EDISON

Electromagnetic Design of
flexIble SensOrs

FNP Team-Tech

Artificial intelligence and machine learning for computer aided design of microwave circuits and antennas

Report 1. Theoretical background and first tests of Bayesian Optimization.

Michał Baranowski
September 18, 2020

European Funds Smart Growth

FNP Foundation for Polish Science

European Union European Regional Development Fund

The „EDISOn - Electromagnetic Design of flexIbleSensOrs”project, agreement no TEAM TECH/2016-1/6, is carried out within the TEAM-TECH programme of the Foundation for Polish Science co-financed by the European Union under the European Regional Development Fund.

1 Introduction

The first part of this report outlines the theoretical background of optimization techniques taken into consideration for computer aided design (CAD) of microwave passive components. Both conventional optimization techniques and a new approach with Gaussian processes and Bayesian optimization have been studied. The second part of the report describes the first tests of Bayesian optimization packages which have been carried out so far. Future goals and further work plans are briefly discussed in the conclusion.

2 Literature study on optimization techniques

This section contains a brief description of the optimization methods that may be used for CAD of microwave electronic devices. Conventional techniques which are well known and widely used in the RF sector have been studied first (omitted in the report). A new approach employing algorithms suitable for machine learning (ML): Gaussian Processes (GPs) and Bayesian Optimization (BO), has been examined more closely.

Bayesian Optimization

Bayesian Optimization is an approach to solving challenging optimization problems, such as optimizing black-box and expensive-to-evaluate functions. The general idea is to treat the optimization problem as a decision problem; it uses a machine learning technique (Gaussian process regression) to estimate the objective function based on past evaluations - replacing the original, deterministic function by a probabilistic model (GP) that is used to make decisions about next function evaluations, according to some acquisition function.

Main idea: model f(x) by **GP**(μ(x), k(x,x'))
mean function: μ(x)
covariance function (**kernel**): k(x,x')
1. Choose a prior measure: Gaussian Process
2. Combine the prior and the likelihood to get a posterior measure over the objective, given some observations (function evaluations).
3. Use the posterior to decide where to take the next evaluation according to some acquisition function.
4. Calculate f(x) for the chosen point of the parameter space.

The acquisition function is used to determine where to sample next. It represents the most promising areas of the parameter space to evaluate, in terms of finding the minimum of the optimized function. There are many widely used acquisition functions, such as probability improvement, expected improvement, and Gaussian process upper confidence bound.

3 Bayesian Optimization in practice

3.1 The bayeso package

The first package chosen to carry out tests on is bayeso [1]. It is relatively simple and the code is well documented, therefore it was chosen as a good starting point for learning to use Bayesian Optimization.

The bayeso package offers Python scripts for setting up and running Bayesian Optimization, given a black box objective function written in Python. The package covers both objective function optimization and model selection (Gaussian Process regression). There are several methods for BO and GP optimization implemented in the package.
The package offers three kernel covariance functions: Squared Exponential (Exponentiated Quadratic), Matern 3/2 and Matern 5/2, as well as six acquisition functions: Augmented Expected Improvement (AEI), Expected Improvement (EI), Probability Improvement (PI), pure exploitation, pure exploration,

and Upper Confidence Bound (UCB).
The functions of the package allow working with multi-dimensional input and a scalar output.

3.2 Software setup

The workspace for Bayesian Optimization testing has been set up on Python 3.6 virtual environment under Windows 10 operating system. The bayeso package and suitable libraries enabling access to InventSim solver through COM interface have been installed.

