Annals of the Institute of Statistical Mathematics (2020) 72:895–913
https://doi.org/10.1007/s10463-019-00714-6

Distributions of pattern statistics in sparse Markov models

Donald E. K. Martin[1]

Received: 28 July 2018 / Revised: 27 January 2019 / Published online: 5 April 2019

Abstract
Markov models provide a good approximation to probabilities associated with many categorical time series, and thus they are applied extensively. However, a major drawback associated with them is that the number of model parameters grows exponentially in the order of the model, and thus only very low-order models are considered in applications. Another drawback is lack of flexibility, in that Markov models give relatively few choices for the number of model parameters. Sparse Markov models are Markov models with conditioning histories that are grouped into classes such that the conditional probability distribution for members of each class is constant. The model gives a better handling of the trade-off between bias associated with having too few model parameters and variance from having too many. In this paper, methodology for efficient computation of pattern distributions through Markov chains with minimal state spaces is extended to the sparse Markov framework.

Keywords Auxiliary Markov chain · Pattern distribution · Sparse Markov model · Variable length Markov chain

1 Introduction

Mining data for exceptional patterns can lead to important discoveries, such as sites on DNA sequences that are recognized by various agents, locations of an outbreak of an epidemic, changes in a production process, and similar segments of biological sequences. In using a pattern statistic for detection of special segments, one models the data sequence under a null hypothesis that characterizes what is typical, and uses the model to quantify exceptionality of the statistic (through the associated p value). Pattern statistics and their distributions may also be used to summarize important properties of a sequence. A method is therefore needed to obtain their distributions.

✉ Donald E. K. Martin
demarti4@ncsu.edu

[1] Department of Statistics, North Carolina State University, 4272 SAS Hall, 2311 Stinson Drive, Raleigh, NC 27695-8203, USA

Whereas Monte Carlo methods may be used to approximate probabilities, large numbers of replicates are typically required for accurate results, especially for events whose occurrence is rare. More accurate methods that are also less computationally expensive are desired.

A simple but powerful approach for computing exact distributions of pattern statistics in an mth-order discrete Markovian sequence $\mathbf{X} \equiv X_1, \ldots, X_n$ is to set up a correspondence between events related to the statistic of interest and events related to an auxiliary Markov chain (AMC) $\mathbf{Y} \equiv Y_1, \ldots, Y_n$. This allows using basic properties of Markov chains to compute the desired probabilities. The approach was forwarded in Brookner (1966) and became popular in more recent years with the work of Fu and Koutras (1994) and Koutras and Alexandrou (1995). In the present paper, the main focus is on extending Markov chain-based methodology for computing distributions of pattern statistics in categorical time series to more flexible and parsimonious models than are supplied by higher-order Markov chains.

1.1 Variable length Markov chains and sparse Markov models

Analysis of a categorical time series $\mathbf{X}$ taking values in a finite set Σ is facilitated by a model that captures the statistical properties of the sequence, while being simple enough so that accurate statistical analysis is feasible. In many cases, the analysis of such time series has been simplified by the sequence having a short memory or Markov property in the sense that the conditional probability of the current observation given the last m data points does not change by conditioning further into the past, i.e., for all $t \geq m + 1$,

$$\begin{aligned} &\Pr[X_t = x_t | X_{t-1} = x_{t-1}, \ldots, X_1 = x_1] \\ &\quad = \Pr[X_t = x_t | X_{t-1} = x_{t-1}, \ldots, X_{t-m} = x_{t-m}]. \end{aligned} \tag{1}$$

Sequences satisfying (1) are said to be mth-order Markovian. An mth-order Markovian sequence $\mathbf{X}$ may be embedded in a first-order Markov chain, with states that are the m-tuples $\tilde{x_t} \equiv (x_{t-m+1}, \ldots, x_t)$ of Σ^m, an initial distribution π over m-tuples and a $|\Sigma|^m \times |\Sigma|^m$ transition probability matrix T that has $|\Sigma|$ nonzero elements per row. ($|\cdot|$ denotes set cardinality or vector length, as appropriate.)

Markov models provide a good approximation to probabilities associated with many categorical time series, and thus, they are applied extensively. However, a major drawback associated with them is that the number of model parameters needed to specify transition probabilities is $|\Sigma|^m(|\Sigma| - 1)$. Therefore, only very low-order models are typically used in applications. Another drawback, as pointed out in Bühlmann and Wyner (1999), is lack of flexibility, in that Markov models give relatively few choices for the number of model parameters. For example, if $|\Sigma| = 4$, the number of parameters is 3×4^m, so that for $m \in \{0, 1, 2, 3, 4, 5, 6\}$ the number of parameters lies in the set $\{3, 12, 48, 192, 768, 3072, 12288\}$, but can take on no values in between. It would be useful to have a class of models that has more flexibility while being more parsimonious in terms of the number of parameters, allowing a better trade-off of the bias that arises from using models of an order lower than the true value, and variance that arises

from estimating many transition probabilities. *Variable length Markov chains* as well as the more general *sparse Markov models* have those properties. These models are defined next.

Definition 1 [*Sparse Markov Model*] (Jääskinen et al. 2014). Let $\mathbf{X}$ be a Markovian sequence of finite order m transformed into a first-order Markov chain. Define $p_{\cdot|k}$ to be a conditional probability distribution given context k and let $\Gamma = \{\gamma_1, \ldots, \gamma_\eta\}$ be a partition of Σ^m into probability equivalence classes such that the transition probability vectors satisfy $p_{\cdot|i} = p_{\cdot|j}$ for all pairs of conditioning contexts $\{i, j\} \in \gamma_l$, $l = 1, \ldots, \eta$. Also let P be the corresponding set of η conditional probability distributions on Σ^m. $\{\Gamma, P\}$ is called a *sparse Markov model (SMM)* (of order, or maximal context depth, m).

Remark 1 In essence, an SMM of order m is an mth-order Markovian sequence for which parameters associated with equal conditional distributions are lumped.

Definition 2 [*Variable length Markov chain*] A variable length Markov chain (VLMC) is a sparse Markov model such that for every $l = 1, \ldots, \eta$, each of the m-tuples of γ_l has a common suffix (see the proof in Jääskinen et al. 2014).

If l_{γ_j} denotes the length of the longest common suffix of strings of probability equivalence class γ_j, then a VLMC has $l_{\gamma_j} > 0 \; \forall \; j$.

In VLMC, the length of the context needed for conditional probabilities depends on the context itself. VLMCs were introduced in Rissanen (1983), who, in the setting of data compression, gave a "context algorithm" that generates a context tree representing segments of variable length needed for conditioning. The name derives from Bühlmann and Wyner (1999), who discussed the model from a more statistical point of view. VLMCs have $L(|\Sigma|-1)$ parameters for transition probabilities, where L denotes the number of leaf nodes in the corresponding context tree. That $L \in \{1, \ldots, |\Sigma|^m\}$ highlights the flexibility of the model. VLMCs are also called Markov sources (Rissanen 1986; Roos and Yu 2009), finite-memory sources (Weinberger et al. 1992, 1995), variable-order Markov (VOM) models (Begleiter et al. 2004), probabilistic suffix trees (Gabadinho and Ritschard 2016), and probabilistic suffix automata (Ron et al. 1996). In addition to information theory, the model has been studied by statisticians, computer scientists, and computational biologists, among others, and applied to statistical applications such as classification (Shmilovici and Ben-gal 2007), prediction (Begleiter et al. 2004), statistical process control (Ben-gal et al. 2003), linguistics (Galves et al. 2012; Gallo and Leonardi 2015; Belloni and Oliveira 2017), spam filtering (Bratko et al. 2006), web navigation (Borges and Levene 2007), and in general as a tool for exploratory data analysis of categorical time series, including DNA nucleotides and other biological sequences (Browning 2006; Bercovici et al. 2012), binary times series from computer science or information theory (Willems et al. 1995), or text (Ron et al. 1996).

