In [3]:

```
# SVM hyperparameter optimization

# Data import
from sklearn.model_selection import GridSearchCV, train_test_split
seed=44
from sklearn.datasets import load_wine
import collections
df=load_wine()

# Count of instances per class
collections.Counter(df['target'])
xtrain, xtest, ytrain, ytest = train_test_split(df['data'],df['target'],test_size=0.24,
random_state=seed)
```

In [ ]:

```
from pyGPGO.covfunc import squaredExponential
from pyGPGO.surrogates.GaussianProcess import GaussianProcess
from pyGPGO.surrogates.RandomForest import RandomForest
from pyGPGO.GPGO import GPGO
from pyGPGO.acquisition import Acquisition

from sklearn import datasets
from sklearn import svm
from sklearn.model_selection import cross_val_score

from sklearn.datasets import load_wine
import collections
df=load_wine()

# Count of instances per class
collections.Counter(df['target'])
xtrain, xtest, ytrain, ytest = train_test_split(df['data'],df['target'],test_size=0.24,
random_state=seed)


def accuracy_SVC(C,coef0,gamma):
    clf = svm.SVC(C=C,coef0=coef0,gamma=gamma,kernel='sigmoid')
    scores = cross_val_score(clf, xtrain, ytrain, cv=10)
    return (scores.mean())

from pyGPGO.covfunc import matern32
from pyGPGO.covfunc import gammaExponential

covf=gammaExponential()
covf2=matern32()

GP = GaussianProcess(covf)
RF = RandomForest()

acq = Acquisition(mode='ProbabilityImprovement')
acq3= Acquisition(mode='UCB')

import numpy as np

bo = GPGO(GP,acq,accuracy_SVC,param)
bo.run(init_evals=30,max_iter=120)


bo6=GPGO(RF,acq3,accuracy_SVC,param)
bo6.run(init_evals=30,max_iter=120)


# Import necessary libraries
from pyGPGO.covfunc import squaredExponential
from pyGPGO.surrogates.GaussianProcess import GaussianProcess
from pyGPGO.surrogates.RandomForest import RandomForest
from pyGPGO.GPGO import GPGO
from pyGPGO.acquisition import Acquisition

from sklearn import datasets
from sklearn import svm
from sklearn.model_selection import cross_val_score

import matplotlib.pyplot as plt
```

```
from mpl_toolkits.mplot3d import Axes3D
from pylab import meshgrid,cm,imshow,contour,clabel,colorbar,axis,title,show

#xtrain_further, xval, ytrain_further, yval = train_test_split(xtrain,ytrain,test_size=
0.25,random_state=seed)

def accuracy_SVC(C,coef0,gamma):
    clf = svm.SVC(C=C,coef0=coef0,gamma=gamma,kernel='sigmoid')
    scores = cross_val_score(clf, xtrain, ytrain, cv=10)
    return (scores.mean())

# defining the search spaces for parameters

param = {'C': ('cont', (0.0001,1000)),'gamma': ('cont', (0.0001,100)), 'coef0': ('cont'
,(0,5))}

# create a GP surrogate model with cov function Matérn 3/2 and RF model
from pyGPGO.covfunc import matern32
from pyGPGO.covfunc import gammaExponential
covf=gammaExponential()
covf2=matern32()

GP = GaussianProcess(covf)
RF = RandomForest()

# set the acquisition function
acq = Acquisition(mode='ProbabilityImprovement')
acq2= Acquisition(mode='ExpectedImprovement')
acq3= Acquisition(mode='UCB')

# create an object Bayesian Optimization and iterate

#set seed for comparable results
import numpy as np
#np.random.seed(seed)
ts=time.time()
bo = GPGO(GP,acq,accuracy_SVC,param)
bo.run(init_evals=30,max_iter=12)
#bo.run(init_evals=30,max_iter=120)
time_GP_acq=time.time()-ts
h=bo.history

np.random.seed(seed)
bo2 = GPGO(RF,acq,accuracy_SVC,param)
#bo2.run(init_evals=30,max_iter=120)
bo2.run(init_evals=3,max_iter=12)
time_RF_acq=time.time()-ts
h2=bo2.history

np.random.seed(seed)
bo3 = GPGO(GP,acq2,accuracy_SVC,param)
#bo3.run(init_evals=30,max_iter=120)
bo3.run(init_evals=30,max_iter=12)
time_GP_acq2=time.time()-ts
h3=bo3.history

np.random.seed(seed)
bo4 = GPGO(RF,acq2,accuracy_SVC,param)
bo4.run(init_evals=30,max_iter=120)
#bo4.run(init_evals=30,max_iter=12)
time_RF_acq2=time.time()-ts
```

```
h4=bo4.history

np.random.seed(seed)
bo5 = GPGO(GP,acq3,accuracy_SVC,param)
#bo5.run(init_evals=30,max_iter=120)
bo5.run(init_evals=30,max_iter=12)
time_GP_acq3=time.time()-ts
h5=bo5.history

np.random.seed(seed)
bo6=GPGO(RF,acq3,accuracy_SVC,param)
#bo6.run(init_evals=30,max_iter=120)
bo6.run(init_evals=30,max_iter=12)
time_RF_acq3=time.time()-ts
h6=bo6.history

import pandas as pd
res=[]
for gpgo in [bo,bo2,bo3,bo4,bo5,bo6]:
    i=gpgo.getResult()[0]
    a=[]
    for j in ['gamma','C','coef0']:
        a.append(round(i[j],3))
    res.append(a)
res.append([par_grid['gamma'],par_grid['C'],par_grid['coef0']])
res=np.resize(np.array(res),(7,3))
res=pd.DataFrame(res,columns=['gamma','C','coef0'])
trials=['GP&ProbabilityImprovement', 'RF&ProbabilityImprovement','GP&ExpectedImprovemen
t','RF&ExpectedImprovement','GP&UCB','RF&UCB','GridSearch']
res.insert(loc=0,column='opt method',value=trials)

res

# Performance evaluation

h

import pylab
from pylab import rcParams
rcParams['figure.figsize'] = 15, 7

r=[h,h2,h3,h4,h5,h6]
color=['blue','orange','green','gold','purple','brown']
n=6

for i in range(6):
    pylab.plot(range(13),r[i], color[i], label='history'+str(i+1))
pylab.legend(loc='upper left')
pylab.title('Best seen evolution')
pylab.xlabel('Iteration')
pylab.ylabel('Score')
pylab.show()
```