A Non-Parametric Bayesian Approach Towards Online Sequence Learning

Stiven S. Dias
Embraer S.A.
R&D Team
Av. Brig. Faria Lima, 2170
S. J. dos Campos, Brazil
stiven.dias@embraer.com.br

Marcelo G. S. Bruno
Instituto Tecnológico de Aeronáutica
Pç. Mar. Eduardo Gomes, 50
S. J. dos Campos, Brazil
bruno@ita.br

Alberto F. De Souza
Universidade Federal do Espírito Santo
Av. Fernando Ferrari, 514
Vitória, Brazil
alberto@inf.ufes.br

ABSTRACT

Forecasting the next sample of an incoming sequence of noisy observations is a recurring problem in several real-world applications. In general, the streamed observations may originate from a possibly non-stationary stochastic process with non-linear dynamics. Very often though, to avoid devising explicit probabilistic models from scratch, we alternatively learn data-driven surrogate models from past observations to approximate complex dynamics. Time-varying systems in turn pose additional hurdles as such models must be continuously updated to accommodate changes in their underlying behavior. In this paper, we introduce a novel grid-based filter designed to inherently deal with the continuous learning problem. Specifically, Bayesian-grounded procedures are employed to both recursively predict observations and incrementally build a suitable non-parametric, probabilistic model of the indirectly observed phenomenon. Moreover, we also present a one-shot learning, memory-based implementation of the proposed filter, a shallow weightless neural network which is able to efficiently store and retrieve associative input-output pairs. Lastly, we demonstrate the effectiveness of our filter in performing simple anomaly detection tasks.

1. INTRODUCTION

Time-series modeling is paramount for many applications such as anomaly detection and prediction [1], stock market forecast [31] and even network attack detection [8]. Moreover, Bayesian inference – e.g. through the Bayes filter – provides a well-grounded theoretical framework relying on probabilistic models to forecast incoming observations of a stochastic phenomenon. However, in several engineering applications, designing an explicit probabilistic model of the time-series by hand is inadvisable as the data distribution can be prohibitively complex and often drifts, changing significantly over time. The ability to continuously learn suitable probabilistic models directly from available observations without supervision thus has the potential to further unlock the application of the Bayesian paradigm to time-series forecast.

1.1 Related Work

Online sequence learning narrows down the range of suitable methods to timely predict future observations, since they are also required to simultaneously update their inner models of the temporal phenomenon using accumulated observations. Several methods exist for time-series prediction using different approaches [23, 25]. Few of them like [1, 30] are able to deal with the continuous learning (CL) problem, i.e. to build and improve a feasible model to accurately predict future observations as incoming observations arrive. Even fewer approaches though rely on non-parametric statistics [32, 28] to simultaneously learn models and predict observations [7,9], but, to the best of our knowledge, none of them have adapted the Bayes filter to recursively learn model's hyperparameters and jointly make predictions using a fully non-parametric, Bayesian inference approach as discussed in this paper.

Furthermore, shallow weightless neural networks (WNNs) [2] have been successfully employed to learn from data and solve challenging problems [13,14,12] as far as proper data encoding schemes [27] are employed. Despite their inherent ability for both one-shot learning and CL, so far, their few enthusiasts have not been able to effectively stack WNN layers into deeper networks to solve increasingly complex problems, as was the case for regular neural networks. On the other hand, although quite useful to successfully solve real-world, time-series forecasting problems [10, 3], deep neural models were not intrinsically designed for CL, as they often require an offline, expensive optimization procedure [5] with salting racks and workarounds [6] to timely / properly converge and adjust their multitude of parameters – e.g. weights and biases – using all accumulated training data so far; avoiding thus catastrophic forgetting issues [22].

An alternative, bio-inspired model which supports CL by design has been incrementally polished by Numenta researchers [19, 20]. However, despite their insightful findings [18] on how the underlying biological mechanisms – in special, biological neurons and layers within cortical cells – of the human neocortex interact with each other to create human intelligence as an emerging behavior, they did not provide so far a rigorous interpretation of their well-known hierarchical temporal memory (HTM) model [19] using e.g. the Bayesian inference framework.

In this paper, we model first the time-series CL and prediction problem using a fully Bayesian approach and propose therefore an explainable non-parametric, grid-based filter accompanied by an efficient, one-shot learning implementation based on a shallow WNN with two layers stacked in a meaningful, Bayesian interpretable way. Nevertheless, we also borrow some ideas from Cui et al. [11] and employ sparse representations over high-dimensional spaces to represent observed symbols.

1.2 Contributions and Paper Outline

The main contribution of this paper is twofold: it introduces i) a novel grid-based filter which allows one to perform inference on high-dimensional binary spaces and effectively predict incoming observations residing in continuous-valued spaces; and ii) an efficient grid-filter implementation based on virtual-generalized random access memory (VG-RAM) nodes [24] which enables time-series CL seamlessly and yields (auditable) outputs which are fully traceable to filter's inputs.

Paper Outline: The paper is divided into six sections. Sec. 1 is this Introduction. In Sec. 2, we state the online sequence learning and prediction problem under a Bayesian framework. In Sec. 3, we introduce our solution approach. Sec. 4 describes our associative random access memory (RAM)-based implementation of the solution proposed in Sec. 3. Simulations results for an application example are presented in Sec. 5. Finally, we offer our conclusions in Sec. 6.

1.3 Notation

We use uppercase letters, e.g. $\mathbf{X}$, to denote random vectors and lowercase letters, e.g. $\mathbf{x}$, to denote samples of random vectors. We also define the unconditional probability of an event A and the conditional probability of event A given the knowledge that an event B has occurred as $\Pr\{A\}$ and $\Pr\{A|\,B\}$, respectively. Moreover, we define the joint probability of events A and B occurring simultaneously as $\Pr\{A, B\}$.

We also employ the Iverson bracket – which takes as argument any logical proposition Λ – defined as $[\![\Lambda]\!] = 1$, if Λ is true, and $[\![\Lambda]\!] = 0$, otherwise. Furthermore, let

$$\mathbb{Z}_2^d \triangleq \{0,1\} \times \ldots \times \{0,1\}$$

denote an d-dimensional binary space, we define then the Hamming distance between two vectors $\mathbf{a} \triangleq \begin{bmatrix} a_1 & \ldots & a_d \end{bmatrix}^\top$ and $\mathbf{b} \triangleq \begin{bmatrix} b_1 & \ldots & b_d \end{bmatrix}^\top$ in $\mathbb{Z}_2^d$ as $d_H(\mathbf{a}, \mathbf{b}) \triangleq \sum_{i=1}^{d} [\![a_i \neq b_i]\!]$. The inner product between any two vectors $\mathbf{a}$ and $\mathbf{b}$ now in $\mathbb{R}^d$ is defined as $\langle \mathbf{a}, \mathbf{b} \rangle \triangleq \sum_{i=1}^{d} a_i \cdot b_i$ and the L^p norm of an arbitrary vector $\mathbf{c} \triangleq \begin{bmatrix} c_1 & \ldots & c_d \end{bmatrix}^\top \in \mathbb{R}^d$ is defined as $||\mathbf{c}||_p \triangleq \left(\sum_{i=1}^{d} |c_i|^p \right)^{\frac{1}{p}}$. Lastly, we define the one-hot indicator vector as $\mathbf{1}_j \triangleq \begin{bmatrix} \iota_{1,j} & \ldots & \iota_{d,j} \end{bmatrix}^\top \in \mathbb{Z}_2^d$ with $||\mathbf{1}_j||_1 = 1$ such that $\iota_{i,j} = [\![i = j]\!]$, $\forall i \in \{1, \ldots, d\}$.

Now, let $\mathbf{X}$ be a d-dimensional random vector whose realizations $\mathbf{x} \sim \mathbf{X}$ assume values in a finite sampling space $\boldsymbol{\Omega} \triangleq \{\boldsymbol{\mu}^{(1)}, \ldots, \boldsymbol{\mu}^{(c)}\} \subseteq \mathbb{R}^d$ with cardinality $c = |\boldsymbol{\Omega}|$. We denote a categorical distribution over $\boldsymbol{\Omega}$ as $Cat(\boldsymbol{\Omega}; \boldsymbol{\pi})$, in which the c-dimensional parameter vector $\boldsymbol{\pi} \triangleq \begin{bmatrix} \pi_1 & \ldots & \pi_c \end{bmatrix}^\top \in \mathbb{R}_+^c$, $||\boldsymbol{\pi}||_1 = 1$, collects the probabilities of the c possible categories such that its probability mass function (p.m.f.) is given by $P(\mathbf{x}) \triangleq Pr\{\mathbf{X} = \mathbf{x}\} = \sum_{i=1}^{c} \pi_i \cdot [\![\mathbf{x} = \boldsymbol{\mu}^{(i)}]\!]$.

Moreover, let us further assume that $\boldsymbol{\pi}$, a valid p.m.f. over $\boldsymbol{\Omega}$, is a realization of a random vector $\boldsymbol{\Pi}$ in the $(c-1)$-dimensional simplex $\mathbb{S}_c$ distributed according to a Dirichlet distribution $Dir(c; \boldsymbol{\alpha})$ with hyperparameters collected by the vector $\boldsymbol{\alpha} \triangleq \begin{bmatrix} \alpha_1 & \ldots & \alpha_c \end{bmatrix}^\top$, $\alpha_i \in \mathbb{R}_+$, $\forall i \in \{1, \ldots, c\}$. We define the probability density function (p.d.f.) of the Dirichlet distribution over $\mathbb{S}_c$ as $p(\boldsymbol{\pi}) = (B(\boldsymbol{\alpha}))^{-1} \prod_{i=1}^{c} \pi_i^{\alpha_i - 1}$, where $B(\boldsymbol{\alpha}) \triangleq (\Gamma(\alpha_0))^{-1} \prod_{i=1}^{c} \Gamma(\alpha_i)$ is the Beta function with $\alpha_0 = \sum_{i=1}^{c} \alpha_i$ and $\Gamma(x) \triangleq \int_0^\infty t^{x-1} e^{-t} dt$ is the Gamma function with x in the set of non-negative real numbers $\mathbb{R}_+$.

2. PROBLEM STATEMENT

Let $\check{\mathbf{y}}_{0:\infty} \triangleq (\check{\mathbf{y}}_0, \check{\mathbf{y}}_1, \check{\mathbf{y}}_2, \ldots)$ be an *online* sequence of observations – e.g. a streamed sequence of dense feature vectors – in a continuous space $\boldsymbol{\Omega}_y \subset \mathbb{R}^{d_y}$ at discrete time instants $t_0 < t_1 < t_2 < \ldots$ Given the sub-sequence of observations $\check{\mathbf{y}}_{0:n} \triangleq (\check{\mathbf{y}}_0, \ldots, \check{\mathbf{y}}_n)$ up to instant n, our ultimate goal is to predict, at instant n, the next observation $\check{\mathbf{y}}_{n+1} \triangleq \begin{bmatrix} \check{y}_{n+1,1} & \ldots & \check{y}_{n+1,d_y} \end{bmatrix}^\top \in \boldsymbol{\Omega}_y$.

2.1 Bayesian formulation

Now, let $\mathbf{m}_{y \to z} : \boldsymbol{\Omega}_y \mapsto \boldsymbol{\Omega}_z$ denote a *known* encoding function which maps each observation $\check{\mathbf{y}}_n$ into a sparse representation $\check{\mathbf{z}}_n \triangleq \begin{bmatrix} \check{z}_{n,1} & \ldots & \check{z}_{n,d_z} \end{bmatrix}^\top$ within a high-dimensional binary space $\boldsymbol{\Omega}_z \subset \mathbb{Z}_2^{d_z}$ such that $d_z \gg d_y$ and $d_z \gg ||\check{\mathbf{z}}_n||_1$. Similarly, let $\mathbf{m}_{z \to y} : \boldsymbol{\Omega}_z \mapsto \boldsymbol{\Omega}_y$ define a *known* decoding function mapping arbitrary encoded observations $\mathbf{z}_n \in \boldsymbol{\Omega}_z$ back to observations $\mathbf{y}_n \in \boldsymbol{\Omega}_y$.

