A REVIEW ON HYPER-PARAMETER OPTIMISATION BY DEEP LEARNING EXPERIMENTS

[1]Rohan Bhattacharjee, [2]Debjyoti Ghosh & [3]Abhirup Mazumder

[1,2,3] *Department of Basic Science and Humanities*

Institute of Engineering & Management

Salt Lake Electronics Complex, Kolkata-700091. India

Corresponding Author Email: [3]abhirupmaz@gmail.com

Abstract

It has been found that during the runtime of a deep learning experiment, the intermediate resultant values get removed while the processes carry forward. This removal of data forces the interim experiment to roll back to a certain initial point after which the hyper-parameters or results become difficult to obtain (mostly for a vast set of experimental data). Hyper-parameters are the various constraints/measures that a learning model requires to generalise distinct data patterns and control the learning process. A proper choice and optimization of these hyper-parameters must be made so that the learning model is capable of resolving the given machine learning problem and during training, a specific performance objective for an algorithm on a dataset is optimised. This review paper aims at presenting a Parameter Optimisation for Learning (POL) model highlighting the all-round features of a deep learning experiment via an application-based programming interface (API). This provides the means of stocking, recovering and examining parameters settings and intermediate values. To ease the process of optimisation of hyper-parameters further, the model involves the application of optimisation functions, analysis and data management. Moreover, the prescribed model boasts of a higher interactive aspect and is circulating across a number of machine learning experts, aiding further utility in data management.

Keywords*: Hyper-Parameters, Parameter Optimization for Learning (POL), Algorithm, Optimisation functions, Application-based programming interface (API), Learning model.*

INTRODUCTION

A wide range of parameters is necessary to tweak and optimize machine learning models. Finding the ideal parameter combination is sometimes difficult and time-consuming since parameter optimization necessitates meticulous monitoring of each batch of data obtained during a training update. These findings should also be examined about other hyper-parameter combinations. Hyper-parameter is a variable in machine learning experiments whose value is used to influence the learning process. During training, one must select a collection of these parameters that authorize model parameters to achieve a combination and boost a certain performance objective on a dataset for an algorithm. The most generally used methodologies for HP optimization are grid and manual search, and in both cases, a large number of HP configurations are performed to see how they affect algorithm training to find the ideal parameters. The following are the primary aspects to consider while trying to identify optimal parameter settings:

- Finding the ideal hyper-parameter configuration requires several complete iterations of training, which is frequently a manual and time-consuming approach that lacks empirical rigour.

- The majority of studies preserve interim results in memory and then trash them, Hence it becomes difficult to go back to a previous variable collection in the experiment or to analyse intermediate trainees later.

-As fewer interfaces are established for transferring data mining and machine learning (ML) methods and parameters, circulating the entire assortment of findings gathered through the process is difficult.

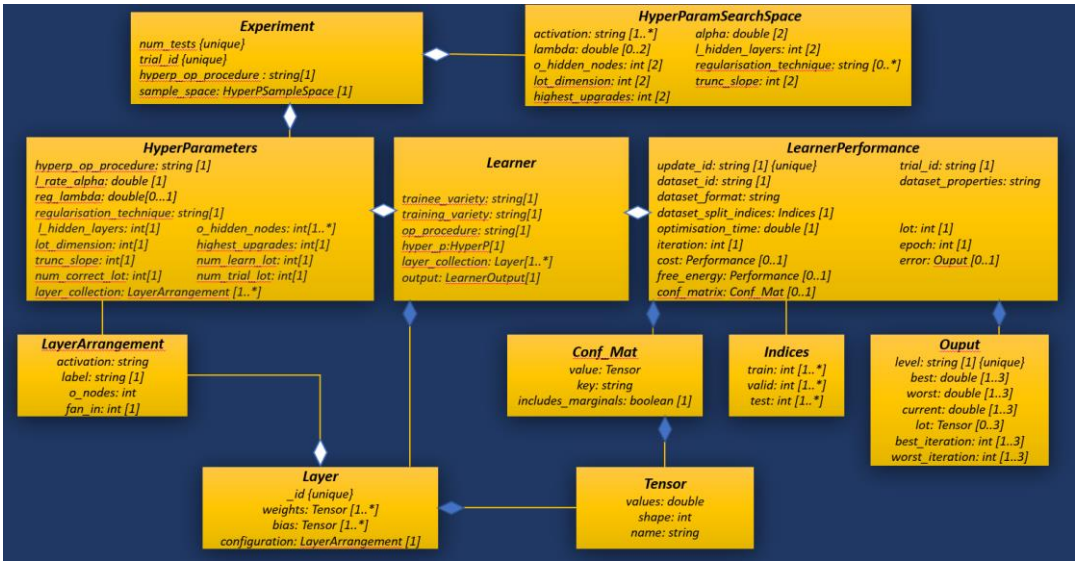
CRISP-DM and SEMMA [1] are two frameworks that have been proposed to resolve the aforementioned issues. However, because these frameworks are abstract, additional fine-grained approaches must be developed before releasing the advantages. To date, ontologies have been developed to represent, for example, machine learning experiments [2] and data mining ideas in general [3]. On the other hand, ontologies are costly to create, frequently tailored to certain fields and need a higher proficiency for ML experts.

