Key Engineering Materials
ISSN: 1662-9795, Vol. 926, pp 2146-2153
doi:10.4028/p-5hf550
© 2022 The Author(s). Published by Trans Tech Publications Ltd, Switzerland.

Submitted: 2021-12-07
Revised: 2022-01-25
Accepted: 2022-01-26
Online: 2022-07-22

On the Identification of Material Constitutive Model Parameters Using Machine Learning Algorithms

Armando Marques[1,a*], André Pereira[1,b], Bernardete Ribeiro[2,c] and Pedro Prates[1,3,d]

[1]University of Coimbra, CEMMPRE, Department of Mechanical Engineering, Portugal

[2]University of Coimbra, CISUC, Department of Informatics Engineering, Portugal

[3]Department of Mechanical Engineering, Centre for Mechanical Technology and Automation (TEMA), University of Aveiro, Aveiro, Portugal

[a]armando.marques@uc.pt, [b]andre.pereira@dem.uc.pt, [c]bribeiro@dei.uc.pt, [d]prates@ua.pt

Keywords: Sheet Metal Forming; Machine Learning; Parameter Identification

Abstract. This work aims to evaluate the predictive performance of various Machine Learning algorithms when applied to the prediction of material constitutive parameters, particularly the parameters of the Swift hardening law. For this, datasets were generated from the results of the numerical simulations of uniaxial tensile tests. The Machine Learning algorithms considered for this study are: Gaussian Process, Multi-layer Perceptron, Support Vector Regression, Decision Tree and Random Forest. These algorithms were used to train metamodels based on training sets considering different numbers of materials and input parameters, which were then used to predict the hardening law parameters. The Gaussian Process algorithm achieved the overall best predictive performances. The results obtained show the potential of Machine Learning algorithms for application on the identification of material constitutive parameters.

Introduction

Sheet metal forming is a manufacturing technique capable of producing a high volume of components at a relatively low cost. For this reason, this technique is widely used in the automotive and aerospace industries. The market in which these industries operate is extremely competitive, and constant innovation is essential to guarantee profits while ensuring that all quality, safety and environmental demands are respected. In this context, the traditional trial-and-error approach to process design proves to be unfeasible, as it leads to significant costs, both in terms of time and scrap losses. In order to increase the efficiency of the design process, researchers look for alternative methods. An analytical approach is unsuitable for this type of problem, as sheet metal forming presents high non-linearity regarding boundary conditions, geometry and material properties. As such, the focus has been on the application of Finite Element Method (FEM) simulations to model forming processes. These simulations rely on the accurate representation of the plastic behaviour of the metal sheets, which is usually performed through constitutive models.

The constant development of new metallic alloys for use in the automotive industry has led to the development of new, more flexible constitutive models, that contain a larger number of parameters to identify. This has led researchers to seek new identification strategies, with a recent focus on the application of Data-Driven approaches, including Machine Learning (ML), to generate metamodels that can predict the constitutive parameters based on measurements extracted from mechanical tests. Morand and Helm [1] developed a mixture of experts model, consisting on multiple Multi-layer Perceptron (MLP) models trained simultaneously, in parallel, for different samples of the dataset, to predict hardening model parameters, based on stress-strain curves. The motivation to apply this kind of model, instead of a simple, single MLP model, comes from the fact that the solutions for the problem in question are non-unique. The authors defend that a single MLP model is unsuitable for such a problem. Guo et al. [2] developed a deep learning model to identify constitutive parameters, based on the combination of a convolutional neural network, used to filter noise in the input data, with a long-short term memory neural network, which is a type of neural network capable of

This article is an open access article under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0)

combining static and sequential data. This model was used to identify the elastic parameters (Young's modulus and Poisson's ratio) and three parameters of an exponential hardening law, based on the results of a uniaxial tensile test, including the strain fields from a rectangular region of the sample, load history and geometry. The work of Liu et al. [3] also represent another successful application of ML algorithms to sheet metal forming, in this case to process parameters. In this work, the authors proposed the application of ML techniques to the prediction of the loading stroke for four-point bending and air bending processes. The focus of the work was on the use of a physics-informed neural network algorithm, which the authors denominated Theory-Guided Deep Neural Network (TG-DNN). This algorithm differs from traditional data-driven neural networks, by applying prior knowledge of a process during the training process.

