all_steps_activity recognition_final_version_split_cycling_7_seconds

January 11, 2021

[95]:
```python
from helpers import math_helper
from sensors.activpal import *
from utils import read_functions
from scipy import signal
from sklearn.model_selection import train_test_split
from sklearn import tree
from sklearn.metrics import f1_score, plot_confusion_matrix, confusion_matrix,␣
 ↪accuracy_score, precision_score, recall_score, confusion_matrix,␣
 ↪classification_report
from sklearn.ensemble import RandomForestClassifier

import pandas as pd
import numpy as np
import statistics
import os
import pickle
import matplotlib.pyplot as plt
```

Adnan Akbas # Feature Extraction

[96]:
```python
activpal = Activpal()

features_columns = ['standard_deviation_x', 'mean_x', 'standard_deviation_y',␣
 ↪'mean_y','standard_deviation_z', 'mean_z', 'activiteit']
#activity_columns = ['activity_cycling', 'activity_walking',␣
 ↪'activity_running', 'activity_jumping', 'activity_standing',␣
 ↪'activity_traplopen', 'activity_sitten']
#activities = ['fietsen licht', 'fietsen zwaar', 'lopen', 'rennen', 'springen',␣
 ↪'staan', 'traplopen', 'zitten']

activity_columns = ['activity_cycling_light', 'activity_cycling_heavy',␣
 ↪'activity_walking', 'activity_running', 'activity_standing',␣
 ↪'activity_sitten']
activities = ['fietsen licht', 'fietsen zwaar', 'lopen', 'rennen', 'staan',␣
 ↪'zitten']
```

```
test_users = ['BMR004', 'BMR034', 'BMR097']
segment_size = 7.0
number_of_trees = 203
```

[97]:

```
def extract_features_from_correspondent(correspondent):
    features_df = pd.DataFrame(columns=features_columns, index=pd.
 ↪to_datetime([]))

    # Getting dataset for a correspodent
    activities_df = read_functions.read_activities(correspondent)

    for activity_name in activities:
        activity = activities_df.loc[activity_name]
        if not activity.empty:
            start_time = activity.start
            stop_time = activity.stop
            activpal_df = activpal.read_data(correspondent, start_time,␣
 ↪stop_time)

            # denormalizing dataset
            activpal_df['x'] = math_helper.
 ↪convert_value_to_g(activpal_df['pal_accX'])
            activpal_df['y'] = math_helper.
 ↪convert_value_to_g(activpal_df['pal_accY'])
            activpal_df['z'] = math_helper.
 ↪convert_value_to_g(activpal_df['pal_accZ'])

            date_range = pd.date_range(start_time, stop_time,␣
 ↪freq=str(segment_size) + 'S')

            for time in date_range:
                segment_time = time + pd.DateOffset(seconds=segment_size)
                activpal_segment = activpal_df[(activpal_df.index >= time) &␣
 ↪(activpal_df.index < segment_time)]

                stdev_x =  statistics.stdev(activpal_segment['x']) if␣
 ↪len(activpal_segment['x']) >= 2 else 0
                mean_x = activpal_segment['x'].mean()

                stdev_y =  statistics.stdev(activpal_segment['y']) if␣
 ↪len(activpal_segment['y']) >= 2 else 0
                mean_y = activpal_segment['y'].mean()

                stdev_z =  statistics.stdev(activpal_segment['z']) if␣
 ↪len(activpal_segment['z']) >= 2 else 0
                mean_z = activpal_segment['z'].mean()
                features_df.loc[segment_time] = [stdev_x, mean_x, stdev_y,␣
 ↪mean_y, stdev_z, mean_z, activity_name]

    features_df.dropna(how='any', inplace=True)

    return features_df
```

[98]:

```
def extract_features_from_correspondents(correspodents):
    all_features_df = pd.DataFrame(index=pd.to_datetime([]))

    for correspodent in correspodents:
        print("Extracting " + correspodent)

        features_df     = extract_features_from_correspondent(correspodent)
        all_features_df = pd.concat([all_features_df, features_df])

    print("Done extracting features")

    return all_features_df

def extract_features_from_all_correspondents(exclude_test_correspodent = True):

    exclude_directory = ['output', 'throughput', 'Test data','.
 ↪ipynb_checkpoints']
    exclude_respodents = ['BMR015','BMR025','BMR027', 'BMR035', 'BMR051',␣
 ↪'BMR054', 'BMR060', 'BMR099', 'BMR100']

    exclude = exclude_respodents + exclude_directory

    if (exclude_test_correspodent):
        exclude = exclude + test_users

    correspodents = []

    for directory in os.walk('../../data'):
        if directory[0] == '../../data':
            correspodents = directory[1]

    for exclude_item in exclude:
        if exclude_item in correspodents:
            correspodents.remove(exclude_item)

    return extract_features_from_correspondents(correspodents)
```

