8 = C_2
A5(7,5)	7.07	1^2+1^2	1.414	$5.5^2+1.5^2$ =	7 = C_2
A6(6,4)	7.211	$0+2^2$	2	$4.5^2+0.5^2$ =	5 = C_2
A7(1,2)	8.06	5^2+4^2 25+16	6.403	$0.5^2+1.5^2$ =	2 = C_3
A8(4,9)	2.23	2^2+3^2	3.6	$2.5^2+5.5^2$ =	8 = C_2
		4+9			

cluster
C1
C3
C2
C2
C2
C2
C3
C2

clu 1
C_1
2,10
4,9

(3, 9.5)

clu 2
8,4
5,8
7,5
6,4

(6.5, 5.25)

C3
(2,5)
(1,2)

(1.5, 3.5)

after iteration 2

$c_1 (3, 9.5)$

$c_2 (6.5, 5.25)$

$c_3 (1.5, 3.5)$

Problem 2

find k=2

A(2,2)

B(3,2)

C(1,2)

D(3,1)

E(1.5, 0.5)

find 1, 4

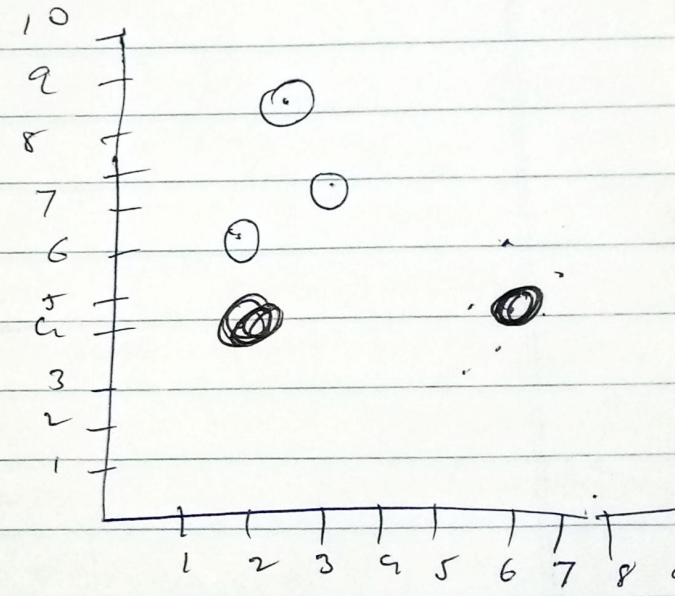
1	1	1
2	1.5	2.8
3	3.0	4.
4	.5	7
5	3.5	5
6	4.5	5
7	3.5	4.5

k mediod clustering

Objects in the cluster.

- iterative clustering alg. with iterates until each rep. obj is actually the mediod of most centric.

i	x	y
x_1	2	6
x_2	3	4
x_3	3	8
x_4	4	7
x_5	6	2
x_6	6	4
x_7	7	3
x_8	7	4
x_9	8	5
x_{10}	7	6



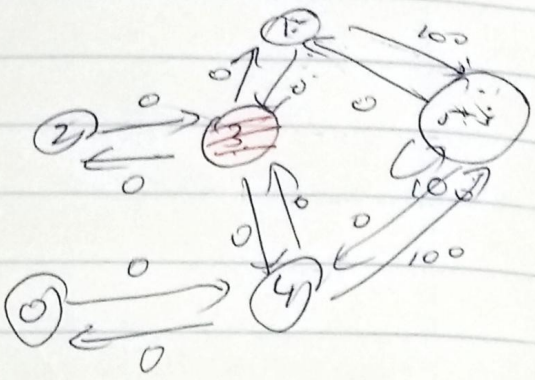
step 1: plot them & select 2 Rando obj.

C_1 C_2

(3, 4) (7, 9)

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

$\gamma = 0.8$



$S = 3$

action = (2, 1, 4)

$r =$ Imm rewar $(3,1) = 0$

$S' = 1$

$$\hat{Q}(3,1) = 0 + 0.8 \max(\hat{Q}(s',a'))$$

$$\hat{Q}(1,\cdot)$$

$$\max(0, 100)$$

Q

1 — 0 — 3

1 — 100 — 5

$$\hat{Q}(3,1) = 0 + 0.8 \times 100 = 80.$$

$S = 1$

action (5, 3)

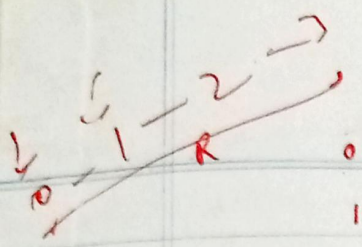
$r = 100$

$S' = 5$

$$\hat{Q}(1,5) = 100 + 0.8 (\max(s',a'))$$

$\max(0,0)$

$$\hat{Q}(1,5) = 100$$



	0	1	2	3	4	5
0	-1	-1	-1	-1	0	-1
1	-1	-1	-1	0	-1	100
2	-1	-1	-1	0	-1	-1
3	-1	0	0	-1	0	-1
4	-1	0	0	1	-1	100
5	-1	0	-1	-1	0	100

