MODELING AND OPTIMIZATION OF A ROBUST GAS SENSOR

Margarita A. Rebolledo C., Sebastian Krey, Thomas Bartz-Beielstein, Oliver Flasch, Andreas Fischbach, Jörg Stork
SPOTSeven Lab, TH Köln, Gummersbach, Germany
margarita.rebolledo@th-koeln.de, sebastian.krey@th-koeln.de, thomas.bartz-beielstein@th-koeln.de, oliver.flasch@th-koeln.de, Andreas.fischbach@th-koeln.de, joerg.stork@th-koeln.de

Abstract In this paper we present a comparison of different data driven modeling methods. The first instance of a data driven linear Bayesian model is compared with several linear regression models, a Kriging model and a genetic programming (GP) model. The models are build on industrial data for the development of a robust gas sensor. The data contain limited amount of samples and a high variance. The mean square error of the models implemented in a test dataset is used as the comparison strategy. The results indicate that standard linear regression approaches as well as Kriging and GP show good results, whereas the Bayesian approach, despite the fact that it requires additional resources, does not lead to improved results.

Keywords: Bayesian modeling, BMA, Design of experiments, Genetic programming, Kriging, Lasso, Linear regression.

1. Introduction

Theoretically, there are many advantages for the implementation of Bayesian analysis [9]. The use of Bayesian models might represent a good alternative for industrial applications as they produce more informative results. The generation of a data-driven model to optimize the development of a carbon-monoxide sensor provides an opportunity to test these assertions on limited and sparse data. As a first approach, Bayesian robust linear regression is implemented and compared to standard regression methods and a genetic programming approach. Our goal is to learn the difference in performance from the tested methods when applied to this kind of data and to set future considerations for working with Bayesian models in a more demanding fashion.

In recent years the need to reduce air pollution levels has gained more importance in the automotive industry. The efficiency increase of the motor combustion process plays an important role for the reduction of pollution levels. This efficiency can be indirectly measured by monitoring the concentrations of carbon monoxide and other harmful gases released into the atmosphere. This paper focuses on the modeling and optimization of a carbon monoxide in-situ sensor. The sensor should be able to discern the carbon-monoxide concentration apart from the other exhaust gases. This is a difficult goal, because the sensor is exposed to and influenced by the other gases. Thus, the sensor output is not expected to be a direct result of the concentration of the gas of interest. Instead it will be the result of an underneath process influenced by all the other gases. Similar or relatable problems have been addressed in the literature. In these instances the sensor had different manufacture methods, submitted to different gas mixtures or operative conditions. The modelling was addressed using either numerical methods based on the sensor composition [1] or data driven methods like neuronal networks and partial least squares [2, 6].

At the end of our analysis we hope to obtain models from different methods with an improved sensitivity to carbon-monoxide concentrations. The models will be compared in order to check the performances differences and possible improvement opportunities.

This paper is structured as follows: Section 2 describes the research configuration, i.e., data and experimental designs. Key features of the algorithms are introduced in Section 3. Section 4 presents results from the experiments. Finally, a discussion of the results in given in Section 5.

2. Problem

2.1 Data Description

The data was collected following a *response surface design of experiments* (RS-DoE). This design constraints itself to the maximum and minimum expected concentration values of each gas under normal working conditions. Given the cost and time consumption required for the experiments, only a limited amount of samples could be measured. The minimum number of samples required to have a good system description and the real limit of possible realizable samples in the industrial testing station was balanced. Finally, a sample size of 80 was chosen. A summary of the data is shown in Table 1. This application example is anonymized due to confidentiality reasons. The data were standardized, meaning that every sample had its mean subtracted and was then divided by the standard deviation. The different gases are denominated as the variables $X1$ to $X7$. The values of interest correspond to the columns denominated $Y1$ and $Y2$, which are the sensor measurements. All the models will use this dataset as the training set.

Table 1: Overview of the standardized dataset used to generate the models of the sensors. Here every input of the model is denoted by an X and every sensor output is denoted by an Y.

