ML-based Automated Systems Identification:A Demonstration for Complex Chemical Processes

Robin Hofmanna, Thomas Grötznera, Isuru A. Udugamab,c, Christoph Bayera,\*

aDepartment of Process Engineering, TH Nuernberg, Nuernberg, Germany

bDepartment of Chemical and Materials Engineering, The University of Auckland, Auckland, New Zealand

cS&D Consulting Inc., Houston, Texas, USA

\*christoph.bayer@th-nuernberg.de

Abstract

Chemical processes often exhibit a high degree of complexity and highly nonlinear dynamics. Accurate system identification is essential for optimizing operations and control, resulting in high production rates while ensuring product quality. Machine Learning (ML) has emerged as a transformative technology, offering new possibilities for automating system identification in complex chemical processes. This manuscript provides a demonstration of the application of ML-based automated systems identification techniques. It explores the significance between LSTM neural networks and data preparation through clustering and weighting of timeseries. The central objective of this demonstration is to highlight the potential of ML with control of complex chemical processes in mind.

**Keywords**: machine learning, LSTM, process engineering, clustering

* 1. Introduction

Chemical processes often exhibit a high degree of complexity and highly nonlinear dynamics. Accurate system identification is essential for optimizing operations and control. Machine Learning (ML) has emerged as a transformative technology, offering new possibilities for automating system identification. This manuscript provides a demonstration of the application of LSTM neural networks with a particular focus on data preparation through clustering and weighting of data.

* 1. Data Sources & Preparation
     1. Process Models

Two simulated processes that vary in size and complexity are used to study the ramifications on the respective neural network and its training. The first model, a simple water mixing tank created by the authors, has three inputs which act on three measured outputs, hence being a 3x3 system. Two streams enter the tank differing in flow and temperature, while a third stream leaves the tank. All streams can be adjusted with valves to control the outflow rate *V3*, the temperature *T3* and the level *h* of the liquid in the tank. The second model (4x9) is a conceptual representation of the reactor of the Tennessee Eastman problem (Downs & Vogel, 1993, and Andersen et al., 2022), published by Ricker (1993).

* + 1. Data Set

The training timeseries data for the Ricker model are generated on a grid of initial values for each model input (Table 1). Successive input values of an emerging timeseries are generated by a random walk. Any input value changes with a 25 % probability per timestep and any change in value (from the initial value or the value of the prior timestep) is normally distributed with a standard deviation of 1 %-pt. However, a timeseries that exceeds the model’s limits, e.g., pressure > 3,000 kPa, is automatically discarded. All resulting timeseries comprise 100 timesteps of data each. Thus, the timeseries are unique while roughly providing an equal distribution of input values. The dataset for training contains approximately 1,500 individual timeseries.

Table 1: Initial input grid for Ricker’s training data set for setpoints U1- U4 in percent

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Input | | min | step | max |
| U1 | Feed 1 Valve | 40 % | 5 %-pt | 80 % |
| U2 | Feed 2 Valve | 10 % | 5 %-pt | 40 % |
| U3 | Product Valve | 20 % | 5 %-pt | 60 % |
| U4 | Reactor Level Setpoint | 40 % | 5 %-pt | 50 % |

Validation data (1,000 timeseries) are generated with identical parameters, though the inputs evolve from random starting points. The protocol for test data generation (10,000 timeseries) is identical with the training data, but 2.5 %-pt grid steps and a standard deviation of 2.5 %-pt were chosen.

The mixing tank’s data are generated with a 50 % probability for input change and a standard deviation of 5 %-pt for a change in value of the respective input.

* + 1. Clustering & Weighting

The timeseries data are preprocessed by scaling the range of values, then clustering and subsequently weighting of whole timeseries. This shall ensure that the neural network is trained on equally distributed information such that every information represented in the dataset is eventually represented by the neural network.

The temporal dependency of data points renders timeseries more difficult to cluster than individual points. To account for the dependency in time, each series is transformed into a single multidimensional point, which can be clustered. For clustering, three different algorithms are used. k-Means usually serves as a first approach to clustering. DBSCAN (Density-Based Spatial Clustering of Applications with Noise) (Ester et al., 1996) is used to identify outliers within the dataset. SOM (self-organizing map) (Kohonen, 1982) is a generalized AI-driven principle component analysis.

The weighting solver is a gradient-based fixed step algorithm. First, it calculates a mean for each output variable. Then, the current weight for each output variable and each cluster is determined, which is updated based on the difference to the target. This is implemented in such a way that it also works if the number of clusters for each output variable differs, as with DBSCAN. To avoid extremely high or low weights, limits on them are imposed. In case of poorly distributed clusters or an insufficient number of timeseries, the weighting will not equilibrate information density across the full parameter space.

