Global Optimization in Machine Learning: the design of a Predictive Analytics application

Antonio Candelieri[1] and Francesco Archetti[1]

[1] University of Milano-Bicocca, Department of Computer Science, Systems and Communication, Milan, 20126, Italy
antonio.candelieri@unimib.it

Abstract. Global Optimization, especially Bayesian Optimization, has become the tool of choice in hyperparameter tuning and algorithmic configuration to optimize the generalization capability of Machine Learning algorithms.
The contribution of this paper is to extend this approach to a complex algorithmic pipeline for predictive analytics, based on time series clustering and Artificial Neural Networks. The software environment R has been used with mlrMBO, a comprehensive and flexible toolbox for Sequential Model Based Optimization. Random Forest has been adopted as surrogate model, due to the nature of decision variables (i.e. conditional and discrete hyperparameters) of the case studies considered. Two acquisition functions have been considered: Expected improvement and Lower Confidence Bound, and results are compared.
The computational results, on a benchmark and a real-world dataset, show that even in a complex search space, up to 80 dimensions related to integer, categorical and conditional variables (i.e., hyperparameters), Sequential Model Based Optimization is an effective solution, with Lower Confidence Bound requiring a lower number of function evaluations than Expected Improviement to find the same optimal solution.

Keywords: hyperparameters optimization, global optimization, machine learning.

1 Introduction

The successful application of Machine Learning (ML) in a broad range of domains has recently led to an increase in the demand of these solutions. The goal of any ML algorithm is to make accurate prediction on new data (generalization). However, there is not a unique choice and an algorithm that scores top on one learning task can score poorly on another, as summarized by the “no free lunch theorem” [1]. While this result emphasized the role of the ML experts in the design of reliable applications, the recent interest is focusing on the possibility to disclose ML potential by making it easy to be used off the shelf also by non-experts, taking, in some sense, the human out of the loop [2]. Indeed, a growing number of commercial enterprises is addressing this demand (e.g., Microsoft's Azure Machine Learning, Google's Prediction API, Vizier and Hypertune, Amazon Machine Learning and SigOpt, among others).

Every ML service needs to solve the fundamental issues of deciding which algorithm to use on a given dataset and how to set all the hyperparameters. While a lot of research have been tackling this problem (also named automated machine learning, AutoML) through the application of meta-heuristics (e.g. Genetic Algorithms), more recent works suggest to use Global Optimization (GO), in particular Bayesian Optimization (BO) [2][3]. AutoML was first addressed in AUTO-WEKA [4] that combines a highly parametric ML framework with BO. More recently, a new and robust AutoML system based on SciKit-Learn has been proposed in [5]. This system, AUTO-SKLEARN, uses several classification algorithms, data preprocessing and feature processing methods, leading to an overall number of 110 hyperparameters, which are handled through BO.

ML algorithms are usually compared according to their generalization capability, which consists in evaluating their loss – or any performance index (e.g. accuracy or root mean squared error in classification and regression tasks, respectively) – on a k-fold cross validation procedure. This means that each algorithm must be trained and tested k times, making this procedure time consuming and significantly expensive, especially for those algorithms whose training is expensive/slow. Finally, usually a limited "budget" to train and test different configurations of an ML algorithm is available, comprising computational resources such as CPU, wall-clock time, memory usage.

Sequential Model Based Optimization (SMBO) is a general framework for GO of black-box expensive objective functions, a well-suited tool for optimal hyperparameters tuning of ML algorithms.

Contribution of this paper is the design, implementation and validation of a SMBO framework for addressing the automatic configuration of a complex ML pipeline – instead of a single algorithm – aimed to perform accurate time-series forecasting based on two consecutive phases: time-series clustering and Artificial Neural Network (ANN). All the hyperparameters in the pipeline are decision variables and the objective is the minimization of the forecasting error at the end of the pipeline.

The rest of the paper is organized as follows: in section 2 the basics on SMBO are presented; section 3 presents the predictive pipeline; section 4 provides the details on the experimental setting, and section 5 summarizes results and provides discussion.

