# Learning Patterns for Complex Event Detection in Robot Sensor Data

Bernhard G. Humm(✉) and Marco Hutter

Hochschule Darmstadt - University of Applied Sciences, Darmstadt, Germany
{bernhard.humm,marco.hutter}@h-da.de

**Abstract.** We present an approach for learning patterns for Complex Event Processing (CEP) in robot sensor data. While the robot executes a certain task, sensor data is recorded. The sensor data recordings are classified in terms of events or outcomes that characterize the task. These classified recordings are then used to learn simple rules that describe the events using a simple, domain specific language, in a human-readable and interpretable way.

**Keywords:** Robots · Time series · Machine learning · Complex event detection · Interpretable AI

## 1 Introduction

Complex Event Processing (CEP [2]) has proven to be an effective and versatile way to and derive conclusions from low-level data streams. Low-level information about events from different data sources is integrated and combined in order to generate high-level events that carry a domain-specific meaning – for example, the appearance of a certain sequence of low-level events within a certain time frame.

The rules for determining whether a high-level event should be emitted can be complex. The interdependencies between different sources and types of low-level events and their relative time characteristics have to be described formally and in a machine-executable form. This can either happen by implementing the rules directly within a general purpose programming language, or based on domain-specific languages for complex event processing or event stream processing. Although these languages allow formulating the queries in a human-readable and machine-executable form, determining the actual structure and contents of the query for a particular use case is still difficult and may require a lot of domain knowledge.

Machine learning approaches can help to automate the process of finding queries that match certain patterns in a stream of events. The basic idea is to use a set of labeled training data consisting of recorded streams of events, and treat the problem of finding a query that correctly classifies the patterns that appear in the data as an optimization problem. We describe an approach that shows how the task of defining a query pattern can be automated.


B. Dorronsoro et al. (Eds.): OLA 2020, CCIS 1173, pp. 138–149, 2020.
https://doi.org/10.1007/978-3-030-41913-4_12

The remainder of this paper is structured as follows: Sect. 2 introduces the use case for which our approach has been applied. In Sect. 3 we review and summarize the related work for complex event processing, stream event processing, and machine learning approaches that have been applied for similar use cases. The problem statement and goals of the approach are described in Sect. 4. Section 5 describes the approach in detail, starting with a high-level conceptual view, and a formalization of the problem. The implementation that our evaluation is based on is described in Sect. 6, and the results of the evaluation are summarized in Sect. 7. In Sect. 8, we conclude with a summary and show future research directions.

## 2 Sample Use Case

An example use case for our approach is to learn patterns that describe error conditions that occur during the automated assembly of electrical components by a robot, as described in [5]. In this scenario, a robot picks an electrical component from a container and mounts it onto a profile rail within a switch cabinet, as shown in Fig. 1.

**Fig. 1.** Automated electrical component assembly by robot

The robot is a sensitive robot that records forces that are exerted during the assembly process, and offers this data in form of numerical time series. Based on this data, it is possible to detect whether the assembly was successful. This information can solely be derived from the force that is recorded in z-direction. Figure 2 shows the time series for a successful and an unsuccessful assembly step.

The successful assembly is indicated by a characteristic curve shape that occurs when the electrical component snaps into the profile rail. Our goal is to learn patterns that describe conditions like this one, based on a simple pattern language, from a small set of recordings of successful and unsuccessful assembly processes.

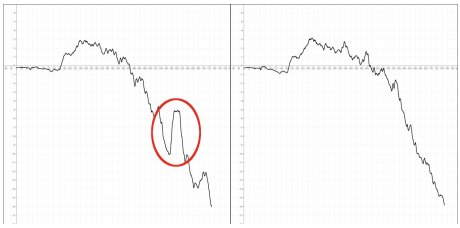

**Fig. 2.** Comparison of force signals at successful assembly (left) and unsuccessful assembly (right)

## 3 Related Work

The field of Complex Event Processing has gained a lot of attention recently, because the concept of deriving higher-level events from streams of low-level events is crucial for decision making in many application areas. For the specific goal of detecting events in time series data, different approaches have been studied. In this section, we present work that is related to our work in terms of goals or the general approach, and point out the differences to our approach in view of the sample use case that we described.

