Explaining Cyber-Physical Systems Using Decision Trees

Swantje Plambeck*, Görschwin Fey†
Institute of Embedded Systems
Hamburg University of Technology
Hamburg, Germany
{*swantje.plambeck, †goerschwin.fey}@tuhh.de

Jakob Schyga‡, Johannes Hinckeldeyn§, Jochen Kreutzfeldt¶
Institute for Technical Logistics
Hamburg University of Technology
Hamburg, Germany
{‡jakob.schyga, §johannes.hinckeldeyn, ¶jochen.kreutzfeldt}@tuhh.de

***Abstract*—Cyber-Physical Systems (CPS) are systems that contain digital embedded devices while depending on environmental influences or external configurations. Identifying relevant influences of a CPS as well as modeling dependencies on external influences is difficult. We propose to learn these dependencies with decision trees in combination with clustering. The approach allows to automatically identify relevant influences and receive a data-related explanation of system behavior involving the system's use-case. Our paper presents a case study of our method for a Real-Time Localization System (RTLS) proving the usefulness of our approach, and discusses further applications of a learned decision tree.**

I. Introduction

Today's technical or industrial systems become more and more complex. They consist both of digital embedded systems as well as sensors which depend on environmental influences. Such systems are known as Cyber-Physical Systems (CPS). Digital components have a finite number of states because of their limited amount of (binary) memory and typically demonstrate a deterministic behavior. Still, the upper limit for the number of states is exponential in the system's memory. CPS have, apart from that, an additional complexity resulting from environmental influences or external configurations. Modeling a system's dependencies on the environment includes multiple challenges because of continuous, possibly noisy signals and a lack of information on the relevant external influences. Nevertheless, information on dependencies between external influences and the system's outputs enables explaining the behavior of a system or predicting system behavior under a given environment. Furthermore, it supports applications like monitoring, optimizing the setup of the system in a productive use-case, or evaluating a system's performance [1, 2].

We consider Decision Tree Learning (DTL) in combination with clustering to represent the influence of environmental factors and configuration parameters to a CPS. DTL is a well-established machine learning approach and has already been applied in several applications ranging from identification of test cases on a given system [3], fault diagnosis in circuits [4], or anomaly detection [5]. For anomaly detection, the approach in [5] combines decision trees and clustering. We consider using both of these methods in a different combination. Decision tree models show advantages especially in their interpretability and simple structure. Furthermore, efficient learning algorithms exist. Clustering supports the construction of reasonable classes, especially for continuos signals, and contributes to the interpretability of class labels.

In our approach – described in Section III – we collect measurements from a CPS under scenarios with differing external influences. To enable DTL on possibly continuous measurements, a categorization is performed. Categorization can take place on an application-dependent basis, i.e., with categories resulting from the considered use-case of a system. If no such categories exist for a use-case, categories are determined by clustering the received data points. The goals of this approach include:

- automatically determining which of the considered influences are relevant,
- identifying correlations between influences and the system's outputs, and
- explaining system behavior on basis of the information included in a learning data set.

We present a case-study with our approach in Section IV considering a Light Detection And Ranging (LiDAR) localization system. This system depends on external influences – especially on the configurations supporting the localization such as the quality of a recorded map of the facility. Furthermore, LiDAR systems are relevant for many applications including automation in warehouses [6, 7]. Thus, we consider a logistics scenario where the localization system is installed on a forklift or a similar vehicle for consignment. In this scenario, we examine whether the localization system is able to serve a logistics application in a given warehouse based on the system's position accuracy. Application-dependent categories for the positioning accuracy of the localization system are "small object-precise", "medium object-precise", and "tray-precise". The results of our case study show that the combination of clustering and DTL allows to identify relevant features. Meanwhile, the decision tree supports explaining the localization accuracy.

Our main contribution is to define a strategy which combines clustering and DTL to automatically learn dependencies between external influences and a system's outputs. The case study provides examples for applications of our approach and proves its usefulness in the results of Section V.

