Théo DEMESSANCE 学号: 6219000082

Hugo PREVOTEAU 学号:6219000083

Christophe HAIKAL 学号:6219000084

Valentin TASSEL 学号: 6219000085

Topic Survey : Bayesian Optimization

Automating Bayesian optimization with Bayesian optimization, NeurIPS, 2018.

Authors : Gustavo Malkomes and Roman Garnet**.**

In many global optimization problems motivated by engineering applications, the number of function evaluations is severely limited by time or cost. Bayesian optimization is a powerful framework for globally optimizing expensive-to-evaluate functions :

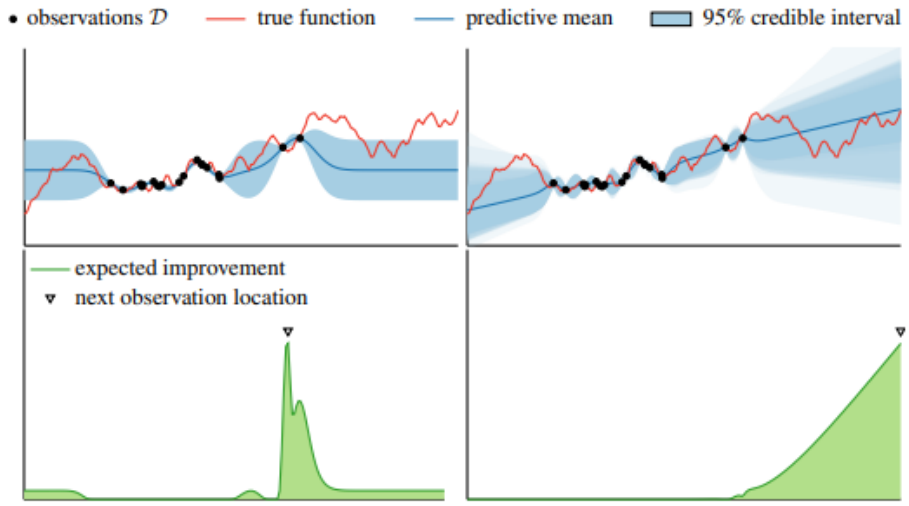
$$\mathbf{x}_{\mathrm{OPT}} = \arg\max_{\mathbf{x}\in X} f(x)$$

.

This framework is very powerful because it can be effective use even if the objective function are : nonconvex, observed without gradients, or " black boxes": we effectively have a black box access to the function. A typical application is to do hyperparameter optimization: can be having a primary tuning of machine learning models for example tweaking the learning rate of a deep neural network, or generalize performance such as accuracy.

In Bayesian optimization there are two main steps first they construct a probabilistic belief about the objective function typically using adoption process model, for example a Gaussian Process model.

In the second they design an acquisition function to guide their search for the optimum value: where to evaluate the function next. there are many different and interesting acquisition functions such as probability of improvement (from Kushner.1964), expected improvement (from Mockus, 1978) and more.



An example here we want to optimize the function in red left is a typical Bayesian optimization model using square exponential where on the right we have our automated based optimization approach which we call ABO ABO automatically finds appropriate models for the objective function we can see that by doing a better job at modeling the function we will find the optimal value in the next iteration of ABO but the simpler model the standard based organization will need more function evaluations because the next observation is far away from the optimal.

Algorithm 1 Automated Bayesian Optimization

Input: function f, budget B, initial data $\mathcal{D}$
$\{\mathcal{M}_i\} \leftarrow$ Initial set of promising models
repeat
  $\{\mathcal{M}_i\} \leftarrow$ update models $(\{\mathcal{M}_i\}, \mathcal{D})$
  $\{\mathcal{M}_i\} \leftarrow \text{ABOMS}(\{\mathcal{M}_i\}, \mathcal{D})$
  $p(\mathcal{M} \mid \mathcal{D}) \leftarrow$ compute model posterior
  discard irrelevant models $p(\mathcal{M}_i \mid \mathcal{D}) < 10^{-4}$
  $\mathbf{x}^* \leftarrow \arg\max_{\mathbf{x} \in X} \alpha_{\text{EI}}(\mathbf{x}; \mathcal{D})$.
  $y^* \leftarrow f(\mathbf{x}^*) + \varepsilon$
  $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mathbf{x}^*, y^*)\}$
until budget B is depleted

