Data Mining and Knowledge Discovery manuscript No.
(will be inserted by the editor)

Discovering Episodes with Compact Minimal Windows

Nikolaj Tatti

the date of receipt and acceptance should be inserted later

Abstract Discovering the most interesting patterns is the key problem in the field of pattern mining. While ranking or selecting patterns is well-studied for itemsets it is surprisingly under-researched for other, more complex, pattern types.

In this paper we propose a new quality measure for episodes. An episode is essentially a set of events with possible restrictions on the order of events. We say that an episode is significant if its occurrence is abnormally compact, that is, only few gap events occur between the actual episode events, when compared to the expected length according to the independence model. We can apply this measure as a post-pruning step by first discovering frequent episodes and then rank them according to this measure.

In order to compute the score we will need to compute the mean and the variance according to the independence model. As a main technical contribution we introduce a technique that allows us to compute these values. Such a task is surprisingly complex and in order to solve it we develop intricate finite state machines that allow us to compute the needed statistics. We also show that asymptotically our score can be interpreted as a P-value. In our experiments we demonstrate that despite its intricacy our ranking is fast: we can rank tens of thousands episodes in seconds. Our experiments with text data demonstrate that our measure ranks interpretable episodes high.

Keywords episode mining; statistical test; independence model; minimal window

Nikolaj Tatti
ADReM, University of Antwerp, Belgium
DTAI, KU Leuven, Belgium
HIIT, Aalto University, Finland
E-mail: nikolaj.tatti@aalto.fi

1 Introduction

Discovering the most interesting patterns is the key problem in the field of pattern mining. While ranking or selecting patterns is well-studied for itemsets, a canonical and arguably the easiest pattern type, it is surprisingly under-researched for other, more complex, pattern types.

Discovering episodes, frequent patterns from an event sequence has been a fruitful and active field in pattern mining since their original introduction by Mannila et al (1997). Essentially, an episode is a set of events that should occur close to each other (gaps are allowed) possibly with some constraints on the order of the occurrences, see Section 2 for full definition. While the concept of support for itemsets is straightforward, it is simply the number of transactions containing the pattern, defining a support for episodes is more complex. The most common way of defining a support is to slide a window of fixed size over the sequence and count in how many windows the pattern occurs. Such a measure is monotonically decreasing and hence all frequent episodes can be found using APriori approach given by Mannila et al (1997). Alternatively we can consider counting minimal windows, that is finding and counting the most compact windows that contain the episode.

The common wisdom is that finding frequent patterns is not enough. Discovering frequent patterns with high threshold will result to trivial patterns, omitting many interesting patterns, while using a low threshold will result in a pattern explosion. This phenomenon has led to many ranking methods for itemsets, the most well-studied pattern type. Unlike for itemsets, ranking episodes is heavily under-developed. Existing statistical approaches for ranking episodes are mostly based on the number of fixed-size windows (see more detailed discussion in Section 6). However, a natural way of measuring the goodness of an episode is the average length of its instances—a good episode should have compact minimal windows. Hence, our goal and contribution is a measure based directly on the average length of minimal windows.

The most straightforward and common way to measure significance for itemsets is to compare the observed support, the number of transactions in which all attributes co-occur, against the independence model: if the observed support deviates a lot from the expectation, we consider the itemset important. In this paper we use the same principle and propose an interestingness measure for an episode by comparing the observed lengths of minimal windows of the episode against the expectation computed from the independence model. Given a set of episodes we can now apply our measure to each episode and rank the episodes, placing episodes with the most abnormal minimal windows on top. While this is an easy task for itemsets, computing statistics turns out to be complex for episodes.

We define our score as follows: given an episode G, we assign a weight to each minimal window of G based on how long it is. The weight will be large for small windows and small for large windows. To compute the expected weight we assume that for each symbol we have a probability of its occurrence in the sequence. We then compute the expected weight based on a model in which the symbols are independent of each other. We say that the episode is significant if the observed average weight is abnormally large, that is, the minimal windows are abnormally short.

Example 1 Assume that we have an alphabet of size 3, $\Sigma = \{a, b, c\}$. Assume that the probabilities for having a symbol are $p(a) = 1/2$, $p(b) = 1/4$, and $p(c) = 1/4$. Let G be a serial episode $a \to b$. Then s is a minimal window for G if and only if it has a form $ac \cdots cb$. Hence the probability of a random sequence s of length k to be a minimal window for G is equal to

$$p(s \text{ is a minimal window of } G, |s| = k) = \frac{1}{2} \times \frac{1}{4} \times \frac{1}{4^{k-2}}.$$

We are interested in a probability of a minimal window having length k. To get this we divide the joint probability by the probability

$$p(s \text{ is a minimal window of } G) = \sum_{k=2}^{\infty} \frac{1}{2} \times \frac{1}{4} \times \frac{1}{4^{k-2}} = 1/6.$$

Using this normalisation we get that the probability of a minimal window having length k is equal to

$$p(|s| = k \mid s \text{ is a minimal window of } G) = 3/4 \times 1/4^{k-2},$$

for $k \geq 2$, and 0 otherwise. If we now weight minimal windows with an exponential decay, say, $1/2^{|s|}$, then the expected weight is equal to $3/14 \approx 0.2$. On the other hand, assume that we have a sequence $s = abcacbcababcab$. There are 4 minimal windows of length 2 and one minimal window of length 3. Hence, the observed average weight is $(4 \times 1/2^2 + 1/2^3)/5 = 0.225$ suggesting that the minimal windows are more compact than what the independence model implies.

Computing the needed statistics turns out to be a surprisingly complex problem. We attack this problem in Section 4 by introducing a certain finite state machine having episodes as the nodes. Then using this structure we are able to compute the statistics recursively, starting from simple episodes and moving towards more complex ones.

Our recipe for the mining process is as follows: Given the sequence we first split the sequence in two. The first sequence is used for discovering candidate episodes, in our case episodes that have a large number of minimal windows. Luckily, this condition is monotonically decreasing and we can mine these episodes using a standard APriori method. We also compute the needed probabilities of individual events from the first sequence. Once we have discovered candidate episodes and have computed the expectation, we compare the expected weight against the average observed weight from the *second* sequence using a simple Z-score. This step allows us to prune uninteresting episodes, which is in our case episodes that obey the independence model.

The rest of the paper is structured as follows. In Section 2 we introduce the preliminary definitions and notation. We introduce our method for evaluating the difference between the observed windows and the independence model in Section 3. In Sections 4–5 we lay out our approach for computing the independence model. We present the related work in Section 6. Our experiments are given in Section 7 and we conclude our work with discussion in Section 8. All proofs are given in Appendix.

2 Preliminaries and Notation

We begin by presenting preliminary concepts and notations that will be used throughout the rest of the paper.

A *sequence* $s = (s_1, \ldots, s_L)$ is a string of symbols coming from a finite *alphabet* Σ, that is, we have $s_i \in \Sigma$. Given a sequence s and two indices i and j, such that $i \leq j$, we denote by $s[i, j] = (s_i, \ldots, s_j)$ a sub-sequence of s.

An episode G is represented by an acyclic directed graph with labelled nodes, that is $G = (V, E, \mathit{lab})$, where $V = (v_1, \ldots, v_K)$ is the set of nodes, E is the set of directed edges, and lab is the function $\mathit{lab} : V \to \Sigma$, mapping each node v_i to its label.

Given a sequence s and an episode G we say that s *covers* the episode if there is an *injective* map f mapping each node v_i to a valid index such that the node and the corresponding sequence element have the same label, $s_{f(v_i)} = \mathit{lab}(v_i)$, and that if there is an edge $(v_i, v_j) \in E$, then we must have $f(v_i) < f(v_j)$. In other words, the parents of the node v_i must occur in s before v_i. Traditional episode mining is based on searching episodes that are covered by sufficiently many sub-windows of certain fixed size.

Example 2 Consider an episode given in Figure 1. This episode has 4 nodes labelled as a, b, c, and d, and requires that a must come first, followed by b and c in arbitrary order, and finally followed by d. Figure 1 also shows an example of a sequence that covers the episode.

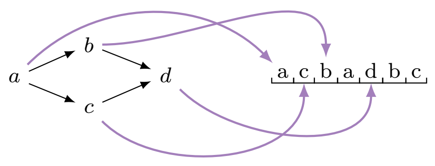


Fig. 1: A toy episode with 4 nodes and an example of a sequence covering the episode

An elementary theorem says that in a directed acyclic graph there exists a sink, a node with no outgoing edges. We denote the set of sinks by $\mathit{sinks}(G)$.

Given an episode G and a node v, we define $G - v$ to be the sub-episode obtained from G by removing v, and the incident edges.

Given an episode G we define a set of *prefix* episodes by

$$pre(G) = \{G\} \cup \bigcup_{v \in sinks(G)} pre(G - v)\,,$$

that is, a prefix episode H is a subepisode of G such that if v_i is contained in H, then all parents (in G) of v_i are also contained in H.

Example 3 Episode given in Figure 1 has 6 prefix episodes. Among of these 6 episodes one is empty, the remaining 5 episodes are given in Figure 2.

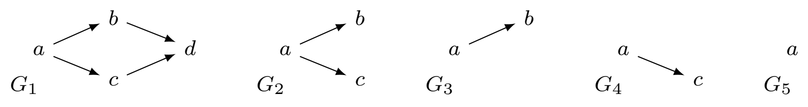


Fig. 2: Non-empty prefix episodes of an episode given in Figure 1

3 Minimal Windows of Episodes

Traditionally, discovering episodes from a single long sequence can be done in two ways. The first approach is to slide a window of fixed sized over the window and count the number of windows in which the episode occurs. The second approach is to count the number of minimal windows. The goal of this paper is to build a measure based minimal windows. If the statistic is abnormal, then we consider this pattern important.

In order to make the preceding discussion more formal, let G be an episode, and let s be a sequence. We say that s is a *minimal window* for G if G is covered by s but not by any proper sub-window of s. In this paper we are interested in discovering episodes that have abnormally compact minimal windows, a natural way of defining the significance of an episode.

Example 4 Consider a toy episode given in Figure 1. The sequence given in Figure 1 covers the episode but it not a minimal window. However, if we remove 2 last symbols from the sequence, then the sequence becomes a minimal window.

Example 5 Consider a serial episode $a \rightarrow b$, that is a pattern stating event a should be followed by an event b, and two sequences '*abababababababab*' and '*abacbadbaxbaybab*'. If we fix the length of a window to be 6 (or larger), then the number of windows covering the episode will be the same for the both sequences. In fact, in this case all windows will contain the episode. However, occurrences of the episode in these sequences are different. In the first sequence, all minimal windows are of length 2, while in the second sequence, we have 2 minimal windows of length 2 and 4 minimal windows of length 3. Our intuition is that $a \to b$ should be considered more significant in the first sequence than in the second.

Our goal in this paper is to design a measure that will indicate if the minimal windows are significantly compact. One approach would be to measure the average length of minimal windows. However, this ratio is susceptible to the variance in large minimal windows: consider that we have two minimal windows: the first is of length 10 and the other is of length 1000. Then the length of the second window dominates the average length, even though the first window is more interesting. In order to counter this phenomenon we suggest using the following statistic. Assume that we are given a parameter $0 < \rho < 1$. Let s be a minimal window for G. We define the *weight* of a window to be $\rho^{|s|}$. Compact windows will have a large value whereas large windows will have a small value. Let r be the average weight of all minimal windows for G.