The work by [4] aims at identifying shapes that may appear in temporal data sets. They present a list of shapes that intuitively capture the behavior of a variable over time. For example, they define a "spike" as a sudden increase followed by a sudden decrease of a value, which is exactly the shape that indicates a successful assembly in our use case. The goal here is to explicitly define and search for known shapes, whereas our goal is to automatically find the relevant shapes (or patterns) in the first place, and provide the result in an interpretable and processable form.

The goal of automatically learning CEP rules was also addressed by [8]. They point out the difficulties of implementing algorithms and rules for complex event detection: These rules either have to be implemented in software, or with an Event Description Language, but in both cases, domain experts may not be able to formulate the rules without the help of a software engineer. Therefore, they propose a special kind of Hidden Markov Model that allows learning event rules from a sequence of events where the domain expert does not have to define or describe the relevant event, but only tags the point in time when the relevant event occurred. The results of this learning process are not interpretable, because, as the name suggests, the actual description of the event is Hidden in the Markov Model.

The iCEP framework presented in [6] describes an approach for learning patterns that describe complex events based on primitive events using CEP operators, like selection, aggregation, and windowing. The problem of rule generation is then decomposed into learning different aspects of the rule, where one module is presented for learning each aspect. The element that most closely

resembles our approach is the *constraint learner* that derives inequality relations for elements of the rule.

Another approach for applying data mining techniques to learn CEP rules from labeled input data was presented by [7]. They extract *shapelets* from the time series data, which are then translated into simple CEP rules, and combined into composite CEP rules. The rules are directly output as statements in EPL, an Event Processing Language that can be used in different CEP engines. So the focus was not on generating *interpretable*, but rather directly *executable* rules.

The approach of extracting patterns from multivariate time series using shapelets was also addressed by [3]. They extract shapelets for all patterns that appear in the time series, and identify representative key shapelets from this set. Similar to our approach, they consider the goal of finding patterns as an optimization problem. Even though shapelets are an interpretable basis for rules and patterns in the context of classification, they cannot directly be translated into domain-specific rules that may be used for complex event processing.

## 4 Problem Statement

We specify the goals of our approach by means of requirements.

*(R1) Automated Process:* The process for finding a pattern should be automated. It should require as few human interaction as possible, and as little domain knowledge as possible.

*(R2) Interpretability:* The patterns should be described in a form that can be interpreted, understood, and therefore be validated by humans.

*(R3) Accuracy:* The patterns that are generated by the process should generate an accuracy that is similar to the accuracy that can be achieved with a pattern that is created by a domain expert.

*(R4) Small Training Data Sets:* The task of creating labeled training data is time consuming and involves a lot of effort and domain knowledge. Therefore, the approach should be capable of finding patterns based on small training data sets that consist of few representative instances for all classes.

## 5 Approach

At the highest level of abstraction, the problem of finding a good pattern – i.e. a query that matches the time series according to their labels – is a non-linear optmization problem. The following section gives an overview of the optimization process and the main building blocks that it consists of.

### 5.1 Overview

Figure 3 shows the conceptual view on the approach presented in this paper.

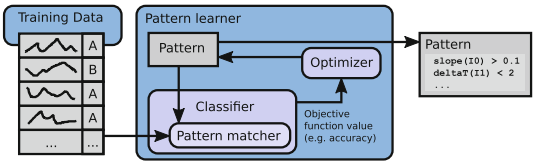


**Fig. 3.** A conceptual overview of the proposed approach: The training data, consisting of labeled time series, is passed to the pattern learner. The pattern learner generates a pattern, applies it to the training data, and computes the classification accuracy. The optimizer modifies the pattern iteratively. The pattern for which the highest classification accuracy has been achieved is returned.

*Training Data:* The training data for our test cases consists of numerical time series instances. The time series are *labeled*, meaning each one is associated with information that indicates whether the assembly of the component was successful or not. A formal definition and different options for preprocessing this time series data for the goal of describing and detecting patterns can be found in [5].

*Pattern:* The representation of the resulting pattern – i.e. the *external* representation – is simply a string that describes the actual query. We use the definition of a pattern based on the pattern language that was presented in [5]. The *internal* representation for the pattern depends on the type of the optimizer. A simple but versatile representation of such a pattern for optimization purposes is that a pattern is stored as a list of constraints, where each constraint involves one of the measures described in [5], the identifier for the segment that it refers to, and the numerical threshold for the inequality.