The key component of their ABO is the model search procedure that they execute in the main Bayesian optimization loop what is interesting is that they propose to use another Bayesian optimization procedure as the engine for their model search : Inner Bayesian optimization maximizing the model evidence

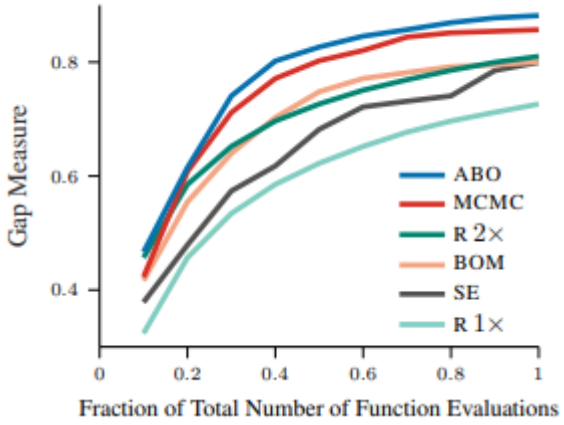
$$\mathcal{M}_{\text{OPT}} = \arg\max_{\mathcal{M} \in \mathbb{M}} g(\mathcal{M}; \mathcal{D}), \quad \text{with } g(\mathcal{M}; \mathcal{D}) = \log p(\mathbf{y} \mid \mathbf{X}, \mathcal{M})$$

the (log) model evidence can be seen as the function to be maximized.

The inner optimization tries to maximize the evidence similar to our previous paper on 2016 here however we have to change our observation model to deal with the fact that the model evidence is changing as we obtain more functional observations.

In their experiments we can see that ABO is extremely simple efficient outperforming many competitive baselines:

real-world objectives	SVM	3	0.903	0.912	0.840	0.938	**0.956**
	LDA	3	0.939	**0.950**	0.925	**0.950**	**0.950**
	Logistic regression	4	0.928	0.774	0.899	0.936	**0.994**
	Robot pushing $3d$	3	0.815	0.927	0.878	**0.967**	0.935
	Robot pushing $4d$	4	**0.824**	**0.748**	0.619	**0.668**	**0.715**
	Neural network Boston	4	0.491	0.594	0.703	0.640	**0.757**
	Neural network cancer	4	**0.845**	**0.645**	**0.682**	**0.773**	**0.749**
	Cosmological constants	9	0.739	0.848	0.859	0.984	**0.999**
	mean gap		0.810	0.800	0.801	0.857	0.882
	median gap		0.834	0.811	0.850	0.937	0.943



They introduced a novel automated Bayesian optimization approach that dynamically selects promising models for explaining the observed data using Bayesian organization in the model pace by doing a better job at modeling he objective function we can be extremely simple efficient.

Towards Automated Bayesian Optimization, ICML 2018

Authors: Gustavo Malkomes and Roman Garnett

One of the key component in Bayesian optimization is the probabilistic model used for the objective function f. The main problem of this choice is that we may not choose properly a model, especially when gathering data is expensive.

In this work, we focus on developing a novel automated Bayesian optimization when we dynamically chose the model for the data, and note without human intervention.

Global optimization of expensive gradient-free functions has long been a critical component of many complex problems in science and engineering. Gradient-free functions can be the hyperparameters in a neural-network. Bayesian optimization has nonetheless shown remarkable success on optimizing expensive gradient-free functions.