The experiment databases which have been stated in [8] and developed in [9], [6], and [7], have been expressed for the sole reason of comparison and do not stress upon deep learning experiments particularly. When compared to the conventional, analogous depository architecture of [7], the provided JSON and NoSQL method is intrinsically more diaphanous and efficient in its structure than the XML-based PMML [4] and represents the branching nature of the aforesaid models.

The proposed model is aimed to represent all the experiment's data attributes. The model has three major components: model parameters, hyper-parameters, and result data superintendence. This experiment, the highest level of abstraction, contains all of the objects and characteristics needed to define a parameter.

MODEL OVERVIEW

The Trial class is the model's entrance, and it has a multifold linkage with the Trainee class, which means that one experiment can include numerous instances of a trainee. The HyperPSampleSpace is present at this level because the search space configuration remains the same throughout the process. The Layer class has a 1-to-many link with the Trainee (one step down from Experiment). The Tensor, Layer and LayerArrangement classes represent variables (weights and biases); the HyperP class represents hyper-parameters; and the TraineeOutput, Output, Conf_Mat, and Tensor classes represent results (output from any iteration of the algorithm).



MODEL SPECIFICATIONS

Space constraints limit a thorough description of the entire model and due to this, a study is presented of two of the most essential classes of the POL data model: Trainee and HyperP.

– *trainee_variety*: The variable identification for the algorithm that was used to make the trainee.
– *training_variety*: The different types of training.

– *development_process*: For example, mini-batch stochastic gradient-descent is an enhancement approach for the learning function (MSGD).

– *hyper_p*: The optimisation process' input and fixed parameters, which are initialized just inside the sample space boundaries.

– *layer_collection*: A collective storage for the objects of Layer that turn characteristics into further abstract characteristics, categories, and prognosis.

– *Output*: A Trainee_Output object that carries a resultant instance after updating completion of training.
– *learned*: A true or false value that indicates whether or not the trainee's model parameters have been optimized.

In the concept, two classes are required to represent hyper-parameter optimisation: HyperPSampleSpace and HyperP. The sample-space class specifies a maximum and minimum limitations for each hyper-parameter, from which num_tests hyper-parameters are created to determine the optimal configuration.Talking about HyperP, which is a solitary arrangement that was formed in the sample area:

– *hyperp_op_procedure*: The variable identification of the hyper-parameter optimization procedure.

– *l_rate_alpha*: The size of variable updates for each stage of GD is determined by the alpha learning rate.

–*reg_lambda*: If there is no regularisation, it sets to zero. It assigns a score for the cases: very high weights and biases and very low weights and biases.

– *regularisation_technique*: The technique of regularisation that has been used in the model.

– *lot_dimension*: The quantity of data lines to be used in Mini-batch Stochastic Gradient Descent that affects the algorithm's training skills. Stochastic Gradient Descent is defined as a size equal to the quantity of training inputs, while Stochastic GD is defined as a size of one.

– *highest_upgrades*: The highest number of Gradient Descent upgrades provided to a training function; boundaries are determined by the number of model parameters and rows in the dataset; used as an exit parameter.

– *trunc_slope:* Indicates the extent to time backpropagation mistakes should be passed.

– *num_learn_lot, num_correct_lot, and num_trial_lot*: Learning, correction, and trial sets each have a different number of lots.