2 Global Optimization via Sequential Model Based Optimization

2.1 SMBO process

Let $f(x): X \rightarrow \mathbb{R}$ be an arbitrary black-box expensive function to be optimized, where $X \subseteq \mathbb{R}^d$ is the search space spanned by the decision variables. Each dimension X_i can be either categorical (i.e., $X_i = \{w_{i1}, \dots, w_{is}\}$, with s the number of possible values) or numeric and bounded (i.e., $X_i = [l_i, u_i]$). The final aim of any GO strategy, including SMBO, is to find the optimum x^*, with

$$x^* = \arg\min_{x \in X} f(x)$$

The first of the two basic elements of SMBO is a *surrogate model* $\hat{f}(x)$ approximating $f(x)$ and cheaper to evaluate. The surrogate model is continuously updated any time a new evaluation of the (actual) objective function is performed. The "sequential" nature of SMBO is related to the identification the next most promising to evaluate. The new point is obtained through a mechanism that is the other basic element of SMBO, usually known as *acquisition function*. The generic SMBO process is summarized in the following steps.

```
Sequential Model Based Optimization process
1: Sample an initial set of points and evaluate the objective function on them to obtain the initial set $D_{1:n} = \{(x_i, f(x_i))\}_{i=1:n}$
2: Fit the surrogate model $\hat{f}(x)$ according to $D_{1:n} = \{(x_i, f(x_i))\}_{i=1:n} = \{(x_i, y_i)\}_{i=1:n}$ where $y_i = f(x_i)$
3: Compute the acquisition function according to the current $\hat{f}(x)$ and identify the next promising point $x_{n+1}$
4: Evaluate the objective function in $x_{n+1}$ and update $D_{n+1} = D_n \cup \{(x_{n+1}, y_{n+1})\}$
5: If a given termination criterion is satisfied STOP, otherwise $n = n + 1$ and go to 2
```

2.2 Surrogate Models

The most widely used surrogate model in SMBO is Gaussian Process (GP), also known as Kriging regression [6]. A GP is a non-parametric model completely defined by its prior mean function $\mu_0: X \to \mathbb{R}$ and a covariance function or a positive-definite kernel $k: X \times \mathrm{X} \to \mathbb{R}$. Intuitively, a GP is analogous to a function, but instead of returning a single numeric value for an arbitrary $\bar{x}$, it returns the mean and variance of a normal distribution over the possible values of $f(\bar{x})$. Therefore, given the set of points and associated evaluations $D_{1:n} = \{(x_i, f(x_i))\}_{i=1:n} = \{(x_i, y_i)\}_{i=1:n}$ and a given point $\bar{x}$, the estimation of $f(\bar{x})$, conditioned on $D_{1:n}$, is normally distributed, where the posterior mean and variance evaluated at any point $\bar{x}$ represent the model's prediction and uncertainty, respectively, in $f(\bar{x})$.

The main drawback of GP is the computational complexity: at every iteration the surrogate is inferred with a complexity $O(n^3)$ where n is the number of evaluations performed so far. This means that also time for the identification of the next point to evaluate, as well as the overall wall-clock time, increases over the SMBO iterations.

Furthermore, this approach scales poorly with the dimensionality of the search space: to reduce the impact of dimensionality some proposals have been recently suggested [7][8]. To mitigate these problems, alternative surrogate models have been proposed: one of the most widely adopted is Random Forest (RF) for regression. RF is an ensemble learning strategy, whose base classifiers/regressors are Decision Trees (DTs). It practically proved to be a scalable and highly parallelizable solution for regression problems and inherits the capability of DTs to deal also with categorical and conditional variables. An RF generates a multitude of DTs, at training time, and provides as output the mean (or median) prediction among the individual trees, when adopted for regression. RF differs from other typical ensemble methods (e.g. voting systems) in the way they introduce random perturbations into the training phase. The basic idea is to combine bagging, to sample examples from the original dataset, and random selection of features, used to train every tree of the forest. The algorithm for training a RF regression model is summarized as follows.