The more general model has been considered in very recent years under the names *minimal Markov models* (García and González-López 2010), *sparse Markov chains* (Jääskinen et al. 2014), *sparse Markov models* (Xiong et al. 2016), and *partition Markov models* (García and González-López 2017; Fernández et al. 2018). Whereas

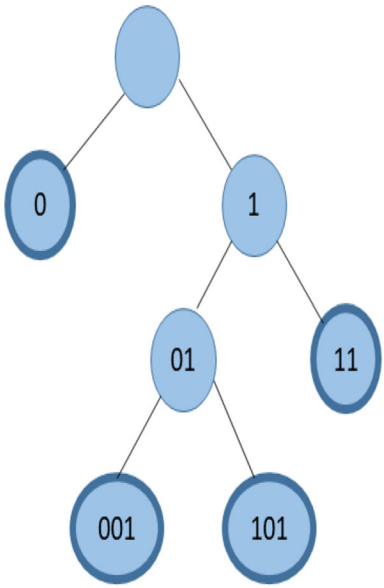

Fig. 1 Context tree corresponding to the variable length Markov chain of Example 1.1. Leaf nodes corresponding to the contexts needed for conditioning have a bold outline

VLMC have the restriction that grouped contexts must have a common suffix, by eliminating that restriction SMM allow a wider variety of models.

Example 1 Consider a sparse Markov model of order $m = 3$, with partition $\Gamma = \{\gamma_1, \ldots, \gamma_4\} = \{\{000, 100, 010, 110\}, \{011, 111\}, \{001\}, \{101\}\}$ for binary $\mathbf{X}$. The model is a VLMC with context tree shown in Fig. 1. Here there are $L = 4$ contexts corresponding to the classes of Γ and represented by the leaf nodes of the graph [instead of the eight possible 3-tuples representing $(x_{t-3}, x_{t-2}, x_{t-1})$]. The leaf nodes give the common suffixes of 3-tuples in the various classes.

Example 2 Let $\mathbf{X}$ be a VLMC of order $m = 4$, with $\Sigma = \{0, 1\}$ and context tree shown in Fig. 2. The five contexts are represented by the leaf nodes of the graph. The model corresponds to the partition Γ of 4-tuples into the classes

$$\Gamma = \{\{0000, 0010, 0100, 0110, 1000, 1010, 1100, 1110\}, \{0001, 0101, 1001, 1101\}, \{0011, 1011\}, \{0111\}, \{1111\}\}.$$

These classes correspond to the number of consecutive 1's at the end of strings. For an mth-order model of this form, the number of leaf nodes in the VLMC would be $L = m + 1$ as opposed to 2^m, a significant reduction in the number of parameters and associated contexts.

Example 3 Let X be a second-order Markovian sequence with $\Sigma = \{A, C, G, T\}$, for which the conditional distribution is constant over the classes

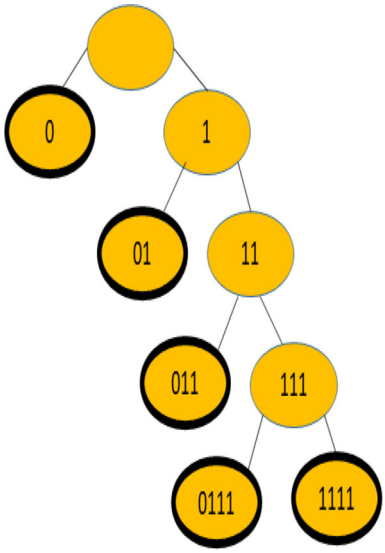

Fig. 2 Context tree of the variable length Markov chain of Example 1.2. Nodes with a bold outline are the contexts of the model and correspond to the number of 1's at the end of the sequence

$$\Gamma = \{\{AA, AC, AG, AT\}, \{CA, CC, CG, CT\}, \{GA, GC, GG, GT\}, \{TA, TC, TG, TT\}\}.$$

The classes γ_l, $l = 1, \ldots, 4$ are the partition of the set of 2-tuples into those with a fixed value of X_{t-2} and an arbitrary value of X_{t-1}. Since the elements of the γ_l do not have a common suffix, the model may not be represented as the context tree of a VLMC, but is a sparse Markov model.

In the next section, as background we give highlights of previous work on the computation of distributions of pattern statistics in mth-order Markovian sequences. Section 3 gives the desired extension to SMM, with an application to spaced seed coverage being presented in Sect. 4. The final section concludes the paper.

2 Distributions of pattern statistics in higher-order Markovian sequences

A *pattern* is a finite string of symbols from Σ. For pattern $w = (w_1, \ldots, w_{|w|})$, $u = (w_1, \ldots, w_h)$, $1 \le h \le |w|$ is a *prefix* of w (a *proper prefix* if $h < |w|$), and $v = (w_j, \ldots, w_{|w|})$, $1 \le j \le |w|$ is a *suffix* of w (a *proper suffix* if $j > 1$). Let W be a collection of patterns that occur in $\mathbf{X}$ and Z a pattern statistic related to W, taking values z in a finite set Φ. For convenience, it will be assumed that all patterns of W have length greater than m; otherwise, minor adjustments can be made. Also, $u \cdot v$ denotes the concatenation of strings u and v. For example, if $u = 11$ and $v = 011$, then $u \cdot v = 11011$. We have, for $z \in \Phi$,

$$\Pr[Z = z] = \sum_{\mathbf{x}:Z(\mathbf{x})=z} \Pr[\mathbf{X} = \mathbf{x}], \tag{2}$$

i.e., $\Pr[Z = z]$ is the sum of probabilities of sequences $\mathbf{x}$ that are mapped by Z into z. However, examining each of the $|\Sigma|^n$ possible sequences $\mathbf{x}$ to carry out the computation is typically not feasible unless it is carried out in an efficient manner.

Using an auxiliary Markov chain (AMC) for computation has provided a way to make the computation feasible when $\mathbf{X}$ is mth-order Markovian, which is the assumption of this section. In Markov chain-based computation, an AMC $\mathbf{Y}$ is set up so that $\Pr[Z = z] = \Pr[Y_n \in \Upsilon_z]$, where $\{\Upsilon_z, z \in \Phi\}$ forms a partition of the state space Υ of $\mathbf{Y}$. Let α_m denote the initial distribution of $\mathbf{Y}$ at time m and Ω its transition probability matrix (which are determined from initial distribution π and transition probability matrix T for $\mathbf{X}$). Then row vectors $\alpha_t, t = m, \ldots, n$ holding probabilities for the AMC lying in the states of Υ may be computed through $\alpha_n = \alpha_m \Omega^{n-m}$, which, beginning with α_m, may be obtained recursively using $\alpha_{t+1} = \alpha_t \Omega$, $t = m, m+1, \ldots, n-1$. $\Pr[Z = z]$ may then be obtained by extracting the probabilities associated with Υ_z from α_n.

Because $\mathbf{Y}$ is to be a Markov chain, its states Υ must include sufficient information so that the transition probabilities of Ω may be computed (the last m observations are needed for mth-order Markovian $\mathbf{X}$), and states must contain information concerning progress toward observing the patterns of interest. Note that the current value of the statistic Z must be kept in some form, though not necessarily in states of Υ. While maintaining the Markov property for $\mathbf{Y}$ can render $|\Upsilon|$ to be large, since the order of Ω is $|\Upsilon|$, keeping $|\Upsilon|$ small is important for computation, especially for the complex pattern types and pattern lengths that arise in practice. An example pattern type follows.