DEPLOYMENT ARCHITECTURE

The system architecture is made up of three layers: data storage, interoperable, and application layers respectively.
Data Storage Layer: The model was executed in JSON and presently uses MongoDB for storing.
This provides a compatible interface to achieve the aims, as mentioned earlier. This connects the JSON API to the NoSQL database (MongoDB).

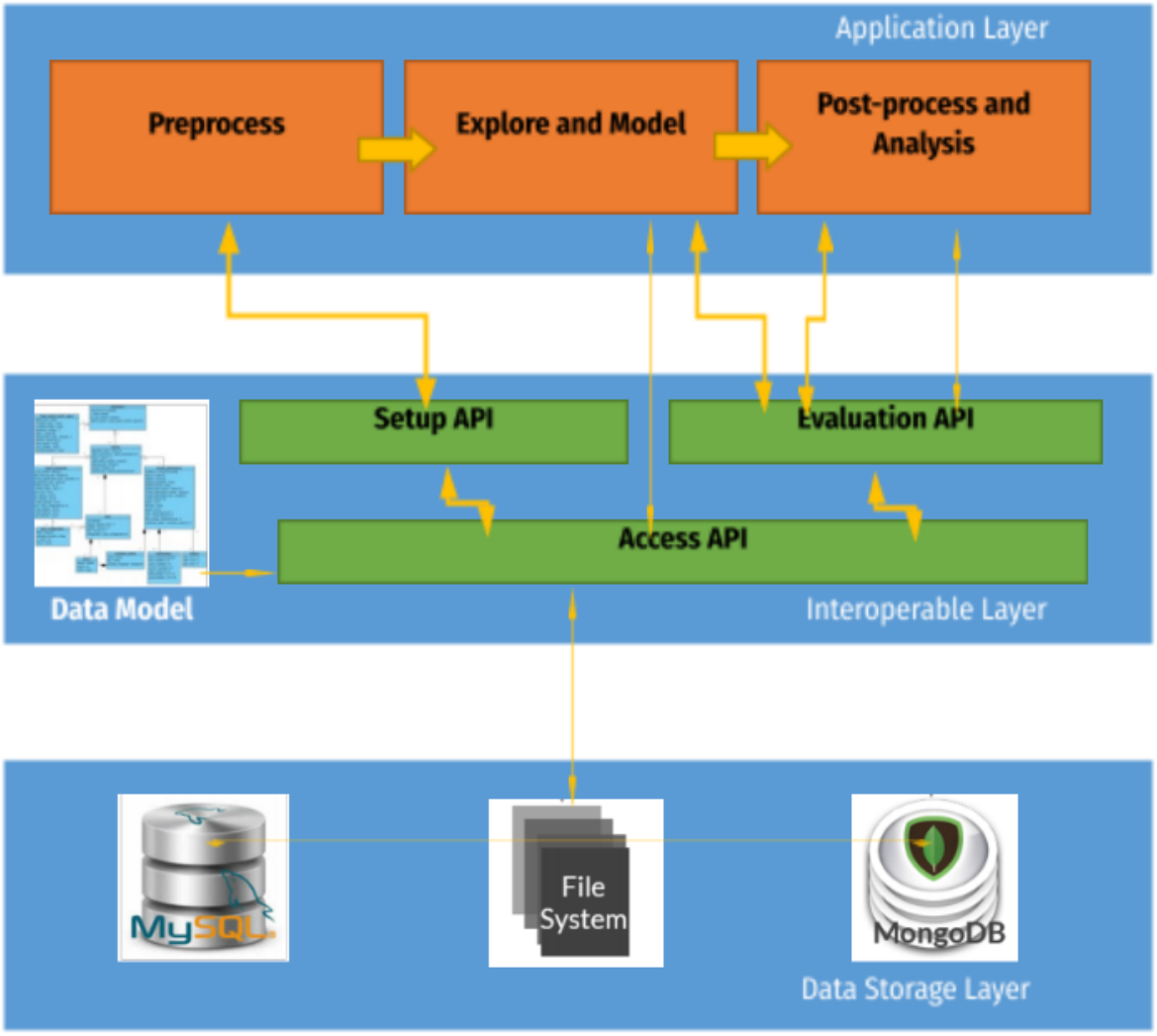
Interoperable Layer: The Interoperable Layer's purpose is to allow for more flexibility in the learning process as well as the easier exchange of outcomes for comparison and analysis. To accomplish these goals, the layer has three libraries:

- Setup library: It comprises methods for initiation, reading in data, and configuring the storage bases so that the instances can be analysed.

