Assignment 3 a, calculation part for cmc5707, 2024.12.05b

Q1-14 and Q15-18

Description After you complete the assignment, you may get a mark (probably 0), it is a dummy result and not the real result because the true answers have not been input to the systems yet. The real result will be published 2 weeks or more after the assignment deadline.

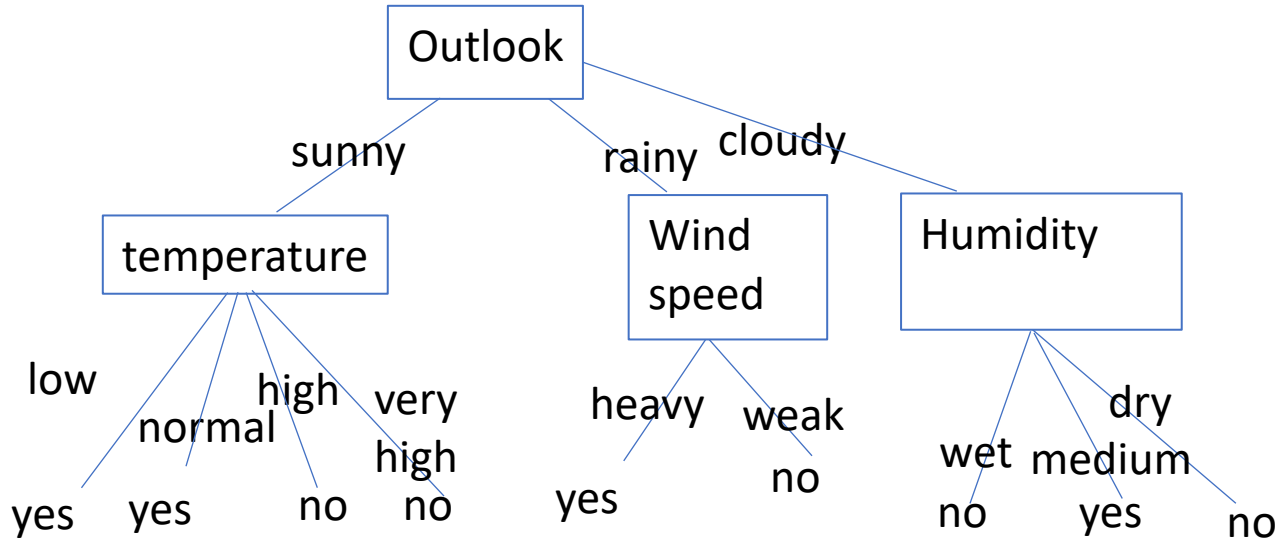
Instructions Hard deadline: late submission is not allowed.
Please click on the “save and submit” button at the end of the page after you completed the assignment.
If your answer is not an integer, give it to the nearest 3 decimal places.
If your answer is an integer, add decimal point has no harm. E.g. 3=3.0=3.00=3.000 etc.

Please complete the assignment before the deadline. Multiple attempts are allowed.

Total 18 questions, 205 marks

A57074.3.1 : Decision tree1

In the diagram, the number of nodes in this tree is N, the number of leaves is E, find N+E.



Answer1:13 (ok4)

+/- 0

N=4,E=9,

N+E=13

//////////////////////

A57074.3.2 : Decision Tree using entropy

As part of the steps for developing a decision tree, we need to find the entropy values of the nodes. Please find the entropy of the parent node of the decision tree using the given table.

Attributes				classes
Feature1	Feature2	Feature3	Feature4	class
A	1	X	L	Decision YES
A	2	Y	H	Decision YES
B	2	X	L	Decision NO
B	1	X	H	Decision NO
C	1	X	L	Decision NO
C	2	Y	L	Decision NO
A	1	Y	H	Decision NO
B	1	Y	H	Decision NO
B	1	X	L	Decision YES
A	2	Y	H	Decision NO

Attributes				classes
Feature1	Feature2	Feature3	Feature4	class
A	1	X	L	Decision YES
A	2	Y	H	Decision YES
B	2	X	L	Decision NO
B	1	X	H	Decision NO
C	1	X	L	Decision NO
C	2	Y	L	Decision NO
A	1	Y	H	Decision NO
B	1	Y	H	Decision NO
B	1	X	L	Decision YES
A	2	Y	H	Decision NO

Answer2: 0.8812 (ok4)
+/- 0.002

Prob(YES)=3/10=0.3
Prob(NO)=7/10=0.7
Entropy(parent)= -0.3*log_2(0.3)- 0.7*log_2(0.7) =0.8812

//
A57074.3.3: Decision Tree using Gini index

As part of the steps for developing a decision tree, we need to find the Gini index of the nodes. Please find the Gini index of the parent node of the decision tree using the given table.

Attributes				classes
Feature1	Feature2	Feature3	Feature4	class
A	1	X	L	Decision 1
A	2	Y	H	Decision 1
B	2	X	L	Decision 2
B	1	X	H	Decision 2
C	1	X	L	Decision 2
C	2	Y	L	Decision 1
A	1	Y	H	Decision 2
B	1	Y	H	Decision 3
B	1	X	L	Decision 2
A	2	Y	H	Decision 1

Attributes				classes
Feature1	Feature2	Feature3	Feature4	class
A	1	X	L	Decision 1
A	2	Y	H	Decision 1
B	2	X	L	Decision 2
B	1	X	H	Decision 2
C	1	X	L	Decision 2
C	2	Y	L	Decision 1
A	1	Y	H	Decision 2
B	1	Y	H	Decision 3
B	1	X	L	Decision 2
A	2	Y	H	Decision 1

Answer3: 0.58 (ok4)
Prob(1)=4/10=0.4
Prob(2)=5/10=0.5
Prob(3)=1/10=0.1

Gini(parent)= 1-0.4*0.4-0.5*0.5-0.1*0.1=0.58

//
A57074.3.4 :Decision tree

The weather and the decision of wearing thick clothes are shown in the given table.
The value of Information gain is A by entropy if Humidity is the root of the tree.
The value of Information gain is B by entropy if Temperature is the root of the tree.
The value of Information gain is C by entropy if Wind is the root of the tree.