	X1	**X2**	**X3**	**X4**	**X5**	**X6**	**X7**	**Y1**	**Y2**
Minimum	-1.13	-1.21	-1.16	-1.13	-1.15	-1.17	-1.00	-1.94	-2.06
1st Quartal	-1.13	-1.21	-1.16	-1.13	-1.15	-1.17	-0.82	-0.63	-0.58
Median	0.09	0.03	0.12	0.08	0.08	0.05	-0.39	0.06	0.09
Mean	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
3rd Quartal	1.30	1.26	1.40	1.29	1.28	1.28	0.59	0.66	0.67
Maximum	1.30	1.26	1.40	1.29	1.31	1.28	3.79	2.32	2.28

A general idea of the system behavior can be obtained by examining the correlation between the system output and inputs as shown in Table 2. Some assumptions can be made about the influence each variable has on the sensor output: not all parameters seem to have the same influence on the sensor output. Also, the sensors do not behave identically. Figure 1 shows the effect the two most strongly correlated parameters, X1 and X4, respectively, have on the sensors signal.

A second dataset, denominated *test set*, which follow the characteristic of the previously described training set was made available to validate the results of the obtained models.

2.2 Experimental Design Considerations

The data described in Section 2.1 was retrieved during tests based on an experimental design suitable for fitting models using the *response surface methodology* (RSM) [10]. First experimental results taken from screening design experiments indicated that a first order polynomial model is not sufficient, due to cross-sensitivity of the sensors. Therefore it was decided to use a RSM with two-factor interactions quadratic effects. In this case central composite designs would have been a logical

Table 2: Correlation between the system output and inputs for the training dataset

	X1	**X2**	**X3**	**X4**	**X5**	**X6**	**X7**
Y1	0.34	-0.19	-0.27	0.73	0.01	-0.00	-0.21
Y2	0.31	-0.16	-0.18	0.78	0.00	-0.03	-0.22

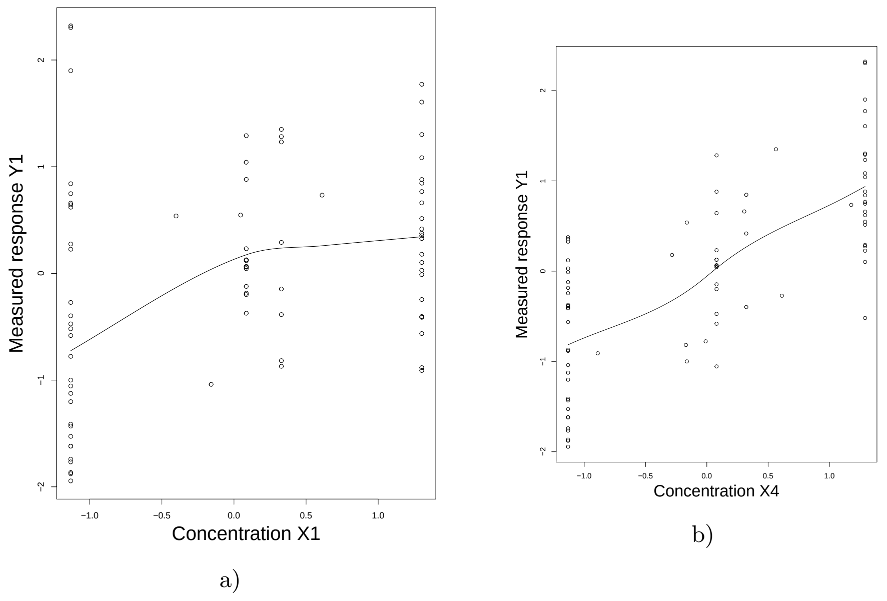

Figure 1: Scatter plots showing the general behavior of Y1 with respect to: a) the influence of X1 and b) the influence of X4.

choice. They are a combination of a box design, typically a full factorial or fractional factorial design and additional star or center points.

