Assignment 5
Advanced Machine Learning
University of Milano-Bicocca
M.Sc. Data Science

Federico Signoretta
ID number: 847343
f.signoretta@campus.unimib.it
December 8, 2019

Hyper-Parameter Optimization

The task of this assignment is to perform an Hyper-Parameter Optimization (HPO) of a neural network, with the aim to maximize its Accuracy on 10 fold cross validation. It consists on the following steps:

- Step 1: HPO for just 2 neural network's hyper-parameters
  - the neural network must have 2 hidden layers with 4 units in the first hidden layer and 2 neurons in the second one
  - the hyper-parameters to be optimized are the **learning rate** (numeric in 0.001 - 0.1) and the **momentum** (numeric in 0.1 - 0.9)
  - 5 initial configurations randomly chosen (initial design)
  - choose a surrogate model between **Gaussian Process** and **Random Forest**
  - perform 20 iterations of Sequential Model Based Optimization (SMBO) using an acquisition function and 20 iterations using another one.
  - compare results against 25 configurations in Grid Search and 25 configurations sampled via Pure Random Search
- Step 2: HPO for just 4 neural network's hyper-parameters
  - the neural network must have 2 hidden layers
  - the hyper-parameters to be optimized are the **learning rate** (numeric in 0.01 - 0.1), the **momentum** (numeric in 0.1 - 0.9) and **the number of units** ranges in 1 to 5 for both the two hidden layers
  - 10 initial configurations randomly chosen (initial design)
  - perform 100 iterations of Sequential Model Based Optimization (SMBO) using an acquisition function and 100 iterations using another one.

1 Dataset

In order to perform the above steps, it was used the "*fertility*" dataset, available on the "OpenML" website: *https://www.openml.org/d/1473*. It consists of around 100 instances (missing value are not present) and 9 numeric features (excluding the class). It is about a binary classification diagnosis:

- normal: 1
- altered: 2

It was found the problem of the unbalanced class was: 88 observations about the class 1 and 12 about the class 2 (figure 1). For this experiment, the unbalanced class problem was not treated because the aim of this experiment is to manage correctly the models which lead to an improvement of the hyper-parameters. The aim of hyper-parameter optimization in machine learning is to find the hyper-parameters of a given machine learning algorithm that return the best performance as measured on a validation set.

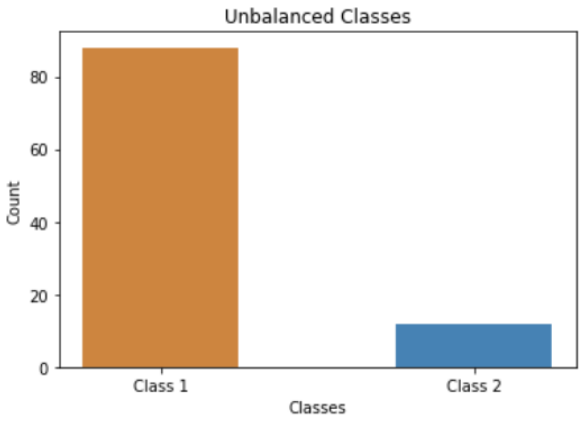

Figure 1: Number of observations for each class

2 Step 1: HPO for just 2 neural network's hyper-parameters

For this step, it was used the **skleran** library due to implement the neural network and the **SMAC3** due to implement the HPO. The Sequential Model-Based Optimization (SMBO) was implemented as follow (SMBO is the formalization of Bayesian Optimization).
The neural network (the **objective function** for the SBMO) was implemented setting just 3 fields of the "Multi-layer Perceptron classifier" function (*skleran*):

1. *hidden_layer_sizes* = (4,2,) obtaining two hidden layer with 4 and 2 units respectively
2. *learning_rate_init* was set as hyper-parameter to be optimized
3. *momentum* was set as hyper-parameter to be optimized

The firsts 5 configurations were randomly chosen and the others were iteratively obtained as result of the HPO process.
The values of the remain fields were left as default values (i.e. activation='relu', solver='adam', etc.). After that, it was build the Configuration Space for the HPO. In particular:

- for the learning rate, *UniformFloatHyperparameter* function (*SMAC*) was used with range of values 0.01-0.1 (its values are sampled from a uniform distribution with values from 0.01 to 0.1)
- for the momentum, *UniformFloatHyperparameter* function (*SMAC*) was used with range of values 0.1-0.9 (its values are sampled from a uniform distribution with values from 0.1 to 0.9)

Through the Configuration Space (that represents **domain** of hyper-parameters over which to search), it was created the *scenario*: in this object, it was set *"runcount-limit"*: 25 to iterate the process 20 times (after the 5 initial configurations).
As **Surrogate model** (Probability Model: is the probability representation of the objective function built using previous evaluations), it was used the **Gaussian Process** (**GP**). Its function is to map from finite training examples to a function that predicts for all possible inputs. The GP model structure is not specified a priori but is determined from the data and for this reason is defined as non-parametric method. The reason why this model was preferred to Random Forest is that GP-based SMBO is suitable for solving problems involving continuous

variables (as in this experiment).
Finally, in order to perform the two experiments, they were chosen the following **acquisition functions**:

1. Lower Confidence Bound (**LCB**): in general, it is the most optimistic estimation on predictions

2. Expected Improvement (**EI**): in general, it deals better with the trade-off between exploitation and exploration and offers significant convergence results.

2.1 Comparison of the the first results

The performances of the two experiments in figure 2 show the cumulative accuracy considering the maximum values reached at each iterations. The experiment with EI as acquisition function

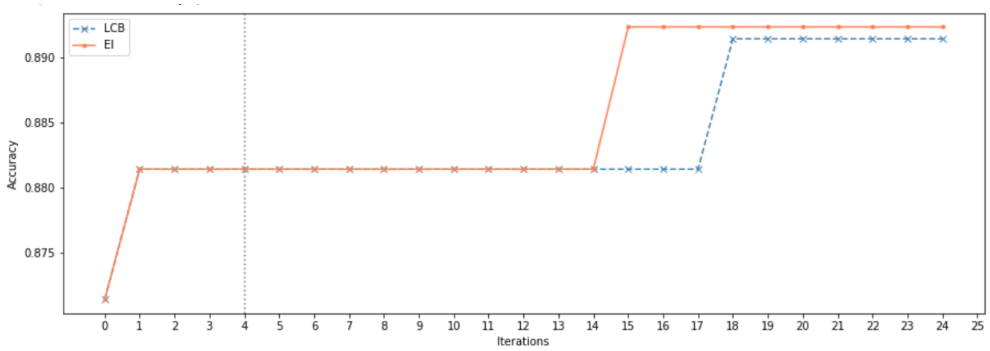

Figure 2: Performances of the two experiments

improves its score from the first iteration (initial configuration) until the 13-th (0.8814) and it improves again (0.8923) until the end. Instead, the experiment with LCB as acquisition function has the same behavior of the EI until the 13-th iteration and at the 16-th it improves (0.8914) until the end. In conclusion, the SBMO with LCB and with EI have about the same performance. It is possible to observe that other experiments with the same parameters could give different results due to the probabilistic nature of the Bayesian Optimization.

2.2 Comparison with Grid and Pure Random Search

Bayesian methods differ from random or grid search in that they use past evaluation results to choose the next values to evaluate. So, the results obtained before were compared against 25 configurations in **Grid Search** and 25 configurations sampled via **Pure Random Search**.
For the Grid Search, the chosen **domain space** is the following:

- 'momentum': [0.1,0.3,0.5,0.7,0.9],
- 'learning_rate_init': [0.01,0.05, 0.075, 0.09, 0.1]

Instead, for the Pure Random Search it was used the *Uniform()* function (from *scipy*) in order to select the value from an uniform distribution:

- 'momentum': uniform(0.1,0.9),
- 'learning_rate_init': uniform(0.01,0.1)

Finally, the results of the four experiment are shown in the following table in figure 5.

	learning rate	momentum	score
GaussianProcess_LCB	0.047432	0.369671	0.891414
GaussianProcess_EI	0.039758	0.624561	0.892323
GridSearch	0.010000	0.900000	0.880000
PureRandomSearch	0.035517	0.960745	0.880000

Figure 3: Results of the two experiments

In this case, the values of accuracy obtained are very similar to each other. But in general, the Bayesian Optimization can return better results than pure or random search.