0 → 4 = 0

1 - 3 = 0

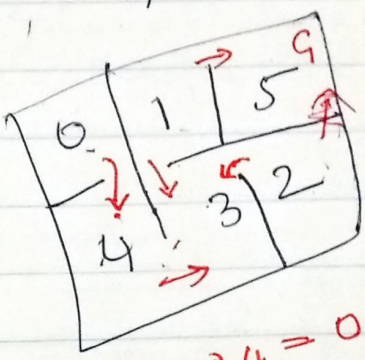
1 - 5 = 100

2 - 3 = 0

3 - 1

3 - 2

4 - 0



Q =

	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	0	0	0	100
2	0	0	0	0	0	0
3	0	0	0	0	0	0
4	0	80	0	0	0	0
5	0	0	0	0	0	0

Q(3, 4)

0 → move → 4 = 0

So Reward -1 -1 -1 -1 0 1

1 → 5 (Goal) = 100

1 → 3 = 0

Avai_act = [3 5]

Q(1, 5)

update (1, 5, 80)

0 1 2 3 4 5 max index

max index.

c_s = 2

act = 5

update (2, 5, 0.8)

max index

0 1 2 3 4 5

max index.

Q in 3 Random.

Q(1, 5) = 100 + 80x0.8

= 100.

maxVal = Q[3, 2) = 0

Q[2, 3] = 0 + 0.8 = 0

Q[2, 3] = 0

3 5

Current state = 1 -1 -1 -1 0 -1 100

action [3, 5]

Rand

action = 5

update (1, 5, 0.8)

Q index 0 1 2 3 4 5 Random

max index 5

max value = Q(5, 5)

= 0

Q(1, 5) = -100 + 0 = 0

= 100

Current st = 4 -1 0 0 -1 -1 100

action = 1 2 5

auto = 2

update (4, 2, 0.8)

0 1 2 3 4 5

Q in 0 0 0 0 0 0 Random.

Ran index

5

~~Q~~ max value = Q(5, 5)

Q(

Curre s = 4

action = 1 2 5

max index

5 sing value

5

~~Q~~ max value(1, 5) = ~~100~~ 0 + 8 0.

Q[4, 1] = 100

(8)

Reinforment Learing

dynamic Prog used to solve optimal Problem

⇒ Agent act in its environment a can learn & choose optimal action

To achieve goal

Appli: mobile Robot
opt operation in factories
learning to play board games

Agent ⇒ perform Action ⇒ Trainer → give penalty / reward

+ +ve Reward ⇒ win
- -ve Reward (penalty) ⇒ loss
0 all other states

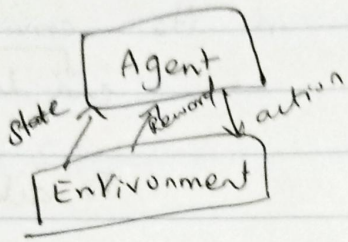
Aim

Task of the agent ⇒ learn from this
choose seq action that produce
greatest cumulative reward.

ex Agent ⇒ Robot
set of sensors ⇒ observe its state
actions ⇒ move forward, turn
policy ⇒ choosing action that achieve its goal.

ex goal move near to battery charger when its battery level is low

Policy : max cumulative reward



$s_0 \xrightarrow[r_0]{a_0} s_1 \xrightarrow[r_1]{a_1} s_2 \xrightarrow[r_2]{a_2} \cdots$

Goal $r_0 + \gamma r_1 + \gamma^2 r_2 + \cdots \quad 0 \le \gamma \le 1$

$\pi : S \to A$

Reinforcement learning differ from othe fun-approx. tasks.

① Delayed Reward

Current State s

optimal action $a = \pi(s)$

prev case trainging ex is in the form of

$\langle s, \pi(s) \rangle$

Agent faces the problem of

temporal credit Assignment

Determining which of action in its sequence are to be credited with producing eventual rewards.

② Exploration

- Reinforment learning decide
  - Exploration – gather new information. unknown state & action
  - Exploitation – already learned yield highest Reward.

③ Partially observed states

- use partial information, use prev observation select best next state

ex camera ⇒ observe front not balle
⇒ use prev info select next action.

④ life long learning

⇒ learning alway new things.

Learning Task

Markov Decision Proces

Assume

S - finite set of states
A - set of Actions
for each time t
$s_t \in S$
$a_t \in A$
Receive immediate Reward r_t
and state changes to s_{t+1}

markov assumption

$$s_{t+1} = \delta(s_t, a_t)$$
$$r_t = r(s_t, a_t)$$

r_t, s_{t+1} ⇒ only on current state and ation

δ and r ⇒ non deterministic
↳ not necessary known to agent

Agent's learning task
learn a policy $\pi : S \to A$

∴ for selecting next action a_t

base on s_t $\pi(s_t) = a_t$

Policy that produce ~~re~~ greater possible cumulative ~~Reward~~ value $V^\pi(s_t)$

$$V^\pi(s_t) = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots$$
$$= \sum_{i=0}^{\infty} \gamma^i r_{t+i}$$