A *full factorial design* (FFD) with three levels for each of the six factors to estimate main effects and all quadratic terms would lead to $3^7 = 2187$ experiment runs at minimum. Choosing to divide factors in two levels, a two level FFD would still need at least $2^7 = 128$ runs, without any repetitions or center points. A valid system description would need even more runs.

To reduce the number of runs and be capable of fitting second order polynomial models a Box-Behnken design comes into consideration. But as a linear constraint on the input variables is limiting the sum of their values, almost all standard designs methods does not meet the requirements. Therefore, a more flexible design is needed. Using the statistical software JMP the RS-DoE was generated following the given constraints and applying the I-optimality criterion [7, 13]. The design was run and optimized a total of 80 times with 80 data points in order to obtain the best possible inference accuracy.

3. Algorithms

3.1 OLS

A linear model estimated by ordinary least squares is the natural first modeling attempt for data generated by an experimental design. Our baseline model for the comparison of the different modeling methods is the linear main effects model

$$f^1 : \hat{y} = \beta_0 + \sum_{i=1}^{7} \beta_i x_i.$$

In this work we used a RSM design, so beside the main effects the parameters of all two-way interactions and quadratic terms of the input variables can be estimated. This results in the full linear model

$$f^7 : \hat{y} = \beta_0 + \sum_{i=1}^{7} \beta_i x_i \sum_{i=1}^{7} \sum_{j=i}^{7} \beta_{ij} x_i x_j.$$

Based on the full linear model f^7 we applied variable selection based on an analysis of variance to get a more sparse model, which can be better interpreted. With a F-Test p-value of $\alpha = 0.01$ as the decision boundary for the inclusion into the final model, we obtained the model

$$f^2 : \hat{y} = \beta_0 + \sum_{i=1}^{4} \beta_i x_i + \beta_{14} x_1 x_4 + \beta_{34} x_3 x_4.$$

The *mean squared error* (MSE) as defined in Eq. 4 on page 276, was used for our comparisons. While the full linear model f^7 has a lower training error, i.e., MSE of 0.11 for sensor $Y1$ and 0.10 for sensor $Y2$, compared to the baseline model (MSE of 0.24 for sensor $Y1$ and 0.23 for sensor $Y2$), the prediction performance on the test dataset is very weak (MSE of 7.76 and 9.08). This is a strong indicator of overfitting.

The model f^2 has a MSE of 0.79 for sensor $Y1$ and 0.80 for sensor $Y2$, which is comparable (for sensor $Y1$) and little higher (for sensor $Y2$) than the baseline model. The residual standard error for the training set is lower (0.43 compared to 0.52 for sensor $Y1$ and 0.41 compared to 0.51 for sensor $Y2$) resulting in narrower confidence intervals for the parameters. The adjusted coefficient of determination (adjusted R^2) is 0.81 compared to 0.73 for sensor $Y1$ and 0.83 compared to 0.74 for sensor $Y2$. This means the inclusion of the two two-way interactions X1:X4 and X3:X4 has a large contribution for the explanation of the variance in the dataset, while the input variables X5, X6 and X7 have very little or no contribution and can be left out of the model.

3.2 Lasso

The *Least Absolute Shrinkage and Selection Operator* (Lasso) implements a selection method for linear models [15]. It selects solutions with fewer parameter values, effectively reducing the number of variables upon which the given solution is dependent. The Lasso trains a linear model with a L_1 prior as regularizer.

Give a set of input measurements $X = \{x_i\}_{i=1}^n$ and an outcome measurement y, the lasso fits a linear model

$$\hat{y} = \beta_0 + \sum_{i=1}^{p} \beta_i x_i.$$

Let $\alpha \geq 0$ be a constant. The Lasso uses the following optimization criterion:

$$\min_{\beta} \frac{1}{2n} ||X\beta - y||_2^2 \text{ under the constraint } ||\beta||_1 \leq \alpha, \tag{1}$$

where $||\cdot||_1$ and $||\cdot||_2$ denote the L_1- and L_2-norm, respectively. The positive constant α is a tuning parameter. For large α values, the constraint $||\beta||_1 \leq \alpha$ in Eq. 1 has no effect and the usual linear least squares regression is performed. For smaller values of α, the solutions are shrunken versions of the least squares estimates. Decreasing the values of α forces the coefficients β_i's to become zero, i.e., choosing α results in selecting the number of predictors to use in a regression model. The Lasso can recover the exact set of non-zero weights (under certain conditions). Coordinate descent is used to fit the coefficients.