[99]:

```
features_dataset = extract_features_from_all_correspondents()
```

```
Extracting BMR012
Extracting BMR030
Extracting BMR044
Extracting BMR043
Extracting BMR011
Extracting BMR098
Extracting BMR014
Extracting BMR036
Extracting BMR052
Extracting BMR002
Extracting BMR031
Extracting BMR008
Extracting BMR033
Extracting BMR064
Extracting BMR055
Extracting BMR041
Extracting BMR053
Extracting BMR042
Extracting BMR018
Extracting BMR058
Extracting BMR040
Extracting BMR032
Done extracting features
```

1 model preperation

[100]:

```
features_dataset[activity_columns] = 0

#features_dataset.loc[(features_dataset['activiteit'] == 'springen'),␣
 ↪'activity_jumping'] = 1
#features_dataset.loc[(features_dataset['activiteit'] == 'traplopen'),␣
 ↪'activity_traplopen'] = 1
features_dataset.loc[(features_dataset['activiteit'] == 'lopen'),␣
 ↪'activity_walking'] = 1
features_dataset.loc[(features_dataset['activiteit'] == 'rennen'),␣
 ↪'activity_running'] = 1
features_dataset.loc[(features_dataset['activiteit'] == 'staan'),␣
 ↪'activity_standing'] = 1
features_dataset.loc[(features_dataset['activiteit'] == 'zitten'),␣
 ↪'activity_sitten'] = 1
features_dataset.loc[(features_dataset['activiteit'] == 'fietsen licht'),␣
 ↪'activity_cycling_light'] = 1
features_dataset.loc[(features_dataset['activiteit'] == 'fietsen zwaar'),␣
 ↪'activity_cycling_heavy'] = 1
```

```python
features_dataset.drop('activiteit', axis=1, inplace=True)

features_dataset.head()
```

[100]:

```
                     standard_deviation_x    mean_x  standard_deviation_y  \
2019-10-14 09:44:07              0.455974 -0.812472              0.169643
2019-10-14 09:44:14              0.485342 -0.812358              0.177395
2019-10-14 09:44:21              0.495641 -0.805809              0.191314
2019-10-14 09:44:28              0.494310 -0.828798              0.183940
2019-10-14 09:44:35              0.495571 -0.844785              0.198304

                       mean_y  standard_deviation_z    mean_z  \
2019-10-14 09:44:07  0.103061              0.200659  0.819161
2019-10-14 09:44:14  0.104989              0.232790  0.789569
2019-10-14 09:44:21  0.104807              0.240046  0.782168
2019-10-14 09:44:28  0.113832              0.216198  0.782426
2019-10-14 09:44:35  0.118934              0.220348  0.786735

                     activity_cycling_light  activity_cycling_heavy  \
2019-10-14 09:44:07                       1                       0
2019-10-14 09:44:14                       1                       0
2019-10-14 09:44:21                       1                       0
2019-10-14 09:44:28                       1                       0
2019-10-14 09:44:35                       1                       0

                     activity_walking  activity_running  activity_standing  \
2019-10-14 09:44:07                 0                 0                  0
2019-10-14 09:44:14                 0                 0                  0
2019-10-14 09:44:21                 0                 0                  0
2019-10-14 09:44:28                 0                 0                  0
2019-10-14 09:44:35                 0                 0                  0

                     activity_sitten
2019-10-14 09:44:07                0
2019-10-14 09:44:14                0
2019-10-14 09:44:21                0
2019-10-14 09:44:28                0
2019-10-14 09:44:35                0
```

1.1 Preparing feature dataset for learning

1.1.1 Splitting in x and y

[101]:
```
x = features_dataset[features_columns[:-1]]
y = features_dataset[activity_columns]

## split
x_train, x_valid, y_train, y_valid = train_test_split(x, y, test_size=0.2,␣
 ↪random_state=0)
```

2 Random tree forest

[103]:
```
ftc = RandomForestClassifier(n_estimators=number_of_trees, random_state=0)
ftc.fit(x_train, y_train)
```

