ชื่อ-สกุล_____ห้อง_____รหัสนักศึกษา_____

ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
วิชา Machine Learning Laboratory

การทดลองที่ 4 : การทดลองปรับค่าพารามิเตอร์เพื่อสร้างโมเดลการจัดกลุ่มข้อมูลด้วยเทคนิคการจัดกลุ่มข้อมูลแบบต่างๆและเปรียบเทียบประสิทธิภาพของโมเดลทดสอบ

วัตถุประสงค์

1. เพื่อศึกษาและทดลองการใช้งานโมเดลการจัดกลุ่มข้อมูลแบบต่างๆ
2. เพื่อศึกษาและทดลองการปรับค่าพารามิเตอร์ที่เหมาะสมกับโมเดลสำหรับชุดข้อมูลทดสอบ
3. เพื่อศึกษาและทดลองเทคนิคการวัดประสิทธิภาพโมเดลการจัดกลุ่มข้อมูล
4. เพื่อศึกษาและทดลองการเปรียบเทียบประสิทธิภาพของโมเดลต่างๆเชิงกราฟ

อุปกรณ์ และเครื่องมือที่ใช้ในการทดลอง

1. โปรแกรม python

ข้อกำหนดในการตรวจการทดลอง

1. แสดงโค๊ดและภาพผลการทดลองที่ทำพร้อมอธิบาย
2. นศ.ที่ได้รับการตรวจจากอาจารย์เรียบร้อย อาจารย์จะเช็คส่งงานในระบบ
3. ให้นศ. นำ source code และ ภาพ figure ที่ให้แสดงทุกภาพ โพสลงใน google classroom ส่งภายในวันที่ 29 พย. 2562 เวลา 17.00 น.

ตอนที่ 1: การทดลองเตรียมข้อมูล ปรับค่าข้อมูล และจัดแบ่งชุด Train, Test เพื่อสอนโมเดล

1.1 Import Lib (numpy, pandas, matplotlib, sklearn, keras)

1.2 ฟังก์ชันใช้งาน

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.svm import SVC
from sklearn import model_selection
from sklearn import preprocessing
from sklearn import metrics
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from keras import optimizers
```

1.3 ให้นศ. โหลดไฟล์ Human Activity Dataset file ทดลอง “skeleton_filtered.csv” จาก

https://drive.google.com/open?id=1OBJK61boetZEX3DbZiDbx8XDgbiLHetL

ข้อมูลเพิ่มเติม

Resourece #1: https://github.com/ChengeYang/Human-Pose-Estimation-Benchmarking-and-Action-Recognition

Preprocessed data: “skeleton_filtered.csv”

Raw data: “skeleton_raw.csv”

Resource #2: https://github.com/felixchenfy/Realtime-Action-Recognition/tree/master/data

Noisy data: (class: ‘kick’, ‘stand’, ‘wave’):

https://drive.google.com/open?id=1wlAp4Bv8FKdPsMDFilgKzyLZ8VLbunbt

1.4 Read ข้อมูลจากไฟล์ที่โหลดมาในข้อ 1.3

read_csv()

โครงสร้างไฟล์ประกอบด้วยตำแหน่ง position_x, position_y ของ 17 ตำแหน่งในร่างกายมนุษย์คู่กับ class คำตอบของ activity 3 class ได้แก่ class: ‘kick’, ‘stand’, ‘wave’

1.5 เตรียมข้อมูล features X และ คำตอบ (ground truth) y

y = ground truth with dimension (จำนวน row)

X 1D-features ของข้อมูลแต่ละ Activity class

X = features with dimension (จำนวน row x จำนวน 34 features)

X 2D-features ของข้อมูลแต่ละ Activity class

X = features with dimension (จำนวน row x time_window x จำนวน 34 features)

โดยใช้ฟังก์ชันต่อไปนี้