978-1-6654-8201-1/22/$31.00 ©2022 IEEE
DOI 10.1109/CAADCPS56132.2022.00006

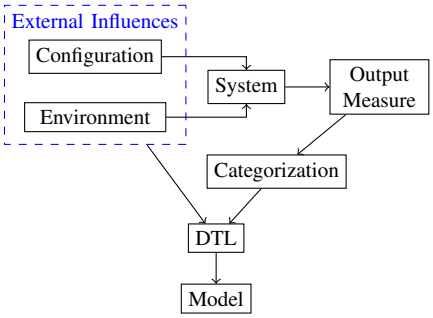

Fig. 1: Process of Modeling

II. Related Work

There exist multiple approaches to model the behavior of CPS which serve different purposes from prediction for monitoring over behavioral models to models on environmental influences.

Machine learning, especially neural networks, are widely used to predict a system's behavior based on previous observation of the system and serve applications like monitoring [1] or anomaly detection [8]. These approaches often show good prediction accuracy. Nevertheless, neural networks suffer from limited interpretability.

Furthermore, behavioral models such as linear models, e.g., linear time-invariant (LTI) systems [9, 10] or discrete automaton models [11, 12] exist. These models display the temporal behavior of a system on a discrete time scale. Hybrid automata [13] display time- and value-continuous CPS by representing continuous behavior within discrete process modes of the system. In contrast, our approach models interdependencies of system output and environmental factors or system configurations.

Environmental influences are often modeled by factor analysis [14] which usually determines linear relationships between external influences and a system's output. This approach, thus, requires a linear dependency of the system output and the environmental influences or configurations. Some approaches to non-linear factor-analysis exist which still require specific mathematical relations [15]. Our approach can model even non-linear dependencies and especially allows modeling of categorical influences and outputs.

III. Decision Tree Modeling

Fig. 1 shows the conceptual procedure of our approach. We have a system that is installed in a real-world environment. The system is considered to include only the system dynamics, i.e., the internal behavior. Thus, besides "pure" environmental influences like, e.g., temperature of the surroundings, system configurations are considered as external influences. Configurations are for example the usage of odometric measurements to support localization in a localization system or the target temperature of a heating system. We summarize environmental influences and configuration parameters under *external influences*.

From the system, we receive measurements of outputs. We are not creating a behavioral model of the system, but seek general dependencies between the system and the external influences. Thus, we determine output measures that are not changing over time but describe the system's status or performance in an abstract manner. Examples for such outputs are the localization accuracy, i.e., the mean expected localization error of a localization system or the mean deviation between the target temperature and current temperature of a heating system. Furthermore, we collect measures of relevant external influences. Output measurements are categorized to achieve an interpretable classification. The categorized outputs together with the information on external influences are input to DTL and result in a model of dependencies between the system's output and the external influences.

A. Categorizing the Output

The first step after collecting data is the categorization of output samples. For a general approach, we consider an output measurement to be multidimensional, i.e., output data $\mathbf{o}$ come from a possibly multidimensional space O. Furthermore, values from each dimension of O might either be continuous, i.e., a subspace of $\mathbb{R}$, or categorical. The categorization for output signals is defined by a function

$$\gamma : O \to \Sigma_O, \tag{1}$$

where Σ_O is a set of categorical values. Without loss of generality, we assume the elements of Σ_O to be one-dimensional such that $\Sigma_O = \{1, \dots, |\Sigma_O|\}$. The mapping γ is performed based on predefined (possibly multidimensional) categorization subspaces of O: $S_1^O, S_2^O, \dots, S_{|\Sigma_O|}^O$. The set size $|\Sigma_O|$ gives the number of categorization subspaces. Each subspace S_j^O maps to a symbol $o^j \in \Sigma_O$. Using the categorization subspaces, we define the categorization function as

$$\gamma_O(\mathbf{o}_c) = o^j, \quad \text{for } \mathbf{o}_c \in S_j^O. \tag{2}$$

The categorization subspaces are disjoint and cover the whole domain O.

