USN | | | | | | | | | | |

CELEBRATING 25 YEARS
CMRIT
CMR INSTITUTE OF TECHNOLOGY, BENGALURU.
ACCREDITED WITH A+ GRADE BY NAAC

Internal Assessment Test 1 – Sept 2020

Sub:	Machine Learning					Sub Code:	17CS73	Branch:	ISE
Date:	15/09/20	Duration:	90 min's	Max Marks:	50	Sem / Sec:	7 / A,B		OBE

	Part-A : Multiple Choice Questions (15 Marks)	MARKS	CO	RBT
1	What is the output of generalizer, in the final design of a learning system ? hypothesis New problem Solution trace Training example Ans: **hypothesis**	1	CO1	L1
2	- Domain: edy, com, org -Platform: Mac, PC, Unix -Browser: Mozilla, Netscape 5, Netscape Communicator, Microsoft IE -Day: Monday - Sunday -Screen: VGA, XVGA -Continent: America, Europe, Africa, Asia, Australia How many syntactically distinct hypotheses and semantically distinct hypothesis are there in the hypothesis space H? Ans: Distinct hypothesis : 3*3*4*7*2*5 =2520 **Syntactically distinct hypothesis = 5*5*6*9*4*7 =37800** **Semantically Distinct Hypothesis = 1+(4*4*5*8*3*6)= 11521**	2	CO1	L2
3	A hypothesis is said to be consistent with training example of the form (x,c(x)),when__ when h(x)=c(x) when h(x)≠c(x) when h(x)>c(x) when h(x)<c(x) Ans: **when h(x)=c(x)**	1	CO1	L1
4	How many distinct animals are there for the given data? Attributes and values of some animals are: Tail(yes, no) Size(small, medium, large) Skin(smooth, furry, slimy) Legs(none, two, four) 24 81 54 11 **Ans: 54**	2	CO1	L2
5	A decision tree is built in _____________ fashion Top-down bottom-up both of the above non- linear **Ans: Top-down**	1		

6	Consider the following sequence of positive and negative training examples describing the concept "pairs of people who live in the same house." Each training example describes an *ordered* pair of people, with each person described by their sex (Male,Female), hair color (black, brown, or blonde), height (tall, medium, or short), and nationality (US, French, German, Irish, Indian, Japanese, or Portuguese). How many distinct hypotheses from the given hypothesis space are consistent with the following training examples? Ans : **2*3*3*7 =126** **Syntactically distinct = 4*5*5*9 = 900** **Semantically distinct = 1+(3*4*4*8) = 385**	2	CO1	L2
7	Consider the following problems. a) Predicting whether stock price of a company will increase tomorrow b) Predicting the gender of a person by his/her handwriting style c) Predicting number of copies a music album will be sold next month Mention the type it belongs to Ans : **Classification,Classification,Regression**	3	CO1	L2
8	There are 2 instances x1,x2 and 3 hypotheses h1,h2.h3 given below. Instances are: x1=<sunny, warm, high, strong, cool, same> x2=<sunny, warm, high, light ,warm, same> hypotheses are : h1=<sunny,?,?,strong,?,?> h2=<sunny,?,?,?,?,?> h3=<sunny,?,?,?,cool,?> Which statement is true for the above data? h3 is more general than h1 h2 more general than h1 and h3 h2 is less general than h1 and h3 h1 is more general than h2 Ans: **h2 more general than h1 and h3**	2	CO1	L2
9	A Decision tree can be used to build models for: Classification problems Regression problems Clustering problem Both Classification and Regression problem Ans : **Both Classification and Regression problem**	1	C O1	**L1**
	Part-B: Short Answer Questions (5 Marks)			
10	Consider Tic-Tac-Toe game and represent the learned function V as a linear combination of board features of your choice. **Ans: Identifying the fetaures mentioned below – 2 Marks** The six features that I extracted from every board state were (a row is 3 subsequent squares… the rows, columns, and diagonals): • x1 = # of instances where there are 2 x’s in a row with an open subsequent square. • x2 = # of instances where there are 2 o’s in a row with an open subsequent square. • x3 = # of instances where there is an x in a completely open row.	2	CO1	L3