3.3 Testing: synthetic functions

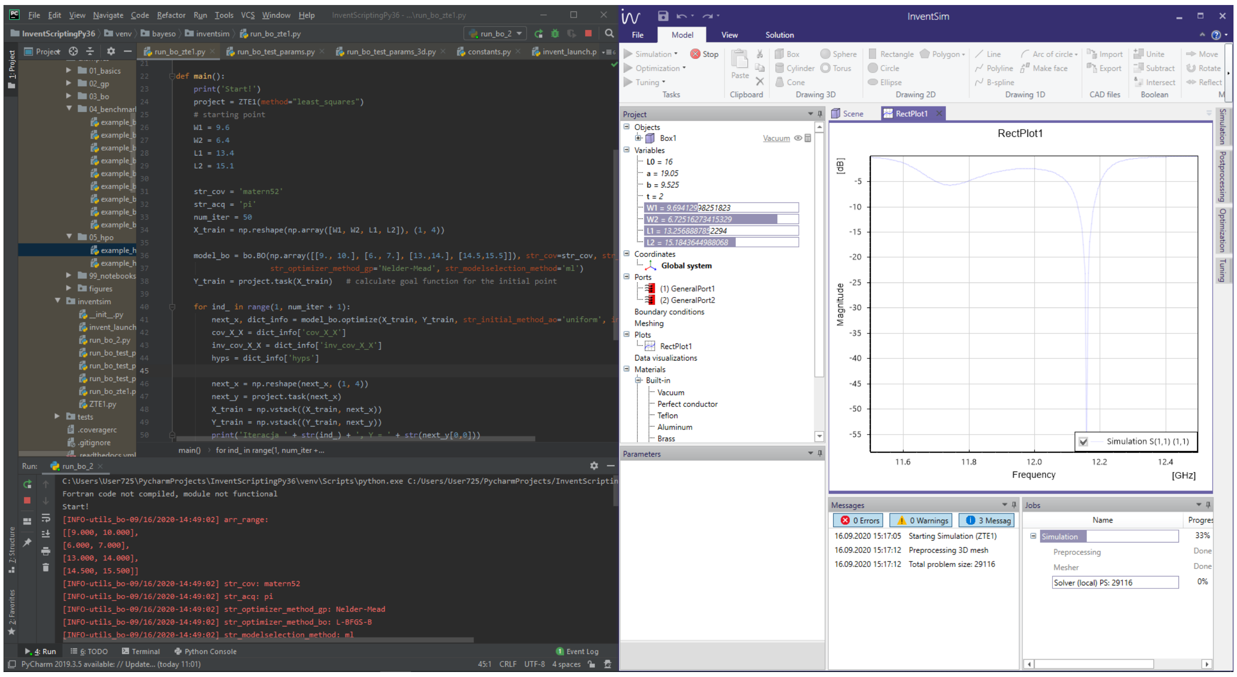
The package has been tested with a set of exemplary scripts included in its contents. These scripts give a good overview of how the package works and show several examples for one- and two-dimensional problems. They allow a clear understanding of what is going on in Bayesian optimization step by step.

However, to test the optimization efficiency for high-dimensional problems, there are several benchmark functions implemented, including Branin (2-dim), Ackley (N-dim), Rosenbrock (N-dim), Hartmann (6-dim) and others. Although the optimization with default settings does well for one-, two- and three-dimensional problems, it fails to minimize some synthetic functions of dimension 4 and higher. The benchmarking results are summarized in the table below.

Function	No. of dimensions	Optimization result
Branin	2	optimized (20 iterations)
Ackley	2	optimized (100 iterations)
Rosenbrock	2	optimized (100 iterations)
Ackley	3	optimized (100 iterations)
Rosenbrock	3	optimized (100 iterations)
Hartmann	6	optimized (100 iterations)
Ackley	4	not optimized (400 iterations)
Rosenbrock	4	not optimized (200 iterations)

3.4 Cavity filter optimization with bayeso and InventSim

To test Bayesian optimization for computer-aided design of microwave components, a simple example of a waveguide cavity filter optimization has been chosen. For this purpose, a dedicated Python class has been created. Its contents include handling InventSim through the COM interface, the main goal function, prepared for two-, three- and four-dimensional input, and functions to read and visualise the results (the scattering parameters).
In order to return a scalar, the output values (calculated for each frequency point) are represented as a sum of squares. The same scalar goal function has been used with Matlab *fmincon* optimization, and has shown to converge quickly: 4 iterations, 65 function evaluations (good starting point), 9 iterations, 100 function evaluations (bad starting point).



```
def main():
    print('Start!')
    project = ZTE1(method="least_squares")
    # starting point
    W1 = 9.6
    W2 = 6.4
    L1 = 13.4
    L2 = 15.1

    str_cov = 'matern52'
    str_acq = 'pi'
    num_iter = 50
    X_train = np.reshape(np.array([W1, W2, L1, L2]), (1, 4))

    model_bo = bo.BO(np.array([[9., 10.], [6., 7.], [13.,14.], [14.5,15.5]]), str_cov=str_cov, str_
                    str_optimizer_method_gp='Nelder-Mead', str_modelselection_method='ml')
    Y_train = project.task(X_train)   # calculate goal function for the initial point

    for ind_ in range(1, num_iter + 1):
        next_x, dict_info = model_bo.optimize(X_train, Y_train, str_initial_method_ao='uniform', i
        cov_X_X = dict_info['cov_X_X']
        inv_cov_X_X = dict_info['inv_cov_X_X']
        hyps = dict_info['hyps']

        next_x = np.reshape(next_x, (1, 4))
        next_y = project.task(next_x)
        X_train = np.vstack((X_train, next_x))
        Y_train = np.vstack((Y_train, next_y))
        print('Iteracja ' + str(ind_) + ', Y = ' + str(next_y[0,0]))
```

```
C:\Users\User725\PycharmProjects\InventScriptingPy36\venv\Scripts\python.exe C:/Users/User725/PycharmProjects/InventScriptin
Fortran code not compiled, module not functional
Start!
[INFO-utils_bo-09/16/2020-14:49:02] arr_range:
[[9.000, 10.000],
[6.000, 7.000],
[13.000, 14.000],
[14.500, 15.500]]
[INFO-utils_bo-09/16/2020-14:49:02] str_cov: matern52
[INFO-utils_bo-09/16/2020-14:49:02] str_acq: pi
[INFO-utils_bo-09/16/2020-14:49:02] str_optimizer_method_gp: Nelder-Mead
[INFO-utils_bo-09/16/2020-14:49:02] str_optimizer_method_bo: L-BFGS-B
[INFO-utils_bo-09/16/2020-14:49:02] str_modelselection_method: ml
```