The present work focuses in evaluating the performance of various ML regression algorithms, namely Gaussian Processes, Multi-Layer Perceptron, Support Vector Regression, Decision Trees and Random Forest, when predicting the Swift hardening law parameters. The main obstacle to the successful application of ML based approaches derives from the large amount of data required to train the metamodels adequately; however, once an adequate dataset is established and the metamodel is trained, predictions for newly tested materials can be obtained almost instantly. As such, for this evaluation, various dataset sizes will be considered, to analyze how the performance of the algorithms evolves with the increase in data amount. These datasets will vary in both the number of materials considered, and the number of input variables included.

Supervised Machine Learning

Supervised learning algorithms are ML algorithms that learn based on available data. This process is generally called "training" and consists in mapping the relationships between input variables and output variables present in the training dataset provided to the algorithm. Based on these relationships, the internal parameters of a metamodel are adjusted, allowing it to predict the outputs for new sets of unseen input data [4]. This training process also includes the calibration of the algorithm hyperparameters. Hyperparameters are algorithm-level parameters that influence the training process, and consequently the characteristics of the resulting metamodel (for example, the number of layers in a neural network).

Gaussian Process (GP). A Gaussian Process (GP) is a collection of random variables that follows a Gaussian distribution and are fully defined by a mean function and a covariance function [5]. Generally, the mean function is assumed to be zero, so the covariance function is enough to completely define the GP. The GP regression metamodel can be represented by:

$$y(\mathbf{x}) = f(\mathbf{x}) + \epsilon, \tag{1}$$

where $y(\mathbf{x})$ is an observed output, $f(\mathbf{x})$ is the corresponding GP variable and ϵ represents a zero mean Gaussian white noise. Assuming that $\mathbf{y}(\mathbf{x}^{\mathbf{t}})$ is the target vector of outputs present in the dataset, and $\mathbf{y}(\mathbf{x}^{\mathbf{p}})$ is the vector of outputs to be predicted, the joint normal probability distribution is given by:

$$\begin{bmatrix} \mathbf{y}(\mathbf{x}^{\mathbf{t}}) \\ \mathbf{y}(\mathbf{x}^{\mathbf{p}}) \end{bmatrix} \sim \mathrm{N}\left(0, \begin{bmatrix} \mathbf{K}(\mathbf{X},\mathbf{X}) + \sigma_\epsilon^2\mathbf{I} & \mathbf{K}(\mathbf{X},\mathbf{X}_*) \\ \mathbf{K}(\mathbf{X}_*,\mathbf{X}) & \mathbf{K}(\mathbf{X}_*,\mathbf{X}_*) \end{bmatrix}\right), \tag{2}$$

with σ_ϵ^2 being the noise variance, $\mathbf{I}$ being the identity matrix and each $\mathbf{K}$ being a covariance matrix evaluated for all considered points, with $\mathbf{X}$ representing the training data from the dataset, and $\mathbf{X}_*$ the unseen data for which the metamodel will make predictions. Finally, the GP metamodel predictions are given by the following equations:

$$\bar{\mathbf{f}}_* = \mathbf{K}(\mathbf{X}_*,\mathbf{X})[\mathbf{K}(\mathbf{X},\mathbf{X}) + \sigma_\epsilon^2\mathbf{I}]^{-1}\mathbf{y}(\mathbf{x}^{\mathbf{t}}), \tag{3}$$

$$\mathbf{cov}(\mathbf{f}_*) = \mathbf{K}(\mathbf{X}_*,\mathbf{X}_*) - \mathbf{K}(\mathbf{X}_*,\mathbf{X})[\mathbf{K}(\mathbf{X},\mathbf{X}) + \sigma_\epsilon^2\mathbf{I}]^{-1}\mathbf{K}(\mathbf{X},\mathbf{X}_*), \tag{4}$$

where $\overline{\mathbf{f}}_*$ is the vector of predictions (mean), and $\mathbf{cov}(\mathbf{f}_*)$ represents the covariance of metamodel outputs, which acts as a measure of prediction uncertainty.