Example 4 Spaced seeds (Ma et al. 2002) are used in an initial filtering step to help with the trade-off between attempting too many costly DNA sequence alignments and missing similar DNA segments by not aligning them. A spaced seed is a pattern $S = s_1, \ldots, s_k$ from $\{1, *\}$, with $s_1 = s_k = 1$, where 1 indicates a match position and * a wildcard position that does not have to match. Let $\mathbf{X}$ be the binary sequence formed by aligning two DNA segments of length n and assigning a value $X_j = 1$ if the jth position of the segments match, and 0 otherwise. Spaced seed S *hits* or occurs in $\mathbf{X}$ at position v if for $j = 1, \ldots, k$, $X_{v-k+j} = 1$ whenever $s_j = 1$. In an occurrence, a 1 at position X_{v-k+j} corresponding to $s_j = 1$ is said to be *covered*.

In typical studies using spaced seeds (see, e.g., Ma et al. 2002; Keich et al. 2004; Buhler et al. 2005), a full alignment is triggered when the seed hits in $\mathbf{X}$. The probability of a single hit (the seed's *sensitivity*) helps to determine a seed that differentiates between random and meaningful similar segments. Computing a spaced seed's sensitivity is relatively easy. Spaced seed coverage, the number of covered positions in $\mathbf{X}$, may also be used as a criterion to trigger a full alignment based on multiple hits in a region. However, overlapping and non-overlapping seed hits must be distinguished when setting up Υ because the update to coverage on a seed hit depends on which positions were previously covered. The many combinations of prefixes of patterns representing overlapping spaced seed hits and positions that may be covered (or not) can render the number of states of an AMC to be extremely large, and the computation

to be much more difficult than for sensitivity. Thus, there are relatively few studies where seed coverage is used as a criterion (see Benson and Mak 2009; Noé and Martin 2014; Noé 2017; Martin and Noé 2017; Martin 2018). The coverage of spaced seeds will be revisited in Sect. 4. The efficient setup of Υ to handle such applications is discussed next.

2.1 Setup of AMC states

The state space Υ of AMC $\mathbf{Y}$ used in Fu and Koutras (1994) consisted of the same set of pattern–progress strings for each value z of the pattern statistic, a setup that can lead to an extremely large number of states. Koutras and Alexandrou (1995) and Aston and Martin (2007) attempted to get past this by restricting Υ to contain only pattern–progress/m-tuple strings, while using another mechanism to keep track of the value of statistic Z. Probability matrices $\Psi_t, t = m, \ldots, n$ were used in the latter paper instead of probability vectors α_t, with the rows of the Ψ_t corresponding to values of Z, and the columns to pattern–progress/m-tuple strings (along with other information, if necessary). However, for complex pattern statistics, $|\Upsilon|$ can still be extremely large.

To help reduce $|\Upsilon|$, several researchers (see, e.g., Lladser 2007; Lladser et al. 2008; Ribeca and Raineri 2008; Marshall and Rahmann 2008) applied computer science theory related to minimizing a deterministic finite automaton (DFA) to minimize $|\Upsilon|$. In that work, after forming a state space consisting of all m-tuples and pattern–progress strings, an algorithm such as the Hopcroft (1971) algorithm was applied to determine classes of states that are equivalent for computing the distribution of the pattern statistic, with only one state representing each equivalence class being retained in Υ. Equivalent states q and q' (in the context of computing the distribution of a pattern statistic) are such that concatenating an arbitrary string r to form $q \cdot r$ and $q' \cdot r$ gives the same update in terms of both the probability and the value of the pattern statistic. The equivalence of states q and q' will be denoted by $q \sim q'$.

While the application of DFA minimization represents a tremendous breakthrough in terms of setting up an AMC with a smaller state space, that approach suffers from the need to first set up a large state space before applying a minimization algorithm to reduce it. Nuel (2008) got around this problem by setting up a non-deterministic finite automaton that greatly reduces the size of the original state space for patterns with many wildcard positions before setting up a minimal DFA. Martin (2018) developed a characterization of equivalent states so that extraneous ones may be deleted during the process of forming the AMC state space. Thus, no extraneous states enter the state space at any stage. The methodology of that paper for developing minimal state spaces will be outlined next.

Let $Q = \tilde{Q} \cup \Sigma^m$, where $\tilde{Q}$ consists of proper prefixes of patterns of W that are of length at least m and Σ^m denotes the set of all m-tuples. Q could correspond to Υ, but that is not necessarily the case, as it could be that additional information is needed in AMC states (as is the case with spaced seed coverage; see Example 4 and Sect. 4). The longest proper suffix of a string $q \in Q$ that is itself in Q is called the *failure state* of q (Aho and Corasick 1975) and is denoted by $fl(q)$. By definition, strings $q \in \Sigma^m$ do not have failure states. Martin (2018) defined the *failure sequence* associated with q to consist of q and its sequence of failure states of decreasing lengths:

$$fs_q = \left(fs_{q,1}, \ldots, fs_{q,|fs_q|}\right) \equiv (q, fl(q), fl(fl(q)), \ldots, (q)_m).$$

Here $(q)_l$ denotes the l-tuple at the end of q. Note that $(q)_m$ may or may not be a pattern prefix. Also defined was

$$\tilde{C}_q = \left\{v : \exists u \in fs_q \text{ such that } u \cdot v = w \in W; |v| < |w|\right\}$$
$$\cup \{v : \exists l \text{ such that } 1 \leq l < m \text{ and } (q)_l \cdot v = w \in W\}.$$

$\tilde{C}_q$ contains *completion strings* for prefix strings of fs_q and suffixes of q of length less than m that are pattern prefixes. The occurrence of strings of $\tilde{C}_q$ may or may not require an update to the pattern statistic Z. (An example of when an occurrence of a pattern does not require the update of the pattern statistic is as follows. Let Z be the indicator of whether or not W occurs in a sequence. If W has already occurred, Z remains 1 with any additional occurrences, and thus the statistic is not updated.) The set of completion strings whose occurrence does require an update to Z is denoted by C_q. Also, define the *direct occurrence* of a $q \in \tilde{Q}$ to be the occurrence of $q \cdot v$, where v is the completion string of q. The transition function for string $q \in Q$ on symbol $x \in \Sigma$ is defined by $\delta(q, x) \equiv$ the longest suffix of $q \cdot x$ that is in Q. Thus either $\delta(q, x) = q \cdot x$, or $\delta(q, x) = fl(q \cdot x)$.

Martin (2018) proved two main theorems, along with a corollary to the second one. The first theorem showed that failure states may be obtained using the transition function δ, which helped with sequentially (over string lengths) obtaining failure states. The second theorem, along with its corollary, gives necessary and sufficient conditions for strings to be equivalent, so that equivalent strings may be determined and combined during the setup of the state space Υ.

Theorem 1 *If* $q \in \Sigma^m$, $fl(q \cdot x) = (q \cdot x)_m$. *For* $q \in Q \backslash \Sigma^m$, $fl(q \cdot x) = \delta(fl(q), x)$.

Theorem 2 *For mth-order Markovian sequence* $\mathbf{X}$, $q \sim q'$ *if and only if* $(q)_m = (q')_m$ *and the updates to the statistic's value are exactly the same when any string of* $C_q \cup C_{q'}$ *occurs.*

Corollary 1 *For statistics and pattern counting techniques such that the direct occurrence of* q *does not preclude updating the statistic on the occurrence of the strings of* C_{fl_q}, *if* $|fs_q| > 1$ *and* $|fs_{q'}| > 1$, $q \sim q'$ *if and only if* $fl(q) \sim fl(q')$ *and the updates to the statistic are exactly the same on the direct occurrences of* q *and* q'.