* 1. Neural Networks
     1. Neuron and Network Architecture

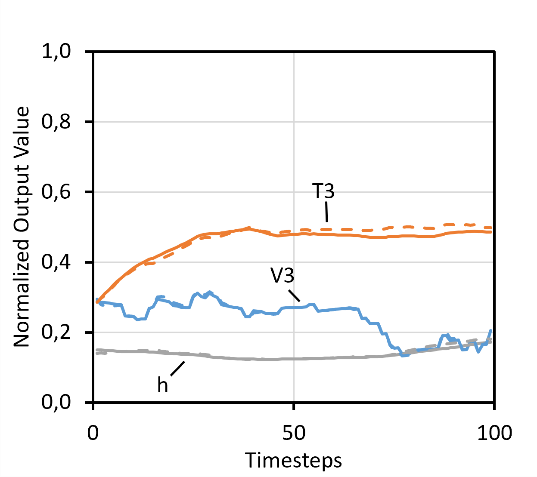
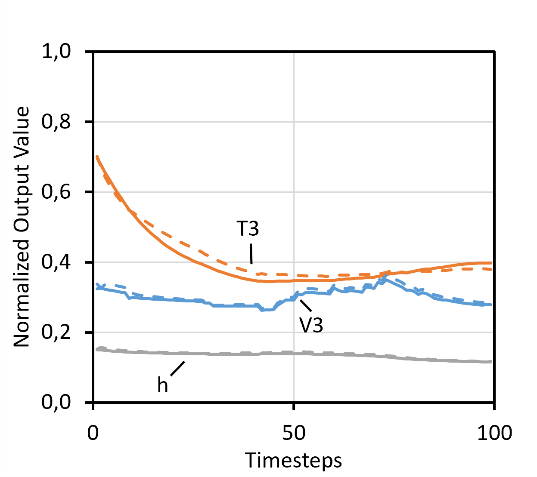
For this study, the neural network and solvers were implemented in C++/CUDA. This approach targets process control implementations on industrial hardware (Udugama et al., 2020).

Two types of neurons are used: (1) Long Short-Term Memory (LSTM), which structurally supports timeseries prediction through recurrency; (2) Feed Forward (ANN) neurons, which increase and shape the mapping complexity inside the network.

All network structures tested were trained in the form of “Input-LSTM-Output”. The input neurons only hold the input information for the network, whereas the output contains sigmoid activations. The LSTM layer is configured as shown by Hochreiter & Schmidhuber (1997). The number of LSTM neurons vary with model complexity. Specifically, mixing tank models are trained with 50 LSTM neurons, while Ricker models are trained with 150 LSTM neurons. Randomized minibatches are used for backpropagation with approximately 1/10 of the whole training set per minibatch.

All data used for training, validation and testing is normalized. Deviating from the usual approach to normalize to a range from 0 to 1, the data are normalized to a range from 0.1 to 0.9, which is due to the sigmoid activation function in the output-layer, which hardly reaches the limits of 0 and 1.

* + 1. Solver Algorithm

The widely used ADAM solver, as published by Kingma and Lei Ba (2015), is implemented and used to update the network parameters in the learning process. Deviating from the original publication, the internal parameter β2 is changed to 0.99 due to estimated higher learning stability. The networks’ learning process is further improved by including learning rate decay as given by Eq. (1).

|  |  |
| --- | --- |
|  | (1) |

denotes the learning rate, *E* the current epoch, *B* the number of minibatches per epoch, and *M* the current number of calculated minibatches in this epoch. *H* is a changeable hyperparameter representing the number of iterations that reduces the learning rate by a factor of 2.

Figure 1: Example of an unweighted mixing tank timeseries; left taken from training set, right from test set; dashed lines represent (true) measurements, whereas solid lines represent network outputs; trained until the loss changed less than 0.002 in 50 epochs.

* + 1. Training and Validation

The gradient required for updating a network’s weights and biases is based on the objective function as given by Eq. (2).

|  |  |
| --- | --- |
|  | (2) |

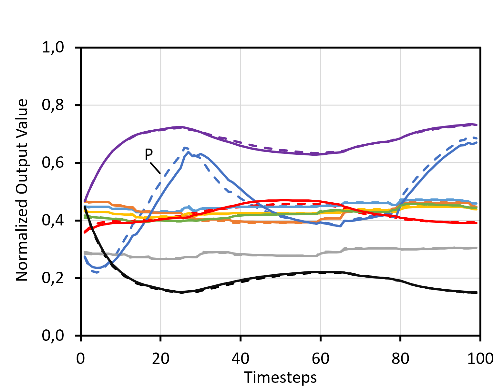
The loss *L* is the sum over *B* minibatches of the squared difference of the true measurements *y* and the output of the network *z*, normalized with the batch size *s*.

The long-range dependencies in time when training on longer timeseries pose difficulties for LSTM neurons. To enhance learning performance, the network is also trained on the previous timestep’s measurement as an additional input (lookback). Since this information is not available when using the network for predictions, the lookback variables are substituted with its own generated predictions.

The number of epochs varies throughout this study depending on the process model’s complexity. More complex models tend to require more epochs for the same level of feature extraction. Given that real plant operations might also have limitations, the model was not trained to perfection, but rather until most information was learned properly. In any case, it was taken care that the training did not stop prematurely. This is achieved by relating the change in both training and validation losses against additional calculation time. The number of training epochs (for all networks) were set so that a network’s loss did improve less than 0.002 over 50 epochs, which equals an improvement of less than 0.005 % compared to the initial loss.

