DOI 10.5162/6EuNetAir2016/13

DAV³E – a comprehensive toolbox for multisensor data fusion not only for gas sensors

Manuel Bastuck[1,2], Tobias Baur[1], Tizian Schneider[1,3], Andreas Schütze[1,3]*
[1] Lab for Measurement Technology, Saarland University, Saarbrücken, Germany
[2] Division of Applied Sensor Science, IFM, Linköping University, Linköping, Sweden
[3] ZeMA – Centre for Mechatronics and Automation gGmbH, Saarbrücken, Germany
**m.bastuck@lmt.uni-saarland.de*

Abstract

DAV³E (Data Analysis and Verification/Visualization/Validation Environment) is an object-oriented MATLAB toolbox developed to facilitate the process of building a statistical model out of data generated by cyclically driven sensors or cyclic (industrial) processes. Evaluation of such data usually involves several steps of preprocessing and dimensionality reduction until a good classification or regression model can be built. However, it is rarely known which combination of algorithms and parameters will produce the best result due to its strong dependence on data structure. DAV³E allows data exploration, including visualizations at each step, with an interactive, modular GUI. When a good workflow has been found for the data structure at hand, it can be exported to a command line script for batch processing of big data collections.

Key words: virtual multisensor, big data, automated evaluation, TCO, condition monitoring

Introduction

Temperature cycled operation (TCO) is often used to increase the sensitivity and selectivity of chemical gas sensors [1]. TCO exploits that a gas sensor changes its reaction to a gas drastically depending on its operating temperature. Thus, a sensor array can be simulated with only one physical sensor operated at different temperatures, a concept known as virtual multisensor [2]. The shape of the temperature cycle depends on the type of the sensor; however, temperature steps or ramps are commonly used. This concept has also been expanded to other device parameters, like the gate bias of silicon carbide based field effect transistors (SiC-FET) [3], and to cyclic processes, e.g. the work cycle of a hydraulic machine for on-line condition monitoring to predict machine faults [4].

Workflow

The resulting array can be interpreted as one virtual sensor for each sample in a cycle, e.g. 600 virtual sensors for a cycle of 60 s, recorded at 10 Hz. Most of those sensors give redundant information as parameter cycling and chemical reactions are relatively slow. Both the large number of sensors and information redundancy can cause problems with classification or regression algorithms, rendering them instable or inapplicable at all [5]. Therefore, the dimension of the data must be reduced. In a first step, this is done manually by feature extraction, i.e. computing shape or statistical features of the cycle signal, e.g. the mean or slope of a plateau or ramp, respectively, or a statistical moment of all samples in a certain area of a cycle. Instead of every sample, these values are then used for further processing, so the aim is to have as few of them as possible which contain as much of the initial information as possible. Alternatively, multivariate methods like principal component analysis (PCA, [6]) can be applied directy. However, manual selection of interesting cycle regions by an experienced scientist can often give better results, especially for new, unknown data structures.

Before and after feature extraction, the data can be preprocessed to remove outliers, normalize cycles to reduce drift, or standardize features to make them equally important to variance-based methods like PCA.

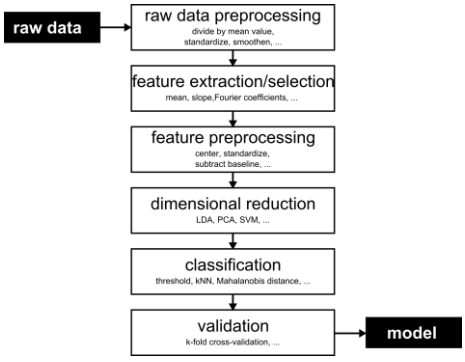

Fig. 1. Steps in the model building process.

Following the feature preprocessing, a classification or regression model is trained. For both kinds of model, the actual target, e.g. a certain gas or fault type, must be known for each cycle. This information is collected in a target vector with as many entries as there are cycles – categorical values for classification or numerical values for regression.

The trained model must be validated to make sure it is not overfitted [5]. Besides another, independent dataset, which is often not available, this can be done with k-fold cross-validation (CV) [7]. Here, the whole dataset is divided into k parts of which k-1 are used to build a model. The remaining part is treated as unknown data predicted by the model, and the results are compared with the correct class known from the target vector. This is repeated k times so that each part has been predicted once. If the model is overfitted, its prediction ability will be poor which manifests in a high false classification rate.