So far, we assumed the categorization subspaces to be given before performing the categorization. Often, meaningful categories result from the considered application. Nevertheless, if this is not the case, we determine a suitable categorization based on the given data. A well-established approach for this is clustering. We consider k-nearest neighbor (kNN) clustering which allows choosing the number of clusters a priori. An advantage of this clustering is as well that linear borders between the clusters exist such that the categorization subspaces can be described as intervals in the dimensions of O. The number of clusters for kNN is set based on prior knowledge of the system. If no prior knowledge exists, we use an elbow criterion to determine the optimal number of clusters. The elbow-criterion determines the optimum as a compromise between the number of clusters and the clustering accuracy [16]. The clustering accuracy, here, is determined by the sum of squared distances of samples to their closest cluster center. An optimum is found in the "elbow" of the generated curve, i.e., when an increase in the number of clusters does not significantly change the clustering accuracy anymore.

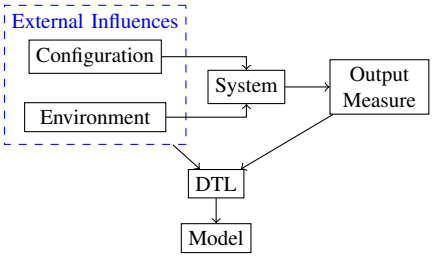

Fig. 2: Process of Modeling for Regression Tree Learning (RTL)

B. Decision Tree Learning

Definition 1. *Decision Tree [17]. A decision tree is a tree* $T = (V, E)$ *with vertices* V *and edges* E *that represents a classifier* $d : \mathcal{X} \to \mathcal{C}$*. The set* $\mathcal{X}$ *consists of vectors of feature values* $\mathbf{f} = [f_1, ..., f_n]$*, while* $\mathcal{C}$ *is a set of classes.*

A decision tree learner constructs a decision tree based on a set of learning samples $\mathcal{L}$, where each element $\rho = \langle \mathbf{f}, c \rangle \in \mathcal{L}$ is a tuple of a feature vector $\mathbf{f} \in \mathcal{X}$ and a class label $c \in \mathcal{C}$. The decision rule d adapts to these samples. The tree T has exactly one vertex without incoming edges, which is the root r. The set of vertices with no outgoing edges are the leaves V_L of the tree. The depth of the tree is the maximum length of a path from a root node to a leaf node. All vertices $V \setminus V_L$ are *inner vertices* of the tree. We associate each vertex v in T to a subset S of $\mathcal{X}$, denoted by $v \sim S$. Each leaf $v_l \sim S$ has a label $c_{v_l} \in \mathcal{C}$ given by

$$c_{v_l} = d(\mathbf{f}) = \underset{c}{\operatorname{argmax}} : \#c, \text{ where} \tag{3}$$
$$\#c = |\{\mathbf{f} : \langle \mathbf{f}, c \rangle \in \mathcal{L} \text{ and } \mathbf{f} \in S\}|.$$

A learned decision tree may be *pruned*, i.e., the depth of the decision tree is limited. This can happen directly or indirectly, e.g., by limiting the minimum number of samples per leaf.

We observe the system in multiple scenarios and receive a set of observations $\mathcal{O}$, where each element of the set is a tuple $(\mathbf{e}, \mathbf{o})$ of concrete external influences $\mathbf{e}$ and an output measure $\mathbf{o}$. For DTL, the collected external influences of an observation form the feature vectors, while the categorized output of the same observation serves as the class label:

$$\forall (\mathbf{e}, \mathbf{o}) \in \mathcal{O} : \quad \mathbf{f} = \mathbf{e}, \tag{4}$$
$$c = \gamma(\mathbf{o}). \tag{5}$$

After this mapping, we receive the set $\mathcal{L}$ of learning samples for DTL.

