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

Name: Mudit Balooja

Roll Number: 2019258

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.**

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.

points]

[Answer]

Code and Explanation

- Importing libraries and dataset

```
Importing the Libraries

[55] import numpy as np
     import pandas as pd
     import matplotlib.pyplot as plt
     from scipy.ndimage import uniform_filter1d
     from scipy.optimize import curve_fit

Load data

[4]  xls = pd.ExcelFile('/content/drive/MyDrive/Data-Assignment2A.xlsx.xlsx')
     df_day1 = pd.read_excel(xls, 'Day1', index_col = 0)
     df_day2 = pd.read_excel(xls, 'Day2', index_col = 0)
     df_day3 = pd.read_excel(xls, 'Day3', index_col = 0)
     df_day4 = pd.read_excel(xls, 'Day4', index_col = 0)
```

- The average of all 5 trials is calculated across every time point to get one averaged signal per day

```
Step - 1

✓ 1s [17] # Taking the average of data across all trials per day for each time point to get one averaged signal per day.
     mean1 = df_day1.mean(axis = 0)
     mean2 = df_day2.mean(axis = 0)
     mean3 = df_day3.mean(axis = 0)
     mean4 = df_day4.mean(axis = 0)
```

- A moving average filter is run across the averaged signal with a window width of 20 ms to get a filtered signal.

```
Step - 2

✓ 0s [18] # running a 'moving average filter' across the averaged signal with a window width of 20 ms to get a filtered signal
     filtered_signal1 = uniform_filter1d(mean1, size = 20)
     filtered_signal2 = uniform_filter1d(mean2, size = 20)
     filtered_signal3 = uniform_filter1d(mean3, size = 20)
     filtered_signal4 = uniform_filter1d(mean4, size = 20)
```

- A full wave rectification is performed on the moving average filtered signal using np.abs()

```
Step - 3

✓ 0s [19] # full wave rectification of the above moving average filtered signal
     rectified_signal1 = np.abs(filtered_signal1)
     rectified_signal2 = np.abs(filtered_signal2)
     rectified_signal3 = np.abs(filtered_signal3)
     rectified_signal4 = np.abs(filtered_signal4)
```

- Amplitude vs Time of raw signal is plotted for each day

Step - 4

```
# Plot the amplitude vs time of the raw signal (as blue curve)
fig, ((plot1, plot2), (plot3, plot4)) = plt.subplots(2, 2, figsize = (16, 12))
fig.suptitle('Amplitude vs Time of raw and rectified signals for each day', y = 0.94, fontweight = 'bold')

plot1.plot(np.arange(1, 1001), mean1, color = 'blue', label = 'raw signal')
plot1.plot(np.arange(1, 1001), rectified_signal1, color = 'red', label = 'rectified signal')
plot1.set_xlabel('Time (ms)')
plot1.set_ylabel('Amplitude (mm)')
plot1.set_title('Amplitude vs Time of one signal for Day 1')
plot1.legend()

plot2.plot(np.arange(1, 1001), mean2, color = 'blue', label = 'raw signal')
plot2.plot(np.arange(1, 1001), rectified_signal2, color = 'red', label = 'rectified signal')
plot2.set_xlabel('Time (ms)')
plot2.set_ylabel('Amplitude (mm)')
plot2.set_title('Amplitude vs Time of one signal for Day 2')
plot2.legend()

plot3.plot(np.arange(1, 1001), mean3, color = 'blue', label = 'raw signal')
plot3.plot(np.arange(1, 1001), rectified_signal3, color = 'red', label = 'rectified signal')
plot3.set_xlabel('Time (ms)')
plot3.set_ylabel('Amplitude (mm)')
plot3.set_title('Amplitude vs Time of one signal for Day 3')
plot3.legend()

plot4.plot(np.arange(1, 1001), mean4, color = 'blue', label = 'raw signal')
plot4.plot(np.arange(1, 1001), rectified_signal4, color = 'red', label = 'rectified signal')
plot4.set_xlabel('Time (ms)')
plot4.set_ylabel('Amplitude (mm)')
plot4.set_title('Amplitude vs Time of one signal for Day 4')
plot4.legend()

caption = """Figure shows Amplitude vs Time(ms) of one signal across a time period of 1000 ms for 4 days.
            The amplitude is averaged over 5 trials for each day (blue curve)
            A moving average filter is passed through it and then rectified (red curve)"""

fig.text(0.5, 0.03, caption, ha = 'center')
```

