LUND UNIVERSITY

Pharmacometric covariate modeling using symbolic regression networks

Wahlquist, Ylva; Soltesz, Kristian; Morin, Martin

Published in:
6th IEEE Conference on Control Technology and Applications

DOI:
10.1109/CCTA49430.2022.9966112

2022

Link to publication

Citation for published version (APA):
Wahlquist, Y., Soltesz, K., & Morin, M. (2022). Pharmacometric covariate modeling using symbolic regression networks. In *6th IEEE Conference on Control Technology and Applications* (pp. 1099-1104). IEEE - Institute of Electrical and Electronics Engineers Inc.. https://doi.org/10.1109/CCTA49430.2022.9966112

Total number of authors:
3

General rights
Unless other specific re-use rights are stated the following general rights apply:
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.
• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal

Read more about Creative commons licenses: https://creativecommons.org/licenses/

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

LUND UNIVERSITY

PO Box 117
221 00 Lund
+46 46-222 00 00

Pharmacometric covariate modeling using symbolic regression networks

Ylva Wahlquist[1], Martin Morin[1], Kristian Soltesz[1]

***Abstract*— A central challenge within pharmacometrics is to establish a relation between pharmacological model parameters, such as compartment volumes and diffusion rate constants, and known population covariates, such as age and body mass. There is rich literature dedicated to the learning of functional mappings from the covariates to the model parameters, once a search class of functions has been determined. However, the state-of-the-art selection of the search class itself is *ad hoc*. We demonstrate how neural network-based symbolic regression can be used to simultaneously find the function form and its parameters. The method is put in relation to the literature on symbolic regression and equation learning. A conceptual demonstration is provided through examples, as is a road map to full-scale employment to pharmacological data sets, relevant to closed-loop anesthesia.**

***Index Terms* – Health and medicine; neural networks; Modeling**

I. Introduction

A. Pharmacokinetics

A common use-case of pharmacological models is to describe the effect of a drug on the individual undergoing treatment. The model is often partitioned into a pharmacokinetic (PK) part that describes uptake, distribution, and elimination of drug within the body, and a pharmacodynamic (PD) part, that relates drug concentration to clinical effect.

Within the context of intravenous anesthesia, the PK model can be used to describe the dynamics between drug administration and blood plasma concentration, while the PD model relates the anesthetic effect to the blood plasma concentration. In this work, we will refine attention to the PK sub-model, although the proposed methodology is also relevant for PD and combined PK-PD modeling.

Pharmacological models have several applications (not limited to anesthesia). Examples include offline drug administration profile optimization, also known as target-controlled infusion (TCI) [1], control law designs for online dosing strategies relying on sensor feedback, also known as closed-loop anesthesia [2], as well as traditional offline *a priori* dosing computations.

Often, and indeed within the context of anesthetic drug delivery, the structure of employed PK models is well-established and usually physiologically motivated. For commonly considered anesthetic drugs, the PK can be adequately modeled using a compartmental linear time-invariant (LTI) system. Particularly, for the intravenously administered anesthetic propofol, a mammillary three-compartment model captures the PK dynamics well, as discussed in [3]. Fig. 1 provides a schematic illustration of the propofol PK model and its parameters.

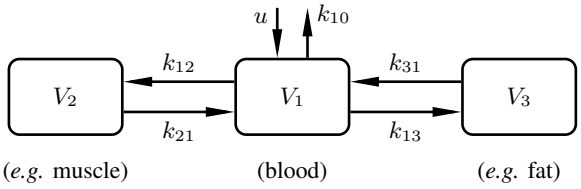

Fig. 1. Three-compartment mammillary compartment PK model for propofol. The drug is intravenously administered to the central compartment, of volume V_1, modeling the blood plasma of the patient, at a rate u [mass/time]. Diffusion processes governed by rate constants k_{ij} [1/time] distribute the drug between the blood plasma and the two peripheral compartments, of volumes V_2 and V_3, modeling the remainder of the body. The rate constant k_{10} models elimination of drug from the blood plasma.

B. Pharmacometric covariate modeling

While the model structure (three-compartment mammillary LTI model) can be assumed known, clinical data indicate important inter-individual variability in compartment volumes and rate constants. Whether the PK model is used for offline dose planning, TCI, or closed-loop controller tuning, the achievable performance is limited by this variability, as we have previously investigated in *e.g.* [4]–[6].