We are interested in testing whether r is significantly large. In order to do that, let s be a random sequence and define a random variable $Y_i = a$ if $s[i, a]$ is a minimal window, if there is no such a we define $Y_i = 0$. Define also $X_i = Y_i > 0$ to be the indicator whether s has a minimal window of G starting at ith index.

We suggest using the following statistic. Given a parameter $0 < \rho < 1$, we define $Z_i = X_i \rho^{Y_i - i + 1}$. Then r is an estimate of a statistic $\sum_{i=1}^{\infty} Z_i / \sum_{i=1}^{\infty} X_i$.

We will show that there is μ and σ such that

$$\sqrt{L}\Big(\sum_{i=1}^{L} Z_i / \sum_{i=1}^{L} X_i - \mu\Big)$$

approaches a normal distribution $N(0, \sigma^2)$. This suggest to define a measure $sc(G) = \sqrt{L}(r - \mu)/\sigma$. This is simply a Z-normalisation of the statistic r.

We can also compute $\Phi(-sc(G))$, where Φ is the cumulative density function of the standard normal distribution $N(0, 1)$, and interpret this quantity as a P-value. However, this interpretation is problematic mainly because the normal distribution estimate is only accurate asymptotically.

Hence, we only consider $sc(G)$ merely as a ranking measure. Nevertheless, this measure makes a lot of sense: it measures how much the observed value deviates from the expectation, a common approach in ranking patterns, and it also takes the account the uncertainty of the measure.

In order to achieve our goal, we need to perform two steps

1. We need to show that $sc(G)$ converges into $N(0, 1)$
2. We need to compute μ and σ^2 that are needed for $sc(G)$.

Both of these steps are non-trivial. Proving asymptotic normality is difficult because X_i, Z_i, and Y_i are not independent, hence we will have to show that the sequence is mixing fast enough. Computing μ and σ^2 will require a set of recursive equations. The remaining theoretical sections are devoted to proving asymptotic normality and computing the mean and the variance.

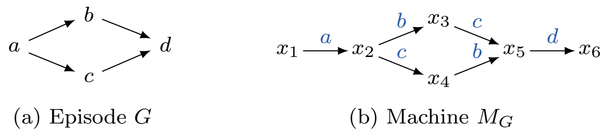


Fig. 3: Toy example of an episode G and the corresponding machine M_G.

4 Detecting Minimal Windows

In this and the next section we establish our main theoretical contribution, which is how to compute $sc(G)$.

We divide our task as follows: In Section 4.1 we build a finite state machine recognising when an episode is covered. In Section 4.2 we modify this machine so that we can use it for subsequent statistical calculations. Using this machine as a base we construct in Section 4.3 a machine that is able to recognise a minimal window of G.

4.1 Constructing finite state machine

We begin by constructing a finite state machine that recognises the coverage of an episode.

In this paper, a *finite state machine* (or simply a machine) M is a DAG with labelled edges and a single source. We allow multiple edges between two nodes.

Given a state x in M we say that s *covers* x if there is a subsequence $t = (s_{i_1}, \ldots, s_{i_N})$ such that x can be reached from the source node using t as an input.

Given an episode G, we define a machine M_G to be a DAG containing prefix graphs as nodes $V(M_G) = \{x_H \mid H \in pre(G)\}$. We add an edge $e = (x_H, x_F)$ if and only if there is a sink node $v \in V(G)$ such that $H = F - v$. We label edge e with the label of v, $lab(e) = lab(v)$.

Example 6 Consider an episode G given in Figure 3a. The corresponding machine M_G is given in Figure 3b. Sink state x_6 corresponds to episode G and source state x_1 corresponds to the empty episode. Intermediate state x_5 corresponds to G_2 given in Figure 2, x_3 corresponds to G_3, x_4 corresponds to G_4, and x_2 corresponds to G_5.

Comparing the definition of coverage of a state in x and the definition of a coverage for episodes gives immediately the following proposition.

Proposition 1 *Given an episode G, a sequence s covers an episode $H \in pre(G)$ if and only if s covers the corresponding state x_H in M_G.*

4.2 Making Simple Machines

In order to be able to compute the needed probabilities in subsequent sections, a machine need to have a crucial property. We say that machine M is *simple* if each state in M does not multiple incoming edges with the same label. If we reverse the direction of edges, then simplicity is equivalent to a finite state machine being deterministic.

In general, M_G is not simple. If an episode G contains two nodes, say v_i and v_j with the same label such that v_i is not an ancestor of v_j and vice versa, then there is a state x_H in M_G, where H is a prefix episode having v_i and v_j as sinks will have (at least) two incoming edges with the same label (see Figures 4a–4b).

Luckily, we can transform M_G into a simple machine. This transformation is almost equivalent to a process of making a non-deterministic finite state machine to deterministic.

In order to make this formal, let us first give some definitions. Assume that we are given a machine M. Given a state x in M, we define

$$\mathit{in}(x) = \{\mathit{lab}(e) \mid e = (y, x) \in E(M)\}$$

to be the set of labels of all incoming edges. If X is a subset of states in M, then we write $\mathit{in}(X) = \bigcup_{x \in X} \mathit{in}(x)$.

Let X be a subset of states in M and let a be a label. We define

$$\mathit{sub}(X; a) = \{y \mid e = (y, x) \in E(M), \mathit{lab}(e) = a, x \in X\}$$

to be the union set of parents of each $v \in X$ connected with an edge having the label a. We also define

$$\mathit{stay}(X; a) = \{x \in X \mid a \notin \mathit{in}(x)\}$$

to be the set of states that have no incoming edge with a label a.

Let i be the (unique) source state in M. We define

$$\mathit{par}(X; a) = \begin{cases} \mathit{sub}(X; a) \cup \mathit{stay}(X; a) & \text{if } i \notin \mathit{sub}(X; a) \\ \{i\} & \text{if } i \in \mathit{sub}(X; a)\,. \end{cases}$$

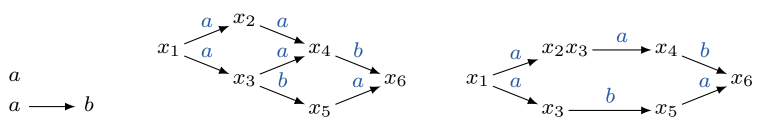
Finally, we define a closure of X inductively to be the collection of sets of states

$$\mathit{cl}(X) = \{X\} \cup \bigcup_{a \in \mathit{in}(X)} \mathit{cl}(\mathit{par}(X; a))\,.$$

We are now ready to define a simple machine $\mathit{sm}(M)$. The states of this machine are

$$V(\mathit{sm}(M)) = \bigcup_{x \in \mathit{sinks}(M)} \mathit{cl}(\{x\})\,.$$

An edge $e = (X, Y)$ with a label a is in $E(\mathit{sm}(M))$ if and only if $a \in \mathit{in}(Y)$ and $X = \mathit{par}(Y; a)$. Since, for each a, there is only one X such that $X = \mathit{par}(Y; a)$, it follows that $\mathit{sm}(M)$ is simple.



(a) G (b) Machine M_G (c) Machine $sm(M_G)$

Fig. 4: Toy example of an episode G and the corresponding machines M_G and $sm(M_G)$.

Example 7 A machine M_G given in Figure 4b is not simple since the state x_4 has two incoming edges with a, each edge correspond to either one of a. In order to obtain $sm(M_G)$, we first observe that the nodes are

$$\begin{aligned}\{\{x_6\}\} \cup cl(\{x_4\}) \cup cl(\{x_5\}) &= \{\{x_6\}, \{x_4\}, \{x_5\}\} \cup cl(\{x_2, x_3\}) \cup cl(\{x_5\}) \\ &= \{\{x_6\}, \{x_4\}, \{x_5\}, \{x_2, x_3\}, \{x_3\}, \{x_1\}\}.\end{aligned}$$

This final machine is given in Figure 4c. Note that $sm(M_G)$ is simple since parents of x_4 are grouped together.

The following proposition reveals the expected result between M and $sm(M)$.

Proposition 2 *Let M be a machine. Let $X = \{x_1, \ldots, x_N\}$ be a state in $sm(M)$. Then a sequence s covers V if and only if s covers at least one x_i.*

The coverage of a machine is based on subsequences and working with subsequences is particularly difficult since there may be several subsequences that cover episode G, which leads to difficulties when computing probabilities.

Instead of working with subsequences directly, we will define a greedy function. Assume that we are given a simple machine M. Let $x \in M$ be a state and let $s = (s_1, \ldots, s_L)$ be a sequence. We define a greedy function recursively

$$g(x, s) = \begin{cases} x & \text{if } L = 0, \\ g(y, s[1, L-1]) & \text{if there is } (y, x) \text{ such that } lab((y, x)) = s_L, \\ g(x, s[1, L-1]) & \text{otherwise.} \end{cases}$$

In other words, the greedy function descends to parent states as fast as possible.

Example 8 Consider a machine M_G given in Figure 3b and sequence $s = acbadbc$ given in Figure 1. We have

$$\begin{aligned} g(x_6, acbadbc) &= g(x_6, acbadb) = g(x_6, acbad) = g(x_5, acba) \\ &= g(x_5, acb) = g(x_4, ac) = g(x_2, a) = g(x_1, \emptyset) = x_1. \end{aligned}$$

The example suggests that a sequence covers an episode if the greedy function reaches the source state in the corresponding machine. This holds in general: the following proposition shows that we can use the greedy function to test for coverage. Note that this crucial property is specific to machine induced from episodes. It will not hold for a general machine.

Proposition 3 *Let G be an episode, then a sequence s covers X, a state in $sm(M_G)$, if and only if $g(X, s) = \{i\}$, the source state of $sm(M_G)$.*

Corollary 1 *Let G be an episode and let X be the sink state of $sm(M_G)$. A sequence s covers G if and only if $g(X, s) = \{i\}$, the source state of $sm(M_G)$.*

4.3 Machine recognising minimal windows

So far we have constructed M_G and $sm(M_G)$ that recognise when a sequence covers G. However, we are interested in finding out when a sequence is a minimal window for G.

Assume that we are given an episode G and let $M = sm(M_G)$. Let $I = \{i\}$ be the source state of M and let $S = \{x_G\}$ be the sink state of M. We define two machines,

1. M_1 is obtained from M by adding a new source state, say J, and adding an edge (J, I) for each possible label.
2. M_2 is obtained from M by adding a new sink state, say T and adding an edge (S, T) for each possible label.

Both M_1 and M_2 are simple.

Let us first consider M_1. Assume that we are given a sequence $s = s_1 \cdots s_L$ such that $g(S, s) = I$. Then we know immediately that s covers G but $s[2, L]$ does not. Now let us consider M_2. Sequence $s[1, L-1]$ covers G if and only $g(T, s) = I$. Consequently, we need to design a machine that simultaneously computes $g(S, s)$ for M_1 and $g(T, s)$ for M_2.

In order to do so we need to define a special machine. Assume that we are given two simple machines M_1 and M_2, and a set of pairs of states $\Theta = \{(x_i, y_i)\}_{i=1}^{N}$, where x_i is a state in M_1 and y_i is a state in M_2. We will now define a join machine, $M^* = co(M_1, M_2, \Theta)$, that is guaranteed to contain the states from Θ. To define the states of this machine, let z_1 be a state in M_1 and let z_2 be a state in M_2. We first define a set of pairs of states recursively

$$f(z_1, z_2) = (z_1, z_2) \cup \bigcup_{a \in in(z_1) \cup in(z_2)} f(g(z_1, a), g(z_2, a)).$$

We define the states of M^* to be $\bigcup_{\theta \in \Theta} f(\theta)$. Two states $\alpha = (y_1, y_2)$ and $\beta = (z_1, z_2)$ are connected with an edge (α, β) if and only if $y_i = g(z_i, a)$ and $a \in in(z_1) \cup in(z_2)$. It follows immediately that M^* is simple.

