14744 Homework (10 pts)

Training Time Series Models

Due: Wednesday March 15

Submission Instructions

- Save a copy of this notebook in your Google Drive by clicking File->Save a copy in Drive
- Use the Python 3 programming language to complete the programming exercises in the provided code cells
- Make sure that all your code and code output appear correctly
- Submit **both** of the following copies of your notebook on Canvas
    - .pdf version (by printing your notebook to pdf)
    - .ipynb version (by clicking File->Download .ipynb)

Starter code

To get started run the cell below by clicking its 'play' icon. If you want to run this cell a second time, you may need to reset the runtime by clicking Runtime->Factory reset runtime.

```
## DO NOT EDIT THIS CELL

# Import libraries
import numpy as np
import matplotlib.pyplot as plt
import os

# Download and unzip accelerometer trace data
!mkdir ./data/
!wget https://www.andrew.cmu.edu/user/dvaroday/14744/data.zip
!mv data.zip ./data
!unzip ./data/data.zip -d ./data
!rm ./data/data.zip

# Initialize paths and list of raw trace filenames
ground_truth_path = '/content/data/ground_truth/'

path = '/content/data/raw_traces/'
filenames = sorted(os.listdir(path))

# Define functions presented in lecture
def brush_indicator(filename, alpha, threshold):
  acceleration = np.genfromtxt(path+filename).astype(float)
  jerk = np.zeros(acceleration.shape)
  jerk[1:,:] = acceleration[1:,:] - acceleration[:-1,:]
  jerk_magnitude = np.sqrt(np.sum(jerk**2, axis=1))
  smoothed = np.zeros(jerk_magnitude.shape)
  smoothed[0] = jerk_magnitude[0]
  for i in range(1, len(smoothed)):
    smoothed[i] = alpha * jerk_magnitude[i] + (1-alpha) * smoothed[i-1]
  indicator = smoothed > threshold
  return indicator

def error_cost_function(filename, alpha, threshold):
  indicator = brush_indicator(filename, alpha, threshold)
  ground_truth = np.genfromtxt(ground_truth_path + 'Truth_' + filename)
  cost = np.sum(indicator != ground_truth)/len(indicator)
  return cost
```

```
--2023-03-13 07:26:35--  https://www.andrew.cmu.edu/user/dvaroday/14744/data.zip
Resolving www.andrew.cmu.edu (www.andrew.cmu.edu)... 128.2.42.53
Connecting to www.andrew.cmu.edu (www.andrew.cmu.edu)|128.2.42.53|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 94022 (92K) [application/zip]
Saving to: ‘data.zip’

data.zip            100%[===================>]  91.82K  --.-KB/s    in 0.1s

2023-03-13 07:26:35 (928 KB/s) - ‘data.zip’ saved [94022/94022]

Archive:  ./data/data.zip
  inflating: ./data/ground_truth/Truth_Accelerometer-2011-04-11-13-28-18-brush_teeth-f1.txt
  inflating: ./data/ground_truth/Truth_Accelerometer-2011-04-11-13-29-54-brush_teeth-f1.txt
  inflating: ./data/ground_truth/Truth_Accelerometer-2011-05-30-08-35-11-brush_teeth-f1.txt
  inflating: ./data/ground_truth/Truth_Accelerometer-2011-05-30-09-36-50-brush_teeth-f1.txt
  inflating: ./data/ground_truth/Truth_Accelerometer-2011-05-30-10-34-16-brush_teeth-m1.txt
  inflating: ./data/ground_truth/Truth_Accelerometer-2011-05-30-21-10-57-brush_teeth-f1.txt
  inflating: ./data/ground_truth/Truth_Accelerometer-2011-05-30-21-55-04-brush_teeth-m2.txt
  inflating: ./data/ground_truth/Truth_Accelerometer-2011-05-31-15-16-47-brush_teeth-f1.txt
  inflating: ./data/ground_truth/Truth_Accelerometer-2011-06-02-10-42-22-brush_teeth-f1.txt
  inflating: ./data/ground_truth/Truth_Accelerometer-2011-06-02-10-45-50-brush_teeth-f1.txt
  inflating: ./data/ground_truth/Truth_Accelerometer-2011-06-06-10-45-27-brush_teeth-f1.txt
  inflating: ./data/ground_truth/Truth_Accelerometer-2011-06-06-10-48-05-brush_teeth-f1.txt
  inflating: ./data/raw_traces/Accelerometer-2011-04-11-13-28-18-brush_teeth-f1.txt
```

```
inflating: ./data/raw_traces/Accelerometer-2011-04-11-13-29-54-brush_teeth-f1.txt
inflating: ./data/raw_traces/Accelerometer-2011-05-30-08-35-11-brush_teeth-f1.txt
inflating: ./data/raw_traces/Accelerometer-2011-05-30-09-36-50-brush_teeth-f1.txt
inflating: ./data/raw_traces/Accelerometer-2011-05-30-10-34-16-brush_teeth-m1.txt
inflating: ./data/raw_traces/Accelerometer-2011-05-30-21-10-57-brush_teeth-f1.txt
inflating: ./data/raw_traces/Accelerometer-2011-05-30-21-55-04-brush_teeth-m2.txt
inflating: ./data/raw_traces/Accelerometer-2011-05-31-15-16-47-brush_teeth-f1.txt
inflating: ./data/raw_traces/Accelerometer-2011-06-02-10-42-22-brush_teeth-f1.txt
inflating: ./data/raw_traces/Accelerometer-2011-06-02-10-45-50-brush_teeth-f1.txt
inflating: ./data/raw_traces/Accelerometer-2011-06-06-10-45-27-brush_teeth-f1.txt
inflating: ./data/raw_traces/Accelerometer-2011-06-06-10-48-05-brush_teeth-f1.txt
```

