CMR INSTITUTE OF TECHNOLOGY

USN | | | | | | | | | |

Internal Assesment Test –III, January 2022

Sub:	MACHINE LEARNING							Code:	18MCA53
Date:	24-01-2022	Duration:	90 mins	Max Marks:	50	Sem:	V	Branch:	MCA

		Marks	OBE	
	Answer Any 5 Questions		CO	RBT
	Part-I			
Q1	Illustrate Bayesian Belief network with the help of an example.	10	CO4	L2
	or			
Q 2	Explain Expectation Maximization algorithm.	10	CO4	L2
	Part-II			
Q 3	Write short note on: a)Sample Error b) True Error c) Variance d) Expected value e) Confidence Interval	10	CO5	L2
	or			
Q 4	What is instance-based learning? Explain Case-Based reasoning method.	10	CO5	L1
	Part-III			
Q 5	Discuss the method of comparing two algorithms. Justify with paired-t method.	10	CO5	L3
	or			
Q 6	Discuss Q-learning algorithm with the help of an example.	10	CO5	L3
	Part-IV			
Q7	Explain the Distance Weighted Nearest Neighbor Algorithm.	10	CO5	L2
	or			
Q8	Explain the locally weighted linear regression.	10	CO5	L2
	Part-V			
Q9	What is reinforcement learning algorithm? Explain.	10	CO5	L2
	or			
Q10	Explain the Radial basis function (RBF) network.	10	CO5	L3

CMR INSTITUTE OF TECHNOLOGY

USN | | | | | | | | | |

CMRIT

Internal Assessment Test 3 – December. 2020

Sub:	MACHINE LEARNING							**Sub Code:**	18MCA53
Date:	15-12-2020	Duration:	90 min's	Max Marks:	50	**Sem**	5th	**Branch:**	**MCA**

Note : Answer FIVE FULL Questions, choosing ONE full question from each Module

	PART I	MARKS	OBE CO	OBE RBT
1)	(see below)	10	CO4	L5

- Consider the setting in which we wish to learn a nondeterministic (probabilistic) function f : X → {0, 1}, which has two discrete output values.
- We want a function approximator whose output is the probability that f(x) = 1. In other words, learn the target function f ` : X → [0, 1] such that f ` (x) = P(f(x) = 1)

How can we learn f ` using a neural network?

- Use of brute force way would be to first collect the observed frequencies of 1's and 0's for each possible value of x and to then train the neural network to output the target frequency for each x.

What criterion should we optimize in order to find a maximum likelihood hypothesis for f' in this setting?

- First obtain an expression for P(D|h)
- Assume the training data D is of the form D = {(x1, d1) . . . (xm, dm)}, where di is the observed 0 or 1 value for f (xi).
- Both xi and di as random variables, and assuming that each training example is drawn independently, we can write P(D|h) as

$$P(D \mid h) = \prod_{i=1}^{m} P(x_i, d_i \mid h) \qquad \text{equ (1)}$$

Applying the product rule

$$P(D \mid h) = \prod_{i=1}^{m} P(d_i \mid h, x_i) P(x_i) \qquad \text{equ (2)}$$

The probability P(di|h, xi)

$$P(d_i|h, x_i) = \begin{cases} h(x_i) & \text{if } d_i = 1 \\ (1 - h(x_i)) & \text{if } d_i = 0 \end{cases} \qquad \text{equ (3)}$$

Re-express it in a more mathematically manipulable form, as

$$P(d_i|h, x_i) = h(x_i)^{d_i}(1 - h(x_i))^{1-d_i} \qquad \text{equ (4)}$$

Equation (4) to substitute for P(di |h, xi) in Equation (5) to obtain

$$P(D|h) = \prod_{i=1}^{m} h(x_i)^{d_i} (1 - h(x_i))^{1-d_i} P(x_i) \qquad \text{equ (5)}$$

We write an expression for the maximum likelihood hypothesis

$$h_{ML} = \underset{h \in H}{\operatorname{argmax}} \prod_{i=1}^{m} h(x_i)^{d_i} (1 - h(x_i))^{1-d_i} P(x_i)$$

The last term is a constant independent of h, so it can be dropped

$$h_{ML} = \underset{h \in H}{\operatorname{argmax}} \prod_{i=1}^{m} h(x_i)^{d_i} (1 - h(x_i))^{1-d_i} \qquad \text{equ (6)}$$

It easier to work with the log of the likelihood, yielding

$$h_{ML} = \underset{h \in H}{\operatorname{argmax}} \sum_{i=1}^{m} d_i \ln h(x_i) + (1 - d_i) \ln(1 - h(x_i)) \qquad \text{equ (7)}$$

Equation (7) describes the quantity that must be maximized in order to obtain the maximum likelihood hypothesis in our current problem setting.

