Unsupervised Mode Detection in Cyber-Physical Systems using Variable Order Markov Models

Barış Gün Sürmeli, Feyza Eksen, Bilal Dinç, Peter Schüller, Borahan Tümer
Faculty of Engineering, Marmara University
İstanbul, Turkey

***Abstract*—Sequential data generated from various sources in a multi-mode industrial production system provides valuable information on the current mode of the system and enables one to build a model for each individual operating mode. Using these models in a multi-mode system, one may distinguish modes of the system and, furthermore, detect whether the current mode is a (normal or faulty) mode known from historical data, or a new mode. In this work, we model each individual mode by a probabilistic suffix tree (PST) used to implement variable order Markov models (VOMMs) and propose a novel unsupervised PST matching algorithm that compares the tree models by a matching cost once they are constructed. The matching cost we define comprises of a subsequence dissimilarity cost and a probability cost. Our tree matching method enables to compare two PSTs in linear time by one concurrent top-down pass. We use this matching cost as a similarity measure for k-medoid clustering and cluster PSTs obtained from system modes according to their matching costs. The overall approach yields promising results for unsupervised identification of modes on data obtained from of a physical factory demonstrator. Notably we can distinguish modes on two levels of granularity, both corresponding to human expert labels, with a RAND score of up to 73 % compared to a baseline of at most 42 %.**

I. Introduction

A production line in an industrial plant operates in multiple modes, interpretable as normal (regular production), break (switching to the production of another product type), or anomalous (the operation deviates from its normal flow, manifesting a potential problem). Data generated by various sources such as sensors planted in the system under analysis provide valuable information on the current state of the system. Both normal and anomalous modes are characterized by significant structures (i.e., sequences of states) that show up in some specific sequential and/or cyclic order as a symptom of the specific nature of the relevant mode. The occurrence and/or order of these structures reveal the temporal dependence of states of the system which form a basis for the relevant mode. As an example, consider a car plant that produces automobiles with automatic versus manual gear shift, and convertible versus standard roof, and combinations thereof.

With the rise of the smart industrial plant systems, self-diagnosis tasks such as anomaly detection and root cause analysis became new challenges. As the prerequisite, the operating modes of the system have to be detected and distinguished in a robust fashion. Maintaining manual models for this task is cumbersome and expensive, therefore we here present an unsupervised approach for distinguishing operating modes based on data streams of an industrial system.

To that end we use Markov models which are a standard tool for characterizing system behaviors. Standard Markov models of fixed order either overgeneralize or overfit if the system shows behavior that does not truly correspond with the chosen order of the model. To overcome this issue, we use Variable order Markov models (VOMMs) that can represent temporal dependence of *variably long structures* and can be learned without any initial knowledge of the system required. VOMMs are mainly used in biological signal processing, where state-of-the-art methods train a VOMM and then ‘run’ it on a test sequence to classify if the sequence matches the VOMM. In this work we follow a different approach: we train multiple VOMMs and compairing their internal representation, which is a Probability Suffix Tree (PST).

We propose a novel *unsupervised PST matching algorithm* that, using a matching cost calculation, compares the tree models of the sequences once they are constructed. Matching cost is composed of two components: *dissimilarity cost* and *probability cost*. They reflect the extent of dissimilarity, in respective order, of the subsequences of nodes matched and that of the probability vector attached to the nodes matched among the two PSTs. Existing tree matching algorithms like Flexible Tree Matching [1] are generic but intractable. We make use of inherent structure and meaning of PSTs and propose a novel tree matching scheme that compares two PSTs *in linear time* by one concurrent top-down pass.

We perform experimental results on data produced by a physical factory demonstrator, and evaluate our method in comparison with a semi-naive baseline. We apply k-medoids clustering [2] to cluster PSTs with respect to their distances (matching costs). Results show that our unsupervised method is able to identify system modes on two levels of granularity, corresponding to two levels of mode labels produced by human experts. Score for mode identification is significantly above a baseline method.

In Section II, we summarize VOMMs and PSTs, Related work is discussed in Section III. In Section IV we discuss how PST models are learned, how matching cost between pairs of PSTs are calculated, and how we perform clustering using this distance measure. Section V presents our experiments and discuss the results. We conclude the paper in Section VI.