Proposition 4 *Let M_1 and M_2 be two simple machines. Let Θ be a set of pairs of states. Define $M^* = co(M_1, M_2, \Theta)$. Let $\alpha = (x_1, x_2)$ be a state in M^*. Then $g(\alpha, s) = (g(x_1, s), g(x_2, s))$.*

We can now define a machine that we will use to test whether sequence is a minimal window of G. Let M_1, M_2, S and T as defined above. Let $M^* = co(M_1, M_2, \{(S, T)\})$. The following proposition demonstrates how we can use M^* to characterise the minimal window.

Proposition 5 *Let M_1, M_2, M^*, and I be as defined above. Let α be a sink state of M^*. Then, a sequence s is a minimal window for G if and only if $g(\alpha, s) \in \Omega$, where $\Omega = \{(I, Y) \mid I \neq Y$ is a state of $M_2\}$.*

For the purpose of recognising minimal windows, there are lot of redundant states in M^*. Any state that is not a child or part of Ω can be removed and the outgoing edges reattached to the source state without effecting the validity of Proposition 5. This is true because once the greedy function reaches any such state then it will never reach Ω. To optimise we remove two types of non-source states: any of form (J, Y), where J is the source state of M_1 and any state of form (Y, Y). We refer to the resulting machine as $minm(G)$.

Example 9 Consider an episode G given in Figure 5a. The machine $sm(M_G)$ is given in Figure 5b and the augmented versions M_1 and M_2 are given in Figures 5c–5d. These machines are then combined to $M^* = co(M_1, M_2, \{(x_1, x_0)\})$, given in Figure 5e.

The final, simplified, machine is given in Figure 5f. In order to a sequence to be a minimal window for G, the greedy function must land either in (x_5, x_3), (x_5, x_1), or in (x_5, x_4). Note that many states from M^* are removed. For example, if we are in x_1x_0 and we see any other symbol than c, then we know that s is not a minimal window since s must end with c in order to be one.

5 Computing Moments

Now that we have defined a machine for recognising a minimal window, we will use it to compute the needed probabilities. In Section 5.1 we demonstrate how to use the machine to compute the expected weight. In Section 5.2 we show the asymptotic normality and in Section 5.3 we demonstrate how to compute the variance. We finish the section by considering computational complexity.

5.1 Computing probabilities

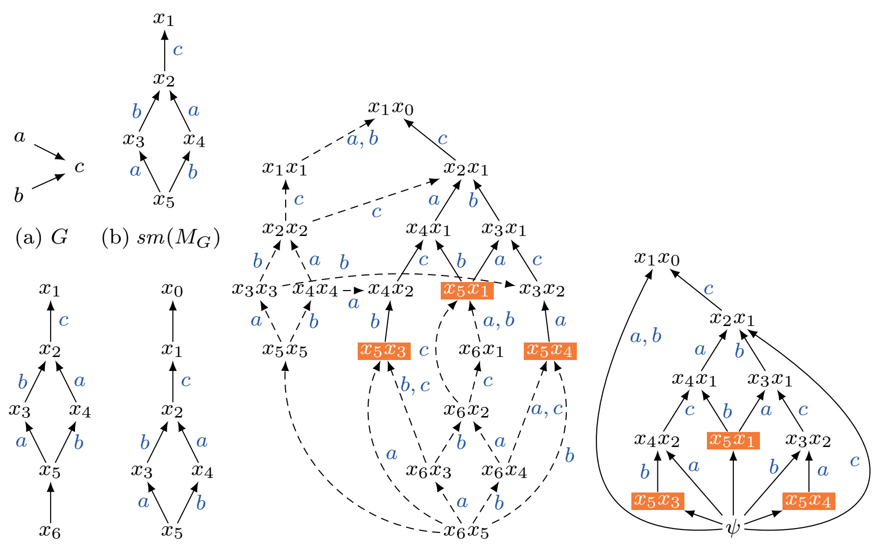
Proposition 5 gives us means to express the minimal window using a machine and the greedy function. In this section we demonstrate how to compute probabilities that the greedy function lands in some particular state.

Let M be a simple machine. Let Y be a set of states in M and let x be a state in M. Let us first define

$$pg(x, Y, L) = p(g(x, s) \in Y \mid |s| = L)$$

to be the probability that a random sequence s of length L reaches one of the states in Y.

Proposition 6 *Let M be a simple machine. Let Y be a set of states in M and let x be a state in M.*



(a) G (b) $sm(M_G)$ (c) M_1 (d) M_2 (e) $co(M_1, M_2, \{(x_1, x_0)\})$ (f) $minm(G)$

Fig. 5: Toy episode and related machines. Figure 5a contains an episode G. A simple machine $sm(M_G)$ is given in Figure 5b. Machines given in Figures 5c–5d are used to construct a machine to recognise a minimal window, Figure 5e. In order to a sequence to be a minimal window we must land to a highlighted node when starting from x_1x_0. The states with dashed outgoing edges are redundant and can be be collapsed, resulting in a machine given in Figure 5f.

Then it holds that for $L > 0$,

$$pg(x, Y, L) = \sum_{a \in \Sigma} p(a) pg(g(x, a), Y, L-1). \tag{1}$$

For $L = 0$, we have

$$pg(x, Y, 0) = \begin{cases} 1 & \textit{if } x \in Y, \\ 0 & \textit{if } x \notin Y. \end{cases}$$

Example 10 Consider a machine $minm(G)$ given Figure 5f. Assume that the individual probabilities are $p(a) = 0.3$, $p(b) = 0.2$, and $p(c) = 0.5$. The according to Proposition 1, $pg(x_4x_2, x_5x_3, 1) = 0.2$ and

$$pg(x_4x_2, x_5x_3, L) = 0.5 pg(x_4x_2, x_5x_3, L-1)$$

for $L > 1$, which implies that $pg(x_4x_2, x_5x_3, L) = 0.2 \times 0.5^{L-1}$. We can verify this by observing that the sequence of L events that leads from x_4x_2 to x_5x_3 must have $L - 1$ events labelled as c followed by one b.

To solve the needed quantities, we need to compute moments,

$$m(x, f, Y) = \sum_{L=1}^{\infty} f(L) pg(x, Y, L) \,.$$

Proposition 5 now immediately implies that we can express the needed statistics using moments.

Proposition 7 *Assume an episode G. Let $M = minm(G)$ and let α and Ω be as in Proposition 5. Let Y_i, X_i and Z_i be defined as in Section 3. Then*

$$\begin{aligned} \mathrm{E}\left[X_1\right] &= m(\alpha, f, \Omega)\,, \quad \textit{for} \quad f(L) = 1, \\ \mathrm{E}\left[Y_1\right] &= m(\alpha, f, \Omega)\,, \quad \textit{for} \quad f(L) = L, \\ \mathrm{E}\left[Z_1\right] &= m(\alpha, f, \Omega)\,, \quad \textit{for} \quad f(L) = \rho^L, \\ \mathrm{E}\left[Z_1^2\right] &= m(\alpha, f, \Omega)\,, \quad \textit{for} \quad f(L) = \rho^{2L}, \\ \mathrm{E}\left[Y_1 Z_1\right] &= m(\alpha, f, \Omega)\,, \quad \textit{for} \quad f(L) = \rho^L L. \end{aligned}$$

Note that the sum has infinite number of terms, hence we cannot compute this by raw application of Proposition 6. Luckily, we can express moments in closed recursive form. First, we need to show that the moments we consider are finite.

Lemma 1 *Let M be a simple machine. Let Y be a set of states in M. Assume that $p(a) > 0$ for all $a \in \Sigma$. Assume that we are given a function f such that $f(L)$ grows at polynomial rate. If the source node is not contained in Y, then $m(x, f, Y)$ is finite for any state x.*

Proposition 8 *Let M be a simple machine. Assume that we have a function f mapping an integer to a real number. Assume also for $L \geq 1$, we have $f(L-1) = cf(L) + h(L)$ for some $c \in \mathbb{R}$ and a function h. Assume that f and g grow at polynomial rate, at maximum. Let $q = 1 - \sum_{a \in in(x)} p(a)$ and set $r = c - q$. Let $i(y) = pg(y, Y, 0)\, f(0)$. Then*

$$m(x, f, Y) = \frac{1}{r}\big(qi(x) - m(x, h, Y) + \sum_{\substack{a \in in(x) \\ y = g(x,a)}} p(a)(m(y, f, Y) + i(y))\big).$$

We can now use Proposition 8 to compute the moments given in Proposition 7.

Proposition 9 *The identity $f(L-1) = cf(L) + h(L)$ holds for the following functions,*

$$\begin{aligned} f(L) &= 1, \quad \textit{for} \quad c = 1,\ h(L) = 0, \\ f(L) &= L, \quad \textit{for} \quad c = 1,\ h(L) = -1, \\ f(L) &= \rho^L, \quad \textit{for} \quad c = \rho^{-1},\ h(L) = 0, \\ f(L) &= \rho^{2L}, \quad \textit{for} \quad c = \rho^{-2},\ h(L) = 0, \\ f(L) &= \rho^L L, \quad \textit{for} \quad c = \rho^{-1},\ h(L) = -\rho^{L-1}. \end{aligned}$$

Example 11 Consider machine $minm(G)$ given in Figure 5e. Let us define $\Omega = \{(x_5, x_3), (x_5, x_1), (x_5, x_4)\}$. Assume also that the probabilities for the symbols are $p(a) = 0.3$, $p(b) = 0.2$, and $p(c) = 0.5$. Let $f(L) = 1$.

Then using Proposition 8 we see that

$$\begin{aligned}
m((x_4, x_2), f, \Omega) &= 0.2/0.5 = 0.4,\\
m((x_3, x_2), f, \Omega) &= 0.3/0.5 = 0.6,\\
m((x_4, x_1), f, \Omega) &= (0.2 + 0.5 \times 0.4)/0.7 = 4/7,\\
m((x_3, x_1), f, \Omega) &= (0.3 + 0.5 \times 0.6)/0.8 = 3/4,\\
m((x_2, x_1), f, \Omega) &= 0.3 \times 4/7 + 0.2 \times 3/4 = 0.32,\\
m((x_1, x_1), f, \Omega) &= 0.5 \times 0.32 = 0.16,
\end{aligned}$$

and the moment for the remaining states is equal to 0.

Proposition 8 gives us means for a straightforward algorithm MOMENTS for computing moments (given in Algorithm 1). MOMENTS takes as input a simple machine M, a map i for initial values, a map h for update values, and a constant c. Note that MOMENTS is linear function of i and h, that is,

$$\begin{aligned}
&\textsc{Moments}(M, k_1 i_1 + k_2 i_2, k_1 h_1 + k_2 h_2, c) =\\
&\quad k_1 \textsc{Moments}(M, i_1, h_1, c) + k_2 \textsc{Moments}(M, i_2, h_2, c)
\end{aligned}$$

for any constants k_1 and k_2. We will use this property later for speed-ups.