- Subsequent plot is shown below

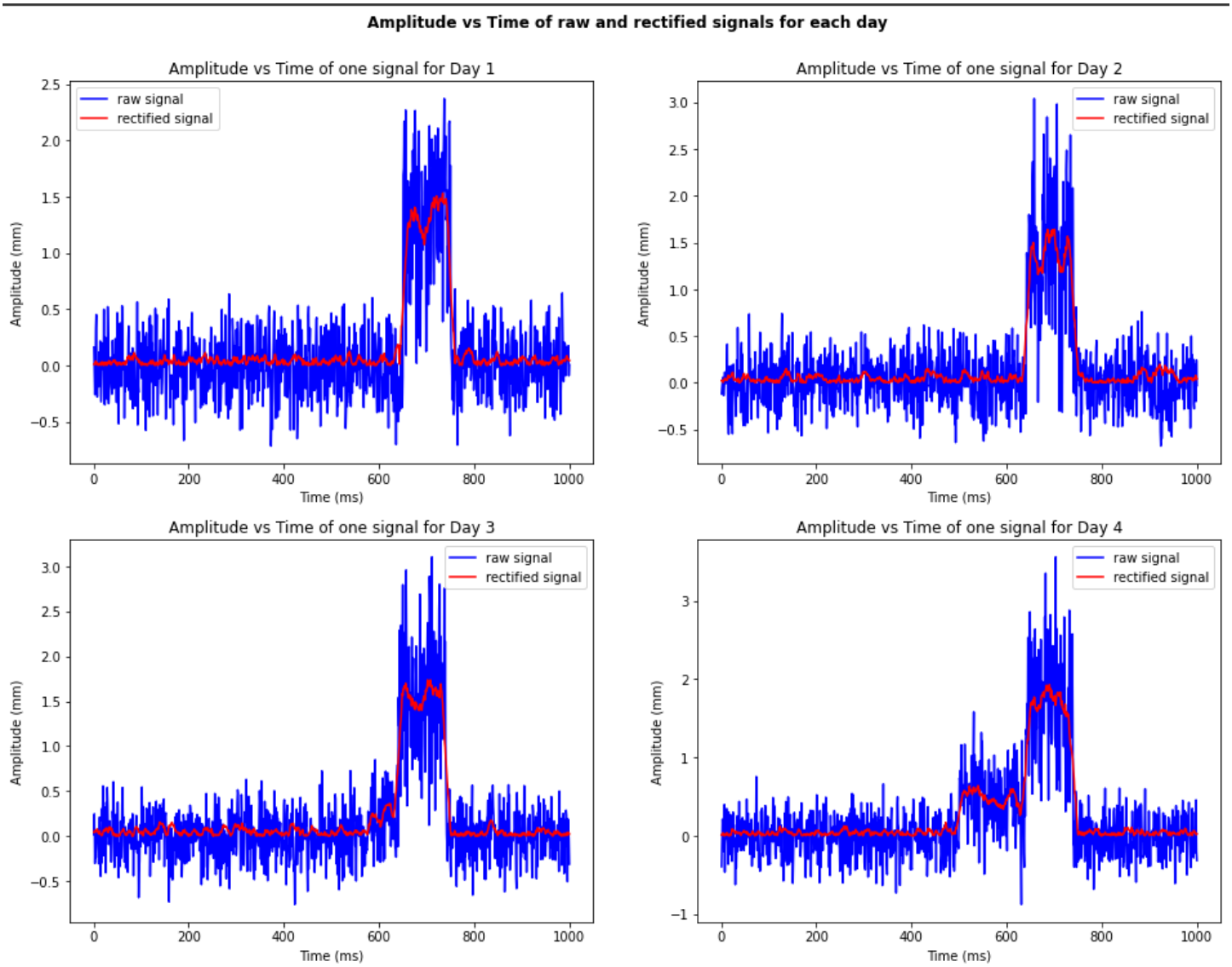
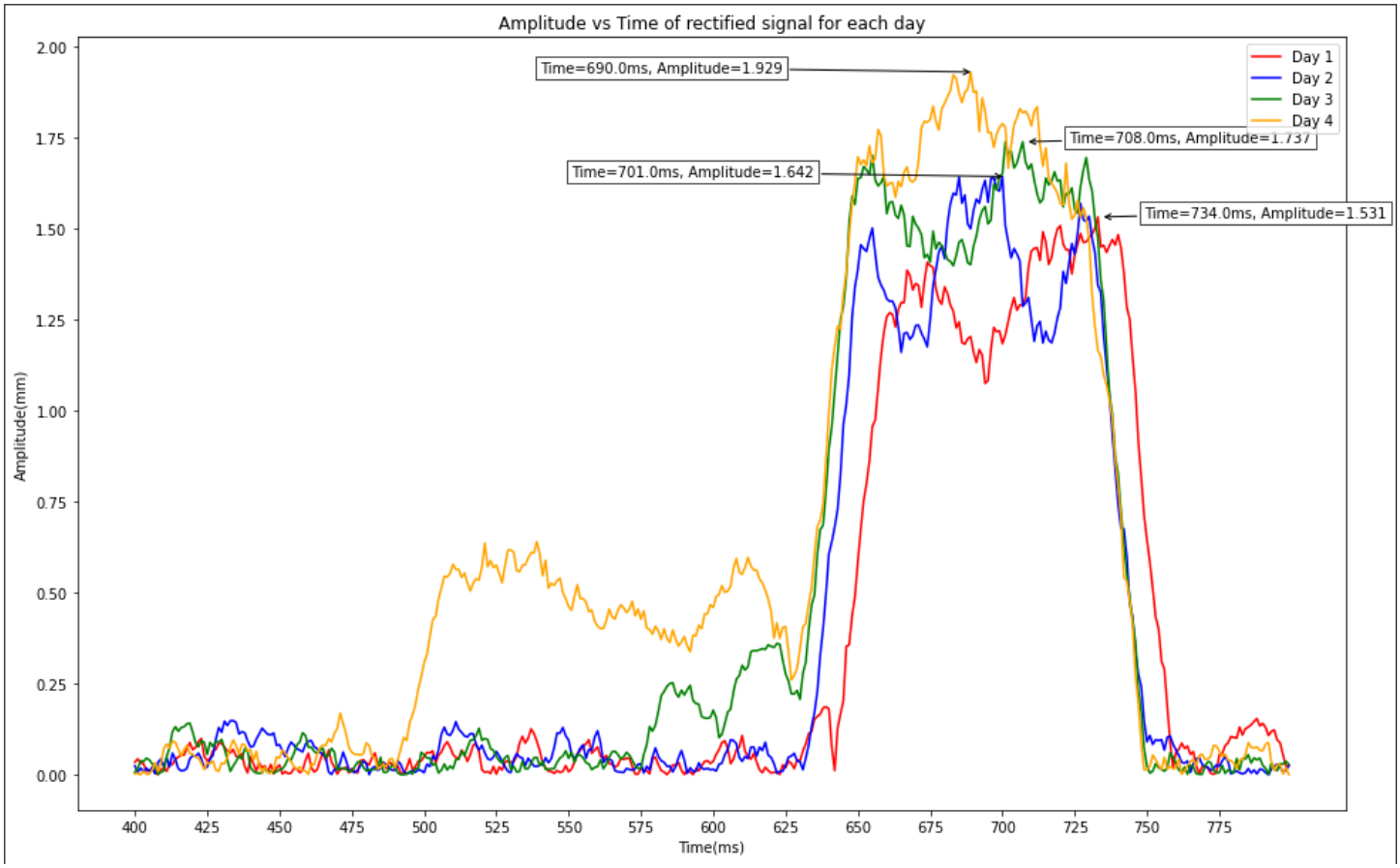


Figure shows Amplitude vs Time(ms) of one signal across a time period of 1000 ms for 4 days.
The amplitude is averaged over 5 trials for each day (blue curve)
A moving average filter is passed through it and then rectified (red curve)

- A further analysis has been performed to provide a clearer understanding.