$a_t = \pi(s_t)$

$a_{t+1} = \pi(s_{t+1})$

$0 \leq \gamma < 1$

set $\gamma = 0$ only immediate reward

only consider

so γ = near to 1

$V^{\pi}(s)$ = discounted cumulative Reward

$\sum_{i=0}^{h} r_{t+i}$ ⇒ finite horizon reward

Agent learn the policy π that max

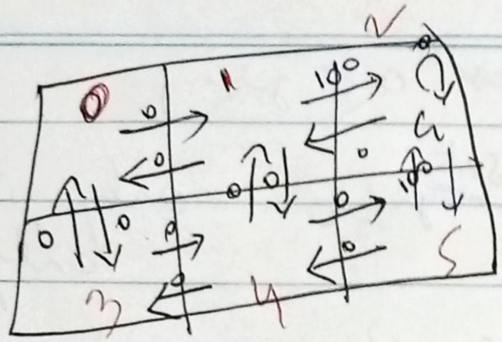
$V^{\pi}(s)$ ∀ all states s.

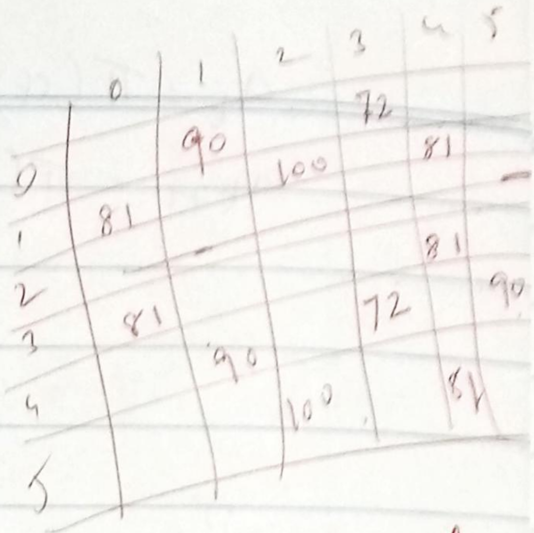
∴ optimal policy

$$\pi^* = \arg\max_{\pi} V^{\pi}(s) \quad \forall(s)$$

G - absorbing state

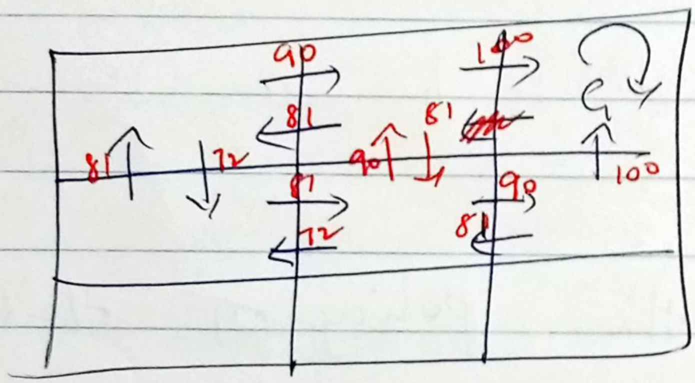
r(s,a) immediate Reward Values



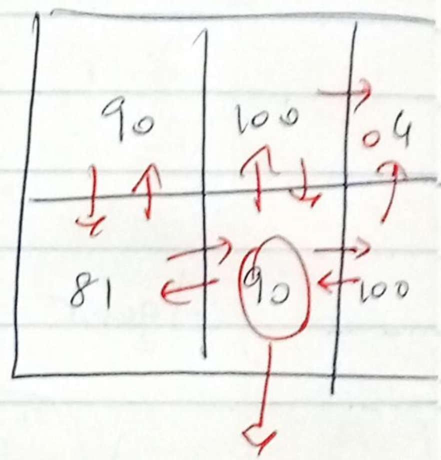


$\gamma = 0.9$

Q(s, a)



V*(s) Values

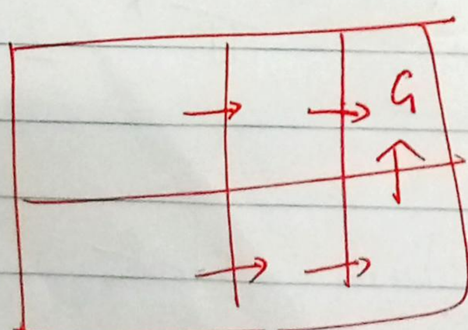


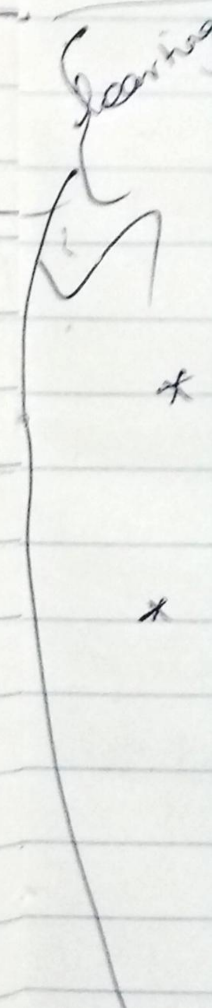
$\gamma = 0.9$

maximal Q values.