The pre-dominating way to reduce the model uncertainty originating from the inter-patient variability is through pharmacometric covariate modeling. This is done by investigating how the individual PK model parameters (volumes and rate constants, *cf.* Fig. 1) depend on covariates that can be determined for the individual patient.

The population covariates typically used in the context of anesthesia are demographic parameters such as patient age, sex, body mass, *etc*. Although not yet popular within anesthetic pharmacometrics, the use of omics data and biomarkers is increasing the search space for potential population covariates.

There are three key challenges to pharmacometric covariate modeling:

1) Selection of covariates;
2) Selection of a parametric functional relation between the covariates and the PK model parameters;
3) Optimizing parameters of said functional relations to minimize the inter-patient variability within the model, which is not explained by the covariates.

*This work was partially funded by the Swedish government through the Swedish Research Council (grant 2017-04989), and partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation. The authors are members of the Excellence Center at Linköping-Lund in Information Technology (ELLIIT).

[1]Wahlquist, Morin and Soltesz are with the Department of Automatic Control, Lund University, Lund, Sweden.

TABLE I

PARAMETER VALUES OF THE COVARIATE MODELS FOR THE PROPOFOL PK COMPARTMENT VOLUMES V_1 AND V_2 OF (1) PUBLISHED IN [8]. UNITS ARE IMPLICITLY DEFINED THROUGH (1).

Parameter	c_1	c_2	c_3	c_4	c_5
Value	9.29	33.55	0.364	-0.0156	0.547

The first and third challenge have received notable research attention, as exemplified by [7] and [8], respectively. However, surprisingly few works tackle the second challenge in-depth. Instead, it is customary to assert one, or a few—typically vaguely motivated—structures for the functions relating the individual covariates to the parameters of the PK model. The third challenge is then normally approached using optimization within a nonlinear mixed-effect model framework. The goal of this optimization is to find parameters of a pre-determined covariate model structure that (with some assumptions) maximize the likelihood of a clinical data set to be explained by the model. Since the 1980s, the software NONMEM [9] has constituted the gold standard for carrying out the numeric optimization, but there is now at least one modern contender in Pumas AI [10]. However, both softwares rely on the second challenge (establishment of covariate model structure) having been externally addressed.

In the context of propofol, and many other drugs—not only anesthetics, the considered data set comprises of two parts: time-aligned blood plasma drug concentration samples and corresponding recorded (intravenous) drug administration profiles; values of the considered demographic covariates for each individual in the set.

For propofol, there exist a multitude of published pharmacometric covariate models aiming to minimize PK uncertainty stemming from inter-individual variability observable within clinical data, *e.g.* [11]–[14]. Recently, there has been an attempt [8] to establish a combined covariate model that is valid across several of these data sets.

C. Covariate model selection

Traditionally, pharmacometric covariate models comprise of compact closed-form functions. For example, in [8] the volumes of the central compartment V_1 and the peripheral compartment V_2 (in the unit of liters) are modeled as

$$V_1 = \frac{c_1 \mathrm{WGT}}{\mathrm{WGT} + c_2}, \tag{1a}$$

$$V_2 = c_3 \mathrm{WGT} \exp(c_4 \mathrm{AGE} + c_5), \tag{1b}$$

where AGE (age in years) and WGT (body mass in kilograms) are the covariates, while the covariate model parameters, $c_1, \ldots, c_5$ determined through NONMEM optimization in [8], are given in Table I.

It is not motivated in [8] as of why the functions that relate WGT and AGE to V_1 and V_2 should have the structures of (1a) and (1b), respectively. Corresponding strong motivations are also lacking in [11]–[14], and indeed quite generally.

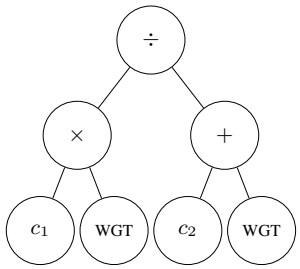

Fig. 2. Graphical representation of the expression tree equivalent to the function from WGT to V_1 defined by (1a).

Instead, it is typically common to optimize over just a few frequently re-occurring but seemingly *ad hoc* structures, often involving linear or exponential relations as those in (1).