[103]:
```
RandomForestClassifier(n_estimators=203, random_state=0)
```

2.1 Validation result

[104]:
```
predictions = ftc.predict(x_valid)
```

Accuracy

[105]:
```
accuracy_score(y_valid, predictions, normalize=True)
```

[105]:
```
0.950920245398773
```

F1

[106]:
```
f1_score(y_valid, predictions, average='micro')
```

[106]:
```
0.9513371328364753
```

Precision

[107]:
```
precision_score(y_valid, predictions, average='micro')
```

[107]:
```
0.9517543859649122
```

Recall

[108]:
```
recall_score(y_valid, predictions, average='micro')
```

[108]: 0.950920245398773

Classification report

```
[109]: print(classification_report(y_valid, predictions,␣
        ↪target_names=activity_columns, zero_division=0))
```

```
                        precision    recall  f1-score   support

activity_cycling_light       0.89      0.87      0.88       191
activity_cycling_heavy       0.87      0.89      0.88       185
      activity_walking       0.99      0.97      0.98       186
      activity_running       0.98      0.99      0.98       181
     activity_standing       0.99      0.99      0.99       209
       activity_sitten       0.99      0.99      0.99       189

             micro avg       0.95      0.95      0.95      1141
             macro avg       0.95      0.95      0.95      1141
          weighted avg       0.95      0.95      0.95      1141
           samples avg       0.95      0.95      0.95      1141
```

Confusion matrix

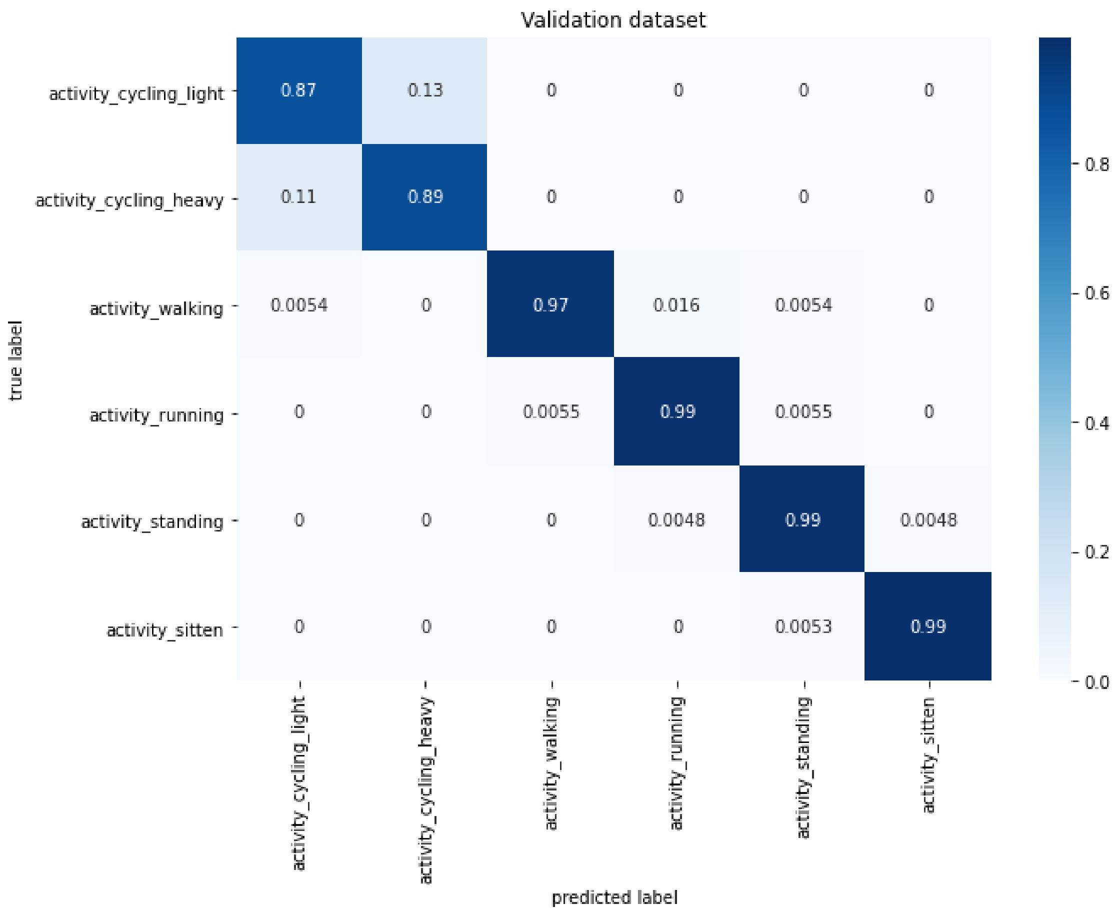
```
[110]: import seaborn as sn

       #confusion_matrix(valid_y, prediction_y)
       cm = confusion_matrix(y_valid.values.argmax(axis=1), predictions.
        ↪argmax(axis=1), normalize='true')

       df_cm = pd.DataFrame(cm, index=activity_columns, columns=activity_columns)
       df_cm.head()
       plt.figure(figsize = (10,7))
       sn.heatmap(df_cm, annot=True, cmap='Blues')

       plt.title("Validation dataset")
       plt.xlabel("predicted label")
       plt.ylabel("true label")
```

```
[110]: Text(68.09375, 0.5, 'true label')
```