Inspired by some recent developments on automated model selection, we show how to automatically and dynamically select appropriate models for Bayesian optimization. Our framework could be extended to any probabilistic model, but here we focus on Gaussian processes models, which are a standard modeling tool for Bayesian optimization.

Imagine we want to optimize the function f. The main goal to optimize f, is to find the global optimum, noted as :

$$\mathbf{x}_{\text{OPT}} = \arg\min_{\mathbf{x} \in X} f(\mathbf{x})$$

We can obtain this x(opt) through a sequence of evaluations of the objective function f. This problem becomes particularly challenging when we have a limited number of function evaluations.

Let's denote D as the set of gathered observations D = (X, y) where X is a matrix representing the variables x(i), and y the respective vector of the function value y(i).

Assume we are given a prior distribution over the objective function p(f) and, after observing new information, we have means of updating our belief about f using Bayes' rule:

$$p(f \mid \mathcal{D}) = \frac{p(\mathcal{D} \mid f)p(f)}{p(\mathcal{D})}.$$

The posterior distribution above is then used for decision making, i.e., selecting the x we should query next. Suppose we are given a collection of probabilistic models P = {Mi} that offer plausible explanations for the data. We take a fully Bayesian approach and we use the model evidence (or marginal likelihood), the probability of generating the observed data given a model M,

$$p(\mathbf{y} \mid \mathbf{X}, \mathcal{M}) = \int_{\Theta_{\mathcal{M}}} p(\mathbf{y} \mid \mathbf{X}, \theta, \mathcal{M}) p(\theta \mid \mathcal{M}) \, \mathrm{d}\theta,$$

as the key quantity for measuring the fit of each model to the data. Then, we can compute the posterior model, which gives us a principled way of combining the beliefs of all models. Infine, we can summarize the distribution as :

$$\begin{aligned} p(f \mid \mathcal{D}) &= \sum_i p(\mathcal{M}_i \mid \mathcal{D}) \, p(f \mid \mathcal{D}, \mathcal{M}_i) \\ &= \sum_i p(\mathcal{M}_i \mid \mathcal{D}) \int_{\Theta_{\mathcal{M}}} p(f \mid \mathcal{D}, \theta, \mathcal{M}_i) \, p(\theta \mid \mathcal{D}, \mathcal{M}_i) \, \mathrm{d}\theta, \end{aligned}$$

Given our belief about f, we want to use this information to select which point x we want to evaluate next. This is typically done by maximizing a so-called acquisition function α: X ⟶ R

$$\mathbf{x}^* = \arg\max_{\mathbf{x} \in X} \alpha(\mathbf{x}; \mathcal{D}).$$

The α function can be defined as the expected improvement. It can be defined as :

$$\begin{aligned} \alpha_{\mathrm{EI}}(\mathbf{x}; \mathcal{D}, \mathcal{M}) &= \mathbb{E}_f\big[\max(f' - f, 0)\big] \\ &= \int_f \max(f' - f, 0) \, p(f \mid \mathbf{x}, \mathcal{D}, \mathcal{M}) \, \mathrm{d}f. \end{aligned}$$

This formula is valid for when we only have one model chosen. But, in our case we want to incorporate multiple models: so we have:

$$\begin{aligned} \alpha_{\mathrm{EI}}(\mathbf{x}; \mathcal{D}) &= \int_f \max(f' - f, 0) \, p(f \mid \mathbf{x}, \mathcal{D}) \, \mathrm{d}f. \\ &= \sum_i p(\mathcal{M}_i \mid \mathcal{D}) \int_f \max(f' - f, 0) \, p(f \mid \mathbf{x}, \mathcal{D}, \mathcal{M}_i) \, \mathrm{d}f \\ &= \sum_i p(\mathcal{M}_i \mid \mathcal{D}) \, \alpha_{\mathrm{EI}}(\mathbf{x}; \mathcal{D}, \mathcal{M}_i) = \mathbb{E}_{\mathcal{M}}\big[\alpha_{\mathrm{EI}}(\mathbf{x}; \mathcal{D}, \mathcal{M})\big]. \end{aligned}$$

In this section, we introduce our automated method for Bayesian optimization. To compute the prior probability to choose our model, we use the Gaussian process, extremely powerful modeling tools. But, these models depend a lot of the mean function and the covariance. To select the best model, we use the following formula :

$$\mathcal{M}_{\text{OPT}} = \arg\max_{\mathcal{M} \in \mathbb{M}} g(\mathcal{M}; \mathcal{D}),$$

where g(M; D) = log p(y|X,M).