While it would be technically viable to for example express the covariate model functions in terms of a standard artificial neural network (ANN) and learn their parameters (weights and biases) from data, such representation becomes opaque and is not likely to be adopted by clinicians.

However, confining to simple explicit functions to model the relation between covariates and PK parameters results in a poorly (exponentially) scaling combinatoric problem, as we will shortly see. For a covariate modeling scenario of realistic complexity, exhaustive search through parameter optimization of each possible model structure becomes computationally intractable, even by modern cloud computing standards.

D. Symbolic regression

The desire to learn "human-readable" closed-form functional expressions as those in (1) from data has emerged within several scientific disciplines. The family of methods dedicated to this end is known as symbolic regression methods.

Symbolic regression was an early research topic within artificial intelligence (AI), with a few noteworthy examples provided in the review [15]. For a long time, the popularity of the paradigm has been overshadowed by rapidly emerging and highly successful deep learning and statistical methods. However, lately, symbolic regression has gained increased research attention, not least through the AI Feynman project [16], which demonstrated a remarkable ability to learn the structure of physical laws from data.

The symbolic regression problem comprises synthesizing an expression tree to represent a function. An example is shown in Fig. 2, representing the function (1a) that maps WGT to V_1 according to the structure suggested in [8].

Whereas an ordinary regression problem would confine to determining the parameters c_1 and c_2 of Fig. 2 from data, the symbolic regression problem additionally entails determining the structure of the expression tree. This is done by generating the tree from a set of pre-determined base expressions, such as addition, multiplication, and division in the case of Fig. 2. For a tree with a fixed number of nodes, there is a discrete choice associated with the selection of which base expressions go where in the tree. Consequently, the

combinatorial problem scales poorly (exponentially) in both the number of considered base expressions and covariates.

E. Equation learning

Another way to perform symbolic regression is through a family of methods that collectively go under the name "equation learning" [17]. Instead of applying perturbations to a nominal expression tree, the equation learning approach is to start with a rich (densely connected and potentially large) nominal expression tree, and then prune it down to obtain one that has a sound balance between expressional complexity and fit to training data.

In the equation learning context, the expression tree is modeled as an artificial neural network (ANN). Instead of customary activation functions (such as sigmoids or the relu), the activation functions are chosen to model the considered base expressions. The covariates (WGT in the case of (1a)) are the inputs to the ANN, the parameters (c_1 and c_2 in the case of (1a)) are modeled through weights and biases of the ANN, and the considered PK model parameter (V_1 in the case of (1a)) constitutes the output of the ANN. The parameters can then be learned from data by training the ANN using (stochastic) gradient descent and classic backpropagation.

To arrive at a tree representing an expression of admissible complexity, the tree must be pruned to the corresponding complexity. This throws us back at the combinatorial problem of selecting which nodes to keep, and which to discard. Instead of solving this problem exactly, equation learning systems typically employ $\mathcal{L}_1$-regularization [18] followed by truncation, or $\mathcal{L}_0$ regularization, where weights are encouraged to be exactly zero [19]. The regularization can be readily achieved by adding a corresponding term to the loss that training of the ANN aims to minimize.

F. Contributions

The main contribution of this work lies in the demonstration of how symbolic regression can be used to obtain structurally optimized, and human-readable, PK covariate models.

We build on the idea of equation learning and device a feed-forward neural network that is capable of learning closed-form expression covariate models. In Section II-B we introduce a prototypical example, that we use to demonstrate how expressions for the covariate model (1) can be learned.

Particular attention is given to ensure (numeric) robustness. To avoid division by (close to) zero denominators, we introduce log-barrier functions into the context of equation learning. We also conduct explicit checks to ensure that the expression tree remains connected throughout training and pruning. These aspects are further detailed in Section II-C.

II. Method

A. Expression tree structure

The expression tree is implemented as a feed-forward ANN with one input layer, one output layer, and d densely connected hidden layers in-between. A schematic illustration is provided in Fig. 3, where each of the $d = 2$ hidden layers is marked by a gray box.

The network inputs are the covariates (AGE and WGT in our example), while the outputs are the considered PK model parameters (V_1 or V_2 in our example).

As in a conventional machine learning ANN, each hidden layer k generates a dense affine mapping of its inputs, that are organized into the vector $\boldsymbol{x}_k$:

$$\boldsymbol{y}_k = W_k \boldsymbol{x}_k + \boldsymbol{b}_k, \tag{2}$$

where W_k and $\boldsymbol{b}_k$ are a real weight matrix and bias vector, respectively.

