ESANN 2019 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2019, i6doc.com publ., ISBN 978-287-587-065-0. Available from http://www.i6doc.com/en/.

Lightweight autonomous bayesian optimization of Echo-State Networks

Luca Cerina[1], Giuseppe Franco[2], Marco D. Santambrogio[1]

1- Politecnico di Milano - Dip. di Elettronica, Informatica e Bioingegneria
Piazza Leonardo da Vinci, Milano - Italy
2- Scuola Superiore Sant'Anna & Università di Pisa, Pisa

Abstract. Echo State Networks (ESN) represent a good option to tackle non-linear, time-dependent problems without the training complexity of standard Recurrent Neural Networks (RNNs), thanks to intrinsic dynamics that arise from untrained sparse networks. However, performance and stability of ESN are determined by their hyper-parameters, e.g. Reservoir dimension and sparsity, and the characteristics of the input, whose optimal values required time consuming procedures to be found. Here we propose an efficient automatic optimization framework for ESN based on the Bayesian Optimization given user-defined objectives, and bounded ranges on hyper-parameters. Results shown performance comparable with exhaustive grid-search optimization algorithms.

1 Introduction

Recently, methodologies based on Neural Networks capable of embedding temporal dynamics, namely Recurrent Neural Networks (RNNs) and Reservoir Computing (RC), started to tackle time-series problems. In this scenario, RC-based Echo State Networks (ESN) [1] machines proved to be extremely valuable in modeling nonlinear time-series. ESN exploit their sparse *reservoir* to project the input in a non-linear, high dimensional, state space and then apply simpler linear models to perform inference on the internal state.

In certain tasks, ESN outperforms regular RNNs for accuracy and training efficiency, although they have more stringent requirements on stability, calculated using the Echo State Property (ESP) [2] or the Lyapunov Exponents [3], due to the intrinsic non-linearity of the reservoir. Furthermore, performance such as network accuracy, memory, and computational efficiency are dependent from a variety of hyper-parameters that are problem-specific given the input dynamics and should be determined before the training. The optimization of these parameters is a time consuming process done by the user through trial-and-error [4] or by sampling points in the configuration space, both randomly, with possibly sub-optimal solutions, or through exhaustive grid-search, which grows exponentially with N_{params}.

Conversely, automatic optimization could be performed using different methods, e.g. gradient descent [5] or genetic algorithms [6]. Another method exploits Bayesian Optimization (BO) [7] to perform robust RC optimization: given a fitness function $\mathrm{E}(\mathbf{x})$, BO tries to find its best approximation and search the maximum on that, determining automatically the points in the parameter space that could be closer to the optimum.

In this paper we propose a lightweight BO framework for ESN built on the LIMBO library [8] which trains the model starting from a high-level abstract model: the user defines only the Training/Validation/Test sets, an error function selected among many, and the parameters, that can be fixed, or bounded inside a range if needed to be optimized. Compared to the method in [7], it does not require to re-implement part of the ESN code to fit the specific problem, but embeds a set of ESN models (e.g. standard and leaky integrator, with different cost functions) which are chosen by the user in the configuration process.

The computing architecture, written in C++ to focus on computational efficiency and reduced memory footprint, samples different points selected by the BO until it reaches the best configuration or a maximum number of iterations. Furthermore, the computing architecture (based on Eigen[1]) can be tuned to run on multi-core machines, or *edge* devices with limited resources, allowing the user to deploy the ESN near the source of data.

The framework is validated against multiple benchmark datasets and obtained performances comparable with the State-of-the-Art (SoA) and shows an exponential speed-up compared to Grid Search.

2 Bayesian optimization of ESN

In ESN the network consists of a *reservoir*, a large set of sparsely connected non-linear units, whose state is used for inference by means of a linear readout mechanism, and their efficiency lies in the possibility to train just the weights associated to the readout layer, granted that the reservoir is stable under the Echo State Property, and leave the reservoir untrained. Compared to other neural nets, ESN are characterized by a lower set of hyper-parameters to be optimized, but their impact on the result (e.g. the network accuracy) is difficult to discern in an analytical way for every problem presented to the network.

