Assignment 2: Learning and Memory PSY 306 (Winter 2023)

Name: Shagun Mohta

Roll Number: 2020468

Instructions: Please write your own responses and do not copy or lift text/code from any source. If you are referring to credible external sources other than the attached paper for your answers, please cite those sources (within the body of text and the provide a reference list at the end) in the APA citation format (https://www.mendeley.com/guides/apa-citation-guide). Word limits given are indicative and less than the indicated numbers may also be used.

Please download this MS word question-cum-response template to TYPE your answers and feel free to add sheets as required. Convert this document to a PDF before submitting. Please note that answers in this template only will be evaluated and hand-written or scanned answer sheets will not be evaluated. Please submit ONLY ONE PDF and no extra files as it increases the time to evaluate them. DO NOT change the basic structure of the template. DO NOT remove the marks assigned for each question.

[Strict deadline for submission: 23 March 2023, 11 PM]

Q2) Please read the following for this question:

- **A researcher recorded electromyogram (EMG) from the extraocular muscles of a human participant as a tone was delivered through headphones and air-puff delivered to the eyes through an apparatus to the participant. The tone stimulus onset is at time = 0 ms (beginning of the trial) and continues until 650 ms. The air-puff stimulus onset is at time = 600 ms and continues for the next 50 ms.**
- **The above was done for five trials/day for four subsequent days and the EMG responses recorded as data. Download the attached data file- Data-Assignment2A.xlsx**
- **Each sheet of the excel file contains EMG recording from one day of experiment. Each sheet has 5 rows (trials) x 1000 columns (EMG amplitudes recorded at an interval of 1 millisecond). Thus each row has 1000 ms (1 second) of recording.**

Now do the following...

Insert a figure (wherever required) and paste the MATLAB/Python code for the same. All figures must be properly labelled, carry necessary units of measurement with accompanying captions/legends to provide all information necessary to interpret the figures.

A) Run the following steps…

- **Take the average of data across all trials per day for each time point to get one averaged signal per day.**
- **Run a 'moving average filter' across the averaged signal with a window width of 20 ms to get a filtered signal. Ensure that the raw and filtered signal are of the same length.**
- **Do a full wave rectification of the above moving average filtered signal.**
- **Plot the amplitude vs time of the raw signal (as blue curve) - one signal for each day in four different subplots of one bigger plot.**
- **Plot the amplitude vs time of the filtered and rectified signal (as red curve) – one signal for each day on top of the raw signal in the same subplots.**

Step1:

```
from google.colab import drive
drive.mount('/content/gdrive')
```

Step2: The sheets corresponding to the days of experiment are read and stored into 4 different variables.

```
import pandas as pd
filep= '/content/gdrive/My Drive/LM_datasets_Assignment/Data-Assignment2A.xlsx'
file = pd.ExcelFile(filep)
day1 = pd.read_excel(file, 'Day1')
day2 = pd.read_excel(file, 'Day2')
day3 = pd.read_excel(file, 'Day3')
day4 = pd.read_excel(file, 'Day4')
```

day1

	Unnamed: 0	Unnamed: 1	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 5	Unnamed: 6	Unnamed: 7	Unnamed: 8	Unnamed: 9	...	Unnamed: 991	Unnamed: 992	Unnamed: 993	Unnamed: 994	Unnamed: 995	Unnamed: 996	Unnamed: 997	Unr
0	Trial1	0.307055	0.331403	0.006869	-0.937137	0.526691	0.573682	0.647855	-0.234308	-0.292786	...	0.763173	-0.730888	0.737645	0.961773	-0.260038	-0.611114	-0.032268	-0.1
1	Trial2	-0.160209	0.318525	-0.160922	0.870977	0.424793	0.905127	0.830739	-0.634645	0.337049	...	0.137907	0.228041	-0.699972	-0.385863	0.452322	0.128223	-0.708533	0.1
2	Trial3	0.347212	-0.412225	-0.715740	0.926689	0.141086	-0.590183	-0.472404	-0.818458	0.357048	...	-0.703881	-0.416329	0.273860	-0.101408	0.733754	0.387311	0.207557	-0.2
3	Trial4	0.949109	-0.837623	-0.143994	-0.836881	0.705629	0.767804	-0.547494	-0.042279	0.277254	...	0.573966	0.076782	0.018069	0.113932	-0.361941	-0.177406	0.808901	0.7
4	Trial5	-0.627351	0.353184	-0.285398	-0.639642	-0.173073	0.601924	-0.497433	0.329976	-0.646070	...	0.163568	0.384635	-0.712877	-0.303991	0.246786	0.717954	-0.056864	0.3