C. Decision Tree Regression

So far, we performed a categorization of the continuous measurements before decision tree learning. Nevertheless, there exist algorithms for decision tree learning based on continuous outputs resulting in so-called *Regression Trees* or decision trees for regression. Fig. 2 shows an alternative procedure with Regression Tree Learning (RTL). In contrast to Fig. 1, the categorization step is omitted as the regression tree itself performs the categorization.

We consider RTL as a parallel approach to DTL with clustering if the number of clusters approaches the number of measurements. A difference between the regression tree and the decision tree remains in the labeling of the leaf nodes, i.e., the class labels. The regression tree labels leaves with the mean value of the output measures of all samples in the considered leaf while the decision tree labels a leaf with the most common class, i.e., discretization subspace, as stated in Equation 3. A regression tree, thus, cannot be used if the output measure of the CPS is not purely continuous, i.e., at least one dimension of the measured output has categorical values.

D. Applications

Our approach offers a possibility to automatically refine measurements from a CPS to determine dependencies between external influences and status or performance measures of the system. Further paragraphs of this section present additional applications which benefit from or extend the learned decision tree.

a) Explaining Behavior: For a given observation, the decision tree explains how the observed output measure, i.e., system behavior, correlates to external influences, i.e., configurations and environmental influences. An observation corresponds to a path in the decision tree. Based on this path, we can conclude which influences from the complete set of external influences were significant to result in the corresponding leaf, i.e., to observe the given output.

b) Identifying Relevant Influences: For many CPS, it is not known which external factors influence the system's behavior. A decision tree provides information on the importance of the used features. The importance displays which and how many inner nodes of the tree consider a feature as the splitting criterion. Based on the importance of features, i.e., external influences, the decision tree enables identifying relevant influences that have a significant impact on the system's output.

c) Predicting Behavior: Provided that the used influences abstract the system behavior from an explicit environment and configuration, the decision tree represents a model of the system that is transferable to new scenarios or environments. In our approach, the decision tree represents correlations or dependencies from external influences based on the given data. An extension to a model of the system is approached if all existing influences are collected and sufficiently many learning samples are provided. Determining modeling capabilities and the necessary amount of data and influences is part of future extensions of our approach. If a decision tree model is achieved, the model can be used to predict the system behavior: given a vector of external influences, the decision tree model gives an estimate for the system output.

d) Optimizing the Exterior: Considering that the decision tree explains the system output, we can use the decision tree model to identify which factors of influence can be adapted to improve the system performance in a specific use-case, e.g., to improve the localization error of a localization system.

Our procedure presents an approach to automatically learn a decision tree that represents dependencies between the system's output and the external influences. Such a representation supports the above described applications and offers capabilities for an extension to a system model.

IV. Case Study - Real-Time Localization Systems

To validate the usefulness of our approach, we present a case study of our method for a Real-Time Localization System (RTLS). The euclidean position error, i.e., the distance between the estimated position and the true position, is the main metric regarding the performance of an RTLS and is therefore regarded as the system's output. The RTLS examined in this case study is based on Light Detection And Ranging (LiDAR). The system determines its position by measuring the distances with respect to the surrounding contour and a prerecorded reference map [18].

A. Experimental Setup

Experiments are conducted at the testing facility of the TUHH Institute for Technical Logistics on an area of $80\,\mathrm{m}^2$. The area is equipped with a high-precision optical motion capture system serving as a positioning reference to determine the positioning error [19]. The coordinate systems of the localization and the reference frames are aligned by applying the Umeyama-Alignment [20] and time synchronization is achieved by using the precision time protocol. The LiDAR system is carried by a robot to enable repeatability of the experiments. The robot automatically follows a trajectory, i.e., a path on the test area. Throughout our case study a rectangular trajectory is used in all scenarios. The position information is received with a frequency of $20\,\mathrm{Hz}$. In addition to the localization and reference system's position measurements, external influences of each experiment are collected.