Step - 5 (Further Analysis)

```
plt.figure(figsize=(16,10))

time = np.arange(1,1001)

xmax1 = time[np.argmax(rectified_signal1)]
ymax1 = rectified_signal1.max()
plt.plot(np.arange(400, 800), rectified_signal1[400:800], color = 'red', label = 'Day 1')
text = "Time={:.1f}ms, Amplitude={:.3f}".format(xmax1, ymax1)
plt.annotate(text, xy = (xmax1, ymax1), xytext = (xmax1 + 15, ymax1), arrowprops=dict(arrowstyle="->"), bbox = dict(boxstyle="square,pad=0.3", fc="w", ec="k", lw=0.72))

xmax2 = time[np.argmax(rectified_signal2)]
ymax2 = rectified_signal2.max()
plt.plot(np.arange(400, 800), rectified_signal2[400:800], color = 'blue', label = 'Day 2')
text= "Time={:.1f}ms, Amplitude={:.3f}".format(xmax2, ymax2)
plt.annotate(text, xy = (xmax2, ymax2), xytext = (xmax2 - 150, ymax2), arrowprops=dict(arrowstyle="->"), bbox = dict(boxstyle="square,pad=0.3", fc="w", ec="k", lw=0.72))

xmax3 = time[np.argmax(rectified_signal3)]
ymax3 = rectified_signal3.max()
plt.plot(np.arange(400, 800), rectified_signal3[400:800], color = 'green', label = 'Day 3')
text = "Time={:.1f}ms, Amplitude={:.3f}".format(xmax3, ymax3)
plt.annotate(text, xy = (xmax3, ymax3), xytext = (xmax3 + 15, ymax3), arrowprops=dict(arrowstyle="->"), bbox = dict(boxstyle="square,pad=0.3", fc="w", ec="k", lw=0.72))

xmax4 = time[np.argmax(rectified_signal4)]
ymax4 = rectified_signal4.max()
plt.plot(np.arange(400, 800), rectified_signal4[400:800], color = 'orange', label = 'Day 4')
text = "Time={:.1f}ms, Amplitude={:.3f}".format(xmax4, ymax4)
plt.annotate(text, xy = (xmax4, ymax4), xytext = (xmax4 - 150, ymax4), arrowprops=dict(arrowstyle="->"), bbox = dict(boxstyle="square,pad=0.3", fc="w", ec="k", lw=0.72))

plt.xlabel("Time(ms)")
plt.ylabel('Amplitude')
plt.xticks(np.arange(400,800,step=25))
plt.title("Amplitude vs Time of one signal for each day")
plt.legend()
plt.show()
```



Caption: Figure shows Amplitude vs Time for the rectified signal for each of the 4 days. A shorter time period is displayed here (between 400 - 800 ms). Peaks in each waveform are clearly highlighted which display the maxiumum amplitude and the time at which it is achieved.

The learning mechanism evident here is Classical Conditioning. The air puff delivered to the eyes is called the Unconditioned Stimulus. The eye blink in response to the air puff is called the Unconditioned Response. The tone delivered initially through the headphones is called the Neural Stimulus. The Conditioned Stimulus is the Neural Stimulus(tone through headphones) paired with the Unconditioned Stimulus(air puff). The Conditioned Response is a learned response to a previously neutral stimulus. In this case, it is the eye blink just before the air puff that the tone through the headphones predicts.

Through this experiment, we are trying to analyze the temporal aspect of classical conditioning. After observing the plots, we can concur that there is delay classical conditioning present. The Conditioned stimulus onset is at time = 0 ms (beginning of the trial) and continues until 650 ms, whereas the Unconditioned stimulus(air-puff) onset is at time = 600 ms and continues for the next 50 ms. They also both co-terminate at 650 ms.