Multi-Layer Perceptron (MLP). The Multi-Layer Perceptron (MLP) is one of the most used types of neural networks. It is a feed forward network, composed of multiple layers of nodes (neurons) [6]. Each node of a given layer connects to the neurons of the next layer, passing information, but never connects with other nodes of the same layer. The input layer is the first layer of an MLP metamodel and contains as many nodes as there are input features in the dataset. It is followed by any number of layers called hidden layers. The number of hidden layers, as well as the number of nodes in each hidden layer are defined before the model training process. In the last layer, called output layer, predictions are made based on the information received from the previous layer. This algorithm is applicable to multi-output problems natively, with the number of nodes in the output layer equalling the number of output variables of the problem. The output of a single node from a hidden layer is given by:

$$z_i=\emptyset\left(\sum_j w_{ij}z'_j+d_i\right), \tag{5}$$

where z_i is the output of node i from the current layer, z'_j is the output of node j from the previous layer, w_{ij} is the weight associated with z'_j, d_i is a bias term and $\emptyset$ is a non-linear activation function. For regression problems, the output layer nodes have a similar formulation, but with no activation function. The training of this type of metamodel consists in adjusting the weights in order to obtain better predictions, in the so-called backpropagation algorithm. This is done by iteratively changing the weight values, in the network, to minimize the prediction error.

Support Vector Regression (SVR). The support vector regression (SVR) algorithm fits a function to the available data, while remaining as flat as possible. This is achieved by considering an error value γ, under which errors are accepted without penalty [7]. In practice, this means finding the function that can encompass the most training data points in the area around it, with a distance of γ or less. To give some flexibility to the metamodel, soft margins can be defined, in the form of slack variables ξ_i and ξ_i^*. Points at a distance between these variables and γ still affect the shape of the function, but under a penalty. When considering a linear problem, this metamodel is given by:

$$\begin{cases} \min\left(\frac{1}{2}\|\mathbf{w}\|^2+V\sum_i(\xi_i+\xi_i^*)\right) \\ \quad\quad\text{with:} \\ y_i - wx_i - \beta_0 \leq \gamma + \xi_i \\ wx_i + \beta_0 - y_i \leq \gamma + \xi_i^* \end{cases}, \tag{6}$$

with $\mathbf{w}$ being the normal weight vector to the surface of the function that is being approximated and V being a constant representing the trade-off between tolerance for deviations above γ and function flatness.

For non-linear problems, this algorithm can be generalized by introducing a kernel trick. A kernel consists of a similarity function between the inputs of the training data and inputs for which the metamodel will make predictions. With a kernel trick, the data is transformed to a higher dimensional space, allowing a linear metamodel to learn non-linear functions without specific mapping.

Decision Tree (DT) and Random Forest (RF). The Decision Tree (DT) algorithm works by applying a continuous splitting process to the training data, based on simple rules, that are chosen to minimize an error metric in the resulting nodes [8]. The most common error metric used on this case is the mean square error (MSE). The splitting process is repeated until each of the resulting final nodes has an MSE under a previously defined threshold, or until all final nodes are under a certain size (regarding the number of training data points that correspond to that node). The predictions from each node correspond to the average of the output values from the training points in the node.

A Random Forest (RF) metamodel is an ensemble metamodel, comprised of a series of decision trees, each trained with a randomly selected sample from the training data [9]. The predictions made by each decision tree are compiled, and the average value is chosen as the final prediction of the RF metamodel.

Dataset Generation

In this study, the inputs of the datasets used to train the metamodels consists of the values of force measured during uniaxial tensile tests. Force values were chosen instead of, for example, stress values because the current work is trying to establish a general methodology applicable to more complex cases, which will make use of results from different mechanical tests, including non-conventional tests where the stress and strain distributions are heterogeneous. In order to obtain these values, numerical simulations of the uniaxial tensile test were carried out with the DD3IMP code, an in-house finite element code optimized for the simulation of sheet metal forming processes [10]. The finite element mesh of the uniaxial tensile test sample is represented in Figure 1. This mesh contains 5160, eight-node hexahedral solid elements. The sample has a thickness of 0.5 mm, a total length of 90 mm and a total width of 20 mm. The inner rectangular (gauge) area has a length of 65 mm and a width of 10 mm.