As illustrated in Fig. 1, let us further suppose that any particular sequence of encoded observations $\check{\mathbf{z}}_{0:n} \triangleq (\check{\mathbf{z}}_0, \ldots, \check{\mathbf{z}}_n)$ up to instant n is emitted by a Markovian process with an underlying sequence of hidden states $\mathbf{s}_{0:n} \triangleq (\mathbf{s}_0, \ldots, \mathbf{s}_n)$ within a high-dimensional binary space $\boldsymbol{\Omega}_s \subset \mathbb{Z}_2^{d_s}$, $\forall n \geq 0$, $d_s \gg ||\mathbf{s}_n||_1$, such that the online sequence of output observations can be written as

$$\check{\mathbf{y}}_{0:n} \equiv (\mathbf{m}_{z \to y}(\check{\mathbf{z}}_0), \ldots, \mathbf{m}_{z \to y}(\check{\mathbf{z}}_n)) \,. \tag{1}$$

Note that, as shown in Fig. 1, the sequences $\check{\mathbf{y}}_{0:n}$, $\check{\mathbf{z}}_{0:n}$ and $\mathbf{s}_{0:n}$ are realizations of the sequences of random vectors $\mathbf{Y}_{0:n} \triangleq (\mathbf{Y}_0, \ldots, \mathbf{Y}_n)$, $\mathbf{Z}_{0:n} \triangleq (\mathbf{Z}_0, \ldots, \mathbf{Z}_n)$ and $\mathbf{S}_{0:n} \triangleq (\mathbf{S}_0, \ldots, \mathbf{S}_n)$, respectively.

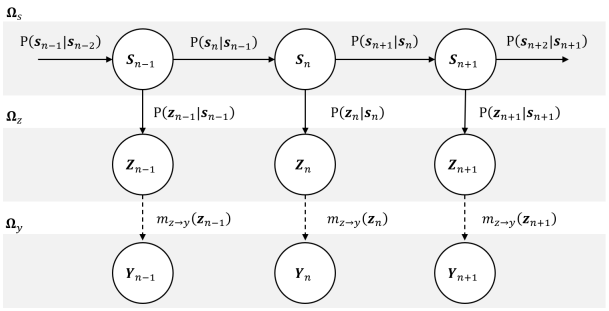

Figure 1: Hidden Markov chain representing the observed stochastic process.

Next, conditioned on the sequence of observations $\check{\mathbf{z}}_{0:n-1}$ up to instant $n-1$, the random state vector $\mathbf{S}_n$ at instant n is distributed according to the conditional p.m.f.

$$P(\mathbf{s}_n \,|\, \check{\mathbf{z}}_{0:n-1}) \triangleq Pr\{\mathbf{S}_n = \mathbf{s}_n \,|\, \mathbf{Z}_0 = \check{\mathbf{z}}_0, \ldots, \mathbf{Z}_{n-1} = \check{\mathbf{z}}_{n-1}\}. \tag{2}$$

Since, by construction, the observed process is first order Markovian and the observations are not correlated over time, the following usual assumptions hold respectively for the likelihood function and the state transition p.m.f.

$$P(\check{\mathbf{z}}_n \,|\, \mathbf{s}_{0:n}, \check{\mathbf{z}}_{0:n-1}) \equiv P(\check{\mathbf{z}}_n \,|\, \mathbf{s}_n) \tag{3}$$

$$P(\mathbf{s}_{n+1} \,|\, \mathbf{s}_{0:n}, \check{\mathbf{z}}_{0:n}) \equiv P(\mathbf{s}_{n+1} \,|\, \mathbf{s}_n). \tag{4}$$

Thus, upon the arrival of the encoded observation $\check{\mathbf{z}}_n = \mathbf{m}_{y \to z}(\check{\mathbf{y}}_n)$ at instant n, the predicted marginal posterior $P(\mathbf{s}_{n+1} \,|\, \check{\mathbf{z}}_{0:n})$ can be recursively computed from the predicted

marginal posterior $P(\mathbf{s}_n \mid \check{\mathbf{z}}_{0:n-1})$ in (2) – built from the encoded observations $\check{\mathbf{z}}_{0:n-1}$ up to instant $n-1$ – as

$$P(\mathbf{s}_n \,|\, \check{\mathbf{z}}_{0:n}) \propto P(\check{\mathbf{z}}_n \,|\, \mathbf{s}_n)\, P(\mathbf{s}_n \,|\, \check{\mathbf{z}}_{0:n-1}) \tag{5}$$

$$P(\mathbf{s}_{n+1} \,|\, \check{\mathbf{z}}_{0:n}) = \sum_{\mathbf{s}_n \in \boldsymbol{\Omega}_s} P(\mathbf{s}_{n+1} \,|\, \mathbf{s}_n)\, P(\mathbf{s}_n \,|\, \check{\mathbf{z}}_{0:n}) \tag{6}$$

with proportionality constant in (5) given by the reciprocal of

$$P(\check{\mathbf{z}}_n \,|\, \check{\mathbf{z}}_{0:n-1}) = \sum_{\mathbf{s}_n \in \boldsymbol{\Omega}_s} P(\check{\mathbf{z}}_n \,|\, \mathbf{s}_n)\, P(\mathbf{s}_n \,|\, \check{\mathbf{z}}_{0:n-1}). \tag{7}$$

Eqs. (5) and (6) follow respectively by the straightforward application of the Bayes rule and the Chapman-Kolmogorov equation given assumptions (3) and (4).

Note also that, conditioned on the sequence of encoded observations $\check{\mathbf{z}}_{0:n}$ up to instant n, the random vector $\mathbf{Z}_{n+1}$ at instant $n+1$ is distributed according to the p.m.f.

$$P(\mathbf{z}_{n+1} \,|\, \check{\mathbf{z}}_{0:n}) = \sum_{\mathbf{s}_{n+1} \in \boldsymbol{\Omega}_s} P(\mathbf{z}_{n+1} \,|\, \mathbf{s}_{n+1})\, P(\mathbf{s}_{n+1} \,|\, \check{\mathbf{z}}_{0:n}). \tag{8}$$

We can compute therefore the maximum a posteriori (MAP) estimate of the next encoded observation $\check{\mathbf{z}}_{n+1}$ given all encoded observations $\check{\mathbf{z}}_{0:n}$ up to instant n as

$$\hat{\mathbf{z}}_{n+1|n} = \arg\max_{\mathbf{z}_{n+1} \in \boldsymbol{\Omega}_z} P(\mathbf{z}_{n+1} \,|\, \check{\mathbf{z}}_{0:n}) \tag{9}$$

and predict then the next output observation $\check{\mathbf{y}}_{n+1}$ at instant $n+1$ as

$$\hat{\mathbf{y}}_{n+1|n} \triangleq \mathbf{m}_{z \to y}(\hat{\mathbf{z}}_{n+1|n}). \tag{10}$$

In this paper, however, we consider that neither the observation model $P(\mathbf{z}_n \,|\, \mathbf{s}_n)$ nor the dynamic model $P(\mathbf{s}_n \,|\, \mathbf{s}_{n-1})$ is provided or known *a priori*. At first glance, this is an ill-defined problem to solve using the conventional, model-based Bayesian framework [21]. However, we introduce here a novel approach to incrementally build – as new, incoming observations are assimilated – non-parametric representations of $P(\mathbf{z}_n \,|\, \mathbf{s}_n)$ and $P(\mathbf{s}_n \,|\, \mathbf{s}_{n-1})$ which, as indicated in Eqs. (5) through (8), entail the required knowledge regarding the observed stochastic process to recursively compute the predicted posteriors $P(\mathbf{s}_{n+1} \,|\, \check{\mathbf{z}}_{0:n})$ and $P(\mathbf{z}_{n+1} \,|\, \check{\mathbf{z}}_{0:n})$ at instant n from the previous posterior $P(\mathbf{s}_n \,|\, \check{\mathbf{z}}_{0:n-1})$ built at instant $n-1$.

2.2 Problem breakdown

Our ultimate goal can be finally unfolded into two tasks: i) *continuously* learn suitable, non-parametric probabilistic models for $P(\mathbf{s}_n \,|\, \mathbf{s}_{n-1})$ and $P(\mathbf{z}_n \,|\, \mathbf{s}_n)$ to be able to recursively approximate, at each time instant n, the marginal posterior $P(\mathbf{z}_{n+1} \,|\, \check{\mathbf{z}}_{0:n})$ from the available data $\check{\mathbf{z}}_{0:n}$ as indicated in Eqs. (5) through (8), and ii) use the predicted marginal posterior $P(\mathbf{z}_{n+1} \,|\, \check{\mathbf{z}}_{0:n})$ given the observations $\check{\mathbf{z}}_{0:n}$ up to instant n to predict the next output observation $\check{\mathbf{y}}_{n+1}$ at instant $n+1$ as shown in Eqs. (9) and (10).

3. SOLUTION APPROACH

We define first the required overall structure of a suitable encoding-decoding (codec) scheme. In the sequel, we derive a novel grid-based filter to perform the aforementioned tasks. In particular, we employ a non-parametric, Bayesian approach to recursively predict the next observation and learn the model hyperparameters.

3.1 Codec structure

Let $\boldsymbol{\Omega}_z \triangleq \{\boldsymbol{\mu}_z^{(\ell)}\}$, $\ell \in \{1, \ldots, c_z\}$, be a set of sparsely encoded, binary vectors $\boldsymbol{\mu}_z^{(\ell)} \in \mathbb{Z}_2^{d_z}$ with cardinality $c_z \triangleq |\boldsymbol{\Omega}_z|$ such that

$$\forall \ell, \qquad ||\boldsymbol{\mu}_z^{(\ell)}||_1 \leq a_z$$
$$\forall \ell_1 \neq \ell_2, \qquad b_z \leq d_H(\boldsymbol{\mu}_z^{(\ell_1)}, \boldsymbol{\mu}_z^{(\ell_2)}) \leq 2a_z,$$

where $a_z \ll d_z$ is the maximum number of activated bits and b_z is the minimum Hamming distance between any two vectors in $\boldsymbol{\Omega}_z$. Thus, a_z controls the sparseness in $\boldsymbol{\Omega}_z$, whereas b_z determines the discriminability of its elements in terms of the Hamming distance. Moreover, let $\{\boldsymbol{\Omega}_y^{(\ell)}\}$, $\ell \in \{1, \ldots, c_z\}$, denote a set partition of the output observation space $\boldsymbol{\Omega}_y$ such that

$$\bigcup_{\ell=1}^{c_z} \boldsymbol{\Omega}_y^{(\ell)} = \boldsymbol{\Omega}_y$$

and its subsets $\boldsymbol{\Omega}_y^{(\ell)}$ are pairwise disjoint, i.e., $\forall \ell_1 \neq \ell_2$, $\boldsymbol{\Omega}_y^{(\ell_1)} \cap \boldsymbol{\Omega}_y^{(\ell_2)} = \emptyset$. We define then the following lossy, encoding function $\mathbf{m}_{y \to z} : \mathbb{R}^{d_y} \mapsto \{\boldsymbol{\mu}_z^{(\ell)}\} \subset \mathbb{Z}_2^{d_z}$

$$\begin{aligned} \mathbf{z} &= \mathbf{m}_{y \to z}(\mathbf{y}) \\ &\triangleq \sum_{\ell=1}^{c_z} \boldsymbol{\mu}_z^{(\ell)} [\![\mathbf{y} \in \boldsymbol{\Omega}_y^{(\ell)}]\!]. \end{aligned} \tag{11}$$

Note that $\{\boldsymbol{\mu}_z^{(\ell)}\}$, $\ell \in \{1, \ldots, c_z\}$ is an encoded alphabet. In particular, $\forall \ell \in \{1, \ldots, c_z\}$, $\boldsymbol{\mu}_z^{(\ell)}$ assigns a sparse, binary symbol to the set partition $\boldsymbol{\Omega}_y^{(\ell)}$ with a_z activated bits and at least b_z bits far from the remaining $c_z - 1$ symbols in $\boldsymbol{\Omega}_z \subset \mathbb{Z}_2^{d_z}$.

Note also that $\mathbf{m}_{y \to z}(\cdot)$ is by construction a deterministic, surjective function. We define therefore the following decoding function $\mathbf{m}_{z \to y} : \{\boldsymbol{\mu}_z^{(\ell)}\} \mapsto \mathbb{R}^{d_y}$

$$\begin{aligned} \mathbf{y} &= \mathbf{m}_{z \to y}(\mathbf{z}) \\ &\triangleq \sum_{\ell=1}^{c_z} \boldsymbol{\mu}_y^{(\ell)} [\![\mathbf{z} = \boldsymbol{\mu}_z^{(\ell)}]\!], \end{aligned} \tag{12}$$

in which $\boldsymbol{\mu}_y^{(\ell)}$ can be e.g. the geometric centroid of the set partition $\boldsymbol{\Omega}_y^{(\ell)}$. In this case, $\forall \ell \in \{1, \ldots, c_z\}$, we can assume that $\boldsymbol{\Omega}_y^{(\ell)}$ is convex by construction such that its centroid $\boldsymbol{\mu}_y^{(\ell)} \in \boldsymbol{\Omega}_y^{(\ell)}$. Lastly, we follow the lead in [29] and further assume that close observations in $\boldsymbol{\Omega}_y$ yield close symbols in $\boldsymbol{\Omega}_z$. In particular, if the Euclidean distance $||\mathbf{y} - \mathbf{y}'||_2 \leq ||\mathbf{y}'' - \mathbf{y}'''||_2$ for $\mathbf{y}, \mathbf{y}', \mathbf{y}'', \mathbf{y}''' \in \boldsymbol{\Omega}_y$, than the Hamming distance $d_H(\mathbf{m}_{y \to z}(\mathbf{y}), \mathbf{m}_{y \to z}(\mathbf{y}')) \leq d_H(\mathbf{m}_{y \to z}(\mathbf{y}''), \mathbf{m}_{y \to z}(\mathbf{y}'''))$.