```
def process_create_WindowTimeSeries(activity, df, activity_start, activity_len, time_window, n_feature, step_stride):
    df_series = df[df['class'] == activity].iloc[:,:-1][activity_start:(activity_start+activity_len)]
    segments = []
    labels = []
    for i in range(0, len(df_series) - time_window, step_stride):
        df_series_feature = df_series.iloc[i: i + time_window]
        label = activity
        segments.append(np.array(df_series_feature))
        labels = np.append(labels,label)

    reshaped_segments = np.asarray(segments, dtype= np.float32).reshape(-1, time_window, n_feature,1)

return reshaped_segments, labels
```

กำหนดให้ activity -> activity class

df -> dataframe from csv

activity_start -> start row ที่เริ่มนำ feature มาจัดเรียง

activity_len -> จำนวนระยะ sample ทั้งหมดใน activity class ที่ต้องการดึงมาสร้าง dataset feature

time_window -> จำนวน feature sample ต่อเนื่อง ที่ต้องการนำมาให้โมเดลเรียนรู้ความสัมพันธ์ของ 36 features ที่ต่อเนื่องในช่วงเวลา

step_stride -> ระยะห่างในการดึงชุด feature ลำดับ row ถัดไปห่างจาก row ของชุดก่อนเท่ากับจำนวน step_stride เช่น time_window = 5 Step_stride = 3 แสดงว่าถ้าข้อมูลชุดแรกจะเริ่มจาก feature sample ลำดับ row ที่ 1 - 5 ข้อมูลชุดถัดไปจะเริ่มที่ row 3 - 8

1.6 นำ Feature ของ X-1D และ X-2D ของแต่ละ Activity class มาเรียงต่อกัน เพื่อเตรียมนำไปสอนโมเดล

X-1D
X-1D class “kick” -> (#rows x 36 features)
X-1D class “stand” -> (#rows x 36 features)
X-1D class “wave” -> (#rows x 36 features)

X-2D
X-2D class “kick” -> (#rows x time_window x 36 features)
X-2D class “stand” -> (#rows x time_window x 36 features)
X-2D class “wave” -> (#rows x time_window x 36 features)

1.7 สร้างชุดข้อมูล train-> train/validate และ test-> test (aTest = %test_set_size)

xtrain, ytrain / x_validate, y_validate		xtest, ytest
X_train, y_train	X_validate, y_validate	X_future_test, y_future_test

```
# Splitting Train/Test a Test%
X_train, X_validate, y_train, y_validate = train_test_split(X, y, train_size=0.8, test_size=0.2, random_state=RANDOM_SEED, stratify=y)
```

1.8 จัดข้อมูล ground truth จาก 1D -> binary N-D class สำหรับผลลัพธ์ของ NN , CNN เช่น ข้อมูลเดิม y -> 3 class ให้จัดรูปแบบ y ใหม่ เป็น binary output ซึ่งให้ค่า ตำแหน่ง class คำตอบที่ถูกต้องเป็น 1 สำหรับ class ที่ผิดที่เหลือจะมีค่าเป็น 0 โดยใช้

Label Encoding

Food Name	Categorical #	Calories
Apple	1	95
Chicken	2	231
Broccoli	3	50

→

One Hot Encoding

Apple	Chicken	Broccoli	Calories
1	0	0	95
0	1	0	231
0	0	1	50

ตอนที่ 2: การทดลองทำ Cross validation และ prediction เพื่อดูค่าความแม่นยำของแต่ละโมเดล SVC

2.1 สร้างโมเดลรูปแบบ cross validation ที่ต้องการใช้ ในที่นี้ใช้ StratifiedKFold

```
seed  #กำหนดเพื่อให้ random ได้ชุดข้อมูลเดียวกันในทุกครั้งที่รัน
k # กำหนดจำนวนชุดข้อมูลย่อยที่จะถูกนำมาทำ StratifiedKFold Cross Validation
skfold = model_selection.StratifiedKFold(n_splits=k, shuffle = True, random_state=seed)
```

2.2 สร้าง prediction model ที่ต้องการนำมาทดสอบ

```
# Support Vector Regression (SVC) Model
c_val # ค่าปรับเข้มงวดกับ outlier bound [10^-6, 10^6] ค่าที่น่าสนใจ 100
gmm # ค่าความยืดหยุ่นต่อความซับซ้อน ควบคุมรูปร่างของโมเดล [10^-6, 10^6]
        ค่าน่าสนใจ 0.1
SVC_lin = SVC(kernel='linear', C=c_val)
SVC_rbf = SVC(kernel='rbf', C=c_val, gamma=gmm)
```

2.3 ทำ cross validation สำหรับโมเดลที่ต้องการระบุชื่อ model_name ตามข้อ 2.2 ดังตัวอย่างด้านล่าง