- Evaluation library: It comprises methods for analysing and ranking the performances of several trial runs.

- Access library: It hides depository features from the learning process. The Access Interface is a direct application of the model, written in JSON. Before and during a deep learning experiment, this interface accounts for reading and writing characteristics.

Application Layer: The toolkit's primary applications cover many parts of processes, including trial initialization, training and analysis. They use the Python library APIs to manipulate experimental data and construct their trials and analysis routines.



LITERATURE SURVEY: A COMPARATIVE STUDY

We have reviewed some more papers based on the recent developments regarding hyperparameter optimisation. We have studied different methods of optimising hyperparameters, compared them and presented them below.

Bayesian method of optimization-

Bayesian method is one of the most popular optimization design, coupled with Blackbox functions, used for tuning and optimizing hyper-parameters in deep neural networks. This method finds its use in several fields like Image Classification [12, 13], Speech Recognition [14] as well as Neural Language Modelling [15]. This method is encompassed with an iterative algorithm with a probabilistic model and an acquisition method to direct the analysis

of points.

The *expected improvement* (E1) is given by the formula:

$$\mathbb{E}[\mathbb{I}(\boldsymbol{\lambda})] = \mathbb{E}[\max(f_{min} - y, 0)]$$

This can be achieved only if the model prediction y at configuration λ follows a normal distribution given by:

$$\mathbb{E}[\mathbb{I}(\boldsymbol{\lambda})] = (f_{min} - \mu(\boldsymbol{\lambda}))\,\Phi\left(\frac{f_{min} - \mu(\boldsymbol{\lambda})}{\sigma}\right) + \sigma\phi\left(\frac{f_{min} - \mu(\boldsymbol{\lambda})}{\sigma}\right),$$

where Ø(.) and Φ (.) are the standard normal density and standard normal distribution function and $f_{\min}$ is the best observed value.

Gaussian Process (GP):

In model-based optimisation literature, Gaussian Process is considered to be a great approach for building loss functions [13]. In the Gaussian Process, it is advisable to use zero-mean functions. However, generic mean functions can be utilised as well but it might make the process a lot more complicated and insufficient. The simpler approach - the zero-mean process can be obtained by centring the values of the functions in a given set of data. During SMBO, a certain trend in the Gaussian Process may lead to encroachment into some undiscovered areas of the SMBO. The fact that GPs are *closed under sampling* coupled with the ability to predict uncertainty makes it a convenient process for obtaining unknown variables from the SMBO algorithm.

Tree-Structured Parzen Estimator Approach (TPE):

Falling under the category of Sequential Model-Based Optimization (SMBO), this method of approach involves models to be constructed sequentially in order to estimate the performance of hyperparameters on the basis of a test on this model. The main significance of the TPE structure is to bring about a substitution in an equally-biased mix of that prior with gaussians centered at each of its respective elements.

Each Gaussian's standard deviation was set to the greater of the distances to its left and right neighbours, but it was clipped to stay within a reasonable range. The TPE method can also be treated as a method of conditional problem which is mapped by P(x|y) and P(y) where x is being used for hyperparameters and y for it's connected standard marks. The formula stands as follows:

$$\mathrm{EI}_{y^*}(x) = \int_{-\infty}^{y^*} (y^* - y)p(y|x)dy = \int_{-\infty}^{y^*} (y^* - y)\frac{p(x|y)p(y)}{p(x)}dy$$

The main disadvantage of this method is that it will only work when Hyperopt is installed, which requires NumPy.

RESULT AND ANALYSIS

The goal of our review was to show how our interoperable toolbox can be used to organise and

analyse learning trials, not to develop the best accurate model. HP search space optimization is the specific goal. The whole set of hyper-parameter boundaries was fine-tuned by assessing intermediate outcomes overall test-runs. The assessment dataset was created using a collection of sensors worn by athletes during Gaelic Football contests, as reported previously [5]. As part of our HP optimisation approach [10], we ran 90 trials, each with two runs, for a total of 180 trial runs.