3.2 Grid-based filter

Let the c_s-dimensional vector

$$\boldsymbol{\pi}_{n|n-1} \triangleq \begin{bmatrix} \pi_{n|n-1}^{(1)} & \cdots & \pi_{n|n-1}^{(\ell')} & \cdots & \pi_{n|n-1}^{(c_s)} \end{bmatrix}^\top \tag{13}$$

collecting the conditional probabilities, $\forall \ell' \in \{1, \ldots, c_s\}$,

$$\pi_{n|n-1}^{(\ell')} \triangleq Pr\{\mathbf{S}_n = \boldsymbol{\mu}_s^{(\ell')} \,|\, \mathbf{Z}_0 = \check{\mathbf{z}}_0, \ldots, \mathbf{Z}_{n-1} = \check{\mathbf{z}}_{n-1}\}$$

represent the predicted marginal posterior $P(\mathbf{s}_n \mid \check{\mathbf{z}}_{0:n-1})$. Similarly, let the c_z-dimensional vector

$$\tilde{\boldsymbol{\pi}}_{n+1|n} \triangleq \begin{bmatrix} \tilde{\pi}_{n+1|n}^{(1)} & \cdots & \tilde{\pi}_{n+1|n}^{(\ell)} & \cdots & \tilde{\pi}_{n+1|n}^{(c_z)} \end{bmatrix}^\top \tag{14}$$

collecting the conditional probabilities, $\forall \ell \in \{1, \ldots, c_z\}$,

$$\tilde{\pi}^{(\ell)}_{n+1|n} \triangleq Pr\{\mathbf{Z}_{n+1} = \boldsymbol{\mu}^{(\ell)}_z \,|\, \mathbf{Z}_0 = \check{\mathbf{z}}_0, \ldots, \mathbf{Z}_{n-1} = \check{\mathbf{z}}_n\}$$

represent the predicted marginal posterior $P(\mathbf{z}_{n+1} \,|\, \check{\mathbf{z}}_{0:n})$. We can combine the update and prediction steps in Eqs. (5) and (6), respectively, into a single, one-step-ahead update rule as

$$P(\mathbf{s}_{n+1} \,|\, \check{\mathbf{z}}_{0:n}) \propto \sum_{\mathbf{s}_n \in \boldsymbol{\Omega}_s} P(\mathbf{s}_{n+1} \,|\, \mathbf{s}_n)\, P(\check{\mathbf{z}}_n \,|\, \mathbf{s}_n)\, P(\mathbf{s}_n \,|\, \check{\mathbf{z}}_{0:n-1}). \tag{15}$$

Thus, by assuming that $\check{\mathbf{z}}_n = \boldsymbol{\mu}^{(k)}_z$ for some $k \in \{1, \ldots, c_z\}$, we can rewrite (15) in a compact notation using matrix-vector multiplication as

$$\boldsymbol{\pi}_{n+1|n} \propto \mathbf{F}^{(k)}_{n+1|n} \cdot \boldsymbol{\pi}_{n|n-1} \tag{16}$$

with proportionality constant such that $||\boldsymbol{\pi}_{n+1|n}||_1 = 1$. Moreover, the i-th element $f^{(k,j,i)}_{n+1|n}$ of the j-th row

$$\mathbf{f}^{(k,j)}_{n+1|n} \triangleq \begin{bmatrix} f^{(k,j,1)}_{n+1|n} & \ldots & f^{(k,j,i)}_{n+1|n} & \ldots & f^{(k,j,c_s)}_{n+1|n} \end{bmatrix}^\top$$

of the modified, $c_s \times c_s$ state transition matrix

$$\mathbf{F}^{(k)}_{n+1|n} \triangleq \begin{bmatrix} \mathbf{f}^{(k,1)}_{n+1|n} & \ldots & \mathbf{f}^{(k,j)}_{n+1|n} & \ldots & \mathbf{f}^{(k,c_s)}_{n+1|n} \end{bmatrix}^\top$$

– which depends on the observation $\check{\mathbf{z}}_n$ – is defined as

$$\begin{aligned} f^{(k,j,i)}_{n+1|n} &\triangleq Pr\{\mathbf{S}_{n+1} = \boldsymbol{\mu}^{(j)}_s \,|\, \mathbf{S}_n = \boldsymbol{\mu}^{(i)}_s\} \\ &\times \; Pr\{\mathbf{Z}_n = \boldsymbol{\mu}^{(k)}_z \,|\, \mathbf{S}_n = \boldsymbol{\mu}^{(i)}_s\} \end{aligned}$$

such that $\pi^{(j)}_{n+1|n} \propto \langle \mathbf{f}^{(k,j)}_{n+1|n}, \boldsymbol{\pi}_{n|n-1} \rangle$, $\forall j \in \{1, \ldots, c_s\}$. Conversely, the predicted marginal posterior $P(\mathbf{z}_{n+1} \,|\, \check{\mathbf{z}}_{0:n})$ in (8), can be rewritten in compact notation as

$$\tilde{\boldsymbol{\pi}}_{n+1|n} \propto \mathbf{H}_{n+1} \cdot \boldsymbol{\pi}_{n+1|n} \tag{17}$$

with proportionality constant such that $||\tilde{\boldsymbol{\pi}}_{n+1|n}||_1 = 1$. Furthermore, the j-th element $h^{(\ell,j)}_{n+1}$ of the ℓ-th row

$$\mathbf{h}^{(\ell)}_{n+1} \triangleq \begin{bmatrix} h^{(\ell,1)}_{n+1} & \ldots & h^{(\ell,j)}_{n+1} & \ldots & h^{(\ell,c_s)}_{n+1} \end{bmatrix}^\top$$

of the $c_z \times c_s$ observation matrix

$$\mathbf{H}_{n+1} \triangleq \begin{bmatrix} \mathbf{h}^{(1)}_{n+1} & \ldots & \mathbf{h}^{(\ell)}_{n+1} & \ldots & \mathbf{h}^{(c_z)}_{n+1} \end{bmatrix}^\top$$

is defined as

$$h^{(\ell,j)}_{n+1} \triangleq Pr\{\mathbf{Z}_{n+1} = \boldsymbol{\mu}^{(\ell)}_z \,|\, \mathbf{S}_{n+1} = \boldsymbol{\mu}^{(j)}_s\}$$

such that $\tilde{\pi}^{(\ell)}_{n+1|n} \propto \langle \mathbf{h}^{(\ell)}_{n+1}, \boldsymbol{\pi}_{n+1|n} \rangle$, $\forall \ell \in \{1, \ldots, c_z\}$. Note though that we can not use Eqs. (16) and (17) to compute the posteriors $P(\mathbf{s}_{n+1} \,|\, \check{\mathbf{z}}_{0:n})$ and $P(\mathbf{z}_{n+1} \,|\, \check{\mathbf{z}}_{0:n})$ since the matrices $\mathbf{F}^{(k)}_{n+1|n}$ and $\mathbf{H}_{n+1}$ are unknown. The Baum-Welch [4] algorithm is a widespread expectation-maximization (EM) procedure to estimate the state transition probabilities $P(\mathbf{s}_{n+1} \,|\, \mathbf{s}_n)$ and the symbol emitting probabilities $P(\check{\mathbf{z}}_n \,|\, \mathbf{s}_n)$ required by both $\mathbf{F}^{(k)}_{n+1|n}$ and $\mathbf{H}_{n+1}$. However, we need to continuously learn these models online, recursively as new observations arrive. Thus, we need to build models for them using a different approach.

3.3 Belief representation

The predicted marginal posterior $P(\mathbf{s}_n \,|\, \check{\mathbf{z}}_{0:n-1})$ can be represented by an alternative vector – referred to as sparsely encoded belief hereafter in the paper – in the high-dimensional binary space $\boldsymbol{\Psi}_s \triangleq \mathbb{Z}_2^{c_s+p_s}$

$$\mathbf{b}_{n|n-1} \triangleq \begin{bmatrix} b^{(1)}_{n|n-1} & \ldots & b^{(c_s+p_s)}_{n|n-1} \end{bmatrix}^\top \in \boldsymbol{\Psi}_s \tag{18}$$

with up to p_s active bits such that $1 \leq ||\mathbf{b}_{n|n-1}||_1 \leq p_s$. As indicated in Appendix A, we can employ a deterministic procedure to build

$$\mathbf{b}_{n|n-1} = Bel(\boldsymbol{\pi}_{n|n-1}; p_s) \tag{19}$$

at any $n \geq 0$. Moreover, we can approximately retrieve the original posterior $\boldsymbol{\pi}_{n|n-1}$ by decoding the sparse belief (see Appendix A for details)

$$\boldsymbol{\pi}_{n|n-1} \approx Bel^{-1}(\mathbf{b}_{n|n-1}; p_s). \tag{20}$$

Hence, the belief $\mathbf{b}_{n|n-1}$ can be seen as a sparse symbol in the high-dimensional binary space $\boldsymbol{\Psi}_s$ corresponding to a valid p.m.f. $\boldsymbol{\pi}_{n|n-1}$ collecting probabilities of symbols in $\boldsymbol{\Omega}_s$. As discussed in Appendix A, the proposed encoding procedure $Bel(\cdot)$ (see Algorithm 4) is lossy in the sense that multiple categorical distributions can map to the same sparse belief. Consequently, as indicated in Eq. (20), the decoding procedure $Bel^{-1}(\cdot)$ (see Algorithm 5) will not be able to fully recover the original belief. Furthermore, this may harm the performance of the proposed grid-based filter, since, as highlighted in the sequel (refer to Algorithm 1), at each time step n, the grid-based filter employs the encoding procedure to build the predicted belief $\mathbf{b}_{n+1|n} = Bel(\boldsymbol{\pi}_{n+1|n}; p_s)$ with up to p_s activated bits, a binary sparse representation of the predicted distribution $\boldsymbol{\pi}_{n+1|n}$.

3.4 Re-interpreting filtering steps

From Eqs. (16) and (20), we can see that, at instant n, the predicted marginal posterior $\boldsymbol{\pi}_{n+1|n}$ is approximately a function $\mathbf{f}_n : \boldsymbol{\Omega}_z \times \boldsymbol{\Psi}_s \mapsto \mathbb{S}_{c_s}$ of the current observation $\check{\mathbf{z}}_n = \boldsymbol{\mu}^{(k)}_z$ and the belief $\mathbf{b}_{n|n-1}$ at instant $n-1$, i.e.

$$\begin{aligned} \boldsymbol{\pi}_{n+1|n} &= C^{-1}_{n+1|n} \cdot \mathbf{F}^{(k)}_{n+1|n} \cdot \boldsymbol{\pi}_{n|n-1} \\ &\approx C^{-1}_{n+1|n} \cdot \mathbf{F}^{(k)}_{n+1|n} \cdot Bel^{-1}(\mathbf{b}_{n|n-1}; p_s) \\ &\equiv \mathbf{f}_n(\check{\mathbf{z}}_n, \mathbf{b}_{n|n-1}), \end{aligned} \tag{21}$$

where the normalization constant $C_{n+1|n}$ is chosen such that $||\boldsymbol{\pi}_{n+1|n}||_1 = 1$. Similarly, by examining Eq. (17) and noting that $\boldsymbol{\pi}_{n+1|n} \approx Bel^{-1}(\mathbf{b}_{n+1|n}; p_s)$, we conclude that, at instant n, the predicted posterior $\tilde{\boldsymbol{\pi}}_{n+1|n}$ is also an approximate function $\mathbf{h}_{n+1} : \boldsymbol{\Psi}_s \mapsto \mathbb{S}_{c_z}$ of the predicted belief $\mathbf{b}_{n+1|n}$ at instant n, i.e.

$$\begin{aligned} \tilde{\boldsymbol{\pi}}_{n+1|n} &= \tilde{C}^{-1}_{n+1|n} \cdot \mathbf{H}_{n+1} \cdot \boldsymbol{\pi}_{n+1|n} \\ &\approx \tilde{C}^{-1}_{n+1|n} \cdot \mathbf{H}_{n+1} \cdot Bel^{-1}(\mathbf{b}_{n+1|n}; p_s) \\ &\equiv \mathbf{h}_{n+1}(\mathbf{b}_{n+1|n}) \end{aligned} \tag{22}$$

with $\mathbf{b}_{n+1|n} = Bel(\boldsymbol{\pi}_{n+1|n}; p_s)$ and $\tilde{C}_{n+1|n}$ selected such that $||\tilde{\boldsymbol{\pi}}_{n+1|n}||_1 = 1$. Thus, at each instant n, it suffices to propagate the sparse beliefs from $\mathbf{b}_{n|n-1}$ to $\mathbf{b}_{n+1|n}$ and plug them into Eqs. (21) and (22) to compute the predicted marginal posteriors $\boldsymbol{\pi}_{n+1|n}$ and $\tilde{\boldsymbol{\pi}}_{n+1|n}$, respectively.