Give you answer as A+B+C.

	Weather			Decision:
Samples	Feature: Humidity	Feature: Temperature	Feature: Wind	Wear thick cloth
1	Dry	Cold	High	Yes
2	Dry	Cold	Low	No
3	Medium	Average	High	Yes
4	Medium	Average	Low	No
5	Medium	Hot	High	No
6	Medium	Hot	Low	No
7	Wet	Cold	Low	Yes
8	Wet	Average	Low	No
9	Wet	Hot	High	Yes

Answer4: 0.4011 (ok4)

Table a.

1a. Humidity	Decision=yes	Decision=no	No of instances
Dry	1	1	2
Medium	1	3	4
Wet	2	1	3
		Total samples	9

–

Table b

1b. Temperature	Selected=yes	Selected=no	No of instances
Cold	2	1	3
Average	1	2	3
Hot	1	2	3
		Total samples	9

–

Table c

1c. Wind	Selected=yes	Selected=no	No of instances
High	3	1	4
Low	1	4	5
		Total samples	9

–

Answer:

========== Step 0 ==========

Gini Approach ……

No calculation needed here

……

Information gain by Entropy approach ……

Note log(0) will give -ve infinity or NaN(Not a number) message, use log_2(0)≈log_2(0.00000001) -26.5

total_parent_samples =9, including selected as player: yes=4,no=5.
Entropy_parent = -(4/9)*log2(4/9) -(5/9)*log2(5/9)= 0.99

====================== Step 1a ============================
Step1a
Table a

1a. Humidity	Decision=yes	Decision=no	No of instances
Dry	1	1	2
Medium	1	3	4
Wet	2	1	3
		Total samples	9

……1a_entropy information gain by Entropy approach ……
Recall; Entropy_parent= 0.99
Weighted_entropy_(Humidity=Dry) =(2/9)* (-(1/2)*log2(1/2) -(1/2)*log2(1/2))= 0.222
Weighted_entropy_(Humidity=Medium) =(4/9)* (-(1/4)*log2(1/4) -(3/4)*log2(3/4)= 0.36056
Weighted_entropy_(Humidity=Wet) =(3/9)* (-(1/3)*log2(1/3) -(2/3)*log2(2/3))= 0.3061

1a_Weighted_entropy_(Humidity)= Entropy_parent- Weighted_entropy_(Humidity=Dry)
- Weighted_entropy_(Humidity=Medium)- Weighted_entropy_(Humidity=Wet)
=0.99-0.2222-0.36056-0.3061=0.101
Clear %to verify
temp1=(2/9)* (-(1/2)*log2(1/2) -(1/2)*log2(1/2))
temp2=(4/9)* (-(1/4)*log2(1/4) -(3/4)*log2(3/4))
temp3=(3/9)* (-(2/3)*log2(2/3) -(1/3)*log2(1/3))
A=0.99-temp1-temp2-temp3
= 0.1011

##
====================== Step 1b ============================
Step 1b
Table b

1b. Temperature	Selected=yes	Selected=no	No of instances
Cold	2	1	3
Average	1	2	3
Hot	1	2	3
		Total samples	9

……1b_entropy information gain by Entropy approach ……
Recall: Entropy_parent=0.99
Weighted_entropy_(temperature = Cold) =(3/9) *(-(2/3)*log2(2/3) -(1/3)*log2(1/3))= 0.306
Weighted_entropy_(temperature = average) =(3/9) *(-(1/3)*log2(1/3) -(2/3)*log(2/3))= 0.306
Weighted_entropy_(temperature = hot) =(3/9) *(-(1/3)*log2(1/3) -(2/3)*log2(2/3))= 0.306

1b_information_gain(emperature)= Entropy_parent - Weighted_entropy_(temperature = Cold) –

Weighted_entropy_(temperature = average)- Weighted_entropy_(temperature = hot)=0.99-0.306-0.306- 0.306=0.0717

clear %clear %to verify
%zero=10^-10
temp1=(3/9) *(-(2/3)*log2(2/3) -(1/3)*log2(1/3))
temp2=(3/9) *(-(1/3)*log2(1/3) -(2/3)*log2(2/3))
temp3=(3/9) *(-(1/3)*log2(1/3) -(2/3)*log2(2/3))
B=0.99-temp1-temp2-temp3
%= 0.0717

##
========== Step 1c==========
##

Table c

1c. Wind	Selected=yes	Selected=no	No of instances
High	3	1	4
Low	1	4	5
		Total samples	9

–

######1_entropy information gain by Entropy approach .....
Recall: Entropy_parent= 0.99
Weighted_entropy_(wind=high) =(4/9)* (-(3/4)*log2(3/4) -(1/4)*log2(1/4))= 0.3606
Weighted_entropy_(wind=low) =(5/9)* (-(1/5)*log2(1/5) -(4/5)*log2(4/5))= 0.4011

1c_information_gain(experience)= Entropy_parent-
C=Weighted_entropy_(wind=high) - Weighted_entropy_(wind=low) = 0.99- 0.3606-0.4011
=0.2283

% ----------finally ----------
A=0.1011
B=0.0717
C=0.2283

A+B+C
%= 0.4011

//
A57074.3.5 : Text processing (BOW)

Given the following corpus of sentences

- Sentence s1 : Upon the road I met seven wives
- Sentence s2 : Every wife had seven sacks
- Sentence s3 : Every sack had seven cats

• Sentence s4 : Every cat had seven kits

In this question, the corpus should be preprocessed by the stemming algorithm, i.e. catty and cats are the same word as cat, etc.

BOW (Bag Of Words) cosine similarity measure (cosine_BOW_x) can be used to determine the similarity between sentences.

Write the highest cosine_BOW_x value among two different sentences in the above corpus.