2.2 k-fold cross validation

[111]:

```
from sklearn.model_selection import cross_val_score
import seaborn as sn
from sklearn.model_selection import cross_val_predict

x = features_dataset[features_columns[:-1]]
y = features_dataset[activity_columns]

accuracy_scores = cross_val_score(␣
 ↪RandomForestClassifier(n_estimators=number_of_trees, random_state=0), x, y,␣
 ↪cv=5, scoring='accuracy')
recall_scores = cross_val_score(␣
 ↪RandomForestClassifier(n_estimators=number_of_trees, random_state=0), x, y ,␣
 ↪cv=5, scoring='recall_micro')
```

```
precision_scores = cross_val_score(␣
 ↪RandomForestClassifier(n_estimators=number_of_trees, random_state=0), x, y ,␣
 ↪cv=5, scoring='precision_micro')

print("Accuracy: %0.2f (+/- %0.2f)" % (accuracy_scores.mean(), accuracy_scores.
 ↪std() ))
print("Precision: %0.2f (+/- %0.2f)" % (precision_scores.mean(),␣
 ↪precision_scores.std() ))
print("Recall: %0.2f (+/- %0.2f)" % (recall_scores.mean(), recall_scores.std()␣
 ↪))
```

```
Accuracy: 0.83 (+/- 0.04)
Precision: 0.84 (+/- 0.04)
Recall: 0.83 (+/- 0.04)
```

2.3 Test result

[112]:

```
test_dataset = extract_features_from_correspondents(test_users)

test_dataset[activity_columns] = 0

test_dataset.loc[(test_dataset['activiteit'] == 'lopen'), 'activity_walking'] =␣
 ↪1
test_dataset.loc[(test_dataset['activiteit'] == 'rennen'), 'activity_running']␣
 ↪= 1
test_dataset.loc[(test_dataset['activiteit'] == 'staan'), 'activity_standing']␣
 ↪= 1
test_dataset.loc[(test_dataset['activiteit'] == 'zitten'), 'activity_sitten'] =␣
 ↪1
test_dataset.loc[(test_dataset['activiteit'] == 'fietsen licht'),␣
 ↪'activity_cycling_light'] = 1
test_dataset.loc[(test_dataset['activiteit'] == 'fietsen zwaar'),␣
 ↪'activity_cycling_heavy'] = 1

test_dataset.drop('activiteit', axis=1, inplace=True)
test_dataset.dropna(how='any', inplace=True)

x = test_dataset[features_columns[:-1]]
y = test_dataset[activity_columns]
```

```
Extracting BMR004
Extracting BMR034
Extracting BMR097
Done extracting features
```

[113]:

```
test_prediction_y = ftc.predict(x)
```

accuracy

[114]:
```
accuracy_score(y, test_prediction_y, normalize=True)
```

[114]:
```
0.8394904458598726
```

F1

[115]:
```
f1_score(y, test_prediction_y, average='micro')
```

[115]:
```
0.8400254939451881
```

Precision

[116]:
```
precision_score(y, test_prediction_y, average='micro')
```

[116]:
```
0.8405612244897959
```

Recall

[117]:
```
recall_score(y, test_prediction_y, average='micro')
```

[117]:
```
0.8394904458598726
```

Classification report

[118]:
```
print(classification_report(y,test_prediction_y, target_names=activity_columns,␣
 ↪zero_division=0))
```

```
                         precision    recall  f1-score   support

activity_cycling_light       0.54      0.79      0.64       129
activity_cycling_heavy       0.60      0.32      0.42       129
      activity_walking       0.96      1.00      0.98       130
      activity_running       1.00      0.93      0.96       139
     activity_standing       0.97      0.99      0.98       129
       activity_sitten       1.00      1.00      1.00       129

             micro avg       0.84      0.84      0.84       785
             macro avg       0.84      0.84      0.83       785
          weighted avg       0.85      0.84      0.83       785
           samples avg       0.84      0.84      0.84       785
```