Figure 2 is used here for values. On Day 1, the amplitude increases at approximately 640 ms. So, the unconditioned response (eye-blink) is caused by the unconditioned stimulus (air-puff) which starts at around 600 ms. The amplitude reaches its peak at 734 ms to a value of 1.531 mm. On Day 2, the amplitude starts increasing earlier than before at around 630 ms. It also reaches it peak earlier at around 701 ms. This shows that the participant is conditioning to the stimulus (it is learning). On Day 3, there is a great difference as the amplitude increases at around 575 ms. This means that the participant's mind predicted the unconditioned stimulus (air puff) at 600 ms, so it exhibits a weak conditioned response to the tone before the air puff. The amplitude reaches maxima as approximately same time (708 ms) as Day 2. The amplitude, however, reaches a higher value than Day 2. On Day 4, the participant has the strongest conditioned response as the amplitude shows an increase at 485 ms, which is way earlier than the onset of unconditioned stimulus (air puff) at 600 ms. It also achieves its maxima earlier than the 3 days at 690 ms.

After these observations, we conclude that initially the participant responds to the unconditioned stimulus (air puff) in the form of an eye blink. As time passes, the participant learns, and the unconditioned response takes place due to conditioned stimulus before the air puff. The participant predicts and assimilates itself to the arrival of the air puff. We also notice that predictions become stronger as time passes which is why the maximum value of the amplitude increases every day.

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 colour) and overlay of the learning curve (blue colour) 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.

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?

[Answer]

[8+2 points]

i)

Code and Explanation

- Load the data

Load the data

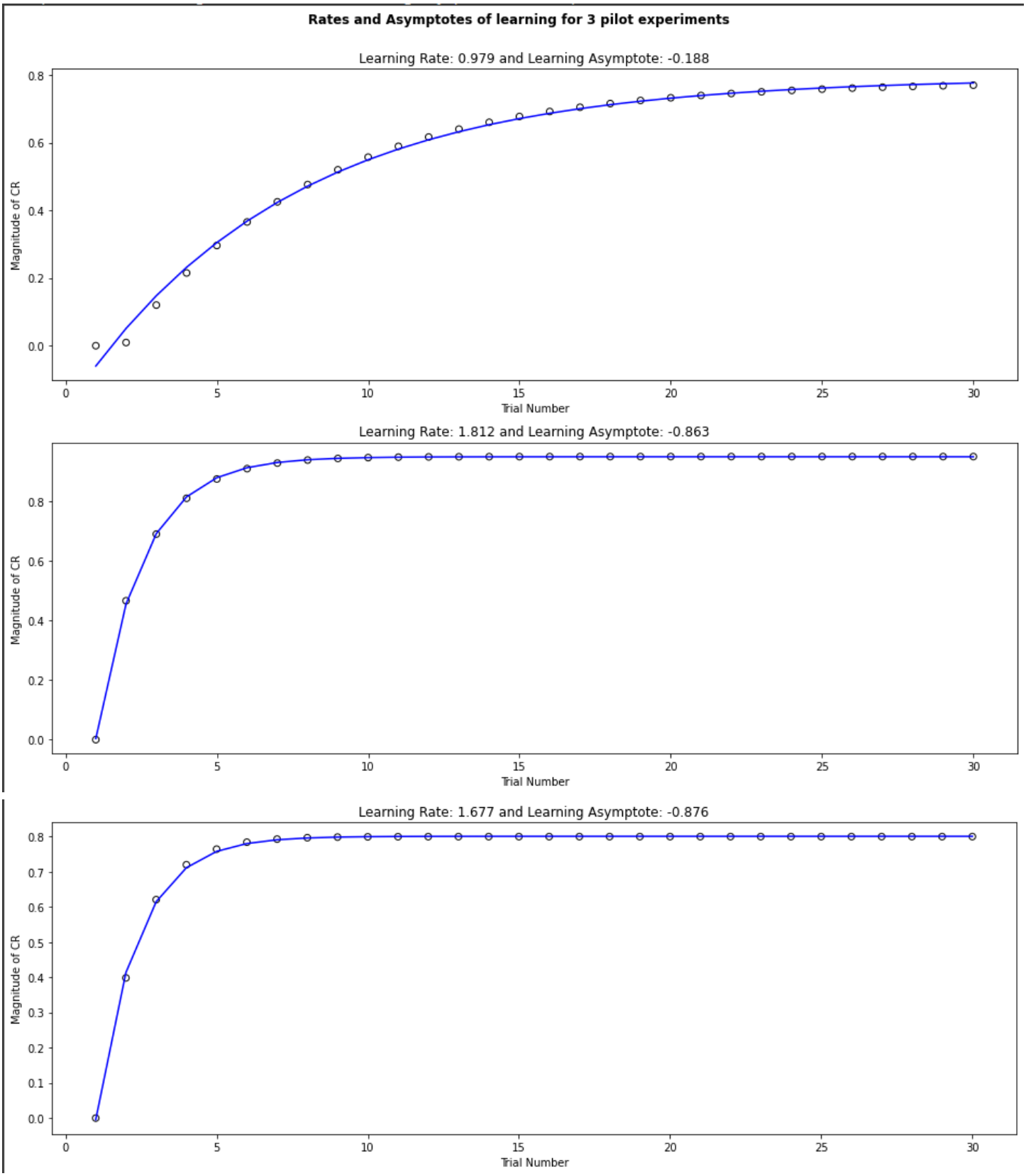
```
[36] exp = pd.ExcelFile('/content/drive/MyDrive/Data-Assignment2B.xlsx')
     df = pd.read_excel(exp, index_col = 0, header = None)
```

