Modeling Recurrent Distributions in Streams using Possible Worlds

Michael Geilke, Andreas Karwath, and Stefan Kramer

Johannes Gutenberg-Universität Mainz, Staudingerweg 9, 55128 Mainz, Germany
Email: {geilke,karwath,kramer}@informatik.uni-mainz.de

***Abstract*—Discovering changes in the data distribution of streams and discovering recurrent data distributions are challenging problems in data mining and machine learning. Both have received a lot of attention in the context of classification. With the ever increasing growth of data, however, there is a high demand of compact and universal representations of data streams that enable the user to analyze current as well as historic data without having access to the raw data. To make a first step towards this direction, we propose a condensed representation that captures the various – possibly recurrent – data distributions of the stream by extending the notion of possible worlds. The representation enables queries concerning the whole stream and can, hence, serve as a tool for supporting decision-making processes or serve as a basis for implementing data mining and machine learning algorithms on top of it. We evaluate this condensed representation on synthetic and real-world data.**

I. Introduction

With increasing amounts of data arriving in the form of streams, there is a high demand for tools that analyze this data and extract relevant information from it. At the time of collecting the data, however, it is often not known what kind of analysis needs to be performed or there are several – possibly even dependent – analysis tasks. Whenever storing the raw data is either not feasible due to the sheer volume or impossible due to privacy concerns, conventional data mining and machine learning algorithms cannot be employed on the raw data. One possible solution to overcome these problems has recently been introduced by Geilke *et al.* [1]. They showed that an online density estimate can be used as condensed representation of the data, on which data mining and machine learning tasks can be performed. However, this condensed representation only represents the current distribution of the data. If the underlying distribution changes, the previous condensed representation adapts to capture the new distribution. This leads to a loss of information, since it is no longer available to the data mining and machine learning algorithms and, therefore, poses a problem for the historical analysis of the data.

A real-world example where recurrent data distributions are likely to occur is the database of a big online store. For each product category (e.g., books, laptops, clothes), we have a stream of incoming transactions, which are the products the customers are buying. The products in a category have a common set of variables, so that each product can be represented as a vector of variables. Then, for each product category, the distribution of the variables changes due to technical advances (e.g., higher CPU clock speeds, larger displays) or trends (e.g., glossy displays, size of shirts, skateboards). Moreover, previously seen distributions can reoccur due to items becoming fashionable again (e.g., lower CPU clock speeds because of tablets, winter clothes vs. summer clothes). We will call these changes in the distribution of the variables *data distribution drift*s and the stationary distribution between consecutive drifts *macro-world*s. Hence, each stream can be divided into a sequence of segments $S_1, \ldots, S_k$ (see Figure 1 for an illustration): segments corresponding to macro-worlds (colored segments) and segments corresponding to data distribution drifts (segments that are dark gray). In this setting, a macro-world is considered recurrent if there are $i, j \in [1; k]$, such that $i \neq j$ and S_i has the same data distribution as S_j.

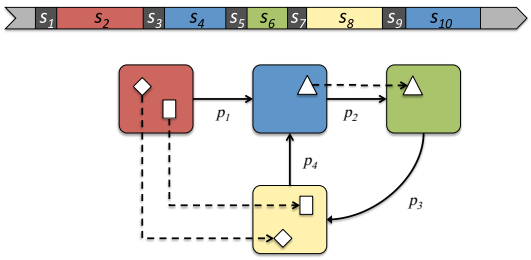

Fig. 1. At the top, a data stream is illustrated that is divided into a sequence of segments $S_1, \ldots, S_k$. Colored segments represent a data distribution and segments that are dark gray represent data distribution drifts. At the bottom is the resulting condensed representation of the data stream with online density estimates for the macro-worlds (colored rectangles), recurrent micro-worlds (white shapes), and transition probabilities (arrows).

In this paper, we address the new problem of representing streams with recurrent macro-worlds. This representation allows the user to query the stream to obtain statements holding for all macro-worlds, some macro-worlds, or no macro-world. In the example given above, this could be useful to make business decisions that are good in most scenarios. We approach this problem in several steps (see Figure 2 for an illustration): First, we identify the macro-worlds and represent them using online density estimates (see Figure 2a). If a macro-world reoccurs, the corresponding estimate is reused. In cases where

a novel macro-world is identified, a new density estimate is initialized. In order to obtain the best possible estimate and to reuse as much as possible of an existing estimate, we also identify independent parts of the macro-worlds, which we call *micro-world*s. For the relationships between the worlds, we construct a graph representing the transition probabilities between the macro-worlds (see Figure 2b) and the recurrence probabilities of the micro-worlds (see Figure 2c). Hence, the resulting condensed representation models the densities of the macro- and micro-worlds, their transition probabilities, as well as the recurrence probabilities of micro-worlds.

The main contributions are as follows:

1) A condensed representation of data that may contain several, possibly recurrent macro-worlds.
2) The introduction of *micro-worlds* additionally to the notion of possible worlds, where a micro-world is part of a world that is treated independently from the rest and can, therefore, be reused in other worlds.
3) The introduction of *modules* to recently proposed online density estimates [2]. From a probabilistic point of view, these modules are parts of the density that are independent from the remaining density and will be used as an estimate for micro-worlds.
4) An evaluation on synthetic and real-world data of modules as estimates and the detection of recurrent macro- and micro-worlds.

The remainder of the paper is structured as follows: In Section II, we discuss related work. Subsequently, in Section III, we introduce possible worlds and extend them for the purpose of representing data streams with recurrent macro- and micro-worlds. In Section IV and V, we present several algorithms that provide a condensed representation of the data and evaluate them in Section VI. We conclude the paper with an outlook on future work and some final remarks (Sections VII and VIII).

II. Related Work

Although there is no work targeting condensed representations that are based on stationary data distributions in the stream (in our case, these are the macro-worlds), concept drifts and recurrent concepts already received some attention in the community – to our knowledge exclusively in the context of classification. For example, Gama *et al.* [3] exploited the relationship between the underlying data distribution of the stream and the error rate of the learning algorithm to detect concept drifts. A similar approach was pursued by Harel *et al.* [4], who resampled from the current data instances to obtain statements via the errors of the underlying algorithms. Another example is the work by Bach and Maloof [5], who performed concept-drift detection based on the distribution of the classifiers' predictions. They employed Bayesian model comparisons, but considered the conditional probability of the prediction given the feature vector instead of the corresponding joint distribution. A rather different direction was taken by Demšar *et al.* [6]. They used the explanation methodology to obtain a stream of explanations, on which they then ran a Page-Hinkley test to detect concept drifts. A more general approach was pursued by Kifer *et al.* [7], who made an attempt at formalizing the detection and quantification of change. They proposed a general framework to capture changes in the distribution of the stream and evaluated several test statistics with respect to their performance. Dries and Rückert [8] also based the detection of concept drifts on the distribution of the underlying stream and proposed several tests such as the *CNF density estimation test*.

