Activity # 9

Classification Model (NN, CNN)




Agenda

9.1 Data Preparation

9.2 NN Model Training and Testing

9.3 CNN Model Training and Testing

9.1 Data Preparation

- **Data Exploration and Cleaning / Transform / Feature Selection /Train-Test-Split**

LIBRARIES

1. import numpy as np
2. import pandas as pd
3. import matplotlib.pyplot as plt
4. from sklearn.preprocessing import StandardScaler
5. from sklearn.model_selection import train_test_split, cross_val_score, StratifiedKFold
6. from sklearn.svm import SVC
7. from sklearn import metrics
8. from sklearn.model_selection import GridSearchCV, RandomizedSearchCV
9. from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
10. import glob
11. from scipy import stats
12. import datetime as dt

9.1.1 -> 8.1 (a) Load and Prepare Data

1

- # Load data from csv 3 files
- # acceleration.txt, heartrate.txt, labeled_sleep.txt
  - ACC = read_csv(acceleration.txt, sep = ' ',names=['timedelta', 'accX', 'accY', 'accZ'])
  - HeartR = read_csv(heartrate.txt, sep = ',',names=['timedelta', 'heartrate'])
  - SleepL = read_csv(labeled_sleep.txt, sep = ' ',names=['timedelta', 'sleep'])

2

- # Check 'timedelta' max(), min() of ACC, HeartR, SleepL (ช่วงเวลาที่มีข้อมูลใกล้กัน)
  - Ex
    - ACC_max_date = ACC['timedelta'].max()
    - ACC_min_date = ACC['timedelta'].min()
    - หา start_timedelta, end_timedelta

```
ACC start:  -124489.16105 ACC end:  17643.046417
HeartR start:  -355241.73971 HeartR end:  34491.1535499
SleepL start:  0 SleepL end:  28530
```

3

- # select only intersected timedelta (ACC, HeartR, SleepL) (ช่วงเวลาที่มีข้อมูลใกล้กัน)
- Ex
  - ACC_new = ACC[(ACC['timedelta'] > start_timedelta) &(df_acc['timedelta'] < end_timedelta)]

9.1.1 -> 8.1 (b) Load and Prepare Data (ACC)

4

- # ------------ Rounding ACC (Rounding to 1 sec) --------------------------------
  - # Convert to datetime and round to second,
    - ACC_new['timedelta'] = pd.DataFrame(pd.to_timedelta(ACC_new['timedelta'], timedelta_unit).round('1s'))

5

- # Average rounding duplicated time
- # df_acc_X = ACC_new.groupby('timedelta')['accX'].mean().reset_index()
- df_acc_Y = ACC_new.groupby('timedelta')['accY'].mean().reset_index()
- df_acc_Z = ACC_new.groupby('timedelta')['accZ'].mean().reset_index()

6

- # acc_X, acc_Y, acc_Z
- Ex
  - pd.concat([df_acc_X, df_acc_Y, df_acc_Z], axis=1)

Before / After

convert datetime and round and average to 1s

```
------ Before convert datetime and round and average to 1s ------------
       timedelta      accX      accY      accZ
0 -124489.161050  0.017487 -0.586700 -0.805771
1 -124489.116395  0.018982 -0.589676 -0.809158
2 -124489.115548  0.020966 -0.580887 -0.815048
3 -124489.114691  0.019485 -0.580872 -0.813583
4 -124489.097700  0.016998 -0.587204 -0.806259
------ After convert datetime and round and average to 1s ------------
        timedelta      accX      accY      accZ
0 0 days 00:00:00 -0.243203  0.895372  0.367591
1 0 days 00:00:01 -0.240757  0.873826  0.415446
2 0 days 00:00:02 -0.244620  0.883943  0.387026
3 0 days 00:00:03 -0.248036  0.902427  0.347812
4 0 days 00:00:04 -0.241778  0.912946  0.321502
```

9.1.1 ->8.1 (c) Load and Prepare Data (Heart rate)

7

- # ------------ Rounding Heart Rate (Rounding to 1 sec) ------------------------------
  - HeartR_new['timedelta'] = pd.DataFrame(pd.to_timedelta(HeartR_new['timedelta'], timedelta_unit).round('1s'))