- I used unconstrained nonlinear optimization to find the optimal parameters of the negatively accelerated learning curve. Our objective function -> y = a*(1 - exp(-b*x)) + c

Objective Function

```
[63] # Equation for negatively accelerated learning curve
     def objective(x, a, b, c):
         y = a*(1 - np.exp(-b*x)) + c
         return y
```

- Curve Fitting is performed as per the above function and subsequent plots are shown below



Caption: The above figure shows 3 different plots for 3 different pilot experiments showing number of trials vs magnitude/value of the conditioned response. Curve fitting has been implemented on the above data points and respective Learning rates and asymptotes are calculated and shown for each plot.

- I am using sum of squared errors as a metric to measure the goodness of curve fit. Their calculation is already present in the code above, values are shown below

Sum of Squared Errors

```
print("Sum of Squared Error in Pilot Experiment 1: " + str(sse1))
print("Sum of Squared Error in Pilot Experiment 2: " + str(sse2))
print("Sum of Squared Error in Pilot Experiment 3: " + str(sse3))
```

```
Sum of Squared Error in Pilot Experiment 1: 0.007118844297144633
Sum of Squared Error in Pilot Experiment 2: 0.00018841919923305846
Sum of Squared Error in Pilot Experiment 3: 0.0004412249891243915
```

The closer the value of sum of squared error is to 0, the better the curve fit. Since all values are extremely small and close to zero, each of our learning curves are of exceptional quality. As per above values, Curve for Experiment 2 fits best as sum of squared error is closest to zero.

ii)
Based on our analysis of above data and curve fits, we note that the intensity of unconditioned stimuli of Experiment 1 is highest of the experiments. This is because the Learning Asymptote calculated for experiment 1 is highest in comparison. Experiment 2 and 3 have similar value of Learning Asymptote so their intensities are similar(although Experiment 2 has marginally higher intensity).