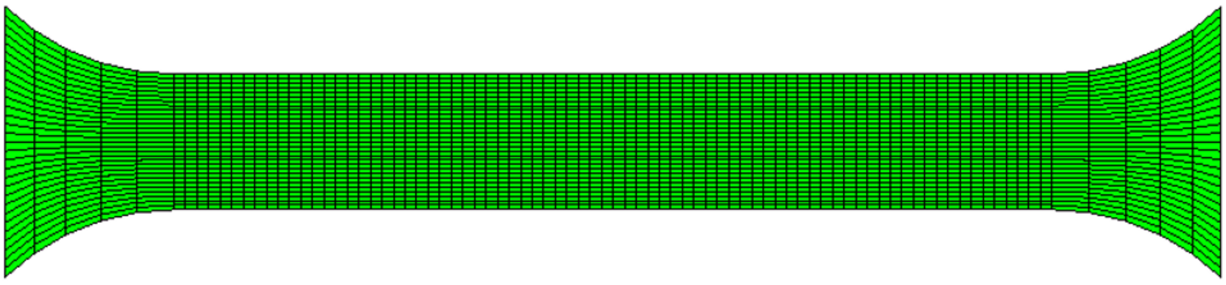
Fig. 1. Finite element mesh of the uniaxial tensile test sample

The materials used for this study were considered isotropic, and their hardening behavior was described by the Swift hardening law, which is expressed by:

$$Y = C(\varepsilon_0+\bar{\varepsilon}^p)^n, \tag{7}$$

where Y, is the yield stress, $\bar{\varepsilon}^p$ is the equivalent plastic strain and C, ε_0 and n are material parameters. Y_0 is the initial yield stress, and is given by:

$$Y_0 = C\varepsilon_0{}^n. \tag{8}$$

In order to guarantee that the various materials considered in this study are evenly distributed in the problem space, a Sobol sequence [11] was used to generate the values of Y_0 and n of the materials. For this, limits were defined for the values of Y_0 (between 150 MPa and 900 MPa) and n (between 0.12 and 0.25). These limits comprise typical values found in steel grades used in sheet metal forming processes. The values of C were calculated for each material using Eq. (8), and by assuming a ε_0 of 0.005.

Three different datasets were generated from the results of the numerical simulations. The first includes, as inputs, the values of force obtained in increments of displacement of 0.5 mm, up to a displacement value of 7.5 mm, for a total of 15 input variables for each material. This value of displacement was chosen because for this value, no simulation has reached the maximum force value. The second dataset is an extension of the first, including the values of force for the same displacement values, but also including the maximum force achieved in each simulation. This way, the dataset includes additional information that describes the force-displacement curves between the displacement value of 7.5 mm and the occurrence of necking. Finally, the third dataset extends the force values included up to a corresponding displacement of 13 mm, for a total of 26 input variables. In this case, for some of the materials considered, the simulation of the uniaxial tensile test surpasses the point at which the maximum force value is achieved, reaching displacement values where necking occurs. In those cases, the extra data can be considered as noise, and may negatively affect the predictive performance of the metamodel if the algorithms are not capable of handling the presence of noise. As such, this dataset was used to test the noise resistance of the various ML algorithms.

Metamodel Training and Performance Evaluation

In order to properly evaluate the predictive performance of the metamodels created by the various ML algorithms, the datasets are divided into training sets and testing sets. For this work, three different training sets are created from each dataset, containing the first 128, 512, and 1024 materials generated by the Sobol sequence. These values were chosen to guarantee that the materials present in each training set formed a uniform distribution within the problem space, as a Sobol sequence guarantees a uniform distribution for each 2^k points, where k is an integer. The training sets, as the name indicates, are used by the algorithms to train the metamodels. The training process consists in mapping the relationships between input variables and output variables present in the training set. An important part of the training process is the optimization of the algorithms hyperparameters. These are algorithm-level parameters that influence the training process, and consequently the characteristics of the resulting metamodels, for example, the number of layers in a neural network. This optimization was achieved through a trial-and-error approach, in which the training dataset was split into two subsets, one with 70 percent of the data, called calibration set, and the other with the remaining 30 percent, called validation set, 30 times. For each split, the algorithm trains a metamodel with the calibration set, then makes predictions for the validation set, and the predictive performance is measured. The group of hyperparameters that leads to the best average predictive performance for the validation set, across the 30 data splits, is then used to train the metamodels used to make predictions for the testing sets. The testing sets for each dataset contain 512 materials.