8

- # Resampling every 1s with median with ffill
- resample_rule = '1s'
- HeartR_new2 = HeartR_new.set_index('timedelta').resample(resample_rule,).median().ffill()

9.1.1 -> 8.1 (d) Load and Prepare Data (Sleep Label)

9

- # ------------ Rounding Sleep Label (Rounding to 1 sec) -------------------------------
  - SleepL_new['timedelta'] = pd.DataFrame(pd.to_timedelta(SleepL_new['timedelta'], timedelta_unit).round('1s'))

10

- # Resampling every 1s with median with ffill
- resample_rule = '1s'
- SleepL_new2 = SleepL_new.set_index('timedelta').resample(resample_rule,).median().ffill()

9.1.1 -> 8.1 (e) Merge Data and Standardized data

1

- # ------------Merge All Data ------------------------------
  - df = []
  - df = pd.merge_asof(ACC_new2, HeartR_new2, on='timedelta')
  - df = pd.merge_asof(df, df_SleepL_new2, on = 'timedelta')

2

- # Fill NA
- # Heart rate
  - Fillna() # using median()
- # Sleep Label
  - Fillna() # with 0
- # Drop column
  - drop('timedelta')

3

- # Standardized data
  - feature_columns = ['accX', 'accY', 'accZ', 'heartrate']
  - label_columns = ['sleep']
  - df_feature = df[feature_columns] <= standardized data of df_feature
  - df_label = df[label_columns]

4

- # Visualize signals
  - df_feature.plot(), df_label.plot()

9.2.1 Create 3d input

1

- # ------------ 1D to 3D feature-------------------------------
- # set sliding window parameter
- slidingW = 100
- Stride_step = 5

- For t in range(0 , len(df_feature), stride_step)
  - F3d= df_feature(t : t + slidingW))
  - df_feature3D.append(F3d)
  - df_feature3D.reshape(slidingW, n_feature, 1)

  - Labels = stats.mode(df_label (t : t+slidingW))
  - df_label_new.append(Labels)

i/p shape: (timestamp_size, features)
(Nrows, 4)

o/p shape: (Nsets, slidingW, n_feature, 1)
(Nsets, 100, 4, 1)

	ACC_X	ACC_Y	ACC_Z	HeartR	label
	1	1	1	1	
	2	2	2	2	
	3	3	3	3	
	4	4	4	4	
set#1	5	5	5	5	L1
	6	6	6	6	majority(L1:10)
	7	7	7	7	
	8	8	8	8	
	9	9	9	9	
	10	10	10	10	
	5	5	5	5	
	6	6	6	6	
	7	7	7	7	
	8	8	8	8	
set#2	9	9	9	9	L2
	10	10	10	10	majority(L5:14)
	11	11	11	11	
	12	12	12	12	
	13	13	13	13	
	14	14	14	14	
	10	10	10	10	
	11	11	11	11	
	12	12	12	12	
	13	13	13	13	
set#3	14	14	14	14	L3
	15	15	15	15	majority(L10:19)
	16	16	16	16	
	17	17	17	17	
	18	18	18	18	
	19	19	19	19	

9.1.2 Train Test Split

2

- # ------------ Train-Test-Split 2D features ------------------------------
  - x_train, x_test, y_train, y_test = train_test_split(df_feature, df_label)

3

- # ------------ Train-Test-Split 3D features ------------------------------
  - x3D_train, x3D_test, y3D_train, y3D_test = train_test_split(df_feature3D , df_label_new)

9.2 NN Model Train and Test

9.2.1 NN Model Train Test

1

- # ------------ NN Architecture parameter ------------------------------
  - Hidden_Layer_param = (30, 30, 30)
  - mlp = MLPClassifier(hidden_layer_sizes = Hidden_Layer_param)
- # View NN model parameters