Fig. 1 shows an overview over the complete workflow for building a statistical model from raw sensor data.

Class framework and implementation

DAV³E is designed in a way that all the previously mentioned steps, starting at the raw data and ending with a validated model, can be done in the same environment. Moreover, a wide variety of common and proprietary file formats can be imported. The data from a file is divided into “sensors”, one for each data stream. This enables large flexibility later, as e.g. the time can be treated as a sensor to see if the model is sensitive to signal drift over time. “Clusters” are groups of “sensors” with identical sampling rate and starting time. If this information is available it is automatically considered throughout the process, so that sensors with e.g. different sampling rates can still be fused. “Clusters” are, in turn, part of a “measurement”. The “measurement” defines the environmental parameters at each point in time, which are later used to build target vectors. Moreover, while “clusters” allow adding more sensors to a model (parallel fusion), “measurements” are fused in series, providing additional cycles for a model.

All algorithms working directly on the data, like preprocessing, dimensionality reduction, visualization, and classification, are derived from a common base class. This class surveys distinct folders for new algorithm definitions in the form of a function with a defined interface. In this way, adding new algorithms to the framework is easy even for less experienced users.

A “model” contains a sorted list of all functions to apply to the data and executes them one by one during the training. When a model shall be used in the command-line, all it takes is to call the train method of the “model” object with the desired raw data.

A “project” contains all objects and data and serves as root to easily load and save all configurations. The “sensor” class is designed in a way that raw data does not have to be stored in the project, but can reside in the original files besides the project, which is especially important for big data (on the order of 100 GB and above).

GUI and modules

The GUI follows a modular approach which provides both guidance for the user through the process and easy extendibility. The modules basically replicate the steps described in the Workflow section.

The Import module reads a wide range of file formats and converts them into the internal sensor/cluster/measurement structure. Virtual sensors can be computed with predefined formulas (e.g. the resistance as voltage divided by current).

The Preprocessing module (Fig. 2) contains two different visualizations of the sensor signal. One is the cyclic view, which shows one or more complete cycles in the same graph. The other one is the quasistatic signal, where the same point out of each cycle is drawn over time, giving the impression of a statically operated sensor. While the cyclic view is good to identify differences in cycle shape for, e.g., two different gases, the quasistatic view gives an impression of the sensitivity of the sensor at a distinct point in the cycle. One or more preprocessing steps can be applied in this module, ranging from eliminating outliers to dividing each cycle by its mean value to reduce baseline drift. The result of the preprocessing is shown together with the raw data in the graphs, helping to assess the effectiveness of the preprocessing. It is possible to create many different preprocessing chains and apply them to the sensors individually. In this way, changes to a certain preprocessing chain influence a whole group of sensors.

In the module Cycle Ranges (Fig. 3), the quasistatic signals chosen during the preprocessing are shown, giving an overview of the whole measurement. In lab tests, the usual case is to have time frames of similar environmental conditions, e.g. the same gas for 100 cycles, followed by another gas for the next 100 cycles. To simplify target vector generation, these ranges of cycles can be selected in this

graph and tagged with the current concentrations of all gases appearing during the measurement. Instead of selecting ranges manually, they can also be loaded from compatible files from measurement equipment.

The Grouping module works on the previously selected cycle ranges. It can generate simple target vectors automatically, based on cycle range tags from the previous module.

Alternatively, target vectors can be built by assigning a value (numerical or categorical) to each range. These are the target values for the classifier training for all cycles contained in that specific range. To assist the user and prevent typos, the ranges are colored with different or identical color depending on whether they belong to different or identical groups.