Recently, there was also a substantial increase of work in the direction of recurrent concepts (not distributions). For example, Gama and Kosina [9] proposed a framework for handling recurrent concepts in a data stream. In order to decide whether to reuse previous models, they train meta-learners. Meta-learners were also used by Gomes *et al.* [10] to select models based on context information for an ensemble classifier. Other approaches were pursued by Sripirakas and Pears [11], who used the discrete Fourier transform to store previously seen concepts in a compact way and to match the concepts without the need for user-defined thresholds, by Lazarescu [12], who presented a multi-resolution learning approach to track concept drift and recurrent concepts, or by Gonçalves Jr and Maior de Barros [13], who detect recurrent concepts based on the distribution of the data by employing statistical tests. As in this paper, the latter approach also incorporates an existing concept-drift detection algorithm into their framework. However, their work targets the classification setting and focuses on reusing classifiers for recurrent concepts.

For the recurrent concepts, most approaches employ some kind of model repository to store previously seen models. Whenever a concept drift is detected, the model repository is checked for a model fitting the emerging concept. If yes, the model is reused, otherwise a new model is initialized. In contrast to this, the approach presented in this paper constructs a graph consisting of macro-worlds and their independent subcomponents, which are the micro-worlds. Whereas the notion of a macro-world is in certain aspects related to the one of a concept (e.g., both capture something stationary of a data stream), micro-worlds have, to the best of our knowledge, no equivalent in current research on concept-drifting data streams.

Besides concept drifts and recurrent concepts, there are also several more specific topics that were studied. For example, the detection of novel and recurrent classes (Masud *et al.* [14]), one-class classifiers in data streams (Krawczyk and Woźniak [15] and Liu *et al.* [16]), or mining conept-drifting data streams with skewed distributions (Gao *et al.* [17]). However, none of these special topics are covered in this paper.

Universal condensed representations have only been considered for data stream mining with an underlying stationary data distribution [1]. Otherwise, condensed representations were usually restricted to specific tasks. For example, in pattern mining, a sufficient subset of all frequent itemsets is considered a condensed representation of them if all frequent itemsets can be derived from them [18], [19].

To the best of our knowledge, possible worlds have not been used to model recurrent macro-worlds, yet. However, they have been used in other contexts in machine learning such as handling uncertain data [20], [21], mining patterns [22], [23], or in the context of uncertain reasoning [24].

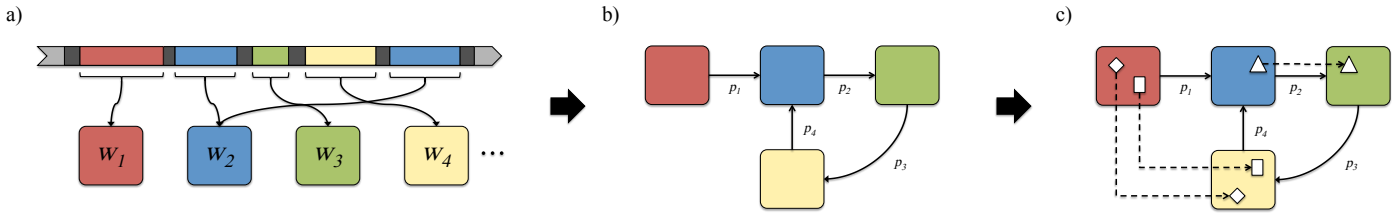

Fig. 2. The main steps for creating the proposed condensed representation for data streams with recurrent macro-worlds. The colored boxes are the macro-worlds, the white shapes are the micro-worlds, and the labeled arrows and the dashed arrows are the transition probabilities, where the labeled arrows are transitions between macro-worlds and the dashed arrows are transitions between micro-worlds.

III. From Streams to Possible Worlds

In the following sections, we propose a condensed representation for data streams with recurrent macro-worlds, which is determined in three main steps (see Figure 2 for illustration): First, the data stream is divided into transition phases representing a data distribution drift (dark gray) and parts belonging to macro-worlds (colored). Second, a graph is constructed representing the relationships between the macro-worlds. The edges are labeled with values expressing the probability of going from one world to another one. Finally, independent parts of the macro-worlds are identified, so-called micro-worlds, that reoccur in other macro-worlds.

To model the different macro-worlds and the relations between them, we employ the notion of possible worlds [25]. Possible worlds consist of propositional letters to which truth values are assigned. The same letter can exist in different possible worlds but may have a different truth value. Given two worlds w and w', an accessibility relation indicates whether w' is accessible from w. This allows to evaluate the truth value of a given statement with respect to several interconnected worlds. In this context, two operators, $\Box$ and $\Diamond$, are important. $\Box$ indicates whether something is necessary (i.e., holds in all worlds), and $\Diamond$ indicates whether something is possible (i.e., holds in at least one world).

The condensed representation has two levels of abstraction. The first level describes the structure of the streams, i.e., which segments exist and how they are related to each other. On this level, each segment with a stationary data distribution (segments $S_2, S_4, S_6, S_8, S_{10}$ in Figure 1) is associated with a possible world. The worlds themselves are described on the second level of the abstraction, for which we require a representation that estimates the underlying data distribution and supports inference tasks. The latter are basic operations, which will enable queries on the condensed representation.

In most cases, only parts of the world will reoccur. For example, in the online store example, only the features CPU clock speed and display size could be recurrent, whereas the overall world is different. Therefore, we introduce the notions of macro- and micro-worlds as addition to possible worlds (the white shapes in Figure 1 are the micro-worlds). The following definition formalizes these ideas and are based on the works by Gamut [25]. Novel compared to Gamut are the notion of macro- and micro-worlds and the introduction of probabilities.