3.5 Non-parametric estimation

We follow the lead of [17] and assume that, conditioned on a particular sequence of hidden states $\check{\mathbf{s}}_{0:n}$ up to instant n, the predicted posterior $\boldsymbol{\pi}_{n+1|n}$ is a realization of a random vector $\boldsymbol{\Pi}_{n+1|n}$ in the (c_s-1)-dimensional simplex $\mathbb{S}_{c_s}$ following a Dirichlet distribution $Dir(c_s;\boldsymbol{\alpha}_{n|n-1})$ with hyperparameters $\boldsymbol{\alpha}_{n|n-1} \triangleq \begin{bmatrix}\alpha^{(1)}_{n|n-1} & \dots & \alpha^{(c_s)}_{n|n-1}\end{bmatrix}^\top \in \mathbb{R}^{c_s}_+$, i.e.

$$\boldsymbol{\Pi}_{n+1|n} \,|\, \check{\mathbf{s}}_{0:n} \sim Dir(c_s;\boldsymbol{\alpha}_{n|n-1}). \tag{23}$$

We also assume that $\boldsymbol{\alpha}_{n|n-1}$ is indexed by the observation $\check{\mathbf{z}}_n$ at instant n and the previous belief $\mathbf{b}_{n|n-1}$ as per Eq. (21). Thus, given the pair $\left(\check{\mathbf{z}}_n, \mathbf{b}_{n|n-1}\right)$, we can access the corresponding hyperparameters as $\boldsymbol{\alpha}_{n|n-1} \equiv \boldsymbol{\alpha}_{n|n-1}(\check{\mathbf{z}}_n, \mathbf{b}_{n|n-1})$ and use (23) to draw the predicted posterior $\boldsymbol{\pi}_{n+1|n}$. In the sequel, we employ Algorithm 4 (see Appendix A) to encode the predicted posterior $\boldsymbol{\pi}_{n+1|n}$ into the sparse belief

$$\mathbf{b}_{n+1|n} = Bel(\boldsymbol{\pi}_{n+1|n}; p_s). \tag{24}$$

Conditioned on the sequence of observations $\check{\mathbf{z}}_{0:n}$ up to instant n, we also assume that the predicted posterior $\tilde{\boldsymbol{\pi}}_{n+1|n}$ is a realization of a Dirichlet random vector $\tilde{\boldsymbol{\Pi}}_{n+1|n}$ in the (c_z-1)-dimensional simplex $\mathbb{S}_{c_z}$ with distribution $Dir(c_z;\tilde{\boldsymbol{\alpha}}_{n|n-1})$ and hyperparameters $\tilde{\boldsymbol{\alpha}}_{n|n-1} \triangleq \begin{bmatrix}\tilde{\alpha}^{(1)}_{n|n-1} & \dots & \tilde{\alpha}^{(c_z)}_{n|n-1}\end{bmatrix}^\top \in \mathbb{R}^{c_z}_+$, i.e.

$$\tilde{\boldsymbol{\Pi}}_{n+1|n} \,|\, \check{\mathbf{z}}_{0:n} \sim Dir(c_z;\tilde{\boldsymbol{\alpha}}_{n|n-1}), \tag{25}$$

in which $\tilde{\boldsymbol{\alpha}}_{n|n-1}$ is yet again indexed by the predicted belief $\mathbf{b}_{n+1|n}$ computed as in (24) following Eq. (22). Thus, we access the hyperparameters as $\tilde{\boldsymbol{\alpha}}_{n|n-1} \equiv \tilde{\boldsymbol{\alpha}}_{n|n-1}(\mathbf{b}_{n+1|n})$ and use (25) to draw the predicted posterior $\tilde{\boldsymbol{\pi}}_{n+1|n}$. Finally, at instant n, we can predict the next encoded observation $\check{\mathbf{z}}_{n+1}$ at instant n from the predicted belief $\tilde{\boldsymbol{\pi}}_{n+1|n}$ as

$$\begin{aligned}\hat{\mathbf{z}}_{n+1|n} &= \boldsymbol{\mu}_z^{(\hat{k})} \in \boldsymbol{\Omega}_z \\ \hat{k} &= \underset{1\le\ell\le c_z}{\arg\max}\, \tilde{\pi}^{(\ell)}_{n+1|n}.\end{aligned} \tag{26}$$

The probability $\tilde{\pi}^{(\hat{k})}_{n+1|n}$ can be seen thus as a confidence score of the filter prediction $\hat{\mathbf{z}}_{n+1|n}$, whereas the probability $1-\tilde{\pi}^{(k)}_{n+1|n}$ with $k \in \{1,\dots,c_z\}$ corresponding to true observed symbol $\check{\mathbf{z}}_{n+1} = \boldsymbol{\mu}_z^{(k)}$ at instant $n+1$ can be seen as a novelty (or anomaly) score of the observation $\check{\mathbf{z}}_{n+1}$.

Algorithm 1 summarizes a single iteration of the proposed non-parametric, grid-based filtering (nP-GbF) procedure at instant n considering some memory-based mechanism $\mathcal{M}$ to store-retrieve the hyperparameters.

Algorithm 1 nP-GbF$\left(\check{\mathbf{z}}_n, \mathbf{b}_{n|n-1}, \mathcal{M}\right)$

1: Retrieve $\boldsymbol{\alpha}_{n|n-1} \equiv \boldsymbol{\alpha}_{n|n-1}\left(\check{\mathbf{z}}_n, \mathbf{b}_{n|n-1}\right)$ from $\mathcal{M}$.
2: Draw $\boldsymbol{\pi}_{n+1|n} \sim Dir(c_s;\boldsymbol{\alpha}_{n|n-1})$ as in (23).
3: Build the sparsely encoded belief $\mathbf{b}_{n+1|n}$ as in (24).
4: Retrieve $\tilde{\boldsymbol{\alpha}}_{n|n-1} \equiv \tilde{\boldsymbol{\alpha}}_{n|n-1}\left(\mathbf{b}_{n+1|n}\right)$ from $\mathcal{M}$.
5: Draw $\tilde{\boldsymbol{\pi}}_{n+1|n} \sim Dir(c_z;\tilde{\boldsymbol{\alpha}}_{n|n-1})$ as in (25).
6: Predict next observation $\hat{\mathbf{z}}_{n+1|n} = \boldsymbol{\mu}_z^{(\hat{k})}$ as in (26).
7: **return** $\left(\hat{\mathbf{z}}_{n+1|n}, \hat{k}, \mathbf{b}_{n+1|n}, \tilde{\boldsymbol{\pi}}_{n+1|n}\right)$

Figures 2 and 3 show a running example of the nP-GbF procedure. In Step 1, we retrieve the pseudo-counts $\boldsymbol{\alpha}_{n|n-1}$ from $\mathcal{M}$ given the observation $\check{\mathbf{z}}_n$ at instant n and the previous belief $\mathbf{b}_{n|n-1}$. In the sequel, we sample the predicted posterior $\boldsymbol{\pi}_{n+1|n}$ from the Dirichlet distribution $Dir(c_s;\boldsymbol{\alpha}_{n|n-1})$ in Step 2 and build the predicted belief $\mathbf{b}_{n+1|n}$ in Step 3. Note that the recovered posterior $Bel^{-1}(\mathbf{b}_{n+1|n};p_s)$ shown in Fig. 2 does not match the sampled posterior $\boldsymbol{\pi}_{n+1|n}$ in Step 2. Specifically, the probabilities $\pi^{(\ell'_2)}_{n+1|n}$ and $\pi^{(\ell'_3)}_{n+1|n}$ in the original posterior $\boldsymbol{\pi}_{n+1|n}$ collapsed into a single probability at location ℓ'_2, since the indexes $\ell'_2 = 12$ and $\ell'_3 = 14$ were too close, i.e. $\ell'_3 - \ell'_2 = 2 < p_s = 8$. On the other hand, note that the probability $\pi^{(\ell'_1)}_{n+1|n}$ at the location $\ell'_1 = 4$ remained virtually intact in the recovered posterior $Bel^{-1}(\mathbf{b}_{n+1|n};p_s)$.

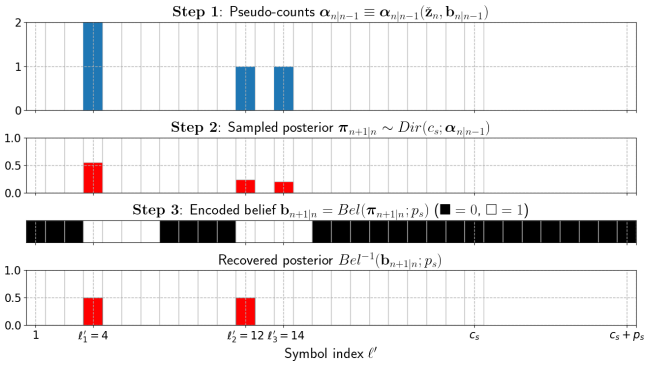

Figure 2: Running example illustrating Steps 1 through 3 of the nP-GbF procedure considering $c_s = 24$, $p_s = 8$ and $\alpha^{(\ell')}_{n|n-1} = 2^{-12}$, $\forall \ell'$, except for $\ell'_1 = 4$, $\ell'_2 = 12$ and $\ell'_3 = 14$.

Similarly, as show in Fig. 3, we retrieve the pseudo-counts $\tilde{\boldsymbol{\alpha}}_{n|n-1}$ from $\mathcal{M}$ in Step 4 using now the predicted belief $\mathbf{b}_{n+1|n}$ at instant n obtained in Step 3. Then, we sample the predicted posterior $\tilde{\boldsymbol{\pi}}_{n+1|n}$ from the Dirichlet distribution $Dir(c_z;\tilde{\boldsymbol{\alpha}}_{n|n-1})$ in Step 5 and, finally, predict the next observation $\hat{\mathbf{z}}_{n+1|n}$ in Step 6 as the symbol $\boldsymbol{\mu}_z^{(\hat{k})} \in \boldsymbol{\Omega}_z$ with the highest probability $\tilde{\pi}^{(\hat{k})}_{n+1|n}$ in the predicted posterior $\tilde{\boldsymbol{\pi}}_{n+1|n}$. In the running example, the highest probability $\tilde{\pi}^{(\ell_2)}_{n+1|n}$ occurs at the location ℓ_2, i.e. $\hat{k} = \ell_2 = 16$.

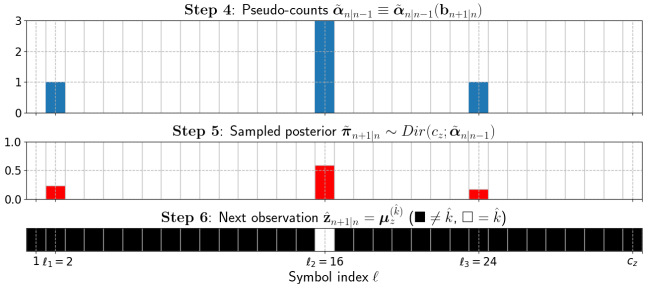

Figure 3: Running example illustrating Steps 3 through 6 of the nP-GbF procedure considering $c_z = 32$ and $\tilde{\alpha}^{(\ell)}_{n|n-1} = 2^{-12}$, $\forall \ell$, except for $\ell_1 = 2$, $\ell_2 = 16$ and $\ell_3 = 24$.

3.6 Hidden states formation

In general, the hidden states are not directly observable. As expected, the filtering steps (see Algorithm 1) do not require one to directly observe them. However, as shown in Appendix B, to be able to update the model hyperparameters

$\boldsymbol{\alpha}_{n|n-1}$ using a frequentist approach, we need to pin the index j of the hidden state symbol $\boldsymbol{\mu}_s^{(j)}$ at instant $n+1$ from which the observation $\check{\mathbf{z}}_{n+1}$ is emitted. To this end, we further assume that the hidden state $\mathbf{s}_n$ represents the last ξ observations. Specifically, at instant $n+1$, upon the arrival of the new encoded observation $\check{\mathbf{z}}_{n+1} = \boldsymbol{\mu}_z^{(k)}$, $k \in \{1, \ldots, c_z\}$, we select a unique index $j \in \{1, \ldots, c_s\}$ and assign it to the sub-sequence of observations $\check{\mathbf{z}}_{n+2-\xi:n+1}$, i.e. we make $\check{\mathbf{s}}_{n+1} = \boldsymbol{\mu}_s^{(j)} \in \boldsymbol{\Omega}_s$. Lastly, in this paper, we increment the selected indexes from $\{1, \ldots, c_s\}$ by p_s to make sure their belief representations will not overlap (see Appendix A). Then, we update the hyperparameters $\boldsymbol{\alpha}_{n|n-1}$ and $\tilde{\boldsymbol{\alpha}}_{n|n-1}$ as follows.