Random Forest learning algorithm

Given:

- *S* the size of the forest (i.e. number of decision trees in the forest)
- *N* the number of examples in the dataset (i.e. evaluations of the objective functions)
- *M* the number of features (i.e. decision variables) representing every example
- *m* the number of features (i.e. decision variables) to be randomly selected from the *M* available for each leaf node of the tree to be split
- n_{min} the minimum node size in the tree

```
1: for $i = 1$ to $S$
2:     sample, with replacement, a subset of $N$ instances from the dataset
3:     for every terminal node (leaf) of the $i$-th decision tree having size lower than $n_{min}$
4:         select $m$ features (i.e. decision variables) at random from the $M$ available
5:         pick the best feature (i.e. decision variable) and split among the possible $m$
6:         split the node in two children nodes (new leaves of the decision tree)
7:     end for
8: end for
```

Injecting randomness simultaneously with both bagging and features selection allows for training a collection of DTs with controlled variance. Variance is crucial in replacing GPs with RF process: the possibility to compute the variance of the RF allows for keeping the overall SMBO process as is, except for the regression method.

2.3 Acquisition Functions

Any acquisition function aims at managing the trade-off between *exploration* and *exploitation*. At each iteration of the SMBO process the new promising point x has to be identified by weighting the advantage of considering the region of the search space currently providing more chance to improve the current solution – which means "exploiting" the current knowledge represented by the surrogate model $\hat{f}(x)$ – rather than moving towards less explored regions of X – which means "exploring" new solutions, in particular according to the uncertainty of the surrogate model $\hat{f}(x)$.

Expected Improvement (EI) [9] and Confidence Bound (CB) are the most adopted acquisition functions [2] and are also considered in this paper. EI measures the expectation of the improvement on $f(x)$ with respect to the predictive distribution of the surrogate model $\hat{f}(x)$, while CB is a strategy that is optimistic with respect to the uncertainty. Lower Confidence Bound (LCB) and Upper Confidence Bound (UCB) are used for minimization and maximization problems, respectively. EI is computed as:

$$EI(x) = \left(f^{+} - \mu(x)\right)\Phi(Z) + \sigma(x)\phi(Z)$$

where f^+ is the best value observed so far ("best seen"), $\mu(x)$ and $\sigma(x)$ are respectively the mean and standard deviation of the surrogate model, $\Phi(Z)$ and $\phi(Z)$ are the distribution and the density of the standard normal distribution, and Z is:

$$Z = \begin{cases} \dfrac{f^+ - \mu(x)}{\sigma(x)} & if\ \sigma(x) > 0 \\ 0 & otherwhise \end{cases}$$

LCB, used in this study, is computed as:

$$\mu(x) - \beta\sigma(x)$$

where β is a parameter, usually known as "inflate", to balance the trade-off between exploration and exploitation (we use $\beta = 1$, in this study).

To solve the original black-box optimization problem the optimization of the acquisition function is required, but this process is usually cheaper and easier than the original one and it is addressed through methods, such as the derivative free DIRECT or multi-start BFGS (Broyden–Fletcher–Goldfarb–Shanno), in the case of continuous decision variables, or "focus search" (proposed in the mlrMBO software) to handle numeric, categorical, mixed and hierarchical/conditional decision variable spaces (thus, well suited for a RF based surrogate model).

3 The Predictive Analytics Application Proposed

3.1 Case Studies and Available Data

To validate the proposed approach, we used a benchmark and a real-world dataset. The former is known as "Appliances energy prediction Data Set"[1] and the latter consists of the urban water demand data in Milan, from 2016-10-01 to 2017-09-30. Unlike the water case study, the energy dataset is multivariate, but we adopted our predictive pipeline to find an accurate univariate forecasting model for the energy consumed by appliances. Urban water demand data were collected through the Supervisory Control And Data Acquisition (SCADA) system monitoring the urban water distribution network in Milan, serving approximately 1.5 million habitants. In both case studies, data were organized into a distinct time-series dataset $D = \{x_1, \ldots, x_l\}$ consisting of l vectors, one for each day in the observation period, and where each vector x_i is a set of 24 ordered values, one for each hour of the day.