*Classifier and Pattern Matcher:* The process of matching a pattern against an input time series is described in detail in [5]. Conceptually, the input time series is divided into *segments*, and the constraints that a pattern consists of are checked for the sequence of the most recent segments that have been received. When the pattern matches the current sequence, a high-level event is emitted. The classifier computes the number of true/false positives/negatives for the input data, and determines the value of the objective function for the optimization. There are different possible choices for the objective value that is to be maximized. It can be the overall classification accuracy, the average F1 score, the informedness, or any other measure that can be computed from the confusion matrix of the classification results.

*Optimizer:* The optimizer is the core element of the pattern learner. Its main task is to either generate a new pattern or modify an existing pattern, with the goal of improving the pattern for the training data, according to the objective function. The following Sect. 5.2, will summarize the optimization approaches that we examined.

### 5.2 Optimization Approaches

The problem of optimizing the pattern in order to match the time series according to their labels can be divided into two sub-problems: The first one is the *symbolic* manipulation of the overall structure of the pattern. This refers to the number of constraints that the pattern consists of and the measures and segments that each of the constraints operates on. The second sub-problem is the *numeric* optimization of the thresholds of the constraints of a pattern. These threshold values may be adjusted in order to tighten the classification rules that are given by one pattern.

**Symbolic Optimization with Genetic Algorithms.** The search space for the basic structure of the pattern is large, and the structure cannot be derived systematically from the labeled input data. Therefore, we applied a genetic algorithm approach for searching an initial set of possible patterns, treating the objective value as the "fitness" in the genetic optimization process. In this approach, a *phenotype* is an element of a population during the execution of the genetic algorithm, and basically combines a *genotype* with a fitness value. The *genotype* consists of a single *chromosome*. Each *chromosome* consists of a sequence of *genes* with arbitrary length. Each *gene* has an *allele* that directly encodes one condition that is part of a pattern. Therefore, each chromosome (and thus, each genotype) directly represents a pattern consisting of multiple conditions.

The evolution then consists of generating an initial (random) population, and optimizing the population throughout several generations. The next generation is computed by applying different mutations to the individuals of the current population:

- Multi-point crossover: The sequences of genes from the chromosomes of two parents are split at multiple points, and recombined to generate the offspring.
- Mutator: Randomly replaces a single gene of a chromosome with a new one.
- Dynamic condition chromosome mutator: Randomly adds or removes genes from a chromosome.
- Condition chromosome mutator: Randomly changes the threshold value of a single condition of one gene by a small amount, relative to the value range that was determined for the respective condition.

Each of these mutations is applied with a small probability to the individuals of one generation, in order to generate the offspring. In each generation, the likelihood of individuals to survive for the next generation is proportional to their fitness.

*Symbolic Search Space.* In the most general case, a pattern $P$ as an $m$-ary predicate on segments that is a conjunction of $q$ conditions: $P = \bigwedge_{j=0}^{j<q} C_j$. In order to narrow the search space for the pattern learning, some constraints can be given to the pattern learner. These constraints refer to the number of segments

$m$ that may appear in a pattern, and to the minimum and maximum number of conditions $q$ that a pattern may consist of. For example, it is possible to enforce $m = 3$, to let the pattern learner search for patterns of the form $P(I_0, I_1, I_2)$. Similarly, the number of conditions $q$ can be restricted to a certain interval. For example, one can enforce $2 <= q <= 4$ to make sure that each pattern contains at least two and at most four conditions. These constraints may either be defined by a domain expert, depending on the complexity of the problem and the associated complexity of the pattern, or by the user, who can use these constraints to set an upper limit for the complexity of the pattern.

**Numerical Optimization.** It is possible to turn the optimization task into a purely *numerical* optimization, by assuming the overall structure of a given pattern to be fixed. This means that for a given pattern like

$$P(I_0) \;=\; \text{Slope}(I_0) > x \;\wedge\; \text{Slope}(I_0) < y$$

the thresholds $x$ and $y$ can be considered as the real arguments of a multivariate function, where the function value is the value of the objective function that is applied to the resulting pattern. Given this definition, many standard methods of numerical optmization may be applied. A special case of this approach is to start the numerical optimization with a pattern that involves all possible conditions, and initially defines the thresholds to be the minimum and maximum values of the respective measures.