Given $E(\mathbf{x})$ as our fitness function (e.g. the network accuracy), we search the vector of hyper-parameters $\mathbf{x}$ that maximizes $E(\mathbf{x})$ and drive the ESN in an optimal state. The simplest method is Grid Search, which evaluates $E(\mathbf{x})$ on the entire space of parameters, but it has many drawbacks, especially regarding the computational cost in terms of resources, time, memory and it considers each sample alone without the *knowledge* that we gain after testing a new set of hyper-parameters $\mathbf{x_n}$. On the contrary, BO builds and internal approximation of the fitness function, exploiting the prior information about $E(\mathbf{x})$, $p(E)$, and the set of observations that collected up to now $D = \{\mathbf{x_1}...\mathbf{x_n}, \mathbf{y_1}...\mathbf{y_n}\}$, through the likelihood function $p(D|E)$. Exploiting the Bayes Theorem, we can obtain the posterior distribution of $E(\mathbf{x})$ as:

$$p(E|D) \propto p(D|E) * p(E) \tag{1}$$

With each new sample, BO reaches a better representation of the shape of $E(\mathbf{x})$ and thus can make a more accurate guess of the location of the function's maximum. A detailed explanation can be found in [9].

[1]eigen.tuxfamily.org

If we consider the architecture of an ESN in its most simple form [5], it consists of $\mathbf{W_{in}}$ of NxK input weights, the reservoir $\mathbf{W_r}$ with NxN units, and the UxL readout $\mathbf{W_{out}}$.

At each time-step, the network updates as:

$$\begin{cases} \mathbf{x}(t) = (1-a) * \mathbf{x}(t-1) + a * f(\mathbf{W_{in}}\mathbf{u}(t) + \mathbf{W_r}\mathbf{x}(t-1)) \\ \mathbf{y}(t) = \mathbf{W_{out}}\mathbf{x}(t) \end{cases} \tag{2}$$

Where f(·) is a sigmoid activation function ($\equiv$ *tanh* in our case), and a is the *leaky factor* $\in [0, 1]$, that allows to control the speed in the reservoir dynamics. Small values of a mean that more time is required to adapt the reservoir to changes in the input. The most important parameter to control is the ESP [1, 2], which grants that every dependence from the initial conditions is progressively lost, and that $\mathbf{W_r}$ asymptotically drives the system to a stable state dependent only on the input sequence $\mathbf{u}$. A practical condition for the ESP can be formulated as $\rho(\mathbf{a} * \mathbf{W_r} + (\mathbf{1} - \mathbf{a}) * \mathbf{I}) < 1$, where $\rho(\cdot)$ represents the spectral radius of the matrix. Although this condition doesn't ensure the ESP, it has been shown that in practice is sufficient in order to obtain a stable system [4, 10].

Other hyper-parameters that drive the network performance are the number of reservoir units N, the sparsity of the matrix $\mathbf{W_r}$, and the number of samples washed away to ignore initial conditions. $\mathbf{W_{in}}$ is initialized from a random uniform distribution in the range $[-W_{in}scale, W_{in}scale]$, another hyper-parameter of the network; in our proposal they are all defined by bounded ranges (e.g. $N[100, 1000]$). An acquisition function decides which point should be evaluated, to determine the balance between exploration (i.e. searching points in space with less information) vs exploitation (i.e. searching regions of space that looks more promising).

3 Proposed solution

We propose here a standalone C++ library with high abstraction configurations that performs Bayesian Optimization for Echo State Networks, and provide plug interfaces to deploy trained models. An overview of the system in visible in Fig. 1. The Bayesian Optimizer exploits the Limbo Library [8], enclosing a custom parametrized ESN library built upon Eigen (eigen.tuxfamily.org) a fast templated library for linear algebra.

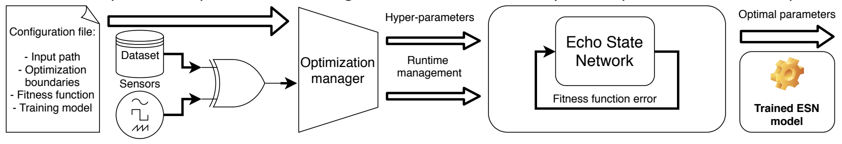

Fig. 1: Graphical depiction of the architecture. The user inputs data along with an abstraction of the problem to generate a likely optimal configuration.