Complete Code of Assignment

```
# -*- coding: utf-8 -*-
"""Untitled6.ipynb

Automatically generated by Colaboratory.

Importing the Libraries
"""

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy.ndimage import uniform_filter1d
from scipy.optimize import curve_fit

"""Load data"""

xls = pd.ExcelFile('/content/drive/MyDrive/Data-Assignment2A.xlsx.xlsx')
df_day1 = pd.read_excel(xls, 'Day1', index_col = 0)
df_day2 = pd.read_excel(xls, 'Day2', index_col = 0)
df_day3 = pd.read_excel(xls, 'Day3', index_col = 0)
df_day4 = pd.read_excel(xls, 'Day4', index_col = 0)
```

```
"""Step - 1"""

# Taking the average of data across all trials per day for each time point to get one averaged signal per day.
mean1 = df_day1.mean(axis = 0)
mean2 = df_day2.mean(axis = 0)
mean3 = df_day3.mean(axis = 0)
mean4 = df_day4.mean(axis = 0)

mean1

"""Step - 2"""

# running a 'moving average filter' across the averaged signal with a window width of 20 ms to get a filtered signal
filtered_signal1 = uniform_filter1d(mean1, size = 20)
filtered_signal2 = uniform_filter1d(mean2, size = 20)
filtered_signal3 = uniform_filter1d(mean3, size = 20)
filtered_signal4 = uniform_filter1d(mean4, size = 20)

"""Step - 3"""

# full wave rectification of the above moving average filtered signal
rectified_signal1 = np.abs(filtered_signal1)
rectified_signal2 = np.abs(filtered_signal2)
rectified_signal3 = np.abs(filtered_signal3)
rectified_signal4 = np.abs(filtered_signal4)

"""Step - 4"""

# Plot the amplitude vs time of the raw signal (as blue curve)
fig, ((plot1, plot2), (plot3, plot4)) = plt.subplots(2, 2, figsize = (16, 12))
fig.suptitle('Amplitude vs Time of raw and rectified signals for each day', y = 0.94, fontweight = 'bold')

plot1.plot(np.arange(1, 1001), mean1, color = 'blue', label = 'raw signal')
plot1.plot(np.arange(1, 1001), rectified_signal1, color = 'red', label = 'rectified signal')
plot1.set_xlabel('Time (ms)')
plot1.set_ylabel('Amplitude (mm)')
plot1.set_title('Amplitude vs Time of one signal for Day 1')
plot1.legend()
```

```
plot2.plot(np.arange(1, 1001), mean2, color = 'blue', label = 'raw signal')
plot2.plot(np.arange(1, 1001), rectified_signal2, color = 'red', label = 'rectified
signal')
plot2.set_xlabel('Time (ms)')
plot2.set_ylabel('Amplitude (mm)')
plot2.set_title('Amplitude vs Time of one signal for Day 2')
plot2.legend()

plot3.plot(np.arange(1, 1001), mean3, color = 'blue', label = 'raw signal')
plot3.plot(np.arange(1, 1001), rectified_signal3, color = 'red', label = 'rectified
signal')
plot3.set_xlabel('Time (ms)')
plot3.set_ylabel('Amplitude (mm)')
plot3.set_title('Amplitude vs Time of one signal for Day 3')
plot3.legend()

plot4.plot(np.arange(1, 1001), mean4, color = 'blue', label = 'raw signal')
plot4.plot(np.arange(1, 1001), rectified_signal4, color = 'red', label = 'rectified
signal')
plot4.set_xlabel('Time (ms)')
plot4.set_ylabel('Amplitude (mm)')
plot4.set_title('Amplitude vs Time of one signal for Day 4')
plot4.legend()

caption = """Figure shows Amplitude vs Time(ms) of one signal across a time period of
1000 ms for 4 days.
            The amplitude is averaged over 5 trials for each day (blue curve)
            A moving average filter is passed through it and then rectified (red
curve)"""

fig.text(0.5, 0.03, caption, ha = 'center')

"""Step - 5 (Further Analysis)"""

plt.figure(figsize=(16,10))

time = np.arange(1,1001)

xmax1 = time[np.argmax(rectified_signal1)]
ymax1 = rectified_signal1.max()
plt.plot(np.arange(400, 800), rectified_signal1[400:800], color = 'red', label = 'Day
1')
text = "Time={:.1f}ms, Amplitude={:.3f}".format(xmax1, ymax1)
```

```
plt.annotate(text, xy = (xmax1, ymax1), xytext = (xmax1 + 15, ymax1),