As we want to select the best model automatically, we need to develop an algorithm, described as followed :

First, we initialize our set of promising models P. Then, at each iteration: we find the next location x*. If the probability is inferior to a threshold, we can then exclude the current model. Then, we use the Mopt to include more promising candidate.

To validate our approach we use test functions commonly used as benchmarks for optimization. The goal is to find the global minimum of each test function given a limited number of function evaluations. To evaluate our model, we compare our approach with several Bayesian optimization alternatives.

test function	domain	RQ	SE	ARD SE	BOM	MCMC	ABO
Branin	$[-5, 10] \times [0, 15]$	0.870	0.856	0.929	0.985	0.901	**0.993**
Six-Hump Camel	$[-3, 3] \times [-2, 2]$	0.771	0.701	0.554	0.851	0.787	**0.873**
Drop-Wave	$[-5.12, 5.12]^2$	0.244	0.372	0.413	0.430	0.362	**0.492**
Mccormick	$[-1.5, 4] \times [-3, 4]$	1.00	1.00	1.00	1.00	1.00	1.00
Ackley	$[-5, 5]^5$	**1.00**	0.794	0.724	0.372	**1.00**	**1.00**
Alpine2	$[0, 10]^2$	0.820	0.862	0.881	0.864	**0.888**	0.851
Mean		0.784	0.764	0.750	0.750	0.823	**0.868**

We can see that ABO, out model : Automatic Bayesian Optimization outperforms the others alternatives, even if MCMC, a model similar to ABO has also good scores.

We introduced a novel automated Bayesian optimization approach that uses multiple models to represent its belief about the objective function f and subsequently decide where to query f next. Our method automatically and efficiently searches for better models as more data is gathered. Initial empirical results show that the proposed algorithm usually outperforms all baselines

Bayesian Optimization Under Uncertainty (NIPS 2017)

Authors: Justin J.Beland, Prasanth B.Nair

Abstract

This paper considers the problem of robust optimization, to sustain a specified measure of performance even in an uncertain problem.

This problem is such a challenge because in fact modelling such a system is not viable computationally, so this paper proposes a framework to solve a class of this problems.

The main idea developed in this paper is to use the Gaussian process for loss together in order to find the robust optimal solution.

Introduction

Take an optimization problem such as we try to minimize a measure of loss, so the optimum is less sensitive to the noise factor.

The primary focus of this paper is to develop Bayesian Optimisation (BO) methods to solve problems such as the one written above that can be expressed as the following form :

$$\boldsymbol{x}^\star = \underset{\boldsymbol{x}\in\{\boldsymbol{x}_1,\boldsymbol{x}_2\}}{\arg\min}\ \mathcal{J}(\boldsymbol{x}) \qquad \text{s.t.} \qquad \Pr[c_j \leq 0] \ \geq\ 1-\eta,\ j=1,\ldots,d_{\boldsymbol{c}},$$

With $J(x)$ denotes a loss function, and $\eta \in [0,1]$ is a user defined parameter that controls the probability of constraint satisfaction.

Bayesian methods are mainly used to locate the minima of complex optimization problems with limited computational budget.