3.3 Kriging

Kriging or Gaussian process regression is a method of interpolation [14]. The n observations in an arbitrary data set, $Y = \{y_i\}_{i=1}^n$ can be associated as a single point sampled from some multivariate (n-variate) Gaussian distribution. The observations and the Gaussian process are related to each other by the covariance or kernel function $k(x_i, x_j)$. Kernel functions compute the distance between two samples in an arbitrary metric and apply a radial function to this distance. The squared exponential kernel, also known as the Gaussian *radial basis function* (RBF) kernel, is used in our study. This kernel is given by

$$k_1(x_i, x_j) = \sigma^2 \exp(-\theta \|x_i - x_j\|_2^2) \text{ with } \theta = \frac{1}{2l^2}. \tag{2}$$

The RBF kernel can be interpreted as a similarity measure, because values of this kernel decrease with distance. They range between zero (in the limit) and one. The length parameter l in Eq. 2 determines the effect of other observations during interpolation at new x values. The RBF kernel was selected in our study, because Gaussian processes with this kernel generate smooth functions. Since noisy data were analyzed in our study, the white noise kernel $k_2(x_i, x_j) = \sigma^2\delta(x_i, x_j)$, where $\delta(x_i, x_j)$ denotes the Kronecker delta function, was added to the RBF kernel. Hence, we used the kernel function $k = k_1 + k_2$.

3.4 Robust Bayesian Modeling

Bayesian modeling is the mathematical relocation of credibility of parameters values for a model according to what can be inferred from the data. From previous knowledge of the combustion process it is expected that not only the main predictors but also the interactions between predictors have an effect on the sensor reading. As a general rule, if the data contains K variables then the expected number of possible models will be 2^K. The total number of variables in the dataset with all the interactions included accounted to 22, that is 4.19×10^3 possible model combinations to describe the sensor reading. To reduce the dimensionality of the problem *Bayesian model averaging* (BMA) is implemented. This provides a way to account for the uncertainty in model selection and provide in average a better predictive ability [5]. BMA is implemented in the statistical programing language R using the Bayesian Model Sampling (BMS) package [16]. The results show that the predictors X1, X3, X4, X1:X4, X3:X4 and X2 seem to be the most important for a good model.

As the first taken approach the reduced model containing only 6 out of the 22 variables is defined using a linear relationship. The model was implemented using *Just Another Gibbs Sampler* (JAGS), which is a program for Bayesian modeling using *Markov Chain Monte Carlo* (MCMC) [11] and rjags [12] as a link between R and JAGS.

The sensor responses $Y1$ and $Y2$ are modeled following a non standardized Student's t-distribution. This distribution was selected assuming that the variance present in the sensor output, illustrated in Fig. 1, served as an indicator of variance in the model response. The mean of the distribution is defined by the canonical linear formula of the linear regression. The spread of the data was set to have a wide range of probable values defined by an uniform distribution. The normality factor, expressed as an exponential distribution, have preference for values close to one. Given the limited prior information available for the experiment, weakly informative priors are assigned to the parameters. The coefficients priors, β_i, are defined to have a normal distribution centered around zero and a large variance. The t-distribution normality factor υ favors values smaller than 30 and σ allows for a wide enough distribution. The prior distributions were chosen as follows (Eq. 3):

$$\alpha \sim N(0,4), \quad \beta_i \sim N(0,4), \quad \upsilon \sim Exp(30), \quad \sigma \sim U(-1^{-4}, 10) \tag{3}$$

