MO431T4

April 29, 2021

1 Exercício 04

April 29, 2021

1.1 Autores:

- Fábio Akahoshi Collado RA: 001658
- Osmir Bresciani Filho RA: 225412
- Victor Antonio Menuzzo RA: 234996

[1]:

```
import numpy as np
from sklearn import svm
```

[2]:

```
X = np.load('X.npy')
y = np.load('y.npy')
```

2 Implementar SVM

[3]:

```
def eval_SVM (X_train, y_train, X_test, y_test, gamma, C, epsilon):
    _svm = svm.SVR(gamma=gamma, C=C, epsilon=epsilon)
    _svm.fit(X_train, y_train)
    y_predicted = _svm.predict(X_test)
    return (np.sum(np.square(y_test-y_predicted))/y_test.shape[0])**(1/2)
```

3 Implementar medida de erro com 5-fold cross-validation

[4]:

```
def evaluate_k_fold(X, y, gamma, C, epsilon):
    div = X.shape[0] // 5
    X1 = X[div*0:div*1]
    X2 = X[div*1:div*2]
    X3 = X[div*2:div*3]
    X4 = X[div*3:div*4]
    X5 = X[div*4:]
    y1 = y[div*0:div*1]
    y2 = y[div*1:div*2]
    y3 = y[div*2:div*3]
    y4 = y[div*3:div*4]
    y5 = y[div*4:]
    p = eval_SVM (np.concatenate((X1, X2, X3, X4)), np.concatenate((y1, y2, y3,␣
↪y4)), X5, y5, gamma, C, epsilon)
    p = p + eval_SVM (np.concatenate((X5, X1, X2, X3)), np.concatenate((y5, y1,␣
↪y2, y3)), X4, y4, gamma, C, epsilon)
    p = p + eval_SVM (np.concatenate((X4, X5, X1, X2)), np.concatenate((y4, y5,␣
↪y1, y2)), X3, y3, gamma, C, epsilon)
    p = p + eval_SVM (np.concatenate((X3, X4, X5, X1)), np.concatenate((y3, y4,␣
↪y5, y1)), X2, y2, gamma, C, epsilon)
    p = p + eval_SVM (np.concatenate((X2, X3, X4, X5)), np.concatenate((y2, y3,␣
↪y4, y5)), X1, y1, gamma, C, epsilon)
    return p / 5
```

4 1) Random search

[5]:

```
%%time
from sklearn.utils.fixes import loguniform
from scipy.stats import uniform
from sklearn.model_selection import RandomizedSearchCV

param_distributions ={'C': loguniform(2**-5, 2**15),
                      'gamma': loguniform(2**-15, 2**3),
                      'epsilon': uniform(0.05, 1)}

optimizer = RandomizedSearchCV(svm.SVR(), param_distributions, random_state=1,␣
↪n_jobs=-1, n_iter=125, cv=5, scoring='neg_mean_squared_error')
result = optimizer.fit(X,y).best_estimator_
epsilon = result.epsilon; C = result.C; gamma = result.gamma
print('gamma = ', gamma, ';  C = ', C, '; epsilon = ', epsilon)
print('RMSE =', evaluate_k_fold(X, y, gamma, C, epsilon))
```

```
gamma =  3.1630280744328535e-05 ;  C =  8584.9285467854 ; epsilon =
0.6236794866722859
RMSE = 3.822453732348343
Wall time: 12.9 s
```

5 2) Grid search

[6]:

```
%%time
from sklearn.model_selection import GridSearchCV
param_distributions ={'C': [2**-5, 2**0, 2**5, 2**10, 2**15],
                      'gamma': [2**-15, 2**-10.5, 2**-6, 2**-1.5, 2**3],
                      'epsilon': [0.05, 0.2875, 0.525, 0.7625, 1]}

optimizer = GridSearchCV(svm.SVR(), param_distributions, cv=5)
```

```
result = optimizer.fit(X,y).best_estimator_
epsilon = result.epsilon; C = result.C; gamma = result.gamma
print('gamma = ', gamma, ';  C = ', C, '; epsilon = ', epsilon)
print('RMSE =', evaluate_k_fold(X, y, gamma, C, epsilon))
```

```
gamma =  3.0517578125e-05 ;  C =  32768 ; epsilon =  0.05
RMSE = 3.7972435612961926
Wall time: 2min 37s
```

6 3) Otimização bayesiana

[7]:

```
%%time
#!pip install scikit-optimize
from skopt import BayesSearchCV
from skopt.space import Real
param_distributions ={'C': Real(2**-5, 2**15, 'log-uniform'),
                      'gamma': Real(2**-15, 2**3, 'log-uniform'),
                      'epsilon': Real(0.05, 1, 'uniform')}

optimizer = BayesSearchCV(svm.SVR(), param_distributions, cv=5, n_iter=125)
result = optimizer.fit(X,y).best_estimator_
epsilon = result.epsilon; C = result.C; gamma = result.gamma
print('gamma = ', gamma, ';  C = ', C, '; epsilon = ', epsilon)
print('RMSE =', evaluate_k_fold(X, y, gamma, C, epsilon))
#https://scikit-optimize.github.io/stable/modules/generated/skopt.BayesSearchCV.
 ↪html#skopt.BayesSearchCV
#https://scikit-optimize.github.io/stable/auto_examples/
 ↪sklearn-gridsearchcv-replacement.html
```

```
C:\ProgramData\Anaconda3\lib\site-packages\skopt\optimizer\optimizer.py:449:
UserWarning: The objective has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
```

```
gamma =  3.0517578125e-05 ;  C =  22214.747615223212 ; epsilon =  0.05
RMSE = 3.7240764575932213
Wall time: 17min 18s
```

7 4) PSO

[8]:

```
%%time
#!pip install pyswarm
from sklearn.model_selection import cross_val_score
from pyswarm import pso

def objective_function(params):
    gamma, C, epsilon = params
    C, gamma, epsilon = 2**C, 2**gamma, epsilon
    clf = svm.SVR(C=C, gamma=gamma , epsilon=epsilon )
    return -cross_val_score(clf, X, y).mean()

ub = [3, 15, 1]
lb = [-15, -5, 0.05]
xopt, fopt = pso(objective_function, lb, ub, swarmsize = 11, maxiter = 11)

gamma = xopt[0]; C = xopt[1]; epsilon = xopt[2]
C, gamma, epsilon = 2**C, 2**gamma, epsilon
print('gamma = ', gamma, ';  C = ', C, '; epsilon = ', epsilon)
print('RMSE =', evaluate_k_fold(X, y, gamma, C, epsilon))
```

```
Stopping search: maximum iterations reached --> 11
gamma =  3.0517578125e-05 ;  C =  25101.027905389386 ; epsilon =
0.2035920874934088
RMSE = 3.7456745982888897
Wall time: 13min 16s
```

8 5) Simulated Annealing

[9]:

```
%%time
#!pip install simanneal
from simanneal import Annealer
from sklearn.model_selection import cross_val_score
import random

def objective_function(params):
    C, gamma, epsilon = params
    clf = svm.SVR(C=C, gamma=gamma, epsilon=epsilon)
    return -cross_val_score(clf, X, y).mean()

class SimulatedAnnealing(Annealer):
    def move(self):
        self.state[0]=2**np.random.uniform(low = -5, high = 15) #C
        self.state[1]=2**np.random.uniform(low = -15, high = 3) #gamma
        self.state[2]=np.random.uniform(low = 0.05, high = 1) #epsilon

    def energy(self):
        C = self.state[0]
        gamma = self.state[1]
        epsilon = self.state[2]
        X = [C, gamma, epsilon]
        return objective_function(X)

initial_state = [0.5, 0.5, 0.5]
annealing = SimulatedAnnealing(initial_state)
annealing.steps = 125
```

```
pos, cost = annealing.anneal()
C = pos[0]; gamma = pos[1]; epsilon = pos[2]
print('C = ', C, ';  gamma = ', gamma, '; epsilon = ', epsilon)
print('RMSE =', evaluate_k_fold(X, y, C=C, gamma=gamma, epsilon=epsilon))
```

```
 Temperature        Energy    Accept   Improve     Elapsed   Remaining
     2.50000         -0.72   100.00%     0.00%     0:01:00     0:00:00
```

```
C =  10301.295809512612 ;  gamma =  3.813353719887769e-05 ; epsilon =
0.3434397416104423
RMSE = 3.827997499778153
Wall time: 1min 2s
```

9 6) CMA-ES

[10]:

```
#!pip install cma
import cma
import random
from sklearn.model_selection import cross_val_score

def objective_function(params):
    gamma, C, epsilon = params
    C, gamma, epsilon = 2**(C*20-5), 2**(gamma*18-15), epsilon*0.95+0.05
    clf = svm.SVR(C=C, gamma=gamma , epsilon=epsilon )
    return -cross_val_score(clf, X, y).mean()

ub = [1, 1, 1]
lb = [0, 0, 0]
es = cma.CMAEvolutionStrategy([random.random(),random.random(),random.
 ↪random()], 0.25, {'bounds': [lb, ub]})

pos = es.optimize(objective_function, iterations=125).result
gamma = pos.xfavorite[0]; C = pos.xfavorite[1]; epsilon = pos.xfavorite[2]
C, gamma, epsilon = 2**(C*20-5), 2**(gamma*18-15), epsilon*0.95+0.05
print('gamma = ', gamma, ';  C = ', C, '; epsilon = ', epsilon)
print('RMSE =', evaluate_k_fold(X, y, gamma, C, epsilon))
```

```
(3_w,7)-aCMA-ES (mu_w=2.3,w_1=58%) in dimension 3 (seed=826405, Thu Apr 29
12:59:15 2021)
Iterat #Fevals   function value  axis ratio  sigma  min&max std  t[m:s]
    1      7 -2.643307820640915e-01 1.0e+00 2.40e-01  2e-01  2e-01 0:00.5
    2     14 -4.560196080174256e-01 1.3e+00 2.42e-01  2e-01  3e-01 0:01.0
    3     21 -6.600327340936797e-01 1.5e+00 2.55e-01  2e-01  3e-01 0:01.7
    5     35 -8.147153721383479e-01 1.4e+00 3.13e-01  3e-01  3e-01 0:16.1
    6     42 -8.206633587308796e-01 1.7e+00 3.30e-01  3e-01  3e-01 0:39.6
    7     49 -6.927143859925462e-01 1.7e+00 2.68e-01  2e-01  2e-01 0:50.5
    9     63 -7.606490501567823e-01 1.6e+00 1.97e-01  1e-01  2e-01 1:25.0
   10     70 -8.136879282292941e-01 1.7e+00 2.30e-01  2e-01  2e-01 1:50.1
```

```
 11      77 -8.120251687036664e-01 2.0e+00 2.13e-01  1e-01  2e-01 2:05.8
 12      84 -7.704118101248827e-01 1.9e+00 1.84e-01  1e-01  2e-01 2:22.3
 13      91 -8.138045200484394e-01 1.8e+00 1.67e-01  9e-02  1e-01 2:38.4
 14      98 -8.196918148717357e-01 2.2e+00 1.65e-01  9e-02  1e-01 2:52.8
 15     105 -8.133226074320280e-01 2.0e+00 1.90e-01  9e-02  2e-01 3:18.6
 16     112 -8.215571784878133e-01 2.6e+00 1.59e-01  6e-02  1e-01 3:51.7
 17     119 -8.208463231815267e-01 2.8e+00 1.27e-01  4e-02  1e-01 4:38.2
 18     126 -8.205687290587594e-01 3.0e+00 1.01e-01  3e-02  8e-02 5:17.5
 19     133 -8.220271355656500e-01 3.5e+00 9.70e-02  2e-02  8e-02 5:49.8
 20     140 -8.208748961474471e-01 4.2e+00 8.77e-02  2e-02  7e-02 6:21.1
 22     154 -8.227172329666128e-01 4.4e+00 7.43e-02  1e-02  6e-02 6:58.1
 23     161 -8.232423123769967e-01 5.2e+00 7.86e-02  1e-02  6e-02 7:18.1
 24     168 -8.238795442462026e-01 4.8e+00 7.74e-02  1e-02  6e-02 7:38.9
 26     182 -8.238540720514174e-01 4.6e+00 6.50e-02  1e-02  5e-02 8:15.4
 28     196 -8.239462851113656e-01 4.6e+00 4.37e-02  7e-03  3e-02 8:59.0
 30     210 -8.238308095524574e-01 3.7e+00 3.71e-02  6e-03  2e-02 9:35.4
 32     224 -8.237785970852698e-01 3.3e+00 2.84e-02  5e-03  1e-02 10:09.9
 34     238 -8.238803872987652e-01 3.8e+00 1.77e-02  3e-03  9e-03 10:41.3
 36     252 -8.239653268996010e-01 4.0e+00 1.89e-02  2e-03  1e-02 11:14.9
 38     266 -8.239197602781733e-01 3.6e+00 1.37e-02  2e-03  5e-03 11:46.9
 40     280 -8.239942081668203e-01 2.4e+00 9.70e-03  1e-03  3e-03 12:20.0
 42     294 -8.239378667870231e-01 2.8e+00 9.28e-03  9e-04  3e-03 12:54.5
 44     308 -8.239651022917334e-01 3.9e+00 1.03e-02  9e-04  3e-03 13:27.7
 46     322 -8.239624596220925e-01 3.5e+00 8.71e-03  7e-04  2e-03 14:00.8
 48     336 -8.239640295799789e-01 2.5e+00 7.73e-03  6e-04  2e-03 14:36.1
 50     350 -8.239685050721185e-01 2.8e+00 1.14e-02  9e-04  3e-03 15:09.4
 53     371 -8.239455918660921e-01 4.4e+00 8.71e-03  6e-04  2e-03 15:59.5
 56     392 -8.239667712136850e-01 3.6e+00 7.48e-03  5e-04  1e-03 16:50.0
 59     413 -8.239827863420240e-01 3.3e+00 4.68e-03  3e-04  7e-04 17:36.3
 62     434 -8.239771008080613e-01 3.4e+00 5.23e-03  3e-04  7e-04 18:23.5
 65     455 -8.239984431464624e-01 2.9e+00 1.14e-02  7e-04  2e-03 19:09.9
 68     476 -8.239996881332594e-01 3.0e+00 7.24e-03  4e-04  1e-03 19:56.5
 71     497 -8.239852323977432e-01 2.8e+00 5.95e-03  3e-04  7e-04 20:42.0
 74     518 -8.239536663542454e-01 4.0e+00 6.60e-03  4e-04  8e-04 21:29.0
 77     539 -8.239521699486773e-01 3.7e+00 5.26e-03  3e-04  6e-04 22:16.7
 80     560 -8.239845417629171e-01 4.5e+00 5.92e-03  3e-04  7e-04 23:03.1
 83     581 -8.239855975270268e-01 7.8e+00 8.76e-03  6e-04  1e-03 23:51.1
 86     602 -8.239745521911729e-01 9.5e+00 1.16e-02  9e-04  2e-03 24:40.1
 89     623 -8.239691376248613e-01 1.0e+01 6.98e-03  4e-04  8e-04 25:31.1
 92     644 -8.239618648525404e-01 1.0e+01 6.89e-03  3e-04  8e-04 26:18.7
 96     672 -8.239509384863482e-01 9.7e+00 6.18e-03  2e-04  6e-04 27:21.4
 99     693 -8.240054955733852e-01 8.6e+00 4.44e-03  2e-04  4e-04 28:11.9
100     700 -8.239677170898781e-01 9.5e+00 4.61e-03  2e-04  4e-04 28:28.8
104     728 -8.239240196880016e-01 1.2e+01 5.51e-03  2e-04  5e-04 29:35.4
108     756 -8.239542431744808e-01 1.2e+01 6.63e-03  2e-04  4e-04 30:41.5
112     784 -8.239892139132692e-01 1.4e+01 8.09e-03  2e-04  5e-04 31:47.2
116     812 -8.239793826963308e-01 1.4e+01 7.96e-03  2e-04  5e-04 32:53.1
120     840 -8.240049737197713e-01 1.2e+01 4.44e-03  9e-05  2e-04 33:58.4
```

```
  124    868 -8.239939905401078e-01 1.1e+01 2.49e-03  4e-05  9e-05 35:04.5
gamma =  3.052581492016438e-05 ;  C =  12114.254425818588 ; epsilon =
0.7120347915584015
RMSE = 3.7850007323010053
```

[ ]:

```

```