In [2]:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.svm import SVR
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import RandomizedSearchCV, GridSearchCV
```

Lendo arquivos

In [3]:

```
x_train = np.load('Xtreino5.npy')
y_train = np.load('ytreino5.npy')

x_test = np.load('Xteste5.npy')
y_test = np.load('yteste5.npy')


#função para reportar os resultados das combinaçẽos de parametros
def report(results, n_top=3):
    for i in range(1, n_top + 1):
        candidates = np.flatnonzero(results['rank_test_score'] == i)
        for candidate in candidates:
            print("Model with rank: {0}".format(i))
            print("Mean validation score: {0:.3f} (std: {1:.3f})".format(
                  results['mean_test_score'][candidate],
                  results['std_test_score'][candidate]))
            print("Parameters: {0}".format(results['params'][candidate]))
            print("")
```

0.1 SVM Regressor

0.2 Medida de erro

In [12]:

```
list_exp_C = range(-5, 15, 5)
list_exp_gamma = range(-15, 3, 3)
list_epsilon = np.arange(0.05, 1, 0.5)


#Lista de parametros para utilizar nos demais
C_list = []
gamma_list = []

best_params = []
best_score = 0
mse = 0
for exp_gamma in list_exp_gamma:
    gamma = 2**exp_gamma
    gamma_list.append(gamma)
    for exp_C in list_exp_C:
        C = 2**exp_C
        C_list.append(C)
        for epilson in list_epsilon:
            clf = SVR(gamma=gamma, C=C, epsilon=epilson)
            clf.fit(x_train, y_train)
            score = clf.score(x_test,y_test)
            if score > best_score:
                best_score = score
                best_params = [gamma, C, epilson]
                mse = mean_squared_error(clf.predict(x_test), y_test)
print("Best Score: ",best_score ," with params: ", best_params," and MSE: ",mse)

C_list = np.unique(C_list)
gamma_list = np.unique(gamma_list)
list_epsilon = np.unique(list_epsilon)
```

```
Best Score:  0.7101346849881802  with params:  [0.000244140625, 1024,
0.05]  and MSE:  19.45487953760004
```

1 Random search

In [64]:

```
param_dist = {"C": [2**-5, 2**0, 2**5, 2**10, 2**15],
              "gamma": [2**-15, 2**-10, 2**-5, 2**0, 2**5],
              "epsilon": [0.05, 0.1, 0.5, 0.8, 1.0]}

clf = SVR()

random_search = RandomizedSearchCV(clf, param_distributions=param_dist, cv=4)

random_search.fit(x_train, y_train)

report(random_search.cv_results_)
```

```
Model with rank: 1
Mean validation score: 0.780 (std: 0.045)
Parameters: {'gamma': 3.0517578125e-05, 'epsilon': 0.5, 'C': 32768}

Model with rank: 2
Mean validation score: 0.725 (std: 0.042)
Parameters: {'gamma': 3.0517578125e-05, 'epsilon': 0.5, 'C': 1024}

Model with rank: 3
Mean validation score: 0.723 (std: 0.040)
Parameters: {'gamma': 3.0517578125e-05, 'epsilon': 1.0, 'C': 1024}
```

2 Grid seach

In [63]:

```
grid_search = GridSearchCV(clf, param_grid=param_dist, cv=5)

grid_search.fit(x_train, y_train)

report(grid_search.cv_results_)
```

```
/home/pazeto/anaconda3/envs/MO431A/lib/python3.7/site-packages/sklear
n/model_selection/_search.py:841: DeprecationWarning: The default of t
he `iid` parameter will change from True to False in version 0.22 and
will be removed in 0.24. This will change numeric results when test-se
t sizes are unequal.
  DeprecationWarning)

Model with rank: 1
Mean validation score: 0.746 (std: 0.040)
Parameters: {'C': 32768, 'epsilon': 1.0, 'gamma': 3.0517578125e-05}

Model with rank: 2
Mean validation score: 0.744 (std: 0.039)
Parameters: {'C': 32768, 'epsilon': 0.5, 'gamma': 3.0517578125e-05}

Model with rank: 3
Mean validation score: 0.744 (std: 0.034)
Parameters: {'C': 32768, 'epsilon': 0.1, 'gamma': 3.0517578125e-05}
```

3 Otimização bayesiana

In [4]:

