Model_obstacles

November 13, 2021

[1]:
```
from pandas import *

# devices
positions_stations = {"esp20":{"name":"esp20","x":7835,"y":8690.
↪84716796875},"esp21":{"name":"esp21","x":955,"y":12270.84716796875},"esp22":
↪{"name":"esp22","x":8935,"y":12120.84716796875}}

df = read_csv("device_all.csv")
df
```

[1]:
```
    device  measure_no  rssi  measured_distance      labelx       labely
0    esp20           1   -87           4.757198  3403.97583  6959.674805
1    esp20           1   -89           4.757198  3403.97583  6959.674805
2    esp20           1   -87           4.757198  3403.97583  6959.674805
3    esp20           1   -88           4.757198  3403.97583  6959.674805
4    esp20           1   -90           4.757198  3403.97583  6959.674805
..     …           …    …                 …          …            …
525  esp22           7   -96           5.312698  5208.35498  8334.440430
526  esp22           7   -96           5.312698  5208.35498  8334.440430
527  esp22           7   -94           5.312698  5208.35498  8334.440430
528  esp22           7   -95           5.312698  5208.35498  8334.440430
529  esp22           7   -97           5.312698  5208.35498  8334.440430

[530 rows x 6 columns]
```

[2]:
```
import math
df["dx"] = df.apply(lambda row: row['labelx'] -␣
↪positions_stations[row['device']]['x'], axis=1)
df["dy"] = df.apply(lambda row: row['labely'] -␣
↪positions_stations[row['device']]['y'], axis=1)
df["angle"] = df.apply(lambda row: math.atan2(row['dy'] , row['dx']) / math.pi␣
↪* 180, axis=1)
```

[3]:
```
df
```

[3]:
```
    device  measure_no  rssi  measured_distance      labelx       labely  \
0    esp20           1   -87           4.757198  3403.97583  6959.674805
```

```
1    esp20           1   -89           4.757198  3403.97583  6959.674805
2    esp20           1   -87           4.757198  3403.97583  6959.674805
3    esp20           1   -88           4.757198  3403.97583  6959.674805
4    esp20           1   -90           4.757198  3403.97583  6959.674805
..     …          …    …              …         …          …
525  esp22           7   -96           5.312698  5208.35498  8334.440430
526  esp22           7   -96           5.312698  5208.35498  8334.440430
527  esp22           7   -94           5.312698  5208.35498  8334.440430
528  esp22           7   -95           5.312698  5208.35498  8334.440430
529  esp22           7   -97           5.312698  5208.35498  8334.440430

             dx           dy       angle
0   -4431.02417 -1731.172363 -158.659735
1   -4431.02417 -1731.172363 -158.659735
2   -4431.02417 -1731.172363 -158.659735
3   -4431.02417 -1731.172363 -158.659735
4   -4431.02417 -1731.172363 -158.659735
..          …          …          …
525 -3726.64502 -3786.406738 -134.544257
526 -3726.64502 -3786.406738 -134.544257
527 -3726.64502 -3786.406738 -134.544257
528 -3726.64502 -3786.406738 -134.544257
529 -3726.64502 -3786.406738 -134.544257

[530 rows x 9 columns]
```

[ ]:

```

```

[4]:

```
groupbymeasure = df.groupby(["device","measure_no"])
```

[5]:

```
g = groupbymeasure.mean()
g = g.reset_index()
```

[6]:

```
g
```

[6]:

```
   device  measure_no       rssi  measured_distance       labelx  \
0   esp20           1 -88.166667           4.757198  3403.975830
1   esp20           3 -84.122449           3.668797  7743.078125
2   esp20           4 -70.346154           1.450394  9275.368164
3   esp20           5 -83.894737           3.173801  5766.853516
4   esp20           6 -98.272727           2.650715  5208.354980
5   esp20           7 -96.700000           2.650715  5208.354980
6   esp21           1 -84.666667           5.848593  3403.975830
7   esp21           2 -88.000000           4.130146  4148.640137
8   esp21           3 -99.666667           6.788644  7743.078125
9   esp21           4 -96.000000           9.126491  9275.368164
10  esp21           5 -87.409091           4.952658  5766.853516
```

```
11  esp21           6 -91.530201           5.795371  5208.354980
12  esp21           7 -91.757143           5.795371  5208.354980
13  esp22           1 -94.555556           7.565046  3403.975830
14  esp22           3 -75.450000           1.215382  7743.078125
15  esp22           4 -87.200000           3.616294  9275.368164
16  esp22           5 -87.380952           3.329080  5766.853516
17  esp22           6 -98.000000           5.312698  5208.354980
18  esp22           7 -96.333333           5.312698  5208.354980

          labely           dx           dy       angle
0    6959.674805 -4431.024170 -1731.172363 -158.659735
1   12358.492188   -91.921875  3667.645020   91.435699
2    8520.606445  1440.368164  -170.240723   -6.740660
3   11098.291016 -2068.146484  2407.443848  130.664663
4    8334.440430 -2626.645020  -356.406738 -172.272788
5    8334.440430 -2626.645020  -356.406738 -172.272788
6    6959.674805  2448.975830 -5311.172363  -65.245624
7    9651.922852  3193.640137 -2618.924316  -39.353248
8   12358.492188  6788.078125    87.645020    0.739740
9    8520.606445  8320.368164 -3750.240723  -24.262525
10  11098.291016  4811.853516 -1172.556152  -13.694973
11   8334.440430  4253.354980 -3936.406738  -42.783728
12   8334.440430  4253.354980 -3936.406738  -42.783728
13   6959.674805 -5531.024170 -5161.172363 -136.981118
14  12358.492188 -1191.921875   237.645020  168.724246
15   8520.606445   340.368164 -3600.240723  -84.599287
16  11098.291016 -3168.146484 -1022.556152 -162.111911
17   8334.440430 -3726.645020 -3786.406738 -134.544257
18   8334.440430 -3726.645020 -3786.406738 -134.544257
```