$V^*(s) = 0 + \gamma\, 100$

$= 90$





Q Learning — optimal policy

$\therefore \Pi^* : S \to A$

A:
Action 'a' maximize sum of ,immede reward r(s,a)

$$\Pi^*(s) = \underset{a}{\operatorname{argmax}} \left[r(s,a) + \gamma V^*(\delta(s,a)) \right]$$

$\delta(s,a) \Rightarrow$ Resulted ~~Select~~ state from 's' to 'a'

learning

$V^* \Rightarrow$ Provided perfect knowledge of immediate reward function r and state trans of

* Agent should prefer S_1 over S_2
where $V^*(S_1) > V^*(S_2)$
Cumulative Reward

* The optimal action in state s is action 'a' that maximizes the sum of immediate reward $r(s,a)$ + value V^* of immediate successor state, discounted by γ

$$\Pi^*(s) = \underset{a}{\operatorname{argmax}} \left[r(s,a) + \gamma V^*(\delta(s,a)) \right]$$

Q function

$$Q(s,a) = r(s,a) + \gamma V^*(\delta(s,a))$$

next imme state

rewrite

$$\pi^* s = \underset{a}{\operatorname{argmax}}\, Q(s,a)$$

algorithm learning Q Training Rule to learn Q

$$V^*(s) = \max_{a'} Q(s,a')$$

$$Q(s_t,a_t) = r(s_t,a_t) + \gamma \max_{a'} Q(s_{t+1}, a')$$

rewrite this

$$Q(s_t,a_t) = r(s,a) + \gamma \max_{a'} Q(\delta(s,a), a')$$

$\hat{Q}$ => large table (learner current app. to Q)

entry of each state - action pair

$\hat{Q}(s,a)$

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

$\hat{Q}$ values for new state s' refine $\hat{Q}(s,a)$ for the prev state s

Q learning algorithm

For each s, a initialize table entry $\hat{Q}(s,a)$ to zero

Observe the current state s

Do forever:

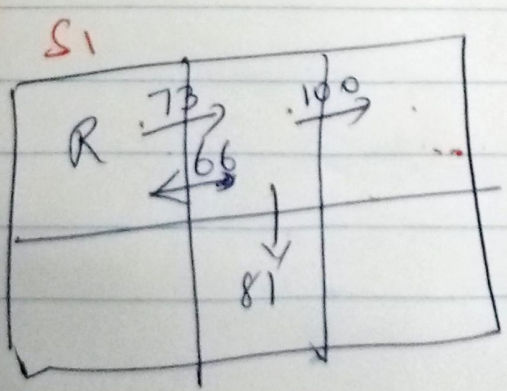
{
- select an action a and execute it
- Receive immediate reward r
- Observe new state s'
- updata table $\hat{Q}(s,a)$

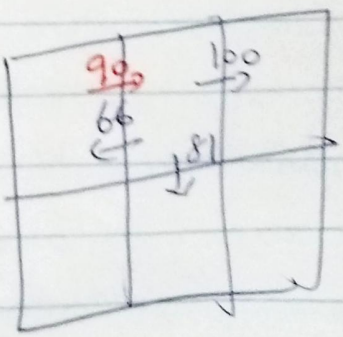
$$\hat{Q}(s,a) = r + \gamma \max_{a'} Q(s',a')$$

- $s = s'$

}

$0 \leq \gamma < 1$





$$\hat{Q}(s_1, a_{right}) = r + \gamma \max_{a'} (s_2, a')$$

$$= 0 + 0.9 \times \max(66, 81, 100)$$

$$= 0.9 \times 100 = 90.$$

Q Table updation 2 properties

① $\hat{Q}$ values never decrease

② $\forall (s, a, n) \quad \hat{Q}_{n+1}(s, a) \geq \hat{Q}_n(s, a)$

② Every $\hat{Q}$ value will remain in interval b/w Zero and its true Q Value

$$\forall (s, a, n) \quad 0 \leq \hat{Q}_n(s, a) \leq Q(s, a)$$

GA

Hyp space search

* GA → seek max fit hypothesis
→ child Rapidly differ from parent.
- less likely fall into same local minima.

One Problem GA

crowding

↳ some indi highly fit than others

∴ copy of indv & very similar indi avail in population. (in large fraction)

dis: Reduce diversity.

avoid this

① use selection instead of
- tournament selection
- Ranking

② fitness sharing
fitness of indi is reduced by presence of others.

③ Restrict the kind of indi allowed to recombine to form offspring.

model evaluation and learning.

① Lamarckian Evaluation

proposed that evaluation over many generations was directly influenced by the experience of indivual organism during their lifetime.

indi : avoid toxic food. (failure)

Baldwin effect

* when there is no indiv leaning the population failed to improve over time

* when learning is applied in early stages the population contains many individuals with many trainable weights

* it achieved high fitness ~~with the number of~~

Parallezing Genetic algorithm.

Coarse grain

- To parallezion subdivide the population in group of indiv (demes)
- each deme assign to diff Compu. node.
- GA search is performed at each node.
- commu. and cross-fertilization between demes occur less freq.
- Tranfer b/w deme occurs by migration proces.
- It reduces crowding problem.
- fall into global optima.

Fine grained

- assign one processor per indivi in the population.
- Recombin ~~took~~ take place among neighbour indiv

Q Learning

= quality learning

policy $Q^{\pi}(s_t, a_t)$ policy.