3 Step 2: HPO for just 4 neural network's hyper-parameters

In order to solve this task, they were used the same libraries mentioned above (SMAC and sklearn).
The neural network (the **objective function** for the SBMO) was implemented setting just 3 fileds of the "Multi-layer Perceptron classifier" function (*skleran*):

1. *hidden_layer_sizes* was set as hyper-parameter to be optimized.
2. *learning_rate_init* was set as hyper-parameter to be optimized
3. *momentum* was set as hyper-parameter to be optimized

The firsts 10 configurations were randomly chosen and the others were iteratively obtained as result of the HPO process.
The values of the remain fields were left as default values (i.e. activation='relu', solver='adam', etc.). After that, it was build the Configuration Space for the HPO. In particular:

- for the hidden_layer_size, *UniformIntegerHyperparameter* function (*SMAC*) was used with range of values 1-5 (its values are sampled from a uniform distribution with values from 1 to 5)
- for the learning rate, *UniformFloatHyperparameter* function (*SMAC*) was used with range of values 0.01-0.1 (its values are sampled from a uniform distribution with values from 0.01 to 0.1)
- for the momentum, *UniformFloatHyperparameter* function (*SMAC*) was used with range of values 0.1-0.9 (its values are sampled from a uniform distribution with values from 0.1 to 0.9)

Through the Configuration Space (that represents **domain** of hyper-parameters over which to search), it was created the *scenario*: in this object, it was set *"runcount-limit"*: 110 to iterate the process 100 times (after the 10 initial configurations).
As **Surrogate model** (Probability Model: is the probability representation of the objective function built using previous evaluations), it was used the **Random Forest** (**RF**). It is an ensemble learning method for both classification and regression and combines Bagging and random selection of features to construct a collection of decision trees with controlled variance. Lastly, RF is better for binary/integer simulation-optimization problems.
In conclusion, in order to perform the two experiments, they were chosen the following **acquisition functions**:

1. Lower Confidence Bound (**LCB**): in general, it is the most optimistic estimation on predictions

2. Expected Improvement (**EI**): in general, it deals better with the trade-off between exploitation and exploration and offers significant convergence results.

3.1 Comparison of the the results

The performances of the two experiments in figure 4 show the cumulative accuracy considering the maximum values reached at each iterations.

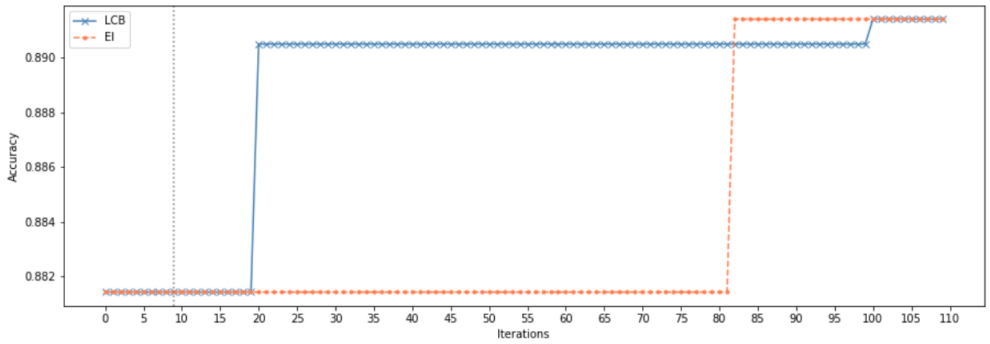

Figure 4: Performances of the two experiments

	learning rate	momentum	neurons_h1	neurons_h2	score
RandomForest_LCB	0.072769	0.493561	5	3	0.891414
RandomForest_EI	0.053593	0.314976	5	3	0.891414

Figure 5: Results

From these results, it is possible to assert that both methods (SMBO with LCB and SMBO with EI) achieve about the same maximum value of accuracy (0.8914) with slightly different learning rate and momentum and same units for each hidden layer. Like on the step 1, it is possible to observe that other experiments with the same parameters could give different results due to the probabilistic nature of the Bayesian Optimization.
Concluding, after several experiments it was possible to assert that thanks to the optimization of these hyper-parameters (learning rate, momentum and number of unit for each hidden layer) this second implementation of the SMBO can reaches values of accuracy greater than in step 1.