	• x4 = # of instances where there is an o in a completely open row. • x5 = # of instances of 3 x's in a row (value of 1 signifies end game) • x6 = # of instances of 3 o's in a row (value of 1 signifies end game) • Vtrain(boardstate) = 100 if end of game and you won. • Vtrain(boardstate) = -100 if end of game and you lost. • Vtrain(boardstate) = 0 if end of game and a draw. • Vtrain(boardstate) = Vestimate(successor(boardstate)) in not the end of the game			
11	Describe the following problems with respect to Tasks, Performance and Experience: i) A checkers learning problem ii) A handwriting recognition learning problem iii) A robot driving learning problem Scheme: Describing the tasks, performance and E for all 3 problems – 3Marks Solution: i)A checkers learning problem: • Task T: playing checkers • Performance measure P: percent of games won against opponents • Training experience E: playing practice games against itself ii)A handwriting recognition learning problem: • Task T: recognizing and classifying handwritten words within images • Performance measure P: percent of words correctly classified • Training experience E: a database of handwritten words with given classifications iii)A robot driving learning problem: • Task T: driving on public four-lane highways using vision sensors • Performance measure P: average distance travelled before an error (as judged by human overseer) • Training experience E: a sequence of images and steering commands recorded while observing a human driver	3		
	Part-C : Descriptive Questions (30 Marks)			
12	Explain the steps in designing learning systems in detail. **Scheme:** Explaining all the 5steps with examples -5*2 = 10 marks 1. Choosing the Training Experience 2. Choosing the Target Function 3. Choosing a Representation for the Target Function 4. Choosing a Function Approximation Algorithm 1. Estimating training values 2. Adjusting the weights 5. The Final Design	[10]	CO1	L2

13

Consider the given below training example which finds malignant tumors from MRI scans.

Example	Shape	Size	Color	Surface	Thickness	Target concept
1	Circular	Large	Light	Smooth	Thick	Malignant
2	Circular	Large	Light	Irregular	Thick	Malignant
3	Oval	Large	Dark	Smooth	Thin	Benign
4	Oval	Large	Light	Irregular	Thick	Malignant
5	Circular	Small	Light	Smooth	Thick	Benign

Show the specific and general boundaries of the version space after applying candidate elimination algorithm. (Note: Malignant is +ve, Benign is —ye).

Scheme:

- Initializing specific hypothesis and generic hypothesis – 2 Marks
- For each positive and negative instances: Finding the maximally specific and generic hypothesis -8 Marks

[10] CO1 L3

14

Find the maximally general hypothesis and maximally specific hypothesis for the training examples given in the table using candidate elimination algorithm.

Size	Color	Shape	Class/Label
Big	Red	Circle	No
Small	Red	Triangle	No
Small	Red	Circle	Yes
Big	Blue	Circle	No
Small	Blue	Circle	Yes

Scheme:

- Initializing specific hypothesis and generic hypothesis – 2 Marks
- For each positive and negative instances: Finding the maximally specific and generic hypothesis -8 Marks

[10] CO1 L3

15

Consider the concept "Economy Car" with the following features {size,trunk,fuel economy,no of passengers,type}

Size	Trunk	Fuel economy	No of Passengers	Type	Target Value
Small	Available	High	4	Economy	Y
Big	Available	Low	2	Sports	N
Small	Available	High	4	Economy	Y
Small	Not Available	Low	2	Sports	N
Medium	Available	High	4	Economy	Y

Compare the above example using Find-S and candidate elimination algorithms and list the observations.

Scheme:

- Initializing specific hypothesis and generic hypothesis – 2 Marks
- For each positive and negative instances: Finding the maximally specific and generic hypothesis -7 Marks
- Comparing both the algorithms and listing the observations – 1 Mark

[10] CO1 L4

<table>
<tr><td>16)a)</td><td>Explain machine learning with an example

Scheme:
Explanation of Machine Learning – 2 Marks
Examples – 2 Marks</td><td>[04]</td><td>CO1</td><td>L2</td></tr>
<tr><td>b)</td><td><table>
<tr><th>Restaurant</th><th>Meal</th><th>Day</th><th>Cost</th><th>Target Value</th></tr>
<tr><td>Sam’s</td><td>Breakfast</td><td>Friday</td><td>Cheap</td><td>Yes</td></tr>
<tr><td>Hilton</td><td>Lunch</td><td>Friday</td><td>Expensive</td><td>No</td></tr>
<tr><td>Sam’s</td><td>Lunch</td><td>Saturday</td><td>Cheap</td><td>Yes</td></tr>
<tr><td>Dannie</td><td>Breakfast</td><td>Sunday</td><td>Cheap</td><td>No</td></tr>
<tr><td>Sam’s</td><td>Breakfast</td><td>Sunday</td><td>Expensive</td><td>No</td></tr>
</table>
Explain the process of finding maximally specific hypothesis for the below dataset using appropriate algorithm