B. Observations

The position error is determined for each experiment at 17 points on the considered trajectory. The system's output is defined as the mean error of all measurements from one experiment. Considered influences on the quality of the localization are the support by reflectors or by an inertial measurement unit (IMU), the map quality of the reference map, and the field of view of the LiDAR sensor. Further possible influences like light conditions, speed of movement, or sharpness of the room's contour exist, but are not considered in this exemplary case study. Parameter values of the considered influences are listed in Table I. Reflector support is provided if designated objects with reflecting surface are placed at fixed positions in the facility. Those reflectors represent reference points for the localization system. An IMU can support the localization by measuring the acceleration of the robot. The map quality describes the matching between the reference map of the localization system and the actual surroundings, e.g., by point cloud matching. The real surroundings typically have a varying contour because objects move, disappear or new objects are placed after recording the reference map. In our evaluation, the map quality is a qualitative parameter, whereby "good" refers to a map that has been recorded immediately before the measurement, "medium" defines a map that has minor differences to the current surroundings and "bad" is a map that has major differences to the current surroundings. Finally, the field of view of the LiDAR sensor is considered as a factor of influence. Depending on the mounting of the sensor on the robot, the field of view is limited. The maximum field of view of the used LiDAR sensor is 270 degree. In our experimental setup, we are able to set the field of view software-wise. Thus, the field of view is a continuous-valued influence.

Factor of Influence	Parameter Values
Reflector Support	on, off
IMU Support	on, off
Map Quality	good, medium, bad
Field of View	$[0°, 270°]$

TABLE I: Parameter values of external influences

	Reflector	IMU	Map	Field of View
Scenario 1	on	off	good	270°
Scenario 2	off	off	good	270°
Scenario 3	off	off	good	90°
Scenario 4	on	off	good	90°
Scenario 5	on	on	good	270°
Scenario 6	off	off	medium	90°
Scenario 7	off	off	bad	90°

TABLE II: Scenarios

C. Categorization

After collecting observations from the system, i.e., running experiments with different specifications of the external influences reflector support, IMU support, map quality, and field of view, the categorization on the output measurements takes place. From a warehouse application perspective, we find three meaningful categories for the mean localization error:

- tray-precise: mean error in $50\,\mathrm{mm}$ - $200\,\mathrm{mm}$
- medium object-precise: mean error in $20\,\mathrm{mm}$ - $50\,\mathrm{mm}$,
- small object-precise: mean error $< 20\,\mathrm{mm}$.

In this scenario, we examine whether the localization system is able to serve automatic consignment of different objects in a warehouse.

Apart from the application-dependent categories, we apply clustering to determine data-based categories emulating a scenario without prior knowledge on a meaningful categorization.

V. Results

In the following, we present learned decision trees that consider the localization quality of the given localization system as class labels. DTL and RTL are done with python, using the scikit learn package [21]. From the system, we collect measurements from seven different scenarios, listed in Table II. For each scenario, three experiments are conducted such that a learning set $\mathcal{L}$ of 21 data points results.

a) Application-Related Validation: First, we learn a decision tree using the application-related categories "tray-precise", "medium object-precise", and "small object-precise". Fig. 3 shows the decision tree model learned from these categories. The learned decision tree considers only the features