*Numerical Search Space.* The search space for the numerical optimization can be bounded by the minimum and maximum values that are observed for the respective measure in the training data. For the above example, the values will be bounded by the minimum and maximum value of the slope that has been observed for any segment. If a domain expert decides that segments with larger or smaller slopes should also be considered, the search space can be broadened based on this domain knowledge.

## 6 Implementation

In order to assess the feasibility of the approaches presented in this paper, we implemented the pattern learning algorithm and applied it to various test data sets. The implementation was made in Java. For the application of the generic algorithms, we used the *jenetics* library [9]. The numerical optimization was done with the Apache Commons Math library [1].

## 7 Evaluation

The following sections describe the test setup that we used for our evaluation, and the results referring the requirements specified in Sect. 4.

### 7.1 Test Data

The learning approach has been applied to a corpus of three test data sets. Each data set contains 60 recordings of the sensor outputs of the robot that have been measured during the assembly process. For each data set, there are 40 recordings where the assembly process succeeded and 20 recordings where the assembly process failed. The relevant sensor is the sensor that records the force in z-direction, measured at a 1 ms interval. Details about the preprocessing and actual pattern matching process can be found in [5].

The data sets that we used for our evaluation refer to the same use case: Detecting whether the assembly of an electrical component was successful or not. The data sets differ in the overall length and the absolute values of the time series that they contain. But in all cases, the successful assembly can be detected by a characteristic "spike" in the force in z-direction, as shown in Fig. 2. Our goal is to find a pattern that describes this characteristic generically, in a form that is applicable to all data sets. Therefore, we applied the pattern learner to a data set that was created by combining the initial ones, yielding 180 recordings of 120 positive and 60 negative cases, and present the resulting pattern as well as the accuracy that this pattern achieves for the combined and the individual data sets.

### 7.2 Configuration

The configuration of the genetic algorithm that performs the symbolic optimization was the same in all our experiments: The probability for crossover mutations, general mutations and mutations that add or remove genes was 0.1. The probability for changing the threshold value of a condition of one pattern was also 0.1, with the change of the value being $+/-$ 0.25 times the original value, clamping the result to be in the valid range. We used a population size of 1024 individuals, with 8 generations, stopping the evolution for the case that the objective value remained stable for 4 generations.

Two dimensions of the search space for the symbolic optimization are the number of segments that should appear in a pattern, and the number of conditions that a pattern may consist of. Without any domain knowledge, the pattern learner could be applied without any constraints for these dimensions. But for our experiments, using the knowledge about the curve shape, we concluded that the pattern should involve at most 3 segments - roughly corresponding to the spike that indicated a successful assembly. We also limited the search space for the symbolic optimization, allowing the pattern learner to generate patterns having 1, 2 or 3 conditions.

A dedicated examination of the effect of different constraints or the influence of the parameters (e.g. the population size) on the final result was not part of our research, as the goal was to be able to generate good patterns without a dedicated parameter space exploration. The implementation focusses on flexibility, simplicity and reproducibility of the results. This means that the implementation is not optimized for efficiency. But with the configuration described above, the

search for the patterns described in the following sections took approximately 11 min on a standard desktop PC.

### 7.3 Results

As mentioned in Sect. 5, there are two steps for the optimization: The symbolic optimization that focusses on the structure of a pattern, and the numerical optimization that optimizes the thresholds of a given pattern.

The best patterns with 1, 2 and 3 conditions and their accuracies are shown here:

| Pattern | | | Accuracy |
|---|---|---|---|
| $P_1$ | $= P(I_0, I_1, I_2)$ | $=$ Slope$(I_2) > 0.04755$ | 0.95 |
| $P_2$ | $= P(I_0, I_1, I_2)$ | $=$ Slope$(I_1) > 0.03535 \wedge$ StartV$(I_2) < -0.42348$ | 0.961 |
| $P_3$ | $= P(I_0, I_1, I_2)$ | $=$ Slope$(I_2) > 0.03228 \wedge$ EndV$(I_0) > -20.87390 \wedge$ EndV$(I_1) < -5.17449$ | 0.933 |

Note that a pattern with a higher number of conditions does not necessarily achieve a higher accuracy. The random nature of the genetic algorithm and the larger search space make it possible that a local optimum for the case of 3 conditions achieves a lower accuracy than one for the case of 2 conditions.