Table 1 shows the summary statistics from the experiment. The experiment included 180 test runs with a total of 40,000 epochs iterated. The average size of a trainee and instance was 0.06MB, resulting in a total of 11MB for Test-Runs and roughly 2.5GB for Updations. The interim results of every epoch had been recorded.

Table 1

Compilation	**Frequency**	**Space occupied (in megabytes)**	**Mean Space occupied by object (in megabytes)**
test_runs	180	11	0.06
updations	40,000	2,377	0.06

Activations: The number of Trainees created using ReLUs is comparable to the ones created using Logistic activations. This shows that the two activations perform similarly, however, because ReLU exceeds logistic by a factor of 11:9, reducing sample space determines that Rectified Linear Unit is the better performing term of this category.

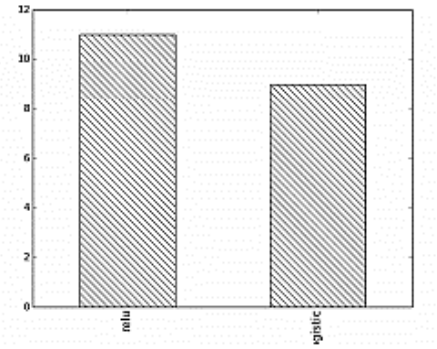


Figure 1

Hidden nodes: Average frequency of the hidden is estimated to be 4.36. The median was determined to be 4. Because the median is lower than the scattering mean, the optimal value for this hyper-parameter is something around 4 or even less, as shown in Figure 2. This indicates that the dormant characteristics that define the inserted data, have a minimum frequency, implying that the dimensionality can be reduced.

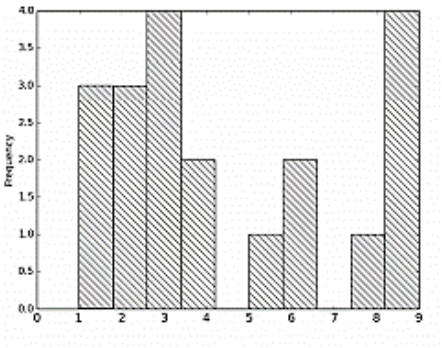


Figure 2

Truncate_gradient: The time-steps count was noted for enhancing backpropagation through time is shown in Figure 3. The truncation gradient value at the 75th percentile is 74.2, with the average at 54 and median at 55, showing a skewed distribution. Around these numbers, distribution concentration occurs around two statements. The time-points spanning around t_{i-55}: t_i causes a larger impact on $t_{i+10,}$ and eventually on the predictions.

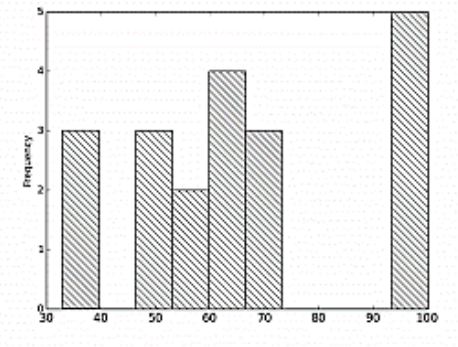


Figure 3

Learning_rate(α): It demonstrates that the median more accurately reflects the central tendency of all parameters Furthermore, as illustrated in the graphs, the scatterings do not aid the evaluation of the parameters. Consequently, we used the statistical methods for individual hyper-parameters formed by coalesce_hyper_parameters to generate a new limitation within the domain (median – standard deviation, median + standard deviation), except for the highest upgrades, where the highest and lowest observations were used since the variables were close to a consistent organisation.

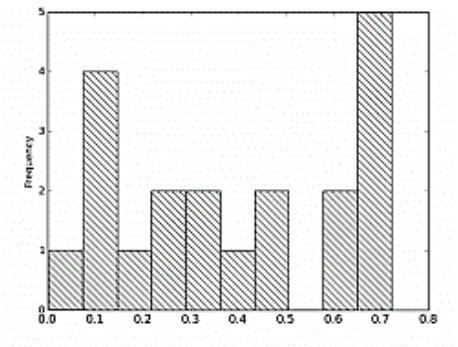


Figure 4

CONTRIBUTION

In this review, the POL data model is presented to encapsulate different perspectives of machine learning processes. The evaluation of the reviewed model gives an overview of the process of achieving a minimized Hyper Parameter sample space. We have gone through several respective topics and have visualized aspects of machine learning experiments thoroughly. The hyper-parameter optimisation procedure will provide many possible ways to ensure greater algorithmic flexibility to ensure efficient programming. A reduced HP-search space will pave the way to reduce the complexity of an algorithm to an extent that will enable even amateurs to take interest in machine learning experiments. In other words, it will provide a new methodology to aid in optimization.