Scheme:
Choosing the suitable algorithm – 1 Mark
Finding the maximally specific hypothesis – 5 Marks</td><td>[06]</td><td></td><td>L3</td></tr>
</table>

Q12)

Solution:

1. Choosing the Training Experience
2. Choosing the Target Function
3. Choosing a Representation for the Target Function
4. Choosing a Function Approximation Algorithm
    1. Estimating training values
    2. Adjusting the weights
5. The Final Design

1. Choosing the Training Experience:

- The type of training experience available can have a significant impact on success or failure of the learner.

There are three attributes which impact on success or failure of the learner

1. One key attribute is whether the training experience provides direct or indirect feedback regarding the choices made by the performance system.
    - For ex in learning to play checkers the system might learn from direct training examples consisting of individual checkers board states and correct move for each.
    - Indirect information consisting of the move sequences and final outcomes of various games played. Here the learner faces an additional problem of credit assignment or determining the degree to which each move in the sequence deserves credit or blame for the final outcome.

    Hence, learning from direct training feedback is typically easier than learning from indirect feedback.

2. A second important attribute of the training experience is the **degree to which the learner controls the sequence of training examples**.
    - For example, the learner might rely on the teacher to select informative board states and to provide the correct move for each.
    - Alternatively, the learner might itself propose board states that it finds particularly confusing and ask the teacher for the correct move. Or the learner may have complete control over both the board states and (indirect) training classifications, as it does when it learns by playing against itself with no teacher present.

3. A third important attribute of the training experience is how well it represents the distribution of examples over which the final system performance P must be measured.

    For ex in checkers game: the performance metric P is the percent of games the system wins in the world tournament.

2. Choosing the Target Function

The next design choice is to determine exactly what type of knowledge will be learned and how this will be used by the performance program. Consider checkers-playing program. The program needs only to learn how to choose the best move from among some large search space. Here we discuss two such methods.

Method-1: Let us use the function ***ChooseMove: B → M***
Which indicate that this function accepts any board from the set of legal board states B as input and produces as output some move from the set of legal moves M.

ChooseMove is a key design choice for the target function in checkers example, this function will turn out to be very difficult to learn given the kind of indirect training experience available to our system.

Method-2: An alternative target function and one that will turn out to be easier to learn in this setting is an evaluation function that assigns a numerical score to any given board state.
Let us call this target function V and again use the notation ***V: B*** →□
Which denote that V maps any legal board state from the set B to some real value in □. If the system can successfully learn such a target function V, then it can easily use it to select the best move from any current board position.

Let us define the target value V(b) for an arbitrary board state b in B, as follows:

- if b is a final board state that is won, then V(b) = 100
- if b is a final board state that is lost, then V(b) = -100
- if b is a final board state that is drawn, then V(b) = 0
- if b is a not a final state in the game, then V(b) = V(b'), where b' is the best final board state that can be achieved starting from b and playing optimally until the end of the game.

3 Choosing a Representation for the Target Function

- Let’s choose a simple representation for any given representation, for any given state the function c is calculated as a linear combination of the following board features.
    - xl: the number of black pieces on the board
    - x2: the number of red pieces on the board
    - x3: the number of black kings on the board
    - x4: the number of red kings on the board

- x5: the number of black pieces threatened by red (i.e., which can be captured on red's next turn)
- x6: the number of red pieces threatened by black

Thus, learning program will represent as a linear function of the form

$$\hat{V}(b) = w_0 + w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + w_5x_5 + w_6x_6$$

Where w_0 through w_6 are the weights to be chosen by the Learning Algorithm.

4 Choosing a Function Approximation Algorithm

- In order to learn the target function *f* we require a set of training examples, each describing a specific board state b and the training value $V_{train}(b)$ for b.
- Each training example is an ordered pair of the form (b, $V_{train}(b)$).
- For instance, the following training example describes a board state b in which black has won the game (note $x_2 = 0$ indicates that red has no remaining pieces) and for which the target function value $V_{train}(b)$ is therefore +100.