3.2 A Two Stage Machine Learning based Pipeline

This section provides the details about the predictive analytics pipeline to optimize. Although we have already presented the idea of two consecutive analysis stages, which are time-series clustering and regression, respectively [10-12], the optimization of the overall pipeline is addressed in this paper for the first time.

[1] https://archive.ics.uci.edu/ml/datasets/Appliances+energy+prediction

Time-series data are organized into a dataset where each row is a vector of V components representing the evolution of data in a time window of interest (e.g., hourly water or energy consumptions during the day).

The basic advantage of using time-series clustering as first stage is that it might allow for the identification of typical patterns within the time window of interest and, consequently, for splitting the dataset into subsets (i.e. clusters) which are then used in the second stage. In this paper, time-series clustering is performed via kernel k-means [13].

At the second stage, an ANN is trained for each cluster: with respect to [10-12], this leads to a reduction of the number of forecasting models to be trained and, potentially, the number of hyperparameters to be optimized. **Fig. 1** summarizes the overall pipeline.

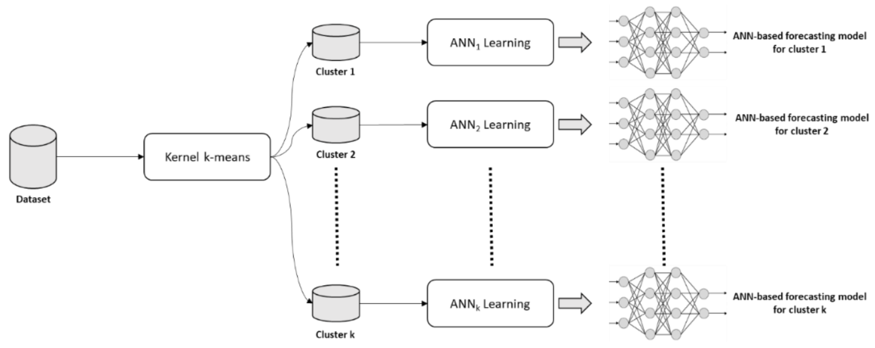
Dataset
Kernel k-means
Cluster 1
Cluster 2
Cluster k
ANN₁ Learning
ANN₂ Learning
ANNₖ Learning
ANN-based forecasting model for cluster 1
ANN-based forecasting model for cluster 2
ANN-based forecasting model for cluster k

Fig. 1. The proposed predictive analytics pipeline: first stage is clustering through kernel k-means; second stage consists to learn an ANN for each cluster

First Stage: Clustering. The kernel k-means algorithm [13] is a generalization of the standard k-means: it implicitly maps data from the Input Space, spanned by the original set of features (i.e. decision variables of the SMBO process), to a higher-dimensional space, namely Feature Space. Therefore, kernel k-means can discover clusters that are non-linearly separable in Input Space. In [14] optimization of kernel k-means through SMBO has been proposed with the aim to replace graph partitioning in a leakage localization application to increase computational efficiency.

Given a set $\mathcal{X}$ of vectors x_1, x_2, …, x_n, the kernel k-means objective can be written as the minimization of:

$$g(x_i) = \sum_{k=1}^{\bar{k}} \sum_{x \in C^k} \|\Phi(x_i - \bar{x}_k)\|^2, \quad \forall\, x_i \in \mathcal{X}$$

Where $\bar{k}$ is the number of clusters, $\Phi(x)$ is a (non-linear) function mapping vectors x_i from the Input Space $\mathcal{X}$ to the Feature Space, and $\bar{x}_k$ is the centroid of cluster C^k.