ESANN 2019 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2019, i6doc.com publ., ISBN 978-287-587-065-0. Available from http://www.i6doc.com/en/.

3.1 Configuration and I/O module

As visible in Fig. 1, the library allows a general data source for pre-built datasets (machine learning or prototyping), and other sources that employ the system online. The dataset on disk must be divided in Training, Test and Validation set, in order to build the network, evaluate the training error and avoid overfitting. It also requires a YAML configuration file, where the user specifies the data and the parameters used in the optimization. Currently the configuration contains the ESN parameters optimized by Limbo, the range values where to search or their fixed value, the type of problem learned by the ESN (i.e. regression vs classification), and other stopping conditions based on the Fitness reached or the Number of Iterations performed.

The *optimization manager* (OM) acts as the frontend, passing the configuration parameters to the optimizer and managing the runtime. For example, given a set of inputs, the user could obtain an ESN that consider them altogether, or optimize multiple models for each problem. The OM controls also the level of parallelization of the system, defining the number of threads used in the initial sampling and in the computation of states.

When the optimization process ends, the system provides a set of files with optimized hyper-parameters and pre-loaded portable weights, plus the observations and outputs generated in the BO, useful for further analyses.

3.2 Computational optimizations

Multiple strategies have been adopted in order to speed-up some of the computation during the training of the network. A fast solver for sparse matrices is used to scale $\mathbf{W_r}$ initialization to the desired ρ over the spectral radius, to ensure the ESP. The update of states of ESN is handled using Eigen SpMV (Sparse Matrix multiplies Dense Vector) optimization. Instead, the computation of readout regression W_{out} is calculated using a Full-Pivot QR solver, which is nearly 2x faster than Moore-Penrose and more robust than pseudo-inversion.

Both in ESN state computations and in the initial random sampling the CPU multi-threading is exploited to reduce the overall computational time.

4 Results and discussion

To validate our system, we optimized a ESN against common benchmark for chaotic time-series, the Santa Fe Laser Data [11], NARMA10 emulation [12], and the prediction of Rossler attractors, useful to evaluate the optimization against multiple outputs. All benchmark tasks required the prediction of one step ahead in time, but the system allows prediction on an arbitrary number of steps in the future. All the tests were run multiple times on a desktop machine: Intel i7-6700@3.4GHz, 32GB DDR4 RAM and compiled with maximum optimizations and OpenMP support.

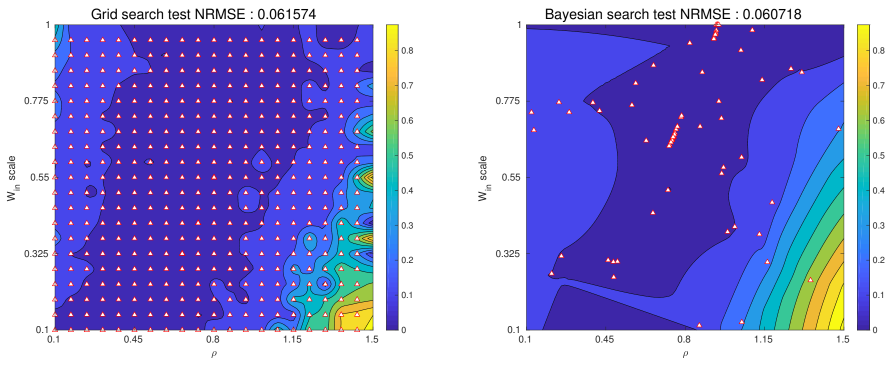

Fig. 2: Parameter Space Exploration of input scale and ρ, with Grid Search vs Bayesian Optimization on Laser Test set.

4.1 Comparison with Grid Search Optimization

In Fig. 2 it is possible to see the Parameter Space Exploration with Grid Search and BO, using NRMSE as $1 - E(\mathbf{x})$. Through BO we can obtain a very good approximation of the parameters' space, combining a smart sampling that ignores high-error regions (in Fig. 2 high ρ and small $W_{in}scale$) while focusing on more promising area. Another test with 3 parameters obtained a substantial speed-up: 13 seconds of BO compared to nearly 44 minutes for Grid Search. It is important to remark that in a real example the time required for Grid Search would increase exponentially, while the BO needs a small amount of iterations (fixed to 40 in Fig. 2) to reach a maximum of $E(x)$.