  ((x_1=3, x_2=0, x_3=1, x_4=0, x_5=0, x_6=0), +100)

Function approximation Procedure:

1. Estimating training values
2. Adjusting the weights

1. Estimating training values:

- The only training information available to our learner is whether the game was eventually **won or lost**.
- The approach is to assign the training value of $V_{train}(b)$ for any intermediate board state *b* to be $\hat{V}$ *(Successor(b))*

  Rule for estimating training values: $V_{train}(b) \leftarrow \hat{V}(Successor(b))$

2. Adjusting the weights:

- One common approach is to define the best hypothesis, or set of weights, as that which minimizes the square error *E between* the training values and the values predicted by the hypothesis *V.*

$$E \equiv \sum_{\langle b, V_{train}(b)\rangle \in \ training\ \ examples} (V_{train}(b) - \hat{V}(b))^2$$

We require an algorithm that will incrementally refine the weights as new training examples become available and that will be robust to errors in these estimated training values.

One such algorithm is called the least mean squares (LMS) training rule.

LMS Weight update rule :

For each training example $\langle b, V_{train}(b)\rangle$

- Use the current weights to calculate $\hat{V}(b)$
- For each weight w_i, update it as

$$w_i \leftarrow w_i + \eta \ (V_{train}(b) - \hat{V}(b)) \ x_i$$

Here η is a small constant (e.g., 0.1) that moderates the size of the weight update.

Working of weight update rule

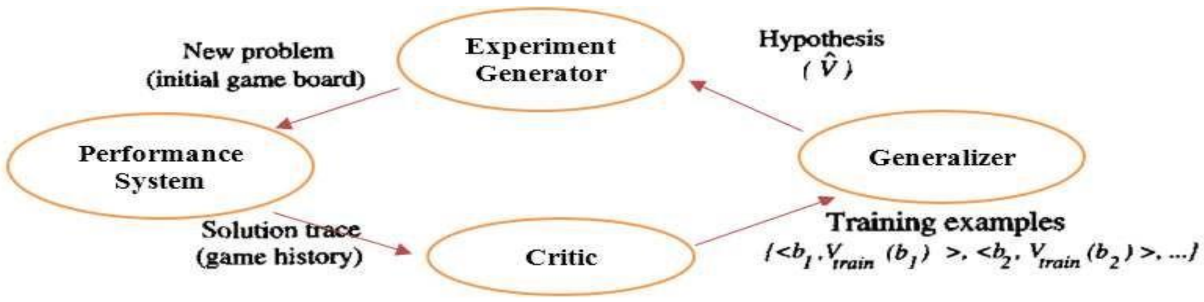
- When the error (Vtrain(b)- V̂(b)) is zero, no weights are changed.
- When (Vtrain(b) - V̂(b)) is positive (i.e., when V̂(b) is too low), then each weight is increased in proportion to the value of its corresponding feature. This will raise the value of V̂(b), reducing the error.
- If the value of some feature x_i is zero, then its weight is not altered regardless of the error, so that the only weights updated are those whose features actually occur on the training example board.

5 The final design:

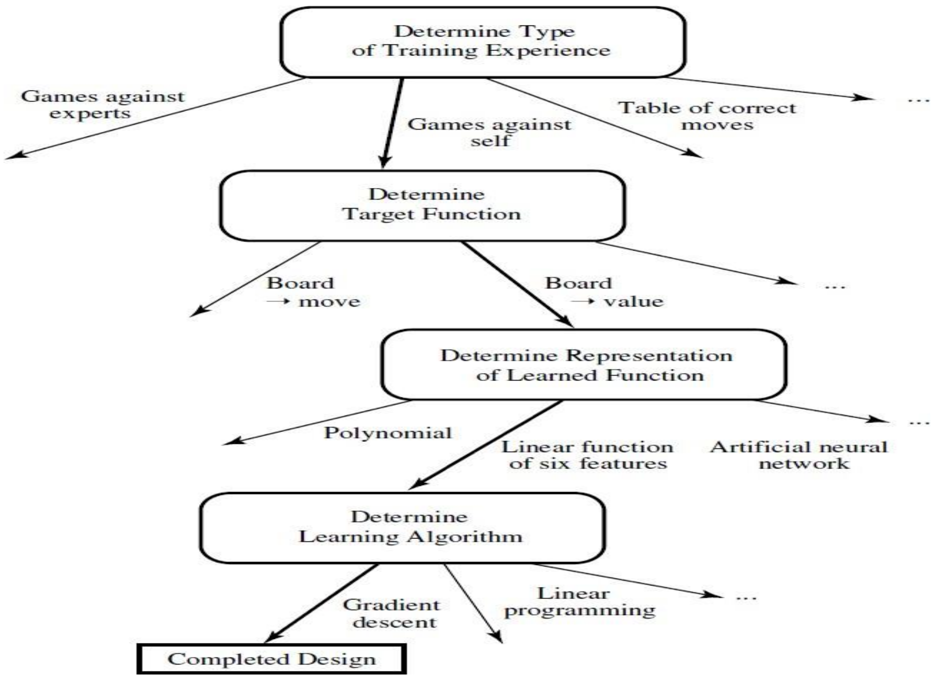
The final design of our checkers learning system can be naturally described by four distinct program modules that represent the central components in many learning systems.