Expanding $\|\Phi(x_i - \bar{x}_k)\|^2$, $g(x_i)$ can be written as:

$$g(x_i) = \Phi(x_i)\Phi(x_i) - \frac{2\sum_{x_j \in C^k} \Phi(x_i)\Phi(x_j)}{|C^k|} + \frac{\sum_{x_j, x_i \in C^k} \Phi(x_j)\Phi(x_i)}{|C^k|^2}.$$

Only inner products are used in the computation of the distance between every vector $x_i \in \mathcal{X}$ and the centroid of the cluster C^k. Thus, given a kernel matrix $K_{ij} = \Phi(x_i)\Phi(x_j)$, these distances can be computed without knowing explicit representations of $\Phi(x)$.

Second Stage: Artificial Neural Network based forecasting. The second stage of the pipeline consists in training $\bar{k}$ different ANNs, one for each one of the $\bar{k}$ clusters resulting from the first stage. The first v values of the time series and the remaining $V - v$ are, respectively, the input and output neurons of every ANN. In this study we set $v = 6$ for both the two datasets, meaning that the first six actual values are sufficient to forecast, in one shoot, the overall consumption pattern for the day.

It is important to highlight that the "internal" organization of the ANN is completely free, only the number of input and output neurons have been set up, while all the other ANN's hyperparameters are optimized through the SMBO process. The list of ANN's hyperparameters is provided in the following section 3.2. Finally, the number of ANNs is not given a priori, it depends on the number k of clusters identified from the first stage. In any case the training process for the k ANNs is performed in parallel.

In the following **Fig. 2** the second stage of the predictive analytics pipeline is summarized, describing how data in a cluster are used to train the corresponding ANN.

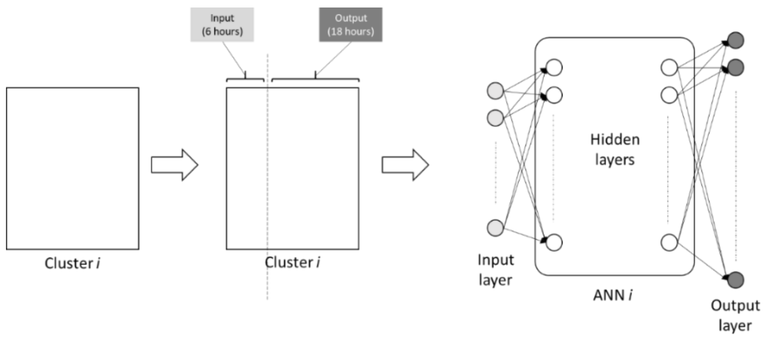

Fig. 2. Second stage of the predictive analytics pipeline: training an ANN on a cluster

3.3 Hyperparameters of the Predictive Analytics Application

The following Table 1 and Table 2 summarize, separately for the two stages, the hyperparameters of the predictive analytics pipelines (i.e., decision variables).

Table 1. - Decision variables / hyperparameters for the clustering phase

Variable	Hyperparameter	Type	Description
x_1	$\bar{k}$	integer	Number of clusters. Possible values are form 3 to 9
x_2	kernel type	categorical	Type of kernel used in the kernel based clustering. Possible kernels are: linear, spline, rbf, laplace, bessel, polynomial

x_3	σ	numeric, conditioned	Hyperparameter of the rbf, laplace and Bessel kernels. Possible values are in [10^{-5}, 10^{5}]
x_4	degree	integer, conditioned	Hyperparameter of the bessel and polynomial kernels. Possible values are 2, 3 or 4