As is also the case in conventional machine learning ANN, the output of each of these dense layer nodes would propagate to an $\mathbb{R} \to \mathbb{R}$ activation function—typically a sigmoid, relu function, or similar. However, here the activation functions are instead defined by the considered base expressions. This means that

- each activation function could take one or more arguments. For example, in Fig. 3 the exponential function "exp" takes one argument, while the multiplication "$\times$" takes two;
- the type and number of considered base expressions define the output dimension of each hidden layer;
- as in an ordinary ANN, the activation functions lack trainable parameters;
- as in an ordinary ANN, each activation function produces a scalar output, that is passed to the next layer.

The real trainable parameters of the network are thus the union $\{W_1, \ldots, W_d\} \cup \{\boldsymbol{b}_1, \ldots, \boldsymbol{b}_d\}$.

B. Case study

In this paper, we focus on the equation learning problem. Particularly, we learn expressions for V_1 and V_2 from a data set obtained by generating "true" V_1 and V_2 values for each of the 727 data tuples (AGE,WGT) defined by the individuals of the [8] data set.

In a real use-case, the covariate functions would instead need to be learned directly from time-aligned blood samples and drug administration data. In Section IV, we discuss further how our simplified setting can be readily expanded into a practically relevant tool for pharmacometric modeling.

In our examples to follow, we have chosen to penalize the difference between the "true" volumes $\boldsymbol{y}$ generated by (1) across the data set, and corresponding values $\hat{\boldsymbol{y}}$ generated by the network being trained. For both V_1 and V_2, we set out with the nominal expression tree shown in Fig. 3. It has $d = 2$ hidden layers and was chosen sufficiently complex to be able to recover the exact expressions for both (1a) and (1b).

C. Training

1) Loss function: The training loss

$$J = \|\boldsymbol{y} - \hat{\boldsymbol{y}}\|_2^2 + \lambda \sum_{k=1}^{d} \|W_k\|_1 + \mu D \tag{3}$$

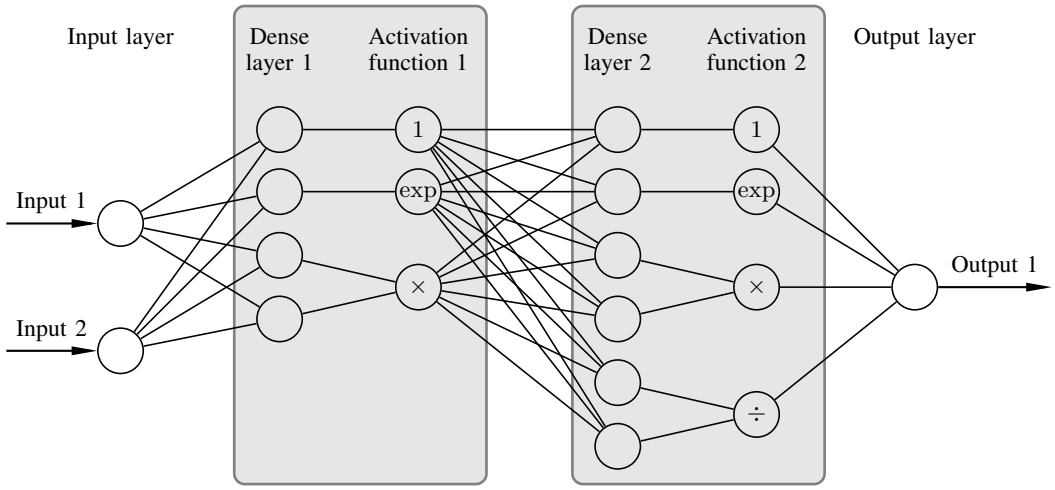

Fig. 3. Nominal expression tree ANN with two hidden layers, each marked by a gray box. The shown network was used in both examples of the paper.

comprises of a first term representing the nominal loss, a second sparsity-enforcing regularization term, and a third term for avoiding numeric errors caused by (close to) zero denominators in division expressions. Here we have chosen the nominal loss to be the $\mathcal{L}_2$-norm of the output error $\boldsymbol{y}-\hat{\boldsymbol{y}}$. The second term constitutes a $\mathcal{L}_1$ regularization, discussed next, while the final term implements log-barrier functions, discussed further below.