3.7 Non-parametric learning

By construction, conditioned on the predicted marginal posterior $\boldsymbol{\pi}_{n+1|n}$ at instant n, the hidden state $\mathbf{S}_{n+1}$ at instant $n+1$ is distributed according to a categorical distribution $Cat(\boldsymbol{\Omega}_s; \boldsymbol{\pi}_{n+1|n})$ over $\boldsymbol{\Omega}_s$, i.e.

$$\mathbf{S}_{n+1} \,|\, \boldsymbol{\pi}_{n+1|n} \sim Cat(\boldsymbol{\Omega}_s; \boldsymbol{\pi}_{n+1|n}). \tag{27}$$

Since the Dirichlet distribution is the conjugate prior of the categorical distribution, it follows from Eqs. (23) and (27) that, conditioned on the built sequence of symbols $\check{\mathbf{s}}_{0:n+1}$ up to instant $n+1$, the random vector $\boldsymbol{\Pi}_{n+1|n}$ is also Dirichlet

$$\boldsymbol{\Pi}_{n+1|n} \,|\, \check{\mathbf{s}}_{n+1} = \boldsymbol{\mu}_s^{(j)}, \check{\mathbf{s}}_{0:n} \sim Dir(c_s; \boldsymbol{\alpha}_{n+1|n}), \tag{28}$$

in which the new hyperparameters are updated using a frequentist approach as (proof details are in Appendix B)

$$\boldsymbol{\alpha}_{n+1|n} = \boldsymbol{\alpha}_{n|n-1} + \mathbf{1}_j. \tag{29}$$

Note that, conditioned on $\tilde{\boldsymbol{\pi}}_{n+1|n}$, the next observation $\mathbf{Z}_{n+1}$ at instant $n+1$ is also distributed according to a categorical distribution $Cat(\boldsymbol{\Omega}_z; \tilde{\boldsymbol{\pi}}_{n+1|n})$ over $\boldsymbol{\Omega}_z$, i.e.

$$\mathbf{Z}_{n+1} \,|\, \tilde{\boldsymbol{\pi}}_{n+1|n} \sim Cat(\boldsymbol{\Omega}_z; \tilde{\boldsymbol{\pi}}_{n+1|n}). \tag{30}$$

Thus, it follows from Eqs. (25) and (30) that the same principle is applicable to $\tilde{\boldsymbol{\Pi}}_{n+1|n}$. Conditioned on the encoded sequence of observations $\check{\mathbf{z}}_{0:n+1}$ up to instant $n+1$, we can write

$$\tilde{\boldsymbol{\Pi}}_{n+1|n} \,|\, \check{\mathbf{z}}_{n+1} = \boldsymbol{\mu}_z^{(k)}, \check{\mathbf{z}}_{0:n} \sim Dir(c_z; \tilde{\boldsymbol{\alpha}}_{n+1|n}), \tag{31}$$

where the new hyperparameters are given by (see Appendix B)

$$\tilde{\boldsymbol{\alpha}}_{n+1|n} = \tilde{\boldsymbol{\alpha}}_{n|n-1} + \mathbf{1}_k. \tag{32}$$

Even though learning happens at instant $n+1$, it is worth noting that the updated hyperparameters $\boldsymbol{\alpha}_{n+1|n}$ must be properly stored indexed by both the observation $\check{\mathbf{z}}_n$ and the previous belief $\mathbf{b}_{n|n-1}$ from instant n such that $\boldsymbol{\alpha}_{n+1|n} \equiv \boldsymbol{\alpha}_{n+1|n}(\check{\mathbf{z}}_n, \mathbf{b}_{n|n-1})$. Conversely, the updated hyperparameters $\tilde{\boldsymbol{\alpha}}_{n+1|n}$ must be stored indexed by the predicted belief $\mathbf{b}_{n+1|n}$, i.e. $\tilde{\boldsymbol{\alpha}}_{n+1|n} \equiv \tilde{\boldsymbol{\alpha}}_{n+1|n}(\mathbf{b}_{n+1|n})$. Thus, in this context, we need to draw a new posterior $\boldsymbol{\pi}_{n+1|n} \sim Dir(c_s; \boldsymbol{\alpha}_{n+1|n})$ using (23) and rebuild the sparse belief $\mathbf{b}_{n+1|n}$ as in (24), before storing the updated hyperparameters $\tilde{\boldsymbol{\alpha}}_{n+1|n}$ – in some preexisting storage $\mathcal{M}$ – indexed by $\mathbf{b}_{n+1|n}$.

The non-parametric, memory-based learning (nP-MbL) procedure is outlined in Algorithm 2. We assume again that all hyperparameters are retrieved from and stored back to the same memory-based mechanism $\mathcal{M}$ and that the observation and the hidden state at instant $n+1$ are respectively $\check{\mathbf{z}}_{n+1} = \boldsymbol{\mu}_z^{(k)}$ and $\check{\mathbf{s}}_{n+1} = \boldsymbol{\mu}_s^{(j)}$. Note that we weighted the increments in (29) and (32) by a learning rate $\eta \geq 1$ to speed-up sequence learning.

Algorithm 2 nP-MbL$\big(j, k, \eta, \check{\mathbf{z}}_n, \mathbf{b}_{n|n-1}, \mathbf{b}_{n+1|n}, \mathcal{M}\big)$

1: Retrieve $\boldsymbol{\alpha}_{n|n-1} \equiv \boldsymbol{\alpha}_{n|n-1}\big(\check{\mathbf{z}}_n, \mathbf{b}_{n|n-1}\big)$ from $\mathcal{M}$.
2: Update $\boldsymbol{\alpha}_{n+1|n} \leftarrow \boldsymbol{\alpha}_{n|n-1} + \eta \cdot \mathbf{1}_j$ as in (29).
3: Store $\boldsymbol{\alpha}_{n+1|n}\big(\check{\mathbf{z}}_n, \mathbf{b}_{n|n-1}\big) \equiv \boldsymbol{\alpha}_{n+1|n}$ into $\mathcal{M}$.
4: Draw $\boldsymbol{\pi}_{n+1|n} \sim Dir(c_s; \boldsymbol{\alpha}_{n+1|n})$ as in (23).
5: Encode $\boldsymbol{\pi}_{n+1|n}$ into $\mathbf{b}_{n+1|n}$ as in (24).
6: Retrieve $\tilde{\boldsymbol{\alpha}}_{n|n-1} \equiv \tilde{\boldsymbol{\alpha}}_{n|n-1}\big(\mathbf{b}_{n+1|n}\big)$ from $\mathcal{M}$.
7: Update $\tilde{\boldsymbol{\alpha}}_{n+1|n} \leftarrow \tilde{\boldsymbol{\alpha}}_{n|n-1} + \eta \cdot \mathbf{1}_k$ as in (32).
8: Store $\tilde{\boldsymbol{\alpha}}_{n+1|n}\big(\mathbf{b}_{n+1|n}\big) \equiv \tilde{\boldsymbol{\alpha}}_{n+1|n}$ into $\mathcal{M}$.
9: **return** $\mathcal{M}$

4. RAM-BASED IMPLEMENTATION

In this section we propose an efficient implementation of the memory-based mechanism $\mathcal{M}$ using a shallow, two-layer depth WNN. Specifically, each layer consists of an one-dimensional array of VG-RAM nodes allowing us thus to i) quickly retrieve input-output pairs indexed by binary input patterns and ii) perform one-shot learning by storing new associative input-output pairs readily.

A VG-RAM node stores a set $\mathbf{M}$ of input-output pairs – a.k.a. lookup table – such that, $\forall\,(\boldsymbol{\beta}', \alpha') \in \mathbf{M}$, $\boldsymbol{\beta}' \in \mathbb{Z}_2^{d_\beta}$ is a d_β-dimensional binary input pattern and $\alpha' \in \mathbb{R}_+$ is the associated output, in our case, a non-negative hyperparameter. In particular, given an input pattern $\boldsymbol{\beta} \in \mathbb{Z}_2^{d_\beta}$, the node finds the hyperparameter $\bar{\alpha}$ whose corresponding input pattern $\bar{\boldsymbol{\beta}}$ has the closest Hamming distance to it. More precisely, the node retrieves the hyperparameter $\bar{\alpha}$ in $\big(\bar{\boldsymbol{\beta}}, \bar{\alpha}\big) \leftarrow \text{FIND}\,(\mathbf{M}, \boldsymbol{\beta})$ from $\mathbf{M}$ for a given $\boldsymbol{\beta}$ such that

$$\big(\bar{\boldsymbol{\beta}}, \bar{\alpha}\big) = \underset{(\boldsymbol{\beta}', \alpha') \in \mathbf{M}}{\arg\min}\; d_H(\boldsymbol{\beta}', \boldsymbol{\beta}).$$

On the other hand, to learn a new input-output pair $(\boldsymbol{\beta}, \alpha)$, the node makes $\mathbf{M} \leftarrow \text{INIT}\,(\mathbf{M}, \boldsymbol{\beta}, \alpha) \triangleq \mathbf{M} \cup \{(\boldsymbol{\beta}, \alpha)\}$. Lastly, to update the hyperparameter $\bar{\alpha}$ associated to an existing input pattern $\bar{\boldsymbol{\beta}}$, replacing it by some $\tilde{\alpha} \in \mathbb{R}_+$, the node makes $\mathbf{M} \leftarrow \text{UPDT}\,\big(\mathbf{M}, \bar{\boldsymbol{\beta}}, \bar{\alpha}, \tilde{\alpha}\big) \triangleq \big(\mathbf{M} \setminus \{\big(\bar{\boldsymbol{\beta}}, \bar{\alpha}\big)\}\big) \cup \{\big(\bar{\boldsymbol{\beta}}, \tilde{\alpha}\big)\}$.

4.1 Non-parametric estimation

Figure 4 graphically illustrates Steps 1 through 3 of Algorithm 1 and summarizes therefore how the function $\mathbf{b}_{n+1|n} \equiv \mathbf{f}_n(\check{\mathbf{z}}_n, \mathbf{b}_{n|n-1})$ is implemented. In particular, the figure shows how we employ a c_s-dimensional array of VG-RAM nodes to retrieve the hyperparameters $\boldsymbol{\alpha}_{n|n-1}$ based on the stored input-output pairs whose input patterns have the closest Hamming distance to the concatenated input vector $\check{\mathbf{z}}_n \oplus \mathbf{b}_{n|n-1} \in \mathbb{Z}_2^{c_z + c_s + p_s}$. Figure 5 depicts Steps 4 through 6 of Algorithm 1, which in turn implements the function $\tilde{\mathbf{b}}_{n+1|n} \equiv \mathbf{h}_n(\mathbf{b}_{n+1|n})$. The figure indicates how an additional c_z-dimensional array of VG-RAM nodes is employed to retrieve $\tilde{\boldsymbol{\alpha}}_{n|n-1}$ based on the stored associative pairs whose input patterns have the closest Hamming distance to the input $\mathbf{b}_{n+1|n} \in \mathbb{Z}_2^{c_s + p_s}$.