The predictive performance evaluation, during both the hyperparameter calibration and for the final metamodels, is based on performance metrics. In this work, the performance metric used is the root mean square relative error (RMSRE), which is given by:

$$\text{RMSRE}=\sqrt{\frac{1}{j}\sum_{i=1}^{j}\left(\frac{y_i - y_i^*}{y_i}\right)^2}, \tag{9}$$

where y and y^* are the real and predicted response values for output variables, respectively, and j is the number of testing points.

The metamodels were generated with Python libraries, specifically, GPy [12] for the GP algorithm and Scikit-learn [13] for the remaining algorithms.

Results and Discussion

In this study, the impact of four main factors in the predictive performance is evaluated. These factors are: the ML algorithm used to train the metamodel; the constitutive parameter for which the prediction is being made; the size of the training set; and the input variables considered in the dataset.

The values of RMSRE obtained for the various metamodels, are represented in Figure 2. This figure contains 9 charts in a 3 * 3 grid, where each row corresponds to a different constitutive parameter being identified, and each column corresponds to different training set size. Each chart includes the results obtained for each of the input parameter configurations considered: force values up to a tool displacement value of 7.5 mm during the uniaxial tensile test, labeled “7.5 mm”; same forces as the first case plus the maximum value of force achieved during the test, labeled “7.5 mm + Fmax”; and force values up to a tool displacement value of 13 mm, labeled “13 mm”.

