Premeable: Hyperparameter Optimization of Generic Deep Autoencoders for Time Series using a Genetic Algorithm

Author: Anika Terbuch

History: \change{1.0}{11-Apr-2023}{Original}

Source: Chair of Automation, University of Leoben, Austria

email: automation@unileoben.ac.at *url:* automation.unileoben.ac.at

Problem Description

This framework extends the framework "Generic Deep Autoencoder for Time-Series" by providing an algorithm for hyperparameter optimization based on a genetic algorithm.

The genetic algorithm was developed to overcome the long runtime and expert knowledge that is needed in manual hyperparameter tuning. Each ML algorithm has several settings, i.e. hyperparameters, which need to be set before executing the learning procedure. The relationship between the hyperparameters and the performance of machine learning is still unclear. It is assumed that some hyperparameters have more impact on the performance than others. Optimizing them and finding regions of stable performance can be achieved by using some kind of meta-heuristic. However, the nature of meta-heuristics does not guarantee convergence towards a global optimum.

A meta-heuristic was chosen because hyperparameter optimization is a multidimensional optimization problem and literature suggests (e.g. https://arxiv.org/abs/1412.0233) that finding a global optimum of the network is not feasible; however, for large networks most minima are equivalent.

What are Hyperparameters

Hyperparameters are parameters of a machine learning algorithm, here: of an autoencoder, which need to be set before the training is started and are not inferred from the data during the training procedure. Examples of hyperparameters are the number of neurons in the layers, the learning rate, or the mini-batch size.

One needs to choose in advance based on the complexity of the problem, available computational time, and data which hyperparameters in what range to optimize.

What is a Genetic Algorithm

A genetic algorithm is a meta-heuristic of the class of evolutionary algorithms and is inspired by the principles of evolution and inheritance. The underlying idea is based on the "survival of the fittest". In each iteration, called generation, a set of valid potential solutions is produced. The potential solutions are referred to as individuals. The individuals in a generation are ranked based on their fitness. The value of the fitness of an individual depends on how well their represented solution fits the problem. This is achieved by an underlying fitness function. The fitness function reflects the problem that should be optimized. The fitness value is the quantitative value that guides the search.

The genetic algorithm converges, hopefully, after several generations. If the search for the genetic algorithm was successful, the best individual of the last generation represents a solution near the global optimum. Summed up: the goal is to obtain a better population every iteration. A search algorithm has several possible solutions and the task is to find the best solution in a fixed number of time steps. Evolutionary algorithms, to which class also genetic algorithms belong, evaluate a large number of individuals and simultaneously perform a directed search.

Fitness Function

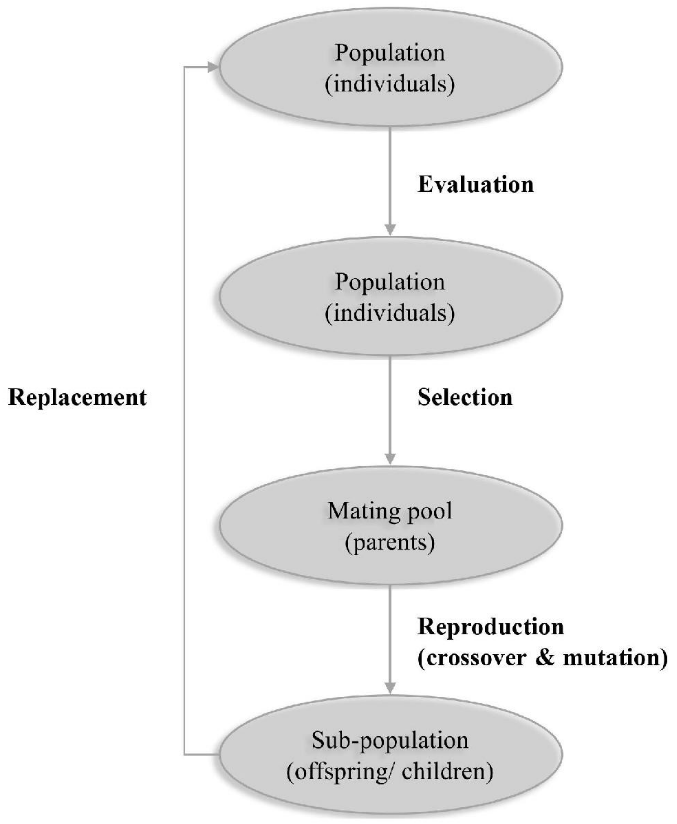
In most of the available literature, the optimization problem is formulated as a maximization problem - a lower fitness is considered to be better. However, in Matlab, it is formulated as a minimization problem. Since each minimization problem can be formulated as a maximization problem and vice versa this is only a matter of definition. In this toolbox, analogous to Matlabs implementation: the genetic algorithm is used to find an individual which minimizes the fitness function (https://de.mathworks.com/help/gads/ga.html).