The MCMC simulation are executed on the defined model to sample the posterior distribution of the parameters of interest, α, β_i, σ, and υ. The chains were specified to run 500 adaptive iterations, followed by 1,500 burn-in iterations. Afterwards, 15,000 samples were taken from the posterior distribution with a thinning factor of 20 steps. Investigating their trace plots and diagnostic statistics of the resulting MCMC object reveals that the chains have converged. A value for the Gelman-Rubin diagnostic statistic [4] of under 1.1 suggest a good convergence. The effective sampling size (ESS) backups this assumption. The visual and numeric diagnostics allows us to think that the resulting MCMC sampling is representative and accurate of the posterior distribution of the different parameters. The posterior distributions obtained from the MCMC sampling for each parameter coefficient can be seen in Table 3 together with the *high density intervals* (HDI) of 95%. The MSE for the fitted models is 0.16 and 0.15 for the sensor $Y1$ and $Y2$, respectively.

Table 3: Posterior mean for the coefficients β_i for $i = 1, ..7$ and the parameters σ and υ for the models of Y1 and Y2. The lower HDI (L-HDI) and upper HDI (U-HDI) limits are indicated for each entry.

				Y1					
	B0	**B1**	**B2**	**B3**	**B4**	**B5**	**B6**	σ	υ
Mean	-0.01	0.32	-0.14	-0.28	0.72	-0.23	-0.14	0.41	37.67
L-HDI	-0.09	0.23	-0.24	-0.38	0.63	-0.33	-0.42	0.34	5.13
U-HDI	0.09	0.42	-0.04	-0.18	0.82	-0.14	-0.05	0.49	117 .66
				Y2					
	B0	**B1**	**B2**	**B3**	**B4**	**B5**	**B6**	σ	υ
Mean	-0.00	0.29	-0.10	-0.20	0.78	-0.26	-0.13	0.39	34.2
L-HDI	-0.09	0.20	-0.19	-0.29	0.69	-0.35	-0.22	0.31	4.44
U-HDI	0.09	0.39	-0.10	-0.11	0.88	-0.17	-0.04	0.47	113.50

3.5 Genetic Programming

Genetic programming is an evolutionary algorithm that searches the set of symbolic expressions defined by a set of basis expressions (building blocks) for expressions that minimize one or multiple loss (fitness) functions. Symbolic regression is the application of genetic programming to regression. In this work, the *Generational Multi-Objective Genetic Programming* (GMOGP) symbolic regression algorithm is used. See [3] for a detailed description of this algorithm.

In this specific case, the set of building blocks $B := \mathbb{R} \cup V \cup F$ consists of the set of real-valued constants $\mathbb{R}$, the set of independent variables $V := \{x_1, x_2, \ldots, x_7\}$ and the set of real-valued functions $F := \{+, -, \times, \div, \sqrt{}, \log, \exp, \sin, \cos\}$.

The GMOGP algorithm performs the following four steps:

1) The algorithm proceeds by initializing a population of $\mu := 100$ random symbolic expressions with maximum node count of 128 based on the building blocks in B.

2) Next, $\lambda := 50$ expressions are created by random recombination and mutation of randomly selected population expressions, together with $\nu := 2$ randomly initialized expressions.

3) The complexity (expression visitation length) and goodness-of-fit (scaled-mean-squared error on training data) for each expression is calculated. The expressions are then sorted into successive Pareto fronts according to both criteria. Solutions on the same front are sorted by crowding distance.

4) If a predefined termination criterion is met (in this case an iteration limit of $300,000$) the algorithm terminates and returns the first Pareto front. Otherwise, the the best $\mu := 100$ expressions are kept and the algorithms enters the next iteration (generation) at step 2).