```
score = model_selection.cross_val_score(model_name, X_train, y_train, cv=skfold)
```

หมายเหตุ ให้ทำการคำนวณ score ซึ่งเป็นค่า (‘Accuracy’) ของโมเดลทั้ง 2 แบบที่สร้างในข้อ 2.2

2.4 plot กราะเปรียบเทียบ score ที่ได้จากการข้อ 2.3 ของโมเดลทั้ง 2 แบบที่คำนวณ

```
# รวบรวม score จากโมเดลทั้ง 4 แบบไว้ในรูปแบบของ dataframe
score_all = pd.DataFrame({'SVC_linear':score_lin, 'SVC_rbf': score_rbf })
# แสดงกราฟสำหรับ cross validation แต่ละ fold
score_all.plot()
```

2.5 แสดงการเปรียบเทียบค่า score และ ค่าทางสถิติของ score ทุก K-fold ที่ได้จากโมเดลทั้ง 2 แบบ โดยค่าเฉลี่ยหาได้จาก score.mean() และค่าเบี่ยงเบนหาได้จาก score.std() เพื่อดูว่าจากการสอนโมเดล มีโมเดลใดที่มีแนวโน้มที่จะทำนายค่าได้แม่นยำกว่ากัน

2.6 ทำการ predict ค่าจากโมเดลทั้ง 2 แบบ ด้วยข้อมูล X_test, y_test ที่แบ่งไว้

```
y_prediction = model_name.fit(X_train, y_train).predict(X_test)
```

2.7 คำนวณค่าตัววัดประสิทธิภาพของการทำนายจากโมเดลทั้ง 2 แบบ โดยวัดค่า precision/recall/f1-score/ confusion matrix

```
classification_report(y_test,y_pred)
```

confusion_matrix(y_test,y_pred)

2.8 ทำการ predict ค่า

y_predict_future = model_name.predict(X_future_test)

2.9 คำนวณค่าตัววัดประสิทธิภาพของการทำนายจากโมเดลทั้ง 2 แบบ โดยวัดค่า precision/recall/f1-score/ confusion matrix เช่นเดียวกับในข้อ 2.7

_____

(อาจารย์ตรวจผลการทดลอง)

ตอนที่ 3: การทดลองทำ Cross validation และ prediction เพื่อดูค่าความแม่นยำของแต่ละโมเดล Neural Network และ Convolutional Neural Network

3.1 สร้าง prediction model ที่ต้องการนำมาทดสอบ

```
# Multi-layer Perceptron (MLP) Neural Network
from sklearn.neural_network import MLPClassifier
mlp = MLPClassifier(hidden_layer_sizes=(30,30,30),max_iter=500, solver='sgd')

# keras CNN
model = Sequential()
#L1
        model.add(Conv2D(CNN_L1, kernel_size=Ker_size, activation=Act_func,
            input_shape=Input_shape,padding='same'))
        model.add(MaxPooling2D(pool_size=P_size))
        model.add(Dropout(0.2))
....
#Ln
        model.add(Conv2D(CNN_L,n kernel_size=Ker_size, activation=Act_func,
            input_shape=Input_shape,padding='same'))
        model.add(MaxPooling2D(pool_size=P_size))
        model.add(Dropout(0.2))

model.add(Flatten())
model.add(Dense(Dense_size , activation= Act_func ))
```

model.add(Dense(#output_class, activation='sigmoid'))

model.compile(loss='categorical_crossentropy',

optimizer=optimizers.Adam(lr=0.001, beta_1=0.9, beta_2=0.999,

amsgrad=False), metrics=['accuracy'])

3.2 ทำการ train mlp model

NN: mlp.fit(X_train, y_train)

CNN: model.summary()

history = model.fit(X_train,y_train, epochs=#Ep,batch_size=#Bs,

validation_data=(X_test,y_test), verbose=1)

3.3 ทำการ predict ด้วยโมเดล mlp ที่ train ไว้ในข้อ 3.2 ด้วยข้อมูล X_test ที่แบ่งไว้

NN: y_pred = mlp.predict(X_test)

CNN: CNN_pred = model.predict(X_test)

CNN_pred_single = [np.argmax(p) for p in CNN_pred]

y_test_single=[np.argmax(p) for p in y_test]

3.4 คำนวณค่าตัววัดประสิทธิภาพของการทำนายจากโมเดล ที่ทดสอบด้วย y_test เช่นเดียวกับข้อ 2.7

Classification report

NN: classification_report(y_test,y_pred)

CNN: classification_report(y_test_single, CNN_pred_single)

Confusion Matrix

NN: confusion_matrix(y_test,y_pred)

CNN: print(confusion_matrix(y_test_single, CNN_pred_single))

3.5 ทำการ predict ด้วยโมเดล mlp ที่ train ไว้ในข้อ 3.2 ด้วยข้อมูล X_future_test ที่แบ่งไว้

NN: mlp_pred_future = mlp.predict(X_future_test)

CNN: CNN_pred_future = model.predict(X_future_ test)

CNN_pred_future_single = [np.argmax(p) for p in CNN_pred_future]

3.6 คำนวณค่าตัววัดประสิทธิภาพของการทำนายจากโมเดลที่ทดสอบด้วย y_future_test เช่นเดียวกับข้อ 3.4 (#Classification report และ #Confusion Matrix)

ตอนที่ 4: การทดลองการค้นหาพารามิเตอร์ที่ดีที่สุดสำหรับโมเดล

4.1 กำหนดรายการพารามิเตอร์ทั้งหมดที่ต้องการทดสอบหาค่าที่ดีที่สุดของโมเดล

4.1.1 SVC แบบใช้ kernel='rbf'