Definition 1: Let W be a non-empty set of possible worlds with $\Omega \subseteq W$ (macro-worlds), $\Theta \subseteq W$ (micro-worlds), $\Omega \cap \Theta = \emptyset$, $\Omega \cup \Theta = W$, and $\forall \theta \in \Theta \; \exists w \in \Omega : \theta \subsetneq w$. Further, let $R : W \times W \rightarrow [0; 1]$ be a binary relation representing the transition probabilities between possible worlds (micro- and macro-worlds), and let V be a valuation function that assigns a probability $V_w(a)$ for every proposition letter a in every context of $w \in W$. Then (Ω, Θ, R, V) is a model for propositional logic.

Micro-worlds have several benefits: They are independent components, which can be treated separately. This means that only the relevant parts are considered when trying to capture them, which yields an overall smaller and probably more accurate representation. Also, similarities between macro-world are easier to detect, since worlds can be compared on a micro-world basis.

IV. Condensed Representation: Level One

The first step in constructing a condensed representation of streams is to determine segments of the stream that belong to certain macro-worlds. A stream can be divided into two types of segments (see Figure 1): *macro-world segments* containing *macro-world instances* (i.e., instances that clearly belong to a certain macro-world) and *transition segments* containing *transition instances* (i.e., instances that belong to transitions from one macro-world to another one). Whereas macro-world instances are used to construct the possible worlds, transition instances are usually not very interesting, since it is not clear to which world they belong and are, therefore, unsuitable for improving the online density estimate of the macro-world.

In order to split the stream into these segments, we will use two algorithms: $detectDrift$ and $equalDensities$. $equalDensities$ compares the distribution between a sample of instances and a density estimate by employing a Wilcoxon rank-sum test, and $detectDrift$ identifies changes in the distribution of the stream using $equalDensities$. The pseudocode of both are left out, as they are explained in detail in Section IV-B. Please notice that these algorithms are not the main focus of the paper, but rather tools that we employ to accomplish the task of building a condensed representation of a stream with recurrent macro-worlds. It is basically possible to use any concept-drift detection method that is based on the data distribution of the underlying stream – some of them were pointed out in the related work section.

In the remainder of this section, we will first explain how to capture the dependencies between the worlds by observing transitions from one macro-world to another, and then discuss how the individual macro-worlds are represented.

Algorithm 1: updateLevelOneRepresentation

Input: unprocessed instances $insts$, new instances $insts'$, the windows S_B, S_M, S_E, density estimate f, the set of density estimates F, a set of edges $m : F \times F \to \mathbb{N}$, significance level α

```
1  if |insts| + |insts'| < |S_B| + |S_M| + |S_E| then
2  |  append insts' to insts
3  else
4  |  matchFound ← false
5  |  for f' ∈ F with equalDensities(f', S_E, α) do
6  |  |  matchFound ← true
7  |  |  m(f, f') ← m(f, f') + 1
8  |  |  f ← f'
9  |  if matchFound = false then
10 |  |  f' ← initialize density estimator
11 |  |  m(f, f') ← 1
12 |  |  append f' to F
13 |  |  f ← f'
14 |  for inst_1 ∈ insts' do
15 |  |  q ← queue with elements S_B, S_M, S_E
16 |  |  inst_2 ← q.pop()
17 |  |  q.push(inst_1)
18 |  |  split q into S_B, S_M, S_E
19 |  |  if detectDrift(f, S_B, S_M, S_E, α) then
20 |  |  |  insts ← insts'[index(inst_1) + 1 :]
21 |  |  |  break loop
22 |  |  else
23 |  |  |  f.update(inst_2)
```

A. Relationships between macro-worlds

In order to capture the relationships between the macro-worlds, we basically construct a graph where the nodes are the macro-worlds of the stream – represented by online density estimates – and an edge between two nodes N_1 and N_2 with label p means that the probability of going from macro-world w_1 to macro-world w_2 is p. Algorithm 1 is responsible for maintaining the relationships between the macro-worlds (lines $1-8$) and for passing instances to the second level of the condensed representation (line 23), which are the online density estimates. To accomplish these tasks, it maintains a window (consisting of the sequences S_B, S_M, S_E), the instances that just arrived ($insts'$), and an additional buffer ($insts$), which we use to ensure that we possess enough instances to detect data distribution drifts. As soon as enough instances for concept drift detection are available (i.e., $|insts| \geq |S_B|+|S_M|+|S_E|$), the algorithm moves the window instance by instance (line $15-18$) and performs a test to detect data distribution drifts. If a drift has been detected (line 19), all instances up to this point are removed, which includes all instances from S_B, S_M, and S_E (see Figure 3), and the algorithm waits until enough instances are available again (line $1-3$). If no drift has been detected (line 22), it updates the counters of m, which will be used to estimate the probabilities of the edges. For any two density estimates $f, f' \in F$, m stores the number of times we transitioned from the macro-world represented by f to the macro-world represented by f'. Using m, the transition probability p of going from $f \in F$ to $f' \in F$ is computed as follows (simply maximum likelihood, i.e., the relative frequency): $p = \frac{m(f,f')}{\sum_{g,g' \in F} m(g,g')}$. In lines $5-7$, the algorithm tries to match the emerging density estimates to one of the already existing ones. If none is found, a new density estimate is initialized. In the last step of the algorithm, the instances that belonged to S_B in the previous iteration (see t_1 in Figure 3) are forwarded to the current online density estimate (line 23).

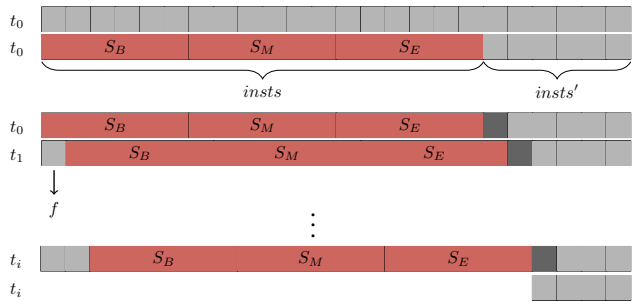

Fig. 3. In order to detect drifts in the data distribution of the stream, a window is used that is split into three segments of equal size: S_B, S_M, and S_E. If enough instances are available to perform drift detection, $insts$ are exactly the instances contained in $S_B \circ S_M \circ S_E$, whereas $insts'$ are the new instances that need to be processed. In order to process the next instance of $insts'$ (dark gray), the algorithm treats $S_B \circ S_M \circ S_E$ as a queue and removes the oldest instance whenever a new instance is appended. The instance that has been removed is then forwarded to the current density estimate f. If a drift is detected (see t_i), all previous instances are discarded and the algorithm waits until enough instances are available for further drift detections.