Algorithm 1: MOMENTS(M, i, h, c) computes moments using Proposition 8.

```
input  : a simple machine $M$, a map $i$ for initial values, a map $h$ for update values,
         and a constant $c$ for recursive update
output : Moment of $f$ for every state $x \in M$
1 for $x \in M$ in topological order do
2     $q \leftarrow 1 - \sum_{a \in in(x)} p(a)$;
3     $r \leftarrow c - q$;
4     $m(x) \leftarrow \frac{1}{r}\left(qi(x) - h(x) + \sum_{a \in in(x), y = g(x,a)} p(a)(m(y) + i(y))\right)$;
5 return $m$;
```

5.2 Asymptotic Normality

We will now prove that our statistic approaches to the normal distribution. The proof is not trivial since the variables X_i and Z_i are not independent. Hence we will use Central Limit Theorem for strongly mixing sequences.

Our first step is to show that the sequence the central limit theorem holds for (Z_i, X_i).

Proposition 10 *Let G be an episode. Sequence $1/\sqrt{L}\sum_{k=1}^{L}(Z_k, X_k) - (q, p)$ converges in distribution to $N(0, C)$, where $q = \mathrm{E}\left[Z_1\right]$, $p = \mathrm{E}\left[X_1\right]$, and C is a 2×2 covariance matrix, $C_{11} = \mathrm{Var}\left[Z_1\right] + 2D_{11}$, $C_{22} = \mathrm{Var}\left[X_1\right] + 2D_{22}$, $C_{21} = C_{12} = \mathrm{Cov}\left[X_1, Z_1\right] + D_{12} + D_{21}$, where*

$$
\begin{aligned}
D_{11} &= \sum_{i=2}^{\infty} \mathrm{E}\left[(Z_1 - q)(Z_i - q)\right], & D_{22} &= \sum_{i=2}^{\infty} \mathrm{E}\left[(X_1 - p)(X_i - p)\right], \\
D_{12} &= \sum_{i=2}^{\infty} \mathrm{E}\left[(Z_1 - q)(X_i - p)\right], & D_{21} &= \sum_{i=2}^{\infty} \mathrm{E}\left[(X_1 - p)(Z_i - q)\right].
\end{aligned}
$$

Since the central limit theorem holds for (Z_i, X_i), we can apply this to obtain the main result.

Proposition 11 *Let G be an episode. Let p, q and C be as in Proposition 10. Define $\mu = q/p$. Then*

$$\sqrt{L}\Big(\frac{\sum_{k=1}^{L} Z_k}{\sum_{k=1}^{L} X_k} - \mu\Big)$$

converges to $N(0, \sigma^2)$ as $L \to \infty$, where $\sigma^2 = p^{-2}\left(C_{11} - 2\mu C_{12} + \mu^2 C_{22}\right)$.

These results suggest that we can use $\Phi(-sc(G))$ as a P-value, where Φ is the cumulative density function of the normal distribution. However, in practice we have several problems:

– The result is accurate only asymptotically. Moreover, the distribution of $sc(G)$ can be heavily skewed so we need a large number of samples in order to estimate become accurate.
– We do not have directly, the probabilities of individual items, instead we will estimate the probabilities from the training sequence. This will introduce some error in prediction making the P-values smaller than they should be.
– We are computing a large number of statistical tests. In such case, it is advisable to use some technique, for example, Bonferroni correction, to compensate for the multiple hypotheses problem. However, it is not obvious which technique should we use.

Because of these problems, instead of interpreting $\Phi(-sc(G))$ as a P-value, we simply use $sc(G)$ to rank patterns and use it as a top-K method. Note that Φ is a monotonic function, hence the larger the score, the smaller the P-value.

By studying the formulas in the above propositions we see that we can compute the necessary statistics p and q using Proposition 7, and consequently we can compute μ. However, in order to compute the variance σ we need to compute D_{11}, D_{12}, D_{21}, and D_{22} given in Proposition 10. We will demonstrate a technique for computing these statistics in the next section.

5.3 Computing Cross-moments

Our final step is to compute cross-moments given in Proposition 10. In order to do so we first need to prove a different formulation of these statistics. This formulation is more fruitful as we no longer have to deal infinite sums.

Proposition 12 *Let p, q, D_{11}, D_{12}, D_{21}, and D_{22} be as in Proposition 10. Define $v = \mathrm{E}\left[Y_1\right]$ and $w = \mathrm{E}\left[Y_1 Z_1\right]$. Then*

$$D_{22} = \mathrm{E}\left[X_1 \sum_{k=2}^{Y_1} X_k\right] - (v-p)p, \qquad D_{12} = \mathrm{E}\left[X_1 Z_1 \sum_{k=2}^{Y_1} X_k\right] - (w-q)p,$$
$$D_{21} = \mathrm{E}\left[X_1 \sum_{k=2}^{Y_1} Z_k X_k\right] - (v-p)q, \quad D_{11} = \mathrm{E}\left[X_1 Z_1 \sum_{k=2}^{Y_1} Z_k X_k\right] - (w-q)q.$$

Our next step is to compute the moments. To that end, let $M = minm(G)$ be a machine recognising the minimal window of G, let α be a sink state in M, and let Ω be the states as in Proposition 5. We will study the probability $p(Y_1 = a, Y_k = a + b)$, where $a \geq k > 1$ and $b \geq 1$. Let $u = s[k, a]$ and $v = s[a{+}1, a{+}b]$. The idea is to break the probability into a sum of probabilities based on the state $g(\alpha, v)$ and $g(\alpha, u)$. These probabilities can be further decomposed into three factors which we can then turn into moments using Proposition 8.

Define a random variable $E = g(\alpha, s[1, a]) \in \Omega$. This variable is true if and only if $Y_1 = a$. In addition, define $F = g(\alpha, s[k, a + b]) \in \Omega$ and $G_\beta = g(\beta, u) \in \Omega$.

Let us write Θ to be all proper intermediate states of M between α and Ω. Since $k > 1$, $Y_1 = a$ implies that $g(\alpha, u) \in \Theta$. Similarly, $Y_k = a + b$ implies that $g(\alpha, v) \in \Theta$. We can now write $p(Y_1 = a, Y_k = a + b)$ as

$$\begin{aligned}
&p(Y_1 = a, Y_k = a + b) = p(E, F) \\
&\quad = \sum_{\beta\in\Theta} p(E, F, g(\alpha, v) = \beta) = \sum_{\beta\in\Theta} p(E, G_\beta, g(\alpha, v) = \beta) \\
&\quad = \sum_{\beta\in\Theta} p(E, G_\beta) p(g(\alpha, v) = \beta) = \sum_{\beta\in\Theta} pg(\alpha, \beta, b)\, p(E, G_\beta) \\
&\quad = \sum_{\beta\in\Theta} pg(\alpha, \beta, b) \sum_{\gamma\in\Theta} p(E, G_\beta, g(\alpha, u) = \gamma) \qquad (2)\\
&\quad = \sum_{\beta\in\Theta} pg(\alpha, \beta, b) \sum_{\gamma\in\Theta} p(g(\gamma, s[1, k-1]) \in \Omega, G_\beta, g(\alpha, u) = \gamma) \\
&\quad = \sum_{\beta\in\Theta} pg(\alpha, \beta, b) \sum_{\gamma\in\Theta} pg(\gamma, \Omega, k-1)\, p(G_\beta, g(\alpha, u) = \gamma).
\end{aligned}$$

The only non-trivial factor in Equation 2 that we cannot solve using M is $p(g(\beta, u) \in \Omega, g(\alpha, u) = \gamma)$. To solve this we construct yet another machine.

Let $M^* = co(M, M, \{(\theta, \alpha) \mid \theta \in \Theta\})$ and let $\Omega^*_\gamma = \{(\omega, \gamma) \mid \omega \in \Omega\}$. Then Proposition 4 implies that

$$p(G_\beta, g(\alpha, u) = \gamma) = pg\big((\beta, \alpha), \Omega^*_\gamma, a - k + 1\big) .$$

This leads to

$$p(Y_1 = a, Y_k = a+b) = \sum_{\beta,\gamma} pg(\alpha, \beta, b)\, pg(\gamma, \Omega, k-1)\, pg\big((\beta, \alpha), \Omega^*_\gamma, a - k + 1\big) . \tag{3}$$

Let us write $A_k = Y_1 - k + 1$. We now define a function f by which we can express the missing cross-moments,

$$f(P, Q, R) = \mathrm{E}\left[X_1 \sum_{k=2}^{Y_1} \rho^{P(k-1)+QA_k+R(Y_k-Y_1)} X_k\right].$$

This function is particularly useful since we can now apply Equation 3 and obtain a closed form using moments,

$$\begin{aligned} f(P, Q, R) &= \sum_{k=2}^{\infty}\sum_{b=1}^{\infty}\sum_{a=k}^{\infty} \rho^{P(k-1)+Q(a-k+1)+Rb} p(Y_1 = a, Y_k = a + b) \\ &= \sum_{\beta,\gamma} m(\alpha, R, \beta)\, m(\gamma, P, \Omega)\, m\big((\beta, \alpha), Q, \Omega^*_\gamma\big) . \end{aligned} \tag{4}$$

Let us now express the cross-moments using f. We see immediately that,

$$\begin{aligned} \mathrm{E}\left[X_1 \sum_{k=2}^{Y_1} X_k\right] &= f(0, 0, 0), \\ \mathrm{E}\left[X_1 \sum_{k=2}^{Y_1} Z_1 X_k\right] &= \mathrm{E}\left[X_1 \sum_{k=2}^{Y_1} \rho^{(k-1)+A_k} X_k\right] = f(1, 1, 0), \\ \mathrm{E}\left[X_1 \sum_{k=2}^{Y_1} Z_k X_k\right] &= \mathrm{E}\left[X_1 \sum_{k=2}^{Y_1} \rho^{A_k+Y_k-Y_1} A_k X_k\right] = f(0, 1, 1), \\ \mathrm{E}\left[X_1 \sum_{k=2}^{Y_1} Z_1 X_k Z_k\right] &= \mathrm{E}\left[X_1 \sum_{k=2}^{Y_1} X_k \rho^{(k-1)+2A_k+Y_k-Y_1}\right] = f(1, 2, 1). \end{aligned}$$

As a final step we describe how we can optimise computation of $f(P, Q, R)$. First recall that MOMENTS, given in Algorithm 1, is linear with respect to its parameters i and h. Consider Equation 4. Instead of computing the sum over β explicitly, we can compute MOMENTS$(M, i, 0, \rho^R)$, where $i(\beta)$ is defined as $\sum_\gamma m\big(\gamma, \rho^P, \Omega\big)\, m\big((\beta, \alpha), \rho^Q, \Omega^*\big)$. We can repeat this trick again to remove the explicit sum over γ. The pseudo-code taking into account these optimisations is given in Algorithm 2.

Algorithm 2: CrossMoments

```
$M \leftarrow minm(G)$;
$\alpha \leftarrow$ sink state of $M$;
$\Omega \leftarrow$ as in defined in Proposition 5;
$\Theta \leftarrow$ intermediate states of $\alpha$ and $\Omega$;
$M^* \leftarrow co(M, M, \{(\theta, \alpha) \mid \theta \in \Theta\})$;
$i_1(\theta) \leftarrow I(\theta \in \Omega)$;
$m \leftarrow$ Moments$(M, i_1, 0, \rho^P)$;
foreach state $x$ in $M^*$ do
    if $x = (\omega, \theta)$, where $\omega \in \Omega$ and $\theta \in \Theta$ then
        $i_2(x) \leftarrow m(\theta)$;
$m \leftarrow$ Moments$(M^*, i_2, \rho^Q)$;
foreach $\theta \in \Theta$ do
    $i_3(\theta) \leftarrow m((\theta, \alpha))$;
$m \leftarrow$ Moments$(M, i_3, 0, \rho^R)$;
return $m(\alpha)$;
```

Example 12 Let us compute $f(0, 1, 1)$ for an episode G given in Figure 5a. Let $M = minm(G)$, given in Figure 6. Note that this machine is the same machine given in Figure 5f. Let us define $\Omega = \{\omega_1, \omega_2, \omega_3\}$. Assume also that the probabilities for the symbols are $p(a) = 0.3$, $p(b) = 0.2$, and $p(c) = 0.5$ and assume that we selected $\rho = 1/2$. Define $h_k(x) = \rho^{kx}$.