Table 2. - Decision variables / hyperparameters for the ANN learning phase

Variable	Hyperparameter	Type	Description
$\bar{x}_1^k$	hidden layers	integer	Number of hidden layers in the artificial neural network. Possible values are 1,2 or 3
$\bar{x}_2^k$	neurons in the hidden layer 1	integer	Number of neurons in the hidden layer 1. Possible values are from 1 to 20
$\bar{x}_3^k$	neurons in the hidden layer 2	integer, conditioned	Number of neurons in the hidden layer 2 (if $\bar{x}_1^k > 1$). Possible values are from 1 to 20
$\bar{x}_4^k$	neurons in the hidden layer 3	integer, conditioned	Number of neurons in the hidden layer 3 (if $\bar{x}_1^k > 2$). Possible values are from 1 to 20
$\bar{x}_5^k$	algorithm	categorical	Type of algorithm used to train the artificial neural network. Possible values are: **back-prop**, **rprop+**, **rprop-**, **sag**, **slr**
$\bar{x}_6^k$	learning rate	numeric, conditioned	Learning rate of the backprop algorithm. Possible values are in the range [0.1, 1.0]
$\bar{x}_7^k$	error function	categorical	Function used to compute training error. Possible values are: **sse** and **ce**
$\bar{x}_8^k$	activation func-tion	categorical	Function used to compute the output of every neuron. Possible values are: **logistic** and **tanh**
$\bar{x}_9^k$	linear output	logical	This hyperparameter defines whether to use a linear combination in the output layer of the artificial neural network or not. Possible values are TRUE or FALSE.

According to the value of the decision variable x_1 (i.e. the number of clusters) the number of decision variables in the second phase is consequently defined: it is x_1 times 9. Since x_1 is an integer ranging from 3 to 9, this means that the overall number of decision variables for the pipeline ranges from $4 + 3 \times 9 = 31$ to $4 + 9 \times 9 = 85$.

3.4 Prediction Error as Objective Function

The performance measure used to evaluate the overall predictive analytics pipeline is based on the Mean Average Percentage Error (MAPE), a widely adopted measure to evaluate time-series forecasting models in different application domains, such as environmental observations or water/energy consumptions. The MAPE between an actual time series and its forecast is computed as follows:

$$MAPE = \frac{1}{T}\sum_{t=1}^{T}\left|\frac{A_t - F_t}{A_t}\right|$$

where A_t and F_t are the actual and forecasted value, respectively, at time t and T is the number of time steps predicted. However, there are 2 specific issues to address in using MAPE for the evaluation of our predictive analytics pipeline:

- we have a MAPE value for every time series in the dataset (computed through cross validation)
- an ANN is trained for every cluster identified in the first stage, so MAPE should be considered for every cluster.

Thus, since we have formulated a single-objective optimization problem, some aggregation on MAPE values is needed to have just one value for the objective function, given a setting of the pipeline's hyperparameters. We have decided to consider the minimization of the worst MAPE computed on every time series data and over the entire set of ANN forecasters. Finally, the optimization problem can be stated as:

$$\min_{x \in X} \left\{ f(x) = \max_{k=1,\dots,\bar{k}} \overline{MAPE}_k \right\}$$

where $\overline{MAPE}_k$ is the worst (i.e. the highest) value of MAPE computed through 10-fold cross validation on the cluster $k = 1, \dots, \bar{k}$. Folds were obtained through a random stratified split, without considering any information about time periods.

4 Experimental Setting

4.1 Set Up of the SMBO Process

We have decided to adopt the following setting for the SMBO process:

- ***Initialization***: 100 initial evaluations of the objective function, through random Latin Hypercube Sampling (LHS)
- ***Surrogate model***: Random Forest – the choice was driven by the nature of the hyperparameters (i.e. discrete and conditioned decision variables)
- ***Acquisition Function***: LCB and EI
- ***Termination Criteria***: Maximum number of evaluations set to 500 (excluding the 100 evaluations for initialization)

The default configuration for RF was used: 500 trees and 1/3 of features selected from those available. The objective function is the one described in the previous section 3.4. An experiment using Random Search (i.e. 600 iterations of LHS) has been also performed to better understand the benefits provided by SMBO.

4.2 Computing System Configuration

Experiments have been performed on the following system: Intel Core i7-4712HQ @ 2.30GHz, RAM 16GB, Microsoft Windows10 (64bit). The framework has been implemented by using R 3.4.0 (x64) with the packages: mlrMBO [15], neuralnet, kernlab.

5 Results and Discussion

This section summarizes the results obtained by applying the proposed approach.