3 Distributions of pattern statistics in SMM

In this section, the methodology for computing distributions of pattern statistics in a stationary mth-order Markovian sequence $\mathbf{X}$ will be extended to SMM. This will allow not only the use of the advantages of SMM for flexible and parsimonious modeling of categorical time series, but also the use of minimization techniques of Martin (2018) to keep the AMC state space as small as possible.

3.1 Setting up the state space Υ

A knowledge of the last m observations at each time point gives sufficient information so that conditional probabilities of a sparse Markov model may be computed. It is then tempting to define the AMC state space Υ to be the state space used for the mth-order Markovian case. However, that approach would ignore the possibility that equivalent m-tuples could be combined, resulting in a smaller state space. Thus as an initial step to set up Υ, conditions for equivalent m-tuples are determined, followed by conditions for longer strings. Those conditions are given next.

Theorem 3 *Strings q and $q^{'}$ are equivalent if and only if they have the same updates to Z on the occurrence of all strings of $C_q \cup C_{q^{'}}$, and in addition*

(i) *If $|q| = |q^{'}| = m$, they lie in the same class γ_j of Γ and when concatenating an arbitrary string r satisfying $|r| \in \{1, \ldots, m - 1 - l_{\gamma_j}\}$, $(q \cdot r)_m$ and $(q^{'} \cdot r)_m$ lie in the same probability equivalence class γ_l for some l.*
(ii) *For $|q|$ and $|q^{'}| > m$, $(q)_m \sim (q^{'})_m$.*

Recall that $l_{\gamma_j} \geq 0$ is the longest common suffix of m-tuples in probability equivalence class γ_j.

Proof The necessity of the conditions is clear, as updates to the statistic and conditional probabilities upon concatenation of an arbitrary string must be exactly the same. To show sufficiency, equality of updates to Z on the occurrence of completion strings implies that all updates to Z must be the same, since the completion strings of C_q and $C_{q^{'}}$ are the proper suffixes of q where the statistic is updated. Any other subsequent pattern occurrences must have a pattern as a suffix of the concatenated string r. The other conditions ensure equivalence of conditional probabilities beginning in the states when concatenating arbitrary strings, as for $|r| \geq m - l_{\gamma_j}$, the resulting strings after concatenation will necessarily have the same suffix of length m. □

Corollary 2 *For statistics and pattern counting techniques such that the direct occurrence of q ($q^{'}$) does not preclude updating the statistic on the occurrence of the strings of $C_{fl(q)}$ ($C_{fl(q^{'})}$), if $|q| > m$ and $|q^{'}| > m$, $q \sim q^{'}$ if and only if $fl(q) \sim fl(q^{'})$ and the updates to the statistic are exactly the same on the direct occurrences of q and $q^{'}$.*

Proof Let $|q| > m$ and $|q^{'}| > m$. If $q \sim q^{'}$, by Theorem 3, $(q)_m \sim (q^{'})_m$ and the updates to the value of Z are exactly the same when any string of $C_q \cup C_{q^{'}}$ occurs, which includes q and $q^{'}$. Since $(q)_m = (fl(q))_m$ and $(q^{'})_m = (fl(q^{'}))_m$, $(q)_m \sim (q^{'})_m$ implies that $(fl(q))_m \sim (fl(q^{'}))_m$. Note also that $fs_{fl(q)}$ is a subsequence of fs_q, and similarly for $fs_{fl(q^{'})}$ and $fs_{fl(q^{'})}$. That, along with the assumption that we are dealing with statistics and pattern counting techniques such that the direct occurrence of q does not preclude updating the statistic on the direct occurrence of strings of $C_{fl(q)}$ and $C_{fl(q^{'})}$ implies that the updates to the statistic's value are exactly the same when any completion string of $C_{fl(q)} \cup C_{fl(q^{'})}$ occurs, and thus $fl(q) \sim fl(q^{'})$.

Conversely, let $fl(q) \sim fl(q^{'})$ with the updates to Z being the same on direct hits of q and $q^{'}$. Then $(fl(q))_m \sim (fl(q^{'}))_m$, which implies that $(q)_m \sim (q^{'})_m$ using the

same reasoning as above. Also since by assumption we have the same updates on the occurrence of all elements of $C_{fl(q)} \cup C_{fl(q')}$ as well as on the direct occurrences of q and q', we have the same updates on all elements of $C_q \cup C_{q'}$, so that $q \sim q'$. □

Theorem 3 leads to the following algorithm for setting up Υ and computing the distribution of pattern statistic Z in the case of an SMM.

Algorithm Given sparse Markov model $\{\Gamma, P\}$ with maximal context depth m, the distribution of Z may be obtained as follows:

(i) Refine probability equivalence classes γ_j, $j = 1, \ldots, \eta$ based on updates to Z. To do this, determine suffixes of m-tuples that are pattern prefixes and their completion strings. Check to see that m-tuples are completed on the same strings with the same update to Z, and separate ones that are not. A separated m-tuple will either be placed into a new class that was previously formed from γ_j (if it has the same completion strings and update to Z as strings in that class), or into an additional new class for which it is the only member.

(ii) Refine probability equivalence classes (and any new classes formed by refining them) so that on concatenation of arbitrary strings r of sequentially increasing lengths $h = 1, \ldots, m - 1 - l_{\gamma_j}$, $(q \cdot r)_m$ and $(q' \cdot r)_m$ lie in the same class. First note that strings with the same $(m-1)$-tuple suffix automatically have the same m-tuple suffix on concatenation of a non-empty string. Now for each class γ_j, determine the destination of its elements on the concatenation of a single symbol from Σ. This is carried out by determining the suffix of length $m-1$ of each m-tuple of γ_j. This suffix will be the prefix of the $|\Sigma|$ m-tuples that will be reached after concatenating a symbol. Compare the class of these $|\Sigma|$ m-tuple destinations for each m-tuple of γ_j, forming new classes as needed (see Example 5). The process of checking destinations for m-tuples on concatenation of a single symbol is repeated until no new classes are formed, as with new classes, transitions may be changed. It suffices to only check the concatenation of a single symbol, for once there is no further splitting of any class in that case, induction implies that there will be no further splits when concatenating multiple symbols.

(iii) Revise matrix T based on the final grouping of m-tuples, calling the reduced matrix $\check{T}$. Also, combine initial probabilities for grouped m-tuples in a single element of new initial vector $\check{\pi}$. In the stationary case, solve for $\check{\pi}$ using $\check{\pi}\check{T} = \check{\pi}$.

(iv) Set up the $|\Phi| \times |\Upsilon|$ initial probability matrix Ψ_m for time m. Its nonzero initial probabilities are given by $\Psi_{m,(i_j,j)} = \check{\pi}(j)$, where i_j is the row corresponding to the value of z for state j of Υ. Due to the assumption that patterns of W have lengths greater than m, $i_j = 1$ (corresponding to $Z = 0$) for all j, but in general, that doesn't have to be the case.

(v) To form the pattern prefixes, determine the destination state for strings of Υ on the concatenation of symbols $x \in \Sigma$, beginning with the m-tuples that are pattern prefixes. If $q \cdot x \in \tilde{Q}$, use Theorem 3 to determine whether there is an equivalent state already in the state space. If there is, map the transition to that state, but if not, form a new state. If $q \cdot x \notin \tilde{Q}$, the transition is obtained using $\delta(q, x) = fl(q \cdot x)$. After the pattern prefixes are formed, states are added

(as needed) using transition function δ and other transition related rules that are required, with equivalent states determined using Theorem 3. Repeat the process of determining new states and their transitions for all new states formed at the last stage, terminating if there are no new states.