II. Preliminaries

A Markov Model is used to model stochastic processes where future states are assumed to depend only on the current state of the system. This definition is extended to define n^{th}-*order* Markov models where the future states are assumed to depend on n previous states from the current state [3].

A. Variable Order Markov Model (VOMM)

An extension of Markov models are Variable Order Markov Models where the conditioning order may vary from state to state [4]. This variation depends on the significance of the information extracted from the observations. As long as an observation occurs in the data significantly many times the information about it is more probable to be kept in the model. With this property, VOMMs are realistic models that can represent natural processes, such as the behavior of a production plant. Their concise and adaptive representation, ability to work with no prior knowledge about the system and considerably small amount of memory requirement compared to models such as HMMs are another appealing properties of VOMMs.

B. Probabilistic Suffix Tree (PST)

One convenient way to implement VOMMs are PSTs [5, 6], which are based on Suffix Trees (STs). Fig. 1 (left) shows an example of a ST. STs contain all suffixes of a given input sequence where traversing from root to each leaf yields a specific suffix. A non-leaf node in a ST exists which contains a sub-sequence only and only if two different possible characters follows that sub-sequence at least once in the input sequence.

Probabilistic Suffix Trees are a probabilistic abstraction of STs, where nodes of the ST are pruned with respect to two parameters: the minimum number t of occurrences of the subsequence of the node within the sequence, and the maximum length L of any subsequence of a node allowed in the tree. Intuitively, t prunes suffixes with fewer witnesses from the tree, and L limits the level of detail represented in the tree. In addition to representing suffixes, PSTs represent the structure and importance of represented suffixes by storing probability information for each node: the probability vector at a node contains the probabilities of occurrence of each character after the subsequence that the node represents. Fig. 1 (right) shows an example PST and the sequence it represents.

III. Related Work

Techniques for clustering and anomaly detection, in particular methods using Markovian techniques and kernel-based methods which employ pairwise dissimilarities, are surveyed in [7]. Performance benefits of VOMMs over Hidden Markov Models with same predictive power were shown in [8]. VOMMs have been extensively used for classification and prediction [8, 9, 10]. A common implementation technique are Probabilistic Suffix Trees (PSTs) [5, 6] which can be constructed in linear time [11, 12]. PSTs have been used in various applications such as protein prediction and classification [8, 9, 11, 12] and outlier detection [13].

Unsupervised methods in industrial signal processing employed Probabilistic Automata [14, 15] and used PSTs for anomaly detection [16], however VOMMs and PSTs are mainly used in bioinformatics. State-of-the-art methods for classification of input streams using PSTs simulate a run of the test sequences on PST models learned in a training phase to decide which class the sequences belong to. A method that directly works on (non-probabilistic) suffix trees by representing them as a vector space document model, and comparing them for clustering, was introduced in [17].

An alternative method to the tree matching we introduce here is Flexible Tree Matching (FTM), which is a generic method for matching trees while taking into account similarities of the nodes in different places [1]. FTM is intractable (NP-complete), moreover it is not beneficial for our goal of comparing PSTs, because it matches trees on various node depths. For comparing PSTs, there are hard constraints which comparisons are meaningful (i.e., we compare only nodes representing a sub- or supersequence of the node in the other tree). Also, PST child nodes are sorted wrt. their represented subsequence, which is a property not used by FTM.

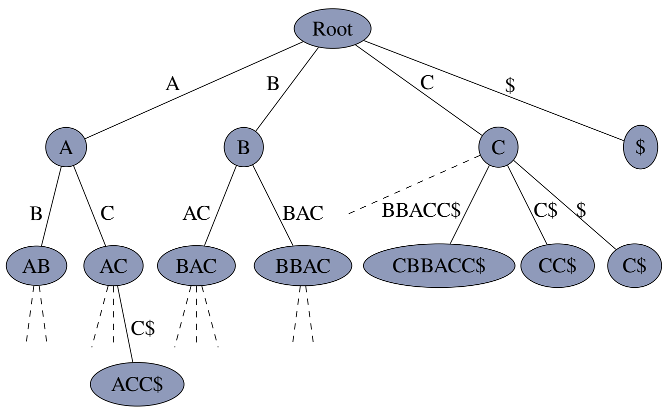
IV. PST Learning, Matching, & Clustering

We next describe our approach for unsupervised mode detection with VOMMs. Section IV-A describes how we first construct PSTs for distinct sections of industrial system data, Section IV-B provides our novel method for computing matching cost between pairs of PSTs, and Section IV-C describes how we use this matching cost to cluster the data sequences and to detect equal and similar modes of the system.