5 rows × 1001 columns

Step3: The averages of all the columns across the five trials are calculated and appended to the arrays for each day. The length of each array is 1000 as the number of columns/ time frames is 1000.

```
#Take the average of data across all trials per day for each time point to get one averaged signal per day.

day1_averages=[]
day2_averages=[]
day3_averages=[]
day4_averages=[]

for column in day1.iloc[: , 1:]:
  day1_averages.append(day1[column].mean())

for column in day2.iloc[: , 1:]:
  day2_averages.append(day2[column].mean())

for column in day3.iloc[: , 1:]:
  day3_averages.append(day3[column].mean())

for column in day4.iloc[: , 1:]:
  day4_averages.append(day4[column].mean())
```

Step3: A moving average filter called filter is generated for the window size of 20. Using numpy's convolve function, the filtered signals are generated. The same mode enables the output array to be the same as the input.

```
import numpy as np
window_size = int(20)
filter = np.ones( window_size) / window_size
day1_filtered_signal = np.convolve(day1_averages, filter, mode='same')
day2_filtered_signal = np.convolve(day2_averages, filter, mode='same')
day3_filtered_signal = np.convolve(day3_averages, filter, mode='same')
day4_filtered_signal = np.convolve(day4_averages, filter, mode='same')
```

Step4:

```
#same length
day1_filtered_signal = day1_filtered_signal[:len(day1_averages)]
day2_filtered_signal = day2_filtered_signal[:len(day2_averages)]
day3_filtered_signal = day3_filtered_signal[:len(day3_averages)]
day4_filtered_signal = day4_filtered_signal[:len(day4_averages)]
```

Step5: To perform rectification, negative values are converted into positive, by taking absolute.

```
# Full wave rectification of the above moving average filtered signal

r_signal_day1= np.abs(day1_filtered_signal)
r_signal_day2= np.abs(day2_filtered_signal)
r_signal_day3= np.abs(day3_filtered_signal)
r_signal_day4= np.abs(day4_filtered_signal)
```

Step6: Using matplotlib library's subplots function, 4 subplots are created. Each subplot then contains the plot of amplitude of raw signals in blue and amplitude of rectified and filtered signals in red.

```
#  Plot the amplitude vs time of the raw signal (as blue curve) - one signal for each day in four different subplots of one bigger plot

import matplotlib.pyplot as plt

time=[]
for i in range(1000):
  time.append(i)

fig, axs = plt.subplots(2, 2, figsize=(20, 15))
axs[0, 0].plot(time,day1_averages,color="blue", label='Raw signal')
axs[0, 0].plot(time,r_signal_day1,color="red", label='Filtered and Rectified signal')
axs[0, 0].set_title("Day1")
axs[0, 0].set_xlabel('Time (milliseconds)')
axs[0, 0].set_ylabel('Amplitude of the signal (metres)')
axs[0, 0].set_xlim(0, 1000, 50 )
axs[0, 0].legend()

axs[0, 1].plot(time,day2_averages,color="blue", label='Raw signal')
axs[0, 1].plot(time,r_signal_day2,color="red", label='Filtered and Rectified signal')
axs[0, 1].set_title("Day2")
axs[0, 1].set_xlabel('Time (milliseconds)')
```

```
axs[0, 1].set_ylabel('Amplitude of the signal (metres)')
axs[0, 1].set_xlim(0, 1000, 50 )
axs[0, 1].legend()

axs[1, 0].plot(time,day3_averages,color="blue", label='Raw signal')
axs[1, 0].plot(time,r_signal_day3,color="red", label='Filtered and Rectified
signal')
axs[1, 0].set_title("Day3")
axs[1, 0].set_xlabel('Time (milliseconds)')
axs[1, 0].set_ylabel('Amplitude of the signal (metres)')
axs[1, 0].set_xlim(0, 1000, 50 )
axs[1, 0].legend()

axs[1, 1].plot(time,day4_averages,color="blue", label='Raw signal')
axs[1, 1].plot(time,r_signal_day4,color="red", label='Filtered and Rectified
signal')
axs[1, 1].set_title("Day4")
axs[1, 1].set_xlabel('Time (milliseconds)')
axs[1, 1].set_ylabel('Amplitude of the signal (metres)')

axs[1, 1].set_xlim(0, 1000, 50 )
axs[1, 1].legend()

fig.suptitle("Plot representing EMG amplitudes recorded at an interval of 1
millisecond for 4 different days of experiment ")
fig.subplots_adjust(top=0.94)


plt.legend()
plt.show()
```