2) Pruning: For the trained network to represent expressions of adequate—human-readable—complexity, some degree of sparsity needs to be enforced upon the nominal expression tree, here exemplified by Fig. 3. This can be obtained by pruning the network and removing connections associated with insignificant weights and biases. There are many methods to achieve this, as further discussed in *e.g.* [20].

The purpose of the $\mathcal{L}_1$ regularization term in (3) is to encourage all but a few weights to be zero. This is typically known as Lasso regression, and constitutes the basis for one of several possible pruning techniques [20]. The scalar hyper-parameter λ in (3) determines the extent of regularization and thus constitutes a trade-off between accuracy and "approximate" sparsity. Increasing λ encourages weights to tend to zero. However, they might not attain the value of exactly zero (thus resulting only in "approximate" sparsity).

One way to enforce "true" sparsity of the network would be to remove all connections corresponding to weights and biases with magnitudes that fall under a certain cut-off size, defined by a hyper-parameter. However, when this pruning should take effect—during or after training, once or in every training epoch—is an aspect that has to be considered.

Letting $2N_e$ be the number of training epochs, we therefore apply the following pruning method:

1) Train without pruning for N_e epochs.
2) After each consecutive epoch, prune weights and biases with magnitudes smaller than a pre-defined hyper-parameter δ.
3) After $2N_e$ epochs, remove the weight or bias with the smallest magnitude.
4) Repeat the step above until the number of remaining weights and biases matches or is lower than a pre-defined hyper-parameter N_f.
5) Simplify the resulting expression using symbolic computations.

To avoid keeping non-influential parameters, a set of simplification rules are enforced at the end of step two. Parameters upstream from a node representing a multiplication by zero can be removed if they thus do not effect the final expression. For example, if the numerator of a division expression evaluates to zero, the entire division activation function, as well as all upstream nodes that do not provide input to other activation functions of the network, are removed. If instead the denominator becomes zero, the division is replaced with a unit function with the value of the nominator divided by the bias term of the denominator. The same idea is applied to the exponential function. If only the bias b remains upon pruning, the exponential function is replaced by the constant value $\exp(b)$. Depending on what base expressions are used, the list of simplification rules could readily be extended or modified.

3) Regularized division: The division base expression is problematic since it leads to numerical instability if the denominator approaches zero. In the literature, this has prompted *ad hoc* fixes, as exemplified by [21]. We approach the problem with modified log-barrier functions, otherwise commonly employed to enforce (approximations of) hard constraints in optimization problems. A hyper-parameter β defines a cut-off limit, and a term $D = -\log(|d|)$ is added to the optimization loss whenever a division denominator d evaluates to a value with $|d| < \beta$. In the last term of (3), D is the sum of all such terms, and μ is a hyper-parameter to be increased until possible numeric instability caused by (near) zero denominators vanishes.

$$
\begin{aligned}
V_2 =&(l_1 \exp(l_2 x_1 + l_3 x_2) + l_4 x_1^2 + l_5 x_1 + l_6 x_1 x_2 + l_7 x_2^2 + l_8 x_2 + l_9)(l_{10} \exp(l_2 x_1 + l_3 x_2) + l_{11} x_1^2 + l_{12} x_1 + l_6 x_1 x_2 + l_7 x_2^2 + l_{13} x_2 + l_{14}) \\
&+ l_{15} x_1^2 + l_{16} x_1 x_2 + l_{17} x_2^2 + l_{18} \exp(l_2 x_1 + l_3 x_2) + l_{19} \exp(l_{20} \exp(l_2 x_1 + l_3 x_2) + l_{15} x_2^2 + l_{21} x_1 x_2 + l_{22} x_2 + l_{23} x_1^2 + l_{24} x_1) \\
&+ l_{25} x_1 + l_{26} x_2 + l_{27} + \frac{l_{28} \exp(l_2 x_1 + l_3 x_2) + l_{29} x_1^2 + l_{30} x_1 + l_{31} x_1 x_2 + l_{32} x_2 + l_{33} x_2^2 + l_{34}}{\exp(l_2 x_1 + l_3 x_2) + l_{35} x_1^2 + l_{36} x_1 + l_{37} x_1 x_2 + l_{38} x_2 + l_{39} x_2^2 + l_{40}}.
\end{aligned} \tag{4}
$$

4) Implementation: We have implemented the proposed method in Julia, which is a language with native support for automatic differentiation. This makes implementing and training of the ANN straightforward. The ANN was implemented using the Flux [22] package, and the stochastic gradient optimizer Adam [23] was employed to perform weight and bias optimization within each training epoch.