To reduce the scope of this task, we assume data streams are already split into relevant sections of interest. We call these sections 'regimes'. Regime detection can be done by supervised or by unsupervised methods, for example using changepoint detection [18].

A. PST Learning

PSTs are constructed with the adaptive PST construction algorithm proposed in [12], which is one of the most efficient PST construction algorithms, takes linear time with respect to the length of the sequence, makes use of Context Trees [6], and the WOTD Lazy Suffix Tree construction algorithm [19]. Different from [12] we here perform only Support Pruning and do not use Similarity Pruning, since the former is sufficient for obtaining reasonably small models and because using only one kind of pruning keeps the approach more simple. The PST construction algorithm in [12] adds auxiliary nodes and Reverse Suffix Links [11] to the PST to be able to run the VOMM on new input sequences, moreover probability values are smoothened to prevent the model to assign zero probabilities to input sequences. For the purpose of our work, we exclude all these components because we do not run the tree model on new data sequences to verify how well they match. Instead, our method directly operates on PSTs and considers missing nodes in the distance measure that is defined in the following.



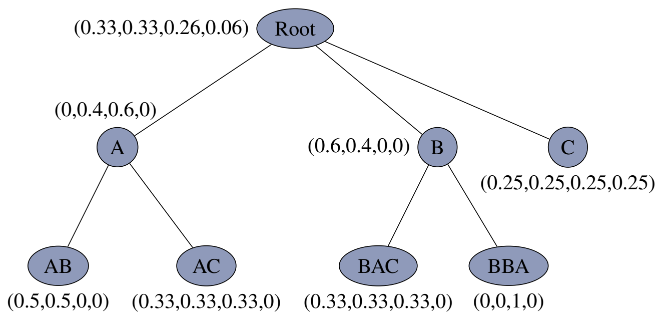


Fig. 1: Example Suffix Tree (left) and Probabilistic Suffix Tree (PST) (right) of the sequence 'ABACABBACBBACC$'. PST is created with pruning parameters $t = 2$ and $L = 3$. Probability vectors represent next symbol probabilities for the sequence of symbols $(A, B, C, \$)$ where \$ indicates end-of-sequence.

B. PST Matching

We next formalize our distance measure for comparing two PSTs. We combine two kinds of matching cost:

(1) Probability Cost: The effect of differences between the probability information of the matched nodes which has the same subsequence, and

(2) Dissimilarity Cost: The effect of dissimilarity between the subsequences of the matched nodes.

Given trees T_1 and T_2 which are models of the sequences of A and B, respectively, we define the set $\{A_1, A_2, \ldots, A_N\}$ of subsequences appearing in nodes of T_1, and similarly the set $\{B_1, B_2, \ldots, B_M\}$ of subsequences in nodes of T_2. Then the distance measure (matching cost) is defined as:

$$C_{T_1,T_2} = \sum_{i}^{N} \sum_{j}^{M} x_{ij} \omega_{ij} (d_{ij} I / L_{ij} + (1 - I) \delta_{ij} \epsilon_{ij} / 2). \quad (1)$$

This matching cost is calculated over all pairwise matchings of substrings of both trees.

Components of the above formula are as follows.

[x_{ij}] Activator Component $x_{ij} \in \{0, 1\}$ is 1 only if A_i is the closest node to B_j in the other tree with respect to length of their subsequence, where the shorter node is a prefix of the longer one.

[w_{ij}] Match Weighting Component w_{ij} is the average of occurrence probabilities of the subsequences that the nodes represent:

$$w_{ij} = (P(A_i) + P(B_j))/2. \quad (2)$$

Each match among the nodes of two trees is multiplied by w_{ij} with the aim of weighting the contribution of the cost proportional to the importance of the corresponding nodes.

[I] Cost Type Weighting Component I, $0 \leq I \leq 1$, allows for scaling between two types of matching costs: Dissimilarity & Probability (see below 'Raw Cost'). The closer I is to 1, the higher the contribution of dissimilarity cost to the total cost.