2)

Step 1: Convert the data into a frequency table.

Play	Frequency
Yes	9
No	5

Outlook	Yes	No
Sunny	2	3
Overcast	4	0
Rain	3	2

Temperature	Yes	No
Hot	2	2
Mild	4	2
Cool	3	1

Humidity	Yes	No
High	3	4
Normal	6	1

Wind	Yes	No
Strong	3	3
Weak	6	2

10 CO4 L5

Step 2: Create Likelihood table

Play	Frequency	Likelihood
Yes	9	9/14
No	5	5/14

Outlook	Yes	No
Sunny	2/9	3/5
Overcast	4/9	0/5
Rain	3/9	2/5

Temperature	Yes	No
Hot	2/9	2/5
Mild	4/9	2/5
Cool	3/9	1/5

Humidity	Yes	No
High	3/9	4/5
Normal	6/9	1/5

Wind	Yes	No
Strong	3/9	3/5
Weak	6/9	2/5

Step 3:
New instance
X= (Outlook = Sunny, Temperature = Cool, Humidity = High, Wind = Strong)
Play Tennis = ?

P(X |Play = Yes) = P(Play = Yes) * P(Outlook = Sunny |Yes) * P(Temperature = Cool|Yes) * P(Humidity = High|Yes) * P(Wind = Strong|Yes)
= 9/14 * 2/9 * 3/9 * 3/9 * 3/9
= 0.0053

P(X|Play = No) = P(Play = No) * P(Outlook = Sunny |No) * P(Temperature = Cool|No) * P(Humidity = High|No) * P(Wind = Strong|No)

= 5/14 * 3/5 * 1/5 * 4/5 * 3/5
= 0.0206

So: 0.0206 > 0.0053
Result : X : PlayTennis = No

3)

- The naive Bayes classifier makes significant use of the assumption that the values of the attributes a1 . . .an are conditionally independent given the target value v.
- This assumption dramatically reduces the complexity of learning the target function

A Bayesian belief network describes the probability distribution governing a set of variables by specifying a set of conditional independence assumptions along with a set of conditional probabilities
Bayesian belief networks allow stating conditional independence assumptions that apply to subsets of the variables

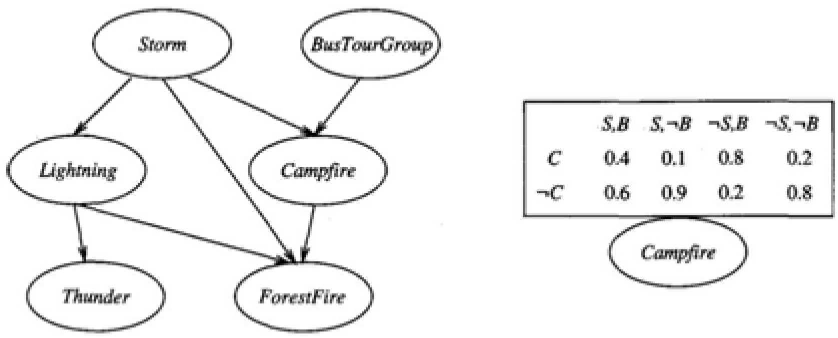
Notation

- Consider an arbitrary set of random variables Y1 . . . Yn , where each variable Yi can take on the set of possible values V(Yi).
- The joint space of the set of variables Y to be the cross product V(Y1) x V(Y2) x. . .V(Yn).
- In other words, each item in the joint space corresponds to one of the possible assignments of values to the tuple of variables (Y1 . . . Yn). The probability distribution over this joint' space is called the joint probability distribution.
- The joint probability distribution specifies the probability for each of the possible variable bindings for the tuple (Y1 . . . Yn).
- A Bayesian belief network describes the joint probability distribution for a set of variables.

Representation

10 CO4 L2

A Bayesian belief network represents the joint probability distribution for a set of variables.
Bayesian networks (BN) are represented by directed acyclic graphs.