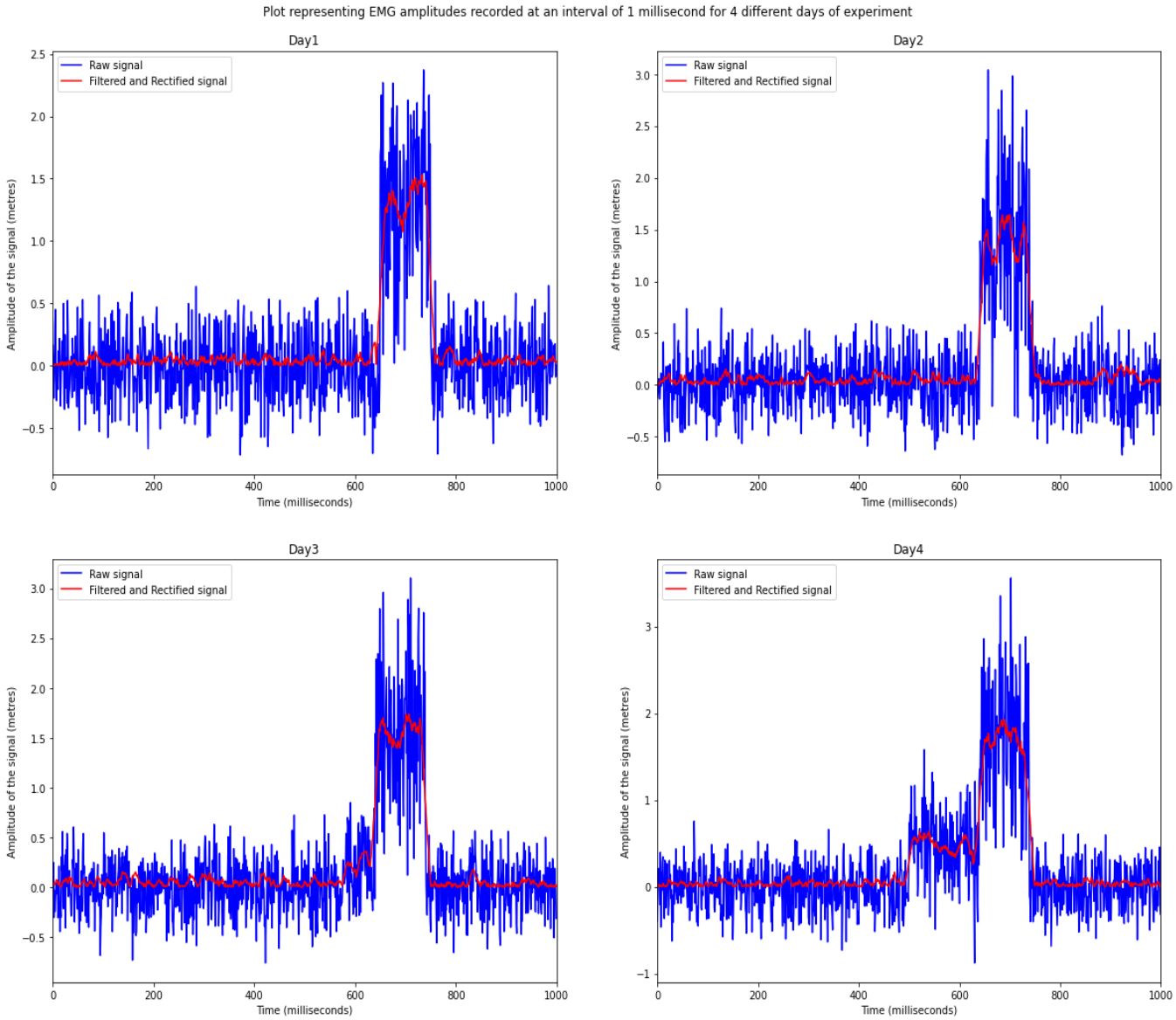

Fig.1. The above figure represents four subplots corresponding to the results obtained for four days of experiment. Each subplot represents raw and unfiltered EMG signals in blue and FIltered and Rectified signals in red curve. The y- axis of the subplots represents the amplitude of the signals in meters and the x-axis represents the time in milliseconds.

After creating the above plot, explain the learning mechanism evident in the above plot with all necessary components of learning that you think are involved in this case. Calculate exact time points of the peaks from the above data to draw your quantitative conclusions about the learning mechanism and its components. [10 points]

The participant gets exposed to 2 stimuli types, one is tone delivered through headphones and the other is air puff delivered to the eyes of the participant. The EMG signals record the electrical brain activity of the participant when he/she is exposed to the stimuli. The filtered and rectified signal is much easier to interpret and analyze than the raw signal because the external noise has been removed.

In the above plots, it can be seen that the signal stays constant over time until there is a sudden spike in the amplitude in the time interval 600- 650 ms. This is also the time interval when the air puff stimulus is introduced to the participant, when he/she is already exposed to the tone stimulus.

What can be inferred from this is that upon interference of 2 stimuli, neuronal activity increases. This leads to an increase in the attention of the participant and also learning is affected.

The following code can be used to find the peak occurrences and their time points of occurrence in the days of experiment

```
#find the peak occurrences
peak=[]
max1= max2= max3= max4=0