Bayesian Optimization Under Uncertainty

By using the Bayes Risk, we can get a formula for our loss function:

$$\mathcal{J}(\boldsymbol{x}) = \int_{\mathcal{Q}} f(\widetilde{\boldsymbol{x}})p(\boldsymbol{\zeta})d\boldsymbol{\zeta}.$$

By using this we can accommodate new alternative robustness metrics such as the aggregate of the mean and the variance, the minimax principle and the horsetail matching.

To guide the research in order to find the minimizer x * while making sure that we keep a high probability of satisfaction, we need to identify the x^{t+1} point where we should next query the deterministic query model.

To locate this query point, we maximize α_x^c , defined as:

$$\alpha_{\boldsymbol{x}}^{\mathbf{c}}(\boldsymbol{x}) = \alpha_{\boldsymbol{x}}(\boldsymbol{x}) \prod_{j=1}^{d_{\mathbf{c}}} \Pr[c_j(\boldsymbol{x}_1, \boldsymbol{x}_2 + \boldsymbol{\delta}, \boldsymbol{\xi}) \leq 0]$$

We then have the following algorithm :

Algorithm 1: Bayesian Optimization Under Uncertainty

$\mathcal{D}^{1:t} = \{\widetilde{\boldsymbol{x}}^{1:t}, \boldsymbol{y}^{1:t}, \boldsymbol{c}_y^{1:t}\}$ // Initialize training dataset with t samples

while $cost \leq budget$ **do**

$f \sim \mathcal{GP}(\mu_f^{\text{pos}}, k_f^{\text{pos}}) \xleftarrow[\text{on } \mathcal{D}^{1:t}]{\text{conditioning}} \mathcal{GP}(\mu_f^{\text{pr}}, k_f^{\text{pr}})$

$\mathcal{J} \sim \mathcal{GP}(\mu_{\mathcal{J}}^{\text{pos}}, k_{\mathcal{J}}^{\text{pos}})$

$c_j \sim \mathcal{GP}(\mu_{c_j}^{\text{pos}}, k_{c_j}^{\text{pos}}) \xleftarrow[\text{on } \mathcal{D}^{1:t}]{\text{conditioning}} \mathcal{GP}(\mu_{c_j}^{\text{pr}}, k_{c_j}^{\text{pr}}),\ j = 1, \ldots, d_{\mathbf{c}}$

$\boldsymbol{x}^{t+1} = \arg\max_{\boldsymbol{x}} \alpha_{\boldsymbol{x}}^{\mathbf{c}}(\boldsymbol{x}) \qquad \boldsymbol{\xi}^{t+1} = \arg\max_{\boldsymbol{\xi}} \alpha_{\boldsymbol{\xi}}^{\mathbf{c}}(\boldsymbol{\xi})$

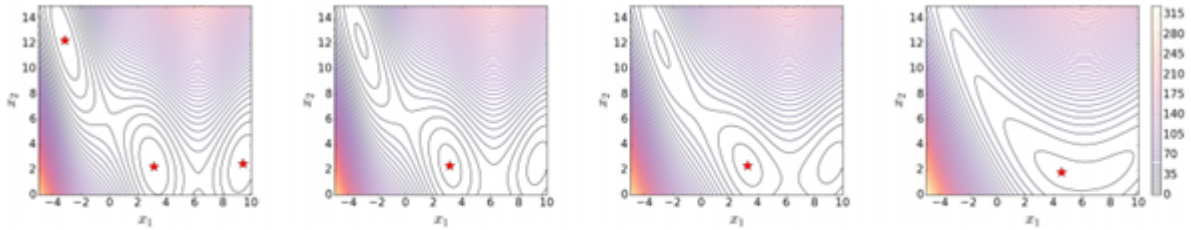
$\mathcal{D}^{1:t+1} \leftarrow \mathcal{D}^{1:t} \cup \{\widetilde{\boldsymbol{x}}^{t+1}, y^{t+1}, \boldsymbol{c}_y^{t+1}\}$

$t \leftarrow t + 1$

Numerical Studies

Using the minimization of the Branin function, under uncertainty. For this study and as said previously, the Bayes Risk is used as the lost function.