[d_{ij}] Length Difference Component d_{ij} between the context of nodes i and j is defined as follows:

$$d_{ij} = abs(\mid A_i \mid - \mid B_i \mid) \quad (3)$$

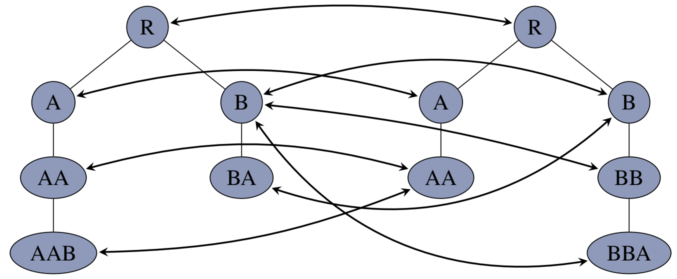


Fig. 2: PST Matching Example: All matchings which have non-zero probability of doing non-zero contribution to the total matching cost are shown.

where $|X|$ indicates the length of the sequence X.

[δ_{ij}] Probability Vector Distance Component δ_{ij} is the sum of the absolute values of the differences for each dimension in the probability vector of two nodes given the alphabet size S for the modeled sequences:

$$\delta_{ij} = \sum_{k}^{S} \mid (\overrightarrow{P_{A_i}})_k - (\overrightarrow{P_{B_j}})_k \mid \quad (4)$$

[ϵ_{ij}] Cost Type Component $\epsilon_{ij} \in \{0, 1\}$ is 1 only if $A_i = B_j$. When 0, probability vector distance cannot contribute to cost because the compared nodes are not identical, i.e., a comparison would be meaningless.

[L_{ij} resp. $/2$] Normalization. To normalize both cost types to stay within $[0, 1]$, dissimilarity cost is divided by the length of the longer subsequence $L_{ij} = \max(|A_i|, |B_j|)$, while the probability cost is divided by 2.

Raw Cost: The factor $(d_{ij} I / L_{ij} + (1 - I) \epsilon_{ij} \delta_{ij} / 2)$, in (1) is called raw (unweighted) cost, which consists of dissimilarity cost $d_{ij} I / L_{ij}$ and probability cost $(1 - I) \delta_{ij} \epsilon_{ij} / 2$.

By using structural properties of PSTs we can compute C_{T_1,T_2} in a single parallel traversal over T_1 and T_2 which is linear in time $N + M$.

C. PST Clustering

Using the PST matching cost C_{T_1,T_2} formulated above, each pair of tree models is compared, a dissimilarity matrix is constructed with the calculated cost values. Based on this dissimilarity matrix, we apply k-medoids clustering to the PST models to classify the behavior of the system in each regime using Partitioning Around Medoids algorithm [2] which ensures optimal selection of medoids. K-medoids chooses data points as centers (medoids or exemplars) and minimizes an arbitrary metrics of distances between these centers and points assigned to clusters. A medoid can be defined as the object of a cluster whose average dissimilarity to all the objects in the cluster is minimal, i.e., it is a most centrally located point in the cluster.

As a result of clustering, we obtain information which PSTs, i.e., which regimes of the system data stream, belong to the same cluster. Regimes that are assigned the same cluster are predicted to originate from the same system mode.

Identifying similar or novel system modes in such an unsupervised way is useful for detecting and classifying system problems or system reconfigurations without the need for building a model of the system.

V. Experimental Evaluation

A. Lego Demonstrator Data

To evaluate our tree matching algorithm we use data generated by a model factory production line created from Lego.

The Lego Demonstrator is shown in Fig. 3. A Lego piece is carried from its initial position in the magazine to the conveyor belt and placed on one of the two sticks. During this process, aligning the piece correctly is an important concern. To achieve this, the piece undergoes several mechanical manipulation steps using a pushing device and a robotic arm. If one of the steps is not performed successfully the product to be constructed is considered faulty.

One *run* of the demonstrator uses six input lego pieces to produces two products on the two sticks on the conveyor belt, by stacking three of the incoming pieces to the first stick and the other three incoming pieces to the second stick on the conveyor belt. Input pieces are processed sequentially. To simulate product variation in the system, the order of sticks that are used for placing the input piece is varied. For example, if we name one of the sticks as 1 and the other one as 2, one run may have the stick sequence 1→1→2→1→2→2 to place all six pieces.

