Practical Bayesian Optimization of Machine Learning Algorithms

By Jasper Snoek and Hugo Larochelle

NIPS 2012

Group 12
Caroline Rippey, Alexandre Dutour, Simon Bayle

POSTECH

Table of contents

Optimization of machine learning algorithms

One could want to optimize an algorithm in order to be :

- Time efficient
- Cost efficient

Optimizing a machine learning algorithm consists of optimizing a **black box** problem. One has multiple ways to optimize it :

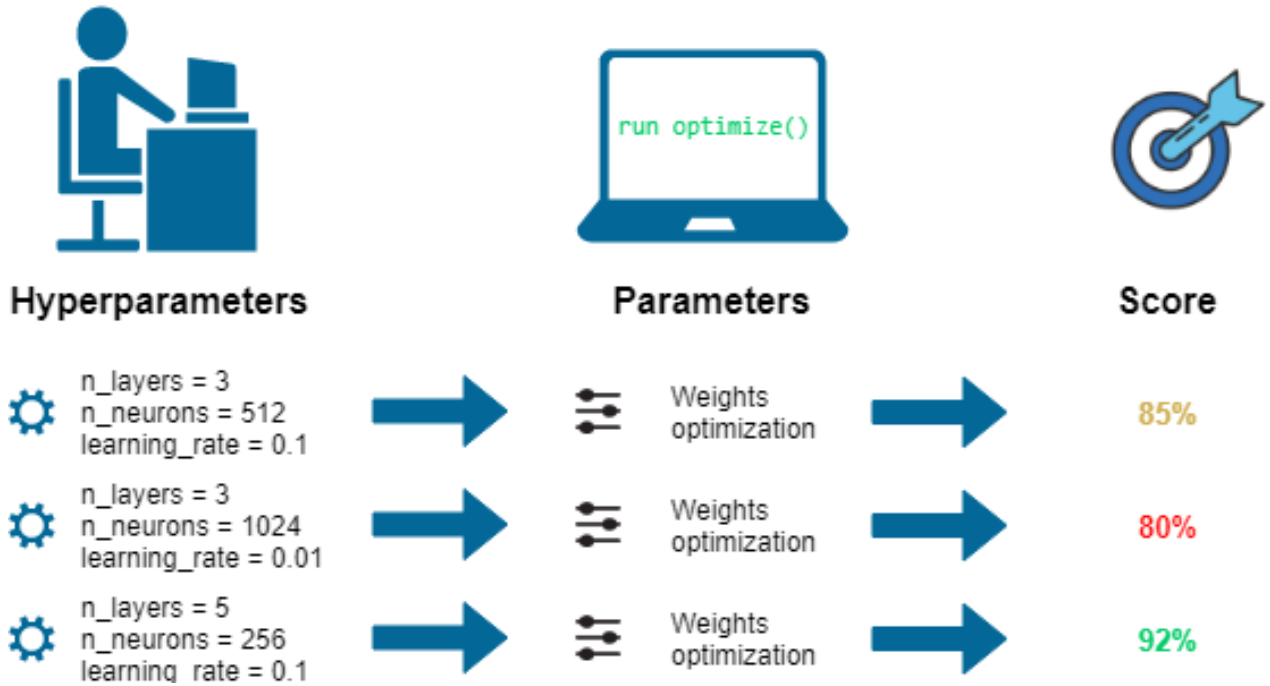
- By simply **increasing the computational power**, we will then improve the execution time at the expense of the costs
- Optimizing the algorithm **hyperparameters**

Hyperparameters of a ML algorithm

A **hyperparameter** is a parameter that is set before the beginning of the learning processes and impacts the effectiveness of a model training.

This can be :

- The learning rate
- Properties of the neural network (number of layers and neurons)
- Batch size
- Number of epochs
- …



Optimization of a hyper parameter

This can be achieved by using multiple techniques such as :

- Grid search
- Random search
- Gradient based optimization
- **Bayesian optimization**

Introduction to Bayesian Optimization

Like classical optimization : $Min\ f(x)\ , x\ \in\ X$

Bayesian optimization constructs a probabilistic model for $f(x)$ and then exploits this model to make decisions about where in X to next evaluate the function, while integrating out uncertainty.

Use all information of the function.

The 2 choices of Bayesian Optimization :

- Select a prior over functions that will express assumptions about the function being optimized (Gaussian)
- Choose an acquisition function that will determine the next point to evaluate.

Bayesian Optimization's first choice

Gaussian Processes (GP) :

Property : Assume that the GP is a function define by $f : X \rightarrow R$. Any finite set of N points $\{x_n \in X\}_{n=1}^{N}$ induces a multivariate Gaussian distribution on R^N . (Gaussian generalized to N dimensional space).