```
# !pip install hyperopt
from hyperopt import hp, tpe, fmin, space_eval

def my_svr_fun(params):
    C, gamma, epsilon = params['C'], params['gamma'], params['epsilon']
    clf = SVR(gamma=gamma, C=C, epsilon=epsilon)
    clf.fit(x_train, y_train)
    score = clf.score(x_test, y_test)
    return -score

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

best_params = fmin(fn = my_svr_fun,
            space = param_hyperopt, algo=tpe.suggest,
            max_evals = 125)

print("Melhores paramtros: ", space_eval(param_hyperopt, best_params))
print("Score: ", my_svr_fun(space_eval(param_hyperopt, best_params))*-1)
```

```
100%|██████████| 125/125 [00:50<00:00,  2.48it/s, best loss: -0.825205
8541104953]
Melhores paramtros:  {'C': 18254.13344755772, 'epsilon': 0.15646708299
651826, 'gamma': 3.194344320388373e-05}
Score:  0.8252058541104953
```

4 PSO

In [160]:

```
from pyswarm import pso

#primeiro elemento é o mais baixo e mais alte de C, segundo de gamma e terceiro de
lb = [2**-5, 2**-15, 0.05]
ub = [2**15, 2**3, 1.0]

#dado que o pso vai esperar o menor valor e
# queremos o melhor score devemos negativar o score, o mais baixo terá melhores par
def my_svr_fun(params):
#     print(params)
    C, gamma, epilson = params
    clf = SVR(gamma=gamma, C=C, epsilon=epilson)
    clf.fit(x_train, y_train)
    score = clf.score(x_test,y_test)
    return -score

best_params, best_score = pso(my_svr_fun, lb, ub, swarmsize=11, maxiter=11)
print("Parametros: ",best_params)
print("Score:", best_score*-1)
```

```
Stopping search: maximum iterations reached --> 11
Parametros:  [1.86712162e+04 3.05175781e-05 5.29600172e-02]
Score: 0.8296045658080052
```

5 Simulated annealing

In [ ]:

```
from simanneal import Annealer
import random

class SearchBestParamsSVR(Annealer):

    def move(self):

        """permuta a lista de parametros possíveis"""

        random.shuffle(self.state['C'])
        random.shuffle(self.state['gamma'])
        random.shuffle(self.state['epsilon'])

    def energy(self):
        """Calcula o melhor score"""

        C, gamma, epilson = self.state['C'][0], self.state['gamma'][0], self.state[
        clf = SVR(gamma=gamma, C=C, epsilon=epilson)
        clf.fit(x_train, y_train)
        score = clf.score(x_test, y_test)

        return -score


param_dist = {"C": [2**-5, 2**0, 2**5, 2**10, 2**15],
              "gamma": [2**-15, 2**-10, 2**-5, 2**0, 2**5],
              "epsilon": [0.05, 0.1, 0.5, 0.8, 1.0]}

tsp = SearchBestParamsSVR(param_dist)

params, score = tsp.anneal()
```

In [119]:

```
print("Parametros C gamma e epilson:",params['C'][0], params['gamma'][0], params['e
print("Score:", score*-1)
```

```
Pametros C gamma e epilson: 32768 3.0517578125e-05 0.05
Score: 0.8220660478366459
```

6 CMA-ES

In [73]:

```
# !pip install cma
import cma
# help(cma.fmin)
def my_svr_fun(params):
    C, gamma, epilson = params
    clf = SVR(gamma=gamma, C=C, epsilon=epilson)
    clf.fit(x_train, y_train)
    score = clf.score(x_test,y_test)
    return -score


lb = [2**-5, 2**-15, 0.05]
ub = [2**15, 2**3, 1]

opt = { 'bounds': [lb , ub], 'verb_disp':1000}
opt['scaling_of_variables'] = [2**10, 2**1, 0.01]

es = cma.fmin(my_svr_fun, [1, 1, 1], 2.5, opt)