* 1. Model Identification
     1. Mixing Tank Model

As shown in Figure 1 (left), the network successfully extracted the features from the training data as to properly mimic the model’s underlying behavior. The same can be seen on the right-hand side for data it was not trained on. The loss for all networks trained on mixing tank data are quite low and therefore indicate proper learning. However, counterintuitively at first, all networks performed better on data they have never seen. Since these datasets depict less random input changes (relative to training), an increase in network performance with reduction in complexity is an expected result. Across all cluster algorithms, k-Means resulted in the best outcome across all weighted datasets whereas DBSCAN performed worst. Regarding data preparation, weighting did not improve overall performance. Overall, the biggest contribution to the loss (71.5 % on average) is due to the tank temperature *T3*.

Ein Bild, das Diagramm, Reihe, Text enthält.

Automatisch generierte Beschreibung

Figure 2: Example of an unweighted Ricker timeseries; left taken from training set, right from test set; dashed lines represent (true) measurements, whereas solid lines represent network outputs; trained until the loss changed less than 0.002 in 50 epochs.

* + 1. Ricker Model

As Figure 2 illustrates, the training delivers a neural network that mimics the true behavior with satisfactory precision. Most network outputs follow the measurement values closely. Only output 5, which represents the reactor pressure *P*, shows a small but visible deviation.

Network loss values are presented in Table 2. Contrary to the mixing tank, the test loss is always higher than the training loss, which is attributable to greater valve dynamics (cp. Figure 2). Most striking are the significantly higher losses of the DBSCAN-weighted datasets in comparison to the others. Nonetheless, weighting of timeseries to adjust for information density in the parameter space does not seem to improve model identification. Further, analysis of individual output variables and their contribution to the network’s loss shows that reactor pressure *P* is on average responsible for 79.5 % of the total. The worst output variable (output variable with the highest contribution to loss) accounts for at least 72 % of the total loss (Table 2). In other words, all other system information is identified far better than the average loss suggests.

Table 2: Summary of results

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | **Loss** | | |  |
| **Model** | **Set** | **Train** | **Validation** | **Test** | **Worst Output Variable** |
| Mixing Tank | Raw | 0.094 | 0.088 | 0.051 | 69 % |
| Mixing Tank | k-Means | 0.162 | 0.137 | 0.108 | 66 % |
| Mixing Tank | DBSCAN | 0.219 | 0.188 | 0.114 | 72 % |
| Mixing Tank | SOM | 0.141 | 0.127 | 0.078 | 79 % |
| Ricker | Raw | 0.007 | 0.006 | 0.036 | 72 % |
| Ricker | k-Means | 0.010 | 0.008 | 0.049 | 74 % |
| Ricker | DBSCAN | 0.228 | 0.169 | 0.405 | 93 % |
| Ricker | SOM | 0.014 | 0.013 | 0.052 | 79 % |

* 1. Summary and Conclusions

All networks are trained on a noise-free but highly nonlinear set of timeseries, which are, of course, meaningful only for the range presented. Due to the nature of neural networks, the training results can vary significantly and need to be validated diligently. For this study, great care was taken to not cherry pick results that fit the expectations, but to demonstrate both capabilities and limitations with the process industry applications in mind.

Generally, the training through input/output mapping leads to unstructured models that properly mimic the behavior of the underlying system. In turn, domain knowledge, though still important, loses significance. Clustering and weighting of timeseries did not improve the model’s predictive capabilities. Further, this demonstration shows that small changes in data can lead to important differences in forecasting precision and network performance. However, these results are sufficiently accurate for further use in model predictive control, the target application.

Acknowledgements

The authors would like to acknowledge the financial support of the AutoKI project by the STAEDTLER foundation.

References

J.J. Downs, E.F. Vogel, (1993). A plant-wide industrial process control problem. *Computers and Chemical Engineering*, *17*(3). https://doi.org/10.1016/0098-1354(93)80018-I.

M. Ester, H.-P. Kriegel, J. Sander, X. Xu, (1996). A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining*.

S. Hochreiter, J. Schmidhuber, (1997). Long Short-Term Memory. *Neural Computation*, *9*(8). https://doi.org/10.1162/neco.1997.9.8.1735.

D. P. Kingma, J. Lei Ba, (2015). Adam: A method for stochastic optimization*. 3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*.

T. Kohonen, (1982). Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43, 59-69.

N. L. Ricker, (1993). Model predictive control of a continuous, nonlinear, two-phase reactor. *Journal of Process Control*, *3*(2), 109–123.

I. A. Udugama, C. L. Gargalo, Y. Yamashita, M. A. Taube, A. Palazoglu, B. R. Young, K. V. Gernaey, M. Kulahci, C. Bayer, (2020). The role of big data in industrial (bio) chemical process operations. *Industrial Engineering and Chemistry Research*, *59*(34), 15283–297.

E.B. Andersen, I.A. Udugama, K. V. Gernaey, A.R. Khan, C. Bayer, M. Kulahci, (2022). An easy to use GUI for simulating big data using Tennessee Eastman process. *Quality and Reliability Engineering International Journal*, *38*(1), 264-282.