To compute $f(0, 1, 1)$ we need to compute moments from three different machines. The obtained moments from a previous machine is fed as initial values to the next machine as shown in Figure 6. We use M for the first and the third machine. The second machine is $co(M, M, \{(\theta_1, \alpha), \ldots, (\theta_5, \alpha)\})$ with redundant states removed. This machine is given in Figure 6.

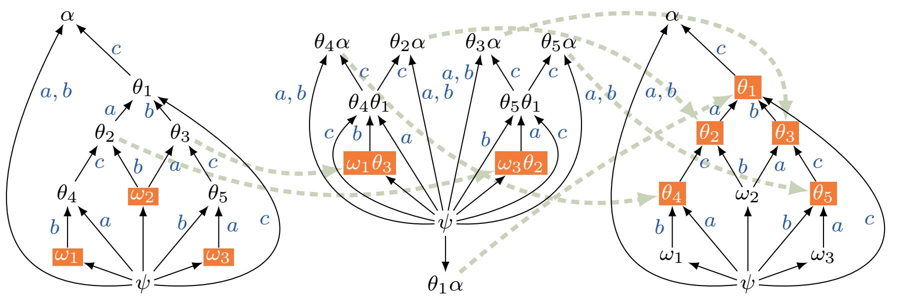


Fig. 6: Machines needed to compute the cross-moments for an episode G given in Figure 5a. The first and the third machines are $M = minm(G)$ and the second machine is $co(M, M, \{(\theta_1, \alpha), \ldots, (\theta_5, \alpha)\})$. We simplified the machine by collapsing all states containing ψ to one state. The arrows between the machines indicate how the moments from the previous machines are passed to the next machine as initial values.

We start with M, and as initial values we set 1 whenever a state is in Ω, and 0 otherwise. This is equivalent to Example 9. We need moments only for two states, θ_2 and θ_3, which are

$$m(\theta_2, h_0, \Omega) = 4/7 \quad \text{and} \quad m(\theta_3, h_0, \Omega) = 3/4.$$

We now use the moments of θ_2 and θ_3 as initial values for (ω_3, θ_2) and (ω_1, θ_3), that is, we set $i_2((\omega_3, \theta_2)) = 4/7$ and $i_2((\omega_1, \theta_3)) = 3/4$, and 0 for other states. We can now compute the moments,

$$\begin{aligned} m((\theta_4, \theta_1), h_1, i_2) &= (0.2 \times 3/4)/2 = 3/40, \\ m((\theta_5, \theta_1), h_1, i_2) &= (0.3 \times 4/7)/2 = 6/70, \\ m((\theta_2, \alpha), h_1, i_2) = m((\theta_4, \alpha), h_1, i_2) &= (0.5 \times 3/40)/2 = 3/160, \\ m((\theta_3, \alpha), h_1, i_2) = m((\theta_5, \alpha), h_1, i_2) &= (0.5 \times 6/70)/2 = 3/140, \end{aligned}$$

and 0 for the remaining states. We feed these moments into initial values i_3 and compute the final moments,

$$\begin{aligned} m(\theta_4, h_1, i_3) &= (0.5 \times 3/160)/1.5 = 1/160, \\ m(\theta_5, h_1, i_3) &= (0.5 \times 3/140)/1.5 = 1/140, \\ m(\theta_2, h_1, i_3) &= (0.3 \times 3/160 + 0.5/160)/1.7 = 14/2720, \\ m(\theta_3, h_1, i_3) &= (0.2 \times 3/140 + 0.5/140)/1.8 = 11/2520, \\ m(\theta_1, h_1, i_3) &= (0.2 \times 14/2720 + 0.3 \times 11/2520)/2 = 0.0012, \\ m(\alpha, h_1, i_3) &= 0.5 \times 0.001/2 = 0.0003. \end{aligned}$$

Consequently, $f(0, 1, 1) = m(\alpha, h_1, i_3) = 0.0003$.

5.4 Computational complexity

Let us now finish this section by discussing the computational complexity. Given a machine M, evaluating moments will take $O(V(M) + E(M))$ time. Hence, we need to study the sizes of our machines. Given an episode G with N nodes, the first machine $M_\mathcal{G}$ may have 2^N states. This happens if G is a parallel episode. In practice, as we will see in the experiments, this is not a problem since N is typically small.

Exponentiality is (most likely) unavoidable since testing whether a sequence covers an episode is known to be **NP**-hard problem (Tatti and Cule, 2011), and since we can use $M_\mathcal{G}$ to test coverage in polynomial time w.r.t. the states in $M_\mathcal{G}$ we must have episodes for which we have exponential number of states.

Simplifying $M_\mathcal{G}$ may also lead to an exponential number of nodes. This may happen if we have a lot of unrelated nodes with same labels. Typically, this will not happen, especially, if the sequence has a large alphabet. Moreover, we can avoid this problem by mining only strict episodes (Tatti and Cule, 2012) in which we require that if there are two nodes with the same label, then one of the nodes must be an ancestor of the other. For such episodes, $M_{\mathcal{G}}$ is already simple.

Computing a joint machine $co(M, M)$ may result into a machine having $\left|V(M)\right|^2$ states. In practice, the amount of states in $minm(G)$ is much smaller since not all pairs are considered. Similarly, a machine needed for computing cross-moments may have $O(\left|V(minm(G))\right|^2)$ nodes. We will see that in our experiments the number of states and edges remains small, making the method fast in practice.

6 Related Work

Our approach can be seen as an extension of (Tatti, 2009) where we developed a statistical test based on average length of minimal windows. We used a recursive update similar to the one given in Proposition 6, however we capped the length of minimal windows and computed explicitly the probabilities of an episode having a minimal window of a certain length. In this work we avoid this by using Proposition 7. Additional limitation of (Tatti, 2009) is that we were forced to simulate cross-moments where in this work we compute them analytically.

Statistical measures for ranking episodes have been considered by Gwadera et al (2005b,a) in which the authors considered episode to be significant if the episode occurs too often or not often enough in windows of fixed size. As a background model the authors used independence model in (Gwadera et al, 2005b) and Markov-chain model in (Gwadera et al, 2005a). The authors' approach in (Gwadera et al, 2005b) is similar to ours: First they construct a finite state machine, essentially $M_{\mathcal{G}}$, and use recursive update similar Proposition 6 in order to compute the mean, that is, the likelihood that the sequence of length L covers the episode under independence assumption. The main difference between our approach and theirs is that we base our measure directly on compactness, the average length of a minimal window, while they base their measure on occurrence, that is, in how many windows the episode occurs.

Working with the general episodes is difficult for two main reasons. Firstly, general episodes are more prone to suffer from pattern explosion due to the fact that there are so many directed acyclic graphs. Secondly, the simplest task such as testing whether a sequence contains an episode is a **NP**-hard problem (Tatti and Cule, 2011). Several subclasses of general episodes have been suggested. Pei et al (2006) suggested mining episodes from set of strings, sequences of unique symbols. Tatti and Cule (2012) suggested discovering closed strict episodes. An episode is strict if two nodes with the same label are always connected. Achar et al (2012) suggested discovering episodes with unique labels possibly with some additional constraints, for example, the number of paths in a DAG. The authors suggested a score based on how evenly unconnected nodes occur in front of each other. Tatti and Cule (2011) considered a broader class of episodes in which nodes are allowed to have multiple labels.

Casas-Garriga (2003) proposed a criterion for episodes by requiring that the consecutive symbols in a sequence should only within a specified bound. While this approach attacks the problem of fixed windows, it is still a frequency-based measure. This measure, however, is not monotonic as it is pointed out by Méger and Rigotti (2004). It would be useful to see whether we can compute an expected value of this measure so that we can compute a P-value based on some background model.

In a related work, Cule et al (2009) considered parallel episodes significant if the smallest window containing each occurrence of a symbol of an episode had a small value. Their approach differ from ours since the smallest window containing a fixed occurrence of a symbol is not necessarily the minimal window. Also, they consider only parallel episodes whereas we consider more general DAG episodes. An interesting approach has been also taken by Calders et al (2007) where the authors define a windowless frequency measure of an itemset within a stream s to be the frequency starting from a certain point. This point is selected so that the frequency is maximal. However, this method is defined for itemsets and it would be fruitful to see whether this idea can be extended into episodes.

Finite state machines have been used by Tronícek (2001); Hirao et al (2001) for discovering episodes. However, their goal is different than ours since the actual machine is built upon a sequence and not the episode set and it is used for discovering episodes and not computing the coverage.

7 Experiments

In this section we present our experiments with the quality measure using synthetic and real-world text sequences.

7.1 Datasets

We conducted our experiments with several synthetic and real-world sequences.

The first synthetic sequence, *Ind* consists of 40 000 events drawn independently and uniformly from an alphabet of 1 000 symbols. The second synthetic sequence, *Plant* also contains 40 000 events independently and uniformly from an alphabet of 1 000 symbols but in addition we planted 5 serial episodes. Each episode consisted of 5 nodes, each node with a unique label. We planted each episode 100 times and we added a gap between two consecutive events with a 10% probability.

Our third dataset, *Moby*, is the novel Moby Dick by Herman Melville.[1] Our fourth sequence, *Nsf* consists of 739 first NSF award abstracts from 1990.[2]

[1] The book was obtained from http://www.gutenberg.org/etext/15.

[2] The abstracts were obtained from http://kdd.ics.uci.edu/databases/nsfabs/nsfawards.html

Our final dataset, *Address*, consists of inaugural addresses of the presidents of the United States.[3] To avoid the historic concept drift—early speeches have different vocabulary than the later ones—we entwined the speeches by first taking the odd ones and then even ones. Our fourth dataset, *Jmlr*, consists of abstracts from Journal of Machine Learning Research.[4] The sequences were processed using the Porter Stemmer and the stop words were removed. The basic characteristics of sequences are summarised in Table 1.

Table 1: Characteristics of the sequences. The second column contains the number of symbols in the sequence. The third column contains the size of the alphabet of each sequence.

Sequence	length	$\|\Sigma\|$
Ind	40 000	1 000
Plant	40 000	1 000
Moby	105 719	10 277
Address	62 066	5 295
Jmlr	75 646	3 859
Nsf	35 370	4 592

7.2 Experimental Setup

Our experimental setup mimics the framework setup by Webb (2007) in which the data is divided into two parts, the first part is used for discovering the patterns and the second part for testing whether the discovered patterns were significant. We divided each sequence into two parts of equivalent lengths. We used the first sequence for discovering the candidate episodes and training the independence model. Then we tested the discovered episodes against the model using the second sequence. We set parameter ρ to $1/2$.

To generate candidate episodes we used a miner given by Tatti and Cule (2012). This miner discovers episodes in a breath-first fashion, that is, an episode is tested if and only if all its sub-episodes are frequent. The miner outputs closed[5] and strict episodes. Requiring episodes to be closed reduces redundancy between candidates considerably as there are typically many episodes describing the same set of minimal windows. The alphabet is large in our sequences, which implies that it is quite unlikely to see the same symbol twice within a short window. Consequently, there are only few non-strict frequent episodes.