a) The **Performance System** is the module that must solve the given performance task, in this case playing checkers, by using the learned target function(s). It takes an instance of a new problem (new game) as input and produces a trace of its solution (game history) as output.
b) The **Critic** takes as input - history or trace of the game and produces as output - a set of training examples of the target function.
c) The **Generalizer** takes as input the training examples and produces an output hypothesis that is its estimate of the target function. It generalizes from the specific training examples, hypothesizing a general function that covers these examples and other cases beyond the training examples.
d) The **Experiment Generator** takes as input the current hypothesis (currently learned function) and outputs a new problem (i.e., initial board state) for the Performance System to explore. Its role is to pick new practice problems that will maximize the learning rate of the overall system.

These four modules are summarized as follows



The sequence of design choices made for the checkers program is summarized in figure given below.



Q14) Solution:

Step1: S0= {'0', '0', '0'}
G0 = {'?', '?', '?' }

Step2: Training Instance d: ('big', 'red', 'circle', 'N') -ve instance
S after removing consistent hypothesis with d
S1={('0','0', '0')}

Consider g: ('?', '?', '?')

g after min specialization:
Replace each ? with opposite pair in –ve instance. (First ? can be either small or big but we already have big in our –ve example so replace with **Small**, next ? is either red or blue and we have red already so replace with **blue** and third ? can be either triangle or circle and we have circle already so replace with **triangle**)

S1: {('0', '0', '0')}
G1: {('?', '?', 'triangle'), ('small', '?', '?'), ('?', 'blue', '?')}

Next Instance d: {small, red, triangle} –ve instance

S2={0,0,0}
G: {('?', '?', 'triangle'), ('small', '?', '?'), ('?', 'blue', '?')}

First two pairs are matching with my –ve instance (i.e. triangle and small) which should be opposite so try to make each ? in the first two pairs with specific ones. {?,blue,?} is opposite to –ve instance d so keep it as it is.

Replace each pair ? with the opposite pair in the negative instance. (In the first pair ? should be replaced by **{big,?,triangle}** bcz small is there in my –ve instance and second ? can be replaced by **{?,blue,triangle}**
Second pair second ? can be replaced by {small,blue,?} and third ? can be replaced by {small,?,circle}

My final G will be as below:
G[2]: {{big,?, triangle} {?,blue, triangle}{small,blue,?} {small,?,circle} ('?', 'blue', '?')

Compare G[2] with d and if it is not consistent then remove that pair.
{big,?,triangle} {small, ?, circle} ,– consistant bcz it is –ve and my instance also negative so consider this one.
{?,blue,triangle} and {small,blue,?} - These two are specific to {?,blue,?} so ignore it.
{?,blue,?} – This is –ve as opposite to red is blue but my instance also –ve so consistant.

My final generic hypothesis after removing less consistant ones are :
G : {{big,?,triangle}{small,?,circle}{?,blue,?}

Next Instance d: {small, red, circle} +ve instance

G after removing inconsistent hypothesis : {small,?,circle}
bcz {big,?,triangle} and {?,blue,?} are not consistant with d so ignore it.
S[3] = {small,red,circle}
G[3] = {small,?,circle}

Next Instance d: {big, blue, circle} -ve instance
S after removing inconsistent hypothesis : {small,red,circle}
G: {small,?,circle} which is negative and my d also –ve so consistant so keep as it is.
G[4] : {small,?,circle}

Next Instance d: {small, blue, circle} +ve instance
G after removing inconsistent hypothesis : {small,?,circle}
S[5] = {small,?,circle}
G[5] = {small,?,circle}

Q13 Solution:

1) S0=0,0,0,0,0
G0=?,?,?,?,?