This setup produces a sensor and actuator data sequence similar to a real industrial plant. Concretely we obtain a log consisting of voltage and current values for two control units (Lego Bricks), sensor output one touch sensor used for stick alignment, moreover motor information (speed, angle, and motor command) for five motors. Overall this yields 20 signals that are logged each 250 ms: 2 real, 12 integer, and 6 binary signals. The data used in experiments in this paper was obtained from 40 runs. Each run moves 6 Lego pieces and produces 2 products. In total our dataset contains a data

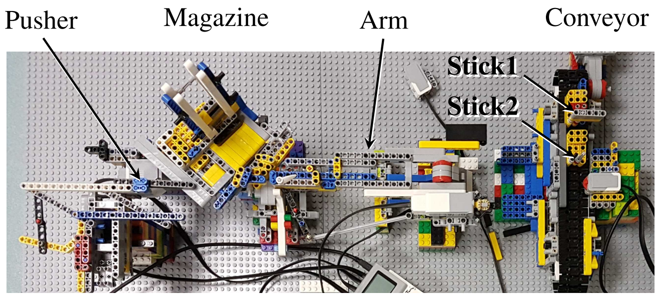


Fig. 3: Lego Demonstrator Photo

sequence of 37341 vectors of size 20 including a timestamp. 8 runs showed anomalous behavior which occurred naturally during physical simulation, i.e., without external provocation.

Labels. To facilitate verification methods in machine learning and potentially supervised methods, we annotate several aspects of the data obtained from the demonstrator. We log production sequences (i.e., the order of sticks). For each run we create a manual label about the quality of the product, i.e., if there was a fault. We also classify that fault according to its root cause, for example 'failed while placing on stick' or 'piece stuck in magazine'. In our dataset, we use two distinct product types where the underlying sequence is 1→1→1→2→2→2 and 2→2→2→1→1→1, respectively, and we performed 20 runs for each product type. One type of anomaly occurred 6 times for the first product type, and another type of anomaly occurred once for each product type. Moreover we produce labels about internal states of the demonstrator control program in two levels of granularity: a high-level label that distinguishes 5 states, and a low-level label with 12 distinct states.

B. Preprocessing

Since the sampling rate in the system is high compared with the rate of change in the system, we perform downsampling on the data and use only each fifth data sample, i.e., we process data as if it was measured all 1.25 sec.

VOMMs require a sequence over a discrete alphabet as input, therefore we perform discretization of the data sequence: first we perform dimensionality reduction with Principal Component Analysis (PCA) and then Hierarchical Clustering [20] which showed consistent results for different runs with the same instances in preliminary experiments.

C. Baseline

We have implemented a simple baseline metric for defining a distance between two regimes. Each regime is represented by a vector where the frequency information for each character is stored. The euclidean distance between these vectors is calculated to retrieve the dissimilarity matrix that contains the distances between each vector. This distance matrix has the same shape as the result of PST Matching and we compare clustering using PST Matching with clustering based on the baseline distance metric.

TABLE I: Results of experimental evaluation for parameters explained in V-D. Adjusted RAND index is used for scoring clusters created with a distance measure based on PST Matching and clusters created with a generic distance baseline (V-C).