Note that, in filtering Steps 2 and 5, the filter assigns different probabilities $\pi_{n+1|n}^{(\ell')}$ and $\tilde{\pi}_{n+1|n}^{(\ell)}$ to symbols in $\boldsymbol{\Omega}_s$ and $\boldsymbol{\Omega}_z$ according to the retrieved hyperparameters $\alpha_{n+1|n}^{(\ell')}$ and $\tilde{\alpha}_{n+1|n}^{(\ell)}$ in Steps 1 and 4, respectively. In Step 3, the filter also assigns an indexable label (symbol in $\boldsymbol{\Psi}_s$) to the predicted posterior $\boldsymbol{\pi}_{n+1|n}$ by building the sparse belief $\mathbf{b}_{n+1|n} = Bel(\boldsymbol{\pi}_{n+1|n})$,

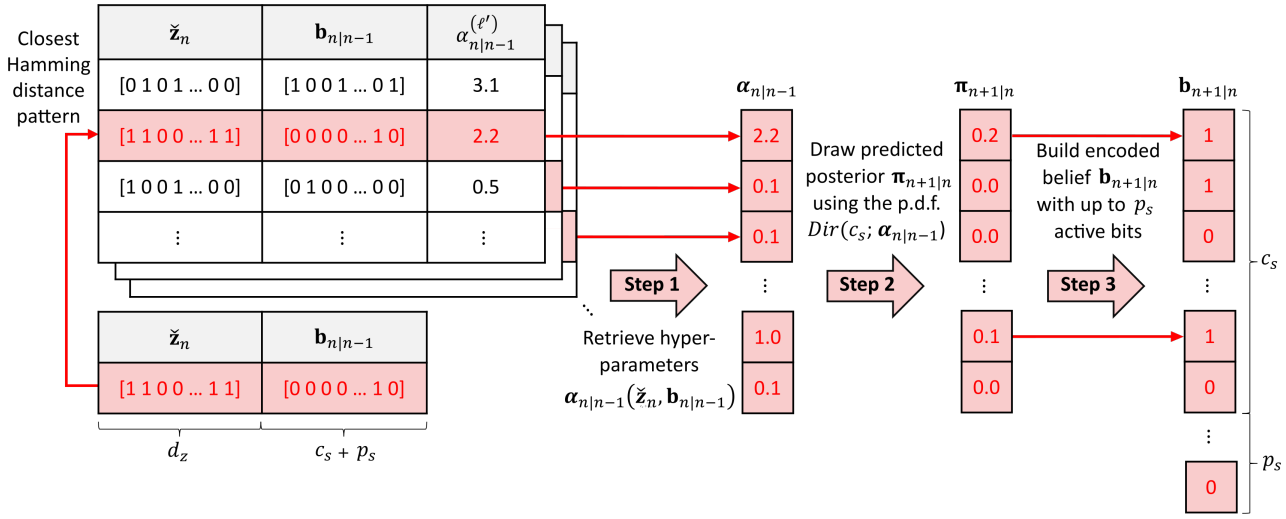

Figure 4: WNN layer $\mathcal{L}_1$: one-step-ahead state belief prediction.

which is in turn employed to retrieve the hyperparameters $\tilde{\alpha}^{(\ell)}_{n+1|n}$ at Step 4. We select finally the predicted observation $\hat{\mathbf{z}}_{n+1|n}$ in Step 6 – based on the predicted marginal posterior $\tilde{\boldsymbol{\pi}}_{n+1|n}$ – as the base vector $\boldsymbol{\mu}^{(\hat{k})}_z$ with the highest probability $\tilde{\pi}^{(\hat{k})}_{n+1|n}$ using (26).

4.2 Non-parametric learning

Let $\mathbf{M}^{(\ell')}_1$ and $\mathbf{M}^{(\ell)}_2$ denote respectively the lookup tables of the ℓ'-th and ℓ-th VG-RAM nodes at WNN layers $\mathcal{L}_1$ and $\mathcal{L}_2$. At instant n, the ℓ'-th VG-RAM node in Fig. 4 retrieves

$$\left(\cdot, \alpha^{(\ell')}_{n|n-1}\right) \leftarrow \text{FIND}\left(\mathbf{M}^{(\ell')}_1, \check{\mathbf{z}}_n \oplus \mathbf{b}_{n|n-1}\right)$$

and, therefore, contributes indirectly to the activation of the ℓ'-th bit of the belief $\mathbf{b}_{n+1|n}$ through the Dirichlet-sampled probability $\pi^{(\ell')}_{n+1|n}$. By the same token, the ℓ-th node in Fig. 5 retrieves

$$\left(\cdot, \tilde{\alpha}^{(\ell)}_{n|n-1}\right) \leftarrow \text{FIND}\left(\mathbf{M}^{(\ell)}_2, \mathbf{b}_{n+1|n}\right)$$

at instant n driving therefore the odds of the ℓ-th symbol $\boldsymbol{\mu}^{(\ell)}_z$ in $\boldsymbol{\Omega}_z$ being selected as the prediction $\hat{\mathbf{z}}_{n+1|n} = \boldsymbol{\mu}^{(\hat{k})}_z$, i.e. $\hat{k} = \ell$, by shaping the Dirichlet-sampled probability $\tilde{\pi}^{(\ell)}_{n+1|n}$. Then, at instant $n+1$, the VG-RAM nodes finally learn the new hyperparameters $\alpha^{(\ell')}_{n+1|n}$ and $\tilde{\alpha}^{(\ell)}_{n+1|n}$ by storing new input-output association pairs

$$\mathbf{M}^{(\ell')}_1 \leftarrow \text{INIT}\left(\mathbf{M}^{(\ell')}_1, \check{\mathbf{z}}_n \oplus \mathbf{b}_{n|n-1}, \alpha^{(\ell')}_{n+1|n}\right)$$

$$\mathbf{M}^{(\ell)}_2 \leftarrow \text{INIT}\left(\mathbf{M}^{(\ell)}_2, \mathbf{b}_{n+1|n}, \tilde{\alpha}^{(\ell)}_{n+1|n}\right)$$

or by updating the existing ones as

$$\mathbf{M}^{(\ell')}_1 \leftarrow \text{UPDT}\left(\mathbf{M}^{(\ell')}_1, \check{\mathbf{z}}_n \oplus \mathbf{b}_{n|n-1}, \alpha^{(\ell')}_{n|n-1}, \alpha^{(\ell')}_{n+1|n}\right)$$

$$\mathbf{M}^{(\ell)}_2 \leftarrow \text{UPDT}\left(\mathbf{M}^{(\ell)}_2, \mathbf{b}_{n+1|n}, \tilde{\alpha}^{(\ell)}_{n|n-1}, \tilde{\alpha}^{(\ell)}_{n+1|n}\right).$$

Intuitively, the ℓ'-th VG-RAM node in layer $\mathcal{L}_1$ learns – in a frequentist fashion – how likely is to activate the corresponding p_s bits $b^{(\ell')}_{n+1|n}, \ldots, b^{(\ell'+p_s)}_{n+1|n}$ along the layer's output belief $\mathbf{b}_{n+1|n}$ when exposed to a given input pattern $\check{\mathbf{z}}_n \oplus \mathbf{b}_{n|n-1}$ such that the likelihood of activating those bits increases as the corresponding input pattern (or a similar one in terms of the Hamming distance) is repeatedly presented to the layer's input. The VG-RAM layer $\mathcal{L}_1$ thus learns how to predict its output belief given the current input pattern. Note that, as we are not estimating the hidden state $\mathbf{s}_{n+1}$, there is no need to explicitly build the set $\boldsymbol{\Omega}_s$. As shown in Appendix B, we just need the symbol indexes to update the hyperparameters $\boldsymbol{\alpha}_{n|n-1}$. The set $\boldsymbol{\Omega}_z$, in turn, is required by both the WNN layer $\mathcal{L}_2$ to make the prediction $\hat{\mathbf{z}}_{n+1|n}$ in Step 6 and the codec scheme to encode the observation $\check{\mathbf{y}}_{n+1}$ as in (11) and decode the prediction $\hat{\mathbf{z}}_{n+1|n}$ as in (12).

5. APPLICATION EXAMPLE

In this section, we illustrate the use of the proposed nP-GbF and nP-MbL procedures in an anomaly detection (AD) application. In particular, we build an anomaly detector to handle streaming, real-time data and assess its performance using four synthetic toy time-series, specifically designed to evaluate AD algorithms.

Algorithm 3 shows how we can employ Algorithms 1 and 2 to build a non-parametric, CL anomaly detection (nP-CLAD) procedure. In this procedure, we compute first the anomaly score Σ^{ad}_{n+1} as the complement of the probability $\tilde{\pi}^{(k)}_{n+1|n}$ in $\tilde{\boldsymbol{\pi}}_{n+1|n}$ corresponding to the encoded observation $\check{\mathbf{z}}_{n+1} = \boldsymbol{\mu}^{(k)}_z$ at instant $n+1$. Then, we verify if the corresponding anomaly score Σ^{ad}_{n+1} exceeds a threshold τ. Note therefore that this test takes into account the predicted probability $\tilde{\pi}^{(k)}_{n+1|n}$ of the observed symbol $\check{\mathbf{z}}_{n+1} = \boldsymbol{\mu}^{(k)}_z$ in $\tilde{\boldsymbol{\pi}}_{n+1|n}$. Alternatively, we can modify this criterion to check if the encoded observation $\check{\mathbf{z}}_{n+1}$ mismatches the predicted observation $\hat{\mathbf{z}}_{n+1|n}$ – symbol in $\boldsymbol{\Omega}_z$ with the highest assigned probability in the predicted (sampled) posterior $\tilde{\boldsymbol{\pi}}_{n+1|n}$ – by a maximum number of bits δ, i.e. $d_H(\check{\mathbf{z}}_{n+1}, \hat{\mathbf{z}}_{n+1|n}) \geq \delta$.

For convenience, we summarize the hyperparameters employed by the sparse codec, the WNN layers – $\mathcal{L}_1$ and $\mathcal{L}_2$ – and the nP-CLAD algorithm in Table 1.

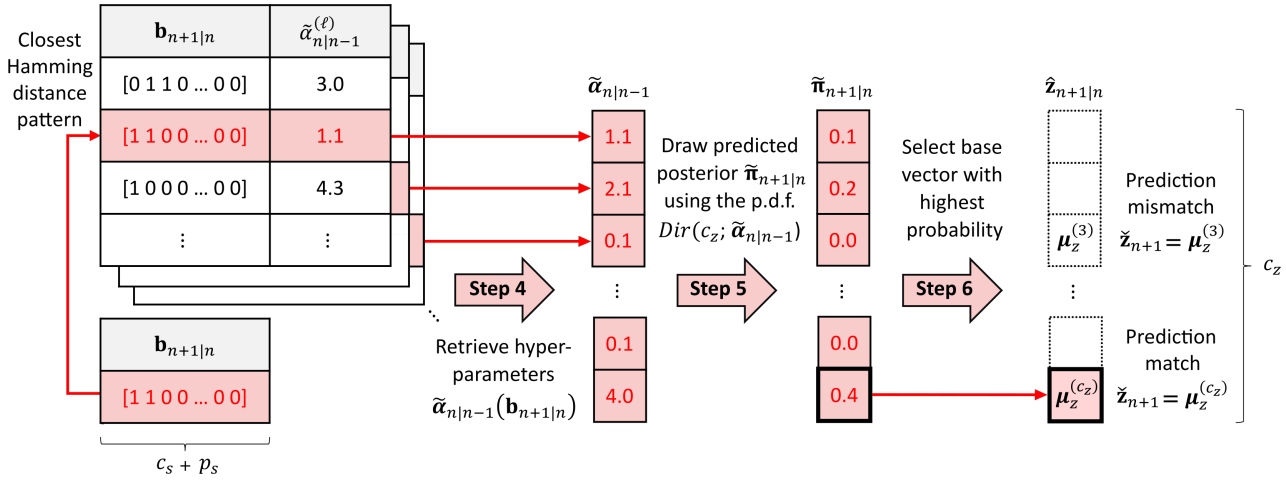

Figure 5: WNN layer $\mathcal{L}_2$: next observation belief prediction & matching.

Algorithm 3 nP-CLAD$\left(\check{\mathbf{y}}_{n+1}, \check{\mathbf{z}}_n, \mathbf{b}_{n|n-1}, \mathcal{M}\right)$

1: Encode observation $\check{\mathbf{y}}_{n+1}$ as $\check{\mathbf{z}}_{n+1} = \boldsymbol{\mu}_z^{(k)}$ using (11).
2: Build the hidden state $\check{\mathbf{s}}_{n+1} = \boldsymbol{\mu}_s^{(j)}$ from $\check{\mathbf{z}}_{n+2-\xi:n+1}$.
3: Predict the observation $\mathbf{z}_{n+1}$ using Algorithm 1

$$\left(\hat{\mathbf{z}}_{n+1|n}, \hat{k}, \mathbf{b}_{n+1|n}, \tilde{\boldsymbol{\pi}}_{n+1|n}\right) \leftarrow \text{nP-GbF}\left(\check{\mathbf{z}}_n, \mathbf{b}_{n|n-1}, \mathcal{M}\right).$$

4: Compute the anomaly score as $\Sigma^{ad}_{n+1} \leftarrow 1 - \tilde{\pi}^{(k)}_{n+1|n}$.
5: Detect the anomaly by checking $\Lambda^{ad}_{n+1} \leftarrow [\![\Sigma^{ad}_{n+1} > \tau]\!]$.
6: Learn the non-parametric model using Algorithm 2

$$\mathcal{M} \leftarrow \text{nP-MbL}\left(j, k, \eta, \check{\mathbf{z}}_n, \mathbf{b}_{n|n-1}, \mathbf{b}_{n+1|n}, \mathcal{M}\right).$$

7: **return** $\left(\Lambda^{ad}_{n+1}, \Sigma^{ad}_{n+1}, \check{\mathbf{z}}_{n+1}, \mathbf{b}_{n+1|n}, \mathcal{M}\right)$

Table 1: Summary of the WNN model hyperparameters.