[7]:
```
p = g.pivot(index=["measure_no"],columns=["device"], values=["rssi",␣
 ↪"angle","measured_distance", "labelx", "labely"])
```

[8]:
```
p = p.reset_index()
```

[9]:
```
p
```

[9]:
```
       measure_no       rssi                             angle             \
device                 esp20      esp21      esp22       esp20      esp21
0                1 -88.166667 -84.666667 -94.555556 -158.659735 -65.245624
1                2        NaN -88.000000        NaN         NaN -39.353248
2                3 -84.122449 -99.666667 -75.450000   91.435699   0.739740
3                4 -70.346154 -96.000000 -87.200000   -6.740660 -24.262525
4                5 -83.894737 -87.409091 -87.380952  130.664663 -13.694973
5                6 -98.272727 -91.530201 -98.000000 -172.272788 -42.783728
6                7 -96.700000 -91.757143 -96.333333 -172.272788 -42.783728
```

```
                   measured_distance                           labelx  \
device       esp22             esp20     esp21     esp22        esp20
0      -136.981118          4.757198  5.848593  7.565046  3403.975830
1              NaN               NaN  4.130146       NaN          NaN
2       168.724246          3.668797  6.788644  1.215382  7743.078125
3       -84.599287          1.450394  9.126491  3.616294  9275.368164
4      -162.111911          3.173801  4.952658  3.329080  5766.853516
5      -134.544257          2.650715  5.795371  5.312698  5208.354980
6      -134.544257          2.650715  5.795371  5.312698  5208.354980

                                        labely
device        esp21        esp22         esp20         esp21         esp22
0       3403.975830  3403.975830   6959.674805   6959.674805   6959.674805
1       4148.640137          NaN           NaN   9651.922852           NaN
2       7743.078125  7743.078125  12358.492188  12358.492188  12358.492188
3       9275.368164  9275.368164   8520.606445   8520.606445   8520.606445
4       5766.853516  5766.853516  11098.291016  11098.291016  11098.291016
5       5208.354980  5208.354980   8334.440430   8334.440430   8334.440430
6       5208.354980  5208.354980   8334.440430   8334.440430   8334.440430
```

[10]:
```
p.columns
```

[10]:
```
MultiIndex([(       'measure_no',      ''),
            (             'rssi', 'esp20'),
            (             'rssi', 'esp21'),
            (             'rssi', 'esp22'),
            (            'angle', 'esp20'),
            (            'angle', 'esp21'),
            (            'angle', 'esp22'),
            ('measured_distance', 'esp20'),
            ('measured_distance', 'esp21'),
            ('measured_distance', 'esp22'),
            (           'labelx', 'esp20'),
            (           'labelx', 'esp21'),
            (           'labelx', 'esp22'),
            (           'labely', 'esp20'),
            (           'labely', 'esp21'),
            (           'labely', 'esp22')],
           names=[None, 'device'])
```

[11]:
```
p["rssi","esp20"]
```

[11]:
```
0   -88.166667
1          NaN
2   -84.122449
3   -70.346154
4   -83.894737
```

```
5   -98.272727
6   -96.700000
Name: (rssi, esp20), dtype: float64
```

[12]:
```python
flatp = p.copy()
flatp.columns = ['_'.join(col).strip() for col in p.columns.values]
```

[13]:
```python
flatp.columns
```

[13]:
```
Index(['measure_no_', 'rssi_esp20', 'rssi_esp21', 'rssi_esp22', 'angle_esp20',
       'angle_esp21', 'angle_esp22', 'measured_distance_esp20',
       'measured_distance_esp21', 'measured_distance_esp22', 'labelx_esp20',
       'labelx_esp21', 'labelx_esp22', 'labely_esp20', 'labely_esp21',
       'labely_esp22'],
      dtype='object')
```