2

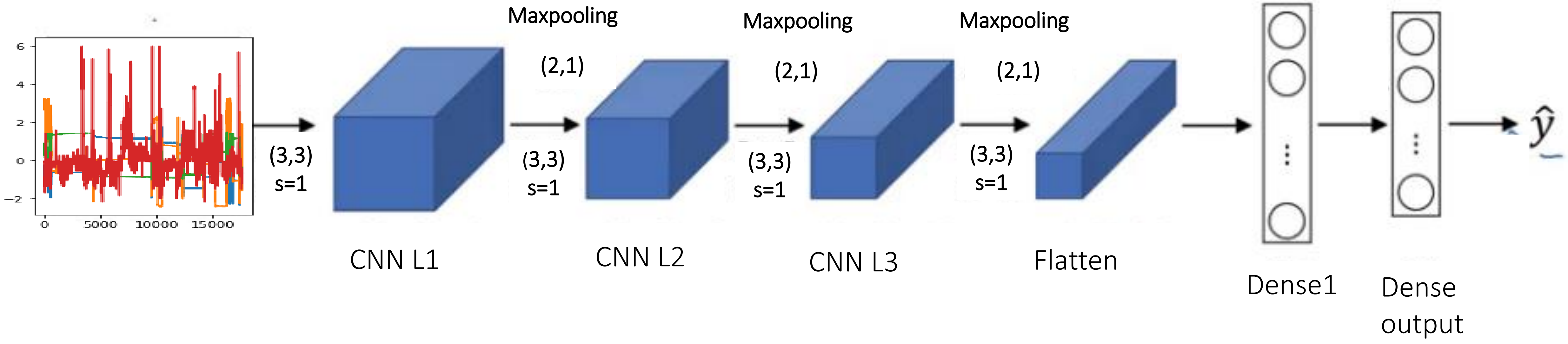
- # ------------ Training NN using 1D features ------------------------------
  - mlp.fit(X_train,y_train)
  - mlp_pred = mlp.predict(X_test)

3

- View Confusion Matrix and Classification Report

9.3 CNN Model Train and Test

CNN Model Architecture



9.3.1 CNN Model Train Test

1

- # ------------ CNN Architecture parameter --------------------------------
- # Nlayer (CNN, dense), Nnode, Activation
  - CNN_L1 = 16, CNN_L2 = 64, CNN_L3 = 128
  - D_L1 = 512, D_out = 6
  - Activation = “Relu”
  - Ker_size = (3,3)
  - Pooling_size = (2,1)
  - Input_shape = (slidingW, n_feature, 1)

9.3.1 CNN Model Train Test

2

- # ------------ Create CNN Model -------------------------------
  - model.add(Conv2D(CNN_L1, kernel_size=Ker_size, activation=Act_func, input_shape=Input_shape,padding='same'))
  - model.add(MaxPooling2D(pool_size=Pooling_size))
  - model.add(Dropout(0.4))

  - model.add(Conv2D(CNN_L2, kernel_size=Ker_size, activation= Act_func, padding='same'))
  - model.add(MaxPooling2D(pool_size= Pooling_size))
  - model.add(Dropout(0.4))

  - model.add(Conv2D(CNN_L3, kernel_size=Ker_size, activation= Act_func,
  - padding='same'))
  - model.add(MaxPooling2D(pool_size= Pooling_size))
  - model.add(Dropout(0.4))

  - model.add(Flatten())
  - model.add(Dense(D_L1_size , activation= Act_func))
  - model.add(Dense(D_out, activation='sigmoid'))

  - model.compile(optimizer='adam', metrics=['accuracy'])
  - model.summary()

9.3.1 CNN Model Train Test

3

- # ------------ Create Optimizer -------------------------------
  - model.compile(optimizer='adam',
  - loss= tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
  - metrics=["acc"])

4

- # ------ Train CNN using 3D feature---
  - history = model.fit(X3D_train, y3D_train, epochs=50, batch_size=64,
  - validation_data=(X3D_test, y3D_test))

5

- # ------- Test CNN -------------------------------
  - CNN_pred = model.predict(X3D_test)

9.3.2 Performnace of CNN Model

3

- # ------------ View Confusion Matrix, Classification Report -------------------------------

4

- # ------ View History Graph --
  - # View Accuracy Graph
    - # summarize history for accuracy
    - plt.plot(history.history['acc'])
    - plt.plot(history.history['val_acc'])
    - plt.show()
  - # View Loss Graph
    - # summarize history for loss
    - plt.plot(history.history['loss'])
    - plt.plot(history.history['val_loss'])
    - plt.show()

History Graph (Accuracy, Loss)