Then the nth of these points is taken to be the function value $f(x_n)$, hence, by using properties of the Gaussian distribution we can compute marginals and conditionals in closed form.

Suppose that $\begin{bmatrix} x_A \\ x_B \end{bmatrix} \sim N\left(\begin{bmatrix} \mu_A \\ \mu_A \end{bmatrix}, \begin{bmatrix} \Sigma_{AA} & \Sigma_{AB} \\ \Sigma_{BA} & \Sigma_{BB} \end{bmatrix}\right)$, where $x_A \in R^m, x_B \in R^n$.

$$Marginal\ density : p(x_A) = \int_{x_B \in R^n} p(x_A, x_B; \mu, \Sigma) dx_B$$

$$\text{Conditional density: } p(x_A | x_B) = \frac{p(x_A, x_B; \mu, \Sigma)}{\int_{x_B \in R^n} p(x_A, x_B; \mu, \Sigma) dx_B}$$

Bayesian Optimization's second choice

Acquisition Function for Bayesian Optimization :

Assuming $f(x)$ is drawn from a Gaussian process prior and that our observations are of the form $\{x_n, y_n\}_{n=1}^{N}$, where $y_n \sim N(f(x_n), \nu)$ and ν is the variance of noise introduced into the function observation.

Next evaluation of $x \in X$: $x_{next} = argmax_x a(x)$, $a : X \to R^+$ is called the acquisition function.
Acquisition functions depend on the previous observations this dependance is written as : $a(x ; \{x_n, y_n\}, \theta)$

As well for predictive mean and variance function $\mu(x ; \{x_n, y_n\}, \theta)$, $\sigma^2(x ; \{x_n, y_n\}, \theta)$.

Let's :
$x_{best} = argmin_{x_n} f(x_n)$

$\Phi(x)$ the cumulative distribution function (CDF) of the standard normal.

Bayesian Optimization's second choice

There are different types of improvements for the Bayesian steps :

- One intuitive strategy is to maximize the probability of improving over the best current value. Under the GP this can be computed analytically as :

$$a_{PI}(x\,;\,\{x_n, y_n\}, \theta) = \Phi(\gamma(x)),\ \gamma(x) = \frac{f(x_{best}) - \mu(x\,;\{x_n, y_n\}, \theta)}{\sigma(x\,;\{x_n, y_n\}, \theta)}$$

- Alternatively, one could choose to maximize the expected improvement (EI) over the current best value. This also has closed form under the Gaussian process:

$$a_{EI}(x\,;\,\{x_n, y_n\}, \theta) = \sigma(x\,;\,\{x_n, y_n\}, \theta)(\gamma(x)\,\Phi(\gamma(x)) + N\,(\gamma(x)\,;\,0, 1))$$

- GP Upper Confidence Bound A more recent development is the idea of exploiting lower confidence bounds (upper, when considering maximization) to construct acquisition functions that minimize regret over the course of their optimization.

$$a_{LCB}(x\,;\,\{x_n, y_n\}, \theta) = \mu(x\,;\,\{x_n, y_n\}, \theta) - \kappa\,\sigma\,(x\,;\,\{x_n, y_n\}, \theta)$$

Applying Bayesian Optimization to Hyperparameters

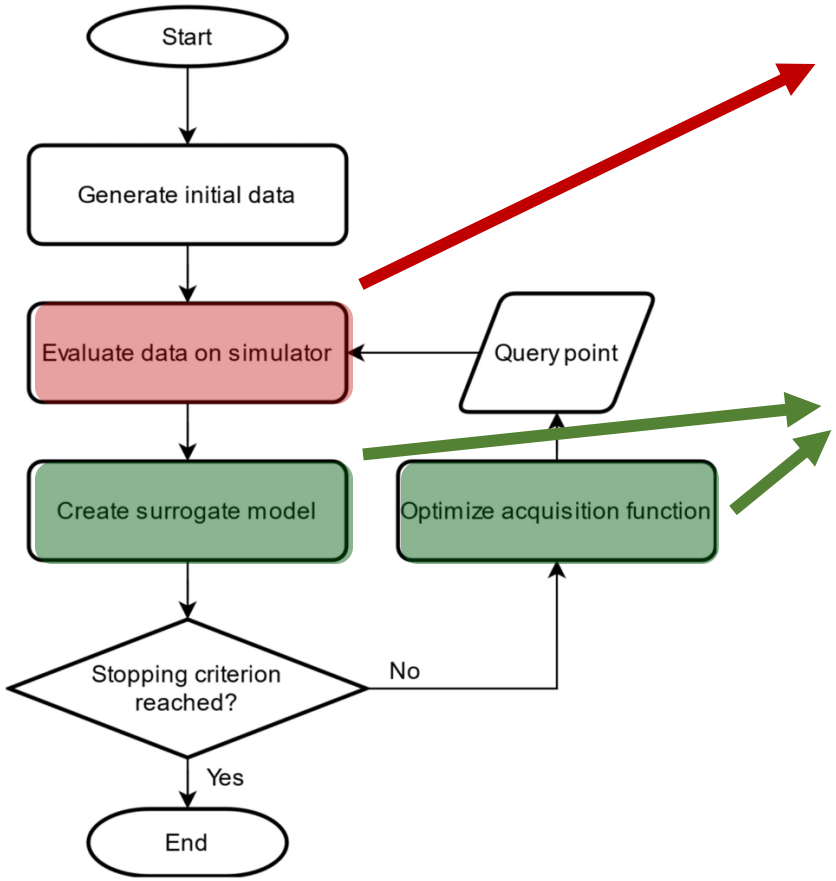
When using Bayesian Optimization for hyperparameter tuning in ML tasks

