Software Impacts 17 (2023) 100526

Contents lists available at ScienceDirect

Software Impacts

journal homepage: www.journals.elsevier.com/software-impacts

Original software publication

SOMO-VCB: A Matlab® software for single-objective and multi-objective optimization for variance counterbalancing in stochastic learning

Dimitra G. Triantali *, Konstantinos E. Parsopoulos, Isaac E. Lagaris

Department of Computer Science and Engineering, University of Ioannina, GR-45110 Ioannina, Greece

ARTICLE INFO

Keywords:
Variance counterbalancing
Stochastic learning
Neural networks
Single-objective optimization
Multi-objective optimization
Matlab®

ABSTRACT

Variance CounterBalancing (VCB) is an effective neural network training method that outperforms traditional stochastic gradient descent techniques. By minimizing both the average and the variance of the network's error over random mini-batches, VCB improves generalization scores and solves the diminishing step-size issue. The SOMO-VCB software is a Matlab® implementation of VCB that can be used for regression problems as a single-objective or multi-objective optimization task. The code is easy to modify and flexible, using only essential Matlab® programming. Recent experiments show that SOMO-VCB is highly competitive compared to state-of-the-art methods like Adam.

Code metadata

Current code version	v1.0
Permanent link to code/repository used for this code version	https://github.com/SoftwareImpacts/SIMPAC-2023-214
Permanent link to Reproducible Capsule	https://codeocean.com/capsule/9169054/tree/v1
Legal Code License	MIT License
Code versioning system used	git
Software code languages, tools, and services used	MathWorks Matlab®
Compilation requirements, operating environments & dependencies	Matlab® R2019a, Ubuntu 18.04
If available Link to developer documentation/manual	ExemplarManual
Support email for questions	d.triantali@uoi.gr

1. Introduction

Artificial neural networks have been placed in a salient position among machine learning methods for solving challenging problems in science and engineering. The stochastic gradient descent (SGD) methods, which constitute the main optimization artillery for training neural networks, are often characterized by slow convergence rates, especially in large datasets. This is a consequence of considering the mean squared error (MSE) of the network over individual mini-batches randomly selected from the training dataset.

The VCB method was proposed [1] as an alternative approach that employs random sets of mini-batches and concurrently minimizes the average and the variance of the network's MSE. Thus, if w is the parameter vector of the neural network (i.e., synapses weights and other possible parameters of its activation functions), then the average, $\bar{E}(w)$, and the variance, $\bar{\sigma}^2(w)$, of the network's MSE over randomly selected sets of mini-batches are used to form the objective function for the network's training. The main gain of this approach is the alleviation of slow convergence through the use of more efficient optimizers, such as quasi-Newton methods, as well as the enhanced generalization capability of the network.

In its early variants, VCB followed a single-objective (SO) optimization approach [1]. Thus, the optimization problem was formulated through the penalty function:

$$\min_{w \in W} \; \bar{F}(w, \lambda_c) = \bar{E}(w) + \lambda_c \, \bar{\sigma}^2(w), \tag{1}$$

where λ_c stands for the penalty coefficient that needs to be properly set. The problem was solved using the BFGS algorithm with strong Wolfe-Powell line search conditions [2]. Recently, the inherent bi-objective nature of VCB has motivated a multi-objective (MO) formulation of the problem [3] based on the minimization of the vectorial objective

The code (and data) in this article has been certified as Reproducible by Code Ocean: (https://codeocean.com/). More information on the Reproducibility Badge Initiative is available at https://www.elsevier.com/physical-sciences-and-engineering/computer-science/journals.

* Corresponding author.
E-mail addresses: d.triantali@uoi.gr (D.G. Triantali), kostasp@uoi.gr (K.E. Parsopoulos), lagaris@uoi.gr (I.E. Lagaris).

https://doi.org/10.1016/j.simpa.2023.100526
Received 22 May 2023; Received in revised form 7 June 2023; Accepted 7 June 2023