$Q^{\pi}(s_t, a_t)$

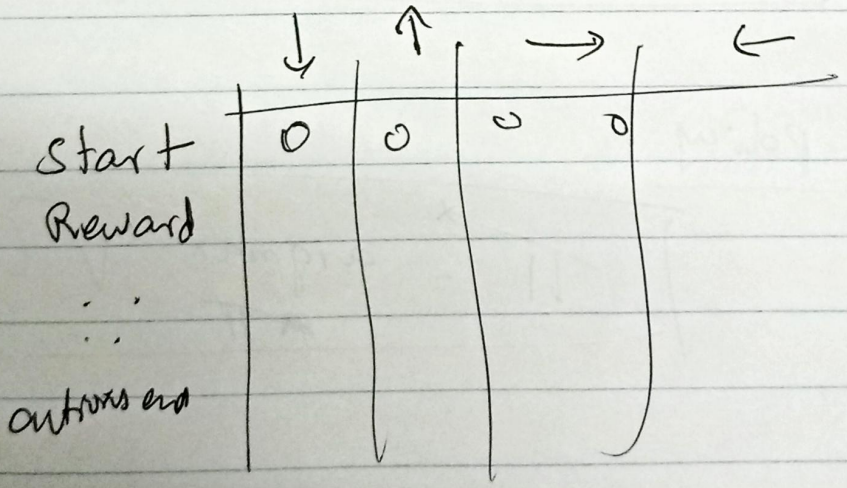
Undertaken by Agent (In a given state what is the best possible action)

Reward ⇒ scalar quantity - measured

Price Rs, E, $ +ve ⇒ ↑ , -ve ⇒ ↓

Q function ⇒ $Q^{\pi}(s_t, a_t) = r + \gamma \max Q(s_t', a_t')$

Q Table ⇒



n x m

Markov decision Process (MDP)

$$s_{t+1} = \delta(s_t, a_t)$$
$$r_t = r(s_t, a_t)$$

} depends on current state and action.

Policy

$\pi : S \to A$ Rule to select next action

$\pi(s_t) = a_t$

provide max Reward at end. Cumulative

$V^{\pi}(s_t)$ discount cumulat Reward

$$V^{\pi}(s_t) = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2}$$

$$\boxed{V^{\pi} S_t = \sum_{i=0}^{\infty} \gamma^i r_{t+i}}$$

optimal policy

$$\boxed{\pi^* = \underset{\pi}{\operatorname{argmax}}\ V^{\pi}(s)}$$

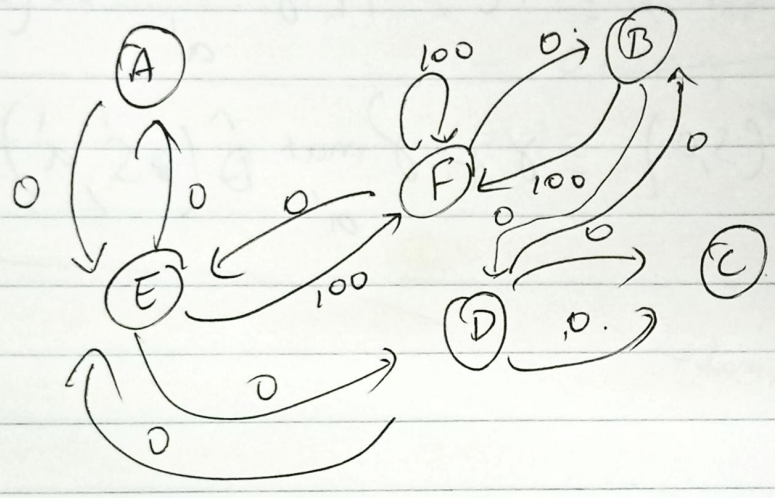
Q learning

How agent learn optimal policy π^*

* only seq rewards.

- evaluation function V^*. (optimal cumulative)

agent prefer s_1 over s_2 wher $V^*(s_1) > V^*_{s_2}$.



start B

start D

* policy: consider actions & choose V^*

policy

$$\pi^*(s) = \underset{a}{\operatorname{argmax}} \left(r(s,a) + \gamma V^*(\delta(s,a)) \right)$$

immediate successor state

Q function.

$$Q(s,a) = r(s,a) + \gamma V^*(\delta(s,a))$$

$$\pi^*(s) = \underset{a}{\operatorname{argmax}}\, Q(s,a)$$

rewrite eqn ①

Algorithm Learning Q ⇒ learn seq of imme. reward r spread over time.