(vi) At each stage, obtain failure states using $fl(q \cdot x) = \delta(fl(q), x)$.

(vii) At each stage, the transition probabilities are given by $\Pr[q \to q'] = \Pr[(q)_m \to (q')_m]$, where transition probabilities for m-tuples are taken from matrix $\check{T}$. These probabilities are recorded in Ω.

(viii) After obtaining the states of Υ and their transition probabilities, for $t = m, \ldots, n-1$, right multiply Ψ_t by Ω and move transition probabilities for entering a "counting state" (a state for which the value of Z should be updated) to the appropriate row. This gives Ψ_{t+1}.

(ix) Extract probabilities from Ψ_n corresponding to Υ_z to obtain $\Pr[Z = z]$.

We now give examples of determining equivalent m-tuples for the partitions of Examples 1–3.

Example 5 Let $\Gamma = \{\gamma_1, \ldots, \gamma_4\} = \{\{000, 100, 010, 110\}, \{011, 111\}, \{001\}, \{101\}\}$ for binary $\mathbf{X}$, with $Pr[X_t = 1|\gamma_j]$, $j = 1, 2, 3, 4$ respectively equal to (0.6,0.8,0.65,0.7), and with Z being the number of overlapping occurrences of $W = \{11111\}$. The class $\{011, 111\}$ would be split since string 111 is completed by 11, but string 11 is not. This gives new class $\gamma_5 = \{111\}$.

With this split, only probability equivalence class γ_1 can possibly be split further. To determine the result after concatenating a single symbol to m-tuples of γ_1, note that for m-tuples in this class, two of the suffixes of length two are 00 and the other two are 10. Thus two transition to 000 and 001 (which are in classes 1 and 3), while the other two transition to 100 and 101 (which are in classes 1 and 4). Thus we split these strings, leaving $\gamma_1 = \{000, 100\}$, and forming new class $\gamma_6 = \{010, 110\}$. The final classes are $\{\gamma_1, \ldots, \gamma_6\} = \{\{000, 100\}, \{011\}, \{001\}, \{101\}, \{111\}, \{010, 110\}\}$, and the transition probability matrix $\check{T}$, using this order, is then

$$\check{T} = \begin{pmatrix} 0.4 & 0 & 0.6 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.8 & 0.2 \\ 0 & 0.65 & 0 & 0 & 0 & 0.35 \\ 0 & 0.7 & 0 & 0 & 0 & 0.3 \\ 0 & 0 & 0 & 0 & 0.8 & 0.2 \\ 0.4 & 0 & 0 & 0.6 & 0 & 0 \end{pmatrix}.$$

Solving $\check{\pi}\check{T} = \check{\pi}$ gives initial probability vector

$$\check{\pi} = \frac{1}{455}[50, 51, 30, 45, 204, 75].$$

Example 6 Since the probability equivalence classes

$$\Gamma = \{\{0000, 0010, 0100, 0110, 1000, 1010, 1100, 1110\}, \{0001, 0101, 1001, 1101\}, \{0011, 1011\}, \{0111\}, \{1111\}\}$$

correspond to the number of consecutive 1's at the end of strings, there will be no splitting based on transitions of m-tuples, as the number of 1's at the end of strings in the same probability equivalence class will remain the same after concatenation of an arbitrary string. If Z is the number of overlapping occurrences of $W = \{11111\}$, the classes would remain intact as the completion strings of all strings in the same class are equal. For other patterns, however, that may not be the case. For example, if Z is the number of overlapping occurrences of pattern 10111, m-tuples 1101 and 0101 of γ_2 both have completion strings 11 and 0111, whereas 0001 and 1001 have lone completion string 0111. Thus γ_2 would be split into two classes.

Example 7 For $\Gamma = \{\{AA, AC, AG, AT\}, \{CA, CC, CG, CT\}, \{GA, GC, GG, GT\}, \{TA, TC, TG, TT\}\}$ with $\Sigma = \{A, C, G, T\}$, all classes would be split into four separate 2-tuples since a transition on a single symbol would map the four members of each class into the four different classes $\gamma_1, \ldots, \gamma_4$. For example, after concatenating symbol A to string AA, the 2-tuple suffix is $AA \in \gamma_1$, whereas concatenating symbol A to string AC gives 2-tuple suffix $CA \in \gamma_2$. Thus, in the end, the state space Υ would be the same as for a second-order Markov chain.

4 Application to spaced seed coverage

Consider now a spaced seed $S = s_1, \ldots, s_k$ with $k > m$, and let W be the set of 2^r patterns obtained by replacing the r stars of the seed by either 0 or 1. The binary sequence $\mathbf{X}$ that gives indicators of matching positions in the two DNA segments is assumed to be stationary and to follow an SMM of maximal depth m. Recall that when reckoning spaced seed coverage in $\mathbf{X}$, a "1" in a seed hit can only be counted once, so that overlapping patterns must be differentiated from non-overlapping ones.

One option for forming an AMC state space Υ is to use prefix strings of the set W_{ext} that contains the extension of patterns of W to all possible overlapping pattern occurrences (Martin and Coleman 2011). W_{ext} is defined by

$$\begin{aligned} W_{\text{ext}} = \{u : \exists w_1, w_2 \in W \text{ such that } u = \nu_{w_1} \cdot \alpha_{w_1 w_2} \cdot \beta_{w_2}, \\ \text{where } \nu_{w_1} \cdot \alpha_{w_1 w_2} = w_1, \alpha_{w_1 w_2} \cdot \beta_{w_2} = w_2, \\ \text{and } \nu_{w_1} \text{ and } \beta_{w_2} \text{ are non-empty}\}. \end{aligned}$$

Here $\alpha_{w_1 w_2}$ is a suffix of w_1 and prefix of w_2, the overlapping portion of the two patterns. As all spaced seeds begin and end with 1, $\alpha_{w_1 w_2}$ is guaranteed to be non-empty. While an option, a representation based on the extended strings of W_{ext} is less than optimal, as it uses excessive storage locations, and increases the difficulty in locating equivalent strings. A more storage friendly representation is established in the following manner. Let $Q = \tilde{Q} \cup \Sigma^m$, as in Sect. 2. Then use strings of Q to represent progress toward patterns, while marking covered positions. An example of

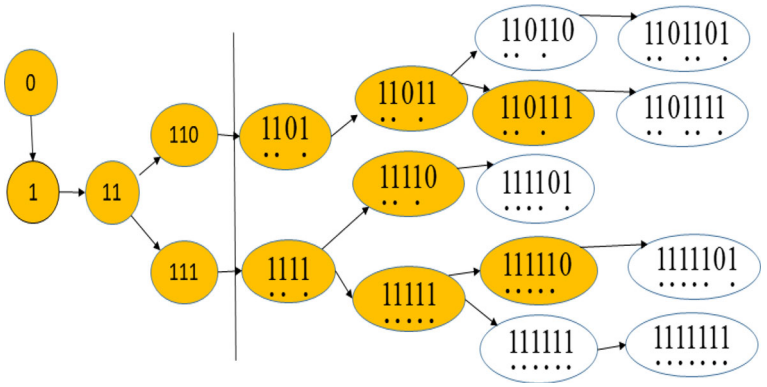

Fig. 3 AMC state space for spaced seed 11*1 using prefixes of extended set W_{ext}. States that are not needed in the minimal state space have white color