[14]:
```python
flatp = flatp.drop(["labelx_esp20", "labelx_esp21", "labelx_esp22",␣
 ↪"labely_esp20", "labely_esp21", "labely_esp22"], axis=1)
```

[15]:
```python
flatp = flatp.dropna()
```

[16]:
```python
flatp
```

[16]:
```
   measure_no_  rssi_esp20  rssi_esp21  rssi_esp22  angle_esp20  angle_esp21  \
0            1  -88.166667  -84.666667  -94.555556  -158.659735   -65.245624
2            3  -84.122449  -99.666667  -75.450000    91.435699     0.739740
3            4  -70.346154  -96.000000  -87.200000    -6.740660   -24.262525
4            5  -83.894737  -87.409091  -87.380952   130.664663   -13.694973
5            6  -98.272727  -91.530201  -98.000000  -172.272788   -42.783728
6            7  -96.700000  -91.757143  -96.333333  -172.272788   -42.783728

   angle_esp22  measured_distance_esp20  measured_distance_esp21  \
0  -136.981118                 4.757198                 5.848593
2   168.724246                 3.668797                 6.788644
3   -84.599287                 1.450394                 9.126491
4  -162.111911                 3.173801                 4.952658
5  -134.544257                 2.650715                 5.795371
6  -134.544257                 2.650715                 5.795371

   measured_distance_esp22
0                 7.565046
2                 1.215382
3                 3.616294
4                 3.329080
5                 5.312698
6                 5.312698
```

[17]:

```
y_columns = ["measured_distance_esp20", "measured_distance_esp21",␣
 ↪"measured_distance_esp22"]
removed = ["measure_no_", "angle_esp20","angle_esp21","angle_esp22"]
```

[18]:

```
# creating the X numpy

keepX = filter(lambda x:not x in y_columns,  flatp.columns.values)
keepX = filter(lambda x:not x in removed,  keepX)

keepX = list(keepX)

print(keepX)

print(type(keepX))
print(type(y_columns))


X = flatp[keepX].copy()
MX = X.to_numpy()
Y = flatp[y_columns].copy()
MY = Y.to_numpy()
```

```
['rssi_esp20', 'rssi_esp21', 'rssi_esp22']
<class 'list'>
<class 'list'>
```

[19]:

```
X
```

[19]:

```
   rssi_esp20  rssi_esp21  rssi_esp22
0  -88.166667  -84.666667  -94.555556
2  -84.122449  -99.666667  -75.450000
3  -70.346154  -96.000000  -87.200000
4  -83.894737  -87.409091  -87.380952
5  -98.272727  -91.530201  -98.000000
6  -96.700000  -91.757143  -96.333333
```

[20]:

```
Y
```

[20]:

```
   measured_distance_esp20  measured_distance_esp21  measured_distance_esp22
0                 4.757198                 5.848593                 7.565046
2                 3.668797                 6.788644                 1.215382
3                 1.450394                 9.126491                 3.616294
4                 3.173801                 4.952658                 3.329080
5                 2.650715                 5.795371                 5.312698
6                 2.650715                 5.795371                 5.312698
```

[21]:

```
MX
```

[21]:
```
array([[-88.16666667, -84.66666667, -94.55555556],
       [-84.12244898, -99.66666667, -75.45      ],
       [-70.34615385, -96.        , -87.2       ],
       [-83.89473684, -87.40909091, -87.38095238],
       [-98.27272727, -91.53020134, -98.        ],
       [-96.7       , -91.75714286, -96.33333333]])
```

[22]:
```
MY
```

[22]:
```
array([[4.75719801, 5.84859252, 7.5650465 ],
       [3.66879675, 6.78864392, 1.2153818 ],
       [1.45039386, 9.12649067, 3.6162942 ],
       [3.17380147, 4.95265809, 3.32907994],
       [2.65071496, 5.79537114, 5.31269791],
       [2.65071496, 5.79537114, 5.31269791]])
```

[ ]:
```

```

[23]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2)
```

[24]:
```
MY
```

[24]:
```
array([[4.75719801, 5.84859252, 7.5650465 ],
       [3.66879675, 6.78864392, 1.2153818 ],
       [1.45039386, 9.12649067, 3.6162942 ],
       [3.17380147, 4.95265809, 3.32907994],
       [2.65071496, 5.79537114, 5.31269791],
       [2.65071496, 5.79537114, 5.31269791]])
```

[25]:
```
from sklearn.linear_model import Ridge
from sklearn.preprocessing import PolynomialFeatures
from sklearn.pipeline import make_pipeline

degre = 2
model = make_pipeline(PolynomialFeatures(degre), Ridge())
print(type(model))
model.fit(MX, MY)
Y_pred = model.predict(MX)
```