B. Identifying macro-worlds

The procedure described above needs to identify segments with macro-world instances and segments with transition instances. In an supervised setting, there exist several methods and frameworks to detect concept drift and recurrent concepts (e.g., [13], [9]), which usually measure some kind of error of the classifiers and use a statistical test or property to detect a concept drift. However, since we want to use joint density estimates as condensed representation, the supervised setting does not apply in our case. Other approaches such as methods proposed by Kifer *et al.* [7] or Dries and Rückert [8] consider the data distribution of the stream to detect data distributions drift. Since the main focus of the paper is the condensed representation of streams with recurrent macro-worlds, we decided in favor of a simple approach, in our case: the *CNF density estimation test* [8], for which the authors pursued a window-based approach. They split the window into three parts and determine whether the first two parts have the same distribution as the last one. In order to capture changes in the distribution, they transform the instances to a binary representation and compare the distributions of the different parts. For this purpose, they introduce a predicate *covers*, use this predicate to describe subsets of variables that are consistent with a given set of instances, and perform a Mann-Whitney test to decide whether there is a significant difference between the distributions.

Following this idea, we employed a window-based approach (a buffer) to ensure that only those instances reach the condensed representation that really belong to the corresponding concept. Therefore, we keep instances in a buffer as long as the density estimation test could still detect a data

distribution drift. Hence, we forward only the instances of S_B to the density estimate of the current macro-world. Instances from S_M and S_E will only be forwarded if they become part of an S_B of another partition, which also means that we probably do not capture all the macro-world instances that are available. To detect changes in the distribution, we perform a Wilcoxon rank-sum test [26], [27], [28] on S_B and S_E using the current density estimate – instead of the predicate *covers*.

V. Condensed representation: level two

In the previous section, we described how to identify the macro-worlds of a stream and how to model the relationships to each other using the possible worlds semantics. In this section, we consider the second level and explain how the macro-worlds can be represented in a condensed way. For this purpose, we introduce *modules* to density estimates that will serve as an estimate for the *micro-worlds*.

Definition 2: Let $X := \{X_1, \ldots, X_n\}$ be a set of variables and let $f(X_1, \ldots, X_n)$ be a discrete joint density. If $X = A_1 \dot\cup \ldots \dot\cup A_k$, $k > 1$, and $A_i \subsetneq X$ for $1 \leq i \leq k$, then the factors of the product $\prod_{i=1}^{k} f_i(A_i)$ are called *modules* of f if, for each $1 \leq i \leq k$, f_i is a discrete joint density and there is no decomposition $A_i = B_1 \dot\cup \ldots \dot\cup B_l$, such that $l > 1$ and $f_i(A_i)$ can be represented by a product $\prod_{j=1}^{l} f_{i,j}(B_j)$ where $f_{i,j}$ is a discrete joint density for $1 \leq j \leq l$.

In other words, modules are the independent factors of a density. For example, if a discrete joint density $f(X_1, X_2, \ldots, X_8)$ can be represented as a product $f_1(X_1, X_3, X_8) \cdot f_2(X_2, X_4, X_5) \cdot f_3(X_6) \cdot f_4(X_7)$ and the f_i, $i \in \{1, 2, 3, 4\}$, cannot be decomposed any further, then f_1, f_2, f_3, and f_4 are the modules of f.

A. Modules in density estimates

For the condensed representation, we will employ the online density estimators, called EDDO [2], which can be used to derive modules for the estimate they represent. For example, if a Hoeffding tree [29] is used as base classifier, then the Hoeffding tree algorithm implicitly chooses a subset of variables to construct the tree. If a graph is constructed that has the classifiers of the chain as nodes and connects two nodes if the classifiers belonging to that node have common variables, then the cliques in the graph and all nodes that do not belong to a clique are the modules of the estimate. However, in this work we will pursue a more explicit approach to modules, which is applicable to data streams consisting of discrete and/or continuous variables. The algorithm we are about to present constantly estimates the structure of the density by grouping the variables into modules, which is achieved by determining connected components of the variables based on the normalized mutual information – a measure for the dependence of two random variables. For each module, an online density estimate is employed to provide a density estimate for this module. When new instances arrive, it may turn out that the proposed module does no longer match the data, because it is not a module at all, a part of a larger module, or only a part of it is a module. In this case, the module is entirely removed, and the estimate of each newly introduced or modified module is then replaced by a new online density estimate.

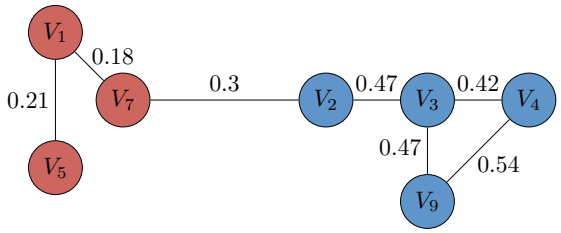

Fig. 4. The figure shows variables (circles) that are dependent on each other according to the normalized mutual information. The values on the edges are the normalized mutual information for a given pair of variables. The algorithm creates two clusters: the variables V_1, V_5, and V_7 are clustered and the variables V_2, V_3, V_4, and V_9. Although there is a connection between V_7 and V_2, the clusters are not joined, since the normalized mutual information values within the clusters are too different (see line 8 of Algorithm 2). For the same reason, the algorithm also does not group the variables V_7, V_2, V_3, V_4, and V_9 together.