The Bayesian network in above figure represents the joint probability distribution over the boolean variables *Storm, Lightning, Thunder, ForestFire, Campfire,* and *BusTourGroup*

A Bayesian network (BN) represents the joint probability distribution by specifying a set of *conditional independence assumptions*

- BN represented by a directed acyclic graph, together with sets of local conditional probabilities

- Each variable in the joint space is represented by a node in the Bayesian network
- The network arcs represent the assertion that the variable is conditionally independent of its non-descendants in the network given its immediate predecessors in the network.

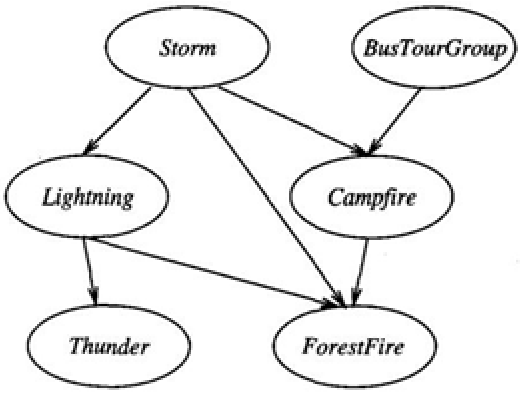
A ***conditional probability table*** **(CPT)** is given for each variable, describing the probability distribution for that variable given the values of its immediate predecessors. The joint probability for any desired assignment of values (y1, . . . , yn) to the tuple of network variables (Y1 . . . Ym) can be computed by the formula

$$P(y_1, \ldots, y_n) = \prod_{i=1}^{n} P(y_i | Parents(Y_i))$$

Where, Parents(Yi) denotes the set of immediate predecessors of Yi in the network.

Example:

Consider the node ***Campfire***. The network nodes and arcs represent the assertion that ***Campfire*** is conditionally independent of its non-descendants ***Lightning*** and ***Thunder***, given its immediate parents Storm and ***BusTourGroup.***



	S,B	$S,\neg B$	$\neg S,B$	$\neg S,\neg B$
C	0.4	0.1	0.8	0.2
$\neg C$	0.6	0.9	0.2	0.8

This means that once we know the value of the variables ***Storm*** and ***BusTourGroup***, the variables ***Lightning*** and ***Thunder*** provide no additional information about ***Campfire*** The conditional probability table associated with the variable ***Campfire.*** The assertion is

P(Campfire = True | Storm = True, BusTourGroup = True) = 0.4

Inference

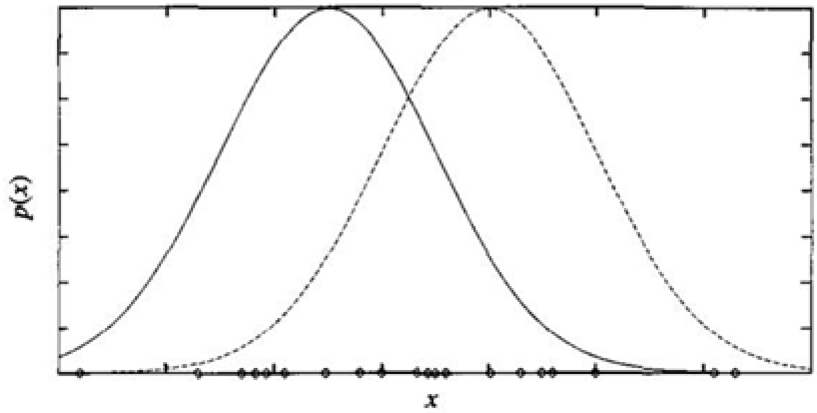
- Use a Bayesian network to infer the value of some target variable (e.g., ForestFire) given the observed values of the other variables.
- Inference can be straightforward if values for all of the other variables in the network are known exactly.
- A Bayesian network can be used to compute the probability distribution for any subset of network variables given the values or distributions for any subset of the remaining variables.
- An arbitrary Bayesian network is known to be NP-hard.

4) **THE EM ALGORITHM** — 10 — CO4 — L2

The EM algorithm can be used even for variables whose value is never directly observed, provided the general form of the probability distribution governing these variables is known.

Estimating Means of k Gaussians

Consider a problem in which the data D is a set of instances generated by a probability distribution that is a mixture of k distinct Normal distributions.