Codec: map observations $\check{\mathbf{y}}_n \in \boldsymbol{\Omega}_y$ into symbols $\check{\mathbf{z}}_n \in \boldsymbol{\Omega}_z$	
a_z	Maximum number of activated bits in $\check{\mathbf{z}}_n$
b_z	Minimum Hamming distance of symbols in $\boldsymbol{\Omega}_z$
d_z	Total number of bits of $\check{\mathbf{z}}_n$
Layer $\mathcal{L}_1$: learn how to predict the posterior $\boldsymbol{\pi}_{n+1\|n}$	
c_s	Number of symbols in $\boldsymbol{\Omega}_s$
$\forall \ell', \alpha_{\ell'}$	Initial Dirichlet $Dir(c_s; \boldsymbol{\alpha}_{0\|-1})$ pseudo-counts
p_s	Number of activated bits in the belief $\boldsymbol{b}_{n\|n-1}$
Layer $\mathcal{L}_2$: learn how to predict the posterior $\tilde{\boldsymbol{\pi}}_{n+1\|n}$	
c_z	Number of symbols in $\boldsymbol{\Omega}_z$
$\forall \ell, \tilde{\alpha}_\ell$	Initial Dirichlet $Dir(c_z; \tilde{\boldsymbol{\alpha}}_{0\|-1})$ pseudo-counts
nP-CLAD: check $\check{\mathbf{y}}_{n+1}$ using its anomaly score Σ^{ad}_{n+1}	
ξ	Length of the sub-sequence $\check{\mathbf{z}}_{n+2-\xi:n+1}$
τ	Anomaly detection threshold
η	Learning rate employed by the WNN model

We initialize $\mathbf{b}_{0|-1}$ as per nP-GbF filter Steps 2 and 3 by making $\boldsymbol{\alpha}_{0|-1} = \boldsymbol{\alpha}_0$ in Algorithm 1. We encode the streamed observations $\check{\mathbf{y}}_{0:\infty}$ in Step 1 using a fixed-range $(0 - 100)$ numeric encoder with 961 buckets – unique symbols in $\boldsymbol{\Omega}_z$ – and $d_z = 2^{10}$ bits to encode the time-series values as indicated in [29]. We also initialize $\check{\mathbf{z}}_{1-\xi:-1}$ by replicating $\check{\mathbf{z}}_0$ to be able to build $\check{\mathbf{s}}_n$, $\forall n \geq 0$, as required in Step 2 of Algorithm 3. There is no need to decode network predictions in this AD application.

Table 2 summarizes the hyperparameters employed to setup the WNN model and run all experiments. We should highlight that only 11 hyperparameters are required to initialize the proposed RAM-based model plus the nP-CLAD algorithm. The hyperparameter d_s is not required, since there is no need to build the set $\boldsymbol{\Omega}_s$.

Table 2: Hyperparameters employed by the codec, the WNN layers, and the nP-CLAD algorithm throughout experiments.

Codec		**Layer $\mathcal{L}_1$**		**Layer $\mathcal{L}_2$**		**nP-CLAD**	
a_z	2^6	c_s	2^{10}	c_z	2^{10}	ξ	2^3
b_z	2^2	$\forall \ell', \alpha^{\ell'}$	2^{-12}	$\forall \ell, \tilde{\alpha}^{\ell}$	2^{-12}	τ	0.75
d_z	2^{10}	p_s	2^3	–	–	η	2^1

Figures 6 through 9 show the AD results considering four artificial time-series with randomly anomalous spikes. Despite being quite simple, these AD experiments illustrate the filter ability to learn increasingly complex sequences online and detect point anomalies. We highlighted the true anomalies – intervals shaded in red – and indicated the detected anomalies – red circles – in the last 500 time instants (green area), since the filter was still learning the time-series in the first half of the simulation (gray area). The nP-CLAD was able to successfully detect all random spikes along the evaluated time-series using the anomaly scores provided by the non-parametric model. Nevertheless, it is worth noting that the filter still needs to be validated in real-world settings using e.g. Numenta Anomaly Benchmark (NAB) [1].

The experiments were conducted in an Intel(R) Core(TM) i7-9750H central processing unit (CPU) @ 2.60GHz with 32GB of RAM. Note that the inference time depends on the number of entries stored in the VGRAM nodes' memory, since the nodes memory is virtual and its size increases as the WNN model learns a new sequence.

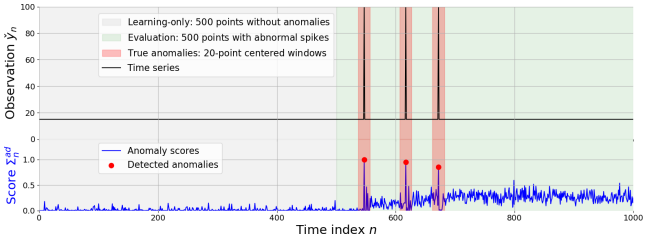

Figure 6: Constant time-series with anomalous spikes.

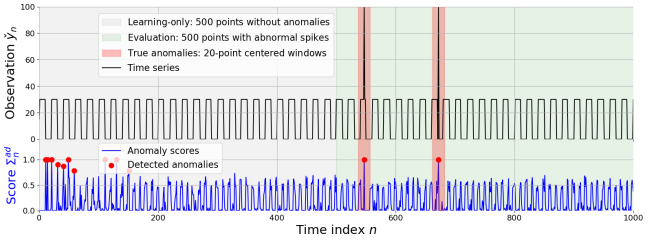

Figure 7: Squared time-series with anomalous spikes.

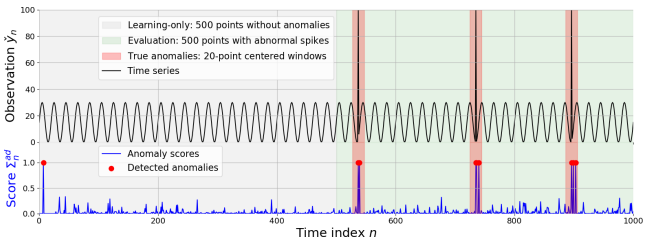

Figure 8: Sinusoidal time-series with anomalous spikes.

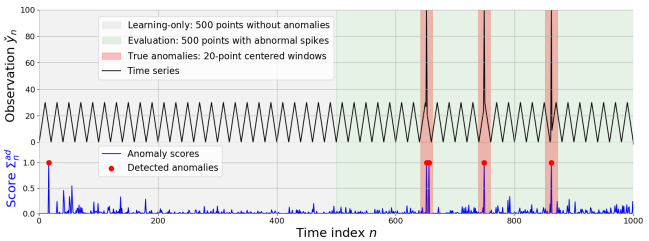

Figure 9: Sawtooth time-series with anomalous spikes.

Table 3 indicates the mean and maximum inference times in milliseconds – considering one iteration of Algorithm 3 –, and the accumulated memory size in kilobytes at the end of each experiment. As expected, the accumulated memory size increases according to the complexity of the learned time series. Finally, a non-optimized Python implementation of the proposed algorithms and reproducible experiments can be found in [15].

Table 3: Computational performance considering different time series. Inference times are in milliseconds (ms), whereas accumulated memory sizes are in kilobytes (kB).

Time series	Inference time		Accumulated memory size
	Average	Maximum	
Constant	10.74 ms	24.82 ms	58.45 kB
Squared	11.22 ms	32.03 ms	151.17 kB
Sinusoidal	11.72 ms	31.93 ms	168.30 kB
Sawtooth	11.50 ms	35.62 ms	178.40 kB

6. CONCLUSIONS

6.1 Summary of paper contributions

We presented in this paper a fully non-parametric, grid-based filter designed to intrinsically learn an interpretable probabilistic model of the underlying Markovian process (see Fig. 1) emitting the incoming sequence of observations, which allows one to make explainable and auditable predictions – relying on clear input-to-output causal chains connected to traceable memory entries – with promising, but still limited results as shown in toy AD examples. Moreover, the proposed RAM-based filter implementation performs one-shot, CL without requiring any special hardware to accomplish it. Indeed, VG-RAM nodes can be efficiently implemented through software – using e.g. C++ to really tame the machine – conceived to run on general-purpose, CPUs using ordinary, abundant RAM hardware, which is orders of magnitude cheaper than state-of-the-art graphics processing units (GPUs) specially designed for machine learning tasks.

6.2 Future research directions

As future research, we propose to adapt the sparse auto-encoder introduced in [26] to learn (offline) a high-dimensional binary representation of the continuous-valued observations in an embedded space. We also intend to employ mismatched predictions as learning signals as proposed by Cui et al. in [11]. Furthermore, we can limit the memory capacity of the VG-RAM nodes and individually force them to eventually replace associative input-output pairs with small hyperparameters by new ones, limiting thus the required network resources by forcing its nodes to gracefully forget in the long run input patterns which are not frequently seen, i.e. presented as input to their corresponding layers. Lastly, we may use sparse binary word vectors as proposed by Faruqui et al. in [16] to encode natural language data and check the suitability of the proposed filter to build a tiny language model which is able to continuously learn from new corpora.

7. ADDITIONAL AUTHORS

Additional authors: Thiago Oliveira-Santos and Claudine Badue (Applied Computational Intelligence Institute, UFES `[todsantos, claudine]@inf.ufes.br`).

8. REFERENCES

[1] S. Ahmad, A. Lavin, S. Purdy, and Z. Agha. Unsupervised real-time anomaly detection for streaming data. *Neurocomputing*, 262:134–147, 2017.

[2] I. Aleksander. From WISARD to MAGNUS: A family of weightless virtual neural machines. *RAM-Based Neural Networks*, pages 18–30, 1998.

[3] S. Barra, S. M. Carta, A. Corriga, A. S. Podda, and D. R. Recupero. Deep learning and time series-to-image encoding for financial forecasting. *IEEE/CAA Journal of Automatica Sinica*, 7(3):683–692, 2020.

[4] L. E. Baum, T. Petrie, G. Soules, and N. Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *The Annals of Mathematical Statistics*, 41(1):164–171, 1970.

[5] L. Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010: 19th International Conference on Computational Statistics*, pages 177–186. Springer, 2010. Paris France, August 22-27, 2010 Keynote, Invited and Contributed Papers.

[6] L. Bottou. Stochastic gradient descent tricks. In *Neural Networks: Tricks of the Trade: Second Edition*, pages 421–436. Springer, 2012.

[7] A. Canale and M. Ruggiero. Bayesian nonparametric forecasting of monotonic functional time series. *Electronic Journal of Statistics*, 10(2):3265–3286, 2016.

[8] Y. Chen, S. Liu, Z. Wang, D. Wu, Y. Li, B. Xing, B. Guo, and L. Chen. Online parallel attack detection method for industrial control based on multi-bandpass filter. *IEEE Internet of Things Journal*, 11(1):880–888, 2024.

[9] C. Cheng, A. Sa-Ngasoongsong, O. Beyca, T. Le, H. Yang, Z. Kong, and S. T. Bukkapatnam. Time series forecasting for nonlinear and non-stationary processes: a review and comparative study. *IIE Transactions*, 47(10):1053–1071, 2015.

[10] V. K. R. Chimmula and L. Zhang. Time series forecasting of COVID-19 transmission in canada using lstm networks. *Chaos, Solitons & Fractals*, 135:109864, 2020.

[11] Y. Cui, S. Ahmad, and J. Hawkins. Continuous online sequence learning with an unsupervised neural network model. *Neural Computation*, 28(11):2474–2504, 2016.

[12] M. De Gregorio and M. Giordano. An experimental evaluation of weightless neural networks for multi-class classification. *Applied Soft Computing*, 72:338–354, 2018.

[13] A. F. De Souza, C. Badue, F. Pedroni, E. Oliveira, S. S. Dias, H. Oliveira, and S. F. de Souza. Face recognition with VG-RAM weightless neural networks. In *Artificial Neural Networks-ICANN 2008: 18th International Conference, Prague, Czech Republic, September 3-6, 2008, Proceedings, Part I 18*, pages 951–960. Springer, 2008.

[14] A. F. De Souza, F. Pedroni, E. Oliveira, P. M. Ciarelli, W. F. Henrique, L. Veronese, and C. Badue. Automated multi-label text categorization with VG-RAM weightless neural networks. *Neurocomputing*, 72(10–12):2209–2217, 2009.

[15] S. S. Dias. Weightless neural network library (wnnlib), 2025. Available from https://github.com/stivenschwanz/wnnlib/tree/safeai.

[16] M. Faruqui, Y. Tsvetkov, D. Yogatama, C. Dyer, and N. A. Smith. Sparse overcomplete word vector representations. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1491–1500, 2015.

[17] Z. Ghahramani. Bayesian non-parametrics and the probabilistic approach to modelling. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 371(1984):20110553, 2013.

[18] J. Hawkins and S. Ahmad. Why neurons have thousands of synapses, a theory of sequence memory in neocortex. *Frontiers in Neural Circuits*, 10(23):1–13, 2016.

[19] J. Hawkins, S. Ahmad, D. Dubinsky, et al. Cortical learning algorithm and hierarchical temporal memory. *Numenta Whitepaper*, pages 1–68, 2011.