[3] The addresses were obtained from http://www.bartleby.com/124/pres68.

[4] The abstracts were obtained from http://jmlr.csail.mit.edu/

[5] An episode is closed if there are no superepisode with the same support.

As a constraint we required that the number of non-overlapping minimal windows must exceed certain threshold in the first sequence. This is a monotonic condition that allows us to discover all candidates efficiently. During mining we also put an upper limit for minimal windows. The parameters and the numbers of candidates are given in Table 2.

Table 2: Parameters used for mining candidate episodes. The second column contains the allowed maximal length of a minimal window during mining. The third column contains threshold for the number of disjoint minimal windows. The fourth column contains the number of non-singleton episodes.

Sequence	max window	threshold	# of episodes
Ind	15	4	1 249
Plant	15	5	734
Moby	20	10	6 043
Address	20	4	41 888
Jmlr	20	10	14 528
Nsf	20	15	2 845

7.3 Computational complexity

Let us first study computational complexity in practice. As we pointed out earlier it is possible that sizes of structures needed to compute the score become exponentially large. To demonstrate the sizes in practice we computed the average number of states and edges in machines used to compute the score. The results are given in Table 3.

From these results we see that the number of nodes and edges stay small. This is due to the fact that majority of episodes are small, typically with 2–3 nodes. Simplification does not add any new nodes or edges since we use strict episodes, where nodes with the same label must be connected, consequently, $M_{\mathcal{G}}$ is simple. Number of nodes and edges are at highest for M^*, a machine needed to compute cross-moments for *Nsf* data. This is due to the fact that *Nsf* contains a lot of phrases where the same words are being repeated. As a consequence, we discover large episodes which in turn generate large machines. Running times given in the last column of Table 3 imply that ranking is fast. Ranking discovered episodes is done within few seconds. For example, in *Address* ranking 40 000 episodes takes less than 5 seconds.

We consider only closed and strict episodes as candidates. If we consider also non-closed episodes, then the distribution of episode types may change as long closed episodes tend to be serial. Consequently, we will have more general episodes. This may result in larger machines as serial episodes have the simplest machines.

Table 3: Average sizes of machines used for ranking episodes. Even columns, labelled with $|V|$, contain the number of nodes, while odd columns, labelled with $|E|$, contain the number of edges. The first machine $M_\mathcal{G}$ recognises when episode is covered, the second machine $sm(M_\mathcal{G})$ is a simplification of $M_\mathcal{G}$. The third machine $minm(G)$ tests whether a sequence is a minimal window, and the last machine M^* is used for computing cross-moments, see Section 5.3. The last columns is the time needed to rank the discovered episodes per dataset.

	$M_\mathcal{G}$		$sm(M_\mathcal{G})$		$minm(\mathcal{G})$		M^*		
Sequence	$\|V\|$	$\|E\|$	$\|V\|$	$\|E\|$	$\|V\|$	$\|E\|$	$\|V\|$	$\|E\|$	time (s)
Ind	3.8	3.7	3.8	3.7	4.7	3.7	3.5	1.7	0.34
Plant	4.4	4.5	4.4	4.5	6.6	6.6	11.5	13.3	0.28
Moby	3.9	3.6	3.9	3.6	4.7	3.9	4.1	2.4	1.37
Address	4.6	5	4.6	5	6.6	6.3	8.7	7.4	4.40
Jmlr	4.7	5	4.7	5	6.6	6.2	8.5	6.8	3.55
Nsf	7.3	9.7	7.3	9.7	14.3	18.1	39.6	49.3	1.09

7.4 Significant Episodes

Let us first consider *Plant* dataset. The first 5 episodes according to our ranking were exactly the planted patterns. The scores of these patterns are between 99 500 and 84 000. The following patterns are typically a combination of an original pattern with an additional parallel symbol or a subset of an original pattern. The scores of these patterns, though significant, are dropping fast: the score of the 6th pattern is 67 000, the score of 7th pattern is 42 000. Note that if we used frequency (or any other monotonic measure) as a score, subsets of these planted patterns would have appeared first in the list.

Our next step is to see what types of episodes does our score preferred. In order to do that, we first consider Figure 7 where we have plotted the number of nodes in an episode as a function of rank. We see that top patterns tend to have more nodes. This is especially prominent with *Address* and *Nsf* datasets.

We continued our experiments by computing the proportion of episode types, that is, whether an episode is a parallel, serial, or general, as a function of rank, given in Figure 8. From figures we see that distribution depends heavily on a sequence. Serial episodes tend to be distributed evenly, parallel episodes tend to be missing from the very top and general episodes tend to be missing from the very bottom.

Finally, let us conclude by demonstrating some of the discovered top patterns from *Address* and *Jmlr* datasets, given in Figure 9. The first three patterns represent phrases that are often said by the presidents. Episode in Figure 9b is particularly interesting since presidents tend to acknowledge vice president(s) and the chief justice at the beginning of their speeches but the order is not fixed. The remaining 3 patterns represent common phrases occurring in abstracts of machine learning articles.

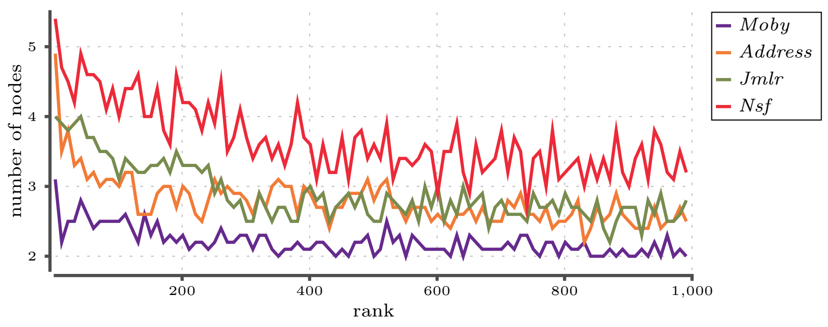


Fig. 7: Number of nodes in top-1 000 episodes as a function of rank. Counts are smoothed by computing averages of batches of ten episodes

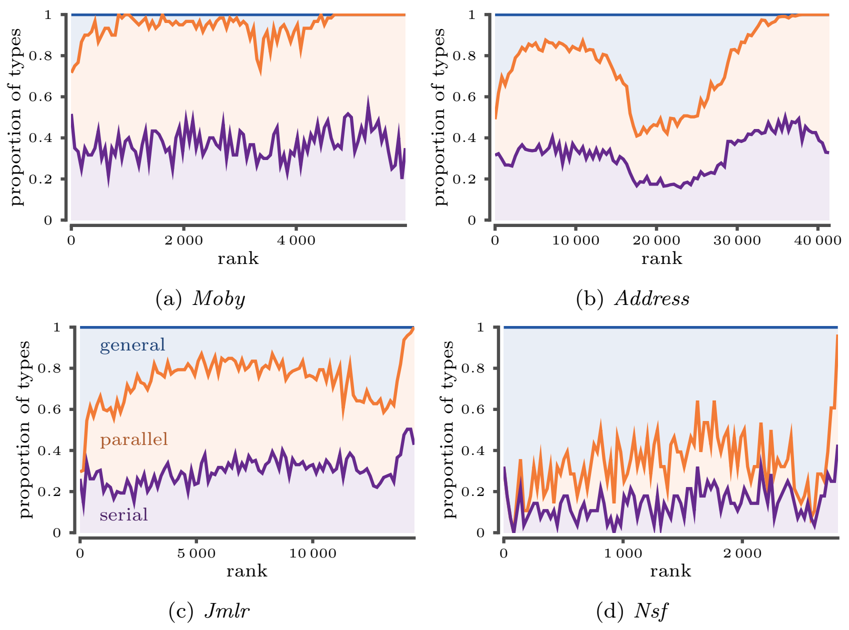


Fig. 8: Proportions of different types of episodes as a function of rank. The top area corresponds go the general episodes, the middle area represents parallel episodes and the bottom area represents serial episodes. Proportions were computed by dividing the ranked patterns into 100 bins

7.5 Asymptotic normality

Proposition 11 implies that if the independence assumption hold in the testing sequence, then $sc(G)$ should behave like a sample from a standard normal

$$\begin{tikzcd}\text{preserv} \arrow[r] & \text{protect} \arrow[r] & \text{defend} \arrow[r] & \text{constitut} \arrow[r] & \text{unit} \arrow[r] & \text{state}\end{tikzcd}$$

(a) *Address*, 1st episode, $sc(G) = 766\,946$

$$\begin{tikzcd}\text{vice} \arrow[r] & \text{presid} \\ \text{chief} \arrow[r] & \text{justice}\end{tikzcd}$$

(b) *Address*, 3rd, $sc(G) = 11\,483$

$$\begin{tikzcd}\text{four} \arrow[r] & \text{year} \arrow[r] & \text{ago}\end{tikzcd}$$

(c) *Address*, 7th episode, $sc(G) = 807$

$$\begin{tikzcd}\text{reproduce} \arrow[r] & \text{kernel} \arrow[r] & \text{hilbert} \arrow[r] & \text{space}\end{tikzcd}$$

(d) *Jmlr*, 1st episode, $sc(G) = 10\,971$

$$\begin{tikzcd}\text{support} \arrow[r] & \text{vector} \arrow[r] & \text{machin} \arrow[r] & \text{svm}\end{tikzcd}$$

(e) *Jmlr*, 2nd episode, $sc(G) = 10\,641$

$$\begin{tikzcd}\text{real} \arrow[r] & \text{world} \arrow[r] & \text{data} \arrow[r] & \text{set}\end{tikzcd}$$

(f) *Jmlr*, 3rd episode, $sc(G) = 4\,269$

Fig. 9: Examples of highly ranked episodes from *Address* and *Jmlr* datasets

distribution as the size of the sequence increases. In this section we test the rate of convergence.

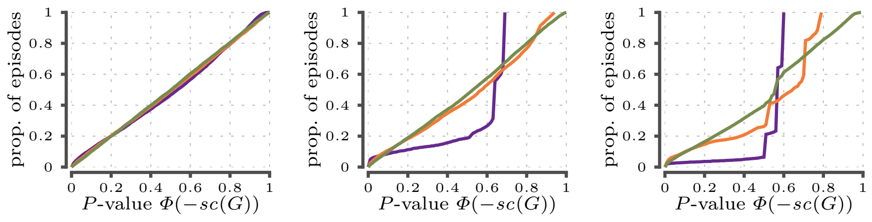
To that end we generated several sequences with independent events, each event having equal probability to occur. We generated three training sequences from alphabets of 100, 500, and 1 000 symbols. Each sequence contained 10 000 events. For each training sequence we generated 3 testing sequences of different lengths, namely 10^4, 10^5, and 10^6.

From each testing sequence we mined frequent episodes. We selected the thresholds such that we got roughly 10 000 episodes, more specifically, we used 12, 3, 2 as thresholds for sequences with 100, 500, 1 000 symbols respectively. We then tested the discovered non-singleton episodes on testing sequences. Note that computing the score requires probabilities of individual events. We computed the scores both by using the true probabilities and by estimating the probabilities from the training sequence.