for i in range(len(r_signal_day1)):
  if r_signal_day1[i]>=max1:
    max1=r_signal_day1[i]


  if r_signal_day2[i]>=max2:
    max2=r_signal_day2[i]


  if r_signal_day3[i]>=max3:
    max3=r_signal_day3[i]


  if r_signal_day4[i]>=max4:
    max4=r_signal_day4[i]
```

```
peak.append(max1)
peak.append(max2)
peak.append(max3)
peak.append(max4)

peak
```

```
[1.5305916323041953, 1.6416850532826397, 1.736882482369475, 1.928549520028098]
```

The above figure represents the peak amplitude of the filtered and rectified signal for the 4 days - day1, day2, day3 and day4 respectively.

```
#find occurences of the peak in all the days
day_1=[]
day_2=[]
day_3=[]
day_4=[]

for i in range(len(r_signal_day1)):
  if r_signal_day1[i]==max1:
    day_1.append(i)


  if r_signal_day2[i]==max2:
    day_2.append(i)


  if r_signal_day3[i]==max3:
    day_3.append(i)


  if r_signal_day4[i]==max4:
    day_4.append(i)

print("The peak occurences of day 1 of experiment occurs at foll time points ",
day_1)
```

```
print("The peak occurences of day 2 of experiment occurs at foll time points ",
day_2)
print("The peak occurences of day 3 of experiment occurs at foll time points ",
day_3)
print("The peak occurences of day 4 of experiment occurs at foll time points ",
day_4)
```

```
The peak occurences of day 1 of experiment occurs at foll time points  [733]
The peak occurences of day 2 of experiment occurs at foll time points  [700]
The peak occurences of day 3 of experiment occurs at foll time points  [707]
The peak occurences of day 4 of experiment occurs at foll time points  [689]
```

To find the peak occurrences of the signal for all the days, maximum is found for the rectified signal's amplitude arrays corresponding to the day, and then the indices at which the maximum occurs is stored, which signifies the time moment when the peak occurs.

