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

Description of source code files

The SOMO-VCB software is organized within a single main folder named `SOMO-VCB/`, as shown in the folder tree of Fig. 1. 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.

(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 contains the following two folders needed for running the provided software as well as the file `README.pdf`, which contains a brief description of all relevant files of the source code.

(a) `data/`: This folder contains all input data needed for the algorithms.

(b) `results/`: This is the folder where all output files are stored.

Input files

All the necessary input files shall be placed in the `data/` folder. The input files are VCB-related or optimizer-related and contain solely numerical values for the corresponding parameters. VCB-related files are common for both optimization approaches and comprise the following files:

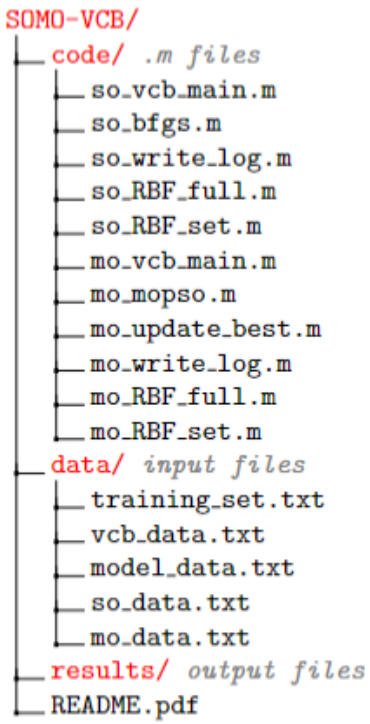

Figure 1: The complete source code of the SOMO-VCB software.

(1) `training_set.txt`: This file contains the complete training set T. Each row contains an n-dimensional training vector x_i followed by its correct output y_i, all values separated by spaces. Thus, a training set of τ training vectors is stored in a file of τ rows. The decimal representation of the vectors (number of decimal digits) is up to the user.

(2) `vcb_data.txt`: It determines the following VCB parameters, one per line, in the specific order:

   (a) The number η of mini-batches for the current set of mini-batches.

   (b) The percentage of patterns used per mini-batch set.

   (c) The maximum number of VCB cycles $c_{\max}$.

   (d) The maximum number of single-pattern network evaluations $t_{\max}$.

   The value of $t_{\max}$ is used as the baseline computation budget of the optimizer. Thus, an experiment (run) is always finished when this number of single-pattern network evaluations is reached. If the user defines a positive value $c_{\max} > 0$ for the number of VCB cycles, then a fixed number of $t_{\max}/c_{\max}$ (rounded to the nearest integer) evaluations are allowed per VCB cycle. On the other hand, if the user sets $c_{\max}$ to a non-positive value, then the optimizer will spend as many evaluations as required until it satisfies its stopping conditions. The total number of VCB cycles in this case cannot be estimated.

(3) `model_data.txt`: This file provides information about the specific neural network model. In our RBF-based implementation, the file includes the following quantities in the specific order:

   (a) The number of neurons L in the hidden layer.

   (b) The lower bound $\mu_{\min}$ of the components of the center vector μ_l.

   (c) The upper bound $\mu_{\max}$ of the components of the center vector μ_l.

   (d) The lower bound $\sigma_{\min}$ of the standard deviations σ_l.

   (e) The upper bound $\sigma_{\max}$ of the standard deviations σ_l.

   (f) The lower bound $\theta_{\min}$ of the neuron weights θ_l.

(g) The upper bound $\theta_{\max}$ of the neuron weights θ_l.

These values are used to initialize the network parameter vectors in the optimizer and to restrict the produced vectors.

(4) `so_data.txt`: This file contains the parameters of the single-objective optimization approach in the specific order:

(a) Desirable number of independent experiments of the algorithm.

(b) Maximum allowed number of iterations $k_{\max}$ of the optimizer. It is used as a termination condition of BFGS if the user does not define a specific number of VCB cycles.

(c) Maximum allowed number of line search iterations $k_{\max}^{[ls]}$ for the satisfaction of Wolfe-Powell conditions.

(d) Lower bound $\lambda_{\min}$ of the penalty term.

(e) Upper bound $\lambda_{\max}$ of the penalty term.

(f) Minimum acceptable relative improvement ε_f of the objective value (used as stopping/restarting condition).

(g) Minimum acceptable gradient-norm tolerance ε_g (used as stopping/restarting condition).

(h) Parameter ρ_1 of the Armijo condition in line search.

(i) Parameter ρ_2 of the curvature condition in line search.

(5) `mo_data.txt`: This file contains the parameters of the multi-objective optimizer (in the specific order):

(a) Number of objective functions. In our case, this number shall be fixed at 2.

(b) Desirable number of independent experiments of the algorithm.

(c) Swarm size M of the MOPSO algorithm.

(d) Size ξ of the repository set R (usually $\xi = M$).

(e) Maximum allowed number of MOPSO iterations $k_{\max}$. It is used as a termination condition if the user does not define a specific number of VCB cycles.

(f) Maximum allowed number of iterations $k_{\max}^{[R]}$ with no change in the repository set (used as restarting condition).

(g) Inertia coefficient ω for the velocity update.

(h) Parameter ϕ_1 for the velocity update.

(i) Parameter ϕ_2 for the velocity update.

(j) Number of intervals δ per dimension for the adaptive-grid partitioning of the objective space.

(k) Percentage of the search-space range for clamping the velocities with $v_{\max}$.

(l) Mutation parameter γ.

(m) Pareto set evaluation strategy. It shall be set to a positive integer $\nu > 0$ to evaluate exactly this number of solutions (or up to the Pareto optimal set size if smaller). The whole Pareto optimal set is evaluated if it is set to a non-positive integer $\nu \leqslant 0$.

No restriction exists on the number of decimal digits used to represent the real-valued parameters.

Output files

Running either the single-objective or the multi-objective version of the software produces three output files stored in the `results/` folder. Depending on the specific optimization 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 it is irrelevant:

(1) `*_log`: This is the log file where the corresponding function writes all input values in the `*_write_log.m` file. The file contains all parameter names (as appear in the software) and their assigned values, properly categorized (model-related, algorithm-related, etc.). Reviewing this file, the user can verify that the desired values have been read from the input files and used in the current run.

(2) `*_report`: This is the main output file. It contains one line per experiment. Each line includes the following information in a specific order:

   (a) The number of the current experiment.

   (b) The value of the full MSE over the whole training set for the best-detected solution of the experiment.

   (c) The number of VCB cycles performed in the experiment.

   (d) The number of total network evaluations performed in the experiment.

   (e) The running time spent for the specific experiment.

   The information is organized in columns in order to facilitate the post-processing of the results.

(3) `*_solution`: This file contains the actual solution vectors of the corresponding experiments. There is one line per independent experiment containing the following information in the specific order:

   (a) The number of the current experiment.

   (b) The complete solution vector.

All output files contain plain text. The user has full access to the output information from the main programs (`*_vcb_main.m` files) and can entirely modify both the written information and the output style. Lastly, it should be noted that the final number of the network (single-pattern) evaluations written in the `*_report` file may slightly exceed the maximum number set by the user. Each evaluation of a candidate solution requires a number of network evaluations over the set of mini-batches. In order to fulfill this requirement, we allow the evaluation to complete before checking for termination. This is why the user-defined number of evaluations may be exceeded, especially in cases where the user has not defined a specific number of VCB cycles, which implies an unspecified number of evaluations per cycle.

The interested user may contact D. Triantali (**d.triantali@uoi.gr**) for further clarifications and discussion.