A small portion of each dataset has been used as independent test set, while the larger part has been used to perform 10 folds-cross validation. The indicators considered to evaluate the optimization process are:

- the "best seen" (i.e. max objective function value obtained over function evaluations)
- the first iteration which "best seen" appears at
- and wall-clock time.

Information on computational time is split in: *training time* (required to fit the surrogate model), *proposition time* (required to identify the next point x_{n+1} to evaluate) and *evaluation time* (required to perform the new function evaluation, $f(x_{n+1})$).

Table 3 and Table 4 summarize results on the two datasets, separately. SMBO found a better configuration of the predictive pipeline than Random Search (i.e. 600 function evaluations on LHS), and with a very small MAPE. Although MAPE starts with higher values on the energy data, the final values are quite similar for the two datasets.

With respect to computational time: for LCB, *training time* is 0.7-0.8% of the overall wall-clock time, and *proposition time* is around 3.0-3.3%. This means the effort is almost completely devoted to evaluate objective function: this is quite logical, as it requires to run clustering and, even more expensive, 10 folds-cross validation on as many ANNs as the clusters identified. Similar percentages have been obtained for EI.

Table 3. – Results on water dataset. Some measures are not applicable for LHS

Acquisition Function	MAPE Best Seen [%]	Best Seen Iteration*	Training Time [secs]	Proposition Time [secs]	Evaluation Time [secs]
EI	0.10	386	1337.69	5855.86	184181.80
LCB	0.10	182	1335.24	5711.23	188850.80
LHS	14.80	-	-	-	59764.04

** preliminary 100 evaluations for initialization are included*

Table 4. – Results on energy dataset. Some measures are not applicable for LHS

Acquisition Function	MAPE Best Seen [%]	Best Seen Iteration*	Training Time [secs]	Proposition Time [secs]	Evaluation Time [secs]
EI	0.08	196	1963.19	6505.95	193732.90
LCB	0.08	147	1584.38	6394.54	195716.20
LHS	6.67	-	-	-	230760.60

** preliminary 100 evaluations for initialization are included*

In the following Fig. 3, the evolution of the "best seen" over the SMBO iterations is reported, for both the LCB and EI, and for the two datasets, separately. In all the cases

SMBO has identified the same optimal solution. Moreover, in the case of LCB, SMBO required a lower number of function evaluations than EI.

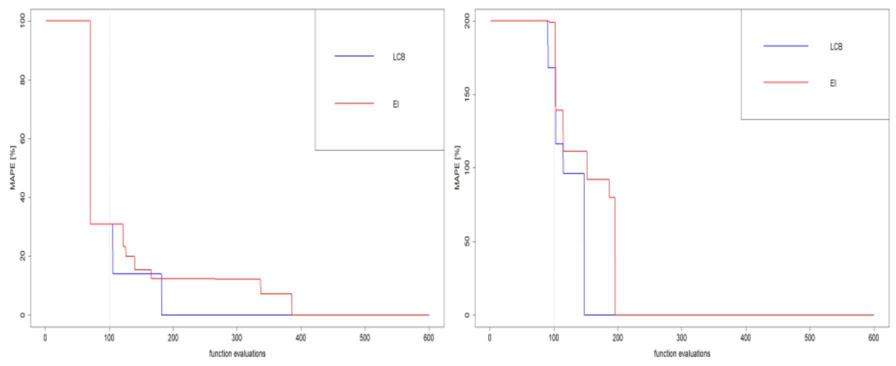

Fig. 3. Results on water data (left) and energy data (right): best seen MAPE over function evaluations, for LCB and EI. First 100 evaluations are for initialization.

When the best forecasting pipelines, one for each dataset, were applied to the corresponding portions of data left as independent test sets, the computed MAPE resulted to be aligned with that obtained in 10 folds-cross validation, more precisely: 0.19% and 0.12% for the water and energy data, respectively.