The learning mechanism here at play is Delayed Conditioning, a type of Classical Conditioning. The participant is provided with two stimuli, one is the tone, and the other is the air puff. The tone was the conditioned stimulus, CS, while the air puff was the unconditioned stimulus, US. The CS onset is 0 ms and continues till 650 ms, while the US onset is 600 ms and continues till 650 ms. Hence they overlap temporally for 50 ms, an example of delayed Conditioning.

We observe that repetition of the experiment increases the peak amplitude (brain neural activity increases). The peak occurrence time decreases with repetition, i.e. the brain activity is more pronounced in days 3 and 4 even before the unconditioned stimulus is introduced at 600 ms suggesting the participant responds to the conditional stimulus in a more pronounced manner with repetition, i.e. strong CR before the US. This denotes that the participant can associate the two stimuli better with repetition.

B) An experimenter carries out three pilot experiments of 30 trials each in human subjects to study the relationship between time (# trials) and Associative learning between the exposure to sets of environmental stimuli (Conditioned and Unconditioned Stimuli). She collects and averages the data across equal number of subjects for each pilot experiment. This data is entered in the Data-Assignment2B.xlsx. Each row = 1 pilot experiment. Each column is the value/magnitude of the CR (arbitrary units). Now carry out the following...

i) Computationally estimate the Rates and Asymptotes of Learning for the three pilot experiments. Create three subplots for three experiments as part of one larger plot to graph the individual data points (as open circle markers; black color) and overlay of the learning curve (blue color) on each subplot. Indicate the Learning rate and Learning asymptote on top of each subplot (as title).

Also, report any one metric of "goodness of fit" for each of the three learning curves to the underlying experimental data and briefly explain the quality of your curve fit to the experimental data based on the metric.

Hint: - Use unconstrained nonlinear optimization to find the optimal parameters of the negatively accelerated learning curve which best describes the relationship within the data, quantitatively.
For a measure of goodness of curve fit to the experimental data, explore and report any one of these metrics - sum of squared errors OR R square OR adjusted R square.

Step1: Dataframe df is created after reading the content from the file descriptor using pandas library in python

```
from google.colab import drive

drive.mount('/content/gdrive')
```

```
import pandas as pd

filep= '/content/gdrive/My Drive/LM_datasets_Assignment/Data-Assignment2B.xlsx'
```

```
file = pd.ExcelFile(filep)
```

```
df = pd.read_excel(file)
```

```
df
```

	Pilot Exp1	0	0.00935922081949758	0.120125321626742	0.214970737441055	0.296183787601989	0.365723885663819	0.425268813835661	0.476255202549655	0.519913191800936	...	0.739595284
0	Pilot Exp2	0	0.466184	0.689913	0.810184	0.874839	0.909595	0.928279	0.938324	0.943723	...	0
1	Pilot Exp3	0	0.398383	0.620782	0.720026	0.764312	0.784075	0.792893	0.796829	0.798585	...	0

2 rows × 31 columns

The figure represents the dataframe df obtained

Step2: The 3 rows are converted to arrays representing the 3 pilot experiments.

```
exp1=list(df)
```

```
exp1=exp1[1:]
```

```
exp2=list(df.iloc[0])
```

```
exp2=exp2[1:]
```

```
exp3=list(df.iloc[1])
```

```
exp3=exp3[1:]
```

```
len(exp3)
```

Step3: A negatively accelerated learning curve is assumed as `a * (1 - np.exp(-b*x)`, **a represents the asymptote, b represents the learning rate.**

```
def learning_curve(x, a, b):
    return a * (1 - np.exp(-b*x))
```

```
import numpy as np
x=[]
for i in range(30):
  x.append(i)



y1=exp1
y2=exp2
y3=exp3
```