In the discrete case, the mutual information [30] of two variables X and Y is defined as $I(X,Y) := \sum_{x \in X} \sum_{y \in Y} p(x,y) \log\left(\frac{p(x,y)}{p(x) \cdot p(y)}\right)$, where $p(x,y)$ is the probability of $(x,y) \in X \times Y$ and $p(x)$ and $p(y)$ are the corresponding marginal probabilities. For continuous variables, the summations are replaced by integrals. However, instead of computing integrals for continuous variables, we will discretize continuous variables using an online discretization method proposed by Gama and Pinto [31]. Since the mutual information does not necessarily lie between 0 and 1, which would be advantageous to define a general threshold that enables us to decide whether two variable are considered dependent on each other, we will use a normalized version of the mutual information (NMI) [32]:

$$NMI(X,Y) := \frac{I(X,Y) \cdot 2}{H(X) + H(Y)},$$

where H is the entropy. The threshold to decide whether two variables are dependent on each other is set to some value θ, i.e., a value smaller than θ means not dependent, and a value larger than or equal to θ means dependent. However, not all variables that are considered as dependent on each other should end up in the same module. In some initial experiments, we noticed that the range of the normalized mutual information is often quite different and we also noticed that the NMI values provide important information about the variables that should be grouped together. Figure 4 illustrates a typical situation. Here, variables V_1, V_5, and V_6 are grouped together, since they have a similar NMI value of around 0.2. Variables V_2, V_3, V_4, and V_9 are grouped together, since they also have a similar NMI value (around 0.48). But both groups are not joined, although V_7 and V_2 are connected to each other.

Algorithm 2 maintains the modules and their estimates. It first updates the normalized mutual information NMI for all pairs of variables (lines $2-3$). Then, every ρ instances (line 4), the structure of the modules is recomputed. In lines $5-6$, D is determined, which defines dependencies between two variables X_i and X_j iff their normalized mutual information is greater than or equal to θ_1. Subsequently, variables with a similar normalized mutual information are grouped together and considered as modules (lines $7-9$), so that we obtain a simple clustering of the variables based on the normalized mutual information. In the remainder, the algorithm reuses

Algorithm 2: updateLevelTwoRepresentation

Input: variables X, recomputation threshold $\rho \in \mathbb{N}$, instance counter $count$, set of modules $modules$, instance $inst$, object NMI, NMI threshold θ_1, clustering threshold θ_2

1 $modules' \leftarrow \emptyset$
2 **for** $(X_i, X_j) \in X \times X,\ 1 \leq i < j \leq n$ **do**
3 | update $NMI(X_i, X_j)$
4 **if** $count \mod \rho = 0$ **then**
5 | initialize $D : X \times X \mapsto \{0, 1\}$
6 | set $D(X_i, X_j) = 1$ iff $NMI(X_i, X_j) >= \theta_1$
7 | $C_0 \leftarrow \{X_i \mid \nexists X_j : D(X_i, X_j) < \theta_1\}$
8 | Create clustering $C_1, \ldots, C_k$, s.t. $\forall X_1, X_2, X_3, X_4 \in C_i :$ $\frac{NMI(X_1, X_2) - NMI(X_3, X_4)}{NMI(X_1, X_2)} < \theta_2$
9 | $\mathcal{C} \leftarrow \{C_0, C_1, \ldots, C_k\}$
10 | **for** $C_i \in \mathcal{C}$ **do**
11 | | $m \leftarrow$ create module from C_i
12 | | $e \leftarrow$ initialize density estimate
| | $modules'.append((m, e))$
13 **for** $(m, e) \in modules'$ **do**
14 | **if** $\exists e' : (m, e') \in modules$ **then**
15 | | $e \leftarrow e'$
16 | $modules'.append((m, e))$
17 | $m.update(inst)$
18 $count \leftarrow count + 1$

density estimates for every module existing before (lines $10 - 12$) and passes the instance $inst$ to all modules (line 17).

B. Shared modules

A module is an estimate for a micro-world that can be shared among several possible worlds. If a new macro-world emerges, we need to examine if it contains already discovered modules. Since macro-worlds are represented by estimates of joint densities, we require an algorithm that tests whether the estimates are constructed from the same distribution. A good choice for this task is usually the KL-divergence, which is, however, not suitable in our setting for several reasons: (1) Computing the KL-divergence is computationally expensive. (2) The estimates can differ with respect to their precision, since one estimate has been constructed from millions of instances, whereas the other one could have been constructed from a few thousand instances only. (3) Even if the estimates have reached the same level of precision, it remains unclear which KL-divergence is acceptable. As an alternative, we will use $equalDensities$, which employs a Wilcoxon rank-sum test to compare two distributions (see Section IV-B for reference). Using this, we can compare the distribution of the already discovered module and the newly emerging module.

C. Diverging of recurrrent micro-worlds

Since a module that is shared among several possible worlds W' is the same object, updating the corresponding module would always affect every world in W'. Hence, if a shared module develops differently, it can no longer be shared. In this case, we make individual copies and assign them to each world belonging to W'. It is crucial that the copies are created before a deviation occurred. Otherwise, this deviation would be a part of all worlds in W'. In order to solve this, we perform data distribution drift detection in combination with a buffering system as described in Section IV.

VI. Evaluation

In this section, we evaluate the algorithms presented in Section IV and Section V. We first measure how modules affect the performance of the density estimate using synthetic and real-world datasets (the corresponding online density estimator will be denoted by $EDDO^{Mod}$). Then the construction of possible worlds is evaluated on real-world data.

A. Evaluating modules

Compared to $EDDO$, $EDDO^{Mod}$ provides a more explicit representation of the density by grouping variables into independent components (modules). This leads to a representation that is easier to interpret by users, since variables that are dependent on each other are grouped together. Despite this more explicit representation, we will show that both methods have a similar performance. Since Geilke *et al.* already compared $EDDO$ to 24 Bayesian net structure learners from the *bnlearn* package [33] with a comprehensive set of experiments and showed that $EDDO$ outperforms standard Bayesian net structure learners, we focus our presentation on a comparison of $EDDO^{Mod}$ and $EDDO$. (In further experiments, we also compared these Bayesian structure learners to $EDDO^{Mod}$. For example, on the US-Census dataset, the best $EDDO^{Mod}$ estimator has an average log-likelihood of -40.47, whereas the best Bayesian structure learner *TABU mle* has an average log-likelihood of -45.3.)

As synthetic data, we generated streams from several Bayesian networks. For each stream, we randomly selected a number of modules $1 < k \leq 5$, and, for each module, we created between 4 and 8 new variables. Subsequently, we randomly generated Bayesian networks for each module using the *random.graph* method of the *bnlearn* package. Its parameter *method* was set to *melancon*, which uses a Markov chain to draw acyclic, directed graphs uniformly at random [34]. In order to generate the conditional probability tables (CPTs) of the nodes, we randomly chose values from the interval $[0; 1)$ for each entry and normalized the columns, such that they sum up to 1. From the streams generated this way, we drew M instances ($M \in \{10^3, 10^4, 10^5, 10^6\}$). As real-world datasets, we used several publicly available datasets: Electricity, Shuttle, and Covertype. Additionally, we modified the water level dataset prepared by Schlüter and Conrad [35], where the water level of rivers in Germany were measured over twelve years. Since the data is very sparse, we only considered two stations (Arloff and Meschede) and removed instances where parts of the measurements were missing. For each station, we had nine features: the values of the closest four neighboring stations of the previous two days (eight values) and the class attribute indicating whether the water level increased, decreased or stayed the same.