Description of the Main Genetic Operators

The creation of the first generation differs from all subsequent generations: the first generation is obtained by initializing the desired number of individuals randomly on the specified domain or using some kind of heuristic. If the search landscape of a given problem is not known in the beginning, it is hard to determine if a guess is "good" or "bad" therefore often a random initialization procedure is chosen.

After the first population was created genetic operators are applied to obtain the subsequent generations. Which classically consists of the following steps:

1. Evaluation: The fitness of each of the individuals of the current generation is evaluated.
2. Selection: The individuals who are candidates for reproduction are chosen using a selection rule. These chosen individuals form the so-called mating pool.
3. Reproduction: The offsprings are formed by applying crossover and mutation. Crossover combines the genes of two or more parental individuals into an offspring. The mutation operator leads to changes with a very small probability of the newly created individual. This should enhance the genetic diversity and counteract the genetic algorithm getting stuck in a bad local optimum.
4. Replacement: The new population is formed by replacing the individuals from the previous generation according to a replacement strategy.



More information on the functionality of the autoencoder framework can be found in the folder AutoencoderDeep or on Mathworks: https://de.mathworks.com/matlabcentral/fileexchange/111110-generic-deep-autoencoder-for-time-series?s_tid=prof_contriblnk

A detetailed descritpion of the used algorithm can be found in https://pure.unileoben.ac.at/portal/de/publications/lstm-hyperparameter-optimization(f2191826-94a4-41b5-84d5-5a308d24ba8a).html?customType=theses

Application examples of the AutoencoderDeep framework using this genetic algorithm for hyperparameter optimization were shown in the following papers and thesis

- Detecting Anomalous Multivariate Time-Series via Hybrid Machine Learning https://ieeexplore.ieee.org/document/10015855
- Optimization of Autoencoders for Anomaly Detection in Multivariate Real-TIme Measurment Data Aquired from Production Machinery https://pure.unileoben.ac.at/portal/de/publications/optimization-of-autoencoders-for-anomaly-detection-in-multivariate-realtime-measurement-data-acquired-from-production-machinery(47ac7e62-96e7-4645-83d2-661485fe03ed).html?customType=theses

- Quality Monitoring in Vibro Ground Improvement: A Hybrid Machine Learning Approach https://onlinelibrary.wiley.com/doi/10.1002/geot.202200028
- Hybrid Machine Learning for Anomaly Detection in Industrial Time-Series Measurment Data https://ieeexplore.ieee.org/document/9806663
- Machine Learning and KPI Analysis applied to Time-Series Data in Physical Systems: Comparison and Combination https://pure.unileoben.ac.at/portal/en/publications/machine-learning-and-kpi-analysis-applied-to-timeseries-data-in-physical-systems(e8b7de4d-dd2c-4aa6-a0df-e9f8cca98cd6).html

Collection of resources to get started:

Autoencoder (http://www.cs.toronto.edu/~hinton/absps/pdp8.pdf)

Variational autoencoder (1312.6114.pdf (arxiv.org))

- Video: Variational inference and deep learning: An intuitive introduction: https://www.youtube.com/watch?v=h0UE8FzdE8U

Long short-term memory (http://www.bioinf.jku.at/publications/older/2604.pdf)

- Video: LSTM networks explained: https://www.youtube.com/watch?v=QciIcRxJvsM

Neural networks in general

- Forecasting with artificial neural networks: state of the art https://www.sciencedirect.com/science/article/pii/S0169207097000447
- Free online book on neural networks and deep learning: http://neuralnetworksanddeeplearning.com/index.html
- Interpretable Machine Learning: A guide for making black box models explainable: https://christophm.github.io/interpretable-ml-book/

Time series modelling

- An introductiory study on time series modeling and forecasting: https://arxiv.org/ftp/arxiv/papers/1302/1302.6613.pdf
- Analytical approaches for anomaly detection in time-series data: A review on outlier/ anomaly detection in time series data https://arxiv.org/pdf/2002.04236.pdf
- A review of unsupervised feature learning and deep learning for time series modeling https://reader.elsevier.com/reader/sd/pii/S0167865514000221?token=D88D2A6389641900AC560EAC2BC164DBE664F55251168236C3A528E6F9F65A86F293636D8CB3B4B3486298FD57246826&originRegion=eu-west-1&originCreation=20221216100734

Hyperparameter optimization

- Hyperparameter tuning for machine learning models https://www.jeremyjordan.me/hyperparameter-tuning/

- Hyperparameter optimization of LSTM network models through genetic algorithm https://ieeexplore.ieee.org/document/8900675/citations?tabFilter=papers#citations
- Beyond manual tuning of hyperparameters: https://link.springer.com/article/10.1007/s13218-015-0381-0
- Discussion of the impacts of hyperparameters on LSTM networks: LSTM: A serach space odyssey https://arxiv.org/abs/1503.04069