arrowprops=dict(arrowstyle="->"), bbox = dict(boxstyle="square,pad=0.3", fc="w",
ec="k", lw=0.72))

xmax2 = time[np.argmax(rectified_signal2)]
ymax2 = rectified_signal2.max()
plt.plot(np.arange(400, 800), rectified_signal2[400:800], color = 'blue', label =
'Day 2')
text= "Time={:.1f}ms, Amplitude={:.3f}".format(xmax2, ymax2)
plt.annotate(text, xy = (xmax2, ymax2), xytext = (xmax2 - 150, ymax2),
arrowprops=dict(arrowstyle="->"), bbox = dict(boxstyle="square,pad=0.3", fc="w",
ec="k", lw=0.72))

xmax3 = time[np.argmax(rectified_signal3)]
ymax3 = rectified_signal3.max()
plt.plot(np.arange(400, 800), rectified_signal3[400:800], color = 'green', label =
'Day 3')
text = "Time={:.1f}ms, Amplitude={:.3f}".format(xmax3, ymax3)
plt.annotate(text, xy = (xmax3, ymax3), xytext = (xmax3 + 15, ymax3),
arrowprops=dict(arrowstyle="->"), bbox = dict(boxstyle="square,pad=0.3", fc="w",
ec="k", lw=0.72))

xmax4 = time[np.argmax(rectified_signal4)]
ymax4 = rectified_signal4.max()
plt.plot(np.arange(400, 800), rectified_signal4[400:800], color = 'orange', label =
'Day 4')
text = "Time={:.1f}ms, Amplitude={:.3f}".format(xmax4, ymax4)
plt.annotate(text, xy = (xmax4, ymax4), xytext = (xmax4 - 150, ymax4),
arrowprops=dict(arrowstyle="->"), bbox = dict(boxstyle="square,pad=0.3", fc="w",
ec="k", lw=0.72))

plt.xlabel("Time(ms)")
plt.ylabel('Amplitude(mm)')
plt.xticks(np.arange(400,800,step=25))
plt.title("Amplitude vs Time of rectified signal for each day")
plt.legend()
plt.show()

"""# Q2 - B

Load the data
"""
```

```
exp = pd.ExcelFile('/content/drive/MyDrive/Data-Assignment2B.xlsx')
df = pd.read_excel(exp, index_col = 0, header = None)

"""Objective Function"""

# Equation for negatively accelerated learning curve
def objective(x, a, b, c):
    y = a*(1 - np.exp(-b*x)) + c
    return y

figure, (p1,p2,p3) = plt.subplots(3,1,figsize = (16,18))
figure.suptitle('Rates and Asymptotes of learning for 3 pilot experiments', y = 0.92,
fontweight = 'bold')

p1.scatter(np.arange(1,31), df.loc['Pilot Exp1'], facecolors='none', edgecolors =
'black')
p1.set_xlabel('Trial Number')
p1.set_ylabel('Magnitude of CR')
popt, _ = curve_fit(objective, np.arange(1,31), df.loc['Pilot Exp1'])
a, b, c = popt
p1.plot(np.arange(1,31), objective(np.arange(1,31), a, b, c), color = 'blue')
p1.set_title('Learning Rate: {:.3f} and Learning Asymptote: {:.3f}'.format(a, c))
# goodness of fit
sse1 = np.sum((df.loc['Pilot Exp1'] - objective(np.arange(1,31), a, b, c))**2)

p2.scatter(np.arange(1,31), df.loc['Pilot Exp2'], facecolors='none', edgecolors =
'black')
p2.set_xlabel('Trial Number')
p2.set_ylabel('Magnitude of CR')
popt, _ = curve_fit(objective, np.arange(1,31), df.loc['Pilot Exp2'])
a, b, c = popt
p2.plot(np.arange(1,31), objective(np.arange(1,31), a, b, c), color = 'blue')
p2.set_title('Learning Rate: {:.3f} and Learning Asymptote: {:.3f}'.format(a, c))
# goodness of fit
sse2 = np.sum((df.loc['Pilot Exp2'] - objective(np.arange(1,31), a, b, c))**2)

p3.scatter(np.arange(1,31), df.loc['Pilot Exp3'], facecolors='none', edgecolors =
'black')
p3.set_xlabel('Trial Number')
p3.set_ylabel('Magnitude of CR')
popt, _ = curve_fit(objective, np.arange(1,31), df.loc['Pilot Exp3'])
a, b, c = popt
p3.plot(np.arange(1,31), objective(np.arange(1,31), a, b, c), color = 'blue')
```

```
p3.set_title('Learning Rate: {:.3f} and Learning Asymptote: {:.3f}'.format(a, c))
# goodness of fit
sse3 = np.sum((df.loc['Pilot Exp3'] - objective(np.arange(1,31), a, b, c))**2)

"""Sum of Squared Errors"""

print("Sum of Squared Error in Pilot Experiment 1: " + str(sse1))
print("Sum of Squared Error in Pilot Experiment 2: " + str(sse2))
print("Sum of Squared Error in Pilot Experiment 3: " + str(sse3))
```