Answer5: 0.8 (ok4)

Range 0.002

Answer:

	1	2	3	4	5	6	7	8	9	10	11	12
	upon	the	road	I	met	seven	wives wife	every	had	Sacks sack	Cats cat	kits kit
S1	1	1	1	1	1	1	1					
S2						1	1	1	1	1		
S3						1		1	1	1	1	
S4						1		1	1		1	1

%Sentence s1 : Upon the road I met seven wives
%Sentence s2 : Every wife had seven sacks
%Sentence s3 : Every sack had seven cats
%Sentence s4 : Every cat had seven kits

```
s1=[1 1 1 1 1 1 1 0 0 0 0 0]'
s2=[0 0 0 0 0 1 1 1 1 1 0 0]'
s3=[0 0 0 0 0 1 0 1 1 1 1 0]'
s4=[0 0 0 0 0 1 0 1 1 0 1 1]'
%If just using BOW for measure similarity
cosine_s1_s2=s1'*s2/(sqrt(s1'*s1)*sqrt(s2'*s2))
cosine_s1_s3=s1'*s3/(sqrt(s1'*s1)*sqrt(s3'*s3))
cosine_s1_s4=s1'*s4/(sqrt(s1'*s1)*sqrt(s4'*s4))
cosine_s2_s3=s2'*s3/(sqrt(s2'*s2)*sqrt(s3'*s3))
cosine_s2_s4=s2'*s4/(sqrt(s2'*s2)*sqrt(s4'*s4))
cosine_s3_s4=s3'*s4/(sqrt(s3'*s3)*sqrt(s4'*s4))
```

```
% cosine_s1_s2 =     0.3381
% cosine_s1_s3 =    0.1690
% cosine_s1_s4 =    0.1690
```

```
% cosine_s2_s3 =    0.8000
% cosine_s2_s4 =    0.6000
% cosine_s3_s4 =    0.8000
```

//
A57074.3.6 : Text processing (TF-IDF)

Given the following corpus of sentences

• Sentence s1 : Upon the road I met seven wives
• Sentence s2 : Every wife had seven sacks
• Sentence s3 : Every sack had seven cats
• Sentence s4 : Every cat had eight kits

TF-IDF (Term Frequency–Inverse Document Frequency) can be used to measure the cosine similarity difference between sentences.

In this question, the corpus should be preprocessed by the stemming algorithm, i.e. catty and cats are the same word as cat, etc.

write the TF-IDF cosine similarity difference between sentence s2 and s3.

Answer6=0.6033 (ok4)

If using TF-IDF is considered

- Sentence s1 : Upon the road I met seven wives
- Sentence s2 : Every wife had seven sacks
- Sentence s3 : Every sack had seven cats
- Sentence s4 : Every cat had eight kits

word	TF				IDF	TF*IDF			
	S1	S2	S3	S4		S1	S2	S3	S4
upon	1/7				Log(4/1)	(1/7)* Log(4/1)= 0.086			
the	1/7				Log(4/1)	(1/7)* Log(4/1) = 0.086			
road	1/7				Log(4/1)	(1/7)* Log(4/1) = 0.086			
I	1/7				Log(4/1)	(1/7)* Log(4/1) = 0.086			
met	1/7				Log(4/1)	(1/7)* Log(4/1) = 0.086			
seven	1/7	1/5	1/5	1/5	Log(4/3)	(1/7)* Log(4/3) = 0.0178	(1/5)* Log(4/3) =0.025	(1/5)* Log(4/3) =0.025	
wives	1/7	1/5			Log(4/2)	(1/7)* Log(4/2)= 0.043	(1/5)* Log(4/2) = 0.06		
Every		1/5	1/5	1/5	Log(4/3)		(1/5)* Log(4/3) = 0.025	(1/5)* Log(4/3) = 0.025	(1/5)* Log(4/3) = 0.025
Had		1/5	1/5	1/5	Log(4/3)		(1/5)* Log(4/3) =0.025	(1/5)* Log(4/3) = 0.025	(1/5)* Log(4/3) = 0.025
Sacks		1/5	1/5		Log(4/2)		(1/5)* Log(4/2) = 0.06	(1/5)* Log(4/2) =0.06	
Cats			1/5	1/5	Log(4/2)			(1/5)* Log(4/2) =0.06	(1/5)* Log(4/2)=0.06
Eight				1/5	Log(4/1)				(1/5)* Log(4/1)=0.12
kits				1/5	Log(4/1)				(1/5)* Log(4/1)= 0.12

```
s1=[0.086,0.086,0.086,0.086,0.086,0.0178,0.043,0,0,0,0,0,0]'
s2=[0,0,0,0,0, 0.025,0.06,0.025,0.025,0.06,0,0,0]'
s3=[0,0,0,0,0, 0.025,0,0.025,0.025,0.06,0.06,0,0]'
s4=[0,0,0,0,0, 0,0,0.025,0.025,0,0.06,0.12,0.12]'
```

```
%matlab
clear
clc
s1=[0.086,0.086,0.086,0.086,0.086,0.0178,0.043,0,0,0,0,0,0]'
s2=[0,0,0,0,0, 0.025,0.06,0.025,0.025,0.06,0,0,0]'
s3=[0,0,0,0,0, 0.025,0,0.025,0.025,0.06,0.06,0,0]'
s4=[0,0,0,0,0, 0,0,0.025,0.025,0,0.06,0.12,0.12]'
size(s1)
size(s2)
size(s3)
size(s4)

cosine_s1_s2=s1'*s2/(sqrt(s1'*s1)*sqrt(s2'*s2))
cosine_s1_s3=s1'*s3/(sqrt(s1'*s1)*sqrt(s3'*s3))
cosine_s1_s4=s1'*s4/(sqrt(s1'*s1)*sqrt(s4'*s4))
cosine_s2_s3=s2'*s3/(sqrt(s2'*s2)*sqrt(s3'*s3))
cosine_s2_s4=s2'*s4/(sqrt(s2'*s2)*sqrt(s4'*s4))
cosine_s3_s4=s3'*s4/(sqrt(s3'*s3)*sqrt(s4'*s4))
'answer='
cosine_s2_s3
%    0.6033
```