- This problem setting is illustrated in Figure for the case where k = 2 and where the instances are the points shown along the x axis.
- Each instance is generated using a two-step process.
  - First, one of the k Normal distributions is selected at random.
  - Second, a single random instance xi is generated according to this selected distribution.
- This process is repeated to generate a set of data points as shown in the figure.
- To simplify, consider the special case
  - The selection of the single Normal distribution at each step is based on choosing each with uniform probability
  - Each of the k Normal distributions has the same variance σ^2, known value.
- The learning task is to output a hypothesis h = (μ1 , . . . ,μk) that describes the means of each of the k distributions.
- We would like to find a maximum likelihood hypothesis for these means; that is, a hypothesis ***h*** that maximizes p(D |h).

$$\mu_{ML} = \underset{\mu}{\operatorname{argmin}} \sum_{i=1}^{m} (x_i - \mu)^2 \qquad (1)$$

In this case, the sum of squared errors is minimized by the sample mean.

$$\mu_{ML} = \frac{1}{m} \sum_{i=1}^{m} x_i \qquad (2)$$

- Our problem here, however, involves a mixture of k different Normal distributions, and we cannot observe which instances were generated by which distribution.
- Consider full description of each instance as the triple (xi, zi1, zi2),
  - where xi is the observed value of the ith instance and
  - where zi1 and zi2 indicate which of the two Normal distributions was used to generate the value xi
- In particular, zij has the value 1 if xi was created by the j^{th} Normal distribution and 0 otherwise.
- Here xi is the observed variable in the description of the instance, and zil and zi2 are hidden variables.
- If the values of zil and zi2 were observed, we could use following Equation to solve for the means p1 and p2
- Because they are not, we will instead use the EM algorithm.

EM algorithm

	Step 1: Calculate the expected value $E[z_{ij}]$ of each hidden variable z_{ij}, assuming the current hypothesis $h = \langle \mu_1, \mu_2 \rangle$ holds. **Step 2:** Calculate a new maximum likelihood hypothesis $h' = \langle \mu'_1, \mu'_2 \rangle$, assuming the value taken on by each hidden variable z_{ij} is its expected value $E[z_{ij}]$ calculated in Step 1. Then replace the hypothesis $h = \langle \mu_1, \mu_2 \rangle$ by the new hypothesis $h' = \langle \mu'_1, \mu'_2 \rangle$ and iterate. Let us examine how both of these steps can be implemer Step 1 must calculate the expected value of each z_{ij}. This $E[z_{ij}]$ ability that instance x_i was generated by the jth Normal distribut $$E[z_{ij}] = \frac{p(x = x_i \mid \mu = \mu_j)}{\sum_{n=1}^{2} p(x = x_i \mid \mu = \mu_n)} = \frac{e^{-\frac{1}{2\sigma^2}(x_i - \mu_j)^2}}{\sum_{n=1}^{2} e^{-\frac{1}{2\sigma^2}(x_i - \mu_n)^2}}$$ Thus the first step is implemented by substituting the current valu the observed x_i into the above expression. In the second step we use the $E[z_{ij}]$ calculated during Ste new maximum likelihood hypothesis $h' = \langle \mu'_1, \mu'_2 \rangle$. maximum likelihood hypothesis in this case is given by $$\mu_j \leftarrow \frac{\sum_{i=1}^{m} E[z_{ij}] \;\; x_i}{\sum_{i=1}^{m} E[z_{ij}]}$$			
5)	**Sample Error –** The sample error of a hypothesis with respect to some sample S of instances drawn from X is the fraction of S that it misclassifies. ***Definition:*** The sample error (***errors(h)***) of hypothesis h with respect to target function f and data sample S is $$error_S(h) \equiv \frac{1}{n} \sum_{x \in S} \delta(f(x), h(x))$$ Where n is the number of examples in S, and the quantity δ(f(x), h(x)) is 1 if f (x) ≠ h(x), and 0 otherwise. **True Error –** The true error of a hypothesis is the probability that it will misclassify a single randomly drawn instance from the distribution **D**. ***Definition:*** The true error (error$_D$(h)) of hypothesis h with respect to target function f and distribution **D**, is the probability that h will misclassify an instance drawn at random according to **D.** $$error_{\mathcal{D}}(h) \equiv \Pr_{x \in \mathcal{D}} [f(x) \neq h(x)]$$ The Variance captures how far the random variable is expected to vary from its mean value.	10	CO5	L2

