Atividade SVM

December 16, 2020

1 Parte 1

Observe o último exercício para verificar como ler e manipular o MNIST dataset.

1.0.1 Exercicio 1.1

Os classificadores SVM são binários e só conseguem prever entre duas classes 0 e 1. Por favor, manipule o conjunto de treinamento e teste para que haja apenas elementos cujo digito seja `0`e `1`

[44]:

```
import numpy as np
from sklearn.datasets import fetch_openml
mnist = fetch_openml('mnist_784', version=1)
mnist.keys()
```

[44]:

```
dict_keys(['data', 'target', 'frame', 'categories', 'feature_names',
'target_names', 'DESCR', 'details', 'url'])
```

Os labels devem ser da forma (Não necessariamente nessa ordem) `mnist.target[:10] array(['0', '1', '1', '1', '0', ...`

Ou seja, remova de `mnist.data` e `mnist.target` todos os elementos que não sejam `0` e `1` Nota: Cuidado para não ter labels e imagens com referencias incorretas. Toda imagem de um caractere `0` deve corresponder ao `mnist.target` com valor `0`. Por exemplo, se você remover o primeiro item do `mnist.target` deve remover também de `mnist.data` e assim por diante.

[45]:

```
# Seu código
import pandas as pd
df = pd.DataFrame(data= np.c_[mnist['data'], mnist['target']],
                  columns= mnist['feature_names'] + ['target'])
```

[46]:

```
df.head()
```

[46]:

```
  pixel1 pixel2 pixel3 pixel4 pixel5 pixel6 pixel7 pixel8 pixel9 pixel10  …  \
0      0      0      0      0      0      0      0      0      0       0  …
1      0      0      0      0      0      0      0      0      0       0  …
2      0      0      0      0      0      0      0      0      0       0  …
3      0      0      0      0      0      0      0      0      0       0  …
4      0      0      0      0      0      0      0      0      0       0  …
```

```
  pixel776 pixel777 pixel778 pixel779 pixel780 pixel781 pixel782 pixel783  \
0        0        0        0        0        0        0        0        0
1        0        0        0        0        0        0        0        0
2        0        0        0        0        0        0        0        0
3        0        0        0        0        0        0        0        0
4        0        0        0        0        0        0        0        0

  pixel784 target
0        0      5
1        0      0
2        0      4
3        0      1
4        0      9

[5 rows x 785 columns]
```

[47]:
```
df_2 = df[(df['target']=='1') | (df['target'] == '0')]
```

[48]:
```
df_2.head()
```

[48]:
```
   pixel1 pixel2 pixel3 pixel4 pixel5 pixel6 pixel7 pixel8 pixel9 pixel10  \
1       0      0      0      0      0      0      0      0      0       0
3       0      0      0      0      0      0      0      0      0       0
6       0      0      0      0      0      0      0      0      0       0
8       0      0      0      0      0      0      0      0      0       0
14      0      0      0      0      0      0      0      0      0       0

    … pixel776 pixel777 pixel778 pixel779 pixel780 pixel781 pixel782  \
1   …        0        0        0        0        0        0        0
3   …        0        0        0        0        0        0        0
6   …        0        0        0        0        0        0        0
8   …        0        0        0        0        0        0        0
14  …        0        0        0        0        0        0        0

   pixel783 pixel784 target
1         0        0      0
3         0        0      1
6         0        0      1
8         0        0      1
14        0        0      1

[5 rows x 785 columns]
```

[49]:
```
df_2np = df_2.to_numpy()
```

[53]:
```
df_2np
```

```
[53]: array([[0.0, 0.0, 0.0, …, 0.0, 0.0, '0'],
              [0.0, 0.0, 0.0, …, 0.0, 0.0, '1'],
              [0.0, 0.0, 0.0, …, 0.0, 0.0, '1'],
              …,
              [0.0, 0.0, 0.0, …, 0.0, 0.0, '1'],
              [0.0, 0.0, 0.0, …, 0.0, 0.0, '0'],
              [0.0, 0.0, 0.0, …, 0.0, 0.0, '1']], dtype=object)
```

1.0.2 Exercicio 1.2

Crie um classificador SVM para o MNIST dataset. Utilize o `sklearn` para tal atividade.