//

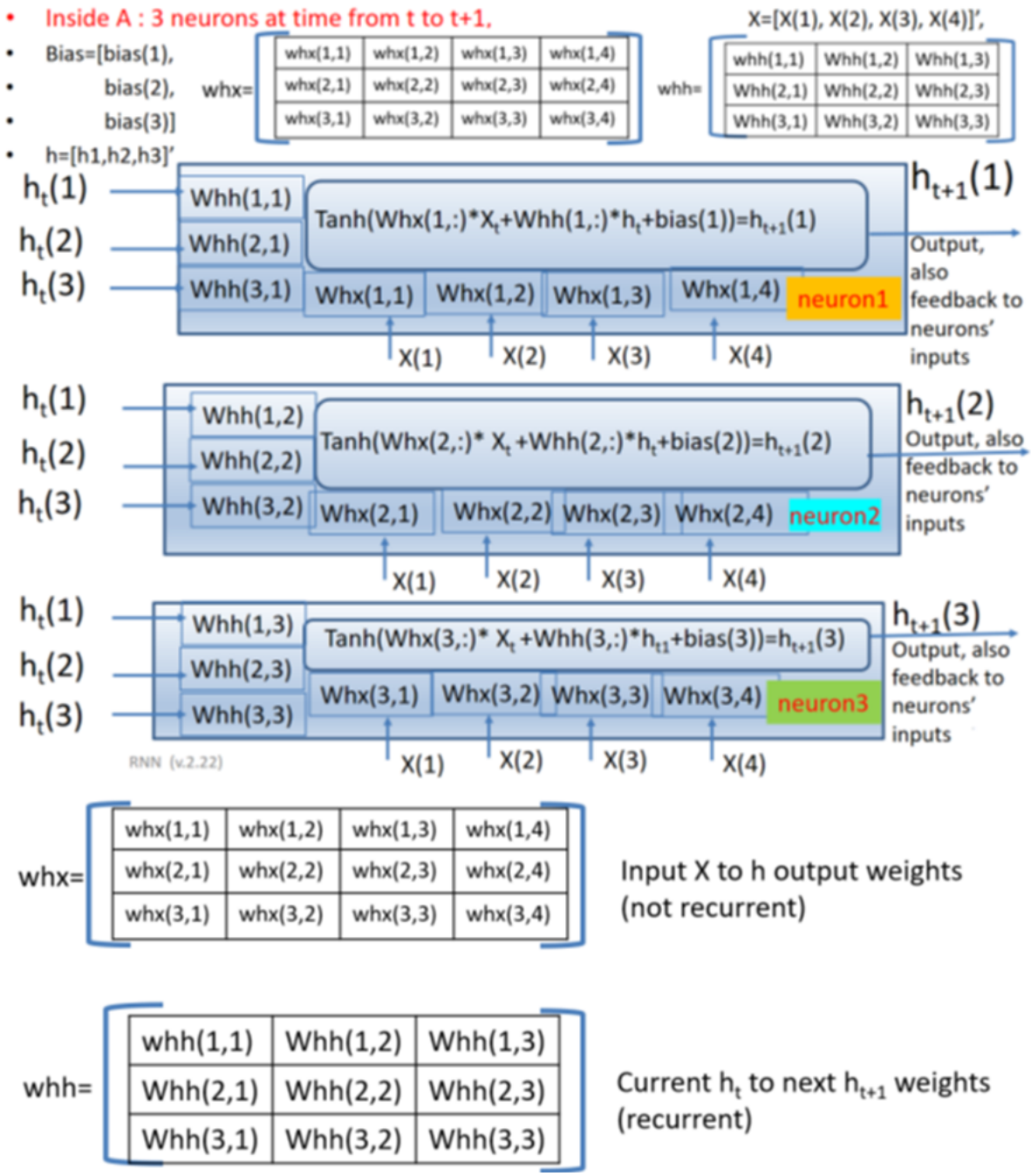
A57074.3.7 : RNN1

Given a 3-neuron recurrent neural network (RNN) as in the diagram attached.
The weights used in the recurrent neural network (RNN) are whx, whh.

```
%assume whx,whh,why are initialized at t=1 as
whx=[0.08 0.64  0.77 0.38
     0.10 0.57 0.29  0.48
     0.23 0.19  0.15 0.29];
whh =[0.21    0.22   0.23
      0.31    0.44   0.26
      0.41    0.54   0.36];
why=[0.47 0.97 0.83
     0.39 0.28 0.65
     0.84 0.29 0.33
     0.51 0.32 0.12];
bias=[0.41, 0.12, 0.63]';%bias initialised
ht(:,1)=[0.3 0.2 0.1]'; %assume ht has these values
initally at t=1
```

The input at time t=1 is [X1,X2,X3,X4]= [1, 0, 0, 0]

Find output h of the third neuron at time t =2, h(3,t=2).



whx=

whx(1,1)	whx(1,2)	whx(1,3)	whx(1,4)
whx(2,1)	whx(2,2)	whx(2,3)	whx(2,4)
whx(3,1)	whx(3,2)	whx(3,3)	whx(3,4)

Input X to h output weights (not recurrent)

whh=

whh(1,1)	Whh(1,2)	Whh(1,3)
Whh(2,1)	Whh(2,2)	Whh(2,3)
Whh(3,1)	Whh(3,2)	Whh(3,3)

Current h_t to next h_{t+1} weights (recurrent)

Answer7: 0.8100 (ok4)

////////////////////////////////////

%MatlabDemo rnn4b.m

%https://stackoverflow.com/questions/50050056/simple-rnn-example-showing-numerics

%https://www.analyticsvidhya.com/blog/2017/12/introduction-to-recurrent-neural-networks/