Problem 1 (3 pts)

In the cell below write a leave-one-out cross-validation procedure for the time series model described in `brush_indicator()`. Each iteration of training should optimize the model over all combinations of parameters $\alpha \in \{0.1, 0.15, 0.2\}$ and $\text{threshold} \in \{7, 8, 9\}$. Your code should print out an average testing error cost of `0.0140` as reported in lecture.

```
## EDIT THE CODE IN THIS CELL

alphas = [0.1, 0.15, 0.2]
thresholds = [7, 8, 9]

avg_testing_error_cost = 0
total_testing_error_cost = 0
for x in range(len(filenames)):
  best_error = 1
  best_alpha = 0
  best_threshold = 0
  for alpha in alphas:
    for threshold in thresholds:
      total = 0
      for i in range (len(filenames)):
        if not i == x:
          total += error_cost_function(filenames[i], alpha, threshold)
      error = total / (len(filenames) - 1)
      if error < best_error:
        best_alpha = alpha
        best_threshold = threshold
      best_error = min(best_error, error)
  test_error = error_cost_function(filenames[x], best_alpha, best_threshold)
  total_testing_error_cost += test_error

avg_testing_error_cost = total_testing_error_cost / len(filenames)
print(avg_testing_error_cost)
```

```
    0.01403845008161896
```

Problem 2 (2 pts)

Now that you have validated that the `brush_indicator()` model is sound, train the model using all the traces. As before you should optimize the model over all combinations of parameters $\alpha \in \{0.1, 0.15, 0.2\}$ and $\text{threshold} \in \{7, 8, 9\}$. Your code should print out the optimal values of α and threshold

```
## EDIT THE CODE IN THIS CELL

alphas = [0.1, 0.15, 0.2]
thresholds = [7, 8, 9]

best_error = 1
best_alpha = 0
best_threshold = 0
for alpha in alphas:
  for threshold in thresholds:
    total = 0
    for i in range (len(filenames)):
      total += error_cost_function(filenames[i], alpha, threshold)
    error = total / len(filenames)
    if error < best_error:
      best_alpha = alpha
      best_threshold = threshold
      best_error = error

print('alpha =', best_alpha)
print('threshold =', best_threshold)
```

```
    alpha = 0.15
    threshold = 8
```

Problem 3 (3 pts)

Complete the implementation of the `brush_indicator2()` model, so that it is identical to `brush_indicator()` except that it uses two thresholds as described in lecture.