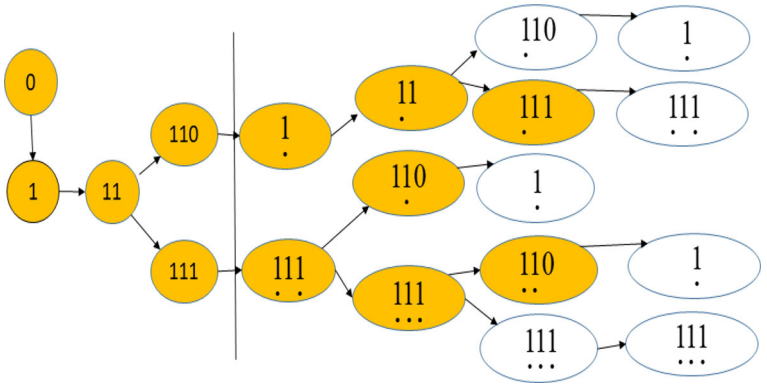

Fig. 4 AMC state space for spaced seed 11*1 with covered positions marked with a dot. States that are not in the minimal state space have white color

the representation based on W_{ext} and based on Q with marked coverage is shown in Figs. 3 and 4 for the small spaced seed 11*1. In those plots, the seven states in white are not needed in the computation. The representation based on Q makes it clear that six of these seven states are not needed, as they have exactly the same Q string and covered positions as strings that would already have been placed in the state space.

Instead of considering updates to Z on the occurrence of all completion strings of fs_q to determine equivalent strings, Martin and Noé (2017) only located equivalent strings q and q' that either have exactly the same coverage or whose failure states are empty, so that only equality of the update to coverage on the direct hits of q and q' needed to be checked. (The latter is the case for seed 11*1 and states 110 and 110 of Fig. 4. The failure state of 110 is empty, and the updates on the direct occurrences of both q and q' are two, so that the states are equivalent.) Martin (2018) extended the methods of the latter paper to find all equivalent states using complete failure sequences as described in Sect. 2. To illustrate the extension of

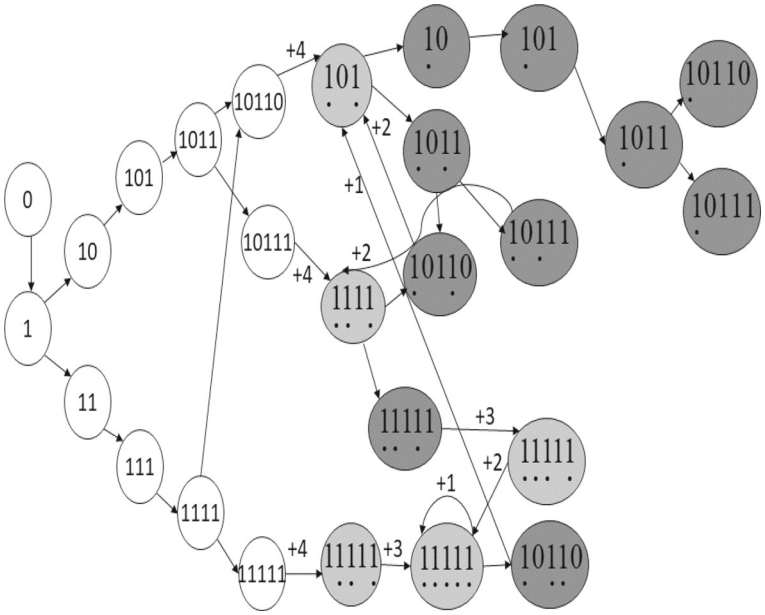

Fig. 5 States of the AMC for seed 1*11*1. The counting states whose entrance signals that the statistic is updated are colored in light gray, and other states with nonzero coverage in darker gray. Coverage updates on transitions into counting states are indicated and covered positions are marked with a dot. For clarity, not all transitions are shown

the method to SMM, we now consider the relatively short spaced seeds 1*11*1 and 111*1*1, and then the longer "Patternhunter seed" 111*1**1*1**11*111 (Ma et al. 2002) to show feasibility of the computation for a seed of a length used in practice. The model is fixed to have probability equivalence classes $\Gamma = \{\gamma_1, \ldots, \gamma_4\} = \{\{000, 100, 010, 110\}, \{011, 111\}, \{001\}, \{101\}\}$, with $Pr[X_t = 1|\gamma_j]$, $j = 1, 2, 3, 4$ respectively equal to (0.6,0.8,0.65,0.7). This VLMC model is input to the algorithm. The length of $\mathbf{X}$ is fixed at $n = 64$, the length considered in Ma et al. (2002).

Spaced seed $S = 1 * 11 * 1$ has $W = \{101101, 101111, 111101, 111111\}$. For this collection of patterns, m-tuples 011 and 111 have different completion strings, and thus $\gamma_2 = \{011, 111\}$ needs to be split. Also, whereas the elements of $\{000, 100\}$ have no pattern progress, those of $\{010, 110\}$ have pattern prefix 10 as their suffix, and thus γ_1 needs to be split into these two classes. Thus $\breve{T}$ and $\breve{\pi}$ are as in Example 5.

The state space Υ has the 26 states shown in Fig. 5. This is to be compared with the 84 states (not shown) that are obtained using prefixes of W_{ext}. In Fig. 5, only the 11 states in white are needed to compute the sensitivity of the seed (along with an absorbing state to indicate the seed's occurrence). The strings 10110 and 11110 are equivalent (from Corollary 2) for computing sensitivity or coverage, as they both have the same failure state, and the update to coverage is +4 when both strings are completed. Thus, on symbol 0, 1111 transitions to 10110, and 11110 is deleted. State 1111 transitions to 10110 on symbol 0 (the second position of the string is no longer marked as being covered since the position cannot possibly be involved in a seed hit

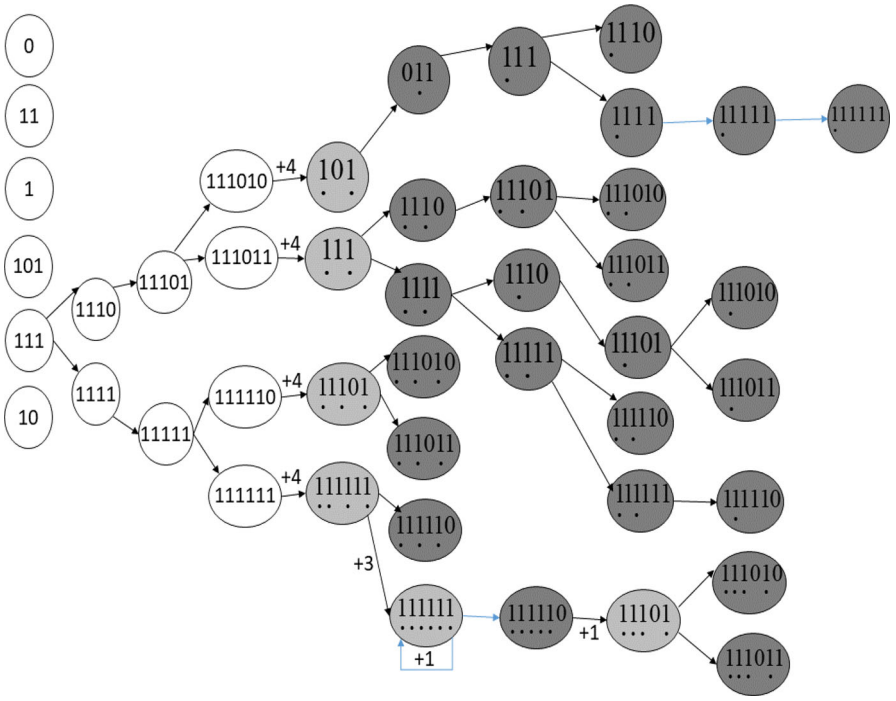

Fig. 6 States of the AMC for seed 111*1*1. The counting states whose entrance signals that the statistic is updated are colored in light gray, and other states with nonzero coverage in darker gray. Coverage on transitions into counting states is indicated and covered positions are marked with a dot. For clarity, not all transitions are shown

that occurs after the direct one). Similarly, state 11111 transitions to 10110 on symbol 0 (instead of 10110 since 10110 and 10110 are equivalent. The equivalence of these states may not be obvious, since for pattern prefix 10, the 1 is covered in one case and not in the other. However, the completion string of 10 is of the form 11*1. The occurrence of the first 1 of this string implies the direct hit of the longer string 10110 with a coverage update of +2 for both states, rendering all 1's as being covered so that the subsequent updates to the statistic must be the same.)