Definition: The variance of a random variable Y, Var[Y], is

$$Var[Y] \equiv E[(Y - E[Y])^2]$$

The variance describes the expected squared error in using a single observation of Y to estimate its mean E[Y].
The square root of the variance is called the standard deviation of Y, denoted σy

The Mean (expected value) is the average of the values taken on by repeatedly sampling the random variable

Definition: Consider a random variable Y that takes on the possible values y1, . . . yn. The expected value (Mean) of Y, E[Y], is

$$E[Y] \equiv \sum_{i=1}^{n} y_i \Pr(Y = y_i)$$

Confidence Intervals for Discrete-Valued Hypotheses

Suppose we wish to estimate the true error for some discrete valued hypothesis h, based on its observed sample error over a sample S, where

- The sample S contains n examples drawn independent of one another, and independent of h, according to the probability distribution **D**
- $n \geq 30$
- Hypothesis h commits r errors over these n examples (i.e., errors (h) = r/n).

Under these conditions, statistical theory allows to make the following assertions:
1. Given no other information, the most probable value of $error_D$ (h) is errors(h)
2. With approximately **95% probability**, the true error $error_D$ (h) lies in the interval

$$error_S(h) \pm 1.96\sqrt{\frac{error_S(h)(1 - error_S(h))}{n}}$$

Example:

Suppose the data sample S contains n = 40 examples and that hypothesis h commits r = 12 errors over this data.

- The ***sample error*** is errors(h) = r/n = 12/40 = 0.30
- Given no other information, ***true error*** is $error_D$ (h) = errors(h), i.e., $error_D$ (h) = 0.30

With the 95% confidence interval estimate for $error_D$ (h).

$$error_S(h) \pm 1.96\sqrt{\frac{error_S(h)(1 - error_S(h))}{n}}$$

= 0.30 ± (1.96 * 0.07)   = 0.30 ± 0.14

3. A different constant, ***ZN***, is used to calculate the **N% confidence interval**. The general expression for approximate N% confidence intervals for $error_D$ (h) is

$$error_S(h) \pm z_N\sqrt{\frac{error_S(h)(1 - error_S(h))}{n}}$$

Example:

Suppose the data sample S contains n = 40 examples and that hypothesis h commits r = 12 errors over this data.

- The ***sample error*** is errors(h) = r/n = 12/40 = 0.30

With the 68% confidence interval estimate for error$_D$ (h)

$$error_S(h) \pm 1.00\sqrt{\frac{error_S(h)(1-error_S(h))}{n}}$$

= 0.30 ± (1.00 * 0.07)
= 0.30 ± 0.07

6)

INTRODUCTION

- Instance-based learning methods such as nearest neighbor and locally weighted regression are conceptually straightforward approaches to approximating real-valued or discrete-valued target functions.
- Learning in these algorithms consists of simply storing the presented training data. When a new query instance is encountered, a set of similar related instances is retrieved from memory and used to classify the new query instance.

- Instance-based approaches can construct a different approximation to the target function for each distinct query instance that must be classified.

CASE-BASED REASONING

- Case-based reasoning (CBR) is a learning paradigm based on lazy learning methods and they classify new query instances by analysing similar instances while ignoring instances that are very different from the query.
- In CBR represent instances are not represented as real-valued points, but instead, they use a ***rich symbolic*** representation.

- CBR has been applied to problems such as conceptual design of mechanical devices based on a stored library of previous designs, reasoning about new legal cases based on previous rulings, and solving planning and scheduling problems by reusing and combining portions of previous solutions to similar problems.

