Phone accelerometer

Documentatie

Suport vector machine - Razvan Cazacu

Pentru clasificarea utilizatorilor de telefoane am ales sa folosesc un clasificator SVM cu un parametru de penalitate cu valoarea C=9,1.

Parametrul de penalitate
a fost ales dupa mai multe incercari pentru a evita overfitting, respectiv underfitting.
Kenrel-ul ales pentru SVM este RBF cu gamma='scale'.

Kernel RBF:

$$K(u,v) = exp(-gamma * ||u - v||^2)$$

Datele primite pentru fiecare persoana sunt inregistrate pe parcursul a 1,5 secunde la o frecventa de 100 Hz, astfel rezultant intr-un nr de valori de la 136 pana la 159.
Pentru ca datele au fost inregistrate in acest mod, o modalitate eficienta de a le prelucra ar fi fost aplicarea unui FFT (Fast Fourier transform).

Pentru a antrena SVM-ul am folosit mai multe metode, iar pentru cazul prezentat aici, am folosit raw data la care am adaugat un feature [average(col_x), average(col_y),average(col_z)] pentru fiecare linie care nu avea suficiente feature-uri pana la 150, iar pentru cele care depaseau 150, nu am utilizat datele respective.

Phone accelerometer

Documentatie

Code

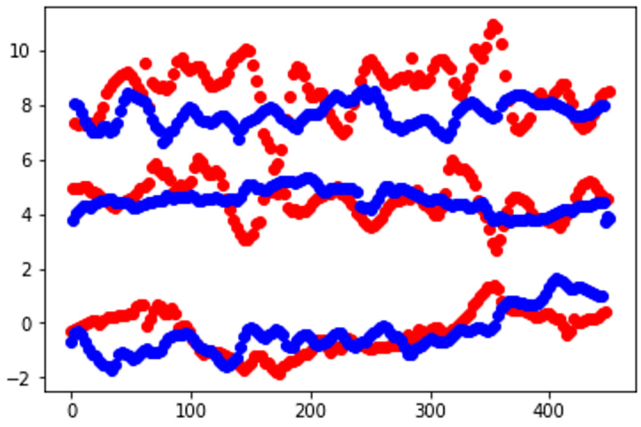
Am folosit pandas si numpy pentru a stoca datele date :

	0	1
0	id	class
1	10003	7
2	10005	13
3	10006	6
4	10007	1

Labels training data

```
[[-0.767342    4.330513    7.996035   ...  0.52852     4.182671
   8.673594  ]
 [-1.105523    4.031836    8.904634   ...  3.943649    3.93926
   3.93028133]
 [-0.226252    3.438672    8.314464   ... -2.042853    4.103063
   7.870339  ]
```

Training data



Ilustrarea datelor pentru 2 utilizatori (x, y, z)

```
train_data_lines = []
for root, dirs, files in os.walk(dataPath +"train/
    for x in files:
        train_data = np.array(pd.read_csv(dataPath
        avgx = train_data[:][0].mean()
        avgy = train_data[:][1].mean()
        avgz = train_data[:][2].mean()
        for i in range(train_data.shape[0],150):
            train_data = np.concatenate(
                (train_data,
                 [[avgx,avgy,avgz]]
                ),axis=0)
        train_data = train_data.flatten()
        train_data_lines.append(train_data[:450])
train_data_lines = np.array(train_data_lines)
plt.plot(train_data_lines[0],"ro")
plt.plot(train_data_lines[1],"bo")
```

Codul respectiv

Am prelucrat datele folosind StandardScaler pentru restrangerea datelor intr-un interval si a imbunatati acuratetea, iar dupa aceea am antrenat SVM-ul pe datele de traning (9000, 450) si am dat predict pe cele de test (5000, 450).

Pentru a nu lucra orbeste si sa nu fac overfitting, mi-am impartit datele de antrenare in 80% training si 20% test pentru a observa comportamentul SVM-ului.

Am folosit si KFold pentru a imparti datele de antrenare, dar in final am ramas la a utiliza 80% - 20%

```
from sklearn.model_selection import KFold
kf = KFold(5,True,1)
# kf.get_n_splits(scaled_train_data)
#cross validation
for train_index, test_index in kf.split(scaled_train_data):
    print("TRAIN:", train_index, "TEST:", test_index)
    X_train, X_test = scaled_train_data[train_index], scaled_train_data[test_index]
    y_train, y_test = np_train_labels[train_index,[1]], np_train_labels[test_index,[1]]
```