[20] A. Iyer, K. Grewal, A. Velu, L. O. Souza, J. Forest, and S. Ahmad. Avoiding catastrophe: Active dendrites enable multi-task learning in dynamic environments. *Frontiers in Neurorobotics*, 16(846219):1–23, 2022.

[21] S. M. Kay. *Fundamentals of Statistical Signal Processing: Estimation Theory*. Prentice Hall PTR, 1993.

[22] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017.

[23] B. Lim and S. Zohren. Time-series forecasting with deep learning: a survey. *Philosophical Transactions of the Royal Society A*, 379(2194):20200209, 2021.

[24] T. B. Ludermir, A. C. P. d. L. F. Carvalho, A. P. Braga, and M. C. De Souto. Weightless neural models: a review of current and past works. *Neural Computing Surveys*, 2:41–61, 1999.

[25] G. Mahalakshmi, S. Sridevi, and S. Rajaram. A survey on forecasting of time series data. In *2016 International Conference on Computing Technologies and Intelligent Data Engineering (ICCTIDE'16)*, pages 1–8. IEEE, 2016.

[26] A. Makhzani and B. Frey. K-sparse autoencoders. *arXiv preprint arXiv:1312.5663*, 2013.

[27] R. J. Mitchell, J. Bishop, S. Box, and J. Hawker. Comparison of some methods for processing 'grey level' data in weightless networks. In *RAM-based Neural Networks*, pages 61–70. World Scientific, 1998.

[28] P. Müller, F. A. Quintana, A. Jara, and T. Hanson. *Bayesian nonparametric data analysis*, volume 1. Springer, 2015.

[29] S. Purdy. Encoding data for HTM systems. *arXiv preprint arXiv:1602.05925*, 2016.

[30] T. Singh, R. Kalra, S. Mishra, Satakshi, and M. Kumar. An efficient real-time stock prediction exploiting incremental learning and deep learning. *Evolving Systems*, 14(6):919–937, 2023.

[31] R. Sousa, T. Lima, A. Abelha, and J. Machado. Hierarchical temporal memory theory approach to stock market time series forecasting. *Electronics*, 10(14):1630, 2021.

[32] J. Xuan, J. Lu, and G. Zhang. A survey on bayesian nonparametric learning. *ACM Computing Surveys (CSUR)*, 52(1):1–36, 2019.

APPENDIX

A. BELIEF ENCODING

Algorithm 4 indicates how to encode the categorical distribution $\boldsymbol{\pi}$ with c categories into a sparsely encoded belief $\mathbf{b}$ in the binary space $\mathbb{Z}_2^{c+p}$ with up to p activated bits. Note that this is a lossy encoding scheme. Besides the floor operation in Line 5, which rounds off the scaled probabilities $p \cdot \pi_i$, $\forall i \in \{1, \ldots, c\}$, non null probabilities $\pi_i > 0$ and $\pi_j > 0$, $i \neq j$, in $\boldsymbol{\pi}$ with close indexes $|i - j| < p$ can generate a sequence of contiguous active bits in $\mathbf{b}$. Thus, as illustrated in the running example in Sec. 3 (inspect Fig. 2), depending on the probability values π_i and π_j, multiple valid categorical distributions can be mapped to the same belief by this algorithm.

Algorithm 4 $Bel\left(\boldsymbol{\pi} = \begin{bmatrix}\pi_1 & \ldots & \pi_c\end{bmatrix}^\top; p\right)$

1: Initialize $\mathbf{b} = \begin{bmatrix}b_1 & \ldots & b_{c+p}\end{bmatrix}^\top$ such that, $\forall i \in \{1, 2, \ldots, c+p\}$, $b_i \leftarrow 0$.
2: $acc \leftarrow 0$
3: **for** $i \leftarrow 1$ **to** $c + p$ **do**
4: **if** $i \leq c$ **then**
5: $acc \leftarrow acc + \lfloor p \cdot \pi_i + \frac{1}{2} \rfloor$
6: **end if**
7: **if** $acc > 0$ **then**
8: $b_i \leftarrow 1$
9: $acc \leftarrow acc - 1$
10: **end if**
11: **end for**
12: **return b**

Algorithm 5 shows in turn how to decode the sparsely encoded belief $\mathbf{b}$ into the posterior $\boldsymbol{\pi}$ within the $(c-1)$-dimensional simplex $\mathbb{S}_c$. It is worth noting that this algorithm will assign a non null probability π_i to the first symbol belonging to a contiguous sequence $b_i, b_{i+1}, b_{i+2}, \ldots$ of activate bits in $\mathbf{b}$. Thus, one should avoid assigning similar symbols to close locations in $\boldsymbol{\pi}$ such that it is unlikely to produce a single contiguous sequence of active bits within $\mathbf{b}$. Otherwise, the $Bel(\cdot)$ procedure in Algorithm 4 will activate overlapping sequences of contiguous bits in the binary space $\mathbb{Z}_2^{c+p}$ for those symbols, therefore yielding an ambiguous representation which cannot be properly recovered by the $Bel^{-1}(\cdot)$ procedure in Algorithm 5.

Algorithm 5 $Bel^{-1}\left(\mathbf{b} = \begin{bmatrix}b_1 & \ldots & b_{c+p}\end{bmatrix}^\top; p\right)$

1: Initialize $\boldsymbol{\pi} = \begin{bmatrix}\pi_1 & \ldots & \pi_c\end{bmatrix}^\top$ such that, $\forall i \in \{1, 2, \ldots, c\}$, $\pi_i \leftarrow 0$.
2: $acc \leftarrow 0$
3: **for** $i \leftarrow c + p$ **to** 1 **do**
4: $acc \leftarrow acc + b_i$
5: **if** $i \leq c \wedge (i = 1 \vee b_{i-1} = 0)$ **then**
6: $\pi_i \leftarrow acc$
7: $acc \leftarrow 0$
8: **end if**
9: **end for**
10: Normalize $\boldsymbol{\pi}$ such that $\sum_{i=1}^{c} \pi_i = 1$.
11: **return** $\boldsymbol{\pi}$

B. SEQUENCE LEARNING

According to assumption (30), $p(\mathbf{z}_{n+1} \mid \tilde{\boldsymbol{\pi}}_{n+1|n})$ is a categorical distribution. Thus, for a particular realization $\check{\mathbf{z}}_{n+1} = \boldsymbol{\mu}_z^{(k)}$ of $\mathbf{Z}_{n+1}$, the likelihood function becomes

$$\begin{aligned} p(\check{\mathbf{z}}_{n+1} \,|\, \tilde{\boldsymbol{\pi}}_{n+1|n}) &= p(\mathbf{z}_{n+1} \,|\, \tilde{\boldsymbol{\pi}}_{n+1|n})\big|_{\mathbf{z}_{n+1}=\check{\mathbf{z}}_{n+1}} \\ &= \sum_{\ell=1}^{c_z} \tilde{\pi}_{n+1|n}^{(\ell)} \cdot [\![\mathbf{z}_{n+1} = \boldsymbol{\mu}_z^{(\ell)}]\!]\Big|_{\mathbf{z}_{n+1}=\boldsymbol{\mu}_z^{(k)}} \\ &= \tilde{\pi}_{n+1|n}^{(k)}. \end{aligned} \tag{33}$$

From Eq. (33) and assumption (25), it follows that we can write the updated posterior $p(\tilde{\boldsymbol{\pi}}_{n+1|n} \,|\, \check{\mathbf{z}}_{0:n+1})$ using the Bayes rule as

$$\begin{aligned} p(\tilde{\boldsymbol{\pi}}_{n+1|n} \,|\, \check{\mathbf{z}}_{0:n+1}) &\propto p(\check{\mathbf{z}}_{n+1} \,|\, \tilde{\boldsymbol{\pi}}_{n+1|n})\, p(\tilde{\boldsymbol{\pi}}_{n+1|n} \,|\, \check{\mathbf{z}}_{0:n}) \\ &= \frac{\tilde{\pi}_{n+1|n}^{(k)}}{B(\tilde{\boldsymbol{\alpha}}_{n|n-1})} \prod_{\ell=1}^{c_z} \left(\tilde{\pi}_{n+1|n}^{(\ell)}\right)^{\tilde{\alpha}_{n|n-1}^{(\ell)}-1} \\ &\propto \tilde{\pi}_{n+1|n}^{(k)} \prod_{\ell=1}^{c_z} \left(\tilde{\pi}_{n+1|n}^{(\ell)}\right)^{\tilde{\alpha}_{n|n-1}^{(\ell)}-1}, \end{aligned} \tag{34}$$

since the Beta function $B(\tilde{\boldsymbol{\alpha}}_{n|n-1})$ is constant for a given $\tilde{\boldsymbol{\alpha}}_{n|n-1}$. We can rewrite thus the right-hand side of (34) as

$$\left(\tilde{\pi}_{n+1|n}^{(k)}\right)^{\left(\tilde{\alpha}_{n|n-1}^{(k)}+1\right)-1} \prod_{\ell=1,\ell\neq k}^{c_z} \left(\tilde{\pi}_{n+1|n}^{(\ell)}\right)^{\tilde{\alpha}_{n|n-1}^{(\ell)}-1}. \tag{35}$$

Thus, given $\check{\mathbf{z}}_{n+1} = \boldsymbol{\mu}_z^{(k)}$, the updated posterior can be rewritten as

$$p(\tilde{\boldsymbol{\pi}}_{n+1|n} \,|\, \check{\mathbf{z}}_{0:n+1}) = \frac{1}{B(\tilde{\boldsymbol{\alpha}}_{n+1|n})} \prod_{\ell=1}^{c_z} \left(\tilde{\pi}_{n+1|n}^{(\ell)}\right)^{\tilde{\alpha}_{n+1|n}^{(\ell)}-1} \tag{36}$$

with new normalization constant given by the reciprocal of $B(\tilde{\boldsymbol{\alpha}}_{n+1|n})$ such that, conditioned on both $\check{\mathbf{z}}_{n+1} = \boldsymbol{\mu}_z^{(k)}$ and $\check{\mathbf{z}}_{0:n}$, $\tilde{\boldsymbol{\Pi}}_{n+1|n}$ is also Dirichlet

$$\tilde{\boldsymbol{\Pi}}_{n+1|n} \,|\, \check{\mathbf{z}}_{n+1} = \boldsymbol{\mu}_z^{(k)}, \check{\mathbf{z}}_{0:n} \sim Dir(c_z; \tilde{\boldsymbol{\alpha}}_{n+1|n})$$

with new hyperparameters $\tilde{\boldsymbol{\alpha}}_{n+1|n} = \tilde{\boldsymbol{\alpha}}_{n|n-1} + \mathbf{1}_k$.
Using a similar approach, given assumptions (23) and (27), we can show that, conditioned on a particular realization $\check{\mathbf{s}}_{n+1} = \boldsymbol{\mu}_s^{(j)}$ of $\mathbf{S}_{n+1}$, the Bayes rule application

$$\begin{aligned} p(\boldsymbol{\pi}_{n+1|n} \,|\, \check{\mathbf{s}}_{0:n+1}) &\propto p(\check{\mathbf{s}}_{n+1} \,|\, \boldsymbol{\pi}_{n+1|n})\, p(\boldsymbol{\pi}_{n+1|n} \,|\, \check{\mathbf{s}}_{0:n}) \\ &= \frac{\pi_{n+1|n}^{(j)}}{B(\boldsymbol{\alpha}_{n|n-1})} \prod_{\ell'=1}^{c_s} \left(\pi_{n+1|n}^{(\ell')}\right)^{\alpha_{n|n-1}^{(\ell')}-1} \\ &\propto \pi_{n+1|n}^{(j)} \prod_{\ell'=1}^{c_s} \left(\pi_{n+1|n}^{(\ell')}\right)^{\alpha_{n|n-1}^{(\ell')}-1} \end{aligned} \tag{37}$$

yields an updated posterior

$$p(\boldsymbol{\pi}_{n+1|n} \,|\, \check{\mathbf{s}}_{0:n+1}) = \frac{1}{B(\boldsymbol{\alpha}_{n+1|n})} \prod_{\ell'=1}^{c_s} \left(\pi_{n+1|n}^{(\ell')}\right)^{\tilde{\alpha}_{n+1|n}^{(\ell')}-1} \tag{38}$$

with new normalization constant given now by the reciprocal of $B(\boldsymbol{\alpha}_{n+1|n})$, which is also Dirichlet

$$\boldsymbol{\Pi}_{n+1|n} \,|\, \check{\mathbf{s}}_{n+1} = \boldsymbol{\mu}_s^{(j)}, \check{\mathbf{s}}_{0:n} \sim Dir(c_s; \boldsymbol{\alpha}_{n+1|n})$$

with new hyperparameters $\boldsymbol{\alpha}_{n+1|n} = \boldsymbol{\alpha}_{n|n-1} + \mathbf{1}_j$.