The best 25 patterns that have been found by the genetic algorithm have subsequently been passed to a simple numerical optimization which increased or decreased the thresholds of a pattern as long as the resulting accuracy did not decrease. Due to the small size and the training set and the simplicity of the resulting patterns, this simple numerical optimization on the (already optimized) patterns did usually not increase the resulting accuracy, but often tightened the thresholds of the involved conditions. For example, for the best pattern with 2 conditions described above, the threshold for the start value of segment $I_2$ could be decreased from $-0.42348$ to $-1.88572$, yielding the pattern

$$P_2' = P(I_0, I_1, I_2) = \text{Slope}(I_1) > 0.03535 \wedge \text{StartV}(I_2) < -1.88572$$

which still achieves an accuracy of 0.961.

Table 1 summarizes the accuracy of the resulting pattern, individually for the three test data sets, as well as for the combined data set:

### 7.4 Purely Numerical Optimization

As a demonstration of the feasibility and usefulness of the numerical optimization step, we applied the numerical optimization to a pattern that involves all possible conditions for a given number of segments. The general procedure was as

**Table 1.** Classification accuracy of the resulting pattern $P_2'$ for the three data sets and the combined data set

| Data set | A | B | C | Combined |
|---|---|---|---|---|
| Accuracy | 0.967 | 1.000 | 0.900 | 0.961 |

follows: For a given number of segments, we generated a pattern that was a conjunction of all conditions that could be applied to the segments, and the thresholds have been chosen to be the minimum and maximum value that appears for the respective measure. For three segments, five measures, and the possible relations <and>, this yields a pattern that involves 30 conditions, and therefore, 30 thresholds. These thresholds have been used as the real arguments of a multivariate function. We then applied the CMA-ES (Covariance Matrix Adaptation Evolution Strategy) optimizer from the Apache Commons Math library [1] to this function. After the optimization, we removed all conditions from the result pattern that could be removed without decreasing the accuracy. The resulting pattern was

$$\begin{aligned} P_3' = P(I_0, I_1, I_2) \;=\; & \mathrm{DeltaT}(I_1) > 26.11010 \;\wedge \\ & \mathrm{StartV}(I_2) > -2.49862 \;\wedge \\ & \mathrm{DeltaV}(I_2) < -1.26727 \end{aligned}$$

with an accuracy of 0.933.

The main purpose of this experiment was to show that even though the numerical optimization did not improve the accuracy for the simple pattern that was generated by the symbolic optimization for the sample use case, it can still be applied to more complex patterns in order to improve the accuracy of the final result.

### 7.5 Evaluation of Requirements

We evaluate our results referring to the requirements specified in Sect. 4.

*(R1) Automated Process:* The process of finding a pattern is completely automated. It is possible, but not necessary, to integrate domain knowledge in the search process. If nothing is known about the structure of the input data, the pattern learner can be treated as a black box that only receives the labeled input data and performs the optimization that results in a pattern.

*(R2) Interpretability:* The patterns that are generated are provided in a simple but expressive pattern language that was described in [5]. The patterns consist of simple conditions that describe basic properties of the *shape* of the time series data. The pattern that achieved the highest accuracy for our application case was

$$P_2' = P(I_0, I_1, I_2) \;=\; \mathrm{Slope}(I_1) > 0.03535 \;\wedge\; \mathrm{StartV}(I_2) < -0.42348$$

The intuitive meaning of the conditions is that there should be a segment $I_1$ which has a noticeably positive slope, followed by a segment $I_2$ that starts at a low point. This matches the expected pattern for the "spike" in the force that is shown in the example in Fig. 2.

*(R3) Accuracy:* The work in [5] presented a pattern using the same pattern language and data sets, which achieved an accuracy of 0.967, 0.983, and 0.867 for the three data sets, respectively. We applied our pattern learner to a data set that was created from combining these data sets, let it search for a single pattern, and evaluated the resulting pattern individually for the data sets, achieving accuracies of 0.967, 1.0 and 0.9, respectively. So the goal of achieving an accuracy that is similar to that of a pattern created by a domain expert is clearly met, and in fact, the accuracy of the generated pattern is even higher for two of the three data sets.