print("Parametros C gamma e epilson: ", es[0])
print("Score: ", es[1]*-1)
```

```
(3_w,7)-aCMA-ES (mu_w=2.3,w_1=58%) in dimension 3 (seed=769965, Wed Ma
y  1 11:49:26 2019)
Iterat #Fevals   function value  axis ratio  sigma  min&max std  t[m:
s]
    1      7 -1.838206063591308e-04 1.0e+00 2.08e+00  2e+00  2e+00 0:0
0.1
    2     14 1.480056132296959e-04 1.4e+00 1.93e+00  1e+00  2e+00 0:0
0.2
    3     21 -9.133457200733330e-03 1.8e+00 1.83e+00  1e+00  2e+00 0:0
0.3
   22    154 -6.326321821002134e-01 1.2e+01 1.54e-01  1e-02  1e-01 0:0
3.5
   34    238 -6.475068819370783e-01 4.6e+01 7.24e-02  1e-03  6e-02 0:0
7.6
   46    322 -7.111023370763685e-01 3.3e+02 1.39e-01  1e-03  2e-01 0:1
3.1
   53    371 -7.681652760797661e-01 9.7e+02 7.36e-01  4e-03  2e+00 0:1
9.7
   56    392 -7.774315533913859e-01 1.1e+03 9.40e-01  3e-03  2e+00 0:2
7.1
   60    420 -7.824298371973549e-01 1.0e+03 8.63e-01  2e-03  1e+00 0:3
6.7
   64    448 -7.975142264359517e-01 1.2e+03 6.91e-01  1e-03  1e+00 0:4
8.6
   67    469 -7.968387811299247e-01 1.3e+03 8.05e-01  9e-04  2e+00 1:0
1.2
   70    490 -8.010360811367336e-01 1.8e+03 7.51e-01  7e-04  1e+00 1:1
3.6
   74    518 -7.956126969487143e-01 1.6e+03 9.39e-01  1e-03  1e+00 1:2
9.4
   78    546 -8.042409170371525e-01 1.6e+03 1.31e+00  2e-03  2e+00 1:4
6.8
   81    567 -7.969227474040904e-01 1.7e+03 1.28e+00  1e-03  1e+00 2:0
4.8
   84    588 -8.013909088537767e-01 2.0e+03 1.39e+00  1e-03  2e+00 2:2
5.1
   88    616 -8.018597706470010e-01 1.5e+03 1.52e+00  1e-03  1e+00 2:4
```

```
  466   3262 -8.256966351184095e-01 2.6e+02 6.48e-02  3e-11  7e-05 53:
07.0
  473   3311 -8.256993397236930e-01 2.3e+02 7.05e-02  3e-11  6e-05 54:
25.5
  482   3374 -8.256917416376404e-01 1.1e+02 1.13e-01  7e-11  5e-05 55:
50.6
  490   3430 -8.256894822402588e-01 9.9e+01 9.33e-02  4e-11  4e-05 57:
10.6
  498   3486 -8.257054469894023e-01 7.9e+01 1.11e-01  5e-11  3e-05 58:
38.8
  508   3556 -8.257005453302735e-01 9.8e+01 1.53e-01  5e-11  4e-05 60:
00.4
  519   3633 -8.256941680231893e-01 2.3e+02 2.29e-01  6e-11  7e-05 61:
22.9
  530   3710 -8.257020333945037e-01 2.1e+02 1.43e-01  3e-11  3e-05 62:
47.5
  541   3787 -8.256941719645042e-01 3.0e+02 2.00e-01  3e-11  7e-05 64:
13.0
  553   3871 -8.257005250102156e-01 3.4e+02 2.49e-01  4e-11  5e-05 65:
44.5
  565   3955 -8.257050357771927e-01 4.4e+02 1.10e-01  1e-11  2e-05 67:
17.8
  577   4039 -8.257007381794392e-01 1.4e+03 4.05e-02  4e-12  8e-06 68:
49.3
  589   4123 -8.256982696352093e-01 1.1e+03 4.68e-02  4e-12  7e-06 70:
18.8
  601   4207 -8.256983777314512e-01 2.1e+03 6.43e-02  7e-12  1e-05 71:
51.3
  613   4291 -8.256978744085418e-01 3.4e+03 7.51e-02  8e-12  1e-05 73:
23.3
  625   4375 -8.256909369531588e-01 4.4e+03 6.49e-02  5e-12  8e-06 74:
55.1
  638   4466 -8.257026310997099e-01 2.9e+03 1.03e-01  4e-12  7e-06 76:
32.5
  650   4550 -8.256973709038034e-01 4.3e+03 6.20e-02  2e-12  4e-06 78:
01.5
termination on tolstagnation=174 (Wed May  1 13:07:30 2019)
final/bestever f-value = -8.257023e-01 -8.257230e-01
incumbent solution: [19397.06710389928, 3.051757888265911e-05, 0.36033
45945951138]
std deviation: [0.0005189705279846922, 4.266018454044987e-12, 3.936943
727446785e-08]
Parametros C gamma e epilson:  [1.94095292e+04 3.05266127e-05 3.605850
58e-01]
Score:  0.8257229759217513
```