					PST Matching Method																				Baseline
				$t =$	1					2					3					4					–
mk	label set	d	k	$L =$	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	–
2	P	0,75	4		1,00	0,08	1,00	1,00	1,00	0,08	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	-0,02	1,00
			11		0,48	0,19	0,03	0,03	0,03	0,48	0,08	0,08	0,48	0,48	0,48	0,08	0,48	0,48	0,48	0,48	0,08	0,48	0,48	0,48	0,48
		0,8	4		0,63	0,08	1,00	1,00	1,00	0,08	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	0,18	0,34
			11		0,08	0,41	0,35	0,02	0,05	0,08	0,08	0,08	0,08	0,08	0,08	0,08	0,08	0,08	0,08	0,08	0,08	0,08	0,08	0,08	0,08
		0,85	4		0,41	0,08	0,15	1,00	0,90	0,08	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	0,18	0,41
			11		0,08	0,24	0,02	0,02	0,02	0,55	0,48	0,48	0,48	0,48	1,00	0,48	0,48	0,48	0,48	1,00	0,48	0,48	0,48	0,48	1,00
	Q	0,75	4		0,06	0,32	0,06	0,06	0,06	0,32	0,06	0,06	0,06	0,06	0,06	0,06	0,06	0,06	0,06	0,06	0,06	0,06	0,06	-0,05	0,06
			11		0,09	0,17	0,13	0,13	0,13	0,09	0,05	0,05	0,09	0,09	0,09	0,05	0,09	0,09	0,09	0,09	0,05	0,09	0,09	0,09	0,09
		0,8	4		0,01	0,32	0,06	0,06	0,06	0,32	0,06	0,06	0,06	0,06	0,06	0,06	0,06	0,06	0,06	0,06	0,06	0,06	0,06	0,01	0,03
			11		0,05	0,12	0,07	0,18	0,09	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05
		0,85	4		-0,02	0,32	**0,35**	0,06	0,04	0,32	0,06	0,06	0,06	0,06	0,06	0,06	0,06	0,06	0,06	0,06	0,06	0,06	0,06	0,01	-0,02
			11		0,05	0,13	0,18	0,18	0,18	0,06	0,09	0,09	0,09	0,09	0,06	0,09	0,09	0,09	0,09	0,06	0,09	0,09	0,09	0,09	0,06
4	P+Q	0,75	4		0,48	0,43	0,44	0,64	0,61	0,50	0,44	0,44	0,59	0,43	0,47	0,40	0,43	0,57	0,30	0,47	0,41	0,41	0,36	0,24	0,41
			11		0,60	0,58	0,42	0,32	0,06	0,60	0,59	0,36	0,37	0,38	0,59	0,57	0,64	0,36	0,59	0,59	0,57	0,55	0,36	0,59	0,36
		0,8	4		0,40	0,19	0,61	0,61	0,61	0,55	0,44	0,61	0,58	0,42	0,40	0,40	0,40	0,58	0,37	0,40	0,41	0,40	0,37	0,24	0,36
			11		0,60	0,39	0,24	0,10	0,07	0,60	0,61	0,59	0,59	0,60	0,38	0,43	0,41	0,58	0,58	0,38	0,43	0,45	0,59	0,59	0,37
		0,85	4		0,32	0,19	0,42	0,37	0,37	0,39	0,44	**0,66**	0,40	0,37	0,36	0,37	0,41	0,40	0,36	0,36	0,37	0,41	0,36	0,10	0,38
			11		0,47	0,23	0,07	0,08	0,08	0,61	0,36	0,44	0,41	0,38	0,37	0,39	0,38	0,44	0,41	0,59	0,39	0,41	0,57	0,36	0,36
	P+F	0,75	4		0,55	0,48	0,48	0,70	0,67	0,56	0,48	0,48	0,67	0,48	0,52	0,44	0,48	0,64	0,35	0,52	0,45	0,45	0,40	0,27	0,39
			11		0,69	0,66	0,35	0,25	0,07	0,69	0,67	0,40	0,41	0,44	0,67	0,66	0,63	0,40	0,66	0,67	0,66	0,63	0,41	0,66	0,42
		0,8	4		0,42	0,19	0,67	0,67	0,67	0,51	0,48	0,67	0,64	0,47	0,42	0,44	0,44	0,64	0,41	0,42	0,45	0,44	0,41	0,28	0,34
			11		0,69	0,46	0,23	0,07	0,06	0,69	0,69	0,68	0,68	0,68	0,41	0,50	0,43	0,67	0,67	0,41	0,50	0,52	0,66	0,67	0,41
		0,85	4		0,30	0,19	0,46	0,40	0,40	0,41	0,48	**0,73**	0,43	0,41	0,38	0,41	0,45	0,42	0,38	0,38	0,41	0,45	0,40	0,04	0,35
			11		0,45	0,21	0,06	0,07	0,07	0,67	0,41	0,46	0,43	0,41	0,41	0,46	0,41	0,46	0,43	0,68	0,46	0,43	0,57	0,40	0,40

D. Setup

To configure preprocessing, pruning of PST learning, and PST matching and clustering, we use the following parameters.

d	The number of dimensions of PCA is adjusted such that the resulting components explain a percentage of the variance of the original data that is closest to d.
k	Number of classes for Hierarchical Clustering.
t	Minimum support parameter, see II-B.
L	Maximum subsequence length parameter, see also II-B.
I	Dissimilarity and probability cost ratio, see IV-B.
mk	Parameter k for k-medoids clustering, see IV-C.