In order to measure the performance of each estimator, we split the stream in two parts, construct the density estimate

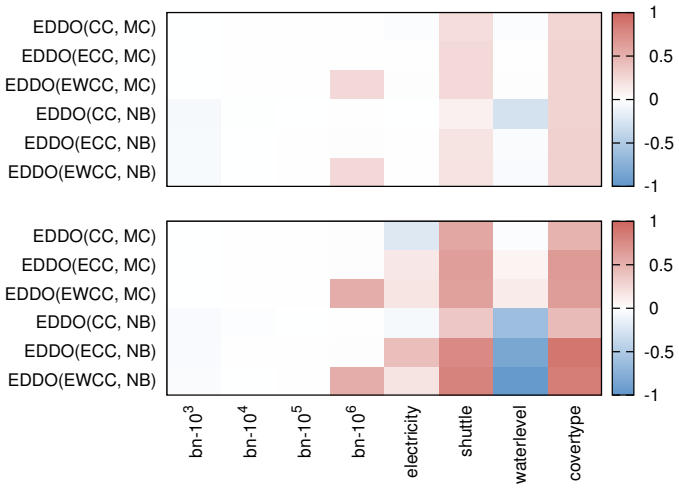

Fig. 5. The figure shows the percentaged difference of the average log-likelihood (upper plot) and the percentaged rank sum (lower plot) between $EDDO$ and $EDDO^{Mod}$ for various datasets. The percentaged values are determined by first computing the average log-likelihood and the rank sum of $EDDO$ and $EDDO^{Mod}$, then computing the difference of the values of $EDDO^{Mod}$ and $EDDO$ and dividing it by the value of $EDDO$. For each category (synthetic and real-world), the datasets are first ordered by their number of attributes and then by their number of instances.

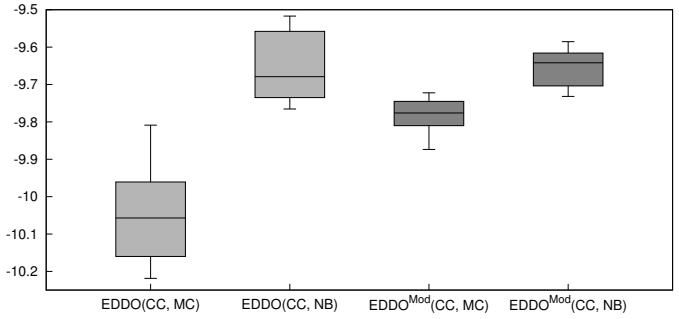

Fig. 6. This figure illustrates the performance (measured in average log-likelihood) on the electricity dataset of $EDDO$ and $EDDO^{Mod}$ using a single random classifier chain. In many cases, $EDDO^{Mod}$ outperforms $EDDO$. For other datasets, such as shuttle or covertype, the opposite can be observed.

from the first part, and measuring the average log-likelihood on the second part. Subsequently, we computed the average rank sum for each estimator and for each dataset. Although the KL-divergence would have been a better choice for the synthetic data, we had to use the log-likelihood due to the large number of attributes (up to 40).

The results are summarized by Figure 5. Although in many cases, the rank sum is lower for $EDDO$ (exceptions are, for example, the water level dataset, some cases of the electricity dataset, or bn-10^3), the heat map at the top shows that the differences are rather low (exceptions are the shuttle and covertype dataset). Especially if the number of instances increases, while the attributes stay fixed, $EDDO^{Mod}$ even outperforms $EDDO$ in many cases. This effect is due to the way the proposed method manages its modules. In the beginning, when only a few instances have been observed, the partitioning of the variables into modules changes quite strongly before it starts stabilizing. At that time, the previous modules have been discarded in favor of new modules, which results in fewer training instances for these modules, thereby yielding an estimate that is not as accurate as an estimate that has been computed with all instances. With more instances, this effect becomes less visible. For datasets with many attributes (e.g., covertype) or many values per attributes (e.g., shuttle), the effect is more visible, because fewer observations are available compared to the number of attribute-value combinations, which result in a worse estimate of the mutual information. As stated in the beginning of the section, there is a, albeit relatively small, prize to be paid for the explicit representation of modules, macro-/micro-worlds and transition probabilities.

B. Recurrent macro- and micro-worlds

Several publications monitor data distributions of streams to detect concept drifts (e.g, Gonçalves Jr and Maior de Barros [13]) and use the detected drifts to train new classifier or to continue training previous classifiers. However, constructing a universal condensed representation that enables queries – and as a result enables data mining and machine learning tasks on top of it – has to the best of our knowledge not been considered before. Hence, there is currently no other method trying to detect something like recurrent macro- and micro-worlds, and we could only perform the evaluation of recurrent worlds based on a real-world data such as the water level dataset.

How quickly the distribution of the dataset changes is visualized by Figure 7 (see the plot at the top), which shows the distance of each instances to the vector $\vec{0}$ – this is not to be confused with the distribution of the data, it only indicates changes in the distribution. Between two data distribution drifts there are often only a few hundred instances, which makes the detection of macro- and micro-worlds a challenging task. However, even in this difficult setting, the proposed algorithm provides a rather accurate description of the stream, as illustrated at the bottom of Figure 7. Large parts of this stream segment are almost exactly captured by the macro-worlds, whereas other parts show only minor deviations. In total, the algorithm finds 416 macro- and 1259 micro-worlds, of which 484 micro- and 110 macro-worlds are unique. Although it is difficult to measure the meaningfulness of the micro-world detection, its classification is performed according to our intuition of the data. For example, it groups variables belonging to the same stations (i.e., measurements from different days), variables indicating the current trend of the stations (i.e., whether the water level decreased, increased, or stayed the same), or variables belonging to the same neighboring stations.