6 Conclusions

Given the increasing systems complexity, black-box optimization will be gaining ever more importance as a design paradigm for such diverse targets as physical systems, user interfaces and ML applications. SMBO is an effective backbone for building one such optimization infrastructure, both statistically sound and flexible enough to cater to different design targets and delivery environments.

Although a major finding of this paper is that SMBO can still handle up to 85 variables, the hundreds of them, as required to tune Deep Learning algorithms, are still a challenge: promising research areas are to use a portfolio of acquisition functions on a parallel architecture and to perform SMBO in a low dimension manifold identified by random, Laplacian or Deep embedding.

Finally, ongoing work is aimed at applying SMBO to optimize the forecasting pipeline on other application domains, even those which, unlike water and energy, could be not characterized by typical behaviours, for instance to forecast price, volume or return of financial stocks.

Compliance with Ethical Standards:

Conflict of Interest: Antonio Candelieri declares that he has no conflict of interest. Francesco Archetti declares that he has no conflict of interest.

Ethical approval: This article does not contain any studies with human participants or animals performed by any of the authors.

References

1. Wolpert, D. H.: The lack of a priori distinctions between learning algorithms. Neural computation, 8(7), 1341-1390 (1996).
2. Shahriari, B., Swersky, K., Wang, Z., Adams, R. P., de Freitas, N.: Taking the human out of the loop: A review of bayesian optimization. Proceedings of the IEEE, 104(1), 148-175, (2016).
3. Snoek, J., Larochelle, H., Adams, R.P.: Practical Bayesian Optimization of Machine Learning Algorithms, arXiv:1206.2944 [stat.ML] (2012).
4. Thornton, C., Hutter, F., Hoos, H.H., Leyton-Brown, K. Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms. In: Proceedings of ACM SIGKDD. 847–855 (2013).
5. Feurer, M., Klein, A., Eggensperger, K., Springenberg, J., Blum, M., Hutter, F.: Efficient and robust automated machine learning. In Advances in Neural Information Processing Systems, 2962-2970 (2015).
6. Huang, D., Allen, T.T., Notz, W.I., Zeng, N.: Global optimization of stochastic black-box systems via sequential kriging meta-models, J. Glob. Optim., 34(3), 441–466, (2006).
7. Wang, Z., Zoghi, M., Hutter, F., Matheson, D., De Freitas, N.: Bayesian Optimization in High Dimensions via Random Embeddings, Proc. Int. Jt. Conf. Artif. Intell., 1778–1784, (2013).
8. Kandasamy, K., Schneider, J., Pòczos, B.: High Dimensional Bayesian Optimisation and Bandits via Additive Models, in International Conference on Machine Learning, 37, 295–304, (2015).
9. Mockus, J., Tiesis, V., Zilinskas, A.: The application of Bayesian methods for seeking the extremum, in Towards Global Optimisation 2, L. Dixon and G. Szego, Eds. Elsevier, 117-130, (1978).
10. Candelieri, A., Archetti, F.: Identifying typical urban water demand patterns for a reliable short-term forecasting - The icewater project approach, Procedia Eng., 89, 1004–1012, (2014).
11. Candelieri, A., Soldi, D., Archetti, F.: Short-term forecasting of hourly water consumption by using automatic metering readers data, Procedia Eng., 119(1), 844–853, (2015).
12. Candelieri, A.: Clustering and support vector regression for water demand forecasting and anomaly detection, Water, 9(3), 224, (2017).
13. Dhillon, I.S., Guan, Y., Kulis, B.: Kernel k-means: spectral clustering and normalized cuts. In: Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 551–556 (2004)
14. Candelieri, A., Giordani, I., Archetti, F.: Automatic Configuration of Kernel-based Clustering: an optimization approach, in International Conference on Learning and Intelligence Optimization, pp. 34-49 (2017). Springer, Cham.
15. Bischl, B., Richter, J., Bossek, J., Horn, D., Thomas, J., Lang, M.: mlrMBO: A Modular Framework for Model-Based Optimization of Expensive Black-Box Functions. arXiv preprint arXiv:1703.03373 (2017).