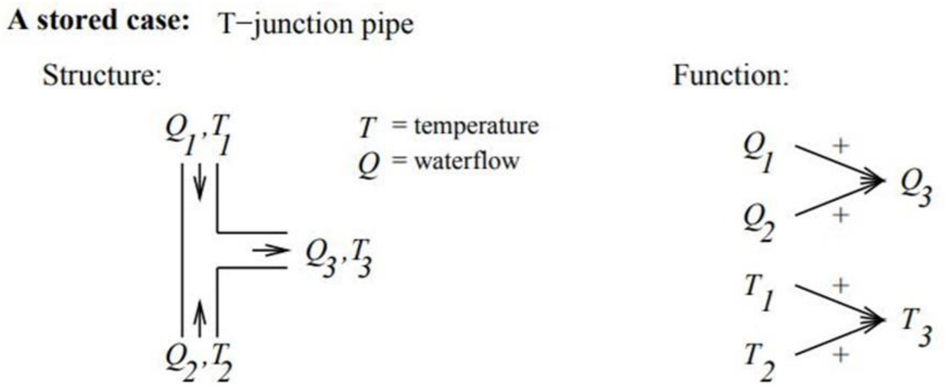
10 CO5 L1

A <u>prototypical example of a case-based reasoning</u>

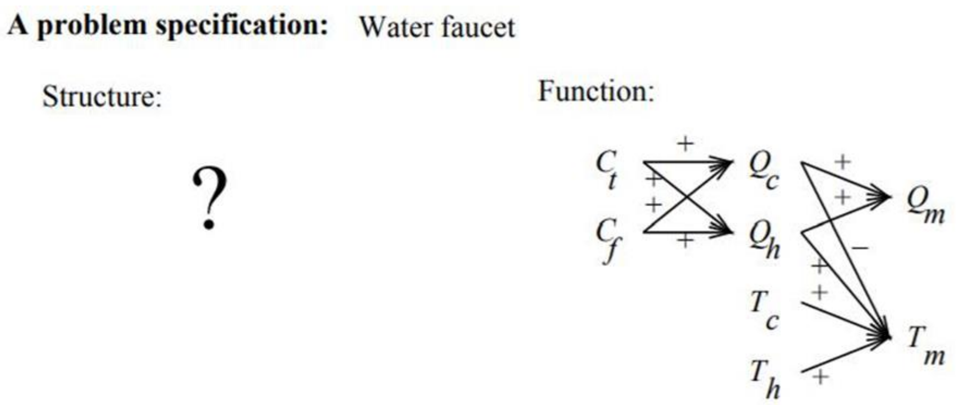
- The CADET system employs case-based reasoning to assist in the conceptual design of simple mechanical devices such as water faucets.
- It uses a library containing approximately 75 previous designs and design fragments to suggest conceptual designs to meet the specifications of new design problems.
- Each instance stored in memory (e.g., a water pipe) is represented by describing both its structure and its qualitative function.

- New design problems are then presented by specifying the desired function and requesting the corresponding structure.

<u>The problem setting is illustrated in below figure</u>



- The function is represented in terms of the qualitative relationships among the water-flow levels and temperatures at its inputs and outputs.
- In the functional description, an arrow with a "+" label indicates that the variable at the arrowhead increases with the variable at its tail. A "-" label indicates that the variable at the head decreases with the variable at the tail.
- Here Qc refers to the flow of cold water into the faucet, Qh to the input flow of hot water, and Qm to the single mixed flow out of the faucet.
- Tc, Th, and Tm refer to the temperatures of the cold water, hot water, and mixed water respectively.
- The variable Ct denotes the control signal for temperature that is input to the faucet, and Cf denotes the control signal for waterflow.
- The controls Ct and Cf are to influence the water flows Qc and Qh, thereby indirectly influencing the faucet output flow Qm and temperature Tm.



- CADET searches its library for stored cases whose functional descriptions match the design problem. If an exact match is found, indicating that some stored case implements exactly the desired function, then this case can be returned as a suggested solution to the design problem. If no exact match occurs, CADET may find cases that match various subgraphs of the desired functional specification.

7)

Hypothesis Testing

- Evaluates 2 mutual exclusive statement on population using sample data.
- Steps:
  1. Make initial Assumptions
  2. Collect Data
  3. Gather Evidence to Reject or accept NULL hypothesis
- What is the probability that

  $error_D(h1) > error_D(h2)$

COMPARING LEARNING ALGORITHMS

- Comparing the performance of 2 learning algorithms LA and LB.
- A reasonable way to define "on average" is to consider the relative performance of these 2 algorithms averaged over all the training sets of size n over Distribution D.

$$\underset{S \subset \mathcal{D}}{E}[error_{\mathcal{D}}(L_A(S)) - error_{\mathcal{D}}(L_B(S))]$$

Where ,
L(S) : Hypothesis output of learning method L when given the sample S of training data .
Here S ⊂ D : The expected value is taken over samples S drawn according to the underlying instance distribution D.