- it is iterative approach

Relationship b/w Q and V

$$V^*(s) = \max_{a'} Q(s,a')$$

⇒ rewrite eqn 1 $Q(s,a) = r(s,a)$ to

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

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

$\hat{Q}$ estimate

Q learning alg

For each s, a initialize table $\hat{Q}(s,a) = 0$

Observe the current state s

Do forever

{ select action a & execute

Receive immediate reward r

observe new state s'

update table entry $\hat{Q}(s,a)$

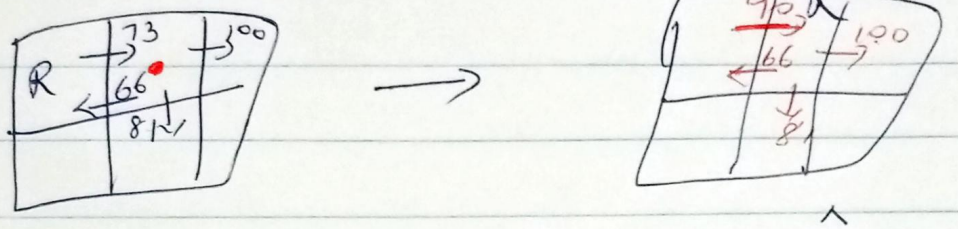
$\hat{Q}(s,a) = r + \gamma \max_{a'} \hat{Q}(s', a')$

$s = s'$

}

$\rightarrow \gamma V^{\pi}(\delta(s,a))$

Ex



$$Q(s_1, \text{Right}) = r + \gamma \max_{a} \hat{Q}(s_2, a)$$

$$= 0 + 0.9 \max\{66, 81, 100\}$$

$$= 90.$$

Unit-5

Learning set of Rules:

① - Use decision tree
- one Rule for each leaf node

② use genetic alog
encode each rule set as bit strin
use genetic operation to explore

Learn set of Rules ⇒ differ from above

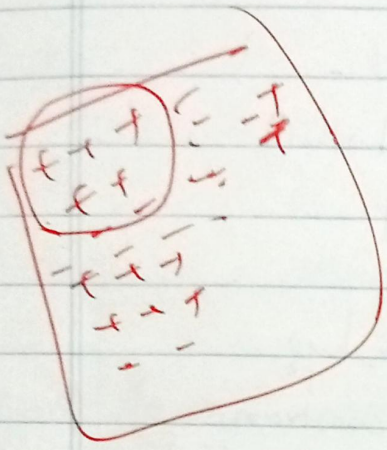
① Learn first order rules that contain Var
② use seq. Covering alg.:- that Var.
learn one rule at a time
and incre grow the first set of

ex ⇒ target fun parent (x, y) y is F/M of
Ancestor (x, y) y is Ans of x

if parent (x, y) Then Ancestor (x, y)
if paren (x, z) ∧ Ancestor (y, z) Then Ancestor (x, y)

use Prolog

Seq covering Alg — Variable free



- Family of algorithms for learning rule sets based on strategy on learning one rule
  - Removing the data it covers
  - iterative this process

```
Sequential_covering (Target_att, Attr, Examples, Threshold)
{   Learned_rule = {}
    Rule = Learn_one_Rule (Target_att, Attr, Examples)
    while performance (Rule, Examples) > Threshold
    {
        Learned_rules = Learned_rules + Rule
        Examples = Examples − {ex. correctly classified by rule}
        Rule = Learn_one_rule (Target_att, att, Examples)
    }
    Learned_rules = sort_learned_rule (Target_att, att, exam)
                    according to PEFORMANCE over exam
    return Learned_rules
}
```

Learn_one_Rule (Target_att, Att, exa, k)

{

Best_hyphothes = {ϕ, ϕ, ϕ ...}

Candidate_hyp = {Best_hyphothes}

whil candidate_hyp != empty

{

1. Generate specific_candidate_hyp.

all_cons = set all constramis of (a=V)

where a = attri
V = value

new_can_s =

for each h in candidat_hyp

for each c in all_cons

create specializ of h by adding c

Remove from new_cons_s of duplicates,

inconsis, not max specif

2. update Best_hyphothesis

for all h in new_can_s

if PErformance (h, examply Tar_att)

> PErformance (BEst_hyp, examp to_att)

Best_hyphothesis = h

3. update candidate_hyphotheses

Candidate_hyphothesis = k best member

of new_can_hyp according to

the performance measure

}

Retur a Rule of the form

If Best_hyp Then prediction

PERformance (h, Example, Target_attr)

h_example = subset of Examples that match h

return (- Entroply (h_examples)

General to specific beam search (Learn one Rule)

use greedy depth first search with no backtracking

① select only the mast promising branch in a tree at each step.

- start general rule
- greedily adding second attr test

attr-value pair yeild best performance

If

Then play_ten = yes

if wind = weak then play_ten = s

if wind = strong then play_ten = s

if humidity = normal then play ten = s

if humidity = normal wind = weak then play_ten = s

if humity = normal wind = strong then play_ten = s

if humi = normal outlook = sunny then play_ten = s

(stop local minima)

disadv: supoptimal choice will be made at any

use beam search

↓ Alg maintain k best candidates and the resulting set is again reduced to k most promising members

Beam search used by CN2 prog

	ID3	CN3
1	Simultaneous covering	seq covering
2	each step ~~set~~ select attri	attr-value pair
3	specific to general	general to specific
4	example-driven approach	generate & ~~st~~ test approach

Measure the performance Learn-one-Rule.

① Relative Frequence.

$$\frac{n_c}{n}$$

n = No. of examples the rule matches

n_c = No. of examples classifies correctly.

step - ② m-estimate of accuracy

$$\frac{n_c + mp}{n + m}$$

n - No. of examples matched

n_c = Correctly predicted by rule.