A MATLAB program was written to implement the algorithm to compute coverage of spaced seeds and run on a Dell PC with an Intel Core i7 CPU 873 with 2.93 GHz and 8 GB RAM. The total computation time for spaced seed $S = 1 * 11 * 1$ was about 0.5 s, 0.1 s to set up the state space, and 0.4 s to carry out the computation.

For spaced seed $S = 111 * 1 * 1$, $W = \{1110101, 1110111, 1111101, 1111111\}$. For this collection of patterns, none of the m-tuples {000, 100, 010, 110} have pattern progress, so that only 011 and 111 need to be split based on this criterion. As in Example 5, {000, 100} and {010, 110} need to be split based on their transitions on a single symbol. The final six classes, initial distribution and transition probability matrix is then the same as for the seed $1 * 11 * 1$. The state space Υ for this seed is shown in Fig. 6. One thing to note is that on symbol 0, state 101 transitions to 10 and

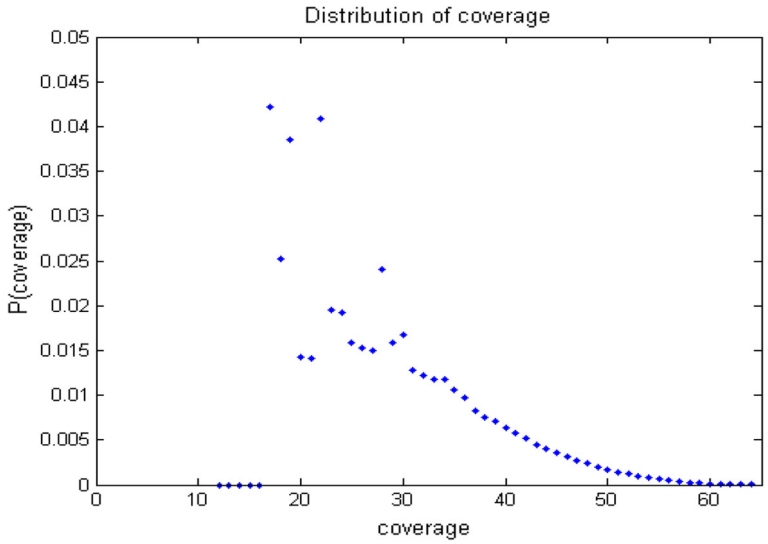

Fig. 7 Coverage distribution for seed 111*1**1*1**11*111 in a 3rd-order SMM with probability equivalence classes (000,010,100,110), (011,111), (001), (101), conditional probabilities given classes respectively given by (0.6,0.8,0.65,0.7), and $n = 64$

not 010. This is because 010 has no suffix that is a pattern prefix, and thus there is no need to indicate that its middle symbol has been covered. The total computation time for this seed was 1.2 s, with 0.3 s used to set up the state space, and 0.9 s to carry out the computation.

For the spaced seed 111*1**1*1**11*111 used in the original version of the Patternhunter software (Ma et al. 2002), $|\Upsilon| = 3815$, compared with the more than 320,000 strings that are obtained using prefixes of W_{ext}. The distribution of coverage is depicted in Fig. 7. The total computation time was about 40 s, 33.8 s to set up the state space, and 6.5 s to carry out the computation.

5 Conclusion

In this paper, Markov chain-based methodology for computing distributions of pattern statistics in mth-order Markovian sequences is extended to sparse Markov models. The models involve general partitions of the set of m-tuples, where m is the maximal depth of conditioning needed to compute conditional probabilities. For m-tuples in the same class of the partition, conditional probability distributions are exactly the same. The model allows one to use less parameters and affords more flexibility than higher-order Markov models, leading to better model fits.

The methodology for computing the distribution of a pattern statistic discussed in the paper involves setting up an auxiliary Markov chain so that events related to the pattern statistic correspond to events for the Markov chain. Properties of the Markov chain are then used to carry out the computation in a simple manner. However satisfying

the Markov property can mean that the state space of the AMC has to be very large. It has been shown that theory related to determining equivalent states can lead to a great reduction in the size of the state space of the AMC (and in fact can facilitate the setup of state spaces in cases that would not be possible otherwise; see Martin 2018 and the example on structured motifs). Using this theory then expands the application of Markov chain-based computation to more general pattern statistics.

The extension of the methodology to sparse Markov models as carried out in this paper allows the use of the techniques for keeping the state space small and also the improved modeling capabilities of sparse Markov models. This extension mainly involves minimizing the set of all m-tuples by determining those having the same conditional distributions and update to the pattern statistic after concatenating an arbitrary string. The lumping of m-tuples into equivalence classes can also result in the grouping of longer states.

5.1 Future work

Two areas of future work have been identified. Whereas it is assumed here that a model is given, model fitting for SMMs (determining a maximal depth m and a partition of m-tuples) is an important problem. The author seeks to add to and improve on the limited work in the literature on fitting SMMs (García and González-López 2010, 2017; Jääskinen et al. 2014; Xiong et al. 2016; Fernández et al. 2018).

Secondly, in Markov chain-based methods for computing pattern distributions, after defining an initial distribution, probabilities of lying in the AMC states at each time point are obtained by multiplying a probability matrix by the Markov chain's transition probability matrix. As the matrices in the procedure depend on probabilities, different input probabilities require the computation to be rerun. A more efficient method was used in Mak and Benson (2009), Benson and Mak (2009) and Noé (2017) for binary sequences. Instead of updating probabilities, counts of sequences in pattern–progress states with constant values of the number of successes and the statistic are updated. The rationale is that the counts themselves do not depend on input probabilities, and thus a set of probabilities may be input into an equation after counts are obtained, saving computation time by eliminating repeated computations. The equation may be obtained through the representation (2) for $\Pr[Z = z]$ and then partitioning $\Pr[\mathbf{X} = \mathbf{x}]$ based on values of sufficient statistics. Extensions of this work from independent trials to first- and second-order Markovian sequences as well as sparse Markov models are planned.

Acknowledgements This material is based upon work supported by the National Science Foundation under Grant No. 1811933. The author would like to thank the reviewer for their insightful comments on the original version of the manuscript.

References

Aho, A. V., Corasick, M. J. (1975). Efficient string matching: An aid to bibliographic search. *Communications of the ACM*, *18*, 333–340.

Aston, J. A. D., Martin, D. E. K. (2007). Waiting time distributions of general runs and patterns in hidden Markov models. *Annals of Applied Statistics*, *1*(2), 585–611.