In order to evaluate the detection of recurrent micro- and macro-worlds, we connected the water level dataset two times in a row. Hence, all macro- and micro-worlds have to reoccur again and should be detected by the algorithm. In this setting, the proposed method rediscovers 100% of the macro-worlds and 78.6% of the micro-worlds, which indicates that $EDDO^{Mod}$ is, in principle, able to rediscover macro- and micro-worlds.

VII. Outlook: Queries on Possible Worlds

The proposed condensed representation is not only able to represent streams with recurrent macro-worlds, but also enables queries on top of it, which can be used to analyze

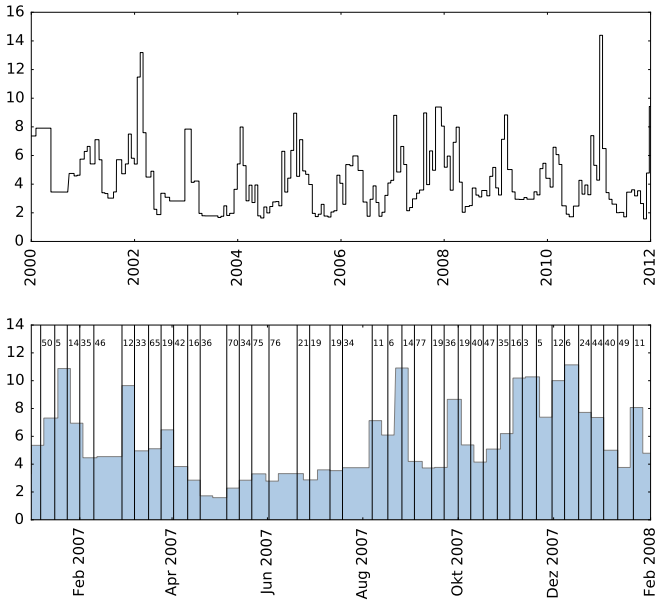

Fig. 7. At the top, changes in the data distribution are illustrated by plotting the distance to the vector $\vec{0}$ for each measurement. At the bottom, macro-worlds are shown that have been detected between January 2007 and February 2008 (recurrent macro-worlds share the same number and every number smaller than or equal to 74 already occurred earlier).

the underlying data. In order to model this, we introduce the following operators:

Definition 3: Let $M = (\Omega, \Theta, R, V)$ be a model for propositional logic, and let p be some probability. Then $V_{M,w}(\phi)$, the probability of ϕ in w given M, is defined by the following clauses, which are an extension of the clauses by Gamut [25]:

1) $V_{M,w}(a) = V_w(a)$, $\forall$ variable-value pairs a
2) $V_{M,w}(\neg\phi) = p$ iff $V_{M,w}(\phi) = 1 - p$
3) $V_{M,w}(\phi \vee \psi) = min(V_{M,w}(\phi), V_{M,w}(\psi))$
4) $V_{M,w}(\phi \wedge \psi) = max(V_{M,w}(\phi), V_{M,w}(\psi))$
5) $V_{M,w}(\phi \rightarrow \psi) = min(V_{M,w}(\neg\phi), V_{M,w}(\psi))$
6) $V_{M,w}(\Box\phi) = min_{w' \in W:R(w,w')}(V_{M,w'}(\phi))$
7) $V_{M,w}(\Diamond\phi) = max_{w' \in W:R(w,w')}(V_{M,w'}(\phi))$

Definition 3 provides the foundation to perform queries on the condensed representation. With inference algorithms, one can first restrict the condensed representation to the parts which are interesting to the user. Subseqently, questions like: *Given world* w*, what is the probability that* a *is true?* can be posed. If the $\Box$ operator is used, this probability is computed with respect to all worlds directly or indirectly connected to w, and if the $\Diamond$ operator is used, this probability is computed with respect to the world that is directly or indirectly connected to w and yields the largest probability for a. To compute those probabilities, inference algorithms as the ones introduced by Geilke *et al.* [2] can be employed.

VIII. Conclusions

We proposed a novel approach of representing data from a stream with recurrent macro-worlds (stationary data distributions) using possible worlds and online density estimates. As the density estimates enable data mining and machine tasks without access to the raw data [1], the proposed representation allows to perform these tasks on a much broader scale. In the experiments, we showed that the representation is accurate, to a certain degree, and can capture not only recurrent macro-worlds, but also enables the detection of micro-worlds. This is, to the best of our knowledge, the first time that possible worlds and micro-worlds are employed in the context of streams with recurrent data distributions.

In the future, we intend to work on several extensions of modules and the way they are computed. In the current version, a module is always considered a joint density, which could be extended to conditional densities. Moreover, we plan to reuse existing estimates more efficiently. If a module changes, the estimate is discarded and replaced by a completely new one. In some cases, however, a module might be split into several smaller modules, such that the current estimate can be reused by marginalizing out the variables. Hence, we could use an estimate that has been constructed from possibly thousands of instances instead of using an estimate that is constructed from only a few instances.

References

[1] M. Geilke, A. Karwath, and S. Kramer, "A probabilistic condensed representation of data for stream mining," in *International Conference on Data Science and Advanced Analytics*, 2014, pp. 297–303. [Online]. Available: http://dx.doi.org/10.1109/DSAA.2014.7058088

[2] M. Geilke, A. Karwath, E. Frank, and S. Kramer, "Online estimation of discrete densities," in *Proceedings of the 13th IEEE International Conference on Data Mining*, 2013, pp. 191–200.

[3] J. Gama, P. Medas, G. Castillo, and P. P. Rodrigues, "Learning with drift detection," in *Advances in Artificial Intelligence - Proceedings of the 17th Brazilian Symposium on Artificial Intelligence*, 2004, pp. 286–295.

[4] M. Harel, S. Mannor, R. El-Yaniv, and K. Crammer, "Concept drift detection through resampling," in *Proceedings of the 31st International Conference on Machine Learning*, 2014, pp. 1009–1017.

[5] S. H. Bach and M. A. Maloof, "A bayesian approach to concept drift," in *Advances in Neural Information Processing Systems 23: Proceedings of the 24th Annual Conference on Neural Information Processing Systems*, 2010, pp. 127–135.

[6] J. Demsar, Z. Bosnic, and I. Kononenko, "Visualization and concept drift detection using explanations of incremental models," *Informatica (Slovenia)*, vol. 38, no. 4, 2014.

[7] D. Kifer, S. Ben-David, and J. Gehrke, "Detecting change in data streams," in *(e)Proceedings of the Thirtieth International Conference on Very Large Data Bases*, 2004, pp. 180–191.