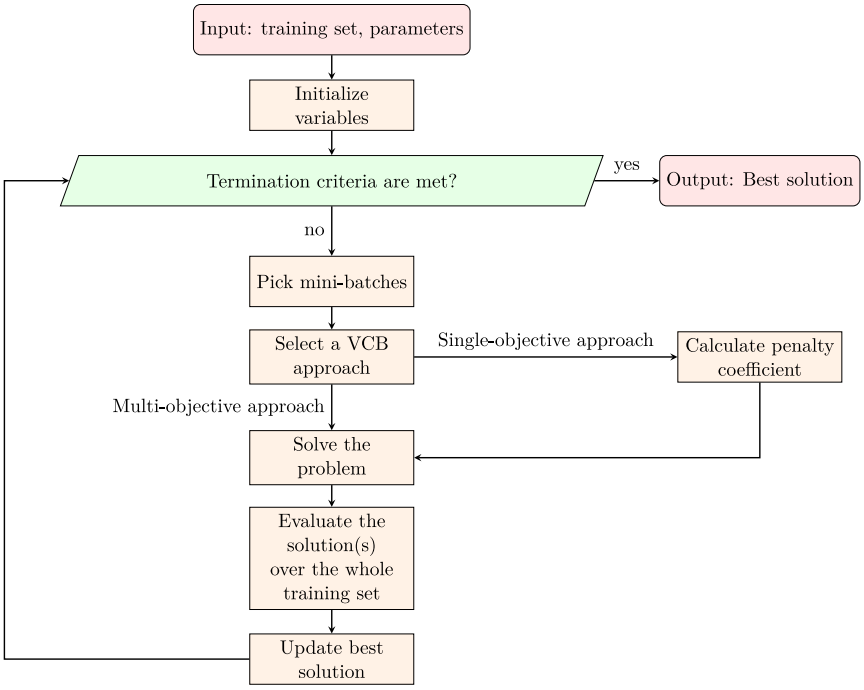

Fig. 1. Flowchart of the Variance CounterBalancing method.

function:

$$\bar{F}(w) = \left[\bar{E}(w) \;\; \bar{\sigma}^2(w)\right]^T, \tag{2}$$

whose Pareto optimal solutions can be detected through state-of-the-art metaheuristics, such as the multi-objective particle swarm optimization (MOPSO) method [4].

The proposed SOMO-VCB software implements both the single-objective and the multi-objective VCB approaches, especially designed for function approximation tasks using RBF neural networks [5]. The recently published proof-of-concept results in [3] verify the feasibility of both approaches as well as their potential in diverse regression tasks. A flowchart of the implemented VCB method is shown in Fig. 1.

Description of the proposed software

The proposed software is implemented in MathWorks Matlab® (https://www.mathworks.com/), a high-level language and interactive environment, and it has been tested in the R2019a release. No existing codes or specialized toolbox functions were used in order to retain source-code portability and completeness. Note that the proposed software can also be easily adapted to run in GNU Octave (https://octave.org/), an open-source alternative software with similar capabilities and program language syntax with Matlab®. Our software was tested under Ubuntu 18.04 since the Linux operating system is the most popular one running on scientific-oriented systems. Thus, Linux/Unix naming conventions have been used. Nevertheless, adaptation in Microsoft Windows® environments requires only minor effort.

The SOMO-VCB software is organized within a single main folder named `SOMO-VCB/`, as shown in the folder tree of Fig. 2. The source code is implemented in m-files, which is the file type of Matlab®, and is stored in the `code/`folder. The files are divided into two groups: the first group implements the SO approach and includes the files with the "`so_`" prefix, while the second group implements the MO approach and refers to files with the "`mo_`" prefix. The contents of each file are briefly outlined below:

(1) `so_vcb_main.m`: This is the main file of the SO approach. It defines all global variables and parameters and implements the VCB method. Also, it is responsible for reading input data and writing output results.

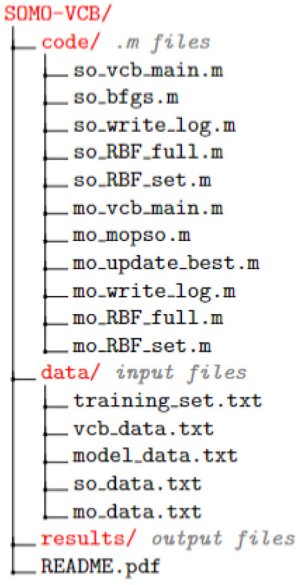

Fig. 2. Organization of files and folders of the source code of the SOMO-VCB software.

(2) `so_bfgs.m`: It contains the BFGS optimizer with Wolfe-Powell line search. The file is self-contained, including all its relevant functions.

(3) `so_write_log.m`: This file contains a function that writes all input variables in a log file for verification purposes. Also, it checks the input for infeasible values, producing corresponding error messages. The user can straightforwardly expand the provided error-traps list.

(4) `so_RBF_full.m`: This is a function that computes the MSE of the RBF network over the complete training set. This result is required to assess the best solution at the end of each VCB cycle.

(5) `so_RBF_set.m`: This file includes a function that calculates the average value and the standard deviation of the MSE of the RBF network over the current set of mini-batches. It is required for objective function and gradient evaluations.
(6) `mo_vcb_main.m`: This is the main file of the MO approach. It contains all global variables, parameters, and the VCB algorithm. Also, it is responsible for reading input data and writing the results.
(7) `mo_mopso.m`: This file includes the MOPSO algorithm with all its relevant functions.
(8) `mo_update_best.m`: This is a function for evaluating each solution in the detected Pareto set according to its MSE over the whole training set. It retains the overall best solution detected so far.
(9) `mo_write_log.m`: This file writes all input variables in a log file. It also checks for faulty input values, producing corresponding error messages. It can be easily extended with additional error traps.
(10) `mo_RBF_full.m`: This function calculates the MSE of the RBF network over the complete training set. It is required to assess the best solution at the end of each VCB cycle.
(11) `mo_RBF_set.m`: This function computes the average and the standard deviation of the MSE of the RBF network over the current set of mini-batches. It is required for objective function evaluations.