[54]:

```
# Seu codigo
from sklearn.model_selection import train_test_split
X = df_2.drop(columns = ['target'])
y = df_2['target']

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

[56]:

```
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
```

[57]:

```
clf = make_pipeline(StandardScaler(), SVC(gamma='auto'))
clf.fit(X_train, y_train)
```

```
[57]: Pipeline(steps=[('standardscaler', StandardScaler()),
                      ('svc', SVC(gamma='auto'))])
```

[59]:

```
clf.score(X_test, y_test)
```

```
[59]: 0.9942489851150202
```

1.0.3 Exercício 1.3

Treine um classificador SVM no conjunto de dados MNIST. Nesse caso com o dataset inteiro. Como os classificadores SVM são binários, você precisará usar uma abordagem “um contra todos” para classificar todos os 10 dígitos.

Referência: https://scikit-learn.org/stable/modules/svm.html

[69]:

```
from sklearn import svm
clf_multi = svm.SVC(decision_function_shape='ovr')
```

[70]:

```
clf_multi.fit(X_train, y_train)
```

```
[70]: SVC()
```

[71]:
```
clf_multi.score(X_test, y_test)
```

[71]:
```
0.9989851150202977
```

2 Parte 2

Utilize o California Housing dataset (https://github.com/ageron/handson-ml/tree/master/datasets/housing) e crie um Regressor SVM. Por favor prevejam o `median_house_value` (target ou label).

2.0.1 Codigo para baixar o dataset

[85]:
```
import os
import tarfile
from six.moves import urllib

DOWNLOAD_ROOT = "https://raw.githubusercontent.com/ageron/handson-ml/master/"
HOUSING_PATH = os.path.join("datasets", "housing")
HOUSING_URL = DOWNLOAD_ROOT + "datasets/housing/housing.tgz"

def fetch_housing_data(housing_url=HOUSING_URL, housing_path=HOUSING_PATH):
    if not os.path.isdir(housing_path):
        os.makedirs(housing_path)
    tgz_path = os.path.join(housing_path, "housing.tgz")
    urllib.request.urlretrieve(housing_url, tgz_path)
    housing_tgz = tarfile.open(tgz_path)
    housing_tgz.extractall(path=housing_path)
    housing_tgz.close()
```

[86]:
```
import pandas as pd

def load_housing_data(housing_path=HOUSING_PATH):
    csv_path = os.path.join(housing_path, "housing.csv")
    return pd.read_csv(csv_path)
```

[93]:
```
fetch_housing_data()
dataset = load_housing_data()
```

[94]:
```
dataset
```

[94]:
```
       longitude  latitude  housing_median_age  total_rooms  total_bedrooms  \
0        -122.23     37.88                41.0        880.0           129.0
1        -122.22     37.86                21.0       7099.0          1106.0
2        -122.24     37.85                52.0       1467.0           190.0
3        -122.25     37.85                52.0       1274.0           235.0
4        -122.25     37.85                52.0       1627.0           280.0
```

```
…          …       …                 …          …             …
20635    -121.09     39.48                25.0       1665.0           374.0
20636    -121.21     39.49                18.0        697.0           150.0
20637    -121.22     39.43                17.0       2254.0           485.0
20638    -121.32     39.43                18.0       1860.0           409.0
20639    -121.24     39.37                16.0       2785.0           616.0

       population  households  median_income  median_house_value  \
0            322.0       126.0         8.3252            452600.0
1           2401.0      1138.0         8.3014            358500.0
2            496.0       177.0         7.2574            352100.0
3            558.0       219.0         5.6431            341300.0
4            565.0       259.0         3.8462            342200.0
…            …         …            …                 …
20635        845.0       330.0         1.5603             78100.0
20636        356.0       114.0         2.5568             77100.0
20637       1007.0       433.0         1.7000             92300.0
20638        741.0       349.0         1.8672             84700.0
20639       1387.0       530.0         2.3886             89400.0

      ocean_proximity
0            NEAR BAY
1            NEAR BAY
2            NEAR BAY
3            NEAR BAY
4            NEAR BAY
…               …
20635          INLAND
20636          INLAND
20637          INLAND
20638          INLAND
20639          INLAND

[20640 rows x 10 columns]
```

2.0.2 Compare a acuracia do seu classificador com um regressor linear.

[95]:

```
# Seu codigo aqui
dataset = dataset.drop(columns = ['ocean_proximity'])
dataset = dataset.fillna(dataset.mean())
X = dataset.drop(columns = ['median_house_value'])
y = dataset['median_house_value']
```

[96]:

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

[97]:

```
clf = make_pipeline(StandardScaler(), SVC(gamma='auto'))
clf.fit(X_train, y_train)
```

[97]:
```
Pipeline(steps=[('standardscaler', StandardScaler()),
                ('svc', SVC(gamma='auto'))])
```

[98]:
```
clf.score(X_test, y_test)
```

[98]:
```
0.05111434108527132
```

Regressão Linear

[102]:
```
from sklearn import linear_model
```

[104]:
```
lr = linear_model.LinearRegression()
lr.fit(X_train, y_train)
```

[104]:
```
LinearRegression()
```

[110]:
```
lr.score(X_test, y_test)
```

[110]:
```
0.6304141760576532
```