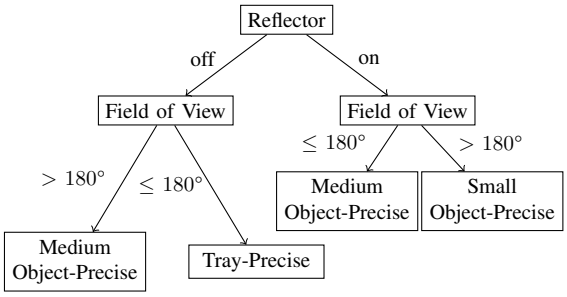

Fig. 3: Decision Tree Model for Application-related Categories

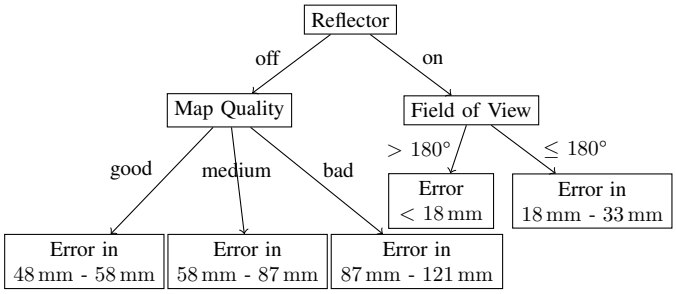

Fig. 4: Decision Tree Model for Seven Clusters

reflector support and field of view. Thus, IMU support and map quality are not or less relevant for the localization quality of the LiDAR system. Furthermore, the decision tree shows the best localization quality, i.e., small object-precise localization, with activated reflector support and a field of view of more than $180°$. Medium object-precise localization is still possible if either the field of view is large or reflector support is activated. Otherwise, only tray-precise localization is achieved. Thus, we could derive actions to adapt the external influences to the use-case's requirements on the localization quality.

b) Fitting the Clustering: If no application-related categories are known or given, categorization is done based on clustering of the measurements. For kNN clustering, we have to define the number of clusters a priori. We present exemplary results for seven clusters, i.e., where the number of clusters equals the number of scenarios from Table II, and for three clusters, which is the optimum number of clusters found with an elbow-criterion.

The clusters for $k = 7$ result to $< 18\,\mathrm{mm}$, $18\,\mathrm{mm}$ - $33\,\mathrm{mm}$, $33\,\mathrm{mm}$ - $48\,\mathrm{mm}$, $48\,\mathrm{mm}$ - $58\,\mathrm{mm}$, $58\,\mathrm{mm}$ - $87\,\mathrm{mm}$, $87\,\mathrm{mm}$ - $121\,\mathrm{mm}$, and $> 121\,\mathrm{mm}$. Fig. 4 shows the decision tree model for these categories. As before, field of view and reflector support are relevant features for the localization quality. Additionally, the map quality is considered to classify the measurements. We observe that not all of the clusters correspond to a leaf of the tree. Thus, the seven experimental scenarios do not have distinct error values, but some of them result in similar localization errors.

Applying kNN with three clusters results in the categories $< 36\,\mathrm{mm}$, $36\,\mathrm{mm}$ - $86\,\mathrm{mm}$, and $> 86\,\mathrm{mm}$. Fig. 5 shows the decision tree learned with these clusters. With three clusters, all classes are present in the leaves of the decision tree which

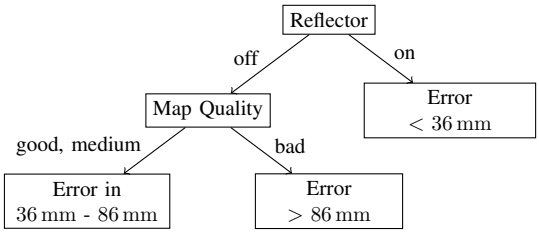

Fig. 5: Decision Tree Model for Three Clusters

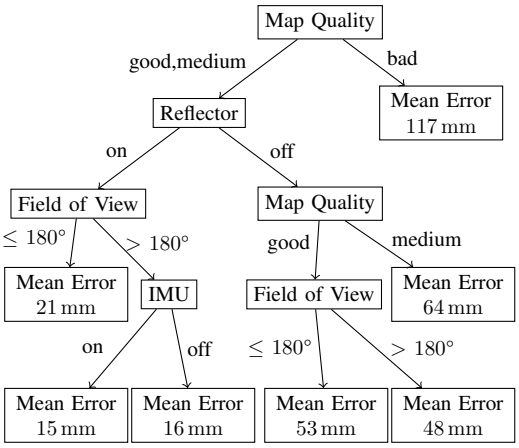

Fig. 6: Decision Tree Model from RTL

shows that the clustering is more useful compared to the result with seven clusters. The relevant features, in this case, are the reflector support and the map quality.