```
c_param # เลือกค่าในช่วง [0.1, 1000] จำนวน 4 ค่า
gamma = # เลือกค่าในช่วง [0.01, 0.5] จำนวน 3 ค่า
tuned_parameters = [{'kernel': ['rbf'], 'C': c_param, 'gamma': gamma}]
```

4.1.2 MLP

```
param_grid = [{
    'hidden_layer_sizes': [(10,10), (20,20), (30,30,30)],
    'max_iter' : [200, 300, 500],
    'solver': ['adam','sgd','lbfgs']
}]
```

4.1.3 CNN

```
batch_size = [10, 20, 40, 60, 80, 100]
epochs = [10, 50, 100]
optimizers = ['adam','adadelta']
param_grid = dict(batch_size=batch_size, epochs=epochs, optimizer=optimizers)
```

4.2 ใช้ฟังก์ชั่น GridSearchCV เพื่อทำ cross validation หาค่าพารามิเตอร์ที่ดีที่สุด

```
# กำหนดโมเดล
Model1 = SVC()
Model2 = MLP()
# ใช้ cross validation (cv) เป็น kfold ที่กำหนดไว้ในตอนข้อ 2.1
skf =model_selection.StratifiedKFold(n_splits=10, shuffle = True, random_state=seed)
gsCV_SVC = model_selection.GridSearchCV(Model1, tuned_parameters, cv = skf)
gsCV_MLP = model_selection.GridSearchCV(Model2, param_grid, cv = skf)
gsCV_CNN = model_selection.GridSearchCV(estimator=model, param_grid=param_grid,
n_jobs=2, cv=3)
```

4.3 นำค่าพารามิเตอร์ที่ดีที่สุดไปสอนโมเดล

```
gsCV_SVC.fit(X_train, y_train)
gsCV_MLP.fit(X_train, y_train)
gsCV_CNN.fit(X_train, y_train)
```

4.4 save ผลลัพธ์จากการทำ GridSearchCV ลงบนไฟล์ .csv

gridsearch_result = pd.DataFrame(gsCV.cv_results_)

gridsearch_result.to_csv('filename.csv')

4.5 แสดงค่าพารามิเตอร์ที่ดีที่สุด

gsCV.best_params_, gsCV.best_score

4.6 ใช้โมเดลที่สอนจากพารามิเตอร์ที่ดีที่สุดมา predict ข้อมูล ชุด x_future_test

SVC_predict_test = gsCV_SVC.predict(X_test)

SVC_predict_forecast = gsCV_SVC.predict(x_future_test)

MLP_predict_test = gsCV_ MLP.predict(X_test)

MLP _predict_forecast = gsCV_ MLP.predict(x_future_test)

CNN_predict_test = gsCV_ CNN.predict(X_test)

CNN _predict_forecast = gsCV_ CNN.predict(x_future_test)

4.7 คำนวณค่าตัววัดประสิทธิภาพของการทำนายจากโมเดลที่ทดสอบด้วย y_future_test เช่นเดียวกับข้อ 3.4

classification_report(y_future_test, y_pred_future)

confusion_matrix(y_future_test, y_pred_future)

_____

(อาจารย์ตรวจผลการทดลอง)