import numpy as np

import matplotlib.pyplot as plt

from functools import partial

from sklearn.gaussian\_process import GaussianProcessRegressor

from sklearn.gaussian\_process.kernels import Matern

%matplotlib inline

!pip install modal

from modAL.models import BayesianOptimizer

from modAL.acquisition import optimizer\_EI, max\_EI

import numpy as np

# SIMILAR DATOS

X = np.linspace(0, 20, 1000).reshape(-1, 1)

y = np.sin(X)/2 - ((10 - X)\*\*2)/50 + 2

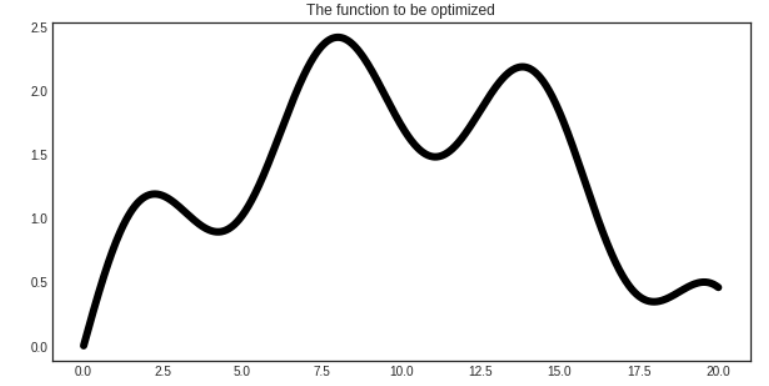
with plt.style.context('seaborn-white'):

    plt.figure(figsize=(10, 5))

    plt.plot(X, y, c='k', linewidth=6)

    plt.title('The function to be optimized')

    plt.show()



# assembling initial training set

X\_initial, y\_initial = X[150].reshape(1, -1), y[150].reshape(1, -1)

# defining the kernel for the Gaussian process

kernel = Matern(length\_scale=1.0)

regressor = GaussianProcessRegressor(kernel=kernel)

# initializing the optimizer

optimizer = BayesianOptimizer(

    estimator=regressor,

    X\_training=X\_initial, y\_training=y\_initial,

    query\_strategy=max\_EI

)

# Bayesian optimization

for n\_query in range(5):

    query\_idx, query\_inst = optimizer.query(X)

    optimizer.teach(X[query\_idx].reshape(1, -1), y[query\_idx].reshape(1, -1))

y\_pred, y\_std = optimizer.predict(X, return\_std=True)

y\_pred, y\_std = y\_pred.ravel(), y\_std.ravel()

with plt.style.context('seaborn-white'):

X\_max, y\_max = optimizer.get\_max()

    plt.figure(figsize=(10, 5))

    plt.scatter(optimizer.X\_training, optimizer.y\_training, c='k', s=50, label='Queried')

    plt.scatter(X\_max, y\_max, s=100, c='r', label='Current optimum')

    plt.plot(X.ravel(), y, c='k', linewidth=2, label='Function')

    plt.plot(X.ravel(), y\_pred, label='GP regressor')

    plt.fill\_between(X.ravel(), y\_pred - y\_std, y\_pred + y\_std, alpha=0.5)

    plt.title('First five queries of Bayesian optimization')

    plt.legend()

    plt.show()