c) Regression Tree Comparison: An alternative to clustering is regression tree learning. As stated earlier, RTL is a parallel approach to DTL when the number of clusters approaches the number of scenarios or samples. The tree from RTL for our case study is shown in Fig. 6. The regression tree uses all features because – without pruning – RTL separates the measurements into all scenarios.

There exist two leaves with mean errors $15\,\mathrm{mm}$ and $16\,\mathrm{mm}$, respectively. Considering measurement noise those errors should form a single class of localization errors which shows that the regression tree overfits the learning data. Similarly, too many clusters lead to overfitting of data. Pruning the tree could overcome the overfitting. Nevertheless, deriving the necessary pruning criterion to avoid use-case-dependent overfitting a priori is difficult. Categorization intervals or the number of clusters, on the other hand, are more easily defined from the application perspective.

From the regression tree, we receive the feature importance on all considered features which are listed in Table III. The feature importance is a value between 0 and 1 and the sum of feature importances over all features is 1. Thus, the feature importance represents in how far a feature contributes to the identification of a valid category in relation to the other features. The results show that the map quality is most important, followed by the reflector support. Field of view and IMU support are less important for the localization quality, i.e., the output of the system.

Reflector	IMU	Map Quality	Field of View
0.27	$4.68 \cdot 10^{-5}$	0.73	$4.4 \cdot 10^{-3}$

TABLE III: Feature Importance

Model	App	$k=3$	$k=5$	$k=7$	$k=11$	$k=16$	RTL
Learning	90	100	95	90	62	52	62
Validation	40	100	60	60	20	0	20

TABLE IV: Accuracy of Decision Tree Models in %

d) Validating the Prediction Accuracy: To evaluate the accuracy of the classification, we classify each of the 21 experiments from the learning set together with five additional validation samples with the learned decision trees and compare the predicted class to the actual continuous output of the experiment. For the regression tree, the leaves do not represent error intervals. Thus, the correct category, i.e., leaf in the regression tree is the one with a mean value closest to a sample's output value.

Table IV shows the ratio of correctly classified samples in percent for clustered categories with different numbers of clusters k, the application-related approach (App), and RTL. The classification accuracy is typically better for smaller k. Especially, for $k = 1$ the classification accuracy trivially results to 100%. Nevertheless, the accuracy provides information whether a classification is useful. Clustering with $k > 7$ is likely to overfit the data because only seven initial scenarios exist which the decision tree could separate. Nevertheless, if the number of scenarios is not known a priori the number of clusters might be chosen larger. The regression tree achieves better results than clustered categories for large k. The reason is that the regression tree returns a mean value while the decision tree selects always one class label. Still, the tree with $k = 7$ performs better than the regression tree. The best classification accuracy results for the decision tree model with three clusters, i.e., with optimal k according to the elbow criterion.

For this case study, only a small number of samples from the complete space of external influences were taken. Improved results could be achieved with a larger learning set.

VI. Conclusion

In this paper, we presented a procedure to create a decision tree that automatically finds a representation of dependencies between external influences and the output of a CPS. Apart from application-motivated categories, we consider regression tree learning and clustering to achieve meaningful categories from possibly continuous signals. In an experimental case study, we observe that best classification accuracy is achieved with clustering if an elbow criterion is used to determine the number of clusters. The learned decision tree offers an automated strategy to determine relevant external influences for systems. Furthermore, we discuss further applications for the decision tree, e.g., to predict or explain the system's behavior or to provide information on possible improvements to a system's setup. Those applications include next steps to extend the decision tree to a model of the system.