In the Feature Extraction module (Fig. 4), shape-describing features of the cycle are defined and computed. One graph shows an average cycle for each cycle range, helping to identify interesting differences between the

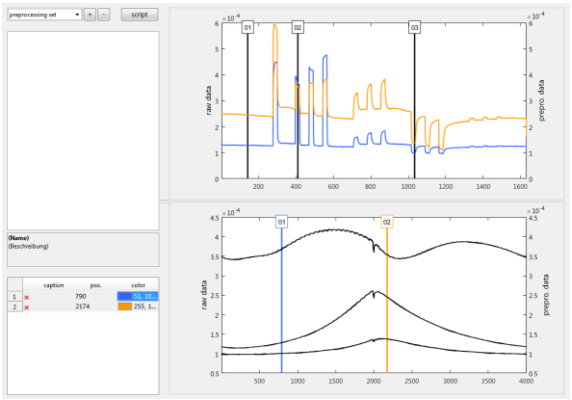

Fig. 2. Preprocessing module. The upper graph shows two quasistatic signals at the two points in a cycle selected in the bottom graph. The bottom graph shows the three cycles selected in the upper graph. The selected points can be dragged with the mouse.

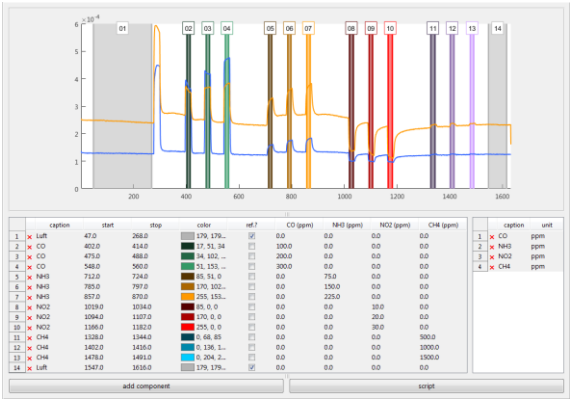

Fig. 3. Cycle Range module. The graphs show the quasistatic signals and the ranges of cycles that will be used for training. Each range can have a color for better identification as well as components, like concentration of a gas or severity of a fault.

groups. As before, ranges can be defined directly in the graph. Each feature, i.e. mean value, slope, or max value, to mention only a few, can have individual ranges because it is rarely useful to compute, e.g., both the slope and the mean of a range. A live preview of the features computed from the average cycles is shown in another graph, helping to estimate the discrimination ability of the chosen features.

In the Model module (Fig. 5), the previously computed features are used as training data for a classification or regression model. Prior to that, they can be preprocessed in different ways, e.g. be centered or standardized. Although standardization is often considered an important step, especially when employing variance-based methods like PCA, there is evidence that standardized features can also produce a worse model then raw features as they over-emphasize noise [8]. This is only one example that there is no standard procedure to deal with this kind of data. The same preprocessing methods can be applied to the target vectors if they are numerical, which can be useful if the model shall be trained on, e.g. double-logarithmic data. Any preprocessing of the target vector is accounted for in the model result, i.e. the result is on a real and not on a preprocessed scale. Different dimensionality reduction algorithms like linear discriminant analysis (LDA, [5]) or PCA, or chains of those algorithms, can then be applied to the features before training a classifier (k nearest neighbors (kNN), Mahalanobis distance, etc.) or regressor (linear regression, partial least squares regression [9], etc.). A validation method like k-fold cross-validation can be added to the training with one click and provides the validation result as root mean square error (RMSE) or misclassification rate for regression or classification, respectively. The results are visualized in many ways, including confusion matrices, scatter plots, histograms or territorial plots (Fig. 5).

Fig. 6 shows an example for a result that has been achieved by applying the described data evaluation techniques with our framework. The raw data comes from a SiC-FET gas sensor driven with a cycle comprising four temperature plateaus within 60 seconds at a sampling rate of 10 Hz. Each cycle was divided by its mean value to eliminate sensor drift, followed by computing the mean value of ten equidistant ranges for each cycle. During the measurement, the sensor was exposed to air and three target gases at three concentrations each, i.e. benzene (1,3,5 ppb), naphthalene (5, 20, 35 ppb) and formaldehyde (50, 100, 150 ppb). The features were standardized,

projected onto two dimensions using LDA, and, eventually, 10-fold cross-validation with a Mahalanobis distance classifier yielded more than 99.5 % correctly classified cycles, demonstrating very good selectivity at very low concentrations using only one physical sensor.