The implementation relies on several hyper-parameters being defined. The perhaps most obvious one is the choice of base expressions and their structural organization into the nominal ANN, as exemplified in Fig. 3. We have also mentioned named hyper-parameters λ, μ, N_e, N_f, δ. In addition, the numeric optimization algorithm employed for back-propagation generally has several hyper-parameters. Finally, more hyper-parameters are needed to define how random repeated initialization, desirable to avoid finding shallow minima of the loss (3), is to be carried out.

What hyper-parameter values are adequate vary between modeling problems. Rather than listing numeric values that are hard to interpret out of context—and to provide additional detail to what is realistically possible to convey in the main text—we have therefore disclosed the implementation used to generate the results of this paper in [24].

III. Results

The expression corresponding to the nominal ANN of Fig. 3 is not quite human-readable even after simplification, as seen in (4), where AGE and WGT have been replaced by x_1 and x_2, respectively. Values of l_i resulting from optimization of (3) with $\lambda = 0$ and $N_f = \infty$ can be found in [24].

However, specifying to keep at maximum $N_f = 12$ parameters, the method manages to identify the expressions

$$\hat{V}_1 = \frac{6.35\text{WGT} - 1.43}{0.685\text{WGT} + 22.7}, \tag{5a}$$

$$\hat{V}_2 = 0.624\text{WGT}\exp(-0.0155\text{AGE}), \tag{5b}$$

for estimation of V_1 and V_2 generated by (1) and the covariates in [8]. The pruned networks corresponding to these expressions are shown in Fig. 4 and Fig. 5, and the model fit across the training data is shown in Fig. 6 and Fig. 7.

Decreasing to $N_f < 7$ instead results in the constant models $\hat{V}_1 = 5.98$ and $\hat{V}_2 = 0.600\,\text{WGT} - 0.00533\,\text{AGE}\,\text{WGT}$, which do not capture the inter-individual variability seen in the training data.

IV. Discussion

This paper demonstrates how symbolic regression networks can be used within pharmacometric covariate modeling. Particularly, the methodology can be used both to find parametric functional relations between covariates and PK model parameters and to optimize the numeric values of these relations.

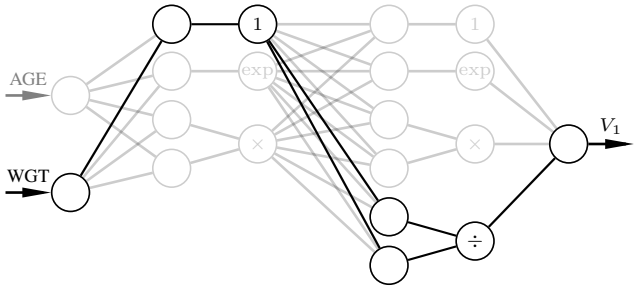

Fig. 4. Resulting trained ANN for the model $\hat{V}_1$ of (5a).

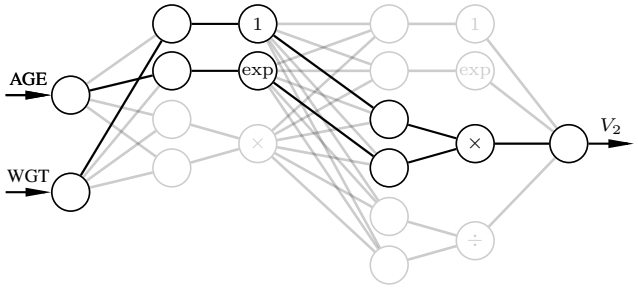

Fig. 5. Resulting trained ANN for the model $\hat{V}_2$ of (5b).

In the two provided examples, the final expressions (5) are of the same form as the "true" model (1), from which the training data was generated. In contrast to previous research on "symbolic regression" where the goal has been to identify physical laws, this is of minor interest in our context. Instead, we primarily wish to find expressions that provide an adequate balance between complexity (expression size) and fit to data. This trade-off is available foremost through the selection of base expressions and structure of the nominal ANN, and the degree of pruning through the final size hyper-parameter N_f.