The following figure shows the contour plot of Bayes risk while varying $\delta_b \in \Re^2$

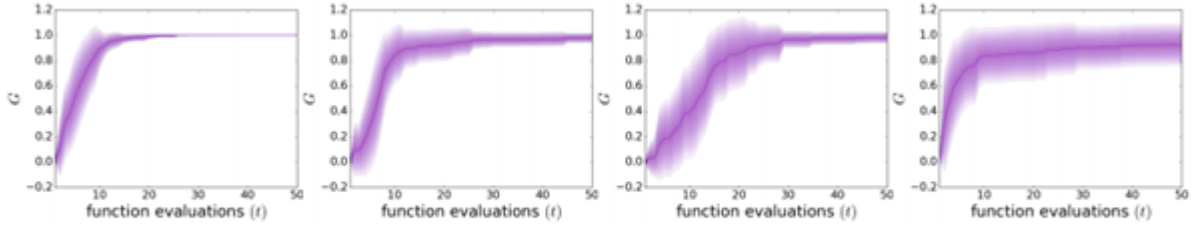


(a) $\boldsymbol{\delta}_b = [0, 0]^{\mathsf{T}}$ (b) $\boldsymbol{\delta}_b = [1, 1]^{\mathsf{T}}$ (c) $\boldsymbol{\delta}_b = [2, 2]^{\mathsf{T}}$ (d) $\boldsymbol{\delta}_b = [3, 3]^{\mathsf{T}}$

Figure 1: Bayes risk for the Branin function where red stars indicate global minima.

The red stars represent the global minima.

Then the study of different acquisition functions, such as POI, EI and UCB then the random one on the sample with $\delta_b \in [1, 1]^T$.



(a) $\alpha_{\boldsymbol{x}}^{\mathbf{c}}$: POI (b) $\alpha_{\boldsymbol{x}}^{\mathbf{c}}$: EI (c) $\alpha_{\boldsymbol{x}}^{\mathbf{c}}$: LCB (d) $\alpha_{\boldsymbol{x}}^{\mathbf{c}}$: random

Figure 2: Convergence of the gap metric G for various acquisition functions.

These graphs represent the convergence of the different methods, and we can see that the POI acquisition function performed very well.

But compared to the randomized function we can see that all the methods have a lower uncertainty in the later iterations.

Conclusion

This paper proposed an efficient framework to solve optimization problem sensitive to uncertainty. Knowing the fact that solving these problems is expansive and solving the optimization very demanding computationally speaking.

It has been shown that by defining a GP model on the objective function, estimating the covariance and mean can be solved analytically. Similarly, applying a GP model on the constraints we can approximate more efficiently the probability of constraint satisfaction.

Finally, it has been showed that by maximizing the specified acquisition function we could update the points used to query the expensive objective function. Some ongoing work would be to accelerate the convergence of the last method, by identifying multiple query points in parallel.

Practical Automated Machine Learning for the AutoML Challenge 2018, **ICML, 2018 AutoML Workshop**

Authors : Matthias Feurer, Katharina Eggensperger, Stefan Falkner, Marius Lindauer, Frank Hutter

This article describes the winning entry to the Auto ML challenge, which combines an automatically preselected portfolio, ensemble building and Bayesian optimization with successive halving.

First let's see the competition format to fully understand the contribution of the article. The challenge focused on supervised learning with featurized data. The competition had five rounds, each of them featuring five new, featurized datasets from unknown domains. Each of its rounds consisted of an auto-phase. The goal is to submit an AutoML-system to be executed without human intervention on the Codalab platform.

The approach is the PoSH Auto-sklearn :

They use the successive halving and also adapted our configuration space to consider classifiers which can be leveraged by SH's budget allocation. SH is use for its simplicity. They combine Bayesian Optimization with Successive halving with the BOHB method since BOHB outperform the others methods.