clear, clc

```
in_S=[1 0 0 0]';
in_C=[0 1 0 0]';
in_R=[0 0 1 0]';
in_T=[0 0 0 1]';
X=[in_S,in_C,in_R, in_T];
%assume whx,whh,why are initialised at t=1as
whx=[0.08 0.64  0.77 0.38
    0.10 0.57 0.29  0.48
    0.23 0.19  0.15 0.29];
whh =[0.21    0.22   0.23
     0.31    0.44   0.26
     0.41    0.54   0.36];
why=[0.47 0.97 0.83
    0.39 0.28 0.65
    0.84 0.29 0.33
    0.51 0.32 0.12];
bias=[0.41, 0.12, 0.63]';%bias initialised
ht(:,1)=[0.3 0.2 0.1]'; %assume ht has these values initally at t=1
%Forward pass only
%for t = 1:length(in)-1 %
%(y_out not feedback to network, only h feeback to network)
y_out(:,1)=why*ht(:,1);%the outputs at t=1 (inital value)
softmax_y_out(:,1)=softmax(y_out(:,1)); %the outputs at t=1(inital value)

for t = 1:3 % assume we want to see 3 steps
   ht(:,t+1)=tanh( whx*X(:,t)+whh*ht(:,t)+bias); %recurrent layer
   y_out(:,t+1)=why*ht(:,t+1) ;
   softmax_y_out(:,t+1)=softmax(y_out(:,t+1)); %output layer
end
%'print result ==================='
X, ht, y_out, softmax_y_out
% %%%result
% ht =
```

```
%
%    0.3000   0.5511   0.8938   0.9430
%    0.2000   0.4028   0.8479   0.8612
%    0.1000   0.8100   0.9146   0.9590
%
%
% y_out =
%
%    0.4180   1.3220   2.0017   2.0746
%    0.2380   0.8542   1.1805   1.2323
%    0.3430   0.8471   1.2985   1.3584
%    0.2290   0.5072   0.8369   0.8716
%
%
% softmax_y_out =
%
%    0.2785   0.3716   0.4451   0.4505
%    0.2326   0.2328   0.1958   0.1941
%    0.2584   0.2311   0.2203   0.2201
%    0.2305   0.1645   0.1389   0.1353
```

//

A57074.3.8: recurrent neural network output system

The output system of a recurrent neural network (RNN) is as follows.

If $h_t(1)=0.45$, $h_t(2)=0.56$, $h_t(3)=0.84$

```
why=[0.24   0.27    0.35
     0.15   0.18    0.21
     0.25   0.26    0.64
     0.86   0.13    0.41]
```

Find Softmax_y_out(3).

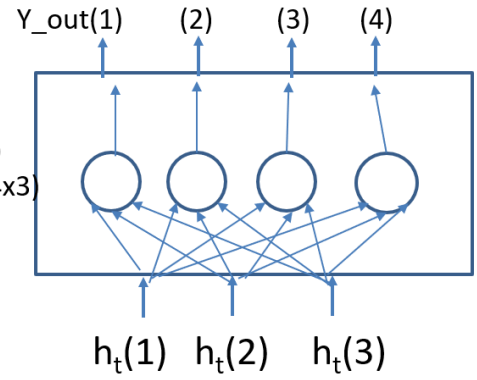
why=

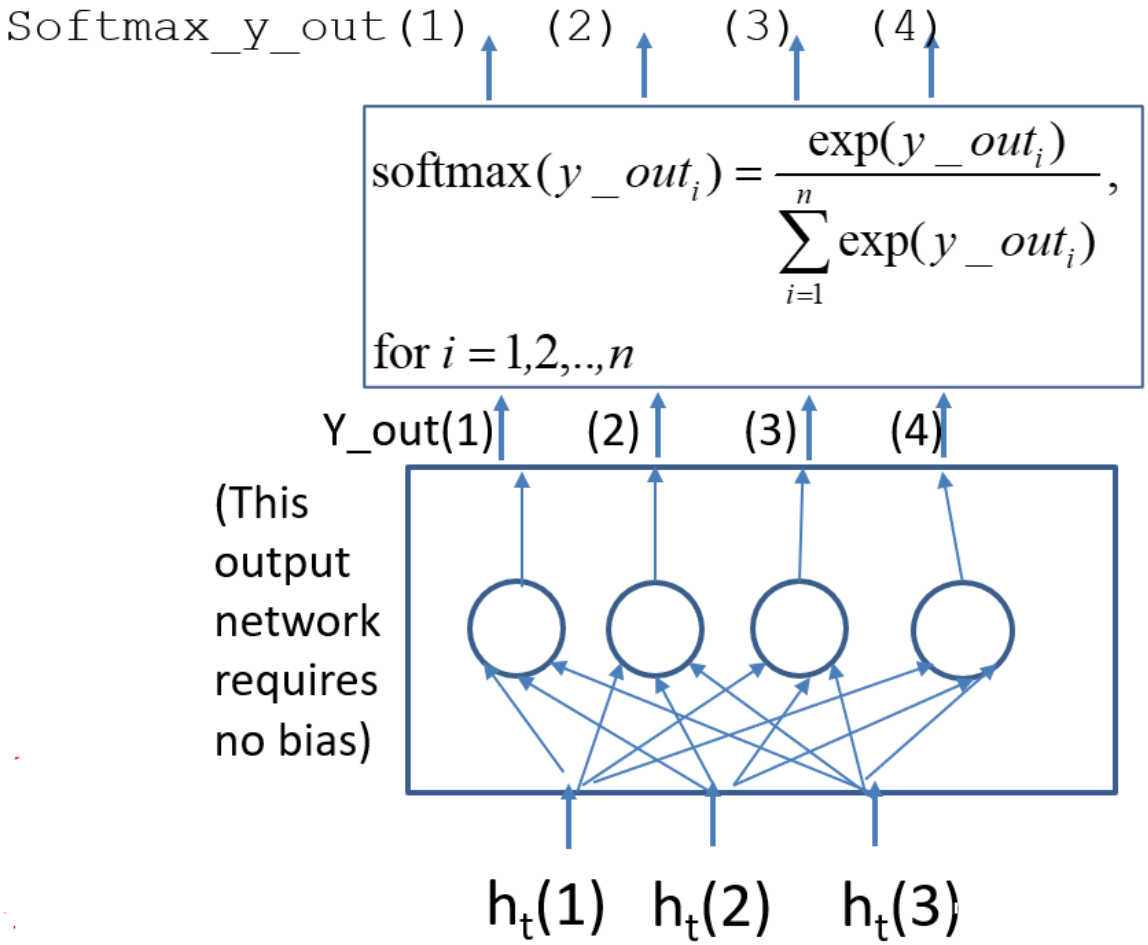
why(1,1)	why(1,2)	why(1,3)
why(2,1)	why(2,2)	why(2,3)
why(3,1)	why(3,2)	why(3,3)
why(4,1)	why(4,2)	why(4,3)