10 | CO5 | L3

Comparing learning algorithms L_A and L_B

1. Partition data D_0 into k disjoint test sets $T_1, T_2, \ldots, T_k$ of equal size, where this size is at least 30.
2. For i from 1 to k, do

   use T_i for the test set, and the remaining data for training set S_i
   - $S_i \leftarrow \{D_0 - T_i\}$
   - $h_A \leftarrow L_A(S_i)$
   - $h_B \leftarrow L_B(S_i)$
   - $\delta_i \leftarrow error_{T_i}(h_A) - error_{T_i}(h_B)$
3. Return the value $\bar{\delta}$, where

$$\bar{\delta} \equiv \frac{1}{k} \sum_{i=1}^{k} \delta_i$$

Paired t Tests

Following is the estimation procedure:

- We are given the observed values of a set of independent, identically distributed random variables Y1, Y2, ….Yk.
- We wish to estimate the mean of the probability distribution governing these Yi.
- The estimator will use sample mean which is given by

$$\bar{Y} \equiv \frac{1}{k} \sum_{i=1}^{k} Y_i$$

- We can request a new training examples drawn according to the underlying instance distribution. Modify the steps on each iteration through the loop it generates a new random training set Si and a new random test set Ti by drawing from this underlying instance distribution instead of drawing from the fixed sample D0.

Confidence interval =

$$\mu = \bar{Y} \pm t_{N,k-1}\, s_{\bar{Y}}$$

where $s_{\bar{Y}}$ is the estimated standard deviation of the sample mean

$$s_{\bar{Y}} = \sqrt{\frac{1}{k(k-1)} \sum_{i=1}^{k} (Y_i - \bar{Y})^2}$$

10)

The *Q* Function

The value of Evaluation function ***Q(s, a)*** is the reward received immediately upon executing action a from state ***s***, plus the value (discounted by ***γ***) of following the optimal policy thereafter

$$Q(s,a) \equiv r(s,a) + \gamma V^*(\delta(s,a)) \qquad \textbf{equ (4)}$$

Rewrite Equation (3) in terms of ***Q(s, a)*** as

$$\pi^*(s) = \underset{a}{\operatorname{argmax}}\, Q(s,a) \qquad \textbf{equ (5)}$$

Equation (5) makes clear, it need only consider each available action ***a*** in its current state ***s*** and choose the action that maximizes ***Q(s, a)***.

An Algorithm for Learning *Q*

- Learning the Q function corresponds to learning the **optimal policy**.

The key problem is finding a reliable way to estimate training values for ***Q***, given only a sequence of immediate rewards ***r*** spread out over time. This can be accomplished through ***iterative approximation***

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

Rewriting Equation

10 CO5 L3

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

Q learning algorithm

Q learning algorithm

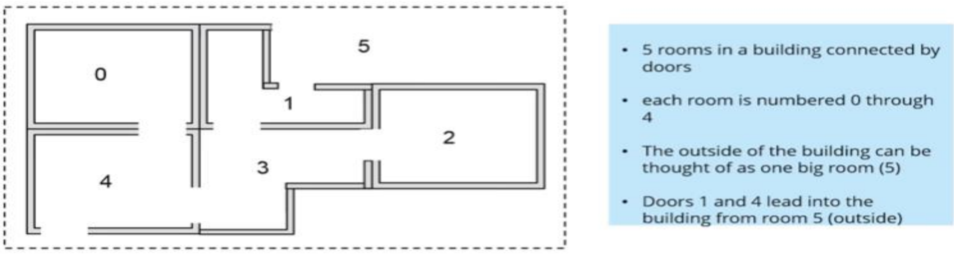
For each s, a initialize the table entry $\hat{Q}(s,a)$ to zero.
Observe the current state s
Do forever:

- Select an action a and execute it
- Receive immediate reward r
- Observe the new state s'
- Update the table entry for $\hat{Q}(s,a)$ as follows:

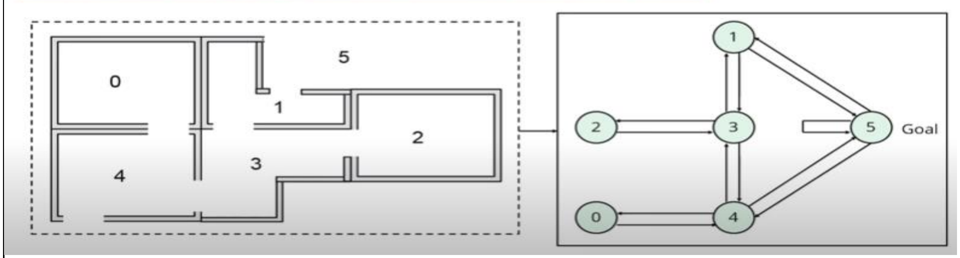
$$\hat{Q}(s,a) \leftarrow r + \gamma \max_{a'} \hat{Q}(s', a')$$

- $s \leftarrow s'$

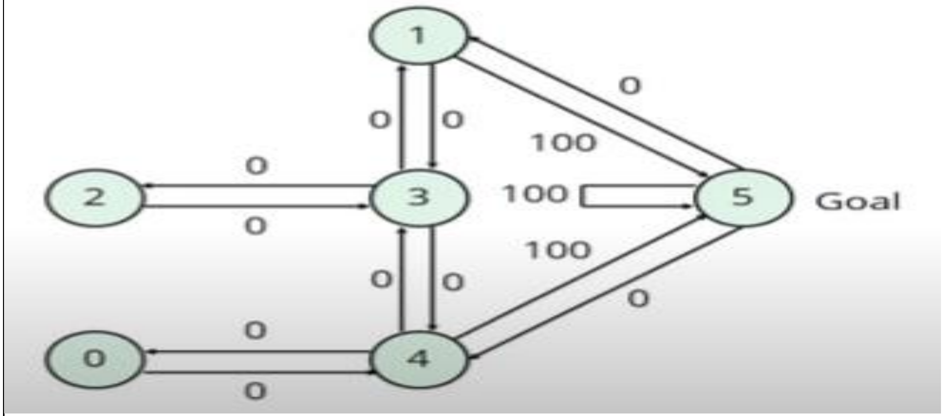
Place an agent in any one of the rooms (0,1,2,3,4) and the goal is to reach outside the building (room 5)



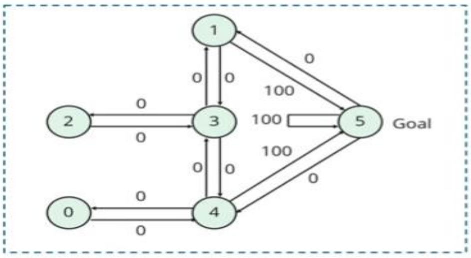
Let's represent the rooms on a graph, each room as a node, and each door as a link



Next Step is to associate reward value to each door.



We can put the state diagram and the instant reward values into a reward table, matrix R.



$R =$

State \ Action	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	0	-1	-1	0	-1	100
5	-1	0	-1	-1	0	100

The -1's in the table represent null values

Next Step is to create Q matrix.

Add another matrix Q, representing the memory of what the agent has learned through experience.

- The rows of matrix Q represent the current state of the agent
- columns represent the possible actions leading to the next state
- Formula to calculate the Q matrix:

*Q(state, action) = R(state, action) + Gamma * Max [Q(next state, all actions)]*

Note

The Gamma parameter has a range of 0 to 1 (0 <= Gamma > 1).

- If Gamma is closer to zero, the agent will tend to consider only immediate rewards.
- If Gamma is closer to one, the agent will consider future rewards with greater weight

<u>Q-Learning Example:</u>

First step is to set the value of the learning parameter Gamma = 0.8, and the initial state as Room 1.

Next, initialize matrix Q as a zero matrix:

- From room 1 you can either go to room 3 or 5, let's select room 5.
- From room 5, calculate maximum Q value for this next state based on all possible actions:

*Q(state, action) = R(state, action) + Gamma * Max[Q(next state, all actions)]*

Q(1,5) = R(1,5) + 0.8 * Max[Q(5,1), Q(5,4), Q(5,5)] = 100 + 0.8 * 0 = 100

$Q =$

	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	0	0	0	0
2	0	0	0	0	0	0
3	0	0	0	0	0	0
4	0	0	0	0	0	0
5	0	0	0	0	0	0

$R =$

State \ Action	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	0	-1	-1	0	-1	100
5	-1	0	-1	-1	0	100