Phone accelerometer

Documentatie

Code

```
[[ 70   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   6   0   0]
 [  0  96   0   0   0   0   0   0   0   0   0   2   0   0   0   0   0   0   0   0]
 [  0   0  89   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0  65   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0  79   0   0   0   5   0   3   0   1   3   0   3   0   0   0   3]
 [  0   0   0   0   0  89   0   0   0   0   0   0   0   0   0   0   0   0   1   0]
 [  0   1   0   0   0   0  75   0   0   0   0   2   0   0   0   0   1   0   0   0]
 [  0   0   0   0   0   0   0  86   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   2   0   0   0  85   0   1   0   0   1   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0  96   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   2   0   0   0   6   0  81   0   0   5   0   0   0   0   0   1]
 [  0   8   0   0   0   0   3   0   0   0   0  72   0   0   3   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   2  84   0   0   0   0   0   0]
 [  0   0   0   0   4   0   0   0   0   0   4   0   0  90   0   1   0   0   0   3]
 [  0   2   0   0   0   0   5   0   0   0   0   6   1   0  76   0   0   0   0   0]
 [  0   0   0   0   5   0   0   0   0   0   1   0   1   1   0  99   0   0   0   4]
 [  0   1   0   0   0   0   1   0   0   0   0   1   0   0   1   0  76   0   0   0]
 [  3   0   0   0   0   0   2   0   0   1   0   0   0   0   0   0   1 102   0   0]
 [  0   0   0   0   0   6   0   0   0   0   0   0   0   0   0   0   0   0  87   0]
 [  0   0   0   0   3   0   0   0   0   0   0   0   0   1   0   6   0   0   0  71]]
9  | train acc =  0.9981944444444445  | test acc =  0.926666666666
[[ 70   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   6   0   0]
 [  0  96   0   0   0   0   0   0   0   0   0   2   0   0   0   0   0   0   0   0]
 [  0   0  89   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0  65   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0  79   0   0   0   5   0   3   0   1   3   0   3   0   0   0   3]
 [  0   0   0   0   0  89   0   0   0   0   0   0   0   0   0   0   0   0   1   0]
 [  0   1   0   0   0   0  75   0   0   0   0   2   0   0   0   0   1   0   0   0]
 [  0   0   0   0   0   0   0  86   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   2   0   0   0  85   0   1   0   0   1   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0  96   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   2   0   0   0   6   0  81   0   0   5   0   0   0   0   0   1]
 [  0   8   0   0   0   0   3   0   0   0   0  72   0   0   3   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   2  84   0   0   0   0   0   0]
 [  0   0   0   0   4   0   0   0   0   0   4   0   0  90   0   1   0   0   0   3]
 [  0   2   0   0   0   0   5   0   0   0   0   6   1   0  76   0   0   0   0   0]
 [  0   0   0   0   5   0   0   0   0   0   1   0   1   1   0  99   0   0   0   4]
 [  0   1   0   0   0   0   1   0   0   0   0   1   0   0   1   0  76   0   0   0]
 [  3   0   0   0   0   0   2   0   0   1   0   0   0   0   0   0   1 102   0   0]
 [  0   0   0   0   0   6   0   0   0   0   0   0   0   0   0   0   0   0  87   0]
 [  0   0   0   0   3   0   0   0   0   0   0   0   0   1   0   6   0   0   0  71]]
9.1  | train acc =  0.9983333333333333  | test acc =  0.927222222222222
[0.9981944444444445, 0.9981944444444445, 0.9983333333333333, 0.9983333333333333, 0.9983333333333333, 0.9983333333333333]
```

```
[[ 70   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   6   0   0]
 [  0  96   0   0   0   0   0   0   0   0   0   2   0   0   0   0   0   0   0   0]
 [  0   0  89   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0  65   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0  79   0   0   0   5   0   3   0   1   3   0   3   0   0   0   3]
 [  0   0   0   0   0  89   0   0   0   0   0   0   0   0   0   0   0   0   1   0]
 [  0   1   0   0   0   0  75   0   0   0   0   2   0   0   0   0   1   0   0   0]
 [  0   0   0   0   0   0   0  86   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   2   0   0   0  85   0   1   0   0   1   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0  96   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   2   0   0   0   6   0  81   0   0   5   0   0   0   0   0   1]
 [  0   8   0   0   0   0   3   0   0   0   0  72   0   0   3   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   2  84   0   0   0   0   0   0]
 [  0   0   0   0   4   0   0   0   0   0   4   0   0  90   0   1   0   0   0   3]
 [  0   2   0   0   0   0   5   0   0   0   0   6   1   0  76   0   0   0   0   0]
 [  0   0   0   0   5   0   0   0   0   0   1   0   1   1   0  99   0   0   0   4]
 [  0   1   0   0   0   0   1   0   0   0   0   1   0   0   0   0  77   0   0   0]
 [  3   0   0   0   0   0   2   0   0   1   0   0   0   0   0   0   1 102   0   0]
 [  0   0   0   0   0   6   0   0   0   0   0   0   0   0   0   0   0   0  87   0]
 [  0   0   0   0   2   0   0   0   0   0   0   0   0   2   0   6   0   0   0  71]]
9.3  | train acc =  0.9983333333333333  | test acc =  0.92722222222
```

Pentru a observa comportamentul diferit am mai folosit date prelucrate diferit: raw data umplute cu 0-uri, cu media facuta pe ultimele 2 elemente, am incercat sa imi creez mai multe feature-uri folosind functii matematice de statistica aplicate pe coloanele x, y, z (ex: mean, stdev, variance etc.).