The `SOMO-VCB/` folder also includes two folders needed for running the provided software:

(1) `data/`: This folder contains all the input data needed for the algorithms. More specifically, the folder comprises the following files:
   (a) `training_set.txt`: This is the complete training set. Each row contains an n-dimensional training vector x_i followed by its correct output y_i, all values separated by spaces.
   (b) `vcb_data.txt`: It determines the VCB parameters.
   (c) `model_data.txt`: This file provides information about the selected neural network model.
   (d) `so_data.txt`: It contains the parameters of the SO approach.
   (e) `mo_data.txt`: It contains the parameters of the MO approach.

   The above files are either VCB-related or optimizer-related. The VCB-related files are common for both the SO and MO approach. The user can appropriately modify their context to further adjust the software to the problem of interest.
(2) `results/`: This is the folder where all results are stored. Running either the software's SO or MO version produces three output files. Depending on the specific approach, the output files may have either the "`so_`" or the "`mo_`" prefix in their names but the same type of content. Below we denote the prefix as "`*`" when irrelevant.
   (a) `*_log`: This file contains all parameter names (as appear in the software) and their assigned values, properly categorized (model-related, algorithm-related, etc.).
   (b) `*_report`: This is the main output file. It contains one line per experiment, where each line includes:
      i. the number of the current experiment,
      ii. the value of the full MSE over the whole training set for the overall best solution of the experiment,
      iii. the number of VCB cycles performed in the experiment,
      iv. the number of total network evaluations performed in the experiment,
      v. the running time spent for the specific experiment.

      This information is organized in columns in order to facilitate the post-processing of the results.
   (c) `*_solution`: This file contains the solution vectors of the corresponding experiments. Each line contains the corresponding experiment's number and the complete solution vector.

The purpose of the proposed software lies in producing the results files. Depending on the specific application and the targeted desirable analysis, the user can then apply external post-processing procedures to the obtained solutions.

Finally, within the `SOMO-VCB/` folder, the reader will find the `README.pdf` file that provides a concise overview of all the provided files.

Software impacts

Efficient training algorithms are crucial for handling big data in demanding applications. Our proposed SOMO-VCB software implements the two major VCB variants, offering comprehensiveness and efficiency. In order to promote user-friendliness for the non-expert users, we avoided using complex syntax that could make the code difficult to understand. This means that some iterative procedures could be further optimized by utilizing certain features of the Matlab® programming language. The simplicity of the proposed code can motivate researchers to explore its functionalities and even extend it for their own investigations.

The SOMO-VCB software has been recently validated by Triantali et al. [3], demonstrating the superiority of VCB approaches against state-of-the-art SGD methods such as Adam. In future development, our goal is to expand the software by integrating additional solvers (e.g., NSGA-II [6], L-BFGS [7]), as well as diverse neural network architectures that may be used in different applications.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: We acknowledge support of this work by the project "Dioni: Computing Infrastructure for Big-Data Processing and Analysis" (MIS No. 5047222) which is implemented under the Action "Reinforcement of the Research and Innovation Infrastructure", funded by the Operational Programme "Competitiveness, Entrepreneurship and Innovation" (NSRF 2014–2020) and co-financed by Greece and the European Union (European Regional Development Fund).

Acknowledgments

We acknowledge support of this work by the project "Dioni: Computing Infrastructure for Big-Data Processing and Analysis" (MIS No. 5047222) which is implemented under the Action "Reinforcement of the Research and Innovation Infrastructure", funded by the Operational Programme "Competitiveness, Entrepreneurship and Innovation" (NSRF 2014–2020) and co-financed by Greece and the European Union (European Regional Development Fund).

References

[1] P.L. Lagari, L.H. Tsoukalas, I.E. Lagaris, Variance counterbalancing for stochastic large-scale learning, Int. J. Artif. Intell. Tools 29 (5) (2020) 2050010:1–10.
[2] R. Fletcher, Practical Methods of Optimization, second ed., John Wiley & Sons, New York, 1987.
[3] D.G. Triantali, K.E. Parsopoulos, I.E. Lagaris, Single-objective and multi-objective optimization for variance counterbalancing in stochastic learning, Appl. Soft Comput. 142 (2023) 110331.

[4] C.A.C. Coello, G.T. Pulido, M.S. Lechuga, Handling multiple objectives with particle swarm optimization, IEEE Trans. Evol. Comput. 8 (3) (2004) 256–279.
[5] D.S. Broomhead, D. Lowe, Multivariable functional interpolation and adaptive networks, Complex Syst. 2 (3) (1988) 321–355.
[6] K. Deb, S. Agrawal, A. Pratab, A fast and elitist multiobjective genetic algorithm: NSGA-II, in: International Conference on Parallel Problem Solving from Nature, Springer, 2002, pp. 849–858.
[7] D.C. Liu, J. Nocedal, On the limited memory BFGS method for large scale optimization, Math. Programm. 45 (1-3) (1989) 503–528.