Then write a leave-one-out cross-validation procedure for `brush_indicator2()`. Fix α to the value you determined in Problem 2. Each iteration of training should optimize the model over combinations of parameters $\text{threshold\_lo}, \text{threshold\_hi} \in \{7, 8, 9\}$. Your code should print out an average testing error cost of `0.0113` as reported in lecture.

```
## EDIT THE CODE IN THIS CELL

# Complete implementation of brush_indicator2() so that it uses two thresholds
def brush_indicator2(filename, alpha, threshold_lo, threshold_hi):
  acceleration = np.genfromtxt(path+filename).astype(float)
  jerk = np.zeros(acceleration.shape)
  jerk[1:,:] = acceleration[1:,:] - acceleration[:-1,:]
  jerk_magnitude = np.sqrt(np.sum(jerk**2, axis=1))
  smoothed = np.zeros(jerk_magnitude.shape)
  smoothed[0] = jerk_magnitude[0]
  for i in range(1, len(smoothed)):
    smoothed[i] = alpha * jerk_magnitude[i] + (1-alpha) * smoothed[i-1]
  indicator = np.zeros(smoothed.shape)

  direction = False
  for i in range(len(indicator)):
    # continue not brushing
    if (not direction) and (smoothed[i] < threshold_hi):
        indicator[i] = 0
    # change from not brushing to brushing
    elif (not direction) and (smoothed[i] >= threshold_hi):
        direction = True
        indicator[i] = 1
    # continue brushing
    elif direction and (smoothed[i] > threshold_lo):
        indicator[i] = 1
    # change from brushing to not brushing
    else:
        direction = False
        indicator[i] = 0

  return indicator

def error_cost_function2(filename, alpha, threshold_lo, threshold_hi):
  indicator = brush_indicator2(filename, alpha, threshold_lo, threshold_hi)
  ground_truth = np.genfromtxt(ground_truth_path + 'Truth_' + filename)
  cost = np.sum(indicator != ground_truth)/len(indicator)
  return cost


# Write a leave-one-out cross-validation procedure for brush_indicator()
thresholds = [7, 8, 9]

avg_testing_error_cost = 0
total_testing_error_cost = 0
for x in range(len(filenames)):
  best_error = 1
  best_threshold_low = 0
  best_threshold_high = 0
  for threshold_low in thresholds:
    for threshold_hi in thresholds:
      total = 0
      for i in range (len(filenames)):
        if not i == x:
          total += error_cost_function2(filenames[i], 0.15, threshold_low, threshold_hi)
      error = total / (len(filenames) - 1)
      if error < best_error:
        best_threshold_low = threshold_low
        best_threshold_high = threshold_hi
      best_error = min(best_error, error)
  test_error = error_cost_function2(filenames[x], 0.15, best_threshold_low, best_threshold_high)
  total_testing_error_cost += test_error

avg_testing_error_cost = total_testing_error_cost / len(filenames)

print(avg_testing_error_cost)
```

```
    0.011318174889272446
```

Problem 4 (1 pt)

Train the `brush_indicator2()` model using all the traces. As before fix α to the value you determined in Problem 2, and optimize the model over combinations of parameters $\text{threshold\_lo}, \text{threshold\_hi} \in \{7, 8, 9\}$. Your code should print out the optimal values of threshold_lo and threshold_hi.

```
## EDIT THE CODE IN THIS CELL

thresholds = [7, 8, 9]

best_error = 1
best_threshold_high = 0
best_threshold_low = 0
for threshold_high in thresholds:
  for threshold_low in thresholds:
    total = 0
    for i in range (len(filenames)):
      total += error_cost_function2(filenames[i], 0.15, threshold_low, threshold_high)
    error = total / len(filenames)
    if error < best_error:
      best_threshold_low = threshold_low
      best_threshold_high = threshold_high
      best_error = error

print('threshold_lo =', best_threshold_low)
print('threshold_hi =', best_threshold_high)
```

```
threshold_lo = 7
threshold_hi = 9
```

Problem 5 (1 pt)

Write code that prints out the average time (in seconds) that the volunteer `f1` spends actively brushing her teeth according to:

- the ground truth
- the `brush_indicator()` model with parameters you found in Problem 2
- the `brush_indicator2()` model with parameters you found in Problem 4

```
## EDIT THE CODE IN THIS CELL

avg_time_f1_ground_truth = 0
avg_time_f1_brush_indicator = 0
avg_time_f1_brush_indicator2 = 0

total_time_f1_ground_truth = 0
total_time_f1_brush_indicator = 0
total_time_f1_brush_indicator2 = 0

rate = 32

count = 0
for filename in filenames:
  if filename[-6:] == "f1.txt":
    count += 1
    ground_truth_indicator = np.genfromtxt(ground_truth_path + 'Truth_' + filename)
    for truth in ground_truth_indicator:
      if truth == 1:
        total_time_f1_ground_truth += 1
    brush = brush_indicator(filename, 0.15, 8)
    for x in brush:
      if x == 1:
        total_time_f1_brush_indicator += 1
    brush2 = brush_indicator2(filename, 0.15, 7, 9)
    for x in brush2:
      if x == 1:
        total_time_f1_brush_indicator2 += 1

avg_time_f1_ground_truth = (total_time_f1_ground_truth / count) / 32
avg_time_f1_brush_indicator = (total_time_f1_brush_indicator / count) / 32
avg_time_f1_brush_indicator2 = (total_time_f1_brush_indicator2 / count) / 32
print('Average brushing time for f1 (ground truth) =',
      avg_time_f1_ground_truth, 'seconds')
print('Average brushing time for f1 (brush_indicator) =',
      avg_time_f1_brush_indicator, 'seconds')
print('Average brushing time for f1 (brush_indicator2) =',
      avg_time_f1_brush_indicator2, 'seconds')
```

```
Average brushing time for f1 (ground truth) = 44.465625 seconds
Average brushing time for f1 (brush_indicator) = 44.271875 seconds
Average brushing time for f1 (brush_indicator2) = 44.54375 seconds
```