\

Step4: Using scipy library, curve_fit function is used to find the optimal values of asymptote and the learning rate.

```
from scipy.optimize import curve_fit

vars=[]
opt, cov = curve_fit(learning_curve, x, y1)
print("\nAsymptote value for Experiment 0 is ", opt[0])
print("Learning Rate for Experiment 0 is ", opt[1])
vars.append(opt)

opt, cov = curve_fit(learning_curve, x, y2)
print("\nAsymptote value for Experiment 1 is ", opt[0])
print("Learning Rate for Experiment 1 is ", opt[1])
vars.append(opt)

opt, cov = curve_fit(learning_curve, x, y3)
print("\nAsymptote value for Experiment 2 is ", opt[0])
```

```
print("Learning Rate for Experiment 2 is ", opt[1])

vars.append(opt)
```

```
Asymptote value for Experiment 0 is  0.8049227102009937
Learning Rate for Experiment 0 is  0.12475744609107828

Asymptote value for Experiment 1 is  0.9493365957078986
Learning Rate for Experiment 1 is  0.6527688198681477

Asymptote value for Experiment 2 is  0.8008660300046294
Learning Rate for Experiment 2 is  0.7297223359177207
```

The above values were observed.

Step5: Using the above parameter, values are predicted and stored into arrays.

```
y1_pred=[]

y2_pred=[]

y3_pred=[]


for i in x:

  y1_pred.append(learning_curve(i,vars[0][0] , vars[0][1]))

  y2_pred.append(learning_curve(i,vars[1][0] , vars[1][1]))

  y3_pred.append(learning_curve(i,vars[2][0] , vars[2][1]))
```

Step6: Matplotlib library is used to create 3 subplots, each subplot will contain a scatter plot with black open circles for data points, which is obtained by making the arg facecolors='None', and the learning curve is plotted in blue using the plot function.

```
import matplotlib.pyplot as plt


fig, axs = plt.subplots(1, 3, figsize=(20, 15))
```

```
axs[0].scatter(x, y1, facecolors='none', color='black', label='Data Points')

axs[0].plot(x, y1_pred, color='blue', label='Learning Curve')

axs[0].set_title('Asymptote value for Experiment 0: '+ str(vars[0][0]) + '
\nLearning Rate for Experiment 0: ' + str(vars[0][1]))



axs[1].scatter(x, y2, facecolors='none', color='black', label='Data Points')

axs[1].plot(x, y2_pred, color='blue', label='Learning Curve')

axs[1].set_title('Scatter plot 2')

axs[1].set_title('Asymptote value for Experiment 1: '+ str(vars[1][0]) + '
\nLearning Rate for Experiment 1: ' + str(vars[1][1]))



axs[2].scatter(x, y3, facecolors='none', color='black', label='Data Points')

axs[2].plot(x, y3_pred, color='blue',  label='Learning Curve')

axs[2].set_title('Scatter plot 3')

axs[2].set_title('Asymptote value for Experiment 2: '+ str(vars[2][0]) + '
\nLearning Rate for Experiment 2: ' + str(vars[2][1]))


axs[0].set_xlim(0,30,5)

axs[0].set_ylim(0,1,0.1)

axs[0].set_xlabel('Number of Trials')

axs[0].set_ylabel('CR')

axs[0].legend()


axs[1].set_xlim(0,30,5)

axs[1].set_ylim(0,1,0.1)

axs[1].set_xlabel('Number of Trials')

axs[1].set_ylabel('CR')
```

```
axs[1].legend()



axs[2].set_xlim(0,30,5)

axs[2].set_ylim(0,1,0.1)

axs[2].set_xlabel('Number of Trials')

axs[2].set_ylabel('CR')

axs[2].legend()

fig.subplots_adjust(top=0.9)

plt.legend()

plt.show()
```