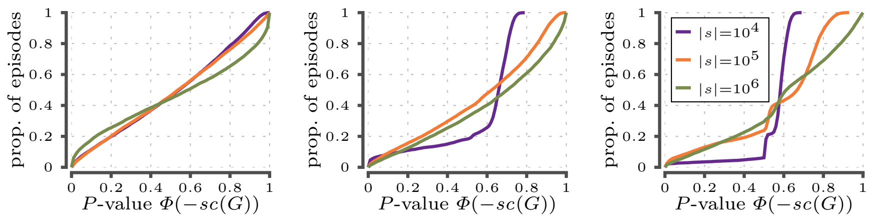
In Figure 10 we plotted the proportion of episodes for which $\Phi\left(-sc(G)\right)$ is smaller than the threshold. Proposition 11 implies that ideally this plot should be the identity line between 0 and 1. We see that this is the case in Figure 10a. As we increase the size of the alphabet, the estimate becomes more and more inaccurate. We believe that this is due to high skewness of the actual distribution. When using true probabilities for individual probabilities, longer testing sequences produce better results. Using estimated values introduces additional errors, as can be seen in Figure 10d where a testing sequence of length 10^6 is less ideal than the sequence of 10^5. However, this phenomenon can be attacked by dividing the sequence to training and testing portion more fairly, thus making the estimates more accurate.

8 Discussion and Conclusions

In this paper we proposed a new quality measure for episodes based on minimal windows. In order to do this, we approached by computing the expected values based on the independence model and compared the expectations to the observed values by computing a Z-score.



(a) $|\Sigma| = 100$, true distr. (b) $|\Sigma| = 500$, true distr. (c) $|\Sigma| = 1\,000$, true distr.



(d) $|\Sigma| = 100$, learned distr. (e) $|\Sigma| = 500$, learned distr. (f) $|\Sigma| = 1\,000$, learned distr.

Fig. 10: Cumulative proportion of episodes as a function of a score $\Phi\left(-sc(G)\right)$ in generated sequences with independent events. Ideally, the proportion is an identity function. The left column represents sequences with 100 symbols, the centre column represents sequences with 500 symbols, and the right column represents sequences with 1 000 symbols. The top row uses true occurrences for individual symbols when computing the moments, while the bottom row estimates the occurrences from training sequence. Each plot contains three lines representing different sizes of testing sequences

Our main technical contribution is a technique for computing the moments of minimal windows. In order to do so we created a series of elaborate finite state machines and demonstrated that we can compute the moments recursively. In this paper we chose to use a specific statistic, namely ρ^d, where d is a length of a minimal window and ρ is a user-given parameter. However, the same principle can be applied also directly on the length of minimal windows.

While the actual computation of statistics is fairly complex and requires a great number of recursive updates, and even may be exponentially slow, our experiments demonstrate that the computation is fast in practice, we can rank tens of thousands of episodes in the matter of seconds.

Our technique has its limitations. In synthetic data, *plant*, after finding 5 true patterns, our method continued scoring high patterns that were either superpatterns of subpatterns of the first 5 patterns. All these patterns are significant in the sense that they deviate significantly from the independence model. Nevertheless, they provide no new information about the underlying structure in the data. This problem occurs in any pattern ranking scheme where the ranking method does not take other patterns into account.

Approaches to further reduce patterns by considering patterns as a set instead of individual patterns have been developed for itemsets. For example, one approach for itemsets involve in partitioning itemsets into subitemsets and applying independence assumption between the individual parts Webb (2010). Transforming this idea to episodes is not trivial. A more direct approach—although using only serial episodes—where episodes were selected using MDL techniques was suggested in Tatti and Vreeken (2012). An extension of this work to general episodes would be interesting.

Proposition 11 implies that we can interpret our measure as a P-value. In practice, this can be problematic as we demonstrate in Section 7.5. Since the distributions are heavily skewed, especially when dealing with a large alphabet, we require a lot of samples before the normality assumption becomes accurate. Nevertheless our experiments with synthetic and text data demonstrate that our score produces interpretable rankings.

Acknowledgements

Nikolaj Tatti was partly supported by a Post-Doctoral Fellowship of the Research Foundation – Flanders (FWO).

References

Achar A, Laxman S, Viswanathan R, Sastry PS (2012) Discovering injective episodes with general partial orders. Data Min Knowl Discov 25(1):67–108

Billingsley P (1995) Probability and Measure, 3rd edn. John Wiley & sons

Calders T, Dexters N, Goethals B (2007) Mining frequent itemsets in a stream. In: Proceedings of the 7th IEEE International Conference on Data Mining (ICDM 2007), pp 83–92

Casas-Garriga G (2003) Discovering unbounded episodes in sequential data. In: Knowledge Discovery in Databases: PKDD 2003, 7th European Conference on Principles and Practice of Knowledge Discovery in Databases, pp 83–94

Cule B, Goethals B, Robardet C (2009) A new constraint for mining sets in sequences. In: Proceedings of the SIAM International Conference on Data Mining (SDM 2009), pp 317–328

Gwadera R, Atallah MJ, Szpankowski W (2005a) Markov models for identification of significant episodes. In: Proceedings of the SIAM International Conference on Data Mining (SDM 2005), pp 404–414

Gwadera R, Atallah MJ, Szpankowski W (2005b) Reliable detection of episodes in event sequences. Knowledge and Information Systems 7(4):415–437

Hirao M, Inenaga S, Shinohara A, Takeda M, Arikawa S (2001) A practical algorithm to find the best episode patterns. In: Discovery Science, pp 435–440

Mannila H, Toivonen H, Verkamo AI (1997) Discovery of frequent episodes in event sequences. Data Mining and Knowledge Discovery 1(3):259–289, DOI http://dx.doi.org/10.1023/A:1009748302351

Méger N, Rigotti C (2004) Constraint-based mining of episode rules and optimal window sizes. In: Knowledge Discovery in Databases: PKDD 2004, 8th European Conference on Principles and Practice of Knowledge Discovery in Databases, pp 313–324

Pei J, Wang H, Liu J, Wang K, Wang J, Yu PS (2006) Discovering frequent closed partial orders from strings. IEEE Transactions on Knowledge and Data Engineering 18(11):1467–1481

Tatti N (2009) Significance of episodes based on minimal windows. In: Proceedings of the 9th IEEE International Conference on Data Mining (ICDM 2009), pp 513–522

Tatti N, Cule B (2011) Mining closed episodes with simultaneous events. In: Proceedings of the 17th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD 2011), pp 1172–1180

Tatti N, Cule B (2012) Mining closed strict episodes. Data Min Knowl Discov 25(1):34–66

Tatti N, Vreeken J (2012) The long and the short of it: summarising event sequences with serial episodes. In: The 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2012, pp 462–470

Tronícek Z (2001) Episode matching. In: Combinatorial Pattern Matching, pp 143–146

van der Vaart AW (1998) Asymptotic Statistics. Cambridge Series in Statistical and Probabilistic Mathematics, Cambridge University Press

Webb GI (2007) Discovering significant patterns. Machine Learning 68(1):1–33

Webb GI (2010) Self-sufficient itemsets: An approach to screening potentially interesting associations between items. TKDD 4(1)

A Proofs

Proof (Proof of Proposition 2) We will prove this by induction. Let i be the source state of M. The proposition holds trivially when $X = \{i\}$, a source state. Assume now that the proposition holds for all parent states of X.

Assume that s covers X. Let t be a subsequence of s that leads $sm(M)$ from the source state $\{i\}$ to X. Let s_e be the last symbol of s occurring in t. Then a parent state $Y = \{y_1, \ldots, y_L\} = par(X; s_e)$ is covered by $s[1, e-1]$. By the induction assumption at least one y_k is covered by $s[1, e-1]$. If there is $x_j \in X$ such that $x_j = y_k$, then x_j is covered by s, otherwise there is x_j that has y_k as a parent state. The edge connecting x_j and y_k is labelled with s_e. Hence s covers x_j also.

To prove the other direction assume that s covers x_j. Let t be a sub-sequence that leads M from i to x_j. Let s_e be the last symbol occurring in t. Let y be the parent state of x_j connected by an edge labelled with s_e. Since $s_e \in in(X)$, we must have Y as a parent state of X such that $y \in Y$. By the induction assumption, $s[1, e-1]$ covers Y. Hence s covers X. □

In order to prove Proposition 3 we need the following lemma.

Lemma 2 *Let G be an episode and assume a sequence $s = (s_1, \ldots, s_L)$ that covers G. Let $\mathcal{H} = \{G - v; v \in \mathit{sinks}(G)\,, \mathit{lab}(v) = s_L\}$. If $\mathcal{H}$ is empty, then $s[1, L-1]$ covers G. Otherwise, there is an episode $H \in \mathcal{H}$ that is covered by $s[1, L-1]$.*

Proof Let f be a valid mapping of $V(G)$ to indices of s corresponding to the coverage. If $\mathcal{H}$ is empty, then L is not in the range of f, then $s[1, L-1]$ covers G. If $\mathcal{H}$ is not empty but L is not in the range of f, then $s[1, L-1]$ covers G, and any episode in $\mathcal{H}$.

Assume now that L is in range of f, that is, there is a sink v with a label s_L. Episode $G - v$ is in $\mathcal{H}$. Moreover, f restricted to $G - v$ provides the needed mapping in order to $s[1, L-1]$ to cover $G - v$. □

Proof (Proof of Proposition 3) If $g(X, s) = \{i\}$, then it is trivial to see that s covers X.

Assume that s covers X. We will prove this direction by induction over L, the length of s. The proposition holds for $L = 0$. Assume that $L > 0$ and that proposition holds for all sequences of length $L - 1$.

Let $Y = g(X, s_L)$. Note that $g(X, s) = g(Y, s[1, L-1])$. Hence, to prove the proposition we need to show that $s[1, L-1]$ covers Y.

If $Y = \{i\}$, then $s[1, L-1]$ covers Y. Hence, we can assume that $Y \neq \{i\}$, that is, $Y = \mathit{sub}(X; s_L) \cup \mathit{stay}(X; s_L)$.

Proposition 2 implies that one of the states of M_G, say $x \in X$, is covered by s. Proposition 1 states that the corresponding episode, say H, is covered by s.

Assume that $x \in Y$. This is possibly only if $x \in \mathit{stay}(X; s_L)$ that is there is no sink node in H labelled as s_L. Lemma 2 implies that $s[1, L-1]$ covers H, Propositions 1 and 2 imply that $s[1, L-1]$ covers Y.

Assume that $x \notin Y$, Then $\mathit{sub}(X; s_L) \subseteq Y$ contains all states of M_G corresponding to the episodes of form $H - v$, where v is sink node of H with a label s_L. According to Lemma 2, $s[1, L-1]$ covers one of these episodes, Propositions 1 and 2 imply that $s[1, L-1]$ covers Y. □

Proof (Proof of Proposition 4) We will prove the proposition by induction over L, the length of s. The proposition holds when $L = 0$. Assume that $L > 0$ and that proposition holds for sequence of length $L - 1$.

Let $\beta = (y_1, y_2) = g(\alpha, s_L)$. Then, by definition of M^*, $y_i = g(x_i, s_L)$. Write $t = s[1, L-1]$. Since

$$g(\beta, t) = g(\alpha, s)\,, \quad g(y_1, t) = g(x_1, s)\,, \quad g(y_2, t) = g(x_2, s)\,.$$

and, because of induction assumption, $g(\beta, t) = (g(y_1, t)\,, g(y_2, t))$, we have $g(\alpha, s) = (g(x_1, s)\,, g(x_2, s))$. □

Proof (Proof of Proposition 5) Assume that s is a minimal window for G. Since s covers S in M, $g(S, s; M) = I$. This implies that $g(S, s; M_1) = I$ or $g(S, s; M_1) = J$. The latter case implies that $s[2, L]$ covers S in M, which is a contradiction. Hence, $g(S, s; M_1) = I$. Let $Z = g(T, s; M_2)$. If $Z = I$, then $s[1, L-1]$ covers S in M, which is a contradiction. Hence $Z \neq I$. Proposition 4 implies that $g(\alpha, s) = (I, Z)$.