Evaluations are done by comparing result cluster solutions with four different gold label sets using Adjusted RAND index. Gold labels contain product type P, product quality Q, and type of failure F. For two-cluster solutions ($mk = 2$) we evaluate against the following gold labels:

P: the sequence used to produce the product (2 classes),

Q: the quality of the product (2 classes: good or faulty).

For $mk > 2$ we evaluate against the following gold labels:

P+Q: product type and quality (4 classes),

P+F: product type and failure type (5 classes).

Note that one of the two failure types only occurred with one product type, so P+F has 5 classes, not 6.

This way we measure to which extent our method separates different types of behavior, where behavior includes regular and failure behavior.

We perform experimental runs for all combinations of the following parameter values: $d \in \{0.75, 0.80, 0.85, 0.90, 0.95\}$, $k \in \{4, 5, 6, 7, 8, 9, 10, 11, 12\}$, $t \in \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$, $L \in \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$, $I \in \{0.3, 0.4, 0.5, 0.6, 0.7\}$, and $mk \in \{2, 4, 5, 6, 7\}$.

E. Results and Discussion

Table I shows RAND index values for those parts of the configuration space that show promising and interesting results. In particular we show results only for parameters $d \leq 0.85$, $k \in \{4, 11\}$ $t \leq 4$, $L \leq 5$, and $I = 0.5$, because other values yield worse results.

P and Q. For these gold cluster labels which contain two classes, best results were naturally obtained with $mk = 2$ and we omit other results. With alphabet size $k = 4$, PST Matching produces a perfect clustering for product types (label set P). The baseline also achieves perfect clustering for two of six preprocessing configurations. Both our method and the baseline fail in distinguishing normal from anomalous production cycles (label set Q), which can be explained by large data sequence variations due to product configuration and small variations due to failures.

P+Q and P+F. For these gold cluster labels which contain four and five classes, respectively (see V-D), best results were obtained with $mk = 4$. Best results are here obtained with a combination of $d = 0.8$ and $k = 11$, which are both preprocessing parameters. Here, a larger symbol alphabet seems to be beneficial for mode detection (as opposed to label sets P and Q). The baseline has a significantly worse performance of at most 0.41 and 0.42 RAND score, compared with our PST Matching distance metric that achieves a score of 0.66 and 0.73 for label sets P+Q and P+F, respectively.

Discussion. About parameters of our experimental setup we can say the following. Parameter d permits a certain amount of noise suppression already while discretizing the input (fewer dimensions mean less noise), L adjusts the level of detail of sequences that are added to the PST (higher L means more detail, potentially overfitting), and t prunes away rarely seen sequences in the PST (higher t means more pruning). Regarding pruning, we see results consistent with [4] where it is shown that $t = 2$ makes PSTs a consistent estimator and is the recommended setting for pruning: $t = 1$ does not prune away noise and maintains all subsequences with only a single witness in the data, while $t = 3$ may cause loss of significant information. Alphabet size k provides best results with values 4 and 11. These numbers are related in an interesting way with the numbers of states created with manual labels within the state machine of the demonstrator (5 coarse-grained and 12 fine-grained states, respectively).

Clustering gives the best results around $I = 0.5$ which balances equally between probability and dissimilarity cost.

Although label set P+F has 5 classes, clustering with $mk = 4$ yields best results, which is probably due to imbalance in the dataset (many good runs, few failed runs).

The baseline is clearly outperformed in all but the simplest case where just the type of product needs to be distinguished (label set P). This shows the capacity of our method to capture complex behavior.

VI. Conclusion

We introduced a method for unsupervised clustering based on PSTs that have been learned from system output. The method identifies system modes significantly better than a semi-naive baseline.