2) 1^{st} Instance: circular,large,light,smooth,thick +ve

S1=circular,large,light,smooth,thick
G1=?,?,?,?,?

2^{nd} Instance: circular,large,light,irregular,thick +ve
S2=circular,large,light,?,thick
G2=?,?,?,?,?
3^{rd}instance:oval,large,dark,smooth,thin –ve
S3=circular,large,light,?,thick
G3=(circular,?,?,?,?)(?,?,light,?,?)(?,?,?,?,thick)

4^{th} instance:oval,large,light,irregular,thick +ve
S4=?,large,light,?,thick
G4=(?,?,light,?,?)(?,?,?,?,thick)

5^{th} instance:circular,small,light,smooth,thick -ve
S5=?,large,light,?,thick
G5=(?,large,light,?,?)(?,large,?,?,thick)

Q15 Solution:

Candidate Elimination Algorithm:
S0=0,0,0,0,0
G0=?,?,?,?,?

small,available,high,4,economy +ve
s1=small,avail,high,4,economy
G1=?,?,?,?,?

big,avail,low,2,sports -ve
S2=small,avail,high,4,economy
G2=(small,?,?,?,?)(?,?,high,?,?)(?,?,?,4,?)(?,?,?,?,economy)

small,avail,high,4,economy +ve
S3=small,avail,high,4,economy
G3=(small,?,?,?,?)(?,?,high,?,?)(?,?,?,4,?)(?,?,?,?,economy)

small,not avail,low,2,sports -ve
S4=small,avail,high,4,economy
G4=(?,?,high,?,?)(?,?,?,4,?)(?,?,?,?,economy)
(small,?,?,?,economy)

medium,avail,high,4,economy +ve
S5=?,avail,high,4,economy
G5=(?,?,high,?,?)(?,?,?,4,?)(?,?,?,?,economy)

Find-S algorithm:

h1=smallavailable,high,4,economy +ve

2^{nd} instance: big,avail,low,2,sports -ve
h2=h1

3^{rd} instance : small,avail,high,4,economy +ve
h3= small,avail,high,4,economy

4^{th} instance: small,not avail,low,2,sports -ve
h4= small,avail,high,4,economy

5^{th} instance: medium,avail,high,4,economy +ve h5=?,avail,high,4,economy Maximally specific hypothesis will be same for both Find-S and candidate elimination algorithms .Only generic hypothesis values will change in candidate elimination.

Q16a)Solution

Machine learning is the art of teaching machines to learn.A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.

• Some tasks cannot be defined well, except by examples (e.g., recognizing people).
• Relationships and correlations can be hidden within large amounts of data. Machine Learning/Data Mining may be able to find these relationships.
• Human designers often produce machines that do not work as well as desired in the environments in which they are used.
• The amount of knowledge available about certain tasks might be too large for explicit encoding by humans (e.g., medical diagnostic).
• Environments change over time.
• New knowledge about tasks is constantly being discovered by humans. It may be difficult to continuously re-design systems “by hand”.

Q)16b Solution:

x1 =< Sam’s,breakfast,Friday,cheap > +
Observing the first training example, it is clear that our hypothesis is too specific. In particular, none of the "Ø" constraints in h are satisfied by this example, so each is replaced by the next more general constraint that fits the example

h1 = =< Sam’s,breakfast,Friday,cheap >

x2 = <Hilton,Lunch,Friday,Expensive>, -
Since it is negative no change in h and h2 = h1

x3= <Sam’s,Lunch,Saturday,Cheap> +
Compare each instance of x3 with h2 and replace it with ?

h3= <Sam’s,?,?,cheap>

x4=<Dannie,breakfast,Sunday,Cheap> -ve
Since it is negative no change in h3 and h4 =h3

x5= <Sam’s,breakfast,Sunday,Expensive> -ve
Since it is negative no change in h4 and h5 = <Sam’s,?,?,cheap>