```
<class 'sklearn.pipeline.Pipeline'>
```

[26]:
```
Y_pred
```

[26]:
```
array([[4.75668209, 5.84828106, 7.56367942],
       [3.66873494, 6.78860696, 1.21527032],
       [1.4504242 , 9.12650815, 3.61638183],
       [3.1744283 , 4.95303786, 3.33079045],
       [2.65279475, 5.79663074, 5.31909184],
       [2.64855574, 5.79406269, 5.3059844 ]])
```

[27]:

```
# check precision / dispersion

MY - Y_pred
```

[27]:

```
array([[ 5.15925316e-04,  3.11459953e-04,  1.36707632e-03],
       [ 6.18115773e-05,  3.69570232e-05,  1.11475378e-04],
       [-3.03371460e-05, -1.74798583e-05, -8.76253358e-05],
       [-6.26835123e-04, -3.79769585e-04, -1.71051077e-03],
       [-2.07978470e-03, -1.25960940e-03, -6.39392976e-03],
       [ 2.15922008e-03,  1.30844187e-03,  6.71351418e-03]])
```

[28]:

```
model.predict([[-90,-90,-80]])
```

[28]:

```
array([[ 1.1520523 ,  1.83776936, -6.05578789]])
```

[29]:

```
# save model to disk with all devices

import pickle

filename = 'model_esp20_esp21_esp22.sav'
pickle.dump(model, open(filename, 'wb'))
```

[ ]:

```

```

[30]:

```
# Create model for all combinaisons

def combinaison(a, n):
    ret = []
    toRemove = len(a) - n
    if toRemove > 0:
        for i in range(0,len(a)):
            r = a.copy()
            r.remove(a[i])
            result = combinaison(r, n)
            for j in result:
                if not j in ret:
                    ret.append(j)
    else:
        ret.append(a)
    return ret
```

[31]:

```
alldevices = list(positions_stations.keys())

import joblib
import sklearn
print(sklearn.__version__)

for i in alldevices:
    label = "rssi_" + i
    dfx = g[g["device"] == i]["rssi"]
    dfy = g[g["device"] == i]["measured_distance"]

    fx = dfx.to_numpy().reshape(-1,1)
    fy = dfy.to_numpy().reshape(-1,1)
    print(fx)
    print(fy)
    degre = 2
    model = make_pipeline(PolynomialFeatures(degre), Ridge())
    model.fit(fx, fy)

    Y_pred = model.predict(fx)
    print(Y_pred)
    print("result fit")
    print(Y_pred - fy)
    filename = 'model_' + i + '.sav'
    joblib.dump(model, filename)
```

```
0.24.2
[[-88.16666667]
 [-84.12244898]
 [-70.34615385]
 [-83.89473684]
 [-98.27272727]
 [-96.7       ]]
[[4.75719801]
 [3.66879675]
 [1.45039386]
 [3.17380147]
 [2.65071496]
 [2.65071496]]
[[3.56157393]
 [3.49162853]
 [1.82615776]
 [3.48203505]
 [2.90501473]
 [3.08521003]]
result fit
```

```
[[-1.19562409]
 [-0.17716823]
 [ 0.3757639 ]
 [ 0.30823358]
 [ 0.25429977]
 [ 0.43449507]]
[[-84.66666667]
 [-88.        ]
 [-99.66666667]
 [-96.        ]
 [-87.40909091]
 [-91.53020134]
 [-91.75714286]]
[[5.84859252]
 [4.13014551]
 [6.78864392]
 [9.12649067]
 [4.95265809]
 [5.79537114]
 [5.79537114]]
[[4.78132496]
 [5.39744627]
 [7.77526884]
 [6.99084265]
 [5.28617459]
 [6.08060965]
 [6.12560601]]
result fit
[[-1.06726756]
 [ 1.26730076]
 [ 0.98662492]
 [-2.13564801]
 [ 0.33351649]
 [ 0.28523851]
 [ 0.33023488]]
[[-94.55555556]
 [-75.45      ]
 [-87.2       ]
 [-87.38095238]
 [-98.        ]
 [-96.33333333]]
[[7.5650465 ]
 [1.2153818 ]
 [3.6162942 ]
 [3.32907994]
 [5.31269791]
 [5.31269791]]
[[5.45115263]
```

```
 [1.18401636]
 [3.79367902]
 [3.83423417]
 [6.23361432]
 [5.85450176]]
result fit
[[-2.11389387]
 [-0.03136544]
 [ 0.17738482]
 [ 0.50515423]
 [ 0.92091641]
 [ 0.54180385]]
```

[ ]: