Tarefa 4: Busca dos melhores hiperparâmetros para uma SVM

Equipe:

- Flávia Érika Almeida Giló Azevedo | RA: 162641
- Elian Raquel Laura Riveros | RA: 265685
- Yuliana Guadalupe Apaza Yllachura | RA: 234986

In [1]:

```
import numpy as np
from sklearn.svm import SVR
from sklearn.model_selection import cross_val_score
```

Leitura dos arquivos de treino e teste

In [2]:

```
!wget https://www.ic.unicamp.br/~wainer/cursos/1s2021/431/X.npy
!wget https://www.ic.unicamp.br/~wainer/cursos/1s2021/431/y.npy
```

```
--2021-04-29 12:39:44--  https://www.ic.unicamp.br/~wainer/cursos/1s
2021/431/X.npy
Resolving www.ic.unicamp.br (www.ic.unicamp.br)... 143.106.7.54, 280
1:8a:40c0:cafe::54
Connecting to www.ic.unicamp.br (www.ic.unicamp.br)|143.106.7.54|:44
3... connected.
HTTP request sent, awaiting response... 200 OK
Length: 52752 (52K)
Saving to: ‘X.npy’

X.npy               100%[===================>]  51.52K   148KB/s
in 0.3s

2021-04-29 12:39:45 (148 KB/s) - ‘X.npy’ saved [52752/52752]

--2021-04-29 12:39:45--  https://www.ic.unicamp.br/~wainer/cursos/1s
2021/431/y.npy
Resolving www.ic.unicamp.br (www.ic.unicamp.br)... 143.106.7.54, 280
1:8a:40c0:cafe::54
Connecting to www.ic.unicamp.br (www.ic.unicamp.br)|143.106.7.54|:44
3... connected.
HTTP request sent, awaiting response... 200 OK
Length: 4176 (4.1K)
Saving to: ‘y.npy’

y.npy               100%[===================>]   4.08K  --.-KB/s
in 0s

2021-04-29 12:39:46 (206 MB/s) - ‘y.npy’ saved [4176/4176]
```

In [3]:

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

In [4]:

```
X.shape
```

Out[4]:

```
(506, 13)
```

In [5]:

```
y.shape
```

Out[5]:

```
(506,)
```

In [6]:

```
# Nesta lista vamos salvar os valores obtidos por cada algoritmo
melhores_valores = []
```

Random search

In [7]:

```
from sklearn.model_selection import RandomizedSearchCV
```

In [8]:

```
# Definição do intervalo de busca dos hiperparâmetros
c = 2**np.random.uniform(-5, 15, 125) # uniforme nos expoentes
gamma = 2**np.random.uniform(-15, 3, 125) # uniforme nos expoentes
epsilon = np.random.uniform(0.05, 1, 125) # uniforme no intervalo

# Dicionário que será passado para a função de busca pelos melhores hiperparâmet
ros
intervalo_hiperparametros = {'C': c, 'gamma': gamma, 'epsilon': epsilon}
```

In [9]:

```
# Busca pelos melhores hiperparâmetros
busca_hiperparametros = RandomizedSearchCV(estimator = SVR(kernel='rbf'), \
                                           param_distributions = intervalo_hiper
parametros, \
                                           n_iter = 125, \
                                           scoring = 'neg_root_mean_squared_erro
r', \
                                           cv = 5)

melhores_hiperparametros = busca_hiperparametros.fit(X, y)

melhor_c = melhores_hiperparametros.best_params_['C']
melhor_gamma = melhores_hiperparametros.best_params_['gamma']
melhor_epsilon = melhores_hiperparametros.best_params_['epsilon']
rmse_busca_hiperparametros = melhores_hiperparametros.best_score_
melhores_valores.append([melhor_c, melhor_gamma, melhor_epsilon, -rmse_busca_hip
erparametros])

# Reporte dos melhores valores de hiperparâmetros obtidos e o RMSE associado
print("Random search - Melhores valores dos hiperparâmetros e RMSE")
print(f"C: {melhor_c}")
print(f"Gamma: {melhor_gamma}")
print(f"Epsilon: {melhor_epsilon}")
print(f"RMSE do 5-fold cross-validation para o melhor conjunto de hiperparâmetro
s: {-rmse_busca_hiperparametros}")
```

```
Random search - Melhores valores dos hiperparâmetros e RMSE
C: 673.9899526046084
Gamma: 0.0001271316998152192
Epsilon: 0.8686368520247699
RMSE do 5-fold cross-validation para o melhor conjunto de hiperparâm
etros: 4.581288084986051
```

Grid search

In [10]:

```
from sklearn.model_selection import GridSearchCV
```

In [11]:

```
# Definição do intervalo de busca dos hiperparâmetros
c = 2**np.random.uniform(-5, 15, 5) # uniforme nos expoentes
gamma = 2**np.random.uniform(-15, 3, 5) # uniforme nos expoentes
epsilon = np.random.uniform(0.05, 1, 5) # uniforme no intervalo

# Dicionário que será passado para a função de busca pelos melhores hiperparâmet
ros
intervalo_hiperparametros = {'C': c, 'gamma': gamma, 'epsilon': epsilon}
```

In [12]:

```
# Busca pelos melhores hiperparâmetros
busca_hiperparametros = GridSearchCV(estimator = SVR(kernel='rbf'), \
                        param_grid = intervalo_hiperparametros, \
                        scoring = 'neg_root_mean_squared_error', \
                        cv = 5)

melhores_hiperparametros = busca_hiperparametros.fit(X, y)

melhor_c = melhores_hiperparametros.best_params_['C']
melhor_gamma = melhores_hiperparametros.best_params_['gamma']
melhor_epsilon = melhores_hiperparametros.best_params_['epsilon']
rmse_busca_hiperparametros = melhores_hiperparametros.best_score_
melhores_valores.append([melhor_c, melhor_gamma, melhor_epsilon, -rmse_busca_hip
erparametros])

# Reporte dos melhores valores de hiperparâmetros obtidos e o RMSE associado
print("Grid search - Melhores valores dos hiperparâmetros e RMSE")
print(f"C: {melhor_c}")
print(f"Gamma: {melhor_gamma}")
print(f"Epsilon: {melhor_epsilon}")
print(f"RMSE do 5-fold cross-validation para o melhor conjunto de hiperparâmetro
s: {-rmse_busca_hiperparametros}")
```

```
Grid search - Melhores valores dos hiperparâmetros e RMSE
C: 2199.031573303586
Gamma: 3.393197597912491e-05
Epsilon: 0.2489651808832541
RMSE do 5-fold cross-validation para o melhor conjunto de hiperparâm
etros: 4.323574850823
```

Otimização bayesiana usando BayesSearchCV do pacote scikit-optimize

In [13]:

```
!pip install scikit-optimize
```

```
Requirement already satisfied: scikit-optimize in /usr/local/lib/pyt
hon3.7/dist-packages (0.8.1)
Requirement already satisfied: scikit-learn>=0.20.0 in /usr/local/li
b/python3.7/dist-packages (from scikit-optimize) (0.22.2.post1)
Requirement already satisfied: pyaml>=16.9 in /usr/local/lib/python
3.7/dist-packages (from scikit-optimize) (20.4.0)
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python
3.7/dist-packages (from scikit-optimize) (1.0.1)
Requirement already satisfied: scipy>=0.19.1 in /usr/local/lib/pytho
n3.7/dist-packages (from scikit-optimize) (1.4.1)
Requirement already satisfied: numpy>=1.13.3 in /usr/local/lib/pytho
n3.7/dist-packages (from scikit-optimize) (1.19.5)
Requirement already satisfied: PyYAML in /usr/local/lib/python3.7/di
st-packages (from pyaml>=16.9->scikit-optimize) (3.13)
```

In [14]:

```
from skopt import BayesSearchCV
```

In [15]:

```
# Definição do intervalo de busca dos hiperparâmetros
c = 2**np.random.uniform(-5, 15, 125) # uniforme nos expoentes
gamma = 2**np.random.uniform(-15, 3, 125) # uniforme nos expoentes
epsilon = np.random.uniform(0.05, 1, 125) # uniforme no intervalo

# Dicionário que será passado para a função de busca pelos melhores hiperparâmet
ros
intervalo_hiperparametros = {'C': c, 'gamma': gamma, 'epsilon': epsilon}
```

In [16]:

```
# Busca pelos melhores hiperparâmetros
busca_hiperparametros = BayesSearchCV(estimator = SVR(kernel='rbf'), \
                                      search_spaces = intervalo_hiperparametros,
\
                                      n_iter = 125, \
                                      scoring = 'neg_root_mean_squared_error', \
                                      cv = 5)

melhores_hiperparametros = busca_hiperparametros.fit(X, y)

melhor_c = melhores_hiperparametros.best_params_['C']
melhor_gamma = melhores_hiperparametros.best_params_['gamma']
melhor_epsilon = melhores_hiperparametros.best_params_['epsilon']
rmse_busca_hiperparametros = melhores_hiperparametros.best_score_
melhores_valores.append([melhor_c, melhor_gamma, melhor_epsilon, -rmse_busca_hip
erparametros])

# Reporte dos melhores valores de hiperparâmetros obtidos e o RMSE associado
print("Otimização bayesiana - Melhores valores dos hiperparâmetros e RMSE")
print(f"C: {melhor_c}")
print(f"Gamma: {melhor_gamma}")
print(f"Epsilon: {melhor_epsilon}")
print(f"RMSE do 5-fold cross-validation para o melhor conjunto de hiperparâmetro
s: {-rmse_busca_hiperparametros}")
```

```
Otimização bayesiana - Melhores valores dos hiperparâmetros e RMSE
C: 15061.272903853895
Gamma: 3.528585255057165e-05
Epsilon: 0.2853673547828034
RMSE do 5-fold cross-validation para o melhor conjunto de hiperparâm
etros: 3.787595441326054
```

Otimização bayesiana usando Hyperopt

In [17]:

```
from hyperopt import fmin, tpe, hp, Trials
from sklearn.model_selection import train_test_split, KFold, cross_val_score
from  sklearn  import  metrics
import numpy as np
```

In [18]:

```
random_state = 42
num_folds=5

# Split data
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.20, shuffle=
True,
                                                   random_state=random_state
)
kf = KFold(n_splits=num_folds, random_state=42, shuffle=True)

# Create space of hyperparameters
space={'C': hp.loguniform('C', np.log(2**-5),  np.log(2**15)),
       'gamma' : hp.loguniform('gamma', np.log(2**-15),  np.log(2**3)),
       'epsilon': hp.uniform('epsilon',0.05, 1.0)
      }

# Define objective function
def f(params, random_state = random_state, cv=kf, X=X_train, y=y_train):

    # create a SV Regressor
    model = SVR(kernel = 'rbf', **params)

    # and then conduct the cross validation with the folds
    score = -cross_val_score(model, X, y, cv=cv, scoring="neg_mean_squared_erro
r").mean()

    return score


# Optimization
trials = Trials()
best = fmin(fn = f,
            space = space,
            algo = tpe.suggest,
            max_evals = 125,
            trials =  trials,
            rstate = np.random.RandomState(random_state))
```

```
100%|██████████| 125/125 [02:46<00:00,  1.33s/it, best loss: 17.7637
2019993554]
```

In [19]:

```
best
```

Out[19]:

```
{'C': 21094.73842148089,
 'epsilon': 0.7914103617911319,
 'gamma': 3.534421343783635e-05}
```

In [20]:

```
# computing the score on the test set
model = SVR(kernel = 'rbf', C = best['C'],
                      gamma = best['gamma'], epsilon = best['epsilon'])
model.fit(X_train, y_train)
pred = model.predict(X_test)

rmse = metrics.mean_squared_error(y_true = y_test,  y_pred = pred, squared=False
)
print("RMSE:  ", rmse )
```

```
RMSE:   4.182742416094278
```

PSO

In [21]:

```
!pip install  pyswarm
```

```
Requirement already satisfied: pyswarm in /usr/local/lib/python3.7/d
ist-packages (0.6)
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dis
t-packages (from pyswarm) (1.19.5)
```

In [22]:

```
from pyswarm import pso
```

In [23]:

```
# Definição do intervalo de busca dos hiperparâmetros
lw_PSO = [-5, -15, 0.05]
up_PSO = [15, 3, 1.0]
```

In [24]:

```
# Busca pelos melhores hiperparâmetros
def funcao_svr(x):
  c = 2**x[0]
  g = 2**x[1]
  e = x[2]
  svr = SVR(kernel = 'rbf', gamma = g, C = c, epsilon = e)
  modelo_svr = cross_val_score(svr, X, y, cv = 5, scoring = 'neg_root_mean_squar
ed_error')
  rmse_svr = modelo_svr.mean()
  return -rmse_svr

xopt, fopt = pso(funcao_svr, lw_PSO, up_PSO, maxiter = 11, swarmsize = 11)

melhor_c = 2**xopt[0]
melhor_gamma = 2**xopt[1]
melhor_epsilon = xopt[2]
rmse_busca_hiperparametros = fopt
melhores_valores.append([melhor_c, melhor_gamma, melhor_epsilon, rmse_busca_hipe
rparametros])

# Reporte dos melhores valores de hiperparâmetros obtidos e o RMSE associado
print("\n\nPSO - Melhores valores dos hiperparâmetros e RMSE")
print(f"C: {melhor_c}")
print(f"Gamma: {melhor_gamma}")
print(f"Epsilon: {melhor_epsilon}")
print(f"RMSE do 5-fold cross-validation para o melhor conjunto de hiperparâmetro
s: {rmse_busca_hiperparametros}")
```

```
Stopping search: maximum iterations reached --> 11


PSO - Melhores valores dos hiperparâmetros e RMSE
C: 20600.394964096373
Gamma: 3.0517578125e-05
Epsilon: 0.12679707438288512
RMSE do 5-fold cross-validation para o melhor conjunto de hiperparâm
etros: 3.717740787122218
```

Simulated annealing

In [25]:

```
# Vamos implementar o simulated annealing com o pacote scipy.optimize.dual-annea
ling
# Ao utilizarmos no_local_search = true, um algoritmo Generalized Simulated Anne
aling tradicional é utilizado, sem estratégia de busca local

from scipy.optimize import dual_annealing
```

In [26]:

```
# Definição do intervalo de busca dos hiperparâmetros
lw_annealing = [-5, -15, 0.05]
up_annealing = [15, 3, 1.0]
```

In [27]:

```
# Busca pelos melhores hiperparâmetros
def funcao_svr(x):
  c = 2**x[0]
  g = 2**x[1]
  e = x[2]
  svr = SVR(kernel = 'rbf', gamma = g, C = c, epsilon = e)
  modelo_svr = cross_val_score(svr, X, y, cv = 5, scoring = 'neg_root_mean_squar
ed_error')
  rmse_svr = modelo_svr.mean()
  return -rmse_svr

retorno = dual_annealing(funcao_svr, bounds = list(zip(lw_annealing, up_annealin
g)), no_local_search = True, maxiter = 125)

melhor_c = 2**retorno.x[0]
melhor_gamma = 2**retorno.x[1]
melhor_epsilon = retorno.x[2]
rmse_busca_hiperparametros = retorno.fun
melhores_valores.append([melhor_c, melhor_gamma, melhor_epsilon, rmse_busca_hipe
rparametros])

# Reporte dos melhores valores de hiperparâmetros obtidos e o RMSE associado
print("Simulated annealing - Melhores valores dos hiperparâmetros e RMSE")
print(f"C: {melhor_c}")
print(f"Gamma: {melhor_gamma}")
print(f"Epsilon: {melhor_epsilon}")
print(f"RMSE do 5-fold cross-validation para o melhor conjunto de hiperparâmetro
s: {rmse_busca_hiperparametros}")
```

```
Simulated annealing - Melhores valores dos hiperparâmetros e RMSE
C: 19263.21534679803
Gamma: 3.122117946365449e-05
Epsilon: 0.052318871342082754
RMSE do 5-fold cross-validation para o melhor conjunto de hiperparâm
etros: 3.7254341254857435
```

CMA-ES

In [28]:

```
pip install cma
```

```
Requirement already satisfied: cma in /usr/local/lib/python3.7/dist-
packages (3.0.3)
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dis
t-packages (from cma) (1.19.5)
```

In [29]:

```
import cma
```

In [30]:

```
# Definição do intervalo de busca dos hiperparâmetros
# Precisamos trabalhar com intervalos de mesmo tamanho para as três variáveis
lw_cma = [0, 0, 0]
up_cma = [1, 1, 1]

# Ponto inicial aleatório
x0 = [0.5, 0.5, 0.5]

# Desvio padrão igual a 1/4 do intervalo de valores das variáveis
sigma = 0.25
```

In [31]:

```
# Busca pelos melhores hiperparâmetros
def funcao_svr(x):
  c = 2**(-5 + x[0] * 20) # manipulação para compensar o fato de estarmos trabal
hando com intervalo [0, 1]
  g = 2**(-15 + x[1] * 18) # manipulação para compensar o fato de estarmos traba
lhando com intervalo [0, 1]
  e = x[2]
  svr = SVR(kernel = 'rbf', gamma = g, C = c, epsilon = e)
  modelo_svr = cross_val_score(svr, X, y, cv = 5, scoring = 'neg_root_mean_squar
ed_error')
  rmse_svr = modelo_svr.mean()
  return -rmse_svr

opts = cma.CMAOptions()
opts.set('bounds', [lw_cma, up_cma])
opts.set('maxfevals', 125)

es = cma.CMAEvolutionStrategy(x0, sigma, inopts = opts)
es.optimize(funcao_svr)

melhor_c = 2**(-5 + es.result.xbest[0] * 20)
melhor_gamma = 2**(-15 + es.result.xbest[1] * 18)
melhor_epsilon = es.result.xbest[2]
rmse_busca_hiperparametros = es.result.fbest
melhores_valores.append([melhor_c, melhor_gamma, melhor_epsilon, rmse_busca_hipe
rparametros])

# Reporte dos melhores valores de hiperparâmetros obtidos e o RMSE associado
print("\n\nCMA-ES - Melhores valores dos hiperparâmetros e RMSE")
print(f"C: {melhor_c}")
print(f"Gamma: {melhor_gamma}")
print(f"Epsilon: {melhor_epsilon}")
print(f"RMSE do 5-fold cross-validation para o melhor conjunto de hiperparâmetro
s: {rmse_busca_hiperparametros}")
```

```
(3_w,7)-aCMA-ES (mu_w=2.3,w_1=58%) in dimension 3 (seed=800200, Thu
Apr 29 13:47:44 2021)
Iterat #Fevals   function value  axis ratio  sigma  min&max std  t
[m:s]
    1      7 5.555650820742432e+00 1.0e+00 2.36e-01  2e-01  2e-01 0:
02.3
    2     14 4.299066944429675e+00 1.2e+00 2.96e-01  3e-01  4e-01 0:
07.8
    3     21 4.282447459001660e+00 1.8e+00 3.30e-01  3e-01  4e-01 0:
29.9
    4     28 5.271643575288637e+00 2.4e+00 3.00e-01  2e-01  3e-01 0:
34.0
    6     42 3.826998217428316e+00 1.7e+00 2.24e-01  2e-01  2e-01 1:
02.4
    7     49 3.820474059402440e+00 1.4e+00 2.63e-01  2e-01  3e-01 2:
23.6
    8     56 3.734530531714269e+00 1.7e+00 2.51e-01  1e-01  2e-01 2:
44.3
    9     63 3.903690462100827e+00 2.1e+00 2.85e-01  1e-01  3e-01 4:
00.1
   10     70 4.870506557094204e+00 2.6e+00 2.61e-01  1e-01  3e-01 5:
17.3
   11     77 4.163424192207741e+00 3.6e+00 2.24e-01  9e-02  3e-01 5:
30.1
   12     84 3.769008336158570e+00 3.6e+00 2.23e-01  9e-02  3e-01 6:
03.4
   13     91 3.770760037665773e+00 3.9e+00 2.26e-01  8e-02  3e-01 7:
28.7
   14     98 3.787755037743595e+00 4.2e+00 2.02e-01  6e-02  2e-01 8:
40.9
   15    105 3.740945282564641e+00 5.0e+00 1.98e-01  5e-02  2e-01 1
0:11.2
   16    112 3.877725348683133e+00 5.6e+00 1.99e-01  5e-02  2e-01 1
1:32.5
   17    119 3.807446014896867e+00 5.6e+00 1.97e-01  4e-02  2e-01 1
2:39.1
   18    126 3.830285062290729e+00 6.2e+00 1.91e-01  4e-02  2e-01 1
3:33.6


CMA-ES - Melhores valores dos hiperparâmetros e RMSE
C: 17991.340111110334
Gamma: 3.0953014006019875e-05
Epsilon: 0.13332341411220963
RMSE do 5-fold cross-validation para o melhor conjunto de hiperparâm
etros: 3.7345305317142694
```

Comentários

O objetivo deste trabalho foi encontrar os melhores hiperparâmetros para um regressor de SVM usando seis algoritmos de otimização para comparar os resultados a partir do valor RMSE.

Os hiperparâmetros são C, Gamma e Epsilon, que são gerados com distribuições uniformes.

Os dados são divididos com o método cross-validation considerando 5 folds.

Os resultados para cada algoritmo são listados a seguir.

Nota-se, da tabela abaixo, que os maiores valores de RMSE foram obtidos com os algoritmos Random search (4.58) e Grid search (4.32), enquanto os menores valores de RMSE foram obtidos com PSO (3.71) e Simulated Annealing (3.72).

In [32]:

```
import pandas as pd
tabla = pd.DataFrame(melhores_valores, columns=['C', 'Gamma', 'Epsilon', 'RMSE'
])
tabla.index = ['Random search', 'Grid search', 'Otimização bayesiana', 'PSO', 'S
imulated annealing', 'CMA-ES']
tabla
```

Out[32]:

	C	Gamma	Epsilon	RMSE
Random search	673.989953	0.000127	0.868637	4.581288
Grid search	2199.031573	0.000034	0.248965	4.323575
Otimização bayesiana	15061.272904	0.000035	0.285367	3.787595
PSO	20600.394964	0.000031	0.126797	3.717741
Simulated annealing	19263.215347	0.000031	0.052319	3.725434
CMA-ES	17991.340111	0.000031	0.133323	3.734531