- `Y_out(1)=why(1,1)*ht(1)+why(1,2)*ht(2)+ why(1,3)*ht(3)`
- `Y_out(2)=why(2,1)*ht(1)+why(2,2)*ht(2)+ why(2,3)*ht(3)`
- `Y_out(3)=why(3,1)*ht(1)+why(3,2)*ht(2)+ why(3,3)*ht(3)`
- `Y_out(4)=why(4,1)*ht(1)+why(4,2)*ht(2)+ why(4,3)*ht(3)`

h , Y_out are fully connected so
Weights(the "why" variable)=(4x3)

(This output network requires no bias)





-

Answer8: 0.29153 (ok4)

```
clear,clc, format shortG
ht=[0.45,0.56,0.84]';

t=0;% so t+1 is 1
why=[0.24  0.27   0.35
    0.15  0.18   0.21
    0.25  0.26   0.64
    0.86  0.13   0.41]
y_out(:,t+1)=why*ht(:,t+1)
softmax_y_out(:,t+1)=softmax(y_out(:,t+1)) %output layer

% softmax_y_out =
%
%      0.22875
%       0.1857
%      0.29153
%      0.29402
```

//

A57074.3.9 : LSTM

For an Long short-term memory (LSTM) neural network , Input =200 neurons, Output = 100 neurons, Hidden Layers = 4, Cells in each hidden layer = 256 neurons (same for all hidden layers).
Note: for the output layer, the last hidden layer uses the sigmoid activation function to be sent to the output neurons.

The total number of weights is W, and the total number of biases is B in this neural network.
Find W+B.

Answer9: 2069604

%Matlab

clear

L=4, n=200, m_i1=256, m_i2=256, m_i3=256,m_i4=256,my=100

Total_weights=4*(m_i1+n)*m_i1 + 4*m_i2*(m_i1+m_i2)+ 4*m_i3*(m_i2+m_i3)+ 4*m_i4*(m_i3+m_i4) +my*m_i4

Total_biases=4*(m_i1) + 4*(m_i1)+4*(m_i1)+4*(m_i1)+my

Total_weights+ Total_biases

%= 2069604

- Reference

LSTM equation
In lecture notes example 1 (page 59)
inhttp://www.cse.cuhk.edu.hk/~khwong/www2/cmsc5707/5707_11_rnn_lstm.pptx

- Question1: Input 39, Output 34, Hidden Layers = 3, Cells in each layer = 1024
- Answers: Each cell in the LSTM has four components: the cell weights, the input gate, the forget gate, and the output gate. Each component has weights associated with all of its input from the previous layer, plus input from the previous time step. So if there are m_i cells in an LSTM layer, and m_{i-1} in the earlier layer, there will be $(m_{i-1}+m_i)$ inputs to each component of the cell. Since there are four components, that means there are $4(m_{i-1}+m_i)$ weights associated with each cell. And since we have m_i cells, that means there are $4m_i(m_{i-1}+m_i)$ weights associated with that layer.
- If the last hidden layer has m_i cells, and the number of real output is m_y. For the output layer, the h of the last layer will be combined by a softmax (or sigmoid) activation function, hence the weights required is m_i*m_y.

- Answer1: Since n_i=num. of neurons in layer i, so we have n=39 (input neuron number) ,$m_{i=1}$= $m_{i=2}$=$m_{i=3}$ =1024, and m_y=34 (output neuron number). So the overall number of weights = input_to_first_layer_connections+ first_to_second_layer_connections+ second_to_third_layer_connections+third_layer_to_output_connections=
- $=4*(m_{i=1}+n)*m_{i=1}+4*m_{i=2}*(m_{i=1}+m_{i=2})+ 4*m_{i=3}*(m_{i=2}+m_{i=3})+ m_y*(m_{i=3})$
- =4*(1024+39)*1024 +4*1024*(1024+1024)+ 4*1024*(1024+1024) +34*+(1024)=21,166,080 (about 21M).

//
//

A57074.3.10 : Autoencoder

Multiple answers

Which of the following statement(s) is/are true?

1) In a classical/vanilla autoencoder the number of input and output neurons are the same.
2) In a variational autoencoder the number of input and output neurons are the same.
3) In a variational autoencoder a random generator is used inside the hidden layer to generate hidden data (z) from the hidden-mean-neurons and hidden-variance-neurons.
4) In a classical/vanilla autoencoder a random generator is used inside the hidden layer to generate hidden data (z) from the hidden-mean-neurons and hidden-variance-neurons.
5) A variational autoencoder can be used to generate new data from a small dataset with similar probability distribution values.

Answer10:
1,2,3 5 are true. (ok4)

//

A57074.3.11 : Autoencoder2 ,Multiple answers

Which of the following statement(s) is/are true?

1) A random process is involved in a variational autoencoder so that backpropagation can be executed directly.
2) A random process is involved in a variational autoencoder so that backpropagation cannot be executed directly.
3) The Re-parameterization trick in a variational autoencoder can turn a random variable into a real number.
4) The Re-parameterization trick in a variational autoencoder can turn a real number into a random variable.

Answer11:
2,3 are true. (ok4)

//////////////////
A57074.3.12: the vanishing gradient problem.

Which of the following statement(s) is/are true?

1) Using Sigmoid as the activation function for a neuron will cause the vanishing gradient problem.
2) Using Relu as the activation function for a neuron will cause the vanishing gradient problem.
3) LSTM can be used to reduce the effect of the vanishing gradient problem.
4) LSTM uses more memory compared to an RNN system with the same number of input and output.

Answer12:
1,3,4 are true. (ok4)

//////////////////////

A57074.3.13 : Transformer

A transformer encoder model similar to that described in the lecture notes (see the attached diagram) has the following specifications:

The system can handle a vocabulary of 50257 tokens.

Input sequence length= 2048.

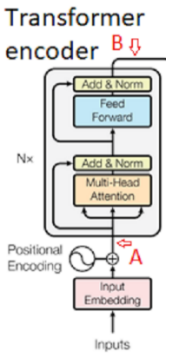
Input embedded vector size = 1024.

Number of attention heads= 96.

Dimension of its hidden layer = 12288.

Calculate the total number of weights used in between point A and B.