Assume that $g(\alpha, s) = (I, Y)$ such that $Y \neq I$. Proposition 4 implies that $g(S, s; M_1) = I$ and $g(T, s; M_2) \neq I$. The former implication leads to $g(S, s; M) = I$ which implies that s covers G.

If $s[2, L]$ covers G, then $g(S, s[2, L]; M) = I$ and so $g(S, s; M_1) = J$, which is a contradiction. Hence $s[2, L]$ does not cover G. The latter implication leads to $g(S, s[1, L-1]; M) \neq I$ which implies that $s[1, L-1]$ does not cover G. This proves the proposition. □

Proof (Proof of Proposition 6) If $L = 0$, then $g(x, s) = x$ which immediately implies the proposition. Assume that $L > 0$. Note that $g(x, s) = g(g(x, s_L)\,, s[1, L-1])$.

$$\begin{aligned}
&p(g(x, s) \in Y \mid |s| = L) \\
&\quad = \sum_{a \in \Sigma} p(a) p(g(x, s) \in Y \mid |s| = L, s_L = a) \\
&\quad = \sum_{a \in \Sigma} p(a) p(g(g(x, a)\,, s[1, L-1]) \in Y \mid |s| = L, s_L = a).
\end{aligned}$$

Since individual symbols in s are independent, it follows that

$$p(g(g(x,a)\,, s[1,L-1]) \in Y \mid |s| = L, s_L = a) = pg(g(x,a)\,, Y, L-1)\,.$$

This proves the proposition. □

Proof (Proof of Lemma 1) Define $q = \sqrt{1 - \min_{a \in \Sigma} p(a)}$. Note that $q < 1$. We claim that for each x there is a constant C_x such that $pg(x, Y, L) \leq C_x q^L = O(q^L)$ which in turns proves the lemma. To prove the claim we use induction over parenthood of x and L.

Since the source node is not in Y, the first step follows immediately. Assume that the result holds for all parent states of x. Define

$$C_x = \max\big(1, \frac{1}{q(1-q)} \sum_{\substack{a \in in(x)\\ y = g(x,a)}} p(a) C_y\big) \text{ which implies } qC_x + q^{-1} \sum_{\substack{a \in in(x)\\ y = g(x,a)}} p(a) C_y \leq C_x.$$

Since $C_x \geq 1$, the case of $L = 0$ holds. Assume that the the induction assumption holds for C_y and for C_x up to $L - 1$. Let $r = 1 - \sum_{a \in in(x)} p(a)$. Note that $r \leq q^2$. According to Proposition 6 we have

$$\begin{aligned} pg(x, Y, L) &= rpg(x, Y, L-1) + \sum_{\substack{a \in in(x)\\ y = g(x,a)}} p(a) pg(y, Y, L-1) \\ &\leq rC_x q^{L-1} + \sum_{\substack{a \in in(x)\\ y = g(x,a)}} p(a) C_y q^{L-1} \\ &\leq q^L \big(qC_x + q^{-1} \sum_{\substack{a \in in(x)\\ y = g(x,a)}} p(a) C_y\big) \leq q^L C_x. \end{aligned}$$

This proves that $pg(x, Y, L)$ decays at exponential rate. □

Proof (Proof of Proposition 8) The proposition follows by a straightforward manipulation of Equation 1. First note that

$$\sum_{L=1}^{\infty} f(L-1) pg(x, Y, L) = cm(x, f, Y) + m(x, h, Y)\,. \tag{5}$$

Equation 1 implies that

$$\begin{aligned} \sum_{L=1}^{\infty} f(L-1) pg(x, Y, L) &= \sum_{\substack{a \in \Sigma\\ y = g(x,a)}} p(a) \sum_{L=1}^{\infty} f(L-1) pg(y, Y, L-1) \\ &= \sum_{\substack{a \in \Sigma\\ y = g(x,a)}} p(a)(i(y) + \sum_{L=1}^{\infty} f(L) pg(y, Y, L)) \\ &= \sum_{\substack{a \in \Sigma\\ y = g(x,a)}} p(a)(i(y) + m(y, f, Y)) \\ &= q(i(x) + m(x, f, Y)) + \sum_{\substack{a \in in(x)\\ y = g(x,a)}} p(a)(i(y) + m(y, f, Y)). \end{aligned} \tag{6}$$

Combining Equations 5 and 6 and solving $m(x, f, Y)$ gives us the result. □

To prove the asymptotic normality we will use the following theorem.

Theorem 1 (Theorem 27.4 in (Billingsley, 1995)) *Assume that U_k is a stationary sequence with $\mathrm{E}[U_k] = 0$, $\mathrm{E}\left[U_k^{12}\right] < \infty$, and is α-mixing with $\alpha(n) = O(n^{-5})$, where $\alpha(n)$ is the strong mixing coefficient,*

$$\alpha(n) = \sup_{k,A,B} |p(A, B) - p(A)p(B)|,$$

where A is an event depending only on $U_{-\infty}, \ldots, U_k$ and B is an event depending only on $U_{k+n}, \ldots, U_{\infty}$. Let $S_k = U_1 + \cdots + U_k$. Then $\sigma^2 = \lim_k 1/k\,\mathrm{E}[S_k]$ exists and $S_k/\sqrt{k}$ converges to $N(0, \sigma^2)$ and $\sigma^2 = \mathrm{E}\left[U_1^2\right] + 2\sum_{k=2}^{\infty} \mathrm{E}[U_1 U_k]$.

Proof (Proof of Proposition 10) Let us write $T_k = (Z_k, X_k) - (q, p)$ and $S_L = 1/\sqrt{L}\sum_{k=1}^{L} T_k$. Assume that we are given a vector $r = (r_1, r_2)$ and write $U_k = r^T T_k$. We will first prove that $r^T S_L$ converges to a normal distribution using Theorem 1.

First note that $\mathrm{E}[U_k] = 0$ and that

$$\mathrm{E}\left[U_k^{12}\right] = \sum_{i=0}^{12} \binom{12}{i} r_1^i r_2^{12-i}\, \mathrm{E}\left[Z_k^i X_k^{12-i}\right] = r_2^{12}\,\mathrm{E}[X_k] + \sum_{i=1}^{12} \binom{12}{i} r_1^i r_2^{12-i}\, \mathrm{E}\left[Z_k^i\right].$$

Since every moment of Z_k and X_k is finite, $\mathrm{E}\left[U_k^{12}\right]$ is also finite. We will prove now that U_k is α-mixing.

Fix k and N. Write W to be an event that $s[k + 1, N]$ covers G. If W is true, then X_l and Z_l (and hence U_l) for $l \leq k$ depends only $s[l, N]$, that is, either there is a minimal window $s[l, N']$, where $N' < N$ or $X_l = Z_l = 0$.

Let A be an event depending only on $U_{-\infty}, \ldots, U_k$ and B be an event depending only on $U_{N+1}, \ldots, U_{\infty}$. Then $p(A, B \mid W) = p(A \mid W)p(B \mid W)$. We can rephrase this and bound $\alpha(n) \leq p(s[1, n-1]$ does not covers $G)$. To bound the right side, let $M = sm(M_G)$, let v be its sink state and let V be all states save the source state. Then the probability is equal to

$$p(s[1, n-1] \text{ does not covers } G) = pg(v, V, n-1)\,.$$

Since V does not contain the source node, the moment $m\big(v, n \to n^5, V\big)$ is finite. Consequently, $n^5 pg(v, V, n) \to 0$ which implies that $\alpha(n) = O(n^{-5})$. Thus Theorem 1 implies that $r^T S_L$ converges to a normal distribution with the variance $\sigma^2 = r_1^2 C_{11} + 2r_1 r_2 C_{12} + r_2^2 C_{22} = r^T C r$. Levy's continuity theorem (Theorem 2.13 van der Vaart, 1998) now implies that the characteristic function of $r^T S_L$ converges to a characteristic function of normal distribution $N(0, \sigma^2)$,

$$\mathrm{E}\left[\exp\Big(itr^T S_L\Big)\right] \to \exp\Big(-1/2t^2 r^T C r\Big)\,.$$

The left side is a characteristic function of S_L (with tr as a parameter). Similarly, the right side is a characteristic function of $N(0, C)$. Levy's continuity theorem now implies that S_L converges into $N(0, C)$. □

Proof (Proof of Proposition 11) Function $f(x, y) = x/y$ is differentiable at (q, p). Since $1/\sqrt{L}\big(\sum_{k=1}^{L}(Z_k, X_k) - (q, p)\big)$ converges to normal distribution, we can apply Theorem 3.1 in (van der Vaart, 1998) so that

$$\sqrt{L}\left(\frac{\sum_{k=1}^{L} Z_k}{\sum_{k=1}^{L} X_k} - \mu\right) = \sqrt{L} f\left(1/L\sum_{k=1}^{L} Z_k, 1/L\sum_{k=1}^{L} X_k\right) - \sqrt{L} f(q, p)$$

converges to $N(0, \sigma^2)$, where $\sigma^2 = \nabla f(q, p)^T C \nabla f(q, p)$. The gradient of f is equal to $\nabla f(q, p) = (1/p, -\mu/p)$. The proposition follows. □

Proof (Proof of Proposition 12) To prove all four cases simultaneously, let us write write A to be either X_1 or Z_1 and let B_k to be either X_k or Z_k. Let $a = \mathrm{E}[A]$ and $b = \mathrm{E}[B_k]$. First note that $\mathrm{E}[(A - a)(B_k - b)] = \mathrm{E}[A(B_k - b)]$, which allows us to ignore a inside the mean.

Assume that we have $0 < n < k$. Then given that $Y_1 = n$, A and X_1 depends only on n first symbols of sequence. Since B_k does not depend on $k-1$ first symbols, this implies that

$$p(A, B_k \mid Y_1 = n) = p(A \mid Y_1 = n) p(B_k \mid Y_1 = n) = p(A \mid Y_1 = n) p(B_k),$$

which in turns implies that $\mathrm{E}\left[A(B_k - b) \mid Y_1 = n\right] = 0$.

Note that for $A = 0$ whenever $Y_1 = 0$. Consequently, we have

$$\begin{aligned}
\mathrm{E}\left[A \sum_{k=2}^{\infty} (B_k - b)\right] &= \sum_{n=1}^{\infty} \mathrm{E}\left[A \sum_{k=2}^{\infty} (B_k - b) \mid Y_1 = n\right] p(Y_1 = n) \\
&= \sum_{n=1}^{\infty} \mathrm{E}\left[A \sum_{k=2}^{n} (B_k - b) \mid Y_1 = n\right] p(Y_1 = n) \\
&= \mathrm{E}\left[A \sum_{k=2}^{Y_1} (B_k - b)\right] = \mathrm{E}\left[A \sum_{k=2}^{Y_1} B_k\right] - \mathrm{E}\left[A \sum_{k=2}^{Y_1} b\right] \\
&= \mathrm{E}\left[A \sum_{k=2}^{Y_1} B_k\right] - \mathrm{E}\left[A(Y_1 - X_1)\right] b \\
&= \mathrm{E}\left[X_1 A \sum_{k=2}^{Y_1} X_k B_k\right] - \mathrm{E}\left[A(Y_1 - X_1)\right] b,
\end{aligned}$$

where the second last equality holds because $\sum_{k=2}^{Y_1} 1 = Y_1 - X_1$ and the last equality follows since $X_k = X_k^2$ and $Z_k = X_k Z_k$ for any k. □