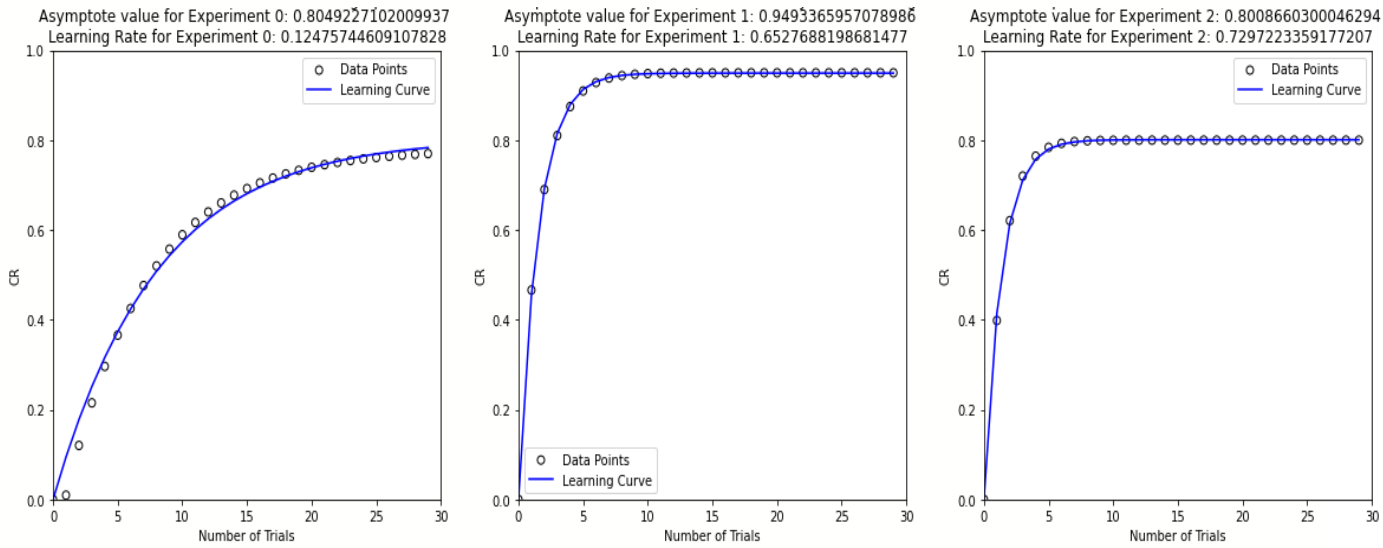

Fig.2. The above plot graphs the individual data points of the experiments and overlay of the learning curve for each subplot, representing the three pilot exeriments, with Number of Trials along the x-axis and CR along the y-axis. The titles of each subplot denote the Asymptote values and the corresponding Learning Rates of the curves. The black open circles represent Data Points and the blue curve represents the Learning Curve.

Step 7: The sse is chosen as the goodness of fit metric.

```
def sse(y_true, y_pred):

  sse=0
```

```
    for i in range(len(y_pred)):
        sse+=(y_true[i]-y_pred[i])**2
    return sse
```

```
print("The sum of squared error for exp0 is", sse(y1, y1_pred))
print("The sum of squared error for exp1 is", sse(y2, y2_pred))
print("The sum of squared error for exp2 is", sse(y3, y3_pred))
```

```
The sum of squared error for exp0 is 0.014802311193713677
The sum of squared error for exp1 is 0.00020274382961384498
The sum of squared error for exp2 is 0.0004650900809714351
```

The goodness of fit metric - sse values for the three experiments.

The goodness of fit measure used here is the sum of squared errors or sse, which is given by the sum of squares of the difference between the true data points and the predicted ones by the learning curve. The above produced overlay of the learning curve seems to overlap the data points and from the sse values being small in magnitude for the curves, the learning curve found is actually a decent fit of the above given data, especially for exp1 and 2, where sse values are very low.

ii) Based on your analysis of the data what can you conclude about the intensities of the Unconditioned Stimuli in the three pilot experiments and why? [8+2 points]

In general, a stronger CR implies a higher intensity of the US. In the plots with higher learning rates and stronger CR, the intensity of US is more, for example, in experiment 2 US might be at higher intensity . While in plots with lower learning rates, the US intensity would be lesser comparitively.