Acknowledgement

This work is supported by the TUHH I^3 Projects funding. Special thanks go to the 3D_Log project team.

References

[1] F. H. Bahnsen and G. Fey, "Local monitoring of embedded applications and devices using artificial neural networks," in *Euromicro Conference on Digital System Design (DSD)*, 2019.

[2] M. Vierhauser, J. Cleland-Huang, S. Bayley, T. Krismayer, R. Rabiser, and P. Grünbacher, "Monitoring cps at runtime - a case study in the uav domain," in *Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, 2018.

[3] P. M. Kruse and J. Wegener, "Test sequence generation from classification trees," in *International Conference on Software Testing, Verification and Validation*, 2012.

[4] M. Chen, H. Haggag, and A. Orailoglu, "Decision tree based mismatch diagnosis in analog circuits," in *VLSI Test Symp. (VTS)*, 2006.

[5] S. R. Gaddam, V. V. Phoha, and K. S. Balagani, "K-Means+ID3: A novel method for supervised anomaly detection by cascading K-means clustering and ID3 decision tree learning methods," *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, 2007.

[6] P. Beinschob and C. Reinke, "Advances in 3d data acquisition, mapping and localization in modern large-scale warehouses," in *International Conference on Intelligent Computer Communication and Processing (ICCP)*, 2014.

[7] M. Beul, D. Droeschel, M. Nieuwenhuisen, J. Quenzel, S. Houben, and S. Behnke, "Fast autonomous flight in warehouses for inventory applications," *IEEE Robotics and Automation Letters*, vol. 3, pp. 3121–3128, 2018.

[8] R. Yasaei, F. Hernandez, and M. A. Al Faruque, "IoT-CAD: Context-aware adaptive anomaly detection in IoT systems through sensor association," in *Int'l Conf. on CAD*, 2020, pp. 1–9.

[9] A. Salamati, S. Soudjani, and M. Zamani, "Data-driven verification under signal temporal logic constraints," *ArXiv*, vol. abs/2005.05040, 2020.

[10] A. Antoulas, D. Sorensen, and S. Gugercin, "A survey of model reduction methods for large systems," *Contemp. Math*, vol. 280, 2006.

[11] M. Merten, "Active automata learning for real life applications," Ph.D. dissertation, Technische Universität Dortmund, 2013.

[12] H. Urbat and L. Schröder, "Automata learning: An algebraic approach," in *ACM/IEEE Symposium on Logic in Computer Science (LICS)*, 2020.

[13] J.-F. Raskin, *An Introduction to Hybrid Automata*. Boston, MA: Birkhäuser Boston, 2005.

[14] R. A. Reyment and K. G. Jvreskog, *Applied Factor Analysis in the Natural Sciences*. Cambridge University Press, 1993, ch. Aims, Ideas, and Models of Factor Analysis, p. 71–88.

[15] Y. Amemiya and I. Yalcin, "Nonlinear Factor Analysis as a Statistical Method," *Statistical Science*, vol. 16, pp. 275 – 294, 2001.

[16] D. J. Ketchen and C. L. Shook, "The application of cluster analysis in strategic management research: An analysis and critique," *Strategic Management Journal*, vol. 17, pp. 441–458, 1996.

[17] M. Moshkov, *Comparative Analysis of Deterministic and Nondeterministic Decision Trees*. Cham: Springer International Publishing, 2020.

[18] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Transactions on Robotics*, vol. 32, pp. 1309–1332, 2016.

[19] C. Hansen, D. Gibas, J.-L. Honeine, N. Rezzoug, P. Gorce, and B. Isableu, "An inexpensive solution for motion analysis," *Proceedings of the Institution of Mechanical Engineers, Part P: Journal of Sports Engineering and Technology*, vol. 228, 2014.

[20] S. Umeyama, "Least-squares estimation of transformation parameters between two point patterns," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, pp. 376–380, 1991.

[21] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.