They choose to start with the same static portfolio candidate configurations for every dataset. They constructed this portfolio as follows: they gathered 421 binary-classification datasets from OpenML . For each dataset they optimized our machine learning pipeline with SMAC for up to 1000 function evaluations or 2 days, whichever was reached first. Next, they evaluated these pipelines on all 421 datasets to obtain a 421×421 matrix containing the test scores. They then used greedy

submodular function minimization, the normalized regret across all datasets in this matrix to obtain a portfolio of machine learning pipelines.

The configuration space was a subspace of the Auto-sklearn configuration space suitable to work with SH's budgets:

To work with their pipeline they preprocess the dataset to this extent :

- feature scaling
- imputation of missing value
- treatment of categorical values

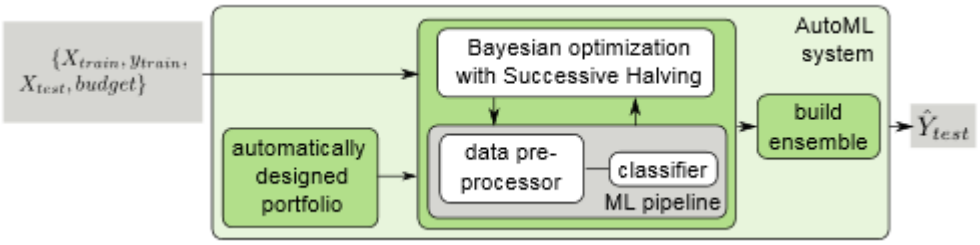
The configuration space take one of four classifier :

- SVM Support vector machine
- RF Random Forest
- Linear classification
- XGBoost

Leading to a total of 37 hyperparameters

They combined all of the above methods to form their challenge entry. The name provide from **Po**rtfolio **S**uccessive **H**alving combined with Auto-sklearn.

Here is a diagram, which summarize the pipeline :



To build ensembles and compare configurations they have trained the system using ⅔ of the available data and ⅓ for validation in SH and building the ensemble. They add manual design decision to have robust result in the challenge.

Here is the dataset provided for the final evaluation :

Name	#Sampl.	#Feat.	Seq.	time (sec)
Riccardo	20 000	4 296		1200
Rm	28 278	89	✓	1200
Pm	29 964	89	✓	1200
Rh	31 498	76	✓	1200
Ri	30 562	113	✓	1200

And the results in this dataset :

Name	#Configs (m)	Ensemble size	Val. Perf (p)	Val. Perf	Test Perf	Rank
Riccardo	39 (0)	2(2/0/0)/ 2	0.9997	0.9997	0.2635	1
Rm	47 (3)	3(2/1/0)/ 4	0.7629	0.7757	0.6766	5
Pm	28 (0)	2(2/0/0)/ 2	0.8669	0.8669	0.5533	3
Rh	32 (0)	6(4/2/0)/23	0.3658	0.3702	0.2839	4
Ri	47 (7)	9(3/5/1)/28	0.3478	0.3806	0.3932	1

Overall, the results indicate that the portfolio provided a robust and diverse enough set of pipelines, so it was hard to find much better configurations in the remaining time.

Conclusion and future study

In the future, they plan to study the impact of our manual design decisions with an ablation study and develop techniques to automate these as well.

Synthesis

All articles presented deal with Bayesian optimization, let's introduce the concept :

Many optimization problems in machine learning are black box optimization problems. A black-box can be a system, a function, a device which can be viewed in terms of its inputs and outputs, without any knowledge of its internal workings. In our case, our black box is an objective function f(x). We do not have the derivatives and the analytical expression of *f* and we evaluate the function by sampling at a point x and getting possibly noisy response. We can sample with grid search, random search or numeric gradient estimation if the evaluation of *f* is cheap. Otherwise it's important to minimize the number of samples drawn from the black box function *f*.

This is the domain where Bayesian optimization techniques are most useful. They attempt to find the global optimum in a minimum number of steps.