CONCLUSION

We found that the ideal configuration of HPs in larger proportions in the resultant values had a reduced complexity and it optimises the model overall. Because of the above statistics and experimental values, the hyper-parameter search space is decreased, resulting in enhanced model efficiency. More robust techniques of empirically decreasing the search boundaries will be the subject of future research. In addition, the authors are now doing experiments on two more datasets with the goal of comparing their methodology to other comparable frameworks.

REFERENCES

1. Azevedo, A., Santos, M.F.: KDD, SEMMA and CRISP-DM: a parallel overview. In:IADIS European Conference on Data Mining 2008, Amsterdam, The Netherlands,July 24-26, 2008. Proceedings. pp. 182-185 (2008)

2. Vanschoren, J., Soldatova, L.: Expose: An ontology for data mining experiments.In: International workshop on third generation data mining: Towards service-oriented knowledge discovery (SoKD-2010). pp. 31-46 (2010)

3. Keet, C., dAmato, C., Khan, Z., Lawrynowicz, A.: Exploring reasoning with the DMOP ontology. In: 3rd Workshop on Ontology Reasoner Evaluation (ORE14).vol. 1207, pp. 64{70 (2014)

4. Nurseitov, N., Paulson, M., Reynolds, R., Izurieta, C.: Comparison of JSON and XML data interchange formats: A case study. Caine 2009, 157-162 (2009)

5. O'Donoghue, J., Roantree, M., Cullen, B., Moyna, N., Sullivan, C.O., McCarren, A.: Anomaly and event detection for unsupervised athlete performance data. In:Proceedings of the LWA 2015 Workshops, Trier, Germany, October 7-9, 2015. pp.205-217 (2015)

6. Vanschoren, J., Blockeel, H., Pfahringer, B., Holmes, G.: Experiment databases. Machine Learning 87(2), 127-158 (2012)

7. Vanschoren, J., van Rijn, J.N., Bischl, B.: {Taking machine learning research online with OpenML}. In: Proceedings of the 4th International Workshop on Big Data, Streams and

Heterogeneous Source Mining: Algorithms, Systems, Programming Models and Applications. pp. 1-4 (2015)

8. Blockeel, H.: Experiment databases: A novel methodology for experimental research. In: Knowledge discovery in inductive databases, pp. 72-85. Springer (2005)

9. Blockeel, H., Vanschoren, J.: Experiment databases: Towards an improved experimental methodology in machine learning. In: Knowledge Discovery in Databases: PKDD 2007, pp. 6-17. Springer (2007)

10. Bergstra, J., Bengio, Y.: Random search for hyper-parameter optimization. J.Mach. Learn. Res. 13, 281-305 (Feb 2012)

11. J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. JMLR, 2012.

12. Snoek, J., Larochelle, H., Adams, R.: Practical Bayesian optimization of machine learning algorithms. In: Bartlett et al. [9], pp. 2960–2968

13. Snoek, J., Rippel, O., Swersky, K., Kiros, R., Satish, N., Sundaram, N., Patwary, M., Prabhat, Adams, R.: Scalable Bayesian optimization using deep neural networks. In: Bach and Blei [7], pp. 2171–2180

14. Dahl, G., Sainath, T., Hinton, G.: Improving deep neural networks for LVCSR using rectified linear units and dropout. In: Adams, M., Zhao, V. (eds.) International Conference on Acoustics, Speech and Signal Processing (ICASSP'13). pp. 8609–8613. IEEE Computer Society Press (2013)

15. Melis, G., Dyer, C., Blunsom, P.: On the state of the art of evaluation in neural language models. In: Proceedings of the International Conference on Learning Representations (ICLR'18)

16. J. Mockus, V. Tiesis, and A. Zilinskas. The application of Bayesian methods for seeking the extremum. In L.C.W. Dixon and G.P. Szego, editors, Towards Global Optimization, volume 2, pages 117–129. North Holland, New York, 1978.