Symbolic regression networks provides a large flexibility in the choice of which functional relations that are included. In this work, we have chosen base expressions that are commonly present in existing anesthetic models, see for example [8], [12]. However, these can be chosen freely to match underlying structures of the data, or based on expert input.

Although it is time-consuming to tune hyper-parameters, the alternative of evaluating all possible combinations of base expressions will be practically infeasible already for small model expressions.

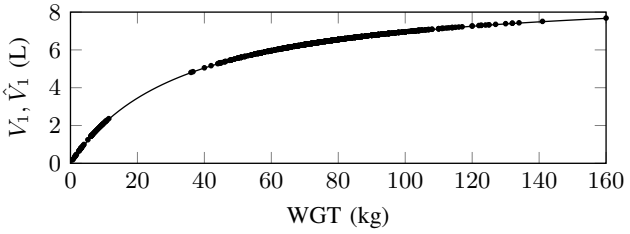

Fig. 6. Covariate WGT against PK parameter V_1 for training data (dots); generated by trained model (5a) (curve).

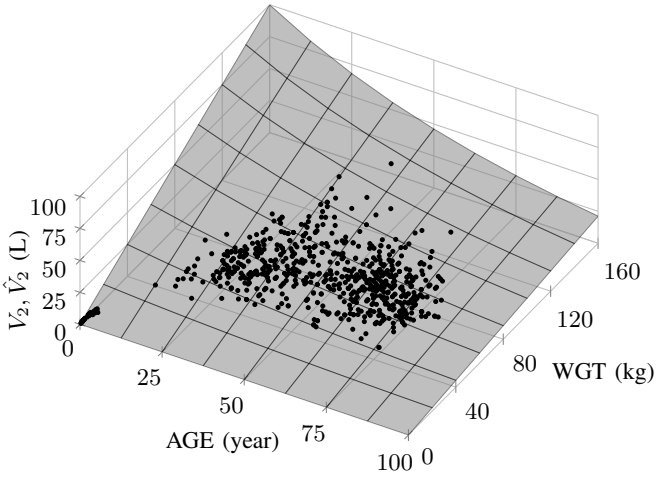

Fig. 7. Covariates AGE and WGT against PK parameter V_2 for training data (dots); generated by trained model (5b) (surface). Although hard to see, the dots lie (almost) perfectly on the surface.

In this paper, the main focus has been on the demonstration of the symbolic regression method, with the assumption that the "true" PK covariate model in (1) is known. However, in a real use-case this will not be the case. Instead, the model prediction should be compared to clinical data, which for this model is blood plasma sample concentrations with the corresponding drug infusion profiles, as described in [8]. Simulation of the PK model candidates gives a possibility to compute the model output error at each sample. By the computation of a statistic measure on the output error, such as the mean absolute error or the $\mathcal{L}_2$-norm, this constitutes our new training loss. While the training methodology remains the same, it now also includes a PK model simulation step. The framework in Julia allows for automatic differentiation through this simulation step.

These promising early results have motivated us to continue with the prospect to evaluating the methodology on actual pharmacological study data.

REFERENCES

[1] A. Absalom, *An overview of TCI & TIVA*. Cambridge, MA: Academic press, 2019.

[3] M. M. Sahinovic, M. M. R. F. Struys, and A. R. Absalom, "Clinical pharmacokinetics and pharmacodynamics of propofol," *Clinical Pharmacokinetics*, vol. 57, pp. 1539–1558, 2018.

[2] M. Ghita, M. Neckebroek, C. Muresan, and D. Copot, "Closed-loop control of anesthesia: Survey on actual trends, challenges and perspectives," *IEEE Access*, vol. 8, pp. 206 264–206 279, 2020.

[4] J. Gonzales-Cava, F. Bagge Carlson, O. Troeng, A. Cervin, K. van Heusden, G. Dumont, and K. Soltesz, "Robust PID control of propofol anaesthesia: Uncertainty limits performance, not PID structure," *Computer Methods and Programs in Biomedicine*, vol. 198, pp. 1–8, 2021.