Begleiter, R., El-Yaniv, R., Yona, G. (2004). On prediction using variable length Markov models. *Journal of Artificial Intelligence*, *22*, 385–421.
Belloni, A., Oliveira, R. (2017). Approximate group context tree. *The Annals of Statistics*, *45*(1), 355–385.
Ben-gal, I., Morag, G., Shmilovici, A. (2003). Context-based statistical process control. *Technometrics*, *45*(4), 293–311.
Benson, G., Mak, D. Y. F. (2009). Exact distribution of a spaced seed statistic for DNA homology detection. *String processing and information retrieval, Lecture Notes in Computer Science*, Vol. 5280, pp. 283–293. Berlin: Springer.
Bercovici, S., Rodriguez, J. M., Elmore, M., Batzoglou, S. (2012). Ancestry inference in complex admixtures via variable-length Markov chain linkage models. *Research in computational molecular biology, RECOMB 2012, Lecture Notes in Computer Science*, Vol. 7262, pp. 12–28. Berlin: Springer.
Borges, J., Levene, M. (2007). Evaluating variable length Markov chain models for analysis of user web navigation. *IEEE Transactions on Knowledge*, *19*(4), 441–452.
Bratko, A., Cormack, G., Filipič, B., Lynam, T., Zupan, B. (2006). Spam filtering using statistical data compression models. *Journal of Machine Learning Research*, *7*, 2673–2698.
Brookner, E. (1966). Recurrent events in a Markov chain. *Information and Control*, *9*, 215–229.
Browning, S. R. (2006). Multilocus association mapping using variable-length Markov chains. *American Journal of Human Genetics*, *78*, 903–913.
Buhler, J., Keich, U., Sun, Y. (2005). Designing seeds for similarity search in genomic DNA. *Journal of Computer and Systems Science*, *70*, 342–363.
Bühlmann, P., Wyner, A. J. (1999). Variable length Markov chains. *Annals of Statistics*, *27*(2), 480–513.
Fernández, M., García, J. E., González-López, V. A. (2018). A copula-based partition Markov procedure. *Communications in Statistics-Theory and Methods*, *47*(14), 3408–3417.
Fu, J. C., Koutras, M. V. (1994). Distribution theory of runs: A Markov chain approach. *Journal of the American Statistical Association*, *89*, 1050–1058.
Gabadinho, A., Ritschard, G. (2016). Analyzing state sequences with probabilistic suffix trees. *Journal of Statistical Software*, *72*(3), 1–39.
Gallo, S., Leonardi, F. (2015). Nonparametric statistical inference for the context tree of a stationary ergodic process. *Electronic Journal of Statistics*, *9*, 2076–2098.
Galves, A., Galves, C., García, J. E., Garcia, N. L., Leonardi, F. (2012). Context tree selection and linguistic rhythm retrieval from written texts. *Annals of Applied Statistics*, *6*, 186–209.
García, J. E., González-López, V. A. (2010). Minimal Markov models. arXiv:1002.0729.
García, J. E., González-López, V. A. (2017). Consistent estimation of partition Markov models. *Entropy*, *19*, 1050–1058.
Hopcroft, J. E. (1971). An n log n algorithm for minimizing states in a finite automaton. In Z. Kohavi & A. Paz (Eds.), *Theory of Machines and Computation*, pp. 189–196. New York: Academic Press.
Jääskinen, V., Xiong, J., Koski, T., Corander, J. (2014). Sparse Markov chains for sequence data. *Scandinavian Journal of Statistics*, *41*, 641–655.
Keich, U., Li, M., Ma, B., Tromp, J. (2004). On spaced seeds for similarity search. *Discrete Applied Mathematics*, *138*(3), 253–263.
Koutras, M. V., Alexandrou, V. A. (1995). Runs, scans and urn models: A unified Markov chain approach. *Annals of the Institute of Statistical Mathematics*, *47*, 743–766.
Lladser, M. E. (2007). Minimal Markov chain embeddings of pattern problems. In *Proceedings of the 2007 information theory and applications workshop*, University of California, San Diego.
Lladser, M., Betterton, M. D., Knight, R. (2008). Multiple pattern matching: A Markov chain approach. *Journal of Mathematical Biology*, *56*(1–2), 51–92.
Ma, B., Tromp, J., Li, M. (2002). PatternHunter: Faster and more sensitive homology search. *Bioinformatics*, *18*(3), 440–445.
Mak, D. Y. F., Benson, G. (2009). All hits all the time: Parameter-free calculation of spaced seed sensitivity. *Bioinformatics*, *25*(3), 302–308.
Marshall, T., Rahmann, S. (2008). Probabilistic arithmetic automata and their application to pattern matching statistics. In: Ferragina, P., Landau, G.M. (eds), *Proceedings of the 19th annual symposium on combinatorial pattern matching (CPM), Lecture Notes in Computer Science*, Vol. 5029, pp. 95–106. Heidelberg: Springer.
Martin, D. E. K. (2018). Minimal auxiliary Markov chains through sequential elimination of states. *Communications in Statistics-Simulation and Computation*. https://doi.org/10.1080/03610918.2017.1406505.

Martin, D. E. K., Coleman, D. A. (2011). Distributions of clump statistics for a collection of words. *Journal of Applied Probability*, *48*, 1049–1059.
Martin, D. E. K., Noé, L. (2017). Faster exact probabilities for statistics of overlapping pattern occurrences. *Annals of the Institute of Statistical Mathematics*, *69*(1), 231–248.
Noé, L. (2017). Best hits of 11110110111: Model-free selection and parameter-free sensitivity calculation of spaced seeds. *Algorithms for Molecular Biology*, *12*(1), 1. https://doi.org/10.1186/s13015-017-0092-1.
Noé, L., Martin, D. E. K. (2014). A coverage criterion for spaced seeds and its applications to SVM string-kernels and *k*-mer distances. *Journal of Computational Biology*, *21*(12), 947–963.
Nuel, G. (2008). Pattern Markov chains: Optimal Markov chain embedding through deterministic finite automata. *Journal of Applied Probability*, *45*, 226–243.
Ribeca, P., Raineri, E. (2008). Faster exact Markovian probability functions for motif occurrences: A DFA-only approach. *Bioinformatics*, *24*(24), 2839–2848.
Rissanen, J. (1983). A universal data compression system. *IEEE Transactions on Information Theory*, *29*, 656–664.
Rissanen, J. (1986). Complexity of strings in the class of Markov sources. *IEEE Transactions on Information Theory*, *32*(4), 526–532.
Ron, D., Singer, Y., Tishby, N. (1996). The power of amnesia: Learning probabilistic automata with variable memory length. *Machine Learning*, *25*(2–3), 117–149.
Roos, T., Yu, B. (2009). Sparse Markov source estimation via transformed Lasso. In *Proceedings of the IEEE Information Theory Workshop (ITW-2009)*, pp. 241–245. Taormina, Sicily, Italy.
Shmilovici, A., Ben-gal, I. (2007). Using a VOM model for reconstructing potential coding regions in EST sequences. *Computational Statistics*, *22*, 49–69.
Weinberger, M., Lempel, A., Ziv, J. (1992). A sequential algorithm for the universal coding of finite memory sources. *IEEE Transactions on Information Theory*, *IT–38*, 1002–1024.
Weinberger, M., Rissanen, J., Feder, M. (1995). A universal finite memory source. *IEEE Transactions on Information Theory*, *41*(3), 643–652.
Willems, F. M. J., Shtarkov, Y. M., Tjalkens, T. J. (1995). The context-tree weighting method: Basic properties. *IEEE Transactions on Information Theory*, *41*(3), 653–664.
Xiong, J., Jääskinen, V., Corander, J. (2016). Recursive learning for sparse Markov models. *Bayesian Analysis*, *11*(1), 247–263.