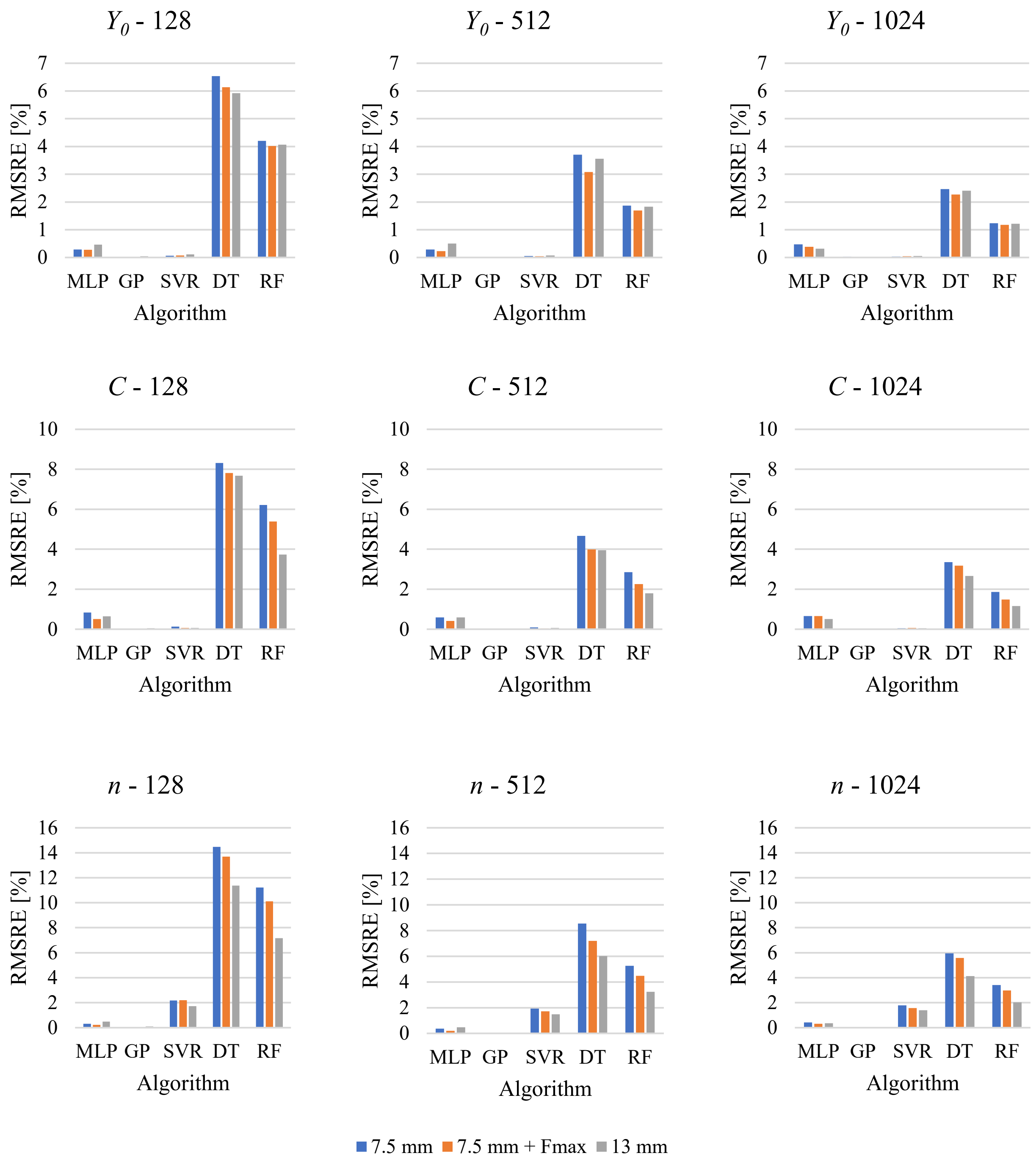

Fig. 2. RMSRE values (%) obtained by the algorithms when predicting the Swift hardening law parameters

In general, the predictive performance of the GP, MLP and SVR algorithms is superior to the performance of the DT and RF algorithms. The GP algorithm in particular presents near perfect predictions for all hardening law parameters, and all training set configurations. In fact, the highest RMSRE value obtained by this algorithm is 0.088 percent, when predicting the n values, with the 128-material training set containing the force values up to 13 mm of tool displacement.

For the GP, MLP and SVR algorithms, the training set changes, both in terms of input variables and training set size, did not lead to significant performance changes. From this we can conclude that the smaller training sets already contain enough information for these algorithms to properly generalize the underlying problem. It is worth considering that this is most likely the case because the problem in question is relatively simple, and the training sets were populated with a uniform distribution of materials, thanks to the use of a Sobol sequence. When applying the same algorithms to more complex problems, for example in the case of anisotropic materials where the parameters of a yield criterion also need to be predicted, it is expected that a significantly larger training set size will be required to obtain good predictive performances.

It is worth noting that, between these three algorithms, the SVR algorithm is the most sensitive to hyperparameter changes. For example, the values of RMSRE achieved by the final metamodels trained by the SVR algorithm when predicting the values of C are between 0.05 percent and 0.1 percent, regardless of other training set characteristics, however, during the hyperparameter optimization process, the RMSRE values obtained for the same cases reached 20 percent, meaning that when training a metamodel with this algorithm, special care must be taken when selecting the hyperparameters.

When looking at the DT and RF algorithms, the increase in training set size leads to a consistent increase in performance. This can be explained by the ability of the models to create more data splits before reaching the final prediction nodes, which gives the models more flexibility when making predictions. Regardless, it is clear that these algorithms require a significantly larger training set to perform well when compared to the GP, MLP and SVR algorithms. Also, it is expected that the increase in training set size will lead to diminishing returns. As such, it is likely that regardless of training set size, the DT and RF algorithms will never achieve performances on par with the remaining algorithms, particularly when compared with the GP algorithm. The change in training set inputs has a more noticeable influence in the performance of the DT and RF algorithms, and it is clear that the inclusion of more information, regardless of noise, leads to marginal improvements, when predicting the C and n parameters.

Conclusions

In this work, five different ML algorithms were used to predict the values of the Swift hardening law parameters, with the goal of evaluating their performance when applied with different training set sizes and input variable configurations.

From the various ML algorithms, the GP algorithm achieved the best performances for all cases, with the MLP and SVR also performing well enough to be considered competitive alternatives. The DT and RF algorithms were clearly inferior.

The increase of the training set size only had a significant impact on the performance of the DT and RF algorithms, which shows that the smaller training set already provided enough information for the remaining algorithms to accurately generalize the problem.

The variance in input parameter configuration does not lead to significant performance changes for the GP, MLP and SVR algorithms, but the DT and RF algorithms saw improvements when the models were trained with the training sets containing more input parameters, despite then being potentially noisy.