- High cost at each iteration to assess the result of a predictive set of hyperparameters
- Consider methods to mitigate this cost
- Additional cost to ensure a better guess of hyperparameters is acceptable

3 proposed changes to deal with high cost of each iteration

1. Modification of Gaussian Process and acquisition function
2. Consideration of varying computation time
3. Parallelization of hyperparameter predictions

Modification to Gaussian Process and Acquisition Function



Evaluation of hyperparameter prediction is very expensive

Increasing cost of these tasks will result in better predictions thus less iterations are necessary

Given the high cost of evaluation for ML applications of Bayesian Optimization this tradeoff is beneficial

Image Source: Loka, Nasrulloh & Couckuyt, Ivo & Garbuglia, Federico & Spina, Domenico & Nieuwenhuyse, Inneke & Dhaene, Tom. (2022). Bi-objective Bayesian optimization of engineering problems with cheap and expensive cost functions. Engineering with Computers.

Consideration of Varying Computation Time

In the task of hyperparameter optimization computation time for evaluating sets of hyperparameter can vary

To optimize Bayesian Optimization for hyperparameter tuning

- Greedy approach: prioritize testing hyperparameters with short computation time
- Addition of a cost function for hyperparameters with long computation time
- Cost function implemented through an independent Gaussian Process

Parallelization of Hyperparameter Prediction

To realistically apply Bayesian Optimization to hyperparameter tuning parallelization must be possible.

- Each evaluation point is decided by the acquisition function based on all previous evaluation outcomes
- Must be able to decide the next evaluation point while some evaluations are still pending
- Can estimate acquisition function of pending evaluations with gaussian distributions
- This estimation will lead to reasonable predictions

1st experiment context

Compare Bayesian optimization with **TPA** (Tree Parzen Algorithm) on two problems :

- The Branin-Hoo function
- Logistic regression classification task on MNIST data

There are 4 hyperparameters taken into account :

- **Learning rate** ([0,1] on a log scale)
- **Regularization** ([0,1])
- **Batch size** (20 to 2000)
- **Number of epochs** (5 to 2000)

The Braning-Hoo algorithm was run 100 times and the logic regression was run 10 times.