As symbolic regression is a randomized algorithm, the implementation uses multiple parallel runs to reduce result variance. The first Pareto front based on all parallel runs is returned as the final result. In this case, five parallel runs of $300,000$ generations each were used, each requiring two minutes of single-core compute time on a 1.8 GHz Intel Core i7 processor, or ten minutes compute time in total. The used GP implementation selects model constants by sampling from a uniform random distribution and optimizes existing constants via mutation by adding samples from a normal random distribution. Crossover operations may lead to the duplication of constants, as seen in GP model 2 in Table 4. To facilitate model comparisons, only the model with best goodness-of-fit on training data is selected from the result Pareto front and reported as the final result. Note, these results numbers are based on the commercial parallel GMOGP-FCA implementation *sourcewerk RSR*.

Table 4: Models used in this study. Model formulas and the corresponding MSE values are shown. The first model, f^1, is included as a baseline. It implements a first order linear model with all effects, but no interactions or higher order terms. Best values are shown in boldface.

j	Name	Model sensor Y1	MSE_1
1	LM (base)	$f_1^1 = 0 + 0.35x_1 - 0.17x_2 - 0.26x_3 + 0.74x_4$ $+0.01x_5 - 0.03x_6 + 0.03x_7$	0.76
2	OLS	$f_1^2 = -0.01 + 0.33x_1 - 0.14x_2 - 0.29x_3$ $+0.73x_4 - 0.23x_1x_4 - 0.15x_3x_4$	0.79
3	Lasso	$f_1^3 = 0.25x_1 - 0.05x_2 - 0.17x_3 + 0.63x_4$	**0.56**
4	Kriging	$f_1^4 : \theta =$ $(0.57, 2.59, 6.08, 3.45, 1.53, 18.53, 18.46)$	0.57
5	Bayes	$f_1^5 = dt(0.32x_1 - 0.14x_2 - 0.28x_3 + 0.72x_4$ $-0.23x_1x_4 - 0.14x_3x_4, 0.41, 37)$	0.79
6	GP	$f_1^6 = -0.02 + 0.31x_1 + (0.64 - 0.12x_1)x_4$	0.58
j	Name	Model sensor Y2	MSE_2
1	LM (base)	$f_2^1 = 0 + 0.31x_1 - 0.07x_2 - 0.19x_3 + 0.78x_4$ $+0.01x_5 - 0.04x_6 + 0.04x_7$	0.67
2	OLS	$f_2^2 = -0.01 + 0.30x_1 - 0.10x_2 - 0.20x_3$ $+0.78x_4 - 0.26x_1x_4 - 0.14x_3x_4$	0.80
3	Lasso	$f_2^3 = 0.22x_1 - 0.08x_3 + 0.68x_4 - 0.02x_7$	0.49
4	Kriging	$f_2^4 : \theta=$ $(0.43, 2.41, 17.53, 4.21, 1.30, 19.96, 19.15)$	0.49
5	Bayes	$f_2^5 = dt(0.29x_1 - 0.1x_2 - 0.2x_3 + 0.78x_4$ $-0,26x_1x_4 - 0.13x_3x_4, 0.39, 34.2)$	0.79
6	GP	$f_2^6 = 0.52 - 0.21x_3 + \frac{0.21(-1.43+x_4)}{1.43+x_1} + 0.42x_4$ $-0.21\cos(x_3 - x_4) + 0.21\cos(x_7)$	**0.27**

4. Results

The comparison is based on the MSE. We consider six different models ($j = 1, \ldots, 6$) and two different data sets ($k = 1, 2$) from Table 4.

Let f_k^j denote the function, which models the relationship between y_{train} and x_{train} ($j = 1, \ldots, 4$, $k = 1, 2$). Let $\hat{y} = f_k^j(x_{\text{test}})$ denote a vector of n predictions on the test data x_{test}, and y_{test} denote the vector of observed (true) values, then the MSE can be estimated by

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}^{(i)} - y_{test}^{(i)})^2 \text{ with } i = 1, \ldots, n. \quad (4)$$

5. Conclusion

Interestingly, the lightweight and simplistic Lasso approach and the heavy weighted, sophisticated genetic programming approach obtained the best results. Lasso performed best on $Y1$, whereas genetic programming was able to find the best MSE on $Y2$. However, given the complexity of the genetic programming model, it is hard to see the real effect each variable has on the sensor output. Lasso on the other side, gives a clear and simple overview of the variables effects while keeping the prediction error low. The Kriging model demonstrated a relatively good performance but, the interpretation of the results is less intuitive compared to the Lasso results.