Variables
L0 = 16
a = 19.05
b = 9.525
t = 2
W1 = 9.69412998251823
W2 = 6.72516273415329
L1 = 13.2568887852294
L2 = 15.1843644988068

Messages
0 Errors | 0 Warnings | 3 Messag
16.09.2020 15:17:05 Starting Simulation (ZTE1)
16.09.2020 15:17:12 Preprocessing 3D mesh
16.09.2020 15:17:12 Total problem size: 29116

Name	Progres
Simulation	33%
Preprocessing	Done
Mesher	Done
Solver (local) PS: 29116	0%

Goal function testing

The conducted tests have shown that BO optimizes the filter design with two and three project variables, in 20 and 50 iterations, respectively. The problems arise with four project variables - then the algorithm requires ca. 200 iterations to find the satisfying solution (for good starting point).
The 4D goal function has been tested for every combination of available acquisition and covariance functions. Although the results are not always satisfying, the tests have shown that the best covariance function is Matern 5/2, while the best acquisition function is Probability Improvement (but it is not the best choice in every situation, however).
The 3D goal function has served as test function for testing all the other parameters, including:

- the model selection method for GP regression,,
- the number of GP samples acquired in every iteration,
- the initialization method for acquisition function optimization,
- the method of optimizing the hyperparameters of the GP,
- the method of optimizing the acquisition function,
- setting own prior mu function (a constant > 0)

3.5 Conclusions

The conducted tests have shown that basic Bayesian optimization encounters problems with high-dimensional function optimization. Nevertheless, the testing of the bayeso package has given a significant intuition of the way the BO works.

4 Further work

The next steps should be focused on further testing of Bayesian optimization algorithms, particularly for high-dimensional problem solving. For this purpose, other software packages will be tested, including Cornell-MOE [2], DPT-BO [3], limbo [4], Dragonfly [5].
These tests should be preceded by further literature studies, including the known usage of BO for high-dimensional CAD of microwave components [6].

References

[1] J. Kim, S. Choi, "bayeso: A Bayesian optimization framework in Python", `http://bayeso.org`, 2017.

[2] J. Wu et al., "Cornell-Metrics-Optimization-Engine", `https://github.com/wujian16/Cornell-MOE`.

[3] H. Torun, M. Swaminathan, "High dimensional Bayesian Optimization with Deep Partitioning Tree", `https://github.com/hakkimerttorun/DPTBO`.

[4] A. Cully, K. Chatzilygeroudis, F. Allocati, J. Mouret, "Limbo: A Flexible High-performance Library for Gaussian Processes modeling and Data-Efficient Optimization", `https://github.com/resibots/limbo`.

[5] K. Kandasamy et al., "Tuning Hyperparameters without Grad Students: Scalable and Robust Bayesian Optimisation with Dragonfly", Journal of Machine Learning Research, 2020. package: `https://github.com/dragonfly/dragonfly/`

[6] H. Torun, M. Swaminathan, "High-Dimensional Global Optimization Method for High-Frequency Electronic Design", IEEE Transactions on Microwave Theory and Techniques, Vol. 67, No. 6, 2019.