P = Prior prob. (12/100 predi by rule $P = 0.12$)

m = weight, (equal no of examples for weighting their prior P)

③ Entropy.

$$- \text{Entrophy}(S) = \sum_{i=1}^{c} P_i \log_2 P_i .$$

Learning First order Rules (ILP)

inductive logic Programing

use Prolog

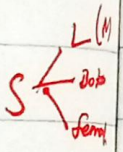
- Data described by attributes

1) name, mother, Father, male, female

Target attr: Daughter

S - B - F

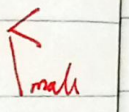
B - M



N_1 = Sharon M_1 = Louise F_1 = Bob M_1 = F fem = T

N_2 = Bob M_{02} = Nor fa_2 = Victor Ma_e = T fem = F

Daughter(1, 2) = True



$$\text{Daughter}_{1,2} = \text{True}$$

Rule

IF father$_1$ = bob $\cap$ Name2 = bob $\cap$ female$_1$ = True

then Daughter$_{1,2}$ = True

general Rule

If Father(x, y) $\cap$ female(x) Then

Daughter(x, y)

x, y = Variable

Terminology

Constant : Bob

Variable : x, y

Predicate Sym : Married, Greater_than. (T/F value)

function Sym : age

Literal : any predicate or negation apply — married(Bob), ¬ greaterthan(age(x), 20)

contain ¬ negative literal for the literal

(F)

clause : any disjunction of literals.

Hornclause : clause contain at most one positive literal.

$$H \cup \neg L_1 \cup \neg L_2 \cdots \cup \neg L_n$$

$$B \cup \neg A = B \leftarrow A$$

$$\neg(A \cap B) = \neg A \cup \neg B$$

$$\therefore \boxed{H \leftarrow L_1 \cap L_2 \cdots \cap L_n}$$

H = head (or) consequent

$L_1 \cap L_2 \cdots L_n$ = antecedents (or) body

ground literal : literal does not contain any Var

¬ Female (Joe)

Learning sets of first order Rules : FOIL

- Similar Seq covering and Learn_one_Rule alg

FOIL

- hypothesis Learned ⇒ set of first order rules

each Rule - Horn clause ⇒ 2 exceptions

① Foil ⇒ more restricted than general Horn clause (no literals)
⇒ Rules more expressive than Horn clause
(Literals appear)
⇒ Seeks only rules that predict target literal is true
⇒ hill climbing search (not beam search)
⇒ each outerloop add one new rule to -

Outer { same as prev gone new rule
inner loop add best literal & pre condition

Foil (Target_pred, pred, Examples)
{
Pos = Example (targ_pred = True)
neg = Example (targ_pred = false)
Learned_rules = [ ]
while pos do
{ ⇒ Learn a new rule
newR = rule predict Target predicate with no Precondition.
newRneg = Neg
while NewRneg
{
⇒ add literal to newrule
can_lits = generate candidate new literals based on ~~Pre condit~~ predicates
Best_lit = $\underset{L \in cand\_lits}{\text{argmax}}$ FoilGain (L, new Rule)
add best_lit to precond new Rule
newRneg = Subset of NewRneg that satisfy newrule Precondin
}
Learned_rules = Learned_rules + newRule
Pos = Pos − (member of Pos Coverd by newRule)
}
Returned Learned_rules
}

alg
rules
clause (no literals)
clause
search)

(specific to general)

Outloop ⇒ disjunctive hyp. conju of literals

innerloop ⇒ fine-grained, general to specific hill climbing

⇓

⇒ add literals one at a time (until all negative examples removed)

in preconditions

Foil

* ~~Inner~~ Inner loop add literals in rules pre conditions

Seq-covering

* Perfomance measer

Foil gain

$$\text{Foilgain}(L,R) = t\left(\log_2 \frac{P_1}{P_1+n_1} - \log_2 \frac{P_0}{P_0+n_0}\right)$$

L - candidate literal

R - Rule

R' - Rule created by adding L to R

P_0 - No. of +ve bindings of rule R

n_0 = No of -ve " " "

P_1 = No. of +ve binding of R+L

n_1 = No. of -ve binding of R+L

t = No. of +ve bindings of rule R that are still covered after adding literal L to R

Foil => (1) extends CN2

(2) general to specific => add new literal to rule precondition.

(3) foil gain to select new literals.

(4) ~~learn~~

Induction as Inverted Deduction

* Give data D
* Some background knowledge B
* Training ex of the form $\langle x_i, f(x_i) \rangle$

Learning

discovering h such that classification $f(x_i)$ of each training instance x_i ~~follows~~ and any background knowledge B

$$(\forall \langle x_i, f(x_i) \rangle \in D)\ (B \cap h \wedge x_i) \vdash f(x_i)$$

of rule R
" " "
of R+L
of R+L

R
adding

example

x_i : Male(Bob), Female(Sharon), Father(Sharon, Bob)

$f(x_i)$: Child(Bob, Sharon)

B : Parent(u, v) ← Father(u, v)

Two hypothesis

h_1 : child(u, v) ← Father(v, u)

h_2 child(u, v) ← Parent(v, u)

for hypothesis h_2 use background knowledge

h_1 use $h_1 \wedge x_i$

$B \wedge h_2 \wedge x_i$

h_2 use predicates (parent) not in x_i

Constructive induction.