[8] A. Dries and U. Rückert, "Adaptive concept drift detection," *Statistical Analysis and Data Mining*, vol. 2, no. 5-6, pp. 311–327, 2009.

[9] J. Gama and P. Kosina, "Recurrent concepts in data streams classification," *Knowledge and Information Systems*, vol. ro. 2013, 2013. [Online]. Available: http://dx.doi.org/10.1007/s10115-013-0654-6

[10] J. B. Gomes, E. M. Ruiz, and P. A. C. Sousa, "Learning recurring concepts from data streams with a context-aware ensemble," in *Proceedings of the 2011 ACM Symposium on Applied Computing*, 2011, pp. 994–999. [Online]. Available: http://doi.acm.org/10.1145/1982185.1982403

[11] S. Sakthithasan and R. Pears, "Mining recurrent concepts in data streams using the discrete fourier transform," in *Proceedings of the 16th International Conference on Data Warehousing and Knowledge Discovery*, 2014, pp. 439–451. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-10160-6_39

[12] M. Lazarescu, "A multi-resolution learning approach to tracking concept drift and recurrent concepts," in *Pattern Recognition in Information Systems, Proceedings of the 5th International Workshop on Pattern Recognition in Information Systems*, 2005, p. 52.

[13] P. M. G. Jr and R. S. M. de Barros, "Rcd: A recurring concept drift framework," *Pattern Recognition Letters*, vol. 34, no. 9, pp. 1018–1025, 2013.

[14] M. M. Masud, T. Al-Khateeb, L. Khan, C. C. Aggarwal, J. Gao, J. Han, and B. M. Thuraisingham, "Detecting recurring and novel classes in concept-drifting data streams," in *Proceedings of the 11th IEEE International Conference on Data Mining*, 2011, pp. 1176–1181.

[15] B. Krawczyk and M. Wozniak, "Incremental learning and forgetting in one-class classifiers for data streams," in *Proceedings of the 8th International Conference on Computer Recognition Systems*, 2013, pp. 319–328.

[16] B. Liu, Y. Xiao, P. S. Yu, L. Cao, Y. Zhang, and Z. Hao, "Uncertain one-class learning and concept summarization learning on uncertain data streams," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 2, pp. 468–484, 2014.

[17] J. Gao, W. Fan, J. Han, and P. S. Yu, "A general framework for mining concept-drifting data streams with skewed distributions," in *Proceedings of the Seventh SIAM International Conference on Data Mining*, 2007, pp. 3–14.

[18] N. Pasquier, R. Taouil, Y. Bastide, G. Stumme, and L. Lakhal, "Generating a condensed representation for association rules," *Journal Intelligent Information Systems*, vol. 24, no. 1, pp. 29–60, 2005.

[19] A. Bykowski and C. Rigotti, "A condensed representation to find frequent patterns," in *Proceedings of the Twentieth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, 2001.

[20] P. B. Volk, F. Rosenthal, M. Hahmann, D. Habich, and W. Lehner, "Clustering uncertain data with possible worlds," in *Proceedings of the 25th International Conference on Data Engineering*, 2009, pp. 1625–1632. [Online]. Available: http://dx.doi.org/10.1109/ICDE.2009.174

[21] P. Parchas, F. Gullo, D. Papadias, and F. Bonchi, "The pursuit of a good possible world: extracting representative instances of uncertain graphs," in *Proceedings of the 2014 International Conference on Management of Data*, 2014, pp. 967–978. [Online]. Available: http://doi.acm.org/10.1145/2588555.2593668

[22] Z. Zhao, D. Yan, and W. Ng, "Mining probabilistically frequent sequential patterns in large uncertain databases," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 5, pp. 1171–1184, 2014. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/TKDE.2013.124

[23] L. Sun, R. Cheng, D. W. Cheung, and J. Cheng, "Mining uncertain data with probabilistic guarantees," in *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2010, pp. 273–282. [Online]. Available: http://doi.acm.org/10.1145/1835804.1835841

[24] E. Wilkins and S. H. Lavington, "Belief functions and the possible worlds paradigm," *J. Log. Comput.*, vol. 12, no. 3, pp. 475–495, 2002. [Online]. Available: http://dx.doi.org/10.1093/logcom/12.3.475

[25] L. Gamut, *Logic, Language, and Meaning: Intensional logic and logical grammar*, ser. Logic, Language, and Meaning. University of Chicago Press, 1991. [Online]. Available: http://books.google.de/books?id=ktqxlzcc5nQC

[26] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics Bulletin*, vol. 1, no. 6, pp. 80–83, Dec. 1945. [Online]. Available: http://dx.doi.org/10.2307/3001968

[27] H. B. Mann and D. R. Whitney, "On a test of whether one of two random variables is stochastically larger than the other," *The Annals of Mathematical Statistics*, vol. 18, no. 1, pp. 50–60, 03 1947. [Online]. Available: http://dx.doi.org/10.1214/aoms/1177730491

[28] D. Moore, G. McCabe, and B. Craig, *Introduction to the Practice of Statistics*. W.H. Freeman, 2009. [Online]. Available: http://books.google.de/books?id=x0kkSwAACAAJ

[29] P. Domingos and G. Hulten, "Mining high-speed data streams," in *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2000, pp. 71–80.

[30] T. M. Cover and J. A. Thomas, *Elements of information theory (2. ed.)*. Wiley, 2006.

[31] J. Gama and C. Pinto, "Discretization from data streams: applications to histograms and data mining," in *Proceedings of the 2006 ACM symposium on Applied computing*, 2006, pp. 662–667.

[32] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to information retrieval*. Cambridge University Press, 2008.

[33] M. Scutari, "Learning Bayesian networks with the bnlearn R package," *Journal of Statistical Software*, vol. 35, no. 3, pp. 1–22, 2010.

[34] G. Melançon and F. Philippe, "Generating connected acyclic digraphs uniformly at random," *Inf. Process. Lett.*, vol. 90, no. 4, pp. 209–213, 2004.

[35] T. Schlüter and S. Conrad, "Hidden markov model-based time series prediction using motifs for detecting inter-time-serial correlations," in *Proceedings of the 2012 ACM Symposium on Applied Computing*, 2012, pp. 158–164. [Online]. Available: http://doi.acm.org/10.1145/2245276.2245308