Confusion matrix

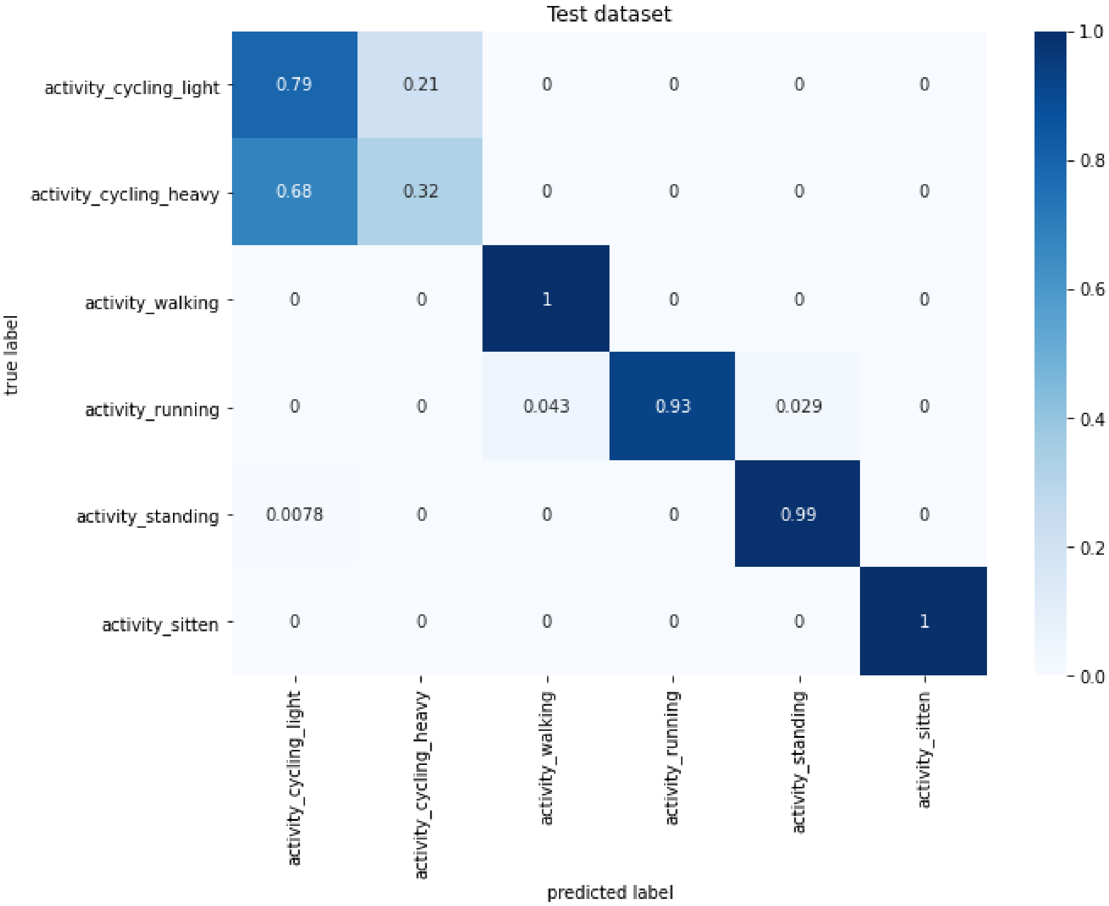
[121]:
```
import seaborn as sn
```

```python
#confusion_matrix(valid_y, prediction_y)
cm = confusion_matrix(y.values.argmax(axis=1), test_prediction_y.
 ↪argmax(axis=1), normalize='true')

df_cm = pd.DataFrame(cm, index=activity_columns, columns=activity_columns)
df_cm.head()
plt.figure(figsize = (10,7))
sn.heatmap(df_cm, annot=True, cmap='Blues')

plt.title("Test dataset")
plt.xlabel("predicted label")
plt.ylabel("true label")
```

```
[121]: Text(68.09375, 0.5, 'true label')
```



3 save model

[120]:

```
#from joblib import dump

#dump(ftc, 'activity.dat')
```