Note: Assume there is only one (linear + RELU) layer in the "Feed Forward" block in our example. (updated 2024.11.12)



Answer: 104857600 (ok4)

+/- 1

%This similar to chatgpt4

%Weights in W_token = 50257*1024 (it is outside A and B, so no need to add to the answer)

%it is from A to B, so the number of tokens is irrelevant (a vocabulary of 50257 tokens.)

%There are 96 heads and the hidden dimension Z is 12288,

%hence each head output size is 12288/96=128. Therefore

n_heads=96

W_Q =2048*128 %Weights in W_Q of a single head

W_K =2048*128 %Weights in W_K of a single head

W_V =2048*128 %Weights in W_V of a single head

W_multihead_linear =12288*2048 %= Dimension of hidden layer *sequence length

% =Weights in W_linear inside multi-head attention

W_feed_forward_linear=2048*2048 %=sequence length*sequence length

%=Weights in feed forward and relu layer,

'Total weight in the encoder is ='

W_total= n_heads*(W_Q+ W_K+ W_V)+ W_multihead_linear +W_feed_forward_linear

%=104857600

//////////////////////////

A57074.3.14:Q learning

Given learning rate =1, discount rate=0.8, if the R and the current Q tables are given as shown. Currently the agent is at state s=2 and action a=3 is selected. An entry in the Q table will be updated, write the updated value of that entry.

$$R = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} -1 & -1 & -1 & -1 & 0 & -1 \\ -1 & -1 & -1 & 0 & -1 & 100 \\ -1 & -1 & -1 & 0 & -1 & -1 \\ -1 & 0 & 0 & -1 & 0 & -1 \\ 0 & -1 & -1 & 0 & -1 & 100 \\ -1 & 0 & -1 & -1 & 0 & 100 \end{bmatrix} \end{array}$$

Action / State

$$Q = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 64 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 80 & 45 & 0 & 130 & 0 \\ 0 & 0 & 0 & 122 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

Answer14: 104 (OK4)
+/- 0.002
s=2, a=3, so the next state is 3.
At s=3 you have 3 choices, so the next state s'=1,2,4
The Q-update equation is
Q(s,a)=R(s,a)+0.8*(max(Q(3,1),Q(3,2),Q(3,4))
Q(2,3)=R(2,3)+0.8*(max(80,45,130)
Q(2,3)=0+ 0.8*(max(80,45,130)
Q(2,3)=0+ 0.8*130=104

- Q(4,3) = R(4,3) + 0.8 argmax(Q(s',a'))
- Q(4,3) = R(4,3) + 0.8 argmax(Q(4,0),Q(4,3),Q(4,5))
- Note: The other choices are not allowed because, i.e. R(4,1),R(4,2),R(4,4) are -1.
- Hence
- Q(4,3) =0+ 0.8 argmax(0,122,100)
- Q(4,3) =0.8*122=97.6

Suggested answers for Q15-18
Ch6 Autoencoder; ch7 RNN/LSTM; ch8 word representation/embedding; Ch9 Transformer; ch10 Decision tree; ch11 Fuzzy logic; ch12 Q learning
//

A57074.3.15: word embedding, COLAB
Run the code in the following link in COLAB
https://colab.research.google.com/drive/1N7HELWImK9xCYheyozVP3C_McbiRo1nb#scrollTo=dfo-brEm-STg
The following modification is needed to run the code successfully:
vocab = vectorizer.get_feature_names_()
should be replaced by
vocab = vectorizer.get_feature_names_out_()
Run the code from the beginning till the part before "Word2Vec and GloVe" to learn how to use the libraries.
You are now given 4 sentences:
(sentence1): 'The sun has long been set.',
(sentence2): 'The stars are out by twos and threes.',
(sentence3): 'The little birds are piping yet.',
(sentence4): 'Among the bushes and trees.',

Find the TF-IDF cosine similarity between sentence 2 and 4.
Hints: use the COLAB program in the above link to find the TF-IDF encoding vector. Then, you may use python (or Matlab) for the cosine similarity calculation.

Answer15: 0.177 (ok4)
+/- 0.002

Answer: code in python, can run in colab

```
from sklearn.feature_extraction.text import TfidfVectorizer
import seaborn as sns
corpus = [
'The sun has long been set.',
'The stars are out by twos and threes.',
'The little birds are piping yet.',
'Among the bushes and trees.',
]
term_frequencies = vectorizer.fit_transform(corpus)
vocab = vectorizer.get_feature_names_out()
vocab
vector_a= tfidf[1, :]
print(vector_a)
vector_b= tfidf[3, :]
print(vector_b)

dot_product = np.dot(vector_a, vector_b)

norm_vector_a = np.linalg.norm(vector_a)

norm_vector_b = np.linalg.norm(vector_b)

cosine_similarity = dot_product / (norm_vector_a * norm_vector_b)

print("Cosine Similarity:", cosine_similarity)
----------printout----------
[0.        0.30887228 0.30887228 0.        0.        0.
 0.39176533 0.        0.        0.        0.39176533 0.
 0.        0.39176533 0.        0.2044394  0.39176533 0.
 0.39176533 0.        ]
```

[0.50676543 0.39953968 0. 0. 0. 0.50676543
 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0.26445122 0. 0.50676543
 0. 0.]
Cosine Similarity: 0.1774709834473492
//
A57074.3.16: word embedding word2vec, programming using Anaconda,
It is suggested you install anaconda (https://www.anaconda.com/) and use it for this exercise. Please also read the tutorial at https://code.google.com/archive/p/word2vec/
Download the pretrained word2vec ('GoogleNews-vectors-negative300.bin.gz') from https://drive.google.com/file/d/0B7XkCwpI5KDYNlNUTTlSS21pQmM/edit?resourcekey=0-wjGZdNAUop6WykTtMip30g
Unzip 'GoogleNews-vectors-negative300.bin.gz' into a directory of your computer.
In the same directory you may use the following .py program segment to read the word2vec data.

```
#code begins ########################################################################
import numpy as np
import genism # >pip install genism #if needed
model = gensim.models.KeyedVectors.load_word2vec_format('GoogleNews-vectors-negative300.bin', binary=True)

rocket = model['rocket']
#Add your own code here.
#
#
#code ends ###########################################################################
```

If
A= cosine_similarity between the word "rocket" and "plane",
B= cosine_similarity between the word "ship" and "banana".