For example, the most popular application of bayesian optimization is the automatic hyperparameter tuning of machine learning algorithm. With this tuning, we obtain the best machine learning algorithms' configuration by optimizing the estimation of the error of these algorithms. Despite being applied with success, bayesian optimization methodologies also have hyperparameters which need to be configured such as the probabilistic surrogate model or the acquisition function used. An inaccurate decision over the configuration of these hyperparameters implies obtaining poor results. Typically, these hyperparameters are tuned by making assumptions of the objective function we want to evaluate but there are scenarios where we do not have any prior information about the objective function.

So first let us summarise our papers really quickly :

The first step in Bayesian optimization is the system automation. In the article written by Gustavo Malkomes and Roman Garnett, they developed a automated Bayesian optimization which dynamically chose the model for the data, without any human intervention.

The main idea of this paper is to automate everything. The model will take in input the promising models. Then, at each iteration, they will compute the probability of where we can evaluate the function next using this model. If the probability is inferior to a given threshold, we can drop the current model, because it'd mean that the model does not fit the given data.

As it has been published in 2 differents conferences (ICML and NeurIPS), we have two different approaches for the same topic. Indeed, in the ICML version, they detailed more the mathematical aspect of the optimization, by explaining us the probability concept behind. While the NeurIPS version is focused on the algorithm and computation aspect. It describes us more precisely how to develop it and what are the expected results.

It has been tested and compared to others existing optimization models, and it outperforms the others by far. The metric used is the global minimum of each model, tested on several functions on different domains and different dimensions. In all of these functions, our model has better results.

The third article of this survey **“Bayesian optimization under uncertainty”** treats the problem of robust optimization with not certain parameters. These problems are not computationally viable as it requires too much resources to compute them.

The idea of this paper is to use the Gaussian Process to find a optimization to such a problem. This method adds a probabilistic dimension to the problem. The bayesian methods are mainly use in order to find an optimum value for problems optimization using a limited computational budget.

Using the Bayesian Risk they are able to find the loss function, they try to find the best solution for the problem, while keeping a high satisfaction in terms of probabilities.

They will try to use that probabilistic approach on the most popular Bayesian optimization techniques, and compare the convergence speed between them, to validate the interest of using such a model in an uncertain problem.

In the fourth article **”Practical Automated Machine Learning for the AutoML Challenge 2018“** we see an application of the Bayesian optimization. *A mettre en dernier dans la synthèse*

The idea proposed in this paper is to use automatically constructed machine learning pipelines, as suggested by SMAC, a Bayesian optimization method, using meta-learning and post-hoc ensemble building. They used the Bayesian Optimization HyperBand (BOHD) because it was the state of the art method outperforming every other Bayesian optimization methods. Using the Bayesian optimization and the ML pipeline, they managed to propose a winning AutoML system that outperformed the rest of the competition. This paper shows the real life applications of the Bayesian optimization and how adaptive this method is.

As a conclusion, it may be said that Bayesian optimization can be used in many way, the first one is used in the paper **“Bayesian optimization under uncertainty”** proposed in this survey. It explains how to solve an optimization problem using Bayesian optimization using uncertain variables. Using the Gaussian Process, to get the loss function, this paper shows how to improve the convergence speed of a method while keeping a high satisfaction in terms of probabilities. They compared the different Bayesian optimization technique on a dataset to compare the convergence speed to find the solutions, and showed a real improvement.

Then, another approach can be without any human intervention. That’s what we can learn by reading the second article **“Towards Automated Bayesian Optimisation”.**

Another approach is the one used in **”Practical Automated Machine Learning for the AutoML Challenge 2018“** where Bayesian optimization is added to a AutoML system to improve the optimization of the hyperparameters of the problem. In classification problems, one of the big issues is to find the correct parameters and for some functions, depending on the parameters, can take a lot of time. This paper then showed an application of Bayesian optimization.