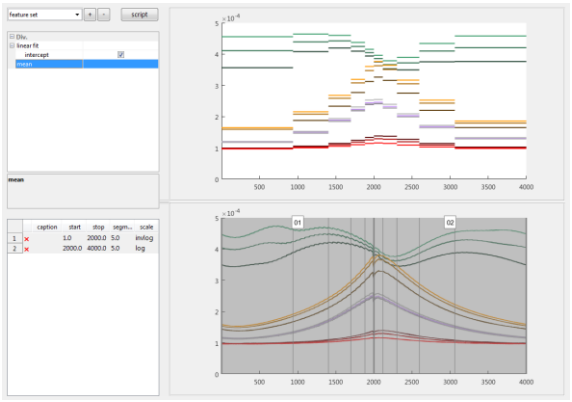

Fig. 4. Feature Extraction module. The bottom graph shows one average cycle for each cycle range. Features are computed for all gray ranges, here with logarithmically de- and increasing width. The upper graph shows the features for the average cycles.

Fig. 5. Model module with territorial plot where the plot area is colored in the color of the class as which a point in this area would be classified. The whole model is defined on the left: a 2D-LDA is followed by a k n classifier and a k-fold cross-validation.

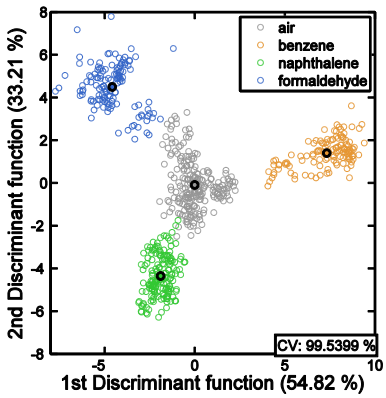

Fig. 6. Example scatter plot demonstrating excellent selectivity. Percentages at the axes give the amount of information contained in the respective dimension.

Conclusion and future work

We have developed a MATLAB framework for easy evaluation of cyclic sensor data. The GUI guides the user through the process and makes it possible to try many different evaluation chains in a quick and easy manner to find the best model for a given data structure. Specialized visualizations throughout the GUI help to find out crucial steps and parameters in the workflow. The resulting model can be used in the command line for batch processing of large data collections. The modular layout of both class framework and GUI makes future expansion easy. We will implement automated parameter search which tries to optimize the validation result as well as on-line evaluation of data with a model.

References

[1] P. Reimann and A. Schütze, “Sensor Arrays, Virtual Multisensors, Data Fusion, and Gas Sensor Data Evaluation”; in: C.-D. Kohl, T. Wagner (eds.): *Gas Sensing Fundamentals*, Springer, 2014, ISBN: 978-3-642-54518-4.

[2] A. Schütze, A. Gramm, and T. Rühl, “Identification of organic solvents by a virtual multisensor system with hierarchical classification,” *IEEE Sens. J.*, vol. 4, no. 6, pp. 857–863, 2004.

[3] C. Bur, M. Bastuck, A. Lloyd Spetz, M. Andersson, and A. Schütze, “Selectivity enhancement of SiC-FET gas sensors by combiningtemperature and gate bias cycled operation using multivariatestatistics,” *Sensors Actuators, B Chem.*, vol. 193, pp. 931–940, 2014.

[4] N. Helwig, E. Pignanelli, and A. Schütze, “Condition monitoring of a complex hydraulic system using multivariate statistics,” in *2015 IEEE International Instrumentation and Measurement Technology Conference (I2MTC) Proceedings*, 2015, pp. 210–215.

[5] R. Gutierrez-Osuna, “Pattern analysis for machine olfaction: A review,” *IEEE Sens. J.*, vol. 2, no. 3, pp. 189–202, Jun. 2002.

[6] H. Abdi and L. J. Williams, “Principal component analysis,” *Wiley Interdiscip. Rev. Comput. Stat.*, vol. 2, no. 4, pp. 433–459, 2010.

[7] M. Browne, “Cross-Validation Methods.,” *J. Math. Psychol.*, vol. 44, no. 1, pp. 108–132, 2000.

[8] M. Bastuck, T. Baur, and A. Schütze, “Fusing Cyclic Sensor Data with Different Cycle Length,” in *MFI 2016 - 2016 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, 2016.

[9] S. Wold, M. Sjöström, and L. Eriksson, “PLS-regression: A basic tool of chemometrics,” *Chemom. Intell. Lab. Syst.*, vol. 58, no. 2, pp. 109–130, 2001.