Find A-B.

//////////////////////////
Answer 16 : A-B= 0.1837 (ok4)
+/- 0.002

```
import numpy as np
import gensim
model = gensim.models.KeyedVectors.load_word2vec_format('GoogleNews-vectors-negative300.bin', binary=True)

rocket = model['rocket']
plane = model['plane']
ship = model['ship']
banana = model['banana']

print(rocket.shape)
print(plane[:10]) #display only 10 numbers, can be more if you use "print(plane)"
print(ship.shape)
print(banana[:10])

#####################################################################
vector1 = rocket
vector2 = plane
dot_product = np.dot(vector1, vector2)
norm_vector1 = np.linalg.norm(vector1)
norm_vector2 = np.linalg.norm(vector2)
A = dot_product / (norm_vector1 * norm_vector2)
```

```
print("Cosine Similarity_A:", A)

#####################################################################
vector1 = ship
vector2 = banana
dot_product = np.dot(vector1, vector2)
norm_vector1 = np.linalg.norm(vector1)
norm_vector2 = np.linalg.norm(vector2)
B = dot_product / (norm_vector1 * norm_vector2)
print("Cosine Similarity B:", B)
print("result ")
print("A-B=",A-B)
#//#result//////////////////////////////////////////////////////////////////////////////////////////////////////////
(base) C: \dataset>python  test1.py
(300,)
[ 0.24804688 -0.23339844 -0.15234375 -0.15039062 -0.03515625 -0.39257812
 -0.25585938 -0.32226562  0.41601562 -0.14746094]
(300,)
[-0.08544922  0.04711914 -0.06933594  0.30273438 -0.1875     -0.03198242
  0.02954102 -0.20507812 -0.09033203  0.29882812]
Cosine Similarity_A: 0.27108428
Cosine Similarity B: 0.0873477
result
A-B= 0.18373658
//////////////////////////////////////
```

A57074.3.17: Reinforcement learning1 (multiple answer question)

Run the following steps to learn how a Q-learning program written in Python works.

1. Study how to run a python programing using anaconda(python system). You may need libraries : gymnasium, numpy, matplotlib, pickle to successfully run the program. I.e. if you missed the library gymnasium, try to run "> pip install gymnasium" under anaconda.

2. Download https://github.com/johnnycode8/gym_solutions. Edit the program frozen_lake_q.py

3. Based on the original frozen_lake_q.py , at the end of the program,

    - change the line "#run(15000)" to "run(999)".

    - Comment the following line

      run(1000, is_training=True, render=True),
      i.e. # run(1000, is_training=True, render=True)
      Run frozen_lake_q.py
      Observe the display and see if the man can find the exit (treasure box) or not.

4. Based on the original frozen_lake_q.py, at the end of the program,

    - change the line "#run(15000)" to "run(15000)".

    - Comment the line : run(1000, is_training=True, render=True),

      i.e. # run(1000, is_training=True, render=True)
      Run frozen_lake_q.py
      Observe the display and see if the man can find the treasure box or not.

5. Run the original frozen_lake_q.py

      Observe the display and see if the man can find the treasure box or not.

If enough episodes were run, the man can find the treasure box.
No need to submit the code, just answer the following question.
Which of the following statement(s) is/are true?

1. When the episode number is set to 999, the learning is not successful to solve the problem.
2. When the episode number is set to15000, the learning is successful to solve the problem.
3. The Q table is stored in the file frozen_lake8x8.png.
4. During learning, the value of epsilon is changing from 0 to 1
5. The value of "reward" in the program starts to rise from episode 12000 onward.

Answer17: 1,2 are true (ok4)

1. True: When the episode number is set to 999, the learning is not successful to solve the problem. True
2. True: When the episode number is set to15000, the learning is successful to solve the problem. True
3. False: The Q table is stored in the file frozen_lake8x8.png. False, it is stored in frozen_lake8x8.pkl.
4. False: During learning, the value of epsilon is changing from 0 to 1. False, it is the reverse, from 1 to 0.
5. False: The value of "reward" in the program starts to rise from episode 12000 onward. False, it is from 10000

//

A57074.3.18: Reinforcement learning2 (multiple answer question)
Run the following steps to learn how a deep Q-learning (dql) program written in Python works.

1) Study how to run a python programing using anaconda(python system). You may need libraries : gymnasium, numpy, matplotlib, pickle to successfully run the program. I.e. if you missed the library gymnasium, try to run "> pip install gymnasium" under anaconda.
2) Download https://github.com/johnnycode8/gym_solutions. Edit the program frozen_lake_dql.py
3) Run frozen_lake_dql.py

No need to submit the code, just answer the following question.
Which of the following statement(s) is/are true?

1. The neural network parameters are stored in frozen_lake_dql.pt.
2. The number of neurons in the hidden layer of the neural network policy_dqn is 32.
3. This DQN employs 2 neural networks of the same size.

4. Replay memory is used to store neural work weights and biases.
5. Copy the target network to policy network once every 10 steps.

//
Answer 18: 1,3 are true (ok4)

1. True : The neural network parameters are stored in frozen_lake_dql.pt.
2. False: The number of neurons in the hidden layer of the neuerl network policy_dqn is 32. False. Because the line says: policy_dqn = DQN(in_states=num_states, h1_nodes=num_states, out_actions=num_actions). Num-stataes=16, you can print this to see.
3. True: This DQN employs 2 neural networks of the same. Size. Ture:

policy_dqn = DQN(in_states=num_states, h1_nodes=num_states, out_actions=num_actions) and
target_dqn = DQN(in_states=num_states, h1_nodes=num_states, out_actions=num_actions)

4. False: Replay memory is used to store neural work weights and biases. False, it is to store experience of states and actions and reward performed this deep Q learning method.
5. False: Copy the target network to policy network once every 10 steps. False: it is the reverse. See # Copy policy network to target network after a certain number of steps

```
if step_count > self.network_sync_rate:
   target_dqn.load_state_dict(policy_dqn.state_dict())
   step_count=0
```