The number of target clusters mk for our method should be close to or a bit below the number of expected distinct modes of the system. The same is true for the number of alphabet symbols k that is produced during discretization. Several levels of abstraction provide high accuracy if the number of clusters is (close to) correct: in our experiments we can predict a binary signal very well with $mk = 2$, moreover we can predict 5 classes of product type and failure type with good RAND index using $mk = 4$ and $k \in \{4, 11\}$. These parameters are related to the properties of the physical system from which we obtained this data: the system has 5 or 12 internal states, depending on level of granularity of annotation.

Regarding other parameters, $I = 0.5$, $d = 0.8$, $t = 2$, and $L = 3$ are parameter values that promise good results. In practice, these values need to be adjusted experimentally.

In summary we conclude that unsupervised clustering for mode detection is bound to make mistakes, but with our method of representing system outputs in an abstract way using PSTs it has a much better chance of distinguishing true modes of the system, and experiments show that this works on multiple levels of granularity.

Acknowledgement

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 678867.

References

[1] R Kumar, J O Talton, S Ahmad, T Roughgarden, and S R Klemmer. Flexible tree matching. In *Proc. IJCAI*, pages 2674–2679, 2011.

[2] L Kaufman and P J Rousseeuw. Partitioning around medoids. In *Finding groups in data: an introduction to cluster analysis*. Wiley Online Library, 1990.

[3] E Alpaydin. *Introduction to Machine Learning*. MIT Press, 2014.

[4] P Bühlmann, A J Wyner, et al. Variable length markov chains. *The Annals of Statistics*, 27(2):480–513, 1999.

[5] D Ron, Y Singer, and N Tishby. Learning probabilistic automata with variable memory length. In *Proc. Computational Learning Theory*, pages 35–46, 1994.

[6] J Rissanen. A universal data compression system. *IEEE Transactions on Information Theory*, 29(5):656–664, 1983.

[7] V Chandola, A Banerjee, and V Kumar. Anomaly detection for discrete sequences: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 24(5):823–839, 2012.

[8] G Bejerano and G Yona. Variations on probabilistic suffix trees: statistical modeling and prediction of protein families. *Bioinformatics*, 17(1):23–43, 2001.

[9] R Begleiter, R El-Yaniv, and G Yona. On prediction using variable order markov models. *Journal of Artificial Intelligence Research*, 22:385–421, 2004.

[10] H Oğul and E Ü Mumcuoğlu. SVM-based detection of distant protein structural relationships using pairwise probabilistic suffix trees. *Computational Biology and Chemistry*, 30(4):292–299, 2006.

[11] A Apostolico and G Bejerano. Optimal amnesic probabilistic automata or how to learn and classify proteins in linear time and space. *Journal of Computational Biology*, 7(3-4):381–393, 2000.

[12] M H Schulz, D Weese, T Rausch, A Döring, K Reinert, and M Vingron. Fast and adaptive variable order markov chain construction. In *International Workshop on Algorithms in Bioinformatics*, pages 306–317, 2008.

[13] P Sun, S Chawla, and B Arunasalam. Mining for outliers in sequential databases. In *Proc. SIAM International Conference on Data Mining*, pages 94–105, 2006.

[14] O Niggemann, B Stein, A Vodencarevic, A Maier, and H K Büning. Learning behavior models for hybrid timed systems. In *AAAI*, pages 1083–1090, 2012.

[15] O Niggemann, A Vodencarevic, A Maier, S Windmann, and H K Büning. A learning anomaly detection algorithm for hybrid manufacturing systems. In *International Workshop on Principles of Diagnosis*, 2013.

[16] M Yoon and G F Ciocarlie. Communication pattern monitoring: Improving the utility of anomaly detection for industrial control systems. In *NDSS Workshop on Security of Emerging Networking Technologies*, 2014.

[17] H Chim and X Deng. A new suffix tree similarity measure for document clustering. In *Proc. WWW*, pages 121–130, 2007.

[18] S Liu, M Yamada, N Collier, and M Sugiyama. Change-Point Detection in Time-Series Data by Direct Density-Ratio Estimation. *Neural Networks*, 43:72–83, 2013.

[19] R Giegerich, S Kurtz, and J Stoye. Efficient implementation of lazy suffix trees. *Software: Practice and Experience*, 33(11):1035–1049, 2003.

[20] L Rokach and O Maimon. Clustering methods. In *Data Mining and Knowledge Discovery Handbook*. Springer, 2005.