New predicates can be introduced into hypothesis even when the predicates is not present in the original description of instance x_i.

Process of augmenting set of predicates based on background knowledge is called

Induction is inverse of deduction

Inverse entailment operator $= O(B, D)$

$O(B, D) = h$ such that use D and B

$\forall (x_i, f(x_i)) \in D) \; (B \wedge h \wedge x_i) \vdash f(x_i)$

Features to formulate learning task

① finding general concept that matches a given set of training exmple w.o. background

② use domain-specific background info. B

h fits $(x, f(x_i))$ as long as $f(x_i)$ followed deductively from $B \wedge h \wedge x_i$

③ use background search h

inductive logic prog: difficulties

① not allow noisy data

② first orde logic => so expressive large hyp => satisfy above formula

③ complexity of hyp search increases as background knowledge B is increased

Resolution Rule

Inverting Resolution

Let L be an arbitary propositional literal P and R be arbitary propositional clauses.

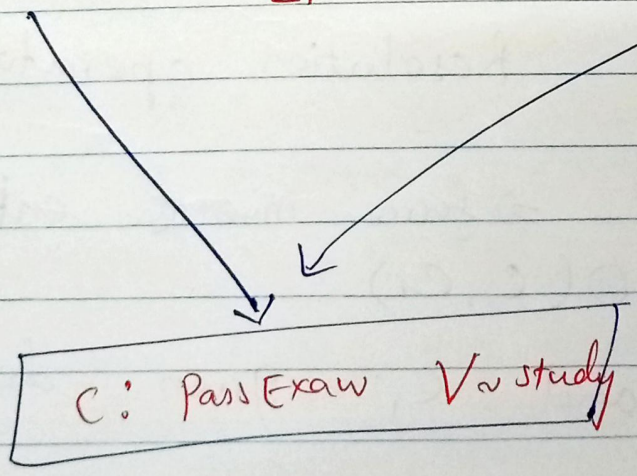
The resolution Rule is

$$\frac{P \vee L \qquad \sim L \vee R}{P \vee R}$$

L (or) ~L any one false. ∴ Result PVR

C1 : PassExam V ~knowmaterial
L1

C2 : knowmater V study
~L1



derived

C conclusion

$C_1 - \{L\}$ = Pass Exam

$C_2 - \{\sim L\}$ = $\sim$ study

Example 2

$C_1 = A \vee B \vee C \vee \sim D$

$C_2 = \sim B \vee E \vee F$

$C : A \vee C \vee \sim D \vee E \vee f$

General form of propositional Resolution operator

$$\therefore C = C_1 - \{L\} \cup C_2 - \{\sim L\}$$

Invert Resolution operator

$\Rightarrow$ form inverse entailment operator $O(C, C_1)$

C and C_1 given derive C_2

$C_1 = B \vee D$

$C = A \vee B$

$C_1 \cap C_2 \vdash C$

① Any Literal in C not in C_1 must present C_2 — A

② Literal in C_1 not in C.

D

$\therefore \sim D$ must present in C_2

$\therefore$ $C_2 = A \vee \sim D$ or $C_2 = A \cup \sim D \vee B$

Inverse Resolution Operator

C_1 : Pass Exam $\vee \sim$ knowmater.

C_2 : knowMat $\vee \sim$ study

C : pass exam $\vee \sim$ study

* ~~Condi~~ many clause u defined use Shortest

~~Inverse~~ Inverse Resolution Operator } $C_2 = (C - (C_1 - \{L\})) \cup \{\sim L\}$

Inverting Resolution: First order case

Substition $\theta = \{x/Bob, y/z\}$

x is replaced by Bob
y is replaced by z

L is Father (x, Bill) ⇓ Bob. $\theta = \{x/Bob, y/z\}$

$L\theta$ = Father (Bob, Bill)

- θ is ~~Unifi~~ Substitution

θ = θ = Unifying substition

Two literals L_1 and L_2

$L_1\theta = L_2\theta$

L_1 = Father (x, y)
L_2 = Father (Bill, z) $\theta = \{x/Bill, z/y\}$

$L_1\theta = L_2\theta$ = Father (Bill, y)

Resolution operator first order form

$$C = (C_1 - \{L_1\})\theta \cup (C_2 - \{L_2\})\theta$$

Inverting Resolution: First order case

$$C = (C_1 - \{L_1\})\theta_1 \cup (C_2 - \{L_2\})\theta_2$$

$$C - (C_1 - \{L_1\})\theta_1 = (C_2 - \{L_2\})\theta_2$$

$$(C - (C_1 - \{L_1\}\theta_1))\theta_2^{-1} = C_2 - \{L_2\}$$

by def Resolution Rule

$L_2 = \neg L_1 \theta_1 \theta_2^{-1}$

$$C_2 = (C - C_1 - \{L_1\}\theta_1)\theta_2^{-1} \cup \{\neg L_1 \theta_1 \theta_2^{-1}\}$$