In this particular application the models need to be adaptable to allow its use on other similar sensors. A model of high complexity together with a difficult interpretability constitutes an extra effort. Bayesian and linear regression approaches generated similar formulas. This was expected as the definition of the Bayesian model was done following the canonical linear formula. Both basic linear models (LM and OLS) and Bayesian model present easily interpretable results but with poor MSE values. This implies that, although new information is available when implemented the Bayesian modeling method, there is no significant difference when compared to the standard linear regression. We expect however to obtain further improvements of the model prediction accuracy by means of more complex and specific model definition for the Bayesian case.

In addition to the six algorithms discussed in this study, further algorithms were tested. For example, a standard random forest algorithm was able to obtain results that are comparable to the Kriging results (MSE's: 0.63 and 0.41 on $Y1$ and $Y2$, respectively). However, due to space limitations, these extended results will be analyzed in a forthcoming publication. Results, presented in this study, are limited to models that are of great practical relevance, i.e., can be immediately interpreted by and discussed with technicians and engineers.

At the end and for this specific application, the Lasso model was our preferred method. A simple and clear formula provides valuable starting points for the discussion with the engineers in charge of the project and makes for a straightforward implementation.

Acknowledgement: This report project was promoted by the Federal Ministry of Economy and Energy under the project funding reference number KF3145101WM3 and KF3145103WM4 . Responsibility for the content of this publication lies with the author.

References

[1] A. Bejaoui, J. Guerin, and K. Aguir. Modeling of a p-type resistive gas sensor in the presence of a reducing gas. *Sensors and Actuators B: Chemical*, 181:340–347, 2013.

[2] T. Eklöv, P. Martensson, and I. Lundström. Selection of variables for interpreting multivariate gas sensor data. *Analitycal Chimica Acta*, 381(2-3):221–232, 1999.

[3] O. Flasch. A Modular Genetic Programming System. Dissertation, TU Dortmund, 2015.

[4] A. Gelman and D. B. Rubin. Inference fron iterative simulation using multiple sequences. *Statistical Science*, 7(4):457–511, 1992.

[5] J. A. Hoeting, D. Madigan, A.E. Raftery, and C.T. Volisnky. Bayesian model averaging: A tutorial. *Statistical Science*, 14(4):382–417, 1999.

[6] A. Jafarian, D. Baleanu, H. Darwish, M. Senel, and S. Okur. Applications of Artificial Neural Network Technique to Polypyrrole Gas Sensor Data for Environmental Analysis. *Computational and Theoretical Nanoscience*, 12:1–7, 2015.

[7] JMP. *Design of Experiments Guide*. 2009.

[8] J. K. Kruschke. *Doing Bayesian Data Analysis: a tutorial with R, JAGS and Stan*. Academic Press, 2nd edition, 2014.

[9] J. K. Kruschke, H. Aguinis, and H. Joo. The time has come: Bayesian methods for data analysis in the organizational sciences. *Organizational Research Methods*, 15(4):722–752, 2012.

[10] D. C. Montgomery. *Design and Analysis of Experiments*. Wiley, 5th ed., 2001.

[11] M. Plummer. *Jags: A program for analysis of Bayesian graphical models using gibbs sampling*, 2003.

[12] M. Plummer. *rjags: Bayesian Graphical Models using MCMC*, 2015. R package version 4-4.

[13] F. Pukelsheim. *Optimal Design of Experiments*. Wiley, 1993.

[14] J. Sacks, W. J. Welch, T. J. Mitchell, H. P. Wynn. Design and analysis of computer experiments. *Statistical Science*, 4(4):409–435, 1989.

[15] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society Series B (Methodolgical*, 58(1):267–288, 1996.

[16] S. Zeugner. *Bayesian Model Averaging with BMS*, 2011. R package version 0.3.4.