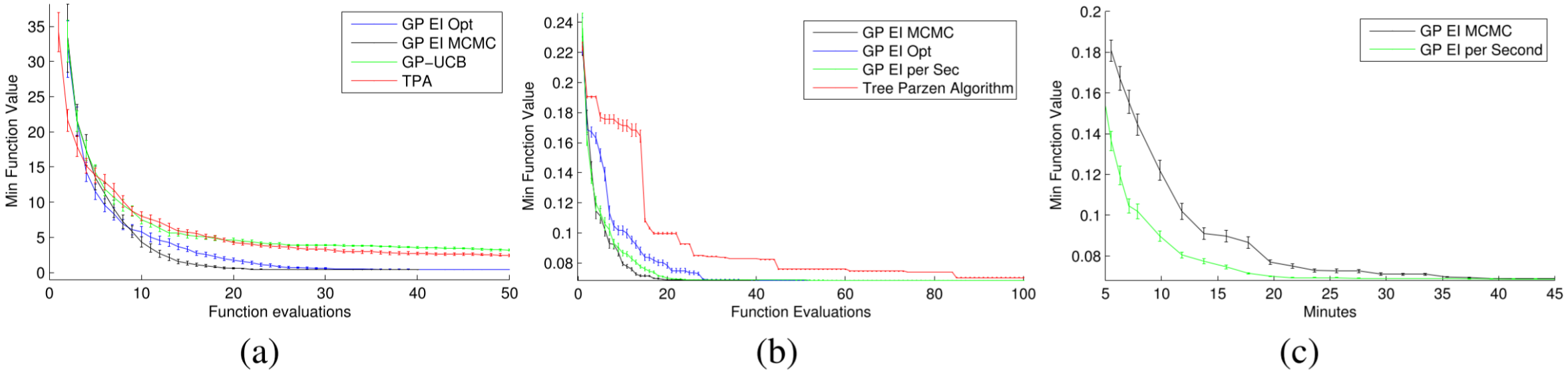


Figure 3: Comparisons on the Branin-Hoo function (3a) and training logistic regression on MNIST (3b). (3c) shows GP EI MCMC and GP EI per Second from (3b), but in terms of time elapsed

2nd experiment context

Compare Bayesian optimization and random grid search on **Online LDA** (Latent Dirichlet Allocation) model

There are 5 hyperparameters taken into account :

- 2 learning parameters τ_0 and κ controlling the **learning rate** $\rho_t = (\tau_0 + t)^{-\kappa}$
- **Mini-batch size**
- η and α the symmetric Dirichlet priors set as $\eta = \alpha = 0.01$ used to categorized the document in a topic

Grid search of size 6 × 6 × 8, for a total of 288 hyperparameter configurations.

Experiment made on a random set of 249 560 Wikipedia articles split into 3 trainings of size 200 000, 24 560 and 25 000.

Documents are represented as vectors of word counts from a vocabulary of 7702 words

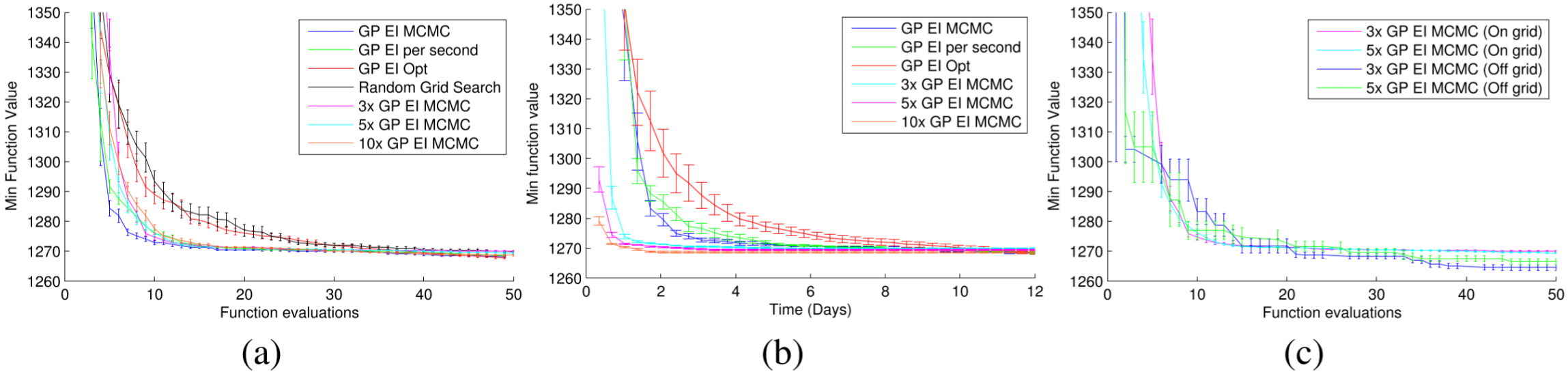


Figure 4: Different strategies of optimization on the Online LDA problem compared in terms of function evaluations (4a), walltime (4b) and constrained to a grid or not (4c).

- (a) and (b) : Each optimization repeated 100 times over the same grid for all strategies
- (c) : Average of 5 runs without restricting the new parameter setting to be on the pre-specified grid

Other experiments

- Comparison with Random Grid search on M3E models
- Comparison on multi-layer convolutional neural networks with a human expert

In both of these experiments, GP model is either faster or more accurate

Conclusion

- With some modification to the traditional Bayesian Optimization algorithm, it can be applied to hyperparameter optimization for ML tasks
- Results showed Bayesian Optimization outperformed previous method such as random search and grid search at hyperparameter optimization in 4 different ML tasks

Our project

We will recreate the Bayesian Optimization algorithm for the hyperparameters on a neural network for image recognition

- Neural Network will use an SGD optimizer
- Optimize the hyperparameters of learning rate and momentum
- Visualize the effect of various sets of hyperparameters compared to the set of optimized hyperparameters