[5] Y. Wahlquist, K. van Heusden, G. Dumont, and K. Soltesz, "Individualized closed-loop anesthesia through patient model partitioning," in *Conference of the IEEE Engineering in Medicine and Biology Society*, Quebec, Canada, 2020, pp. 361–364.

[6] K. van Heusden, J. M. Ansermino, K. Soltesz, S. Khosravi, N. West, and G. A. Dumont, "Quantification of the variability in response to propofol administration in children," *IEEE Transactions on Biomedical Engineering*, vol. 60, pp. 2521–2529, 2013.

[7] N. E. Jonsson and M. O. Karlsson, "Automated covariate model building within NONMEM," *Pharmaceutical research*, vol. 15, pp. 1463—-1468, 1998.

[8] D. J. Eleveld, P. Colin, A. R. Absalom, and M. M. R. F. Struys, "Pharmacokinetic–pharmacodynamic model for propofol for broad application in anaesthesia and sedation," *British Journal of Anaesthesia*, vol. 120, pp. 942–959, 2018.

[9] L. B. Sheiner and S. L. Beal, "Evaluation of methods for estimating population pharmacokinetics parameters. I. Michaelis-Menten model: routine clinical pharmacokinetic data," *Journal of Pharmacokinetics and Pharmacodynamics*, vol. 8, pp. 553–571, 1980.

[10] "Pumas AI," https://pumas.ai, accessed: 2022-01-26.

[11] B. Marsh, M. White, N. Morton, and G. Kenny, "Pharmacokinetic model driven infusion of propofol in children," *British Journal of Anaesthesia*, vol. 67, pp. 41–48, 1991.

[12] T. W. Schnider, C. F. Minto, P. L. Gambus, C. Andresen, D. B. Goodale, S. L. Shafer, and E. J. Youngs, "The influence of method of administration and covariates on the pharmacokinetics of propofol in adult volunteers," *Anesthesiology*, vol. 88, pp. 1170–1182, 1998.

[13] J. Schüttler and H. Ihmsen, "Population pharmacokinetics of propofol: A multicenter study," *Anesthesiology*, vol. 92, pp. 727–738, 2000.

[14] M. M. R. F. Struys, M. J. Coppens, N. De Neve, E. P. Mortier, A. G. Doufas, J. F. P. Van Bocxlaer, and S. L. Shafer, "Influence of administration rate on propofol plasma–effect site equilibration," *Anesthesiology*, vol. 107, pp. 386–396, 2007.

[15] P. Orzechowski, W. La Cava, and J. H. Moore, "Where are we now? a large benchmark study of recent symbolic regression methods," in *Proceedings of the Genetic and Evolutionary Computation Conference*, Kyoto, Japan, 2018, pp. 1183—1190.

[16] S. Udrescu and M. Tegmark, "AI Feynman: A physics-inspired method for symbolic regression," *Science Advances*, vol. 6, 2020.

[17] G. Martius and C. H. Lampert, "Extrapolation and learning equations," in *Proceedings of the 5th International Conference on Learning Representations (ICLR)*, Toulon, France, 2017.

[18] R. Tibshirani, "Regression shrinkage and selection via the lasso: A retrospective: Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 73, pp. 273–282, 2011.

[19] C. Louizos, M. Welling, and D. P. Kingma, "Learning sparse neural networks through l0 regularization," in *Proceedings of the 6th International Conference on Learning Representations (ICLR)*, Vancouver, Canada, 2018.

[20] M. Zhu and S. Gupta, "To prune, or not to prune: Exploring the efficacy of pruning for model compression," *arXiv:1710.01878 [cs, stat]*, 2017.

[21] S. S. Sahoo, C. H. Lampert, and G. Martius, "Learning equations for extrapolation and control," in *Proceedings of the 35th International Conference on Machine Learning*, Stockholm, Sweden, 2018.

[22] M. Innes, E. Saba, K. Fischer, D. Gandhi, M. C. Rudilosso, N. M. Joy, T. Karmali, A. Pal, and V. Shah, "Fashionable modelling with Flux," in *Proceedings of the 32nd Conference on Neural Information Processing Systems (NIPS)*, Montreal, Canada, 2018.

[23] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, San Diego, CA, 2015.

[24] Y. Wahlquist, "Pharmacometric covariate modeling," https://github.com/wahlquisty/pharmacometric-covariate-modeling, 2022, commit 71ea476.