4.2 Comparison with State of the Art Optimization

The comparisons with the SoA, which comprises various ESN architectures, has been carried out using the Normalized Root Mean Squared Error (NRMSE) as $1 - E(\mathbf{x})$ and optimizing for the following parameters: size and density of $\mathbf{W_r}$, spectral radius $\rho(\cdot)$, input scaling, washout samples and leaky factor a.

Table 1: Prediction performance NRMSE against SoA implementations.

Task	**SoA**	**Our proposal**
Narma10	0.0833±0.0295 [7]	0.0817±0.0114
Laser	0.0060 [13]	0.0057
Rossler	4.647e-05 [14]	6.48e-12

It is visible from Table 1 that our proposal is comparable or better than those available in the SoA (using similar testing settings), using a *standard* ESN optimized along different parameters.

5 Conclusions

In this paper, we presented a framework for the autonomous bayesian optimization of echo-state networks. The proposed system is easily configurable and does not require a specific knowledge of the underlying architecture, while being able to achieve performance comparable to those in the State of the Art. The specific implementation is portable, efficient in both computation and memory an it is suitable also for low-end devices.

Current limitations include the absence of other sigmoid activation functions (e.g. logistic sigmoid), other Echo-State network topologies such as Deep Reservoir or Cyclic Reservoir and modular implementation of $E(x)$. These extensions will be added in future developments. Other improvements will include the optimization of parameters with respect to memory and Lyapunov's stability instead of Echo-State Property.

References

[1] Herbert Jaeger. The echo state approach to analysing and training recurrent neural networks-with an erratum note.

[2] Mantas Lukoševičius and Herbert Jaeger. Reservoir computing approaches to recurrent neural network training. *Computer Science Review*, 3(3):127–149, 2009.

[3] Claudio Gallicchio, Alessio Micheli, and Luca Silvestri. Local lyapunov exponents of deep echo state networks. *Neurocomputing*, 298:34–45, 2018.

[4] Mantas Lukoševičius. A practical guide to applying echo state networks. In *Neural networks: Tricks of the trade*, pages 659–686. Springer, 2012.

[5] Herbert Jaeger, Mantas Lukoševičius, Dan Popovici, and Udo Siewert. Optimization and applications of echo state networks with leaky-integrator neurons. *Neural networks*, 20(3):335–352, 2007.

[6] Aida A Ferreira, Teresa B Ludermir, and Ronaldo RB De Aquino. An approach to reservoir computing design and training. *Expert systems with applications*, 40(10), 2013.

[7] Jan Yperman and Thijs Becker. Bayesian optimization of hyper-parameters in reservoir computing. *arXiv preprint arXiv:1611.05193*, 2016.

[8] A. Cully, K. Chatzilygeroudis, F. Allocati, and J.-B. Mouret. Limbo: A Flexible High-performance Library for Gaussian Processes modeling and Data-Efficient Optimization. *The Journal of Open Source Software*, 3(26):545, 2018.

[9] Eric Brochu, Vlad M Cora, and Nando De Freitas. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv:1012.2599*, 2010.

[10] Gilles Wainrib and Mathieu N Galtier. A local echo state property through the largest lyapunov exponent. *Neural Networks*, 76:39–45, 2016.

[11] Neil A Gershenfeld and Andreas S Weigend. The future of time series. Technical report, Xerox Corporation, Palo Alto Research Center, 1993.

[12] Amir F Atiya and Alexander G Parlos. New results on recurrent network training: unifying the algorithms and accelerating convergence. *IEEE transactions on neural networks*, 11(3):697–709, 2000.

[13] Davide Bacciu and Andrea Bongiorno. Concentric ESN: assessing the effect of modularity in cycle reservoirs. *CoRR*, abs/1805.09244, 2018.

[14] Hoang Minh Nguyen, Gaurav Kalra, Tae Joon Jun, and Daeyoung Kim. A novel echo state network model using bayesian ridge regression and independent component analysis. In *International Conference on Artificial Neural Networks*, pages 24–34. Springer, 2018.