Overall, the results obtained highlight the capabilities of Machine Learning algorithms when applied to the identification of material constitutive parameters. As such, future work will involve the application of these algorithms to more complex problem, i.e., with more parameters to be identify. In particular, anisotropic yield criterion parameters will be included in the identification process.

Acknowledgements

This research is sponsored by FEDER funds through the programs COMPETE – “Programa Operacional Factores de Competitividade” and Centro2020 – “Centro Portugal Regional Operational Programme”, and by national funds through FCT – “Fundação para a Ciência e a Tecnologia”, under the projects UIDB/00285/2020, UIDB/00326/2020, UIDB/00481/2020, UIDP/00481/2020 and CENTRO-01-0145-FEDER-022083. It was also supported by projects PTDC/EME-EME/31243/2017 (POCI-01-0145-FEDER-031243) and PTDC/EME-EME/31216/2017 (POCI-01-0145-FEDER-031216), co-funded by FCT, by FEDER through the program Portugal-2020 (PT2020), and by POCI. The first author, Armando Marques, was supported by a grant for scientific research from the Portuguese Foundation for Science and Technology (ref. 2020.08449.BD). All supports are gratefully acknowledged.

References

[1] L. Morand, D. Helm, A mixture of experts approach to handle ambiguities in parameter identification problems in material modeling, Computational Materials Science. 167 (2019) 85-91. doi.org/10.1016/j.commatsci.2019.04.003

[2] Z. Guo, R. Bai, Z. Lei, H. Jiang, D. Liu, J Zou, C. Yan, CPINet: Parameter identification of path-dependent constitutive model with automatic denoising based on CNN-LSTM, European J. of Mechanics. 90 (2021). doi.org/10.1016/j.euromechsol.2021.104327

[3] S. Liu, Y. Xia, Z. Shi, Z. Li, J. Lin, Deep learning in sheet metal bending with a novel theory-guided deep neural network, IEEE/CAA J. of Automatica Sinica. 8 (2021) 565-581. doi.org/10.1109/JAS.2021.1003871

[4] P. Cunningham, M. Cord, S.J. Delany, Supervised Learning, in: Machine Learning Techniques for Multimedia, Cognitive Technologies, Springer, Berlin, pp. 21-49. doi.org/10.1007/978-3-540-75171-7_2

[5] D. J. Lin, L. Huang, H.B. Zhou, Forming defects prediction for sheet metal forming using Gaussian process regression, in: Proceedings of the 29th Chinese Control and Decision Conference, Chongqing, China, 2017. doi.org/10.1109/CCDC.2017.7978140

[6] C. Silva, B. Ribeiro, Redes neuronais, in: Aprendizagem Computacional em Engenharia, Imprensa da Universidade de Coimbra. (2018) pp. 27-66. doi.org/10.14195/978-989-26-1508-0

[7] A. J. Smola, B. A. Schölkopf, Tutorial on support vector regression, Statistics and Computing. 14 (2014) 199-222. doi.org/10.1023/B:STCO.0000035301.49549.88

[8] D. Kaur, D. Wilson, M. Forrest, L. Feng, Regression tree and neuro-fuzzy approach to system identification of laser tap welding, in: Proceedings of the 2005 Annual Meeting of the North American Fuzzy Information Processing Society, Detroit, USA, 2005. doi.org/10.1109/NAFIPS.2005.1548554

[9] L. Breiman, Random Forests, Machine Learning. 45 (2001) 5-32. doi.org/10.1023/A:1010933404324

[10] L.F. Menezes, C. Teodisiu, Three-dimensional numerical simulation of the deep-drawing process using solid finite elements, J. of Materials Processing Technology. 97 (2000) 100-106 doi.org/10.1016/S0924-0136(99)00345-3

[11] S.S. Garud, I.A. Karimi, M. Kraft, Design of computer experiments: a review, Computers and Chemical Engineering. 106 (2017) 71-95. doi.org/10.1016/j.compchemeng.2017.05.010

[12] GPy: A gaussian process framework in python. Available online: http://github.com/SheffieldML/GPy (visited on 26 November 2021)

[13] F. Pedregosa, G. Varoquaux, A. Gramfort, et al., Scikit-learn: Machine learning in Python, J. of Machine Learning Research. 12 (2011) 2825-2830.