*(R4) Small Training Data Sets:* The training data for our use case consisted of three different data sets, each having 40 positive and 20 negative instances, which we combined in order to compare the resulting pattern to the baseline pattern that was created manually by a domain expert. The actual value domains of the three data sets differ noticeably. For example, the total duration or absolute value of the recorded force are different. The main similarity of the data sets are the characteristics of the "spike" that indicates a successful assembly. And these characteristics have properly been captured by the pattern learner, even though the actual data contains only 180 training instances which have been created by combining different, even smaller training data sets.

## 8 Conclusions and Future Work

We have successfully applied the approach of automatically learning patterns for complex event detection based on segmented time series data to our main use case. The results are promising in that the process is fully automatic, and generates interpretable patterns that achieve a high classification accuracy, even with small training data sets.

There are several possible directions for future research. One of them is application-driven, namely trying to learn more complex patterns that may appear in other use cases. Anther possible task is a more detailed examination of the influence of constraints and learning parameters on the result, or possible improvements in accuracy that can be achieved with different numerical optimizers. The distinction between the symbolic and the numerical optimization allows different ways of interweaving both optimization methods: One could start with the symbolic optimization and apply the numeric optimization to the resulting pattern, or start with a pattern that describes all possible conditions, optimize it numerically, and use this pattern to initialize the first population of the genetic algorithm. We will continue to publish our results on this.

**Acknowledgment.** This work was done within the project ProDok 4.0, funded by the German Ministry of Education and Research (BMBF) within the framework of the Services 2010 action plan under funding no. 02K14A110. Executive steering committee is the Karlsruher Institut für Technologie - Karlsruhe Institute of Technology (KIT). Project partners are KUKA Deutschland GmbH, ISRA VISION AG, dictaJet Ingenieurgesellschaft mbH and Hochschule Darmstadt - University of Applied Sciences.

## References

1. Apache Software Foundation (2019). http://commons.apache.org/. Accessed 05 Sept 2019
2. Cugola, G., Margara, A.: The complex event processing paradigm. In: Colace, F., De Santo, M., Moscato, V., Picariello, A., Schreiber, F.A., Tanca, L. (eds.) Data Management in Pervasive Systems. DSA, pp. 113–133. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-20062-0_6
3. Ghalwash, M.F., Radosavljevic, V., Obradovic, Z.: Extraction of interpretable multivariate patterns for early diagnostics. In: 2013 IEEE 13th International Conference on Data Mining, pp. 201–210. IEEE (2013)
4. Gregory, M., Shneiderman, B.: Shape identification in temporal data sets. Master's thesis, University of Maryland (2009)
5. Humm, B., van der Meer, G.: Detecting domain-specific events based on robot sensor data. In: Proceedings of the 16th International Conference on Informatics in Control, Automation and Robotics (ICINCO 2019), vol. 1, pp. 398–405 (2019)
6. Margara, A., Cugola, G., Tamburrelli, G.: Learning from the past: automated rule generation for complex event processing. In: Proceedings of the 8th ACM International Conference on Distributed Event-Based Systems, DEBS 2014, pp. 47–58. ACM, New York (2014). https://doi.org/10.1145/2611286.2611289, http://doi.acm.org/10.1145/2611286.2611289
7. Mousheimish, R., Taher, Y., Zeitouni, K.: Automatic learning of predictive cep rules: bridging the gap between data mining and complex event processing. In: Proceedings of the 11th ACM International Conference on Distributed and Event-based Systems, DEBS 2017, pp. 158–169. ACM, New York (2017). https://doi.org/10.1145/3093742.3093917, http://doi.acm.org/10.1145/3093742.3093917
8. Mutschler, C., Philippsen, M.: Learning event detection rules with noise hidden Markov models. In: Benkrid, K., Merodio, D. (eds.) Proceedings of the 2012 NASA/ESA Conference on Adaptive Hardware and Systems (AHS-2012), pp. 159–166 (2012)
9. Wilhelmstötter, F.: (2019). http://jenetics.io/. Accessed 04 Sept 2019